TRABAJO FIN DE MÁSTER

MÁSTER UNIVERSITARIO EN INGENIERÍA INFORMÁTICA

Control de estrés en mayores

Aplicación para SmartWatch

Autor
Víctor José Rubia López

Directores
María José Rodríguez Fórtiz
María Luisa Rodríguez Almendros

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, enero de 2025

Control de estrés en mayores

Aplicación para SmartWatch.

Autor
Víctor José Rubia López

Directores
María José Rodríguez Fórtiz
María Luisa Rodríguez Almendros

Control de estrés en mayores: Aplicación para SmartWatch

Víctor José Rubia López

Palabras clave: estrés, reloj inteligente, fotopletismografía, variación de frecuencia cardíaca, sistemas distribuidos, Machine Learning, limpieza de señal, metodologías ágiles, web, móvil, Android, WearOS, Azure, FaaS

Resumen

Las personas pueden experimentar distintos niveles de estrés en sus actividades cotidianas, especialmente las de mayor edad, debido a factores internos como la edad, salud o experiencias previas, o externos como ruido, temperatura o aglomeraciones. Identificar los factores que incrementan el nivel de estrés resulta clave para abordarlos desde un punto de vista clínico. En este Trabajo Fin de Máster se propone un entorno multisistema que permite monitorizar, analizar y predecir el estrés de manera poco invasiva y a bajo coste, ofreciendo herramientas que facilitan a los terapeutas el análisis del estrés percibido. El sistema consta de un reloj inteligente, un dispositivo móvil y una plataforma web multiusuario como núcleo central. El reloj recopila señales fisiológicas PPG, además de acelerómetro, GPS y pasos, en tiempo real, permitiendo etiquetar momentos de mayor tensión. El dispositivo móvil gestiona la autenticación multiusuario y transmite credenciales seguras al reloj. Finalmente, la plataforma web almacena los datos en una base de datos distribuida, gestionando usuarios, actividades y etiquetas, y permitiendo visualizar el detalle de los resultados para su análisis clínico. La evaluación y predicción del estrés se basan en técnicas de procesamiento de señales y algoritmos de Aprendizaje Automático desplegados como microservicios en Kubernetes. Gracias a esta arquitectura desacoplada, se filtran las señales PPG y se extraen métricas de variabilidad de la frecuencia cardíaca para estimar el nivel de estrés en ventanas de un minuto, generando, además, un perfil de estrés por usuario. Este perfil se fundamenta en una experimentación llevada a cabo con sujetos evaluados mediante el test DASS-21, que mide el estrés percibido. El conjunto de datos resultante permite entrenar un modelo para calcular el perfil de estrés percibido en función de las actividades registradas. Adicionalmente, para predecir niveles de estrés, alto, medio y bajo, se han usado los conjuntos de datos *SWELL* y *WESAD*, cuya combinación, en base a estudios publicados, produce un modelo con mayor adaptabilidad a diversas situaciones donde pueda generarse estrés. Así, se facilita el seguimiento clínico por parte de terapeutas, brindando una visión integral del estado de la persona y permitiendo analizar los factores que originan estrés en las actividades diarias.

Monitoring elderly stress: SmartWatch App

Víctor José, Rubia López

Keywords: stress, smartwatch, photoplethysmography, heart rate variability, distributed systems, Machine Learning, signal filtering, agile methodologies, web, mobile, Android, WearOS, Azure, FaaS

Abstract

People can experience varying levels of stress in their daily activities, particularly older adults, due to both internal factors, such as age, health status, or past experiences, and external factors, such as noise, temperature, or crowding. Identifying determinants that increase stress levels is critical for addressing them from a clinical perspective. This Master's Thesis proposes a multi-system framework designed to non-invasively and cost-effectively monitor, analyze, and predict stress, providing therapists with tools to facilitate the assessment of perceived stress. The proposed system comprises a smartwatch, a mobile device, and a multi-user web platform. The smartwatch continuously collects physiological signals, specifically photoplethysmography (PPG), alongside accelerometer, GPS, and step-count data, enabling the labeling of moments with heightened tension. The mobile device manages multi-user authentication, securely transmitting credentials to the smartwatch. Finally, the web platform serves as the central hub, storing all collected data in a distributed database, managing users, activities, and labels, and offering detailed visualizations for clinical analysis. Stress evaluation and prediction are performed through signal processing techniques and Machine Learning algorithms deployed as microservices on a Kubernetes cluster. This decoupled architecture allows the system to filter PPG signals and extract heart rate variability metrics in one-minute windows, generating a personalized stress profile for each user. This profile is grounded in an experimental study in which participants were assessed using the DASS-21 test, a standardized measure of perceived stress. The resulting dataset was used to train a model that computes perceived stress profiles based on recorded activities. Furthermore, in order to predict three stress levels—high, medium, and low—the system incorporates additional datasets, specifically *SWELL* and *WESAD*. According to published research, combining these datasets yields a model with enhanced adaptability to diverse scenarios where stress may be triggered. In this way, clinical monitoring is streamlined for therapists, offering a comprehensive view of each individual's condition and enabling a thorough analysis of the factors that contribute to stress in daily life.

D. **María José Rodríguez Fórtiz**, Profesora del Departamento Lenguajes y Sistemas Informáticos de la Universidad de Granada.

D. **María Luisa Rodríguez Almendros**, Profesora del Departamento Lenguajes y Sistemas Informáticos de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Control de estrés en mayores, Aplicación para SmartWatch***, ha sido realizado bajo su supervisión por **Víctor José Rubia López**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 20 de enero de 2025.

Los directores:

María José Rodríguez Fórtiz **María Luisa Rodríguez Almendros**

Agradecimientos

En primer lugar, a mi familia y a mis amigos por inspirarme y apoyarme a llegar hasta aquí. En segundo lugar, a mis tutoras María José Rodriguez Fórtiz y a María Luisa Rodríguez Almendros, por brindarme la oportunidad de realizar un proyecto de esta envergadura y darme su apoyo y consejos cuando lo necesitaba.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Planteamiento

El estrés es una respuesta fisiológica y psicológica ante una percepción de amenaza, la cual recibe el nombre en la comunidad científica de estresor. El estrés prepara al cuerpo para la "lucha" o "huida" del individuo, de modo que si se tiene un estrés prolongado en el tiempo puede tener efectos negativos en la salud. A medida que envejecemos, nuestra capacidad para hacer frente a situaciones estresantes disminuye. La importancia del estrés en personas mayores viene motivada por la relación entre la salud mental que ellos perciben y la forma de afrontarlo o la resiliencia (Luján Henríquez, 2015b).

Las personas mayores con estrés suelen manifestar comportamientos negativos, justificando sus fallos, evaluándose continuamente de forma negativa e intentando controlarlo todo de forma anticipatoria. Todos estos problemas están estrechamente relacionados con problemas con habilidades cognitivas y emocionales, de modo que la manera de tratarlos es mediante estrategias cognitivo-emocionales. De esta forma, podemos mejorar la inteligencia emocional, tanto en las relaciones consigo mismo como con el resto de personas.

El estrés aumenta el nivel de cortisol en el cuerpo, haciendo que se produzca vasodilatación y un aumento de la presión arterial. Esto aumenta el tiempo de recuperación de enfermedades y el riesgo de sufrir accidentes cardiovasculares. Asimismo, el estrés agudo contribuye al crecimiento de células malignas y a que los tumores se expandan (Guragai et al., 2020a).

La respuesta fisiológica a la variación del cortisol se puede medir con sensores fisiológicos, lo que resulta en una medición del nivel de estrés y si este resulta ser crónico o no. Entre los sensores que suelen usarse para medir las reacciones fisiológicas encontramos el EEG, para medir la actividad cerebral, el ECG o PPG, para medir el ritmo cardíaco, el SKT, para medir la temperatura de la piel, y el GSR o EDA para medir la actividad electrodérmica de la piel, entre otros. Estos sensores miden características fisiológicas, las cuales presentan una alta precisión y fiabilidad de una forma no invasiva y económica,

puesto que son sencillos de utilizar y baratos de fabricar. Asimismo, nos permiten hacer una lectura de forma continua, y no son susceptibles a diferencias culturales (S.S & P, 2019). También se puede evaluar el estrés realizando una medición del comportamiento de la persona mediante observación, para lo cual suelen utilizarse cámaras de vídeo, con sistemas de reconocimiento de imágenes para detectar gestos faciales, y micrófonos para la detección de la voz y de características del habla (Kołakowska et al., 2020).

Atendiendo a las investigaciones que hay sobre este tema en el área de "Machine Learning (ML)", se han explorado los rendimientos de diferentes algoritmos como lógica difusa, máquinas de vector soporte (SVM), árboles de decisión, así como algoritmos de k-vecinos más cercanos (KNN), redes de Bayes, Gaussianas, y Random Forest. Estos algoritmos se encuentran entre los más utilizados para detectar la presencia de estrés basándose en datos fisiológicos recogidos con sensores (Guragai et al., 2020a) (S.S & P, 2019).

Si nos centramos en personas mayores, en España, la población de la tercera edad se situó en 2021 en 9,38 millones de personas (Statista, 2021), de las cuales un 6.5 % afirmaban tener algún síntoma de estrés en el último año (INE, 2017).

La mayoría de las personas mayores perciben que tienen una buena calidad de vida y manifiestan un incremento en su felicidad subjetiva cuando mantienen relaciones familiares y sociales, con afecto positivo, en presencia también de salud y materiales de apoyo. Mismamente, las emociones positivas tienen efectos positivos sobre la salud, concretamente el sistema cardiovascular e inmunológico. Estos efectos son una mayor longevidad y un menor riesgo de padecer accidentes cerebro-vasculares, así como un menor tiempo de recuperación en caso de enfermedad o situación médica aguda (García Martínez, 2017). En cambio, las emociones negativas se asocian a problemas de salud mental, como depresión, y a un aumento de riesgo de mortalidad en personas mayores. Por esta razón, suele incrementarse la frecuencia de sentir emociones negativas cuando la persona tiene problemas de salud o de apoyo social. En un trabajo del departamento de psicología y sociología de la Universidad de Las Palmas de Gran Canaria (Luján Henríquez, 2015c) se realizó un experimento en el que se demostró que la salud mental percibida por mayores está directamente relacionada con la emotividad y la capacidad de sobreponerse o resiliencia. Los mayores con problemas de depresión, ansiedad o estrés suelen manifestar conductas de negatividad, justificación de fallos, evaluación selectiva negativa, defensividad, discordancia e hipercontrol anticipatorio, todo ello relacionado con problemas en las habilidades cognitivas y emocionales. A la luz de este experimento, se sugiere trabajar en estrategias cognitivo-emocionales para modificar emociones, comportamientos, y mejorar el control de las inteligencias emocional, interpersonal e intrapersonal.

En los últimos años se han realizado numerosos estudios (Schwartz, s.f.) (Stults-Kolehmainen & Sinha, 2014) acerca de cómo influye el estrés en el abandono de la realización de actividades de la vida cotidiana, especialmente mayores, que sufren el riesgo de quedarse aislados y caer en depresión por tener una concepción de no adaptabilidad a la vida actual.

De ello, resulta necesario admitir mi inspiración a desarrollar un sistema que use tecnología para analizar el grado de estrés en el transcurso de actividades de la vida diaria, mediante el conocimiento de factores fisiológicos, ambientales y sociales que rodean al usuario, como si se encuentra solo o acompañado. Por esto, me mueve querer facilitar y extender el uso de dispositivos wearables al máximo número de personas posible.

1.2. Objetivos

El objetivo de este trabajo fin de máster es el análisis, desarrollo e implementación de un entorno multisistema para la monitorización de datos fisiológicos para todo tipo de usuarios durante el transcurso de cualquier actividad diaria, siendo de especial utilidad para personas mayores o con alguna discapacidad.

Este sistema permitirá, por un lado, modelar una actividad en cuanto a grado de estrés producido y conocer los factores que la producen, por otro lado, otorgará a los terapeutas la posibilidad de hacer un seguimiento más preciso a sus pacientes mediante el registro de sus usuarios y la visualización de sus actividades junto con el análisis de los datos fisiológicos y etiquetados que se recopilan. Todo esto se llevará a cabo de una forma poco invasiva y con una solución de bajo coste, a través del uso de un reloj inteligente con sensores y una plataforma web para el terapeuta. Este objetivo general se conseguirá mediante la consecución de los objetivos específicos que se enumeran a continuación:

1. Estudiar el dominio del problema a resolver, el estrés, sus causas, síntomas y mecanismos de medición, incluidos aquellos que recogen datos fisiológicos con sensores.
2. Analizar cómo la realización de actividades de la vida diaria puede ser estresante, y esto puede medirse a través de datos fisiológicos y etiquetado.
3. Desarrollar un método para obtener mediciones fisiológicas de un usuario a través de un reloj inteligente.
4. Realizar un estudio acerca de las metodologías actuales para la limpieza/preprocesado de señales fisiológicas.
5. Desarrollar un método para calcular el nivel de estrés de un usuario a partir de sus mediciones fisiológicas.
6. Proporcionar una herramienta para la supervisión de usuarios mayores por parte de terapeutas, un sistema que, a modo de panel de control, permita registrar usuarios y visualizar los resultados de estrés de éstos cuando realizan cualquier actividad de la vida diaria.
7. Optimizar los recursos del reloj inteligente, incluyendo la gestión eficiente de la batería y la prevención de la pérdida de mediciones causada por la suspensión de la aplicación en el sistema WearOS de Android.

8. Incorporar el uso de más sensores del reloj como parte de la medición de actividades, como el GPS, de modo que se trace la ruta del usuario en la realización de su actividad.

9. Desarrollar una aplicación para relojes inteligentes que registre actividades de diferentes tipos en la plataforma del terapeuta asociado.

1.3. Estructura de la memoria

Para cumplir con estos objetivos planteados, esta memoria de trabajo fin de grado se estructura en cinco capítulos, cuyos contenidos se describen a continuación:

- En el capítulo 1, se encuentra la Introducción, donde se ha descrito la motivación y justificación para llevar a cabo este proyecto, explicando la necesidad existente junto con los problemas de la actualidad. Además, se proponen mejoras sobre lo ya existente y se explica porqué se ha escogido esta temática. De igual modo, se expone un objetivo general junto con objetivos más específicos justificados. Adicionalmente, se incluye la planificación temporal mediante un Diagrama de Gantt y el presupuesto que conllevaría realizar el trabajo.

- En el capítulo 2, se describe el Estado del Arte y Tecnologías, donde se describen los antecedentes al proyecto mediante la exposición de tecnologías base potenciales, la explicación del dominio del problema a resolver, la descripción de trabajos relacionados y analizando las aplicaciones similares que existen en la actualidad.

- En el capítulo 3, se realiza la Propuesta, donde se elige una metodología de trabajo de forma justificada y se detallan su desarrollo y los resultados junto a la exposición de herramientas de ingeniería, como los diagramas y plantillas que se usan.

- En el capítulo 4, se indican las conclusiones obtenidas a partir del desarrollo que se ha realizado en este trabajo fin de grado. Asimismo, se proponen trabajos futuros sobre este proyecto.

- Como anexo A, se proporciona un glosario de términos que explican los acrónimos y abreviaturas que se utilizan a lo largo de la memoria.

- Como anexo C, se proporciona un manual de instalación de los sistemas desarrollados.

- Como anexo D, se proporciona un manual de usuario para explicar las funcionalidades del sistema.

- Como anexo E, se proporciona el repositorio y la documentación del código desarrollado para la aplicación del reloj y móvil.

1.4. Planificación temporal

Para la realización del presente Trabajo Fin de Máster, se ha realizado una planificación basada en iteraciones para llevar a cabo el desarrollo. La planificación del proyecto abarca desde la primera reunión con las tutoras, donde se desarrolla una primera aproximación del proyecto, hasta las últimas revisiones de esta memoria. La duración total del proyecto es de aproximadamente 6 meses.

Para la planificación se ha tenido un periodo previo al comienzo de las iteraciones, donde se aporta la explicación por parte de las tutoras al desarrollador, de modo que se puedan especificar los requisitos generales a abordar, en forma de historias de usuario, y puedan corregirse. De este modo, se puede elaborar un plan de iteraciones haciendo un reparto de historias de usuario de las cuales se ha estimado un tiempo de realización, y obtener el número de iteraciones que se van a necesitar en total. Se ha intentado distribuir la carga de trabajo de la forma más equitativa posible entre iteraciones. Se puede visualizar el diagrama de Gantt en la figura 1.1.

A continuación, se explica el diagrama de Gantt de la figura 1.1, con más detalle:

- Al comienzo del proyecto, se establece un periodo de 10 días en el que se tiene la reunión inicial con las tutoras, donde se abarca la redacción de las historias de usuario que cumplen las necesidades para la realización del proyecto. Con esto, se elabora un plan de iteraciones y se comienza con la investigación, instalación y configuración del entorno de desarrollo necesario para comenzar las iteraciones.
- En el periodo de iteraciones, se realiza la extracción de las pruebas de aceptación de las historias de usuario correspondientes a la iteración, junto con la redacción de las tareas asociadas. Tras esto, se realiza una planificación mediante un diagrama de Gantt. A continuación, se procede a realizar las tareas asociadas a las historias de usuario, para finalmente, obtener documentos entregables que poder mostrar en la reunión de retrospectiva con las tutoras.
- Al mismo tiempo, se comienza con el desarrollo de la memoria, escribiendo los capítulos que asientan las bases del presente proyecto asegurando su calidad.

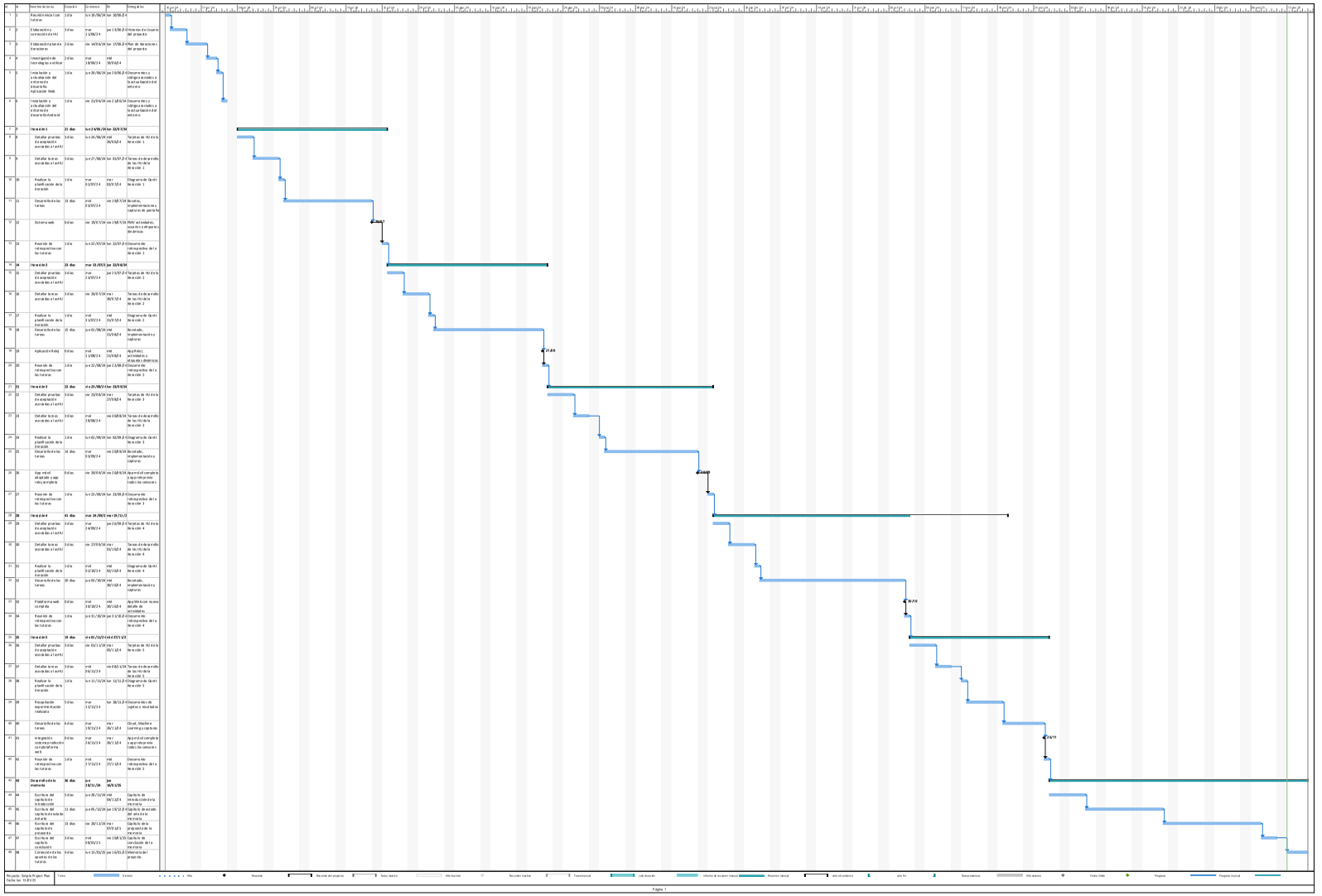


Figura 1.1: Diagrama de Gantt que describe la planificación del proyecto

1.5. Presupuesto

El presupuesto para llevar a cabo el presente trabajo fin de máster puede desglosarse en los siguientes puntos:

- **Equipo de desarrollo**. Es indispensable contar con un equipo a través del cual diseñar e implementar los diferentes componentes del sistema: aplicaciones móviles, la aplicación para reloj inteligente y la plataforma web.
- **Servidor Web**. Un equipo que albergue el servidor web para el terapeuta y que provea las funcionalidades necesarias para el buen funcionamiento de las aplicaciones para el móvil y reloj inteligente. Además, almacenará los datos de los usuarios en una base de datos.
- **Reloj inteligente**. Un dispositivo WearOS versión 2.0 en adelante para poder ejecutar la aplicación desarrollada de medición de actividades.
- **Teléfono móvil**. Un dispositivo móvil Android para poder ejecutar la aplicación desarrollada para cambiar el usuario del reloj inteligente.
- **Dominio Web**. Una dirección web para no depender de una IP dinámica, de modo que los distintos dispositivos puedan conectarse al servidor desde fuera de la red local.
- **Conexión a internet**. Se necesita de una red de banda ancha para la interconexión de los sistemas desde fuera de una red local.
- **Coste humano**. El coste por las horas de trabajo realizado por el desarrollador del proyecto.

A continuación, se han estudiado las necesidades de los sistemas y se hace una propuesta para cada sección del desglose anterior.

1.5.1. Servidor Web

Se recomienda un equipo capaz de manejar concurrentemente varios usuarios en el sistema, así como para dar soporte a la ejecución de la lógica necesaria para la detección y clasificación de estrés.

Además, el servidor debe contar con amplio almacenamiento para guardar toda la información que se recibe en crudo de los sensores de los usuarios. Para una disponibilidad de servicio más estable se recomienda conectar el equipo a través de cable Ethernet a internet.

Tras estudiar las necesidades del sistema web, de cara a poner el proyecto en producción, se recomienda la adquisición del equipo Dell PowerEdge T50, cuyo precio es 689,75 € sin IVA o un modelo similar. Este equipo posee las siguientes características técnicas:

- **Procesador**.
  - Modelo: Intel Xeon E-2314
  - Frecuencia base: 2,8 GHz
  - Frecuencia turbo: 4,5 GHz
  - Núcleos: 4
  - Caché: 8 MB
- **Memoria**.
  - Capacidad: 8 GB DDR4-SDRAM
  - Velocidad: 3200 MHz
  - Ranuras: 4 DIMM
  - Capacidad máxima: 32 GB
  - ECC: Sí
- **Almacenamiento**.
  - Capacidad: 1 TB
  - Interfaz: SATA
  - Velocidad de rotación: 7200 RPM
  - Tamaño del disco: 3,5”
  - Compatibilidad con RAID: No
- **Red**.
  - Interfaz: Gigabit Ethernet
  - Puertos Ethernet (RJ-45): 2
  - Controlador LAN: Broadcom 5720
- **Puertos e Interfaces**.
  - USB 2.0: 5
  - USB 3.2 Gen 1: 2
  - VGA (D-Sub): 1
  - Puerto serial: 1
- **Diseño y dimensiones**.
  - Tipo de chasis: Bastidor (4U)
  - Ancho: 175 mm
  - Profundidad: 453,8 mm

  - Altura: 360 mm
  - Peso: 11,7 kg

- **Administración y soporte**.
  - Gestión remota: iDRAC9 Basic 15G
  - Compatibilidad con sistemas operativos: Microsoft Windows Server, Red Hat Enterprise Linux, SUSE Linux Enterprise Server, Canonical Ubuntu Server LTS.

Por otro lado, para el desarrollo a nivel personal del proyecto se ha utilizado un ordenador personal modelo MacBook Pro de 16çon 32 GB de RAM, 1 TB de SSD y un procesador M1 Max. El coste inicial en 2021 de este producto fue de 3 600 €, por lo que considerando una vida útil estimada de 5 años, que son 60 meses, el coste proporcional asignado a los 6 meses empleados en el desarrollo del proyecto asciende a 360 €.

Además, para tareas específicas de *Machine Learning* y el despliegue del proyecto, se utilizará una máquina virtual de Azure del tipo *Standard B2ms* con 2 vCPU y 8 GB de RAM) en la región *France Central*, junto con otros servicios relacionados. Atendiendo a la facturación de diciembre de 2024, los costes asociados a Azure en diciembre de 2024 se desglosan en:

- Uso de la máquina virtual: 0,57 € (6.383 horas de proceso).
- Dirección IP pública estática: 3,54 € (744 horas).
- Registro de contenedores: 4,92 € (31 unidades de registro/día).
- Disco SSD estándar: 2,50 € (1 disco/mes).
- Disco SSD premium: 1,42 € (1 disco/mes).

En total, los costes acumulados de los servicios de Azure para diciembre de 2024 ascienden a 12,95 €. Suponiendo un consumo similar durante los 6 meses del proyecto, el coste estimado total de Azure sería de aproximadamente 77,7 €.

Por tanto, sumando los costes del uso del equipo local y los servicios en la nube, el coste total asociado al desarrollo del TFM se estima en 437 €.

Amortización del equipo para producción

El coste del equipo, sumando el IVA del 21 %, asciende a **834,60 €**. De acuerdo con las normativas fiscales españolas (Agencia Tributaria, 2020), los equipos informáticos suelen amortizarse linealmente al **25 % anual**, lo que corresponde a un periodo de amortización de 4 años.

Si el período de desarrollo del proyecto es de **un año**, la parte del gasto amortizable sería:

$$\text{Amortización anual (25 \% del coste total)}: 834,60\,€ \times 0.25 = 208,65\,€.$$

Por tanto, el gasto amortizable correspondiente a este equipo durante un año sería **208,65 €**.

Consideraciones adicionales

- **Garantía**. Este equipo incluye una garantía estándar de **1 año**, con opciones de ampliación según las necesidades del proyecto.
- **Soporte adicional**. Si se contrata soporte técnico o extensiones de garantía, estos costes pueden imputarse como gastos adicionales del proyecto.
- **Compatibilidad futura**. El equipo permite la ampliación de memoria RAM y almacenamiento, garantizando que pueda adaptarse a los requerimientos que surjan en el futuro.

1.5.2. Reloj inteligente

Para asegurar la compatibilidad con la aplicación desarrollada, el reloj inteligente debe poseer entre sus características un sensor de fotopletismografía (PPG) que permita recoger información sobre el pulso cardiaco y el volumen de sangre. El que se detalla a continuación es un modelo sucesor al que se ha usado para el desarrollo del proyecto. Se propone un reloj que tenga conectividad 4G para poder usarlo independientemente de tener el móvil cerca al reloj o no.

El dispositivo propuesto es el TicWatch Pro 3, que tiene un coste de 284,39 € sin IVA, o un modelo similar. Este dispositivo posee las siguientes características técnicas:

- Dimensiones y peso: 47 x 48 x 12,2 mm. 42 gramos.
- Tamaño y tecnología de pantalla: AMOLED de 1,4 pulgadas
- Procesador: Snapdragon Wear 4100
- Memoria RAM: 1 GB
- Almacenamiento interno: 8 GB
- Capacidad de batería: 577 mAH
- Conectividad: Bluetooth Low Energy 4.2.
- Sensores: Sensor de ritmo cardíaco PPG, Giroscopio, Barómetro, GPS y NFC.
- Versión WearOS: 2.0 con futura actualización a 3.0

1.5.3. Teléfono móvil

Los requerimientos del dispositivo móvil son escasos, ya que tan solo se necesita un teléfono Android con una versión superior a la 11.0. Este tendrá la aplicación de WearOS con el reloj inteligente enlazado y deberá disponer de una SIM con un plan de datos suficiente para las mediciones. Se propone una tarifa de teléfono con 12 GB de datos para consumir a lo largo de un mes, con un coste de 7 € sin IVA en el operador O2.

El dispositivo móvil que se propone, un Samsung Galaxy A13 posee una gran pantalla, ideal para personas mayores con dificultad de visión y con un precio contenido de 245,48 € sin IVA, o un modelo similar.

Este dispositivo cuenta con las siguientes características:

- Tamaño y tecnología de pantalla: TFT LCD de 6,6 pulgadas
- Procesador: Exynos 850
- Memoria RAM: 4 GB
- Almacenamiento interno: 64 GB
- Capacidad de batería: 5.000 mAh
- Conectividad: 5G, WiFi, Bluetooth Low Energy 5.0
- Sensores: GPS, Acelerómetro, Giroscopio.

1.5.4. Dominio web

Para poder hacer uso de nuestro sistema fuera de una red local es necesario que éste lo permita. Para solventar el coste adicional que supone que las operadoras establezcan una IP fija, se optará por el uso de un dominio web. Se propone el servicio usado para la realización del presente proyecto, que tiene un coste anual de 37,92 € sin IVA y está ofrecido por Google Domains.

1.5.5. Conexión a internet

Para el buen funcionamiento del servidor web, es necesario poseer una conexión a internet de banda ancha, a ser posible, mediante fibra óptica, ya que reduce bastante los tiempos de latencia en las peticiones que se le realicen a nuestro servidor.

Se propone, pues, una tarifa de la compañia O2 que ofrece 300 megabits de conexión simétrica por fibra óptica por un precio de 27 € al mes sin IVA.

1.5.6. Coste humano

Atendiendo a que el perfil del desarrollador del presente trabajo fin de máster sería de Junior Software Developer, tanto por sus estudios, como por las necesidades del proyecto, se ha procedido a consultar en diversas plataformas como Glassdoor (Glassdoor, 2024) y Talent (Indeed, 2024) el sueldo medio actual que se ingresa netamente por este tipo de perfiles en España. Lo que se ha encontrado, de media, es un sueldo mensual de 1833 €. Esto supone que la hora de trabajo se fije en 10,42 €. Sin embargo, al incluir los costos adicionales que asume la empresa, es necesario recalcular el coste total.

En primer lugar, se debe añadir la cuota patronal correspondiente a la Seguridad Social, que para un contrato eventual asciende al 32,1 % del sueldo bruto. Además, se considera la retención del IRPF (Impuesto sobre la Renta de las Personas Físicas), que en el caso de un perfil junior puede oscilar en torno al 15 % del sueldo bruto. Estos porcentajes se aplican al sueldo bruto mensual, lo cual incrementa significativamente el coste total asumido por la empresa.

Tomando estos valores en cuenta, el coste total para la empresa no solo incluye el sueldo neto del empleado, sino también la cuota patronal de la Seguridad Social y la retención del IRPF. Este coste combinado elevaría el gasto mensual a aproximadamente 2.517,65 €. Esto a su vez implica que el coste por hora de trabajo del desarrollador, incluyendo todos los conceptos, asciende a 14,52 €.

1.5.7. Presupuesto final

Como la duración del proyecto es de 6 meses, el coste total del proyecto, sumando los dispositivos, servicios necesarios y coste humano, sería de 1.574,41 € al mes, desglosándose en los siguientes conceptos:

- **Equipo de desarrollo**:
  - Servidor web: 834,60 € (incluyendo IVA, amortización anual de 208,65 €).
  - Reloj inteligente: 284,39 € (sin IVA).
  - Teléfono móvil: 150,02 € (sin IVA).
- **Servicios necesarios**:
  - Dominio web: 37,92 € (anual, sin IVA).
  - Conexión a internet: 27 €/mes × 6 meses = 162 € (sin IVA).
  - Plan de datos móviles: 7,9 €/mes × 6 meses = 47,4 € (sin IVA).
- **Coste humano**:
  - Número total de horas trabajadas: 4 horas.
  - Coste por hora (incluyendo impuestos y tasas): 14,52 €/hora.
  - Coste humano total: 4 horas × 14,52 €/hora = 58,08 €.

Capítulo 2

Estado del Arte y Tecnologías

2.1. Contexto del trabajo

Este trabajo se enmarca dentro de la post-pandemia, donde muchos usuarios, especialmente personas mayores o con alguna discapacidad, sienten mucho estrés a la hora de retomar sus actividades cotidianas. En España, según el Ministerio de Sanidad, el trastorno de ansiedad afecta al 6,7 % de la población con tarjeta sanitaria, manteniéndose relativamente estable entre los 35 y 84 años (Subdirección General de Información Sanitaria, 2021). Esto subraya la importancia de abordar este problema, dado que las personas mayores enfrentan desafíos adicionales que pueden exacerbar su vulnerabilidad emocional. Por ello, se vuelve imperativo que los entornos y servicios se diseñen pensando en estas necesidades. El objetivo debe ser minimizar el estrés y maximizar la accesibilidad y la facilidad de realización de actividades cotidianas para todas las personas, independientemente de su edad o habilidad.

En el caso de los mayores, este estrés se une al que ya poseen por presentar muchos de ellos problemas de movilidad o deterioros cognitivos. Por eso, la consideración de las limitaciones físicas y los desafíos cognitivos en el diseño y la implementación de servicios es fundamental. Es imprescindible promover la independencia y la autonomía de las personas mayores, al mismo tiempo que se garantiza su seguridad y bienestar mejorando su salud mental y emocional, reduciendo el aislamiento y promoviendo la inclusión y la participación activa en la sociedad.

Dado que las personas viven cada vez más vidas longevas y con una buena salud, no se puede negar que éstas tienen potencial para realizar contribuciones importantes a la sociedad. Sin embargo, las personas mayores son vulnerables, muy frecuentemente, a situaciones de exclusión, marginalización y discriminación.

La relación de población mayor respecto a la población mundial ocupa una proporción cada vez más elevada, siendo la población de la cuarta edad (80 años y superior) la que crece más rápido (Bank, 2024). Mismamente, la población mayor que alcanza la

jubilación está cada vez más sana y en forma. El incremento de la esperanza de vida y la mejora en el estado de salud de personas mayores es un logro importante y tiene un gran potencial en términos de capacidad laboral, capacitación y experiencia, que la sociedad debe aprovechar de forma productiva. Se observa que las personas mayores activas están más integradas en la sociedad, tienen una mayor calidad de vida, y viven de una forma más longeva y sana.

La gran mayoría de los adultos mayores sienten que disfrutan de un alto nivel de bienestar y experimentan un aumento en su contento personal cuando mantienen vínculos familiares y sociales fuertes (afectos positivos), especialmente cuando tienen una buena salud y tienen acceso a recursos que les sirven de apoyo. De igual forma, los sentimientos positivos generan beneficios sobre la salud, concretamente en el sistema cardiovascular e inmunológico, y están vinculados con una vida más larga, menor probabilidad de sufrir accidentes cerebrovasculares y tiempos de recuperación más cortos en caso de enfermedades o condiciones médicas agudas (García-Martínez et al., 2021). En contraste, las emociones negativas están ligadas a problemas de salud mental como la depresión y a un mayor riesgo de muerte en los mayores. Se observa un incremento en la tendencia a experimentar emociones negativas cuando el individuo enfrenta problemas de salud o carece de apoyo social.

La sociedad necesita considerar, de una forma más activa, cómo integrar personas mayores y asegurar su participación en una sociedad cohesiva de todo tipo de edades. Las personas mayores se integran en la sociedad de muchos modos. Forman parte de los núcleos sociales de sus amigos y familiares, participan de forma activa en clubes y asociaciones, realizan voluntariados y son económicamente activos. Sin embargo, las personas mayores pueden ser vulnerables a la exclusión. Las principales barreras para lograr una participación igualitaria de las personas mayores en la sociedad incluyen la pobreza, la mala salud, el bajo nivel educativo, la falta de medios de transporte, la accesibilidad a los servicios y la discriminación por edad. Por ello, lograr la inclusión y la participación social de este grupo abarca varios aspectos. Involucra a todos los grupos sociales e individuos en las estructuras políticas, sociales, culturales y económicas de la sociedad para que puedan participar en la toma de decisiones sobre los asuntos que les afectan. Esto requiere que exista un consenso para reducir y eliminar la exclusión, asegurando que todos los afectados reciban ayuda por parte de la sociedad. (United Nations Economic Commission for Europe, 2009)

Para lograrlo, se deben tener en cuenta todas las esferas de actividades cotidianas que desempeña una persona mayor en su día a día. Entre otras, debemos tener en cuenta la capacidad de preparar los alimentos y de consumirlos, incluyendo facilidades para abrir envases, cocinar sin riesgos y acceder a lugares donde poder comprarlos, la movilidad dentro del hogar, pudiéndose desplazar con seguridad y facilidad e incluyendo subir y bajar escaleras o moverse entre habitaciones sin riesgos de caídas, la interacción con otras personas, esencial para mantener vínculos sociales, el transporte, de modo que se puedan desplazar fuera de sus hogares, el ocio y la salud. La facilidad para realizar estas tareas sin obstáculos reduce la sensación de dependencia, permite mantener una rutina

diaria y fortalece el sentimiento de pertenencia y participación activa en la sociedad. Sin accesibilidad, se pueden generar sentimientos de aislamiento, frustración y deterioro de la salud mental y física.

España legisló sobre estos problemas de forma específica mediante la Ley de Igualdad de Oportunidades, No Discriminación y Accesibilidad Universal de 2003 (Boletín Oficial del Estado, 2003). Esta ley fue derogada en 2013 y su reemplazo es en la actualidad la Ley General de derechos de las personas con discapacidad y de su inclusión social (Boletín Oficial del Estado, 2022).

En cuanto al marco legal Europeo existen también directivas que garantizan que las personas con discapacidad o con dificultades puedan disfrutar plenamente de sus derechos en la UE. Se centra en áreas como la igualdad, la no discriminación y la promoción de la participación activa en la sociedad, incluyendo la accesibilidad a servicios, productos y entornos (Europea, 2021).

En este contexto, el desarrollo de herramientas tecnológicas accesibles y efectivas para la monitorización de datos fisiológicos adquiere una relevancia crucial. La evidencia científica demuestra que el uso de dispositivos portátiles para el seguimiento de datos biométricos puede mejorar significativamente el manejo de enfermedades crónicas y reducir la ansiedad en poblaciones vulnerables, como personas mayores o con discapacidad (Escalona & Basto, 2024). Estudios recientes también destacan el papel positivo de la tecnología en la promoción del envejecimiento activo y la reducción del aislamiento social, factores determinantes en la calidad de vida (Yáñez-Yáñez & Draguicevic, 2021).

Al combinar dispositivos de bajo coste, como relojes inteligentes, con una plataforma web diseñada para terapeutas, se abre la posibilidad de abordar estos desafíos de manera práctica e inclusiva. Este trabajo busca ser una contribución significativa en este ámbito, ofreciendo una solución que ayude a detectar el estrés en personas mayores, así como a identificar sus posibles causas, de modo que se pueda planificar una intervención preventiva. De esta manera, se da un paso hacia una sociedad más equitativa y accesible, que aproveche las capacidades y experiencias de todos sus integrantes, sin dejar a nadie atrás.

2.2. Dominio del problema

2.2.1. Definición de estrés

Con el mundo cambiando muy rápido y el ajetreo diario, a menudo se afirma estar estresado. De hecho, el estrés es la causa principal por la que las personas acuden a un médico, psicólogo o especialista. La Organización Mundial de la Salud (OMS) define el estrés como «el conjunto de reacciones fisiológicas que prepara al organismo frente a la acción». (Torrades, 2007) Por tanto, se afirma que el estrés es una respuesta que nuestro organismo realiza durante determinadas situaciones, a priori amenazantes, y que hace que estemos listos para lo que venga. (Ramírez Loeffler & Rodríguez Vílchez, 2012)

El problema surge cuando el estrés se vuelve crónico, es decir, el cuerpo se mantiene en alerta incluso cuando no hay peligro, causando problemas de salud con síntomas físicos, motores, cognitivos y emocionales (Ramírez Loeffler & Rodríguez Vílchez, 2012). A continuación se exponen algunos de estos síntomas pertenecientes a cada grupo:

- **Síntomas físicos**. Entre otros, se dan taquicardias, temblores, sudoración, náuseas, dolor de cabeza y fatiga.
- **Síntomas motores**. Encontramos que se tiende a evitar situaciones temidas, tipos de respuesta como hiperactividad, intranquilidad motora, entre otros.
- **Síntomas cognitivos**. Cuando una persona está en situación de estrés puede sufrir desorientación, olvidos frecuentes, miedo, temor a que los demás se den cuenta de nuestras dificultades, agotamiento físico y psíquico, entre otros.
- **Síntomas emocionales**. Una persona con ansiedad se sentirá incapaz de afrontar situaciones debido a sus auto-valoraciones negativas, que hacen que se imagine todos los males que pueden derivarse de la situación. Estas respuestas emocionales se caracterizan por el miedo, la tristeza, la preocupación, la sensación de inseguridad, enfado o ira e irritabilidad.

Entre las actividades cotidianas que generan más estrés a las personas mayores encontramos aquellas que están relacionadas con la gestión de su independencia, como tareas domésticas, la gestión de sus medicamentos y las visitas médicas.

Por ejemplo, una actividad que deriva en estrés es la de tener que realizar un desplazamiento a causa de algunas de las actividades que se han mencionado en el párrafo anterior. De acuerdo con un estudio realizado en Suecia (Hansson et al., 2011) donde se explica que un desplazamiento largo conlleva, en la mayoría de ocasiones, el realizar un trasbordo entre buses o trenes, hacer conexiones y no perder de vista las paradas. En consecuencia, el usuario tiene un mayor riesgo de equivocarse o sufrir un retraso o pérdida de conexión, debido a que es algo que le es imprevisible e incontrolable, produciéndose así un mayor grado de estrés y rechazo a la hora de realizar esta actividad.

Así pues, los desplazamientos pueden suponer tener que ajustar la vida cotidiana a los horarios de los autobuses o trenes (Kluger, 1998), lo que provoca inflexibilidad y pérdida de control.

2.2.2. Estrés y salud mental en mayores

La relevancia del estrés en las personas mayores radica en su vínculo con la percepción de la salud mental y las capacidades de afrontamiento (respuesta emocional frente a situaciones críticas) o resiliencia (Luján Henríquez, 2015a). Las personas mayores que experimentan estrés suelen presentar actitudes negativas, justificación de fracasos, evaluaciones desfavorables, defensividad, discrepancias y un excesivo control anticipatorio,

que impacta en sus habilidades cognitivas y emocionales. Las intervenciones se centran en desarrollar estrategias cognitivo-emocionales, orientadas a transformar emociones y conductas, fortaleciendo las inteligencias emocional, interpersonal e intrapersonal.

El estrés surge tanto de factores internos, como pensamientos, creencias y actitudes, como de factores externos, que incluyen circunstancias y relaciones interpersonales. El estrés puede tener un impacto significativo en la salud física, emocional y psicológica, influenciado por variables como la edad, el género, las experiencias previas y las condiciones fisiológicas. Estas condiciones elevan los niveles de cortisol y alteran la actividad de la insulina, lo que provoca vasodilatación, aumento de la presión arterial y una recuperación más lenta de enfermedades, además de incrementar el riesgo de padecer problemas cardiovasculares. Asimismo, un estrés prolongado reduce la actividad de las células asesinas naturales en el organismo, lo que facilita el desarrollo de células malignas y favorece la propagación de tumores (Guragai et al., 2020b).

Emociones y sentimientos en salud mental

Al revisar la literatura, se encuentran en el trabajo de Cartensen (Carstensen, 2019) dos teorías principales que explican la relación entre cognición y emoción en las personas mayores. La primera teoría, de enfoque sociocognitivo, plantea que los adultos mayores tienden a priorizar relaciones sociales significativas que les aporten emociones positivas. Estas elecciones contribuyen a que experimenten vivencias emocionales más satisfactorias, reduciendo las experiencias negativas. Por otro lado, la segunda teoría, con un enfoque neuropsicológico, sostiene que las respuestas adaptativas a estímulos pueden verse afectadas por el envejecimiento, debido a deterioros en áreas específicas del cerebro que influyen en la vida emocional.

Diversos estudios respaldan la primera teoría, destacando la presencia de un sesgo positivo en procesos de atención y memoria, así como una correlación entre la frecuencia de emociones positivas y la participación social. Esto genera un ciclo de retroalimentación positivo: las emociones agradables fomentan el deseo de interactuar socialmente, mejorando la percepción de los demás y potenciando la confianza, el optimismo y la generosidad. Este proceso estimula la inteligencia emocional, ya que las personas mayores desarrollan una mayor capacidad de autorregulación emocional mediante estrategias preventivas, incrementan su tolerancia a la frustración y aprenden a transformar emociones negativas en positivas (Tena et al., 2019).

Además, se ha observado que, en las personas mayores, la intensidad de las experiencias emocionales y su control tienden a disminuir, debido a una mayor estabilidad y madurez emocional. Esto favorece una adaptación más saludable a las demandas emocionales de la vida. Cuando las personas mayores mantienen relaciones familiares y sociales, suelen sentir que tienen una mayor calidad de vida y expresan un incremento en su felicidad subjetiva, sobre todo cuando esto se complementa de salud y recursos materiales de apoyo. (García Martínez et al., 2017)

Al mismo tiempo, las emociones positivas tienen un efecto positivo sobre la salud, especialmente en el sistema cardiovascular e inmunológico. Todo esto tiene una relación directa con una mayor longevidad y con un menor riesgo de padecer accidentes cerebrovasculares, así como un menor tiempo de recuperación frente a enfermedades o cuadros médicos agudos (García Martínez et al., 2017). Las emociones negativas, por contrario, se asocian a problemas de salud mental como depresión y aumento de riesgo de mortalidad en personas mayores. Estas emociones negativas incrementan su frecuencia cuando la persona tiene problemas de salud o de apoyo social.

En (Luján Henríquez, 2015a) se demuestra a través de un estudio que la salud mental que perciben los mayores está relacionada con la emotividad y la capacidad de sobreponerse, o resiliencia. Los mayores con problemas de estrés, suelen manifestar conductas de negatividad, justificación de fallos, evaluación selectiva negativa, defensividad, discordancia e hipercontrol anticipatorio, estando todo ello relacionado con problemas con las habilidades cognitivas y emocionales. A raíz de este experimento se sugiere trabajar en estrategias cognitivo-emocionales para modificar emociones y comportamientos, junto con aspectos de inteligencia emocional, interpersonal e intrapersonal.

2.2.3. Métodos para la medición del estrés

La forma principal de evaluar los niveles de estrés es utilizar una escala de autoinforme en un marco clínico (Chen et al., 2021). A continuación, se describen tres de ejemplos donde se utiliza esta forma de evaluación del nivel de estrés:

- La creación de escenarios que inducen estrés ayudan a los investigadores a recopilar y probar señales fisiológicas o de comportamiento asociadas con el estrés. Estas pruebas se basan, generalmente, en que se les pide a los sujetos que realicen una tarea específica que puede causar estrés, rellenen unos cuestionarios y cumplan con ciertas condiciones, como ruidos ambientales, que se esté en un ambiente que no influya sobre el estrés. También se pregunta por el padecimiento de ciertas enfermedades, como la fibromialgia. Tras esto, se puede inferir la relación con el estrés, al monitorizar los cambios fisiológicos durante el experimento y compararlos con las respuestas a los cuestionarios.
- El DASS-21 (Lovibond & Lovibond, 1995), incrustado en el Apéndice B es en un cuestionario autoadministrado de 21 ítems que evalúa los estados emocionales de depresión, ansiedad y estrés. Se le solicita al sujeto que responda a cada afirmación basándose en su experiencia durante la última semana, utilizando una escala Likert de 4 puntos que va desde "No me ha sucedido"hasta "Me ha sucedido la mayor parte del tiempo". Cada una de las tres subescalas contiene 7 ítems específicos, permitiendo medir con precisión cada constructo emocional. Tras completar el cuestionario, se calculan las puntuaciones para cada subescala, facilitando la identificación de niveles elevados de depresión, ansiedad o estrés en el individuo evaluado.

- En cuanto al uso de dispositivos *wearables*[1], descritos en la Sección 2.9, en la detección de estrés, se muestra cómo los sensores de estos son capaces de recoger marcadores o señales fisiológicas útiles. Los más comunes son la actividad cardíaca (ECG), la acitividad cerebral (EEG), la actividad muscular (EMG), la actividad electrodérmica de la piel (EDA) y la temperatura de la piel y cuerpo. (Chen et al., 2021)

2.2.4. Medición de estrés a través de experimentación

Para la realización de este Trabajo Fin de Máster, se planteó hacer un experimento real con sujetos mayores en centros de día, de modo que sirviese como base para crear un conjunto de datos que permitiese entrenar un modelo de Aprendizaje Automático capaz de predecir el estrés al que se someten las personas en su día a día durante la realización de actividades cotidianas.

Planteamiento del experimento

El presente experimento se llevó a cabo en un centro de día, donde se reclutaron participantes interesados en formar parte de tres actividades distintas, con una duración total de 45 minutos. Durante este período, a cada participante se le colocaron tres pulseras inteligentes—dos en un brazo y una en el otro—para registrar la actividad de los sensores a lo largo de toda la sesión. Las pulseras utilizadas fueron dos dispositivos Android Wear y una Empatica E4.

La Empatica E4 es una pulsera inteligente ampliamente utilizada en investigaciones para monitorizar señales fisiológicas, como la actividad electrodérmica de la piel, la frecuencia cardíaca, la aceleración tridimensional y la temperatura de la piel (Stuyck et al., 2022).

Una de las actividades que se realizaron, dentro del marco de un proyecto I+D andaluz en el que estaba contratado, fue la que se ha tomado para el desarrollo del Trabajo Fin de Máster. Esta consistió en realizar preguntas personales a los participantes, tales como su altura, peso y padecimiento de enfermedades. Posteriormente, se les evaluó utilizando el cuestionario DASS-21 (Osman et al., 2012). Si el participante deseaba ampliar o aclarar alguna respuesta, se le permitía hacerlo, dedicando así la mayor parte del tiempo a esta fase. Esta actividad tuvo una duración aproximada de 25 minutos, durante los cuales el participante permanecía sentado cómodamente, con los brazos sobre la mesa, moviéndolos ocasionalmente mientras hablaba. Todos los valores recogidos por los sensores de los relojes y los resultados del cuestionario DASS-21, son los que se utilizan en el desarrollo del proyecto.

[1] Un wearable es un dispositivo electrónico que se usa en el cuerpo humano y que interactúa con otros dispositivos para transmitir o recopilar datos. Los ejemplos más claros y más conocidos de wearables son los relojes inteligentes y las pulseras de actividad.

Resultados del experimento

En total, participaron 102 personas en el experimento; sin embargo, tras realizar el análisis de los datos fisiológicos y verificar la validez de las mediciones, el conjunto se redujo a 39 sujetos. Los datos de los sensores tuvieron que ser desechados en varios casos, especialmente al inicio de la experimentación, debido a errores en la transmisión y almacenamiento de datos, causados por el uso simultáneo de dos relojes recogiendo y enviando información al mismo dispositivo. Asimismo, no fue sino hasta tiempo después que se comenzó a recopilar los valores del sensor PPG, del cual se puede extraer información cardíaca.

El objetivo final de este experimento es recopilar estos datos para construir un modelo que, mediante técnicas de aprendizaje automático, sea capaz de detectar estrés crónico en nuevos sujetos.

2.3. Sensores para medir el estrés

Como el estrés deriva en múltiples consecuencias fisiológicas, se pueden usar sensores para lograr su detección. A continuación, se describen los sensores que existen en la actualidad y que se usan para medir el estrés.

- **Sensor ECG**. Es capaz de obtener la actividad eléctrica del corazón, pudiendo representarse de forma gráfica a modo de electrocardiograma. Normalmente, de esta actividad eléctrica se pueden distinguir tres tipos de ondas: la onda P, que representa la despolarización auricular (Cadogan & Buttner, 2022); el complejo QRS, asociado a la parte del electrocardiograma que muestra el proceso en el que los ventrículos del corazón se activan eléctricamente para contraerse y bombear sangre (Larkin, 2021); y la onda T, que refleja el proceso en el que el músculo cardíaco de los ventrículos recupera su estado eléctrico original (reposo) después de la contracción (Burns & Buttner, 2021). La mayoría de los estudios sobre la actividad cardíaca están relacionados con tres componentes del corazón: el dominio del tiempo, el dominio de la frecuencia y las características no lineales del corazón. La investigación en el dominio del tiempo se centra en parámetros como la frecuencia cardíaca (FC), los intervalos entre latidos (RR) y la variabilidad de frecuencia cardíaca (VFC). (Hernando et al., 2018) Para los intervalos RR se pueden estudiar tanto su valor medio, como la desviación estándar o la raíz cuadrada de las medias. (Chen et al., 2021) (Ahn et al., 2019)
- **Sensor EEG**. El sensor de electroencefalografía es capaz de obtener la actividad eléctrica del cerebro. Cuando una persona se somete a estrés, el cerebro activa muchos sistemas secretores de neuropéptidos en respuesta. Como resultado de esta activación, se liberan hormonas corticosteroides suprarrenales, que se conocen como "hormonas del estrés". Esta actividad es la que se puede medir para conocer el estrés de dicha persona.(Subhani et al., s.f.)

- **Sensor EMG**. El sensor de electromiografía permite medir la actividad eléctrica de los músculos. Existe un estudio relacionado con la tensión que se acumula en el trapecio, en el que se demuestra cómo la señal EMG posee amplitudes significativamente mayores durante el estrés en comparación con el descanso, y menos lagunas (períodos de relajación). (Wijsman et al., 2010)

- **Sensor EDA**. El sensor de actividad electrodérmica mide los cambios de conductividad producidos en la piel debido al aumento de la actividad de las glándulas sudoríparas. En la actualidad, existen dispositivos que lo incorporan para medir el nivel de sudoración en las manos, pues está directamente relacionado con el estrés. (Gradl et al., 2019).

- **Sensor PPG**. El sensor de fotopletismografía realiza una medición óptica de forma sencilla y barata, y se utiliza a menudo para controlar la frecuencia cardíaca. Es un sensor no invasivo que utiliza una fuente de luz y un fotodetector en la superficie de la piel para medir las variaciones volumétricas de la circulación sanguínea. Mediante la aplicación de diversas técnicas de limpiado de señal, podemos obtener la Variación de Frecuencia Cardíaca (VFC). Existen varios estudios que consiguen medir el nivel de estrés basándose en la VFC obtenida de esta señal. (Heo et al., 2021a) (F. Li et al., 2018)

2.4. Detección de estrés mediante sensor PPG

En los últimos años la demanda de dispositivos *wearables* que monitoricen nuestra salud en tiempo real ha crecido bastante, como se puede observar en la figura 2.1 y, como se ha visto en la sección anterior, la detección de estrés a través de señales fisiológicas mediante dispositivos *wearables* ya es posible.

Por otro lado, la señal PPG representa una onda arterial variable durante cada ciclo cardíaco, y los datos se recogen mediante de sensores en la muñeca. (Greene et al., 2016) Por consiguiente la incorporación de este sensor en dispositivos *wearables* es barata y beneficiosa. La mayoría de estudios sobre medición en estrés que se hicieron anteriormente utilizaron señales ECG ya que la señal PPG contiene más ruido que la ECG. No obstante, se han realizado varios estudios (Benchekroun et al., 2022; Mohan et al., 2016) que constatan la posibilidad de medir eficazmente el nivel de estrés a partir de una sola señal PPG, ya que es más simple y más apropiada para la vida cotidiana.

La figura 2.2 muestra la forma que tiene una señal PPG típica. Sin embargo, cuando el usuario lleva este sensor en su muñeca y está realizando actividades cotidianas, la señal se muestra con bastante ruido e incluso con zonas valle en las que no se ha medido nada, como se puede apreciar en la figura 2.3. Esto hace que la detección de picos, de sístole y diástole, y de los valles sea muy complicada. Para esto, existe una librería llamada HeartPy (van Gent, 2019) que provee métodos para la limpieza de ruido y procesamiento de señales PPG obtenidas desde un *smartwatch*.

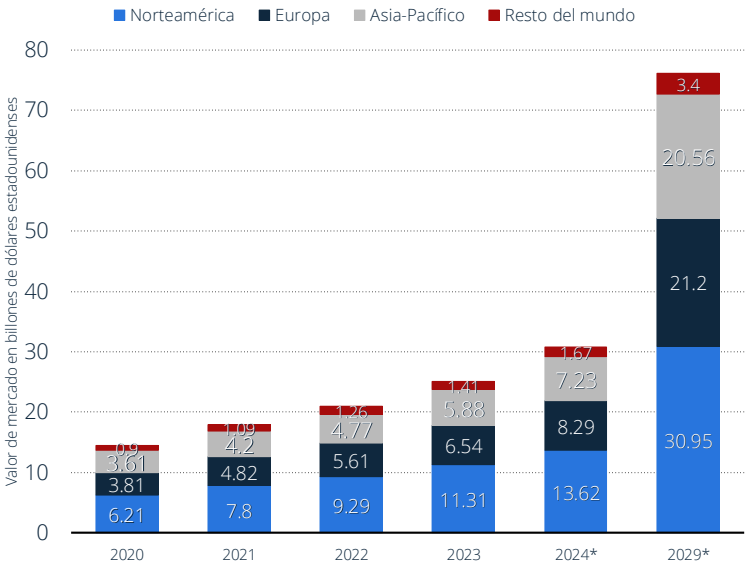


Figura 2.1: Predicción de ingresos del mercado de dispositivos *wearables* médicos en todo el mundo entre 2020 y 2029. (Statista, 2024)

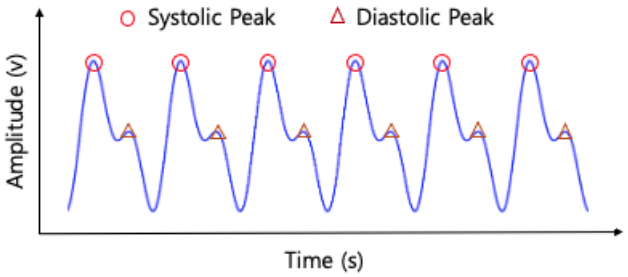


Figura 2.2: Señal PPG característica. (Heo et al., 2021b)

Para obtener el estrés de una señal PPG se debe atender a la variación de frecuencia cardíaca (VFC) (Shaffer & Ginsberg, 2017). La VFC se calcula a partir de los cambios en los intervalos de picos sucesivos (McCraty & Shaffer, 2015). Las señales deben dividirse en tamaños de ventana adecuados para calcular la VFC. Las características más usadas se listan en la tabla 2.1.

