

S.I.G.P.D.

Programación Full Stack

VifraSoft

Rol	Apellido	Nombre	C.I	Email
Coordinador	Reyes	Franco	5.676.219-7	franco07sierra@gmail.com
Sub-Coordinador	Bittencourt	Luis	5.710.007-1	santiagobittencourt17@gmail.com
Integrante 1	Larrosa	Maria	5.633.663-5	victolarrosa13@gmail.com

Docente: Laporta, Emanuel.

**Fecha de
culminación
15/09/2025**

SEGUNDA ENTREGA

I.S.B.O.

3MI

Índice

Índice.....	2
Introducción al proyecto.....	4
Justificación Tecnológica.....	5
Análisis de PHP.....	5
Sistema gestor de base de datos.....	5
Framework seleccionado.....	6
Laravel como Framework.....	7
Relación con PHP.....	7
Herramientas para el control de versiones y colaboración.....	8
Recomendaciones de entornos de desarrollo.....	8
Modelo de datos.....	10
Diagrama Entidad-Relación.....	10
Pasaje a tablas.....	11
Participa (N:N entre Jugador y Partida).....	12
Lanza_dado (Jugador–Turno–Dado).....	12
Colocación (Jugador–Turno–Recinto–Consola).....	13
Normalización.....	13
Tablas en 3FN.....	14
Restricciones no estructurales.....	14
Base de datos en MySQL.....	15
Configuración del entorno de desarrollo.....	19
Proceso de instalación y configuración.....	19
Requisitos de Software.....	19
Configuración del IDE (Visual Studio Code) y extensiones útiles.....	20
Instrucciones para replicar el entorno.....	20
Git y GitHub.....	21
Comandos importantes de Git y GitHub.....	22
1. git init.....	22
2. git clone.....	22
3. git add.....	22
4. git commit.....	22
5. git status.....	23
6. git log.....	23
7. git branch.....	23
8. git checkout.....	23

9. git merge.....	23
10. git remote add origin.....	24
11. git push.....	24
12. git pull.....	24
13. git fetch.....	24
Principales ramas y su utilidad.....	24
1. main (o master en repositorios antiguos).....	24
2. develop.....	25
3. feature/* (rama de funcionalidades).....	25
4. hotfix/* (rama de correcciones rápidas).....	25
5. release/* (rama de versiones).....	25
Control de versiones y Repositorio.....	26
Creación de branch feature.....	28
Crear los remotes de los 3 branches principales.....	29
Creación de branch release.....	29
Git Flow y su utilidad en proyectos.....	30
Reglas del flujo Git Flow.....	30
Explicación del código Frontend.....	31
Página index.html.....	31
¿Qué es Bootstrap?.....	32
Cabecera y metadatos.....	32
Header y barra de navegación.....	33
Sección principal.....	34
Sección del botón “Jugar”.....	35
Sección de Información y Reglas del juego.....	36
Sección de Clasificación.....	39
Página Login.html.....	41
Header.....	41
Contenedor Principal.....	41
Input de Usuario.....	42
Campos del formulario en general.....	43
Página Register.html.....	44
Header.....	44
Contenedor principal.....	44
Campos del formulario.....	45
CSS styles.css.....	46
Script de JS explicación.....	47
Nuestra aplicación en laravel.....	50
Anexos.....	53

Introducción al proyecto

Nuestro proyecto se basa en crear una aplicación web en la cual su función será funcionar como tracker o aplicación de seguimiento sobre el juego de mesa **Draftosaurus**, deberá poder validar las reglas del juego y también que pueda hacer un conteo de puntos, ideal para ser un acompañante en el desarrollo de una partida normal del juego de mesa tradicional. En resumen nuestra aplicación web no suplantarán al juego original como una alternativa, solo servirá como un seguimiento de una partida con distintas herramientas para ayudar en el recorrido de una partida

En esta carpeta se documentará el proceso en el cual el equipo de VifraSoft trabajo en la diseñando frontend y la programación del backend y lógica de la aplicación web, siguiendo cada punteo y criterio de la mejor forma posible para cumplir con la satisfacción de los requerimientos de la materia Programacion Full Stack dentro de la letra del proyecto S.I.G.P.D.

Justificación Tecnológica

Análisis de PHP

PHP es un lenguaje de programación interpretado, orientado principalmente al desarrollo web del lado del servidor. Su mayor fortaleza radica en la capacidad de generar contenido dinámico, gestionar la comunicación con bases de datos y manejar la lógica necesaria para aplicaciones en línea.

Entre sus ventajas se destacan su amplia comunidad y documentación, lo que facilita el aprendizaje y la resolución de problemas, así como su madurez y estabilidad tras más de dos décadas de uso en proyectos de todo tipo. PHP ofrece además una integración nativa con servidores web como Apache y con bases de datos como MySQL y MariaDB, lo que lo convierte en una opción natural para entornos basados en LAMP. Otro aspecto relevante es su curva de aprendizaje accesible, que permite a desarrolladores con distintos niveles de experiencia trabajar con eficiencia.

Comparado con alternativas como Node.js, Python o Ruby, PHP mantiene la ventaja de ser altamente compatible con la mayoría de los servidores y de contar con un ecosistema consolidado, lo que lo convierte en una tecnología confiable y práctica para el desarrollo de aplicaciones web backend.

Sistema gestor de base de datos

Para el proyecto se seleccionó MySQL como sistema gestor de base de datos debido a su rendimiento, estabilidad y compatibilidad con el entorno LAMP. MySQL es una de las bases de datos más utilizadas en el mundo, lo que garantiza una amplia comunidad de soporte y una extensa documentación técnica.

Su integración nativa con PHP facilita la comunicación entre la aplicación y la base de datos, permitiendo realizar consultas y transacciones de manera eficiente. Además, MySQL es un software maduro, seguro y multiplataforma, lo que asegura un funcionamiento confiable tanto en entornos de desarrollo con XAMPP como en la implementación final sobre GNU/Linux.

La elección de MySQL se justifica frente a alternativas como PostgreSQL o SQLite por su equilibrio entre facilidad de uso, rendimiento y soporte en entornos web, convirtiéndose en una opción adecuada y optimizada para proyectos de gestión y seguimiento como el presente.

Framework seleccionado

Para el desarrollo del frontend se optó por utilizar Bootstrap, un framework ampliamente reconocido por su facilidad de uso y su capacidad para crear interfaces modernas y adaptables. Ofrece un sistema de diseño responsivo, lo que garantiza que la aplicación pueda visualizarse correctamente en distintos dispositivos sin necesidad de implementar configuraciones adicionales complejas.

Entre sus ventajas se destacan la rapidez en el desarrollo, gracias a la gran variedad de componentes predefinidos, y la consistencia en el diseño, asegurando una presentación uniforme en toda la aplicación. Asimismo, cuenta con una amplia documentación y comunidad activa, lo que facilita su aprendizaje y resolución de dudas.

La elección de Bootstrap se justifica frente a otras opciones por su ligereza, compatibilidad con PHP y facilidad de integración, características que lo convierten en una herramienta adecuada para proyectos web que requieren simplicidad y eficiencia en el diseño visual.

Laravel como Framework

Laravel es un **framework de desarrollo web en PHP**.

Un framework es como una “caja de herramientas” que trae ya listas un montón de funciones, librerías y estructuras que los programadores suelen usar mucho, como autenticación de usuarios, manejo de bases de datos, enrutamiento de URLs, plantillas de vistas, etc.

Laravel te da una base estructurada y organizada para crear aplicaciones web.

- Arquitectura **MVC (Modelo - Vista - Controlador)**: separa los datos, la lógica de negocio y la interfaz de usuario, lo que hace el código más ordenado y mantenible.
- **Eloquent ORM**: facilita trabajar con bases de datos usando clases y objetos en lugar de solo SQL.
- **Sistema de rutas**: te permite manejar las direcciones web fácilmente (/usuarios, /productos, etc.).
- **Blade**: motor de plantillas que simplifica la creación de vistas **HTML** dinámicas.
- **Seguridad y autenticación**: incluye sistemas listos para manejar logins, encriptación y protección contra ataques comunes.

Relación con PHP

Laravel está escrito en **PHP**, y es uno de los frameworks más populares de este lenguaje.

- PHP por sí solo es muy útil: podés crear cualquier cosa, pero puede volverse desordenado si el proyecto crece.
- Laravel es como una construcción prefabricada sobre PHP, usa la base del lenguaje pero añade estructura, buenas prácticas y utilidades que te facilitan la vida.

Herramientas para el control de versiones y colaboración

Para la gestión del código fuente y la colaboración entre los integrantes del equipo se seleccionó Git como sistema de control de versiones y GitHub como plataforma de alojamiento del repositorio.

Git permite mantener un historial detallado de cambios, crear ramas para el desarrollo de nuevas funcionalidades y facilitar la integración de aportes de cada miembro, reduciendo conflictos en el código. Por su parte, **GitHub** ofrece un entorno en línea que centraliza el repositorio, permite la colaboración en tiempo real y brinda herramientas adicionales como seguimiento de incidencias, gestión de ramas y revisiones de código.

La combinación de **Git** y **GitHub** garantiza un proceso de desarrollo organizado, seguro y colaborativo, aspectos fundamentales en un proyecto realizado en equipo.

Recomendaciones de entornos de desarrollo

Durante el proceso de construcción del sistema se empleará **XAMPP** en su versión 8.x como entorno de desarrollo. Esta herramienta integra de manera sencilla **Apache**, **PHP** y **MySQL**, lo que facilita la configuración inicial y permite realizar pruebas de manera rápida en un entorno local dentro de Windows.

Para la implementación final se recomienda la utilización de un servidor **LAMP** (**Linux**, **Apache**, **MySQL/MariaDB** y **PHP**), configurado manualmente en un sistema **GNU/Linux**. Este entorno garantiza una mayor estabilidad, seguridad y compatibilidad, además de representar un estándar en la puesta en producción de aplicaciones web.

