IES INFANTA ELENA

CICLO FORMATIVO DE GRADO SUPERIOR

DESARROLLO DE APLICACIONES WEB

PROYECTO FINAL PARA EL MÓDULO DESARROLLO WEB EN ENTORNO SERVIDOR

Web Personal Desarrollador

Autor: Franco Benavides García

Tutor/a: Sarah González Ilva

Curso: 2025-2026

TÍTULO: Personal Developer Portfolio

AUTOR: Franco Manuel Benavides García

TUTOR DEL PROYECTO: Sarah Ilva González Camacho

FECHA DE LECTURA:

CALIFICACIÓN:

Tabla de contenidos

Tabla de ilustraciones

1. INTRODUCCIÓN

Este documento es una memoria completa sobre el proceso de desarrollo del proyecto final para el módulo de Desarrollo Web en Entorno Servidor (DWES) del Ciclo Formativo de Grado Superior de Desarrollo de Aplicaciones Web.

El proyecto consiste en crear una web donde mostrar mi currículum, los distintos proyectos que tengo en marcha y una plataforma de venta de cursos (ficticia), con su respectivo carrito de compra, autenticación de usuarios y gestión de los cursos.

1.1. MOTIVACIÓN

He decidido realizar este proyecto de esta forma porque se adapta perfectamente a los contenidos aprendidos durante el ciclo formativo y a los requisitos técnicos que se pedían, permitiéndome poner en práctica todos los conocimientos adquiridos hasta el momento en desarrollo web, bases de datos, programación frontend y backend, gestión de usuarios y diseño responsivo.

Además, es una aplicación que me puede servir el día de mañana si le añado algunas funcionalidades extra como una pasarela de pago real o un sistema de encriptado de contraseñas o para mostrar nuevos proyectos complejos que realice.

1.2. OBJETIVOS PRINCIPALES (GENERALES Y ESPECÍFICOS)

El objetivo principal es crear una aplicación web completamente funcional aunque con ciertas limitaciones, para la gestión online de cursos. En ella podremos:

- Visualizar los cursos disponibles.
- Añadir cursos a un carrito de compra.
- Gestionar su sesión mediante un sistema de autenticación.

La aplicación contará con un sistema de roles, donde un usuario con rol de Administrador podrá gestionar los cursos disponibles. Este administrador podrá:

- Dar de alta nuevos cursos.
- Modificar cursos existentes.
- Eliminar cursos.

Para poder acceder a estas funciones, el sistema exigirá que el usuario esté autenticado con un rol de tipo Administrador, garantizando así la seguridad y el correcto uso de la plataforma.

2. METODOLOGÍA UTILIZADA

El desarrollo de este proyecto fue dividido en semanas donde en cada una de ellas he organizado el trabajo por secciones para gestionar mejor los cambios y alcanzar con éxito una entrega correcta y a tiempo.

Cada etapa consiste en una fase distinta del proyecto y supone una nueva versión del mismo cada vez que se avanza a la siguiente etapa.

En cada fase he ido probando las funcionalidades de forma práctica para asegurarme de que todo funcionara correctamente antes de pasar a la siguiente etapa.

Durante el desarrollo, he ido integrando los distintos elementos de la web, como el login, el carrito y el CRUD de cursos, para ir viendo cómo interactúan entre sí. Esto me ha ayudado a comprender mejor el flujo de la aplicación y a detectar errores a tiempo, para solucionarlos en ese momento sin dejar que la aplicación escale con errores. Cada semana se ha convertido en un pequeño ciclo de planificación, desarrollo, prueba y corrección. Además, he dedicado tiempo a aprender y consultar documentación externa cuando encontraba dudas sobre PHP, Bootstrap o MySQL.

Cada iteración en la organización se compondrá de las siguientes tareas que se pueden apreciar a continuación:

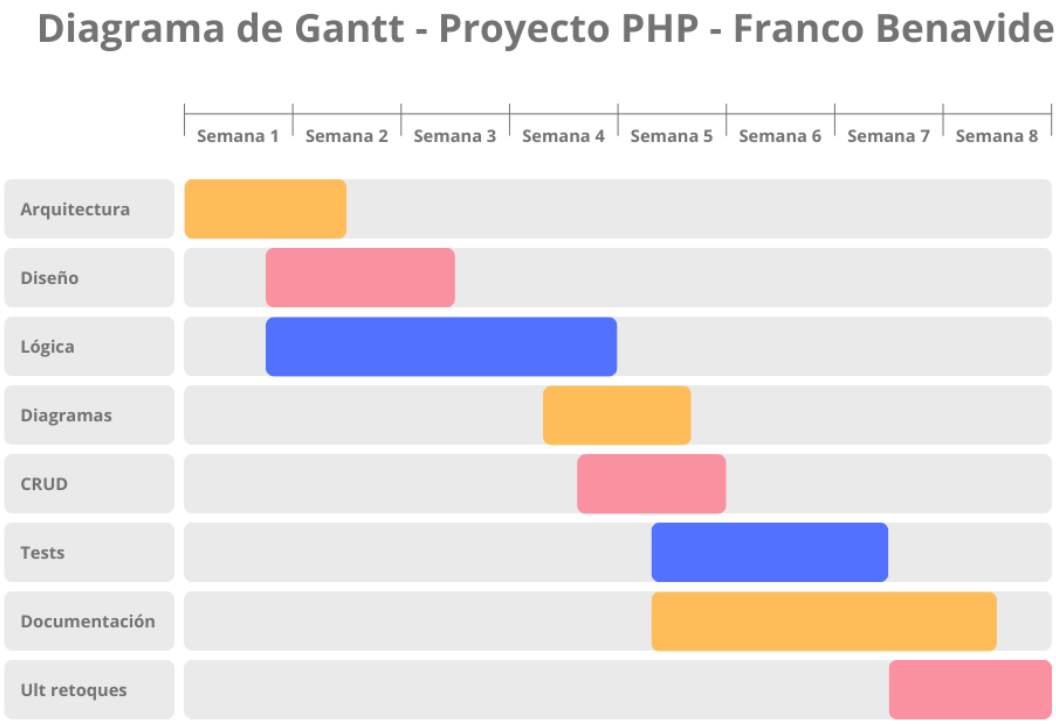

Ilustración 1: Diagrama de Gantt

A continuación, muestro detalladamente qué he ido desarrollando en cada fase y los distintos caminos en los que me enfoqué en cada momento:

- Análisis inicial y arquitectura

En esta primera semana se analizaron todos los requisitos que se exigían para realizar la actividad y desarrollé el plan para llevarlos a cabo en mi aplicación web, definiendo las funcionalidades mínimas que debe ofrecer la plataforma y creando una forma inicial, un primer boceto para ese contenido dividiéndolo principalmente en registro de usuarios, venta de cursos y gestión del contenido como elementos clave de la misma.

