GRADO EN INGENIERÍA INFORMÁTICA

TRABAJO DE FINAL DE GRADO

DESARROLLO DE UN PLUGIN DE WORDPRESS PARA LA CREACIÓN DE ECOMMERCE

Autora:

Aroob Amjad Ahmed

Supervisor:

Carlos Sánchez Castillo

Tutor académico:

Vicente Ramón Tomás López

Curso académico 2021/2022

Fecha de lectura: 22/06/2022

RESUMEN

Este documento contiene la memoria del Trabajo Final de Grado, desarrollado durante la estancia en prácticas en la empresa llamada Aticsoft por la alumna Aroob Amjad Ahmed.

El proyecto consiste en el desarrollo de un plugin de WordPress que permite crear un sitio web de comercio electrónico de manera sencilla, juntando el proceso de programación y diseño. Con esto, se pretende simplificar el proceso intensivo de la implementación de una página web desde cero.

El proyecto se sitúa en el ámbito de desarrollo de software, tanto frontend como backend, utilizando como herramienta principal WordPress.

PALABRAS CLAVE

WordPress, Plugin, eCommerce

KEYWORDS

WordPress, Plugin, eCommerce

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

Capítulo 1

Introducción

En este primer capítulo, se introduce al proyecto además de la empresa dónde se ha desarrollado éste. Además, se explica la motivación, es decir, el problema que resuelve este proyecto, los objetivos, el alcance y los beneficios.

1.1. Contexto y descripción del proyecto

1.1.1. Introducción a la empresa

El proyecto, cuya descripción se presenta en este documento, se ha desarrollado en la empresa Aticsoft. Se trata de una empresa de consultoría y desarrollo de software a medida para otras empresas. El principal objetivo de la empresa es el diseño y desarrollo web además de marketing digital y desarrollo en la nube.

Aticsoft ofrece varios servicios relacionados a este ámbito de tecnologías. Por ello, para el desarrollo y el manejo de este tipo de productos, cuenta con varios departamentos especializados en distintas áreas. Algunos de estos departamentos son:

- **Comercial/consultor:** dedicados especialmente a la comunicación con los clientes u otras empresas.
- **Programación:** son los programadores de la empresa que se dedican a implementar los productos software, WordPress, web, eCommerce, etc.
- **Diseño:** se trata del equipo de diseñadores web y gráficos.
- **SEO** (optimización para motores de búsqueda, Search Engine Optimization en inglés) **/SEM** (Search Engine Marketing)**:** especialistas en analítica, posicionamiento y optimización web.

Por tanto, teniendo en cuenta la descripción de la empresa, el proyecto de prácticas se sitúa en este ámbito y ha consistido en implementar un plugin para la herramienta WordPress. Un plugin es un programa que añade funcionalidad extra y complementaria a la herramienta WordPress y que permite a los usuarios adaptarla a sus necesidades específicas. El trabajo se ha realizado en coordinación con los departamentos de diseño y programación de la empresa.

1.1.2. Motivación

Hoy en día, el proceso de digitalización está avanzando a gran velocidad. La mayoría de las actividades que se realizaban en el pasado de forma analógica o manual se han transformado a formato digital. Un claro ejemplo es el comercio electrónico. Actualmente, es imprescindible para cualquier empresa tener su propia página web a través de la cual los clientes puedan consultar los servicios ofrecidos o los productos, y, sobre todo comprarlos o contratarlos.

La tarea de una creación de una página web de comercio electrónico no es sencilla y rápida. Se necesita dedicar gran cantidad de horas para verificar su correcto funcionamiento y su diseño que pueda atraer a los usuarios finales. Así pues, una de las fases importantes del modelo de negocio es el diseño para conseguir máxima usabilidad y en poco tiempo. Por esta razón, hay gran demanda de diseñadores y programadores que puedan crear este tipo de productos y lo más pronto posible.

Uno de los productos a los que se dedica a crear y diseñar Aticsoft son estos: páginas web de comercio electrónico. A pesar de que existen varios plugins capaces de crear sitios de comercio electrónico (*eCommerce*) en poco tiempo, la mayoría tienen funcionalidades mucho más complejas de lo que el programador necesita para la creación de la página o están en constantes modificaciones y actualizaciones, lo que dificulta el proceso continuo de desarrollo. Así pues, es necesario crear plugins, como este proyecto, que se ajusten a las necesidades de estos programadores.

1.1.3. Objetivos y alcance

Objetivos del proyecto

El principal objetivo de este proyecto es la creación de un plugin para WordPress que facilite y simplifique la creación de una página web *eCommerce*, ya que puede ser un proceso muy repetitivo por tener la misma estructura en la mayoría de los casos y solo difiriendo en el tipo de producto que se ofrece. Gracias a este plugin, también se consigue una plantilla con un tema de interfaz de usuario, esto es, una configuración que agrega un estilo estético a los componentes visuales de la interfaz, que tan sólo hay que adaptarla a la forma que la empresa desee. Con todo esto, se consigue que los diseñadores y programadores no dependan uno del otro en todo momento y, así, se acelera el proceso en general del desarrollo y diseño.

Adicionalmente, otros objetivos que se desean alcanzar mediante este proyecto son:

- Reducir la carga de trabajo de los programadores y facilitar a los diseñadores la creación de una página de comercio electrónica pudiendo realizar las configuraciones mediante el plugin sin tener la necesidad de apoyarse constantemente en los programadores para implementar ciertas funcionalidades.

- De acuerdo con lo mencionado anteriormente, con el plugin se desea acelerar la creación de este tipo de páginas web y reducir el tiempo que se tarda normalmente en programar una web desde cero.

- Conseguir un estilo de la interfaz de usuario con gran usabilidad.

- Ofrecer al cliente final un plugin en WordPress que permite:
    - Consultar un catálogo de productos, junto con su descripción detallada y precio y realizar compras de éstos.
    - Consultar la descripción detallada y el precio del producto.
    - Realizar compras de productos.
    - Configurar y gestionar productos del catálogo de manera sencilla. Asimismo, al ser un plugin dinámico, se pueden añadir nuevas familias o categorías para clasificar los productos.
    - La creación de propias cuentas de usuario, es decir, el registro. Además, se pueden configurar estas cuentas o modificar los datos personales de manera sencilla.

Objetivos del producto

A nivel estratégico y operativo, el mismo producto desarrollado puede ser utilizado por la propia empresa a la hora de trabajar en los proyectos que estén incluidos en el ámbito de creación de páginas web de negocio electrónico. Como se ha mencionado anteriormente, esto reducirá el tiempo dedicado a desarrollar las funcionalidades comunes en todos sitios *eCommerce* mediante el uso de este plugin y, así poder dedicar más tiempo a los requisitos más específicos y diferentes que pide cada cliente.

Alcance del proyecto

El alcance del proyecto es el desarrollo completo del plugin e incluye como actividad inicial la planificación de éste y un análisis de los posibles riesgos (que se describen en la Tabla 2 del capítulo 2) que pueden ocurrir durante la creación del plugin.

Tras realizar estas dos tareas, se comienza con la implementación de las funcionalidades que utilizará el administrador para crear diferentes tipos de productos, rellenando los campos de información sobre éste, y modificarlos cuando sea necesario. Además, puede visualizar los pedidos realizados por los usuarios.

Las siguientes actividades para el desarrollo del plugin son la implementación de varias interfaces de usuarios, sobrescribiendo los ficheros del tema que esté utilizando el administrador para configurar la estética de la página. Gracias a esto, el administrador conseguirá los elementos visuales que distinguen a una página de comercio electrónico de una simple página web, como por ejemplo las páginas de ver carrito o ver catálogo de productos.

Finalmente, la última tarea es crear un tema simple que el administrador pueda utilizar para dar un aspecto visual más uniforme y atractivo a la página, mediante su activación. Pero, el

alcance de este proyecto no incluye diseñar un tema para el panel de administrador propio ya que WordPress podrá ser utilizado por éste para gestionar los productos y pedidos.

Alcance del producto

Alcance funcional

El alcance funcional del producto incluye la instalación del plugin en cualquier panel de administrador de WordPress para poder trabajar con él. Tras su activación, creará diversos elementos tanto en la parte de administrador como del cliente. Por ejemplo, se crearán los tipos de posts, que son los tipos de entradas o elementos que puede tener WordPress, necesarios para la creación de productos e interfaces como la del catálogo, carrito de compras, pedido o registro de los clientes.

En cuanto a la parte de administración, la persona encargada de gestionar esta sección podrá crear nuevos productos, añadiéndoles un precio, una descripción, título, stock e identificador. Además, podrá añadirles atributos personalizados como por ejemplo los distintos colores, dependiendo del tipo de producto que se trate, e imágenes destacadas. Al finalizar la creación, se añadirán éstos al catálogo o lista de productos y luego podrá modificar su información y borrarlos.

Otro tipo de acciones que podrá realizar el administrador es el manejo de los pedidos realizados por los clientes y configurar la plataforma, es decir, añadir nuevos métodos de pago, de envío o nuevas categorías para clasificar los productos listados. Dentro de este tipo de usuario, existen los trabajadores que solo tienen acceso al manejo de pedidos.

En cuanto a los usuarios compradores, podrán registrarse e iniciar sesión en la plataforma rellenando un formulario con sus datos personales. El registro de la cuenta es necesario para realizar el pago del pedido, pero no para otras acciones.

Otras funcionalidades a las que tendrán acceso son consultar el catálogo de productos, que están clasificados por categorías o filtros. Estos filtros son ordenar los productos por su fecha y precio, ascendentes o descendentes. Además, podrán agregar productos al carrito de compras, ver el subtotal e incrementar o decrementar la cantidad de un producto que está añadido a la cesta o simplemente quitarlo. También existirá la posibilidad de ver la cesta en formato más grande en una página separada. Una vez agregado un producto al carrito, el sistema mostrará un mensaje de confirmación de la agregación.

El alcance del sistema también incluye acceder a información detallada del producto seleccionado, como, por ejemplo, su precio, tamaño, color, imágenes, etc. dependiendo del tipo de producto que sea y los atributos que tenga. Sin embargo, el alcance del producto no incluye añadir productos en su lista de deseos.

Por último, los usuarios podrán realizar el pedido de los productos añadidos al carrito y finalizar la compra rellenando el formulario sobre los datos del usuario, su dirección y método de pago.

Alcance organizativo

El alcance organizativo de este proyecto abarca el estudiante, el supervisor del proyecto y los compañeros de la empresa que pueden ayudar en ciertos aspectos relacionados a la creación del plugin.

Los principales departamentos involucrados en el desarrollo del producto son de programación y diseño. Por tanto, la implementación de la parte de administrador, es decir, la funcionalidad de crear y agregar nuevos productos a la base de datos y la sobreescritura de los ficheros de interfaces se desarrollará como parte del departamento de programación. Por otra parte, el diseño de un tema de WordPress para aportar estilo a la página será del ámbito de diseñadores.

Alcance informático

En el alcance informático, el sistema utilizará una base de datos de WordPress que se estructurará e implementará utilizando MySQL. El sistema seguirá la arquitectura de cliente-servidor, que está montado en WordPress, cuya funcionalidad se implementará en PHP. El alcance informático no incluye la conexión a otros servicios externos. Además, los ficheros subidos al servidor serán accedidos por el programa de cliente FTP (File Transfer Protocol) llamado FileZilla para poder añadir el código de funcionalidades y luego visualizarlos o modificarlos.

1.1.4. Beneficios y breve descripción del producto

El principal beneficio que aporta este proyecto consiste en ofrecer una ayuda a las empresas secundarias o clientes un plugin casi completo y dinámico que puedan utilizar para el desarrollo de sitios web para sus negocios. En caso de que el producto lo esté desarrollando la propia empresa Aticsoft, éstos pueden utilizarla en la creación de la página para su cliente. Además, pueden modificarlo y adaptarlo a sus necesidades o estilos. Por tanto, gracias a la incorporación de este plugin en sus plataformas, no tendrán que realizar el trabajo de desarrollo de una página de comercio electrónico desde cero sino rellenar información y adaptarla a su temática.

Tras la activación de este plugin, la empresa dispondrá de una estructura bien definida de su base de datos, un conjunto de interfaces de usuario y otros recursos imprescindibles para obtener el correcto funcionamiento. Además, se dispondrá de algunos elementos para el administrador para que se pueda manejar y configurar el negocio. Por ejemplo, es el caso de crear un producto y modificar sus campos de información cuando sea necesario.

En otras palabras, el resultado final de este proyecto ha sido conseguir un plugin que pueda facilitar todo el proceso de la creación de un sitio de comercio electrónico, unificando las fases de programación y diseño visual.

1.2. Organización del documento

En esta sección, se explica la organización del resto de documento que describe el desarrollo del proyecto realizado durante la estancia en prácticas: el plugin de tipo *eCommerce*.

En primer lugar, se describe la planificación del proyecto entero, nombrando la metodología utilizada y cada una de las fases. Además, se explica la descripción de los posibles riesgos analizados y el seguimiento del proyecto.

Posteriormente, en el capítulo 3 se explican los recursos utilizados para el desarrollo del plugin. Se mencionan todas las tecnologías, lenguajes de programación y otros programas que han servido o han tenido alguna utilidad para implementar el plugin. Finalmente, se muestra una breve descripción de los costes obtenidos teniendo en cuenta los recursos, humanos y tecnológicos utilizados.

El capítulo 4 explica el análisis y diseño del sistema. En primer lugar, se introducen los requisitos funcionales mediante diagramas de casos de uso y tablas con la especificación de cada. Además, se explican los requisitos de usabilidad y calidad mediante otras tablas. En cuanto al diseño, se explica la arquitectura y la base de datos del plugin a través de figuras. Finalmente, se explica el diseño visual de las interfaces con mockups.

En el capítulo 5, se explican todos los detalles relacionados a la implementación del plugin, partiendo de los requisitos iniciales.

El capítulo 6 se enfoca a la descripción de las pruebas realizadas para comprobar las funcionalidades del plugin junto con un ejemplo para mostrar su funcionamiento.

Finalmente, en el capítulo 7 se presentan las conclusiones técnicas y personales sobre el proyecto en general y los resultados finales.

Capítulo 2

Planificación el proyecto

En este segundo capítulo se realiza una descripción del proyecto en más detalle. En primer lugar, se explica la metodología utilizada para el desarrollo del proyecto. A continuación, se describen algunos riesgos iniciales y, finalmente, el seguimiento del proyecto a lo largo de la estancia en la empresa.

2.1. Metodología

Un proyecto de calidad depende de varios factores. Uno de ellos es la correcta elección de la metodología para conseguir un desarrollo eficiente y continuo. La metodología que se ha aplicado para la planificación de este proyecto, que también se ha estudiado a lo largo del grado, es la metodología predictiva, según descrita en el PMBOK.

Sin embargo, la empresa sigue la metodología *SCRUM* para la planificación y desarrollo de proyectos. Se trata de definir historias de usuario que incluyen funcionalidades a implementar y cada una de estas historias están planeadas a desarrollar en una franja temporal. Además, para realizar seguimiento del proyecto se realizan reuniones llamadas *Sprints* que sirven para comentar los avances, dudas o cualquier detalle importante que haya ocurrido en esa historia.

El funcionamiento de los *sprints* en la empresa depende mayoritariamente del tamaño del proyecto, tipo de cliente y las fechas de entrega establecidas. Algunos proyectos tienen reuniones más frecuentes que otras. Para este proyecto se ha optado por la metodología predictiva, ya que se ajusta más al tipo de objetivos y requisitos que tiene y, así conseguir mejor organización de trabajo.

Sin embargo, tras analizar con la empresa las características específicas de trabajo a desarrollar, como el tiempo o la extensión del proyecto, hace que la metodología predictiva se adapte mejor.

La metodología predictiva consiste en dividir el proyecto en unas determinadas fases y subfases para llevar a cabo las tareas, estableciendo unos límites temporales para cada una de éstas. Se ha elegido este tipo de planificación por ser la más adecuada teniendo en cuenta las características del proyecto. Ya que no se esperan cambios en el alcance del proyecto y los requisitos están claramente definidos antes del inicio del desarrollo, la mayor facilidad de cambio de las metodologías ágiles no será necesarias. Asimismo, se prioriza el ajustarse a los límites temporales y a cumplir estrictamente con los requisitos definidos, por lo que se favorece el uso de metodologías predictivas.

Tras tener los requisitos del producto final definidos claramente mediante una entrevista inicial con el cliente, se identificaron todas las fases y subfases necesarias para el desarrollo del

proyecto. Gracias a esto, se consiguió una visión más clara de cada una de las tareas a realizar junto con los límites temporales en los que se ha de terminar.

Además, para verificar el cumplimiento de cada una de las tareas se planificaron y realizaron reuniones con el jefe del proyecto, es decir el supervisor en este caso, y comentar el avance del proyecto.

2.2. Planificación

En cuanto a la planificación inicial, el proyecto se compone de cinco fases: inicio, configuración y bases de datos, implementación de la funcionalidad, desarrollo de las interfaces de usuario y despliegue. Además, cada una de estas fases contiene sus propias subfases y tareas más específicas, es decir, las funcionalidades del plugin.

En la primera fase, llamada inicio, se ha realizado un estudio, a través de una entrevista al cliente, sobre los requisitos del producto. Una vez realizada la definición de los objetivos a conseguir y la funcionalidad a implementar, se ha elaborado la planificación general de todo el proyecto y se ha diseñado la base de datos para entender la distribución de la información y el aspecto visual del plugin mediante mockups. Como paso final, se ha definido los posibles riesgos que pueden aparecer durante el proyecto, que se explicarán en el apartado 2.3.

La segunda fase se ha dedicado al arranque del proyecto mediante las instalaciones de los programas necesarios para implementar el plugin, el estudio de las funciones de WordPress y otras tecnologías que se van a utilizar. Además, aunque estaba planificado la implementación de la base de datos, no se ha llegado a realizar ya que WordPress ya viene definida con su propia base de datos, bien estructurada y que se encaja en las necesidades del proyecto.

En la tercera fase, se ha realizado toda la parte de programación de las funcionalidades del plugin, utilizando las funciones de WordPress que se estudiaron en la fase anterior, y la creación de ficheros de interfaces. Gracias a la planificación detallada de cada una de las tareas con sus límites temporales, se ha podido realizar un seguimiento sin tener grandes retrasos en la implementación de cada función.

La cuarta fase se ha dedicado a crear páginas adicionales para la página web y crear un tema de WordPress para desarrollar las interfaces de usuario y aportar un aspecto visual más atractivo.

En la última fase, el despliegue, se ha subido el plugin final al servidor de WordPress.

A continuación, se presenta en la figura 1 el diagrama de Gantt resumido y en la tabla 1 se presenta el detalle de la planificación inicial con las horas estimadas. El diagrama de Gantt completo se presenta en el Anexo A.

