DESARROLLO DE UN SISTEMA INTELIGENTE PARA SELECCIONAR Y RECOMENDAR ELEMENTOS PERSONALIZADOS

DEVELOPMENT OF AN INTELLIGENT SYSTEM TO SELECT AND RECOMMEND PERSONALIZED ELEMENTS

TRABAJO FIN DE GRADO
CURSO 2024-2025

AUTOR
ISABEL ZAMARRÓN MATEO

DIRECTOR
ANTONIO SARASA CABEZUELO

CALIFICACIÓN: 9

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE UN SISTEMA INTELIGENTE PARA SELECCIONAR Y RECOMENDAR ELEMENTOS PERSONALIZADOS

DEVELOPMENT OF AN INTELLIGENT SYSTEM TO SELECT AND RECOMMEND PERSONALIZED ELEMENTS

TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

AUTOR
ISABEL ZAMARRÓN MATEO

DIRECTOR
ANTONIO SARASA CABEZUELO

CONVOCATORIA: JUNIO 2025

GRADO EN INGENIERÍA DEL SOFTWARE
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

21 DE MAYO DE 2025

DEDICATORIA

A mi familia, amigos y a mí misma.

AGRADECIMIENTOS

Agradecer a mi tutor Antonio por su acompañamiento durante este proceso, por permitirme desarrollar una idea personal y por su confianza.

A mi familia y amigos, por acompañarme y apoyarme durante todo el camino.

RESUMEN

Desarrollo de un sistema inteligente para seleccionar y recomendar elementos personalizados

Este Trabajo de Fin de Grado presenta la creación de una aplicación web diseñada con el objetivo de digitalizar el armario de ropa de los usuarios. Ofrece una serie de funciones que permiten la gestión de la ropa mediante su categorización y etiquetado, así como la posible combinación de prendas para la creación de outfits.

Además de ser una herramienta que puede facilitar el día a día de los usuarios al permitirles elaborar sus outfits previamente, es una excelente forma de maximizar el uso de las prendas que posee el usuario pudiendo así evaluar su utilidad y evitar el consumo masivo de prendas innecesarias.

La aplicación también cuenta con una comunidad de usuarios que pueden publicar sus combinaciones de prendas, proporcionando una gran fuente de inspiración.

Palabras clave

Aplicación web, prenda, outfit, armario, publicaciones, vestimenta, combinación, interacción.

ABSTRACT

Development of an intelligent system for selecting and recommending custom items

This Final Degree Project presents the creation of a web application designed with the aim of digitizing the users' wardrobe. It offers a series of functions that allow the management of clothes through categorization and labeling, as well as the possible combination of garments for the creation of outfits.

In addition to being a tool that can facilitate the day-to-day life of users by allowing them to prepare their outfits beforehand, it is an excellent way to maximize the use of the garments that the user owns, thus being able to evaluate their usefulness and avoid the massive consumption of unnecessary garments.

The app also has a community of users who can post their garment combinations, providing a great font of inspiration.

Keywords

Web applications, clothing item, outfit, wardrobe, post, garment, matching, interaction.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

En este capítulo se detallan los motivos que han llevado a realizar este TFG, los objetivos que se quiere alcanzar y cómo se ha llevado a cabo.

1.1 Motivación

En la actualidad, la moda ha tomado un papel muy importante en nuestra sociedad, se trata de un fenómeno social, cultural y generacional que se transforma rápidamente y que para muchas personas se ha convertido en un medio de expresión.

Esto a su vez ha provocado hoy en día lo que se conoce como "fast fashion", un término que se refiere a un modelo de negocio centrado en la producción masiva, rápida y a bajo coste de prendas de vestir. Este fenómeno no solo afecta a la sociedad, sino también al medio ambiente. Según Greenpeace, "*la industria de la moda produce entre el 8% y el 10% de las emisiones mundiales de CO_2 (entre 4.000 y 5.000 millones de toneladas anuales)*". Y se considera que estos datos podrían aumentar en los próximos años debido al impacto de la digitalización, el cual acompañado del marketing online, el comercio electrónico y la influencia de las redes sociales están fomentando este consumo rápido e insostenible.

Por estos motivos con el desarrollo de esta aplicación se busca poder conocer cuál es el uso real de las prendas de nuestro armario, valorarlo y buscar sacar el mayor partido a éstas mediante la creación de múltiples combinaciones, pudiendo evitar la compra y acumulación de prendas innecesarias.

Así mismo, la presencia de una comunidad de usuarios en la aplicación, donde éstos pueden compartir sus outfits, permitiría inspirar a otros usuarios fomentando la posible reutilización de prendas que ya poseen y evitando la necesidad de comprar ropa nueva.

Por último, mediante el uso de esta aplicación se pretende facilitar al usuario la gestión de su vestuario en su día a día, pudiendo mantener un armario ordenado, planificar sus outfits con antelación y reduciendo el tiempo asociado a la elección diaria de vestimenta.

1.2 Objetivos

El objetivo principal de este proyecto es crear una herramienta que permita digitalizar los armarios de los usuarios.

Para alcanzar el objetivo principal, se definen los siguientes objetivos específicos:

- Análisis del mercado: se realizará un estudio para obtener un mayor conocimiento sobre el sector actual tanto a nivel moda como tecnología. Ésto junto al análisis de otras soluciones del mercado nos permitirá identificar unas funcionalidades que satisfagan las necesidades de los usuarios.
- Diseño de una interfaz intuitiva y accesible: se buscará una fácil gestión y visualización del armario virtual por parte de los usuarios.
- Gestión de outfits y prendas: se desarrollará un sistema que permita al usuario añadir y organizar cada prenda de forma individual y realizar combinaciones para crear outfits personalizados.
- Incorporación de una comunidad de usuarios: se desarrollará un sistema que permita a los usuarios interaccionar entre ellos dentro de la aplicación y que les permita compartir sus publicaciones.

1.3 Plan de trabajo

En esta sección se detalla el plan de trabajo seguido para alcanzar los objetivos descritos en el apartado anterior.

A continuación, se muestra la estimación realizada al inicio del proyecto, sobre la duración de cada fase de trabajo.

Nombre de tarea	Fecha de inicio	Fecha final
	2024/09/25	**2025/05/08**
Especificación de requisitos	2024/09/25	2024/09/30
Planificación	2024/10/01	2024/10/04
Desarrollo	**2024/10/07**	**2025/04/25**
Iteración 1	2024/10/07	2024/11/01
Hito 1	2024/11/04	2024/11/04
Iteración 2	2024/10/08	2024/12/02
Hito 2	2024/12/03	2024/12/03
Iteración 3	2024/12/04	2025/02/07
Hito 3	2025/02/10	2025/02/10
Iteración 4	2025/02/11	2025/04/24
Hito 4	2025/04/25	2025/04/25
Maquetación y revisión	2025/04/28	2025/05/08

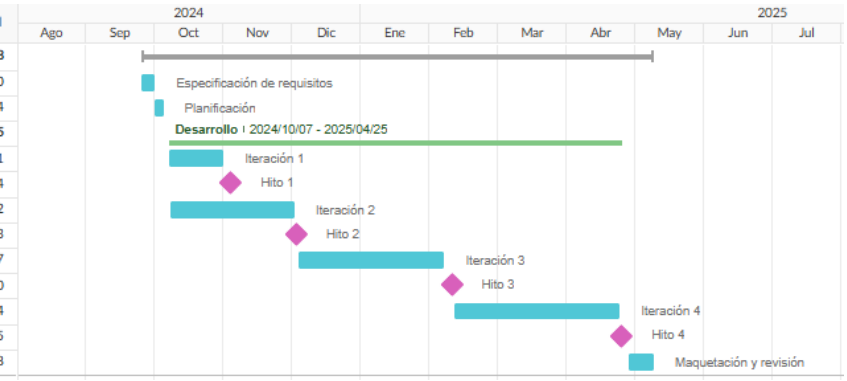


Figura 1- Diagrama de Gantt

1.3.1 Especificación de requisitos

Durante esta primera fase, se abordó la idea inicial de la que se partía y se consensuó con el tutor los casos de uso a realizar para lograr los objetivos marcados. Se llevaron a cabo varias correcciones hasta alcanzar una versión final de éstos.

1.3.2 Planificación

En esta segunda fase se procedió a organizar cómo se realizaría el desarrollo de los requisitos anteriormente fijados, en este caso se decidió dividirlo en 4 iteraciones.

Se establecieron 4 hitos, los cuales funcionaron como reunión de seguimiento, donde se revisaba el avance del proyecto.

1.3.3 Desarrollo

En esta fase, siguiendo la planificación establecida se llevaron a cabo 4 iteraciones.

1.3.3.1 1ª Iteración

En esta primera iteración, se sentaron las bases del proyecto, se creó una estructura de directorios, la base de datos y se realizaron las configuraciones correspondientes. A continuación, se llevó a cabo el desarrollo de las funciones básicas del módulo de gestión de usuarios y del administrador.

1.3.3.2 2ª Iteración

En esta segunda iteración, se abarcaron las funciones relativas al módulo de prendas, se implementó el código necesario para su correcta gestión.

1.3.3.3 3ª Iteración

En esta tercera iteración, se abarcaron las funciones relativas al módulo de outfits, para ello se empleó una lógica similar a la utilizada previamente en el módulo de prendas, estando además esté presente para determinadas funciones.

1.3.3.4 4ª Iteración

En esta última iteración, la implementación se centró en las funciones del módulo comunidad, donde se gestionan las publicaciones de outfits de los usuarios, junto a la creación de la lógica de interacción entre ellos.

También se realizó una función perteneciente al módulo de gestión de usuarios, "Ver estadísticas", pues ya teniendo desarrollado los módulos de prendas y outfits podía llevarse a cabo un análisis de sus parámetros.

1.3.4 Maquetación y revisión

En esta última fase, una vez finalizado el desarrollo del código de la aplicación, se procedió a su revisión y a la corrección de posibles fallos no detectados previamente.

Finalmente se elaboró la memoria del proyecto donde se recogería toda la documentación correspondiente a su diseño e implementación.

Chapter 1 - Introduction

This chapter details the reasons that have led to carry out this TFG, the objectives to be achieved and how it has been carried out.

1.1 Motivation

Nowadays, fashion has played an essential role in our society, it is a social, cultural and generational phenomenon that is rapidly transforming and that for many people has become a means of expression.

This in turn has today caused a phenomenon known as "fast fashion", a term that refers to a business model focused on the massive, fast and low-cost production of clothing. This phenomenon not only affects society, but also the environment. According to Greenpeace, *"the fashion industry produces between 8% and 10% of global CO_2 emissions (between 4,000 and 5,000 million tons per year)"*. And it is considered that these data could increase in the coming years due to the impact of digitalization, which together with online marketing, e-commerce and the influence of social networks are promoting this rapid and unsustainable consumption.

For these reasons, with the development of this application, the aim is to be able to know what the real use of garments in our wardrobe is, value it and seek to get the most out of them by creating multiple combinations, being able to avoid the purchase and accumulation of unnecessary garments.

Likewise, the presence of a community of users in the application, where they can share their outfits, would inspire other users, encouraging the possible reuse of garments they already own and avoiding the need to buy new clothes.

Finally, using this application, it is intended to make it easier for the user to manage their wardrobe in their day-to-day life, being able to keep a tidy wardrobe, plan their outfits in advance and reduce the time associated with the daily choice of clothing.

1.2 Goals

The main objective of this project is to create a tool that allows users to digitize their lockers.

To achieve the main objective, the following specific objectives are defined:

- Market analysis: a study will be carried out to obtain greater knowledge about the current sector both in terms of fashion and technology. This, together with the analysis of other solutions on the market, will allow us to identify functionalities that meet the needs of users.
- Design of an intuitive and accessible interface: easy management and visualization of the virtual cabinet by users will be looked for.
- Outfit and garment management: a system will be developed that allows the user to add and organize each garment individually and make combinations to create personalized outfits.
- Incorporation of a community of users: a system will be developed that allows users to interact with each other within the application and that allows them to share their publications.

1.3 Work plan

This section details the work plan followed to achieve the objectives described in the previous section.

Below is the estimate made at the beginning of the project, on the duration of each phase of work.

Nombre de tarea	Fecha de inicio	Fecha final
	2024/09/25	**2025/05/08**
Especificación de requisitos	2024/09/25	2024/09/30
Planificación	2024/10/01	2024/10/04
Desarrollo	**2024/10/07**	**2025/04/25**
Iteración 1	2024/10/07	2024/11/01
Hito 1	2024/11/04	2024/11/04
Iteración 2	2024/10/08	2024/12/02
Hito 2	2024/12/03	2024/12/03
Iteración 3	2024/12/04	2025/02/07
Hito 3	2025/02/10	2025/02/10
Iteración 4	2025/02/11	2025/04/24
Hito 4	2025/04/25	2025/04/25
Maquetación y revisión	2025/04/28	2025/05/08

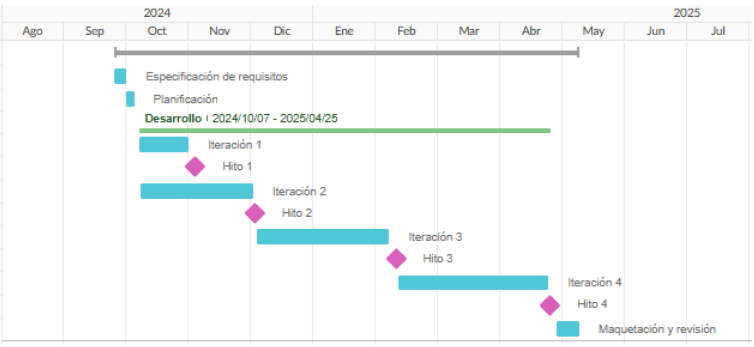


Figure 1- Gantt chart

1.3.1 Specification of requirements

During this first phase, the initial idea was discussed and the use cases to be carried out to achieve the objectives set were agreed with the tutor. Several corrections were made until a final version of these was reached.

1.3.2 Planning

In this second phase, we proceeded to organize how the development of the previously established requirements would be carried out, in this case it was decided to divide it into 4 iterations.

4 milestones were established, which functioned as a follow-up meeting, where the progress of the project was reviewed.

1.3.3 Development

In this phase, following the established planning, 4 iterations were carried out.

1.3.3.1 1st Iteration

In this first iteration, the foundations of the project were laid, a directory structure was created, the database was created and the corresponding configurations were made. Next, the development of the basic functions of the user management module and the administrator was carried out.

1.3.3.2 2nd Iteration

In this second iteration, the functions related to the garment module were covered, the necessary code for its correct management was implemented.

1.3.3.3 3rd Iteration

In this third iteration, the functions related to the outfit module were covered, for this a logic similar to that previously used in the garment module was used, being also present for certain functions.

1.3.3.4 4th Iteration

In this last iteration, the implementation focused on the functions of the community module, where users' outfit posts are managed, along with the creation of the interaction logic between them.

A function belonging to the user management module, "View statistics", was also carried out, since already having developed the clothing and outfit modules, an analysis of their parameters could be carried out.

1.3.4 Layout and revision

In this last phase, once the development of the application code was completed, it was reviewed and possible previously undetected bugs were corrected.

Finally, the project report was prepared where all the documentation corresponding to its design and implementation would be collected.

Capítulo 2 - Estado de la cuestión

En este capítulo se describen algunas aplicaciones actualmente en el mercado con funcionalidades similares a este proyecto, con el objetivo de identificar posibles debilidades o limitaciones de cara a satisfacer las necesidades de los usuarios.

A continuación, se detallan varias aplicaciones con una breve descripción de sus características.

2.1 GetWardrobe

GetWardrobe es una aplicación disponible en dispositivos móviles y en web. Posee una versión gratuita que permite almacenar hasta 100 prendas y una versión premium con suscripción mensual con almacenaje ilimitado y acceso a funciones avanzadas.

Entre sus principales funciones destaca por la creación de listas de viaje, la integración de datos meteorológicos para sugerir combinaciones de acuerdo con el clima y generación de estadísticas de uso permitiendo al usuario analizar su estilo.

2.2 Acloset

Acloset es una aplicación para dispositivos móviles, cuya versión gratuita permite almacenar hasta 100 prendas o por el contrario requiere de suscripción.

Se trata de un armario virtual impulsado por IA, que permite analizar los detalles de las prendas del usuario facilitando su proceso de creación, ofrece recomendaciones de outfits adaptadas a distintos factores como el clima o actividades diarias, y también analiza y registra hábitos de compra.

2.3 Whering: Tu armario virtual

Whering: Tu armario virtual es una aplicación para dispositivos móviles. Ofrece una versión gratuita limitada a funciones básicas, siendo necesaria una suscripción de pago para hacer uso de funciones más avanzadas.

Entre sus principales funciones destaca por la interacción social que permite el intercambio y valoración de outfits entre usuarios y por la creación de "moodboards" cuya función es estimular la creatividad del usuario y funcionar como una herramienta inspiracional.

Capítulo 3 - Tecnologías empleadas

En este capítulo se detallan las principales tecnologías que se han empleado para el desarrollo del proyecto.

3.1 Frontend

3.1.1 HTML

HTML (HyperText Markup Language) es un lenguaje estándar de marcado que es utilizado para estructurar y desplegar una página web y sus contenidos. Define la base, el contenido e información de una página web mediante el uso de elementos y etiquetas.

3.1.2 CSS

CSS (Cascading Style Sheets) es un lenguaje de hojas de estilo utilizado para describir la presentación de un documento escrito en HTML. Es el encargado de definir y controlar el diseño y la apariencia de una página facilitando su personalización y consistencia.

Este lenguaje se ha empleado con el objetivo de personalizar la apariencia de la interfaz y que así se alinee con la estética definida para este proyecto.

3.1.3 Bootstrap

Bootstrap es un framework de código abierto, diseñado para facilitar el proceso de desarrollo de sitios web responsivos. Combina una colección de herramientas CSS y JavaScript, proporcionando una serie de componentes predefinidos, como botones, formularios, menús de navegación que permiten a los desarrolladores construir sitios web de manera rápida y eficiente.

En este proyecto se ha empleado este framework con el objetivo de desarrollar fácilmente una interfaz con un diseño responsive gracias al uso de sus distintos componentes predefinidos y su sistema de grillas.

3.1.4 Tailwind CSS

Tailwind es un framework de código abierto que permite diseñar interfaces de usuario de forma rápida y eficiente gracias a sus clases predefinidas directamente en el HTML. A diferencia de Bootstrap, este framework ofrece mayor flexibilidad y personalización, lo que permite construir diseños que se adaptan a las necesidades de cada proyecto.

En este proyecto se ha empleado específicamente para el diseño y personalización de la barra de navegación lateral.

3.1.5 JavaScript

JavaScript es un lenguaje de programación que permite implementar funciones complejas aportando interactividad y dinamismo en páginas web. Se ejecuta en el navegador del usuario y permite manipular el DOM, manejar eventos, y realizar solicitudes asíncronas.

En este proyecto este lenguaje se ha empleado junto con HTML para definir la estructura de la vista y tal y como se menciona en siguientes apartados. También se ha utilizado en el desarrollo del backend mediante node.js.

3.2 Backend y servidor

3.2.1 Node.js

Node.js es un entorno de ejecución multiplataforma de Javascript diseñado con una arquitectura basada en eventos y un modelo asíncrono.

Cuenta con npm (Node Package Manager), un gestor de paquetes que permite instalar librerías de terceros y herramientas globales, además de permitir la gestión de dependencias.

Su uso viene motivado por la facilidad que ofrece a hora de crear aplicaciones web escalables y eficientes, gracias al uso de código JavaScript tanto en el lado del cliente como en del servidor.

Se ha optado por esta herramienta dado su conocimiento previo al inicio del proyecto, además de por la extensa documentación del entorno y por la calidad de aplicaciones que permite desarrollar.

3.2.2 Express.js

Express.js es un framework utilizado para crear aplicaciones web. Está basado en Node.js y proporciona un conjunto de características como son el manejo de rutas, uso de middlewares, y gestión de solicitudes HTTP. Estas características permiten a los desarrolladores construir aplicaciones web y API de manera eficiente y escalable.

En este proyecto se ha empleado para complementar a Node.js facilitando significativamente la estructura y rendimiento de la aplicación.

3.2.3 Python

Python es un lenguaje de programación de alto nivel. Ofrece estructuras de datos eficientes y un sistema de programación orientado a objetos. Se caracteriza por su tipado dinámico y es ampliamente usado en el desarrollo rápido de aplicaciones y scripting en múltiples plataformas.

Este lenguaje es utilizado en la librería externa Rembg, una herramienta que permite eliminar el fondo de las imágenes. Ésta se encuentra implementada en Python debido a que este lenguaje cuenta con un ecosistema muy desarrollado en tareas de procesamiento de imágenes e inteligencia artificial, lo que hace que la eliminación de fondos se haga con mucha precisión.

3.3 Librerías externas

3.3.1 Rembg

Rembg es una librería de Python de código abierto que, mediante modelos de inteligencia artificial pre-entrenados permite eliminar de forma automática los fondos de las imágenes.

Esta librería ha sido elegida para realizar el procesamiento de las imágenes que el usuario adjunta cuando desea añadir una prenda a su armario pues la mayoría de

las IA de eliminación de fondo requieren del pago de una suscripción; sin embargo, posee la limitación de no aceptar archivos de gran peso.

3.3.2 Html2canvas.js

Html2canvas es una librería de Javascript que permite renderizar la página actual en un elemento <canvas>, leyendo el DOM y los diferentes estilos aplicados a los elementos. Se caracteriza por la flexibilidad de configuración pues permite controlar la escala, ancho, alto y filtros CSS. Además, realiza la conversión de HTML a imagen en el navegador del usuario sin necesitar el envío de datos al servidor.

En este proyecto se ha escogido esta librería debido a la facilidad de integración con el frontend que proporciona, por la calidad que ofrecen las capturas de pantalla que realiza y porque además no carga en exceso el servidor.

3.3.3 Interact.js

Interact.js es una librería de código abierto de Javascript que facilita la creación de interfaces interactivas y el manejo de eventos de arrastre de manera eficiente. Esta librería soporta tres tipos de acciones, permite mover elementos, implementar aplicaciones de arrastrar y soltar y redimensionar elementos.

