Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

SMARTCART

Alumno: Fernando Scott Humanes Carchenilla

Tutor/a/es: José Vicente Álvarez

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que me han apoyado desde el inicio de este camino.

A mis padres y a mi hermano, gracias por ser una fuente inagotable de amor y fortaleza, por creer siempre en mí y por darme el impulso necesario para seguir adelante.

A mis amigos de la carrera, que han compartido este viaje conmigo, muchas gracias por estar a mi lado en cada etapa, por las risas, el apoyo y las palabras de aliento.

Especialmente, quiero agradecer a la Universidad, mi compañera durante estos años, y a los profesores que me han inspirado a superarme. Quiero destacar mi profunda gratitud hacia mi tutor, José Vicente Álvarez, cuya guía y dedicación han sido esenciales para que todo esto sea posible. Gracias de corazón.

Resumen

Internet ha transformado significativamente la forma en que realizamos nuestras compras, ofreciendo acceso a una vasta gama de productos y opciones al instante. Sin embargo, esta abundancia puede resultar abrumadora, dificultando la elección del mejor producto al mejor precio. Inspirado en motores de búsqueda de viajes como Skyscanner, surge la idea de crear un comparador de productos que permita a los usuarios encontrar y comparar precios, características y condiciones de venta de diversos productos en diferentes tiendas online.

El proyecto SMARTCART tiene como objetivo desarrollar una aplicación web que sirva como comparador de productos vendidos online. Esta herramienta está diseñada para simplificar y agilizar el proceso de compra, proporcionando una plataforma centralizada con información detallada y actualizada en tiempo real sobre productos de diversas categorías. Utiliza técnicas de web scraping para recolectar datos de distintos sitios web y una base de datos no relacional (MongoDB) para gestionar y consultar la información de manera eficiente.

El proyecto sigue una metodología de desarrollo iterativo en cascada, dividida en fases de análisis, diseño, implementación, pruebas y mantenimiento. A lo largo de estas fases, se asegura la calidad y adecuación a los requisitos definidos, así como la satisfacción del objetivo principal: mejorar la experiencia de compra online para los usuarios.

Además, se destaca la importancia de los proveedores de datos y la infraestructura tecnológica para el funcionamiento de SMARTCART, y se abordan las posibles limitaciones como la dependencia de políticas de sitios web externos y el cumplimiento normativo.

-**Palabras clave**: web scraping, comparador de productos, comercio electrónico, e-commerce, optimización de compras.

Abstract

This project aims to create a tool that simplifies and streamlines the online shopping process by allowing users to quickly and easily compare products available on different sales sites. Inspired by the price comparison model of "Skyscanner" for flights and hotels, SMARTCART intends to offer a centralized platform where users can find detailed information about the products they want to buy and compare prices, features, and sales conditions across different online stores.

The tool uses web scraping techniques to obtain updated and accurate data about products and prices from multiple websites, utilizing a non-relational database like MongoDB to manage the collected information. The project was developed using an iterative waterfall methodology, enabling iterations within each phase to continuously improve the product and ensure its quality and compliance with the defined requirements.

Keywords: Technology, web scraping, e-commerce, buying, selling, product searching, smart shopping.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Descripción del proyecto

1.1.- Introducción

Internet ha revolucionado la forma en que se realizan las compras, proporcionando acceso a una amplia variedad de productos y opciones de manera rápida y eficiente. No obstante, esta abundancia de opciones puede resultar abrumadora para los consumidores, quienes a menudo enfrentan dificultades para encontrar el mejor producto al mejor precio y con las mejores garantías entre la multitud de alternativas disponibles en línea.

Este proyecto surgió a partir de la experiencia de un usuario navegando en un portal de ventas de productos online, intentando identificar el mejor producto al menor precio y con las mejores condiciones. Esta situación llevó a la inspiración de "Skyscanner", un motor de búsqueda de viajes que permite comparar precios de vuelos, hoteles y alquileres de coches en una sola página. Skyscanner filtra las mejores opciones disponibles en el mercado online en función de criterios como fechas de viaje, aerolíneas, horarios y precios. De esta observación surgió la idea de crear un comparador de todo tipo de productos, similar a cómo Skyscanner facilita la comparación de todas las opciones de viajes.

En este contexto, se detecta la necesidad de una herramienta que simplifique y agilice el proceso de compra en línea, permitiendo a los usuarios comparar rápida y fácilmente los productos disponibles en diferentes sitios de venta. El comparador de productos propuesto en este TFG busca satisfacer esta necesidad, ofreciendo una plataforma centralizada que permita a los usuarios obtener información detallada sobre los productos que desean adquirir, así como comparar precios, características y condiciones de venta en diversas tiendas en línea.

Este proyecto tiene como objetivo principal facilitar la experiencia de compra de los usuarios, al tiempo que contribuye al crecimiento y desarrollo del entorno empresarial en el que se integra. Al proporcionar una herramienta que permite a los usuarios tomar decisiones informadas y encontrar las mejores ofertas disponibles en el mercado online, se espera aumentar la satisfacción del cliente y fomentar la fidelidad a la marca, lo que resultará en un incremento de las ventas y en el éxito a largo plazo de las empresas involucradas.

1.2.- Objetivos del Proyecto

En esta sección se detallarán los objetivos generales del proyecto enfocados en el desarrollo de una aplicación de software para un comparador de productos vendidos online. Se describirá la funcionalidad básica de la aplicación y se explicarán las necesidades que se cubrirán con la aplicación desarrollada, así como su integración en el entorno empresarial correspondiente.

La funcionalidad básica de la aplicación consistirá en proporcionar a los usuarios una plataforma centralizada donde puedan buscar, comparar y tomar decisiones informadas sobre la compra de productos vendidos en diferentes sitios de venta online. Para lograr esto, la aplicación ofrecerá las siguientes características principales:

Objetivo 1	Desarrollar una aplicación web capaz de realizar scraping en múltiples sitios de proveedores de productos, recopilando datos relevantes como precios y características de productos.
Objetivo 1.1	Implementar medidas de seguridad para la administración de la aplicación sobre los datos de scraping
Objetivo 2	Desarrollar un sistema de búsqueda y comparación de productos que permita a los usuarios encontrar los mejores precios de productos específicos.
Objetivo 3	Brindar opciones de filtros dentro de una keyword introducida, ya sea por título, precio o proveedores ya buscados.
Objetivo 5	Usar una colección de tecnologías basadas en JS que incluyan: MongoDB como Sistema de Bases de Datos no relacionales, Express.js como framework para el backend, React para el frontend y Node.js como entrono de tiempo de ejecución de JavaScript para el servidor.

Tabla 1: Objetivos de Proyecto

Esta aplicación cubrirá tanto las necesidades del usuario (facilitar búsqueda y comparación de productos en línea) como las necesidades de las empresas (mejorando la experiencia de compra de los usuarios y premiando a las empresas con mejores productos con relación a la búsqueda específica de cada usuario).

1.4.- Entorno de la aplicación

El comercio electrónico ha transformado la manera en que realizamos nuestras compras, brindando acceso instantáneo a una amplia gama de productos y servicios. Sin embargo, la abundancia de opciones disponibles puede resultar abrumadora para los consumidores, quienes enfrentan el reto de encontrar el mejor producto al mejor precio y con las mejores condiciones de compra. Esto genera la necesidad de contar con herramientas que faciliten la comparación rápida y eficiente de los productos en diferentes plataformas de venta en línea.

Inspirados en esta problemática, SMARTCART surge como una solución innovadora que busca optimizar la experiencia de compra en línea al ofrecer un comparador centralizado de productos. Similar al funcionamiento de motores de búsqueda de viajes como Skyscanner, SMARTCART permitirá a los usuarios comparar precios, características y condiciones de venta de una amplia variedad de productos provenientes de múltiples proveedores en una única plataforma.

Esta plataforma no solo facilitará la experiencia de compra, sino que también contribuirá al crecimiento del entorno empresarial en el que se integre. Al ofrecer una herramienta que ayuda a los usuarios a tomar decisiones más informadas, SMARTCART busca incrementar la satisfacción del cliente y, a su vez, la fidelidad a la marca, favoreciendo el éxito comercial a largo plazo.

La aplicación está diseñada para ser utilizada tanto en ordenadores como en dispositivos móviles, garantizando una experiencia de usuario ágil y accesible. En cuanto a su desarrollo, SMARTCART estará basada en tecnología web y se implementará utilizando el stack MERN, compuesto por MongoDB, Express.js, React y Node.js. Esta combinación tecnológica asegura una plataforma escalable, rápida y eficiente, optimizada para manejar grandes volúmenes de datos y garantizar un rendimiento óptimo.

MongoDB proporcionará una base de datos flexible y escalable para gestionar los datos de los productos, Express.js facilitará la creación de APIs robustas, React permitirá crear interfaces de usuario dinámicas y reutilizables, mientras que Node.js garantizará un rendimiento elevado en el procesamiento de peticiones y conexiones. El uso de JavaScript tanto en el lado del cliente como del servidor asegura un desarrollo más coherente y eficiente, lo que convierte a SMARTCART en una solución ideal para integrar un sistema avanzado de comparación de productos dentro del comercio electrónico actual.

En resumen, SMARTCART no solo busca simplificar el proceso de compra en línea para los usuarios, sino también ofrecer una solución tecnológica moderna, confiable y accesible que impacte positivamente en el ecosistema empresarial digital.

1.5.- Tecnologías

Aquí, se describirán las tecnologías empleadas para el desarrollo software de este proyecto.

1.5.1.- MongoDB

MongoDB es una base de datos no relacional (NoSQL) que almacena datos en formato de documentos JSON. Su facilidad de uso y flexibilidad nos permite gestionar todos los datos de nuestra aplicación de manera eficiente. MongoDB es ideal para aplicaciones que requieren un manejo dinámico y escalable de grandes volúmenes de datos. En nuestro proyecto, hemos utilizado MongoDB para almacenar y consultar datos de forma rápida y eficaz, lo que ha contribuido a mantener un rendimiento óptimo y a simplificar el desarrollo del backend.

Figura 1: Logo MongoDB

1.5.2.- Express

Express es un framework para Node.js que facilita la creación de APIs. Se utiliza para crear el backend de aplicaciones. En nuestro proyecto ha sido implementado para el manejo de rutas y solicitudes del servidor de manera eficiente, permitiendo una comunicación fluida entre el frontend y la base de datos.

Figura 2: Logo Express

1.5.3.- React

React es una biblioteca de JavaScript diseñada para construir interfaces de usuario. Es especialmente útil para aplicaciones web, ya que permite crear componentes reutilizables y mantener una arquitectura de código modular y eficiente. En nuestro proyecto, hemos utilizado React para el desarrollo del frontend de nuestra aplicación, lo que nos ha permitido construir una interfaz de usuario dinámica y reactiva.

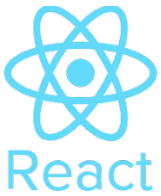


Figura 3: Logo React

1.5.4.- Node.js

Node.js es un entorno de ejecución para JavaScript. Permite construir aplicaciones web rápidas y escalables. Se usa para desarrollar el backend manejando solicitudes del servidor de manera eficiente facilitando la comunicación entre el frontend y la base de datos. Gracias a su

arquitectura basada en eventos y su modelo de E/S no bloqueante, Node.js es ideal para aplicaciones en tiempo real y de alta demanda.

Figura 4: Logo Node.js

1.5.5.- Puppeteer

Puppeteer es una biblioteca de Node.js diseñadas para automatizar navegadores web mediante código. Se utiliza para realizar pruebas, extracción de datos (web scraping), generar capturas de pantalla, y automatizar tareas repetitivas en aplicaciones web. Funciona controlando instancias de Chrome, y permite interactuar con páginas web simulando acciones de un usuario real, como hacer clic, escribir, y navegar.

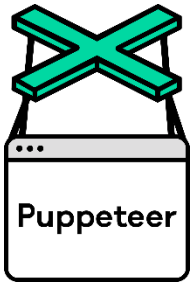


Figura 5: Logo Puppeteer

1.5.6.- Axios

Axios es una biblioteca de JavaScript que se utiliza para hacer solicitudes HTTP desde el navegador o desde un entorno de servidor, como Node.js. Facilita la comunicación con APIs y servidores externos, permitiendo enviar y recibir datos a través de métodos como GET, POST, PUT, DELETE, entre otros. Axios es conocido por su simplicidad, ya que utiliza promesas (Promises) para manejar las respuestas asíncronas, lo que hace que el código sea más legible y fácil de mantener.

Figura 6: Logo Axios

1.5.7.- Visual Studio

Visual Studio es un entorno de desarrollo integrado (IDE) para programar. Se ha tomado la decisión de usar este entorno para la realización del código del proyecto ya que es muy completo y te permite implementar varios lenguajes de programación. Además, te permite añadir de una forma muy sencilla el control de versiones de GitHub facilitando la creación de la aplicación software.

Figura 7: Logo Visual Studio Code

1.5.8.- Microsoft Word

Microsoft Word es un programa utilizado para crear y editar documentos de texto. Su uso ha sido fundamental para la elaboración de esta documentación. Con sus diversas herramientas de edición y formato, Microsoft Word nos ha permitido organizar y presentar la información de manera clara y profesional.

Figura 8: Logo Word

1.5.10.- GitHub

GitHub es una plataforma en la nube que emplea un sistema de control de versiones, permitiendo a los desarrolladores compartir y colaborar en su código de manera eficiente. Utiliza Git para hacer un seguimiento detallado de los cambios realizados en el código. Esta plataforma proporciona herramientas para la revisión de código, gestión de proyectos y automatización de flujos de trabajo, haciendo que el desarrollo sea más organizado y productivo. En nuestro proyecto, utilizamos GitHub para llevar un control exhaustivo de las versiones del proyecto a medida que evolucionaba. Cada cambio realizado en el código se registraba con un commit, permitiendo un historial claro y detallado de las modificaciones.

Figura 9: Logo GitHub

1.5.11.- Npm

Npm es un gestor de paquetes para JavaScript. Permite instalar librerías y herramientas de una manera muy sencilla. Se utiliza principalmente para la instalación de dependencias de Node.js, lo que ayuda a mantener y actualizar las librerías necesarias para el desarrollo y ejecución de aplicaciones JavaScript en el entorno de servidor. Además, npm ofrece funcionalidades para la gestión de versiones y la automatización de tareas, haciendo el desarrollo más eficiente y organizado.

Figura 10: Logo NPM

1.5.11.- Postman

Postman es una herramienta que se utiliza principalmente para desarrollar, probar y documentar APIs. Permite enviar solicitudes HTTP a los endpoints de una API, obtener respuestas y verificar que funcionen correctamente. Además, es una herramienta muy útil para simular distintos tipos de peticiones (GET, POST, PUT, DELETE, etc.), lo que facilita las pruebas de desarrollo y la depuración de errores en las APIs.

Figura 11: Logo Postman

1.6.- Estructura de la documentación

En este apartado se explica cómo está estructurada la documentación en diferentes capítulos. Cada capítulo, a su vez, se divide en múltiples secciones. A continuación, se proporciona una descripción general del contenido tratado en cada una de ellas.

1. **Capítulo 1: Descripción del proyecto:** En esta sección se definen los aspectos generales del proyecto a modo de introducción, proporcionando el contexto de la plataforma, una motivación detallada sobre el origen del proyecto, los objetivos, el entorno tecnológico y las tecnologías.
2. **Capítulo 2: Estado del arte:** En este capítulo se realiza un análisis de las diferentes plataformas que comparten características similares a e-Change, permitiendo así un estudio comparativo de sus funcionalidades para determinar su efectividad potencial.
3. **Capítulo 3: Planificación, estimación y presupuesto:** En este apartado se explica la metodología empleada durante el desarrollo del proyecto, además de una planificación temporal y una estimación de costes según el método Albrecht. Por último, se presenta un desglose de los diferentes costes involucrados en la realización del proyecto.
4. **Capítulo 4: Análisis del sistema:** Este capítulo incluye los requisitos propuestos por el usuario para cada funcionalidad del sistema. También se describe a los diferentes actores que interactúan con él y las diversas acciones que el usuario puede realizar mediante los casos de uso y sus especificaciones.

5. **Capítulo 5: Diseño:** En esta sección se detallan las diferentes arquitecturas del sistema, incluyendo la arquitectura lógica, que define el correcto funcionamiento de la plataforma, y la arquitectura física, con sus diagramas correspondientes. Además, se describe el modelo E-R con su diccionario de datos, el modelado de datos, los diagramas de secuencia y, por último, el diseño de la interfaz.
6. **Capítulo 6: Implementación:** En este apartado se detalla la organización del código, incluyendo la estructura de carpetas y archivos, así como los modelos utilizados.
7. **Capítulo 7: Pruebas:** En este capítulo se explican las diferentes pruebas realizadas para verificar que la plataforma cumple con todos los requisitos y funciona correctamente.
8. **Capítulo 8: Manuales:** Esta sección incluye las guías y manuales de usuario que detallan el proceso de instalación, así como las instrucciones para aprender y utilizar todos los aspectos esenciales de la aplicación desarrollada.
9. **Capítulo 9: Conclusiones:** En este apartado se presentan las diversas conclusiones obtenidas tras la finalización de la plataforma, así como una serie de posibles mejoras futuras que podrían implementarse más adelante.
10. **Capítulo 10: Webgrafía:** Este capítulo recopila todas las fuentes de información consultadas para la elaboración de la documentación.

Capítulo 2
Estado del arte

En este capítulo, se realiza un estudio de las plataformas web con características similares a SMARTCART, con el objetivo de analizar las funcionalidades y servicios que proporcionan en relación con la comparación de productos y la optimización de la experiencia de compra online. Se examinarán las principales plataformas que permiten comparar precios de productos en tiempo real en múltiples proveedores.

A continuación, se presentan algunas aplicaciones y plataformas relacionadas con la **comparación de precios en línea**:

2.1.- Google Shopping

Google Shopping es una herramienta de comparación de precios integrada en el motor de búsqueda de Google. Permite a los usuarios buscar productos y comparar precios ofrecidos por distintos vendedores en línea. Google Shopping recopila información sobre productos y precios a través de la indexación de sitios de comercio electrónico o mediante la integración de estos con Google Merchant Center.

Sus funcionalidades son las siguientes:

1. Búsqueda de Productos:
    a. Los usuarios pueden buscar productos específicos utilizando palabras clave. Google Shopping muestra resultados de varios vendedores con una variedad de precios y opciones.
2. Comparación de Precios
    a. Muestra una lista de vendedores con sus respectivos precios, permitiendo a los usuarios elegir la mejor oferta.
3. Filtros de Búsqueda:
    a. Los usuarios pueden filtrar los resultados según precio, marca, tipo de envío, y otras características.
4. Reseñas de Productos:
    a. Incluye valoraciones y reseñas de productos, ayudando a los compradores a tomar decisiones informadas.
5. Información de Tiendas y Envíos:
    a. Muestra detalles sobre el envío, las políticas de devolución y la calificación del vendedor.

Figura 12 Logo Google Shopping

2.2.- Idealo

Idealo es una plataforma de comparación de precios ampliamente utilizada en Europa. Ofrece una herramienta que permite a los usuarios comparar los precios de una amplia gama de productos, desde tecnología y electrodomésticos hasta moda y alimentos.

Sus funcionalidades son las siguientes:

1. Comparación de Precios:
    a. Los usuarios pueden comparar productos de múltiples tiendas en línea, permitiendo seleccionar la oferta más conveniente.
2. Historial de Precios:
    a. Idealo ofrece un seguimiento del historial de precios, mostrando cómo han fluctuado a lo largo del tiempo para ayudar a los usuarios a identificar el mejor momento para comprar
3. Alerta de Precios:
    a. Los usuarios pueden establecer alertas para recibir notificaciones cuando el precio de un producto baja
4. Valoraciones de Productos:
    a. Los usuarios Los usuarios pueden acceder a reseñas y valoraciones tanto de productos como de tiendas, mejorando la confianza en las compras.
5. Aplicación Móvil:
    a. Ofrece una aplicación móvil que facilita la comparación de precios desde cualquier dispositivo, mejorando la accesibilidad.

Figura 13: Logo Idealo

2.3.- CamelCamelCamel

CamelCamelCamel es una plataforma de comparación de precios centrada en Amazon. Se especializa en ofrecer información sobre la evolución de los precios de productos en esta tienda en línea y permite a los usuarios comparar precios actuales con el historial de precios anteriores.

Sus funcionalidades son las siguientes:

1. Seguimiento de Precios:
    a. Los usuarios pueden ver gráficos históricos que muestran la evolución de los precios de los productos vendidos en Amazon.
2. Alerta de Precios:
    a. CamelCamelCamel permite a los usuarios crear alertas de precios para que se les notifique cuando un producto baja de precio.
3. Extensiones de Navegador:
    a. Ofrece extensiones para navegadores que permiten acceder rápidamente a la información de precios y realizar comparaciones mientras se navega por Amazon.
4. Filtración de Precios:
    a. Filtra los precios por categorías, tiendas específicas dentro de Amazon, y estado de los productos (nuevos, usados, etc.).

Figura 14: Logo CamelCamelCamel

2.4.- Skyscanner (en el contexto de productos)

Skyscanner es principalmente una plataforma de comparación de precios en el sector de viajes, pero su modelo es relevante como inspiración para SMARTCART. Permite a los usuarios comparar vuelos, hoteles y alquileres de coches en tiempo real, buscando la mejor oferta disponible en una amplia variedad de proveedores.

Sus funcionalidades son las siguientes:

1. Búsqueda y Comparación:
    a. Busca en tiempo real entre numerosos proveedores de viajes para encontrar la oferta más barata o conveniente.
2. Filtros Avanzados:
    a. Ofrece filtros de búsqueda específicos, como fechas, escalas, aerolíneas, precios y más, para ayudar a los usuarios a personalizar sus búsquedas.
3. Alertas de Precios:
    a. Los usuarios pueden configurar alertas para recibir notificaciones cuando el precio de un vuelo o servicio cambia.
4. Multiplataforma:
    a. Disponible tanto como aplicación móvil como en su versión web, lo que permite el acceso desde cualquier dispositivo.

Figura 15: Logo Skyscanner

2.4.- Comparativa

Después de analizar las diferentes plataformas de comparación de precios, es posible realizar un análisis comparativo entre estas y SMARTCART para identificar las fortalezas y oportunidades de mejora de la nueva aplicación.

	SMARTCART	Google Shopping	Idealo	CamelCamelCamel	Skyscanner
Comparación de precios en tiempo real	Sí	Sí	Sí	No	Sí
Historial de precios	Sí	No	Sí	Sí	No
Alertas de precios	No	No	Sí	Sí	Sí
Búsqueda por filtros avanzados	Sí	Sí	Sí	No	Sí
Extensión de navegador	No	No	No	Sí	No
Información de tiendas y políticas	No	Sí	Sí	No	No

Tabla 2: Comparativa Aplicaciones Similares

En conclusión, tras analizar diferentes plataformas de comparación de precios, se observa que SMARTCART ofrece funcionalidades clave como la comparación de precios en tiempo real, un historial de precios y búsqueda con filtros avanzados, lo cual la posiciona competitivamente frente a otras aplicaciones consolidadas como Idealo. A pesar de no contar con características como las alertas de precios o una extensión de navegador, que sí ofrece Idealo, SMARTCART sigue siendo una opción sólida para los usuarios que buscan una herramienta ágil y eficiente. Su capacidad para mejorar radica en incorporar estas funciones adicionales y así aumentar su competitividad en el mercado.