ABREVIATURA	DESCRIPCIÓN	UNIDAD	ECUACIÓN
MEAN_RR	Media de todos los intervalos RR	*ms*	$\frac{1}{N}\sum_{i=1}^{N} RR_i$
MEDIAN_RR	Mediana de todos los intervalos RR	*ms*	Mediana de los valores RR_i

SDRR	Desviación estándar de los intervalos RR	ms	$\frac{\sqrt{\sum_{i=1}^{N-1}(RR_{i+1}-RR_i)^2}}{N-1}$
RMSSD	Raíz cuadrada de la media de la suma de los cuadrados de la diferencia entre intervalos RR adyacentes.	ms	$\sqrt{\frac{\sum_{i=1}^{N-1}(RR_{i+1}-RR_i)^2}{N-1}}$
SDSD	Desviación estándar de todo el intervalo de diferencias entre intervalos RR adyacentes	ms	$\sqrt{\frac{\sum_{i=1}^{N-1}\text{variance}(RR_{i+1}-RR_i)}{N}}$
SDRR_RMSSD	Ratio (división) entre SDRR y RMSSD	-	$\frac{SDRR}{RMSSD}$
HR	Frecuencia cardíaca (latidos por minuto)	$l/$-min	$\frac{60}{\text{MEAN\_RR (en segundos)}}$
pNN50	% de intervalos RR adyacentes que difieren en más de 50 ms	$\%$	$\frac{\sum_{i=1}^{N}(\lvert R_i-R_{i+1}\rvert>50\text{ ms})}{N-1}$
SD1	Descriptor del gráfico de Poincaré de la VFC (Var. Frec. Card.) a corto plazo	ms	$\sqrt{\text{variance}\left(\frac{RR_i-RR_{i+1}}{\sqrt{2}}\right)}$
SD2	Descriptor del diagrama de Poincaré del H a largo plazo	ms	$\sqrt{\text{variance}\left(\frac{RR_i+RR_{i+1}}{\sqrt{2}}\right)}$
MEAN_REL_RR*	Media de todos los intervalos RR relativos	ms	$\frac{1}{N}\sum_{i=1}^{N} RR_{\text{rel},i}$
MEDIAN_REL_RR*	Mediana de todos los intervalos RR relativos	ms	Mediana de los valores $RR_{\text{rel},i}$
SDRR_REL_RR*	Desviación estándar de todo el intervalo RR relativo	ms	$\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(RR_{\text{rel},i}-\overline{RR_{\text{rel}}})^2}$
RMSSD_REL_RR*	Raíz cuadrada de la media de la suma de los cuadrados de la diferencia entre el intervalo RR relativo adyacente	ms	$\sqrt{\frac{\sum_{i=1}^{N-1}(RR_{\text{rel},i+1}-RR_{\text{rel},i})^2}{N-1}}$
SDRR_RMSSD_REL_RR*	Relación entre SDRR_REL_RR y RMSSD_REL_RR	-	$\frac{\text{SDRR\_REL\_RR}}{\text{RMSSD\_REL\_RR}}$

Cuadro 2.1: Características de la VFC más relevantes para predecir el nivel de estrés

Para la obtención de las características marcadas en la tabla 2.1 mediante un asterisco, es necesario implementar sus cálculos modificando la librería HeartPy (van Gent, 2019), que es posible gracias a que posee una licencia MIT. El resto de las características se obtienen directamente de la implementación base de esta librería.

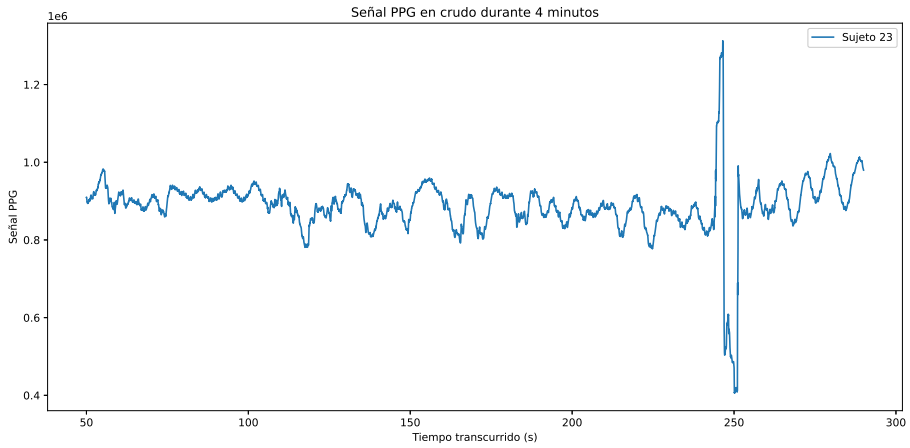


Figura 2.3: 4 minutos de una medición real realizada con el *smartwatch* cedido para este proyecto.

2.5. Trabajos previos sobre tratamiento de señales y datos

2.5.1. Librería Python para el tratamiento de señales

En esta sección, se analizarán los trabajos que se han realizado anteriormente y que tienen relación con el presente trabajo fin de máster.

Para comenzar, la librería HeartPy (van Gent, 2019) nos permite cargar un conjunto de señales PPG en crudo, para su posterior procesamiento. Es necesario indicarle cuál es el ratio de muestreo, por lo que, en nuestro caso, al ser un *smartwatch* puede presentar variaciones. Para resolver este inconveniente, todos los datos que se recojan se han de etiquetar con una etiqueta de tiempo, de modo que la librería sea capaz de detectar cuál es el ratio de muestreo. Es importante recalcar que, según el *smartwatch* con el que se trabaje, se obtiene un ratio distinto. Por esto, la librería indica que para determinar la variación de frecuencia cardíaca (VFC) se necesitaría al menos una señal de 100 Hz.

Habiendo hecho esto, la librería es capaz de detectar los picos sistólicos y de calcular valores relacionados con la variación de frecuencia cardíaca (VFC), al igual que las pulsaciones por minuto (PPM) de las distintas ventanas temporales que componen una medición completa.

Para la comprobación de su funcionamiento, se referencian tres ejemplos de interés en cuanto a la aplicación de esta librería sobre un conjunto de mediciones PPG. (Paulvangentcom, 2019) (Paulvangentcom, 2021) (Paulvangentcom, s.f.)

2.5.2. Conjuntos de datos sobre estrés existentes

SWELL: Smart Reasoning for Well-being at Home and at Work

En segundo lugar, profundizamos en el estado actual de la predicción de nivel de estrés en base a las características expuestas en la tabla 2.1. Unos investigadores de la *Facultad de Computación y Ciencias de la Información de la Universidad de Radboud*, en los Países Bajos, realizaron en 2014 un proyecto, denominado SWELL (Koldijk et al., 2018), para detectar situaciones de estrés mientras se trabajaba en oficinas, mediante sensores que no son intrusivos. En este proyecto se obtuvo un conjunto de datos de distintos sensores entre los que se encontraba un *Microsoft Kinect* (Cong & Winters, 2017), para la detección de posturas, un sensor para medir la variación de frecuencias cardíacas y otro sensor para medir la conductividad eléctrica de la piel.

En este experimento participaron 25 personas realizando trabajos de oficina típicos, como escribir informes, hacer presentaciones, leer el correo electrónico y buscar información, entre otros. Los factores que usaron para manipular sus condiciones laborales, y así, inducir situaciones de estrés, fueron tanto interrupciones mediante correos electrónicos y la presión por el límite de tiempo. La experiencia subjetiva de los participantes sobre la carga de trabajo que tenían, el esfuerzo mental que requería, sus emociones y el estrés que percibían se evaluaron mediante cuestionarios que se tomaron como válidos de base para el experimento.

WESAD: Wearable Stress and Affect Detection

En esta misma línea de investigación, el conjunto de datos WESAD (Wearable Stress and Affect Detection) (Schmidt et al., 2018), introducido en 2018 por el Instituto de Tecnología de Karlsruhe, se ha posicionado como referencia clave para la evaluación del estrés y las emociones mediante wearables. A diferencia del proyecto SWELL, que se enfocaba en entornos de oficina, WESAD adopta un enfoque multimodal que combina señales fisiológicas y de movimiento para modelar estados emocionales en un entorno controlado de laboratorio.

Para el experimento del WESAD se cuenta con 15 participantes que se someten a tres estados diferentes: un estado basal neutral, un estado de estrés inducido a través del Trier Social Stress Test (TSST) (Labuschagne et al., 2019) y un estado de diversión generado mediante videos humorísticos. Durante estas sesiones, se emplearon dos dispositivos: el RespiBAN Professional, colocado en el pecho, y una Empatica E4, usada en la muñeca. Estos dispositivos capturaron una amplia gama de datos, como actividad electrodérmica (EDA), frecuencia cardíaca, variabilidad de la frecuencia cardíaca (VFC), patrones de respiración, temperatura de la piel y aceleración en tres ejes.

WESAD incluye también etiquetado que complementa las mediciones objetivas, permitiendo el desarrollo de algoritmos más robustos que integran tanto dimensiones objetivas como subjetivas del estrés.

Combinación de SWELL y WESAD

Siguiendo esta línea de investigación y con el objetivo de mejorar la robustez en el análisis del estrés, se han combinado los conjuntos de datos SWELL y WESAD basándonos en el enfoque propuesto por (Dahal et al., 2023). Este estudio presenta un modelo global de detección del estrés que utiliza ocho características de la variabilidad de la frecuencia cardíaca (VFC) seleccionadas mediante el método de mínima redundancia y máxima relevancia (mRMR), junto con un algoritmo de *Machine Learning*, *Random Forest*, para la clasificación.

La combinación de ambos conjuntos de datos permite aprovechar las fortalezas y complementariedades de cada uno, aumentando la diversidad y volumen de los datos disponibles para el entrenamiento del modelo. Al integrar las características más relevantes de VFC de ambos conjuntos, se puede desarrollar una versión más robusta para el análisis del estrés que mejora la precisión en la detección y clasificación de los niveles de estrés en diferentes contextos laborales y situaciones emocionales. Este enfoque nos acerca a la implementación de modelos de detección de estrés más generalizables y efectivos en entornos reales, contribuyendo al bienestar laboral y a la identificación temprana de factores estresantes.

2.5.3. Trabajo Fin de Grado previo

En el Trabajo Fin de Grado *"Control de estrés en mayores: Aplicación para SmartWatch"* (Rubia López, 2022), se abordó el tratamiento de señales fisiológicas con un enfoque específico en la medición y análisis del estrés mediante una arquitectura multisistema compuesta por un smartwatch, un dispositivo móvil y una plataforma web. Este proyecto tenía como objetivo principal monitorizar los niveles de estrés en personas mayores durante actividades que implicasen el uso del transporte público.

Para ello, se utilizó la señal de fotopletismografía (PPG) capturada a través de un smartwatch, procesándola con herramientas específicas y basándose en los trabajos previos que se han expuesto en puntos anteriores.

Entre las principales contribuciones relacionadas con el tratamiento de señales, podemos encontrar que el trabajo se servía del dataset SWELL como punto de partida para entrenar y validar modelos predictivos de estrés.

Para el análisis de las señales PPG, se empleó la librería HeartPy (van Gent, 2019), especializada en el procesamiento de datos de fotopletismografía. A través del uso de esta librería se pueden extraer las métricas, filtrando artefactos y ruido a través de algoritmos de corrección en señales necesarias para poder predecir, a posteriori, el nivel de estrés en base al modelo creado.

Sabiendo qué métricas se podían obtener de esta señal, se entrenaron modelos predictivos usando el dataset SWELL. Entre los algoritmos de Machine Learning que se probaron se incluyen Random Forest y Support Vector Machines. Los modelos demos-

traron una buena capacidad para predecir niveles de estrés, validando su aplicabilidad en contextos reales.

La integración de HeartPy y los datos de SWELL permitió un enfoque robusto y bien fundamentado en el análisis de señales PPG, destacando las ventajas de utilizar herramientas automatizadas para procesar grandes volúmenes de datos, mejorando la precisión y reduciendo el esfuerzo manual en el análisis.

Durante el proyecto, se llevaron a cabo experimentos prácticos donde sujetos utilizaron el sistema en actividades que implicaban el desplazamiento en transporte público. Los datos recogidos se analizaron para comparar las predicciones con el estrés percibido por los usuarios, evidenciando una correlación significativa.

No obstante, este proyecto anterior se limitaba a monitorizar el estrés únicamente durante la actividad de uso de medios de transporte, resultando necesario expandirlo para abarcar más situaciones y actividades que también puedan generar estrés. Además, al generar el modelo de clasificación de estrés, se empleó principalmente el conjunto de datos SWELL, el cual está sesgado hacia situaciones de oficina y cuenta con un número reducido de muestras. Por ello, se plantea la utilización de un nuevo conjunto de datos, como la combinación con el conjunto del WESAD, para enriquecer el modelo con mayor variedad de contextos y aumentar la robustez y generalización de los resultados.

2.6. Metodologías que se pueden utilizar

Para comenzar esta sección, veremos que se pueden encontrar metodologías diversas en cuanto al desarrollo de software, en cuanto a patrones arquitectónicos de diseño de software y en cuanto al diseño del software.

2.6.1. Metodologías de desarrollo software

En las tablas 2.2, 2.3, 2.4 se hace una comparativa sobre las metodologías de desarrollo de software que exponen las características de cada una, junto con sus fortalezas y sus debilidades. Esto se realiza atendiendo a varios criterios como tipologías de proyectos, sus tamaños, el entorno de desarrollo y la disponibilidad de recursos.

METODOLOGÍA	CARACTERÍSTICAS
Cascada	La documentación debe ser bastante detallada y se debe realizar una planificación muy cuidada. Al ser un proceso lineal, cada fase tiene sus propios entregables.
Iterativo e incremental	El Product Owner del proyecto se involucra totalmente. Se tiene un número de iteraciones construidas en él como modelo inicial. Buen diseño por encima de buena documentación.
Programación extrema	El Product Owner decide qué tarea debe iniciarse primero. Se acelera la publicación de resultados. Se hacen pruebas unitarias. Contacto con el Product Owner de forma continua, debido a que se trabaja con él.
Ágil: Scrum	Desarrollo iterativo. Reuniones diarias. El equipo de desarrollo es auto-organizado

Cuadro 2.2: Comparativa de las características de metodologías de desarrollo software más comunes. (Saeed et al., 2019)

METODOLOGÍA	FORTALEZAS
Cascada	Sus fases son simples y están muy bien definidas, es simple de manejar y fácil de entender. Encaja mejor con proyectos pequeños.
Iterativo e incremental	El Product Owner del proyecto da su opinión constantemente. Se realizan numerosas revisiones durante el proyecto. El código se entrega en fases tempranas del proyecto. Encaja mejor con proyectos de medio y gran tamaño.
Programación extrema	Todo se hace rápido. El código se entrega en fases tempranas del proyecto. Se obtiene retroalimentación continua por parte del Product Owner. Encaja bien con proyectos tanto pequeños, como medianos y grandes.
Ágil: Scrum	Los productos se entregan en poco tiempo. Se obtiene retroalimentación de forma continua por parte del Product Owner. La inclusión de nuevos requisitos es rápida. Encaja bien con proyectos tanto pequeños, como medianos y grandes.

Cuadro 2.3: Comparativa de las fortalezas de metodologías de desarrollo software más comunes. (Saeed et al., 2019)

METODOLOGÍA	DEBILIDADES
Cascada	El código del proyecto suele entregarse tarde, no gestionándose bien la inclusión de nuevos requisitos, y no permite mucho cambio para resolver errores de diseño o planificación.
Iterativo e incremental	Cada iteración es inflexible pues sus fases son como un proyecto en cascada a pequeña escala.
Programación extrema	Falta documentación. Los desarrolladores no están dispuestos a hacer programación en pareja. Los programadores no están dispuestos a escribir pruebas antes de desarrollar código. Es necesario reunirse con frecuencia.
Scrum	La necesidad de desarrolladores con experiencia previa. La escasez de documentación. Es difícil estimar el coste y el tiempo al principio de proyectos grandes.

Cuadro 2.4: Comparativa de las debilidades de metodologías de desarrollo software más comunes. (Saeed et al., 2019)

2.6.2. Patrones arquitectónicos de diseño software

Como en nuestro proyecto desarrollaremos varias aplicaciones (móvil, reloj inteligente y web) que deben poder comunicarse entre sí, se deben tener en cuenta los patrones arquitectónicos de diseño software que existen en la actualidad, y cuáles son las recomendaciones para cada lenguaje de programación. El propósito de seguir un patrón de diseño es el de incrementar la modularidad, la flexibilidad, la testeabilidad y la mantenibilidad del presente proyecto.

En cuanto a patrones de diseño software para aplicaciones de Android, pues en nuestro caso será este el sistema del *smartwatch* y del *smartphone* vinculado, encontramos que los más extendidos son, el patrón Modelo-Vista-Controlador (MVC), el patrón Modelo-Vista-Presentador (MVP) y el patrón Modelo-Vista-ViewModel (MVVM). En la tabla 2.5 se exponen las distintas características que poseen, y en la figura 2.4 se ilustran las relaciones entre vista, controlador, modelo, presentador y ViewModel.

Las principales diferencias que existen en cuanto a estos patrones arquitectónicos se pueden clasificar de acuerdo con los siguientes puntos (Lalani, 2022):

- **Arquitectura**. Los tres modelos se construyen de forma diferente. En MVC, la capa superior de la arquitectura es la vista, que se integra con el controlador y la capa del modelo se encuentra por debajo de estos. En MVP ocurre lo mismo, salvo que la vista da acceso directo al modelo. Esta exposición a toda la vista del modelo puede dar algunos problemas de seguridad, por ello, MVVM los solventa mediante una construcción diferente y la limitación de exposición entre cada capa.

- **Rendimiento**. Durante pruebas de rendimiento de Interfaz de Usuario, MVP es el mejor por tener una mayor fiabilidad. Lo que la diferencia del resto es que no pone obstáculos mientras se renderizan los cuadros de la aplicación. Por el contrario, MVVM está un poco por detrás en rendimiento debido a que tiene que hacer un

PATRÓN	CARACTERÍSTICAS
MVC	El punto de entrada es con el Controlador. Relación muchos a muchos entre la Vista y el Controlador. La Vista no conoce al Controlador. La Vista conoce qué debe recibir del Modelo.
MVP	El punto de entrada es con la Vista. Relación uno a uno entre la Vista y el Presentador. La Vista tiene una referencia a su Presentador y el Presentador conoce sus Vistas. La vista no conoce al modelo. El Presentador actualiza el modelo.
MVVM	El punto de entrada es con el Controlador. Relación uno a uno con el ViewModel y la Vista. El ViewModel no conoce a sus Vistas. La Vista no conoce al modelo. El ViewModel actualiza la vista

Cuadro 2.5: Características de las tres de arquitecturas de desarrollo software más extendidas en desarrollo en Android. (Dang, 2020)

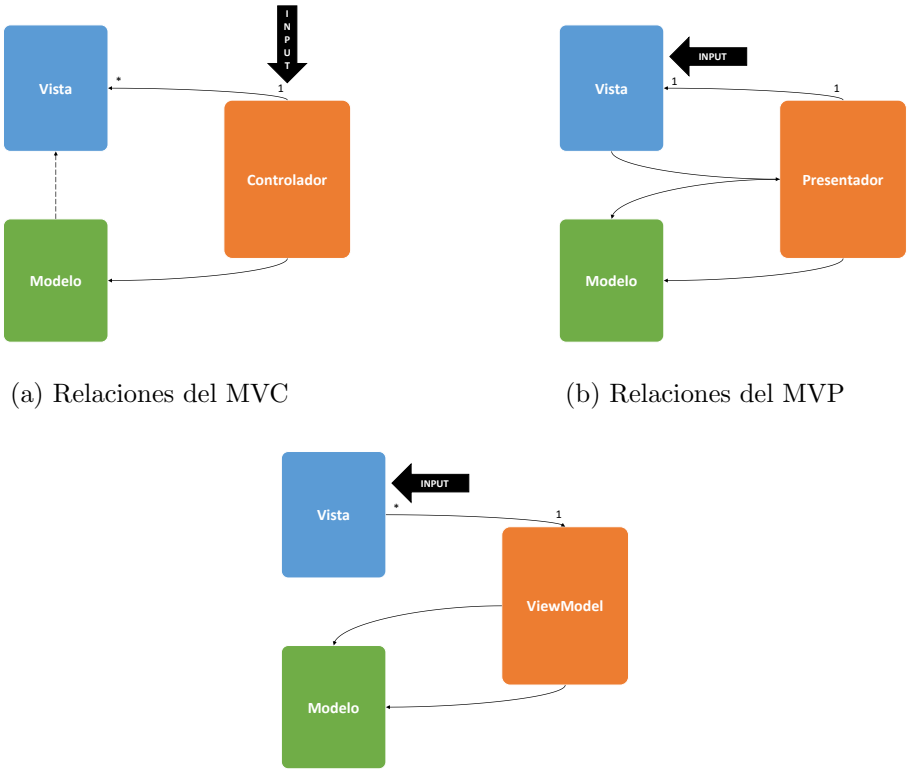
mayor procesamiento. Por ejemplo, el Data Binding[2] crea una sobrecarga adicional que puede afectar drásticamente al rendimiento cuando se trata de realizar tareas complejas. Por otro lado, MVC presenta un rendimiento en medio de estos dos, ocupando una posición intermedia en cuanto a la evaluación del rendimiento.

- **Reusabilidad**. En cuanto a reusabilidad, MVVM se pone a la cabeza. Ofrece la mejor reusabilidad, especialmente cuando estamos procesando datos. Aunque el Data Binding no sea bueno para el rendimiento, tiene un impacto positivo en la compatibilidad. Tras este se situaría MVP, ya que MVC ocuparía el último lugar debido a su reusabilidad limitada y a sus problemas de mantener el estado de la aplicación.

- **Modularidad**. Tanto MVP como MVVM se sitúan a la cabeza, y es este último el que intenta separar el desarrollo de la Interfaz de Usuario de la lógica de negocio. El MVC tiene limitaciones en la modularización.

- **Tamaño de proyecto**. El patrón MVC es adecuado para proyectos pequeños, a diferencia del patrón MVVM que es conveniente para proyectos grandes. El patrón MVP sería adecuado tanto para proyectos pequeños como grandes.

[2]Vincula los componentes de la IU en los diseños con las fuentes de datos de la aplicación mediante un formato declarativo.



(a) Relaciones del MVC

(b) Relaciones del MVP

(c) Relaciones del MVVM

Figura 2.4: Principales patrones arquitectónicos de diseño software en Android. (Dang, 2020)

Dada esta información sobre las tres arquitecturas, resulta necesario exponer las recomendaciones que Google detalla en su *Guía para Desarrolladores Android* ("Guide to app architecture", s.f.).

Entre los principios recomendados por Android respecto a la arquitectura que se ha de seguir, encontramos la modularidad. Es un error común escribir todo el código en una *Activity*[3]. Las clases de Interfaz de Usuario, como la *Activity*, deben contener únicamente la lógica con la que interactúa el usuario. Debemos mantener estas clases lo más limpias posibles, para así evitar problemas de ciclos de vida y mejorar la testeabilidad de las clases.

Adicionalmente, en la misma guía, se indica que se deben usar modelos de datos persistentes, que son independientes de la UI y de otros componentes de una aplicación.

[3]Una *Activity*(actividad) en Android es una pantalla de la interfaz de usuario de la aplicación Android. En ese sentido, una actividad de Android es muy similar a las ventanas de una aplicación de escritorio. Una aplicación Android puede contener una o más actividades, es decir, una o más pantallas. La aplicación Android comienza mostrando la actividad principal, y a partir de ahí la aplicación puede permitir abrir actividades adicionales.

Estos modelos son ideales debido a que la aplicación puede seguir funcionando en caso de que una red de conexión no esté disponible. Si nos basamos en este principio, la aplicación será más robusta y testeable.

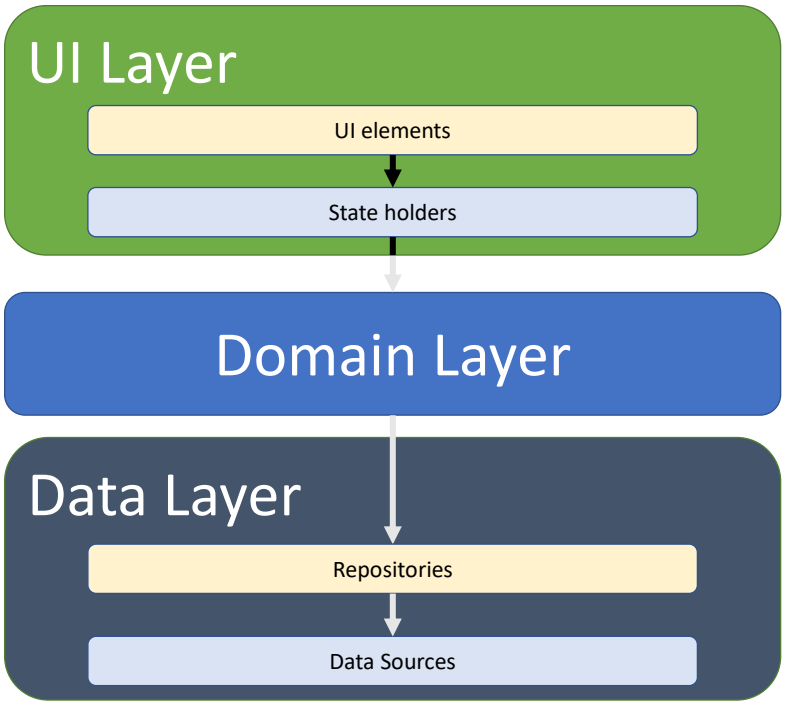


Figura 2.5: Diagrama de la arquitectura de una aplicación Android típica. ("Guide to app architecture", s.f.)

En cuanto a las capas arquitectónicas que debe tener una aplicación, encontramos la capa de Interfaz de Usuario (UI Layer), encargada de mostrar los datos de la aplicación en la pantalla, la capa de Dominio (Domain layer), para simplificar y reusar las interacciones entre la capa de UI y la de Datos, y por último la capa de Datos (Data Layer), encargada de contener la lógica de negocio de la aplicación y los datos. Estas se pueden ver representadas en la figura 2.5.

La capa de Interfaz de Usuario se compone tanto de los elementos que se renderizan en la pantalla, como los State Holders[4], que manejan los datos y los exponen a la interfaz de usuario, gestionando también la lógica.

En cuanto a la capa de Datos, es importante señalar que esta contiene la lógica de negocio y determina cómo la aplicación crea, almacena y modifica los datos. Bajo la

[4]Un ViewModel puede mantener el estado de los componentes en una Vista. El ViewModel proporciona a la UI acceso a las otras capas, como la de dominio y la de datos. Entonces se pueden definir las variables de estado usando LiveData, y definir los métodos que cambiarán el estado de estas variables. Por esto se les llama State Holders.

filosofía de Clean Architecture y el patrón de diseño MVVM, se recomienda contar con un repositorio para cada tipo de información que se maneje en la aplicación. Por ejemplo, si se deben procesar datos de diferentes sensores, como el GPS, PPG, acelerómetro, entre otros, cada uno requerirá su propio repositorio para centralizar y exponer dichos datos al resto de la aplicación, resolver posibles conflictos con múltiples fuentes de datos y abstraerlas del resto de los componentes. De esta manera, cada repositorio se convierte en el responsable de la lógica de negocio y la administración de sus datos específicos, asegurando un diseño modular, claro y fácil de mantener.

Por último, se explica la capa de Dominio, que es la encargada de encapsular lógicas de negocio que son reusadas por varios ViewModel. Esta capa favorece la reusabilidad y aminora la complejidad.

Además se debe usar inyección de dependencias para mejorar la reusabilidad, la refactorización y la facilidad de testeo ("Dependency injection in Android", s.f.) pues en las aplicaciones hay clases que dependen de otras para funcionar correctamente.

2.6.3. Diseño de la Interfaz en Android

Es fundamental prestar especial atención al diseño de la interfaz, ya que la aplicación va a ser utilizada por personas mayores. Seguir unas guías de diseño estandarizadas ayuda a mejorar la usabilidad para este tipo de usuarios, asegurando una interacción más intuitiva y accesible, e impactando positivamente en la experiencia de uso.

En la actualidad, Android posee unas directrices de diseño que se acuñan como Material Design. (Google, 2014) A finales de 2021 y comienzos de 2022, sufrió la última revisión, alcanzando la versión 3. Seguir estas normas asegura que el diseño de nuestras aplicaciones sea accesible, universal y consistente, pudiendo diferenciar y entender los distintos componentes de la interfaz.

Además, al ser estas directrices las más generalizadas en torno al mercado de aplicaciones Android, se aprovecha un lenguaje visual bien conocido. Esto hace que, subconscientemente, los usuarios tengan un mayor nivel de confianza y seguridad en la aplicación, ya que se asocia con Google. (Woodhead, 2018)

Asimismo, con el uso de la versión 3 de Material Design, no debemos preocuparnos por la gama de colores a usar en nuestra aplicación, ya que es el sistema el encargado de escogerlo en función de la configuración escogida por el usuario.

2.7. Tecnologías que se pueden utilizar

En cuanto a tecnologías que podemos usar para el desarrollo del presente Trabajo Fin de Máster encontramos tres categorías. Las tecnologías disponibles para el desarrollo de aplicaciones de la web, para *smartphone* Android y para *smartwatch* WearOS.

2.7.1. Tecnologías para desarrollar aplicaciones basadas en Web

Para desarrollar una aplicación web debemos tener en cuenta la parte del cliente (frontend) y el del lado del servidor (backend). Además de estas dos, debemos atender también a las tecnologías de base de datos que existen actualmente.

Tecnologías para la parte del cliente (frontend)

Las tecnologías que se usan para el desarrollo frontend son principalmente Angular, Express.js, Vue.js, HTML, CSS o SCSS, JavaScript, jQuery y Next.JS entre otras (Kohan, 2022).

Las tecnologías mencionadas se pueden dividir en frameworks, lenguajes y librerías. En cuanto a frameworks se describen los mencionados:

- **Angular**. Permite el desarrollo de aplicaciones para todas las plataformas. Es un *framework* basado en componentes, de código libre, que usa *TypeScript* como lenguaje para su programación basada en componentes creando aplicaciones web escalables. Posee una colección de bibliotecas bien integradas y un conjunto de herramientas que ayudan al desarrollo y pruebas.
- **Express.js**. Es un framework de JavaScript que crea un entorno para desarrollar aplicaciones de tipo servidor especialmente aquellas que sirven una API.
- **Next.JS (React)**. React es una librería de JavaScript basada en componentes para construir interfaces de usuario de forma declarativa. Cuando se combina con Next.js, un framework que extiende React, se obtienen herramientas para renderizado del lado del servidor (*server-side rendering*) y generación de sitios estáticos (*static site generation*), lo que permite crear aplicaciones web altamente optimizadas para SEO y rendimiento.

En cuanto a los lenguajes se describen los mencionados:

- **HTML**. Es un lenguaje de marcado que se usa para estructurar una página web y su contenido. Los Frameworks mencionados anteriormente utilizan este lenguaje para su funcionamiento.
- **CSS / SCSS**. Es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado como HTML.
- **JavaScript**. Es un lenguaje de programación que permite implementar funciones complejas en páginas web estáticas, de modo que actualicen su contenido dinámicamente, se puedan mostrar mapas interactivos, gráficos 2D y 3D, entre otros.

Por último, las librerías mencionadas se describen a continuación:

- **Leaflet**. Es una librería para crear mapas interactivos. Permite integrar mapas dinámicos con funcionalidades como marcadores, capas y eventos personalizados, todo ello utilizando datos de servicios como OpenStreetMap.

- **Chart.js**. Es una librería para crear gráficos en la web. Ofrece una amplia variedad de tipos de gráficos, como de barras o de líneas, y está basado en HTML5 renderizarlo.

- **Uppy**. Es una librería para gestionar la subida de archivos. Ofrece una interfaz fácil de usar, soporte para múltiples fuentes (como almacenamiento local, URL y servicios en la nube), y opciones para la previsualización y edición de archivos antes de su carga.

- **jQuery Sortable**. Es un complemento de jQuery que permite crear listas y elementos reordenables mediante *drag-and-drop*. Mejora la experiencia de usuario donde sea necesario modificar el orden de elementos de forma dinámica.

Tecnologías para la parte del servidor (backend)

Para el backend, encontramos tecnologías como ASP.NET, Ruby on Rails, Flask, Django, PHP y Perl, entre otros. (Kohan, 2022) En cuanto a los Frameworks mencionados, se describirán a continuación:

- **ASP.NET**. Es un framework basado en .NET y C#, que se diferencia del resto, por ser de código abierto y multiplataforma para crear aplicaciones web dinámicas. Puede usarse tanto en proyectos grandes como pequeños y tiene una gran comunidad de soporte detrás.

- **Ruby on Rails**. Es un framework basado en Ruby, factor diferencial, que sigue el paradigma del patrón MVC. Aboga por la simplicidad, brindando la posibilidad de desarrollar aplicaciones del mundo real con menos código que otros Frameworks y con una configuración mínima.

- **Flask**. Es un framework basado en Python minimalista que permite crear aplicaciones web basadas en componentes, por lo que es altamente escalable y su configuración es sencilla.

- **Django**. Es otro framework que se basa en Python muy configurable y aún más escalable que Flask. Tiene múltiples librerías que hacen que tareas básicas de implementación de funciones en sitios web como Autentificación o creación de APIs REST sean realmente simples.

A continuación se describen los lenguajes de programación mencionados:

- **PHP**. Es un lenguaje de programación de código abierto que se usa en desarrollo web y puede ser incrustado en HTML. Lo que le distingue de los Frameworks que se han desarrollado anteriormente, es que el código es ejecutado en el servidor, generando HTML que se envía directamente al cliente, de modo que no se sepa el código subyacente.

- **Perl**. Es un lenguaje de programación interpretado de propósito general que solía ser el más popular en cuanto a desarrollo web debido a su capacidad de manipulación de texto y su rápido ciclo de desarrollo. En la actualidad existen Frameworks que permiten la creación de aplicaciones web modernas.

Tecnologías para bases de datos

De acuerdo con un estudio realizado recientemente, entre las bases de datos más extendidas en la actualidad, encontramos que MySQL, PostgreSQL, SQLite y MongoDB ocupan los primeros puestos. (StackOverflow, 2022)

Estos se pueden clasificar en cuanto a si son relacionales o no relacionales (NoSQL). Para su clasificación, se abordará primeramente la definición y las diferencias que existen entre ambas.

- **Bases de datos relacionales**. Una base de datos relacional almacena y proporciona acceso a datos que están relacionados entre sí. Se basa en el modelo relacional, que es una forma intuitiva y sencilla de representar datos en tablas. En este tipo de base de datos, cada fila de una tabla es un registro con una identificación única (clave). Las columnas de la tabla contienen los atributos de dichos datos y cada registro suele tener un valor para cada atributo, lo que permite establecer relaciones entre los datos. (Oracle, s.f.)

- **Bases de datos no relacionales**. Las bases de datos no relacionales almacenan datos de forma no tabular, basándose en estructuras de datos como los documentos. Un documento puede ser muy detallado y contener distintos tipos de información en diferentes formatos, por lo que es capaz de digerir y organizar varios tipos de datos de forma conjunta haciéndolo muy flexible. Se suelen usar cuando es necesario organizar grandes cantidades de datos complejos y diversos. (MongoDB, s.f.)

Las diferencias que existen entre ambas son que las bases de datos no relacionales son más flexibles que las relacionales ya que los datos no se limitan a un grupo estructurado y el acceso a datos en no relacionales es más rápido que en relacionales. Por otro lado, en bases de datos relacionales los datos se estructuran fácilmente en categorías, estando siempre en estado consistente y se pueden definir relaciones entre dichos datos de forma sencilla. (Pawlan, 2022)

Tecnologías de despliegue de aplicaciones web

Para lograr que una aplicación web que sea desarrollada se pueda visualizar desde fuera de la máquina en la que se desarrolla, su código, configuraciones y archivos necesitan ser desplegados en otra máquina con recursos suficientes que esté asociada a una url para poder acceder a ella desde cualquier lugar (Safwany, 2018). Existen varias formas de abordar esto. En cuanto a despliegue de aplicaciones web para desarrollo local encontramos que la opción más recurrida son los contenedores. Los contenedores son una forma de sistema operativo virtualizado. Un contenedor puede usarse para ejecutar desde pequeños microservicios hasta aplicaciones más grandes. Dentro de un contenedor se encuentran todos los ejecutables, código binario, librerías y archivos de configuración necesarios. Los contenedores tienen la ventaja de que son bastante ligeros y portables. ("What are containers?", 2022) En cuanto a despliegue de aplicaciones también se puede optar por el alquiler de soluciones en la nube como Google Cloud, Amazon Web Services, Heroku, Azure, entre otros, que hacen mucho más simple las tareas de configuración.

2.7.2. Tecnologías para desarrollar aplicaciones para *smartphone* Android

Para desarrollar aplicaciones para Android, debemos cumplir ciertos requisitos. El primero de ellos es tener en un ordenador el SDK de Android descargado. Este se puede descargar automáticamente al instalar Android Studio. Existen programas alternativos para el desarrollo de aplicaciones Android, como IntelliJ IDEA, Visual Studio Code y Eclipse, aunque Google recomienda usar su solución Android Studio desarrollada en conjunto con JetBrains.

Una vez hecho esto, es necesario valorar el estado del arte actual en cuanto a desarrollo Android. Existen hoy en día numerosos Frameworks que hacen que el desarrollo sea más rápido y sencillo. Entre estos, encontramos que los más extendidos son Flutter, Ionic, React Native y Angular.

Debemos tener en cuenta también que estos Frameworks trabajan sobre distintos lenguajes de programación. Flutter lo hace sobre Dart, Ionic sobre HTML5, y React Native junto con Angular sobre JavaScript/TypeScript.

En la Tabla 2.6 se presenta una comparativa entre algunos de los principales *Frameworks* de desarrollo de aplicaciones móviles para Android, tomando en cuenta diversos estudios que se han hecho (Biørn-Hansen et al., 2020; Mahendra & Anggorojati, 2021). Se han seleccionado los aspectos que se consideran más relevantes para quien deba tomar una decisión a la hora de elegir un *framework*:

- **Interfaz de Usuario**: Para indicar si se renderiza de forma nativa o a través de capas web.
- **Mercado y Comunidad**: Porque contar con una gran comunidad facilita encontrar documentación, ejemplos y soluciones.

- **Rendimiento**: Factor esencial que influye en la velocidad y fluidez de la aplicación.
- **Plataformas**: Muestra la compatibilidad con diferentes versiones de Android (e incluso otras plataformas).
- **Lenguajes**: Determina la curva de aprendizaje y la disponibilidad de desarrolladores.
- **Precio**: Aunque en su mayoría son gratuitos, algunos ofrecen planes de pago para ciertas funciones empresariales.

Las expresiones empleadas en la tabla, como *"No muy bueno"*, *"Muy bueno"* y *"El mejor"*, son etiquetas generales que sirven para transmitir, de forma resumida, el rendimiento o experiencia de usuario comparados entre los distintos *frameworks*. Por ejemplo, cuando se califica como *"No muy bueno"*, se hace referencia a limitaciones detectadas al compilar o ejecutar en entornos reales; *"Muy bueno"* indica un desempeño sólido y una experiencia fluida; mientras que *"El mejor"*, en el caso de Kotlin, alude a que, al ser un lenguaje nativo para Android, ofrece un rendimiento muy superior y un soporte oficial mayor.

	INTERFAZ DE USUARIO	MERCADO Y COMUNIDAD	RENDIMIENTO	PLATAFORMAS	LENGUAJES	PRECIO
REACT NATIVE	Nativa	La más grande	Muy cercano a nativo	Android 4.0.3+	JavaScript y React.JS	Gratuito
FLUTTER	Nativa	El segundo más grande	Muy bueno	Android 4.1.x+	Dart (basado en Java y C++)	Gratuito
IONIC	CSS y HTML no nativo	El tercero más grande	No muy bueno	Android 4.4+	CSS, HTML5 y JavaScript	Gratuito con versión de pago
KOTLIN	Nativa	Últimas posiciones, aunque creciendo rápidamente	El mejor	Android 1.0+	Es un lenguaje de propósito general	Gratuito

Cuadro 2.6: Comparativa de los *Frameworks* de desarrollo de aplicaciones Android más relevantes

Respecto a los lenguajes de programación usados en Android, tal y como indica Google en uno de sus artículos publicados en el blog de desarrolladores Android (Winer, 2019)), desde que lanzaron soporte para que Android ejecutase aplicaciones escritas en Kotlin, se ha incrementado en gran medida el número de desarrolladores que usan este lenguaje a día de hoy. Ellos establecen que en torno al 60 % del top 1.000 aplicaciones Android en 2019 contenían parte de su código escrito en Kotlin.

Adicionalmente, lo que valoran los desarrolladores de usar Kotlin para desarrollar sus aplicaciones en Android es que es un lenguaje expresivo y conciso, brinda más seguridad a las aplicaciones evitando errores de programación comunes, es interoperable, ya que se puede llamar código Kotlin desde Java o viceversa y, por último, tiene una simultaneidad estructurada, lo que hace que se simplifiquen las tareas de codificar funciones bloqueantes y asíncronas mediante corrutinas. Además, a diferencia de los Frameworks que existen en el mercado, Kotlin permite acceso a bajo nivel a funciones de la API de Android. (Google, s.f.-b)

En cuanto a diseño de Interfaces de Usuario, Google brinda a los desarrolladores varias opciones. Entre ellas nos encontramos una apuesta por el diseño declarativo de interfaces de usuario, dejando de lado al tradicional método imperativo (XML). Esta metodología se refleja tanto en la forma con la que Flutter diseña sus interfaces, como la nueva tecnología lanzada por Google llamada Jetpack Compose. (Yiğit, 2022) Jetpack Compose es un kit de herramientas para compilar Interfaces de Usuario nativas, simplificando y acelerando el desarrollo de las mismas en Android. Esto se consigue ya que se necesita menos código para mostrar los mismos componentes que con XML y es más intuitivo y legible, ya que es declarativo y por componentes[5].

2.7.3. Tecnologías para desarrollar aplicaciones para *smartwatch* WearOS

Al ser WearOS un sistema operativo basado en Android, la metodología para el desarrollo de aplicaciones para esta plataforma es parecida a la subsección anterior.

Sin embargo, al ser el reloj inteligente un dispositivo que tiene características distintas a las de un *smartphone*, los Frameworks que se pueden usar para agilizar el desarrollo de aplicaciones se reducen considerablemente.

Por un lado, encontramos que Flutter, de forma nativa, no tiene soporte para WearOS. Sin embargo, al tener una comunidad muy activa, han lanzado una librería para poder diseñar las interfaces adaptándose a las características de pantalla que poseen estos dispositivos. (Community, 2022)

En esencia, Flutter tiene un desarrollo bastante poco avanzado en cuanto a dispositivos *wearables* y no posee los componentes de interfaz de usuario recomendados por las directrices de diseño de Material Design para WearOS.

Como hemos comentado, en la actualidad, Material Design 3 se ha establecido como estándar de diseño de interfaces de Usuario y, según Google, la mejor opción es usar Jetpack Compose para incluir este diseño en las aplicaciones para WearOS. (Google, s.f.-a)

Por esto, también se recomienda en las guías para desarrolladores Android, elaborada por Google, que se use Kotlin como lenguaje de programación junto con Jetpack Compose para desarrollar aplicaciones para WearOS con versiones 2.0 en adelante. (Google, s.f.-c)

2.8. Trabajos relacionados

En la presente sección se analizan algunas evidencias científicas y revisiones recientes de uso de tecnologías o metodologías para resolver un problema similar al del presente

[5]Jetpack Compose permite desarrollar componentes pequeños, que serían partes de la Interfaz de Usuario, sin que estén vinculados a un *Activity* concreto, de modo que sea fácil su reutilización, su mantenimiento y su testeo.

trabajo fin de máster.

2.8.1. Estudios sobre detección de estrés mediante señal PPG

Comenzaremos por un proyecto desarrollado por el Instituto de Planificación y Evaluación de las Tecnologías de la Información y la Comunicación (IITP) financiado por el Ministerio de Ciencia del Gobierno Coreano, en el que tres investigadores consiguen detectar el estrés con un sensor PPG mediante la orquestación de múltiples métodos de eliminación de ruido y de detección de picos. En la figura 2.6 se puede ver el proceso general del método que han desarrollado. (Heo et al., 2021a)

El objetivo de este proyecto fue mejorar el rendimiento de la detección de estrés mediante el procesamiento de señales basadas en fotopletismografía. Debido a que esta señal se puede recoger de dispositivos *wearables* pero se ve afectada por ruidos tanto internos como externos, es necesario desarrollar un método de eliminación de ruidos. Se propone un método en dos pasos, filtrando primeramente el ruido en términos de frecuencia y posteriormente eliminándolo en términos de tiempo. También se desarrolló un método de detección de picos múltiples dado un conjunto para extraer características de dicha señal.

Probaron dicha funcionalidad usando un conjunto de datos público desarrollado por la Universidad de Siegen llamado WESAD (Schmidt et al., 2018). Se probaron varios clasificadores, entre ellos *9-Nearest Neighbor*, *Decision Tree*, y *Support Vector Machine*, pero el mejor rendimiento se obtuvo con un clasificador del tipo *Linear Discriminant Analysis* (LDA), alcanzando una tasa de acierto del 95.07 %.

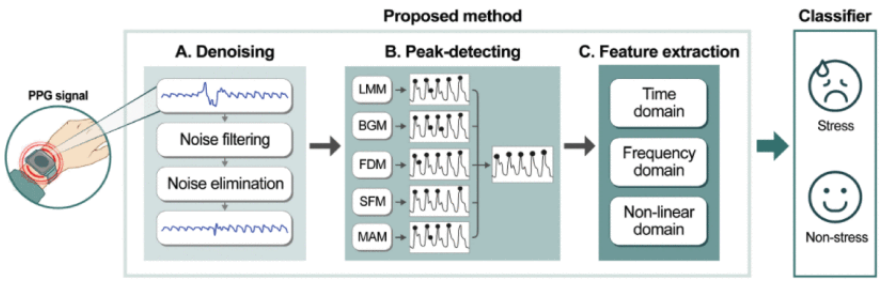


Figura 2.6: El proceso general del método que proponen en el proyecto desarrollado por el IITP (Heo et al., 2021a)

Por otro lado, un proyecto publicado en la Revista de Ingeniería Eléctrica y Ciencias de la Computación de Indonesia estudia cómo desarrollar un método nuevo para reconocer estrés mediante una señal de fotopletismografía (PPG). (Halim et al., 2017)

Justifican el uso de este sensor ya que es de bajo consumo, barato y fácil de manejar. Para abordar el desarrollo, se comienza con el estudio de la señal PPG, indicando que se obtiene mediante la pulsioximetría. La absorción de la luz por parte de la pulsioximetría permite medir la saturación de oxígeno en la sangre (SpO2) y la frecuencia cardíaca

(FC).

Un pulsioxímetro identifica una señal pulsada a partir de la señal PPG, por lo que, cualquier movimiento durante una prueba puede causar ruidos en la medición de la señal. La señal PPG cuenta con cuatro puntos diferenciados, siendo estos los puntos diastólicos, los puntos sistólicos, la muesca dicrótica y la onda dicrótica.

A partir del análisis de esta señal se experimenta con 5 sujetos para poder hacer una extracción de características y una clasificación para modelar el nivel de estrés mediante el análisis de la amplitud de la señal obtenida. Para obtener resultados en este estudio, se les pidió a los sujetos que se relajasen durante 1 minuto. Tras esto, se les comienza a medir y se les pide que realicen el Test de Stroop (Halim et al., 2017), que muestra el tiempo de respuesta de una tarea y se utiliza a menudo para ilustrar la naturaleza del procesamiento espontáneo frente al control visual consciente.

También se puede utilizar para inducir estrés al encuestado. Tras representar las señales obtenidas durante el periodo de inactividad y el periodo de estrés se puede apreciar que la amplitud de la señal no es la misma. Se concluye que el pico sistólico de la señal de estrés es significativamente mayor comparado con el estado normal. Este resultado se obtuvo al aplicar el método *Orchestrating Multiple Denoising and Peak-Detecting Methods* (OMDP), que incluye un filtrado adaptativo en frecuencia y tiempo para eliminar ruido de las señales PPG, junto con un enfoque de detección de picos basado en un conjunto de cinco métodos:

- **Local Maxima Method (LMM)**: identifica máximos locales en la señal y elimina aquellos con amplitudes menores al promedio global.
- **Block Generation Method (BGM)**: divide la señal en bloques basados en el promedio global y selecciona el punto con la mayor amplitud en cada bloque como pico.
- **First Derivative Adaptive Method (FDA)**: utiliza la primera derivada para identificar cambios en la pendiente, seleccionando picos según un umbral adaptativo en bloques de 2 segundos.
- **Slope Sum Function Method (SFA)**: aplica funciones de suma de pendientes para filtrar puntos no ascendentes y detecta picos adaptativamente según el valor mediano de las últimas cinco detecciones.
- **Moving Average Dynamic Method (MAD)**: utiliza promedios móviles con un umbral dinámico para identificar picos de mayor amplitud dentro de bloques predefinidos.

Los análisis realizados indicaron que, bajo condiciones de estrés inducido mediante el Test de Stroop, la amplitud media de los picos sistólicos aumentó en un rango del 15 % al 25 %, en comparación con el estado basal. Este incremento refleja la activación

cardiovascular característica del estrés, validada mediante un análisis estadístico de la señal basado en el conjunto de datos WESAD (Heo et al., 2021a).

2.8.2. Predicción del estrés usando conjuntos SWELL y WESAD

En paralelo, se describe un procesamiento realizado por un investigador de la Universidad de *New Brunswick*, el cual usa el conjunto de datos SWELL (Koldijk et al., 2018) para hacer una predicción del nivel de estrés (PJ, 2021). Este conjunto contiene datos recopilados, a partir de 25 sujetos que participaron en un experimento de 3 horas de duración, de sensores que miden la Variación de Frecuencia Cardíaca, la actividad electrodérmica de la piel junto con otros sensores como un *Kinect* que medía las posturas corporales, una cámara web que medía las expresiones faciales y programas de análisis de interacción con el ordenador. Para hacerlo, obtiene del conjunto de datos las condiciones, que resultan ser tres *sin estrés*, *interrupción* y *momento de presión*. Se observa que el conjunto de datos está desbalanceado, habiendo más muestras de *sin estrés* que el resto. Para finalizar, se define un modelo mediante *Random Forest* y lo entrena con el conjunto de entrenamiento de SWELL. Para comprobar el rendimiento de este modelo, se toma el conjunto de test, dado también por el conjunto SWELL, y se obtiene un subconjunto formado por 20 mediciones, para las cuales el modelo entrenado acierta la totalidad.

Por último, cabe mencionar la aportación a esta investigación sobre la detección de estrés que se hizo por parte de un grupo de investigadores de la Universidad de Memphis a través de un artículo (Dahal et al., 2023) donde se explora cómo combinar los datos de los conjuntos SWELL y WESAD. Para ello se sirven de la variabilidad de la frecuencia cardíaca (HRV), la actividad electrodermal (EDA) y datos de movimiento corporal, para mejorar la precisión en la clasificación de niveles de estrés. Este trabajo destaca por remarcar la importancia de utilizar conjuntos de datos multimodales y métodos de aprendizaje híbridos, los cuales permiten generalizar el modelo a diversas poblaciones y contextos.

2.8.3. Predicción del estrés usando otros conjuntos

Además de este, se han publicado otros artículos que buscan la mejora en la predicción del estrés a través de la combinación de diversos conjuntos de datos que existen en la actualidad. (Vos et al., 2023)

Por último, un enfoque destacado fue presentado por un equipo de investigadores de la Universidad de James Cook en Australia, quienes propusieron un modelo de aprendizaje automático basado en un conjunto de datos sintetizado para mejorar la generalización en la predicción del estrés usando dispositivos portátiles (Vos et al., 2023). El estudio abordó una de las principales limitaciones de las investigaciones previas: el uso de conjuntos de datos pequeños y específicos de un único protocolo experimental, que dificultan la capacidad de los modelos para generalizar en datos nuevos y no vistos. Para llevarlo a cabo, se recopilaron diferentes conjuntos de datos relacionados con el estrés, los cuales se

muestran en la tabla 2.7. Además, se estudiaron diferentes artículos sobre aplicaciones de Machine Learning para predecir estrés que se tienen en la actualidad, los cuales se exponen en la tabla 2.8.

Conjunto de datos	Año	Sujetos	Duración	Biomarcadores
SWELL	2014	25	138 min	EDA, HRV, ECG
Neurological Status (NEURO)	2017	20	31 min	ACC, EDA, TEMP, HR, SPO2
WESAD	2018	15	120 min	ACC, EDA, BVP, IBI, HR, TEMP, ECG, EMG, RESP
AffectiveROAD	2018	10	118 min	EDA, HR, TEMP
Toadstool	2020	10	50 min	ACC, EDA, BVP, IBI, HR, TEMP
UBFC-Phys	2021	56	20 min	EDA, BVP
Wearable Exam Stress Dataset	2022	10	180 min	EDA, HR, BVP, TEMP, IBI, ACC
Nurses Stress Dataset	2022	15	Variable	EDA, HR, ST, BVP, ACC, IBI

Cuadro 2.7: Conjuntos de datos relacionados con el estrés con diversos biomarcadores.

Utilizando este conjunto de datos sintetizado, desarrollaron un ensemble-model[6] que combina el clasificador *Gradient Boosting* y Redes Neuronales. Dicho modelo alcanzó una precisión promedio del 85 % al ser evaluado con validación cruzada de dejar-un-sujeto-fuera (LOSO), demostrando ser superior a modelos entrenados en conjuntos de datos individuales. Además, al aplicar este modelo a datos no vistos del conjunto WESAD, se logró mantener un rendimiento alto, lo cual prueba que posee una amplia capacidad de generalización.

Para abordar esta limitación, los autores combinaron datos de seis conjuntos públicos, entre los que se encuentran el *WESAD* y el *SWELL* en un único conjunto más grande, denominado *StressData*, que incluye un total de 99 sujetos. Este conjunto se generó aplicando técnicas de ingeniería de características, como resúmenes estadísticos en ventanas deslizantes de 25 segundos, para capturar la variabilidad fisiológica. Adicionalmente, mediante muestreo aleatorio de segmentos de estrés y no estrés, se construyó *SynthesizedStressData*, un conjunto ampliado que simula datos de 200 sujetos, permitiendo capturar mejor la variabilidad fisiológica necesaria para obtener modelos con un mayor rendimiento.

Para crear el conjunto de *SynthesizedStressData* se realiza, en primer lugar, un proceso de fusión entre conjuntos de datos públicos de estrés, *SWELL*, *NEURO*, *WESAD* y *UBFC-Phys* para formar un conjunto inicial, *StressData*, resultando en un conjunto con 99 sujetos y proporcionando una base más diversa de registros fisiológicos. Para cumplir con la diversidad fisiológica en cada uno de los protocolos de medición, se aplicaron ventanas deslizantes de 25 segundos sobre los biomarcadores, como el EDA y el HR. Para cada ventana, se extraen características estadísticas, como la media y la desviación estándar entre otras. Para simular el conjunto de 200 sujetos se llevó a cabo un proceso de muestreo aleatorio basado en los segmentos etiquetados como "estrés" y "no estrés" en el conjunto *StressData*, tomando combinaciones aleatorias de estos segmen-

[6]Un *ensemble model* es una técnica de aprendizaje automático que combina múltiples modelos individuales para mejorar la precisión y robustez de las predicciones. En este caso, se utilizó un enfoque híbrido que combina un clasificador basado en *Gradient Boosting* (XGBoost) y Redes Neuronales. Este enfoque permite reducir el sesgo y la varianza, mejorando la generalización.

Paper	Año	Modelo	Conjunto de Datos	Precisión	Sujetos	Características	Validación Cruzada	Ventana
Stress Detection in Working People (Sriramprakash et al., 2017)	2017	SVM	SWELL	92.75 %	25	17	10-Fold	60s
Feature selection for stress level classification into a physiological signals set (Jiménez-Limas et al., 2018)	2018	Regresión Lineal	NEURO	81.38 %	20	7	División 80/20	5m
Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection (Schmidt et al., 2018)	2018	Random Forest, LDA, AdaBoost	WESAD	93.00 %	15	82	LOSO	0.25s, 5s, 60s
The Effect of Person-Specific Biometrics in Improving Generic Stress Predictive Models (Nkurikiyeyezu et al., 2019)	2019	Random Forest, Extra-Trees	WESAD, SWELL	93.90 %	15, 25	94	10-Fold	5min, 10min
Using Deep Learning for Assessment of Workers' Stress and Overload (Eskandar & Razavi, 2020)	2020	Red Neuronal	NEURO	85 %	20	4	División 80/20	20s
Comparison of Regression and Classification Models for User-Independent and Personal Stress Detection (Siirtola & Röning, 2020)	2020	Conjunto de árboles combinados (Bagged Tree Ensemble)	AffectiveROAD	82.30 %	9	119	LOSO	60s, 0.5s de solapamiento
Stress Detection from Multimodal Wearable Sensor Data (Indikawati & Winiarti, 2020)	2020	Random Forest	WESAD	92.00 %	15	4	División 60/40	0.25s
Stress detection using deep neural networks (R. Li & Liu, 2020)	2020	Red Neuronal	WESAD	99.80 %	15	8	División 70/30	5s
A Sensitivity Analysis of Biophysiological Responses of Stress for Wearable Sensors in Connected Health (Iqbal et al., 2021)	2021	Regresión Logística	WESAD	85.71 %	14	5	14-Fold	60s
Advancing Stress Detection Methodology with Deep Learning Techniques (Liapis et al., 2021)	2021	SVM	WESAD	93.20 %	15	36	5-Fold	-
Analysing the Performance of Stress Detection Models on Consumer-Grade Wearable Devices (Ninh et al., 2022)	2021	SVM	WESAD, AffectiveROAD	87.5 %, 81.13 %	15, 9	1	60s, 30s	-
Semi-Supervised Generative Adversarial Network for Stress Detection Using Partially Labeled Physiological Data (Khan & Sarkar, 2022)	2022	Red Neuronal	SWELL	90.00 %	25	30	LOSO	60s

Cuadro 2.8: Resumen de artículos sobre detección de estrés con diferentes modelos, conjuntos de datos y metodologías.

tos, manteniendo la coherencia de los patrones fisiológicos originales. De esta forma, se incrementa la variedad de patrones sin perder características de una medición de datos en condiciones reales de experimentación y generan el conjunto *SynthesizedStressData*.

2.9. Aplicaciones similares

A día de hoy, existen múltiples dispositivos *wearables* en el mercado que cuentan con aplicaciones preinstaladas para medir el nivel de estrés. Estos dispositivos han evolucionado significativamente en los últimos años, incorporando sensores más precisos y nuevas métricas. En la Figura 2.7 se muestran algunos ejemplos. Además, cabe resaltar que, mientras que anteriormente estas características se encontraban únicamente en modelos tope de gama, actualmente se ha democratizado, extendiéndose a todo tipo de precios y diferentes marcas.

(a) Samsung Galaxy Watch

(b) Xiaomi Mi Band

(c) Fitbit

Figura 2.7: Ejemplos de dispositivos y aplicaciones para la medición de estrés.

A modo de ejemplo, los últimos modelos de dispositivos *wearable* de Samsung, como relojes o incluso anillos, ofrecen mediciones del estrés a través de su aplicación *Samsung*

Health (Samsung, s.f.). Esta medición se basa, entre otros factores, en la Variabilidad de la Frecuencia Cardíaca (VFC) y en la impedancia bioeléctrica y para llevarla a cabo se necesita que el usuario sitúe sus dedos sobre los botones del reloj para un análisis más preciso.

Entre los pioneros en medición de estrés, encontramos las pulseras Fitbit, las cuales destacan por incorporar tecnología más avanzada utilizando sensores de actividad electrodérmica (EDA) para detectar cambios en la conductancia de la piel. Estos cambios, asociados al sistema nervioso simpático, permiten evaluar la respuesta fisiológica al estrés de una forma más directa (Google, 2024b).

Por otro lado, encontramos dispositivos más asequibles, como la Xiaomi Mi Band, que también ofrece mediciones basadas en la VFC comparada con modelos de estrés preentrenados. (Xiaomi, 2024).

Por último, destacar uno de los dispositivos wearable más extendidos, el Apple Watch. Este ofrece monitorización de parámetros fisiológicos, incluyendo la variabilidad de la frecuencia cardíaca (VFC). La VFC se considera un indicador no invasivo para el estudio de la función del Sistema Nervioso Autónomo (SNA), siendo sensible a cambios tanto fisiológicos como psicológicos (Hernando et al., 2018; Jerath et al., 2023). Por este motivo, su empleo se ha asociado a la evaluación y gestión del estrés, ya que diferentes estados emocionales pueden manifestarse en alteraciones de la VFC.

Se han llevado a cabo investigaciones que prueban la fiabilidad de las mediciones de la VFC del Apple Watch comparándolas con dispositivos validados clínicamente, como por ejemplo el *Polar H7*, que registra intervalos RR de alta precisión (Hernando et al., 2018). Estos estudios han demostrado que las series RR obtenidas a través del Apple Watch presentan alta concordancia, baja dispersión y un elevado grado de similitud con las mediciones de referencia. Sin embargo, se han observado ciertas limitaciones, como la presencia de intervalos perdidos, los cuales pueden afectar especialmente al análisis en el dominio de la frecuencia de la VFC (Hernando et al., 2018). Sin embargo, estas limitaciones no impiden que las mediciones que se pueden extraer de la VFC, como SDNN o RMSSD, se mantengan estables y esto nos permita estudiar el estado de estrés del individuo.

