

# Checkter - Aplicación de Ajedrez



*Por Martín Carballo Otero,*

*como proyecto de fin de ciclo en Aula Nosa.*

# Índice

1. Introducción.....	5
Introducción al Proyecto.....	5
Introducción a la Empresa.....	5
Estructura de la empresa.....	6
2. Tecnologías empleadas.....	8
Interfaz de usuario (Frontend).....	8
Backend.....	9
Inteligencia Artificial.....	9
Almacenamiento de datos.....	10
Documentación.....	11
3. Arquitectura.....	12
Definición de la arquitectura de micro-servicios:.....	12
Ventajas generales de la arquitectura de micro-servicios:.....	12
Extensión en el mercado:.....	12
Causas específicas de la selección de arquitectura de micro-servicios:.....	13
Implementación de la arquitectura de micro-servicios:.....	13
4. Análisis de Requisitos.....	14
Historias de Usuarios.....	14
Como planteamos cumplir los requisitos.....	15
Metodología de desarrollo.....	15
Fases temporales del desarrollo.....	16
5. Diseño Estático.....	17
Diagrama de clases UML.....	17
Diagrama de secuencia de un caso de uso.....	21

6. Modelo de Datos.....	23
Diagrama de ER.....	23
Explicación Diagrama ER.....	24
7. Manuales de Instalación y Usuario.....	27
Manual de Instalación.....	27
Manual de Usuario.....	35
8. Viabilidad.....	44
Estudio de mercado.....	44
Análisis económico.....	44
Estudio de riesgos.....	46
DAFO.....	48
9. Calidad.....	49
Pruebas Unitarias (Unit Testing).....	49
Pruebas de Usabilidad (Usability Testing).....	50
Pruebas de Rendimiento (Performance Testing).....	50
10. Conclusiones.....	51
Estado actual.....	51
Mejoras a futuro.....	52
11. Bibliografía.....	53

# 1. Introducción

## Introducción al Proyecto

En el ámbito del entretenimiento y la estrategia, el ajedrez ha sido el juego predominante a lo largo de los siglos. Su fusión de habilidad táctica, planificación estratégica y desafío mental lo convierte en un juego fascinante para una amplia gama de jugadores, independientemente de su edad o nivel de destreza. A medida que la tecnología ha avanzado, el ajedrez ha encontrado un nuevo espacio en el ámbito digital, donde la innovación y la accesibilidad convergen para elevar este juego clásico a nuevos niveles.

En este proyecto, asumimos la tarea de desarrollar una aplicación de ajedrez que no solo conserve la esencia clásica del juego, sino que también aproveche las capacidades tecnológicas para ofrecer una experiencia única, envolvente y refrescante desde la comodidad de dispositivos móviles u ordenadores.

Nuestra aplicación de ajedrez ofrece diversas opciones para los entusiastas del juego. Además de partidas locales amistosas, los jugadores pueden desafiar a amigos en línea y enfrentarse a la máquina en partidas individuales. En este documento, exploraremos el proceso de desarrollo de nuestra aplicación de ajedrez, desde la concepción de la idea hasta la implementación de características clave.

## Introducción a la Empresa

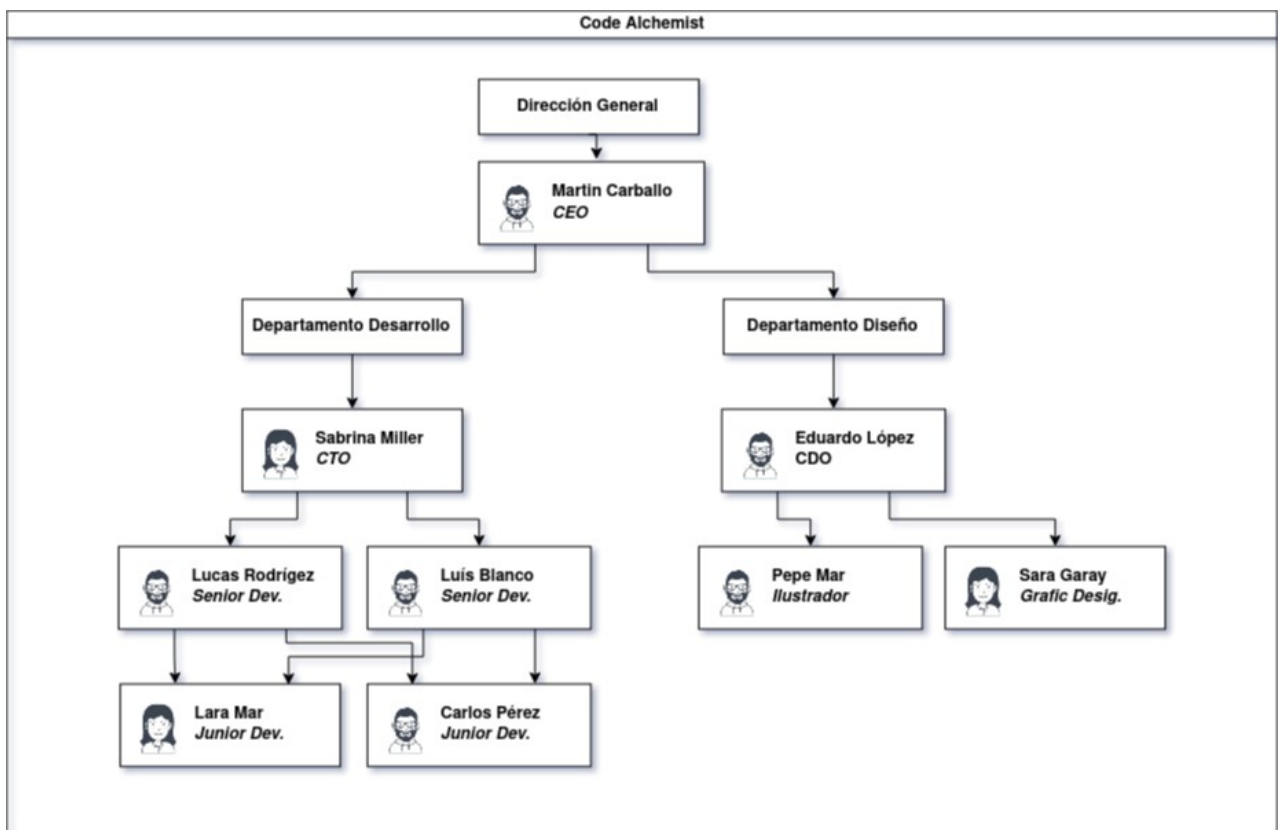
**Descripción:** Code Alchemists es una empresa especializada en el desarrollo de software. Constituida como una Sociedad de Responsabilidad Limitada (SRL), cuenta ya con varias aplicaciones publicadas para dispositivos móviles y ordenadores. En este caso la empresa plantea la publicación de una aplicación nueva centrada en el ajedrez.

**Sector:** En el contexto del mercado tecnológico, el sector del software ha experimentado un crecimiento exponencial, generando una diversificación notable en el ámbito de las aplicaciones móviles. Dentro de este panorama, los juegos de estrategia han surgido como una categoría prominente, atrayendo a una amplia audiencia con propuestas de entretenimiento interactivas y desafiantes.

**Mercado Objetivo:** Las audiencias que más tiempo dedican en juegos digitales tienden a ser personas de entre 25 y 34 años (29,5%), jóvenes de 16 a 24 años (28,3%) y de 35 a 44 años (23,1%de toda la audiencia de juegos móviles).

## Estructura de la empresa

A continuación se trata la estructura de la empresa haciendo uso de un organigrama, del cual se desglosaran sus elementos clave.



*Figura 1, organigrama de la empresa.*

La dirección general de de Code Alchemist viene de la mano de Martín Carballo, el CEO de la empresa. El CEO es responsable de la gestión general y la toma de decisiones estratégicas para la compañía.

## **Departamento de Desarrollo**

Este departamento está dirigido por Sabrina Miller, quien ocupa el puesto de CTO (Chief Technology Officer). Sus funciones incluyen la supervisión de los proyectos de desarrollo de software, la implementación de nuevas tecnologías y la gestión del equipo técnico.

Dentro del Departamento de Desarrollo, se encuentran:

- Lucas Rodríguez y Luís Blanco (Senior Dev.): Diseñan la arquitectura del software, revisan código y lideran técnicamente, mentorando a los desarrolladores junior.
- Lara Mar y Carlos Pérez (Junior Dev.): Realizan tareas de codificación y desarrollo bajo supervisión, aprendiendo y contribuyendo a proyectos asignados.

## **Departamento de Diseño**

Este departamento es liderado por Eduardo López, el CDO (Chief Design Officer), quien se encarga de la supervisión de todos los aspectos visuales y de diseño de los productos de la empresa.

En el Departamento de Diseño están:

- Pepe Mar (Ilustrador): Diseña ilustraciones y gráficos para productos y materiales de marketing.
- Sara Garay (Gráfico Desig.): Crea elementos visuales y gráficos, manteniendo una identidad visual coherente.

## 2. Tecnologías empleadas

Nuestra aplicación se estructura en diversas capas y funcionalidades, cada una aprovechando tecnologías específicas por distintas razones. En este análisis, exploraremos las decisiones detrás de cada aspecto de la aplicación y los motivos que impulsaron su selección. Desde la capa de interfaz de usuario hasta las funciones de inteligencia artificial y conexión en línea.

### Interfaz de usuario (Frontend)

La interfaz con la que el jugador interactúa es un elemento clave en la aplicación. Debe ser capaz de implementar gráficamente todas las funcionalidades que el usuario va a utilizar, así como parte de la lógica que la partida de ajedrez necesita. Además, la tecnología empleada debe poder utilizarse tanto en dispositivos móviles como en ordenadores, ya sea en forma de aplicación de escritorio o en formato de navegador.

Siguiendo esta lista de requisitos el lenguaje de programación a emplear ha sido Dart, haciendo uso del framework Flutter y con ayuda del entorno de desarrollo Visual Studio Code.



### Causas de la selección:

Flutter permite el desarrollo multiplataforma con rendimiento nativo y rapidez gracias a Hot Reload. VS Code proporciona un entorno de desarrollo ligero y altamente extensible. La comunidad activa y el soporte de Google aseguran un proceso de desarrollo fluido y actualizaciones constantes. Además, la documentación exhaustiva y la accesibilidad de código abierto hacen que esta combinación sea altamente competitiva en términos de costo y eficiencia.

Comparado con otras tecnologías como React Native, Xamarin o SwiftUI, Dart + Flutter + VS Code ofrece un buen equilibrio entre rendimiento, flexibilidad y productividad durante el desarrollo complementándose entre ellas.

## Backend

El backend de la aplicación es responsable de manejar la lógica de negocio, el almacenamiento de datos, la gestión de usuarios y la comunicación entre el cliente y el servidor. En este caso hemos optado por una arquitectura basada en micro-servicios empleando una API.

Para el desarrollo de esta API se ha hecho uso de Python, con ayuda de la librería de FlaskAPI, trabajando en el IDE, PyCharm. Las pruebas se realizaron con ayuda de Insomnia.



### Causas de la selección:

La selección de Python junto a Flask API se ve justificada por la necesidad de un desarrollo ágil, ambas herramientas destacan con su simplicidad sobre otras como podrían ser Java con Spring haciendo que los tiempos de desarrollo se reduzcan. La selección de PyCharm sobre Visual Studio esta dada por la facilidad que aporta PyCharm a la hora de debugear el código. La selección de Insomnia para el testing de la API sobre otras opciones como Postman se debe al igual que en la selección de Python o Flask en el deseo de hacer uso de una herramienta mas simple pero funcional.

## Inteligencia Artificial

Para poder implementar partidas contra la maquina de diversas dificultades a sido necesitar recurrir a herramientas de inteligencia artificial. El modulo seleccionado a sido Stockfish, un modelo especializado en partidas de ajedrez.





