

Doctoral Thesis

PhD Program on Information and Communication Technologies

e-MoDe: A Framework to support the development of mobile telemonitoring platforms



UGR

Universidad
de **Granada**

Departamento de Lenguajes y Sistemas Informáticos

Author:

Ángel Ruiz Zafra

Supervisors:

Manuel Noguera García

Kawtar Benghazi Akhlaki

2016

Editor: Universidad de Granada. Tesis Doctorales
Autor: Ángel Ruiz Zafra
ISBN: 978-84-9163-003-6
URI: <http://hdl.handle.net/10481/44274>

El doctorando Ángel Ruiz Zafra y los directores de la tesis Manuel Noguera García y Kawtar Benghazi Akhlaki garantizamos, al firmar esta tesis doctoral, que el trabajo ha sido realizado por el doctorando bajo la dirección de los directores de la tesis y hasta donde nuestro conocimiento alcanza, en la realización del trabajo, se han respetado los derechos de otros autores a ser citados, cuando se han utilizado sus resultados o publicaciones.

Granada, 24 de mayo de 2016

Director/es de la Tesis

Doctorando

Fdo.:

Fdo.:

The PhD candidate Ángel Ruiz Zafra and the thesis supervisors Manuel Noguera García and Kawtar Benghazi Akhalki guarantee, by signing this thesis, that the work has been done by the PhD candidate under the guidance of the directors of the thesis and, as far as our knowledge reaches, in the realization of the work, the copyrights of the cited authors have been respected, when their results or publications have been used.

Granada, May 24, 2016

Supervisors

PhD Candidate

Sign.:

Sign.:

Resumen

Las plataformas de telemonitorización, o seguimiento remoto de la actividad, están cada vez más presentes en numerosos ámbitos. Estas plataformas ofrecen recursos y funcionalidades, a través del uso de las TICs, para percibir y registrar qué ocurre o cómo se comporta un usuario sin necesidad de encontrarnos en su misma ubicación.

En el ámbito de salud y bienestar las plataformas de telemonitorización están teniendo cada vez más aceptación. Debido al incremento de la esperanza de vida, y el consecuente incremento del gasto sanitario, las plataformas de telemonitorización se posicionan como una alternativa a enfoques tradicionales para reducir significativamente estos costes, gracias a las funcionalidades (soportadas por las TICs) que ofrecen.

Aunque el abanico de tecnologías disponibles es amplio, son tres las tecnologías clave que soportan la mayoría de las soluciones de telemonitorización actuales: (1) la computación en la nube, (2) la computación móvil y la (3) computación *wearable*. El uso combinado de estas tecnologías otorga a las plataformas de telemonitorización características o propiedades como, deslocalización geográfica, acceso en tiempo real a la información y recogida de información automatizada.

El desarrollo de una plataforma de telemonitorización lleva implícito dar soporte a objetivos inherentes al hecho de monitorizar remotamente una actividad (reducir desplazamientos, reducción de costes, ubicuidad, etc.) y otros propios del ámbito u objeto para el que se diseña cada plataforma (telerehabilitación de física de pacientes, seguimiento de algún tipo de parámetro fisiológico, control de corredores, vigilancia de niños, etc.). Estos objetivos se especifican durante el proceso de desarrollo o vienen previamente definidos en el llamado *protocolo de telemonitorización*, esto es, los pasos o etapas para llevar a cabo la supervisión de un usuario, objetivos, duración, etc.

Cuando un desarrollo parte de un protocolo de telemonitorización, la plataforma resultante es una herramienta de soporte al protocolo, por lo que un requisito es que su diseño permita que sea adaptable a los posibles cambios en dicho protocolo a lo largo del tiempo.

Además de esta adaptación al protocolo, la adaptación al usuario monitorizado es otro requisito que condiciona el desarrollo de una plataforma de telemonitorización. El usuario monitorizado es el actor principal en una plataforma de telemonitorización, y puesto que la mayoría de las soluciones de telemonitorización pertenecen al ámbito de salud y bienestar, el usuario monitorizado suele presentar necesidades especiales, como problemas de salud, deficiencias cognitivas o, simplemente, falta de cultura tecnológica.

Aunque en el desarrollo de una plataforma de telemonitorización se intenta garantizar la usabilidad y funcionalidad de las aplicaciones, este requisito no siempre se considera al mismo nivel de importancia que otros (por ejemplo, tecnológicos), pasando a segundo plano las necesidades del usuario monitorizado. Para evitar esto, y garantizar que dichas necesidades especiales de los usuarios monitorizados sean consideradas, lo idóneo es abordar el proceso de desarrollo considerando las necesidades del usuario una prioridad.

El desarrollo de una plataforma de telemonitorización se lleva a cabo de forma colaborativa, normalmente, entre expertos en el ámbito de la telemonitorización e ingenieros de software. El soporte al protocolo de telemonitorización, la importancia que se le presta a las necesidades del usuario monitorizado y el hecho de que cada actor participante en el desarrollo tiene una visión particular de la plataforma y unos objetivos concretos, ocasionan que cada proceso de desarrollo sea distinto. Además de estos factores, existen otros que condicionan el desarrollo y están relacionados con la tecnología.

El uso de la tecnología implica encontrar soluciones para dar soporte a funcionalidades que normalmente están presentes en cualquier plataforma de telemonitorización: soporte colaborativo, integración de *wearables*, añadir una nueva tarea de telemonitorización para adaptarse a un determinado protocolo, etc.

Muchas de estas funcionalidades presentan retos a nivel de diseño software, esto es, diseñar una solución que permita dar soporte a la funcionalidad. No obstante, otras funcionalidades están directamente relacionadas con la dificultad que conlleva implementar una solución, como es el caso de la integración de *wearables*.

Debido a que hay numerosos *wearables* en el mercado, y que cada uno (1) está soportado por una tecnología específica, (2) proporciona una información distinta y (3) el acceso a esta información puede hacerse también de manera distinta, el proceso de integración e interacción de un *wearable* por parte de un programador es complejo y tedioso, siendo necesarios conocimientos específicos para poder integrarlo en una plataforma de telemonitorización.

Dichos retos mencionados, entre otros comúnmente presentes en el proceso de desarrollo de una plataforma de telemonitorización, ocasionan que dicho proceso sea un proceso complejo.

En esta tesis se presenta e-MoDe, un framework que pretende sistematizar y simplificar el desarrollo de una plataforma de telemonitorización, a través de tres elementos: (1) una metodología para guiar el proceso de desarrollo; (2) un conjunto de modelos como solución a retos presentes en la etapa de diseño software (modelo de arquitectura, modelo para la gestión de tareas de telemonitorización y modelo para el soporte colaborativo) y (3) un conjunto de herramientas para simplificar la implementación del sistema (herramienta *WearIt* para la integración de *wearables* y *Zappa* para simplificar el desarrollo de aplicaciones móviles).

e-MoDe se ha aplicado en dos plataformas de telemonitorización: CloudRehab, plataforma de telerehabilitación para pacientes con daño cerebral, y ClouFit, plataforma de telemonitorización para la gestión integral de entrenamientos deportivos.

Abstract

Telemonitoring platforms, or remote monitoring of activity, are increasingly present in many areas. These platforms provide resources and functionalities, through the use of ICTs, to receive and record what happens or how a user behaves without being in the same location.

In the healthcare and wellness area the telemonitoring platforms are having more acceptance. Because the increase of life expectancy, and the consequent increase of healthcare cost, telemonitoring platforms are positioned as an alternative to traditional approaches, in order to significantly reduce costs, thanks to the functionalities (supported by the ICTs) provided.

Although there is a wide the range of available technologies, there are three key technologies that support the most of the currently telemonitoring solutions: (1) cloud computing, (2) mobile computing and (3) wearable computing. The integrated use these technologies and its use in telemonitoring platforms provides a number of advantages or features, like geographical relocation, access to information in real time, automation information gathering, etc.

The development of a telemonitoring platform implicitly provides supports to telemonitoring goals (reduce displacements, cost reduction, ubiquity, etc.) and others related with the area in which each platform is designed (physical telerehabilitation of patients, monitoring of some physiological parameter, monitoring children, etc.). These goals are specified during the development by the experts in the area or are previously specified in the so-called telemonitoring protocol or methodology, that is, the steps or stages to carry out the supervision of a monitored user, the goals, duration, etc.

When the development of a telemonitoring platform is supported by a telemonitoring protocol, the platform obtained as result of the development process is a tool to support the protocol, so should be adaptable to changes in the protocol, thus having a long service life.

In addition of this support to the protocol, the adaptation of the monitored user is another requirement that conditions the development of a telemonitoring platform. The monitored user is the main actor of the platform, and because most of solutions belong to healthcare and wellness area, the monitored user usually has special needs, health problems, some chronic disease or cognitive problems or, simply, lack of technological culture.

Although during the development process of a telemonitoring platform the usability and functionality of the different applications is addressed, this requirement is not at the same level as other (e.g. technological), because the monitored user needs are not a main requirement. To avoid this situation, is appropriate to consider as priority the monitored user needs during the development process

The development of a telemonitoring platform is carried out in a collaborative way, usually, between software engineers and domain experts. The adaptation of the telemonitoring protocol, the importance that is given to the user needs and the interaction between the different actors to carry out the development cause that each development process is

different and addressed in a custom way. Besides these factors, there are others related with the use of technology that influence the development of a telemonitoring platform.

Use of technology involves finding solutions to provide support to functionalities that are commonly present in most telemonitoring platforms: support to the collaboration between supervisors, wearables integration, add new telemonitoring task to adapt to changes in the protocol, etc.

Support some of these functionalities are related with the ability to design suitable solutions, that is, design the solution to support the functionality. However, other functionalities are directly related with the implementation process, like the wearables integration process.

Because of a wide range of wearables in the market, and each one (1) is supported by a custom technology, (2) provides different information and (3) the access to this information can also be done differently, the integration process and the interaction with a wearable by programmers is complex and tedious, being necessary specific knowledge to integrate it in a telemonitoring platform.

These challenges, along with others related with the development process (how to approach and address the process, define the goals, etc.), cause that the development of a telemonitoring platform is complex process.

In this thesis is presented e-MoDe, a framework that aims to systematize and simplify the development of a telemonitoring platform, thanks to three elements: (1) a methodology to guide the development process; (2) a set of model as solution to some challenges presented in the software design stage (architecture model, task management model and collaborative support model) and (3) a set of tool to simplify the implementation of the system (WearIt tool for the integration of wearables and Zappa, a platform to simplify the development of mobile applications).

e-MoDe has been applied in two telemonitoring platforms: CloudRehab, a telerehabilitation platform for brain-injured patients, and CloudFit, a telemonitoring platform for the management of sport trainings.

Índice General

Resumen	9
Abstract	11
Índice General	13
Índice de Figuras	17
Índice de Tablas	21
Prefacio	23
1.1. Introducción	25
1.2. Motivación	26
1.3. Planteamiento del problema	30
1.4. Objetivos de la tesis	31
1.5. Estructura de la tesis	31
Preface	33
1.1. Introducción	35
1.2. Motivation	36
1.3. Problem Statement	39
1.4. Goals	40
1.5. Thesis Structure	40
Plataformas de Telemonitorización	43
2.1. Introducción	45
2.2. Contexto de aplicación: casos de uso	46
2.2.1. Telemonitorización para el bienestar de personas mayores	47
2.2.2. Telemonitorización de dolencias crónicas.....	49
2.2.3. Telemonitorización para la rehabilitación.....	52
2.2.4. Fitness.....	54
2.3. Arquitectura Lógica de las plataformas de telemonitorización	57
2.4. Desarrollo de plataformas de telemonitorización	59
2.5. Objetivos de las plataformas de telemonitorización	61
2.6. Conclusiones	64
Antecedentes	65
3.1. Introducción	67
3.2. Computación en la nube	68
3.2.1. Introducción.....	68
3.2.2. Arquitectura Cloud	69
3.2.3. Características.....	71
3.2.4. Cloud Computing en plataformas telemonitorización.....	73
3.3. Computación <i>wearable</i>	74
3.3.1. Introducción.....	74
3.3.2. Atributos y características	74
3.3.3. Redes Inalámbricas de Área Personal (<i>WBAN</i>).....	75
3.3.4. <i>Wearable Computing</i> en plataformas telemonitorización	77
3.4. Computación móvil	78
3.4.1. Introducción.....	78
3.4.2. Características.....	78

3.4.3. Computación móvil en plataformas telemonitorización	79
3.5. Arquitecturas Software	81
3.5.1. Introducción.....	81
3.5.2. Características.....	81
3.5.3. Arquitecturas Orientadas a Servicios (SOA)	82
3.5.4. Arquitecturas Orientadas a Recursos (ROA).....	84
3.5.5. Arquitectura Software en plataformas de telemonitorización.....	86
3.6. Desarrollo dirigido por modelos (<i>Model-Driven Development - MDD</i>).....	88
3.6.1. Introducción.....	88
3.6.2. Transformación de modelos	89
3.6.3. MDD en plataformas telemonitorización.....	90
3.7. Desarrollo orientado a componentes	91
3.7.1. Introducción.....	91
3.7.2. Características.....	92
3.7.3. Desarrollo orientado a componentes en plataformas de telemonitorización ..	92
3.8. Revisión de enfoques para el diseño y desarrollo	92
3.9. Conclusiones.....	95
Framework e-MoDe	97
4.1. Introducción	99
4.2. Consideraciones preliminares de la propuesta.....	102
4.2.1. Enfoque para el desarrollo.....	102
4.2.2. Modelo de Arquitectura	106
4.3. Elementos de e-MoDe	108
4.3.1. Introducción.....	108
4.3.2. Metodología.....	110
4.3.3. Modelos y Herramientas Software.....	111
Metodología para el desarrollo de plataformas de telemonitorización	113
5.1. Introducción	115
5.2. Etapa 1: Especificación de Requisitos.....	115
5.2.1. Registro y análisis de la actividad	118
5.2.2. Especificación de interacciones y necesidades especiales del usuario	119
5.3. Etapa 2: Diseño	121
5.3.1. Arquitectura.....	122
5.3.2. Gestión de tareas	124
5.3.3. Colaboración	125
5.3.4. Usuario	126
5.3.5. Consideraciones finales sobre diseño	126
5.4. Etapa 3: Implementación.....	127
5.5. Etapa 4: Despliegue	130
5.6. Etapa 5: Validación	131
5.7. Visión general y esquema de aplicación de la metodología.....	131
Soporte al Diseño: Arquitectura y Modelos	137
6.1. Introducción	139
6.2. Arquitectura	139
6.2.1. Vista Hardware/Software	141
6.2.2. Vista de Interacción.....	143
6.2.3. Distribución de funcionalidades	144
6.3. Gestión de tareas de telemonitorización	148
6.3.1. Introducción.....	148
6.3.2. Proceso para la gestión de tareas	149

6.3.3. Modelo de Tareas	150
6.3.4. MVC como soporte a la gestión de modelos de tareas.....	152
6.4. Escenarios colaborativos en plataformas de telemonitorización.....	156
6.4.1. Introducción.....	156
6.4.2. Objetivos.....	156
6.4.3. Escenarios colaborativos.....	158
6.4.4. Modelo colaborativo	160
6.4.5. Arquitectura para la gestión de notificaciones	162
Soporte a la implementación: Zappa y WearIt.....	167
7.1. Introducción	169
7.2. Zappa	169
7.2.1. Introducción.....	169
7.2.2. Arquitectura.....	170
7.3. WearIt.....	177
7.3.1. Introducción.....	177
7.3.2. Práctica actual para la interacción con <i>wearables</i>	178
7.3.3. Nuevo proceso para la interacción con <i>wearables</i>	180
7.3.4. WearIt.....	184
Aplicación de e-MoDe.....	199
8.1. Introducción	201
8.2. CloudRehab.....	201
8.2.1. Introducción.....	201
8.2.2. Desarrollo de CloudRehab	201
8.3. CloudFit.....	217
8.3.1. Introducción.....	217
8.3.2. Aproximación clásica.....	217
8.3.3. Desarrollo de CloudFit.....	218
Conclusiones y Trabajo Futuro	245
9.1. Conclusiones.....	247
9.2. Trabajo Futuro.....	249
Conclusions and Future Work.....	251
9.1. Conclusions.....	253
9.2. Future Work	255
Bibliography.....	257

Índice de Figuras

Figura 1 - Interacción con dos <i>wearables</i>	29
Figure 1 – Interaction with two different wearables	38
Figura 2 - Interacciones y vistas de la aplicación móvil en VITAL [Diaz 2010]	48
Figura 3 - Escenario que pretende soportar MindGym [Gusev 2014]	48
Figura 4 - Arquitectura y capturas de la aplicación móvil [Pierleoni 2014]	49
Figura 5 - Arquitectura de REACTION [Reaction Web]	50
Figura 6 - Arquitectura de AMICA [Crespo 2010]	51
Figura 7 - Arquitectura de Dem@Care [DemoCare Web]	51
Figura 8 - Vista general de PREVIRNEC [Solana 2011]	52
Figura 9 - Arquitectura de KiReS [Anton 2013]	53
Figura 10 - Captura de la aplicación CloudRehab	53
Figura 11 – Capturas de la aplicación móvil Runtastic [Runtastic Web]	54
Figura 12 - Arquitectura de la plataforma nutricional presentada en [Six 2010]	55
Figura 13- Capturas de la aplicación MyFitnessPal	56
Figura 14- Capturas de CloudFit	57
Figura 15 - Arquitectura lógica de las plataformas de telemonitorización	58
Figura 16 - Objetivos de una plataforma de telemonitorización	64
Figura 17 - Tecnologías y conceptos usados en las diferentes propuestas del framework ..	67
Figure 18 – Modelo de tres niveles para la computación en la nube	69
Figure 19 – Ejemplo de una Red Inalámbrica de Área Personal (WBAN)	76
Figura 20 - Arquitectura del sistema con WBAN presentada en [Jovanov 2005]	78
Figura 21 - Usos del smartphone en una plataforma de telemonitorización	80
Figura 22 - Elementos de <i>SOA (Service Oriented Architecture)</i>	83
Figura 23 - Estructura de un mensaje SOAP	84
Figura 24 - Ejemplo de consumo de un recurso (REST)	85
Figura 25 - Ejemplo de la transformación entre modelos en MDA	89
Figura 26 – Enfoque clásico para el diseño propuesto por Roger, Sharp y Preece	93
Figura 27 - Elementos que conforman e-MoDe	100
Figura 28 - Esquema general del objetivo de la propuesta	101
Figura 29 – Modelo de arquitectura	107
Figura 30 – Vista general de los distintos artefactos de e-MoDe	109
Figura 31 - Análisis del protocolo de telemonitorización para la obtención de requisitos a través de las dos fases de análisis	117
Figura 32 - Proceso de participación y resultado de la etapa 1: Especificación de Requisitos	118
Figura 33 - Obtención de requisitos funcionales y no funcionales de una tarea de ejemplo (centrado en la actividad)	119
Figura 34 - Obtención de requisitos funcionales y no funcionales centrado en las necesidades del usuario y distintos tipos de interacciones	121
Figura 35 - División de la etapa de diseño en áreas de interés	122
Figura 36 Objetivos y resultados del bloque de <i>arquitectura</i>	124
Figura 37 - Visión de conjunto del modelo propuesto para la gestión de tareas	125
Figura 38 - Vista general de la etapa de diseño	127
Figura 39 - Distribución de tareas por grupos para la etapa de implementación	130
Figura 40 - Etapas y aspectos relevantes de la metodología	132

Figura 41 - Visión de conjunto de la composición de las distintas etapas de la metodología	133
Figura 42 - Visión de conjunto de la metodología y la participación de cada actor	134
Figura 43 – Guía de aplicación de la metodología	136
Figura 44 - Vista de la arquitectura hardware/software para plataformas de telemonitorización	142
Figura 45- Interacción de los diferentes módulos software.....	143
Figura 46 - Ejemplo de identificación de tareas y diseño de la API para el acceso a las funcionalidades.....	146
Figura 47 - Arquitectura híbrida con soporte cloud para plataformas de telemonitorización	147
Figura 48- Proceso para dar soporte a un nuevo rol.....	148
Figura 49 - Proceso de identificación y diseño de tareas de telemonitorización.....	149
Figura 50 - Metamodelo para la representación de tareas de telemonitorización.....	150
Figura 51 - Identificación y representación de tareas siguiendo el modelo	151
Figura 52 - Representación de una tarea en JSON a partir del modelo	153
Figura 53 - Gestión de tareas siguiendo un MVC.....	154
Figura 54- Adición de una nueva tarea de telemonitorización siguiendo el modelo para la gestión de tareas	155
Figura 55 - Escenario colaborativo para entornos de telemonitorización	159
Figura 56 - Modelo colaborativo para plataformas de telemonitorización	161
Figura 57 - Ejemplo de interacción entre supervisores y usuario monitorizado.....	162
Figura 58 - Distintos elementos del sistema de gestión de notificaciones	164
Figura 59 - Diseño de componentes en Zappa.....	173
Figura 60 - Arquitectura de la plataforma Zappa	174
Figura 61 - Gestión de diferentes interacciones con G.....	175
Figura 62 - Artefactos que componen el componente gAndroid.....	176
Figura 63 - Serialización de datos de diferentes wearables	179
Figura 64 -Método actual para la integración (o uso) de un wearable	180
Figura 65 – Método propuesto para la integración de wearables	181
Figura 66 - Integración de un wearable con el método actual	183
Figura 67 - Integración de un <i>wearable</i> usando el método actual.....	184
Figura 68 – Elementos (artefactos) que conforman WearIt.....	186
Figure 69 - Metamodelo para la representación de wearables.....	187
Figura 70 – Instanciación del metamodelo en un modelo de un wearable.....	189
Figura 71 - Herramienta de diseño de WearIt.....	190
Figura 72 - Generación de modelos usando XSLT.....	191
Figura 73 - Arquitectura del coordinador	191
Figura 74 - Algoritmo para la obtención del <i>payload</i>	192
Figura 75 - Proceso de integración de wearables usando WearIt.....	194
Figura 76- Integración de un wearable desde su diseño hasta su uso	195
Figura 77 - Diseño de un wearable multifuncional	197
Figura 78 - Código del modelo correspondiente al wearable	197
Figura 79 - Uso del wearable usando la plataforma Android.....	198
Figura 80 - Protocolo de telemonitorización de CloudRehab.....	203
Figura 81 - Especificación (parcial) de requisitos de CloudRehab	204
Figura 82 - Ejemplos de servicios en G	206
Figura 83 - Modelo colaborativo en CloudRehab	207
Figura 84 - Modelo para la tarea en CloudRehab.....	209

Figura 85 - Modelo del <i>wearable</i> usado en CloudRehab	211
Figura 86 – Interacción con el <i>wearable</i> a través del coordinador.....	211
Figura 87 - Plataforma web de CloudRehab	213
Figura 88 – Aplicación móvil CloudRehab (Vista Supervisor)	214
Figura 89 – Aplicación móvil CloudRehab (Vista Paciente)	215
Figura 90- Protocolo de telemonitorización de CloudFit.....	220
Figura 91 - Especificación (parcial) de requisitos de CloudFit.....	222
Figura 92 – Acceso a un recurso en CloudFit.....	223
Figura 93 – Modelo colaborativo para CloudFit	224
Figura 94 – Modelo para tarea de CloudFit.....	226
Figura 95 - Plataforma Web CloudFit.....	231
Figura 96 - Aplicación móvil de CloudFit	232
Figura 97 – Metodología para la evaluación y comparación de aplicaciones fitness.....	236
Figura 98 – Capturas de pantalla Endomondo	237
Figura 99– Capturas de pantalla Runtastic	237
Figura 100 – Metodología para la evaluación y comparación de aplicaciones fitness.....	238
Figura 101 – Campo de evaluación a completar por los evaluadores.....	238
Figura 102 – Registro de errores (violaciones) en las diferentes aplicaciones	240
Figura 103 – Número de errores por categoría y aplicación	242
Figura 104 – Interfaz de CloudFit antes de aplicar el estudio.....	243
Figura 105 – Interfaz de CloudFit después de aplicar el estudio.....	243

Índice de Tablas

Tabla 1 – Enfoques para el desarrollo [Saffer 2009].....	94
Tabla 2 - Resumen de contribuciones del framework y reto/objetivo a cumplir.....	110
Tabla 3 - Tipos de notificaciones principales del modelo.....	161
Tabla 4 – Heurísticas para evaluación de aplicación fitness [Silva 2014]	235
Tabla 5 – Resultado del estudio para las diferentes aplicaciones.....	239
Tabla 6 – Datos completos del estudio, por aplicación y categoría	241

Capítulo 1

Prefacio

1.1. Introducción

Las plataformas de telemonitorización permiten el seguimiento remoto de la actividad de unos usuarios (*usuarios monitorizados*) por parte de otros usuarios (*supervisores*). Estas plataformas de telemonitorización están cada vez más presentes en numerosos ámbitos: educación [Takahashi 2015], carrera del espacio [Zhan 2014], militar [Kumar 2012] y especialmente, salud y bienestar [Panicker 2015] [Kumar 2014], donde han supuesto un significativo avance al permitir ofrecer nuevos servicios o mejorar los ya existentes [Averwater 2005] [Berenson 2009].

Este aumento del uso de plataformas de telemonitorización, particularmente en el ámbito de salud y bienestar en países desarrollados, está directamente relacionado con el aumento de la esperanza de vida, acompañado de un aumento del tiempo en el que las personas conviven con enfermedades y discapacidades [Benyahia 2013].

En la actualidad, el patrón epidemiológico dominante está representado por las enfermedades crónicas. Los sistemas de salud, como respuesta a estos cambios demográficos y epidemiológicos, intentan dar una atención sanitaria continuada a los pacientes crónicos, además de fomentar los hábitos saludables en las poblaciones para permitir un envejecimiento activo y prevenir enfermedades [Gregoski 2012] [Pinnock 2013]. Esto conlleva un incremento significativo del coste sanitario (personal, recursos) que puede amenazar la sostenibilidad de estos sistemas.

Actualmente, con el fin de abordar este problema, se intentan aprovechar los avances de las tecnologías de la información y campos relacionados. En efecto, existen varias tecnologías que pueden extender la capacidad de los sistemas sanitarios, permitiendo la supervisión remota de pacientes desde sus propios domicilios, mejorando la monitorización, y diagnóstico y fomentando la independencia de los pacientes.

Las tecnologías clave que soportan los sistemas de monitorización remota o plataformas de telemonitorización actuales son: la computación *wearable*, computación móvil y computación en la nube.

Los **wearables** son dispositivos electrónicos colocados en alguna parte del cuerpo, capaces de proporcionar datos acerca del entorno o sujeto que los porta [Mann 1997]. Los datos que proporcionan estos dispositivos suelen ser fisiológicos (pulso cardíaco, temperatura corporal), inerciales (movimiento) o contextuales (posición, temperatura ambiente).

La **computación en la nube** [Ambrust 2010] permite ofrecer servicios deslocalizados en tiempo real y escalables. Estas características pueden responder a las demandas usuales de una plataforma de telemonitorización: acceso en cualquier momento a la información, número incremental de usuarios, robustez en el acceso a servicios o funcionalidades, etc.

Por último, los **smartphones y tablets** [Wang 2016] se han convertido en dispositivos de uso cotidiano y ha revolucionado el ámbito de la computación móvil. El uso de aplicaciones móviles ha permitido dotar a las plataformas de telemonitorización de la posibilidad de interactuar con el usuario monitorizado en cualquier momento, acceder a la información que necesite, realizar actividades relacionadas con la telemonitorización, captura y colección de datos proporcionados por los *wearables*, etc.

El uso integrado de la computación en la nube, smartphones/tablets y *wearables*, y su uso en plataformas de telemonitorización, otorga una serie de ventajas o propiedades, entre las que destacan:

- **Deslocalización geográfica:** El uso del smartphone, junto con el despliegue de servicios y almacenamiento de datos en la nube, permite que un usuario que disponga de conexión a Internet pueda realizar desde cualquier lugar tareas de telemonitorización.
- **Acceso a la información en tiempo real:** La habitualidad de los dispositivos inteligentes, como los smartphones, en nuestra vida cotidiana, y la robustez de los servicios de la nube, hace posible acceder a la información de un usuario o a datos concretos de un proceso de telemonitorización en tiempo real. Esto permite que, tanto usuarios monitorizados, como supervisores, puedan tener permanentemente acceso a información actualizada sobre el proceso de telemonitorización.
- **Automatización de la recogida de información:** Tradicionalmente, las soluciones de monitorización o seguimiento de usuarios estaban basadas en el registro manual o semiautomático de datos, donde el tratamiento de los mismos dependía en gran medida de la pericia y disciplina de los usuarios monitorizados (para, por ejemplo: anotar fechas, tomar el pulso cardíaco, registrar tiempos de ejecución o resultado de una tarea, traspasar a la computadora datos de formularios rellenos a mano, etc.). Este proceso afectaba a la calidad del seguimiento debido a la escasa fiabilidad de los datos recogidos, ya que, al realizarse de forma manual, con frecuencia aparecían omisiones, imprecisiones y/o errores. Las soluciones TIC actuales permiten automatizar la recogida de esta información, garantizando la fiabilidad de los datos de cara a evaluar, por ejemplo, el estado de salud de un usuario a través de un conjunto de valores de pulso cardíaco.
- **Captura de datos fisiológicos, inerciales y contextuales:** La posibilidad de usar dispositivos (*wearables*) para obtener determinados datos del usuario monitorizado permite la toma de estos de forma automatizada y continuada, fortaleciendo así su fiabilidad y permitiendo detectar anomalías puntuales que pasarían desapercibidas en una sesión de medición en laboratorio.

Durante el desarrollo de una plataforma de telemonitorización surgen determinados retos que deben abordarse y solucionarse para permitir dotar de dichas ventajas a la plataforma de telemonitorización: integración de *wearables*, desarrollo de las diferentes aplicaciones para llevar a cabo las tareas de telemonitorización, desarrollo de la arquitectura software que permita desplegar los servicios y su acceso en tiempo real por los diferentes usuarios a través de diferentes dispositivos, etc.

Estos retos, junto con otros que puedan derivarse del propio proceso de desarrollo (como abordar el proceso, definir los objetivos, etc.), ocasionan que el proceso de desarrollo de una plataforma de telemonitorización sea complejo.

1.2. Motivación

El desarrollo de una plataforma de telemonitorización lleva implícito dar soporte a objetivos propios del proceso de telemonitorización, como reducir desplazamientos, mejorar los procesos de seguimiento, eficiencia, reducción de costes, etc. [Seto 2008] [Meystre 2005].

No obstante, la plataforma resultante del proceso de desarrollo pretende satisfacer unos objetivos concretos sobre la actividad que se pretende monitorizar, como pueden ser: la telerehabilitación de algún paciente con alguna discapacidad, el seguimiento del estado de

salud mediante el pulso cardíaco, el entrenamiento de un deportista o la supervisión de personas mayores en casa, entre otros. Estos objetivos se especifican por los expertos en el ámbito o están previamente especificados en el denominado *protocolo* o *metodología de telemonitorización*.

Un protocolo de telemonitorización establece los pasos o etapas para llevar a cabo la supervisión de un usuario monitorizado, los objetivos, la duración, etc., fijados por expertos del ámbito donde se explota la plataforma como, por ejemplo, la rehabilitación del ligamento cruzado [Shelbourne 1990] o la supervisión de pacientes con problemas cardíacos [Ronaldson 2011].

Abordar el desarrollo de una plataforma de telemonitorización partiendo de un protocolo de telemonitorización simplifica a los ingenieros el proceso de desarrollo, puesto que las necesidades de los usuarios, las tareas de telemonitorización, los objetivos y las funcionalidades de la futura plataforma ya están especificadas en el protocolo.

Dicho esto, el proceso de desarrollo de una plataforma de telemonitorización está condicionado por tres factores (no tecnológicos):

1. **Soporte al protocolo.** La plataforma de telemonitorización suele ser una herramienta de soporte al protocolo de telemonitorización. El desarrollo está condicionado al protocolo y a los cambios de este, ya que es el protocolo especifica los objetivos y tareas de telemonitorización (entre otros aspectos) de la plataforma.
2. **Adaptación al usuario.** El usuario monitorizado, que será el actor principal de la plataforma, suele presentar necesidades o características muy particulares, ya son habituales los usuarios con problemas de salud, alguna enfermedad crónica o problemas cognitivos, lo que requiere un tratamiento personalizado, además de cumplir criterios de usabilidad y funcionalidad (*Technology Acceptance Model* - TAM) [Davis 1989]. Aunque durante el proceso desarrollo de una plataforma de telemonitorización se intenta garantizar la usabilidad y funcionalidad de las aplicaciones por parte del usuario, este requisito no se suele considerar al mismo nivel que otros, especialmente tecnológicos, por lo que las necesidades del usuario monitorizado la mayoría de las veces pasan a un segundo plano. Para evitar esta situación, lo idóneo es tratar como prioridad las necesidades del usuario monitorizado durante el proceso de desarrollo.
3. **Interacción entre actores.** Normalmente, el desarrollo de la plataforma se lleva a cabo colaborativamente por expertos en el dominio de la plataforma de telemonitorización e ingenieros de software [VanWormer 2012] [Schulz 2012] [Reddy 2011]. Esta interacción es compleja, puesto que cada uno tiene su punto de vista particular acerca de la plataforma y un bagaje formativo diferente, lo que dificulta con frecuencia la comunicación y el entendimiento. Normalmente, los expertos del dominio actores de apoyo en el proceso de desarrollo, ayudando a ingenieros y programadores a especificar las tareas de telemonitorización, definir los objetivos, hacer una correcta interpretación del protocolo de telemonitorización, detectar las necesidades y *feedback* de los usuarios monitorizados, etc.

La adaptación que se hace del protocolo, la importancia que se le presta a las necesidades del usuario monitorizado y la interacción entre los diferentes actores para llevar a cabo el

desarrollo ocasiona que cada proceso de desarrollo sea distinto y se aborde de forma diferente.

Como alternativa a esta manera (ad-hoc) de enfocar el desarrollo, existen metodologías para abordar el desarrollo de soluciones de telemonitorización que pretenden, en mayor o menor medida, guiar el proceso paso a paso [Lasierra 2012] [Wong 2008].

Sin embargo, la mayoría de las metodologías que se pueden encontrar en la literatura tienen dos carencias o desventajas comunes:

- **Contextualización.** Aunque es cierto que la mayoría de las plataformas de telemonitorización pertenecen al ámbito de la salud y bienestar, las metodologías para el desarrollo suelen estar orientadas a plataformas que afectan a grupos poblacionales particulares, condicionando los objetivos y la mayoría de las veces, el desarrollo. Esto implica que los propósitos de la plataforma y las necesidades del usuario están predefinidos, por lo que las metodologías son difícilmente reutilizables con otros protocolos de telemonitorización y usuarios (con otras necesidades).
- **Escasa utilidad para ingenieros o responsables del desarrollo.** En la mayoría de las soluciones existentes, ya sea porque no se profundiza en cómo abordar cada etapa (qué objetivos tiene, quién interviene, cómo se realiza correctamente, etc.), porque no se dan pautas para orientar el diseño software o la implementación (por ejemplo, a través de modelos o herramientas) o por usar metodologías ya conocidas y que se aplicarían igual que en cualquier otro contexto, este tipo de metodologías, aunque dedicadas para plataformas de telemonitorización, no ayudan a simplificar el proceso de desarrollo significativamente.

Además de estos factores, existen otros condicionantes relacionados con la utilización de la tecnología.

Aunque los retos pueden depender en gran medida de cada plataforma, existen funcionalidades normalmente presentes en cualquier plataforma de telemonitorización y que dependen de factores principalmente tecnológicos como, por ejemplo: el soporte a la colaboración entre supervisores, añadir nuevas funcionalidades para adaptarse a cambios en el protocolo o integrar un *wearable* en la plataforma para proporcionar una funcionalidad o dato necesario para la telemonitorización.

Dar soporte a algunas de estas funcionalidades está relacionado con la capacidad de diseñar soluciones adecuadas. Sin embargo, el soporte a otras funcionalidades está directamente relacionado con la dificultad que conlleva su implementación, como puede ser la integración de *wearables*.

En efecto, a la hora de interactuar con un *wearable*, se deben considerar tres aspectos: (1) el protocolo de comunicación para interactuar con él (Bluetooth, Wifi, ZigBee, etc.), (2) los datos que proporciona (pulso cardíaco, temperatura) y (3) cómo se proporciona dicha información cuando se envía, esto es, la estructura (orden de los datos, longitud).

Debido al amplio abanico de *wearables* existente, y que rara vez coinciden estas tres características, cada *wearable* presenta una configuración prácticamente única (ver Figura 1). Esto resulta en una amplia heterogeneidad de formas para interactuar con cada uno de ellos.

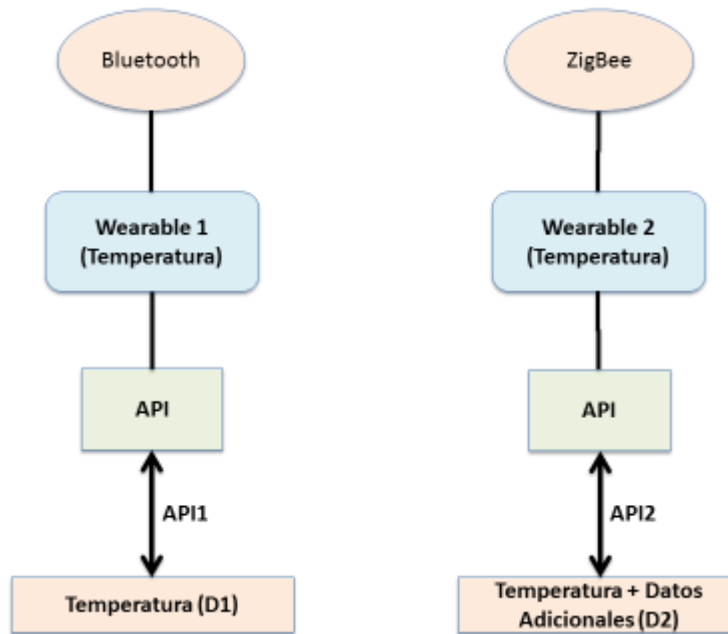


Figura 1 - Interacción con dos wearables

En la Figura 1, se representa un ejemplo de esta situación donde un *wearable (1)* usa protocolo de comunicación (Bluetooth) y una orden particular para obtener la temperatura (API1), devolviendo como dato únicamente dicha temperatura en un formato específico que ha definido su fabricante (D1). Sin embargo, otro *wearable (2)* además de estar basado en otro protocolo de comunicación totalmente distinto (ZigBee) y tener otra orden para obtener la temperatura (API2), cuando se le solicita, devuelve la temperatura junto a otra información adicional en otro formato diferente (D2).

Por tanto, con carácter general, para interactuar con un *wearable* un programador necesita:

- Conocer el protocolo de comunicación que usa el *wearable* en la plataforma donde se vaya a interactuar con él (Android, iOS, Linux, etc.).
- Estudiar cómo funciona el *wearable* a través de su *datasheet* (hoja de especificaciones) para conocer, entre otras cosas, las diferentes expresiones/órdenes para hacer uso de alguna funcionalidad o solicitarle al *wearable* un dato en concreto (API1, API2 en la Figura 1).
- Aprender conceptos y técnicas de bajo nivel (o específicas) para poder obtener la información relevante (o “*payload*”) del *wearable*, puesto que la información que proporciona un *wearable* se envía en un buffer de datos (desplazamientos, buffer circular, verificación de paquetes). Además de las órdenes o comandos necesarios para interactuar con él.

A partir de la información aprendida con estas actividades, el desarrollador puede generar una solución particular (código fuente) para interactuar con un *wearable* concreto, ya que cada programador implementa las funcionalidades necesarias para utilizar el protocolo de comunicación y el procesamiento de datos para obtener la información relevante del *wearable*.

Esto ocasiona que no exista una estandarización en la interacción (o uso) con *wearables*, dificultando la reutilización de dicha solución por otros programadores.

1.3. Planteamiento del problema

A modo de resumen, podemos identificar los diferentes retos presentes en el desarrollo de una plataforma de telemonitorización:

- **Heterogeneidad tecnológica.** Debido al amplio número de tecnologías que intervienen en una plataforma de telemonitorización (computación en la nube, *wearables*, *smartphones/tablets*, diferentes lenguajes de programación, soporte de datos, aplicaciones en diferentes plataformas, etc.), se hace necesario un amplio espectro de conocimientos de muchas tecnologías para afrontar cada desarrollo.
- **Integración de wearables.** La integración de *wearables* en una plataforma de telemonitorización es un reto para programadores, puesto que se requiere conocimiento sobre cómo funciona el dispositivo además de la tecnología adicional necesaria para poder usarlo. Este hecho, unido a la heterogeneidad existente (un gran número de dispositivos con diferentes tecnologías y que ofrecen diferentes funcionalidades) y la falta de estandarización para la interacción con *wearables* (dificultando la interoperabilidad de código) aumenta la complejidad de uso de estos dispositivos.
- **Soporte a la colaboración.** Las plataformas de telemonitorización suelen ser herramientas multidisciplinarias, donde expertos de diferentes ámbitos llevan a cabo procesos de telemonitorización sobre un mismo usuario. La necesidad de dar soporte a la colaboración entre supervisores en las plataformas de telemonitorización está cada vez más presente, lo que es un reto añadido para ingenieros y programadores, que deben diseñar e implementar los diferentes mecanismos que permitan dicha colaboración.
- **Usuarios con necesidades especiales.** Como se ha comentado anteriormente, con frecuencia ocurre que los usuarios monitorizados de plataformas de telemonitorización presentan algún tipo de discapacidad o déficit de cultura tecnológica. Considerar a estos, y sus necesidades, como elemento fundamental a tener en cuenta en el desarrollo de la plataforma es crucial para garantizar que la plataforma sea una herramienta usable y funcional.
- **Soporte al protocolo de telemonitorización.** En ocasiones, el desarrollo de cada plataforma se basa en un protocolo de telemonitorización, donde están reflejados los diferentes procesos de supervisión, la duración, los objetivos, etc. La plataforma resultante se convierte, por tanto, en una herramienta de soporte al protocolo, por lo que debe estar preparada para adaptarse a cambios que puedan producirse en dicho protocolo y tener así una vida útil lo más larga posible.
- **Soporte para el desarrollo.** En el desarrollo de una plataforma es habitual que participen actores de diferentes ámbitos. Unas líneas generales que permitan identificar las diferentes etapas del desarrollo, los objetivos, los actores involucrados, etc. simplificarían el proceso de desarrollo.
- **Reutilización de funcionalidades.** La mayoría de las plataformas de telemonitorización, especialmente si son del mismo ámbito, suelen compartir funcionalidades comunes (gestión de información, obtención del pulso cardíaco, acceso a la nube para guardar información, etc.). Para optimizar el desarrollo de una plataforma, sería muy útil poder reutilizar funcionalidades ya implementadas para otras plataformas.

1.4. Objetivos de la tesis

El trabajo presentado en esta tesis se centra en el proceso de desarrollo de plataformas de telemonitorización. Dada la ausencia de soporte metodológico y tecnológico a este tipo de procesos, el principal objetivo de este trabajo es definir un *framework* para simplificar el desarrollo de plataformas de telemonitorización. Para conseguir este objetivo general, se plantean los siguientes objetivos específicos:

- Proponer una **metodología** para abordar el desarrollo de una plataforma de telemonitorización de forma sistemática con el objetivo de facilitar esta tarea al desarrollador, especificando en cada etapa los objetivos, actores que participan y el resultado esperado.
- Desarrollar herramientas que **simplifiquen el proceso de integración de un *wearable*** y solucionen los problemas ya mencionados (heterogeneidad y falta de acceso estandarizado). Para esto, se pretenden definir APIs de alto nivel para que los programadores puedan acceder a los *wearables* fácilmente y, además, estandarizar el acceso a las funcionalidades de este tipo de dispositivos, de tal manera que las soluciones implementadas utilizando las herramientas propuestas sean reutilizables.
- Definir mecanismos para soportar la **interacción y colaboración** entre los actores de una plataforma de telemonitorización como elemento fundamental en los procesos de telemonitorización, especialmente para permitir la colaboración entre supervisores de diferentes ámbitos.
- Definir mecanismos para la **representación y gestión de tareas** en las plataformas de telemonitorización, con el objetivo de gestionar su creación, representación y uso, además de permitir incorporar nuevas tareas para soportar cambios en el protocolo de telemonitorización, esto es, que una plataforma de telemonitorización sea extensible en cuanto a tareas de telemonitorización.
- Desarrollar una herramienta que **simplifique el desarrollo de las aplicaciones móviles** de una plataforma de telemonitorización y que permita soportar cambios en el protocolo de telemonitorización y la reutilización de código para simplificar futuros desarrollos.
- Diseñar un **modelo de arquitectura** para plataformas de telemonitorización que permita representar e identificar los elementos comunes de una plataforma de telemonitorización y garantice la adaptación para los posibles cambios que se puedan realizar en el protocolo.
- **Demostrar la utilidad práctica de las propuestas** aplicándolas en plataformas de telemonitorización concretas.

1.5. Estructura de la tesis

La tesis se estructura de la siguiente manera:

- El capítulo 2 muestra una vista general de las plataformas de telemonitorización, mostrando algunos ejemplos, identificando los diferentes objetivos, como se plantea el desarrollo de una plataforma de telemonitorización y la arquitectura lógica de las plataformas de telemonitorización.
- En el capítulo 3 se abordan diferentes conceptos y tecnologías presentes o aplicables en el desarrollo de una plataforma de telemonitorización.

- El capítulo 4 introduce e-MoDe, el framework para el desarrollo de plataformas de telemonitorización y que es la principal aportación de la tesis. En este capítulo se muestra una vista general y los diferentes elementos que lo componen.
- En el capítulo 5 se aborda más en profundidad la metodología propuesta como parte del framework para guiar el desarrollo de plataformas de telemonitorización.
- En el capítulo 6 se muestran los modelos propuestos como soporte a la etapa de diseño de la metodología propuesta e incluidos en el framework.
- En el capítulo 7 se presentan las herramientas de soporte propuestas para la etapa de implementación de las aplicaciones móviles en una plataforma de telemonitorización.
- La validación del framework propuesto se realiza en el capítulo 8, a través de dos plataformas de telemonitorización concretas.
- Las conclusiones y el trabajo futuro se presentan en el capítulo 9.

Chapter 1

Preface

1.1. Introducción

Telemonitoring platforms allow remote supervision of user's activities (*monitored users*) by other users (*supervisors*). These platforms are increasingly presents in many areas: education [Takashi 2015], space race [Zhan 2014], military [Kumar 2012] and specially, healthcare and wellness [Panicker 2015] [Kumar 2014], where they have made significant progress enabling new services or improving the existing ones [Averwater 2005] [Berenson 2009].

The increased use of telemonitoring platforms, especially in healthcare and wellness area in the developed countries, is related with the increase of life expectancy and the number of people who are living with diseases and disabilities [Benyahia 2013].

Nowadays, chronic diseases represent the dominant epidemiological pattern. Healthcare systems, as support to this demographic and epidemiological changes, intend to provide continuing health care assistant to patients and promote health habits to enable the active aging and prevent diseases [Gregoski 2012] [Pinnock 2013]. This entails a significant increase in health care costs (staff, resources) that can threaten the suitability of these systems.

In order to address this goal, telemonitoring area takes advantage of advances in technology. Indeed, there are many technologies that enhance the healthcare systems, allowing the remote supervision of patients in their own homes improving the monitoring, diagnosis and promoting the independence of patients.

Currently, the key technologies that support the remote monitoring systems or telemonitoring platforms are: wearable computing, mobile computing and cloud computing.

Wearables are electronic devices that are worn under, with or on top of clothing, and are able to provide environmental information or information about the user who uses them [Mann 1997]. The information provided by these devices is, normally, physiological (heart rate, body temperature), inertial (movement) or environmental/contextual (location, temperature).

Cloud Computing [Ambrust 2010] can offer delocalized and scalable services in real time. These features can fit with the telemonitoring platforms demands: access to the information at any time, increasing number of users, robustness in the use of services or functionalities, etc.

Finally, the **smartphones and tablets** [Wang 2016] have become an everyday device, revolutionizing the field of mobile computing. The use of mobile applications provides to telemonitoring platform the ability to interact with the monitored user at any time, access to the information required, perform telemonitoring activities/tasks, capture and data collection provided by wearables, etc.

The integrated use of cloud computing, smartphone/tablets and wearables, and its use in telemonitoring platforms, provides a number of advantages or features, among which are:

- **Geographical relocation:** Use the smartphone, along with the deployment of services and cloud data storage, allows to users with Internet connection to perform telemonitoring tasks anywhere.

- **Access to information in real time:** The daily use of smart devices, as smartphones, and the robustness of cloud services, allows access to user information or specific information of a telemonitoring process in real time. This enables to monitored users and supervisors the access to updated information about the telemonitoring process at any time.
- **Automation information gathering:** Traditionally, monitoring solutions are based in the manual or semiautomatic data logging, where the management of these documents depends on the skills of the user (for example: note dates, take heartbeat, etc.). This process affects the quality of the monitoring process by the unreliability of the data collected, because when is done manually, often there were errors, inaccuracies and omissions. Current ICT solutions automate data collection, ensuring the reliability of the information to evaluate it, for example, the health status of a user through a set of heart rate values.
- **Physiological, inertial and contextual data capture (wearables):** The possibility to use wearable to get information related with the monitored user allow the data capture in an automatic way, ensuring its reliability and allowing the detection of specific abnormalities which are not detected in laboratory session.

During the development of a telemonitoring platform certain challenges that arise must be addressed and solved, in order to provide these advantages/features to the telemonitoring platform: wearables integration, development of the different application to perform telemonitoring tasks, development of the software architecture to deploy services and the access in real time by the different users, etc.

These challenges, along with others related with the development process (how to approach and address the process, define the goals, etc.), cause that the development of a telemonitoring platform is complex process.

1.2. Motivation

The development of a telemonitoring platform implicitly provides supports to telemonitoring goals, such as reduce displacements, improve monitoring process, efficiency, cost saving, etc. [Seto 2008] [Meystre 2005]

However, the resulting platform of the development process intents to satisfy specific goals in the telemonitoring area, such as the telerehabilitation of some type of patient with some disability, health status monitoring through heart rate values, the training of an athlete and the supervision of elderly people at home, among others. These goals are specified during the development by the experts in the area or are previously specified in the so-called telemonitoring protocol or methodology.

A telemonitoring protocol establishes the steps or stages to carry out the supervision of a monitored user, the goals, duration, etc., and is defined by the domain experts in the area where the platform will be used, like, for example, cruciate ligament rehabilitation [Shelbourne 1990] or monitoring of patients with cardiac problems [Ronaldson 2011].

Address the development of a telemonitoring platform using a telemonitoring protocol helps engineers to understand the user's needs, the telemonitoring tasks, the goals and the features of the future platform.

The development process of a telemonitoring platform is conditioned by three non-technological factors:

1. **Protocol Support.** The telemonitoring platform usually is a tool to support the telemonitoring protocol. Because of this, the development is conditioned by the protocol and its changes.
2. **User Adaptation.** The monitored user, who is the main actor of the platform, usually has special needs or particular features, because they are users with health problems, some chronic disease or cognitive problems, which requires a personalized management and satisfy with some usability and functionalities criteria (*Technology Acceptance Model - TAM*) [Davis 1989]. Although during the development process of a telemonitoring platform the usability and functionality of the different applications is addressed, this requirement is not at the same level as other (technological), because the monitored user needs are not a main requirement. To avoid this situation, is appropriate to consider as priority the monitored user needs during the development process.
3. **Actors interaction.** Usually, the development of a telemonitoring platform is carried out between software engineers and domain experts [VanWormer 2012] [Schulz 2012] [Reddy 2011]. This is a complex interaction, because each actor has specific goals and their own point of view of the platform, and a custom background, which often hinders the communication and understanding. Usually, domain experts should be assistants in the development process, providing help to engineers and programmers to specify the telemonitoring tasks, define goals, understand the telemonitoring protocol, detect the user needs and their feedback, etc.

The adaptation of the telemonitoring protocol, the importance that is given to the user needs and the interaction between the different actors to carry out the development cause that each development process different and addressed in a custom way.

As an alternative to this way (ad-hoc) to lead the development, there are methodologies to approach the development of telemonitoring solutions that intend to guide the development step by step [Lasierra 2012] [Wong 2008].

However, most of the methodologies that can be found in the literature have two main commons disadvantages or shortcomings:

- **Contextualization.** Because the most of telemonitoring platforms belong to the healthcare and wellness area, the methodologies for the development often they are focused to platforms for particular population groups, conditioning the goals and often, the development. This entails that the purposes of the platform and the user needs are predefined, so the methodologies are not able to be re-used with other protocols and users with other needs.
- **Useless for engineers or development leaders.** These types of methodologies don't simplify the development in a significant way, because of the lack of guidelines for the software design or implementation (for example, using models or tools), the use of already known methodologies applied in the same way that in other contexts, etc.

Besides these factors, there are others related with the use of technology that influence the development of a telemonitoring platform.

Although the challenges depend in some way of each platform, there are common functionalities normally presents in each telemonitoring platform such as the support to the collaboration between supervisors, add new functionalities to adapt to changes in the protocol and integrate a wearable in the platform to provide a functionality of required information for the telemonitoring, among others.

Support some of these functionalities are related with the ability to design suitable solutions, that is, design the software solution to support the functionality. However, other functionalities are directly related with the implementation process, like the wearables integration process.

Indeed, to interact with a wearable three aspects should be considered: (1) the communication protocol to interact with it (Bluetooth, WiFi, ZigBee), (2) the information provided (heart rate, temperature) and (3) how this information is provided, that is, the structure (data order, length).

Because of the wide range of wearable devices, and the rarely coincide of these three features, each wearable has an (almost) unique configuration (Figure 1). This cause a wide heterogeneity of ways to interact with each one.

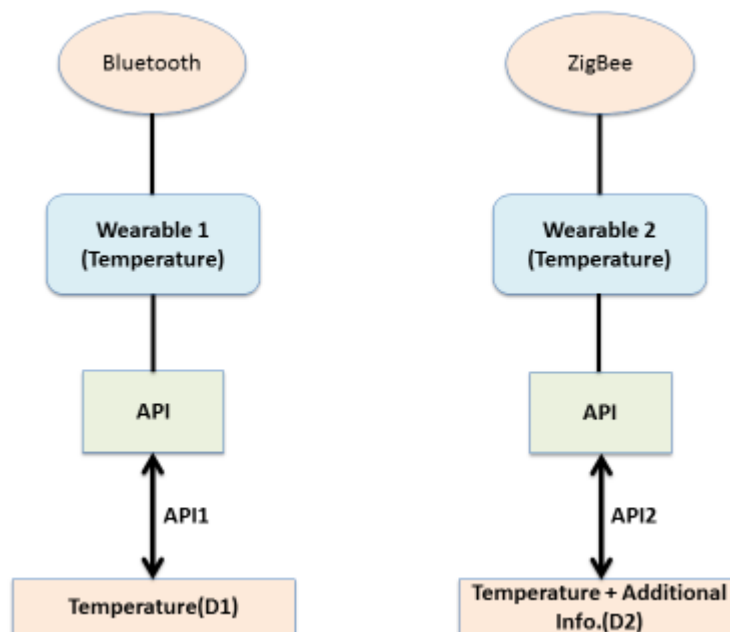


Figure 1 – Interaction with two different wearables

Figure 1 shows an example of this situation, where a wearable (1) uses a communication protocol (Bluetooth) and a specific command to get temperature (API1), returning as data the temperature in a specific format defined by the vendor (D1). However, other wearable (2) based on another communication protocol (ZigBee) uses other command to get the temperature (API2), getting the temperature and other additional information in a different format (D2).

Therefore, generally, to interact with a wearable a programmer needs:

- Learn about the communication protocol of the wearable in the platform where will be used (Android, iOS, Linux, etc.).
- Study how the wearable works through the datasheet to know the different commands to access to the information, how the wearable provide the information, etc.
- Learn concepts and techniques to manage the wearable data and obtain the payload, because the information is sent in a buffer, that is, serialized (offset, circular buffers, packet verification).

From the knowledge acquired with this three activities, programmers can generate a custom solution (source code) to interact with the wearable. Because of this, there is no a common standardization hinder the reuse of the solution by other programmers.

1.3. Problem Statement

To summarize, we can identify several challenges presented in the development of a telemonitoring platform:

- **Technological heterogeneity.** Because of the wide range of technologies that are involved in a telemonitoring platform (cloud computing, wearables, smartphones/tablets, several programming languages, data support, multiplatform applications, etc.), it is necessary a wide range of knowledge of many technologies to face each development.
- **Wearables integration.** The integration of wearables in a telemonitoring platform is a challenge for programmers, because it is necessary the knowledge about how the wearable works and the additional technology to interact with it. This fact, along with the device heterogeneity (a wide range of devices with several technologies and provide several functionalities) and the lack of standardization in the access to interact with a wearable (hindering the code interoperability), increase the complexity of use of a wearable.
- **Collaboration support.** Telemonitoring platforms occasionally are multidisciplinary tools, where experts from several areas carry out telemonitoring process with the same monitored user. The need to support the collaboration between supervisors in the telemonitoring platforms is increasingly a requirement, which is an added challenge to engineers and programmers, who have to design and implement the different mechanisms to enable the collaboration.
- **Users with special needs.** As it mentioned above, usually the users of the telemonitoring platform have some kind of disability or lack of technological culture. Consider these kind of users, and their needs, as a main element in the development of a telemonitoring is crucial to ensure the usability and functionality of the telemonitoring platform.
- **Telemonitoring protocol support.** Sometimes the development of a telemonitoring platform is supported by a telemonitoring protocol, where are specified the supervision process, duration, goals, etc. The platform obtained as result of the development process is a tool to support the protocol, so should be adaptable to changes in the protocol, thus having a long service life.
- **Development support.** In the development of a telemonitoring platform is common the participation of actors from several areas. A common guideline to identify the different

steps or stages of the development process, the goals, the actors involved, etc. can simplify the development process.

- **Reuse functionalities.** Most of telemonitoring platforms, especially in the same area, often share common functionalities (management of the information, heart rate data collection, access to the cloud to store information, etc.). To optimize the development of a platform, would be very useful reuse functionalities already implemented in other platforms.

1.4. Goals

The work presented in this thesis focuses in the process of the development of telemonitoring platforms. Because of lack of methodological and technological support to this kind of developments, the main goal of this work is to define a framework to simplify the development of telemonitoring platforms. To achieve this main goal, the following specific objectives arise from this aim:

- Propose a **methodology** to address the development of a telemonitoring platform in a systematic way, with the aim to facilitate this task to developers, specifying in each stage the goals, actors involved and the results expected.
- Development of tools to **simplify the integration process of a wearable** and solve the problems already mentioned (heterogeneity and lack of standardized access). To achieve this, high-level APIS will be defined to programmers, who can interact with wearable easily and, in addition, standardize the access to the functionalities of the wearable, enabling that the solutions implemented using the tools will be reusable.
- Define mechanism to support **interaction and collaboration** between actors in a telemonitoring platform as crucial element of the telemonitoring process, specially, to allow the collaboration between supervisors from different areas.
- Define mechanism for the **representation and management of tasks** in the telemonitoring platforms, with the goal to manage the creation, representation and use of these tasks. In addition, the possibility to add easily new telemonitoring tasks to support changes in the telemonitoring protocol, enabling an extensible telemonitoring platform.
- Develop a new tool to **simplify the development of the mobile applications** of a telemonitoring platforms and, can support changes in the telemonitoring protocol and the reuse of source code to simplify future developments of telemonitoring platforms.
- Design an **architecture model** for telemonitoring platforms to represent and identify the common elements presented in a telemonitoring platform, and ensure the adaptation to telemonitoring protocol changes.
- **Demonstrate the practical useful of these proposals** applying them in specific telemonitoring platforms.

1.5. Thesis Structure

The thesis is structured as follows:

- Chapter 2 provides an overview of the telemonitoring platforms, presenting some examples, identifying the different goals, addressing the development process and the logic architecture of the telemonitoring platforms.

- In the chapter 3 are addressed the different concepts and technologies presented or used in the development of a telemonitoring platform and used in the contributions of this thesis.
- e-MoDe, a framework for the development of the telemonitoring platforms, is presented in chapter 4. This chapter provides an overview of the proposal and the different elements that compose it.
- In the chapter 5 is addressed the methodology proposed for the development process of a telemonitoring platform.
- Chapter 6 provides the different models proposed as support to the design stage.
- Chapter 7 presents the different tools proposed to be used in the implementation of the mobile applications, in the implementation stage of the methodology.
- The framework validation is proposed in the chapter 8, through two specific telemonitoring platforms: CloudRehab and CloudFit.
- Conclusions and future work are presented in chapter 9.

Capítulo 2

Plataformas de Telemonitorización

2.1. Introducción

En el marco de esta tesis, el término *telemonitorización* se define de la siguiente manera:

El seguimiento remoto de la actividad de un tipo de usuario (usuarios monitorizados) por parte de otro tipo de usuario (usuarios supervisores), independientemente del área en el que se enmarque dicha actividad o de los objetivos que se pretendan alcanzar a través de dicho seguimiento

La telemonitorización, como vocablo y concepto, suele asociarse principalmente al área de salud y bienestar, puesto que la mayoría de las soluciones de telemonitorización pertenecen a dicho ámbito [Chaudhry 2010] [Schmidt 2010]. Por esto, es común encontrar numerosas definiciones en la literatura sobre qué se entiende por *telemonitorización*, como la expuesta en [Pandor 2013]: **la telemonitorización es el uso de la TICs (Tecnologías de la Información y la Comunicación) para transmitir información relacionada con la salud y bienestar del paciente entre usuarios individuales separados geográficamente.**

La puesta en práctica del concepto de telemonitorización a través del uso de elementos software (aplicaciones móviles, plataformas web, aplicaciones de escritorio, servicios, documentos) y hardware (ordenadores, smartphome, *wearables*) da lugar a las denominadas plataformas de telemonitorización.

Las plataformas de telemonitorización persiguen dar soporte a ciertas tareas de supervisión como, por ejemplo, controlar la realización de actividades de rehabilitación de un paciente o monitorizar las caídas de una persona mayor. No obstante, e independientemente del ámbito/área, la telemonitorización plantea una serie de objetivos generales que pueden extrapolarse de [Rice 2011]:

- **Prescindir de los desplazamientos:** Permitir proporcionar a los usuarios monitorizados sin la necesidad de estar físicamente en el mismo sitio.
- **Detectar eventos o situaciones concretas:** A través de la captura de datos, se pueden controlar cuándo suceden determinados eventos o cuándo el usuario monitorizado se encuentra en una determinada situación. Por ejemplo, planificar una monitorización a la espera de que el pulso cardíaco del usuario monitorizado supere un umbral y activar una alerta.
- **Monitorización completa o continuada:** La telemonitorización permite registrar cualquier evento o dato proveniente del usuario monitorizado (pulso cardíaco, geoposición, que estaba haciendo, etc.) y almacenarlo en registros que podrán ser consultados a posteriori por supervisores con el objetivo de hacer un diagnóstico, determinar su evolución, su estado de salud en momento concreto, etc.

Estos objetivos generales, también llamados **independientes del dominio**, están implícitos en el propio concepto de telemonitorización y, por tanto, suelen estar soportados por las soluciones de telemonitorización.

A través de distintas herramientas software y hardware que conforman una plataforma de telemonitorización se permite al supervisor/experto planificar el proceso de

telemonitorización: estableciendo, dependiendo ya de cada aplicación concreta, cuestiones como, por ejemplo, sesiones de rehabilitación que a posteriori el usuario monitorizado tiene que realizar, intervalos de registro del pulso cardíaco, entrenamientos deportivos en soluciones fitness, etc.

El usuario monitorizado, mediante herramientas software (aplicaciones) y hardware (smartphone, *wearables*, ordenador), realiza las actividades correspondientes a través de una interacción persona-computador que queda registrada y almacenada. A posteriori, toda esta información se almacena en un soporte de datos, normalmente externo, siendo accesible por el supervisor quien evalúa dicha ejecución para determinar el progreso o evolución del usuario monitorizado.

La evolución tecnológica ha permitido cada vez disponer de herramientas más potentes y versátiles, lo que ha ocasionado que las plataformas de telemonitorización hayan experimentado un creciente auge durante la última década.

Este crecimiento ha sido especialmente notable en el área de la salud [Vandelanotte 2016] [Whitehouse 2014], desde las primeras plataformas basadas en el intercambio de SMSs (*Short Message Service*) [Ostojic 2005] [Büher 1999] hasta las plataformas basadas en tecnología *cloud*, *wearables* y *smartphones* [Ruiz-Zafra 2013] [Wang 2014] [Bisio 2015] [Sarria-Ere 2015].

Además de este aspecto tecnológico, el auge de estas plataformas, su buena aceptación y, sus buenas expectativas para el futuro se deben también a otros factores, entre los que destacan:

- Gran parte del procesamiento de los datos se puede realizar de manera automática [Allen 1999], reduciendo considerablemente el trabajo del personal supervisor y garantizando la consistencia y fiabilidad de los datos.
- La automatización de procesos de telemonitorización, gracias a las TICs, lleva implícito una reducción de costes, al ahorrar, tanto recursos, como tiempo. Numerosos textos de la literatura a través de diversos estudios han demostrado cómo las plataformas de telemonitorización permiten ahorrar costes, tanto para el usuario monitorizado (reducción de desplazamientos, reducción de tiempos), como para los servicios públicos o privados (menos recursos, reducción de personal) [Dimmick 2000] [Halvorsen 1996] [Johnston 2000] [Lobley 1997] [Mair 2000] [Seto 2008].
- Existen grupos poblacionales, como pueden ser las personas mayores que viven solas [Scanail 2006] o pacientes con alguna dolencia crónica [Paré 2007] que, debido a su necesidad de recibir una supervisión o tratamiento continuado, demandan unos recursos de tiempo, personal, material que incrementan sustancialmente los costes para los servicios de salud. Las plataformas de telemonitorización se presentan como una alternativa a explotar en estos grupos poblacionales, con el objetivo de reducir costes a largo plazo.

2.2. Contexto de aplicación: casos de uso

Las plataformas de telemonitorización suelen enmarcarse en categorías dependiendo de los objetivos [Meystre 2005]. Por ejemplo, existen diferentes casos dentro del área de salud y bienestar, como por ejemplo la supervisión de personas mayores, la telerehabilitación de

pacientes con daño cerebral, pacientes con algún tipo de dolencia cardíaca, atletas que necesitan una supervisión experta, etc.

En este capítulo se pretende mostrar plataformas de telemonitorización de diferentes categorías del ámbito de la salud y bienestar, al ser el ámbito más destacable. Estas categorías son:

- Telemonitorización para el bienestar de personas mayores
- Telemonitorización dolencias crónicas
- Telemonitorización para la rehabilitación
- Fitness

2.2.1. Telemonitorización para el bienestar de personas mayores

VITAL

VITAL (*Vital Assistance for The Elderly*) es una plataforma de telemonitorización, enmarcada en un proyecto europeo, que tiene el objetivo de mejorar la calidad de vida de personas mayores [Díaz 2010].

Esta plataforma está formada por un conjunto de herramientas para la asistencia personal de los usuarios, dando soporte a necesidades básicas en este ámbito concreto, como pueden ser: comunicación entre actores, asesoramiento personal, capacidad para moverse con seguridad por el entorno físico, etc.

VITAL, a nivel tecnológico, está basado en un conjunto de servicios desplegados en un servidor central, el uso de la televisión y el teléfono móvil como herramientas del usuario monitorizado, el dispositivo del operador/guía y el software necesario (aplicaciones) para dar soporte a las distintas funcionalidades.

El objetivo de esta herramienta no es exclusivamente la de proporcionar asistencia personal en casa para personas con reducida movilidad, sino dar asistencia en otros escenarios como pueden ser museos, espacios públicos o privados en interiores. A través del dispositivo móvil el operador (guía o supervisor en este escenario) ayuda a los usuarios en caso de que estos lo necesiten, proporcionándole una asistencia personalizada.

Un escenario concreto, presentado en el artículo, podría ser el usuario (persona mayor de 60 años) que asiste por primera vez a un museo. Este usuario, usando su dispositivo móvil con conexión a Internet accede a los servicios proporcionados por VITAL, determina su localización y se detecta que se encuentra en ese museo en concreto. A partir de esta información se le proporciona al usuario la guía completa (texto, audio y vídeo) que se descarga y visualiza en su propio dispositivo, permitiéndole tener un plan de actuación, ayudándolo así con asistencia sin necesidad de tener una compañía física permanente durante toda la visita (ver Figura 2).



Figura 2 - Interacciones y vistas de la aplicación móvil en VITAL [Diaz 2010]

MindGym

MindGym es una plataforma de telemonitorización para personas mayores basada en el concepto de intercambio de información a través de la televisión por protocolo de Internet o IPTV en inglés (*Internet Protocol Television*) [Gusev 2014].

Esta plataforma se basa en la idea de que las personas mayores tienen una enorme dificultad para aprender nuevas tecnologías y para acostumbrarse a su uso, mientras que por una tradición cultural son relativamente dependientes de la televisión donde se encuentran en un escenario más familiar por su uso cotidiano.

El objetivo de la plataforma es usar tecnología Cloud para, a través de IPTV, generar una serie de tareas o actividades que permitan entrenar y mantener activa a las personas mayores, todo esto a través de la propia televisión.

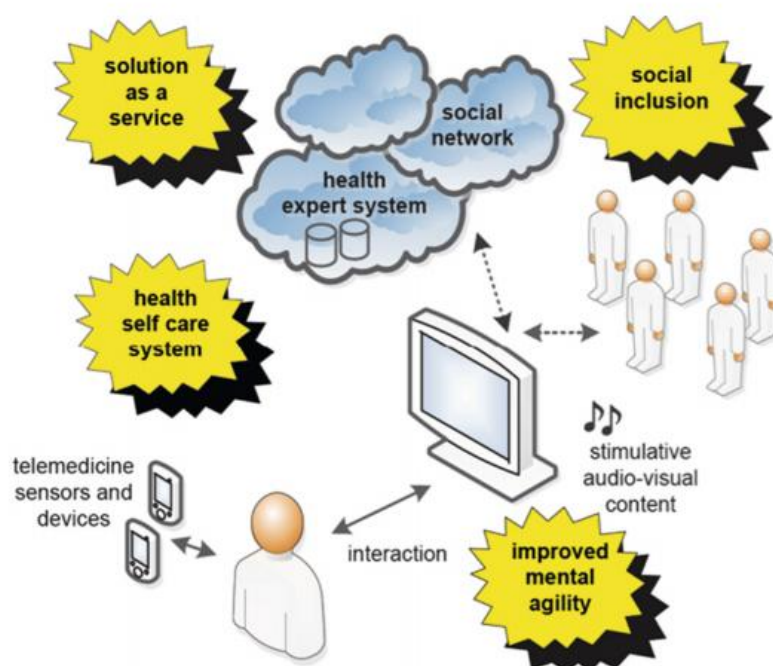


Figura 3 - Escenario que pretende soportar MindGym [Gusev 2014]

Por ejemplo, mostrar a través de la televisión juegos interactivos donde entrenarán su capacidad de memoria o reacción a través de la propia televisión o dispositivos móviles, usar programas de televisión educativos (tutoriales) específicamente creados para el proyecto, etc. (ver Figura 3).

El software para mostrar la información por televisión está basado en Java y es totalmente interactivo, siendo registrada toda esta interacción por parte del usuario en la nube. A posteriori, los supervisores acceden a dicho registro para determinar cómo ha realizado el ejercicio y poder así determinar su interés, si es la actividad adecuada, si deben proponer otro tipo de ejercicios, subir el nivel, etc.

Sistema de telemonitorización cardiovascular para personas mayores

El trabajo presentado en [Pierleoni 2014] es una plataforma de telemonitorización para personas mayores (aunque también aplicable a personas con problemas cardíacos) basada en el uso de *wearables* (un pulsómetro en este caso) y smartphones/tablets (Android).

El objetivo de esta solución es llevar a cabo una monitorización constante para determinar, mediante un algoritmo, el nivel de estrés del paciente. Si dicho nivel de estrés, que indicaría un problema cardíaco o una alteración que necesitaría asistencia, es inferior o superior a unos umbrales determinados para ese usuario en concreto (configurable en la aplicación) se notifica mediante SMS a los familiares para que lo atiendan lo antes posible. Además, se comunica con un servidor que a su vez hace una petición de ayuda a los servicios de emergencia más cercanos (ver Figura 4).

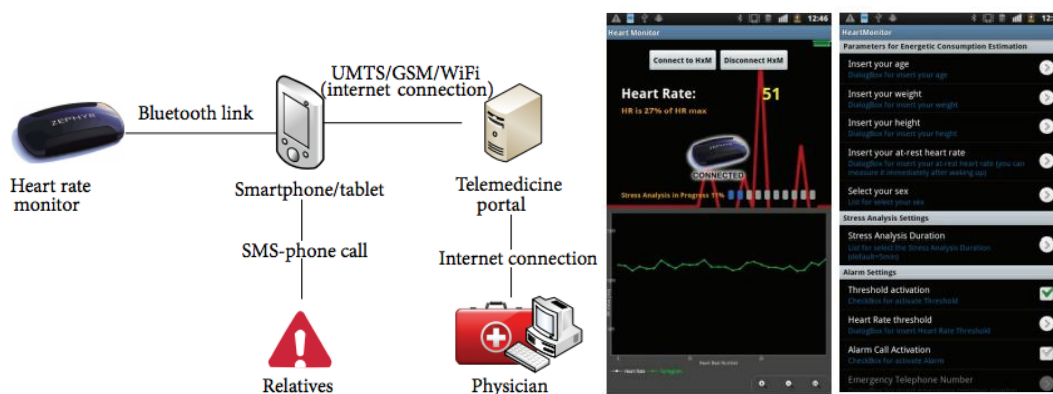


Figura 4 - Arquitectura y capturas de la aplicación móvil [Pierleoni 2014]

De esta manera, la plataforma pretende supervisar automáticamente la salud del paciente a través del pulso cardíaco, actuando cuando es necesario y contactando con aquellos actores que pueden ser de ayuda.

2.2.2. Telemonitorización de dolencias crónicas

REACTION

REACTION (*Remote Accessibility to Diabetes Management and Therapy in Operational healthcare Networks*) es una plataforma de telemonitorización financiada con fondos europeos y centrada en pacientes con diabetes [Reaction Web] (ver Figura 5).

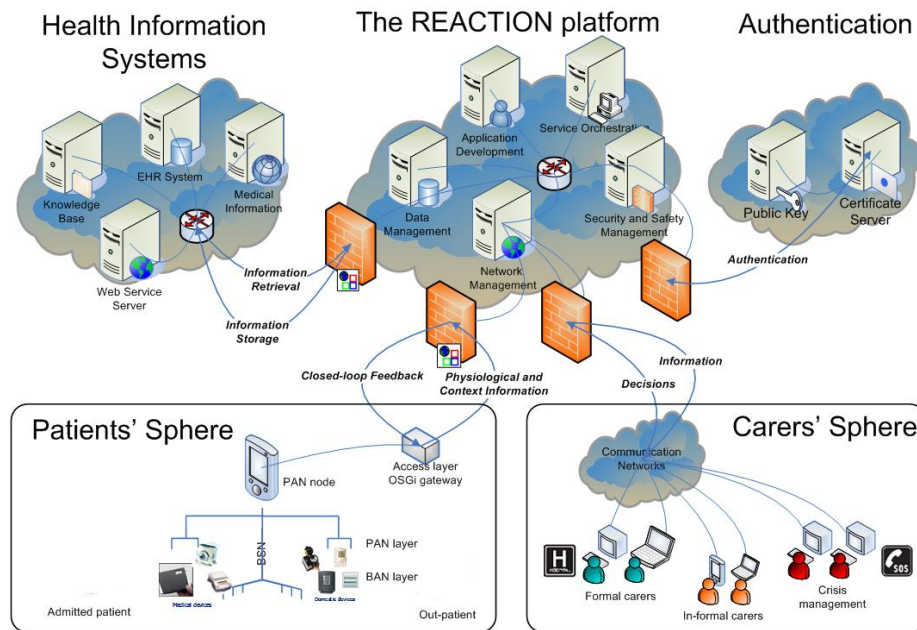


Figura 5 - Arquitectura de REACTION [Reaction Web]

Esta plataforma usa *wearables* como medidores de glucosa y dispensadores de insulina, y su diseño se basa en una arquitectura orientada a servicios (SOA) soportada por tecnología *cloud*.

La plataforma permite servicios profesionales para la monitorización y gestión para diferentes pacientes con diabetes, entre los que destacan:

1. Ayuda a la decisión de profesionales sobre el diagnóstico.
2. Fiabilidad en la monitorización.
3. Largos periodos de monitorización sin necesidad de estar en el centro médico.
4. Cuidados sobre los aspectos relacionados con la diabetes.
5. Soporte para adaptarse a los cambios de vida de los pacientes.

AMICA

AMICA (*Autonomy, Motivation & Individual Self Management for COPD Patients*) es una plataforma de telemonitorización para detectar obstrucciones pulmonares crónicas (COPD - *Chronic Obstructive Pulmonary Disease*- en inglés) [Crespo 2010].

Esta plataforma combina tecnologías móviles como PDAs y *wearables*. A través del pulsómetro, electrocardiograma y medidor de oxígeno se van recopilando datos por parte del paciente que a posteriori son visualizados por los expertos que dictaminan el estado de salud del paciente y realizan un pronóstico de su evolución para un futuro (ver Figura 6).

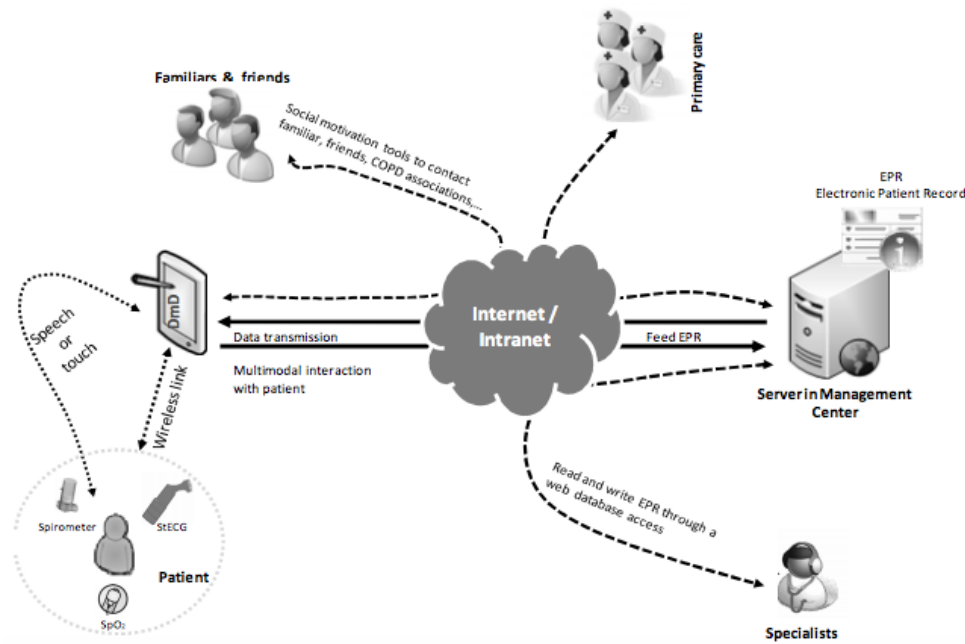


Figura 6 - Arquitectura de AMICA [Crespo 2010]

Dem@Care

Dem@care (*Dementia Ambient Care*) es un proyecto que pretende ser una herramienta para el diagnóstico, asesoramiento e independencia de los pacientes con demencia [Hopper 2015] [Democare Web]. Este proyecto, financiado con fondos europeos, es uno de los que más repercusión y expectativas ha tenido en este ámbito tanto por los recursos de los que dispone como por sus asociados (Philips, Airbus, IBM, Universidad de Bordeux, etc.).

Esta plataforma está basada en el concepto de aunar un entorno ubicuo (sensores de temperatura, cámaras de video, Kinect) junto con una red de área de área corporal -WBAN- (micrófonos, cámara, sensores inerciales) para monitorizar todas las actividades de una persona con demencia en su entorno en concreto, como puede ser su casa.



Figura 7 - Arquitectura de Dem@Care [DemoCare Web]

Esta captura de datos de un entorno sensible al contexto y los datos de los wearables que lleva el paciente permiten generar un mejor diagnóstico y determinar el mejor tratamiento o modo de actuación con el paciente en concreto, proporcionándole un *feedback* sobre las tareas que esté realizando (ver Figura 7).

Además, gracias a todas las pruebas con pacientes realizadas a cabo, Dem@Care está generando numerosas líneas de investigación relacionadas con la telemonitorización, pero en otro contexto más allá de una asistencia remota, como, por ejemplo: uso de ontologías para detectar patrones de movimiento en personas con alzhéimer, estudios de usabilidad de las TICs en personas con alzhéimer, etc.

2.2.3. Telemonitorización para la rehabilitación

PREVIRNEC

PREVIRNEC es una plataforma de telemonitorización centrada en la telerehabilitación para pacientes con problemas cognitivos, que tiene como objetivo reforzar y mejorar las relaciones entre neuropsicólogos y los propios pacientes [Solana 2011].

Esta plataforma de telemonitorización está basada en tecnología web, donde pacientes y supervisores van a interactuar. A través de herramientas web los pacientes realizarán ejercicios o juegos propuesto por los supervisores. Una vez completados los ejercicios o tareas serán revisados por los propios supervisores (ver Figura 8).

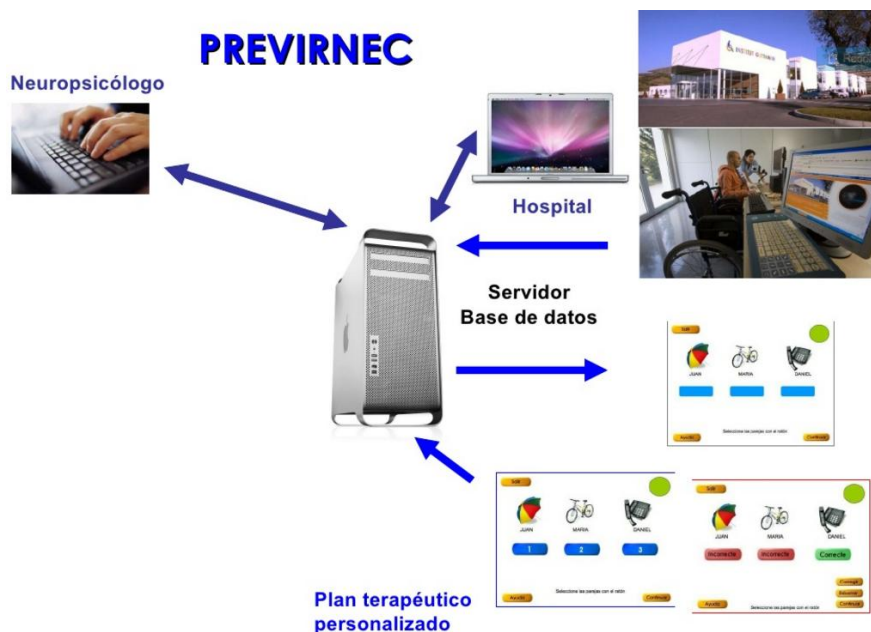


Figura 8 - Vista general de PREVIRNEC [Solana 2011]

La plataforma hace especial hincapié en dos aspectos fundamentales: la comunicación y la usabilidad. Debido a que se basa en soluciones web, la usabilidad por parte del paciente es un punto importante para garantizar que sea una herramienta útil y práctica. De igual manera, la comunicación entre supervisores y pacientes es otro aspecto relevante, especialmente para que el paciente, para que este no se sienta incomunicado en todo el proceso de rehabilitación.

KiReS

KiReS (*A Kinect-based telerehabilitation System*) es una plataforma de telemonitorización centrada en la telerehabilitación de persona con escasa movilidad, que haciendo uso de un sistema de reconocimiento de formas capaz de determinar movimientos compuestos [Anton 2013].

Este tipo de soluciones de telerehabilitación ha tenido mucho auge durante la última década gracias a tecnologías como Kinect de Microsoft y otros dispositivos de ocio como Wii de Nintendo.

Kinect, gracias al uso de marcadores, permite hacer un reconocimiento de formas más preciso, lo que permitiría determinar con gran precisión si el movimiento que ha realizado el paciente es correcto (con respecto al movimiento ideal). Esta información, que es almacenada en un servidor donde además hay un sistema de procesamiento para determinar el movimiento, es revisada por el supervisor pudiendo así evaluar la ejecución del ejercicio y así poder planificar mejor futuras sesiones de rehabilitación (ver Figura 9).