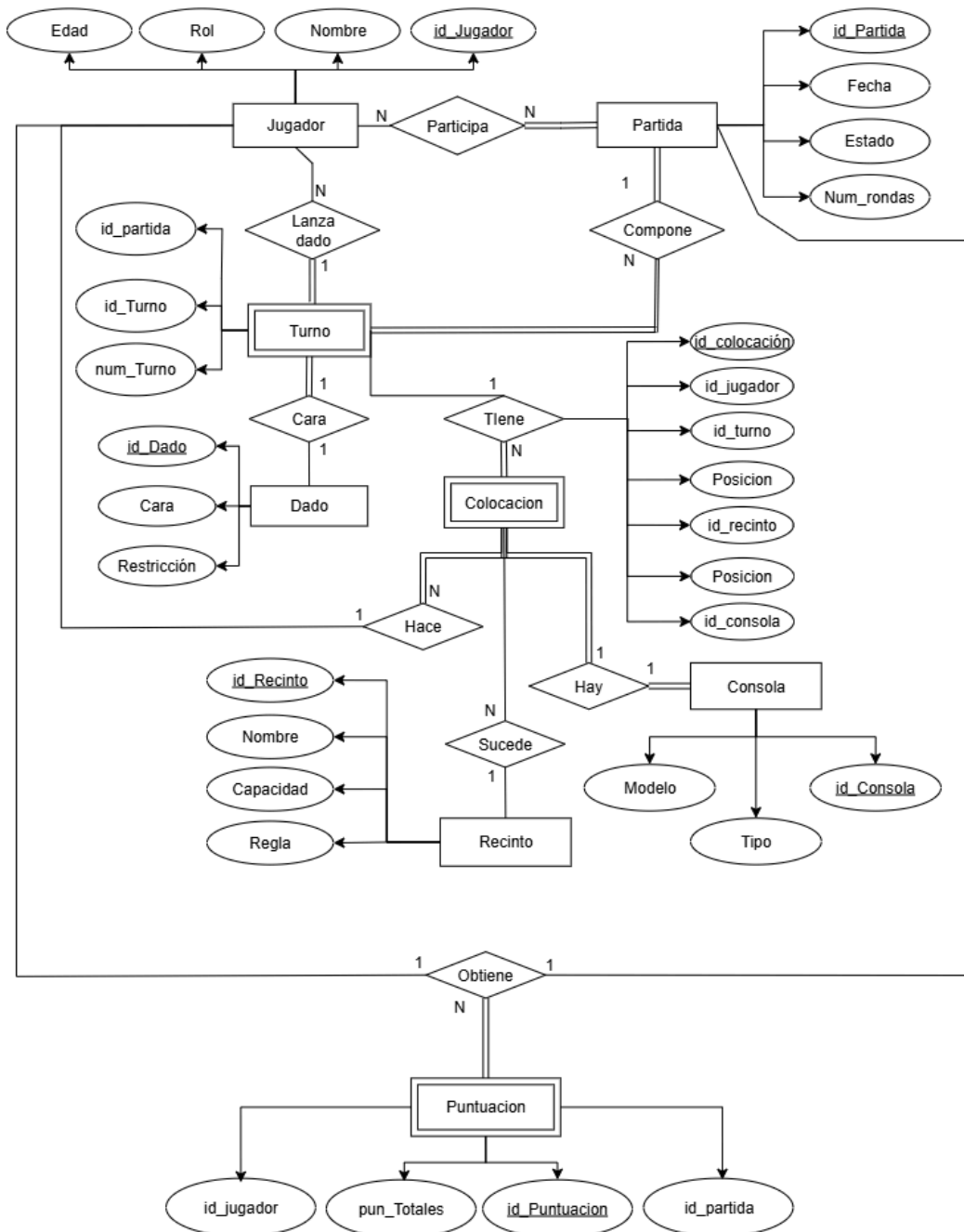
La elección de esta estrategia de entornos asegura un desarrollo ágil y flexible en la etapa inicial, junto con una implementación robusta y confiable en el despliegue definitivo del proyecto.

I.S.B.O.

3MI

Modelo de datos

Diagrama Entidad-Relación



Pasaje a tablas

PK: clave primaria

FK: clave foránea

Jugador

- ❖ id_jugador (PK)
- ❖ nombre
- ❖ edad
- ❖ rol

Partida

- ❖ id_partida (PK)
- ❖ fecha
- ❖ estado
- ❖ num_rondas

Turno

- ❖ id_turno (PK)
- ❖ id_partida (FK → Partida)
- ❖ num_turno
- ❖ cara

Dado

- ❖ id_dado (PK)
- ❖ cara
- ❖ restricción

Recinto

- ❖ id_recinto (PK)
- ❖ nombre
- ❖ capacidad
- ❖ regla

Consola

- ❖ id_consola (PK)
- ❖ modelo
- ❖ tipo

Puntuación

- ❖ id_puntuacion (PK)
- ❖ id_jugador (FK → Jugador)
- ❖ id_partida (FK → Partida)
- ❖ pun_totales

Participa (N:N entre Jugador y Partida)

- id_jugador (FK → Jugador)
- id_partida (FK → Partida)
- PK compuesta (id_jugador, id_partida)

Lanza_dado (Jugador–Turno–Dado)

- id_jugador (FK → Jugador)
- id_turno (FK → Turno)
- id_dado (FK → Dado)
- PK compuesta (id_jugador, id_turno)

Colocación (Jugador–Turno–Recinto–Consola)

- id_colocacion (PK)
- id_jugador (FK → Jugador)
- id_turno (FK → Turno)
- id_recinto (FK → Recinto)
- id_consola (FK → Consola)
- posición

Normalización

1FN

Cada atributo debe contener valores atómicos, sin grupos repetidos ni multivalorados.

En el pasaje del DER a tablas ya se habían definido atributos simples y no hay atributos compuestos ni multivaluados. Por lo tanto, el modelo ya estaba en 1FN.

Por lo tanto no se hacen cambios

2FN

Estar en 1FN y que todos los atributos no clave dependen de la clave completa (eliminar dependencias parciales).

Las únicas tablas con clave compuesta son Participa y Lanza_dado en ambas, los atributos dependen de la clave completa. No hay atributos que dependan solo de una parte de la clave.

Por lo tanto no hay cambios

3FN

estar en 2FN y eliminar dependencias transitivas (atributos no clave que dependen de otros atributos no clave).

Ninguna tabla presenta dependencias transitivas.

Por lo tanto no se hacen cambios

Tablas en 3FN

Jugador(id_jugador PK, nombre, edad, rol)

Partida(id_partida PK, fecha, estado, num_rondas)

Turno(id_turno PK, num_turno, id_partida FK → Partida(id_partida))

Dado(id_dado PK, cara, restriccion)

Recinto(id_recinto PK, nombre, capacidad, regla)

Consola(id_consola PK, modelo, tipo)

Puntuacion(id_puntuacion PK, id_jugador FK → Jugador(id_jugador), id_partida FK → Partida(id_partida), pun_totales)

Participa(id_jugador FK → Jugador(id_jugador), id_partida FK → Partida(id_partida), PK=(id_jugador,id_partida))

Lanza_dado(id_jugador FK → Jugador(id_jugador), id_turno FK → Turno(id_turno), id_dado FK → Dado(id_dado), PK=(id_jugador,id_turno))

Colocacion(id_colocacion PK, id_jugador FK → Jugador(id_jugador), id_turno FK → Turno(id_turno), id_recinto FK → Recinto(id_recinto), id_consola FK → Consola(id_consola), posicion)

Restricciones no estructurales

Capacidad de Recinto

Un recinto no puede superar su capacidad máxima de consolas.

Se implementa con lógica en backend.

Reglas del Recinto

Cada recinto define condiciones de colocación (ejemplo: solo consolas portátiles, retro, etc.).

Se implementa con validaciones en backend antes de insertar en Colocacion.

Límite de Rondas

Una partida no puede generar más turnos de los que permite num_rondas.

Control en aplicación o trigger en Turno.

Cálculo de Puntuación

pun_totales se debe calcular según consolas y reglas, no ingresar libremente.

Implementación: procedimiento almacenado o backend.

Estado de Partida

Solo se pueden registrar Colocaciones o Puntuaciones si la partida está en estado = “En curso”.

Implementación: validación antes de insertar.

Base de datos en MySQL

```
CREATE TABLE Jugador (  
    id_jugador INT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    edad INT,  
    rol VARCHAR(50)  
);
```

```
CREATE TABLE Partida (  
    id_partida INT PRIMARY KEY,  
    fecha DATE,  
    estado VARCHAR(20),  
    num_rondas INT  
);
```

```
CREATE TABLE Turno (  
    id_turno INT PRIMARY KEY,  
    num_turno INT,  
    id_partida INT,  
    FOREIGN KEY (id_partida) REFERENCES Partida(id_partida)  
);
```

```
CREATE TABLE Dado (  
    id_dado INT PRIMARY KEY,  
    cara VARCHAR(20),  
    restriccion VARCHAR(100)  
);
```

```
CREATE TABLE Recinto (  
    id_recinto INT PRIMARY KEY,  
    nombre VARCHAR(100),  
    capacidad INT,  
    regla VARCHAR(255)  
);
```

```
CREATE TABLE Consola (  
    id_consola INT PRIMARY KEY,  
    modelo VARCHAR(100),  
    tipo VARCHAR(50)  
);
```



```
CREATE TABLE Puntuacion (  
    id_puntuacion INT PRIMARY KEY,  
    id_jugador INT,  
    id_partida INT,  
    pun_totales INT,  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador),  
    FOREIGN KEY (id_partida) REFERENCES Partida(id_partida)  
);
```

```
CREATE TABLE Participa (  
    id_jugador INT,  
    id_partida INT,  
    PRIMARY KEY (id_jugador, id_partida),  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador),  
    FOREIGN KEY (id_partida) REFERENCES Partida(id_partida)  
);
```

```
CREATE TABLE Lanza_dado (  
    id_jugador INT,  
    id_turno INT,  
    id_dado INT,  
    PRIMARY KEY (id_jugador, id_turno),  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador),  
    FOREIGN KEY (id_turno) REFERENCES Turno(id_turno),  
    FOREIGN KEY (id_dado) REFERENCES Dado(id_dado)  
);
```

```
CREATE TABLE Colocacion (  
    id_colocacion INT PRIMARY KEY,  
    id_jugador INT,  
    id_turno INT,  
    id_recinto INT,  
    id_consola INT,  
    posicion INT,  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador),  
    FOREIGN KEY (id_turno) REFERENCES Turno(id_turno),  
    FOREIGN KEY (id_recinto) REFERENCES Recinto(id_recinto),  
    FOREIGN KEY (id_consola) REFERENCES Consola(id_consola)  
);
```

Configuración del entorno de desarrollo

Proceso de instalación y configuración

Para trabajar en el proyecto, necesitamos un entorno básico donde podamos desarrollar y probar la aplicación. Este entorno estará compuesto por un servidor web, un gestor de base de datos y las herramientas de programación necesarias.