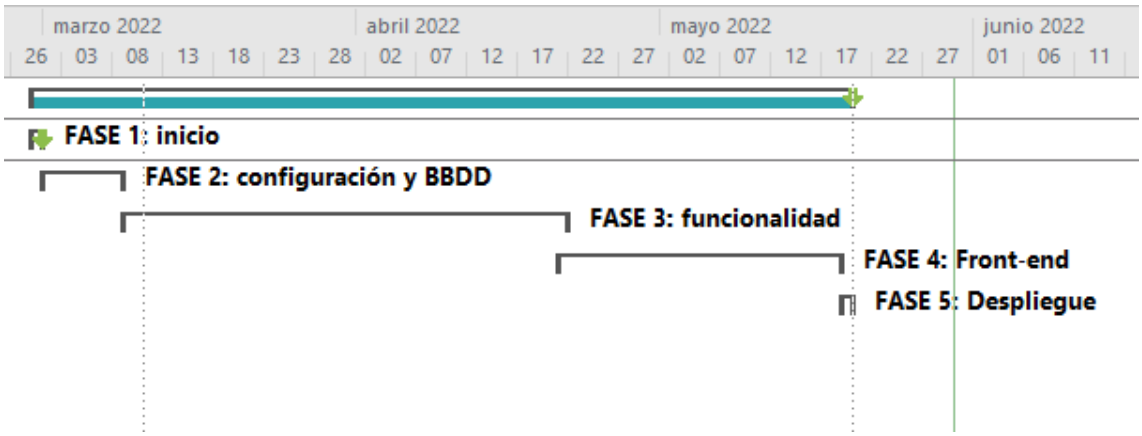

Figura 1: Diagrama de Gantt resumido del proyecto

FASE/SUBFASE/TAREA	HORAS ESTIMADAS
FASE 1: inicio	**5h**
Análisis funcionalidad y requisitos	**1h**
Análisis de riesgos	**1h**
Diseño base de datos	**1h**
Diseño mockups	**1h**
Planificación del proyecto	**1h**
FASE 2: configuración y base de datos	**31h**
Instalación y configuración del entorno (WordPress, xampp)	**5h**
Estudio y análisis de las funciones	**16h**
Implementación de la base de datos	**10h**
FASE 3: funcionalidad	**180h**
Implementación de las funciones - admin	**101h**
Registro post type productos	14h
Registro de metaboxes	60h
Registro post type pedidos	27h
Implementación de las funciones - cliente	**79h**
Inicio sesión	12h
Registro de usuario	12h

Sobreescritura fichero de productos	30h
Sobreescritura página de índice	25h
FASE 4: Front-end	**82h**
Diseño de prototipos	**6h**
Diseño frontend - cliente	**61h**
Página de inicio con filtros	20h
Página de producto	20h
Página de carrito de compras	7h
Página de iniciar sesión/registro	7h
Página de pago/pedido	7h
Diseño frontend - tema	**15h**
Diseño componentes de las interfaces	15h
FASE 5: Despliegue	**2h**
Subida del plugin creado al servidor WordPress	**2h**
TOTAL	**300h**

Tabla 1: Planificación del proyecto, incluyendo la dedicación horaria

Sin embargo, medida que ha avanzado el proyecto, se han modificado algunas fases y tareas. En concreto, se han eliminado algunas tareas como, por ejemplo, las funcionalidades de inicio sesión, modificar y borrar producto, ya que WordPress tiene su propio panel de administración para realizar estas acciones. Por la misma razón, se ha omitido la fase de diseño de interfaces del panel de administrador para manejar los productos y el catálogo, dedicando más tiempo a mejorar las funcionalidades del cliente y su aspecto visual.

En cuanto al límite temporal, la única restricción ha sido realizar todo el proyecto en 300 horas ya que al ser un proyecto para uso propio e interno en un principio no hay un cliente real. Más adelante, en la sección 2.4, se presenta la tabla 3 dónde se puede ver la estimación inicial de las fechas junto a la real.

2.3. Riesgos

Realizar un análisis de riesgos permite conocer las posibles amenazas, sus causas o situaciones inesperadas que pueden afectar al desarrollo del proyecto de alguna manera, como por ejemplo su duración total o el resultado del producto final. Ésta ha sido una de las tareas

iniciales para comenzar el proyecto. Además, junto a su estudio se han propuesto las posibles soluciones para superar estos obstáculos. A continuación, se comentan algunos riesgos que pueden ocurrir durante el desarrollo del proyecto:

- Falta, cambio o fallo en la comprensión de la definición de requisitos.
- Falta de experiencia en las herramientas utilizadas, por ejemplo, WordPress, o la creación de un plugin.
- Cambio de la versión de WordPress que se está utilizando.
- Estimación incorrecta de tiempo, sin dejar un margen de horas por si se producen retrasos en algunas tareas.

La tabla 2 muestra un análisis completo de los riesgos, junto con algunas soluciones:

ID	Riesgo	Consecuencias sobre el proyecto	Disparador	Planes de Prevención
R01	Falta, cambio o fallo en el entendimiento de la definición de requisitos	Creación de un producto incorrecto o incompleto	El miembro no ha realizado un análisis completo y en detalle sobre los requisitos del proyecto	Realizar un análisis completo sobre los requisitos y la funcionalidad a implementar
R02	Falta de experiencia en la herramienta utilizada (WordPress, creación de plugin)	Retrasos en la implementación de algunas tareas del proyecto	El miembro no ha dedicado tiempo en estudiar las herramientas a utilizar	Realizar un estudio sobre la tecnología empleada antes de comenzar a trabajar con ella
R03	Cambio de la versión de la herramienta utilizada (WordPress)	Fallos o incompatibilidad entre las funcionalidades implementadas	Se ha utilizado funcionalidad de la herramienta que ha ido cambiando con el tiempo	Utilizar las funciones principales o *core*, que no cambien de una versión a otra
R04	Estimación incorrecta de tiempo	Retrasos en la entrega final del producto	El miembro no ha creado una planificación correcta con suficiente margen de tiempo extra	Realizar una planificación detallada de todas las fases y tareas, dejando cierto margen de tiempo para posibles retrasos

Tabla 2: Análisis de riesgos

2.4. Seguimiento del proyecto

Tal y como se ha comentado en el apartado 2.1, se sigue la metodología predictiva. Se han dividido cada una de las tareas en fases y subfases para tener mejor organización de la carga de trabajo. Además, la planificación tiene detallada las fechas estimadas para completar las tareas propuestas. Por tanto, para comprobar que el proyecto no sufre retrasos se ha tenido en cuenta el diagrama de Gantt que se presenta en la planificación (ver Figura 1) en todo momento y la tabla con las fechas aproximadas. De esta forma, se ha podido controlar el límite temporal para completar el desarrollo del plugin.

Además, se han realizado reuniones cada martes y jueves con el supervisor para comentar el avance del proyecto y comentar las posibles dudas o cualquier aspecto importante. Las dos reuniones más importantes durante el proyecto han sido la primera y la última. En la primera reunión se tomaron las decisiones importantes como por ejemplo la planificación del proyecto, los requisitos y los objetivos. En la última reunión, se realizó una revisión general de todo el proyecto, comprobando todas las funcionalidades.

Para recapitular el trabajo realizado en las semanas previas y comentar el trabajo que se está realizando actualmente y el que está previsto para los días siguientes, se ha ido redactando un informe quincenal. Este informe también incluía las incidencias o dificultades que han ocurrido durante esos días.

En cuanto al avance real, la tabla 3 muestra las fechas previstas y reales para comparar los ligeros retrasos en algunas fases.

En general, no ha habido mayores retrasos en el proyecto. El primer retraso que ocurrió fue a la hora de implementar los elementos personalizados de WordPress, que se retrasó por dos días. Esto fue debido a que había problemas a la hora de guardar de manera adecuada la información de cada campo para cada producto. Además, en esta fase se necesitó estudiar un poco sobre el funcionamiento y sintaxis de la librería jQuery.

A causa del retraso anterior, hubo retraso de un día comparado a la fecha estimada en implementar los ficheros de sobreescritura, pero se recuperó este tiempo perdido a la hora de crear las páginas adicionales y el diseño de interfaces de usuario, que fue un proceso bastante más sencillo.

Por último, durante el diseño de la página de pedido hubo también retraso de un día ya que al crear el pedido se necesitaba realizar una llamada desde JavaScript a PHP, por lo que se necesitó realizar un estudio para poder utilizar la tecnología AJAX.

FASE/SUBFASE/TAREA	Fecha fin prevista	Fecha fin real
FASE 1: inicio	28/02/2022	28/02/2022
Análisis funcionalidad y requisitos	28/02/2022	28/02/2022
Análisis de riesgos	28/02/2022	28/02/2022

Diseño base de datos	28/02/2022	28/02/2022
Diseño mockups	28/02/2022	28/02/2022
Planificación del proyecto	28/02/2022	28/02/2022
FASE 2: configuración y base de datos	08/03/2022	08/03/2022
Instalación y configuración del entorno (WordPress, xampp)	01/03/2022	01/03/2022
Estudio y análisis de las funciones	04/03/2022	**08/03/2022**
Implementación de la base de datos	08/03/2022	08/03/2022
FASE 3: funcionalidad	21/04/2022	**25/04/2022**
Implementación de las funciones - admin	04/04/2022	**06/04/2022**
Implementación de las funciones - cliente	21/04/2022	**25/04/2022**
FASE 4: Front-end	**18/05/2022**	**19/05/2022**
Diseño de prototipos	22/04/2022	22/04/2022
Diseño frontend - cliente	12/05/2022	**13/05/2022**
Diseño frontend - tema	18/05/2022	**19/05/2022**
FASE 5: Despliegue	19/05/2022	19/05/2022
Subida del plugin creado al servidor WordPress	19/05/2022	19/05/2022

Tabla 3: Comparación de fechas del avance de proyecto

Capítulo 3

Recursos y costes

En este capítulo se describen todas las tecnologías y otros recursos utilizados para llevar a cabo el desarrollo del plugin. Para ello, se explican los lenguajes de programación y las herramientas utilizados en la implementación. Por otro lado, se explica la estimación de recursos humanos y tecnológicos y sus costes finales.

3.1. Tecnologías

3.1.1. Herramientas

El desarrollo de un software involucra distintos tipos de herramientas y lenguajes de programación en función de lo que se desee realizar o implementar en ciertas fases del proyecto. Dependiendo de la extensión del producto el número de tecnologías puede ser o muy extenso o no. A continuación, se mencionan las herramientas utilizadas para el desarrollo del plugin:

- **WordPress:** es la herramienta principal en todo el desarrollo del plugin y sirve para gestionar contenido. También es conocido por su acrónimo CMS (Content Management System) [1]. Es un sistema que permite diseñar y crear páginas web de manera sencilla, sin tener experiencia en programación, y publicar contenido de cualquier tipo (artículos, contenido de multimedia, blogs, páginas de comercio electrónico, etc.). Esta sencillez se debe a que el propio panel de administrador de WordPress permite al usuario descargar pequeños programas que facilitan la labor del diseñador. Estos programas pueden ser plugins, que añaden funcionalidades a la página, o temas, que automáticamente aportan un aspecto visual atractivo.

- **Xampp:** es un servidor web independiente que permite crear y realizar pruebas en tiempo real sobre los programas localizados en el servidor web local para verificar sus funcionalidades. La gran ventaja que aporta es que se trata de un paquete multiplataforma con una base de datos (MariaDB), líneas de comandos y otros servidores [2].

- **MySQL-Front:** es una interfaz que maneja la base de datos MySQL [3]. Aunque el programa Xampp también contiene una interfaz para controlar esta base de datos, su estructura visual es bastante más compleja y completa, con gran cantidad de funcionalidades. Por ello, para este proyecto se ha utilizado este programa, también

llamado *frontend*, ya que contiene un aspecto más simple para realizar las acciones básicas del manejo de las tablas de la base de datos.

- **FileZilla:** es un programa cliente FTP (File Transfer Protocol) que permite subir, modificar o borrar los directorios o ficheros al servidor FTP propio, manejándolo de manera remota [4]. Ha tenido gran utilidad durante todo el proyecto ya que ha sido el lugar dónde se han guardado los ficheros y directorios del plugin, y accedidos desde cualquier dispositivo, sin tener que realizar descargas o subidas adicionales.

- **Visual Studio Code:** es un editor de código fuente simple, muy flexible y compatible con la gran mayoría de lenguajes de programación existentes. Este editor ha sido utilizado para las implementaciones de las funcionalidades del plugin en PHP, HTML, CSS y JavaScript.

3.1.2. Lenguajes de programación

- **PHP (Hypertext Preprocessor):** es un lenguaje de programación de código abierto utilizado para implementar aplicaciones y páginas web. Su principal característica es que es un lenguaje del lado servidor, por lo que favorece a la conexión entre éste y la interfaz de usuario [5]. Además, se puede combinar o incrustar con HTML. Por esta razón, es el lenguaje que se ha utilizado principalmente en el desarrollo del plugin, cubriendo todas las funcionalidades de la parte servidora.

- **MySQL:** es el sistema de gestión de base de datos utilizada para este plugin y también la utilizada por WordPress. Es de código abierto y una de las más utilizadas en general.

  Para este proyecto no se ha tenido que implementar una base de datos desde cero ya que la instalación de WordPress crea una automáticamente con una estructura bien organizada y todas las tablas básicas y necesarias. Además, para este plugin no se ha necesitado añadir cualquier tabla adicional a la base de datos.

- **JavaScript:** es un lenguaje de programación utilizado para implementar las funciones complejas de la parte de interfaz de usuario [6]. Permite crear contenido dinámico dependiendo de la interactividad del usuario con los elementos de la interfaz.

  El uso de JavaScript ha sido muy útil en el desarrollo del plugin ya que ha aportado más dinamismo, sea en el manejo de la interfaz de la parte de cliente a la hora de visualizar los productos del catálogo o para la creación de productos con atributos o campos variados.

- **jQuery:** es una librería de JavaScript que, tal como se ha descrito en el apartado anterior, aporta más dinamismo a las interfaces y así conseguir mejor experiencia de usuario y variedad [7].

  En la implementación del plugin, se ha utilizado jQuery para manipular el DOM (Document Object Model), para la captación de eventos y añadir más interactividad. Por ejemplo, en la creación de un producto, tras pulsar el botón de crear campos, jQuery

capta el evento de click y con el id de este botón comprueba si ha pulsado o no. En caso positivo, llama a la función para agregar un nuevo campo de texto en la interfaz para que el usuario pueda utilizarlo.

- **AJAX (Asynchronous JavaScript and XML):** es un conjunto de tecnologías que se utilizan para implementar aplicaciones web de manera asíncrona. Dicho de otra forma, permite aportar mayor interactividad en las páginas sin necesidad de recargar la interfaz para realizar modificaciones [8].

  Durante el desarrollo del plugin, en ocasiones jQuery no ha sido suficiente para realizar ciertas funciones. Por ejemplo, cuando se desee enviar datos de JavaScript a PHP para realizar ciertos cambios, se ha utilizado AJAX. Por ejemplo, cuando un usuario realiza un pedido, los datos de este pedido se envían por AJAX a la función de PHP. Esta función de PHP es la encargada de recibir la llamada de AJAX y crear el *post type* de pedido con los datos que se acaban de recibir y enviar mensaje de éxito de vuelta.

- **HTML (HyperText Markup Language):** es un lenguaje de etiquetas utilizado para crear páginas web. También es definido como el esqueleto de una página ya que aporta una estructura básica.

  En el proyecto, se ha utilizado HTML para la creación de la parte de interfaz de usuario, es decir, el tema y las plantillas que sobrescriben los ficheros del tema, y para la creación de los *metaboxes.* En otras palabras, cualquier elemento implementado que aparece en la pantalla del usuario es creado por HTML.

- **CSS (Cascading StyleSheets):** es un lenguaje de diseño, que tal como indica su nombre, aporta estilo a cualquier elemento escrito en HTML. Sirve para definir un diseño a las interfaces de usuario o cualquier elemento visual.

  En el desarrollo del plugin, se ha utilizado CSS junto con HTML para mejorar su diseño y dar un aspecto más ordenado. Su uso ha sido mayor en el diseño del tema para cambiar tamaños, colores o para organizar y distribuir mejor los elementos presentes en las interfaces.

- **Funciones de WordPress y Hooks:** las funciones de WordPress sirven para agregar funcionalidades a la implementación, bien para cambiar o ampliar su funcionamiento por defecto [9]. Para que estas funciones puedan activarse en algún momento necesitan engancharse con los llamados *hooks*. Estos *hooks* llaman a las funciones en ciertos momentos o manipulan los datos para alterar el comportamiento sin cambiar el código fuente.

  Durante la programación del plugin, se han utilizado varias funciones de WordPress. Una de las más importantes ha sido "*register_post_type()"* para añadir el tipo de post de productos y pedidos en el panel de administrador de WordPress. Además, los dos tipos de *hooks* existentes, *"action hooks"* y *"filter hook*s", se han utilizado para llamar a estas funciones (ver Figura 2). Por ejemplo, el *action hook* llama a la función *"update_post_meta()"* cada vez que el usuario actualiza los cambios de un producto para guardar la nueva información en la base de datos. Pero, estos ganchos a su vez

también pueden hacer llamadas a las propias funciones implementadas por el programador y no solo limitarse a las funciones predefinidas de WordPress.

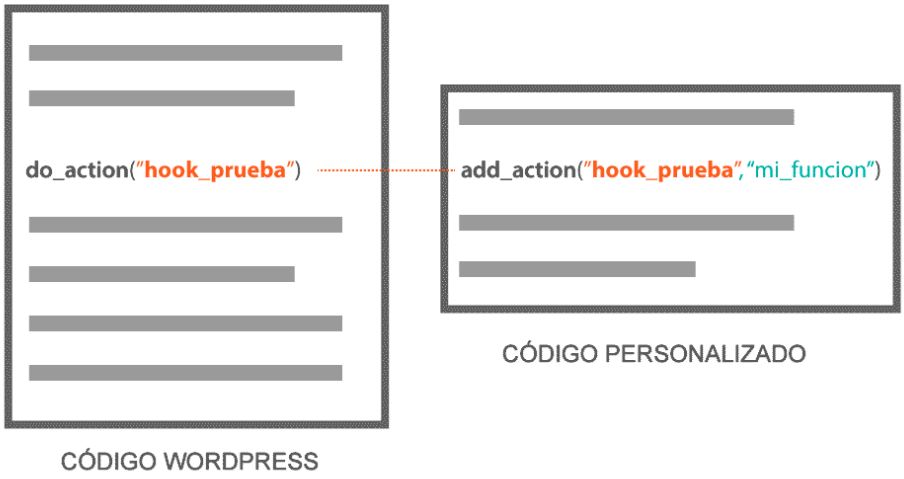

Figura 2: Interacción de los *Hooks* de WordPress con el código personalizado

3.1.3. Gestión de proyecto

- **Slack:** es una aplicación formato web y móvil que se utiliza mayoritariamente en el mundo laboral para enviar mensajes en tiempo real y como un medio de comunicación entre los compañeros y equipos de trabajo. Además, ofrece la posibilidad de integrar otras herramientas o aplicaciones como por ejemplo Google Drive, Google Calendar, Dropbox, etc.

  Slack es una herramienta que se utiliza bastante diariamente en la empresa para comunicar cualquier detalle importante, incidencia o dudas entre los compañeros acerca de los proyectos en desarrollo.

  En el caso de este proyecto, la utilidad de Slack ha sido muy favorable ya que se han podido consultar cualquier duda o pregunta con el supervisor mediante esta aplicación y, así permitir realizar trabajo de forma remota en algunos días de la semana.

- **Microsoft Project:** es una herramienta que se utiliza para la gestión de proyectos. Gracias a este programa, se pueden crear distintos tipos de diagramas y realizar las planificaciones de tareas en más detalle y de forma más organizada.

  Para la gestión de este proyecto, Project ha sido muy útil sobre todo para realizar el seguimiento y planificar las fases. Además, el diagrama de Gantt, que se presenta en la figura 1, también ha sido creado con esta herramienta.

3.2. Estimación de recursos y costes

En cuanto a los costes y recursos utilizados para el desarrollo del proyecto, solo se tiene de momento el coste de empresa ya que en principio está pensado tener el plugin como uso interno y propio de la empresa y no hay un cliente externo. Por tanto, no hay costes externos.

