



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
10/11/2025**

TERCERA ENTREGA

I.S.B.O.

3°MI



ÍNDICE

ÍNDICE.....	2
Justificación Tecnológica (Documento técnico).....	4
Lenguaje de programación backend: PHP 8.2.....	4
Framework backend: Laravel 10.....	4
Gestor de base de datos: MySQL 8.0.....	5
Framework frontend: Bootstrap 5.3.....	5
Control de versiones: Git y GitHub.....	6
Entorno de desarrollo: Visual Studio Code y XAMPP 8.2.12.....	6
Relación con los requerimientos del juego.....	6
Ejemplo 1: Juegos simples en PHP + MySQL.....	7
Ejemplo 2: Uso de Bootstrap en prototipos de juegos.....	7
Ejemplo 3: GitHub para proyectos educativos.....	7
Prototipo Visual.....	8
Página de inicio o bienvenida.....	8
Página de Jugar / Creación de Partida.....	8
Página de Tablero de Juego.....	9
Página de Resultados Finales.....	9
Panel de Administración.....	10
Página de Perfil de Usuario.....	10
Diseño Responsivo.....	11
Ejemplos de cómo se vería en Mobile-First (320px - 576px).....	12
Ejemplos de cómo se vería en Tablet (577px - 992px).....	13
Ejemplos de cómo se vería en Escritorio (≥993px).....	14
Esquema relacional normalizado.....	16
Esquema relacional normalizado (nueva base de datos).....	17
3FN - Base de Datos Inicial.....	18
1FN.....	18
2FN.....	18
3FN.....	19
3FN - Base de Datos Final.....	20
1FN.....	20
2FN.....	20
3FN.....	22
DER a Modelo Relacional - Pasaje a Tablas.....	22
Restricciones no estructurales del juego y como se implementan.....	23
Configuración del Entorno de Desarrollo.....	25
Instalación de XAMPP (Apache + MySQL).....	25



Instalación de herramientas adicionales.....	26
Instalación de Visual Studio Code y extensiones.....	26
Clonación del proyecto y configuración inicial.....	27
Cómo correr el prototipo.....	28
Control de Versiones.....	29
Archivo README.....	30
Conclusión.....	30
Anexos.....	1
Anexo I. DER.....	1
Anexo II. Diagramas UML.....	1
Anexo III. Base de Datos.....	1
Anexo IV . Descargas necesarias.....	1
Anexo V . Repositorio de GitHub.....	1
Hoja testigo.....	1



Justificación Tecnológica (Documento técnico)

Lenguaje de programación backend: PHP 8.2

Para el desarrollo del proyecto se utilizó **PHP 8.2** como lenguaje principal del backend. PHP es un lenguaje muy popular y ampliamente usado para crear aplicaciones web dinámicas. Su sintaxis es sencilla, lo que facilita el aprendizaje y acelera el proceso de desarrollo, especialmente para equipos que están dando sus primeros pasos en la programación web. La versión 8.2 de PHP incluye mejoras importantes en velocidad, rendimiento y seguridad respecto a versiones anteriores, como PHP 5.6 o 7.x, lo cual garantiza una mejor compatibilidad con frameworks modernos como Laravel 10 y con bases de datos como MySQL 8.0. Además, PHP es de código abierto, gratuito y compatible con casi todos los servidores y servicios de hosting, lo que permite desplegar el proyecto fácilmente en internet. Su gran comunidad y la cantidad de recursos disponibles también aseguran soporte y soluciones a posibles problemas durante el desarrollo.

Framework backend: Laravel 10

Para estructurar el código del backend se implementó el **framework Laravel 10**, una herramienta moderna que permite trabajar de manera ordenada siguiendo el patrón MVC (Modelo-Vista-Controlador). Laravel ofrece múltiples ventajas, como una gestión eficiente de rutas, controladores y migraciones, así como un ORM (Eloquent) que simplifica la manipulación de bases de datos sin necesidad de escribir consultas SQL complejas. Además, incluye mecanismos de seguridad integrados, como la protección contra ataques de inyección SQL o CSRF, y herramientas que optimizan el rendimiento y la organización del código. Gracias a estas características, Laravel resulta ideal para el sistema del juego Draftosaurus, que requiere manejar usuarios, partidas, puntuaciones y rankings de forma segura y estructurada.



Gestor de base de datos: MySQL 8.0

Como gestor de base de datos se eligió **MySQL 8.0**, por su integración nativa con PHP y Laravel, y por ser uno de los sistemas más utilizados en el desarrollo web. MySQL es un gestor relacional, estable y confiable, que permite manejar grandes volúmenes de información de manera eficiente. Además, su instalación e integración dentro del entorno XAMPP facilita las pruebas locales sin necesidad de configuraciones adicionales. MySQL es también de código abierto y posee una amplia documentación, lo que lo convierte en una excelente opción para el manejo de datos estructurados como jugadores, partidas y puntajes dentro del proyecto.

Framework frontend: Bootstrap 5.3

Para el diseño del frontend se empleó **Bootstrap 5.3**, un framework CSS que permite crear interfaces web atractivas y adaptables a distintos tamaños de pantalla. Bootstrap cuenta con un sistema de rejilla que facilita la organización de los elementos visuales y proporciona componentes ya diseñados como botones, formularios, tarjetas y menús, lo que acelera el trabajo visual sin necesidad de escribir todo el código CSS desde cero. Además, su integración con Laravel Blade y HTML permite mantener un diseño limpio y coherente en todas las pantallas del juego, incluyendo el inicio, el tablero y los resultados. Esta herramienta asegura que la aplicación sea responsiva y ofrezca una buena experiencia de usuario tanto en dispositivos móviles como en computadoras.



Control de versiones: Git y GitHub

Para el control de versiones del proyecto se utilizaron **Git y GitHub**. Git permite registrar y gestionar los cambios en el código fuente, mientras que GitHub sirve como repositorio en línea para almacenar y compartir el proyecto. Estas herramientas facilitan el trabajo colaborativo, ya que permiten a los integrantes del equipo trabajar en paralelo, mantener un historial de versiones, recuperar versiones anteriores en caso de error y documentar el progreso mediante mensajes de commit. Además, GitHub proporciona un espacio centralizado donde se puede alojar el código, la documentación técnica y las capturas del proyecto, lo cual mejora la organización general y facilita la presentación del trabajo.

