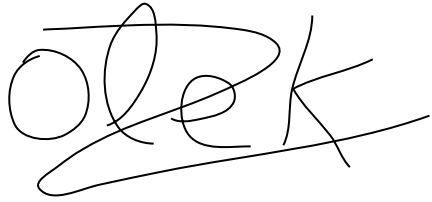


COMPRATHOR: Comparador personal de productos



Declaración de autoría:

Yo, Oleksandr Olefirenko, declaro la autoría del Trabajo Fin de Máster titulado “COMPRATHOR: Comparador personal de productos” y que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual. El material no original que figura en este trabajo ha sido atribuido a sus legítimos autores.

A handwritten signature in black ink, appearing to read 'olek' followed by a stylized 'K' and a long horizontal stroke.

Valencia, 26 de Febrero de 2024

Firmado: Oleksandr Olefirenko

Sinopsis

COMPRATHOR: Comparador personal de productos es un proyecto que consiste en el desarrollo de una plataforma Full Stack que tiene la finalidad de permitir a los usuarios comparar de forma sencilla los diferentes productos existentes con el objetivo de tomar mejores decisiones de compra. La plataforma proporcionará la capacidad de registrar comparativas de productos que estén relacionados con procesos de toma de decisiones de compra. En cada comparativa individual, los usuarios tendrán la posibilidad de ingresar información detallada acerca de los productos, incluyendo características específicas e imágenes.

La aplicación va a permitir a los usuarios finales la capacidad de comparar los productos que se encuentren dentro de cada comparativa haciendo uso de un sistema de puntuación.

Palabras clave

Comparador de productos, visualización en tiempo real, interacción móvil, decisiones de compra, facilidad de uso, comparativas, información asociada.

Índice

1. Objetivos específicos	6
2. Descripción general/Introducción	7
2.1. Backend	7
2.2. Frontend Web.....	9
2.3. App Móvil.....	11
3. Cronología	13
3.1. Análisis y planificación.....	13
3.2. Creación tablas en base de datos	13
3.3. Backend - Entidades y Repositorios.....	15
3.4. Backend - Modelo.....	15
3.7. Backend - Keycloak.....	15
3.8. Backend - Seguridad	16
3.9. Pruebas	16
3.10. Frontend - Proyecto y Estructura de Carpetas.....	17
3.11. Frontend - Integración de componentes	18
3.12. Frontend - Estilos y Diseño	18
3.13. Frontend - Navegación	18
3.14. Frontend - Comunicación con el Backend	18
3.15. Frontend - Ejecución de la aplicación	18
3.16. Frontend - Inicio de sesión.....	19
3.17. Frontend - Lista de Productos.....	20
3.18. Frontend - Comparativas	21
3.19. App Móvil - Proyecto y Estructura de Carpetas	23
3.20. App Móvil - Desarrollo con Flutter.....	24
3.21. App Móvil - Diseño interfaz de usuario	24

3.22.	App Móvil - Estilos	24
3.23.	App Móvil - Navegación	24
3.24.	App Móvil - Comunicación con el Backend	24
3.25.	App Móvil - Inicio de sesión	25
3.26.	App Móvil - Lista de productos.....	26
3.27.	App Móvil - Comparativas	26
4.	Detalle y Justificación.....	29
4.1.	Diagramas de flujo.....	29
4.2.	Backend	31
4.3.	Frontend	34
4.4.	App Móvil.....	35
5.	Trabajo futuro.....	37
6.	Conclusiones.....	38
7.	Bibliografía	39

Índice de ilustraciones

Ilustración 1 - Keycloak backend.....	15
Ilustración 2 - Postman pruebas.....	16
Ilustración 3 - Iniciar sesión con Keycloak.....	19
Ilustración 4 - Registrarse con Keycloak	19
Ilustración 5 - Integración de Keycloak en Frontend	20
Ilustración 6 - Lista de productos Frontend	20
Ilustración 7 - Visualización de comparativas Frontend	21
Ilustración 8 - Visualización de comparativas en tiempo real Frontend.....	22
Ilustración 9 - Añadir comparativa Frontend.....	22
Ilustración 10 - Función fetchProductos	24
Ilustración 11 - Clase general que conecta con Backend desde App móvil	25
Ilustración 12 - Configuración Keycloak App móvil	25
Ilustración 13 - Menú de navegación App móvil.....	26
Ilustración 14 - Lista de productos App móvil.....	26
Ilustración 15 - Comparativas de productos App Móvil	27
Ilustración 16 - Comparativas de productos en tiempo real App Móvil.....	27
Ilustración 17 - Añadir comparativa App móvil	28
Ilustración 18 - Diagrama casos de uso	29
Ilustración 19 - Diagrama de flujo.....	30
Ilustración 20 - Diagrama de base de datos.....	30
Ilustración 21 - Wireframes	31

1. Objetivos específicos

En el contexto actual, donde la diversidad de opciones de productos en el mercado ha llegado a un nivel considerado, es un desafío para los consumidores tomar decisiones informadas de compra. La aplicación “COMPRATHOR: Comparador personal de productos” es una respuesta a este problema, permitiendo a los usuarios comparar de forma sencilla y efectiva una gran variedad de productos disponibles. Se justifica en el requerimiento de hacer más simple el proceso de toma de decisiones de compra, optimizando de esta manera su experiencia y aumentando su nivel de confianza a la hora de elegir productos.

Tiene como objetivo principal brindar a los usuarios finales una solución eficiente y completa con la que puedan comparar productos. Mediante una estructura Full Stack, la plataforma proporcionará la capacidad a los usuarios de visualizar, registrar y comparar productos de forma intuitiva.

La aplicación ofrecerá funcionalidades como la posibilidad de poder registrar comparativas individuales, incluyendo información detallada sobre los productos, como pueden ser características e imágenes. Por otra parte, se va a utilizar un sistema de puntuaciones con el fin de destacar las diferencias existentes entre productos. Asimismo, también va a permitir la incorporación rápida de nuevos productos y la capacidad de visualizar en tiempo real resúmenes comparativos.

Por otra parte, también se van a implementar medidas de seguridad robustas con el objetivo de proteger la información asociada a cada usuario y, de esta manera, poder garantizar la privacidad de los datos que se almacenan en la plataforma. Asimismo, se van a realizar pruebas exhaustivas de la plataforma con el fin de poder garantizar la estabilidad y el rendimiento de la misma. Fomentar la participación de los usuarios es otro objetivo que cumplir, se les proporcionará la posibilidad de realizar comentarios y reseñas para establecer una comunidad colaborativa y comprometida.

Estos objetivos específicos se han planificado con el fin de abordar de forma integral los desafíos que se han identificado en el contexto actual de saturación de opciones de productos. De esta manera, la aplicación será considerada una herramienta útil y valiosa para los usuarios.

2. Descripción general/Introducción

En el diseño y desarrollo de la aplicación “COMPRATHOR: Comparador personal de productos”, se ha realizado una evaluación de las tecnologías actuales con el fin de seleccionar las que proporcionen una mayor escalabilidad, eficiencia y una mejor experiencia para el usuario. En este apartado se van a presentar las tecnologías seleccionadas para cada componente de la aplicación y las razones por las cuales se ha realizado dicha selección.

2.1. Backend

2.1.1. Java

Se trata de una plataforma informática de lenguaje de programación que fue fundada en 1995 por Sun Microsystems. Se caracteriza por ser una plataforma fiable donde se elaboran muchas aplicaciones y servicios. En ella se reducen costos, se acortan los plazos de desarrollo y se mejoran los servicios de las aplicaciones.

Se utiliza para codificar aplicaciones web. Se trata de una opción ampliamente utilizada por los desarrolladores durante más de dos décadas, con millones de aplicaciones Java existentes en el presente. Consiste en un lenguaje seguro y rápido con el fin de codificar todo de manera confiable, desde software empresarial y aplicaciones móviles hasta soluciones de macrodatos y tecnologías del servidor.

Algunas aplicaciones frecuentes de Java abarcan:

- Computación en la nube

Java es conocido como WORA (escribir una vez y ejecutar en cualquier lugar), por lo cual es buena elección para aplicaciones que se encuentren descentralizadas en la nube.

- Videojuegos

Juegos para computadoras y móviles son creados con Java. Asimismo, con Java también se puede incorporar la realidad virtual o el machine learning en el ámbito de los videojuegos.

- Macrodatos

La tecnología de java también se emplea en motores de procesamiento de datos capaces de manejar conjuntos de datos complejos y también grandes volúmenes de información en tiempo real.

Java dispone de muchos recursos de aprendizaje calificados de alta calidad, esto se debe a su antigüedad. Los libros completos, la documentación detallada y los cursos realizados son una ayuda para aquellos desarrolladores que se encuentren en la curva de aprendizaje. Por otra parte, Java ofrece muchas funciones y bibliotecas integradas, por lo que el desarrollador no tiene la necesidad de crear cada función, puede utilizar las ya existentes.

