

S.I.G.P.D.

Programación Full Stack

PlayCode

Rol	Apellido	Nombre	C.I	Email
Coordinador	Farías	Paula	5.694.837-1	paulafar2007@gmail.com
Sub-Coordinador	López	Alex	5.646.776-5	aleexpias@gmail.com
Integrante 1	Morales	Luciana	5.700.540-5	luuumoralees@gmail.com

Docente: Laporta, Emanuel

**Fecha de
culminación
15/09/2025**

SEGUNDA ENTREGA

I.S.B.O.

3°MI



ÍNDICE

ÍNDICE.....	2
Primera Entrega.....	4
Justificación Tecnológica (Documento técnico).....	4
Ventajas de usar PHP para el backend.....	4
Elección del gestor de base de datos: MySQL.....	4
Framework Frontend: Bootstrap 5.3.....	5
Herramientas para control de versiones: Git + GitHub.....	6
Entorno de desarrollo: Visual Studio Code + XAMPP.....	6
Relación con los requerimientos del juego.....	7
Ejemplo 1: Juegos simples en PHP + MySQL.....	7
Ejemplo 2: Uso de Bootstrap en prototipos de juegos.....	7
Prototipo Visual.....	9
Página de inicio/bienvenida.....	9
Página del Tablero de Juego.....	9
Página de Resultados Finales.....	10
Diseño Responsivo.....	11
Esquema relacional normalizado.....	12
Normalización hasta la Tercera Forma Normal (3FN).....	14
1FN.....	14
2FN.....	14
3FN.....	14
DER a Modelo Relacional - Pasaje a Tablas.....	15
Restricciones no estructurales del juego y como se implementarían.....	16
Configuración del Entorno de Desarrollo.....	19
Instalación de XAMPP (Apache + MySQL).....	19
Pasos generales de instalación:.....	19
Instalación de VS Code y extensiones.....	20
Configuración mínima para ejecutar PHP.....	20
Cómo correr el prototipo.....	21
Configuración del Proyecto y Primeros Pasos.....	21
Verificación de herramientas instaladas.....	21
Creación del proyecto Laravel.....	21
Instalación de dependencias de Node.....	22
Ejecutar el servidor Laravel.....	22
Creación de layouts y views.....	22
Configuración de Git y flujo de trabajo.....	23
Control de Versiones.....	24
Repositorio de GitHub.....	24



Archivo README.....	24
Conclusión.....	25
Anexos.....	1
Anexo I. DER.....	1
Anexo II. Base de Datos.....	1
Anexo III . Descargas necesarias.....	1
Anexo IV . Repositorio de GitHub.....	1
Hoja testigo.....	1



Primera Entrega

Justificación Tecnológica (Documento técnico)

Ventajas de usar PHP para el backend

PHP es un lenguaje muy popular y usado hace años para hacer páginas web dinámicas. Es fácil de aprender y usar, incluso para quienes están empezando en programación. Esto ayuda a que el desarrollo sea más rápido y sencillo.

Además, PHP es compatible con casi todos los servidores y proveedores de hosting, así que no hay problemas para subir el proyecto a internet. También funciona bien con bases de datos como MySQL, que es lo que usaremos para guardar la información del juego.

Otro punto importante es que PHP tiene mucha comunidad y recursos, por lo que si surge alguna duda, siempre es fácil encontrar ayuda o tutoriales. Además, es un lenguaje abierto y gratis, lo que lo hace accesible para cualquier proyecto.

Con las versiones más nuevas, PHP mejoró mucho en velocidad y rendimiento, así que es bastante eficiente para juegos o aplicaciones web que no sean extremadamente complejos.

Por ello, elegimos PHP, porque es confiable, fácil de usar, compatible con la mayoría de los sistemas, y tiene todo lo que necesitamos para que nuestro juego funcione bien.

Elección del gestor de base de datos: MySQL

Para el proyecto elegimos MySQL como sistema gestor de base de datos, porque es uno de los más usados en el desarrollo web y se lleva muy bien con PHP. Es fácil de usar, tiene buena documentación y está disponible en la mayoría de los servidores y servicios de hosting.