Entorno de desarrollo: Visual Studio Code y XAMPP 8.2.12

El entorno de desarrollo se configuró con **Visual Studio Code y XAMPP 8.2.12**. Visual Studio Code es un editor liviano, gratuito y altamente personalizable que ofrece extensiones para autocompletado, resaltado de sintaxis, integración con Git y depuración del código. XAMPP, por su parte, incluye Apache, PHP 8.2 y MySQL 8.0, lo que permite ejecutar el proyecto de forma local y simular el entorno de un servidor real sin necesidad de conexión a internet. Este entorno facilita las pruebas, la depuración y la puesta a punto del sistema antes de su despliegue final.

Relación con los requerimientos del juego

Todas las tecnologías fueron seleccionadas en función de los requerimientos específicos del sistema. Como Draftosaurus es un juego web que incluye registro de usuarios, partidas y rankings, era necesario contar con un backend robusto y una interfaz visual dinámica. PHP y Laravel permiten gestionar la lógica y la base de datos de forma eficiente, mientras que Bootstrap se encarga de la parte visual, asegurando una buena experiencia en cualquier dispositivo. Git y GitHub mantienen el control del código y la colaboración ordenada entre los integrantes, y el entorno de desarrollo basado en VS Code y XAMPP asegura un trabajo fluido durante todo el proceso de programación.



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 o bienvenida

La pantalla principal del sistema presenta el encabezado con el logo de PlayCode, el nombre estilizado del proyecto y un menú de navegación que incluye las secciones Inicio, Jugar, Panel (accesible solo para administradores) y Perfil. También se incorpora un selector de idioma que permite alternar entre español e inglés, mejorando la accesibilidad del sitio.

En el cuerpo de la página se incluye un carrusel de imágenes que muestra ejemplos del tablero, fichas y componentes del juego. Debajo, se añaden opiniones ficticias de jugadores, una sección informativa sobre la empresa con su ubicación, y un apartado de merchandising con artículos personalizados de PlayCode.

El diseño visual mantiene una estética moderna y limpia, con fondo claro y tipografía combinada entre Montserrat y Orbitron, logrando una apariencia profesional y atractiva. El objetivo de esta pantalla es dar una primera impresión atractiva, explicar brevemente de qué trata el sistema y motivar al usuario a unirse a la experiencia digital.

Página de Jugar / Creación de Partida

Para acceder a la sección Jugar, el usuario debe estar registrado y tener la sesión iniciada. En caso contrario, el sistema lo redirige automáticamente a la página de Inicio de sesión o Registro.

Dentro de esta sección, se puede crear una nueva partida, ingresando un nombre para la partida y la cantidad de jugadores, hasta un máximo de cinco (incluyendo al creador). Además, deben agregarse los ID de jugador correspondientes. Si alguno de los jugadores ingresados no se encuentra registrado en la base de datos, la creación de la partida no será posible.

Una vez creada correctamente, el sistema redirige al tablero de juego, donde se visualizan las reglas, el tablero principal, el turno actual, el nombre del jugador en turno, la ronda en curso y dos tablas: una para registro de movimientos y otra para



puntuaciones. Esta pantalla tiene el propósito de simular la dinámica real del juego y facilitar el seguimiento de la partida en tiempo real.

Página de Tablero de Juego

El tablero mantiene un encabezado unificado con el logo y menú principal. Su diseño está dividido en dos columnas: a la izquierda se encuentra la imagen principal del tablero (tablero.jpeg), junto con el título “Tablero de Juego”; y a la derecha, una lista de fichas disponibles representadas con tarjetas de color rojo que incluyen el nombre de cada animal o zona.

La interfaz busca transmitir una experiencia de juego clara, permitiendo visualizar el desarrollo de la partida y comprender fácilmente qué elementos están activos en cada momento.

Página de Resultados Finales

Al finalizar la partida, el sistema muestra la pantalla de resultados. En ella se presenta un mensaje destacado que anuncia al ganador, por ejemplo: “*¡Felicidades, Jugador 2! Ganaste esta partida.*”

Debajo, se despliega una tabla con las posiciones finales, los nombres de los jugadores y los puntos obtenidos. También se incluyen dos botones de navegación: “Volver a jugar” y “Ir a inicio”, que permiten reiniciar o salir del juego.

En esta etapa se incluye también un resumen completo de la partida, con un podio visual y una tabla detallada que muestra el rendimiento de cada participante, brindando una visión general de los resultados obtenidos.



Panel de Administración

El panel es una sección exclusiva para usuarios con rol de administrador. Desde allí se pueden visualizar los detalles de todos los usuarios registrados, incluyendo su nombre, correo electrónico, ID y rol (administrador o jugador).

Además, el administrador tiene la posibilidad de editar datos como nombre o correo, eliminar usuarios del sistema y agregar nuevos registros. Esta sección está diseñada para ofrecer un control total sobre la gestión de usuarios, manteniendo la integridad y organización del sistema.

Página de Perfil de Usuario

La sección de perfil personal permite al usuario consultar su nombre, correo electrónico, ID personal y una tabla con las partidas jugadas. Desde aquí, el jugador puede acceder rápidamente a su historial y verificar su progreso dentro del sistema.

El objetivo de esta pantalla es ofrecer un espacio individual donde cada usuario pueda visualizar su información y actividad, reforzando la sensación de pertenencia dentro de la comunidad PlayCode.



Diseño Responsivo

Para el diseño visual y la adaptabilidad, se utilizó Bootstrap 5.3, aprovechando su sistema de grillas (container, row, col-lg-...) y clases como d-flex, img-fluid y navbar-collapse. Esto permite que el sitio se adapte automáticamente a distintos tamaños de pantalla, mostrando un formato en dos columnas en escritorio y una vista simplificada en dispositivos móviles.