Java puede ejecutarse en cualquier plataforma subyacente, como Linux, Windows, iOS o Android, sin la necesidad de reescribir el código. Asimismo, proporciona diversas herramientas para la depuración, las pruebas, la implementación y la administración de cambios. También posee unos altos niveles de seguridad ya que, al descargar código externo, este no puede infectar el sistema ni puede leer o escribir archivos del disco duro.

2.1.2. Framework Spring-Boot

Java Spring Framework consiste en un marco de trabajo empresarial de código abierto ampliamente adoptado, este se usa para la creación de aplicaciones autónomas de producción que son ejecutadas en una máquina virtual JAVA (JVM).

Por otra parte, Java Spring Boot es una herramienta complementaria que simplifica y acelera el desarrollo de microservicios y aplicaciones web que hagan uso de Spring Framework. Esta se caracteriza por tener una configuración automática, la capacidad de creación de aplicaciones autónomas y un enfoque de configuración sencillo.

Se va a utilizar con el fin de abordar el tiempo y los conocimientos necesarios para la configuración, instalación y el despliegue de las aplicaciones Spring.

Esto lo vamos a conseguir gracias a tres puntos:

- **Configuración automática**

Esto se puede definir como que las aplicaciones arrancan con las dependencias predefinidas sin la necesidad de ninguna configuración manual. De esta manera, los paquetes de la base de Spring Framework se configuran de forma automática como los externos.

- **Aplicaciones autónomas**

Se proporciona la capacidad de crear aplicaciones autónomas que se puedan ejecutar por sí solas.

- **Enfoque sencillo**

Spring Boot realiza la elección de qué paquetes instalar y qué valores predeterminados usar, según las necesidades del proyecto, con el fin de que el usuario no tenga que configurar todo de forma manual.

2.2. Frontend Web

2.2.1. Angular

Se trata de un framework de diseño de aplicaciones y plataforma de desarrollo para la creación de aplicaciones eficientes y sofisticadas de una sola página.

Es una plataforma construida sobre TypeScript. Esta plataforma incluye, con el fin de construir aplicaciones web escalables, un marco de trabajo que se encuentra basado en componentes. Además, proporciona herramientas que ayudan a que desarrolles, construyas, pruebes y actualices el código. Por otra parte, también incluye una colección de bibliotecas que se encuentran integradas y que engloban diferentes funciones (gestión de formularios, enrutamiento...).

El TypeScript es una herramienta incluida por defecto en todas las aplicaciones Angular con el fin de facilitar una mejor mantenibilidad y herramientas mejoradas. Por otro lado, la Interfaz de Línea de Comandos (CLI) es utilizada con el fin de optimizar el código abstrayendo la complejidad de las herramientas.

Angular tiene como objetivo proporcionar una estructura para que el proyecto se encuentre organizado en partes manejables y bien organizadas, haciendo del código escalable y mantenible.

2.2.2. React

Se trata de una librería Javascript de código abierto que está diseñada para la creación de interfaces de usuario con el fin de hacer más sencillo el desarrollo de aplicaciones en una sola página. Estas interfaces de usuario se construyen a partir de componentes, los cuales se pueden combinar formando así pantallas, páginas y aplicaciones. Estos componentes reciben datos y devuelven lo que debe aparecer en la pantalla, es decir, se pueden pasar datos en respuesta a una interacción.

React te proporciona la capacidad de agrupar dichos componentes, pero no prescribe cómo hacer el enrutamiento y como obtener los datos. Los frameworks por los que están compuestos también te dan la capacidad de obtener datos en componentes asíncronos ejecutados en el servidor o en la compilación.

Con React se pueden construir tanto aplicaciones web como nativas haciendo uso de las mismas habilidades. Además, proporciona la capacidad de servir HTML bajo demanda a la vez que aún se están cargando datos y también puede hacer uso de APIs web con el fin de mantener la interfaz *responsive*. Por otra parte, también está la posibilidad de crear aplicaciones nativas para Android e iOS.

2.2.3. Comparación Angular – React

Parámetros	Angular	React
Tipo	Framework completo	Biblioteca JavaScript
Tipo DOM	DOM real	DOM virtual
Abstracción	Media	Fuerte
Escrito en	Typescript	JavaScript
Enlace de datos	Datos bidireccionales	Vinculación de datos unidireccional
Inclusión en el código fuente la biblioteca JavaScript	No	Si
Libertad	Limitada	Facilita la selección de estructura, herramientas y bibliotecas.
Pruebas	Herramienta completa	Conjunto extra de herramientas

Para nuestro proyecto, se ha optado por la tecnología de React en lugar de Angular. A continuación, se presentan las razones divididas por secciones:

Facilidad de Aprendizaje y Desarrollo Rápido

Con react, la curva de aprendizaje es más suave, teniendo así la capacidad de aprender y crear un nuevo proyecto en menos tiempo que haciendo uso de otras plataformas.

Enfoque en Desarrollo Móvil

React es preferida en el desarrollo móvil, debido al React Native. Esto hace más sencilla la tarea de crear aplicaciones nativas para dispositivos móviles. De esta manera, se amplía la maniobrabilidad de los usuarios a la hora del desarrollo, ofreciendo una experiencia más fluida y eficiente.

Combinación HTML y JavaScript

A través del uso de JSX en React, se proporciona la capacidad de combinar el HTML y el JavaScript en el código, haciendo más simple la integración de dichas tecnologías.

Estabilidad en el código

Este código es caracterizado por su estabilidad debido a que hace uso de un flujo de datos descendente. Los cambios que se realicen en los componentes hijos no afectarán en los componentes padre. Gracias a esto, a la hora de depurar, los errores se pueden identificar de forma más sencilla y eficaz.

Componentes

Por otra parte, React también ofrece una estructura independiente basada en componentes donde estos pueden ser reciclados en otras partes pertenecientes a la aplicación.

Flexibilidad

React se caracteriza por un diseño de interfaz de usuario sencillo, ofreciendo gran variedad de extensiones que respaldan de forma integral la arquitectura de la aplicación, proporcionando de esta manera flexibilidad y versatilidad en el desarrollo.

2.3. App Móvil

2.3.1. Dart

Dart es un lenguaje de programación de código abierto creado con el fin de proporcionar a los desarrolladores la capacidad de utilizar un lenguaje orientado a objetos y con análisis estático de tipo. Es un lenguaje estructurado pero flexible para programación web. Este se puede utilizar en aplicaciones web, aplicaciones de consola, servidores y aplicaciones móviles.

Dart se caracteriza por poseer una programación estructurada y flexible, un lenguaje basado en clases e interfaces, por ser un lenguaje sencillo de aprender y porque permite adaptarse a una nueva herramienta a cualquier navegador web.

2.3.2. Framework Flutter

Flutter consiste en un framework de código abierto desarrollado y compatible con Google. Permite crear una interfaz de usuario de aplicación usable para diversas plataformas a través de un único código base, plataforma como iOS, Android, Windows...

Flutter, junto al lenguaje de programación Dart, se compila en código máquina, por lo que los dispositivos host entienden este código y de esta manera se obtiene un rendimiento eficaz y rápido. Por otro lado, Flutter hace uso de una biblioteca gráfica de código abierto llamada Skia con el fin de renderizar la interfaz de usuario, proporcionando de esta manera un rendimiento más rápido, personalizable y consistente. Además, es un framework caracterizado por ser de fácil uso, teniendo a disposición de los desarrolladores herramientas como la recarga en caliente o el inspector de widgets.

Dart se encuentra optimizado para crear interfaces de usuario y muchas de estas características son usadas en Flutter. Un ejemplo sería la seguridad de los nulos, haciendo más sencilla la detección de los errores más comunes, también denominados errores de nulos. Esto permite una mayor eficiencia para los desarrolladores, ya que no necesitan invertir tanto tiempo en el mantenimiento del código.

3. Cronología

La cronología de este proyecto ha sido desarrollada de forma secuencial, siguiendo un enfoque estructurado que abarca el diseño y la implementación de las diversas capas de la aplicación.

3.1. Análisis y planificación

En primer lugar, se ha realizado un análisis exhaustivo de los requisitos del proyecto, obteniendo información tanto de las necesidades de los usuarios como de las características principales. Se han definido los hitos del proyecto y las tecnologías que se van a utilizar para el desarrollo del mismo.

3.2. Creación tablas en base de datos

Se crea una conexión MySQL y posteriormente se crean las tablas necesarias para el proyecto.

