**DOCUMENTO DE ESTUDIO DEFENSA ORAL**

**Tecnologías utilizadas en el proyecto**

**✅ 1. ¿QUÉ ES CODEIGNITER?**

🔹 ¿Qué es?

CodeIgniter es un framework de desarrollo web en PHP. Es decir, una estructura prehecha que nos ayuda a crear sitios web más rápido, organizado y seguro.

🔹 ¿Para qué sirve?

* Para organizar el proyecto con el modelo MVC: Modelo, Vista y Controlador.
* Para separar la lógica del backend (PHP) del diseño (HTML, CSS).
* Para evitar repetir código.
* Para facilitar las rutas y el manejo de formularios.

🔹 ¿Qué hace CodeIgniter en tu proyecto?

* Controla qué vista se carga con cada URL.
* Centraliza la lógica de las páginas en los controladores (Home.php).
* Usa vistas (.php) para mostrar contenido con HTML + PHP.
* Permite trabajar con rutas limpias, como home/quienes.

**✅ 2. ¿QUÉ ES HTML?**

🔹 ¿Qué es?

HTML significa HyperText Markup Language, y es el lenguaje base de todas las páginas web.

🔹 ¿Para qué sirve?

* Estructura el contenido de una página.
* Define títulos, párrafos, imágenes, formularios, botones, etc.

🔹 En tu proyecto:

* Todo lo que ves en pantalla (textos, botones, títulos) está hecho en HTML dentro de las vistas .php.
* Por ejemplo: <h2>Contacto</h2> crea un título.

**✅ 3. ¿QUÉ ES CSS?**

🔹 ¿Qué es?

CSS significa Cascading Style Sheets (hojas de estilo en cascada).

🔹 ¿Para qué sirve?

* Para darle estilo visual al HTML.
* Define colores, tamaños, márgenes, fondos, animaciones, etc.

🔹 En tu proyecto:

* El archivo estilo.css define los colores del fondo, el diseño de las tarjetas, los botones, el fondo de secciones, etc.
* Por ejemplo:

css

CopiarEditar

.section-dark {

background-color: #1e1e1e;

color: white;

}

Cambia el color de fondo y texto de una sección oscura.

**✅ 4. ¿QUÉ ES BOOTSTRAP?**

🔹 ¿Qué es?

Bootstrap es un framework de CSS.

🔹 ¿Para qué sirve?

* Te da componentes listos para usar como botones, formularios, columnas, tarjetas, etc.
* Facilita que el sitio sea responsive (se vea bien en PC y celular).
* Usa una estructura de grillas con clases como row, col-md-6, etc.

🔹 En tu proyecto:

* Las tarjetas, botones, containers, formularios y colores están hechos con Bootstrap.
* Ejemplo:

html

CopiarEditar

<div class="container">

<div class="row">

<div class="col-md-6">Contenido</div>

</div>

</div>

**✅ 5. ¿QUÉ ES PHP?**

🔹 ¿Qué es?

PHP es un lenguaje de programación del lado del servidor. Es lo que genera las páginas web en base a lógica y datos.

🔹 ¿Para qué sirve?

* Para construir dinámicamente HTML.
* Para mostrar información según lógica (ej: mostrar una vista distinta según la URL).
* Se usa para manejar formularios, bases de datos y rutas.

🔹 En tu proyecto:

* PHP se usa en las vistas para mostrar contenido:

php

CopiarEditar

<img src="<?= base\_url('assets/img/tobias.png') ?>">

* <?= ?> es una forma de imprimir texto en HTML.
* También se usa para cargar las vistas (return view('contacto') en el controlador).

*✅ ¿CÓMO RESPONDER SI TE PREGUNTAN?*

**¿Qué es CodeIgniter y para qué lo usaron?** — Es un framework PHP que nos ayudó a organizar el proyecto siguiendo el patrón MVC. Lo usamos para separar el diseño (vistas) de la lógica (controladores), y para tener URLs limpias y una estructura ordenada.

**¿Qué hace Bootstrap en su proyecto?** — Bootstrap nos da componentes y clases listas para usar, como formularios, columnas y botones. Nos ayudó a hacer que el sitio se vea bien en cualquier dispositivo (responsive).

**¿Qué diferencia hay entre HTML y CSS?** — HTML estructura el contenido (como los títulos y textos), mientras que CSS lo diseña visualmente (colores, márgenes, fondos, etc).

¿Cómo se conectan las vistas con el controlador? — En el controlador usamos return view('nombre'), y eso carga un archivo .php con HTML y PHP mezclado para mostrar la página al usuario.

¿Qué es una vista? — Una vista es una página que muestra el contenido. Es un archivo .php que mezcla HTML con PHP para mostrar la información que viene del controlador.

**6. Vistas (Views/)**

* **Archivos clave:**
  + nueva\_plantilla.php: página principal con hero section y productos destacados.
  + quienes.php: información sobre el equipo, misión y trayectoria.
  + comercializacion.php: tipos de envío, formas de pago y entregas.
  + contacto.php: formulario y datos de la empresa.
  + terminos.php: condiciones de uso.
  + nuevos\_jugadores.php: introducción al juego D&D.
* **Funcionan con:** <?= $this->extend('layout') ?> y <?= $this->section('contenido') ?> para usar la plantilla layout.php.

**7. layout.php (Plantilla base)**

* Contiene el esqueleto HTML común: encabezado <head>, inclusión de estilos, barra de navegación y footer.
* Dentro, <?= $this->renderSection('contenido') ?> indica dónde se va a insertar el contenido de cada página.

**8. assets/**

* assets/css/estilo.css: contiene todos los estilos personalizados.
* assets/img/: contiene las imágenes utilizadas en cada sección.
* assets/js/bootstrap.bundle.min.js: archivo JS local de Bootstrap.