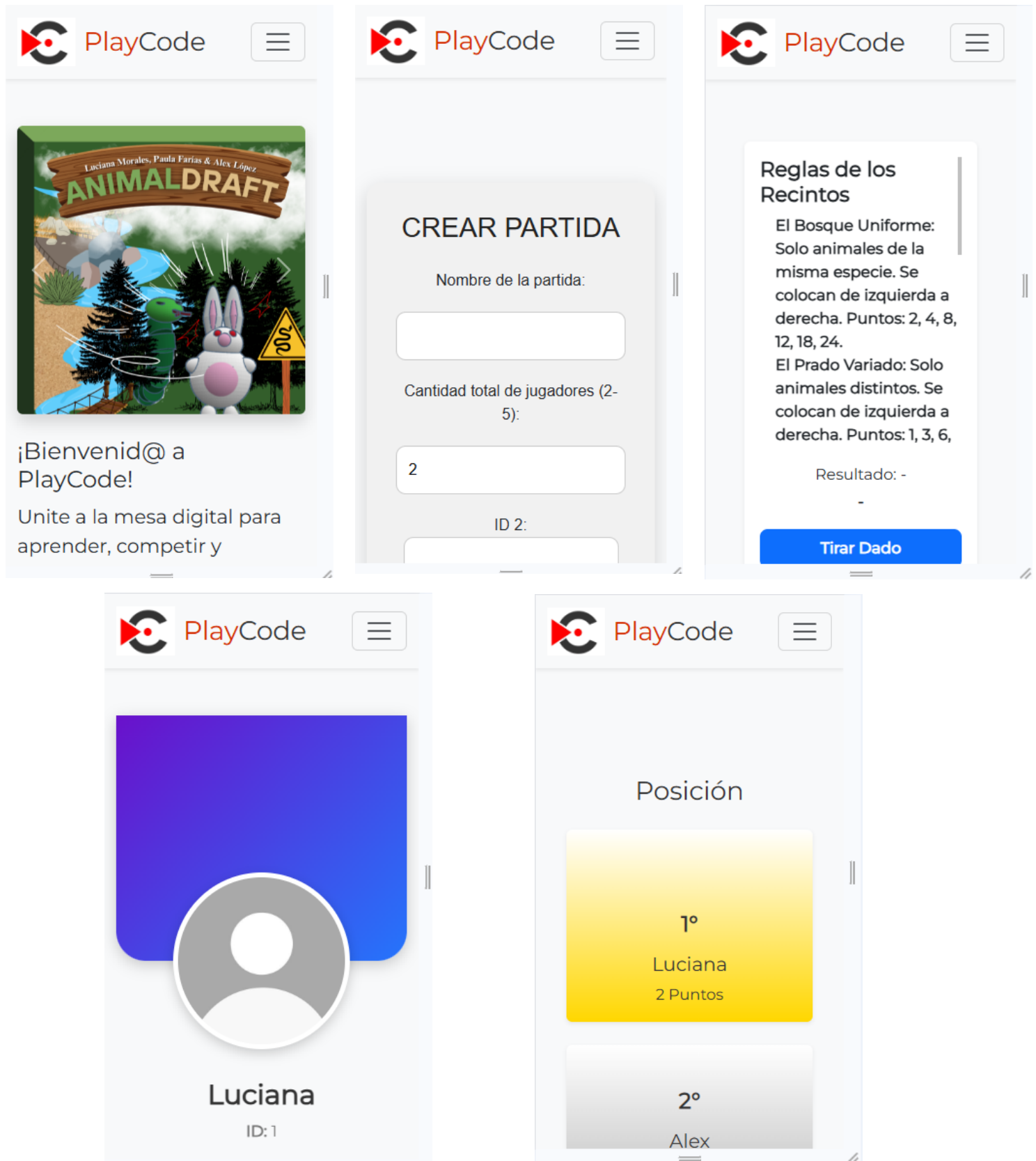
De esta forma, se asegura una navegación fluida, legible y consistente, sin importar desde qué dispositivo acceda el usuario.



PlayCode

10 de Noviembre

Ejemplos de cómo se vería en Mobile-First (320px - 576px)

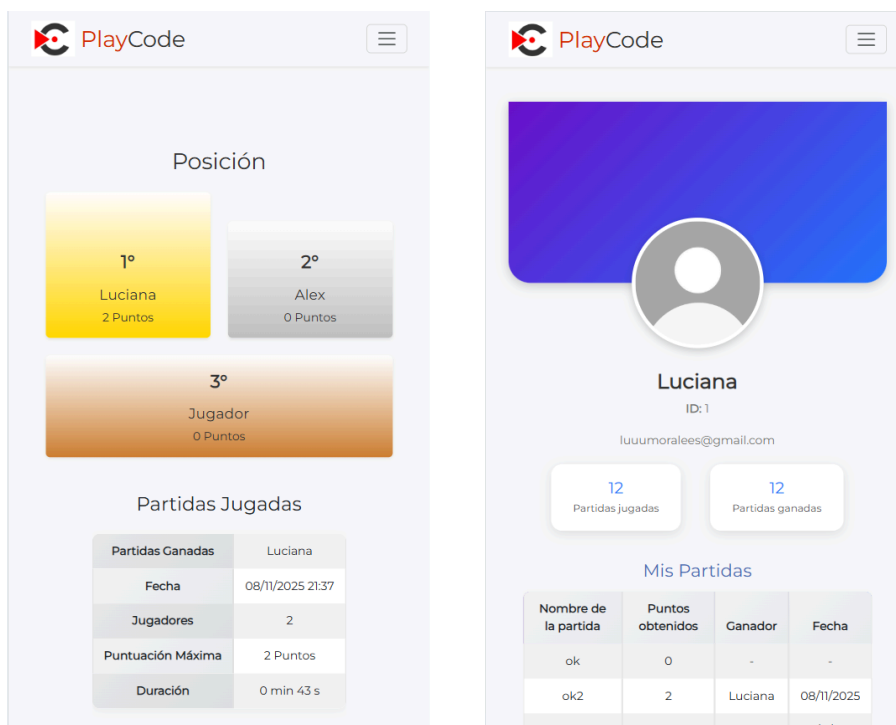
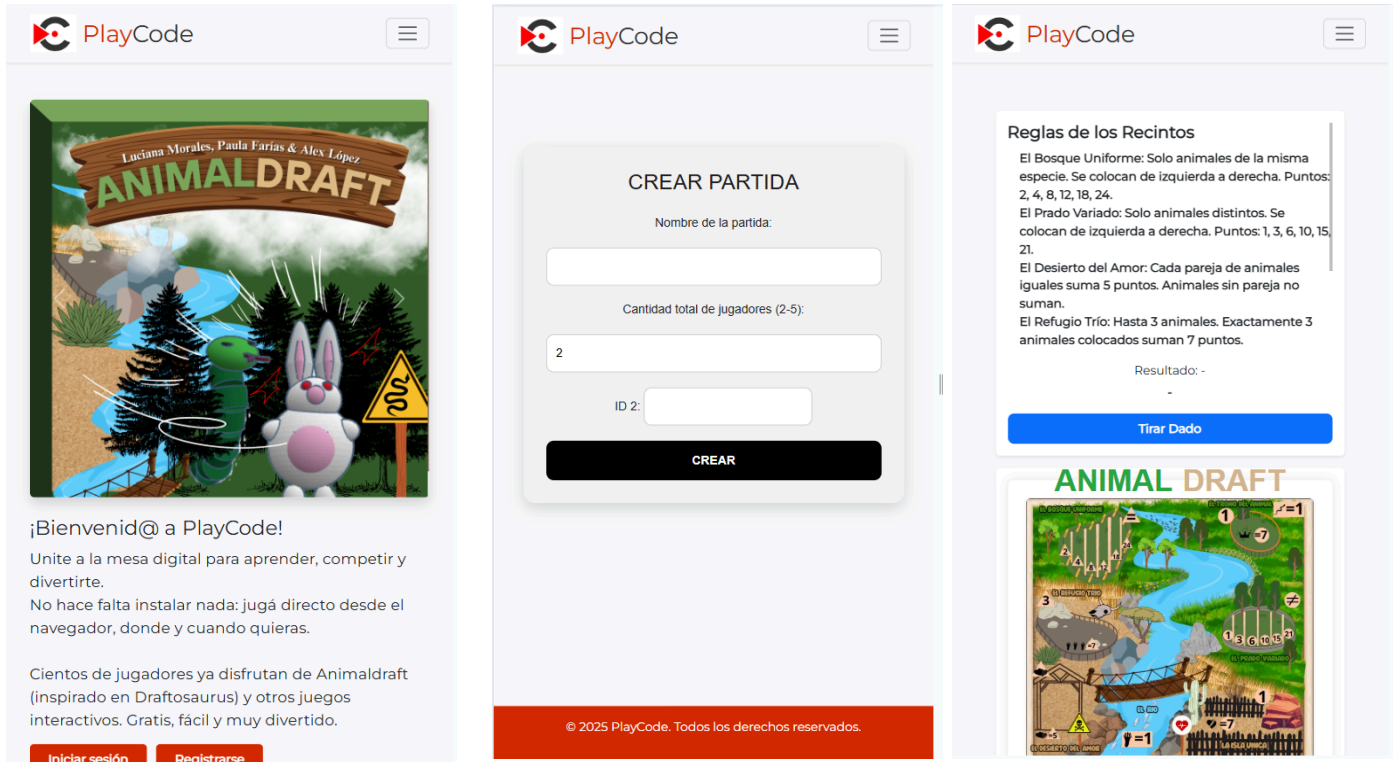




