Este proyecto trata el desarrollo software realizado sobre un gestor de almacenamiento de componentes electrónicos utilizado en un laboratorio de la Universidad de Granada. Este gestor debido a falta de tiempo y otros factores quedó incompleto, el reto que se plantea es la actualización del gestor antiguo: mejorar el funcionamiento sobre las funcionalidades ya existentes y añadir nuevas a demanda del cliente.

El sistema se encuentra implementado en NodeJS y MySQL, utilizándose en el ladodel cliente ReactJS. El resultado que se ha tratado conseguir ha sido un nuevo sistema completo y funcional, con una interfaz de usuario sencilla de utilizar, visualmente agradable y útil. Del mismo modo se ha actualizado una extensión existente para Google Chrome que permite adquirir de una forma rápida y sencilla los datos de los componentes electrónicos desde las páginas de los proveedores.

Jesús Rodríguez Pérez estudiante procedente de Granada, que finaliza con este proyecto sus estudios en el Grado enIngeniería Informática en la Universidad de Granada.

Andrés Maria Roldán Aranda profesor de la Universidad de Granada del departamento de Electrónica y Tecnología de Computadores. Es el tutor o profesor a cargo de este Trabajo de Fin de Grado.

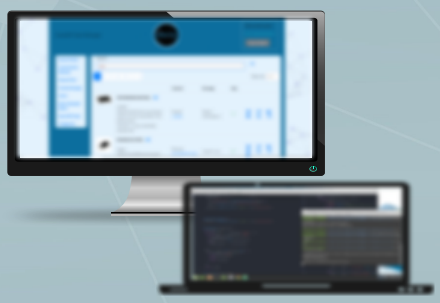
Trabajo de Fin de Grado

Software de gestión de inventario para componentes electrónicos

Jesús Rodríguez Pérez

Ingeniería Informática

Universidad de Granada
Grado en Ingeniería Informática



Trabajo de Fin de Grado

Software de gestión de inventario para componentes electrónicos

Jesús Rodríguez Pérez

2019/2020

Tutor: Andrés María Roldán Aranda

TRABAJO FIN DE GRADO

INGENIERÍA EN INFORMÁTICA

Software de gestión de inventario para componentes electrónicos

Autor
Jesús Rodríguez Pérez

Director
Andrés Roldan Aranda

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, septiembre de 2020

Software de gestión de inventario para componentes electrónicos

Jesús Rodríguez Pérez

Palabras clave: servicio web, desarrollo software, Javascript, gestor de almacenamiento

Resumen

Este proyecto trata el desarrollo software realizado sobre un gestor de almacenamiento de componentes electrónicos utilizado por el grupo GranaSat de la Universidad de Granada. Este sistema antiguo se encuentra incompleto y con algún error, por lo que el reto que se plantea es mejorar el funcionamiento sobre las funcionalidades ya existentes y añadir nuevas a demanda del cliente.

El sistema se encuentra implementado en NodeJS y MySQL, utilizándose en el lado del cliente ReactJS. Aunque se da la posibilidad de utilizar las tecnología que se estime oportuna, tras un estudio de las posibles opciones, se ha decidido continuar con esta ya que ha parecido bastante apropiada. El resultado que se ha tratado conseguir ha sido un nuevo sistema completo y funcional, con una interfaz de usuario sencilla de utilizar, visualmente agradable y útil.

Del mismo modo se ha actualizado una extensión existente para Google Chrome que permite adquirir de una forma rápida y sencilla los datos de los componentes electrónicos desde las páginas de los proveedores. Se han añadido más páginas de proveedores y arreglado o actualizado el código para los proveedores ya reconocidos por la extensión.

Inventory management software for electronic components

Jesús Rodríguez Pérez

Keywords: web service, software development, Javascript, inventory management

Abstract

This project aims to develop on an existing software that was used by the GranaSat group from the University of Granada. The antiquated system lacked some functionality and had some bugs that needed to be solved. New functionality was incorporated into the system, and the existing functionalities were greatly improved.

The back-end part of the system was implemented with NodeJs and MySQL. For the client side, ReactJS was used. The main goal of the project was the development of an updated and completed system with an usable and clear user interface. In addition, a Google Chrome extension was developed. This extension automatically extracts data from the vendors websites in a fast and simple way.

Yo, **Jesús Rodríguez Pérez**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 75941518A, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jesús Rodríguez Pérez

Granada a septiembre de mes de 2020 .

D. **Andrés Roldan Aranda (tutor)**, Profesor del Departamento Electrónica y Tecnología de Computadores de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Software de gestión de inventario para componentes electrónicos***, ha sido realizado bajo su supervisión por **Jesús Rodríguez Pérez**, 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 septiembre de mes de 2020

Los directores:

Índice general

Índice de figuras

Ťable Index

Glosario

Aplicación web Se entiende como aplicación web a aquellos programas o herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de internet o de una intranet mediante un navegador.

Back-end Este término refiere a la parte de una aplicación que se encuentra en el servidor, conocido como el lado del servidor. El back-end suele estar formado por servidor, aplicaciones y bases de datos.

Certificado HTTPS El certificado HTTPS es un fichero necesario para emplear el protocolo HTTPS, es emitido por las autoridades de certificación.

Código abierto La expresión código abierto o open-source se utiliza para denotar aquel software cuyo código se encuentra accesible, puede ser modificado y distribuido por cualquier persona de manera totalmente gratuita.

Código QR Un código QR es un código de barras bidimensional cuadrado que almacena datos codificados.

Diagrama de Gantt El diagrama de Gantt es una herramienta gráfica cuyo objetivo es exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

Diagrama de secuencia Diagrama que muestra la interacción de un conjunto de objetos de una aplicación a través del tiempo y se modela para cada caso de uso. El objetivo del diagrama es describir como se comporta el sistema para cada posible operación.

Framework Es un entorno de trabajo utilizado para facilitar el desarrollo y organización de un software determinado.

Front-end Término empleado en informática para referirse a la parte de una aplicación que interactúa con los usuarios, conocida como el lado del cliente. El front-end se encarga de la experiencia de usuario, lo que el usuario ve cómo puede ser la interfaz de usuario.

Github Plataforma de desarrollo colaborativo que utiliza un control de versiones llamado Git. Github es bastante popular para principalmente el desarrollo de código software, ya que facilita la colaboración y por su control de versiones .

Google Chrome Es un navegador web de código cerrado desarrollado por Google. `https://www.google.com/intl/es_es/chrome/`.

GranaSAT El proyecto GranaSAT es una actividad académica dirigida a los alumnos de ciencia e ingeniería de la Universidad de Granada que persigue el acercamiento al mundo aeroespacial y a las empresas del sector. Está coordinada por el Grupo de Electrónica Aeroespacial y actualmente desarrolla el proyecto de diseño y fabricación de un picosatélite universitario conocido como GranaSAT-I.

Listener En programación generalmente se llama listener a aquel método que se encarga de controlar un evento determinado, espera a que determinado evento suceda (un click, por ejemplo) y realiza una serie de acciones.

PartKeepr Gestor de inventario especializado en componentes electrónicos, de código abierto. Su código se encuentra disponible en `https://github.com/partkeepr/PartKeepr` y su página web es `https://partkeepr.org/`.

PartManager Nombre de la aplicación web desarrollada por el grupo GranaSat para la gestión de componentes electrónicos de laboratorio, y actualizada por Jesús Rodríguez Pérez (tal y como relata este documento).

Script Término informal utilizado en informática para referir a una secuencia de comandos, se podría traducir cómo un "'programa relativamente simple"'.

StockPile Gestor de inventario online gratuito (`http://www.thecanvus.com/#homePage`).

Token de sesión Un token de sesión dentro de la autenticación de usuarios, es una cadena de símbolos que permite identificar la sesión de un usuario de forma inequívoca y segura. Suele almacenarse en el lado del cliente (navegador).

Siglas

BSD Berkeley Software Distribution.

CSS Cascade Style Sheet.

HTML HyperText Markup Language.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

JSON JavaScript Object Notation.

UGR Universidad de Granada.

URL Uniform Resource Locator.

Capítulo 1

Introducción

En este documento se va a detallar cómo se ha desarrollado el software necesario para abordar un problema real, planteado por el grupo de GranaSAT (que será considerado como el cliente). En concreto el software que se va a desarrollar consiste en una aplicación web que será utilizada para la gestión del inventario de un almacén en un laboratorio de la Universidad de Granada (UGR), se tratará de que el resultado sea totalmente funcional y útil. Destacar que en este caso no se trabajará desde cero, si no que se dispondrá de un sistema anterior que ha estado siendo utilizado hasta ahora, aunque se encuentra incompleto: algunas de las funciones que implementa presentan errores y carece de algunas otras funciones necesarias (ver la sección 2.1.3 para más detalles acerca de que funciones se encuentran sin implementar).

Para cumplir este cometido, se va a hacer uso de los conocimientos adquiridos a lo largo del Grado en Ingeniería Informática. Aclarar que en este proyecto se va a tratar de:

- Utilizar tecnología actual y más adecuada.
- Aportar el máximo de soluciones posibles al problema que se plantea.
- Priorizar dentro de las posibles soluciones las que sean gratuitas, optando por siempre por alternativas de código abierto en los casos que sea posible, además se tendrá acceso a un servidor del propio laboratorio, el cual hospedaría a la aplicación web y facilita bastante este propósito.

1.1. Contexto y Motivación

1

El proyecto ha sido realizado en GranaSAT, un grupo académico de la UGR cuya actividad va dirigida a alumnos de Ciencias e Ingenierías . Este grupo, que persigue acercar a los alumnos al mundo aeroespacial y de la electrónica, fue concebido para el diseño y creación de un picosatélite universitario conocido como "GranaSAT-I". Aunque actualmente participa (además de en su principal cometido) en numerosos proyectos de diferentes temáticas dentro del sector, algunos en colaboración con empresas.

Figura 1.1 – *Logo actual de GranaSAT*

GranaSAT para cumplir con sus objetivos y proyectos debe utilizar una cantidad considerable de componentes electrónicos, y es de este importante requerimiento donde surge la creación de un proyecto llamado "PartManager": un software de apoyo a la gestión y almacenamiento de herramientas y equipo de laboratorio (diodos, condensadores, transistores, resistencias, circuitos integrados, baterías, conectores, etc). Este proyecto, consiguió en parte su objetivo, permitiendo registrar y conocer el inventario del almacén y algunos datos relevantes.

Sin embargo, aunque este proyecto ha sido suficiente para cumplir de manera mínima su propósito, debido a la falta de tiempo y otros motivos el sistema quedó incompleto, presenta varios errores y carece de algunas funciones necesarias. De este punto es de donde parte este proyecto, se requiere completar y/o mejorar el sistema de manera que ofrezca las funciones que faltan para gestionar el almacén y guardar información relevante para su manipulación y obtención.

1.2. Objetivos

Anteriormente se han comentado algunos objetivos, aunque no viene mal detallar de forma más específica los principales objetivos de este proyecto:

- **Objetivos Principales**
  - Desarrollar un sistema capaz de cumplir con los requerimientos del cliente.
  - Mantener un coste económico mínimo.
  - Trasladar los datos de inventario del sistema antiguo al nuevo.
  - Que pueda ser actualizada con facilidad.
- **Objetivos Secundarios**
  - Utilizar los conocimientos adquiridos durante el grado.
  - Adquirir los conocimientos necesarios para abordar los problemas que se plantean.
  - Que permita la realización de una aplicación móvil de consulta en un futuro.

1.3. Estructura del proyecto

Para el desarrollo del proyecto se ha seguido el modelo de desarrollo software, conocido como modelo en V.

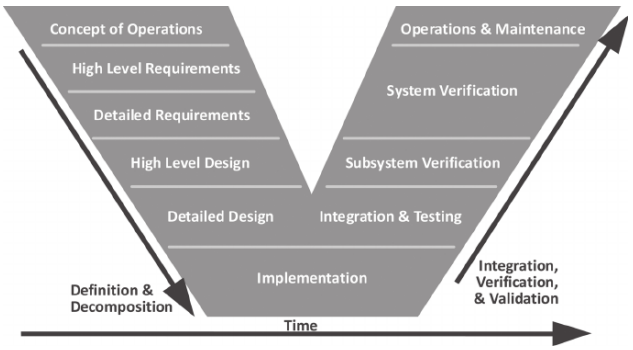


Figura 1.2 – *Modelo en V, figura extraída del artículo "Journal of Cognitive Engineering and Decision Making" [20]*

Basándose en el anterior modelo (figura 3.1 el documento se estructurará en:

1

- **Capítulo 2: Análisis**, obtención de requisitos y que herramientas se van a utilizar. ¿Qué se va a hacer?
- **Capítulo 3: Diseño**, estructura del proyecto. ¿Cómo se va a hacer?
- **Implementación**.
- **Capítulo 4: Resultados**. A nivel de funcionalidad y desde la perspectiva de la interfaz de usuario. ¿Qué se ha logrado hacer?
- **Capítulo 5: Conclusión y futuras mejoras**. Opinión una vez terminado el proyecto y posibles mejoras. ¿Se han obtenido los resultados esperados? ¿En que se puede mejorar?

Capítulo 2

Análisis

Conocer correctamente los requerimientos del sistema es de vital importancia, ya que servirá de base o punto de partida en el desarrollo de la aplicación web. Cualquier fallo en las primeras etapas del desarrollo de un sistema software suele suponer un gran esfuerzo, cuando se trate de solucionar en etapas avanzadas del proyecto.

En esta sección se va a tratar de identificar las diferentes necesidades o requerimientos del cliente, dicho de otra forma: averiguar qué es lo que el sistema debe hacer. Por otra parte, se analizarán las tecnologías que se emplearán para desarrollar el sistema.

2.1. Requisitos

En este punto es muy importante la comunicación con el cliente para conocer sus necesidades, qué es lo que se traducirá en el sistema como requisitos. También comprender el entorno o ámbito del problema es esencial. Destacar que en este caso se dispone de un sistema incompleto que puede ser usado como referencia para plantear los requisitos y conocer mejor lo que quiere el cliente. En el siguiente estudio de requisitos no se va a tener en cuenta los requisitos ya implementados, esto se tratará más adelante.

Tras varias reuniones con el cliente se ha obtenido la información necesaria para poder conseguir los requisitos, a continuación se van a mostrar estos desde una forma general a una más específica:

Descripción general del problema, se necesita un sistema que sea capaz de almacenar información sobre los componentes electrónicos de un laboratorio. Esta información reflejará cantidad disponible, detalles técnicos del componente y datos del proveedor (nombre, página web, número de referencia..) que faciliten su compra.

Además el sistema debe cumplir con los siguientes puntos:

- El sistema debe restringir el acceso sólo al personal autorizado, para ello empleará un sistema de autenticación con usuario y contraseña.
- Se deben almacenar archivos en formato texto con información para cada componente.
- La forma en la que se almacenan y muestran los datos de cada componente debe estar estructurada por categorías o similar para facilitar su acceso. Así mismo, se deben poder crear o modificar estas categorías.
- Se debe facilitar la creación de nuevos componentes, extrayendo los datos directamente de las páginas web de los proveedores, así mismo estas categorías pueden ser modificadas fácilmente.
- Se debe incorporar funcionalidad para el reconocimiento de códigos de barras o código QR, con el fin de poder etiquetar y identificar rápidamente los componentes.
- El sistema debe ser capaz de reflejar las transacciones de componentes, es decir, modificar la cantidad de cada componente si se utilizan o añaden. Así como llevar un control de quien extrae un componente.

- El sistema debe alertar al administrador o administradores cuando un componente se encuentre bajo mínimo. Así mismo, se deberá facilitar la información de los componentes bajo mínimo mediante una hoja de cálculo o similar.
- Se debe poder diferenciar dos componentes iguales pero de diferente proveedor. El componente electrónico es llamado "part" mientras que el "stock" esta formado por un "part" y el proveedor.
- El sistema debe almacenar información sobre los diferentes lugares de almacenaje del laboratorio, para poder ubicar los componentes.
- El sistema debe almacenar información sobres los diferentes proveedores de componentes.
- El uso de sistema debe ser fácil y intuitivo.
- Existirán dos roles dentro del sistema: administrador y usuario normal. Un usuario normal solo podrá hacer uso acciones de búsqueda de componentes y retirar o añadir componentes.

A partir de la información anterior se extraerán dos tipos de requisitos con información más específica: requisitos funcionales, que reflejarán funciones del sistema y requisitos no funcionales que representan restricciones o la forma en la que los requisitos funcionales deben actuar.

2.1.1. Requisitos Funcionales

- **Gestión de Usuarios**
  - **RF1. Inicio de sesión en el sistema**: un usuario debe ser capaz de poder acceder al sistema utilizando un nombre de usuario y una contraseña.
  - **RF2. Dar de alta usuario**: el sistema debe permitir dar de alta a nuevos usuarios.
  - **RF3. Modificar usuario**: el sistema debe permitir modificar los datos de los usuarios ya existentes.
  - **RF4. Eliminar usuario**: el sistema debe permitir dar de baja a un usuario.
  - **RF5. Cerrar Sesión**: el sistema debe permitir cerrar la sesión activa cuando se requiera.
- **Gestión de inventario**

- **RF6. Crear componente**: el sistema debe ser capaz de permitir añadir nuevos componentes electrónicos.
- **RF7. Editar componente**: se debe poder modificar la información sobre cualquier componente.
- **RF8. Búsqueda de componentes**: se debe permitir buscar componentes mediante nombre y/o otros campos.
- **RF9. Crear Stock**: el sistema debe ser capaz de permitir añadir nuevo stock. El stock estará vinculado a un componente ya existente y a un proveedor entre otros datos que se especificarán más adelante.
- **RF10. Modificar Stock**: el sistema debe permitir modificar información relativa a un elemento de stock.
- **RF11. Eliminar Stock**: el sistema debe permitir eliminar un elemento de stock.
- **RF12. Búsqueda de Stock**: se debe permitir buscar elementos de stock mediante nombre y otros campos.

- **Gestión de categorías**
  - **RF13. Crear nueva categoría**: se debe permitir crear nuevas categorías en las que organizar el stock, ya sea una categoría principal o subcategoría.
  - **RF14. Renombrar categorías**: se debe permitir modificar el nombre de las categorías existentes.
- **Gestión de lugares de almacenamiento**
  - **RF15. Crear lugar de almacenamiento**: se debe permitir crear nuevos puntos de almacenaje dentro del laboratorio, lugares donde se almacenarán los elementos de stock.
- **Gestión de archivos**
  - **RF16. Adjuntar archivo a componente**: se debe permitir adjuntar archivos de información a los componentes.
- **Gestión de proveedores**
  - **RF17. Crear proveedor**: se debe permitir la creación de proveedores en el sistema.

2.1.2. Requisitos No Funcionales

- **RNF1**: se deben restringir ciertas funciones a los usuarios no administradores, estos sólo podrán consultar información y retirar o añadir componentes. Los administradores podrán acceder a todas las funciones disponibles.

- **RNF2**: todas las transacciones, es decir, la retirada o incorporación en la cantidad de un stock debe ser reflejada en el sistema, guardando información de cantidad, fecha y usuario.
- **RNF3**: se debe poder acceder a la información del stock a partir de un código QR o código de barras.
- **RNF4**: se debe facilitar la creación de nuevos componentes y/o stock, extrayendo los datos de las páginas de proveedores directamente. Utilizando una extensión o similar.
- **RFN5**: se debe notificar de alguna forma a los administradores cuando el stock se encuentra bajo mínimo.
- **RFN6**: se debe facilitar la información relativa a stock bajo mínimo mediante la generación de hojas de cálculo o similar.
- **RFN7**: se tiene que permitir filtrar la búsqueda por categorías o algún parámetro extra a parte de sólo el nombre.

2.1.3. Requisitos sin implementar

Como se ha explicado, la lista de requisitos anterior no tiene en cuenta el sistema ya existente. Este proyecto se va a centrar en desarrollar las funcionalidades que faltan, que son:

- Añadir funciones a la **gestión de usuarios** de modificar usuarios existentes, eliminar usuarios y permitir cerrar sesión. (**Requisitos RF3, RF4 y RF5**).
- Respecto a **gestión de inventario** permitir modificar los datos de componentes ya existentes, así como también poder modificar el stock ya existente o eliminarlo (**Requisitos RF7,RF10,RF11**).
- En cuanto a la **gestión de categorías** se permitirán crear nuevas categorías y renombrar las existentes (**Requisitos RF13, RF14**).
- **Integración de código QR** (Requisito RNF3).
- **Actualización de la extensión para Google Chrome** (Requisito RNF4).
- **Añadir control de la cantidad de stock**, mediante notificaciones, generación de hojas de cálculo con datos relevantes o gráficos de uso (RNF5 y RNF6).