Una de sus ventajas es que permite manejar bien grandes cantidades de datos sin complicarse. Además, es estable y seguro, lo que es importante para que nuestro sistema funcione correctamente y sin errores.

Otra razón para elegir MySQL es que ya viene incluido en herramientas como XAMPP o MAMP, que usamos para crear un entorno local de desarrollo. Esto nos



facilita trabajar y hacer pruebas desde nuestras computadoras sin necesidad de instalar cosas por separado.

También consideramos MariaDB, que es una alternativa muy parecida (de hecho, es un “fork” de MySQL), pero decidimos usar MySQL porque es más conocido, más usado en tutoriales y más fácil de encontrar ejemplos y soporte.

En resumen, usamos MySQL porque es confiable, funciona muy bien con PHP, está ampliamente soportado, y cumple con todo lo que necesitamos para guardar y consultar los datos del juego de forma rápida y segura.

Framework Frontend: Bootstrap 5.3

Para el diseño del frontend del juego elegimos Bootstrap 5.3 porque nos facilita muchísimo el trabajo visual y la estructura del sitio. Bootstrap viene con un sistema de grid muy útil que permite que todo se acomode automáticamente según el tamaño de pantalla (diseño responsivo), sin tener que escribir tanto CSS desde cero.

También incluye muchos componentes ya listos como botones, tarjetas, barras de navegación, formularios, etc., que podemos usar directamente sin tener que diseñarlos desde cero. Esto nos ahorra tiempo y asegura que la página se vea bien y sea fácil de usar en cualquier dispositivo.

Además, Bootstrap es fácil de integrar con HTML y tiene una documentación muy clara, lo que es ideal para este proyecto. Como el juego Draftosaurus tiene varias pantallas (inicio, tablero, resultados), nos conviene usar un framework que nos ayude a mantener todo bien organizado y visualmente coherente.



Herramientas para control de versiones: Git + GitHub

Para llevar el control del código del proyecto, usamos Git como sistema de control de versiones y GitHub como plataforma para alojar el repositorio online. Estas herramientas nos permiten:

- Guardar el historial de cambios en el código.
- Volver a versiones anteriores si algo falla.
- Trabajar en equipo sin pisarnos el trabajo.
- Documentar claramente los avances con mensajes de commit.

GitHub también nos permite compartir el proyecto fácilmente con los docentes o compañeros, y tener todo el código organizado en un solo lugar. Además, se puede agregar documentación técnica directamente en el repositorio, lo cual es útil para explicar cómo instalar o usar el sistema.

Entorno de desarrollo: Visual Studio Code + XAMPP

Para programar usamos Visual Studio Code (VS Code) porque es un editor liviano, gratuito y muy completo. Tiene muchas extensiones que ayudan a escribir código más rápido, como:

- Autocompletado de HTML, CSS, PHP
- Resaltado de sintaxis
- Previsualización en vivo
- Git integrado para trabajar con GitHub desde el editor

Además, usamos XAMPP como entorno local para correr el servidor web y la base de datos. XAMPP ya viene con Apache, PHP y MySQL, por lo que nos permite simular cómo funcionará el juego en un servidor real, directamente desde nuestras computadoras.

Todo esto nos permite probar, modificar y depurar el sistema sin depender de internet, y luego subirlo al repositorio de GitHub cuando esté listo.



Relación con los requerimientos del juego

Todas estas tecnologías se eligieron pensando en lo que necesita nuestro sistema. Como Draftosaurus es un juego web con varias pantallas, requiere una interfaz clara, que se vea bien en celulares, y que permita almacenar datos como jugadores, puntajes o resultados. Con estas herramientas:

- Podemos crear una interfaz visual rápida y responsiva (Bootstrap)
- Guardar la información del juego (MySQL + PHP)
- Tener el código bien organizado y compartido (Git + GitHub)
- Trabajar de forma local sin errores (VS Code + XAMPP)

Ejemplo 1: Juegos simples en PHP + MySQL

Sistema de puntuación para juegos en línea

- Muchos desarrolladores crean juegos simples tipo quiz, adivinanzas o dados usando PHP para manejar usuarios y guardar puntuaciones.
- Usan formularios HTML, PHP para procesar datos y MySQL para almacenarlos.