PlayCode

10 de Noviembre

Ejemplos de cómo se vería en Tabletas (577px - 992px)





PlayCode

10 de Noviembre

Ejemplos de cómo se vería en Escritorio (≥993px)

Inicio

Jugar

Panel

Mi perfil

Español

PlayCode

Inicio

Jugar

Panel

Mi perfil

Español

PlayCode

Reglas de los Recintos

El Bosque Uniforme: Solo animales de la misma especie. Se colocan de izquierda a derecha. Puntos: 2, 4, 8, 12, 18, 24.

El Prado Variado: Solo animales distintos. Se colocan de izquierda a derecha. Puntos: 1, 3, 6, 10, 15, 21.

El Desierto del Amor: Cada

Resultado: -

Tirar Dado

ANIMAL DRAFT

Ronda 1

Turno de: Luciana

Trackeo

Animal

Tortuga

Recinto:

El Bosque Uniforme

Registrar Movimiento

Finalizar Partida

Registro de Movimientos

Jugador	Animal	Recinto	Ronda
---------	--------	---------	-------

Puntuación

Jugador	Puntos	Fichas
Luciana	0	6

Inicio

Jugar

Panel

Mi perfil

Español

PlayCode

Luciana

ID: 1

luu.morales@gmail.com

10

Partidas jugadas

10

Partidas ganadas

Mis Partidas

Nombre de la partida	Puntos obtenidos	Ganador	Fecha
ok	0	-	-
ok2	2	Luciana	08/11/2025


S.I.G.P.D. - ISBO 3°MI

14



PlayCode

10 de Noviembre

 PlayCode

InicioJugarPanelMi perfilEspañol

Posición

1°
Luciana
2 Puntos


2°
Alex
0 Puntos

3°
Jugador
0 Puntos

Partidas Jugadas

Partidas Ganadas	Luciana
Fecha	08/11/2025 20:26
Jugadores	2
Puntuación Máxima	2 Puntos
Duración	0 min 37 s

Volver

 PlayCode

InicioJugarPanelMi perfilEspañol

CREAR PARTIDA

Nombre de la partida:

Cantidad total de jugadores (2-5):

2

ID 2:

CREAR

© 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



Ranking

PK: id_jugador

FK: id_jugador

Atributos: partidas_jugadas, partidas_ganadas, puntos_totales,
promedio_puntos

Esquema relacional normalizado (nueva base de datos)

jugadores

PK: id

Atributos: nombre, correo, contrasena

partidas

PK: id

Atributos: creador_id (FK → jugadores.id), fecha_inicio, fecha_fin, ganador_id
(FK → jugadores.id)

jugador_partida

PK: id

FK1: jugador_id → jugadores.id

FK2: partida_id → partidas.id

Atributos: puntuacion

ranking

PK/FK: jugador_id → jugadores.id

Atributos: partidas_jugadas, partidas_ganadas, puntos_totales,
promedio_puntos



3FN - Base de Datos Inicial

1FN

<i>Tabla</i>	<i>Cumple con la 1FN</i>
Jugador	Todos los atributos son atómicos (nombre, correo, contraseña, tipo_usuario), tiene PK id_jugador
Partida	Todos los valores son simples, sin listas, PK id_partida
Jugador_Partida	Los campos son atómicos (una sola puntuación por jugador/partida), PK id_jugador_partida
Animal	Campos simples (nombre, color)
Recinto	La descripción y restricción son textos pero no multivaluados
Recinto_Jugador	Campos simples, FK bien definidos
Colocacion	Cada colocación se refiere a un solo animal, recinto y jugador
Dado	Valor automatico
Ranking	Campos numéricos atómicos, PK id_jugador

2FN

<i>Tabla</i>	<i>Cumple con la 1FN</i>
--------------	--------------------------



Jugador	PK simple (id_jugador), dependencias completas
Partida	PK simple
Jugador_Partida	PK simple (id_jugador_partida). No hay dependencia parcial
Animal	PK simple
Recinto	PK simple
Recinto_Jugador	PK simple
Colocacion	PK simple
Dado	PK simple
Ranking	PK simple, campos calculados dependen totalmente del jugador