- Permitir el **filtrado de búsqueda de stock** por parámetros relevantes como puede ser el proveedor o la categoría (RNF7).
- A parte de implementar por completo los requisitos anteriores, se deberá mejorar el funcionamiento de los requisitos ya existentes, mejorar su interfaz de usuario y arreglar errores si los hubiese.

2.2. Análisis de tecnología

Antes de entrar en el diseño del sistema, una vez se conoce qué debe hacer el sistema lo siguiente a plantear es **qué herramientas se van a utilizar**.

En este caso en particular se dispone de varias opciones, se dispone de un sistema incompleto de gestión de almacén (el antiguo "PartManager") que puede servir de base ya que implementa algunas funciones, incluso se podría crear un sistema desde cero o ¿por qué no utilizar uno de los ya existentes gestores de almacén?

"¿Para qué reinventar la rueda?". Al fin y al cabo, existe multitud de software de gestión de inventario, y muchos de ellos de código abierto, por lo que no habría ningún problema de tipo económico. Por ello, no hay motivo para no incluir este tipo de software como una posibilidad.

2.2.1. Análisis de software de gestión de almacén de terceros

Descartando todos aquellos gestores de almacén que no sean **totalmente gratuitos**, entendiendo por gratuito, que puedan ser utilizados sin limitaciones en funcionalidad o duración, las opciones se reducen bastante.

En base a facilidad de uso, apoyo en la web y aspecto, se han seleccionado dos gestores a poner a prueba : PartKeepr [16] y StockPile [19]. Estos gestores han sido analizados y puestos a prueba con el fin de comprobar si son adecuados para el propósito del proyecto, los resultados de este estudio se muestran a continuación.

StockPile

StockPile es un sistema de gestión de inventario online totalmente gratuito, destinado a la pequeña empresa. El código de este gestor no se encuentra disponible, y es online por lo que sólo requiere registrarse para utilizarlo y no necesita ninguna instalación.

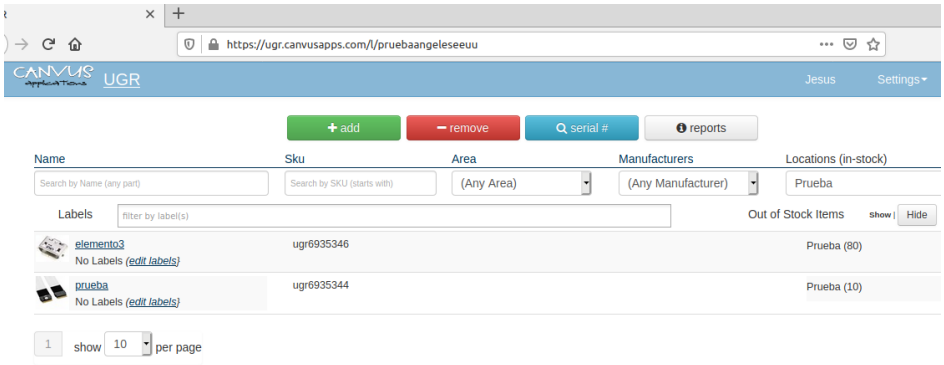


Figura 2.1 – *Captura lista de inventario, StockPile*

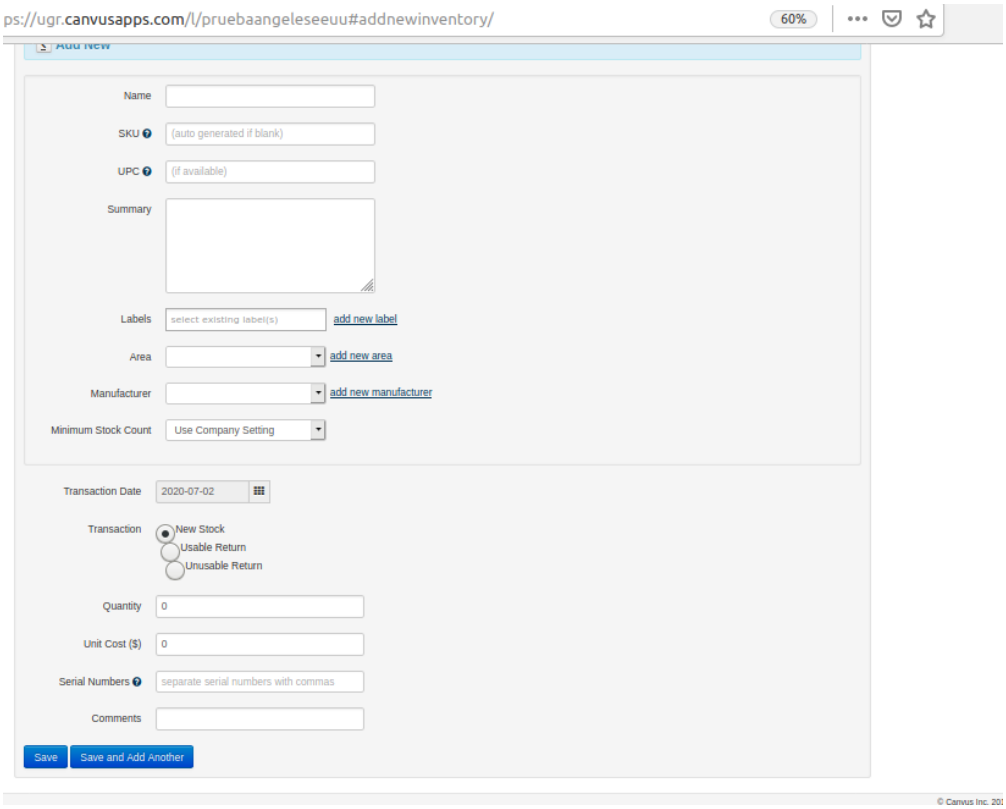


Figura 2.2 – *Crear inventario, StockPile*

Es rápida y fácil de utilizar. No obstante, para añadir un elemento en el inventario se necesitan demasiados campos que no son necesarios en el laboratorio, no se puede personalizar y no se adapta a las necesidades del cliente, entre otras cosas no permite adjuntar archivos a un componente o diferenciar lugares de almacenamiento dentro del propio almacén.

2 PartKeepr

PartKeepr es un proyecto código abierto escrito en php y su código se encuentra totalmente accesible en github. La última versión 1.4.0 para ser utilizada requiere de:

- Sistema Operativo UNIX (Linux, BSD, etc.).
- Servidor Web (Apache2 o nginx).
- Versión de PHP 5.6 o superior.
- Base de datos MySQL o PostgreSQL.

Es un sistema de gestión de inventario especializado en componentes electrónicos, este es un gran punto a favor ya que es justo lo que se necesita.

PartKeepr si necesita ser instalado para su utilización,pero en su página oficial hay una "demo" online disponible, que es más que suficiente para poner a prueba su funcionamiento:

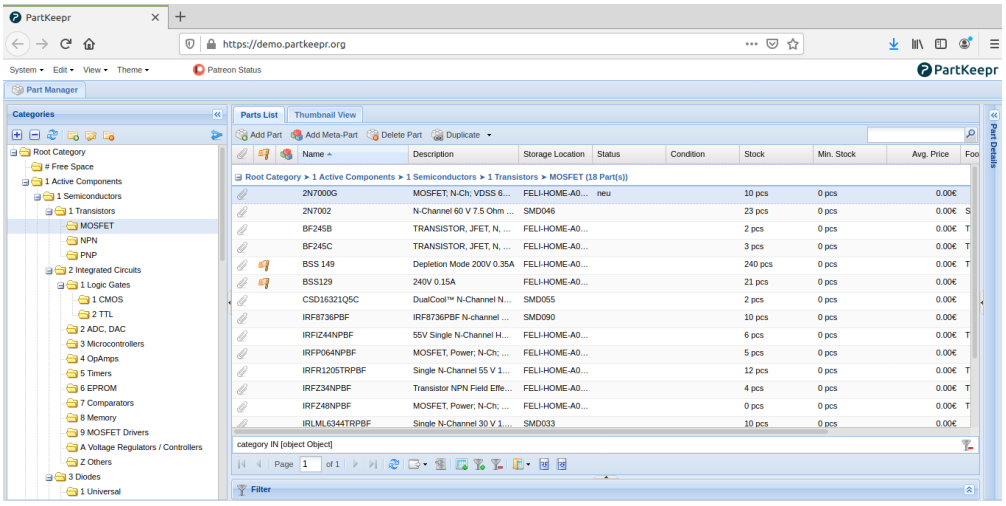


Figura 2.3 – *Captura lista de inventario, PartKeepr*

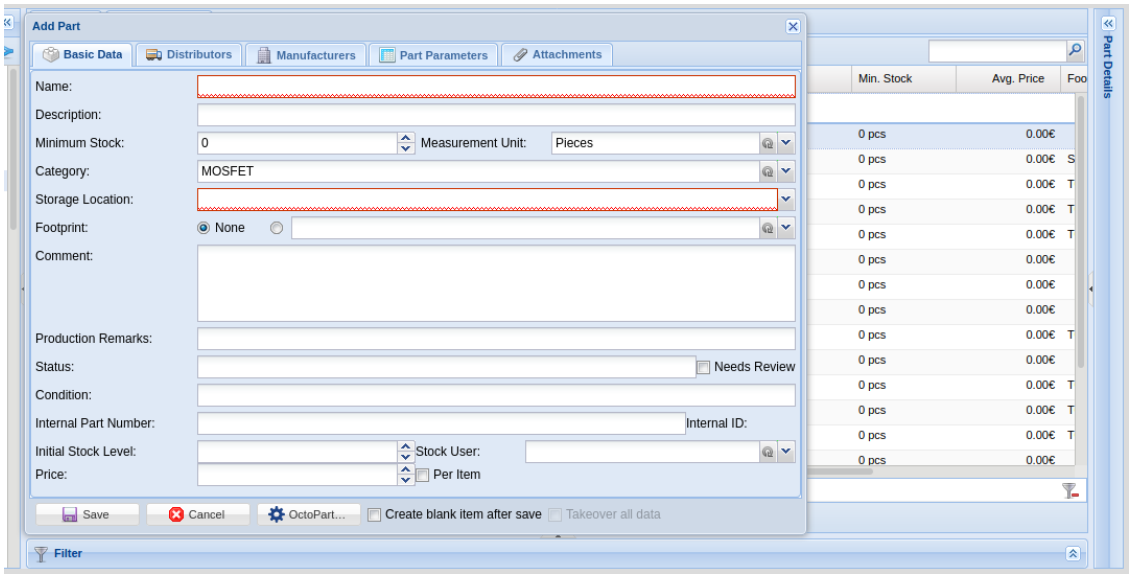


Figura 2.4 – *Crear inventario, PartKeepr*

Este software es algo más difícil de utilizar que el gestor anterior, aunque es algo normal ya que es mucho más completo, y además cumple con la mayoría de requisitos del cliente.

Este gestor está muy cerca de lo que se necesita, pero requeriría de bastantes modificaciones (personalizar la interfaz, añadir código QR para acceder a los datos..). Su código se encuentra disponible a todo el mundo, por lo que una posibilidad sería modificarlo y adaptarlo. Sin embargo, esta opción puede ser muy laboriosa y no se podría obtener el mismo resultado que realizar un proyecto desde cero o a partir del proyecto antiguo que se dispone, por lo que queda descartado.

A modo de resumen del estudio anterior, se ha extraído la siguiente tabla comparativa 2.1.

	StockPile	PartKeepr
Descripción	Gestor de inventario online para pequeña empresa.	Gestor de inventario destinado a componentes electrónicos.
Dificultad de uso	Fácil.	Dificultad media.
Posibilidad de modificación	No se dispone del código fuente	Disponibilidad del código fuente escrito en php, aunque su estructura y volumen lo hace muy complicado.
Nivel de requerimientos	Bajo.	Nulo, no requiere instalación.

Table 2.1 – *Tabla comparativa de gestores de inventario, StockPile vs PartKeepr*

2.2.2. Análisis de herramientas a utilizar

En el anterior punto se ha estudiado software de terceros muy interesante para la gestión de inventario. Sin embargo, estas soluciones han sido descartadas por no cumplir con los requerimientos que necesita el problema que se plantea.

Finalmente, ante la ausencia de más alternativas viables, se ha decidido completar el sistema de gestión ya existente en GranaSAT. Puede ser tedioso tratar con los fallos a nivel de código que pueda existir o el hecho de tener que comprender el código de terceros suele resultar complicado y aún más si se suma la escasa o nula documentación. Aún así, la mayoría de código puede ser de utilidad y cumple con algunas funciones de una forma aceptable.

A continuación, se ha realizado un análisis de las herramientas y framework que utiliza este sistema: **ReactJS** y **Bootstrap** en el front-end, y **Nodejs** y MySQL en el back-end.

Tecnología en el lado de cliente: ReactJS

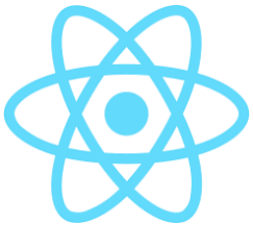


Figura 2.5 – *Logo de ReactJS [17]*

ReactJS es una biblioteca de JavaScript de código abierto utilizada para desarrollar interfaces de usuario reutilizables. ReactJS está basado en componentes y es utilizado como la "Vista" dentro del modelo MVC ("Modelo-Vista-Controlador"). Actualmente es una de las librerías JS (JavaScript) más utilizadas en front-end, y cuenta con el soporte de Facebook, Instagram y una gran comunidad de desarrolladores.

Principales ventajas y desventajas en la tabla 2.2.

Ventajas	Desventajas
Lenguaje actual y con buen soporte.	La curva de aprendizaje puede ser pronunciada.
Interfaces de usuario dinámicas e interactivas.	Implementa sólo la parte de la Vista.
Cuenta con gran cantidad de documentación y ayuda en la web.	
Es bastante eficiente	

Table 2.2 – *Ventajas y Desventajas, ReactJS*

A pesar de las desventajas, es es evidente de que Reactjs es una de las opciones más potentes para desarrollar front-end actualmente y en este caso el aprendizaje no es un impedimento.

Tecnología en el lado de cliente: Bootstrap 4 y CSS

Figura 2.6 – *Logo de Bootstrap [2]*

Bootstrap es un framework que combina HTML y Cascade Style Sheet (CSS), utilizado en el front-end para la creación de sitios web de una manera rápida y vistosa. Ofrece estilos y componentes como "navigations", "forms" o "buttons" entre otros, que pueden ser utilizados en un sitio web de una forma cómoda y sencilla.

Principales ventajas y desventajas, presentadas en la tabla siguiente 2.3

Ventajas	Desventajas
Gran cantidad de documentación	Es necesario aprender a utilizarlo, aunque la curva de aprendizaje es liviana ,es algo a tener en cuenta.
Ofrece un diseño adaptable, es decir, permite ajustar el sitio web a diferentes dispositivos.	Es probable que se necesite incorporar componentes o estilos que no existen en bootstrap por lo que en algunas ocasiones seguirá haciendo falta utilizar CSS por separado.
Es compatible con la mayoría de los navegadores actuales	
El hecho de disponer de diferentes componentes ya diseñados facilita el trabajo.	

Table 2.3 – *Ventajas y Desventajas, Bootstrap 4*

En reactjs, bootstrap 4 se encuentra adaptado en reactstrap. Como se ha visto utilizar este framework ahorra mucho código, aunque en ocasiones no se podrá evitar utilizar reglas CSS propias.

Tecnología en el lado del servidor: NodeJS

Figura 2.7 – *Logo de NodeJS [15]*

Nodejs es un entorno de ejecución JavaScript, de código abierto, utilizado en el lado del servidor. Nodejs opera ejecutando un único proceso, utilizando un modelo de entrada y salidas asíncronas lo que permite servir de manera concurrente a miles de conexiones. El hecho de que utilice un único subproceso ahorra tener que crear nuevos subprocesos o cambiar entre estos, lo que se traduce en que tiene muy poca sobrecarga.

Es muy utilizado en la actualidad, considerado uno de los framework más populares.

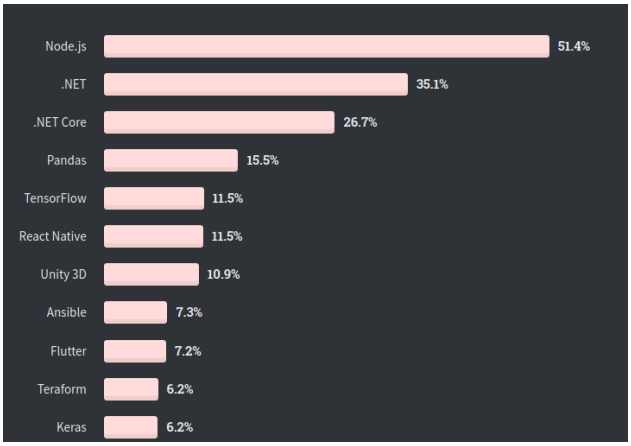


Figura 2.8 – *Ranking de los framework más usados en 2020, según StackOverflow [21]*

En la tabla 2.4 se reflejan sus ventajas e inconvenientes.

Ventajas	Desventajas
Fácil de aprender si conoces JavaScript	No recomendable para principiantes, JavaScript es difícil de aprender en comparación con otros lenguajes
Una gran comunidad y una gran cantidad de documentación	Requiere escribir más código que con otros framework del lado del servidor.
Gran cantidad de librerías disponibles y fácil instalación con npm	
Buen rendimiento	

Table 2.4 – *Ventajas y Desventajas, NodeJS*

NodeJS es una de las mejores herramientas que se podrían utilizar para el este proyecto, y teniendo en cuenta que se va trabajar tanto en front-end como en back-end, tener la posibilidad de utilizar el mismo lenguaje (JavaScript) en ambos, es algo a valorar.

Tecnología en el lado del servidor: Base de datos MySQL

Figura 2.9 – *Logo de MySQL [9]*

Por último, análizar una de las partes más importantes en la mayoría de servicios web: la base de datos. En este caso se utilizará MySQL, la base de datos "open source" relacional más usada y popular actualmente.

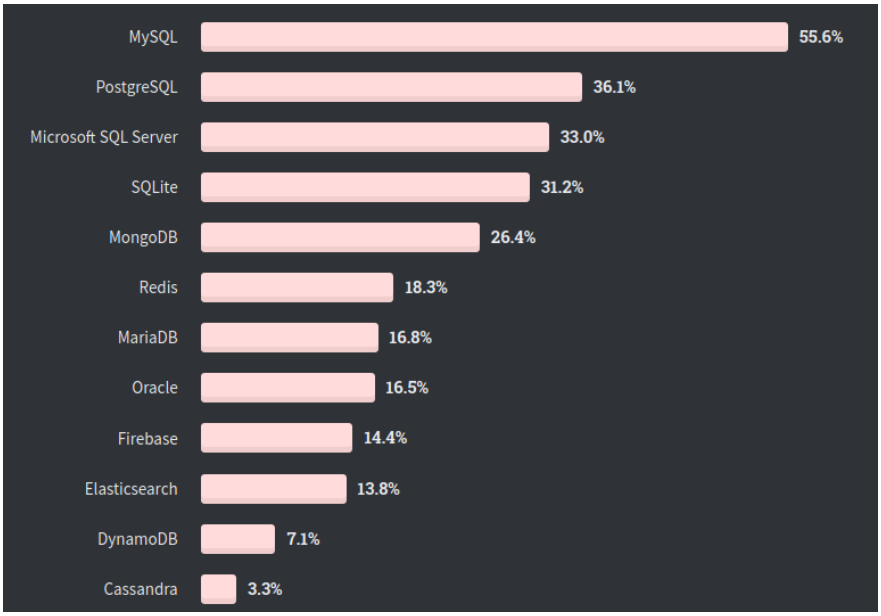


Figura 2.10 – *Bases de datos más utilizadas en 2020 según StackOverflow [21]*

En la tabla 2.5 se presentan sus ventajas y desventajas

Ventajas	Desventajas
Dispone de una gran comunidad y apoyo en la web.	Algunas funcionalidades no se encuentran documentadas apropiadamente.
Trabaja rápido si la cantidad de datos no es muy elevada (en el caso del proyecto no lo será).	
Requiere de pocos recursos para funcionar	

Table 2.5 – *Ventajas y Desventajas, MySQL*

En el proyecto se ha utilizado como apoyo MySQL WorkBench [10], una herramienta muy útil, que va a permitir diseñar y manipular la base de datos de una manera rápida y visual.