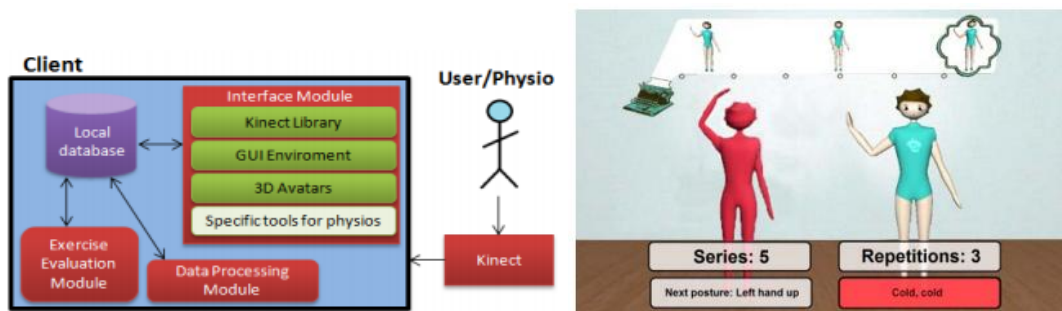


Figura 9 - Arquitectura de KiReS [Anton 2013]

CloudRehab

CloudRehab es una plataforma de telemonitorización desarrollada para recuperar, en la medida de lo posible, las capacidad cognitivas y motrices de pacientes que han sufrido un daño cerebral adquirido (DCA) [Ruiz-Zafra 2013] [Urbano 2014].

Esta plataforma usa tecnología *cloud*, *wearables* (pulsómetro) y *smartphone*. La plataforma, compuesta por una plataforma web y una aplicación móvil (Android), permite a los supervisores programar sesiones de rehabilitación basadas en vídeo tutoriales, que a posteriori, y en casa, los pacientes realizarán usando el dispositivo móvil (ver Figura 10).

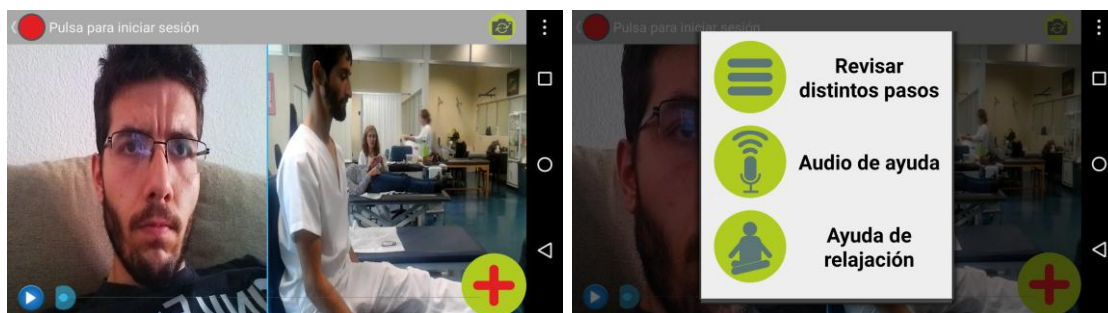


Figura 10 - Captura de la aplicación CloudRehab

El pulsómetro se usa para determinar el nivel de estrés y en caso de superar determinados umbrales, mostrar mensajes de ayuda que sirvan al paciente a reducir el nivel de estrés.

Esta plataforma, como aportación de la presente tesis se encuentra extensamente explicada en el capítulo 8.

2.2.4. Fitness

Monitorización de la actividad

La mayoría de las plataformas de telemonitorización relacionadas para gestionar la actividad física están diseñadas para ser autogestionadas por el propio usuario, esto es, es el propio usuario quien, sin ningún tipo de supervisión, a priori, realiza los ejercicios que considera oportunos y evalúa su propia evolución.

Haciendo uso de smartphones y *wearables*, a través de estas herramientas (apps) se hace un seguimiento continuo de la posición (usando el GPS) para determinar la distancia recorrida, los valores cardíacos durante el mismo, las calorías gastadas, el ritmo, la comparación con ejercicios anteriores, etc.

Toda esta información se almacena en el propio móvil o en un servidor externo que a posteriori si puede ser consultada por quien corresponda. No obstante, este tipo de plataformas por lo general no están pensadas para llevar una telemonitorización basada en una planificación y unos objetivos de salud (en este caso) concretos.

Algunos ejemplos de este tipo de soluciones serían los presentados en los trabajos [Eskofier 2008] [Mccarthy 2013] [Depari 2013] [Runtastic Web]. La Figura 11 muestra una captura de Runtastic, una de las principales aplicaciones relacionadas con la gestión de la actividad física, siendo además una de las más comerciales [Runtastic Web].

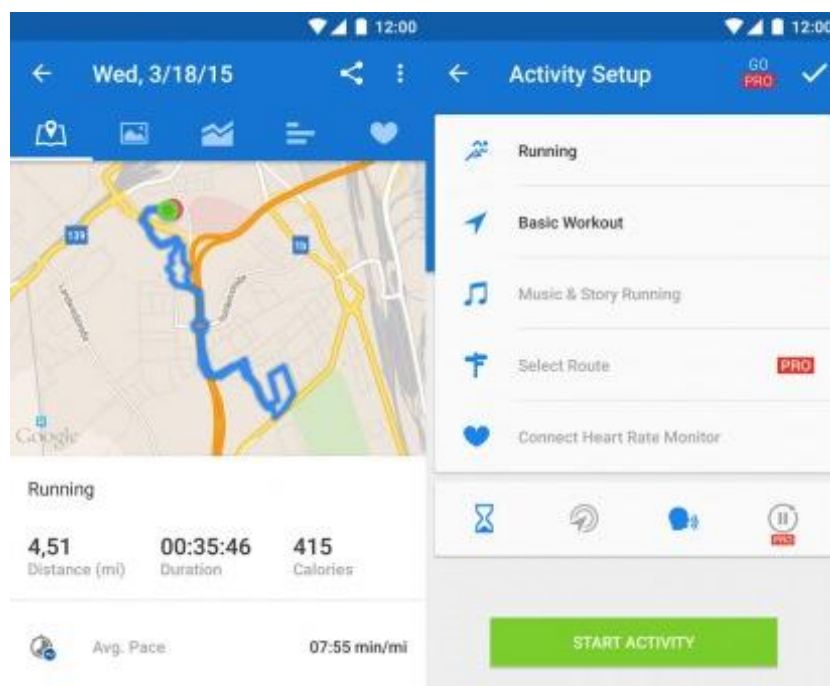


Figura 11 – Capturas de la aplicación móvil Runtastic [Runtastic Web]

Monitorización nutricional

Los usuarios enmarcados dentro del ámbito del deporte (atletas, deportistas), además de monitorizar su actividad física, requieren de soluciones de telemonitorización para gestionar su ingesta calórica, pudiendo así planificar una telemonitorización a lo largo del tiempo para conseguir los objetivos propuestos.

Existen soluciones de telemonitorización que permiten a los usuarios ir registrando la ingesta calórica, introduciendo los alimentos y porciones que ingieren, para al final del día tomar las calorías propuestas.

En algunas soluciones, que serían las consideradas plataformas de telemonitorización, toda esta información se almacena en una fuente externa como un servidor para, a posteriori, ser revisada por un experto (nutricionista, endocrino) que es quien determina si se van cumpliendo los objetivos.

Algunas de estas soluciones son las expuestas en [Six 2010] y [Lee 2010], mostrando en la figura 12 la arquitectura de la solución propuesta en [Six 2010].

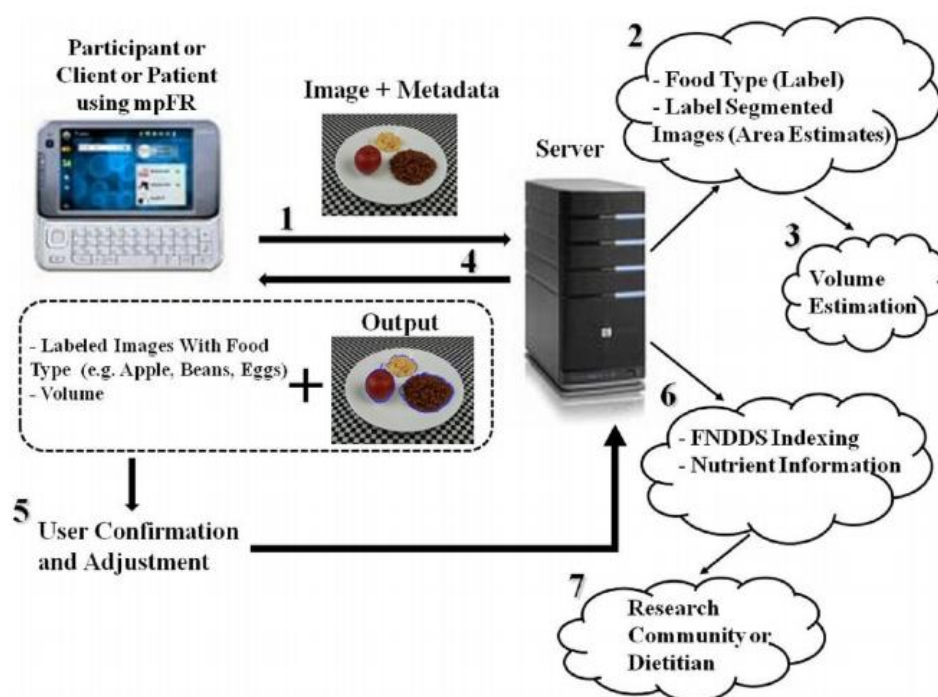


Figura 12 - Arquitectura de la plataforma nutricional presentada en [Six 2010]

Actualmente, además de numerosos proyectos de investigación como los ya mencionados, existen varias soluciones más comerciales y que están teniendo una gran aceptación al ser soluciones colaborativas, donde los usuarios además de registrar su ingesta calórica pueden añadir nuevos alimentos a una base de datos común, y por lo tanto ir mediante una colaboración de la comunidad ir ampliando la base de datos de alimentos disponible.

Incluso, alguna de estas soluciones, permiten mediante el escaneo de código de barras, determinar qué alimento se trata, lo que simplifica mucho el registro de datos por parte del

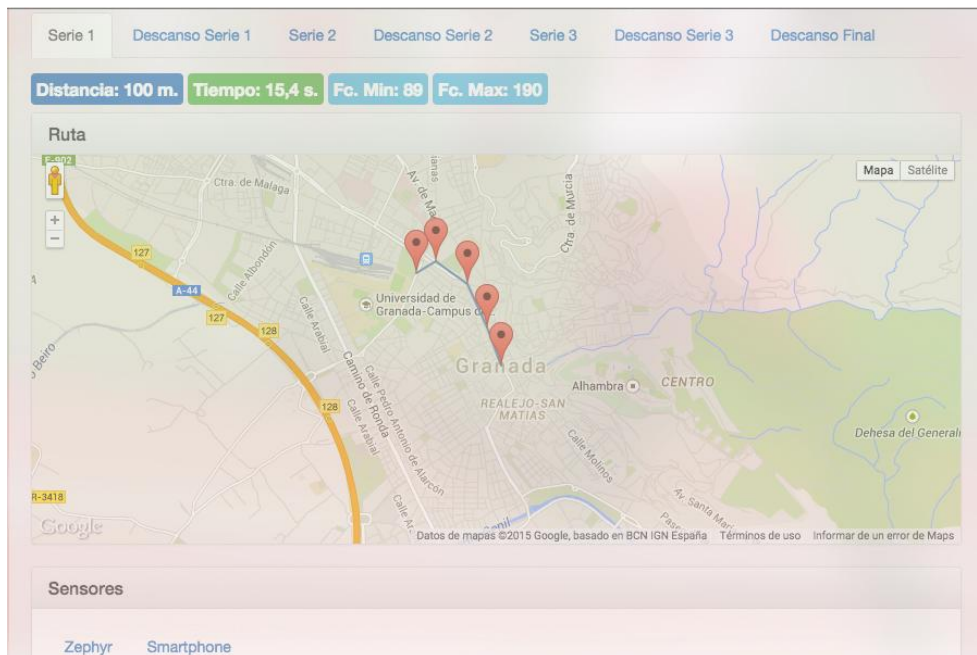
usuario. Algunas de estas soluciones son MyFitnessPal (ver Figura 13) [MyFitnessPal Web] o FitMacro [FitMacro Web].



Figura 13- Capturas de la aplicación MyFitnessPal

CloudFit

CloudFit es una plataforma de telemonitorización multidisciplinar para la gestión integral del bienestar y salud de diferentes tipos de usuarios, esto es, gestionar la actividad física, recuperación, estado de salud, nutrición, etc. Esta plataforma de telemonitorización, que en su versión actual da soporte a atletas y entrenadores para la gestión de entrenamientos, está basada en una plataforma web y una aplicación móvil [Ruiz-Zafra 2014] (ver Figura 14).



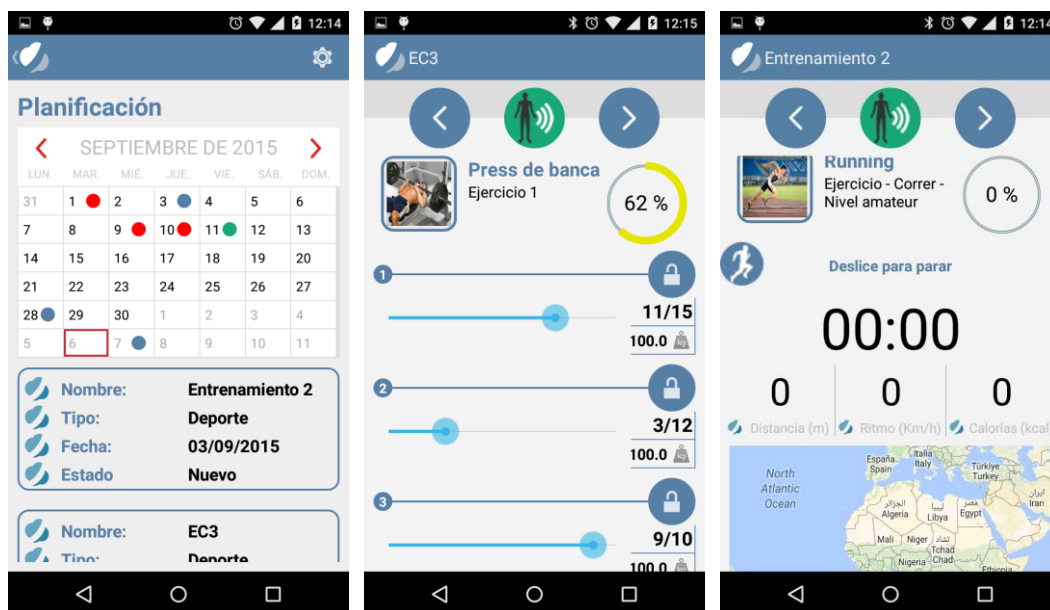


Figura 14- Capturas de CloudFit

A través de la plataforma web el entrenador gestiona los entrenamientos y el atleta a través del dispositivo móvil realiza dichos entrenamientos. Estos entrenamientos, una vez realizados, se almacenan en la nube y serán revisados por el entrenador para comprobar la consecución de objetivos.

Además, el atleta usa wearables como pulsómetro y acelerómetro para ir monitorizando las constantes vitales y la cantidad de movimiento mientras se realiza las distintas actividades físicas dentro del entrenamiento.

Esta plataforma, como contribución, se encuentra extensamente explicada en el capítulo 8.

2.3. Arquitectura Lógica de las plataformas de telemonitorización

Al abordar el desarrollo de una plataforma de telemonitorización se estudia la tecnología existente y que funcionalidades se pueden llegar a soportar a través de dicha tecnología.

Revisando la literatura en periodos de tiempo determinados, se puede apreciar que todas aquellas plataformas que tienen el mismo objetivo (rehabilitación de un determinado tipo de pacientes, monitorización de personas mayores) estaban soportadas principalmente por la misma tecnología. Como sucede en la mayoría de los ámbitos dentro del software, las soluciones están marcadas por la tecnología de su época.

Por ejemplo, soluciones de telemonitorización en la década de los 90 que usaban ordenadores personales y teléfonos móviles para la recepción de SMS [Ostojic 2005], ya que era la tecnología pública más avanzada de la que se disponía. De igual manera que durante esta década hay una tendencia predominante del uso de smartphones y *wearables* [Ruiz-Zafra 2014].

En los últimos años la tendencia general en el desarrollo de plataformas de telemonitorización está marcada por tres tecnologías que están normalmente presentes en cualquier plataforma de telemonitorización (ver Figura 15):

- **Smartphone.** Uno de los aspectos fundamentales en una plataforma de telemonitorización es el uso de software por parte del usuario monitorizado para la realización de alguna tarea o interacción, que servirá para el propósito de la telemonitorización. El smartphone, por sus características tales como precio, eficiencia, portabilidad, potencia, versatilidad, etc.; es una tecnología muy usada en la mayoría de las plataformas de telemonitorización actuales, siendo un elemento indispensable al dar soporte a muchas de las funcionalidades que suelen estar presentes en la mayoría de las plataformas de telemonitorización: realización de tareas, recepción y envío de mensajes para una interacción en tiempo real, coordinador con *wearables* y con sistemas de almacenamiento externo, etc.
- **Wearables.** Dentro de un proceso de telemonitorización, los datos fisiológicos, inerciales o contextuales del usuario monitorizado pueden ser necesarios para el objetivo de la telemonitorización. Para dar soporte a esta funcionalidad se usan *wearables*, dispositivos electrónicos de reducido tamaño que proporcionan una funcionalidad concreta y que, junto con el uso del smartphone, permiten de manera no tan intrusiva obtener diferentes tipos de valores, mostrarlos y procesarlos a través del smartphone. Un ejemplo de *wearable* podría ser un pulsómetro, un GPS, un acelerómetro, un Smartwatch o las Google Glass.
- **Sistemas de almacenamiento.** Una de las características fundamentales de cualquier plataforma de telemonitorización actual es el intercambio de información en tiempo real, desde cualquier sitio y entre diferentes tipos de dispositivos. Por esto, el sistema de almacenamiento, así como, los servicios para dar soporte al intercambio de información son un pilar fundamental para garantizar la propia telemonitorización. Durante esta última década la tecnología en la nube, por sus características (capítulo 3), se ha convertido en un referente y una de las tecnologías más usadas para dar soporte a plataformas de telemonitorización.

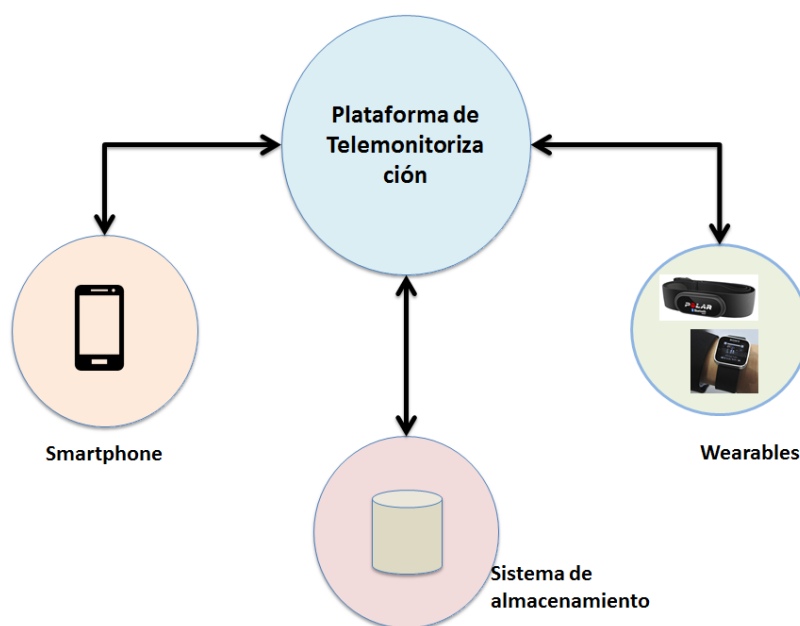


Figura 15 - Arquitectura lógica de las plataformas de telemonitorización

Estos tres elementos (smartphone, *wearables* y sistema de almacenamiento) conforman la arquitectura lógica de la mayoría de las soluciones de telemonitorización actuales.

La propuesta de framework presentada en esta tesis está basada en esta arquitectura lógica, que se extenderá a posteriori en los capítulos 4 y 6. De esta manera, los retos presentados se abordarán para intentar dar soluciones integrales relacionadas con esta área: mejorar la integración de wearables, generación de vistas en smartphone para dar soporte a varias tareas, que modelo de arquitectura software encaja para dar soporte al almacenamiento e interacción, etc.

2.4. Desarrollo de plataformas de telemonitorización

El desarrollo de una plataforma de telemonitorización en determinadas ocasiones es realizado únicamente por ingenieros de software y programadores, sin contar con expertos en el ámbito de la plataforma [Mehta 2012] [Diab 2013] [Sundaram 2013].

Esta manera de abordar el desarrollo de una plataforma de telemonitorización (sin contar con expertos en el área) limita bastante las posibilidades de éxito de la solución, puesto que las soluciones desarrolladas, aunque funcionales a nivel tecnológico, no siempre son las más adecuadas para los usuarios monitorizados en cuanto a qué servicios proporcionan y la ya mencionada usabilidad. Esto ocasiona que se obtenga una herramienta poco práctica con una vida útil relativamente corta.

Por otro lado, la participación directa de expertos en el ámbito en el proceso de desarrollo de una plataforma de telemonitorización no conlleva, de facto, un éxito, tanto en el propio proceso como en la plataforma resultante.

Hay que considerar que, aunque expertos en su área, este tipo de actores difícilmente pueden colaborar con los ingenieros de software en las diferentes etapas del proceso de desarrollo, al no tener conocimientos sobre la especificación de requisitos, diseño software, implementación, tecnologías disponibles y funcionalidades, etc. Por lo tanto, en los desarrollos donde participan tienen funciones de apoyo (o soporte) para especificar las tareas de telemonitorización, determinar los objetivos de la telemonitorización, transmitir las necesidades del usuario monitorizado, hacer una correcta interpretación del protocolo de telemonitorización, etc. En definitiva, ayudar a los ingenieros y programadores con su conocimiento para que la plataforma cumpla con todas las expectativas de los diferentes usuarios.

En el desarrollo de una plataforma de telemonitorización donde existe dicha colaboración se suele aplicar una metodología que permite guiar todo este proceso. Esta metodología suele ser *ad-hoc*, es decir, se define particularmente para cada desarrollo, donde, dependiendo de los grupos de trabajo, se marcan los objetivos, el ritmo de trabajo, interacciones de equipo, etc.

Esta manera de abordar el desarrollo puede ocasionar una mala planificación por una metodología deficiente, lo que ocasiona problemas en el propio proceso de desarrollo o incluso no conseguir satisfacer algún objetivo, ya sea por la falta de comunicación, por no especificar dicho objetivo cuando era oportuno, no solucionar problemas de la manera adecuada que repercuten en la solución a largo plazo, etc.

Otra alternativa es optar por metodologías para el desarrollo específicamente creadas para plataformas de telemonitorización, como las expuestas en [Lasierra 2012] [Wong 2008].

[Lasierra 2012] propone una metodología para el desarrollo de soluciones de telemedicina que comienza con la identificación de los objetivos y termina con la implementación del sistema. En esta propuesta se especifica, en la etapa de desarrollo, considerar cuestiones como aspectos técnicos, clínicos y sociales (pacientes). No obstante, la propuesta no especifica detalladamente cómo abordar estos tres aspectos, ni cómo llevarlos a cabo en la etapa de implementación.

[Wong 2008] propone seguir la metodología de desarrollo clásica (cascada) para abordar el desarrollo de una plataforma de telemedicina con soporte web y ontologías, al considerar dicha metodología adecuada.

Las principales carencias o desventajas de las propuestas existentes en la bibliografía son: (1) la contextualización de la propuesta, es decir, soluciones orientadas a un ámbito de telemonitorización concreto (pacientes con daño cerebral, pacientes con problemas cardíacos) y la dificultad de ser reutilizada en otro ámbito, al estar definidos unos procesos orientados a conseguir unos determinados objetivos; (2) la poca utilidad práctica, al proponer una metodología que no difiere con respecto a otros desarrollos de software o no ahonda lo suficiente en las diferentes etapas para especificar objetivos, actores participantes, resultado, etc.

Independientemente de la metodología que se aplique, el desarrollo de una plataforma de telemonitorización suele estar orientado a satisfacer unos objetivos concretos como pueden ser: la telerehabilitación de algún tipo de paciente con algún tipo de discapacidad, el seguimiento del estado de salud mediante el pulso cardíaco o supervisión de personas mayores, entre otros. Estos objetivos son definidos durante el propio desarrollo por los expertos en el área o vienen previamente especificados en el denominado protocolo de telemonitorización, también llamado metodología de telemonitorización.

Un protocolo de telemonitorización es creado por los expertos en el ámbito de la telemonitorización donde se exponen los pasos o etapas para llevar a cabo la supervisión de un usuario, los objetivos, la duración, etc. El protocolo de telemonitorización debe recoger, al menos:

1. Tareas propias para realizar la telemonitorización (p.ej. registro del pulso cardíaco)
2. Eventos asociados a esas tareas (p.ej. superación de ciertos valores de pulso cardíaco)
3. Necesidades especiales de los usuarios monitorizados (p.ej. problemas cognitivos, falta de cultura tecnológica)
4. Identificación de los distintos tipos de usuarios, así como, de las interacciones entre ellos.

En un escenario de telemonitorización existen distintos actores y roles, por lo que identificar cómo y cuándo van a interactuar es crucial para soportar los diferentes procesos de telemonitorización.

Algunos ejemplos de protocolo de telemonitorización son los expuestos en [Urbano 2014], [Shelbourne 1990] y [Ronaldson 2011], donde se presentan protocolos de telemonitorización en diferentes ámbitos (telerehabilitación de pacientes con daño cerebral, de pacientes con problemas cardíacos y para la recuperación del ligamento superior cruzado).

Comenzar el desarrollo de la plataforma de telemonitorización con el protocolo definido simplifica la tarea de los ingenieros, al estar ya especificadas las características de la plataforma, los actores, las necesidades del usuario, objetivos, etc.

Estos protocolos pueden evolucionar de acuerdo a resultados de investigación o estudios de campo, por lo que las plataformas de telemonitorización deben estar preparadas para adaptarse a cambios en el protocolo, pudiendo así dar soporte a nuevas funcionalidades.

Aunque en todas las soluciones software la calidad de los servicios que reciben los distintos actores es fundamental, en una plataforma de telemonitorización la calidad de estos servicios se considera un factor crítico. Un problema de funcionalidad, disponibilidad o incluso usabilidad de alguno de los servicios proporcionados por la plataforma puede ocasionar problemas que afectan directamente a la salud o estado del usuario monitorizado.

Por ejemplo, una plataforma de monitorización desarrollada para supervisar el pulso cardíaco en usuarios con algún tipo de dolencia cardíaca que no funcionase correctamente y no actuase como corresponde ante una subida del pulso cardíaco, puede tener desastrosas consecuencias sobre la salud o incluso la vida del usuario.

La mayoría de estas soluciones, al desarrollarse bajo una metodología de trabajo donde no se considera activamente la participación del usuario monitorizado y supervisor en el desarrollo de la misma, puede implicar que, aunque su funcionamiento sea adecuado, no cumple con determinados requisitos en cuanto a usabilidad y aceptación por parte del usuario, o bien no soporta adecuadamente el protocolo de telemonitorización.

Por otro lado, la heterogeneidad tecnológica presente en una plataforma de telemonitorización complica el proceso de desarrollo, ya que se hace necesario tener un extenso abanico de conocimientos en diferentes tecnologías para poder llevar a cabo el desarrollo.

Durante el proceso de desarrollo los ingenieros tienen que diseñar la arquitectura de la plataforma y las diferentes propuestas que permitan dar soporte a determinadas funcionalidades, con el propósito de cubrir objetivos determinados, como, por ejemplo, diseñar el soporte a la colaboración entre supervisores, tomar las decisiones de diseño adecuadas para garantizar la extensibilidad de la plataforma, permitir la adición de nuevas tareas de telemonitorización, etc.

De igual manera, los programadores durante la etapa de implementación se enfrentan a retos como puede ser la integración de *wearables* en la plataforma.

Todo lo anteriormente comentado ocasiona que el desarrollo de una plataforma de telemonitorización sea un proceso complejo.

2.5. Objetivos de las plataformas de telemonitorización

Cuando se aborda el desarrollo de una plataforma de telemonitorización, el objetivo es generar dicha plataforma software y que esta asegure la consecución de varios objetivos propios de la plataforma, normalmente especificados en el protocolo de telemonitorización.

Además de estos objetivos, la plataforma debe tener ciertas características que garanticen la consecución de objetivos propios de la telemonitorización, esto es, objetivos independientes del dominio [Rice 2011]:

1. Prescindir de desplazamientos.
2. Detectar eventos.
3. Monitorización completa y continua.

Estos objetivos se clasifican como **independientes del dominio** puesto que son objetivos asociados al concepto de telemonitorización, y por lo tanto cualquier aplicación del concepto de telemonitorización (no solo en el ámbito de ciencias de la computación) debe cumplir dichos objetivos.

Otros objetivos presentes en una plataforma de telemonitorización son los propios intrínsecos al concepto de plataforma, siendo estos los que se alcanzan por la aplicación de las TICs o por el propio uso de la plataforma. Estos objetivos, que llamaremos **objetivos de plataforma**, son:

- **Eficiencia y reducción de costes.** Las plataformas de telemonitorización, gracias al uso de las TICs, permiten automatizar y simplificar numerosos procesos, como puede ser la gestión de documentos, reuniones o realizar procesos telemáticos como puede ser la propia monitorización. Gracias a las plataformas de telemonitorización se mejora la eficiencia, lo que lleva directamente a una reducción de costes, tanto económicos como de inversión de tiempo.
- **Mejorar la calidad del servicio.** La mejora de la eficiencia y la consecuente reducción de costes es un factor relevante, pero no es el principal objetivo de una plataforma de telemonitorización. El objetivo fundamental es dar un servicio de telemonitorización, haciendo uso de las TICs, que se presupone mejor que el hasta ahora existente. Características, gracias a las TICs, como son la constante comunicación entre supervisor y usuario monitorizado, el acceso en tiempo real a la información o la fiabilidad de datos obtenidos a través de *wearables* son ejemplos de cómo una plataforma de telemonitorización puede mejorar los servicios de monitorización.
- **Evidencia práctica.** Las plataformas de telemonitorización tienen un claro objetivo de ser soluciones prácticas con resultados tangibles (mejorar la calidad de vida del paciente, controlar algún tipo de valor fisiológico que determine problemas de algún tipo, etc.). Dependiendo del ámbito, estos resultados serán validados bajo criterios de expertos del dominio. Por ejemplo, una plataforma de telerehabilitación para pacientes con daño cerebral tiene que servir para su propósito y, además, mejorar la rehabilitación de los pacientes de acuerdo a criterios médicos. Esta evidencia práctica y real de que cumple su objetivo es el principal objetivo de cualquier plataforma de telemonitorización, y debe ser un objetivo crucial a cumplir.
- **El usuario monitorizado es el actor principal.** Dentro de un escenario de telemonitorización, uno de los actores fundamentales es el usuario monitorizado, que es a quien está dirigida la plataforma de telemonitorización. Por lo tanto, tanto a nivel de desarrollo como de uso, la plataforma de telemonitorización deben garantizar todos los servicios necesarios que demande el usuario monitorizado, así como, cubrir criterios de modelo de aceptación de la tecnología (usabilidad y funcionalidad).

- **Mejorar la interacción.** Uno de los principales requisitos, y que gracias a las TICs es posible, es la interacción entre usuario monitorizado y usuario supervisor. Este objetivo es fundamental para garantizar una correcta monitorización, ya que el seguimiento continuo y el *feedback* del supervisor sobre el usuario monitorizado es crucial para que la monitorización vaya cumpliendo con las expectativas. Gracias a las nuevas tecnologías hoy en día es posible mantener una interacción en tiempo real y desde cualquier sitio.
- **Deslocalización.** En un contexto de telemonitorización, los diferentes actores del mismo no tienen por qué estar físicamente en el mismo sitio. Esto hace que la deslocalización con respecto al acceso de la información sea crucial para garantizar que cualquier usuario y desde cualquier sitio pueda acceder a la información, permitiendo así poder realizar los objetivos de la telemonitorización correctamente. Con las nuevas tecnologías como la computación en la nube, este acceso deslocalizado a la información, por las características de la propia tecnología, está cubierta.
- **Modelo de aceptación de la tecnología (Technology Acceptance Model - TAM).** En una plataforma de telemonitorización es crucial que los distintos actores que usan las distintas herramientas se sienten cómodos usándolas y que las encuentren útiles, llegando a tener estos la sensación de que su uso mejora su rendimiento en la tarea que esté realizando (TAM) [Davis 1989] [Venkatesh 2000] [Adams 1992]. Esto se conseguiría cumpliendo dos requisitos fundamentales: que sea útil (evidencia práctica) y que sea usable. Este cumplimiento de que las herramientas que forman parte de la plataforma de telemonitorización deben ser útiles y usables, son requisitos fundamentales para garantizar el éxito de la plataforma de telemonitorización, al ser percibida por el usuario monitorizado como una herramienta útil y cómoda que le ayuda a mejorar/progresar.

Además de estos objetivos independientes del dominio y de plataforma, existe un tercer grupo de objetivos que depende totalmente del ámbito para el que se cree la plataforma de telemonitorización.

Estos objetivos, llamados **dependientes del dominio**, son objetivos muy concretos y por lo general centrados en cumplir necesidades propias del protocolo de telemonitorización: determinar cuántas veces un usuario realiza determinada acción, permitir que la plataforma genere informes de revisión sobre un usuario, etc. Estos objetivos, normalmente, corresponden con las tareas de telemonitorización.

De esta manera, el desarrollo de una plataforma de telemonitorización suele estar orientada a satisfacer estos tres bloques básicos de objetivos, para garantizar que la plataforma es una herramienta útil en cuanto a funcionalidad y en cuanto a utilidad por parte del usuario al satisfacer los distintos objetivos.

La figura 16 muestra estos tres bloques de objetivos.

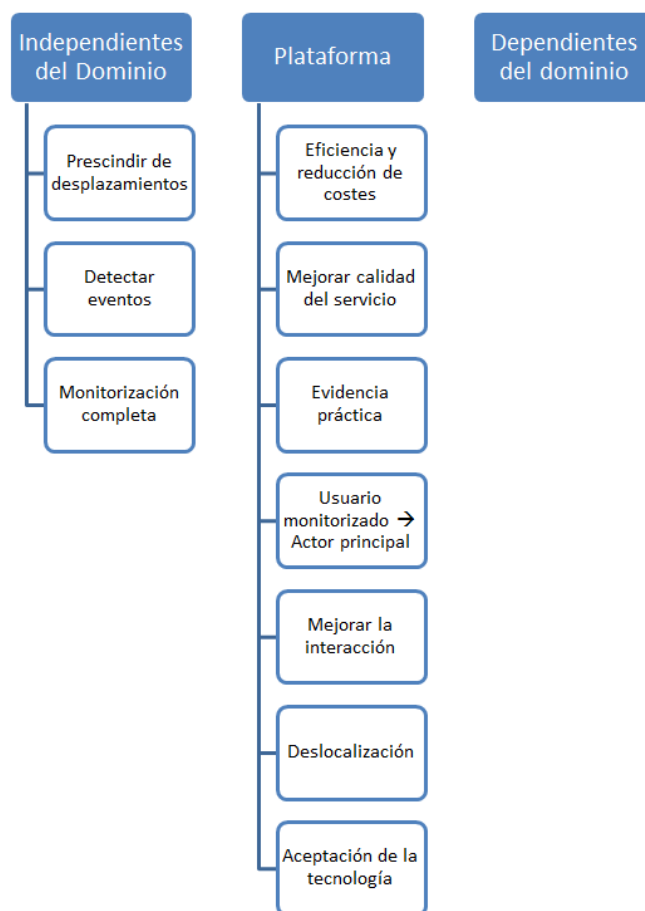


Figura 16 - Objetivos de una plataforma de telemonitorización

2.6. Conclusiones

Las plataformas de telemonitorización son soluciones muy extendidas hoy en día en diferentes ámbitos, siendo especialmente relevante el área de salud y bienestar, donde por su clara aplicación práctica y sus buenos resultados están cada vez más en auge.

Las plataformas de telemonitorización, como cualquier otra solución soportada por las TICs, han evolucionado de acuerdo a la tecnología de la época, pasando de las plataformas basadas en simples ordenadores personales y dispositivos móviles para el envío y recepción de SMS hasta las plataformas basadas en el uso del smartphone, wearables y sistemas de almacenamiento externo (computación en la nube, servidor externo, etc.).

No obstante, aunque las plataformas tienen unos objetivos claros (dependientes del dominio, independientes del dominio y de plataforma), el proceso de desarrollo, y las decisiones que se toman en el mismo, pueden ocasionar que dichos objetivos no se alcancen, ya que dicho proceso de desarrollo es complejo en sí y se presentan numerosos retos que deben abordarse correctamente.

Capítulo 3

Antecedentes

3.1. Introducción

En el capítulo anterior, plataformas de telemonitorización, se han identificado diferentes objetivos (**independientes del dominio y de plataforma**) que suelen estar presentes en las plataformas de telemonitorización: monitorización completa, eficiencia, reducción de costes, interacción entre los distintos actores, etc. Además de estos objetivos, existen otros propios de la plataforma en el ámbito en concreto (**dependientes del dominio**).

La consecución de cada uno de estos objetivos suele acarrear implícitamente un reto, ya sea a nivel de diseño software (garantizar extensibilidad y reusabilidad), implementación (simplificar la integración de *wearables*) o relacionado con un factor no tecnológico (sistematizar el proceso de desarrollo).

Se pretende dar soluciones a todos estos retos a través de e-MoDe, un framework donde se propone una metodología para abordar el de desarrollo de una plataforma de telemonitorización, así como propuestas para agilizar la etapa de diseño software y herramientas para simplificar la etapa de implementación.

e-MoDe se introduce en el capítulo 4, no obstante, en este capítulo se hace una revisión de distintos conceptos y tecnologías que constituyen la base de las distintas propuestas del framework, y que ayudan a entender mejor las razones que justifican la elección de ciertas tecnologías usadas como parte del framework.

La figura 17 muestra los distintos elementos (tecnologías y enfoques) que se abordarán en este capítulo y que serán de soporte para las diferentes soluciones propuestas en el framework: *Cloud Computing* (Computación en la nube), *Wearable Computing* (Computación Wearable), Arquitecturas Software, MDD (*Model-Driven Development*- Desarrollo orientado a modelos), CBSD (*Component-Based Software Development* - Desarrollo basado en componentes), *Mobile Computing* (Computación móvil) y enfoques de diseño para el desarrollo de soluciones software.

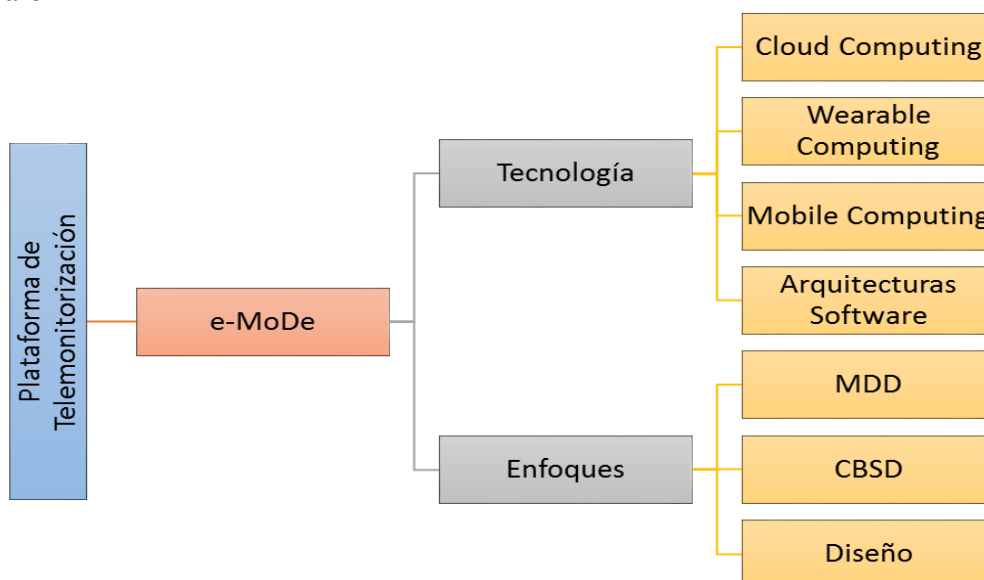


Figura 17 - Tecnologías y conceptos usados en las diferentes propuestas del framework

3.2. Computación en la nube

3.2.1. Introducción

La computación en la nube (*cloud computing*) es un paradigma basado en la oferta de servicios y recursos de computación a través de Internet de forma que la localización de dichos servicios y recursos sea transparente para el usuario [Buyya 2009] [Ambrust 2010] [Mell 2011]. Aunque esta idea y modelo de negocio no es nuevo (como concepto), el término sí es relativamente reciente, donde diversas fuentes citan su acuñamiento a lo largo de 2007 [Wang 2010], mientras empresas materializaban el concepto ofreciendo los primeros servicios en 2006, como Amazon a través de *Amazon Compute Elastic Cloud*

No obstante, más allá del término, el concepto o definición de qué es “la computación en la nube” no es para nada fácil de acotar, existiendo numerosos elementos (software y hardware) involucrados: servicios, virtualización, distribución de carga, seguridad, etc.

Dependiendo del enfoque, cada autor proporciona su propia definición. Por ejemplo, Peter Mell y Timothy Grance del *National Institute of Standards and Technology* (NIST) de Estados Unidos definen la computación en la nube de la siguiente manera, haciendo especial hincapié en el aspecto hardware (compartir recursos) [Mell 2011]:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction

Por el contrario, Michael Ambrust de la Universidad de Berkeley [Ambrust 2010] centra su definición de *cloud computing* en la parte software, y más concretamente en el modelo de servicio:

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services.

Independientemente de la definición en la que nos basemos para intentar determinar qué es la computación en la nube, es posible acotar o especificar objetivamente en qué consiste a través de las ventajas y novedades que aporta [Ambrust 2010], principalmente:

- **Recursos ilimitados:** Gracias a la nube, usuarios y desarrolladores pueden usar tantos recursos como necesiten y cuando necesiten (bajo demanda), como si estos fueran ilimitados. Independientemente del uso, por ejemplo, para almacenar documentos o dar soporte a un sistema con un gran número de usuarios, la nube permite soportar cualquier tipo de sistema haciendo uso de los recursos que sean necesarios.
- **Eliminación de recursos propios:** La computación en la nube permite externalizar servicios que antes eran propios. En efecto, en el pasado, numerosas empresas tenían sus propios servidores y equipo informático para gestionar toda la información, aumentando considerablemente sus gastos en infraestructura al tener que ir actualizando

el hardware cada cierto tiempo. Gracias a la nube estas empresas han podido prescindir de equipos propios reduciendo gastos, contratando servicios a terceros que ofrecen la misma funcionalidad [Kondo 2009] [Marston 2011]. De esta manera, las empresas que podían ir pagando bajo demanda en función de sus requisitos conforme iban creciendo, en vez de optar por la opción tradicional, esto es, aumentar la red de computadoras de la empresa para atender sus necesidades.

- **Modelo “pago por uso”:** Una de las grandes desventajas era alquilar o adquirir de antemano un conjunto de servicios que a posteriori pudieran no usarse o infrautilizarse, con la correspondiente pérdida económica. Gracias a la nube, es posible gestionar más ajustadamente el consumo: servicios, espacio, peticiones, etc. De esta manera, el usuario que usa algún servicio en la nube tiene un control total de sus gastos, pagando justamente por lo que usa.

Estas tres novedades principales que proporciona el uso de la computación en la nube son características (entre muchas otras) que se han conseguido gracias a la arquitectura en la que está basada todo este paradigma.

3.2.2. Arquitectura Cloud

La nube se estructura en torno a un modelo de servicio basado en tres capas o niveles: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS) y *Software as a Service* (SaaS) (Infraestructura como Servicio, Plataforma como Servicio y Software como Servicio).

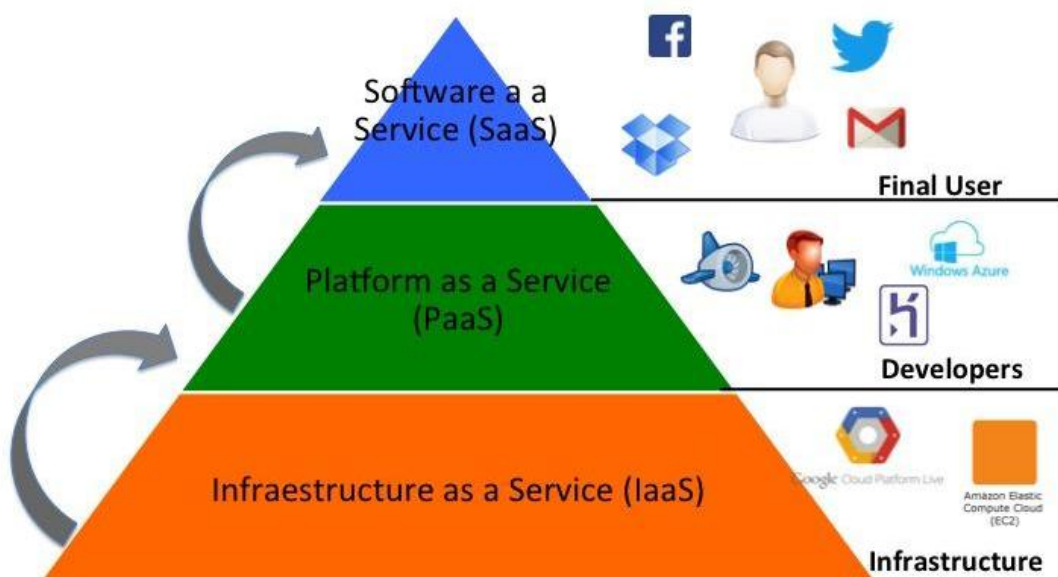


Figure 18 – Modelo de tres niveles para la computación en la nube

El modelo basado en tres capas (ver Figura 18), presenta un modelo estructuras-piramidal donde cada capa está soportada por las capas inferiores. Cada una de estas tiene un propósito en concreto y ofrece unas características concretas, como se detalla a continuación.

Infrastructure as a Service (IAAS)

La capa (o nivel) llamada IaaS (*Infrastructure as a Service* – Infraestructura como Servicio) es la capa inferior y soporta a las otras dos capas. Esta capa representa la parte física de toda aplicación basada en tecnologías *cloud*: servidores, máquinas virtuales, almacenamiento, balanceo de carga, redes, *routers*, *switches*, etc. [Bhardwaj 2010].

El modelo en capas hace esta capa transparente a los desarrolladores y usuarios de las capas superiores (PaaS y SaaS), ya que los distintos tipos de usuarios no tienen que tratar con cuestiones relacionadas con el hardware.

Con una IaaS se consigue no sólo dar soporte físico a sistemas o aplicaciones, sino también monitorizar la gestión de recursos: cuánto almacenamiento se ha usado, número de peticiones por unidad de tiempo, cómo distribuir las peticiones para balancear la carga, etc.

Esta capa permite a los usuarios desplegar soluciones sin necesidad de disponer de una infraestructura propia, sencillamente usando recursos externos de proveedores especializados, en función de su uso. Esto conlleva que el consumidor se olvide de gestionar equipos y su respectivo mantenimiento, limitándose a pagar por los recursos proporcionados por el proveedor.

Además, IaaS permite:

1. Escalabilidad automática, lo que hace posible demandar recursos según las necesidades
2. La posibilidad de tener entornos virtualizados, esto es, simular características físicas mediante el uso de software.

Los primeros pasos (en desarrollo de la tecnología) dentro de la computación en la nube y sobre todo los primeros servicios que se ofrecieron fueron en esta capa/nivel, como puede ser el caso de Amazon (EC2) en 2006 [AmazonEC2]. Otros notables ejemplos actuales de IaaS son Google Cloud [GoogleCloud] o IBM Cloud Computing [IBMCloud].

Platform as a Service (PAAS)

Las soluciones en esta segunda capa, PaaS (*Platform as a Service* – Plataforma como Servicio), tienen como objetivo ofrecer plataformas de servicios para dar soporte al desarrollo y despliegue de aplicaciones por parte de desarrolladores/programadores [Beimborn 2011].

Estas plataformas proveen las herramientas necesarias para crear y desplegar nuevas aplicaciones, así como para desplegar las ya existentes, como pueden ser: entornos de desarrollo, sistemas de gestión de bases de datos, gestión de estadísticas, etc.

Cuando un software se despliega, a través de una plataforma comercial, esta capa permite usar los servicios de la capa inferior (IaaS) para almacenar dicho software, detectar los distintos servicios para determinar el gasto económico al usar cada uno de ellos, etc.

Por ejemplo, un desarrollador de páginas web que sin conocimiento a nivel hardware, puede desplegar su solución en un entorno *cloud* gracias a esta capa, que abstrae totalmente al desarrollador de los detalles de recursos físicos ofreciendo, además, todas las funcionalidades

propias de IaaS ya mencionadas (escalabilidad, distribución de recursos, gestión del almacenamiento, etc).

Ejemplos concretos dentro de esta capa serían Google App Engine [GoogleAppEngine], Heroku [Heroku] y Windows Azure [Azure].

Software as a Service (SAAS)

Esta última capa/nivel corresponde a la capa superior, ofreciendo software (aplicaciones, API) en forma de servicios. Otras aplicaciones o bien directamente los usuarios utilizarán estos servicios a través de aplicaciones concretas (plataforma web, de escritorio, móvil, etc.) [Buxxman 2008] [Choudhary 2007].

En esta capa el proveedor SaaS ofrece el software que los usuarios y otros servicios/aplicaciones consumen directamente. Estos usuarios/aplicaciones consumen dichos servicios sin necesidad de instalar ningún software adicional.

Además, puesto que el software depende del proveedor en vez del consumidor, es el proveedor el encargado de actualizar las distintas versiones del software y de su respectivo mantenimiento. Por lo tanto, los servicios o aplicaciones que consumen los distintos usuarios están siempre actualizados, utilizando así la versión más reciente.

Asimismo, aunque estos servicios para el usuario parezcan funcionar como un único sistema, que siempre accede de la misma manera, gracias al modelo de tres capas estos servicios suelen estar replicados en varios *clusters* o equipos concretos a nivel IaaS para garantizar su robustez y que siempre puedan ser consumidos.

Los principales objetivos y características de esta capa son: reducir costes, nuevas actualizaciones automáticas, acceso deslocalizado y en tiempo real, facilidad de uso y accesibilidad.

Un ejemplo de software en esta capa podría ser el servicio de mensajería de correo de Google (Gmail) que se muestra con varias vistas (interfaces de usuario) dependiendo del dispositivo: plataforma web, dispositivo móvil, cliente de correo entorno, etc. Todos ellos usan los mismos servicios creados por Google y los cambios afectan por igual a todos los clientes. Otros ejemplos notables podrían ser Dropbox, YouTube, una API REST o cualquier red social (Facebook, Twitter, Instagram).

3.2.3. Características

Gracias al modelo de tres niveles comentado en el apartado anterior, la computación en la nube proporciona una serie de características o ventajas que son el principal motivo por el que, tanto usuarios (consumidores), como desarrolladores (proveedores), se deciden a usar soluciones basadas en tecnología *cloud* [Ambrust 2010] [Mell 2011]. No obstante, la computación en la nube también presenta ciertas desventajas, principalmente relacionadas con la seguridad y privacidad de la información.

Ventajas

- **Reducción de costes:** La posibilidad de poder desplegar software sin disponer de una infraestructura propia ha sido uno de los grandes beneficios de la computación en la nube,

principalmente para el sector empresarial. Esto permite usar infraestructuras a medida, a través del modelo “pago por uso” y la virtualización, evitando cualquier coste a posteriori de mantenimiento.

- **Recursos ilimitados:** La percepción del usuario, principalmente desarrolladores, de poder desarrollar o desplegar sistemas sin que el espacio y recursos sean una limitación ha abierto un abanico de posibilidades a toda la comunidad de desarrolladores. Las nuevas soluciones que demandan ingentes (o indeterminadas) cantidades de recursos son ahora posibles gracias a la computación en la nube.
- **Escalabilidad:** Gracias a los servicios de la capa IaaS es posible balancear la carga de trabajo de todos los usuarios de un sistema, permitiendo que un gran número de ellos pueda usar la misma solución software sin que el sistema se vea afectado. Un ejemplo claro podría ser Facebook, donde día tras día se registran nuevos usuarios, pero la calidad del servicio se mantiene.
- **Despliegue:** Gracias a la capa PaaS, que ofrece todo un conjunto de herramientas para el desarrollador, es sencillo desplegar sistemas ya creados o nuevos sistemas basados en la arquitectura *cloud*. La posibilidad de desarrollar y probar directamente soluciones en la nube con la misma facilidad con que se puede hacer en un entorno privado anima a los desarrolladores a desplegar soluciones en la nube.
- **Acceso deslocalizado:** La nube permite acceder a servicios informáticos en cualquier momento, desde cualquier lugar y desde cualquier dispositivo. Este acceso deslocalizado ha sido una característica fundamental que ha dado pie a que los desarrolladores opten por usar la tecnología *cloud*.
- **Fiabilidad:** Gracias al modelo de arquitectura basado en capas, donde la capa inferior (IaaS) está representado por “granjas de ordenadores”, el despliegue de aplicaciones en la nube garantiza la disponibilidad “online” y permanente de dichas aplicaciones. Este nivel de disponibilidad es crucial en muchos contextos donde debe permitirse un acceso continuo a la información o recursos: sistemas de salud, gestión de emergencias, etc.
- **Pago por uso:** Con la nube, el usuario puede controlar su gasto en cuanto a qué espacio está consumiendo, cuántos recursos está usando, el nivel de sus transacciones, etc. Este control permite que el usuario pague a posteriori dependiendo de este gasto, no pagando por recursos que no ha usado como sucede en entornos clásicos.

Desventajas

- **Seguridad:** Este aspecto ha sido uno de los más controvertidos del uso comercial de la tecnología *cloud*, ya que el hecho de almacenar grandes volúmenes información personal en recursos externos y conectados a Internet (es decir, en “la nube”), ha reforzado más la idea de que la seguridad en este contexto es crucial. Por esto, numerosos artículos y autores centran sus esfuerzos en asegurar que una seguridad robusta y eficaz es totalmente necesaria, aunque siempre hay ataques y fugas de información, desafortunadamente [Kandukuri 2009].
- **Privacidad:** Más allá de la seguridad, aparece la cuestión de la privacidad. Muchas soluciones soportadas por tecnología *cloud* tienen un objetivo, o bien social, o bien de negocio, donde los usuarios suben y almacenan su contenido personal a la nube que, en

la práctica, está formada por recursos hardware que pertenecen a otra compañía u organismo. Esta sensación de pérdida de control sobre la información personal, y el desconocimiento sobre qué se hace con dicha información es uno de los grandes aspectos negativos de la nube y el motivo por el que mucha gente es aún reacia a usar ningún software donde no sea propietario de su información.

- **Propenso a ataques:** La centralización de toda la información bajo un mismo marco (un entorno *cloud*) es motivo para que algunos sectores, organismos o grupos intenten acceder a toda esa información personal para obtener un beneficio comercial o de otra índole.

3.2.4. Cloud Computing en plataformas telemonitorización

Las nuevas tecnologías y los nuevos paradigmas de computación extienden su uso más allá del relacionado directamente con la computación, debido a los beneficios que ofrecen. El dar soporte a soluciones de telemonitorización en un entorno *cloud*, debido a las características de la nube, ha abierto un sinnúmero de posibilidades que, tanto compañías, como desarrolladores han visto y aprovechado, desplegando nuevas soluciones o portando las ya existentes a este nuevo paradigma.

Entre los principales motivos por los que se decide desarrollar y desplegar plataformas de telemonitorización en la nube destacan, principalmente:

- **Acceso deslocalizado:** La posibilidad de acceder a cualquier información en tiempo real de un usuario monitorizado posibilita que la interacción y supervisión de dicho usuario por parte de un supervisor mejore considerablemente, al disponer de toda la información existente sobre ese usuario cuando se desee.
- **Mejorar la calidad del servicio:** El poder crear un escenario donde haya una total monitorización las 24 horas del día, ya sea con supervisión o sin ella, ha dado lugar a que la gente tenga conciencia sobre cuán importante es dependiendo de sus objetivos. Por esto, los usuarios usan soluciones software desplegadas en la nube que permiten controlar su actividad diaria o tareas concretas a través de un acceso permanente a toda la información. Además, ello ahorra costes de manera explícita al prescindir de desplazamientos y requerir menos personal.
- **Escalabilidad:** Al igual que con cualquier otro sistema, el poder desplegar un sistema bajo una tecnología y garantizar que su rendimiento no se va alterar es un motivo de peso para usar tecnología *cloud*. En un contexto de telemonitorización, donde el número de usuarios monitorizados es indeterminado y puede ir en aumento, una escalabilidad automática es un requisito fundamental, hoy muy asequible gracias a la nube.

En los últimos años ha habido un auge en la aplicación de tecnología *cloud* en plataformas de telemonitorización, por las ventajas ya mencionadas. Entre todas las posibles aplicaciones, el campo que más se ha beneficiado y crecido es el área de la salud/bienestar, donde numerosos sistemas se han ido creando y desplegando recientemente [Piette 2011] [Calabrese 2015] [Muñoz 2015].

Dentro de esta área, se han desarrollado sistemas para la gestión de datos médicos (formularios, ficheros, bases de datos, etc.), sistemas de telemonitorización que usan

dispositivos móviles o sistemas de rehabilitación basados en reconocimiento de formas, todos ellos articulados en torno a la nube.

Por ejemplo, en [Rolim 2010] se presenta una solución para instituciones de ámbito sanitario que permite gestionar toda la información referente a pacientes: fichas personales, fichas médicas, datos médicos, revisiones, etc.

Además, los resultados de la presente tesis también se han aplicado al desarrollo de plataformas basadas en tecnología *cloud*, como son:

- CloudRehab: Una plataforma de telerehabilitación para pacientes con daño cerebral basada en tecnología *cloud*, dispositivos móviles y sensores de pulso cardíaco [Ruiz-Zafra 2013].
- CloudFit: Una plataforma de ámbito deportivo para gestionar entrenamientos, permitiendo la supervisión de expertos en el ámbito del deporte [Ruiz-Zafra 2014].

3.3. Computación *wearable*

3.3.1. Introducción

La computación “*wearable*” (*wearable computing*) se refiere al uso de *wearables*: dispositivos electrónicos que son colocados en alguna parte del cuerpo y que proveen una serie de funcionalidades concretas [Mann 1997].

Un *wearable* se considera cualquier objeto que se coloque en alguna parte del cuerpo. No obstante, el concepto que se tiene de *wearable* (dentro de *wearable computing*) tiene ciertas características que lo identifican [Watier 2003]:

- Dispositivo electrónico colocado en alguna parte del cuerpo.
- Recopilar información.
- Procesar información: aplicar algoritmos u otros métodos de cálculo.
- Proporciona una interfaz para el intercambio de información.

El término *wearable computing* es relativamente reciente. Dicho término fue creado en la década de los 90 [Watier 2003], siendo Steve Mann durante la década de los 70 el primero en crear un *wearable* tal y como se conocen actualmente. Después de esta primera creación, Steve Mann continuó trabajando en el área de *wearable computing*, creando el grupo de investigación “*Wearable Computer Project*” en el MIT en 1991, y siendo considerado el padre de esta área [Mann 1997]. El uso de *wearables* está cada vez más extendido, siendo usados en numerosas áreas como salud, entretenimiento, proveedor de servicios o incluso en el ámbito militar.

3.3.2. Atributos y características

Según Steve Mann, y tal y como exponía en la primera conferencia de *wearable computing* en 1998 [Mann 1998], los *wearables* presentan seis atributos:

1. **No acaparar la atención.** El hecho de usar *wearables* no implica tener que dedicarle una atención exclusiva. El usuario tiene que poder atender a otros eventos o acciones.
2. **No ser restrictivo.** Permitir al usuario, aunque use varios *wearables*, realizar cualquier acción y poder llevar una vida normal.
3. **Visible.** El usuario debe ser capaz de identificar claramente aquellos objetos que son *wearables* (dispositivos electrónicos) y los que no lo son.

4. **Ajustable.** El usuario debe poder controlar el dispositivo en cualquier momento: apagarlo, encenderlo, cambiarlo de posición, etc.
5. **Sensible al contexto/entorno.** El *wearable* debe mejorar la situación o estado del usuario mediante la información que vaya recopilando (del propio usuario o del entorno) y los servicios que vaya proporcionando.
6. **Comunicativo.** Debe ser posible comunicarse con el *wearable*, ya sea para obtener información o realizar peticiones.

Las características de un *wearable*, algunas deducidas de los atributos anteriores y mencionadas en [Mann 1998], son:

1. **Uso permanente.** Un *wearable* debe estar normalmente encendido y funcionando. Puede tener periodos de inactividad hasta que se lance un evento, de cualquier modo, siempre debe estar disponible para su uso y cumplir su función.
2. **Recolector de información (Logger).** Un *wearable* por lo general recopila información, ya sea sobre las características del sujeto que lo usa o cómo se usa, que será necesario para proveer los servicios para los que está creado.
3. **Liviano y cómodo.** Si un *wearable* está diseñado para ser usado durante largos periodos de tiempo, se considera indispensable que el usuario no sea consciente de que lo lleva puesto, o como mínimo, que no le moleste su uso, por lo que el hecho de que sea ergonómico es crucial.
4. **Batería.** Para garantizar que está funcionando permanentemente, es indispensable que el tiempo de uso (entre carga y carga de batería) sea lo más grande posible.
5. **Fácil de usar.** Se espera de un *wearable* que sea *plug-and-play*, es decir, ponérselo y comenzar a usar.
6. **Compartir el mismo espacio físico.** Mientras en otros escenarios los sensores son meros actuadores en un contexto donde los roles principales van cambiando (sujetos), en *wearable computing* el rol principal siempre es el mismo sujeto que usa los *wearables*, y dichos *wearables* siempre deben ir con el usuario para cumplir su función.

Aunque estos atributos y características se mencionan en [Mann 1998], y aunque en su momento estos atributos cumplían con los requisitos dentro del concepto de *wearable computing*, a día de hoy algunos de ellos no tienen vigencia, debido a la evolución de la tecnología o incluso por demandas del mercado.

Por ejemplo, el atributo 3 (Visible) a día de hoy no siempre se cumple. Los dispositivos son cada vez más pequeños debido a la evolución tecnológica, lo que hace que puedan introducirse pequeños componentes electrónicos en prendas de vestir de tal manera que resulta muy difícil determinar qué es componente textil y qué es componente electrónico. Por ejemplo, en [Yang 2008] se presentan unos calcetines con acelerómetro y giroscopio incorporados donde el componente tecnológico se combina perfectamente con la parte textil para que el sujeto no advierta la presencia de los *wearables*.

3.3.3. Redes Inalámbricas de Área Personal (Wireless Body Area Networks – WBAN)

Normalmente, en un proyecto o sistema basado en *wearable computing*, se suele usar más de un *wearable* simultáneamente, ya que en muchas ocasiones los *wearables* son unifuncionales o incluso, aunque sean multifuncionales, con un único dispositivo no es posible cubrir todos los requisitos que demanda el sistema.

Por ejemplo, en una plataforma de telemonitorización donde se necesite monitorizar datos contextuales, fisiológicos e inerciales (por ejemplo: geoposición, pulso cardíaco y variaciones de aceleración) suele ser muy difícil encontrar un dispositivo que cubra estas tres funcionalidades, que además sea ergonómico y que tenga un precio razonable. En este caso, lo más aconsejable y por operatividad, es usar tres dispositivos independientes donde cada uno proporcione una funcionalidad particular: GPS, banda de pulso cardíaco y acelerómetro.

En estas situaciones, donde se usan varios wearables, se configuran redes de área personal inalámbrica (*Wireless Body Area Networks* - WBAN) [Latré 2011]: un conjunto de *wearables* usados al mismo tiempo sobre una misma persona, donde cada uno de ellos ofrece una funcionalidad específica (por ejemplo, registro del pulso cardíaco, geoposición), pero que entre todos dotan al sistema de las funcionalidades requeridas para cumplir los objetivos. La figura 19 representa un ejemplo de una WBAN.

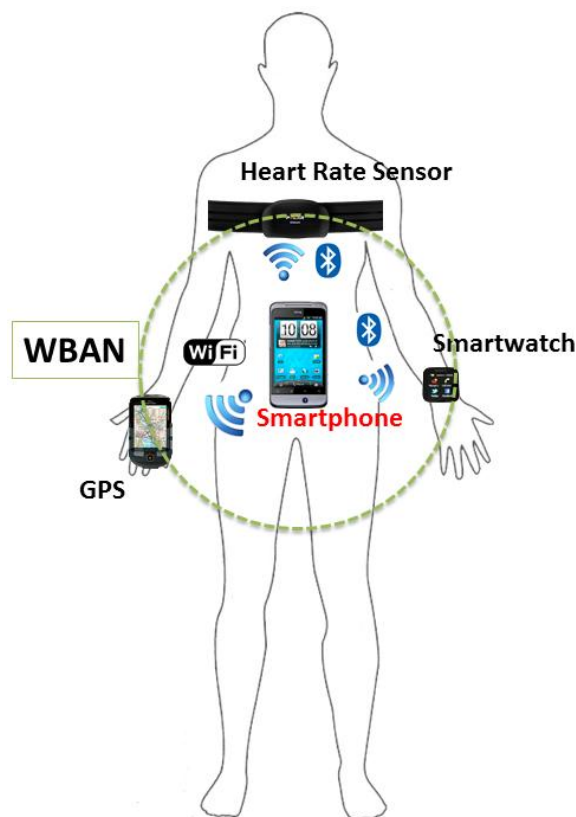


Figure 19 – Ejemplo de una Red Inalámbrica de Área Personal (WBAN)

En las WBAN, los dispositivos se suelen enmarcar dentro de tres posibles tipos [Latré 2011]:

- Tipo 1 - **Nodo sensor**: Es un dispositivo que se encarga de recopilar, procesar y enviar la información. En el contexto de las plataformas de telemonitorización, con este tipo de dispositivos se obtiene información fisiológica del usuario monitorizado (temperatura, pulso cardíaco), así como información del contexto. Un ejemplo podría ser un sensor de pulso cardíaco, un GPS o un sensor de temperatura ambiental.
- Tipo 2 - **Nodo actuador**: Es el dispositivo encargado de actuar dependiendo de un evento. Este dispositivo, al igual que el nodo sensor, recibe información de los sensores, la procesa y actúa.

- Tipo 3 - **Dispositivo personal**: Este dispositivo se encarga de recolectar toda la información de todos los sensores y también se le conoce como Unidad de Control del Cuerpo (*Body Control Unit – BCU*).

Estos dispositivos están basados en diversas tecnologías, proporcionando diferentes funcionalidades y usando diferentes protocolos de comunicación (WiFi, Bluetooth), por lo que, aunque dos proyectos se basen en redes WBAN, no siempre es fácil la integración de una red, o parte de ella, en otra ya existente.

Las WBAN están cada vez más extendidas, principalmente en el ámbito de la salud/bienestar, donde se usan numerosos dispositivos para crear sistemas de monitorización que permitan supervisar a un usuario las 24 horas del día, con el objetivo de mejorar su calidad de vida o detectar posibles problemas de salud (en tiempo real o bien analizando a posteriori los datos obtenidos de los *wearables*).

Tomando como base los tres tipos de categorías de dispositivos identificados en WBAN, la mayoría de las redes aplicadas al ámbito de la salud se basan principalmente en los dispositivos tipo 1 (sensor) y 3 (unidad de control) donde los sensores de tipo 1 suelen ser bandas de pulso cardíaco, sensores para determinar el electrocardiograma, el nivel de oxígeno, etc., y el sensor de tipo 3 suele ser un smartphone, por sus prestaciones, potencia y usabilidad.

3.3.4. Wearable Computing en plataformas telemonitorización

Aunque el uso de *wearables* se ha extendido a muchas áreas, es especialmente relevante la popularidad que están alcanzando en plataformas de telemonitorización, donde se han convertido prácticamente en dispositivos indispensables.

Obtener datos fisiológicos, inerciales, contextuales, etc., de usuarios monitorizados de manera poco intrusiva y sin necesidad de realizar desplazamientos facilita no sólo la obtención de datos en cualquier momento, sino la fiabilidad de los mismos, ya que son procesados automáticamente y presentados al personal correspondiente (supervisores), con diferentes diseminaciones.

Por ejemplo, [Gong 2011] presenta un proyecto para gestionar el pulso cardíaco y su procesamiento usando *wearables*. El proyecto presentado en [Jovanov 2005] presenta un sistema basado en una red WBAN donde se usan *wearables*, tanto para parámetros fisiológicos (pulso cardíaco u oxígeno), como inerciales (locomoción y la cantidad de movimiento) (ver Figura 20).

Otros ejemplos de proyectos son [Ruiz-Zafra 2013], donde se usan sensores de pulso cardíaco para determinar a partir de estos el nivel de estrés de pacientes con daño cerebral en una plataforma de telerehabilitación o el proyecto presentado en [Ruiz-Zafra 2014], donde se usan sensores fisiológicos, inerciales y contextuales para medir diversos parámetros de atletas.

Además, el uso de *wearables* dentro del ámbito de la salud/bienestar también se aplica con objetivos relacionados con la investigación. Por ejemplo, los trabajos presentados en [Ruiz-Zafra 2014b] [Ruiz-Zafra 2015] usan distintos *wearables*, en este caso acelerómetros, para medir la cantidad de movimiento de una actividad física. Partiendo de esta medida, se puede

determinar no sólo la intensidad de la actividad física, sino la cantidad de movimiento, lo que puede ser útil para la promoción de hábitos saludables, principalmente en grupos poblacionales como personas mayores.

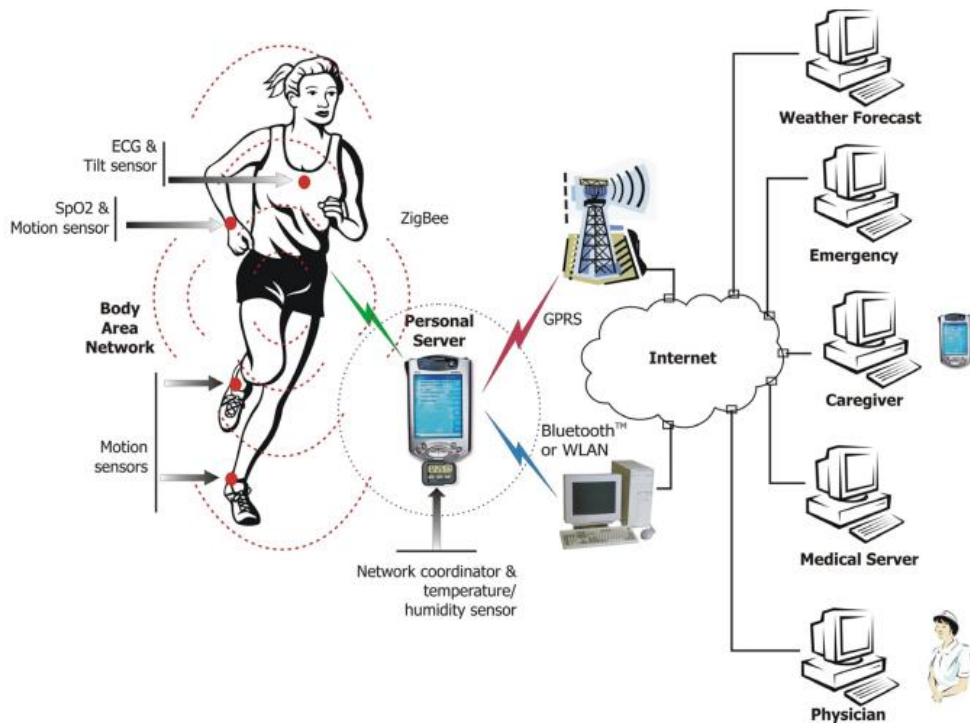


Figura 20 - Arquitectura del sistema con WBAN presentada en [Jovanov 2005]

3.4. Computación móvil

3.4.1. Introducción

La computación móvil se define **como el uso de dispositivos electrónicos independientemente del espacio físico donde estemos (transportable), gracias a su autonomía y al recurso de las redes inalámbricas (WiFi, 3G, 4G)** [Forman 1994]. Estos dispositivos electrónicos permiten acceder a servicios para proporcionar funcionalidades o mantener a los usuarios conectados entre sí.

Este tipo de computación, por la creciente red de infraestructuras para la conexión a Internet, unido al abaratamiento de costes en la producción de dispositivos, es probablemente uno de los campos más en auge dentro de las Ciencias de la Computación, donde su exponente más representativo, el smartphone, ha copado con su uso a la mayoría de la población (consumidores).

3.4.2. Características

Dentro de la computación móvil hay un rango extenso de clasificaciones: computadores portátiles, teléfonos móviles, teléfonos inteligentes, *wearables*, tablets, etc.

Cada grupo de estos suele usarse con un fin: trabajo, ocio, servicios, etc. No obstante, todos ellos comparten características comunes que son las que determinan que todos se engloban dentro de la computación móvil.

Las características (y atributos) más destacadas de la computación móvil son:

- **Movilidad.** Los usuarios que poseen un dispositivo móvil pueden desplazarse sin limitaciones, al ser este dispositivo parte de su equipaje como una prenda de vestir u otro objeto (llaves, cartera, etc.). El uso del dispositivo en cualquier momento les permite en tiempo real acceder a los servicios que este ofrece.
- **Alcance y ubicuidad.** El usuario puede usar el dispositivo en cualquier momento y en cualquier lugar.
- **Comodidad.** Por sus características físicas y potencia del dispositivo, su uso resulta es prácticamente constante a lo largo del día por los usuarios.
- **Conectividad.** Estos dispositivos suelen estar conectados permanentemente a Internet, permitiendo así una total interconectividad con otros usuarios o el consumo de servicios.
- **Localización de productos y servicios.** Gracias al conocimiento de su localización física, es posible ofrecer servicios y recursos dependiendo de esta localización.

No obstante, también existen una serie de limitaciones, entre las que destacan las siguientes [Satyanarayanan 1996]:

- **Limitación de recursos.** La ventaja de que sea por lo general sean dispositivos de reducido tamaño y fácilmente portable, tiene la desventaja de que esto limita los recursos, en almacenamiento y potencia, que pueden ofrecer con respecto a dispositivos estáticos o “no móviles”. No obstante, gracias al avance de la tecnología esta barrera cada vez se va superando, hasta el punto de que muchos dispositivos móviles actuales son ordenadores en cuanto a potencia y capacidad.
- **La movilidad no siempre es una ventaja.** Aunque por su reducido tamaño es fácilmente portable, este tamaño también implica que se puede dañar, perder o robar más fácilmente.
- **La conectividad es variable.** Aunque cada vez más esto no es un problema, gracias al 3G y 4G, hubo épocas donde, fuera de edificios con conexión a Internet, el dispositivo móvil no tenía ningún tipo de conectividad, sobre todo en el exterior.
- **Limitación energética.** Dispositivos de reducido tamaño cada vez con más potencia y recursos, demandan cada vez de más cantidades de energía. Aunque la batería cada vez va mejorando, ampliando su uso diario, es cierto que estos demandan de recargas diarias.

3.4.3. Computación móvil en plataformas telemonitorización

La computación móvil es un modelo de computación que tiene una gran aceptación en plataformas de telemonitorización, principalmente por el uso de dispositivos móviles para la realización de las distintas tareas de telemonitorización y las características que estos proporcionan: comodidad, movilidad, etc.

Aunque el uso de *wearables*, ya comentada en el epígrafe anterior, forma parte de la propia computación móvil, se pretende mostrar la importancia del smartphone como pieza clave en las plataformas de telemonitorización y como actor fundamental en las propuestas de la presente tesis.

Los principales motivos por los que el smartphone es un elemento crucial (a nivel tecnológico y de funcionalidad), en el framework propuesto en esta tesis, son:

- **Realización de tareas de telemonitorización.** Parte fundamental de cualquier plataforma de telemonitorización es la realización de tareas o actividades por parte de los usuarios monitorizados, como para de la propia monitorización. Para esto, el smartphone es un dispositivo que por su potencia, costes, comodidad y eficiencia permite realizar tareas a través de la interacción hombre-máquina (juegos, tutoriales, uso de apps). El smartphone es un elemento fundamental con el que interactuará día a día el usuario monitorizado.
- **Coordinador de wearables.** El uso de *wearables* es de uso común en plataformas de telemonitorización. No obstante, estos *wearables* permiten conectarse a ellos para obtener algún tipo de funcionalidad y muchos de ellos no permiten almacenar información al no disponer de memoria propia. El smartphone serviría como dispositivo conector/coordinador de todos los *wearables* que se usen con las aplicaciones de la plataforma de telemonitorización.
- **Mediador/Conector de tecnologías.** El smartphone sirve de mediador entre las diferentes tecnologías, ya que permite, tanto conectarse a *wearables*, como conectarse a servicios de almacenamiento externo (*cloud* o servidor). Esta característica hace que la información recuperada de un *wearable*, por ejemplo, sea posible guardarla en un servidor externo gracias al smartphone, que permite ser el intermediario entre estas dos tecnologías.

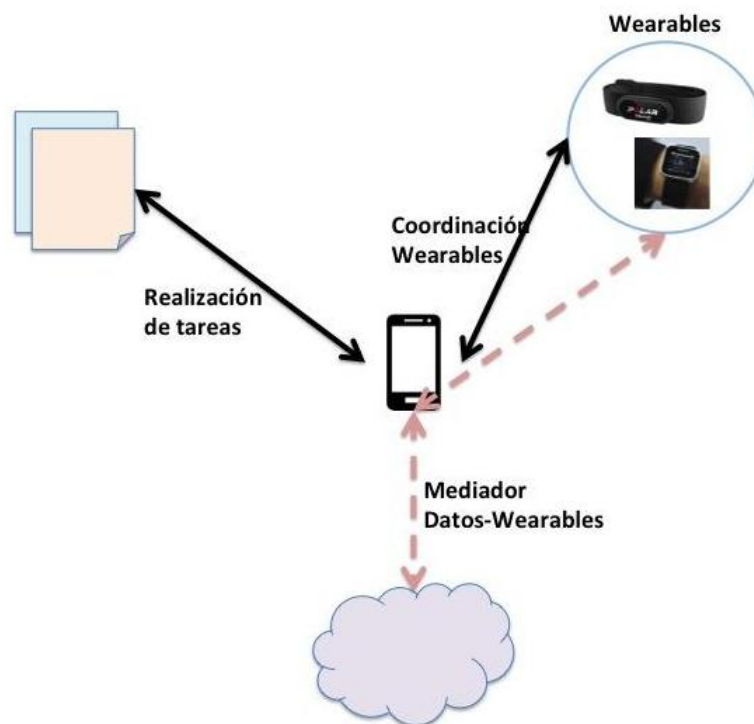


Figura 21 - Usos del smartphone en una plataforma de telemonitorización

Estos tres motivos hacen del smartphone una pieza clave tanto para las plataformas de telemonitorización como parte fundamental del framework propuesto en la presente tesis (ver Figura 21).

3.5. Arquitecturas Software

3.5.1. Introducción

Una plataforma de telemonitorización pretende representar, a través de las TICs, un protocolo de telemonitorización. Estos protocolos de telemonitorización, definidos por supervisores y expertos en el ámbito concreto, representan diferentes procesos donde el usuario monitorizado realiza ciertas tareas y el supervisor observa el resultado para determinar su evolución.

Los protocolos suelen variar dependiendo del ámbito, del tipo de usuario monitorizado, de los recursos disponibles, de la experiencia o conocimiento de los supervisores o expertos, etc. Debido a esto, es posible que un protocolo, inicialmente definido por supervisores y posteriormente desarrollado, cambie por algún tipo de restricción (recursos) o bien como mejora por el resultado previo de un estudio o investigación. Esto implica que, dependiendo de los cambios en el protocolo, la plataforma de telemonitorización debe adaptarse para representar los cambios en el nuevo protocolo.

Partiendo de esta versatilidad, es imprescindible usar tecnología que permita en un futuro dar soporte a nuevas funcionalidades y requisitos, con el objetivo de que cambios en un protocolo de monitorización puedan ser soportados en la plataforma de telemonitorización, permitiendo así modificar funcionalidades o añadir nuevas.

3.5.2. Características

Considerando el estudio previo planteado en el capítulo 2, plataformas de telemonitorización, de los objetivos identificados (*independientes del dominio, de plataforma y dependiente del dominio*) y de esta versatilidad del protocolo de telemonitorización, se deben asegurar ciertas características que no solo harán posible el soporte a una solución de telemonitorización, sino también facilitarán la labor de los programadores. Estas características son:

- **Interoperabilidad.** La evolución de la tecnología hace que nuevos dispositivos de muy diversas características formarán parte de una plataforma de telemonitorización: dispositivos móviles de diferentes plataformas, ordenadores personales con aplicaciones de escritorio, plataformas web, etc. Es crucial que, independientemente de las características de los dispositivos, estos sean capaz de comunicarse entre ellos o bien con el servidor, permitiendo así que exista una interacción completa entre los distintos tipos de usuarios.
- **Extensibilidad.** La extensibilidad garantiza que nuevas funcionalidades puedan ser añadidas al sistema sin perjudicar su rendimiento y funcionamiento. Esto es un requisito importante para dar soporte a nuevas necesidades que puedan definirse en un futuro de acuerdo a criterios del ámbito concreto en el que se enmarque la plataforma de telemonitorización.
- **Escalabilidad.** Aunque no es un requisito imprescindible, ya que muchas soluciones de telemonitorización son dedicadas y están desarrolladas para pequeños grupos

poblacionales, o incluso grupos concretos para estudios determinados, si es una característica a tener en cuenta en soluciones de ámbito comercial o con una gran proyección, con el objetivo de dar soporte a un número elevado, pero no determinado, de usuarios sin afectar al rendimiento del sistema.

De acuerdo a estas características, hay dos modelos de arquitectura software que además de encajar con estas características son adecuadas para las tecnologías ya mencionadas, estas son las Arquitecturas Orientadas a Servicios (SOA – *Service Oriented Architecture*) y las Arquitecturas Orientadas a Recursos (ROA – *Resource Oriented Architecture*).

3.5.3. Arquitecturas Orientadas a Servicios (SOA)

Las Arquitecturas Orientadas a Servicios (*Services Oriented Architecture* – SOA) son un modelo de arquitectura software para desplegar soluciones de sistemas distribuidos [Papazoglou 2007] [Erl 2005].

Este modelo de arquitectura tiene como elemento clave el concepto de *servicio*: una unidad o elemento software independiente, autocontenido, sin estado y con una funcionalidad en concreto, que teniendo una interfaz bien definida acepta peticiones devolviendo una respuesta [Bianco 2007].

Estos servicios, como concepto, se pueden orquestar (orquestación de servicios) para proporcionar funcionalidades compuestas o bien componer (composición de servicios) para crear nuevos servicios. Los servicios no dependen de ningún protocolo o tecnología en concreto, son meras funcionalidades proporcionadas por proveedores y usadas por consumidores (sistemas, aplicaciones, o incluso otros servicios).

Gracias al concepto de servicio, con una arquitectura orientada a servicios se consiguen una serie de principios o características entre los que destacan [Papazoglou 2007] [Erl 2005]:

- **Descripción de servicios.** A través de una interfaz bien definida es posible consumir servicios, sabiendo cómo consumirlos y que proveen.
- **Acoplamiento débil.** Los servicios son independientes, esto hace que sea posible el despliegue en cualquier sistema de un servicio en concreto.
- **Sin estado.** Los servicios no consideran estados anteriores para producir resultados. Cada petición y uso de un servicio se realiza exactamente igual que cualquier otra, tomando unos datos de entrada y produciendo unos resultados dependiendo de estos datos de entrada.
- **Abstracción.** El uso de servicios, a través de su interfaz (API), abstrae totalmente la implementación del mismo, facilitando su uso y ahorrando tiempo.
- **Composición de servicios.** Es posible crear nuevos servicios o funcionalidades a través de otros servicios.

Además del concepto de servicio, otros elementos son imprescindibles para determinar la estructura de una SOA. Según [Krafzig 2005] los distintos elementos que caracterizan una SOA son los que corresponden con la figura 22.