En este proyecto se ha escogido esta librería debido a la facilidad de integración que proporciona su código predefinido y a la amplia gama de interacciones que ofrece. Es una solución que resuelve adecuadamente la problemática surgida para que el usuario pueda mover las prendas a su gusto en un área determinada.

3.4 GitHub

GitHub es una plataforma basada en la nube que aloja un sistema de control de versiones llamado Git. Esta herramienta nos permite almacenar, compartir y trabajar en proyectos compartidos que requieren un seguimiento.

En el caso de este proyecto se ha dedicado principalmente a seguir los cambios en el código de una forma más eficiente, pues gracias a su historial de versiones permitía tener un mayor control sobre los cambios que se realizaban.

Capítulo 4 - Arquitectura de la aplicación y modelo de datos

En este capítulo se detalla la arquitectura y el modelo de datos empleados en este proyecto.

4.1 Arquitectura de la aplicación

Para estructurar este proyecto se ha empleado una arquitectura en capas con el objetivo de hacer a los componente independientes, correspondiéndole a cada uno una responsabilidad especifica dentro de la aplicación web. La comunicación entre capas se caracteriza por ser únicamente con la capa inmediatamente inferior.

Esta arquitectura se ve reforzada por el patrón cliente-servidor, donde el cliente realiza solicitudes HTTP que posteriormente son atendidas por un servidor estructurado en capas lógicas, facilitando el mantenimiento, la escalabilidad y la modularidad de la aplicación.

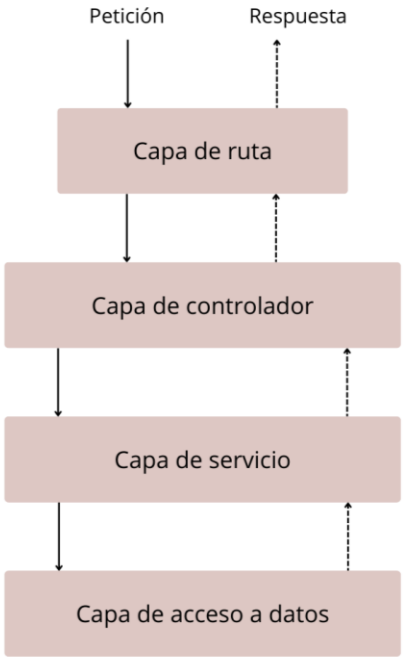


Figura 2- Diagrama de arquitectura de la aplicación

En concreto, se ha optado por la utilización de cuatro capas, mediante las cuales se quería lograr cumplir los criterios de separación de responsabilidades, escalabilidad y obtener una organización clara del código.

A continuación se describen las capas que conforman la arquitectura:

- **Capa de rutas**: esta capa es la encargada de recibir las solicitudes HTTP del cliente y redirigirlas al método correspondiente que está presente en la capa del controlador, esperar su respuesta y servir el recurso solicitado.
- **Capa de controladores:** esta capa recibe las solicitudes de la capa de rutas y delega la ejecución de la lógica de negocio a la capa de servicios. Una vez procesada la información la respuesta es devuelta al cliente.
- **Capa de servicios:** esta capa es la encargada de la lógica de negocio, transforma y procesa los datos que proceden de la capa de controlador para adaptarlos y que puedan ser usados en la capa de acceso a datos.
- **Capa de acceso a datos:** esta capa es la encargada de comunicarse con la base de datos. Realiza las instrucciones en SQL para consultar, insertar y actualizar la información pasada por parámetro desde la capa de servicio.

4.2 Estructura de la base de datos

Para lograr un almacenamiento eficiente de la información en este proyecto se optó por el uso de una base de datos relacional. Esta decisión se debe a la necesidad de organizar de forma estructurada la información de los elementos que componen el armario virtual y las combinaciones realizadas por el usuario con él. Además, garantiza la integridad de los datos, asegurando que éstos son consistentes y fiables y mediante la definición de restricciones evitaría su posible duplicación e incoherencia.

El uso de XAMPP proporcionó un entorno local donde desarrollar la aplicación web. Éste, al incluir phpMyAdmin, una herramienta de administración de bases de datos, facilitó tareas como son la creación de la base de datos y sus correspondientes elementos, gestión de datos y ejecución de consultas SQL.

Para crear la estructura de la base de datos se comenzó con el diseño del modelo entidad relación, se establecieron las correspondientes entidades

acompañadas de sus atributos y relaciones. A continuación, tomando como referencia el diagrama ER, se definieron las tablas con sus correspondientes claves primarias y foráneas y restricciones de integridad.

4.2.1 Diagrama de Entidad Relación

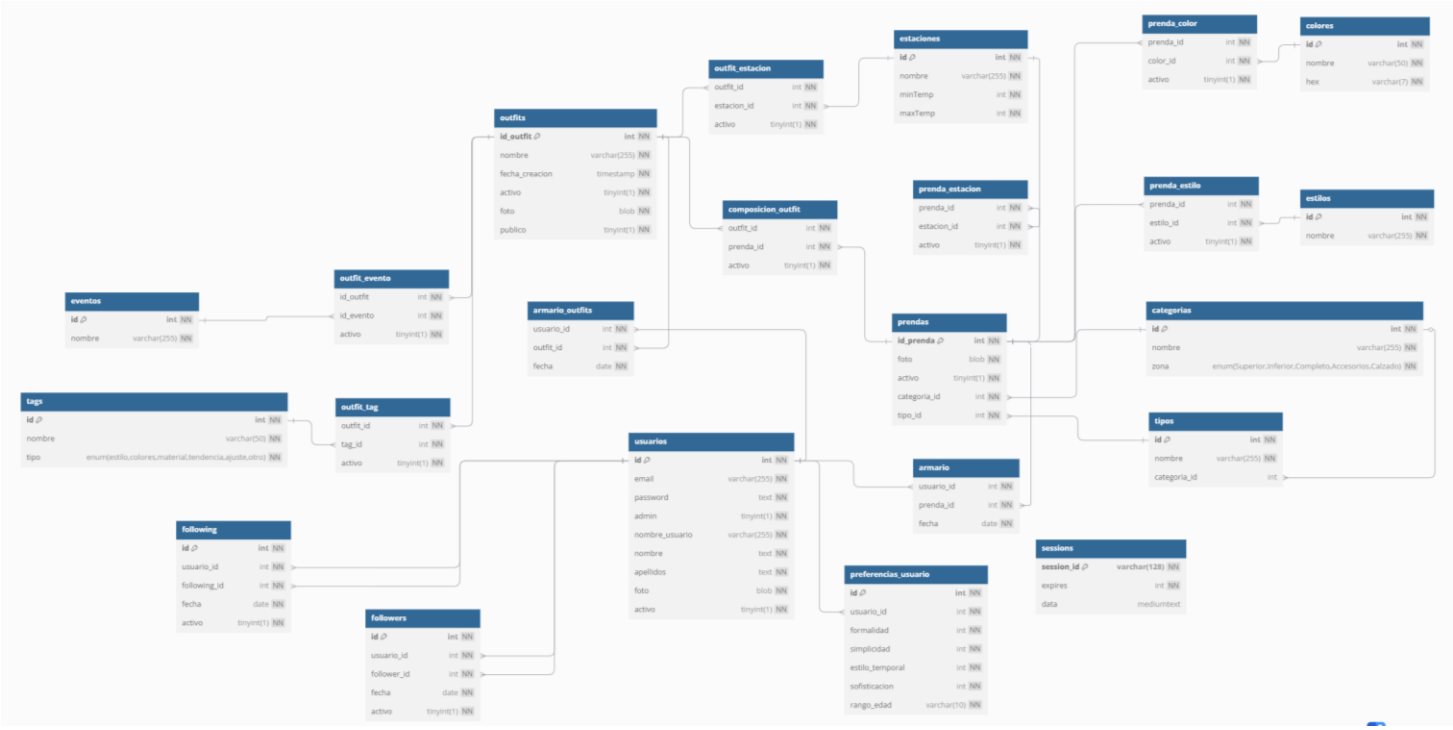


Figura 3- Diagrama Entidad Relación

4.2.2 Tablas

A continuación, se detallan las tablas de la base de datos de las principales entidades del proyecto.

4.2.2.1 Tabla de usuarios

En esta tabla se almacena la información de los usuarios que es imprescindible para la gestión de éstos y para el proceso de autenticación en la aplicación.

Entre todos sus atributos, cabe destacar:

- El atributo *id* es el identificador único asignado a cada usuario y clave primaria (PK).

- El atributo *admin* permite conocer cuál es el rol del usuario, cuando tiene valor true se corresponde a un usuario administrador, mientras que cuando tiene el valor false corresponde con un usuario común.
- El atributo *activo* permite conocer el estado de la cuenta del usuario, tomando el valor false en el caso de que un usuario se hubiese dado de baja.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	email	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
3	password	text	utf8mb4_general_ci		No	*Ninguna*		
4	admin	tinyint(1)			No	*Ninguna*		
5	nombre_usuario	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
6	nombre	text	utf8mb4_general_ci		No	*Ninguna*		
7	apellidos	text	utf8mb4_general_ci		No	*Ninguna*		
8	foto	blob			No	*Ninguna*		
9	activo	tinyint(1)			No	*Ninguna*		

Figura 4- Tabla de usuarios

4.2.2.2 Tabla de sessions

En esta tabla se almacena información sobre las sesiones activas de los usuarios de la aplicación.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	session_id	varchar(128)	utf8mb4_bin		No	*Ninguna*		
2	expires	int(11)		UNSIGNED	No	*Ninguna*		
3	data	mediumtext	utf8mb4_bin		Sí	*NULL*		

Figura 5- Tabla de sessions

4.2.2.3 Tabla de preferencias de usuario

En esta tabla se almacenan las preferencias de estilo del usuario. Esta tabla complementa a la tabla de usuarios; para ello, se emplea una clave foránea (FK) en *usuario_id* que permite establecer una relación con dicha tabla.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	usuario_id	int(11)			No	*Ninguna*		
3	formalidad	int(11)			No	*Ninguna*		
4	simplicidad	int(11)			No	*Ninguna*		
5	estilo_temporal	int(11)			No	*Ninguna*		
6	sofisticacion	int(11)			No	*Ninguna*		
7	rango_edad	varchar(10)	utf8mb4_general_ci		No	*Ninguna*		

Figura 6- Tabla de preferencias de usuario

4.2.2.4 Tabla de prendas

En esta tabla se almacena la información de las prendas que añaden los usuarios. Entre todos sus atributos, cabe destacar:

- El atributo *id_prenda* es el identificador único asignado a cada prenda y clave primaria (PK).
- El atributo *activo* permite conocer el estado de la prenda, tomando el valor false en el caso de que una prenda hubiese sido eliminada.

Por último, se emplean dos claves foráneas (FK) en *categoría_id* y *tipo_id* que permiten establecer una relación con las tablas Categorias y Tipos respectivamente.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_prenda	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	foto	blob			No	*Ninguna*		
3	activo	tinyint(1)			No	*Ninguna*		
4	categoria_id	int(11)			No	*Ninguna*		
5	tipo_id	int(11)			No	*Ninguna*		

Figura 7- Tabla de prendas

4.2.2.5 Tablas de parámetros de prendas

En estas tablas se almacenan los parámetros de una prenda. La estructura se repite para los parámetros categorías, tipos, estaciones, estilos y colores (cada uno cuenta con su propia tabla).

Cuentan con el atributo *id* que sirve de identificador del parámetro en cuestión y es la clave primaria (PK), junto con el atributo nombre.

En el caso de la tabla de colores, tiene la peculiaridad de tener otro atributo con el código hexadecimal RGB de su color correspondiente.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(10)			No	*Ninguna*		AUTO_INCREMENT
2	nombre	varchar(50)	utf8mb4_general_ci		No	*Ninguna*		
3	hex	varchar(7)	utf8mb4_general_ci		No	*Ninguna*		

Figura 8- Ejemplo de tabla de parámetro de prenda

4.2.2.6 Tablas de relación de parámetros con prenda (categoría, tipo, estilo, color y estación)

En estas tablas se almacenan las relaciones entre los parámetros y una prenda. La estructura se repite para los parámetros categorías, tipos, estaciones, estilos y colores. Se componen de un atributo *activo*, el cual facilita la gestión de los parámetros que caracterizan la prenda.

La relación con la tabla de cada parámetro se establece mediante una clave foránea (FK) en el *parámetro_id*. Además, *prenda_id* y *parámetro_id* se establecen como claves primarias (PK), lo que asegura que la combinación es única y no se producen duplicados.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	prenda_id	int(11)			No	*Ninguna*		
2	color_id	int(11)			No	*Ninguna*		
3	activo	tinyint(1)			No	*Ninguna*		

Figura 9- Ejemplo de tabla de relación de parámetro con prenda

4.2.2.7 Tabla de armario (prendas)

En esta tabla se almacenan las referencias correspondientes a las prendas que posee un usuario y cuando fueron añadidas.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	usuario_id	int(11)			No	*Ninguna*		
2	prenda_id	int(11)			No	*Ninguna*		
3	fecha	date			No	*Ninguna*		

Figura 10- Tabla de armario prendas

4.2.2.8 Tabla de outfits

En esta tabla se almacena la información de los outfits que añaden los usuarios. Entre todos sus atributos, cabe destacar:

- El atributo *id_outfit* es el identificador único asignado a cada outfit y clave primaria (PK).
- El atributo *activo* permite conocer el estado del outfit, tomando el valor false en el caso de que un outfit hubiese sido eliminado.
- El atributo *público* permite conocer si un outfit es visible únicamente para el usuario creador o para toda la comunidad.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_outfit	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	nombre	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		
3	fecha_creacion	timestamp			No	current_timestamp()		
4	activo	tinyint(1)			No	*Ninguna*		
5	foto	blob			No	*Ninguna*		
6	publico	tinyint(1)			No	*Ninguna*		

Figura 11- Tabla de outfits

4.2.2.9 Tablas de parámetros de outfits

En estas tablas se almacenan los parámetros de un outfit. La estructura se repite para los parámetros eventos, tags y estaciones. (cada uno cuenta con su propia tabla).

Cuentan con el atributo id que sirve de identificador del parámetro en cuestión y es la clave primaria (PK), junto con el atributo nombre.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	nombre	varchar(255)	utf8mb4_general_ci		No	*Ninguna*		

Figura 12- Ejemplo de tabla de parámetro de outfit

4.2.2.10 Tablas de relación de parámetros con outfits (eventos, estaciones y tags)

En estas tablas se almacenan las relaciones entre los parámetros y un outfit. La estructura se repite para los parámetros eventos, tags y estaciones. Se componen de un atributo *activo*, el cual facilita la gestión de los parámetros que caracterizan el outfit.

La relación con la tabla de cada parámetro se establece mediante una clave foránea (FK) en el *parámetro_id*. Además, outfit_*id* y *parámetro_id* se establecen como claves primarias (PK), lo que asegura que la combinación es única y no se producen duplicados.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id_outfit	int(11)			No	*Ninguna*		
2	id_evento	int(11)			No	*Ninguna*		
3	activo	tinyint(1)			No	*Ninguna*		

Figura 13- Ejemplo de tabla de relación de parámetro con outfit

4.2.2.11 Tabla de armario (outfits)

Tabla donde se almacenan las referencias correspondientes a los outfits que posee un usuario y cuando fueron añadidos.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	usuario_id	int(11)			No	*Ninguna*		
2	outfit_id	int(11)			No	*Ninguna*		
3	fecha	date			No	*Ninguna*		

Figura 14- Tabla de armario de outfits

4.2.2.12 Tabla de followers

Tabla donde se almacena información sobre los usuarios que siguen a un usuario. Siendo *usuario_id* el usuario a quien se sigue y *follower_id* el usuario que lleva a cabo la acción. Podemos conocer el estado de la relación con el atributo *activo*.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	usuario_id	int(11)			No	*Ninguna*		
3	follower_id	int(11)			No	*Ninguna*		
4	fecha	date			No	*Ninguna*		
5	activo	tinyint(1)			No	*Ninguna*		

Figura 15- Tabla de followers

4.2.2.13 Tabla de following

Tabla donde se almacena información sobre los usuarios que comienzan a seguir a otro usuario. Siendo *usuario_id* el usuario que lleva a cabo la acción y *following_id* el usuario seguido. Podemos conocer el estado de la relación con el atributo *activo*.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra
1	id	int(11)			No	*Ninguna*		AUTO_INCREMENT
2	usuario_id	int(11)			No	*Ninguna*		
3	following_id	int(11)			No	*Ninguna*		
4	fecha	date			No	*Ninguna*		
5	activo	tinyint(1)			No	*Ninguna*		

Figura 16- Tabla de following

Capítulo 5 - Implementación y diseño

En este capítulo se presenta una descripción detallada del diseño utilizado en la aplicación, así como de su implementación.

5.1 Diseño de la aplicación

5.1.1 Interfaz de usuario

La interfaz de usuario planteada tiene el objetivo de ofrecer una experiencia de usuario de calidad, para ello se ha tratado de facilitar en la medida de lo posible los desplazamientos entre menús mediante una navegación intuitiva, el uso de modales y un diseño responsive que se adapta a multitud de dispositivos.

En cuanto a la arquitectura de la navegación, el diseño sigue la siguiente estructura común:

- Barra de navegación (Navbar): se ubica en la parte superior de la interfaz. Ofrece acceso al menú principal, acceso rápido a la función de cerrar sesión y permite activar la barra lateral.
- Barra lateral (Sidebar): se ubica en el lateral izquierdo de la interfaz. Se hace visible cuando se activa desde la navbar, y dentro de ésta encontramos acceso directo a diversas funcionalidades de la aplicación.
- Cuerpo principal: es el área donde los usuarios pueden visualizar e interactuar con el contenido fruto de las diversas funcionalidades que posee la aplicación.
- Footer: se ubica en la parte inferior de la interfaz. Contiene información complementaria.

El estilo visual que se ha querido seguir en el proyecto sigue una línea limpia y moderna, caracterizada por elementos simples que evitan sobrecargar visualmente la aplicación.

La paleta de colores escogida se caracteriza por tener tonos suaves, que buscan transmitir una sensación agradable y elegante.

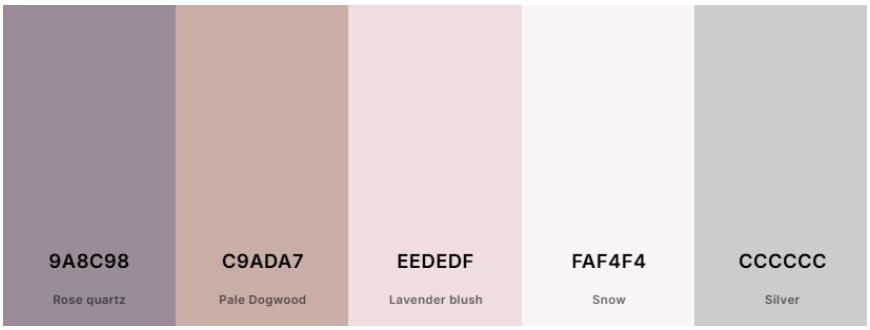


Figura 17- Paleta de colores

Para llevar a cabo este diseño, se han empleado hojas de CSS para establecer el estilo de forma general en la aplicación.

Se utilizó Tailwind específicamente para el diseño de la barra lateral (sidebar), ya que este framework permitía una gran personalización del componente y presentaba una mayor facilidad en cuanto a la gestión de la animación de apertura y cierre de la sidebar, mientras que con Bootstrap esta animación requería de más funciones.

Y por último, se empleó Bootstrap para la creación de la estructura responsive y uso de los componentes predefinidos como por ejemplo la navbar, footer e iconos.

Cabe destacar el abundante uso de modales de Bootstrap debido a la mejora de la interactividad y fluidez que proporcionan. En este proyecto los podemos encontrar en la presentación de formularios, confirmación de acciones y en la visualización de información.

5.1.2 Organización del proyecto y puesta en marcha

La distribución de carpetas creada respeta la arquitectura de aplicación definida, donde se divide cada componente en los distintos módulos descritos.

La estructura de directorios obtenida se muestra en la Figura 18.

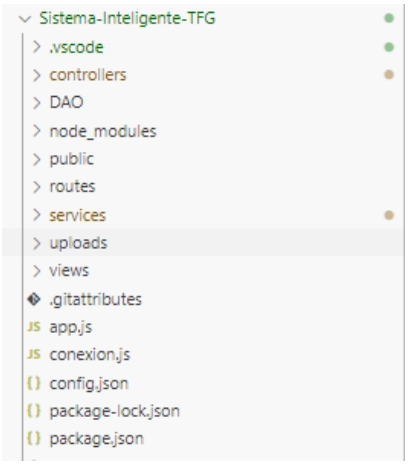


Figura 18- Estructura de directorios del proyecto

Para la puesta en marcha del servidor, en el archivo *app.js,* se importaron los módulos y dependencias necesarias, se configuraron los correspondientes routers que manejarían las rutas de la aplicación, se inicializó la aplicación mediante la creación de una instancia de Express, se configuraron el motor de plantillas y los directorios a archivos estáticos y, por último, se definió el puerto para el servidor con el cual se iniciará.

```
//importacion modulos
const express = require('express');
const path = require('path');
const bodyParser = require("body-parser");
const session = require("express-session");
const multer = require("multer");
const upload = multer({ dest: 'uploads/' });
const MySQLStore = require('express-mysql-session')(session);
const fileUpload = require('express-fileupload');
//routers
const userRouter = require("./routes/usuarios");
const authRouter = require("./routes/autenticacion");
const prendasRouter = require("./routes/prendas");
const outfitsRouter= require ("./routes/outfits");
const comunidadRouter=require("./routes/comunidad");
const app = express();

//configuracion vistas y directorios de archivos estaticos
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use('/public', express.static(path.join(__dirname, 'public')));
app.use(express.static(path.join(__dirname, 'public')));
app.use('/uploads', express.static(path.join(__dirname, 'uploads')));

//config servidor
const PORT = 3000;
app.listen(PORT, () => {
  console.log(`Server running on http://localhost:${PORT}/auth`);
});
```

Figura 19- Código de puesta en marcha del servidor

Se estableció la conexión a la base de datos mediante el código desarrollado en *conexión.js*, complementado con el archivo *config.json* que contiene la información correspondiente a la configuración de la base de datos.

```
"use strict";
const mysql = require("mysql");
var fs = require('fs');
var configPath = './config.json';
var parsed = JSON.parse(fs.readFileSync(configPath, 'UTF-8'));

const pool = mysql.createPool({
    host : parsed.host,
    user : parsed.user,
    password : parsed.password,
    database : parsed.database,
    port: parsed.port
});

function getConnection(done) {

    pool.getConnection(function (err, connection) {
      if (err) {
        console.log("Error de conexión a la base de datos.");
      } else {
        done(connection);
      }
    });

}
  module.exports = getConnection;
```

Figura 20- Código de configuración de la base de datos

5.2 Implementación de la aplicación

La implementación de las funcionalidades de la aplicación se ha basado en el desarrollo individual de los distintos módulos en los que se divide la aplicación web.