3.2.1. Usuario

ID_Usuario → Clave primaria de la tabla

Nombre → Almacena el nombre del usuario

Correo_electronico → Guarda la dirección de correo electrónico del usuario

Contraseña → Contiene la contraseña del usuario

Fecha_Registro → Muestra la fecha en la que el usuario se ha registrado

Tipo_usuario → Campo que puede obtener dos valores: Regular o Administrador

3.2.2. Producto

ID_Producto → Clave primaria

ID_Metadato → Identificador asociado a metadatos específicos del producto

Nombre → Almacena el nombre del producto

Descripcion → Contiene una descripción detallada del producto

Fabricante → Fabricante del producto

Precio → Precio del producto

Valoracion → Calificación del producto

Imagen → URL de la imagen asociada al producto

ID_Usuario → Identificador del usuario asociado al producto

3.2.3. Categoria

ID_Categoria → Clave primaria

Nombre → Almacena el nombre de la categoria

Descripcion → Contiene una descripción de la categoria

Icono → Nombre el icono asociado a la categoria

Cantidad_Productos → Cantidad de productos asociados a la categoria

3.2.4. Comparativa

ID_Comparativa → Clave primaria

ID_Usuario → Representa al identificador del usuario asociado a la comparativa

ID_Producto_1 → Representa al identificador del primer producto asociado a la comparativa

ID_Producto_2 → Representa al identificador del segundo producto asociado a la comparativa

Titulo → Nombre para la comparativa

Descripcion → Descripción detallada de la comparativa

Valoracion → Puntuación asignada

3.2.5. Metadatos

ID_Metadato → Clave primaria

ID_Categoria → Identificador de la categoría asociada al metadato.

Nombre → Nombre del metadato

Tipo_Dato → Tipo de dato, puede ser texto, número o fecha

3.3. Backend - Entidades y Repositorios

Posteriormente, se han implementado las entidades de base de datos que corresponden a los componentes del sistema. Además, se ha implantado una infraestructura inicial desarrollando los repositorios con el fin de interactuar con la base de datos.

3.4. Backend - Modelo

Se ha desarrollado el modelo de datos que define la estructura y relaciones entre las entidades de la aplicación.

3.5. Backend - Servicios

A continuación, se han creado los servicios que contienen la lógica de negocio de la aplicación, de esta manera, se gestiona la interacción entre los controladores y repositorios.

3.6. Backend - Controladores y API

Se han desarrollado los controladores, estos tienen el objetivo de exponer las funcionalidades de la aplicación mediante una API REST.

3.7. Backend - Keycloak

Para la integración de Keycloak en el backend se han añadido las dependencias necesarias. Además, se ha configurado el URI del emisor del token JWT agregando en el archivo "application.properties" la siguiente línea:

```
spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/SpringBootKeycloak
```

Ilustración 1 - Keycloak backend

3.8. Backend - Seguridad

Se ha creado una clase llamada WebSecurityConfig que se encarga de la seguridad de la aplicación. En la que se habilita la seguridad web en la aplicación Spring con “@EnableWebSecurity”. Además, con el método de “addCorsMappings” permitimos que ciertos orígenes, en nuestro caso, “http://localhost:3000” y “<http://localhost:8888>”), puedan acceder a los recursos del backend. Los métodos HTTP permitidos también se especifican (GET, POST, PUT, DELETE).

3.9. Pruebas

Para realizar pruebas de API se ha hecho uso de una herramienta llamada Postman. Durante esta fase, se ha verificado el correcto funcionamiento de la API REST desarrollada. Para ello, se han realizado pruebas para cada “endpoint” definido en los controladores, validando el comportamiento y la integridad de los datos que se transmiten entre el cliente y el servidor.



Ilustración 2 - Postman pruebas

3.10. Frontend - Proyecto y Estructura de Carpetas

En la parte del desarrollo del frontend, se ha creado un proyecto React con el objetivo de construir la interfaz de usuario. El proyecto contiene una estructura de carpetas que sigue un enfoque modular y organizado con el fin de facilitar la escalabilidad y el mantenimiento del código. La estructura es la siguiente:

- src/componentes/categorias
 - Categorías.js: Componente que muestra y almacena las categorías de los productos.
 - Categorías.css: Estilos para el componente.
- src/componentes/comparativas
 - CompararComparativa.js: Componente para mostrar las comparativas realizadas por los usuarios.
 - CompararComparativa.css: Estilos para el componente.
 - AnadirComparativa.js: Componente que almacena los productos y las comparativas realizadas por los usuarios.
 - AnadirComparativa.css: Estilos para el componente.
 - ComparativaTReal.js: Componente utilizado para mostrar comparativas en tiempo real de productos seleccionados por el usuario.
 - ComparativaTReal.css: Estilos para el componente.
- src/componentes/inicio
 - PaginaInicio.js: Componente que muestra la página inicial de la aplicación.
 - PaginaInicio.css: Estilos para el componente.
- src/componentes/listaproductos
 - ListaProductos.js: Componente para mostrar y gestionar la lista de productos.
 - ListaProductos.css: Estilos para el componente.
- src/componentes/metadatos
 - Metadatos.js: Componente para mostrar y gestionar los metadatos.
 - Metadatos.css: Estilos para el componente.
- src/componentes/sidebar
 - Sidebar.js: Componente principal de la barra lateral. Es la estructura general, la que incluye la lógica para gestionar el estado y la visibilidad.
 - Sidebar.css: Estilos para el componente.
 - SidebarData: Contiene información sobre los elementos del menú que se mostrarán en la barra lateral.
 - SidebarMenu: Parte integral que se utiliza para renderizar elementos de menú secundarios o desplegados.

- src/pages/
 - Comparativa.js: Página que utiliza el componente Comparativas con el fin de mostrar comparativas específicas.
 - Inicio.js: Se utiliza el componente Inicio para mostrar el inicio de la aplicación.
 - Login.js: Página que hace uso del componente Login para la gestión de inicio de sesión.
 - Producto.js: Se utiliza el componente ListaProductos junto al componente Metadatos para mostrar y filtrar los productos.
 - Pages.css: Estilos generales para las páginas.

3.11. Frontend - Integración de componentes

Con el fin de construir la estructura visual de la aplicación, los componentes anteriormente mencionados se han integrado en las páginas correspondientes. Por otra parte, hay coherencia en el diseño y la navegación entre las distintas secciones.

3.12. Frontend - Estilos y Diseño

Se ha aplicado un diseño responsive y agradable con el objetivo de mejorar la experiencia del usuario. Los archivos CSS correspondientes a cada componente y página son los que poseen los estilos definidos.

3.13. Frontend - Navegación

Se ha desarrollado una barra lateral para facilitar la navegación entre las secciones de la aplicación. Además, esta ha sido integrada de forma global para ofrecer una experiencia de usuario consistente.

3.14. Frontend - Comunicación con el Backend

Algunos de los componentes necesitan comunicarse con el backend, para ello se han desarrollado llamadas a la API mediante la biblioteca *fetch*.

3.15. Frontend - Ejecución de la aplicación

En primer lugar, para cargar la parte del backend y la base de datos, se hace uso del comando “mvn spring-boot:run”. Con esto iniciamos el servidor de Spring Boot y desplegamos la aplicación en el entorno de desarrollo. A continuación, mediante el comando de “npm start” se carga el frontend, iniciando de esta manera el servidor de React y proporcionando la capacidad de ver la interfaz de usuario creada en el navegador web.

3.16. Frontend - Inicio de sesión

Para implementar el sistema de inicio de sesión de la aplicación se ha hecho uso de Keycloak. Con el fin de acceder a la aplicación, los usuarios tienen que autenticarse mediante Keycloak, el cual actúa como proveedor de identidad. Posteriormente, aquellos usuarios que se encuentren autenticados tendrán acceso a las diferentes funcionalidades de sus respectivos roles.

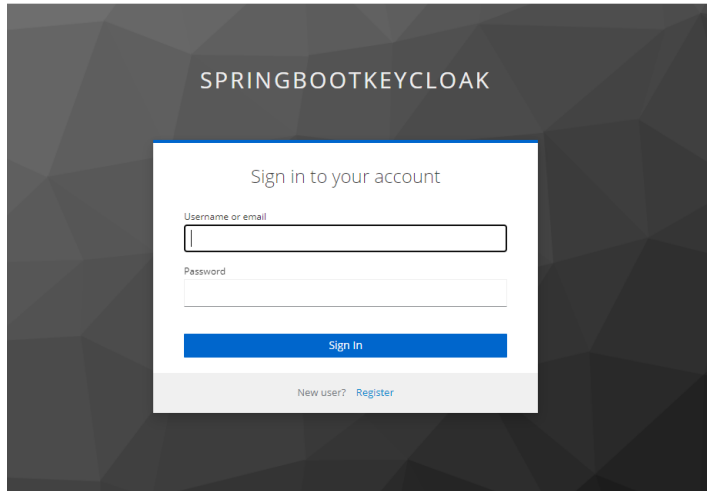


Ilustración 3 - Iniciar sesión con Keycloak

Por otra parte, a través de Keycloak también nos podemos registrar como nuevo usuario. Para ello tienen que completar un registro en el que deberán proporcionar el nombre y apellido, el correo electrónico, el nombre de usuario y la contraseña.