Capítulo 3
Planificación, estimación y presupuesto

3.1.- Metodología de Trabajo

En este apartado se explica de forma detallada, la metodología que se ha llevado cabo para el desarrollo de este proyecto de fin de grado.
La metodología seleccionada para desarrollar esta aplicación ha sido la metodología de desarrollo iterativa incremental.
La metodología Iterativa Incremental es un enfoque de desarrollo de software en el que el sistema se construye de manera progresiva, mediante la realización de múltiples ciclos o 'iteraciones'. Cada iteración incluye planificación, diseño, desarrollo y pruebas, pero a pequeña escala, y produce una versión parcial o un "incremento" del sistema.

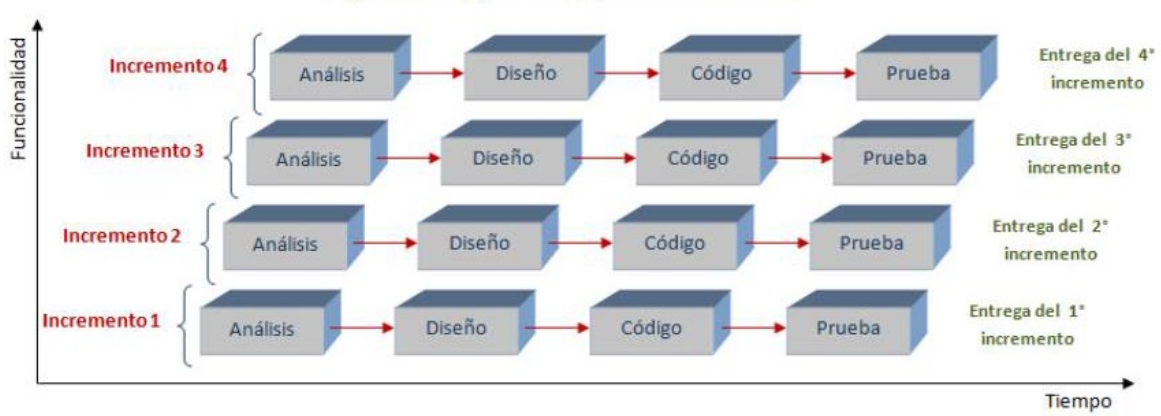


Figura 16: Esquema metodología Iterativa Incremental

El proceso de esta metodología, como su propio nombre indica, se caracteriza por dos aspectos clave:

1. Iterativo: Repetición de ciclos de desarrollo. A través de múltiples iteraciones, se revisan, mejoran y ajustan partes del sistema. Si surgen problemas o nuevas ideas, estos pueden ser corregidos en ciclos posteriores.
2. Incremental: El producto final no se construye todo de una vez, sino que se va construyendo gradualmente en partes (o incrementos). Cada iteración aporta una parte funcional y operativa del sistema, que se integra con las anteriores.

El flujo general de esta metodología es:

Paso 1: Identificación de Requisitos: Se identifican los requisitos generales del proyecto, sin entrar en todos los detalles. Estos se desglosan en tareas que se abordarán en las diferentes iteraciones. No es necesario definir todo desde el principio, ya que los detalles pueden ajustarse durante el proceso.

Paso 2: Planificación de Iteración: Para cada iteración, se selecciona un subconjunto de funcionalidades o partes del sistema que se van a desarrollar. Este subconjunto se planifica de manera detallada: qué se va a hacer, qué tareas y objetivos deben completarse, y los recursos necesarios.

Paso 3: Diseño: Se diseña la solución de las funcionalidades o módulos a implementar en la iteración. Este diseño puede incluir diagramas, modelado de base de datos, estructura del código, etc.

Paso 4: Desarrollo: Se escribe el código correspondiente a las funcionalidades planeadas. Los desarrolladores implementan el software según el diseño acordado.

Paso 5: Pruebas: Una vez implementado el código, se prueban las funcionalidades desarrolladas. Estas pruebas aseguran que las nuevas características funcionen correctamente y que el sistema en su conjunto no se vea afectado por los nuevos incrementos.

Paso 6: Revisión y Retroalimentación: Al final de cada iteración, se revisa el progreso y se recibe retroalimentación de los usuarios o del equipo. Esta revisión ayuda a ajustar los requisitos para las siguientes iteraciones, mejorando el enfoque y corrigiendo errores o añadiendo mejoras según las necesidades identificadas.

Paso 7: Entrega del Incremento: Después de cada iteración, se entrega un incremento funcional del sistema. Este puede ser utilizado por los usuarios finales o puede ser simplemente una mejora interna en la funcionalidad.

Ventajas e Inconvenientes

La metodología iterativa incremental permite ajustar requisitos o cambiar funcionalidades a medida que se avanza en el desarrollo, que los usuarios pueden revisar los incrementos de manera frecuente, lo que permite detectar problemas o necesidades de cambio a tiempo, y al desarrollar el sistema en partes funcionales, se reduce el riesgo de fallos graves y errores críticos en el producto final.

En este proyecto, la metodología en cascada resulta adecuada, ya que los requisitos son claros y estables desde el inicio, lo que permite un análisis exhaustivo y un desarrollo estructurado y bien organizado.

3.2.- Planificación temporal

En esta sección de la documentación se llevará a cabo una medición del esfuerzo necesario para la realización del proyecto. Con los resultados obtenidos se realizará un Diagrama de Gantt para ver gráficamente la duración del proyecto.

3.2.1.- Estimación del Esfuerzo

Para la estimación del esfuerzo se utilizará el método de puntos de función, conocido como el Método de Albrecht. Este método mide la funcionalidad proporcionada por el software desde la perspectiva del usuario, evaluando el tamaño del software en términos de puntos de función. Considera entradas, salidas, consultas, archivos internos y archivos externos para proporcionar una estimación precisa del esfuerzo requerido. El proceso se dividirá en tres fases:

1. **Identificación:** Reconocer y listar todos los elementos funcionales del sistema.
2. **Cálculo de los Puntos de Función No Ajustados:** Asignar valores a cada elemento funcional identificado.
3. **Ajuste de los Puntos de Función:** Aplicar factores de ajuste para reflejar las características específicas del proyecto y obtener una estimación final precisa.

Para la fase de identificación, necesitaremos listar todos los elementos funcionales del sistema en base a estas directrices:

1. **Entradas externas (EE):** Datos o controladores introducidos al sistema por el usuario.
2. **Salidas externas (SE):** Información generada por el sistema y enviada al usuario.
3. **Consultas externas (CE):** Entradas que requieren una respuesta del sistema.
4. **Archivos lógicos internos (ALI):** Conjuntos de datos mantenidos y utilizados internamente por el sistema.
5. **Archivos de interfaz externos (AIE):** Conjuntos de datos utilizados por el sistema, pero mantenidos por otro sistema externo.

Para comenzar con el método, primero se deben clasificar los elementos de cada categoría según su complejidad. Se basa en el número de Tipos de Datos Elementales (TED) y el número de Tipos de Registros (TR).

1. **Tipos de Datos Elementales (TED):** Se refiere a un campo único, no repetitivo y comprensible para el usuario en un Archivo Lógico Interno (ALI) o en un Archivo de Interfaz Externo (AIE).
2. **Tipos de Registros (TR):** Es un subgrupo de datos reconocibles para el usuario dentro de un ALI o AIE.

Hay varias tablas que nos ayudan a determinar el grado de complejidad de cada elemento, dependiendo de la cantidad de TED y TR que contengan. En este caso, se eligen:

ALI o AIE	1 a 19 TED	20 a 50 TED	>51 TED
1 TR	Baja	Baja	Media
2 a 5 TR	Baja	Media	Alta
6 o más TR	Media	Alta	Alta

Tabla 3: Grado de Complejidad ALI y AIE

EE y CE	1 a 4 TED	5 a 15 TED	>15 TED
0-1 TR accedidos	Baja	Baja	Media
2 TR accedidos	Baja	Media	Alta
> 2 TR accedidos	Media	Alta	Alta

Tabla 4: Grado de Complejidad para EE y CE

SE	1 a 5 TED	6 a 19 TED	>19 TED
0-1 FR accedidos	Baja	Baja	Media
2-3 FR accedidos	Baja	Media	Alta
> 3 FR accedidos	Media	Alta	Alta

Tabla 5: Grado de Complejidad para SE

El siguiente paso será analizar cada uno de los elementos del proyecto:

Entradas de Usuario (EE)	
Descripción	**Grado de Complejidad**
Introducir Keyword Búsqueda	Baja
Seleccionar Proveedores	Baja
Ajustar Filtros de Búsqueda	Media
Introducir Datos Nuevo Proveedor	Baja
Seleccionar Proveedor a Probar	Baja
Introducir filtrado por nombre	Baja

Tabla 6: Entradas de Usuario (EE)

Salidas Externas (SE)	
Descripción	**Grado de Complejidad**
Resultados de la Búsqueda Directa	Alta
Listado de Proveedores Disponibles	Baja
Pop-up gestión de proveedores	Media

Tabla 7: Salidas Externas(SE)

Consultas Externas (CE)	
Descripción	**Grado de Complejidad**
Búsqueda Básica de Productos	Baja
Listado de Proveedores	Baja
Gestionar Datos de un Proveedor	Bajo
Visualizar Datos Proveedores	Bajo

Tabla 8: Consultas Externas (CE)

Archivos Lógicos Internos (ALI)	
Descripción	**Grado de Complejidad**
Archivo de Usuarios	Baja
Archivo de Modelos (proveedores)	Medio
Archivo de Productos	Baja

Tabla 9: Archivos Lógicos Internos (ALI)

Archivos de Interfaz Externos (AIE)	
Descripción	**Grado de Complejidad**
Interfaz de Recuperación de Productos	Media

Tabla 10: Archivos de Interfaz Externos (AIE)

Una vez recopiladas las diferentes complejidades de los elementos de nuestro sistema, los resultados deberán ser trasladados a la tabla de conversión, asignando un peso a cada tipo de componente según su complejidad.

Descripción/Complejidad	**Baja**	**Media**	**Alta**
Entradas	x 3	x 4	x 6
Salidas	x 4	x 5	x 7
Consultas externas	x 3	x 4	x 6
Ficheros internos	x 7	x 10	x 15
Ficheros externos	x 5	x 7	x 10

Tabla 11: Conversión por Complejidad

Puntos de Función Sin Ajustar (PFSA)			
Elemento	**Complejidad**	**Número**	**Total por elemento**
Entradas	Baja (x3) Media(x4) Alta(x6)	5 1 0	15 4 0
Salidas	Baja (x4) Media (x5) Alta (x7)	1 1 1	4 5 7

Consultas externas	Baja (x3) Media (x4) Alta (x6)	4 0 0	12 0 0
Ficheros internos	Baja (x7) Media (x10) Alta (x15)	2 1 0	14 10 0
Ficheros externos	Baja (x5) Media (x7) Alta (x10)	0 1 0	0 7 0
Total			78

Tabla 12: Puntos de Función sin Ajustar (PFSA)

Una vez calculados los PFSA totales, será necesario realizar un ajuste. Para ello, se debe calcular mediante el Factor de Ajuste (FA). Los cálculos de la FA se basan en 14 características generales de los sistemas que miden la funcionalidad y la complejidad/influencia de la aplicación. Todas las características se clasifican en una escala de cero a cinco. Un cero indica que el factor no tiene influencia ni complejidad, y cinco sería la mayor influencia o complejidad alcanzable.

Factor de Ajuste (FA)	
Factor	**Complejidad**
1. Comunicación de datos	2
2. Funciones distribuidas	2
3. Prestaciones	3
4. Gran uso de la configuración	2
5. Velocidad de las transacciones	5
6. Entrada de datos en línea	4
7. Diseño para la eficiencia del usuario final	3
8. Actualización de datos en línea	4
9. Complejidad del proceso lógico interno de la aplicación	3
10. Reusabilidad del código por otras aplicaciones	2
11. Facilidad de instalación	2
12. Facilidad de operación	4
13. Localizaciones múltiples	2
14. Facilidad de cambios	3
Total	**41**

Tabla 13: Factor de Ajuste (FA)

Una vez que se han asignado los valores a cada uno de los factores, se procede a calcular el Factor de Ajuste (FA). Para ello, utilizamos la siguiente ecuación:

$$FA = 0.65 + 0.01 * \sum_{i=1}^{14} complejidad\ i = 0.65 + 0.01 * 0.41 = 1.06\ FA$$

A continuación, se calculan los Puntos de Función Ajustados (PFA) multiplicando los Puntos de Función Sin Ajustar por el Factor de Ajuste.

$$PFA = PFSA \cdot FA = 78 \cdot 1{,}06 = 82{,}68$$

Para finalizar, se calcula el tiempo total que se estima que va a durar la realización del proyecto, en mi caso he asignado que punto de función ajustado equivale a 6 horas de trabajo.

$$Tiempo\ estimado = PFA \cdot Nº Horas/PFA = 82{,}68\ \cdot 6\ horas = 496{,}08\ horas$$

El tiempo estimado que tendrá la realización del proyecto es de 477 horas.

Para calcular cuánto sería en meses tomamos con que la jornada laboral es, de lunes a viernes y se trabaja en torno a 8 horas diarias durante 21 días laborables.

$$21\ días/mes * 8 horas/día = 168\ horas/mes$$

$$Tiempo\ en\ meses = \frac{496{,}08\ horas}{168\ horas/mes} = 2.95\ meses$$

Este sería el tiempo ideal, sin contar con los incidentes y contratiempos que el proyecto ha sufrido, sumandole el horario variable que se ha tenido en la realización del proyecto…

3.2.2.- Presupuesto económico

Antes de iniciar el proyecto, es fundamental estimar los costos de desarrollo. Para realizar esta estimación, se considerarán las herramientas de hardware y software utilizadas, así como los factores de impacto relacionados con la duración del proyecto y los gastos derivados del personal. Esta evaluación se basará en la estimación de horas planificadas y en los roles específicos asignados a cada tarea.

El costo total del proyecto se dividirá en cuatro categorías principales: hardware, licencias de software, otros costes y costos de personal.

Tabla Coste Hardware			
Componentes	**Coste (€)**	**Uso(%)**	**Total (€)**
Ordenador portátil	1000,00€	5%	50,00€
Ratón	30,00€	2%	0,60€
Monitor	160,00€	5%	8,00€

Teclado	40,00€	2%	0,80€
Total			**59,40€**

Tabla 14: Costes Hardware

Tabla Coste Software			
Licencia	**Coste (€)**	**Uso(%)**	**Total (€)**
MongoDB	0,00€		0,00€
Visual Studio	0,00€		0,00€
Web Scraping	0,00€		0,00€
GitHub	0,00€		0,00€
Google Chrome	0,00€		0,00€
Windows 11	35,00€	10%	3,50€
Word	0,00€		0,00€
Total			**3,50€**

Tabla 15: Costes Software

En este apartado, se calcularán otros costes que necesitamos contemplar en nuestro proyecto.

Tabla Otros Costes				
Servicio	**Coste (€)/mes**	**Tiempo (mes)**	**Porcentaje Uso**	**Total (€)**
Conexión a internet	60,00€	5	20%	70,00€
Costes eléctricos	25,00€	5	60%	75,00€
Total				**145,00€**

Tabla 16: Otros Costes Relevantes

Para calcular los costos asociados al personal del proyecto, es esencial considerar los salarios anuales de los roles asignados. En este caso, se han identificado dos categorías: Desarrollador Junior y Analista. Según el Colegio Oficial de Ingenieros en Informática de España y diversos estudios salariales del sector tecnológico, el salario promedio anual de un Desarrollador Junior es de alrededor de 22.450 euros, mientras que el de un Analista se sitúa en torno a los 35.000 euros. Estos datos permiten una estimación precisa de los costos del proyecto, teniendo en cuenta el tiempo y los recursos humanos necesarios.

De las 477 horas estimadas para el proyecto, procederemos a dividir las horas asignadas según cada rol para determinar la carga de trabajo correspondiente a cada persona.

1. **Desarrollador Junior:** 60% de 496,08 horas = 297,6 horas
2. **Analista:** 40% de 496,08 horas = 198,43 horas

Para el Cálculo del Costo por Hora:

$$Desarrollador\ Junior = \frac{22.450€}{12\ meses * 173{,}3\ horas/mes} = 10{,}79€/hora$$

$$Analista = \frac{35.000€}{12\ meses * 173{,}3\ horas/mes} = 16{,}83€/hora$$

Tabla Coste Recursos Humanos			
Rol	**Horas (h)**	**Coste/Hora(€)**	**Total (€)**
Desarrollador	297,65	10,79€	3.211,64€
Analista	198,43	16,83€	3.339,58€
Total			**6.551,22 €**

Tabla 17: Costes Recursos Humanos

Tras haber calculado todos los costes por separado, realizamos una tabla que recoja el total de todos los costes.

Tabla Coste Total	
Hardware	59,40€
Software	3,50€
Otros Costes	145,00€
Recursos Humanos	6.551,22 €
Total	**6.759,11€**

Tabla 18: Coste Total Estimado

Nombre de tarea	Comienzo	Fin
Fase Inicial	mar 09/04/24	lun 22/04/24
Fase de análisis	lun 22/04/24	mié 01/05/24
Fase de Diseño	mié 01/05/24	jue 02/05/24
Fase de Implementación	jue 02/05/24	mié 17/07/24
Fase de Pruebas	mié 17/07/24	dom 21/07/24

Figura 17: Estimación Fases Proyecto

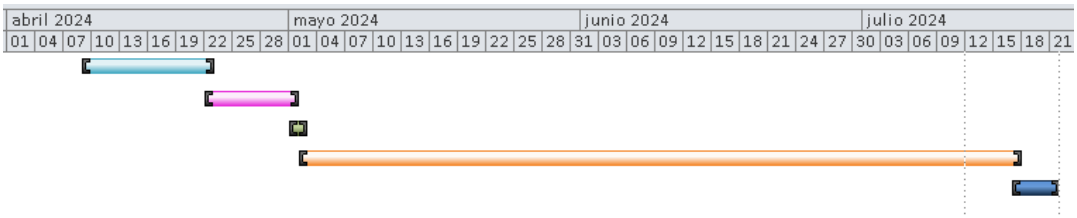


Figura 18: Diagrama Gantt Estimación

3.2.3.- Balance final

La planificación inicial del proyecto no se ha cumplido por varias razones, lo que ha resultado en un aumento significante de la duración real del proyecto.

El proyecto comenzó el 9 de abril de 2024.El proyecto seguía la estimación realizada, sin embargo, debido a numerosos incidentes y contratiempos :realización de la asignatura de prácticas con prácticas extracurriculares hasta mediados de junio; cambios de la planificación inicial y enfoque que tendría la aplicación, como por ejemplo el cambio de uso de APIs de productos a web scraping como solución adoptada; retraso en la realización de tareas prolongándolas más de lo debido ; numerosas vacaciones programadas anteriormente al comienzo del proyecto que han resultado en días que no se ha realizado nada, como por ejemplo la primera quincena de Julio, todo el mes de agosto hasta el 18, y posteriormente la primera semana de Septiembre.

Debido a estos contratiempos, el proyecto se ha alargado hasta finales de agosto, con fecha fin a **20 de Agosto de 2024.**

Respecto a lo estimado hay una más de 1 mes, período durante el cual se ha tenido que retrasar el proyecto por las causas mencionadas. Debido a esto se ha tenido que hacer un reajuste en el presupuesto.

Siguiendo los horarios anteriormente mencionados, se suman 80 horas adicionales al proyecto (no más debido a que los días vacacionales no se cuentan como retrasos debido a que estaban previstos en los riesgos posibles). Este incremento representa aproximadamente el 16,77% del tiempo planificado. Para calcular el presupuesto ajustado, multiplicamos nuestro presupuesto estimado de 6.759,11€ por 1,677 , lo que nos da 11.335,03€ como presupuesto real.

Nombre de tarea	Duración	Comienzo	Fin
Fase Inicial	10 días	mar 09/04/24	lun 22/04/24
Fase de análisis	8 días	lun 22/04/24	mié 01/05/24
Fase de Diseño	9 días	mié 01/05/24	dom 12/05/24
Fase de Implementación	63 días	jue 02/05/24	dom 28/07/24
Fase de Pruebas	4 días	dom 18/08/24	mié 21/08/24

Figura 19: Fases Proyecto Real

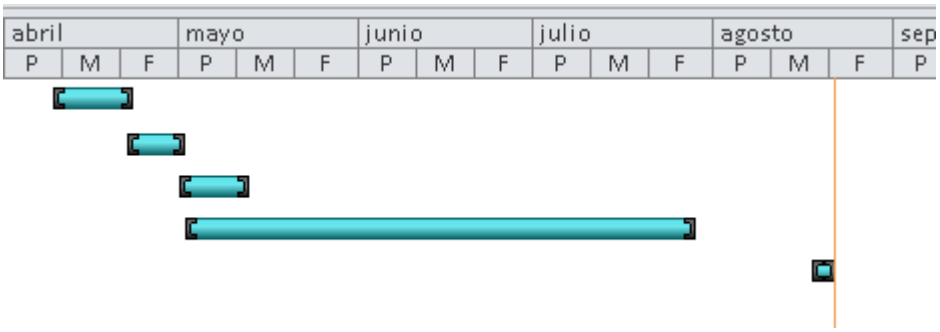


Figura 20: Diagrama de Gantt Real

Capítulo 4
Análisis del sistema

Aquí se desglosarán los distintos actores considerados en el sistema, incluyendo los requisitos que el sistema debe cumplir hasta las especificaciones y diagramas que los representan. Esta fase es prescindible para asegurarse de que el software se diseñe y construya correctamente.

4.1.- Actores del sistema

Los actores representan los diversos roles que interactúan con el sistema. En este proyecto, se han identificado los siguientes:

ID	Nombre	Descripción
AC_01	Usuario Anónimo	Este actor representa a los usuarios del sistema, que no necesitan estar logueados para usar la búsqueda básica, búsqueda directa o la búsqueda completa, seleccionar proveedores y los aplicar filtros asociados a la búsqueda.
AC_02	Administrador	Este actor representa a los usuarios administradores, que una vez están logueados, permite añadir un nuevo proveedor, probar proveedores (para activar su operatividad) y editar proveedores existentes

Tabla 19: Actores del Sistema

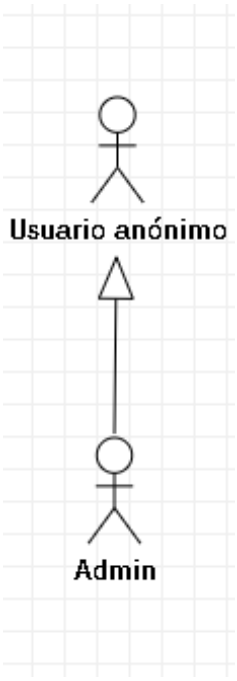


Figura 21 : Jerarquía de Actores

4.2.- Requisitos de usuario

Son todas aquellas funcionalidades y características que deberían satisfacer las necesidades del usuario final. La forma en que vamos a dividir los requisitos será por el actor que va a interactuar con la aplicación, lo dividiremos de la siguiente manera:

ID	Nombre	Descripción
CU_01	Realizar Búsqueda Básica	El usuario anónimo puede realizar una búsqueda básica de productos guardados en la plataforma de los proveedores seleccionados.
CU_02	Realizar Búsqueda Directa	El usuario anónimo puede realizar una búsqueda directa y en tiempo real (web scraping) de productos de los proveedores seleccionados.
CU_03	Realizar Búsqueda Completa	El usuario anónimo puede realizar una búsqueda directa y básica simultáneamente para mezclar ambos resultados de los proveedores seleccionados.
CU_04	Seleccionar Proveedores	El usuario anónimo puede seleccionar diferentes proveedores de productos antes de realizar la búsqueda para buscar en dichos proveedores directamente o los registros asociados guardados en la aplicación
CU_05	Aplicar Filtros de Búsqueda	El usuario anónimo puede aplicar filtros como precio, proveedores o nombre de producto para refinar la búsqueda.
CU_06	Añadir Nuevo Proveedor	El administrador puede añadir nuevos proveedores a la plataforma para ser considerados en las búsquedas de productos.
CU_07	Probar Proveedor Existente	El administrador puede realizar pruebas sobre los proveedores existentes para verificar que su operatividad es correcta.