**9. Recursos externos**

* Font Awesome para los íconos (<i class="fas fa-dragon"></i>)
* Bootstrap para estilos y estructura
* Google Fonts o fuentes personalizadas como Cinzel

**📂 Archivo: layout.php**

Es la **plantilla base** que se reutiliza en **todas las páginas** de tu sitio web. Define la estructura general del HTML y **centraliza partes comunes** como el header (navbar), footer, y los archivos CSS/JS necesarios.

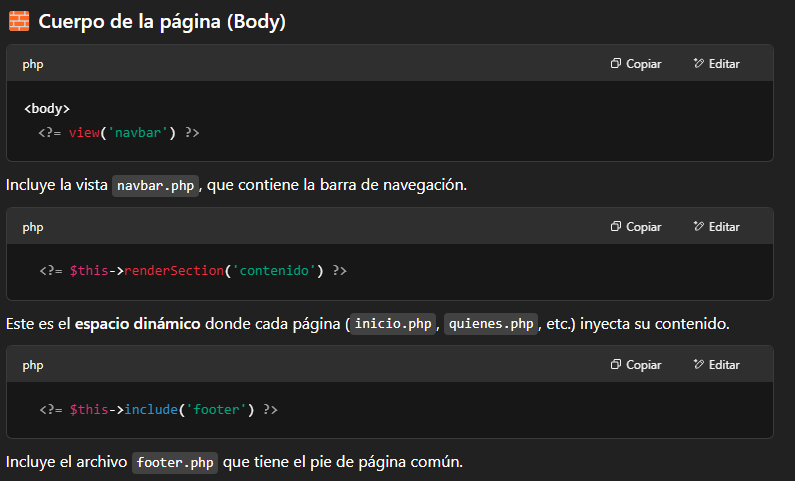
**📄 Código completo explicado línea por línea**



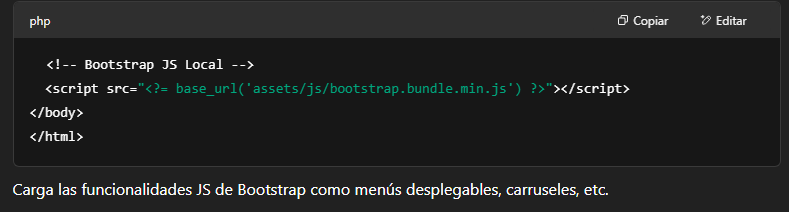
**🎨 CSS y Bootstrap**

* **Bootstrap**: Framework CSS para diseño responsivo.
* **estilo.css**: Tus estilos personalizados.
* **Font Awesome**: Íconos personalizados como dragones, sobres, etc.

**🧱 Cuerpo de la página (Body)**



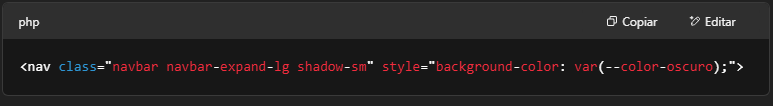
**🔧 JS (interactividad)**



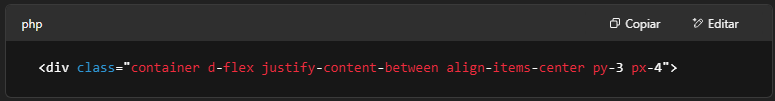
**🧠 ¿Qué tenés que entender y decir?**

1. **Sirve para reutilizar el diseño base** del sitio.
2. Evita repetir código en todas las páginas.
3. Usa las funciones extend(), section() y include() de CodeIgniter.
4. Permite mantener consistencia visual y facilita el mantenimiento.

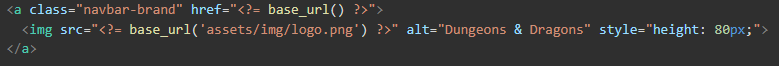
**📂 Archivo: navbar.php**

****

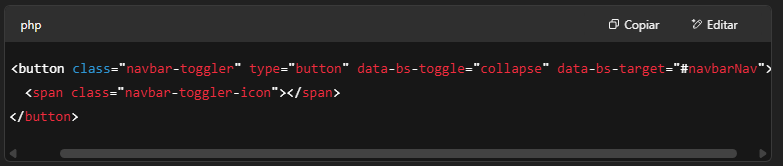
* Crea una **barra de navegación** (<nav>) con clases de Bootstrap para que sea responsive (navbar, navbar-expand-lg) y tenga una sombra sutil (shadow-sm).
* Se le asigna un **color de fondo oscuro** usando una variable definida en estilo.css.



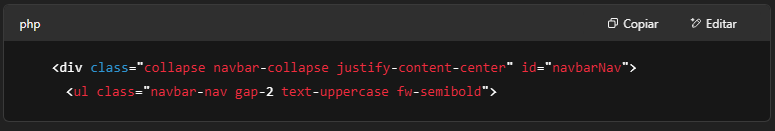
* container: agrupa el contenido en un ancho centrado.
* d-flex justify-content-between align-items-center: distribuye el logo y los ítems de menú de forma espaciada y alineada verticalmente.
* py-3 px-4: agrega padding (espacio interno) vertical y horizontal.



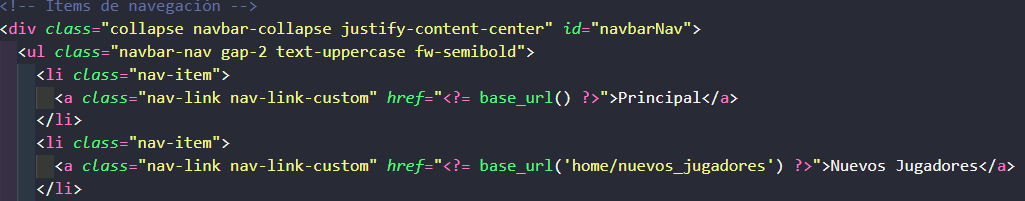
* **Logo del sitio**, que al hacer clic lleva a la página principal.
* Usa base\_url() para generar correctamente la URL en CodeIgniter.