Los costes internos se pueden dividir en tres grupos:

- **Recursos personales:** se trata del programador/diseñador, es decir el estudiante de prácticas que ha desarrollado el plugin. Los costes de un programador han sido proporcionados por la empresa y son de 12 € la hora y teniendo en cuenta que se han empleado 300 horas en la empresa, tal y como estaba en la planificación, el coste total del programador es de unos 3.600 €. Además, en los recursos personales se incluyen los costes del supervisor, que son 15,60 € la hora y, teniendo en cuenta que ha dedicado 60 horas, el coste total sale de 936 €. La tabla representa ambos costes:

Recurso	Salario por hora	Horas	Coste total
Programador	12,00 €	300	3.600,00 €
Supervisor	15,60 €	60	936,00 €
Total			4.536,00 €

Tabla 4: Costes de recursos humanos

Los costes de recursos humanos también incluyen los costes de contratación y los costes indirectos, que son 25 % y 20 %, respectivamente, aplicados al salario del programador. Por tanto, los costes de contratación son 25 % de 3.600 € y los costes indirectos son 20 % de 3.600. El cálculo total sale 900 € y 720 €.

La suma total de los recursos humanos es:

Salario programador + salario supervisor + costes de contratación + costes indirectos =

= 3.600 + 936 + 900 + 720 = 6.153,00 €

- **Recursos hardware:** los recursos hardware utilizados en el desarrollo del proyecto son el ordenador personal del estudiante, el ordenador de la empresa, el coste de internet y el coste del servidor.

  El coste del ordenador de empresa es de unos 1.000 € y tiene una garantía de 3 años. Pero, al ser trabajo semipresencial, solo se ha accedido a la oficina dos días a la semana por lo que su coste se reduce bastante. El proyecto se ha desarrollado en 3 meses. Por tanto, el coste final de este ordenador es:

  1.000 € / 36 meses = 27,78 €/mes = 0,92 €/día
  0,92 €/día x 8 días/mes = 7,36 €/mes x 3 meses = 22,08 €

El coste del ordenador personal sigue criterios similares a los del ordenador de empresa, teniendo un coste de 1.500 € y 3 años de garantía. Este ordenador se ha utilizado de forma diaria para el desarrollo del proyecto, por lo que su coste total de 3 meses es:

1.500 € / 36 meses = 41,67 €/mes

41,67 €/mes x 3 meses = 125 €

Además, al ser trabajo remoto, se incluyen los costes de internet, que son 25 € mensuales, y al ser 3 meses, el total son 75,00 €.

Finalmente, se incluyen los costes de servidor, que son 22,46 € mensuales. En total son 67,38 €.

La tabla 5 resume todos estos costes:

Recurso	Coste mensual	Coste total (3 meses)
Ordenador de empresa	7,36 €	22,08 €
Ordenador personal	41,67 €	125,00 €
Internet	25,00 €	75,00 €
Servidor	22,46 €	67,38 €
Total		289,46 €

Tabla 5: Costes de recursos hardware

- **Recursos software:** en general, todas las tecnologías y programas utilizados no tienen coste de licencia y han sido de uso gratuito excepto la herramienta Microsoft Project, que se ha utilizado para la gestión del proyecto. El coste de licencia de un mes es de 8,40 €. Por tanto, el coste total de 3 meses sale de 25,20 €.

Teniendo en cuenta todos los detalles comentados anteriormente, el coste final del proyecto es:

Recursos humanos + Recursos hardware + Recursos software =

= 6.153,00 + 289,46 + 25,20 = 6.467,66 €

Capítulo 4

Análisis y diseño del sistema

En este capítulo, se realiza un análisis del sistema. Se presentan los diagramas de casos de uso para mostrar las interacciones de los distintos roles con el sistema. A continuación, se presentan las tablas de requisitos y, finalmente se explica el diseño establecido.

4.1. Diagramas de casos de uso

Para este plugin, se han definido tres tipos de actores: el administrador, que maneja el panel de administrador de WordPress; el usuario visitante, que tiene acceso a las interfaces de usuario correspondientes a la tienda; y, el usuario registrado, que además del acceso a las interfaces de usuario tiene acceso a su página de ajustes y la acción de realizar pedidos.

4.1.1. Administrador

El rol del administrador se trata de la persona que maneja el panel de administrador de WordPress. Este administrador puede ser el que crea los productos desde cero o el encargado del mantenimiento y nuevas modificaciones. Por tanto, tiene conocimientos en WordPress pero no necesariamente una formación muy alta o en general en la informática. En la figura 3 se muestran los casos de uso correspondientes a este actor:

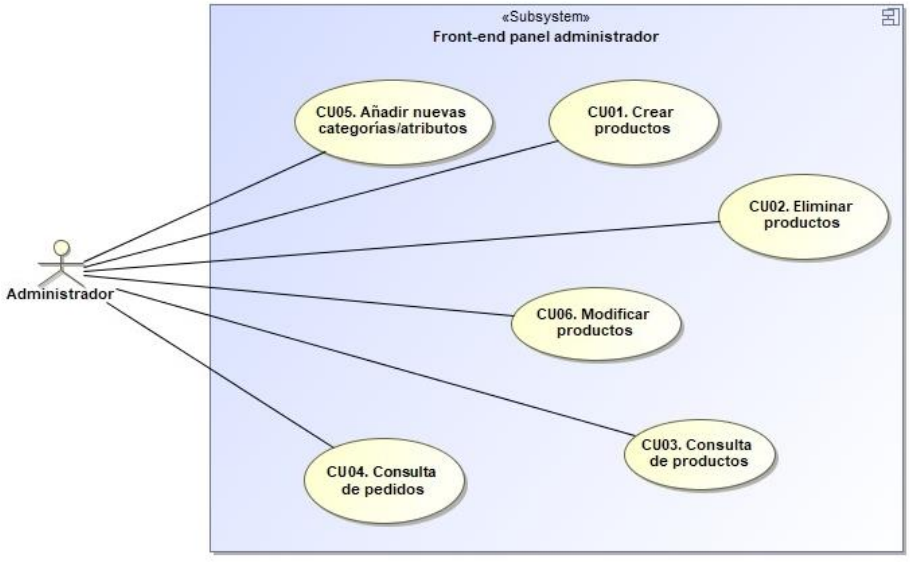

Figura 3: Diagrama de casos de uso del administrador

4.1.2. Usuario cliente

Los siguientes dos roles, usuario visitante y registrado, se pueden agrupar en uno, ya que el segundo también tiene acceso a todas las interacciones que puede realizar el visitante. Pero, algunas funcionalidades solo están disponibles mediante el registro en la plataforma. Estos actores no tienen acceso al panel de administrador de WordPress, sino a las interfaces de la tienda, es decir, consulta de productos y realización de pedidos. Además, no necesitan tener un nivel alto en el uso de ordenador ya que se proporciona un aspecto visual sencillo de utilizar. A continuación, en la figura 4 se detallan los casos de uso de cada uno de los usuarios:

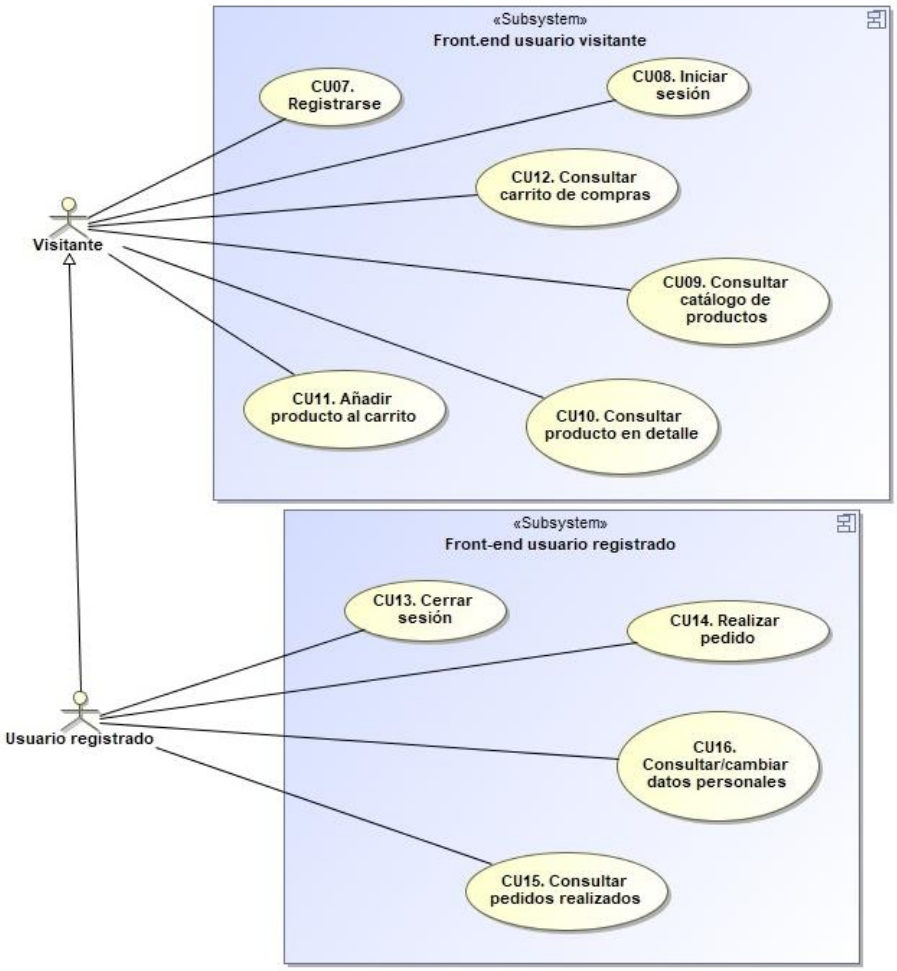

Figura 4: Diagrama de casos de uso del usuario cliente

4.2. Requisitos funcionales

Seguidamente, se especifica los requisitos funcionales con más detalle que se han representado en los anteriores diagramas de caso de uso. Esta especificación se realiza mediante tablas, que contienen la información que se ha considerado más importante.

4.2.1. FR01 – Crear productos

Requisito funcional	
ID	FR01
Nombre	Crear productos
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador crear un nuevo producto.
Precondiciones	Ninguna
Actor principal	Administrador
Secuencia de pasos	1. El actor elige la opción de añadir nuevo producto en la sección de productos. 2. El actor rellena los campos necesarios para la creación del producto. 3. El actor pulsa el botón de “Publicar producto”. 4. El producto se agrega a la base de datos y al catálogo de productos.
Postcondición	Tanto el catálogo como el listado de productos mostrará el nuevo producto añadido junto con los antiguos.
Comentarios	Ninguno

Tabla 6: Requisito funcional FR01 (crear producto)

4.2.2. FR02 – Eliminar productos

Requisito funcional	
ID	FR02
Nombre	Eliminar productos

Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador eliminar un producto del listado de productos.
Precondiciones	El producto seleccionado existe en la base de datos.
Actor principal	Administrador
Secuencia de pasos	1. Se ejecuta FR03, una o varias veces. 2. El actor pulsa la opción de “Eliminar producto”. 3. El sistema envía el producto a la papelera. 4. El actor pulsa la opción de “Borrar permanentemente” en la papelera. 5. La base de datos elimina el producto del listado y catálogo de productos.
Postcondición	Tanto el catálogo como el listado de productos mostrará los productos existentes excepto el eliminado recientemente.
Comentarios	Ninguno

Tabla 7: Requisito funcional FR02 (eliminar productos)

4.2.3. FR03 – Consulta de productos

Requisito funcional	
ID	FR03
Nombre	Consulta de productos
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador consultar el listado de productos.
Precondiciones	Hay uno o más productos ya creados previamente por el actor.
Actor principal	Administrador
Secuencia de pasos	1. El actor pulsa en la opción de “Ver productos” en la sección de productos. 2. El actor selecciona el producto deseado. 3. El actor consulta su información.

Postcondición	Se ejecuta FR02, FR06 o nada.
Comentarios	Ninguno.

Tabla 8: Requisito funcional FR03 (consulta de productos)

4.2.4.FR04 – Consulta de pedidos

Requisito funcional	
ID	FR04
Nombre	Consulta de pedidos
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador consultar el listado de pedidos realizados por usuarios registrados.
Precondiciones	Hay uno o más pedidos ya realizados previamente por usuarios registrados.
Actor principal	Administrador
Secuencia de pasos	1. El actor pulsa en la opción de “Ver pedidos” en la sección de pedidos. 2. El actor selecciona el pedido deseado. 3. El actor consulta su información.
Postcondición	Ninguna.
Comentarios	Ninguno.

Tabla 9: Requisito funcional FR04 (consulta de pedidos)

4.2.5.FR05 – Añadir nuevas categorías/atributos

Requisito funcional	
ID	FR05
Nombre	Añadir nuevas categorías/atributos
Versión	1.0

Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador agregar nuevas categorías o atributos de productos.
Precondiciones	Ninguna.
Actor principal	Administrador
Secuencia de pasos	1. El actor selecciona la opción de “Categorías” o “Atributos” en la sección de productos. 2. El actor selecciona la opción de “Añadir nueva”, dependiendo de si se trata de una categoría o atributo. 3. El actor rellena los campos necesarios para su creación. 4. El actor pulsa el botón de “Añadir”. 5. La nueva categoría o atributo se añade a la base de datos.
Postcondición	Tanto el listado de categorías como atributos mostrará las categorías o atributos existentes junto con el añadido recientemente.
Comentarios	Ninguna.

Tabla 10: Requisito funcional FR05 (añadir nuevas categorías/atributos)

4.2.6.FR06 – Modificar productos

Requisito funcional	
ID	FR06
Nombre	Modificar productos
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al administrador modificar un producto del listado de productos.
Precondiciones	El producto seleccionado existe en la base de datos.
Actor principal	Administrador
Secuencia de pasos	1. Se ejecuta FR03. 2. El actor cambia la información de los campos que desea modificar. 3. El actor pulsa el botón de “Actualizar producto”. 4. La base de datos modifica la información del producto.

Postcondición	La información modificada es visible tanto en el listado de productos como en el catálogo de productos.
Comentarios	Ninguno.

Tabla 11: Requisito funcional FR06 (modificar productos)

4.2.7.FR07 – Registrarse

Requisito funcional	
ID	FR07
Nombre	Registrarse
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir a un usuario visitante registrarse en la plataforma.
Precondiciones	El usuario no está registrado previamente.
Actor principal	Usuario visitante
Secuencia de pasos	1. El actor selecciona el botón de “Registrarse”. 2. El actor rellena el formulario con su información personal. 3. El actor pulsa el botón de “Crear cuenta”. 4. El sistema verifica los datos y agrega la nueva cuenta a la base de datos.
Postcondición	Si la cuenta ya existe o los datos no son válidos, se mostrará un mensaje de error. En caso contrario, se loguea al usuario y se redirige a la página principal.
Comentarios	Ninguno.

Tabla 12: Requisito funcional FR07 (registrarse)

4.2.8.FR08 – Iniciar sesión

Requisito funcional	
ID	FR08

Nombre	Iniciar sesión
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario registrado iniciar sesión en su cuenta.
Precondiciones	El usuario ya está registrado en la plataforma.
Actor principal	Usuario visitante
Secuencia de pasos	1. El actor selecciona la opción de “Iniciar sesión”. 2. El actor rellena el formulario con sus datos requeridos para iniciar sesión. 3. El actor pulsa el botón de “iniciar sesión”. 4. El sistema verifica los datos introducidos por el usuario.
Postcondición	Si los datos son incorrectos o la cuenta no existe, se mostrará un mensaje de error. En caso contrario, se loguea al usuario y se redirige a la página principal.
Comentarios	Ninguno.

Tabla 13: Requisito funcional FR08 (iniciar sesión)

4.2.9.FR09 – Consultar catálogo de productos

Requisito funcional	
ID	FR09
Nombre	Consultar catálogo de productos
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario consultar el catálogo de productos.
Precondiciones	Existen uno o más productos en la base de datos, creados por el administrador.
Actor principal	Usuario visitante
Secuencia de pasos	1. El actor consulta el catálogo de productos de la página principal.
Postcondición	Se ejecuta FR10 o nada.

Comentarios	Ninguno.

Tabla 14: Requisito funcional FR09 (consultar catálogo de productos)

4.2.10.FR10 – Consultar producto en detalle

Requisito funcional	
ID	FR10
Nombre	Consultar producto en detalle
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario consultar la información detallada de un producto.
Precondiciones	Existen uno o más productos en la base de datos, creados por el administrador.
Actor principal	Usuario visitante
Secuencia de pasos	1. Se ejecuta FR09. 2. El actor pulsa el botón de “Ver producto” de un producto del catálogo. 3. El sistema redirige a la página del producto seleccionado. 4. El actor consulta la información del producto.
Postcondición	Se ejecuta FR09, FR11 o nada.
Comentarios	Ninguno

Tabla 15: Requisito funcional FR10 (consultar producto en detalle)

4.2.11.FR11 – Añadir producto al carrito

Requisito funcional	
ID	FR11
Nombre	Añadir producto al carrito
Versión	1.0

Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario añadir un producto al carrito de compras.
Precondiciones	Existen uno o más productos en la base de datos, creados por el administrador.
Actor principal	Usuario visitante
Secuencia de pasos	1. Se ejecuta FR10. 2. El usuario selecciona la cantidad que desea comprar. Si se trata de un producto variante, elige sus atributos. 3. El sistema agrega el producto al carrito de compras. 4. El sistema calcula el subtotal del producto, multiplicado por la cantidad agregada.
Postcondición	Se ejecuta FR09, FR12, FR14 o nada.
Comentarios	Ninguno.

Tabla 16: Requisito funcional FR11 (añadir producto al carrito)

4.2.12. FR12 – Consultar carrito de compras

Requisito funcional	
ID	FR12
Nombre	Consultar carrito de compras
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario consultar los productos agregados al carrito de compras.
Precondiciones	Existen uno o más productos en el carrito, agregados por el usuario.
Actor principal	Usuario visitante
Secuencia de pasos	1. Se ejecuta FR11. 2. El actor selecciona la opción de “Ver cesta”. 3. El sistema redirige al actor a la página de carrito. 4. El actor consulta los productos agregados.
Postcondición	Se ejecuta FR14 o el actor puede modificar la cantidad agregada o eliminar un producto del carrito.

Comentarios	Ninguno.

Tabla 17: Requisito funcional FR12 (consultar carrito de compras)

4.2.13.FR13 – Cerrar sesión

Requisito funcional	
ID	FR13
Nombre	Cerrar sesión
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario desloguearse de la plataforma.
Precondiciones	El usuario está logueado en la plataforma.
Actor principal	Usuario registrado
Secuencia de pasos	1. El actor pulsa el botón de “Cerrar sesión”. 2. El sistema desloguea al usuario.
Postcondición	El sistema redirige al usuario a la página principal.
Comentarios	Ninguno.

Tabla 18: Requisito funcional FR13 (cerrar sesión)

4.2.14.FR14 – Realizar pedido

Requisito funcional	
ID	FR14
Nombre	Realizar pedido
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario realizar un pedido.
Precondiciones	Se ejecuta FR08 y FR11.

Actor principal	Usuario registrado
Secuencia de pasos	1. Se ejecuta FR12. 2. El actor pulsa el botón de “Realizar pedido”. 3. El sistema redirige al actor a la página de pedido. 4. El actor rellena el formulario con sus datos personales. 5. El actor pulsa el botón de “Pagar”. 6. El sistema agrega el nuevo pedido a la base de datos.
Postcondición	Se muestra al usuario la página de pedido con mensaje de confirmación y su información.
Comentarios	Ninguno.