En esta primera etapa, el proceso de instalación y configuración será documentado de manera general, ya que todavía no se ha comenzado con el desarrollo práctico. Más adelante, se detallarán los comandos y configuraciones específicas.

- En la máquina virtual con **Fedora Server 42** instalaremos un servidor web (Apache), el lenguaje de programación PHP y un sistema gestor de base de datos (MariaDB/MySQL).
- En los equipos de desarrollo utilizaremos **XAMPP** para realizar pruebas locales más simples antes de subir cambios al servidor.
- Se trabajará con un editor de código (Visual Studio Code) para la programación y organización del proyecto.

Requisitos de Software

Los principales programas que se usarán en el entorno de desarrollo son:

- **Servidor Apache:** encargado de alojar la aplicación y permitir que se acceda desde el navegador.
- **PHP 8.x:** lenguaje de programación con el que está hecho el backend del proyecto.
- **MySQL/MariaDB:** sistema de base de datos que guarda la información de las partidas, jugadores y reglas.
- **XAMPP 8.x:** paquete que incluye Apache, PHP y MySQL, usado en las computadoras de los desarrolladores para pruebas locales.

- **Fedora Server 42 en VirtualBox:** sistema operativo del servidor virtual.
- **Visual Studio Code:** editor de texto recomendado para escribir el código

Configuración del IDE (Visual Studio Code) y extensiones útiles

El editor recomendado es **Visual Studio Code**, ya que es gratuito, liviano y tiene muchas extensiones que facilitan el trabajo.

Algunas extensiones que se prevé utilizar son:

- **PHP Intelephense:** autocompletado y ayuda al programar en PHP.
- **PHP Debug:** para depuración básica.
- **GitLens:** para visualizar los cambios en el código y trabajar en equipo con GitHub.
- **SQLTools:** permite conectarse a la base de datos desde el editor.

Instrucciones para replicar el entorno

En esta primera entrega, las instrucciones se describen de manera general, sin entrar en detalles técnicos:

1. Instalar Fedora Server 42 en una máquina virtual (VirtualBox).
2. Instalar un servidor web (Apache), PHP y MariaDB en la máquina virtual.
3. Instalar XAMPP en las computadoras de los desarrolladores para hacer pruebas rápidas.
4. Instalar Visual Studio Code y agregar las extensiones recomendadas.
5. Crear un repositorio en GitHub para el trabajo en equipo y subir ahí el código del proyecto.

Con estos pasos básicos, cualquier integrante del grupo o docente podrá replicar el entorno de trabajo en su propia computadora.

Git y GitHub

En el desarrollo de nuestro proyecto utilizamos **Git** y **GitHub**, dos herramientas fundamentales para el trabajo colaborativo y la organización del código.

Git: Git es un sistema de control de versiones que nos permite llevar un registro detallado de todos los cambios que realizamos en los archivos del proyecto. Gracias a él, cada integrante del equipo puede trabajar de forma independiente en su propia copia del proyecto, realizar modificaciones y luego integrarlas al trabajo final sin riesgo de perder información o sobrescribir lo que otro compañero haya hecho. Una de sus mayores ventajas es que guarda un historial completo de los cambios, lo que facilita volver a versiones anteriores en caso de cometer errores, identificar quién realizó cada modificación y trabajar en diferentes ramas para separar versiones de prueba de la versión principal y estable del sistema.

GitHub: Por otro lado, GitHub es una plataforma en la nube que funciona como complemento de Git, ya que nos ofrece un espacio centralizado donde almacenar el proyecto y compartirlo con todo el equipo. Se puede considerar como una especie de red social para desarrolladores, en la que además de guardar el código de manera segura, se pueden gestionar tareas, documentar el avance del proyecto y coordinar el trabajo entre varios integrantes. Con GitHub logramos que nuestro proyecto esté siempre disponible y actualizado, lo que nos permite trabajar desde distintos lugares y dispositivos sin depender de copias locales. Otra de sus ventajas es que permite revisar y aprobar cambios antes de que se integren a la versión principal, asegurando que el trabajo sea más ordenado y con menos errores.

Comandos importantes de Git y GitHub

1. git init

Este comando se utiliza para **inicializar un repositorio Git** en una carpeta del proyecto. A partir de este momento, Git comenzará a registrar los cambios que se hagan en los archivos.

- Uso: `git init`

2. git clone

Sirve para **clonar (copiar)** un repositorio que ya existe en GitHub o en otra computadora, creando una copia completa con su historial.

- Uso: `git clone <url_del_repositorio>`

3. git add

Agrega archivos al área de preparación (staging area), es decir, indica qué cambios queremos guardar en el próximo commit.

- Uso: `git add archivo.txt` (para un archivo)
- Uso: `git add .` (para todos los archivos modificados)

4. git commit

Guarda los cambios en el repositorio local, creando un punto en el historial del proyecto. Se debe incluir un mensaje que describa el cambio realizado.

- Uso: `git commit -m "Descripción de los cambios"`

5. git status

Muestra el estado del repositorio: qué archivos han sido modificados, cuáles están listos para el commit y cuáles no.

- Uso: `git status`

6. git log

Muestra el historial de commits realizados, con detalles como autor, fecha y mensaje.

- Uso: `git log`

7. git branch

Permite crear, listar o eliminar ramas del proyecto. Las ramas son versiones paralelas donde se puede trabajar sin afectar la principal.

- Uso: `git branch` (ver ramas existentes)
- Uso: `git branch nombre_rama` (crear nueva rama)

8. git checkout

Se utiliza para **cambiar de rama** o moverse entre diferentes versiones del proyecto.

- Uso: `git checkout nombre_rama`

9. git merge

Combina los cambios de una rama con otra. Generalmente se usa para integrar la rama de trabajo con la rama principal (main o master).

- Uso: `git merge nombre_rama`

10. git remote add origin

Conecta el repositorio local con un repositorio remoto (como GitHub).

- Uso: `git remote add origin <url_del_repositorio>`

11. git push

Sirve para **subir los cambios** del repositorio local al repositorio remoto (GitHub).

- Uso: `git push origin main`

12. git pull

Descarga e integra los cambios que se hicieron en GitHub al repositorio local.

- Uso: `git pull origin main`

13. git fetch

Descarga los cambios desde GitHub, pero no los integra automáticamente. Se utiliza para revisar antes de actualizar la versión local.

- Uso: `git fetch`

Principales ramas y su utilidad

1. main (o master en repositorios antiguos)

-Es la **rama principal** del proyecto.



- Contiene la versión más estable y lista para entregar o desplegar.
- Generalmente solo se fusionan aquí los cambios ya probados y aprobados.

2. develop

- Es una rama que se usa mucho en equipos grandes.
- Funciona como rama de integración: aquí se juntan las funciones nuevas antes de pasarlas a **main**.
- Permite que el equipo trabaje en paralelo sin arriesgar la estabilidad de la rama principal.

3. feature/* (rama de funcionalidades)

- Se crean a partir de develop (o main, según la estrategia).
- Cada rama se usa para desarrollar **una nueva característica específica** del proyecto (por ejemplo, feature/login, feature/pantalla-principal).
- Cuando la funcionalidad está terminada y probada, se fusiona con develop.

4. hotfix/* (rama de correcciones rápidas)

- Se crean a partir de la rama **main**.
- Se utilizan para corregir errores urgentes en la versión en producción.
- Una vez arreglado el error, se fusiona tanto en main como en develop para mantener la coherencia.

5. release/* (rama de versiones)

- Se crean desde develop cuando el proyecto ya está casi listo para publicarse o entregarse.

I.S.B.O.

3MI



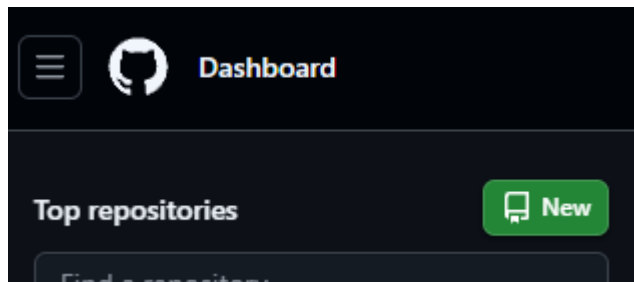
-Aquí se hacen pruebas finales, correcciones menores y ajustes antes de lanzar la versión estable.

-Luego se fusiona en main (para la entrega) y en develop (para mantener las mejoras).

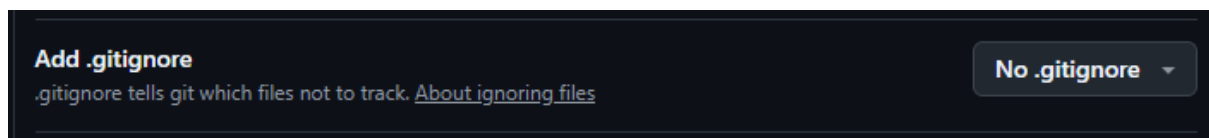
Control de versiones y Repositorio

los pasos que tuvimos que seguir para la creación y el cómo lo estructuramos para seguir las prácticas requeridas, iniciamos con la creación de una cuenta y la instalación de [GitHub Desktop](#).