La lógica se hace con PHP y los resultados se guardan y muestran desde la base de datos.

Resumen: Nuestro juego Draftosaurus también tiene puntuaciones y jugadores, así que este tipo de estructura encaja perfectamente.

Ejemplo 2: Uso de Bootstrap en prototipos de juegos

Juego Trivia en línea

- Varios prototipos usan Bootstrap para hacer el diseño visual de pantallas de juegos sin tener que crear el CSS desde cero.
- Por ejemplo: pantallas con preguntas y respuestas, pantallas de bienvenida, ranking de puntos.
- Bootstrap se usa para los botones, la organización en columnas, y para que se vea bien en celular o PC.

Resumen: Nosotros también vamos a tener pantallas parecidas (inicio, tablero, resultados), por lo que usar Bootstrap es ideal.



Ejemplo 3: GitHub para proyectos educativos

Repositorios de proyectos escolares o de facultad

- Muchos estudiantes usan GitHub para subir su código, con un README.md explicando cómo correrlo y con carpetas organizadas (html/, css/, js/, docs/).
- Algunos suben también el documento técnico, las capturas de pantalla, y el código fuente por separado.

Usar GitHub demuestra organización y permite compartir el proyecto con docentes o compañeros fácilmente.



Prototipo Visual

Página de inicio/bienvenida

¿Qué muestra?

- Encabezado con logo de PlayCode, nombre estilizado y menú de navegación (Inicio, Jugar, Ranking).
- Carrusel de imágenes de los juegos disponibles (por ejemplo: Animaldraft y tablero general).
- Mensaje de bienvenida con descripción del proyecto:
"Sumate a la mesa digital para aprender y competir..."
- Botón "Iniciar sesión" que simula el acceso al sistema.
- Estética visual limpia, con fondo claro y tipografía moderna (Montserrat + Orbitron).

Objetivo: Dar una primera impresión atractiva del sistema, explicar de qué trata el juego, e invitar a jugar.

Página del Tablero de Juego

¿Qué muestra?

- Encabezado igual al del index (logo + menú de navegación unificado).
- Diseño en dos columnas:
 - Izquierda:
 - Título "Tablero de Juego"
 - Imagen grande del tablero (tablero.jpeg)
 - Derecha:
 - Lista de fichas disponibles (animalitos y zonas) en tarjetas de color rojo
 - Cada ficha tiene nombre

Objetivo: Simular la visualización del tablero de juego y mostrar qué fichas hay disponibles para jugar.



Página de Resultados Finales

¿Qué muestra?

- Encabezado unificado (logo, nombre, menú).
- Un mensaje destacado de ganador, por ejemplo:
“¡Felicidades, Jugador 2! Ganaste esta partida.”
- Una tabla de puntuaciones con:
 - Posiciones (1°, 2°, 3°)
 - Nombres de jugadores
 - Puntos obtenidos
- Dos botones de navegación:
 - “Volver a jugar”
 - “Ir a inicio”

Objetivo: Simular una pantalla de final de partida con puntajes claros, ganador destacado y opciones para volver a empezar o salir.