* Es el **botón hamburguesa**, que aparece en dispositivos móviles para desplegar el menú.



* Contenedor que se **colapsa o despliega** cuando se usa el botón hamburguesa.
* ul.navbar-nav: lista de elementos de menú.
* gap-2: separa los ítems.
* text-uppercase y fw-semibold: pone el texto en mayúscula y en negrita media.



* Cada li representa un **ítem de navegación**.
* nav-link-custom es una clase personalizada de tu CSS para hover y color.

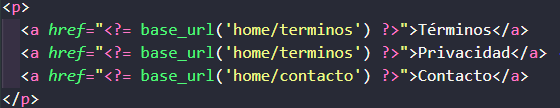
**📂 Archivo: footer.php**



* Pie de página que se alinea y ajusta automáticamente al centro de la página.



* Muestra el copyright.
* <?= date('Y') ?> imprime el **año actual** automáticamente.



* Links a las páginas importantes del sitio.
* Usa base\_url() de CodeIgniter para que las URLs funcionen sin importar en qué entorno esté alojado el sitio.

**🔁 Relación con layout.php**

Tanto navbar.php como footer.php son **incluidos en el layout.php**, y eso permite que cada vista tenga una estructura común sin repetir código.

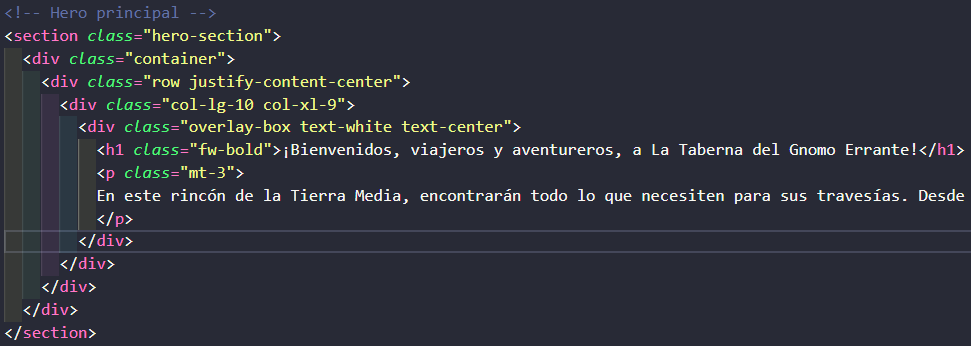
**📂 Archivo: nueva\_plantilla.php**

Este archivo representa la **vista principal** de tu sitio web dentro del framework **CodeIgniter 4**, y está diseñada para mostrarse cuando un visitante entra al inicio de tu proyecto. Se trata de una página de bienvenida con secciones informativas, visualmente atractiva gracias a **Bootstrap**, **CSS personalizado**, e imagenes.

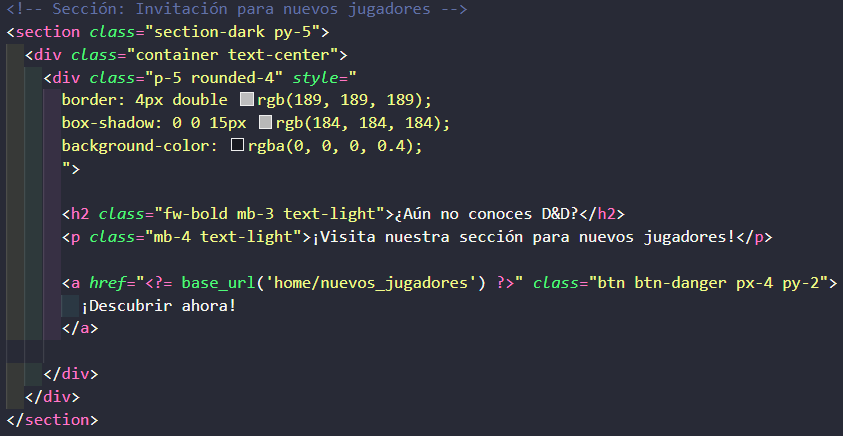
**1. CABECERA DE VISTA**



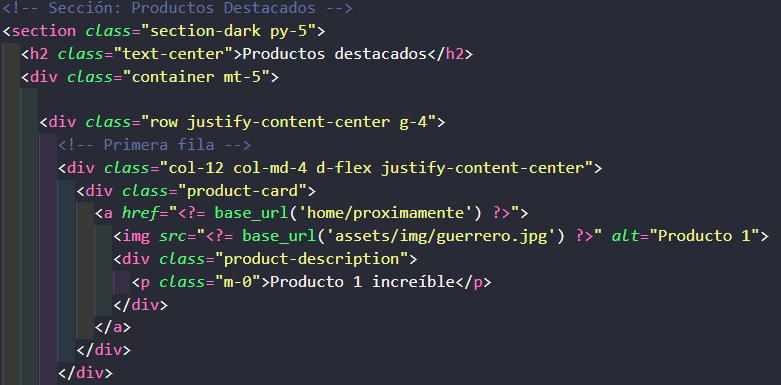
* **extend('layout'):** hace que esta vista se **inserte** dentro del archivo **layout.php**, que contiene la estructura común (navbar, footer, etc.).
* **section('contenido'):** abre una sección llamada "contenido" que se colocará en la parte definida por **<?= $this->renderSection('contenido') ?>** dentro del layout.