Por otro lado, el Apple Watch, al igual que otros dispositivos que hemos mencionado, incorpora funciones orientadas a la gestión del estrés. Estas incluyen algoritmos y aplicaciones que permiten al usuario monitorizar y analizar de forma continua su variabilidad de la frecuencia cardíaca (VFC) a lo largo del tiempo (Jerath et al., 2023), facilitando la detección temprana de cambios en la respuesta al estrés. Esto permite al usuario adoptar medidas proactivas, como practicar técnicas de relajación, como respiraciones guiadas, o registrar estados emocionales en momentos específicos para trabajar sobre ellos posteriormente. Algunas de estas funcionalidades se ilustran en las figuras 2.8 y 2.9.

Por ello, podemos concluir que a pesar de las ventajas que ofrece el Apple Watch en cuanto a facilidad de uso, diseño intuitivo e integración con un ecosistema ampliamente

Figura 2.8: Funciones de control de estrés en la suite de Mindfulness del Apple Watch

Figura 2.9: Registro de estados de ánimo en la suite de Mindfulness del Apple Watch

aceptado, es importante considerar las limitaciones que pueden surgir al emplearlo con fines de investigación. La falta de acceso directo a datos crudos y la dependencia de algoritmos propietarios dificultan la realización de estudios fisiológicos de alta resolución, que es esencial para estudios que requieran validez clínica. No obstante, su éxito comercial y la amplia aceptación por parte de los usuarios se combina con la conveniencia de contar con un dispositivo capaz de proporcionar, de forma simple, un panorama global de la salud cotidiana. De esta forma, aunque su precisión no alcance a la de instrumentos médicos especializados, la combinación de funcionalidades prácticas, integración tecnológica y su amplia base de usuarios convierten al Apple Watch en una opción muy extendida para el monitoreo diario del estrés.

Capítulo 3

Propuesta

3.1. Metodología de trabajo

Para comenzar, comentaremos cómo vamos a llevar a cabo el desarrollo del presente trabajo fin de máster, explicando, primeramente, el marco de trabajo a seguir, junto a la descripción de recursos que serán necesarios, y terminaremos exponiendo los métodos de investigación que se van a aplicar.

3.1.1. Marco de trabajo

Para comenzar con nuestra propuesta, estableceremos antes la forma en la que se trabajará sobre este problema. Primeramente, se va a seguir una metodología de desarrollo ágil, dividiendo el trabajo en iteraciones de un mes de duración. Al final de cada iteración, se agendará una reunión de revisión con las tutoras de este trabajo fin de máster a fin de mostrar los avances hasta el momento y obtener una retroalimentación con la que poder pulir aspectos durante el desarrollo del mismo.

3.1.2. Métodos de investigación

Para este proyecto, se escogerá una combinación de métodos de investigación, tales como la revisión bibliográfica, entrevista a las tutoras y sus compañeros de grupo de investigación, y la observación de resultados recogidos mediante la experimentación. Llevaremos a cabo una investigación a partir de una combinación cuantitativa, a través de estudios empíricos recogiendo datos y procesándolos.

3.1.3. Descripción del alcance del sistema

El presente Trabajo Fin de Máster consiste en la ampliación de un entorno multisistema distribuido desarrollado en el trabajo fin de grado, permitiendo mostrar más información sobre sensores, y mejorando el modelo de predicción de estrés, apoyándonos en nuevos conjuntos de datos y aplicando técnicas de Machine Learning para la detección del estrés. Todo esto será posible mediante una arquitectura que posibilite la comunicación entre los sistemas que lo componen, que son:

- Una aplicación web que interpreta los datos recogidos por los sensores en las actividades y que permite la gestión de usuarios mayores por parte de terapeutas.
- Funciones como servicio para proveer la funcionalidad de la detección de estrés mediante modelos de aprendizaje automático.
- Una aplicación de reloj inteligente para la medición de estrés y sus causas en la realización de actividades de la vida diaria mediante etiquetado.
- Una aplicación para móvil que permita intercambiar el usuario que usa el reloj, a modo de optimizar los recursos.

La aplicación web para el terapeuta debe incluir las siguientes características:

- Identificación del terapeuta para que los datos recabados de los usuarios mayores sean de acceso restringido.
- Un panel de control de acceso restringido donde se encuentre la información asociada a los usuarios mayores.
- En el panel de control deben existir las siguientes secciones:
  - Una sección para la gestión de los usuarios mayores, darlos de alta en el sistema, poder visualizarlos, editarlos, eliminarlos y asignarles actividades.
  - Una sección para la gestión de actividades del sistema, poder crearlas, editarlas y eliminarlas.
  - Una sección para la gestión de etiquetas para actividades, de modo que podamos crear nuevas, editar las ya existentes y eliminarlas si es necesario.
- En el panel de control debe existir además, una sección donde se puedan visualizar, para cada usuario, los niveles de estrés procesados y las etiquetas de una actividad.
- En el panel de control, se deben crear nuevos tipos de actividad, pudiendo asignarles un nombre de actividad y el nombre con el que aparecerá en el reloj. Debe llevar asociado un icono descriptivo de la actividad.

- En el panel de control debemos gestionar un banco de etiquetas, las cuales conformarán parte de una actividad. Cada etiqueta debe tener un nombre, un nombre con el que aparecerá en el reloj, un tipo (estado o contexto) y el icono que la representa.
- Cada actividad debe ser configurable, pudiendo añadirle etiquetas del repositorio en un orden determinado.
- La visualización de actividades para un usuario debe estar categorizada por tipo de actividad.
- Para la vista del procesamiento de una actividad se debe mostrar el nivel de estrés mediante un gráfico y una tabla con las etiquetas asociadas a las etiquetas que conforman las actividades, estando divididas en contextos, estados, emociones y sentimientos. Además, las etiquetas deberán aparecer sobre el gráfico cuando sea oportuno.
- Junto al gráfico y la tabla, se debe mostrar un mapa que permita interpretar los recogidos por el sensor de GPS. Este debe mostrar el camino que ha seguido el usuario durante la realización de la actividad.
- Los datos en crudo de medición de sensores deben poder exportarse por si se quiere hacer un estudio posterior.

En cuanto a las funciones como servicio, se debe proveer de la siguiente funcionalidad:

- Se debe publicar una función como servicio (FaaS[1]) para la detección del nivel de estrés, dado un conjunto de mediciones por los sensores.
- Publicar una FaaS que permita elaborar un perfil de usuario para la detección de estrés crónico, de acuerdo con el DASS-21.

En cuanto a la aplicación móvil, se debe cumplir con las siguientes funcionalidades:

- Debe contener una pantalla de identificación mediante la cual se cambiará el usuario del reloj inteligente.
- Se quiere notificar al usuario cual es el estado de la conexión con su reloj inteligente.
- La aplicación no necesita estar activa en pantalla para que se establezca una comunicación con el reloj (puede estar en background).

[1]Una FaaS (Function as a Service) es un modelo de computación en la nube que permite ejecutar fragmentos de código en respuesta a eventos, sin necesidad de gestionar la infraestructura subyacente. En este modelo, el proveedor de servicios en la nube se encarga del aprovisionamiento, escalado y mantenimiento de los servidores, mientras que el desarrollador solo se enfoca en el código de la función. Se utiliza típicamente para tareas específicas, como procesamiento de datos, automatización de procesos o respuestas en tiempo real.

- La aplicación debe guardar la sesión hasta que no se cierre.
- Se debe poder cerrar sesión para cambiar de usuario.
- Debe contar con un diseño accesible.

Para terminar, la aplicación del reloj inteligente debe contar con las siguientes características:

- Debe comunicarse con el móvil para identificarse en el sistema con el usuario que esté identificado en el móvil.
- Cuando se quiere comenzar una actividad, se muestra un listado con los tipos de actividades asignados para dicho usuario.
- Se debe indicar mediante una pantalla adicional qué se va a mostrar y qué se debe hacer a continuación.
- En el menú principal durante el transcurso de la actividad se debe poder detener la medición de la actividad en transcurso y registrar etiquetas.
- El registro de etiquetas debe realizarse por categorías, siendo las categorías, estados de la actividad, como las situaciones por las que el usuario está pasando; contextos, compuesto de etiquetas en las que se describen condiciones externas que rodean la actividad; emociones, como la felicidad, la tristeza, el miedo, la ira, la sorpresa y el asco; y por último sentimientos, como nervioso o tranquilo.
- La aplicación debe seguir activa en segundo plano, evitando la detención de la medición de los sensores y dándole la posibilidad al usuario de poder volver a la aplicación para registrar etiquetas, a fin de ahorrar batería.

3.1.4. Recursos

Para el desarrollo del proyecto, se facilita un dispositivo *smartwatch* modelo TicWatch Pro de la marca Mobvoi (Figura 3.1), el cual posee WearOS 2.0 y un sensor PPG. Además, se ha utilizado una instancia virtual de Android Wear para facilitar el desarrollo de la interfaz, mientras que esta parte fuese independiente del uso de sensores.

Además de esto, se necesitará un *smartphone* que cuente con sistema operativo Android para enlazarlo al *smartwatch*, junto con un servicio cloud de máquina virtual para alojar la aplicación web del terapeuta, que almacene las bases de datos y provea las FaaS. Esto nos permitirá exponer una API que permita conectarnos al sistema desde fuera de la red local donde desarrollaremos. El desarrollo se realizará en un ordenador personal.

Figura 3.1: Mobvoi TicWatch Pro

3.1.5. Diagrama de arquitectura

La solución que se propone se organiza de acuerdo al siguiente diagrama de arquitectura mostrado en la figura 3.2. A continuación, se detalla y se justifica el por qué de la decisión de llevar a a cabo una arquitectura como la que se muestra.

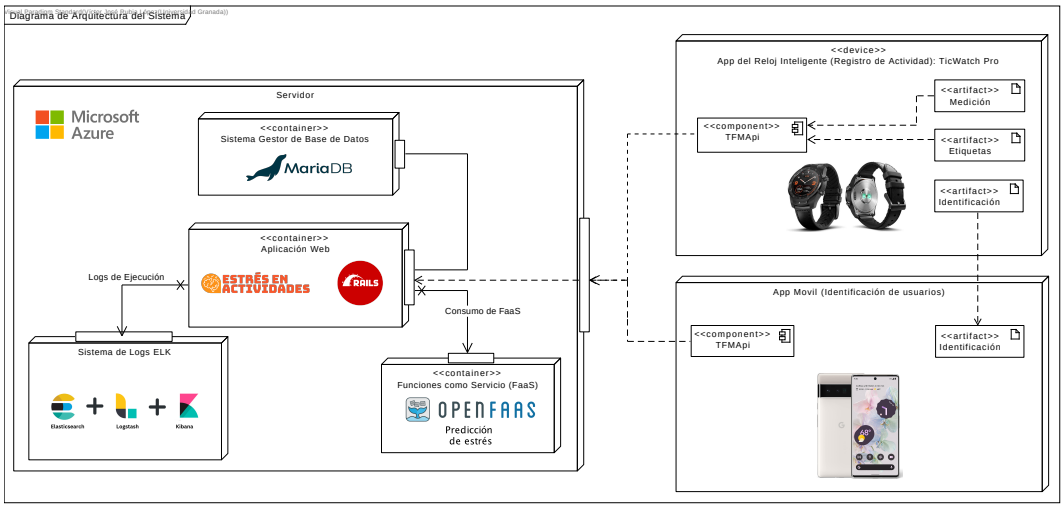


Figura 3.2: Diagrama de Arquitectura del Sistema

Se ha diseñado una arquitectura modular siguiendo un enfoque basado en sistemas distribuidos y apoyándonos en las tecnologías que se han revisado en la sección 2.7 del estado del arte, para poder gestionar de forma óptima la información de los diferentes sistemas. Este lo conforman tres sistemas, que se comunican entre sí para lograr funcionar: un servidor, una aplicación móvil y una aplicación de reloj inteligente. A su vez, el servidor está compuesto de 4 módulos, entre los que encontramos la aplicación web y las Funciones como Servicio (FaaS). Estos módulos se despliegan en el servidor como contenedores, que son necesarios para el correcto funcionamiento de la aplicación web.

En primer lugar, se desarrollará el módulo del servidor, que está dividido en cuatro sistemas:

- **Base de Datos**. Para desplegar una base de datos sobre la que poder almacenar la información, nos valdremos de la imagen de contenedor de MariaDB publicado en su repositorio oficial de DockerHub (MariaDB Foundation, 2024). Desplegaremos durante el desarrollo, esta imagen sobre el ordenador personal, mediante Docker. De esta forma, para desplegarlo en producción podremos usar dicha imagen para levantar un contenedor sobre una máquina virtual en un entorno Cloud tipo Azure. La base de datos no queda expuesta a la red, ya que se utiliza una red interna virtual entre diferentes contenedores que componen el sistema para la comunicación.
- **Aplicación Web**. Se ha desarrollado un contenedor que ejecuta Ruby On Rails, un framework de Ruby usado para la creación del sistema web de modo que sea fácilmente escalable en funcionalidad. Este se conecta a la base de datos que se encuentra alojada en el sistema mencionado en el punto anterior. Tiene la responsabilidad de almacenar la información de los sensores de los dispositivos, así como los diferentes usuarios que van a formar parte del sistema y las actividades que pueden realizar, junto con las etiquetas que las representan. Además, este sistema es el encargado de ejecutar el procedimiento de predicción de obtención del nivel de estrés y guardarlo. Al igual que anterior, uno de los propósitos de contenedorizar este apartado es el posterior despliegue en un servicio Cloud.
- **Sistema de Logs ELK**. Se implementan tres contenedores que forman parte de un Stack conocido como ELK (ElasticSearch, Logstash y Kibana). En conjunto son herramientas muy potentes para la gestion de logs que produzcan los diferentes sistemas que tenemos integrados en nuestra arquitectura, de forma que a posteriori, se puedan consultar resultados o errores de ejecución, entre otros.
- **OpenFaaS**. Es una plataforma que permite implementar funciones como contenedores y ejecutarlas bajo demanda de forma eficiente. Su diseño facilita la creación, despliegue y gestión de aplicaciones *serverless* de manera escalable. Esto estará instalado sobre el sistema que vamos a desplegar los contenedores. Ejecutaremos contenedores que tienen un propósito al estilo de una función para obtener un resultado. Esto nos proporciona una forma distribuida y escalable de poder calcular el nivel de estrés para las diferentes actividades dejando la responsabilidad para cada ejecución de la función.

En cuanto al módulo del dispositivo móvil, App Móvil en el diagrama, se describen a continuación las características:

- **Dispositivo móvil**. Se trata de un teléfono inteligente modelo Pixel 6 de la marca Google el cual posee 8 GB de RAM y 128 GB de almacenamiento. Como versión de Android tiene la 15, que es la última disponible en el mercado. Este dispositivo debe estar enlazado al reloj inteligente a utilizar y contendrá una aplicación desarrollada para poder identificar a distintos usuarios en el reloj inteligente a modo de que pueda ser utilizado por varios, optimizándose así los recursos mediante la conexión con el servidor de la aplicación web.

Para el módulo del reloj inteligente, App del Reloj Inteligente en el diagrama, describimos las características que posee:

- **Dispositivo reloj inteligente**. Se proporciona un reloj de la marca Mobvoi modelo TicWatch Pro el cual posee 512 MB de RAM y 4 GB de almacenamiento. Este contendrá la aplicación de medición de sensores y registro de etiquetas para las actividades, al igual que un módulo de identificación de usuario para el cual se necesitará estar conectado al dispositivo móvil. Por otro lado, el reloj se conectará al servidor de la aplicación web para poder realizar las mediciones de actividades.

Cuando se haya terminado el desarrollo, nos valdremos de los servicios de Azure para poder desplegarlos en Cloud y poder probar la funcionalidad del sistema, ya que al estar pensado para manejo de datos a través de sistemas distribuidos y fuera de espacios de red local, nos debemos conectar a través de una IP o dominio público.

Para esto, nos valdremos del servicio de Azure que provee máquinas virtuales. Atendiendo a las características del servidor que se propone para alojar el módulo de la aplicación web que se exponen en la sección 1.5 de la introducción, elegiremos uno con las siguientes características:

- **Familia**: B-series.
- **vCPU**: 2 núcleos virtuales.
- **Memoria RAM**: 8 GiB.
- **Almacenamiento temporal**: 16 GiB.
- **Rendimiento de red**: Moderado.
- **Disco administrado máximo**: Hasta 4 TB (dependiendo de la configuración del disco).
- **Tipo de CPU**: Procesador basado en x64.

Esta máquina virtual es ideal para hospedar un servidor web que ejecute aplicaciones ligeras, entornos de desarrollo y pruebas, bases de datos no muy grandes y servicios con un grado de uso intermitente.

3.2. Plan de iteraciones del proyecto

3.2.1. Lista inicial del producto (Product Backlog)

La tabla 3.1 contiene las historias de usuario abordadas durante el desarrollo del proyecto, las cuales han sido ordenadas por prioridad. Adicionalmente, se ha añadido

la estimación del esfuerzo para cada historia, utilizando el método del *Planning Poker* (Gandomani et al., 2019).

ID	Historia de Usuario	Estimación	Prioridad	Entrega
HU-32	Como terapeuta quiero asignar actividades específicas a los usuarios registrados en la web para personalizar su experiencia.	2	1	2
HU-33	Como terapeuta quiero registrar nuevos tipos de actividad, definiendo su nombre, identificativo, nombre visible en el reloj, y un icono alojado en un servicio AWS S3.	5	1	1
HU-34.1	Como terapeuta quiero crear etiquetas con nombre, identificativo, nombre visible en el reloj, y un icono guardado en AWS S3.	3	1	1
HU-34.2	Como terapeuta quiero previsualizar cómo se mostrará cada etiqueta en el reloj para asegurar consistencia.	2	1	2
HU-35.1	Como terapeuta quiero seleccionar etiquetas asociadas a una actividad en el orden específico en que aparecerán.	2	1	2
HU-35.2	Como terapeuta quiero editar la configuración de etiquetas en actividades ya creadas para ajustarlas si es necesario.	1	2	2
HU-36	Como terapeuta quiero acceder a un nuevo apartado para visualizar las actividades realizadas por los usuarios y sus perfiles de probabilidad de estrés crónico generados con un modelo de machine learning.	13	2	5

ID	Historia de Usuario	Estimación	Prioridad	Entrega
HU-37	Como terapeuta quiero tener buscadores en todas las secciones (usuarios, actividades, etiquetas, registros) para ahorrar tiempo y mejorar la navegación.	3	1	2
HU-38	Como terapeuta quiero ver para cada usuario una lista con los tipos de actividad que ha registrado, incluyendo el icono y nombre de cada actividad.	2	1	3
HU-39.1	Como terapeuta quiero un gráfico interactivo que muestre etiquetas registradas al pasar el ratón por puntos específicos.	10	2	4
HU-39.2	Como terapeuta quiero mantener la tabla de etiquetado para complementar la visualización gráfica.	2	2	4
HU-40.1	Como terapeuta quiero ver la ubicación inicial y final de la actividad en un mapa.	3	2	4
HU-40.2	Como terapeuta quiero visualizar la ruta completa seguida por el usuario durante una actividad.	2	2	4
HU-41.1	Como usuario quiero que el reloj descargue automáticamente actividades asignadas desde la plataforma.	3	1	2
HU-41.2	Como usuario quiero que las actividades descargadas incluyan toda la información (iconos, etiquetas, textos).	2	1	2
HU-42	Como usuario quiero poder listar las etiquetas asociadas a una actividad durante su registro, seleccionando las necesarias para marcar eventos importantes.	2	1	3

ID	Historia de Usuario	Estimación	Prioridad	Entrega
HU-43.1	Como usuario quiero que la medición continúe en segundo plano cuando el reloj cierra la aplicación por inactividad.	5	2	3
HU-43.2	Como usuario quiero que se notifique en la pantalla del reloj que la medición sigue activa mientras está en segundo plano.	3	3	3

Cuadro 3.1: Nuevas historias de usuario del proyecto

3.2.2. Cálculo de velocidad

En metodologías ágiles, la velocidad se usa para medir la cantidad media de trabajo que un equipo puede completar en una iteración, en nuestro caso, de duración de un mes. Se usarán puntos de historia para realizar la estimación de las historias de usuario.

Para comenzar el proyecto, se seguirán recomendaciones para estimar la velocidad inicial (digitalAI, s.f.), teniendo en cuenta que no podré dedicar el 100 % del tiempo planeado. Aunque las jornadas serán de 4 horas, es probable que parte de ese tiempo se pierda en descansos, interrupciones, compromisos externos o tareas como reuniones y documentación. Por eso, se estima que el tiempo efectivo será aproximadamente el 70 % del total.

El tiempo disponible por iteración se calcula como sigue:

$$\text{Horas/iteración} = 4\,\text{horas/día} \times 10\,\text{días} = 40\,\text{horas} \times 0.7\ \%\ \text{de dedicación} = 28\,\text{horas}.$$

Para estimar la velocidad inicial en puntos de historia:

$$\text{Velocidad inicial estimada} = 28\,\text{PH/iteración}.$$

Se utilizarán estas métricas iniciales y se ajustarán después de completar las primeras iteraciones para refinar la planificación y asignación de tareas en el backlog.

	Persona	Horas ajustadas
	Víctor José Rubia López	28
Total		28

Cuadro 3.2: Velocidad del desarrollador del proyecto

3.2.3. Descripción de las entregas

Dado que el esfuerzo necesario suma 68 puntos de historia y la velocidad del desarrollador es de 30 puntos de historia por iteración, el proyecto se estructurará en cinco entregas, cada una correspondiente a una iteración de 15 días naturales (10 días laborales). El desarrollo del proyecto comenzó el 15 de septiembre de 2024. En la tabla 3.3, se detalla de qué trata cada entrega y cuál es su fecha.

Entregas	Objetivos	Fecha de entrega
1	Tener una aplicación web básica que permita gestionar usuarios, registrar actividades y crear etiquetas.	30 de septiembre de 2024
2	Incorporar funcionalidades avanzadas de configuración de etiquetas, actividades, y sincronización de datos con el reloj inteligente.	15 de octubre de 2024
3	Añadir visualización de actividades en la plataforma, junto con funcionalidades básicas del reloj para listar etiquetas y enviar mediciones de sensores.	30 de octubre de 2024
4	Completar funcionalidades avanzadas como gráficos interactivos, rutas en mapas, y etiquetado en tiempo real desde el reloj.	14 de noviembre de 2024
5	Implementar y probar las funciones OpenFaaS: predicción del nivel de estrés en ventanas de tiempo y probabilidad de sufrir estrés crónico.	1 de diciembre de 2024

Cuadro 3.3: Descripción de las entregas del proyecto.

Las secciones que se describen a continuación no siguen una descripción del desarrollo iterativo del proyecto, ya que esto implicaría una extensión considerable del documento. En lugar de esto, se opta por estructurar las siguientes secciones detallando cada uno de los sistemas que conforman el proyecto: la aplicación web, la aplicación móvil, el reloj inteligente y el proceso realizado para la predicción del nivel de estrés. Cada subsección se centra en detallar los aspectos relevantes de diseño, desarrollo e implementación de cada uno de estos componentes.

3.3. Aplicación Web

3.3.1. Introducción

Para comenzar con la sección, detallaremos el desarrollo realizado sobre la aplicación web, núcleo de nuestra arquitectura multisistema propuesta. Este desarrollo parte de

una base ya existente, concretamente de mi Trabajo Fin de Grado previo, sobre el cual se plantean mejoras y ampliaciones significativas para cumplir con los requisitos que se marcan en el proyecto.

La misión principal de la aplicación web es gestionar la información relativa a los usuarios, sus actividades y las mediciones fisiológicas. Además, en esta plataforma residen las APIs que permiten al reloj inteligente (detallado en la Sección 3.6) y al móvil (detallado en la Sección 3.4) realizar las operaciones de lectura y escritura de datos.

3.3.2. Situación previa y motivaciones de la actualización

Antes de abordar las tareas específicas, actualizaremos el entorno de desarrollo para garantizar la compatibilidad y seguridad del sistema. La versión anterior de la plataforma se desarrolló empleando la versión `3.1.0` de Ruby junto con Rails en su versión `7.0.1`. Las necesidades actuales hacen imprescindible su actualización y la introducción de nuevas funcionalidades como:

- Mayor seguridad de los datos aplicando encriptación de contraseñas.
- Soporte para bases de datos distribuidas, de modo que podamos deslocalizar la información para tener un mayor grado de seguridad.
- Posibilidad de crear, asociar y gestionar dinámicamente etiquetas y tipos de actividad.
- Recepción de nuevos datos de sensores del reloj.
- Integración con funciones *serverless* (FaaS) para el análisis del estrés.

Para garantizar la compatibilidad con las últimas versiones del ecosistema y librerías disponibles, se procede a actualizar Ruby a la versión `3.3` y Rails a la versión `7.2.2`, que son las últimas disponibles en el momento del desarrollo de la aplicación. Durante este proceso, se revisaron las dependencias y se resolvieron los conflictos de dependencias entre las diferentes gemas.

Tras la actualización, se volvieron a lanzar las pruebas unitarias y de integración, definidas en el Trabajo Fin de Grado, mediante el script creado para ello, para confirmar la funcionalidad del sistema, al igual que se exploraron las diferentes funcionalidades de forma manual. Los resultados

3.3.3. Tareas de desarrollo realizadas

Para acotar la extensión de la memoria, no se detallarán aquí todas las tareas y tarjetas generadas para cada una de las Historias de Usuario; sin embargo, detallaremos aquellas que han tenido más relevancia y daremos una breve introducción a lo que ya se tenía desarrollado.

Tareas de empaquetado para el contenedor

Partimos de una base en la que se tiene un contenedor con la implementación de la aplicación web. Esto implica que el frontend y el backend estén en un mismo contenedor, aprovechándonos de la facilidad que nos ofrece el framework Ruby On Rails de proveer la lógica siguiendo la filosofía del *Convention Over Configuration*[2] para el desarrollo de vistas, modelos de datos y endpoints de API.

Nos seguimos apoyando en la tecnología Docker para ejecutar el contenedor en nuestra máquina de desarrollo, con la intención de posteriormente desplegarlo en cualquier arquitectura, como por ejemplo en alguna en Cloud, de modo que así podamos desacoplar la ejecución del sistema operativo huésped. En el código 3.1 se muestra la creación de una imagen Docker que contenga nuestra aplicación web.

```
1  FROM ruby:3.3
2  RUN apt-get update -qq && apt-get install -y build-essential default-mysql-client
       ↪ default-libmysqlclient-dev
3  RUN apt remove cmdtest -y
4  RUN curl -sL https://deb.nodesource.com/setup_20.x | bash
5  RUN apt-get install nodejs
6  RUN npm install -g npm@latest
7  RUN npm install -g yarn
8  WORKDIR /tfm
9  COPY Gemfile /tfg/Gemfile
10 COPY Gemfile.lock /tfg/Gemfile.lock
11 RUN bundle install
12
13 COPY entrypoint.sh /usr/bin/
14 RUN chmod +x /usr/bin/entrypoint.sh
15 ENTRYPOINT ["entrypoint.sh"]
16 EXPOSE 80
17
18 CMD [ "bundle", "exec", "puma", "config.ru"]
```

Listing 3.1: Dockerfile capaz de virtualizar un sistema que ejecute Ruby On Rails con las dependencias necesarias para la plataforma a desarrollar

En este marco, se ha eliminado la necesidad de instalar *Python*, junto a las librerías de Machine Learning que eran necesarias para calcular para cada actividad su grado de estrés en ventanas de medición, ya que se ha decidido desacoplar esta responsabilidad pasándose a desarrollar funciones serverless (FaaS). Esto nos es ventajoso, por un lado, para agilizar la respuesta del servidor, que anteriormente se quedaba bloqueado por la ejecución de la lógica de cálculo de grado de estrés y, por otro lado, para reducir el peso de la imagen que contiene nuestra aplicación web.

De igual forma, se ha hecho uso de la tecnología de Docker Compose, ya utilizada anteriormente para administrar los diferentes contenedores que conforman la aplicación web. Estos son:

[2]La filosofía "Convention Over Configuration" (CoC) consiste en que se establezcan una serie de convenciones por defecto para la estructura y configuración de la aplicación, de modo que, siguiendo dichas convenciones, reducimos la necesidad de configuraciones específicas, simplificando el desarrollo y mantenimiento del proyecto.

- **Aplicación Web**. Se han actualizado las versiones de Ruby y Ruby On Rails, junto a todas las dependencia, lo que permite optimizar la forma en la que se despliega este contenedor, atendiendo las guías de despliegue más recientes.
- **Sistema de Gestión de Base de Datos MySQL**. Se ha cambiado el Sistema Gestor de MariaDB a MySQL debido a requisitos específicos de compatibilidad y rendimiento en el entorno de producción.
- **PHPMyAdmin**. Útil durante el desarrollo para interactuar con las BBDD, ya que se trata de una herramienta que nos permite visualizar en una interfaz gráfica las diferentes bases de datos, tablas y filas que tenemos. Es de ayuda para ver rápidamente si lo que se crea en el modelo de datos se está haciendo adecuadamente.
- **Elasticsearch**. Implementado como parte de la solución para la indexación y consulta eficiente de logs de la aplicación web.
- **Logstash**. Configurado para recopilar y transformar los logs de la aplicación web antes de enviarlos a Elasticsearch.
- **Kibana**. Utilizado para la visualización de los logs de la aplicación web, permitiendo análisis y monitoreo en tiempo real.

Tareas de despliegue

Para solventar la problemática que se tenía respecto a poder acceder a nuestro backend desde internet, fuera de la red local, previamente teníamos un servidor el cual se exponía a través de un dominio a internet. Para mejorar esto, se ha decidido utilizar un servicio cloud de máquina virtual, como el que ofrece Azure. Azure tiene una suscripción especial para estudiantes, a través de un acuerdo con la Universidad de Granada que nos permite tener ciertas horas de uso de máquinas virtuales.

De esta forma, crearemos una máquina virtual con Linux seleccionando la arquitectura que mencionamos en el punto 3.1.5 donde comentamos la arquitectura del sistema. Esta máquina está prevista de una IP pública a través de la cual podemos acceder a los recursos que allí se desplieguen. Tras algunas configuraciones iniciales de apertura de puertos y de control de accesos y actividad, podemos conectarnos a ella por SSH y empezar a desplegar el entorno de producción.

Para lograrlo, instalaremos Docker y todas sus dependencias asociadas. Una vez hecho esto, podemos desplegar allí el proyecto y configurar el archivo de variables de entorno. De esta forma podríamos ejecutar el conjunto de contenedores en Cloud permitiendo establecer un puente entre los puertos que se exponen en los contenedores y los puertos que la máquina virtual expone a internet.

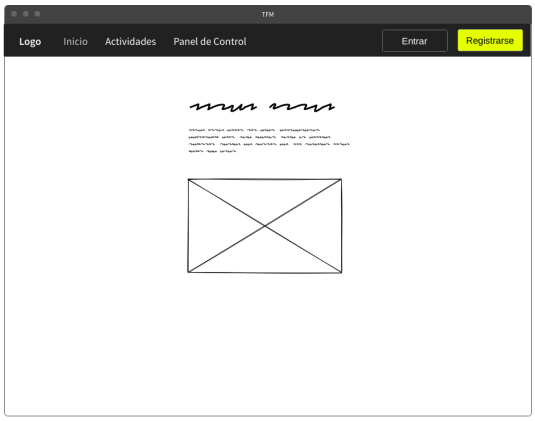
De esta forma, nos despreocuparemos de las limitaciones y complejidades asociadas a mantener y asegurar un entorno local. Algunas ventajas de desplegar el entorno en la nube son la escalabilidad, de modo que, cuando la aplicación experimente picos de carga

o requiera más potencia, podamos ampliar los recursos sin interrumpir el servicio. Otra ventaja es la alta disponibilidad, pues podemos balancear la carga, configurar réplicas y mantener operativa la aplicación incluso ante fallos en un nodo o en la propia máquina virtual. A nivel de seguridad, la nube ofrece servicios y herramientas de protección avanzadas, como cortafuegos, autenticación y cifrado, que robustecen la aplicación frente a ataques. Combinando este despliegue con flujos de CI/CD, nos permite que cada cambio en el código se pruebe y se distribuya automáticamente en el entorno productivo, acortando los ciclos de desarrollo y mejorando la confiabilidad del servicio. Los flujos de CI/CD (Integración Continua/Despliegue Continuo) son prácticas fundamentales en el desarrollo de software en la actualidad que automatizan la integración de cambios en el código, asegurando que estos sean validados mediante pruebas automatizadas antes de ser desplegados de manera progresiva en los entornos de producción y garantizando una entrega más rápida y segura de nuevas funcionalidades y correcciones.

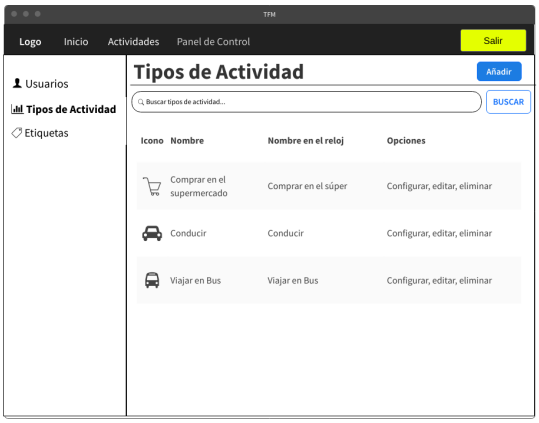
Tareas de bocetado

Para la realización de las tareas asociadas con la parte de la aplicación web nos apoyaremos en el bocetado que ya se hizo previamente. Se optó por aplicar un diseño basado en *Bootstrap 5* (Twbs, 2022), ya que está bastante estandarizado y posee muchos elementos que nos interesa usar, como las barras de navegación, los paneles de control y las ventanas emergentes (modales), y estas siguen siendo a día de hoy vigentes.

En cuanto a los iconos a utilizar en la plataforma web, seguiremos optando por el uso de los que proporciona *Feather*, ya que son simples, universales y de código abierto (Feathericons, 2022).



(a) Adición al Navbar de la nueva sección de Actividades



(b) Nueva sección para la configuración de actividades dinámicas

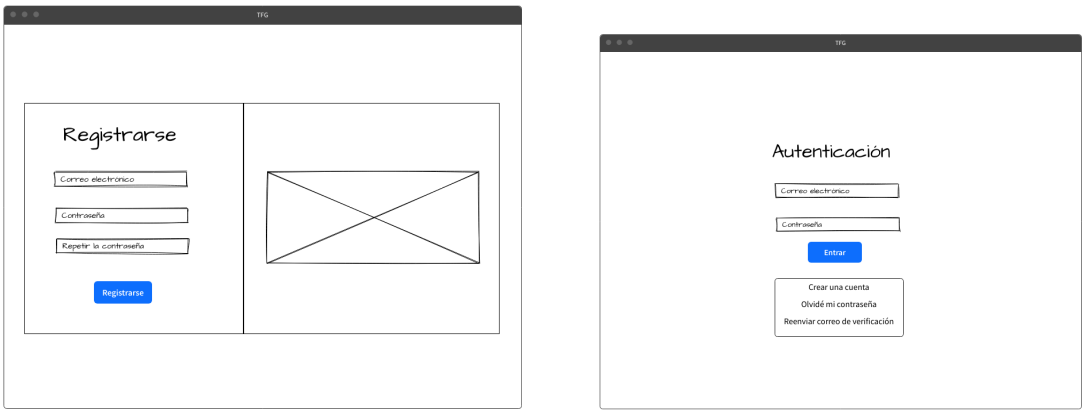
(c) Nueva visualización del detalle de actividades

Figura 3.3: Muestra de bocetado actualizado para incluir la nueva funcionalidad.

Como vemos en la página inicial del sistema web expuesto en la Figura 3.3a, se optó por un diseño compuesto por una barra de navegación en color negro que contiene, de izquierda a derecha, el logotipo del proyecto, los enlaces al inicio, al apartado de actividades realizadas por los usuarios y un enlace final al panel de control. Al final de la barra de navegación, encontramos botones tanto para identificarse como para registrarse.

Las pantallas de registro y autenticación se mantendrán tal y como se desarrollaron, pues aquí no tenemos ninguna modificación que les afecte en cuanto a diseño.

La Figura 3.4a muestra un formulario de registro con el logotipo, basado en el uso de correo electrónico para identificar al usuario y recuperar la contraseña. Por otro lado, la Figura 3.4b presenta un formulario de inicio de sesión con enlaces para crear cuenta, recuperar contraseña o reenviar verificación.



(a) Página de registro (b) Página de identificación

Figura 3.4: Comparativa entre las páginas de registro e identificación de la aplicación web.

En este punto, se vio necesario añadir a las tareas una que fuese la actualización del logotipo del proyecto para poder adoptar la visión genérica que nos aporta el poder añadir tipos de actividad de forma dinámica.

Para ello, se ha optado por usar un icono compuesto de una cabeza y unos rayos que simbolizan el estrés en personas. Como la temática del proyecto es medir el estrés en actividades de la vida diaria, el logotipo acuñará estas palabras. Se muestra en la Figura 3.5.

Figura 3.5: Logotipo del proyecto

Tareas de mejoras en la seguridad

En cuanto a las librerías que aportan la funcionalidad de identificación y registro al framework de Ruby On Rails, se encuentran *Devise* y *RodAuth*. En el trabajo previo se hizo una comparativa para saber qué opción era mejor escoger, por lo que haremos una revisión de si seguimos optando por la misma vía. A continuación, en la tabla 3.4, se muestra una tabla comparativa actualizada de ambas librerías de acuerdo con los resultados de sus repositorios de GitHub, en los que hay datos como el número de personas que participan activamente en el desarrollo del proyecto, cada cuánto se hace una nueva release, etc.

Como viendo que, a pesar que la popularidad de Rodauth no es tan elevada como la

Rodauth	Repositorio	**Devise**
1,700	Estrellas	24,000
27	Observadores	454
97	Forks	5,500
31 días	Ciclo de lanzamientos	100 días
Hace 1 mes	Última versión	Hace 8 meses
Casi diario	Último commit	Hace 1 mes
Ruby	Lenguaje	Ruby
MIT License	Licencia	MIT License

Cuadro 3.4: Comparación entre los repositorios de estas librerías de gestión de usuarios para RoR.

de Devise, su ciclo de lanzamientos y mantenimiento es mucho más activo, se ha optado por elegir RodAuth como encargado de gestionar tanto los accesos a la API como a la plataforma web. Además RodAuth permite, de forma sencilla, establecer en el archivo de configuración de rutas de RoR los niveles de acceso según la identificación de usuarios. (**awesomeruby˙2022**)

En cuanto a mejoras que se han implementado en cuanto a la seguridad, decir que se ha optado por eliminar la funcionalidad de envío de contraseña a las personas que eran registradas en nuestra plataforma. Ahora, cuando se les registra se les envía un correo pidiéndoles que entren para establecer su contraseña. De esta forma, además, podemos encriptar la contraseña del usuario antes de guardarla. Del mismo modo, si se le olvida y quiere recuperarla, se da autonomía para ello, evitando así que sea el terapeuta el que tuviese que consultarlo él mismo desde el perfil del usuario en el apartado de gestión de usuarios del panel de control.

De igual forma, se ha mejorado el guardado de las contraseñas de los usuarios, centrándonos en evitar el almacenamiento de credenciales en texto plano. Para lograr esta implementación, se emplean funciones de cifrado unidireccionales que generan valores imposibles de revertir para obtener la contraseña original, lo cual imposibilita que un tercero obtenga el dato real incluso si llegase a tener acceso a la base de datos. Esto lo logramos a través del uso del *SecurePassword* de Rails y *lockbox* de acuerdo con la documentación oficial (Kane, s.f.; on Rails API Documentation, s.f.; on Rails Guides, s.f.). Para reforzar este proceso, se inyecta una porción aleatoria de información en cada secuencia, lo que se conoce como "sal" y que dificulta aún más la labor de un posible atacante que intente recrear las combinaciones generadas. Asimismo, cualquier cambio o creación de nuevas claves obliga a que el sistema aplique de nuevo dichos procesos de cifrado.

Además se ha separado en dos bases de datos diferentes la gestión de los usuarios del resto de la lógica de la aplicación. De este modo podemos tener la base de datos de los usuarios, que contiene información sensible, como el correo electrónico, en un entorno más securizado que el resto, evitando que un posible atacante a nuestro sistema gestor

de bases de datos le dé acceso a toda la información en su conjunto. Esta segregación de datos sensibles permite reducir la superficie de ataque al limitar la exposición de información crítica en caso de un posible compromiso del sistema. Además, proporciona un control granular de acceso, lo que facilita la implementación de políticas específicas que refuercen la protección de los datos personales de los usuarios. Al mantener estos datos en un entorno separado, se asegura un mejor cumplimiento normativo frente a regulaciones como el RGPD (Europea & de Justicia y Consumidores, 2018), ya que se prioriza la protección de la información personal.

A continuación, se procede a la implementación de las tareas detalladas para la web siguiendo el diseño definido a partir de los bocetos desarrollados, una vez han sido corregidos y aprobados por las tutoras del proyecto.

Base de datos

Si atendemos a la Figura 3.6 se expone el esquema que describe la estructura y relaciones entre los diferentes modelos que conforman la aplicación web. Este esquema representa las relaciones y atributos clave para lograr la gestión de los usuarios, las actividades dinámicas, las etiquetas que las conforman, las mediciones de los sensores del reloj, y los niveles de estrés computados.

Entre ellos, además, se incluyen los dedicados a la autenticación y gestión de cuentas, responsables de almacenar la información de terapeutas y proteger las credenciales y claves de acceso mediante cifrado y recuperación segura. Esta parte de la aplicación busca, como ya hemos mencionado en el punto anterior, facilitar el cumplimiento de buenas prácticas en ciberseguridad y permitir la separación de la información en distintas bases de datos para salvaguardar los datos más sensibles. De esta forma, aseguramos que se cumple la Ley de Protección de Datos y Garantía de Derechos Digitales (BOE, 2018).

Por otro lado, se encuentran los modelos que representan a los usuarios finales y su interacción con las actividades asignadas. Para cada actividad, se ha habilitado la posibilidad de adjuntar etiquetas, configuradas con texto e imágenes. Estas etiquetas pueden ordenarse y se muestran en los dispositivos *wearable* de acuerdo con la lógica definida por los terapeutas.

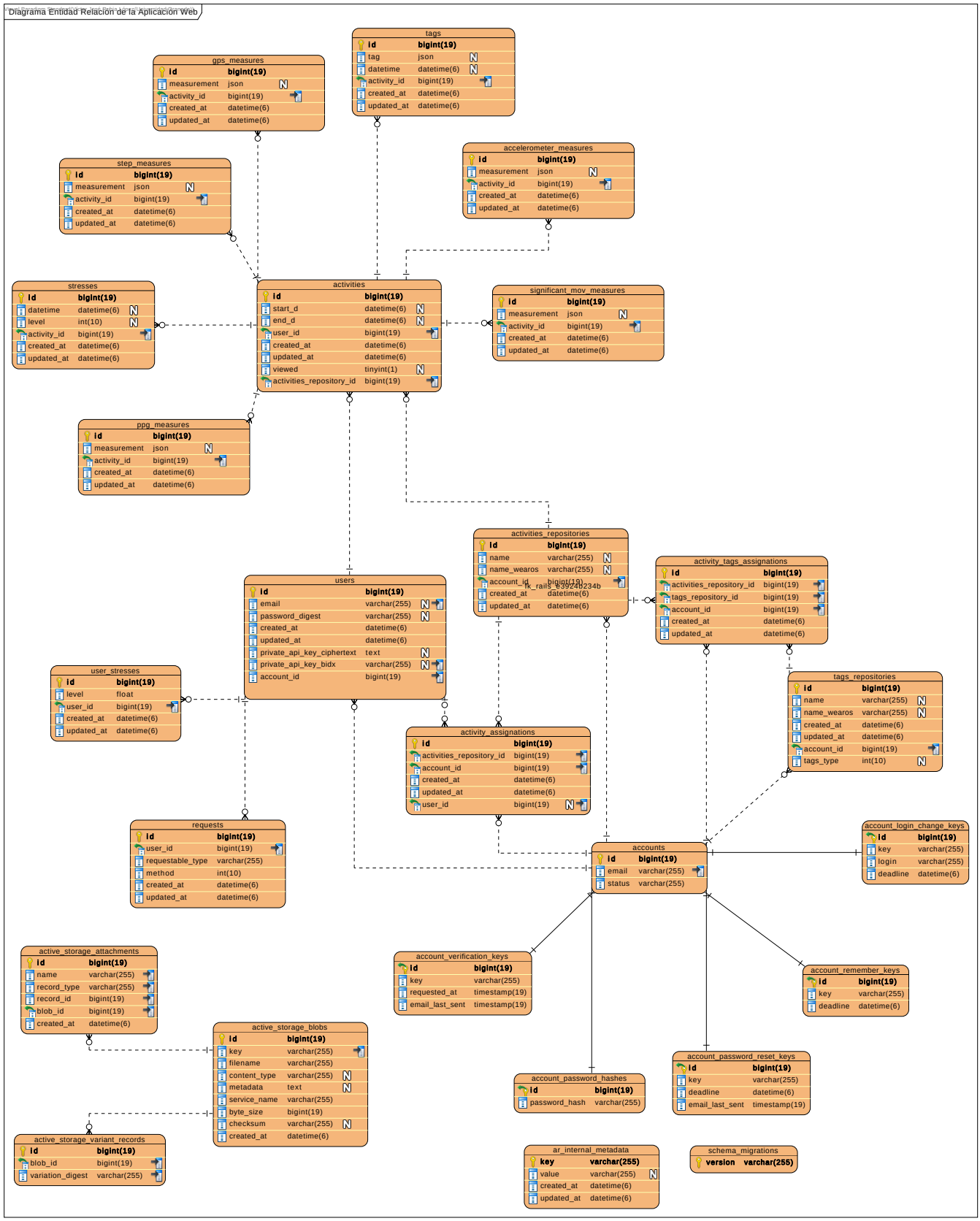
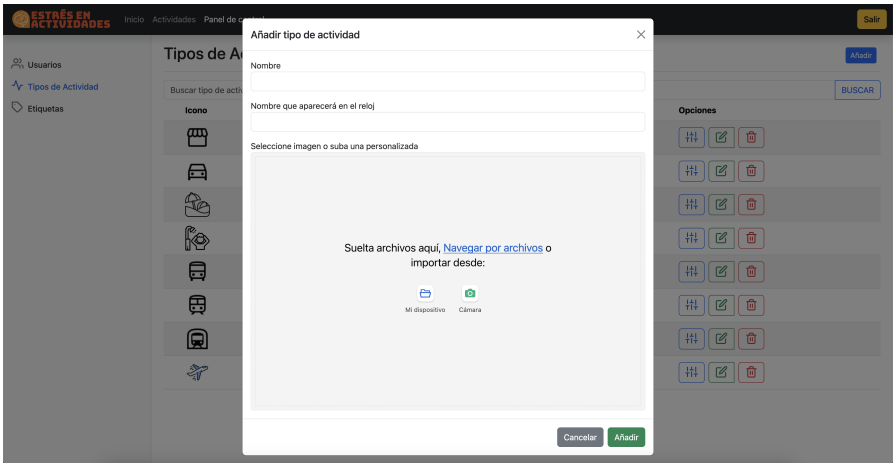
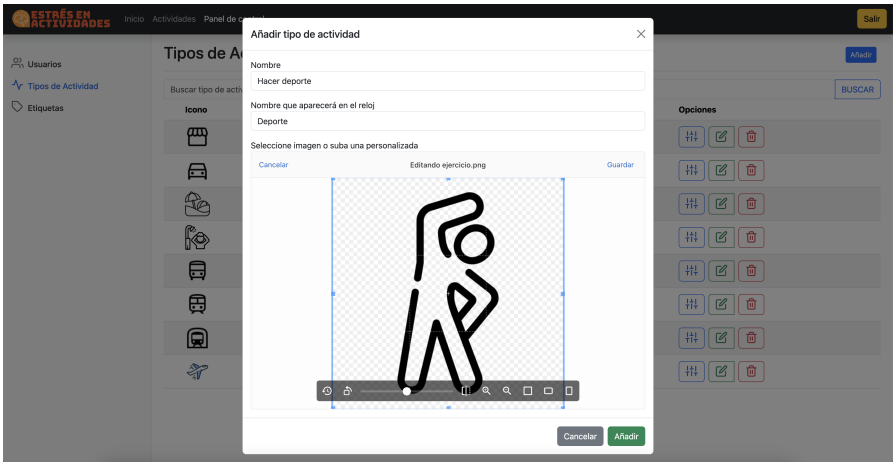


Figura 3.6: Esquema de tablas generadas en la Base de Datos por RoR

La aplicación también cuenta con modelos diseñados específicamente para almacenar mediciones de los sensores del reloj inteligente, como aquellos que gestionan datos de señales de fotopletismografía (PPG), registro de la posición vía GPS y acelerómetro, entre otros. Este enfoque permite mantener organizada la información fisiológica y contextual de cada sesión, lo que a su vez se refleja en la plataforma web a través de gráficas y vistas geolocalizadas. Además, gracias a la integración de bibliotecas de subida y edición de imágenes (como *Uppy*) y a la vinculación con servicios externos (`Active Storage` con `AWS S3`), se facilita la administración de elementos multimedia que ilustran cada tipo de actividad o etiqueta, como se muestra en las Figuras 3.7a y 3.7b.



(a) Formulario para añadir nuevas actividades



(b) Apartado para edición y personalización del icono subido

Figura 3.7: Formulario para la adición de nuevos tipos de actividad junto con la subida de imágenes con *Uppy* y *AWS S3*.

Para la estimación y análisis de estrés, la lógica de procesado de la señal fisiológica

reside en el modelo de datos de niveles de estrés y estrés de los usuarios. El primero de ellos se encarga de calcular el nivel de estrés en ventanas de tiempo para la realización de una actividad. Por otro lado, el segundo se encarga de calcular la probabilidad de que un usuario padezca estrés en base a las actividades que lleva realizadas según su histórico.

Tras la generación de modelos necesarios y adaptación de los ya existentes, necesitamos implementar los controladores y las vistas para implementar las funcionalidades que se piden.

Implementación de APIs para interacción con otros dispositivos

Partiendo del trabajo que ya se tenía desarrollado, se tiene la implementación de una API con acceso restringido mediante un esquema de autenticación *Bearer*. Esta aproximación, ampliamente recomendada en la bibliografía consultada (Polito, 2021; Shekh-Yusef et al., 2015), consiste en la generación de un *token bearer* que es remitido al cliente, como la aplicación móvil o el reloj inteligente, tras la validación de las credenciales de usuario y contraseña en un punto de acceso específico de la API. Con este token, el cliente puede, posteriormente, realizar diversas solicitudes a la plataforma, tales como iniciar una actividad, registrar datos de sensores o finalizar la actividad, entre otros, manteniendo un adecuado nivel de seguridad en las comunicaciones.

Este tipo de autenticación supone que, al *iniciar sesión* correctamente, el servidor genera y devuelve un *token bearer* cifrado, cuya validez se fundamenta en una clave privada almacenada como variable de entorno. Una vez obtenido, el cliente incluye dicho token en las cabeceras (*headers*) de sus peticiones HTTP, acreditando con ello la identidad del usuario en la plataforma. Para que esta estrategia sea efectiva, se requiere que el sistema web opere bajo HTTPS y cuente con un certificado digital SSL válido, protegiendo así la confidencialidad y la integridad de la información intercambiada. En entornos de desarrollo y producción podremos trabajar con un certificado auto-firmado, ya que, de otra forma, necesitaríamos contactar con una entidad acreditada para ofrecer este tipo de certificados.

Durante el desarrollo de la aplicación, se fueron adaptando, del trabajo previo, los distintos puntos de entrada (*endpoints*) que sirven esta autenticación y que se muestran en la Tabla 3.5. En conjunto, permiten al usuario, debidamente autenticado, llevar a cabo operaciones tales como la obtención de su token, la recuperación de contraseña, la creación y consulta de actividades, la finalización de actividades y el registro de medidas de sensores, entre otras. A continuación, se describen brevemente algunos de estos *endpoints*:

En la práctica, cada una de estas operaciones requiere la presentación de un *token bearer* válido como prueba de autenticación. Con ello se garantiza que un usuario no acceda a los datos de otros ni que se generen registros en el sistema sin identificar quién los realiza. Esta estrategia cumple con los principios de ***OAuth 2.0***, en los que

TIPO	DIRECCIÓN	DESCRIPCIÓN
GET	/users/get_api_key	Para obtener el token Bearer dados un correo y una contraseña registradas en el sistema.
GET	/users/password_recovery	Para solicitar un correo con el recordatorio de contraseña dado un correo electrónico.
GET	/activities_from_user	Obtiene los tipos de actividades asociadas a un usuario.
POST	/activities	Dado un tipo de actividad concreta a realizar, se registra.
PUT	/activities/:id	Termina la actividad iniciada estableciendo una fecha de finalización.
POST	/ppg_measures	Registra un conjunto de medidas PPG asociadas a una actividad.
POST	/accelerometer_measures	Registra un conjunto de medidas del acelerómetro asociadas a una actividad.
POST	/gps_measures	Registra un conjunto de medidas del GPS asociadas a una actividad.
POST	/step_measures	Registra un conjunto de medidas del sensor de pasos asociadas a una actividad.
POST	/significant_mov_measures	Registra un conjunto de medidas del sensor de movimiento significativo asociadas a una actividad.
POST	/tags	Asocia un conjunto de etiquetas seleccionadas por el usuario a la actividad en curso.

Cuadro 3.5: Endpoints desarrollados para la funcionalidad de la aplicación web

se fundamenta la mayoría de los sistemas modernos de autenticación y autorización en aplicaciones distribuidas.

Implementación de un sistema de loging en contenedores con el stack ELK

Siguiendo buenas prácticas para el desarrollo de aplicaciones web, se integra un sistema de registro de logs que permite recopilar, analizar y visualizar toda la información relacionada con las operaciones y el uso de la aplicación web. Para ello, se ha optado por desplegar el stack ELK (Elasticsearch, Logstash y Kibana) dentro de un contenedor, aprovechando así las ventajas de la virtualización y la escalabilidad propia de los entornos basados en contenedores. En el siguiente listado aportamos los objetivos y beneficios destacables:

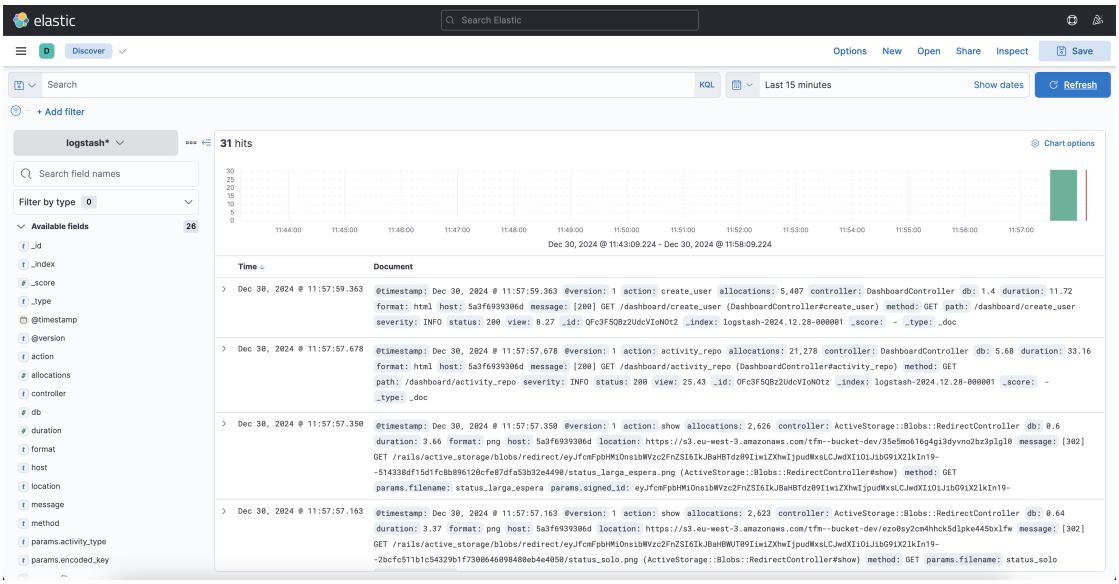
- **Trazabilidad de operaciones**. Registrar todas las interacciones realizadas en la aplicación, identificando al usuario que realizó la acción, el tipo de operación (con-

sultas, modificaciones, errores, etc.) y los tiempos de respuesta. Esta trazabilidad facilita la detección y resolución de problemas de manera rápida y eficiente.

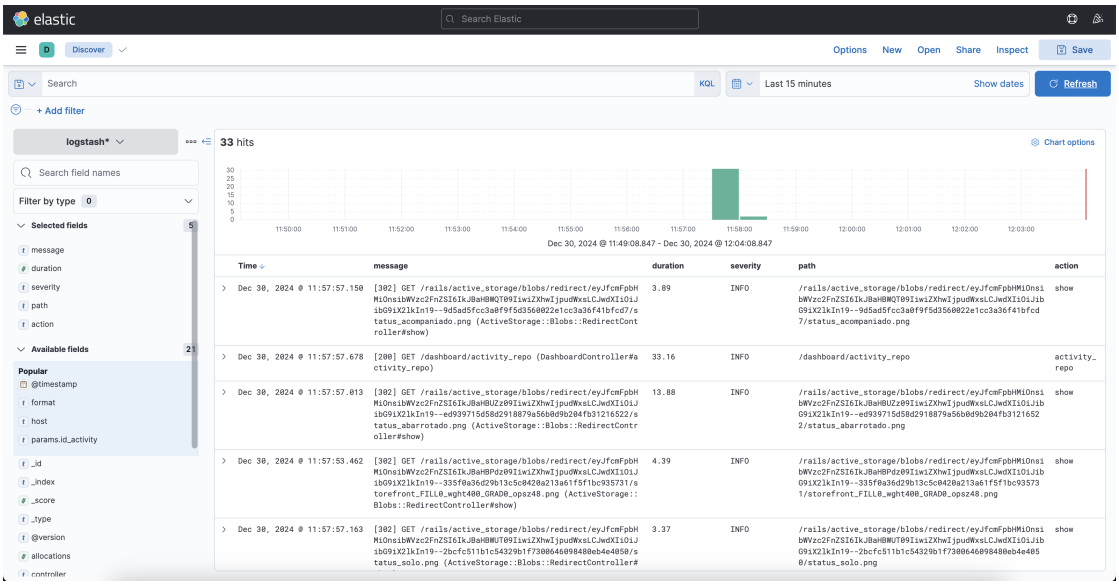
- **Centralización de logs**. Todos los eventos generados por la aplicación se almacenan en un único sistema centralizado, simplificando la consulta y el análisis de datos. Esto incluye información relevante como estadísticas de uso, métricas de rendimiento y excepciones críticas que requieren atención.
- **Visualización y análisis avanzado**. Gracias a Kibana, los datos pueden convertirse en cuadros de mando interactivos (dashboards) y KPIs relevantes. Esto ayuda en:
  - La monitorización en tiempo real del estado de la aplicación.
  - La creación de informes detallados sobre el uso de la plataforma, que pueden ser consultados por administradores o equipos de soporte.
  - La identificación de patrones de uso y la detección temprana de incidencias, aportando *insights* para una mejora continua.
- **Generación de informes y análisis de uso**. El sistema facilita la obtención de informes específicos a nivel de usuario, para detectar tendencias de uso y comportamientos particulares que permiten optimizar la experiencia de la plataforma.
- **Interoperabilidad con otros sistemas**. Al almacenar la información en formato bruto y estructurado en Elasticsearch, los logs están disponibles para su posterior consumo por otros servicios o herramientas de análisis. Esto favorece la integración con sistemas de monitorización adicionales y la generación de sinergias con otras plataformas.
- **Escalabilidad y flexibilidad**. La arquitectura basada en contenedores y el uso del stack ELK facilitan la ampliación de la capacidad de procesamiento de los logs según las necesidades del proyecto. Asimismo, esta flexibilidad permite integrar el sistema de logging con otros microservicios y plataformas sin afectar al desempeño de la aplicación principal.

En las Figuras 3.8a y 3.8b se puede apreciar el resultado de la implementación y la funcionalidad de los logs.