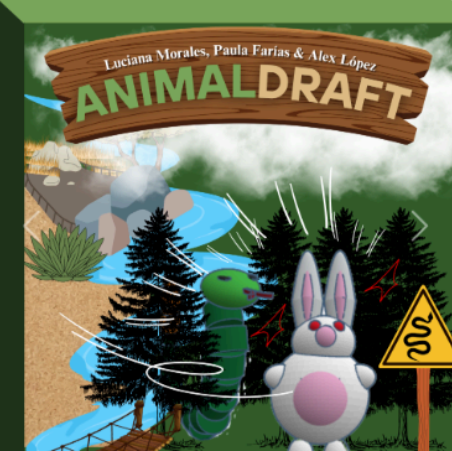
Diseño Responsivo

Se utilizó Bootstrap 5.3 para lograr un diseño totalmente adaptable a diferentes dispositivos. Mediante el sistema de grillas (container, row, col-lg-...) y clases como d-flex, img-fluid, navbar-collapse, se logra que los elementos:

- Se ven en dos columnas en el escritorio.
- Se adapta automáticamente en celulares.
- Mantener el orden visual deseado.

Ejemplos de cómo se vería en un celular

 PlayCode




¡Bienvenid@ a PlayCode!

Únete a la mesa digital para aprender, competir y divertirte.


No hace falta instalar nada: jugá directo desde el navegador, donde y cuando quieras.

Cientos de jugadores ya disfrutan de Animaldraft (inspirado en Draftosaurus) y otros juegos interactivos.

Gratis, fácil y muy divertido.

 PlayCode

AnimalDraft




Fichas disponibles

Caracol

Conejo

Ratón de campo

Camello

 PlayCode

Inicio

Jugar

Ranking

Español

¡Felicidades, Luciana! Ganaste esta partida.

Posición	Jugador	Puntaje
1	Luciana	23 puntos
2	Alex	18 puntos
3	Paula	14 puntos

Volver a jugar

Ir a inicio

© 2025 PlayCode. Todos los derechos reservados.



Esquema relacional normalizado

Jugador

PK: id_jugador

Atributos: nombre, correo, contrasena

Partida

PK: id_partida

Atributos: fecha_inicio, estado

Jugador_Partida

PK: id_jugador_partida

FK1: id_jugador

FK2: id_partida

Atributos: puntuacion

Animal

PK: id_animal

Atributos: nombre, color

Recinto

PK: id_recinto

Atributos: nombre, descripcion, restriccion

Colocacion

PK: id_colocacion

FK1: id_jugador_partida

FK2: id_animal

FK3: id_recinto

Atributos: turno

Dado

PK: id_dado

FK: id_partida

Atributos: turno, valor



PlayCode

15 de Septiembre

Ranking

PK: id_jugador

FK: id_jugador

Atributos: partidas_jugadas, partidas_ganadas, puntos_totales,
promedio_puntos



Normalización hasta la Tercera Forma Normal (3FN)

1FN

- Cada atributo es atómico (sin valores repetidos o compuestos).
- Todas las tablas cumplen con 1FN.

2FN

Eliminamos dependencias parciales:

- En Jugador_Partida, la clave primaria es un solo campo id_jugador_partida, así que no hay dependencias parciales.
- En Ranking, los atributos calculados (promedio_puntos) están como campo generado, no rompen 2FN.

3FN

Verificamos dependencias transitivas:

- Correo y contraseña dependen directamente de id_jugador.
- En Colocacion, turno depende solo de id_colocacion y no de otras tablas indirectamente.
- No hay atributos no clave que dependan de otros no clave.

En conclusión vemos que se cumplen las 3FN debido a que:

- Cada tabla tiene datos atómicos (1FN).
- Los campos dependen de toda la clave primaria (2FN).
- No hay dependencias entre campos no clave (3FN).



DER a Modelo Relacional - Pasaje a Tablas

En el DER original (el cual se encuentra en el Anexo I) las entidades y relaciones se transforman así:

- Entidad fuerte Jugador → tabla Jugador
- Entidad fuerte Partida → tabla Partida
- Relación Participa entre Jugador y Partida → tabla Jugador_Partida
- Entidad fuerte Animal → tabla Animal
- Entidad fuerte Recinto → tabla Recinto
- Relación Coloca → tabla Colocacion
- Entidad fuerte Dado → tabla Dado
- Entidad derivada Ranking → tabla Ranking



Restricciones no estructurales del juego y como se implementarían

Un jugador no puede jugar dos veces la misma partida

- Se verifica antes de que un jugador se una a una partida que no haya participado previamente.