A continuación, se describe la implementación seguida a nivel general en algunos componentes seguido de las principales funcionalidades de cada módulo.

5.2.1 Implementación funciones de la capa controlador

Las funciones desarrolladas en el controlador siguen un patrón común que permite manejar las solicitudes y las respuestas HTTP. La función se exporta para que pueda ser utilizada en otros módulos de la aplicación.

Los parámetros necesarios se extraen de *req (request),* objeto que contiene la información de la petición HTTP y a continuación, se hace una llamada al servicio con un los correspondientes argumentos.

También existen funciones que dada su simplicidad no requieren de una función de servicio.

La función de servicio realizará la lógica de negocio y obtendrá una respuesta que será manejada en función de si tiene lugar correctamente o se produce un error.

Para el manejo de errores se ha implementado una función *handleError*() basada en la documentación de Express.js. En caso de éxito en la función encontramos dos tipos de respuestas recurrentes, aquellas donde se renderiza una vista y aquellas que devuelven información.

```
exports.showPrenda = (req, res) => {
    const prendaId = req.params.id;
    const usuarioId = req.user.id;

    prendaService.showPrenda({ prendaId, usuarioId },(err,prenda)=>
        if (err) {
            return handleError(res, err, 500);
        } else if (prenda) {
            res.status(200).json(prenda);
        }
        else {
            return handleError(res, 'Prenda no encontrado.', 404);
        }
    });
};
```

```
exports.listPrendas = (req, res) => {

        const userId = req.user.id;
        const categoria = req.query.categoria;

        prendaService.list(userId, (err, data) => {
            if (err) {
                return handleError(res, err, 500);
            }

            res.render('armarioPrendas', {
                prendas: data.prendas,
                categorias: data.categorias,
                tipos: data.tipos,
                u: req.user,
                categoria: categoria
            });
        });

};
```

Figura 21- Ejemplo funciones controlador

5.2.2 Implementación funciones de la capa servicio

Las funciones desarrolladas en el servicio siguen un patrón común. Constituyen la capa de lógica del negocio.

En las funciones implementadas lo primero que observamos es el parámetro *data* con la información necesaria recibida del *controller* para realizar las operaciones, en segundo lugar, *callback*, la función de devolución de la respuesta.

El servicio se encargará de llamar a la función del DAO correspondiente (en el caso de la implementación de este proyecto hacemos referencia a la capa de acceso a datos mediante este nombre) y del manejo de errores.

```
showPrenda:(data, callback) => {

    daoPrenda.getPrendaById(data.prendaId,data.usuarioId, (error, prenda) => {
        if (error) {
            return callback('Error al obtener prenda: ' + error, null); //error
        }
        else if (!prenda) {             //no existe la prenda
            return callback(null, null);
        }
       else{
        return callback(null, prenda);//devolver info prenda
       }
    });
},
```

Figura 22- Ejemplo de función de servicio

5.2.3 Implementación funciones de la capa de acceso a datos

Las funciones desarrolladas en la capa de acceso a datos siguen un patrón común con el cual se logra interactuar con la base de datos.

Esta función puede recibir o no argumentos de entrada para llevar a cabo las consultas SQL. El argumento *callback* se encargará de manejar las respuestas obtenidas. Seguidamente, encontramos la variable *query*, donde se construye la consulta a ejecutar en la base de datos y una variable con los parámetros necesarios en caso de serlo.

Para manejar la conexión a la base de datos se emplean *pools* de conexiones, con los que una vez obtenida la conexión se ejecuta la consulta. A continuación, la función reléase() permite liberar la conexión. Por último, se manejan los resultados tanto en caso de error como de éxito.

```
//Metodo para mostrar todas las prendas
list(callback) {
    let query = "SELECT * FROM PRENDAS WHERE activo = 1 ";
    let params = [];

    // Utiliza el pool de conexiones para manejar la conexión a la base de datos
    pool(function (connection) {

        connection.query(query, params, (error, resultados) => {

            connection.release();// Libera la conexión después de realizar la consulta
            if (error) {
                console.log("Error al mostrar las prendas:", error);
                callback(error, null); // Llama al callback con el error y sin resultados
            } else {
                callback(null, resultados);
            }
        });
    });
}
```

Figura 23- Ejemplo función de capa de acceso a datos

5.2.4 Implementación rutas

Para los distintos módulos de la aplicación se han definido rutas, cada una asociada al método HTTP correspondiente: POST, GET, PUT. Para aquellas que requieren subida de archivos se ha incluido multer, un middleware que permite recibir y procesar archivos adjuntos enviados desde formularios HTML.

Por último, cada ruta está vinculada a una función del controlador que le corresponde, el cual maneja la lógica específica de esa ruta.

```
var express = require('express');
var router = express.Router();
const multer = require('multer');
const multerFactory = multer({ storage: multer.memoryStorage() });
const userController = require('../controllers/usuariosController');

router.post('/create', multerFactory.single("foto"), userController.createUser);
router.post('/delete', userController.deleteUser);
router.get('/photo/:id', userController.getUserPhoto);
router.get('/indexAdmin', userController.indexAdmin);
router.get('/inicio',userController.inicio);
router.post('/updateProfile/:id', multerFactory.single("foto"), userController.updateProfile);
router.post('/updateAdmin', userController.updateAdmin);
router.get('/obtenerUsuario/:id', userController.getUserById);
router.get('/logout', userController.logout);

//... resto de funciones
module.exports = router;
```

Figura 24- Ejemplo de capa de rutas

5.2.5 Gestión de parámetros

En la gestión de prendas y outfits de este proyecto, se menciona de forma recurrente el uso de una serie de parámetros que los caracterizan.

Estos parámetros tienen como objetivo mejorar la estructura y organización del armario virtual, permitiendo categorizar las prendas o los outfits facilitando su búsqueda filtrado y en el caso de prendas combinación.

Podemos encontrar dos grupos de parámetros, aquellos que hacen referencia a prendas como son:

- Colores: permite etiquetar una prenda según su tonalidad.
- Estilos: permite etiquetar una prenda según su estética.
- Estaciones: permite etiquetar una prenda según la temporada en la que es más adecuada.

- Categorías: permite clasificar una prenda según la función que tiene.
- Tipos: permite clasificar prendas dentro de una determinada categoría.

Por otro lado, para outfits encontramos:

- Estaciones: permite etiquetar un outfit según la temporada en la que es más adecuado.
- Eventos: permite etiquetar un outfit según el entorno u ocasión para el que puede ser usado.
- Tags: son palabras clave que permiten complementar la información del outfit.

Para una mejor organización del código y evitar su repetición se han definido unas funciones que permiten cargar, añadir, eliminar y guardar los parámetros. El objetivo de estas funciones es facilitar el desarrollo de la lógica de prendas y outfits utilizando esta gestión común.

5.2.5.1 Cargar parámetros

Esta es una función utilizada en el front-end cuando queremos cargar dinámicamente los campos de los parámetros correspondientes dentro un modal.

En este caso se proporciona el endPoint desde donde obtenemos nuestro parámetro y se detalla el ID del contenedor donde se quieren añadir estos campos. Por último, se actualiza el valor del input con los IDs de los correspondientes parámetros seleccionados, en caso de haberlos.

```
//Cargar parametros
cargarColores('/prendas/colores', 'colorOpciones', 'prendaColores');
cargarCategorias('/prendas/categorias', 'categoriaOpciones', 'prendaCategoria','tipoOpciones');
cargarEstilos('/prendas/estilos', 'estiloOpciones', 'prendaEstilos');
cargarEstaciones('/prendas/estaciones','estacionOpciones','prendaEstacion');
```

Figura 25- Llamada a función de carga de parámetros

```
function cargarEstilos(endpoint, containerId, inputId) {
 fetch(endpoint)
    .then(response => response.json())
    .then(estilos => {

        const containerEstilos = document.getElementById(containerId);
        const estilosSeleccionados = [];

        estilos.forEach(estilo => {
            const estiloOption = document.createElement('div');
            estiloOption.className = 'option d-flex align-items-center';
            estiloOption.style.cursor = 'pointer';
            estiloOption.textContent = estilo.nombre;
            estiloOption.setAttribute('data-value', estilo.id);

            estiloOption.onclick = () => {
                estiloOption.classList.toggle('selected');

                if (estilosSeleccionados.includes(estilo.id)) {
                    estilosSeleccionados.splice(estilosSeleccionados.indexOf(estilo.id), 1);
                } else {
                    estilosSeleccionados.push(estilo.id);
                }
                if (inputId) {
                    document.getElementById(inputId).value = estilosSeleccionados.join(',');
                }
            };

            containerEstilos.appendChild(estiloOption);
        });
    })
    .catch(error => console.error('Error al cargar estilos:', error));
}
```

Figura 26- Código de función de carga de parámetro

5.2.5.2 Añadir/eliminar parámetros

Esta es una función empleada cuando se quiere añadir, modificar o eliminar una prenda o un outfit.

El proceso que sigue esta función es, en primer lugar, tomar el ID de la prenda/outfit junto con el ID del correspondiente parámetro de aquellos que el usuario ha seleccionado desde la intefaz, esto es enviado a la capa de acceso a datos donde la relación entre estos dos IDs se activa o desactiva según la necesidad.

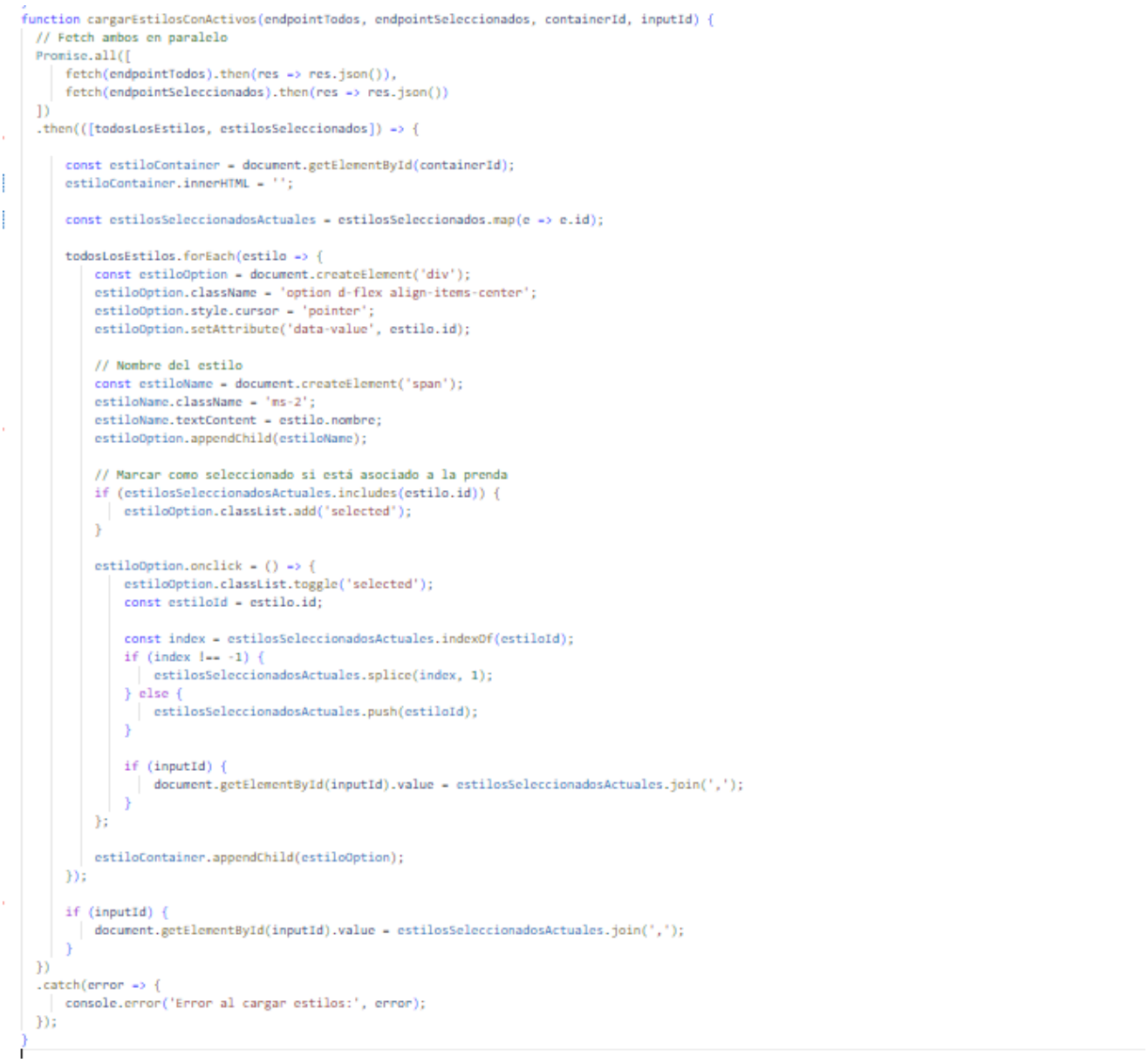
```
addEstilosToPrenda(prendaId, estilos, callback){
    if (estilos && estilos.length > 0) {
    const query = `
    INSERT INTO prenda_estilo (prenda_id, estilo_id, activo)
    VALUES ?
    ON DUPLICATE KEY UPDATE activo = 1;`;

    const values = estilos.map(estiloId => [prendaId, estiloId,1]);

    pool(function (connection) {
        // Realiza la consulta SQL para insertar los estilos
        connection.query(query, [values], (error, resultados) => {
            // Liberamos la conexión
            if (error) {
                console.log("Error al insertar estilos:", error);
                return callback(error);
            }
                console.log("Estilos insertados correctamente:", resultados);
                return callback(null);
        });
    });
}else{
    callback(null);
}
};
```

Figura 27- Código para añadir parámetro a la BBDD

5.2.5.3 Obtener parámetros

En la función mostrar una prenda u outfit se puede volver a observar cómo los parámetros son cargados de forma dinámica, y en este caso, entre ellos puede haber algunos visiblemente marcados. Estos parámetros "seleccionados" son los que están activos y que por lo tanto son aquellos que tiene la prenda o el outfit definidos.

En este caso se realiza un fetch en paralelo pues necesitamos conocer cuáles son todos los parámetros disponibles y cuáles son los que están seleccionados.



```
function cargarEstilosConActivos(endpointTodos, endpointSeleccionados, containerId, inputId) {
  // Fetch ambos en paralelo
  Promise.all([
      fetch(endpointTodos).then(res => res.json()),
      fetch(endpointSeleccionados).then(res => res.json())
  ])
  .then(([todosLosEstilos, estilosSeleccionados]) => {

      const estiloContainer = document.getElementById(containerId);
      estiloContainer.innerHTML = '';

      const estilosSeleccionadosActuales = estilosSeleccionados.map(e => e.id);

      todosLosEstilos.forEach(estilo => {
          const estiloOption = document.createElement('div');
          estiloOption.className = 'option d-flex align-items-center';
          estiloOption.style.cursor = 'pointer';
          estiloOption.setAttribute('data-value', estilo.id);

          // Nombre del estilo
          const estiloName = document.createElement('span');
          estiloName.className = 'ms-2';
          estiloName.textContent = estilo.nombre;
          estiloOption.appendChild(estiloName);

          // Marcar como seleccionado si está asociado a la prenda
          if (estilosSeleccionadosActuales.includes(estilo.id)) {
              estiloOption.classList.add('selected');
          }

          estiloOption.onclick = () => {
              estiloOption.classList.toggle('selected');
              const estiloId = estilo.id;

              const index = estilosSeleccionadosActuales.indexOf(estiloId);
              if (index !== -1) {
                  estilosSeleccionadosActuales.splice(index, 1);
              } else {
                  estilosSeleccionadosActuales.push(estiloId);
              }

              if (inputId) {
                  document.getElementById(inputId).value = estilosSeleccionadosActuales.join(',');
              }
          };

          estiloContainer.appendChild(estiloOption);
      });

      if (inputId) {
          document.getElementById(inputId).value = estilosSeleccionadosActuales.join(',');
      }
  })
  .catch(error => {
      console.error('Error al cargar estilos:', error);
  });
}
```

Figura 28- Código carga de parámetro con activos

5.2.6 Módulo de gestión del administrador

Las funciones desarrolladas en este módulo permiten a un administrador gestionar los usuarios de la aplicación. Añadir, modificar, buscar, actualizar estado de administrador y eliminar siguen una estructura donde se usa la función fetch con su correspondiente ruta y petición, la cual se encarga de iniciar dicha solicitud HTTP al servidor.

Para las operaciones de modificar, actualizar estado de administrador y eliminar se toma el ID del usuario de forma dinámica desde el listado presente en la interfaz del administrador.

Las funciones añadir y modificar incluyen una validación previa de los campos antes de enviarse al servidor.

Para una mejor visualización se ha implementado una paginación que permite mostrar el listado de usuarios en páginas de diez en diez.

5.2.7 Módulo de gestión de usuarios

5.2.7.1 Registro de usuarios

Para está función se ha optado por el uso de un modal que incluye un formulario con varios pasos. A medida que el usuario rellena los correspondientes campos el sistema comprueba su validez permitiendo al usuario avanzar en éste o al contrario, mostrándole un mensaje de error.

A continuación, en la figura 20 se muestra una captura del código empleado para implementar la validación de cada campo en cada paso correspondiente.

```
//Validacion pasos del formulario
function validateStep(stepIndex) {

    if (stepIndex === 0) {
        const email = document.getElementById('input_email').value.trim();
        const name = document.getElementById('input_name').value.trim();
        const surname = document.getElementById('input_surname').value.trim();
        const username = document.getElementById('input_username').value.trim();
        const password = document.getElementById('input_password').value;
        const password2 = document.getElementById('input_password2').value;

        if (!email || !name || !surname || !username || !password || !password2|| username === '@') {
            errorMessage.textContent = 'Completa todos los campos del paso 1.';
            errorMessage.style.display = 'block';
            return false;
        }
        const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
        if (!emailRegex.test(email)) {
            errorMessage.textContent = 'Introduce un correo electrónico válido.';
            errorMessage.style.display = 'block';
            return false;
        }
        if (password !== password2) {
            errorMessage.textContent = 'Las contraseñas no coinciden.';
            errorMessage.style.display = 'block';
            return false;
        }

        const passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{8,}$/;
        if (!passwordRegex.test(password)) {
            errorMessage.textContent = 'La contraseña debe tener al menos 8 caracteres, incluir mayúsculas, minúsculas, números y un símbolo.';
            errorMessage.style.display = 'block';
            return false;
        }
    }

    if (stepIndex === 1) {
        const formalidad = document.getElementById('input_formalidad').value;
        const simplicidad = document.getElementById('input_simplicidad').value;
        const estiloTemporal = document.getElementById('input_estilo_temporal').value;
        const sofisticacion = document.getElementById('input_sofisticacion').value;

        if (!formalidad || !simplicidad || !estiloTemporal || !sofisticacion) {
            errorMessage.textContent = 'Completa todos los campos del paso 2.';
            errorMessage.style.display = 'block';
            return false;
        }
    }

    if (stepIndex === 2) {
        const rangoEdad = document.getElementById('input_edad').value;
        const photoElement = document.getElementById('input_photo');
        if (!rangoEdad || !photoElement.files.length) {
            errorMessage.textContent = 'Completa todos los campos del paso 3.';
            errorMessage.style.display = 'block';
            return false;
        }
    }

    errorMessage.style.display = 'none';
    return true;

}
```

Figura 29- Función de validación de campos de registro

Una vez validados estos datos, se recogen mediante la función fetch, se realiza la solicitud HTTP a la ruta /auth/signup y por último se envían los datos al servidor.

```
// Manejo del formulario de registro
  const formSignup = document.getElementById('form-signup');
  formSignup.addEventListener('submit', async (event) => {
      event.preventDefault();

    //step1
    const email = document.getElementById('input_email').value;
    const name = document.getElementById('input_name').value;
    const surname = document.getElementById('input_surname').value;
    const username = document.getElementById('input_username').value;
    const password = document.getElementById('input_password').value;
    //step2
    const formalidad = document.getElementById('input_formalidad').value;
    const simplicidad = document.getElementById('input_simplicidad').value;
    const estiloTemporal = document.getElementById('input_estilo_temporal').value;
    const sofisticacion = document.getElementById('input_sofisticacion').value;
    //step3
    const photoElement = document.getElementById('input_photo');
    const rangoEdad = document.getElementById('input_edad').value;

      const reader = new FileReader();
      reader.onloadend = async () => {
          const photoBase64 = reader.result.split(',')[1];

          try {
              const response = await fetch('/auth/signup', {
                  method: 'POST',
                  headers: { 'Content-Type': 'application/json' },
                  body: JSON.stringify({
                  email,
                  password,
                  admin: 0,
                  nombre_usuario: username,
                  nombre: name,
                  apellidos: surname,
                  foto: photoBase64,
                  activo: 1,
                  preferencias: {
                      formalidad,
                      simplicidad,
                      estiloTemporal,
                      sofisticacion,
                      rangoEdad
                  }
              })
              });
              if (response.ok) {
                  const data = await response.json();
                  window.location.href = data.redirectUrl;
              } else {
                  const data = await response.json();
                  errorMessage.textContent = data.message;
                  errorMessage.style.display = 'block';
              }
          } catch (error) {
              console.error('Error:', error);
              alert('Error al conectar con el servidor');
          }
      };
```

Figura 30- Función de manejo de formulario de registro

En la capa de servicio se realizan varias comprobaciones para asegurar que no exista ya un usuario activo con el mismo correo o nombre de usuario.