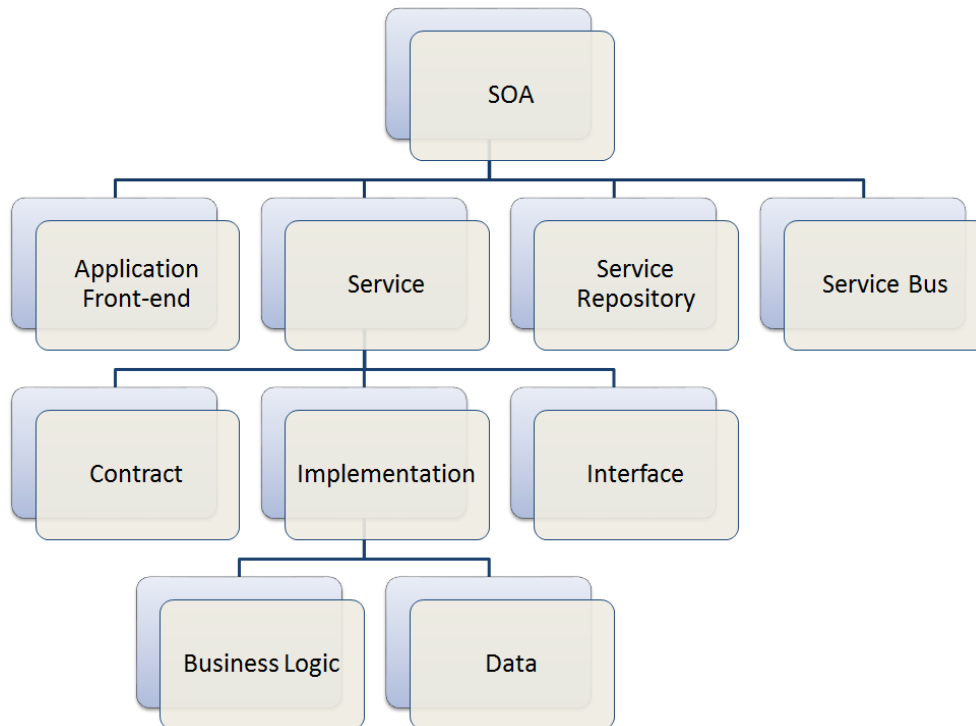


Figura 22 - Elementos de SOA (Service Oriented Architecture)

Una SOA está formada por los ya mencionados Servicios (*Service*), por un repositorio donde almacenarlos (*Service Repository*), una manera de consumirlos (*Service Bus*) y una aplicación para mostrarlos (*Application Front-end*). Estos servicios, además de tener su implementación concreta con una interfaz bien definida, debe tener una especificación estándar y clara para formalizarlo lo máximo posible, teniendo así el servicio descrito de una manera independiente de su implementación y pudiendo implementarse usando otras tecnologías (contrato – *contract*).

Debido a que SOA está alineado totalmente con procesos de negocio, ya que es una solución muy apta para empresas por su reducción de costos y facilidad de proveer servicios, todo servicio de una SOA representa un proceso o producto dentro de una lógica de negocio. Esta lógica de negocio, implementada y haciendo uso de los datos necesarios, es la que conformará el servicio en sí.

En la definición o especificación de SOA no está definido ni la tecnología en la que se basa un servicio (implementación), ni donde se almacenan (repositorio) ni cómo se consumen o usan (bus).

Un ejemplo, y probablemente uno de los más conocidos, es SOAP (*Simple Object Access Protocol*), un protocolo estándar que permite el intercambio de información/objetos mediante XML y extendido para la implementación de servicios web, que es sin duda la implementación más conocida de SOA [Curbera 2002].

Es posible implementar SOAP bajo cualquier protocolo, siendo HTTP el más común para su implementación como servicios Web y WSDL (*Web Services Description Language*) como el lenguaje más común de especificación para definir estos servicios.

Debido a su modelo basado en el intercambio de objetos, casi todos los lenguajes soportan SOAP y facilitan su implementación en lenguajes concretos, permitiendo el intercambio de información en servicios implementados con diferentes tecnologías.

Como aspecto negativo a destacar, SOAP es un protocolo pesado en cuanto a la cantidad de información que se intercambia al usar sus servicios, ya que además de los resultados esperados hay información adicional como especificación del protocolo (Figura 23). Este hecho, unido al auge de los dispositivos móviles los cuales demandan transacciones livianas, hace que SOAP, aunque sea un estándar y siga estando en vigencia y uso, esté perdiendo terreno con respecto a otras tecnologías o implementaciones concretas.

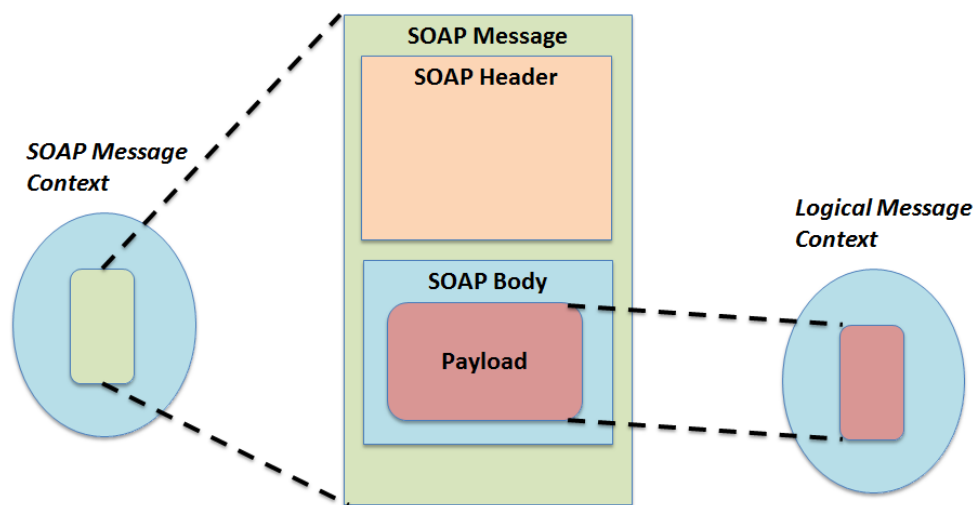


Figura 23 - Estructura de un mensaje SOAP

3.5.4. Arquitecturas Orientadas a Recursos (ROA)

Una arquitectura orientada a recursos (*ROA – Resources Oriented Architecture*) es un modelo de arquitectura software orientada a proporcionar recursos con una interfaz REST (*Representational State Transfer*) [Guinard 2010].

Al igual que en las arquitecturas orientadas a servicios el elemento fundamental era el servicio, en este marco es el *recurso*: entidad software que representa un objeto y que puede ser consumido a través de una diseminación concreta de su estado. De acuerdo a esto, y considerando que los principios RESTful se basan en el protocolo http, el acceso a estos recursos se hace a través de una URI, por ejemplo www.dominio.com/recurso/id (www.prueba.com/user/24) [Richardson 2008].

Estos recursos poseen un identificador único y global. A diferencia de un servicio, un recurso si tiene un estado interno, obteniendo la representación del mismo con un modelo de datos concreto: texto plano, XML, HTML, JSON.

A diferencia de SOA, donde todo estaba centrado en servicios, que podría traducirse como funciones, en una ROA todo se centra en recursos, siendo su homólogo el concepto de objeto, por lo que no es posible definir funcionalidades. Por esto, en ROA se permiten cuatro funcionalidades básicas para alterar el estado público de un recurso, que coinciden con cuatro de los verbos del protocolo HTTP.

- **Create (Crear).** Permite crear un nuevo recurso usando el verbo POST. Esta creación puede ser a partir de un estado previo o un recurso completamente nuevo y con un estado por defecto.
- **Read (Leer).** Con esta operación, a través del verbo GET, se recupera una representación del estado del recurso en el momento en el que se solicita bajo una notación concreta (JSON, XML, texto plano, una imagen, un vídeo, etc).
- **Update (Actualizar).** Esta operación permite modificar el estado de un recurso usando el verbo PUT.
- **Delete (Eliminar).** Eliminar un recurso en concreto a través del verbo DELETE.

Este modelo, llamado *CRUD*, posibilita que todas las opciones de gestión sobre el recurso se realice usando el mismo identificador (por ejemplo, www.prueba.com/recurso/4), pero realizando distintas operaciones dependiendo del verbo que se use cuando se realice la petición (*GET, POST, DELETE, PUT*).

Al definir de esta manera una arquitectura orientada a recursos y considerando que toda la infraestructura web sigue unos principios orientados a recursos, la implantación de una solución REST para proporcionar servicios es considerablemente más sencilla que una SOA tradicional (o SOAP concretamente), ya que REST se adapta a la tecnología y conceptos ya establecidos y usados [Fielding 2008] (ver Figura 24).

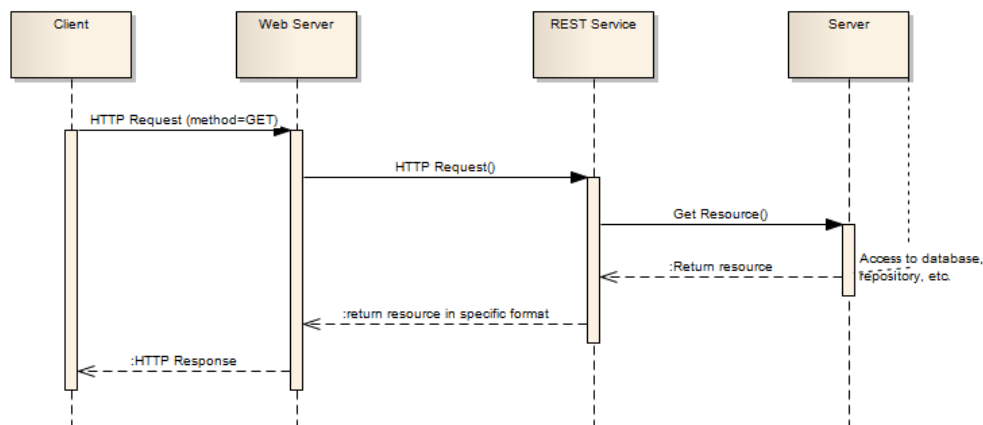


Figura 24 - Ejemplo de consumo de un recurso (REST)

Un modelo de arquitectura que sigue los principios REST presenta, entre otras, las siguientes características:

- **Rendimiento.** La interacción con los recursos es muy simple, siendo así más fácil y eficiente el acceso a los mismos
- **Escalabilidad.** Permite añadir nuevos recursos sin alterar el comportamiento del sistema.
- **Simplicidad.** Debido a que ROA se basa en el acceso a recursos a través de una URI usando la infraestructura de la web y el protocolo http, el acceso a estos recursos resulta muy simple para desarrolladores, lo que simplifica el desarrollo en sí.

Esto hace posible que cualquier dispositivo, sin necesidad de soportar un protocolo concreto (como SOAP) puede acceder a usar un servicio basado en REST a través de HTTP usando cualquier software, independientemente de la plataforma que lo consuma.

Las arquitecturas REST están teniendo un gran auge tanto por desarrolladores particulares como por grandes empresas, quienes publican su API de servicios siguiendo un modelo RESTful, en vez de otras opciones como SOAP o servicios Web (XML-RPC).

No obstante, muchas de las APIs relacionadas con sistemas no son ni íntegramente REST (ROA) ni SOAP/XML-RPC (SOA), sino que son híbridas, en el sentido de que siguen una estructura y tecnología REST al seguir el modelo de la arquitectura web de acceder a recursos a través de URI, pero permiten la adición de parámetros para realizar filtros o crear nuevas operaciones/funcionalidades concretas. Por ejemplo, si tenemos un sistema donde queremos obtener los datos de un usuario con el identificador 5, los distintos modelos de acceso serían:

XML-RPC

`www.dominio.org/getUser.php?id=5`

REST

`www.dominio.org/user/5`

Híbrido V1

`www.dominio.org/user?id=5`

De esta manera con el enfoque híbrido se obtiene el mismo resultado, pero realizando la petición siguiendo un enfoque XML-RPC.

También puede suceder el caso contrario, usando peticiones XML-RPC con los parámetros adecuados, la ejecución de ese servicio puede estar programado para gestionar un nuevo recurso, por lo que usando a través del protocolo GET se permitirían la creación o eliminación de recursos. Esta opción teóricamente va en contra de los principios REST, pero a nivel práctico es realmente útil para desarrolladores y diseñadores, habilitando nuevas maneras de describir funcionalidades.

Híbrido V2

`www.dominio.org/user?method=crear&name=aa&lastname=bb`

3.5.5. Arquitectura Software en plataformas de telemonitorización

Una implementación concreta para dar soporte a una plataforma de telemonitorización se podría hacer siguiendo varios modelos y tecnologías. El uso de la tecnología adecuada con las guías de diseño apropiadas son claves para garantizar características como extensibilidad, escalabilidad y robustez (objetivos de la plataforma).

De las dos arquitecturas ya comentadas (SOA y ROA), cualquiera de las dos podría ser válida (con sus pros y contras) para dar soporte a una plataforma de telemonitorización, esto es, que sería posible implementar una plataforma de telemonitorización siguiendo cualquiera de estas arquitecturas.

No obstante, en las plataformas de telemonitorización es indispensable y básico el uso de dispositivos electrónicos inalámbricos, siendo estos actualmente y en la mayoría de los casos dispositivos inteligentes (smartphones). De hecho, la mayoría de las plataformas de telemonitorización actuales suelen representar escenarios basados en el uso de smartphone, cómo se ha podido ver en los ejemplos mostrados en el capítulo 2.

Además del uso de dispositivos inteligentes, las plataformas de telemonitorización están basadas en varias aplicaciones (para el usuario monitorizado, para los supervisores, etc) por lo que son diversas aplicaciones las que tienen que interactuar entre ellas, en este caso a través de un servidor externo. Esto implica que el acceso a las funcionalidades debe ser sencillo, para permitir a los desarrolladores crear nuevas soluciones que sean fácilmente integrables.

Por esto, el modelo que más se adecúa a una plataforma de telemonitorización es un modelo de arquitectura orientada a recursos, basado en los principios REST, por su fácil implantación y acceso.

Aunque es el que más aproximado para dar soporte a sistemas de este tipo, la limitación del modelo a las cuatro operaciones básicas (CRUD) podría complicar el desarrollo en cuanto a la necesidad de tener muchos tipos de recursos distintos para representar una plataforma de telemonitorización.

Un modelo híbrido (SOA-ROA), que ofrece la facilidad de uso e implementación de una arquitectura REST junto con un acceso de servicio web es la mejor opción de cara a soportar cualquier posible función, además de añadir nuevas funcionalidades bajo un mismo recurso en caso de que fuese necesario, y posibilitando así que un recurso además de ser recurso pueda ser un servicio. Se considera la propuesta más adecuada por:

- **Acceso fácil a los recursos.** La simplicidad para acceder a un servicio/recurso (a través de una URI) junto con su fácil consumo (HTTP) garantiza la facilidad para consumir servicios.
- **Facilidad de integración.** Para desarrolladores, es mucho más simple entender cómo funciona REST a cómo funciona SOAP, por lo que la puesta en marcha de una aplicación que use una API REST es mucho más rápida que una que use una API SOAP.
- **Rendimiento-eficiencia.** El intercambio de datos, gracias a nuevos estándares como JSON, posibilita que no solo la petición consume muy pocos recursos, si no que la respuesta de la misma está optimizada para devolver solo información relevante (*payload*), sin ser necesario cabeceras u otra información irrelevante. Esta reducción del gasto es importante en sistemas móviles con recursos limitados.
- **Extensibilidad.** En caso de que el sistema de telemonitorización requiera nuevas funcionalidades, se pueden añadir nuevos recursos que implementan funcionalidades concretas, o bien añadir nuevos parámetros de especificación a recursos ya creados, con el objetivo de crear dicha funcionalidad.

En el capítulo 6, sección 1, se extiende la descripción de esta arquitectura híbrida y como su aplicación se puede materializar para plataformas de telemonitorización.

3.6. Desarrollo dirigido por modelos (*Model-Driven Development – MDD*)

3.6.1. Introducción

El desarrollo de un sistema siguiendo un enfoque clásico suele constar de una serie de etapas más o menos estructuradas, que comienzan con la descripción del sistema y que concluyen con el despliegue del mismo [Larman 2003].

Este enfoque de desarrollo clásico se basa en la idea de construir soluciones a problemas concretos lo más interoperables y extensibles posibles con el objetivo de facilitar las tareas de mantenimiento. Algunos ejemplos de este enfoque clásico podrían ser el desarrollo en cascada o un desarrollo en espiral [Larman 2003].

La principal desventaja de este enfoque es que cambios en una de las etapas tienen repercusión en las sucesivas etapas. Es decir, un simple cambio en una etapa ocasionará cambios en el resto de etapas que en muchas ocasiones (sobre todo a nivel de implementación) son costosos de abordar, lo que incrementará el tiempo de desarrollo.

Una alternativa a estos enfoques tradicionales o clásicos son los enfoques dirigidos por modelos (*Model-Driven Development - MDD*) [Selic 2003].

Estos enfoques pretenden, a través de modelos (representación abstracta de conceptos de un dominio concreto [Poole 2001]) y transformaciones (proceso de convertir un modelo de un sistema en otro modelo del mismo sistema [Truyen 2006]), pasar de una esquematización de un sistema a la creación del mismo, generando diferentes modelos a través de las reglas de transformación, hasta llegar al código fuente o aplicaciones del sistema.

Las principales ventajas de MDD con respecto a un modelo tradicional son [Selic 2003]:

- **Abstracción.** El poder modelar un sistema usando un lenguaje de especificación de alto nivel, como UML, permite abstraer al ingeniero y modelar el sistema de una manera mucho más simple. Esto posibilita que, con las herramientas adecuadas, y con el proceso lo suficientemente automatizado, no se requerirían grandes conocimientos para generar nuevas soluciones software, puesto que sería sencillamente trabajar con conceptos a alto nivel.
- **Automatización.** El objetivo de MDD es ser capaces de generar software para su uso a partir de modelos a alto nivel. A través de distintas herramientas se permite hacer transformaciones de modelos desde el más alto nivel de abstracción hasta el código fuente, todo de manera automática.
- **Estándares.** Gracias a los estándares existentes, ya sea por organismos como OMG u empresas privadas, el concepto de generar soluciones a través de modelos se puede materializar usando herramientas concretas, que, al ser de uso común, permite una interoperabilidad de soluciones entre dos propuestas distintas.

3.6.2. Transformación de modelos

Uno de los principales retos de los enfoques orientados a modelos es cómo abordar, a nivel tecnológico, el reto de permitir generar a partir de una especificación a alto nivel (modelo conceptual representado en un estándar) el software final (código fuente o aplicación), a través de sucesivos pasos que van generando diferentes modelos.

Para esto, se realizan procesos llamados de transformación de modelos, a partir de los cuales y tomando el modelo original, se realizan las transformaciones necesarias y usando la tecnología adecuada, hasta llegar al producto final.

Como soporte para este proceso de transformación existen varios estándares, siendo el más relevante MDA (*Model-Driven Architecture*), una iniciativa de la OMG (*Object Management Group*) para dar soporte a MDD [Selic 2008].

MDA se caracteriza por estar formados por modelos y transformaciones, dividiendo los modelos en diferentes niveles, y establecer procesos de transformación entre ellos (ver Figura 25). Estos modelos son [Kleppe 2003]:

- **CIM** (*Computation Independent Model* - Modelo Independiente de la Computación). Es el modelo que representa el dominio y la lógica de negocio, siendo por lo tanto independiente de la de la tecnología (computación)
- **PIM** (*Platform Independent Model* - Modelo independiente de la plataforma). Modelo que representa una descripción de la funcionalidad del sistema de forma independiente de las características de la implementación.
- **PSM** (*Platform Specific Model* - Modelo específico de plataforma). Modelo para la descripción del sistema en términos de una plataforma concreta, como Java o .Net.

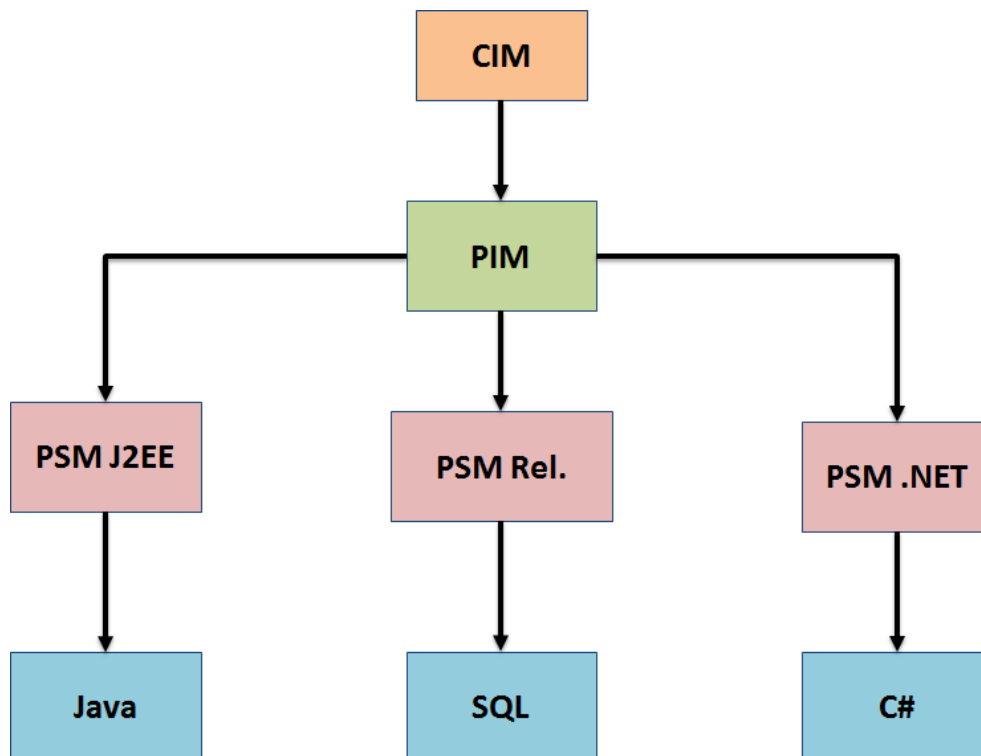


Figura 25 - Ejemplo de la transformación entre modelos en MDA

Dentro de MDA existen varias herramientas para la creación de modelos (UML, MOF), para la transformación de modelos (QVT, MOF) y para definir procesos (SPEM).

Además de MDA, existen otros estándares para la generación de modelos como puede ser XSLT (*EXtensible Stylesheet Language*) presentado por W3C, donde con unos datos de entrada (hoja de estilo) y un modelo se permite generar otro, por lo que, aunque no hay un alto nivel de abstracción como en MDA, si se mantiene la filosofía de generar unos modelos a partir de otros gracias a transformaciones.

3.6.3. MDD en plataformas telemonitorización

Debido a que un enfoque orientado por modelos es aplicable en cualquier contexto para el que se desarrolle una herramienta software: servicios, educación, ocio, salud, aplicaciones móviles, aplicaciones de escritorio, etc. también es aplicable al desarrollo de plataformas de telemonitorización.

No obstante, abordar el desarrollo de plataformas de telemonitorización usando un enfoque orientado a modelos no es el objetivo principal de presentar esta tecnología en este capítulo de antecedentes.

El uso concreto de la teoría del desarrollo dirigido por modelos ha sido aplicado concretamente en una de las soluciones a los retos que se plantearán: mejorar la integración y uso de *wearables* (abordado en el capítulo 7).

Existen un gran abanico de *wearables*, ya sea por la funcionalidad o funcionalidades que proporcionan, así como por la tecnología que se usa (heterogeneidad). De la misma manera, el modo de interacción con cada uno de ellos depende de un fabricante a otro, ya que son estos quienes deciden qué información envía el *wearable* y cómo la envía (orden de los datos, longitud, etc.) (falta de estandarización en el acceso).

Esto hace que el uso de cada uno de estos requiere de un proceso de desarrollo particular, donde el programador debe crear, bajo el estudio para adquirir los conocimientos necesarios, el código para poder usarlo.

La propuesta de usar en MDD para este problema parte de simplificar la implementación para programadores, definiendo un metamodelo que representa todas las características identificadas en los *wearables* para representar sus características y cómo interactuar con estos.

El objetivo es, a partir de esta metamodelo, y usando XSLT, generar modelos para los diferentes *wearables* (un modelo por sensor), donde se representan las características del *wearable* que permitirán, con las herramientas adecuadas, interactuar con él.

De esta manera, un usuario con las herramientas adecuadas genera un modelo para un *wearable*, a partir del metamodelo general, y un programador usando las herramientas necesarias usa dicho modelo para usar el *wearable*.

3.7. Desarrollo orientado a componentes

3.7.1. Introducción

El desarrollo orientado a componentes es un proceso de construcción o creación de nuevo software basado en el concepto de componente software.

Según [Szyperski 1995] un componente software se define como: **“A unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.”**

Esta definición persigue identificar al componente software al mismo nivel que se tiene en otras áreas, como en el ámbito industrial: elemento que proporciona unas funcionalidades concretas, se puede usar independientemente y se puede intercambiar cuando sea necesario por otro que proporcione las mismas funcionalidades, no afectando así al funcionamiento del sistema.

Este modelo de desarrollo de software surge como evolución a la programación orientada a objetos por las deficiencias de esta:

- Dificil reutilización de objetos
- No es fácilmente distribuible y empaquetable
- No separa aspectos computacionales y composicionales

Estas desventajas, unidas a la cada vez más aceptación de los sistemas distribuidos, hicieron que durante la década de los 80 la computación orientada a componentes software se postulara como una propuesta viable para el desarrollo de software.

Debido a esta necesidad de modularizar e independizar funcionalidades software, surgieron numerosas plataformas y soluciones comerciales y estándar, entre las que destacan:

- **CORBA** (*Common Object Request Broker Architecture*). Es un estándar definido por la OMG (*Object Management Group*) para facilitar el intercambio de información entre distintos tipos de sistemas basado en objetos [Siegel 2000].
- **Java RMI**. Tecnología perteneciente a Java para el intercambio de información entre distintas aplicaciones distribuidas basadas en Java [Pitt 2001].
- **DCOM** (*Distributed Component Object Model*). Plataforma creada por Microsoft para el intercambio de información a través de componentes software en plataformas Windows [Brown 1998].

Estas plataformas, usadas durante la década de los 80 y 90, están cada vez más en desuso debido al auge del concepto de servicio. Todas las funcionalidades que se implementan a través de componentes software, derivaron en modelos concretos de arquitectura (SOA) y con diferentes paradigmas (computación en la nube), ya que el consumo de servicios a través de estas nuevas tecnologías es más simple que a través de componentes software (por ejemplo: servicios web).

3.7.2. Características

A partir de la definición de [Szyperki 1995], un componente software se puede considerar como tal si cumple tres características fundamentales:

- *is a unit of independent deployment;*
- *is a unit of third-party composition;*
- *has no (externally) observable state*

Además de estas características, a un componente se le presuponen otro tipo de atributos como robustez, eficiencia, bien documentado, etc.

3.7.3. Desarrollo orientado a componentes en plataformas de telemonitorización

La programación orientada a componentes se suele aplicar como buena praxis en el desarrollo de soluciones software, con el objetivo de reutilizar código en un futuro y que la adición de nuevas funcionalidades (extensibilidad) sea más rápido y funcional.

Una plataforma de telemonitorización debe adaptarse a lo largo del tiempo a cambios en el protocolo de telemonitorización, esto es, a nuevos requisitos y características para cumplir los objetivos esperados y proporcionar nuevas funcionalidades.

En el caso de las plataformas de telemonitorización la aplicación de un proceso de desarrollo basado en componentes software puede aportar numerosas ventajas, entre las que destacan:

- Añadir nuevos componentes software para dar soporte a nuevas funcionalidades que representen cambios en una metodología de trabajo, sin alterar la estructura del código de la aplicación.
- Cambiar componentes software sin alterar el comportamiento del sistema para el usuario.
- Reusar componentes software. Aunque las plataformas de telemonitorización sean muy dinámicas, soluciones con distintos objetivos comparten parte de la funcionalidad al seguir un mismo enfoque y arquitectura, por lo que parte del código de una solución podría reutilizarse fácilmente en otra solución usando componentes software.
- Las plataformas de telemonitorización suelen dividir todas las funcionalidades en varias categorías: almacenamiento, interacción con el almacenamiento exterior, interacción con los dispositivos, interacción con el usuario. El desarrollo y soporte de estas funcionalidades englobadas en componentes software permite categorizar fácilmente el software, que a su vez hace que sea más reusable y más fácilmente portable o modificable.

En la propuesta de esta tesis, el desarrollo orientado a componentes software tendrá especial relevancia en el desarrollo de las aplicaciones móviles que forman parte de una plataforma de telemonitorización.

3.8. Revisión de enfoques para el diseño y desarrollo

El proceso de desarrollo de una plataforma de telemonitorización no dista mucho del desarrollo de otra solución software en otro contexto o ámbito, ya que tanto las tecnologías como modelos para el desarrollo están automatizados.

Aunque el diseño y desarrollo suele ser similar para la mayoría de las soluciones software, el enfoque con el que se aborda dicha fase de creación es crucial para garantizar que la solución cumpla con los objetivos.

Roger, Sharp y Preece [Preece 1994] presentan un enfoque clásico para el diseño y desarrollo de soluciones que podría ser utilizado para el diseño y desarrollo de plataformas de telemonitorización. Este enfoque, representado en la figura 26, está compuesto por 3 fases:

1. Identificación del problema (requisitos/requerimientos)
2. Proponer, analizar y elegir una solución
3. Evaluar la solución implementada

Roger, Sharp y Preece proponen que en la segunda tarea (proponer, analizar y elegir una solución) se creen prototipos de la solución que se vayan validando en cada iteración, ya que el proceso de diseño y desarrollo que ellos proponen es iterativo, concluyendo una vez que la evaluación de la solución es satisfactoria para tener un producto final estable y usable. De esta manera, se irían depurando los requisitos, que darían pie a nuevas soluciones (prototipos) que se irían validando iterativamente hasta conseguir el producto final deseado.

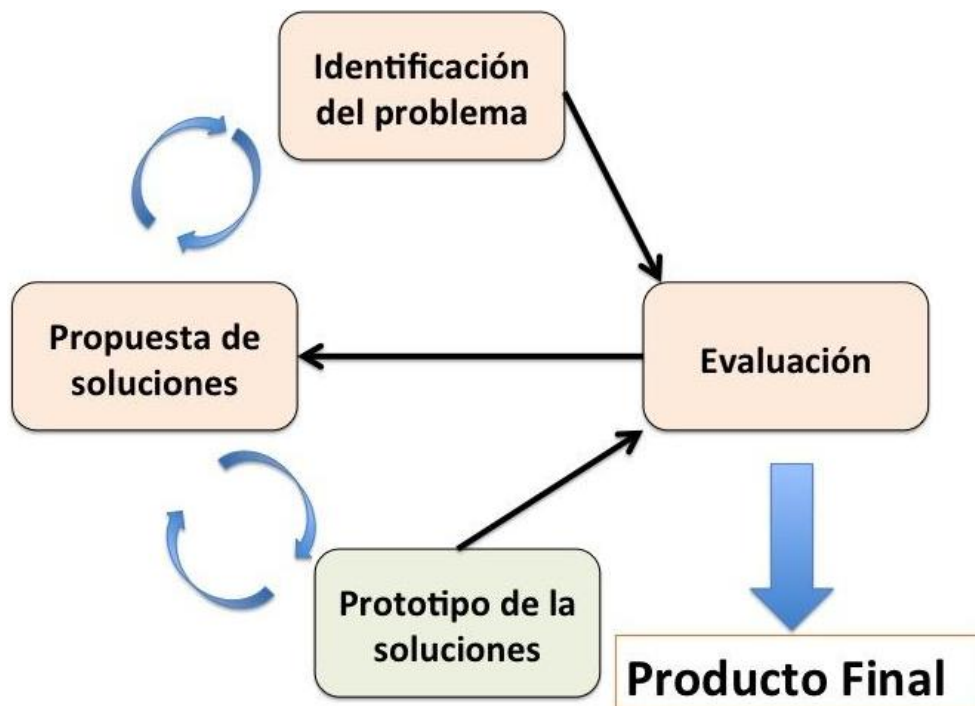


Figura 26 – Enfoque clásico para el diseño propuesto por Roger, Sharp y Preece

Aunque este enfoque es muy usado en la metodología clásica, y podría ser aplicable a plataformas de telemonitorización no es una buena estrategia por varias desventajas:

- La evaluación se centra únicamente en asegurar su usabilidad en términos de diseño, por lo que los mecanismos de evaluación del ámbito concreto quedarían fuera de esta metodología.
- Tanto en la identificación del problema como propuesta de soluciones no intervienen roles externos. Debido a esto, en ningún momento se le consulta al usuario monitorizado

o supervisor sobre el sistema, pudiendo sufrir carencias una vez el producto final llega a ser usado por los usuarios.

- El enfoque se centra en el sistema, siendo este el actor principal.

Además de este enfoque clásico, hay otros enfoques para el diseño y desarrollo de sistemas que podrían aplicarse a para el desarrollo de plataformas de telemonitorización. Por ejemplo, Dan Saffer recopila en [Saffer 2009] cuatro enfoques distintos: centrado en el usuario, centrado en el sistema, centrado en acciones/actividades y centrado en conocimiento experto. La tabla 1 resume los distintos enfoques.

Diseño	Orientación	Acciones principales
Centrado en el Usuario	Centrado en las características y necesidades del usuario.	Obtener las necesidades del usuario y sus características, para que el producto satisfaga todas las necesidades.
Centrado en las Acciones/actividades	Todo el proceso de diseño se realiza de acuerdo a las acciones/tareas que realiza el usuario	Detectar las distintas acciones o tareas llevadas a cabo por el usuario, así como su comportamiento, para determinar un diseño más correcto.
Centrado en el sistema	Centrado en la funcionalidad propia del sistema. Estas funcionalidades son el núcleo para la fase de diseño.	Determinar los requisitos y restricciones del sistema.
Centrado en el conocimiento	El encargado del desarrollo es el que posee el conocimiento necesario para llevar a cabo el diseño, sin necesidad de colaboración o feedback de ningún tipo.	Usar la experiencia del desarrollador para el diseño íntegro del producto.

Tabla 1 – Enfoques para el desarrollo [Saffer 2009]

Cualquiera de estos enfoques podría usarse para el diseño de nuevas soluciones, independientemente del área en el que se enmarque, aunque ciertamente hay enfoques que encajan mejor dependiendo del área, por las características y requisitos de esta.

Para soluciones de telemonitorización, donde se van a crear soluciones que van a usarse a diario por personal que puede ser ajeno externo al ámbito de las TIC, por ejemplo, un enfoque centrado en el sistema no suele ser lo más adecuado.

El centrarse únicamente en las funcionalidades del sistema sin tener en cuenta al usuario final (monitorizado) o supervisores en el proceso de creación, puede dar lugar a soluciones de baja usabilidad y pocas funcionalidades.

La mayoría de plataformas de telemonitorización deberían estar basadas en un enfoque centrado en el usuario. El uso de este enfoque va a garantizar, al menos, que el producto resultante va a cumplir al menos las necesidades de los usuarios en cuanto a usabilidad. En un ámbito donde son usuarios no relacionados con las TICs y que en muchos casos no tienen conocimientos básicos, es una ventaja importante.

Por ejemplo, [Teixeira 2012] estudia la ingeniería de requisitos en sistemas de la salud con un enfoque centrado en usuarios, aplicando los resultados en el campo de la hemofilia.

[Butz 2006] diseña y desarrolla un sistema mediante un enfoque centrado en el usuario para automatizar y gestionar historiales clínicos, con la colaboración personal experto hospitalario.

Aunque por lo general, y según la literatura, las soluciones de telemonitorización se basan en un enfoque centrado en el usuario, el aporte de la presente tesis, el cual se describe en el capítulo 4, se basa en un enfoque híbrido, esto es, uso del enfoque centrado en el usuario y las actividades.

3.9. Conclusiones

A lo largo de la presente tesis se presenta una propuesta de framework para abordar el desarrollo de una plataforma de telemonitorización, presentando diferentes propuestas que ayudan y simplifican el proceso de desarrollado, tanto para los encargados del mismo (ingenieros) como programadores, a través de herramientas software.

Estas propuestas están soportadas por diferentes tecnologías y enfoques, que de entre los existentes, se consideran la opción más adecuada por sus características y ventajas.

Tecnología como la computación *cloud*, computación móvil, *wearables* o bien enfoque para el desarrollo o programación orientada a componentes etc. son puntos cruciales que serán la base de los distintos aspectos que se presentan como las distintas partes de la propuesta de la tesis.

Capítulo 4

Framework e-MoDe

4.1. Introducción

El concepto de framework es ambiguo, al existir diferentes tipos de frameworks dependiendo de su ámbito de aplicación: ciencias de la computación, ámbito legal, educación, etc. En el ámbito de las ciencias de la computación existen, a su vez, diferentes tipos de frameworks, dependiendo de la aplicación u objetivo que tengan: framework de arquitectura [Linington 2011], EA framework (*Enterprise Architecture Framework*) [Urbaczewski 2006], framework de software, donde se enmarcarían los frameworks de aplicación, orientados a objetos, etc. [Mattson 1996] [Fayad 1997] [Butler 2001].

De los diferentes tipos de framework y definiciones, la que más se adapta a la propuesta de framework presentada a continuación es la de *framework de software*, entendiendo este como: **un conjunto de artefactos de diseño e implementaciones que permite generar soluciones concretas en un dominio particular** [Riehle 2000] [Johnson 1988] [Fayad 1997] [Campbell 1992] [Lewis 1995] [Fayad 1999]. Esto es, dentro del ámbito de la telemonitorización, proporcionar los elementos necesarios (conceptos, prácticas, herramientas) para permitir generar soluciones de telemonitorización concretas.

e-MoDe es un framework para el desarrollo de plataformas de telemonitorización que tiene como objetivo principal sistematizar y simplificar dicho desarrollo, gracias a los distintos elementos que lo componen (ver Figura 27):

- **Metodología:** El objetivo es dar soporte al proceso de desarrollo de una plataforma de telemonitorización, desde la primera reunión con los distintos actores participantes hasta el despliegue y validación de la plataforma, a través de cinco etapas e identificando las características, objetivos y resultados de cada etapa.
- **Modelos:** Herramientas a usar (por los ingenieros) en la etapa de diseño de la metodología y que permiten solucionar algunos retos presentes en el desarrollo, asegurando la consecución de algunos de los objetivos, como por ejemplo soporte a la interacción entre los distintos actores de la plataforma.
- **Herramientas software:** Herramientas usadas por los desarrolladores/programadores en la etapa de implementación para simplificar el desarrollo y garantizar algunos objetivos como reusabilidad y extensibilidad de las soluciones móviles o la integración de *wearables*.

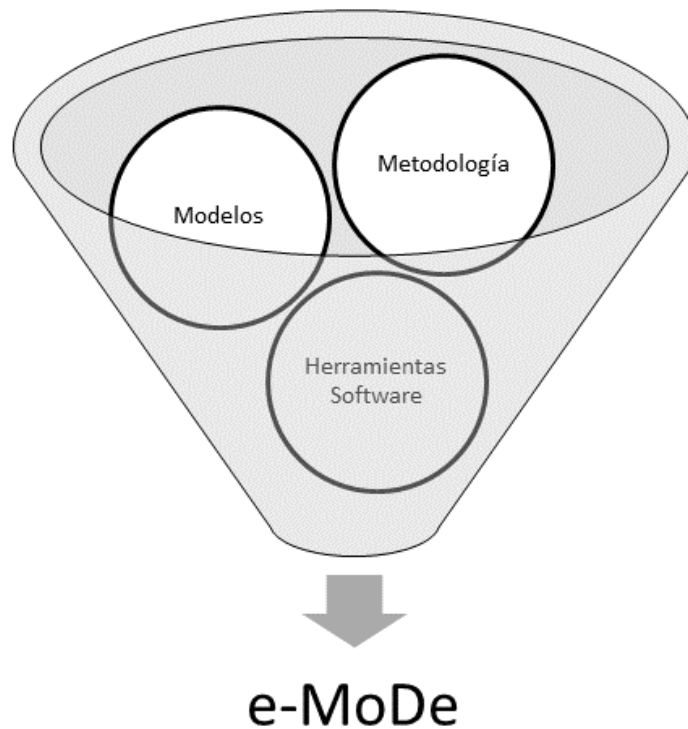


Figura 27 - Elementos que conforman e-MoDe

El framework está orientado a cubrir los diferentes retos presentes en el proceso de desarrollo y que estén relacionados con los objetivos ya mencionados en el capítulo 2:

- **Objetivos independientes del dominio** [Rice 2011]:
 - Minimizar desplazamientos, al ser posible realizar los procesos de telemonitorización desde cualquier localización.
 - Notificar eventos o situaciones concretas, como por ejemplo resultados de tareas de telemonitorización o relacionados con el estado de un usuario monitorizado.
 - Monitorización completa y continua, esto es, una supervisión remota integral (24 horas al día) del usuario monitorizado.
- **Objetivos de plataforma:**
 - Eficiencia y reducción de costes, al realizar la mayoría de los procesos a través de las TICs, en vez de usar mecanismos más tradicionales (papel, reuniones presenciales, desplazamientos)
 - Mejorar la calidad de los procesos de telemonitorización, esto es, hacer uso de las TICs para, por ejemplo, automatizar la gestión de información o interacción.
 - Evidencia práctica, mostrando mediante estudios o resultados concretos como una plataforma es útil como soporte a la telemonitorización.
 - Satisfacer los requisitos de usuario en cuanto a usabilidad y funcionalidad.
 - Mejorar la interacción entre los distintos actores (supervisores/expertos y usuarios monitorizados).
 - Extensibilidad para permitir añadir nuevas funcionalidades o características.
 - Reusabilidad para a partir de una plataforma de telemonitorización, usar elementos comunes en otras plataformas: conexión concreta con la nube, uso de un wearable, etc.

- **Objetivos dependientes del dominio:** Son objetivos que dependen del ámbito en el que se esté utilizando la plataforma como, por ejemplo, determinar el pulso cardíaco cada cierto tiempo o registrar alguna actividad concreta. La mayoría de estos objetivos corresponden con las tareas de telemonitorización.

La satisfacción de estos objetivos genera desafíos/retos para los actores involucrados en el desarrollo de la plataforma, como, por ejemplo:

- Constante comunicación entre los diferentes usuarios de la plataforma para garantizar una telemonitorización completa.
- Dar soporte tecnológico para la detección de situaciones concretas y notificarlas correctamente.
- Diseñar la plataforma y sus componentes de forma abierta, para permitir que la plataforma se adapte a futuros cambios.
- Asegurar, en la medida de lo posible, la reusabilidad de todo el software que se cree para permitir reusar funcionalidades en nuevas plataformas de telemonitorización.
- Realizar correctamente la captura de datos.
- Integrar cualquier dispositivo (*wearable*) en una plataforma de telemonitorización para acceder a los datos capturados, así como a las opciones del mismo.
- Definir y dar soporte a entornos colaborativos para que haya una correcta interacción entre los distintos actores de la plataforma con objetivo de mejorar el propio proceso de telemonitorización

La figura 28 muestra un esquema que pretende transmitir esta idea, de cómo a partir de los objetivos definidos se generarán unos retos para satisfacer dichos objetivos. Estos retos pretenden solucionarse gracias al framework presentado en este capítulo, y extendido en capítulos posteriores, a través de los diferentes elementos que lo componen.

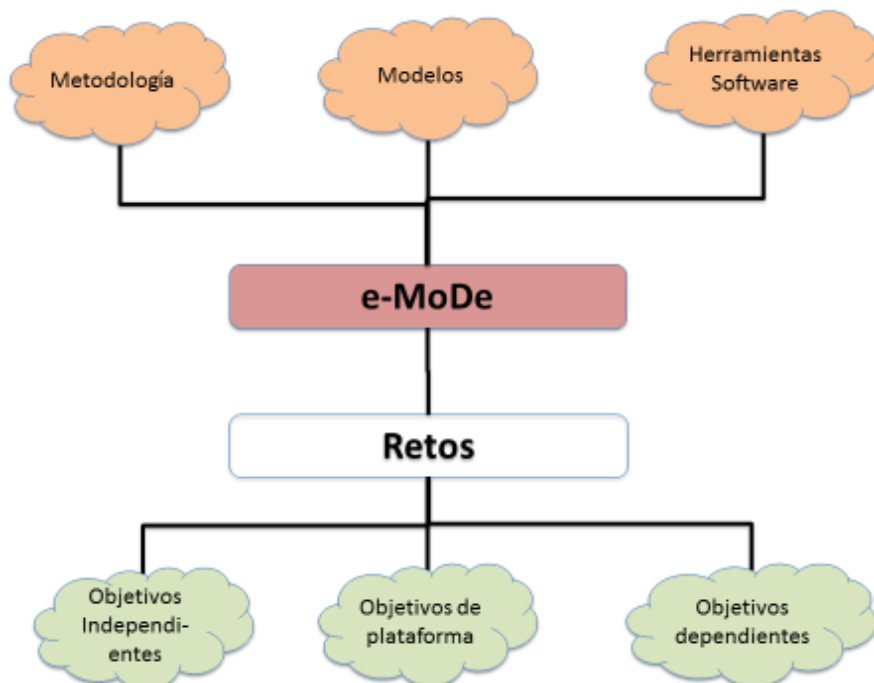


Figura 28 - Esquema general del objetivo de la propuesta

4.2. Consideraciones preliminares de la propuesta

El framework está fundamentado en dos elementos o conceptos principales: (1) el enfoque para el diseño y desarrollo adoptado en el framework para plantear el desarrollo; (2) el modelo de arquitectura o visión de conjunto de una plataforma de telemonitorización.

El enfoque para el desarrollo es crucial en este contexto, donde los servicios que perciben los usuarios monitorizados son de especial importancia para una telemonitorización eficiente y eficaz.

Basar toda la propuesta en un enfoque determinado permitirá que se preste especial atención a determinados aspectos de la plataforma y, por lo tanto, la consecución de ciertos objetivos no estaría garantizados como, por ejemplo, imponer aspectos de usabilidad al mismo nivel de importancia que cualquier funcionalidad de la plataforma.

Con respecto al modelo de arquitectura, revisando los trabajos expuestos en el capítulo 2 y la mayoría de las plataformas recientes de la literatura, se puede observar que la mayoría sigue un modelo de arquitectura bastante similar, donde se pueden identificar claramente los distintos elementos del modelo: dispositivos móviles, uso de computación en la nube u otro soporte de almacenamiento externo, aplicaciones móviles, wearables, servicios web, etc.

La elección de un modelo de arquitectura que permita representar a alto nivel escenarios de telemonitorización asegura que, la mayoría de plataformas de telemonitorización podrían implementarse siguiendo dicho modelo, teniendo así las características propias del modelo como facilidad de uso, implementación y mantenimiento.

Estas dos propuestas, que se desarrollan a continuación, **enfoque para el desarrollo y modelo de arquitectura**, son propuestas que son la base de los distintos elementos del framework (modelo, metodología y herramientas software), y cualquier decisión de diseño o cualquier propuesta planteada tiene como base la consideración de dichas propuestas.

4.2.1. Enfoque para el desarrollo

4.2.1.1. Introducción

Como se describió en el capítulo tres (antecedentes), existen diferentes enfoques para abordar el desarrollo de soluciones software: centrado en el usuario, en el sistema, en las acciones/actividades y en el conocimiento [Saffer 2009]. Dependiendo del enfoque, se hace especial hincapié en aspectos determinados, como por ejemplo usabilidad, privacidad o funcionalidades del sistema.

La decisión sobre qué enfoque tomar no es un aspecto crucial en otro tipo de soluciones (aunque sí recomendable), ya que muchos programadores/ingenieros no siguen explícitamente un enfoque para el desarrollo. No obstante, en el ámbito de las plataformas de telemonitorización las posibles limitaciones de los usuarios monitorizados se traducen en requisitos a cubrir por la plataforma de telemonitorización.

El tipo de usuario monitorizado puede variar de una plataforma a otra, aunque normalmente suelen compartir una característica en común: la limitada capacidad de utilización de las TICs y barreras en la propia telemonitorización por las características del usuario [Boyne 2009].

Este hecho dificulta a nivel práctico, por lo general, el uso de las TICs, y por lo tanto de la plataforma, por lo que el énfasis en la usabilidad y funcionalidad de las aplicaciones de la plataforma debe ser mayor que para otro tipo de sistemas.

Además de esta limitada capacidad de utilización de las TICs, los problemas que ello conlleva, y la especial atención que necesita el usuario monitorizado, hay otro aspecto fundamental: el conjunto de procesos que permiten alcanzar el objetivo para el que está desarrollada la plataforma (por ejemplo, mejorar la calidad de vida un paciente, controlar su estado de salud las 24 horas del día, detectar problemas cardíacos, etc.).

Estos procesos se suelen agrupar en las denominadas tareas de telemonitorización, por lo que la correcta gestión y desarrollo de estas determinará si la plataforma es funcional o no.

Para el desarrollo de una plataforma de telemonitorización, y partiendo de los objetivos ya mencionados (independientes del dominio, de plataforma y dependientes del dominio), se hace necesario elegir un enfoque que garantice que se tiene una especial atención a las necesidades del usuario (en términos de usabilidad y funcionalidad) y en el propio proceso de telemonitorización, esto es, las tareas de telemonitorización (o actividades que realiza el usuario monitorizado).

4.2.1.2. Objetivos

Dentro de los grupos de objetivos planteados (independientes del contexto, de plataforma y dependientes del contexto) existen distintos objetivos que podrían enmarcarse por áreas, como, por ejemplo, objetivos desde el punto de vista del desarrollador (extensibilidad, reusabilidad) o propios de la metodología de telemonitorización (monitorización constante, minimizar desplazamientos).

Entre los objetivos planteados hay un especial interés en el bienestar del usuario monitorizado, así como la calidad de los servicios que percibe, al ser este el principal actor de la plataforma y el que necesita (por sus posibles limitaciones) de una especial atención para conseguir que dicha plataforma de telemonitorización sea útil y usable.

Al igual que parte de los objetivos tecnológicos se solucionan tomando las decisiones oportunas de diseño o con la creación de nuevo software, para asegurar la consecución de objetivos relacionados con el usuario monitorizado se propondrá un enfoque que garantice que estos objetivos van a considerarse durante el desarrollo de la plataforma y que tendrán que cumplirse una vez concluya el desarrollo de la plataforma.

Para ello, de esta lista de objetivos (independientes del dominio, de plataforma y dependientes del dominio) se derivan los siguientes requisitos cuya satisfacción garantiza, en cierta medida, la calidad del servicio que percibe el usuario monitorizado:

- **Usabilidad.** Los usuarios monitorizados no tienen por qué estar familiarizados con las TICs. De hecho, muchas soluciones están enfocadas en grupos poblacionales donde ya se presupone una falta de conocimiento o cultura tecnológica (personas mayores, usuarios con problemas cognitivos). Esta falta de conocimiento o pericia en el manejo de las soluciones TICs debe suplirse por parte de los diseñadores y desarrolladores. Por esto, la usabilidad del sistema es crucial para garantizar que cualquier usuario sea capaz de usarlo de manera intuitiva [Kaufman 2003] [Finkelstein 1992].

- **Ergonomía.** Los usuarios van a disponer de herramientas o aplicaciones de telemonitorización con las que pueden llegar a interactuar un gran número de horas al día. Estas herramientas deben ser fáciles e intuitivas en su uso, por lo que deben estar diseñadas para que su uso continuo no resulte pesado al usuario.
- **Robustez.** Las distintas soluciones software que conformen la plataforma de telemonitorización deben ser robustas en cuanto a la gestión de los posibles fallos. Un usuario de una plataforma de telemonitorización que tenga que estar lidiando diariamente con fallos derivados de una mala tarea del programador dejará de usar el sistema. Esta robustez se presupone en cualquier solución software, pero en una plataforma de telemonitorización de uso continuado, incluso de hasta 24 horas al día, que dicha plataforma sea robusta es un requisito fundamental, no sólo por la percepción del usuario, sino porque podría interferir en el objetivo de la propia monitorización: fallos en la captura de datos, fallo en el envío o recepción de información, pérdida de datos, datos erróneos, etc.
- **Privacidad.** Un sistema de este tipo puede gestionar información sensible, al estar gestionando datos personales del usuario monitorizado. El correcto funcionamiento del sistema es necesario de cara al usuario, que lo percibe como un instrumento efectivo, así como de cara a los profesionales del sector, a quienes les resulta una herramienta útil para poder llevar a cabo sus tareas relacionadas con su ámbito concreto [Layouni 2009].
- **Transparencia.** Muchos sistemas de monitorización funcionan en *background* las 24 horas del día realizando ciertas tareas de captura de datos, que posteriormente se envían a soportes externos para consulta o análisis. Es importante que esta captura de datos sea automática y transparente para el usuario monitorizado, sin que este tenga que intervenir cada cierto tiempo para hacer posible esta captura.

La consecución de estos objetivos asegura cubrir varios de los objetivos planteados (independientes, de plataforma y dependientes) como puede ser monitorización completa o satisfacer los requisitos del usuario (muchos de estos requisitos serán en cuanto a usabilidad), al menos parcialmente, ya que hay un importante factor tecnológico en la consecución de estos objetivos que se cubrirá con las decisiones correctas de diseño software e implementación.

Además de estos objetivos, que están relacionados directa o indirectamente con la calidad del servicio que percibe el usuario monitorizado, durante el desarrollo de la plataforma se debe hacer especial hincapié en el elemento fundamental de toda plataforma: la tarea de telemonitorización.

La tarea de telemonitorización, una vez soportada por la plataforma, es la que representa los diferentes procesos de telemonitorización especificados en el protocolo de telemonitorización. Por lo tanto, su correcta representación e implementación en la plataforma (para representar los diferentes procesos especificados en el protocolo) y la posibilidad de que dicha tarea evolucione para adaptarse a cambios en el protocolo, es un aspecto fundamental (junto con los objetivos anteriores) de cara a abordar el proceso de desarrollo de una plataforma de telemonitorización.

4.2.1.3. Propuesta: Enfoque Híbrido

Partiendo de los requisitos ya mencionados (usabilidad, ergonomía, privacidad, etc.) y revisando los enfoques presentados en el capítulo 3 (Tabla 1), el enfoque que se podría aplicar para garantizar la consecución de estos objetivos sería un enfoque centrado en el usuario, donde se considerase este como parte fundamental de la plataforma y, por lo tanto, un actor central a considerar durante el proceso de desarrollo de la misma para especificar sus necesidades y con el objetivo de garantizar la usabilidad y funcionalidad.

No obstante, además de estos objetivos, una plataforma de telemonitorización pretende ser una herramienta de soporte a unos procesos de telemonitorización, normalmente especificado en el protocolo de telemonitorización. En este protocolo de telemonitorización se suelen definir unos procesos conformados por una serie de tareas que dan lugar a la telemonitorización en sí, por ejemplo: (1) programar capturas de datos a una hora de terminada, (2) realizar dichas capturas de datos y (3) visualizar la información resultante.

Es común que dicho protocolo de telemonitorización evolucione con el objetivo de mejorar el propio proceso de telemonitorización, por lo que la plataforma de telemonitorización que soporte este protocolo debe estar preparada para adaptarse a los cambios y dar soporte a las modificaciones en las pautas del protocolo.

Esto hecho ocasiona que durante la fase de desarrollo deba prestarse una especial atención a las tareas a las que va a dar soporte la plataforma, así como garantizar mediante el diseño y la tecnología que esta plataforma puede ir evolucionando de acuerdo al protocolo.

Orientar el proceso de desarrollo de un software enfocado en un aspecto concreto, ya sea el usuario o la seguridad del sistema, normalmente no representa una desventaja que perjudique al software una vez desarrollo ni al proceso de desarrollo en sí.

No obstante, aunque la elección de un enfoque en particular ni siquiera se considere explícitamente en la mayoría de los desarrollos de software, puede que la elección de dicho enfoque no garantice la consecución de los diferentes objetivos, y sea necesario recurrir a un segundo enfoque (con el mismo o diferente nivel de prioridad).

De los enfoques ya mencionados en el capítulo 3, y centrándonos en el desarrollo de plataformas de telemonitorización y los diferentes objetivos planteados, el enfoque centrado en el usuario garantiza la consecución de parte de estos objetivos, pero por otro lado no permitiría resaltar la importancia en el proceso de desarrollo de las distintas tareas que conforman la propia telemonitorización y que son parte fundamental del objetivo de la plataforma, al representar los procesos de telemonitorización identificados en el protocolo de telemonitorización, y por lo tanto el proceso de telemonitorización en sí.

Para orientar el desarrollo en las tareas, y revisando la tabla 1 del capítulo 3, se necesitaría de un enfoque para el desarrollo orientado a las actividades del sistema. Puesto que ambos, de manera independiente, conforman una solución incompleta, la propuesta que se presenta, y que marcará el desarrollo de una plataforma de telemonitorización usando el framework presente en esta tesis es un enfoque usuario-actividad, o enfoque híbrido.

Este enfoque híbrido (usuario-actividad) hace énfasis durante el desarrollo de la plataforma en:

1. Las actividades propias que constituirán la base de la telemonitorización (tareas)
2. Las necesidades especiales del usuario en cuanto a usabilidad y utilidad de la plataforma (considerando a los usuarios parte fundamental en el propio desarrollo).

El uso de este enfoque tiene especial consideración en las primeras etapas de desarrollo, donde se definen las características, requisitos y objetivos de la plataforma de telemonitorización.

Por lo tanto, en el enfoque propuesto el usuario monitorizado es un actor central en el propio proceso desarrollo. Esto permitiría, mediante el método de validación adecuado, garantizar la usabilidad y funcionalidad de la plataforma resultante (TAM, [Davis 1989]). No obstante, debido a las limitaciones tecnológicas propias del usuario monitorizado es difícil que este tipo de usuario forme parte activa de un grupo de trabajo de expertos en distintas áreas, pasando el usuario monitorizado, dentro del desarrollo de la plataforma, a ser un actor pasivo.

Por este motivo, supervisores/expertos interactúan con el usuario monitorizado para determinar, de acuerdo a esta interacción, así como resultado de su propia experiencia y conocimiento:

1. Cómo representar las tareas (interfaz de usuario y flujo de ejecución).
2. Detectar las distintas necesidades de los usuarios monitorizados para cumplir criterios de usabilidad y efectividad del sistema.

Esta interacción entre supervisores/expertos y usuarios monitorizados como actores pasivos se realizará durante el desarrollo de la plataforma, en las etapas donde sea necesario, como por ejemplo para la especificación de las características de la plataforma o en el diseño de las interfaces de usuario.

4.2.2. Modelo de Arquitectura

4.2.2.1. Introducción

En el ámbito de las plataformas de telemonitorización, y revisando la bibliografía y soluciones existentes como algunas de las mencionadas en el capítulo 2, se puede observar que la mayoría de las soluciones contemporáneas revisadas en este ámbito comparten ciertos elementos comunes, que van desde el uso de ciertas tecnologías hasta el modelo de arquitectura (arquitectura lógica), independientemente del objetivo de la solución. Algunos ejemplos de estas arquitecturas se presentan en [Aranki 2014] [Lasierra 2010] [Piro 2014] [Alonso 2011] [Keerthika 2013], así como en el capítulo 2.

Haciendo una revisión a varios trabajos del área se detecta que ciertos elementos comunes están presentes en la mayoría de las soluciones relacionadas con la telemonitorización. Estos elementos son:

1. **Soporte externo de datos.** Tecnología que permite almacenar de manera persistente la información de un escenario de telemonitorización (datos personales de usuarios, interacciones, tareas y el resultado de su ejecución, etc.). Este soporte a la gestión de datos suele basarse en sistemas de almacenamiento externos, como servidores [Alonso 2011], o en estos últimos años, la computación en la nube [Ruiz-Zafra 2013].

2. **Software de monitorización.** Software utilizado en una plataforma de telemonitorización y que se puede clasificar en tres categorías: (1) software usado por el usuario monitorizado (en el dispositivo móvil en la mayoría de ocasiones); (2) software usado por los supervisores para visualizar toda la información (ya procesada) generada por los usuarios monitorizados y que servirá para determinar su evolución o estado; (3) software de soporte que permiten al software presentado en (1) y (2) funcionar adecuadamente (base de datos, servicios de procesamiento de datos).
3. **Dispositivos de registro de la actividad.** Dispositivos como smartphone, usados para la realización tareas relacionadas con la telemonitorización, o *wearables* (sensor de temperatura, banda de pulso cardíaco, acelerómetro) que proporcionan servicios adicionales [Mann 1998].

La figura 28 muestra el modelo de arquitectura con estos tres elementos.

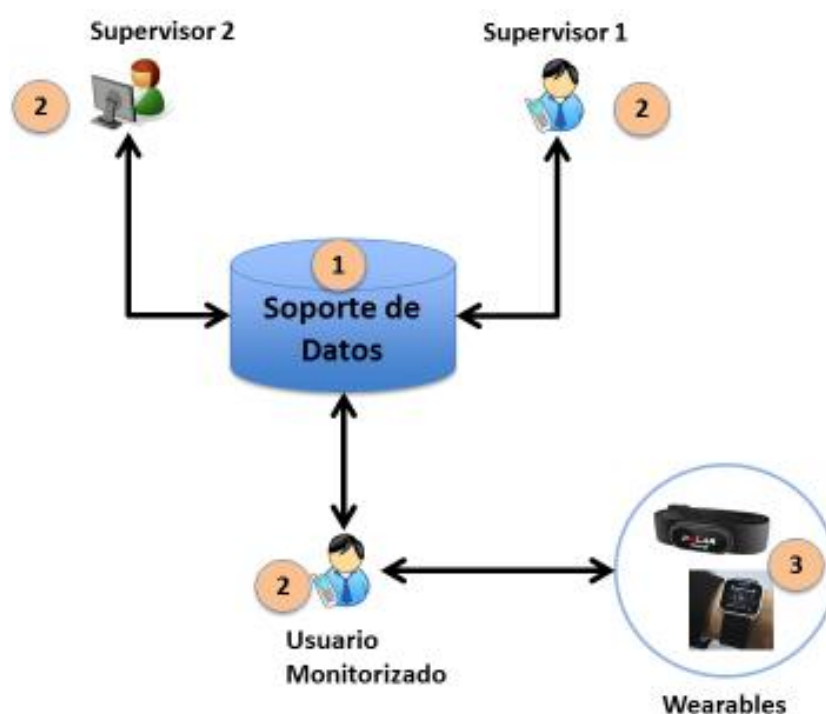


Figura 29 – Modelo de arquitectura

4.2.2.2. Características

Este modelo de arquitectura, aunque no está especificado como un estándar, a nivel práctico, constituye un modelo que muy extendido y popularizado por varias razones fundamentales:

- **Existencia de recursos disponibles.** Como pasa en cualquier otra época, las soluciones software son un reflejo del estado y evolución tecnológica, ya que nuevas tecnologías forman parte de las soluciones que se van creando. Actualmente, términos como computación en la nube o smartphone son ampliamente conocidos, y como pasa en muchas otras áreas, son elementos usados en la mayoría de las soluciones relacionadas con la telemonitorización. El hecho de que este modelo sea muy usado es consecuencia de la tecnología actualmente disponible (computación en la nube como soporte de datos,

uso de wearables por su asequible precio, smartphones por su potencia y gama de servicios).

- **Adaptabilidad tecnológica.** Gracias a las distintas tecnologías (entornos de desarrollo, plataformas PaaS, frameworks de programación, etc.), generar nuevas soluciones de telemonitorización y desplegarlas para su uso es cada vez más simple. Siguiendo este modelo de arquitectura, es fácil reemplazar la tecnología disponible en cada uno de los elementos. Por ejemplo, el soporte de datos solía consistir en un servidor clásico para almacenar información, mientras que en la actualidad son más comunes los servicios de la computación en la nube; o los dispositivos de telemonitorización antes eran PDAs (*Personal Digital Assistant*) y ahora son smartphones. La tecnología o software puede ir evolucionando, pero se respeta el mismo modelo.
- **Extensibilidad.** Una plataforma de telemonitorización con el diseño y software adecuado podría asegurar la extensibilidad de la plataforma para dar soporte a nuevas funcionalidades, esto es, cambiar o añadir nuevas funcionalidades sin que la plataforma se vea afectada en cuanto a rendimiento o costes de desarrollo. Este modelo, unido al uso de una arquitectura software que permita dicha extensibilidad y soportada por tecnología que permita la escalabilidad (como la tecnología cloud) permitiría, por ejemplo, añadir nuevas funcionalidades para dar soporte a nuevos roles o usuarios sin que el sistema se vea afectado. Esto hace que dicho modelo sea adecuado para adaptarse a cualquier cambio de un protocolo de telemonitorización.
- **Simplicidad del modelo.** Para los ingenieros resulta muy adecuado plantear una plataforma de telemonitorización siguiendo dicho modelo e identificar los distintos elementos. De la misma manera, para desarrolladores, una plataforma de telemonitorización soportada por este modelo es simple de mantener y desplegar, al existir numerosas herramientas y ayuda disponible que permiten hacerlo.

4.2.2.3. Base de la propuesta

Dicho modelo de arquitectura permite cubrir la mayoría de los escenarios de telemonitorización, ya que con la tecnología oportuna cualquier funcionalidad puede proporcionarse a través de software o hardware. Haciendo un proceso inverso, a partir de este modelo y estas características, las plataformas planteadas en este mismo capítulo como referencia o en el capítulo 2, encajan con este modelo.

La propuesta de esta tesis (Framework), y sus contribuciones, se hace partiendo de este modelo de arquitectura como soporte a las plataformas de telemonitorización, donde en los siguientes capítulos se estudiará a fondo la mejor opción en relación a los distintos aspectos (diseño y software) relacionados con los retos tecnológicos.

4.3. Elementos de e-MoDe

4.3.1. Introducción

e-MoDe es un framework que pretende simplificar el desarrollo de las plataformas de telemonitorización. Para esto, como se ha mencionado anteriormente, e-MoDe está conformado por tres elementos que pretenden ser herramientas a usar por los distintos actores involucrados en el desarrollo de la plataforma:

1. **Metodología:** conjunto de etapas para sistematizar el proceso de desarrollo (desde la primera reunión entre los actores hasta el despliegue de la plataforma) y servir de guía a los encargados del proceso de desarrollo (normalmente ingenieros de software) sobre qué aspectos deben considerar en cada una de las etapas.
2. **Conjunto de modelos:** Son dos modelos a considerar en la etapa de diseño de la metodología ya mencionada. Uno orientado a dar soporte a escenarios colaborativos y otro orientado hacia la gestión de tareas (diseño y representación)
3. **Conjunto de herramientas software:** Usadas por los desarrolladores para simplificar la implementación del sistema. Son aplicadas en la etapa 3 de la mencionada metodología (implementación) y son dos herramientas principales, Zappa (plataforma para el desarrollo de plataformas de telemonitorización) y WearIt (herramienta para garantizar la integración y uso de wearables)

La figura 30 muestra los distintos elementos del framework, que se explicarán brevemente en este capítulo y más extendidos en capítulos posteriores.

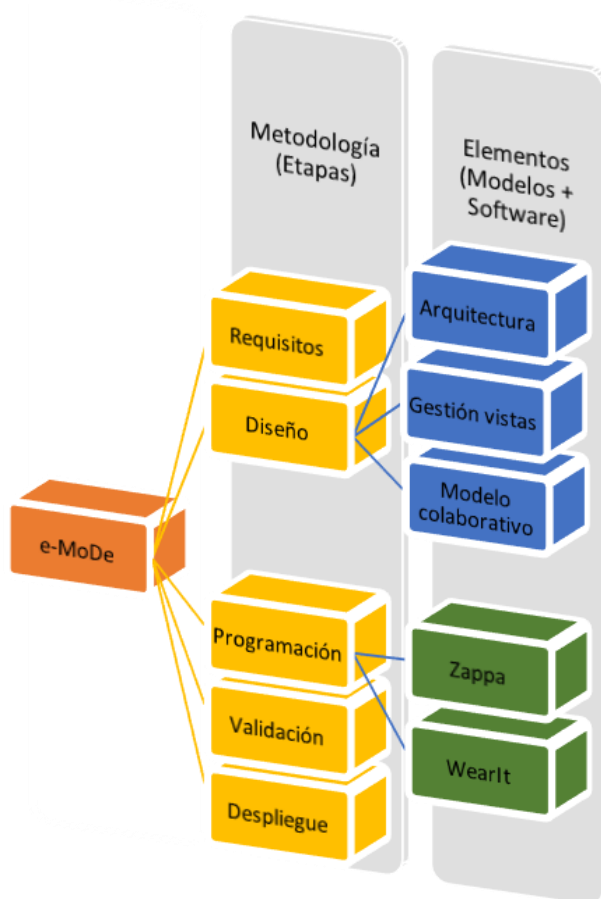


Figura 30 – Vista general de los distintos artefactos de e-MoDe

Además de simplificar el desarrollo, el objetivo es asegurar que una plataforma desarrollada usando dicho framework garantice la consecución de los distintos objetivos ya planteados (independientes del dominio, de plataforma y dependientes del dominio).

Con el compromiso de cumplir estos objetivos, a lo largo del desarrollo surgirán retos/desafíos a afrontar por los distintos actores involucrados, que van desde cómo abordar el propio proceso de desarrollo (desde la primera reunión hasta el despliegue de la plataforma) hasta solucionar problemas de ámbito tecnológico (integración de *wearables*, reusabilidad, extensibilidad, etc.).

Algunos de estos desafíos ya se han mencionado a nivel general en el capítulo 4, sección 1 (Introducción). No obstante, la Tabla 2 recoge las diferentes contribuciones que se realizan como parte del framework y qué retos u objetivos pretenden cubrir.

Reto/objetivo	Propuesta	Capítulo
Sistematización del desarrollo	Metodología	5
Extensibilidad Servicios	Arquitectura (Arquitectura software)	6, sección 1
Extensibilidad App móvil	Zappa	7, sección 1
Reusabilidad de Software	Arquitectura (Arquitectura Software), Zappa, WearIt	6, sección 1 y capítulo 7
Integración de wearables	WearIt	7, sección 2
Soporte a la colaboración	Modelo colaborativo	6, sección 2
Soporte a las tareas de telemonitorización	Modelo para la gestión de tareas	6, sección 3

Tabla 2 - Resumen de contribuciones del framework y reto/objetivo a cumplir

En este capítulo se presenta un breve resumen de las distintas aportaciones que forman parte del framework (metodología, arquitectura, gestión de tareas, modelo colaborativo, Zappa y WearIt) y en capítulos posteriores se explican con detalle.

4.3.2. Metodología

Uno de los elementos que conforman e-MoDe es una metodología para sistematizar el proceso de desarrollo de una plataforma de telemonitorización.

Dicha metodología está constituida por un conjunto de etapas que sirven de guía para los involucrados en el proceso de desarrollo (ingenieros principalmente, ya que serán por norma general los que guiarán el desarrollo) y que permiten seguir, paso a paso, todo el proceso de desarrollo, especificando qué objetivos tiene cada etapa y cuál es el resultado de cada una.

En este capítulo se hace un breve resumen de las cinco etapas (ya mencionadas en la figura 29) y que en el capítulo 5 se presentan con más detalle:

- **Especificación de requisitos:** En esta primera etapa se define una vista general de la futura plataforma, en cuanto a funcionalidades, objetivos y características, a través de dos enfoques: (1) análisis y registro de la actividad y (2) especificación de las necesidades del usuario. Como resultado se generan los requisitos funcionales y no funcionales.
- **Diseño:** Con los requisitos funcionales y no funcionales de la etapa anterior, los ingenieros de software realizan una fase de diseño software, planteando los distintos diseños del sistema: arquitectura, ingeniería de datos, usabilidad, etc. Como aportación, en esta etapa de diseño se proporcionan tres modelos: (1) un modelo de arquitectura para plataformas de telemonitorización y una recomendación para la implementación de la arquitectura software; (2) un modelo para dar soporte a una plataforma de telemonitorización colaborativa y (3) para permitir la gestión de tareas de telemonitorización. El resultado de esta etapa es un diseño software completo de la plataforma (diagramas, especificaciones, notas), a utilizar por los programadores en la etapa de implementación.
- **Implementación:** Con los diseños de la etapa anterior, los programadores desarrollan las distintas aplicaciones y funcionalidades de la plataforma. Para esta etapa se aportan dos herramientas: Zappa y WearIt. Estas herramientas permiten simplificar el desarrollo de las aplicaciones móviles y de la integración de wearables, además de garantizar otras características como extensibilidad y reusabilidad.
- **Despliegue:** Puesta en marcha de la plataforma para su uso público.
- **Validación:** Comprobar que la plataforma cumple los distintos objetivos anteriormente planteados (independientes del dominio, de plataforma y dependientes del dominio).

4.3.3. Modelos y Herramientas Software

Como se puede ver en la Figura 29, dentro de cada etapa de la citada metodología hay distintos elementos que representan las soluciones aportadas como parte de esta tesis y consideradas dentro del framework e-MoDe.

Estos elementos servirán para dar soporte a la toma de decisiones sobre aspectos de diseño y herramientas software para simplificar el proceso de desarrollo:

- **Arquitectura (Modelo).** Se propone un modelo de arquitecturas para plataformas de telemonitorización donde se identifican los elementos comúnmente presentes en cualquier plataforma de telemonitorización. De igual manera, se dan pautas o recomendaciones de cara a la implementación de la arquitectura software.
- **Gestión de vistas/tareas (Modelo).** La gestión de tareas (representación y generación) es una de los principales aspectos a abordar para permitir la extensibilidad de las funcionalidades del sistema a largo plazo. En la etapa de diseño se proporciona un modelo, así como una implementación basada en MVC para garantizar la gestión de vistas, permitiendo la modificación de vistas y obtener diferentes disseminaciones para los diferentes posibles usuarios.

- **Soporte a la colaboración (Modelo).** En las soluciones de telemonitorización hay dos roles claramente diferenciados: usuario que monitoriza y usuario monitorizado. No obstante, dentro de cada rol puede haber distintos tipos de usuarios dependiendo de su responsabilidad o funciones asignadas. En contextos de monitorización suele haber un intercambio de información entre los distintos tipos de usuarios para conseguir que el proceso de monitorización sea más productivo y eficiente. Por esto, una plataforma de telemonitorización debe dar soporte a la interacción colaborativa para permitir que los distintos tipos de usuarios puedan interactuar.
- **Zappa (Herramienta).** Además de la generación de vistas, se aportará de una herramienta software (Zappa) basada en una programación orientada a componentes como ayuda al desarrollador para el desarrollo de nuevas plataformas de telemonitorización, permitiendo añadir nuevas funcionalidades que puedan surgir en un futuro y proporcionando funcionalidades usadas en plataformas de telemonitorización para simplificar el desarrollo.
- **WearIt (Herramienta).** El uso de *wearables* está presente en casi cualquier plataforma de telemonitorización. Como parte del framework se proporciona una herramienta software para mejorar la integración de un wearable en una plataforma de telemonitorización sin necesidad de conocimiento experto por parte del programador, lo que simplifica el desarrollo.

Capítulo 5

Metodología para el desarrollo de plataformas de telemonitorización

5.1. Introducción

Uno de los principales elementos que constituyen e-MoDe es la metodología de desarrollo de plataformas de telemonitorización y que se presenta en este capítulo.

El principal objetivo de la metodología es sistematizar el proceso de desarrollo de una plataforma de telemonitorización a través de una serie de etapas. En cada etapa se especifican qué actores participan en la misma, qué aspectos deben considerarse y qué resultados se obtienen de la misma.

La metodología está formada por cinco etapas, que se explican a continuación: Especificación de requisitos, Diseño, Implementación, Despliegue y Validación.

5.2. Etapa 1: Especificación de Requisitos

Como en la mayoría de los procesos de desarrollo de un producto software, en la metodología propuesta se comienza llevando a cabo varias reuniones entre los distintos actores involucrados (*stakeholders* del sistema) para intentar especificar el sistema a desarrollar, esto es, las funcionalidades del sistema y sus características (requisitos no funcionales) [Boehm 1988]. En las plataformas de telemonitorización los requisitos no funcionales estarán orientados a dotar al sistema de propiedades como privacidad, usabilidad, portabilidad, accesibilidad, escalabilidad y mantenibilidad.

Debido a que en el proceso de desarrollo de una plataforma de telemonitorización suelen estar involucrados diferentes actores (supervisores, expertos en el área, ingenieros de software, etc.), el proceso de obtención de requisitos suele ser tedioso por la falta de conocimiento tecnológico de los actores ajenos al campo de ingeniería del software o ciencias de computación [Herzog 2002].

Esta primera etapa no difiere significativamente de cualquier otro proceso de desarrollo en cuanto a resultados (requisitos funcionales y no funcionales), pero sí en cuanto a cómo llevarla a cabo para la obtención de dichos requisitos.

Para entender este punto, debemos analizar el propósito de una plataforma de telemonitorización. Una plataforma de telemonitorización se concibe para permitir la supervisión remota de usuarios, y en determinadas situaciones, para dar soporte a un protocolo de telemonitorización (también llamado metodología de telemonitorización), esto es, un conjunto de procesos diseñados para alcanzar determinados objetivos basándose en la telemonitorización.

Por ejemplo, una funcionalidad recurrente en los protocolos de telemonitorización de personas mayores consiste en notificar a los supervisores cuando el valor de un dato recibido de un sensor está fuera de un determinado umbral (temperatura, presión arterial, etc.). Los supervisores establecen aquellos hechos o situaciones que quieren detectar para los usuarios monitorizados. Cuando se produce dicho hecho, la plataforma proporciona una respuesta automática, o bien notifica a los supervisores para que estos actúen de acuerdo al protocolo de telemonitorización (llamar al usuario, enviar un mensaje, avisar a algún familiar o a su supervisor, etc.) [Scanail 2006] [Kiss 2011].

El protocolo de telemonitorización es definido y diseñado principalmente por supervisores y expertos. No obstante, en determinadas situaciones también se necesita la colaboración de

los usuarios monitorizados, ya que se requiere la ayuda o retroalimentación de éstos últimos para determinar qué tareas se muestran más adecuadas, identificar necesidades especiales, deducir características de la plataforma que mejoren el proceso de telemonitorización, etc. [Koopman 2014].

Por tanto, esta colaboración entre supervisores y usuarios monitorizados para el diseño del protocolo se suele realizar mediante una participación indirecta por parte del usuario monitorizado, tomando este un rol pasivo.

Por otro lado, el usuario monitorizado suele tener un papel más secundario en el proceso de desarrollo de la plataforma de telemonitorización, excepto en la etapa de despliegue donde su *feedback* e impresiones directas son consideradas de cara a cambios o mejoras en la plataforma, de cara a mejorar cuestiones como la usabilidad del sistema. En el resto de etapas su participación se canaliza a través del supervisor, que es quien interactúa con el usuario monitorizado para intentar, mediante preguntas o cuestionarios, ver su reacción en determinadas situaciones y así determinar las necesidades del usuario.

Por su parte, el supervisor hace de nexo entre usuarios monitorizados y responsables de desarrollo (ingenieros de software, analistas, etc.) para transmitir las distintas necesidades del usuario monitorizado (problemas de visión, falta de conocimiento del uso de algún tipo de dispositivo), así como posibles características de la plataforma que permitan mejorar los procesos de telemonitorización (como notificar cambios en la monitorización, mostrar adecuadamente que se está realizando correctamente la tarea de telemonitorización).

Toda esta interacción entre supervisor y usuario monitorizados no siempre se refleja en el protocolo de telemonitorización, ya que características como la usabilidad de la plataforma se abordan en esta etapa de especificación de requisitos.

El protocolo de telemonitorización define el marco para la obtención de requisitos, tanto funcionales, como no funcionales. Esta obtención de requisitos es especialmente crítica en soluciones de este tipo, donde puede haber actores con necesidades especiales, cuya aceptación de la tecnología (usabilidad y funcionalidad) determina en buena medida el éxito de la plataforma [Broens 2007].

El cumplimiento de estos requisitos mediante un método de validación concreto (encuesta, heurísticas, nivel de puntuación, etc.) permitiría garantizar que la plataforma satisface el modelo *TAM* (*Technology Acceptance Model* [Venkatesh 2000] [Davis 1989]), es decir, que el usuario perciba la plataforma como una herramienta útil y fácil de usar.

Además, al ser la plataforma de telemonitorización una solución de soporte al protocolo de telemonitorización, la definición de las distintas funcionalidades o tareas de telemonitorización en la plataforma resulta crucial para garantizar la correcta representación del protocolo de telemonitorización a través de los distintos servicios (funcionalidades) y elementos (hardware y software) de la plataforma.

Con la premisa de garantizar que la plataforma satisfaga todas las necesidades del usuario y las diferentes funcionalidades definidas en el protocolo que dan soporte a los procesos de telemonitorización, se propone abordar esta etapa de especificación de requisitos siguiendo el enfoque híbrido (usuario-actividad) presentado en el capítulo 4. Esto implica que durante

la fase de obtención de requisitos de esta etapa se haga especial hincapié en las necesidades del usuario y en las diferentes tareas de telemonitorización identificadas en el protocolo.

La adopción del enfoque híbrido (usuario-actividad) posibilita que los requisitos funcionales y no funcionales resultantes de esta etapa cubran, tanto las necesidades de los usuarios monitorizados, como las diferentes funcionalidades o tareas que dan soporte a los procesos de telemonitorización.

Los responsables del desarrollo (ingenieros de software principalmente) toman el protocolo de telemonitorización ya definido, y con la colaboración de los supervisores, llevan a cabo la especificación de requisitos a través de dos fases de análisis (ver Figura 31):

1. **Registro y análisis de la actividad:** Se determinan tareas de telemonitorización, datos a registrar, etc.
2. **Especificación de interacciones y necesidades especiales de usuarios:** Se recogen necesidades especiales del usuario monitorizado y las diferentes interacciones entre usuarios.

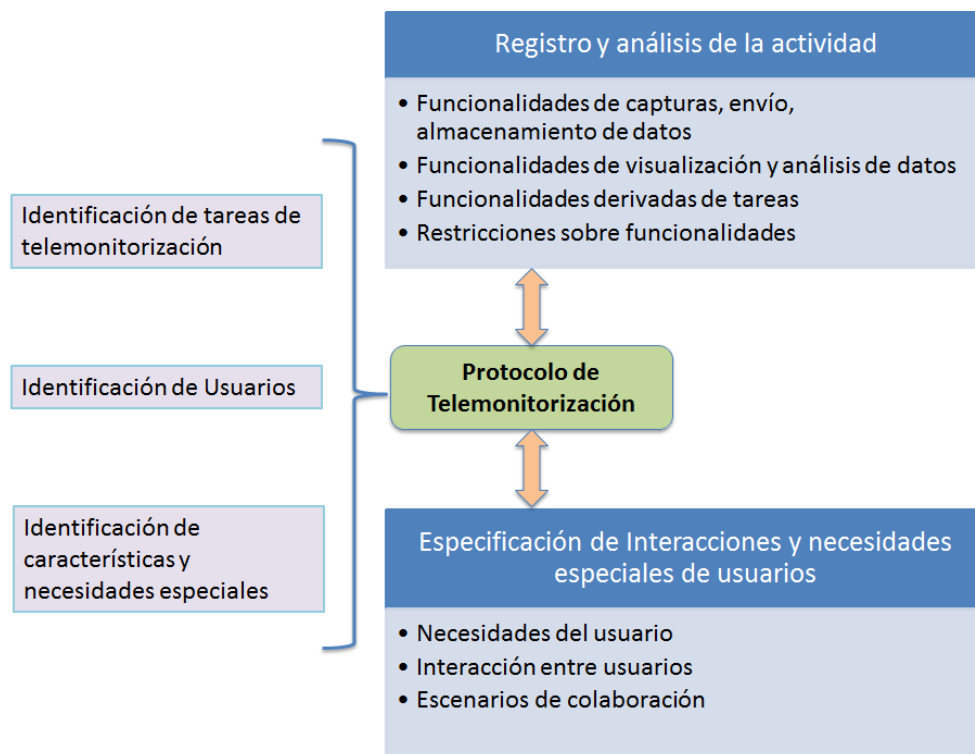


Figura 31 - Análisis del protocolo de telemonitorización para la obtención de requisitos a través de las dos fases de análisis

Mediante estas dos fases de análisis, (1) registro y análisis de la actividad y (2) especificación de interacciones y necesidades especiales de usuario, se obtiene una especificación de funcionalidades y características de la plataforma, así como necesidades del usuario que dan lugar a los requisitos funcionales y no funcionales de la plataforma de telemonitorización. Estos requisitos serán los parámetros de entrada para la siguiente etapa de la metodología: Diseño.

Por otro lado, es probable que durante la obtención de requisito se detecten problemas y posibles mejoras del propio protocolo. Estos problemas pueden deberse a la funcionalidad para mejorar procesos de telemonitorización, nuevos procesos que pueden surgir gracias al uso de tecnologías que podían ser desconocidas por los expertos en el área del protocolo o nuevos criterios de usabilidad recomendables en el desarrollo de aplicaciones y que no se adaptan a lo especificado en el protocolo.

Por este motivo, la metodología contempla la posibilidad de modificar o añadir ajustes al protocolo durante esta etapa de especificación de requisitos, es decir, volver a realizar la etapa de especificación de requisitos las veces que se considere, por si es necesario modificar algún aspecto o contemplar nuevas posibilidades de telemonitorización, como ajustar algún proceso de acuerdo a la tecnología o posibles interacciones.

La figura 32 muestra este mismo proceso simplificado, destacando la participación de los distintos actores en las distintas fases de esta primera etapa.