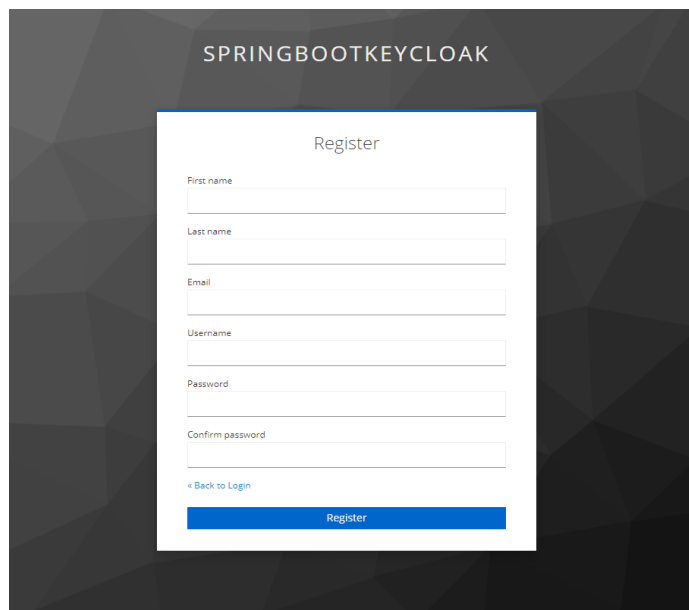


Ilustración 4 - Registrarse con Keycloak

Para la integración de keycloak en la aplicación se han importado los módulos necesarios y se han configurado las opciones necesarias para su inicialización.

```
const httpClient = axios.create();

let inicializarOpciones = {
  onLoad: 'login-required',
  checkLoginIframe: true,
  pkceMethod: 'S256',
  url: "http://localhost:8080/",
  realm: "SpringBootKeycloak",
  clientId: "login-app",
}

export const keycloak = new Keycloak(inicializarOpciones);

keycloak.init(inicializarOpciones).then((auth) => {
  if (!auth) {
    window.location.reload();
  } else {
    httpClient.defaults.headers.common['Authorization'] = 'Bearer ' + keycloak.token;

    keycloak.onTokenExpired = () => {
      console.log('token expired')
    }
  }
}, () => {
  console.error("Authentication Failed")
});
```

Ilustración 5 - Integración de Keycloak en Frontend

3.17. Frontend - Lista de Productos

Para mostrar la lista de productos se ha implementado de manera que se tenga que seleccionar una categoría de forma previa por parte del usuario. Al entrar a la página de lista de productos podemos observar un desplegable en el que, dependiendo de la categoría seleccionada, se podrá filtrar por unos productos u otros. En este caso se ha seleccionado “Teclado”, y muestra el nombre, descripción, precio, fabricante y valoración de cada teclado.

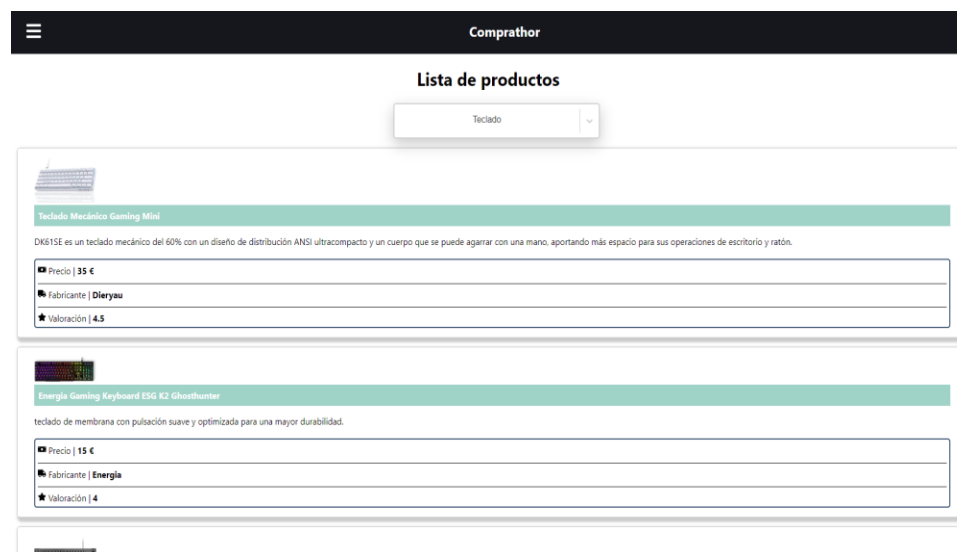


Ilustración 6 - Lista de productos Frontend

3.18. Frontend - Comparativas

En esta sección se podrán visualizar y analizar las comparativas entre productos realizadas en la aplicación.

3.18.1. Visualización

En esta página los usuarios tienen la posibilidad de explorar y filtrar las comparativas de productos realizadas en la aplicación. Se proporciona la capacidad de filtrar por comparativas hechas por otros usuarios o aquellas que han sido creadas por el propio usuario. Además, también se puede filtrar por el producto, en este caso, el teclado.

Cada comparativa presenta su correspondiente título y descripción, proporcionando de esta manera aspectos destacados de la comparación. También incluye una tabla en la que se pueden comparar los dos productos, permitiendo una evaluación rápida de las diferencias entre ellos.

Comprathor

Comparativa Tiempo Real

Añadir Comparativa

Comparativas de otros usuarios

Teclado

Teclados Dieryau y Energia

El teclado Dieryau destaca por su diseño ergonómico y retroiluminación ajustable, el teclado Energia se distingue por su durabilidad y rendimiento excepcional

	Nombre	Descripción	Fabricante	Precio	Valoración
	Teclado Mecánico Gaming Mini	DK61SE es un teclado mecánico del 60% con un diseño de distribución ANSI ultracompacto y un cuerpo que se puede agarrar con una mano, aportando más espacio para sus operaciones de escritorio y ratón.	Dieryau	35 €	4.5 ★
	Energia Gaming Keyboard ESG K2 Ghosthunter	Teclado de membrana con pulsación suave y optimizada para una mayor durabilidad.	Energia	15 €	4 ★

Teclado QWERTY y KEMOVA

El "Teclado con Cable, QWERTY" ofrece funcionalidad y comodidad para un uso general y diario, el "Teclado KEMOVA" está diseñado específicamente para satisfacer las necesidades de los jugadores profesionales



	Nombre	Descripción	Fabricante	Precio	Valoración
	Teclado con Cable, QWERTY	Rioi RK907 teclado USB posee tamaño completo, ultra delgado y compacto. Cuenta con cable y 105 teclas con teclado numérico. Una gama completa de funciones	Rioi	13 €	3 ★
	Teclado Gamer Mecánico	El teclado mecánico K98SE presenta un diseño de 98 teclas que maximiza la eficiencia del espacio, preservando letras esenciales, números y teclas de función.	KEMOVA	50 €	5 ★

Ilustración 7 - Visualización de comparativas Frontend

3.18.2. Visualización en Tiempo real

Este apartado de la aplicación permite a los usuarios visualizar comparativas de productos en tiempo real. Al seleccionar un tipo, se actualizan los datos de los dos desplegables de los productos y al seleccionar un producto, se actualiza automáticamente la interfaz reflejando los cambios, proporcionando de esta manera la información que necesitan los usuarios de manera rápida y eficiente.

The screenshot shows the 'Comprathor' application interface. At the top, there's a navigation bar with a menu icon and the title 'Comprathor'. Below it, a dropdown menu is set to 'Teclado'. A second dropdown menu shows 'Teclado con Cable, QWERTY'. The main content area displays two product cards. The first card is for 'Teclado con Cable, QWERTY' by Rilo, priced at 13 €, with a 3-star rating. The second card is for 'Teclado Gamer Mecánico' by KEMOVA, priced at 50 €, with a 5-star rating. Each card includes a small image of the product and a brief description.

Ilustración 8 - Visualización de comparativas en tiempo real Frontend

3.18.3. Añadir Comparativa

Los usuarios también tienen la capacidad de insertar una comparativa entre productos, pudiendo compartir opiniones y experiencias con el resto de los usuarios. Cada comparativa está compuesta por un título y su descripción, así como de los dos productos a comparar, de los cuales se necesita su nombre, descripción, fabricante, precio, valoración e imagen.

The screenshot shows the 'Añadir una comparativa' form in the 'Comprathor' application. The form has a dropdown for 'Tipo' and a 'Compartir' button. Below these are input fields for 'Título' and 'Descripción'. The main part of the form is divided into two columns: 'Producto 1' and 'Producto 2'. Each column contains input fields for 'Nombre', 'Descripción', 'Fabricante', 'Precio', 'Valoración', and 'Imagen (URL)'. At the bottom, there's a button labeled 'Añadir nueva comparativa'.