En la capa de acceso a datos, para poder almacenar las contraseñas en la base de datos de forma segura, se ha empleado Bcrypt, una función de hash que toma la contraseña introducida por el usuario y la transforma en una secuencia de caracteres de longitud fija.

En caso de éxito se redirigiría al usuario a su menú de inicio, y en caso de error se mostraría un mensaje con la causa del problema.

5.2.7.2 Inicio de sesión

Para esta función se ha optado por un formulario común, compuesto de dos campos, correo electrónico y contraseña. Tal y como se muestra a continuación, los datos se recogen y en este caso se realiza la solicitud HTTP a través de la ruta /auth/login.

```
formLogin.addEventListener('submit', async (event) => {
    event.preventDefault();
    errorMessage.style.display = 'none';
    const email = emailInput.value;
    const password = passwordInput.value;

    try {
        const response = await fetch('/auth/login', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ email, password })
        });


        if (response.ok) {
            const data = await response.json();
            errorMessage.style.display = 'none';
            window.location.href = data.redirectUrl;
        } else if (response.status === 401) {
            errorMessage.textContent = '❌ Usuario o contraseña incorrectos.';
        } else if (response.status === 403) {
            errorMessage.textContent = '⚠️ Tu cuenta está inactiva. Contacta con soporte.';
        } else {
            errorMessage.textContent = '❌ Ocurrió un error. Inténtalo de nuevo.';
        }
        if (!response.ok) {
        errorMessage.style.display = 'block';
        }
} catch (error) {
    console.error('Error:', error);
    errorMessage.textContent = '❌ Error al conectar con el servidor. Inténtalo más tarde.';
    errorMessage.style.display = 'block';
}
});
});
```

Figura 31- Función de manejo de formulario de inicio de sesión

En la capa de servicio se comprueba si las credenciales introducidas son correctas. Para ello, la contraseña introducida se hashea con Bcrypt y se compara con el hash almacenado anteriormente, de forma que sí coinciden la autenticación tiene éxito, pudiendo en este caso acceder al menú de inicio de la aplicación.

Al contrario, en caso de datos incorrectos se mostraría un mensaje de error. O en el caso de que el usuario haya dado de baja su cuenta previamente se le mostraría un mensaje de aviso.

5.2.7.3 Ver estadísticas

Para la implementación de los gráficos que conformarían la interfaz de estadísticas del usuario se han empleado la librería Chart.js, la cual mediante los datos que se obtenían de forma dinámica se generaron los diagramas de "donut" y de barras.

En cuanto a la visualización de calendario que recoge las prendas creadas se realizó mediante la librería FullCalendar.

En ambos casos sus códigos predefinidos facilitaron la tarea de implementación.

5.2.8 Módulo de prendas

5.2.8.1 Añadir prendas

Para la implementación de esta funcionalidad se ha empleado un modal donde el usuario debe adjuntar obligatoriamente una imagen y seleccionar una categoría, un tipo y al menos una estación, un color y un estilo.

Dentro de los puntos a destacar de la implementación de este módulo encontramos cómo se realizó la incorporación de Rembg, una librería de Python mediante la cual se llevó a cabo la eliminación del fondo de la imagen que adjuntamos en el formulario. Para ello, mediante Node.js se ejecutó el script que se muestra en la Figura 32 como un proceso externo.

```python
import sys
from rembg import remove
import base64

def remove_background(base64_image):
    input_data = base64.b64decode(base64_image)
    output_data = remove(input_data)
    return base64.b64encode(output_data).decode('utf-8')

if __name__ == "__main__":
    base64_image = sys.stdin.read()
    result = remove_background(base64_image)
    print(result)
```

Figura 32- Script para la eliminación del fondo de imágenes

El funcionamiento de este script consiste en la recepción de una imagen codificada en Base64 como parámetro de entrada, su decodificación a su forma original y a continuación la utilización de la función remove() de la librería Rembg. Finalmente, vuelve a codificar la imagen sin fondo en Base64 y la muestra como resultado en la salida.

Para una mejor experiencia de usuario se incluyó un spinner que se muestra mientras se procesa la eliminación del fondo de la imagen.

Una vez el usuario pulsa al botón añadir, se procede a validar los datos correspondientes, y a realizar la solicitud HTTP a la ruta /prendas/create.

En caso de éxito en la función se incorporó un modal con un mensaje que confirmaba que la prenda se había añadido correctamente.

```
document.getElementById('addPrendaForm').addEventListener('submit', (event) => {
  event.preventDefault();

  document.querySelectorAll('.text-danger').forEach(error => error.classList.add('d-none'));

   const userIdElement = document.getElementById('userId');
   userId = userIdElement ? userIdElement.value : null;

  const photoElement = document.getElementById('prendaFoto');
  const categoria = document.getElementById('prendaCategoria').value;
  const tipo = document.getElementById('prendaTipo').value;
  const colores = document.getElementById('prendaColores').value.split(',');
  const estacion =  document.getElementById('prendaEstacion').value.split(',');
  const estilos = document.getElementById('prendaEstilos').value.split(',');

  // Validar campos
  if (!photoElement.files.length) {
    document.getElementById('errorFoto').classList.remove('d-none');
    return;
  }
   //Validacion resto de campos..
    const reader = new FileReader();
    reader.onloadend = async () => {
      const photoBase64 = reader.result.split(',')[1];
    // Mostrar cargando
    document.getElementById('spinner').style.display = 'flex';
      try {
        const response = await fetch('/prendas/create', {
          method: 'POST',
          headers: {
            'Content-Type': 'application/json'
          },
          body: JSON.stringify({
            foto: photoBase64,
            activo:1,
            categoria,
            tipo,
            colores,
            estacion,
            estilos,
            usuario:userId
          })

        });

        if (response.ok) {
          const modalAddPrenda = bootstrap.Modal.getInstance(document.getElementById('addPrendaModal'));
          if (modalAddPrenda) {
           modalAddPrenda.hide();
          }
          mostrarModalExito();
        } else {
          const errorText = await response.text();
          console.error('Error de respuesta:', errorText);
          alert('Error creando prenda: ' + errorText);
        }
      } catch (error) {
        console.error('Error:', error);
        alert('Error creando prenda');
      }
       finally {
      // Ocultar cargando
      document.getElementById('spinner').style.display = 'none';
      }
      };

      reader.readAsDataURL(photoElement.files[0]);
  });
  });
```

Figura 33- Manejo de formulario de añadir prenda

5.2.8.2 Listar prendas

Para mostrar las prendas del armario del usuario se implementaron una serie de pestañas dinámicas, las cuales permiten mostrar las prendas por su correspondiente categoría y tipo.

Estas pestañas corresponden a "Todas las prendas ", "Todas las prendas de una categoría" y una vez seleccionada la categoría, "Todas las prendas de un categoría por tipo".

En este caso se optó por el uso de plantillas ejs donde ejecutamos código JavaScript entre <% %> para insertar los valores correspondientes en el HTML.

```
<div class="tab-content" id="categoryTabsContent">
    <!-- Pestaña Todo -->
    <div class="tab-pane fade show active" id="all" role="tabpanel" aria-labelledby="all-tab">
        <div class="album py-5 bg-light">
            <div class="container">
                <div class="row" id="ropaGrid">
                    <% if (prendas && prendas.length > 0) { %>
                        <% prendas.forEach(prenda => { %>
                            <div class="col-md-2 mb-4">
                                <div class="card mb-4 shadow-sm d-flex h-100 align-items-center justify-content-center fixed-card" onclick="openPrendaModal('<%= prenda.id_prenda %>')">
                                    <img src="/prendas/photoPrenda/<%= prenda.id_prenda %>" class="bd-placeholder-img card-img-top" alt="<%= prenda.id_prenda %>">
                                </div>
                            </div>
                        <% }) %>
                    <% } else { %>
                        <p class="text-center">No hay prendas disponibles en esta categoría.</p>
                    <% } %>
                </div>
            </div>
        </div>
    </div>
```

Figura 34- Función para listar prendas en pestañas

Mediante la función openPrendaModal() podemos conocer más información y ver en detalle cada una de las prendas.

5.2.8.3 Mostrar prenda

Esta funcionalidad ha sido desarrollada de forma que se inicie cuando un usuario pulsa sobre una prenda, ésto permite hacer la petición fetch al endpoint del servidor /prendas/obtenerPrenda/${prenda_id} pasando el ID de la prenda seleccionada y poder obtener la información que posteriormente se mostrará.

```
function openPrendaModal(prenda_id) {

    fetch(`/prendas/obtenerPrenda/${prenda_id}`)
        .then(response => {
            if (!response.ok) {
                throw new Error('Error al obtener los detalles de la prenda');
            }
            return response.json();
        })
        .then(prenda => {
            console.log(prenda);
            mostrarPrendaModal(prenda);
        })
        .catch(error => {
            console.error('Error:', error);
            alert('No se pudieron obtener los detalles de la prenda.');
        });
}
```

Figura 35- Función obtención información de prenda

A continuación se muestra toda la información de la prenda, obtenida de forma dinámica y mostrada en sus respectivos contenedores.

```
function mostrarPrendaModal(prenda) {

    const modal = document.getElementById('prendaModal');

    modal.querySelector("#modalImage").src = `/prendas/photoPrenda/${prenda.id_prenda}`;
    modal.querySelector("#modalCategoria").innerText = prenda.categoria;
    modal.querySelector("#modalTipo").innerText = prenda.tipo;

    //estaciones
    const estacionesDiv = modal.querySelector("#modalEstacion");
    estacionesDiv.innerHTML = ''; // Limpiar contenido previo
    if (prenda.estaciones && Array.isArray(prenda.estaciones)) {
        prenda.estaciones.forEach(estacion => {
        const badge = document.createElement('span');
        badge.className = 'badge bg-secondary me-1 mb-1';
        badge.textContent = estacion.nombre.trim();
        estacionesDiv.appendChild(badge);
       } );
    } else {
        estacionesDiv.innerText = 'Sin estaciones asignadas';
    }
    //
    //estilos,colores    ------------------------
    // Mostrar el modal
    const bootstrapModal = new bootstrap.Modal(modal);
    bootstrapModal.show();

}
```

Figura 36- Función para mostrar información de prenda

5.2.8.4 Modificar prenda

Esta funcionalidad se ha implementado partiendo de la base del desarrollo de la función mostrar prenda. Para ello, una vez se visualiza la prenda que se desea modificar, tal y como se mencionaba en el apartado de gestión de parámetros, al pulsar sobre la

sección del nombre del parámetro que se quiere cambiar, se desplegaría un contenedor con todos los campos pertenecientes a ese parámetro .

Se observa que los parámetros activos se encuentran visiblemente marcados y es el usuario el que puede quitar o añadir otros nuevos simplemente pulsando sobre ellos.

En la Figura 37 se muestra una captura del código que se corresponde a la implementación de la lógica de apertura y cierre de la sección de cada parámetro y la correspondiente función de carga en cada una de ellas.

```
function mostrarParametros(section) {
    const sections = ['categoria', 'tipo', 'colores', 'estacion', 'estilos'];

    sections.forEach(sec => {
        const editDiv = document.getElementById(`edit${sec.charAt(0).toUpperCase() + sec.slice(1)}Div`);
        if (sec === section) {
            const mostrar = editDiv.style.display === 'none';
            editDiv.style.display = mostrar ? 'block' : 'none';

            if (mostrar) {
                if (sec === 'categoria' && !categoriasCargadas) {
                    cargarCategorias('/prendas/categorias', 'categoriaOpcionesDetalles', 'prendaCategoria');
                    categoriasCargadas = true;
                } else if (sec === 'tipo' && !tiposCargados) {
                    getCategoria().then(categoriaId => {
                        cargarTipos(categoriaId, 'tipoOpcionesDetalles');
                        tiposCargados = true;
                    }).catch(error => console.error('Error al obtener categoria:', error));
                } else if (sec === 'colores' && !coloresCargados) {
                    cargarColoresConActivos('/prendas/colores', `/prendas/${id_prenda}/colorByPrenda`, 'colorOpcionesDetalles', 'prendaColores');
                    coloresCargados = true;
                } else if (sec === 'estilos' && !estilosCargados) {
                    cargarEstilosConActivos('/prendas/estilos', `/prendas/${id_prenda}/estiloByPrenda`, 'estiloOpcionesDetalles', 'prendaEstilos');
                    estilosCargados = true;
                }
                else if (sec === 'estacion' && !estacionesCargadas) {
                    cargarEstacionesConActivos('/prendas/estaciones', `/prendas/${id_prenda}/estacionByPrenda`, 'estacionOpcionesDetalles', 'prendaEstacion');
                    estacionesCargadas = true;
                }
            }
        } else {
            editDiv.style.display = 'none';
        }
    });
}
```

Figura 37- Función para mostrar parámetros a modificar

La función clave para guardar los cambios es saveChanges(field), donde field hace referencia al parámetro que se quiere cambiar, y a continuación éste le deriva a la función de gestión que le corresponda.

```
function saveChanges(field) {
    let value;
    switch (field) {
        case "categoria":
            value = document.getElementById('prendaCategoria').value;
            break;
        case 'tipo':
            value = document.getElementById('prendaTipo').value;
            break;
            case 'colores':
                gestionColores();
                 break;
        case 'estacion':
            gestionEstaciones();
            break;
        case 'estilos':
           gestionEstilos();
            break;
        default:
            console.error('Campo no válido para guardar cambios');
            return;
    }


    fetch('/prendas/updatePrenda', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({
            prendaId: id_prenda,
            field: field,
            value: value
        })
    })
    //...
```

Figura 38- Función guardar cambios para modificación de prenda

Las funciones de gestión son las encargadas de recoger todos los IDs de los parámetros seleccionados y de enviar una petición POST al servidor para actualizar esas relaciones de la prenda.

```
function gestionEstilos() {
    const estilosSeleccionados = document.getElementById('prendaEstilos').value
        .split(',')
        .map(e => e.trim())
        .filter(e => e !== '');

    fetch(`/prendas/${id_prenda}/estilos`, {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ estilos: estilosSeleccionados })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            alert('Estilos actualizados correctamente');
            fetch(`/prendas/obtenerPrenda/${id_prenda}`)
                .then(r => r.json())
                .then(p => {
                    actualizarModal(p);
                    mostrarParametros("estilos");
                    estilosOriginales = p.estilos.map(e => e.id);
                });
        } else {
            alert('Error al actualizar estilos');
        }
    })
    .catch(err => {
        console.error("Error al actualizar estilos:", err);
        alert("Hubo un problema al guardar los estilos");
    });
}
```

Figura 39- Función de gestión de parámetro para modificación de prenda

5.2.8.5 Filtrar y buscar prendas

En esta función se emplea el uso de parámetros tal y como se menciona en el apartado 5.2.5.1.

En el caso del filtrado, la función toma aquellos campos que hemos seleccionado y nos muestra el resultado, mientras que en el caso de la búsqueda el campo es un texto que introduce el usuario.

Cuando el sistema encuentra resultados, el contenedor actual que incluye las prendas desaparece y aparece otro con los resultados correspondientes.

Para volver atrás en el caso de la búsqueda, si el campo de búsqueda se encuentra vacío vuelve a aparecer el contenedor predefinido, mientras que en el caso del filtrado se ha implementado un botón de "reestablecer".

5.2.9 Módulo outfits

5.2.9.1 Añadir outfits

La implementación seguida para la función añadir outfits, sigue una estructura similar a la mencionada anteriormente en el apartado de "añadir prendas".

El modal de añadir outfit a diferencia del utilizado en el módulo de prendas, consta de sus parámetros, un atributo nombre y un paso en el modal que consiste en un "contenedor" donde se muestra la previsualización de las prendas que conformaran el outfit.

Las prendas disponibles del usuario, es decir, aquellas que el usuario haya añadido anteriormente, se mostraran divididas en categorías en un acordeón desarrollado con Bootstrap. Su carga se realiza tal y como se muestra en la Figura 40.

```
async function cargarPrendas() {
    try {
        const response = await fetch('/prendas/listarPrendasJSON');
        if (!response.ok) throw new Error('No se pudo obtener las prendas');

        const data = await response.json();

        // Limpiar contenedores
        const contenedores = document.querySelectorAll('.accordion-body');
        contenedores.forEach(contenedor => contenedor.innerHTML = '');

        // Clasificar y mostrar las prendas en sus contenedores correspondientes
        data.prendas.forEach(prenda => {
            const div = document.createElement('div');
            div.classList.add('prenda-item');

            const img = document.createElement('img');
            img.src = `/prendas/photoPrenda/${prenda.id_prenda}`;
            img.alt = prenda.nombre;
            img.alt = prenda.id_prenda;
            img.classList.add('prenda-image');
            img.dataset.id = prenda.id_prenda;
            img.dataset.categoria = prenda.categoria;

            img.addEventListener('click', () => {
                img.classList.toggle('seleccionada');
                actualizarPrevisualizacion();
            });

            const containerId = prenda.categoria.replace(/\s+/g, '').toLowerCase();
            const contenedor = document.getElementById(containerId);
            if (contenedor) {
                contenedor.appendChild(div);
                div.appendChild(img);
            } else {
                console.warn(`Contenedor no encontrado para la categoria: ${prenda.categoria}`);
            }

        });
    } catch (error) {
        console.error('Error al cargar las prendas:', error);
        alert('Hubo un error al cargar las prendas.');
    }

}
```

Figura 40- Función para cargar prendas en el acordeón

En el contenedor de previsualización el usuario puede desplazar las prendas de ropa escogidas del acordeón pudiendo colocarlas a su gusto. Para ello se utilizó en primer lugar una función donde se utiliza código de la biblioteca Interact.js junto con una función para la gestión de la previsualización de prendas.

```
function makeDraggable(el) {
  interact(el)
    .draggable({
      listeners: {
        move(event) {
          const target = event.target;
          const x = (parseFloat(target.getAttribute('data-x')) || 0) + event.dx;
          const y = (parseFloat(target.getAttribute('data-y')) || 0) + event.dy;

          target.style.transform = `translate(${x}px, ${y}px) scale(${target.dataset.scale || 1}) rotate(${target.dataset.rotate || 0}deg)`;
          target.setAttribute('data-x', x);
          target.setAttribute('data-y', y);
        }
      }
    })
    .resizable({
      edges: { left: true, right: true, bottom: true, top: true },
      listeners: {
        move(event) {
          const target = event.target;
          let { width, height } = event.rect;

          // Quitar restricciones visuales
          target.style.maxWidth = 'none';
          target.style.maxHeight = 'none';

          // Aplicar tamaño actualizado
          target.style.width = width + 'px';
          target.style.height = height + 'px';
        }
      },
      modifiers: [
        interact.modifiers.restrictSize({
          min: { width: 200, height: 200 },
          max: { width: 3000, height: 3000 }
        })
      ]
    });
```

Figura 41- Función de manejo de arrastre de prendas

Para una posterior visualización del outfit final, el resultado obtenido se convirtió en imagen mediante la librería html2canvas.js. Tal y como se muestra en la captura de código de la Figura 42 solo fue necesaria una llamada a su función y detallar el formato de guardado.

```
// Guardar imagen
const previewContainer = document.getElementById('previewContainer');
html2canvas(previewContainer).then(async (canvas) => {
    const imagenBase64 = canvas.toDataURL('image/png'); // Convertir a base64

    // Crear el objeto con los datos del outfit
    const outfitData = {
        nombre: nombre,
        prendas: selectedPrendas,
        estacion: estacion,
        eventos: eventos,
        tags: tags,
        imagen: imagenBase64, // Añadir la imagen en base64
        publico: esPublico
    };

    try {
        // Enviar los datos al servidor
        const response = await fetch('/outfits/create', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(outfitData)
        });

        if (response.ok) {
            const modalAddOutfit = bootstrap.Modal.getInstance(document.getElementById('createOutfitModal'));
            if (modalAddOutfit) {
                modalAddOutfit.hide();
            }
            mostrarModalExitoOutfit();

        } else {

            const errorText = await response.text();
            console.error('Error de respuesta:', errorText);
            alert('Error creando outfit: ' + errorText);

        }
    } catch (error) {
        console.error('Error al guardar el outfit:', error);
        alert('Hubo un error al guardar el outfit.');
    }
```

Figura 42- Código de perteneciente a la función de guardar outfit

El objetivo que se quería lograr con esta función era favorecer la visualización al seleccionar prendas para conformar un outfit.

5.2.9.2 Listar outfits

La implementación de esta función se basó en la utilizada para listar prendas, tal y como se menciona en el apartado 5.2.8.2.

Para ello se modificaron las pestañas anteriores por una clasificación de outfits públicos o no públicos. También la interfaz del listado de outfits cambio, mostrándose en este caso la imagen de cada outfit más grande acompañada de su nombre.

5.2.9.3 Mostrar outfits

La implementación de esta función se basó en la utilizada para mostrar los detalles de una prenda, tal y como se menciona en el apartado 5.2.8.3.

Para ello se incorporaron los parámetros propios de un outfit y se añadió un botón de opciones en su esquina izquierda, con la intención de facilitar al usuario la eliminación del outfit y la descarga de la foto creada con la combinación de prendas.

5.2.9.4 Modificar outfits

La implementación de esta función se basó en la utilizada para modificar las características de una prenda, tal y como se menciona en el apartado 5.2.8.4.

En este caso se realizó la implementación con los parámetros requeridos para outfits, se desarrolló un botón deslizante que permitiera cambiar el estado de publicado/ no publicado de un outffit, y un botón que al desplegarse permitiera modificar el nombre del outfit de una forma sencilla.