Establecí la arquitectura general que seguiría a lo largo del desarrollo y concreté sus distintas capas para una mejor organización… aludiendo al **Modelo-Vista-Controlador**, para futuras referencias **MVC.**

En la siguiente ilustración, se muestra la arquitectura elegida para este proyecto:

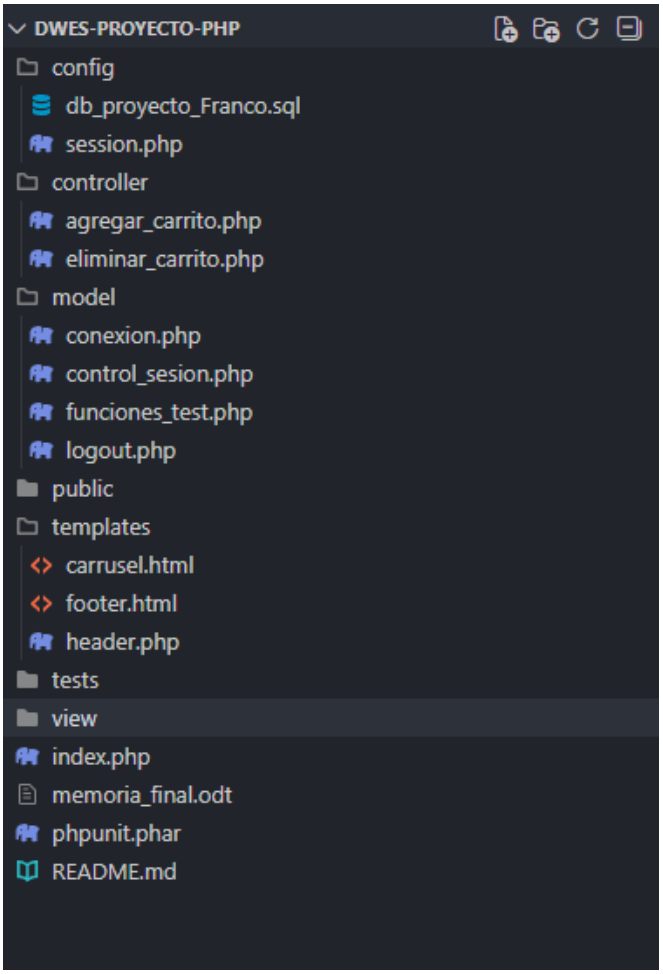

Ilustración 2: Arquitectura del proyecto

Explicación del árbol de directorios:

/config, contiene los archivos de configuración del proyecto, en este caso, la Base de Datos sobre la que se trabajará, para futuras referencias bbdd.

/controller, recibe las peticiones del usuario, procesa la lógica y coordina vistas

/model, gestiona los datos y la lógica relacionada con la base de datos.

/public, contiene los archivos accesibles públicamente como CSS, JavaScript, imágenes, tipografía, pdfs...

/view, muestra la interfaz al usuario y sus distintas vistas.

/tests, donde se almacenan los distintos tests unitarios para comprobar la compatibilidad y funcionamiento de la aplicación.

Decidí crear un nuevo directorio llamado **/templates** , donde se crearon plantillas para reutilizar en las vistas, como header.php y footer.html… estas son plantillas que se repiten y por ello era conveniente separarlas e incluirlas para no repetir código y tener una mayor facilidad para modificarlas si fuese necesario.

Por último, se incluye la memoria explicativa de las distintas decisiones del creador a lo largo del proyecto y un fichero de texto **README.md** que incluye las instrucciones para el correcto funcionamiento y ejecutar los tests funcionales del sitio web.

– Diseño del sistema y de la interfaz

Durante la segunda semana se trabajó sobre el diseño del ecosistema, creando las distintas vistas que lo componen, con las respectivas plantillas que se usarán en ellas.

Esta es la fase donde se determinó la paleta de colores que se iba a utilizar y el propósito de la misma, en este caso un aspecto sobrio y profesional.

Además, se determinó la estructura visual y el orden que iban a tener los distintos elementos tanto en la barra de navegación como en los elementos interactivos (carrusel de imágenes).

– Desarrollo de la lógica principal

En esta fase se implementará la lógica principal de la aplicación, incluyendo las operaciones básicas relacionadas con usuarios, cursos y accesos.

Se comenzará también la programación de las funciones principales del sistema, asegurando que los procesos internos funcionen correctamente antes de integrarlos con la interfaz.

– Elaboración de diagramas y estructura de datos

Durante esta semana se diseñaron y crearon los diagramas necesarios, como el Diagrama de Gantt, Diagrama de Entidad-Relación y el Diagrama Relacional que permite visualizar las relaciones entre los distintos componentes y entidades que interactúan o se almacenan en la bbdd.

Además, se diseñará y ajustará la base de datos, garantizando que soporte correctamente la persistencia de información como usuarios, cursos o carrito.

- Implementación del sistema CRUD

En esta fase se desarrollarán las operaciones CRUD (Crear, Leer, Actualizar y Eliminar) para el usuario con rol de tipo *Administrador.*

Estas funcionalidades permitirán gestionar correctamente los datos distintos cursos que los usuarios con rol de tipo *Usuario* pueden adquirir y así cumplir con los requisitos establecidos para la aplicación.