CU_08	Editar Proveedores Existentes	El administrador puede modificar la información y configuración de proveedores no operativos ya existentes.
CU_09	Admin Log In	El administrador podrá iniciar sesión con sus credenciales (username, password)
CU_10	Visualizar Proveedores	El administrador podrá visualizar todos los proveedores guardados en la base de datos y sus datos asociados
CU_11	Eliminar Proveedores	El administrador podrá eliminar proveedores de la aplicación

Tabla 20: Requisitos de Usuario

4.3.- Requisitos funcionales

Aquí se recogerán el conjunto de requisitos funcionales de la aplicación, siendo estos todas aquellas funcionalidades y características que debe tener el sistema para satisfacer las necesidades del usuario final y cumplir con sus objetivos.

ID	Descripción de los requisitos funcionales
RF_01	El sistema debe permitir a los usuarios anónimos realizar búsquedas de productos guardados en la base de datos
RF_02	El sistema debe permitir a los usuarios anónimos buscar un producto directamente (en tiempo real) de los proveedores seleccionados.
RF_03	El sistema debe permitir a los usuarios anónimos buscar un producto directamente (en tiempo real) y junto a los datos de la base de datos (búsqueda básica), sacar una lista de productos actualizados de los proveedores seleccionados
RF_04	El sistema debe permitir a los usuarios anónimos seleccionar uno o varios proveedores para buscar productos específicos de cada uno.
RF_05	El sistema debe proporcionar a los usuarios anónimos una serie de filtros (por ejemplo, precio, nombre…) para refinar los resultados de la búsqueda.
RF_06	El sistema debe permitir al administrador añadir , editar y activar o desactivar proveedores en la base de datos.
RF_07	El sistema debe permitir al administrador realizar pruebas de los proveedores añadidos antes de que estén disponibles en las búsquedas.
RF_08	El sistema debe permitir a un administrador iniciar sesión en la sección de administrador
RF_09	El sistema debe garantizar que solo los administradores autenticados y autorizados puedan acceder a funciones administrativas sensibles, como la gestión de proveedores, mediante la implementación de medidas de seguridad avanzadas (por ejemplo, autenticación de dos factores).

Tabla 4.21: Requisitos Funcionales

4.4.- Requisitos no funcionales

Aquí se recogerán el conjunto de requisitos no funcionales de la aplicación, siendo estos todas aquellas características y condiciones que debe tener el sistema para asegurar su calidad, rendimiento y usabilidad.

4.4.1.- Usabilidad

ID	Descripción
RNF_01	El sistema debe contar con una interfaz de usuario amigable e intuitiva, que permita a los usuarios realizar búsquedas, aplicar filtros y comparar productos de manera sencilla, sin necesidad de instrucciones extensivas.
RNF_02	El sistema debe tener una interfaz de usuario responsive que se adapte automáticamente a diferentes tamaños de pantalla y resoluciones.
RNF_03	En la documentación se incluirán manuales de usuario que expliquen el funcionamiento de la aplicación.
RNF_04	El sistema debe proporcionar mensajes de error que sean fáciles de entender para el usuario.

Tabla 21: Requisitos de Usabilidad

4.4.2.- Eficiencia

ID	Descripción
RNF_05	El tiempo de respuesta del sistema debe ser inferior a 10 segundos para cualquier acción del usuario, e inferior a 25 segundos en caso de la búsqueda directa.
RNF_06	El sistema debe optimizar el uso de los recursos del servidor, asegurando que no consuma más del 70% de la CPU y RAM bajo una carga normal de operación
RNF_07	El sistema debe poder escalar horizontalmente para manejar hasta 1,000 usuarios concurrentes sin una disminución significativa en el rendimiento.

Tabla 22: Requisito de Eficiencia

4.4.3.- Mantenibilidad

ID	Descripción
RNF_08	El sistema debe garantizar en caso de fallos la consistencia de daotsdel sistema.
RNF_09	La aplicación será capaz de restaurarse en la corrección de errores.

Tabla 23: Requisitos de Mantenibilidad

4.4.4.- Seguridad

ID	Descripción
RNF_10	El sistema debe encriptar las credenciales de los administradores utilizando algoritmos de encriptación (bcrypt).

Tabla 24: Requisito de Seguridad

4.4.5.- Disponibilidad

ID	Descripción
RNF_13	El sistema deberá de tener una disponibilidad del 95% del tiempo o superior cada vez que el usuario intente acceder.

Tabla 25: Requisitos de Disponibilidad

4.4.6.- Portabilidad

ID	Descripción
RNF_14	El sistema debe ser accesible desde los principales navegadores web (Chrome, Firefox, Brave, Edge).
RNF_15	El sistema debe ser portable a diferentes entornos (Windows, Linux, macOS) sin necesidad de grandes modificaciones en el código fuente.

Tabla 26: Requisitos de Portabilidad

4.4.7.- Implementación

ID	Descripción
RNF_16	La aplicación se desarrollará utilizando el entorno de desarrollo de VS Code

Tabla 27: Requisitos de Implementación

4.5.- Requisitos de información

Aquí se recogerán el conjunto de requisitos de información de la aplicación, siendo estos todas aquellas necesidades de datos y estructuras de información que el sistema debe manejar para cumplir con sus objetivos.

ID	Descripción
RI_01	El sistema debe almacenar la información de los administradores.
RI_02	El sistema almacenará los datos de los productos, usuarios y proveedores de manera eficiente utilizando una base de datos NoSQL (MongoDB), con índices optimizados para acelerar las consultas.

RI_03	El sistema registrará todas las actividades críticas en un log (logErrores), con la hora, la fecha, el lugar donde ocurrió el error y el error en cuestión.

Tabla 28: Requisitos de Información

4.6.- Diagrama de casos de uso

En este punto, se presentan los diagramas de casos de uso que muestran las interacciones entre los diferentes actores y el sistema, destacando las principales acciones que cada tipo de usuario puede realizar.

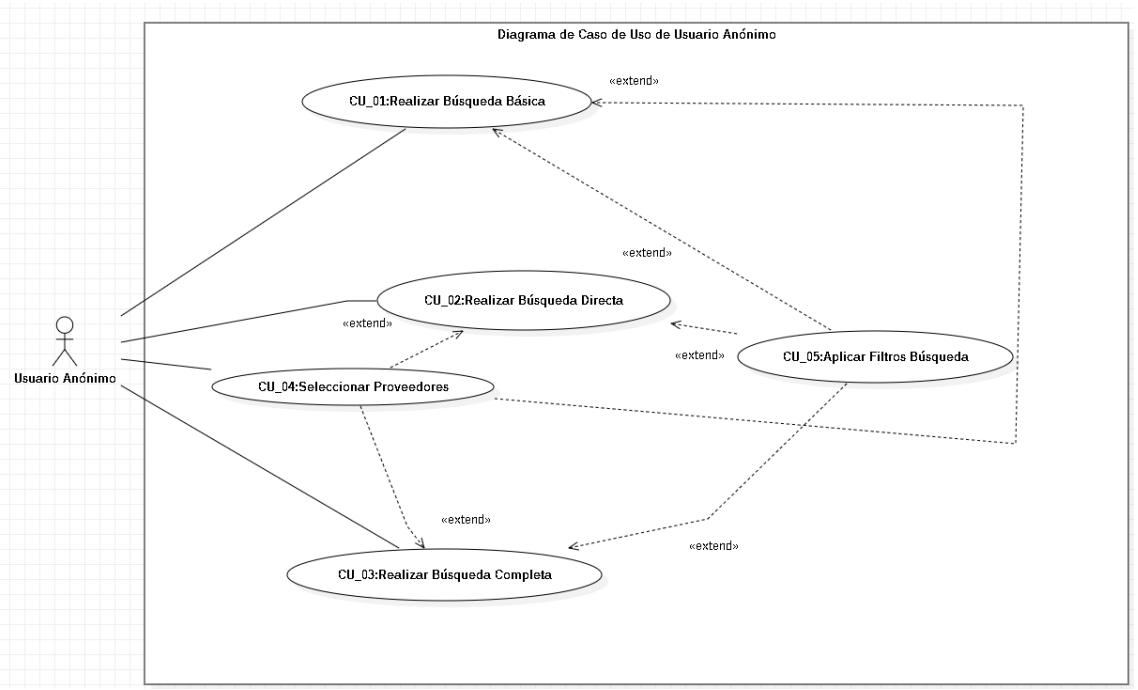


Figura 22: Diagrama de CU de Usuario Anónimo

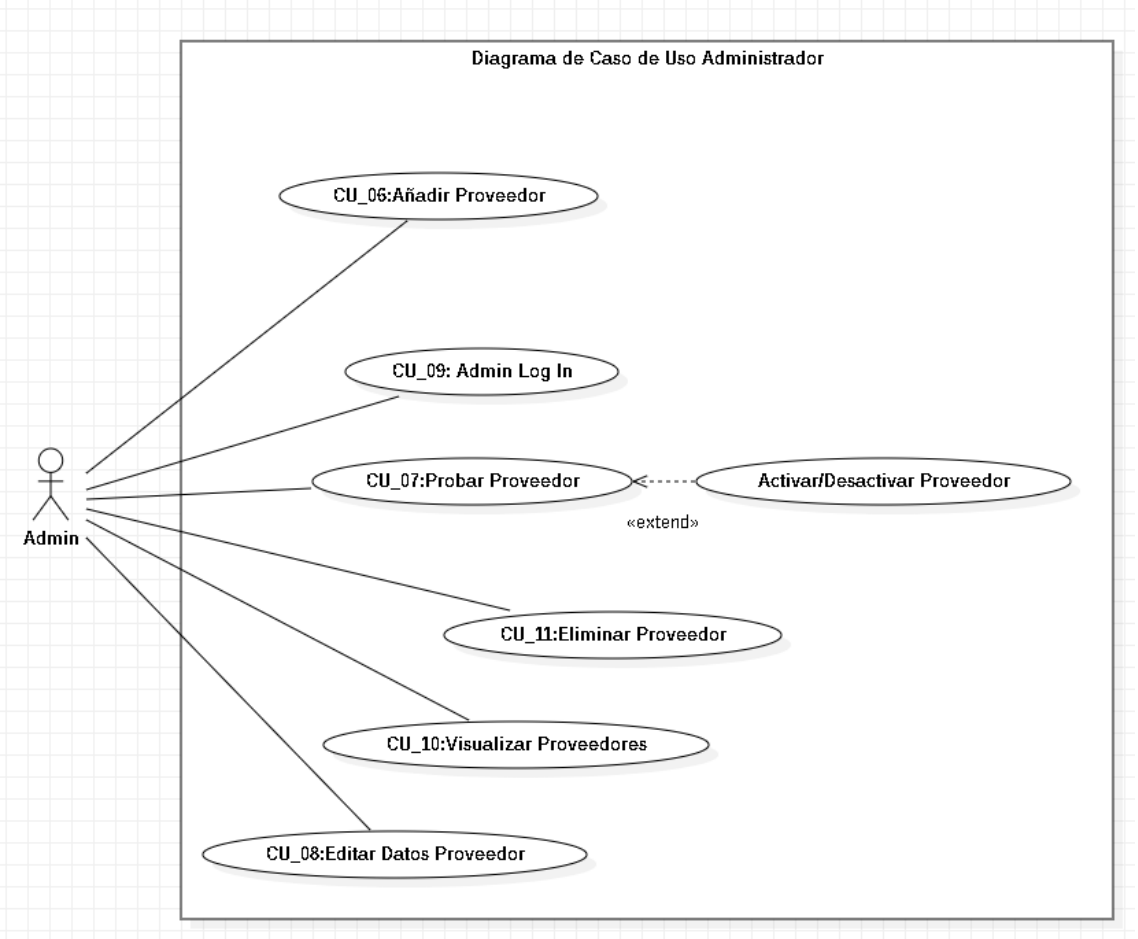


Figura 23: Diagrama de CU de Admin

4.7.- Especificación de casos de uso

CU_01	**Realizar Búsqueda Básica**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El caso de uso permite al usuario anónimo realizar una búsqueda básica de productos almacenados en la plataforma de proveedores seleccionados. La búsqueda básica se lleva a cabo dentro de los registros previamente guardados en la base de datos del sistema.
Actores	Usuario Anónimo (AC_01)
Precondiciones	1. Los proveedores deben estar preseleccionados o ser seleccionados por el usuario. 2. El sistema debe tener productos almacenados en la base de datos.
Flujo normal	1. El usuario navega hasta la página de Búsqueda Básica 2. El usuario introduce un término de búsqueda (palabra clave) en el campo correspondiente y selecciona los proveedores. 3. El usuario pulsa el botón de búsqueda. 4. El sistema busca productos relacionados con la palabra clave dentro de su base de datos, filtrando según los proveedores seleccionados.

	5. El sistema muestra al usuario una lista de productos coincidentes de los proveedores seleccionados.
Postcondiciones	El sistema muestra una lista de productos coincidentes con el término de búsqueda, disponibles a través de los proveedores seleccionados.
Excepciones	1a. Si no se encuentra ningún producto coincidente con el término de búsqueda, el sistema muestra un mensaje indicando que no hay resultados disponibles. 1b. Si no hay proveedores preseleccionados ni seleccionados, el sistema solicita al usuario que seleccione al menos un proveedor.
Importancia	Alta

Tabla 29: Especificación de CU_01

CU_02	**Realizar Búsqueda Directa**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El usuario anónimo puede realizar una búsqueda directa y en tiempo real utilizando web scraping para obtener productos de los proveedores seleccionados.
Actores	Usuario Anónimo (AC_01)
Precondiciones	Los proveedores deben estar preseleccionados o ser seleccionados por el usuario.
Flujo normal	1.El usuario selecciona la opción de búsqueda directa. 2.Introduce un término de búsqueda. 3.Selecciona los proveedores. 4.Inicia la búsqueda. 5.El sistema realiza scraping en tiempo real y muestra los resultados obtenidos.
Postcondiciones	Se muestran los productos obtenidos mediante scraping de los proveedores seleccionados.
Excepciones	2a. Error en la conexión o en el scraping: Se informa al usuario que hubo un error en la búsqueda. 2b. No hay resultados
Importancia	Alta

Tabla 30: Especificación de CU_02

CU_03	**Realizar Búsqueda Completa**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El usuario anónimo puede realizar una búsqueda que combine resultados de una búsqueda básica y una búsqueda directa simultáneamente.
Actores	Usuario Anónimo (AC_01)
Precondiciones	El sistema debe contar con productos en la base de datos y tener acceso a proveedores en tiempo real.

Flujo normal	1.El usuario selecciona la opción de búsqueda completa. 2.Introduce un término de búsqueda. 3.Selecciona los proveedores. 4.Inicia la búsqueda. 5.El sistema ejecuta la búsqueda básica y directa de manera simultánea. 6.Los resultados se muestran de manera combinada.
Postcondiciones	Se muestran productos tanto de la base de datos como de la búsqueda en tiempo real.
Excepciones	3a. Error en la conexión o scraping: Se informa al usuario que hubo un error en la búsqueda directa. 3b. No hay productos coincidentes: Se muestra un mensaje indicando que no se encontraron productos.
Importancia	Alta

Tabla 31: Especificación de CU_03

CU_04	**Seleccionar Proveedores**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El usuario anónimo puede seleccionar diferentes proveedores antes de realizar una búsqueda para decidir en qué sitios buscar productos.
Actores	Usuario Anónimo (AC_01)
Precondiciones	Deben existir proveedores en el sistema.
Flujo normal	1.El usuario accede a la selección de proveedores. 2.Selecciona los proveedores disponibles para realizar la búsqueda. 3.El sistema guarda la selección y permite realizar la búsqueda con los proveedores seleccionados.
Postcondiciones	Los proveedores seleccionados están habilitados para realizar la búsqueda.
Excepciones	4a. No hay proveedores disponibles: Se informa al usuario que no hay proveedores para seleccionar.
Importancia	Media

Tabla 32: Especificación de CU_04

CU_05	Aplicar Filtros Búsqueda
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El usuario anónimo puede aplicar filtros como precio, proveedor o nombre de producto para refinar los resultados de una búsqueda.
Actores	Usuario Anónimo (AC_01)
Precondiciones	El usuario debe haber iniciado una búsqueda previamente.
Flujo normal	1.El usuario realiza una búsqueda. 2.Selecciona los filtros disponibles (precio, proveedor, nombre, etc.). 3.El sistema ajusta los resultados de la búsqueda según los filtros aplicados.
Postcondiciones	Los resultados de la búsqueda son refinados según los filtros aplicados.
Excepciones	5a.No hay resultados después de aplicar filtros: Se muestra un mensaje indicando que no se encontraron productos.
Importancia	Media

Tabla 33 :Especificación de CU_05

CU_06	Añadir Nuevo Proveedor
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	Permite al usuario autenticarse e iniciar sesión en su cuenta.
Actores	Administrador (AC_02)
Precondiciones	El administrador debe estar logueado en el sistema.
Flujo normal	1.El administrador accede al panel de administrador. 2.Selecciona la opción de añadir nuevo proveedor. 3.Introduce los datos requeridos para el proveedor. 4.El sistema registra el nuevo proveedor.
Postcondiciones	El nuevo proveedor queda registrado y disponible para las búsquedas.
Excepciones	6a. Datos incompletos o incorrectos: Se solicita al administrador que corrija la información.
Importancia	Alta

Tabla 34: Especificación de CU_06

CU_07	Probar Proveedor Existente
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El administrador probar la operatividad de un proveedor respecto al web scraping (si, con los tags y etiquetas guardados, se consiguen recuperar los productos buscados).
Actores	Administrador (AC_02)

Precondiciones	El administrador debe estar logueado en el sistema.
Flujo normal	1.El administrador selecciona un proveedor existente 2.Inicia una prueba (llamada) para probar su operatividad 3.El sistema realiza la prueba 4.Se notifica al administrador sobre la operatividad del proveedor, y lo cambia en caso de ser contraria a la anterior a la prueba
Postcondiciones	En caso de tener una operatividad distinta a la anterior a la prueba, se cambia el estado del proveedor.
Excepciones	7a. Error en la prueba: Se informa al administrador que hubo un problema durante la prueba.
Importancia	Media

Tabla 35 : Especificación de CU_07

CU_08	**Editar Proveedor Existente**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El administrador puede modificar la información de proveedores no operativos ya existentes.
Actores	Administrador (AC_02)
Precondiciones	El administrador debe estar logueado en el sistema. El proveedor debe estar registrado en el sistema.
Flujo normal	1.El administrador accede a la lista de proveedores. 2.Selecciona el proveedor que desea modificar. 3.Realiza los cambios necesarios en la información y 4.configuración. 5.Guarda las modificaciones. 6.El sistema actualiza la información del proveedor.
Postcondiciones	El proveedor queda actualizado en la plataforma.
Excepciones	8a. Error en la modificación: Se informa al administrador del problema y se solicita que revise la información.
Importancia	Media

Tabla 36 :Especificación de CU_08

CU_09	**Log In**
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El administrador puede acceder al panel de control de administrador al iniciar sesión
Actores	Administrador (AC_02)
Precondiciones	N/E

Flujo normal	1.El administrador accede a 'Visualizar proveedores' . 2. El sistema devuelve la información de todos los proveedores
Postcondiciones	El administrador podrá ver el panel de control y se guardarán las credenciales en LocalSotrage
Excepciones	9a. Error en el Log In: Log in incorrecto: Credenciales incorrectas
Importancia	Media

Tabla 37: Especificación de CU_09

CU_10	Visualizar Proveedor
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El administrador puede visualizar todos los proveedores de la aplicación y sus datos
Actores	Administrador (AC_02)
Precondiciones	El administrador debe estar logueado
Flujo normal	1.El usuario anónimo accede al log in del admin 2.Introduce el username y password 3.Se hace una llamada a /login 4.El sistema identifica al usuario admin 5.Se accede al panel de control
Postcondiciones	El administrador podrá ver la información de todos los proveedores en una tabla
Excepciones	10a. Error de conexión con el backend o DB.
Importancia	Baja

Tabla 38: Especificación de CU_10

CU_11	Eliminar Proveedor
Versión	1.2
Autor	Fernando Scott Humanes Carchenilla
Descripción	El administrador puede eliminar un proveedor de la aplicación.
Actores	Administrador (AC_02)
Precondiciones	El administrador debe estar logueado
Flujo normal	1.El administrador accede a la sección de eliminar proveedores 2. El sistema devuelve el listado de proveedores guardados. 3.El administrador selecciona un proveedor a borrar 4. El sistema borra el proveedor mandado.
Postcondiciones	El proveedor quedará eliminado de la base de datos

Excepciones	11a. Error de conexión con el backend o DB.
Importancia	Baja

Tabla 39: Especificación de CU_11

Capítulo 5
Diseño

En esta sección del documento se presentarán los aspectos más importantes del diseño de la aplicación, incluyendo las diferentes arquitecturas utilizadas, el modelo E-R con el diccionario de datos y el modelo lógico de datos documentado.

5.1.- Arquitectura lógica

La arquitectura lógica representa la estructura conceptual del sistema, incluyendo la organización de los componentes de software y su interacción. Esta arquitectura define cómo se distribuyen las funciones y responsabilidades dentro de la aplicación, como la separación entre la capa de presentación, la capa de negocio y la capa de datos. A través de diagramas y modelos, se ilustra cómo los módulos y servicios se comunican entre sí, facilitando la comprensión del flujo de información y las dependencias entre componentes.