La puntuación de un jugador no puede ser negativa

- Se controla que cualquier actualización de puntuación mantenga valores mayores o iguales a cero.

Una partida solo puede tener un máximo de jugadores

- Antes de permitir que un jugador se una, se comprueba que la partida no haya alcanzado el límite de participantes.



Un animal de tipo “único” solo puede estar en un recinto a la vez

- Se valida que no exista otra colocación del mismo animal antes de permitir su ubicación en un recinto.

Un recinto no puede superar su capacidad máxima de animales

- Se controla cuántos animales ya están en el recinto y se impide colocar más de los permitidos.

El turno de colocación debe seguir un orden consecutivo

- Se asegura que los jugadores coloquen en el orden correcto, sin saltos de turno.

El valor del dado debe estar entre 1 y 6

- Se garantiza que los valores generados para el dado siempre estén dentro del rango permitido por el juego.

Cada jugador solo puede lanzar el dado una vez por turno

- Se verifica que un jugador no repita su tirada dentro del mismo turno.

El turno del dado debe coincidir con el turno del jugador correspondiente

- Solo el jugador que tiene el turno activo puede lanzar el dado.

Los puntos solo se cuentan de partidas finalizadas

- Solo se incluyen en el ranking los puntos de partidas cuyo estado sea “finalizada”.



El promedio de puntos se calcula como puntos totales / partidas jugadas

- Se realiza un cálculo que divide los puntos acumulados entre las partidas efectivamente jugadas.

No se permite puntuación negativa; mínimo 0 puntos

- Se asegura que cualquier puntuación mínima sea cero, evitando valores negativos.



Configuración del Entorno de Desarrollo

Instalación de XAMPP (Apache + MySQL)

XAMPP es un paquete que incluye **Apache** (servidor web), **MySQL** (base de datos) y PHP, lo que permite montar un entorno de desarrollo local para proyectos web de manera rápida y sencilla.

Pasos generales de instalación:

1. Descarga

- Se debe descargar XAMPP desde la página oficial.
- El link de descarga se encuentra en los anexos de esta documentación.

2. Ejecución del instalador

- Ejecutar el archivo descargado y seguir los pasos del asistente de instalación.
- Durante la instalación, seleccionar **Apache y MySQL**, que son los servicios necesarios para este proyecto.

3. Finalización e inicio

- Finalizar la instalación.
- Abrir **XAMPP Control Panel** y arrancar los módulos **Apache y MySQL** haciendo clic en "Start".
- Verificar que ambos servicios estén corriendo correctamente (el fondo de los módulos se pone verde).

4. Verificación

- Abrir el navegador y acceder a <http://localhost/> para comprobar que Apache funciona.
- Abrir **phpMyAdmin** desde el panel de XAMPP para comprobar que MySQL funciona.



Mantener XAMPP abierto y con los servicios iniciados mientras se trabaja en el proyecto, ya que Laravel y la base de datos dependen de Apache y MySQL activos.

Instalación de VS Code y extensiones

- Instalar el programa siguiendo los pasos por defecto.
- Abrir VS Code y agregar extensiones recomendadas:
 - **PHP Intelephense** → autocompletado y navegación en PHP.
 - **Laravel Blade Snippets** → atajos para archivos Blade.
 - **Prettier - Code Formatter** → formateo automático de código.
 - **Live Server** (opcional, para probar HTML/JS rápido).

Configuración mínima para ejecutar PHP

- Verificar que PHP esté instalado (XAMPP ya lo incluye).
- Configurar VS Code para reconocer PHP:
 - Ir a Settings → Extensions → PHP y agregar la ruta del ejecutable PHP (xampp/php/php.exe).
 - Probar PHP en la terminal: `php -v`
- Debe mostrar la versión instalada.