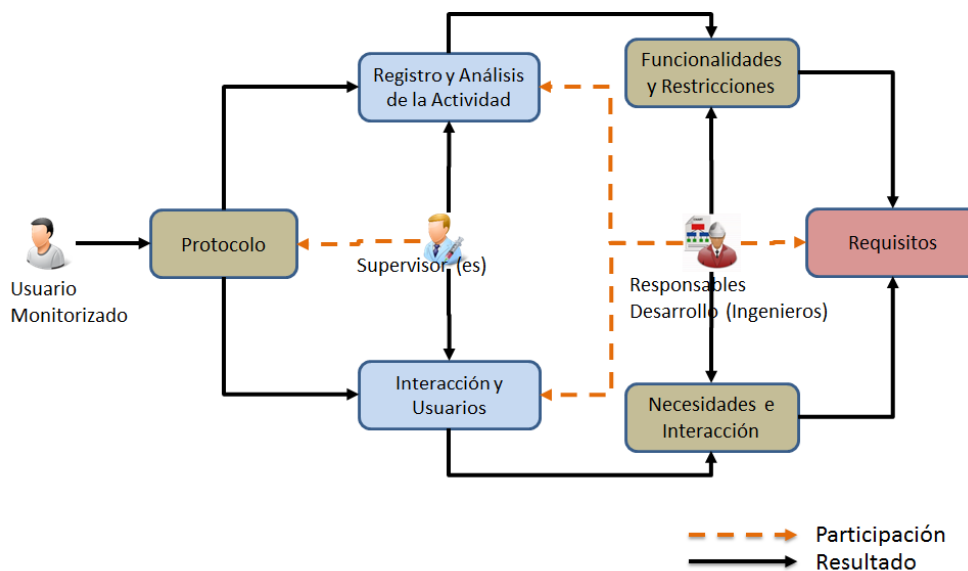


Figura 32 - Proceso de participación y resultado de la etapa 1: Especificación de Requisitos

A continuación, se explican en detalle ambas fases de análisis.

5.2.1. Registro y análisis de la actividad

En esta fase se deben especificar, a partir de las tareas de telemonitorización identificadas en el protocolo de telemonitorización, las distintas funcionalidades que tendrá el sistema.

Se deben definir las funcionalidades relacionadas con el tratamiento de datos (qué datos se obtendrán y su origen -wearables, por ejemplo-, qué procesamiento se llevará a cabo con los mismos, dónde se almacenará, cómo se visualizarán y analizarán, etc.) y especificar funcionalidades propias de las aplicaciones que dan soporte a la ejecución de tareas (por ejemplo, responder a ciertos eventos derivados de una interacción del usuario con el dispositivo). Asimismo, se deben identificar restricciones sobre estas funcionalidades, esto es, requisitos no funcionales, como, por ejemplo, la robustez en la ejecución de las tareas que representen esas funcionalidades o la necesidad de registrar datos asociados a una tarea en tiempo real.

De esta forma, en una plataforma de telemonitorización para usuarios con problemas cognitivos (memoria, aprendizaje, percepción), una tarea podría ser realizar una actividad cotidiana como comer. La peculiaridad de esta tarea dentro de la plataforma de telemonitorización es que debe realizarse a unas horas determinadas (3 veces al día), se debe registrar su ejecución con la cámara del dispositivo móvil y el pulso cardíaco mientras se realiza dicha tarea.

Esta tarea plantea dos requisitos funcionales: registrar el pulso cardíaco y grabar el vídeo de la ejecución; y dos requisitos no funcionales: robustez en la captura de datos del pulso cardíaco y calidad del vídeo de la ejecución de la actividad (con el objetivo de que el supervisor pueda evaluar correctamente la ejecución).

La figura 33 muestra este ejemplo en términos de obtención de requisitos funcionales y no funcionales. Siguiendo el mismo proceso en todas las tareas de la plataforma se obtienen un conjunto de requisitos funcionales y no funcionales como resultado de esta fase de análisis.

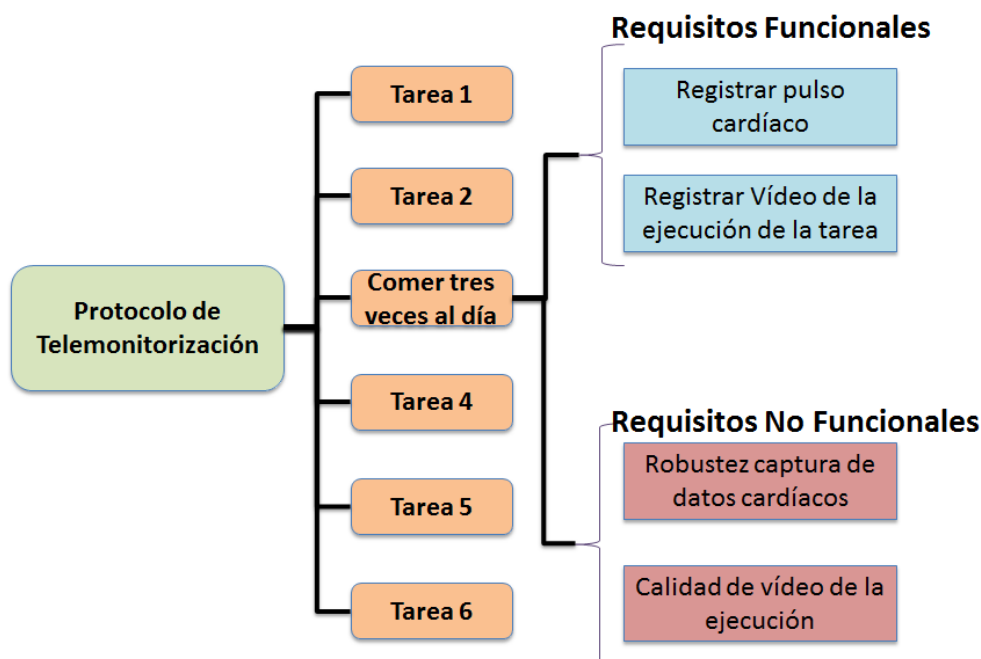


Figura 33 - Obtención de requisitos funcionales y no funcionales de una tarea de ejemplo (centrado en la actividad)

5.2.2. Especificación de interacciones y necesidades especiales del usuario

En esta fase de análisis se deben especificar las necesidades especiales del usuario monitorizado e identificar las distintas interacciones entre los actores de la plataforma.

Se parte de los diferentes perfiles de usuario (características y necesidades especiales) identificados en el protocolo de telemonitorización. Con estos datos, se suelen describir los diferentes requisitos que necesita satisfacer la plataforma para permitir su uso por parte de los usuarios monitorizados, esto es, usualmente, aspectos relacionados con la usabilidad.

Por ejemplo, en una plataforma de telemonitorización orientada a personas mayores puede surgir una necesidad especial relativa al tamaño del dispositivo donde se realizarán las tareas de telemonitorización (donde por problemas de visión se requiera un tamaño mínimo de pantalla), y además una necesidad especial relativa a la visualización, donde los iconos de las diferentes interfaces en la aplicación tengan un tamaño mínimo para asegurar que son accesibles.

Además de especificar las necesidades especiales y los requisitos derivados de ellas, en esta fase se deben identificar qué usuarios interactúan entre sí y con qué objetivo, ya que la posibilidad de interacción en una plataforma de telemonitorización es crucial [McAlister 2004] [Inglis 2010].

El soporte a la interacción es especialmente importante en este ámbito, ya que es común que protocolos y plataformas de telemonitorización se sitúen en escenarios multidisciplinares, donde expertos o supervisores de diferentes ámbitos (medicina, deporte, nutrición, psicología) supervisan usuarios cada uno con unos objetivos dependientes de su ámbito.

Por ejemplo, en una plataforma de telemonitorización para atletas (usuarios monitorizados) supervisados por entrenadores deportivos y nutricionistas (supervisores), los cambios por parte de un supervisor nutricionista sobre la dieta del atleta deben ser notificados al entrenador para que este adapte el entrenamiento de acuerdo a la nueva dieta con el objetivo de mantener el rendimiento deportivo. De igual manera, cambios en el entrenamiento deberían notificarse al nutricionista por si considera oportuno modificar la dieta.

De cara a ilustrar cómo sería la puesta en práctica de esta fase, se toma el ejemplo anteriormente mencionado en esta sección: una plataforma de telemonitorización para personas mayores. En particular vamos a considerar que tenemos tres tipos de usuarios distintos: médicos (que desempeñarán el rol de supervisores), familiares (que desempeñarán el rol de supervisores) y personas mayores (que desempeñarán el rol de usuarios monitorizados). A partir de estos tres tipos podríamos identificar tres situaciones o interacciones:

- Tipo de interacción 1: El supervisor notifica al usuario monitorizado cuándo debe realizar una tarea.
- Tipo de interacción 2: El usuario monitorizado notifica al supervisor cuándo ha completado una tarea, así como el resultado o resumen de la misma.
- Tipo de interacción 3: El usuario monitorizado notifica a sus familiares cuándo ha completado una tarea.

En este ejemplo se vuelve a obtener una lista de requisitos funcionales y no funcionales. La figura 34 muestra la lista de requisitos resultantes.

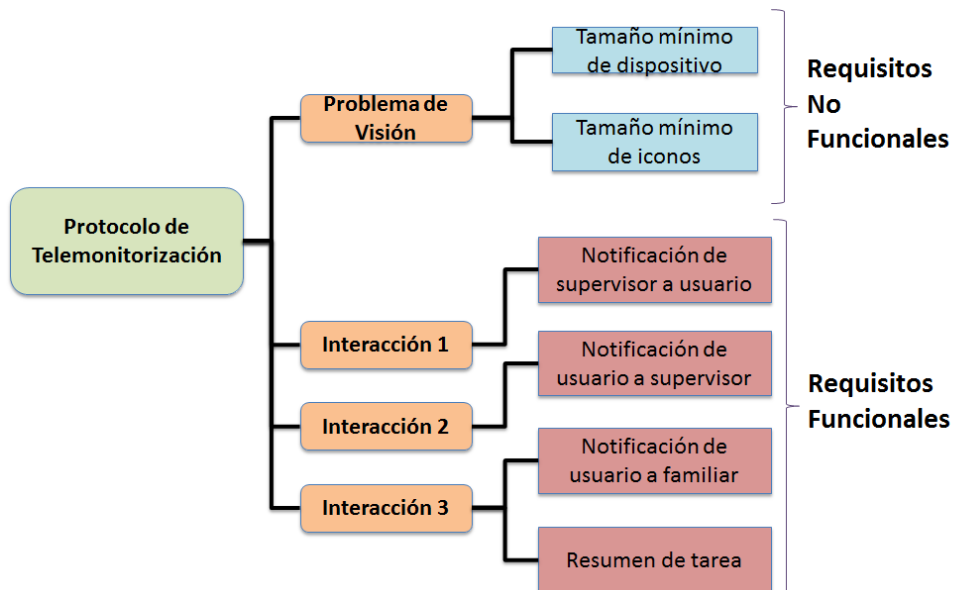


Figura 34 - Obtención de requisitos funcionales y no funcionales centrado en las necesidades del usuario y distintos tipos de interacciones

Debido a que expertos/supervisores de diferentes áreas pueden estar involucrados en un proceso de telemonitorización, es necesario que estos puedan interactuar con el objetivo de notificar cambios en sus tareas de telemonitorización, ya que estas pueden repercutir en otras áreas.

Al igual que la fase de *registro y análisis de la actividad*, como resultado de esta fase se obtiene una lista de requisitos funcionales y no funcionales.

5.3. Etapa 2: Diseño

Esta segunda etapa de la metodología parte de los requisitos (funcionales y no funcionales) de la etapa anterior como parámetros de entrada y tiene como objetivo abordar el diseño software de la plataforma de telemonitorización, generando como resultado el conjunto de modelos y especificaciones que sirvan a los programadores para abordar la etapa de implementación, esto es, plasmar la estructura o componentes del sistema, definir la arquitectura, la arquitectura de datos (modelo y gestión de datos), etc.

En esta etapa de diseño, como aplicación del enfoque híbrido usuario-actividad considerado, se suele involucrar también a expertos en el área o supervisores para el diseño de interfaces de usuario, ya que estos diseñan o validan dichas interfaces para asegurar que estas resultan usables para el usuario monitorizado, ya sea realizando pruebas de aceptación informales (*feedback* del usuario monitorizado) o llevando a cabo una validación más formal basada en algún método particular [Lettner 2012].

Para facilitar el diseño de la plataforma y la posterior implementación, se propone organizar el diseño por áreas de interés. De acuerdo a los requisitos funcionales y no funcionales derivados de las dos fases de análisis abordadas ((1) registro y análisis de la actividad; (2) especificación de interacciones y necesidades especiales de los usuarios) se definen cuatro áreas de interés relevantes sobre los que se centrará la etapa de diseño:

- Arquitectura
- Gestión de tareas

- Colaboración
- Necesidades del usuario

El análisis de cada una de estas áreas de interés generará unos resultados (ver Figura 35), que abarcan diagramas de diseño, interfaces o cualquier otra especificación, y que se utilizarán en la siguiente etapa (implementación).

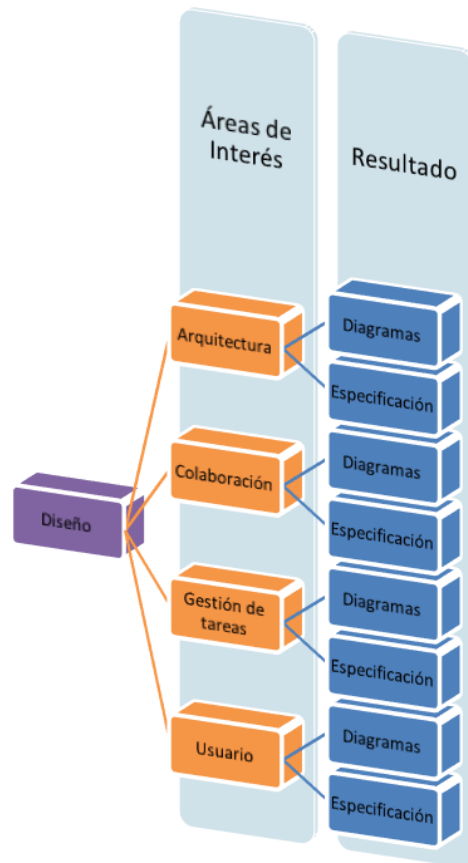


Figura 35 - División de la etapa de diseño en áreas de interés

Para alguna de estas áreas de interés, como gestión de tareas o colaboración y como aportación de la tesis y elementos del framework, se facilitan diagramas y especificaciones concretas que pueden utilizarse para dar soporte a algunas funcionalidades que probablemente estén presentes en cualquier plataforma de telemonitorización, sin necesidad de plantear un diseño concreto (capítulo 6). El uso de las soluciones propuestas permite la consecución de varios de los objetivos ya mencionados (independientes del dominio, de plataforma), como extensibilidad o interacción entre los distintos usuarios.

A continuación, se detallan las diferentes áreas de interés.

5.3.1. Arquitectura

Una de las principales tareas a abordar en la etapa de diseño es definir la arquitectura del sistema, esto es, descomponer todo el sistema en subsistemas, especificando las características de estos y cómo van a interactuar entre sí [Clement 2002]. Además, también se pretende especificar qué funcionalidades ofrece cada subsistema de las especificadas en la etapa anterior (requisitos funcionales).

En esta primera área de interés de la etapa de diseño se intenta abordar esta tarea a través de dos objetivos:

1. **Arquitectura de la plataforma:** Definir el tipo de arquitectura, aplicaciones y tecnología (dispositivos, soporte a la arquitectura, *wearables*). Más concretamente, se persigue identificar los distintos elementos hardware y software involucrados en la plataforma (aplicaciones, dispositivos, tecnología específica, etc.).
2. **Distribución de funcionalidades:** Definidos los distintos elementos (subsistemas) de la arquitectura (aplicaciones, tecnología, dispositivos) es necesario especificar las distintas funcionalidades de las que se encargará cada elemento, y que los programadores implementarán posteriormente con el objetivo de dar soporte a las funcionalidades de la plataforma.

Para alcanzar estos objetivos, en el capítulo 6, sección 1, se presenta la propuesta completa de arquitectura para plataformas de telemonitorización, así como una distribución de funcionalidades entre los distintos dispositivos y subsistemas. Esta propuesta, basada en el modelo de arquitectura propuesto en el capítulo 4 y en la tecnología presentada en el capítulo 3 (computación en la nube, arquitectura software híbrida, dispositivos móviles, *wearables*), define mediante diferentes vistas una arquitectura para plataformas de telemonitorización.

Estas vistas de la arquitectura identifican los diferentes subsistemas de una plataforma de telemonitorización y la interacción entre ellos. Dentro de la arquitectura se identifican varios subsistemas presentes en cualquier plataforma de telemonitorización (gestión de tareas, gestión de notificaciones) y se distribuyen las diferentes funcionalidades de la plataforma en los diferentes subsistemas.

Estas funcionalidades se distribuyen entre:

- **Back-end:** Funcionalidades que se implementarán en servicios web con soporte *cloud*, accesibles mediante interfaces como las especificadas en la propuesta de arquitectura híbrida (capítulo 3). El diseño de estas interfaces es responsabilidad de los ingenieros de software, no obstante, en el capítulo 6.1 se extiende este aspecto de diseño y se muestran varios ejemplos como soporte o ayuda adicional a ingenieros software.
- **Aplicaciones:** El resto de funcionalidades software se implementarán en las aplicaciones móviles: captura de datos, registro de la interacción del usuario monitorizado con el dispositivo móvil, etc.

Con la satisfacción del objetivo de arquitectura se pretende simplificar el diseño de la arquitectura de la plataforma, además de abordar la arquitectura software para identificar así la distribución de las funcionalidades de la plataforma.

Además de estas funcionalidades, los responsables del proyecto (ingenieros) deben abordar, en función de los requisitos y disponibilidad de recursos, qué software y tecnología se van a utilizar: plataformas en la nube para el despliegue y lenguaje de programación para el *back-end*, qué *wearables* por calidad-precio son los adecuados, en qué plataformas se van a desarrollar las aplicaciones móviles, etc.

Por lo tanto, del diseño arquitectónico, que tiene como objetivo organizar los distintos componentes de la plataforma, distribuir las funcionalidades y determinar la tecnología que se va a usar, obtendremos:

- Identificación de los diferentes componentes o módulos del sistema y la relación entre ellos.
- *Back-End*: Interfaz de los distintos servicios web para las funcionalidades del back-end (API)
- Decisiones a nivel técnico y de implementación: qué lenguajes o herramientas se van a usar para el desarrollo, o qué dispositivos de captura de datos (*wearables*) serán usados por los programadores.
- Especificación de las funcionalidades que tendrán que implementarse en las aplicaciones móviles.

La figura 36 muestra un resumen del aspecto *arquitectura*, con los objetivos y los resultados.

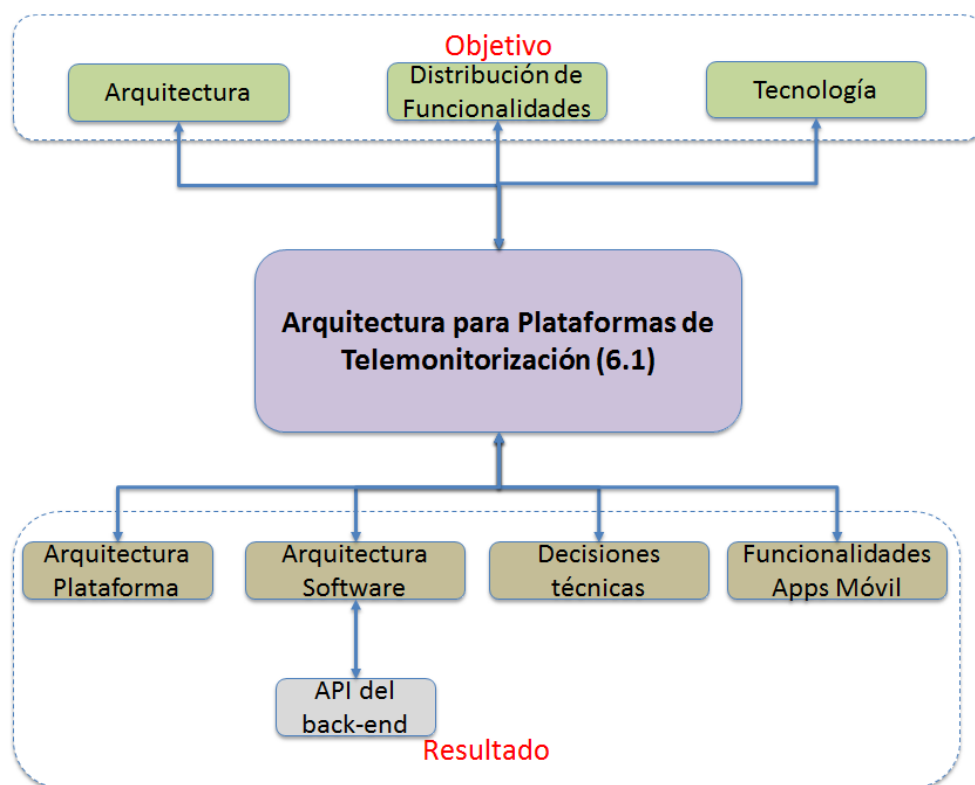


Figura 36 Objetivos y resultados del bloque de *arquitectura*

5.3.2. Gestión de tareas

En un protocolo de telemonitorización, uno de los elementos clave son las tareas de telemonitorización, es decir, las distintas actividades o acciones que tienen que realizar los usuarios monitorizados como parte del proceso de telemonitorización: controlar el pulso cardíaco a determinadas horas, notificar a su supervisor algún tipo de evento, acometer una acción y registrarla, grabarse en vídeo realizando alguna actividad cotidiana, realizar algún ejercicio físico, etc.

Estos protocolos suelen ir variando conforme aparecen nuevos avances tecnológicos en el dominio de aplicación de la plataforma o bien por intentar mejorar los procesos de telemonitorización. Este hecho ocasiona que sea crucial que la plataforma de telemonitorización que representa y da soporte al protocolo de telemonitorización permita añadir o modificar nuevas tareas para, de esa forma, dar soporte a nuevas funcionalidades. La plataforma debe permitir incorporar nuevas tecnologías (software o hardware) con el objetivo de adaptarse a los cambios en el protocolo de telemonitorización e ir evolucionando paralelamente.

En este aspecto de la etapa de diseño se propone un modelo para la gestión de tareas y que se explica con más detalle en el capítulo 6, sección 2. Este modelo forma parte de una propuesta basada en el uso del patrón de arquitectura software MVC (Modelo-Vista-Controlador), con el objetivo de representar tareas independientes de la plataforma, permitiendo representar y gestionar la tarea de acuerdo a las necesidades del usuario (ver Figura 37).

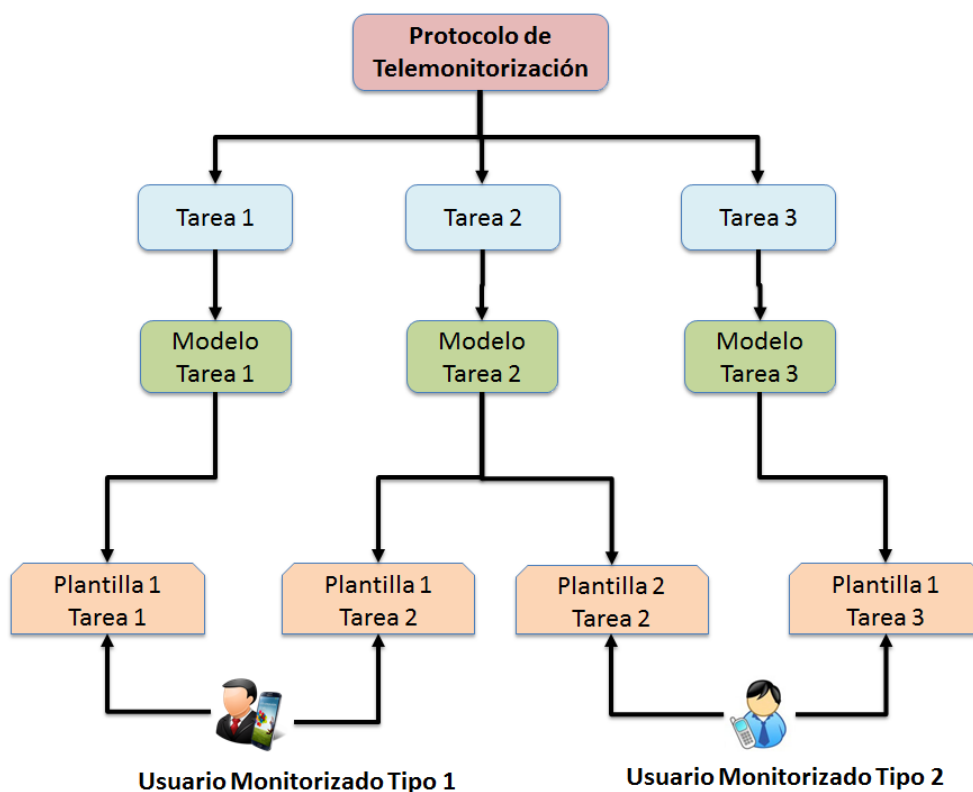


Figura 37 - Visión de conjunto del modelo propuesto para la gestión de tareas

El resultado de este aspecto en la etapa de diseño es el modelo para la gestión de tareas que, como ya se ha mencionado, se extiende en el capítulo 6 sección 2.

5.3.3. Colaboración

Uno de los objetivos planteados es la interacción entre los distintos tipos de usuarios, principalmente supervisor y usuario monitorizado. No obstante, en numerosas ocasiones, y por demandas del proceso de telemonitorización, esta interacción no suele ser única, en el sentido de que puede haber distintos tipos de interacciones para representar distintos eventos o acciones.

Además, cada vez es más frecuente que una plataforma de telemonitorización sea multidisciplinar, esto es, que haya distintos tipos de supervisores y/o usuarios monitorizados con diferentes objetivos o responsabilidades, ya que el proceso de telemonitorización de un usuario se lleva a cabo por varios supervisores de distintas disciplinas.

La metodología propuesta proporciona un modelo para entornos colaborativos que facilita a ingenieros el diseño de estas funcionalidades, simplificando cómo abordar el soporte a la colaboración. El resultado de este aspecto es dicho modelo colaborativo, que se explica detalladamente en el capítulo 6 sección 3.

5.3.4. Usuario

La adopción del enfoque híbrido usuario-actividad en el proceso de desarrollo ocasiona que, tanto el usuario, como las tareas de telemonitorización tengan especial relevancia y consideración en el desarrollo de la plataforma.

El usuario es quien determina mediante su uso continuado y *feedback* si la plataforma es útil y usable, por lo que adecuar la plataforma a sus necesidades es obligatorio para alcanzar este objetivo.

Por esta razón, en la fase de diseño es importante abordar el diseño de interfaces de usuario de forma que garanticen la correcta usabilidad por parte del usuario monitorizado y cualquier otra característica que repercuta en su correcta interacción con la plataforma, como el tamaño del dispositivo donde se realizarán las tareas (smartphone/Tablet) o tamaño y localización de los wearables.

Debido a que esta área de interés está totalmente orientada al diseño de las interfaces de usuario y, en definitiva, requisitos no funcionales, se hace especial hincapié para que los encargados del desarrollo tengan especial consideración y se aborde con el mismo nivel de importancia que cualquiera de los otros aspectos.

Fruto del análisis de esta área relacionada con los requisitos no funcionales orientado al usuario (usabilidad principalmente) se obtienen interfaces para las distintas aplicaciones, así como una serie de restricciones que usarán los desarrolladores en la etapa de implementación, para garantizar la aceptación de la tecnología por parte del usuario monitorizado.

5.3.5. Consideraciones finales sobre diseño

Como resultado de esta etapa de diseño, se debe obtener un conjunto de diagramas o especificaciones de los diferentes aspectos: arquitectura, distribución de funcionalidades, soporte a la colaboración, gestión de tareas y las diferentes interfaces de usuario y restricciones.

Como soporte a esta etapa de diseño, y contribución de la tesis, en el capítulo 6 se facilitan modelos para la arquitectura, el soporte a la colaboración y la gestión de tareas; con el objetivo de simplificar dicha etapa de diseño. Los desarrolladores toman dichos modelos y los adaptan (o instancian) para, a través de las pautas que se proponen, dotar a la plataforma de dichas características (soporte a la colaboración, gestión de tareas) o identificar más fácilmente los diferentes artefactos de la plataforma (arquitectura).

La figura 38 muestra un resumen de esta etapa de diseño, donde se muestran como entrada los requisitos de la etapa anterior (especificación de requisitos) y el resultado de dicha etapa de diseño. Además de los elementos o propuestas de la imagen, existen otros elementos a abordar dentro de la etapa de diseño, como pueden ser la modelo y gestión de los datos, seguridad, etc., que están fuera del ámbito de esta tesis, pero que están presentes y deberán abordarse por los ingenieros correspondientes.

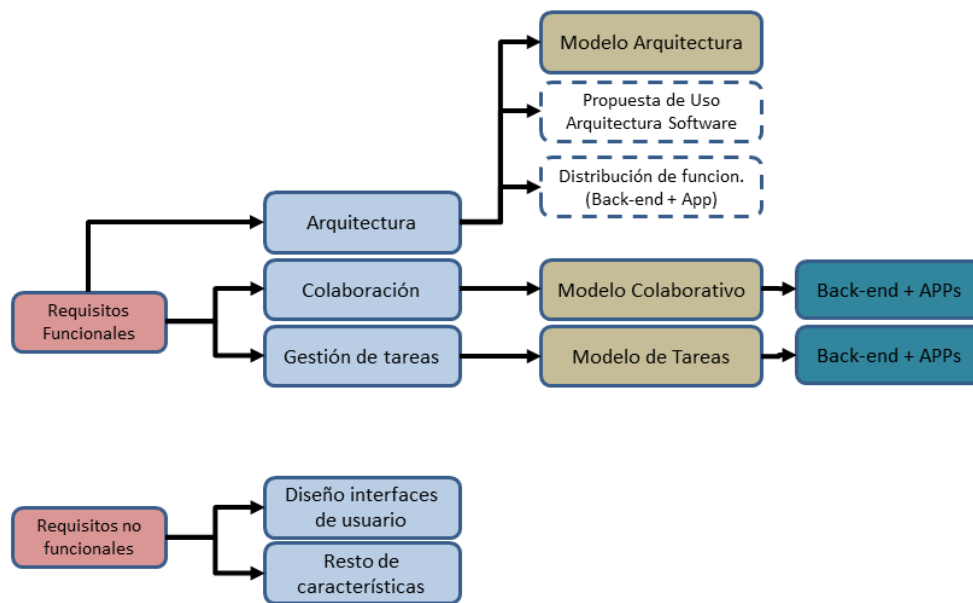


Figura 38 - Vista general de la etapa de diseño

5.4. Etapa 3: Implementación

Esta tercera etapa consiste en la implementación por parte de los programadores de las diferentes funcionalidades que conforman la plataforma.

Según la descripción de la etapa anterior, en esta etapa los programadores deben recibir documentación completa de la especificación de la plataforma de telemonitorización y varios diseños de la misma. Para abordar esta etapa de implementación de una manera eficiente se proponen dos grupos de programadores distintos, siguiendo la distribución de funcionalidades propuesta en el área de interés de *Arquitectura* de la etapa de diseño:

- **Programadores de *Back-end*:** Implementación del software presente en la parte del servidor/cloud. Esto incluye: servicios web, la ingeniería de datos, seguridad, protocolos de seguridad, procesamiento complejo de datos, etc.
- **Programadores de aplicaciones móviles:** Implementación de las distintas aplicaciones móviles para las diferentes plataformas. Dentro de estas aparecen, por ejemplo: interfaces de usuario, funcionalidades derivadas al dispositivo móvil, integración de dispositivos para la captura de datos, etc.

Los programadores del *back-end* se encargarán de implementar:

- Los servicios web que darán soporte a la telemonitorización, siguiendo la API definida por los ingenieros software y la tecnología o plataforma elegida.
- La instanciación del modelo colaborativo propuesto aplicado al escenario en concreto, es decir, los diferentes servicios que permitan implementar el modelo colaborativo: envío y recepción de mensajes, gestión de eventos, etc.
- La parte correspondiente al modelo para la gestión de vistas, ya que es necesario implementar dichos modelos y definir los servicios para gestionarlo, pudiendo así crear, modificar, eliminar o recuperar modelos. De la misma manera, esto generará un conjunto de servicios para dar soporte al acceso a tareas por parte del resto de usuarios.

Los programadores de aplicaciones móviles usarán estos servicios para obtener las distintas funcionalidades. Estos desarrolladores de aplicaciones móviles se encargarán de implementar:

- Las funcionalidades correspondientes para dar soporte al modelo colaborativo, esto es, usar los servicios web ya creados e implementar las funcionalidades derivadas en la aplicación móvil para dar soporte a la colaboración.
- El soporte al modelo de gestión de vistas, esto es, acceder a los servicios web para obtener las tareas correspondientes, procesarlas y mostrarlas de acuerdo a la diseminación correspondiente (interfaz de usuario).
- Las distintas interfaces de usuario propuestas en la etapa de diseño.
- Las aplicaciones móviles, esto es, implementar las funcionalidades que forman parte de las diferentes aplicaciones: gestión de información interna, interacción con el *back-end*, interacción con el usuario, captura y registro de datos procedentes de dispositivos, etc.

En esta etapa de implementación no se presentan premisas sobre cómo un programador debe abordar dicha etapa, planteando la misma como cualquier etapa de implementación de un proyecto software y dejándole que actúe bajo su criterio, implementando de la mejor manera posible aquellas funcionalidades especificadas en la etapa de diseño.

Recordando los objetivos mencionados a lo largo de la tesis (dependientes del dominio, de plataforma e independientes del dominio), encontramos dos objetivos concretos estrechamente ligados a la etapa de implementación: reusabilidad y extensibilidad. Es decir, garantizar que el software que se implementa es reusable y extensible para adaptarse a nuevas necesidades.

Por las características de la arquitectura software propuesta (arquitectura híbrida), esta posibilita que la parte del *back-end* de la plataforma sea extensible y reusable, ya que el usar servicios web nos permite explotar las posibilidades de encapsulación y modularidad. Esto, a su vez, permite añadir nuevos servicios sin afectar al rendimiento de la plataforma (extensibilidad) y además, al entender un servicio como una funcionalidad encapsulada sin estado que recibe unos parámetros de entrada y produce unos resultados de salida, se puede sustituir dicho servicio (si la API no cambia), o bien usarlo en otros proyectos fácilmente (reusabilidad).

Aunque esta extensibilidad y reusabilidad esté garantizada en el *back-end*, por el tipo de tecnología y propuesta que se usa, en el desarrollo de aplicaciones móviles hay que optar por

otra tecnología o diseño de soluciones que permita a las aplicaciones, de igual manera, ser extensibles y reusables para ser fácilmente adaptables..

La mayoría de las aplicaciones móviles de una plataforma de telemonitorización comparten funcionalidades en común: almacenar información, interactuar con la parte del *back-end*, procesar diferente información procedente de los servicios, usar los elementos del smartphone como cámara o micrófono, etc.

Esto hace que la buena praxis en la programación de estas funcionalidades pueda permitir reutilizar este código fácilmente en futuras aplicaciones, ahorrando tiempo de desarrollo. Asimismo, al igual que en la parte del *back-end*, estas aplicaciones deben estar preparadas para añadir a nuevos usuarios/roles, tareas de telemonitorización o funcionalidades concretas para adaptarse a cambios en el protocolo de telemonitorización.

Para abordar esta reusabilidad y extensibilidad en aplicaciones móviles se propone una plataforma para el desarrollo de plataformas de telemonitorización llamada **Zappa**, explicada en el capítulo 7, sección 1.

Otro problema con el que suelen encontrarse los desarrolladores de aplicaciones móviles para plataformas de telemonitorización es la interacción con wearables. A través de estos dispositivos es posible conseguir determinadas funcionalidades que no sería posible alcanzar mediante software. El principal inconveniente estriba en que para trabajar con estos dispositivos se requiere conocimiento previo acerca de cómo funciona el propio *wearable* y, además, programar la interacción con el mismo y su posterior procesamiento de datos, tarea que en muchas ocasiones resulta tediosa, ocasionando que la integración de *wearables* para su uso en plataformas de telemonitorización puede complicar el desarrollo, ya que implica tratar con detalles de bajo nivel.

Para este problema se plantea una propuesta llamada **WearIt**: una solución integral para la integración de *wearables* que permite al programador usar cualquier dispositivo sin conocimiento experto, simplificando la integración y por lo tanto el propio desarrollo. Esta solución se presenta extensamente en el capítulo 7 sección 2.

La figura 39 muestra una visión general de esta etapa de implementación, con los diferentes grupos de programadores y las tareas que realiza cada grupo, así como las dos aportaciones para esta etapa de implementación (Zappa y WearIt).

Dentro del desarrollo de una plataforma de telemonitorización encontramos varios supervisores/expertos, varios ingenieros y varios desarrolladores. De este último grupo, cada uno, de acuerdo a los conocimientos que posea, tendrá asignadas diferentes tareas.

Partiendo de la distribución de funcionalidades realizada, *back-end* y aplicaciones móviles, cada uno de los desarrolladores probablemente se enmarque en uno de los dos grupos y, por lo tanto, sus tareas serán implementar funcionalidades relativas al *back-end* o bien a las aplicaciones móviles.

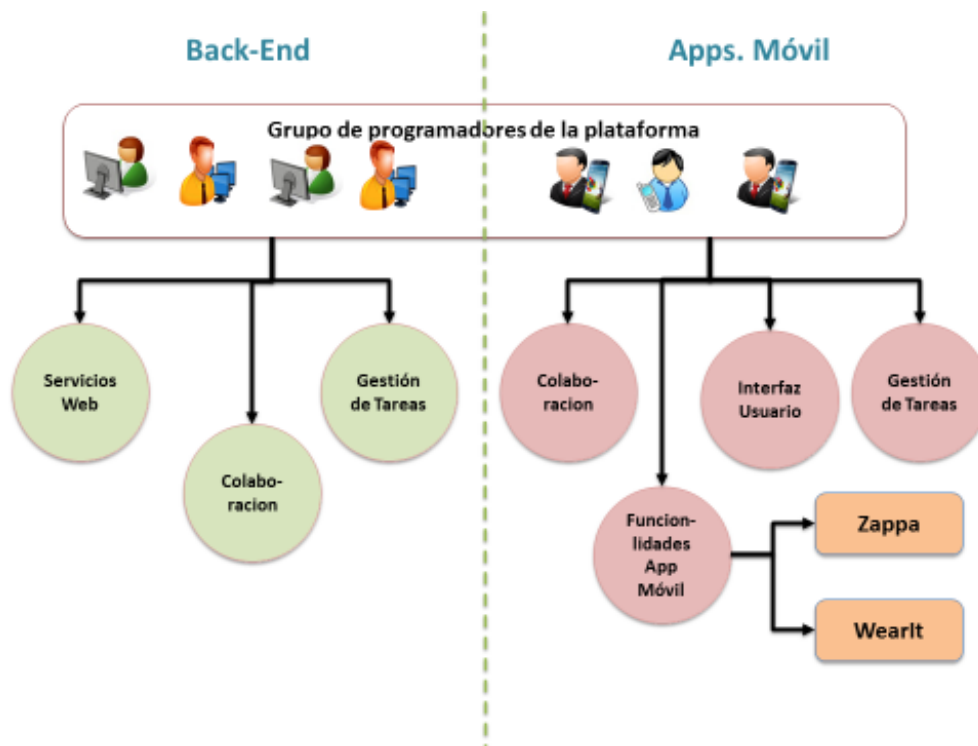


Figura 39 - Distribución de tareas por grupos para la etapa de implementación

5.5. Etapa 4: Despliegue

La cuarta etapa de la metodología consiste en el despliegue de la plataforma. Al igual que en cualquier proceso de desarrollo de software, en esta etapa se despliega la plataforma para su uso público o privado, donde los usuarios del sistema no serán ahora los propios desarrolladores, sino cualquier tipo de usuario.

Esta fase de despliegue se lleva a cabo por los ingenieros y programadores, quienes, usando la tecnología acordada en la etapa de diseño, habilitan las soluciones software que hasta ahora estaban aún en desarrollo para que comiencen las diferentes fases de testeo, tanto a nivel funcional, como de interacción.

En esta fase se habilita una primera versión final al público en un periodo de prueba (fase de testeo Beta, siendo la fase *Alpha* la realizada por los propios programadores etapa de implementación). Esta fase sirve a los programadores para detectar errores que pasaron inadvertidos en la fase *Alpha* y así poder solucionarlos antes de desplegar completamente la solución con todas las funcionalidades habilitadas, además de obtener *feedback* por parte de los usuarios de la plataforma por si fuese necesario desarrollar alguna funcionalidad no implementada o corregir algún cambio en términos de usabilidad o diseño.

Cualquier cambio a realizar o error a subsanar se volvería a plantear en una segunda iteración en la etapa que fuese necesario (especificación de requisitos, diseño o implementación), de tal manera que la metodología es bidireccional para poder volver a etapas anteriores para realizar los cambios oportunos.

5.6. Etapa 5: Validación

La última etapa de la metodología consiste en la validación de la plataforma de telemonitorización. Con esta etapa se pretende validar los distintos objetivos presentados en el capítulo 2: dependientes del dominio, de plataforma e independientes del dominio.

Muchos de estos objetivos se pueden validar por el simple uso de la plataforma, comprobando la funcionalidad y usabilidad de la misma. Otros objetivos, como reusabilidad y extensibilidad vendrán dados por el uso de las propuestas planteadas (arquitectura software, Zappa, WearIt) y otros por el uso de los modelos planteados (interacción, extensibilidad).

No obstante, algunos objetivos, como la evidencia práctica de la eficacia de la plataforma o la mejora de la calidad del servicio, deben ser validados de acuerdo a criterios objetivos, como por ejemplo estudios sobre usuarios.

Esta parte de validación es totalmente dependiente del contexto, ya que dichos procesos de validación se definen por expertos en el ámbito concreto en el que se enmarque la plataforma.

Por lo tanto, debido a que muchos objetivos se cumplen implícitamente por la aplicación del framework para el desarrollo de la plataforma de telemonitorización, la validación de objetivos contextualizados (dependientes del dominio) se deja bajo el criterio de los distintos expertos con los métodos que consideren oportunos.

5.7. Visión general y esquema de aplicación de la metodología

Con el objetivo de entender mejor la metodología propuesta en este framework y las distintas etapas, así como disponer de un resumen de la misma de cara a aplicarla en un caso real, en esta sección se muestran una serie de diagramas y figuras que pretenden servir de guía a los encargados del desarrollo.

La figura 40 muestra un resumen de las diferentes etapas de la metodología, así como los aspectos relevantes de cada etapa.



Figura 40 - Etapas y aspectos relevantes de la metodología

A continuación, se muestran diferentes figuras que pretenden resumir la aplicación de la metodología propuesta:

1. La figura 41 muestra una visión general de las distintas etapas y los distintos elementos que conforman cada etapa y el resultado de cada una.
2. La figura 42 muestra una visión de conjunto, pero centrada en que usuario forman parte o intervienen en cada etapa y los resultados de cada etapa.
3. La figura 43 muestra una guía completa para aplicar la metodología.

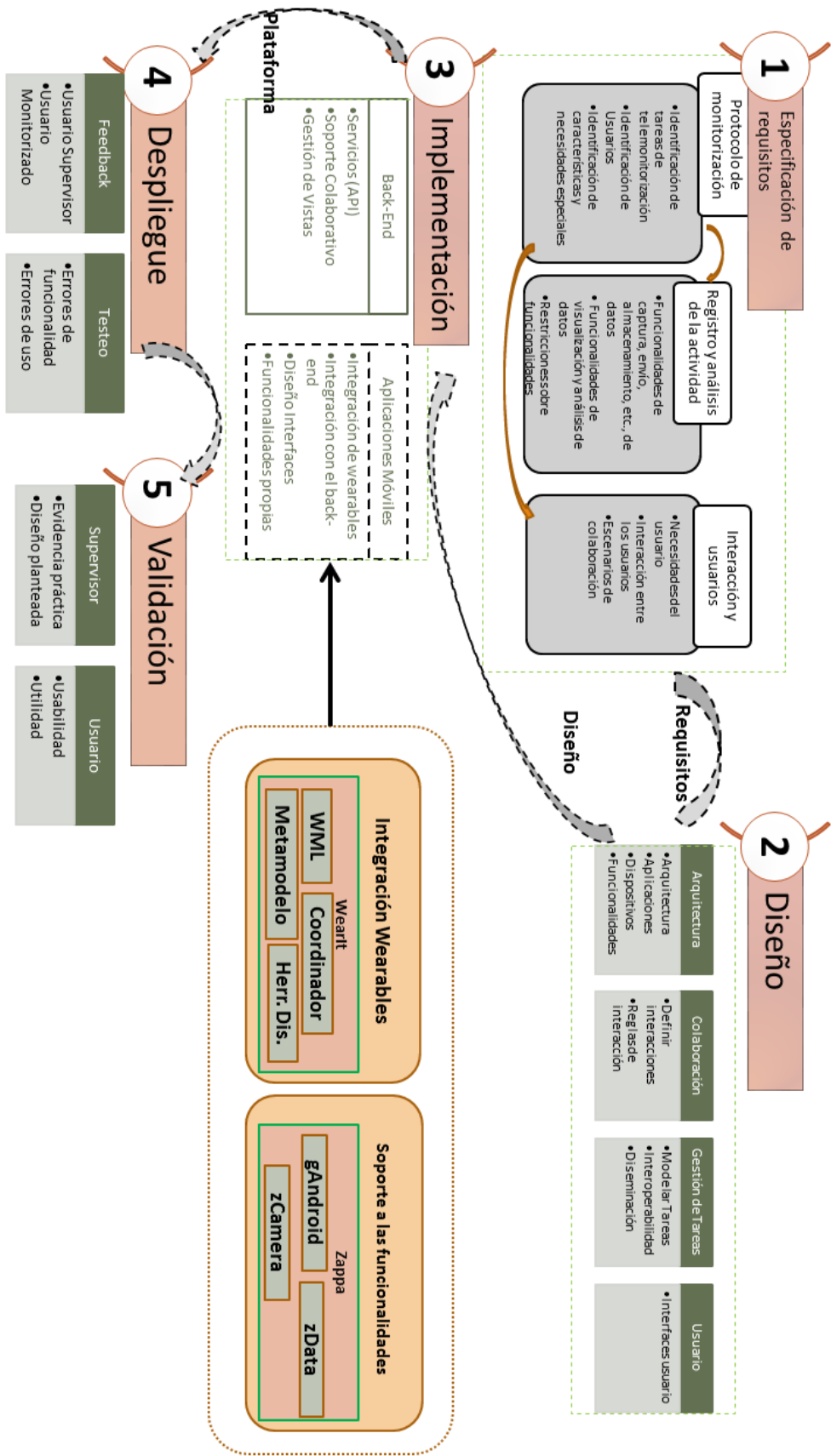


Figura 41 - Visión de conjunto de la composición de las distintas etapas de la metodología

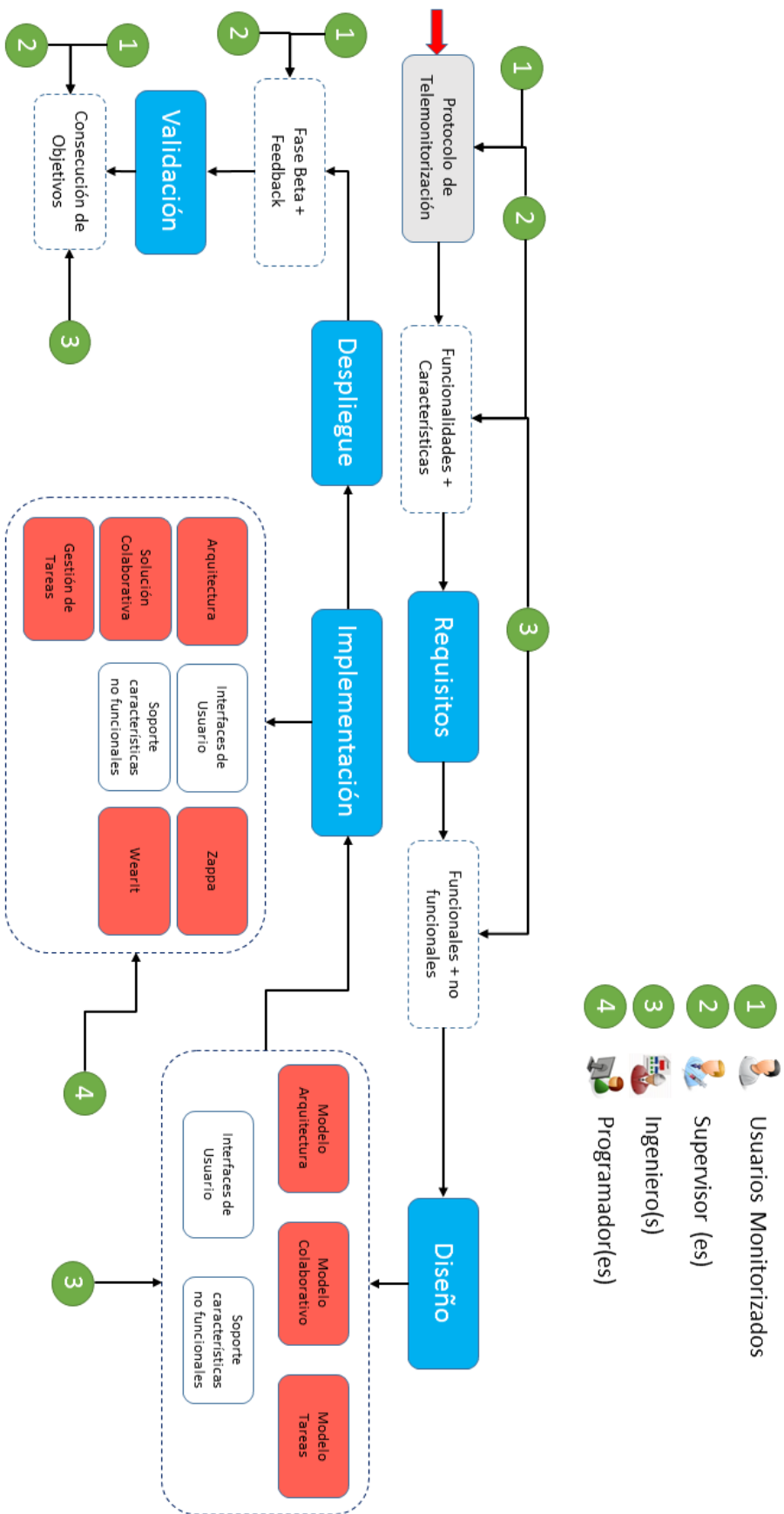
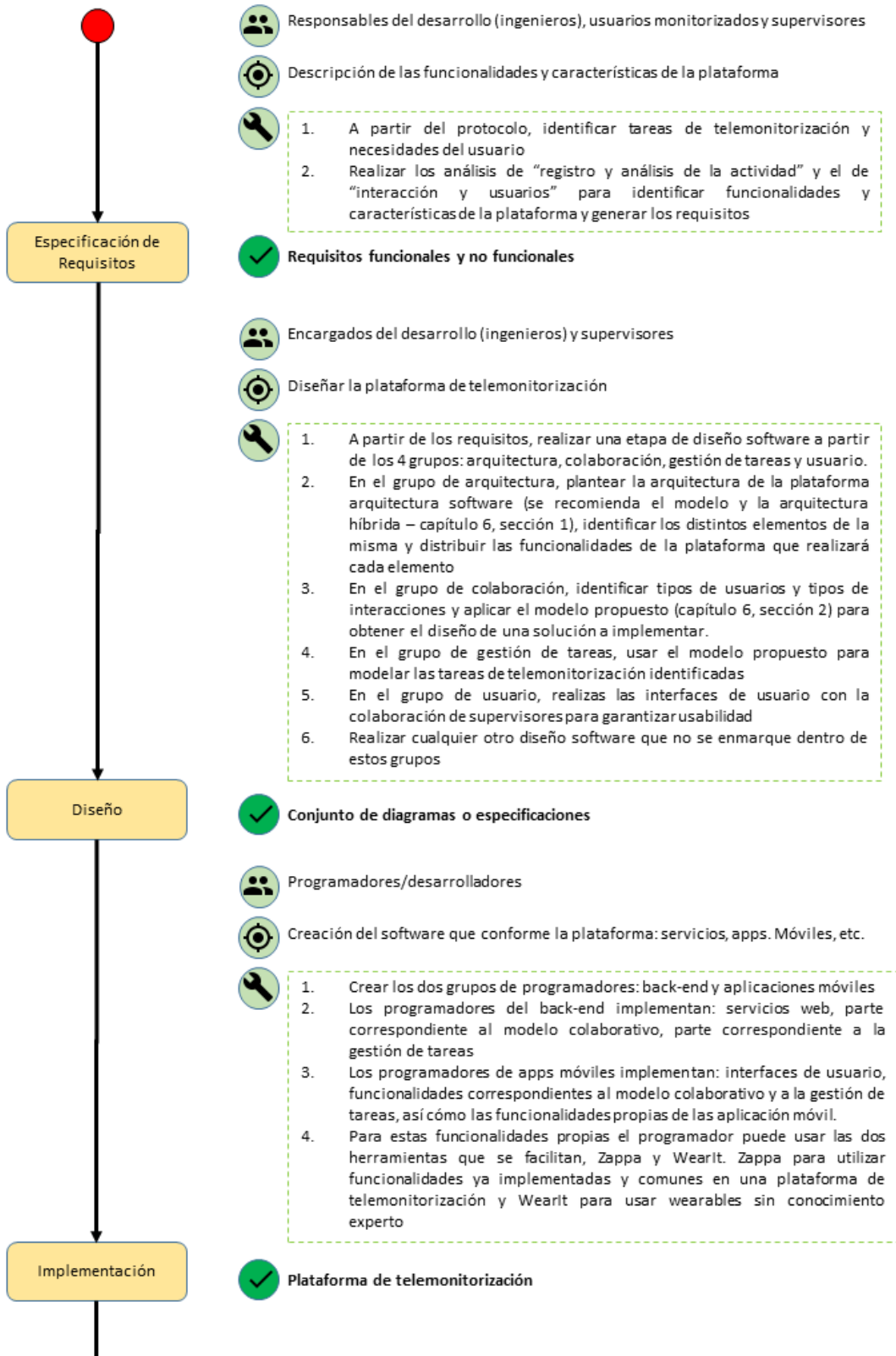
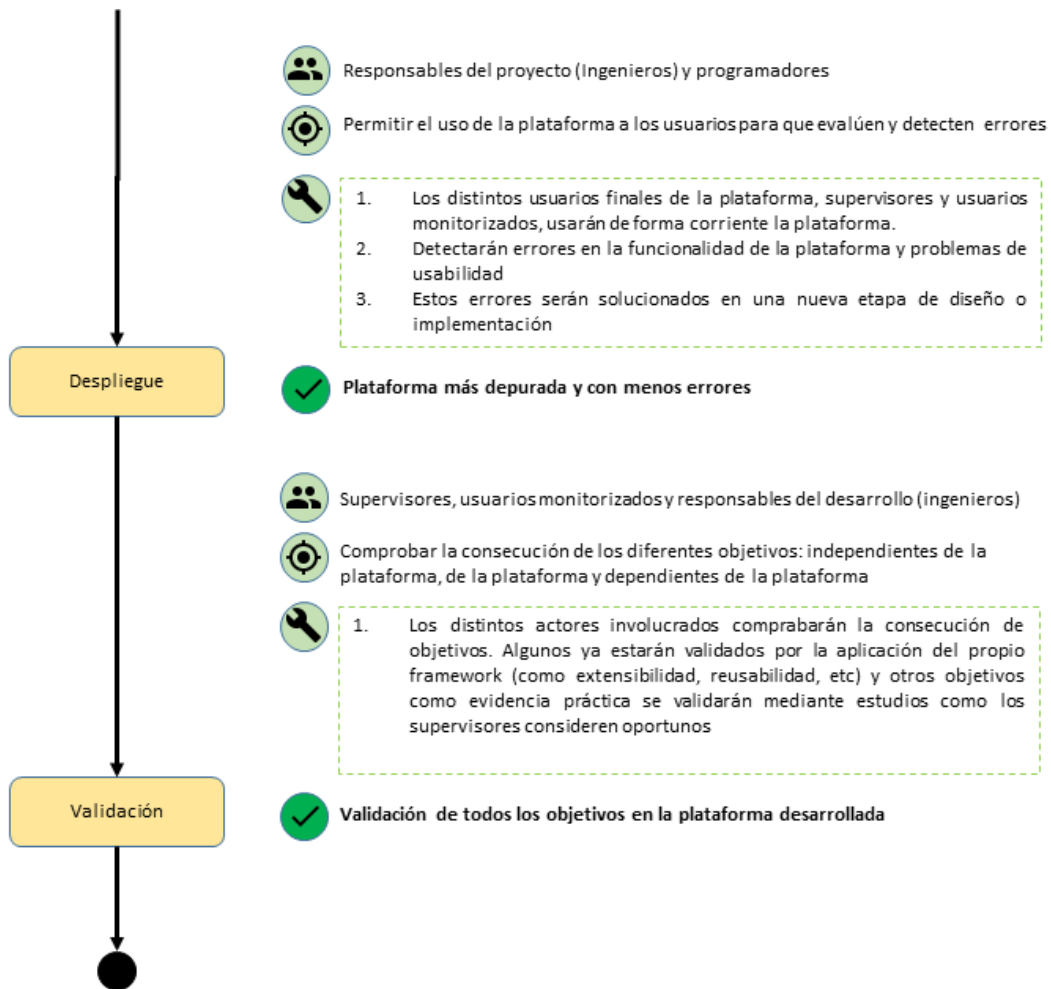


Figura 42 - Visión de conjunto de la metodología y la participación de cada actor





Leyenda

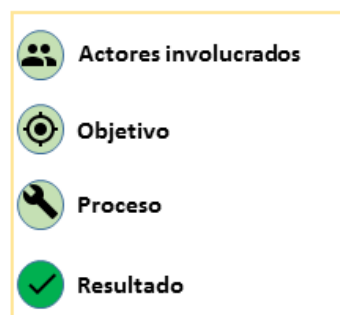


Figura 43 – Guía de aplicación de la metodología

Capítulo 6

Soporte al Diseño: Arquitectura y Modelos

6.1. Introducción

En este capítulo se presentan los diferentes elementos propuestos en la etapa de Diseño y que forman parte del framework propuesto. Estos elementos son, por orden de aparición:

- **Arquitectura:** Se presenta un modelo de arquitectura para plataformas de telemonitorización, identificando los distintos elementos hardware, módulos software habituales en las plataformas de telemonitorización, así como las conexiones entre ellos. Asimismo, se presenta una propuesta de arquitectura software que por sus características (extensibilidad, fácil despliegue, interoperabilidad) resulta adecuada para las plataformas de telemonitorización. Se pretende extender la descripción ya mencionada en el capítulo 5 acerca de la etapa de diseño de la metodología para que sirva de ayuda a ingenieros y programadores.
- **Modelo para la gestión de tareas:** Una tarea de telemonitorización da soporte a uno o varios procesos de telemonitorización definidos por los supervisores/expertos en el protocolo de telemonitorización. Una adecuada gestión de estas tareas que permita representar tareas y añadir otras nuevas para que la plataforma se adapte a la propia evolución del protocolo de telemonitorización se hace necesario de cara a garantizar el mantenimiento de la propia plataforma de telemonitorización a lo largo del tiempo.
- **Modelo de interacción para entornos colaborativos:** La interacción o comunicación directa entre supervisor y usuario monitorizado ayudan a una adecuada telemonitorización. No obstante, cada vez son más frecuentes las plataformas de telemonitorización multidisciplinares donde varios supervisores de diferentes dominios monitorizan a un usuario, con el objetivo de ofrecer un seguimiento más completo. Por este motivo, es necesario dar soporte a una correcta interacción entre supervisores y usuarios monitorizados, así como a las interacciones o intercambios de información entre supervisores de distintos ámbitos, con el objetivo de habilitar entornos colaborativos multidisciplinares dentro de las plataformas de telemonitorización [Ruiz-Zafra 2015b].

6.2. Arquitectura

Los protocolos de telemonitorización suelen evolucionar a lo largo del tiempo con el objetivo de mejorar los procesos de telemonitorización. Los cambios en el protocolo de telemonitorización hacen imprescindible disponer de soluciones flexibles que permitan en un futuro dar soporte a nuevas funcionalidades y requisitos con el objetivo de que cambios en el protocolo de telemonitorización se materialicen como ampliaciones de la plataforma de telemonitorización que lo soporta.

Siguiendo el modelo de arquitectura propuesto en el capítulo 4 (ver Figura 28), como consideración preliminar de soporte al framework e-MoDe, y considerando los distintos objetivos definidos en el capítulo 2 (dependientes del dominio, de plataforma e independientes del dominio), se pueden identificar las siguientes características deseables en una plataforma de telemonitorización y que facilitarán la labor a los desarrolladores/programadores:

- **Interoperabilidad.** La evolución de la tecnología permite que, con el paso del tiempo, nuevos dispositivos de muy diversas características puedan pasar a formar parte de una plataforma de telemonitorización: dispositivos móviles de diferentes plataformas, ordenadores personales con aplicaciones de escritorio, plataformas web, etc. En este

sentido es necesario que, independientemente de las características de los dispositivos, estos sean capaces de comunicarse entre ellos (directa o indirectamente, por ejemplo, usando un soporte externo como un servidor o la nube).

- **Extensibilidad.** Esta propiedad se refiere a la capacidad de posibilitar añadir fácilmente nuevas funcionalidades a una plataforma de telemonitorización. Esta característica es importante para dar soporte a nuevos requisitos que puedan definirse en un futuro, cumpliendo así nuevos criterios del ámbito concreto en el que se enmarque la plataforma de telemonitorización. Asegurar la extensibilidad de una plataforma de telemonitorización permite adaptarse a cambios en el protocolo.
- **Escalabilidad.** Aunque no es una propiedad imprescindible, ya que muchas soluciones de telemonitorización son dedicadas y están creadas para grupos poblacionales cerrados, es una característica a tener en cuenta en soluciones de ámbito comercial o con una gran proyección, con el objetivo de dar soporte a un número elevado e indeterminado de usuarios, sin afectar al rendimiento del sistema.

En las plataformas de telemonitorización son comunes los dispositivos electrónicos inalámbricos, siendo estos actualmente y en la mayoría de los casos dispositivos inteligentes, como, por ejemplo, smartphones. De hecho, la mayoría de las plataformas de telemonitorización actuales suelen dar lugar a configuraciones o escenarios en el que varios sensores colocados en distintas partes del cuerpo del sujeto monitorizado envían los datos que perciben a un smartphone (ver figura 28 del capítulo 4), convirtiendo a este dispositivo en imprescindible.

Además de dispositivos inteligentes, las plataformas de telemonitorización suelen integrar varios tipos de aplicaciones (para el usuario monitorizado, para los supervisores, para el administrador, etc.), que tienen que intercambiar información, en este caso a través de un servidor externo (o soporte de datos). Esto implica que el acceso a las funcionalidades debe ser por medio de interfaces bien definidas para permitir a los desarrolladores crear nuevas soluciones que sean fácilmente integrables.

Partiendo del concepto de arquitectura propuesto en [Clement 2002]: “*Conjunto de estructuras del sistema formadas por componentes con determinadas características que se relacionan entre ellos*”, se pretende definir una arquitectura que encaje con el modelo propuesto en el capítulo 4 (Figura 28), que cumpla con propiedades de escalabilidad, extensibilidad, eficiencia o reusabilidad y que integre los elementos tecnológicos propuestos en el capítulo anteriores de la presente tesis (computación en la nube, smartphones, *wearables*), por sus beneficios para el soporte a los procesos de telemonitorización.

Continuando la identificación de los distintos elementos propuestos en [Clement 2002] para la definición de una arquitectura, se definirán diferentes vistas para representar la arquitectura general para plataformas de telemonitorización. Estas vistas son:

- **Vista Hardware/software:** representando los diferentes elementos hardware que conforman el sistema, junto con los diferentes módulos o componentes de la plataforma.
- **Vista de Interacción:** representando las diferentes interacciones entre los diferentes elementos del sistema y de los diferentes componentes.

6.2.1. Vista Hardware/Software

A nivel hardware, la vista hardware/software de la arquitectura de una plataforma de telemonitorización encaja con la propuesta en el modelo del capítulo 4 (ver Figura 28), por su simplicidad y por ser uno de los modelos más usados a día de hoy.

Basada conceptualmente en una arquitectura cliente-servidor, encontramos diferentes elementos hardware: soporte de datos a través de un servidor o conjunto de servidores, dispositivos móviles u ordenadores personales para albergar el software de telemonitorización y los dispositivos de registro de la actividad (*wearable*). Estos elementos interactúan entre sí a través de los módulos software mediante de un intercambio de información que permiten la telemonitorización en sí.

La figura 44 muestra la vista general de la arquitectura de una plataforma de telemonitorización, identificando los elementos hardware (*device* en la figura) y los diferentes módulos software que se presentan como relevantes en una plataforma de telemonitorización.

En esta vista de arquitectura, y a nivel hardware, se distinguen tres tipos de subsistemas distintos: (1) soporte externo, (2) dispositivo de telemonitorización (ya sea usado por supervisores o usuarios monitorizados) y (3) *wearables*, como dispositivos independientes para determinadas funcionalidades.

En el plano del software se identifican dos niveles: software general¹ y específico².

El acceso a las funcionalidades implementadas en los submódulos de software general (soporte externo) se hace a través de una API. Estas funcionalidades, empaquetadas en diferentes componentes (gestión de tareas, notificaciones, etc.), se encargan de realizar procesamientos de cálculo y gestionar datos del soporte de datos general.

Los accesos a las funcionalidades específicas contenidas en los diferentes módulos se emplean para acceder a los datos específicos almacenados en el dispositivo, acceder a las funcionalidades generales a través de un componente/software específico y para implementar tareas propias de la aplicación.

Este modelo pretende representar los elementos básicos (software y hardware) de una arquitectura para una plataforma de telemonitorización. Los datos, tanto generales y específicos, se gestionarán conforme a un proceso de ingeniería de datos particular que determine el uso de un sistema para la gestión de información (base de datos, documentos, etc.).

¹Software general: el software utilizado por todos los actores del sistema, y que se encuentra alojado en el soporte externo

²Software específico: el software utilizado por cada uno de los actores particulares, alojado en los dispositivos de telemonitorización

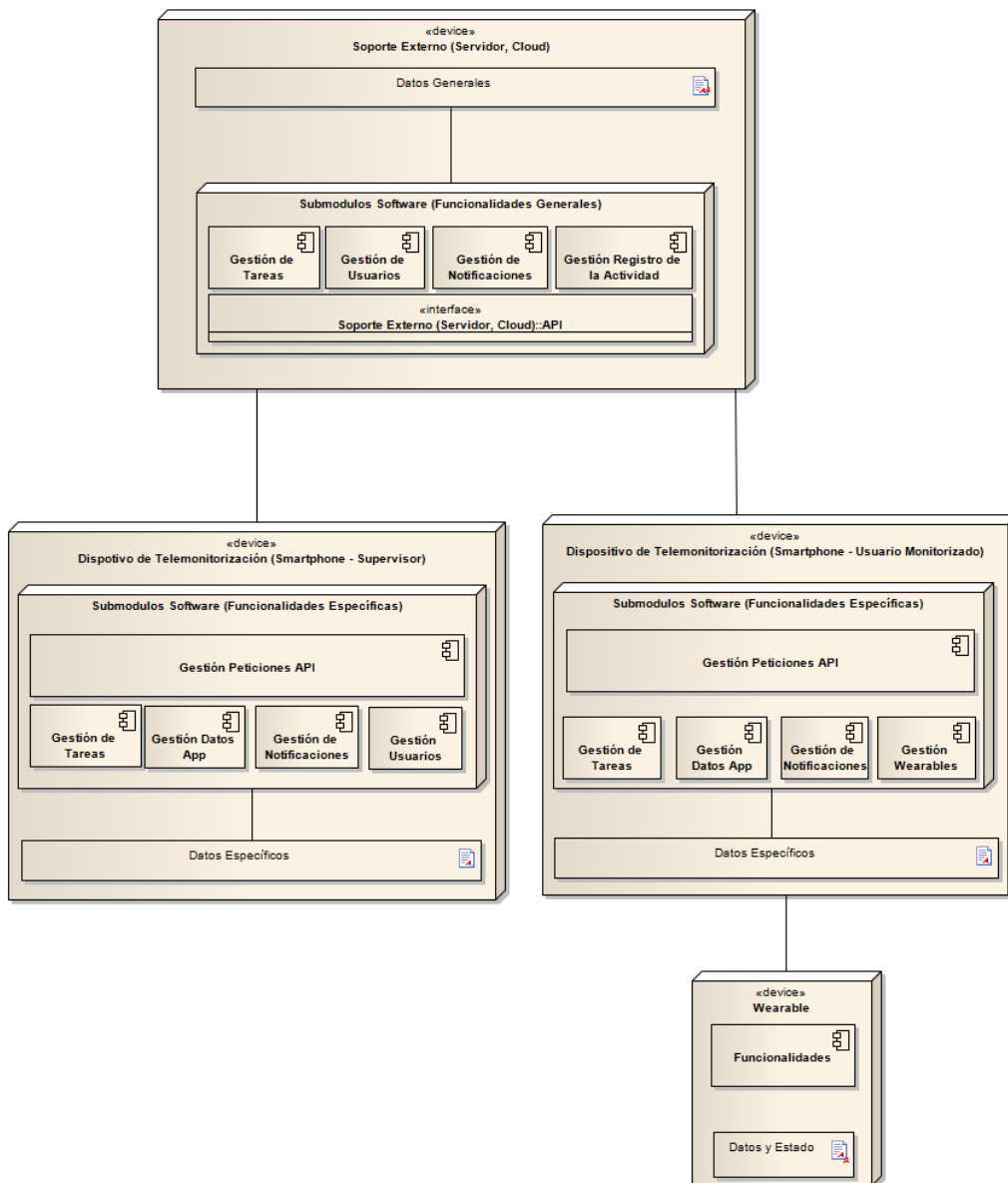


Figura 44 - Vista de la arquitectura hardware/software para plataformas de telemonitorización

Con respecto a las funcionalidades, generales y específicas, estas son bastante dependientes del contexto. No obstante, habrá módulos generales, como pueden ser módulos para la gestión de tareas de telemonitorización, de usuarios, de registro de la actividad, seguridad, etc.

Los módulos presentes en la Figura 44 son lo que, usualmente, estarán presentes en prácticamente cualquier plataforma de telemonitorización, al contener funcionalidades necesarias para los procesos de telemonitorización. Por ejemplo, el módulo para la gestión de tareas permitirá crear o asignar tareas de telemonitorización, de igual manera que el módulo para la gestión de notificaciones será crucial para el soporte a la colaboración.

Puesto que la plataforma de telemonitorización es dependiente del contexto y de la tecnología, las funcionalidades concretas de cada módulo, así como su implementación, dependerán de cada plataforma.

6.2.2. Vista de Interacción

Los módulos hardware de cada subsistema (*device*) identificados en la arquitectura interactúan entre sí a través de protocolos de comunicación, dependiendo de la tecnología de soporte a la plataforma.

Los módulos software interactúan internamente a través de las APIs de los propios componentes. En el caso de existir una interacción entre dos módulos software de diferentes subsistemas (hardware), dicha interacción se realiza a través del protocolo de comunicación correspondiente (por ejemplo, el acceso a una funcionalidad del soporte externo por parte del dispositivo de telemonitorización).

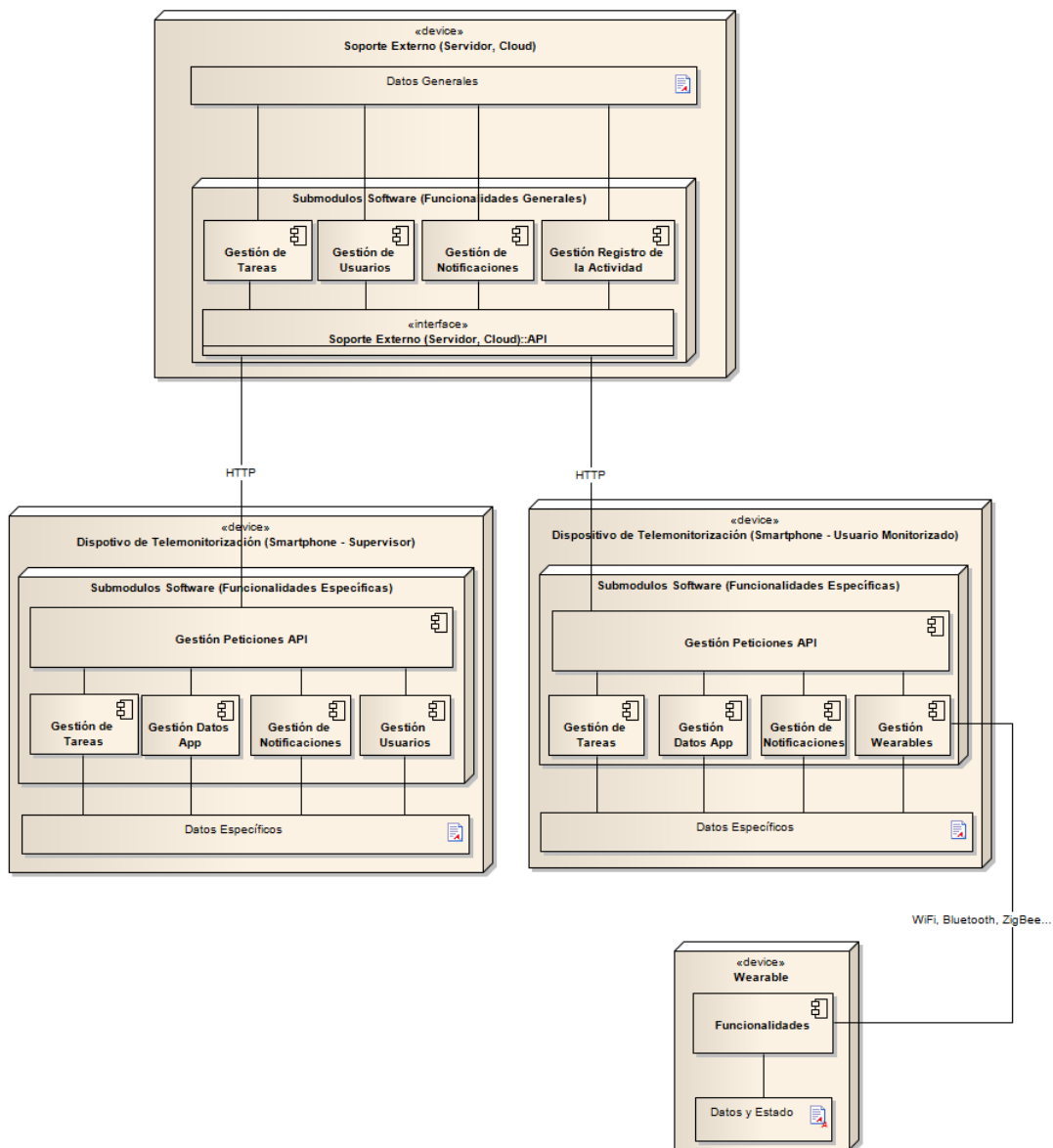


Figura 45- Interacción de los diferentes módulos software

La figura 45 muestra la interacción, por medio de diferentes protocolos (WIFI, BlueTooth, ZigBee), entre componentes o módulos software. Los diferentes componentes o módulos

software de los dispositivos móviles para la telemonitorización acceden a funcionalidades generales (soporte externo) a través de la API y usando el protocolo HTTP. De igual manera, el acceso a las funcionalidades de cada *wearable* por parte del dispositivo móvil se hace mediante un protocolo de comunicación de corto alcance como Wifi o Bluetooth.

6.2.3. Distribución de funcionalidades

Partiendo de esta vista simplificada de la arquitectura para una plataforma de telemonitorización, y con la propuesta de diseño de la metodología donde se distinguía entre funcionalidades del *back-end* o de dispositivos móviles, se hace evidente que estas funcionalidades del *back-end* corresponden con las funcionalidades generales de la arquitectura y las funcionalidades de las aplicaciones móviles con las funcionalidades específicas.

Las funcionalidades específicas, que se implementarán en el dispositivo móvil, son dependientes del contexto. En el capítulo 7 se presenta la plataforma Zappa y WearIt, herramientas software que proporcionan funcionalidades específicas ya implementadas, como por ejemplo el acceso a las funcionalidades de un *wearable*, la gestión de datos de la propia aplicación o la gestión de peticiones e información usando el protocolo HTTP.

Con respecto a las funcionalidades generales de la arquitectura, estas deben ser accesibles por cualquier dispositivo, fomentando la interoperabilidad entre componentes, además de garantizar otros objetivos ya mencionados, como extensibilidad o escalabilidad.

En el capítulo 3 se presentaron dos modelos de referencia de arquitectura software que podrían aplicarse para implementar estas funcionalidades generales. Estos dos modelos son: *Service Oriented Architecture* (SOA) [Erl 2005] y *Resource Oriented Architecture* (ROA) [Guinard 2010].

Aunque ROA es el modelo más adecuado para una plataforma de telemonitorización, por las ventajas que ya se explicaron en el capítulo 3, la limitación del modelo a las cuatro operaciones básicas (CRUD) podría complicar el desarrollo por la necesidad de tener muchos tipos de recursos distintos para representar una plataforma de telemonitorización normal.

El modelo híbrido (ROA-SOA) presentado en el capítulo 3, que ofrece la facilidad de uso e implementación de una arquitectura REST junto con un acceso a servicios web, es una opción adecuada, ya que posibilita dar soporte a cualquier funcionalidad y permite añadir nuevas funcionalidades bajo un mismo recurso en caso de que fuese necesario, posibilitando que un recurso pueda considerarse como un servicio web. Esta propiedad del diseño va a proporcionar:

- **Facilidad de acceso a los recursos.** La simplicidad para acceder a un servicio/recurso (a través de una URI), invocando una URI y esperando recibir una salida simplifica la invocación de servicios y sus funcionalidades.
- **Facilidad de despliegue.** La puesta en marcha de una arquitectura REST, por parte de los programadores, es simple por la cantidad de ayuda y soporte disponible actualmente.
- **Rendimiento-eficiencia.** El intercambio de datos, gracias a notaciones como JSON, posibilita que la petición de una funcionalidad de un servicio consuma muy pocos recursos y, además, que la respuesta de la misma esté optimizada para devolver únicamente información relevante (*payload*), sin que sean necesarias cabeceras u otra

información del protocolo para el intercambio de mensajes, como ocurre en SOAP. Esta reducción de costes es importante en sistemas móviles con recursos limitados.

- **Extensibilidad.** En caso de que la plataforma de telemonitorización requiera nuevas funcionalidades, se pueden añadir nuevos recursos que implementen funcionalidades, o bien parametrizar la especificación a recursos ya creados, con la misma finalidad. Esto es, definir nuevos parámetros para un mismo recurso con el objetivo de dotar al sistema de nuevas funcionalidades.

El paradigma de computación en la nube y más concretamente la capa SaaS (*Software as a Service*), por estar concebida y diseñada fundamentalmente en torno a servicios, puede tomarse como base tecnológica para esta arquitectura híbrida (ROA-SOA).

Aunque una plataforma de telemonitorización podría implementarse siguiendo un enfoque tradicional cliente-servidor, el uso de la estructura *cloud* garantiza todas las ventajas ya descritas anteriormente (pago por uso, escalabilidad ante incrementos de la demanda), que unidas a las características de una arquitectura software híbrida (extensibilidad, facilidad de acceso a los recursos), posibilitan que cualquier plataforma amplíe su funcionalidad sencillamente añadiendo nuevos servicios para ser consumidos por cualquier desarrollador fácilmente.

Asimismo, la computación en la nube posibilita escalar automáticamente y de manera transparente al usar los recursos que se vayan necesitando bajo demanda, esto es, ir ampliando las capacidades de computación y almacenamiento conforme sea necesario (por un incremento en el número de usuarios, de funcionalidades, etc.).

Esta capacidad de escalado automático de los recursos, tanto de procesamiento, como de almacenamiento, a largo plazo conlleva una reducción de costes, ya que los desarrolladores no tienen que planificar ampliaciones del sistema (recursos) cada cierto tiempo, a diferencia de como ocurre en sistemas clásicos (migraciones de datos a otros modelos con más capacidad, mover todo el software a otro servidor con más potencia, etc.), sino que esto se gestiona automáticamente por el modelo de capas de la nube (IaaS – PaaS – SaaS) de manera transparente al desarrollador.

Un ejemplo de la aplicación de este modelo de arquitectura híbrido (ROA-SOA) puede ser el proceso de desarrollo de una plataforma para pacientes con problemas pulmonares crónicos.

Siguiendo la metodología propuesta en el framework (capítulo 5), en la primera fase, se identifican, entre otras cosas, las tareas de telemonitorización. Por ejemplo, llevar a cabo una sesión de telemonitorización que consiste en andar diez minutos y almacenar el valor medio de pulsaciones cardíacas de esa caminata. De esta tarea de telemonitorización los ingenieros identifican dos recursos, “caminar” y “pulso cardíaco”, y las siguientes opciones sobre estos: guardar y recuperar caminata, y, guardar y recuperar pulso cardíaco asociado a una caminata.

A partir de estos elementos y opciones, los ingenieros diseñan la APIs (siguiendo el modelo híbrido) que a posteriori los programadores implementarán bajo la tecnología seleccionada en la etapa de diseño (ver Figura 46). Esto es, a partir de los diferentes recursos, dotar a cada uno de un identificador (*caminar* y *pulsocardíaco* en la figura) y definir los diferentes parámetros

para representar todas las funcionalidades necesarias (guardar y recuperar). En el caso de recuperar se accede mediante el recurso y el parámetro *identificador* y para almacenar o guardar información se hace uso de una serie de parámetros donde se presenta la información a almacenar.

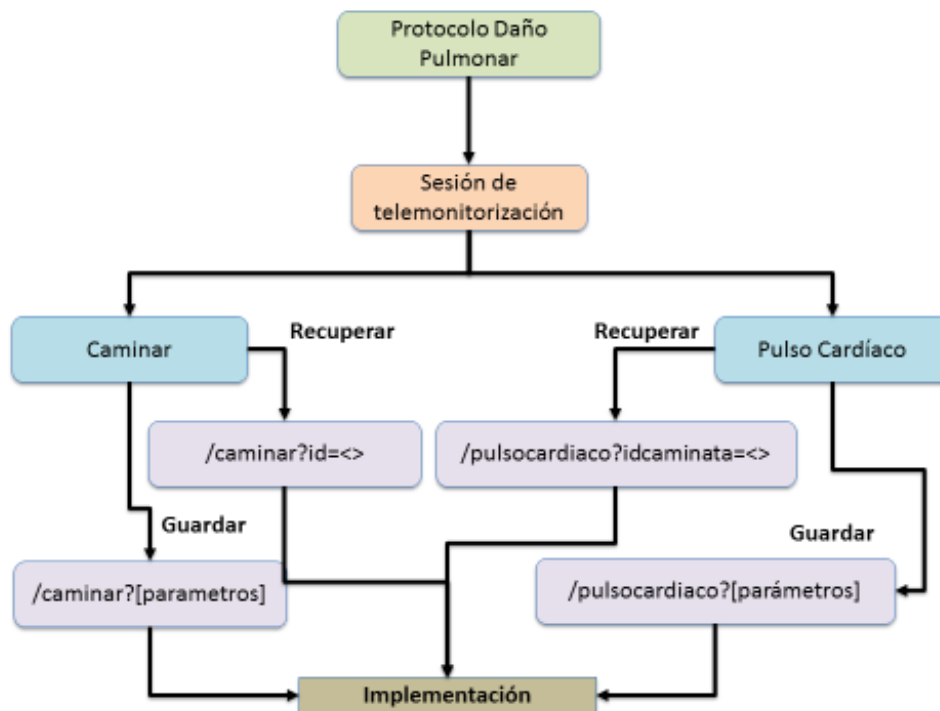


Figura 46 - Ejemplo de identificación de tareas y diseño de la API para el acceso a las funcionalidades

Usando esta arquitectura híbrida con soporte *cloud*, la arquitectura software de una plataforma de telemonitorización (una vez conocida la metodología para el desarrollo), estaría representada en la figura 47.

Esto es, a partir del protocolo de telemonitorización (definido por supervisores y usuarios monitorizados) los ingenieros definen las diferentes funcionalidades que tendrá la plataforma. Con estas funcionalidades definen cuales se implementará en el *back-end* y cuales en los dispositivos móviles. Para las funcionalidades derivadas al *back-end* se diseñan las diferentes APIs, siguiendo el modelo híbrido (SOA-ROA), y este diseño se les pasa a los programadores.

A partir de la API los programadores *del back-end* implementarán dichas funcionalidades, y los programadores de las aplicaciones móviles usarían en los dispositivos móviles dichas funcionalidades accediendo a través de la API.