– Pruebas del sistema

Durante esta semana se realizarán pruebas unitarios en PHP sobre los distintos módulos desarrollados, comprobando que cada parte del sistema funcione correctamente.

Se corregirán errores detectados y se mejorará el rendimiento y la estabilidad de la aplicación.

– Documentación del proyecto

En esta fase se elaborará la documentación y la memoria final, explicando el funcionamiento del sistema, su arquitectura, el uso de la aplicación y las decisiones adoptadas durante el desarrollo. Esta documentación facilitará el mantenimiento futuro del sistema y su comprensión por parte de otros desarrolladores o usuarios.

– Últimos retoques y entrega final

En la última semana se realizarán los ajustes finales del sistema, mejorando la interfaz, corrigiendo pequeños errores y funcionalidades y optimizando el rendimiento general de la aplicación.

Finalmente, se preparará la versión final del proyecto para su entrega, asegurando que cumpla todos los requisitos establecidos.

3. TÉCNICAS Y HERRAMIENTAS UTILIZADAS

Para la planificación y realización del diagrama de Gantt he usado la herramienta web gratuita **Canva.**

Para ambos diagramas (Entidad-Relación y Relacional) he usado la aplicación también gratuita **drawio.**

La implementación ha sido desarrollada íntegramente siguiendo la arquitectura MVC. Para llevarla a cabo se ha hecho uso del lenguaje PHP de forma manual, si usar ningún tipo de *Framework* ya que su utilización supondría un gasto muy grande en cuanto al aprendizaje previo que tendría que emplear para poder hacer un uso efectivo de este.

Sin embargo, si se han organizado los distintos archivos en los directorios específicos para cada uno, de esta forma separando claramente la parte cliente (vistas), de la parte modelo (model) y de la parte servidora (controlador).

Para la parte cliente se ha usado **HTML5** y **JavaScript** y para la parte de diseño de interfaces **CSS3,** aunque principalmente de su *Framework* **Bootstrap.**

Para la parte de persistencia de datos, se ha decidido usar **MySQL** debido a que éste es el empleado en el módulo de Desarrollo Web en Entorno Servidor y uno de los más usados a nivel mundial.

Además, usa un modelo de código abierto y está disponible tanto si se desarrolla en Windows, donde se puede instalar a este propósito **XAMPP** o por el contrario si se desarrolla en Linux (como es mi caso), se usa **LAMPP** que corresponde a la tecnología

nativa de Linux de XAMPP. Estas herramientas incorporan un servidor web *Apache,* el intérprete de PHP y la gestión de la base de datos a través de **PHPMyAdmin.**

El editor de código elegido para la ocasión es **Visual Studio Code,** el cual es un entorno de desarrollo gratuito, altamente personalizable, incluye una cantidad enorme de extensiones y lenguajes y por supuesto, está disponible en los principales sistemas operativos en la actualidad (Windows, Linux, macOS).

Por último, que no menos importante, se utilizará la herramienta de *Sistema de Control de Versiones* mas usada del planeta, **Git** y para el almacenamiento de la misma se hará uso de **GitHub,** la principal plataforma para alojar distintos proyectos de desarrollo *Open Source.*

4. DESARROLLO DEL PROYECTO

4.1. INICIO

El proyecto consiste en crear una plataforma web dedicada a promocionar mi persona como Desarrollador Web y a la oferta de cursos sobre las distintas herramientas y lenguajes que yo, como desarrollador, conozco. Se decide desarrollar este tipo de aplicación porque la formación online es uno de los medios de aprendizaje más utilizados actualmente y permite a los usuarios mejorar sus conocimientos desde cualquier lugar y en cualquier momento.

Para llevar esta idea a cabo se realiza una página web en la que se muestran los distintos cursos disponibles y desde la que los usuarios pueden añadirlos a un carrito y gestionarlos cómodamente tras iniciar sesión.

Además, se incluye un rol *Administrador* que puede realizar diversas acciones sobre estos cursos, acciones que persisten en el tiempo y que afectan también al rol de *Usuario.*

4.2. ANÁLISIS

El proyecto debe cumplir los siguientes requisitos:

- Requisitos funcionales:

- Cualquier persona podrá acceder a la página web para consultar los diferentes cursos disponibles
- Si una persona quiere añadir cursos al carrito, tendrá que estar registrada en el sistema como *usuario*
- El *administrador* podrá gestionar los cursos, añadiendo, modificando o eliminando los ya existentes
- Se podrán consultar los cursos mostrados en un catálogo general.
- Los cursos se podrán identificar fácilmente por su nombre y descripción.
- El usuario podrá visualizar el precio de cada curso.
- Los usuarios registrados pueden eliminar productos de su carrito
- Para realizar el inicio de sesión, el sistema solicitará el nombre de usuario y contraseña.

- Requisitos no funcionales:

- El diseño debe ser sencillo, atractivo y adaptable a varios dispositivos

4.3. DIAGRAMAS

4.3.1. Modelo Entidad-Relación

El diagrama para generar el esquema para la **Base de Datos** es el siguiente:

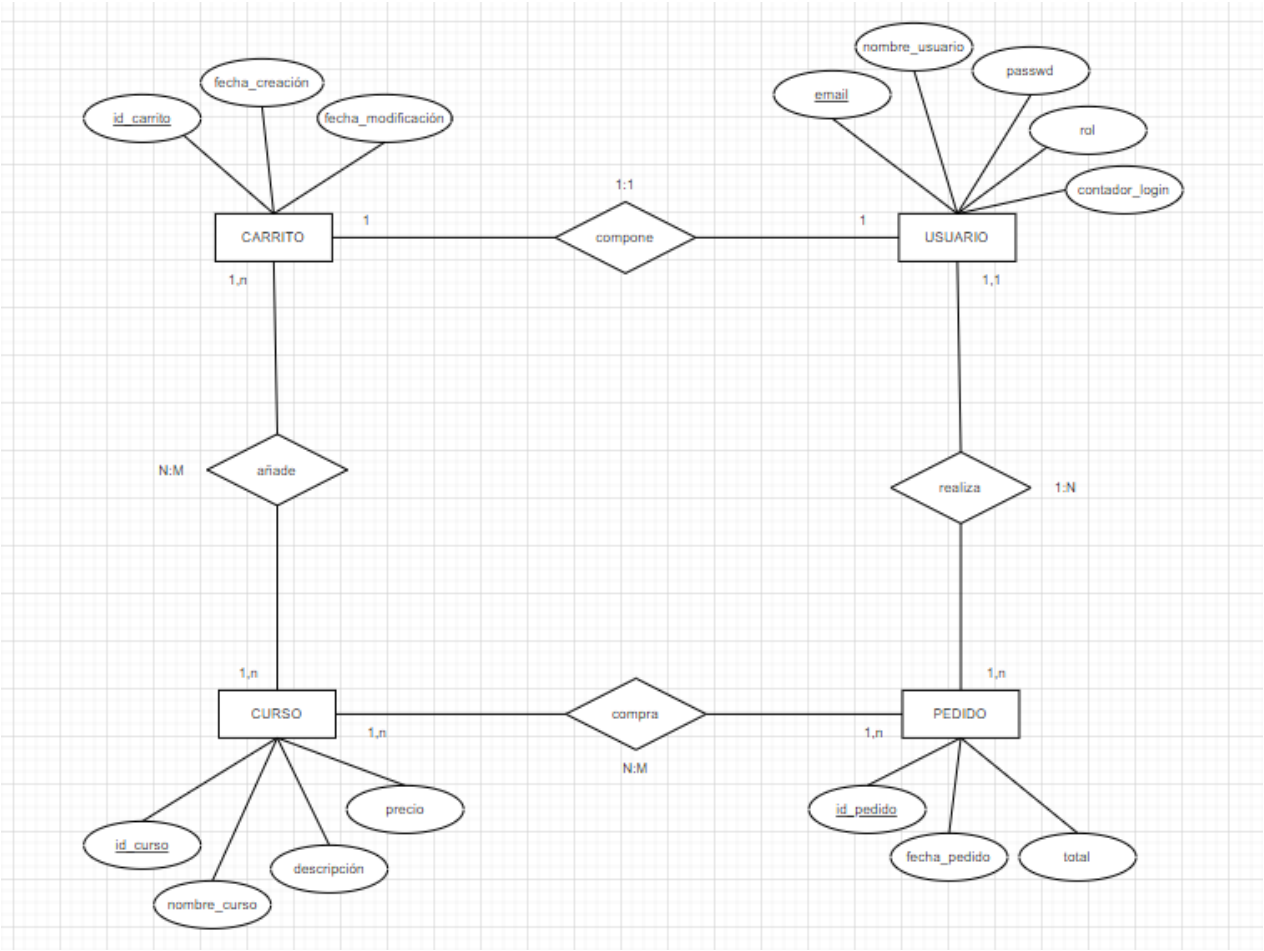

Ilustración 3: Diagrama Modelo Entidad-Relación

El paso del modelo Entidad-Relación al modelo relacional permite estructurar los datos en tablas físicas, asegurando la integridad referencial mediante claves primarias (PK) y foráneas (FK).

- Definición de Tablas Principales

- USUARIO: Se establece nombre_usuario como clave primaria. Contiene los datos de perfil como fecha del registro, el contador de visitas personalizado de cada usuario y las credenciales.

- CARRITO: Incluye una clave foránea nombre_usuario que conecta con la tabla USUARIO, implementando la relación de pertenencia.
- CURSO: Define el catálogo de cursos con id_curso como PK y almacena los atributos descriptivos y de costo.
- PEDIDO: Vincula cada transacción a un usuario mediante la FK nombre_usuario. Se decidió incluir esta tabla para acercar el proyecto lo más posible a un entorno de producción real, aunque aún no exista pasarela de pago.

- Resolución de Relaciones Muchos a Muchos (N:M)

Para normalizar el modelo y permitir que varios elementos se relacionen entre sí sin duplicar datos, se han creado dos tablas asociativas:

- CARRITO_CURSOS: Esta tabla intermedia vincula id_carrito con id_curso. Permite que un usuario gestione múltiples cursos en su cesta simultáneamente. Sus claves foráneas son, a su vez, una clave primaria compuesta.
- PEDIDO_CURSOS: Relaciona id_pedido con id_curso. Es fundamental para el histórico de ventas, ya que permite ver los cursos que componen cada factura o pedido realizado por el usuario. Esta tabla fue incluida para asemejar el proyecto a un entorno real donde los clientes si podrán realizar pagos y comprar estos cursos de forma segura.

- Integridad Referencial

El diseño garantiza que no existan huérfanos en el sistema, es decir, que la tabla pedido o carrito no existan por su cuenta.

Estas tablas necesitan de un **usuario válido** para poder existir.

Las tablas intermedias garantizan que solo se puedan comprar o añadir al carrito Cursos que existan previamente en el catálogo.