Tabla 19: Requisito funcional FR14 (realizar pedido)

4.2.15.FR15 – Consultar pedidos realizados

Requisito funcional	
ID	FR15
Nombre	Consultar pedidos realizados
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario consultar el listado de pedidos realizados.
Precondiciones	El usuario está logueado en la plataforma y ha realizado pedidos en el pasado.
Actor principal	Usuario registrado
Secuencia de pasos	1. El actor pulsa el botón de “Ver perfil”. 2. El sistema redirige a la página personal del actor. 3. El actor pulsa el botón de “Ver pedidos”. 4. El actor consulta el listado de pedidos realizados.
Postcondición	El usuario, si desea, puede pulsar en un pedido y ver su información detallada.
Comentarios	Ninguna.

Tabla 20: Requisito funcional FR15 (consultar pedidos realizados)

4.2.16. FR16 – Consultar/modificar datos personales

Requisito funcional	
ID	FR16
Nombre	Consultar/cambiar datos personales
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de permitir al usuario consultar o cambiar sus datos personales.
Precondiciones	El usuario está logueado en la plataforma.
Actor principal	Usuario registrado
Secuencia de pasos	1. El actor pulsa el botón de “Ver perfil”. 2. El sistema redirige a la página personal del actor. 3. El actor pulsa el botón de “Ver ajustes”. 4. El actor consulta o modifica los campos de su información personal.
Postcondición	En caso de modificación, el sistema actualiza la base de datos con la información modificada.
Comentarios	Ninguno.

Tabla 21: Requisito funcional FR16 (consultar/cambiar datos personales)

4.3. Requisitos de usabilidad

Los requisitos del sistema también incluyen unos requisitos de usabilidad ya que esta plataforma será utilizada por distintos tipos de usuarios. Para ello, en la implementación del plugin y la implementación de un aspecto visual mediante un tema de WordPress, se han seguido principios de minimalismo y simplicidad. En los siguientes apartados se muestran tablas que especifican los requisitos para cada tipo de interfaces de usuario.

4.3.1. UR01 – Interfaz de productos del administrador

Requisito de usabilidad	
ID	UR01

Nombre	Interfaz de productos del administrador
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de proporcionar una interfaz de administrador simple en el panel de creación de productos de WordPress.
Precondiciones	El administrador está logueado en la plataforma.
Actor principal	Administrador
Secuencia de pasos	1. El administrador entra en la sección de creación de nuevo producto. 2. El plugin carga los metaboxes a la interfaz. 3. El administrador accede a la interfaz y rellena los campos de los metaboxes.
Postcondición	Ninguna.
Comentarios	Cualquier administrador puede manejar el panel de administración, sin tener conocimientos muy altos en la informática o WordPress.

Tabla 22: Requisito de usabilidad UR01 (interfaz de productos del administrador)

4.3.2.UR02 – Interfaz de usuario de la tienda

Requisito de usabilidad	
ID	UR02
Nombre	Interfaz de usuario de la tienda
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de proporcionar una interfaz de usuario de la tienda con gran usabilidad y simplicidad.
Precondiciones	Ninguna.
Actor principal	Usuario visitante.
Secuencia de pasos	1. El actor entra en la plataforma. 2. El plugin y el tema visual cargan los ficheros que muestran las interfaces de usuario. 3. El actor accede a la plataforma.

Postcondición	Ninguna.
Comentarios	Cualquier usuario puede utilizar y desplazarse por la plataforma, independientemente de su nivel de conocimiento en el uso de dispositivos electrónicos.

Tabla 23: Requisito de usabilidad UR02 (interfaz de usuario de la tienda)

4.4. Requisitos de calidad

Además de los requisitos de usabilidad y funcionalidad, el sistema debe tener un requisito de calidad.

La implementación del plugin y el tema se ha realizado utilizando las funciones predefinidas de WordPress. En algunas ocasiones, hay funciones que van quedando obsoletas con el tiempo por lo que no se pueden utilizar. A su vez, WordPress ofrece nuevas funciones que realizan la misma acción, pero con algunas mejoras. Por tanto, uno de los requisitos para este plugin y el tema es el uso de las funciones nuevas y actualizadas para conseguir mayor compatibilidad con la versión de WordPress que se haya instalado. Si en algún caso se utiliza la versión antigua de una función, esto puede causar problemas y, posteriormente, el funcionamiento incorrecto del plugin o el tema.

4.4.1. QR01 – Versión del sistema

Requisito de calidad	
ID	QR01
Nombre	Versión del sistema
Versión	1.0
Autor	Aroob Amjad Ahmed
Descripción	El sistema ha de utilizar funciones de WordPress recientes y compatibles con la última versión actualizada del programa.
Precondiciones	Ninguna.
Actor principal	Servidor/backend.
Postcondición	Ninguna.

Tabla 24: Requisito de calidad QR01 (versión del sistema)

4.5. Arquitectura

4.5.1. Arquitectura del plugin

Para el desarrollo del plugin se ha seguido la arquitectura cliente-servidor. Ya que el plugin funcionará a través de su activación en WordPress y está diseñado especialmente para esta herramienta, su arquitectura sigue el modelo básico de todos los plugins que existen en WordPress.

La figura 5 representa cómo se produce la interacción entre los usuarios y WordPress, siguiendo la arquitectura cliente-servidor:

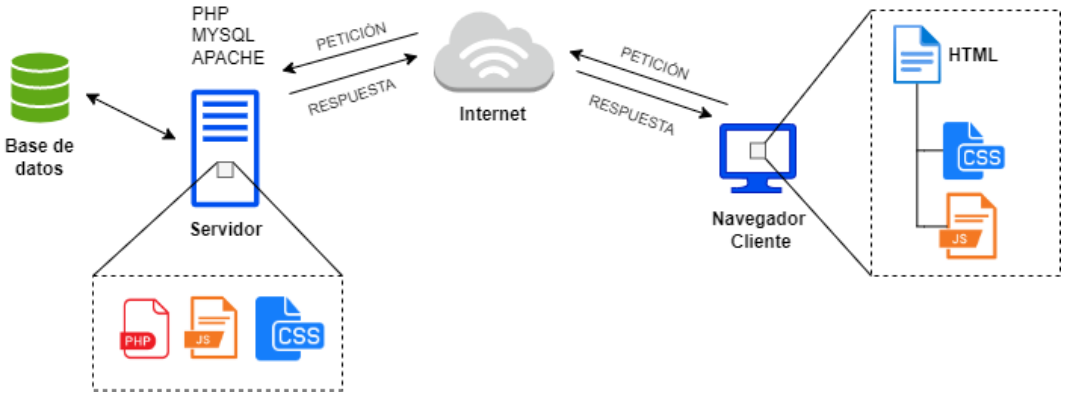

Figura 5: Interacción entre el cliente y el servidor WordPress

La estructura de la jerarquía de ficheros del plugin está formada por un componente importante: el fichero del plugin. Este fichero es que el activa el plugin y conecta a otros ficheros secundarios. Además, es el elemento que debe tener el directorio del plugin como mínimo. También es conocido como el fichero de configuración o funciones.

Por otra parte, el plugin contiene el directorio “inc”, que contiene los ficheros secundarios. Por ejemplo, los ficheros que agregan nuevos tipos de post, metaboxes o que sobrescriben los ficheros de las interfaces. Gracias a estos ficheros secundarios, se han podido realizar modificaciones en las funcionalidades de WordPress y la adición de nuevas, pudiendo obtener nuevas páginas para las interfaces o campos para el panel de administrador.

Para comenzar, el directorio del plugin se encuentra en la carpeta de plugins de WordPress y, el tema diseñado, en la carpeta de *themes*. Además, en el plugin está implementada la funcionalidad se subir imágenes por cada tipo de producto que se crea. Por tanto, estas imágenes se almacenan en la carpeta de *uploads*, una biblioteca de imágenes.

En la figura 6 se muestra la organización de los directorios del plugin y el tema dentro de las carpetas de WordPress:

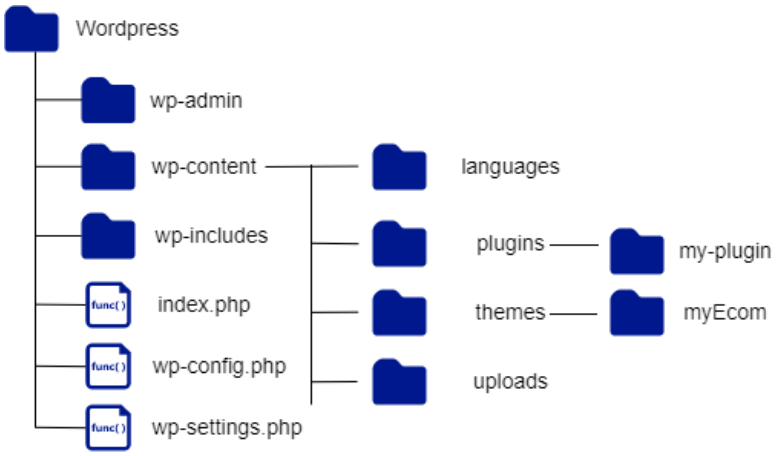

Figura 6: Jerarquía de directorios de WordPress

Tal como indica el nombre, “my-plugin” es el directorio del plugin implementado. Este directorio contiene el fichero principal con el mismo nombre que la carpeta, “my-plugin.php” y el directorio “inc” con el resto de los ficheros secundarios. A continuación, se muestra su jerarquía en la figura 7:

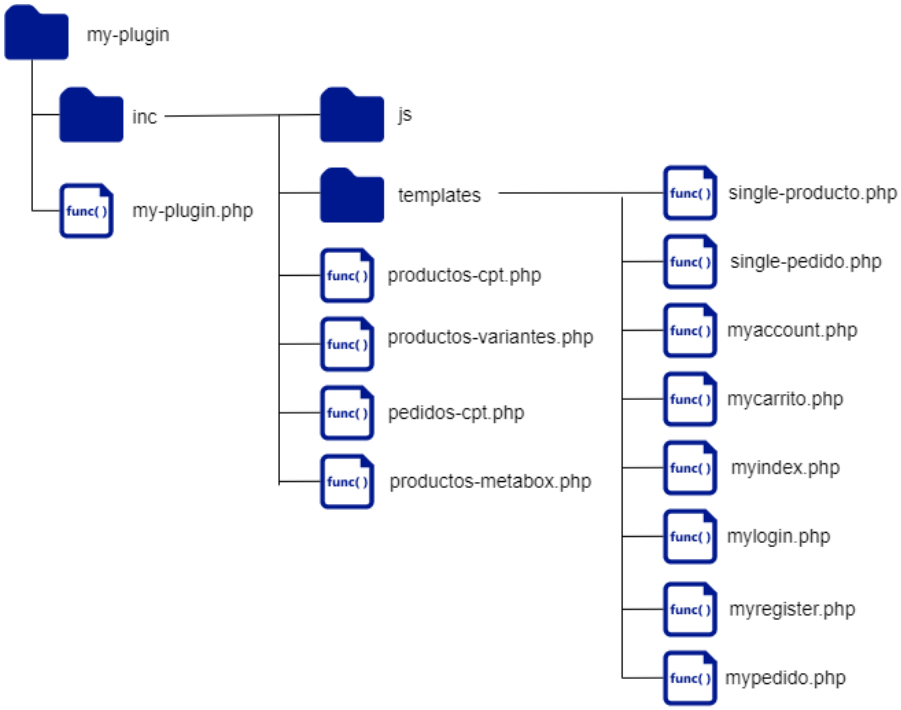

Figura 7: Jerarquía de ficheros y directorios del plugin

Los ficheros presentes en el directorio “inc” son encargados de agregar funcionalidades al panel de administrador de WordPress. Además, los ficheros dentro del directorio “templates” sirven para la sobreescritura de los ficheros HTML del tema visual activo. Por ejemplo, en lugar de mostrar el fichero por defecto “index.php” del tema o theme, gracias al plugin se muestra el

fichero “myindex.php”. Tras la activación del plugin, automáticamente se crean estas páginas por las que se puede desplazar el usuario como, por ejemplo, la página de registro.

Un detalle importante para comentar es que el plugin no contiene un fichero para su activación o desactivación. Para realizar esta función, se selecciona la opción de activar/desactivar presente en la sección de plugins en el panel de administrador de WordPress.

Teniendo en cuenta lo anterior, se pueden apreciar distintas capas que trabajan conjuntamente en la herramienta WordPress. Estas capas son:

- La capa encargada de la base de datos, que se utiliza para almacenamiento de datos a través de MySQL.

- La capa de presentación, que es representada por los ficheros plantilla y del tema, que utilizan código en HTML y CSS.

- La capa de aplicación, que se representa a través de la herramienta de WordPress propia y los ficheros escritos en PHP que agregan funcionalidades adicionales. Estas funcionalidades se enganchan a WordPress mediante *hooks* y *filters,* que se han mencionado anteriormente.

4.5.2. Estructura del tema

La implementación de un tema sigue una arquitectura similar a la de un plugin. En este caso, el fichero importante es el “functions.php”, que contiene las funciones imprescindibles para que el tema comporte de la manera correcta, enganchando los ficheros o componentes de la página. Además, el fichero general de estilos, “styles.css”, también es importante para aportar un aspecto visual ordenado.

El comportamiento del tema de WordPress sigue unos criterios. Trabaja por profundidad de los ficheros, es decir, desde el fichero general “index.php” hasta el fichero personalizado y creado por el propio programador, que no es un fichero básico y por defecto. Por ejemplo, si el fichero “single.php” no existe, que es el fichero encargado de mostrar una entrada de un post, un producto en este caso, el tema redirige al usuario a la página de “404.php”. Ésta es la página encargada de mostrar el error de “página no encontrada”. Pero, si este fichero tampoco existe, se redirige al fichero “index.php”.

A continuación, en la figura 8 se muestra la jerarquía de ficheros del tema creado para este proyecto:

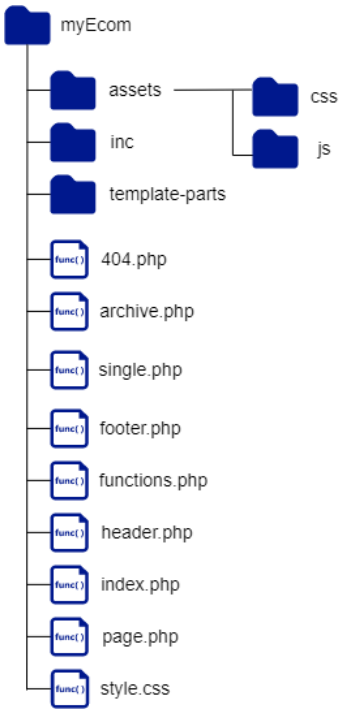

Figura 8: Jerarquía de ficheros y directorios del tema de WordPress

4.6. Base de datos

Una de las mayores ventajas que presenta WordPress es que la herramienta en sí tiene su propia base de datos relacional. Por tanto, en este proyecto no ha sido necesario crear una base de datos desde cero o agregar nuevas tablas en éste ya que las presentes eran suficientes para el manejo de todos los datos.

En cuanto a las tablas, las más importantes han sido:

- wp-posts: esta tabla almacena las distintas entradas creadas para cada tipo de post. En el caso de comercio electrónico, son los productos y los pedidos.

- wp-postmeta: esta tabla almacena la información de cada producto o entrada. En el caso de producto, almacena sus atributos, precio, stock, etc. En el caso de pedidos, almacena el identificador del usuario que ha realizado el pedido, los productos que ha comprado, el subtotal, la dirección, etc.

- wp-users: esta tabla almacena todos los usuarios registrados en la plataforma, incluyendo el administrador.

- wp-usermeta: igual que la tabla “postmeta”, esta tabla almacena los datos relacionados al usuario, como por ejemplo sus datos personales, dirección de envío, listado de pedidos realizados, etc.

La figura 9 muestra el diagrama lógico simplificado con las tablas de la base de datos más utilizadas, la clave primaria de cada tabla y las relaciones entre ellas. En el Anexo B se encuentra el diagrama lógico completo con la descripción de todos los atributos de cada una de las tablas.

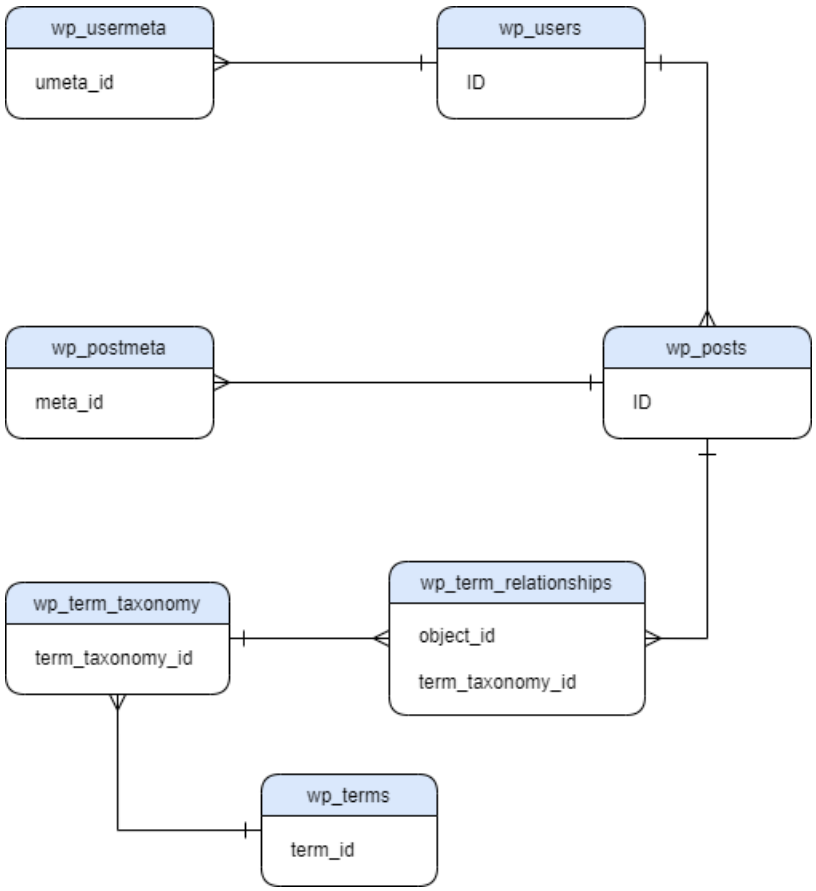

Figura 9: Tablas de la base de datos de WordPress. En el Anexo B se encuentra el diagrama lógico completo con la descripción de todos los atributos de cada una de las tablas.

4.7. Interfaz

4.7.1. Mockups de interfaces

Dentro de la interfaz, se pueden distinguir dos tipos de componentes: los elementos visuales creados para el panel de administración de WordPress y las páginas de la tienda para los usuarios.