3FN

- Jugador (PK id_jugador, nombre, correo, contrasena, tipo_usuario)
- Partida (PK id_partida, fecha_inicio, estado)
- Jugador_Partida (PK id_jugador_partida, FK id_jugador, FK id_partida, puntuacion, es_ganador)
- Animal (PK id_animal, nombre, color)
- Recinto (PK id_recinto, nombre, descripcion, restriccion)
- Recinto_Jugador (PK id_recinto_jugador, FK id_jugador_partida, FK id_recinto)
- Colocacion (PK id_colocacion, FK id_jugador_partida, FK id_animal, FK id_recinto_jugador, cantidad)
- Dado (PK id_dado, FK id_partida, turno, valor)



- Ranking (PK id_jugador, partidas_jugadas, partidas_ganadas, puntos_totales, promedio_puntos)

3FN - Base de Datos Final

1FN

<i>Tabla</i>	<i>Cumple con la 1FN</i>
users	Todos los atributos son atómicos (name, email, password, rol, etc.), tiene PK simple id.
password_reset_tokens	Campos simples (email, token, created_at), PK email.
sessions	Campos atómicos (id, ip_address, user_agent, payload, etc.), PK id.
partidas	Atributos simples (nombre, fecha_inicio, fecha_fin), PK id.
partida_jugador	Cada fila representa un jugador por partida con una sola puntuación; PK id.
turnos	Cada registro representa un turno único con valores simples; PK id.
movimientos	Todos los campos son atómicos (animal, recinto, puntos, etc.), PK id.
ranking	Campos numéricos simples, PK jugador_id.

2FN

<i>Tabla</i>	<i>Cumple con la 1FN</i>
--------------	--------------------------



users	PK simple (id), todos los atributos dependen completamente de la clave.
password_reset_tokens	PK simple (email), dependencias completas respecto a la clave.
sessions	PK simple (id), todos los atributos dependen del identificador de sesión.
partidas	PK simple (id), dependencias completas; creador_id y ganador_id son claves foráneas, no generan dependencias parciales.
partida_jugador	PK simple (id), los atributos dependen completamente del registro; sin dependencias parciales.
turnos	PK simple (id), todos los atributos dependen completamente de la clave.
movimientos	PK simple (id), los atributos dependen completamente de la clave; sin dependencias parciales.
ranking	PK simple (jugador_id), los campos calculados dependen completamente del jugador.



3FN

- users (PK id, name, email, email_verified_at, password, rol, remember_token, created_at, updated_at)
- password_reset_tokens (PK email, token, created_at)
- sessions (PK id, FK user_id, ip_address, user_agent, payload, last_activity)
- partida (PK id, FK creador_id, FK ganador_id, nombre, fecha_inicio, fecha_fin, created_at, updated_at)
- partida_jugador (PK id, FK partida_id, FK jugador_id, turno, puntuacion, created_at, updated_at)
- turnos (PK id, FK partida_id, FK jugador_id, orden, created_at, updated_at)
- movimientos (PK id, FK partida_id, FK jugador_id, ronda, animal, recinto, puntos, created_at, updated_at)

ranking (PK jugador_id, partidas_jugadas, partidas_ganadas, puntos_totales, promedio_puntos)

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 implementan

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.



PlayCode

10 de Noviembre

No se permite puntuación negativa; mínimo 0 puntos

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



Configuración del Entorno de Desarrollo

Esta sección detalla cómo preparar un entorno local completo para ejecutar el proyecto ResponDi2, basado en Laravel, utilizando XAMPP, Composer, Node.js y Visual Studio Code. El objetivo es garantizar que todos los integrantes del equipo puedan replicar el entorno de manera uniforme y sin errores de compatibilidad.

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 de forma rápida y sencilla.

Para instalarlo, primero se debe descargar el instalador desde la página oficial, cuyo enlace se encuentra disponible en los anexos de esta documentación. Una vez descargado, se debe ejecutar el archivo e iniciar el asistente de instalación, seleccionando los servicios Apache y MySQL, que son los requeridos por el proyecto.

Al finalizar la instalación, se debe abrir el XAMPP Control Panel y arrancar los módulos Apache y MySQL haciendo clic en el botón Start. Cuando ambos servicios estén activos, el fondo de los módulos se mostrará en color verde, lo cual indica que están funcionando correctamente. Para verificar el funcionamiento de Apache, se puede abrir el navegador y acceder a la dirección <http://localhost/>. Luego, desde el panel de XAMPP, se puede ingresar a phpMyAdmin para comprobar que MySQL también esté operativo.

Es fundamental mantener XAMPP abierto y con los servicios iniciados durante el desarrollo, ya que Laravel y la base de datos dependen del correcto funcionamiento de Apache y MySQL.



Instalación de herramientas adicionales

Además de XAMPP, el proyecto requiere la instalación de otras herramientas necesarias para ejecutar Laravel de manera adecuada.

Composer es el gestor de dependencias de PHP que permite instalar los paquetes requeridos por Laravel. Puede descargarse desde <https://getcomposer.org/> y debe instalarse siguiendo los pasos por defecto. Una vez completada la instalación, se puede verificar su correcto funcionamiento ejecutando el comando `composer -V` en la terminal.

Asimismo, se necesita instalar Node.js y npm, ya que Laravel utiliza Vite para la compilación de los recursos del frontend (CSS, JS, etc.). Ambas herramientas pueden descargarse desde <https://nodejs.org/> (se recomienda la versión LTS) y verificarse mediante los comandos `node -v` y `npm -v`.

Por último, se requiere Git, una herramienta de control de versiones necesaria para clonar el repositorio del proyecto. Puede descargarse desde <https://git-scm.com/downloads> y comprobarse su instalación ejecutando el comando `git --version`.

Instalación de Visual Studio Code y extensiones

El editor de código recomendado para trabajar en el proyecto es Visual Studio Code (VS Code), el cual puede descargarse desde <https://code.visualstudio.com/>. Este entorno de desarrollo ofrece gran compatibilidad con PHP y Laravel, además de extensiones que facilitan la programación.