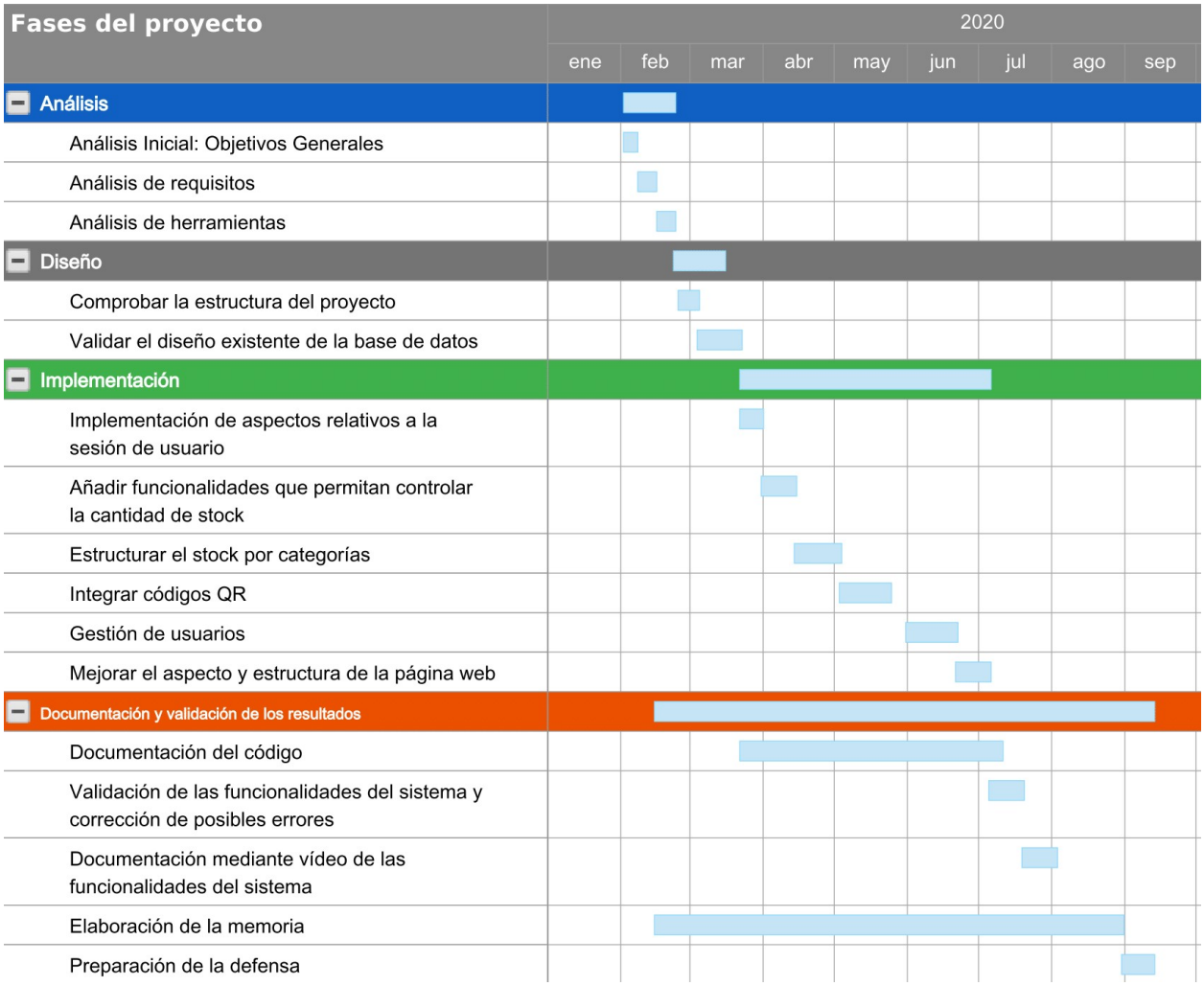


Figura 2.11 – *Diagrama de Gantt del proyecto.*

Capítulo 3

Diseño

El diseño es una de las partes principales dentro del desarrollo software, en el se materializarán los requisitos obtenidos en la fase de análisis. Un buen diseño se traduce en una buena organización que va a permitir tratar de forma independiente las diferentes partes del sistema, esto va a facilitar la depuración de código durante el desarrollo y el mantenimiento del software, y ,entre otras cosas, también va a hacer que el proyecto sea mucho más fácil de entender o intuitivo .

Después conocer las necesidades del sistema en el capítulo anterior, en este se va definir el diseño del sistema. Se va a especificar la arquitectura, es decir, como las diferentes partes que componen el sistema se comunican entre sí. De igual forma, se realizará el diseño de la base de datos, en orden descendente de nivel de abstracción.

3.1. Estructura del proyecto

La arquitectura del sistema se basa en el modelo MVC (Modelo-Vista-Controlador). Este modelo es uno de los más usados actualmente y permite separar la lógica de la aplicación de la vista. En este modelo se diferencian 3 partes:

- **Modelo**, se encarga de la manipulación de datos. Trabaja sobre la base de datos.
- **Vista**, se encarga de la representación visual, es decir, la interfaz de usuario.
- **Controlador**, se encarga de procesar las peticiones del usuario y comunicar el Modelo y la Vista.

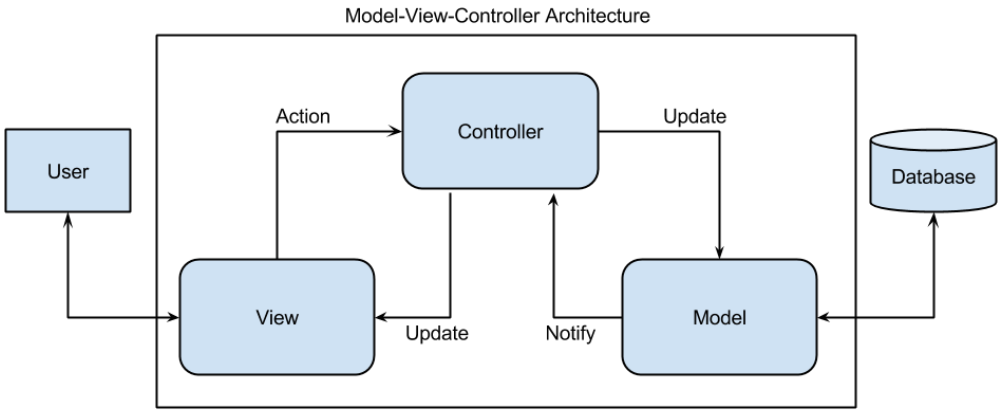


Figura 3.1 – *Imagen que ilustra la arquitectura Modelo-Vista-Controlador [7]*

Gracias a esta arquitectura (figura 3.1) se pueden tratar las diferentes partes de manera independiente, por lo que si se modifica (por ejemplo) el Modelo no afectará al resto del sistema. Esto facilita en gran medida el mantenimiento del código.

En este proyecto la arquitectura MVC se garantiza distinguiendo dos partes: front-end, ejecutado en el lado del cliente, se encarga de la interfaz de usuario, y back-end, ejecutado en el lado del servidor, se encarga de gestionar los datos y la lógica. Estas dos partes se comunican mediante mensajes JSON enviados por medio de mensajes HTTP, todo esto se ilustra en la figura 3.2.

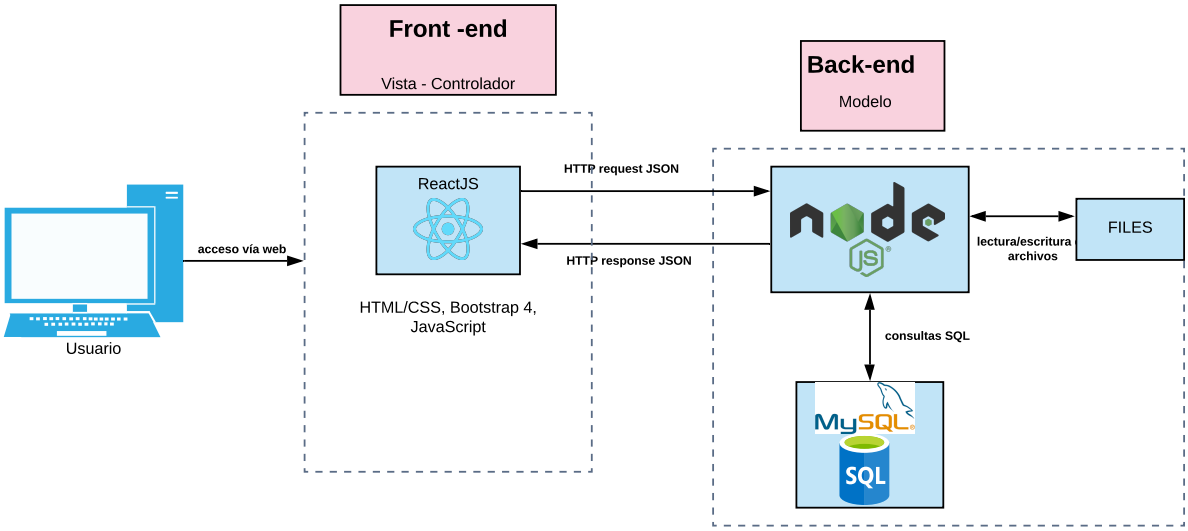


Figura 3.2 – *Arquitectura del sistema*

3.1.1. Diagramas de secuencia

Con el fin de ilustrar de una forma más detallada cómo se comunican e intervienen las diferentes partes del sistema, se han representado algunos diagramas de secuencia, con las operaciones que suelen ser más utilizadas.

- **Login, acceder al sistema**
  El siguiente diagrama muestra los procesos o mensajes generados cuando un usuario intenta iniciar sesión. Al realizar una autenticación con éxito se genera un token de sesión, que será añadido junto a información del usuario en los mensajes HTTP. Una vez el usuario esté autenticado, se le mostrarán parte de las funcionalidades del sistema (en caso de ser usuario normal) o todas las funciones (si es administrador).

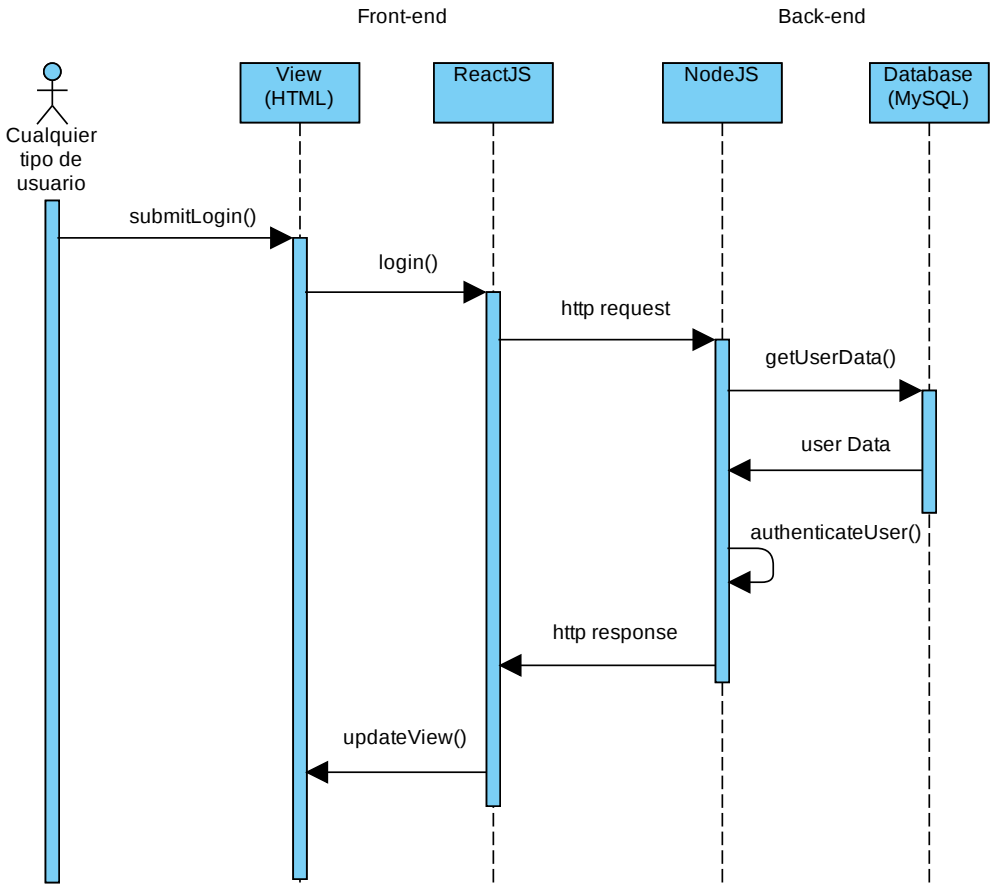


Figura 3.3 – *Diagrama de secuencia, login*

- **Dar de alta a usuario** En este caso el actor debe ser administrador, en caso de insertarse el nuevo usuario el servidor devolverá un mensaje de éxito, de lo contrario un mensaje de error.

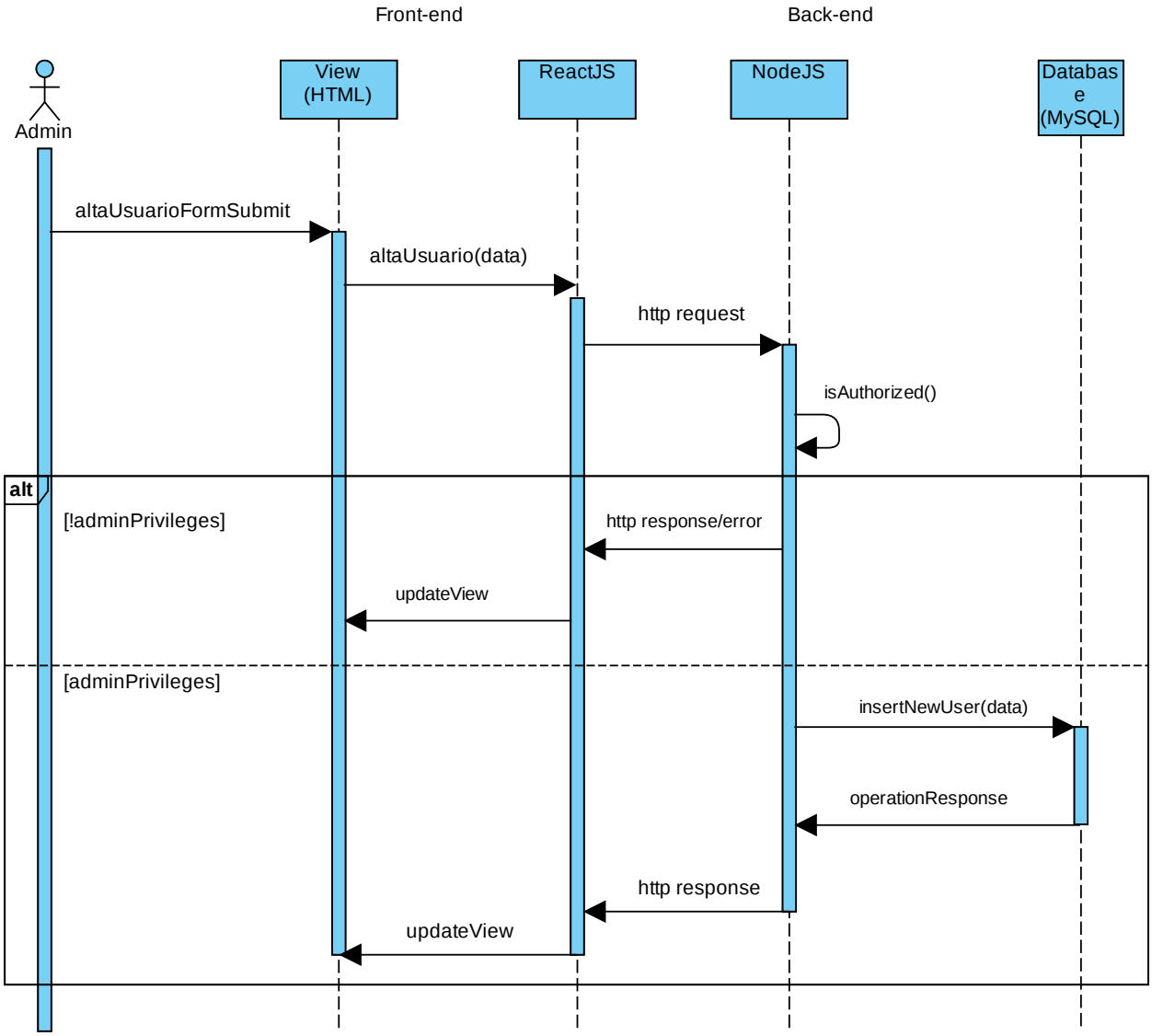


Figura 3.4 – *Diagrama de secuencia, dar de alta usuario*

- **Búsqueda de stock** Esta es una de las funciones que más se va a utilizar en la aplicación. Se inicia cuando el usuario introduce algún texto en la barra de búsqueda, el usuario obviamente debe estar autentificado en el sistema con anterioridad.

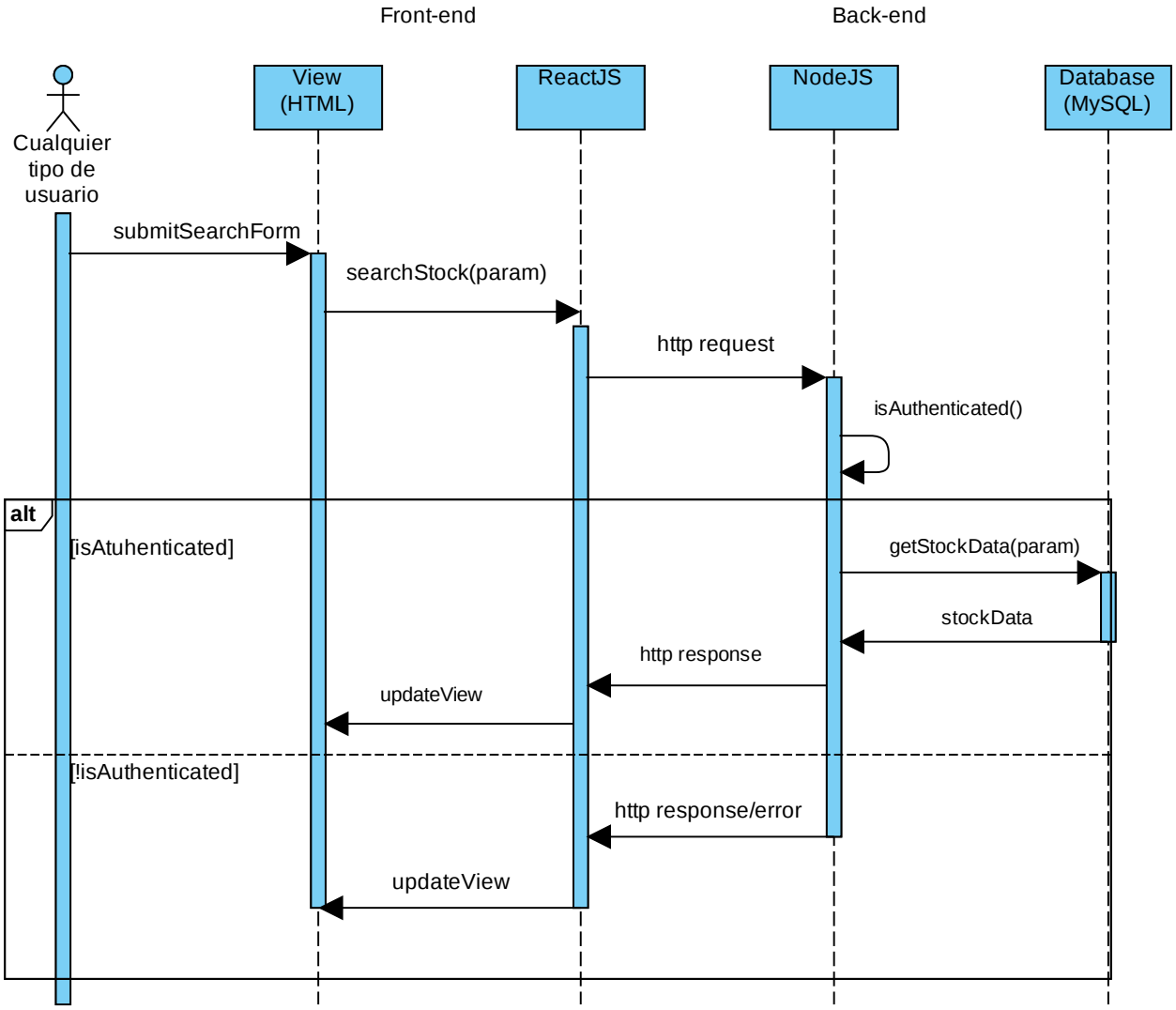


Figura 3.5 – *Diagrama de secuencia, búsqueda de stock*

3.2. Diseño de la base de datos

Actualmente, la base de datos es una de las partes más importantes dentro de la mayoría de servicios web. Al fin y al cabo, será la que almacene y gestione la información que utilizará el sistema. Por ello, el diseño de la base de datos es esencial.

En esta sección no solo se va a mostrar el diseño actual de la base de datos, si no también el proceso a seguir para llegar a dicho diseño. Se recuerda que para este proyecto ya se dispone de una base de datos repleta de información y con el fin de comprobar la validez del diseño de esta, se va a plantear el diseño desde la obtención de requisitos. Este diseño, sin tener en cuenta el diseño físico, constará de 2 partes: **diseño conceptual** y **diseño lógico**.