5.2.9.5 Otras funciones

La implementación de las funciones filtrar outfits, buscar outfits siguen las mismas estructuras a la mencionadas en el apartado 5.8.2.5 de filtrar y buscar prendas.

5.2.10 Modulo comunidad

En este módulo se han implementado las funciones de comunidad, mediante las cuales los usuarios pueden ver los outfits que otros usuarios hacen públicos.

5.2.10.1 Buscar usuarios

Para realizar esta función en primer lugar se debe tomar la búsqueda introducida por el usuario. Al igual que en anteriores funciones de búsqueda, el contenedor predefinido (en este caso con los usuarios) desaparece una vez el usuario introduce texto en la barra de búsqueda y en caso de ésta estar vacía, el contenedor vuelve a su estado inicial.

A continuación, se realiza una solicitud fetch a la ruta especificada con el término buscado por el usuario. En caso de éxito en la respuesta, se muestran los elementos del listado con su información acompañados de un botón de "seguir", al contrario se mostraría un mensaje de aviso por no encontrarse coincidencias.

```
async function buscarUsuario() {
    const searchTerm = inputBusqueda.value.trim().toLowerCase();
    sugerenciasUsuarios.innerHTML = '';

    if (searchTerm === '') {
        cargarSugerencias(); // Volver a mostrar discovery
        return;
    }

    try {
        const response = await fetch(`/comunidad/buscar?query=${encodeURIComponent(searchTerm)}`);
        const contentType = response.headers.get("content-type");

        if (!response.ok || !contentType.includes("application/json")) {
            throw new Error("Respuesta no válida del servidor");
        }

        const usuarios = await response.json();

        if (!Array.isArray(usuarios) || usuarios.length === 0) {
            sugerenciasUsuarios.innerHTML = '<p class="text-muted">No se encontraron coincidencias.</p>';

        }
         sugerenciasUsuarios.innerHTML = '';
            usuarios.forEach(usuario => {
        const userHTML = `//codigo tarjeta
    `;

    // Insertar el HTML generado en sugerenciasUsuarios
    sugerenciasUsuarios.insertAdjacentHTML('beforeend', userHTML);
});

    } catch (error) {
        console.error('Error al buscar usuarios:', error);
        sugerenciasUsuarios.innerHTML = '<p class="text-muted">Error al buscar usuarios.</p>';
    }
}
```

Figura 43- Función de búsqueda de usuarios de la comunidad

5.2.10.2 Añadir/Eliminar usuario seguidor/seguido

Para realizar estas funciones, son necesarios los IDs de los dos usuarios que desean seguirse o dejar de seguirse desde el DOM. Estos argumentos se envían a la función fetch, encargada de realizar la solicitud al servidor. Dependiendo de la acción a tomar, en la base de datos se procederá a insertar la relación en caso de no existir o modificar el valor activo.

```
function seguirUsuario(userId) {

    fetch(`/comunidad/addFollowing`, {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ following_id: userId,usuario_id:u })
    })
       .then(response => response.json())
        .then(data => {

            mostrarModalExito();
            cargarSugerencias(); //recargar lista despues de seguir
        })
        .catch(error => console.error("Error al seguir usuario:", error));

        obtenerSeguido(userId);
}
```

Figura 44- Función para gestionar relaciones con usuarios de la comunidad

5.2.10.3 Listar usuarios seguidores/seguidos

Para realizar estas funciones y mostrar a los usuarios de la comunidad sus correspondientes seguidores o seguidos, se ha empleado un modal donde se carga esta información dinámicamente.

En este caso, se realiza una solicitud fetch para obtener los usuarios correspondientes. En caso de éxito en la respuesta, se muestran los elementos del listado con su información acompañados de un botón de eliminación.

```
function cargarSeguidos() {
    const followingList = document.getElementById("listaSeguidos");
    followingList.innerHTML = "<li class='list-group-item'>Cargando...</li>";
    const userId = document.getElementById("userId").value;
    fetch(`/user/getFollowing/${userId}`)
        .then(response => {
            if (!response.ok) {
                throw new Error(`Error en el servidor}`);
            }
            return response.json();
        })
        .then(seguidos => {
            followingList.innerHTML = ""; // Limpiar la lista

            if (!seguidos || seguidos.length === 0) {
                followingList.innerHTML = "<li class='list-group-item'>No sigues a nadie.</li>";
                return;
            }

            seguidos.forEach(user => {
                const li = document.createElement("li");
                li.classList.add("list-group-item", "d-flex", "align-items-center", "justify-content-between");
                    li.innerHTML = `
                //tarjetas usuarios
                    `;


                followingList.appendChild(li);
            });
        })
        .catch(error => {
            console.error("Error al obtener seguidos:", error);
            followingList.innerHTML = `<li class='list-group-item text-danger'>Error al cargar seguidos.</li>`;
        });
}
```

Figura 45- Función para mostrar listado de seguidores

Conclusiones y trabajo futuro

5.3 Conclusiones

Este Trabajo de Fin de Grado materializa una idea personal mediante la aplicación de los conocimientos adquiridos a lo largo del grado. Se ha desarrollado una herramienta que cumple con los objetivos planteados y abre un abanico de posibilidades para continuar incorporando nuevas funcionalidades.

Se ha logrado que esta aplicación web permita a los usuarios introducir sus prendas en un armario virtual, facilitando su posterior gestión; ello, complementado con la creación de outfits, permite ayudar al usuario a maximizar el uso de su vestuario y hacer un correcto uso de él.

Cabe mencionar que la implementación de una comunidad no solo mejora la experiencia del usuario, sino que sirve de gran fuente de inspiración, fomenta la creatividad y el intercambio de ideas.

5.4 Trabajo futuro

Durante el desarrollo de esta aplicación web surgieron posibles funcionalidades que, debido a la limitación del tiempo por tratarse de un desarrollo individual, quedaron fuera del alcance del proyecto.

A continuación se describen algunas de estas funciones, acompañadas de un análisis en términos de impacto y viabilidad técnica.

Cada función ha sido clasificada en niveles alto, medio y bajo permitiendo conocer cuáles son más factibles de implementar partiendo de la tecnología empleada y cuáles podrían tener un mayor impacto en caso de un desarrollo futuro.

5.4.1 Incorporación de un sistema de recomendación

Esta propuesta consiste en la incorporación de un sistema de recomendaciones basado en las preferencias que ingresa el usuario al registrarse, en los parámetros de las prendas del armario virtual y en las interacciones con publicaciones de outfits de otros usuarios.

Impacto

Alto. Está funcionalidad reforzaría la ayuda al usuario a la hora de escoger su vestimenta diaria gracias a las sugerencias personalizadas que podría recibir, lo que mejoraría, significativamente, la experiencia del usuario.

Viabilidad técnica

Media. Esta propuesta requeriría una evaluación previa para determinar qué tipo de sistema de recomendación sería el óptimo teniendo en cuenta su integración en la aplicación actual y su capacidad de generar sugerencias personalizadas.

5.4.2 Mejora de las funciones de comunidad

Esta propuesta consiste en mejorar las funciones de comunidad. El usuario podría recibir solicitudes de amistad y tener la opción de aceptarlas o rechazarlas. También se mejoraría la interfaz donde se visualizan las publicaciones y las interacciones con ellas permitiendo al usuario guardar aquellas que le gusten.

Impacto

Medio. Esta propuesta mejoraría significativamente la experiencia del usuario al crear una comunidad más activa e interactiva, facilitar el seguimiento de otros usuarios y mejorar la visualización de las publicaciones fomentando la búsqueda de inspiración entre los usuarios.

Viabilidad técnica

Alta. Gracias al uso de unas herramientas robustas y escalables estas nuevas funcionalidades se podrían llevar a cabo sin necesidad de realizar cambios en la arquitectura de la aplicación actual.

5.4.3 Optimización del algoritmo de eliminación del fondo de imágenes

Esta propuesta consiste en mejorar el algoritmo de eliminación del fondo de imágenes para permitir que un usuario pueda incluir imágenes de mayor tamaño y se reduzca su tiempo de procesamiento.

Impacto

Alto. La optimización de este algoritmo aumentaría la eficiencia de la aplicación y la satisfacción del usuario a la hora de añadir sus prendas al armario, le ofrecería una mayor libertad a la hora de adjuntar la imagen de la prenda y podría realizar esta acción en un tiempo menor.

Viabilidad técnica

Media. La implementación de esta propuesta requeriría la búsqueda u optimización de una biblioteca gratuita o la evaluación de otras opciones comerciales, las cuales son mayoritariamente mediante suscripción. Así mismo, serían necesarias pruebas que aseguren que la nueva solución cumple con los requisitos de eficiencia y calidad que requiere la aplicación.

5.4.4 Incorporación de un Chatbot

Esta propuesta consiste en la incorporación de un Chatbot (asistente virtual) que mejore la experiencia del usuario ofreciendo un soporte instantáneo a preguntas y sugerencias personalizadas basadas en sus interacciones y preferencias.

Impacto

Medio. La incorporación de un Chatbot mejoraría la accesibilidad a la aplicación y proporcionaría un soporte instantáneo, lo que a su vez se traduce en una experiencia de usuario más fluida y una mayor satisfacción del mismo

Viabilidad técnica

Media. La implementación de un Chatbot sería sencilla pues no requiere construir la funcionalidad desde cero dada la existencia de multitud de APIs y servicios que cumplen esa función. Su integración con la tecnología empleada en el proyecto sería factible, siendo el único problema el mantenimiento, el posible coste de los servicios y los estándares de seguridad y privacidad que debería cumplir.

5.5 Repositorio del código

Url del repositorio : isabelzamarron/Armario-Virtual-TFG

Chapter 6 - Conclusions and future work

5.1 Conclusions

This Final Degree Project materializes a personal idea through the application of the knowledge acquired throughout the degree. A tool has been developed that meets the objectives set and opens up a range of possibilities to continue incorporating new functionalities.

This web application has been able to allow users to introduce their garments in a virtual wardrobe, facilitating their subsequent management; This, complemented with the creation of outfits, allows the user to maximize the use of their wardrobe and make correct use of it.

It is worth mentioning that the implementation of a community not only improves the user experience, but also serves as a great source of inspiration, encourages creativity and the exchange of ideas.

5.2 Future Work

During the development of this web application, possible functionalities arose that, due to the time limitation due to being an individual development, were outside the scope of the project.

Some of these functions are described below, accompanied by an analysis in terms of impact and technical feasibility.

Each function has been classified into high, medium and low levels, allowing us to know which are more feasible to implement based on the technology used and which could have a greater impact in the event of future development.

5.2.1 Adding a recommendation system

This proposal consists of the incorporation of a recommendation system based on the preferences that the user enters when registering, on the parameters of the garments in the virtual wardrobe and on interactions with outfit publications of other users.

Impact

High. This functionality would reinforce the help to the user when choosing their daily clothing thanks to the personalized suggestions they could receive, which would significantly improve the user experience.

Technical feasibility

Medium. This proposal would require a prior assessment to determine what type of recommendation system would be optimal taking into account its integration into the current application and its ability to generate personalized suggestions.

5.2.2 Improving community features

This proposal consists of improving the functions of the community. The user could receive friend requests and have the option to accept or decline them. The interface where publications and interactions with them are viewed would also be improved, allowing the user to save those they like.

Impact

Medium. This proposal would significantly improve the user experience by creating a more active and interactive community, making it easier to follow other users, and improving the display of posts by encouraging users to seek inspiration.

Technical feasibility

High. Thanks to the use of robust and scalable tools, these new functionalities could be carried out without the need to make changes to the architecture of the current application.

5.2.3 Optimizing the Image Background Removal Algorithm

This proposal consists of improving the algorithm for removing the background from images to allow a user to include larger images and reduce their processing time.

Impact

High. Optimizing this algorithm would increase the efficiency of the app and user satisfaction when adding their garments to the closet, give them greater freedom when attaching the image of the garment and be able to perform this action in a shorter time.

Technical feasibility

Medium. The implementation of this proposal would require the search or optimization of a free library or the evaluation of other commercial options, which are mostly subscription-based. Likewise, tests would be necessary to ensure that the new solution meets the efficiency and quality requirements required by the application.

5.2.4 Onboarding a Chatbot

This proposal consists of the incorporation of a Chatbot (virtual assistant) that improves the user experience by offering instant support to personalized questions and suggestions based on their interactions and preferences. ´

Impact

Medium. The addition of a Chatbot would improve accessibility to the application and provide instant support, which in turn translates into a smoother user experience and greater user satisfaction.

Technical feasibility

Medium. The implementation of a Chatbot would be simple because it does not require building the functionality from scratch given the existence of a multitude of APIs and services that fulfill this function. Its integration with the technology used in the project would be feasible, the only problem being maintenance, the possible cost of services and the security and privacy standards that it would have to meet.

5.3 Code repository

Repository's URL : isabelzamarron/Armario-Virtual-TFG

BIBLIOGRAFÍA

[1] «La industria de la moda está acabando con el planeta: las consecuencias de la moda rápida,» [En línea]. Available: https://www.nationalgeographic.com.es/mundo-ng/industria-moda-esta-acabando-planeta-consecuencias-moda-rapida_18586 [Último acceso: 13 05 2025]

[2] La moda como medio de expresión | TALLER. [En línea]. Available: https://taller-fdp.com/la-moda-como-medio-de-expresion/ [Último acceso: 5 05 2025].

[3] Greenpeace UK. How fast fashion fuels climate change, pollution and violence. [En línea]. Available: https://www.greenpeace.org/international/story/62308/how-fast-fashion-fuels-climate-change-plastic-pollution-and-violence/ [Último acceso: 19 05 2025]

[4] Greenpeace México, «Fast fashion: de tu armario al vertedero - Greenpeace México,» [En línea]. Available: https://www.greenpeace.org/mexico/blog/9514/fast-fashion/. [Último acceso: 5 05 2025]

[5] IBM. ¿Qué es una base de datos relacionales? | IBM. [En línea]. Available: https://www.ibm.com/es-es/topics/relational-databases [Último acceso: 19 05 2025]

[6] «HTML,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/HTML. [Último acceso: 13 05 2025].

[7] «Bootstrap,» [En línea]. Available: https://getbootstrap.com/. [Último acceso: 13 05 2025].

[8] Tailwind CSS. (s.f.). Sidebar Navigation. [En línea]. Available: https://tailwindcss.com/plus/ui-blocks/application-ui/navigation/sidebar-navigation [Último acceso: 19 05 2025]

[9] «CSS,» MDN Web Docs, [En línea]. Available: https://developer.mozilla.org/en-US/docs/Web/CSS. [Último acceso: 13 05 2025].

[10] «Express.js - Node.js web application framework,» [En línea]. Available: https://expressjs.com/es/. [Último acceso: 16 05 2025].

[11] «Node.js — Sobre Node.js®,» [En línea]. Available: https://nodejs.org/es/. [Último acceso: 13 05 2025].

[12] Yevhenii Herasymenko, «Bulletproof node.js project architecture ,» Medium, [En línea]. Available: https://dev.to/santypk4/bulletproof-node-js-project-architecture-4epf. [Último acceso: 16 05 2025].

[13] «XAMPP,» Apache Friends, [En línea]. Available: https://www.apachefriends.org/index.html. [Último acceso: 16 05 2025].

[14] «El tutorial de Python — documentación de Python - 3.13.3,» [En línea]. Available: https://docs.python.org/es/3/tutorial/. [Último acceso: 16 05 2025].

[15] «rembg,» PyPI, [En línea]. Available: https://pypi.org/project/rembg/. [Último acceso: 16 05 2025].

[16] «Acerca de GitHub y Git - Documentación de GitHub,» GitHub Docs, [En línea]. Available: https://docs.github.com/es/get-started/. [Último acceso: 16 05 2025].

[17] Carlos García, «Cómo iniciar un servidor con Node JS y Express JS | by Carlos García | Medium,» [En línea]. Available: https://medium.com/@carlosgarcia3535/c%C3%B3mo-iniciar-un-servidor-con-node-js-y-express-js-460b6fd672e6. [Último acceso: 16 05 2025]

[18] «interact.js,» [En línea]. Available: https://interactjs.io/. [Último acceso: 16 05 2025].

[19] «html2canvas,» npm, [En línea]. Available: https://www.npmjs.com/package/html2canvas. [Último acceso: 16 05 2025].

[20] «Widget de tiempo adaptable gratuito para el sitio web,» [En línea]. Available: Widget de tiempo adaptable gratuito para el sitio web [Último acceso: 13 05 2025].

[21] «GetWardrobe - Your Digital Closet,» [En línea]. Available: https://getwardrobe.com/ [Último acceso: 16 05 2025].

[22] «Acloset - AI Fashion Assistant,» [En línea]. Available: https://www.acloset.app/ [Último acceso: 16 05 2025].

[23] «Whering - Sustainable Fashion App,» [En línea]. Available: https://www.whering.co.uk/ [Último acceso: 16 05 2025].

APÉNDICES

Apéndice A - Casos de uso

Este apéndice está dedicado a la descripción de los actores y la especificación de requisitos del proyecto.

Actores

A continuación, se describen los actores de la aplicación:

- **Usuario no registrado**: Actor que no tiene acceso a ninguna función de la aplicación, necesita realizar el registro para obtener una cuenta e iniciar sesión.
- **Usuario registrado:** Actor que ha realizado el registro previamente en la aplicación, posee una cuenta con la que iniciando sesión obtiene acceso a todas las funciones de la aplicación, como son gestionar las prendas de su armario virtual, crear outfits mediante la combinación de prendas y poder compartirlos con la comunidad e interaccionar con las publicaciones de otros usuarios.
- **Administrador:** Actor encargado de la gestión de las cuentas de los usuarios registrados.

Diagramas de caso de uso

A continuación, se detallan los diagramas de casos de uso de cada actor con el objetivo de mostrar su rol en la aplicación.

Diagrama de usuario no registrado

Figura 46- Diagrama de casos de uso de usuario no registrado

Diagrama de usuario registrado

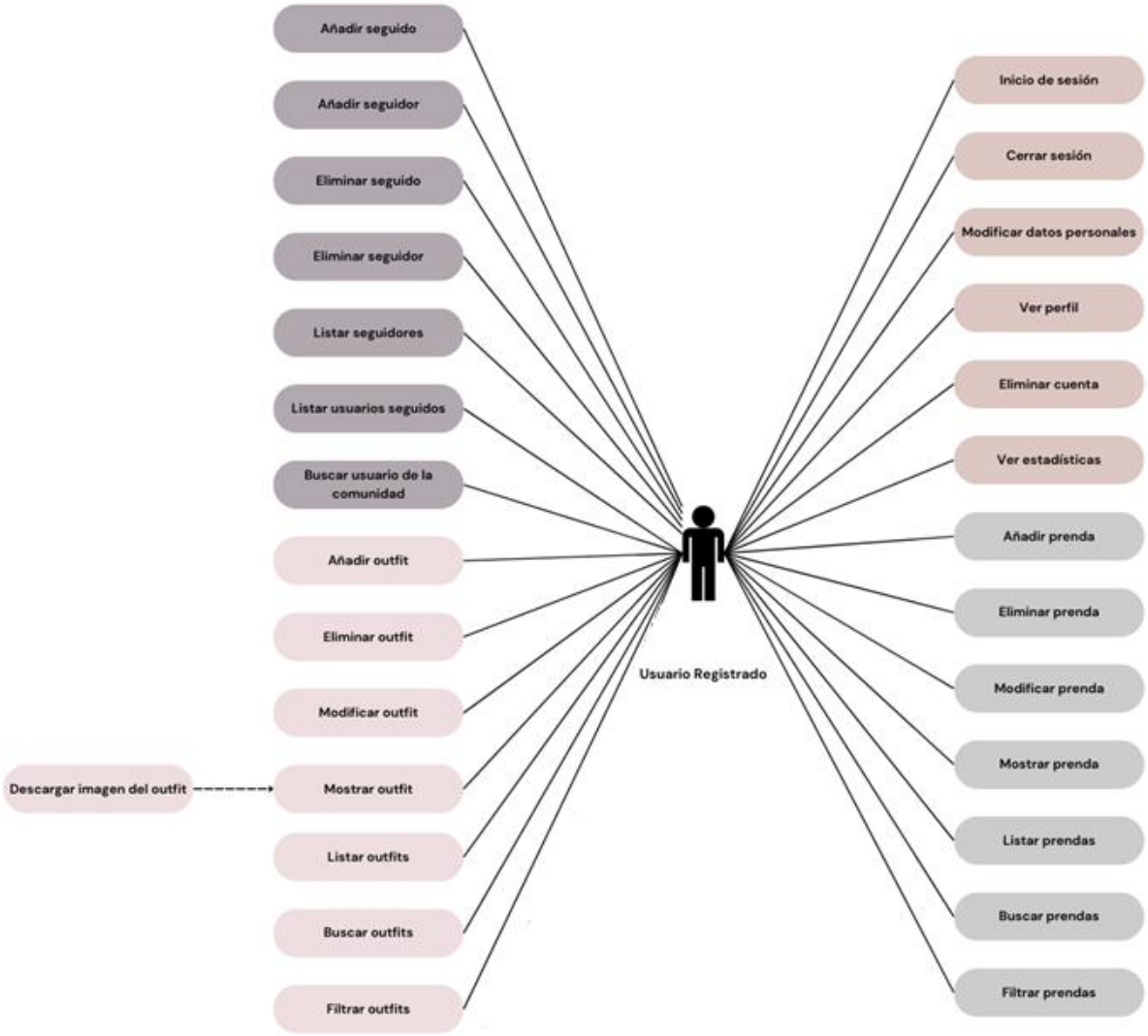


Figura 47- Diagrama de casos de uso de usuario registrado

Diagrama de administrador

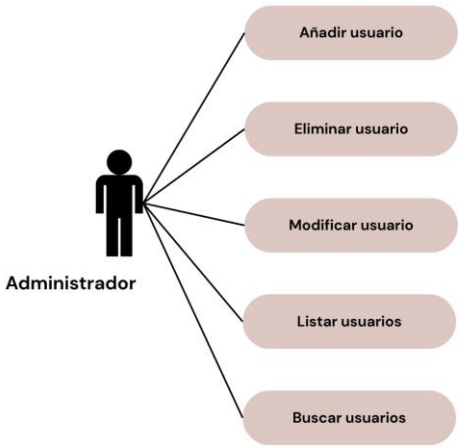


Figura 48- Diagrama de casos de uso de administrador

Especificación de requisitos

En esta sección se detallan los requisitos definidos para la aplicación. Para una mejor comprensión se ha optado por una división en módulos:

1- Modulo gestión del administrador
2- Modulo gestión de usuarios
3- Modulo prendas
4- Modulo outfits
5- Modulo comunidad

Módulo de gestión del administrador

Los requisitos que conforman este módulo son:

- Añadir usuario
- Eliminar usuario
- Modificar usuario
- Listar usuarios
- Buscar usuarios

Requisito	Añadir usuario	
Identificador	1.1	
Prioridad	Alta	
Precondición	Un usuario debe haber iniciado sesión y tener rol de administrador.	
Descripción	El administrador puede añadir un nuevo usuario al sistema.	
Entrada	Datos personales (nombre, apellidos, nombre de usuario, correo electrónico, contraseña), preferencias de estilo, rango de edad, foto.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario administrador pulsa sobre el botón de añadir usuario.
	2	Se muestra un modal con un formulario que consta de varios pasos.
	3	El usuario administrador completa los campos de cada correspondiente paso.
	4	El sistema valida que la información introducida es correcta.
	5	Se añade el nuevo usuario al sistema.