4.3.2. Relacional

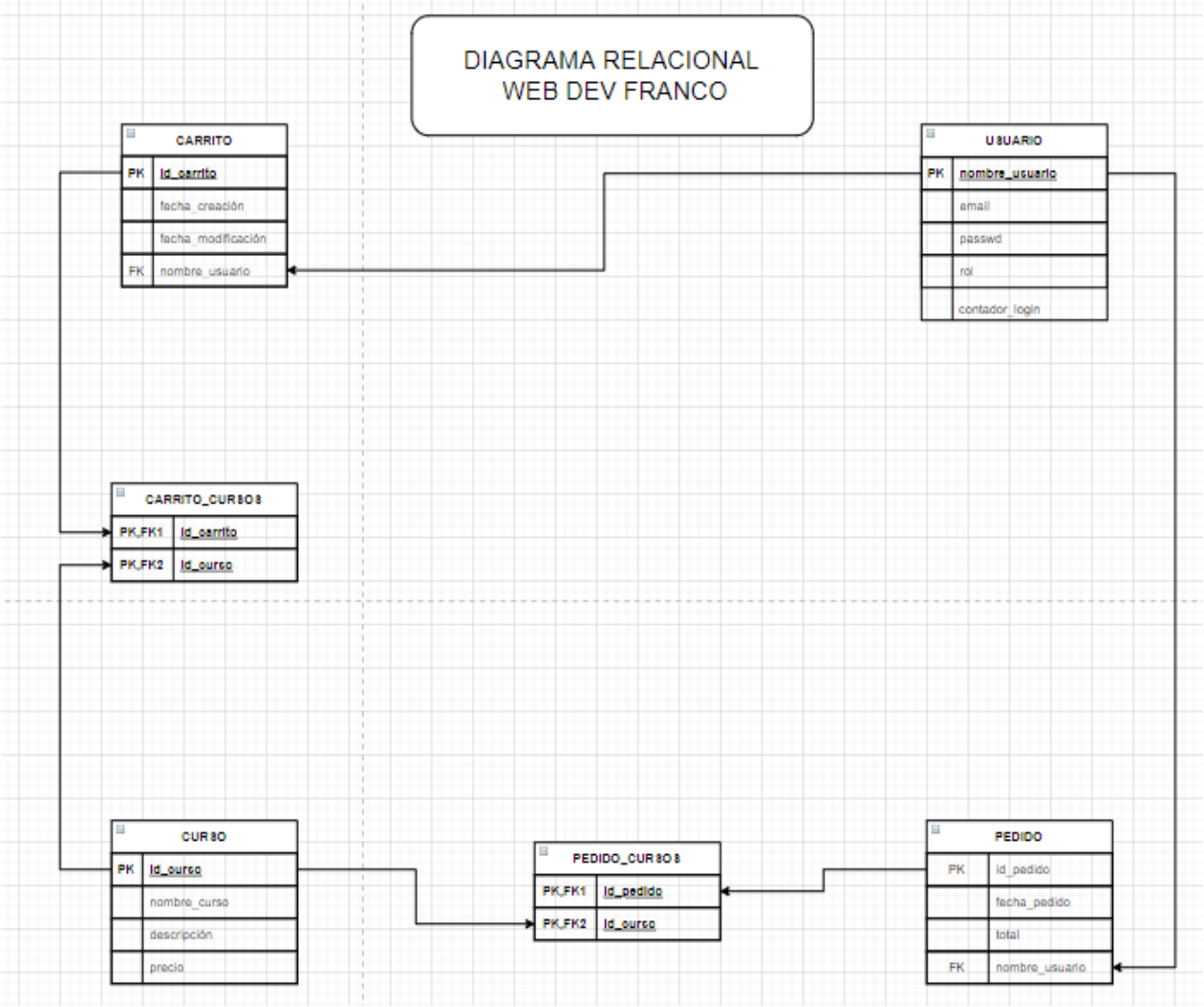

Ilustración 4: Diagrama Modelo Relacional

4.4. ARQUITECTURA DETALLADA

El proyecto utiliza una estructura modular que separa la lógica de negocio, el acceso a datos y la interfaz de usuario para facilitar el mantenimiento y la escalabilidad. En la siguiente página muestro el árbol de directorios de la misma:

4.4.1. Directorio /controller (Controladores)

Es el "cerebro" de la aplicación. Actúa como intermediario entre la vista y el modelo, gestionando las peticiones del usuario.

- **agregar_carrito.php:** Contiene la lógica para añadir un curso al carrito, validando si existe y actualizando la base de datos.
- **eliminar_carrito.php:** Gestiona la lógica para eliminar un curso del carrito del usuario.

4.4.2. Directorio /model (Modelos)

Se encarga de la interacción directa con la base de datos y la gestión de la sesión.

- **conexion.php:** Es el script que se reutiliza en las distintas vistas para mantener la conexión a la bbdd configurada a través del protocolo PDO que nos proporciona el lenguaje PHP.
- **logout.php:** Contiene la lógica para cerrar la sesión del usuario, se encarga de destruir las variables de sesión del usuario y redirigir a la página principal del sitio web.

4.4.3. Directorio /templates (Fragmentos Re utilizables)

Almacena los componentes de interfaz que se repiten en múltiples páginas para evitar la duplicación de código, mejorar el escalado futuro y mantener coherencia entre las distintas vistas, reduciendo los errores.

- **carrusel.html:** Estructura visual dinámica de los proyectos destacados anunciados en la página principal.
- **footer.html y header.php:** Pie de página y cabecera (menú de navegación) que mantienen la consistencia visual en todo el sitio.

Cabe destacar el uso de extensión del fichero .php en vez de .html estático para la cabecera ya que este es un **componente dinámico** que reacciona al estado del usuario y de la aplicación, algo que el estándar HTML plano no puede procesar; se necesita así

para poder gestionar las distintas sesiones, condicionales o poder usar las funciones nativas del lenguaje PHP como include o require.

4.4.4. Directorio /view (Vistas)

Contiene las páginas finales que el usuario visualiza. Estas páginas llaman a los controladores y muestran la información procesada.

- **carrito.php:** Es la interfaz donde el usuario revisa los cursos seleccionados antes de 'comprar'.
- **cv.php y contacto.php:** Página donde el autor promociona su currículum vitae y página donde aparece un formulario de contacto en el que se le puede contactar enviándole un mensaje a su correo electrónico.
- **crud.php:** Panel de administración para realizar operaciones de Crear, Leer, Actualizar y Borrar sobre los datos (seguramente de los cursos). Esta página sólo está visible si el usuario autenticado tiene el rol de *admin.*
- **cursos.php:** Catálogo visual que muestra la lista de cursos disponibles extraídos de la base de datos.
- **login.php y portfolio.php:** Página que contiene un formulario sencillo de autenticación y página donde se pueden visualizar los trabajos mas destacados del autor y su consiguiente enlace al repositorio público alojado en la nube de GitHub.

