PROYECTO INTEGRADO

PEDALNEST

Francisco Manuel Martínez Vázquez

ÍNDICE

INTRODUCCIÓN

Lo que haremos en este paso, será desarrollar el manual de administrador, veremos los objetivos del proyecto, la planificación, el código fuente relevante, etc.
Creo que es buena idea este proyecto ya que a parte de ser una tienda para comprar bicicletas y accesorios, tiene un apartado de reservas que lo hace distinto de la competencia. Al final la idea es vender productos relacionados con el mundo de las bicicletas y gestionar reservas.

Objetivos del proyecto

- Objetivo final:
  El objetivo principal del proyecto es desarrollar una tienda online de bicicletas, llamada *PedalNest*, que permita a los usuarios registrarse, iniciar sesión, consultar un catálogo de productos, realizar compras y reservas de bicicletas, y dejar reseñas sobre su experiencia, todo a través de una plataforma web funcional, intuitiva y moderna.
- Objetivos principales:
  - Permitir a los usuarios consultar un catálogo de bicicletas y de productos.
  - Facilitar la compra de productos mediante un sistema de carrito.
  - Implementar un sistema de reservas que permita seleccionar fecha, hora y número de participantes para alquilar bicicletas (Máximo 8).
  - Permitir a los usuarios registrarse, iniciar sesión y gestionar su perfil desde la aplicación.
  - Recoger reseñas de los usuarios sobre la tienda o los productos, y almacenarlas para su posterior consulta.
  - Garantizar el almacenamiento permanente y estructurado de toda la información (usuarios, productos, compras, reservas y reseñas) mediante una base de datos MySQL.
  - Ofrecer un diseño responsive y visualmente atractivo, adaptado a distintos dispositivos para mejorar la experiencia de usuario.

Planificación del proyecto

El proyecto ha sido desarrollado por mí, asumiendo todas las tareas y funciones de análisis, diseño, desarrollo, pruebas y despliegue.
A continuación, pondré una tabla con la planificación detallada de las tareas necesarias para el desarrollo completo de la tienda PedalNest. En dicha tabla, aparecen las tareas, el responsable, como están de desarrolladas, las horas estimadas y el coste estimado (Por ejemplo: 15*1.5*25 = 562.5€).

Tarea	Responsa...	Estado	Horas estimadas	Coste estimado
Planificación	F	Listo	15	562,5
Desarrollo FrontEnd	F	Listo	40	1500
Desarrollo BackEnd	F	Listo	120	4500
Desarrollo Base de Datos	F	Listo	2	75
Conexión con la BBDD	F	Listo	2	75
Desarrollo sistema de pago	F	Listo	10	375
Desarrollo sistema de reser...	F	Listo	10	375
Pruebas iniciales	F	Listo	40	1500
Despliegue	F	Listo	1	37,5
+ Agregar tarea				
			240 Total	9000 Total

En la planificación se ve que el tiempo estimado serán de unas 240 horas con un coste estimado de unos 9000€.
Cada tarea está asociada a un objetivo claro:

- Planificación: Organización inicial del proyecto.
- Front-End y Back-End: Desarrollo completo de la interfaz de usuario y la lógica de servidor.
- Base de datos y conexión: Diseño e integración con el Back-End.
- Sistemas de pago y reservas: Funcionalidades clave para la gestión del negocio.
- Pruebas: Verificación de que todo funciona correctamente.
- Despliegue: Subida del proyecto al servidor.

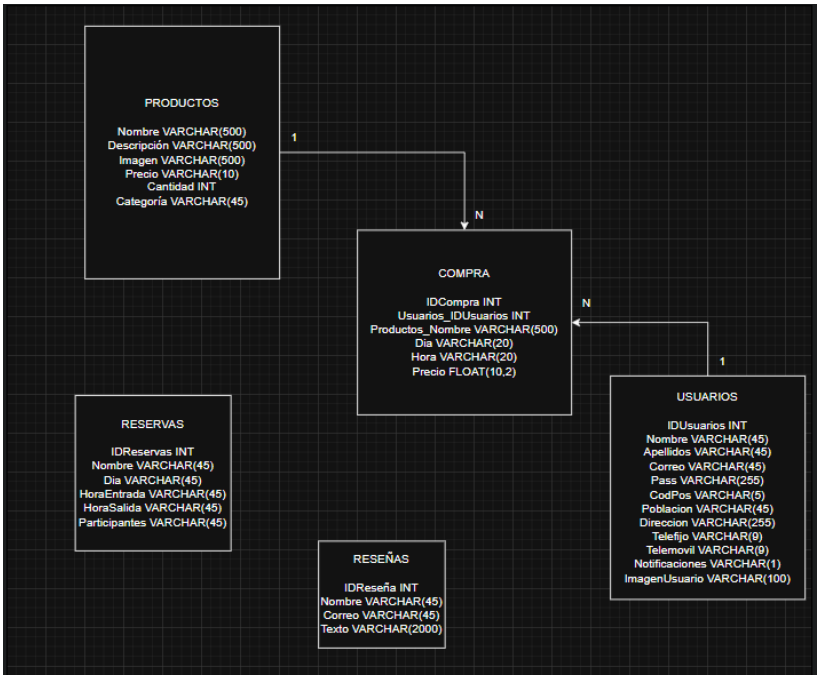
Análisis y diseño del sistema

Como dice el documento, el objetivo de este proceso es la obtención de una especificación detallada del sistema. Para ello se hará un diagrama entidad-relación (ER) y explicaré como crear a base de datos con unos datos ya integrados.

10.4.1. Modelo conceptual (diagrama entidad-relación)

En el mismo proyecto hay dos archivos, uno en XML y otro en JSON del diagrama de entidad relación. Se llaman “DiagramaER.xml” y “DiagramaER.json”.

Pondré una imagen del diagrama de mí base de datos para que también esté aquí reflejado.



10.4.2. Creación de la base de datos y consultas principales

Mí base de datos es relacional, por lo tanto a parte de tener el script de dicha base de datos, hay uno que se llama “Inserts.sql” que contiene una serie de datos para ponerlos en la base de datos y que se vea la web con un par de cosas ya en ella y se vea bien formada.

Para ello lo que se hará será:

1. Ejecutar el archivo “PedalNest.sql”.
2. Ya con la base de datos creada y en uso, ejecutamos el el archivo “Inserts.sql”.

Y ya tendríamos los datos en la base de datos y la web lista para verla correctamente.

Especificaciones del sistema

En este apartado, vamos a justificar el uso de las tecnologías y del software empleado, la instalación y configuración de la aplicación, y las especificaciones hardware del sistema.

10.5.1. Justificar el uso de la tecnología y el software empleado

Para el desarrollo de este proyecto, he utilizado las siguientes tecnologías y herramientas:

- Front-End:
  - HTML5/CSS3/JavaScript: Son tecnologías estándar para la estructura, estilos y funcionalidades del sitio web.
  - Bootstrap 5: Es un framework CSS utilizado para facilitar el diseño responsive y profesional. He utilizado también Bootstrap Icons para incorporar iconos modernos y ligeros.
  - JQuery: Es una biblioteca de JavaScript que se emplea para facilitar la manipulación del DOM, eventos y AJAX de forma más concisa y compatible.
- Back-End:
  - PHP: He elegido este lenguaje por lo simple que es, por el bajo consumo de recursos y por la amplia compatibilidad con servicios web como Apache. No he utilizado ningún framework adicional para así mantener el código accesible y personalizado desde cero.
  - PHPUnit: Es un framework de pruebas unitarias para PHP. Lo he utilizado para hacer las pruebas unitarias al código.
- Base de datos:
  - MySQL: Sistema de gestión de bases de datos relacional, gratuito, escalable y muy bien documentado.
- Entorno de desarrollo:
  - Visual Studio Code: Es un editor de código versátil con soporte para PHP, HTML, CSS, JavaScript y para más lenguajes y extensiones como lo son Prettier, Live Server, entre muchas otras.
  - XAMPP: Es un entorno local que incluye Apache, PHP y MySQL, ideal para hacer pruebas durante el desarrollo.

10.5.2. Instalación y configuración de la aplicación

Los requisitos previos para la instalación y la configuración de la aplicación son tener instalado XAMPP en el sistema y tener un navegador web moderno.

Pasos de instalación en entorno local:

1. Instalar XAMPP (https://www.apachefriends.org/es/download.html).
2. Copiar la carpeta del Proyecto en la ruta “C:\xampp\htdocs\xampp\”.
3. Iniciar los servicios de Apache y MySQL desde el panel de control de XAMPP.
4. Añadir la base de datos que hay en la carpeta “DATABASE” del proyecto (PedalNest.sql) a tu administrador de base de datos. También añadir el script “Inserts.sql” para tener algunos datos ya dentro de la base de datos.
5. Abrir el navegador y acceder a “http://localhost/xampp/PedalNest”.

Librerías utilizadas:

- Bootstrap 5: Para estilos responsive y componentes como botones, alertas, etc.
- Bootstrap Icons: Para los iconos del sistema como por ejemplo las flechas.
- JQuery: Se utiliza para simplificar tareas comunes como manipular el DOM, AJAX y eventos.

Despliegue en servidor web:

Para subir el proyecto a un servidor de producción, se necesita:

- Un servidor con Apache, soporte PHP 7.4+ y MySQL.
- Acceso FTP para subir archivos y gestionar la base de datos.
- Hay una opción alternativa como es el despliegue en servidores de pago, Hostinger o un VPS. En mí caso, lo subiré a una VPS (http://185.230.52.229/back/controladorInicio.php).