### **Causas de la selección:**

Stockfish es un modelo ya funcional capaz de predecir con alta precisión los mejor movimientos a realizar. Además permite ajustar sus dificultades en base al sistema ELO de forma sencilla. Cuenta con una librería dedicada en Python que permite su fácil implementación. En general es un modelo potente y ya preparado para ser empleado, mejor opción que desarrollar un modelo propio.

### **Almacenamiento de datos**

El almacenamiento de datos ha sido llevado a cabo haciendo uso de SQL Server.



### **Causas de la selección:**

Seleccionar SQL Server es ideal para organizaciones que buscan una integración sólida con el ecosistema Microsoft, seguridad avanzada, alto rendimiento y escalabilidad.

Comparado con otras tecnologías, SQL Server ofrece mejor rendimiento y soporte comercial frente a MySQL, más características empresariales que PostgreSQL, es más económico y fácil de usar que Oracle, y es más adecuado para datos estructurados y transaccionales que MongoDB, que es más flexible para datos no estructurados.

## Documentación

Las herramientas empleadas para el desarrollo de la documentación han sido LibreOffice Writer para el desarrollo y edición del texto y Photopea para la edición de imágenes.



### Causas de la selección:

La selección de estas herramientas sobre otras como podrían ser Microsoft Word o Adobe Photoshop viene dada por la preferencia del desarrollador por la herramientas de código abierto debido a su abaratamiento de costes, además de que el mismo desarrollador contaba con amplia experiencia previa trabajando con ellas.

### 3. Arquitectura

Para el desarrollo del proyecto se ha optado por una arquitectura de micro-servicios.

#### **Definición de la arquitectura de micro-servicios:**

La arquitectura de micro servicios es un método de desarrollo de aplicaciones software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. En ella, cada micro servicio es un código que puede estar en un lenguaje de programación diferente, y que desempeña una función específica. Los micro servicios se comunican entre sí a través de APIs, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.

#### **Ventajas generales de la arquitectura de micro-servicios:**

- **Escalabilidad:** Cada micro servicio puede ser escalado independientemente según la demanda, lo que permite una gestión de recursos más eficiente.
- **Despliegue independiente:** Los servicios pueden ser desplegados y actualizados de manera independiente sin afectar a todo el sistema.
- **Focalización de equipos:** Permite que diferentes equipos trabajen en diferentes micro servicios de forma autónoma, lo que acelera el desarrollo y la innovación.
- **Flexibilidad tecnológica:** Se pueden utilizar diferentes tecnologías y lenguajes de programación para diferentes servicios, optimizando cada uno para su tarea específica.

#### **Extensión en el mercado:**

La arquitectura de micro servicios es ampliamente adoptada en la industria, especialmente por grandes empresas tecnológicas como Netflix, Amazon y Google, debido a sus ventajas en escalabilidad y mantenimiento.

## Causas específicas de la selección de arquitectura de micro-servicios:

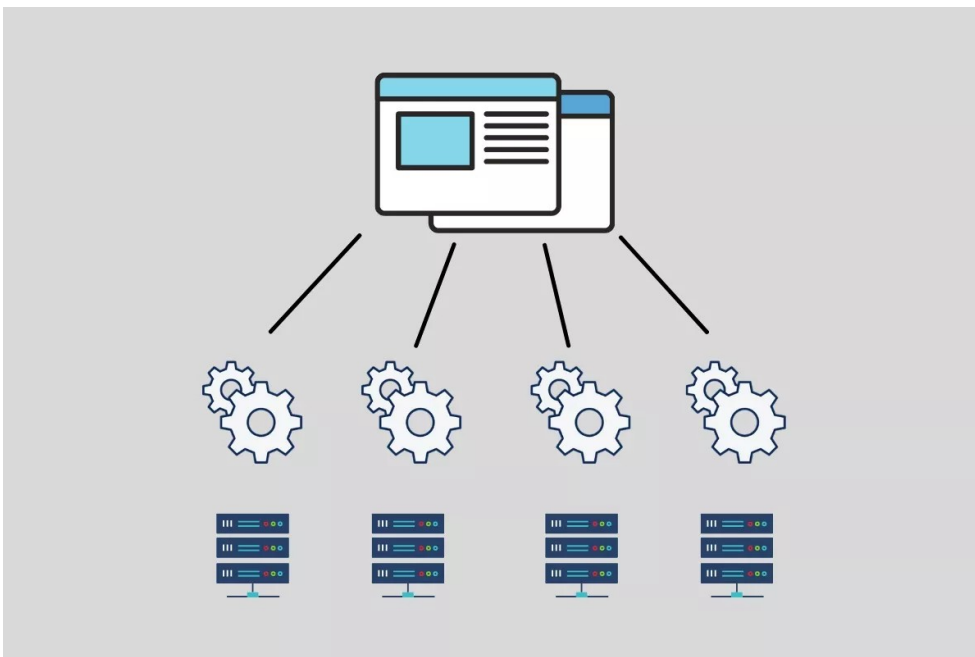
Se ha seleccionado la arquitectura de micro-servicios para el proyecto debido a su capacidad para escalar y desplegar cada componente de manera independiente, optimizando recursos y mejorando la solidez del sistema. Esta arquitectura facilita el desarrollo y mantenimiento al permitir que los servicios trabajen en paralelo y se adapten rápidamente a cambios. Además, la arquitectura de micro-servicios permite atender a varios clientes de forma simultánea de manera que la implementación de servicios multiplayer a futuro sería viable.

## Implementación de la arquitectura de micro-servicios:

Para la implementación de la arquitectura de servicios se ha hecho uso de la implementación de una API que habilita los distintos servicios que la aplicación puede consumir en su estado actual.

La API cuenta con el servicio de conexión a base de datos, que le permite el almacenamiento de datos así como el consumo de aquella información que ya se encuentre almacenada así como un servicio que conecta el modelo de inteligencia artificial Stockfish a la aplicación.

En el futuro se plantea que la API albergue más servicios como el control de múltiples usuarios o habilitar la descarga de la apk o exe de la aplicación.



*Figura 2, arquitectura de micro-servicios.*

## 4. Análisis de Requisitos

### Historias de Usuarios


Las historias de usuario son una herramienta muy útil a la hora de definir los requisitos de nuestra aplicación ya que nos dejan trabajar directamente con el feedback de usuarios, entender sus necesidades y crear requisitos en base a ellas. La historia de usuario debe responder a 3 preguntas:

Quien? - Cual es el rol de la persona que plantea la historia.

Que quiere? - Cual es la necesidad o característica que esta solicitando.


Para? - Cual es el objetivo que cumple esa solicitud.

A continuación se mostraran una serie de historias de usuario que se han podido extraer de comentarios realizados a múltiples aplicaciones de ajedrez de la competencia, los comentarios han sido editados para ser mas breves y fáciles de relacionar con las tres preguntas explicadas.

 **User 1**


★★★★★ 11/6/18

Como jugador ocasional de ajedrez, quiero poder jugar partidas locales contra un amigo o familiares en el mismo dispositivo, para disfrutar del juego cara a cara.

 **User 2**


★★★★★ 11/6/18

Como jugador solitario de ajedrez, quiero poder desafiar a una inteligencia artificial de diferentes niveles de dificultad, para mejorar mis habilidades y disfrutar del juego cuando no tengo compañeros disponibles.

 **User 3**


★★★★★ 11/6/18

Como jugador de ajedrez, quiero poder guardar las partidas que juego, para poder revisarlas más tarde, analizar mis movimientos y aprender de mis errores.

 **User 4**


★★★★★ 11/6/18

Como jugador de ajedrez, quiero poder revisar mis partidas anteriores, para analizar mis movimientos, identificar áreas de mejora y aprender de mis errores.

 **User 5**

★★★★★ 11/6/18

Como jugador de ajedrez, quiero poder enfrentarme a oponentes de todo el mundo en partidas online, para disfrutar de la competencia y poner a prueba mis habilidades contra jugadores de diferentes niveles.

 **User 6**

★★★★★ 11/6/18

Como jugador de ajedrez, quiero poder ajustar la dificultad de las partidas contra la IA, para poder desafiarme a mí mismo y mejorar gradualmente mis habilidades sin sentirme abrumado por un oponente demasiado difícil.

A partir de estas historias de usuario se pueden extraer como requisitos prioritarios de la aplicación:

- Implementación de un sistema de juego local y otro en línea.
- Integración de una inteligencia artificial con múltiples niveles de dificultad.
- Capacidad de guardar y revisar partidas que ya han terminado.

## **Como planteamos cumplir los requisitos**

Nuestra aplicación habilita las partidas en local contra otras personas y contra la maquina, además esta en la lista de objetivos llegar a implementar una funcionalidad que habilite el multiplayer en su formato de juego Online con clasificaciones.

Planteamos implementar diferentes configuraciones predefinidas de Stockfish para cada nivel de dificultad, o permitir que los usuarios personalicen la configuración de la AI según sus preferencias.

Agregaremos funcionalidades para guardar el estado de la partida al finalizar y almacenarlo en nuestra base de datos e implementaremos una sección en nuestra aplicación donde los usuarios puedan ver el historial de partidas jugadas, con la posibilidad de cargar y revisar cualquier partida almacenada.

## **Metodología de desarrollo**

El desarrollo de la aplicación se está planteando a un plazo de 3 meses hasta la publicación de la app en su estado inicial, un espacio de tiempo demasiado breve para llevar acabo una metodología ágil, en la cual se debe aportar valor en forma de demos utilizables de forma constante. Para el desarrollo inicial se ha planteado una metodología en cascada ya que se han definido unos requisitos claros y será necesario superar todas las fases planteadas a continuación de forma secuencial hasta el día de publicación de la aplicación. Una vez publicada se planteará una metodología ágil.

## Fases temporales del desarrollo

### Fase 1: Análisis de Requisitos (Semana 1)

- **Contenido:** Reunión inicial para recopilar y documentar todos los requisitos del proyecto, incluidas las funcionalidades básicas del juego de ajedrez, los requisitos de interfaz de usuario y cualquier otra especificación relevante.
- **Objetivos:** Obtener una comprensión clara de lo que se espera de la aplicación y definir el alcance del proyecto.

### Fase 2: Diseño (Semana 2-3)

- **Contenido:** Diseño detallado de la arquitectura de la aplicación, incluidos los componentes del software, la estructura de datos y la interfaz de usuario.
- **Objetivos:** Establecer una base sólida para el desarrollo de la aplicación y garantizar que se cumplan todos los requisitos identificados en la fase de análisis.

### Fase 3: Desarrollo (Semana 4-8)

- **Contenido:** Implementación de la aplicación según las especificaciones y el diseño establecidos en las fases anteriores. Desarrollo de funcionalidades básicas, lógica del juego, interfaz de usuario y cualquier otra característica requerida.
- **Objetivos:** Construir la aplicación de acuerdo con los requisitos y el diseño definidos, asegurando que cumpla con las expectativas del cliente.

### Fase 4: Pruebas (Semana 9-10)

- **Contenido:** Realización de pruebas exhaustivas de la aplicación para identificar y corregir errores.
- **Objetivos:** Garantizar la calidad y la fiabilidad de la aplicación mediante la detección y corrección de errores antes del lanzamiento.

### Fase 5: Preparación Publicación (Semana 11-12)

- **Contenido:** Preparación para el lanzamiento de la aplicación, incluida la instalación en el entorno de producción y la capacitación del personal si es necesario. Además, se establece un plan de mantenimiento para futuras actualizaciones y correcciones de errores.
- **Objetivos:** Llevar la aplicación al mercado y garantizar su funcionamiento continuo a largo plazo mediante el mantenimiento adecuado.