Cómo correr el prototipo

1. Abrir terminal en la carpeta del proyecto Laravel.
2. Instalar dependencias PHP: `composer install`
3. Instalar dependencias JS si se usan (Vite/Node):
 - a. `npm install`
 - b. `npm run dev`
4. Correr el servidor Laravel: `php artisan serve`
5. Abrir el navegador en la URL que indica Laravel (por defecto: `http://127.0.0.1:8000`).
6. Probar navegación entre vistas, formularios y prototipo visual del juego.

Configuración del Proyecto y Primeros Pasos

Verificación de herramientas instaladas

Antes de comenzar, se debe comprobar que las herramientas necesarias estén correctamente instaladas ejecutando:

- `composer -v` → muestra la versión de Composer.
- `php -v` → muestra la versión de PHP.
- `node -v` → muestra la versión de Node.js.
- `npm -v` → muestra la versión de NPM.

Si todas muestran su versión, las herramientas están listas para usar.

Creación del proyecto Laravel

1. Crear un nuevo proyecto Laravel: `composer create-project laravel/laravel Segunda_Entrega1`
2. Entrar en la carpeta del proyecto recién creado: `cd Segunda_Entrega1`



Instalación de dependencias de Node

1. Ejecutar npm install para instalar todas las dependencias de Node necesarias.
2. Compilar CSS y JS usando Vite: npm run dev

Ejecutar el servidor Laravel

Iniciar el servidor local con: `php artisan serve`, accedemos al proyecto desde el navegador en la URL que indique Laravel (por defecto `http://127.0.0.1:8000`).

Creación de layouts y views

1. Abrir el proyecto en **Visual Studio Code**.
2. En resources/views, crear una carpeta llamada layouts.
3. Dentro de layouts, crear app.blade.php que contendrá:
 - `<head>`
 - `<body>`
 - `<header>`
 - `<footer>`

Estructura base que se reutilizará en todas las vistas.

4. Crear las views específicas del proyecto (bienvenida, tablero, resultados) y los archivos CSS correspondientes en resources/css.



Configuración de Git y flujo de trabajo

1. Inicializar el repositorio: `git init`
2. Cambiar a la rama develop para trabajar desde ahí: `git checkout -b develop`
3. Crear una feature para subir las views completas: `git checkout -b feature/views`
4. Añadir los cambios: `git add .`
5. Hacer commit describiendo los cambios realizados: `git commit -m "Agregar views y layout iniciales"`
6. Subir los cambios al repositorio remoto: `git push origin feature/views`
7. Hacer merge de la feature a develop para integrar los cambios: `git merge feature/views`



Control de Versiones

Repositorio de GitHub

El repositorio del proyecto se organiza siguiendo el modelo **GitFlow**, con tres ramas principales: master, develop y feature. La rama master contiene la versión estable del proyecto y no se trabaja directamente sobre ella; únicamente se realizan merges desde develop cuando se libera una versión lista para producción. La rama develop es la rama de integración, donde se combinan todas las funcionalidades en desarrollo y se realizan pruebas antes de pasar a master. Cada nueva funcionalidad se desarrolla en una rama específica denominada feature, creada a partir de develop. Por ejemplo, feature/views para las vistas y layouts o feature/forms para el formulario de trackeo. Una vez que la feature está completa y probada, se hace merge a develop y, cuando develop está estable, se integra a master. Este flujo permite mantener el proyecto organizado, facilita la colaboración y asegura que los cambios se integren de manera controlada, además de permitir un historial de commits limpio y separado por funcionalidades.

Archivo README

El archivo README.md del proyecto contiene toda la información básica necesaria para comprender y utilizar el proyecto. Incluye el nombre del proyecto, la lista de integrantes que participaron en su desarrollo y el link al repositorio de documentación donde se encuentra la descripción completa, instrucciones de instalación, configuración del entorno de desarrollo, detalles sobre las vistas y layouts, flujo de trabajo con GitFlow, y demás recursos necesarios para ejecutar y mantener el proyecto. Su objetivo es servir como guía rápida y referencia para cualquier usuario o desarrollador que desee trabajar con el proyecto.