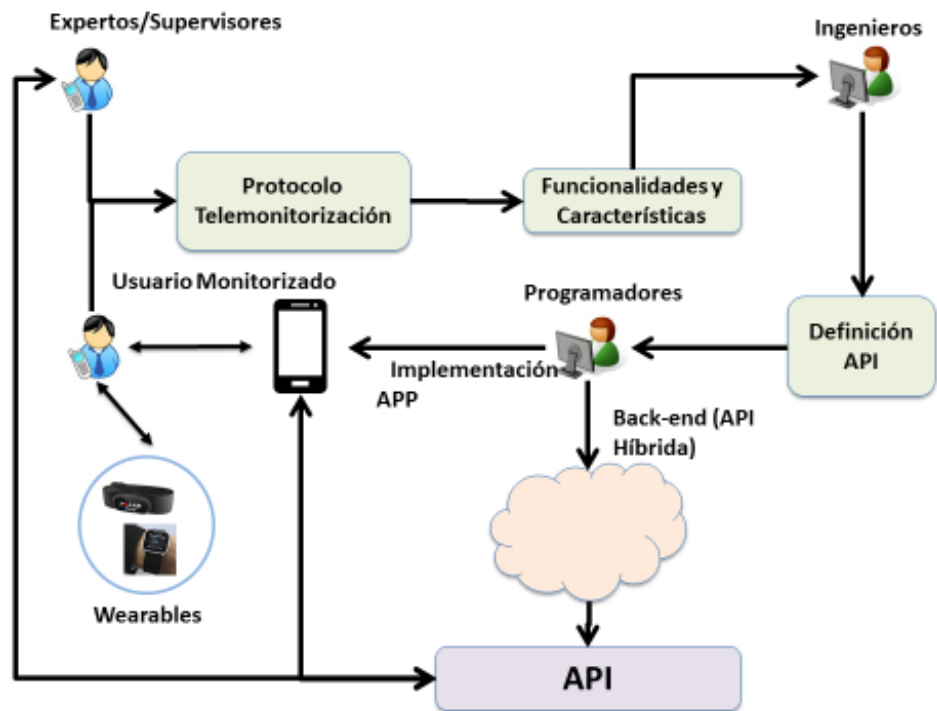


Figura 47 - Arquitectura híbrida con soporte cloud para plataformas de telemonitorización

De esta manera, los programadores no tienen que conocer los detalles de todas las etapas anteriores (especificación de requisitos y diseño), de igual manera que a los expertos/supervisores y usuarios monitorizados de la plataforma se les ocultan detalles sobre las tecnologías usadas.

Además, este tipo de arquitectura permite diseñar las funcionalidades de forma extensible y añadir nuevos roles de usuario (diferentes tipos de usuarios monitorizados, supervisores u otro tipo de usuarios que formen parte del entorno de telemonitorización como, por ejemplo, familiares).

En efecto, la inclusión de un nuevo rol a la plataforma normalmente se propondrá por el personal experto (supervisor) correspondiente, que determinará en última instancia los requisitos de la plataforma y si se quiere dar soporte a otro tipo de usuario (un nuevo tipo de usuario monitorizado, supervisor u otro rol con otra categoría distinta).

De este nuevo rol se obtendrán los requisitos acerca de las funciones y sus objetivos. A partir de estos requisitos, los diseñadores establecerán las funcionalidades en la fase de diseño que derivarán en una serie de servicios, que a posteriori serán implementados por los programadores, añadidos a los ya existentes y desplegados en la nube.

A nivel de desarrollo, un rol de usuario se concibe como un conjunto de funcionalidades que, esto es, las nuevas tareas o responsabilidades que tenga un rol dentro de la plataforma corresponden con un conjunto de funcionalidades para dotar a la plataforma de dichas tareas o responsabilidades.

La inclusión en la plataforma de un rol nuevo a soportar se traduce en la adición de nuevos servicios que se desplegarán en la nube y estarán accesibles mediante su correspondiente API

(ver Figura 48). Esto es, a partir de un rol se identifican los nuevos requisitos de donde se obtendrán las funcionalidades, y éstas, siguiendo el proceso mencionado (ver Figura 45) se implementan como nuevos servicios (*back-end*) o bien en el propio dispositivo móvil.

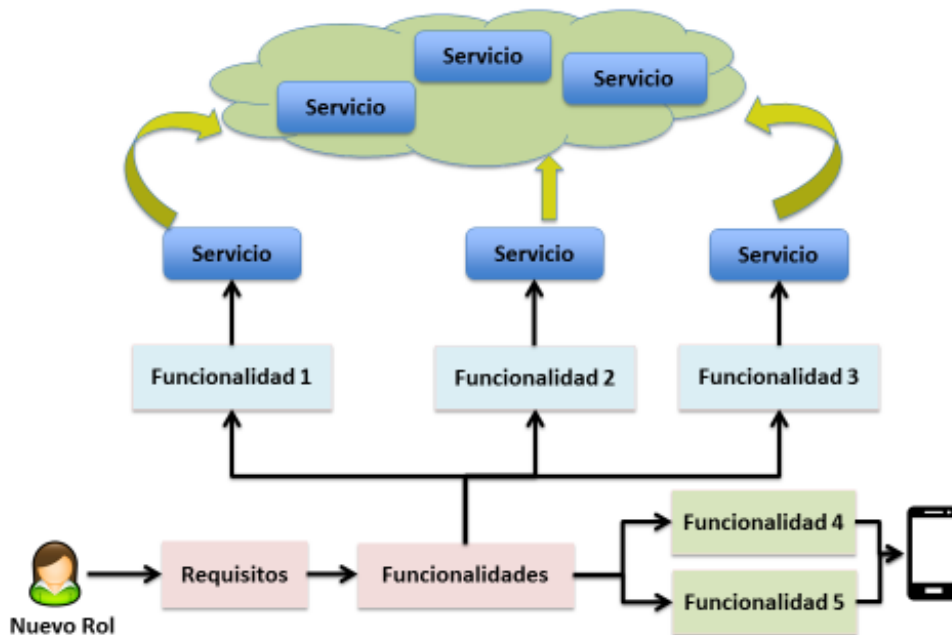


Figura 48- Proceso para dar soporte a un nuevo rol

Conviene destacar que no se propone una arquitectura software propia, puesto que dicha arquitectura software es extensamente conocida, sino la forma de adoptarla para alcanzar varios de los objetivos (independientes del dominio, de plataforma y dependientes del dominio) además de ahorrar esfuerzos de diseño a los ingenieros de software y simplificar significativamente la implementación a los programadores.

6.3. Modelo para la gestión de tareas de telemonitorización

6.3.1. Introducción

Un proceso de telemonitorización entre supervisor y usuario monitorizado consta de una serie de tareas o actividades que el usuario monitorizado realiza y que el supervisor examina para determinar su evolución.

En una plataforma de telemonitorización dichas tareas se realizan con la ayuda de las TIC, esto es, dispositivos electrónicos y software (aplicaciones móviles, de escritorio, etc.). Esto facilita la realización de la tarea por parte del usuario monitorizado y automatiza la toma y registro de datos, además de mejorar la continuidad del seguimiento.

Uno de los aspectos más relevantes en las plataformas de telemonitorización se refiere a la gestión de estas tareas, ya que en última instancia materializan la forma en que los usuarios monitorizados utilizan la plataforma cada vez. Estas tareas pueden ir desde registrar el pulso cardíaco hasta completar un cuestionario como parte de un entrenamiento deportivo.

Debido al amplio abanico de posibilidades que encajarían en el concepto de tarea dentro de un ámbito de telemonitorización, se hace necesario definir modelo para las mismas

independientemente del dominio y tecnología, para permitir la representación y gestión de cualquier tipo de tarea en cualquier plataforma de telemonitorización.

6.3.2. Proceso para la gestión de tareas

Los supervisores y expertos definen las diferentes tareas que podrán llevar a cabo los usuarios monitorizados, quedando éstas plasmadas en el protocolo de telemonitorización y especificadas en la etapa de especificación de requisitos (etapa 1) de la metodología.

En la etapa de diseño, los ingenieros de software toman esta especificación de tareas, junto con los requisitos funcionales y no funcionales, para definir la API de los diferentes servicios que dan soporte a las tareas de la plataforma, siguiendo el modelo de interfaz orientado a recursos de la arquitectura híbrida propuesta (SOA-ROA).

Asimismo, se definen las distintas interfaces de usuario para la realización de las tareas, siendo posible la participación del propio usuario monitorizado en este proceso con el objetivo de considerar su *feedback* para, por ejemplo, determinar su satisfacción o comodidad.

Posteriormente, la API diseñada en la etapa de diseño, junto con las especificaciones necesarias, se toma como parámetro de entrada para la etapa de implementación para que los programadores del *back-end* implementen los diferentes servicios que den soporte una tarea concreta (ver Figura 49).

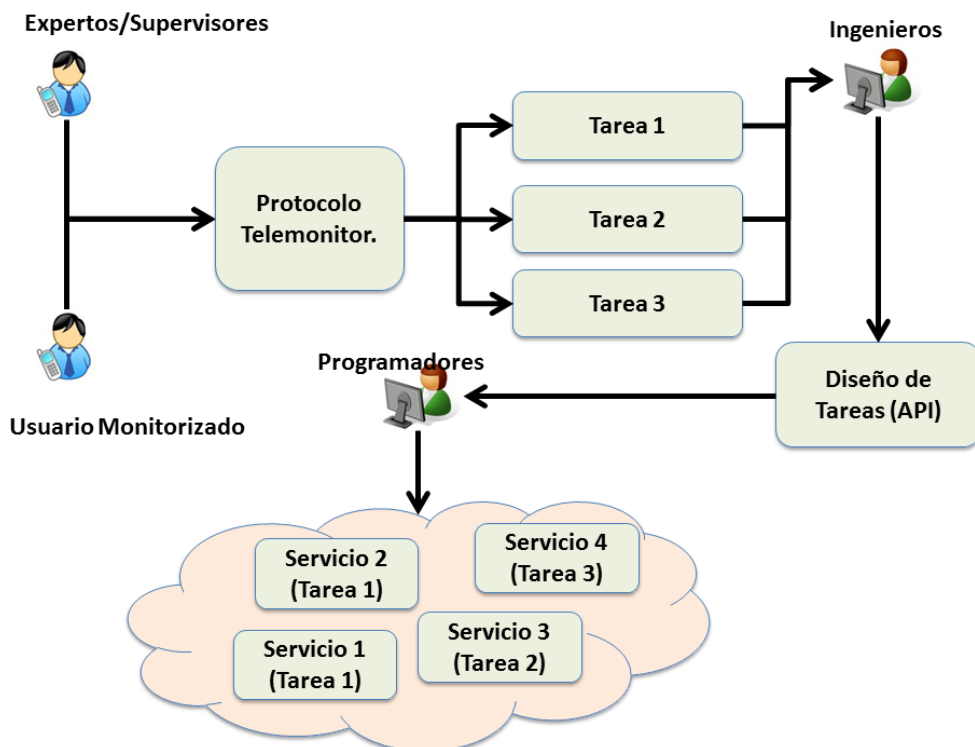


Figura 49 - Proceso de identificación y diseño de tareas de telemonitorización

Este proceso de creación de tareas no resulta complejo, al ser un proceso de creación de software (*back-end*) a medida para cada tarea, esto es, implementar las funcionalidades o servicios necesarios para dar soporte a esa tarea. No obstante, esto ocasiona que la creación de una tarea se reduzca a la implementación de código y, por lo tanto, sea muy dependiente

de la plataforma donde esté implementada o la tecnología utilizada, tanto a nivel de implementación, como gestión de datos o representación.

Para abordar esta problemática, se ha diseñado un modelo de representación de tareas de telemonitorización con el objetivo de que el diseño de éstas sea independiente de plataformas o aplicaciones particulares, para poder así compartir tareas entre diferentes plataformas de telemonitorización.

6.3.3. Modelo de Tareas

Cuando los supervisores o expertos definen un protocolo de telemonitorización, especifican, entre otras cosas, los diferentes procesos que permiten alcanzar los objetivos de la telemonitorización.

Una tarea de telemonitorización es una acción o actividad definida por uno o varios supervisores, realizada por un usuario y con un objetivo particular, como puede ser anotar el pulso cardíaco, grabarse realizando una actividad física, consultar cierta información en el dispositivo, responder una pregunta, reproducir un documento multimedia, etc.

Normalmente, dentro de un proceso de telemonitorización se debe realizar un conjunto de tareas para alcanzar un único objetivo. A este conjunto de tareas se le conoce como **sesión de telemonitorización**.

Aunque el concepto de tarea es simple, en Ingeniería del Software se plantea el desafío sobre cómo representar una tarea para hacerla independiente de la plataforma y permitir, además, representar cualquier tarea, independientemente del contexto.

En el marco de esta tesis una tarea de telemonitorización se representa como un conjunto de datos, estos son, los diferentes atributos que identifican la tarea y los valores de estos.

Por ejemplo, la tarea “*andar tres kilómetros y registrar el pulso cardíaco medio*” se puede representar a través de una distancia recorrida y unos valores de pulso cardíaco, que se pueden representar con datos primitivos (*int, boolean, char*, etc.). No obstante, la gestión de tiempo, determinar la distancia recorrida mediante el uso del GPS y el pulso cardíaco depende de la implementación de la tarea en la plataforma o aplicación correspondiente.

En este capítulo se presenta un metamodelo (ver Figura 50) para la representación de tareas y el conjunto de procesos para permitir su monitorización desde interfaces de usuario. La lógica de control de la tarea, esto es, la implementación necesaria para permitir su realización por parte del usuario monitorizado se dejará a cargo del programador.

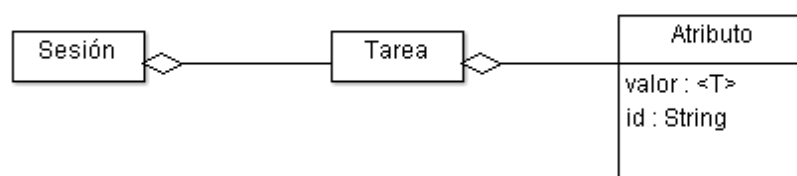


Figura 50 - Metamodelo para la representación de tareas de telemonitorización

Una sesión de telemonitorización está compuesta por un conjunto de tareas, donde cada una de ellas estará compuesta por un conjunto de atributos. Un atributo se define como un par $\langle \text{identificador}, \text{valor} \rangle$ donde *identificador* etiqueta una característica de la tarea (distancia, nombre, etc.), y *valor*, que puede ser o bien un tipo primitivo, un vector o incluso una tarea (conjunto de pares $\langle \text{identificador}, \text{valor} \rangle$), es el valor de dicha característica.

De esta manera, la representación de una tarea consiste en identificar los distintos atributos que conforman la tarea (identificador y valor).

Los ingenieros, junto con los supervisores si fuese necesario, identifican los distintos elementos que definirán cada una de las tareas. Siguiendo el metamodelo propuesto (Figura 48), se genera la representación de cada una de las tareas (modelos) con los atributos correspondientes (características propias de la tarea) más los atributos que sean necesarios para representar una tarea, como puede ser el *nombre o título*.

Un ejemplo podría ser la siguiente situación (ver Figura 51): en un protocolo de telemonitorización del ámbito del deporte para el seguimiento remoto de entrenamientos deportivos, una sesión de telemonitorización (llamada entrenamiento en este ámbito), tiene dos tareas, (1) carrera continua de duración determinada (especificada en segundos, por ejemplo), registrando el pulso cardíaco mínimo, máximo y medio, así como la distancia y posiciones del atleta (GPS) y , (2) responder a una pregunta multirespuesta formada por tres preguntas.

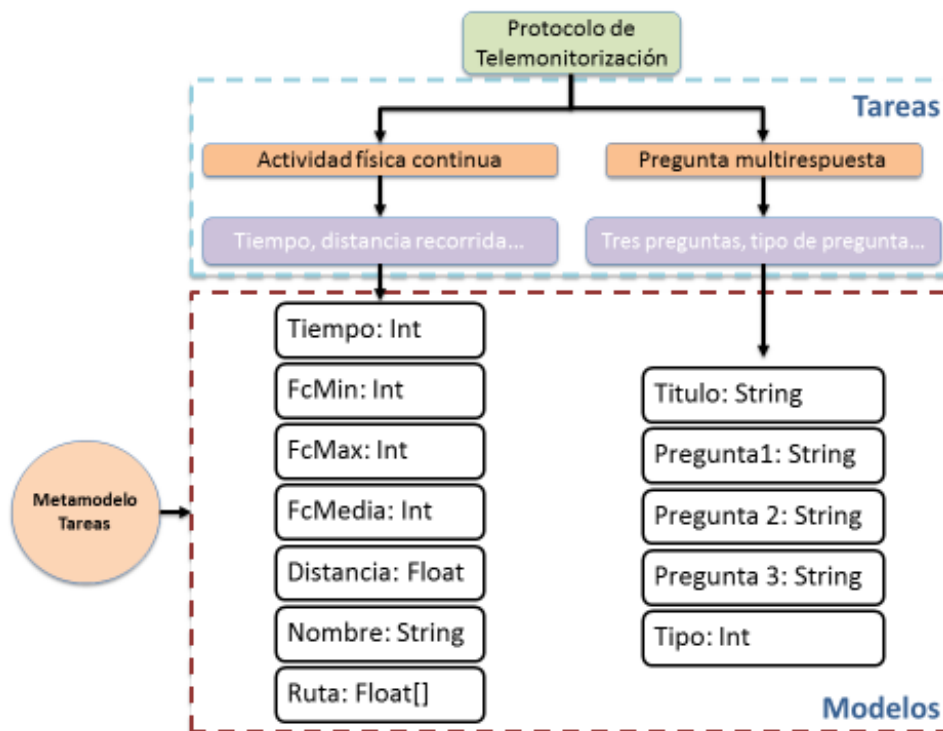


Figura 51 - Identificación y representación de tareas siguiendo el modelo

En la figura 51, partiendo del protocolo de telemonitorización, se identifican las dos tareas. De la descripción de estas dos tareas, se obtienen los diferentes atributos que es necesario considerar para representar la tarea: la distancia recorrida, el tiempo de realización de la actividad física, el pulso cardíaco mínimo y máximo de dicha actividad, etc.

Con esta identificación de los diferentes atributos que definen la tarea, se define para cada uno de ellos un identificador (distancia, tiempo, *fcmin*, *fcmax*, *pregunta1*, *pregunta3*, tipo, etc.) y se le asigna un tipo de dato acorde al atributo. Por ejemplo, la distancia se consideraría en metros (número real, para poder representar fracciones) y el pulso cardíaco como un valor entero.

Estos conjuntos de atributos conforman el modelo de la tarea (instancia del metamodelo), obteniendo así una representación para la tarea independiente del contexto y la tecnología.

6.3.4. MVC como soporte a la gestión de modelos de tareas

El metamodelo presentado en la figura 50 para la representación de tareas de telemonitorización permite representar dichas tareas de forma independiente a la tecnología.

No obstante, en una plataforma de telemonitorización, es necesario tener una representación “tangible” de los distintos modelos generados, ya que las aplicaciones trabajan con estos modelos con el objetivo de poder representar correctamente tareas a través de la interfaz de usuario, permitiendo así ser realizadas por los usuarios monitorizados.

De cara a materializar (o implementar) el modelo de una tarea, lo más adecuado es usar una notación cuya adaptación a partir del modelo (modelo → software) sea lo más automático y directo posible. Como recomendación a la propuesta del metamodelo propuesto se recomienda usar notaciones como JSON o XML (prioritariamente JSON al tener la misma potencia de expresión y por lo tanto ser más adecuada para dispositivos móviles, al reducir la cantidad total de información transferida). Además, la notación JSON sigue el mismo diseño de elementos (*identificador: valor*), por lo que la adaptación del modelo a esta notación es automática.

Esto permitirá pasar el modelo de una tarea (Figura 51) a un documento con estructura ya definida como es JSON, que además garantiza la interoperabilidad, al ser independiente de la plataforma (Figura 52).

La tecnología JSON nos va a permitir, además, que la transformación de JSON a un objeto software sea casi automática gracias a las distintas librerías (incluso nativas) de la mayoría de los lenguajes de programación. Esto ocasiona que la transformación del modelo de una tarea a un objeto software (en un lenguaje concreto) se simplifique, y se pueda hacer sin software adicional en la mayoría de las ocasiones.

De esta manera, a nivel de diseño se plantean los modelos de las distintas tareas para que en la etapa de implementación los programadores codifican el software que genere los documentos JSON correspondientes a los modelos de las tareas.

En una plataforma de telemonitorización, donde se han identificado y diseñado los modelos de las tareas, se habrán especificado las tareas en JSON. Usando las herramientas software que se facilitan dentro de la propia plataforma, los supervisores crearán nuevas tareas. Esto es, a partir de dicha herramienta software los supervisores generan documentos JSON que representan tareas concretas a partir del modelo planteado en el diseño.

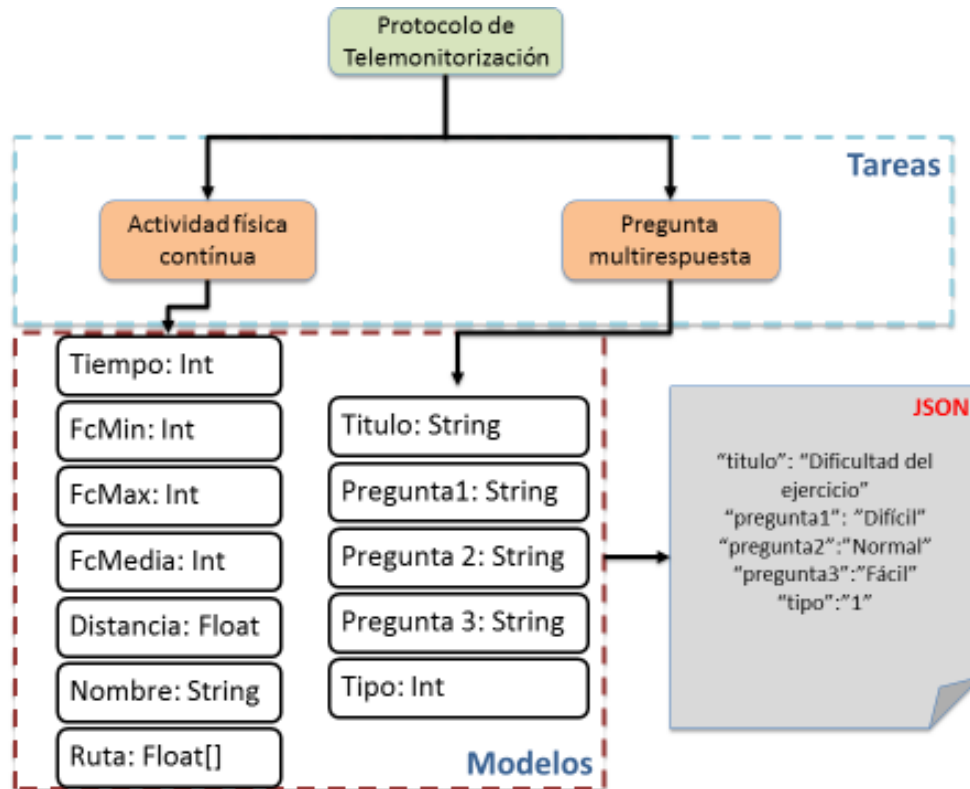


Figura 52 - Representación de una tarea en JSON a partir del modelo

De la misma manera, esta representación de la tarea en JSON, y que se almacena en la nube (siguiendo la arquitectura propuesta en la sección anterior 6, sección 1), es procesada a través de los distintos servicios de la plataforma por las distintas aplicaciones. Una vez consumida, las aplicaciones la procesan y la muestran con un interfaz concreto.

Esto permite que el modelo de una tarea representado en JSON y con un estado concreto, pueda tener diferentes disseminaciones dependiendo del usuario que la vaya a desarrollar dentro de un proceso de telemonitorización.

No obstante, aunque este metamodelo permite representar cualquier tarea, y las instancias de este (modelos de la tarea) se pueden representar en JSON para obtener la implementación de una tarea, es necesario implementar la lógica de control, esto es, los procesos que permiten que sea posible para el usuario llevar a cabo dicha tarea.

Por lo tanto, partiendo del modelo de una tarea, se propone el patrón de arquitectura conocido como Modelo-Vista-Controlador (MVC) [Krasner 1988], para separar este modelo, de la vista que lo representa y del controlador del mismo.

Este patrón está formado por tres elementos: Controlador (Lógica de control), Modelo (datos) y la Vista (interfaz). El modelo corresponde a la representación en JSON de la tarea, la vista de cómo se representan estos datos (JSON) se corresponde con la interfaz de usuario y el controlador es el encargado de gestionar la interacción con el usuario y la ejecución de la propia tarea (lógica de control).

La lógica se deja a expensas del programador, ya que este tiene que implementar el software correspondiente para dar soporte a esa tarea: interacción con la interfaz de la aplicación o de la tarea, gestión de datos, etc.

La parte de la interfaz de usuario tiene especial relevancia en este aspecto, ya que, además de permitir representar visualmente el modelo de la tarea, también debe cumplir ciertos criterios de usabilidad. Para esto se propone que las distintas interfaces se implementen, en la plataforma correspondiente, como plantillas / *template* (interfaz que muestra la estructura, pero no el contenido) para además de la usabilidad (por el propio diseño), permitir la reusabilidad y adaptabilidad a cambios en la propia tarea (datos-JSON).

De esta manera, la representación de una tarea, gracias a JSON (datos), es procesada por la aplicación móvil, a través de un servicio, generando una vista usando dicho modelo y un *template* (interfaz). A partir de aquí, el controlador se encarga de gestionar las peticiones y actualizaciones sobre el modelo de acuerdo a la ejecución de la tarea, y una vez finalizada, el modelo actualizado se guarda para ser consultado por supervisores/expertos (ver Figura 53).

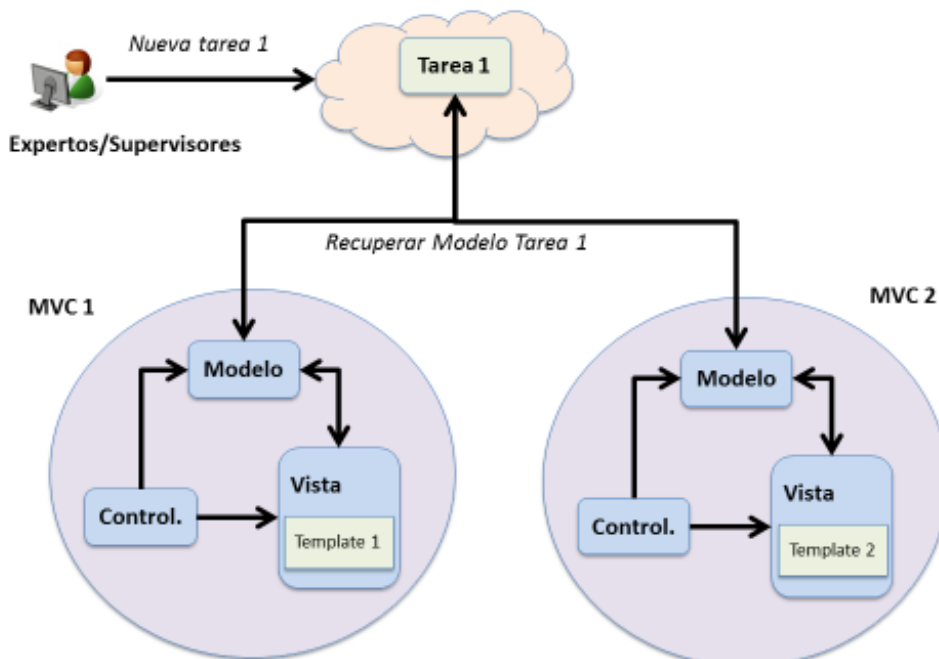


Figura 53 - Gestión de tareas siguiendo un MVC

La creación de una nueva tarea da lugar a un único modelo de representación (instancia del metamodelo), que una vez representado en JSON se utiliza como modelo en el dispositivo móvil, gracias al MVC. A partir de este modelo y al *template* de la aplicación o usuario monitorizado para ese tipo de tareas, esta tarea tendrá una representación u otra, para adaptarse a las necesidades del usuario monitorizado.

Esta manera de codificar una tarea como un JSON garantiza que cambios en una vista no afectan al modelo, y viceversa. Si una plantilla (*template*), que representa la interfaz/vista de una tarea, se cambia para mejorarla en términos de usabilidad, esto no afecta al modelo. De

igual manera que si hay cambios en el estado de una tarea, no afecta a la plantilla y por lo tanto a la estructura de la interfaz.

Siguiendo la metodología aplicada e implementando la arquitectura software recomendada, la adición de una nueva tarea implicaría seguir los siguientes pasos (ver Figura 54):

- Los ingenieros, partiendo del protocolo, definen los requisitos y características de la tarea, con la ayuda de supervisores si fuese necesario.
- Los ingenieros, en la etapa de diseño, partiendo de los requisitos diseñan nuevos servicios para representar todas las funcionalidades que puedan permitir la realización de esa tarea y el modelo de la misma. En esta misma etapa se diseña la interfaz de usuario para dicha tarea, que posteriormente se implementará por los programadores.
- Estos nuevos servicios y la especificación del modelo se les pasa a los programadores. Los programadores del *back-end* implementarán las funcionalidades necesarias y el servicio para consumir esa tarea, que además genera el modelo de la misma (JSON). Los programadores de la aplicación móvil con la ayuda del personal experto, diseñadores o un diseñador gráfico implementarán, por un lado, el *template* y, por otro lado, la lógica de gestión de la tarea en el controlador, esto es, cómo se almacena su estado, qué acciones tomar dependiendo de las acciones del usuario monitorizado, etc.

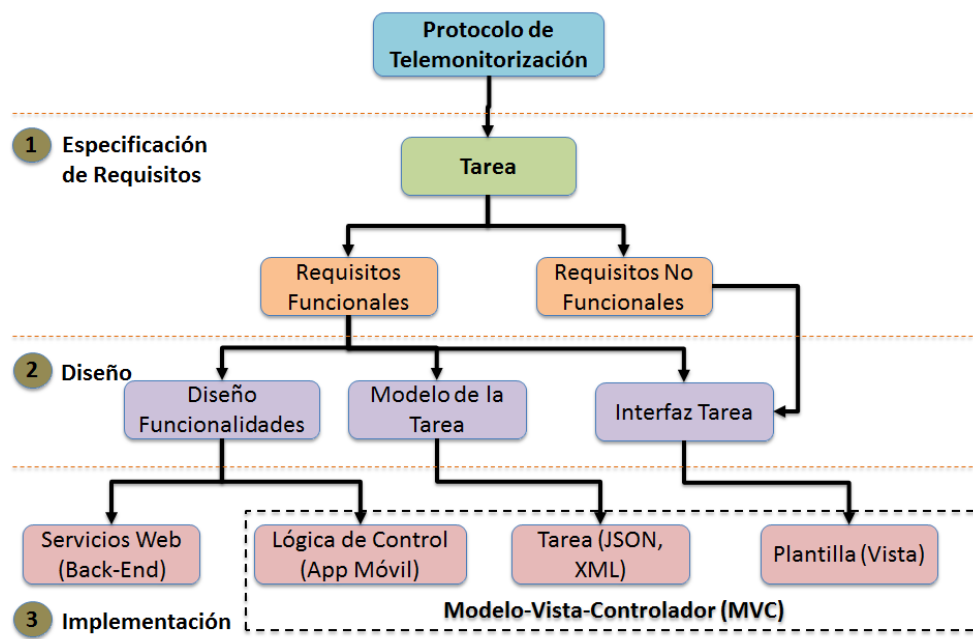


Figura 54- Adición de una nueva tarea de telemonitorización siguiendo el modelo para la gestión de tareas

Gracias al patrón MVC junto con el uso de una arquitectura híbrida (SOA-ROA) se garantiza la posibilidad de añadir cualquier tipo de funcionalidad o tarea sin importar el tipo de tarea o la complejidad de la misma (dando soporte a cualquier tipo de usuario) y sin afectar al rendimiento o uso del sistema. Para ellos basta simplemente con la adición de nuevos servicios a los ya existentes.

6.4. Modelo para escenarios colaborativos en plataformas de telemonitorización

6.4.1. Introducción

En las plataformas de telemonitorización hay dos roles claramente diferenciados: usuarios monitorizados y usuarios que monitorizan, también llamados supervisores. No obstante, dentro de cada rol pueden encontrarse distintas especializaciones de estos ya que es habitual que en un escenario de telemonitorización intervengan distintas disciplinas/áreas.

Esta caracterización de tipos de usuarios se suele llevar a cabo dependiendo de las funcionalidades, tareas o responsabilidades, ya que normalmente no todas las tareas de una plataforma de telemonitorización a las que da soporte las lleva a cabo un único tipo de usuario.

Normalmente, en una plataforma de telemonitorización donde se supervisan una serie de usuarios monitorizados, se pretende determinar su evolución, grado de ejecución de las tareas y consecución de objetivos del usuario monitorizado.

Para evaluar el progreso de los usuarios monitorizados, normalmente se requiere del conocimiento experto de varios supervisores de diferentes áreas o ámbitos, ya que a menos que una solución de telemonitorización se centre en un grupo poblacional concreto, determinar si evoluciona favorablemente implicará varias áreas y por lo tanto la coordinación de los diferentes supervisores.

La necesidad de plantear una solución que posibilite la colaboración entre los distintos roles y los distintos tipos de usuarios permite habilitar plataformas de telemonitorización multidisciplinares, que ofrezcan un soporte integral a la telemonitorización.

6.4.2. Objetivos

Tradicionalmente, la colaboración que se realiza en un escenario de telemonitorización se suele llevar a cabo a través de reuniones diarias, semanales o mensuales donde los supervisores/expertos intercambian información, puntos de vista u opiniones sobre el usuario monitorizado.

Este tipo de escenario, donde se requiere una colaboración multidisciplinar, tiene claras desventajas: reuniones presenciales, desplazamientos, incremento de coste, tiempo, etc, además, de las desventajas del intercambio de información sin soporte TICs (documentos físicos, organización de estos documentos, problemas a la hora de buscar un dato o documento exacto, etc.).

Con la aparición de las TICs es posible proponer modelos colaborativos que automaticen la gestión de información, como, por ejemplo, soluciones basadas en el intercambio de emails o conversaciones en tiempo real (chat), tanto a nivel general [Kraemer 1988] como soluciones relacionadas con la telemonitorización [Pereira 2014] [Yang 2014] [Ruiz-Zafra 2013].

Normalmente, una plataforma de telemonitorización suele tener unos objetivos claros (mejorar el sistema cognitivo de un usuario o controlar su estado de salud), que, por lo general, se pretenden alcanzar en un grupo poblacional concreto (personas mayores,

pacientes de rehabilitación, adultos con problemas cardíacos), ya que diferentes grupos poblacionales tienen objetivos distintos y por lo tanto se suelen generar soluciones independientes.

Por esto, la mayoría de las plataformas de telemonitorización están enfocadas en un grupo poblacional concreto y, por lo general, con un único tipo de supervisor que se encarga de asegurar de que los objetivos se van alcanzando. Por lo tanto, la colaboración multidisciplinar que se espera en una de estas plataformas de telemonitorización no siempre es necesaria y por lo tanto no siempre está soportada.

Gracias a las tecnologías en las que están basadas las plataformas de telemonitorización actuales, muchas de ellas mencionadas en el capítulo 3, como la computación en la nube, smartphones, arquitecturas software, etc. las características de estas tecnologías (deslocalización, acceso a datos en tiempo real) permiten una colaboración en tiempo real, desde cualquier sitio y en cualquier momento. Es decir, intercambiar cualquier tipo de información en tiempo real si fuese necesario sin necesidad de compartir el mismo espacio físico

Esta necesidad de dar soporte a la colaboración en determinados escenarios donde se usan plataformas de colaboración se plantea como un reto, al ser un requisito la interacción entre los diferentes tipos usuarios, independientemente del objetivo que tengan.

El objetivo que se plantea, como propuesta dentro del framework y que se desarrolla en este capítulo, es proponer un modelo para escenarios colaborativos multidisciplinarios en plataformas de telemonitorización junto con la tecnología adecuada para implementar dicho modelo.

Para abordar este objetivo, se plantean dos propuestas, una a nivel conceptual y de diseño (modelo), y otra a nivel tecnológico (soporte al modelo), que pretenden dar forma a la solución siguiendo las tecnologías mencionadas en el capítulo 3:

- **Modelo colaborativo.** En un escenario colaborativo es necesario identificar los diferentes actores que van a interactuar y cómo lo van a hacer, esto es, que tipos de interacciones son necesarias para dar soporte a las diferentes funcionalidades relacionadas con la colaboración. Con el objetivo de soportar escenarios colaborativos, independientemente del tipo de usuarios, roles y tipos de interacciones, se propondrá un modelo.
- **Arquitectura para la gestión de notificaciones.** Además del modelo colaborativo que permita el diseño de las diferentes interacciones a través de notificaciones (evento asíncrono que se produce de un emisor a un receptor y que suele transmitir información de algún tipo (mensaje, alerta, documento)), es necesario una arquitectura hardware y software robusta, estable, escalable y extensible que implemente dichas notificaciones y permita implementar otras nuevas para dar soporte a nuevas colaboraciones o funcionalidades.

6.4.3. Escenarios colaborativos

Dentro de un escenario de telemonitorización, donde hay presentes varias disciplinas, es necesaria la colaboración entre los diferentes supervisores/expertos de cada área para asegurar la correcta telemonitorización del usuario monitorizado.

En estos escenarios los distintos supervisores deben planificar sus respectivas sesiones de telemonitorización (conjunto de tareas) para el usuario monitorizado. Algunas de estas tareas pueden entrar en conflicto con otras disciplinas, por lo que el responsable de dicha disciplina debe estar de acuerdo o no en la realización de ciertas tareas y quizás sea necesario ajustar o replantear las tareas de su disciplina para mantener la consecución de objetivos.

Por ejemplo, en una plataforma de telemonitorización para adultos con problemas cardíacos, pueden existir supervisores que desempeñarán el rol de doctor, cardiólogo y nutricionistas. Ambos tipos de supervisores definirán diferentes tareas dependiendo de su objetivo. Por ejemplo, el cardiólogo propondrá medir el pulso cardíaco dos veces al día y la presión arterial o tomar un medicamento concreto, y el nutricionista seguir una dieta acorde a las necesidades del usuario monitorizado.

Puesto que la dieta es un componente fundamental en la salud de una persona, cualquier cambio por parte del nutricionista sobre la dieta del usuario monitorizado debe ser notificado al cardiólogo, para que se tenga en cuenta a la hora de evaluar los datos registrados de pulso cardíaco y la presión arterial del usuario monitorizado.

Sin la coordinación y colaboración entre ambos supervisores, la supervisión individual sobre un usuario monitorizado puede ser incluso perjudicial, afectando negativamente su evolución dentro de la telemonitorización o su salud.

Sobre la base de estas consideraciones previas se plantea un esquema de interacción a aplicar en entornos de telemonitorización para dar soporte a escenarios colaborativos y mostrado en la Figura 55. Esta figura representa las interacciones o situaciones (etiquetadas en la figura de 1 a 5) entre los distintos tipos de usuarios en un escenario general de telemonitorización con soporte colaborativo:

1. Cada supervisor de un área o disciplina tiene una reunión personal con el usuario a monitorizar, para detectar necesidades, requisitos o simplemente obtener un *feedback* que sea útil a la hora de definir su telemonitorización en esa disciplina.
2. Una vez que cada supervisor ha mantenido su reunión personal con el usuario a monitorizar, todos ellos tienen una reunión (presencial u online) para intercambiar impresiones, opiniones, información relevante, planeamiento del ámbito de telemonitorización, etc. En esta fase los supervisores deben ser conscientes de cómo se va a planificar la telemonitorización por parte de otros supervisores en otras disciplinas, para así ajustar su plan de telemonitorización, esto es, cada uno debe ser consciente del tipo de tareas y la frecuencia con la que serán realizadas por el usuario monitorizado en cada área o disciplina.
3. Cada supervisor, a partir de la información obtenida del resto de supervisores, define las distintas tareas del usuario monitorizado en su área/disciplina.
4. El usuario monitorizado realiza las distintas tareas de los distintos ámbitos.

- Los supervisores interactúan entre ellos y con el usuario monitorizado, quien mientras realiza las tareas puede interactuar con el supervisor, o redefinir nuevas tareas que el usuario debe llevar a cabo con respecto a cambios que puedan ocurrir.

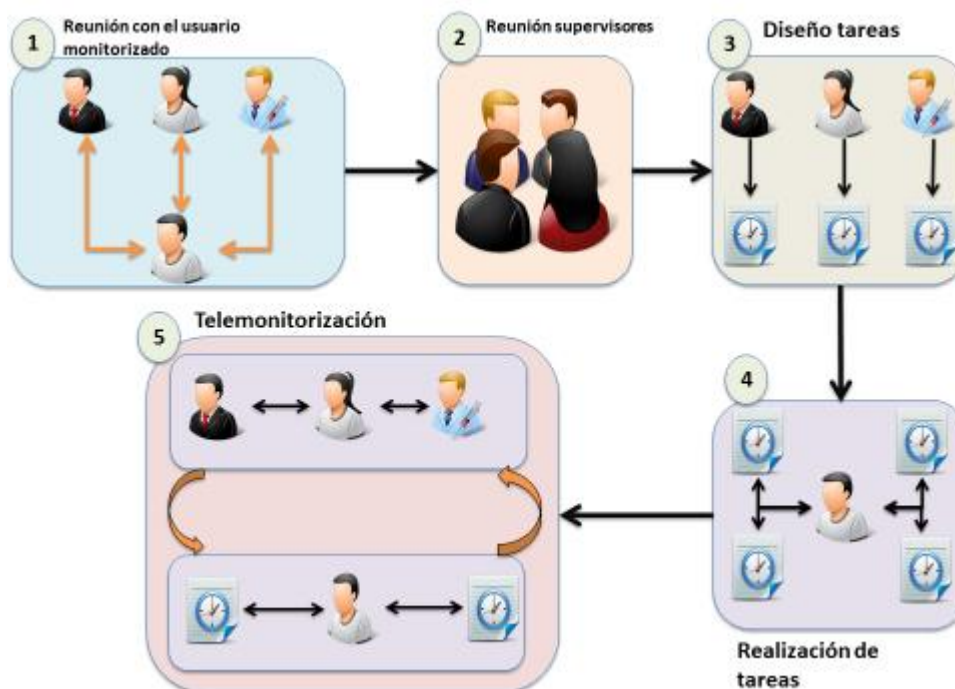


Figura 55 - Escenario colaborativo para entornos de telemonitorización

Un escenario colaborativo como el representado en la figura 55 debe tener presente una serie de características que permita que esa colaboración sea posible. Estas características son:

- **Soporte a la comunicación en tiempo real.** En otros sistemas el tiempo de respuesta no es un parámetro importante (correo electrónico, tiempo de descarga), pero en un contexto de telemonitorización con soporte colaborativo, permitir la interacción en tiempo real entre ciertos tipos de usuarios (profesionales principalmente) es necesario en determinadas situaciones para garantizar la efectividad de la telemonitorización, y que las decisiones tomadas se puedan considerar con puntualidad.
- **Diferentes tipos de interacciones.** En este escenario propuesto se presentan tres tipos de interacciones: de supervisor a supervisor, de supervisor a usuario monitorizado y de usuario monitorizado a supervisor.
- **Diferentes tipos de notificaciones.** Una notificación es un evento asíncrono que se produce de un emisor a un receptor y que suele transmitir información de algún tipo (mensaje, alerta, documento). Debido a que hay distintos tipos de interacciones, es necesario soportar distintos tipos de notificaciones para dar soporte a todas las colaboraciones necesarias.

Partiendo del escenario presentado y de estas tres características, se presenta como propuesta para permitir la colaboración en plataformas de telemonitorización: (1) un modelo colaborativo que se sustenta en la identificación de los diferentes roles y diferentes tipos de

interacciones, y (2) la arquitectura necesaria para dar soporte a ese modelo satisfaciendo estas características.

Esta propuesta, considerando el escenario presentado en la figura 54, está concebida para dar soporte a la interacción en las etapas 3, 4 y 5 de dicha figura, ya que son estas donde hay un uso de la tecnología y por lo tanto donde el modelo y arquitectura son aplicables.

6.4.4. Modelo colaborativo

Un entorno de trabajo colaborativo suele involucrar varias tareas que van desde el intercambio de documentos hasta la gestión de información generada en una reunión, o la coordinación de las reuniones semanales o mensuales. Algunos enfoques tradicionales han decidido aplicar las TICs para mejorar estas tareas, permitido a través de herramientas de trabajo colaborativas la realización de reuniones no presenciales, el intercambio de información o la gestión edición de documentos colaborativos (*Google Docs, Hangout, Basecamp*, etc.).

Las nuevas tecnologías permiten celebrar reuniones o intercambiar información de una manera mucho más directa y fácil a través de dispositivos personales, como smartphones. Estas tecnologías permiten dar soporte a no de los principales requisitos en un escenario de telemonitorización: la interacción entre supervisores y usuarios monitorizados, gracias a las características propias de la tecnología: deslocalización, portabilidad, eficiencia, etc.

En un escenario de telemonitorización, un cambio en el comportamiento del usuario monitorizado o una expectativa con respecto a su evolución por debajo de la esperada puede afectar a su telemonitorización, en términos de redefinir sus objetivos y por tanto las tareas de telemonitorización. Esto puede ocasionar una adaptación del resto de tareas en el resto de disciplinas.

Por ejemplo, en una plataforma de telemonitorización en el ámbito del deporte, si un supervisor (entrenador) modifica la intensidad de ciertos ejercicios en el entrenamiento de un usuario monitorizado (atleta), el nutricionista (otro tipo diferente de supervisor con otras responsabilidades) debe ser consciente de dicho cambio para ajustar la dieta y que, así el atleta, pueda completar correctamente los ejercicios, para mantener el rendimiento y cumplir con sus objetivos.

Con el objetivo de dar soporte a la colaboración en una plataforma de telemonitorización, se propone una solución (modelo) basada en la posibilidad de los diferentes roles de supervisores, y por ende de responsabilidades/funcionalidad, y en el concepto de notificación: evento asíncrono que se produce de un emisor a un receptor y que suele transmitir información de algún tipo (mensaje, alerta, documento).

La figura 56 muestra el modelo con diferentes conceptos para representar un entorno colaborativo en un escenario de telemonitorización: tipos de usuarios y tipos de notificaciones. Este modelo tiene como objetivo representar diferentes tipos de notificación en un escenario de telemonitorización con soporte colaborativo entre supervisores.

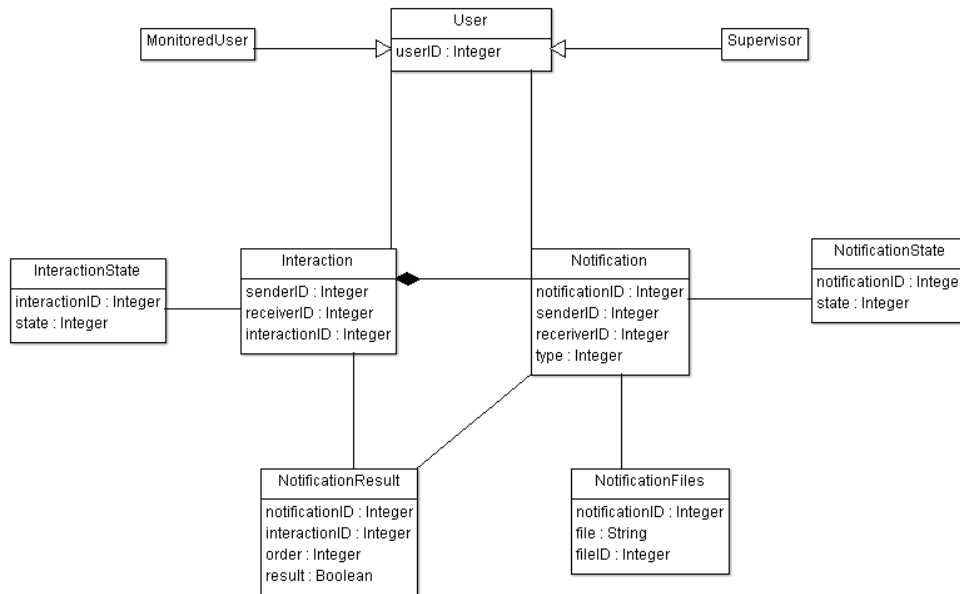


Figura 56 - Modelo colaborativo para plataformas de telemonitorización

Con respecto a los tipos de notificaciones, se han identificado distintos tipos para dar soporte a situaciones que por lo general están presentes en plataformas de telemonitorización, que requieren de soporte colaborativo y que se podrían implementar siguiendo el modelo propuesto. Estos distintos tipos de notificaciones están recogidos en la Tabla 3.

Tipo de not.	Usuarios implicados	Descripción
Tipo 1	Supervisor -> Supervisor	Un supervisor comunica al resto de supervisores una decisión tomada en su ámbito sobre las actividades o evolución del usuario monitorizado.
Tipo 2	Supervisor -> Supervisor (aprobación con escrutinio)	Un supervisor comunica al resto de supervisores una decisión tomada en su ámbito que debe ser aceptada por el resto de supervisores antes de enviarse al usuario monitorizado, ya que otros ámbitos están involucrados y por lo tanto estos supervisores deben ser notificados.
Tipo 3	Supervisor -> Usuario monitorizado (con notificación de cambios)	Un supervisor cambia una tarea de un usuario monitorizado y este es notificado.
Tipo 4	Supervisor -> Usuario monitorizado	Un supervisor cambia una tarea de un usuario monitorizado y este no es notificado.
Tipo 5	Usuario monitorizado -> Supervisor	El usuario monitorizado notifica al supervisor o supervisores sobre su evolución o una tarea realizada.
Tipo 6	Usuario monitorizado -> Supervisor (meta alcanzada)	El usuario monitorizado notifica al supervisor o supervisores sobre una meta alcanzada.

Tabla 3 - Tipos de notificaciones principales del modelo

Esta tabla muestra seis tipos de notificaciones frecuentes en la telemonitorización donde intervienen expertos de diferentes disciplinas. No obstante, el modelo presentado en la Figura 56 permite definir nuevos tipos de notificaciones a estas seis, con el objetivo de cubrir otra funcionalidad o necesidad.

La figura 57 muestra un ejemplo representativo, mediante un diagrama de secuencia, acerca de cómo se produciría los diferentes eventos ocasionados por el envío de una notificación para representar una de las notificaciones propuestas en la tabla 3, en este caso, por ejemplo, una de tipo 2 (entre supervisores con escrutinio).

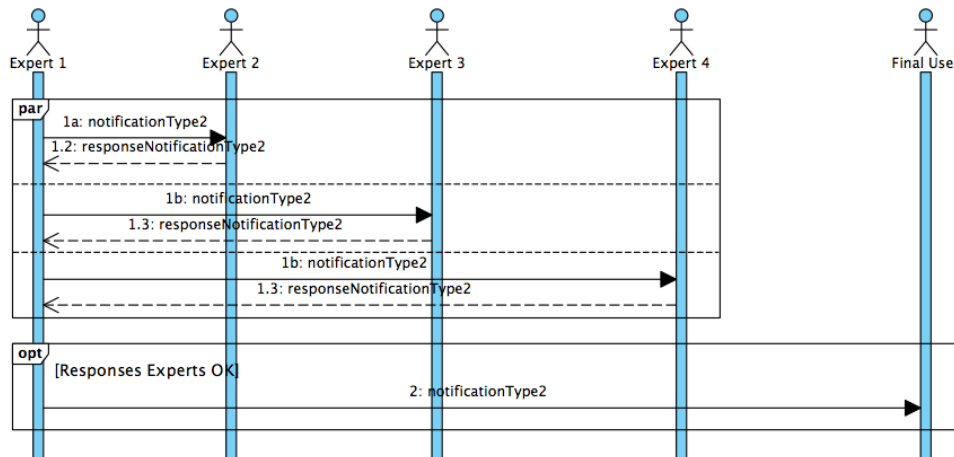


Figura 57 - Ejemplo de interacción entre supervisores y usuario monitorizado

Un supervisor quiere enviar una notificación a un usuario monitorizado, pero requiere de la aprobación del resto de supervisores (Tipo 2). El supervisor 1 envía al resto de supervisores la notificación (mediante una notificación emergente, por ejemplo) y espera la respuesta de estos. Si todos ellos aceptan y aprueban el envío de la notificación, respondiendo “OK”, la notificación es finalmente enviada al usuario monitorizado.

Esto implica que la negociación o deliberación de las tareas a realizar dentro del protocolo que se realicen durante la monitorización de un usuario son transparentes para este, al ser un proceso donde solo participan supervisores. El usuario monitorizado únicamente recibe la notificación final.

6.4.5. Arquitectura para la gestión de notificaciones

La arquitectura de una plataforma de telemonitorización debe contemplar en su diseño, además del resto de elementos software para dar soporte a la telemonitorización, componentes dedicados para la gestión de notificaciones.

La propuesta de arquitectura híbrida (SOA-ROA) presentada en el capítulo 3, extendida en este mismo capítulo (6 sección 1), y las tecnologías ya presentadas (tecnología móvil, computación en la nube, etc.), son elementos adecuados para el soporte de un sistema de gestión de notificaciones, ya que soportan muchas características y ventajas necesarias, entre las que destacan:

- **Tiempo real.** La tecnología *push/pull* es cada vez un recurso más utilizado, sobre todo en escenarios con dispositivos móviles [Burgsahler 2014]. Muchas de las tecnologías actuales (librerías software, servicios web) dan soporte a desarrolladores para desplegar soluciones basadas en esta tecnología. Su utilización, unida a la robustez de la computación nube y la posibilidad de la deslocalización proporcionada por los dispositivos móviles permite, entre otras cosas, enviar o recibir notificaciones en tiempo real desde cualquier lugar, en cualquier momento y desde cualquier tipo de dispositivo.

- **Escalabilidad.** En una plataforma de telemonitorización se desconoce el número de usuarios que van a hacer uso del sistema. De igual manera, para un contexto colaborativo no se sabe el número de usuarios que van a colaborar entre sí ni el número de interacciones diarias. Por esto, el uso de tecnología *cloud*, al igual que posibilita la escalabilidad para plataformas de telemonitorización, permite la misma escalabilidad para entornos colaborativos, y especialmente en este caso que ambos escenarios forman parte de la misma solución.
- **Decisiones “sobre la marcha”.** En una plataforma de telemonitorización que se desarrolle siguiendo el framework propuesto será muy habitual que, tanto usuarios supervisores como usuarios monitorizados, usen aplicaciones móviles como parte del soporte a la telemonitorización. El uso de smartphones de manera cotidiana (acceso inmediato y disponible para su uso) simplifica poner en contacto a un usuario en concreto, tomando una decisión en un momento determinado que pueda ser notificada en tiempo real al usuario monitorizado y que este, en ese mismo momento, pueda tomar una decisión o ser informado.
- **Extensibilidad.** El modelo de arquitectura híbrida (SOA-ROA) presentado para plataformas de telemonitorización permite dar soporte a las notificaciones y soportar nuevos tipos de notificaciones en un futuro, permitiendo así, cubrir nuevas funcionalidades, implementar nuevos servicios que codifiquen dichas funcionalidades.
- **Información centralizada.** Aunque la tecnología *cloud* permite un acceso deslocalizado, desde cualquier sitio y en cualquier momento, conceptualmente y a nivel práctico se entiende como un conjunto de información centralizada. Esto hace que desde distintos dispositivos se pueda acceder y gestionar la misma información, por lo que usuarios y supervisores pueden tomar decisiones desde varios dispositivos considerando información actualizada.

La Figura 58 representa los distintos módulos software necesarios que hay en un sistema de gestión de notificaciones basado en una arquitectura híbrida (SOA-ROA).

En un escenario de telemonitorización con soporte colaborativo, a nivel de software debe haber distintos módulos (librerías, herramientas) para posibilitar estos entornos colaborativos: espacio de trabajo común para dar soporte a reuniones tradicionales, almacenamiento persistente para almacenar todos los cambios y notificaciones, etc.

Dentro de los diferentes módulos software necesarios para dar soporte a entornos colaborativos, se presuponen al menos dos para garantizar la colaboración: conjunto de servicios para la gestión de notificaciones y un coordinador de notificaciones.

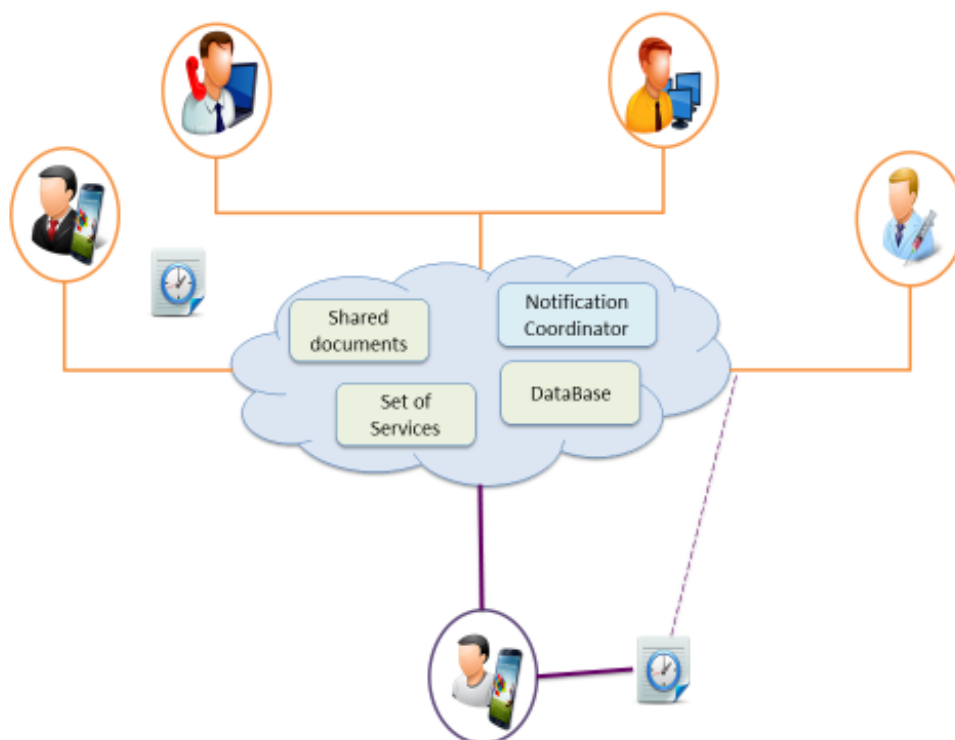


Figura 58 - Distintos elementos del sistema de gestión de notificaciones

Servicios

Aunque el modelo propuesto junto la arquitectura permite dar soporte a diferentes tipos de notificaciones, es necesario implementar las funcionalidades que permitan codificar esas notificaciones (enviar notificación de un emisor a un receptor, eliminar notificación, responder a notificación pendiente, etc.).

Por la propuesta de arquitectura híbrida (SOA-ROA) estas funcionalidades se implementarán en servicios, que serán usados por las distintas aplicaciones de la plataforma y que den soporte al envío de notificaciones entre los diferentes tipos de usuarios.

Un acceso fácil al consumo de servicios y la posibilidad de añadir nuevos servicios para implementar nuevos tipos de notificaciones permite que un entorno colaborativo sea fácilmente extensible y gestionable, mediante el uso de servicios a través de su API.

No obstante, además de estos servicios es necesario el software adecuado para gestionar internamente dichas notificaciones, esto es, su estado: si ha llegado correctamente, si está pendiente de recepción o respuesta, etc.

Coordinador

Aunque el acceso a la gestión de notificaciones por parte de supervisores y usuarios monitorizados se realiza por el acceso y consumo a los diferentes servicios a través de las aplicaciones, internamente, y correspondiente al *back-end* del sistema, se hace necesario gestionar todas estas notificaciones correctamente, dependiendo del tipo que sean.

El modelo presentado en la figura 56 considera notificaciones, interacciones (grupo de notificaciones) y el estado y respuesta de notificaciones y consideraciones, esto es, una

representación de los diferentes estados entre notificaciones con el objetivo de dar soporte correctamente a las diferentes interacciones en un escenario colaborativo.

Dentro de la solución propuesta (modelo y arquitectura) se hace necesaria la figura del **coordinador de notificaciones**, un componente software usado por los distintos servicios que implementan funcionalidades de colaboración que pretende implementar el modelo de la figura 56 gestionando el estado de las diferentes notificaciones/interacciones.

Cuando un supervisor envía una notificación a un usuario monitorizado y esta notificación debe ser aprobada por el resto de supervisores, es necesario almacenar tanto el envío de petición desde el supervisor al cómo la respuesta de todos ellos.

El coordinador de notificaciones es el encargado de: enviar la notificación (*push/pull*), almacenar ese envío en el soporte de datos, almacenar en el mismo los distintos estados del procesamiento dependiendo del tipo de notificación que se procese, determinar cuándo una notificación está pendiente de aprobación o no, detectar cuando ha sido aprobada para enviarse al usuario monitorizado, el orden en el que se envían las notificaciones etc.

En esencia, el coordinador es el software que garantiza el buen funcionamiento de un sistema colaborativo, al encargarse tanto del envío correcto de notificaciones como el estado de estas.

Capítulo 7

**Soporte a la
implementación:
Zappa y WearIt**

7.1. Introducción

En este capítulo se presentan las herramientas del framework que van a dar soporte en la etapa de implementación. Estas son:

- **Zappa:** Zappa es una plataforma basada en componentes software que tiene como objetivo facilitar el desarrollo de plataformas de telemonitorización, al dar soporte a funcionalidades comunes en las plataformas (gestión de datos, interacción con el soporte de datos externo), proporcionando propiedades como reusabilidad y extensibilidad, permitiendo que los sistemas basados en esta plataforma den soporte a nuevas funcionalidades añadiendo nuevos componentes.
- **WearIt:** WearIt es una herramienta que facilita la integración de un *wearable* en un sistema sin necesidad de conocimientos técnicos de bajo nivel sobre ningún aspecto del mismo, simplificando el tiempo de desarrollo [Ruiz-Zafra 2014c] [Ruiz-Zafra 2015c].

7.2. Zappa

7.2.1. Introducción

Uno de los principales objetivos del framework propuesto es asegurar que las plataformas de telemonitorización desarrolladas bajo dicho framework sean adaptables a lo largo del tiempo para dar soporte a nuevas funcionalidades, roles, tareas etc.

Como solución para el *back-end* se ha propuesto una arquitectura híbrida (ROA-SOA), donde la integración de nuevas funcionalidades se basa en la adición de nuevos servicios web, ya que, por el modelo de arquitectura y la tecnología, esta adición no afecta al rendimiento de la plataforma de telemonitorización.

Las plataformas de telemonitorización, además de todos estos servicios (*back-end*), están formadas por las diferentes aplicaciones usadas por los actores de la plataforma, especialmente aplicaciones móviles (smartphones).

Usualmente, para la creación de estas aplicaciones se suele implementar un nuevo código que permita dotar a la aplicación móvil de funcionalidades como la gestión de datos, procesamiento de información desde la nube, etc.

Estas funcionalidades son en cierta medida reusables de un proyecto a otro. No obstante, esta reusabilidad se basa en hacer una copia exacta del código e intentar adaptarlo a otra solución, con los cambios de código que sean necesarios. Dependiendo de la calidad, claridad y documentación del código original, este proceso puede ser inmediato (copiar, pegar y usar) o demorarse por requerir modificaciones en el mismo para poder adaptarlo.

Esta falta de estandarización sobre cómo definir el código para permitir que sea totalmente reutilizable (con respecto a funcionalidad y tiempo de integración) hace que en muchas ocasiones la integración de código fuente de un proyecto anterior en un nuevo proyecto no sea automática, generando problemas de funcionalidad, lo que implica un incremento de tiempo y esfuerzo en el desarrollo de la solución.

Una solución habitual es diseñar componentes que puedan reutilizarse [Szyperski 1995]: unidades de software/código encapsuladas/empaquetadas que ofrecen un conjunto de

funcionalidades a través de una interfaz bien definida (concepto ya abordado en el capítulo 3 – Desarrollo orientado a componentes).

El uso de este enfoque permite el mantenimiento a largo plazo de las aplicaciones móviles, con el objetivo de poder ir añadiendo nuevas funcionalidades para soportar cambios en el protocolo. Este enfoque, además de mejorar la reusabilidad, ya que los componentes son módulos software independientes, también permite ser independientemente extensible, esto es, si puede ser dinámicamente extendido y donde se pueden combinar extensiones independientes sin conocimiento unas de otras [Szyperski 1995].

Añadir una nueva funcionalidad a un componente ya desplegado, consiste en implementar esa funcionalidad y publicar la API/interfaz, para dar conocer cómo usarla, qué parámetros necesita y qué resultado devuelve. Asimismo, como un componente se concibe como una caja negra, cambios en un protocolo solo implicaría sustituir un componente por la nueva versión, con las nuevas interfaces de acceso para dichas funcionalidades.

Aunque el uso de este enfoque de programación no es obligatorio, reduce considerablemente el esfuerzo de los programadores, permitiendo a través de las interfaces de los componentes crear nuevas aplicaciones que, por lo general, comparten una serie de funcionalidades, ahorrando tiempo y ganando eficiencia (crear más aplicaciones, más rápido y con niveles estándares de calidad).

Con este objetivo se ha diseñado y desarrollado (dentro del marco de la metodología y herramientas de soporte para la creación de soluciones de telemonitorización) una plataforma para la creación de plataformas de telemonitorización llamada Zappa [Ruiz-Zafra 2013].

Zappa es una plataforma orientada a componentes formada por diversos componentes software, desarrollados para el sistema operativo Android. Estos componentes proporcionan una serie de funcionalidades que suelen estar presentes en plataformas de telemonitorización, como la gestión de datos internos o la comunicación con el soporte externo de datos, agilizando y simplificando así el desarrollo.

7.2.2. Arquitectura

Zappa presenta una serie de componentes orientados a proporcionar funcionalidades habituales en plataformas de telemonitorización (cabe destacar que dichos componentes pueden ser reusados en otro tipo de sistemas no relacionados con la telemonitorización).

Estas funcionalidades se pueden agrupar en las siguientes categorías:

- **Gestión de datos internos.** En las plataformas de telemonitorización, aunque exista una dependencia entre aplicaciones móviles y los servicios externos alojados en la nube, en muchos casos es necesario el almacenamiento interno de datos, ya sea por consistencia para futuras visualizaciones, como para realizar con ellos un proceso previo de procesamiento antes de subirlos a la nube.
- **Interacción con la nube (servicios externos).** La constante interacción e intercambio de información entre usuarios finales y supervisores es crucial, y en el modelo propuesto se posibilita gracias a los servicios soportados por la infraestructura de la nube. Con objetivo de facilitar al desarrollador esta interacción, se proponen componentes para

gestionar todas las interacciones con la nube o bien facilitar el consumo de recursos vía HTTP.

- **Representación de datos (Diseminación).** Durante las sesiones de telemonitorización, donde el usuario final realiza las tareas definidas por los profesionales/supervisores, se genera por lo general de sus propias acciones y de la información recopilada por sensores externos una gran cantidad de información. Debido a que el usuario final no suele estar familiarizado con lectura e interpretación de grandes cantidades de datos, se hace necesario mostrarlos de una manera más intuitiva para que sea capaz de interpretarlos y poder saber así cómo es su evolución o conocer su estado. Es por esto, que dentro de la plataforma se ofrecen componentes para facilitar esta visualización.
- **Gestión de sensores/hardware internos.** Puesto que un recurso muy usado en las plataformas de telemonitorización es la utilización de los diferentes dispositivos internos del dispositivo móvil, Zappa permite la gestión de los sensores internos del dispositivo móvil: GPS, cámara, el acelerómetro

Existen otros componentes de soporte que proporcionan funcionalidades de bajo nivel y que permiten posibilitar funcionalidades de otros componentes (p.e. uso del protocolo HTTP); así como otros con funcionalidades específicas que se han ido añadiendo debido a demandas de procesos de desarrollo particulares, como pueden ser para gestionar la cámara o la gestión de material audiovisual (vídeo).

La adopción de un enfoque orientado a componentes resulta muy adecuada para determinados tipos de soluciones, como pueden ser las plataformas de telemonitorización, ya que tiene unas claras ventajas:

- **Portabilidad-reusabilidad.** Dentro de la plataforma sobre la que se programan los componentes es muy fácil portar éstos para su utilización en otros proyectos. Gracias a una interfaz bien definida y un desarrollo robusto, la integración, interacción y uso se realiza en un tiempo mínimo
- **Extensibilidad.** Debido a la independencia de los propios componentes, la adición de nuevas funcionalidades a un componente o bien de nuevos componentes no afecta a la plataforma, pudiendo añadir nuevas funcionalidades independientes y reutilizables.
- **Composición.** Gracias a una interfaz bien definida de los componentes, es posible crear nuevas funcionalidades específicas a través de estas interfaces que permitan generar nuevos componentes con sus propias interfaces.
- **Abstracción.** El uso de un componente por parte del resto de componentes o un programador es entendido como una caja negra, accediendo a las funcionalidades a través de una API, pero sin conocimiento de cómo están implementadas esas funcionalidades. El cambio de un componente por otro, siempre y cuando no cambie su interfaz, no afectaría al cambio del sistema.

Disponer de una solución que permita ahorrar tiempo en la creación de software y que, además, permita reusar dicha solución fácilmente, y extenderla con nuevas funcionalidades permitiría dar soporte a nuevos requisitos.

Esta ventaja posibilita, junto con una arquitectura orientada a servicios en la parte del *back-end* (como la arquitectura híbrida propuesta - capítulo 6, sección 1-), que cualquier funcionalidad pueda implementarse en un sistema ya desplegado, como pueden ser desde nuevas funcionalidades para cubrir nuevas tareas de telemonitorización o bien nuevas tareas necesarias por la adición de nuevos roles.

Como se comentó en el capítulo 3, a nivel conceptual un componente software es: “*A unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties.*” [Szyperski 1995]. Además, a un componente software se le atribuye tres características principales, de acuerdo a [Szyperski 1995]:

- is a unit of independent deployment;
- is a unit of third-party composition;
- has no (externally) observable state;

Este concepto ha variado a lo largo de los años dependiendo de las adaptaciones del mismo a la tecnología y sobre todo a los estándares que han ido apareciendo (CORBA, DCOM, etc.), donde cada uno representa una implementación concreta de este concepto.

Muchas de estas tecnologías pretendían emular arquitecturas orientadas a servicios, permitiendo que dichos componentes se desplegaran en los dispositivos con el objetivo de proporcionar servicios no sólo a otros componentes internos, sino a otro software de dispositivos externos (a través de un protocolo de comunicación como Wifi o Bluetooth).

Aunque esta opción es viable y fue (es) muy usada durante mucho tiempo, para el diseño y desarrollo de Zappa no se ha continuado con este diseño por dos razones (o desventajas) fundamentales:

- Desplegar servicios en dispositivos móviles, para que estos sean accesibles por otros dispositivos tiene un coste computacional añadido, ya que cualquier procesamiento que requieran dichos servicios deberá realizarse en el propio dispositivo móvil, incrementando la carga de trabajo de este y por lo tanto comprometiendo su rendimiento, batería, etc.
- Desplegar servicios en un dispositivo móvil tiene el requisito añadido de mantener el dispositivo siempre encendido para que el servicio esté disponible. La necesidad de consumir un servicio, pero no poder hacerlo porque el dispositivo móvil está apagado, sin conexión a internet o fuera de rango (Bluetooth), restringe la disponibilidad y puede influir en el flujo de funcionamiento del sistema.

Por estas desventajas, en el diseño y desarrollo de la plataforma Zappa se decidió que toda funcionalidad común y que pueda ser usada por diferentes usuarios/dispositivos/aplicaciones, necesitando por lo tanto ser accesible y devolviendo un resultado interoperable, se implemente en la nube. Es decir, delegar la carga computacional de carácter generalista a la nube (*cyber foraging*) [Satyanarayanan 2001].

De esta manera, las funcionalidades generales pueden ser consumidas por cualquier dispositivo más fácilmente que si tuvieran que acceder al propio dispositivo móvil para usar dicho servicio a través de otros protocolos o tecnologías específicas, además de los

inconvenientes presentes de la no disponibilidad del dispositivo o problemas de rendimiento para ofrecer determinados servicios.

De acuerdo a esto, y siguiendo los principios de componentes software presentados en [Szyperki 1995], se ha optado por la creación de componentes propios en lugar de usar tecnología estándar, donde estos serán unidades de software independientes, con una API bien definida y que se despliegan sobre una plataforma concreta.

Estos componentes están diseñados para gestionar funcionalidades internas del dispositivo móvil (Android en este caso) y no para ofrecer servicios al exterior, por lo que no tienen un canal de comunicación con otro dispositivo o aplicación.

Las funcionalidades que proporciona cada componente son código organizado en clases, funciones independientes, librerías, etc. Todos estos elementos son privados, no pudiendo ser accedidos desde fuera del componente, por lo que sólo los propios elementos del componente podrán usar las funcionalidades del resto de elementos.

El acceso a las funcionalidades se lleva a cabo a través de una API que consiste en métodos de clase, por lo que su uso pasa por invocar al método con los parámetros oportunos. Esta API está en una clase cuya funcionalidad está implementada mediante elementos con el atributo *privado* para garantizar la opacidad en cuanto al código y simplificar así el uso por parte del desarrollador/programador. Esta implementación de componentes puede verse en la figura 59.

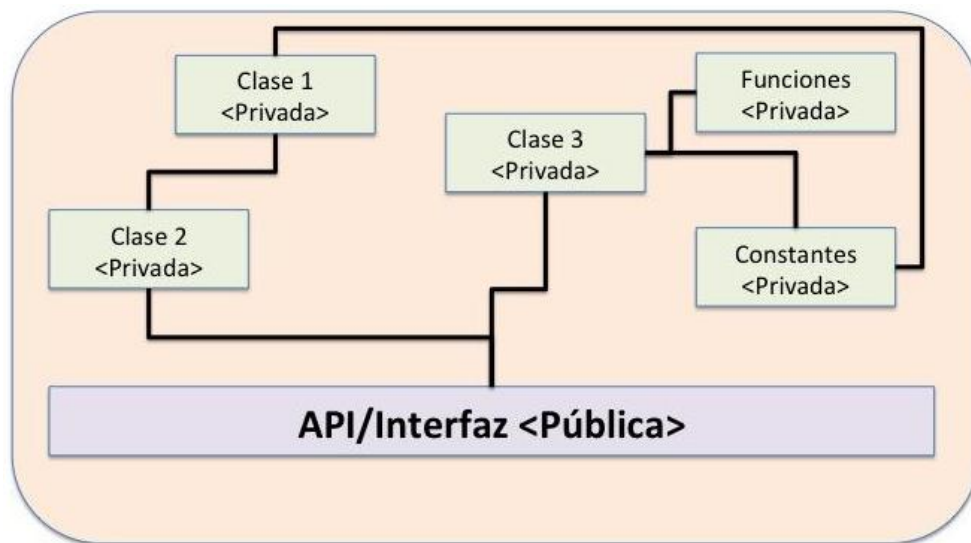


Figura 59 - Diseño de componentes en Zappa

En la figura 60 pueden verse los distintos componentes, agrupados por paquetes, que ofrece actualmente la plataforma. A continuación, se detalla cada uno de estos componentes, agrupados por paquetes.

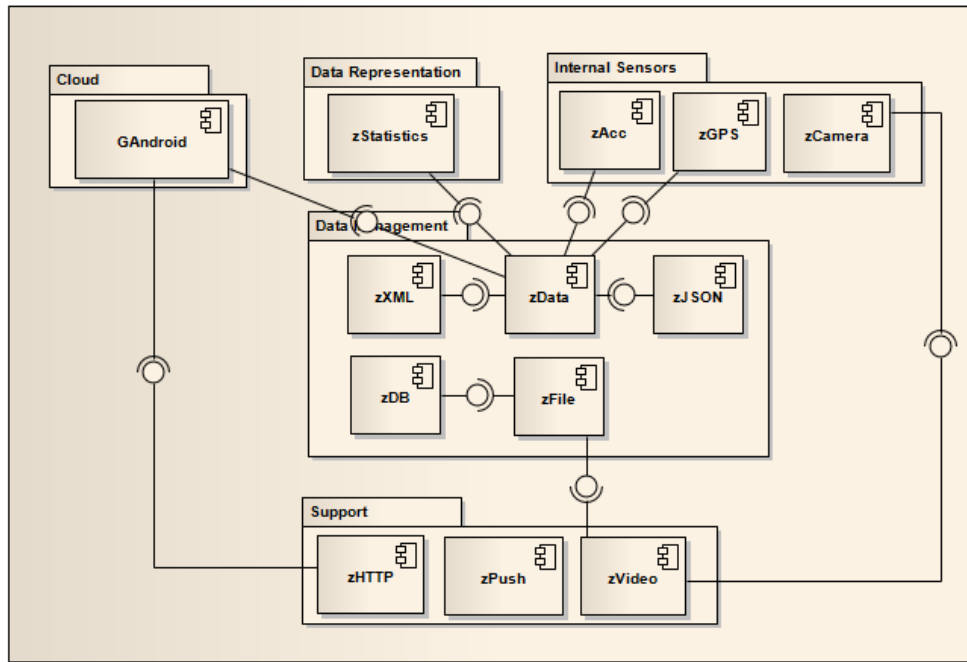


Figura 60 - Arquitectura de la plataforma Zappa

Paquete “Data Management”

Para la gestión de datos internos (a nivel de dispositivo móvil) se proporcionan los siguientes componentes:

- **zDB:** Componente orientado a la gestión de bases de datos internas basadas en un SGBD SQLite. Este componente permite realizar cualquier operación a nivel de base de datos (*insert, delete, update, etc.*), así como, automatizar la creación y gestión de la base de datos.
- **zFile:** Componente para la gestión de la fuente de almacenamiento externo e interno. Creación, lectura, escritura y eliminación de ficheros para su posterior tratamiento a través de la propia aplicación.
- **zXML:** Componente para gestionar ficheros o cadenas de caracteres en notación XML.
- **zJSON:** Componente para gestionar ficheros o cadenas de caracteres en notación JSON.
- **zData:** Componente para transformar las especificaciones en algún tipo de notación en objetivos software, como pueden ser JSON o XML o las tuplas de datos enviadas por el GPS (latitud y longitud), el acelerómetro (coordenadas x, y, z), etc.

Esta gestión automática de los datos posibilita almacenar y recuperar fácilmente información derivada de la propia ejecución de tareas o recuperada de los *wearables*, en plataformas de telemonitorización.

Paquete “Cloud”

Para interactuar con tecnología *cloud* existe un único componente enmarcado en esta categoría, llamado **gAndroid**.

gAndroid es un componente software desarrollado para Android que permite interactuar, por parte del desarrollador, de una manera rápida y fácil con la tecnología G [Gnubila Web].

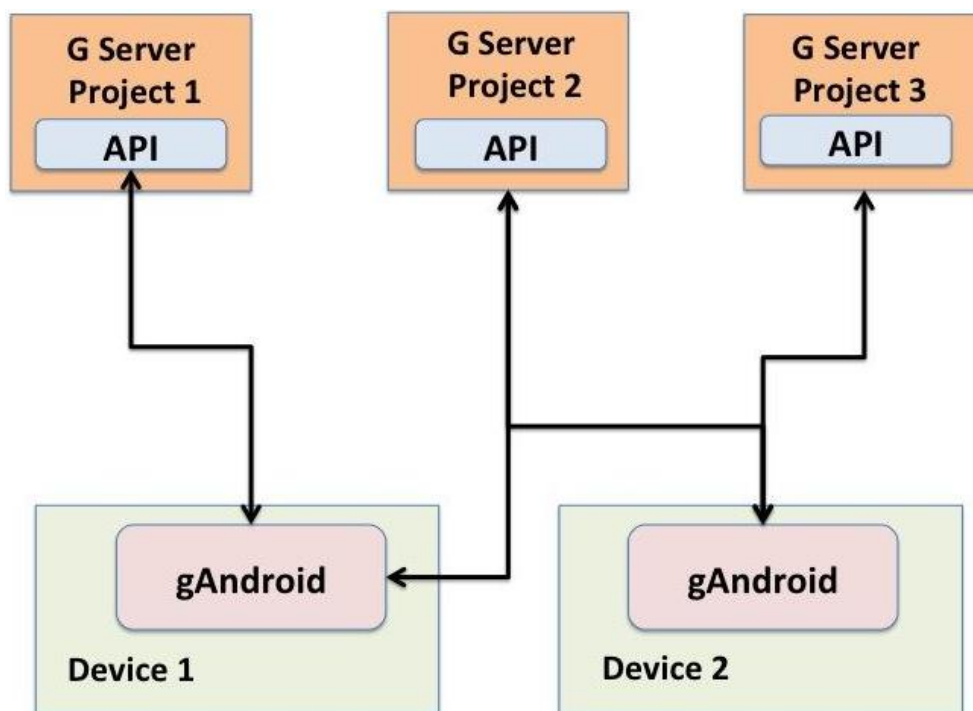


Figura 61 - Gestión de diferentes interacciones con G

Este componente permite:

- Abstractar al desarrollador de la API de servicios (web) de la plataforma G.
- Gestionar automáticamente la autenticación con el servidor y la gestión de cookies, manteniendo de una manera transparente una permanente conexión con la nube.
- Gestionar, bajo el mismo componente, la conexión e interacción con distintos proyectos soportados por la plataforma G (ver Figura 61)
- Gestión de ficheros (subir y descargar) a través de la plataforma G.
- Gestión de cola de acciones, para que en caso de que se pierda la conexión a Internet, el componente almacenaría dichas peticiones en una cola interna que se ejecutarán cuando se vuelva a tener conexión a Internet.
- Funcionamiento como servicio (*gService*) (proceso que se ejecuta en segundo plano), e integración automática con la aplicación para permitir que haya un flujo constante de información entre el dispositivo/aplicación y la nube.

Este componente, además de usar otros componentes de la plataforma Zappa, como *zHttp* (para la gestión de conexiones a través del protocolo http), está compuesto por una serie de elementos software, como puede verse en la figura 62, siendo el elemento *gService* el más relevante de todos ya que proporciona la API de acceso a las funcionalidades al desarrollador o al resto de componentes.

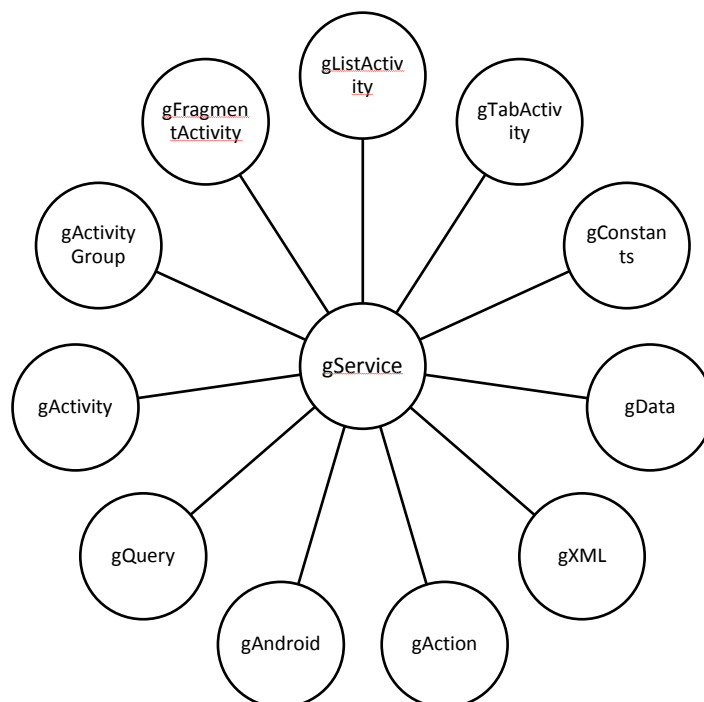


Figura 62 - Artefactos que componen el componente gAndroid

Paquete “Data Representation”

En las plataformas de telemonitorización, la representación de los datos de forma comprensible por los actores es importante para dar un *feedback* adecuado sobre su evolución o el estado de su monitorización.

En la plataforma (Zappa), el componente llamado **zStatistics** permite generar fácilmente representaciones de datos en gráficos (Barras, Sectores, Puntos, etc.).

Paquete “Internal Sensors”

Debido a que es común en las plataformas de telemonitorización utilizar datos como la posición, la cantidad de movimiento o registrar la ejecución de las tareas de telemonitorización, se proporcionan los siguientes componentes:

- **zAcc:** Componente para gestionar el acelerómetro interno del dispositivo móvil, con el objetivo de obtener de manera asíncrona las diferentes variables que determinan la aceleración.
- **zGPS:** Este componente proporciona funcionalidades para la gestión del dispositivo GPS, permitiendo activarlo, desactivarlo, determinar la posición actual, ir recibiendo la información asíncrona cuando se realice un cambio de posición, etc.
- **zCamera:** Gestiona las cámaras del dispositivo móvil, así como las distintas funcionalidades relacionadas con estas (grabación, captura de fotos, etc.).

Paquete “Support”

La plataforma proporciona otros componentes que son usados por el resto de componentes para implementar diferentes funcionalidades:

- **zHTTP:** Proporciona una serie de funcionalidades para interactuar a través del protocolo http y a través de las operaciones del protocolo del mismo (GET, PUT, POST, DELETE). Este componente es uno de los más importantes de la plataforma al ser el componente por el que pasan todas las peticiones a la nube y por donde fluyen los datos que esas peticiones generan.
- **zVideo:** Gestiona ficheros de vídeo, tanto su reproducción como escalado, saltos, etc.
- **zPush:** Automatiza la gestión de notificaciones *push* a través de *Google Cloud Messaging* (GCM).