10.5.3. Elaboración de las especificaciones hardware del sistema

Requisitos del equipo de desarrollo:

- Procesador: Intel i5 o AMD Ryzen 5 (2.0 GHz mínimo).
- Memoria RAM: 8 GB.
- Disco duro: 500 MB libres para XAMPP y archivos del proyecto.
- Sistema operativo: Windows 10, Windows 11 o distribución Linux moderna.
- Conexión a Internet: para acceso a librerías externas y actualizaciones.

Requisitos del servidor web:

- Procesador: 2 núcleos.
- RAM: 2 GB mínimo.

- Disco duro: 1 GB libre.
- Sistema operativo: Linux o Windows Server.
- Software:
  - Apache 2.4.
  - PHP 7.4 o más.
  - MySQL 5.7 o superior.

Requisitos del cliente (usuario final):

- Navegador actualizado (Chrome, Firefox, Edge…).
- Acceso a Internet para visualizar correctamente los iconos desde CDN.

Especificaciones del software

En este apartado, descibiré las principales operaciones de la aplicación que he desarrollado y hablaré de las librerías de terceros utilizadas.

TECNOLOGÍAS UTILIZADAS:

Las librerías utilizadas han sido Bootstrap y las JQuery.

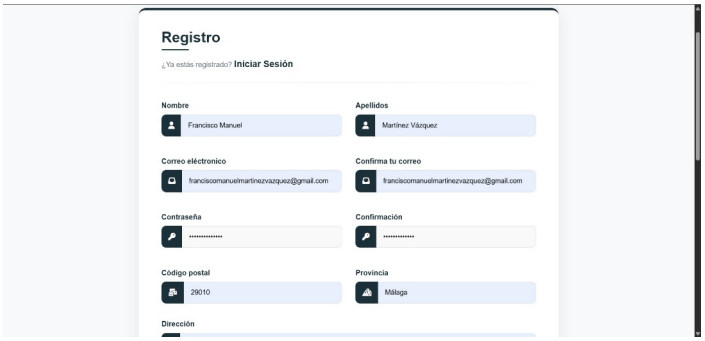
- Bootstrap es un framework de CSS (y algo de JavaScript) que sirve para crear interfaces web responsivas y visualmente atractivas de forma rápida y sencilla.
- JQuery es una librería de JavaScript que simplifica el trabajo con el DOM, eventos, animaciones y peticiones AJAX.
- PHPUnit es un framework de pruebas unitarias para PHP.

10.6.1. Descripción de las operaciones

Voy a describir las operaciones de mí web:

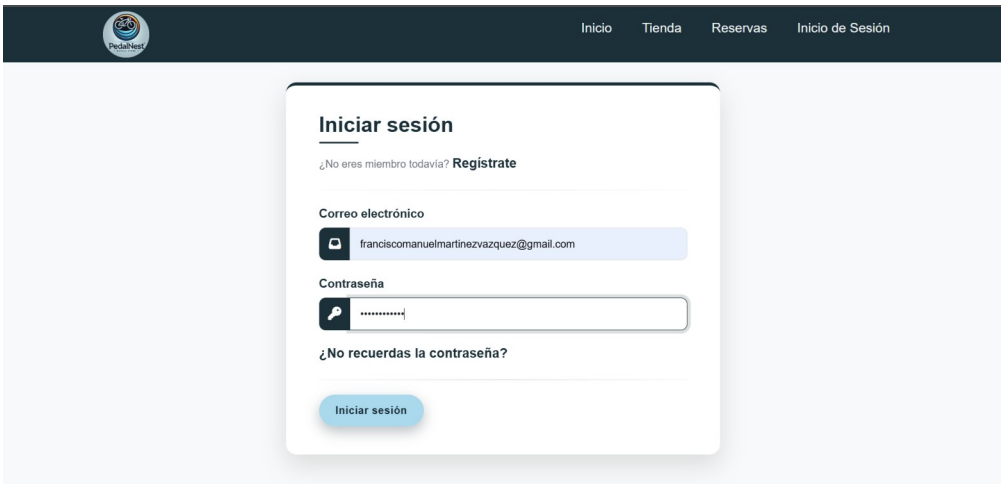
- **Operación: Registro de Usuario**
  - **Nombre**: Registro de Usuario
  - **Actor**: Usuario no registrado
  - **Resultado**: Creación de una cuenta y acceso automático al sistema
  - **Descripción**: El usuario introduce sus datos personales para registrarse.
  - **Precondiciones**:
    - El usuario no debe estar registrado previamente.
  - **Postcondiciones**:
    - El usuario es registrado y su sesión se inicia automáticamente.
  - **Parámetros**:
    - Nombre (texto, obligatorio)
    - Apellidos (texto, obligatorio)
    - Email (texto, obligatorio)
    - Contraseña (texto, obligatorio)
    - Código postal (texto, obligatorio)
    - Ciudad (texto, obligatorio)
    - Dirección (texto, obligatorio)
    - Teléfono (número, obligatorio)

- **Excepciones**:
  - Email ya registrado: Muestra mensaje de error.
  - Datos inválidos: Notifica formato incorrecto.
- **Flujo Principal**:
  - Acceder a la página de registro.
  - Introducir datos del formulario.
  - Enviar formulario.
  - Validar los datos.
  - Crear el usuario en base de datos.
  - Iniciar sesión automáticamente.



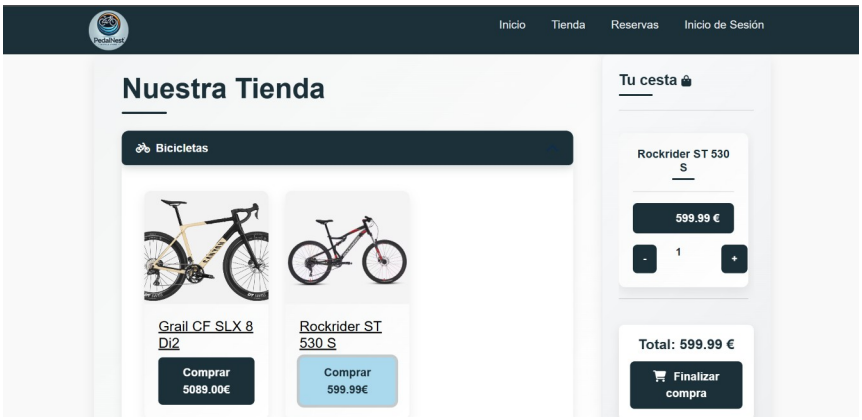
- **<u>Operación: Inicio de Sesión</u>**
  - **Nombre**: Inicio de Sesión
  - **Actor**: Usuario registrado
  - **Resultado**: Acceso autorizado al sistema
  - **Descripción**: El usuario inicia sesión con email y contraseña.
  - **Precondiciones**:
    - Usuario debe estar registrado.
  - **Postcondiciones**:
    - Usuario autenticado y sesión iniciada.
  - **Parámetros**:
    - Email (texto, obligatorio)
    - Contraseña (texto, obligatorio)

- **Excepciones**:
  - Credenciales incorrectas: Muestra error.
  - Usuario no registrado: Notifica inexistencia.
- **Flujo Principal**:
  - Acceder a la página de login.
  - Introducir email y contraseña.
  - Enviar formulario.
  - Validar credenciales.
  - Iniciar sesión y redirigir.



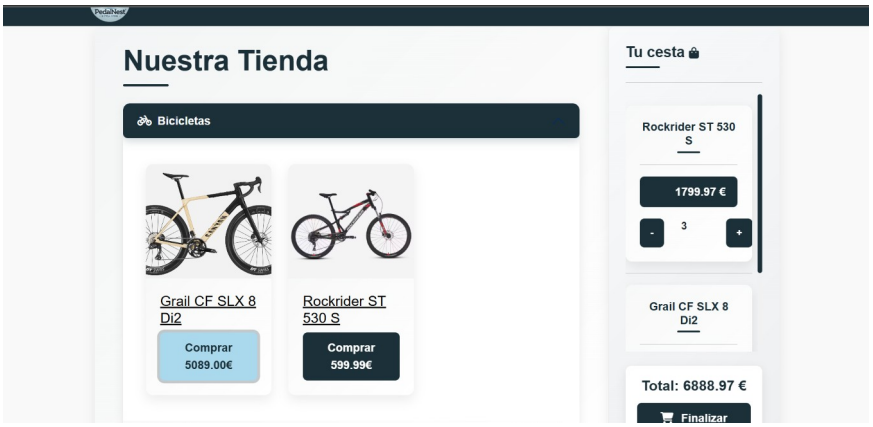
- **<u>Operación: Selección de Productos</u>**
  - **Nombre**: Añadir productos al carrito
  - **Actor**: Usuario
  - **Resultado**: Producto añadido al carrito
  - **Descripción**: El usuario navega por el catálogo y selecciona productos para comprarlos.
  - **Precondiciones**:
    - Producto disponible en stock.
  - **Postcondiciones**:
    - Producto añadido al carrito.
  - **Parámetros**:

        - ID del producto (entero)
        - Cantidad (entero)
    - **Excepciones**:
        - Stock insuficiente: Notifica al usuario.
        - Error interno: Muestra mensaje de error.
    - **Flujo Principal**:
        - Visualizar productos disponibles.
        - Seleccionar un producto.
        - Indicar cantidad.
        - Añadir al carrito.