## 5. Diseño Estático

El diseño estático afronta la estructura y la organización de los componentes de la aplicación sin considerar el comportamiento dinámico de estos componentes en tiempo de ejecución. Para tratarlo se hará uso de un diagrama UML que explicará las clases de la aplicación y un ejemplo de un caso mostrado en forma de diagrama de secuencia.

### Diagrama de clases UML

El diagrama UML (Unified Modeling Language) es una representación visual utilizada para especificar, visualizar, construir y documentar los componentes de un sistema de software. En este caso se aplica a las clases de nuestra aplicación de ajedrez.

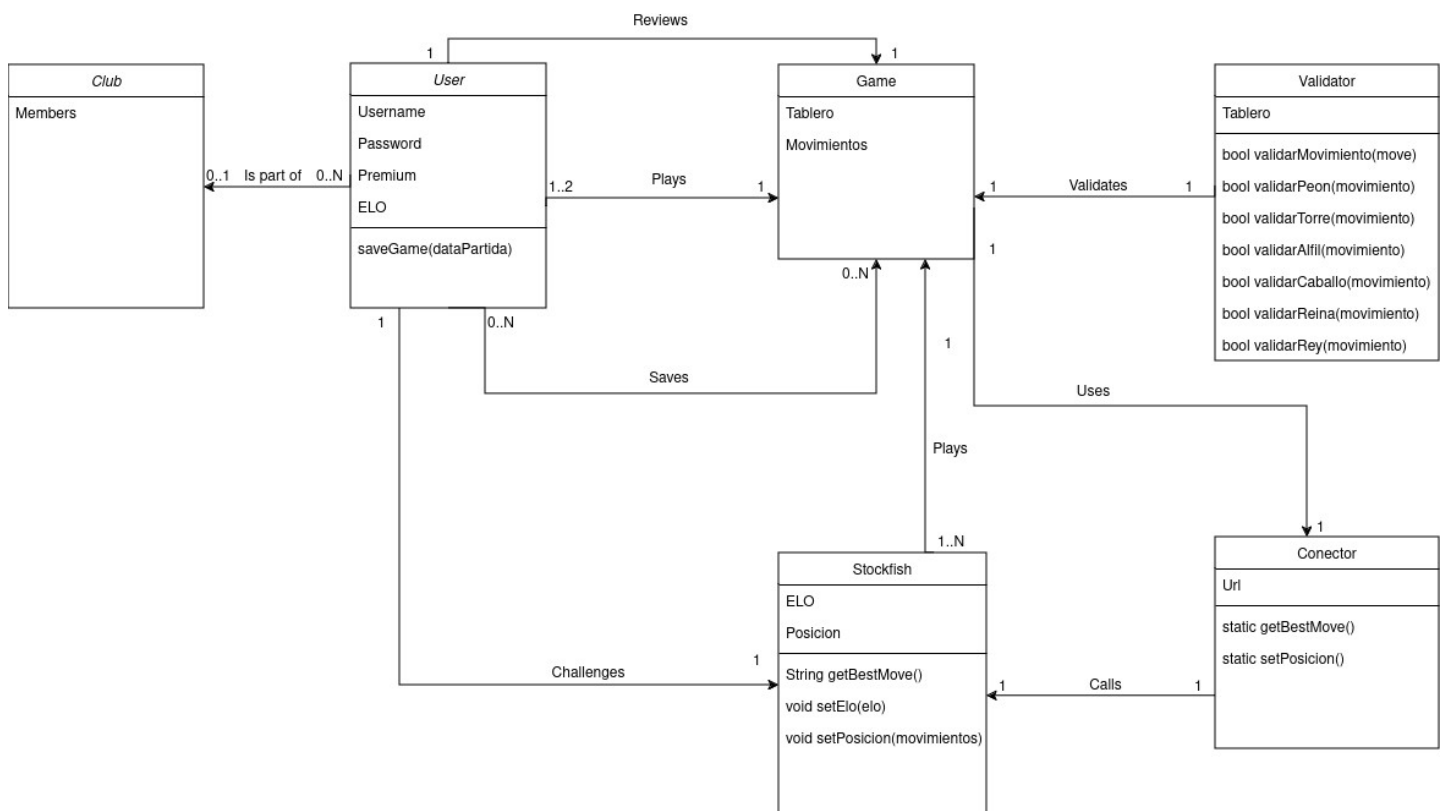


Figura 3, diagrama de clases UML.

A continuación se desglosaran las partes del diagrama en detalle. Tratando la descripción, campos, métodos y relaciones que puedan tener las distintas clases.



## **1. Club:**

Descripción: Almacena la información de los clubes de ajedrez.

Campos:

- Members: Lista de miembros del club.

Relaciones:

- Un club puede tener entre 0 y muchos usuarios (User) como miembros, y un usuario puede ser parte de 0 o 1 club.

## **2. User (Usuario):**

Descripción: Almacena la información de los usuarios.

Campos:

- Username: Nombre de usuario.
- Password: Contraseña.
- Premium: Indica si el usuario tiene una cuenta premium.
- ELO: Puntaje ELO del usuario.

Métodos:

- saveGame(dataPartida): Guarda una partida de ajedrez.

Relaciones:

- Un usuario puede jugar 1 partida simultáneamente.
- Un usuario puede guardar entre 0 y muchas partidas.
- Un usuario puede estar en 0 o 1 club.
- Un usuario puede desafiar a Stockfish.

### 3. Game (Partida):

Descripción: Almacena la información sobre las partidas de ajedrez.

Campos:

- Tablero: Estado del tablero de ajedrez.
- Movimientos: Lista de movimientos realizados en la partida.

Relaciones:

- Una partida es revisada por 1 usuario.
- Una partida puede ser jugada por 1 o 2 usuarios.
- Una partida puede ser guardada por 0 o muchos usuarios.
- Una partida es validada por 1 validador.
- Una partida puede ser jugada por 1 Stockfish.

### 4. Validator (Validador):

Descripción: Valida los movimientos en una partida de ajedrez.

Campos:

- Tablero: Estado del tablero de ajedrez. Métodos:
- bool validarMovimiento(move): Valida un movimiento.
- bool validarPeon(movimiento): Valida un movimiento de peón.
- bool validarTorre(movimiento): Valida un movimiento de torre.
- bool validarAlfil(movimiento): Valida un movimiento de alfil.
- bool validarCaballo(movimiento): Valida un movimiento de caballo.
- bool validarReina(movimiento): Valida un movimiento de reina.
- bool validarRey(movimiento): Valida un movimiento de rey.

Relaciones:

- Un validador valida 1 partida a la vez.

## 5. Stockfish:

Descripción: Representa la inteligencia artificial que juega ajedrez.

Campos:

- ELO: Puntaje ELO de Stockfish.
- Posicion: Posición actual en el tablero. Métodos:
- String getBestMove(): Obtiene el mejor movimiento posible.
- void setElo(elo): Establece el puntaje ELO.
- void setPosicion(movimientos): Establece la posición basada en movimientos.

Relaciones:

- Stockfish puede jugar entre 1 y muchas partidas.
- Stockfish puede ser desafiado por 1 usuario.
- Stockfish es llamado por un conector.

## 6. Conector:

Descripción: Gestiona la comunicación con la API.

Campos:

- Url: URL del conector. Métodos:
- static getBestMove(): Método estático para obtener el mejor movimiento.
- static setPosicion(): Método estático para establecer la posición.

Relaciones:

- Un conector llama a 1 Stockfish.

## Diagrama de secuencia de un caso de uso

A continuación se muestra el diagrama de secuencia en el que se desglosa un caso de uso en el que el usuario juega una partida contra la maquina y guarda la partida en la base de datos.

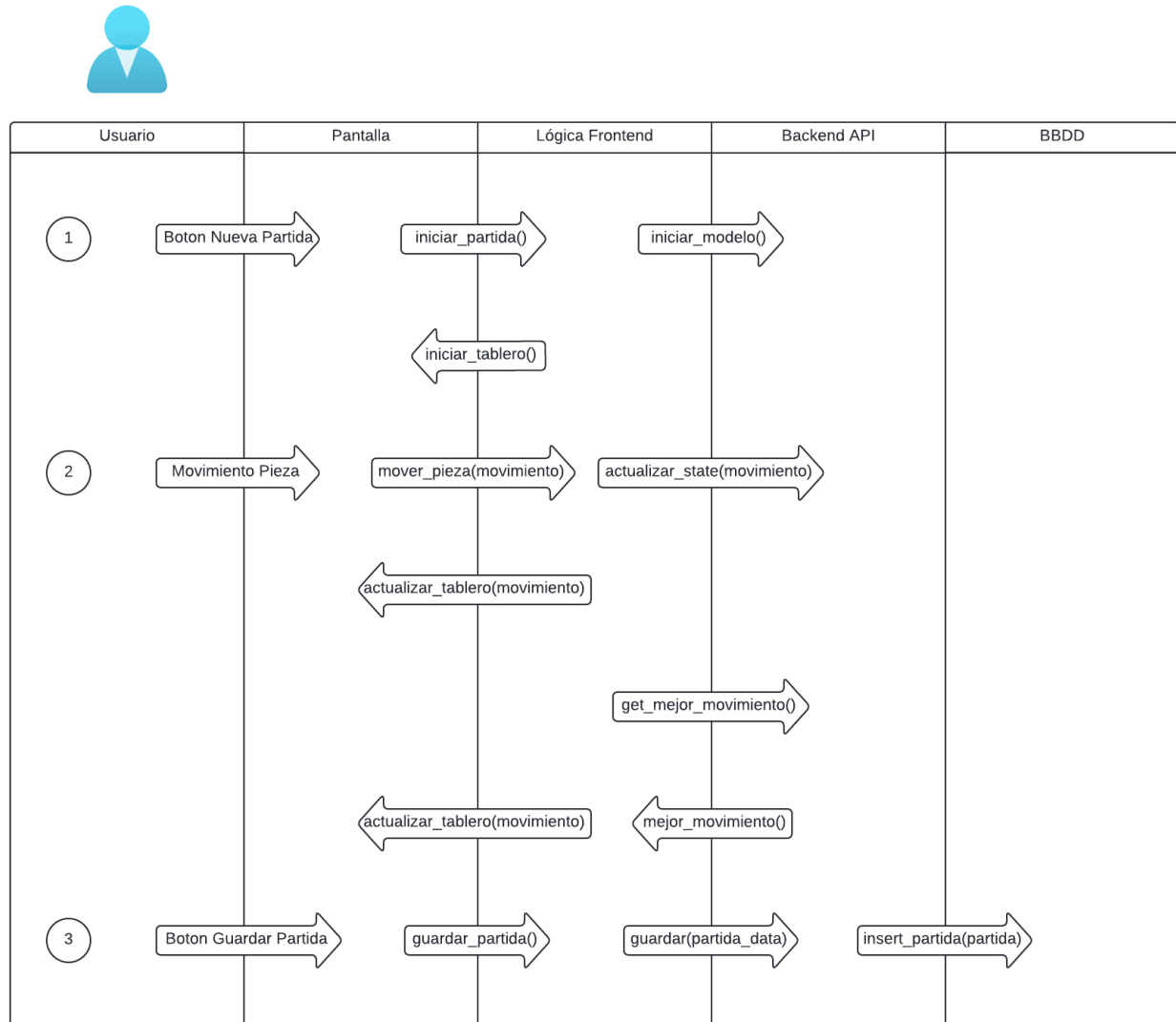


Figura 4, caso de uso usuario VS maquina.

### Desglose del diagrama de secuencia

El punto uno del diagrama refleja como el usuario selecciona en el menú la opción de iniciar una nueva partida pulsando el botón de nueva partida en la pantalla, esto activa la función de `iniciar_partida()` que prepara todos los elementos necesarios y hace una llamada a la API para iniciar el modelo. Una vez finaliza se ejecuta la función `iniciar_tablero()` que da comienzo a la partida mostrando el tablero por pantalla.