3.2.1. Diseño conceptual de la base de datos

El diseño conceptual es la fase con mayor nivel de abstracción. Se utilizarán los requisitos obtenidos previamente para definir un "esquema conceptual" que plasmará las entidades y las relaciones entre estas.

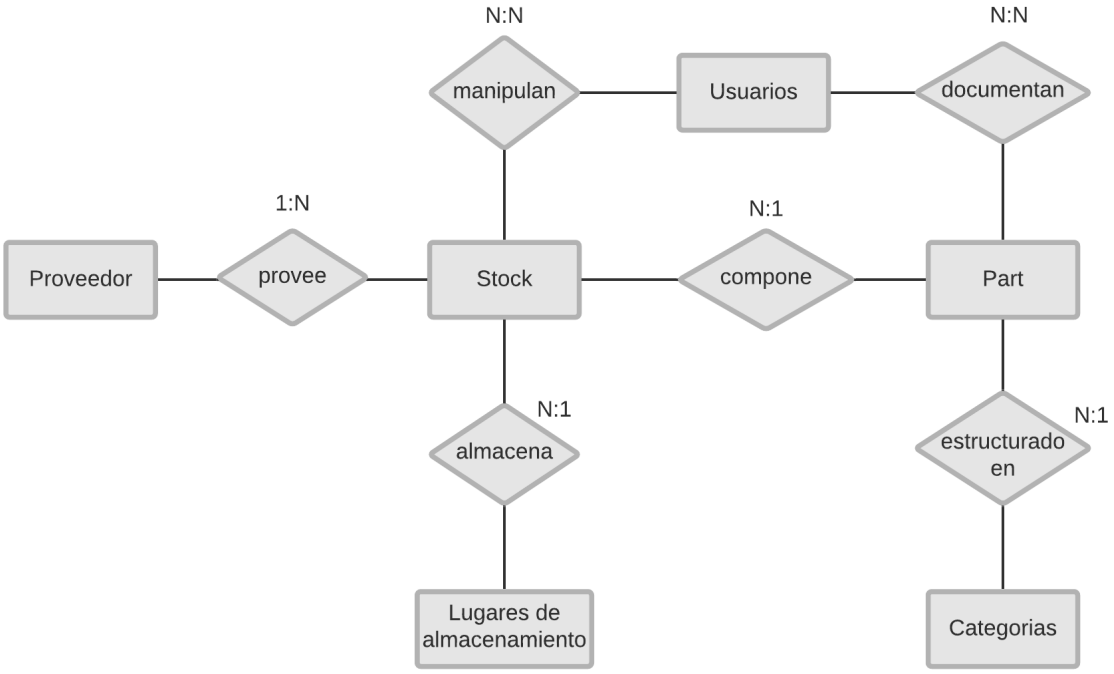


Figura 3.6 – *Esquema conceptual obtenido a partir de los requisitos*

3.2.2. Diseño lógico de la base de datos

En esta fase intermedia se tratará de acercar el esquema conceptual a un modelo más cercano a la representación de los datos. El resultado de este proceso será el "esquema relacional".

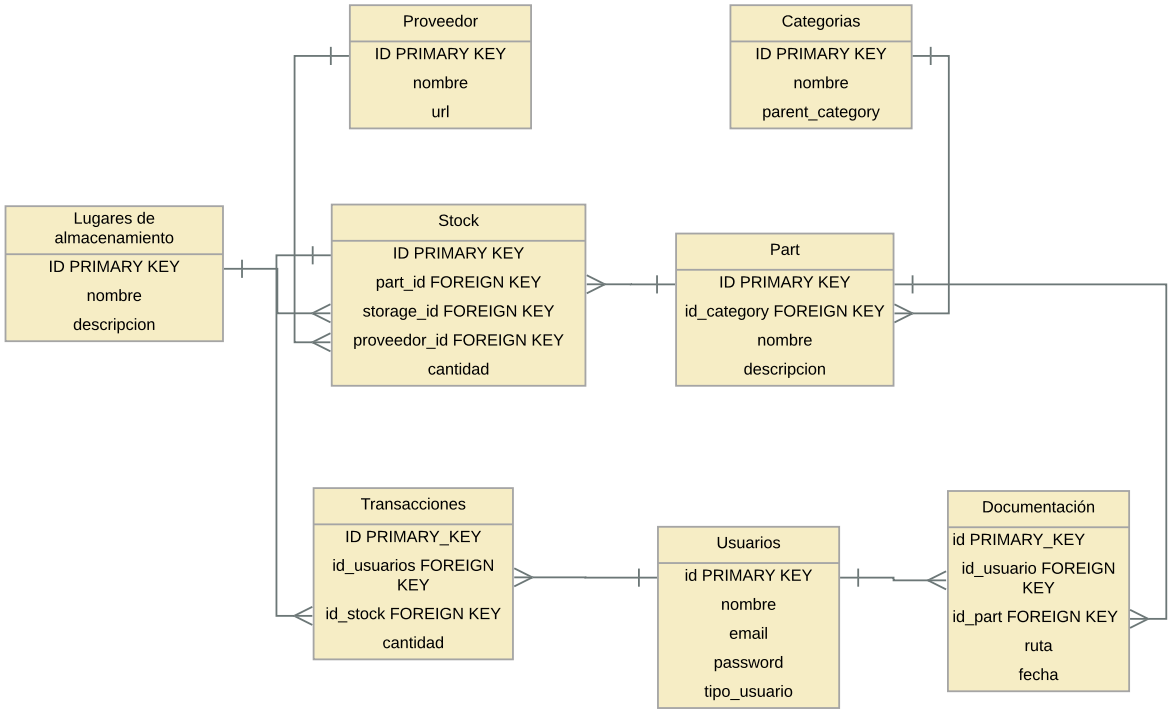


Figura 3.7 – *Esquema relacional, obtenido a partir del esquema conceptual.*

Nótese que las tablas "Documentación" y "Transacciones" que aparecen en la figura 3.7 provienen de las relaciones en esquema conceptual "documentan" y "manipulan" respectivamente. Destacar que este esquema se centra en ilustrar las posibles tablas y relaciones entre estas, por lo que no se ha visto relevante añadir todos los atributos propios de cada tabla.

3.2.3. Resultado de la fase de diseño de la base de datos

La base de datos actual, tiene demasiadas tablas, muchas de ellas ni si quiera son usadas y no tienen ninguna función clara. Al parecer en la implementación del gestor antiguo se crearon algunas tablas para cumplir unas funciones que finalmente no se añadieron, otras tablas como "usergroup" si son utilizadas pero son perfectamente prescindibles (ya que solo van a existir 2 roles añadiendo una columna en la tabla "users" bastaría). El siguiente esquema relacional **de la base de datos antigua** hace evidente esta conclusión (figura 3.8).

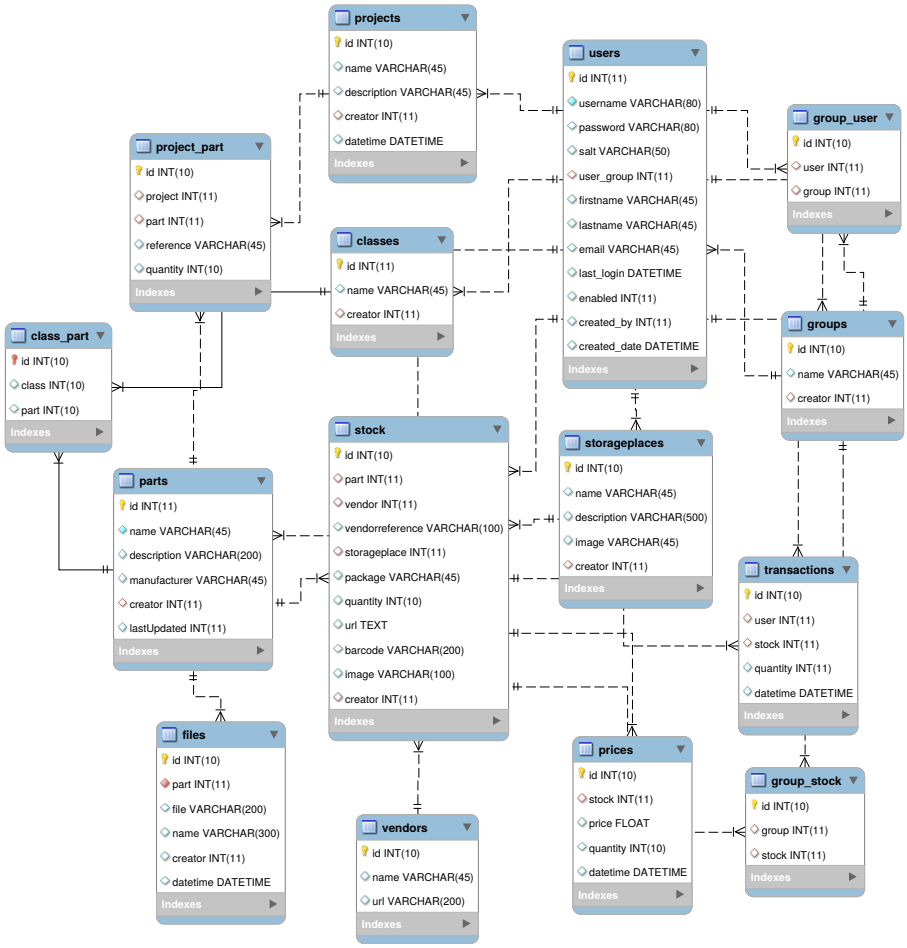


Figura 3.8 – *Esquema relacional de la base de datos ya existente en el servidor*

Tras el diseño lógico que se ha realizado en la subsección anterior, es sencillo conocer los cambios que necesita el actual diseño de la base de datos. Aplicando las modificaciones pertinentes cómo eliminación de tablas innecesarias, incluir las claves foráneas necesarias y crear nuevas tablas, se obtiene el **esquema relacional definitivo** representado en la figura 3.9.

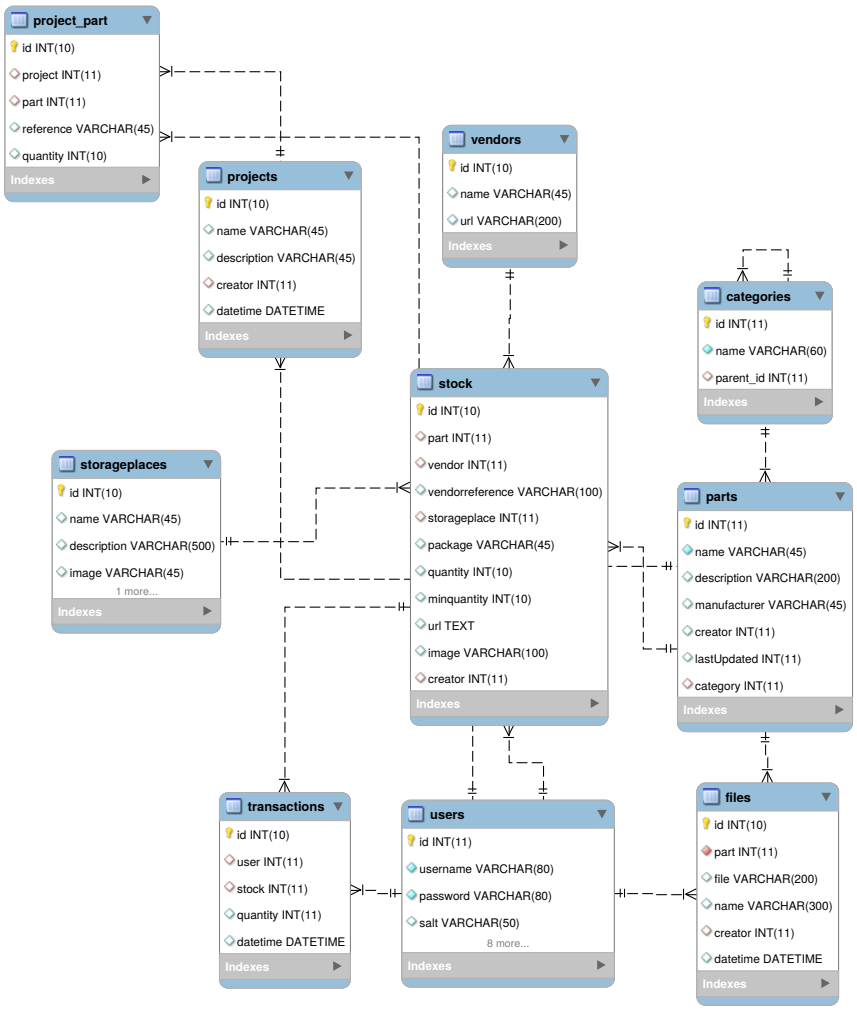


Figura 3.9 – *Esquema relacional de la base de datos definitivo*

Aclarar que las tablas "projects" y "project-part" se han mantenido en la base de datos a petición del cliente, ya que en un futuro permitirán crear proyectos concretos y contendrán una lista de los componentes usados para esos mismos proyectos. Este propósito no se encuentra dentro de los objetivos de este trabajo por lo que simplemente se va a ignorar.

Capítulo 4

Implementación

Se podría decir que la fase de implementación es la principal dentro todo tipo de desarrollo software. En esta etapa es donde se va a materializar todo lo planificado, analizado y diseñado en etapas anteriores mediante el desarrollo de código.

El objetivo de este capítulo no es mostrar todo el código implementado, ya que el objetivo no es enseñar al lector programar en JavaScript, y además el código es demasiado extenso cómo para poder enseñarlo correctamente . No obstante se va a mostrar una pequeña parte del código desarrollado durante el proyecto, con el objetivo de mostrar cómo se han resuelto en un mayor nivel de detalle algunos de los problemas o requerimientos más relevantes e interesantes.

4.1. Código desarrollado

Este capítulo se va a dividir en:

- Intercambio de mensajes entre front-end y back-end.
- Creación del servidor HTTPS.
- Estructurar información en categorías.
- Integración de códigos y escáner QR.
- Extracción de datos con extensión de Google Chrome.

4.1.1. Intercambio de mensajes entre front-end y back-end

Como ya se ha visto en el capítulo de diseño, esta aplicación se basa en un modelo MVC por lo que se trata de forma independiente el lado del cliente y el servidor. Para el funcionamiento de la aplicación es imprescindible la comunicación entre estas partes, en este caso por medio de mensajes HTTP, que van a contener JSON con la información necesaria en cada caso. Este intercambio de mensajes va a ser muy utilizado en la aplicación para obtener información sobre usuarios, stock, parts, proveedores, categorías, transacciones, documentación o lugares de almacenamiento.

Todos estos mensajes tienen una estructura muy parecida. A modo de ejemplo, se va a mostrar el código que interviene (tanto en el lado del cliente como en el lado del servidor) en el intercambio de mensajes HTTP, para obtener información sobre stock en función de unos parámetros de búsqueda.

- **Lado del cliente**, se utiliza la librería "axios" para realizar la petición.

```
export const searchStock = (search,vendor,category) => {
    return axios.get('/api/stock', {
        params: {
            search: search,
            vendor: vendor,
            category: category
        }
    })
}
```

Listado 4.1 – *Http request example: searchStock.*

- **Lado del servidor**, tiene habilitada una ruta ("/api/stock") donde va a recibir las solicitudes, en este caso tipo "GET". En función de los parámetros del request actúa de una forma u otra. Nótese que se comprueba que el usuario está autenticado ("isAuthenticated").

```
1   router.get('/api/stock', isAuthenticated, function(req, res) {
2   var user = req.user
3   var data = req.query
4
5   var f = null
6   if (data.hasOwnProperty("search")) {   // Si la petición tiene parámetro search es una búsqueda normal
7       f = dbManager.searchStock
8   }else if(data.hasOwnProperty("id")){   // Búsqueda de un stock concreto por id
9       f = dbManager.getStockbyid
10  } else{
11      f = dbManager.getStock             // En caso contrario obtener todo el stock a partir de 'part' y
            'vendor'
12  }
13  f(user, data).then((results) => {
14      res.status(200).json({
15          results: results,
16      })
17  }).catch((error) => {
18      if (error) logger.error(error);
19      res.status(400).json({
20          error: error
21      })
22  });
23 });
```

Listado 4.2 – *Http response example: searchStock.*

4.1.2. Crear servidor HTTPS

El sistema en un principio utilizaba un servidor HTTP. Obviamente, esto no afecta a nivel de funcionalidad, pero si puede provocar problemas en el momento de acceder al servicio web con algunos navegadores. Otra razón para utilizar HTTPS es que aporta seguridad y hace que la aplicación web tenga un aspecto de "confianza" de cara al usuario.

Gracias a que GranaSAT ya dispone de certificado https, sólo se ha tenido que modificar el código dentro del archivo principal, en el lado del servidor ("/granasat-pm-server/app.js"), y utilizando el módulo "https" [11] , el código es el que se muestra en la figura **??** .

```
1   \label{listing:https}
2     // Importar modulos necesarios
3     var path = require("path")
4     var express = require('express')
5     var app = express()
6     var helmet = require('helmet')
7     var fs = require('fs')
8     var https = require('https')
9     app.use(helmet())
10
11    var HOST = 'granasat2.ugr.es';
12    var WEBPORT = 9876;
13
14    app.use(require('./apiserver.js'));
15
16    // Default index webpage
17    app.get('*', function(req, res, next) {
```

```
18          res.sendFile(path.join(__dirname + '/../granasat_pm_client/build/index.html'))
19      });
20
21      // Se comprueba si existen los certificados https
22      fs.stat('/certificados/', function(err) {
23          // Si existen certificados, utilizar https
24          if (!err) {
25              console.log('certificate , running https');
26              // Run using https
27              https.createServer({
28                key: fs.readFileSync('/certificados/granasat2_ugr_es.key'),
29                cert: fs.readFileSync('/certificados/granasat2_ugr_es.crt')
30              }, app).listen(WEBPORT, HOST);
31          }
32
33          // Si no existen certificado utilizar http
34          else if (err.code === 'ENOENT') {
35              console.log('certificates does not exist, running http');
36              //Run using http
37              app.listen(WEBPORT, HOST);
38          }
39      });
40
41      console.log("App and API running on " + HOST + ":" + WEBPORT)
```

Listado 4.3 – *Código*

4

4.1.3. Estructurar información en categorías

En esta subsección se va a tratar de mostrar cómo se ha estructurado la información que se muestra, no sólo a través de la interfaz de usuario, si no también en la propia estructura interna de directorios del servidor.

En este caso no sólo se ha tenido que escribir código JS si no también echar mano de código SQL. Se ha creado una tabla llamada “categories”, que contiene todas las categorías que va a existir en el sistema. Estas categorías a su vez se dividen en subcategorías: en la tabla se ha añadido una columna llamada “parentCategory”, si se trata de una subcategoría el contenido de esta columna sera el “id” de la categoría padre y si se trata de una categoría padre será “null”. Para más información consultar el esquema relacional del capítulo de diseño 3.8.

Cada componente va a contener una clave foránea con la subcategoría que pertenece, a nivel de código basta con pasar cómo argumento de búsqueda la categoría, es un código sencillo por lo que no se ve necesario añadirlo.

Lo siguiente por realizar, a petición del cliente, será transformar la antigua estructura de directorios de documentos. Esta antigua estructura, situada en el lado del servidor, caótica y poco intuitiva (los documentos están organizados por carpetas con id) se ha sustituido por una nueva estructura organizada en categorías. Para ello se ha creado un script que realiza dicha transformación, tal y como muestra la siguiente figura 4.1.