4.4.5. Directorio /config (Configuración)

- **db_proyecto_Franco.sql:** Contiene el script de creación de la base de datos para su posterior usos
- **session.php:** Este pequeño script sirve para centralizar la gestión de inicio de sesión. Cuando es ejecutado crea un ID de sesión, crea la variable global *$_SESSION* y permite guardar los datos del usuario entre páginas

5. DESPLIEGUE Y PRUEBAS

5.1. INTRODUCCIÓN AL PLAN DE PRUEBAS

En este apartado se describe la estrategia utilizada para asegurar que la aplicación es estable y cumple con los requisitos. El enfoque principal ha sido verificar que la lógica de programación en **PHP** y la estructura de la base de datos **MySQL** funcionan de la mano, evitando fallos en la experiencia del usuario final.

5.2. OBJETIVO DEL PLAN

El objetivo es validar que el flujo de información sea consistente desde las interfaces en las distintas vistas hasta su almacenamiento en la base de datos. Con la batería de tests se busca que:

- **La arquitectura MVC responde correctamente:** Se comprueba que los controladores en `/controller` procesan las peticiones (como añadir productos o validar login) sin errores de transferencia de datos.
- **La seguridad por roles es efectiva:** Validar que los scripts de sesión (`login.php` y `logout.php`) actúan como un muro real, impidiendo que un perfil con rol de "usuario" acceda a herramientas administrativas como `crud.php`.
- **La integridad del proceso de compra:** Asegurar que las operaciones de suma y resta de cursos en el carrito se reflejan con exactitud en el total del pedido.

5.2.1. Alcance de las pruebas

Para cumplir con los requisitos de entrega del proyecto y abarcar todas las funcionalidades posibles de la aplicación web, se han realizado una serie de pruebas unitarias en PHP que comprueben el funcionamiento de cada uno de estos elementos.

El alcance de estas pruebas se limita a los procesos críticos que garantizan el funcionamiento de la web:

- **Control de Acceso y Sesiones:** Verificación que las distintas vistas están disponibles para el público y que hay algunas que no lo están para asegurar que el sistema identifica correctamente a usuarios y administradores.

- **Ciclo de Vida del Carrito:** Se realiza un testeo exhaustivo de las funciones de agregar y eliminar cursos en los usuarios logueados, validando que la tabla asociativa `CARRITO_CURSOS` y la sesión del usuario se mantengan sincronizadas sin generar inconsistencias entre ellas.
- **Gestión Administrativa (CRUD):** Comprobación de que las funciones de creación y edición de cursos generan cambios reales en la base de datos y su consistencia general.
- **Consistencia de la Interfaz:** Confirmar que el uso de `header.php` permite mostrar menús dinámicos según el rol detectado en la sesión.

5.2.2. Lo que no se prueba

Por limitaciones de tiempo se excluyen estas pruebas que pueden ser realizadas en el futuro en un entorno de producción real:

- **Pruebas de carga masiva:** No se evaluará el rendimiento del servidor ante un tráfico de usuarios simultáneos extremo.
- **Pasarelas de pago reales:** La transacción se considera finalizada al insertar el registro en la tabla `PEDIDO` de la base de datos, omitiendo la conexión con bancos externos.
- **Compatibilidad retroactiva:** El diseño CSS con Bootstrap y la lógica JavaScript están pensados para los distintos navegadores modernos, por lo que no se garantiza el funcionamiento en versiones antiguas y obsoletas como Internet Explorer.

NOTA: Todas estas pruebas se estado probando de *forma manual* a lo largo del desarrollo durante cada una de las fases del mismo ya que son las que determinan el correcto funcionamiento de la aplicación.

5.3. PLAN DE PRUEBAS

TEST Y NOMBRE	ESPECIFICACIÓN DE PRUEBA
Acceso a vistas públicas sin autenticación	**Requisito:** El sistema debe permitir que un usuario no autenticado acceda a las vistas públicas del sistema. **Precondiciones:** No existe sesión iniciada **Pasos de prueba:** • Acceder a la URL index.php. • Acceder a la URL login.php. **Resultado esperado:** • Ambas vistas se cargan correctamente. • No se produce ninguna re dirección forzada. • No se muestran errores de sesión. **Resultado obtenido:** Favorable
Bloqueo de vistas protegidas sin login	**Requisito:** El sistema debe impedir el acceso a vistas protegidas a usuarios no autenticados. **Precondiciones:** Usuario no autenticado. **Pasos de prueba:** • Acceder manualmente a carrito.php. • Acceder manualmente a crud.php. **Resultado esperado:** • El sistema redirige al usuario a index.php o login.php. • No se muestra contenido protegido. **Resultado obtenido:** Favorable
Inicio de sesión como usuario	**Requisito:** El sistema debe permitir el inicio de sesión de usuarios registrados. **Precondiciones:** Usuario existente en la base de datos con rol “usuario”.

	Pasos de prueba: • Acceder a login.php. • Introducir credenciales válidas. • Enviar formulario. **Resultado esperado:** • Se inicia la sesión correctamente. • Se almacenan las variables $_SESSION['usuario'] y $_SESSION['rol']. **Resultado obtenido:** Favorable
	Requisito: El sistema debe permitir a usuarios autenticados añadir cursos al carrito. **Precondiciones:** Usuario autenticado. Curso existente en la base de datos.
Añadir cursos al carrito	**Pasos de prueba:** • Acceder a la vista de cursos. • Pulsar el botón “Añadir al carrito”. **Resultado esperado:** • El curso se añade correctamente a la sesión. • El carrito refleja el nuevo curso. **Resultado obtenido:** Favorable
	Requisito: El usuario debe poder visualizar los cursos añadidos al carrito. **Precondiciones:** Usuario autenticado. Al menos un curso añadido.
Visualización del carrito	**Pasos de prueba:** • Acceder a carrito.php. **Resultado esperado:** • Se muestran los cursos añadidos previamente. • Se muestran correctamente nombre y precio.