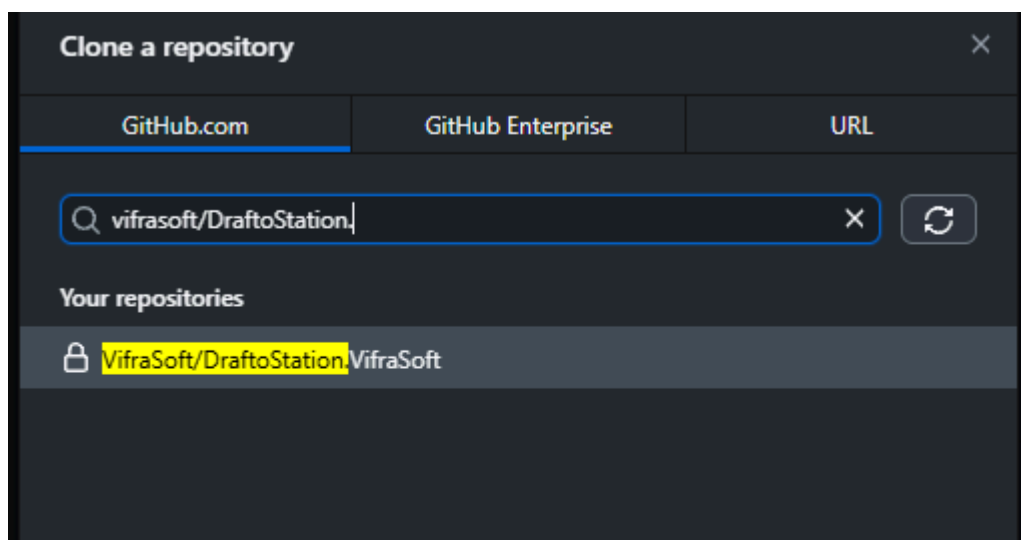
En la página oficial de github, en la parte izquierda superior tocaremos el botón de “New Repository”.



asegurándonos de no marcar “initialize with README”.

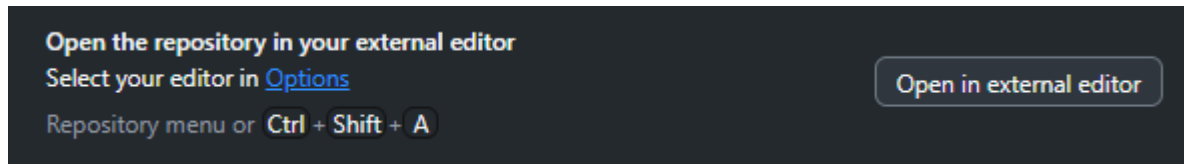


Una vez creado, abriremos la aplicación de **GitHub Desktop** y en el menú, entrar con la misma cuenta que utilizamos en la página web de **GitHub** y una vez hecho eso, tocar “Clone Repository” y seleccionar el proyecto.



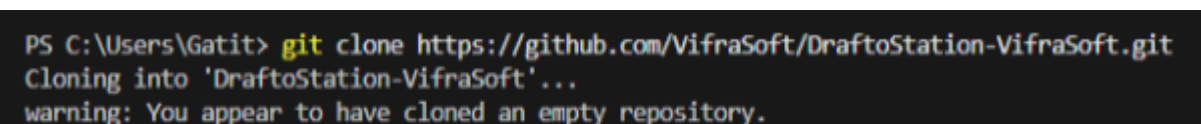


Seleccionaremos Visual Studio Code como aplicación predeterminada y abriremos el proyecto con eso.



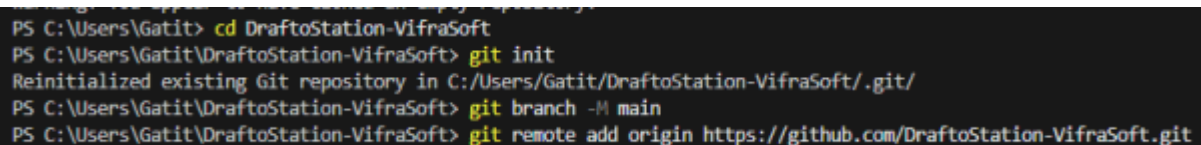
Inicializamos **Visual Studio Code** y entramos en la terminal (ctrl + ñ), y escribimos el siguiente comando:

git clone <https://github.com/VifraSoft/DraftoStation-VifraSoft.git>



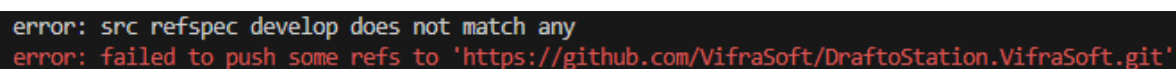
Nos dará advertencia de repositorio vacío, ya que todavía no creamos el **.gitignore**.

Posicionados en el proyecto, ejecutaremos los siguientes comandos:



git **branch -M main** nos permite renombrar el branch principal a “**main**” ya que en otras versiones puede llamarse **master** y confundir sintaxis.

Ahora, para crear la branch **develop**, primero tendremos que crear un archivo en main para que se cree la branch master, luego tendremos que hacer un **checkout -b develop** para que nos haga un **switch** a esta nueva branch, y en esta crear un **.md** de testeo ya que para pushear a origin debemos tener un archivo, sino nos mandara este error:



I.S.B.O.

3MI

Ejecutamos los comandos en la terminal

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> echo "# DraftoStation-VifraSoft" > README.md
PS C:\Users\Gatit\DraftoStation-VifraSoft> git add README.md
PS C:\Users\Gatit\DraftoStation-VifraSoft> git commit -m "Commit inicial con README"
[main (root-commit) 940eafb] Commit inicial con README
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
PS C:\Users\Gatit\DraftoStation-VifraSoft> git push -u origin main
Enumerating objects: 3, done.
```

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout -b develop
Switched to a new branch 'develop'
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git push -u origin develop
```

Creación de branch feature

Para crear nuestra branch **feature** la cual será siempre se hará un merge a **develop**, ejecutamos estos comandos en la terminal:

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout -b feature
Switched to a new branch 'feature'
```

En caso de querer mergear alguna cosa de **feature** a **develop**, primero nos posicionamos en **develop** y usaremos el comando:

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> git merge feature/la_feature
```



Crear los remotes de los 3 branches principales

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout feature
Switched to branch 'feature'
PS C:\Users\Gatit\DraftoStation-VifraSoft> git push -u origin feature
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:   https://github.com/VifraSoft/DraftoStation-VifraSoft/pull/new/feature
remote:
To https://github.com/VifraSoft/DraftoStation-VifraSoft.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git merge feature -m "merge(feature): poner los cambios de feature en develop"
Already up to date.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git push origin develop
Everything up-to-date
```

Simplemente tenemos que ir a la branch (comando git checkout) y hacer:

git push -u origin “branch-actual”

También hicimos un merge feature con develop.

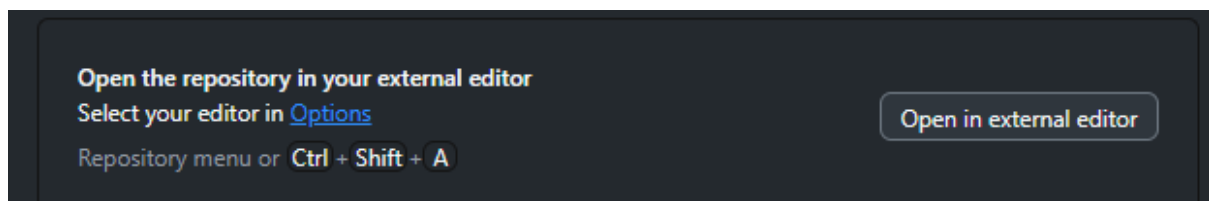
Creación de branch release

```
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout develop
Already on 'develop'
Your branch is up to date with 'origin/develop'.
PS C:\Users\Gatit\DraftoStation-VifraSoft> git checkout -b release/0.1.0
Switched to a new branch 'release/0.1.0'
```

Siempre que queramos traer un merge de release, siempre lo haremos posicionado en main.

git merge release/0.1.0 <- Merge: traemos el release a main.

Ahora siempre que queramos trabajar en algo, usaremos el botón de open in external editor en GitHub Desktop.



I.S.B.O.

3MI

Git Flow y su utilidad en proyectos

Git Flow es una metodología de organización en Git que define cómo usar las ramas de un repositorio para trabajar en equipo de forma ordenada. Su objetivo es mantener la rama principal siempre estable y permitir que el desarrollo avance sin conflictos.

En este flujo, la rama **main** contiene la versión final y estable, mientras que **develop** reúne las nuevas funciones antes de ser lanzadas. Las ramas **feature** se usan para crear funcionalidades específicas, las **release** para preparar entregas y las **hotfix** para corregir errores urgentes en producción.

De esta manera, Git Flow facilita la colaboración, reduce errores y acerca el trabajo académico a la forma en que se desarrollan proyectos profesionales.

Reglas del flujo Git Flow.

Todo feature deberá formar parte de **develop** en algún momento.

Nunca pushear a main directamente, solo se actualiza cuando hagas una **release**.

Cada **feature** tiene que ser de un tema específico, no amontonar código/archivos.

Usar commits claros (**feat**, **fix**, **docs**, etc.). Dejar una descripción breve y clara de los cambios hechos o nuevas implementaciones.

GitHub Desktop te ayuda a visualizar ramas y merges, VS Code para escribir código y commits.

Tipos de commit:

feat: nueva funcionalidad

fix: corrección de bug

docs: cambios solo en documentación

style: cambios de formato/código que no afectan lógica (indentación, espacios, comas...)

refactor: refactorización sin cambio funcional (mejora de código)

test: agregar/cambiar tests

Ejemplo:

commit “fix(ui): corregir bug en formulario de registro”.

Explicación del código Frontend

En esta parte se va a explicar el código frontend del proyecto VifraSoft, el cual consiste en una página web para presentar y gestionar el juego DraftoStation. Se analizarán en detalle los archivos HTML, CSS y JavaScript, describiendo qué contiene cada archivo y cuál es su función dentro del proyecto.

Se explicará en términos generales, pero también se destacarán los elementos más importantes de cada archivo, incluyendo:

- La **estructura y secciones de los archivos HTML**, como la página principal ([index.html](#)) con su barra de navegación, secciones de información del juego, reglas, clasificación y botones de interacción.
- Los **formularios de inicio de sesión y registro** ([Login.html](#) y [Register.html](#)) y cómo se utilizan las clases de Bootstrap para un diseño responsivo y consistente.
- Los **estilos personalizados** en [styles.css](#), que ajustan colores, tamaños de imágenes y otros detalles visuales para mantener la identidad del proyecto.
- El **archivo JavaScript** ([script.js](#)), que añade interactividad a la página, como la funcionalidad del botón de “Jugar” y la visualización de mensajes dinámicos.

El objetivo es ofrecer una visión clara y completa del frontend, explicando tanto la estructura del código como la experiencia que genera para el usuario, resaltando lo más importante para entender cómo funciona la página y cómo interactúan los distintos archivos entre sí.

Página [index.html](#)

La página [index.html](#) es la **página principal** del proyecto VifraSoft y funciona como **interfaz de presentación y acceso al juego DraftoStation**. Está diseñada con **HTML5**, utiliza **Bootstrap 5** para diseño responsivo y **CSS personalizado** para ajustar colores y tamaño de imágenes según la identidad visual del proyecto.

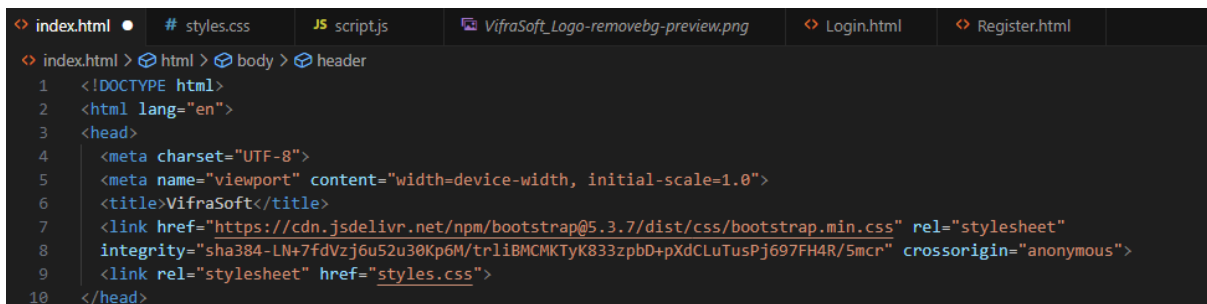
¿Qué es Bootstrap?

Bootstrap es un **framework de diseño web** que se usa para crear páginas de forma rápida, atractiva y que se adapten a distintos tamaños de pantalla como computadoras, tablets o celulares. Funciona a través de un conjunto de clases ya preparadas en **HTML, CSS y JavaScript**, que permiten agregar fácilmente menús, botones, formularios, tablas, colores y diseños responsivos sin tener que programar todo desde cero.

Su principal ventaja es que ahorra mucho tiempo al programador y asegura que el sitio tenga un diseño moderno y ordenado. Por ejemplo, con solo aplicar unas clases de Bootstrap se puede hacer que un botón tenga estilo profesional, que una página se vea bien en cualquier dispositivo y que los elementos se acomoden automáticamente al tamaño de la pantalla.

Cabecera y metadatos

En la sección `<head>` se definen los elementos esenciales del documento, los cuales no se van a mostrar al usuario, pero si estarán detrás de la página:



```
index.html > # styles.css JS script.js VifraSoft_Logo-removebg-preview.png Login.html Register.html
index.html > html > body > header
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>VifraSoft</title>
7   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css" rel="stylesheet"
8     integrity="sha384-LN+7fdVzj6u52u30Kp6M/trliBMCMTyK833zpbD+pXdcLuTusPj697FH4R/5mcr" crossorigin="anonymous">
9   <link rel="stylesheet" href="styles.css">
10 </head>
```

El meta de **charset="UTF-8"** asegura que todos los caracteres, incluidos acentos y símbolos, se muestren correctamente.

El meta de **viewport** permite que la página sea responsiva, ajustando el contenido al tamaño de la pantalla.

Luego utilizamos un link que nos permite conectarnos con **Bootstrap** y se importa desde un CDN para aprovechar sus componentes y clases responsivas.

Por último el link de **styles.css** conecta al archivo CSS que hay en la carpeta, que contiene estilos personalizados como **.main-image** y **.white-text** para controlar ancho de imágenes y



colores de texto.

Header y barra de navegación

El `<header>` contiene la **barra de navegación** con el logo del proyecto, enlaces a las secciones de la página y botones de **Iniciar sesión** y **Registrarse**.

La barra es **responsive**, lo que significa que en dispositivos pequeños se colapsa en un **menú desplegable**.

Los enlaces permiten navegar por las secciones internas: Información, Jugar, Clasificación y más.

```
<header>
<div class="container-fluid bg-dark">
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <!-- Logo o título -->
      
      <!-- Botón para menú colapsable en móviles -->
      <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#menu" aria-controls="menu" aria-expanded="false"
        <span class="navbar-toggler-icon"></span>
      </button>

      <!-- Links del menú -->
      <div class="collapse navbar-collapse" id="menu">
        <ul class="navbar-nav ms-auto pe-3">
          <li class="nav-item">
            <a class="nav-link active" aria-current="page" href="#Informacion">Informacion</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#Jugar">Jugar</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#Clasificación">Clasificación</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Jugadores</a>
          </li>
          <li class="nav-item">
            <a class="nav-link" href="#">Acerca de</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>
</div>
```

I.S.B.O.

3MI

```

42     </li>
43 </ul>
44
45 <!-- Botones a la derecha -->
46 <div class="d-flex">
47   <a href="Login.html" target="_blank" class="btn btn-outline-light me-2">Iniciar sesión</a>
48   <a href="Register.html" target="_blank" class="btn btn-primary">Registrarse</a>
49 </div>
50
51 </div>
52 </div>
53 </nav>
54 </div>
55 </header>

```

La barra de navegación de la página utiliza Bootstrap para ser responsiva y visualmente atractiva. El contenedor `container-fluid` hace que ocupe todo el ancho de la pantalla, mientras que `bg-dark` le da un fondo oscuro. La clase `navbar navbar-expand-lg navbar-dark bg-dark` define la barra como un componente de navegación que cambia el color del texto a blanco (`navbar-dark`) y se adapta en pantallas pequeñas usando un menú colapsable (`navbar-expand-lg`).

Dentro del navbar, el logo se coloca con `navbar-brand`, que identifica la marca, y en móviles se incluye un botón colapsable (`navbar-toggler`) con el ícono de hamburguesa (las barritas) (`navbar-toggler-icon`) que permite abrir y cerrar el menú. Los enlaces de navegación se agrupan con `navbar-nav`, y `ms-auto` empuja los elementos hacia la derecha, mientras que `nav-link active` resalta el enlace activo. Finalmente, los botones de Iniciar sesión y Registrarse se organizan con `d-flex` para un layout horizontal; `btn btn-outline-light` crea un botón con borde blanco y fondo transparente, `btn btn-primary` un botón azul sólido, y `me-2` separa los botones entre sí con margen a la derecha.

Sección principal

El contenedor principal de la página está diseñado para presentar la información más relevante del juego **DraftoStation** y se organiza usando el **sistema de grillas de Bootstrap** para que sea responsivo y se vea bien en cualquier dispositivo.

```
<!-- Contenedor principal -->
<div class="container mt-5">
  <div class="row">
    <div class="col-12 col-md-6 col-lg-6 p-5">
      <h1>DraftoStation</h1>
      <p>DraftoStation es un juego de colección y draft donde los jugadores seleccionan consolas de videojuegos para colocarlas en distintos re
      El objetivo es puntuar lo máximo posible utilizando la estrategia correcta para cada espacio.</p>
    </div>
    <div class="col-6 py-5 px-1">
      
    </div>
  </div>
</div>
```

Se utiliza un `<div class="container mt-5">` que centraliza el contenido y aplica un margen superior (`mt-5`) para separar la sección del header. Dentro de este contenedor, los elementos se organizan en una **fila (row)** con dos columnas principales: una columna (`col-12 col-md-6 col-lg-6 p-5`) que muestra la **descripción del juego** con título y texto explicativo, y otra columna (`col-6 py-5 px-1`) que contiene la **imagen principal**, que se adapta automáticamente al tamaño del contenedor gracias a la clase `img-fluid` y se puede ajustar con la clase personalizada `.main-image`.

Esta estructura permite que en **pantallas pequeñas** la columna de texto ocupe todo el ancho, mientras que en tablets y computadoras de escritorio se muestre junto a la imagen de manera equilibrada.

Bootstrap asegura que la disposición sea **responsiva**, mientras que el padding (`p-5`, `py-5`, `px-1`) proporciona espacio interno para que el contenido no se vea pegado a los bordes. Esta sección cumple la función de **introducir el juego** de manera visual y textual, captando la atención del usuario y guiándolo hacia la sección de acción, como entrar a jugar o explorar más información.

Sección del botón “Jugar”

Esta sección permite al usuario **entrar a una partida**. Se encuentra dentro de un contenedor `container mt-5 bg-dark py-5` que centra los elementos, aplica un fondo oscuro y espacio vertical. El título está centrado (`text-center`) y en blanco (`.white-text`) para destacarlo sobre el fondo.

```

69  <!-- Contenedor botón de Jugar -->
70
71  <div id="Jugar" class="container mt-5 bg-dark py-5">
72    <div class="row">
73      <h2 class="text-center mt-3 my-5 white-text">Entrar a una partida</h2>
74      <div class="d-grid gap-2 col-6 mx-auto">
75        <button class="btn btn-success" type="button">Jugar</button>
76      </div>
77      <!-- ALERTA DE BOOTSTRAP (invisible al inicio) -->
78      <div id="mensaje" class="alert alert-success mt-3 d-none" role="alert">
79        ¡La partida comenzó!
80      </div>
81    </div>
82  </div>
83

```

El botón se coloca en un `d-grid gap-2 col-6 mx-auto`, que lo centra horizontalmente y le da separación, y utiliza `btn btn-success` para resaltar la acción principal. Debajo se incluye un **mensaje de alerta** (`alert alert-success d-none`) que se muestra con JavaScript al iniciar la partida, dando feedback al usuario.

Sección de Información y Reglas del juego

La sección de Información y Reglas está diseñada para mostrar de manera clara y ordenada todos los detalles del juego **DraftoStation**, utilizando los componentes **cards de Bootstrap**.