El punto dos afronta el ciclo de juego en el que el jugador mueve piezas y el modelo responde.

Para hacerlo funcionar una vez el jugador finaliza el desplazamiento de una pieza se registra ese movimiento con la función `mover_pieza(movimiento)`, esto actualiza el tablero en la app (no la pantalla) y hace una llamada a la API con `actualizar_state(movimiento)` que actualiza el tablero con el que trabaja el modelo, una vez finaliza actualiza la pantalla con `actualizar_tablero()` y llama a la función `get_mejor_movimiento()` que hace una llamada a la API forzando al modelo a responder con el mejor movimiento para ese tablero a través de la función `mejor_movimiento()`. Una vez se ha recogido el movimiento se actualiza el tablero con `actualizar_tablero()`. Este proceso se repite hasta que la partida llega a su fin donde el usuario tiene la opción de guardar la partida si así lo desea, en este caso se simula la situación en la que lo hace.

El punto tres muestra el flujo que se desencadena al pulsar el botón de guardar la partida, es bastante directo. Al ser pulsado se ejecuta el método `guardar_partida()` que recoge toda la información que se desea almacenar en la app y se la pasa a la función `guardar(partida_data)`, la cual hace una llamada a la API con la data a guardar como parámetro. La API finaliza el proceso realizando un insert en la tabla de partida con la función `insert_partida(partida)`.

## 6. Modelo de Datos

Como se menciona en el apartado de tecnologías empleadas la base de datos ha emplear ha sido SQL Server. Para poder sacar el máximo rendimiento y robustez a esta gestor de bases de datos es necesario contar con un modelo de datos estructurado, bien definido y solido.

Para poder cumplir con estos requisitos antes de empezar a desarrollar el script que creara la base de datos en la herramienta se ha definido un diagrama ER, este diagrama detalla la relación de las diferentes tablas y datos que se pueden encontrar en la base de datos. Tomando este diagrama como base se pudo definir un modelo relacional que posteriormente dio lugar al modelo físico empleado en producción.

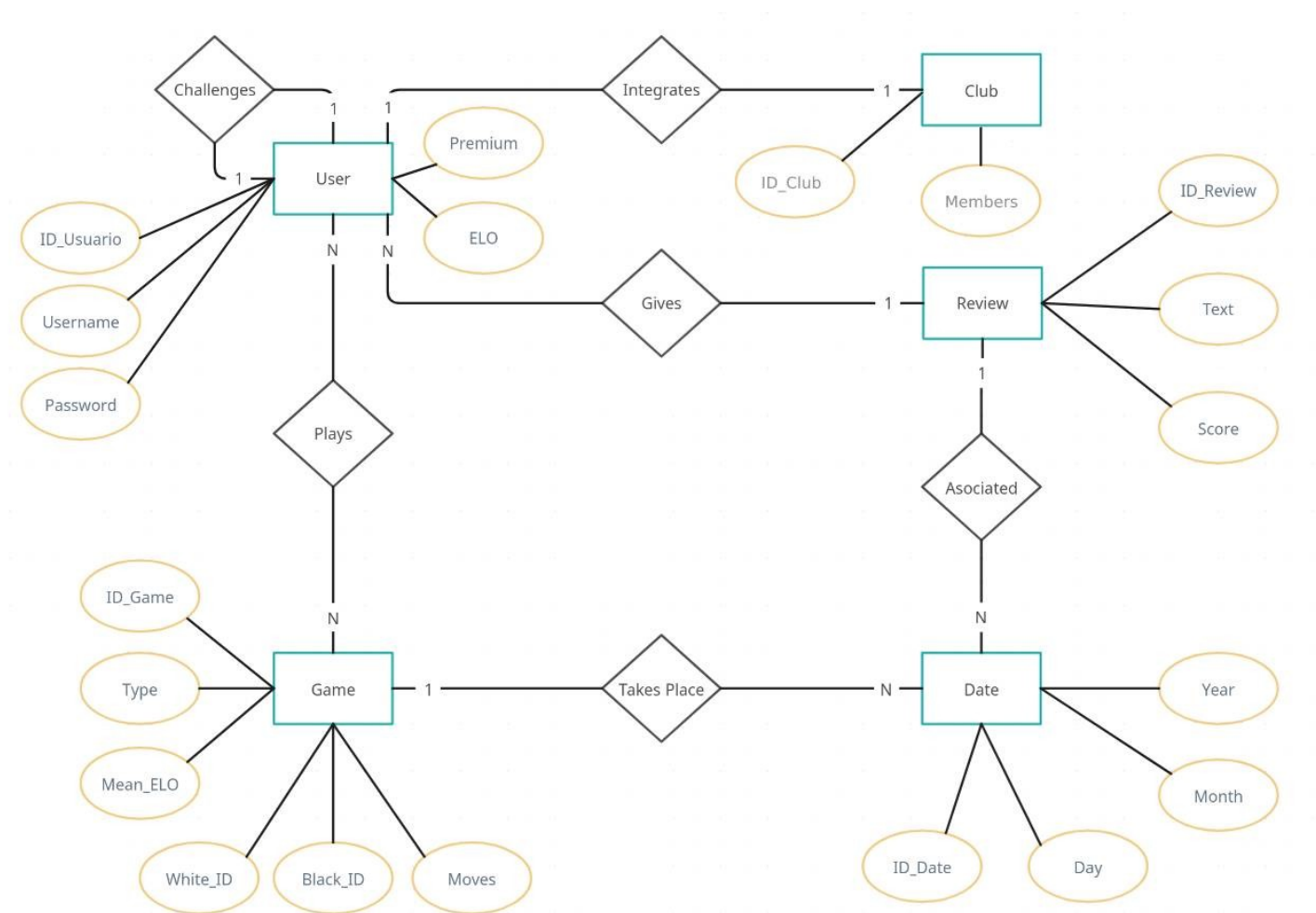


Figura 5, diagrama ER.

A continuación se explicara la funcionalidad de cada tabla y campo dentro de la base de datos en mas detalle para aclarar posibles ambigüedades de entendimiento del diagrama.

### **1. User:**

Descripción: Almacena la información de los usuarios.

Campos:

- ID\_Usuario: Identificador único del usuario.
- Username: Nombre de usuario.
- Password: Contraseña del usuario.
- Premium: Indica si el usuario tiene una cuenta premium.
- ELO: Puntuación ELO del usuario.

### **2. Club:**

Descripción: Almacena la información de los clubes de ajedrez.

Campos:

- ID\_Club: Identificador único del club.
- Members: Lista con los id de los miembros del club.

### **3. Review:**

Descripción: Almacena las reseñas que los usuarios dejan.

Campos:

- ID\_Review: Identificador único de la reseña.
- Text: Texto de la reseña.
- Score: Puntuación otorgada en la reseña.
- ID\_Usuario: Identificador del usuario que dejó la reseña.

### **4. Game:**

Descripción: Almacena la información sobre las partidas de ajedrez.

Campos:

- ID\_Game: Identificador único de la partida.
- Type: Tipo de partida (e.g., Blitz, Rapid).
- Mean\_ELO: Puntaje ELO promedio de los jugadores.
- White\_ID: Identificador del jugador con piezas blancas.
- Black\_ID: Identificador del jugador con piezas negras.
- Moves: Movimientos realizados en la partida.

## **5. Date:**

Descripción: Almacena las fechas.

Campos:

- ID\_Date: Identificador único de la fecha.
- Year: Año.
- Month: Mes.
- Day: Día.

## **6. Usuario\_Club:**

Descripción: Tabla intermedia que representa la relación de integración de usuarios en clubes.

Campos:

- ID\_Usuario: Identificador del usuario.
- ID\_Club: Identificador del club.

## **7. Usuario\_Challenges:**

Descripción: Tabla intermedia que representa los desafíos entre usuarios.

Campos:

- Challenger\_ID: Identificador del usuario que realiza el desafío.
- Challenged\_ID: Identificador del usuario desafiado.

## **8. Usuario\_Game:**

Descripción: Tabla intermedia que representa la relación de los usuarios con las partidas que juegan.

Campos:

- ID\_Usuario: Identificador del usuario.
- ID\_Game: Identificador de la partida.

## **9. Review\_Date:**

Descripción: Tabla intermedia que asocia las reseñas con las fechas.

Campos:

- ID\_Review: Identificador de la reseña.
- ID\_Date: Identificador de la fecha.



## **10. Game\_Date:**

Descripción: Tabla intermedia que asocia las partidas con las fechas en las que se jugaron.

Campos:

- ID\_Game: Identificador de la partida.
- ID\_Date: Identificador de la fecha.

## 7. Manuales de Instalación y Usuario

### Manual de Instalación

#### *Introducción*

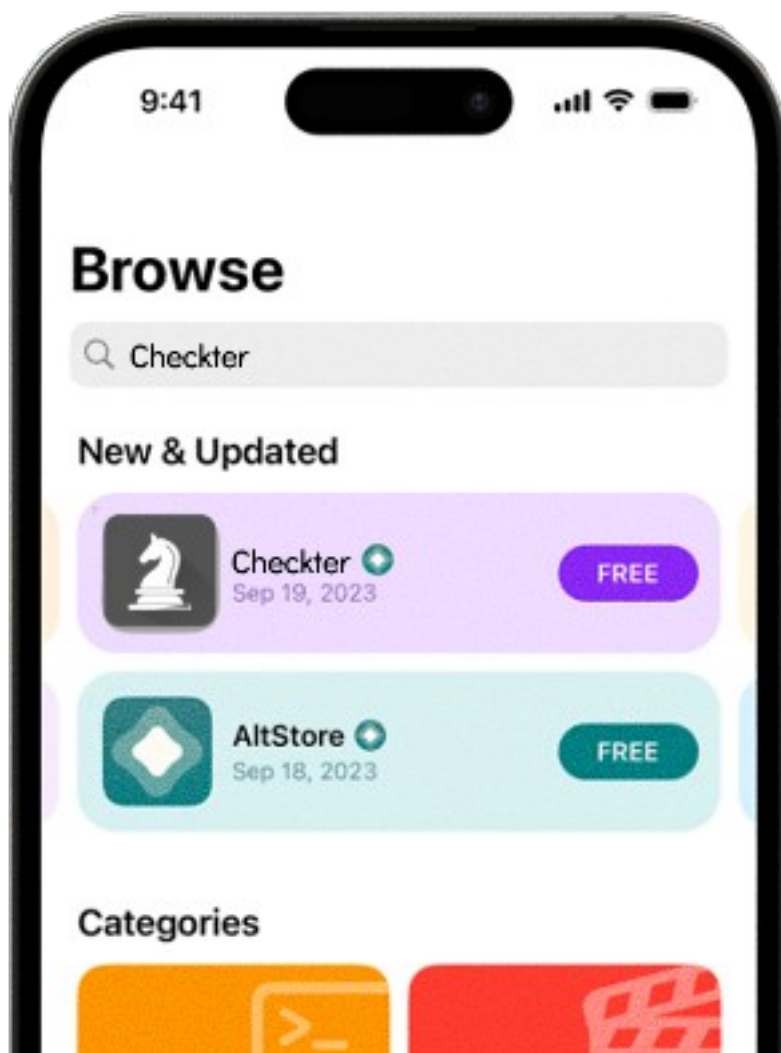
Este documento está diseñado para guiarlo paso a paso en el proceso de instalación, asegurando que pueda comenzar a disfrutar de todas las características y funcionalidades de nuestra app sin inconvenientes. La aplicación de ajedrez que está a punto de instalar es una herramienta avanzada creada para jugadores de todos los niveles, desde principiantes hasta expertos, y ofrece una amplia gama de opciones de juego, análisis y aprendizaje. A continuación se tratará la instalación en los distintos medios para los que esta disponible nuestro producto.