Se recomienda instalar las siguientes extensiones: PHP Intelephense, que mejora el autocompletado y la navegación en archivos PHP; Laravel Blade Snippets, que proporciona atajos y resaltado de sintaxis para plantillas Blade; Prettier - Code Formatter, que permite mantener un formato de código limpio y uniforme; y Live Server, una herramienta opcional útil para previsualizar archivos HTML o JavaScript de manera rápida.



Dado que XAMPP ya incluye PHP, solo es necesario configurar VS Code para que lo reconozca. Para hacerlo, se debe ir a *Settings* → *Extensions* → *PHP* y agregar la ruta del ejecutable, que por defecto se encuentra en `C:\xampp\php\php.exe`. Finalmente, se debe verificar que PHP funcione correctamente ejecutando el comando `php -v` en la terminal.

Clonación del proyecto y configuración inicial

Con el entorno preparado, se procede a clonar el proyecto ResponDi2 desde su repositorio remoto. Para ello, se abre la terminal y se ejecutan los siguientes comandos:

1. `git clone https://github.com/playcodecompany/Segunda_Entrega1`
2. `cd Segunda_Entrega1`
3. `git fetch origin develop`
4. `git checkout develop`

Estos comandos descargan el código fuente del proyecto y cambian a la rama de desarrollo. Una vez dentro de la carpeta del proyecto, se deben instalar las dependencias necesarias para su funcionamiento.

Las dependencias de PHP se instalan con el comando `composer install`, mientras que las dependencias de Node.js se instalan ejecutando `npm install`.

Posteriormente, se debe configurar el archivo de entorno `.env`, el cual contiene las variables de configuración del proyecto, como la conexión a la base de datos. Para crearlo, se debe copiar el archivo de ejemplo mediante el comando `cp .env.example .env` y luego editarlo con los valores correspondientes:



PlayCode

10 de Noviembre

DB_CONNECTION=mysql

DB_HOST=127.0.0.1

DB_PORT=3306

DB_DATABASE=playcodee

DB_USERNAME=root

DB_PASSWORD=

Una vez configurado, se genera la clave de la aplicación con el comando `php artisan key:generate`, que garantiza la seguridad del sistema. Luego, desde phpMyAdmin, se crea la base de datos con el nombre `playcodee`, y se ejecutan las migraciones con `php artisan migrate` para crear las tablas necesarias en la base de datos.

Cómo correr el prototipo

Para ejecutar el prototipo, se debe abrir la terminal en la carpeta raíz del proyecto Laravel. En primer lugar, se ejecuta el comando `composer run dev`, que se encarga de compilar los recursos del frontend utilizando Vite. A continuación, se levanta el servidor local de Laravel con `php artisan serve`.

Una vez iniciado, la consola mostrará una URL (por defecto `http://127.0.0.1:8000`), la cual se debe abrir en el navegador. Desde allí se podrá navegar por las diferentes vistas del sistema, probar los formularios y verificar el correcto funcionamiento de la interfaz del juego y del servidor local.



Control de Versiones

El control de versiones del proyecto se realizó mediante el uso de Git y GitHub, aplicando el modelo de trabajo GitFlow para mantener una estructura organizada y facilitar la colaboración entre los integrantes del equipo. El repositorio está estructurado 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 recibe los cambios desde develop cuando se libera una versión lista para producción. La rama develop cumple la función de espacio de integración, donde se reúnen y prueban las nuevas funcionalidades antes de su incorporación definitiva.

Por su parte, las ramas feature se crean a partir de develop para el desarrollo de funcionalidades específicas, tales como *feature/views* para las vistas y layouts o *feature/forms* para los formularios de trackeo. Sin embargo, en este proyecto no se implementó el uso de ramas feature, ya que no todos los integrantes trabajaron en la misma materia ni de forma simultánea sobre el código. Al ser un desarrollo gestionado principalmente por una sola persona, no existía riesgo de conflictos o interferencias entre ramas. Por este motivo, se optó por realizar los commits y actualizaciones directamente sobre la rama develop, manteniendo igualmente un control organizado y funcional del repositorio.

Este flujo de trabajo garantiza un desarrollo limpio, controlado y con un historial de commits bien estructurado, facilitando la trazabilidad y el mantenimiento del proyecto.



Archivo README

El archivo README.md del repositorio contiene la información esencial para comprender y utilizar el proyecto. Incluye el nombre del proyecto, la lista de integrantes, el enlace al repositorio de documentación, las instrucciones necesarias para la instalación y configuración del entorno de desarrollo, así como la descripción de las vistas, layouts y flujo de trabajo con GitFlow. Su objetivo principal es servir como una guía rápida y práctica para cualquier usuario o desarrollador que desee ejecutar o colaborar en el proyecto, brindando una visión general clara de su estructura y funcionamiento.

Conclusión

En esta entrega se logró establecer una base sólida para el proyecto, utilizando el framework Laravel como eje principal del desarrollo. Se organizó correctamente la estructura del sistema, incluyendo las vistas, los layouts y las carpetas con los archivos CSS y JS, garantizando un código limpio y ordenado. Además, se configuró el entorno de desarrollo completo y se aplicó un flujo de trabajo con GitFlow, adaptado a las necesidades del equipo, lo que permitió mantener una metodología estructurada y eficiente.

Durante esta fase se desarrollaron las pantallas principales del prototipo: la pantalla de bienvenida, el tablero del juego, el formulario de trackeo y la pantalla de resultados, completando así la interfaz visual del proyecto. Estas implementaciones sentaron las bases para las próximas etapas, donde se incorporará la lógica del juego, incluyendo el cálculo de puntajes, la interacción entre animales y recintos, y la gestión de turnos y del dado.



PlayCode

10 de Noviembre

La experiencia de trabajo fue muy positiva, ya que permitió reforzar conocimientos técnicos en Laravel, Composer, Node.js y el control de versiones con Git/GitHub. Asimismo, se afianzaron las habilidades de organización y desarrollo individual, adaptando las buenas prácticas del flujo GitFlow a un contexto de trabajo donde un solo integrante gestiona el código, logrando así un proceso de desarrollo ordenado, documentado y profesional.