Ya que se pretende seguir un estilo sencillo y fácil de usar, para el manejo de productos, se han diseñado los llamados “metaboxes”, es decir, unas cajas con unos campos que se rellenan para añadir información sobre los productos. Los siguientes mockups (ver Figuras 10, 11, 12 y 13) muestran el aspecto visual de estos elementos:

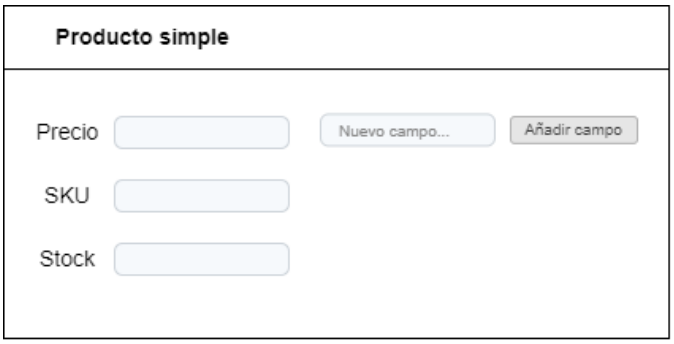

Figura 10: mockup del metabox de producto simple

Para los productos variantes, que tienen varios atributos, se ha diseñado un metabox un poco más complejo. Este metabox se va expandiendo a medida que va aumentado la variedad de un producto, como se puede observar en la figura 12 y en la 13:

Figura 11: Mockup del metabox de producto variante

Figura 12: Mockup del metabox de producto variante con atributos

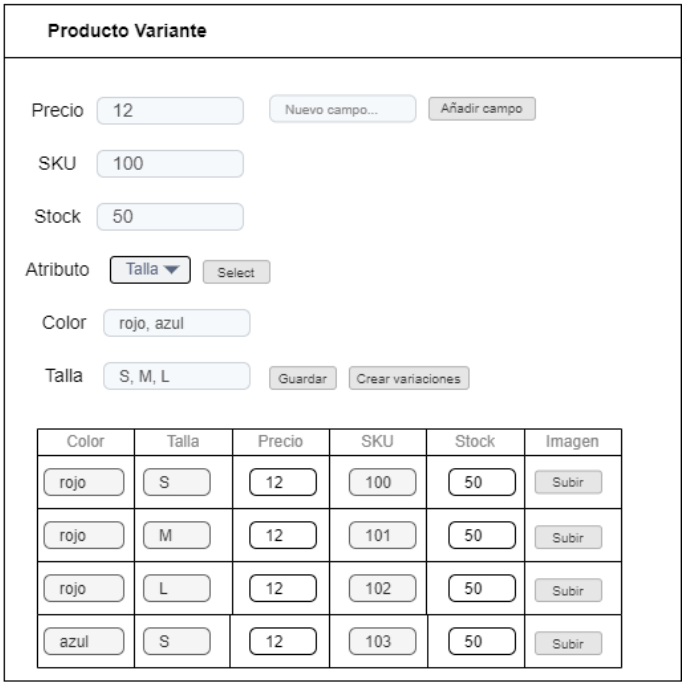

Figura 13: Mockup del metabox de producto variante con la tabla de variaciones

Para el diseño de las interfaces de usuario de la parte del cliente comprador, se ha elegido un estilo que contenga los elementos mínimos necesarios para el funcionamiento de la plataforma para así conseguir mayor usabilidad. Además, se trata de una plataforma en formato web.

Las tres interfaces que tienen un diseño más elaborado son las páginas de catálogo de productos, página de producto y la página personal del usuario registrado. Las figuras 14, 15 y 16 muestran los mockups de estas páginas:

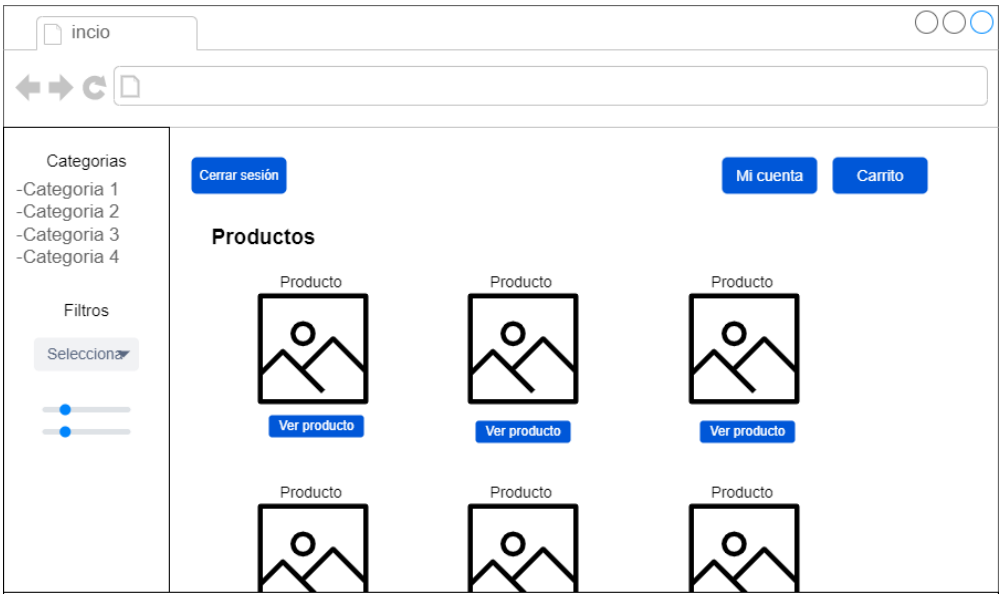

Figura 14: Mockup de la interfaz de catálogo de productos

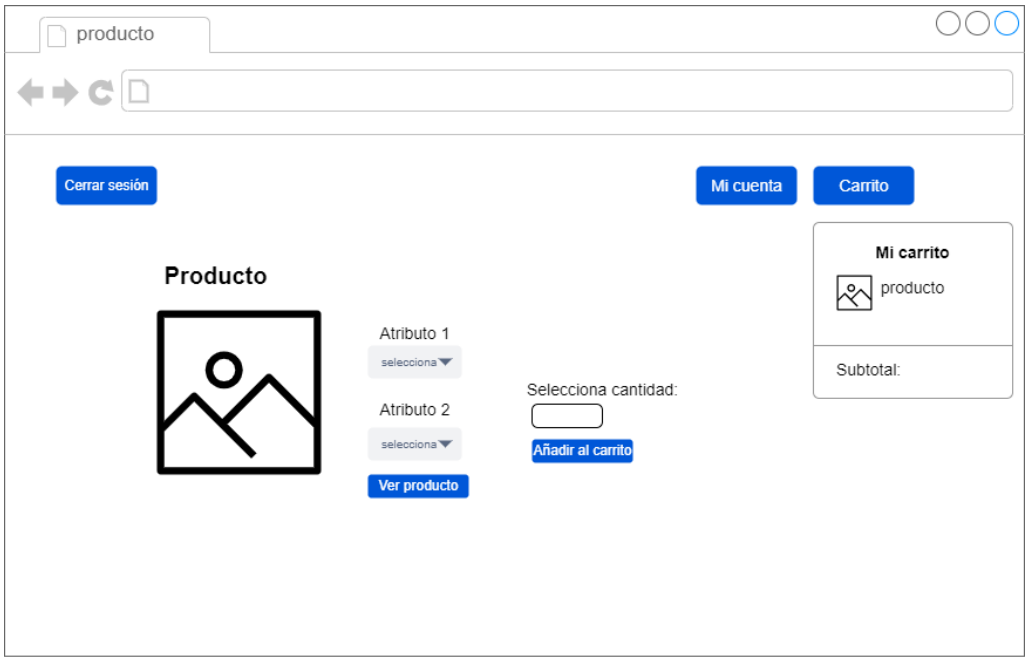

Figura 15: Mockup de la página de producto junto con el desplegable del carrito

Figura 16: Mockup de la pantalla de ajustes del usuario registrado

4.7.2. Guía de estilo

El estándar de estilos que se ha seguido para diseñar las interfaces de usuario son de tipo *Material Design*. La característica importante de este tipo de estilo es la obtención de máxima usabilidad, manteniendo un aspecto estético y atractivo, sin sobrecargar la pantalla.

Paleta de colores

Tal como se ha indicado anteriormente, siguiendo el estilo *Material Design*, los colores elegidos (ver Figura 17) presentan mayor contraste entre los elementos pequeños (botones) y el fondo, que tiene un color neutro. Así, se consigue destacar los componentes que son clicables.

#f9f9f9 #0630b1 #36454f #000000

Figura 17: Paleta de colores del tema

Iconografía

Para no sobrecargar la página con mucho texto, también se han utilizado algunos íconos. En el panel de administrador de WordPress se han utilizado dos íconos para representar la sección de producto y pedidos, respectivamente (ver Figuras 18 y 19):

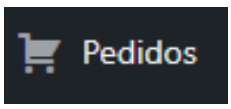

Figura 18: Ícono de pedidos

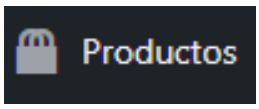

Figura 19: Ícono de productos

Además, para las interfaces de usuario del tema se han utilizado los siguientes íconos representados en la figura 20:

Figura 20: Íconos de las interfaces de usuario

Componentes dentro de la interfaz

Dependiendo del tipo de acción, se han utilizado diferentes tipos de componentes. Estos elementos son distintos tipos de botones. Por ejemplo, para seleccionar los atributos o los productos por criterios de ordenación, se han utilizado botones desplegables. Para seleccionar los rangos de precios, se ha escogido ese tipo de botón. Finalmente, para botones para realizar acciones sencillas, se han elegido los botones normales. En las siguientes figuras 21, 22 y 23 se pueden ver los botones mencionados:

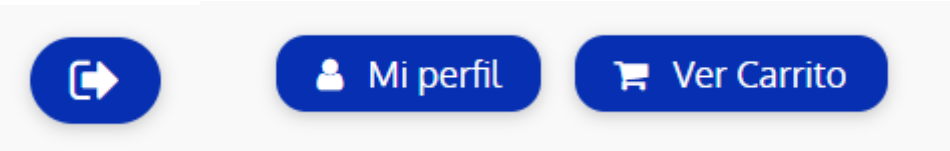

Figura 21: Botones simples de la interfaz de usuario

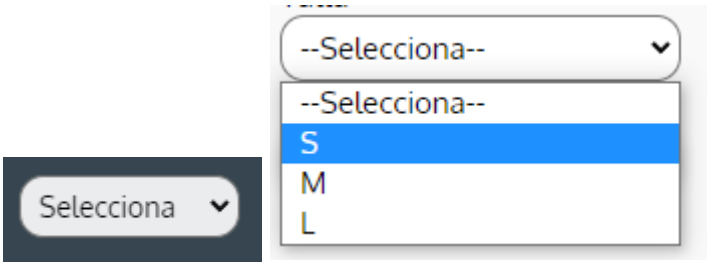

Figura 22: Botones desplegables de la interfaz de usuario

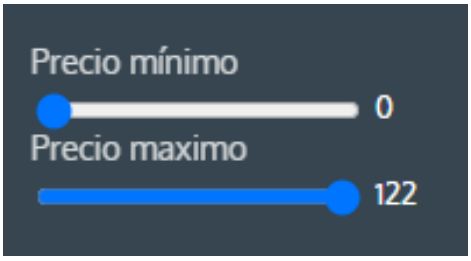

Figura 23: Botones de rangos de números

Capítulo 5

Implementación y pruebas

En este quinto capítulo se describe la implementación del plugin y el tema de WordPress. Para ello, se detallan las decisiones tomadas a la hora de desarrollar ambos y las estrategias seguidas para su obtención. Finalmente, se comenta la metodología elegida para realizar las pruebas y verificar su correcto funcionamiento.

5.1. Requisitos iniciales

Para comenzar el desarrollo del plugin, se ha necesitado instalar el programa más importante: WordPress. Este WordPress ha sido instalado en un servidor que permite acceso a la base de datos y los ficheros importantes de configuración. Tras este paso, se instalaron otros programas secundarios pero que también son necesarios para poder implementar.

En esta fase se ha descargado la versión más reciente de Visual Studio Code, la herramienta que se ha descrito en el capítulo 3, para así poder empezar a crear la jerarquía de ficheros del plugin. Para el desarrollo, además se creó un subdominio del dominio de la empresa (atic.blue.es) para crear el hosting de este plugin.

A continuación, se ha descargado e instalado FileZilla, dónde se han almacenado los directorios de WordPress que además contiene la carpeta recién creada del plugin. Para tener acceso a estos ficheros, se autenticaba con el subdominio creado y la contraseña establecida.

5.2. Detalles de la implementación

Durante la implementación de todo el plugin, el lenguaje de programación que se ha utilizado mayoritariamente es PHP. Esto es debido a que es el lenguaje que utiliza el propio WordPress. Por tanto, para desarrollar todas las funcionalidades del servidor que se comentarán más adelante han sido escritas con PHP, utilizando la librería de funciones predefinidas de WordPress.

Las características de este tipo de plugin, como la estructura o la arquitectura, no permite desarrollar patrones de diseño. Para conseguir implementar una característica similar, se han ido separando las distintas funcionalidades y componentes del sistema en diferentes ficheros, que, finalmente se conectan entre ellos a través de llamadas o el fichero de configuración “my-plugin.php”.

Los tres componentes o conceptos importantes durante el desarrollo del plugin han sido:

- Creación de elementos personalizados (*custom post types*).
- Creación de campos personalizados (*custom metaboxes*).
- Sobreescritura de ficheros del tema visual (WordPress Theme) activo.

Cada uno de estos componentes están claramente separados en diferentes ficheros. Pero, para que el plugin pueda reconocerlos e incluirlos en el panel de WordPress, se han especificado explícitamente en el fichero "my-plugin.php".

5.2.1. Creación de elementos personalizados de WordPress (*custom post types)*

Un *custom post type* es un elemento muy importante en el panel de WordPress [11]. Se trata de un tipo de contenido personalizado que se puede crear y agregar en la barra izquierda de WordPress. Para este plugin, se decidió crear los tipos de post de "Productos" y "Pedidos". Aunque al principio también se pensó crear uno para los "Usuarios", al final no se llegó a realizar ya que WordPress ya viene con esta sección por defecto.

En cuanto a los productos, se pueden apreciar dos tipos de productos: los productos simples y los productos variantes. La diferencia entre ambos está en el hecho de que los productos simples no tienen atributos o variedades. Son únicos. En cambio, los productos variantes de un mismo producto pueden tener diferentes rasgos como color, talla, material, etc. Por tanto, para este plugin se decidió crear el tipo de post de productos simples, que se encuentra en el fichero "productos-cpt.php", y productos variantes, en el fichero "productos-variantes.php".

Por tanto, cada vez que se crea una entrada para el tipo de post de producto, WordPress agrega esta entrada a la base de datos y se puede visualizar en el listado de productos existentes.

Para realizar el registro de estos dos *custom post types*, se utilizó la función de WordPress "register_post_type()" y para engancharla al panel de administración se utilizó el *hook* "add_action". A esta función de registro se le pasó una variedad de atributos que aportan información sobre el *post type*, como por ejemplo su nombre, descripción, etc. En el Código 1 se puede observar dicha implementación.

```
1  function productos_cpt() {
2     $args = array(
3                     'label'          => __( 'Producto', 'text_domain' ),
4                     'description'    => __( 'Los productos', 'text_domain' ),
5                       ...
6     );
7
8     register_post_type( 'productos', $args );
9  }
10
11 add_action( 'init', 'productos_cpt' );
```

Código 1: Registro del *post type* de productos simples

A este punto, había que tomar una decisión importante y era sobre cómo sería el funcionamiento de productos simples y variantes de manera conjunta. Para ello, se siguió la siguiente estrategia: a la hora de registrar el tipo de post de productos variantes, se pasaría a la función la opción de ser invisible en el panel de administrador. De esta forma, a la hora de crear productos, sean del tipo que sean, solo se trabajaría con el *post type* "productos", y este a su vez determinaría si es un producto simple o variante. En el segundo caso, automáticamente crearía las entradas para cada una de las variedades como producto variante.

Para aclarar mejor, se explica el siguiente ejemplo de la camiseta: para crear una camiseta se accede a la sección, es decir, *post type* de "productos". Una vez entrado, se añade la información de precio, stock, etc. Pero, si esta camiseta tiene variedades, es decir, diferentes colores o tallas, el backend automáticamente determinará que se trata de un producto variante y no simple. Por tanto, se generarán entradas para cada tipo de combinación de atributos del tipo "producto-variante", sin tener la necesidad de que el propio administrador vaya a la sección de "productos-variantes" y cree cada producto uno por uno.

Los atributos tienen un papel importante a la hora de distinguir un producto variante del simple. Para ello, se ha registrado lo que se llama en WordPress "taxonomías" [12]. Estas taxonomías son diferentes tipos de clasificaciones que pueden tener las entradas. En este plugin se ha registrado la taxonomía de "atributos". Estos atributos pueden ser cualquier característica que puede tener un producto variante. En el caso de ropa, pueden ser color, talla, material o cualquiera otra que el administrador desee agregar. Además, cada uno de estos atributos tiene sus valores. En el caso del atributo color, sus valores pueden ser rojo, azul, negro, etc. La siguiente figura 24 muestra el resultado final, tras el registro de *post type* de producto y su taxonomía:

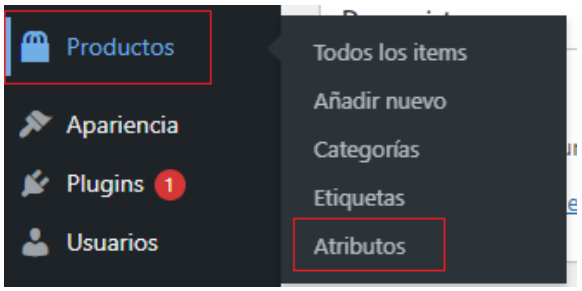

Figura 24: Custom post type de Productos y la taxonomía Atributos

Finalmente, el tercer tipo de elemento personalizado que se ha registrado para el plugin es el de "pedidos", que está implementado en el fichero "pedidos-cpt.php". Este *post type* está visible en el panel de administrador para que esta persona pueda consultar el listado de pedidos existentes. Pero, las entradas del *post type* de pedidos solo se crean cuando un usuario cliente realiza un pedido desde la interfaz de la tienda.

El registro del *post type* de pedidos se ha realizado de la misma manera que los productos simples. En el Código 2 se observa la implementación.

```php
function pedidos_cpt() {
   $args = array(
                  'label'          => __( 'Pedido', 'text_domain' ),
                  'description'    => __( 'Los pedidos', 'text_domain' ),
                   ...
   );

   register_post_type( 'pedidos', $args );
}

add_action( 'init', 'pedidos_cpt' );
```

Código 2: Registro del *post type* de pedidos

La figura 25 muestra el resultado final del panel de WordPress con el registro de todas las taxonomías:

Figura 25: Panel de administrador de WordPress con los nuevos *post types*

5.2.2. Creación de campos personalizados (*custom metaboxes)*

Una vez registrados los *post types,* el siguiente paso ha sido la creación de campos de entrada para agregar diferentes tipos de información dentro del tipo de post de productos. Para ello, se registraron dos cajas o los llamados *metaboxes,* que aparecen por pantalla y recogen información relacionada a un producto [13].

Los *metaboxes* solo se han utilizado para la sección de productos, ya que es lo único que crea el administrador. En este caso, se decidió crear dos metaboxes: uno para la creación del producto simple y otro para producto variante, pero ambos dentro del post type de productos.

Este tipo de componente es más complejo que el registro de post types. Por esta razón, se ha necesitado apoyarse en otros lenguajes de programación ya que es un elemento que no solo trabaja en la parte servidora, pero también aparece por pantalla. Por tanto, para la implementación se han utilizado los lenguajes HTML, CSS y JavaScript, con su librería de jQuery.