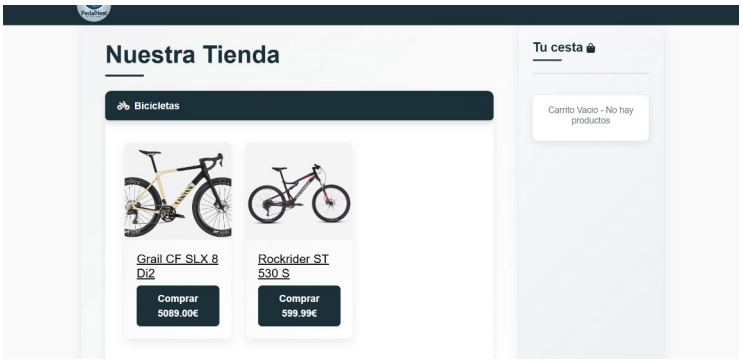
- **<u>Operación: Modificar Carrito</u>**
    - **Nombre**: Edición del carrito de compra
    - **Actor**: Usuario
    - **Resultado**: Carrito actualizado
    - **Descripción**: Permite cambiar cantidades o eliminar productos del carrito.
    - **Precondiciones**:
        - Carrito con productos.
    - **Postcondiciones**:
        - Carrito modificado.
    - **Parámetros**:
        - ID del producto (entero)

        - Nueva cantidad (entero)
    - **Excepciones**:
        - Cantidad inválida: Se muestra error.
        - Fallo en la actualización: Muestra mensaje de error.
    - **Flujo Principal**:
        - Acceder al carrito.
        - Cambiar cantidad o eliminar producto.
        - Guardar cambios.



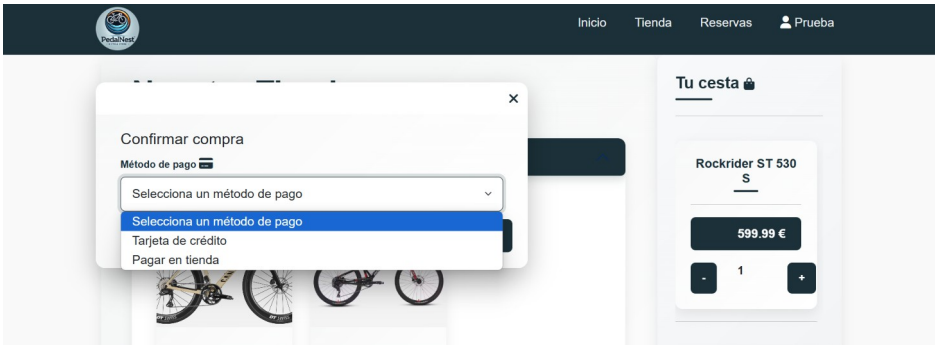
- **<u>Operación: Vaciar Carrito</u>**
    - **Nombre**: Vaciar carrito de compra
    - **Actor**: Usuario
    - **Resultado**: Carrito vacío
    - **Descripción**: Permite eliminar todos los productos del carrito.
    - **Precondiciones**:
        - Carrito con productos.
    - **Postcondiciones**:
        - Carrito completamente vacío.
    - **Parámetros**: Ninguno
    - **Excepciones**: Ninguna crítica
    - **Flujo Principal**:
        - Acceder al carrito.

- Pulsar opción de vaciar.
- Confirmar acción.



- **Operación: Confirmar Compra**
  - **Nombre**: Realizar compra
  - **Actor**: Usuario registrado
  - **Resultado**: Pedido almacenado en la base de datos
  - **Descripción**: Finaliza el proceso de compra registrando el pedido y vaciando el carrito.
  - **Precondiciones**:
    - Usuario autenticado
    - Carrito con productos
  - **Postcondiciones**:
    - Pedido registrado
    - Carrito vacío
  - **Parámetros**:
    - ID del usuario (entero)
    - Lista de productos (array)
    - Método de pago (texto)
  - **Excepciones**:
    - Error en la base de datos
    - Fallo en el pago
  - **Flujo Principal**:

- Revisar carrito
- Seleccionar método de pago
- Confirmar pedido
- Registrar en base de datos
- Mostrar confirmación



- **<u>Operación: Realizar Reserva de Bicicleta</u>**
  - **Nombre**: Reserva de bicicleta
  - **Actor**: Usuario
  - **Resultado**: Reserva almacenada con fecha y hora
  - **Descripción**: El usuario selecciona una bicicleta para reservar durante un período determinado.
  - **Precondiciones**: Ninguna
  - **Postcondiciones**:
    - Reserva registrada
  - **Parámetros**:
    - Nombre usuario (texto)
    - Fecha (fecha)
    - Hora inicio (hora)
    - Hora fin (hora)
  - **Excepciones**:
    - Fecha/hora inválidas
    - Conflicto con otra reserva

- **Flujo Principal**:
  - Introducir datos necesarios
  - Confirmar reserva
  - Registrar en base de datos
  - Mostrar confirmación

10.6.2. Interfaz y Navegación de la aplicación

Vamos a describir como pasar de unos sitios a otros en la web, para ello iré poniendo imágenes, textos, etc, para que sea lo más entendible posible.

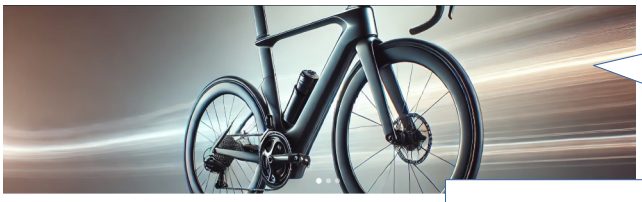
CABECERA Y FOOTER:

Es el navbar, podemos ir navegando entre los distintos apartados de la web, este está presente en todas las páginas

Es el footer, al igual que el navbar, está presente en todas las páginas de nuestra web. Contiene las redes sociales de la web, los derechos y un apartado con los contactos

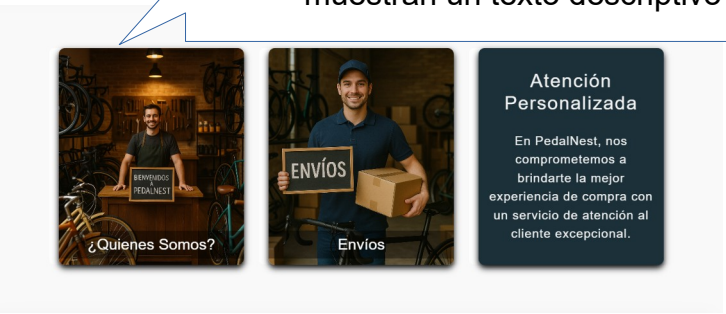
INICIO:

La página de inicio es algo sencillo e informativo, agradable a la vista y una interfaz moderna.



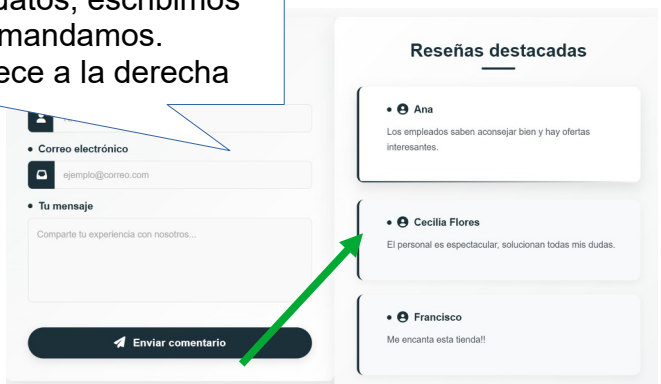
Lo primero que vemos es un Slider, que cada cierto tiempo cambia las imágenes

Seguidamente tenemos estas tres “Flip cards” descriptivas. Al pasar por encima se giran y muestran un texto descriptivo de cada apartado



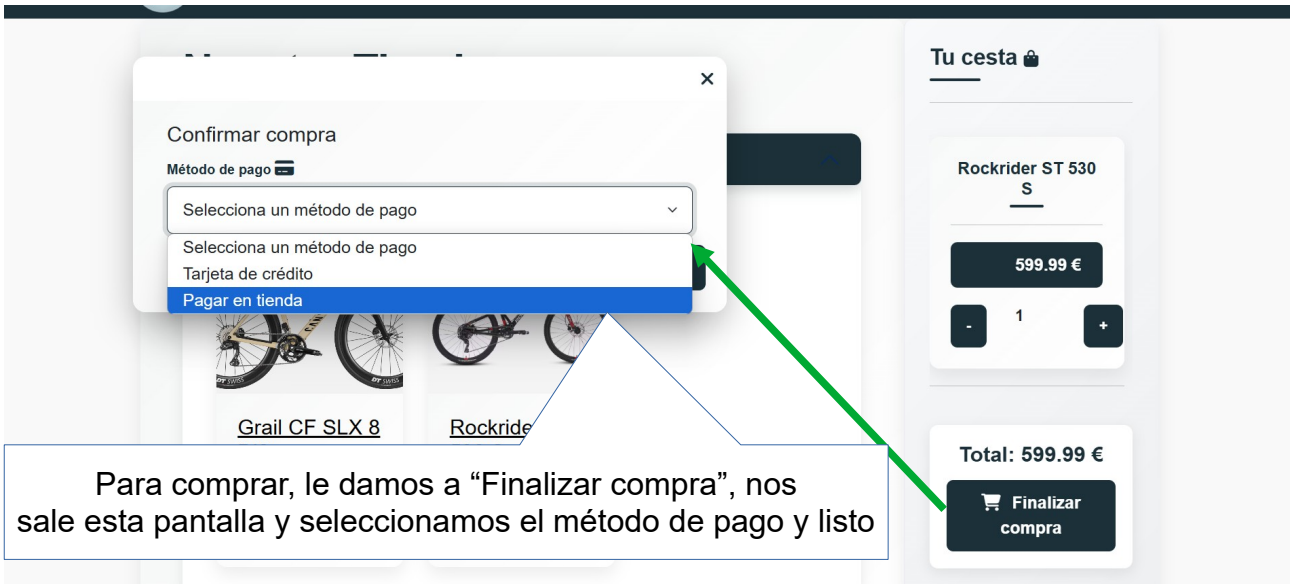
Esta sección es bastante sencilla, simplemente es un mapa que muestra donde se encuentra la tienda “PedalNest”