**2. HERO PRINCIPAL**

* **.hero-section:** definido en CSS para mostrar una imagen de fondo.
* **.container, .row, .col-lg-10:** estructuras de Bootstrap para centrar el contenido.
* **.overlay-box:** capa oscura encima del fondo para mejor lectura.
* Contenido introductorio del sitio, con un mensaje de bienvenida y descripción del objetivo de la "taberna".

**3. SECCIÓN PARA NUEVOS JUGADORES**

* **.section-dark:** fondo oscuro definido en CSS.
* **.rounded-4**: bordes redondeados con estilo visual.
* **base\_url('home/nuevos\_jugadores'):** genera un link a otra vista que introduce el juego a los nuevos.
* Visualmente esta sección actúa como un llamado a la acción para los visitantes nuevos.

**4. PRODUCTOS DESTACADOS**

* Otra sección oscura con productos destacados.
* Se usa un **container** y **row** para distribuir los productos en filas y columnas.
* Cada producto se muestra en un **.product-card** (estilizado por CSS):
* **.product-card:** tarjeta con sombra y efecto hover.
* **.product-description:** pie de la tarjeta con el nombre del producto.
* **base\_url('home/proximamente'):** simula un link que en un futuro podría llevar a la página de compra o detalle.

Se muestran **6 productos** en total, distribuidos en 2 filas de 3 productos cada una.

**5. CIERRE DE LA SECCIÓN**

**<?= $this->endSection() ?>**

* Esto cierra el bloque de contenido iniciado con **section('contenido').**
* Todo lo que está entre **section()** y **endSection()** será inyectado dentro del layout.php en el lugar correspondiente.

**RESUMEN DE TECNOLOGÍAS USADAS**

* **PHP**: lenguaje del backend para procesar vistas y controladores.
* **CodeIgniter 4**: framework MVC que organiza el proyecto por secciones (Modelos, Vistas, Controladores).
* **HTML**: estructura de las páginas.
* **Bootstrap**: para clases de diseño como row, col-md-4, text-center, btn, etc.
* **CSS personalizado** (archivo estilo.css): para efectos especiales como product-card, overlay-box, hero-section, etc.
* **Imágenes**: para enriquecer la estética del sitio y representar productos o temáticas del juego.

**📂 Archivo: nuevos\_jugadores.php**



📌 **¿Qué hace?**  
Indica que esta vista **hereda** una plantilla base llamada layout.php.  
Esto forma parte de la funcionalidad de **plantillas** en CodeIgniter (vista extendida), que permite evitar repetir el <html>, <head>, <footer>, etc., en cada página.  
La plantilla general define dónde se inserta el contenido con <?= $this->renderSection('contenido') ?>.



📌 **¿Qué hace?**  
Inicia una sección de contenido que se va a insertar dentro del layout.  
El contenido entre section('contenido') y endSection() se "inyecta" en la plantilla base (layout.php) donde esté renderSection('contenido').

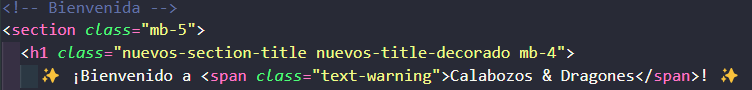


📌 **¿Qué hace?**  
Inicia un div con clase personalizada que **estiliza el fondo, los colores y el espaciado**.  
Está definido en tu estilo.css, normalmente con un **fondo animado, oscuro y blur** (estilo "épico").

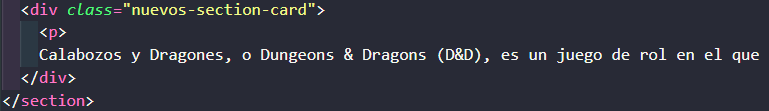


📌 **¿Qué hace?**  
Es una clase de **Bootstrap** que centra el contenido y lo alinea al medio.  
container le da márgenes automáticos y un ancho adaptable, mientras que text-center alinea el texto.

**🧙 Sección: Bienvenida**

📌 **¿Qué hace?**  
Crea un título principal.

* nuevos-section-title y nuevos-title-decorado: clases personalizadas para darle tamaño, color y sombra al título.
* text-warning: clase de Bootstrap que pinta de amarillo el texto (⚠️), para resaltar “Calabozos & Dragones”.

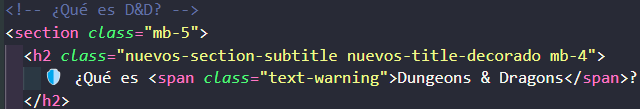
📌 **¿Qué hace?**  
Es un párrafo **informativo** envuelto en una **carta (card)** de estilo *glassmorphism* con blur y fondo translúcido.

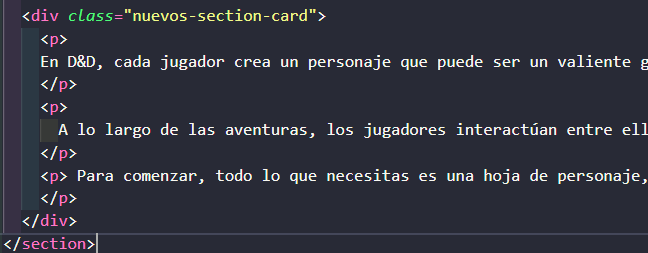
**➖ Divisor**



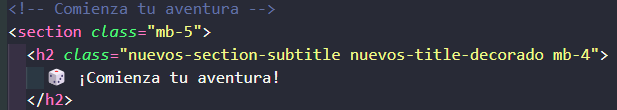
📌 **¿Qué hace?**  
Agrega una línea horizontal **decorativa y con gradiente rojo**, definida en el CSS como separación entre secciones.