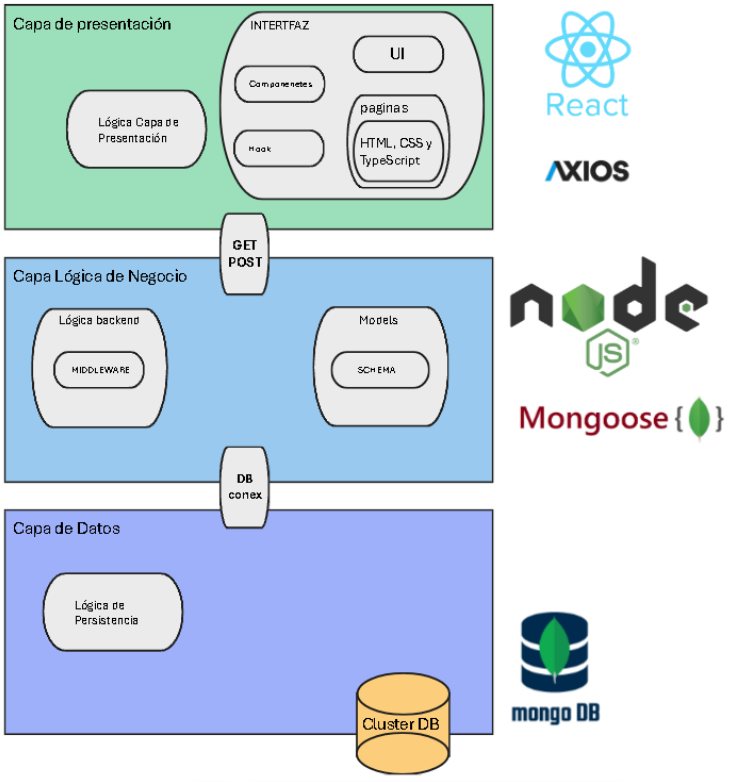


Figura 24: Diseño Lógico No Detallado

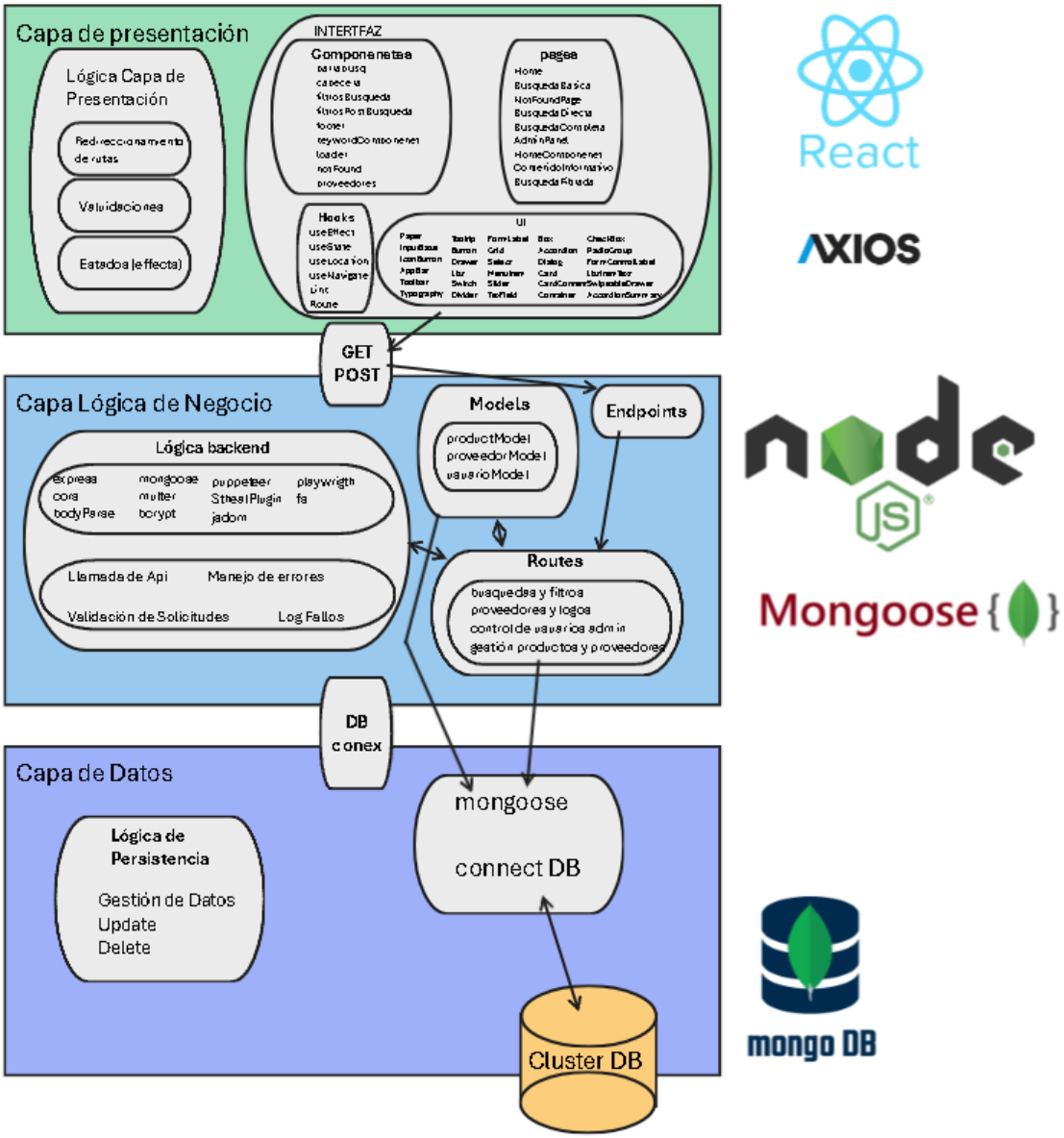


Figura 25: Diseño Lógico Detallado

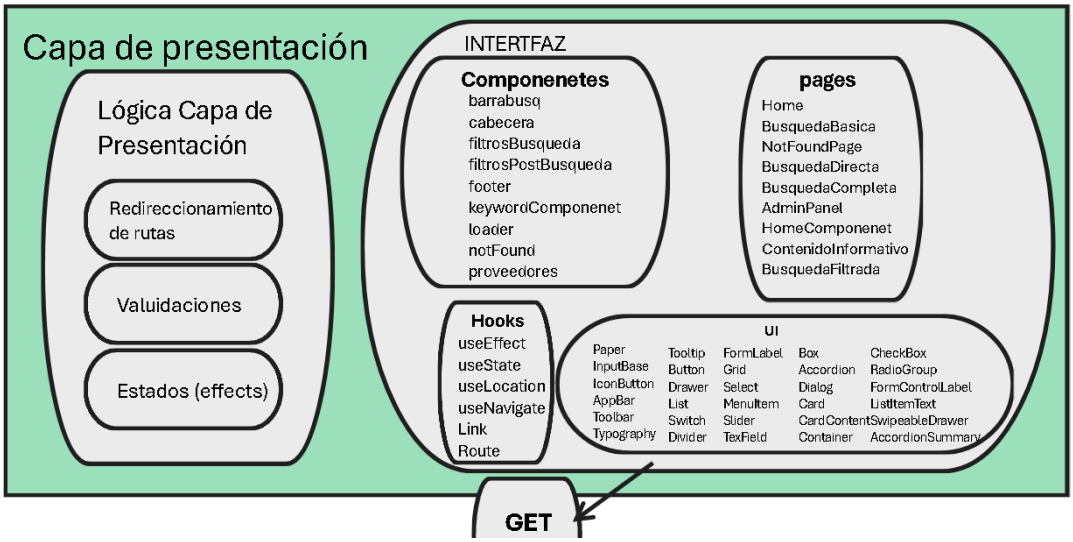


Figura 26: Figura Capa de Presentación Diagrama Lógico

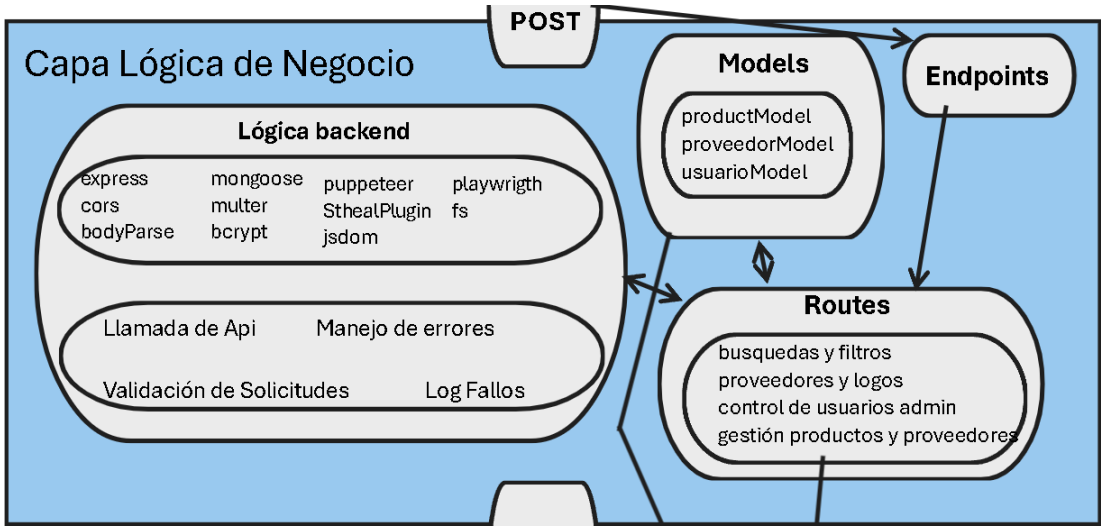


Figura 27:Figura Capa Lógica de Negocio Diagrama Lógico

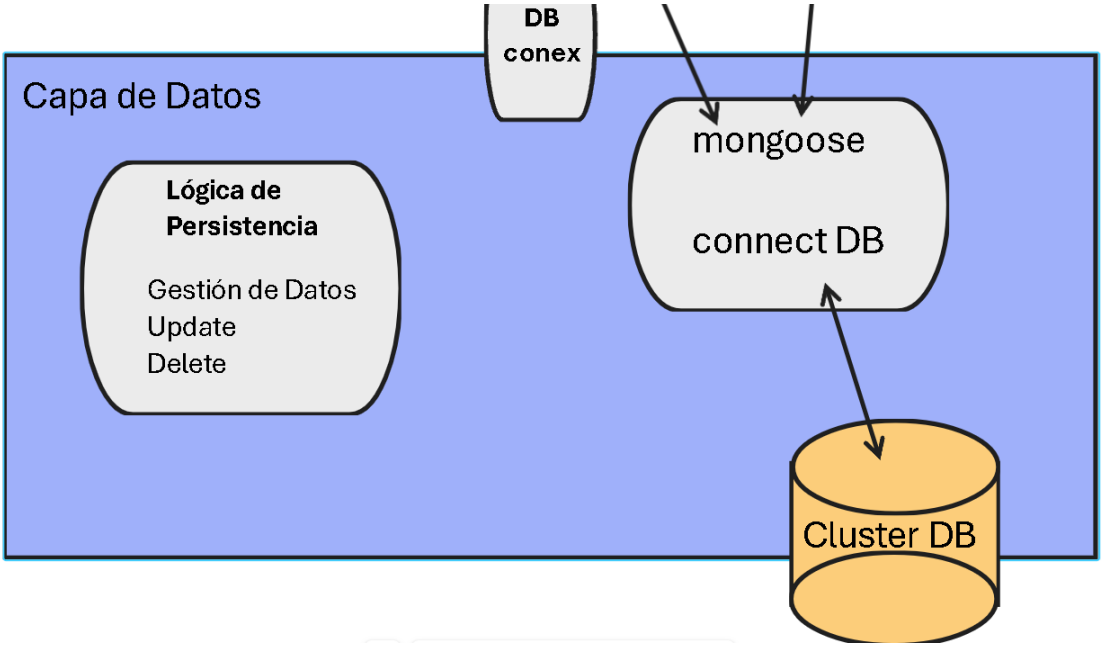


Figura 28: Capa de Datos Diagrama Lógico

5.2.- Arquitectura física

La arquitectura física representa la disposición concreta del hardware y la red que soportan la aplicación. Incluye detalles sobre los servidores, dispositivos de almacenamiento, y otros componentes de infraestructura que se utilizan para ejecutar y mantener el sistema. Esta arquitectura define cómo los distintos elementos se conectan entre sí, así como las configuraciones específicas para asegurar un rendimiento óptimo y una alta disponibilidad.

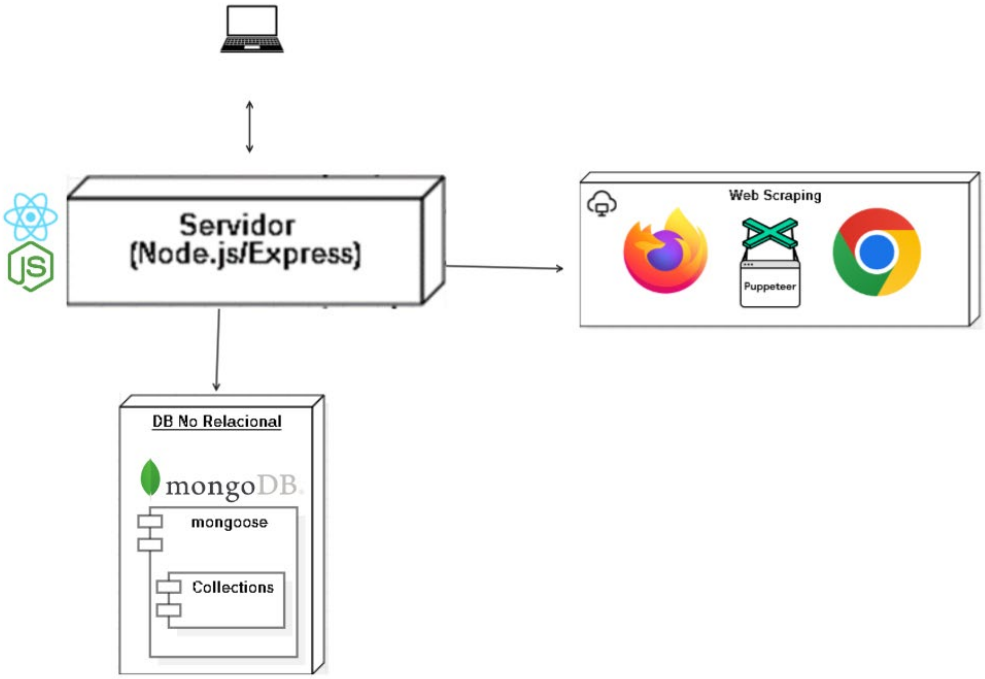


Figura 29: Diagrama Arquitectura Física

5.3.- Modelo E-R

El modelo entidad-relación (E-R) es una herramienta lógica que permite representar los datos de manera sencilla y comprensible. Su principal función es representar de forma gráfica los datos diseñando un modelo conceptual, lo que facilita visualizar cómo interactúan y se relacionan los diferentes elementos entre sí.

Aquí, se presenta el modelo Entidad-Relación que abarcará el proyecto.

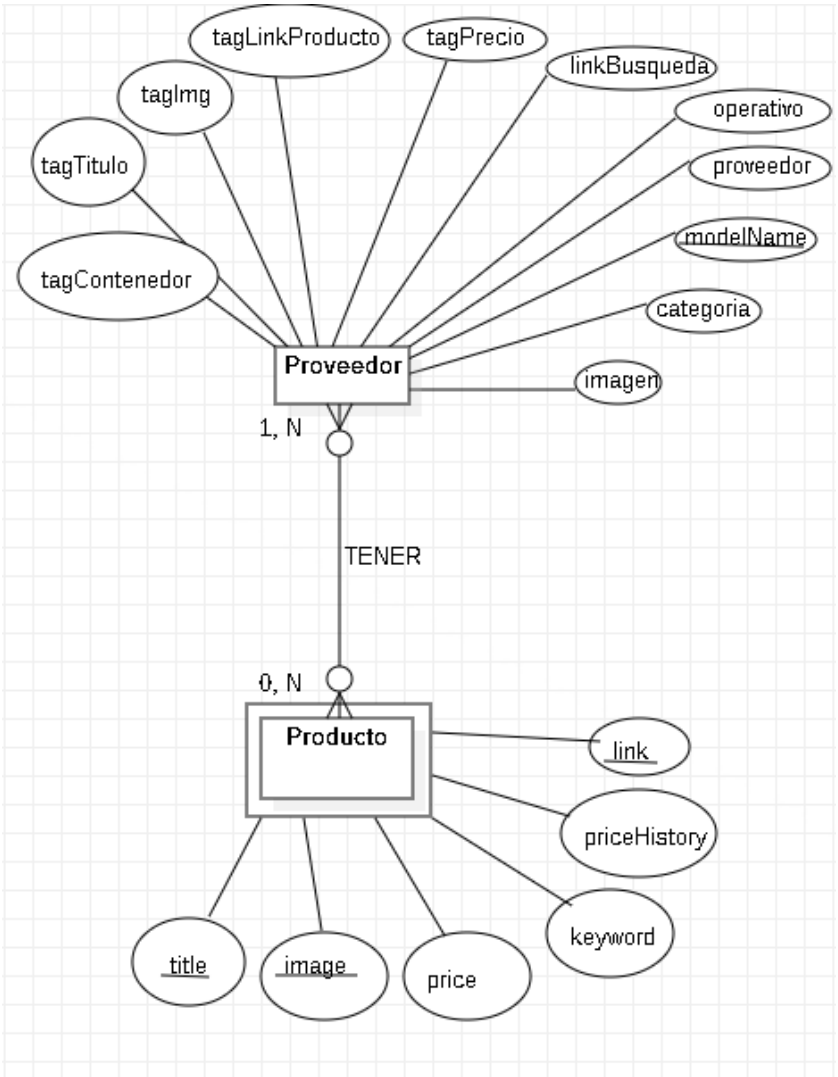


Figura 30: Modelo E-R

En un modelo relacional, una relación N a N genera una nueva tabla. Sin embargo, aunque se haya diseñado un modelo conceptual con una relación N a N, al no ser una base de datos relacional tradicional, no se sigue este esquema y no se genera otra tabla en el modelo lógico. Además, aunque el administrador esté en la base de datos para autenticación, se considera un caso especial y no forma parte del modelo E-R. Por eso, se decide excluirlo del modelo, ya que solo se usan sus datos para autenticación, no como parte de la estructura del modelo.

5.4.- Diccionario de datos

El diccionario de datos representa una recopilación detallada de todos los elementos de datos utilizados en la aplicación, describiendo cada uno de ellos y especificando sus atributos, tipos de datos. Incluye restricciones, campos, proporcionando una referencia para cualquier usuario que quiera entender el modelo E-R de nuestro sistema. Se va a omitir la propiedad '_id', debido a que se genera automáticamente para todas las

colecciones en MongoDB, y , aunque es el identificador para el propio mongo, no es la que se está utilizando como clave primaria.

Entidad Administrador			
Atributos	**Tipo**	**Único (unique)**	**Required**
username	String	Sí	Sí
password	String	No	Sí
isAdmin	Boolean	No	No

Figura 31: Diccionario de datos Usuario Administrador

Entidad Producto			
Atributos	**Tipo**	**Único (unique)**	**Required**
title	String	Sí	Sí
image	String	Sí	Sí
price	String	No	Sí
keyword	Array: [String]	No	Sí
link	String	Sí	Sí
priceHistory	Array: [String]	No	No

Figura 32: Diccionario de datos Productos

Entidad Proveedor			
Atributos	**Tipo**	**Único (unique)**	**Required**
tagContenedor	String	No	Sí
tagTitulo	String	No	Sí
tagImg	String	No	Sí
tagLinkProducto	String	No	Sí
tagPrecio	String	No	Sí
proveedor	String	No	Sí
linkBusqueda	String	No	Sí
modelName	String	Sí	Sí
operativo	Boolean	No	Sí
imagen	Buffer	No	No
categoria	Enum String: *['General', 'Motor', 'Electrónica', 'Cosmética', 'Moda', 'Deportes', 'Genérico']*	No	No

Figura 33: Diccionario de datos Proveedor

5.5.- Modelo lógico de datos

El modelo lógico de datos representa una abstracción del diseño de la base de datos, mostrando cómo se organizan y relacionan los datos sin entrar en detalles físicos. El modelo lógico de datos es crucial para asegurar que todos los requisitos de datos se capturen correctamente y que la base de datos sea coherente y eficiente.

De la misma manera que se ha explicado en el modelo E-R, el administrador se utiliza para autenticación no forma parte del modelo lógico de datos, y no se crea otra tabla para la relación N a N entre proveedor y producto

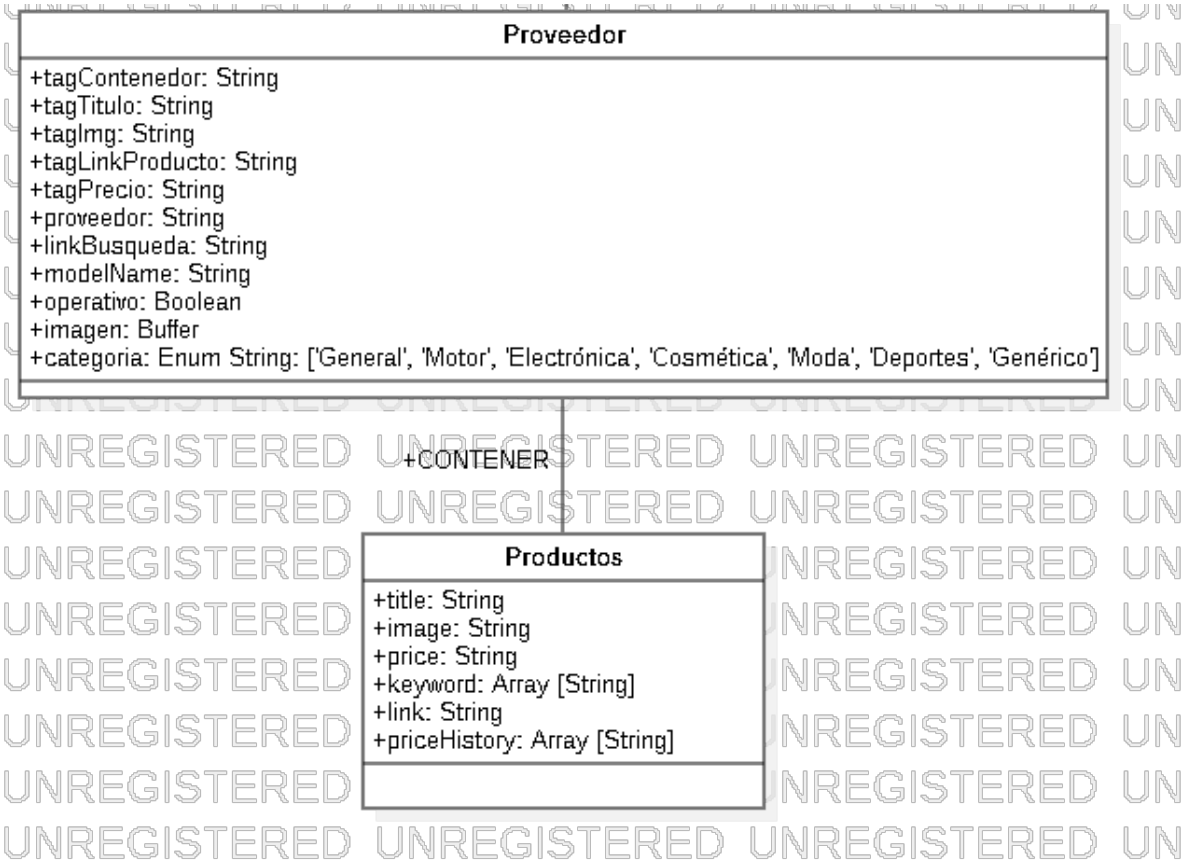


Figura 34: Modelo Lógico de Datos

No se genera una tabla adicional entre proveedores y productos porque el sistema se modela de manera dinámica. Esto significa que, al ser una base de datos no relacional, los datos del proveedor se incluyen en tiempo de ejecución cuando es necesario. No existe una tabla fija que relacione directamente proveedores y productos como en un modelo relacional. En su lugar, los datos del proveedor y del producto no están vinculados de forma permanente en tablas, sino que se manejan según se requieran, evitando duplicaciones o relaciones fijas entre ellos en la base de datos.