Ilustración 9 - Añadir comparativa Frontend

3.19. App Móvil - Proyecto y Estructura de Carpetas

En la parte del desarrollo de la aplicación móvil, se ha organizado el proyecto de manera modular para facilitar el mantenimiento y escalabilidad del código. La estructura de carpetas es la siguiente:

- Comparativas
 - `anadircomparativa.dart`: Contiene el componente para añadir nuevos productos y comparativas.
 - `compararproductos.dart`: Componente que se utiliza para mostrar comparativas en tiempo real de los productos seleccionados por el usuario.
 - `comparativas.dart`: Contiene el componente para añadir nuevas comparativas entre productos.
- Header
 - `header.dart`: Contiene el componente del menú navegación que se muestra en la esquina superior de la aplicación.
- Inicio
 - `inicio.dart`: Componente que muestra la pantalla de inicio de la aplicación. En ella se muestra una barra de búsqueda donde los usuarios pueden buscar productos por su nombre.
- ListaProductos
 - `ListaProductos.dart`: Componente que sirve para mostrar y gestionar la lista de productos. Proporciona la capacidad de filtrar los productos por nombre y tipo de producto.
- Productos
 - `productos.dart`: Página donde se obtienen los parámetros de búsqueda para la lista de productos. Después de obtener el tipo de producto deseado se muestran los productos llamando al componente ListaProductos.
- Config
 - `config.dart`: Contiene la configuración y métodos para interactuar con el backend. Incluye métodos para realizar solicitudes HTTP al servidor y de esta manera poder obtener e insertar datos en la base de datos.
- Main
 - `main.dart`: Punto de entrada principal de la aplicación. Se inicia la ejecución del código Dart y se define la estructura general de la aplicación. Asimismo, también se configura e inicializa Keycloak con el fin de gestionar la autenticación de los usuarios.

3.20. App Móvil - Desarrollo con Flutter

En primer lugar, se inicia un proyecto en Flutter donde se establecen las configuraciones iniciales del proyecto, como el nombre, el paquete y la versión. Se instalan los SDKs requeridos y se añaden las dependencias necesarias en el archivo “pubspec.yaml”.

3.21. App Móvil - Diseño interfaz de usuario

Se ha realizado un diseño donde se establece la estructura de navegación de la aplicación, los elementos de la interfaz como botones, campos de entrada, formularios y barras de navegación.

3.22. App Móvil - Estilos

Se han aplicado los estilos de la aplicación, como colores, tipografías, etc. Para ello se ha hecho uso de widgets de Flutter como “TextStyle”.

3.23. App Móvil - Navegación

Se ha implementado el enrutamiento de la aplicación a través de un desplegable en el “header”. Las rutas y transiciones entre las diversas pantallas se han definido haciendo uso del widget “Navigator”.

3.24. App Móvil - Comunicación con el Backend

A través del paquete http de Flutter se realizan las solicitudes al backend. Se establecen las rutas de la API del backend la cuales serán utilizadas por la aplicación móvil. Estas incluyen “endpoints” para poder crear, obtener, actualizar y eliminar datos que tengan relaciones con los productos, comparativas, etc.

Un ejemplo sería la función de “fetchProductos()” en la que se manejan los casos de éxito y error en la obtención de productos:

```
Future<void> fetchProductos() async {  
  final response = await ApiService.fetchData('productos');  
  
  if(response.statusCode == 200) {  
    setState(() {  
      productos = jsonDecode(response.body);  
    });  
  } else {  
    throw Exception('Fallo cargando los productos');  
  }  
}
```

Ilustración 10 - Función fetchProductos

Para este caso se utiliza otra clase general que es la que facilita la comunicación con el backend.

```
class ApiService {
  static const String BaseUrl = 'http://localhost:8081';

  static Future<http.Response> fetchData(String endpoint) async {
    return http.get( Uri.parse('$BaseUrl/$endpoint') );
  }

  static Future<http.Response> postData(String endpoint, dynamic data) async {
```

Ilustración 11 - Clase general que conecta con Backend desde App móvil

3.25. App Móvil - Inicio de sesión

Se ha integrado un sistema de inicio de sesión en el que los usuarios pueden autenticarse mediante el uso de Keycloak. Para ello, se ha utilizado la dependencia “keycloak_flutter: ^0.0.21” junto a la configuración del URI del emisor, el realm, y el clientID.

```
late KeycloakService keycloakService;

Future<void> main() async {
  keycloakService = KeycloakService(KeycloakConfig(
    url: "http://localhost:8080/",
    realm: "SpringBootKeycloak",
    clientId: "login-app",
  )); // KeycloakConfig, KeycloakService
  await keycloakService.init(
    initOptions: KeycloakInitOptions(
      onLoad: 'login-required',
      checkLoginIframe: true,
      pkceMethod: 'S256',
    ),
  );

  runApp(MyApp());
}
```

Ilustración 12 - Configuración Keycloak App móvil

3.26. App Móvil - Lista de productos

Para mostrar la lista de productos se ha creado un “header” con un desplegable en el que muestra las diferentes categorías. Al entrar a una de ellas carga los productos para seleccionar en un desplegable, y al seleccionar un producto procede a mostrar los productos con sus características.

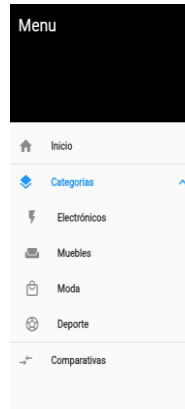


Ilustración 13 - Menú de navegación App móvil

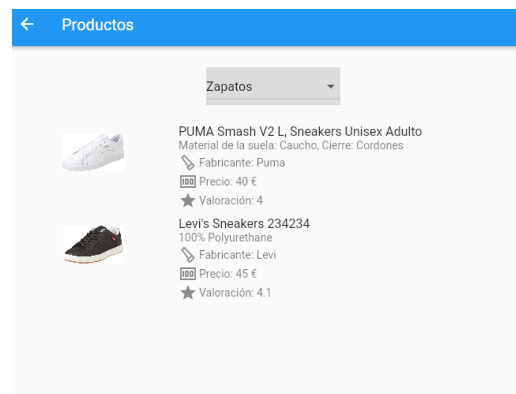


Ilustración 14 - Lista de productos App móvil

3.27. App Móvil - Comparativas

En esta sección se podrán visualizar y analizar las comparativas entre productos realizadas en la aplicación.

3.27.1. Visualización

Se ha creado una interfaz en la que los usuarios pueden explorar las comparativas que ya están creadas. Cada comparativa se compone de un título y su descripción junto a los dos productos comparados.

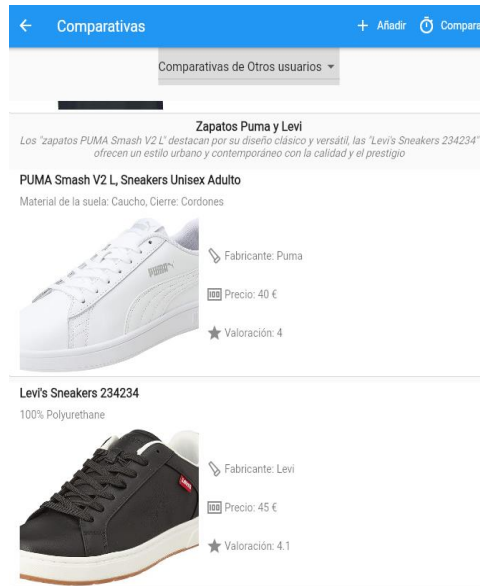


Ilustración 15 - Comparativas de productos App Móvil

3.27.2. Visualización en Tiempo real

En esta sección los usuarios tienen la capacidad de observar comparativas de productos en tiempo real. Al seleccionar un producto en el desplegable se actualiza la interfaz de forma automática para que el usuario pueda realizar las correspondientes comparaciones de manera fluida.

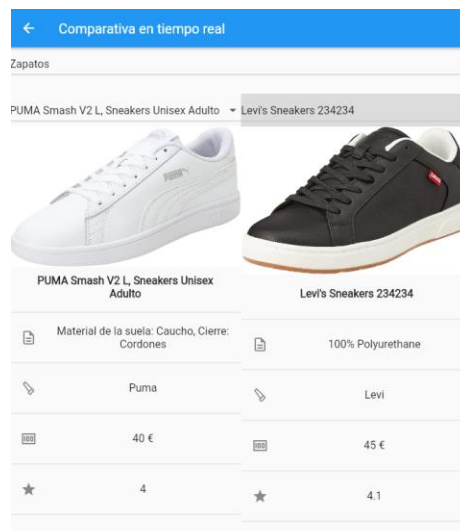


Ilustración 16 - Comparativas de productos en tiempo real App Móvil

3.27.3. Añadir Comparativa

Los usuarios pueden crear nuevas comparativas seleccionando el producto en el desplegable y rellenando los campos necesarios para guardar una comparativa.