#### *Instalación en Dispositivos iOS (App Store)*

1. En tu iPhone o iPad, abre la app App Store.
2. Seleccione la opción “Buscar” y escriba Checkter.



3. Toca o haz clic en el botón Free. Si ves el botón Open en lugar de el botón Free, quiere decir que ya has descargado nuestra app.



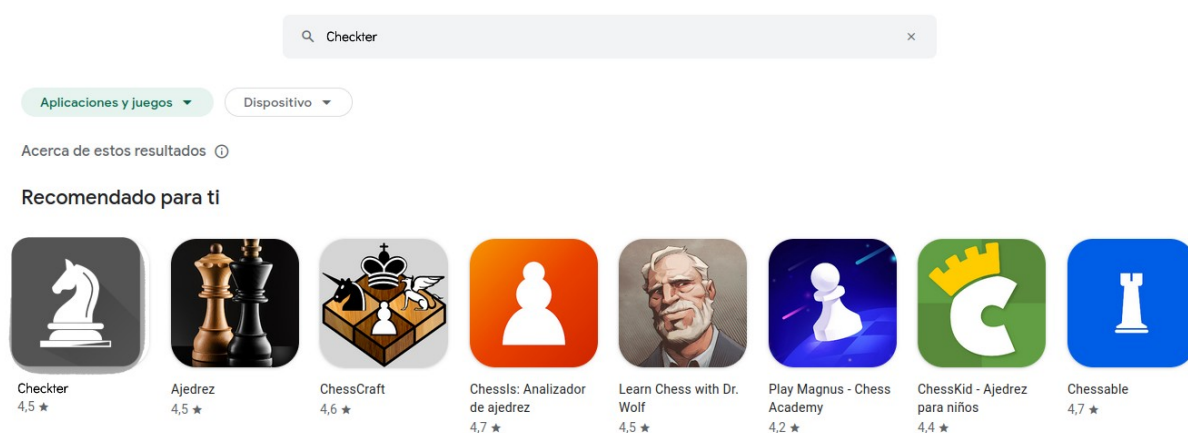
4. Para abrir la app seleccione la herramienta de “Aplicaciones” dentro de su dispositivo y haga click en aquella app que tenga el icono de Checker, si no es capaz de encontrar la aplicación seleccione la herramienta de buscar y escriba el nombre “Checker”, pulse en la aplicación resultante de la búsqueda.

## ***Instalación en Dispositivos Android (Play Store)***

1. En tu dispositivo Android, abre la app Play Store.
2. Seleccione la opción “Buscar aplicaciones y juegos” y escriba Checkter.



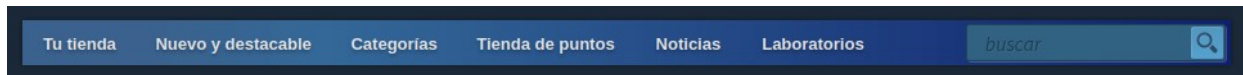
3. Toca o haz click en el icono de la aplicación y pulsa instalar. Si ves el botón “Abrir” en lugar de el botón “Instalar”, quiere decir que ya has descargado nuestra app.



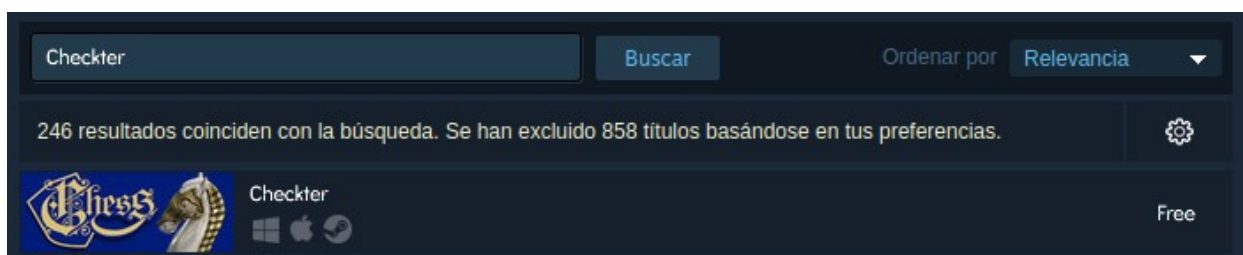
4. Para abrir la app seleccione la herramienta de “Aplicaciones” dentro de su dispositivo y haga click en aquella app que tenga el icono de Checkter, si no es capaz de encontrar la aplicación seleccione la herramienta de buscar y escriba el nombre “Checkter”, pulse en la aplicación resultante de la búsqueda.

### *Instalación en PC (Steam)*

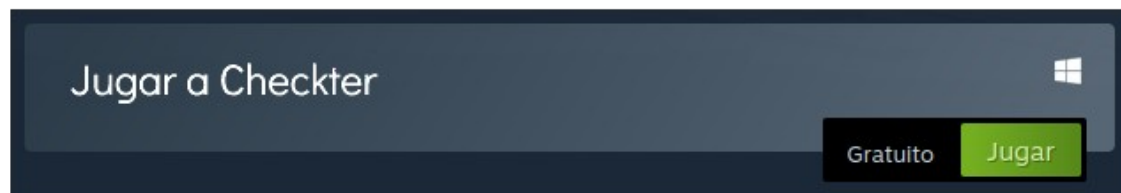
1. Instala Steam. Si no sabes como visita este [link](#).
2. Inicia Steam.
3. Selecciona la herramienta “buscar” y escriba Checkter.



4. Seleccione el resultado que aparece en pantalla.



5. Seleccione la opción de “Jugar”.



6. Una vez instalado ejecute la aplicación de Checkter, puede hacerlo desde la biblioteca de Steam o desde su dispositivo. Si no puede encontrar la aplicación se recomienda hacer uso de la herramienta de búsqueda de su sistema operativo.

## ***Actualizaciones de la Aplicación***

### ***iOS: Pasos para actualizar la aplicación en un dispositivo iOS***

1. Abra la App Store: Toque el ícono de la App Store en su pantalla de inicio.
2. Acceda a "Actualizaciones": En la barra inferior, toque el ícono de su perfil en la esquina superior derecha.
3. Busque actualizaciones: Desplácese hacia abajo para encontrar la sección "Actualizaciones disponibles".
4. Encuentre la aplicación de ajedrez: Busque la aplicación de ajedrez en la lista de aplicaciones que necesitan actualización.
5. Toque "Actualizar": Si la aplicación de ajedrez aparece en la lista, toque el botón "Actualizar" junto a ella. Si desea actualizar todas las aplicaciones a la vez, puede tocar "Actualizar todo".

### ***Android: Pasos para actualizar la aplicación en un dispositivo Android***

1. Abra Google Play Store: Toque el ícono de Google Play Store en su pantalla de inicio o en el menú de aplicaciones.
2. Acceda al menú: Toque el ícono de su perfil en la esquina superior derecha.
3. Seleccione "Administrar apps y dispositivo": Dentro del menú, toque "Administrar apps y dispositivo".
4. Verifique actualizaciones disponibles: En la sección "Actualizaciones disponibles", toque "Ver detalles".
5. Encuentre la aplicación de ajedrez: Busque la aplicación de ajedrez en la lista de aplicaciones que tienen actualizaciones disponibles.
6. Toque "Actualizar": Toque el botón "Actualizar" junto a la aplicación de ajedrez. Si desea actualizar todas las aplicaciones a la vez, puede tocar "Actualizar todo".

*PC (Steam): Pasos para actualizar la aplicación desde Steam*

1. Abra Steam: Inicie el cliente de Steam en su PC.
2. Acceda a su Biblioteca: Haga clic en "Biblioteca" en el menú superior para ver su lista de juegos.
3. Busque actualizaciones: Steam generalmente actualiza las aplicaciones automáticamente. Sin embargo, si una actualización está pendiente, verá una notificación en la aplicación de ajedrez en su lista de juegos.
4. Seleccione la aplicación de ajedrez: Haga clic en el nombre de la aplicación de ajedrez para ver su página en Steam.
5. Inicie la actualización manualmente (si es necesario): Si la actualización no se ha iniciado automáticamente, puede haber un botón "Actualizar" disponible. Haga clic en él para comenzar la actualización.
6. Espere a que se complete la actualización: Steam descargará e instalará la actualización automáticamente. Una vez completada, podrá iniciar la aplicación normalmente.

## ***Desinstalación de la Aplicación***

### ***iOS: Pasos para desinstalar la aplicación de un dispositivo iOS***

1. Ubique la aplicación: En la pantalla de inicio de su dispositivo iOS, busque el ícono de la aplicación de ajedrez.
2. Mantenga presionado el ícono: Toque y mantenga presionado el ícono de la aplicación hasta que aparezca un menú contextual o los íconos comiencen a moverse.
3. Seleccione "Eliminar app": En el menú que aparece, toque la opción "Eliminar app".
4. Confirme la desinstalación: Aparecerá una ventana emergente pidiéndole que confirme la eliminación de la aplicación. Toque "Eliminar" para confirmar.

### ***Android: Pasos para desinstalar la aplicación de un dispositivo Android***

1. Acceda a la configuración: Abra el menú de aplicaciones y vaya a "Configuración" (puede variar según la marca y modelo del dispositivo).
2. Seleccione "Aplicaciones" o "Administrador de aplicaciones": Dentro de la configuración, busque y seleccione la opción "Aplicaciones" o "Administrador de aplicaciones".
3. Encuentre la aplicación de ajedrez: Desplácese por la lista de aplicaciones instaladas y toque la aplicación de ajedrez que desea desinstalar.
4. Toque "Desinstalar": En la pantalla de información de la aplicación, toque el botón "Desinstalar".
5. Confirme la desinstalación: Aparecerá una ventana emergente solicitando confirmación. Toque "Aceptar" o "Desinstalar" para confirmar.



*PC (Steam): Pasos para desinstalar la aplicación desde Steam*

1. Abra Steam: Inicie el cliente de Steam en su PC.
2. Acceda a su Biblioteca: Haga clic en "Biblioteca" en el menú superior para ver su lista de juegos.
3. Encuentre la aplicación de ajedrez: En la lista de juegos de su biblioteca, busque la aplicación de ajedrez que desea desinstalar.
4. Haga clic derecho en la aplicación: Haga clic derecho en el nombre de la aplicación para abrir un menú contextual.
5. Seleccione "Administrar" y luego "Desinstalar": En el menú contextual, seleccione "Administrar" y luego haga clic en "Desinstalar".
6. Confirme la desinstalación: Steam le pedirá que confirme la desinstalación. Haga clic en "Desinstalar" para confirmar y eliminar la aplicación de su PC.

# **Manual de Usuario**

## **Introducción**

Bienvenido al Manual de Usuario de Checkter, una innovadora aplicación de ajedrez desarrollada por Martín Carballo Otero como proyecto de fin de ciclo en Aula Nosa. Checkter es una aplicación diseñada para ofrecer una experiencia de ajedrez envolvente y moderna, accesible desde dispositivos móviles y ordenadores.

## **Propósito del Manual**

Este manual está diseñado para guiar a los usuarios en el uso eficiente y efectivo de la aplicación Checkter, proporcionando una visión detallada de las funcionalidades de la aplicación.

## **Público Objetivo**

Checkter está dirigido a una amplia gama de jugadores de ajedrez, desde principiantes hasta expertos. La aplicación ofrece partidas amistosas locales, desafíos en línea con amigos, y partidas contra la máquina, utilizando un motor de inteligencia artificial para ofrecer diferentes niveles de dificultad. Este manual es útil tanto para nuevos usuarios que desean aprender a usar la aplicación como para usuarios experimentados que buscan aprovechar al máximo todas las funcionalidades que Checkter ofrece.