En la Figura 3.8a se presenta una vista, en bruto, de los logs recopilados en el sistema utilizando ElasticSearch, mostrando todos los campos disponibles. Esta vista incluye detalles como la fecha de las solicitudes, las acciones realizadas, los controladores utilizados, las rutas solicitadas, la duración de las operaciones y los mensajes asociados. Por otro lado, en la Figura 3.8b se observa una configuración más depurada, donde se seleccionan atributos específicos como el mensaje, la duración, la severidad, la ruta y la acción. Esta personalización permite una visualización de las interacciones que se están realizando en la web.



(a) Vista en Kibana de logs en bruto, mostrando todos los datos disponibles sin filtros específicos.



(b) Vista en Kibana personalizada con atributos seleccionados, enfocada en las interacciones realizadas con la web.

Figura 3.8: Ejemplo de vistas en Kibana: logs en bruto frente a una vista personalizada con atributos seleccionados.

Flujo de uso

Para comenzar a utilizar la aplicación web, conforme accedemos, encontraremos la página inicial, mostrada en la Figura 3.9, la cual se compone de la barra de navegación, como se detalló en los bocetos. Este elemento de la interfaz nos sirve para acceder a los diferentes recursos de la aplicación web.

Figura 3.9: Página inicial de la aplicación Web

Lo primero que debemos hacer es darnos de alta como terapeuta en la plataforma "Registrarse", utilizando el apartado destinado para ello, tal y como se muestra en la Figura 3.10a. Una vez hecho esto, el usuario recibirá un correo electrónico que le notificará de la creación de la cuenta, tal y como se muestra en la Figura 3.10b.

Tras esto ya podremos identificarnos cuando sea necesario en la aplicación web mediante el botón de la barra de navegación "Entrar", mostrándose el formulario en la Figura 3.11.

Una vez identificados, podemos acceder a los recursos de la aplicación web, como el apartado *Actividades* o *Panel de Control*.

Comenzando a explorar por el *Panel de Control*, nos encontramos con 3 submenús, *Usuarios*, *Tipos de actividad* y *Etiquetas*. El menú de usuarios nos permite añadir, editar y eliminar a nuestros usuarios de medición de estrés, como se muestra en la Figura 3.12 y 3.13. Además, en la Figura 3.14 vemos un apartado para la asignación de los tipos de actividad que hay en el sistema creados.

Continuando con el submenú de *Tipos de actividad*, será aquí donde definamos nuevas actividades para que los usuarios puedan realizarlas. Una actividad se caracteriza por un nombre, con el que aparece en la aplicación web, un nombre para el reloj, el cual le aparecerá al usuario y un icono descriptivo, tal y como se muestra en la Figura 3.15. Desde esta sección, además, podemos editarlas, eliminarlas y configurarlas. La configu-

ración de actividades nos permite asignarles etiquetas en cierto orden del repositorio que se tiene en el sistema. Para configurarlas, como se señala en la Figura 3.16, nos aparecerán dos cajas habilitando el *drag & drop* entre ellas, una con el repositorio de etiquetas creadas en el sistema y otra con las etiquetas que definen la actividad en cierto orden secuencial de aparición en el reloj a gusto del terapeuta.

Por último, para el apartado de *Panel de Control*, encontramos el submenú de *Etiquetas*. En esta sección, encontramos todas las etiquetas ya registradas a modo de repositorio, teniendo la opción de añadir etiquetas nuevas, así como editar o eliminar las ya existentes. Todo esto se puede apreciar en las Figuras 3.17 y 3.18, donde remarcamos la funcionalidad de poder ver cómo va a visualizar el usuario en su reloj dichas etiquetas. Se puede comprobar su buena visibilidad y la forma de registrar etiquetas, que están caracterizadas por un nombre, con el que aparece en la aplicación web, un nombre para el reloj, un icono que les aparecerá a los usuarios en el reloj, y al tipo de etiqueta que pertenecen, *Contexto* o *Estado*.

Prosiguiendo con el recurso de *Actividades* dentro de la barra de navegación de la aplicación web. Este es el apartado donde encontraremos las mediciones asociadas a actividades que hayan hecho los usuarios. Por ello, lo primero que encontramos cuando entramos a este apartado es un listado con los usuarios que tengamos creados. Este listado se caracteriza por el nombre del usuario, un botón para consultar sus actividades realizadas y una columna que define la probabilidad de que el usuario padezca estrés crónico de acuerdo con el test psicológico del DASS-21, como queda reflejado en la Figura 3.19. Al pulsar el botón azul, se despliega el listado de actividades asignadas a los usuarios, como se aprecia en la Figura 3.20, y si tiene medidas de alguna de las actividades, se mostrarán a modo de listado como se observa en la Figura 3.21. En este apartado podemos ver, tanto el histórico de actividades realizadas de este tipo para dicho usuario, como aquellas que están dándose en directo, pudiéndose eliminar o visualizar el detalle.

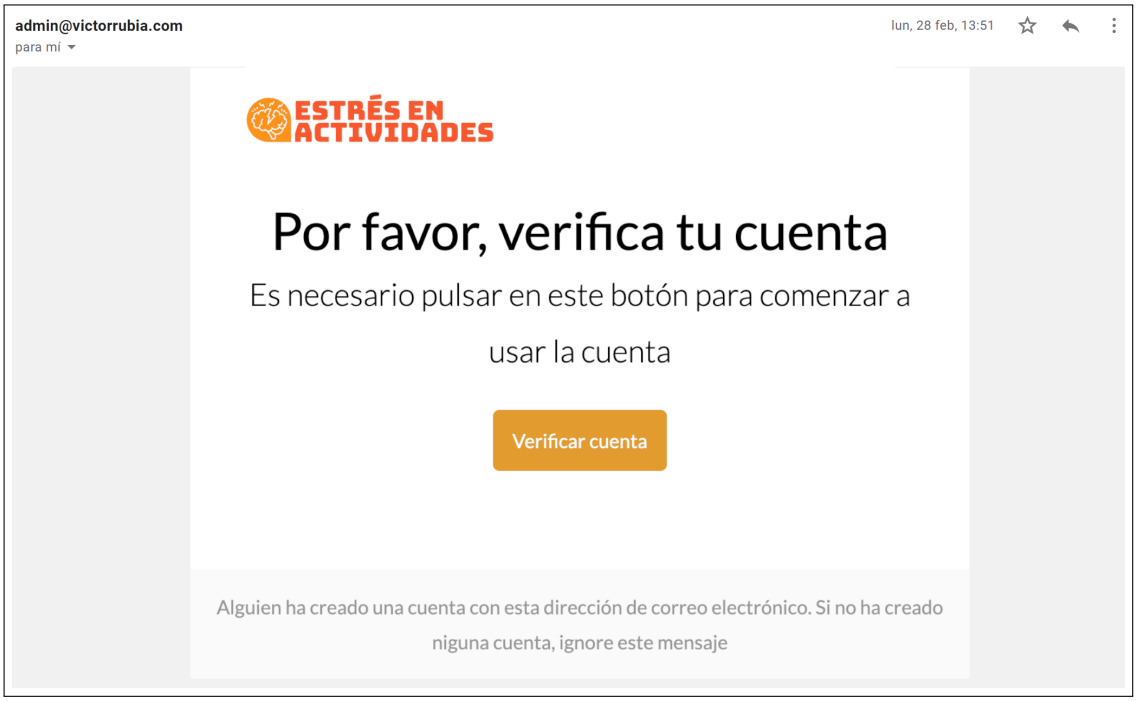
En la Figura 3.22 queda reflejado el detalle de una actividad. Esta vista está compuesta por una parte superior, donde se detalla el nombre de la actividad y quién la ha realizado, junto con unos botones de navegación para volver hacia atrás, y un grupo de dos botones para exportar en crudo la medición de los sensores o si se quiere reprocesar, en caso de que el algoritmo de cálculo de estrés haya cambiado.

En el cuerpo de la vista podemos ver tres cajas, la principal de ellas destinada al gráfico de niveles de estrés durante el periodo de tiempo transcurrido para la realización de la actividad. Definimos 3 niveles de estrés de acuerdo con el algoritmo implementado y detallado más adelante en la Sección **??**, *Alto Estrés*, *Medio Estrés* y *Bajo Estrés*. Este gráfico, que se muestra en la Figura 3.23, es interactivo, es decir, si el punto está representado por una estrella, significa que tenemos un registro de etiquetado para ese instante y, en caso contrario, encontraremos un simple círculo. Para acceder al registro, simplemente debemos posicionar el ratón sobre la estrella del gráfico.

Las otras dos cajas, en la parte inferior se corresponden, por una parte a un mapa, en

el que se muestra el recorrido realizado por el usuario en el transcurso de su actividad, tomando las mediciones del sensor de GPS, y por otra a una tabla en la que se puede ver el registro de etiquetados de una forma más esquemática para facilitar la visualización del registro de etiquetas llevado en el transcurso de la actividad.

(a) Página para registrarse como terapeuta



(b) Correo que se envía al usuario mayor para que conozca sus datos de acceso

Figura 3.10: Registro y verificación de terapeuta en la plataforma web

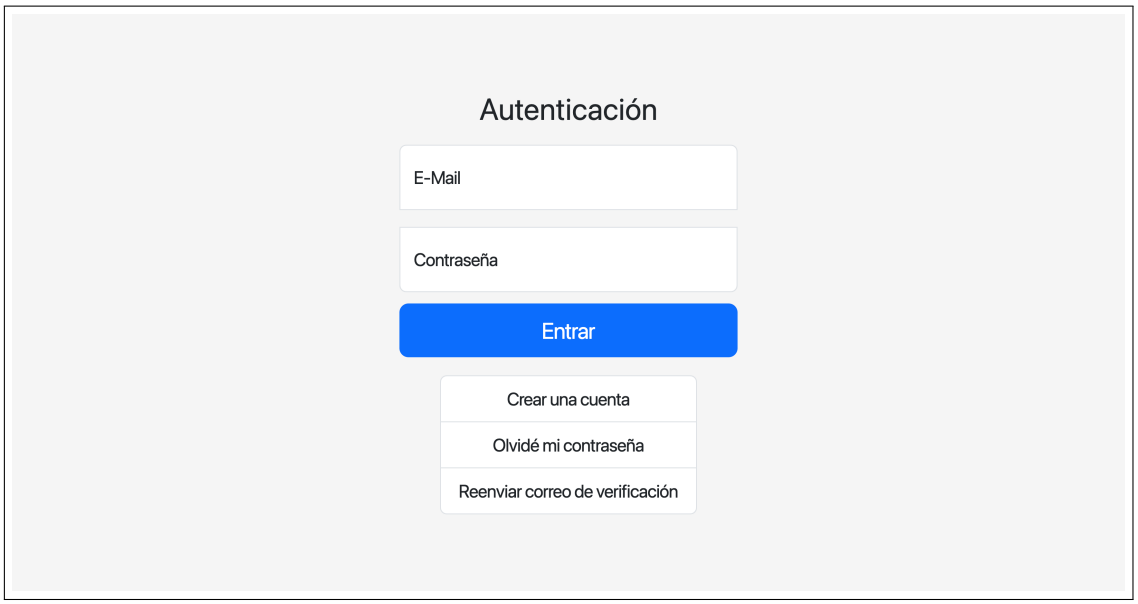


Figura 3.11: Página para la autenticación cuando ya se dispone de un usuario como terapeuta

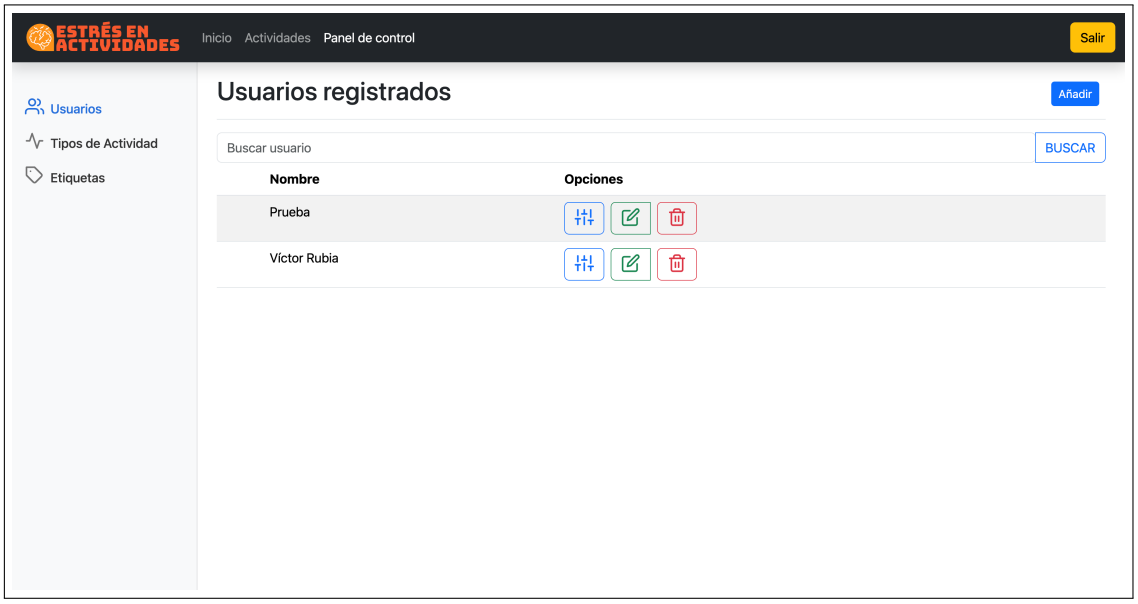


Figura 3.12: Usuarios para hacer mediciones registrados en el sistema.

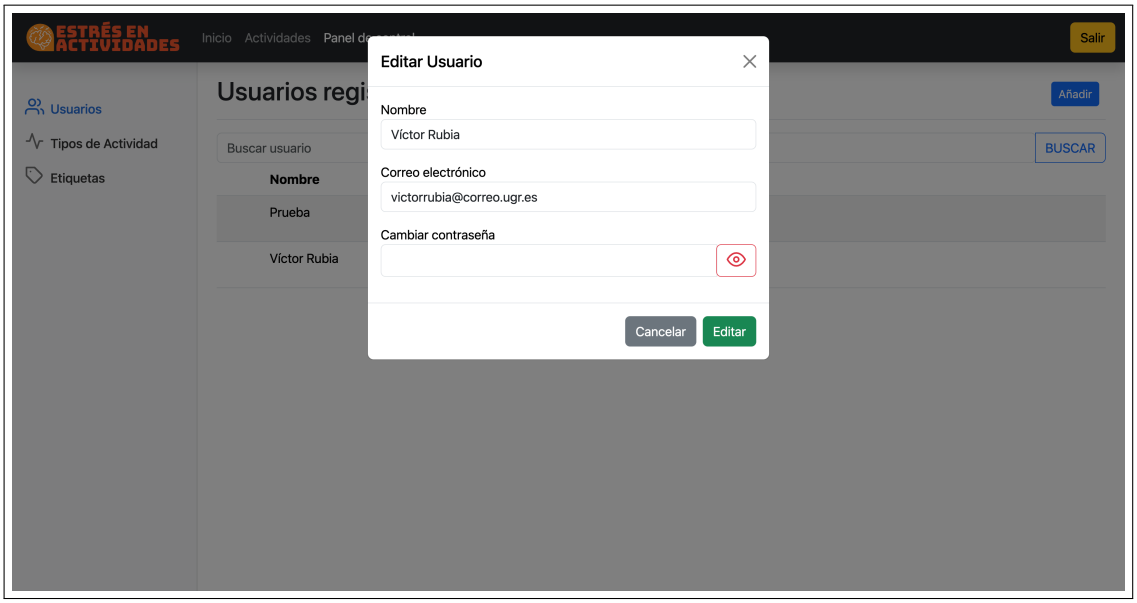


Figura 3.13: Formulario para editar un usuario existente.

Figura 3.14: Apartado para asignar tipos de actividad a un usuario.

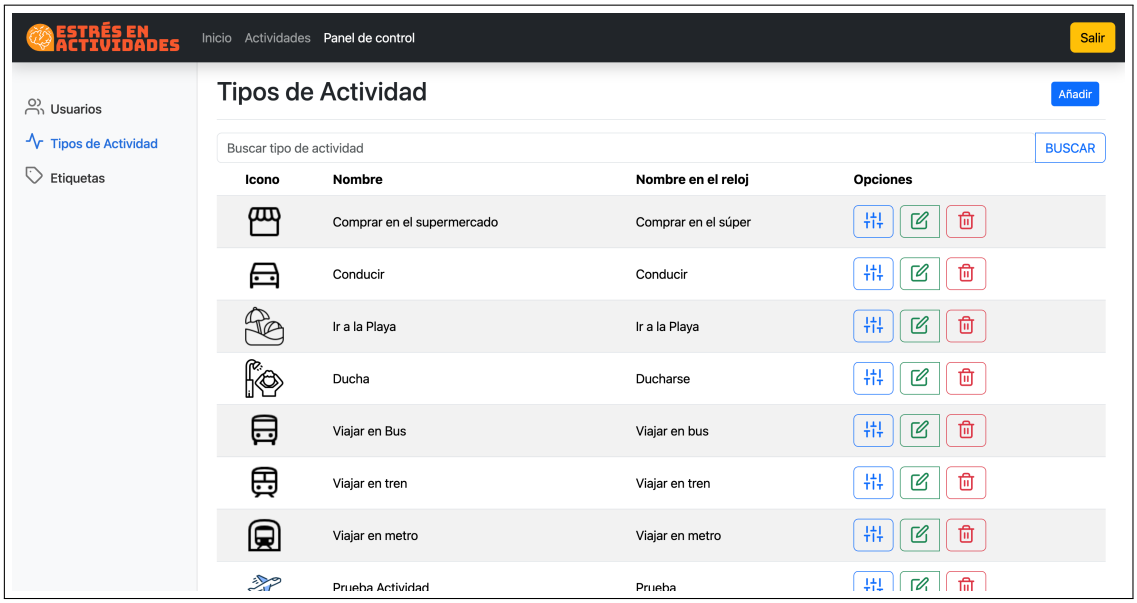


Figura 3.15: Listado de tipos de actividad registrados en el sistema.

Figura 3.16: Configuración de etiquetas y su orden para una actividad.

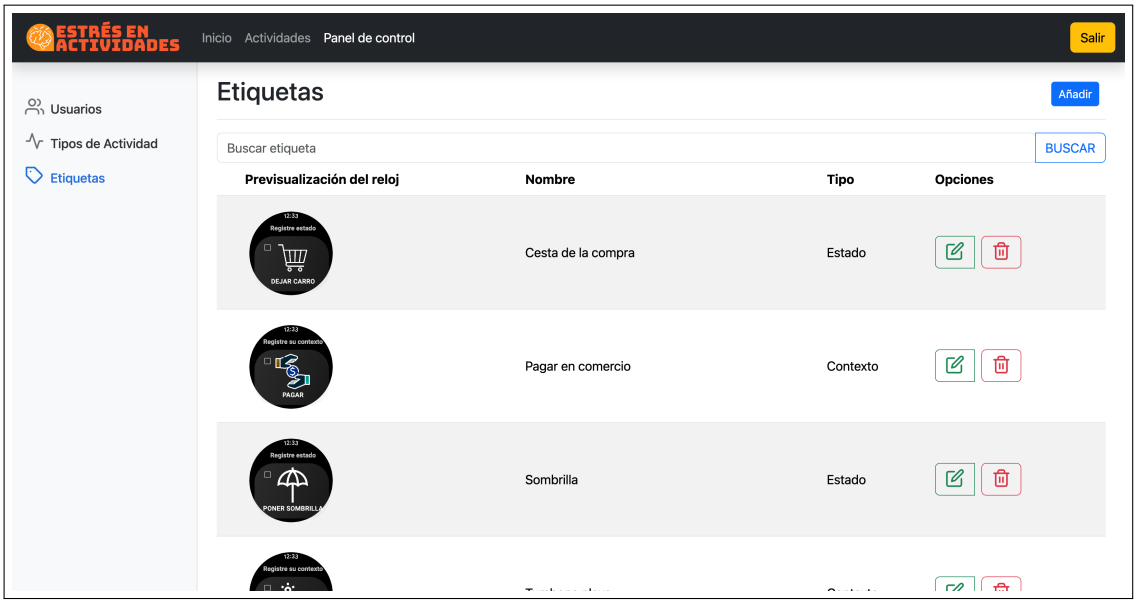


Figura 3.17: Listado de etiquetas registradas en el sistema.

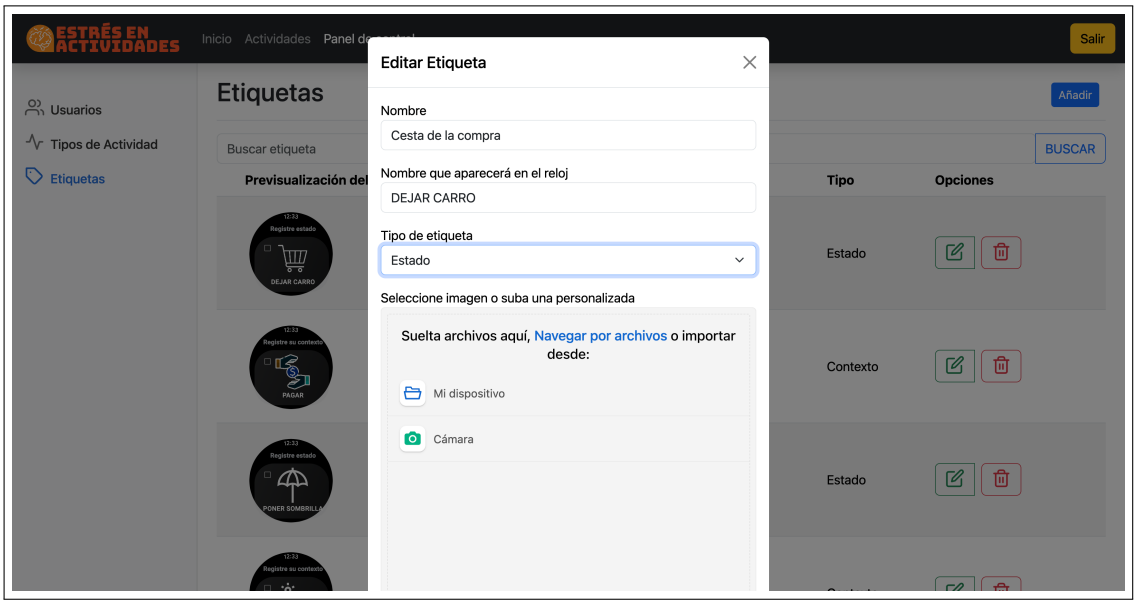


Figura 3.18: Formulario de edición de una etiqueta registrada en el sistema.

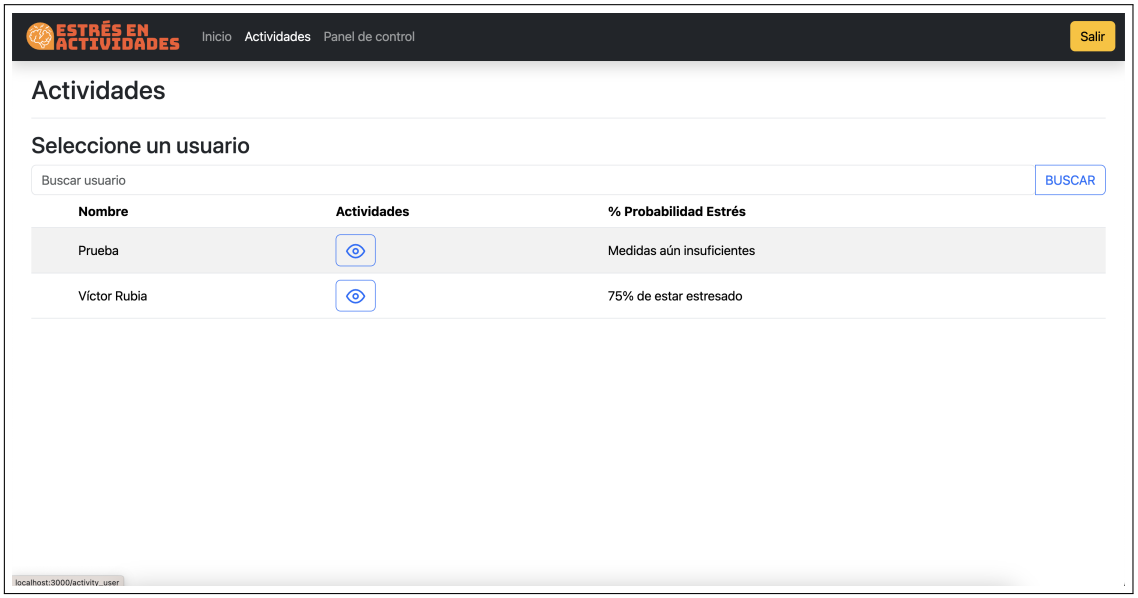


Figura 3.19: Listado de usuarios dados de alta en el sistema para los que ver sus mediciones.

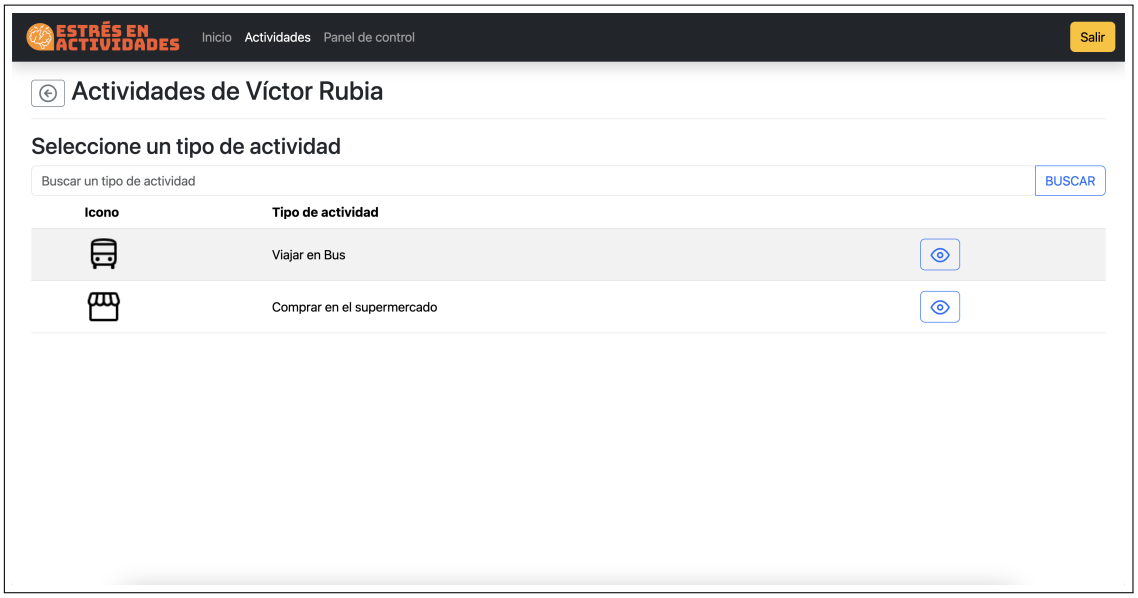


Figura 3.20: Listado de actividades asignadas para un usuario.

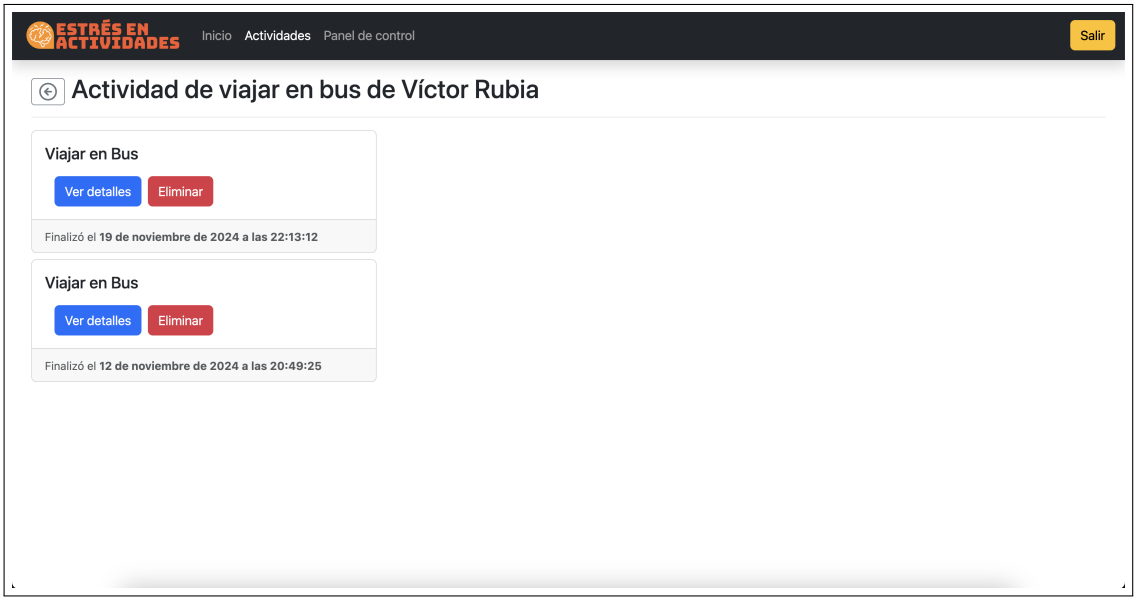


Figura 3.21: Registro de mediciones para un tipo de actividad y usuario.

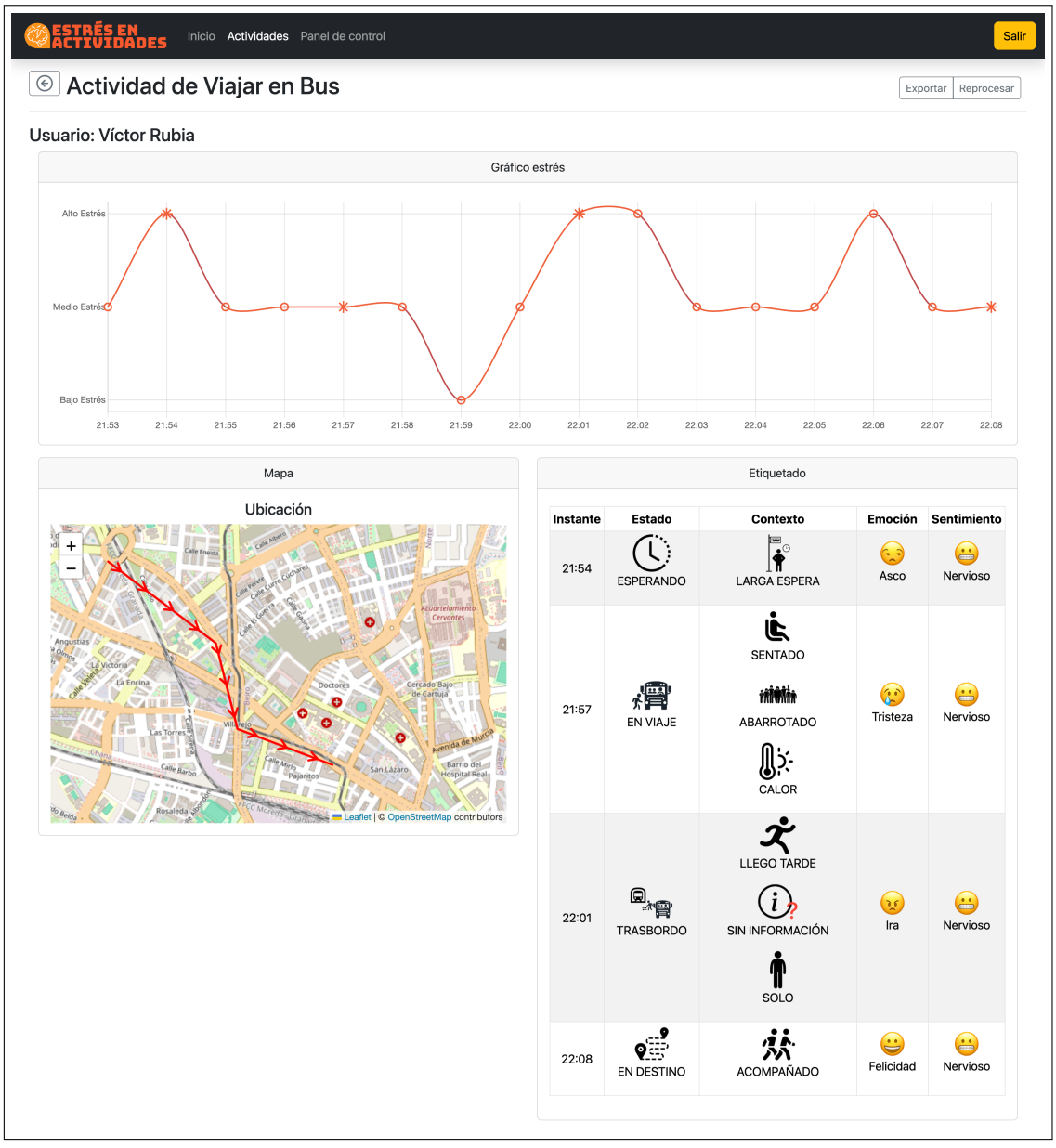


Figura 3.22: Vista detallada de una actividad realizada.

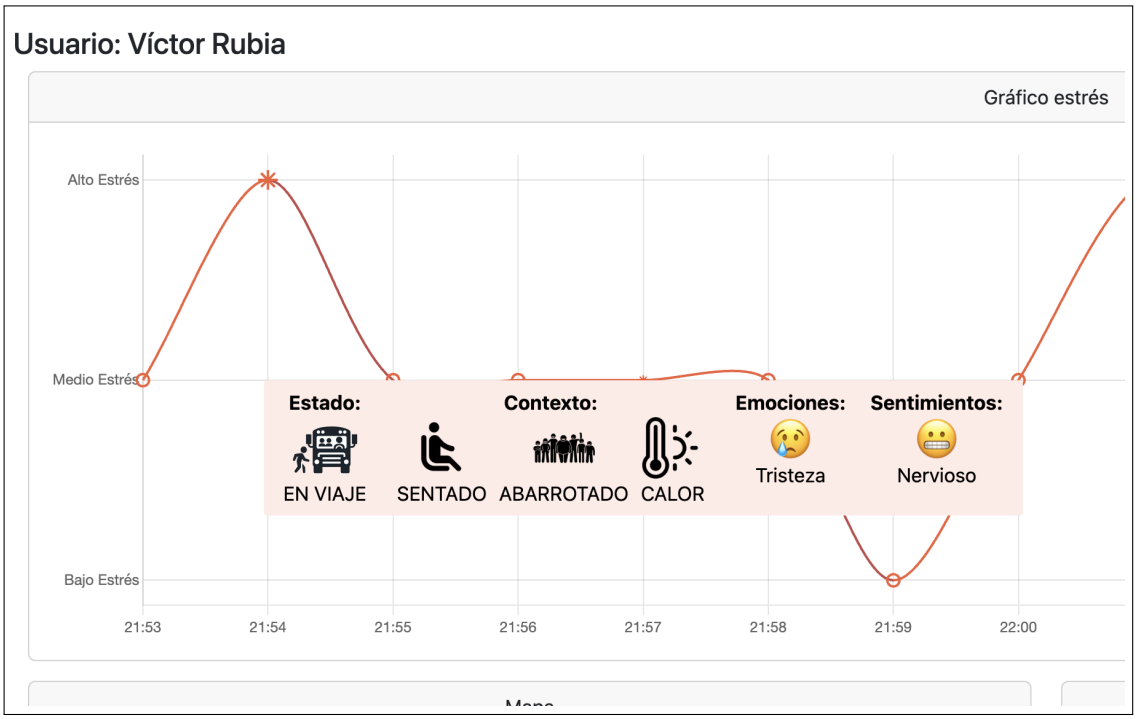


Figura 3.23: Interacción con el gráfico para ver el etiquetado registrado.

Pruebas de aceptación

En cuanto a las pruebas de aceptación realizadas para el sistema web, se han desarrollado tests de Ruby que comprueben que la funcionalidad desarrollada cumple con los requerimientos. Los resultados pueden observarse en la tabla de la Figura 3.24 junto con la tabla de la Figura 3.6, la cual no se expone aquí de forma completa por acortar la extensión del documento.

ID	PRUEBA DE ACEPTACIÓN	ESTADO	OBSERVACIONES
1	Poder registrarse como terapeuta introduciendo datos de correo y contraseña	Correcto	Comprobado en test unitario *test_creating_an_account*
2	Registrarse con un correo que está previamente registrado debe dar error	Correcto	Comprobado de forma manual, se muestra un error
5	Iniciar sesión con las credenciales correctas y que no exista ningún error	Correcto	Comprobado en test unitario *test_logging_in*
...	...	...	...
11	El terapeuta debe estar identificado para acceder al apartado de gestión de usuarios	Correcto	Comprobado mediante test unitario *test_should_get_users* y *test_required_authentication*
12	Se listan los usuarios registrados actualmente en el sistema	Correcto	Comprobado mediante test unitario *test_should_get_users*
15	Se muestra un botón en la sección de gestión de usuarios para añadir un usuario	Correcto	Comprobado mediante test unitario *test_creating_an_user* y *test_should_create_user*
16	Introducir un correo electrónico ya existente hará que no se guarde el usuario	Correcto	Comprobado de forma manual, se muestra un pop-up de error
...	...	...	...
29	El terapeuta debe estar identificado para visualizar los niveles de estrés de una actividad concreta	Correcto	Comprobado mediante test unitario *test_required_authentication*
30	El gráfico muestra si existe alguna medición para dicha actividad	Correcto	Comprobado mediante test unitario *test_should_get_activity_details*

31	El gráfico debe enlazar los puntos mediante una línea suave, incluso si no existe medición en tiempos correlativos	Correcto	Comprobado de forma manual, se unen los puntos aunque no sean correlativos, sin dejar un espacio en blanco
32	El terapeuta debe estar identificado para visualizar las actividades categorizadas	Correcto	Comprobado mediante test unitario *test_required_authentication*
...	...	...	...
72	El correo debe enviarse a la misma dirección con la que se registró el terapeuta	Correcto	Se ha comprobado de forma manual con el código y la ejecución
73	El correo debe contener un enlace para confirmar el estado de la cuenta	Correcto	Se ha comprobado de forma manual con el código y la ejecución
74	Al cerrar sesión no se puede acceder a contenido restringido para usuarios identificados	Correcto	Se ha comprobado mediante test unitario *test_required_authentication*
75	Al cerrar sesión se puede identificar nuevamente	Correcto	Se ha comprobado mediante test unitario *test_loggin_in_and_loggin_out*

Cuadro 3.6: Resultados de realizar las pruebas de aceptación más relevantes de la aplicación web

TestRun	Total	Passed
All Tests - Generated by RubyMine on 26/12/24 9:38	32	32
Api::V1::ActivityControllerTest::Unauthorized		
test_should_not_get_activities	1542ms	passed
test_should_not_update_another's_activity	758ms	passed
Api::V1::TagControllerTest::Authenticated		
test_should_create_tag	761ms	passed
ActivitiesControllerTest		
test_should_destroy_activity	761ms	passed
test_should_update_activity	182ms	passed
DashboardControllerTest		
test_should_get_view_activities	707ms	passed
test_required_authentication	296ms	passed
test_should_get_activity_details	1521ms	passed
test_should_get_activities	230ms	passed
test_should_get_index	279ms	passed
test_should_get_users	238ms	passed
Api::V1::UsersControllerTest::Unauthorized		
test_should_not_get_user	157ms	passed
UsersControllerTest		
test_should_update_user	232ms	passed
test_should_create_user	216ms	passed
test_should_destroy_user	372ms	passed
DashboardTest		
test_searching_an_user	289ms	passed
test_creating_an_user	350ms	passed
test_edit_an_user	307ms	passed
test_remove_an_user	465ms	passed
PpgMeasuresControllerTest		
test_should_destroy_ppg_measure	312ms	passed
test_should_update_ppg_measure	231ms	passed
WelcomeControllerTest		
test_should_get_index	30ms	passed
AuthenticationTest		
test_creating_an_account	49ms	passed
test_logging_in_and_logging_out	298ms	passed
Api::V1::PpgMeasureControllerTest::Authenticated		
test_should_create_ppg_measure	247ms	passed
TagsControllerTest		
test_should_destroy_tag	215ms	passed
test_should_update_tag	189ms	passed
Api::V1::UsersControllerTest::Authenticated		
test_should_get_user	178ms	passed
test_should_get_user's_apiKey	150ms	passed
Api::V1::ActivityControllerTest::Authenticated		
test_should_create_activity	297ms	passed
test_should_handle_404	229ms	passed
test_should_update_activity	1972ms	passed

Figura 3.24: Tests unitarios realizados sobre la aplicación web

3.4. Aplicación Móvil

3.4.1. Introducción

Antes de dar inicio a este punto, resulta necesario explicar la necesidad de tener una aplicación para el móvil. La aplicación del móvil se concibe a raíz de la necesidad de tener que dar soporte a un sistema multiusuario en la aplicación del reloj inteligente (descrita en la sección 3.6). El desarrollo parte de lo existente ya para el Trabajo Fin de Grado realizado, sobre el cual se hacen ligeras adaptaciones y mejoras para poder lograr implementar las nuevas funcionalidades en los diferentes sistemas.

El objetivo de la aplicación móvil es servir al terapeuta o al usuario de una interfaz capaz de identificar a un usuario registrado por un terapeuta en la plataforma web para que el reloj y la web puedan asociar al usuario que está realizando la medición con el envío de datos.

3.4.2. Situación previa y motivaciones de la actualización

La justificación de la necesidad de tener una aplicación móvil pasa por un estudio que se detalló en el Trabajo Fin de Grado. La motivación del estudio se basó en encontrar una forma cómoda, óptima y cercana a lo estándar en este tipo de sistemas para llevar a cabo una autenticación. El resultado de dicho análisis determinó que la introducción de credenciales directamente en el reloj inteligente resultaba poco práctica y accesible, una conclusión que se alinea con las directrices oficiales de Google (Google, 2024d) y Apple (Apple, 2024), que específicamente recomienda minimizar la entrada de texto en dispositivos wearables debido a las limitaciones inherentes de sus interfaces.

Por ello, se optó por desarrollar una aplicación móvil complementaria que facilitara este proceso de autenticación, aprovechando la mayor comodidad que ofrece la pantalla táctil del teléfono y su interfaz más amplia para la introducción de datos.

Atendiendo a las diferentes opciones que tenemos disponibles para el desarrollo, se optó por comenzar a desarrollar en Flutter, por tener conocimientos de haberlo usado anteriormente y por ser un framework que permite desarrollar aplicaciones de calidad de una forma bastante rápida. Sin embargo, a medio desarrollar la aplicación surgió un problema y era el cómo este framework implementaba la conexión con relojes inteligentes. El hecho es que, a pesar de que existen algunas librerías que ofrecen cierta funcionalidad, no está soportado de forma nativa y las guías oficiales de desarrolladores de Android no indican cómo hacerlo de esta forma, sino que, únicamente, indican cómo hacerlo mediante el desarrollo de una aplicación Android nativa mediante Java o Kotlin.

Por esta razón, se decide abandonar el desarrollo de la aplicación móvil en Flutter y se comienza a estudiar cómo realizarla de forma nativa mediante Kotlin. Se establece que seguiremos el patrón arquitectónico Model-View-ViewModel (MVVM) ya que es el que recomienda Google y el que más extendido está dentro de sus ejemplos.

Partiendo del Trabajo Fin de Grado, donde la aplicación se concibió sobre el SDK 31 de Android, se desarrolla utilizando la versión 8 de Java y dependencias que, en aquel momento, se consideraban estables y estándares en el ecosistema de desarrollo Android. Esta configuración respondía a la disponibilidad tecnológica del momento, que con el transcurso de los años, tanto Android como las bibliotecas asociadas han evolucionado de forma sustancial. En concreto, algunas librerías clave, incluyendo componentes de la familia *AndroidX* y herramientas de conexión a servicios web como Retrofit, ofrecen funcionalidades más eficientes y seguras en sus versiones más recientes. Sin embargo, esas novedades exigen contar con una versión más avanzada del SDK de Android y con un estándar de Java superior al inicialmente adoptado.

Por tanto, en el presente Trabajo Fin de Máster se aborda una actualización al SDK 34, alineada con la versión 14.0 de Android. Gracias a esto contamos con las últimas características del sistema y podemos soportar dispositivos más actuales. Este cambio se acompaña con la migración de Java 8 a Java 17, motivada por las mejoras en el rendimiento que se ofrecen en esta versión. Con ello, se garantizan ciclos de desarrollo más robustos y se simplifica la incorporación de bibliotecas que, de otro modo, no podrían aprovecharse de manera completa.

Asimismo, se han renovado todas las dependencias con miras a mejorar la estabilidad y la seguridad de la aplicación. El incremento en la fiabilidad de librerías como *Retrofit*, junto con el soporte de características más recientes en las librerías de *AndroidX*, repercute positivamente en la experiencia tanto de uso, como de desarrollo, y permite la implementación de características más avanzadas de Kotlin, un tratamiento más eficiente de corrutinas, y un soporte de tests más estable.

3.4.3. Tareas de desarrollo realizadas

De forma análoga al apartado donde se describe la aplicación web, para reducir la extensión de la memoria, no detallaremos todas las tareas y tarjetas generadas para cada una de las Historias de Usuario relativas a la aplicación móvil; sin embargo, detallaremos aquellas que han tenido más relevancia y daremos una breve introducción a lo que ya se tenía desarrollado.

Tareas de bocetado

Para llevar a cabo las tareas relacionadas con la aplicación móvil, nos basaremos en los bocetos elaborados con anterioridad, donde se han tratado de representar los elementos característicos de la interfaz de usuario proporcionada por el sistema operativo. Para los iconos, se opta por los que proporciona Material Icons (Google, 2024c), por ser también universales y accesibles en el universo Android.

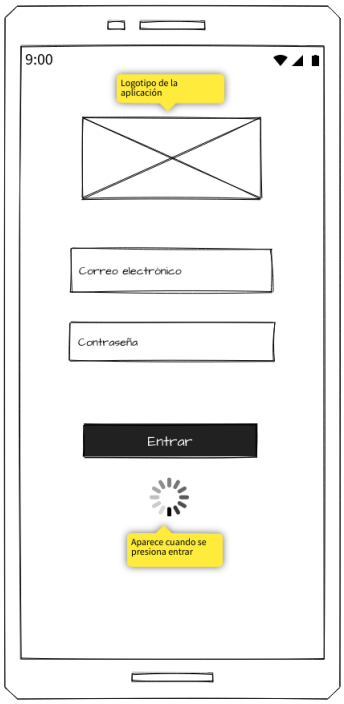


Figura 3.25: Pantalla inicio de sesión en Android

El primer boceto es el de la pantalla de inicio de sesión de la Figura 3.25. En esta pantalla encontramos el logotipo del proyecto junto a un formulario de identificación. Cuando el usuario introduce datos incorrectos, se le avisa mediante indicadores en el propio campo de texto. Una vez se pulse en el botón para entrar, se muestra un indicador de progreso y de corrección en las credenciales.

Figura 3.26: Pantalla de usuarios mayores identificados en la aplicación Android

Cuando se ha comprobado que las credenciales aportadas son correctas, se continúa con la pantalla de la figura 3.26. Está compuesta por una barra superior y una parte principal en la que se ven, englobados en una tarjeta, los datos para el usuario identificado. Además, se proporciona un indicador de estado de conexión con el reloj.

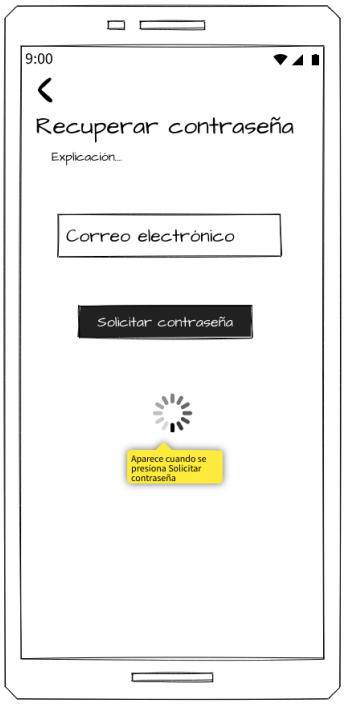


Figura 3.27: Pantalla de recuperación de contraseña en Android

Por último, se muestra en la figura 3.27 la pantalla de recuperación de contraseña por parte del usuario mayor en la aplicación Android. Esta se compone por un botón de regreso a la pantalla principal, y por un formulario en el que se debe introducir el correo electrónico para poder recuperar la contraseña.

Para mostrar la navegabilidad de la aplicación, se muestran en la figura 3.28 estos bocetos con flechas que indican hacia qué boceto se dirige cuando se pulsa sobre un elemento.

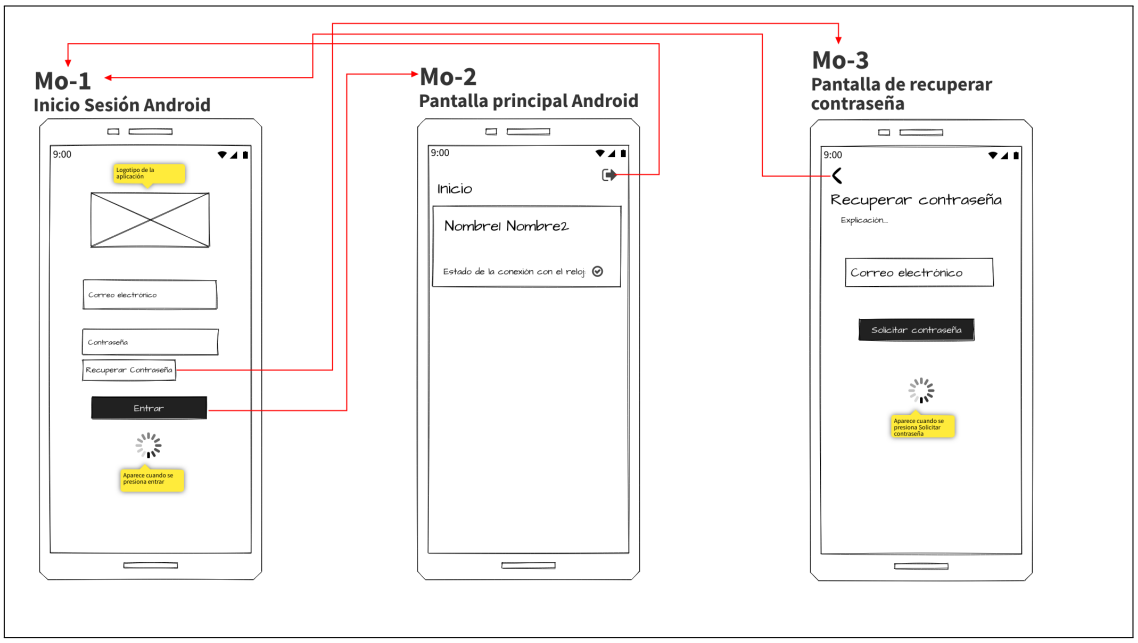


Figura 3.28: Diseño de la navegabilidad de la aplicación Android

Tras la aprobación de los bocetos por parte de las tutoras, realizaremos la implementación de las pantallas. Se puede visualizar el resultado en las figuras 3.29a, 3.29b y 3.29c. Las funcionalidades son poder cerrar sesión una vez esté el usuario identificado. Para la implementación de la recuperación de contraseña, como se representa en la Figura 3.29c se ha optado por una animación *Lottie* para describir gráficamente la finalidad, junto con una descripción en texto.

(a) Pantalla Principal de App Android

(b) Pantalla de Identificados App Android

(c) Pantalla de Recuperación de Contraseña

Figura 3.29: Resultados de implementación en Android de los bocetos

Arquitectura de desarrollo de la aplicación

La estructura a seguir en el proyecto se expone en el diagrama de paquetes de la figura 3.30 siguiendo con las directrices del patrón arquitectónico Model-View-ViewModel (MVVM). En ella se aprecia cómo se separa la capa de datos de la capa de dominio y de la capa de presentación.

El paquete correspondiente a la **capa de datos** almacena los repositorios que contienen las funciones y los modelos para la gestión de los datos con la API proporcionada por el sistema web y el almacenamiento local, ya que se quiere que la aplicación mantenga la sesión aunque se cierre la aplicación. Por otro lado, la **capa de dominio** es la encargada de contener los casos de uso y funciona a modo de interfaz con los repositorios de la capa de datos. El paquete que se corresponde con la **capa de presentación** contiene las dependencias necesarias para el buen funcionamiento de la aplicación, las pantallas, que se componen de vistas, y la ejecución de los casos de uso. Para la conexión con la base de datos se usa la librería *RetroFit 2*.

Por último, se proporcionan los diagramas de clases correspondientes a cada paquete en las figuras 3.31, 3.32 y 3.33.

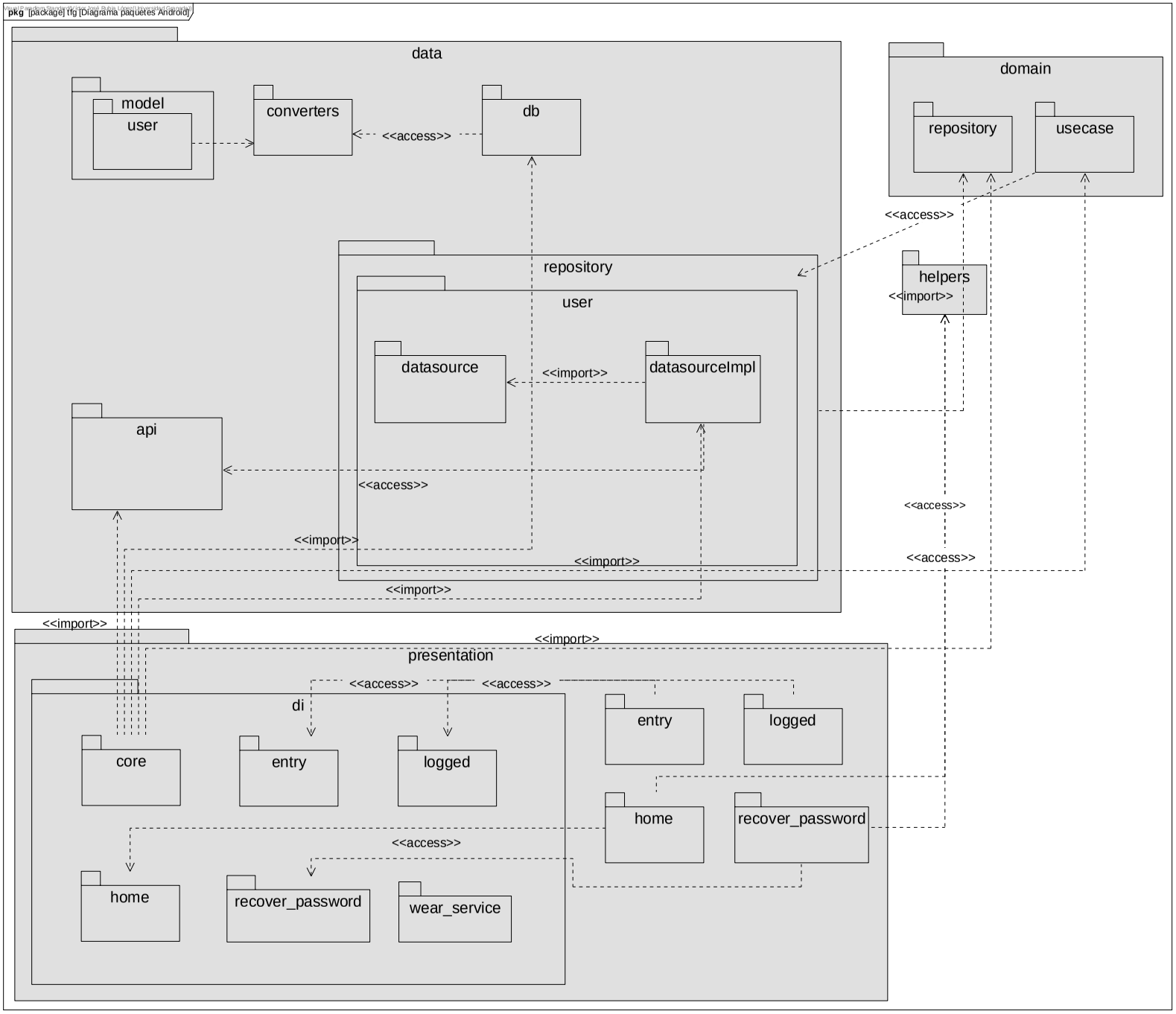


Figura 3.30: Diagrama de paquetes para la aplicación Android.