7.3. WearIt

7.3.1. Introducción

Las plataformas de telemonitorización normalmente hacen uso de *wearables* [Mann 1997] para obtener información acerca de un usuario o su entorno, como puede ser su localización, su frecuencia cardíaca, la temperatura del entorno, etc.

Cada uno de estos dispositivos tiene unas características específicas que dependen y varían de un fabricante a otro, difiriendo en las funcionalidades o datos que proporcionan (pulso cardíaco, geolocalización, temperatura ambiente, etc.) y la tecnología en la que se basa (WiFi, Bluetooth, Serie, etc.).

Esta heterogeneidad también se observa en las interfaces de acceso a las funcionalidades del dispositivo (empaquetadas normalmente como APIs), de manera que cada dispositivo suele ofrecer una API distinta.

Debido a la singularidad de cada *wearable*, la forma de crear sistemas y aplicaciones basados en estos dispositivos normalmente implica que los programadores estudien una hoja de especificaciones —conocida como *datasheet*— o manual/documentación, que proporciona cada fabricante donde se especifica las características del dispositivo, la serialización de los datos, el tipo de operaciones, etc., es decir, especificar la forma de cómo interactuar con el dispositivo.

Dicho proceso es único para cada *wearable*, en el sentido de que cada programador realiza su propio proceso para interactuar con él. Esto ocasiona que dos programadores distintos generen soluciones distintas y rara vez interoperables (cada uno estructurará el código fuente según su criterio y habilidades), dificultando la creación de soluciones reutilizables y generando diferentes maneras de interactuar con el dispositivo.

Esto no sólo dificulta la interacción con el dispositivo o *wearable*, incrementando el tiempo de desarrollo del proyecto, sino que implica una complejidad adicional para el programador al tener que comprender el funcionamiento del dispositivo a bajo nivel para poder integrarlo en un sistema o aplicación.

En esta sección se presenta una propuesta que pretende abordar, dar soporte y solucionar los dos principales problemas antes mencionados: heterogeneidad (dispositivos distintos con numerosas funcionalidades y basados en diferentes tecnologías que van evolucionando y cambiando) y la inexistencia de APIs de alto nivel que permitan acceder a las funcionalidades o datos que proporciona un *wearable* por parte de los programadores.

En la literatura existen proyectos o estudios que pretenden abordar el problema de la integración e interacción con un *wearable*, con el objetivo de abordar esta heterogeneidad y falta de estandarización en el acceso a las funcionalidades de un *wearable*.

Los proyectos más relevantes en la literatura presentan diseños o herramientas (por ejemplo, *middleware*) que pretenden mejorar el proceso de integración de un *wearable*. Por ejemplo, en [Jo 2008] se presenta una aproximación para mejorar el proceso de integración de *wearables* basados en Bluetooth. En [Carr 2012] y [Castillejo 2013] se presenta un *middleware* para la integración de *wearables* en soluciones de e-Salud, y [Hadim 2006] se presenta un estudio completo de diferentes *middlewares*.

La mayoría de los *middlewares* desarrollados para la integración de *wearables* tienen la principal desventaja de que están asociados a un protocolo de comunicación concreto, como el expuesto en [Jo 2008], o bien centrados en un área en particular como [Carr 2012] y [Castillejo 2013], lo que implica que solo funciona con un número determinado de dispositivos. Estos *middlewares* normalmente no dan soporte a la heterogeneidad de dispositivos, ocasionando que los programadores tengan que desarrollar nuevo código fuente para usar nuevos dispositivos.

Otras propuestas han abordado el problema de la heterogeneidad y falta de estandarización en el acceso a través de un enfoque basado en modelos. Por ejemplo, [Akbal-Delibas 2009] y [Losillas 2007] proponen soluciones para la adquisición de datos de un *wearable* a través de una aproximación basada en modelos. Otras propuestas más completas abordan la heterogeneidad y la falta de estandarización en el acceso, como pueden ser las propuestas por [Bellifemine 2011] y [Fortino 2013].

No obstante, las propuestas basadas en modelos que pretenden abordar los problemas de integración e interacción, como las presentadas en [Bellifemine 2011] y [Fortino 2013], tienen limitaciones, entre las que destacan:

- La integración de un nuevo *wearable* y la gestión de datos en un nuevo sistema no está soportado completamente.
- El uso de estas soluciones está asociado a un conocimiento técnico que implica una etapa de aprendizaje para el uso de la propia herramienta.

7.3.2. Práctica actual para la interacción con *wearables*

Independientemente de la tecnología en la que se base un *wearable* (o dispositivo), y de cómo se estructure la información que envía o recibe, los dispositivos están diseñados para serializar y gestionar la información en vectores (*arrays*) de bytes con una longitud y estructura estática o dinámica.

El diseño particular de un *wearable* hace que cada uno tenga una API que proporciona la información serializada con una estructura concreta. Esta serialización de la información puede ser totalmente distinta a la de otros dispositivos, aun teniendo el mismo propósito (medir la frecuencia cardíaca, por ejemplo), ya que han sido diseñados por otros fabricantes, con otros requisitos y criterios. De esta manera, el tamaño de paquete, la información enviada o incluso el orden en que se envía la información pueden ser diferentes.

La Figura 63 muestra un ejemplo de posibles serializaciones de datos de *wearables* con la misma y distinta funcionalidad (dos pulsómetros y un GPS). Ambos pulsómetros sirven para proporcionar el pulso cardíaco (PC), pero cada uno envía un paquete de información de distinta longitud (16 y 32 bits), proporciona distinto tipo de información (por ejemplo, el pulsómetro 1 proporciona el pulso cardíaco y el intervalo RR, entre otros campos) y además está estructurada de distinta manera (orden de los campos). El tercer dispositivo (GPS), facilita en 8 bits la latitud, longitud y altitud.

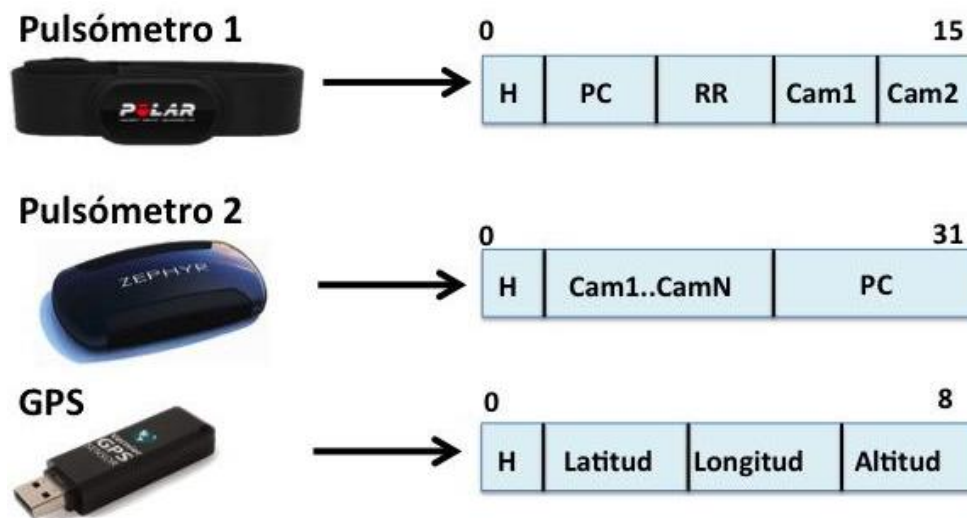


Figura 63 - Serialización de datos de diferentes wearables

El método de integración de un *wearable*, tal y como se realiza en la mayoría de los casos, pasa por un estudio de numerosas tecnologías antes de empezar la etapa de implementación. Este método de integración e interacción está constituido por una serie de etapas donde en cada una de estas se abordan diferentes retos de ámbito tecnológico.

Este método, que sigue un programador de cara a interactuar con un *wearable*, se compone de cuatro etapas (ver Figura 64):

1. Conocer las características del dispositivo y analizar cómo interactuar con él, estudiando el *datasheet* (información proporcionada por el fabricante con toda la información, cómo por ejemplo el tamaño, peso, tecnología que usa y las funcionalidades o datos que provee), y otra información necesaria para tener un conocimiento lo más completo posible sobre el dispositivo.
2. Dependiendo de la tecnología del dispositivo y de los requisitos del sistema a construir, el programador probablemente tenga que realizar una segunda fase de aprendizaje sobre alguna tecnología en concreto, ya sea por desconocimiento sobre cómo funciona un protocolo de comunicación o cómo integrarlo en la plataforma o tecnología sobre la que vaya a construir la solución (Android, iOS, Linux).
3. Una vez realizadas estas dos etapas de estudio, el programador ya puede comenzar con la implementación del nuevo código fuente que usará para poder trabajar con el dispositivo e integrarlo en el sistema.
4. Con este código fuente, creado a medida, ya puede usar ese dispositivo en concreto.



Figura 64 -Método actual para la integración (o uso) de un wearable

Dicho método para la integración de un *wearable* en un sistema presenta desventajas notables, entre las que destacan:

- **Incremento de tiempo de desarrollo.** El hecho de que un programador, cuya tarea inicial sería llevar a cabo un proceso de implementación, tenga que llevar a cabo etapas de aprendizaje adicionales que incluyen el estudio de detalles de bajo nivel para cada dispositivo en concreto, incrementa considerablemente el tiempo del proceso de desarrollo.
- **Complejidad de mantenimiento.** Este método de integración implica que un cambio en algún parámetro (software o hardware) en un futuro para satisfacer un nuevo requisito o funcionalidad, puede implicar la necesidad de volver a iniciar de nuevo el ciclo de desarrollo al necesitar conocimiento adicional además de una nueva etapa de desarrollo.

Además, este método implica que el cambio en una etapa daría lugar a la revisión en cascada de las sucesivas etapas, de tal forma que la integración de un nuevo *wearable* en el sistema implicaría implementar nuevo código. Esto ocasiona que esta forma de desarrollar sea muy variable en el sentido de que el menor cambio en alguna de las etapas afecta al resto, repercutiendo en una inversión de tiempo y esfuerzo.

El método actual de integración de un *wearable* es un proceso costoso no exento de retos. Este método presenta además claras desventajas que afectan directamente en el tiempo de desarrollo del proyecto, así como en su difícil mantenimiento.

7.3.3. Nuevo proceso para la interacción con *wearables*

El método actual para la integración de un wearable, independientemente del sistema donde se pretenda utilizar, conlleva un proceso costoso y que presenta numerosas desventajas, como la difícil reutilización de soluciones o el incremento del tiempo de desarrollo.

Partiendo de estas desventajas, con el objetivo de proponer soluciones para las mismas y simplificar así la integración del *wearable*, se presentan las siguientes premisas que serán pilar fundamental a considerar en el diseño de la nueva propuesta de método:

- **La tarea del programador es interactuar con el dispositivo.** El tiempo que requiere conocer a bajo nivel cómo funciona un *wearable* repercute directamente en el tiempo invertido por un programador para usar un dispositivo.
Con el nuevo método la única responsabilidad del programador va a consistir en implementar el código fuente necesario para interactuar con el dispositivo, es decir, acceder a la información que proporciona mediante interfaces y funciones de alto nivel, abstrayéndose de detalles de bajo nivel, como formato y transmisión de dichos datos.
- **Los cambios en un dispositivo no afectan al proceso de integración.** Aunque haya cambios de firmware o tecnológicos en un dispositivo, éstos no afectarán al proceso de integración del dispositivo en un sistema o aplicación. Esto es, cambios en un *wearable* no implican un trabajo extra significativo de mantenimiento para el programador.
- **Interoperabilidad y acceso común.** Las soluciones software creadas para interactuar con un *wearable* podrán ser reutilizadas por cualquier programador, facilitando así su integración y permitiendo que todos los programadores acceden a las funcionalidades de un *wearable* de la misma manera.

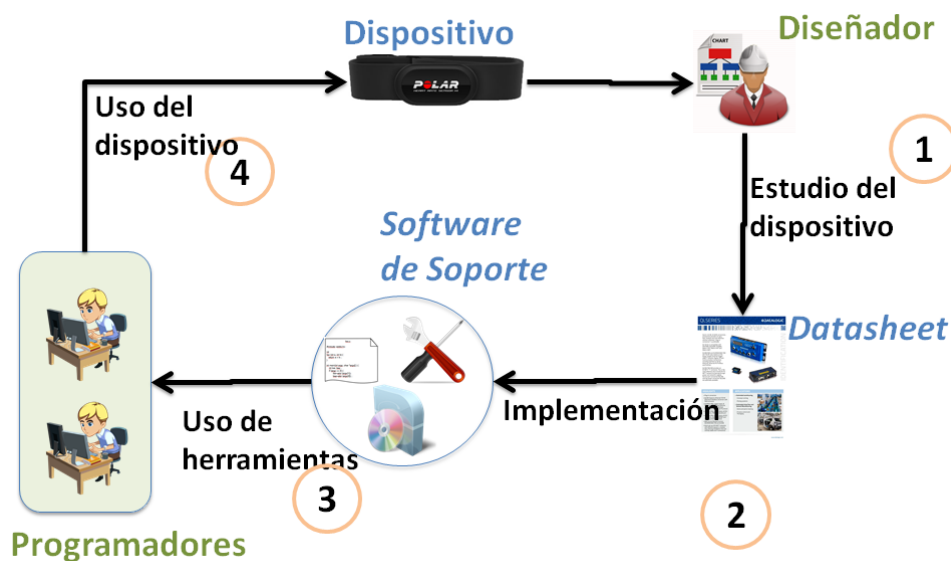


Figura 65 – Método propuesto para la integración de wearables

Este método es similar al método actual de integración (ver Figura 64), en el sentido de que los procesos son similares, pero no se comparte el actor que los lleva a cabo:

1. Estudio del dispositivo, al igual que en el método actual, usando el *datasheet* o cualquier otra información necesaria.
2. Implementación de las herramientas para usar dicho dispositivo, con el estudio tecnológico previo y necesario de cara a la etapa de implementación.
3. El programador, usando las herramientas o software de soporte, ya puede comenzar con la implementación del nuevo código fuente que se usará para poder trabajar con el dispositivo e integrarlo en el sistema.

4. Con este código fuente, creado a medida, ya puede usar ese dispositivo en concreto.

La diferencia de este método (ver Figura 65) con respecto al método actualmente usado (ver Figura 64) difiere en dos aspectos fundamentales:

- **Roles:** La identificación de los distintos roles presentes en el nuevo método (programador y diseñador) y las responsabilidades de estos roles ayudará a balancear o distribuir la carga de trabajo mejor para el programador, en términos de eficiencia, la integración final de un *wearable*. Como se verá a continuación, se define un nuevo rol con respecto al método actual, llamado diseñador.
- **Software de soporte:** Además de los roles, estos necesitan ciertas herramientas software para llevar a cabo las tareas que tienen asignadas. Esto es, el software que desarrolle el diseñador y que permita interactuar con el *wearable*, y que usarán los programadores para integrarlo y usarlo en el sistema o aplicación correspondiente.

Con el objetivo de reducir en la medida de lo posible la carga de trabajo del programador, siendo este además el único rol en el método actual (Figura 63), se propone en este nuevo proceso dos roles de acuerdo al nivel de conocimientos y por tanto a su responsabilidad. Estos dos nuevos roles son:

- **Diseñador.** El diseñador tiene la responsabilidad del proceso correspondiente a la obtención de los conocimientos necesarios para saber cómo funciona el *wearable*, como interactuar con él y facilitar las herramientas necesarias a los programadores para poder integrar dicho dispositivo, esto es, la etapa de estudio del *wearable* (1 en la figura 65). Además, el diseñador creará el software de soporte que permita interactuar con el *wearable* (2).
- **Programador.** El usuario que crea del código de aplicaciones o sistemas y el que integra el *wearable* en el sistema. La diferencia con respecto al programador del método actual (Figura 64) es que este programador no tiene ni requiere conocimiento experto para usar el *wearable*, por lo que evita esa fase de aprendizaje de cómo funciona el *wearable* (ahorrando tiempo y esfuerzo) y pasa directamente a usarlo e integrarlo en su sistema, gracias a las herramientas o artefactos software que el diseñador le ha facilitado (etapas 3 y 4 de la figura 65).

El diseñador, con una fase de un estudio previo si fuese necesario, crea el software para interactuar con el *wearable* (código fuente, conjunto de librerías o clases, componente software). Estas herramientas serán usadas por el programador para integrar el dispositivo, abstrayendo a este de los detalles internos del mismo y sus características o tecnología.

Esto permite que para usar un *wearable* todos los programadores reutilicen la misma herramienta, consiguiendo que el código sea interoperable. Además, se consiguen reducir los diferentes procesos de estudio, desarrollo e integración (un proceso por cada *wearable* en particular y programador) del método actual (Figura 66), ya que con el nuevo método se realizaría un único proceso de estudio y desarrollo para cada *wearable* y varios de integración para cada uno de los programadores (Figura 67).

Considerando el método propuesto, y partiendo de la división de roles para la división de la carga de trabajo, el diseñador proporciona una herramienta, llamada software de soporte, que permite al programador usar el *wearable*.

Este software de soporte, dependiendo de la implementación que se haga por parte del diseñador, puede variar desde funcionalidades independientes para permitir el uso del *wearable*, hasta un componente software bien estructurado, o bien librerías.

Puesto que un software de soporte es creado explícitamente para un *wearable* (existiendo por lo tanto un software particular para cada *wearable*), todos los programadores que necesiten usar un *wearable* particular usarán el mismo software de soporte, estandarizando así el acceso y permitiendo por lo tanto la interoperabilidad de código entre dos proyectos donde se use el mismo *wearable*.

Además, errores detectados en el uso de un *wearable* por parte del diseñador, se solucionarían sustituyendo el software de soporte por una versión actualizada, evitando así que el programador tenga que realizar ninguna tarea de mantenimiento.

El método propuesto pretende solucionar los problemas derivados de la metodología actual (incremento del tiempo de desarrollo y problema de mantenibilidad).

Para esto, se hace énfasis en dos aspectos: (1) diferentes roles dependiendo de la responsabilidad (diseñador y programador) y (2) el uso de software de soporte para permitir a todos los programadores usar el *wearable* de la misma manera.

Además, este método da pie a la interoperabilidad de código y dar soporte a cualquier *wearable* ahorrando tiempo con respecto a la metodología actual, por lo que simplifica el proceso de integración (para el programador) y ahorra tiempo.

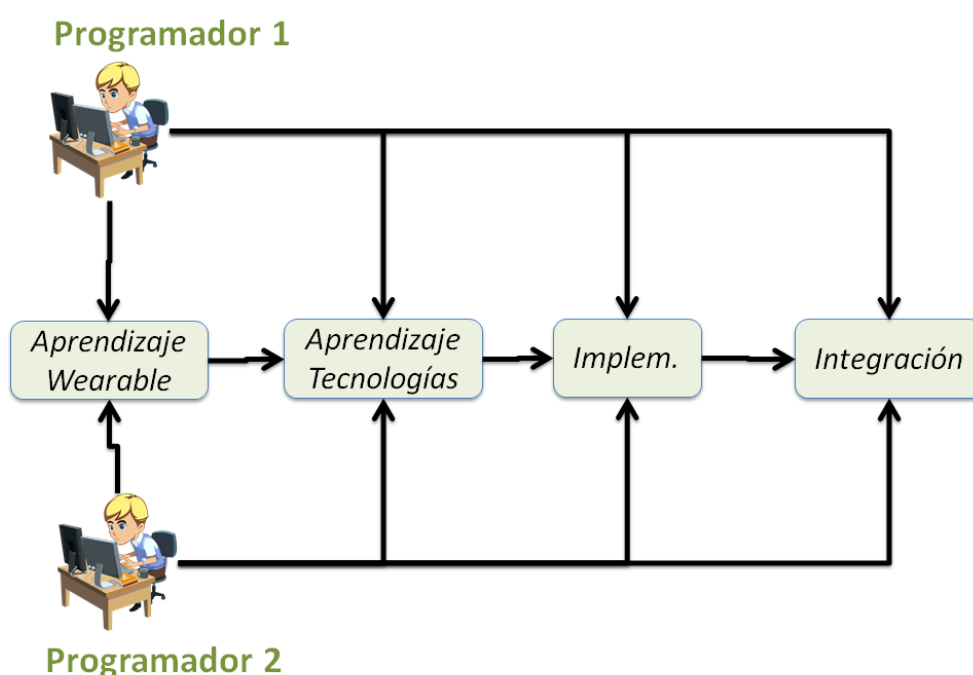


Figura 66 - Integración de un wearable con el método actual

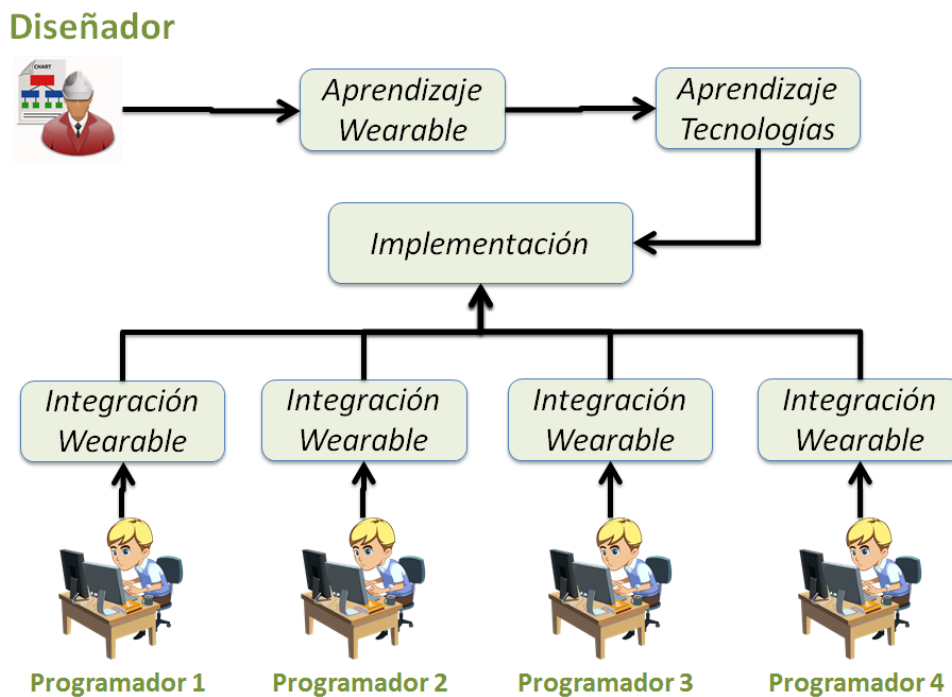


Figura 67 - Integración de un *wearable* usando el método actual

7.3.4. WearIt

7.3.4.1. Introducción

WearIt es una herramienta para la integración de *wearables* que pretende solucionar los problemas de heterogeneidad y falta de estandarización en el acceso de este tipo de dispositivos.

WearIt se ha concebido para dar soporte al método para la integración de wearables propuesto en la sección anterior, considerando como usuarios de la herramienta tanto a programadores como a diseñadores, y proporcionando los diferentes elementos que permitan construir el software de soporte para integrar el dispositivo.

Para dar soporte a la segunda etapa del método propuesto, se ha optado por un enfoque dirigido por modelos. Un modelo permitiría representar todas las características de un *wearable*, así como añadir especificaciones que indican cómo interactuar con él para su integración.

Además de la propia representación de un *wearable* como un modelo, la generación de un modelo general o metamodelo permitiría representar cualquier característica de un *wearable* y añadir nuevas para cumplir futuros requisitos o funcionalidades.

Del mismo modo, el uso de modelos garantiza que estos podrían consumirse sin muchos recursos, al ser solo necesario descargarlos, y por la propia naturaleza de un enfoque dirigido por modelos, estos al estar definido en un lenguaje estándar como XML, JSON serían multiplataforma e interoperables.

Con respecto a la etapa 3, con el uso de este software (que hemos definido como modelos) para interactuar con el wearable, hay que considerar que el uso del *wearable* viene directamente especificado en el modelo y por lo tanto dicho software debe soportar todas las funcionalidades del modelo en cuanto a cómo acceder a las funcionalidades del *wearable*.

Se hace necesario que este software sea lo suficientemente dinámico y extensible para añadir nuevas funcionalidades que se correspondan con cambios en el modelo de los wearable y además que permita gestionar toda la información para abstraer al desarrollador de conceptos de bajo nivel, proveyendo la información útil para el desarrollador.

Partiendo de este requisito, se ha optado por una solución orientada a componentes software. Un componente software permite encapsular funcionalidades en una caja negra y ofrecer dichas funcionalidades a través de una interfaz bien definida [Szyperski 1995].

Por lo tanto, la adición de nuevas funcionalidades implicaría programar el código necesario para dar soporte a estas funcionalidades y añadir su correspondiente método de acceso en la interfaz. Esto posibilita una extensibilidad en cuanto a las funcionalidades que soporta un sistema, además de otras muchas ventajas como ya se ha visto en el capítulo 3.

Debido a que el uso de *wearables* se usa principalmente en escenarios concretos como WBAN y soluciones parecidas, se propondrán una serie de características y requisitos básicos e implementar una versión de este software en las distintas plataformas.

7.3.4.2. Elementos que conforman WearIt

WearIt está compuesto por cuatro elementos distintos (ver Figura 68):

- **Metamodelo.** El metamodelo pretende representar una serie de atributos que permiten representar cualquier *wearable*, soportando nuevos atributos para permitir la representación de nuevos dispositivos en un futuro de acuerdo a la aparición de nuevas tecnologías.
- **Herramienta de diseño.** Se pretende dotar a los usuarios con conocimiento experto (diseñadores) de una herramienta para que con cierta facilidad pueden crear diseños de los mismos que represente sus características (modelos), así como, la API para interactuar con el dispositivo (Etapa 2).
- **Generación de modelos.** Un proceso de generación de modelos para generar un modelo para cada dispositivo a partir del metamodelo (instancias del metamodelo) (Etapa 2).
- **Coordinador.** El coordinador es un componente software diseñado y desarrollado para transformar esos modelos de dispositivos en elementos software para así poder integrar e interactuar con el dispositivo a través de una API de una manera fácil y sin necesidad de conocer detalles de bajo nivel por parte del desarrollador (Etapa 3).

En las siguientes secciones se describen en detalle cada uno de estos elementos.

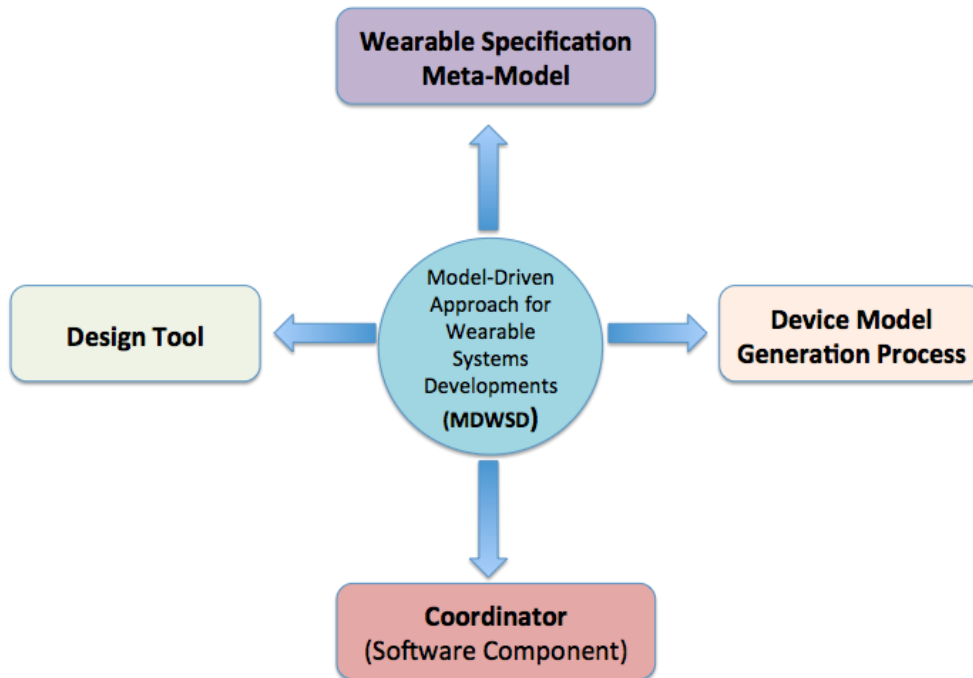


Figura 68 – Elementos (artefactos) que conforman WearIt

7.3.4.3. Metamodelo

La heterogeneidad en los *wearables* hace que cada dispositivo tenga unas características concretas, que van desde la tecnología que usa, a como proporciona la información o cómo acceder a la misma.

Con el objetivo de tener una solución que represente cualquier característica de un dispositivo con el objetivo de abordar dicha heterogeneidad, así como los elementos necesarios para definir la manera de interactuar con el (API), en la propuesta presentada se ha optado por un enfoque basado en modelos.

Un enfoque basado en modelos permitiría modelar cualquier concepto siguiendo un estándar (UML o XML) y, además, usar las técnicas ya documentadas en la literatura para la transformación de modelos, pudiendo así generar sub-modelos o instancias del meta-modelo con unas características concretas y que representen a un *wearable* determinado.

Por lo tanto, se ha diseñado un metamodelo con esta premisa presentado en la Figura 69.

Este metamodelo es, por definición, dinámico, pudiendo así representar nuevas funcionalidades y características para asegurar que cualquier dispositivo puede representarse en un modelo, independientemente de sus características y funcionalidades.

El metamodelo es una representación conceptual de todos los posibles parámetros/características (hasta la fecha) que suelen estar presentes en un *wearable*, así como una especificación adicional para definir cómo interactuar con el mismo (API).

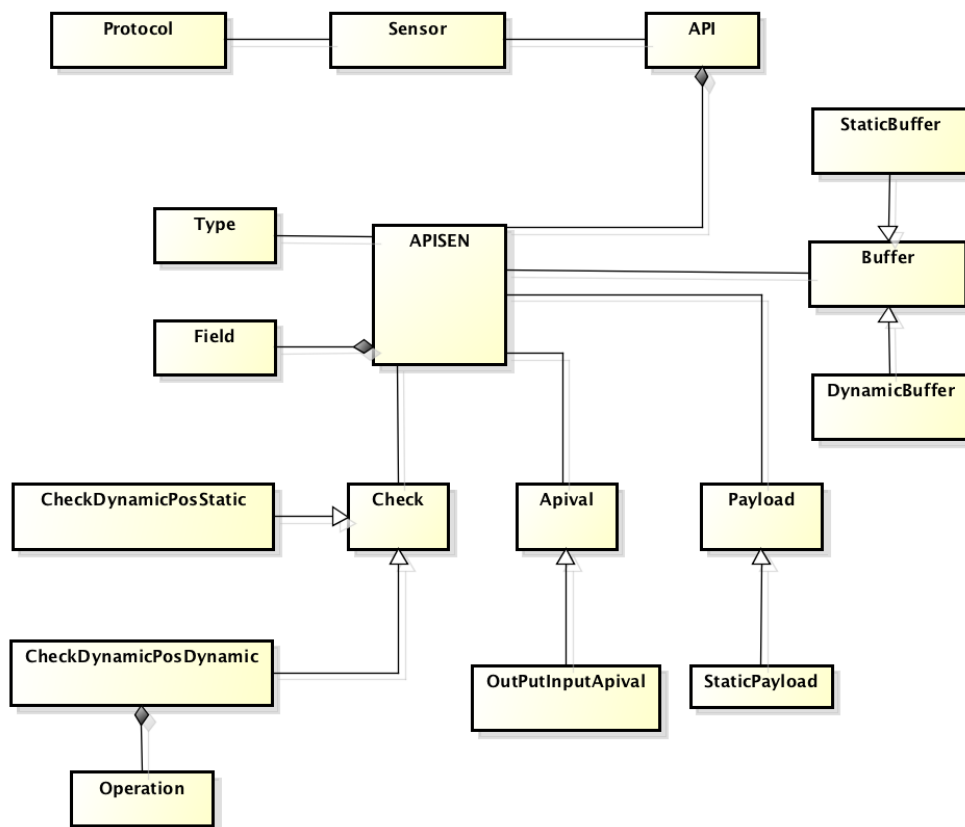


Figure 69 - Metamodelo para la representación de wearables

Para diseñar el metamodelo se han tenido en cuenta conceptos cruciales/importantes que han sido detectados como característica fundamental de cualquier *wearable*. Estos conceptos son:

- Todo *wearable*, sobre todo los más actuales, tiene un protocolo de comunicación que permite comunicarse con el exterior para intercambiar información. Aunque el protocolo de comunicación está especificado en el metamodelo como una característica, su gestión se realiza en el coordinador, ya que, aunque el protocolo de comunicación sea un estándar, dependiendo de la plataforma se usará dicho protocolo de una manera u otra.
- Los dispositivos proporcionan la información a través un vector de bytes (serializada) y de dos maneras posibles: a través de un flujo constante (*stream*) o mediante peticiones (*request*).
- La información que proporciona un wearable difiere, por lo general, en relación a cualquier otra en cuanto a estructura, contenido y tamaño.
- Cuando un dispositivo proporciona información donde el tamaño del vector serializado es dinámico/variable, este tamaño viene especificado en el propio vector en una posición concreta.
- Cualquier información que se envíe desde un wearable, al proporcionarse en un vector serializado, viene marcado por un valor que representa la cabecera (*header*).
- La información que envía un wearable no tiene por qué ser correcta, ya que por problemas de transmisión de datos o internos del *wearable* puede enviarse corrupta, incompleta, etc.

Por esto, cada paquete de información tiene un campo que determina si el paquete es correcto o no.

- La información proporcionada por un wearable está agrupada en bytes. La longitud y posición de estos grupos de bytes puede ser predeterminada y estática o variable.

A partir del modo en que se gestiona la información a través de los protocolos de comunicación, se han tenido en cuenta distintas características que se han incluido en el metamodelo con el objetivo de modelar cualquier dispositivo:

- **Tipo de operación:** Puede ser de entrada (stream-flujo), salida o ambas (petición).
- **Elemento *Pivot*:** Es el elemento que indica dónde comienza la información enviada por un dispositivo para una funcionalidad concreta. Cada petición de servicio a un dispositivo genera un paquete de información, pudiendo compartir todos el mismo *pivot* o no.
- **Información relevante (*payload*):** La parte de la información transmitida donde se encuentra la información relevante. Esta sería la información que espera obtener el desarrollador y por el que decide usar dicho servicio (en el caso de un dispositivo de pulso cardíaco, el *payload* sería el valor que representa el pulso cardíaco).
- **Campos:** Distinta información enviada desde el dispositivo. Además del *payload*, existen otros campos dentro de un paquete que proporcionan información adicional.
- **Método de verificación:** Un conjunto de operaciones para validar la información del paquete. Puede ser un simple valor que indica que el paquete es correcto, un campo que indica la longitud del paquete o un conjunto de operaciones.
- **Tipo de buffer de cada funcionalidad/servicio:** Existen dos tipos de buffer, estático o dinámico, esto es, con tamaño o longitud fija o variable, respectivamente.

Uno de los objetivos del metamodelo es conseguir que sea personalizable, de tal manera que puedan representarse en un futuro nuevas tecnologías y características de los dispositivos a partir de las demandas del mercado, o de a la propia evolución de la tecnología.

De esta manera, es posible crear nuevos modelos para nuevos dispositivos añadiendo sencillamente nuevos elementos al metamodelo que representen nuevos conceptos o tecnologías, para soportar estas nuevas características.

Los modelos generados desde el metamodelo (instancias) están representados en un lenguaje de marcas específico llamado WML (*Wearable Markup Language*), donde los elementos del metamodelo son las etiquetas del propio lenguaje.

Estos modelos contienen toda la información necesaria para interactuar con el wearable: protocolo de comunicación usado, valores y funcionalidades proporcionadas, información del propio wearable, etc., y son sencillamente partes del metamodelo con valores concretos de acuerdo a las características del dispositivo (ver Figura 70).

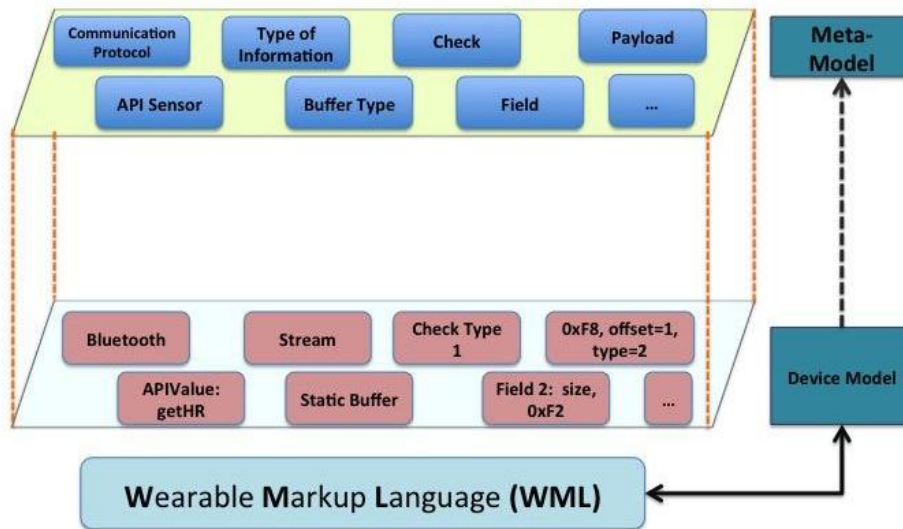


Figura 70 – Instanciación del metamodelo en un modelo de un wearable

7.3.4.4. Herramienta de Diseño

La herramienta de diseño es una plataforma web creada y desarrollada para permitir el diseño de *wearables*, esto es, crear el modelo del *wearable* (características, protocolo de comunicación, tipo de información) así como los servicios que proporciona, por parte del diseñador (ver Figura 71).

El diseñador utiliza esta herramienta para modelar el dispositivo. Por ejemplo, si está trabajando con un dispositivo de pulso cardíaco el diseñador define los distintos elementos contenidos en un paquete de información, desde la cabecera hasta el *payload*.

Esta plataforma, una vez el diseñador haya modelado el dispositivo, será la encargada de generar el modelo usando el proceso de generación de modelos explicado en el siguiente capítulo, facilitando al programador (y al público en general) el identificador del dispositivo (*WUI - Wearable Unique Identifier*) y la API.

De igual manera, cambios que sean necesarios realizar en un modelo para solucionar algún error o añadir alguna funcionalidad se harán a través de la propia herramienta de diseño, con el objetivo de que la tarea del mantenimiento del dispositivo sea fácil.

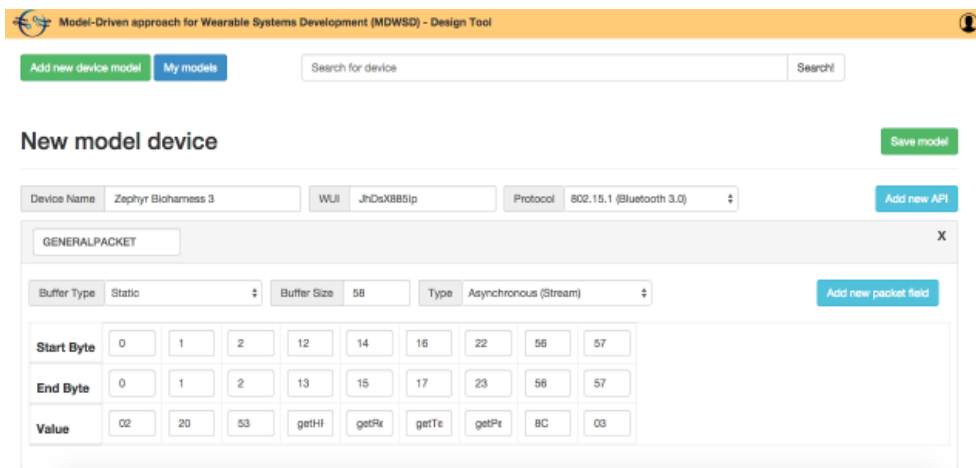


Figura 71 - Herramienta de diseño de WearIt

7.3.4.5. Generación de modelos

El metamodelo contiene las distintas características de cualquier dispositivo de acuerdo al trabajo de investigación realizado hasta la fecha actual: protocolo de comunicación, mecanismo para el envío de información, método de verificación de paquete, etc.

Para definir un modelo de un dispositivo (instancia del metamodelo), algunas de estas características serán usadas para representar las características del dispositivo, siendo necesario un proceso para obtener modelos a partir del metamodelo de acuerdo a unas características concretas.

Este proceso de a partir de un metamodelo generar instancias del mismo se consideraría dentro de la rama de transformación de modelos. Dentro de las posibles herramientas, métodos y tecnologías existentes para la transformación de modelos, se ha optado por realizarla a través de una transformación XSLT (*Extensible Stylesheet Language Transformation*) [XSLT Web].

Se ha optado por esta tecnología ya que cumple con las necesidades impuestas en esta propuesta, que se basa en generar a partir de un modelo general un submodelo donde solo aparezcan algunos atributos del metamodelo y con un estado en concreto (valores). Aunque hay otras herramientas relacionadas con la ingeniería de modelos como QVT o ATL, complicaría la propuesta sin aportar ninguna ventaja extra.

El metamodelo representado en UML en la figura 69 tiene un homólogo en XLST y el diseñador, usando la herramienta de diseño, diseña/modela el dispositivo definiendo las características del dispositivo, así como su API, generando una hoja de especificaciones (fichero XML) basado en XSLT que representa estas características del dispositivo.

Un servicio, que usa un procesador XSLT, usando la hoja de especificaciones generada por la herramienta de diseño junto con el metamodelo (fichero XML) almacenado en un servidor o en la nube, genera un fichero personalizado que representa el modelo del dispositivo en el lenguaje antes mencionado *Wearable Markup Language* (WML) (ver Figura 72).

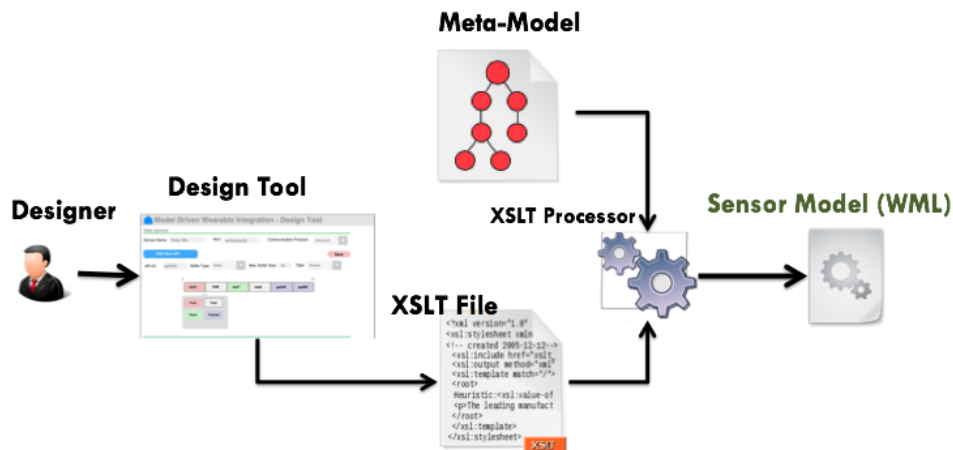


Figura 72 - Generación de modelos usando XSLT

7.3.4.6. Coordinador

El coordinador es el software responsable de servir de capa intermedia entre el desarrollador (sin conocimiento acerca de detalles de bajo nivel) y el dispositivo.

Este software se encarga de la integración de los modelos (detalles a bajo nivel) y su traducción en elementos software que permitan la automatización de la integración y uso del dispositivo por parte del desarrollador.

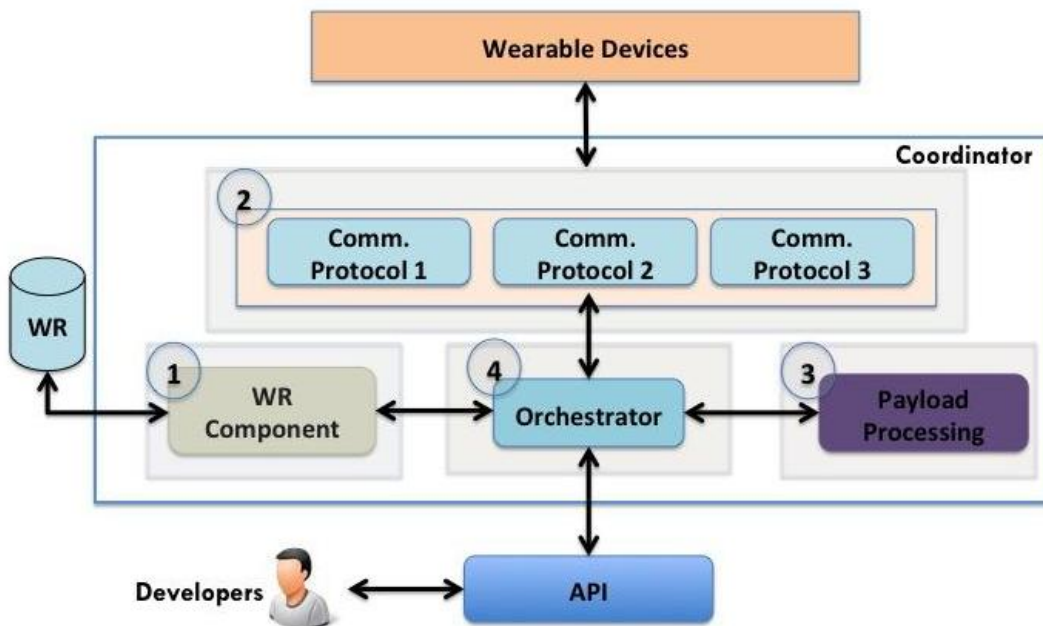


Figura 73 - Arquitectura del coordinador

La figura 73 muestra la arquitectura del coordinador y los distintos artefactos que lo componen. Son cuatro los elementos/artefactos que componen el coordinador, cada uno con un propósito específico:

1. **Repositorio de modelos (WR: *Wearable Repository*)**: Este componente es el responsable de interactuar con un repositorio alojado en un servidor externo (o en la nube) donde se encuentran todos los ficheros correspondientes a todos los modelos de los dispositivos.

El componente (marcado como 1 en la Figura 73) usa el servicio especificando el WUI (*Wearable Unique Identifier*) para obtener el modelo en concreto.

2. **Protocolos de comunicación:** Con el objetivo de manejar los distintos dispositivos, es necesario interactuar con estos a través de diferentes protocolos de comunicación, tales como Bluetooth, Wifi o ZigBee. Debido a esto, existe un componente software por cada protocolo de comunicación (marcado como 2). Todos los dispositivos basados en el mismo protocolo de comunicación son gestionados por el mismo componente.

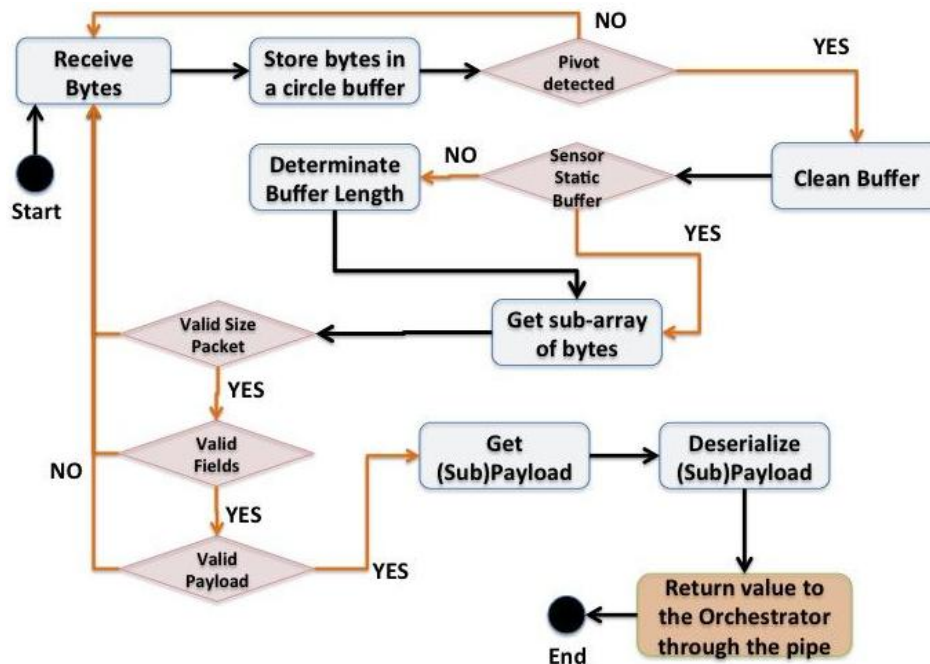


Figura 74 - Algoritmo para la obtención del *payload*

3. **Procesamiento de la información relevante (*payload*):** Una vez que el dispositivo está enviando información, es necesario procesar toda esta información para en primer lugar asegurarse que es correcta y obtener la información que el desarrollador realmente necesita. Este componente (marcado como 3) es el responsable de transformar el vector de bytes que envía el dispositivo en un valor (numérico, texto, etc.) que el programador es capaz de entender (asegurando la calidad del servicio –QoS), dependiendo del dispositivo (ver Figura 74).
4. **Orquestador:** Es el componente responsable de consumir los servicios de los tres componentes mencionados y orquestarlos para proporcionar la información correcta al desarrollador a través de la API. La API también puede ser usada por el desarrollador para interactuar con el coordinador, con el objetivo de conocer qué dispositivos se están usando, otros parámetros de los dispositivos almacenados internamente, activar o desactivar un dispositivo o activar o desactivar notificaciones.

Con el objetivo de gestionar una interacción asíncrona con los dispositivos, el coordinador usa un elemento software propio e interno llamado *pipe*. Un *pipe* es un elemento software

dirigido por eventos que es el responsable de transmitir la información desde el dispositivo hasta programador (vector de bytes à pulso cardíaco, latitud, longitud, etc.).

De esta manera, cuando el programador usa el coordinador para gestionar un dispositivo, el coordinador le devuelve al desarrollador un *pipe* (un *pipe* por cada dispositivo). Cuando el dispositivo envía la información, esta es procesada, verificada internamente por el coordinador y enviada a través del *pipe* correspondiente al programador, permitiendo gestionar dicha información en función a sus necesidades.

7.3.4.7. Aplicación del método propuesto

El método propuesto (Figura 65) presenta dos roles (diseñador y programador), donde el diseñador crea un software (software de soporte) para interactuar con un *wearable* y el programador usa dicho software.

Este método tiene diferentes posibilidades en relación al software de soporte. Cómo se ha comentado en la descripción de la propuesta, este puede ser desde un simple fichero de código con una serie de funciones implementadas que permitan interactuar con el *wearable*, hasta un componente software bien definido y ya compilado, o una librería.

El principal problema de entender el software de soporte del método propuesto como una implementación concreta, esto es, código fuente o compilado (independientemente de cómo esté estructurado), es la gran cantidad de trabajo que se requiere si el número de *wearables* es elevado.

Por ejemplo, supongamos una situación concreta donde tenemos un programador, un diseñador y siguiendo el método propuesto se pretende dar soporte a 10 *wearables* distintos para dos plataformas diferentes (Android y iOS), entendiendo el software de soporte como, por ejemplo, una librería.

Siguiendo el método, el diseñador tendría que estudiar cada uno de estos 10 *wearables* para conocer sus características y entender cómo funciona. Además, tendría que aprender las tecnologías de ambas plataformas (Android y iOS) que necesite para poder interactuar con los *wearables* (protocolo de comunicación, tratamiento interno de datos, etc.).

Una vez estudiados los diferentes *wearables* y las tecnologías necesarias, tendría que crear una librería por cada *wearable* y por cada plataforma, puesto que asumimos no existe interoperabilidad entre dos plataformas diferentes y se requieren librerías independientes. Esto concluiría con un total de 20 librerías, una por cada *wearable* y plataforma.

Aunque esta opción sigue solucionado (al programador) el problema del acceso estandarizado y la heterogeneidad, ya que todos los programadores usarían la misma librería para el mismo *wearable* y plataforma, y sería posible soportar cualquier *wearable* (bastaría implementar una nueva librería), es una propuesta laboriosa por la gran cantidad de trabajo necesario para dar soporte a un *wearable* y por su tedioso mantenimiento (cambios en un *wearable* implican cambios en las librerías de las diferentes plataformas, y todos los programadores deben actualizar las diferentes librerías correspondientes a las plataformas).

Para el diseño de la herramienta aquí presentada (**WearIt**), que como soporte al método planteado (diferentes roles, software de soporte, etapas), se ha optado por un enfoque de

integración de *wearables* basados en la representación de un *wearable* a través de un modelo, siendo el software de soporte necesario para su interacción un componente software llamado coordinador.

7.3.4.8. Integración de *wearables*

El proceso de usar un nuevo wearable usando WearIt, bajo el método propuesto, se muestra en la Figura 75.

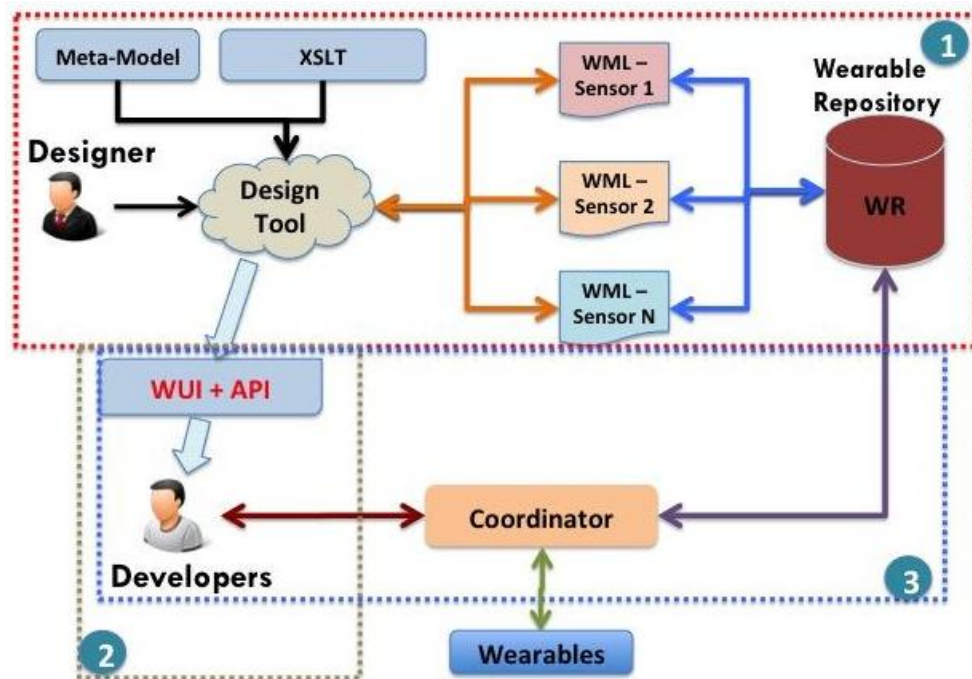


Figura 75 - Proceso de integración de wearables usando WearIt

Los pasos para dicho proceso de integración son:

1. **Diseño:** El diseñador usa la herramienta de diseño para modelar el wearable y su API de acuerdo a sus características y especificaciones indicadas en el *datasheet* o en otra fuente de información. Durante esta etapa, el diseñador usa la documentación del dispositivo para modelar el wearable, definir una API personalizada para interactuar con él, y usar los servicios que éste proporciona. Una vez que el diseñador concluye con la fase de diseño, la herramienta genera una hoja de estilos en XSLT. La plataforma accede a un servicio el cual, con el metamodelo original y esta hoja de estilos generada por la propia herramienta de diseño, genera el modelo del wearable a través del proceso de generación de modelos basado en XSLT. Este modelo se almacena en un directorio o repositorio para ser accedido por cualquier desarrollador.
2. **Publicación de la API:** Una vez que el diseño está finalizado y el modelo generado, la herramienta de diseño proporciona/pública la API especificada por el diseñador y el identificador único para acceder a dicho modelo, WUI (generado automáticamente por la herramienta).
3. **Uso del dispositivo:** Usando la API y el WUI generado por la herramienta de diseño, así como usando la API del coordinador, los desarrolladores pueden gestionar fácilmente el dispositivo, sencillamente usando dicha API y sin necesidad de conocimiento experto.

El proceso de integración a partir de WearIt tiene las siguientes ventajas:

- El programador puede usar cualquier dispositivo (heterogeneidad) sin conocimiento experto de protocolos de comunicación, características internas del dispositivo o cómo funciona internamente.
- La posibilidad de definir un API personalizada y única para un *wearable* garantiza un acceso estandarizado a la información del sensor (todos los desarrolladores usarán la misma API y la información se presentará de la misma manera).
- En caso de que hubiese un error en el modelo del dispositivo, el diseñador puede redefinir el modelo solucionando el problema. Si el modelo no implica cambios en la API del dispositivo, el coordinador usará este nuevo modelo de una manera transparente para los programadores. De esta manera los programadores usarán correctamente el dispositivo sin alterar ni una línea de código.
- Los modelos son independientes de la plataforma y pueden usarse por cualquier coordinador, independientemente de la plataforma que lo soporte (iOS, Android, Windows).
- Cada wearable tiene un único modelo.
- El metamodelo y coordinador están diseñados para ser dinámicos, garantizando el soporte a nuevas tecnologías y requisitos. Cambios en el metamodelo implican cambios en el nuevo modelo del dispositivo y, normalmente, en el coordinador. Por ejemplo, aunque un nuevo protocolo de comunicación pueda ser soportado en un modelo de un dispositivo, debe ser implementado el correspondiente componente software en el coordinador para poder usar dicho protocolo.

7.3.4.9. Caso de estudio

Con el objetivo de ilustrar el proceso de uso o integración de un wearable en una aplicación o software, la figura 76 muestra un diagrama de secuencia que representa el uso del wearable, desde el diseño hasta su uso final.

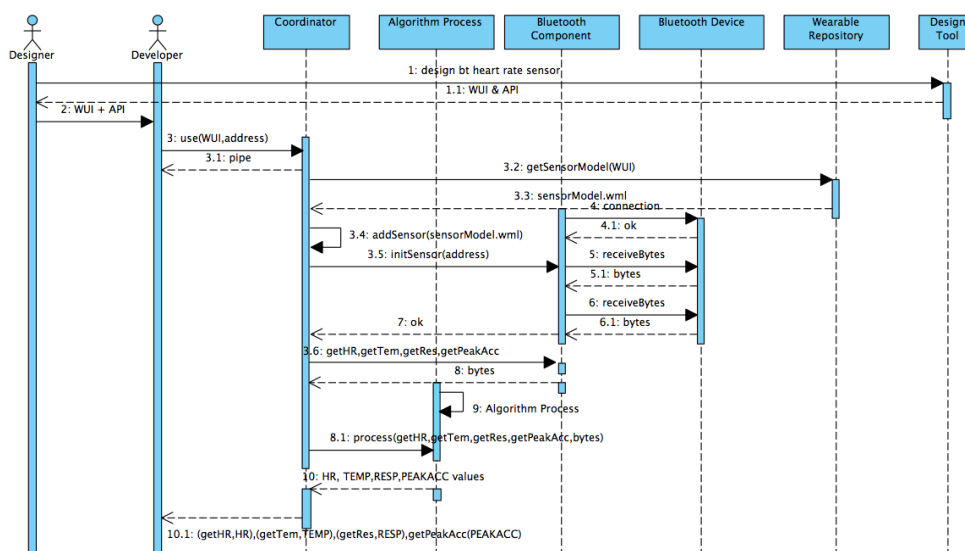


Figura 76- Integración de un wearable desde su diseño hasta su uso

El ejemplo propuesto es un wearable multifuncional (pulso cardíaco, temperatura, acelerómetro, etc.) basado en Bluetooth con la API específica, y definida a través de la herramienta de diseño (ver Figure 77):

- **getHR**: Obtener el pulso cardíaco
- **getResp**: Obtener el ritmo respiratorio
- **getTemp**: Obtener la temperatura corporal
- **getPeakAcc**: Obtener la aceleración lineal

Los diferentes pasos para usar el wearable son:

1. El diseñador o experto estudia e investiga para saber cómo funciona el wearable, usando el *datasheet* u otra información proporcionada por el proveedor.
2. El diseñador/experto, sabiendo cómo funciona el wearable (protocolo de comunicación, paquetes de datos, etc.), usa la herramienta de diseño (ver Figura 77) para definir el modelo del wearable de acuerdo a sus características y define su API (alto nivel). La figura 76 representa un diseñador/experto usando la herramienta de diseño para modelar el wearable a partir del *datasheet*, definiendo por ejemplo *getHR* como la etiqueta para representar la información relacionada con el pulso cardíaco dentro del *payload* o *getTemp* para representar la temperatura corporal.
3. Una vez el diseñador/experto finaliza con el proceso de diseño, la herramienta genera el modelo del sensor, usando el proceso de generación de modelos ya explicado. La figura 78 representa varios extractos del fichero WML con el modelo del wearable que generaría la herramienta de diseño.
4. Además de generar este modelo, la herramienta de diseño genera y hace públicos el WUI (Identificador universal del wearable), que en este caso corresponde con el código *JhDsX8851p*, y además libera la API definida por el diseñador (*getHR*, *getResp*, *getTemp*, *getPeakAcc*) para los desarrolladores. Estas etiquetas/identificadores de la API serán los usados por el coordinador para procesar correctamente la información.
5. El programador/desarrollador añade el coordinador a su proyecto, como un componente software (por ejemplo, en el caso de Android, se añade un fichero *.jar*).
6. El programador/desarrollador usa el coordinador junto con el WUI y la API del wearable hechas públicas para interactuar e integrar el wearable, como muestra la figura 74. El programador/desarrollador usa el WUI para obtener el fichero que contiene el modelo en WML (ver Figura 79) a través del coordinador. El coordinador usa el componente WR para obtener dicho modelo y gestiona este modelo para generar los distintos elementos software para poder usar el wearable. De estos elementos software el coordinador devuelve al programador/desarrollador un elemento llamado *zPipe*, que es el cauce que usará el coordinador para devolver la información procesada de este wearable al desarrollador/programador.

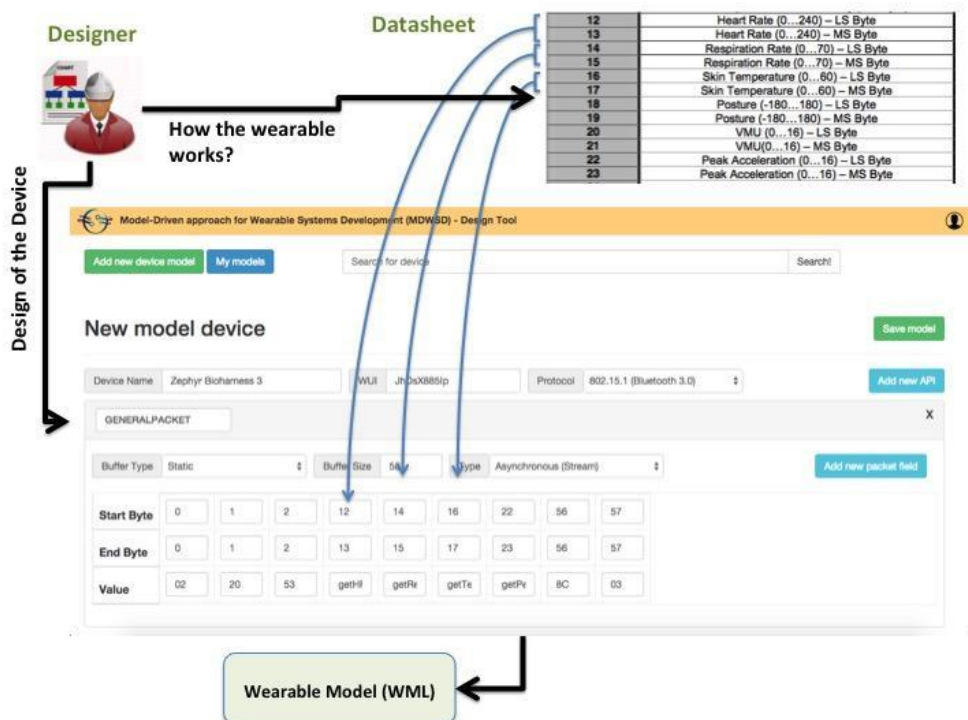


Figura 77 - Diseño de un wearable multifuncional

```

<?xml version="1.0" encoding="UTF-8"?>
<sensor>
  <id>JhdSx885Ip</id>
  <protocol>802.15.1</protocol>
  <name>Zephyr Bioharness 3</name>
  <api>
    <apisen>
      <id>GENERALPACKET</id>
      <type>0</type>
      <maxbufferize>58</maxbufferize>
      <buffertype>1</buffertype>
      <pivot>0x02</pivot>
      <payload>
        <start>3</start>
      </payload>
      <fields>
        <field>
          <id>idop</id>
          <value>0x20</value>
          <start>1</start>
          <offset>0</offset>
        </field>
        <field>
          <id>length</id>
          <value>0x53</value>
          <start>2</start>
          <offset>0</offset>
        </field>
      </fields>
    </apisen>
  </api>
</sensor>

<apival>
  <id>getHR</id>
  <start>12</start>
  <end>13</end>
  <datatype>0</datatype>
</apival>
<apival>
  <id>getResp</id>
  <start>14</start>
  <end>15</end>
  <datatype>0</datatype>
</apival>
<apival>
  <id>getTemp</id>
  <start>16</start>
  <end>17</end>
  <datatype>0</datatype>
</apival>
<apival>
  <id>getPeakAcc</id>
  <start>22</start>
  <end>23</end>
  <datatype>0</datatype>
</apival>
</apisen>
</api>

```

Figura 78 - Código del modelo correspondiente al wearable

```

zCom.init();
zCom.setContext(getApplicationContext());
String WUI ="JhDsX885Ip";
String MAC="00:22:D0:02:4C:49";
zPipe pipe = new zPipe(){
    public void handleMessage(Message msg) {
        super.handleMessage(msg);
        if(msg.getData().containsKey("getHR")){
            Object hr=msg.getData().get("getHR");
            //manage heart rate value
        }
        else if(msg.getData().containsKey("getTemp")){
            Object temp=msg.getData().get("getTemp");
            //manage skin temperature
        }
        else if(msg.getData().containsKey("getResp")){
            Object resp=msg.getData().get("getResp");
            //manage respiration rate
        }
        else if(msg.getData().containsKey("getPeakAcc")){
            Object peak=msg.getData().get("getPeakAcc");
            //manage linear acceleration
        }
    }
};
zCom.addDevice(WUI,MAC,pipe);

```

Figura 79 - Uso del wearable usando la plataforma Android

En la aplicación que use esta herramienta, una vez que el *wearable* haya sido detectado, conectado y sincronizado, el coordinador empieza a interactuar con el wearable a través de los componentes oportunos (dependiendo de su protocolo de comunicación) y procesa la información que recibe de acuerdo al procedimiento para procesar la información relevante (ver Figura 74).

Cuando el coordinador obtiene un paquete de información correcto, de acuerdo al método para detectar la fiabilidad del paquete y especificado en el modelo del wearable, lanza un evento en el correspondiente cauce (pipe) donde se envía la información al desarrollador a través de su *zPipe* de manera comprensible para el desarrollador/programador (por ejemplo, *getHr* devolvería 80 que simbolizan las pulsaciones por minuto o *getTemp* devolvería 37.4 que representa la temperatura en grados Celsius).

Esta información, ahora en posesión del desarrollador/programador, la gestiona como cree conveniente: mostrándola en pantalla, almacenando en la base de datos, almacenando en un servidor externo, procesarla o aplicarle algún otro tipo de proceso matemático, etc.

Capítulo 8

Aplicación de e-MoDe

8.1. Introducción

En este capítulo se presentan dos plataformas de telemonitorización desarrolladas siguiendo el framework expuesto en la presente tesis: (1) CloudRehab, plataforma de telemonitorización para la rehabilitación de pacientes con daño cerebral, y (2) CloudFit, plataforma de telemonitorización para la gestión de entrenamientos deportivos.

8.2. CloudRehab

8.2.1. Introducción

CloudRehab es una plataforma de telemonitorización para la telerehabilitación de pacientes con daño cerebral que pretende, mediante la supervisión remota de las actividades diarias de los pacientes y de sus constantes vitales (pulso cardíaco), ayudar al paciente en la recuperación de sus capacidades cognitivas, motoras y emocionales.

El principal objetivo de CloudRehab, como plataforma de telemonitorización en el área de telerehabilitación, es facilitar a los supervisores (doctores ocupacionales, terapeutas, psicólogos, etc.) la posibilidad de supervisar remotamente el proceso de rehabilitación del paciente, aumentando la independencia de éste y reduciendo así el número de desplazamientos.

CloudRehab está formada por una plataforma web, una aplicación móvil, un pulsómetro (*wearable*) y una serie de servicios *cloud* que son los que permiten ese intercambio de información entre pacientes (usuarios monitorizados) y supervisores.

A través de la plataforma web o la aplicación móvil los supervisores crean nuevas sesiones de rehabilitación y los pacientes, a través de la aplicación móvil, realizan dichas sesiones.

En cada una de estas sesiones de rehabilitación se genera distinta información: vídeo de la ejecución de la tarea, datos de pulso cardíaco, si el pulso cardíaco se elevó por encima de determinados umbrales (alertas), etc. Esta información se almacena, tanto en el propio dispositivo móvil, como en la nube, estando así disponible para los supervisores.

8.2.2. Desarrollo de CloudRehab

El desarrollo de CloudRehab se ha realizado siguiendo el framework propuesto (e-MoDe). La metodología, y su aplicación en CloudRehab, se explica a continuación etapa por etapa.

8.2.2.1. Especificación de Requisitos

En esta primera etapa se llevaron a cabo diferentes reuniones entre ingenieros de software, psicólogos y terapeutas de cara a obtener una primera vista y descripción de la plataforma.

Los terapeutas y psicólogos (expertos del área de la salud) ya habían definido el protocolo de telemonitorización, planteando un proceso de telerehabilitación para pacientes con daño cerebral basado en seis etapas:

1. **Rehabilitación con el paciente en el centro médico.** Como cualquier programa de rehabilitación, al comienzo de este nuevo protocolo el paciente debe realizar las sesiones de rehabilitación en el centro médico. El paciente realiza una serie de entrenamientos con el objetivo de poder realizar un ejercicio de rehabilitación. Este ejercicio de rehabilitación estará compuesto de una serie de pasos y deberá realizarse bajo la supervisión de un

profesional médico para garantizar que se realiza correctamente y el paciente aprende dicha ejecución, para a posteriori poder realizarlo sin supervisión.

2. **Grabación del ejercicio de rehabilitación.** Una vez que el paciente ha realizado toda la fase de rehabilitación y consigue realizar correctamente el ejercicio, el paciente lo realizará una vez más de una manera totalmente correcta en la ejecución. Esta ejecución será grabada en vídeo por el profesional de la salud y será usada a posteriori por el paciente para mejorar en la ejecución del mismo a lo largo del tiempo.
3. **Entrenamiento del paciente en casa.** En esta fase, el paciente realizará las distintas sesiones o ejercicios de rehabilitación desde casa. Haciendo uso del vídeo del paso anterior, el paciente realizará el ejercicio con una frecuencia definida por el supervisor (diaria, semanal, etc.) y se grabará a sí mismo realizando dicho ejercicio mientras visualiza el vídeo que muestra una correcta ejecución. De esta manera, el paciente podrá verse a sí mismo en tiempo real cómo está realizando el ejercicio y cómo debe realizarse. Esto ayudará al paciente cada vez a realizarlo mejor sin ningún tipo de supervisión.
4. **Monitorización del pulso cardíaco.** Se controlará en el paciente su pulso cardíaco mientras este está realizando el ejercicio. Esta medida de control se establece ya que si el paciente se pone nervioso o sufre alguna alteración que haga elevar su nivel de estrés mientras realiza el ejercicio, esta alteración queda reflejada en la variación del pulso cardíaco. El control de la variación del pulso cardíaco junto con una serie de opciones adicionales que permitan al paciente reducir su nivel de estrés ayudarán al paciente a realizar correctamente los ejercicios de rehabilitación.
5. **Programación de sesiones de rehabilitación.** Los profesionales de la salud deben poder generar o crear distintas sesiones de rehabilitación que a posteriori serán llevadas a cabo por los pacientes sin ningún tipo de supervisión y en casa. Esto permitiría realizar más sesiones de las que se realizan de una manera tradicional, lo que haría al paciente evolucionar más rápido.
6. **Recolección de datos del proceso de telerehabilitación y comunicación entre supervisores y pacientes.** Todos los datos procesados durante todo el proceso de rehabilitación serán revisados por el profesional (supervisor), quien evaluará los mismos para determinar la evolución del paciente en su rehabilitación. De esta manera, el profesional puede detectar errores en la ejecución de los ejercicios que en un modelo de rehabilitación tradicional podrían no detectarse, pudiendo así corregir al paciente y ayudarlo o motivarlo en su proceso de rehabilitación.

Este protocolo de telemonitorización fue definido por diferentes expertos del área de la salud (psicólogos, terapeutas, doctores) de acuerdo a su experiencia previa en procesos de telerehabilitación.

Con este protocolo de telemonitorización, y con varias reuniones se terminó de definir el escenario general que se pretendía conseguir a través de la plataforma de telemonitorización.

Este escenario, representado en la Figura 80, representa dicho protocolo, pero incorporando elementos fundamentales que se añadieron como sugerencia en las diferentes reuniones y que han sido expuestos en esta tesis, cómo puede ser la computación en la nube, los dispositivos móviles como smartphone y otros como sensores de pulso cardíaco (*wearables*).

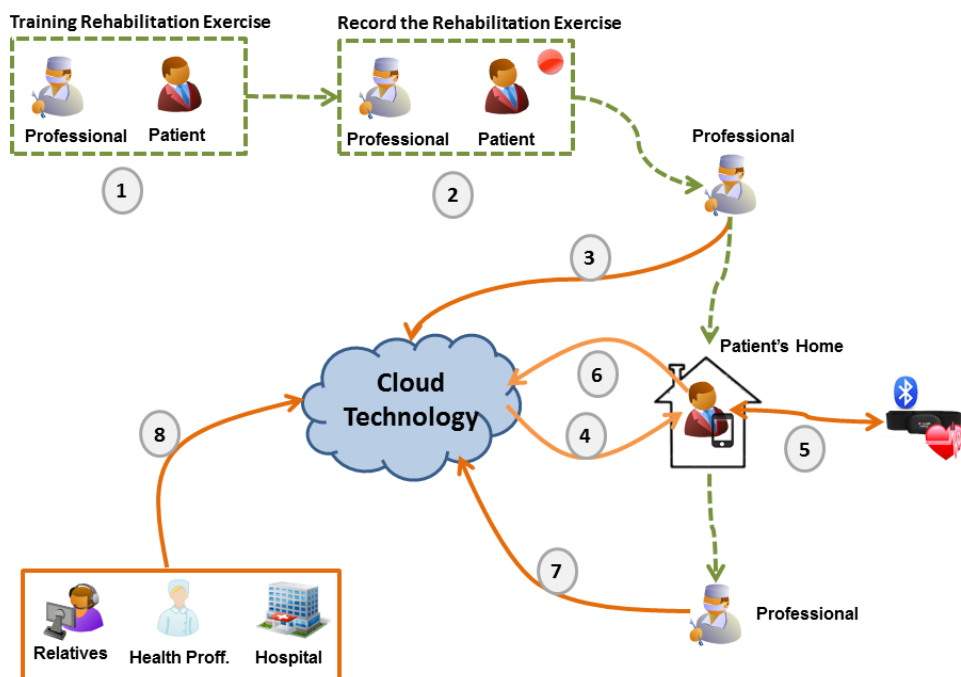


Figura 80 - Protocolo de telemonitorización de CloudRehab

Las líneas verdes representan las distintas fases del proceso de rehabilitación y las líneas naranjas los diferentes pasos del protocolo de telemonitorización donde se usa la plataforma de telemonitorización (CloudRehab).

Una vez especificada la vista general de la plataforma mostrada en la Figura 79, y partiendo del protocolo de telemonitorización, se realizaron los dos análisis definidos en esta etapa: *registro y análisis de la actividad e interacción y usuario*.

Para ilustrar la aplicación de estas dos fases, nos centraremos en la tarea principal: *el paciente, tomando el smartphone y la banda de pulso cardíaco, realiza cierta actividad programada por el terapeuta a través de una aplicación móvil, mientras se graba con el propio smartphone y mientras este recoge valores de pulso cardíaco del pulsómetro*.

Registro y análisis de la actividad

En esta fase de análisis se identificaron las distintas funcionalidades de la plataforma, partiendo de las tareas identificadas, así como otras funcionalidades de la plataforma necesarias para dar soporte a esas tareas (p.ej. gestión de vídeos) y restricciones sobre dichas funcionalidades (requisitos no funcionales, como tamaño máximo de los vídeos o calidad de los mismos, para agilizar la subida y descarga de información).

De esta fase se definieron numerosos requisitos funcionales, entre los que destacan: reproducción y grabación de vídeo con un smartphone, registro y captura de pulso cardíaco de una banda, gestión de diferente información en la nube, recuperar tareas, unir documentos de vídeo, convertir documentos de vídeo, gestión de información en el smartphone previa al almacenamiento en la nube, etc.

Como restricciones a las funcionalidades, se recopilaron algunos requisitos no funcionales entre los que destacan: calidad del vídeo que recoge las ejecuciones de las tareas, fiabilidad de los datos del sensor de pulso cardíaco, etc.

Especificación de interacciones y necesidades especiales del usuario

Debido a que CloudRehab es una plataforma de telerehabilitación para pacientes con daño cerebral, la interacción entre los diferentes usuarios y las necesidades especiales de los mismos se hace especialmente relevante.

En esta fase, los supervisores y expertos en el área determinaron algunas de las necesidades especiales de los pacientes, como pueden ser: botones grandes con iconos y texto, interfaces limpias y simples, que el sensor de pulso cardíaco sea ergonómico y cómodo (de ahí que se optase por una banda), etc.

Además, para dar soporte a una correcta telemonitorización los supervisores indicaron que era necesaria una interacción entre supervisor y paciente bidireccional, esto es, que el paciente puede ponerse en contacto con el supervisor cuando desee y viceversa.

Dentro de esta interacción, los supervisores determinaron que eran necesarios cinco tipos de interacciones: mensajes simples, mensajes con audio, mensajes con video, mensajes con imágenes y mensajes con otro tipo de documentos.

La figura 81 muestra la etapa de especificación de requisitos a través del análisis de las dos fases, partiendo del protocolo.

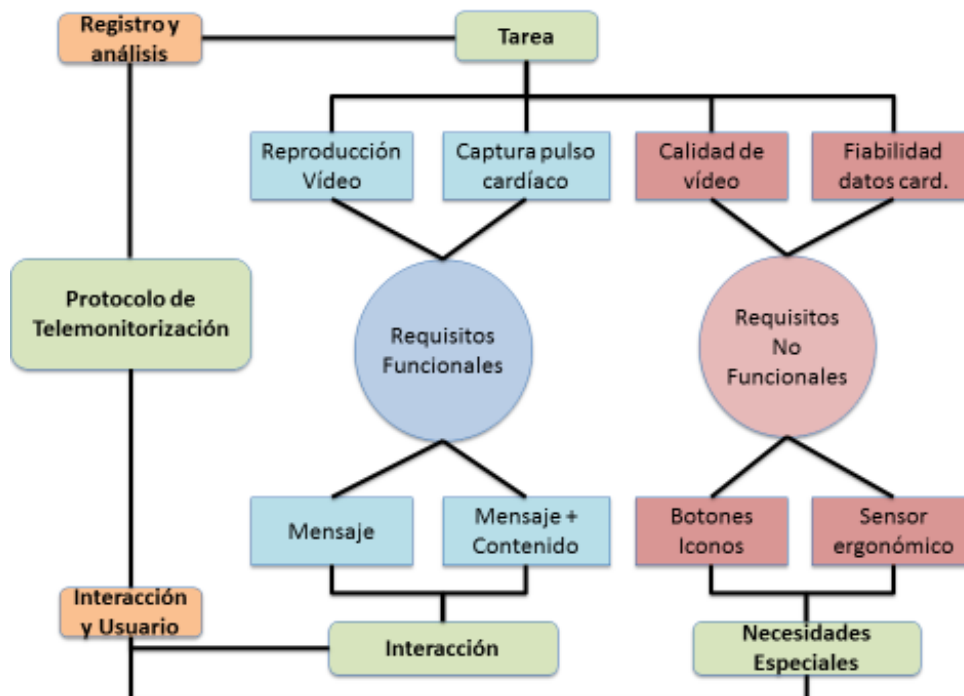


Figura 81 - Especificación (parcial) de requisitos de CloudRehab

8.2.2.2. Diseño

Con los requisitos funcionales y no funcionales de la etapa anterior los ingenieros abordan el diseño de la plataforma. Para esto, los ingenieros centran dicha etapa de diseño en las cuatro áreas de interés propuestas: arquitectura, colaboración, gestión de tareas y necesidades del usuario.

a) *Arquitectura*

Dentro del diseño, en este aspecto los ingenieros, de acuerdo a requisitos no funcionales y recursos disponibles determinan la tecnología a usar. Partiendo de la arquitectura propuesta en el capítulo 6, donde están identificados los diferentes elementos, se diseñan los diferentes servicios y submodulos software que representan todas las funcionalidades plasmadas en los requisitos funcionales y que en la siguiente etapa (implementación) los programadores implementarán.

Tecnología

Con respecto al ámbito tecnológico, CloudRehab está soportada por G, una plataforma *cloud* perteneciente a la compañía Indra/GNúbila. Se decidió usar dicha tecnología por ser una tecnología válida para los requisitos especificados (adaptable a la arquitectura híbrida, escalabilidad, deslocalización, etc.) y como apuesta por tecnología nacional, al ser la única tecnología nacional del Magic Quadrant [Gnubila 2015].

Esta tecnología permite una arquitectura software híbrida como la propuesta en el framework. G, mediante verbos propios (*critinsert, modify*), permite interactuar con los recursos o datos del sistema sin necesidad de programar directamente un servicio siempre y cuando el servicio está orientado hacia el tratamiento de información con la base de datos. Cualquier otro servicio que requiera de un procesamiento más complejo tipo si debe implementarse (en G se llaman *templates* y corresponden con un XML-RPC tradicional).

G usa un modelo de datos no relacional (orientado a grafos), que, aunque no es obligatorio en sistemas de este tipo, si es un valor añadido al garantizar la adición de nuevos atributos a nodos ya definidos en tiempo real sin rediseñar el esquema de datos ni realizar modificaciones en la implementación.

Como otros elementos tecnológicos, se optó por smartphone con sistema operativo Android y una banda de pulso cardíaco Polar WearLink Bluetooth. Esta decisión se tomó principalmente por el reducido coste de ambos dispositivos y la ergonomía del sensor.

Distribución de funcionalidades (*Back-end* y dispositivos móviles)

Partiendo de los requisitos funcionales, en esta etapa de diseño los ingenieros diferencian entre las funcionalidades de uso común, y que corresponden al *back-end* como servicios en la nube (*back-end*) y los que quedarán dentro del propio dispositivo móvil. De acuerdo a la arquitectura software híbrida (SOA-ROA) propuesta y explicada en el capítulo 6, todas aquellas funcionalidades que estén relacionadas con la gestión de información y en este caso, con la gestión de videos y contenido multimedia, se implementarán como servicios siguiendo la tecnología G.

Puesto que, en G, a través de su plataforma GPaaS, no es necesario definir servicios propios para la gestión de información, muchas de estas funcionalidades quedan cubiertas de facto por la propia tecnología. Si es necesario conocer la tecnología ya que cuenta con verbos propios como *critinsert* o *modify* para insertar o modificar un nuevo elemento, respectivamente.

Por lo tanto, se tuvo que realizar un estudio de la tecnología para especificar como iba a ser el consumo de estos servicios (API) para dar soporte a los requisitos funcionales (funcionalidades).