5.6.- Diagrama de secuencia

El diagrama de secuencia representa una visión detallada de cómo interactúan los diferentes componentes del sistema a lo largo del tiempo para llevar a cabo un proceso específico. Este diagrama muestra los objetos involucrados, los mensajes que se envían entre ellos y el orden en que ocurren estas interacciones. Es una herramienta crucial para entender el flujo dinámico de control y datos en el sistema, permitiendo identificar posibles puntos de fallo y optimizar la eficiencia del proceso.

A continuación, se muestra el diagrama de secuencia para que un usuario anónimo pueda iniciar sesión como administrador en la aplicación.

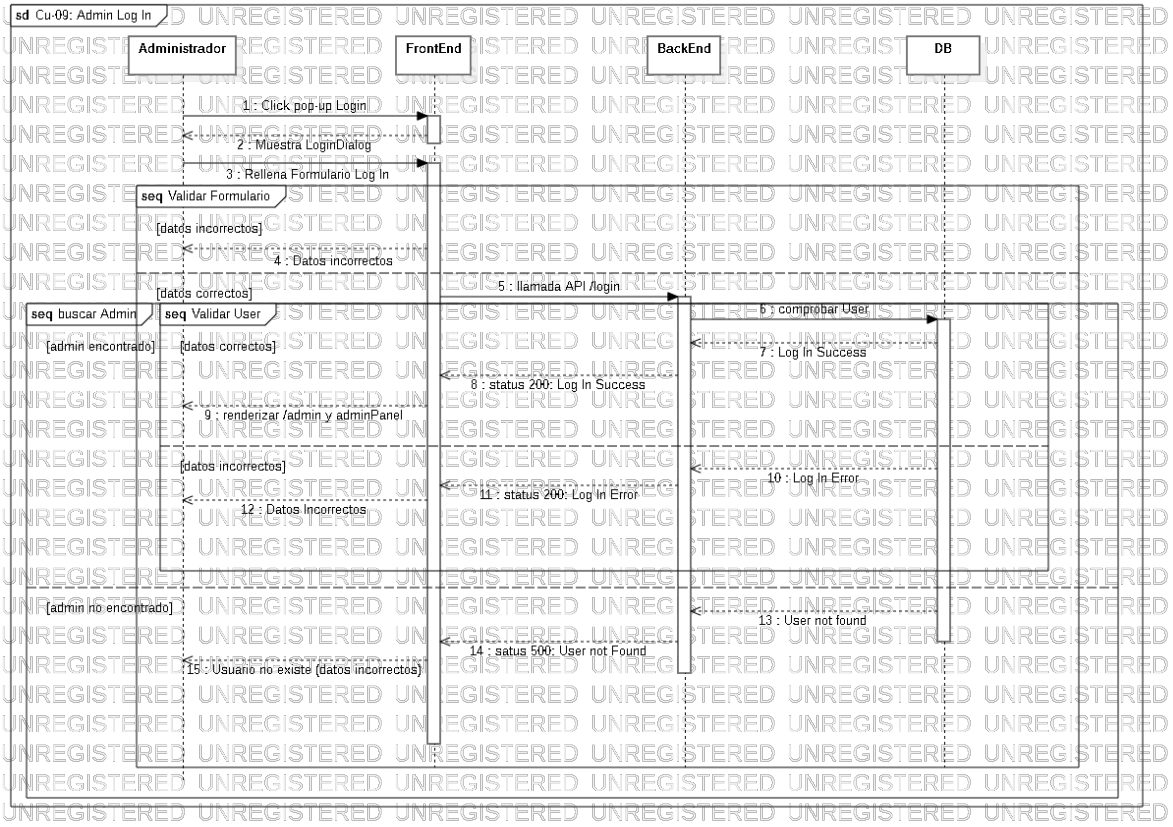


Figura 35: Diagrama de secuencia CU_09: Log In

A continuación, se muestra el diagrama de secuencia para que un usuario autenticado o admin cree un objetivo.

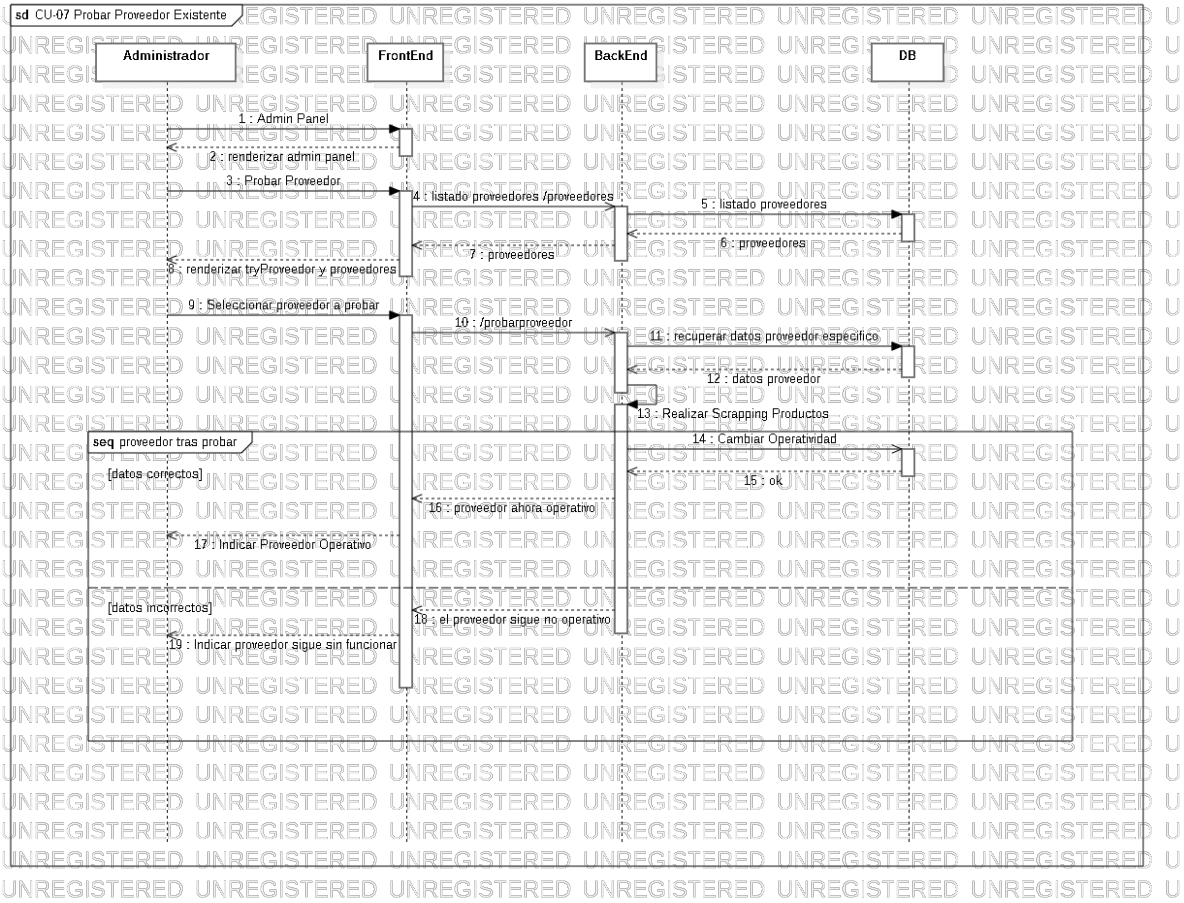


Figura 36: Diagrama de Secuencia CU_07: Probar Proveedor Existente

5.7.- Diseño de interfaz

En este apartado se describirán las vistas que se mostrarán al usuario de la aplicación. El diseño de interfaz abarca tanto la disposición visual de los elementos como la usabilidad y la experiencia del usuario. Se detallarán los componentes de cada pantalla, como botones, formularios, menús y otros elementos interactivos, asegurando que la navegación sea intuitiva y eficiente.

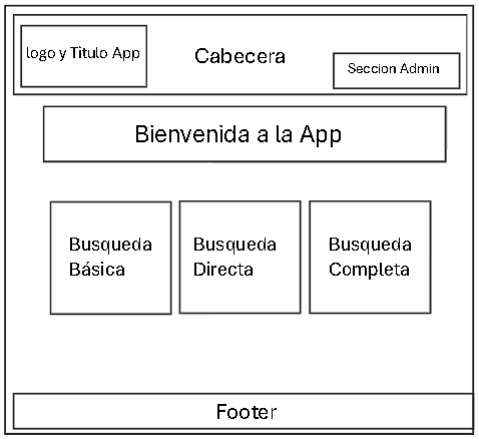
Nombre	Vista Principal
Descripción	Ventana principal que muestra los tipos de búsqueda y una breve explicación al navegar por encima
Activación	Al iniciar la aplicación
Boceto	
Eventos	1. Si pulsamos en un tipo de búsqueda, nos llevará a dicha ruta 2. Si pulsamos el navBar para iniciar como admin, saltará el Pop-Up para iniciar sesión

Tabla 40: Diseño Interfaz Vista Principal

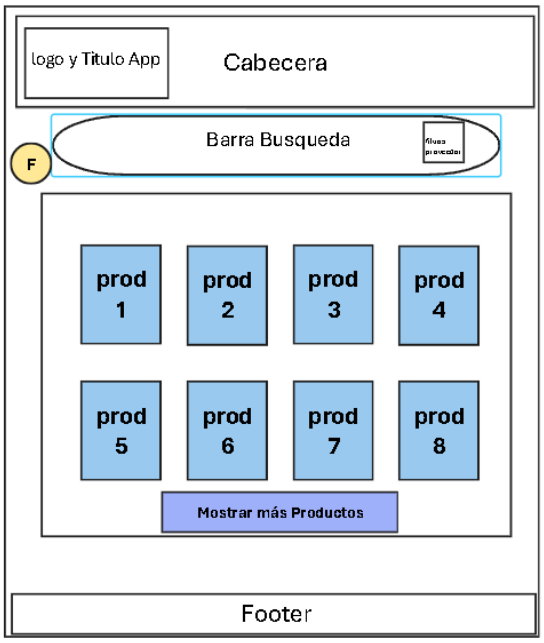
Nombre	Vista Búsqueda
Descripción	Ventana para cualquier tipo de búsqueda, ya que la lógica de esta va por detrás, no es renderizada en la propia interfaz
Activación	Al pulsar el cualquier tipo de búsqueda
Boceto	
Eventos	1. Si pulsamos el botón de búsqueda *

Tabla 41: Diseño Interfaz Vista Búsqueda

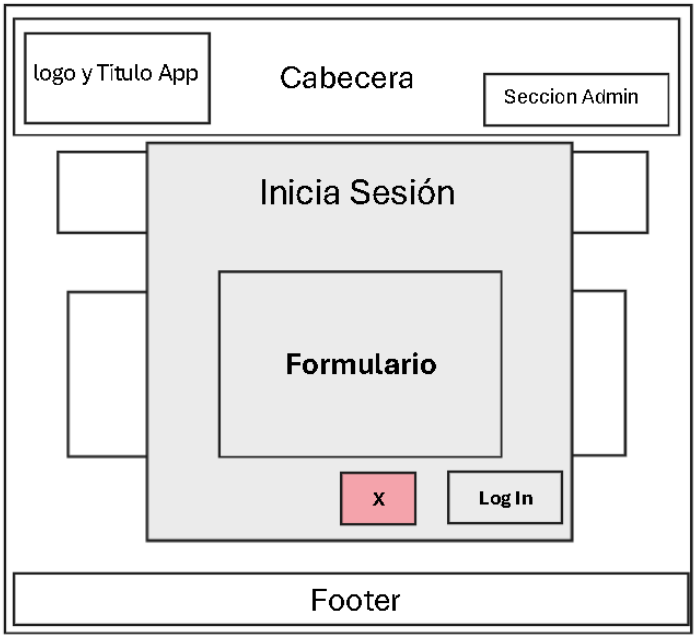
Nombre	Vista Log In
Descripción	Ventana para que el administrador se identifique
Activación	Al pulsar la sección de Admin de la cabecera (aparece en el boceto por encima porque es un Pop-up
Boceto	
Eventos	1. Si pulsamos el botón de sección de Administrador

Tabla 42: Diseño Interfaz Vista Log In

Nombre	Vista Panel Administrador
Descripción	Ventana para que el administrador pueda gestionar los proveedores de la aplicación
Activación	Al loguearse como administrador

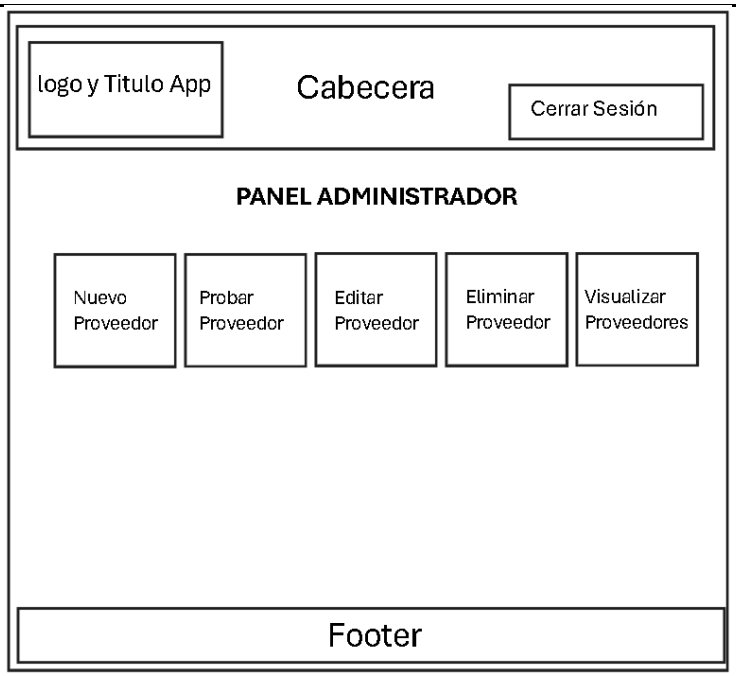
Boceto	
Eventos	1. Si seleccionamos el nuevo proveedor, se abrirá un pop-Up con un formulario para introducir un nuevo proveedor 2. Si seleccionamos el probar proveedor, se abrirá un pop-Up con la lista de proveedores guardados en la aplicación para probarlos 3. Si seleccionamos el editar proveedor, se abrirá un pop-Up con la lista de proveedores no operativos para, una vez seleccionados, podamos editar los datos de este 4. Si seleccionamos el eliminar proveedor, se abrirá un pop-Up con la lista de proveedores para poder seleccionar uno y eliminarlo 5. Si seleccionamos el visualizar proveedores, se abrirá un pop-Up con una tabla con la información de todos los proveedores 6. Si seleccionamos el icono de el logo y titulo, o el de cerrar sesión, borraremos la sesión del Local Storage y volveremos a la vista inicial

Tabla 43: Diseño Interfaz Vista Panel Administrador

5.8.- Paleta de Colores

Para el estilo de esta aplicación, la paleta de colores se caracteriza por una combinación de tonos oscuros, azules y verdes profundos, algunos toques metálicos y detalles en colores claros. A continuación, se destacan los principales colores identificados:

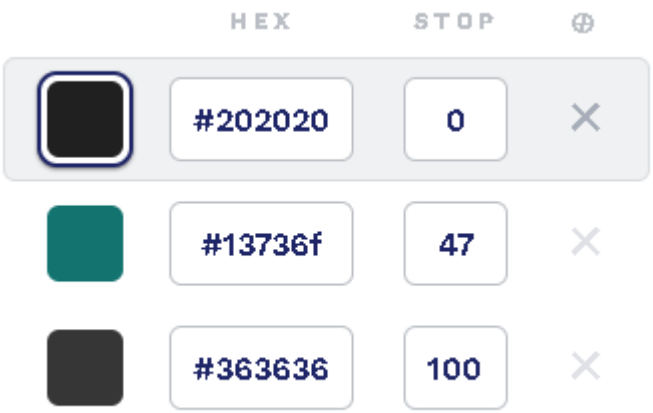


Figura 37: Paleta de Colores

La interfaz ha sido creada para ser completamente adaptable, ajustándose de manera automática a distintas resoluciones y tamaños de pantalla, garantizando una experiencia uniforme en dispositivos tanto de escritorio como móviles. Los elementos de navegación están claramente definidos y son de fácil acceso, lo que permite a los usuarios desplazarse por la aplicación de forma sencilla.

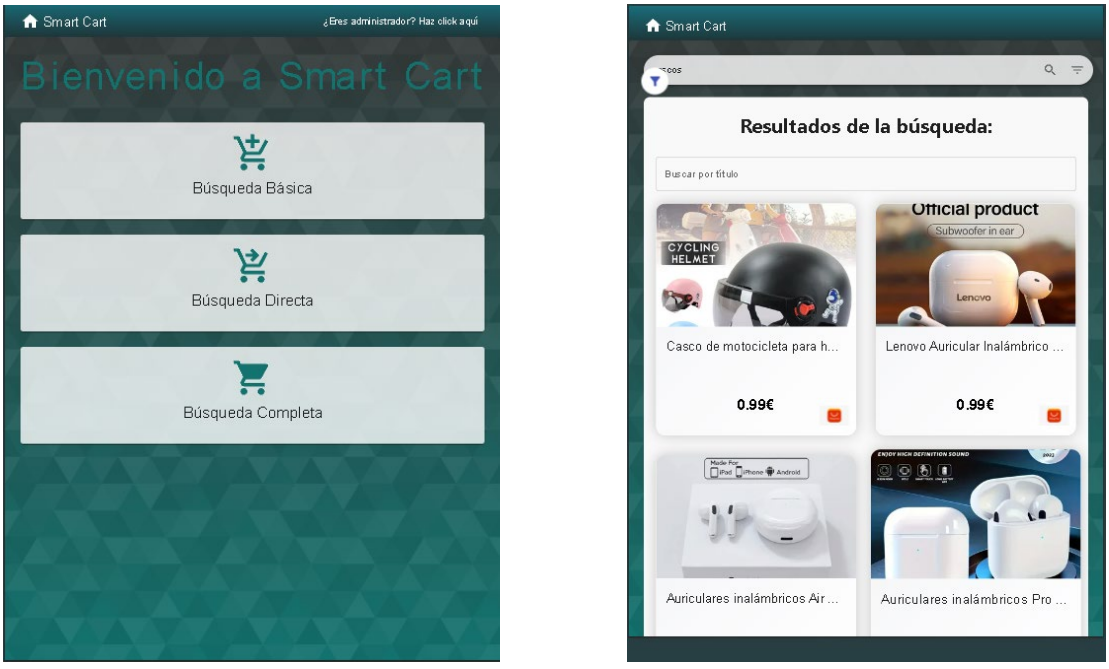


Figura 38: Página Inicial y Página de Búsqueda Responsive

Cada página dentro de la aplicación ha sido diseñada con especial atención para proporcionar una navegación fluida e intuitiva, asegurando que todos los componentes visuales y funcionales estén en sintonía con las necesidades del usuario. Se ha prestado especial cuidado a la presentación de la información, logrando un balance entre estética y funcionalidad, de modo que la interfaz no solo resulte atractiva, sino también fácil de manejar.

Capítulo 6
Implementación

En este capítulo se abordarán todos los aspectos relacionados con la construcción y desarrollo de la aplicación. Se explicarán en detalle los elementos técnicos.
Dividiremos este capítulo en cuatro secciones:

6.1.- Backend

La implementación del backend se divide en varias carpetas diferentes que abarcan varias áreas del servidor del proyecto basado en Node.js. A continuación, se detalla cada parte de esta estructura:

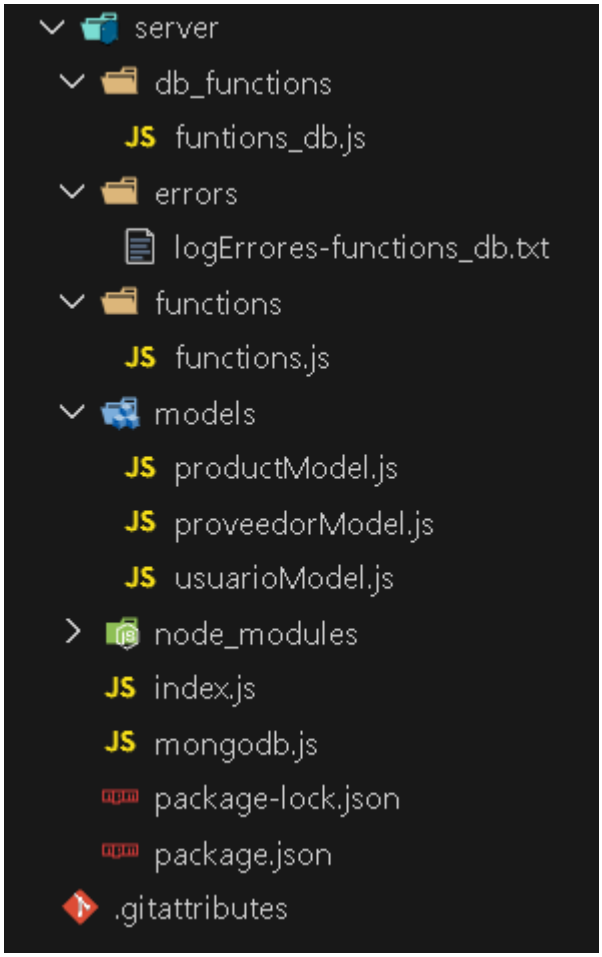


Figura 39: Estructura Carpetas Backend

1.- **db_functions:**

functions_db.js: Este archivo maneja todas las interacciones directas con la base de datos, junto con la implementación de una función de scraping, que realiza la búsqueda y extracción de datos de páginas web.

Respecto a la conexión con la base de datos, tiene distintas funciones que recogen el sistema CRUD (Crearte, read, update and delete), para los distintos modelos de la aplicación

Respecto la función de web scraping, se encarga de realizar peticiones HTTP a una página web específica, extraer información relevante y almacenarla en la base de datos. En este caso, esta información son los productos, y se extraen del link de búsqueda de proveedor.