## **Requisitos previos al manual de usuario**

Es recomendable leer primero el manual de instalación porque proporciona instrucciones esenciales para descargar y configurar Checkter correctamente en tu dispositivo. Completar esta etapa inicial sin problemas asegura que la aplicación funcione de manera óptima, evitando dificultades técnicas. De este modo, los usuarios pueden aprovechar plenamente las funcionalidades descritas en el manual de usuario.

## **Iniciando la aplicación**

### **En Dispositivos Android**

1. Localiza la Aplicación: En la pantalla de inicio o en el menú de aplicaciones, busca el ícono de Checkter.
2. Abre la Aplicación: Toca el ícono de Checkter. La aplicación se abrirá y mostrará la pantalla de inicio de la aplicación.

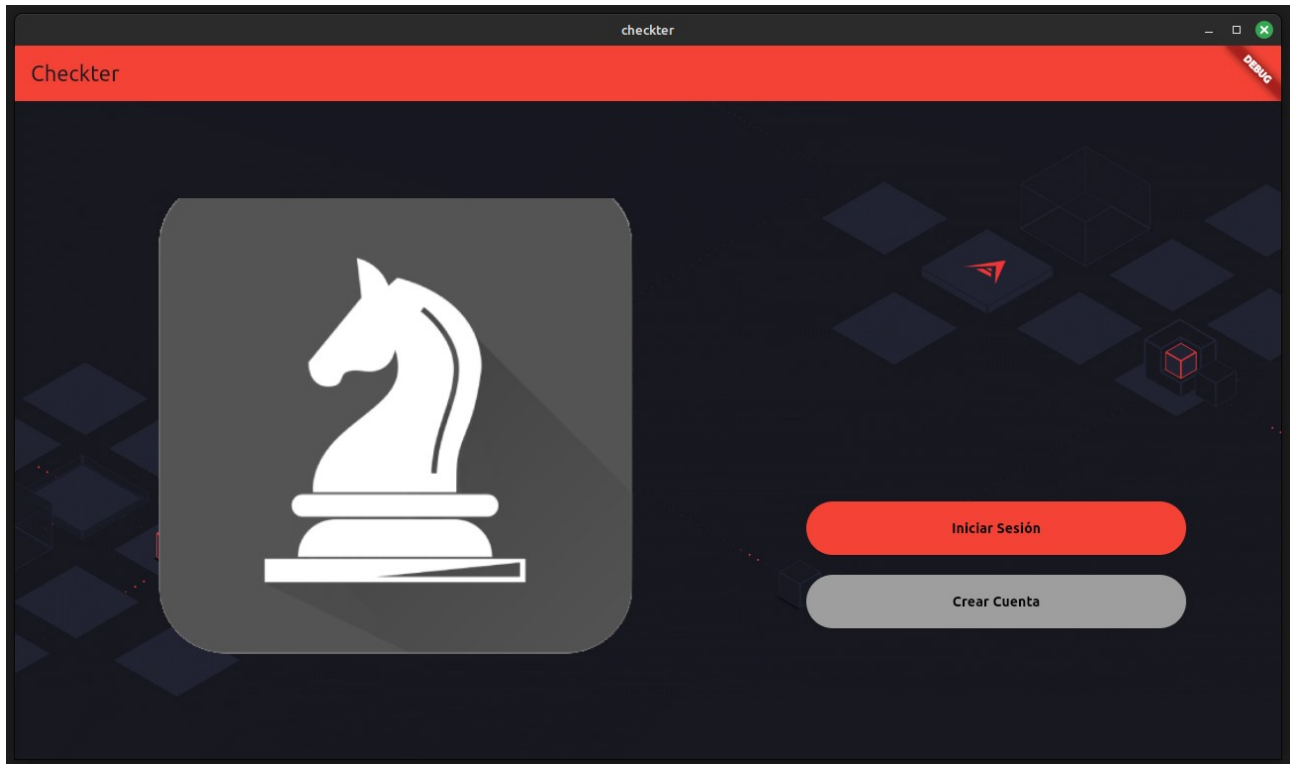
### **En Dispositivos iOS**

1. Localiza la Aplicación: En la pantalla de inicio de tu iPhone o iPad, busca el ícono de Checkter.
2. Abre la Aplicación: Toca el ícono de Checkter. La aplicación se abrirá y mostrará la pantalla de inicio de la aplicación.

### **En PC (Steam)**

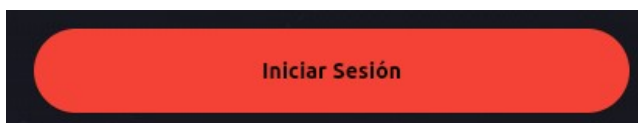
1. Abre Steam: Inicia el cliente de Steam en tu PC.
2. Accede a tu Biblioteca: Haz clic en "Biblioteca" en el menú superior para ver tu lista de juegos.
3. Localiza la Aplicación: Busca Checkter en la lista de juegos instalados.
4. Inicia la Aplicación: Haz clic en "Jugar" para iniciar la aplicación. Checkter se abrirá y mostrará la pantalla de inicio de la aplicación.

## Pantalla de inicio



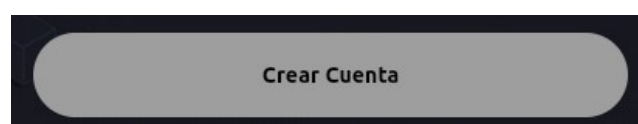
### Botón "Iniciar Sesión":

- Ubicado en la parte inferior derecha, este botón de color rojo permite a los usuarios existentes acceder a sus cuentas. Al hacer clic en "Iniciar Sesión", se abrirá una nueva pantalla o ventana donde el usuario podrá ingresar sus credenciales (nombre de usuario y contraseña) para acceder a sus datos y partidas guardadas.

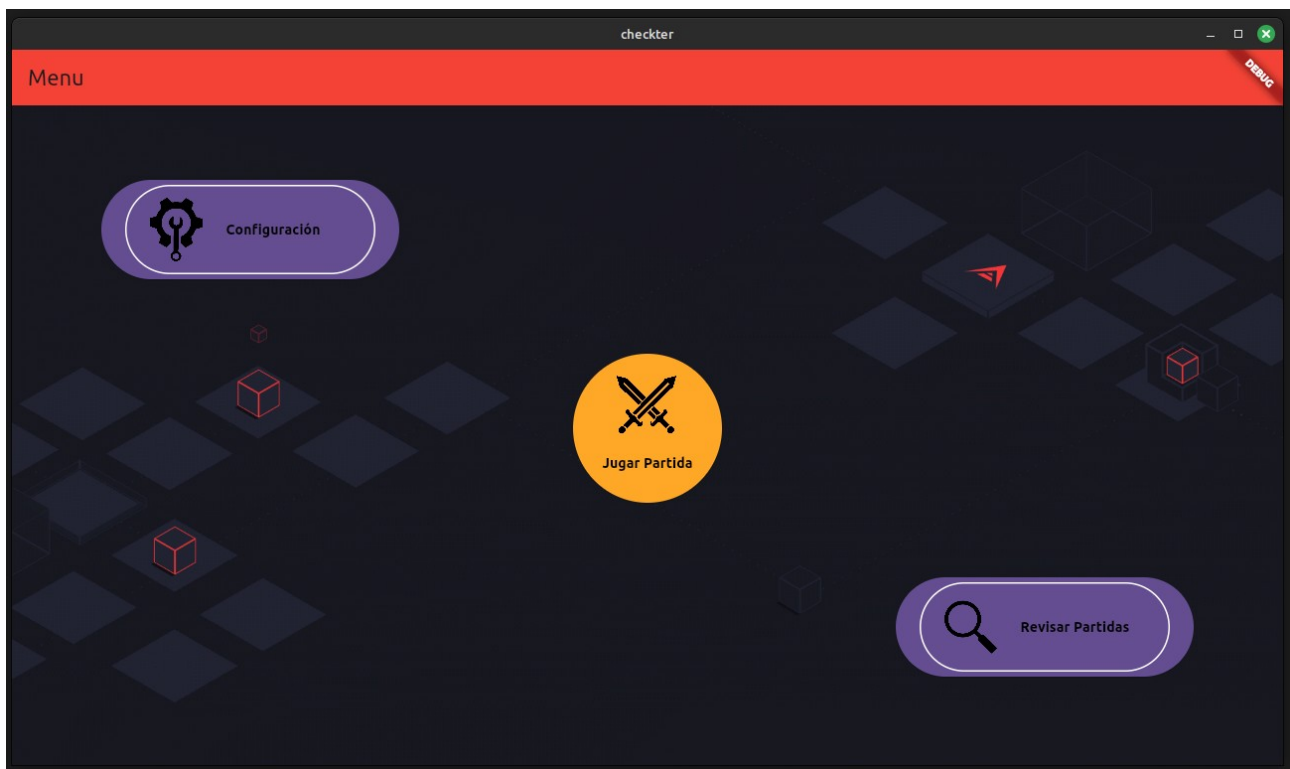


### Botón "Crear Cuenta":

- Justo debajo del botón de iniciar sesión, se encuentra el botón "Crear Cuenta" en color gris. Este botón está destinado a nuevos usuarios que aún no tienen una cuenta en Checkter. Al hacer clic en "Crear Cuenta", se abrirá una pantalla donde el usuario podrá registrarse proporcionando la información requerida (como nombre, correo electrónico y una contraseña).



## Pantalla de menú



Botón "Configuración":

- Ubicado en la parte superior izquierda de la pantalla, dentro de un óvalo morado, este botón lleva al usuario a la sección de configuración de la aplicación. Aquí, los usuarios pueden ajustar diversas preferencias y configuraciones, como sonido, notificaciones, y otras opciones personalizables.



Botón "Jugar Partida":

- En el centro de la pantalla, dentro de un círculo naranja con dos espadas cruzadas, este botón permite al usuario iniciar una nueva partida de ajedrez. Al hacer clic en este botón, el usuario podrá seleccionar entre diferentes modos de juego, como partidas locales, contra la inteligencia artificial, o en línea contra otros jugadores.

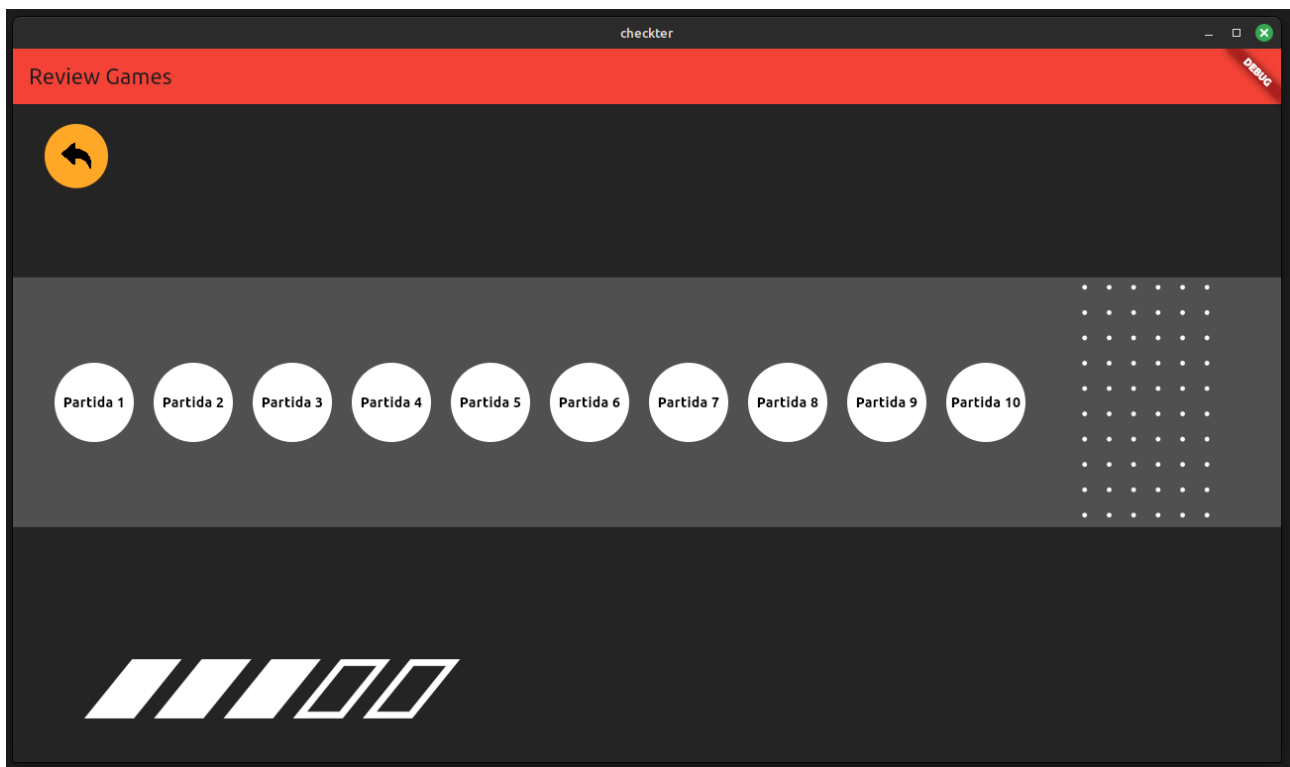


Botón "Revisar Partidas":

- En la parte inferior derecha, dentro de un óvalo morado con un icono de lupa, este botón lleva al usuario a una sección donde pueden revisar partidas anteriores. Aquí, los usuarios pueden cargar y analizar sus partidas guardadas, revisar movimientos y estudiar estrategias.