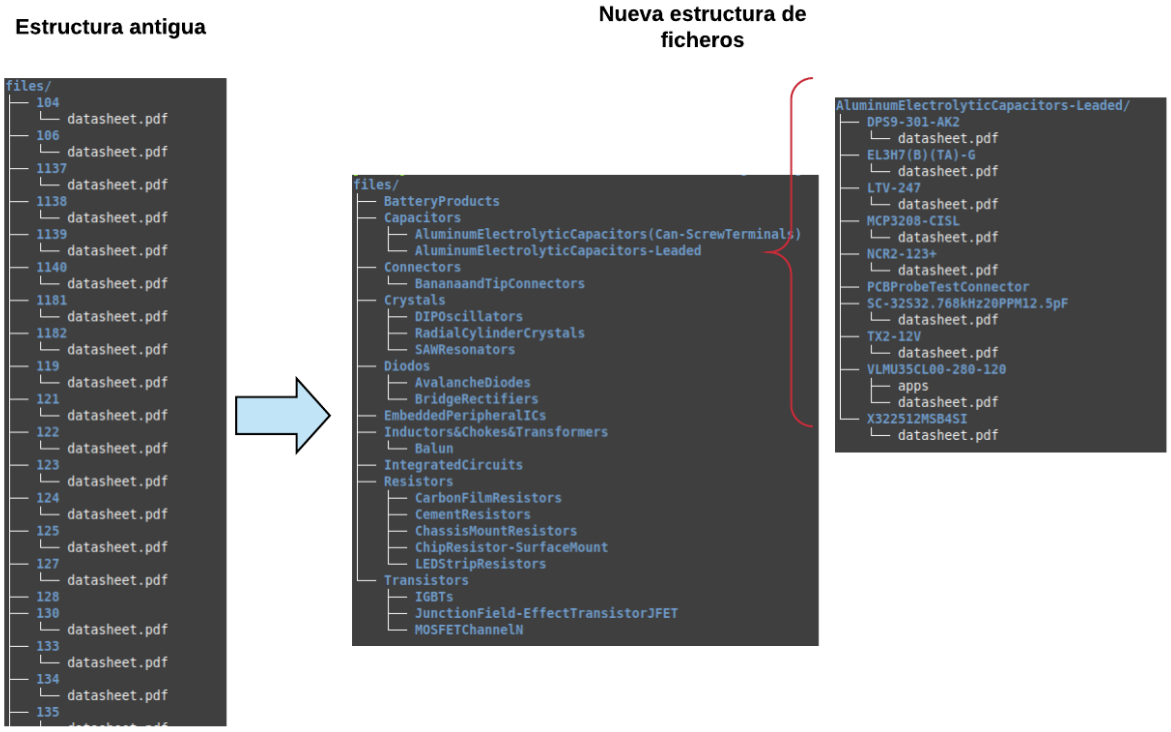


Figura 4.1 – *Conversión de la estructura de directorios de documentos en el servidor*

```
// importar módulos necesarios
var dbManager = require('./db.js')
var fs = require('fs');
var db = require('./dbConnect')
var path = require('path');

var dir = path.join(__dirname, './') + "files/" ;

console.log(dir)

if (!fs.existsSync(dir)) {
    fs.mkdirSync(dir);
}

db.on('error', function (err) {
    logger.error("DATABASE ERROR: " + error)
});

/**************** CREA ESTRUCTURA DE DIRECTORIOS */

var cat_result
// Crea la estructura de categorias en base a la tabla de la bd
new Promise(async (resolve, reject) => {
  db.query("SELECT * FROM categories",
      function (error, results, fields) {
```

```
26          if (error) {
27              logger.error(error)
28              return reject("Categories search error.")
29          } else {
30              return resolve(results)
31          }
32      })
33  }).then( (results)=>{
34    cat_result = results
35    var maincategories = [];
36    cat_result.map( (c)=> {
37      if (c.parent_id == null){
38        maincategories.push(c.name);
39      }
40    })
41
42    maincategories.map( (c)=> {
43        const subdir = dir + c.replace(/\s+/g, '')  // \s-> espacios en blanco, g-> global flag
44        if (!fs.existsSync(subdir)) {
45            fs.mkdirSync(subdir);
46        }
47    })
48  })
49
50 // Por cada stock, comprueba si existen documentos y de existir, los mueve a la nueva estructura
51   new Promise(async (resolve, reject) => {
52     db.query("SELECT * FROM stockcomplete",
53         function (error, results, fields) {
54             if (error) {
55                 logger.error(error)
56                 return reject("Stock search error.")
57             } else {
58                 return resolve(results)
59             }
60         })
61     }).then( (results)=>{
62
63       var stock = results;
64
65       stock.map( s=>{
66
67         new Promise(async (resolve, reject) => {
68           db.query('SELECT files.*,CONCAT(users.firstname," ",users.lastname) as user FROM files LEFT JOIN users
              ON files.creator=users.id WHERE files.part=?',
69              [parseInt(s.idpart)],  function (error, results, fields) {
70                    if (error) {
71                        logger.error(error)
72                        return reject("Categories search error.")
73                    } else {
74                        return resolve(results)
75                    }
76                })
77             }).then ((results)=>{
78               var files = results
79
80               if(files.length > 0){
81                 var newdir = dir + s.parent_category.replace(/\s+/g, '')+ "/" + s.category.replace(/\s+/g, '')
82                 var olddir =  path.join(__dirname, '../granasat_pm_client/public/static') + "/files/" + s.idpart
83
84                 // Comprueba si existe el directorio para la subcategoria, si no existe lo crea
85                 if (!fs.existsSync(newdir)) {
86                     fs.mkdirSync(newdir);
87                 }
88
89                 newdir = newdir + "/"+ s.name.replace(/\s+/g, '')
90
91                 // Comprueba si existe el directorio para el componente en cuestio, si no existe lo crea
92                 if (!fs.existsSync(newdir)) {
93                     fs.mkdirSync(newdir);
94                 }
95                 files.map( (f) => {
96                   const newdirfile = newdir + "/" +  f.name;
97                   const olddirfile = olddir + "/" + f.name;
98                          var source = fs.createReadStream(olddirfile);
99                          var dest = fs.createWriteStream(newdirfile);
100                         source.pipe(dest);
101                         source.on('end', function() { });
102                     source.on('error', function(err) {  });
103                 })
104                 console.log("Old dir :" + olddir );
105                 console.log("New dir :" + newdir );
106               }
107             })
108     })  })
```

4.1.4. Integración de escáner QR

El objetivo de esta subsección es dotar al sistema de la capacidad de identificar los elementos de stock por medio de un código QR. De esta forma se facilitará la identificación de los componentes electrónicos tras etiquetarlos con estos códigos. El motivo de no utilizar un código de barras es sencillo, es que un código QR es más compacto y es leído más rápido.

Para conseguir identificar de manera única cada stock del almacén y mostrar su información en la aplicación web se ha realizado:

- **Creación de una ruta única para cada stock**, para ello se ha añadido una ruta a la aplicación utilizando el módulo "react-router-dom" [14]. La página principal se ejecutará en la ruta "/" mientras que la información de stock se mostrará en la ruta "/stock/<id>". Para ello se ha añadido en el archivo principal las siguientes líneas de código **??**.

```
1    const routing = (
2      <Router>
3        <Switch>
4          <Route exact path="/" component={App} />
5          // Esta es la forma de pasar un parametro mediante Route
6          <Route path="/stock/:id" render={({match})=> <App showStock={match.params.id} />}/>
7          <Route component={App}/>
8       </Switch>
9       </Router>
10   )
11   \label{listing:routes}
```

Listado 4.4 – *Código*

  Aclarar que se ha decidido crear una ruta por stock, ya que permitirá en un futuro acceder a esta información sin tener la página web abierta, escaneando el código QR directamente desde un "smartphone" (de hecho actualmente se puede realizar, pero no es aconsejable ya que la página web aún no esta adaptada a móviles).

- **Permitir desde la web usar un escáner usb**, para ello se ha creado una vista llamada "Scanner.jsx", que utilizando el módulo "react-barcode-reader" [13] espera a que se capture un código QR y redirige a la página con la información.

```
1        import React, { Component } from 'react'
2        import BarcodeReader from 'react-barcode-reader'
3
4        class Scanner extends Component {
5          constructor(props){
6          super(props)
7            this.state = {
8            result: 'No result',
9          }
10           this.handleScan = this.handleScan.bind(this)
11       }
```

```
12          handleScan(data){
13            this.setState({
14            result: data,
15            })
16            // Abre la página que se ha leido con el escaner
17            window.open(this.state.result)
18          }
19          handleError(err){
20            console.error(err)
21          }
22          render(){
23            return(
24              <div>
25                <BarcodeReader
26                  onError={this.handleError}
27                  onScan={this.handleScan}
28                  preventDefault= {true}
29                  />
30                  <div id="scan">
31                    <div> <p>Realice el escaneado del código QR ahora</p> </div>
32                    <div class="qr-img">
33                      <div class="scan-ani">
34                      </div>
35                    </div>
36                  </div>
37            </div>
38                  )
39                }
40              }
41          export default Scanner;
```

Listado 4.5 – *Contenido de Scanner.jsx*

4

- **Generación de un código QR**, por último queda mostrar cómo los códigos QR son generados. Para ello se ha hecho uso del módulo "qrcode-react" [12] de la forma que muestra el siguiente código **??**.

```
1         <QRCode value={window.location.protocol+"//" +document.domain+":" + window.location.port + "/stock/
           "+this.state.transactionstock.id} logo="/public/granasatlogo.png"></QRCode>
2         \label{listing:generarqr}
```

Listado 4.6 – *Trozo de código para generar un código QR.*

Notaciones sobre el código:

- "window.location" y document son variables utilizadas para obtener información sobre la página en la que se ejecuta, como el protocolo,el puerto o el dominio.
- La variable de estado "transactionsstock" contiene información sobre el stock que se va a mostrar. En este caso se obtiene el id para forma el URL.
- Con el parámetro "logo" podemos incrustar una imagen en el código QR.

El resultado sería un código QR que contiene un URL con la información sobre un stock determinado, por ejemplo "https://granasat2.ugr.es:9876/stock/170".

Figura 4.2 – *código QR generado por el sistema para identificar stock*

4.1.5. Extracción de datos con extensión de Google Chrome

Uno de los requerimientos del proyecto es la extracción de datos de componentes electrónicos de las páginas web de proveedores, y de esta forma poder "pegar"' estos datos en el sistema ,en el momento de crear un nuevo componente. Específicamente se pide una extensión para el navegador Google Chrome.

La extensión utiliza jquery y la librería "clipboardjs" [6] para copiar información al portapapeles. Una parte importante de una extensión para Chrome es el "manifest" que indica al navegador toda la información que necesita como puede ser el nombre, versión, script o dominio web que afecta entre otras cosas.

```
{
  "manifest_version": 2,

  "name": "Component Data Exporter",
  "description": "Copy product from RS, Mouser, Farnell,Avnet,Lcsc, Digikey to Excell",
  "version": "1.1",

  "browser_action": {
   "default_icon": "icon.png"
  },

  "icons": { "16": "icon.png",
             "48": "icon.png",
            "128": "icon.png" },

  "content_scripts": [
  {
    "matches": ["https://es.rs-online.com/*","https://www.mouser.es/*","https://eu.mouser.com/*","http://es.
     farnell.com/*","https://es.farnell.com/*","https://www.arrow.com/*","https://www.digikey.es/*","https://www.
     avnet.com/*","https://lcsc.com/*"],
    "css": ["mystyles.css"],
    "js": ["jquery.js","clipboard.js", "myscript.js"],
    "run_at": "document_end"
  }
],
"background": {
  "scripts": ["background.js"],
  "persistent": false
},
"permissions": ["tabs", "https://es.rs-online.com/*","https://www.mouser.es/*","http://es.farnell.com/*","https://
  www.arrow.com/*","https://www.digikey.es/*","https://www.avnet.com/*"]

}
```

Listado 4.7 – *Manifest de extensión de Google Chrome*

La funcionalidad de la extensión viene definida en un script, que se encarga de extraer del código HTML de la página web la información necesaria del componente. Un trozo de código que se ha implementado para ello sería el siguiente.

```
var code = $( "div.detail-info h1.info-title").text().trim()

  if(code != ""){
    $( "div.detail-info").prepend( '<div id="msg"><p>Pagina soportada por RS Exporter</p> <button id="copyExcel">
     COPY EXCEL</button><button id="info">Info</button></div>' )

    $("#info").on('click',function(){
      alert(info)
    })
    new Clipboard('#copyExcel', {
    text: function(trigger) {
        var table = $("table.info-table tbody")[0]
       var fab = table.rows[0].getElementsByTagName("td")[1].getElementsByTagName('a')[0].innerHTML.trim()
        code =  table.rows[1].getElementsByTagName("td")[1].innerHTML.trim()
        var codefab = code
        var description = table.rows[7].getElementsByTagName("td")[1].getElementsByTagName("p")[0].innerHTML.trim
          ()
        var price = $( "span.unitPrice").text().trim()
        var minquantity = $( ".main-step span.step-min").text().split(":")[1].trim()

        var pdf = table.rows[5].getElementsByTagName("td")[1].getElementsByTagName("p")[0].getElementsByTagName("a
          ")[0].getAttribute('href')
        var image = document.getElementsByClassName("main-img")[0].src

        alert("copiado")
        return ["Lcsc",url,code,codefab,fab,description,price,minquantity,pdf,image].join("\t");
      }
    });
  }

  \label{listing:script_categorias}
```

En el script anterior (**??**, lo que se hace es extraer la información necesaria a partir de cómo se estructuran los elementos HTML para esa determinada web.Una vez se ha copiado la información, para "pegarla" en la aplicación web, se ha añadido un "pasteListener". Este listener estará a la espera de que se copie algo y "rescatará" los datos del portapapeles.

Capítulo 5

Resultados

Este capítulo se centrará en mostrar los resultados obtenidos tras la implementación del código. Estos resultados se mostrarán desde el punto de vista del usuario, es decir, mediante la interfaz de usuario una de las partes más importantes de un servicio web. En resumen, se mostrarán las funcionalidades principales del sistema y cómo el usuario accederá a ellas.

5.1. Introducción

Uno de los objetivos del proyecto es crear una interfaz amigable, intuitiva y fácil de utilizar. Para ello se ha optado por la simplicidad, ya que demasiados elementos podrían abrumar de alguna forma al usuario.

Esta sección se va a dividir en bloques basándose en la funcionalidad:

- Estructura general(5.2).
- Gestión de inventario(5.3).
- Gestión de categorías(5.4).
- Gestión de usuarios(5.6).
- Gestión de lugares de almacenamiento (5.7).
- Gestión de archivos(5.8).
- Escáner QR (5.9).
- Extensión de Google Chrome(5.10).

5

Se encuentra disponible un vídeo de algo menos de 6 minutos, que resume las funciones que se muestran a continuación `https://youtu.be/x4tI51J-340`. No todo lo que se muestra en el vídeo ha sido implementado en este proyecto (el vídeo pretende mostrar la aplicación web en su totalidad), más adelante se indica que es lo que se ha añadido durante este proyecto y que funciones estaban antes.

5.2. Estructura general

La estructura general de la página es bastante sencilla, consiste en un encabezado simple con un panel para cerrar sesión, una barra de navegación desde donde se podrá acceder a la mayoría de funciones y en el centro el contenido principal que variará en función de que vista este mostrando la aplicación.

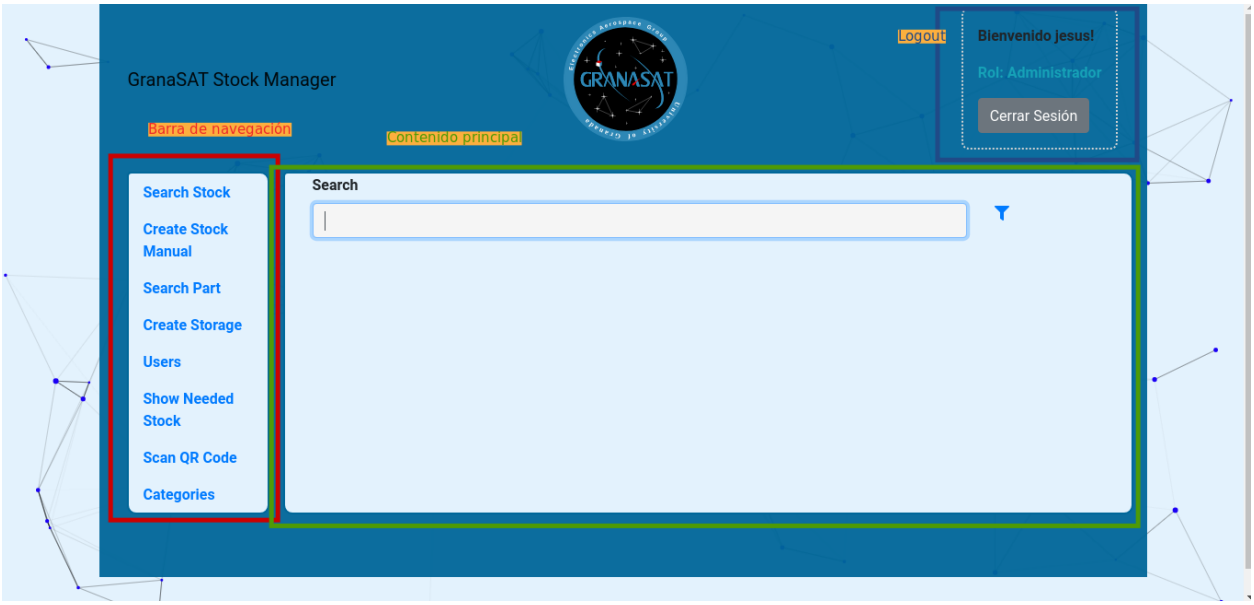


Figura 5.1 – *Interfaz de usuario, pantalla principal*

La barra de navegación anterior muestra todas las funcionalidades ya que es una cuenta de "Administrador", si accedemos con una cuenta de usuario normal, su aspecto sería el de la figura 5.2 (sólo se podría acceder a las funciones "Search Stock" y "Scan QR Code").

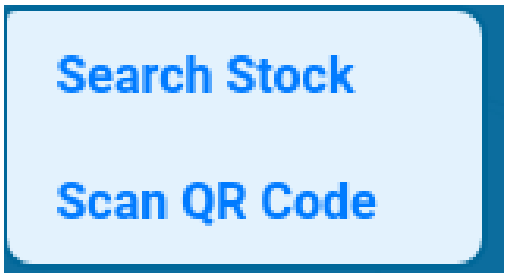


Figura 5.2 – *Interfaz de usuario, barra de navegación para usuario normal*

5.3. Gestión de inventario

En esta sección se mostrarán los resultados relativos a las funciones que se pueden realizar sobre los componentes electrónicos.

5.3.1. Búsqueda de stock

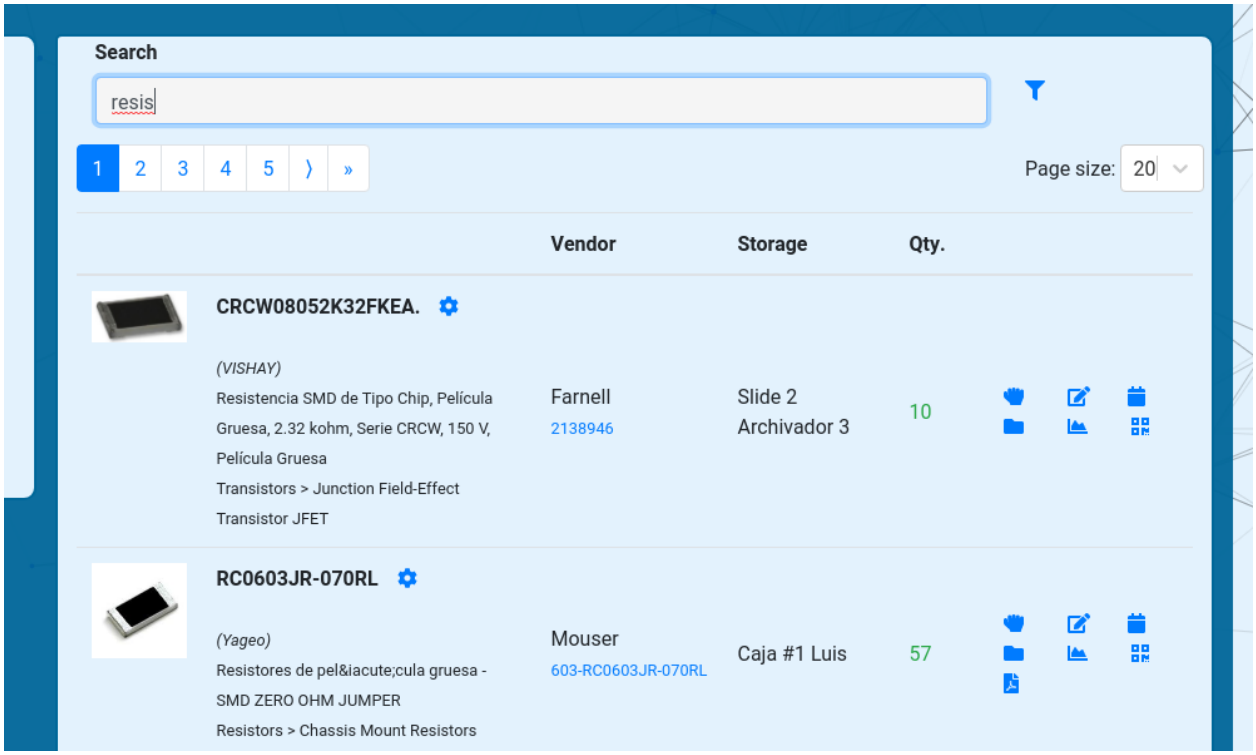


Figura 5.3 – *Interfaz de usuario, búsqueda de stock*

Se podría decir que esta es la principal función del sistema, a través de la barra de búsqueda se puede consultar el stock en función de un texto y en caso de darse unos parámetros. Esta función ya se encontraba implementada, pero en este proyecto **se ha añadido** paginación, de tamaño variable (20 elementos por página por defecto).