**🛡️ Sección: ¿Qué es D&D?**

📌 **¿Qué hace?**  
Subtítulo con ícono, colores y estilo similar al título anterior.

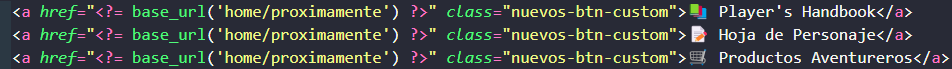
📌 **¿Qué hace?**  
Tres párrafos con información explicativa sobre la dinámica del juego de rol.  
El div les da un fondo uniforme translúcido para que el texto sea fácil de leer.

**🎲 Sección: ¡Comienza tu aventura!**

📌 Subtítulo final con ícono temático.

📌 Este bloque usa **Bootstrap** para mostrar tres botones:

* En **columna vertical** en pantallas chicas (flex-column)
* En **fila horizontal** en pantallas medianas o grandes (flex-md-row)
* gap-3: deja espacio entre botones

📌 **¿Qué hace cada uno?**  
Botones que simulan enlaces a recursos importantes:

* Manual del jugador
* Hoja de personaje
* Tienda de productos

La clase nuevos-btn-custom les da estilo personalizado con transparencias, bordes, hover y sombras.



📌 **¿Qué hace?**  
Finaliza la sección de contenido. Todo lo que hay entre section() y endSection() se va a mostrar en el layout dentro del renderSection('contenido').

**📂 Archivo: quienes.php**



🧩 Esta línea indica que **se extiende desde una plantilla base**, en este caso layout.php.  
Esto es típico de CodeIgniter 4 y evita repetir HTML general como el <head>, el navbar o el footer.  
Todo lo que esté dentro de section('contenido') va a ser "inyectado" dentro del layout.



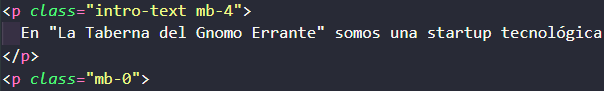
🔧 Inicia una sección llamada contenido, que es el bloque principal que luego se mostrará en <?= $this->renderSection('contenido') ?> del layout.

**🧙‍♂️ Sección: ¿Quiénes Somos?**

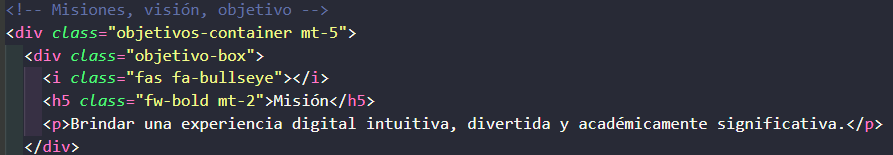
🧱 Inicia una sección con clase personalizada section-quienes (estilizada en tu estilo.css), y con text-center de Bootstrap para centrar el texto.



📦 container es un contenedor Bootstrap para centrar contenido y darle márgenes automáticos.  
fw-bold = fuente en **negrita**.  
mb-4 = margin-bottom: 1.5rem

📘 Dos párrafos explicando quiénes son, con clases que ajustan su espaciado (mb-4 = margen abajo, mb-0 = sin margen).  
Uno tiene clase intro-text, que en tu CSS define un **ancho máximo y tipografía más grande** para que sea legible.

**🎯 Misión, Visión, Objetivo**

🎯 Este contenedor agrupa 3 tarjetas horizontales: Misión, Visión, Objetivo.

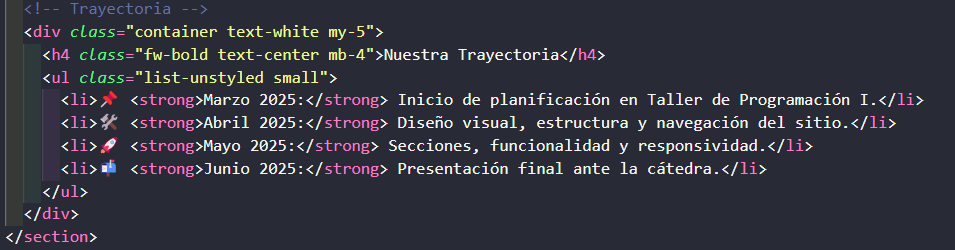
📌 Cada **objetivo-box** representa una de las tarjetas.

* <i class="fas fa-bullseye"></i> es un ícono de Font Awesome (blanco y temático).
* Título (h5) en negrita con margen arriba
* Párrafo explicativo

Esto se repite 3 veces cambiando el ícono y texto:

* fa-eye para **Visión**
* fa-compass-drafting para **Objetivo**

**🕰️ Trayectoria**

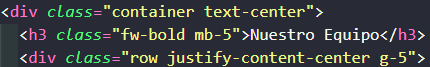
📌

* list-unstyled elimina viñetas.
* small achica la fuente.
* Cada li tiene un ícono (emoji) + fecha + descripción.
* Muy útil para mostrar organización y etapas del desarrollo.

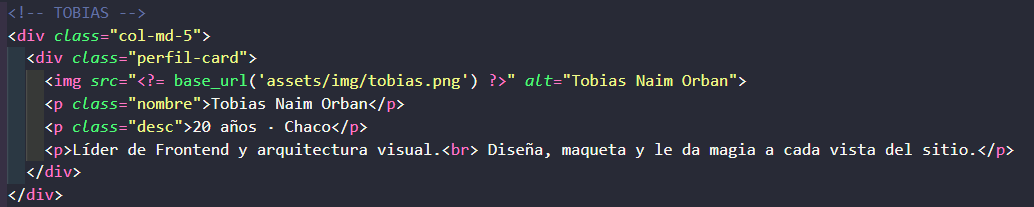
**🧑‍💻 Nuestro Equipo**



🌑 Sección con fondo oscuro. Tiene diseño de "tarjetas de perfil" y muestra quiénes trabajaron en el proyecto.