Y por último tenemos un formulario de reseñas. Metemos los datos, escribimos un comentarios y la mandamos. La reseña mandada aparece a la derecha

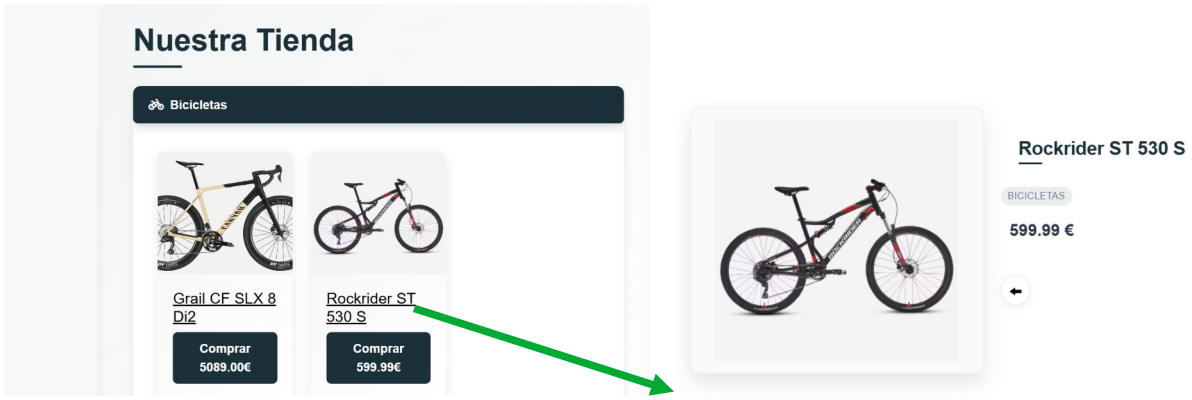


TIENDA:


Inicio
Tienda
Reservas
Inicio de Sesión
Nuestra Tienda
Bicicletas
Accesorios
Tu cesta
Carrito vacío
Carrito vacío - No hay productos
Grail CF SLX 8 Di2
Comprar 5089.00€
Rockrider ST 530 S
Comprar 599.99€
Pasamos al apartado de la tienda, aquí vemos dos desplegables que si le damos muestran los productos de cada sección
Si le damos al botón de los productos, se añaden a la cesta y podemos editar la cesta.
Rockrider ST 530 S
599.99 €
1
Total: 599.99 €
Finalizar compra



Para comprar, le damos a “Finalizar compra”, nos sale esta pantalla y seleccionamos el método de pago y listo

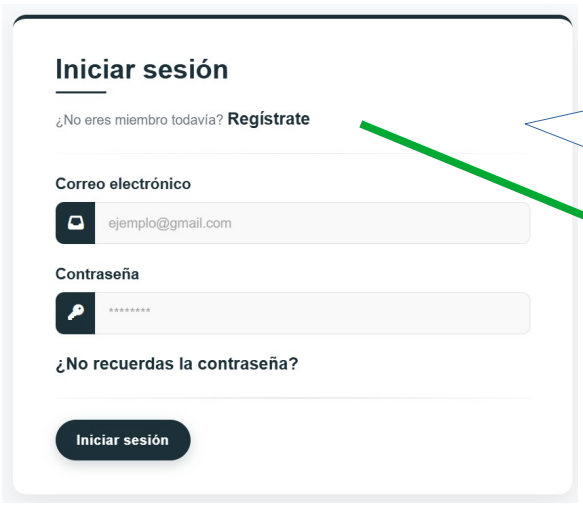


Por último y no menos importante, si clicamos en el producto, nos aparece una ventana que nos describe más detalladamente el producto

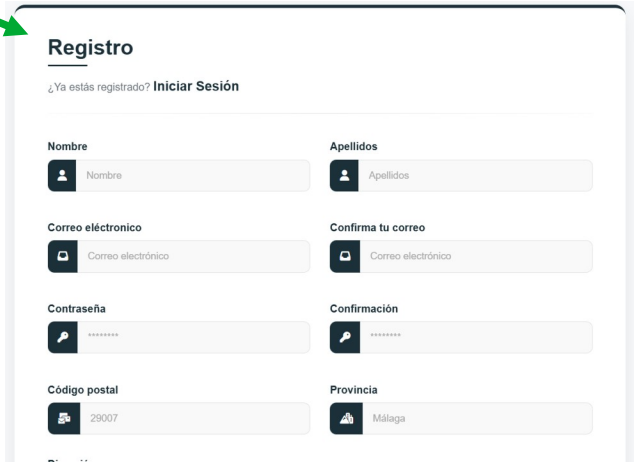
RESERVAS:

El apartado de reserva es el más sencillo, simplemente accedemos a él mediante el navbar, como con todos, y gestionamos la reserva metiendo los datos, le damos a “Enviar” y listo

INICIO DE SESIÓN:



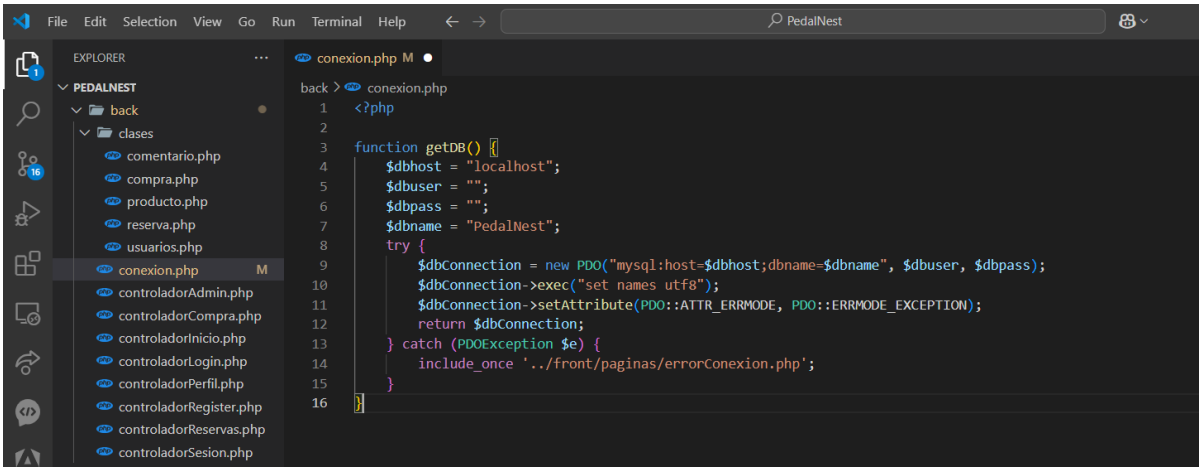
En el apartado de “Inicio de sesión”, al darle aparecemos en esta página, si no estamos registrados, le damos a “Regístrate” y nos podemos registrar.



Código fuente relevante

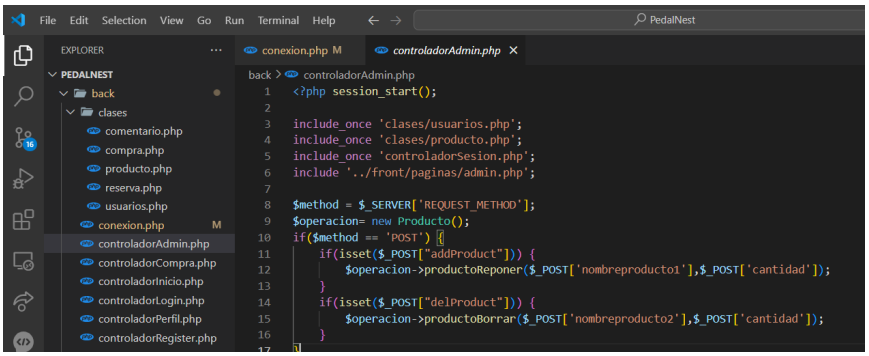
En este apartado, contaré un poco sobre partes o códigos importantes a tener en cuenta. Pondré imágenes y explicaré qué hace dicho código.

Para conectarnos a la base de datos he utilizado este método bastante sencillo:



Básicamente, esta función crea y devuelve una conexión a la base de datos MySQL utilizando PDO. Usa UTF-8 y activa las excepciones para errores de base de datos, por si hay un error que lo veamos mediante una página de error. Esta función se utilizará en los archivos que se conecten a la base de datos para hacer las operaciones correspondientes.

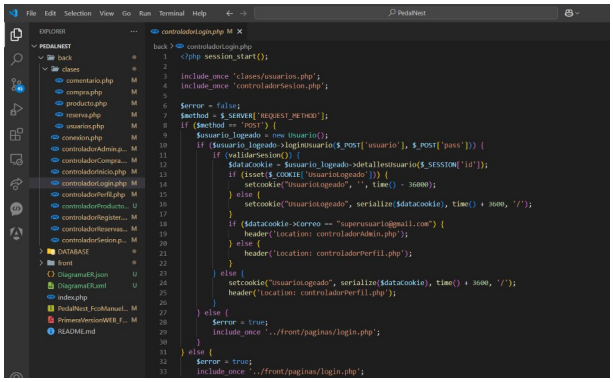
Algo muy importante es como el administrador de la página, puede añadir productos y quitarlos:



Este código, mediante un formulario, gestiona acciones de administración de los productos de la tienda.
Lo primero que hace es cargar las clases “Usuario” y “Productos” y el controlador de sesión, después muestra la página de administración, detecta si la petición es POST, y depende de lo que reciba el POST, repone o elimina los productos.