Para la creación de este campo personalizado, se utilizó la función predefinida de WordPress llamada "add_meta_box()". A esta función se le pasan cuatro argumentos: el primero es el identificador del *metabox*, el segundo es el nombre que se le asigna al *metabox*, el tercero es el nombre de la función, que se implementó posteriormente, que se encarga de crear la estructura visual del *metabox* utilizando HTML y el cuarto es el nombre del *post type* al que se desea agregar este campo personalizado. En este caso, se desea agregar el *metabox* al *post type* de productos simples. Finalmente, se enganchó la función con el *hook* de WordPress "add_meta_boxes" utilizando el *action hook* "add_action". En el Código 3 se puede observar la implementación:

```
function productos_add_metabox() {
    add_meta_box(
        'productos_box_id',
        'Producto Simple',
        'productos_metabox_html',
        ['productos']
    );
  }

add_action( 'add_meta_boxes', 'productos_add_metabox');
```

Código 3: Registro del *metabox* de productos simples

El *metabox* del producto simple ha sido más fácil de implementar ya que no necesitaba gran variedad de campos. El único componente dinámico ha sido la opcionalidad que se ofrece al administrador de añadir sus propios campos adicionales que quiera. La figura 26 muestra su resultado final:

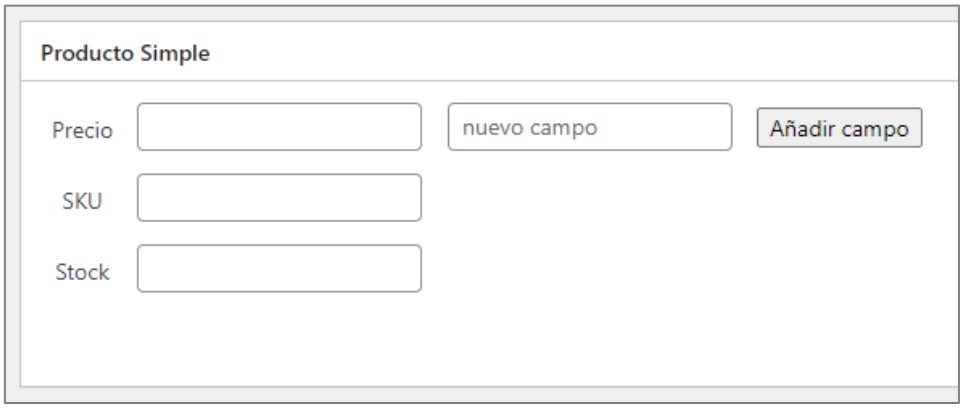

Figura 26: Metabox del producto simple

En cambio, la implementación del *metabox* del producto variante ha tenido ciertas complejidades.

La primera dificultad se ha basado en cómo implementar la funcionalidad de mostrar los diferentes atributos que puede tener un producto variante para que el administrador pueda seleccionarlos y crear las variaciones. Para ello, se decidió mostrar todos los atributos registrados dentro de la taxonomía de "atributos" dentro del propio *metabox*. Además, tras la selección de cada atributo, se permite al administrador agregar los valores e ir guardando.

La segunda dificultad estaba en cómo mostrar las combinaciones de atributos que previamente se han guardado. Aunque la primera decisión fue directamente generar los productos variantes y guardarlos en la base de datos, esto no fue lo óptimo ya que puede darse el caso de que una combinación tenga un precio distinto al otro. Por ejemplo, en el ejemplo de la camiseta, la azul puede tener un precio distinto a la roja.

La solución a este problema fue la siguiente: al guardar los atributos con sus valores, se añade el botón de generar combinaciones que crea una tabla con todas las posibles combinaciones de atributos. Además, esta tabla contiene campos de precio, stock y sku (código de producto) para que el administrador pueda realizar cambios para cada elemento individual. La figura 27 muestra el resultado final de la tabla de combinaciones con el ejemplo de la camiseta:

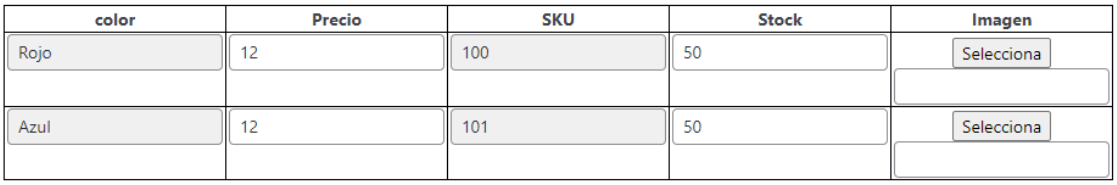

Figura 27: Tabla de variaciones de un producto variante

Finalmente, se decidió añadir más dinamicidad con la opción se agregar imagen por cada producto dentro de la tabla de combinaciones. A este punto también hubo dificultades ya que la subida de una imagen desde el ordenador a la carpeta de "uploads" de WordPress necesita varias operaciones intermedias, sin olvidar que cada una de estas imágenes deben ser adjuntadas como imagen principal de cada producto. En principio, esta funcionalidad fallaba ya que no cogía la ruta completa cuando se seleccionaba una imagen. Para solucionarlo, se optó por la llamada a la función de WordPress que se encarga de la subida de imágenes cada vez que se pulsa el botón de "subir imagen" y todas las operaciones necesarias para su reconocimiento y almacenamiento. La siguiente figura 28 muestra la llamada a "media_handle_upload()" cada vez que se pulsa el botón de "Selecciona":

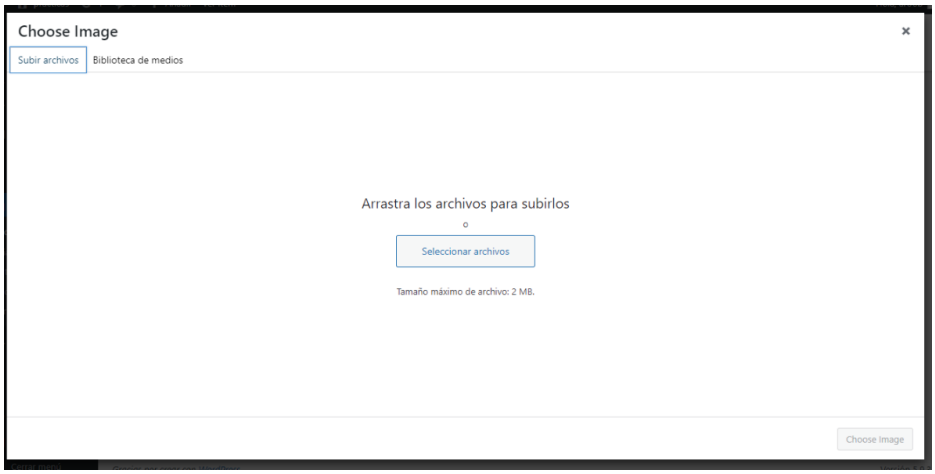

Figura 28: Pestaña de subida de imágenes de WordPress

Tal y como se ha mencionado anteriormente, en esta fase se ha utilizado otros lenguajes como HTML y CSS, además de JavaScript. Para aportar más dinamicidad, se ha utilizado jQuery para acciones como añadir, quitar campos o botones al pulsar otro botón o crear la tabla de variaciones. Por ejemplo, cada vez que se pulsa el botón de "añadir campo" (Figura 29), jQuery realiza la acción de coger el nombre del campo introducido y crear debajo del campo "Stock" el nuevo campo con su título y elemento de entrada de texto. Las figuras 29 y 30 representan esta funcionalidad:

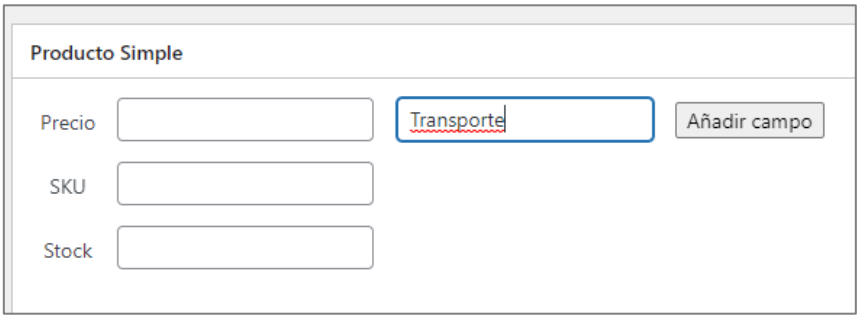

Figura 29: Agregar nuevo campo en el metabox de producto simple

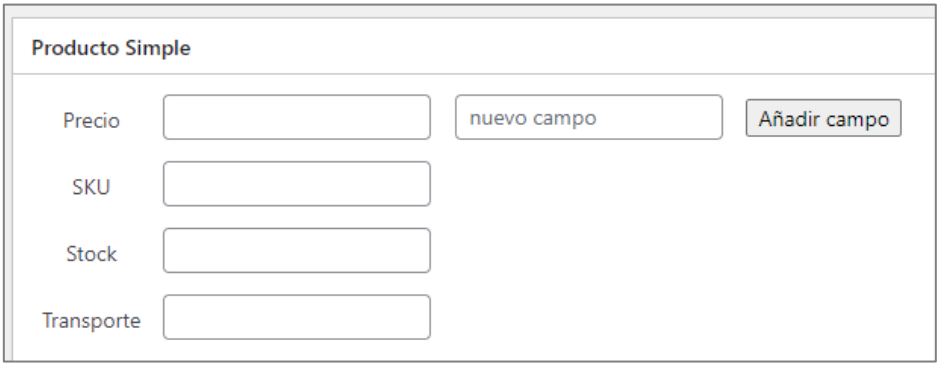

Figura 30: Nuevo campo agregado en el metabox de producto simple

Finalmente, en la figura 31 se presenta el resultado completo del *metabox* entero de productos variantes:

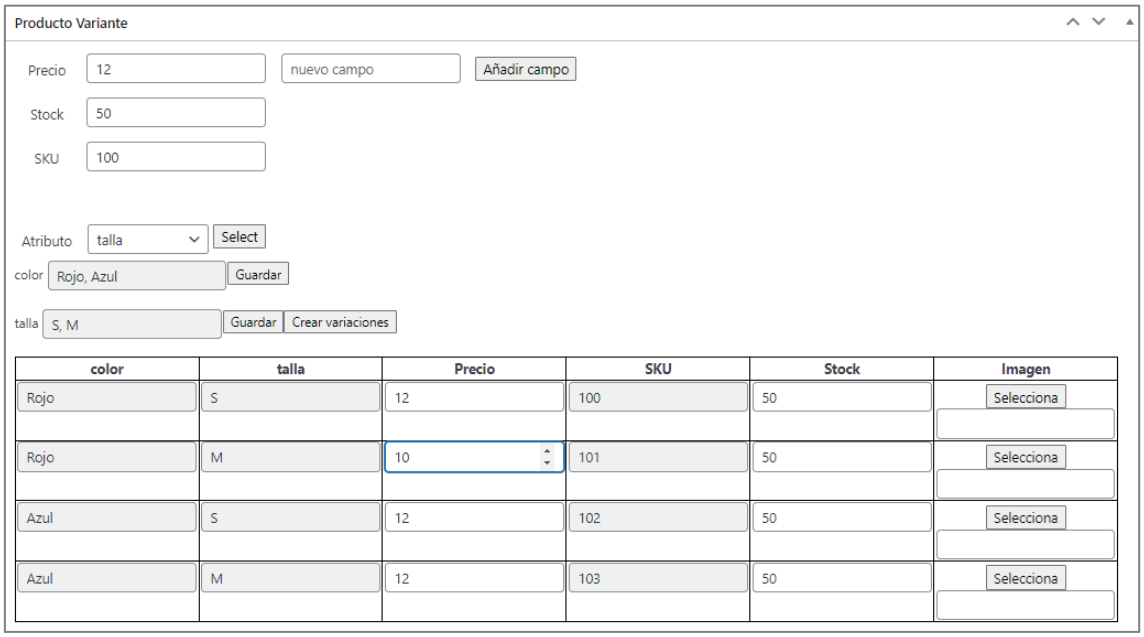

Figura 31: Metabox de productos variantes dentro del post type de productos

El paso final para acabar la implementación de la parte de administrador ha sido el almacenamiento correcto de la información de cada producto creado, sea simple o variante. La función WordPress utilizada para ello ha sido “update_post_meta()”. A esta función se le pasan tres parámetros: el primero es el identificador del producto, el segundo es el nombre o la clave con la que se guardará la información en la base de datos y el tercero, la información que se desea guardar.

En el caso de productos simples, siempre tendrán como mínimo tres claves: el precio, stock y sku. Cada una de estas claves contendrán su información que se añadió previamente en los metaboxes, respectivamente.

El caso de los productos variantes es distinto. En el ejemplo de la camiseta, en la base de datos habrá los tres campos mismos que el del producto simple y los campos adicionales que representan los atributos. Por ejemplo, la camiseta de roja de talla S tendrá en la base de datos las claves “color” y “talla” con sus valores “rojo” y “S”. En esta fase surgió el problema de que todos los productos variantes guardaban los mismos valores. Esto se solucionó mediante la correcta nomenclatura a cada clave de producto. Por tanto, el atributo color estaba guardado junto con su identificador, es decir, “color-1234”. Así, había una clara distinción entre las claves de cada producto sin tener problemas.

5.2.3. Sobreescritura de ficheros del tema visual

Este plugin no solo pretende agregar funcionalidades en el panel de WordPress sino que también aportar una estructura básica de formato de comercio electrónico a través de las interfaces de usuario. Por tanto, cada vez que se activa el plugin, se sobrescriben los ficheros del tema visual que esté activo. Esto se debe a que la estructura por defecto de un tema de WordPress es parecida a un blog con entradas y un menú principal. Pero, para un comercio electrónico no es lo que queremos.

En esta fase destacan dos subfases importantes: la creación de ficheros que sobrescriben los ficheros básicos del tema visual y la creación de otros ficheros adicionales de la temática de comercio electrónico.

En primer lugar, se implementó el fichero “single-productos.php”. Este fichero sobrescribe al fichero “single.php”, que es el utilizado por WordPress por defecto cada vez que el usuario desee ver un producto. El principal cambio que tiene este nuevo fichero al antiguo es que coge los campos del *metabox*, que se han comentado previamente, y los muestra por pantalla. Si el campo es un atributo, muestra un botón desplegable junto con sus opciones, que son los valores de ese atributo.

La implementación del fichero “single-pedido.php” ha sido bastante similar. Pero, en este caso, muestra los campos de información relacionados al pedido. Esto es, la dirección de envío, los productos comprados, la información del usuario que ha realizado la compra, etc.

Por último, se implementó el fichero “myindex.php”, que sobrescribe al fichero “index.php”. Este fichero muestra el catálogo de todos los productos existentes en la base de datos mediante un bucle. Este bucle comprueba que mientras haya productos, los va listando uno por uno, junto con la información básica como título, precio e imagen de cada elemento.

Una cuestión importante en esta etapa ha sido cómo distinguir entre el fichero “single-productos.php” y “single-pedido.php” ya que ambos sobrescriben al fichero “single.php”. Para ello, se decidió agregar dos funciones al fichero “my-plugin.php”. Esta función comprueba cada vez que se abre una entrada y mira qué tipo de post es. Si la entrada abierta es de un post de “productos”, entonces redirige al fichero “single-productos.php”. En caso opuesto, redirige al fichero “single-pedido.php”. Además, si la página abierta es la principal, “is_home()” o “is_front_page()”, entonces redirige al fichero “myindex.php”.

La función, que se puede observar en el Código 4, comprueba si la página actual se corresponde al *post type* de productos y existe el fichero “single-products.php” en la jerarquía de ficheros. En caso positivo, se redirecciona al fichero “single-products.php”. En caso contrario, se redirecciona al fichero original. Finalmente, utilizando el *filter hook* “add_filter” se engancha la función. Se ha utilizado “add_filter” ya que el parámetro pasado a la función se puede devolver modificado en caso de que se trate del *post type* de productos, siendo el nuevo valor la ruta a este fichero.

```php
function redirect_product_template($template) {
    global $post;
    if ($post->post_type == "productos") {
        $plugin_path = plugin_dir_path( __FILE__ );
        $template_name = 'inc/templates/single-product.php';
        if($template === get_stylesheet_directory() . '/' . $template_name  ||
!file_exists($plugin_path . $template_name)) {
                return $template;
         }
         return $plugin_path . $template_name;
    }
    return $template;
}

add_filter('single_template', 'redirect_product_template');
```

Código 4: Redireccionamiento al fichero "single-products.php"

La función del Código 5 comprueba si la página actual es la página principal. En este caso se redirecciona al fichero “myindex.php” y en caso contrario se redirige a la página original. Finalmente, utilizando el *filter hook* se engancha la función implementada.

```php
function index_redirect($template) {
    $plugin_path = plugin_dir_path( __FILE__ );
    $template_name = 'inc/templates/myindex.php';
  if (is_home() || is_front_page()) {
    return $plugin_path . $template_name;
  }
  return $template;
}

add_filter('template_include', 'index_redirect');
```

Código 5: Redireccionamiento al fichero "myindex.php"

Con estos tres ficheros, se consiguió redirigir de manera correcta a la página que tocaba. El siguiente paso ha sido crear los ficheros adicionales. Estos ficheros son elementos importantes para el funcionamiento de una página de comercio electrónico. En concreto, son las páginas de realizar un pedido, ver el carrito de compras, registro, autenticación de usuario y la página de ajustes del usuario registrado.

A la hora de implementar la página de pedidos, había una confusión en cómo representarla a través de la página de “single-pedido.php”. Pero, se decidió separar la página que muestra la información de un pedido realizado y la página de plantilla para rellenar el formulario de pedido. Esta segunda se implementó en el fichero “mypedido.php”, que muestra los campos de dirección, información personal, tipo de pago y los productos seleccionados para comprar. Una vez pulsado el botón de pagar, se crea un *post type* de pedido con la información adjuntada y se lista en la lista de pedidos del usuario. Además, como paso final se redirige al usuario a la página de ”single-pedido.php”, para que pueda ver su pedido realizado junto con toda la información.

La siguiente página que se implementó ha sido la del carrito, en el fichero “mycarrito.php”. Este fichero muestra todos los productos que se han añadido al carrito hasta el momento. Cada uno de los productos se representan mediante su imagen, título, el precio y la cantidad seleccionada. Los últimos dos son modificables en esta página, en caso de que el usuario desee aumentar o disminuir la cantidad de un producto o simplemente quitarlo de su cesta. Finalmente, esta página también muestra el subtotal calculado de todos los productos añadidos.

Para que los usuarios puedan iniciar sesión y registrarse, se implementaron los ficheros “myregister.php” y “mylogin.php”. En esta fase surgió el problema a la hora de iniciar sesión. Cada vez que el usuario se registraba o iniciaba sesión, en lugar de redirigir a la página de índice, se redirigía a la misma página otra vez. Esto se debía a que, a la hora de iniciar sesión de usuario, se estaba utilizando una función de WordPress que estaba obsoleta, por lo que no realizaba el correcto funcionamiento y volvía a la misma página. Esto se solucionó encontrando la nueva función de WordPress, que es “wp_signon()”.

El último fichero adicional que se implementó ha sido el de “myaccount.php”. Esta página ha sido bastante simple en implementar ya que solo muestra información del usuario y se utiliza para realizar pequeñas modificaciones en la cuenta del usuario. En concreto, se utiliza para cambiar la contraseña. Para ello, se ha utilizado jQuery además de AJAX, para conectar el código JavaScript con PHP. La llamada AJAX envía los nuevos datos, es decir, la nueva contraseña. A continuación, la función WordPress recibe esta llamada AJAX y mediante código PHP cambia la contraseña y devuelve un mensaje de éxito a AJAX. Finalmente, al recibir esta respuesta, AJAX muestra un mensaje de confirmación al usuario de que la contraseña ha sido cambiada.

Otra información que se puede cambiar en esta página de ajustes es la dirección por defecto de envío del pedido.

La página de ajustes también muestra el listado de pedidos realizados por el usuario, junto con la opción de ver cada uno de estos en detalle y el estado de cada pedido.