Algunos ejemplos de estos servicios, por ejemplo, para las funcionalidades de recuperar las sesiones de un paciente y obtener los valores cardíacos de una sesión, son (ver Figura 81):

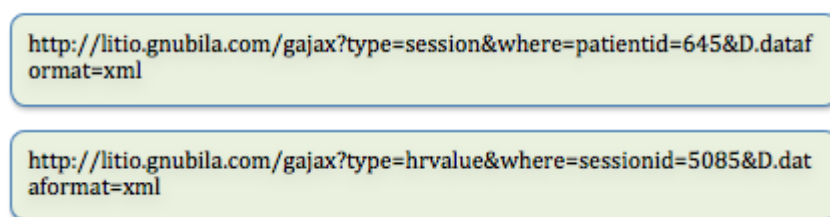


Figura 82 - Ejemplos de servicios en G

A través del uso de la propia tecnología se consumen servicios que soportan funcionalidades tales como:

- Dar de alta nuevos pacientes o profesionales.
- Crear, modificar y recuperar sesiones de rehabilitación.
- Intercambio de mensajes entre profesionales y pacientes (bidireccional).
- Crear, modificar y gestionar alertas (eventos que notifican al paciente a una determinada hora).
- Gestión de parámetros fisiológicos (añadir o recuperar dependiendo de parámetros de tiempo).
- Gestión de videoteca (vídeos de utilidad para el paciente)

Además de estos servicios, que se usan directamente, se han implementado funcionalidades para la gestión de vídeos en servicios independientes. Estos servicios se encargan de unir vídeos, rotarlos o convertirlos para hacer posible que sean visibles en cualquier dispositivo (usando en este caso *codecs* genéricos y contenedor de video WebM). Toda esta funcionalidad se ha derivado a la nube para evitar toda esa carga computacional en el dispositivo móvil, lo que implicaría un gasto de batería y tiempo adicional [Satyanarayanan 2001].

El resto de funcionalidades, que son intrínsecas a la acción de monitorización, como puede ser registrar la sesión de telerehabilitación (grabación de vídeo, reproducción de vídeo o audio, etc.), el pulso cardíaco o las distintas acciones del usuario se implementa directamente en la aplicación móvil.

Por lo tanto, de este aspecto de la etapa de diseño, se genera la API para el acceso de las funcionalidades del *back-end* (en este caso de acuerdo a la nomenclatura de G) y la especificación de las diferentes funcionalidades que se implementarán en la aplicación móvil.

b) Modelo Colaborativo

De acuerdo a la propuesta presentada como modelo colaborativo para plataformas de telemonitorización (capítulo 6), para esta solución en concreto (CloudRehab), se usa una instancia del modelo propuesto.

En CloudRehab únicamente hay dos tipos de usuarios: doctores/terapeutas/psicólogos (todos con las mismas responsabilidades), que ejercen de supervisores, y los pacientes con daño cerebral, que son los usuarios monitorizados.

Del modelo original se realiza una instancia del mismo añadiendo cinco tipos de interacciones, para dar soporte al envío de mensajes sin contenido con contenido (audio, video, imagen, documento) (ver Figura 83).

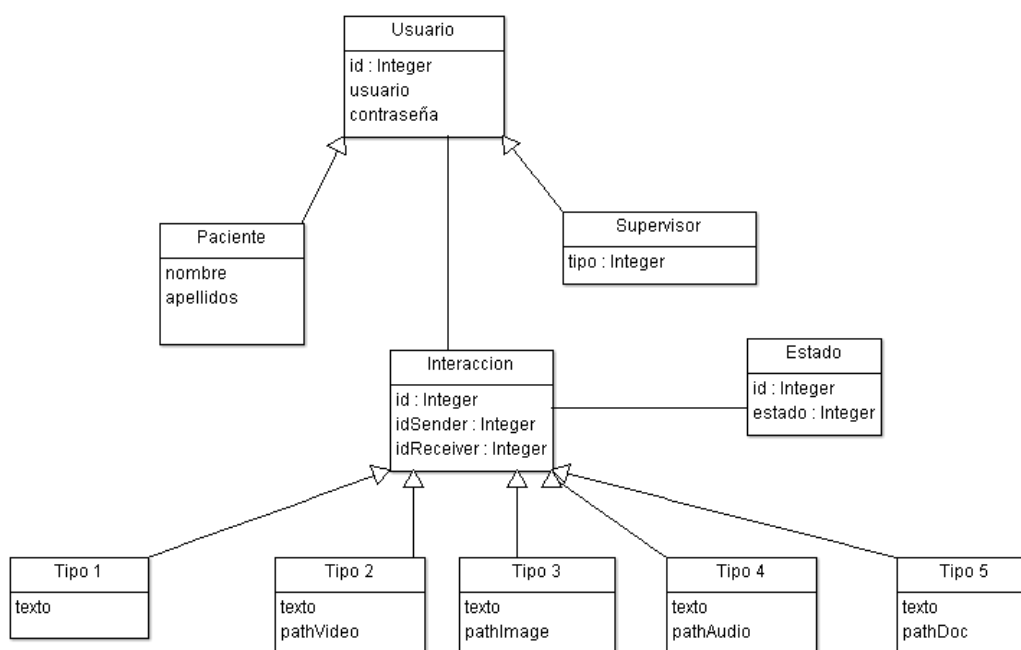


Figura 83 - Modelo colaborativo en CloudRehab

El modelo representado en la figura 83, y que parte del modelo original, aunque añade nuevos elementos para dar soporte al tipo de interacción de CloudRehab, identifica los usuarios del modelo original con atributos concretos (nombre, apellidos, usuario, contraseña, etc.).

La otra parte del modelo especifica los distintos tipos de interacciones, cinco tipos en ese caso como se comentó en la etapa de especificación de requisitos: envío de mensajes simple, con audio, con texto, con imagen y con otro tipo de documento.

Además de especificar estas interacciones, se mantiene la gestión de interacciones a través de su estado y de su emisor y receptor, para poder enviar entre usuario monitorizado y supervisor y viceversa gestionando cuando se ha enviado y recibido, ya que por los requisitos es importante que tanto el supervisor como el paciente sean conscientes de que su mensaje ha llegado correctamente.

c) **Gestión de tareas de telemonitorización**

La aplicación móvil, que es la que permite la realización de las sesiones de rehabilitación, es la encargada de mostrar las distintas tareas de dichas sesiones, de acuerdo al modelo propuesto en el capítulo 6.

Siguiendo este modelo se han diseñado las diferentes tareas de telemonitorización, diseñando cada tarea como un conjunto de datos que posteriormente se materializaron con una notación concreta, en este caso XML puesto que era la notación usada por la tecnología G, y dejando la lógica de control sobre las tareas y la implementación de la vista como parte de las funcionalidades de la aplicación móvil, siguiendo el patrón MVC planteado.

En CloudRehab la tarea principal, como ya se ha comentado, es: *el paciente, tomando el smartphone y la banda de pulso cardíaco, realiza cierta actividad programada por el terapeuta a través de una aplicación móvil, mientras se graba con el propio smartphone y mientras este recoge valores de pulso cardíaco del pulsómetro.*

Aunque esta tarea sea la inicialmente especificada en los requisitos, por el propio protocolo e interés de los supervisores tiene más información adicional de cara a ser más útil para su evaluación, como, por ejemplo: puntuación del paciente sobre la realización de la tarea una vez realizada, si visualizó el vídeo de ayuda, etc.

Partiendo de esta tarea, se diseñó la misma siguiendo el modelo para la gestión de tareas, identificando los diferentes elementos:

- Vídeo de ayuda → El vídeo que usará el paciente como ayuda para realizar las sesiones de telerehabilitación.
- Fecha → Fecha de la tarea.
- Hora de inicio → Cuando comienza la tarea.
- Hora de fin → Cuando finaliza la tarea.
- Identificador → Número para identificar a la tarea.
- Puntuación → Puntuación de 1 a 5 que realiza el paciente sobre cómo le ha ido la ejecución de la tarea.
- Frecuencia cardíaca mínima → Umbral inferior permitido. En caso de superarse el paciente recibe ayuda de la aplicación.
- Frecuencia cardíaca máxima → Umbral superior permitido. En caso de superarse el paciente recibe ayuda de la aplicación.
- Audio de finalización → El paciente puede grabar un audio una vez finalice la sesión para manifestar una opinión.

Con esta especificación de la tarea, esto es, los distintos elementos que conforman la tarea, siguiendo el modelo de la propuesta se crea un modelo para esta tarea en concreto, como muestra la Figura 84.

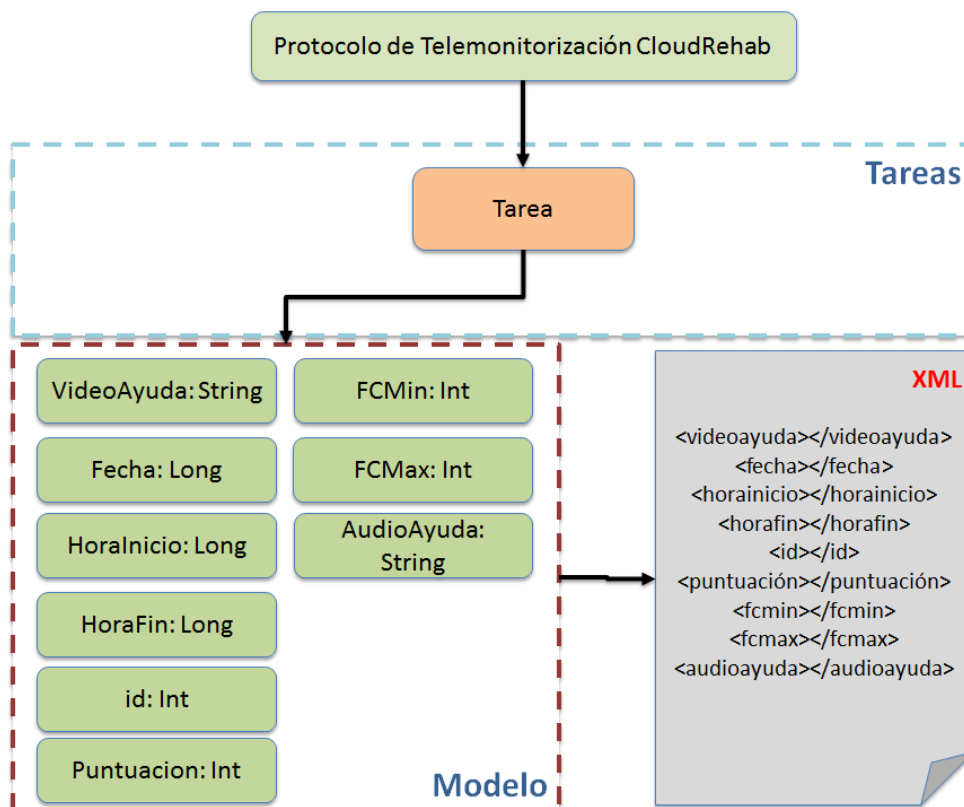


Figura 84 - Modelo para la tarea en CloudRehab

Este modelo es diseñado por los ingenieros a partir de la especificación de la tarea en los requisitos funcionales y en el propio protocolo. Este modelo se usa para, a través de una notación como es XML, representar dicha tarea.

Esta tarea, ya representada en XML, es consumida por las diferentes aplicaciones, y siguiendo el patrón MVC propuesto, se usará para representar las vistas en las aplicaciones y la lógica de control que se gestionará en los dispositivos móviles.

Como se ha comentado, se usará Android como sistema operativo de la aplicación móvil. En Android la interfaz de esta tarea estará representada en XML (Vista), la lógica de control (Controlador) se implementa directamente en la aplicación y el modelo sería el documento XML con la propia tarea.

Por lo tanto, la aplicación consume un servicio que devuelve una tarea concreta siguiendo el modelo de la figura 82 pero con un estado concreto, el controlador se encargaría de tomar dicho modelo y la interfaz implementada en Android, y completar dicha interfaz con los datos del modelo para mostrar la tarea correspondiente. Después, a través de la implementación adecuada se interactúa con el software para la realización de la tarea.

d) Necesidades del usuario

Los pacientes con daño cerebral suelen tener problemas cognitivos, por lo que el uso de una aplicación móvil para ellos representa un reto. Por esto, y de acuerdo a los requisitos no funcionales determinados en la etapa de especificación de requisitos, se prestó especial hincapié en la usabilidad de las aplicaciones móviles.

Por esto, se diseñaron las interfaces de usuario de acuerdo a las líneas de los expertos y supervisores, quienes previamente habían determinado que tipo de interfaces son más adecuadas para este tipo de usuarios: botones grandes, texto color negro y claro, interfaces lo más sencillas posibles, sin muchas funcionalidades en pantalla, etc.

De este aspecto de diseño, se generaron diferentes interfaces de usuario que posteriormente se implementaron en los dispositivos móviles para representar adecuadamente tanto tareas como la propia interfaz de la aplicación.

8.2.2.3. Implementación

a) *Implementación de funcionalidades*

La implementación de las distintas funcionalidades la realizan los diferentes programadores dependiendo del ámbito de la funcionalidad: *back-end* o en los dispositivos móviles.

Las funcionalidades del *back-end* (servicios web) se implementaron (en su mayoría) a través de la propia tecnología G, por lo que no fue necesario implementar líneas de código para todas aquellas funcionalidades que sólo requerían interactuar con la base de datos de la plataforma (guardar, actualizar, crear y eliminar).

Los únicos servicios implementados explícitamente fueron:

1. Para la edición de vídeos: unir diferentes documentos de vídeo y cambiar el formato del mismo para hacerlo compatible con cualquier dispositivo, ya que el formato registrado por el dispositivo móvil no era compatible con la plataforma web.
2. Para enviar los diferentes tipos de notificaciones: dar soporte a la interacción entre los usuarios, esto es, permitir enviar desde un emisor a un receptor un tipo de notificación.

La implementación de las distintas funcionalidades para la aplicación móvil parte del uso de la plataforma Zappa, explicada en el capítulo 7, sección 1 y como herramienta de soporte para el framework propuesto.

Zappa proporciona funcionalidades implementadas, para que desarrolladores a través de la API de los distintos componentes usen las funcionalidades, sin necesidad de implementar estas funcionalidades.

Aunque la plataforma está compuesta por numerosos componentes con diferentes funcionalidades, no todos los componentes se usan en CloudRehab, usándose las siguientes funcionalidades de Zappa:

- Intercambio de datos con la nube, a través del componente gAndroid (especialmente implementado para este proyecto)
- Gestión de la cámara: iniciar, parar, grabación (tanto de la cámara frontal como trasera)
- Gestión de vídeos: reproducción, interacción (con los distintos formatos soportados por Android).
- Gestión de datos internos: Gestión de la estructura de directorios interna de Android (tarjeta SD) para guardar, recuperar, eliminar, mover documentos.

- Transformación de modelos XML a objetos: Usado para transformar las tareas en objetos software (en Java), para así poder gestionar la tarea.

b) Integración de wearables

Para este proyecto se usó un único dispositivo, una banda de pulso cardíaco (Polar WearLink Bluetooth). Usando la herramienta presentada, WearIt, se modeló dicho dispositivo para su integración en la plataforma y su interacción con las aplicaciones móviles. La figura 85 muestra parte del código del modelo de dicho dispositivo y la figura 86 el código (Android) de uso del coordinador que interactúa con el *wearable*.

```

<sensor>
  <id>aoFdcj02z20</id>
  <protocol>002.15</protocol>
  <name>Polar iWL</name>
  <api>
    <apisen>
      <id>getinfo</id>
      <type>0</type>
      <maxbuffersize>16</maxbuffersize>
      <buffertype>1</buffertype>
      <pivot>0xFE</pivot>
      <payload>
        <start>5</start>
      </payload>
      <fields>
        <field>
          <id>seq</id>
          <value>0x06</value>
          <start>0</start>
          <offset>3</offset>
        </field>
      </fields>
    </apisen>
  </api>
</sensor>

</check>
<apival>
  <id>getHR</id>
  <start>5</start>
  <end>5</end>
  <datatype>0</datatype>
</apival>
<apival>
  <id>getRR</id>
  <start>6</start>
  <end>7</end>
  <datatype>0</datatype>
</apival>
</apisen>
</api>

```

Figura 85 - Modelo del *wearable* usado en CloudRehab

```

zCom.init();
zCom.setContext(getApplicationContext());
String WUI = "aoFdcj02z20";
String MAC = "00:22:D0:02:4C:49";
zPipe pipe = handleMessage(msg) -> {
    super.handleMessage(msg);
    if(msg.getData().containsKey("getHR")){
        Object hr=msg.getData().get("getHR");
        //manageHeartRateValue(hr);
    }
    else if(msg.getData().containsKey("getRR")){
        Object rr=msg.getData().get("getRR");
        //manageRRInterval(rr);
    }
};
zCom.addDevice(WUI,MAC,pipe);

```

Figura 86 – Interacción con el *wearable* a través del coordinador

c) Aplicaciones

CloudRehab consta de dos aplicaciones, una aplicación móvil y una plataforma web. No obstante, la tendencia que se está siguiendo es que sea una plataforma de telemonitorización soportada únicamente por computación móvil (smartphone/Tablet).

Esto hará que en un futuro esta solución no disponga de plataforma web y todas las funcionalidades deriven al móvil, pero con distintos perfiles (Supervisor/experto y paciente), de ahí que en la versión actual la vista de supervisor en el dispositivo móvil tenga más funcionalidades que la plataforma web.

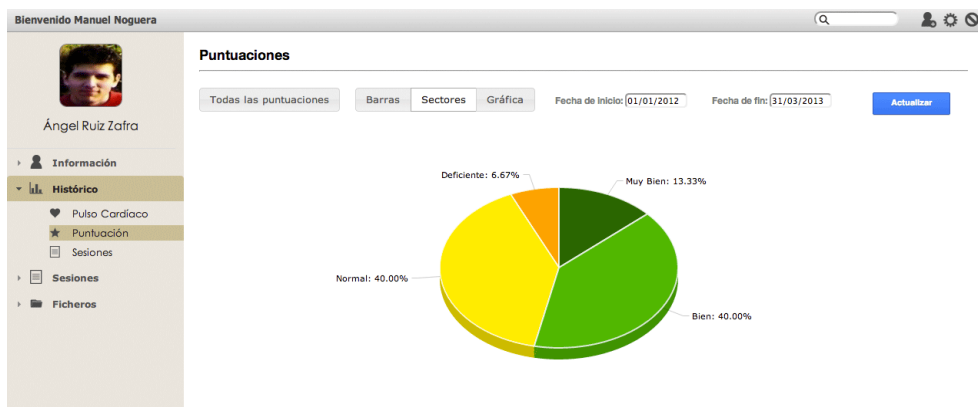
Plataforma Web

La plataforma web está diseñada para permitir a los profesionales de la salud gestionar toda la información:

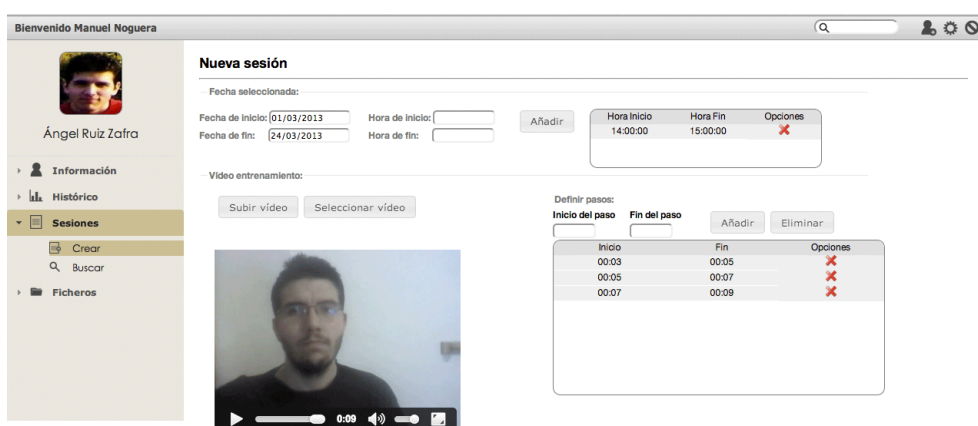
- **Gestión de la información del paciente:** Los profesionales médicos pueden dar de alta en el sistema sus propios pacientes. Definiendo sus datos personales y diversa información médica relevante para las sesiones de rehabilitación.
- **Gestionar las sesiones del paciente:** Una sesión está compuesta por un vídeo de entrenamiento grabado en una sesión de entrenamiento, un conjunto de audios e imágenes usadas como apoyo para reducir el nivel de estrés del paciente, el vídeo resultante de la grabación generada, los valores de pulso cardíaco y las alertas declaradas durante la sesión, entre otros. Los profesionales pueden definir nuevas sesiones con nuevas actividades de entrenamiento, y una vez que el paciente haya realizado la sesión, revisar toda la información para evaluar su progreso.
- **Revisar la información generada por el paciente,** como por ejemplo las sesiones finalizadas, valores cardíacos entre un rango de fechas, gráficos estadísticos que muestran la evolución del paciente a lo largo del tiempo, etc.
- **Monitorización en tiempo real de las sesiones:** Los profesionales médicos y los familiares del paciente pueden ver los valores cardíacos y la grabación del vídeo en tiempo real.

La Figura 87 muestra algunas capturas de la plataforma web.





Cloud rehab



Cloud rehab

Figura 87 - Plataforma web de CloudRehab

Aplicación Móvil

La aplicación móvil está desarrollada para Android (versión 2.2 y superiores) y será la herramienta usada por pacientes para llevar a cabo sus tareas de rehabilitación y también revisar toda la información por parte de los expertos o supervisores.

Esta aplicación, dependiendo del usuario que acceda a la misma tiene dos vistas: vista paciente y vista supervisor.

Vista Supervisor

Cuando un terapeuta accede a la aplicación, usando su usuario y contraseña, se muestra una vista con una serie de funcionalidades (muchas de ellas soportadas por la plataforma web) para garantizar tanto la creación de nuevas sesiones de rehabilitación como otro tipo de contenido.

Esta vista permite las siguientes funcionalidades:

- Dar de alta nuevos pacientes o modificar pacientes ya existentes.
- Intercambio de mensajes con pacientes.
- Crear o modificar alertas: notificaciones que le aparecerán a los pacientes a una hora determinada y con cierto contenido concreto (texto, video, audio, imagen u otro tipo de fichero).

- Crear nuevas sesiones de rehabilitación.
- Gestionar vídeos personales: crear nuevos vídeos que sirvan de ayuda extra al paciente en su proceso de rehabilitación.

La Figura 88 muestra algunas capturas de pantalla de la vista del profesional o supervisor.

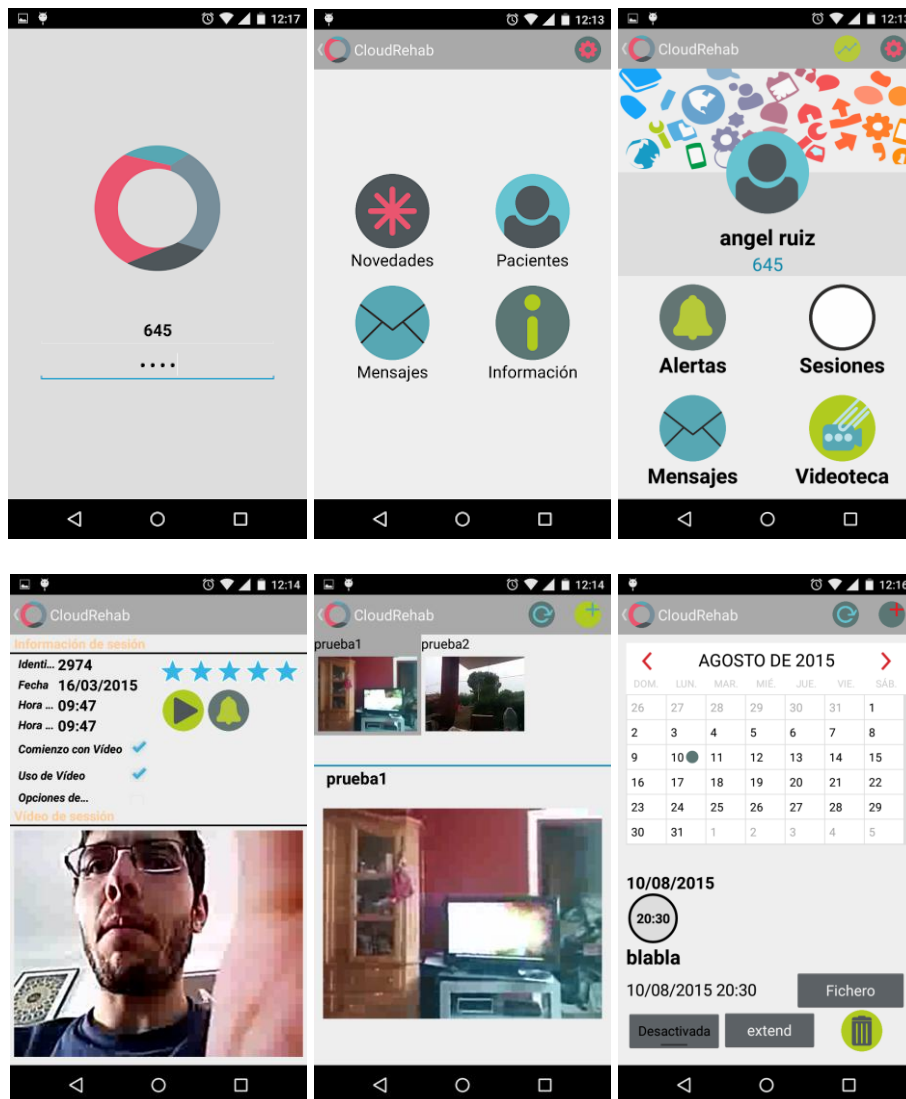


Figura 88 – Aplicación móvil CloudRehab (Vista Supervisor)

Vista Paciente

Permite a los pacientes grabar la realización del ejercicio diseñado por el profesional y monitorizar su pulso cardíaco usando una banda pulsómetro para controlar el nivel de estrés.

La aplicación divide la pantalla del dispositivo en dos partes. La parte derecha muestra la cámara frontal, mientras que la mitad izquierda muestra el vídeo de entrenamiento grabado previamente en las sesiones de entrenamiento entre el paciente y el profesional.

Esto provee una retroalimentación en tiempo real muy útil al paciente sobre cómo realizar correctamente el ejercicio y como lo está realizando. Mejorando así su ejecución y evolucionando en su proceso de rehabilitación.

Mientras el paciente realiza el ejercicio de rehabilitación, si la aplicación detecta que el pulso cardíaco del paciente ha alcanzado un nivel límite (definido por los profesionales en la aplicación web o vista de profesional), la aplicación reproduce un sonido informativo y muestra un diálogo emergente con diferentes opciones de relajación (revisar el vídeo de entrenamiento paso a paso, reproducir un audio personalizado de ayuda o visionar imágenes y audios de relajación) , con el objetivo de reducir su nivel de estrés.

La figura 89 muestra diferentes capturas de esta vista.

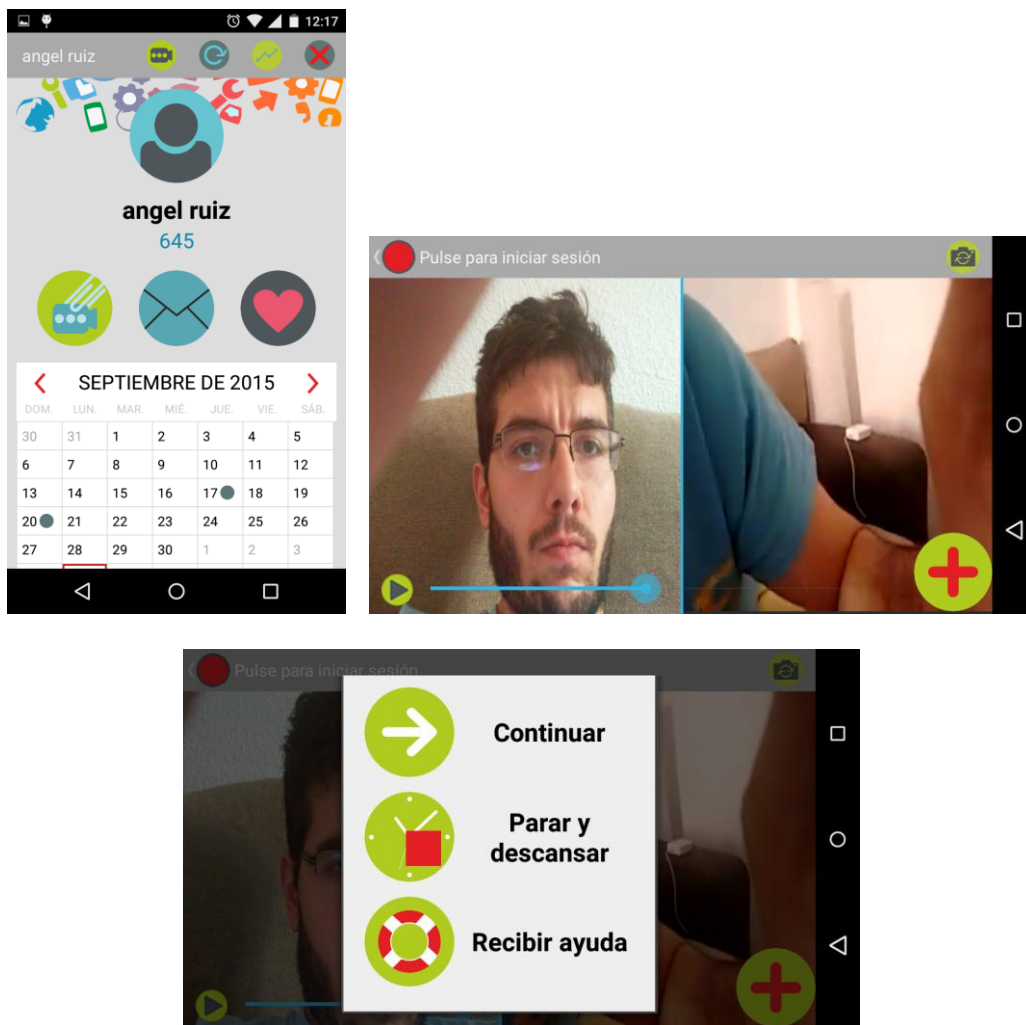


Figura 89 – Aplicación móvil de CloudRehab (Vista Paciente)

8.2.2.4. Despliegue

El despliegue de la aplicación se ha realizado bajo el IaaS ofrecido por GNúbila/Indra. Gracias a GPaaS, la plataforma de despliegue de aplicaciones en la nube basada en la tecnología G en la que está basada CloudRehab, la fase de despliegue se ha realizado como en cualquier otra aplicación y sin ningún interés adicional relacionado con el framework aquí propuesto.

Además, normalmente en la fase de despliegue y antes de usar la misma de manera convencional, se suele realizar ya en producción una fase de testeo beta por parte de los distintos usuarios.

En este caso se realizó con dos psicólogos y un único paciente, para detectar y corregir errores, así como completar la funcionalidad que faltase de la aplicación, pero sin realizar un proceso de testeo concreto o verificado, sencillamente basándose en el uso de la misma y *feedback* de usuarios hacia desarrolladores/ingenieros.

8.2.2.5. Validación

El proceso de validación, como plataforma de telemonitorización (telerehabilitación) para pacientes con daño cerebral, se está llevando a cabo por psicólogos y terapeutas en un hospital público.

Este proceso de validación se ha realizado planteando un estudio con pacientes reales que han sufrido algún tipo de daño cerebral adquirido, donde la validación ha consistido en dos etapas distintas.

La primera etapa ha sido una fase de rehabilitación sin usar CloudRehab, esto es, realizando un proceso de monitorización tradicional:

1. El paciente se desplaza al centro médico.
2. El terapeuta le indica lo que tiene que hacer.
3. El paciente vuelve a su casa y realiza dichos ejercicios durante periodo determinado.
4. Finalmente, el paciente vuelve al centro médico y realiza la misma ejecución para que el terapeuta determine la evolución con respecto a la última vez que estuvo.

La segunda etapa ha consistido en realizar un proceso de telemonitorización usando CloudRehab y la metodología expuesta en la figura 79.

No obstante, debido a que este proyecto lleva en activo desde el 2012, a día de hoy el proceso de validación se ha realizado en dos marcos temporales distintos, uno en el periodo de Enero-Marzo de 2013 como pruebas piloto de la plataforma, y un segundo periodo que comenzó en Marzo de 2013 como un estudio completo.

El resultado de esta prueba piloto puede consultarse en [Ruiz-Zafra 2013] y [Urbano 2014].

8.3. CloudFit

8.3.1. Introducción

CloudFit es una plataforma de telemonitorización de salud y bienestar que pretende ayudar a distintos tipos de usuarios mediante la monitorización remota de sus actividades diarias, así como la recopilación de datos fisiológicos (pulso cardíaco, temperatura), inerciales (movimiento) y contextuales (geoposición, temperatura ambiente) mediante el uso de *wearables*.

Los usuarios, a través de una aplicación móvil, realizan diversas actividades propuestas por los supervisores, y estos, a través de esta plataforma web o aplicación móvil, pueden acceder a toda la información generada por los usuarios.

El principal objetivo de CloudFit es facilitar a los profesionales (supervisores) de diversos ámbitos (doctores, nutricionistas, entrenadores, etc.) la capacidad de supervisar remotamente alguna actividad o evento de un usuario monitorizado.

Los profesionales usan Cloudfit para monitorizar las actividades de estos, aumentando su independencia, reducir así el número de desplazamientos y mejorar su evolución. Para conseguir este objetivo, los profesionales diseñan tareas de telemonitorización que los usuarios monitorizados realizan y posteriormente se evalúan.

CloudFit ha sido diseñada, siguiendo el framework propuesto, para ser una plataforma multidisciplinar que permita dar soporte a cualquier tipo de área relacionada con la salud y el bienestar, desde sesiones de entrenamiento para deportistas de élite, a control nutricional o revisiones periódicas médicas.

En la versión actual de la plataforma se da soporte a entrenadores (supervisores/expertos) y atletas (usuarios monitorizados). Estos supervisores, a través de CloudFit, generan entrenamientos que a posteriori el atleta realiza a través de su dispositivo móvil.

8.3.2. Aproximación clásica

La monitorización de cada área del ámbito de la salud y bienestar tiene unos parámetros y características distintos, ya que cada una tiene unos objetivos que pueden ir desde el diagnóstico de acuerdo de la información fisiológica hasta determinar un parámetro sobre cómo vive una persona normal, los alimentos que puede tomar o cómo se comporta en determinadas situaciones.

En el caso de la primera aproximación de la plataforma, supervisión de atletas (usuarios monitorizados) por parte de entrenadores (supervisores), los supervisores en un escenario de monitorización tradicional suelen planificar, sin uso de las TICs (calendarios, papel de especificaciones), un calendario con los diferentes entrenamientos del atleta.

Una vez definido dicho calendario, es entregado a los atletas junto con información adicional como cuestionarios o formularios, para que después de cada entrenamiento o semanalmente el atleta los rellene.

Ya sea semanalmente, mensualmente o una vez se complete toda la etapa de entrenamiento, el atleta va entregando dichos “*records*” al entrenador, quien va evaluando de acuerdo a las respuestas de los mismos la evolución del atleta.

Este método de monitorización tiene las desventajas de cualquier sistema de monitorización sin el uso de las TICs: desplazamientos e incremento de costes de tiempo y, sobre todo, un problema derivado de la fiabilidad por el registro y gestión de los datos de manera manual.

En muchas ocasiones, un parámetro de estos documentos es registrar el pulso cardíaco medio de un ejercicio o justo el valor capturado al terminar el mismo, resultando un problema ya que no todos los usuarios saben hacerlo de la manera correcta, obteniendo así un valor incorrecto.

De la misma manera, la realización de los *records* no siempre se realiza cuando debido a descuidos de los atletas. Las preguntas que están destinadas a responderse justo después de un entrenamiento (lesiones físicas, nivel de satisfacción, velocidad, etc.) pueden ser respondidas tiempo después, distorsionando los datos y por lo tanto los resultados reales.

Estos dos problemas hacen que una vez que el supervisor o entrenador obtiene toda la información, parte de la base de que dicha información no tiene por qué ser del todo correcta, lo que hace que pueda errar en su diagnóstico sobre el atleta y por lo tanto perjudicar a corto y largo plazo su entrenamiento.

El objetivo de CloudFit es automatizar todo este proceso, para que una vez que el atleta finalice un entrenamiento envíe todos los datos inmediatamente al supervisor y éste obtenga los datos correctos con los que poder realizar una correcta evaluación.

8.3.3. Desarrollo de CloudFit

El desarrollo de CloudFit se ha realizado siguiendo el framework propuesto. La metodología, y su aplicación en CloudFit, se explica a continuación etapa por etapa.

Una peculiaridad con respecto a CloudRehab, y la mayoría de las plataformas de telemonitorización, es que en CloudFit es posible que el usuario supervisor (entrenador) y el usuario monitorizado (atleta) sean la misma persona.

La única peculiaridad en la aplicación de e-MoDe es que el usuario monitorizado y supervisor no suele presentar problemas típicos de usuarios de plataformas de telemonitorización (limitación de conocimiento tecnológico, problemas cognitivos, etc.), lo que simplifica la especificación de requisitos. De esta manera, el resto de la metodología es similar y el framework propuesto es perfectamente aplicable.

8.3.3.1. Especificación de requisitos

Los usuarios del sistema son entrenadores (supervisores) y atletas/deportistas (usuarios monitorizados), que son los que van a usar la plataforma de telemonitorización y conocen los requisitos para que dicha solución sea práctica.

Los expertos en el área, con la ayuda directa de atletas, partiendo de experiencias previas y a través del estudio de trabajos de campo, definieron un nuevo protocolo de

telemonitorización que permita solucionar los problemas que ellos detectaron en la aproximación clásica.

Este nuevo protocolo está compuesto por las siguientes etapas:

1. **Estudio previo y entrevista con el atleta/deportista.** Como primera etapa, el entrenador tiene una reunión o entrevista con el atleta para conocerlo personalmente y determinar sus datos fisiológicos (edad, estatura, peso, etc.), su experiencia (tiempo que lleva practicando deporte, qué deportes práctica, cuántas veces practica deporte a la semana, records personales de diferentes ejercicios, etc.) y su disponibilidad de cara a la nueva etapa de entrenamiento (cuántos días a la semana puede dedicarle, en cuántas semanas o meses se estipula el entrenamiento, etc.).
2. **Determinar los elementos del entrenamiento.** Una vez definidas las características del atleta y analizados los datos del mismo, el supervisor o entrenador determina todos los elementos del entrenamiento. Estos elementos, independientes entre sí, son ejercicios englobados en alguna categoría en concreto (ejercicios continuos, ejercicios por series, ejercicios de musculación, descanso pasivo, etc.) u otros elementos para determinar el *feedback* del atleta (por ejemplo, diferentes tipos de preguntas).
3. **Construcción de entrenamientos.** Con estos elementos o bloques básicos (ejercicios y preguntas) los supervisores construyen entrenamientos completos. Por ejemplo, a partir de elementos principales, como preguntas, el entrenador construye encuestas/cuestionarios que el atleta debe contestar justo antes o después de un ejercicio. De igual manera, puede crear grupos de ejercicios para facilitar englobar objetivos (resistencia, fuerza, etc.).
4. **Planificación del entrenamiento.** Una vez definidos los distintos entrenamientos, que conformarán toda la etapa de entrenamiento del atleta, el supervisor planifica los mismos a lo largo del tiempo desde el primer día hasta el último de supervisión. Estos entrenamientos irán siendo asignados diaria o semanalmente para conformar una planificación completa de todo un entrenamiento planificado y con unos objetivos marcados.
5. **Parámetros fisiológicos, contextuales e inerciales.** Además de la realización de entrenamientos por parte del atleta, de acuerdo a cómo los hayan creado los supervisores/entrenadores, estos demandan de otros datos para determinar otros parámetros claves de la evolución de un atleta y su salud. Estos datos son los provenientes de parámetros fisiológicos, inerciales y contextuales. Por esto, junto a la ejecución de un determinado ejercicio, se hace necesario un resumen informativo sobre el pulso cardíaco, valores inerciales de movimiento y la distancia recorrida. Estos tres parámetros son claves, aunque no están presentes en todos los ejercicios.
6. **Realización de entrenamientos.** Una vez planificados por parte del entrenador todos los entrenamientos para un periodo determinado de tiempo, el atleta, y dependiendo de dicha programación, los realizará en su hogar, centro deportivo o por espacios públicos (dependiendo del ejercicio). Mientras realiza el entrenamiento se debe determinar los datos fisiológicos, contextuales e inerciales necesario para posteriormente compartirlos con el entrenador.
7. **Supervisión y comunicación.** Una vez realizado un entrenamiento, el entrenador puede revisar toda la ejecución y determinar de acuerdo a lo que está programado y lo que se ha realizado, el éxito de dicho entrenamiento, para ver hasta qué punto el atleta lo ha cumplido o realizado correctamente. Además, podrá revisar todos los parámetros extras (fisiológicos, contextuales, inerciales) para determinar posibles problemas de salud o ver

su estado de salud concreto para un ejercicio determinado que está programado con una intensidad concreta.

Se llevaron a cabo diferentes reuniones entre ingenieros de software y los distintos expertos en el área (algunos eran supervisores y atletas al mismo tiempo) con el objetivo de, a partir del protocolo, definir el escenario de telemonitorización que se pretendía crear.

Este escenario, representado en la Figura 90, representa el protocolo, pero incorporando elementos fundamentales ya expuestos en esta tesis, y que forman parte del soporte tecnológico al framework, como puede ser la computación en la nube, los dispositivos móviles como smartphone y otros como sensores de pulso cardíaco (*wearables*) o el propio GPS o acelerómetro del smartphone.

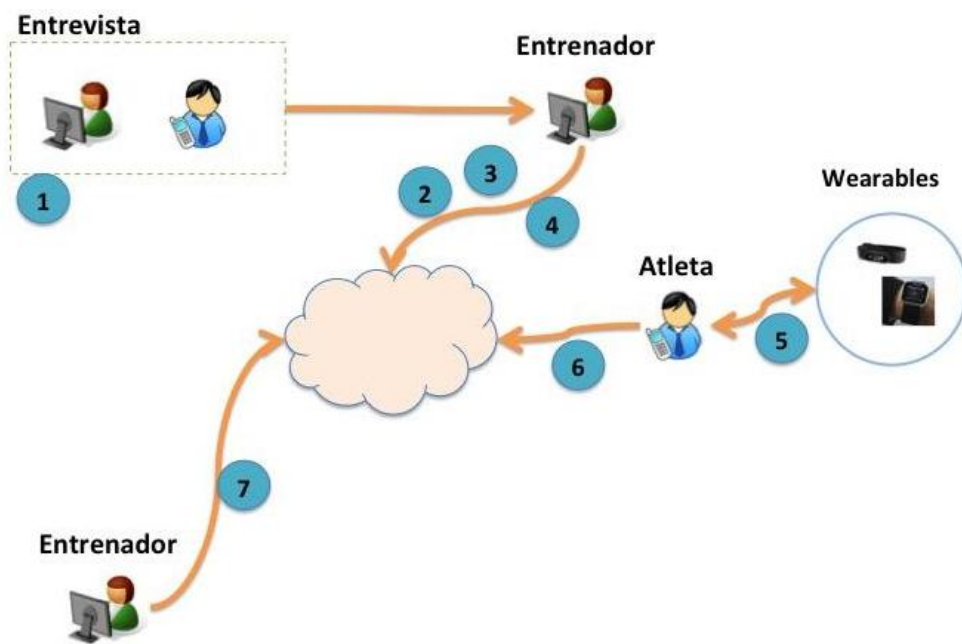


Figura 90- Protocolo de telemonitorización de CloudFit

De este protocolo y escenario representado en la Figura 90, existen varias tareas de telemonitorización. El atleta realiza un entrenamiento, que está conformado por un conjunto de tareas, como pueden ser responder a una pregunta (de los 5 tipos diferentes que hay), correr 10 kilómetros, realizar 3 series 100 metros o realizar un descanso completo de 5 minutos.

Para explicar cómo se ha aplicado el framework con CloudFit, vamos a suponer una de estas tareas: *andar 10 kilómetros y almacenar el valor mínimo, máximo y medio del pulso cardíaco capturado durante la realización del ejercicio.*

Partiendo de esta tarea, del protocolo de telemonitorización, y de acuerdo a la metodología, se aplican las dos fases de análisis correspondientes a esta etapa: registro y análisis de la actividad y especificación de las necesidades del usuario e interacción.

Registro y análisis de la actividad

En esta fase de análisis se identificaron las distintas funcionalidades de la plataforma, partiendo de las tareas identificadas, así como otras funcionalidades de la plataforma necesarias para dar soporte a esas tareas (por ejemplo, tratamiento de mapas geográficos) y restricciones sobre dichas funcionalidades (requisitos no funcionales).

De esta fase se definieron numerosos requisitos funcionales, entre las que destacan: visualización de recorridos usando mapas, captura de pulso cardíaco, guardar datos de pulso cardíaco en la nube, soporte a diferentes tipos de actividades físicas o ejercicios, recuperar tareas formadas por un conjunto de datos, gestión de información en el smartphone previa al almacenamiento en la nube, etc.

Como restricciones a las funcionalidades, se recopilaron algunos requisitos no funcionales como por ejemplo la fiabilidad de los datos de pulso cardíaco (*wearable*) o la fiabilidad del cálculo de posiciones del GPS del smartphone

Especificación de interacciones y necesidades especiales del usuario

Debido a que CloudFit es una plataforma de telemonitorización con el objetivo de que un usuario experto (supervisor) asesore a un usuario monitorizado sobre las actividades diarias para así mejorar su calidad de vida, la interacción entre ambos actores es fundamental para garantizar una correcta telemonitorización.

En esta fase, los supervisores y expertos en el área determinaron algunas de las necesidades especiales de los atletas, como pueden ser: interfaces limpias y simples, que el *wearables* ergonómicos ya que se usará mediante la realización del ejercicio, que hubiese notificaciones sonoras y táctiles para conocer el comienzo y fin de un determinado ejercicio, etc.

Además, para dar soporte a una correcta telemonitorización los supervisores indicaron que era necesaria una colaboración entre supervisor y atleta bidireccional, esto es, que el atleta pueda ponerse en contacto con el supervisor cuando desee y viceversa.

Aunque la plataforma está orientada a ser un sistema colaborativo multidisciplinar, como ya se ha comentado, en esta primera versión únicamente hay dos actores. Por esto, aunque la plataforma esté orientada a varios tipos de notificaciones entre diferentes tipos de usuarios, la versión desplegada actualmente permite interacción entre entrenador (supervisor) y atleta/deportista (usuario monitorizado).

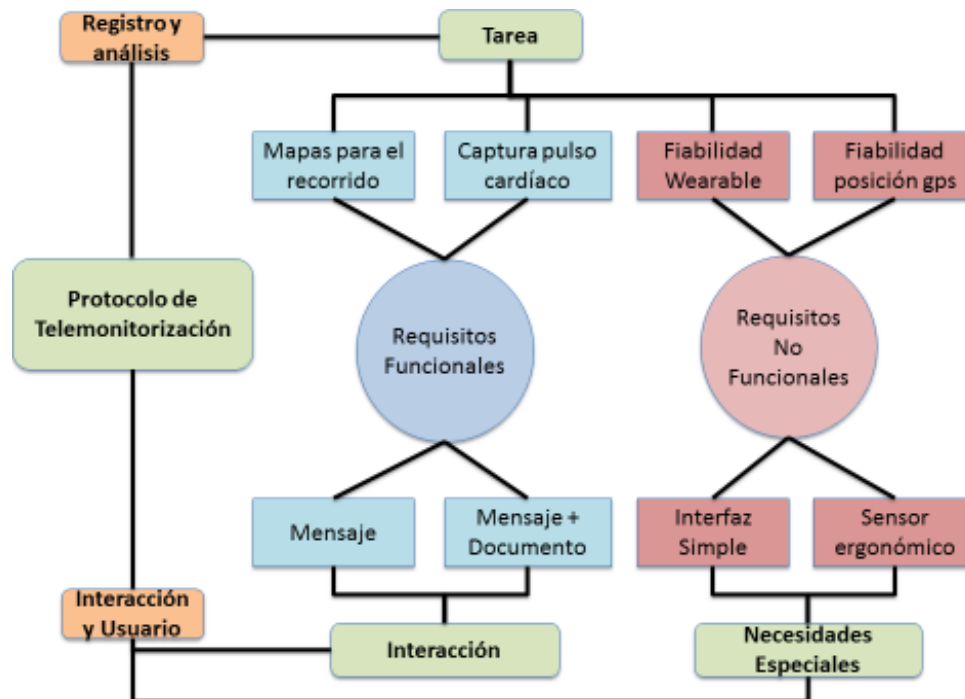


Figura 91 - Especificación (parcial) de requisitos de CloudFit

Dentro de esta interacción, los supervisores determinaron que debía haber 2 tipos de interacciones: mensajes simples y mensajes con un documento adjunto

La figura 91 muestra la etapa de especificación de requisitos a través del análisis de las dos fases, partiendo del protocolo.

8.3.3.2. Diseño

Con los requisitos funcionales y no funcionales de la etapa anterior, los ingenieros abordan el diseño de la plataforma. Para esto, los ingenieros centran dicha etapa de diseño en las cuatro áreas de interés propuestas en la metodología: arquitectura, colaboración, gestión de tareas y necesidades del usuario.

a) Arquitectura

Dentro del diseño, en este aspecto los ingenieros, de acuerdo a requisitos no funcionales y recursos disponibles determinan la tecnología a usar. Partiendo de la arquitectura propuesta en el capítulo 6, donde están identificados los diferentes elementos, se diseñan los diferentes servicios y submodulos software que representan todas las funcionalidades plasmadas en los requisitos funcionales y que en la siguiente etapa (implementación) los programadores implementarán.

Tecnología

Con respecto al ámbito tecnológico, se usará, como plataforma *cloud*, la tecnología Google Cloud (*Google App Engine*) de la compañía Google, como apuesta por una plataforma tecnológica comercial que garantice la robustez del sistema para un elevado número de usuarios.

En la capa de SaaS de dicha tecnología se crearán los servicios que se usarán en la capa PaaS por parte de las aplicaciones del sistema. Estos servicios seguirán una estructura híbrida, como la propuesta en el framework, para poder acceder a la información concreta.

Se usará un modelo de datos relacional pero basado en Google SQL (no relacional en la implementación, pero relacional en el uso), permitiendo que en un futuro la transformación a un modelo no relacional (grafos, por ejemplo) sea automática en caso de ser necesario.


Como otros elementos tecnológicos se han usado smartphone con sistema operativo Android y una banda de pulso cardíaco Polar WearLink Bluetooth, el sensor multifuncional Zephyr y el acelerómetro y GPS internos del propio dispositivo móvil.

Distribución de funcionalidades (*Back-end* y dispositivos móviles)

Partiendo de los requisitos funcionales, en esta etapa de diseño los ingenieros diferencian entre los que derivaron como servicios a la nube (*back-end*) y los que quedarán dentro del propio dispositivo móvil.

De acuerdo a la arquitectura software híbrida (SOA-ROA) propuesta y explicada en el capítulo 6, todas aquellas funcionalidades que estén relacionadas con la gestión de información y en este caso, con la gestión de videos y contenido multimedia, derivarán a la nube como servicios web basado en REST.

Un ejemplo de estos servicios es el mostrado en la figura 92, que permite recuperar toda la información de un entrenamiento correspondiente a un atleta en concreto.



<http://cloudfitimuds0-1076.appspot.com/training/20>

Figura 92 – Acceso a un recurso en CloudFit

A través del uso de la propia tecnología se consumen servicios que soportan funcionalidades tales como:

- Dar de alta nuevos atletas o entrenadores.
- Crear, modificar y recuperar sesiones de entrenamiento.
- Gestionar elementos básicos (ejercicios de los distintos grupos o preguntas) así como grupos de estos (grupos de ejercicios y encuestas)
- Planificar mediante un calendario personalizado un entrenamiento a largo plazo para un atleta.
- Gestión de parámetros fisiológicos, inerciales o contextuales (añadir o recuperar dependiendo a parámetros de tiempo).

El resto de funcionalidades, que son intrínsecas a la acción de monitorización, como puede ser registrar el entrenamiento (tiempos, distancia, etc.), el pulso cardíaco o las distintas acciones del usuario, se implementa directamente en la aplicación móvil

b) Modelo colaborativo

Como se ha comentado, CloudFit es una plataforma de gestión integral para el bienestar, que, aunque actualmente se encuentra en su primera etapa de creación y se centra únicamente

en deportistas y entrenadores como tipos de usuario del sistema, en un futuro y por su diseño se pretende que de soporte a otros tipos de usuarios de otras disciplinas.

En este caso, el modelo presentado para entornos colaborativos es válido al ser multidisciplinar (diferentes tipos de supervisores), haciendo uso así los distintos tipos de notificaciones. De esta manera, y en un futuro, cuando haya distintos tipos de expertos que necesiten interactuar entre sí, la implementación de acuerdo al modelo exacto permitiría dar soporte colaborativo entre varios expertos, ya que hasta la fecha únicamente hay dos tipos de usuarios y solo se da soporte a las interacciones entre supervisores (entrenadores) y usuarios monitorizados (atletas).

Como se especificó en la fase de análisis de interacción y usuario de la etapa de especificación de requisitos se identificaron dos tipos de interacciones: mensajes de texto y mensajes de texto con un documento adjunto, tanto del entrenador hacía al atleta como del atleta hacia el entrenador.

Para esto, partiendo del modelo presentado se extiende e instancia el mismo para representar el modelo colaborativo de CloudFit en su versión actual (ver Figura 93).

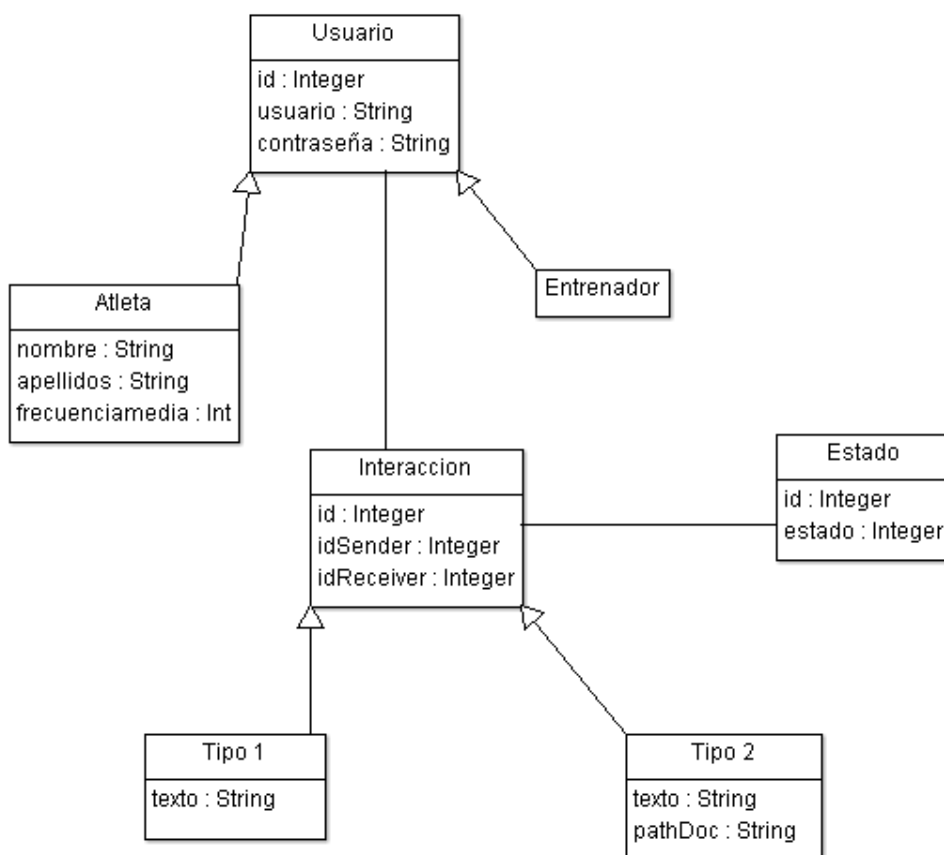


Figura 93 – Modelo colaborativo para CloudFit

El modelo representado en la figura 92, y que parte del modelo original para dar soporte al tipo de interacción de CloudFit, identifica los usuarios del modelo original con atributos concretos (nombre, apellidos, usuario, contraseña, etc.).

La otra parte del modelo especifica los distintos tipos de interacciones, dos tipos en ese caso como se comentó en la etapa de especificación de requisitos: envío de mensajes simple y con un documento adjunto. Además de especificar estas interacciones, se mantiene la gestión de las mismas a través de su estado y de su emisor y receptor, para poder enviar entre atleta y supervisor y viceversa gestionando cuando se ha enviado y recibido, ya que debido a los requisitos es importante que tanto el supervisor como el usuario monitorizado sean conscientes de que su mensaje ha llegado correctamente

c) Gestión de tareas de telemonitorización

Siendo la aplicación móvil la que permite la ejecución de los entrenamientos, es la encargada de mostrar las distintas vistas de dichas tareas y gestionar la lógica de control, de acuerdo al modelo propuesto.

Siguiendo este modelo se han diseñado las diferentes tareas de telemonitorización, diseñando cada tarea como un conjunto de datos que posteriormente se materializaron con una notación concreta, en este caso JSON, y dejando la lógica de control sobre las tareas como parte de las funcionalidades de la aplicación móvil.

En CloudFit una de las tareas, que hemos puesto de ejemplo, es: *andar 10 kilómetros y almacenar el valor mínimo, máximo y medio del pulso cardíaco mientras se realiza ese ejercicio.*

Aunque esta tarea sea la inicialmente especificada en los requisitos, por el propio protocolo e interés de los supervisores tiene más información adicional de cara a ser más útil para su evaluación, como, por ejemplo: ritmo del atleta, hora de inicio y hora de fin, número de calorías gastadas, etc.

Partiendo de esta tarea, se diseñó la misma siguiendo el modelo para la gestión de tareas, identificando los siguientes elementos:

- Fecha → Fecha de la tarea.
- Distancia → Distancia a recorrer (en metros)
- Hora de inicio → Cuando comienza la tarea.
- Hora de fin → Cuando finaliza la tarea.
- Identificador → Número para identificar a la tarea.
- Ritmo → Ritmo medio en la ejecución de la tarea (km/h).
- Calorías → Calorías consumidas en la realización de la tarea.
- Frecuencia cardíaca mínima → Valor mínimo alcanzado durante la realización de la tarea.
- Frecuencia cardíaca máxima → Valor máximo alcanzado durante la realización de la tarea.
- Frecuencia cardíaca media → Valor medio de todos los valores registrados durante la realización del ejercicio.

Este modelo es diseñado por los ingenieros a partir de la especificación de la tarea en los requisitos funcionales y en el propio protocolo. Este modelo se usa para, a través de una notación como es JSON, representar dicha tarea (ver Figura 94).

Esta tarea, ya representada en JSON, es consumida por las diferentes aplicaciones, y siguiendo el patrón MVC propuesto, se usará para representar las vistas en las aplicaciones y su lógica de control se gestionará en los dispositivos móviles.

Como se ha comentado, se usará Android como sistema operativo de la aplicación móvil. En Android la interfaz de esta tarea estará representada en XML (Vista), la lógica de control (Controlador) se implementa directamente en la aplicación y el modelo sería el documento JSON que representa la tarea.

Por lo tanto, la aplicación consume un servicio que devuelve una tarea concreta siguiendo el modelo de la figura 93 pero con un estado concreto, el controlador se encargaría de tomar dicho modelo y la interfaz implementada en Android y completar dicha interfaz con los datos del modelo para mostrar la tarea correspondiente. Después, a través de la implementación adecuada se interactúa con el software para la realización de la tarea.

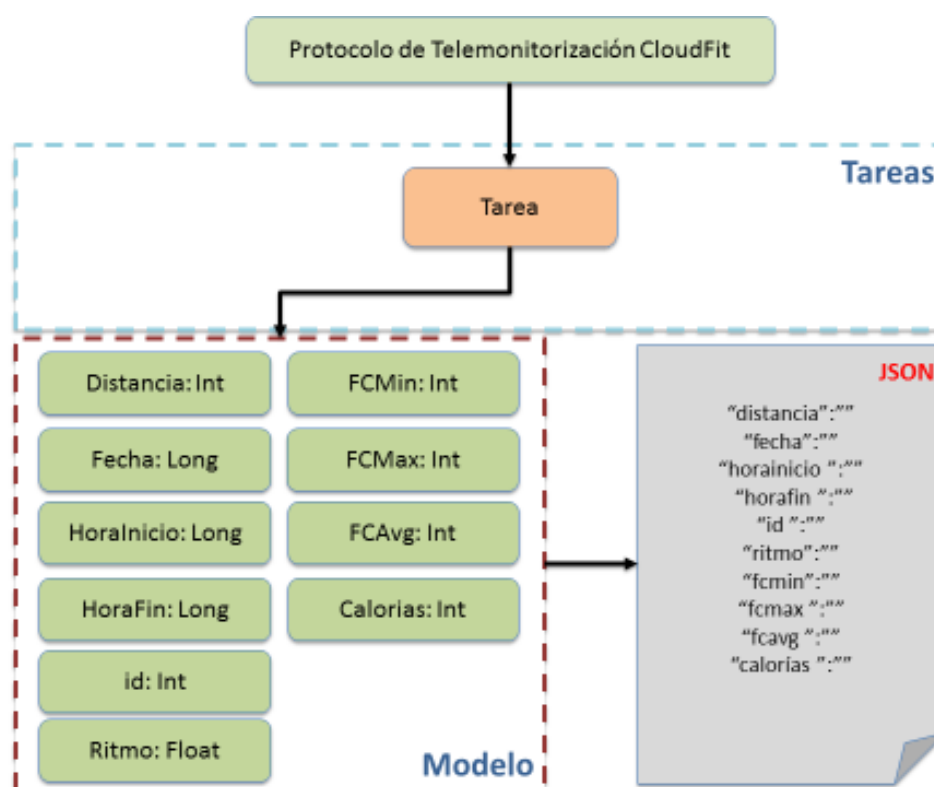


Figura 94 – Modelo para tarea de CloudFit

d) **Necesidades del usuario**

Los usuarios monitorizados que usan la aplicación no tienen ningún tipo de restricción cognitiva, por lo tanto, no había necesidades especiales relevantes, más allá de las de cualquier aplicación móvil: interfaz usable y simple, iconos que intuyan la funcionalidad que representan, etc.

Si se hizo especial hincapié en que la ejecución de determinadas tareas sea usable para el atleta (diseño correcto de la interfaz de usuario). Esto es, por ejemplo, bloquear la pantalla cuando se está realizando una actividad para no parar involuntariamente, notificar mediante sonidos cuando se pulsa un botón de inicio de ejercicio concreto, que la pantalla del dispositivo móvil no se apague por inactividad, etc.

De este aspecto de diseño, se generaron diferentes interfaces de usuario con la ayuda de entrenadores y atletas, y se mejoraron con la ayuda de un diseñador externo. Estas interfaces posteriormente se implementaron en los dispositivos móviles para representar adecuadamente tanto tareas como la propia interfaz de la aplicación.

8.3.3.3. Implementación

a) Implementación de funcionalidades

La implementación de las distintas funcionalidades la realizan los diferentes programadores dependiendo del ámbito de la funcionalidad: *back-end* o en los dispositivos móviles.

Las funcionalidades del *back-end* (servicios web) se implementaron usando el lenguaje de programación Java junto con JAX-RS (Jersey como tecnología) para implementar la arquitectura software híbrida (ROA-SOA). A diferencia de CloudRehab, donde la mayoría de los servicios son implementados por la propia tecnología, en CloudFit se implementaron los diferentes servicios.

La implementación de las distintas funcionalidades para la aplicación móvil parte del uso de la plataforma Zappa, explicada en el capítulo 6, sección 4 y como herramienta de soporte para el framework propuesto.

Zappa proporciona funcionalidades implementadas, para que desarrolladores a través de la API de los distintos componentes usen las funcionalidades, sin necesidad de implementar estas funcionalidades.

Aunque la plataforma está compuesta por numerosos componentes con diferentes funcionalidades, no todos los componentes se usan en CloudFit, usándose las siguientes funcionalidades de Zappa:

- Intercambio de datos con la nube, a través del componente zHTTP
- Gestión de datos internos: Gestión de la estructura de directorios interna de Android (tarjeta sd) para guardar, recuperar, eliminar, mover documentos.
- Transformación de modelos JSON a objetos: Usado para transformar las tareas en objetos software (en Java), para así poder gestionar la tarea
- Acceso a los sensores internos del dispositivo (GPS y acelerómetro)

b) Integración de wearables

Para este proyecto se han usado varios dispositivos, como el Polar WearLink Bluetooth ya mostrado en la figura 83 o el sensor multifuncional Zephyr (pulso cardíaco, temperatura, acelerómetro) cuyo modelo e interacción corresponde con el caso de estudio puesto en el capítulo explicativo de WearIt (capítulo 8).

c) Aplicaciones

CloudFit consta de dos aplicaciones, una aplicación móvil y una plataforma web.

La plataforma web se usa por los profesionales/entrenadores, ya que la creación de contenido a través de esta es más fácil que a través de un dispositivo móvil, al tener que trabajar con un calendario y desplazamiento de contenido.

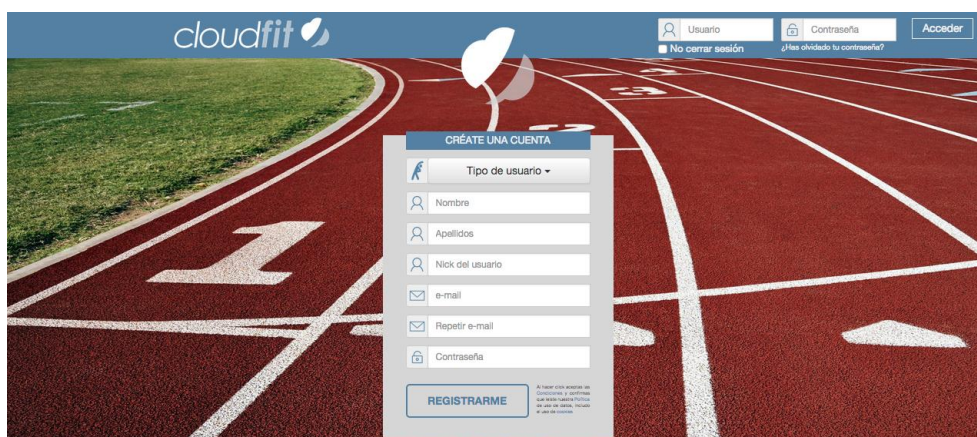
La aplicación móvil es usada por los atletas para la realización de los entrenamientos creados por los supervisores y atletas, así como revisar sus propios entrenamientos.

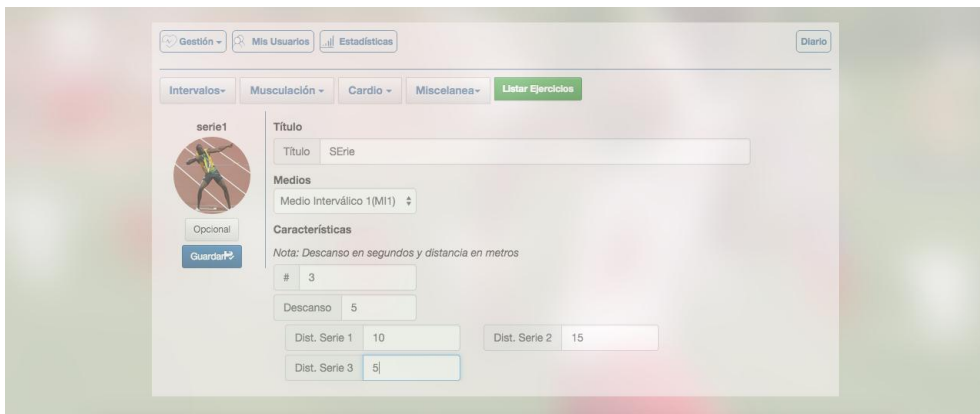
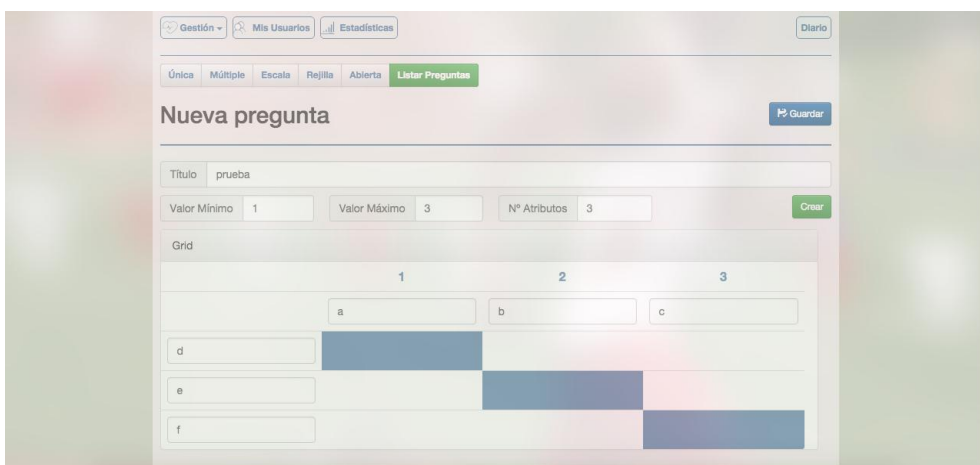
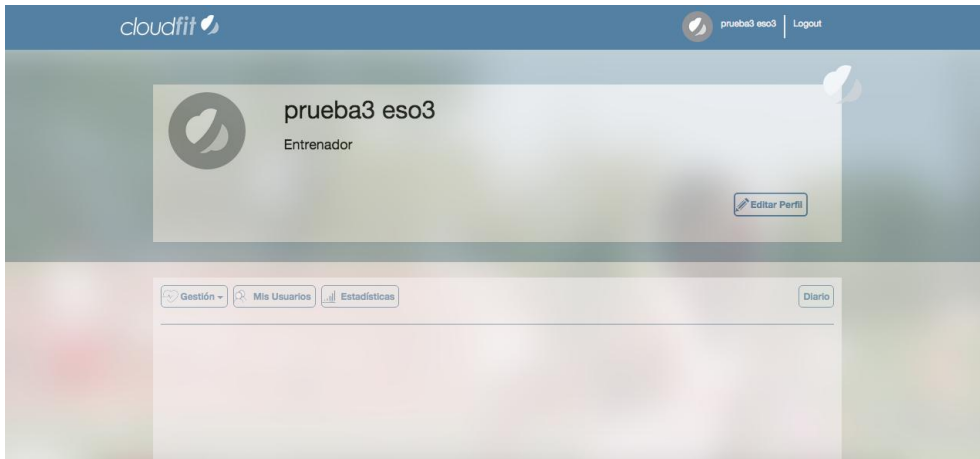
Plataforma Web

La plataforma web está diseñada para permitir a los profesionales gestionar toda la información:

- **Gestión de contenido propio:** A través de la plataforma web los entrenadores crean elementos propios que usarán para componer nuevos entrenamientos. Los entrenadores crean preguntas y ejercicios concretos como elementos básicos. Componiendo estos elementos crean nuevos entrenamientos que asignarán a atletas.
- **Gestionar entrenamientos del atleta:** A partir de los entrenamientos creados según elementos básicos (preguntas, cuestionarios, ejercicios y grupos de ejercicios) los entrenadores asignan estos entrenamientos a los atletas. Los entrenadores disponen de un calendario personalizado por cada atleta donde pueden ver los entrenamientos que tienen, reordenar, asignar nuevos, modificarlos, etc. De esta manera, es muy fácil e intuitivo para el entrenador ver toda la planificación de un atleta a lo largo del tiempo (semanas, meses o años).
- **Revisar la información generada por el atleta,** como por ejemplo los entrenamientos realizados, valores cardíacos entre un rango de fechas, gráficos estadísticos que muestran la evolución del atleta a lo largo del tiempo o toda la información concreta de un entrenamiento.

La Figura 95 muestra varias capturas de la plataforma web.





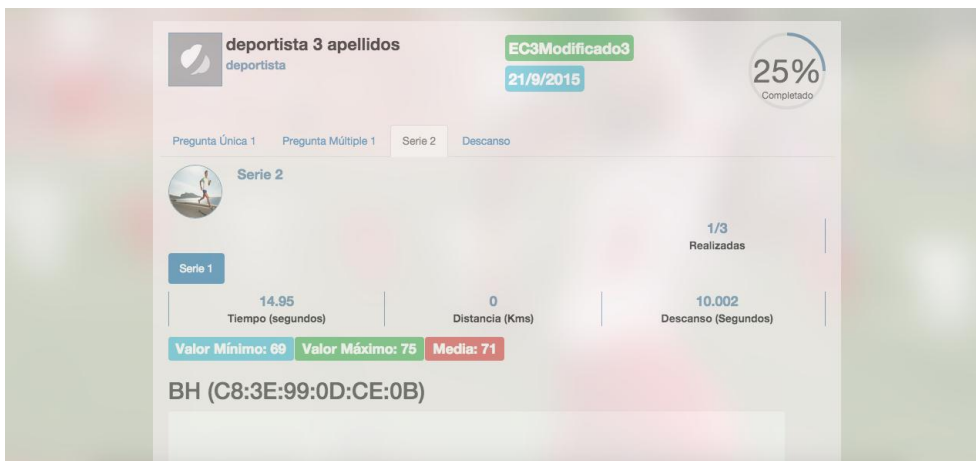
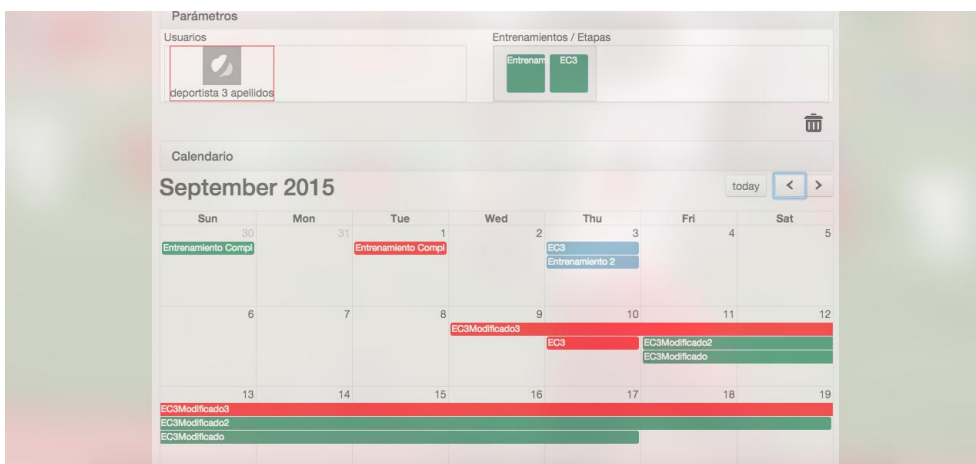
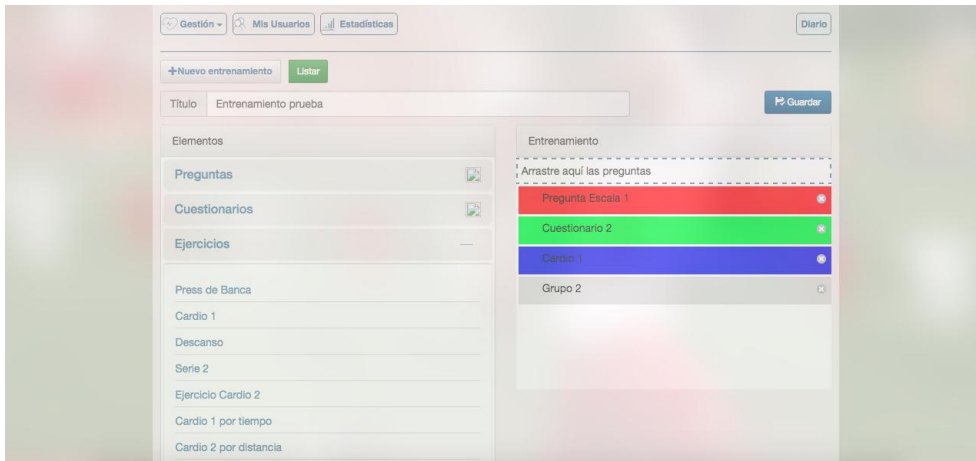




Figura 95 - Plataforma Web CloudFit

Aplicación Móvil

La aplicación móvil está desarrollada para Android (versión 4 y superiores) y será la herramienta usada por atletas para llevar a cabo sus entrenamientos y también.

A través de la aplicación los atletas visualizar su calendario de entrenamiento y seleccionar el entrenamiento que quieran realizar o revisar una sesión de entrenamiento ya realizada.

Una vez que comienzan el entrenamiento, esta muestra paso a paso los distintos elementos que componen dicho entrenamiento, en el mismo orden que ha sido definido por el entrenador, de tal manera que el atleta lo único que tiene que hacer es ir completando el entrenamiento de acuerdo a la interfaz de la aplicación y pasar al siguiente cuando lo haya completado.

Una vez se comienza un entrenamiento la aplicación automáticamente empieza a escanear para detectar dispositivos. Cuando detecta algunos de los dispositivos compatibles conecta con estos y empieza a monitorizar y a recuperar datos, los cuales son guardados una vez el usuario comience algún ejercicio concreto.

Cuando el entrenamiento finaliza, se crea un resumen de los datos fisiológicos (valor mínimo, máximo y media del pulso cardíaco) y son adjuntados como parte del entrenamiento. Una vez está todo el procesamiento completado se sube a la nube usando un servicio concreto para que sea visualizado por el entrenador.

La figura 96 muestra imágenes de la aplicación.

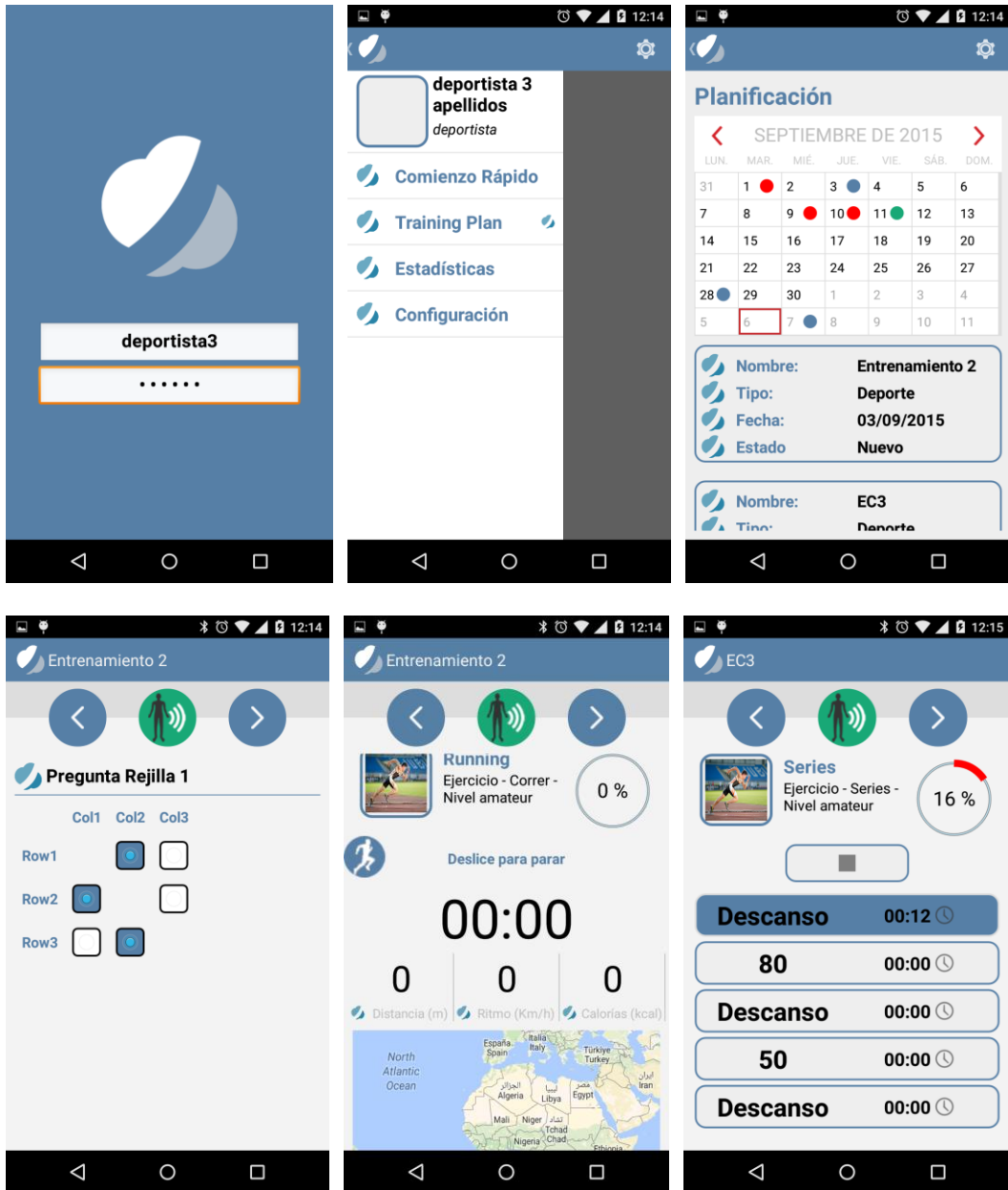


Figura 96 - Aplicación móvil de CloudFit

8.3.3.4. Despliegue

El despliegue de la solución se ha realizado en el IaaS ofrecido por Google (Google Cloud), usando *Google App Engine* como herramienta software para la integración con el IDE y despliegue automático en la nube.

De igual manera, se usa la plataforma *Google Cloud Console* para gestionar todo el tráfico de información y determinar los gastos, bases de datos, uso de servicios, etc.

Actualmente la plataforma se encuentra en producción, pero en una fase de testeo Beta, donde está siendo usada por un número controlado de usuarios para detectar y corregir errores y añadir las funcionalidades que sea necesarias para cumplir con las expectativas.

8.3.3.5. Validación

Debido a que actualmente la aplicación está en producción, pero en una fase beta, no se tienen datos empíricos o estudios comparativos que avalen la eficiencia de CloudFit como plataforma de telemonitorización de acuerdo a la satisfacción de objetivos.

No obstante, aunque el sistema no esté validado desde un punto de vista de eficiencia y funcionalidad, se realizó un estudio para su validación en relación a la usabilidad de la aplicación móvil de cara al usuario monitorizado.

Este estudio, explicado a continuación, se realizó en colaboración con el grupo de investigación *Quality of Life (QoL)* de la Universidad de Ginebra, siendo Katarzyna Wac la máxima responsable y siendo ella misma y varios miembros del grupo partícipes del estudio.

Validación de CloudFit bajo un modelo heurístico

Introducción

Las aplicaciones fitness están diseñadas para registrar las acciones de los usuarios, tanto en la ejecución de tareas como en el registro de parámetros fisiológicos, inerciales y contextuales. Además, toda la información y servicios que proporcionan está accesible en tiempo real y desde cualquier sitio gracias a tecnologías como la computación en la nube o los smartphone.

No obstante, aunque estas aplicaciones son descargadas por millones de usuarios por su buen funcionamiento, en muchas ocasiones estas reflejan ciertos problemas relacionados con la usabilidad o la experiencia del usuario, como por ejemplo el tamaño de letra que dificulta la lectura para determinado grupo de usuarios o difícil accesibilidad a ciertas funcionalidades del sistema.

En esta sección se pretende validar la usabilidad y experiencia del usuario monitorizado (atleta) de CloudFit a través de un estudio formal. Con el objetivo de medir la usabilidad y experiencia del usuario se realizó un estudio con varios usuarios (evaluadores) usando un método de evaluación heurístico [Silva 2014].

En este estudio, a usando el método heurístico comentado, primero se evalúa la usabilidad y experiencia de usuario de dos de las aplicaciones similares (y ampliamente conocidas) a CloudFit: Endomondo y Runtastic. Una vez obtenidos los resultados, usaremos estos para mejorar la usabilidad y experiencia de CloudFit.

Una vez CloudFit esté mejorado de acuerdo a la información obtenida del estudio de las otras dos aplicaciones, se aplicará el mismo método heurístico para determinar la usabilidad y experiencia de usuario y poder compararlo así objetivamente con Endomondo y Runtastic.

Metodología de Evaluación

Después de un proceso previo estudiando los diferentes métodos de evaluación de usabilidad y experiencia de usuario de una aplicación software, para este estudio se ha optado por el método de evaluación presentado en [Silva 2014]. Las razones por las que se ha decidido usar este método para la evaluación de usabilidad y experiencia de usuario para el estudio aquí presentado son:

- El método de evaluación está diseñado específicamente para evaluar aplicaciones fitness, que, por el estado actual del sistema a analizar, es el contexto en el que nos centramos. Esto representa una enorme ventaja en relación a usar una metodología general (aplicable a cualquier software) y adaptarla bajo nuestro criterio y necesidades, lo cual puede implicar una interpretación personal de la metodología con lo que se obtendrían resultados muy condicionados y por lo tanto cuestionables.
- El proceso de aplicar esta metodología a una aplicación fitness es simple para el evaluador, no requiere mucho tiempo y tampoco es necesario que dichos evaluadores sean expertos en el dominio o aplicaciones fitness.
- En el estudio presentado en [Silva 2014] se evalúan, de acuerdo a esta metodología, dos aplicaciones: Nike+ y RunKeeper. Los resultados, que son usados para comparar la usabilidad y experiencia del usuario de ambas aplicaciones, demuestra cómo de útil y práctica es dicha metodología.

En [Silva 2014] se proponen un conjunto heurísticas como resultado de la investigación de combinar el resultado de otros trabajos de la misma área. Este conjunto está compuesto por 35 heurísticas que están organizadas en 6 categorías: percepción, cognición, destreza, navegación, contenido y diseño visual.