- Si se envió el formulario para añadir producto, llama al método para añadirlo.
- Si se envió el formulario para eliminar producto, llama al método para eliminarlo.

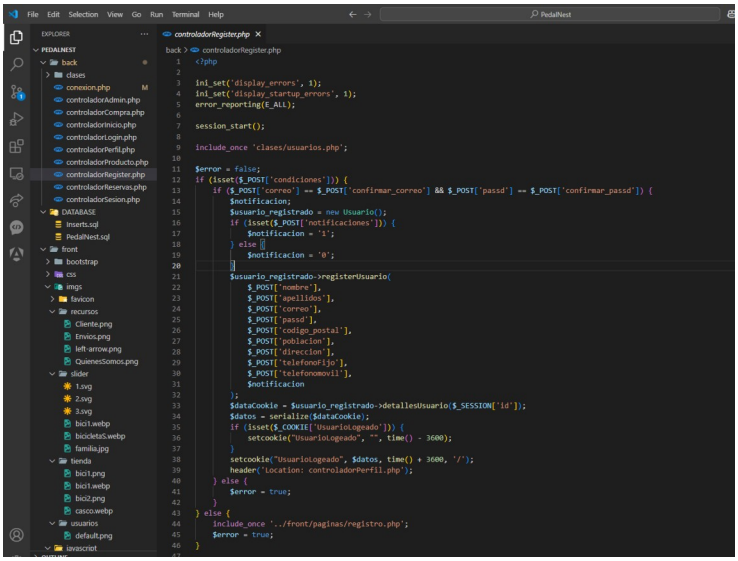
Ahora vamos a ver un código muy importante que es como sabe la aplicación si el usuario es un usuario normal o el administrador. Para ello veremos el *controladorLogin.php*.



```php
<?php session_start();

include_once 'clases/usuarios.php';
include_once 'controladorSesion.php';

$error = false;
$method = $_SERVER['REQUEST_METHOD'];
if ($method == 'POST') {
    $usuario_logeado = new Usuario();
    if ($usuario_logeado->loginUsuario($_POST['usuario'], $_POST['pass'])) {
        if (validarSesion()) {
            $dataCookie = $usuario_logeado->detallesUsuario($_SESSION['id']);
            if (isset($_COOKIE['UsuarioLogeado'])) {
                setcookie("UsuarioLogeado", '', time() - 36000);
            } else {
                setcookie("UsuarioLogeado", serialize($dataCookie), time() + 3600, '/');
            }
            if ($dataCookie->Correo == "superusuario@gmail.com") {
                header('Location: controladorAdmin.php');
            } else {
                header('Location: controladorPerfil.php');
            }
        } else {
            setcookie("UsuarioLogeado", serialize($dataCookie), time() + 3600, '/');
            header('Location: controladorPerfil.php');
        }
    } else {
        $error = true;
        include_once '../front/paginas/login.php';
    }
} else {
    $error = true;
    include_once '../front/paginas/login.php';
```

En este script si la petición es de tipo POST, crea una instancia de la clase Usuario y llama al método loginUsuario() con los datos enviados desde el formulario, si las credenciales son correctas, comprueba si la sesión es válida. Si la sesión es válida, obtiene los datos del usuario con detallesUsuario() y maneja una cookie llamada “UsuarioLogeado”, si ya existe, la elimina, y si no la crea con los datos serializados y una duración de una hora. Luego redirige al usuario a un controlador distinto según su correo: si es "superusuario@gmail.com", lo envía al panel de administrador, si no, lo redirige a su perfil. Si la sesión no es válida, también crea la cookie y redirige al perfil. En caso de que el login falle, o si la solicitud no fue enviada por POST, se muestra nuevamente la página de login con una variable de error activada.

Otro código que también considero importante es *controladorRegister.php*.



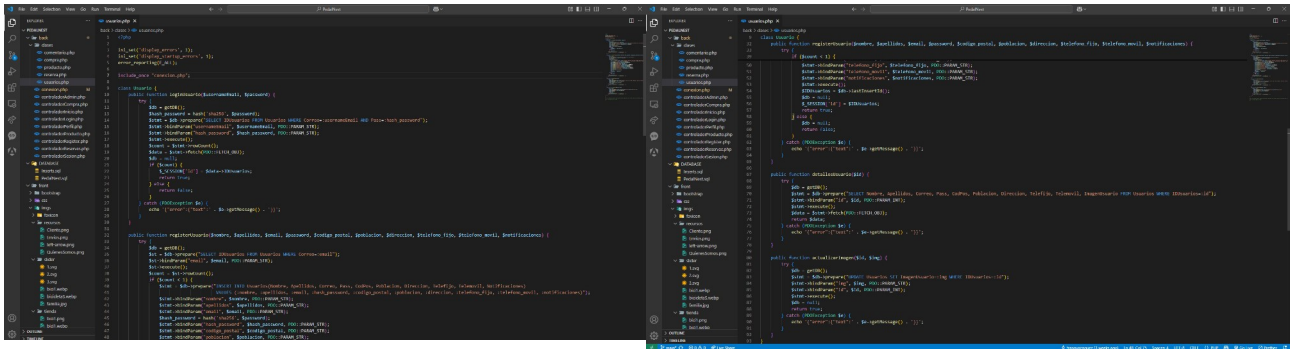
```php
<?php

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

session_start();

include_once 'clases/usuarios.php';

$error = false;
if (isset($_POST['condiciones'])) {
    if ($_POST['correo'] == $_POST['confirmar_correo'] && $_POST['passd'] == $_POST['confirmar_passd']) {
        $notificacion;
        $usuario_registrado = new Usuario();
        if (isset($_POST['notificaciones'])) {
            $notificacion = '1';
        } else {
            $notificacion = '0';
        }
        $usuario_registrado->registerUsuario(
            $_POST['nombre'],
            $_POST['apellidos'],
            $_POST['correo'],
            $_POST['passd'],
            $_POST['codigo_postal'],
            $_POST['poblacion'],
            $_POST['direccion'],
            $_POST['telefonoFijo'],
            $_POST['telefonomovil'],
            $notificacion
        );
        $dataCookie = $usuario_registrado->detallesUsuario($_SESSION['id']);
        $datos = serialize($dataCookie);
        if (isset($_COOKIE['UsuarioLogeado'])) {
            setcookie("UsuarioLogeado", "", time() - 3600);
        }
        setcookie("UsuarioLogeado", $datos, time() + 3600, '/');
        header('Location: controladorPerfil.php');
    } else {
        $error = true;
    }
} else {
    include_once '../front/paginas/registro.php';
    $error = true;
}
```

Este archivo PHP es responsable de gestionar el proceso de registro de un nuevo usuario en el sistema. Su complejidad reside en la validación de datos, el manejo de cookies, sesiones, la inserción en base de datos y la redirección condicional.

El código, hace lo siguiente:

1. Activa los errores.
2. Inicio de sesión y carga de clases.
3. Valida los datos.
4. Captura si el usuario acepta recibir notificaciones y lo guarda en la base de datos.
5. Procede a registrar al usuario.
6. Después del registro, se recuperan los detalles del usuario y se serializan para guardarlos como una cookie personalizada llamada **UsuarioLogeado**.
7. Y por último, redirige al usuario al controlador del perfil para que pueda ver sus datos personales justo después de registrarse.

Ahora pasaremos al código de *usuarios.php*:



Este archivo, se conecta con la base de datos y permite realizar las operaciones clave de: *registro*, *login*, *recuperación de datos* y *actualización de imagen de perfil*.

Hace lo siguiente:

1 Activa los errores y conecta con la base de datos.

2 Define métodos para manejar toda la lógica de usuarios.

2.1 Primer método: Autentica al usuario por correo y contraseña.

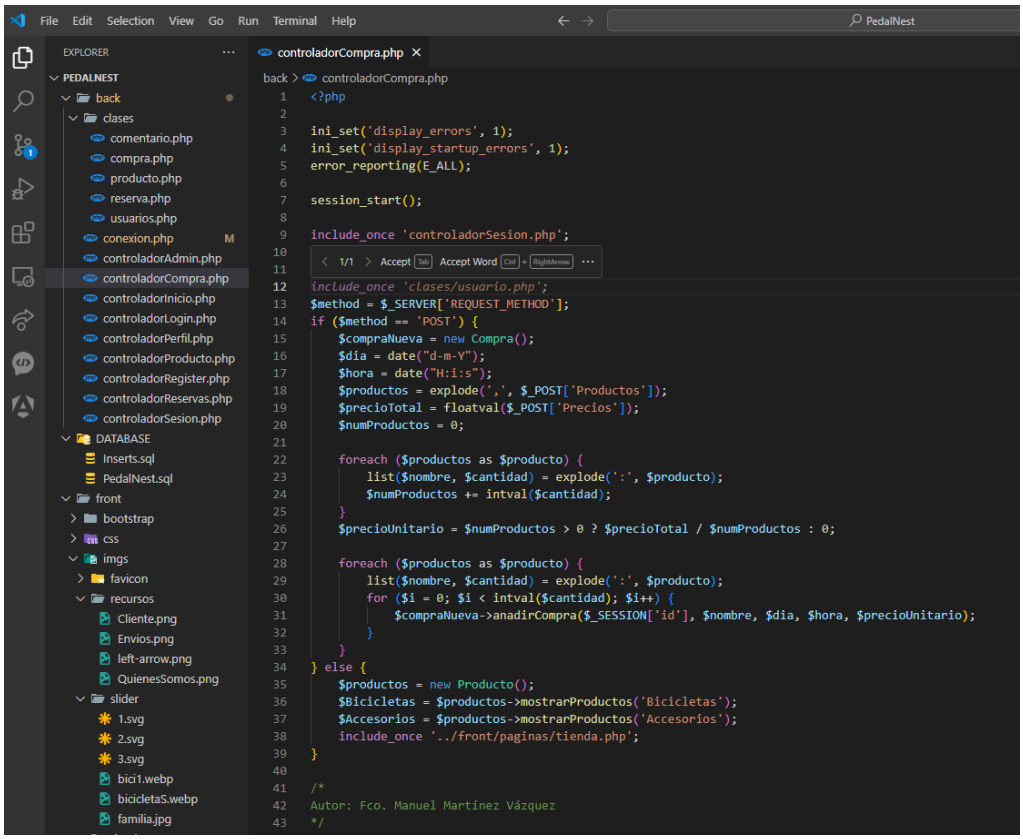
2.2 Segundo método: Registra un nuevo usuario.

2.3 Tercer método: Obtiene los datos personales de un usuario.

2.4 Cuarto método: Actualiza la foto de perfil del usuario.

Es un archivo clave en el desarrollo, ya que encapsula toda la lógica relacionada con el ciclo de vida de un usuario.

Seguidamente, voy a explicar el archivo *controladorCompra.php*:



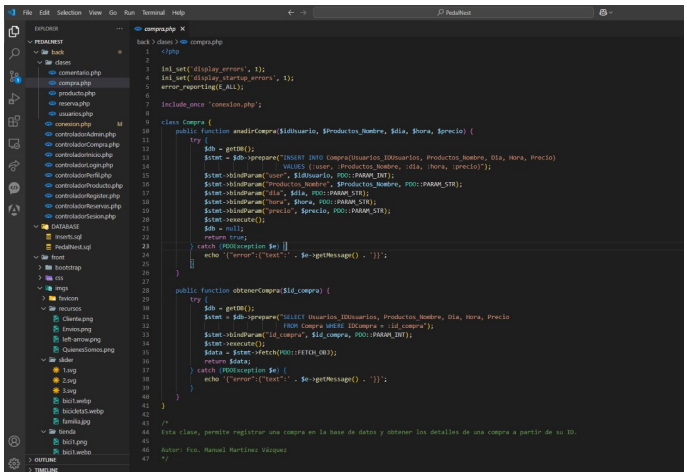
Este archivo se encarga de registrar las compras realizadas por los usuarios en la tienda PedalNest. Distingue entre peticiones POST (cuando se realiza una compra) y GET (cuando se muestra la tienda).

Se recoge la información de la compra enviada por el frontend (carrito).

- Se crea un objeto “Compra” para registrar la operación.
- Se obtiene la fecha ($dia) y la hora ($hora) de la compra.
- Se transforma el string “Productos” recibido (con formato Producto1:2,Producto2:1) en un array para separar cada producto y su cantidad.
- Se calcula el número total de productos y el precio unitario promedio.

Cada unidad de producto se inserta individualmente en la base de datos llamando a “anadirCompra”.
Si no es una compra (GET), se carga la tienda mostrando productos de tipo “Bicicletas” y “Accesorios”, utilizando la clase “Producto”.

A continuación vamos con *compra.php*:



El código, contiene la lógica para registrar y consultar compras en la base de datos.

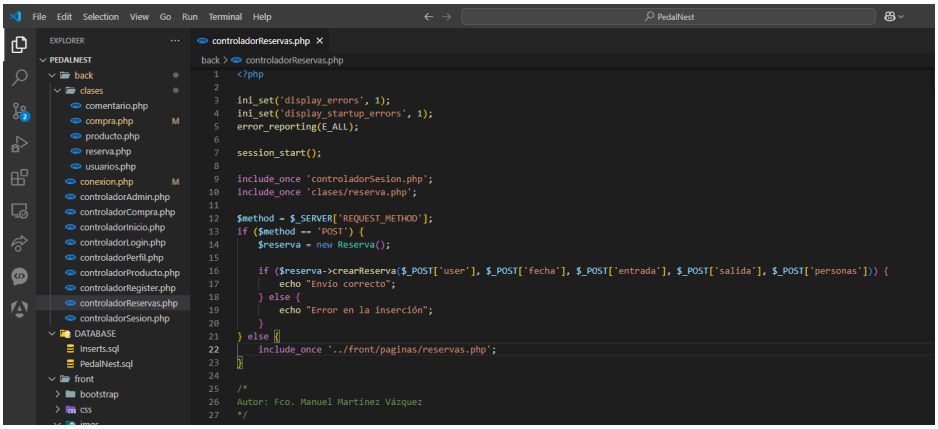
La función “anadirCompra”, registra una línea de compra con los siguientes campos:

- ID del usuario comprador.
- Nombre del producto.
- Día y hora de la compra.
- Precio.

Se hace un *INSERT* en la tabla “Compra”.

Y la función “obtenerCompra($id_compra)”, consulta una compra por su ID y devuelve los detalles: ID de usuario, producto, día, hora y precio.

Vamos con la gestión de reservas, para ello veremos primero el archivo *controladorReservas.php*:



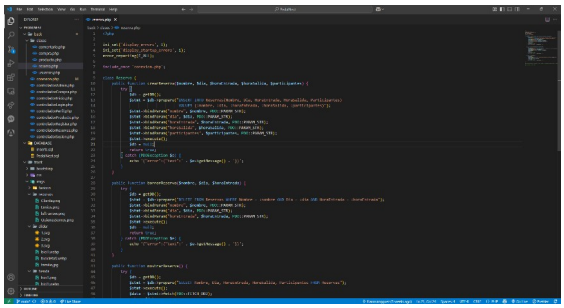
Este controlador se encarga de gestionar las reservas que hacen los usuarios en el sistema *PedalNest.*

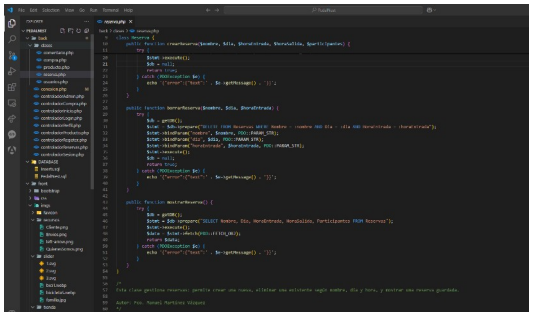
Si recibe una solicitud POST (crear una reserva):

1. Se crea un nuevo objeto Reserva.
2. Se recogen los datos enviados por el formulario:
   - 2.1 Usuario que realiza la reserva (user).
   - 2.2 Fecha (fecha).
   - 2.3 Hora de entrada (entrada) y salida (salida).
   - 2.4 Número de personas (personas).
3. Se llama al método “crearReserva()” de la clase “Reserva” para insertar la reserva en la base de datos.
4. Si la reserva se registra correctamente, se muestra "Envío correcto"; si no, "Error en la inserción".

Si es una solicitud GET (mostrar la vista), se incluye la página de reservas del frontend, donde los usuarios pueden ver el formulario para hacer la reserva.

Ahora pasamos a la clase *reserva.php*:





Contiene la lógica relacionada con el “registro”, “borrado” y “visualización” de reservas realizadas por los usuarios.

La función “crearReserva()”, inserta una nueva reserva en la tabla “Reservas” con los siguientes campos:

- Nombre: usuario que realiza la reserva.
- Día: fecha de la reserva.
- HoraEntrada: hora de inicio.
- HoraSalida: hora de finalización.
- Participantes: número de personas que participarán.

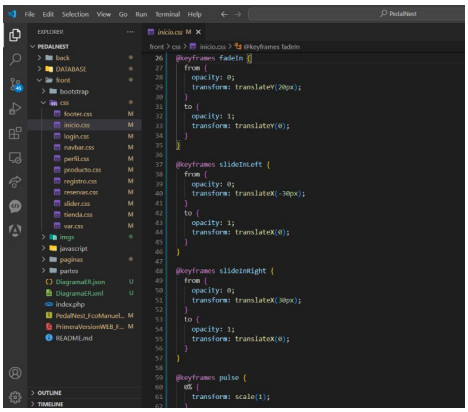
La función “borrarReserva ()”, permite eliminar una reserva concreta combinando tres condiciones:

- Nombre del usuario.
- Día.
- Hora de entrada.

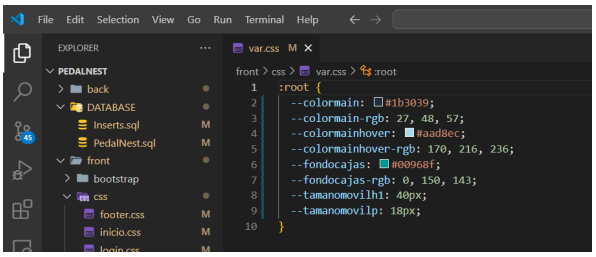
Esto garantiza que se elimina una reserva específica.

Y la función mostrarReserva(), consulta una única reserva de la base de datos.