5.2.4. Otros elementos

Durante la implementación de ficheros de interfaces, se han creado elementos dinámicos aparte de ficheros adicionales. Es el caso del pequeño menú desplegable de carrito de compras. Se decidió añadir este elemento en la página de índice y “single-products.php”. Adicionalmente, se ha añadido una barra lateral izquierda en la página de catálogo de productos, para que el usuario pueda ordenarlos o solo ver productos que una categoría específica. Además, si el usuario ha iniciado sesión, siempre se mostrará el botón de cerrar sesión en la pantalla.

En cuanto al usuario, se ha analizado en qué momentos necesita iniciar sesión y cuando no. Y, se llegó a la conclusión de que el usuario puede ver el catálogo de productos, el producto en detalle y la página de cesta sin la necesidad de registro o inicio de sesión. Pero, para realizar el pedido, siempre se necesitará que el usuario tenga una cuenta y haya iniciado sesión.

Finalmente, los últimos días del proyecto se han dedicado a mejorar el aspecto visual. Para ello, se ha creado un tema visual de WordPress (*theme*) dentro de la carpeta de temas. Este tema ha seguido la estética mencionada en el capítulo 4, apartado 4.7.2. La creación del tema ha sido relativamente sencilla, ya que se han creado los ficheros básicos que necesita un tema, sin añadirles mayor complejidad ya que los ficheros de sobreescritura del plugin se encargan de esto.

El tema se ha enfocado más en la escritura de código CSS, para aportar un estilo único a todos los elementos que añadan a la página. Por ejemplo, cada vez que se añade un botón, no habrá que diseñarlo desde cero ya que el tema ya heredará su estética a este elemento.

Capítulo 6

Funcionamiento y pruebas

En este capítulo se explica cómo se ha llevado a cabo la fase de pruebas y se muestra el plugin en funcionamiento con un ejemplo de comercio electrónico de ropa.

6.1. Pruebas

Las pruebas son una fase importante a la hora de implementar un producto software. Estas pruebas nos ayudan a comprender el funcionamiento y encontrar los posibles errores o fallos inesperados.

Debido al tipo de producto desarrollado, no se ha podido implementar una batería de pruebas explícitas para comprobar el funcionamiento del plugin. En cambio, la estrategia que se ha seguido ha sido realizar pruebas de manera manual para comprobar que se cumplen todos los requisitos, con diferentes tipos de entradas o interacciones.

Gracias al tipo de planificación que se ha realizado, separando lo máximo posible e independizando cada tarea de la funcionalidad a implementar, se ha ido probando su funcionamiento a lo largo de todo el desarrollo del plugin.

En el caso del funcionamiento del panel de administrador, se han ido creando diferentes tipos de productos. Para cada producto, se ha agregado gran variedad de tipos de información y características para probar cómo lo maneja, sobre todo para los productos variantes.

Una vez creado el producto, se comprobaba en la base de datos que su título y otra información se había almacenado de manera correcta. Lo mismo se comprobaba para los atributos elegidos para cada producto variante y que cada uno de estos tenga los datos que le tocan. Además, se realizaban las mismas comprobaciones cuando se modificaban los datos como atributos o precios de productos.

Las pruebas realizadas para el catálogo de productos eran comprobar que cada vez que se aplicaba un filtro como ordenar por precio ascendente, los productos se mostraban por pantalla con ese criterio de ordenación.

En cuanto al carrito de compras, la principal prueba se basaba en comprobar que cada vez que se aumentaba la cantidad a comprar de un producto seleccionado, el subtotal también variaba. Además, lo mismo se comprobaba cuando se eliminaba un producto del carrito.

Las pruebas de verificación y validación del registro del usuario eran simples. Se comprobaba mediante los datos rellenados en el formulario que el correo introducido no exista en la base de datos, las contraseñas coincidan y tengan un cierto tamaño como mínimo. En el caso de iniciar

sesión, se comprobaba que la contraseña sea correcta. Las siguientes figuras 32 y 33 muestran estos mensajes de error:

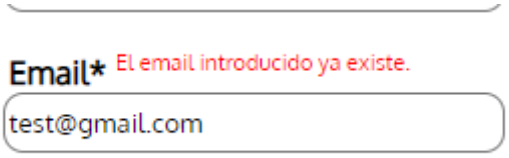

Figura 32: Mensaje de error de cuenta existente en el formulario de registro

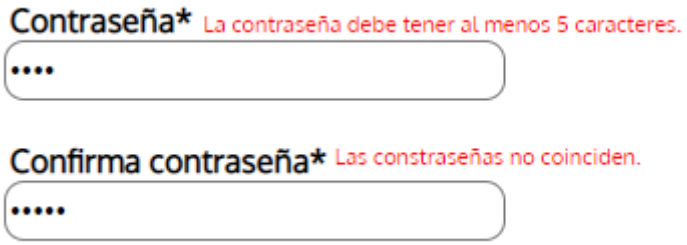

Figura 33: Mensajes de error de contraseñas en el formulario de registro

En cuanto a las pruebas sobre la combinación de producto seleccionado en la interfaz, cada vez que se cambia de atributo, por ejemplo, se elige otra talla, se comprueba que la interfaz muestra correctamente el precio de esa variedad y si tiene, su imagen destacada. Las siguientes dos figuras, 34 y 35, verifican esta funcionalidad:

Figura 34: Interfaz con el producto seleccionado de color verde

Figura 35: Interfaz con el producto seleccionado de color blanco

6.2. Ejemplo de funcionamiento

La prueba final para comprobar el correcto funcionamiento del plugin en general ha sido crear una página de comercio electrónico de ropa. Para ello, se han creado productos de ropa utilizando el *post type* de productos y los *metaboxes* de producto simple y variante. Además, para clasificar a los productos, se han añadido categorías como por ejemplo “ropa”, “camisetas”, “pantalones”, etc.

A cada uno de estos productos se ha asignado una imagen destacada, un título, precio y otra información importante relacionada al elemento.

Tras crear estos productos desde el panel de WordPress, el plugin ha creado automáticamente toda la página frontend de la tienda, sin perder tiempo en diseñarla desde cero. La figura 36 muestra la navegación de un usuario por todas las pantallas de la tienda, comenzando desde la página de registro o entrada hasta la página de pedido y en el Anexo C se representan cada una de las interfaces de usuario en más detalle.

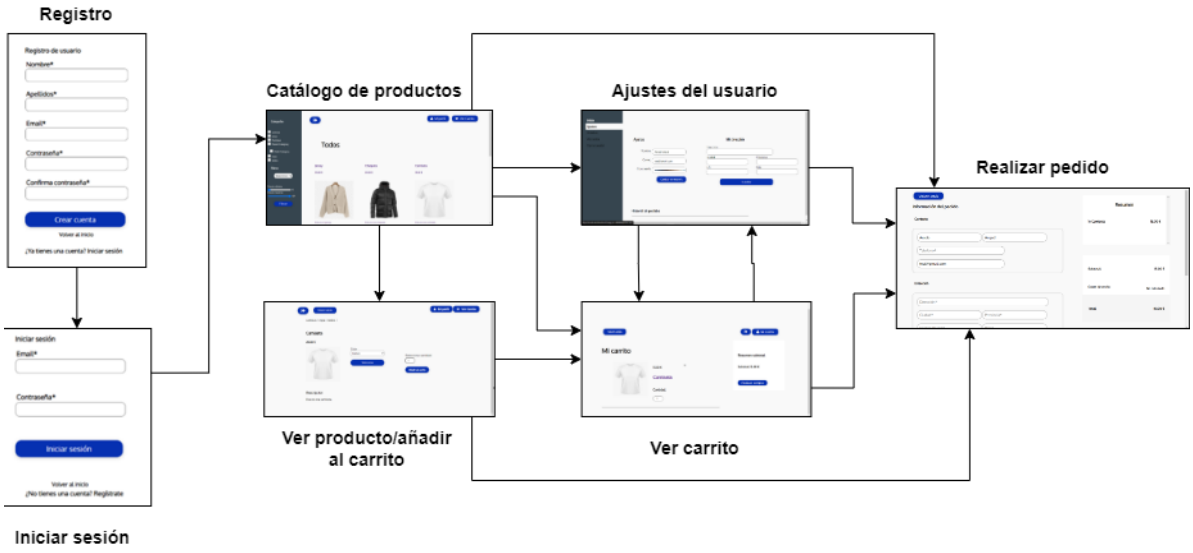

Figura 36: Diagrama con navegación por las interfaces de usuario

Capítulo 7

Conclusiones

En este último capítulo se detallan las conclusiones tanto técnicas como personales sobre el trabajo realizado durante el desarrollo del proyecto. Además, se reflexiona sobre los resultados finales obtenidos.

7.1. Técnicas

Para concluir, se puede decir que, en general, se ha podido implementar todas las funciones que se habían propuesto y marcado como requisitos finales del producto.

La experiencia trabajando con WordPress ha resultado bastante positiva en general. Una de las mayores ventajas que presenta esta herramienta es la reducción de tiempo en diseñar páginas web. Las personas con poca experiencia en el ámbito de desarrollo de páginas web también pueden utilizarla. Aunque en este proyecto se ha elegido el método de programación para diseñar la parte visual de las interfaces existen plugins que ayudan a diseñar los elementos de la pantalla. De esta forma, se puede conseguir un diseño en más poco tiempo, sin necesidad de escribir código CSS.

A lo largo del desarrollo del plugin, no han ocurrido graves problemas técnicos que hayan causado impactos negativos sobre el proyecto en general. Algunas pequeñas incidencias técnicas durante la programación han sido errores de código por falta de experiencia en el lenguaje de programación o conceptos relacionados. Pero, gracias a la ayuda de compañeros y el supervisor, han sido solucionados en poco tiempo y no han retrasado el desarrollo del proyecto.

Además, el uso de diferentes lenguajes de programación ha permitido obtener un plugin más dinámico y con funcionalidades adaptables y extensibles, dependiendo del tipo de administrador que lo esté utilizando o el tipo de producto al que esté destinada la página *eCommerce*.

7.2. Personales

Desde la perspectiva personal, la experiencia ha sido totalmente nueva. Siendo este mi primer contacto con el mundo laboral, al principio experimenté nervios ya que nunca había trabajado en este tipo de proyecto y por los conocimientos en las herramientas utilizadas. Sin embargo, con el tiempo estos nervios desaparecieron al conocer más a la empresa, a la forma

en la que trabajaban los compañeros y al supervisor, que tenían unas actitudes muy positivas y amables. Por tanto, se puede afirmar que todo ha funcionado bastante bien comparado a lo que pensaba al principio.

Una de las razones por las que la forma de trabajar en el proyecto ha sido productiva es porque habían algunos días de la semana de teletrabajo y otros presencialmente en la oficina. En los días no presenciales, trabajando en la implementación del plugin era una experiencia parecida a la forma de trabajar para proyectos de clase, siguiendo una rutina parecida a la de la vida como un estudiante. Los otros días en la oficina me ayudaban a entender cómo funciona el mundo laboral y el flujo de trabajo de los compañeros de oficina. Por tanto, la combinación de ambos me ha ayudado a entender el funcionamiento de una oficina poco a poco, sin tener un cambio radical en mi rutina diaria.

En cuanto al proyecto, gracias a esta experiencia he aprendido bastante a nivel técnico como programadora. Este aprendizaje incluye los distintos lenguajes de programación como PHP o JavaScript, entre otros. Además, he llegado a conocer nuevas herramientas como WordPress u otros programas que han servido de apoyo en la implementación del plugin, como MySQL-Front (ver sección 3.1.1).

Al ser mi primera vez desarrollando un plugin, estoy bastante satisfecha con el resultado final. Ha habido momentos en los cuales me he enfrentado a errores de programación sin saber cómo solucionarlos. Pero, dedicándoles más tiempo y prestando atención, he conseguido solucionarlos. Por esta razón, también he mejorado en la habilidad de resolver problemas por mi cuenta y en la depuración del código.

Por último, cabe destacar que los contenidos aprendidos durante la carrera me han ayudado para enfrentarme a este tipo de proyecto, sean a nivel técnico o gestión de un proyecto. Pero, la principal diferencia que he notado ha sido que los proyectos de clase eran más limitados que los proyectos del mundo laboral. Gracias a la flexibilidad que he tenido para tomar decisiones, ya sean durante la implementación de una funcionalidad o el diseño de la página, los proyectos del mundo laboral son más amplios con una inmensa variedad de tecnologías diferentes que se pueden utilizar para desarrollar un producto software.

En conclusión, puedo decir que se aprende mucho más trabajando en un proyecto ya que llegas a conocer sobre distintos ámbitos relacionados a éste y su funcionamiento. Teniendo en cuenta todo lo anterior, puedo afirmar que gracias a esta experiencia tengo mucha más motivación hacia la profesión como desarrolladora software en el futuro.

7.3. Resultados del proyecto

El resultado final del plugin y el proyecto en general ha cumplido con sus expectativas. Aunque ha habido ciertos retrasos durante la implementación de la parte administradora, se ha compensado con la implementación más rápida de la parte de interfaces de usuario. Y, gracias a la constante ayuda del supervisor para resolver los problemas, no hubo ningún atasco grave durante el desarrollo.

El funcionamiento del plugin cumple los requisitos que se establecieron al comienzo del proyecto. De momento no se ha utilizado el plugin para ningún proyecto real, pero con el

ejemplo probado y que se ha explicado en el apartado anterior, se ha podido demostrar que funciona como toca.

Por otra parte, dependiendo de la empresa que lo utilice o el ámbito de negocios al que se vaya a enfocar, el plugin puede sufrir algunos pequeños cambios, siempre y cuando el administrador lo desee adaptarlo. O, también puede agregarle posibles mejoras o nuevas funcionalidades, como por ejemplo agregar productos a la lista de deseos.

BIBLIOGRAFÍA

[1] Martínez, G. *Qué es WordPress y sus características principales*. Webempresa. https://www.webempresa.com/WordPress/que-es-WordPress.html (Consulta: 1 de diciembre de 2021)

[2] Diaz, D. *Qué Es Xampp Usos, Características, Opiniones, Precios*. Mundobytes. https://mundobytes.com/xampp/ (Consulta: 28 de marzo de 2021)

[3] MySQL front. (n.d.). MySQL Front. https://www.roseindia.net/mysql/mysql-front.shtml (Consulta: 22 de mayo de 2022)

[4] C. *¿Qué es FileZilla y para qué sirve? - Neo Wiki*. NeoAttack. https://neoattack.com/neowiki/filezilla/ (Consulta: 22 de julio de 2021)

[5] Monster Digital Agency. *¿Qué es PHP y para qué sirve?* Epitech España. https://www.epitech-it.es/que-es-php/ (Consulta: 10 de febrero de 2022)

[6] *¿Qué es JavaScript? - Aprende sobre desarrollo web | MDN*. Mozilla. https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript (Consulta: 11 de febrero de 2021)

[7] C. *¿Qué es jQuery y para qué sirve? - Neo Wiki*. NeoAttack. https://neoattack.com/neowiki/jquery/ (Consulta: 6 de marzo de 2021)

[8] B., G. *¿Qué es AJAX y cómo funciona?* Tutoriales Hostinger. https://www.hostinger.es/tutoriales/que-es-ajax (Consulta: 2 de marzo de 2022)

[9] Marquez, C. R. *1.001 Funciones para WordPress*. Brandominus. https://brandominus.com/blog/creatividad/funciones-para-WordPress-hasta-por-las-orejas/#:%7E:text=Las%20funciones%20para%20WordPress%20sirven,comportamiento%20por%20defecto%20de%20WordPress (Consulta: 17 de diciembre de 2019)

[10] WordPress. *MySQL database*. https://codex.WordPress.org/File:WP_27_dbsERD.png (Consulta: 22 de mayo de 2022)

[11] *Registering Custom Post Types | Plugin Developer Handbook*. WordPress Developer Resources. https://developer.WordPress.org/plugins/post-types/registering-custom-post-types/ (Consulta: 22 de abril de 2022)

[12] *Working with Custom Taxonomies | Plugin Developer Handbook*. WordPress Developer Resources. https://developer.WordPress.org/plugins/taxonomies/working-with-custom-taxonomies/ (Consulta: 20 de marzo de 2022)

[13] *Custom Meta Boxes | Plugin Developer Handbook*. WordPress Developer Resources. https://developer.WordPress.org/plugins/metadata/custom-meta-boxes/ (Consulta: 20 de marzo de 2022)