La tabla 4 muestra las 35 heurísticas que conforman el método y que se han usado para este estudio.

Heuristic Number	Heuristic Description
Cognition	
H1	Focus on one task at a time instead of requiring the user to actively monitor two or more tasks, and clearly indicate the name and status of the task at all times.
H2	Avoid the use of interaction timeouts and provide ample time to read information.
H3	Avoid the use of animation and fast-moving objects.
H4	Leverage mental models familiar to older adults.
H5	Reduce the demand on working memory by supporting recognition rather than recall.
H6	Aim at creating an aesthetical user interface, by using pictures and/or graphics purposefully and adequately to minimize user interface clutter and avoid extraneous details.
Content	
H7	Give specific and clear instructions and make help and documentation available. Remember that it is better to prevent an error than to recover from it.
H8	Provide clear feedback and when presenting error messages make them simple and easy to follow.
H9	Make sure errors messages are descriptive and use meaningful words and verbs when requiring an action.
H10	Write in a language that is simple, clear and adequate to the audience.
Dexterity	
H11	Avoid pull down menus.
H12	Avoid the use of scrolling.
H13	Enlarge the size of user interface elements in general; targets should be at least 14mm square.
Navigation	
H14	Keep the user interface navigation structure narrow, simple and straightforward.
H15	Use consistent and explicit step-by-step navigation.
H16	Make sure that the "Back" button behaves predictably.
H17	Support user control and freedom.

H18	Disable inactive user interface objects.
Perception	
H19	Allow users to fine tune the volume.
H20	Do not rely on color alone to convey information. Be aware of color blindness.
H21	Provide not only visual feedback, but also tactile and auditory.
H22	Make information accessible through different modalities.
H23	Use lower frequencies to convey auditory information such as confirmation tones and alerts.
H24	Do not use pure white or rapidly changing contrast backgrounds.
H25	Make it easy for people to change the text size directly from the screen.
H26	Allow users to fine-tune screen brightness and contrast.
Visual Design	
H27	Use high-contrast color combinations of font and/or graphics and background to ensure readability and perceptibility; avoid using blue, green and yellow in close proximity.
H28	Use color conservatively, limiting the maximum number of colors in use to ~four.
H29	Make sure text uses types, styles and sizes appropriate to older adults, for instance, but not exclusively: sans serif, non-condensed typefaces, non-italic, left justified and 12-14 point font.
H30	Make links and buttons clearly visible and distinguishable from other user interface elements.
H31	Make information easy to read, skim (or) and scan.
H32	Group information visually (make good use of color, text, topics, etc.).
H33	Allow sufficient white space to ensure a balanced user interface design.
H34	Use user interface elements consistently and adhere to standards and conventions if those exist.
H35	Use simple and meaningful icons.

Tabla 4 – Heurísticas para evaluación de aplicación fitness [Silva 2014]

No obstante, aunque el método está compuesto de dichas heurísticas, se hace necesario definir una metodología para aplicarlas correctamente y poder así determinar objetivamente la usabilidad o experiencia de usuario de una aplicación concreta.

La figura 97 muestra los pasos necesarios para llevar a cabo el estudio a partir del método de evaluación propuesto, los cuales son:

1. Buscar un grupo de evaluadores.
2. Escoger las aplicaciones fitness que se van a evaluar.
3. Determinar las diferentes vistas (pantallas) de las aplicaciones a evaluar, ya que aplicaciones tienen numerosas pantallas y se suele centrar el estudio en las más relevantes (la vista para ejecutar la tarea, configuración, vista principal, etc.).
4. Aplicar las 35 heurísticas sobre las distintas vistas y determinar cuándo una heurística se cumple o no. Si una heurística no se cumple se indica que dicha vista tiene una *vulnerabilidad (violación) o fallo*.
5. El número total de vulnerabilidades o fallos es el parámetro usado para determinar la usabilidad de la aplicación fitness.

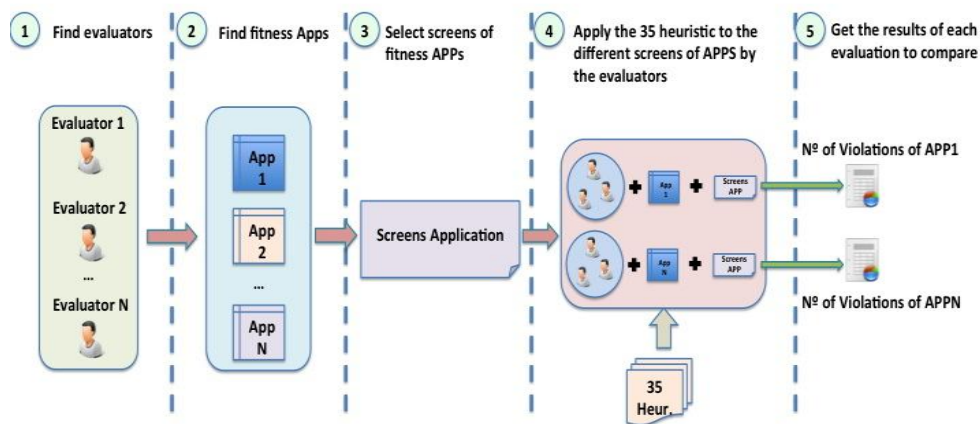


Figura 97 – Metodología para la evaluación y comparación de aplicaciones fitness

Estudio

En el estudio, como se ha comentado, se evaluó la usabilidad y experiencia de usuario de CloudFit. Con el objetivo de garantizar la fiabilidad de los resultados y que el estudio se realiza de la manera apropiada, dicho estudio se ha realizado teniendo en cuenta las siguientes premisas:

- Las heurísticas deben aplicarse en las mismas vistas (pantallas) de las diferentes aplicaciones, para que la comparación tenga coherencia y pueda ser comparable el resultado de una aplicación y otra.
- El número de evaluadores debe ser un número fijo determinado a lo largo de todo el estudio.
- Con el objetivo de obtener unos resultados independientes e imparciales de cada aplicación, se evaluó una aplicación por semana.
- Para garantizar la independencia de los evaluadores, estos no pueden tener interacciones entre ellos durante la evaluación de las aplicaciones.
- Las vistas seleccionadas para ser evaluadas se evaluarán en el mismo orden.

Una vez asumidas estas premisas que deben cumplirse, las distintas etapas del estudio son las siguientes:

1. **Encontrar el grupo de evaluadores.** Este estudio se realizó con seis evaluadores (4 hombres y 2 mujeres), miembros del grupo de investigación QoL (*Quality of Life*) de la Universidad de Ginebra.
2. Siguiendo la metodología del trabajo original [Silva 2014], **se eligen las aplicaciones a evaluar**, además de CloudFit. Se decidió evaluar Endomondo [Endomondo Web] y Runtastic [Runtastic Web], por ser dos de las aplicaciones más conocidas y usadas en el área fitness.
3. **Definir las interfaces/vistas a evaluar.** Siguiendo una de las premisas, debíamos garantizar que las vistas seleccionadas debían estar en las tres aplicaciones. Se decidieron evaluar las vistas de: configuración, entrenamiento/ejercicio y revisar datos de un entrenamiento/ejercicio.

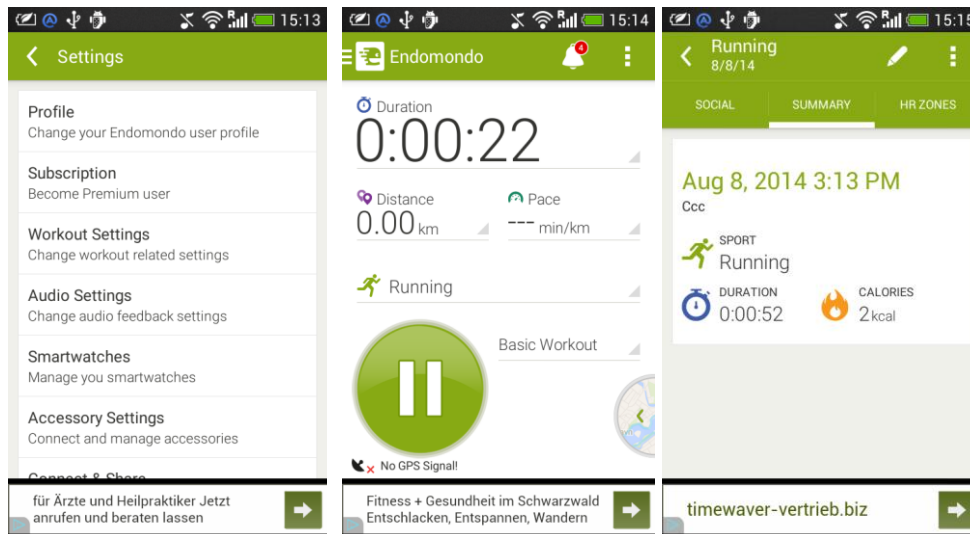


Figura 98 – Capturas de pantalla Endomondo

4. La primera semana, siguiendo las premisas, **se evaluó Endomondo** (ver Figura 98). La información generada por los evaluadores se procesó para obtener resultados y conclusiones útiles. Con esta información procesa se mejoró CloudFit en términos de usabilidad y experiencia de usuario.
5. La segunda semana **se evaluó Runtastic** (ver Figura 99). Al igual que con Endomondo, se procesó la información de los evaluadores y se mejoró CloudFit de acuerdo a esta información

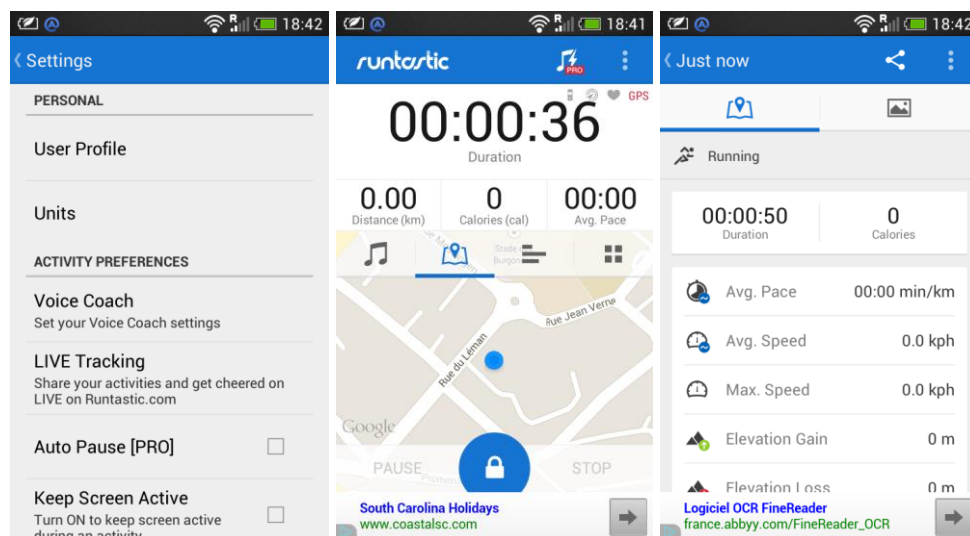


Figura 99– Capturas de pantalla Runtastic

6. La tercera semana **se evaluó CloudFit**, procesando la información del mismo modo que las aplicaciones anteriores.
7. Una vez evaluadas las tres aplicaciones y procesados los datos, se compararon para determinar qué solución tenía mejor puntuación en términos de usabilidad y experiencia del usuario.

La figura 100 muestra las diferentes etapas del estudio.

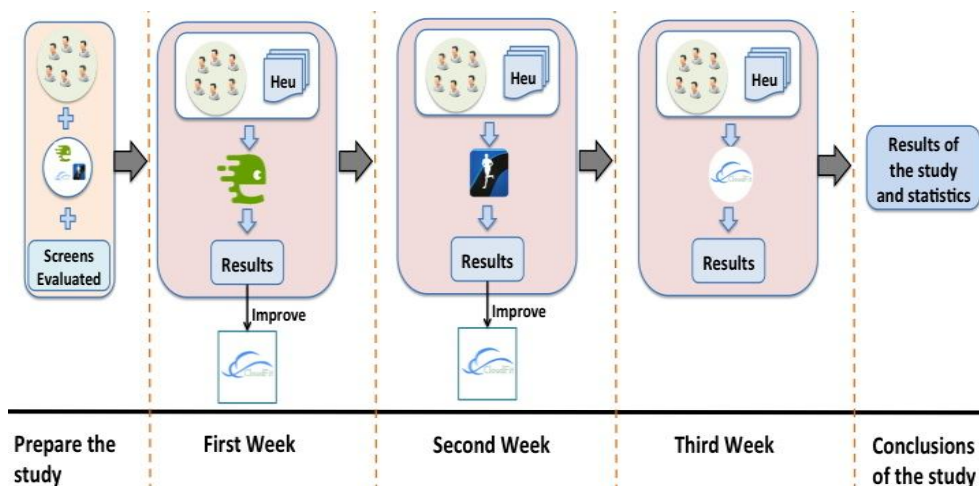


Figura 100 – Metodología para la evaluación y comparación de aplicaciones fitness

Cada vez que se realizaba el proceso de evaluación de una aplicación, a cada evaluador se le proporcionaba una hoja donde deberían completar, de acuerdo a la vista que estaban evaluando: el número de vulnerabilidades o fallos y la opción de añadir un comentario adicional. La figura 101 muestra un ejemplo de un campo de dicha hoja. Existían 35 campos similares, uno por cada heurística del método.

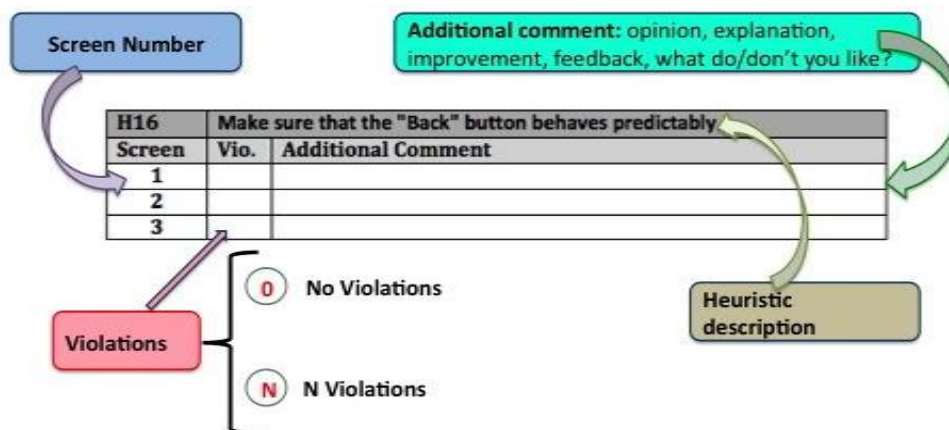


Figura 101 – Campo de evaluación a completar por los evaluadores

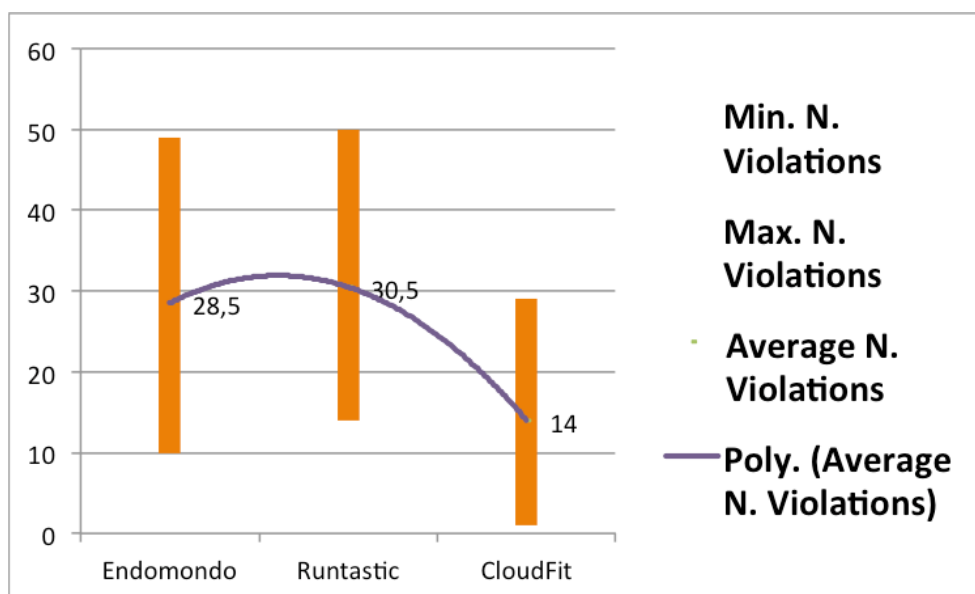
Resultados

El resultado presenta el valor mínimo, máximo, media y el número total de vulnerabilidad/errores de cada aplicación, así como el mínimo, máximo y media del tiempo invertido para realizar la evaluación de cada aplicación por cada evaluador (en minutos).

	Endomondo	Runtastic	CloudFit
Nº Evaluators	6	6	6
Min. N. Violations	10	14	1
Max. N. Violations	49	50	29
Avg. N. Violations	28,5	30,5	14
Total N. Violations	171	183	84
Min. Time	32	25	28
Max. Time	52	35	37
Avg. Time	41,5	31,16	32,3

Tabla 5 – Resultado del estudio para las diferentes aplicaciones

La figura 102 muestra varias gráficas que presentan la variación en el número de errores/vulnerabilidades, siendo esta la variable o parámetro más significativo.



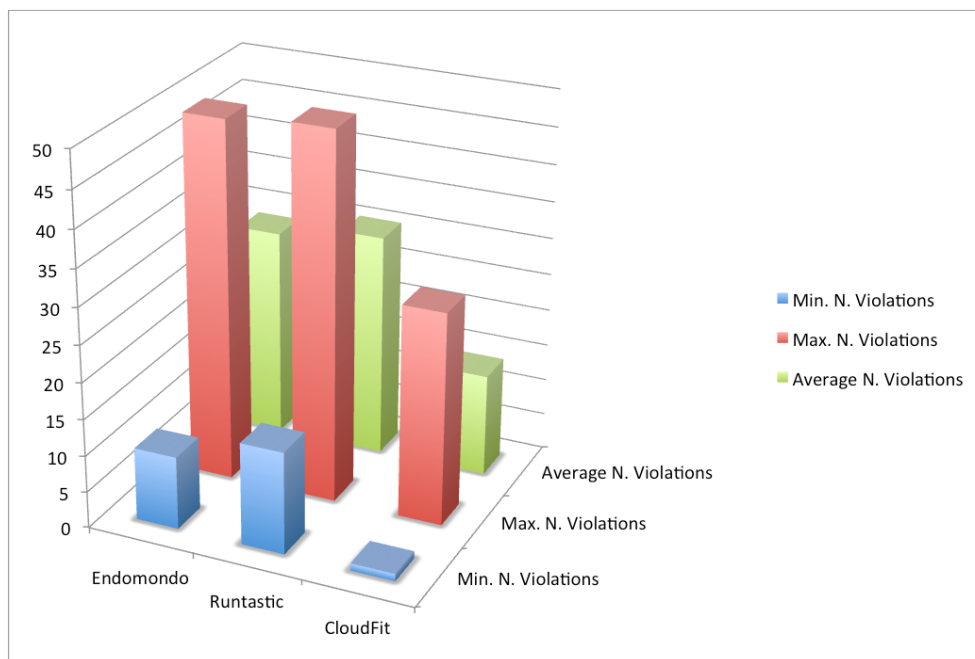


Figura 102 – Registro de errores (violaciones) en las diferentes aplicaciones

Aunque estos gráficos resultan útiles, toda la información del estudio es crucial para entender dichos resúmenes y poder dilucidar unas conclusiones claras.

La tabla 6 muestra toda la información del estudio: el número de vulnerabilidades o errores por aplicación en total y heurística, el porcentaje de acuerdo a la categoría evaluada y el porcentaje del total.

	Heuristic Number	Endomondo	Runtastic	ABC	Total By Heuristic	% within Category	% in Total
COGNITION	H1	3	3	0	6	20	1,3953
	H2	0	0	0	0	0	0
	H3	3	3	0	6	20	1,3953
	H4	3	2	0	5	16,6666	1,1627
	H5	0	1	1	2	6,66666	0,4651
	H6	5	4	2	11	36,6666	2,5581
	Total	14	13	3	30	100	6,9767
CONTENT	H7	5	10	5	20	74,0740	4,6511
	H8	1	1	1	3	11,1111	0,6976
	H9	1	0	1	2	7,40740	0,4651
	H10	0	0	2	2	7,40740	0,4651
	Total	7	11	9	27	100	6,2790
DEXTERITY	H11	2	6	0	8	17,7777	1,8604
	H12	11	15	8	34	75,5555	7,9069
	H13	2	1	0	3	6,6666	0,6976
	Total	15	22	8	45	100	10,465
NAVIGATION	H14	5	11	7	23	45,0980	5,3488
	H15	3	4	4	11	21,5686	2,5581
	H16	0	1	5	6	11,7647	1,3953
	H17	1	0	1	2	3,92156	0,4651
	H18	4	5	0	9	17,6470	2,0930
	Total	13	21	17	51	100	11,860
PERCEPTION	H19	5	5	0	10	5,71428	2,3255

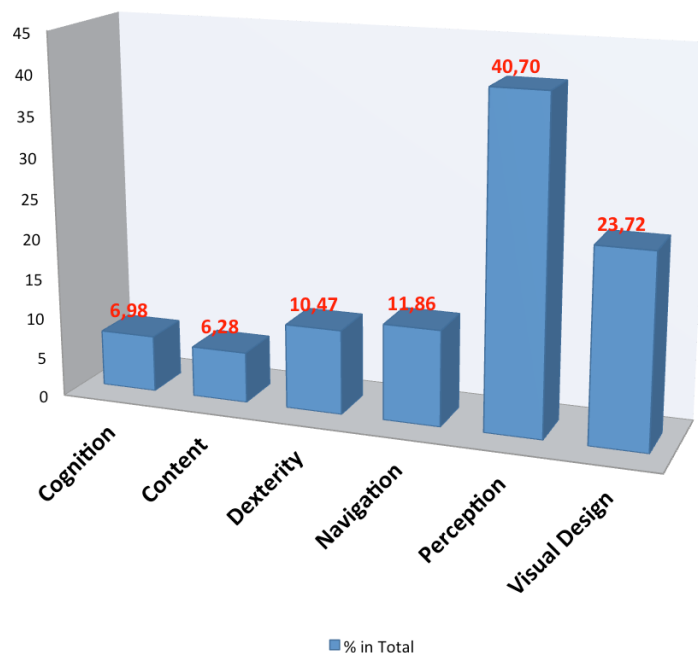
	H20	1	0	0	1	0,57142	0,2325
	H21	16	11	2	29	16,5714	6,7441
	H22	11	8	4	23	13,1428	5,3488
	H23	5	3	0	8	4,57142	1,8604
	H24	9	12	12	33	18,8571	7,6744
	H25	18	17	0	35	20	8,1395
	H26	15	18	3	36	20,5714	8,3720
	Total	80	74	21	175	100	40,697
VISUAL DESIGN	H27	0	4	4	8	7,84313	1,8604
	H28	4	0	0	4	3,92156	0,9302
	H29	9	15	4	28	27,4509	6,5116
	H30	12	14	4	30	29,4117	6,9767
	H31	0	0	1	1	0,98939	0,2325
	H32	1	0	0	1	0,98939	0,2325
	H33	0	1	1	2	1,96078	0,4651
	H34	2	0	1	3	2,94117	0,6976
	H35	10	6	9	25	24,5098	5,8139
	Total	38	40	24	102	100	23,720
Total by APP	167	181	82	430	100	100	

Tabla 6 – Datos completos del estudio, por aplicación y categoría

Las heurísticas están organizadas en diferentes categorías y el número de errores o vulnerabilidades de cada categoría ayuda a entender los problemas de usabilidad y experiencia de usuario de la aplicación.

La figura 103 muestra diferentes gráficas, a modo de resumen, con el número total de vulnerabilidades o errores de cada categoría para las diferentes aplicaciones fitness evaluadas (Endomondo, Runtastic y CloudFit).

La categoría con más errores es la de percepción, con el 40,7% de los errores, relacionada con aspectos de interacción con el usuario o la falta de éstos (volumen, colores, contacto táctil, etc.).



Summary Fitness Applications by Categories

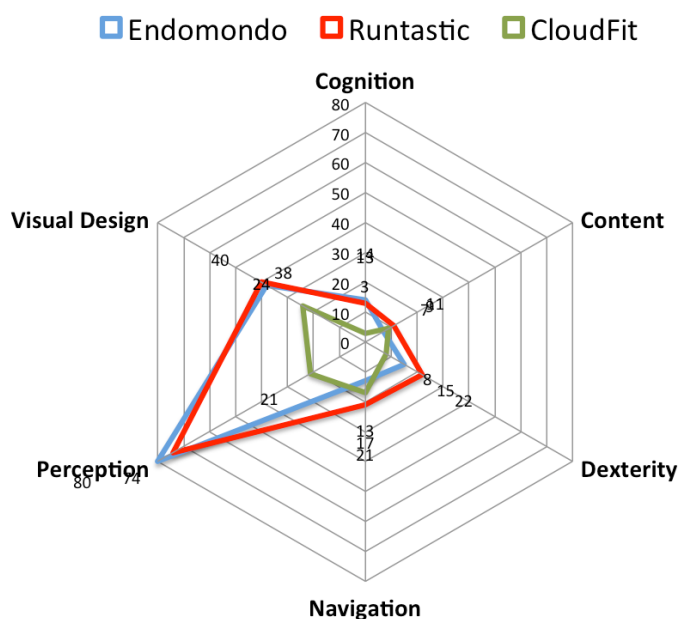


Figura 103 – Número de errores por categoría y aplicación

Gracias a este estudio la aplicación presentada, CloudFit, se ha mejorado significativamente gracias a los resultados obtenidos de los experimentos realizados a Endomondo y Runtastic, además de los clásicos comentarios de los evaluadores, que de primera mano opinaban como usuarios sobre la usabilidad y experiencia de usuario y como debían mejorar las aplicaciones que estaban evaluando.

Las principales mejoras que se han realizado en CloudFit gracias al estudio son:

- Diseño e implementación de la vista/interfaz de configuración. La primera versión de la aplicación no tenía esta vista implementada, pero considerando que la mayoría de las aplicaciones tiene un espacio de configuración de aplicación, además de que se hizo necesario para poder evaluar la aplicación (ya que todas las apps debían evaluarse en las mismas vistas), se implementó para el estudio.
- Posibilidad de ajustar el tamaño de letra, brillo y volumen de cada vista de la aplicación.
- Rediseño de algunas vistas para limitar el número de colores y ajustes visuales para reducir el número de errores.
- Rediseño de ciertos iconos para hacer la interfaz más intuitiva.
- Opciones de configuración (vibración y sonido) cuando el valor del pulso cardíaco del atleta sobrepasa ciertos umbrales.
- Se añadió un tutorial para explicar los distintos ítems presentes en la pantalla de entrenamiento.

La figura 104 muestra la interfaz de la aplicación antes de realizar el estudio y la figura 105 las distintas vistas una vez finalizado el estudio.

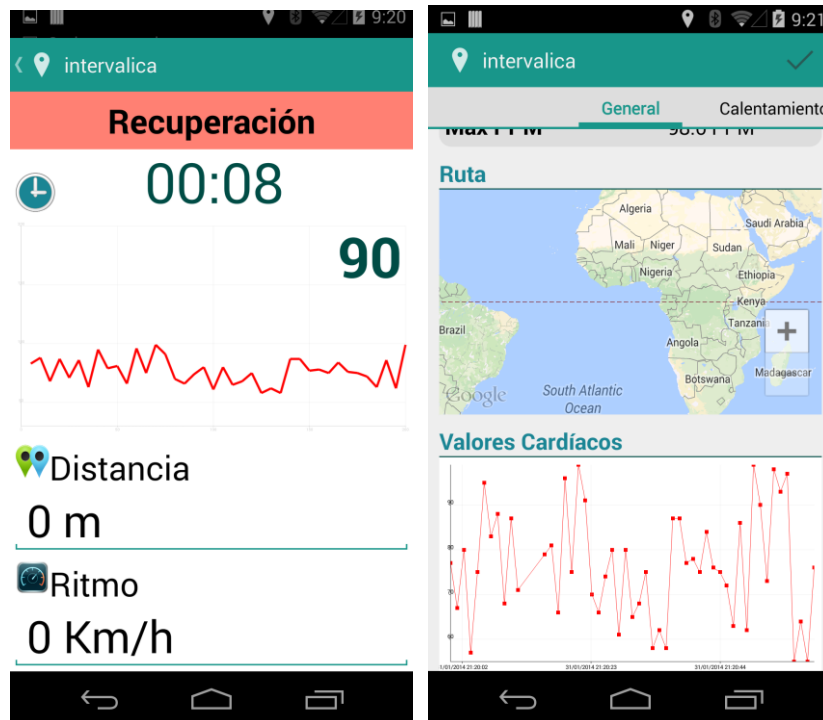


Figura 104 – Interfaz de CloudFit antes de aplicar el estudio

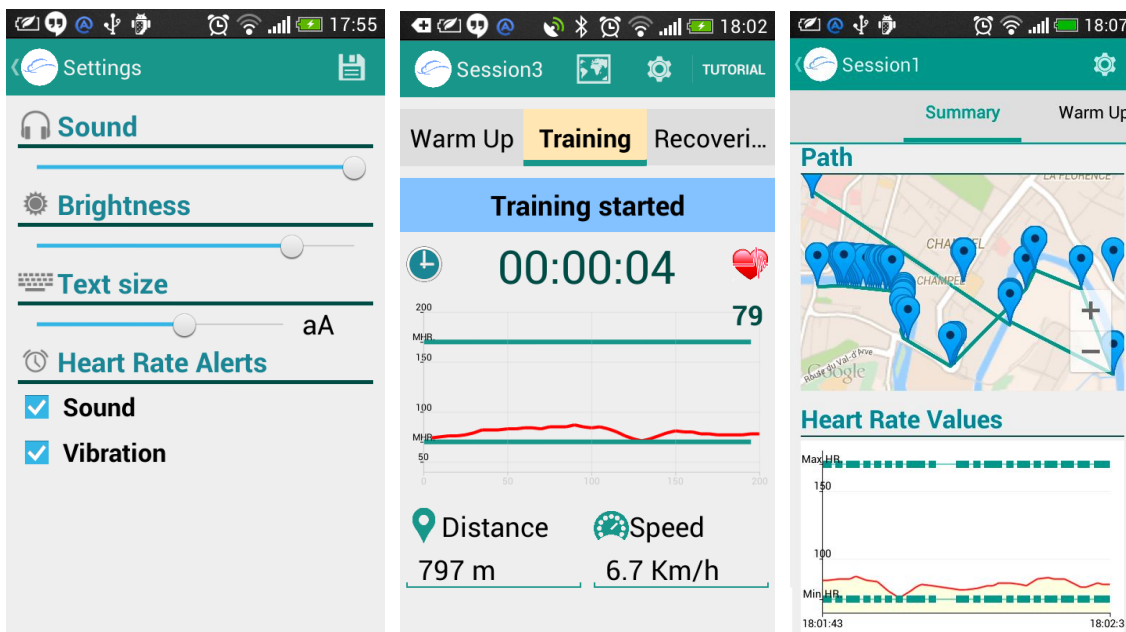


Figura 105 – Interfaz de CloudFit después de aplicar el estudio

Aclarar que, debido a la diferencia entre estas capturas y las mostradas en la figura 93, este estudio se realizó en la versión 1.0 de la aplicación y el estado actual corresponde con la versión 2.0.

Aunque el estudio se realizó para la versión 1.0, en el nuevo diseño de la aplicación para la versión 2.0 se mantuvieron conceptos aprendidos de este estudio para no cometer errores que empeorasen la usabilidad y experiencia del usuario.

Discusión

Como muestran los resultados de la tabla 6, Endomondo y Runtastic obtuvieron resultados muy similares en términos de número de errores (la media para Endomondo fue 28.5 y 30.5 para Runtastic). Esto probablemente es debido al hecho de que Endomondo y Runtastic fueron creadas con un claro objetivo comercial, donde los diseñadores siguieron guías de diseño orientadas a generar productos que obtuvieron el máximo número de descargas y usuarios activos.

Si extendemos este estudio y evaluamos otras aplicaciones comerciales (Nike+, Runkeeper, SportTracker, etc) probablemente obtendremos unos resultados similares ya que todas ellas tienen interfaces, funcionalidades y diseños similares.

El principal objetivo en el diseño de CloudFit es que sea usada por supervisores y usuarios monitorizados (atletas) con el objetivo de mejorar su bienestar y sus hábitos saludables.

Debido a que actualmente CloudFit está orientada a aspectos de investigación (no una aplicación comercial), es posible incluir cambios como la posibilidad de cambiar texto o el volumen, reducir el número de colores o cambiar los iconos con el objetivo de reducir al máximo el número de errores de acuerdo a la heurística del método de evaluación (14 errores).

Es por esto crucial determinar el enfoque de aplicación (comercial, investigación) donde vamos a aplicar el método de evaluación (para medir la usabilidad y experiencia del usuario) con el objetivo de entender los resultados finales. Además, independientemente del enfoque de la aplicación, toda la información del estudio que se procesa y presenta puede revelar interesantes resultados sobre el porqué una aplicación tiene más usabilidad que otras.

El alto número de errores en ciertas categorías es un buen estimador para determinar la usabilidad y experiencia del usuario. Como muestra la figura 100, Endomondo y Runtastic tienen el mayor número de errores en la misma categoría (Percepción), seguido de diseño visual y navegación. De hecho, se obtienen los mismos resultados en el estudio realizado en [Silva 2014], donde RunkeePer y Nike+ son evaluadas y se obtienen los mayores niveles de errores en estas tres categorías.

Esto revela un número de deficiencias debido principalmente al enfoque comercial, donde los diseñadores se centraron más en un aspecto totalmente estético de la aplicación para atraer el mayor número de clientes, en vez de tener una buena usabilidad para atraer al mayor número de usuarios por su buena experiencia de usuario.

En general, los resultados obtenidos son muy esclarecedores para entender la naturaleza de la usabilidad y experiencia de usuarios en aplicaciones fitness. Con una correcta interpretación de los resultados del estudio, más allá del resumen del mismo, estos resultados pueden ser muy útiles para mejorar la usabilidad y experiencia del usuario con el objetivo de crear mejor aplicaciones fitness.

Capítulo 9

Conclusiones y Trabajo Futuro

9.1. Conclusiones

Las plataformas de telemonitorización permiten el seguimiento de la actividad de unos usuarios (*usuarios monitorizados*) por parte de otros (*usuarios supervisores*), eliminando la necesidad de que el control y observación de dicha actividad deba realizarse con ambos tipos de usuarios en un mismo sitio y lugar. Al desarrollo y popularidad de este tipo de plataformas han contribuido los avances en las tecnologías de *wearable* y *cloud computing*, haciendo que los dispositivos de medición y registro de la actividad sean más portables y cómodos en el primer caso, y habilitando el acceso a la información desde cualquier lugar mediante aplicaciones más robustas, en el segundo.

No obstante, el estado incipiente de las tecnologías mencionadas conlleva la concurrencia de ciertos retos o dificultades a la hora de abordar el desarrollo de este tipo de plataformas y que dificultan este proceso en sí mismo complejo. Entre otras dificultades destacan la integración de *wearables* en plataformas de telemonitorización o la heterogeneidad tecnológica presentes en una plataforma de telemonitorización.

A ello se une el hecho de que la mayoría de las soluciones existentes hasta la fecha y en la mayoría de los ámbitos, han sido desarrolladas siguiendo enfoques ad-hoc, es decir, sin guías o pautas claras que marquen u orienten estos complejos procesos de desarrollo de forma sistemática.

Esta tesis se enmarca en el contexto del desarrollo de las plataformas de telemonitorización. Particularmente, se ha llevado a cabo un proceso de investigación para identificar las principales dificultades de este proceso y proveer técnicas, mecanismos y herramientas para simplificarlo.

La principal aportación de la tesis se concreta en la definición de e-MoDe, un Framework para sistematizar y dar soporte al proceso de desarrollo de plataformas de telemonitorización.

e-MoDe está formado por tres elementos:

1. Una metodología de desarrollo.
2. Un conjunto de modelos de soporte al diseño.
3. Un conjunto de herramientas de soporte a la implementación.

El Framework permite sistematizar el desarrollo de la plataforma y simplificar el diseño e implementación de la misma, de tal manera que el proceso de desarrollo esté guiado y que, a través de la metodología, modelos y herramientas que ofrece el framework, se cumplan ciertos requisitos en la plataforma desarrollada, tanto a nivel de supervisión remota (telemonitorización) (evitar desplazamientos, reducción de costes, evidencia práctica) como a nivel tecnológico (reutilización de código, extensibilidad de la plataforma).

Las diferentes contribuciones que se han realizado para la creación del Framework son las siguientes:

- Definición de una nueva metodología para orientar/guiar el desarrollo de plataformas de telemonitorización, basada en cinco etapas, y donde se especifica en cada etapa los objetivos, los actores que intervienen y el resultado de la misma. Esta metodología permite orientar el desarrollo de una plataforma de telemonitorización, desde la primera reunión de equipo hasta el despliegue de la plataforma para su uso. Esta propuesta comparte

similitudes con otras metodologías tradicionales (cascada), pero difiere en la forma de abordar cada etapa, donde se hace especial hincapié en satisfacer diferentes requisitos (necesidades especiales de los usuarios, extensibilidad, tareas de telemonitorización, etc.).

- Propuesta de un método adicional para simplificar la integración y uso de *wearables* a través de la definición de nuevos roles para el diseño e interacción con estos dispositivos. En el método actual, un programador se encarga de realizar todas las tareas, complicando el proceso de integración e interacción con el *wearable* por el conocimiento que se requiere del mismo y las tecnologías necesarias. En este nuevo método se propone una división de las tareas por roles, definiendo el rol de diseñador, que será quien modele el *wearable*, y el programador el encargado de interactuar con él.
- Diseño e implementación de un conjunto de herramientas llamado *WearIt*, para simplificar la integración y uso de *wearables* en plataformas de telemonitorización. *WearIt*, que hace uso del método propuesto para la integración de *wearables*, está basada en diferentes herramientas o artefactos software que forman parte de la contribución de la tesis:
 - Metamodelo: Un metamodelo que permite representar las características de los *wearables* y la posibilidad de especificar cómo interactuar con ellos. De este metamodelo se generan instancias que corresponden con modelos para representar las características y el acceso a las funcionalidades de los *wearables*.
 - *Wearable Markup Language (WML)*: El nombre dado a los diferentes elementos en el metamodelo, en las instancias de éste (modelos de los *wearables*) se corresponden con las etiquetas de un lenguaje de marcas propuesto y definido en esta tesis llamado *Wearable Markup Language (WML)*. Este lenguaje permite representar las características de un *wearable* y especificar cómo interactuar con él a través de APIs de alto nivel.
 - Generación de modelos: Puesto que a partir del metamodelo es necesario generar diferentes modelos en WML, uno por cada *wearable*, se han creado servicios para llevar a cabo dicha transformación entre modelos. Dichos servicios usan un procesador XLST para, a partir del metamodelo y de una hoja de estilos en XLST, generar el modelo de un *wearable* para interactuar con él.
 - Herramienta de diseño: De acuerdo a la metodología propuesta, el diseñador es el encargado de modelar el *wearable*, esto es, especificar sus características y cómo acceder a sus funcionalidades. Para dar soporte a esta tarea, se ha diseñado e implementado una herramienta web que permite al diseñador, a través de formularios y de forma sencilla e intuitiva, modelar el *wearable*. Esta herramienta genera una hoja de estilos que, junto con el procesador para la generación de modelos, genera el modelo del *wearable*.
 - Coordinador: Se trata de un componente software que permite al programador interactuar con un *wearable* a partir de su modelo en WML. Este componente usa el modelo del *wearable* para transformarlo en artefactos softwares y, a través de su API, permitir al programador integrar el *wearable* en otras aplicaciones.
 - Algoritmo "*wearable payload*": Este algoritmo permite procesar la información procedente de un *wearable* y obtener la información relevante. El coordinador es el encargado de diferentes funciones, desde conectar con el *wearable* hasta devolver la

información relevante (*payload*) al programador. Internamente el coordinador usa un algoritmo desarrollado *ex profeso* para asegurar que la información devuelta es la correcta.

- Desarrollo de Zappa, una plataforma orientada a componentes que permite a los programadores reutilizar funcionalidades naturalmente presentes en las plataformas de telemonitorización (gestión de datos internos, disseminación estadística de los datos, acceso a los servicios de la arquitectura software, etc.).
- Desarrollo de gAndroid, un componente software para el acceso a servicios de la tecnología G, una tecnología *cloud* de la compañía Gnúbila-Indra, sin necesidad de conocimientos sobre dicha tecnología.
- Definición de una propuesta para el soporte a la colaboración en una plataforma de telemonitorización. Esta propuesta está formada por un modelo donde se identifican los diferentes elementos presentes en un entorno colaborativo (actores, interacciones, tipos de interacciones, etc.), junto con la arquitectura software y pautas necesarias para llevar a cabo su implementación.
- Diseño de un modelo para representar tareas de telemonitorización. Además, se presenta una propuesta para llevar a la práctica dicho modelo a través del patrón MVC, permitiendo así la gestión de tareas de telemonitorización, facilitando la adición de nuevas tareas.

Con objeto de validar las contribuciones anteriores se han desarrollado dos plataformas de telemonitorización aplicando el framework e-MoDe:

- CloudRehab: Se trata de plataforma de telerehabilitación para pacientes con daño cerebral que pretende, mediante un protocolo de telerehabilitación, mejorar las capacidades cognitivas de los diferentes usuarios. CloudRehab está formada por una aplicación web, una aplicación móvil, los diferentes servicios *cloud* (basados en la tecnología G) y el uso de una banda de pulso cardíaco. Esta plataforma ha sido desarrollada en colaboración con psicólogos de la Universidad de Granada y terapeutas del Hospital de Rehabilitación y Traumatología de Granada.
- CloudFit: Consiste en una plataforma multidisciplinar para la gestión de la salud y bienestar de los usuarios, que se centra en la gestión de entrenamientos deportivos. La plataforma consta de una plataforma web, una aplicación Android, una aplicación iOS, diferentes servicios cloud soportados por Google Cloud y diferentes *wearables*. CloudFit ha sido desarrollada con la colaboración de los expertos del Instituto Mixto de Deporte y Salud (IMUDS).

9.2. Trabajo Futuro

La complejidad del proceso de desarrollo de plataformas de telemonitorización hace que puedan surgir numerosas líneas de investigación a partir de las contribuciones planteadas.

Ejemplos de estas líneas de investigación son:

- Abordar la gestión de datos en una plataforma de telemonitorización. Puesto que en una plataforma de telemonitorización se genera gran cantidad de información, se hace necesario definir pautas, modelos y herramientas que permitan no sólo la gestión eficiente

de esta información, sino que los modelos de datos estén preparados para adaptarse a cambios en el protocolo de telemonitorización (nuevas tareas, nuevos roles o nuevas funcionalidades implican nuevos modelos de datos).

- Unido a esta gestión de datos hay otro aspecto cada vez más demandado en plataformas de telemonitorización: privacidad y seguridad. La privacidad y seguridad de la información en una plataforma de telemonitorización resultan cruciales, puesto que se gestionan datos personales de usuarios con información sensible. Se hace necesario abordar estudios para determinar la mejor manera de garantizar la privacidad y seguridad en una plataforma de telemonitorización, y añadir al Framework las propuestas resultantes para que una plataforma desarrollada con dicho Framework asegure unas garantías mínimas respecto a la seguridad y privacidad de los datos.
- El soporte a la decisión en las plataformas de telemonitorización es otro concepto cada vez con más presencia en este ámbito. Esto es, plataformas de telemonitorización donde no sea estrictamente necesaria la intervención de un usuario supervisor que dictamine la evolución de un usuario monitorizado o que incluso planifique la telemonitorización. Varias plataformas ya incorporan sistemas expertos basados en inteligencia artificial para la toma de decisiones, donde a partir de los datos que va generando el usuario monitorizado con su telemonitorización, la plataforma va guiando al usuario a lo largo del tiempo. Esta línea de investigación resulta muy interesante por las ventajas que adquiere la plataforma de telemonitorización. No obstante, sería necesario abordar un estudio completo sobre sistemas expertos e inteligencia artificial y poder añadir así propuestas al Framework que ayuden a los ingenieros y programadores a dotar de dichas características a las plataformas desarrolladas.
- Validación objetiva de plataformas de telemonitorización. Como se ha comentado en la definición de la metodología, esta etapa de validación depende de la plataforma de telemonitorización y dicha etapa se deja a expensas de los ingenieros y expertos en el área de telemonitorización. En este sentido, sería interesante dotar al Framework de las propuestas necesarias para que esta validación sea lo más objetiva posible: a través de cuestionarios, pruebas de validación concretas, sistemas de puntuaciones donde se garantice una puntuación mínima, etc. Esto permitiría, de manera objetiva, medir el nivel de validación de dos plataformas distintas, y comparar así la calidad de las propuestas.

Chapter 9

Conclusions and Future Work

9.1. Conclusions

Telemonitoring platforms allow the activity monitoring of some users (monitored users) by other users (supervisors), avoiding that the supervision of the activity must be performed by both types of users in the same place and at the same time. Wearable computing and cloud computing have contributed to the rise and popularity of these platforms, enabling that measurement devices and activity logger being more portable and comfortable in the first case, and enabling the access to the information from anywhere through robust applications, in the second one.

However, the growing state of these technologies entails challenges or difficulties when addressing the development of telemonitoring platforms, complicating an already complex process.

In addition, most solutions to date and in most areas have been developed following an ad-hoc approach, that is, without a guideline to address these complex processes in a systematic way.

This thesis is framed in the context of the development of telemonitoring platforms. A research process has been carried out in order to identify the main challenges of this development process and to provide techniques, mechanisms and tools to simplify it.

The main contribution of the thesis is the definition of e-MoDe, a framework to systematize and provide support to the development process of telemonitoring platforms.

e-MoDe is composed by three elements:

1. A methodology to guide the development.
2. Set of models to support the software design of the platform.
3. Set of tools to simplify the implementation of the platform.

e-MoDe allows to systematize the development of a telemonitoring platform and simplify the software design and implementation, so the development process is guided and, through the methodology, models and tools, certain requirements are met on the platform developed. These requirements are related with the remote supervision (telemonitoring, reduce displacements, saving costs) and the technology (reuse functionalities/code, extensibility).

The main contributions that have been made for the creation of the framework (e-MoDe) are the following:

- Definition of a new methodology to guide the development of telemonitoring platforms, from the first team meeting to the deploy of the platform. This methodology is based on five different stages and specifies the goals, involved actors and results of each stage. This proposal shares similarities with traditional methodologies (waterfall), but differ in the way to address each stage, where special emphasis is made to satisfy different requirements (special needs of users, extensibility, etc.).
- Proposal of an additional method to simplify the integration and use of wearables, through the definition of new roles for the design and interaction with these devices. In the current method, the programmer performs all the tasks, complicating the use of the wearable because the extensive knowledge required about the device and technologies.

This new method proposes the division of tasks in different roles, defining a new role called designer, who model the wearable, and the programmer, who use the wearable.

- Design and implementation of a set of tools to simplify the integration and use of wearable in telemonitoring platforms, called WearIt. WearIt, which uses the new method proposed to integrate wearables, is based in different tools or software artefact as contributions of this thesis:
 - Metamodel: A metamodel to represent the wearable's features and the possibility to specify how to interact with them. The metamodel instances correspond with wearables models that represent the features and how to access to the functionalities of a wearable.
 - Wearable Markup Language (WML): The name given to the different elements of the metamodel and used in the instances of this metamodel (wearables models) correspond with tags of a new markup language proposed and defined in this thesis, called Wearable Markup Language (WML). This language allows represent features of a wearable and how to interact with it through high-level APIs.
 - Model generation: To generate models from the metamodel, one per each wearable, several services have been created to carry out this transformation. These services use a XSLT process, which uses the metamodel and a stylesheet specified in XSLT, to generate a new wearable model.
 - Design Tool: According to the methodology proposed, the designer models the wearable, that is, specify the features and how to access to the different functionalities. To provide support to this task, a web tool has been designed and implemented, allowing to the designer, through forms and in an intuitive way, model a wearable. This tool generates a stylesheet to be used by the processor to generate the model of a wearable.
 - Coordinator: A software component that allows to programmer to interact with a wearable from its model in WML. This component uses the model of a wearable to generate internal software elements (objects) and, through its API, allow to programmer to interact with the wearable.
 - Wearable payload Algorithm: This algorithm enables to process the information provided by a wearable and obtain the relevant information, that is, the payload. The coordinator uses this algorithm to ensure the information provided by a wearable and returned to a programmer is correct.
- Development of Zappa, a component-software oriented platform that allows to programmer reuse functionalities, naturally presents in telemonitoring platforms (internal data management, statistical data dissemination, access to web services, etc.), in order to simplify the implementation.
- Development of gAndroid, a software component to access to cloud services based on G technology (a property cloud technology by Gnubila-Indra), without technology background.
- Definition of a new proposal to support collaborative scenarios in a telemonitoring platform. This proposal is composed by a model, where the different collaborative

elements are identified (actors, interactions, types of interactions, etc.), along the software architecture and necessary guidelines to carry out the implementation.

- Design a model to represent telemonitoring tasks. In addition, a proposal to implement this model through MVC pattern is presented, allowing the management of telemonitoring tasks in a technological independent way, facilitating the addition of new telemonitoring tasks.

In order to validate the above contributions, two different telemonitoring platforms have been developed using e-MoDe framework:

- CloudRehab: A telerehabilitation platform for brain-injured patients that intends to improve the cognitive abilities of the different users through a telemonitoring protocol. CloudRehab is composed by a web application, an Android application, different cloud services (based on G technology) and the use of a heart rate band. This platform has been developed with the collaboration of psychologist of the University of Granada and therapist of “Hospital de Rehabilitación y Traumatología de Granada”.
- CloudFit. A multidisciplinary platform to promote health and wellness of users, which focuses on the management of sport trainings. The platform is composed by a web platform, an Android application, an iOS application, different cloud services supported by Google Cloud and the use of different wearables. CloudFit has been developed with the collaboration of experts from “Instituto Mixto de Deporte y Salud (IMUDS)”.

9.2. Future Work

The complexity of the development process of telemonitoring platform cause that there are many research lines from the contributions proposed in this thesis.

Some of these research lines are:

- Address the data management in a telemonitoring platform. Because a lot of information is generated in a telemonitoring platform, it is necessary to define guides, models and tools that allow to manage this information in an efficient way, and data models prepared to adapt to changes in the telemonitoring protocol (new tasks, new roles or new functionalities entails new data models).
- In addition to this data management, there is another concern increasingly demanded in the telemonitoring platforms: privacy and security, because sensitive information is managed. It is necessary to address a research to define the best way to ensure the privacy and security in a telemonitoring platform, and add to the framework (e-MoDe) the resulting proposal. This entails that a telemonitoring platform developed using e-MoDe, ensure a minimum guarantee about privacy and security of the information.
- The decision support used in telemonitoring platforms is another concept increasingly presented in this area. That is, telemonitoring platform where is not strictly necessary the intervention of a supervisor to determine the evolution of a monitored user. Several platforms use expert systems based on artificial intelligence to take decisions, where from the data generated by the monitored user the platform guides the telemonitoring process. This research line is really interesting because of the advantages acquired by a

telemonitoring platform. However, it would be necessary to address a full research about expert's system and artificial intelligence in order to design a solution that supports engineers and programmer to provide these features to a telemonitoring platform.

- Objective evaluation of telemonitoring platforms. In the evaluation stage of the methodology proposed, this stage is carried out by engineers and domain experts of the telemonitoring platform. In this way, could be interesting provide to the framework several proposals to guarantee that this validation is as objective as possible: through polls, concrete validation tests, score system, etc. This allow, in an objective way, determine the validation of different telemonitoring platforms and compare the quality of the proposals.

Bibliography

[Adams 1992] Adams, D. A., Nelson, R. R., & Todd, P. A. (1992). Perceived usefulness, ease of use and usage of information technology: A replication. *MIS quarterly*, 227-247.

[Akbal-Delibas 2009] Akbal-Delibas, B., Boonma, P., & Suzuki, J. (2009). Extensible and precise modeling for wireless sensor networks. In *Information Systems: Modeling, Development, and Integration* (pp. 551-562). Springer Berlin Heidelberg.

[Allen 1990] Allen, A., Doolittle, G. C., Boysen, C. D., Komoroski, K., Wolf, M., Collins, B., & Patterson, J. D. (1999). An analysis of the suitability of home health visits for telemedicine. *Journal of Telemedicine and Telecare*, 5(2), 90-96.

[Alonso 2011] Alonso, R. S., Tapia, D. I., Bajo, J., & Rodríguez, S. (2011). Improving a telemonitoring system based on heterogeneous sensor networks. In *Advances in Computational Intelligence* (pp. 661-668). Springer Berlin Heidelberg.

[AmazonEC2] aws.amazon.com/ec2

[Ambrust 2010] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., ... & Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53(4), 50-58.

[Anton 2013] Anton, D., Goni, A., Illarramendi, A., Torres-Unda, J. J., & Seco, J. (2013, October). KiReS: A Kinect-based telerehabilitation system. In *e-Health Networking, Applications & Services (Healthcom), 2013 IEEE 15th International Conference on* (pp. 444-448). IEEE.

[Aranki 2014] Aranki, D., Kurillo, G., Yan, P., Liebovitz, D. M., & Bajcsy, R. (2014, September). Continuous, real-time, tele-monitoring of patients with chronic heart-failure: lessons learned from a pilot study. In *Proceedings of the 9th International Conference on Body Area Networks* (pp. 135-141). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Azure] <https://azure.microsoft.com/>

[Beimborn 2011] Beimborn, D., Miletzki, T., & Wenzel, S. (2011). Platform as a service (PaaS). *Business & Information Systems Engineering*, 3(6), 381-384.

[Bellifemine 2011] Bellifemine, F., Fortino, G., Giannantonio, R., Gravina, R., Guerrieri, A., & Sgroi, M. (2011). SPINE: a domain-specific framework for rapid prototyping of WBSN applications. *Software: Practice and Experience*, 41(3), 237-265.

[Benyahia 2013] Benyahia, A. A., Hajjam, A., Hilaire, V., Hajjam, M., & Andres, E. (2013, July). E-care telemonitoring system: extend the platform. In *Information, Intelligence, Systems and Applications (IISA), 2013 Fourth International Conference on* (pp. 1-4). IEEE.

[Bhardwaj 2010] Bhardwaj, S., Jain, L., & Jain, S. (2010). Cloud computing: A study of infrastructure as a service (IAAS). *International Journal of engineering and information Technology*, 2(1), 60-63.

[Bianco 2007] Bianco, P., Kotermanski, R., & Merson, P. F. (2007). Evaluating a service-oriented architecture.

- [Bisio 2015] Bisio, I., Lavagetto, F., Marchese, M., & Sciarrone, A. (2015). A smartphone-centric platform for remote health monitoring of heart failure. *International Journal of Communication Systems*, 28(11), 1753-1771.
- [Boehm 1988] Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, 21(5), 61-72.
- [Boyne 2009] Boyne, J. J., & Vrijhoef, H. J. (2013). Implementing telemonitoring in heart failure care: barriers from the perspectives of patients, healthcare professionals and healthcare organizations. *Current heart failure reports*, 10(3), 254-261.
- [Broens 2007] Broens, T. H., Vollenbroek-Hutten, M. M., Hermens, H. J., van Halteren, A. T., & Nieuwenhuis, L. J. (2007). Determinants of successful telemedicine implementations: a literature study. *Journal of telemedicine and telecare*, 13(6), 303-309.
- [Brown 1998] Brown, N., & Kindel, C. (1998). Distributed component object model protocol—dcom/1.0. *Online*, November.
- [Büher 1999] Bühler, C., & Knops, H. (1999). Social alarms go mobile: emergency assistance for mobile users. *Assistive Technology on the Threshold of the New Millennium*, 6, 106.
- [Burgstahler 2014] Burgstahler, D., Richerzhagen, N., Englert, F., Hans, R., & Steinmetz, R. (2014, June). Switching Push and Pull: An Energy Efficient Notification Approach. In *Mobile Services (MS), 2014 IEEE International Conference on* (pp. 68-75). IEEE.
- [Butler 2001] Butler, G. (2001, June). Object Oriented Frameworks. In *15th European Conference on Object-Oriented Programming, Tutorial Budapest* (pp. 1-70).
- [Butz 2006] Butz, A., & Krüger, A. (2006, November). User-centered development of a pervasive healthcare application. In *Pervasive Health Conference and Workshops, 2006* (pp. 1-8). IEEE.
- [Buxxman 2008] Buxmann, P., Hess, T., & Lehmann, S. (2008). Software as a Service. *Wirtschaftsinformatik*, 50(6), 500-503.
- [Buyya 2009] Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6), 599-616.
- [Calabrese 2015] Calabrese, B., & Cannataro, M. (2015). Cloud computing in healthcare and biomedicine. *Scalable Computing: Practice and Experience*, 16(1), 1-18.
- [Campbell 1992] Campbell, R. H., Islam, N., & Madany, P. (1992). Choices, frameworks and refinement. *Computing Systems*, 5(3), 217-257.
- [Carr 2012] Carr, D., O'Grady, M. J., O'Hare, G. M., & Collier, R. (2012). SIXTH: a middleware for supporting ubiquitous sensing in personal health monitoring. In *Wireless Mobile Communication and Healthcare* (pp. 421-428). Springer Berlin Heidelberg.
- [Castillejo 2013] Castillejo, P., Martinez, J. F., Rodriguez-Molina, J., & Cuerva, A. (2013). Integration of wearable devices in a wireless sensor network for an E-health application. *Wireless Communications, IEEE*, 20(4), 38-49.

- [Chaudhry 2010] Chaudhry, S. I., Mattera, J. A., Curtis, J. P., Spertus, J. A., Herrin, J., Lin, Z., ... & Krumholz, H. M. (2010). Telemonitoring in patients with heart failure. *New England Journal of Medicine*, 363(24), 2301-2309.
- [Choudhary 2007] Choudhary, V. (2007, January). Software as a service: Implications for investment in software development. In *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* (pp. 209a-209a). IEEE.
- [Crespo 2010] L. Crespo, D. S. Morillo, M. Crespo, A. Leon, S. Astorga, K. Giorkas, and I. Kouris, "Telemonitoring in AMICA: A design based on and for COPD," in *Information Technology and Applications in Biomedicine (ITAB)*, 2010 10th IEEE International Conference on. IEEE, 2010, pp. 1–6.
- [Curbera 2002] Curbera, F., Duftler, M., Khalaf, R., Nagy, W., Mukhi, N., & Weerawarana, S. (2002). Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. *IEEE Internet computing*, 6(2), 86.
- [Davis 1989] Davis, F. D., Bagozzi, R. P., & Warshaw, P. R. (1989). User acceptance of computer technology: a comparison of two theoretical models. *Management science*, 35(8), 982-1003.
- [Democare web] <http://www.demcare.eu/>
- [Depari 2013] Depari, A., Flammini, A., Rinaldi, S., & Vezzoli, A. (2013). Multi-sensor system with Bluetooth connectivity for non-invasive measurements of human body physical parameters. *Sensors and Actuators A: Physical*, 202, 147-154.
- [Diab 2013] Diab, M. O., Marak, R. A., Dichari, M., & Moslem, B. (2013, January). The smartphone accessory heart rate monitor. In *Computer Medical Applications (ICCMA)*, 2013 International Conference on (pp. 1-5). IEEE.
- [Diaz 2010] Díaz, U., García, Á., & De Felipe, A. (2010, June). A new tourist audio guide service for elderly people integrated in the Mobile Phone: Preliminary results. In *Proceedings of the 3rd International Conference on Pervasive Technologies Related to Assistive Environments* (p. 17). ACM.
- [Dimmick 2000] Dimmick SL, Mustaleski C, Burgiss SG, et al. A case study of benefits and potential savings in rural home telemedicine. *Home Health Nurse* 2000; 18:125–134.
- [Erl 2005] Erl, T. (2005). *Service-oriented architecture: concepts, technology, and design*. Pearson Education India.
- [Eskofier 2008] Eskofier, B., Hartmann, E., Kühner, P., Griffin, J., Schlarb, H., Schmitt, M., & Hornegger, J. (2008). Real time surveying and monitoring of athletes using mobile phones and GPS. *International Journal of Computer Science in Sport*, 7(1), 18-27.
- [Fayad 1997] Fayad, M., & Schmidt, D. C. (1997). Object-oriented application frameworks. *Communications of the ACM*, 40(10), 32-38.
- [Fayad 1999] Fayad, M., Schmidt, D. C., & Johnson, R. E. (1999). *Building application frameworks: object-oriented foundations of framework design*. Wiley.
- [Fielding 2000] Fielding, R. T. (2000). *Architectural styles and the design of network-based software architectures* (Doctoral dissertation, University of California, Irvine).

[Finkelstein 1992] Finkelstein, S. M., Lindgren, B., Prasad, B., Snyder, M., Edin, C., Wielinski, C., & Hertz, M. (1992). Reliability and validity of spirometry measurements in a paperless home monitoring diary program for lung transplantation. *Heart & lung: the journal of critical care*, 22(6), 523-533.

[FitMacro Web] <http://www.fitmacro.com/>

[Forman 1994] Forman, G. H., & Zahorjan, J. (1994). The challenges of mobile computing. *Computer*, 27(4), 38-47.

[Fortino 2013] Fortino, G., Giannantonio, R., Gravina, R., Kuryloski, P., & Jafari, R. (2013). Enabling effective programming and flexible management of efficient body sensor network applications. *Human-Machine Systems, IEEE Transactions on*, 43(1), 115-133.

[Gnubila 2015] <http://www.gnubila.com/post298609/once-again-gartner-locates-the-indra-gnubila-s-platform-as-a-service-in-the-magic-quadrant-for-enterprise-application-platform-as-a-service-published-in-march-2015>

[Gnubila Web] www.gnubila.com

[Gong 2011] Gong, J., Lu, S., Wang, R., & Cui, L. (2011, June). PDhms: pulse diagnosis via wearable healthcare sensor network. In *Communications (ICC), 2011 IEEE International Conference on* (pp. 1-5). IEEE.

[GoogleAppEngine] <https://cloud.google.com/appengine/>

[GoogleCloud] <https://cloud.google.com/>

[Gregoski 2012] Gregoski, M. J., Mueller, M., Vertegel, A., Shaporev, A., Jackson, B. B., Frenzel, R. M., ... & Treiber, F. A. (2012). Development and validation of a smartphone heart rate acquisition application for health promotion and wellness telehealth applications. *International journal of telemedicine and applications*, 2012, 1.

[Guinard 2010] Guinard, D., Trifa, V., & Wilde, E. (2010, November). A resource oriented architecture for the web of things. In *Internet of Things (IOT), 2010* (pp. 1-8). IEEE.

[Gusev 2014] Gusev, M., Tasic, J., Tasic, D. R., Patel, S., Patel, D., & Veselinovska, B. (2014). MindGym-IPTV for elderly people. In *Pervasive Computing Paradigms for Mental Health* (pp. 155-164). Springer International Publishing.

[Hadim 2006] Hadim, S., & Mohamed, N. (2006). Middleware: Middleware challenges and approaches for wireless sensor networks. *IEEE distributed systems online*, (3), 1.

[Halvorsen 1996] Halvorsen PA, Kristiansen IS. Radiology services for remote communities: cost minimization study of telemedicine. *BMJ* 1996; 312:1333–1336.

[Heroku] <https://www.heroku.com/>

[Herzog 2002] Herzog, R. (2002) 'Definition and requirements of trial services', *MobiHealth D1.1*

[Hopper 2015] Hopper, L., Joyce, R., Melander, C., Kikhia, B., Karakostas, A., & Savenstedt, S. (2015). Dementia ambient care: a holistic approach to the management of dementia in multiple care settings.

[IBMCloud] <http://www.ibm.com/cloud-computing/>

[Inglis 2010] Inglis, S. (2010). Structured telephone support or telemonitoring programmes for patients with chronic heart failure. *Journal of Evidence-Based Medicine*, 3(4), 228-228.

[Jo 2008] Jo, T. W., You, Y. D., Choi, H., & Kim, H. S. (2008). A bluetooth-UPnP bridge for the wearable computing environment. *Consumer Electronics, IEEE Transactions on*, 54(3), 1200-1205.

[Johnson 1988] Johnson, R. E., & Foote, B. (1988). Designing reusable classes. *Journal of object-oriented programming*, 1(2), 22-35.

[Johnston 2000] Johnston B, Wheeler L, Deuser J, et al. Outcomes of the Kaiser Permanente Tele-Home Health Research Project. *Arch Fam Med* 2000; 9:40–45.

[Jovanov 2005] Jovanov, E., Milenkovic, A., Otto, C., & De Groen, P. C. (2005). A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and rehabilitation*, 2(1), 6.

[Kandukuri 2009] Kandukuri, B. R., Paturi, V. R., & Rakshit, A. (2009, September). Cloud security issues. In *Services Computing, 2009. SCC'09. IEEE International Conference on* (pp. 517-520). IEEE.

[Kaufman 2003] Kaufman, D. R., Patel, V. L., Hilliman, C., Morin, P. C., Pevzner, J., Weinstock, R. S., ... & Starren, J. (2003). Usability in the real world: assessing medical information technologies in patients' homes. *Journal of biomedical informatics*, 36(1), 45-60.

[Keerthika 2013] Keerthika, A., & Ganesan, R. (2013, April). Pervasive health care system for monitoring oxygen saturation using pulse oximeter sensor. In *Information & Communication Technologies (ICT), 2013 IEEE Conference on* (pp. 819-823). IEEE.

[Kiss 2011] Kiss, N., Patai, G., Hanák, P., Lipic, T., Skoda, P., Gjenero, L., ... & Michieli, I. (2011, May). Vital fitness and health telemonitoring of elderly people. In *MIPRO, 2011 Proceedings of the 34th International Convention* (pp. 279-284). IEEE.

[Kleppe 2003] Kleppe, A. G., Warmer, J., Bast, W., & Explained, M. D. A. (2003). The model driven architecture: practice and promise.

[Kondo 2009] Kondo, D., Javadi, B., Malecot, P., Cappello, F., & Anderson, D. P. (2009, May). Cost-benefit analysis of cloud computing versus desktop grids. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on* (pp. 1-12). IEEE.

[Koopman 2014] Koopman, R. J., Wakefield, B. J., Johanning, J. L., Keplinger, L. E., Kruse, R. L., Bomar, M., ... & Mehr, D. R. (2014). Implementing home blood glucose and blood pressure telemonitoring in primary care practices for patients with diabetes: lessons learned. *Telemedicine and e-Health*, 20(3), 253-260.

[Krafzig 2005] Krafzig, D., Banke, K., & Slama, D. (2005). *Enterprise SOA: service-oriented architecture best practices*. Prentice Hall Professional.

[Kumar 2012] Kumar L. (2012). Teleintimation garment:a wearable electronic garment for soldier's status monitoring application .RMUTP. International Conference: Textiles & Fashion 2012 July 3-4, Bangkok Thailand

- [Kumar 2014] Kumar, K. M., & Venkatesan, R. S. (2014, May). A design approach to smart health monitoring using android mobile devices. In *Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on* (pp. 1740-1744). IEEE.
- [Larman 2003] LARMAN, Craig; BASILI, Victor R. Iterative and incremental development: A brief history. *Computer*, 2003, no 6, p. 47-56.
- [Lasierra 2010] Lasierra, N., Alesanco, A., & Garcia, J. (2010, November). Home-based telemonitoring architecture to manage health information based on ontology solutions. In *Information Technology and Applications in Biomedicine (ITAB), 2010 10th IEEE International Conference on* (pp. 1-4). IEEE.
- [Lasierra 2012] Lasierra B. (2012) An ontology-driven architecture for data integration and management in home-based telemonitoring scenarios (PhD Thesis)
- [Latré 2011] Latré, B., Braem, B., Moerman, I., Blondia, C., & Demeester, P. (2011). A survey on wireless body area networks. *Wireless Networks*, 17(1), 1-18.
- [Layouni 2009] Layouni, M., Verslype, K., Sandikkaya, M. T., De Decker, B., & Vangheluwe, H. (2009). Privacy-preserving telemonitoring for ehealth. In *Data and Applications Security XXIII* (pp. 95-110). Springer Berlin Heidelberg.
- [Lee 2006] Lee, G., Tsai, C., Griswold, W. G., Raab, F., & Patrick, K. (2006, April). PmEB: a mobile phone application for monitoring caloric balance. In *CHI'06 Extended Abstracts on Human Factors in Computing Systems* (pp. 1013-1018). ACM.
- [Lettner 2012] Lettner, F., & Holzmann, C. (2012, December). Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. In *Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia* (pp. 118-127). ACM.
- [Lewis 1995] Lewis, T. G. (1995). *Object-oriented application frameworks*. Manning Publications Co..
- [Linington 2011] Linington, P. F., Milosevic, Z., Tanaka, A., & Vallecillo, A. (2011). *Building enterprise systems with ODP: an introduction to open distributed processing*. CRC Press.
- [Lobley 1997] Lobley D. The economics of telemedicine. *J Telemed Telecare* 1997;3:117–125.
- [Losilla 2007] Losilla, F., Vicente-Chicote, C., Álvarez, B., Iborra, A., & Sánchez, P. (2007). Wireless sensor network application development: An architecture-centric mde approach. In *Software Architecture* (pp. 179-194). Springer Berlin Heidelberg.
- [Mair 2000] Mair FS, Haycox A, May C, et al. A review of telemedicine cost-effectiveness studies. *J Telemed Telecare* 2000; 6:38–40.
- [Mann 1997] Mann, S. (1997). Wearable computing: A first step toward personal imaging. *Computer*, 30(2), 25-32.
- [Mann 1998] Mann, S. (1998, May). Wearable computing as means for personal empowerment. In *Proc. 3rd Int. Conf. on Wearable Computing (ICWC)* (pp. 51-59).
- [Marston 2011] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. *Decision support systems*, 51(1), 176-189.

- [Mattson 1996] Mattsson, M. (1996). Object-oriented frameworks. *Licentiate thesis*.
- [McAlister 2004] McAlister, F. A., Stewart, S., Ferrua, S., & McMurray, J. J. (2004). Multidisciplinary strategies for the management of heart failure patients at high risk for admission: a systematic review of randomized trials. *Journal of the American College of Cardiology*, 44(4), 810-819.
- [McCarthy 2013] McCarthy, M. W., James, D. A., & Rowlands, D. D. (2013). Smartphones: Feasibility for real-time sports monitoring. *Procedia Engineering*, 60, 409-414.
- [Mehta 2012] Mehta, B., Rengarajan, D., & Prasad, A. (2012). Real time patient tele-monitoring system using LabVIEW. *International Journal of Scientific and Engineering Research*, 3(4), 435-445.
- [Mell 2011] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing.
- [Meystre 2005] Meystre, S. (2005). The current state of telemonitoring: a comment on the literature. *Telemedicine Journal & e-Health*, 11(1), 63-69.
- [MyFitnessPal Web] <https://www.myfitnesspal.com/>
- [Muñoz 2015] Muñoz, D., Cornejo, R., Gutierrez, F. J., Favela, J., Ochoa, S. F., & Tentori, M. (2015). A social cloud-based tool to deal with time and media mismatch of intergenerational family communication. *Future Generation Computer Systems*, 53, 140-151.
- [Ostojic 2005] Ostojic, V., Cvorisec, B., Ostojic, S. B., Reznikoff, D., Stipic-Markovic, A., & Tudjman, Z. (2005). Improving asthma control through telemedicine: a study of short-message service. *Telemedicine Journal & e-Health*, 11(1), 28-35.
- [Pandor 2013] Pandor, A., Thokala, P., Gomersall, T., Baalbaki, H., Stevens, J. W., Wang, J., ... & Fitzgerald, P. (2013). Home telemonitoring or structured telephone support programmes after recent discharge in patients with heart failure: systematic review and economic evaluation.
- [Panicker 2015] Panicker, N. V., & Kumar, A. (2015, October). Design of a telemonitoring system for detecting falls of the elderly. In *Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on* (pp. 800-803). IEEE.
- [Papazoglou 2007] Papazoglou, M. P., Traverso, P., Dustdar, S., & Leymann, F. (2007). Service-oriented computing: State of the art and research challenges. *Computer*, (11), 38-45.
- [Paré 2007] Paré, G., Jaana, M., & Sicotte, C. (2007). Systematic review of home telemonitoring for chronic diseases: the evidence base. *Journal of the American Medical Informatics Association*, 14(3), 269-277.
- [Piette 2011] Piette, J. D., Mendoza-Avelares, M. O., Ganser, M., Mohamed, M., Marinec, N., & Krishnan, S. (2011). A preliminary study of a cloud-computing model for chronic illness self-care support in an underdeveloped country. *American journal of preventive medicine*, 40(6), 629-632.
- [Pierleoni 2014] Pierleoni, P., Pernini, L., Belli, A., & Palma, L. (2014). An Android-based heart monitoring system for the elderly and for patients with heart disease. *International journal of telemedicine and applications*, 2014, 10.
- [Pinnock 2013] Pinnock, H., Hanley, J., McCloughan, L., Todd, A., Krishan, A., Lewis, S., ... & Pagliari, C. (2013). Effectiveness of telemonitoring integrated into existing clinical services on

hospital admission for exacerbation of chronic obstructive pulmonary disease: researcher blind, multicentre, randomised controlled trial.

[Piro 2014] Piro, N. E., Baumann, L., Tengler, M., Piro, L., & Blechschmidt-Trapp, R. (2014). Telemonitoring of patients with Parkinson's disease using inertia sensors. *Appl Clin Inform*, 5(2), 503-511.

[Pitt 2001] Pitt, E., & McNiff, K. (2001). *Java. rmi: The Remote Method Invocation Guide*. Addison-Wesley Longman Publishing Co., Inc.

[Poole 2001] POOLE, John D. Model-driven architecture: Vision, standards and emerging technologies. En *Workshop on Metamodeling and Adaptive Object Models, ECOOP*. 2001.

[Preece 1994] Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S., & Carey, T. (1994). *Human-computer interaction*. Addison-Wesley Longman Ltd.

[Reaction Web] <http://www.reaction-project.eu/>

[Rice 2011] Rice, P. (2011). Teleconsultation for healthcare services. *A workbook for implementing new service models*. Bradford, UK: Yorkshire & Humber HIEC.

[Richardson 2008] Richardson, L., & Ruby, S. (2008). *RESTful web services*. " O'Reilly Media, Inc."

[Riehle 2000] Riehle, D. (2000). *Framework design* (Doctoral dissertation, Diss. Technische Wissenschaften ETH Zürich, Nr. 13509, 2000).

[Rolim 2010] Rolim, C. O., Koch, F. L., Westphall, C. B., Werner, J., Fracalossi, A., & Salvador, G. S. (2010, February). A cloud computing solution for patient's data collection in health care institutions. In *eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED'10. Second International Conference on* (pp. 95-99). IEEE.

[Ronaldson 2011] Ronaldson, K. J., Fitzgerald, P. B., Taylor, A. J., Topliss, D. J., & McNeil, J. J. (2011). A new monitoring protocol for clozapine-induced myocarditis based on an analysis of 75 cases and 94 controls. *Australian and New Zealand Journal of Psychiatry*, 45(6), 458-465.

[Endomondo Web] <https://www.endomondo.com/>

[Ruiz-Zafra 2013] Ruiz-Zafra, Á., Benghazi, K., Noguera, M., & Garrido, J. L. (2013). Zappa: An open mobile platform to build cloud-based m-health systems. In *Ambient Intelligence-Software and Applications* (pp. 87-94). Springer International Publishing.

[Ruiz-Zafra 2013] Ruiz-Zafra, A., Noguera, M., Benghazi, K., Garrido, J. L., Cuberos, G., & Urbano, A. C. CloudRehab: Plataforma para la TeleRehabilitación de Pacientes con Daño Cerebral.

[Ruiz-Zafra 2013b] Ruiz-Zafra, A., Noguera, M., Benghazi, K., Garrido, J. L., Cuberos Urbano, G., & Caracuel, A. (2013, May). A mobile cloud-supported e-rehabilitation platform for brain-injured patients. In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on* (pp. 352-355). IEEE.

[Ruiz-Zafra 2014] Ruiz-Zafra, A., Noguera, M., Benghazi, K., & Jiménez, J. M. H. (2014). CloudFit: A Cloud-Based Mobile Wellness Platform Supported by Wearable Computing. In *Ambient Intelligence-Software and Applications* (pp. 151-159). Springer International Publishing.

[Ruiz-Zafra 2014b] Ruiz-Zafra, Á., Gonzalez, E. O., Noguera, M., Benghazi, K., & Jiménez, J. M. H. (2014). Energy expenditure analysis: A comparative research of based on mobile accelerometers. In *Ambient Assisted Living and Daily Activities* (pp. 38-45). Springer International Publishing.

[Ruiz-Zafra 2014c] Ruiz-Zafra, Á., Noguera, M., & Benghazi, K. (2014). Towards a Model-Driven Approach for Sensor Management in Wireless Body Area Networks. In *Internet and Distributed Computing Systems* (pp. 335-347). Springer International Publishing.

[Ruiz-Zafra 2015] Ruiz-Zafra, A., Orantes-González, E., Noguera, M., Benghazi, K., & Heredia-Jimenez, J. (2015). A Comparative Study on the Suitability of Smartphones and IMU for Mobile, Unsupervised Energy Expenditure Calculi. *Sensors*, 15(8), 18270-18286.

[Ruiz-Zafra 2015b] Ruiz-Zafra, A., Noguera, M., Benghazi, K., & Ochoa, S. F. (2015, May). A Cloud collaborative approach for managing patients wellness. In *Computer Supported Cooperative Work in Design (CSCWD), 2015 IEEE 19th International Conference on* (pp. 637-642). IEEE.

[Ruiz-Zafra 2015c] Ruiz-Zafra, A., Noguera, M., Benghazi, K., & Ochoa, S. F. (2015). A Model-Driven Approach for Wearable Systems Developments. *International Journal of Distributed Sensor Networks*, 2015.

[Runtastic Web] <https://www.runtastic.com/>

[Saffer 2009] Saffer, D. (2009). *Designing for interaction: creating innovative applications and devices*. New Riders.

[Sarria-Ere 2015] Sarria-Ere, A., Garc, B., & Gialelis, J. (2015, December). Wearable sensor-based system to promote physical activity among elderly people. In *2015 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)* (pp. 100-104). IEEE.

[Satyanarayanan 1996] Satyanarayanan, M. (1996, May). Fundamental challenges in mobile computing. In *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing* (pp. 1-7). ACM.

[Satyanarayanan 2001] Satyanarayanan, M. (2001). Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4), 10-17.

[Scanail 2006] Scanail, C. N., Carew, S., Barralon, P., Noury, N., Lyons, D., & Lyons, G. M. (2006). A review of approaches to mobility telemonitoring of the elderly in their living environment. *Annals of Biomedical Engineering*, 34(4), 547-563.

[Schmidt 2010] Schmidt, S., Schuchert, A., Krieg, T., & Oeff, M. (2010). Home telemonitoring in patients with chronic heart failure. *Dtsch Arztebl Int*, 107(8), 131-8.

[Selic 2003] Selic, B. (2003). The pragmatics of model-driven development. *IEEE software*, (5), 19-25.

[Selic 2008] Selic, B. (2008). MDA manifestations. *The European Journal for the Informatics Professional*, IX (2), 12-16.

[Seto 2008] Seto, E. (2008). Cost comparison between telemonitoring and usual care of heart failure: a systematic review. *Telemedicine and e-Health*, 14(7), 679-686. ISO 690

[Shelbourne] Shelbourne, K. D., & Nitz, P. (1990). Accelerated rehabilitation after anterior cruciate ligament reconstruction. *The American journal of sports medicine*, 18(3), 292-299.

[Siegel 2000] Siegel, J. (2000). *CORBA 3 fundamentals and programming* (Vol. 2). New York, NY, USA:: John Wiley & Sons.

[Silva 2014] Silva, P. A., Holden, K., & Nii, A. (2014). Smartphones, Smart Seniors, But Not-So-Smart Apps: A Heuristic Evaluation of Fitness Apps. In *Foundations of Augmented Cognition. Advancing Human Performance and Decision-Making through Adaptive Systems* (pp. 347-358). Springer International Publishing.

[Six 2010] Six, B. L., Schap, T. E., Zhu, F. M., Mariappan, A., Bosch, M., Delp, E. J., ... & Boushey, C. J. (2010). Evidence-based development of a mobile telephone food record. *Journal of the American Dietetic Association*, 110(1), 74-79.

[Solana 2011] Solana Sánchez, J., Cáceres Taladriz, C., Gómez Aguilera, E. J., Ferrer Celma, S., Ferre Bergada, M., & Garcia Lopez, P. (2011). PREVIRNEC A new platform for cognitive tele-rehabilitation.

[Sundaram 2013] Sundaram, P. (2013). Patient Monitoring System Using Android Technology. *International Journal of Computer Science and Mobile Computing (IJCSMC)*, 191-201.

[Szyperski 1995] Szyperski, C. (1995). Component software: beyond object-oriented programming. 1998. Harlow, England: Addison-Wesley.

[Takahashi 2015] Takahashi, T., Asahi, K., Suzuki, H., Kawasumi, M., & Kameya, Y. (2015, July). A Cloud Education Environment to Support Self-Learning at Home-Analysis of Self-Learning Styles from Log Data. In *Advanced Applied Informatics (ILAI-AAI), 2015 ILAI 4th International Congress on* (pp. 437-440). IEEE.

[Teixeira 2012] Teixeira, L., Ferreira, C., & Santos, B. S. (2012). User-centered requirements engineering in health information systems: a study in the hemophilia field. *Computer methods and programs in biomedicine*, 106(3), 160-174.

[Truyen 2006] Truyen, F. (2006). The Fast Guide to Model Driven Architecture The Basics of Model Driven Architecture. *Cephas Consulting Corp.*

[Urbaczewski 2006] Urbaczewski, L., & Mrdalj, S. (2006). A comparison of enterprise architecture frameworks. *Issues in Information Systems*, 7(2), 18-23.

[Urbano 2014] Urbano, G. C., Ruiz-Zafra, A., Noguera, M., Benghazi, K., & Caracuel, A. (2014, May). Self-monitoring and professional feedback through CloudRehab, a Mobile Cloud Platform for Neuro-rehabilitation. In *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare* (pp. 266-269). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Vandelanotte 2016] Vandelanotte, C., Müller, A. M., Short, C. E., Hingle, M., Nathan, N., Williams, S. L., ... & Maher, C. A. (2016). Past, Present, and Future of eHealth and mHealth Research to Improve Physical Activity and Dietary Behaviors. *Journal of nutrition education and behavior*, 48(3), 219-228.

[Venkatesh 2000] Venkatesh, V., & Davis, F. D. (2000). A theoretical extension of the technology acceptance model: Four longitudinal field studies. *Management science*, 46(2), 186-204.

- [Wang 2010] Wang, L., Von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., & Fu, C. (2010). Cloud computing: a perspective study. *New Generation Computing*, 28(2), 137-146.
- [Wang 2014] Wang, X., Gui, Q., Liu, B., Jin, Z., & Chen, Y. (2014). Enabling smart personalized healthcare: a hybrid mobile-cloud approach for ECG telemonitoring. *Biomedical and Health Informatics, IEEE Journal of*, 18(3), 739-745.
- [Wang 2016] Wang, D., Xiang, Z., & Fesenmaier, D. R. (2016). smartphone use in everyday life and travel. *Journal of Travel Research*, 55(1), 52-63.
- [Watier 2003] Watier, K. (2003). Marketing Wearable Computers to Consumers: An Examination of Early Adopter Consumers' Feelings and Attitudes Toward Wearable Computers. *Washington: Georgetown University*.
- [Whitehouse 2014] Whitehouse, D., & Wilson, P. (2014). Introducing eHealth: Past, Present and Future. In *Managing eHealth* (pp. 1-15). Palgrave Macmillan UK.
- [Wong 2008] Wong, J. H., Lin, W. W., Wong, A. K., & Dillon, T. S. (2008). An ontology supported meta-interface for the development and installation of customized web based telemedicine systems. In *Software Technologies for Embedded and Ubiquitous Systems* (pp. 233-244). Springer Berlin Heidelberg.
- [XSLT Web] <http://www.w3schools.com/xsl/>
- [Yang 2008] Yang, C. M., Hu, J. S., Yang, C. W., Wu, C. C., & Chu, N. (2011, November). Dancing game by digital textile sensor, accelerometer and gyroscope. In *Games Innovation Conference (IGIC), 2011 IEEE International* (pp. 121-123). IEEE.
- [Zhan 2014] Zhan, W., Smith, J., Garcia, A., Davis, C., & Kim, J. (2014). Development of Wearable Monitoring Systems for Future NASA Deep Space Exploration Missions. *Journal of Management & Engineering Integration*, 7(1), 23.