Postcondición	El usuario queda registrado en la aplicación.	
Excepciones	Paso	Acción
	4	Se muestra un mensaje de error si no se rellenan todos los campos.
	4	Se muestra un mensaje de error si la contraseña no cumple los requisitos.
	4	Se muestra un mensaje de error si el correo no cumple los requisitos.
	4	Se muestra un mensaje de error si el nombre de usuario introducido ya está activo.
	4	Se muestra un mensaje de error si el correo electrónico introducido ya está activo.
Comentarios	NA	
Actores	Administrador	

Tabla 1- Añadir usuario

Requisito	Eliminar usuario
Identificador	1.2
Prioridad	Media
Precondición	Un usuario debe haber iniciado sesión y tener rol de administrador.

Descripción	El administrador puede eliminar un usuario del sistema.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Mensaje de usuario eliminado	
Secuencia normal	Paso	Acción
	1	El usuario administrador pulsa sobre el botón de "eliminar" del usuario correspondiente.
	2	Se muestra un modal para confirmar la eliminación, el usuario administrador pulsa "eliminar".
	3	El sistema elimina el usuario.
Postcondición	El usuario ya no posee una cuenta en la aplicación.	
Excepciones	Paso	Acción
	2	El usuario debe confirmar que quiere realizar la acción.
Comentarios	NA	
Actores	Administrador	

Tabla 2- Eliminar usuario

Requisito	Modificar usuario
Identificador	1.3
Prioridad	Media

Precondición	Un usuario debe haber iniciado sesión y tener rol de administrador.	
Descripción	El administrador puede modificar los datos de un usuario del sistema.	
Entrada	Datos personales (nombre, apellidos, nombre de usuario, correo electrónico), preferencias de estilo, foto.	
Salida	Mensaje de datos de usuario modificados con éxito	
Secuencia normal	Paso	Acción
	1	El usuario administrador pulsa sobre el botón de "Modificar" del usuario correspondiente.
	2	El usuario administrador modifica los campos a cambiar y pulsa "Guardar cambios".
	3	El sistema valida la información introducida.
	4	Se modifica la información del usuario del sistema.
Postcondición	Se muestra actualiza la información del usuario actualizado con los cambios correspondientes.	
Excepciones	Paso	Acción
	3	Se muestra un mensaje de error si no se rellenan todos los campos.

	3	Se muestra un mensaje de error si el nombre de usuario introducido ya está activo.
	3	Se muestra un mensaje de error si el correo electrónico introducido ya está activo.
	3	Se muestra un mensaje de error si el correo no cumple los requisitos.
Comentarios	NA	
Actores	Administrador	

Tabla 3- Modificar usuario

Requisito	Listar usuarios	
Identificador	1.4	
Prioridad	Media	
Precondición	Un usuario debe haber iniciado sesión y tener rol de administrador.	
Descripción	El administrador puede visualizar los usuarios registrados en el sistema.	
Entrada	NA	
Salida	Listado de usuarios.	
Secuencia normal	Paso	Acción

	1	El usuario accede a su menú de inicio y puede navegar por las distintas páginas y visualizar el listado de usuarios
Postcondición	Se muestra una lista con los usuarios de la aplicación.	
Excepciones	Paso	Acción
	1	No existe ningún usuario registrado.
Comentarios	NA	
Actores	Administrador	

Tabla 4- Listar usuarios

Requisito	Buscar usuarios
Identificador	1.5
Prioridad	Baja
Precondición	Un usuario debe haber iniciado sesión y tener rol de administrador.
Descripción	El administrador puede buscar usuarios ingresando uno o varios criterios.
Entrada	Nombre, apellido, correo electrónico.
Salida	Usuario o lista de usuarios que coincidan con los criterios de búsqueda.

Secuencia normal	Paso	Acción
	1	El usuario accede a su menú de inicio e ingresa unos criterios de búsqueda.
Postcondición	Se muestra un usuario o lista de usuarios de la aplicación que cumplan los criterios de búsqueda introducidos.	
Excepciones	Paso	Acción
	1	No existe ningún usuario registrado que cumpla los criterios de búsqueda.
Comentarios	NA	
Actores	Administrador	

Tabla 5- Buscar usuarios

Módulo de gestión de usuarios

Los requisitos que conforman este módulo son:

- Registro de usuario
- Inicio de sesión
- Cerrar sesión
- Modificar datos personales
- Ver perfil
- Eliminar cuenta
- Ver estadísticas

-

Requisito	Registro de usuario	
Identificador	2.1	
Prioridad	Alta	
Precondición	El usuario no posee una cuenta en la aplicación.	
Descripción	Una persona que utiliza la aplicación por primera vez debe crearse una cuenta para disponer de todas las funcionalidades.	
Entrada	Datos personales (nombre, apellidos, nombre de usuario, correo electrónico, contraseña), preferencias de estilo, rango de edad, foto.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	En el menú principal se muestra un enlace que abre un modal con un formulario que consta de distintos pasos.
	2	El usuario completa todos los campos de cada paso y pulsa "Crear".
	3	El sistema valida la información introducida por el usuario.
	4	Se redirige al usuario a su menú de inicio
Postcondición	El usuario posee una cuenta en la aplicación y está autenticado.	

Excepciones	Paso	Acción
	3	Se muestra un mensaje de error si no se rellenan todos los campos.
	3	Se muestra un mensaje de error si la contraseña no cumple los requisitos.
	3	Se muestra un mensaje de error si el correo no cumple los requisitos.
	3	Se muestra un mensaje de error si el nombre de usuario introducido ya está activo.
	3	Se muestra un mensaje de error si el correo electrónico introducido ya está activo.
Comentarios	NA	
Actores	Usuario no registrado	

Tabla 6- Registro de usuario

Requisito	Inicio de sesión
Identificador	2.2
Prioridad	Alta
Precondición	El usuario debe haberse registrado en la aplicación previamente.

Descripción	Un usuario registrado puede acceder a su cuenta usando sus credenciales.	
Entrada	Correo electrónico, contraseña.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario se encuentra en el menú de inicio de la aplicación.
	2	El usuario rellena los campos correo electrónico y contraseña del formulario y pulsa "Acceder"
	3	El sistema verifica la información introducida por el usuario.
	4	En caso de éxito se redirige al usuario a su menú de inicio.
Postcondición	El usuario se encuentra autenticado en la aplicación	
Excepciones	Paso	Acción
	3	El correo o la contraseña introducidos no son correctos.
	3	La cuenta se encuentra desactivada.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 7- Inicio de sesión

Requisito	Cerrar sesión	
Identificador	2.3	
Prioridad	Alta	
Precondición	El usuario debe haber iniciado sesión previamente.	
Descripción	Un usuario que ha iniciado sesión puede desconectarse de la aplicación.	
Entrada	NA	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario desde su menú principal pulsa en el botón de cerrar sesión
	2	El sistema cierra la sesión del usuario.
	3	El sistema redirige al usuario a la pantalla inicial
Postcondición	El usuario ya no se encuentra autenticado en el sistema.	
Excepciones	Paso	Acción
	2	Se muestra un mensaje de error en caso de no poder cerrar sesión.

Comentarios	NA
Actores	Usuario registrado

Tabla 8- Cerrar sesión

Requisito	Modificar datos personales	
Identificador	2.4	
Prioridad	Media	
Precondición	El usuario debe estar registrado y haber iniciado sesión	
Descripción	El usuario puede actualizar la información de su perfil	
Entrada	Datos personales, preferencias de estilo, foto	
Salida	Mensaje de datos de usuario modificados con éxito	
Secuencia normal	Paso	Acción
	1	El usuario accede a la página de editar perfil
	2	El usuario modifica los campos que necesite cambiar
	3	El sistema valida que la nueva información introducida
	4	El sistema muestra los datos del perfil actualizados

Postcondición	Se muestra actualiza la información del perfil del usuario actualizado con los cambios correspondientes.	
Excepciones	Paso	Acción
	2	El usuario deja algún campo vacío
	3	Se ha introducido un correo electrónico que ya posee otro usuario
	3	Se ha introducido un nombre de usuario que ya posee otro usuario
Comentarios	NA	
Actores	Usuario registrado	

Tabla 9- Modificar datos personales

Requisito	Ver perfil
Identificador	2.5
Prioridad	Media
Precondición	El usuario debe estar registrado y haber iniciado sesión.
Descripción	El usuario puede visualizar su información personal.
Entrada	NA
Salida	Información de usuario.

Secuencia normal	Paso	Acción
	1	El usuario accede "Editar perfil" y pulsa el botón "eliminar cuenta" en la parte inferior de la pagina
	2	El usuario confirma que quiere eliminar su cuenta
	3	El sistema desactiva el usuario y se redirige al menú inicial
Postcondición	El usuario ya no cuenta con una cuenta en la aplicación	
Excepciones	Paso	Acción
	2	El usuario debe confirmar que quiere realizar la acción
Comentarios	NA	
Actores	Usuario registrado	

Tabla 10- Ver perfil

Requisito	Eliminar cuenta
Identificador	2.6
Prioridad	Media
Precondición	El usuario debe estar registrado y haber iniciado sesión.

Descripción	El usuario puede eliminar su cuenta si no desea seguir utilizando la aplicación.	
Entrada	NA	
Salida	Mensaje de cuenta eliminada con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario accede a "Editar perfil" y pulsa el botón "Eliminar cuenta" en la parte inferior de la página.
	2	El usuario confirma que quiere eliminar su cuenta.
	3	El sistema desactiva el usuario y se redirige al menú inicial.
Postcondición	El usuario ya no cuenta con una cuenta en la aplicación	
Excepciones	Paso	Acción
	2	El usuario debe confirmar que quiere realizar la acción.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 11- Eliminar cuenta

Requisito	Ver estadísticas
Identificador	2.7

Prioridad	Media	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	El usuario puede eliminar su cuenta si no desea seguir utilizando la aplicación.	
Entrada	NA	
Salida	Mensaje de cuenta eliminada con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario accede a "Editar perfil" y pulsa el botón "Eliminar cuenta" en la parte inferior de la página.
Postcondición	El usuario ya no cuenta con una cuenta en la aplicación	
Excepciones	Paso	Acción
	2	El usuario debe confirmar que quiere realizar la acción.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 12 - Ver estadísticas

Módulo de prendas

Los requisitos que conforman este módulo son:

- Añadir prenda
- Eliminar prenda

- Modificar prenda
- Mostrar prenda
- Listar prendas
- Buscar prendas
- Filtrar prendas

Requisito	Añadir prenda	
Identificador	3.1	
Prioridad	Alta	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	El usuario puede añadir una nueva prenda a su armario virtual.	
Entrada	Foto, categoría, tipo, parámetros (color, estación, estilo).	
Salida	Mensaje de prenda creada con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de prendas y pulsa el botón "+ Añadir prenda"
	2	El usuario rellena los campos correspondientes.
	4	El usuario pulsa el botón de guardar.
	5	El sistema añade la prenda al sistema.
Postcondición	La prenda ha sido creada y añadida a la lista de prendas.	
Excepciones	Paso	Acción

	2	Se muestra un mensaje de error si no se han rellenado todos los campos.
	2	Se muestra un mensaje de error en el caso de que se haya introducido un archivo muy pesado o de otro formato distinto a una imagen.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 13- Añadir prenda

Requisito	Eliminar prenda	
Identificador	3.2	
Prioridad	Alta	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente una prenda.	
Descripción	Los usuarios pueden eliminar las prendas de su armario virtual.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Mensaje de prenda eliminada con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre la prenda a eliminar, una vez se muestran los detalles, pulsa sobre el botón de "Eliminar".

	2	El usuario confirma que quiere eliminar la prenda.
	3	El sistema elimina la prenda.
Postcondición	La prenda es eliminada de la aplicación.	
Excepciones	Paso	Acción
	2	El usuario no confirma la eliminación de la prenda.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 14 - Eliminar prenda

Requisito	Modificar prenda	
Identificador	3.3	
Prioridad	Alta	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente una prenda.	
Descripción	El usuario puede actualizar la información de una prenda de su armario virtual.	
Entrada	Categoría, tipo, parámetros (color, estación, estilo), foto.	
Salida	Mensaje de prenda modificada con éxito.	
Secuencia normal	Paso	Acción

	1	El usuario pulsa sobre la prenda, una vez se muestran los detalles, pulsa sobre el campo que quiere modificar.
	2	El usuario realiza los cambios correspondientes en los parámetros desplegados del campo seleccionado.
	3	El sistema valida la información introducida.
	4	Se muestra la prenda con la información actualizada.
Postcondición	Se muestra la prenda actualizada con los cambios correspondientes	
Excepciones	Paso	Acción
	3	Se muestra un mensaje de error si no se han rellenado todos los campos
Comentarios	NA	
Actores	Usuario registrado	

Tabla 15 - Modificar prenda

Requisito	Mostrar prenda
Identificador	3.4
Prioridad	Alta

Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente una prenda.	
Descripción	El usuario puede ver la información de una prenda de su armario virtual.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Categoría, tipo, parámetros (color, estación, estilo) y foto.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre la prenda que quiere ver en detalle.
	2	El sistema muestra la información detallada de la prenda.
Postcondición	Se muestra la información detallada de la prenda.	
Excepciones	Paso	Acción
	2	No se pueden obtener los datos del sistema.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 16 - Mostrar prenda

Requisito	Filtrar prendas
Identificador	3.5

Prioridad	Baja	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente una prenda.	
Descripción	El usuario puede filtrar las prendas de su armario que se corresponden a su criterio de filtrado.	
Entrada	Parámetros de estación, estilo o color.	
Salida	Prendas filtradas	
Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de prendas y pulsa en el botón de "filtrado".
	2	Se abre un modal donde selecciona los parámetros por los que desea filtrar.
	3	El sistema muestra prendas que poseen esos parámetros seleccionados.
Postcondición	Se muestran las prendas que cumplen con los criterios de filtrado seleccionados por el usuario.	
Excepciones	Paso	Acción
	3	En caso de no encontrar prendas que cumplan esos parámetros se muestra mensaje de prendas no encontradas

Comentarios	NA
Actores	Usuario registrado

Tabla 17- Filtrar prendas

Requisito	Buscar prendas	
Identificador	3.6	
Prioridad	Baja	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	El usuario puede buscar prendas por sus características.	
Entrada	Parámetros de prenda.	
Salida	Prenda o lista de prendas que coincidan con los criterios de búsqueda.	
Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de prendas e ingresa unos criterios de búsqueda.
	2	El sistema muestra los resultados de la búsqueda correspondiente
Postcondición	Se muestra una prenda o lista de prendas del usuario que cumplan con los criterios de búsqueda introducidos.	
Excepciones	Paso	Acción

	2	Si no se encuentra ningún resultado se muestra un mensaje indicando que no se ha encontrado ninguna prenda.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 18 - Buscar prendas

-

Requisito	Listar prendas	
Identificador	3.7	
Prioridad	Alta	
Precondición	El usuario debe estar registrado y debe haber iniciado sesión.	
Descripción	El usuario puede visualizar las prendas de su armario virtual	
Entrada	NA	
Salida	Listado de prendas	
Secuencia normal	Paso	Acción
	1	El usuario accede a la página de su armario virtual de prendas y las puede visualizar.
Postcondición	Se muestra un listado con las prendas del armario del usuario.	
Excepciones	Paso	Acción

	1	No existe ninguna prenda creada.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 19 - Listar prendas

Módulo de outfits

Los requisitos que conforman este módulo son:

- Añadir outfit
- Eliminar outfit
- Modificar outfit
- Mostrar outfit
- Listar outfits
- Buscar outfits
- Filtrar outfits
- Descargar imagen del outfit

-

Requisito	Añadir outfit
Identificador	4.1
Prioridad	Alta
Precondición	El usuario debe estar registrado y haber iniciado sesión.
Descripción	Los usuarios pueden crear outfits con las prendas disponibles en su armario.
Entrada	Nombre, parámetros (evento, estación, tags), prendas.
Salida	Mensaje de outfit creado con éxito.

Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de outfits y pulsa el botón "+ Añadir outfit".
	2	El usuario rellena los campos correspondientes y selecciona las prendas que conforman el outfit.
	3	El usuario pulsa "Guardar" o "Guardar y publicar".
	4	El sistema añade el outfit al sistema.
Postcondición	El outfit ha sido creado y añadido a la lista de outfits con éxito.	
Excepciones	Paso	Acción
	2	Se muestra un mensaje de error si no se han rellenado todos los campos.
	2	Se muestra un error si no se ha seleccionado al menos una prenda.
Comentarios	Un usuario puede decidir hacer público su outfit y que sea visible para otros usuarios.	
Actores	Usuario registrado	

Tabla 20 - Añadir outfit

Requisito	Eliminar outfit
Identificador	4.2

Prioridad	Alta	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente un outfit.	
Descripción	Los usuarios pueden eliminar un outfit de su armario virtual.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Mensaje de outfit eliminado con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el outfit a eliminar, una vez se muestran los detalles, pulsa sobre el botón de "Opciones" y a continuación en el botón "Eliminar".
	2	El usuario confirma que quiere eliminar el outfit.
	3	El sistema elimina el outfit.
Postcondición	El outfit es eliminado de la aplicación.	
Excepciones	Paso	Acción
	2	El usuario no confirma la eliminación del outfit.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 21 - Eliminar outfit

Requisito	Modificar outfit	
Identificador	4.3	
Prioridad	Alta	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente un outfit.	
Descripción	El usuario puede actualizar la información de un outfit de su armario virtual.	
Entrada	Nombre, parámetros (evento, estación, tags), prendas.	
Salida	Mensaje de prenda modificada con éxito.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el outfit, una vez se muestran los detalles, pulsa sobre el campo que quiere modificar.
	2	El usuario realiza los cambios correspondientes en los parámetros desplegados del campo seleccionado.
	3	El sistema valida la información introducida.
	4	Se muestra el outfit con la información actualizada.
Postcondición	Se muestra el outfit actualizado con los cambios correspondientes.	

Excepciones	Paso	Acción
	3	Se muestra un mensaje de error si no se han rellenado todos los usuarios.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 22 - Modificar outfit

Requisito	Mostrar outfit	
Identificador	4.4	
Prioridad	Alta	
Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente un outfit.	
Descripción	El usuario puede ver la información de un outfit de su armario virtual.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Nombre, parámetros (eventos, estaciones, tags), prendas.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el outfit que quiere ver en detalle.

	2	El sistema muestra la información detallada del outfit.
Postcondición	Se muestra la información detallada del outfit.	
Excepciones	Paso	Acción
	2	No se pueden obtener los datos del sistema.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 23- Mostrar outfit

Requisito	Filtrar outfits	
Identificador	4.5	
Prioridad	Baja	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	El usuario puede filtrar los outfits de su armario que se corresponden a su criterio de filtrado.	
Entrada	Parámetros evento, tags, estaciones.	
Salida	Outfits filtrados.	
Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de outfits y pulsa en el botón de “filtrado”.

	2	Se abre un modal donde selecciona los parámetros por los que desea filtrar.
	3	El sistema muestra outfits que poseen esos parámetros seleccionados.
Postcondición	Se muestran los outfits que cumplen con los criterios de filtrado seleccionados por el usuario.	
Excepciones	Paso	Acción
	3	En caso de no encontrar outfits que cumplan esos parámetros se muestra un mensaje de outfits no encontrados.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 24 - Filtrar outfits

Requisito	Buscar outfits
Identificador	4.6
Prioridad	Media
Precondición	El usuario debe estar registrado y haber iniciado sesión.
Descripción	El usuario puede buscar outfits por sus características.
Entrada	Parámetros de prenda y nombre.