## Pantalla de revisar partidas



Botón "Regresar Menú":

- El botón "Regresar Menú" se encuentra en varias secciones de la aplicación Checkter y sirve para llevar al usuario de vuelta a la pantalla de menú principal. Este botón es esencial para la navegación, permitiendo al usuario salir de cualquier submenú o funcionalidad en la que se encuentre y retornar fácilmente al menú principal. Esto es útil para acceder rápidamente a otras secciones de la aplicación, como iniciar una nueva partida, revisar configuraciones o analizar partidas anteriores.



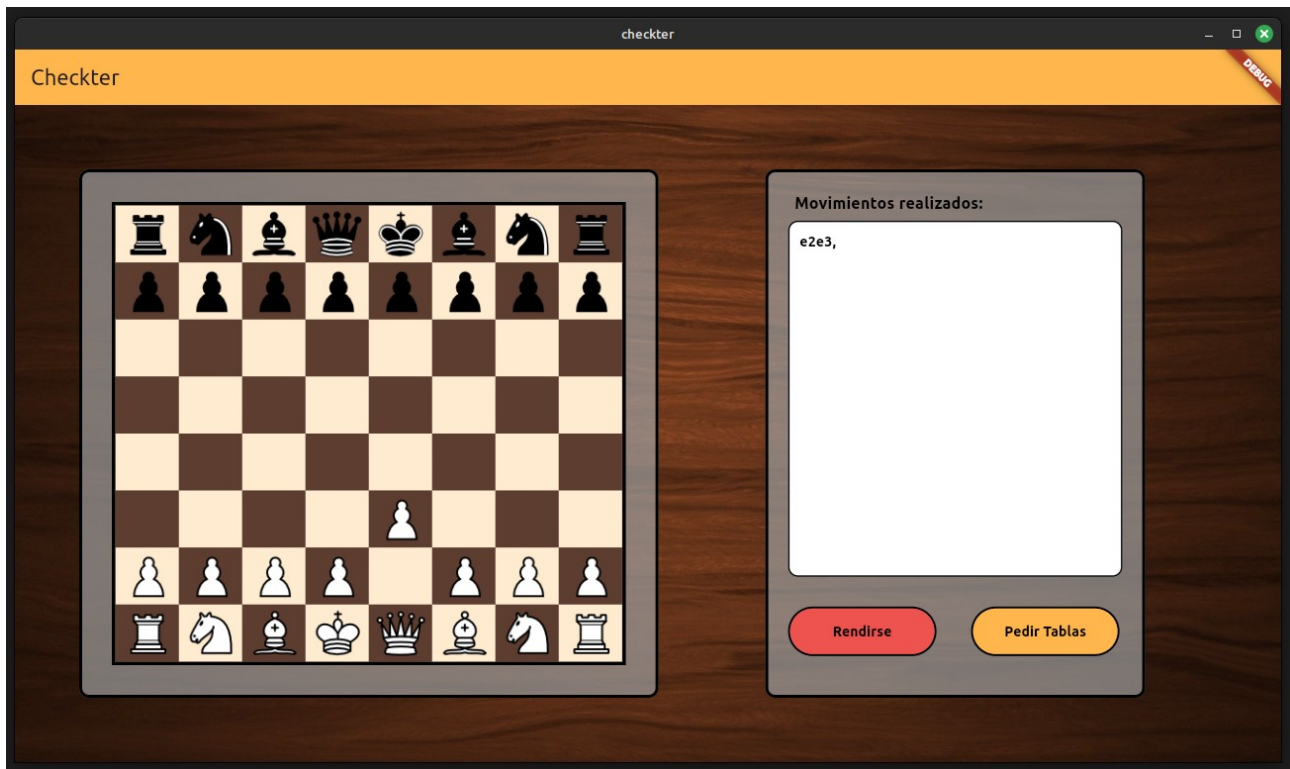
#### Botones "Analizar Partida":

- Los botones "Analizar Partida" son esferas blancas ubicadas en la sección de revisión de partidas dentro de la aplicación Checkter. Al ser pulsados, estos botones abren una pantalla dedicada a la herramienta de análisis de partidas. Esta herramienta permite al usuario analizar en profundidad las partidas de ajedrez jugadas, evaluando los movimientos realizados, identificando errores, sugiriendo movimientos alternativos y proporcionando una visión detallada de la estrategia utilizada. Esta funcionalidad es especialmente útil para los jugadores que desean mejorar sus habilidades, aprender de sus errores y comprender mejor las tácticas y estrategias del ajedrez.



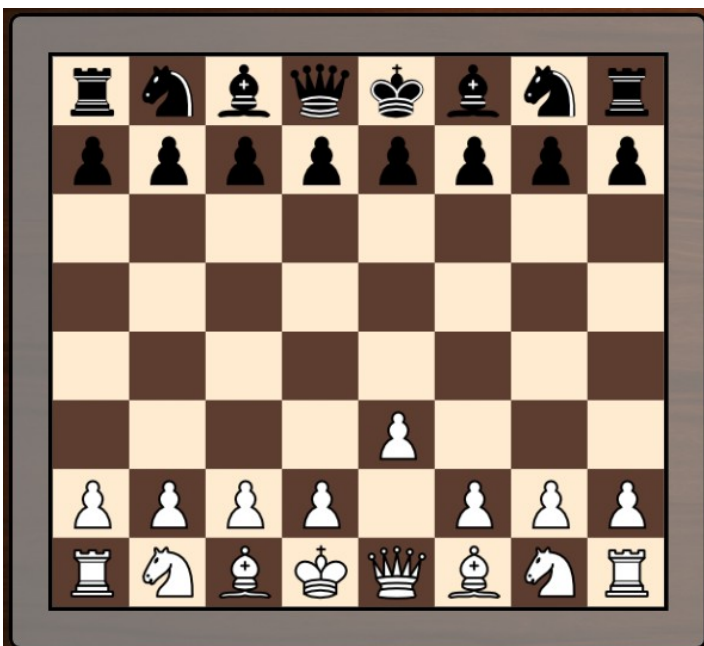


## Pantalla de partida



### Tablero de Ajedrez:

- El tablero de ajedrez se encuentra en el lado izquierdo de la pantalla, mostrando la posición inicial de las piezas. Aquí es donde los jugadores realizan sus movimientos durante la partida. El tablero es interactivo y permite mover las piezas según las reglas del ajedrez.



#### Lista de Movimientos Realizados:

- A la derecha del tablero, hay un cuadro titulado "Movimientos realizados". Este cuadro muestra una lista de los movimientos que se han realizado hasta el momento en la partida, comenzando con "e2e3". Esta lista se actualiza automáticamente con cada movimiento realizado por los jugadores.



#### Botón "Rendirse":

- En la parte inferior derecha, dentro de un botón rojo, está la opción "Rendirse". Al pulsar este botón, el jugador puede rendirse, lo que pone fin a la partida y declara al oponente como ganador.



#### Botón "Pedir Tablas":

- Al lado del botón "Rendirse", dentro de un botón amarillo, está la opción "Pedir Tablas". Este botón permite al jugador ofrecer tablas al oponente, lo que significa proponer un empate. Si el oponente acepta, la partida termina en empate.



## **8. Viabilidad**

La línea de negocio que plantea la empresa es la de una aplicación móvil de ajedrez que busca generar beneficio a partir de anuncios y un sistema de suscripciones en las que el usuario adquiere una serie de beneficios dentro de la aplicación (acceso a más espacios de guardado para sus partidas, revisión de partida con asistente, etc.).

### **Estudio de mercado**

El mercado de aplicaciones móviles mantiene un crecimiento acelerado desde hace años y parece que continuará la misma tendencia de cara a futuro.

El tamaño del mercado mundial de desarrollo de aplicaciones móviles fue de 10430 millones de dólares en 2021 y alcanzará los 25500 millones de dólares en 2028, exhibiendo una tasa compuesta anual del 13,2% durante el período de pronóstico.

Actualmente el mercado más amplio es el de América del Norte, y el que más crecimiento está teniendo es el de Asia en el Pacífico.

Dentro del mercado de aplicaciones los actores más potentes son: AWS, AppsFlyer, Adobe, Kochava y Google Analytics.

El público de usuarios que más aplicaciones móviles utiliza es el de 18 a 24 años y el que más gasta en ellas es el de 25 a 34 años.

### **Análisis económico**

#### **Costes:**

- Salarios, asumiendo un equipo compuesto por dos desarrolladores juniors y un senior se pueden prever unos costes salariales de 1290\$ por junior y 2600\$ por senior, además se contaría con un diseñador gráfico con experiencia media para el apartado visual con un salario de 1.504\$, asumiendo desarrollo de 3 meses da a unos costes de 16182\$.
- Alojamiento Cloud, seleccionando AWS se pueden asumir unos costes de 390-450\$ al mes.
- Licencias publicación, publicando en Play Store, App Store y Steam contaríamos con unos costes de publicación de 150\$.

- Marketing, siguiendo la regla general de inversión del 25% del coste del desarrollo se pueden deducir unos costes de marketing de  $21720 * 0.25 = 5430\$$ .

### **Inversión:**

- Económica, debe ser capaz de hacer frente a los costes que se den durante el desarrollo y los costes fijos previos a la publicación de la aplicación. Teniendo en cuenta los costes previstos se estima una inversión de 22212\$.
- Tiempo, debido al coste de oportunidad el tiempo invertido en el desarrollo y mantenimiento de la aplicación también se debe considerar parte de la inversión.

### **Ventas:**

De forma general la cifra aproximada de usuarios gasta dinero en micro transacciones dentro de aplicaciones y juegos oscila entre el 4.5% y el 6.5% dependiendo del tipo de juego. El gasto aproximado de estos usuarios que adquieren productos de pago es de 1.5\$ en Play Store y 10.4\$ en App Store cada mes. Steam por otro lado no ofrece una cifra tan clara a nivel mensual pero a partir del gasto promedio anual por jugador (23,66\$) se extrae un gasto aproximado de 1,97\$ por mes.

Es importante destacar que cada plataforma aplica una retención al ingreso bruto, 15% en App y Play Store y un 7.5% en Steam.