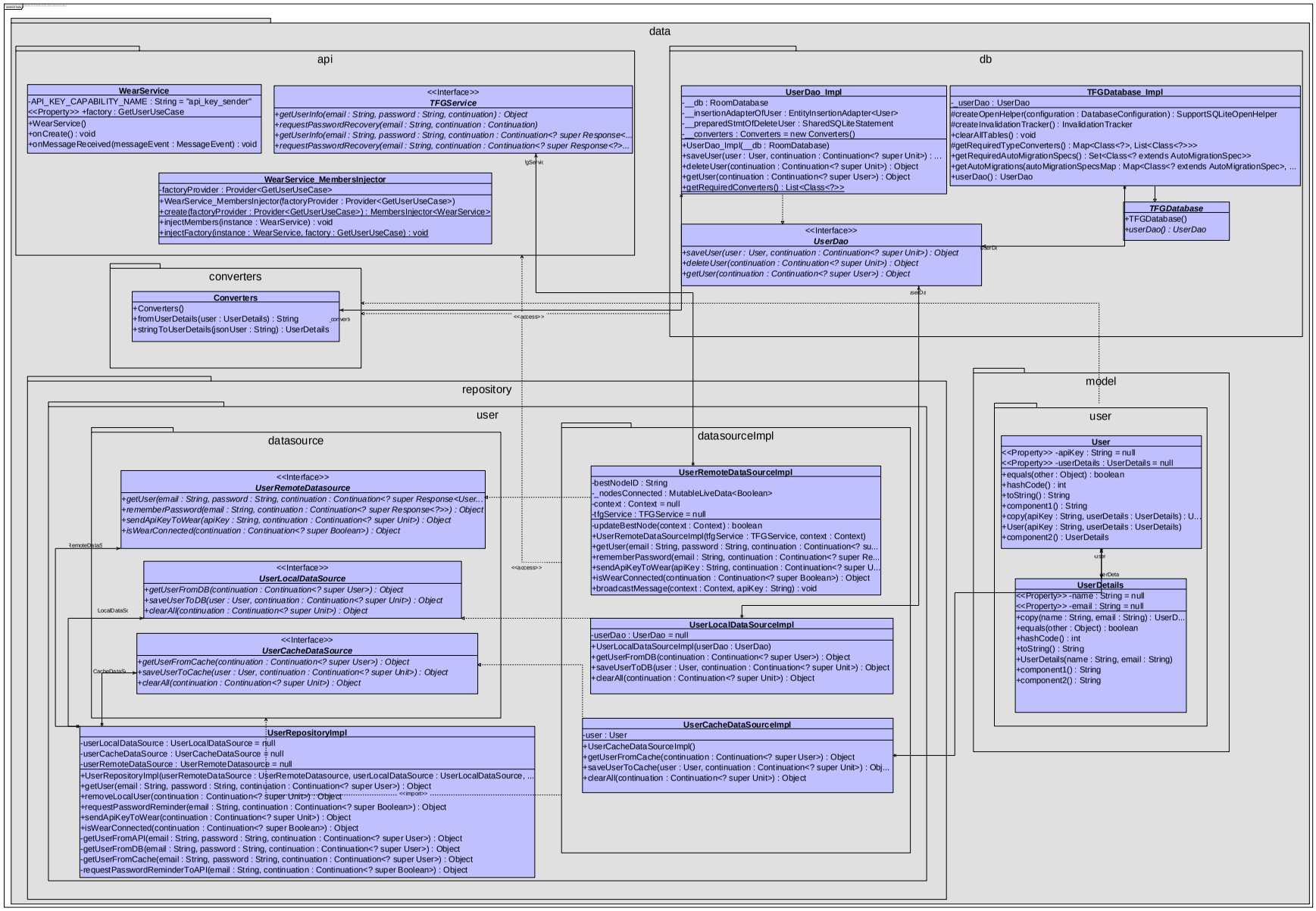


Figura 3.31: Diagrama de clases correspondiente al paquete *data* de la aplicación Android

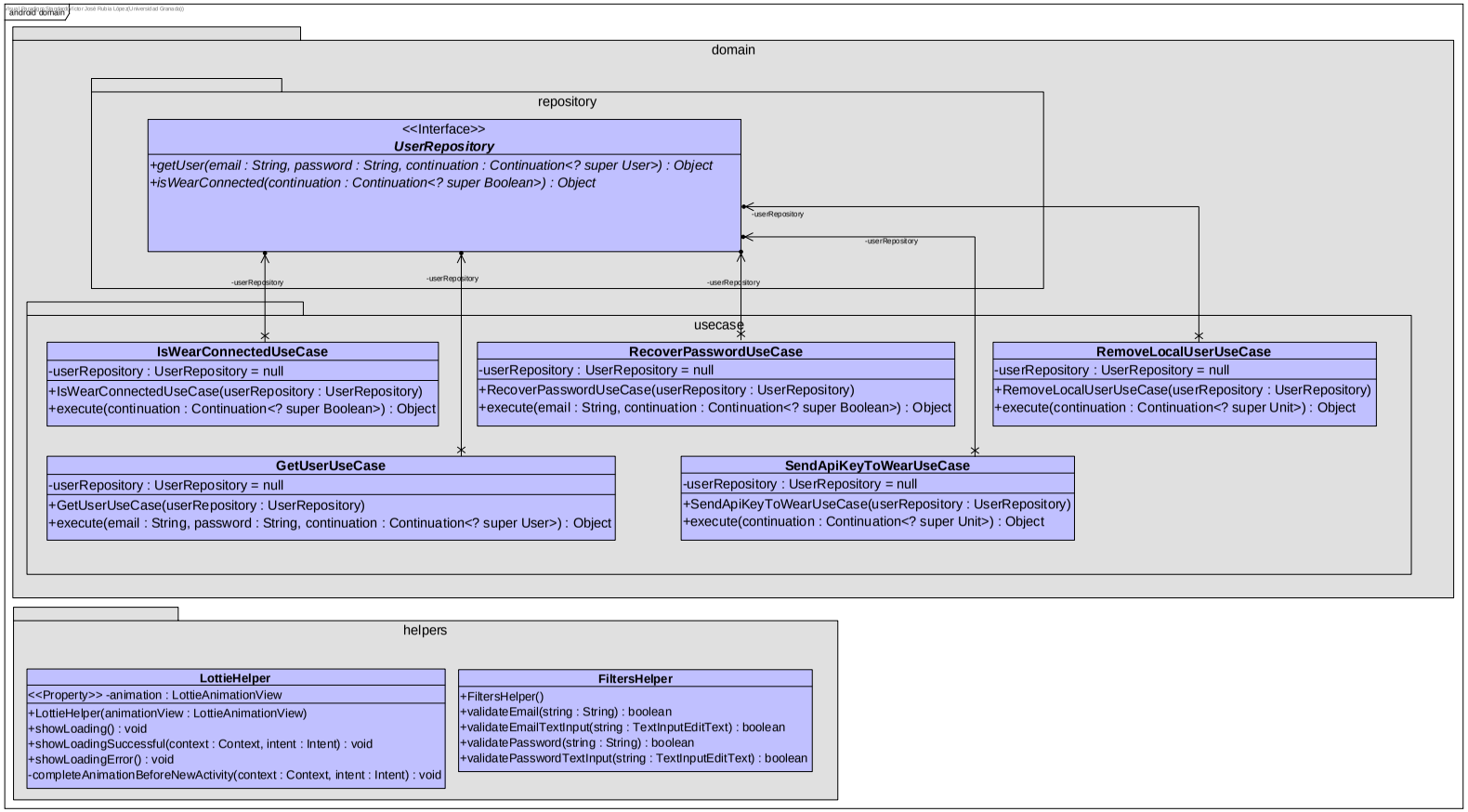


Figura 3.32: Diagrama de clases correspondiente al paquete *domain* de la aplicación Android

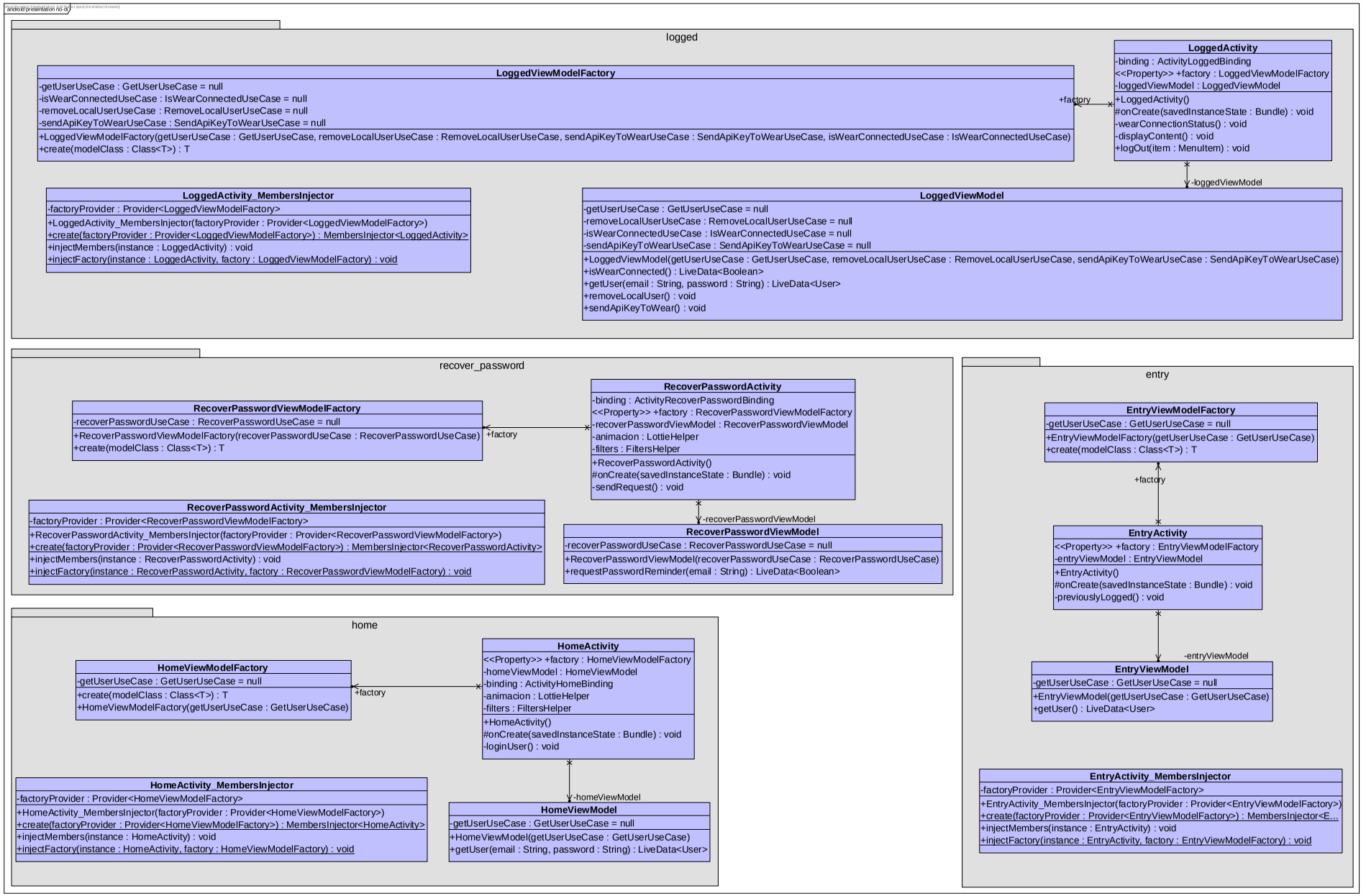


Figura 3.33: Diagrama de clases correspondiente al paquete *presentation* de la aplicación Android

Flujo de comunicaciones

A continuación se plantea cómo debe ser el flujo de comunicaciones entre los diferentes sistemas en la Figura 3.34, y explicaremos la forma en la que interactúan los diferentes sistemas.

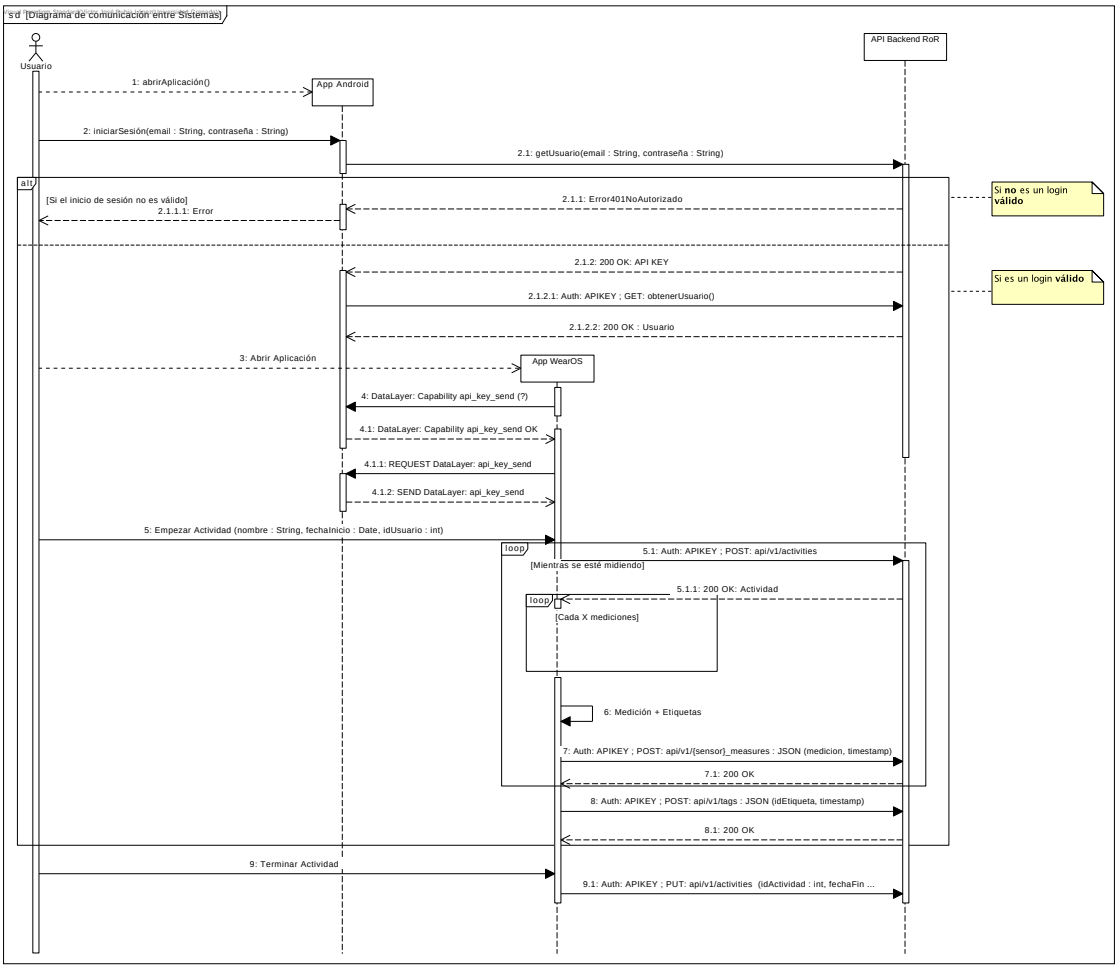


Figura 3.34: Diagrama de comunicación entre sistemas

El proceso comienza cuando el usuario abre la aplicación de su móvil e ingresa sus credenciales. La aplicación verifica estos datos con el sistema web, de modo que, si son válidas, el sistema genera una clave de acceso, que es un token *Bearer*, representado en el diagrama de comunicación como *Auth:APIKEY*, que se transmite al reloj inteligente mediante un mecanismo de transferencia de datos en el que el móvil actúa como intermediario para conseguir identificar al usuario que va a realizar las mediciones en el reloj. Por tanto, este token será el que utilice el reloj para interactuar con el sistema web de forma segura.

Tras esta autenticación, el usuario puede comenzar una actividad desde el reloj, acorde con el punto 3 del diagrama de comunicación, registrando qué actividad va a realizar, la hora de inicio y el identificador de usuario en el sistema web, quedando preparada para recibir datos en tiempo real. Durante la actividad, el reloj estará recopilando las mediciones de sensores, tales como el PPG, el GPS, entre otros, y las enviará periódicamente al servidor para su almacenamiento y posterior análisis, tal y como se indica en el punto 7 del diagrama de comunicación. Además, el usuario puede etiquetar momentos relevantes, como por ejemplo en picos de intensidad, que también se envían al sistema web, de modo que se cumpla con poder proporcionar un contexto adicional para la interpretación de la información. Estas etiquetas se reflejan en el punto 6 del diagrama de comunicación.

Cuando el usuario decide finalizar la actividad, se notifica al sistema web con la hora de parada y cualquier otro dato que reste por enviar. Con esta señal, el servidor se encarga de procesar y organizar toda la información recibida, de modo que se tenga la visión completa del evento y de las etiquetas asociadas. Por último, en caso de que las credenciales que se introducen en la aplicación sean inválidas, el sistema devuelve un mensaje de error del tipo 401 (No autorizado), no pudiendo realizar ninguna acción en el sistema.

Flujo de comunicación con el reloj

A raíz de la explicación del token *Bearer* necesario por parte del reloj para poder enviar sus datos al servidor y que este debe proporcionárselo la aplicación móvil, resulta necesario explicar más en detalle la forma en la que se produce esta comunicación.

La aplicación móvil implementa la funcionalidad necesaria para que cuando un usuario inicie la sesión con sus credenciales, entonces envíe al reloj el token de dicho usuario y el reloj limpie la sesión del usuario previamente identificado. Esto se logra a través de la API que ofrece Google Play llamada *MessageClient* y *ConnectionsClient* (Developers, 2024), la cual es un proceso que está ejecutándose en segundo plano de forma permanente, de modo que, aunque la aplicación del reloj o del móvil estuviesen cerradas, el servicio de envío y recepción de datos seguiría activo. Esto evita que ambas aplicaciones deban estar abiertas para poder ofrecer su funcionalidad.

Esta API se basa en el establecimiento de "capacidades" que son etiquetas que se registran en los dispositivos que se quieren comunicar. Los dispositivos conectados se les llama "nodos", por lo que aquellos "nodos" que quieran transmitir un mensaje a otro nodo, deben primero conocer qué nodos a los que están conectados tienen la "capacidad" a la cual quieren transmitir el mensaje. Aquellos nodos que tienen dicha capacidad, se les conoce como "nodos potenciales" y para establecer comunicación con ellos se hará mediante un identificador de nodo.

Posteriormente, en base a ese identificador del nodo, se le enviará, para esta "capacidad", un vector de bytes con el contenido del mensaje. Esta lógica, por ende, nos

permite que el reloj pueda consultar el *token* del usuario que se identificó por última vez en la aplicación.

Accesibilidad y diseño

Antes de cerrar esta sección, resulta necesario mencionar que para cuidar la accesibilidad y usabilidad en la aplicación móvil, se ha seguido la guía de accesibilidad de Material Design (Google, 2024a). En la implementación se puede apreciar que se ha optado por seguir las directrices de diseño de Material Design en su versión 3, la versión más actualizada de Google. Las razones para seguirlas son la facilidad del usuario a la hora de reconocer elementos de la interfaz, ya que se unifica el diseño de la aplicación con el resto de las aplicaciones del dispositivo. No existe la necesidad de establecer una paleta de colores para la aplicación, ya que el sistema tiene unos predefinidos y éste adapta la tonalidad de la interfaz dependiendo de los ajustes del usuario.

Esto, junto con la implementación de lo necesario para habilitar el uso de TalkBack, que es el lector de pantalla de Android, nos permite describir cada elemento que aparece en pantalla de izquierda a derecha y de arriba hacia abajo, ofreciendo una accesibilidad adecuada.

3.5. Aplicación del Reloj

3.5.1. Introducción

Continuando con la propuesta, expondremos en este punto el desarrollo para la aplicación del reloj inteligente, elemento fundamental para dar completitud a la funcionalidad del resto de los sistemas. Para este sistema, como en los anteriores, partimos de una base sobre la que ya se desarrolló una primera versión del aplicativo, permitiéndonos implementar en ella las mejoras y cambios necesarios para cumplir con la funcionalidad que se especifica en las Historias de Usuario.

La finalidad de la aplicación del reloj inteligente es ofrecer al usuario una herramienta poco invasiva para registrar sus actividades diarias. Mientras las realiza, el reloj recopila datos de sus sensores y los almacena para que se analice, posteriormente, el nivel de estrés en intervalos de tiempo específicos. Además, la aplicación permite añadir etiquetas que describen el contexto o estado del usuario dentro de cada actividad, lo que ayuda a interpretar con mayor precisión los momentos en los que se evalúa el estrés.

3.5.2. Situación previa y motivaciones de la actualización

La aplicación del reloj inteligente con la que se parte de base está desarrollada con una versión 31 del SDK de Android y se utiliza una versión de Java 8. Esta aplicación está concebida siguiendo un diseño basado en *Material Design* para WearOS usando

Jetpack Compose, un framework que hace que el diseño de interfaces en Android se realice siguiendo una filosofía de componentes.

No obstante, al comienzo del desarrollo se planteó, al igual que en la aplicación móvil, utilizar el framework de desarrollo *Flutter*, basado en el lenguaje Dart, que permite realizar código fiable en un tiempo bastante reducido en comparación a otras alternativas. Siguiendo esta línea, se investigó acerca de las posibilidades que se tenían para interactuar con relojes inteligentes *Android Wear*, encontrándonos con una librería llamada *wear* (fluttercommunity.dev, 2021), la cual está mantenida por la comunidad. Sin embargo, la funcionalidad que aporta la librería es únicamente el poder manejar el diseño de interfaces de usuario de estos formatos de pantalla que caracterizan a los relojes inteligentes. De igual forma, se investigaron librerías para el manejo de sensores, como *Sensors Plus* (fluttercommunity.dev, 2024). Sin embargo, ésta se centra más en sensores físicos como el acelerómetro o el giroscopio y no tanto en sensores fisiológicos como el PPG o ECG. Esto motivó a abandonar el desarrollo en este framework y optar por realizarlo en Android Nativo.

La aplicación del reloj implementa la funcionalidad para obtener el usuario identificado en el móvil y, con ello, ofrecer una lista de tres actividades preestablecidas relacionadas con el uso del transporte público. Estas actividades vienen definidas con ciertas etiquetas tanto para los *Estados* como para los *Contextos*. Mientras la aplicación está ejecutándose y la actividad se haya marcado como iniciada, el reloj estará recogiendo datos de su sensor PPG para enviarlos al servidor y que, posteriormente, se procesen. Finalmente, el reloj posee la funcionalidad de indicar que una actividad ha finalizado, concluyéndose la medición y pudiendo crear una nueva medición para una actividad de las que se tienen preestablecidas.

Previo a abordar las tareas que se extraen de las Historias de Usuario, debemos analizar la situación previa y evaluar las necesidades de actualización para garantizar la compatibilidad y la seguridad. Al igual que en la aplicación móvil, se llevó a cabo una actualización a la versión 34 del SDK de Android junto con la migración de Java 8 a Java 17.

3.5.3. Tareas de desarrollo realizadas

A fin de mantener la extensión de esta memoria dentro de límites razonables, no se presentan de manera exhaustiva todas las tareas y tarjetas que se generan a partir de cada una de las Historias de Usuario. No obstante, se exponen en detalle aquellas que tienen una mayor relevancia y, además, se proporcionará una breve introducción de las funcionalidades que ya se tenían desarrolladas.

Tareas de bocetado

El desarrollo de esta aplicación comienza con la investigación de la forma en la que se diseñan las aplicaciones para relojes inteligentes, ya que están claramente condicionados por su forma y tamaño, haciendo que el diseño de interfaces sea sustancialmente diferente. En la Figura 3.35 se muestra el bocetado y navegación completos para representar una primera aproximación de la aplicación del reloj, que fue contrastada con las ideas de las tutoras del presente Trabajo Fin de Máster.

El bocetado comienza con la pantalla que aparece cuando el usuario abre la aplicación, en esta pantalla se mostrará al usuario un feedback de que se están cargando los datos. Previamente, solo se recibía el token *Bearer* con el que, posteriormente, se realizarían las peticiones a la API en la aplicación web. Para lograr la funcionalidad de poder incluir actividades y etiquetas dinámicas para cada usuario, esta lógica ha debido cambiar, ya que cuando se obtiene este token se debe consultar qué actividades tiene este usuario asociadas, qué etiquetas son las que componen dichas actividades y, por último, descargar el contenido multimedia como los iconos que las definen.

En el caso en el que no hubiese red disponible o si el reloj no estuviese conectado a ningún dispositivo con la aplicación de identificación, se indicaría de ambas formas que no se tiene conexión a internet, mediante un icono y un texto descriptivo.

Tras acabar la sincronización, se mostrará al usuario una pantalla con un botón a través del cual puede iniciar una actividad. Cuando el usuario presione este botón, se le mostrará el listado de actividades que tiene asignadas por su terapeuta, entre las que debe elegir aquella que concuerde más con el tipo de actividad que va a realizar. Una vez seleccionada la actividad del listado, se le muestra al usuario una pantalla a modo de feedback que avisa en relación al comienzo de la medición para su actividad. Tras unos segundos, esta pantalla da lugar a otra en la que se muestra un menú, el cual se compone de registrar las etiquetas en el transcurso de la actividad y un botón para cuando se desee finalizar la medición.

Cuando el usuario decide registrar etiquetas para su medición, se le hace una categorización en cuatro secciones, dos para los estados, que se dividen en "Estado" y en "Contexto". *Estado* son aquellas etiquetas que indican en qué momento nos encontramos dentro de la realización de la actividad. Por ejemplo, para una compra en el supermercado, podemos estar eligiendo los productos, pagando en caja o saliendo del supermercado. Por otro lado, *Contexto* serán aquellas etiquetas que definen aspectos concretos del entorno en el que se hace la actividad y que pueden ser estresores, como por ejemplo, hay mucha gente, hace calor, estoy solo, entre otros. Por otro lado, se le muestra una lista de emociones predefinidas, que se extraen de la convención de emociones básicas establecidas por psicólogos recientemente (Cherry, 2024). Estas son felicidad, tristeza, miedo, ira, sorpresa y asco. Para concluir con el etiquetado, se le pide al usuario que registre si se siente nervioso/a, o si por el contrario se siente tranquilo/a. El objetivo es que se pueda contrastar y asociar esta información que nos da el usuario con la que el algoritmo utilizará para procesar el nivel de estrés.

Tras el registro de esta información, la aplicación navegará a la pantalla principal de medición, dando la opción nuevamente de registrar etiquetas o terminar la actividad.

Cabe mencionar que, avanzando en el estudio de Material Design para Wear OS, nos percatamos de que el bocetado realizado estaba basado en las guías de una versión anterior, Android Watch, ya que la herramienta no ofrecía la última versión. De este modo, se optó por realizar un nuevo bocetado de alta fidelidad utilizando Figma y las guías de estilo oficiales publicadas por Google.

Esto viene motivado porque la forma en la que se deben ahora disponer los elementos en la interfaz ha cambiado. Se sustituyen las listas de elementos horizontales por las verticales, ya que el navegar entre las listas puede confundirse con el gesto de volver hacia la pantalla anterior. El resto de bocetos de alta fidelidad se han intentado adaptar a los originalmente propuestos con la previa aprobación de los bocetos de baja fidelidad, ofreciendo así una visión más realista del producto final. El resultado puede verse en la Figura 3.36.

En el bocetado de alta fidelidad, además, se tiene una pantalla para cuando el usuario pierde la conexión a Internet.

Con esto y habiendo seguido las recomendaciones de las guías de estilo de Google, se constata que la aplicación para el reloj cumple estándares de accesibilidad y usabilidad a través de la simpleza de la pantalla, la no desaparición de elementos de forma repentina, indicándose cada acción a realizar y teniendo botones y elementos suficientemente grandes para que puedan ser pulsados sin mayor problema, siempre teniendo en cuenta que se trata de una aplicación de reloj inteligente y que por ello ya se tienen numerosas limitaciones de accesibilidad.

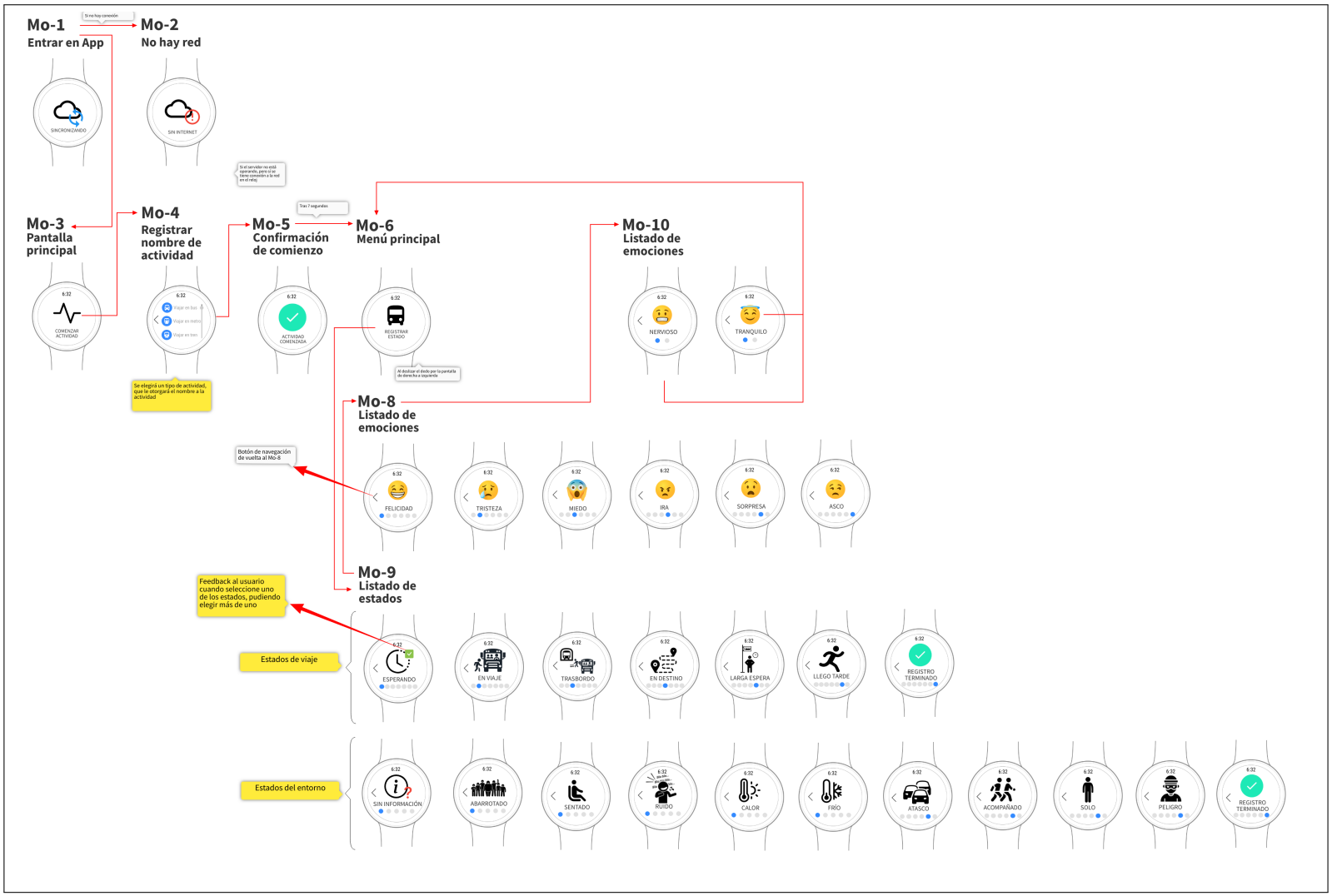


Figura 3.35: Bocetado y navegabilidad en una primera aproximación al desarrollo de la aplicación WearOS

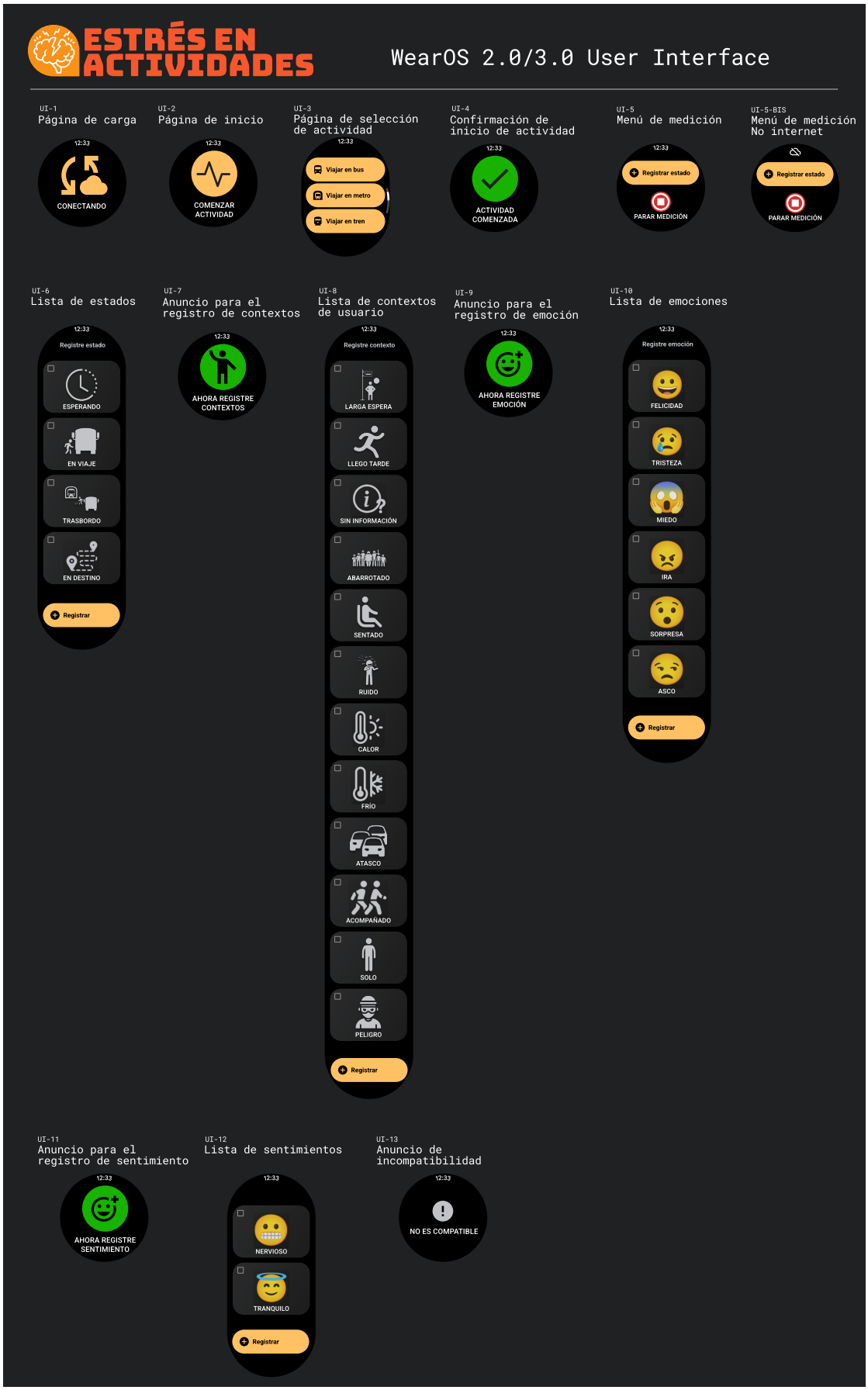


Figura 3.36: Bocetado de alta fidelidad con Figma de la aplicación de reloj inteligente Wear OS

Implementación del bocetado

El resultado de la implementación es idéntico a lo mostrado en la Figura 3.36, por lo que no incluiremos capturas de pantalla asociadas a esto. Para el desarrollo de la aplicación para el reloj inteligente se ha seguido un patrón arquitectónico Model-View-ViewModel (MVVM), ya que es el que se recomienda siguiendo las guías de desarrollo de Android Developers y es lo más extendido en la industria, por su facilidad en mantenimiento y pruebas.

Prueba de ello se muestra en la Figura 3.37, donde se expone el diagrama de paquetes resultante del desarrollo. Cabe mencionar que, para mejorar la legibilidad, solo se ha incluido el paquete correspondiente al almacenamiento del sensor PPG (*ppg_measures*). No obstante, en la práctica se tiene desarrollado un paquete, siguiendo la misma filosofía que para el del sensor PPG, para cada sensor, tanto en el paquete correspondiente al modelo como en el repositorio que se muestra en el diagrama de paquetes.

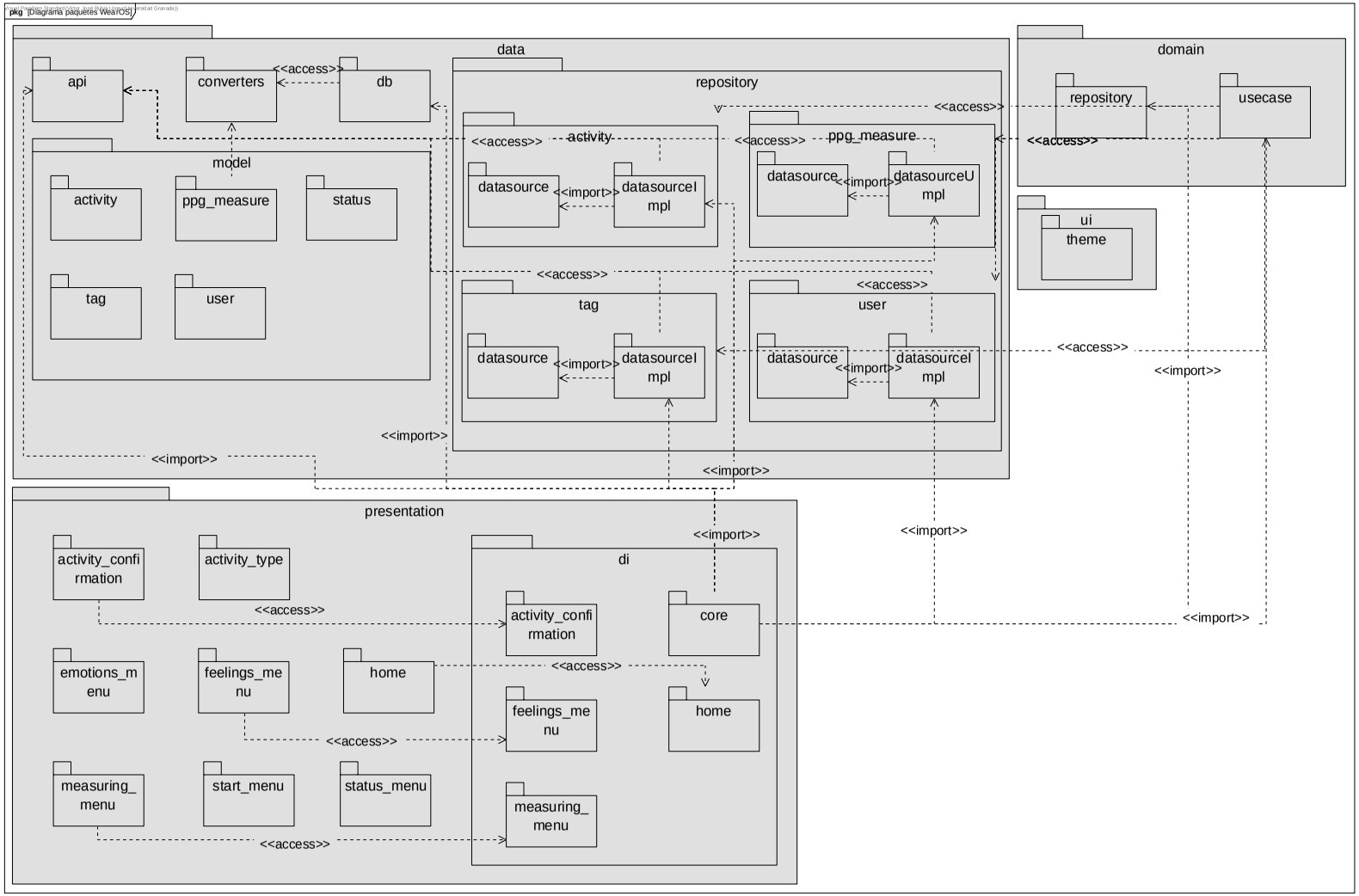


Figura 3.37: Diagrama de paquetes de la aplicación de reloj inteligente

Dicho esto, resulta necesario exponer la forma en la que se implementa la medición de los sensores en el reloj inteligente. Esto se encuentra detallado en el diagrama de flujo expuesto en el apartado anterior en la Figura 3.34. Como se opta por la medición del estrés mediante la Variación de la Frecuencia Cardíaca (VFC), por ser un método bastante extendido, debemos estudiar el cómo obtener de los sensores del reloj inteligente las ondas generadas por la actividad cardíaca.

Para conocer si el reloj inteligente que se proporciona para el proyecto posee sensores de representación de actividad cardíaca, se codifica un script que lista los sensores disponibles. El resultado del mismo se aprecia en la Figura 3.2.

```
Accelerometer /Type_String: android.sensor.accelerometer /Type_number: 1
Gyroscope /Type_String: android.sensor.gyroscope /Type_number: 4
    Name: Magnetometer /Type_String: android.sensor.magnetic_field /Type_number: 2
    Name: Light /Type_String: android.sensor.light /Type_number: 5
Orientation /Type_String: android.sensor.orientation /Type_number: 3
Significant Motion /Type_String: android.sensor.significant_motion /Type_number:
    ↪ 17
    Name: Gravity /Type_String: android.sensor.gravity /Type_number: 9
Linear Acceleration /Type_String: android.sensor.linear_acceleration /Type_number:
    ↪  10
    Name: Rotation Vector /Type_String: android.sensor.rotation_vector /
        ↪ Type_number: 11
Geomagnetic Rotation Vector /Type_String: android.sensor.
    ↪ geomagnetic_rotation_vector /Type_number: 20
    Name: Game Rotation Vector /Type_String: android.sensor.game_rotation_vector /
        ↪ Type_number: 15
    Name: Magnetometer (uncalibrated) /Type_String: android.sensor.
        ↪ magnetic_field_uncalibrated /Type_number: 14
Gyroscope (uncalibrated) /Type_String: android.sensor.gyroscope_uncalibrated /
    ↪ Type_number: 16
Heart Rate PPG /Type_String: android.sensor.heart_rate /Type_number: 21
Tilt Detector /Type_String: android.sensor.wrist_tilt_gesture /Type_number: 26
Activity Recognizer /Type_String: android.sensor.mobvoi_activity_recognizer /
    ↪ Type_number: 69632
    Name: Step Detector /Type_String: android.sensor.step_detector /Type_number:
        ↪ 18
Step Counter /Type_String: android.sensor.step_counter /Type_number: 19
    Name: Activity Data /Type_String: android.sensor.mobvoi_activity_data /
        ↪ Type_number: 69633
    Name: Off Body Detector /Type_String: android.sensor.
        ↪ low_latency_offbody_detect /Type_number: 34
Heart Rate PPG Raw Data /Type_String: com.google.wear.sensor.ppg /Type_number:
    ↪ 65572
```

Listing 3.2: Listado de sensores con sus identificadores del TicWatch Pro

Dentro de los sensores que posee el dispositivo, encontramos el sensor PPG (Ver el último de la figura 3.2), el cual nos permite conocer la frecuencia cardíaca. Es por esto, por lo que aquí se opta por usar este sensor para medir el nivel de estrés en pro de otros. El sensor identificado por *com.google.wear.sensor.ppg* será el que escojamos para obtener los datos en crudo.

Para obtener los datos de este sensor, se debe, en primer lugar solicitar permisos al usuario, explicándole brevemente para qué se necesita acceso. A través de una clase *Gestor de Sensores*, podremos registrar y suprimir las vinculaciones que tengamos a los

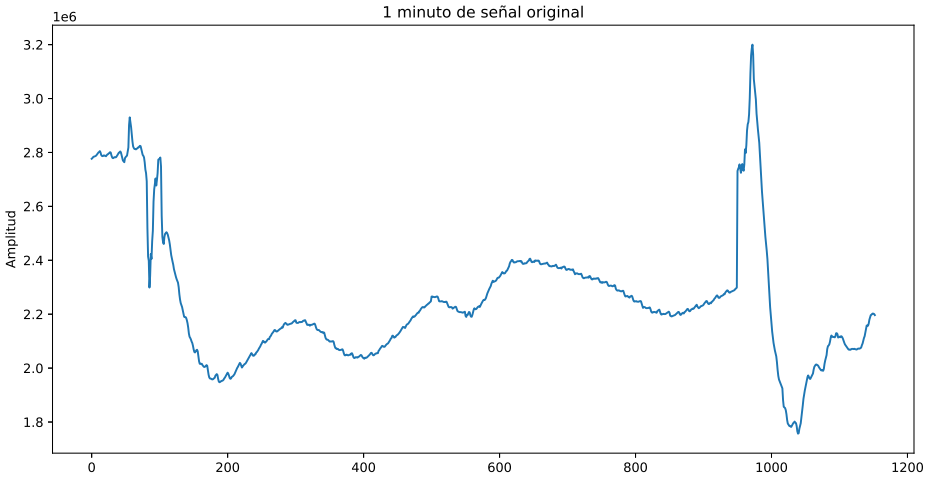
mismos en el transcurso de la aplicación. Para registrar a través de este, al sensor PPG debemos indicarle la frecuencia de muestreo en la que deseamos obtener los datos. En nuestro caso estableceremos que el sensor PPG debe funcionar a una frecuencia de 100 Hz, ya que de acuerdo con la literatura consultada, es una frecuencia aceptable para conocer las variaciones en la frecuencia cardíaca.

Otros sensores que se registran para complementar la medición de estrés son los que se listan en la siguiente lista. Estos se tienen para estudiar la posible influencia sobre el estrés en la realización de actividades, de cara a mejorar las predicciones de estrés.

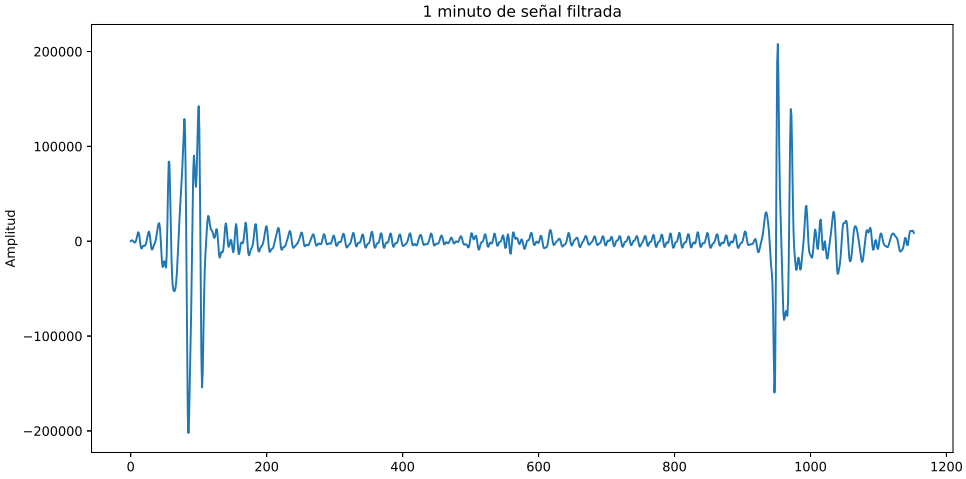
Sensor	**Frecuencia de muestreo / Intervalo**
PPG	100 Hz
Acelerómetro	~32 Hz
Movimiento significativo	Evento basado en activación
Contador de pasos	0.033 Hz
GPS	Intervalos de 5,000 ms (mínimo) a 10,000 ms (por defecto)

Cuadro 3.7: Sensores y sus frecuencias de muestreo

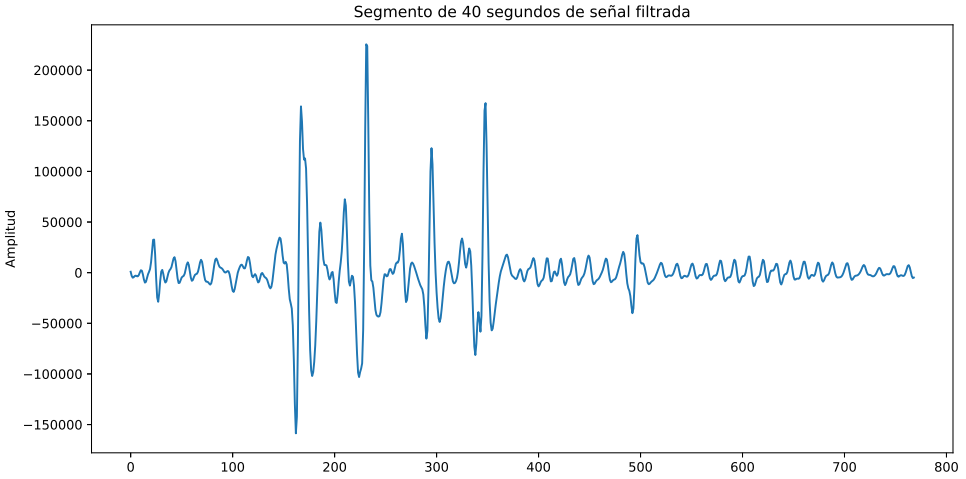
Tras el análisis de la señal PPG que se obtiene en crudo, como se muestra en la Figura 3.38a, se observa que es bastante irregular, de forma que se constata que el sensor, al estar ubicado en un reloj es susceptible a ruidos y será necesario implementar un proceso de limpieza a posteriori, como se muestra en la Figura 3.38c para obtener las características de esta señal.



(a) Un minuto de medición en crudo.



(b) Un minuto de medición con el filtro aplicado.



(c) 40 segundos de medición con el filtro aplicado.

Figura 3.38: Filtrado de paso de banda mediante Butterworth de la señal PPG

Otro aspecto que se implementa en la aplicación es el de hacer la medición de los sensores un servicio en primer plano. Esto, que se muestra de forma visual en las figuras 3.39a y 3.39b nos permite que la aplicación, cuando empiece a medir una actividad, el proceso se ejecute no desde la pantalla desde donde se pueden registrar etiquetas y parar la medición, sino que se ejecuta el proceso desde un servicio.

(a) Página de registro

(b) Página de identificación

Figura 3.39: Ejecución del servicio de la aplicación del reloj que realiza la medición de los sensores

Este servicio estará en constante ejecución hasta que sea notificado de que debe finalizar. De este modo, podemos evitar que cuando el usuario deje de utilizar el reloj, al entrar en segundo plano, pase la aplicación a un modo de "hibernación", cuya misión es preservar la batería del dispositivo y minimizar por ello los recursos.

Esto hace que continúe la medición de los sensores y que se pueda, en cualquier momento, retomar la aplicación para registrar etiquetas, o en otro caso parar la medición de la actividad. La forma de hacer esto pasa por desplegar el apartado de notificaciones del reloj y se mostrará allí una que no puede ser descartada y que presenta dos posibles acciones con botones tal y como se muestra en la Figura 3.39.

3.6. Predicción de estrés

3.6.1. Introducción

Para concluir con la propuesta, se detallará el módulo de predicción de estrés, sobre el cual se apoya la aplicación web para mostrar los niveles de estrés sobre una actividad realizada. Previamente, en el Trabajo Fin de Grado, esta funcionalidad se integra en la propia aplicación web, que es responsable de desencadenar un proceso que retornaba esta información.

El propósito de esta sección es justificar la necesidad de separar en un módulo fuera de la aplicación web, detallar la forma en la que se han realizado los modelos de Aprendizaje Automático que se han desarrollado, y exponer la forma en la que se ha implementado una solución basada en funciones Cloud para desacoplar la lógica de predicción de estrés de la plataforma web.

3.6.2. Motivación y justificación para el desacoplamiento de la predicción

Para justificar la decisión de desacoplar la predicción del nivel de estrés, resulta necesario exponer una serie de motivaciones que se tienen al haber estudiado las debilidades existentes en lo implementado previamente en el Trabajo Fin de Grado.

En el proceso de implementación de la aplicación web, se identificaron algunas limitaciones que hacen necesario buscar soluciones arquitectónicas más sólidas.

La primera restricción surge de la detección del bloqueo del proceso del servidor web. El problema se da a raíz de la invocación de la predicción del estrés, que desencadena la ejecución de un script de Python para obtener las mediciones de cierta actividad, a través de la carga de un modelo preentrenado, y realizando ciertas tareas de limpieza y preprocesamiento sobre dichos datos. Con esto, se obtienen los niveles de estrés en ventanas de tiempo preestablecidas. Dicho script, recibe como parámetro el ID de la actividad, con la cual obtenía a través de la API que ofrece el sistema web, las mediciones del sensor y estas son procesadas. Sin embargo, como el script de Python se ejecutaba con una llamada síncrona al sistema cuando el usuario marcaba la finalización de su actividad en curso, esto se traducía en que la aplicación web, con el *Framework* de *Ruby On Rails*, se quedase bloqueada mientras esperaba recibir el resultado de la ejecución de dicho script. Esto repercute tanto en el hecho de servir el contenido al resto de clientes como en el tiempo que se demora en guardar los datos en su modelo y en la renderización de las vistas. Como consecuencia, el rendimiento global del sistema se veía afectado, provocando latencias y bloqueos puntuales.

La segunda limitación se relaciona con la necesidad de que la predicción del nivel de estrés pueda escalar y, a la vez, mantenerse desacoplada de la aplicación web. Resulta esencial contar con un mecanismo que permita ejecutar concurrente y asíncronamente múltiples instancias del proceso de predicción, para evitar así el bloqueo del resto de procesos y mejorar la capacidad de respuesta ante un posible aumento en la demanda o tamaño de procesamiento.

Con esto, se hace evidente la necesidad de estandarizar y normalizar la forma en la que se realiza la predicción del estrés. Para ello, se opta por realizar una función que reciba las propias mediciones en sí, en lugar de un ID de aplicación a través del cual solicitar a la API de la aplicación web los datos. De esta forma, además, podemos aislar el proceso de predicción de estrés del procedimiento que se ha de seguir para descargar los datos de cierta actividad haciendo necesario tener conexión directa con

la aplicación web, resultando en una función atómica que recibe simplemente datos y, tras un procesamiento, devuelve un resultado. Habiendo expuesto esto, la función debe preprocesar y predecir, en base a cierto modelo, los niveles de estrés. Además, gracias a esta abstracción, sería factible reemplazar con facilidad el modelo subyacente o incluso parametrizarlo, facilitando la evaluación comparativa de diferentes enfoques sobre el mismo conjunto de datos.

Estas necesidades convergen en la propuesta de diseñar una arquitectura basada en Funciones como Servicio (FaaS) desplegable en un entorno Cloud. Las ventajas de optar por esta vía se potencian gracias al empaquetado de la función propuesta en un contenedor y a la orquestación de los mismos mediante Kubernetes y un sistema como OpenFaaS. OpenFaas permite ofrecer una función mediante llamadas API, controlando eficientemente el balanceo de carga del sistema y permitiendo escalar los recursos dinámicamente. Este planteamiento no solo simplifica el despliegue y la administración de diferentes modelos de estrés basados en diferentes funciones, sino que también permite incrementar la robustez y rendimiento del sistema en su conjunto, repercutiendo directamente en la experiencia de los usuarios.

3.6.3. Creación del recurso Cloud e implementación de OpenFaaS

Para comenzar con la implementación de lo necesario para llevar a cabo el desacople de la predicción de estrés, se expone lo realizado para crear en un servicio Cloud los recursos necesarios para desplegar esta arquitectura.

Optaremos por usar el proveedor de Microsoft Azure, por tener un convenio con la Universidad de Granada, que ofrece una suscripción para estudiantes. Esta suscripción incluye un crédito inicial para usarse en el catálogo de servicios y, además, ofrece ciertos servicios populares de forma gratuita bajo limitaciones. Todo esto se ofrece, además, sin necesidad de introducir una tarjeta de crédito. Entre los servicios gratuitos se encuentra el de las máquinas virtuales con ciertas configuraciones.

Para llevar a cabo nuestro proceso nos valdremos de una con la configuración B2ms (2 vCPU, 8 GB de memoria), ya que esta configuración proporciona un equilibrio adecuado entre capacidad de procesamiento y memoria, siendo suficiente para ejecutar tanto OpenFaaS como los contenedores necesarios para procesar las funciones de predicción de estrés e incluso desplegar aquí la aplicación web.

Durante la creación de la máquina virtual, se seleccionará un sistema operativo basado en Linux, como Ubuntu Server 22.04 LTS, por ser compatible con las herramientas necesarias y ofrecer estabilidad y seguridad para el despliegue. Una vez creada, se procederá a configurar el acceso SSH para una administración segura y a instalar los paquetes necesarios para el funcionamiento de Kubernetes y OpenFaaS, siguiendo las mejores prácticas de instalación y asegurando que los puertos necesarios, como el 8080 para el Gateway, estén debidamente habilitados. De igual forma, se abre el puerto 3000 para permitir el acceso al sistema web. Esta máquina tiene asociada una IP fija que utilizare-

mos para acceder a la misma, tanto para controlarla por SSH, como para acceder a los recursos en red que se exponen.

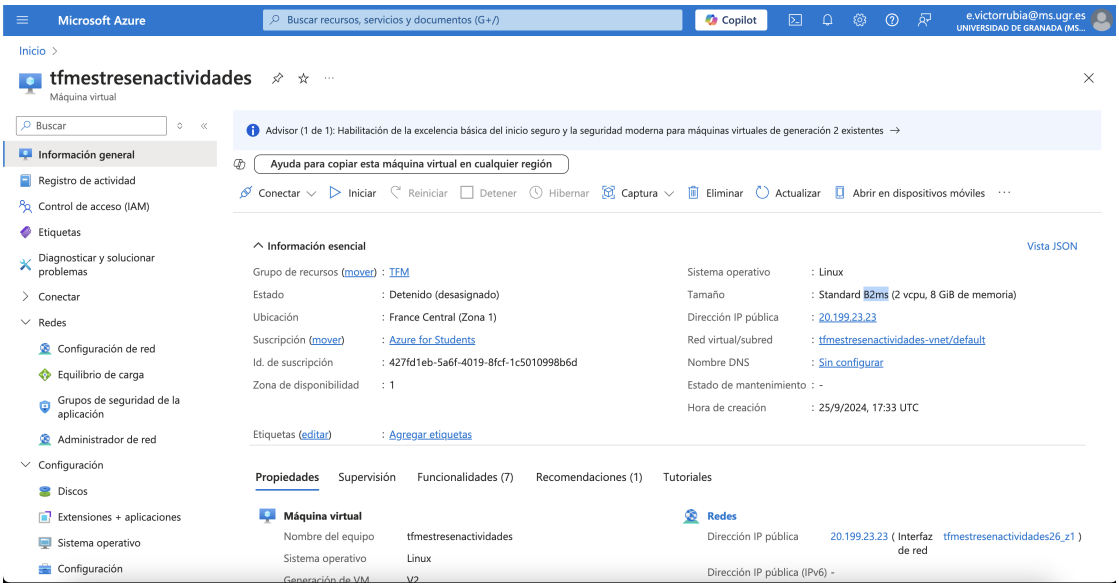


Figura 3.40: Pantalla del recurso de Máquina Virtual creado para nuestro sistema web en Azure

Una vez creada, debemos iniciar la instancia en la pantalla del recurso tal y como se muestra en la Figura 3.40. Hecho esto, accederemos a ella a través de SSH e instalaremos todas las dependencias necesarias como Docker y MicroK8S, que es una implementación sencilla de Kubernetes a fin de poder desplegar OpenFaaS sobre ella sin tener que realizar numerosas configuraciones.

Una vez instalado todo esto, tenemos todo lo necesario para continuar con la creación de nuestras funciones como servicio para la predicción de estrés.

Creación de un conjunto de medición de niveles de estrés propio

Tal y como se revisa en la Sección 2.2.4, se decide abordar la creación de un modelo de estrés propio a través de los datos recogidos en la experimentación que se hizo con el DASS-21.

Este modelo nos serviría para compararlo con el que ya se usaba en la versión del sistema creado para el Trabajo Fin de Grado. El modelo que se usó fue resultado del entrenamiento de un clasificador de *Random Forest* con el conjunto de datos SWELL, el cual se ajustaba muy bien a la predicción de estrés en este tipo de entornos a los que sometieron los sujetos.

Por ello, se hace un trabajo de recopilar los datos de los 39 sujetos para los que se tienen medición y, de ellos, extraemos la información del sensor que nos aporta in-

formación sobre la Variabilidad de la Frecuencia Cardíaca (VFC), el PPG. Como en el experimento se tenían para medir dos relojes idénticos al que estamos usando en este Trabajo Fin de Máster, extraeremos la información de ambos y la usaremos para hacer nuestro conjunto de datos.

Para formar el conjunto de datos, por un lado se tiene la medición del sensor PPG en función del tiempo, acotándolo en el rango en el que tenían que responder al cuestionario del DASS-21 y algunas preguntas de carácter médico. Una vez tenemos la medición del sensor PPG para este periodo, entonces se decide representar la señal tal y como vemos en la Figura 3.41, donde se aprecian ambas señales superpuestas y mostrando para cada reloj ambas señales PPG, pues el sensor que este posee tiene dos canales de señal, que se pueden interpolar para obtener un mejor rendimiento.

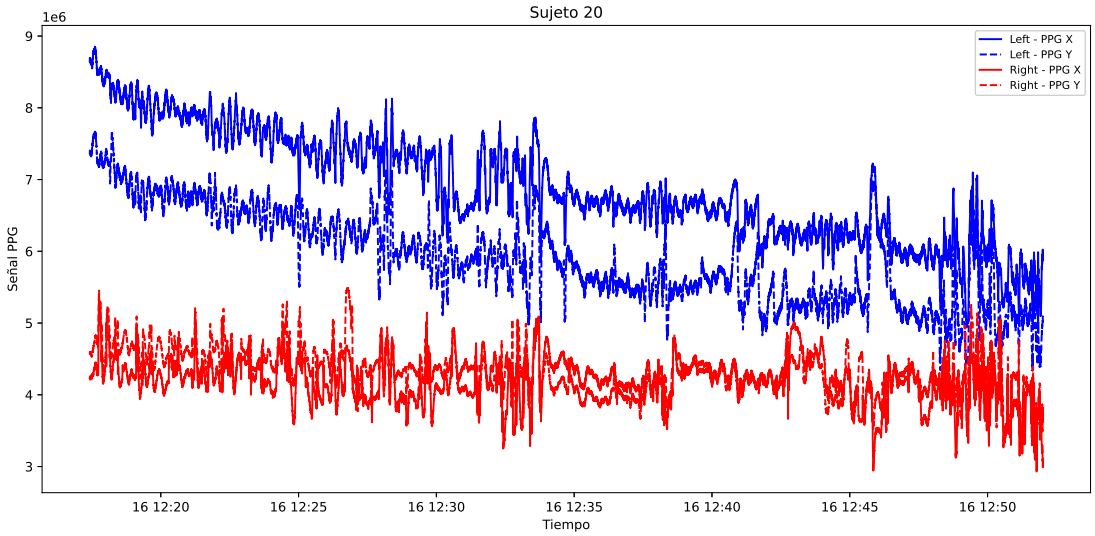


Figura 3.41: Señal PPG original.

Atendiendo a la señal de la figura y estudiando cómo podrían extraerse características de la VFC, se estudia la posibilidad de un análisis de mejora de señal puesto que, al final, debemos quedarnos con una señal única. Se estudia que, si falta algún trozo de medición en alguno de los canales, se pueda suplir con otro que si que tenga medición para ese espacio de tiempo, o, por ejemplo, si algún fragmento de la medición presenta mucho ruido, poder interpolar dicha medición con el resto de canales a fin de poder limpiarla. Esto es así, debido a que durante la experimentación, el sujeto puede tener ciertos movimientos que hagan que se empobrezca la calidad de la señal en alguno de los relojes y/o canales de los sensores. Sin embargo, como el estudio de teorías y aplicaciones de técnicas de pre-procesamiento de señales no es un objetivo del proyecto, unido a la falta de tiempo, se propondrá como trabajo futuro, ya que es un proceso que es más complicado de automatizar, por ejemplo por tener que establecer ciertas reglas para determinar en qué puntos la calidad de una señal es mejor que la otra, entre otras.