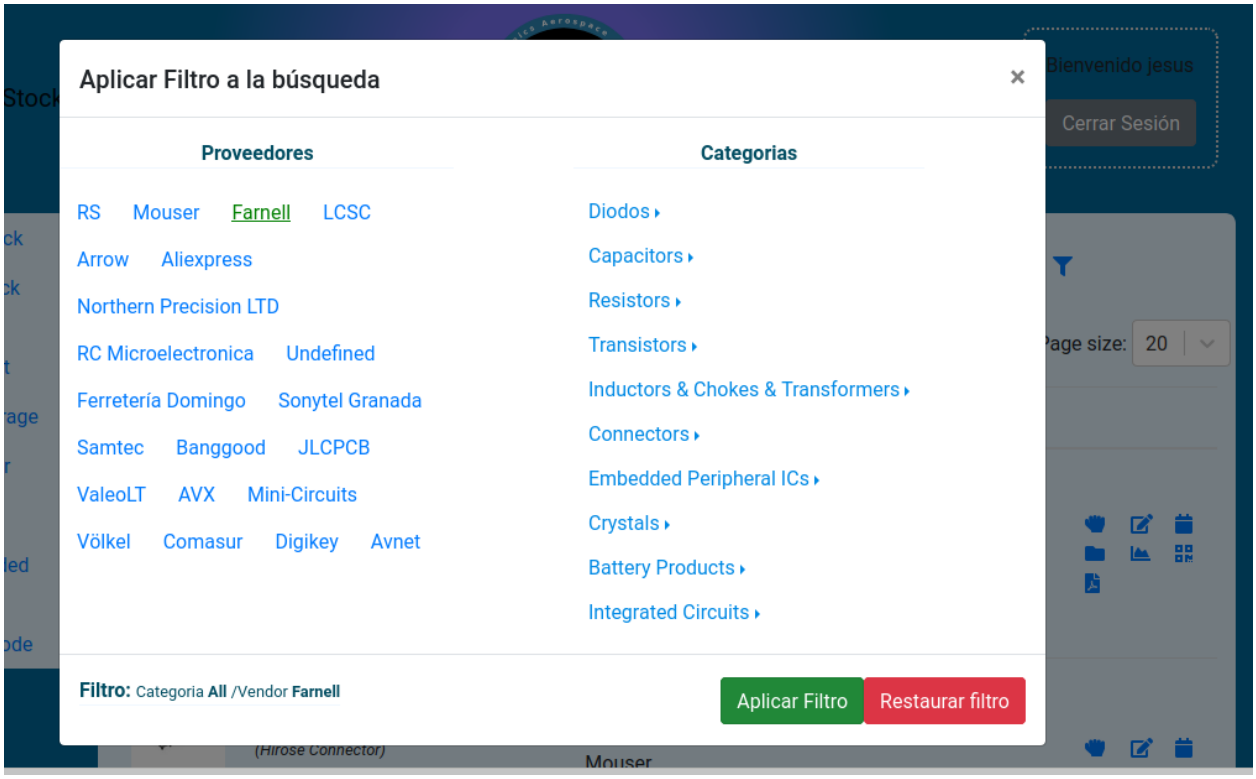


Figura 5.4 – *Interfaz de usuario, filtro de búsqueda de stock*

Otro aspecto que **se ha añadido** a la búsqueda en este proyecto es el filtrado, tras pulsar el icono de filtro (botón a la derecha de la barra de búsqueda) aparecerá la ventana de la imagen anterior, permitiendo filtrar la búsqueda por proveedor o categoría.

5.3.2. Creación de stock

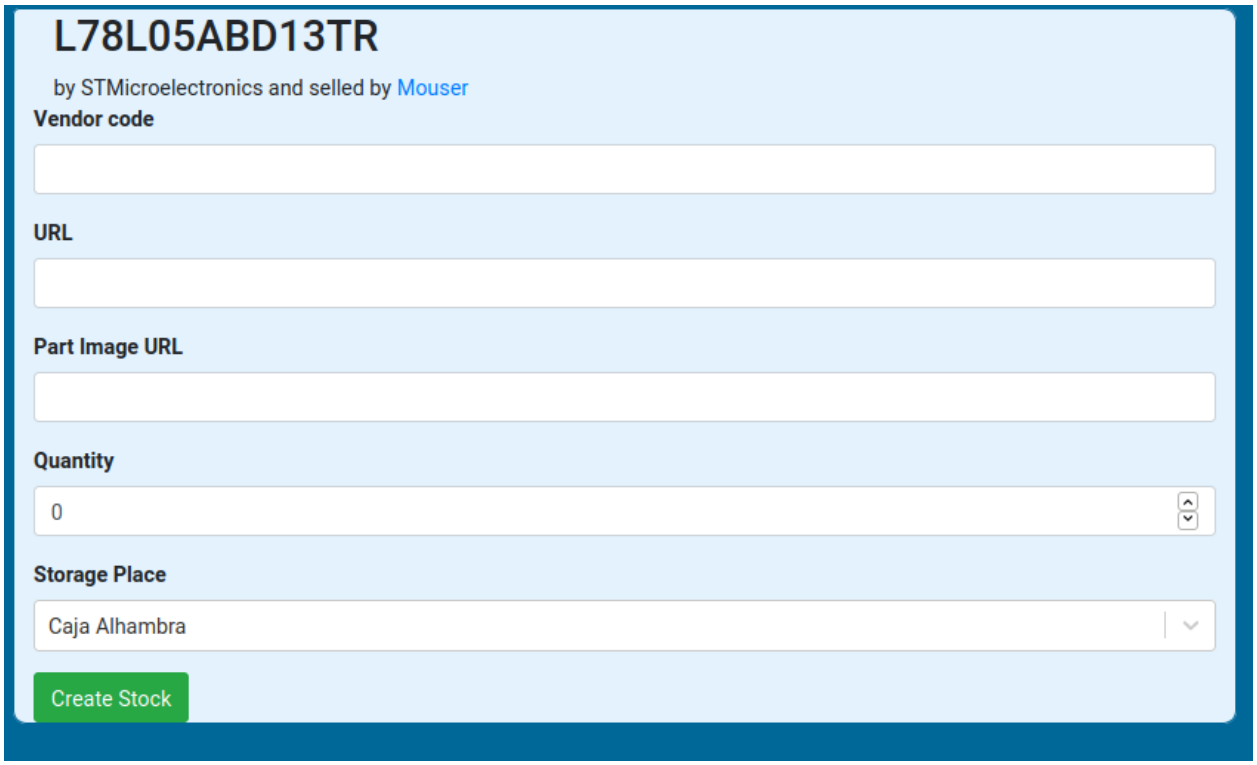


Figura 5.5 – *Interfaz de usuario, crear stock*

5

Para crear un stock es necesario seleccionar anteriormente un componente (“part’”) y un proveedor (esta función ya se encontraba implementada).

5.3.3. Modificar y eliminar stock

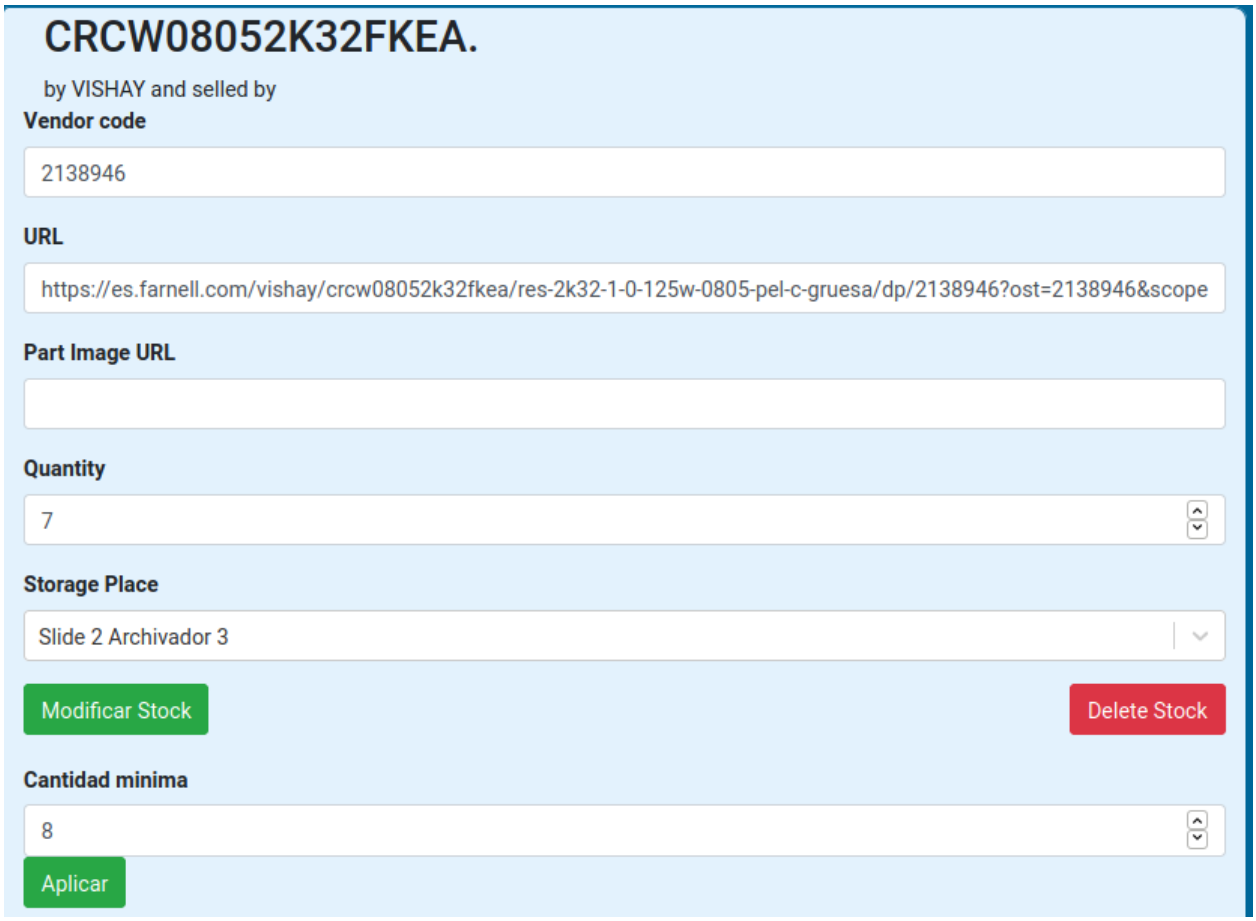


Figura 5.6 – *Interfaz de usuario, modificar stock*

Se ha añadido el formulario anterior 5.6 con el que se puede modificar el stock o eliminarlo (con el botón rojo "Delete Stock").

5.3.4. Búsqueda de componente

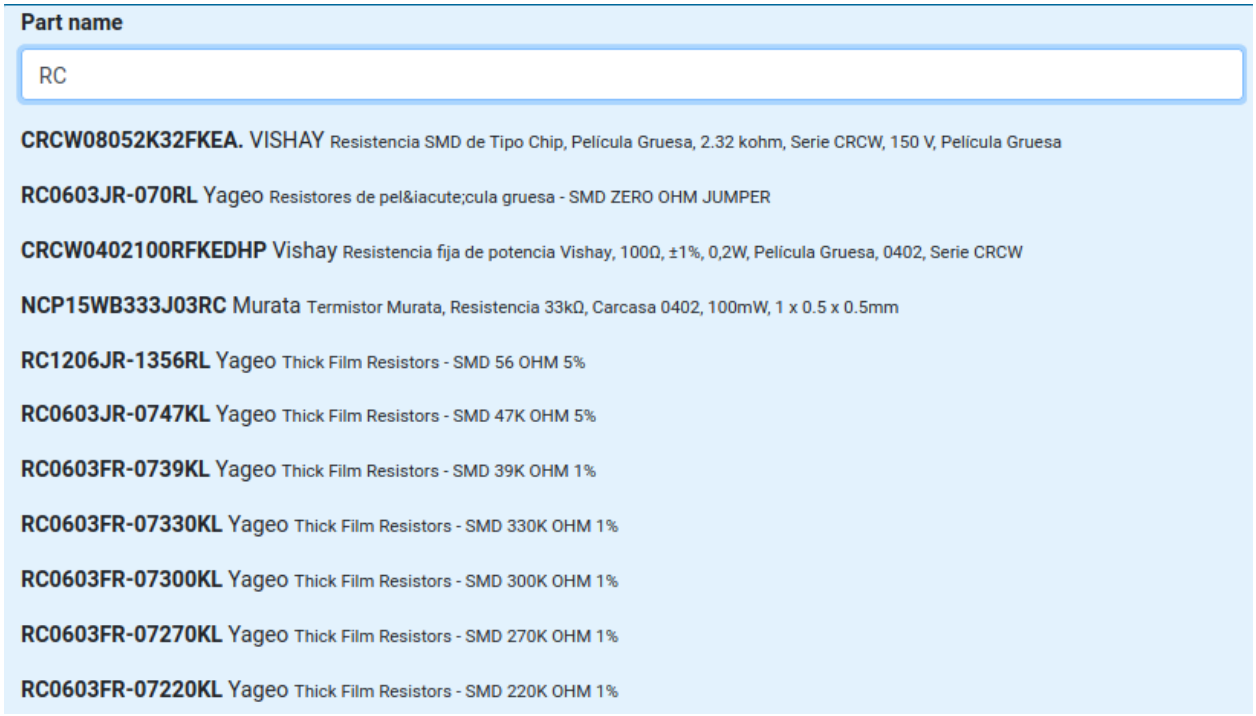


Figura 5.7 – *Interfaz de usuario, búsqueda de componente*

5.3.5. Creación de componente

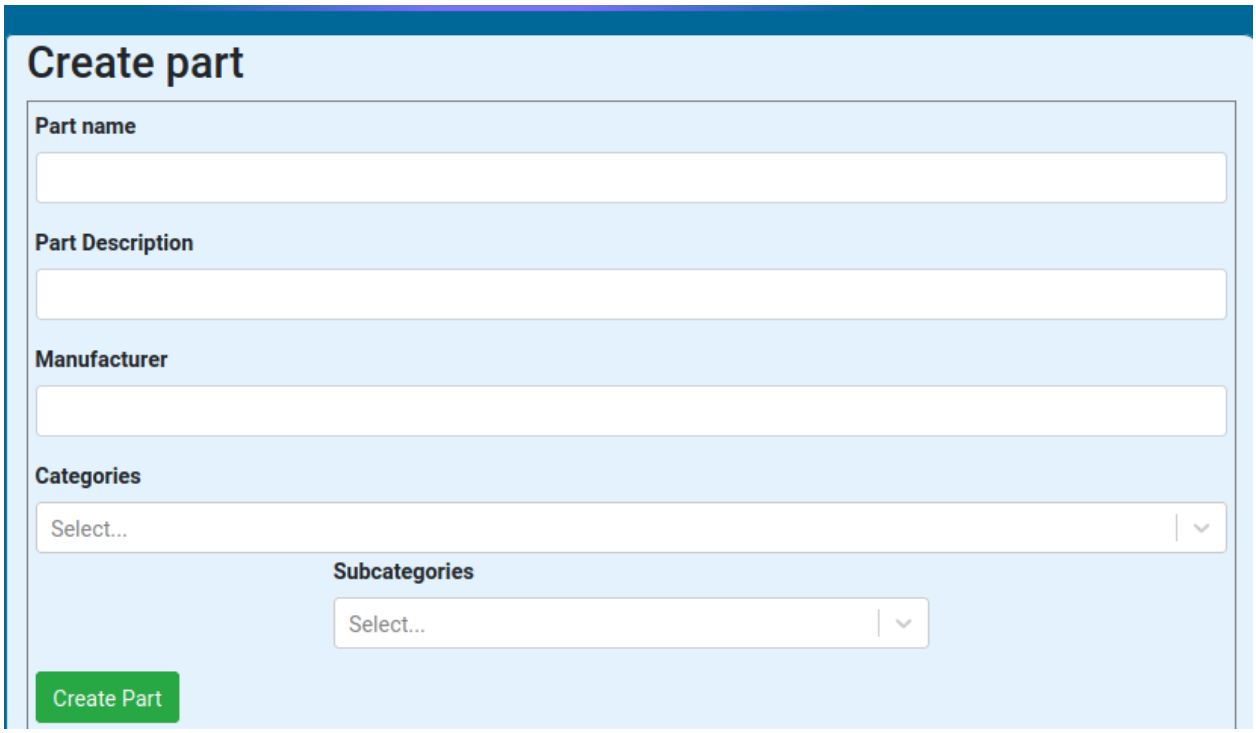


Figura 5.8 – *Interfaz de usuario, creación de componente*

5.3.6. Stock bajo mínimo

Se ha modificado el sistema de forma que cuando la cantidad de un stock baja de un mínimo fijado, se mostrará la siguiente alerta a todos los administradores.

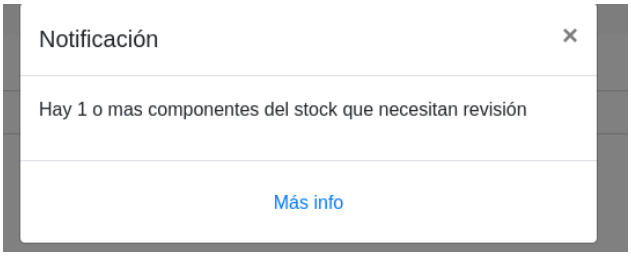


Figura 5.9 – *Interfaz de usuario,alerta stock bajo mínimo*

Si pulsamos “más información” nos dirigirá a la vista "NeededStock"(accesible también mediante el panel de navegación). En esta vista se podrán visualizar todos los componentes bajo mínimo, además si se pulsa el botón "Generar hoja de cálculo"se

puede descargar una hoja de cálculo con los datos del stock bajo mínimo.

Figura 5.10 – *Interfaz de usuario, lista stock bajo mínimo*

5.4. Gestión de categorías

Todo lo relacionado a categorías **se ha implementado desde cero**. La información (el stock) como ya se comentó anteriormente se estructurará en categorías. Ha sido creada una vista para poder añadir nuevas categorías y/o editar las existentes.

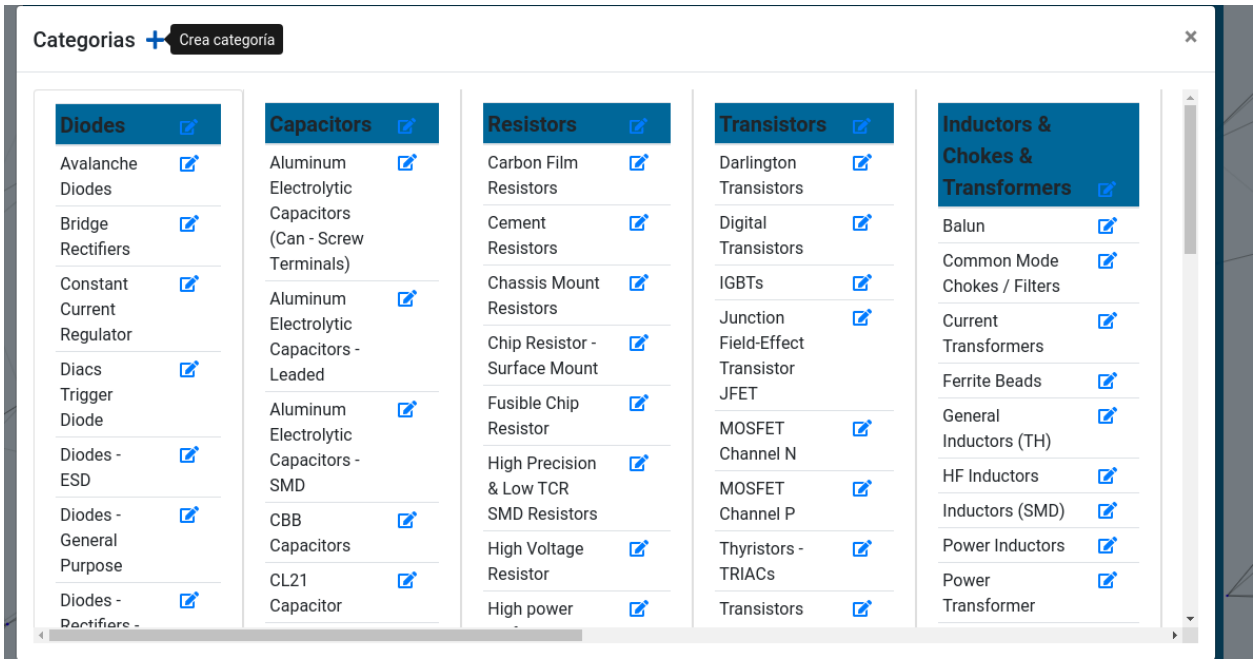


Figura 5.11 – *Interfaz de usuario, lista de categorías*

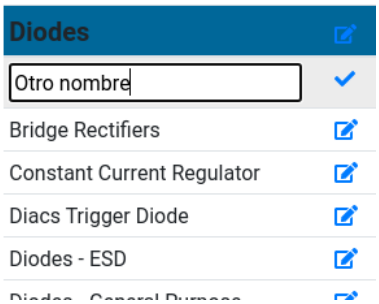


Figura 5.12 – *Interfaz de usuario, renombrar categoría*

5.5. Gestión de transacciones

Entendiendo como transacción la modificación de cantidad de stock por parte de un usuario, se tiene:

5.5.1. Añadir o retirar stock

Se accede a esta función a partir del siguiente botón, ligado a cada stock.