Título centrado en negrita.

**👤 Tarjeta: Tobias**

* Se usa una **columna de 5/12 del ancho total** (col-md-5) para que la tarjeta no ocupe toda la pantalla.
* perfil-card es una clase personalizada que da sombra, borde redondeado, centrado, color blanco, etc.
* Se muestra imagen, nombre, edad + descripción de su rol.

**👤 Tarjeta: Juan**

🟫 Igual que Tobias pero con otros datos e imagen (juan.png).  
Se resalta que es el encargado de la lógica y backend, como los controladores y rutas.



🧩 Finaliza la sección de contenido. Todo el contenido entre section('contenido') y endSection() será incrustado en el layout principal.

**📂 Archivo: comercializacion.php**



🧩 Esta línea extiende el archivo layout.php, que contiene la estructura general del sitio: <head>, <body>, navbar y footer. Así evitás repetir código.



🔧 Marca el comienzo del contenido principal que se va a insertar dentro del layout. Todo lo que esté entre section y endSection será la parte dinámica de la página.

**🛒 Sección de Comercialización**



📦 Inicia una sección HTML con clase comercial-section que está definida en tu CSS para tener un fondo con imagen, efecto dark y padding.

* py-5 (padding vertical grande) proviene de Bootstrap.

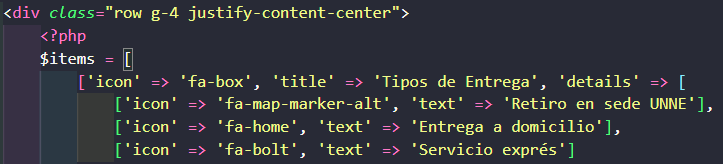


🧱 container centra el contenido y le da margen automático a los lados.  
text-center alinea todo el texto al centro.

📌 Título y subtítulo.

* fw-bold = texto en negrita
* mb-4 y mb-5 = margen inferior
* text-light = texto blanco (Bootstrap)

**🧙 Lógica PHP para mostrar tarjetas**

📜 Se define un **array $items** que contiene las secciones que se van a mostrar:

* Cada item tiene un ícono (icon)
* Un título (title)
* Un array details con 3 elementos, cada uno con su ícono y texto.

Este array representa visualmente las **4 columnas** de la sección:

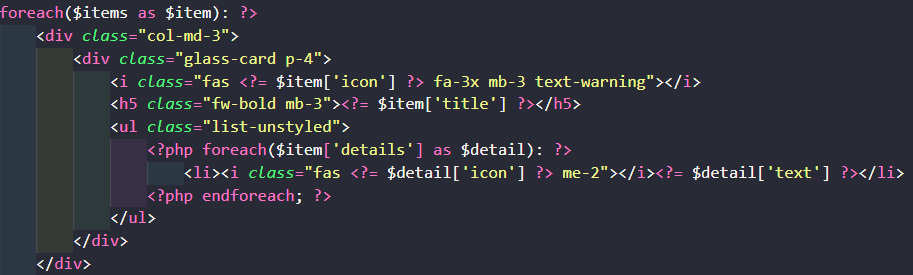
1. Tipos de entrega
2. Formas de envío
3. Formas de pago
4. Información útil

Esta es una práctica común: tener datos en un array y renderizarlos dinámicamente con foreach.

**🔁 Renderizado dinámico con foreach**



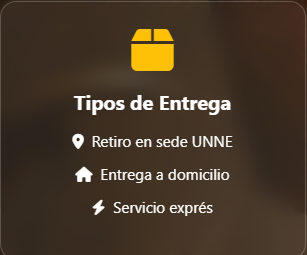
🔁 Este bucle recorre cada elemento del array $items.



🧱 Por cada elemento ($item) se genera una **columna Bootstrap de 3 columnas (col-md-3)**, es decir, 4 columnas por fila en pantallas medianas en adelante.

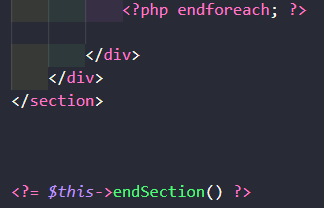
* glass-card: clase personalizada que da efecto "cristal" (glassmorphism)
* <i class="fas ...">: ícono principal de la tarjeta, estilo Font Awesome
* ul.list-unstyled: lista sin viñetas
* Segundo foreach: recorre cada subelemento de details para crear ítems de lista con ícono.

Ejemplo visual de cómo se ve uno:



Esto se repite para cada grupo del array (Formas de envío, etc.).

**🔚 Cierre**



🧩 Se cierra el bucle foreach, el row, el container, la section y finalmente el bloque de contenido.

**🧠 ¿Qué podés decir en la defensa?**

Esta sección cumple con la consigna de detallar la comercialización. Lo hicimos dividiendo la información en tarjetas visuales y dinámicas, agrupadas con Bootstrap, y usando PHP para que el código sea escalable. Cada tarjeta tiene un ícono temático, fondo con efecto glass, y texto claro. Todo el contenido es generado desde un array, lo que facilita futuras modificaciones.

**📂 Archivo: contacto.php**



📦 Extiende el layout general (layout.php), lo cual evita repetir todo el HTML base, incluyendo el <head>, navbar y footer. Todo lo que se agregue aquí será renderizado dentro de <?= $this->renderSection('contenido') ?> en ese layout.



🧩 Abre el bloque que indica *dónde se coloca el contenido principal específico de esta vista* dentro del layout.