Salida	Outfit o lista de outfits que coincidan con los criterios de búsqueda.	
Secuencia normal	Paso	Acción
	1	El usuario se dirige al armario de outfits.
	2	El usuario introduce en la barra de búsqueda su criterio de búsqueda.
	3	El sistema muestra los resultados de la búsqueda correspondiente.
Postcondición	El usuario puede ver los resultados que coincidan con sus criterios de búsqueda.	
Excepciones	Paso	Acción
	3	Si no se encuentra ningún resultado se muestra un mensaje indicando que no se ha encontrado ningún outfit.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 25 - Buscar outfits

Requisito	Listar outfits
Identificador	4.7

Prioridad	Alta	
Precondición	El usuario debe estar registrado y debe haber iniciado sesión.	
Descripción	El usuario puede visualizar los outfits de su armario virtual.	
Entrada	NA	
Salida	Listado de outfits.	
Secuencia normal	Paso	Acción
	1	El usuario accede a la página de su armario virtual de outfit y los puede visualizar.
Postcondición	Se muestra un listado con los outfits del armario del usuario.	
Excepciones	Paso	Acción
	1	No existe ningún outfit creado.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 26 - Listar outfits

Requisito	Descargar imagen del outfit
Identificador	4.8
Prioridad	Alta

Precondición	El usuario debe estar registrado, debe haber iniciado sesión y haber creado previamente un outfit.	
Descripción	Los usuarios pueden descargar la imagen de un outfit de su armario virtual.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Imagen del outfit.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el outfit deseado, una vez se muestran los detalles, pulsa sobre el botón de "Opciones" y a continuación en el botón "Descargar Imagen".
	2	El sistema descarga la imagen y la almacena en el dispositivo.
Postcondición	Imagen descargada correctamente y almacenada en el dispositivo del usuario.	
Excepciones	Paso	Acción
	2	Ha habido un problema y no se ha podido descargar la imagen del outfit.
Comentarios	NA	
Actores	Usuario registrado	

Módulo de funciones de comunidad

Los requisitos que conforman este módulo son:

- Añadir seguido
- Eliminar seguido/seguidor
- Añadir seguidor
- Listar usuarios seguidos
- Listar seguidores
- Buscar usuario de la comunidad
- Listar outfits públicos de usuarios seguidos

-

Requisito	Añadir seguido	
Identificador	5.1	
Prioridad	Media	
Precondición	El usuario debe estar registrado, haber iniciado sesión y el usuario a seguir debe existir.	
Descripción	El usuario puede seguir a otro usuario de la aplicación.	
Entrada	Selección de usuario desde la interfaz.	
Salida	Mensaje de usuario seguido con éxito	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre sobre el botón "Seguir" del usuario correspondiente.
	2	El sistema valida la información y actualiza el contador de seguidores

Postcondición	El usuario seguido aparece en la lista de usuarios seguidos y aumenta el contador de usuarios seguidos.	
Excepciones	Paso	Acción
	1	Ha habido un problema y no se ha podido añadir el seguido.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 27 - Añadir seguido

Requisito	Eliminar seguido/seguidor	
Identificador	5.2	
Prioridad	Media	
Precondición	El usuario debe estar registrado y haber iniciado sesión	
Descripción	El usuario puede dejar de seguir a un usuario o eliminar un seguidor de la lista,	
Entrada	Selección de usuario desde la interfaz.	
Salida	NA	
Secuencia normal	Paso	Acción
	1	El usuario se dirige a la lista de seguidores o seguidos y pulsa en el botón "X".

	2	Se abre un modal para confirmar la acción.
	3	El sistema elimina el seguidor/ seguido,
Postcondición	El usuario no aparece en la lista de seguidores /lista de seguidos.	
Excepciones	Paso	Acción
	2	El usuario no confirma la acción.
	3	Ha habido un problema y no se ha podido realizar la eliminación.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 28 - Eliminar seguido/seguidor

Requisito	Añadir seguidor
Identificador	5.3
Prioridad	Media
Precondición	El usuario debe estar registrado, haber iniciado sesión y deben existir tanto el usuario seguido como el seguidor.
Descripción	Los usuarios pueden ser seguidos por otros usuarios de la comunidad.
Entrada	NA
Salida	NA

Secuencia normal	Paso	Acción
	1	El sistema detecta que un usuario ha seguido a otro.
	2	El usuario que es seguido incrementa el número de seguidores
Postcondición	El usuario seguidor aparece en la lista de usuarios seguidores y aumenta el contador de usuarios seguidores.	
Excepciones	Paso	Acción
	1	Ha habido un problema y no se ha podido añadir el seguidor.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 29 - Añadir seguidor

Requisito	Listar usuarios seguidos
Identificador	5.4
Prioridad	Media
Precondición	El usuario debe estar registrado y haber iniciado sesión
Descripción	Los usuarios pueden visualizar los usuarios de la comunidad que siguen

Entrada	NA	
Salida	Listado de usuarios seguidos.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el botón "seguidos".
	2	Se despliega un modal con el listado de usuarios seguidos.
Postcondición	Se muestra una lista con los usuarios seguidos por un usuario de la aplicación.	
Excepciones	Paso	Acción
	2	En caso de no encontrarse ningún usuario que se mostrará un mensaje de "No sigues a nadie".
	2	Ha habido un problema y no se ha podido mostrar el listado de usuarios seguidos.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 30 - Listar usuarios seguidos

Requisito	Listar seguidores
Identificador	5.5
Prioridad	Media

Precondición	El usuario debe estar registrado y haber iniciado sesión	
Descripción	Los usuarios pueden visualizar los usuarios de la comunidad que les han comenzado a seguir.	
Entrada	NA	
Salida	Listado de usuarios seguidores.	
Secuencia normal	Paso	Acción
	1	El usuario pulsa sobre el botón "seguidores".
	2	Se despliega un modal con el listado de usuarios seguidores.
Postcondición	Se muestra una lista con los usuarios seguidores de usuario de la aplicación.	
Excepciones	Paso	Acción
	2	En caso de no encontrarse ningún usuario que se mostrará un mensaje de "No tienes seguidores".
	2	Ha habido un problema y no se ha podido mostrar el listado de usuarios seguidores.
Comentarios	NA	
Actores	Usuario registrado	

Tabla 31 - Listar seguidores

Requisito	Buscar usuario de la comunidad	
Identificador	5.6	
Prioridad	Media	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	Un usuario puede buscar otros usuarios de la comunidad ingresando uno o varios criterios.	
Entrada	Nombre, apellidos, nombre de usuario.	
Salida	Lista de usuarios que coinciden con los criterios de búsqueda.	
Secuencia normal	Paso	Acción
	1	El usuario desde el menú principal ingresa nombre o nombre de usuario deseado
	2	El sistema muestra los resultados a la búsqueda realizada
Postcondición	Se muestra un usuario o lista de usuarios de la aplicación que cumplan los criterios de búsqueda introducidos.	
Excepciones	Paso	Acción
	2	En caso de no encontrarse ningún usuario que corresponda a la búsqueda se mostrará un mensaje de " No se encontraron usuarios".
Comentarios	NA	

Actores	Usuario registrado

Tabla 32 - Buscar usuario de la comunidad

Requisito	Listar outfits públicos de usuarios seguidos	
Identificador	5.7	
Prioridad	Media	
Precondición	El usuario debe estar registrado y haber iniciado sesión.	
Descripción	Un usuario puede ver los outfits que han publicado los usuarios que sigue de la comunidad.	
Entrada	NA	
Salida	Lista de outfits públicos.	
Secuencia normal	Paso	Acción
	1	El usuario desde el menú principal visualiza los outfits.
Postcondición	Se muestra una lista de outfits publicados por los usuarios que seguidos.	
Excepciones	Paso	Acción
	2	Ha habido un problema y no se ha podido mostrar el listado de outfits.
Comentarios	NA	

Actores	Usuario registrado

Tabla 33- Listar outfits públicos de usuarios seguidos

Apéndice B - Manual de usuario

El objetivo de este manual es guiar al usuario a través de las distintas funcionalidades de la aplicación. A continuación, se detallan las funciones acompañadas de capturas de pantalla para un mayor entendimiento.

Inicio de sesión

La primera vista a la que se tiene acceso una vez se inicia la aplicación posee el formulario inicio de sesión junto con un link al modal donde se muestra el formulario de registro de usuario.

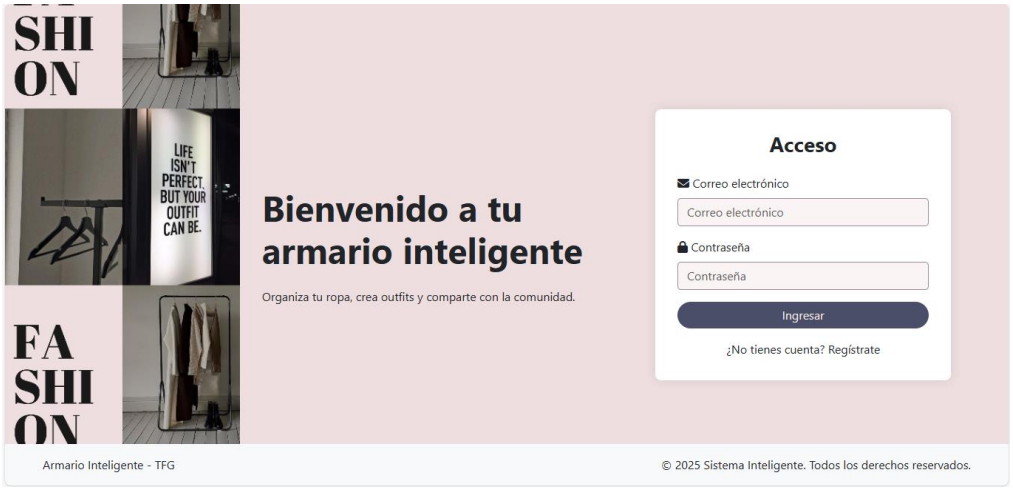


Figura 49- Vista menú inicial de la aplicación

En caso de introducir los campos de forma incorrecta o en caso de cuenta desactivada se mostrará un mensaje de aviso.

Registro de usuario

El modal de registro se divide en 3 pasos para mejorar la experiencia de usuario. Consta de un primer paso donde el usuario completa sus datos personales. Un segundo paso donde el usuario escoge cuáles son sus estilos preferidos (o con que estilos se siente

más identificado) y un último donde el usuario puede escoger su imagen de perfil y su rango de edad.

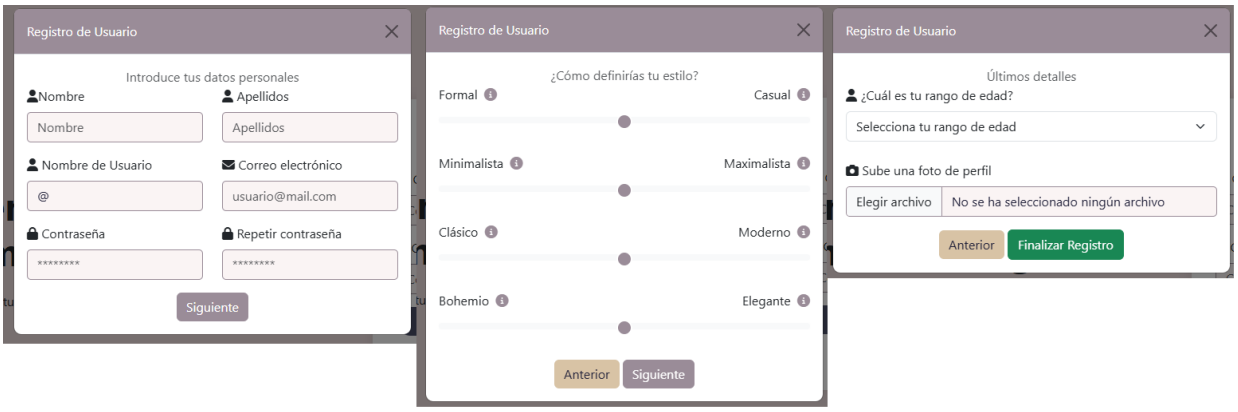


Figura 50- Modal registro usuario

Vista principal con sesión iniciada

Administrador

En caso de iniciar sesión como administrador se mostrará un menú que contiene un listado de usuarios. Desde este menú podrá añadir, buscar, eliminar, modificar usuarios y hacer administrador a otros usuarios.

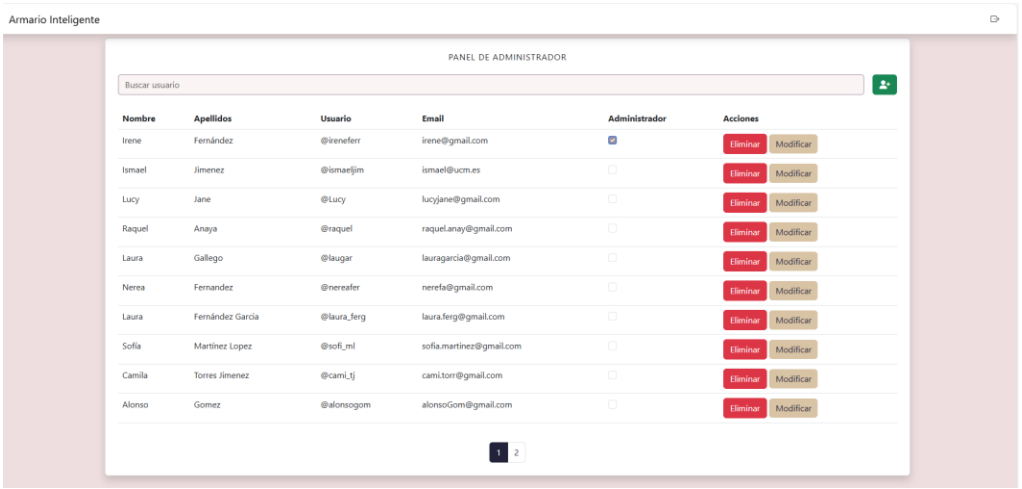


Figura 51- Vista del menú del administrador

Podrá buscar usuarios introduciendo texto sobre la barra de búsqueda.

En caso de pulsar el botón de añadir usuario aparecerá el modal de añadir usuario. Éste consistirá en un modal con un formulario similar al utilizado en el modal de registro en el que el usuario deberá completar los 3 pasos.

Figura 52- Modal de añadir usuario (administrador)

En caso de querer modificar, eliminar o cambiar el estado de administrador, el usuario administrador podrá realizar estas acciones pulsando a los botones pertenecientes al usuario deseado.

Figura 53- Modal de editar usuario (administrador)

Usuario común

El usuario cuando inicia sesión correctamente o se registra y accede por primera vez, es redirigido a su menú principal, desde el que tiene acceso a varias de las funcionalidades.

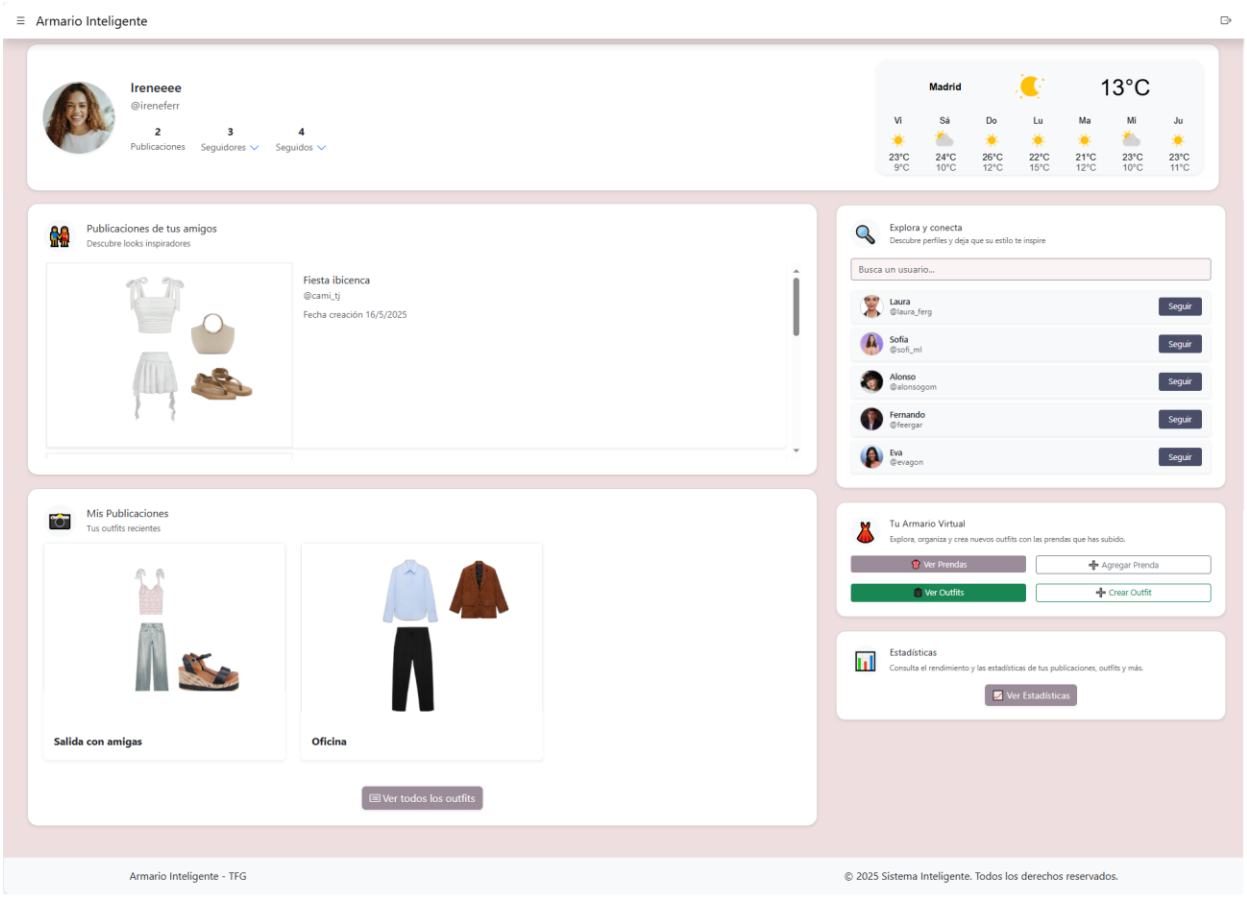


Figura 54- Vista menú principal de usuario

Modulo usuario

Para editar los datos de su cuenta de usuario, éste deberá dirigirse a la zona "Editar perfil". Desde esta interfaz el usuario podrá hacer los cambios que considere sobre su información, una vez rellenos los campos a modificar pulsará sobre "Guardar cambios".

En caso de querer desactivar la cuenta el usuario deberá pulsar sobre el botón inferior "Desactivar cuenta", y posteriormente confirmar la decisión.

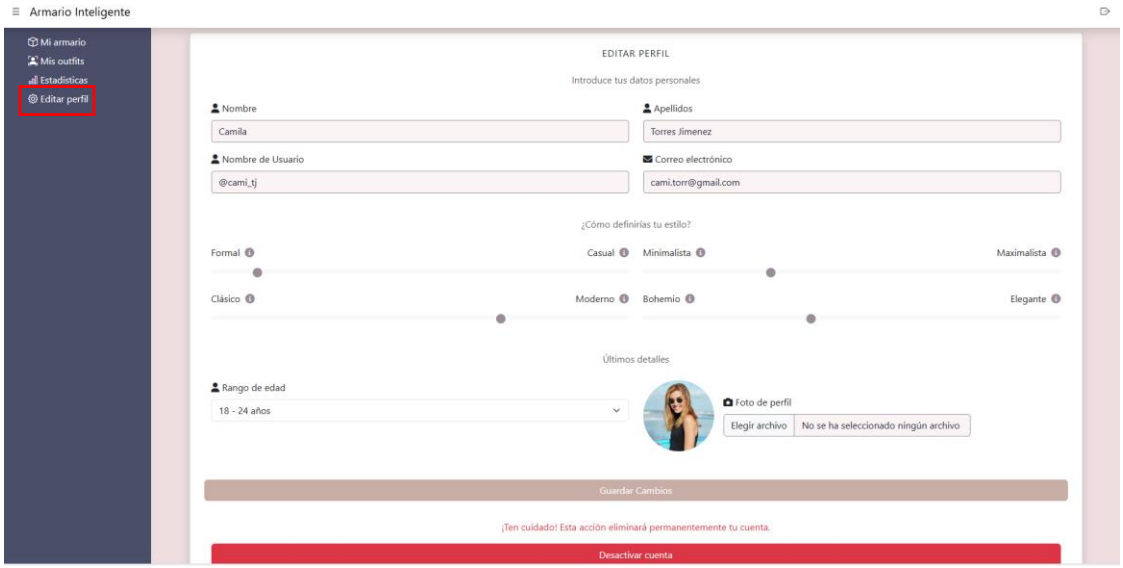


Figura 55- Vista de editar perfil de usuario

Para ver sus estadísticas el usuario se deberá desplazar a la zona "Estadísticas", en esta interfaz podrá obtener información relevante de su uso de la aplicación.

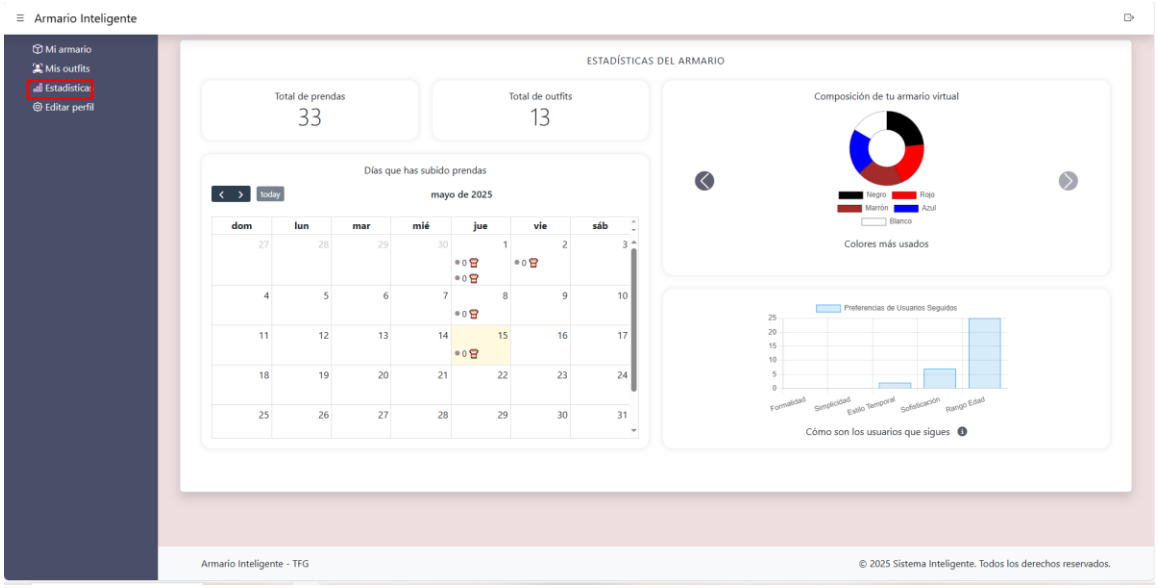


Figura 56- Vista de estadísticas de usuario

Módulo de prendas

Para poder utilizar las funciones del módulo de prendas el usuario tiene que dirigirse a la zona de "Mi armario".