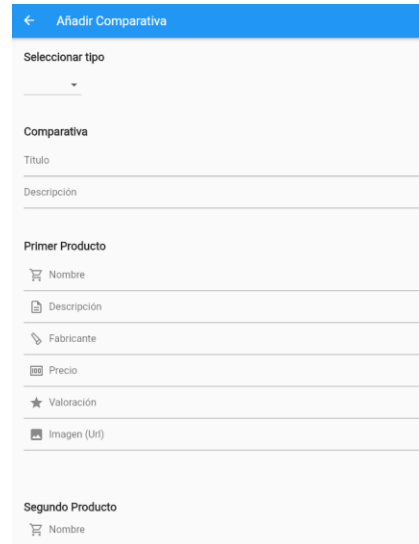


Ilustración 17 - Añadir comparativa App móvil

4. Detalle y Justificación

4.1. Diagramas de flujo

4.1.1. Diagrama de casos de uso

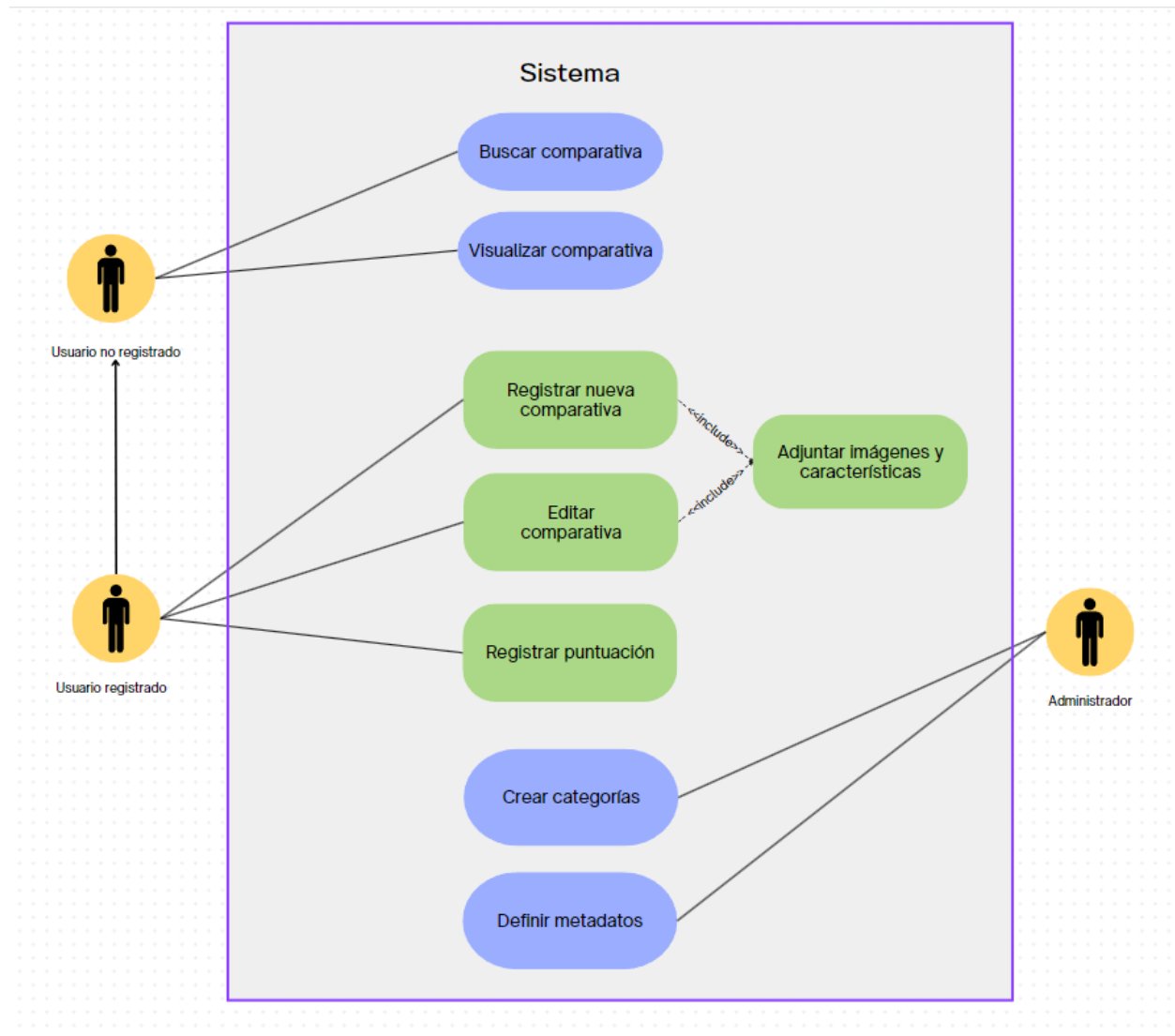


Ilustración 18 - Diagrama casos de uso

4.1.2. Diagrama de flujo

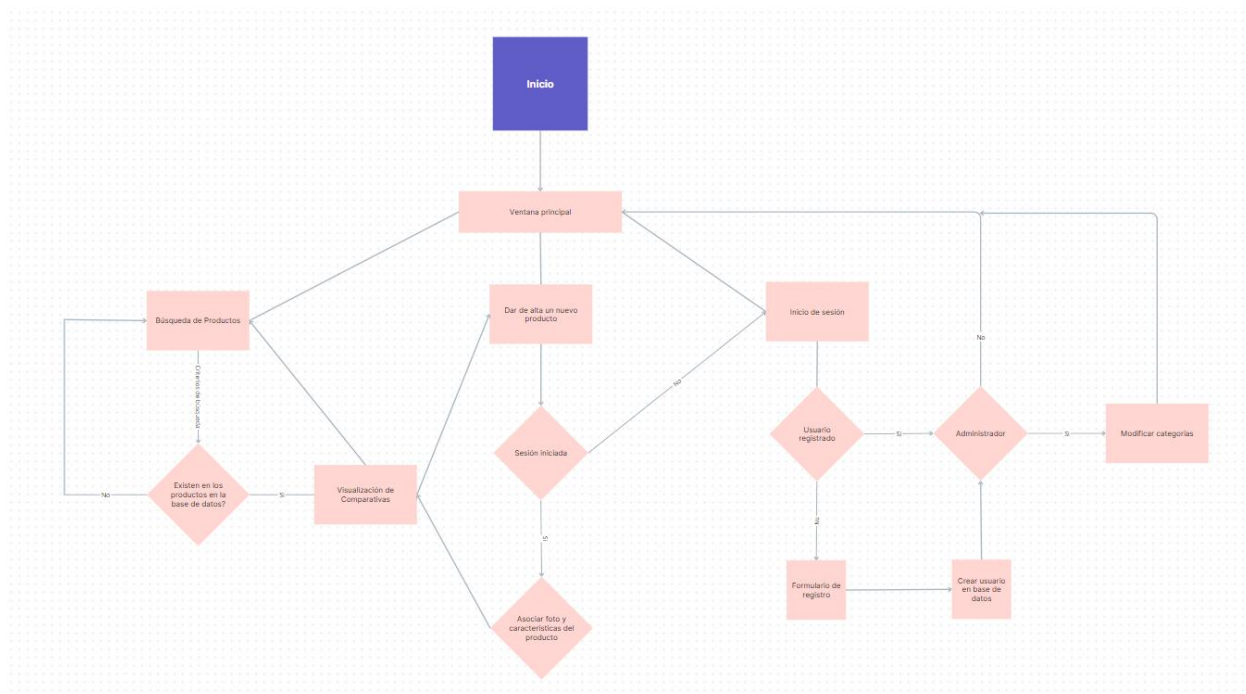


Ilustración 19 - Diagrama de flujo

4.1.3. Diagrama de base de datos

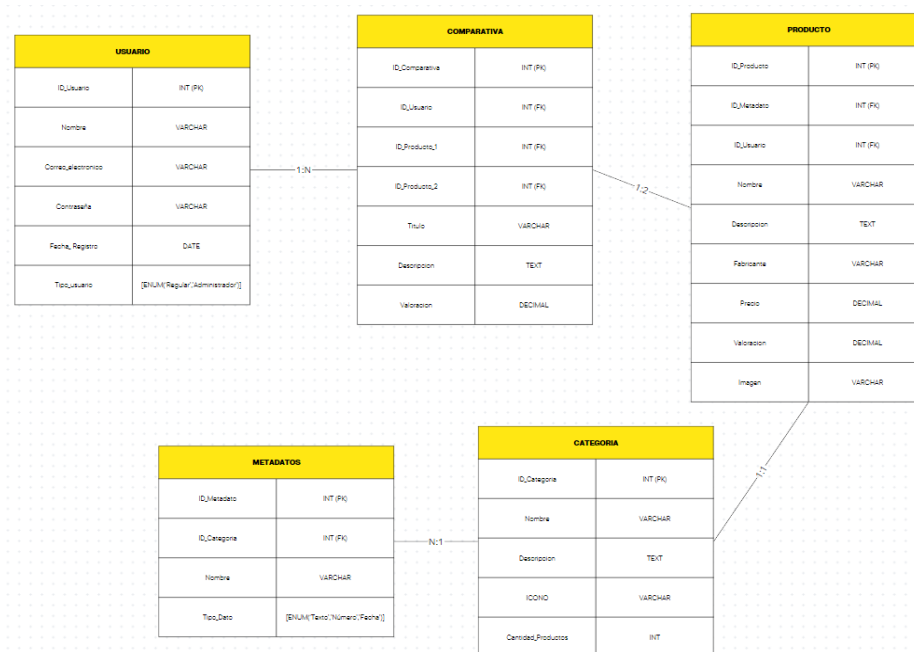


Ilustración 20 - Diagrama de base de datos

4.1.4. Wireframes

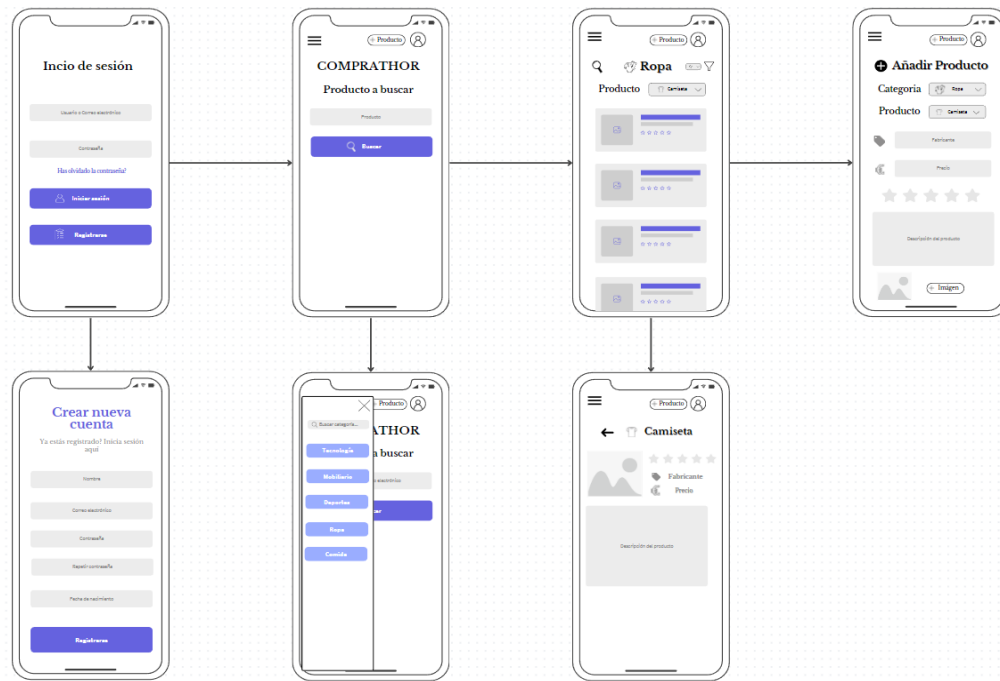


Ilustración 21 - Wireframes

4.2. Backend

La arquitectura del backend de la aplicación está compuesta por las capas de controladores, servicios y repositorios.

4.2.1. Controladores

Estos tienen como objetivo gestionar las solicitudes HTTP y las respuestas. Se ha decidido separar de esta forma para definir de forma clara y sencilla las rutas de la API. Asimismo, separando la lógica de negocio con la lógica de la interfaz de usuario, mejoramos la mantenibilidad. En este caso los controladores de la aplicación son:

- **CategoriaController:** Encargado de gestionar operaciones relacionadas con las categorías de productos.
- **ComparativaController:** Gestiona las comparativas realizadas por los usuarios.
- **MetadatosController:** Gestiona las operaciones relacionadas con los metadatos.
- **ProductoController:** Encargado de gestionar las operaciones relacionadas con los productos.
- **UsuarioController:** Maneja las operaciones relacionadas con los usuarios.

Cada uno de ellos define las rutas de la API para realizar las operaciones CRUD correspondientes.

4.2.2. Servicios

Estos contienen la lógica de negocio de la aplicación y tienen la función de comunicar con la capa de repositorios. La lógica de negocio se ha separado en servicios con el fin de facilitar las pruebas unitarias y reutilizar código. Además, de esta manera, el sistema es más modular y sencillo de entender. En este contexto, los servicios que posee la aplicación son:

- CategoriaService/CategoriaServiceImpl: Incluye la lógica de negocio para las categorías de productos. Facilita la comunicación entre el controlador de categorías y el repositorio que corresponde.
- ComparativaService/ComparativaServiceImpl: Contiene la lógica de negocio para el manejo de comparativas. Actúa como intermediario entre el controlador y el repositorio de comparativas.
- MetadatosService/MetadatosServiceImpl: Incluye la lógica de negocio para los metadatos. Facilita la comunicación entre el controlador de metadatos y el repositorio correspondiente.
- ProductoService/ProductoServiceImpl: Contiene la lógica de negocio para el manejo de productos. Facilita la comunicación entre el controlador y el repositorio de productos.
- UsuarioService/UsuarioServiceImpl: Incluye la lógica de negocio para las operaciones que tengan relación con los usuarios. Facilita la comunicación entre el controlador de usuarios y el repositorio correspondiente.

4.2.3. Repositorios

Estos se encargan de gestionar la interacción con la base de datos. Con esta separación podemos mejorar la gestión de la persistencia de datos y también, en caso necesario, nos proporciona la capacidad de cambiar la fuente de datos. Para este caso, se tienen los siguientes repositorios:

- CategoriaRepository: Encargado de gestionar las operaciones de persistencia de datos para las categorías.
- ComparativaRepository: Gestiona las operaciones de persistencia relacionadas con las comparativas.
- MetadatosRepository: Responsable de las operaciones de persistencia relacionadas con los metadatos
- ProductoRepository: Maneja las operaciones de persistencia de datos para los productos.
- UsuarioRepository: Encargado de las operaciones de persistencia relacionadas con los usuarios

4.2.4. Java y Framework Spring-Boot