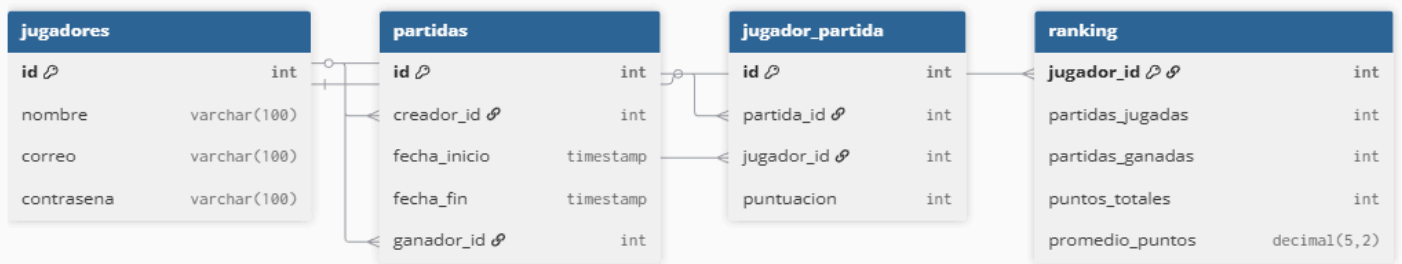
Durante el proceso de control de versiones se cometió un pequeño error en la nomenclatura de los commits, ya que fueron redactados en español cuando debían realizarse en inglés, debido a que nos olvidamos al momento de registrar los cambios.