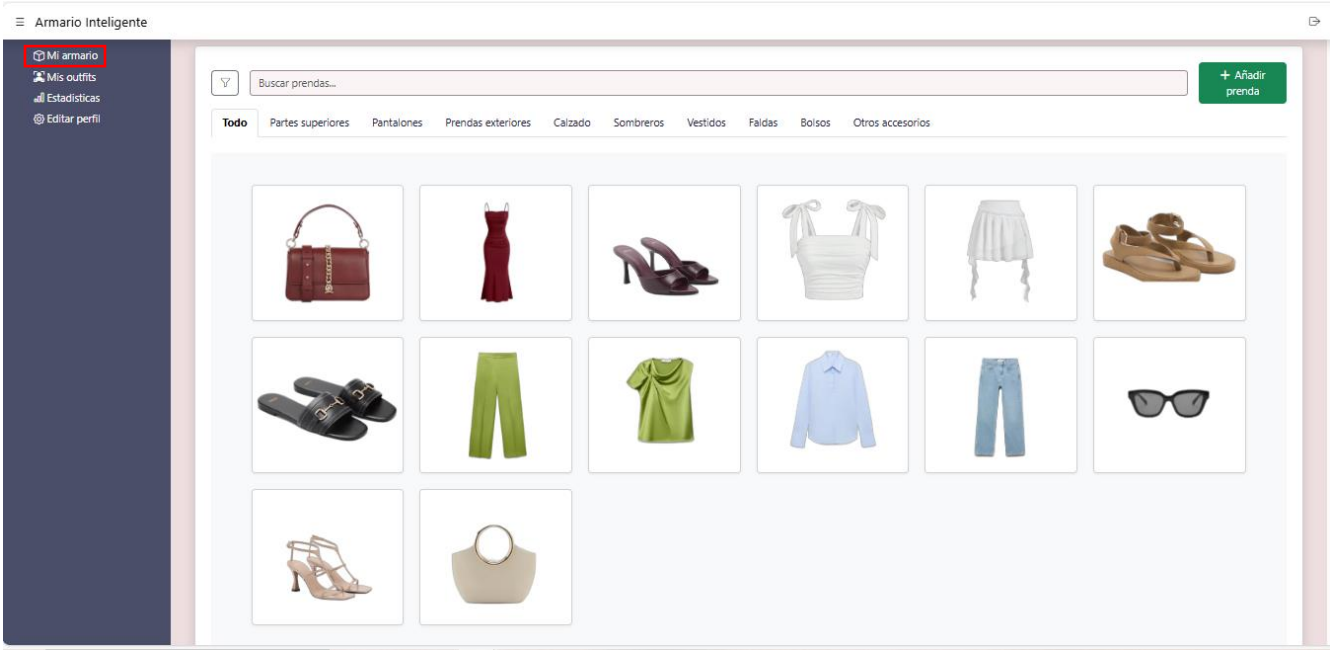


Figura 57- Vista armario de prendas

Desde esta interfaz el usuario podrá buscar prendas introduciendo texto en la barra de búsqueda y visualizar las prendas divididas por categorías y tipos.

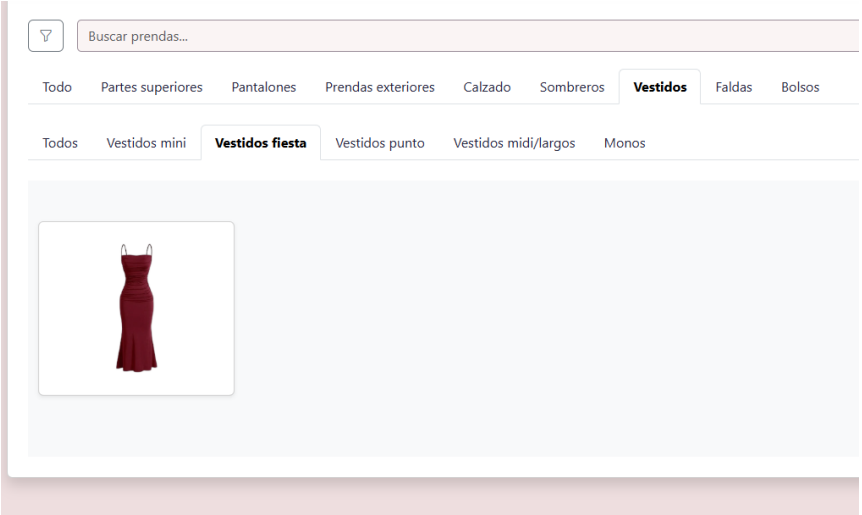


Figura 58- Vista armario de prendas por pestañas

Para poder visualizar una prenda con más detalle el usuario podrá pulsar sobre la prenda deseada y se mostrará un modal con la información de ésta. Desde éste se podrá eliminar pulsando sobre el botón situado en la esquina superior izquierda.

Figura 59- Modal detalles de prenda

En caso de querer modificar algún parámetro de la prenda, el usuario simplemente tendrá que desplegar la sección que desee y pulsar en "Guardar cambios".

Figura 60- Modificar parámetros prenda

Para añadir una prenda el usuario pulsará sobre el botón "+ Añadir prenda", éste le abrirá un modal donde deberá rellenar todos los campos.

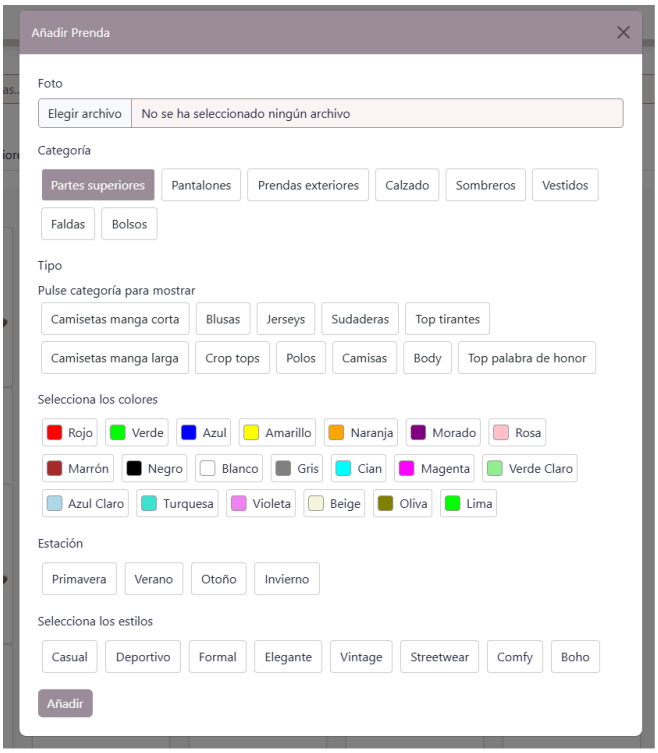


Figura 61- Modal de añadir prenda

Una vez finalizado el formulario se mostrará al usuario un spinner durante el tiempo que tarde la imagen introducida en procesarse.

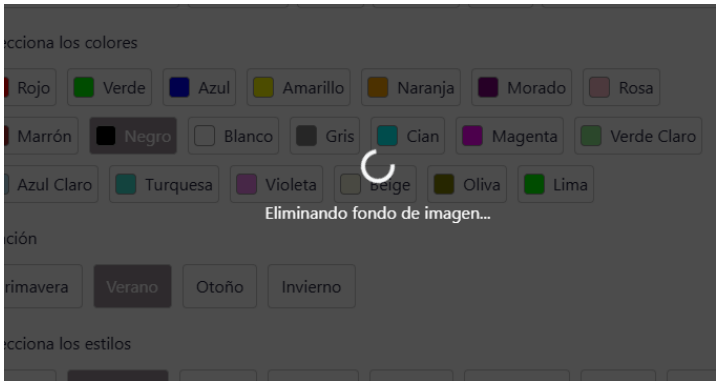


Figura 62- Spinner del proceso de eliminación de fondo

En caso de éxito en la acción se mostrará un mensaje.

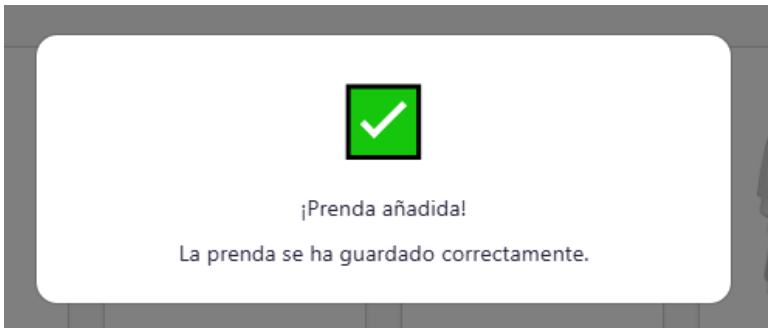


Figura 63- Mensaje de confirmación de prenda añadida

Para filtrar el usuario deberá pulsar sobre el icono de filtrado, y a continuación seleccionar los parámetros por los que desea filtrar.

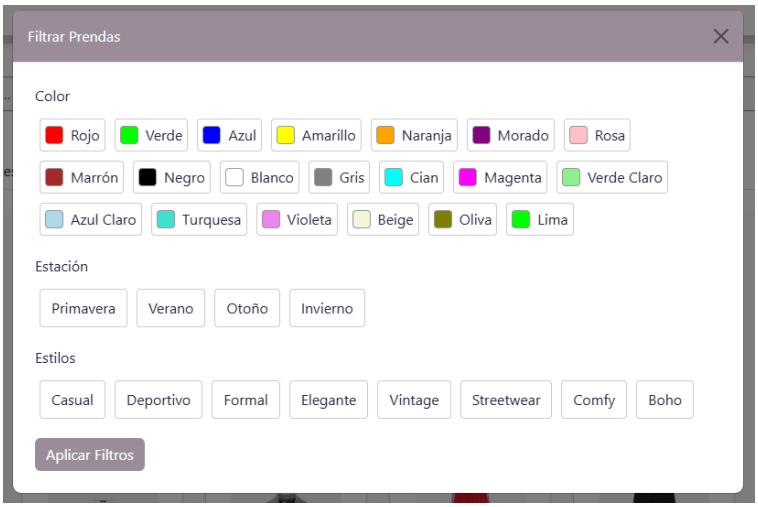


Figura 64- Modal de filtrar prendas

Modulo outfit

Para poder utilizar las funciones del módulo de prendas el usuario tiene que dirigirse a la zona de "Mis outfits".

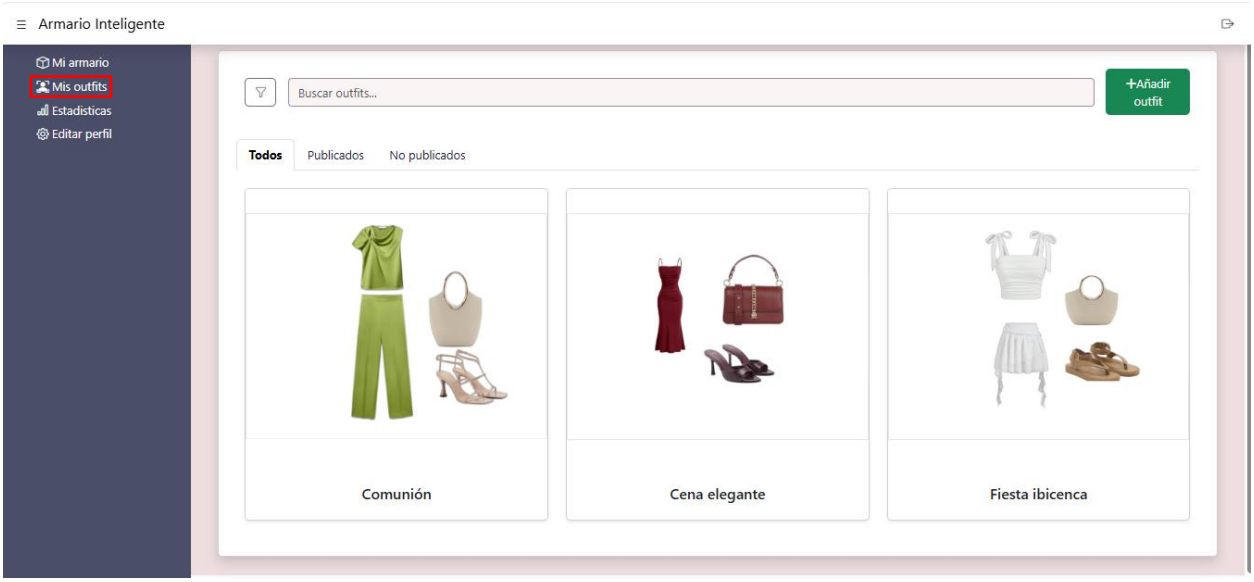


Figura 65- Vista armario de outfits

Desde esta interfaz el usuario podrá buscar outfits introduciendo texto en la barra de búsqueda y visualizar los outfits divididos por todos, outfits publicados y outfits no publicados.

Para poder visualizar un outfit con más detalle el usuario podrá pulsar sobre el outfit deseado y se mostrará un modal con sus detalles. Desde el botón de opciones situado en la esquina superior izquierda se podrá eliminar y descargar la imagen del outfit.

Figura 66- Modal de detalles de outfit

En caso de querer modificar algún parámetro del outfit, el usuario simplemente tendrá que desplegar la sección que desee y pulsar en "Guardar cambios". También podrá cambiar el estado de público a su deseo y el nombre del outfit.

Figura 67- Modificar parámetros de outfit

Para añadir un outfit el usuario pulsará sobre el botón "+ Añadir outfit", éste le abrirá un modal compuesto por dos pasos, donde en primer lugar deberá rellenar todos los campos.

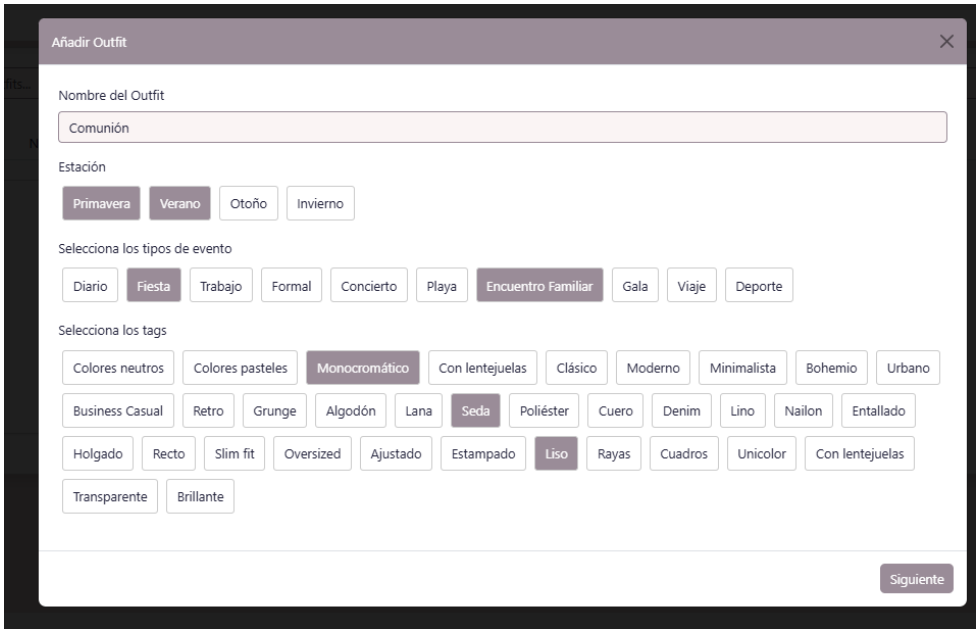


Figura 68- Modal de añadir outfit en paso 1

Y en segundo lugar podrá navegar por las distintas categorías seleccionando las prendas que desee.

Por último podrá decidir si mantenerlo privado o publicarlo para que lo puedan ver otros usuarios.

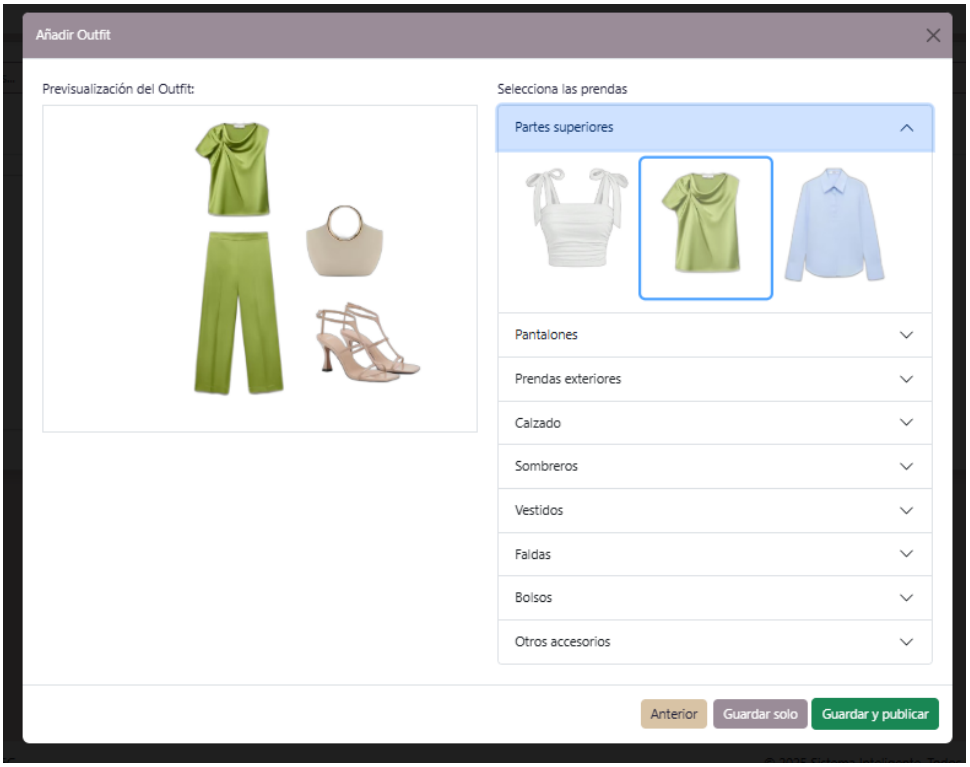


Figura 69- Modal de añadir outfit en paso 2

Para filtrar el usuario deberá pulsar sobre el icono de filtrado, y a continuación seleccionar los parámetros por los que desea filtrar.

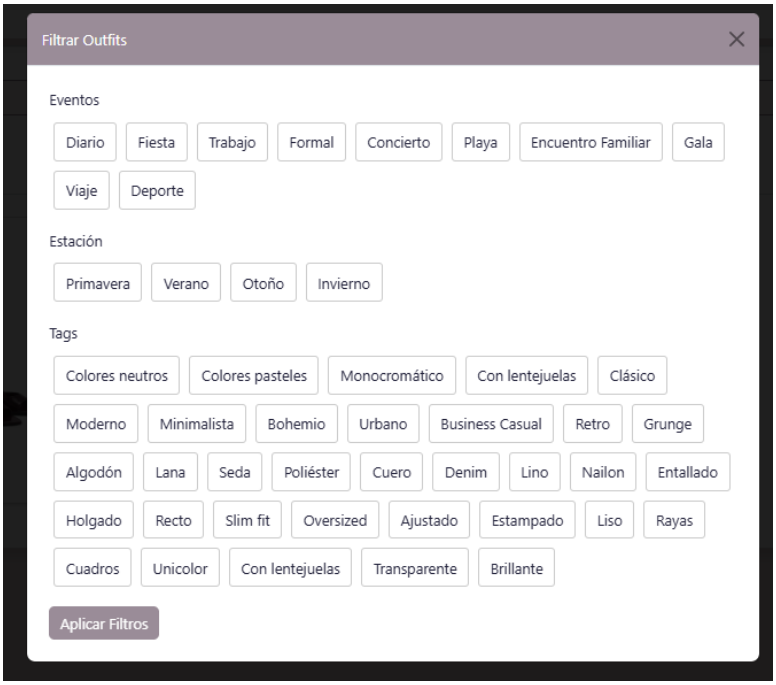


Figura 70- Modal de filtrar outfits

Modulo comunidad

El usuario tendrá acceso a las funciones de comunidad desde el propio menú de inicio. En la parte superior podrá observar un contador de outfits publicados, seguidores y seguidos.

Figura 71- Contadores del perfil de usuario

Una vez el usuario pulsa sobre la flecha se le desplegará un listado con sus seguidores y usuarios seguidos.

Figura 72- Modal con listados de usuarios seguidores y seguidos

En caso de no encontrarse ningún usuario se mostraría el listado vacío con un mensaje de aviso.

El usuario podrá añadir más usuarios mediante el siguiente apartado de sugerencias, donde también podrá buscar por usuarios específicos.

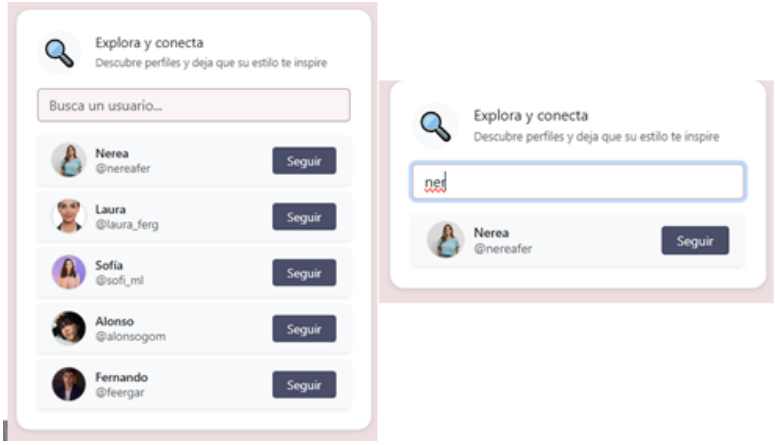


Figura 73- Sección de búsqueda de usuarios de la comunidad