	Resultado obtenido: Favorable
Eliminación de cursos del carrito	**Requisito:** El sistema debe permitir eliminar cursos del carrito. **Precondiciones:** Usuario autenticado. Carrito con cursos añadidos. **Pasos de prueba:** • Acceder al carrito. • Pulsar el botón “Eliminar” sobre un curso. **Resultado esperado:** • El curso desaparece del carrito. • La sesión se actualiza correctamente. **Resultado obtenido:** Favorable
Inicio de sesión como administrador	**Requisito:** El sistema debe permitir iniciar sesión a administradores. **Precondiciones:** Usuario con rol “admin” existente. **Pasos de prueba:** • Acceder a login.php. • Introducir credenciales de administrador. • Enviar formulario. **Resultado esperado:** • Sesión iniciada correctamente. • Rol “admin” almacenado en sesión. **Resultado obtenido:** Favorable

Visualización del apartado CRUD	**Requisito:** El administrador debe visualizar el apartado de administración. **Precondiciones:** Usuario autenticado con rol “admin”. **Pasos de prueba:** • Acceder a la página principal. **Resultado esperado:** • Se muestra un enlace o apartado de administración (CRUD). **Resultado obtenido:** Favorable
Gestión completa de cursos por el administrador	**Requisito:** El administrador debe poder gestionar los cursos. **Precondiciones:** Usuario con rol “admin”. **Pasos de prueba:** • Acceder al CRUD. • Crear un curso. • Editar un curso existente. • Eliminar un curso. **Resultado esperado:** • Las operaciones se realizan correctamente. • Los cambios se reflejan en la base de datos. **Resultado obtenido:** Favorable

5.4. PRUEBAS UNITARIAS FRAMEWORK PHPUNIT

Para comprobar que algunas partes importantes del proyecto funcionan correctamente, se han realizado pruebas unitarias utilizando PHPUnit.

Para instalarlo en Linux Mint como es mi caso se puede usar el comando *sudo apt install phpunit* o descargar la versión más reciente con el archivo PHAR de su web oficial, que actúa de paquete sin necesidad de instalar la aplicación en el ordenador.

Los tests se ejecutan desde la terminal situándose en la raíz del proyecto con el comando:

php phpunit.phar tests/ProyectoTest.php

Al ejecutarlo, PHPUnit muestra los resultados indicando qué pruebas pasaron, cuáles fallaron y el número de aserciones realizadas, lo que permite verificar rápidamente la correcta implementación de la lógica del sistema.

El proyecto está desarrollado en PHP puro y no sigue una estructura basada en clases, por lo que no se pueden probar directamente las vistas, ya que estas contienen re direcciones, sesiones y salidas del programa.

Por este motivo, se ha separado parte de la lógica en funciones sencillas, que pueden ejecutarse de forma independiente y permiten comprobar si el comportamiento es el esperado.

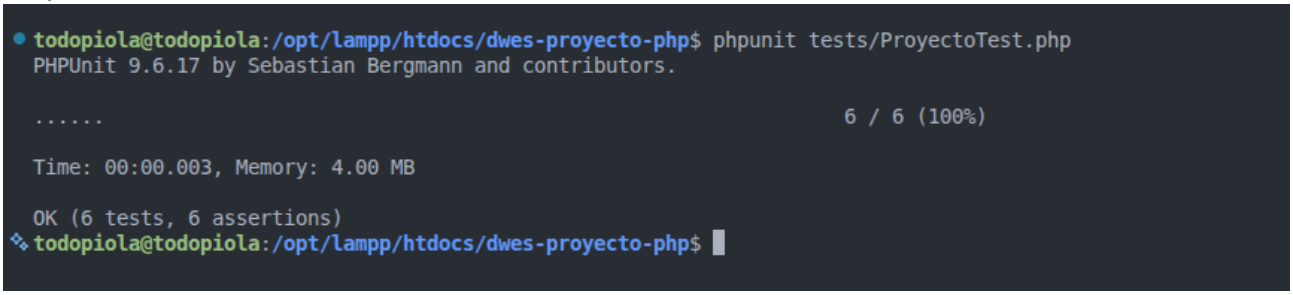

Ilustración 5: Resultados test PHPUnit

Las pruebas se centran principalmente en tres aspectos del proyecto: la sesión del usuario, el control de roles y el funcionamiento básico del carrito.

En primer lugar, se comprueba si un usuario está autenticado o no. Para ello se valida si existe la variable de sesión que identifica al usuario. Se prueba tanto el caso en el que la sesión no existe, como el caso en el que el usuario ya ha iniciado sesión correctamente.

Estas pruebas de autenticación se pueden comprobar mediante la imagen de la página siguiente:

ProyectoTest.php

tests > ProyectoTest.php

```php
use PHPUnit\Framework\TestCase;

require_once __DIR__ . '/../model/funciones_test.php';

class ProyectoTest extends TestCase
{
    protected function setUp(): void
    {
        $_SESSION = [];
    }

    /** @test */
    public function usuario_no_autenticado_no_tiene_sesion()
    {
        $this->assertFalse(usuarioAutenticado());
    }

    /** @test */
    public function usuario_autenticado_tiene_sesion()
    {
        $_SESSION['usuario'] = 'juan';
        $this->assertTrue(usuarioAutenticado());
    }
```

Ilustración 6: Tests de autenticación

También se realizan pruebas para verificar el rol del usuario. En el sistema existen usuarios normales y administradores, diferenciados por el valor del atributo `rol` guardado en la sesión. Las pruebas permiten comprobar que un usuario con rol “admin” es reconocido correctamente y que un usuario normal no obtiene permisos de administrador.

```php
/** @test */
public function usuario_admin_es_detectado_correctamente()
{
    $_SESSION['rol'] = 'admin';
    $this->assertTrue(esAdmin());
}

/** @test */
public function usuario_normal_no_es_admin()
{
    $_SESSION['rol'] = 'usuario';
    $this->assertFalse(esAdmin());
}
```

Ilustración 7: Tests de roles

funciones_test.php

model > funciones_test.php

```php
<?php
function usuarioAutenticado(): bool {
    return isset($_SESSION['usuario']);
}

function esAdmin(): bool {
    return isset($_SESSION['rol']) && $_SESSION['rol'] === 'admin';
}

function carritoTieneCursos(array $cursos): bool {
    return count($cursos) > 0;
}
```