**📍 Sección principal de contacto**



🌌 Esta línea inicia una sección con fondo oscuro (section-dark) y espaciado vertical (py-5). section-dark es una clase que está definida en estilo.css para establecer fondo negro y texto blanco.



📦 container de Bootstrap que centra el contenido horizontalmente y aplica padding.



🧾 Título y subtítulo:

* fw-bold = fuente en negrita
* text-center = centrado
* mb-4 / mb-5 = márgenes inferiores

**📇 Tarjeta con Datos y Formulario**



🃏 Se crea una tarjeta (card) con sombra grande (shadow-lg), padding (p-4) y fondo oscuro.



📐 Divide el contenido en 2 columnas con row, que usará col-md-6 para dividir 50% y 50% en pantallas medianas hacia arriba.

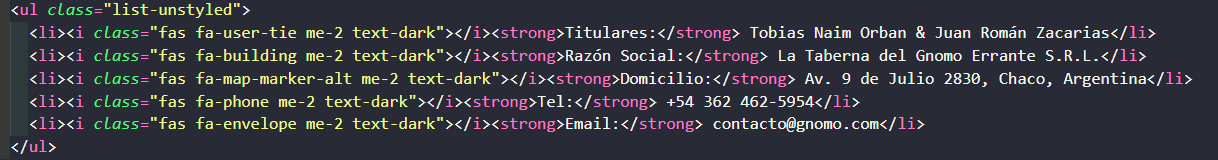
**🧾 Columna 1: Datos de la Empresa**



🔻 Columna de 6 columnas (en grid de 12) + border-end = borde derecho vertical para separar visualmente.

🐉 Título con ícono de dragón (Font Awesome), color rojo (text-danger) y margen derecho (me-2 = margin-end). Visualmente:

🐉 Datos de la Empresa

📑 Lista sin viñetas (list-unstyled), donde se muestran los datos:

* Titulares
* Razón social (nombre de fantasía de la empresa)
* Domicilio
* Teléfono
* Correo

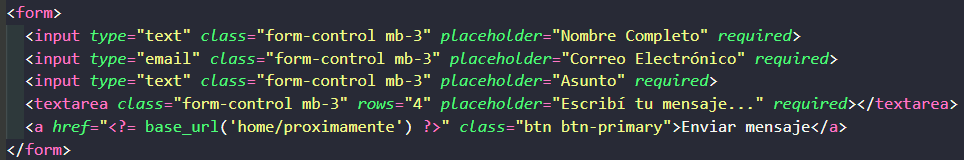
Cada <li> tiene un ícono relevante.

**📝 Columna 2: Formulario de Contacto**



📄 Otra columna del mismo tamaño para el formulario.

📨 Ícono de avión de papel, con estilo Bootstrap.

📬 Formulario con 4 campos:

1. Nombre completo
2. Correo electrónico
3. Asunto
4. Mensaje

🔵 El botón de envío está hecho con un enlace simulado (<a>) que va a la vista proximamente.

En una versión con backend, este botón sería <button type="submit"> y enviaría datos reales a un servidor.



🔚 Finaliza el bloque contenido.

**🧠 ¿Qué podés decir en la defensa?**

Esta página está pensada para que cualquier usuario pueda comunicarse con nosotros de manera clara. Usamos Bootstrap para dividir el contenido en dos columnas: una para los datos formales de contacto, y otra para un formulario. Aunque el formulario no está conectado a un backend real, la estructura está preparada para una integración futura. Visualmente usamos un diseño oscuro con tarjetas sombreadas, íconos temáticos y organización clara.

**📂 Archivo: terminos.php**

**🔹 Primeras líneas: conexión con la plantilla base**



* <?= $this->extend('layout') ?>: Esta línea indica que este archivo va a **heredar** el diseño base que está definido en layout.php. Es decir, no repite la cabecera, el <body>, el menú ni el footer.
* <?= $this->section('contenido') ?>: Marca el inicio de la **sección de contenido principal** de la vista. Esta parte se insertará dentro de <?= $this->renderSection('contenido') ?> en el layout.

**🔸 Sección visual con fondo épico y estilo pergamino**



* Esto abre una sección con la clase section-scroll, que en estilo.css le da:
  + **Fondo** de tipo pergamino (background-image).
  + **Overlay** oscuro encima para mejorar la lectura.
  + **Padding amplio**, para que el contenido no esté pegado al borde.
  + Un diseño visual coherente con el resto del sitio.

**🧾 Contenedor principal tipo "scroll de términos legales"**



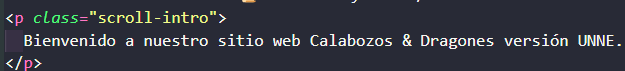
* scroll-container es una clase que en CSS:
  + Le da **efecto glassmorphism** (fondo translúcido y desenfocado).
  + Un ancho máximo (max-width: 1000px) para no ocupar toda la pantalla.
  + Sombra (box-shadow) y borde redondeado.

**📜 Título principal**

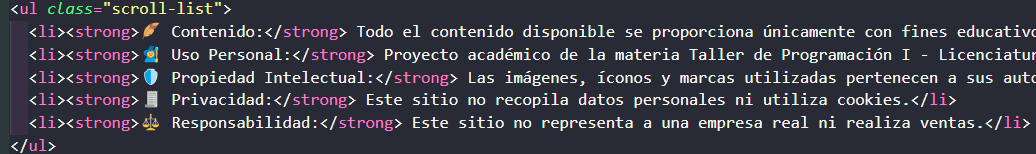


* Es el título de la sección, con una clase que lo hace estilizado, con tipografía épica y color claro.