Conclusión

En esta primera entrega se logró establecer la base del proyecto utilizando Laravel, incluyendo la organización de las vistas, layouts y la estructura de carpetas con CSS y JS separados, así como la configuración del entorno de desarrollo y el flujo de trabajo con GitFlow. Se implementaron las pantallas principales del prototipo, como la bienvenida, el tablero del juego, el formulario de trackeo y la pantalla de resultados, dejando lista la interfaz visual para futuras funcionalidades.

Queda pendiente en próximas fases la implementación de la lógica del juego, incluyendo el cálculo de puntajes, el comportamiento de los animales y recintos, y la integración de los turnos y el dado.

La experiencia del grupo con las tecnologías fue positiva; se reforzó el conocimiento en Laravel, gestión de dependencias con Composer y Node y control de versiones con Git y GitHub. Además, se fortalecieron las habilidades de trabajo colaborativo y organización mediante el uso de ramas feature y el flujo GitFlow, lo que permitió un desarrollo ordenado y eficiente.



Anexos

Anexo I. DER

Realizamos el DER con la página dbdiagram.io

Jugador	
PK	<u>id_jugador</u> INT AUTO INCREMENT
	nombre VARCHAR(100) NOT NULL
	correo VARCHAR(100) NOT NULL
	contrasena VARCHAR(100) NOT NULL
	tipo_usuario ENUM('Normal','Admin') DEFAULT 'Nor'

Partida	
PK	<u>id_partida</u> INT AUTO INCREMENT
	fecha_inicio TIMESTAMP DEFAULT CURRENT_TIM
	estado ENUM('En curso', 'Finalizada') DEFAULT 'En

Jugador_Partida	
PK	<u>id_partida</u> INT
PK	<u>id_jugador</u> INT
PK	<u>id_jugador_partida</u> INT AUTO INCREMENT
	puntuacion INT DEFAULT 0
	es_ganador BOOLEAN DEFAULT FALSE, -- saber q
	FOREIGN KEY (id_jugador) REFERENCES Jugador
	FOREIGN KEY (id_partida) REFERENCES Partida(i

Animal	
PK	<u>id_animal</u> INT AUTO INCREMENT
	nombre VARCHAR(50) NOT NULL
	color VARCHAR(20)

Recinto	
PK	<u>id_recinto</u> INT AUTO INCREMENT
	nombre VARCHAR(100)
	descripcion TEXT
	restriccion TEXT

Recinto_Jugador	
PK	<u>id_recinto</u> INT
PK	<u>id_jugador_partida</u> INT
PK	<u>id_recinto_jugador</u> INT AUTO INCREMENT
	FOREIGN KEY (id_jugador_partida) REFERENCES
	FOREIGN KEY (id_recinto) REFERENCES Recinto(i

Colocacion	
PK	<u>id_recinto_jugador</u> INT
PK	<u>id_animal</u> INT
PK	<u>id_jugador_partida</u> INT
PK	<u>id_colocacion</u> INT AUTO INCREMENT
	cantidad INT DEFAULT 1, -- cantidad de animales de
	FOREIGN KEY (id_jugador_partida) REFERENCES
	FOREIGN KEY (id_animal) REFERENCES Animal(i
	FOREIGN KEY (id_recinto_jugador) REFERENCES

Dado	
PK	<u>id_partida</u> INT
PK	<u>id_dado</u> INT AUTO INCREMENT
	turno INT
	valor VARCHAR(50)
	FOREIGN KEY (id_partida) REFERENCES Partida(i

Ranking	
PK	<u>id_jugador</u> INT
	partidas_jugadas INT DEFAULT 0
	partidas_ganadas INT DEFAULT 0
	puntos_totales INT DEFAULT 0
	promedio_puntos DECIMAL(5,2) GENERATED ALW
	CASE
	WHEN partidas_jugadas > 0 THEN puntos_totales /
	ELSE 0
	END



