UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

TRABAJO FIN DE GRADO

"Análisis de precisión en partidas de ajedrez"

AUTOR
Óscar Davila Sampedro

DIRECTOR
Pablo González González

Agradecimientos

En primer lugar, quiero agradecer a mi madre y a mi hermano, cuyo apoyo y fe incondicionales me han permitido llegar tan lejos. También quiero expresar mi gratitud a mis tíos y primo, quienes son como una segunda familia para mí y siempre han estado a mi lado cuando más los he necesitado.

Agradezco a los profesores que supieron ver mis capacidades y me ayudaron a alcanzar esta meta. En especial, a mi tutor Pablo, ya que este proyecto no hubiera sido posible sin su guía y aportaciones.

Gracias a todos los compañeros y amigos que hice durante la carrera, quienes han sabido valorar lo que muchos otros no apreciaban.

Por último, gracias a aquellos que ya no están. Intento seguir vuestros pasos y espero que estéis orgullosos. Estoy seguro de que me animáis en cada uno que doy, desde primera fila, y no os canséis, porque aún me quedan muchos por dar.

Resumen

A pesar de su antigüedad, el ajedrez continúa atrayendo a millones de jugadores en plataformas en línea, muchos de ellos interesados en mejorar su desempeño. Sin embargo, estos sitios limitan el estudio de partidas en sus servidores para ahorrar recursos o requieren que los usuarios dependan de la conexión a internet para acceder a estas funciones de análisis.

Este proyecto ha desarrollado una aplicación de escritorio que permite analizar la precisión de los movimientos jugados por un usuario en ajedrez, evaluando sus partidas importadas de la plataforma online Lichess de manera local utilizando el módulo de código abierto Stockfish. Estos análisis se utilizan para mostrar el rendimiento del jugador en diferentes ritmos de juego, frente a diferentes tipos de oponentes o al jugar en un lado específico, entre otras estadísticas.

Durante el desarrollo de esta aplicación, se utilizaron lenguajes de programación web convencionales junto con el framework Electron, lo que permite que sea multiplataforma y funcione en Windows, macOS y Linux.

Palabras Clave

Partidas de ajedrez, Stockfish, Electron, Análisis de precisión

Abstract

Despite its age, chess continues to attract millions of players on online platforms, many of them interested in improving their performance. However, these sites limit the study of games on their servers to save resources or require users to rely on an internet connection to access these analysis functions.

This project has developed a desktop application to analyse the accuracy of a user's chess moves by evaluating their imported games from the online Lichess platform locally using the open source Stockfish module. These analyses are used to show the player's performance in different game rhythms, against different types of opponents or when playing on a specific board side, among other statistics.

During the development of this application, conventional web programming languages were used along with the Electron framework, which allows it to be cross-platform and work on Windows, macOS and Linux.

Keywords

Chess Games, Stockfish, Electron, Precision Analysis

Índice General

Índice de Figuras

Capítulo 1. Memoria del Proyecto

En este capítulo se presentan la motivación y los objetivos principales del proyecto, junto con un breve resumen de todos los aspectos recogidos en este documento.

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

El motivo principal detrás del desarrollo de este proyecto es abordar las limitaciones que presentan los servidores de ajedrez en línea al analizar las partidas jugadas en sus plataformas.

Una de estas restricciones es la limitación en el número de partidas que los usuarios pueden analizar, debido a la carga en los servidores. Estos no pueden evaluar el número de partidas que deseen, sino que están condicionados a un número reducido de estas cada cierto tiempo.

Además, dependen de una conexión estable a internet y del estado del servidor para poder acceder a estos servicios y sus resultados, que a menudo se presentan a través de una interfaz innecesariamente compleja, elevando la curva de aprendizaje y limitando su uso a un público específico y con experiencia.

Para superarlas, se propuso el desarrollo de una aplicación de escritorio capaz de analizar bajo demanda las partidas descargadas de uno de estos sitios. La aplicación utilizará el módulo de código abierto Stockfish en el propio equipo del usuario, lo que eliminará la necesidad de depender de un servidor externo. La aplicación también almacenará localmente tanto las partidas descargadas como los análisis realizados, lo que reducirá la necesidad de acceder a internet y evitará repeticiones innecesarias en los análisis. Todo esto se logrará con una interfaz sencilla, diseñada para que la aplicación sea accesible para un público amplio y no solo para usuarios avanzados.