Anexos

Anexo I. DER

Realizamos el DER con la página dbdiagram.io





Anexo II. Diagramas UML

Diagrama de Secuencias hecho en PlantUML

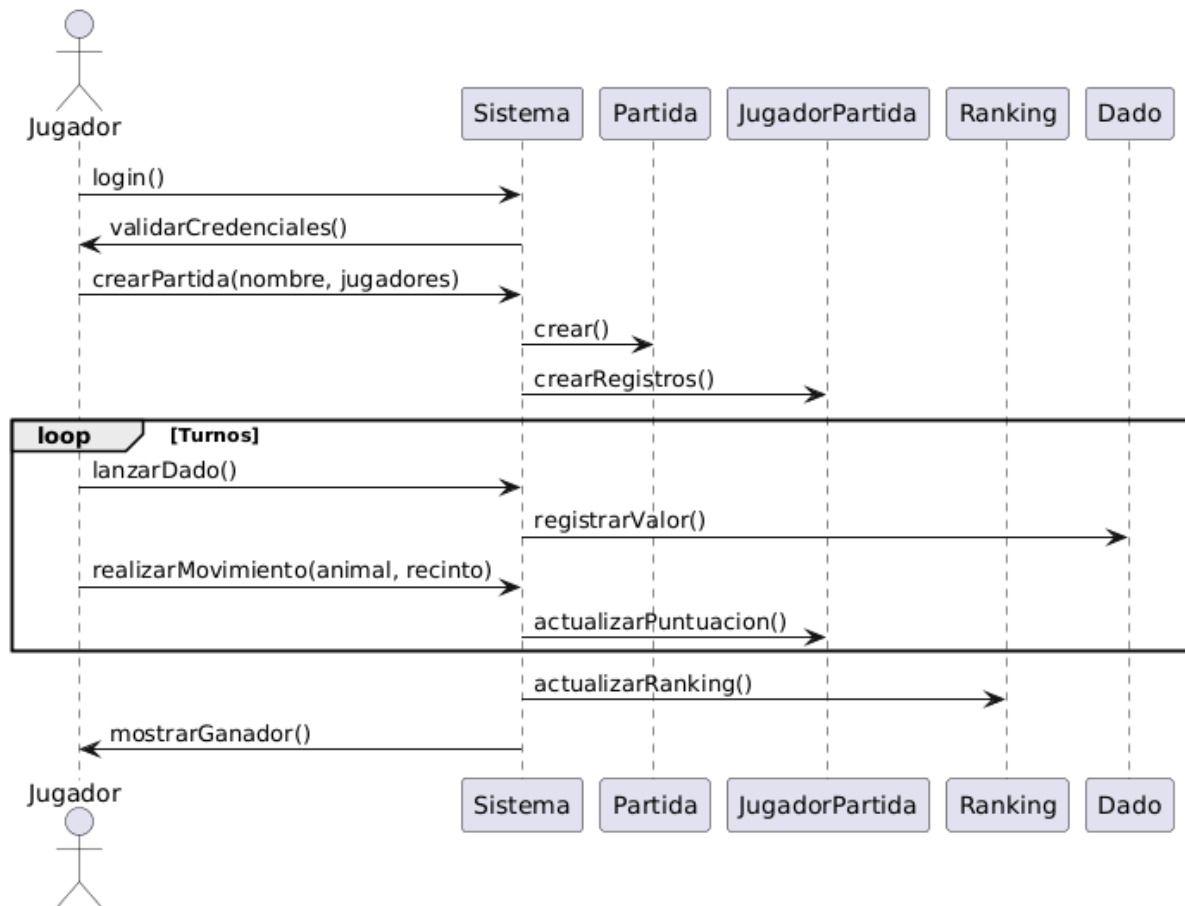




Diagrama de Clases hecho en PlantUML

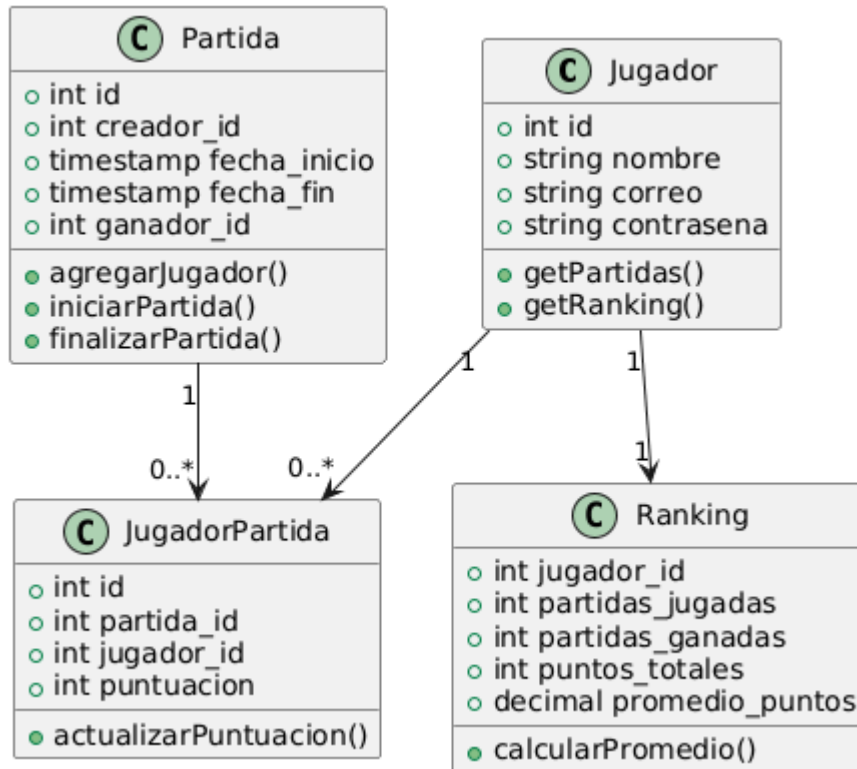
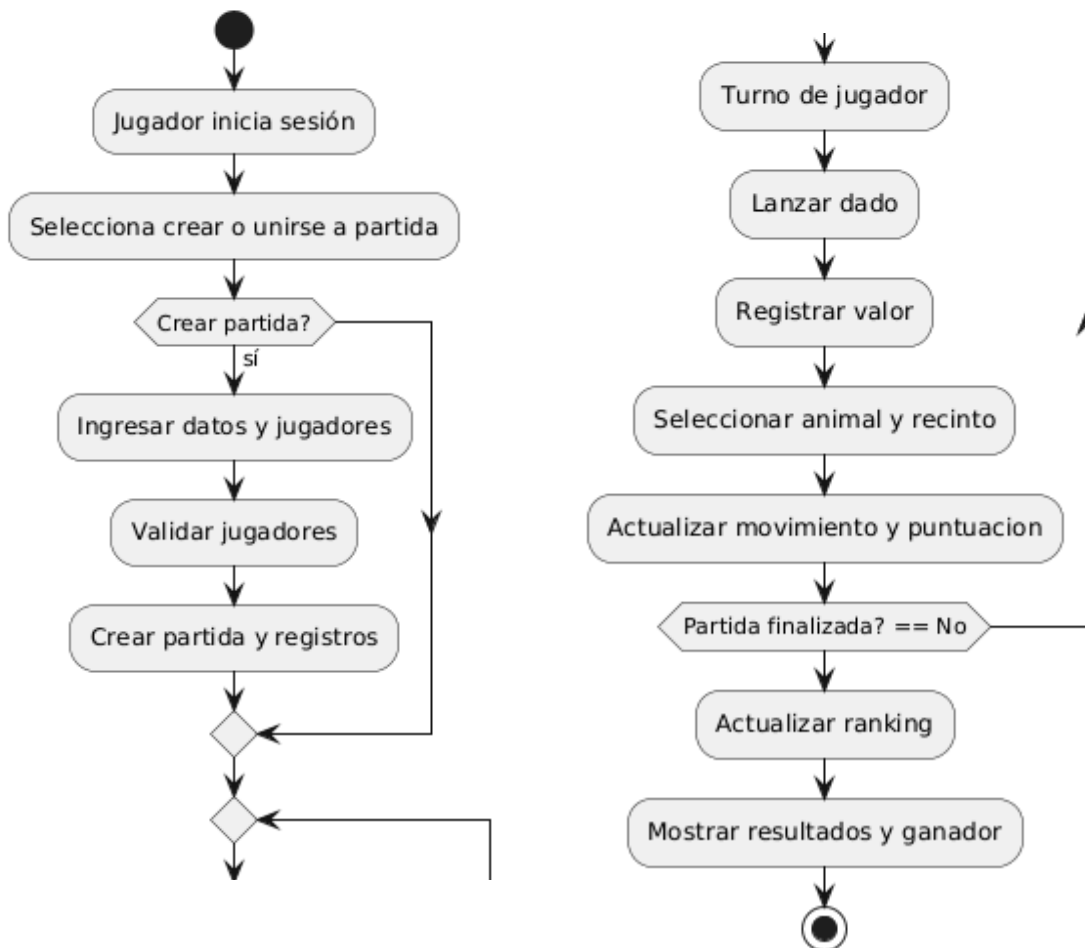




Diagrama de Actividad en PlantUML





Anexo III. Base de Datos

Código de la base de datos

```
CREATE DATABASE ProyectoFinal;  
USE ProyectoFinal;
```

-- Jugadores locales

```
CREATE TABLE jugadores (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    correo VARCHAR(100) NOT NULL,  
    contraseña VARCHAR(100) NOT NULL  
);
```

-- Partidas creadas en la PC

```
CREATE TABLE partidas (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    creador_id INT, -- jugador que creó la partida  
    fecha_inicio TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
    fecha_fin TIMESTAMP NULL,  
    ganador_id INT NULL,  
    FOREIGN KEY (creador_id) REFERENCES jugadores(id),  
    FOREIGN KEY (ganador_id) REFERENCES jugadores(id)  
);
```

-- Jugadores que participan en cada partida

```
CREATE TABLE jugador_partida (  
    id INT AUTO_INCREMENT PRIMARY KEY,  
    partida_id INT,  
    jugador_id INT,  
    puntuacion INT DEFAULT 0,  
    FOREIGN KEY (partida_id) REFERENCES partidas(id),  
    FOREIGN KEY (jugador_id) REFERENCES jugadores(id)  
);
```

-- Ranking acumulado

```
CREATE TABLE ranking (  
    jugador_id INT PRIMARY KEY,  
    partidas_jugadas INT DEFAULT 0,  
    partidas_ganadas INT DEFAULT 0,
```



PlayCode

10 de Noviembre

```
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 (jugador_id) REFERENCES jugadores(id)  
);
```



Anexo IV . 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

10 de Noviembre

Anexo V . Repositorio de GitHub

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

Link del repositorio con documentación más detallada →

https://github.com/playcodecompany/Documentacion_Proyecto/tree/main



PlayCode

10 de Noviembre



PlayCode

10 de Noviembre

Hoja testigo

Montevideo 10 de noviembre del 2025

ACUSE DE RECIBO PARA ENTREGAS DE PROYECTO

Corresponde a ENTREGA N° 3

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-Coordina dor	López	Alex	5.646776-5	aleexpias@gmail.com
Integrante N°1	Morales	Luciana	5.700540-5	luuumoralees@gmail.com

Por contacto al correo: companyplaycode@gmail.com

Firmas:

COORDINADOR

SUBCOORDINADOR

INTEGRANTE 1



PlayCode

10 de Noviembre



FARÍAS, PAULA

COORDINADOR



LÓPEZ, ALEX

SUBCOORDINADOR



MORALES, LUCIANA

INTEGRANTE 1