Por ello, se decide continuar como si fuesen señales independientes y válidas en toda su extensión de tiempo. Lo siguiente que se realiza es un filtrado paso bajo utilizando un filtro de Butterworth, como se muestra en la Figura 3.42. Este tipo de filtro está muy extendido en cuanto al procesamiento de señales por su capacidad de eliminar frecuencias altas no deseadas, preservando frecuencias bajas relevantes, como las asociadas a la actividad cardíaca en señales PPG. Como parámetros del filtro, utilizaremos una frecuencia de corte de 4.0 Hz. Esta frecuencia se ha seleccionado debido a que las señales cardíacas relevantes se encuentran por debajo de este umbral, lo que permite mantener dichas señales mientras se atenúan las frecuencias superiores. Por otro lado, la frecuencia de muestreo se estableció en torno a los 20 Hz, valor que resulta adecuado para capturar correctamente las variaciones en las señales cardíacas durante las tomas realizadas.

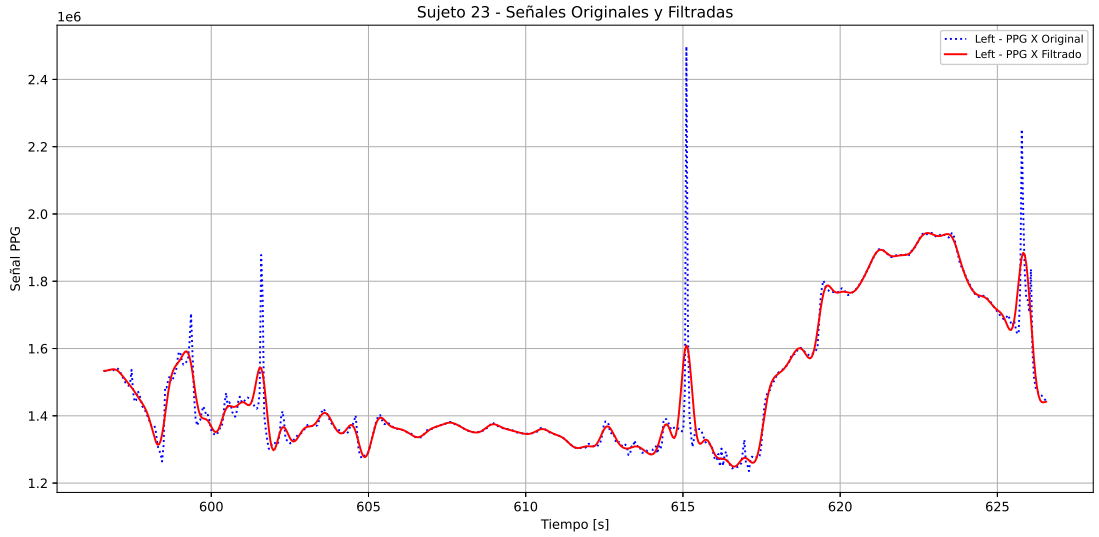


Figura 3.42: Señal PPG original superpuesta con la filtrada.

Tras este estudio, pasamos al uso de la librería HeartPy, la cual es ampliamente utilizada en el análisis de datos fisiológicos. Esta nos permite trabajar con señales PPG para extraer información directamente relacionada con la VFC. Además, cuenta con métodos para aplicar preprocesamiento de señales y filtros de paso banda como los que se estudiaron anteriormente, lo cual facilita mucho el obtener las características de la VFC. En la Figura 3.42 se muestra el resultado tras aplicar las funciones de preprocesamiento y filtrado de paso banda Butterworth, junto con un remuestreo de la señal. Las frecuencias en este caso para detectar el ritmo cardíaco se establecen entre 0.75 Hz, que son unos 45 latidos por minuto en períodos de reposo, y 3.5 Hz, que son unos 210 latidos por minuto en períodos de alta actividad física.

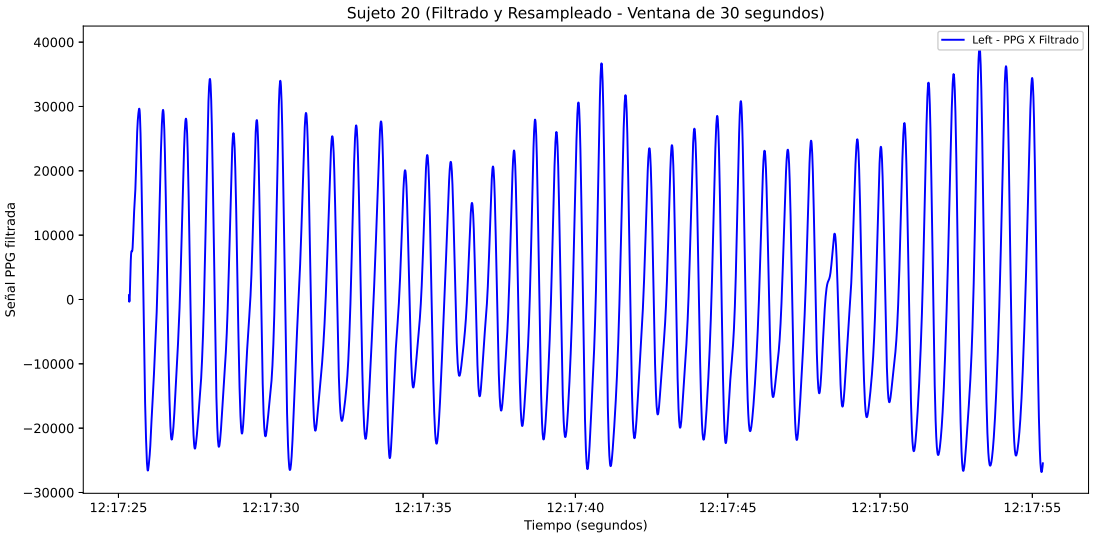


Figura 3.43: Señal PPG filtrada y re-muestreada.

Una vez hecho esto sobre nuestro conjunto de datos, ya tendremos una señal PPG preprocesada, filtrada y remuestreada a 100 Hz, donde podemos observar mejor las variaciones, como se muestra en la figura 3.43. Para completar este conjunto de datos, ahora tenemos, para cada registro, el sujeto al que pertenece la medición y la mano con la que ha sido registrada. Esto, sin embargo, no es suficiente para poder hacer un modelo de predicción, ya que en realidad lo que queremos es obtener las características de la VFC que se observa a raíz de esta señal PPG.

Para esto nos valdremos nuevamente de la librería HeartPy, que tras hacer todo el preprocesamiento, nos permite en ventanas de tiempo, de por ejemplo 30 segundos, obtener características como las mostradas en la Tabla 2.1 a través del análisis de esta ventana que hace el módulo que podemos ver en las Figuras 3.44, 3.45 y 3.46. Por último, nos remitimos a los tests DASS-21 que se pasaron a los sujetos para asociar a las mediciones una etiqueta para lo que mide, siendo esto el nivel de estrés, de depresión y de ansiedad. A cada una de las mediciones entonces, asociamos estos valores, para tener un conjunto resultante cuya variable objetivo para nuestra predicción de estrés será el nivel de estrés obtenido (bajo, medio o alto).

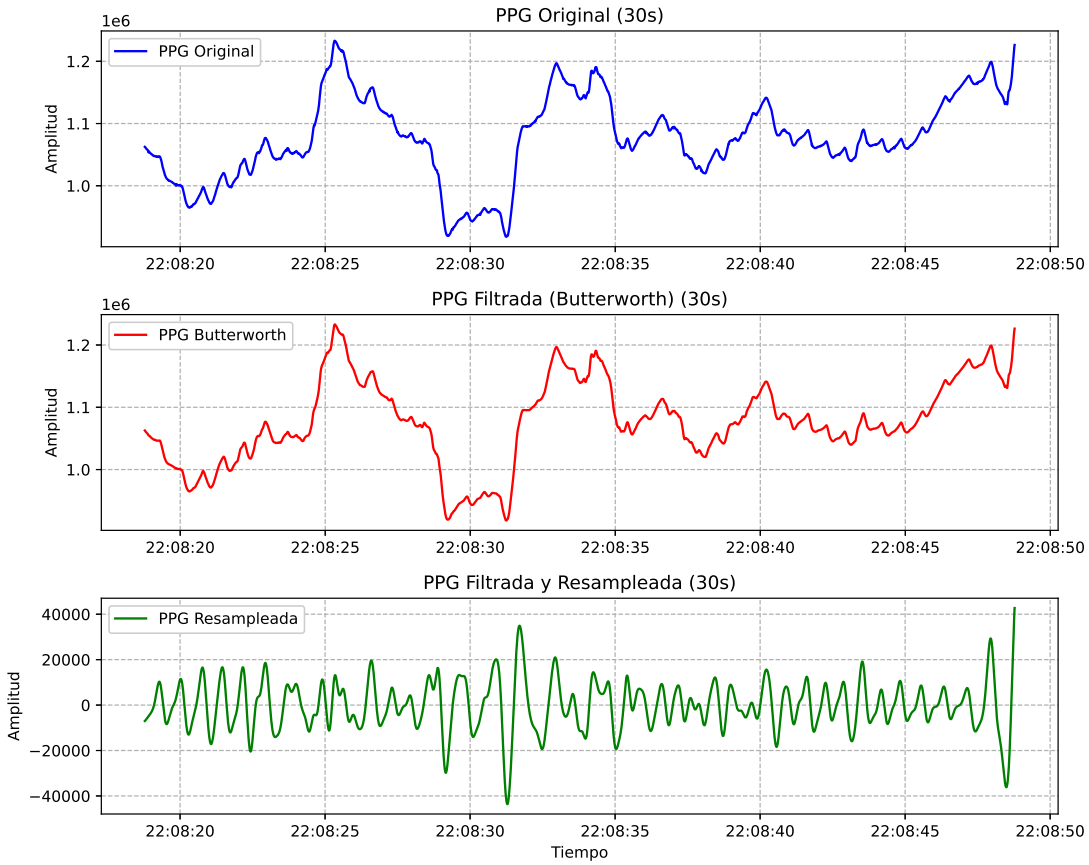


Figura 3.44: Ventana de 30 segundos de medición PPG: señal original, filtrada y remuestreada.

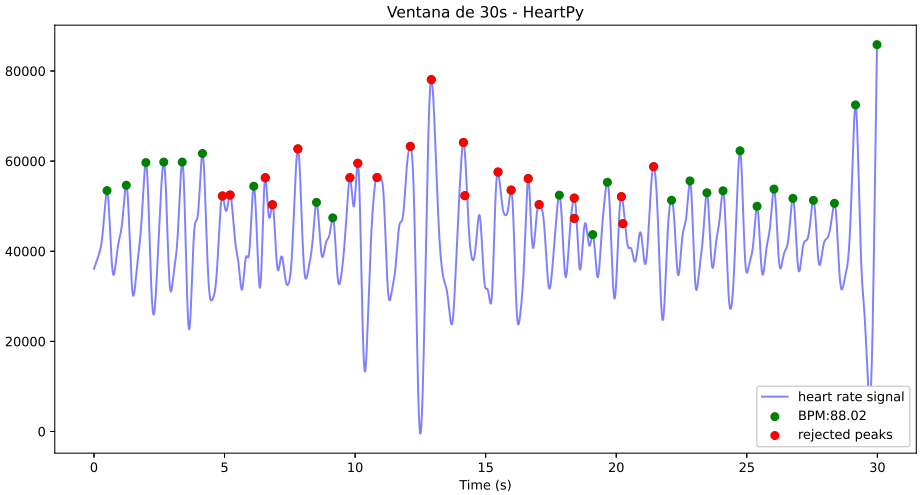


Figura 3.45: Ventana de medición de 30 segundos PPG filtrada y re-muestreada procesada para obtener características de VFC.

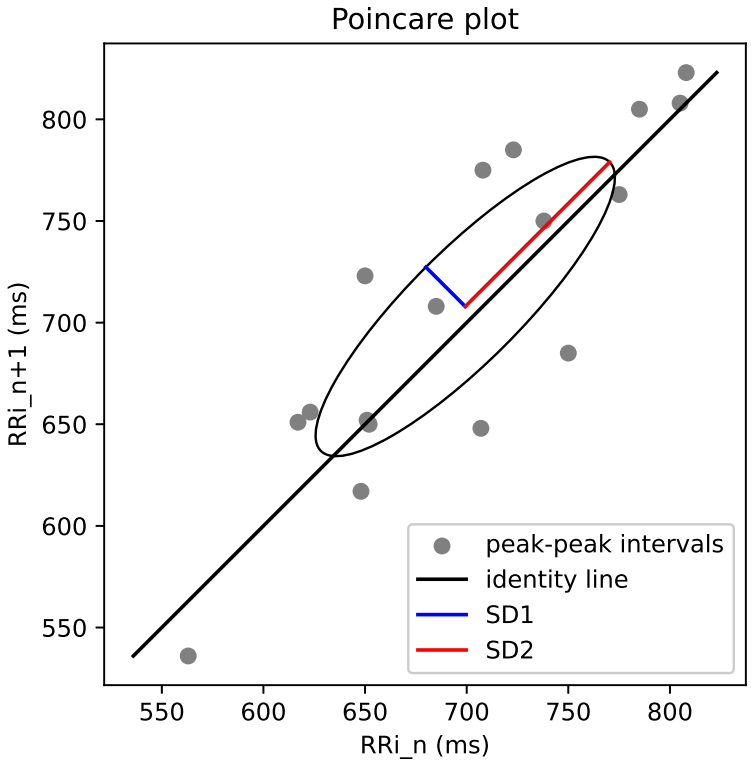


Figura 3.46: Diagrama Poincaré de la señal PPG. Se muestran los intervalos RR consecutivos con la línea de identidad y la elipse que representa la dispersión de los puntos.

Parámetro	**Valor**	**Parámetro**	**Valor**
MEAN_REL_RR	-0.020070	**MEDIAN_REL_RR**	-0.000769
SDRR_REL_RR	0.829865	**RMSSD_REL_RR**	0.231740
SDRR_RMSSD_REL_RR	3.581017	**Mo**	805.000000
VR	1273.000000	**AMo**	4.545455
SI	0.000222	**BPM**	88.015817
IBI	681.695652	**SDNN**	94.531998
SDSD	23.849441	**RMSSD**	39.674410
pNN20	0.588235	**pNN50**	0.294118
HR_MAD	65.000000	**SD1**	27.379620
SD2	100.471398	**S**	8642.108898
SD1/SD2	0.272512	**breathingrate**	0.319040

Cuadro 3.8: Características de la VFC obtenidas de la ventana de 30 segundos de medición.

Finalmente, tendremos un conjunto de datos con la distribución de la clase de estrés que se muestra en la Tabla 3.9, teniendo un total de 7996 muestras para nuestro conjunto de datos.

Categoría	Cantidad de Muestras
Alto estrés	5349
Medio estrés	1598
Bajo estrés	1049

Cuadro 3.9: Distribución de las muestras por nivel de estrés en nuestro conjunto basado en el DASS-21.

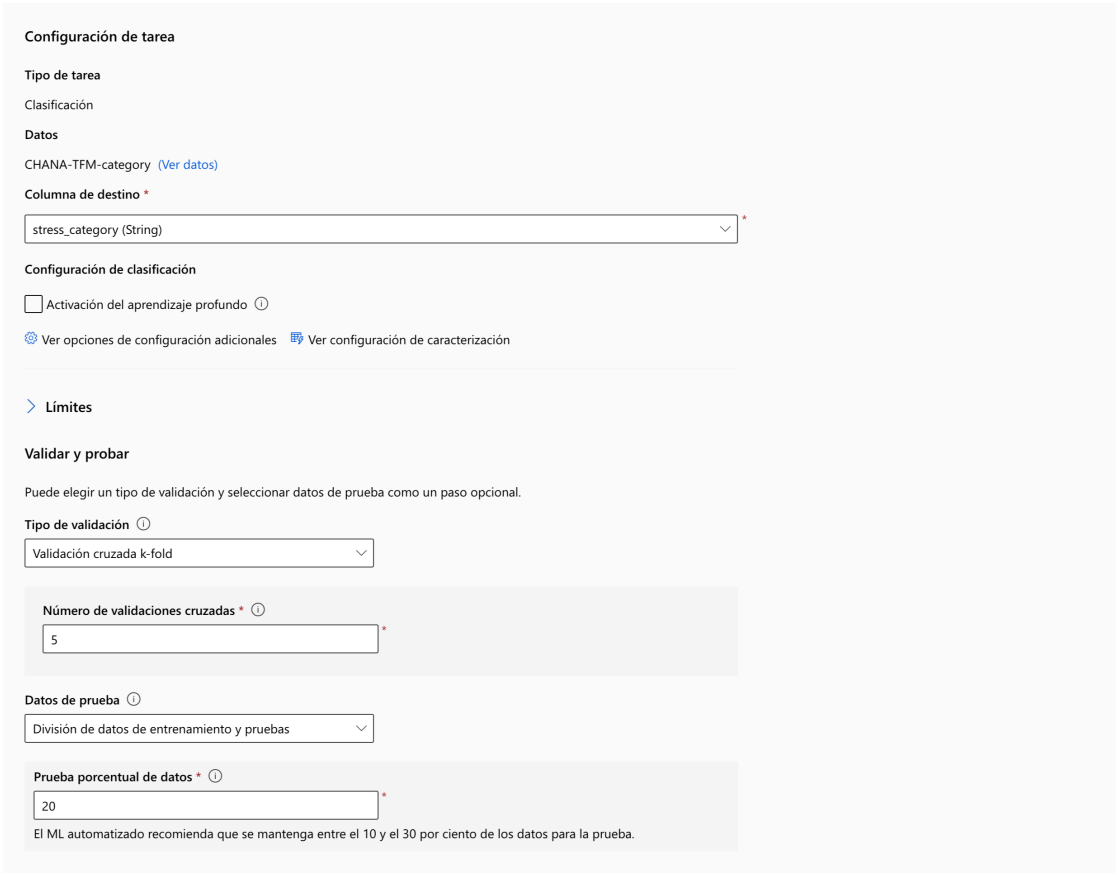
Debido a la falta de tiempo para desarrollar un modelo óptimo, se ha decidido utilizar uno de los servicios ofrecidos por Azure, Azure Machine Learning, como punto de partida. Azure Machine Learning es un portal que permite cargar conjuntos de datos en la nube y, mediante algunos ajustes, ejecutar múltiples combinaciones de clasificadores e hiperparámetros. Esto facilita identificar la mejor configuración según la métrica seleccionada, acelerando significativamente y mejorando la fiabilidad del proceso de desarrollo.

Se describirá a continuación el proceso realizado para llevar a cabo un modelo base a partir del conjunto de datos creado de nuestro experimento donde pasamos el test DASS-21 a los sujetos.

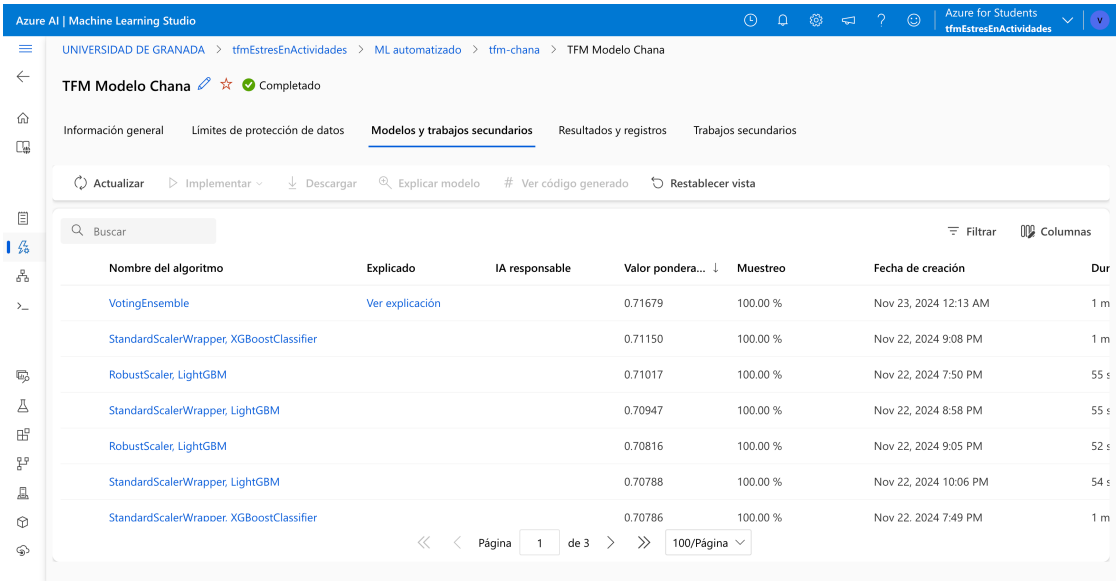
Creación del modelo propio de predicción de nivel de estrés

Para comenzar, crearemos un recurso en Azure que nos permita almacenar archivos de gran tamaño. Para ello existen Cuentas de Almacenamiento, que también pueden usarse gracias a la suscripción de estudiante que nos ofrece la Universidad. En una de estas cuentas depositaremos en formato CSV nuestro conjunto de datos.

Una vez hecho esto, nos crearemos un recurso de Machine Learning, el cual nos permitirá acceder al *Machine Learning Studio* de *Azure AI*. Desde este panel se nos permite el uso del AutoML, que es un conjunto de herramientas que permiten automatizar el proceso de crear, entrenar y optimizar modelos de aprendizaje automático. Esto se combina con MLFlow, que es una plataforma que ayuda a gestionar diferentes modelos que se creen a modo de llevar un control de versiones y poder comparar entre ellos de forma sencilla.



(a) Configuración para realizar los entrenamientos



(b) Modelos entrenados ordenados por métrica de área bajo la curva ROC.

Figura 3.47: Interfaz de Azure Machine Learning para realizar AutoML

Todo esto confluye en esta herramienta de Azure, donde a partir de seleccionar nuestro conjunto de datos, se deciden qué columnas nos son interesantes y se quitan aquellas que no deben formar parte del conjunto final, como el identificador del sujeto. De igual forma, se establece cuál será la variable objetivo.

Además, se puede definir qué clasificadores queremos que se prueben, qué tipo de división se quiere realizar para el conjunto de entrenamiento y pruebas, y si se quiere realizar algún tipo de validación, como por ejemplo una validación cruzada.

Asimismo, se define automáticamente un detector de tipo de dato para cada característica y, para aquellos valores nulos, una técnica para completarlos (imputación de valores[3]), en base a los datos que contiene. Por último, aquí debemos ajustar los parámetros específicos, como el número de nodos y el tamaño de las instancias en los clústeres Cloud donde se quiere ejecutar el experimento, de forma que se pueda hacer incluso paralelamente. Esto se muestra en la Figura 3.47a.

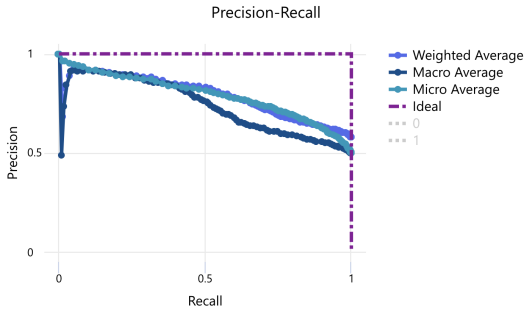
Tras la ejecución de todos los modelos, basados en combinaciones de flujos de preprocesamiento y clasificadores, se obtiene una lista ordenada por mayor métrica escogida. En nuestro caso estamos comprobando el área bajo la curva ROC, para poder capturar el mayor número de casos, en lugar de la precisión, y comparando entre los dos primeros nos quedaremos con el de preprocesado StandardScalerWrapper y clasificador XGBoostClassifier, por tener una mejor métrica de *accuracy* y muy similar área bajo la curva ROC que la primera opción del clasificador VotingEnsemble. El resultado de la ejecución de todos los modelos puede verse en la Figura 3.47b y las métricas obtenidas para este modelo se muestran en el Cuadro 3.10, 3.48a, 3.48b y 3.48c.

Métrica	**Valor**
accuracy	0.7309256
AUC_macro	0.7115031
AUC_micro	0.8007922
AUC_weighted	0.7115031
average_precision_score_macro	0.6923852
average_precision_score_micro	0.7938886
average_precision_score_weighted	0.7535674
balanced_accuracy	0.5877381
f1_score_macro	0.5851129
f1_score_micro	0.7309256
f1_score_weighted	0.6837481
log_loss	0.5434687
matthews_correlation	0.2594351
norm_macro_recall	0.1754762

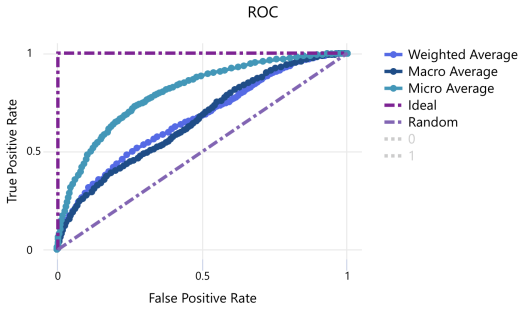
[3]En el marco de la ciencia de datos, la imputación de valores se refiere al proceso de asignar valores estimados a datos faltantes o incompletos dentro de un conjunto de datos. Esto se realiza mediante técnicas estadísticas, modelos predictivos o reglas predefinidas, con el objetivo de preservar la integridad del análisis y minimizar el sesgo causado por la ausencia de datos.

Métrica	Valor
precision_score_macro	0.6918925
precision_score_micro	0.7309256
precision_score_weighted	0.7118726
recall_score_macro	0.5877381

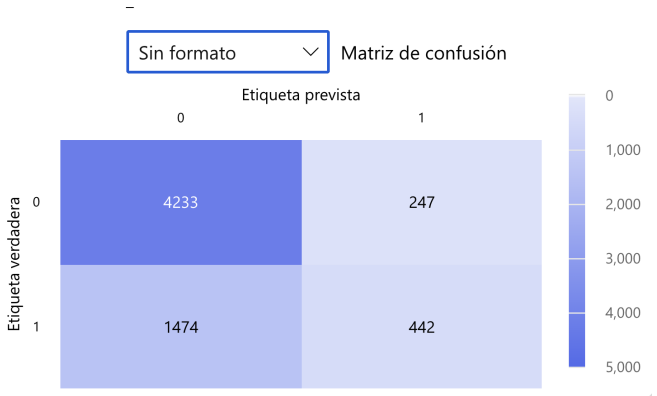
Cuadro 3.10: Resultados de métricas del modelo



(a) Precision en función del recall de nuestro modelo de DASS-21.



(b) Curva ROC tras el entrenamiento del mejor modelo para nuestro conjunto de experimentación DASS-21.



(c) Matriz de confusión resultante del entrenamiento del modelo basado en el conjunto del DASS-21.

Figura 3.48: Gráficos resultantes del entrenamiento del modelo basado en el conjunto de experimentación DASS-21.

Teniendo este proceso que de forma distribuida y rápida nos permite asentar una base sobre la que comenzar a construir nuestro modelo, obteniendo los hiperparámetros del mejor de los resultados.

A partir de estos, se replica la forma en la que se realiza el entrenamiento del modelo en local, aplicando técnicas de preprocesamiento no disponibles en el entorno de Azure, como SMOTE o ADASYN para mitigar el desbalanceo. De esta forma, podemos construir

nuestro modelo de predicción de niveles de estrés, aún mejor que el que se realiza en Azure y en un formato exportable para crear posteriormente nuestra función.

Cabe destacar que también se prueban varias técnicas de Deep Learning, utilizando TensorFlow y Keras, para este conjunto de datos. El mejor modelo obtenido fue diseñado con las siguientes capas:

- Una capa inicial de entrada
- Tres capas densas intermedias con 256, 128 y 64 unidades, respectivamente, cada una con activación ReLU.
- Capas de normalización por lotes después de cada capa densa para estabilizar y acelerar el entrenamiento.
- Capas de Dropout con una tasa del 30 % para mitigar el sobreajuste.
- Una capa de salida con activación softmax para predecir las probabilidades de las diferentes clases.

El entrenamiento se realiza con una función de pérdida de entropía cruzada categórica para etiquetas codificadas de manera ordinal y el optimizador RMSProp con una tasa de aprendizaje inicial de 0.0005. Se emplearon técnicas de balanceo de clases como ADASYN y el cálculo de pesos de clase para manejar el desbalanceo en las etiquetas. Además, se implementaron técnicas de aumento de datos añadiendo ruido gaussiano. El modelo se entrena durante un máximo de 100 épocas con callbacks para la detención temprana y la reducción de la tasa de aprendizaje, utilizando un conjunto de validación para controlar el sobreajuste. Sin embargo, tras mucho esfuerzo por buscar la mejor combinación de capas y técnicas, los resultados que se obtienen son un F1-score de 71.22 % y un accuracy de 72.16 %, valores que, aunque no son malos, sugieren que es necesario buscar alternativas para mejorar el rendimiento. Por lo tanto, se plantea continuar con la aproximación que se comenta en los párrafos anteriores, con un algoritmo de clasificación, como mejora.

Modelo combinado de SWELL y WESAD

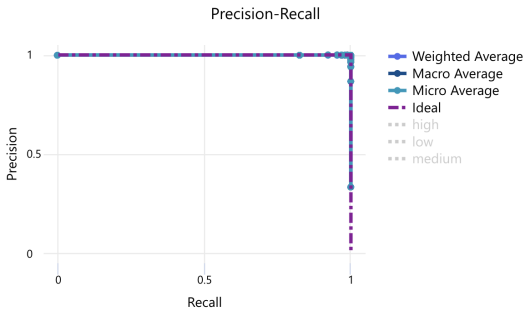
Por otro lado, se construye un modelo basado en la combinación del SWELL y WESAD atendiendo a las lecturas realizadas en el Apartado 2.5.2. Este conjunto de datos se toma del artículo consultado y se realiza un procedimiento similar al punto anterior con la herramienta de Azure Machine Learning. Para el mejor de los modelos obtenidos, que se trata de uno con el clasificador XGBoost y un preprocesamiento con MaxAbsScaler, replicamos su arquitectura en local para poder manejarlo en la posterior función que crearemos.

Esta mejora consiste en combinar el modelo SWELL, utilizado originalmente en el Trabajo Fin de Grado, con uno nuevo, el WESAD. Los artículos consultados, como los

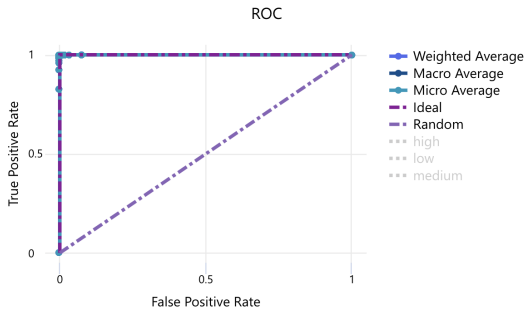
de Dahal (Dahal et al., 2023) y Vos (Vos et al., 2023), demuestran que esta combinación aumenta la precisión en la detección de niveles de estrés. Además, al incorporar una experimentación diferente en la recolección de datos, el modelo se vuelve más versátil y puede adaptarse a distintos escenarios, algo esencial para el proyecto que estamos llevando a cabo. Las métricas obtenidas del entrenamiento, usando diferentes clasificadores y preprocesadores, para este modelo se muestran en las Figuras 3.11, 3.49a, 3.49b y 3.49c.

Métrica	**Valor**
accuracy	0.9998803
AUC_macro	0.9999999
AUC_micro	0.9999999
AUC_weighted	0.9999999
average_precision_score_macro	0.9999999
average_precision_score_micro	0.9999999
average_precision_score_weighted	0.9999999
balanced_accuracy	0.9998803
f1_score_macro	0.9998804
f1_score_micro	0.9998803
f1_score_weighted	0.9998803
log_loss	0.007076714
matthews_correlation	0.9998205
norm_macro_recall	0.9998205
precision_score_macro	0.9998806
precision_score_micro	0.9998803
precision_score_weighted	0.9998803
recall_score_macro	0.9998803

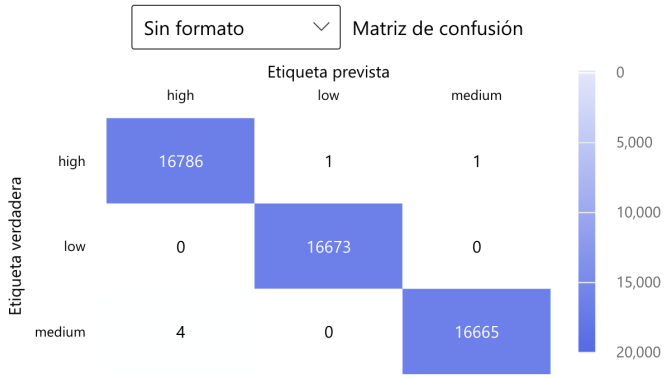
Cuadro 3.11: Resultados de métricas del modelo combinado SWELL y WESAD



(a) Precision en función del recall de nuestro modelo combinado SWELL-WESAD.



(b) Curva ROC tras el entrenamiento del mejor modelo para el conjunto combinado SWELL-WESAD.



(c) Matriz de confusión resultante del entrenamiento del modelo basado en el conjunto combinado SWELL-WESAD.

Figura 3.49: Gráficos resultantes del entrenamiento del modelo basado en el conjunto de combinado SWELL-WESAD.

Comparación con nuestro modelo

Una vez teniendo nuestro modelo creado, junto con el de la combinación de los conjuntos SWELL y WESAD, se procede a comparar sus rendimientos. Esto viene motivado por la idea de que, al combinar diversos conjuntos de datos como SWELL y WESAD, es posible detectar estrés en un mayor rango de situaciones, no solo en condiciones de experimentación con las que se recopilaron originalmente sus datos.

Por ello, se proponen dos ejercicios: en el primero se evalúa el modelo SWELL-WESAD con los datos de nuestro experimento basado en el DASS-21; en el segundo se evalúa nuestro modelo basado en el DASS-21 con datos del conjunto SWELL-WESAD a fin de conocer sus métricas de desempeño.

En el caso del primero de los ejercicios, atendiendo a la Tabla 3.12, podemos com-

probar cómo el modelo combinado SWELL-WESAD, que funciona muy bien a la hora de validar su rendimiento con un subconjunto de sus datos a modo de prueba, cuando lo exponemos a validarse con un conjunto tomado bajo una experimentación diferente, aún tomándose los mismos datos, el rendimiento pasa a ser bastante pobre. Esto refuerza la idea de que se deben combinar varios datasets de medición de estrés para poder captar el nivel de estrés en el mayor número de escenarios posibles.

Métrica	**Valor**
Accuracy	0.3179
Macro Average (Precision)	0.32
Macro Average (Recall)	0.32
Macro Average (F1-Score)	0.31
Support (Macro Avg)	14442
Weighted Average (Precision)	0.32
Weighted Average (Recall)	0.32
Weighted Average (F1-Score)	0.31
Support (Weighted Avg)	14442

Cuadro 3.12: Resultados de las métricas del modelo SWELL-WESAD sobre un conjunto de datos de prueba extraído de nuestro experimento DASS-21.

Por otro lado, en el segundo de los ejercicios, observamos a través de la Tabla 3.13 que, aunque el rendimiento obtenido no es óptimo, el modelo generado a partir del experimento con el DASS-21 muestra una capacidad superior para abstraerse de las características específicas del experimento original. Esto significa que el modelo no depende estrictamente de las condiciones en las que se obtienen los datos, lo que le permite generalizar mejor su comportamiento. En este sentido, afirmamos que nuestro modelo demuestra una mayor flexibilidad y capacidad de adaptación a diferentes situaciones, lo cual es crucial para su aplicabilidad en contextos diversos y acercándose más al objetivo de medir el estrés en la realización de actividades de la vida cotidiana.

Métrica	**Valor**
Accuracy	0.338
Macro Average (Precision)	0.24
Macro Average (Recall)	0.34
Macro Average (F1-Score)	0.18
Support (Macro Avg)	35529
Weighted Average (Precision)	0.24
Weighted Average (Recall)	0.34
Weighted Average (F1-Score)	0.18
Support (Weighted Avg)	35529

Métrica	Valor

Cuadro 3.13: Resultados de las métricas del modelo DASS-21 sobre un conjunto de datos de prueba extraído de la combinación del SWELL-WESAD.

Selección final y conclusiones del proceso de modelado

Con la evidencia presentada en los apartados anteriores, se concluye que, para la tarea de predecir el nivel de estrés asociado a una actividad, el modelo basado en la combinación de los conjuntos de datos SWELL y WESAD será una mejor opción. Sus ventajas se basan en su variedad de situaciones de estrés, por la combinación de las condiciones experimentales bajo las cuales fueron obtenidos sus datos, y en la elevada métrica de exactitud y sensibilidad demostrada en los resultados de validación. Este modelo contiene tres etiquetas para los niveles de estrés: alto, medio y bajo. Las usaremos para dar forma a un gráfico que determine dicho nivel de estrés.

A su vez, el modelo diseñado con el conjunto de datos propio (DASS-21) aporta un alto valor para escenarios más cercanos al ámbito de la experimentación con cuestionarios psicológicos y mediciones puntuales del nivel de estrés que está sufriendo una persona en los últimos días. Sin embargo, su naturaleza hace que se limite, en cierta forma, a las condiciones específicas bajo las cuales se llevó a cabo la recogida de datos, por lo que su capacidad de generalización a otros contextos se ve comprometida. Por esto se decide integrar finalmente el modelo combinado SWELL-WESAD en la plataforma web para la evaluación continua de estrés en base a la medición de los sensores durante la realización de actividades cotidianas por parte de los usuarios.

No obstante, el modelo DASS-21 se empleará de manera complementaria para la elaboración del perfil de estrés de cada persona usuaria de la plataforma, manteniendo un enfoque más binario en cuanto a la presencia o ausencia de estrés, clasificado como 0 para estrés mínimo y 1 para estrés máximo, y obteniendo la probabilidad de que el usuario esté estresado (pertenencia a la clase 1). Dicho enfoque se justifica debido a que el propio cuestionario DASS-21 ofrece una puntuación para el nivel de estrés percibido por la persona, simplificando así su integración en la lógica de la plataforma y ofreciendo información adicional para diagnósticos preliminares o estudios más focalizados en el uso del cuestionario.

Desarrollo de funciones como servicio en OpenFaaS

Para llevar a cabo la separación entre la aplicación web y la lógica de predicción, se desarrollan dos funciones Cloud. Estas se encargan de realizar todo el proceso de inferencia de estrés, llevando a cabo las tareas de preprocesado y filtrado que se definen sobre la señal PPG, y se utilizan ambos modelos que se exponen en los apartados anteriores, el combinado SWELL-WESAD y el modelo propio en base al experimento con el DASS-21. Hacer de esto una función nos da la versatilidad de poder intercambiar el modelo

de predicción con otros conforme se realicen cambios, garantizando la escalabilidad y la mejora continua sin que la aplicación web se vea afectada.

El flujo propuesto para la función de predicción, una vez se tiene un conjunto de mediciones PPG, se resume en los siguientes pasos:

1. **Recepción de datos.** El cuerpo de la petición incluye las mediciones (en bruto) del sensor PPG, junto con los instantes de tiempo de la medición, en vez de un identificador de la actividad. De esta forma la función es llamada directamente con esta información y no tiene que depender de conectarse para recibir los datos a través de la API como se realizaba anteriormente.

2. **Preprocesamiento.** Se aplica un filtrado de Butterworth de paso banda para eliminar altas frecuencias indeseadas, un remuestreo de la señal, a 100 Hz y otras transformaciones necesarias para preparar los datos de entrada.

3. **Extracción de características.** Utilizando la librería HeartPy, se obtiene un conjunto de características relacionadas con la VFC en ventanas de 30 segundos, incluyendo la frecuencia cardíaca, la variación entre latidos (como SDNN o RMSSD) o el índice de pulso (como pNN50 o SD1/SD2).

4. **Predicción de estrés.** El conjunto de características se pasa al clasificador entrenado con el conjunto de datos combinado SWELL-WESAD, devolviendo una etiqueta de estrés (como *alto*, *medio*, *bajo*) para cada ventana de medición, la cual será utilizada por la aplicación web para informar al usuario.

5. **Perfil de estrés según DASS-21.** Como función adicional, se calculará en base a la actividad realizada el perfil de estrés de dicha persona, obteniendo la probabilidad de pertenecer a la clase 1 de nuestro modelo binario basado en el DASS-21, donde 0 es estrés mínimo y 1 es estrés máximo.

6. **Respuesta a la plataforma.** Se formatea la salida en un objeto JSON que devuelve, para cada ventana de tiempo, los valores de estrés estimados por el modelo combinado SWELL-WESAD y el perfil de estrés basado en el modelo DASS-21.

En la Figura 3.3 se muestra el fragmento de código que encapsula el preprocesamiento de la señal fisiológica. Por otro lado, en la Figura 3.4, se muestra cómo los datos preprocesados son utilizados por el modelo para generar las predicciones de estrés, incluyendo el manejo de zonas horarias, procesamiento de fechas y mapeo a etiquetas de estrés. Por último, en la Figura 3.5 se muestra la función núcleo del FaaS, que conecta todo el flujo de trabajo, desde la entrada de datos hasta la salida con las predicciones.

```
1 def filter_signal(signal, sample_rate):
2     filtered_signal = heartpy.filter_signal(signal, [0.7, 3.5], sample_rate=
          ↪ sample_rate, order=3, filtertype='bandpass')
3     return filtered_signal
4
```

```
def extract_features_using_segmentwise(filtered_signal, sample_rate, start_time,
    ↪ segment_duration=60):
wd, measures = heartpy.process_segmentwise(
    filtered_signal,
    sample_rate=sample_rate,
    segment_width=segment_duration,
    segment_overlap=0,
    high_precision=True,
    clean_rr=True
)
```

Listing 3.3: Código para el preprocesamiento de señales fisiológicas

```
def predict_stress(data_frame, model):
    madrid_tz = pytz.timezone('Europe/Madrid')
    data_frame['DATETIME'] = pd.to_datetime(data_frame['DATETIME'], unit='ms',
        ↪ errors='coerce').dt.tz_localize('UTC').dt.tz_convert(madrid_tz)
    results = []
    for idx in range(len(data_frame)):
        fecha = data_frame.iloc[idx]['DATETIME'].strftime("%d/%m/%Y %H:%M")
        prediction = model.predict(data_frame.iloc[idx:idx+1])
        measure_value = {'low': -2, 'medium': 0, 'high': 2}.get(prediction[0],
            ↪ None)
        if measure_value is not None:
            results.append({"date": fecha, "measure": measure_value})
    return results
```

Listing 3.4: Código para el procesamiento de niveles de estrés

```
def handle(data):
    try:
        csv_data = data
        if not csv_data:
            return jsonify({"error": "No CSV data provided"}), 400
        df = pd.read_csv(StringIO(csv_data))
        if df.empty:
            return jsonify({"error": "Empty CSV data provided"}), 400

        timer = df['timestamp'].to_numpy()
        signal = df['ppg'].to_numpy()
        sample_rate = get_sample_rate(timer)
        filtered_signal = filter_signal(signal, sample_rate)
        start_time = timer[0]
        data_features, error_records = extract_features_using_segmentwise(
            ↪ filtered_signal, sample_rate, start_time)
        header = ['DATETIME', 'MEAN_RR', ...]
        data_frame = pd.DataFrame(data_features, columns=header)
        stress_results = predict_stress(data_frame, model)
        all_results = stress_results + error_records
        sorted_results = sorted(all_results, key=lambda x: datetime.strptime(x['
            ↪ date'], '%d/%m/%Y %H:%M'))
        return jsonify(sorted_results)
    except Exception as e:
        return jsonify({"error": str(e)}), 500
```

Listing 3.5: Código núcleo de la función FaaS

Con este código que generamos para gestionar las peticiones a nuestro FaaS, podemos pasar a construir el contenedor que levantará OpenFaaS para cada llamada a nuestra

función. El proceso de construcción y despliegue de la función se estructura en varias fases.

La parte inicial del *Dockerfile* implementa un enfoque de construcción multi-etapa para optimizar el tamaño final de la imagen. Se hereda el binario *fwatchdog* desde la imagen base *of-watchdog*, componente esencial para la ejecución de funciones en OpenFaaS. Posteriormente, se establece un entorno basado en *python:3.12-alpine* que servirá tanto para la instalación de dependencias como para la ejecución del código Python.

La instalación de dependencias abarca dos niveles: las librerías del sistema operativo como *openssl-dev*, *gcc*, *g++* y *musl-dev*, y las dependencias Python especificadas en *requirements.txt*. Este último archivo incluye bibliotecas especializadas para el procesamiento de señales, destacando *heartpy*, junto con herramientas fundamentales para el análisis de datos como *pandas*, *numpy* y *scikit-learn*.

En suma, el *Dockerfile* resultante posibilita la construcción de una imagen ligera y funcional. Al ejecutarse en OpenFaaS, cada instancia de la función, orquestada a través de Kubernetes, o *MicroK8s* en nuestro caso, podrá responder de forma aislada y escalable a las peticiones de predicción de estrés.

```
version: 1.0
provider:
  name: openfaas
  gateway: http://127.0.0.1:8080
functions:
  predict-model:
    lang: python3-flask
    handler: ./predict-model
    image: victorrubia/tfm-predict-model:lates
```

Listing 3.6: Archivo de definición stack.yml que describe la función y sus configuraciones

```
faas-cli build -f stack.yml
faas-cli push -f stack.yml
faas-cli deploy -f stack.yml
```

Listing 3.7: Comando para construir, subir y desplegar la función en OpenFaaS

Tal y como se muestra en el listado de comandos anterior, para desplegar la función se realizan tres pasos principales:

1. `faas-cli build`: Se construye la imagen Docker localmente, basándose en el archivo `stack.yml`, que contiene la configuración de la función.

2. `faas-cli push`: Sube la imagen Docker generada al registro configurado, en nuestro caso Docker Hub.

3. `faas-cli deploy`: Despliega la función en la pasarela (gateway) de OpenFaaS, creando un *deployment* en Kubernetes y haciéndola accesible mediante el endpoint configurado.

Gracias a este enfoque, podemos contar con tantas réplicas de las funciones como se requieran, y OpenFaaS se encargará de balancear las peticiones entrantes entre ellas, además de crear o destruir réplicas según la demanda. De esta forma, se cumple uno de los objetivos de desacoplamiento y escalabilidad que motivaron la creación de las funciones de predicción de estrés en lugar de mantener toda la lógica dentro de la aplicación web.

Invocación y uso de la función FaaS de predicción de estrés

Una vez desplegada la función en OpenFaaS, podemos verificar su disponibilidad a través de la interfaz del Gateway de la Figura 3.50a, donde se listan las funciones, junto con información relativa a su estado y número de réplicas activas. Adicionalmente, desde esta misma consola es posible ejecutar de manera directa una petición, lo que resulta útil para pruebas rápidas sin necesidad de desplegar toda la aplicación web. Los resultados, tras ejecutar una llamada a dicha función con los parámetros de entrada definidos, que son las mediciones PPG y sus marcas temporales, se muestran en la Figura 3.50b.

Para invocar la función desde cualquier cliente, se debe realizar una llamada HTTP tipo `POST` a la URL que expone el Gateway de OpenFaaS, la cual, en un entorno de pruebas local, tendría la forma que se muestra a continuación.

`http://127.0.0.1:8080/function/predict-model`

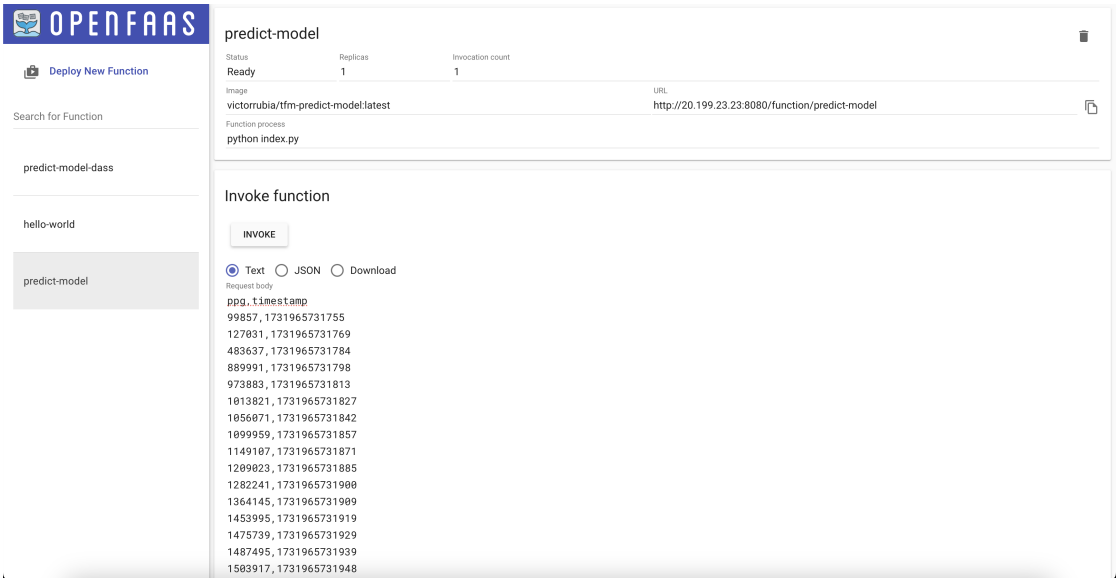
El cuerpo de la petición (*payload*) debe contener las mediciones en bruto (por ejemplo, en formato CSV con el timestamp y la señal PPG) para que la función pueda llevar a cabo su rutina de preprocesado y predicción. Un ejemplo de llamada podría realizarse mediante `cURL`:

```
-H "Content-Type: text/plain"
--data-binary @mediciones_ppg.csv
http://20.199.23.23:8080/function/predict-model
```

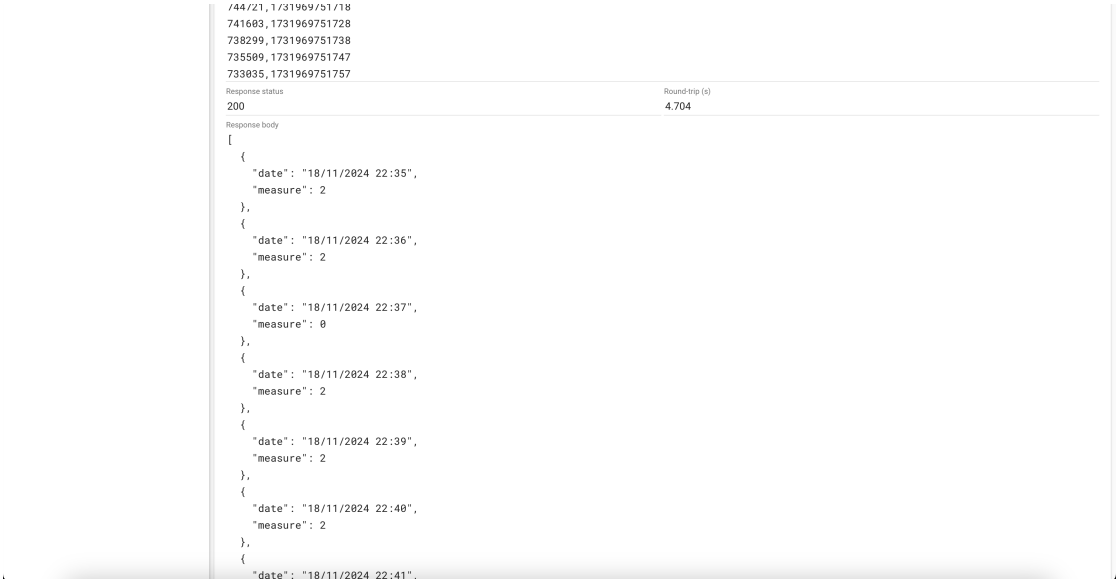
Listing 3.8: Ejemplo de invocación de la función FaaS con cURL

Tal y como se observa en el código del Listing 3.8, el fichero `mediciones_ppg.csv` contiene las lecturas del sensor PPG y la marca de tiempo asociada. La función recibirá este CSV y devolverá un JSON con las predicciones de estrés en ventanas de 30 segundos, junto con su fecha y hora.

Por último, una vez el servidor Ruby on Rails recibe la respuesta en formato JSON, se procesan los campos incluidos en la respuesta y se incorporan al modelo de datos correspondiente, ya sea para representar el estrés relacionado con una actividad específica o para el perfil de estrés de un usuario, según el caso.



(a) Interfaz de usuario de OpenFaas. Se introducen datos de prueba para invocar a la función



(b) Resultado de la llamada de la función.

Figura 3.50: Interfaz de OpenFaaS donde se listan las funciones y permite su gestión

Capítulo 4

Conclusiones y Trabajos Futuros

En este último capítulo, se describe cómo se han cumplido los objetivos de este Trabajo Fin de Máster, y se indican algunas líneas futuras de desarrollo.

4.1. Conclusiones

Para empezar, cabe señalar que este Trabajo Fin de Máster se ha podido realizar gracias a los conocimientos adquiridos en las asignaturas cursadas en el máster, junto con la capacidad de formación autodidacta igualmente obtenida a lo largo del grado y del máster. Gracias a esto, se ha podido seguir una metodología de trabajo eficaz y se ha podido cumplir con las necesidades planteadas para el proyecto por las tutoras.

En esencia, se han podido cumplir todos los objetivos marcados al comienzo del proyecto, mediante el desarrollo de un entorno multisistema para la monitorización de datos fisiológicos, para todo tipo de usuarios y durante el transcurso de actividades diarias, pudiendo ser esto configurable mediante etiquetas personalizadas. Este entorno permite, por un lado, modelar una actividad en cuanto a grado de estrés producido y, por otro lado, conocer los factores implicados. Por otro lado, otorga la posibilidad a los terapeutas de hacer un seguimiento preciso de sus pacientes mediante el registro de ellos en la plataforma y el poder consultar las actividades que realizan, a fin de poder detectar ciertos momentos de dificultad y poder actuar de manera más precisa sobre ellos.

A continuación, se exponen todos los objetivos específicos que se habían planteado y que se han logrado en el desarrollo del presente Trabajo Fin de Máster:

- Para el objetivo de *estudiar el dominio del problema a resolver, el estrés, sus causas, síntomas y mecanismos de medición* (Objetivo 1), se ha profundizado en las Secciones 2.2.1 y 2.2.2, conociendo los factores que pueden producir estrés, en diversas situaciones de la vida cotidiana. Esto incluye actividades relacionadas con la gestión de la independencia, entendiendo los elementos que impactan de manera

particular al grupo de personas mayores.

- Con respecto al objetivo de *analizar cómo la realización de actividades de la vida diaria puede ser estresante y cómo esto puede medirse* (Objetivo 2), en la sección 2.2.3 se han investigado los mecanismos de medición de estrés que existen en la actualidad, y que pueden ser útiles para este proyecto, observando que la mayoría son aún manuales o dependen de auto-informes subjetivos.

- En relación con *desarrollar un método para obtener mediciones fisiológicas de un usuario a través de un reloj inteligente* (Objetivo 3), en las Secciones 3.9 y 3.5.3 se ha desarrollado una metodología para medir el estrés a partir de datos fisiológicos en crudo obtenidos mediante el sensor PPG de un reloj inteligente. Los datos, de los que mediante técnicas de filtrado y limpieza, se pueden extraer las características de la variabilidad de frecuencia cardíaca, nos servirán para procesarlos con los modelos de predicción de estrés.

- Para el objetivo de *realizar un estudio acerca de las metodologías actuales para la limpieza/preprocesado de señales fisiológicas* (Objetivo 4), en la sección 3.6.3 se ha investigado y usado un módulo para la limpieza de la señal obtenida por el sensor PPG del reloj, ya que originalmente se obtiene con bastante ruido. La limpieza está basada en un filtro Butterworth de paso banda, para poder obtener las características necesarias para que los modelos desarrollado sean capaces de predecir el nivel de estrés dado un conjunto de medidas.

- Con relación a *desarrollar un método para calcular el nivel de estrés de un usuario a partir de sus mediciones fisiológicas* (Objetivo 5), se ha entrenado un modelo basado en un conjunto de datos propio. Este conjunto fue diseñado a partir de una experimentación realizada con sujetos a quienes se les aplicó el test DASS-21, avalado por la comunidad científica de psicología para medir el nivel de estrés percibido. Además, se ha desarrollado un modelo para la predicción de niveles de estrés (alto, medio y bajo) utilizando un conjunto de datos combinado SWELL y WESAD. Este modelo mejora la precisión del sistema previamente existente, que también se basaba en parámetros cardíacos, pero empleaba un modelo de predicción menos robusto y basado únicamente en el conjunto de datos SWELL. Para el entrenamiento de los modelos se utilizaron técnicas de AutoML, lo que permitió acelerar la búsqueda de un modelo y preprocesamiento óptimo. Aunque también se experimentó con técnicas de Deep Learning en el caso del modelo basado en un conjunto de datos propio, las métricas indicaban que no se superaba en rendimiento a los clasificadores tradicionales. Con estos dos modelos y el conjunto de características extraídas de las mediciones fisiológicas procesadas mediante el módulo HeartPy, se consigue calcular el perfil de estrés para un usuario y el nivel de estrés en ventanas de tiempo para una actividad realizada tal y como se explica en la Sección 3.6.

- En cuanto a *proporcionar una herramienta para la supervisión de usuarios mayores*

por parte de terapeutas (Objetivo 6) y a *desarrollar una aplicación para relojes inteligentes que registre actividades de diferentes tipos en la plataforma del terapeuta asociado* (Objetivo 9), en la extensión del capítulo 3 se han desarrollado mejoras sobre la aplicación web para que los terapeutas puedan registrar y visualizar los resultados de estrés que conlleva la realización de actividades de la vida diaria para sus usuarios. Para lograrlo, se han introducido mejoras tanto en la aplicación web como en la aplicación para reloj inteligente, de modo que ahora es posible registrar tipos de actividades y etiquetas personalizadas. Las actividades se configuran mediante asignación de etiquetas de contextos y estados. Ahora las actividades se descargan dinámicamente en el reloj según la asignación del terapeuta al usuario. Además, se ha incorporado la opción de etiquetar las distintas situaciones estresantes que puedan surgir durante el desarrollo de dichas actividades, al tiempo que se recogen mediciones de múltiples sensores, no solo el fisiológico, sino también el GPS y el acelerómetro, y se envían al servidor. Con ello, se cumple el objetivo 7 de *ampliar el número de sensores utilizados* y, como nueva funcionalidad, la aplicación web muestra la ruta en un mapa seguida por el usuario durante la actividad.

- Para el objetivo de *optimizar los recursos del reloj inteligente* (Objetivo 7), en la sección 3.7 se ha optimizado el uso de los recursos existentes de modo que la aplicación del reloj realiza la medición de los sensores desde un servicio en segundo plano, así que la medición no se detiene aún si la aplicación se suspende. De este modo el reloj puede estar en modo espera pero se seguirán tomando datos relacionados con la actividad que se esté realizando, pudiendo retomar el etiquetado de momentos en cualquier instante.

- Para la realización del proyecto se han usado metodologías ágiles, incluyendo Scrum para la planificación iterativa y Kanban para el seguimiento del flujo de trabajo, para llevar un desarrollo continuo, equilibrado y con retroalimentación constante por parte de las tutoras, de modo que se ha obtenido un software que cumpla todos los requisitos planteados y con calidad.

4.2. Trabajos futuros

El desarrollo que se ha llevado a cabo en este Trabajo Fin de Máster puede extenderse en el futuro de varias formas, una relacionada con la extensión de las tres aplicaciones desarrolladas, y otra referente al modelo de predicción y selección de características que se usan para medir el nivel de estrés en el usuario.

En relación a la extensión de las aplicaciones desarrolladas, se desglosan a continuación las propuestas para el futuro para cada aplicación:

4.2.1. Aplicación Web

Para la aplicación web se proponen las siguientes funcionalidades:

- Soporte para que varios terapeutas coexistan en el sistema cada uno con sus usuarios registrados, de modo que no tengan que compartir los usuarios registrados en común.
- Mejora en la visualización del gráfico de estrés, que dinámicamente se renderice el gráfico conforme el transcurso de la actividad.
- Migrar la parte frontal de plataforma web hacia otra tecnología más extendida de desarrollo web como React.
- Almacenar los datos de forma encriptada y realizar la instalación del sistema web en un servidor seguro cifrado mediante el certificado SSL, para que la transmisión de datos se realice con un protocolo seguro como HTTPS.