Se utiliza principalmente el módulo de puppeteer, que se utiliza para renderizar una página web en un navegador sin cabeza para extraer datos dinámicos generados por JavaScript. Encontramos dentro del flujo general del scraping 6 pasos:

1. Verificación inicial del modelo: La función recibe dos parámetros: modelos (una lista de proveedores para realizar la búsqueda) y keyword (la palabra clave para buscar). Si la lista de modelos está vacía, la función retorna un objeto indicando que no hay modelos para buscar.

```
async function busquedaDirecta(modelos, keyword) {
    if (modelos.length < 1) {
        return {
            success: false,
            message: "No hay modelos para buscar",
            error: 0,
            productos: []
        };
    }

```

Figura 40: Función búsqueda Directa

2. Función autoScroll: Esta función realiza un scroll automático en la página web. Esto es importante porque algunas páginas solo cargan completamente el contenido a medida que el usuario hace scroll (ej. scroll infinito), y para intentar simular la interacción humana para páginas contra protección de bots. Simula el desplazamiento hacia abajo de la página incrementando la altura total en intervalos hasta que se haya cargado toda la página.

```
const autoScroll = async (page) => {
        await page.evaluate(async () => {
            await new Promise((resolve) => {
                let totalHeight = 0;
                const distance = 100;
                const timer = setInterval(() => {
                    const scrollHeight = document.body.scrollHeight;
                    window.scrollBy(0, distance);
                    totalHeight += distance;

                    if (totalHeight >= scrollHeight - window.innerHeight) {
                        clearInterval(timer);
                        resolve();
                    }
                }, 50);
            });
        });
    };
```

Figura 41: Función AutoScroll

3. Función searchWithTimeout: Recibe un modelo (proveedor) , una keyword y un tiempo de espera especifico. Aquí entra Puppeteer una biblioteca que permite controlar un navegador Chrome/Chromium sin cabeza (headless), para navegar a las páginas web de los proveedores de productos. Primero, se construye un link de búsqueda basado en la keyword, y Puppeteer carga la página con un timeout específico. Seguido de esto, se intenta manejar automáticamente los pop-ups de cookies, haciendo clic en los botones de aceptar o rechazar cookies si están presentes para poder renderizar la página tras el pop-up. Seguido de esto se extraen los datos de los productos (título, imagen, precio, enlace) utilizando selectores CSS específicos definidos en los tags de cada proveedor además de corregir enlaces de productos para convertirlos en una URL. Por último, la función retorna una lista de productos que contienen un título, una imagen, un precio y un enlace válidos.

4. Proceso principal (Promise.allSettled): La función principal itera sobre la lista de proveedores y llama a searchWithTimeout para cada uno de ellos, utilizando Promise.allSettled para manejar las promesas de búsqueda de forma concurrente. Se utiliza devolverProveedor para obtener los datos del proveedor asociado al modelo y verificar si está operativo. Si el proveedor no está operativo o no se encuentra, se maneja el error. Si se encuentran productos, estos se guardan en la base de datos mediante guardarProductos (para la búsqueda básica).

```
const resultados = await Promise.allSettled(modelos.map(async (modelo) => {
        try {
            const model = await devolverProveedor(modelo);
            if (!model) throw new Error(`Model ${modelo} not found`);
            const prv = await proveedor.findOne({ modelName: modelo });
            if (!prv || !prv.operativo) throw new Error(`Provider ${modelo}
is not operational`);
            const productos = await searchWithTimeout(prv, keyword);
            if (productos.length > 0) {
                await guardarProductos(productos, model, keyword);
            }
            return productos;
        } catch (error) {
            console.error(`Error in model ${modelo} en la keyword "${keywor
d}": ${error.message}`);
            return [];
        }
    }));
```

Figura 42: Proceso Principal Promises

5. Filtrado de resultados: Una vez completadas todas las búsquedas, se filtran los resultados exitosos y se formatean los precios de los productos encontrados.

```
const prodFinal = resultados
    .filter(result => result.status === 'fulfilled')
    .flatMap(result => result.value);

prodFinal.forEach(producto => {
    producto.price = f.formatearPrecio(producto.price);
});
```

Figura 43: Filtrado de Resultados

6. Retorno Final: La función devuelve un objeto con los productos encontrados y un mensaje de éxito, o un error en caso de que no se hayan encontrado productos.

La función establece un tiempo límite para la carga de las páginas (timeout) y maneja los errores que pueden ocurrir durante el scraping, como tiempos de espera excedidos o problemas de red.

2.- errors:

logErrores-functions_db.txt: Este archivo almacena todos los errores importantes producidos en el archivo functions_db.js, donde tenemos en cada entrada o línea la fecha del error , la hora en la que ocurrió, el proceso en el que ha ocurrido, y la explicación del error

```
[ 2024-07-01 / 13:40:29 ] - BUSQ_DIR ---> Error in model pccomponentsproductos para la keyword 'casco
[ 2024-07-01 / 13:40:29 ] - BUSQ_DIR ---> Error in model decathlonproductos para la keyword 'cascos':
[ 2024-07-01 / 13:40:29 ] - BUSQ_DIR ---> Error in model carrefourproductos para la keyword 'cascos':
[ 2024-07-01 / 13:40:48 ] - BUSQ_DIR ---> Error in model amazonproductos para la keyword 'cascos': Ti
[ 2024-07-01 / 13:40:48 ] - BUSQ_DIR ---> Error in model motocenterproductos para la keyword 'cascos'
```

Figura 44: Log Errores archivo logErrores.txt

3.- functions:

functions.js: Este archivo contiene utilidades y funciones relacionadas con el manejo de errores, corrección y formateo de precios y validación de enlaces. Estas funciones se utilizan principalmente para manejar el procesamiento y formateo de datos antes de ser

almacenados en una base de datos, y para gestionar errores que surgen durante la ejecución del programa. Son funciones auxiliares que proporcionan un soporte sólido para otras partes del sistema.

3.- models:

productModel.js: Este archivo define el modelo de productos en MongoDB usando Mongoose, una biblioteca que facilita las interacciones con bases de datos MongoDB. Este archivo sirve para definir la estructura de los productos que serán guardados en la base de datos y proporcionar funciones para crear modelos de productos de manera incremental (por proveedores). Facilita la creación y manejo de objetos de productos dentro de MongoDB.

```
export const createModel = (modelName) => {
  modelName = String(modelName);
  if (typeof modelName !== 'string') {
    throw new Error('modelName debe ser una cadena');
  }
  if (mongoose.models[modelName]) {
    return mongoose.models[modelName];
  }
  const nombreModelo = String(modelName);
  return mongoose.model(nombreModelo, new mongoose.Schema(baseSchema));
};
// Crear modelos utilizando la función createModel con el esquema base
const producto = createModel('producto');
```

Figura 45: Función Crear Modelo Productos

proveedorModel.js: Este archivo define un modelo de proveedores en MongoDB, también utilizando Mongoose. Define la estructura y los datos que se almacenan para los proveedores en la base de datos MongoDB. Gestiona información relacionada con cómo se extraen los productos de diferentes sitios web (como las etiquetas HTML para localizar los productos en el scraping) y permite manejar los proveedores de manera estructurada.

```
export const createModel = (modelName) => {
    modelName = String(modelName);
    if (typeof modelName !== 'string') {
      throw new Error('modelName debe ser una cadena');
    }
    if (mongoose.models[modelName]) {
      return mongoose.models[modelName];
    }
    const nombreModelo = String(modelName);
    return mongoose.model(nombreModelo, new mongoose.Schema(proveedorSchema));
  };
// Creación del modelo
const proveedor = mongoose.model('proveedor', proveedorSchema);

export default proveedor;
```

Figura 46: Función Crear Modelo Proveedor

usuarioModel.js: Este archivo define un modelo de usuarios en MongoDB, también utilizando Mongoose. Define la estructura y los datos que se almacenan para los administradores en la base de datos MongoDB. Gestiona información relacionada con el login de los mismos, con su usuario y contraseña.

```
const createModel = (schema, modelName) => {
  const modelSchema = new mongoose.Schema(schema);
  return mongoose.model(modelName, modelSchema);
};

const userSchema = {
  username: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true
  },
  isAdmin: {
    type: Boolean,
    default: false
  }
};


const User = createModel(userSchema, 'User');

export default User;
```

Figura 47: Función Crear Modelo Usuario

4.- index.js

Este archivo se encarga de manejar el servidor Express y definir las rutas para la interacción con la base de datos, junto con la lógica para recibir y procesar archivos (como imágenes) y realizar búsquedas de productos en varios proveedores.

- Configuración del Servidor: El archivo index.js define el servidor HTTP usando Express y lo configura con middleware para recibir y procesar solicitudes HTTP. El servidor también está preparado para recibir archivos mediante multer, que permite manejar la carga de imágenes. Dentro de los middlewares usados, tenemos:
  - express.json() y body-parser: Permiten que el servidor procese solicitudes con cuerpos JSON y URL codificados.
  - cors(): Habilita el uso compartido de recursos entre dominios diferentes, permitiendo que el frontend interactúe con el backend sin restricciones de origen.
  - multer: Configura el almacenamiento en memoria para procesar archivos (como imágenes) directamente sin almacenarlos en disco.

```
const app = express();
const upload = multer({ storage: multer.memoryStorage() });

// Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.json());
app.use(cors());
```

Figura 48: Configuración Inicial Servidor index.js

- Conexión a la DB: El servidor establece una conexión con MongoDB a través de mongoose.connect() recogiendo el link de la base de datos de un archivo privado para garantizar la seguridad de esta. La base de datos se usa para almacenar y recuperar datos de proveedores, productos y usuarios administradores.

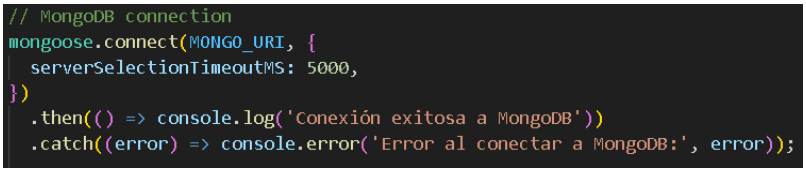


Figura 49: Conexión Backend con la DB

- Gestión de Proveedores: Varias rutas del servidor están destinadas a la gestión de proveedores. Estas rutas permiten la creación, actualización y recuperación de proveedores, incluyendo el manejo de sus imágenes.

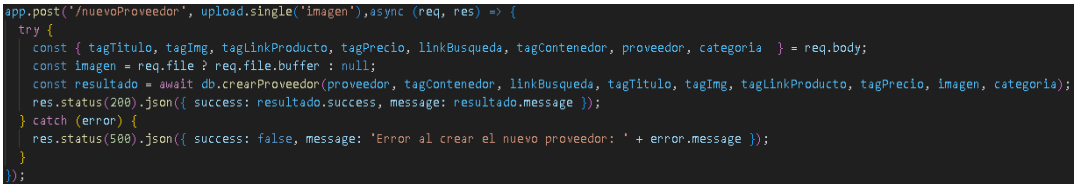


Figura 50: Endpoint nuevoProveedor

Figura 51: Endpoint updateProveedor

- Búsqueda de Productos: El servidor también gestiona rutas para realizar búsquedas de productos en múltiples proveedores, permitiendo consultas de productos con diferentes niveles de detalle.

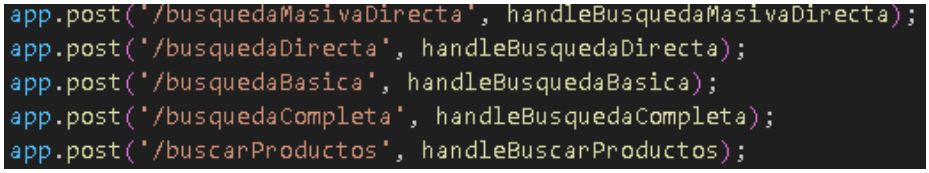


Figura 52: Endpoints tipos de Búsqueda

- Gestión de Administradores: Además de manejar proveedores y productos, el servidor permite la gestión de usuarios administradores, quienes tienen acceso a funciones más avanzadas como la adición de nuevos administradores o la autenticación

```
app.post('/addAdmin', async (req, res) => {
  try {
    const { username, password, adminPassword } = req.body;
    if (adminPassword !== ENCRYPTION_SECRET) {
      return res.status(403).json({ message: 'Contraseña de administrador incorrecta.' , success: false});
    }else{
    var result= await db.addUser(username, password);
    }
    if(result.success){
      res.status(200).json({result });
    }else{
      if(result.error!==0){
        throw new Error(result.message);
      }
      res.status(406).json({result });
    }
  } catch (error) {
    res.status(500).json({ message:  error.message, success: false });
  }
});
```

Figura 53: Endpoint addAdmin

```
app.post('/login', async (req, res) => {
  try {
      const decryptData = (encryptedData) => {
        const bytes = CryptoJS.AES.decrypt(encryptedData, ENCRYPTION_SECRET);
        const decryptedData = JSON.parse(bytes.toString(CryptoJS.enc.Utf8));
        return decryptedData;
      };
      const { data: encryptedData } = req.body;
      const { username, password } = decryptData(encryptedData);
      var result = await db.logIn(username, password);
      if(result.success){
        res.status(200).json({result });
      }else{
        if(result.error!==0){
          throw new Error(result.message);
        }
        res.status(200).json({result });
      }
    } catch (error) {
      res.status(500).json({ message:  error.message, success: false });
    }
});
```

Figura 54: Endpoint login

- Creación de Servidores en múltiples puertos: El archivo index.js crea tres servidores independientes en los puertos 3001, 3002 y 3003. Esto es útil para distribuir la carga de trabajo y mejorar la escalabilidad del sistema, permitiendo que el backend maneje un mayor número de solicitudes simultáneas.

```
// Function to create server
Codiumate: Options | Test this function
const createServer = (port) => {
  app.listen(port, () => {
    console.log(`Servidor escuchando en el puerto ${port}`);
  });
```

Figura 55: Creación de Servidor y Puertos

4.- mongodb.js

El archivo mongodb.js se encarga de establecer y manejar la conexión con una base de datos MongoDB utilizando el cliente de MongoDB (MongoClient). Este archivo establece la configuración necesaria para interactuar con la base de datos en la nube y realiza una prueba inicial de conectividad.

6.- package-lock.json:

Este archivo se genera automáticamente y detalla la estructura exacta de las dependencias instaladas en node_modules. Su función principal es garantizar que futuras instalaciones sean coherentes con las versiones previamente especificadas

.7.- package.json:

Archivo de configuración central del proyecto Node.js, que contiene los metadatos del proyecto, scripts de npm y una lista de dependencias junto con sus respectivas versiones.

6.2.- Frontend

La implementación del frontend tiene como objetivo principal crear una interfaz diseñada para satisfacer varias necesidades cruciales tanto para los usuarios como para la eficiencia global de la aplicación. Un frontend bien estructurado no solo mejora la apariencia visual, sino que también optimiza la usabilidad, accesibilidad y la experiencia del usuario. Está compuesto por varios directorios y archivos, principalmente desarrollados en React.

1.- node_modules:

Contiene todas las dependencias y paquetes instalados mediante npm, esenciales para el correcto funcionamiento del proyecto.

2.- public:

Guarda archivos públicos que se entregan directamente al cliente, como el archivo index.html, imágenes y otros recursos estáticos.

3.- src:

Este es el directorio principal donde se desarrolla la funcionalidad completa del frontend

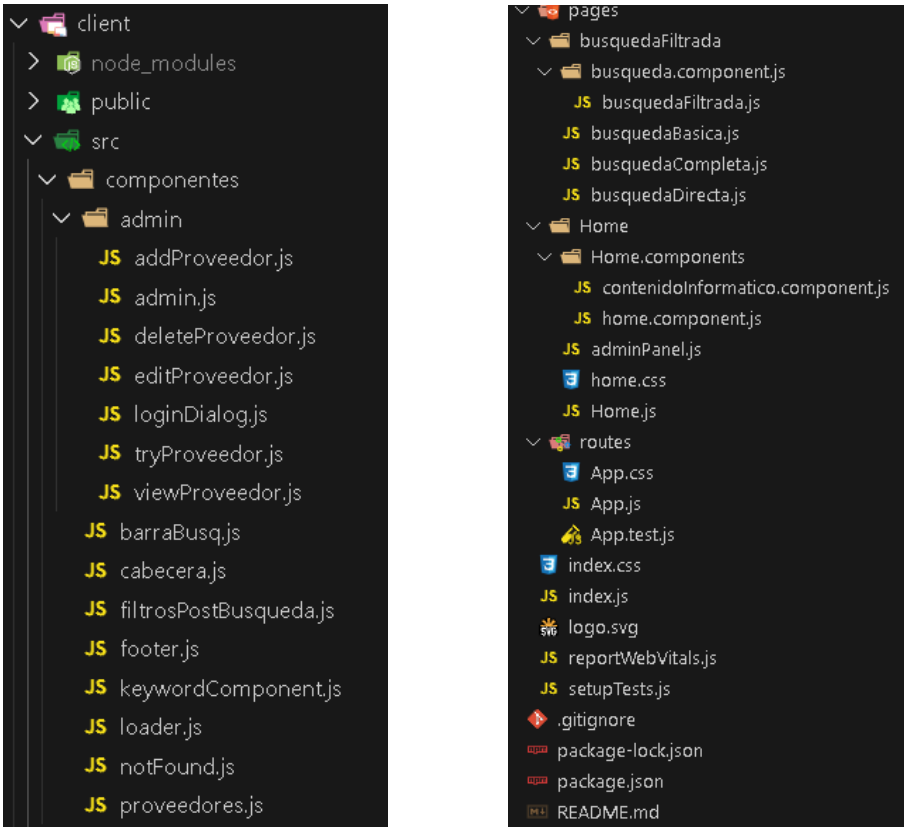


Figura 56: Estructura de carpetas Frontend

Además, cada directorio contiene archivos o subdirectorios esenciales para la definición del frontend. A continuación, se detalla con mayor profundidad la función de cada uno de ellos:

En el directorio **src**, se encuentra el subdirectorio **componentes**, el cual alberga los diversos componentes reutilizables de la interfaz de usuario que pueden ser implementados en diferentes páginas.

barraBusq.js: Componente que es la barra de búsqueda en todos los tipos de búsqueda.

Cabecera.js: Componente que renderiza una cabecera distinta dependiendo de donde nos encontremos en la aplicación

filtrosPostBusqueda.js: Componente encargado de mostrar y aplicar a la lista de productos, los filtros que el usuario requiera para afinar la búsqueda. Dentro de estos filtros, encontramos el rango de precios y el listado de proveedores, además de ordenar por el precio de mayor a menor y viceversa.

footer.js: Componente que es el pie de página

keywordComponenet.js: Componente que permite compartir el estado de la "keyword" y una función para actualizarlo entre diferentes componentes sin necesidad de pasar propiedades manualmente. Es utilizado para redefinir dinámicamente el conjunto de productos buscados con una keyword específica.
loader.js.: Componente de separador visual.

notFound.js: Página que se muestra cuando una ruta no existe.

proveedores.js: Componente encargado de la lista de proveedores anterior a la búsqueda

Carpeta admin: Dentro de esta carpeta, encontramos todos los componentes que se encargan de las funcionalidades del panel de administrador. Los componentes encontrados son:

- **addProveedor.js:** En este componente se encuentra el pop-up, siendo este un formulario con los datos necesarios para incluir un nuevo proveedor, y la llamada al backend para añadirlo a la aplicación.
- **admin.js:** Este es el componente de vista general de administrador, donde tenemos las 3 secciones u opciones que un administrador puede utilizar en la aplicación.
- **editProveedor.js:** En este componente se encuentra el pop-up encargado de la edición de datos de los proveedores no operativos de la aplicación. Hay casos en los que la búsqueda no es efectiva debido a que alguna etiqueta (tag) del proveedor es incorrecta (proveedor no operativo). En esta sección se pueden cambiar casi todos los datos de un proveedor para solucionar esto
- **loginDialog.js:** Este componente es el pop-up encargado del inicio de sesión del administrador en la aplicación.
- **tryProveedor.js:** Este componente se encarga de probar los proveedores de la aplicación, y comprobar su operatividad con las etiquetas existentes.
- **deleteProveedor.js:** Este componente permite al administrador borrar a proveedores existentes de la aplicación
- **viewProveedor.js** Este componente permite al administrador ver todos los proveedores de la aplicación y sus datos.

También se encuentra el subdirectorio **pages** que contiene las páginas (rutas) de la aplicación y las utilidades relacionadas directamente con las búsquedas de productos, además de la vista principal y el css de la aplicación. El directorio tienes los siguientes subdirectorios:

busquedaFiltrada: En esta carpeta, encontramos todos los componentes relacionados con la funcionalidad principal de la aplicación, es decir, la búsqueda de productos. Los componentes son:

- **busquedaFiltrada.js:** Dentro del subdirectorio de búsqueda.component, tenemos este componente, encargado de mostrar los productos recibidos del backend en la búsqueda. Se encarga de cargar todos los logos necesarios de los proveedores, de modificar dinámicamente el listado de productos con los filtros, y de mostrar todos los productos encontrados en cualquier tipo de búsqueda.
- **busquedaBasica.js:** Este componente se encarga de la búsqueda básica de la aplicación, junto a la llamada al backend para recuperar los productos guardados y el loader para las esperas.
- **busquedaDirecta.js:** Este componente se encarga de la búsqueda directa de la aplicación, junto a la llamada al backend para realizar el scraping los productos en tiempo real y el loader para las esperas.

- **busquedaCompleta.js:** Este componente se encarga de la búsqueda completa de la aplicación, haciendo una llamada al backend para realizar dinámicamente los dos tipos de búsqueda y mostrarlos con el componente de busquedaFiltrada

Home: En esta carpeta, encontramos todos los componentes relacionados con la vista principal de la aplicación, el panel de administrador y el css de la aplicación.

- **Home.components:** Contiene la página principal de la aplicación, con el contenido informativo explicativo del funcionamiento general de la aplicación (contenidoInformativo.componenet.js), y la página principal donde se encontrará dicho componente (home.componenet.js).
- **adminPanel.js:** Contiene el componente encargado de mostrar la vista principal del panel de control de proveedores al que puede acceder el administrador logueado
- **Home.js:** Contiene el componente principal que, junto a la cabecera y el pie de página (footer), conforman la vista inicial de la aplicación.
- **Home.css:** Estilo css de toda el frontend.

routes: Por último, tenemos la carpeta de routes, encargada de todas las rutas y páginas posibles de la aplicación. Esto se consigue en el archivo **App.js** , que gracias a react-router-dom, gestiona la navegación de la aplicación.

```
import './App.css';
import { Routes, Route } from 'react-router-dom';
import { Home } from '../Home/Home';
import NotFoundPage from '../../componentes/notFound';
import BusquedaBasica from '../busquedaFiltrada/busquedaBasica';
import BusquedaDirecta from '../busquedaFiltrada/busquedaDirecta';
import BusquedaCompleta from '../busquedaFiltrada/busquedaCompleta.js';
import { AdminPanel } from '../Home/adminPanel.js';



Codiumate: Options | Test this function
function App() {
  return (
    <Routes>
      <Route path='/' element={<Home />} />
      <Route path='*' element={<NotFoundPage />} />
      <Route path='/busquedaBasica' element={<BusquedaBasica />} />
      <Route path='/busquedaDirecta' element={<BusquedaDirecta />} />
      <Route path='/busquedaCompleta' element={<BusquedaCompleta />} />
      <Route path='/admin' element={<AdminPanel/>} />

    </Routes>
  );
}

export default App;
```

Figura 57: Rutas Frontend

Archivos de Configuración: components.json:

Configuración específica de componentes. **package-lock.json:**

Archivo generado automáticamente que describe la estructura exacta de las dependencias instaladas en node_modules.

package.json:

Archivo principal de configuración del proyecto Node.js que incluye metadatos del proyecto, scripts de npm, y la lista de dependencias y sus versiones.