Ahora vamos a pasar a un trocito de código que tengo en un par de CSS.



Es bastante sencillo, básicamente es lo que gestiona las animaciones CSS que se definen mediante @keyframes, en concreto son 4 animaciones. La animación *fadeIn* hace que un elemento aparezca suavemente desde abajo aumentando su opacidad y subiendo 20 píxeles. La animación *slideInLeft* hace que el elemento entre desde la izquierda desplazándose horizontalmente 30 píxeles y volviéndose visible, *slideInRight* hace lo mismo pero desde la derecha. Por último, *pulse* crea un efecto de latido o crecimiento suave, ampliando el tamaño del elemento ligeramente al 50% de la animación y luego volviendo a su tamaño original. Estas animaciones se usan normalmente para dar dinamismo y transiciones suaves a los elementos de una página web.
Como digo las hay en diferentes CSS para que de algo de “vida” a la web, cada uno hace algo distinto pero en general son lo mismo.



Este CSS, contiene unos estilos generales para utilizarlos en todos los archivos CSS y que sea más fácil de desarrollar la web.

Conclusiones del proyecto

El desarrollo del proyecto *PedalNest* ha sido una gran experienza a nivel personal, ha sido muy enriquecedor haber desarrollado esta aplicación tanto a nivel personal como técnico. Durante el proceso se han abordado múltiples áreas del desarrollo web: diseño frontend, programación backend en PHP, gestión de base de datos con MySQL, seguridad, experiencia de usuario, etc.

Uno de los principales logros del proyecto ha sido la integración completa de funcionalidades claves como son el registro e inicio de sesión de usuarios, la gestión dinámica del carrito de compra, la forma de hacer las compras y la visualización de productos mediante una interfaz agradable para el usuario.

Soy consciente de que aún estoy en un proceso de aprendizaje y me queda mucho recorrido, hay cosas que pulir y mejorar que con el paso de los años iré perfeccionando pero me siento orgulloso haber desarrollado algo de tal magnitud.

Futuras versiones y mejoras

Para un futuro, se contemplan varias mejoras (si esto llegase a algo más que a ser el TFG, lo implementaría):

- Implementación de un sistema de recomendaciones inteligentes mediante IA.
- Añadir pasarela de pago real (Stripe, PayPal, Bizum, etc.).
- Mejorar la personalización de bicicletas desde el frontend.
- Panel de administración para gestionar productos, pedidos y usuarios.
- Una parte visual mucho más moderna y profesional.

Motivos de operaciones no implementadas

Algunas operaciones como el sistema de personalización completa de bicicletas o la IA recomendadora, no se han implementado por falta de tiempo y/o por tratarse de funcionalidades avanzadas que requerirían una mayor inversión en el desarrollo y las pruebas.

Comparación de la temporalización inicial y final

El tiempo de desarrollo se ha cumplido así que no habrá que cambiar nada y las pruebas se han terminado en la fecha estimada.

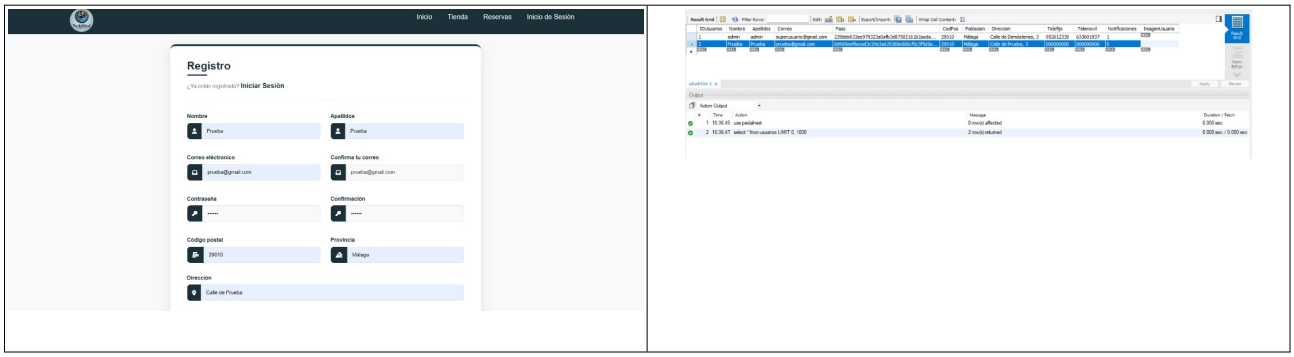
Apéndices

Este apartado lo que haré será demostrar mediante pruebas que funciona el programa y contrastar que lo he hecho yo.

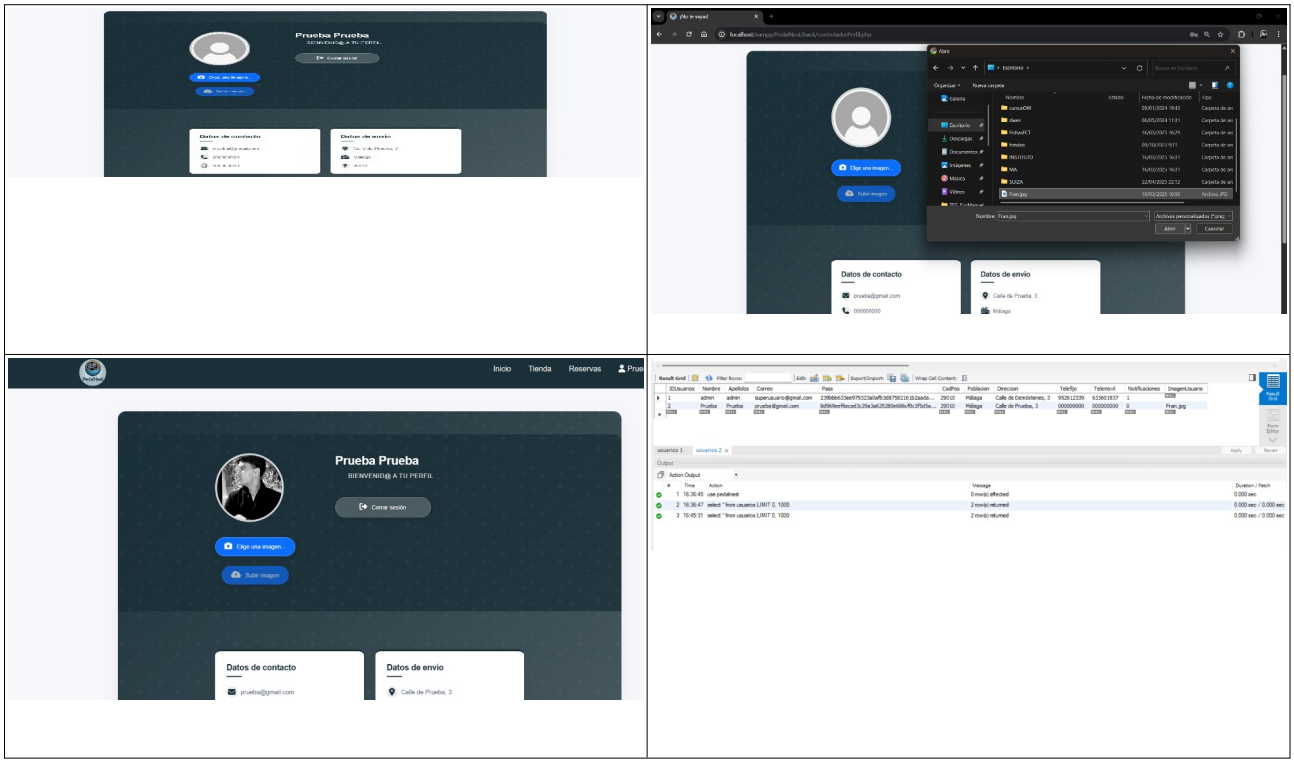
Primero, voy a utilizar datos de prueba para mostrar que se pueden crear usuarios desde la web. Crearé el usuario “prueba” y realizaré una compra.

Creación:

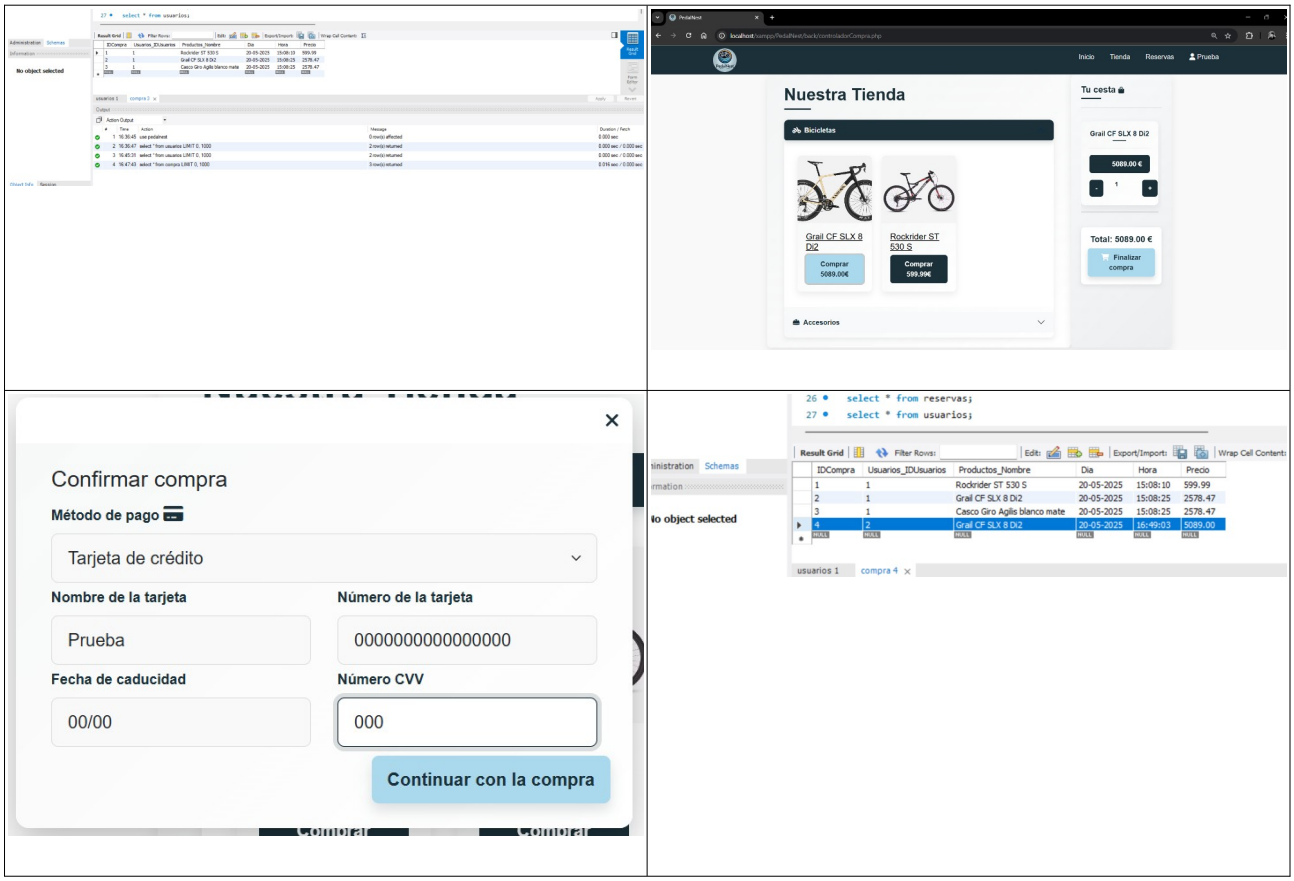
1. Registramos el usuario.



2. Editamos la foto de perfil.



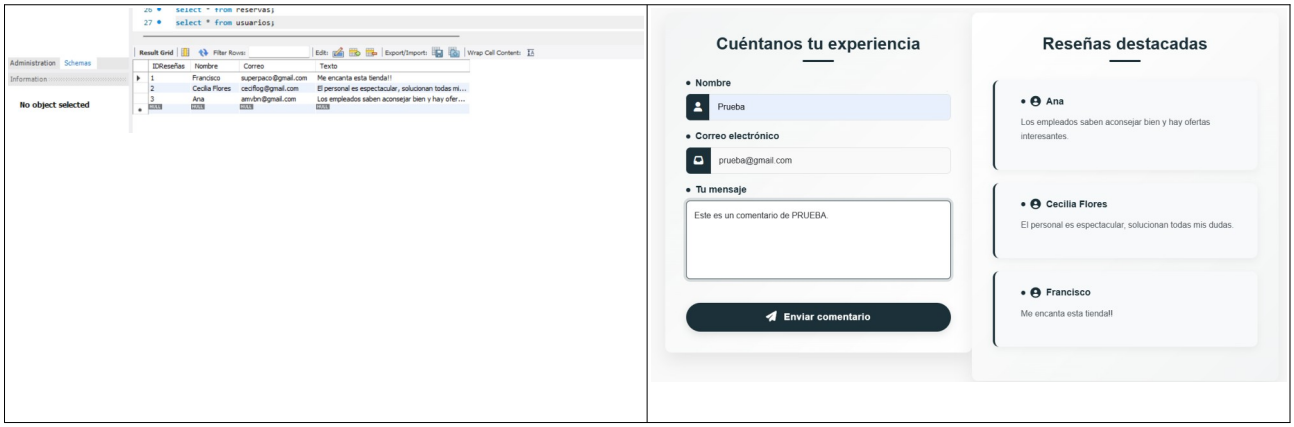
Compra:

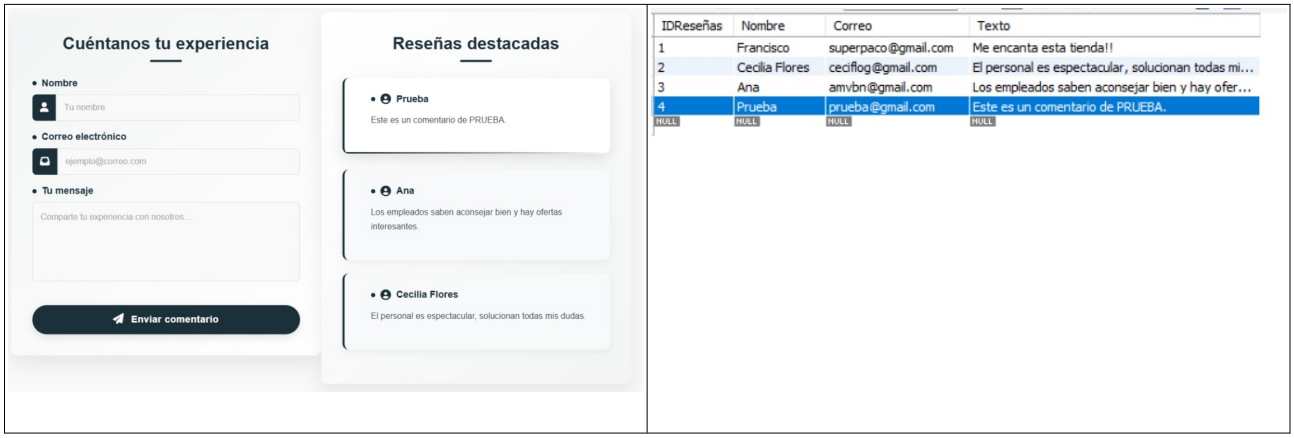


He hecho un proceso sencillo, he creado un usuario “Prueba”, le he cambiado la foto y he realizado una compra de ejemplo, comprobando que todo funciona contrastanto con la BBDD.

Ahora demostraré con el mismo usuario, como poner una reseña en la web.

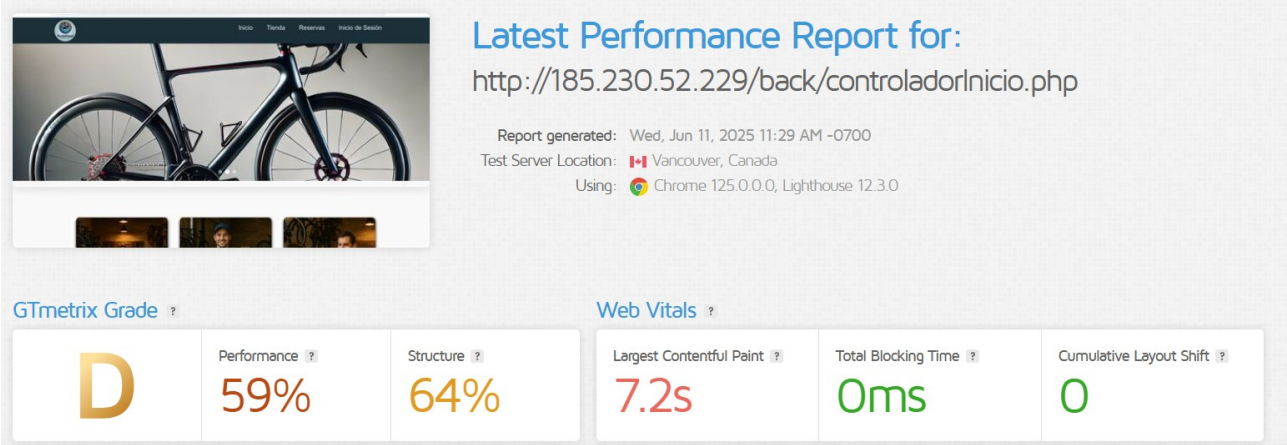
Reseña:





Ahora haré una prueba con una herramienta que analiza la velocidad de las páginas web (https://gtmetrix.com/), voy a pasar la página de inicio ya que es a la que accede el usuario.

Mí web la tengo subida en un VPS de prueba (http://185.230.52.229/back/controladorInicio.php).



Bibliografía

A parte de las documentaciones oficiales de cada lenguaje que he utilizado para el desarrollo, que son estas:

- https://www.php.net/ → Documentación oficial de PHP.
- https://developer.mozilla.org/es/ → MDN Web Docs: HTML, CSS y JavaScript.
- https://getbootstrap.com/ → Bootstrap.
- https://dev.mysql.com/doc/ → Documentación oficial de MySQL.
- https://fontawesome.com/ → FontAwesome (Para los iconos de la web).
- https://fonts.google.com/ → Google Fonts.

También he utilizado códigos de terceros, como son:

- https://getbootstrap.com/docs/5.0/components/accordion/ → Acordeón de la tienda.
- https://abelosh.com/slider-responsive-con-html-css-y-javascript/ → Slider de la página de inicio.
- https://developer.mozilla.org/en-US/docs/Web/API/Element/blur_event → El evento Blur, que cuando abandonas la web, pone “!No te vayas¡”.
- https://www.patreon.com/posts/tarjetas-65358058?utm_medium=clipboard_copy&utm_source=copy_to_clipboard&utm_campaign=postshare → Flip Cards de la pantalla de inicio.
- https://developer.mozilla.org/en-US/docs/Web/CSS/@keyframes → Animaciones CSS de los elementos de las páginas de la web.