```

84  <!-- Información del juego -->
85  <div class="card shadow-lg border-0 mb-4">
86    <div class="card-header bg-primary text-white">
87      <h4 class="mb-0">Información del juego</h4>
88    </div>
89    <div class="card-body">
90      <p><strong>Jugadores:</strong> 2 a 5</p>
91      <p><strong>Duración:</strong> 15-20 minutos</p>
92      <p><strong>Edad recomendada:</strong> 10+</p>
93
94      <h5 class="mt-4">Componentes:</h5>
95      <ul class="list-group list-group-flush">
96        <li class="list-group-item">Bolsita con consolas de distintos tipos.</li>
97        <li class="list-group-item">Tableros individuales con recintos.</li>
98        <li class="list-group-item">Marcadores de puntuación.</li>
99        <li class="list-group-item">Una consola Atari, que cumple el rol del T-Rex en Draftosaurus.</li>
100      </ul>
101    </div>
102  </div>
103
104  <!-- Reglas del juego -->
105  <div class="card shadow-lg border-0 mb-4">
106    <div class="card-header bg-success text-white">
107      <h4 class="mb-0">Reglas del juego</h4>
108    </div>
109    <div class="card-body">
110      <p>En cada ronda, los jugadores eligen una consola de las que tienen en mano y pasan el resto al jugador siguiente. Cada consola se debe
111

```

Cada card funciona como un contenedor independiente con sombra (`shadow-lg` o `shadow-sm`), bordes opcionales (`border-0`) y separación entre cards (`mb-3` o `mb-4`) para que la presentación sea atractiva y fácil de leer.

La primera card presenta la información general del juego, incluyendo número de jugadores, duración, edad recomendada y componentes. Su encabezado utiliza `card-header bg-primary text-white` para resaltar el título con fondo azul y texto blanco. Dentro del cuerpo de la card (`card-body`) se muestran listas de elementos con la clase `list-group list-group-flush`, que permite que cada item tenga separación y se vea organizado sin bordes adicionales.

```

105 <!-- Reglas del juego -->
106 <div class="card shadow-lg border-0 mb-4">
107   <div class="card-header bg-success text-white">
108     <h4 class="mb-0">Reglas del juego</h4>
109   </div>
110   <div class="card-body">
111     <p>En cada ronda, los jugadores eligen una consola de las que tienen en mano y pasan el resto al jugador siguiente. Cada consola se debe
112     Al final de la partida se suman los puntos de todos los recintos, más los de la Atari (T-Rex) si corresponde.</p>
113   </div>
114 </div>
115
116 <!-- Circuito Uniforme -->

```

A continuación, se muestran las reglas de cada recinto del juego en cards individuales. Cada card tiene un título (`card-title`) y un texto explicativo que detalla cómo se deben colocar las consolas y cómo se puntúa. Algunas reglas incluyen listas (`ul`) para enumerar pasos o condiciones específicas, y los colores de encabezado (como `bg-success` o `bg-danger`) ayudan a diferenciar visualmente secciones importantes, como las reglas generales o el fin del juego.

```

115
116 <!-- Circuito Uniforme -->
117 <div class="card shadow-sm border-0 mb-3">
118   <div class="card-body">
119     <h5 class="card-title">Circuito Uniforme</h5>
120     <p>Aquí se premia la constancia tecnológica.</p>
121     <ul>
122       <li>Solo puedes colocar consolas del mismo tipo.</li>
123       <li>Cuantas más consolas iguales logres en este recinto, mayor será la puntuación.</li>
124       <li>La escala de puntos es progresiva: 2, 4, 8, 12, 18, 24.</li>
125     </ul>
126   </div>
127 </div>
128
129 <!-- Triple A -->
130 <div class="card shadow-sm border-0 mb-3">
131   <div class="card-body">
132     <h5 class="card-title">Triple A</h5>
133     <p>El escaparate de los grandes éxitos.</p>
134     <ul>
135       <li>Coloca aquí consolas del mismo tipo.</li>
136       <li>Si logras al menos tres iguales, ganas 7 puntos fijos.</li>
137       <li>No importa si colocas más de tres, la puntuación máxima siempre será 7.</li>
138     </ul>
139   </div>
140 </div>
141

```

```

142 <!-- El Cuartel Multijugador -->
143 <div class="card shadow-sm border-0 mb-3">
144   <div class="card-body">
145     <h5 class="card-title">El Cuartel Multijugador</h5>
146     <p>Donde las consolas se conectan entre sí.</p>
147     <ul>
148       <li>Cada consola colocada en este recinto vale 5 puntos de vida.</li>
149       <li>No importa el tipo, todas suman el mismo valor.</li>
150     </ul>
151   </div>
152 </div>
153
154 <!-- Console of the Year -->
155 <div class="card shadow-sm border-0 mb-3">
156   <div class="card-body">
157     <h5 class="card-title">Console of the Year</h5>
158     <p>El premio a la mejor consola del año.</p>
159     <ul>
160       <li>Solo se permite colocar una consola en este recinto.</li>
161       <li>Si esa consola no se repite en otro recinto, otorga 7 puntos extra.</li>
162       <li>Si se repite, este espacio queda anulado y no puntúa nada.</li>
163     </ul>
164   </div>
165 </div>

```

```

167 <!-- Píxeles Variados -->
168 <div class="card shadow-sm border-0 mb-3">
169   <div class="card-body">
170     <h5 class="card-title">Píxeles Variados</h5>
171     <p>El rincón de los coleccionistas.</p>
172     <ul>
173       <li>Cada consola debe ser de un tipo distinto.</li>
174       <li>Según la cantidad de consolas diferentes, se ganan los puntos: 1, 3, 6, 10, 15, 21.</li>
175     </ul>
176   </div>
177 </div>
178
179 <!-- La Edición Limitada -->
180 <div class="card shadow-sm border-0 mb-3">
181   <div class="card-body">
182     <h5 class="card-title">La Edición Limitada</h5>
183     <p>Un espacio exclusivo para verdaderas joyas.</p>
184     <ul>
185       <li>Este recinto solo puntúa si colocas exactamente dos consolas iguales.</li>
186       <li>Si cumples, otorga 7 puntos.</li>
187       <li>Si pones más o menos de dos, no suma nada.</li>
188     </ul>
189   </div>
190 </div>

```

```

192 <!-- La Atari (T-Rex) -->
193 <div class="card shadow-sm border-0 mb-3">
194   <div class="card-body">
195     <h5 class="card-title">La Atari (T-Rex)</h5>
196     <p>La Atari es una consola especial, que funciona igual que el T-Rex en Draftosaurus.</p>
197     <ul>
198       <li>No puede colocarse en ningún recinto.</li>
199       <li>En su lugar, se coloca en un espacio libre junto al tablero.</li>
200       <li>No cuenta para las reglas de los recintos, pero sí otorga 1 punto fijo por estar en juego.</li>
201     </ul>
202   </div>
203 </div>
204
205 <!-- Fin del juego -->
206 <div class="card shadow-lg border-0">
207   <div class="card-header bg-danger text-white">
208     <h4 class="mb-0">Fin del juego</h4>
209   </div>
210   <div class="card-body">
211     <p>La partida termina cuando todas las consolas fueron colocadas.<br>
212     Se suman los puntos de cada recinto, más la puntuación de la Atari.<br>
213     El jugador con más puntos se convierte en el <strong>Maestro de DraftoStation</strong>.</p>
214   </div>
215 </div>
216

```

Esta sección permite que el usuario comprenda fácilmente cómo se juega, cuáles son los objetivos de cada recinto y cómo se determina el ganador. Al combinar cards, listas y colores, se logra un diseño visualmente atractivo, organizado y responsivo, gracias a las clases de Bootstrap, mientras que los estilos personalizados aseguran coherencia con la identidad visual del proyecto.

Sección de Clasificación

La sección de Clasificación muestra una **tabla con los jugadores y sus posiciones**. Está dentro de un contenedor centrado (`container text-center my-5`) que separa la sección del resto de la página.

```

218 <div class="container text-center my-5">
219   <h3>Clasificación</h3>
220   <table class="table table-success table-striped" id="Clasificación">
221     <thead>
222       <tr>
223         <th scope="col">#</th>
224         <th scope="col">First</th>
225         <th scope="col">Last</th>
226         <th scope="col">Handle</th>
227       </tr>
228     </thead>
229     <tbody>
230       <tr>
231         <th scope="row">1</th>
232         <td>Mark</td>
233         <td>Otto</td>
234         <td>@mdo</td>
235       </tr>
236       <tr>
237         <th scope="row">2</th>
238         <td>Jacob</td>
239         <td>Thornton</td>
240         <td>@fat</td>
241       </tr>
242       <tr>
243         <th scope="row">3</th>
244         <td>John</td>
245         <td>Doe</td>
246         <td>@social</td>
247       </tr>
248     </tbody>
249   </table>
250 </div>
251 </div>
252 </div>
253 </div>
254 </div>
255

```

La tabla utiliza clases de Bootstrap como `table`, `table-success` y `table-striped` para darle estilo, color de fondo verde suave y filas alternadas, lo que facilita la lectura. La estructura incluye encabezados (`thead`) y cuerpo (`tbody`) para organizar los datos de cada jugador, como nombre, apellido y usuario.

Esta sección permite visualizar de manera clara y ordenada la posición de los jugadores, manteniendo coherencia visual con el resto de la página gracias a Bootstrap y los estilos generales del proyecto.

Página Login.html

La página `Login.html` es la interfaz para que los usuarios puedan **iniciar sesión** en el proyecto. Está diseñada con **HTML5** y utiliza **Bootstrap 5** para mantener un diseño limpio, responsivo y consistente con el resto del proyecto.

Header

```
10 <body>
11   <!--Boton en el header que manda hacia atras-->
12   <header class="p-3 bg-light">
13     <a href="prueba.html" class="btn btn-sm btn-outline-dark">← Atrás</a>
14   </header>
15
```

En la parte superior se encuentra un `<header>` con **fondo claro (bg-light)** y padding (`p-3`), que incluye un **botón de retroceso** (`btn btn-sm btn-outline-dark`) para regresar a la página anterior. Este botón es pequeño, con borde oscuro y fondo transparente, manteniendo el estilo simple y funcional.

Contenedor Principal

El contenido principal está dentro de un `container text-center mt-5`, que centra los elementos y agrega un **margen superior** para separarlo del header. Se muestra un **título destacado** (`h1 fw-bold`) que indica “Iniciar Sesión”.