Figura 5.13 – *Interfaz de usuario, botón iniciar transacción*

Tras pulsarlo nos aparecerá la siguiente ventana, que permite modificar la cantidad del stock.

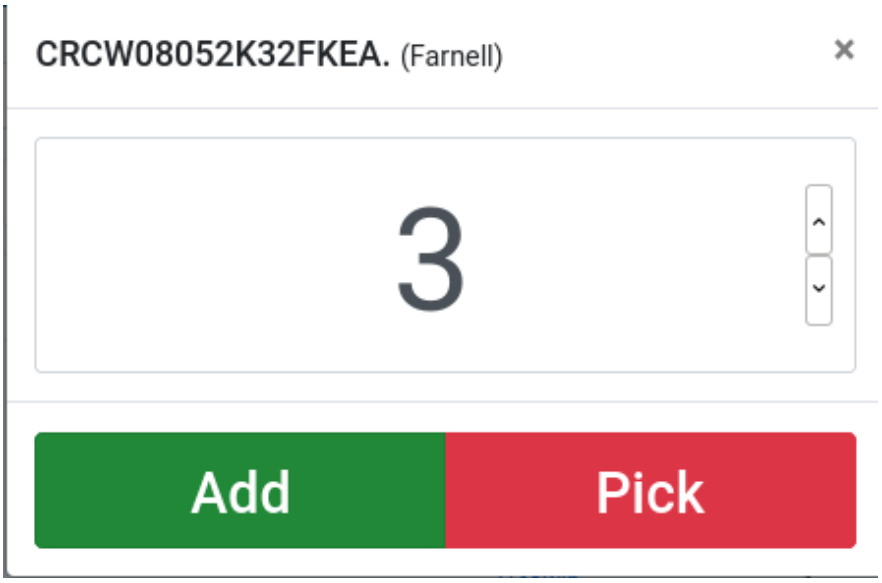


Figura 5.14 – *Interfaz de usuario, realizar transacción*

5.5.2. Lista de transacciones

En el sistema también se tiene la opción de ver el historial de transacciones, de esta forma poder llevar un seguimiento de quien retira o añade algún stock.

Figura 5.15 – *Interfaz de usuario, botón historial transacciones.*

CRCW08052K32FKEA. (Farnell) ×

07/07/20 11:24	Jesus TFG	▲ 3
06/07/20 17:50	Jesus TFG	▼ 9
29/06/20 17:03	Jesus TFG	▼ 0
29/06/20 17:00	Jesus TFG	▲ 2
29/06/20 16:59	Jesus TFG	▼ 1
30/05/20 14:07	Jesus TFG	▼ 0
26/03/20 18:46	Jesus TFG	▲ 1
26/03/20 18:46	Jesus TFG	▲ 11
21/03/20 19:54	Jesus TFG	▼ 1
21/03/20 19:53	Jesus TFG	▼ 1
21/03/20 19:53	Jesus TFG	▲ 1
21/03/20 19:52	Jesus TFG	▲ 1
21/03/20 19:51	Jesus TFG	▲ 1
21/03/20 19:50	Jesus TFG	▲ 1
21/03/20 18:18	Jesus TFG	▼ 1
21/03/20 18:17	Jesus TFG	▼ 5
20/03/20 21:52	Jesus TFG	▲ 1

Figura 5.16 – *Interfaz de usuario, historial de transacciones.*

5.5.3. Gŕafico cantidad de stock

Otra utilidad que **se ha añadido** para facilitar el seguimiento de las transacciones es un gráfico, con las variaciones de cantidad de stock en los últimos 7 días. Esto permite ver las transacciones de una manera sencilla y visual.

Figura 5.17 – *Interfaz de usuario, botón gŕafico de transacciones.*

Figura 5.18 – *Interfaz de usuario, gráfico de la cantidad de stock en los últimos 7 días.*

5.6. Gestión de Usuarios

Todo lo relativo a manipulación de usuarios (crear usuarios, modificar usuarios o eliminar usuarios) será llevado a cabo desde la vista "Users". Como ya se ha comentado, sólo los administradores tendrán acceso a esta vista. De esta parte sólo se

encontraba implementada la creación de usuarios y la acción de acceder al sistema, el resto se ha desarrollado en este proyecto.

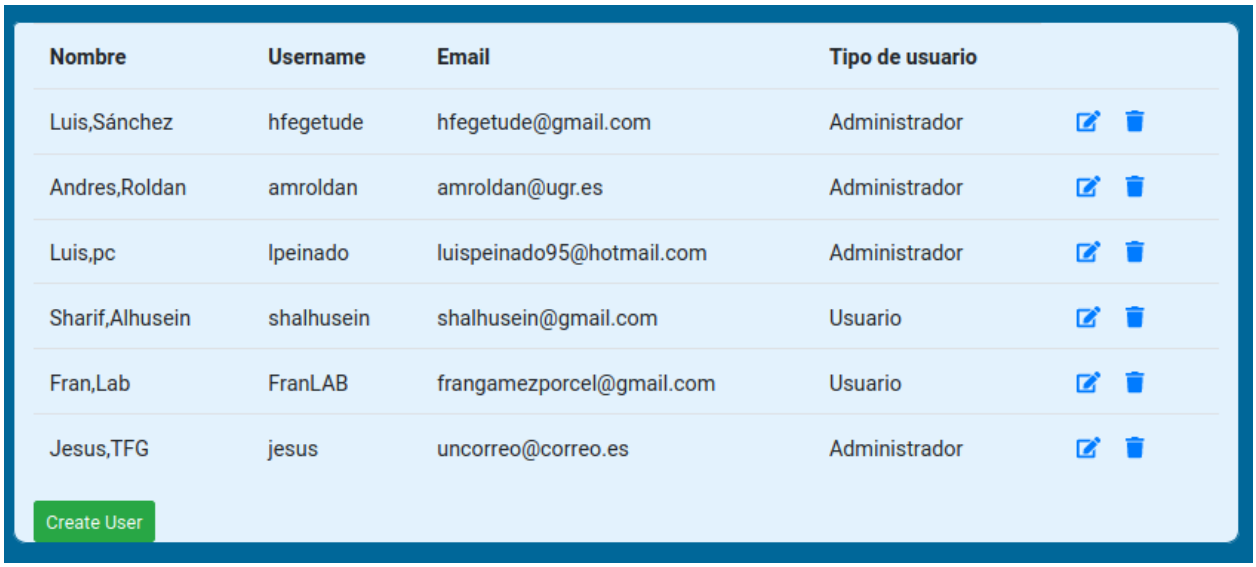


Figura 5.19 – *Interfaz de usuario, vista de usuarios.*

5.6.1. Dar de alta a un usuario

Para crear un usuario se deberá rellenar el siguiente formulario. No admite ningún campo vacío o una contraseña menor a 4 carácteres.

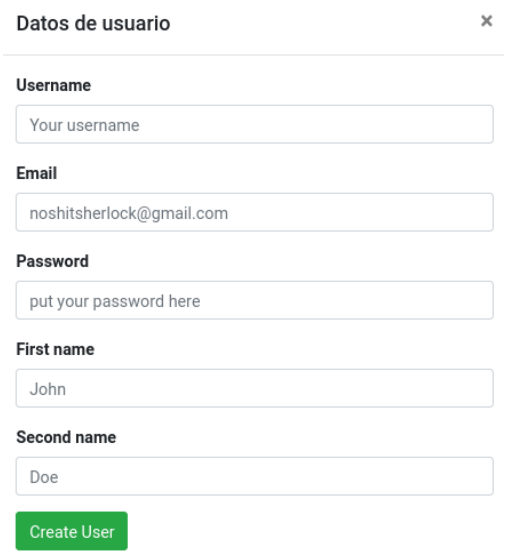


Figura 5.20 – *Interfaz de usuario, crear usuario.*

5.6.2. Modificar y eliminar usuarios

Cada usuario en la vista de usuarios tendrán asociados un botón para editar y otro para eliminar.

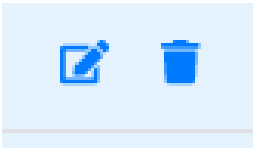


Figura 5.21 – *Interfaz de usuario, botones editar y eliminar de la vista de usuario.*

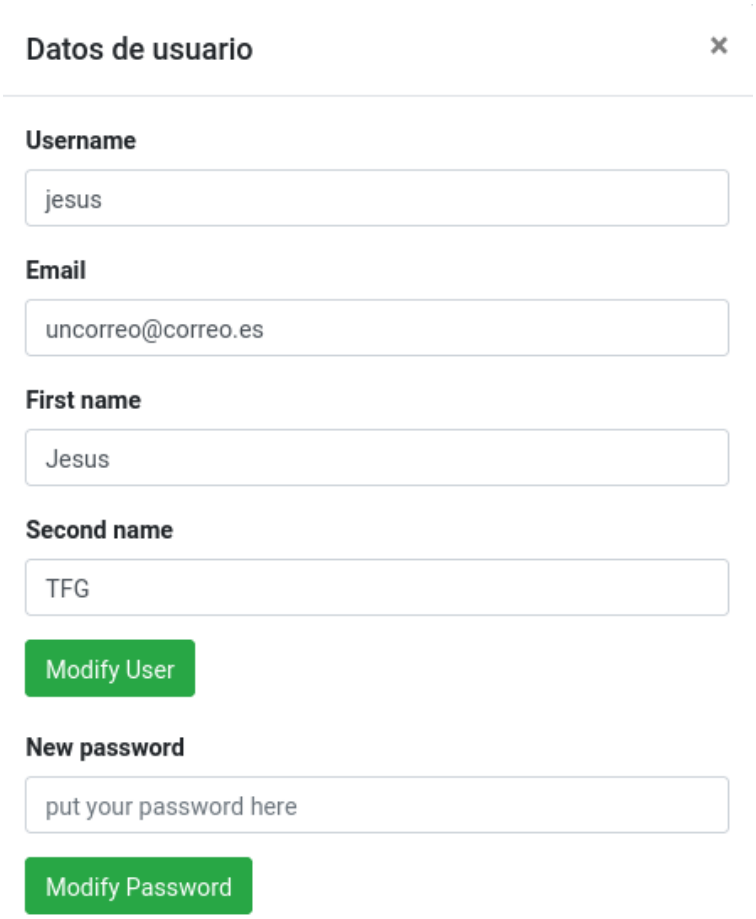


Figura 5.22 – *Interfaz de usuario, modificar usuario.*

Aclarar que el campo contraseña se trata por separado del resto de datos del formulario. Un administrador va a poder modificar la contraseña de cualquier usuario normal y su propia contraseña (no va a poder modificar la contraseña de otros administradores).

5.6.3. Acceder al sistema y desconectarse del sistema

Para acceder al sistema se dispondrá del clásico formulario de acceso "usuario/contraseña".

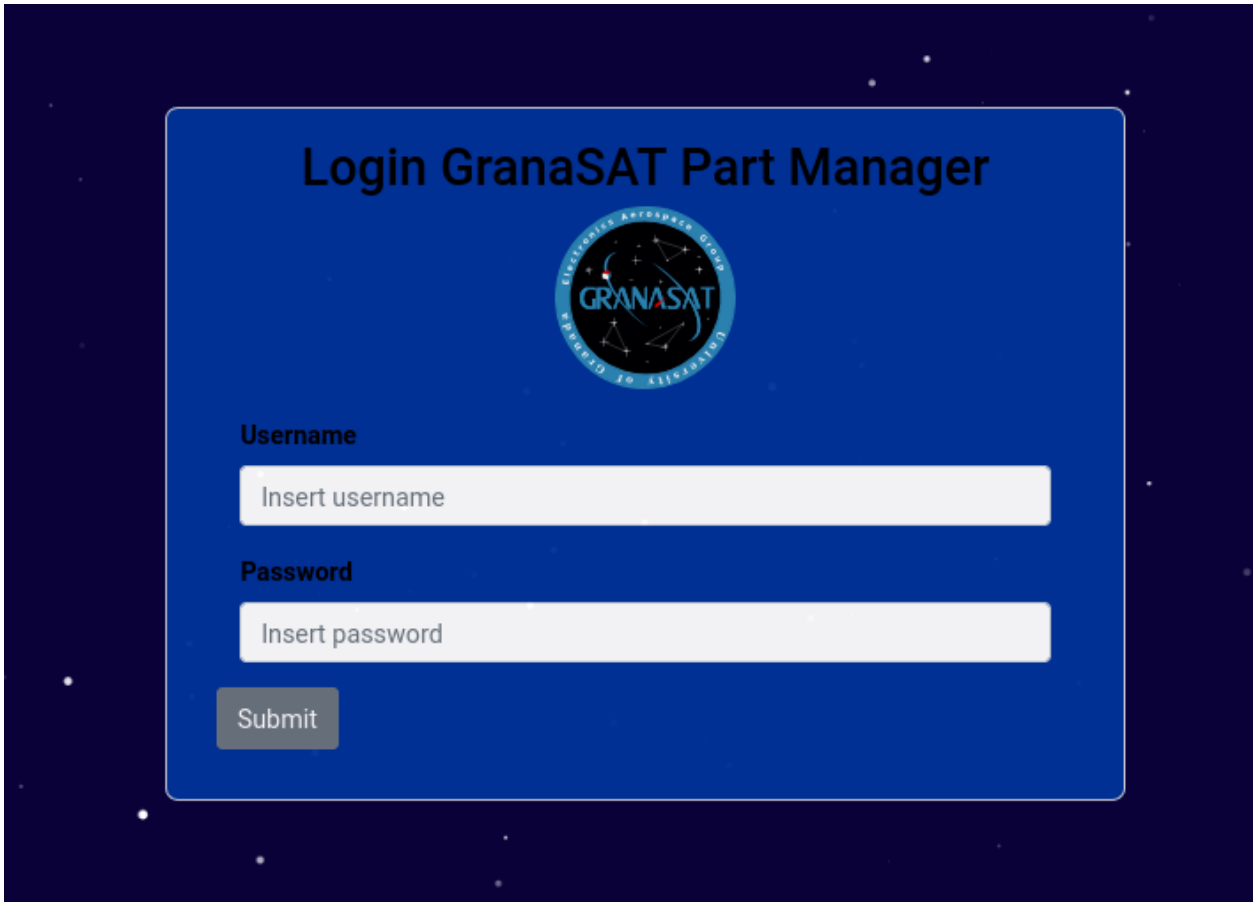


Figura 5.23 – *Interfaz de usuario, acceder al sistema.*

Para permitir la desconexión, se ha habilitado un botón en la esquina superior derecha de la vista general de la página.

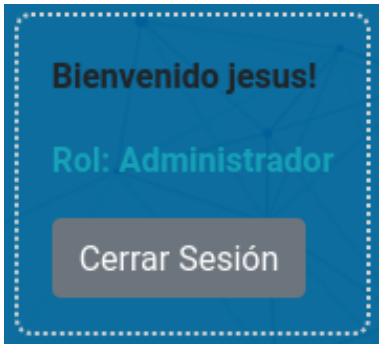


Figura 5.24 – *Interfaz de usuario, desconectarse del sistema.*

5.7. Gestión de lugares de almacenamiento

5.7.1. Crear lugar de almacenamiento

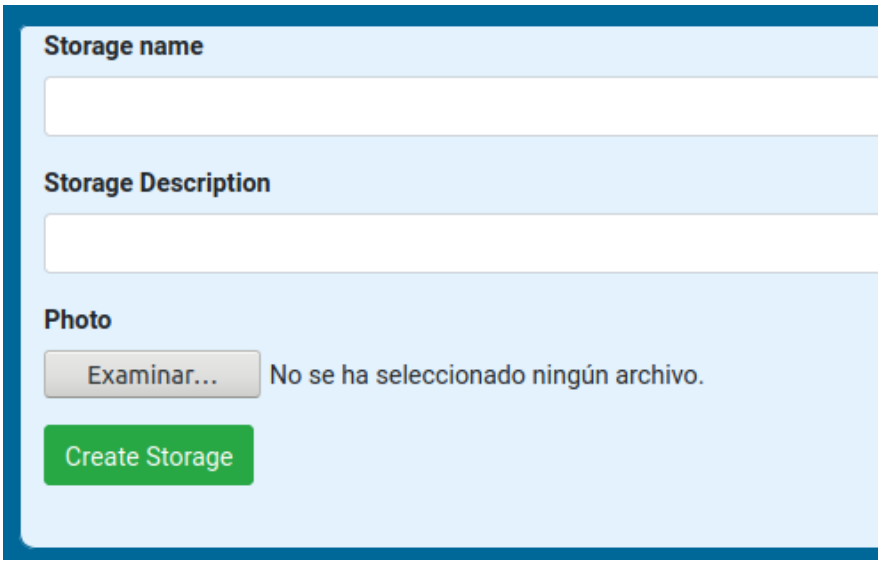


Figura 5.25 – *Interfaz de usuario, crear lugar de almacenamiento.*

5.8. Gestión de archivos

El sistema va a permitir adjuntar a cada componente archivos de texto, el resultado desde el punto de vista del usuario es el siguiente.

5.8.1. Descargar o subir archivos

En la vista de búsqueda de stock, existe un botón adjunto a cada stock que va a permitir acceder al siguiente ventana.

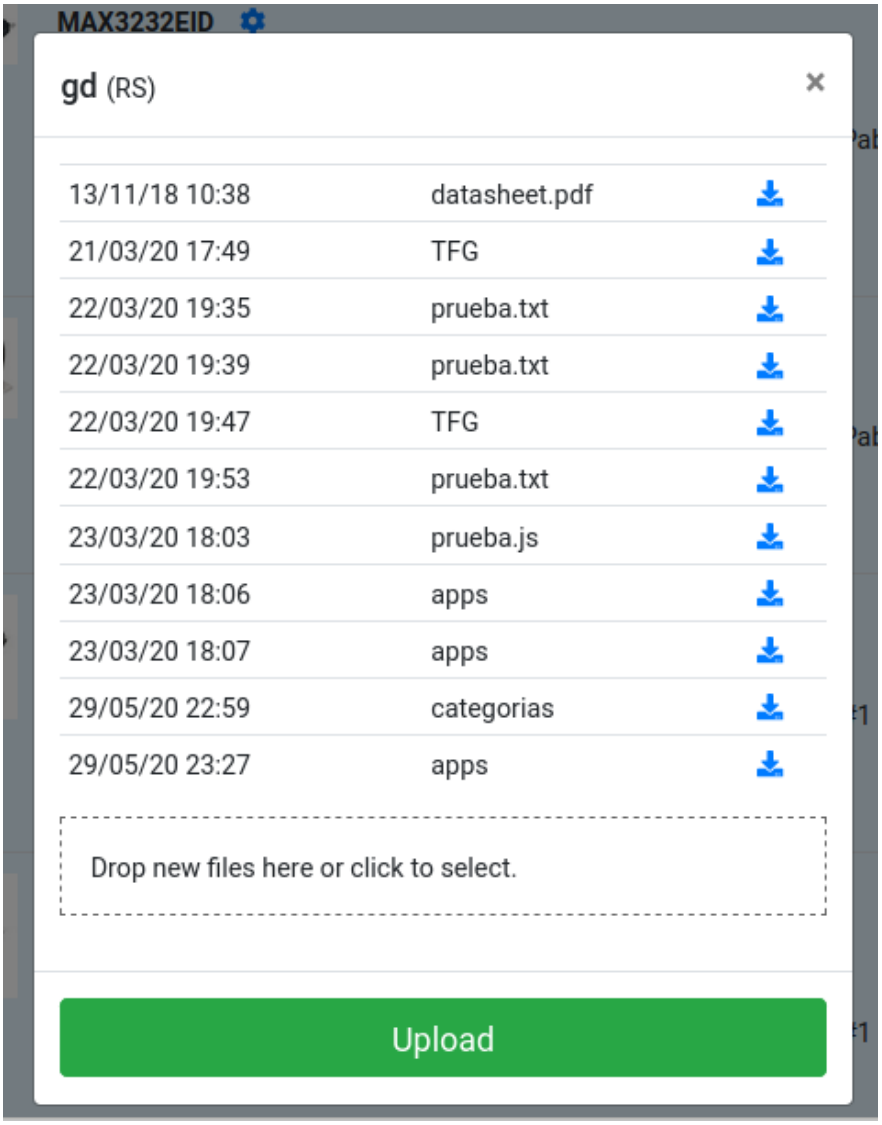


Figura 5.26 – *Interfaz de usuario, subir o descargar documentos adjuntos a un componente.*

Como se puede observar, se permite descargar cualquier archivo ya adjunto al componente o subir uno nuevo desde el sistema.