Por otro lado, se ha seleccionado java como lenguaje de programación debido a su robustez y eficacia en el desarrollo. Además, posee un gran ecosistema de bibliotecas y también la capacidad de ejecutarse en cualquier plataforma.

El Framework Spring-Boot se ha seleccionado debido a que tiene como objetivo simplificar la configuración y el desarrollo de aplicaciones Java que estén basadas en el framework Spring.

4.2.5. Lombok

Se trata de una biblioteca de Java que se integra de forma automática con el editor y herramientas de compilación. Se ha seleccionado esta biblioteca porque facilita la experiencia con Java, ya que escribe automáticamente los “getters” y “setters”, crea constructores y hace más sencillo el registrar variables.

4.2.6. Swagger

Se ha integrado la biblioteca de Springdoc OpenAPI:

```
<dependency>
  <groupId>org.springdoc</groupId>
  <artifactId>springdoc-openapi-ui</artifactId>
  <version>1.6.12</version>
</dependency>
```

Esto facilita la generación automática de la documentación OpenAPI. Se puede comprobar la documentación accediendo a: <http://localhost:8081/swagger-ui/index.html#/>

4.2.7. Seguridad y Keycloak

Para garantizar la seguridad de la aplicación y la autenticación de los usuarios, se ha integrado el framework Spring Security junto con Keycloak. Esta combinación proporciona una capa robusta de seguridad que protege tanto el backend como el frontend de la aplicación. Las dependencias clave utilizadas son:

- spring-boot-starter-security: Proporciona las funcionalidades básicas para la configuración de la seguridad en aplicaciones Spring Boot.
- spring-boot-starter-oauth2-client: Se usa con el fin de habilitar el cliente OAuth2 en la aplicación Spring Boot.

- spring-boot-starter-oauth2-resource-server: Habilita el servidor de recursos OAuth2 en la aplicación.
- spring-boot-starter-oauth2-jose: Proporciona el soporte necesario para manejar tokens JWT en Spring Security.

4.3. Frontend

Se trata de la interfaz mediante la cual los usuarios interactúan con la aplicación, por lo tanto, se requiere un buen diseño y funcionalidad para una mejor experiencia de usuario. En este apartado se detallan las elecciones y estrategias desarrolladas.

4.3.1. Librerías y Frameworks

La biblioteca principal usada para el desarrollo del frontend es React. Esta sirve para construir interfaces de usuario eficientes y reactivas. Por otra parte, también se han utilizado bibliotecas y frameworks adicionales:

- React Router: Gestiona la navegación en la aplicación. Ofrece la posibilidad de crear rutas y transiciones entre componentes.
- Styled components: Se ha hecho uso de esta librería para aplicar estilos de forma más modular.
- Axios: Utilizado para realizar solicitudes HTTP desde el cliente hacia el servidor, permitiendo de forma asíncrona la comunicación entre el frontend y el backend de la aplicación.
- React-icons: Gran variedad de iconos disponibles que se han integrado en la aplicación con el fin de mejorar la experiencia de usuario.
- React-bootstrap y bootstrap: Bibliotecas que proporcionan componentes que ya están estilizados y funcionalidades de diseño basadas en Bootstrap. De esta manera se facilita la creación de interfaces de usuario responsivas y atractivas.

4.3.2. Organización del Código:

La estructura del proyecto se ha dividido en componentes reutilizables para diferentes partes de la interfaz de usuario, mejorando de esta manera la modularidad y legibilidad del código.

- Componentes: Se han desarrollado componentes para las diversas partes de la interfaz, por ejemplo, *Categorias*, *Comparativas*, *ListaProductos*, etc.
- Pages: Esta carpeta posee componentes que representan páginas completas de la aplicación.

4.3.3. Manejo de Estado

Para un manejo de estado de forma efectivo se ha usado el hook `useState`. Con el fin de mejorar la claridad y previsibilidad del código, los componentes mantienen su propio estado interno cuando se requiere.