**💬 Introducción al contenido**

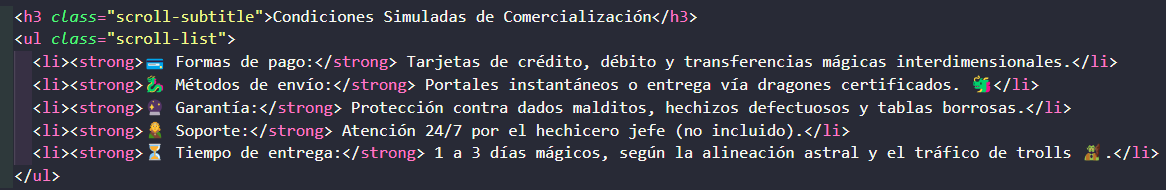


* Un párrafo introductorio donde se explica el **propósito educativo del sitio**, y que al navegarlo, el usuario acepta lo que se detalla más abajo.

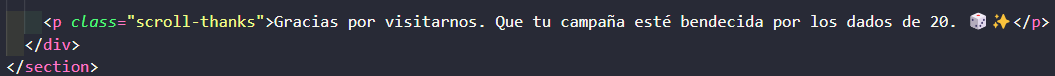
**🔐 Lista de términos legales (simulados)**

* Es una lista ordenada visualmente que contiene puntos claves como:
  + Contenido ficticio
  + Uso académico
  + Propiedad intelectual
  + Privacidad
  + Limitación de responsabilidad

🧠 **Esto sirve para que el visitante entienda que el proyecto no tiene fines comerciales ni reales, y que todo es una simulación académica.**

**📦 Condiciones simuladas de comercialización**

* Se detalla cómo *ficticiamente* funcionarían los métodos de pago, garantía, soporte y tiempos de entrega si esto fuera una tienda real.
* Se hace en tono humorístico y acorde a la temática (dragones, magia, trolls).

**🎲 Cierre del texto**

* Frase final que mantiene el tono rolero/épico, cierra de forma simpática y temática.

**🔚 Fin del bloque de contenido**



* Cierra el bloque iniciado con section('contenido').

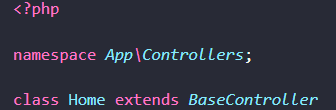
**🧠 ¿Cómo lo podés explicar oralmente?**

Esta página simula un aviso legal profesional, pero adaptado al universo de Dungeons & Dragons. Usamos estilos visuales que evocan un pergamino mágico, manteniendo coherencia estética con el resto del sitio. Aunque no tiene validez legal, ayuda a aclarar que esto es un proyecto educativo, no una tienda real. Se usa HTML para estructurar, CSS para darle estilo visual, y PHP para integrarlo con CodeIgniter. Esta página también demuestra cómo incluimos humor temático dentro de una estructura seria.

**🧱 Estructura de directorios del proyecto**



**📂 Archivo: Home.php**

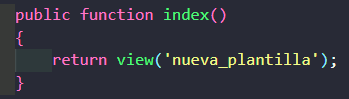


* Se define la clase Home, que extiende de BaseController.
* Esto significa que hereda funciones básicas de CodeIgniter, como helpers, validación, etc.

**🧠 ¿Qué es un controlador?**

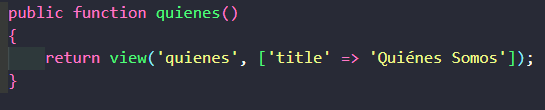
* Es el **encargado de recibir las solicitudes del usuario (desde el navegador)**, y decidir **qué vista mostrar**.
* En resumen: cuando alguien entra a una URL, CodeIgniter va a este archivo, busca qué función corresponde, y carga la vista.

**📌 Método 1: index()**



* Este método es el que se llama cuando el usuario entra a la raíz del sitio (ej: localhost/repositorio/public).
* view('nueva\_plantilla') le dice a CodeIgniter que cargue la vista app/Views/nueva\_plantilla.php.

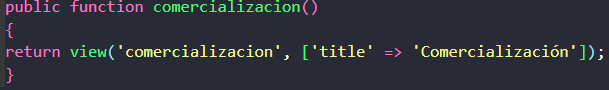
**📌 Método 2: quienes()**



Cuando se entra a /home/quienes, se carga la vista quienes.php.

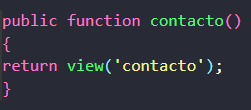
* Además, se le pasa un **título personalizado** (title), que luego puede usarse en el layout.

**📌 Método 3: comercializacion()**



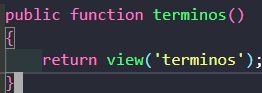
* Igual que el anterior, pero carga la vista comercializacion.php y le pasa el título 'Comercialización'.

**📌 Método 4: contacto()**



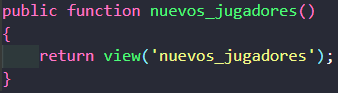
* Muestra la página de contacto, sin pasarle un título (aunque se podría agregar si quisieras).

**📌 Método 5: terminos()**



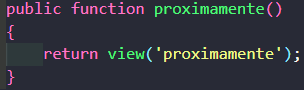
* Carga la vista terminos.php, que es la que contiene los Términos y Condiciones del sitio.

**📌 Método 6: nuevos\_jugadores()**



* Muestra una página especial para los nuevos usuarios que no conocen Dungeons & Dragons.

**📌 Método 7: proximamente()**



* Muestra una vista de “próximamente” para secciones que todavía no están listas.

**✅ ¿Qué aprendiste?**

* Cada método se asocia a una **ruta específica**, y cada uno **carga una vista distinta**.
* Si querés agregar una nueva página, tenés que:
  1. Crear la vista nueva (app/Views/nombre.php)
  2. Crear una nueva función en este archivo
  3. Declarar esa ruta en Routes.php