5.9. Códigos QR

Cómo ya se explicó en el capítulo de implementación (todo esto **se ha implementado desde cero**) , cada stock va a tener asociada una ruta con información del mismo, así cómo un código QR con dicha ruta, tal y cómo se ilustraba en la figura 4.2.

5.9.1. Vista de stock

Desde la página principal existe una ventana que permite leer un código QR a través de un escáner usb. Tras leer un código QR válido, el sistema abrirá una nueva ventana con la vista del stock.

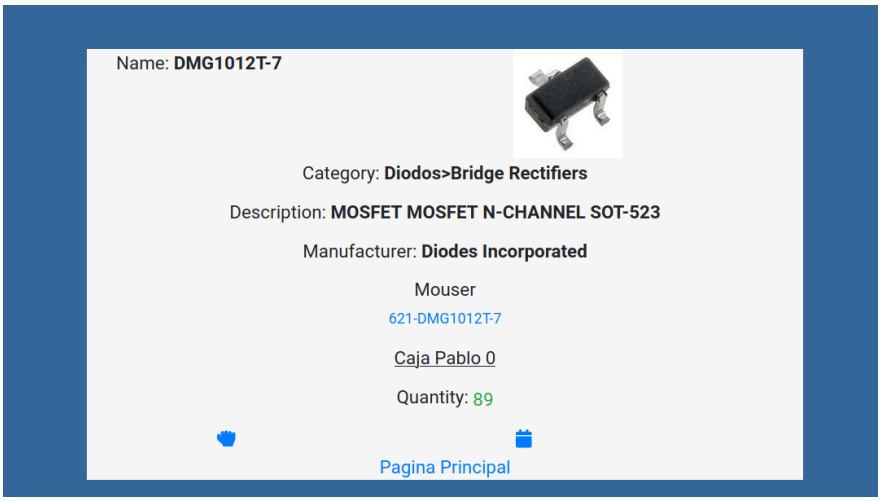


Figura 5.27 – *Interfaz de usuario, vista de stock.*

En esta vista permite realizar algunas operaciones relacionadas con transacciones.

Breve vídeo que muestra el funcionamiento del escaneado de códigos QR en PartManager `https://youtu.be/U5aKJY2VMLY`.

5.10. Extensión de Google Chrome

Figura 5.28 – *Icono de "Component Data Exporter".*

La extensión "Component Data Exporter" (que es el resultado de la actualización de una extensión anterior) es bastante útil para poder copiar los datos directamente desde la página de los proveedores al PartManager, lo que ahorra tener que introducir manualmente todos los datos comunes como puede ser el nombre, fabricante, proveedor, URL de la imagen, pdf de hoja de detalles técnicos, etc.

Es muy sencillo de utilizar, simplemente nos situamos en una página web de un proveedor reconocido por la extensión (los proveedores reconocidos son: "RS" [18], "Farnell" [4], "Mouser" [8], "Avnet" [1], "Lcsc" [5] y "Digikey" [3]) , pulsamos "Copy Excell" como muestra la figura 5.29 y una vez copiado el componente nos vamos a PartManager y "pegamos".

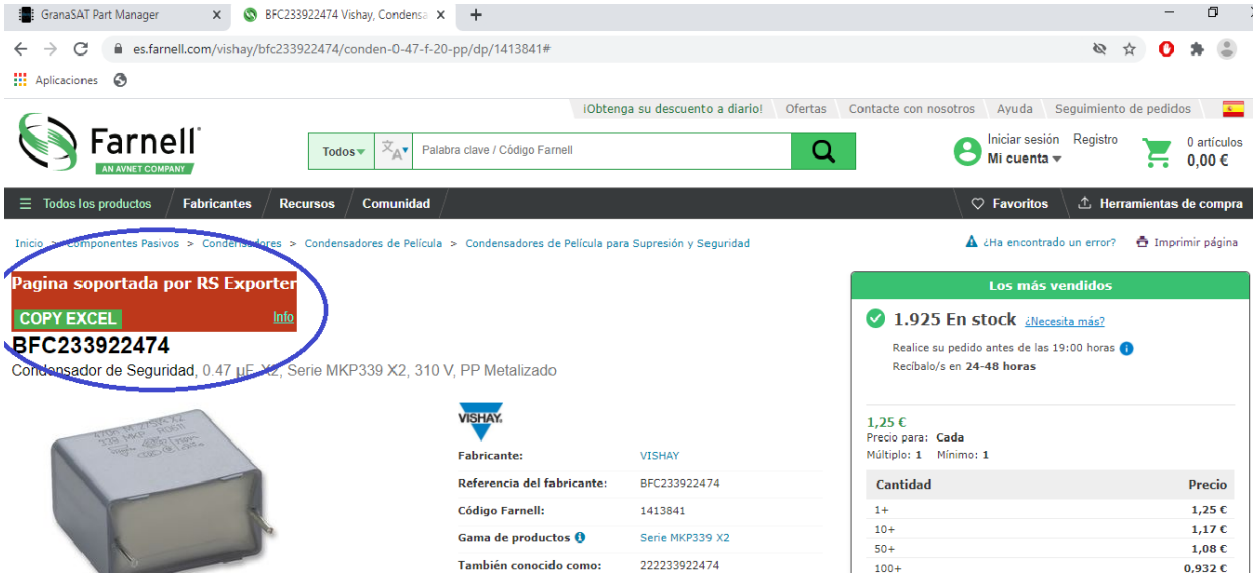


Figura 5.29 – *Panel generado por la extensión dentro de la página de un proveedor.*

La instalación de esta extensión como cualquier otra en Google Chrome es relativamente sencilla, en el Anexo 2 B se muestra los pasos a seguir en la instalación.

Vídeo que resume el funcionamiento de la extensión: `https://youtu.be/amoaEhtmx5k`.

5

Capítulo 6

Conclusiones y futuras mejoras

En este último capítulo, gracias a los conocimientos y experiencia adquirida en la realización del proyecto, se tratará proponer posibles mejoras del sistema, respecto a funcionalidad principalmente. Así mismo, que conclusiones se pueden sacar de este proyecto de forma breve, y la opinión subjetiva que este proyecto me puede suscitar desde mi punto de vista como estudiante.

6.1. Conclusión

Una vez finalizada la implementación del sistema y la comprobación de resultados, se puede concluir que se ha cumplido con todos los objetivos que se planteaban al inicio este proyecto. A partir de un sistema incompleto, se ha conseguido obtener otro completo que cubre la mayoría de carencias del anterior. El nuevo sistema es fácil de usar, útil a la hora de poder gestionar el inventario del laboratorio y bastante completo, que si bien siempre existe margen de mejora como se detallará más adelante, se ha cumplido con las exigencias del cliente.

A continuación, se va a tratar de exponer una breve conclusión o opinión de una forma subjetiva, desde mi punto de vista como estudiante.

Lo principal a valorar sobre este proyecto, es la gran cantidad de conocimientos que he adquirido. Se ha trabajado en un sistema bastante completo en el que he tenido que desarrollar en back-end y front-end, una gran oportunidad en la que he podido aplicar todo lo aprendido en estos últimos 4 años y para sobre todo aprender.

El hecho de emplear tecnologías totalmente actuales y de las más utilizadas ha sido un gran punto a favor.En lo personal estaba bastante interesado sobre JavaScript y todas sus aplicaciones (no solo en la parte del cliente), ya que pienso que es un lenguaje bastante utilizado e interesante. En mi caso conocía este lenguaje (aunque nada en comparación con después de realizar el proyecto), pero no había tenido la ocasión de poder aplicarlo utilizando exclusivamente framework de JavaScript, como son nodejs o reactjs.

Por otro lado, la oportunidad de haber podido trabajar en un sistema que va a ser usado realmente es de agradecer y ha sido un gran incentivo. Conocer que el sistema que estas desarrollando va a ser usado posteriormente hace querer realizar todo a la perfección y es algo más cercano a proyectos que se podrían realizar en el mundo laboral.

6.2. Posibles mejoras

Tras cumplir con todos los requisitos del cliente se da el proyecto por finalizado. Durante y después de la realización del proyecto, se han podido notar algunas posibles mejoras o ampliaciones dentro de la funcionalidad del sistema, que si bien no se han añadido ya que quedaban fuera del ámbito de este proyecto, no esta de más señalarlas

brevemente para posibles futuras versiones.

- **Diseño adaptable**, un gran punto de mejora de cualquier sitio web es hacerlo adaptable a dispositivos móviles. En este caso para gestionar el inventario del laboratorio sería algo bastante cómodo.
- **Mejorar la gestión de lugares de almacenamiento**, aunque no estaba dentro de los requisitos iniciales, los lugares de almacenamiento que se usan para poder ubicar el stock dentro del sistema, se pueden crear pero una vez creados no se permite editarlos o eliminarlos. No es algo que afecte a la funcionalidad pero no vendría mal añadir estas funciones.
- **Mejorar la gestión de documentos**, en concreto permitir eliminar documentos. Con la gestión de documentos pasa algo similar que con los lugares de almacenamiento.
- **Incluir la funcionalidad necesaria para añadir proyectos** Los proyectos de forma breve, son un conjunto de componentes. Tener constancia de estos proyectos en el sistema facilitará conocer previamente que componentes se van a necesitar para algún proyecto (siempre y cuando se halla realizado anteriormente). En el capítulo de diseño, se comentó que en el pasado se intentó implementar esta funcionalidad por lo que ya existen tablas en la base de datos si ninguna utilidad, pero se han mantenido en el sistema ya que en un futuro se podrían utilizarse para añadir proyectos.

Bibliografía

[1] Avnet. https://www.avnet.com/.

[2] Bootstrap 4. https://getbootstrap.com/docs/4.0/getting-started/introduction/.

[3] Digikey. https://www.digikey.es/.

[4] Farnell. https://es.farnell.com/.

[5] Lcsc. https://lcsc.com/.

[6] Librería js clipboardjs. https://clipboardjs.com/.

[7] Modelo mvc. https://www.patricksoftwareblog.com/tag/mvc/.

[8] Mouser. https://www.mouser.es/.

[9] Mysql. https://www.mysql.com/.

[10] Mysql workbench. https://www.mysql.com/products/workbench/.

[11] Módulo js npm, https. https://www.npmjs.com/package/https.

[12] Módulo js npm, qrcode-react. https://github.com/cssivision/qrcode-react.

[13] Módulo js npm, react-barcode-reader. https://www.npmjs.com/package/react-barcode-reader.

[14] Módulo js react-router-dom. https://reactrouter.com/web/guides/quick-start.

[15] Nodejs. https://nodejs.org/.

[16] Partkeepr. https://partkeepr.org/.

[17] Reactjs. https://es.reactjs.org/.

[18] Rs. https://es.rs-online.com/.

[19] Stockpile. http://www.thecanvus.com/homePage.

[20] Elm, W., Gualtieri, J., McKenna, B., Tittle, J., Peffer, J., Szymczak, S., and Grossman, J. Integrating cognitive systems engineering throughout the systems engineering process. Journal of Cognitive Engineering and Decision Making 2 (12 2008), 249–273.

[21] Stackoverflow. Ranking tecnologías más utilizadas en desarrollo web. https://insights.stackoverflow.com/survey/2020technology-web-frameworks-all-respondents.

Anexos

Apéndice A

Anexo I: Instalación de "PartManager"

Para poder utilizar la aplicación web se necesita:

- Tener **NodeJS** [15] instalado, (este incluye npm).
- **MySQL server**, versión recomendada 5.7.30.

Los pasos a seguir en la instalación son los siguientes:

- **Instalar dependencias.** Una vez descargado el directorio que contiene el "PartManager", normalmente va a ser necesario instalar las dependencias del proyecto, para ello nos situamos en la parte del servidor (*"/PartManager/granasat_pm_server"*) y ejecutamos: *"npm install"*. Del mismo modo nos situamos en la parte del cliente (*"/PartManager/granasat_pm_cliente"*) y ejecutamos ***"npm install"***.

- **Generar build del cliente**. El servidor utiliza una build del cliente, por lo cual para generar esta nos situamos en la parte del cliente y ejecutamos ***"npm run-script build"***.

- **Importar base de datos**. En la carpeta del proyecto existirá un archivo ".sql" que será necesario para importar la base de datos, se recuerda que sin la base de datos la aplicación web no funcionaría.

- **Ejecutar el servidor modo producción**. Situándonos en el lado del servidor, ejecutamos ***"nodejs app.js"***. Con esto quedaría la aplicación web accesible y operativa desde *"https://granasat2.ugr.es:9876"* (el dominio se puede modificar).

- **Ejecutar la aplicación en modo desarrollo**. Situándose en el lado del cliente, ejecutando ***"npm start"*** se arranca la aplicación web en modo desarrollo (*"https:granasat2.ugr.es:3000"*).

Cuestiones a tener en cuenta tras la instalación:

- **Cambiar el dominio o puertos donde se ejecuta la aplicación web**. Modificando las variables "HOST" y "WEBPORT" en el archivo *"./granasat_pm_server/app.js* (figura A.1, podemos hacer que se ejecute la aplicación en *"localhost:9876"* (por ejemplo).

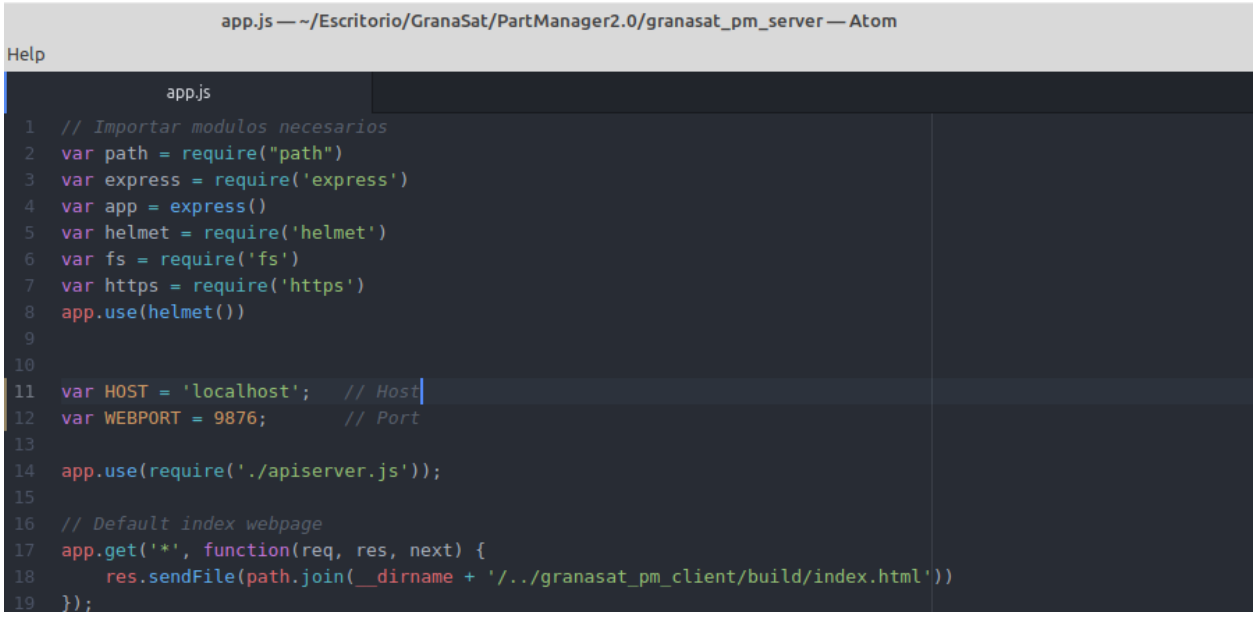


Figura A.1 – *"./granasat_pm_server/app.js*

- **Credenciales de la base de datos**. Para cada sistema normalmente se va a tener que modificar el archivo *"./granasat_pm_server/dbconnect.js"* con la información de la base de datos, como en la figura B.3.

Figura A.2 – *"./granasat_pm_server/dbconnect.js*

- **Directorio "Files"**. En caso de que el directorio *"./granasat_pm_server/files"* no exista o se encuentre vacío (aún existiendo categorías dentro de la base de datos) o la estructura de directorios sea como en el antiguo "PartManager" como se detalla en 4.1, será necesario ejecutar el script situado en el lado del servidor basta con ejecutar ***"nodejs script "***.

Apéndice B

Anexo II: Instalación de la extensión de Google Chrome

Obviamente para utilizar la extensión "Component Data Exporter" es necesario instarlo previamente. La instalación es sencilla y se puede realizar en los siguientes pasos:

- **Administrar extensiones**.

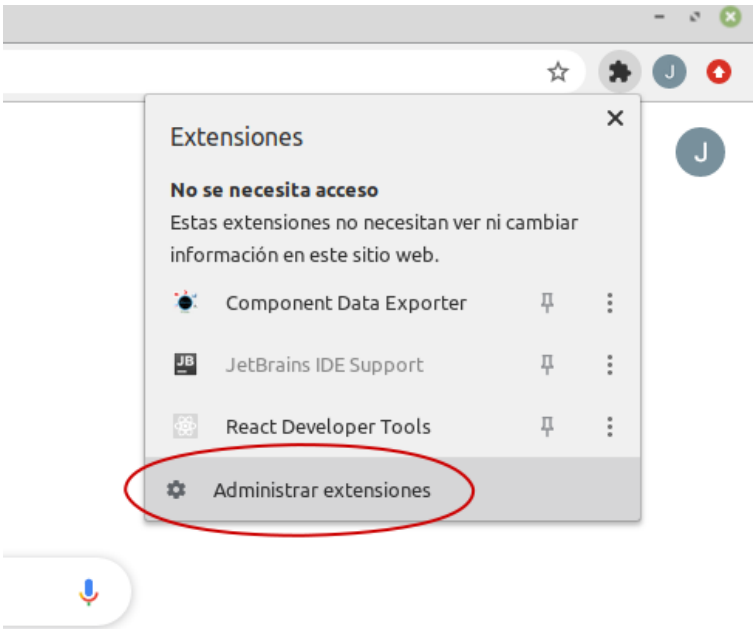


Figura B.1 – *Administrar extensiones en Google Chrome*

- **Habilitar el modo desarrollador** para poder cargar la extensión y **cargar la extensión** desde el fichero después de descomprimir.

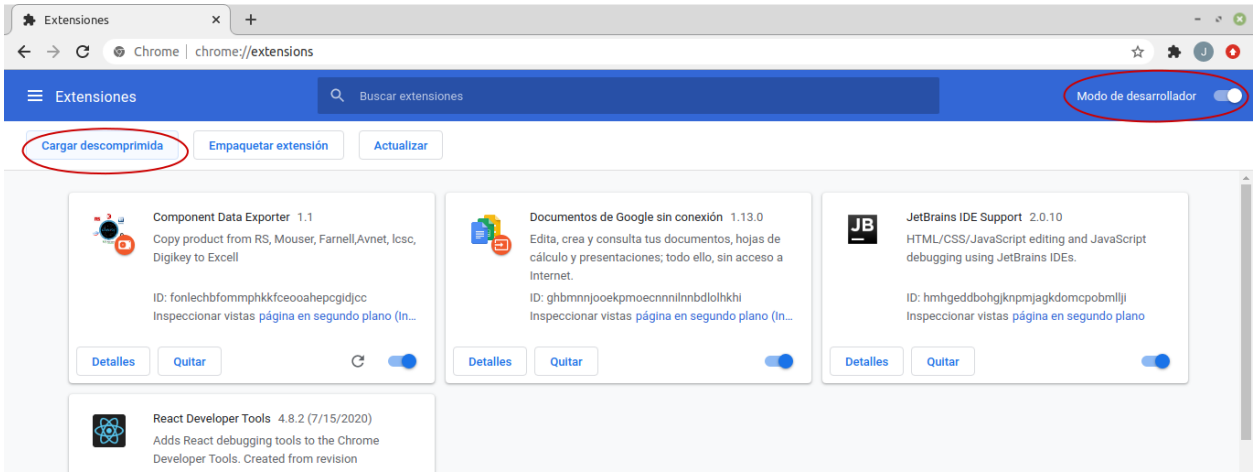


Figura B.2 – *Cargar fichero descomprimido a Google Chrome*

- **Activar ejecución en segundo plano**, y ¡listo para funcionar!

Figura B.3 – *Extensión instalada en Google Chrome*