El resto de los archivos, son los generados automáticamente al crear el frontend de React.

6.3.- Web Scraping

Esta aplicación implementa una solución de **web scraping** con el objetivo de recolectar información de productos desde proveedores online, utilizando las bibliotecas Puppeteer y Playwright para automatizar la navegación por páginas web. La aplicación interactúa con todos los proveedores guardados en la aplicación y sus diferentes sitios web de e-commerce para extraer datos clave de productos, como títulos, precios, imágenes y enlaces, los cuales se almacenan en una base de datos para su posterior uso.
El scraping es llevado a cabo de manera dinámica, simulando interacciones humanas y eludiendo mecanismos anti-bots, gracias a técnicas como el uso de User-Agent aleatorios y la desactivación de la propiedad navigator.webdriver.

A continuación, se explica el funcionamiento del proceso de Web Scraping, y sus funciones asociadas:

1. **Configuración del Navegador:** La aplicación utiliza Playwright (modulo encargado de automatizar navegadores para realizar pruebas automatizadas y hacer scraping de datos web) y Puppeteer(enfocado en automatizar Google Chrome, y ejecutar scripts en un entorno de navegador controlado) para realizar scraping. Inicialmente, se configura el navegador:
   - User Agents Aleatorios: Se selecciona un agente de usuario aleatorio de una lista para imitar diferentes dispositivos y evitar bloqueos por parte de los sitios web.
   - Desactivación de Web Driver: Para evitar ser detectados como bots, se redefine la propiedad navigator.webdriver para que devuelva false, simulando que la navegación es realizada por un usuario real.

Figura 58: Lanzamiento de Navegador con Puppeteer

2. **Navegación y Carga de Página:** El navegador automatizado accede al enlace de búsqueda del proveedor, donde se sustituye el valor ***** (almacenado de esa manera en el link de búsqueda del proveedor) en la URL por una palabra clave.

```
const link = model.linkBusqueda.replace('*****', encodeURIComponent(keyword));
await page.goto(link, { waitUntil: 'networkidle0', timeout });
```

Figura 59: Scraping enlace de Búsqueda

Además, se implementa una lógica para manejar los pop-ups de cookies:

```
try {
    await page.evaluate(() => {
        const cookieButtons = Array.from(document.querySelectorAll('button'));
        cookieButtons.forEach(button => {
            const text = button.innerText;
            if (text && /OK|aceptar|consent|ok|allow|agree|Aceptar todas|Rechazar todas|
                if (!button.disabled && button.offsetParent !== null) {
                    button.click();
                }
            }
        });
    });
} catch (buttonError) {
    console.log("Error en los botones, " + buttonError);
}
```

Figura 60: Scraping gestión de Botones de Cookies

3. **Interacción con la Página:** Simula movimientos y clics con el ratón para evitar ser bloqueado por mecanismos anti-scraping.

```
const autoScroll = async (page) => {
    await page.evaluate(async () => {
        await new Promise((resolve) => {
            let totalHeight = 0;
            const distance = 100;
            const timer = setInterval(() => {
                const scrollHeight = document.body.scrollHeight;
                window.scrollBy(0, distance);
                totalHeight += distance;

                if (totalHeight >= scrollHeight - window.innerHeight) {
                    clearInterval(timer);
                    resolve();
                }
            }, 50);
        });
    });
};
```

Figura 61: Función AutoScroll

4. **Extracción de Productos(Scraping):** Una vez la página cargada con los resultados de la búsqueda, se comienza el scraping de estos. Para ello, con los tags guardados en los proveedores, son los selectores HTML de los componentes que se desean extraer. Que en este caso son el título del producto, la imagen, el precio y el enlace al propio producto. Esto se consigue se utilizan las funciones

'$$eval' de Puppeteer para iterar sobre los contenedores de productos y extraer los datos.

```
const productos = await page.$$eval(model.tagContenedor, (elements, tags) => {
    return elements.map(el => {
        const title = el.querySelector(tags.title)?.innerText || null;
        let image = el.querySelector(tags.image)?.src || el.querySelector(tags.image)?.srcset;
        const price = el.querySelector(tags.price)?.innerText || null;
        let link = el.querySelector(tags.link)?.href || null;
        if (!link) {
            link = el.getAttribute('href') || null;
        }
        if (link && link.startsWith('/')) {
            link = 'https://www.' + tags.constr.replace(/\s+/g, '').replace(/[^\x00-\x7F]/g, '') + '.es' + link;
        }

        return { title, image, price, link };
    });
}, {
    title: model.tagTitulo,
    image: model.tagImg,
    price: model.tagPrecio,
    link: model.tagLinkProducto,
    constr: model.proveedor.toLowerCase()
});
```

Figura 62: Scraping de los tags HTML

5. **Filtrado de datos y cierre de navegador:** Después de recolectar todos los productos de la página, se realiza un filtrado para todos aquellos productos que no vengan con la información requerida.

```
await browser.close();
return productos.filter(producto => producto.title && producto.image && producto.price && producto.link);
```

Figura 63: Filtrado Productos Scraping

6. **Guardado de datos:** Una vez se han extraído todos los datos , estos se guardan en la base de datos (actualizándolos en lugar de que ya sean existentes en caso de que algún dato (como el precio) haya cambiado). Luego, los usuarios pueden realizar búsquedas basadas en palabras clave que consultan directamente en la base de datos (búsqueda básica) o que disparan nuevas rondas de scraping en los proveedores correspondientes (búsqueda directa). La búsqueda completa complementa estas dos búsquedas, quedándose con el producto de la búsqueda directa en caso de repetición, ya que es el producto más actualizado.

```
await guardarProductos(productos, model, keyword);
```

Figura 64: Llamada Función guardarProductos

Las funciones encargadas de hacer algún tipo de scraping son:

- busquedaDirecta: Encargada de realizar la búsqueda por la keyword pasada por el usuario a los modelos de proveedores especificados.
- tryProveedor: Esta función hace una llamada a la función testProveedor, que es una función genérica que devuelve un booleano , en caso de que devuelva algún producto o no, es decir, que el proveedor esté operativo o no.

6.4.- Dificultades

Durante el desarrollo del proyecto, me he topado con varios obstáculos que me han llevado a enfrentarme al código en repetidas ocasiones. Hubo situaciones en las que, al solucionar un problema o ajustar un componente para una página, terminaba provocando que algo más dejara de funcionar correctamente.

El problema inicial fue el cambio de enfoque que tuvo que tomar la aplicación debido a el problema surgido con las APIs de productos esperados. Cuando surgió la idea de este proyecto, el enfoque inicial se basaba en recoger los productos de las APIs públicas que ofrecen algunos proveedores de productos, pero a la hora de desarrollar este proyecto, se encuentra que, dependiendo de cada proveedor, estas APIs no eran del todo públicas. En el ejemplo del proveedor Amazon, esta API de productos era accesible, tan solo si se es afiliado a Amazon dentro del sector de affiliate marketing. Esto requería haber vendido al menos tres productos en la plataforma. En el caso de Ali-Express, hacía falta un número de teléfono chino para verificar que eras humano. Es por eso por lo que se tuvo que optar por otro enfoque para el desarrollo de la aplicación y de ahí surgió la idea de Web Scraping.

Otro de los mayores problemas al desarrollar la aplicación fue este scraping, ya que un método que funcionaba para algún tipo de proveedor, para otro proveedor era inútil y no conseguía recuperar ningún dato. El desafío más difícil, fue, sin duda, lograr que esta función pueda ser genérica para cualquier selector pasado de cualquier proveedor sin mucha necesidad de cambio en el código de scraping. Cuando conseguí encontrar el modelo ahora utilizado para el scraping, este problema se solucionó, aunque apareció el problema que requería un mantenimiento constante en caso de que los selectores de un proveedor cambiaran. De aquí nació la solución de incluir el usuario administrador, encargado de gestionar estos proveedores, que inicialmente no estaba contemplado. Además, otra sección que dio problema fue el de los filtros de productos, ya que, al ser componentes distintos, la comunicación entre estos para que los productos se actualicen automáticamente con los filtros seleccionados no ha sido fácil. La implementación de el keyword.componenet, ayudó a encontrar la solución con la comunicación de variables entre componentes.

Capítulo 7
Pruebas

En este capítulo se realizan las pruebas necesarias para garantizar el funcionamiento adecuado de la aplicación web. El contenido se divide en dos partes principales: pruebas de caja negra y pruebas de caja blanca.

7.1.- Caja negra

Las pruebas de caja negra, también conocidas como pruebas funcionales, se enfocan en comprobar el comportamiento externo del software sin necesidad de conocer su estructura interna. El propósito es verificar que las funcionalidades de la aplicación cumplen con los requisitos establecidos y que los resultados obtenidos son correctos. Estas pruebas se basan en las entradas y salidas del sistema, evaluando si el software responde de manera adecuada a diferentes entradas y escenarios.

PCN_01:Realizar Búsqueda Directa	
Objetivo	Verificar que el sistema permite realizar búsquedas básicas de productos almacenados en la plataforma.
Precondiciones	-Deben existir productos registrados en la base de datos de la aplicación. -Deben existir proveedores registrados en la aplicación
Datos de entrada	-Término de búsqueda: "Smartphone" -Listado de Proveedores a Buscar
Acción esperada	Al realizar la búsqueda, deberían mostrarse productos almacenados que coincidan con el término.
Resultado	Positivo

Tabla 44: PCN_01 Realizar Búsqueda Directa

PCN_02: Realizar Búsqueda Directa	
Objetivo	Verificar que el sistema permite realizar búsquedas directas en tiempo real mediante web scraping en los proveedores seleccionados.
Precondiciones	Deben existir proveedores registrados en la aplicación
Datos de entrada	-Término de búsqueda: "Laptop" -Listado de Proveedores a Buscar
Acción esperada	El sistema debería conectarse con los proveedores seleccionados y mostrar productos disponibles en tiempo real.
Resultado	Positivo

Tabla 45: PCN_02 Realizar Búsqueda Directa

PCN_03: Realizar Búsqueda Completa	
Objetivo	Verificar que el sistema permite realizar búsquedas simultáneas, tanto básicas como directas, mezclando ambos resultados.
Precondiciones	Deben existir productos en la base de datos Deben existir proveedores registrados en la aplicación
Datos de entrada	-Término de búsqueda: "Televisor" -Listado de Proveedores a Buscar
Acción esperada	Los algoritmos de combinación deben mezclar correctamente los resultados sin duplicados ni errores.
Resultado	Positivo

Tabla 46: PCN_03 Realizar Búsqueda Completa

PCN_04: Autenticación del Administrador	
Objetivo	Verificar que el sistema de autenticación de administradores funciona correctamente.
Precondiciones	El administrador debe existir en la DB
Datos de entrada	-admin username -admin password
Acción esperada	Las credenciales deben ser validadas correctamente y el sistema debe gestionar las sesiones de manera segura.
Resultado	Positivo

Tabla 47: PCN_04 Autenticación del Administrador

PCN_05: Agregar Nuevo Proveedor	
Objetivo	Verificar que el sistema permite a los administradores añadir nuevos proveedores
Precondiciones	El administrador debe de estar logueado.
Datos de entrada	-Nombre Proveedor -Tag Titulo -Tag Imagen Producto -Tag link producto -Tag Precio -Link Búsqueda General -Categoria -Imagen
Acción esperada	Al pulsar el botón “Guardar Proveedor”, se guardará el proveedor en la aplicación
Resultado	Positivo

Tabla 48: PCN_05 Agregar Nuevo Proveedor

PCN_06: Probar Proveedor	
Objetivo	Verificar que el sistema permite a los administradores probar la operatividad de los proveedores

Precondiciones	-El administrador debe existir en la DB -Debe existir proveedores en la aplicación
Datos de entrada	-Proveedor a probar
Acción esperada	Al pulsar el botón “Probar Proveedor” se mostrará un mensaje indicando si el proveedor es operativo o no.
Resultado	Positivo

Tabla 49: PCN_06 Probar Proveedor

PCN_07: Editar Proveedor	
Objetivo	Verificar que el sistema permite a los administradores modificar ciertos datos de los proveedores del sistema
Precondiciones	-El administrador debe existir en la DB -Debe existir proveedores en la aplicación -El proveedor debe estar no operativo
Datos de entrada	-Modelo de proveedor a editar -Nombre Proveedor -Tag Titulo -Tag Imagen Producto -Tag link producto -Tag Precio -Link Búsqueda General -Categoria -Imagen
Acción esperada	Al pulsar el icono “Actualizar Proveedor” se mostrará un mensaje indicando si el proveedor es operativo o no y se actualizarán los datos.
Resultado	Positivo

Tabla 50: PCN_07 Editar Proveedor

PCN_08: Visualizar Proveedor	
Objetivo	Verificar que el sistema permite a los administradores visualizar los datos de todos los proveedores
Precondiciones	-El administrador debe existir en la DB -Debe existir proveedores en la aplicación
Datos de entrada	N/E
Acción esperada	Al pulsar el icono “Visualizar Proveedores” se mostrará una tabla con todos los datos de los proveedores
Resultado	Positivo

Tabla 51: PCN_06 Visualizar Proveedor

PCN_09: Eliminar Proveedor	
Objetivo	Verificar que el sistema permite a los administradores eliminar proveedores del sistema
Precondiciones	-El administrador debe existir en la DB -Debe existir proveedores en la aplicación -El proveedor debe estar no operativo
Datos de entrada	-Modelo de proveedor a eliminar
Acción esperada	Al pulsar el icono "Eliminar Proveedor" se mostrarán los proveedores, se seleccionará uno y mostrará un mensaje indicando si desea eliminar el proveedor .
Resultado	Positivo

Tabla 52: PCN_09 Eliminar Proveedor

7.2.- Caja blanca

Las pruebas de caja blanca se enfocan en el análisis minucioso del código fuente de la aplicación. Estas pruebas son fundamentales para evaluar y validar la funcionalidad interna de cada módulo, garantizando que todos los caminos lógicos del programa se comporten como se espera.

Durante estas pruebas, se examina el comportamiento interno del software para identificar posibles errores en el código. Este proceso es crucial para asegurar la integridad y la precisión del software. En el contexto del proyecto e-Change, se han realizado diversas pruebas de caja blanca para asegurar la calidad y la solidez de la aplicación.

1.- **Conexión con la Base de Datos**: Verificar que la aplicación pueda establecer correctamente la conexión con la base de datos y que estas conexiones se manejen adecuadamente, tanto al abrir como al cerrar.

2.- **Entrada y Salida de Datos con el Servidor de la Base de Datos**: Asegurarse de que los datos se envíen y recuperen correctamente desde la base de datos, sin pérdida ni errores.

3.- **Manejo de Errores en la Conexión**: Comprobar que la aplicación gestiona de manera adecuada los errores de conexión con la base de datos, proporcionando mensajes de error claros y precisos al usuario.

4.- **Autenticación y Creación de Cuentas de Usuario**: Verificar que los procesos de registro y autenticación de administradores funcionan correctamente, incluyendo la encriptación de contraseñas y la gestión de sesiones.

5.- **Rutas Introducidas por el Usuario**: Asegurar que las rutas de navegación dentro de la aplicación se gestionen correctamente, guiando al usuario a la página deseada sin errores.

6.- **Manejo de Sesiones**: Evaluar la gestión segura de las sesiones de usuario, incluyendo el inicio y cierre de sesión, así como la persistencia de la sesión durante la navegación.

Capítulo 8
Manual de usuario

Este capítulo presenta la documentación esencial en forma de manual, diseñada para que cualquier usuario pueda comprender el funcionamiento de la aplicación. Los manuales están organizados en dos categorías principales:

8.1.- Manual de instalación

Este manual ofrece una guía detallada, paso a paso, sobre cómo realizar la instalación de la aplicación. Incluye todos los requisitos previos, como el software y hardware necesarios, y proporciona instrucciones específicas para distintos sistemas operativos. Además, se abordan soluciones a problemas frecuentes que podrían surgir durante la instalación, garantizando que los usuarios puedan completar el proceso sin dificultades.

Antes de comenzar, se instalarán los siguientes componentes:

1.- Node.js

2.- npm (Node Package Manager): Se instala automáticamente con Node.js.

Para ello, se clonará un repositorio de la aplicación desde GitHub para llevar un control de versiones y una rama a la que se van subiendo modificaciones.

git clone https://github.com/tu-usuario/tu-repositorio.git

cd tu-repositorio

A continuación, se configurará MongoDB Atlas. Para ello, primero se necesita una cuenta de MongoDB. Una vez creada la cuenta (se puede hacer desde su sitio web), se configura un nuevo clúster que hará referencia a nuestra BD.

Después de realizar estos pasos, necesitarás obtener la cadena de conexión, que será una dirección de MongoDB que hará referencia al clúster que has creado.

mongodb://<URI o URL de la DB>:<puerto>

A continuación, serán necesario instalar varias dependencias del backend, para ello será necesario moverse al directorio del servidor e instalar las dependencias necesarias.

cd project -> cd server

npm install

De esta manera , se instalarán todos los módulos necesarios para el correcto funcionamiento de la aplicación.

Una vez hecho eso, se puede iniciar el servidor insertando en la terminal la siguiente línea:

npm run server

Tendremos la parte del servidor completamente configurada y ejecutándose en 3 puertos: 3001, 3002 y 3003

Para el frontend, será necesario instalar las dependencias requeridas, para lo cual deberemos cambiar de directorio. Se recomienda utilizar dos terminales separadas: una para el backend y otra para el frontend. El IDE Visual Studio Code facilita la gestión de múltiples terminales de manera sencilla.

cd project -> cd client

npm install

Una vez realizados los pasos anteriores, podremos iniciar nuestra aplicación de React. En nuestro caso estará alojada en el 3000

npm start

Una vez realizados estos pasos, tendremos la aplicación SMARTCART funcionando perfectamente.

8.2.- Manual de usuario

Este manual ha sido estructurado para satisfacer las necesidades de diversos tipos de usuarios. Proporciona instrucciones precisas y detalladas sobre el uso de la aplicación, cubriendo todas sus funcionalidades principales. Incluye secciones específicas adaptadas a diferentes roles, como usuarios anónimos, usuarios autenticados y administradores, con guías personalizadas que simplifican la comprensión y el uso eficiente de la plataforma. Además, se presentan ejemplos prácticos que ayudan a los usuarios a resolver cualquier duda que surja durante su interacción con la aplicación.

8.2.1.- Usuario anónimo:

Este manual está diseñado para los usuarios anónimos. Como usuario anónimo, puedes buscar productos de los proveedores guardados. Dentro de los tipos de búsqueda tienes la búsqueda básica, que te permite buscar productos ya guardados en la aplicación de una manera rápida y sencilla (aunque puede haber productos con el precio desactualizado ya que ha podido cambiar); la búsqueda directa que, aunque es algo más lenta, permite buscar los productos buscados de los proveedores seleccionados en tiempo real; y por último está la búsqueda completa que complementa las dos búsquedas a la par.

Al entrar en la aplicación, tienes una ventana de bienvenida que muestra los tipos de búsqueda posibles. Además, al pasar el ratón sobre estos tipos, aparece un pequeño texto explicativo sobre dicha búsqueda.

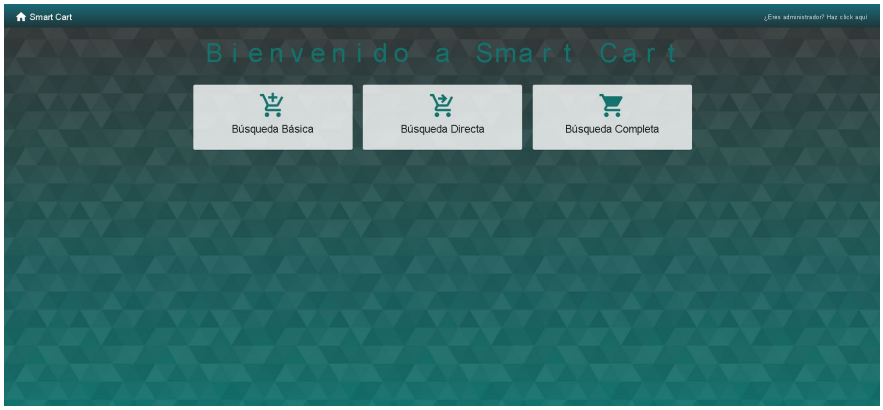


Figura 65: Página Inicial

Figura 66: Información Búsqueda Directa

Al seleccionar cualquiera de los tipos de búsqueda, aparecerá el título del tipo de búsqueda, y una barra de búsqueda para buscar productos, y un botón para aplicar los proveedores a la derecha de la barra de búsqueda.

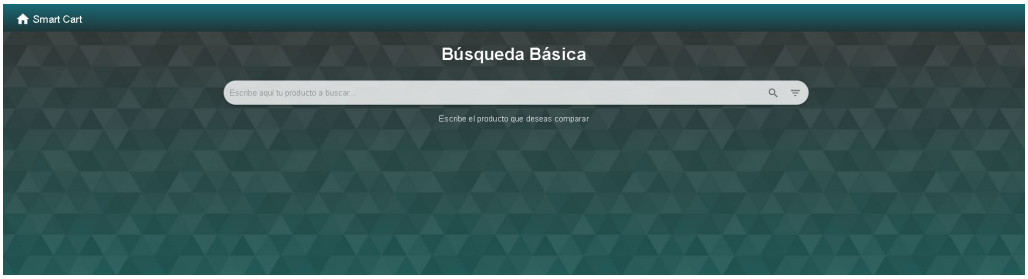


Figura 67: Página Principal Búsqueda Básica

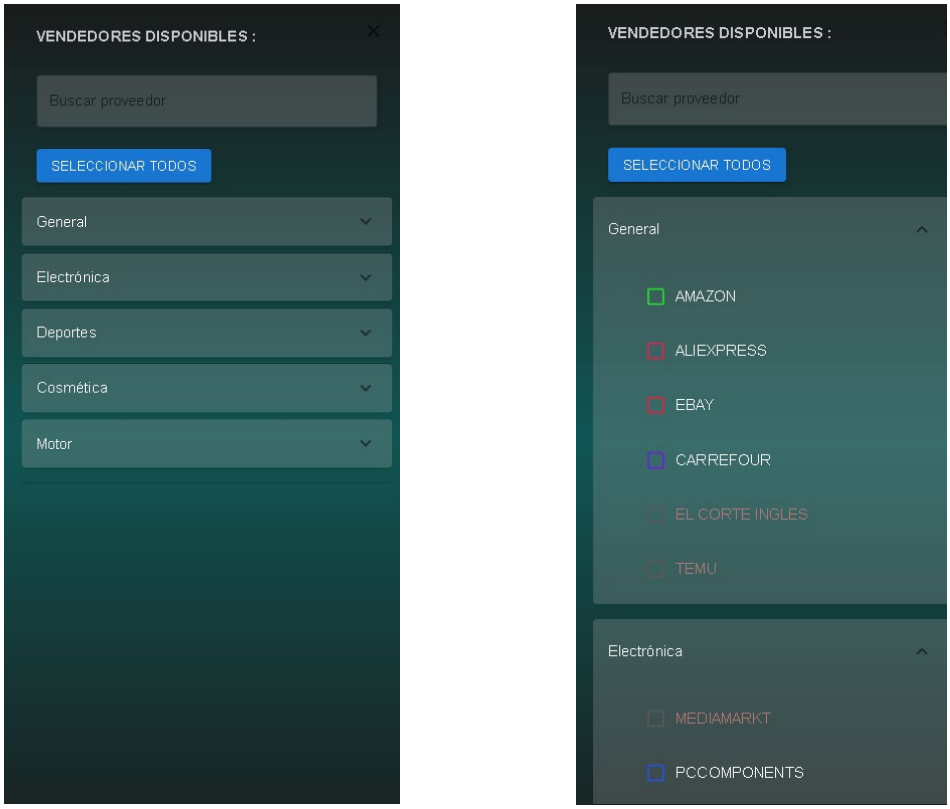


Figura 68: Filtro Proveedores

En caso de que realices una búsqueda sin haber seleccionado ningún proveedor, se mostrará un mensaje de error a la derecha de la ventana:

Figura 69: Error Búsqueda Sin Proveedores

Cuando seleccionas a los proveedores de los que deseas buscar, escribes lo que quieres buscar, y presionas enter, aparece un loader (con una animación) hasta que aparezcan los resultados, y después de un tiempo, aparece la lista de resultados:

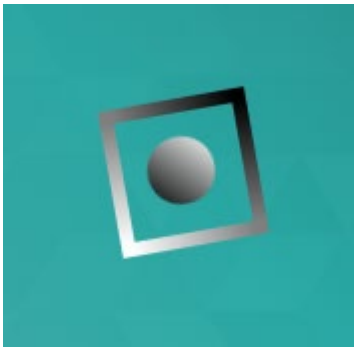


Figura 70: Loader Pausado

Figura 71: Búsqueda Básica Realizada

Aquí, podemos observar la información más relevante de los productos, así como el precio, el título del producto , la imagen, y un logo que n os indica el proveedor al que pertenece el producto

Con la lista de productos ya buscada, aparece un botón a la izquierda que permite desplegar filtros para afinar la búsqueda:

Figura 72: Filtros Productos Post Búsqueda

De esta manera se pueden aplicar filtros , que harán que se actualice aún más la búsqueda de productos.