4.2.2. Aplicación Móvil

Para la aplicación móvil se sugieren las siguientes mejoras relacionadas con dar feedback al usuario:

- Mostrar al usuario identificado un emoticono o imagen, un sonido o vídeo personalizados, para que le ayuden a relajarse cuando se detecta estrés. De esta forma la solución puede formar parte de un programa de intervención, no solo de evaluación.
- Ampliar la funcionalidad de la aplicación móvil, permitiéndole al usuario consultar su perfil de estrés, sus actividades realizadas y el detalle para cada una de ellas.
- Mostrar al usuario un gráfico donde pueda consultar los niveles de estrés acumulados en las últimas actividades, clasificándolas por “hoy”, “últimos días”, “desde hace una semana”, “desde hace un mes”, o seleccionando por rango de fechas.

4.2.3. Aplicación Reloj Inteligente

Para la aplicación del reloj inteligente se plantean la siguiente mejora:

- Ampliar el soporte de la aplicación del reloj a otros modelos y marcas, estandarizando la toma de medidas del sensor fisiológico.

En relación a la extensión del modelo de predicción desarrollado, se pueden sugerir mejoras en cuanto al uso de otros conjuntos de datos, o en cuanto al uso en combinación con otros sensores del reloj inteligente, a modo de mejorar la predicción del nivel de estrés.

4.2.4. Modelos de predicción de estrés

Respecto a las mejoras de los modelos de predicción de estrés, se sugieren las siguientes mejoras:

- Validar los modelos creados para la predicción de estrés mediante experimentación controlada, para comprobar el correcto funcionamiento.
- Estudiar otros modelos y posibles combinaciones para mejorar la predicción de estrés.
- Investigar mediante el uso de otros sensores, como el acelerómetro y el giroscopio, si se puede mejorar el gráfico de estrés, mediante la detección de estar haciendo actividad física, ya que fisiológicamente se estaría dando una respuesta similar al estrés.

4.3. Valoración personal

La realización del presente Trabajo Fin de Máster ha supuesto un gran reto personal, ya que se ha estudiado en profundidad un problema, como es el estrés, que afecta a gran parte de la sociedad pero que se trata de invisibilizar o normalizar. Además, se ha tenido que tomar conciencia de los problemas de usuarios mayores en la realización de actividades diarias, lo cual me ha hecho empatizar aún más con las necesidades de estos colectivos.

En cuanto a la realización de un proyecto de esta envergadura, debido a la complejidad de la puesta en conjunto de tres sistemas que se comuniquen de forma simultánea, ha sido necesario realizar algún curso para conocer el desarrollo en Android nativo mediante Kotlin o el desarrollo de funciones como servicio (FaaS). La toma de decisiones en cuanto al uso de ciertas tecnologías, ha hecho que mis habilidades como ingeniero se apliquen y se extiendan bastante. Valoro de forma positiva que este trabajo se pueda ampliar en bastantes direcciones en un futuro. En cuanto a lo personal, considero que he sido muy constante en la realización de este proyecto, esforzándome al máximo en todos los ámbitos que abarca este trabajo, comunicándome regularmente con las tutoras y formándome en todo aquello que era necesario para realizar el desarrollo de una forma correcta y siguiendo convenciones, a fin de que este proyecto pueda ser mantenido y ampliado en un futuro.

Ha sido muy gratificante tanto ver cómo en el proceso de desarrollo el sistema iba tomando su forma final, diluyendo las numerosas dificultades encontradas en el camino. Esta experiencia me ha ayudado a ver cómo, mediante la constancia, se puede lograr realizar un buen estudio, propuesta y desarrollo de un proyecto que resuelva un problema presente en nuestra sociedad.

Apéndice A

Abreviaciones y Acrónimos

ADASYN Adaptive Synthetic Sampling. Método de sobremuestreo para manejar conjuntos de datos desbalanceados, generando ejemplos sintéticos de la clase minoritaria.

AndroidX Extensiones de Android. Conjunto moderno de bibliotecas que facilitan el desarrollo de aplicaciones Android.

Apple Watch Reloj inteligente desarrollado por Apple que incluye funciones avanzadas de salud y bienestar, incluida la gestión del estrés.

Auth:APIKEY Token de autenticación tipo *Bearer* utilizado para la interacción segura con nuestro sistema web.

AutoML Automated Machine Learning. Automatización del proceso de selección y entrenamiento de modelos de aprendizaje automático.

AWS Amazon Web Services. Plataforma de computación en la nube de Amazon que ofrece servicios como almacenamiento, bases de datos y despliegue de aplicaciones.

Azure AI Plataforma de inteligencia artificial ofrecida por Microsoft Azure, que incluye servicios para entrenar y desplegar modelos de Machine Learning.

B-Series Azure Familia de máquinas virtuales de bajo costo diseñadas para cargas de trabajo intermitentes en la plataforma Azure.

BBDD Bases de Datos. Sistema que almacena y gestiona información de forma estructurada.

BPM Beats Per Minute (Latidos por minuto). Medida de la frecuencia cardíaca.

Cascada Metodología de desarrollo de software secuencial con fases bien definidas (análisis, diseño, implementación, pruebas, mantenimiento).

CI/CD Continuous Integration/Continuous Deployment. Conjunto de prácticas de ingeniería de software para la integración continua de código y el despliegue automatizado.

Cloud Computing Modelo que permite el acceso remoto a recursos computacionales, como almacenamiento, procesamiento y servicios web bajo demanda.

CoC Convention Over Configuration (Convención sobre Configuración). Filosofía de desarrollo que reduce la complejidad al adoptar convenciones preestablecidas.

Complejo QRS Conjunto principal de ondas en el ECG que representa la *despolarización ventricular*.

CSV Comma-Separated Values. Formato de archivo utilizado para almacenar datos en forma de tabla, donde cada valor se separa con comas.

DASS-21 Depression Anxiety Stress Scales - 21 items. Cuestionario que mide niveles de depresión, ansiedad y estrés.

Diagrama de Gantt Herramienta gráfica para planificar y gestionar proyectos mostrando tareas y su duración en una línea de tiempo.

Docker Plataforma que permite crear, desplegar y gestionar aplicaciones dentro de contenedores virtualizados.

ECG Electrocardiograma. Mide la actividad eléctrica del corazón (ritmo cardíaco).

EDA Actividad Electrodérmica (Electrodermal Activity). Mide los cambios en la conductividad de la piel relacionados con la actividad del sistema nervioso simpático.

EEG Electroencefalograma (Electroencefalografía). Mide la actividad eléctrica del cerebro a través de sensores colocados en el cuero cabelludo.

ELK / ELK Stack Elasticsearch, Logstash y Kibana. Conjunto de herramientas para la búsqueda, análisis y visualización de trazas de ejecución en tiempo real.

Elasticsearch Sistema de búsqueda y análisis distribuido basado en texto. Permite almacenar, buscar y analizar grandes volúmenes de datos en tiempo real, siendo el núcleo de la suite ELK Stack.

EMG Electromiografía. Mide la actividad eléctrica de los músculos.

F1-Score Métrica que combina *precisión* y *recall* para evaluar el rendimiento de un modelo de clasificación.

FaaS (Function as a Service) Modelo de computación en la nube que permite ejecutar funciones bajo demanda sin necesidad de gestionar servidores.

FC Frecuencia Cardíaca. Número de latidos del corazón por unidad de tiempo.

Flutter Framework multiplataforma desarrollado por Google que permite crear aplicaciones nativas para iOS, Android y web usando Dart.

GSR Respuesta Galvánica de la Piel (Galvanic Skin Response). Término empleado para referirse a la actividad electrodérmica (EDA).

GPS Global Positioning System. Sistema de posicionamiento global utilizado para localización.

Gradient Boosting Técnica de aprendizaje automático que combina múltiples modelos débiles (generalmente árboles) para mejorar la precisión de las predicciones.

HeartPy Librería de Python para el análisis de datos de fotopletismografía (PPG), enfocada en la extracción de características del ritmo cardíaco.

HRV / VFC Heart Rate Variability / Variabilidad de la Frecuencia Cardíaca. Mide las fluctuaciones en los intervalos entre latidos del corazón; se utiliza para evaluar el estrés o el estado de salud cardiovascular.

Hz Hercios. Unidad de frecuencia que indica ciclos por segundo.

IRPF Impuesto sobre la Renta de las Personas Físicas. Impuesto personal que se impone sobre la renta de las personas.

Iterativo e incremental Metodología de desarrollo de software que construye el producto en ciclos cortos, entregando incrementos funcionales en cada iteración.

Jetpack Compose Kit de herramientas declarativas de Android para construir interfaces de usuario modernas y simplificar el desarrollo a través de componentes.

K8s (Kubernetes) Plataforma de orquestación de contenedores que automatiza el despliegue, la escala y la gestión de aplicaciones en contenedores.

Kibana Interfaz de usuario para la visualización e inspección de datos almacenados en Elasticsearch.

KNN K-Nearest Neighbors (K-Vecinos más Cercanos). Algoritmo de aprendizaje automático basado en la similitud con los k vecinos más cercanos.

KPI Key Performance Indicator (Indicador Clave de Desempeño). Métrica cuantitativa utilizada para medir el éxito en un área específica.

Licencia MIT / MIT License Licencia de software libre y de código abierto (permisiva) desarrollada originalmente por el Massachusetts Institute of Technology.

Logstash Herramienta de procesamiento de datos que recopila, filtra y envía logs y eventos a un almacenamiento o motor de búsqueda.

LOSO Leave-One-Subject-Out. Técnica de validación cruzada donde se usa un sujeto para pruebas y el resto para entrenamiento, rotando sucesivamente.

MAD Mean Absolute Deviation. Desviación media absoluta, utilizada en estadística para medir la variabilidad de un conjunto de datos.

MariaDB Sistema de gestión de bases de datos relacional derivado de MySQL, utilizado en entornos web y empresariales.

Material Design Sistema de diseño de Google con lineamientos visuales y de interacción para crear aplicaciones consistentes, accesibles y asegurando la usabilidad.

Material Design 3 Versión más reciente del sistema de diseño de Google, enfocada en accesibilidad, usabilidad, consistencia y personalización visual.

Mindfulness Práctica de atención plena que ayuda a manejar el estrés y la ansiedad mediante la concentración en el presente.

ML Machine Learning (Aprendizaje Automático). Disciplina de la inteligencia artificial que desarrolla algoritmos capaces de aprender a partir de datos.

MLFlow Machine Learning Flow. Plataforma para gestionar el ciclo de vida de modelos de aprendizaje automático (experimentación, reproducibilidad y despliegue).

mRMR Minimum Redundancy Maximum Relevance. Método de selección de características que busca maximizar la relevancia y minimizar la redundancia.

MVP Modelo-Vista-Presentador. Patrón arquitectónico donde el Presentador actúa como intermediario entre la Vista y el Modelo.

MVVM Modelo-Vista-ViewModel. Patrón arquitectónico que utiliza un *ViewModel* para exponer datos y lógica a la Vista, facilitando la separación de responsabilidades.

NFC Near Field Communication (Comunicación de Campo Cercano). Tecnología inalámbrica de corto alcance usada para intercambio de información o pagos.

O2 Operador de telecomunicaciones propuesto para la conexión a internet.

OAuth Open Authorization. Protocolo abierto de autorización que permite el acceso seguro a recursos protegidos.

OMS Organización Mundial de la Salud (World Health Organization).

Onda T Onda del ECG asociada a la *repolarización de los ventrículos.*

OpenFaaS Framework de código abierto para desplegar funciones *serverless* de manera escalable utilizando contenedores (Docker/Kubernetes).

Pixel 6 Teléfono inteligente de Google con sistema operativo Android.

Planning Poker Técnica de estimación ágil para asignar puntos de historia a tareas mediante consenso del equipo, ayudando a predecir esfuerzo.

PPG Fotopletismografía (Photoplethysmography). Técnica para medir cambios en el volumen sanguíneo mediante luz y un fotodetector.

Programación Extrema (XP) Metodología ágil que enfatiza la entrega temprana, pruebas unitarias frecuentes y la colaboración continua con el cliente.

Random Forest Algoritmo de aprendizaje automático basado en múltiples árboles de decisión para mejorar la precisión y evitar sobreajuste.

React Native Framework de JavaScript basado en React.js que permite desarrollar aplicaciones nativas multiplataforma.

RGPD Reglamento General de Protección de Datos. Marco legal de la Unión Europea para la protección de datos personales.

RoR Ruby on Rails. Framework de desarrollo web basado en Ruby que sigue el patrón MVC y la filosofía *Convention over Configuration*.

RMSSD Root Mean Square of Successive Differences. Métrica derivada de HRV/VFC que calcula la raíz cuadrada media de las diferencias sucesivas entre intervalos de latidos.

RR Intervalo entre latidos en el ECG, es decir, la distancia entre dos complejos QRS consecutivos.

S3 Simple Storage Service. Servicio de almacenamiento de objetos en la nube de AWS.

SDNN Standard Deviation of NN intervals. Desviación estándar de los intervalos entre latidos sucesivos (NN) en señales cardíacas.

SDK Software Development Kit. Conjunto de herramientas y bibliotecas para desarrollar aplicaciones, por ejemplo, en Android u otras plataformas.

SKT Temperatura de la piel (Skin Temperature). Sensor que mide la temperatura cutánea, a menudo asociado a la detección de estrés.

SMOTE Synthetic Minority Oversampling Technique. Técnica para balancear conjuntos de datos desbalanceados sobremuestreando la clase minoritaria.

SSH Secure Shell. Protocolo de red que permite el acceso remoto seguro a un sistema o servidor.

SSL Secure Sockets Layer. Protocolo criptográfico para garantizar la seguridad de la comunicación en redes de ordenadores.

SVM Máquina de Vectores de Soporte (Support Vector Machine). Algoritmo de aprendizaje automático usado para clasificación y regresión.

SWELL Smart Reasoning for Well-being at Home and at Work. Proyecto de investigación sobre estrés en oficinas, también asociado a un conjunto de datos.

TalkBack Lector de pantalla de Android para mejorar la accesibilidad, proporcionando retroalimentación audible de la interfaz.

TicWatch Pro Reloj inteligente de la marca Mobvoi que ejecuta WearOS y dispone de sensores como PPG.

TSST Trier Social Stress Test. Prueba que induce estrés en un entorno controlado para estudios psicológicos y fisiológicos.

UNECE United Nations Economic Commission for Europe. Comisión Económica para Europa de las Naciones Unidas.

vCPU Unidad Central de Procesamiento Virtual (virtual CPU). Recurso de CPU virtual utilizado en máquinas virtuales.

WESAD Wearable Stress and Affect Detection. Conjunto de datos multimodal para investigar el estrés y emociones usando dispositivos portátiles.

WearOS Sistema operativo de Google basado en Android diseñado específicamente para relojes inteligentes (*smartwatches*).

XGBoost Extreme Gradient Boosting. Algoritmo de aprendizaje automático basado en árboles de decisión para problemas de clasificación y regresión.

Apéndice B

DASS-21

DASS-21

Por favor lea las siguientes afirmaciones y coloque un círculo alrededor de un número (0, 1, 2, 3) que indica en qué grado le ha ocurrido a usted esta afirmación *durante la semana pasada*. La escala de calificación es la siguiente*:*

0: No me ha ocurrido; 1: Me ha ocurrido un poco, o durante parte del tiempo; 2: Me ha ocurrido bastante, o durante una buena parte del tiempo; 3: Me ha ocurrido mucho, o la mayor parte del tiempo.

1.	Me ha costado mucho descargar la tensión	0	1	2	3
2.	Me di cuenta que tenía la boca seca	0	1	2	3
3.	No podía sentir ningún sentimiento positivo	0	1	2	3
4.	Se me hizo difícil respirar	0	1	2	3
5.	Se me hizo difícil tomar la iniciativa para hacer cosas	0	1	2	3
6.	Reaccioné exageradamente en ciertas situaciones	0	1	2	3
7.	Sentí que mis manos temblaban	0	1	2	3
8.	He sentido que estaba gastando una gran cantidad de energía	0	1	2	3
9.	Estaba preocupado por situaciones en las cuales podía tener pánico o en las que podría hacer el ridículo	0	1	2	3
10.	He sentido que no había nada que me ilusionara	0	1	2	3
11.	Me he sentido inquieto	0	1	2	3
12.	Se me hizo difícil relajarme	0	1	2	3
13.	Me sentí triste y deprimido	0	1	2	3
14.	No toleré nada que no me permitiera continuar con lo que estaba haciendo….	0	1	2	3
15.	Sentí que estaba al punto de pánico	0	1	2	3
16.	No me pude entusiasmar por nada.....	0	1	2	3
17.	Sentí que valía muy poco como persona	0	1	2	3
18.	He tendido a sentirme enfadado con facilidad	0	1	2	3
19.	Sentí los latidos de mi corazón a pesar de no haber hecho ningún esfuerzo físico	0	1	2	3
20.	Tuve miedo sin razón	0	1	2	3
21.	Sentí que la vida no tenía ningún sentido.....	0	1	2	3

Depression Anxiety and Stress Scale - 21 (DASS-21)

Referencia original: Antony, M. M., Bieling, P. J., Cox, B. J., Enns, M. W., & Swinson, R. P. (1998). Psychometric properties of the 42-item and 21-item versions of the Depression Anxiety Stress Scales (DASS) in clinical groups and a community sample. *Psychological Assessment, 10*, 176-181. doi: 10.1037/1040-3590.10.2.176

Validación en Colombia: Ruiz, F. J., García-Martín, M. B., Suárez-Falcón, J. C., & Odriozola-González, P. (2017). The hierarchical factor structure of the Spanish version of Depression Anxiety and Stress Scale - 21. *International Journal of Psychology and Psychological Therapy*, *17*, 97-105.

Modo de corrección: el DASS-21 posee tres subescalas, Depresión (ítems: 3, 5, 10, 13, 16, 17 y 21), Ansiedad (ítems: 2, 4, 7, 9, 15, 19 y 20) y Estrés (ítems: 1, 6, 8, 11, 12, 14 y 18). Para evaluar cada subescala por separado, se deben sumar las puntuaciones de los ítems correspondientes a cada una. Pueden obtenerse un indicador general de síntomas emocionales sumando las puntuaciones de todos los ítems.

Interpretación: a mayor puntuación general, mayor grado de sintomatología.

Puntos de corte comúnmente utilizados:

Depresión:
5-6 depresión leve
7-10 depresión moderada
11-13 depresión severa
14 o más, depresión extremadamente severa.

Ansiedad:
4 ansiedad leve
5-7 ansiedad moderada
8-9 ansiedad severa
10 o más, ansiedad extremadamente severa.

Estrés:
8-9 estrés leve
10-12 estrés moderado
13-16 estrés severo
17 o más, estrés extremadamente severo.

Apéndice C

Manual de instalación

En este capítulo se expondrá cómo instalar las plataformas desarrolladas en este Trabajo Fin de Máster. El código completo del proyecto se encuentra alojado en el repositorio público del enlace a **GitHub**.

C.1. Sistema web

Para realizar la instalación del sistema web, debemos tener un computador con unas características similares a las que se exponen en la sección 1.5. Se mostrarán los pasos a seguir para la instalación del sistema en un sistema operativo basado en Linux como Ubuntu 22.04.

Comenzaremos con la instalación de Docker en nuestro computador actualizando los índices de paquetes de apt para poder usar repositorios sobre HTTPS.

```
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl
```

Añadiremos, tras esto, la clave oficial GPG de Docker:

```
$ sudo install -m 0755 -d /etc/apt/keyrings
s$ udo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/
    ↪ keyrings/docker.asc
$ sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Después de ejecutar lo anterior, configuraremos el repositorio:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
      ↪ https://download.docker.com/linux/ubuntu \
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Habiendo realizado los pasos anteriores, instalaremos el motor Docker ejecutando los siguientes comandos:

```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin
    ↪ docker-compose-plugin
```

Teniendo ya instalado el motor de Docker en el servidor, debemos localizar el árbol de directorios del proyecto. Esto se puede hacer, bien consultando los archivos adjuntos a esta memoria, o bien clonando el repositorio de GitHub asociado a este proyecto, mediante el siguiente comando.

```
$ git clone https://github.com/VictorRubia/TFM.git
```

Una vez tenemos localizados los archivos y carpetas del proyecto, nos situaremos con un terminal dentro de la carpeta "backend". Habiendo hecho esto, ejecutaremos el comando que despliega el servidor.

```
$ cd backend
$ docker compose up
```

Finalmente, una vez que termine la instalación, podremos acceder al servidor en la dirección http://localhost:3000/.

C.2. Aplicación móvil

Para instalar la aplicación móvil podemos acceder al siguiente directorio del proyecto, donde encontraremos un archivo denominado *mobile-debug.apk*, que podemos transferir al teléfono e instalar habilitando instalación por orígenes desconocidos.

```
$ cd app/mobile/build/outputs/apk/debug/
```

De forma alternativa se puede obtener el *.apk*, tras realizar la instalación y el despliegue del servidor web, accediendo desde el navegador del ordenador donde se haya desplegado el servidor a la dirección http://localhost:3000/download_android_apk.

C.3. Aplicación reloj inteligente

Para poder instalar la aplicación en un reloj inteligente debemos tener activado el modo desarrollador, que se puede realizar accediendo a la configuración del reloj, desplazándonos a la categoría "Sistema", tras esto pulsaremos sobre "Información" y pulsaremos 7 veces sobre "Número de compilación", hasta que se muestre un mensaje por pantalla donde se indique que se ha activado correctamente el modo desarrollador.

Una vez hemos hecho esto, podemos activar dentro de las opciones de desarrollador la "Depuración ADB sobre Wi-Fi". Nos aparecerá una dirección IP que debemos de guardar para introducirla posteriormente en el terminal del ordenador que usemos para transferir la aplicación.

En el computador debemos tener instalado ADB, y si no se tiene ya instalado se debe realizar la instalación mediante la ejecución de del siguiente comando y comprobar que se ha instalado correctamente

```
$ sudo apt install android-tools-adb
$ adb version
```

A continuación debemos establecer una conexión entre el ordenador, sobre el que hemos ejecutado los comandos del paso anterior, y el reloj inteligente introduciendo el siguiente comando en el terminal e introduciendo la IP que nos guardamos previamente correspondiente al reloj inteligente.

```
$ adb connect (IP_RELOJ_INTELIGENTE)
```

Tras esto, se debe enviar el apk a la memoria interna del reloj inteligente e instalarlo mediante el siguiente comando. El archivo *.apk* para el reloj inteligente se puede encontrar en la ruta que se muestra a continuación. De igual forma, se puede descargar el archivo *.apk* habiendo instalado y desplegado el servidor web en el siguiente enlace http://localhost:3000/download_wearos_apk