Operando con la media de estos datos de compra por usuario(5.5%) y los precios correspondientes a cada plataforma se nos plantearía la siguiente ecuación de ingreso aproximado por mes:

$$(Players App Store * 0.055 * 10.4 * 0.85) + (Players Play Store * 0.055 * 1.5 * 0.85) + (Players Steam * 0.055 * 1.97 * 0.925) = Ingreso Ventas$$

### **Previsión esperada:**

Basándonos en la entrada al mercado de aplicaciones similares a nuestro producto podríamos asumir una cuota de usuarios en el primer mes de entre 3000 a 6000 usuarios por plataforma, cifra que podría crecer con el paso del tiempo, asumiendo esta cuota inicial de jugadores podríamos asumir un rango de ingresos de:

$$(3000 * 0.055 * 10.4 * 0.85) + (3000 * 0.055 * 1.5 * 0.85) + (3000 * 0.055 * 1.97 * 0.925) =$$

*Ingreso Ventas Mínimo = 1969,64\$*

$$(6000 * 0.055 * 10.4 * 0.85) + (6000 * 0.055 * 1.5 * 0.85) + (6000 * 0.055 * 1.97 * 0.925) =$$

*Ingreso Ventas Máximo = 3939,29\$*

Estos cálculos se basan en datos aproximados de aplicaciones de perfil similar y no aportan una garantía real, sólo deben ser utilizados como valores de referencia aproximada.

### **Estudio de riesgos**

Los riesgos que puede experimentar la aplicación se pueden dividir en 3 apartados:

#### **Técnicos:**

- Fallos de software: Bugs y errores de compatibilidad.
- Seguridad: Vulnerabilidades y posibles hackeos.
- Rendimiento: Lentitud y fallos bajo alta carga.

#### **Negocio:**

- Competencia: Nuevas aplicaciones competidoras.
- Monetización: Dificultades para generar ingresos.

#### **Usuario:**

- Experiencia: Interfaz no intuitiva, sonidos inadecuados, etc.
- Desinstalación: Pérdida del interés del usuario.

En base a estos riesgos se ha diseñado 3 planes de acción que buscan impedir la ocurrencia de estos riesgos o en caso de no ser posible mitigar sus efectos.

**Técnicos:** Se realizaran pruebas tanto durante el desarrollo del código como en su etapa de mantenimiento, tanto de integración, sistema y usuario. Para asegurar la compatibilidad se realizara el testing en distintos dispositivos, publicando una lista de aquellos que han sido probados.

Para mejorar la seguridad se mantendrán actualizadas las dependencias, se cifraran los datos de la aplicación y se planteara la contratación de un servicio de auditoria.

Para mejorar el rendimiento se realizaran pruebas de carga y se mantendrá un monitoreo regular.

**Negocio:** Se realizará investigación de mercado regular analizando tendencias y nuevas apps. Se buscara diversificar los métodos de ingreso en compras, suscripciones, publicidad. Ademas se tratara de dar acceso al mayor numero de métodos de pago posible.

**Usuario:** La interfaz será diseñada para ser simple y centrada en la comodidad del usuario y sera mejorada con el feedback que estos provean. Se buscara mantener el interés de los usuarios mediante actualizaciones regulares, notificaciones y un programa de recompensas por usar la app diariamente.

## DAFO

En base a los datos expuestos previamente se ha diseñado un DAFO que busca condensar la información tratada y mostrarla de forma simplificada.



*Figura 6, DAFO.*

## 9. Calidad

La calidad del desarrollo es un apartado clave para el correcto desarrollo de una aplicación móvil, ayuda a mantener una estructura clara, facilita la identificación de bugs y mejora la eficiencia y facilidad para mantener la app una vez está publicada. Por otra parte también supone un coste en tiempo que ralentiza los tiempos de desarrollo.

Code Alchemist está comprometida con el desarrollo seguro y de calidad por lo que ha destinado parte del tiempo de desarrollo a la implementación de un plan de pruebas.

A continuación se desglosarán los tipos de pruebas implementadas y algunos ejemplos de las mismas.

### Pruebas Unitarias (Unit Testing)

Centradas en la validación de segmentos de código aislados para poder trabajar con la garantía de su correcto funcionamiento. Son útiles para detectar errores en etapas tempranas, facilitar el refactoring del código y pueden ayudar a la compresión del código.

Ejemplo:

```
def test_guardar_game(self):
    response = self.client.post('/game', data=json.dumps({
        'moves': 'e4 e5 Nf3 Nc6'
    }), content_type='application/json')
    self.assertEqual(response.status_code, 201)
    data = json.loads(response.get_data(as_text=True))
    self.assertIn('message', data)
    self.assertEqual(data['message'], 'Partida guardada.')
    self.assertIn('game_id', data)
```

Esta prueba unitaria verifica que cuando se envía una solicitud para guardar un juego de ajedrez, el servidor responde con un código de estado 201, un mensaje de éxito y un identificador de juego.



## Pruebas de Usabilidad (Usability Testing)

Evalúan la facilidad de uso y la experiencia del usuario final con el sistema, también verifican que el sistema cumple con los requisitos funcionales especificados.

Ejemplo:



Se han realizado múltiples pruebas que aseguran el correcto funcionamiento del tablero y las piezas durante el transcurso de una partida de acuerdo al reglamento oficial de ajedrez internacional.

## Pruebas de Rendimiento (Performance Testing)

Evalúan el rendimiento del sistema bajo diferentes condiciones, normalmente se pueden dividir en pruebas de carga, estrés y escalabilidad.

Ejemplo:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
sh-5.2$ pip install locust # Incluimos locust a la lista de modulos
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
sh-5.2$ locust -f locustfile.py
```

Hacemos uso de Locust (módulo de testing de Python) para testear el numero de solicitudes por segundo, el tiempo de respuesta y el porcentaje de éxito que nos está dando nuestra API.

## 10. Conclusiones

Tras el seguimiento del plan previamente definido a lo largo de estas 12 semanas se han cumplido varios de los objetivos fijados y se han tenido que posponer otros por la presión del tiempo.

A continuación se darán unas conclusiones formadas a partir de finalizar el seguimiento de este plan de desarrollo.

### Estado actual

El primer punto a analizar va a ser el estado actual de la aplicación, funcionalidades que si que se han logrado llegar a implementar y están funcionales. Se dividirá el análisis en los mismos puntos que se dividió ‘Tecnologías empleadas’ para mantener la continuidad del documento.

**Frontend:** La interfaz de usuario a sido el apartado que más a sufrido debido a los tiempos de desarrollo, el frontend de nuestra aplicación es capaz de dar soporte a las partidas, tanto locales como contra AI, pero ha hecho un sacrificio en las pantallas intermedias (Pantalla de inicio y menús). Las partidas se guardan pero aún no se pueden revisar.

**Backend (API):** El backend a cumplido la mayoría de objetivos que buscaba implementar, es capaz de comunicar todos los elementos de la aplicación de forma exitosa, tanto el micro-servicio de almacenamiento y consumo de datos como el de inteligencia artificial han sido llevados a cabo.

**Inteligencia artificial:** Se ha conseguido implementar el modelo de inteligencia artificial Stockfish en local de forma exitosa, se permite la obtención de movimientos, la evaluación de los mismos y se comunica de forma adecuada con el frontend de la app.

**Base de datos:** Se ha completado el desarrollo de un script que implementa la estructura de la base de datos de la que hacemos uso y su ha podido conectar tanto con la API como con la app (A través del micro-servicio de la API).

**Documentación:** La documentación ha sido completada exitosamente, tanto manuales de usuario como de instalación están listos para ser publicados junto a este mismo documento.

## Mejoras a futuro

A continuación se tratarán elementos que se buscará implementar a partir de este momento para cumplir con objetivos que quedaron pendientes e implementar las nuevas ideas que surgieron durante el desarrollo. Una vez más se dividirá la clasificación en los 5 puntos mencionados anteriormente.

**Frontend:** El objetivo principal de cara a futuro es mejorar sustancialmente el apartado estético de la aplicación, crear distintas pantallas como de menú o pantalla inicial con mayor atractivo visual. Es también prioritario finalizar la funcionalidad de revisión de partidas.

**Backend (API):** El backend sigue una arquitectura muy fácil de escalar de cara a futuro, sus principales objetivos serían la ampliación de los micro-servicios con los que ya contamos y la implementación de nuevos, siendo el prioritario un micro-servicio que habilite el multiplayer en su formato Online y otro que permita la descarga de la apk o el exe desde la propia API.

**Inteligencia artificial:** Aunque el modelo con el que contamos (Stockfish) en este momento es perfectamente funcional y se mantendrá como modelo de evaluación en todo momento puede llegar a hacerse repetitivo el jugar contra el ya que mantiene ciertos patrones de juego, mirando a futuro sería interesante implementar nuevos modelos contra los que poder jugar o diseñar modelos propios de nuestra empresa empleando PyTorch.

**Base de datos:** La base de datos ya cumple su función de una forma adecuada, las únicas mejoras que se plantean de cara a futuro son la corrección de los errores que se puedan encontrar durante el uso de la misma y la ampliación de su estructura de tablas para poder albergar más información que pueda ser interesante conservar.

**Documentación:** El único objetivo para la documentación de cara a futuro es escalarla con las mejoras que se vayan implementando con el paso del tiempo, actualizar manual de uso, diagramas ER o UML, etc.

## 11. Bibliografía

- Martínez, J. (2022). **Arquitectura de micro-servicios: Beneficios y aplicaciones.** Universidad de Desarrollo de Software.  
<https://www.secdevoops.es/arquitectura-de-microservicios/>
- **Metodología tradicional vs ágil para la gestión de proyectos de software.** Boletín UPIITA.  
<https://www.boletin.upiita.ipn.mx/index.php/ciencia/925-cyt-numero-83/1901-metodologia-tradicional-vs-agil-para-la-gestion-de-proyectos-de-software>
- Fernández, R. (2023). **Diagramas UML estructurales para la Ingeniería del Software.** Universidad Politécnica Valenciana.  
<https://www.edx.org/es/learn/software-engineering/universitat-politecnica-de-valencia-diagramas-uml-estructurales-para-la-ingenieria-del-software>
- Rodríguez, L. (2022). **Modelo de Datos y Diagrama ER: Una guía práctica.** Universidad Don Bosco.  
[https://www.udb.edu.sv/udb\\_files/recursos\\_guias/informatica-moviles/base-de-datos-\(moviles\)/2019/i/guia-1.pdf](https://www.udb.edu.sv/udb_files/recursos_guias/informatica-moviles/base-de-datos-(moviles)/2019/i/guia-1.pdf)
- Pérez, G. (2023). **Análisis del Mercado de Aplicaciones Móviles en España: El Gasto de los Usuarios Sigue Creciendo.** Asnet.com.  
<https://www.asnet.es/analisis-del-mercado-de-aplicaciones-moviles-en-espana-el-gasto-de-los-usuarios-sigue-creciendo/>

- Rivera, A. (2023). **Indicadores económicos del desarrollo de software.** 233gradosdeti.com.  
<https://233gradosdeti.com/articulos/indicadores-economicos-del-desarrollo-software/>
- Gottdiener, Z (2022). **Gestión de riesgos en el desarrollo de software: 7 riesgos comunes**  
<https://www.door3.com/es/blog/understanding-risk-management-in-software-development-7-common-risks>
- Navarro, F. (2023). **6 herramientas UML para cualquier ocasión.**  
<https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/las-mejores-herramientas-uml/>
- Gómez, P. (2023). **Arquitectura de microservicios: por qué la prefieren los gigantes de la industria.** Fusiona.com.  
<https://fusiona.cl/blog/tecnologia/arquitectura-de-microservicios-por-que-la-prefieren-los-gigantes-de-la-industria/>