Ilustración 8: Funciones sencillas pruebas PHPUnit

Por último, se prueba el estado del carrito de la compra. En este caso, las pruebas verifican si el carrito contiene cursos o está vacío. Esto es útil para controlar situaciones como mostrar mensajes al usuario o impedir acciones cuando el carrito no tiene productos.

```php
    /** @test */
    public function carrito_vacio_no_tiene_cursos()
    {
        $cursos = [];
        $this->assertFalse(carritoTieneCursos($cursos));
    }

    /** @test */
    public function carrito_con_cursos_devuelve_true()
    {
        $cursos = [
            ['id_curso' => 1, 'nombre_curso' => 'PHP']
        ];

        $this->assertTrue(carritoTieneCursos($cursos));
    }
}
```

Ilustración 9: Tests del carrito

Estas pruebas permiten detectar errores de forma temprana y comprobar que los cambios realizados en el código no afectan al funcionamiento básico del sistema. Aunque no se prueban directamente las vistas, sí se valida la lógica principal que utilizan, lo que mejora la fiabilidad general del proyecto.

5.5. VALORACIÓN FINAL DE PRUEBAS

Una vez completada la batería de pruebas, se pudo comprobar que la aplicación cumple con los requisitos la conclusión es que la aplicación es estable y cumple con la lógica de negocio planteada al principio.

Lo más positivo de este proceso de testeo ha sido que el sistema de permisos realmente funciona: un usuario estándar o uno que no está logueado, no tiene forma de visualizar el panel de administración, y el header.php reacciona bien cambiando los menús según quién esté logueado.

Durante las pruebas del carrito, se verificó que la comunicación entre los controladores y la base de datos es sólida. Los cursos se añaden y se quitan de las tablas intermedias sin dejar datos huérfanos, lo que garantiza que el carrito sea consistente con los

Quisiera destacar que el uso del modelo **MVC** ha facilitado mucho la corrección de pequeños fallos detectados durante las pruebas.

Al tener la lógica separada en /controller y las vistas en /view, cualquier error en el flujo de datos se pudo localizar y arreglar rápidamente sin romper el diseño de la web.

6. CONCLUSIONES

6.1. OBJETIVOS ALCANZADOS

Se han cumplido la mayoría de los objetivos funcionales que se plantearon al inicio del proyecto y en definitiva, como autor estoy bastante satisfecho con el resultado final alcanzado.

El sistema permite que los usuarios se registren, inicien sesión, vean los cursos disponibles, agreguen productos al carrito y gestionen los cursos desde la cuenta de administrador.

6.2. CONCLUSIONES DEL TRABAJO

Desarrollar este proyecto ha sido bastante exigente en cuanto a tiempo se refiere.

Ha supuesto no solo programar las vistas y la lógica de la aplicación, sino también saber como manejar eficientemente sesiones, roles, cookies, y a estructurar correctamente la base de datos para que el carrito y los usuarios funcionen bien.

A pesar de las dificultades, el trabajo ha resultado satisfactorio porque se ha logrado un sistema funcional, que se puede probar y que cumple con lo esperado en los requisitos primarios.

6.3. TODO: FUTURAS MEJORAS

Hay varias mejoras que se podrían hacer en el futuro:

- Añadir un sistema de pagos (pasarela de pago real) para que los usuarios puedan comprar los cursos desde la aplicación.
- Enviar confirmaciones de compra por correo electrónico.
- Mejorar la interfaz visual para hacerla más atractiva y fácil de usar.
- Incorporar frameworks de frontend como React para una experiencia más dinámica, y frameworks de backend como Laravel para un código más organizado y seguro.
- Ampliar y personalizar los detalles de los cursos, por ejemplo con imágenes específicas, valoraciones y categorías.
- Añadir mas contenido en las distintas vistas
- Analizar el mercado para posicionar la aplicación web en las primeras posiciones en los distintos buscadores

En general, el proyecto sirve como una base sólida para seguir desarrollando nuevas funcionalidades y mejorar la experiencia del usuario.

7. GLOSARIO TECNOLOGÍAS Y HERRAMIENTAS

SEO

Search Engine Optimization. Conjunto de técnicas para mejorar la visibilidad de una página web en motores de búsqueda como Google.

React

Librería de JavaScript para construir interfaces de usuario dinámicas mediante componentes re utilizables.

Laravel

Framework de PHP que facilita la creación de aplicaciones web con estructuras limpias, rutas, base de datos y seguridad integrada.

CSS

Cascading Style Sheets. Lenguaje que define la apariencia de los elementos HTML en una página web, como colores, fuentes y diseño.

Bootstrap

Framework de CSS que permite diseñar páginas web responsivas y visualmente atractivas de forma rápida con clases predefinidas.

CRUD

Acrónimo de Create, Read, Update, Delete. Conjunto de operaciones básicas para manejar datos en una base de datos.

Git

Sistema de control de versiones que permite guardar y gestionar los cambios realizados en un proyecto de forma ordenada.

GitHub

Plataforma en línea que utiliza Git para almacenar repositorios, colaborar en proyectos y gestionar el desarrollo de software.

XAMPP

Paquete que incluye Apache, MySQL, PHP y Perl, usado para montar un servidor web local para pruebas y desarrollo.

MySQL
Sistema de gestión de bases de datos relacional que permite almacenar, consultar y organizar datos de manera estructurada.

draw.io
Aplicación web para crear diagramas, esquemas, flujogramas y mapas conceptuales de manera rápida y colaborativa.

PHPUnit: Herramienta de testing para PHP que permite crear y ejecutar pruebas unitarias de funciones y clases, verificando que el código se comporte correctamente ante distintos escenarios.

8. BIBLIOGRAFÍA WEB

- **PHP: Manual Oficial** – https://www.php.net/manual/es/
  Documentación oficial de PHP, consultada para funciones de sesión, manejo de formularios y conexión a bases de datos.
- **Bootstrap 5: Documentación** – https://getbootstrap.com/docs/5.3/getting-started/introduction/
  Referencia para la maquetación, estilos responsivos y componentes visuales utilizados en la interfaz de la web.