4.3.4. Variables globales

Se ha definido una variable global llamada “`API_BASE_URL`” la cual almacena la URL base para las solicitudes al backend. Esta variable es utilizada para construir las URL completas de los “endpoints” a los que se realizarán las solicitudes HTTP. En este caso, la URL se establece como “`http://localhost:8081`”.

4.4. App Móvil

La aplicación móvil proporciona a los usuarios una interfaz intuitiva y eficiente para la interacción con la plataforma a través de dispositivos móviles. En este apartado se van a detallar las principales decisiones y estrategias que se han implementado con el fin de garantizar una experiencia fluida y satisfactoria.

4.4.1. Framework y Herramientas

El desarrollo de la aplicación está basado en el uso del framework Flutter. Este framework es famoso por su capacidad de crear interfaces de usuario de alto rendimiento y atractivas para el usuario. Como complemento, se ha usado:

- Flutter Keycloak: Con el fin de integrar el sistema de autenticación basado en Keycloak.
- Paquete HTTP: La obtención y el envío de datos se ha realizado mediante solicitudes HTTP al servidor backend.

4.4.2. Estructura del proyecto

La aplicación móvil se encuentra organizada de manera modular y estructurada, facilitando de esta manera el mantenimiento y la escalabilidad.

- Directorios: Se han creado directorios para cada tipo de componente y funcionalidad, como Inicio, Head, Comparativas, ListaProductos, etc.
- Archivos: Cada archivo posee componentes o funcionalidades que se encuentran relacionadas entre sí, lo que facilita la legibilidad y claridad del código.

4.4.3. Configuración y Comunicación con el Backend

La aplicación móvil se conecta con el backend mediante solicitudes HTTP con el fin de obtener y enviar datos. En el archivo `config.dart` es donde se encuentra la configuración para estas solicitudes, en dicho archivo están definidos los métodos necesarios para la comunicación con el servidor.

4.4.4. Keycloak Flutter

Para la integración del Keycloak en la aplicación móvil, se ha añadido la dependencia “keycloak_flutter” en la versión ^0.0.21. Esta ofrece las funcionalidades que se requieren para la autenticación y gestión de usuarios mediante Keycloak. Asimismo, se ha añadido la carpeta “keycloak-js” en la estructura del proyecto, la cual es necesaria para la configuración de Keycloak en el lado del cliente. De esta manera se puede establecer una comunicación segura con el servidor de Keycloak y gestionar los tokens de autenticación de manera adecuada.

4.4.5. Variables globales

En la clase `ApiService` se ha definido una variable global para facilitar el acceso y la gestión de las solicitudes HTTP hacia el servidor backend:

- `BaseUrl`: Variable global que almacena la URL base del servidor backend al que se realizan las solicitudes ([‘http://localhost:8081’](http://localhost:8081)).

Por otra parte, hay funciones globales, como *fetchData* y *postData* que sirven para realizar solicitudes GET y POST tomando como parámetros un “endpoint”, y en el caso del post, también toma un parámetro adicional llamado “data”.

5. Trabajo futuro

El proyecto se ha diseñado con flexibilidad y escalabilidad en mente. A continuación, se van a presentar áreas clave para el desarrollo futuro con el fin de mejorar aún más la experiencia de usuario y la funcionalidad.

5.1. Ampliación de la funcionalidad de Comparativas

La aplicación permite a los usuarios realizar una comparativa de solamente dos productos a la vez. Para un futuro, se espera extender dicha funcionalidad para que los usuarios tengan la capacidad de comparar más de dos productos en una misma comparativa. Con este cambio se va a aumentar el detalle de las comparativas.

5.2. Sistema de comentarios y reseñas

Con el fin de aumentar la participación de los usuarios, en un futuro se va a desarrollar un sistema de comentarios y reseñas para los productos y comparativas. De esta manera, se permitirá a los usuarios realizar preguntas, compartir experiencia y comentar sobre productos específicos.

5.3. Funcionalidad de favoritos

Para que el usuario obtenga la capacidad de crear listas personalizadas de productos y comparativas preferidas, se va a implementar una funcionalidad nueva para que los usuarios puedan marcar dichos productos o comparativas como favoritas. Además, esto también va a permitir un acceso más rápido y fácil a aquellos productos y comparativas más relevantes para el usuario.

5.4. Notificaciones y actualizaciones

Introducir la capacidad de que los usuarios puedan recibir notificaciones sobre nuevas comparativas, interacciones en sus comparativas o actualizaciones de comparativas o productos.

5.5. Soporte multilingüe

Con el fin de llegar a una audiencia más amplia, se va a desarrollar un soporte multilingüe. Se va a desarrollar a través de traducciones y opciones de idioma.

6. Conclusiones

El desarrollo del Trabajo Final de Máster (TFM) en Full Stack Developer ha sido una experiencia enriquecedora la cual ha permitido aplicar de manera práctica los conocimientos adquiridos durante el curso. Durante la realización de este proyecto, se ha puesto en práctica una amplia variedad de habilidades técnicas y metodológicas, lo que ha contribuido de manera significativa a mi crecimiento profesional y personal.

Una de las principales conclusiones derivadas de este proyecto es la importancia de tener una arquitectura de solución bien diseñada y estructurada. La planificación de la arquitectura ha sido fundamental para garantizar la escalabilidad, eficiencia y mantenibilidad del propio proyecto. Asimismo, el diseño que posee el Front-End y el Back-End se ha orientado con el fin de proporcionar una experiencia de usuario óptima.

Por otra parte, el desarrollo de la aplicación móvil usando Dart y el Framework Flutter ha permitido la posibilidad de extender la funcionalidad de la plataforma a dispositivos móviles. La capacidad proporcionada a los usuarios de poder acceder a las comparativas de productos en movilidad ha mejorado de forma significativa la accesibilidad de los mismos.

Además, se han implementado las APIs para el intercambio de información entre los diferentes módulos de la aplicación, lo que ha sido esencial para garantizar la cohesión y la interoperabilidad del sistema. La comunicación entre el Backend, el Frontend y la aplicación móvil han sido facilitadas por dichas APIs.

En resumen, el desarrollo del Trabajo Final de Grado “COMPRATHOR: Comparador personal de productos” ha sido un proceso desafiante, pero a la vez gratificante. Mediante este trabajo he podido demostrar mi capacidad con el diseño, desarrollo y despliegue de soluciones software Full Stack. Esta experiencia me ha preparado para enfrentar con éxito los desafíos del mundo laboral en el campo del desarrollo de software.

7. Bibliografía

1. ¿Qué es Java y por qué lo necesito?. [En línea]. Disponible en: https://www.java.com/es/download/help/whatis_java.html
2. Oracle Java. [En línea]. Disponible en: <https://www.oracle.com/es/java/#:~:text=Oracle%20Java%20es%20la%20plataforma,los%20servicios%20de%20las%20aplicaciones>
3. ¿Qué es Java?. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/java/>
4. ¿Qué es Java Spring Boot?. [En línea]. Disponible en: <https://www.ibm.com/es-es/topics/java-spring-boot>
5. Angular. [En línea]. Disponible en: <https://angular.io/>
6. React Wikipedia. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/React>
7. Sitio oficial de React. [En línea]. Disponible en: <https://es.react.dev/>
8. Zahid Powell. Angular vs React: una comparación en profundidad. [En línea]. 2023. Disponible en: <https://kinsta.com/es/blog/angular-vs-react/#angular-vs-react-una-comparacin-en-profundidad>
9. React vs Angular: Similitudes y diferencias. [En línea]. 2022. Disponible en: <https://www.epitech-it.es/react-vs-angular/>
10. Dart Wikipedia. [En línea]. Disponible en: <https://es.wikipedia.org/wiki/Dart>
11. Miguel José Martínez Córdón. ¿Qué es el lenguaje de programación Dart?. [En línea]. 2021. Disponible en: <https://www.hiberus.com/crecemos-contigo/que-es-el-lenguaje-de-programacion-dart/>
12. ¿Qué es el Flutter?. [En línea]. Disponible en: <https://aws.amazon.com/es/what-is/flutter/#:~:text=interfaz%20de%20usuario,-.%C2%BFQu%C3%A9%20lenguaje%20de%20programaci%C3%B3n%20utiliza%20Flutter%3F,Dart%2C%20que%20tambi%C3%A9n%20desarroll%C3%B3%20Google.>
13. Project Lombok. [En línea]. Disponible en: <https://projectlombok.org/>
14. Brian Design. React Sidebar with Dropdown Menu Tutorial - Create Sub Navigation. [YouTube]. Disponible en: https://www.youtube.com/watch?v=mN3P_rv8ad4
15. Documentación React Router. [En línea]. Disponible en: <https://reactrouter.com/es/main/start/concepts>
16. Paquete Keycloak_flutter. [En línea]. Disponible en: https://pub.dev/packages/keycloak_flutter
17. Michael Good. A Quick Guide to Using Keycloak With Spring Boot. [En línea]. 2024. Disponible en: <https://www.baeldung.com/spring-boot-keycloak>