ANEXO A

Diagrama de Gantt detallado

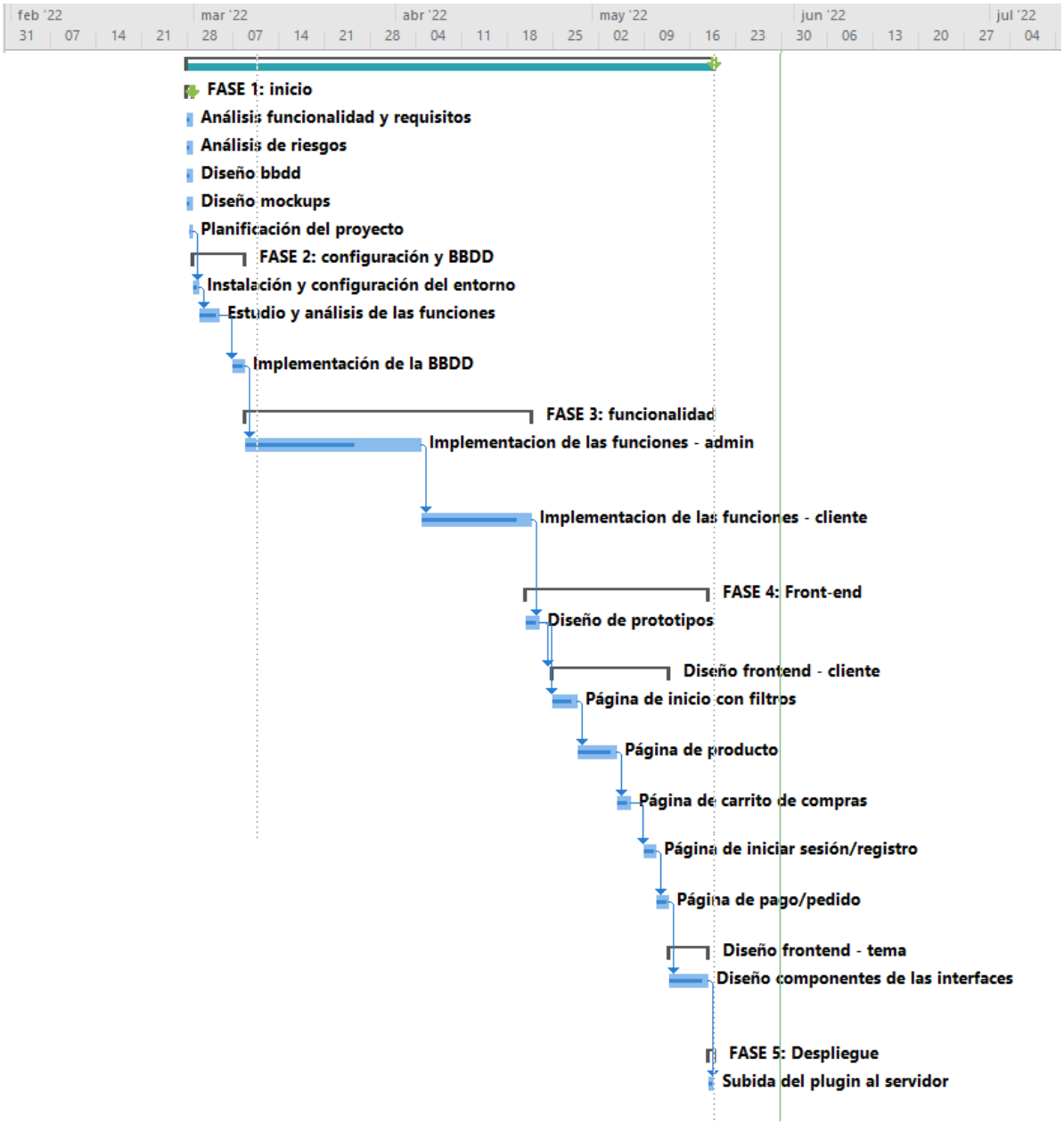

Figura 37: Diagrama de Gantt completo del proyecto

ANEXO B

Diagrama de la base de datos de WordPress

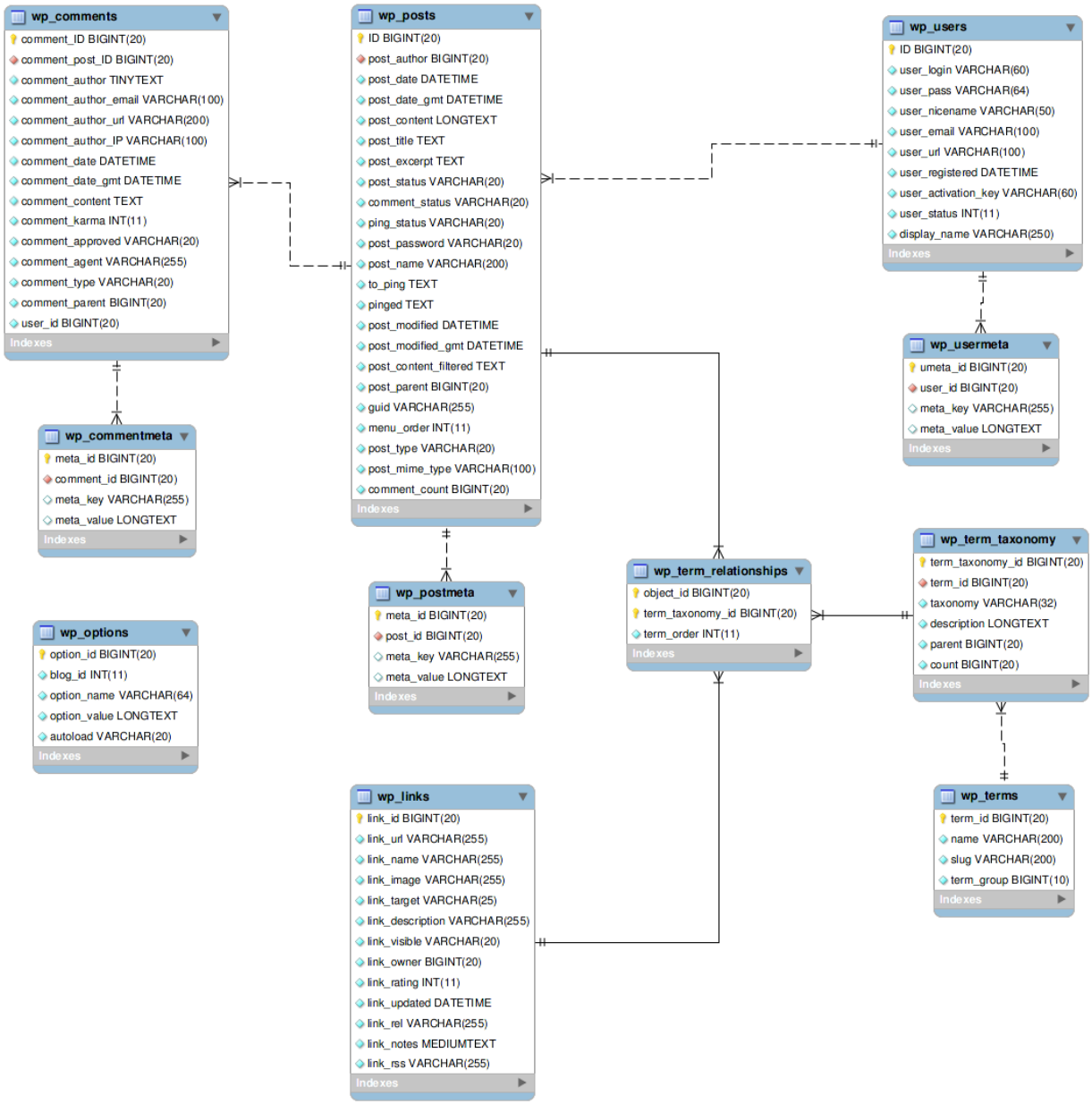

Tabla 25: Tablas de la base de datos completa de WordPress [10]

ANEXO C

Interfaces de usuario de la página de comercio electrónico

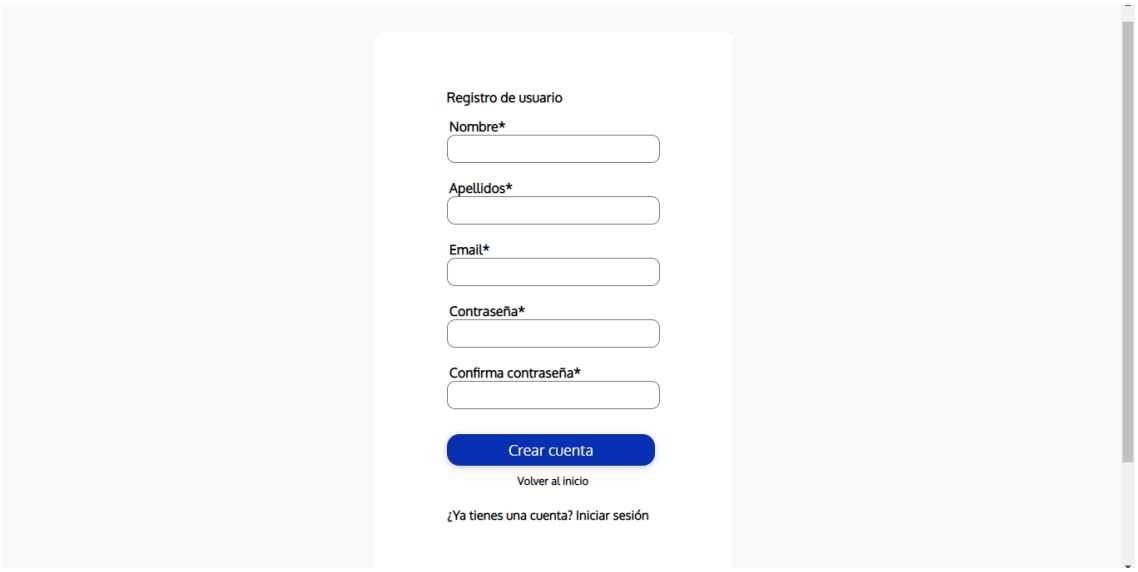

Figura 38: Interfaz de usuario para registrarse

Figura 39: Interfaz de usuario para iniciar sesión

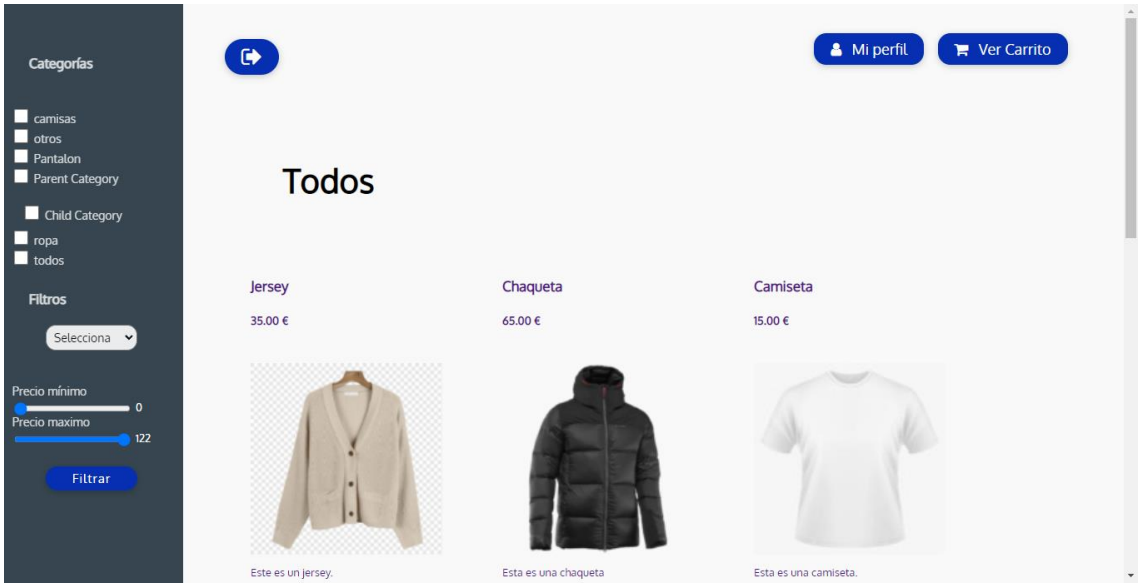

Figura 40: Interfaz de usuario del catálogo de productos

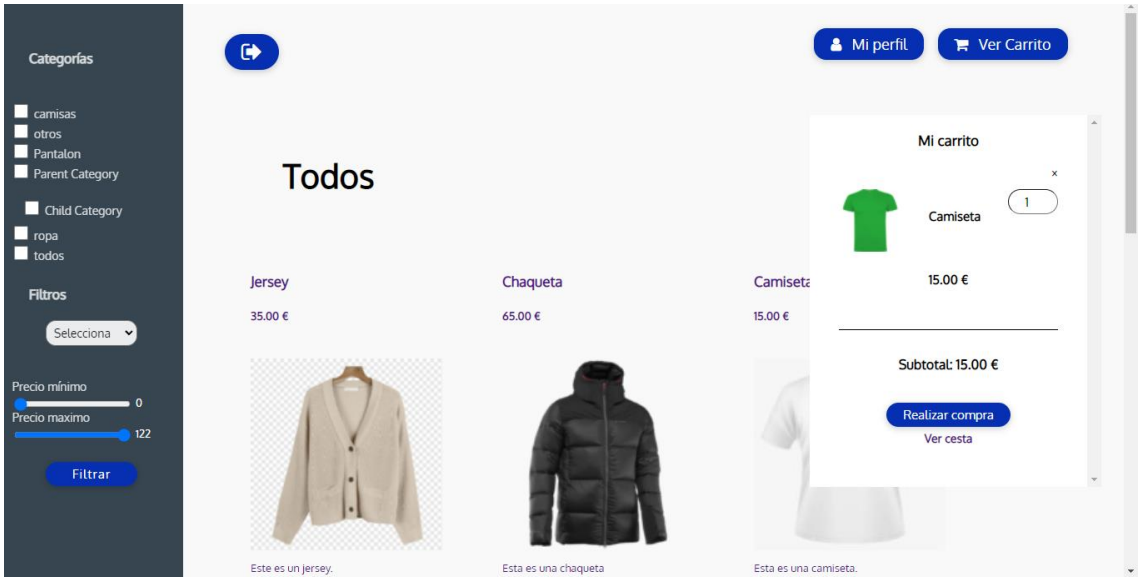

Figura 41: Interfaz de usuario del catálogo de productos con la cesta

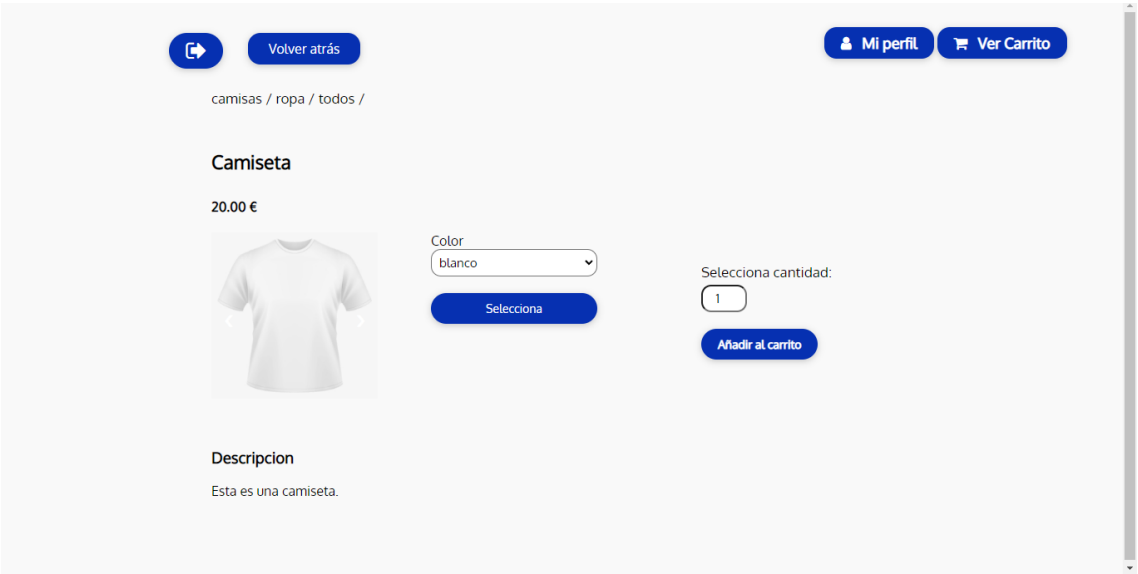

Figura 42: Interfaz de usuario del producto

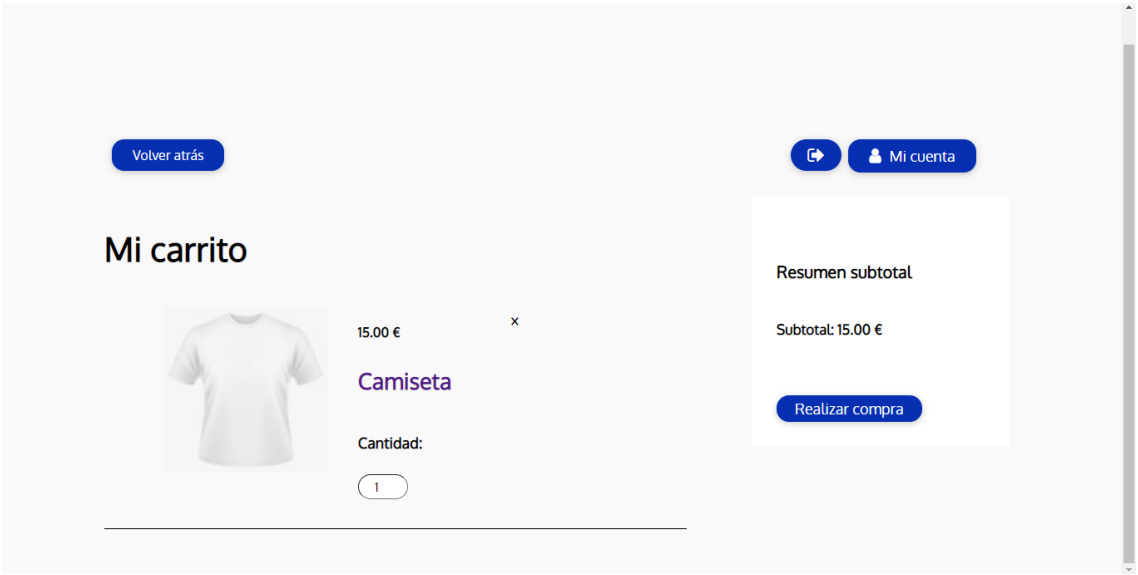

Figura 43: Interfaz de usuario del carrito de compras

Figura 44: Interfaz de usuario de los ajustes

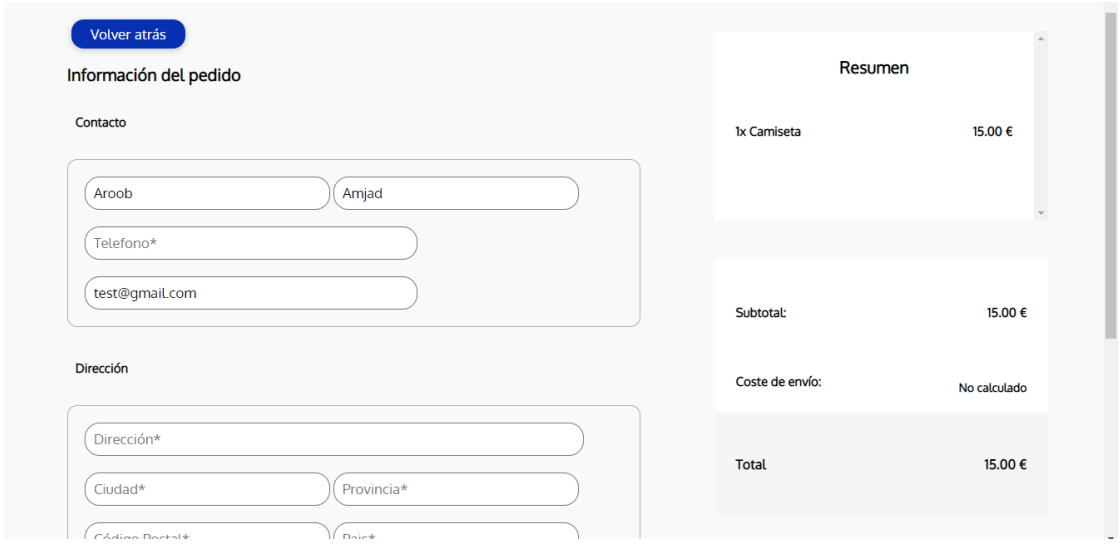

Figura 45: Interfaz de usuario del formulario de pedido (1)

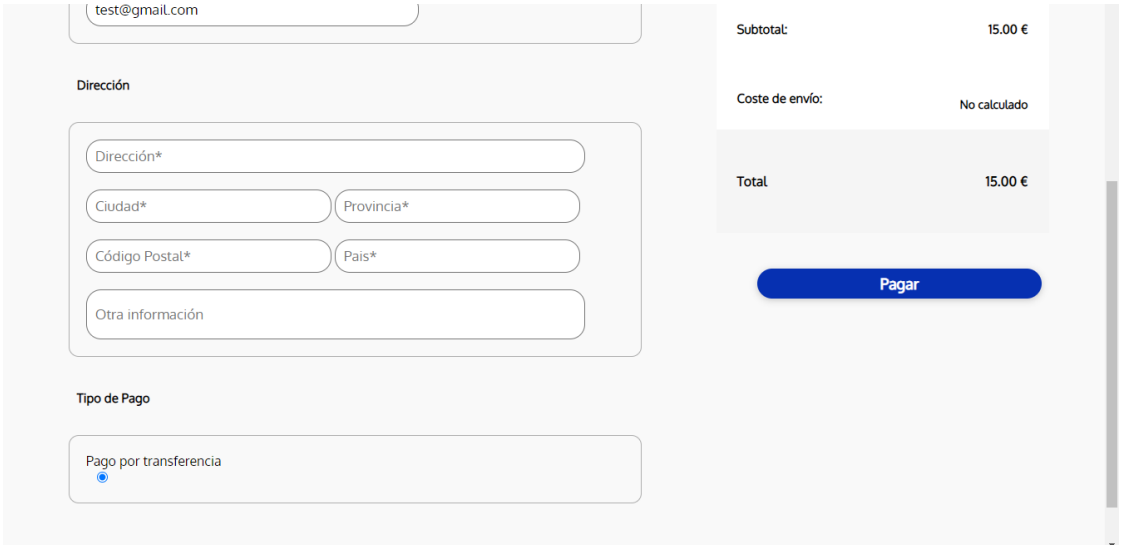

Figura 46: Interfaz de usuario del formulario de pedido (2)