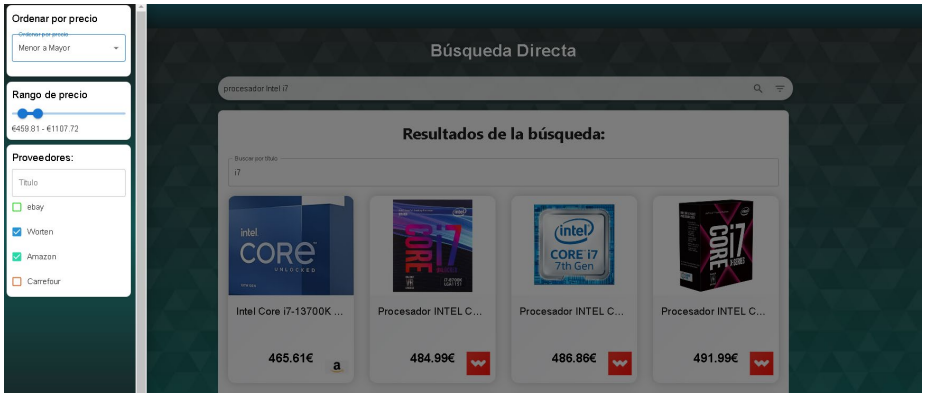


Figura 73: Búsqueda Directa con filtros aplicados

Para el resto de los tipos de búsquedas, es igual , lo único que cambia es la lógica de la propia búsqueda.

8.2.2.- Administrador:

Este manual está diseñado para administradores de SmartCart. Para poder acceder al panel de control de administrador, tendrás que hacer clic en la esquina superior derecha, donde aparece el texto '¿ Eres Administrador? Haz clic aquí' .

Figura 74: Pantalla Principal Acceso Admin

Al hacer clic en esa sección, se abrirá un pop-Up para que el administrador pueda loguearse en la plataforma. Si se intenta acceder de otra manera que no sea está, la página retornará a la inicial. Una vez se acceda con las credenciales, se redigirá directamente a la página con el panel de administrador

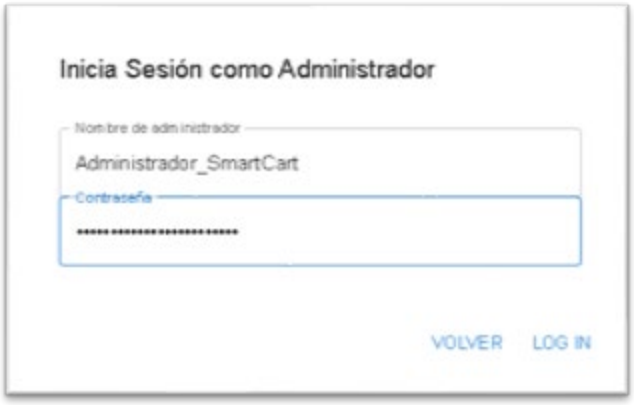


Figura 75: Login Administrador

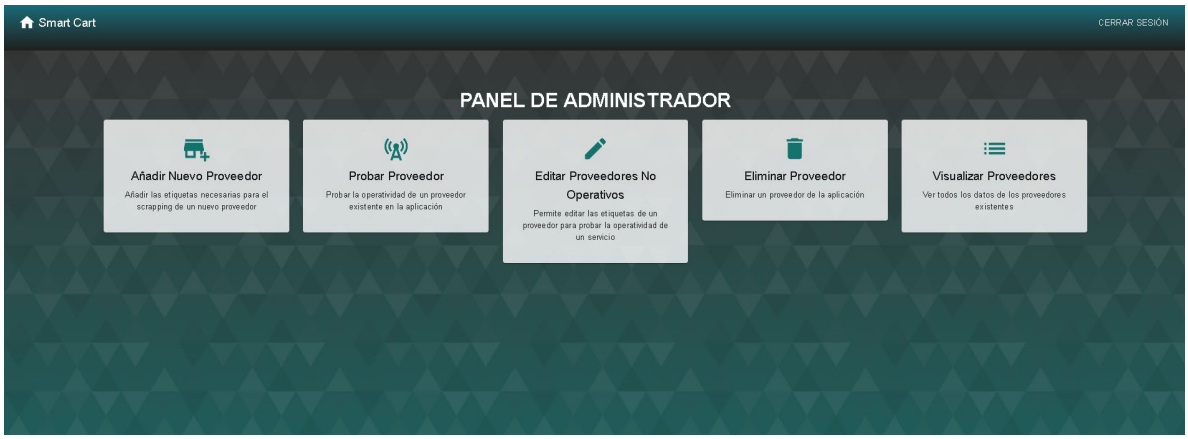


Figura 76: Panel de Control Administrador

En este panel de administrador , el administrador podrá añadir un nuevo proveedor, probar proveedores existentes en la aplicación para cambiar su operatividad, editar los tags de un proveedor existente que no esté operativo, eliminar un proveedor de la aplicación, y visualizar todos los proveedores de la aplicación y la información asociada de estos.

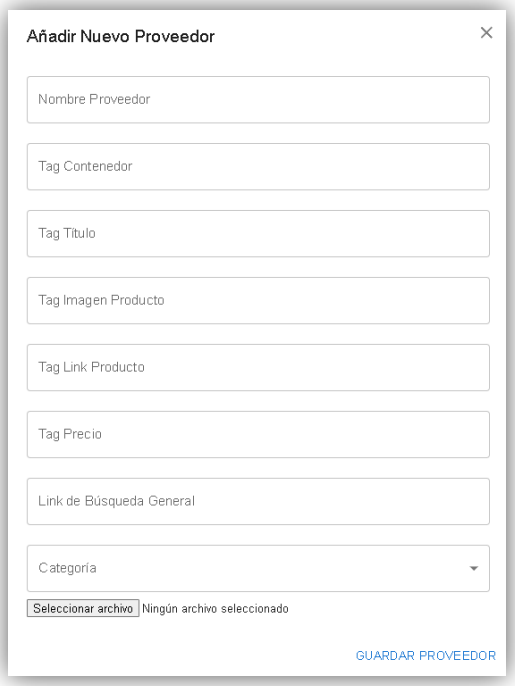


Figura 77: Sección Añadir Nuevo Proveedor

En la sección de probar proveedor, los proveedores que aparecen en verde están actualmente operativos en la aplicación, y en rojo los que no están operativos

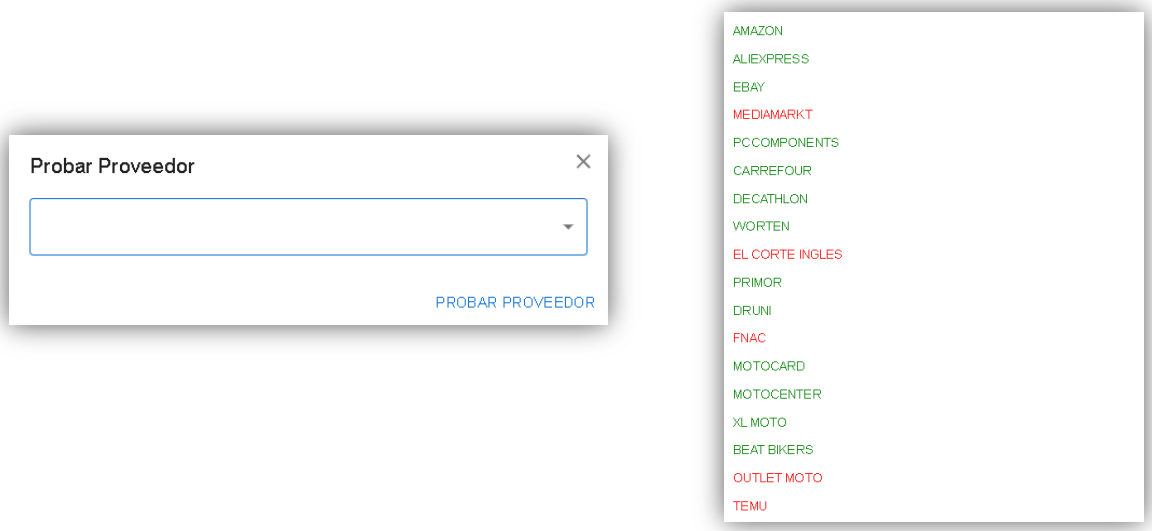


Figura 78: Sección Probar Proveedor

En la opción de editar un proveedor, primero se mostrarán todos los proveedores no operativos para seleccionar uno para editar, y posteriormente el formulario con los datos actuales para editarlos

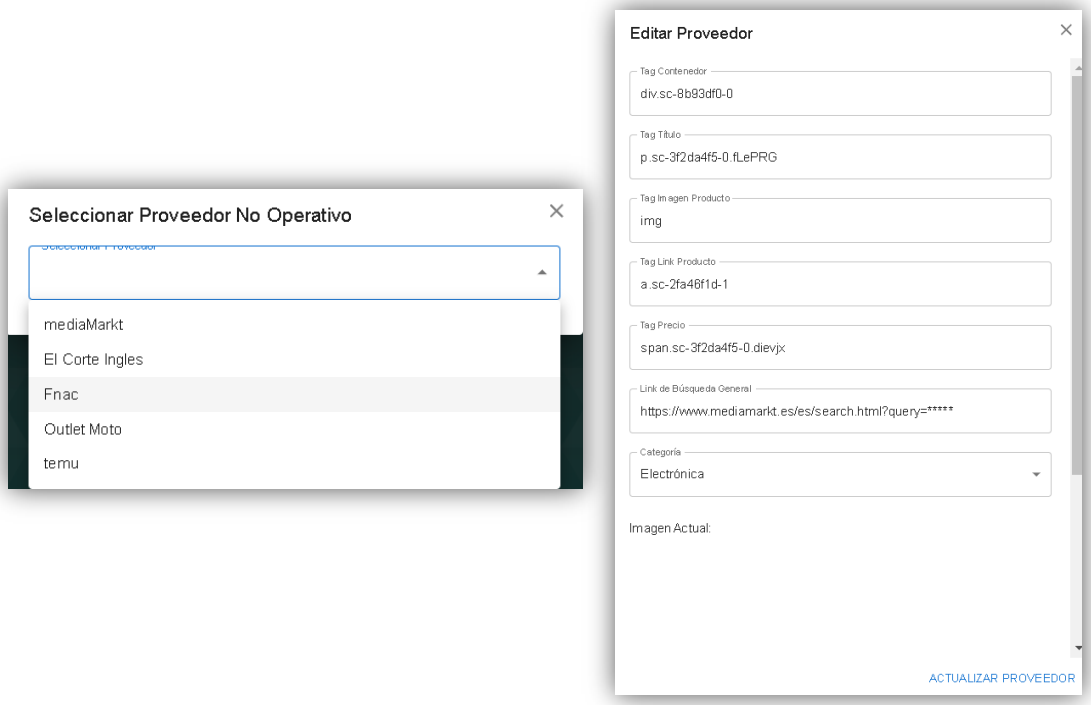


Figura 79: Sección Editar Proveedor No Operativo

Al eliminar proveedores, al igual que para probar proveedores, será un seleccionable para elegir el proveedor a borrar, apareciendo en verde los que están operativos y rojos los que no.

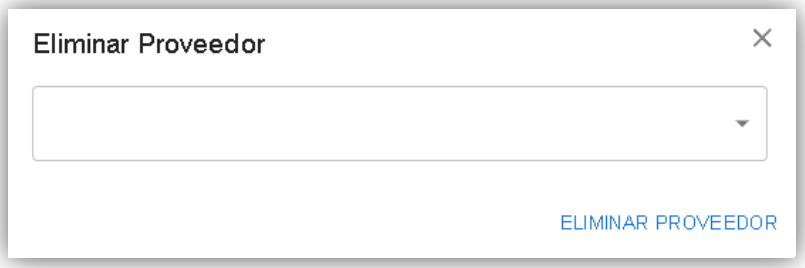


Figura 80: Sección Eliminar Proveedor

Por último, al visualizar los proveedores, aparecerá una tabla con todos los datos de los proveedores de la aplicación.

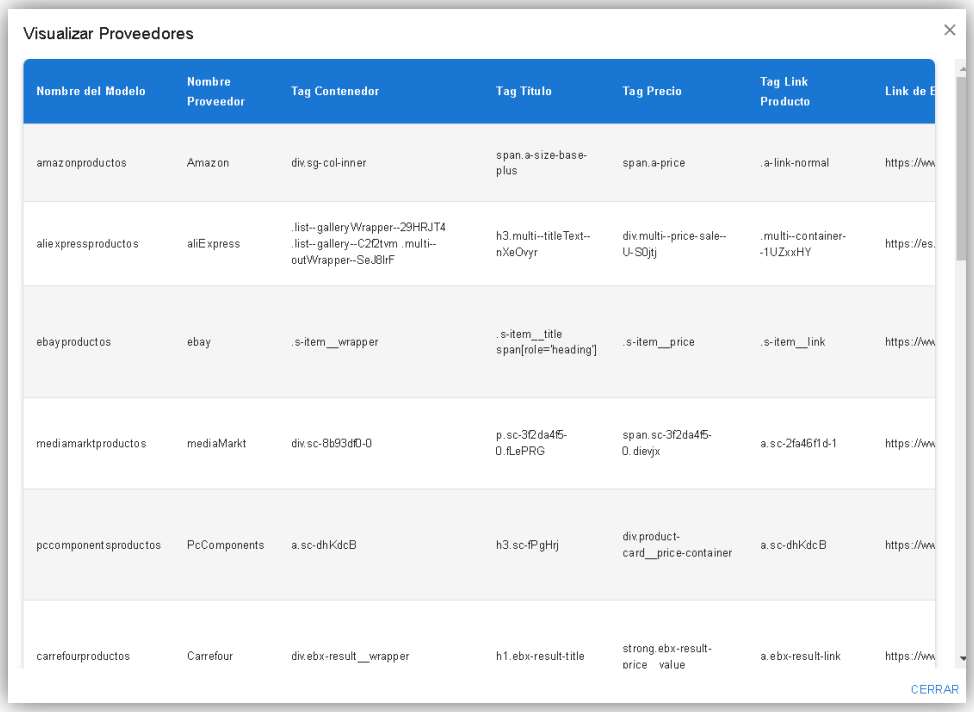


Figura 81: Sección Visualizar Proveedores

Capítulo 9
Conclusiones y trabajo futuro

En esta sección, se exponen las conclusiones derivadas del desarrollo de este trabajo de fin de grado, junto con una serie de posibles mejoras y ampliaciones que podrían añadirse a la plataforma en el futuro.

9.1.- Conclusiones

En conclusión, el desarrollo de esta aplicación web ha cumplido con todos los objetivos planteados, ofreciendo una solución funcional y moderna. El proceso ha sido desafiante, especialmente en la sección de web scraping, ya que ha sido la más compleja y consumió la mayor parte del tiempo.

9.2.- Valoraciones Personales

El desarrollo de esta aplicación ha supuesto una experiencia sumamente enriquecedora y formativa. A lo largo de todo el proceso, he tenido la oportunidad de enfrentar numerosos retos y superar obstáculos que, en un principio, parecían insuperables. Me he visto en la necesidad de aprender y adaptarme a nuevas tecnologías que hasta ahora no había utilizado, lo que ha contribuido significativamente a mi crecimiento tanto a nivel técnico como personal.

Aunque el camino no ha sido fácil y ha habido momentos de frustración y dudas, el resultado final me llena de satisfacción. Puedo afirmar que todos los objetivos planteados al inicio del proyecto se han cumplido, y he logrado construir una aplicación funcional y moderna, capaz de ofrecer una experiencia atractiva al usuario. Me siento particularmente orgulloso de la velocidad conseguida con el scraping, ya que he conseguid recudir tiempos de espera de casi dos minutos a apenas veinte segundos en una búsqueda de más proveedores.

Este proyecto no solo me ha permitido mejorar mis competencias técnicas, sino que también me ha enseñado la importancia de la persistencia, la planificación y la dedicación para alcanzar metas ambiciosas. Estoy convencido de que, con un mayor refinamiento y trabajo continuo, mi aplicación tiene el potencial de ser ampliamente utilizada. Además, con algo de suerte y las mejoras adecuadas, incluso podría llegar a convertirse en una fuente de ingresos en el futuro.

En resumen, este proyecto ha sido una oportunidad invaluable para aprender, desarrollar nuevas habilidades y superar retos importantes. Estoy muy satisfecho con los logros alcanzados y estoy entusiasmado por las posibilidades futuras que podría ofrecer esta aplicación, tanto en términos de utilidad práctica como de impacto personal y profesional.

9.3.- Trabajo futuro

Aunque se han alcanzado numerosos objetivos, existen varias áreas que podrían mejorarse y expandirse para aumentar el valor y las capacidades de la plataforma.

9.3.1.- Usuarios Identificados

Implementar un sistema de usuarios identificados permitiría que los nuevos usuarios registrados puedan guardar productos en su carrito y acceder a una sección personalizada. Esto no solo mejoraría la experiencia del usuario, permitiendo un seguimiento continuo de los productos de interés, sino que también facilitaría recomendaciones personalizadas, mejorando la retención de clientes y permitiendo funciones adicionales como la recuperación de carritos abandonado.

9.3.2.- Guardado de datos más específicos del producto

Actualmente, solo se manejan cuatro datos básicos de los productos. La inclusión de más detalles, como especificaciones técnicas, garantías, políticas de envío y devoluciones, permitiría una comparación más precisa y útil entre productos. Esto enriquecería la experiencia del usuario al ofrecer más información para tomar decisiones de compra informadas.

9.3.3.- Sistema de Carrito y Guardado de Productos

La implementación de un carrito de compras y la opción de guardar productos favoritos mejoraría la funcionalidad de la aplicación. Los usuarios identificados podrían organizar y acceder rápidamente a los productos que más les interesan, creando una experiencia más fluida y personalizada.

9.3.4.- Actualizaciones de precios

Con un sistema de usuarios identificados y productos guardados, sería posible desarrollar una funcionalidad que permita actualizar automáticamente los precios de los productos guardados en intervalos regulares o mediante un refresh. Esto mantendría a los usuarios informados sobre las fluctuaciones de precios y les permitiría aprovechar las mejores ofertas.

9.3.5.- Notificaciones y ofertas

Basado en las actualizaciones de precios, se podría desarrollar un sistema de notificaciones que avise a los usuarios cuando el precio de un producto descienda o alcance un valor objetivo establecido por el usuario. Además, el historial de precios permitiría identificar descuentos inusuales, que podrían destacarse en una sección especial de ofertas, incentivando así la compra.

9.3.6.- Mejoras en la sección de administración

La sección de administración podría ampliarse para ofrecer mayor control sobre la plataforma. Se podrían incluir opciones para gestionar usuarios administradores y los nuevos usuarios identificados para controlar el acceso y sus funciones , y administrar directamente la operatividad de los proveedores. Estas mejoras facilitarían la gestión y la escalabilidad de la aplicación.

9.3.7.- Sistema de Fidelidad

La implementación de un sistema de fidelización podría fortalecer la relación entre la aplicación y los proveedores. Con un programa de fidelización, los usuarios podrían recibir descuentos y promociones exclusivos de los proveedores que más utilicen. Esta estrategia no solo incentivaría la compra recurrente, sino que también atraería nuevos usuarios interesados en aprovechar las ventajas del programa.

9.3.8.- Redes sociales

Integrar la aplicación con redes sociales permitiría promover de manera más efectiva las funcionalidades, ofertas y novedades. Las redes sociales son una herramienta poderosa para atraer nuevos usuarios, generar tráfico y crear engagement con la marca, aprovechando las tendencias de marketing digital y la capacidad de viralización.

9.3.9.- Aplicación móvil

El desarrollo de una versión móvil de la aplicación para Android y iOS optimizaría la experiencia de usuario, permitiendo un acceso más fácil y eficiente desde dispositivos móviles. Con el creciente uso de smartphones para realizar compras online, una app móvil aumentaría la accesibilidad y la comodidad, mejorando el alcance y la satisfacción del cliente

Capítulo 10
Webgrafía

-Node.js:

https://nodejs.org/en

Último acceso: Agosto 2024

-UI:

https://ui.shadcn.com/docs/

Último acceso: Junio 2024

-MongoDB:

https://www.mongodb.com/products/platform/atlas-database

Último acceso: Julio 2024

-React:

https://es.react.dev/learn

Último acceso: Agosto 2024

-GitHub:

https://github.com/

Último acceso: Junio 2024

-Npm:

https://www.npmjs.com/ Último acceso: Julio 2024

-Express:

https://expressjs.com/es/

Último acceso: Junio 2024

-Material UI:

https://mui.com/material-ui

Último acceso: Agosto 2024

-Puppeteer:

https://pptr.dev/

Último acceso: Julio 2024

-Playwright:

https://playwright.dev/

Último acceso: Julio 2024

-Awesome Loader:

https://awesome-loaders.netlify.app/docs/loaders/sunspotloader/

Último acceso: Junio 2024