1.2 Resumen de Todos los Aspectos

Este documento está organizado en 13 capítulos, presentados a continuación:

- **Capítulo 1:** Resumen de la motivación, objetivos y alcance del proyecto.
- **Capítulo 2:** Justificación y objetivos del proyecto, junto a un estudio de la situación actual de sistemas similares y las tecnologías a utilizar.
- **Capítulo 3:** Descripción y justificación de los conceptos, herramientas y tecnologías empleados en el proyecto.
- **Capítulo 4:** Planificación y presupuesto iniciales.
- **Capítulo 5:** Análisis del sistema, definición mediante requisitos y casos de uso, así como especificación del plan de pruebas.
- **Capítulo 6:** Diseño del sistema, su interfaz gráfica y pruebas, acompañados de diagramas para facilitar su comprensión.
- **Capítulo 7:** Implementación del sistema, consistente en la descripción de las diferentes tecnologías utilizadas y problemas encontrados.
- **Capítulo 8:** Resultados de las pruebas realizadas.
- **Capítulo 9:** Manuales de instalación, ejecución, usuario y programador.
- **Capítulo 10:** Conclusiones y posibles ampliaciones del proyecto.
- **Capítulo 11:** Planificación y presupuestos finales.
- **Capítulo 12:** Referencias bibliográficas.
- **Capítulo 13:** Glosario y descripción del contenido entregado en el archivo adjunto.

Capítulo 2. Introducción

En este capítulo se razonarán la justificación y objetivos del proyecto, junto a un estudio de alternativas y sistemas similares.

2.1 Justificación del Proyecto

Los jugadores de ajedrez en línea se enfrentan a limitaciones al tratar de analizar sus partidas. Los servidores de ajedrez (donde se juegan dichas partidas) restringen la cantidad de análisis que un jugador puede realizar, lo que dificulta mejorar su juego de manera efectiva.

Para superar esta limitación, se ha desarrollado una aplicación de escritorio que permita a los jugadores realizar tantos análisis como deseen, sin depender de los recursos limitados de los servidores en Internet. Al permitirles analizar sus partidas en detalle, entenderán mejor sus fortalezas y debilidades, lo que les ayudará a desarrollar estrategias más efectivas. Además, al almacenar las partidas localmente, los jugadores tienen la libertad de revisarlas en cualquier momento, sin preocuparse por las restricciones de los servidores en línea.

Comparado con otras soluciones disponibles, esta aplicación ofrece una mayor flexibilidad y control sobre el análisis de partidas; brindando a los jugadores de una herramienta efectiva y conveniente para elevar su nivel.

2.2 Objetivos del Proyecto

En este proyecto, el objetivo principal es desarrollar un programa que permita analizar localmente la precisión de los movimientos en las partidas de ajedrez jugadas por un usuario, eliminando la necesidad de depender de los servidores en Internet. Para lograr esto, se establecen una serie de objetivos específicos:

1. Desarrollar una aplicación de escritorio que permita analizar las partidas de ajedrez descargadas desde Lichess.
2. Utilizar el módulo de código abierto Stockfish para llevar a cabo el análisis de las partidas en el ordenador del usuario.
3. Implementar un sistema de almacenamiento local para guardar las partidas evaluadas y evitar así posibles análisis repetidos.
4. Generar estadísticas sobre el rendimiento del jugador en diferentes ritmos de juego y periodos de tiempo, utilizando los análisis realizados.
5. Utilizar el framework Electron para el desarrollo de la aplicación de escritorio.
6. Facilitar la adición de nuevas métricas y análisis en el futuro, además de las que ya se encuentran disponibles.
7. Proporcionar una interfaz de usuario sencilla, para que sea accesible a un público amplio y no solo a usuarios avanzados.

2.3 Estudio de la Situación Actual

En este apartado se identificarán y describirán sistemas ya existentes que permitan analizar partidas de ajedrez, estudiando sus ventajas e inconvenientes. Asimismo, se evaluarán las posibles herramientas o tecnologías utilizables para el proyecto y se determinará cuáles se adaptan mejor a sus necesidades concretas.

A continuación, se describen los sistemas estudiados. Aunque todos presentan ventajas, también sufren de ciertas limitaciones previamente mencionadas: modelo de pago, dependencia de internet o interfaz poco intuitiva.