```
$ cd app/wear/build/outputs/apk/debug/
$ adb push wear-debug.apk /sdcard/
$ adb -e install wear-debug.apk
```

Apéndice D

Manual de usuario

En este capítulo se explica cómo usar los distintos sistemas y las funcionalidades que ofrece.

D.1. Aplicación móvil

Tras la instalación de la aplicación en el dispositivo móvil, la iniciaremos pulsando sobre el icono de la aplicación, como se muestra en la figura D.1

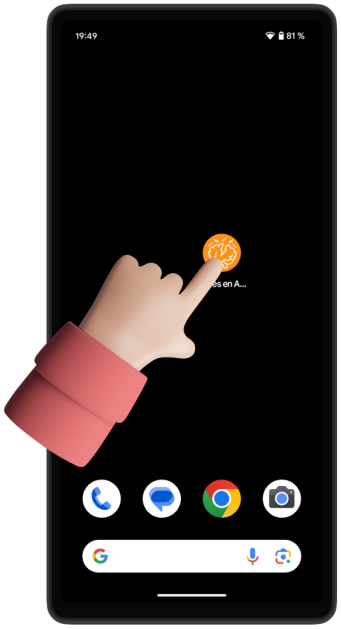


Figura D.1: Pantalla inicial de un teléfono Android.

D.1.1. Identificación del usuario

A continuación, en la figura D.3, se puede ver una pantalla donde existen dos campos para introducir un correo electrónico y una contraseña. Estas deben haberse creado tras recibir un correo de verificación de cuenta cuando el terapeuta lo dio de alta en la plataforma y debe tener la forma que se muestra en la figura D.2. Procedemos a introducirlas en los campos y a pulsar sobre entrar para identificarnos y poder empezar a realizar mediciones en el reloj inteligente.

Figura D.2: Correo electrónico recibido tras el registro.

Figura D.3: Pantalla para la identificación de un usuario.

D.1.2. Pantalla de usuario identificado

En la figura D.4, se muestra el aspecto de la pantalla tras haberse identificado el usuario en la aplicación. Esta pantalla muestra el nombre o seudónimo del usuario junto al estado de la conexión con el reloj inteligente. Será un icono de verificación verde cuando el reloj esté correctamente conectado al teléfono, y en caso contrario, se mostrará una cruz de color rojo como se ve en la figura D.5. En este punto el usuario puede dejar de usar la aplicación del teléfono, incluso cerrándola desde el administrador de tareas, ya que será capaz de identificar en el reloj al usuario que se encuentre identificado en el teléfono cuando quiera comenzar a medir sus actividades.

Figura D.4: Pantalla tras el inicio de sesión del usuario con conexión correcta al reloj inteligente.

Figura D.5: Pantalla tras el inicio de sesión del usuario sin conexión al reloj inteligente.

D.1.3. Recuperar contraseña

Si se quiere recuperar la contraseña, porque se ha olvidado, se puede pulsar sobre el botón "Recuperar contraseña" de la pantalla para la identificación (Figura D.3). En esta pantalla (Figura D.6) debemos introducir el correo electrónico con el que el terapeuta nos registró en el sistema. La solicitud de restablecimiento contraseña será enviada al correo que se ha indicado (Figura D.7).

Figura D.6: Pantalla de recuperación de contraseña.

Figura D.7: Correo recibido tras la solicitud de contraseña.

D.1.4. Cerrar sesión

Para cambiar el usuario que utiliza el reloj, se puede usar la función de cerrar sesión. En la figura D.8 se muestra arriba a la derecha un botón para el cierre de sesión de un usuario ya identificado. De esta forma, se volverá a la pantalla de identificación que se muestra en la figura D.3.

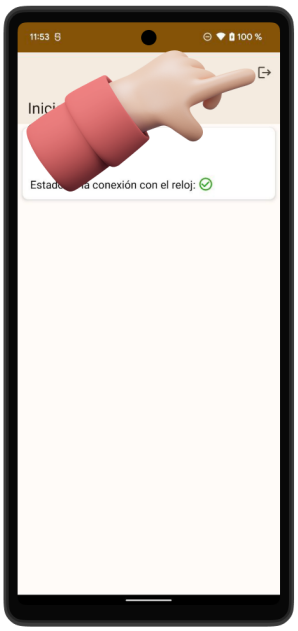


Figura D.8: Pantalla de usuario identificado para cerrar sesión.

D.2. Plataforma Web

En la plataforma web los terapeutas pueden darse de alta para monitorizar las actividades que realicen los usuarios de la plataforma. En esta plataforma el terapeuta puede gestionar a los usuarios, registrándolos, editándolos y eliminándolos junto con el poder visualizar el procesamiento de niveles de estrés acumulados en el transcurso de las actividades realizadas por dichos usuarios. En los siguientes apartados se expone cómo usar la plataforma web.

D.2.1. Registro de terapeuta

Para comenzar a usar la plataforma web es indispensable que un terapeuta se registre en el sistema. Para ello, nos situaremos en la página de inicio y situaremos el cursor sobre

el botón situado en la barra negra en la esquina superior derecha llamado “Registrarse” tal y como se muestra en la figura D.12. A continuación, veremos una pantalla como la de la figura D.10, en la que debemos rellenar los datos que se piden, correo electrónico y contraseña. Tras esto, pulsando el botón registrarse, llegará al buzón de correo de la dirección indicada un mensaje para verificar la cuenta como el de la figura D.11.

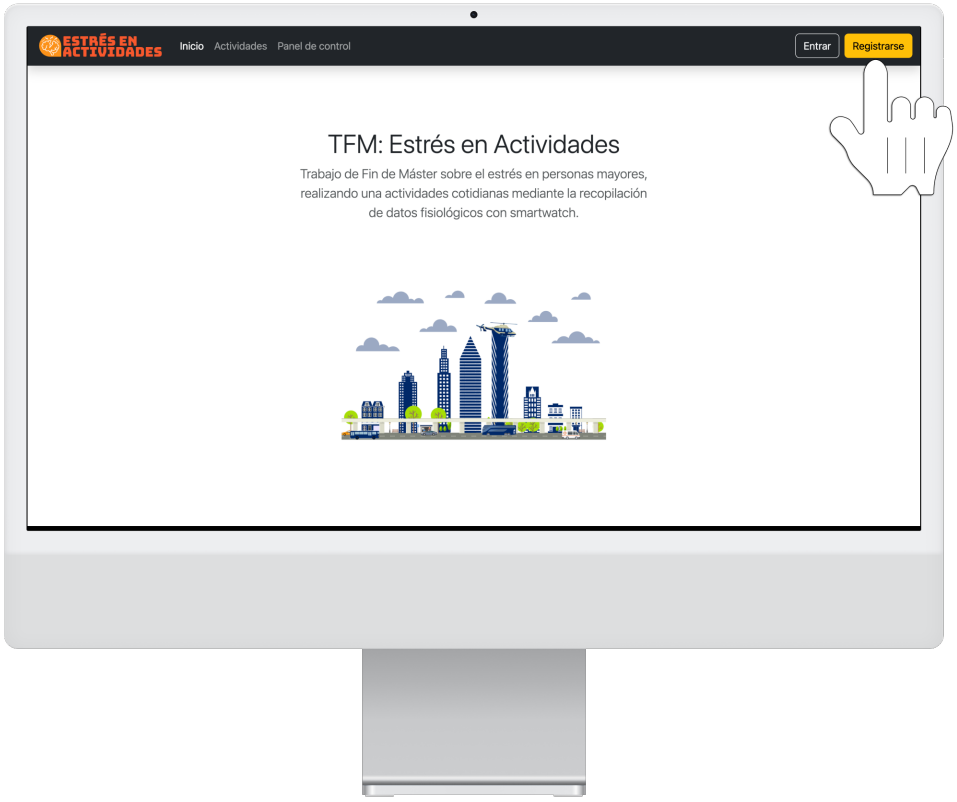


Figura D.9: Pantalla inicial de la plataforma web.

Figura D.10: Pantalla de registro de la plataforma web.

Figura D.11: Correo de verificación de cuenta para el terapeuta. Para poder empezar a usar la plataforma es necesario pulsar sobre el botón de "Verificar Cuenta"

D.2.2. Identificación de terapeuta y recuperación de contraseña

En el caso en el que se tenga ya una cuenta registrada como terapeuta en la plataforma y queramos identificarnos en el sitio web, debemos situarnos en la pantalla inicial y pulsar sobre el botón de "Entrar" situado en la barra de navegación negra en la esquina superior derecha de la pantalla, como aparece en la figura D.12. Tras esto, debemos rellenar el formulario con el correo electrónico y contraseña del terapeuta que va a identificarse y pulsar sobre el botón "Entrar" como se muestra en la figura D.13. En esta página podemos encontrar también la opción para crear una cuenta, para recuperar la contraseña si se ha olvidado y la opción de volver a enviar el correo de verificación de cuenta en el caso en el que no haya llegado al buzón de correo del terapeuta.

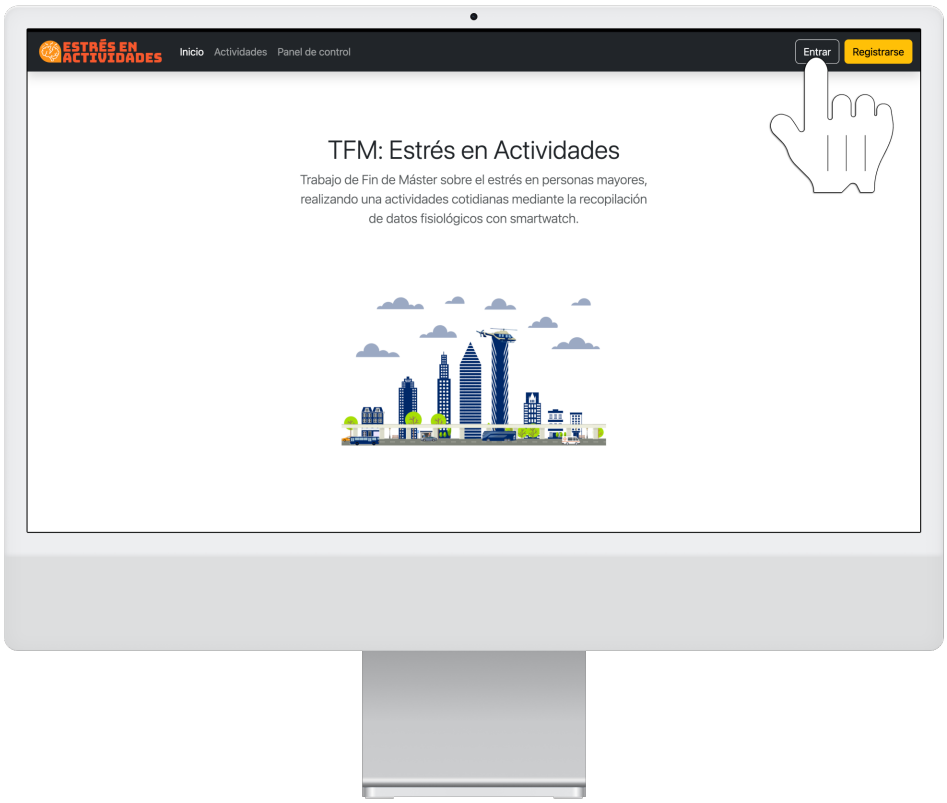


Figura D.12: Pantalla inicial de la plataforma web.

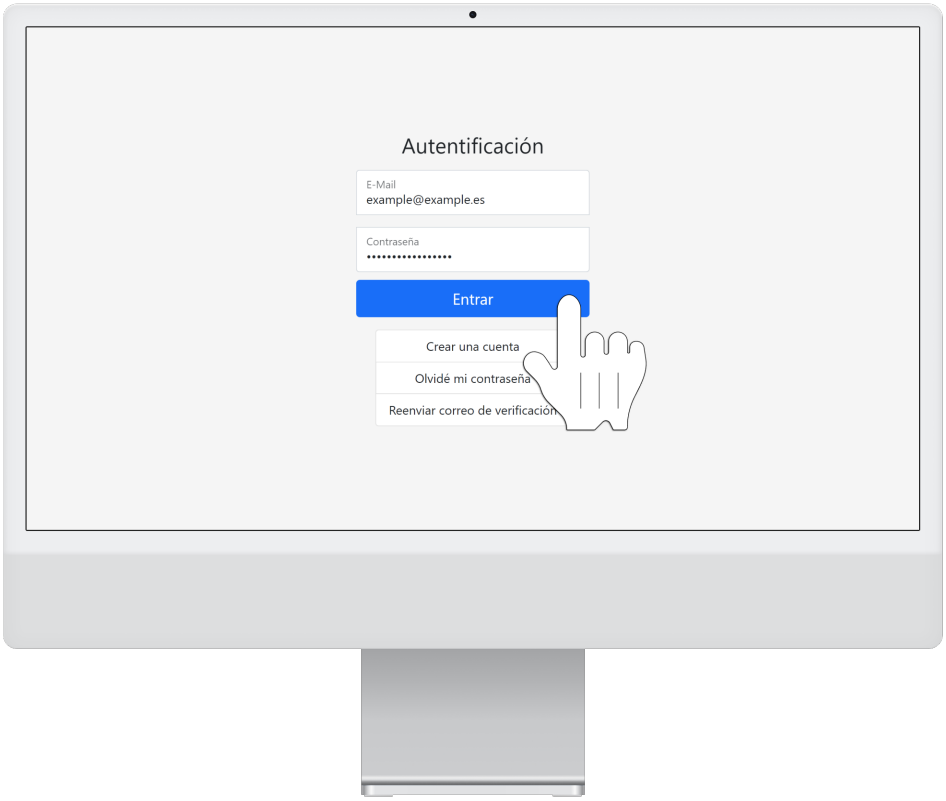


Figura D.13: Pantalla para identificarse en la plataforma web.

D.2.3. Gestión de usuarios

Una vez está el terapeuta identificado en la plataforma, se puede proceder visitar el panel de control donde poder gestionar los usuarios que tenemos en la plataforma, como se muestra en la figura D.14. Para ello, debemos, en primer lugar, pulsar sobre "Usuarios" en la barra lateral izquierda tal y como se muestra en la figura D.15. Tras esto, vemos en un listado a los usuarios que tenemos registrados en la plataforma como se muestra en la figura D.16. Se proporciona además, un buscador para facilitar encontrar al usuario deseado en las ocasiones en las que tenemos numerosos usuarios.

Figura D.14: Acceso a panel de control desde la pantalla de inicio.

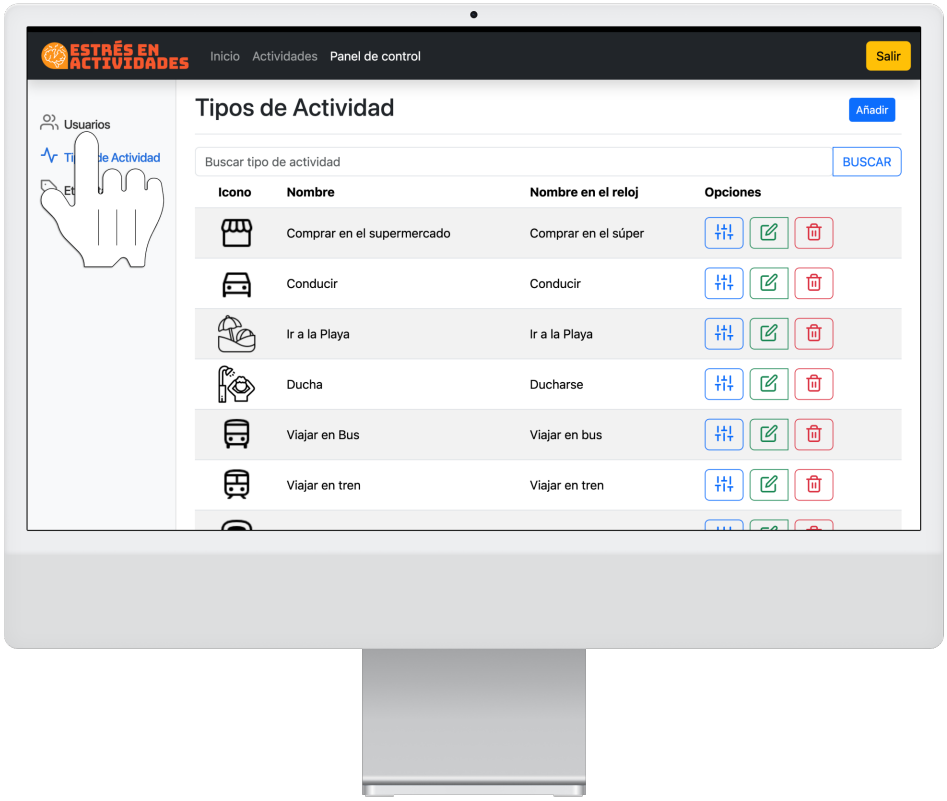


Figura D.15: Panel de control acceso a gestión de usuarios.

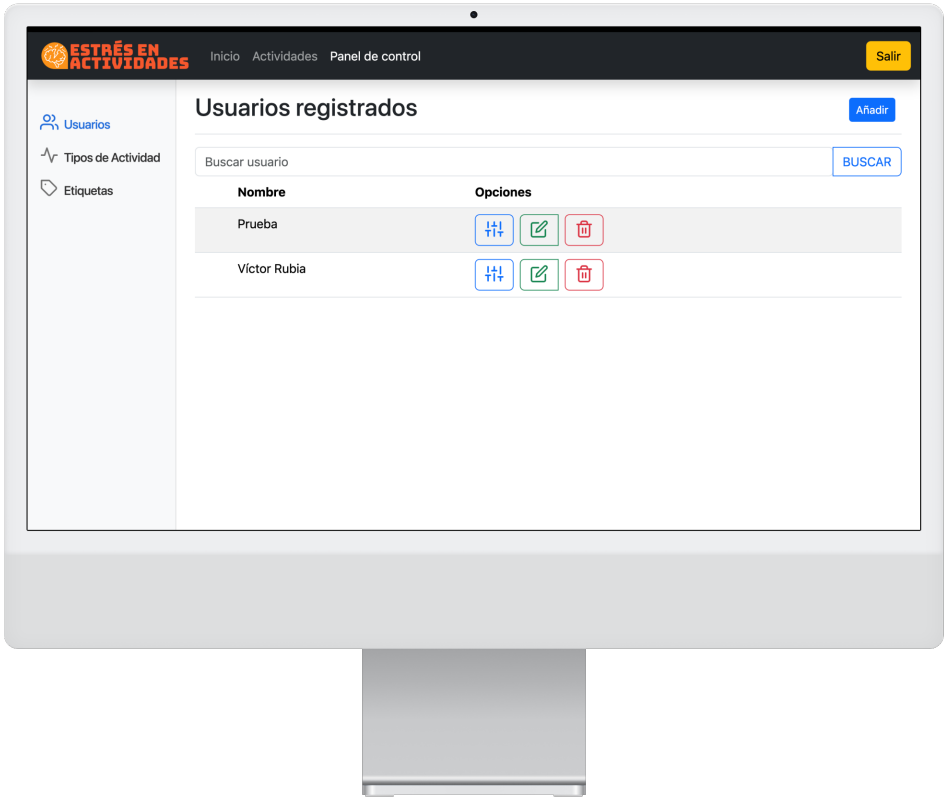


Figura D.16: Panel de control sección de gestión de usuarios.

Registrar un usuario

Para registrar a un usuario en la plataforma web, pulsaremos sobre el botón azul de la esquina superior derecha llamado “Añadir”, como aparece en la figura D.17. Aparecerá entonces una ventana en medio de la pantalla en la que se encuentra un formulario a rellenar con el nombre o apodo del usuario, el correo del usuario y una contraseña para que el usuario pueda identificarse en la plataforma, como se muestra en la figura D.18. La contraseña la puede establecer el terapeuta o puede consultarlo las preferencias del usuario. Tras rellenarlo todo, se debe pulsar sobre el botón verde “Añadir” para completar el registro del nuevo usuario. Esto hará que se le envíe un correo a la dirección proporcionada con los datos de identificación introducidos, como se muestra en la figura D.2.

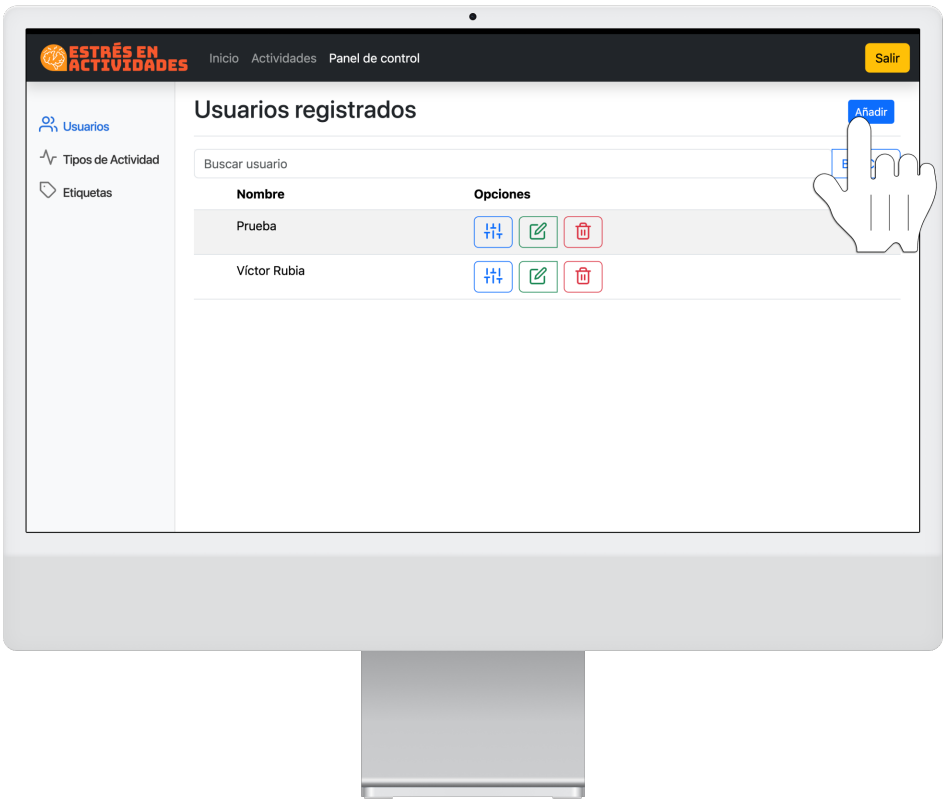


Figura D.17: Gestión de usuarios registrar un usuario.

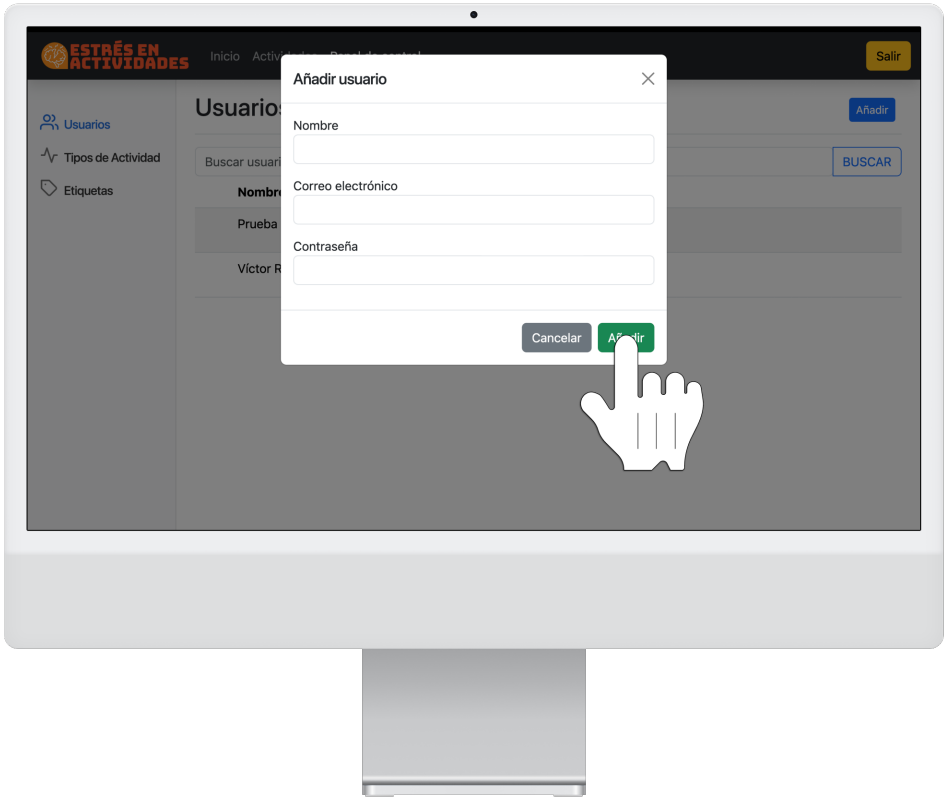


Figura D.18: Formulario para introducir información para registrar un usuario.

Edición de usuario

Para poder cambiar la información de un usuario registrado nos situaremos sobre el apartado de gestión de usuarios dentro del panel de control. Tras esto, pulsaremos sobre el botón verde indicado mediante un lápiz bajo la columna "Opciones" de la tabla en la fila donde se encuentre el usuario para el que queramos editar la información, como se muestra en la figura D.19. Tras esto, se muestra una ventana sobre la pantalla con un formulario que se encuentra relleno con la información perteneciente al usuario seleccionado. Sobre esto, podemos realizar las modificaciones y poder guardarlas pulsando sobre "Editar", como se muestra en la figura D.20.

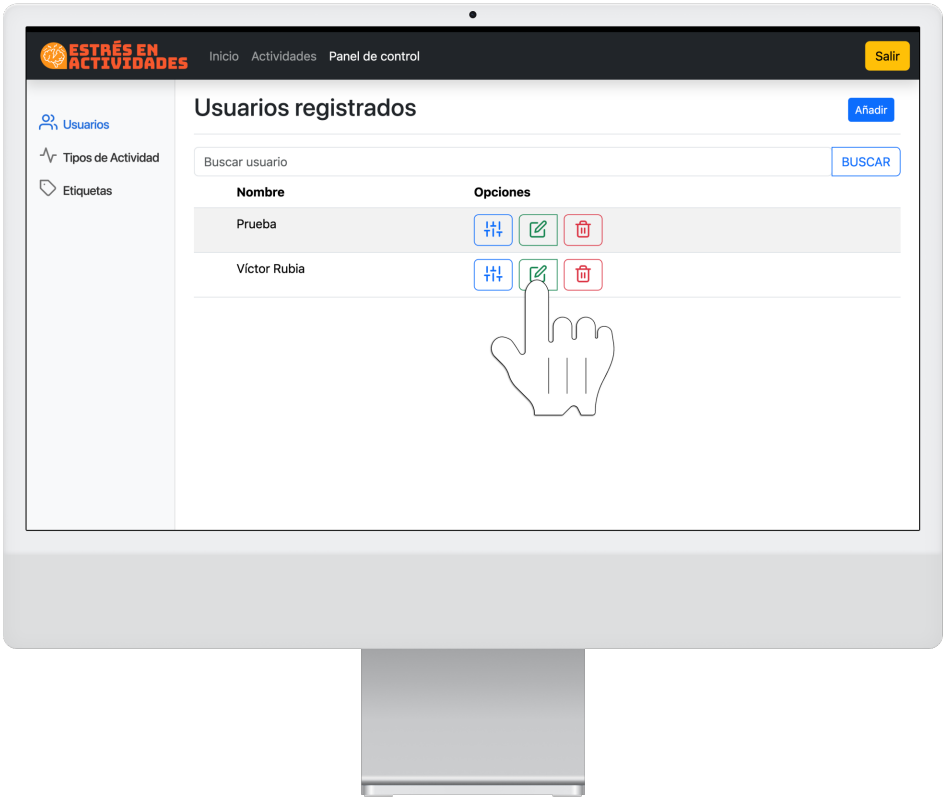


Figura D.19: Gestión de usuarios editar un usuario.

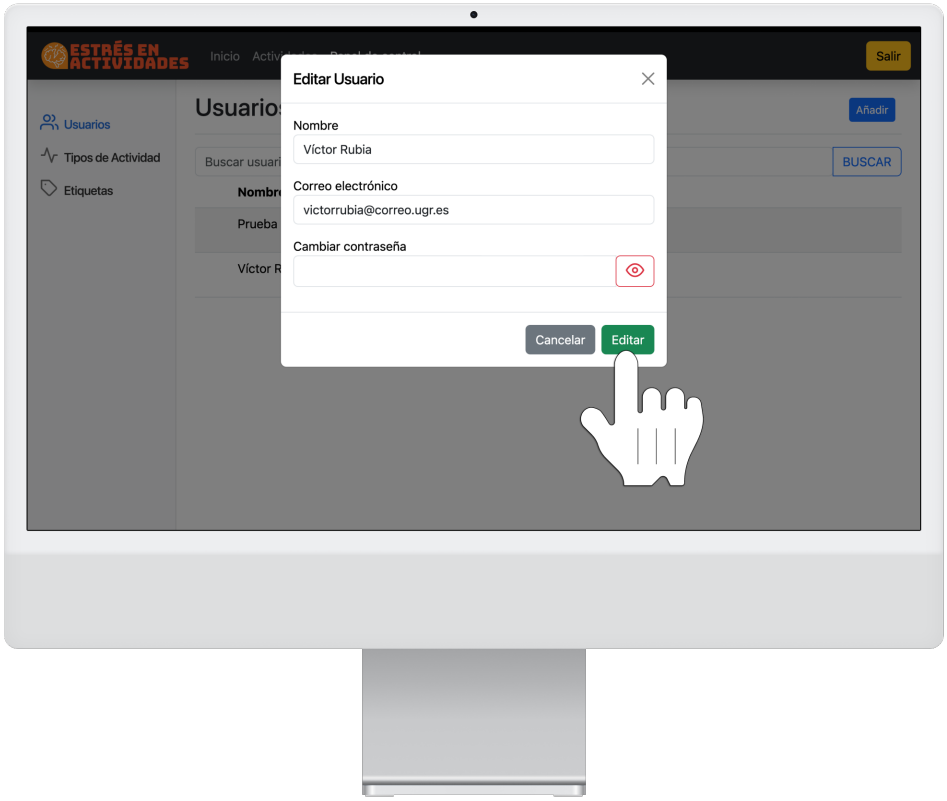


Figura D.20: Formulario para editar la información de un usuario registrado.

Eliminar usuario

Si se quiere eliminar un usuario que está registrado en el sistema, nos volveremos a situar sobre el apartado de gestión de usuarios dentro del panel de control. En esta página, pulsaremos sobre el botón rojo indicado mediante una papelera bajo la columna "Opciones" de la tabla en la fila donde se encuentre el usuario que queramos eliminar, como se muestra en la figura D.21. Tras esto se mostrará una ventana emergente para confirmar la acción de borrado.

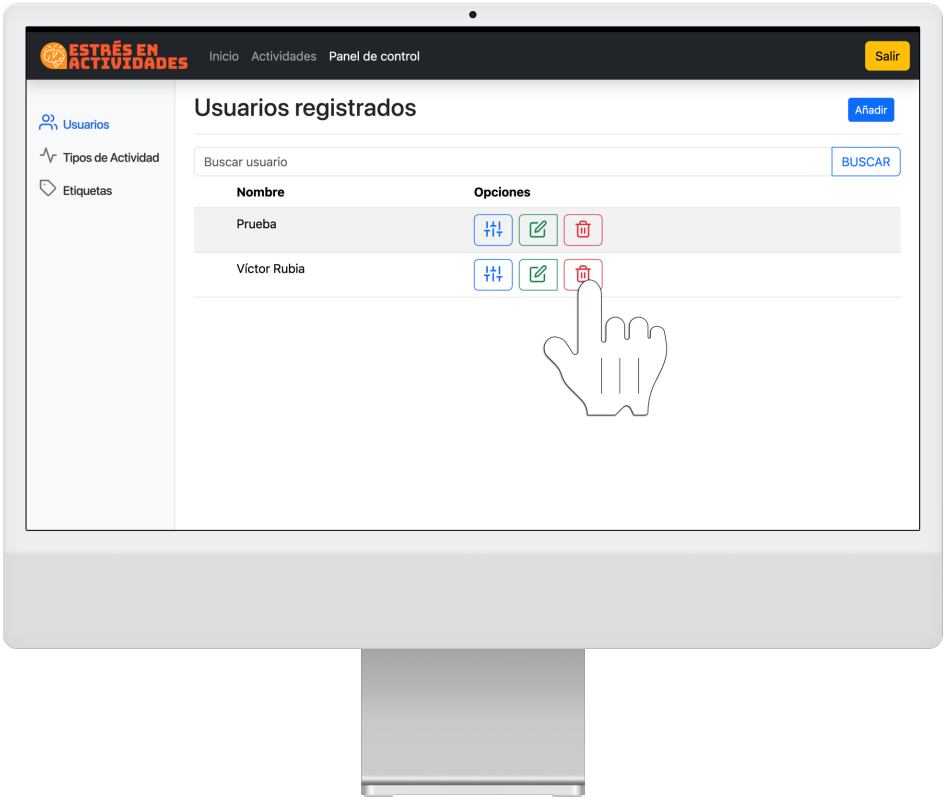


Figura D.21: Eliminar un usuario registrado.

Asignación de actividades a un usuario

Para asignar tipos de actividades registradas en el sistema a los usuarios, para que les aparezca en su reloj a la hora de registrar mediciones, debemos pulsar sobre el botón azul que se muestra en la Figura D.22 bajo la columna "Opciones". Tras esto se muestra un listado con las actividades disponibles y se deben marcar aquellas que se desean asignar para este usuario, tal y como se muestra en la Figura **??**

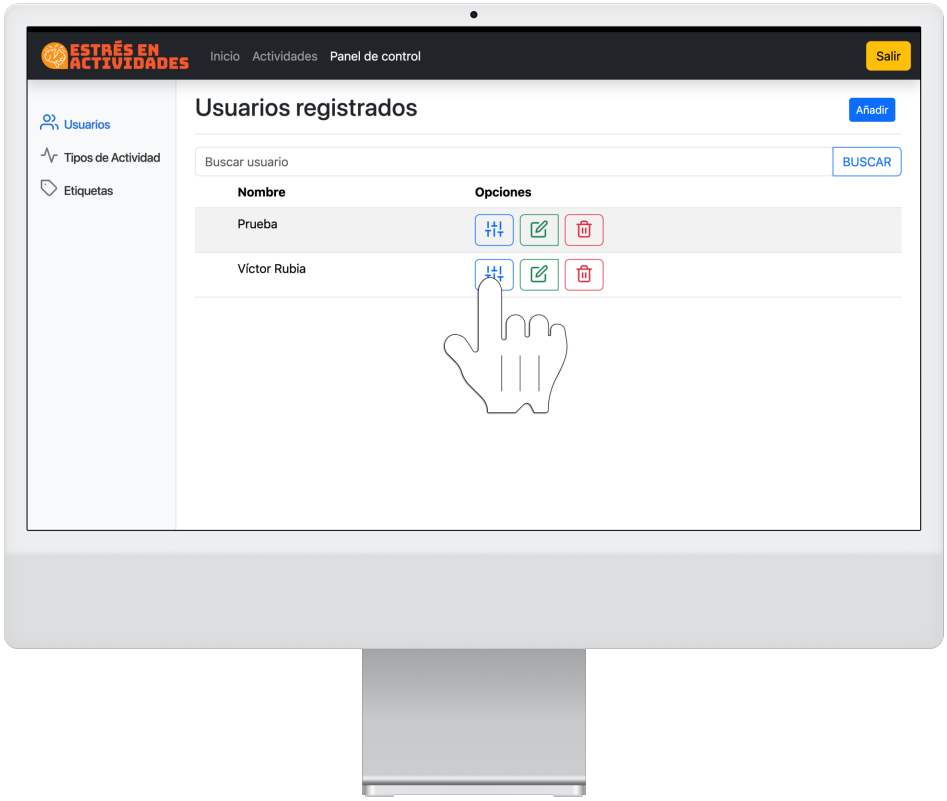


Figura D.22: Botón para asignar actividades a un usuario.

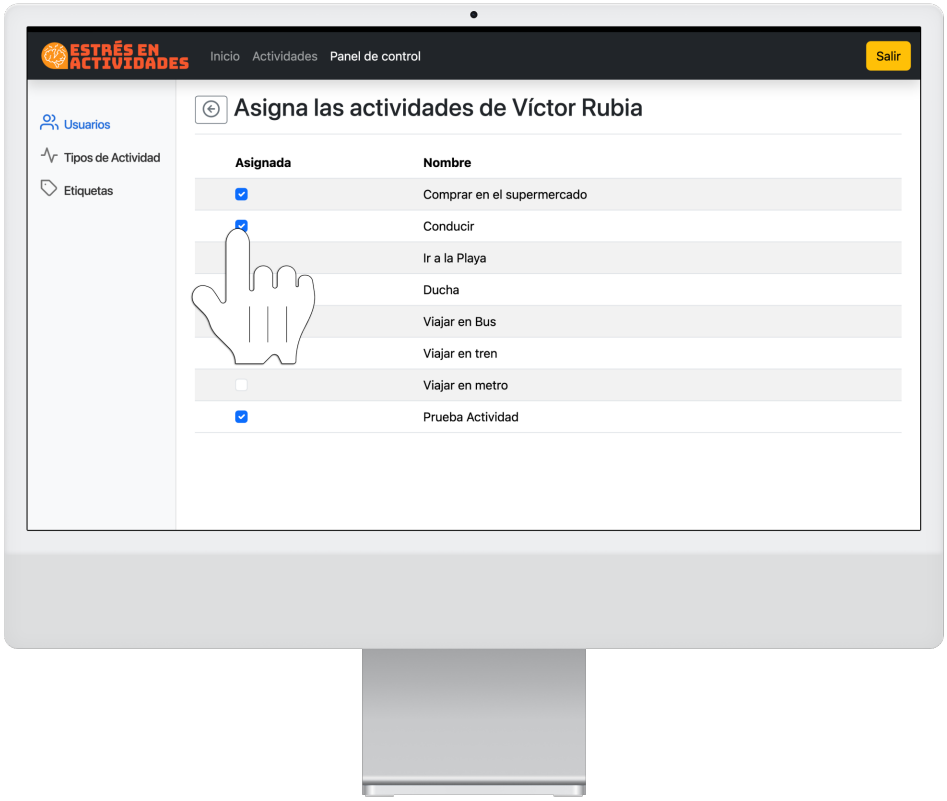


Figura D.23: Formulario para asignar tipos de actividad a un usuario.

D.2.4. Gestión de actividades

Si se quieren añadir nuevos tipos de actividad a la plataforma bastaría con dirigirnos al apartado designado para ello, tal y como se muestra en la Figura D.24. Esto nos permite, al pulsar sobre el botón azul de "Añadir", situado arriba a la derecha, rellenar un formulario en el que debemos introducir la información para dicha actividad, junto con un icono que la describa. Este formulario se muestra en la Figura D.25. Igualmente, las actividades pueden configurarse, con el botón azul, editarse, con el botón verde, y eliminarse, con el botón rojo. La configuración de la actividad se realiza para asignarle en cierto orden las etiquetas que la componen, tal y como se muestra en la Figura D.26.

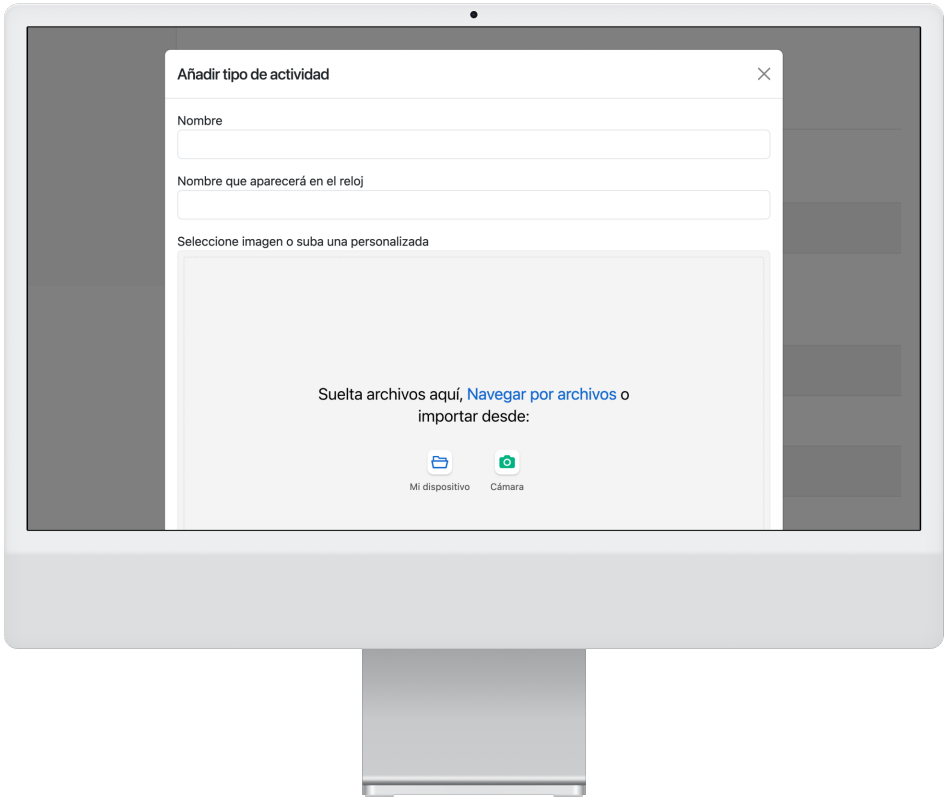


Figura D.24: Botón para añadir un nuevo tipo de actividad al sistema.

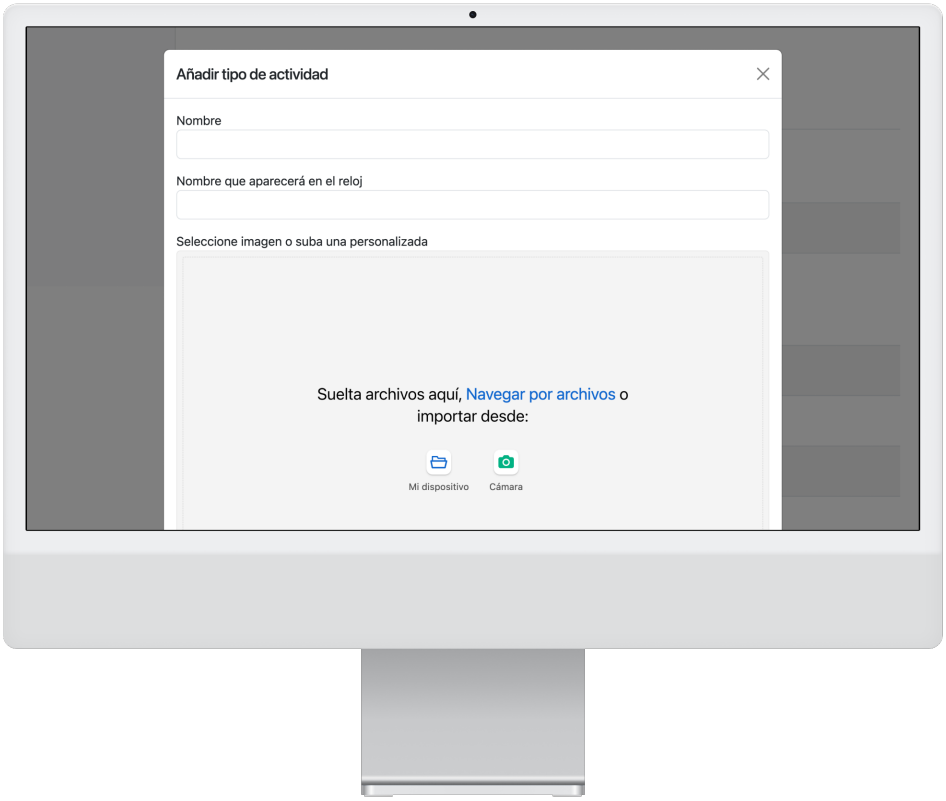


Figura D.25: Formulario para añadir una nueva actividad.

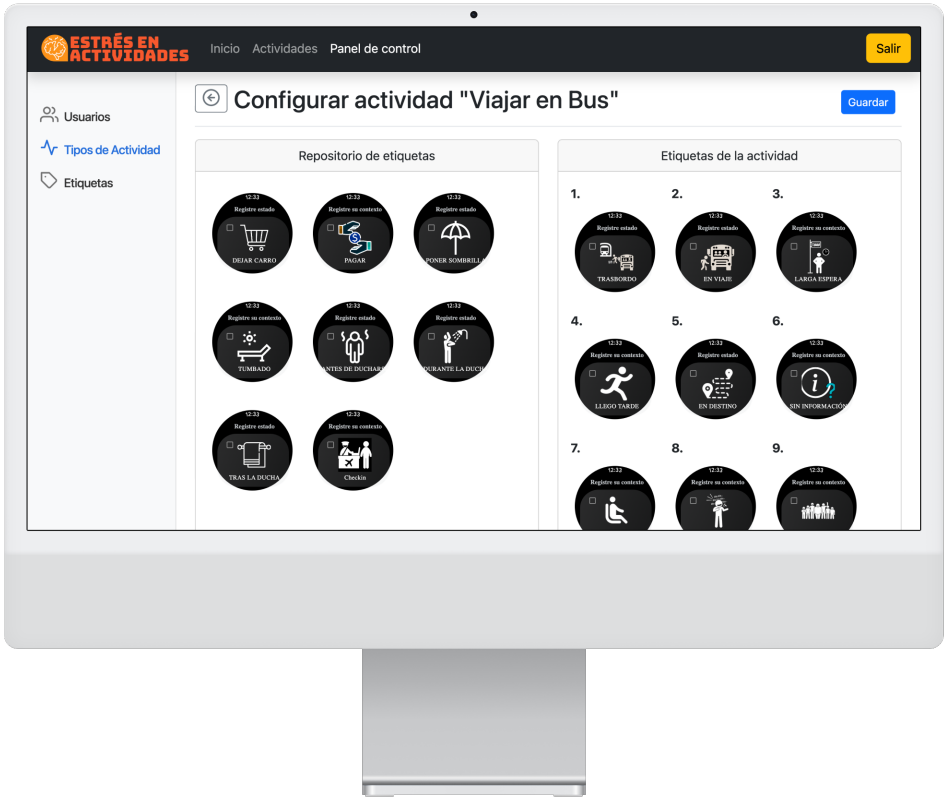


Figura D.26: Configurar las etiquetas de una actividad.

D.2.5. Gestión de etiquetas

Para añadir nuevas etiquetas para las actividades a la plataforma debemos dirigirnos al apartado de "Etiquetas", tal y como se muestra en la Figura **??**. Esto nos permite, al pulsar sobre el botón azul de "Añadir", situado arriba a la derecha, rellenar un formulario en el que debemos introducir la información para dicha etiqueta, el tipo del que se trata, junto con un icono que la describa. Este formulario se muestra en la Figura **??**. Igualmente, las etiquetas se pueden editar, con el botón verde, y eliminarse, con el botón rojo.

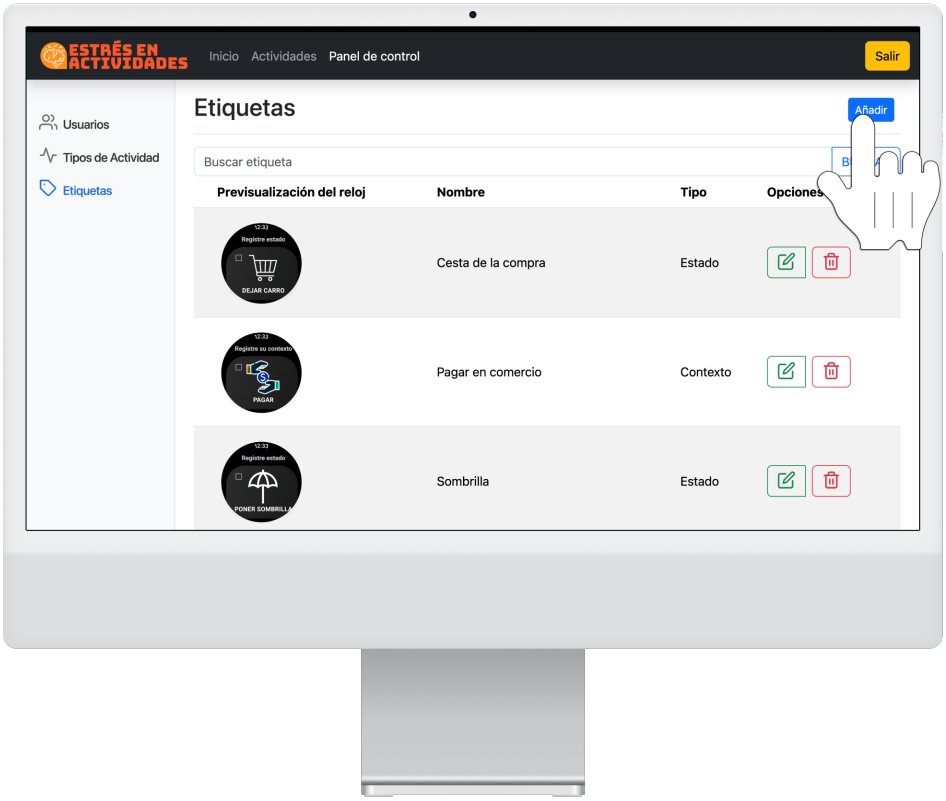


Figura D.27: Botón para añadir una etiqueta nueva al sistema.

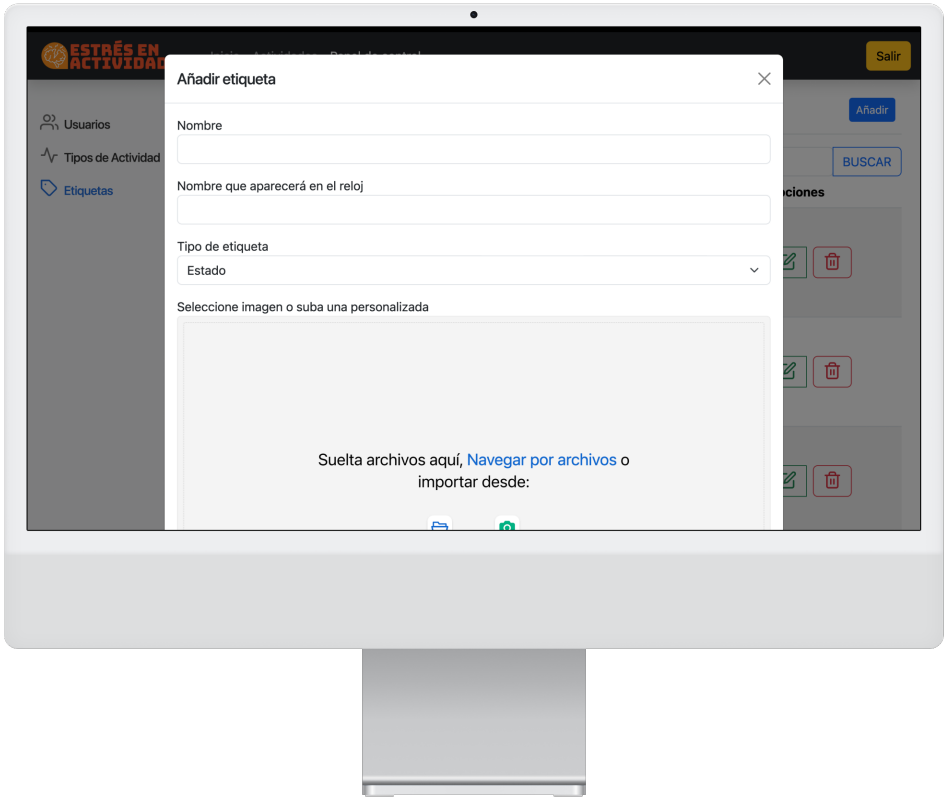


Figura D.28: Formulario para añadir una nueva etiqueta.

D.2.6. Visualización de actividades

Para la visualización de las actividades realizadas por los usuarios registrados en el sistema debemos dirigirnos a la sección “Actividades” de la barra superior de navegación, pulsando sobre el enlace “Actividades”. Tras esto, se muestra una página donde se pueden ver en una tabla los usuarios registrados en el sistema para los que podemos ver actividades realizadas y su perfil de estrés. Se proporciona además un buscador para facilitar la búsqueda de usuarios en situaciones en las que existan numerosos usuarios registrados. Se puede ver esta página en la figura D.29

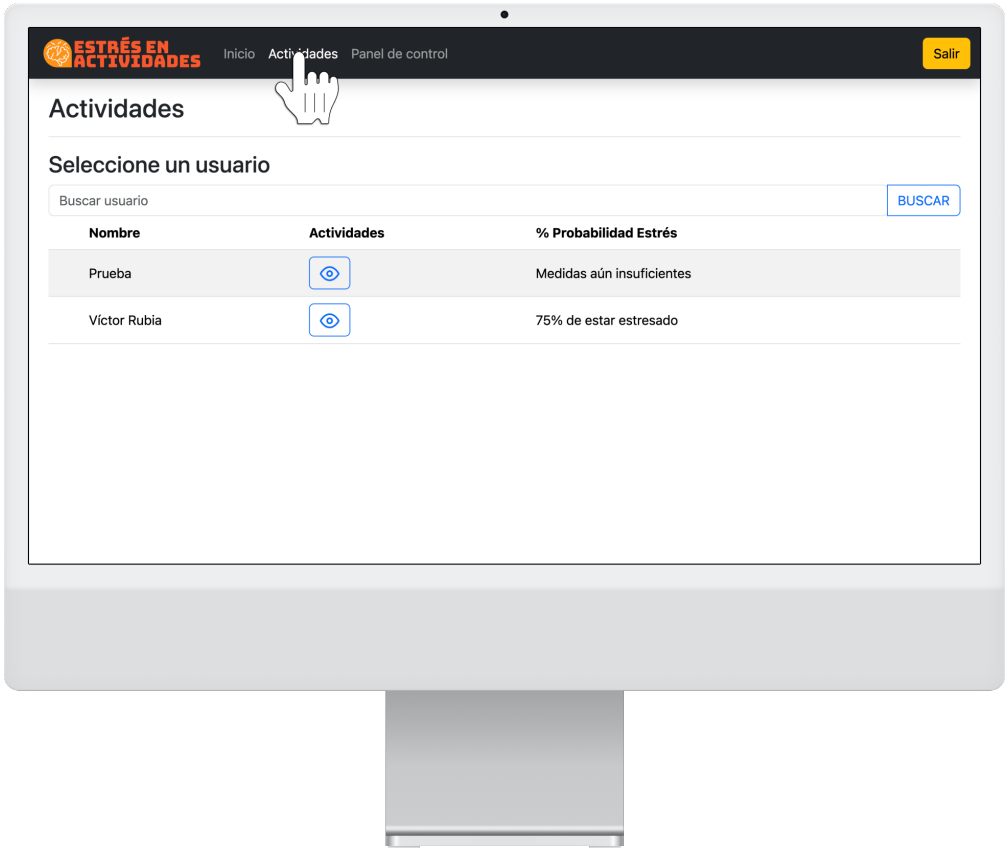


Figura D.29: Sección de selección de usuarios para ver el detalle de actividades realizadas.

Visualización de actividades para un usuario

Para visualizar las actividades que ha realizado un usuario pulsaremos sobre el botón de color azul indicado mediante un ojo bajo la columna “Actividades” de la tabla para la fila donde se encuentre el usuario para el que queremos visualizar las actividades, como se puede ver en la figura D.30. Tras esto, se muestra en una lista los tipos de actividades realizadas por el usuario, como se ve en la figura D.31.

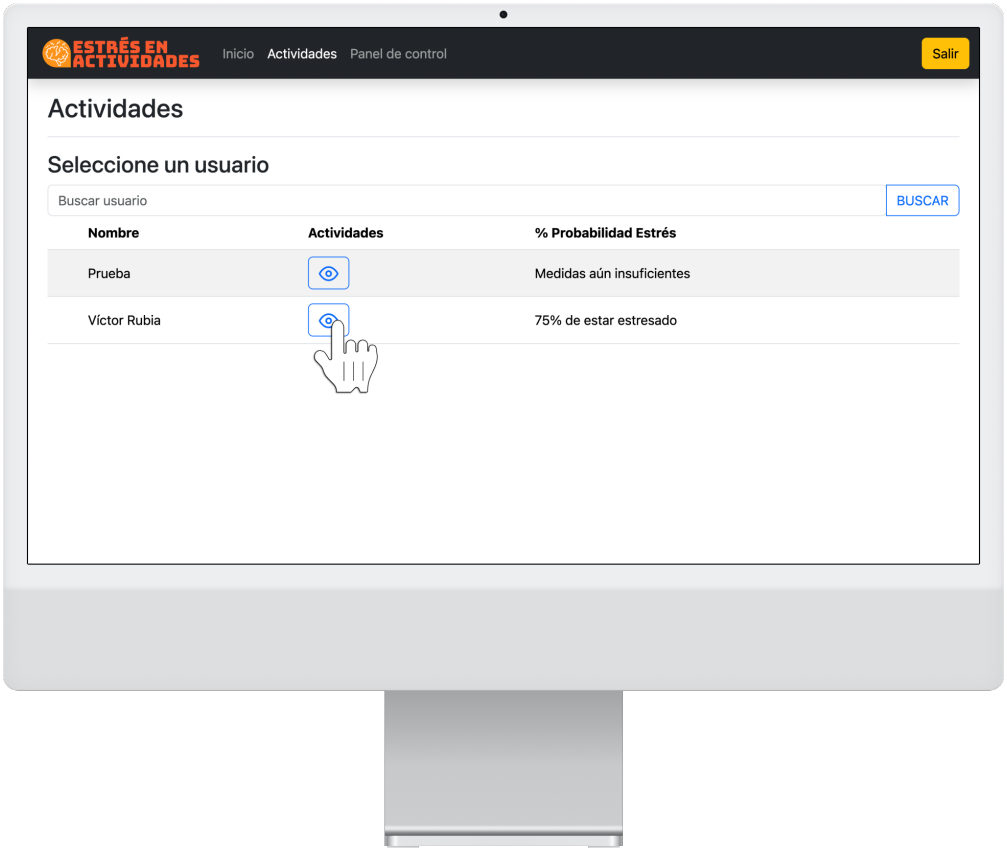


Figura D.30: Sección de ver las mediciones de actividades en la plataforma

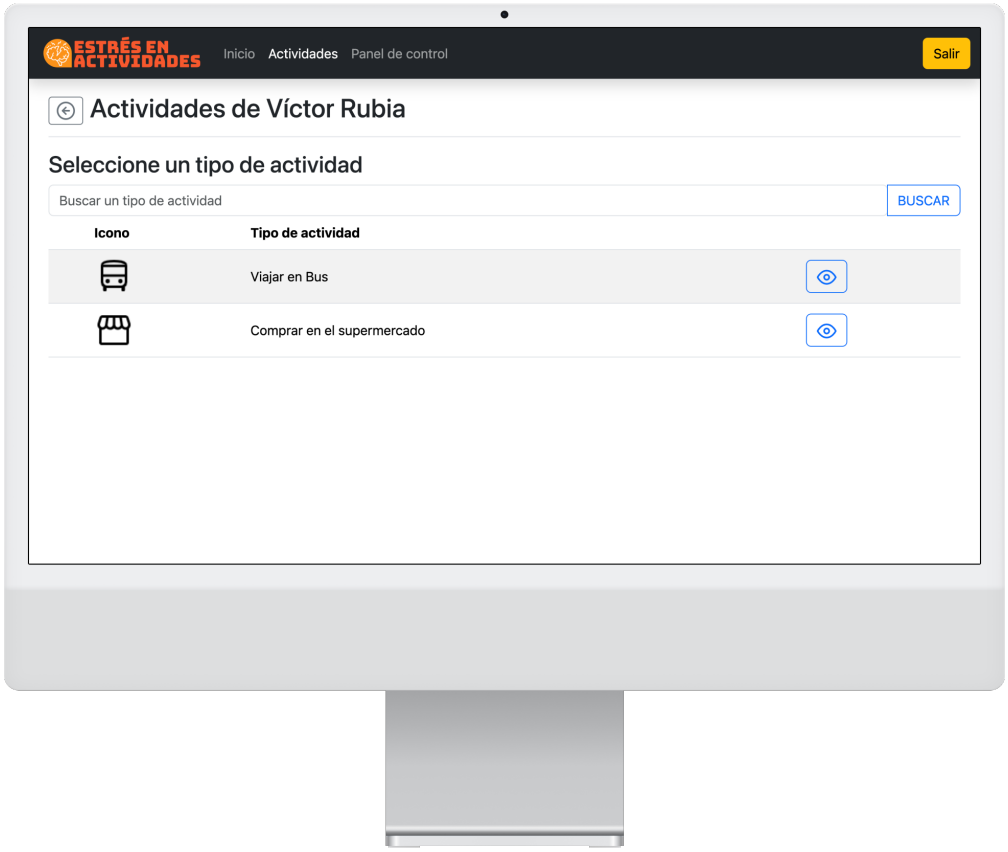


Figura D.31: Página de visualización de las actividades de un usuario.

Mediciones para actividades realizadas por un usuario

En esta página se muestran las mediciones tomadas para la actividad seleccionada según el punto anterior por el usuario ordenadas por fecha de finalización descendente e indicando dicha fecha de finalización en la parte baja de la tarjeta de actividad. Para eliminar una actividad registrada en el sistema basta con pulsar sobre el botón rojo etiquetado por "Eliminar" que se sitúa en la tarjeta de la actividad deseada, como se muestra en la figura D.32. Aparecerá una ventana emergente para confirmar la eliminación de la actividad seleccionada.

Si se quiere visualizar el procesamiento de la actividad realizada por el usuario se debe pulsar sobre el botón azul etiquetado por "Ver detalles" situado en la tarjeta de la actividad deseada, como en la figura D.32. Tras esto, se muestra una página como en la figura D.33, en la que se muestra en la parte superior un botón para volver atrás, junto con el tipo de actividad y un botón para extraer los datos recabados del sensor fisiológico en crudo en formato .CSV y un botón para volver hacer el procesamiento en caso de que el terapeuta detecte anomalías. En el caso en el que la actividad esté transcurriendo en directo, se muestra un indicador junto al tipo de actividad que se detalla. Tras esto, se muestra el nombre del usuario al que pertenece esta actividad junto con el gráfico interactivo con los niveles de estrés procesados a lo largo del tiempo por el que ha transcurrido la actividad y las etiquetas registradas. Por último, se muestra un mapa con la ruta seguida por el usuario junto a una tabla con las etiquetas registradas por el usuario en el transcurso de la actividad junto al instante de tiempo. De este modo podemos unir los registros de etiquetas con la línea de tiempo del gráfico de niveles de estrés de la actividad.

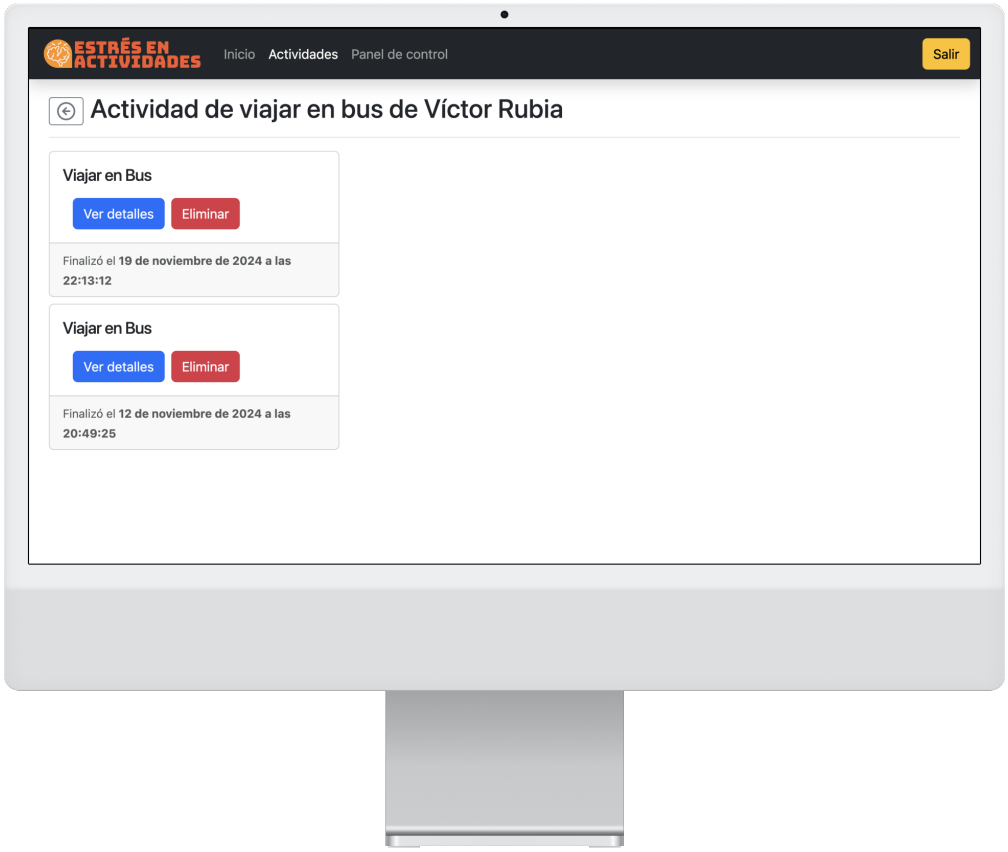


Figura D.32: Página para ver las mediciones registradas para una actividad de un usuario.

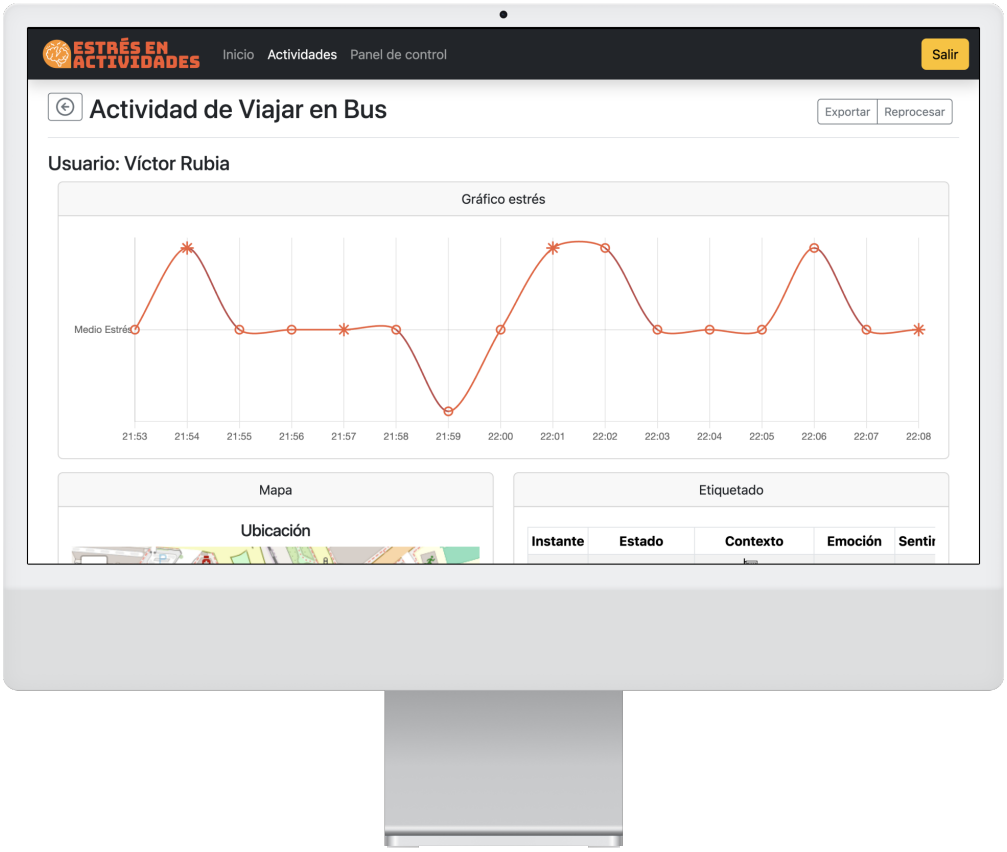


Figura D.33: Detalle de una medición de una actividad que ha realizado un usuario.

D.3. Reloj inteligente

Al iniciar la aplicación del reloj, se muestra una pantalla como la visible en la figura D.34, que obtiene el usuario identificado tras unos segundos necesarios de conexión con el teléfono móvil. Cuando lo recibe, pasa a la pantalla inicial de "Iniciar actividad". Si se da la situación de que en el teléfono no existe un usuario identificado, el reloj avisará con una pantalla como la que se muestra en la figura D.35, pudiendo indicar cuando el usuario haya iniciado sesión en el móvil que vuelva a iniciarse la aplicación pulsando sobre el botón "Listo". En el caso en el que el reloj no esté conectado al teléfono, se avisará con una pantalla como la que se muestra en la figura D.36, para que cuando se haya conectado el usuario el reloj al teléfono pueda pulsar sobre el botón "Reintentarlo". Por último se podría mostrar una pantalla a modo de aviso indicando que la aplicación no es compatible con el modelo de reloj en el que se ha instalado, como aparece en la figura D.37 y no pudiendo utilizar la aplicación.

Figura D.34: Pantalla tras el inicio de sesión del usuario sin conexión al reloj inteligente.

Figura D.35: Pantalla de aviso de usuario no identificado en el teléfono.

Figura D.36: Pantalla de aviso cuando el reloj no se encuentra conectado al móvil.

Figura D.37: Pantalla de aviso cuando el modelo de reloj no es compatible con la aplicación.

D.3.1. Iniciar actividad

Para que un usuario inicie la medición de una actividad se debe pulsar sobre el botón "Comenzar actividad" situado en primer plano de la pantalla, como se muestra en la figura D.38. Antes de comenzar la actividad, el usuario debe elegir a qué categoría pertenece el tipo de actividad, de entre las que tiene asignadas, que va realizar de una lista que se muestra por pantalla mediante el toque sobre el ítem de la lista que más se ajuste, como se muestra en la figura D.39. Tras esto se debe visualizar una pantalla de aviso que confirme el comienzo de la medición como se muestra en la figura D.40. Una vez hecho esto, se debe visualizar el menú de medición, representado en la figura D.41, en la cual tenemos la opción de detener la actividad y de registrar etiquetas.

Figura D.38: Pantalla inicial para el inicio de la medición de una actividad.

Figura D.39: Pantalla de selección del tipo de actividad de una lista.

Figura D.40: Pantalla de aviso del comienzo de la medición para la actividad iniciada.

Figura D.41: Pantalla con un menú para parar la actividad y para registrar etiquetas.

D.3.2. Registrar etiquetas

Para poder registrar etiquetas en el transcurso de una actividad debemos pulsar sobre el botón “Registrar estado”, como se muestra en la figura D.42 para visualizar un listado con las siguientes etiquetas clasificadas en las siguientes categorías.

Estados

Dinámicos en función de la actividad a realizar.

Contextos

Dinámicos en función de la actividad a realizar.

Emociones

- Felicidad.
- Tristeza.
- Miedo.
- Ira.
- Sorpresa.
- Asco.

Sentimientos

- Nervioso.
- Tranquilo.

Cada categoría esta separada en una pantalla dedicada con un listado que contiene las etiquetas anteriormente expuestas con iconos ilustrativos y un texto que los describe, se muestra un ejemplo de cómo se ve la pantalla cuando se pulsa sobre "Registrar estado" en la figura D.43. Para seleccionar una etiqueta se debe pulsar sobre el recuadro que lo engloba, de modo que se vea, cuando está seleccionada, un fondo más claro y un icono de verificación verde en el borde superior izquierdo, como se muestra en la figura D.44. En cada pantalla, se muestra al final un botón para confirmar el registro de etiquetas asociadas a la categoría en la que se sitúe el usuario, como se muestra en la figura D.45. Cuando se presiona este botón, se muestra durante unos segundos una pantalla explicativa para indicar al usuario qué categoría se va a mostrar a continuación. En el momento en el que el usuario termina de pasar por todas las categorías de etiquetas, se vuelve a la pantalla mostrada en la figura D.41.

Figura D.42: Pantalla de menú de medición de actividad.

Figura D.43: Pantalla con el listado de etiquetas pertenecientes a la categoría Estados.

Figura D.44: Pantalla con una etiqueta seleccionada del listado.

Figura D.45: Botón al final del listado de etiquetas para confirmar la selección.

D.3.3. Detener una actividad en transcurso

Para detener una actividad que se encuentra iniciada en el reloj, se debe pulsar sobre el botón circular rojo indicado mediante un cuadrado en la parte inferior del menú de medición y etiquetado por "Parar medición". Se muestra en la figura D.46 la forma en la que detener una actividad. Tras esto, se vuelve a la pantalla D.38.

Figura D.46: Pantalla de medición donde se muestra el botón para detener la medición de la actividad.

De igual forma, se puede parar la actividad desde el área de notificaciones del reloj cuando la aplicación está midiendo en segundo plano. Asímismo, se puede recuperar la interfaz de la aplicación para registrar algún estado. Esto se muestra en las Figuras D.47

y D.48.

Figura D.47: Área de notificaciones cuando la aplicación se ejecuta en segundo plano.

Figura D.48: Acción desde las notificaciones para parar la medición de la actividad sin reanudar la aplicación.

Apéndice E

Documentacion

Tras comentar todo el código realizado para las plataformas Android y WearOS, se ha usado un motor de generación de documentación de APIs en formato HTML llamado Dokka. La generación de estos HTML se puede consultar en el repositorio git tanto para Android como para WearOS. El código completo del proyecto se encuentra alojado en el repositorio público del enlace a **GitHub**.

Para consultar la función como servicio (FaaS) que sirve el modelo de SWELL y WESAD combinados, se da enlace a **DockerHub**. Igualmente para el modelo basado en el DASS-21 y los datos recogidos en nuestra experimentación puede consultarse en el enlace a **DockerHub**. Para los datos subyacentes del modelo del DASS-21 y los datos de la experimentación, pueden descargarse desde **Kaggle**.

Bibliografía

Agencia Tributaria. (2020). Coeficientes de amortización lineal. https://www3.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/requisitos-generales/coeficientes-amortizacion-lineal.html

Ahn, J. W., Ku, Y., & Kim, H. C. (2019). A Novel Wearable EEG and ECG Recording System for Stress Assessment [s19091991[PII]]. *Sensors (Basel, Switzerland)*, *19*(9), 1991. https://doi.org/10.3390/s19091991

Apple. (2024). Human Interface Guidelines: Text Fields. https://developer.apple.com/design/human-interface-guidelines/text-fields

Bank, W. (2024). Distribución porcentual de la población de España de 2005 a 2023, por rango de edad. https://es.statista.com/estadisticas/501064/poblacion-de-espana-por-grupo-de-edad/

Benchekroun, M., Chevallier, B., Beaouiss, H., Istrate, D., Zalc, V., Khalil, M., & Lenne, D. (2022). Comparison of Stress Detection through ECG and PPG signals using a Random Forest-based Algorithm. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 3150-3153. https://doi.org/10.1109/EMBC48229.2022.9870984

Biørn-Hansen, A., Rieger, C., Grønli, T.-M., Majchrzak, T. A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, *25*(4), 2997-3040. https://doi.org/10.1007/s10664-020-09827-6

BOE. (2018). *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. https://www.boe.es/diario_boe/txt.php?id=BOE-A-2018-16673

Boletín Oficial del Estado. (2003). Ley 51/2003, de 2 de diciembre, de igualdad de oportunidades, no discriminación y accesibilidad universal de las personas con discapacidad. https://www.boe.es/eli/es/l/2003/12/02/51/con

Boletín Oficial del Estado. (2022). Ley 6/2022, de 31 de marzo, de modificación del Texto Refundido de la Ley General de derechos de las personas con discapacidad y de su inclusión social, aprobado por el Real Decreto Legislativo 1/2013, de 29 de noviembre, para establecer y regular la accesibilidad cognitiva y sus condiciones de exigencia y aplicación. https://www.boe.es/eli/es/l/2022/03/31/6

Burns, E., & Buttner, R. (2021). T wave. https://litfl.com/t-wave-ecg-library/

Cadogan, M., & Buttner, R. (2022). P wave. https://litfl.com/p-wave-ecg-library/

Carstensen, L. L. (2019). Integrating cognitive and emotion paradigms to address the paradox of aging [PMID: 30394173]. *Cognition and Emotion*, *33*(1), 119-125. https://doi.org/10.1080/02699931.2018.1543181

Chen, J., Abbod, M., & Shieh, J.-S. (2021). Pain and stress detection using wearable sensors and devices—A review. *Sensors*, *21*(4), 1030.

Cherry, K. (2024). The 6 types of basic emotions and their effect on human behavior. https://www.verywellmind.com/an-overview-of-the-types-of-emotions-4163976

Community, F. (2022). Wear: Flutter Package. https://pub.dev/packages/wear

Cong, R., & Winters, R. (2017). How does the xbox kinect work. https://www.jameco.com/jameco/workshop/howitworks/xboxkinect.html

Dahal, K., Bogue-Jimenez, B., & Doblas, A. (2023). Global Stress Detection Framework Combining a Reduced Set of HRV Features and Random Forest Model. *Sensors*, *23*, 5220. https://doi.org/10.3390/s23115220

Dang, A. T. (2020). MVC VS MVP VS MVVM. https://levelup.gitconnected.com/mvc-vs-mvp-vs-mvvm-35e0d4b933b4

Dependency injection in Android. (s.f.). https://developer.android.com/training/dependency-injection

Developers, G. (2024). MessagesClient - Nearby Messages API Reference. https://developers.google.com/android/reference/com/google/android/gms/nearby/messages/MessagesClient

digitalAI. (s.f.). Agile velocity. https://digital.ai/glossary/agile-velocity

Escalona, S. A., & Basto, Y. C. A. (2024). Impacto de las tecnologías portátiles en la salud del paciente [Fecha de recepción: 22/07/2024; Fecha de aceptación: 19/08/2024]. *Ocronos*, *7*(8), 1137. https://revistamedica.com/impacto-tecnologias-portatiles-salud-paciente/

Eskandar, S., & Razavi, S. (2020). Using Deep Learning for Assessment of Workers' Stress and Overload. *Proceedings of the 37th International Symposium on Automation and Robotics in Construction (ISARC)*, 872-877.

Europea, C. (2021). Una Unión de la Igualdad: Estrategia sobre los derechos de las personas con discapacidad para 2021-2030 [COM(2021) 101 final]. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2021:101:FIN

Europea, C., & de Justicia y Consumidores, D. G. (2018). *RGPD : nuevas oportunidades, nuevas obligaciones : lo que las empresas tienen que saber sobre el Reglamento general de protección de datos de la Unión Europea*. Oficina de Publicaciones. https://doi.org/doi/10.2838/351308

Feathericons. (2022). FeatherIcons. https://github.com/feathericons/feather

fluttercommunity.dev. (2021). Wear Flutter Package. https://pub.dev/packages/wear

fluttercommunity.dev. (2024). $Sensors_plus : Flutter Package$. https://pub.dev/packages/sensors_plus

Gandomani, T. J., Faraji, H., & Radnejad, M. (2019). Planning Poker in cost estimation in Agile methods: Averaging Vs. Consensus. *2019 5th Conference on Knowledge*

Based Engineering and Innovation (KBEI), 066-071. https://doi.org/10.1109/KBEI.2019.8734960

García Martínez, M. (2017). *Las emociones y el bienestar en las personas mayores* (Tesis doctoral).

García Martínez, M., Palmero Cantero, F., & Universitat Jaume I. Departament de Psicologia Bàsica, C. i. P. (2017). Las emociones y el bienestar en las personas mayores [Bachelor's thesis, Universitat Jaume I, 2017]. http://hdl.handle.net/10234/169492

García-Martínez, B., Fernández-Caballero, A., Martínez-Rodrigo, A., & Novais, P. (2021). Analysis of Electroencephalographic Signals from a Brain-Computer Interface for Emotions Detection, 219-229. https://doi.org/10.1007/978-3-030-85030-2_18

Glassdoor. (2024). Sueldo: Junior software developer. https://www.glassdoor.es/Sueldos/junior-software-developer-sueldo-SRCH_KO0,25.htm

Google. (s.f.-a). Design principles WearOS. https://developer.android.com/training/wearables/design/design-principles

Google. (s.f.-b). Enfoque de Prioridad de Kotlin en Android. https://developer.android.com/kotlin/first?hl=es-419

Google. (s.f.-c). Why compose: Jetpack Compose. https://developer.android.com/jetpack/compose/why-adopt

Google. (2014). Material Design Guidelines. https://material.io/design/introduction

Google. (2024a). Accessibility Material Design. https://material.io/archive/guidelines/usability/accessibility.html#accessibility-hierarchy-focus

Google. (2024b). How do I track and manage stress with my Fitbit device? https://support.google.com/fitbit/answer/14237928

Google. (2024c). Material icons. https://fonts.google.com/icons

Google. (2024d). Wear OS Design Principles: Foundations. https://developer.android.google.cn/design/ui/wear/guides/foundations/design-principles

Gradl, S., Wirth, M., Richer, R., Rohleder, N., & Eskofier, B. M. (2019). An Overview of the Feasibility of Permanent, Real-Time, Unobtrusive Stress Measurement with Current Wearables. *Proceedings of the 13th EAI International Conference on Pervasive Computing Technologies for Healthcare*, 360-365. https://doi.org/10.1145/3329189.3329233

Greene, S., Thapliyal, H., & Caban-Holt, A. (2016). A Survey of Affective Computing for Stress Detection: Evaluating technologies in stress detection for better health. *IEEE Consumer Electronics Magazine*, *5*(4), 44-56. https://doi.org/10.1109/MCE.2016.2590178

Guide to app architecture. (s.f.). https://developer.android.com/topic/architecture

Guragai, B., Pal, R., Patel, P., Li, J., Heyat, M. B. B., & Akhtar, F. (2020a). Role of Machine Learning in Human Stress: A Review. https://doi.org/10.1109/ICCWAMTIP51612.2020.9317396

Guragai, B., Pal, R., Patel, P., Li, J., Heyat, M. B. B., & Akhtar, F. (2020b). Role of Machine Learning in Human Stress: A Review. *Proceedings of the 17th International Computer Conference on Wavelet Active Media Technology and Information*

Processing (ICCWAMTIP). https://doi.org/10.1109/ICCWAMTIP51612.2020.9317396

Halim, N., Sidek, K., & Mansor, H. (2017). Stress Recognition Using Photoplethysmogram Signal. *Indonesian Journal of Electrical Engineering and Computer Science*, *8*, 495-501. https://doi.org/10.11591/ijeecs.v8.i2.pp495-501

Hansson, E., Mattisson, K., Björk, J., Östergren, P.-O., & Jakobsson, K. (2011). Relationship between commuting and health outcomes in a cross-sectional population survey in southern Sweden. *BMC Public Health*, *11*(1), 834. https://doi.org/10.1186/1471-2458-11-834

Heo, S., Kwon, S., & Lee, J. (2021a). Stress Detection With Single PPG Sensor by Orchestrating Multiple Denoising and Peak-Detecting Methods. *IEEE Access*, *9*, 47777-47785. https://doi.org/10.1109/ACCESS.2021.3060441

Heo, S., Kwon, S., & Lee, J. (2021b). Stress Detection With Single PPG Sensor by Orchestrating Multiple Denoising and Peak-Detecting Methods. *IEEE Access*, *PP*, 1-1. https://doi.org/10.1109/ACCESS.2021.3060441

Hernando, D., Roca, S., Sancho, J., Alesanco, Á., & Bailón, R. (2018). Validation of the Apple Watch for Heart Rate Variability Measurements during Relax and Mental Stress in Healthy Subjects. *Sensors*, *18*(8). https://doi.org/10.3390/s18082619

Indeed. (2024). Sueldo de Programador/a junior en España.

Indikawati, F. I., & Winiarti, S. (2020). Stress Detection from Multimodal Wearable Sensor Data. *IOP Conference Series: Materials Science and Engineering*, *771*(1), 012028. https://dx.doi.org/10.1088/1757-899X/771/1/012028

INE. (2017). Encuesta Nacional de Salud ENSE. https://www.sanidad.gob.es/estadEstudios/estadisticas/encuestaNacional/encuestaNac2017/SALUD_MENTAL.pdf

Iqbal, T., Redon-Lurbe, P., Simpkin, A. J., Elahi, A., Ganly, S., Wijns, W., & Shahzad, A. (2021). A Sensitivity Analysis of Biophysiological Responses of Stress for Wearable Sensors in Connected Health. *IEEE Access*, *9*, 93567-93579. https://doi.org/10.1109/ACCESS.2021.3082423

Jerath, R., Syam, M., & Ahmed, S. (2023). The Future of Stress Management: Integration of Smartwatches and HRV Technology. *Sensors*, *23*(17). https://doi.org/10.3390/s23177314

Jiménez-Limas, M. A., Ramírez-Fuentes, C. A., Tovar-Corona, B., & Garay-Jiménez, L. I. (2018). Feature selection for stress level classification into a physiologycal signals set. *2018 15th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 1-5. https://doi.org/10.1109/ICEEE.2018.8533968

Kane, A. (s.f.). Lockbox. https://github.com/ankane/lockbox

Khan, N., & Sarkar, N. (2022). Semi-Supervised Generative Adversarial Network for Stress Detection Using Partially Labeled Physiological Data. https://arxiv.org/abs/2206.14976

Kluger, A. N. (1998). Commute variability and strain. *Journal of Organizational Behavior*, *19*(2), 147-165. https://doi.org/https://doi.org/10.1002/(SICI)1099-1379(199803)19:2⟨147::AID-JOB830⟩3.0.CO;2-Y

Kohan, B. (2022). Guide to web application development. https://www.comentum.com/guide-to-web-application-development.html

Kołakowska, A., Szwoch, W., & Szwoch, M. (2020). A Review of Emotion Recognition Methods Based on Data Acquired via Smartphone Sensors. *Sensors*, *20*(21). https://doi.org/10.3390/s20216367

Koldijk, S., Neerincx, M. A., & Kraaij, W. (2018). Detecting Work Stress in Offices by Combining Unobtrusive Sensors. *IEEE Transactions on Affective Computing*, *9*(2), 227-239. https://doi.org/10.1109/TAFFC.2016.2610975

Labuschagne, I., Grace, C., Rendell, P., Terrett, G., & Heinrichs, M. (2019). An introductory guide to conducting the Trier Social Stress Test. *Neuroscience Biobehavioral Reviews*, *107*, 686-695. https://doi.org/https://doi.org/10.1016/j.neubiorev.2019.09.032

Lalani, S. (2022). MVC VS MVP VS MVVM : 10 differences you should know. https://xperti.io/blogs/mvc-vs-mvp-vs-mvvm/

Larkin, J. (2021). QRS interval. https://litfl.com/qrs-interval-ecg-library/

Li, F., Xu, P., Zheng, S., Chen, W., Yan, Y., Lu, S., & Liu, Z. (2018). Photoplethysmography based psychological stress detection with pulse rate variability feature differences and elastic net. *International Journal of Distributed Sensor Networks*, *14*(9), 1550147718803298. https://doi.org/10.1177/1550147718803298

Li, R., & Liu, Z. (2020). Stress detection using deep neural networks. *BMC Medical Informatics and Decision Making*, *20*(11), 285. https://doi.org/10.1186/s12911-020-01299-4

Liapis, A., Faliagka, E., Antonopoulos, C., Voros, N., & Keramidas, G. (2021). Advancing Stress Detection Methodology with Deep Learning Techniques Targeting UX Evaluation in AAL Scenarios: Applying Embeddings for Categorical Variables. *Electronics*, *10*. https://doi.org/10.3390/electronics10131550

Lovibond, P. F., & Lovibond, S. H. (1995). The structure of negative emotional states: comparison of the Depression Anxiety Stress Scales (DASS) with the Beck Depression and Anxiety Inventories. *Behaviour Research and Therapy*, *33*(3), 335-343. https://doi.org/10.1016/0005-7967(94)00075-u

Luján Henríquez, I. (2015a). Aspectos socio-emocionales y salud mental y física percibidas en personas mayores. *Revista INFAD de Psicología. International Journal of Developmental and Educational Psychology*, *2*(1), 81-90. https://doi.org/10.17060/ijodaep.2015.n1.v2.115

Luján Henríquez, I. (2015b). Aspectos socio-emocionales y salud mental y física percibidas en personas mayores. *Revista INFAD de Psicología. International Journal of Developmental and Educational Psychology.*, *2*(1), 81-90. https://doi.org/10.17060/ijodaep.2015.n1.v2.115

Luján Henríquez, I. (2015c). Aspectos socio-emocionales y salud mental y física percibidas en personas mayores. *Revista INFAD de Psicología. International Journal of Developmental and Educational Psychology.*, *2*(1), 81-90. https://doi.org/10.17060/ijodaep.2015.n1.v2.115

Mahendra, M., & Anggorojati, B. (2021). Evaluating the performance of Android based Cross-Platform App Development Frameworks. *Proceedings of the 6th International Conference on Communication and Information Processing*, 32-37. https://doi.org/10.1145/3442555.3442561

MariaDB Foundation. (2024). MariaDB Docker Official Image.

McCraty, R., & Shaffer, F. (2015). Heart Rate Variability: New Perspectives on Physiological Mechanisms, Assessment of Self-regulatory Capacity, and Health risk. *Glob Adv Health Med*, *4*(1), 46-61.

Mohan, P. M., Nagarajan, V., & Das, S. R. (2016). Stress measurement from wearable photoplethysmographic sensor using heart rate variability data. *2016 International Conference on Communication and Signal Processing (ICCSP)*, 1141-1144. https://doi.org/10.1109/ICCSP.2016.7754331

MongoDB. (s.f.). What is a non-relational database? https://www.mongodb.com/databases/non-relational

Ninh, V.-T., Smyth, S., Tran, M.-T., & Gurrin, C. (2022). Analysing the Performance of Stress Detection Models on Consumer-Grade Wearable Devices. https://arxiv.org/abs/2203.09669

Nkurikiyeyezu, K., Yokokubo, A., & Lopez, G. (2019). The Effect of Person-Specific Biometrics in Improving Generic Stress Predictive Models. https://arxiv.org/abs/1910.01770

on Rails API Documentation, R. (s.f.). ActiveModel::SecurePassword::ClassMethods. https://api.rubyonrails.org/classes/ActiveModel/SecurePassword/ClassMethods.html

on Rails Guides, R. (s.f.). SecurePassword. https://guides.rubyonrails.org/active_model_basics.html#securepassword

Oracle. (s.f.). What is a relational database? https://www.oracle.com/database/what-is-a-relational-database/

Osman, A., Wong, J. L., Bagge, C. L., Freedenthal, S., Gutierrez, P. M., & Lozano, G. (2012). The Depression Anxiety Stress Scales-21 (DASS-21): Further examination of dimensions, scale reliability, and correlates [Epub 2012 Aug 28]. *Journal of Clinical Psychology*, *68*(12), 1322-1338. https://doi.org/10.1002/jclp.21908

Paulvangentcom. (s.f.). Analysing smart ring data. https://github.com/paulvangentcom/heartrate_analysis_python/blob/master/examples/4_smartring_data/Analysing_Smart_Ring_Data.ipynb

Paulvangentcom. (2019). Analysing a ppg signal. https://github.com/paulvangentcom/heartrate_analysis_python/blob/master/examples/1_regular_PPG/Analysing_a_PPG_signal.ipynb

Paulvangentcom. (2021). Analysing smartwatch data. https://github.com/paulvangentcom/heartrate_analysis_python/blob/master/examples/3_smartwatch_data/Analysing_Smartwatch_Data.ipynb

Pawlan, D. (2022). Relational vs. Non-Relational Database: Pros amp; Cons. https://aloa.co/blog/relational-vs-non-relational-database-pros-cons

PJ, S. (2021). Swell dataset analysis for stress prediction. https://www.kaggle.com/code/shreyaspj/swell-dataset-analysis-for-stress-prediction

Polito, S. (2021). Build an API in rails with authentication. https://stevepolito.design/blog/build-an-api-in-rails-with-authentication/

Ramírez Loeffler, P., & Rodríguez Vílchez, Z. (2012). *GESTIÓN Y CONTROL DEL ESTRÉS* (1.ª ed.). Conzepto Comunicación Creativa.

Rubia López, V. J. (2022). *Control de estrés en mayores: Aplicación para SmartWatch* (Trabajo Fin de Grado) [Directoras: María José Rodríguez Fórtiz y María Luisa Rodríguez Almendros]. Universidad de Granada. Granada, España.

Saeed, S., Jhanjhi, N. Z., Naqvi, M. R., & Humayun, M. (2019). Analysis of Software Development Methodologies. *International Journal of Computing and Digital Systems*. https://journal.uob.edu.bh/handle/123456789/3583

Safwany, O. (2018). Web Application Deployment: Meaning. https://www.quora.com/What-does-the-deployment-of-a-web-application-mean-in-software-development-and-how-can-it-be-done

Samsung. (s.f.). Measure your stress level with Samsung Health. https://www.samsung.com/us/support/answer/ANS00080574/

Schmidt, P., Reiss, A., Duerichen, R., Marberger, C., & Van Laerhoven, K. (2018). Introducing WESAD, a Multimodal Dataset for Wearable Stress and Affect Detection. *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 400-408. https://doi.org/10.1145/3242969.3242985

Schwartz, A. (s.f.). The impact of small stresses in daily life. https://www.mentalhelp.net/blogs/the-impact-of-small-stresses-in-daily-life/

Shaffer, F., & Ginsberg, J. P. (2017). An Overview of Heart Rate Variability Metrics and Norms. *Front Public Health*, *5*, 258.

Shekh-Yusef, R., Ahrens, D., & Bremer, S. (2015). HTTP Digest Access Authentication. https://doi.org/10.17487/RFC7616

Siirtola, P., & Röning, J. (2020). Comparison of Regression and Classification Models for User-Independent and Personal Stress Detection. *Sensors (Basel)*, *20*(16), 4402. https://doi.org/10.3390/s20164402

Sriramprakash, S., Prasanna, V. D., & Murthy, O. R. (2017). Stress Detection in Working People [7th International Conference on Advances in Computing Communications, ICACC-2017, 22-24 August 2017, Cochin, India]. *Procedia Computer Science*, *115*, 359-366. https://doi.org/https://doi.org/10.1016/j.procs.2017.09.090

S.S, P., & P, G. (2019). A survey of machine learning techniques in physiology based mental stress detection systems. *Biocybernetics and Biomedical Engineering*, *39*(2), 444-469. https://doi.org/10.1016/j.bbe.2019.01.004

StackOverflow. (2022). StackOverflow Developer Survey 2022. https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-databases

Statista. (2021). Número de habitantes de 65 años o más en España de 2002 a 2021 (en millones). https://es.statista.com/estadisticas/630678/poblacion-de-espana-mayor-de-65-anos/

Statista. (2024). Wearable healthcare devices market worldwide 2020-2029, by region [Published by Conor Stewart, Apr 16, 2024]. https://www.statista.com/statistics/1461412/wearable-healthcare-devices-market-trend-worldwide-by-region/

Stults-Kolehmainen, M. A., & Sinha, R. (2014). The Effects of Stress on Physical Activity and Exercise. *Sports Medicine*, *44*(1), 81-121. https://doi.org/10.1007/s40279-013-0090-5

Stuyck, H., Dalla Costa, L., Cleeremans, A., & Van den Bussche, E. (2022). Validity of the Empatica E4 wristband to estimate resting-state heart rate variability in a lab-based context [Epub 2022 Oct 14]. *International Journal of Psychophysiology*, *182*, 105-118. https://doi.org/10.1016/j.ijpsycho.2022.10.003

Subdirección General de Información Sanitaria. (2021). Salud mental en datos: prevalencia de los problemas de salud y consumo de psicofármacos y fármacos relacionados a partir de los registros clínicos de atención primaria [Informe basado en datos de 2017, publicado en diciembre de 2020]. https://www.sanidad.gob.es/estadEstudios/estadisticas/estadisticas/estMinisterio/SIAP/Salud_mental_datos.pdf

Subhani, A. R., Xia, L., & Malik, A. S. (s.f.). EEG Signals to Measure Mental Stress.

Tena, M. J. F., del Carmen Ortega Navas, M., & Reis, C. S. (2019). El envejecimiento activo y la inteligencia emocional en las personas mayores. *Familia. Revista de Ciencias y Orientación Familiar*, (57), 125-137. http://summa.upsa.es/viewer.vm?id=ris&page=1

Torrades, S. (2007). Estrés y burn out. Definición y prevención. *Offarm*, *26*(10), 104-107. https://www.elsevier.es/es-revista-offarm-4-articulo-estres-burn-out-definicion-prevencion-13112896

Twbs. (2022). Bootstrap. https://github.com/twbs/bootstrap

United Nations Economic Commission for Europe. (2009). Integration and participation of older persons in society. https://unece.org/DAM/pau/_docs/age/2009/Policy_briefs/4-Policybrief_Participation_Eng.pdf

van Gent, P. (2019). HeartPy - Python Heart Rate Analysis Toolkit.

Vos, G., Trinh, K., Sarnyai, Z., & Rahimi Azghadi, M. (2023). Ensemble machine learning model trained on a new synthesized dataset generalizes well for stress prediction using wearable devices. *Journal of Biomedical Informatics*, *148*, 104556. https://doi.org/10.1016/j.jbi.2023.104556

What are containers? (2022). https://www.netapp.com/devops-solutions/what-are-containers

Wijsman, J., Grundlehner, B., Penders, J., & Hermens, H. (2010). Trapezius Muscle EMG as Predictor of Mental Stress. *ACM Transactions on Embedded Computing Systems (TECS)*, *12*, 155-163. https://doi.org/10.1145/2485984.2485987

Winer, D. (2019). Android's commitment to Kotlin. https://android-developers.googleblog.com/2019/12/androids-commitment-to-kotlin.html

Woodhead, W. (2018). Should you use material design? https://medium.com/pilcro/should-you-use-material-design-bfb596a04bae

Xiaomi. (2024). Xiaomi Smart Band 8 FAQ [Punto 20: Does the Xiaomi Smart Band 8 support stress detection?]. https://www.mi.com/global/support/faq/details/KA-231471

Yáñez-Yáñez, R., & Draguicevic, N. M. A. (2021). Personas mayores y su incorporaciÃa las nuevas tecnologÃas, muestra de su resiliencia y derrota de estereotipos viejistas. *Revista mÃde Chile*, *149*, 1097-1098. http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0034-98872021000701097&nrm=iso

Yiğit, M. (2022). Say hello to jetpack compose and compare with XML. https://blog.kotlin-academy.com/say-hello-to-jetpack-compose-and-compare-with-xml-6bc6053aec13