```

16 <div class="container text-center mt-5">
17 <div class="row">
18 <h1 class="fw-bold">Iniciar Sesión</h1>
19 </div>
20
21 <div class="input-group flex-nowrap"> <!-- flex-nowrap = lo que hace es que si se achica el tamaño no se rompe el diseño -->
22
23 <span class="input-group-text id="addon-wrapping">@</span> <!-- Span es un contenedor en línea para mostrar un texto o ícono-->
24 <!--input-group-text Es una clase especial de Bootstrap que se usa dentro de un input-group, le da estilo de caja gris clara-->
25 <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="addon-wrapping">
26 <!--form-control = hace que el input ocupe todo el ancho disponible dentro del input gorup y tenga estilo de formulario de Bootstrap
27 <!--placeholder = Muestra el texto de ayuda "Username"-->
28
29 </div>
30 <!--Gmail-->
31 <form> <!--Form = crear un formulario donde el usuario pone sus datos y despues se envian al servidor-->
32 <!--Password-->
33 <div class="mb-3">
34 <label for="exampleInputPassword1" class="form-label">Contraseña</label>
35 <input type="password" class="form-control" id="exampleInputPassword1">
36 </div>
37 <!--Botón-->
38 <div class="mb-3 form-check">
39 <input type="checkbox" class="form-check-input" id="exampleCheck1">
40 <label class="form-check-label" for="exampleCheck1">Confirmar</label>
41 </div>
42 <button type="submit" class="btn btn-primary">Login</button>
43 </form>
44
45 </div>

```

Input de Usuario

```

21 <div class="input-group flex-nowrap"> <!-- flex-nowrap = lo que hace es que si se achica el tamaño no se rompe el diseño -->
22
23 <span class="input-group-text id="addon-wrapping">@</span> <!-- Span es un contenedor en línea para mostrar un texto o ícono-->
24 <!--input-group-text Es una clase especial de Bootstrap que se usa dentro de un input-group, le da estilo de caja gris clara-->
25 <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="addon-wrapping">
26 <!--form-control = hace que el input ocupe todo el ancho disponible dentro del input gorup y tenga estilo de formulario de Bootstrap
27 <!--placeholder = Muestra el texto de ayuda "Username"-->
28
29 </div>

```

Esta sección corresponde al campo de usuario dentro del formulario y utiliza las clases de Bootstrap para formularios avanzados:

1. **input-group:**

Es un contenedor que permite agrupar un campo de entrada (**input**) con elementos adicionales, como texto, íconos o botones, dentro de una sola fila visual. Esto crea un diseño uniforme y estilizado, propio de los formularios de Bootstrap.

2. **flex-nowrap:**

Es una clase de utilidad de Bootstrap que evita que los elementos dentro del grupo se “rompan” o salgan de la fila cuando la pantalla se hace más pequeña. Así, el icono @ y el campo de texto siempre permanecen en la misma línea.

3. **@:**

-`input-group-text` es una clase especial de Bootstrap que da estilo a un texto o ícono dentro del `input-group`, mostrando un recuadro gris claro alrededor del contenido.

-Este `` sirve como un elemento visual complementario, en este caso mostrando el símbolo `@` al inicio del campo de usuario, indicando que allí se debe ingresar un nombre de usuario.

-El `id="addon-wrapping"` se usa para asociar accesibilidad con el input mediante `aria-describedby`.

4. `<input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="addon-wrapping">`:

-`form-control`: hace que el campo ocupe todo el ancho disponible dentro del `input-group` y le aplica el estilo de Bootstrap para formularios.

-`placeholder="Username"`: muestra un texto de ayuda dentro del input mientras el usuario no ha escrito nada.

-`aria-label` y `aria-describedby`: mejoran la accesibilidad, indicando a los lectores de pantalla cuál es la función del campo y relacionándolo con el texto `@`.

Campos del formulario en general

1. Usuario:

Se usa un `input-group flex-nowrap` para agrupar un ícono o texto (`input-group-text`) con el campo de entrada (`form-control`). Esto asegura que el diseño no se rompa en pantallas pequeñas. El campo incluye un `placeholder` que indica que allí se debe ingresar el nombre de usuario.

2. Contraseña:

Se crea un `div` con `mb-3` (margen inferior) que contiene un `label` y un `input` de tipo `password` con clase `form-control`, para un estilo uniforme con Bootstrap.



3. Checkbox:

Un **form-check** con un **input** tipo checkbox y su etiqueta permite confirmar alguna opción, como recordar sesión o aceptar términos.

4. Botón de envío:

El botón de tipo **submit** (**btn btn-primary**) envía el formulario y tiene un **color azul sólido**, destacándose como acción principal.

Página Register.html

La página **Register.html** es la interfaz para que los usuarios puedan **crear una nueva cuenta** en el proyecto. Está diseñada con **HTML5** y **Bootstrap 5**, manteniendo un estilo limpio, centrado y consistente con el resto del proyecto.

Header

```
Register.html > html > body > div.container.text-center.mt-5 > div.input-group.flex-nowrap
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Registro</title>
7    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-LN+7fdVzj6u52u30Kp6M/
8
9  </head>
10 <body>
11   <!--Boton en el header que manda hacia atras-->
12   <header class="p-3 bg-light">
13     <a href="prueba.html" class="btn btn-sm btn-outline-dark">← Atrás</a>
14   </header>
15
```

En la parte superior se encuentra un **<header>** con fondo claro (**bg-light**) y padding (**p-3**), que incluye un **botón de retroceso** (**btn btn-sm btn-outline-dark**) para regresar a la página anterior.

Contenedor principal

El contenido principal está dentro de un **container text-center mt-5**, que centra los elementos y agrega un margen superior. Se muestra un **título destacado** (**h1 fw-bold**) indicando “Registrarse”.

I.S.B.O.

3MI

```

16 <div class="container text-center mt-5">
17 <div class="row">
18 <h1 class="fw-bold">Registrarse</h1>
19 </div>
20 <div class="input-group flex-nowrap"> <!-- flex-nowrap = lo que hace es que si se achica el tamaño no se rompe el diseño -->
21
22 <span class="input-group-text" id="addon-wrapping">@</span> <!-- Span es un contenedor en línea para mostrar un texto o ícono -->
23 <!--input-group-text Es una clase especial de Bootstrap que se usa dentro de un input-group, le da estilo de caja gris clara-->
24 <input type="text" class="form-control" placeholder="Username" aria-label="Username" aria-describedby="addon-wrapping">
25 <!--"form-control" = hace que el input ocupe todo el ancho disponible dentro del input group y tenga estilo de formulario de Bootstrap
26 <!--placeholder = Muestra el texto de ayuda "Username"-->
27
28 </div>
29 <!--Gmail-->
30 <form> <!--Form = crear un formulario donde el usuario pone sus datos y despues se envian al servidor-->
31 <div class="mb-3">
32 <label for="exampleInputEmail1" class="form-label">Correo electrónico</label>
33 <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp">
34 <div id="emailHelp" class="form-text">No compartimos tu correo electrónico con nadie más.</div>
35 </div>
36 <!--Password-->
37 <div class="mb-3">
38 <label for="exampleInputPassword1" class="form-label">Contraseña</label>
39 <input type="password" class="form-control" id="exampleInputPassword1">
40 </div>
41 <!--Botón-->
42 <div class="mb-3 form-check">
43 <input type="checkbox" class="form-check-input" id="exampleCheck1">
44 <label class="form-check-label" for="exampleCheck1">Confirmar</label>
45 </div>
46 <button type="submit" class="btn btn-primary">Hecho</button>
47 </form>
48
49 </div>

```

La página Register.html es muy similar en estructura a la página Login.html, por lo tanto vamos a resumir su explicación.

Campos del formulario

1. Usuario:

Se utiliza un `input-group flex-nowrap` con un `span` mostrando `@` al inicio y un `input` de tipo texto. Esto crea un campo estilizado, alineado y responsive donde el usuario ingresa su nombre de usuario.

2. Correo electrónico:

Un campo de tipo `email` con `form-control` y un `label` describe el contenido. Debajo se incluye un pequeño texto de ayuda (`form-text`) para informar que el correo no será compartido.

3. Contraseña:

Campo de tipo `password` con `form-control` y su respectivo `label`.

4. **Checkbox:**

Permite confirmar condiciones o aceptar términos mediante un `form-check`.

5. **Botón de envío:**

Botón `btn btn-primary` que envía el formulario y destaca como acción principal con color azul sólido.

CSS styles.css

El archivo `styles.css` contiene **estilos personalizados** que complementan los de Bootstrap, ajustando principalmente la apariencia de imágenes y texto en la página.

```
<> index.html ● # styles.css ✕ JS script.js
# styles.css > ...
1  /* Imagen principal */
2  .main-image {
3    width: 650px;
4    height: auto;
5  }
6
7  /* Texto blanco */
8  .white-text {
9    color: ■ #ffffff;
10 }
11
```

1. **.main-image:**

Esta clase se aplica a la **imagen principal** de la página (`ImagenPrincipal.png`).

`width: 650px;` fija el ancho de la imagen en 650 píxeles.

`height: auto;` mantiene la **proporción original** de la imagen, evitando que se distorsione al cambiar el ancho.

2. **.white-text:**

Esta clase se usa para los textos que deben mostrarse en **blanco**, especialmente sobre fondos



oscuros como el de la sección de Jugar.

`color: #ffffff;` define el color de la letra como blanco puro.

En resumen, este CSS **controla detalles visuales específicos**, asegurando que la imagen principal tenga un tamaño adecuado y que el texto blanco resalte sobre fondos oscuros, manteniendo la coherencia visual del proyecto.

Script de JS explicación

El archivo `script.js` cumple la función de dar interactividad a la página, enfocándose principalmente en el botón de jugar y el mensaje de alerta

Este pequeño script espera a que el DOM esté listo, busca el botón de “Jugar” y el div de mensaje, y cuando el usuario hace click en el botón muestra la alerta, cambia el texto del botón y lo desactiva para que no se pueda volver a pulsar.

A screenshot of a code editor with a dark theme. The top bar shows three tabs: 'index.html', 'styles.css', and 'script.js'. The 'script.js' tab is active. The code in the editor is as follows:

```
JS script.js > ...
1  document.addEventListener('DOMContentLoaded', () => {
2    const playButton = document.querySelector('#Jugar button');
3    const mensaje = document.getElementById('mensaje');
4
5    playButton.addEventListener('click', () => {
6      // Mostrar el alert
7      mensaje.classList.remove('d-none');
8      // Cambiar texto del botón
9      playButton.textContent = '¡Partida iniciada!';
10     playButton.disabled = true;
11   });
12 });
```

¿Qué es un DOM?

El DOM en JavaScript, conocido como Document Object Model, es la forma en que el navegador representa internamente una página web. Se puede imaginar como un árbol donde cada parte del documento HTML, como etiquetas, atributos o textos, se convierte en un nodo que puede ser manipulado. Gracias a esto, JavaScript puede acceder a los elementos de la página, modificarlos, eliminarlos o agregar nuevos, lo que permite transformar una página estática en una página dinámica e

I.S.B.O.

3MI

interactiva. El DOM actúa como un puente entre el HTML y JavaScript, facilitando que el código pueda comunicarse con lo que el usuario ve en el navegador.

Para trabajar con el DOM es común utilizar funciones como:

```
JS script.js > ...  
1 document.addEventListener('DOMContentLoaded', () => {
```

`document.addEventListener('DOMContentLoaded', ...)`, que sirven para asegurarse de que todo el contenido HTML esté completamente cargado antes de que el código JavaScript intente ejecutarse. Esto evita errores que ocurren cuando se busca un elemento que todavía no existe en el documento.

Dentro de esta función se declara la constante `playButton`, que selecciona con `document.querySelector('#Jugar button')`

```
document.addEventListener('DOMContentLoaded', () => {  
  const playButton = document.querySelector('#Jugar button');  
  const mensaje = document.getElementById('mensaje');
```

Dentro de este evento se utiliza `document.querySelector('#Jugar button')`, que permite seleccionar desde el DOM el botón ubicado dentro del contenedor con el id Jugar. Esta instrucción sirve para poder manipular ese botón en específico y agregarle comportamientos dinámicos.

También se utiliza `document.getElementById('mensaje')`, que selecciona directamente el elemento cuyo id es `mensaje`, en este caso un texto oculto que se mostrará más adelante.

```
5 playButton.addEventListener('click', () => {  
6   // Mostrar el alert  
7   mensaje.classList.remove('d-none');  
8   // Cambiar texto del botón  
9   playButton.textContent = '¡Partida iniciada!';  
10  playButton.disabled = true;  
11  });  
12  });
```

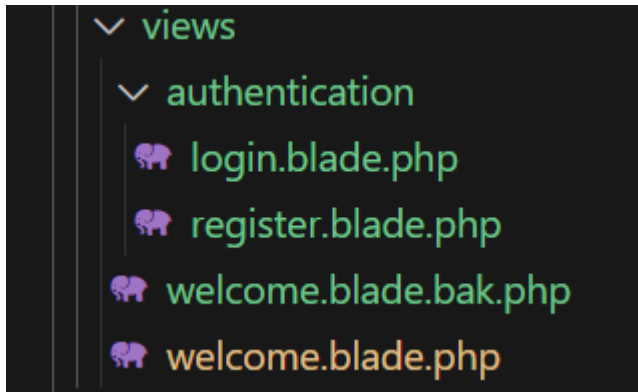

Una vez seleccionados estos elementos, se agrega otro escuchador de eventos con `playButton.addEventListener('click', ...)`, el cual está pendiente de que el usuario haga clic en el botón de jugar. Cuando ese evento ocurre, dentro de la función se ejecutan tres instrucciones importantes: se utiliza `mensaje.classList.remove('d-none')` para eliminar la clase que mantenía oculto el mensaje en la página, mostrando así el contenido; luego `playButton.textContent = '¡Partida iniciada!'` cambia el texto del botón para dar un feedback inmediato al usuario; finalmente `playButton.disabled = true` desactiva el botón para que no pueda volver a presionarse y evitar que el evento se repita.

De esta forma, el código en conjunto permite que la página pase de ser un documento estático a una aplicación interactiva, ya que responde a las acciones del usuario, modifica el contenido en tiempo real y controla la experiencia visual y funcional.

Nuestra aplicación en laravel

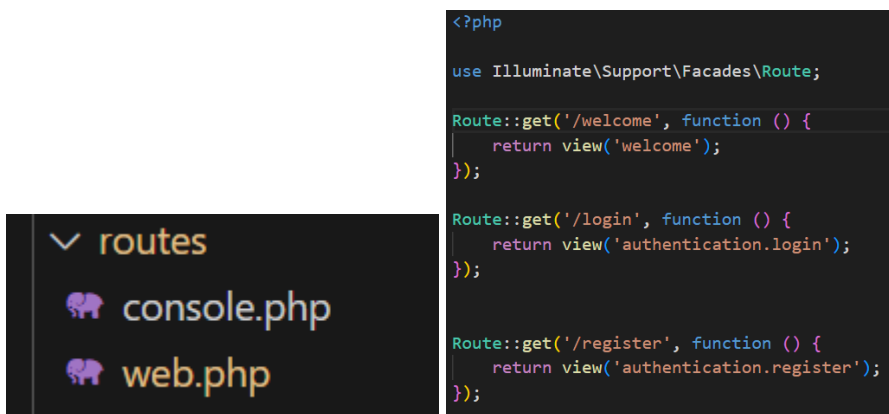
Mostraremos cómo subimos nuestras primeras pestañas a nuestro proyecto laravel

Primero subimos las pestañas al resources/view



la página principal la hemos llamado **welcome.blade.php** y el welcome.blade original lo dejamos como auxiliar para consultar código y abrimos un directorio nuevo llamado **authentication** para dejar las pestañas de inicio de sesión y registro para una mejor organización

Después especificamos las rutas de los archivos en routes/web.php para usarlas en la pestaña principal

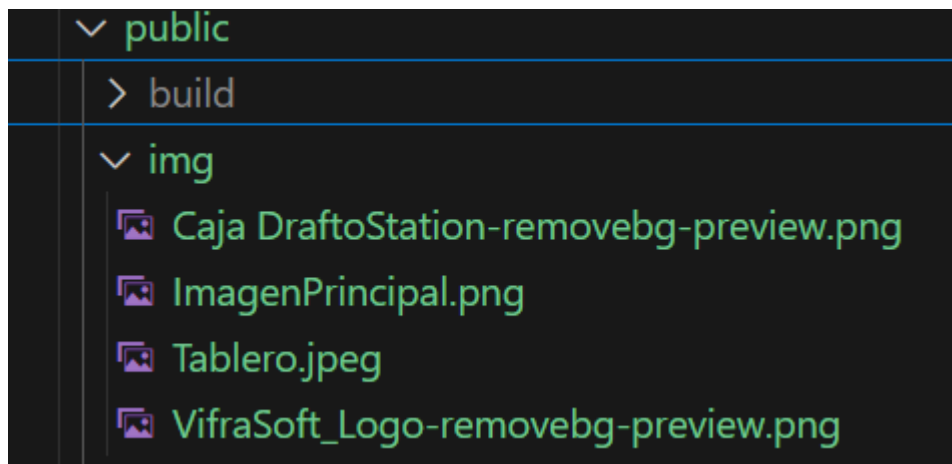




Lo siguiente que hicimos fue dejar las rutas en la pestaña para que al momento de apretar los botones lleven a la pestaña deseada

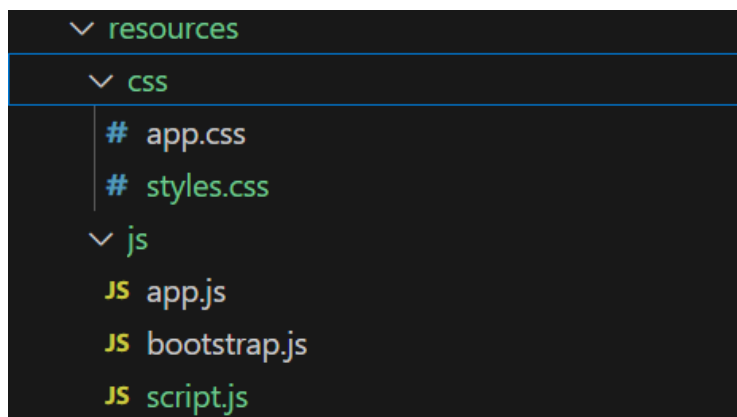
```
<div class="d-flex">
  <a href="/login" class="btn btn-outline-light me-2">Iniciar Sesión</a>
  <a href="/register" class="btn btn-primary">Registrarse</a>
</div>
```

Nuestro siguiente paso fue dejar las imágenes en public/img lo único que el directorio img lo creamos nosotros porque no estaba, lo siguiente que hicimos fue dejar la ruta en el código para que usara las imágenes



```
src="{{ asset('img/VifraSoft_Logo-removebg-preview.png') }}"
```

Nuestro siguiente paso fue cargar el archivo css y el archivo js dentro de **resources/css** para el archivo css y **resources/js** para el archivo js respectivamente



I.S.B.O.

3MI



los de color verde son los archivos que agregamos, los que ya estaban decidimos no tocarlos como manera de usarlos de auxiliar

Siguiente a subirlos tuvimos que hacer los enrutamientos de ambos

```
<link rel="stylesheet" href="{asset('css/styles.css')}}">
```

```
<script src="{ asset('js/scrip.js')}"></script>
```

Hasta aqui tenemos completado la implementación de nuestras pestañas hechas, para ver el resultado tendremos que ejecutar en simultáneo dentro de la terminal en VSC los dos comandos: **composer run dev** y **php artisan serve** y con la ip que nos de el segundo comando podremos ver la página web

```
PS C:\Users\Franco\Desktop\DraftoStation-VifraSoft\DraftoStation> php artisan serve
```

```
INFO Server running on [http://127.0.0.1:8000].
```

```
Press Ctrl+C to stop the server
```

Haciendo ctrl+click sobre el link nos debería abrir la página correctamente

Por último hicimos un commit hacia el github, específicamente al develop, con todos los cambios que hemos hecho con los siguientes comandos en este orden:

git branch

git checkout develop

git status

git add .

git commit -m "Add new Blade views and integrate tabs into Laravel project"

git push origin develop

Y así fue como subimos nuestras pestañas boceto hacia nuestro proyecto laravel.

I.S.B.O.

3MI

Anexos

Link del repositorio en github de la documentación:

<https://github.com/VifraSoft/Proyecto-ISBO/tree/00a54aa345280f7ace7e34ff6da178a4a5cf0e71/Proyecto-ISBO-2025>

Link del repositorio en github de la programación del proyecto:

<https://github.com/VifraSoft/DraftoStation-VifraSoft.git>