Anexo II. Base de Datos

Código de la base de datos

```
CREATE DATABASE ProyectoFinal;  
USE ProyectoFinal;  
SHOW TABLES;
```

```
CREATE TABLE Jugador (  
    id_jugador INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    correo VARCHAR(100) NOT NULL,  
    contrasena VARCHAR(100) NOT NULL  
);
```

```
CREATE TABLE Partida (  
    id_partida INT AUTO_INCREMENT PRIMARY KEY,  
    fecha_inicio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    estado ENUM('En curso', 'Finalizada') DEFAULT 'En curso'  
);
```

```
CREATE TABLE Jugador_Partida (  
    id_jugador_partida INT AUTO_INCREMENT PRIMARY KEY,  
    id_jugador INT,  
    id_partida INT,  
    puntuacion INT DEFAULT 0,  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador),  
    FOREIGN KEY (id_partida) REFERENCES Partida(id_partida)  
);
```

```
CREATE TABLE Animal (  
    id_animal INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(50) NOT NULL,  
    color VARCHAR(20)  
);
```



```
CREATE TABLE Recinto (  
    id_recinto INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100),  
    descripcion TEXT,  
    restriccion TEXT  
);  
  
CREATE TABLE Colocacion (  
    id_colocacion INT AUTO_INCREMENT PRIMARY KEY,  
    id_jugador_partida INT,  
    id_animal INT,  
    id_recinto INT,  
    turno INT,  
    FOREIGN KEY (id_jugador_partida) REFERENCES  
Jugador_Partida(id_jugador_partida),  
    FOREIGN KEY (id_animal) REFERENCES Animal(id_animal),  
    FOREIGN KEY (id_recinto) REFERENCES Recinto(id_recinto)  
);  
  
CREATE TABLE Dado (  
    id_dado INT AUTO_INCREMENT PRIMARY KEY,  
    id_partida INT,  
    turno INT,  
    valor VARCHAR(50),  
    FOREIGN KEY (id_partida) REFERENCES Partida(id_partida)  
);  
  
CREATE TABLE Ranking (  
    id_jugador INT PRIMARY KEY,  
    partidas_jugadas INT DEFAULT 0,  
    partidas_ganadas INT DEFAULT 0,  
    puntos_totales INT DEFAULT 0,  
    promedio_puntos DECIMAL(5,2) GENERATED ALWAYS AS (  
        CASE  
            WHEN partidas_jugadas > 0 THEN puntos_totales / partidas_jugadas  
            ELSE 0  
        END  
    ) STORED,  
    FOREIGN KEY (id_jugador) REFERENCES Jugador(id_jugador)  
);
```



Anexo III . Descargas necesarias

XAMPP (para PHP y MySQL) → <https://www.apachefriends.org/es/index.html>

Composer (para dependencias PHP) → <https://getcomposer.org/download/>

Node.js + npm (para dependencias front) → <https://nodejs.org/es/download>



PlayCode

15 de Septiembre

Anexo IV . Repositorio de GitHub

Link del repositorio → https://github.com/playcodecompany/Segunda_Entrega1.git



PlayCode

15 de Septiembre

Hoja testigo

Montevideo 14 de julio del 2025

ACUSE DE RECIBO PARA ENTREGAS DE PROYECTO

Corresponde a ENTREGA N° 1

Los alumnos de 3°MI del turno matutino integrantes del grupo de proyecto PlayCode

ROL	APELLIDO	NOMBRE	CI	E-MAIL
Coordinador	Farías	Paula	5.694.837-1	paulafar2007@gmail.com
Sub-Coordinador	López	Alex	5.646776-5	aleexpias@gmail.com
Integrante N°1	Morales	Luciana	5.700540-5	luuumorales@gmail.com

Por contacto al correo: companyplaycode@gmail.com

Firmas:

COORDINADOR

SUBCOORDINADOR

INTEGRANTE 1



PlayCode

15 de Septiembre



FARÍAS, PAULA

COORDINADOR



LÓPEZ, ALEX

SUBCOORDINADOR



MORALES, LUCIANA

INTEGRANTE 1