2.3.1 ChessBase

ChessBase es un popular programa de bases de datos para almacenar y buscar juegos de ajedrez que corre bajo Microsoft Windows. La base de datos ChessBase integra módulos de análisis, como Fritz, Junior, Shredder (todos productos de Chessbase), y varios módulos no comerciales, incluyendo Crafty escrito por el profesor Robert Hyatt, Comet, y Anaconda (*ChessBase - Wikipedia, la enciclopedia libre*, 2024).

Figura 2.1 Interfaz de ChessBase 17

Fuente: Chessbase (Friedel, 2022)

2.3.1.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este sistema:

- Gestión de bases de datos: Ofrece herramientas para la gestión de grandes bases de datos de partidas de ajedrez, facilitando la organización y búsqueda de partidas.
- Recursos de entrenamiento: Incluye numerosos recursos educativos, como entrenadores de tácticas y módulos de entrenamiento específicos.
- Profundidad de análisis: ChessBase permite realizar análisis detallados y profundos con motores avanzados como Fritz y Stockfish.
- Actualizaciones regulares: La empresa ofrece actualizaciones frecuentes con nuevas funcionalidades y mejoras.

2.3.1.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este sistema:

- Costo: ChessBase es un software comercial, y su versión completa puede ser bastante costosa.
- Curva de aprendizaje: La gran cantidad de funcionalidades puede resultar abrumadora para los nuevos usuarios.
- Requiere instalación: Es necesario instalar el software en un ordenador, lo que puede ser una limitación para aquellos que prefieren soluciones basadas en la web.

2.3.2 Lichess

Lichess es un servidor de ajedrez en Internet, gratuito y de código abierto. Lichess no tiene publicidad y todas las funciones son gratuitas; estas incluyen rompecabezas de ajedrez, análisis por ordenador, torneos y variantes de ajedrez (*Lichess - Wikipedia, la enciclopedia libre*, 2024).

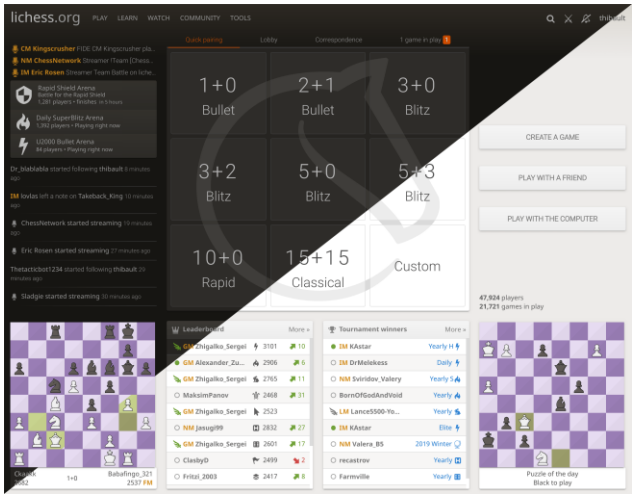


Figura 2.2 Página de inicio de Lichess

*Fuente: Wikipedia (*Lichess - Wikipedia, la enciclopedia libre*, 2024)*

2.3.2.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este sistema:

- Gratuito: Todas las funcionalidades de Lichess, incluyendo el análisis con Stockfish, son completamente gratuitas.
- Funciones colaborativas: Permite realizar análisis colaborativos y compartir partidas fácilmente con otros usuarios.
- Interfaz intuitiva: La interfaz es fácil de usar, lo que facilita la experiencia de usuario tanto para principiantes como para jugadores avanzados.
- Accesibilidad: Al ser una plataforma en línea, no requiere instalación y puede ser utilizada desde cualquier dispositivo con acceso a internet.

2.3.2.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este sistema:

- Dependencia de internet: Requiere una conexión a internet estable para su uso.
- Personalización: Las opciones de personalización son más limitadas en comparación con programas de pago como ChessBase.
- Limitaciones del motor: Aunque Stockfish es muy potente, algunas funcionalidades avanzadas de otros motores no están disponibles.
- Límite de análisis diario: Solo se pueden analizar 40 partidas por día, lo que puede ser insuficiente para usuarios que necesiten revisar muchas partidas.

2.3.3 Chess.com

Chess.com es un servidor de ajedrez en Internet con un modelo "freemium" en el que algunas funciones están disponibles de forma gratuita y otras para cuentas con suscripciones. Ofrece una gran variedad de servicios, incluyendo la posibilidad de jugar partidas, analizar juegos y acceder a contenido educativo (*Chess.com - Wikipedia, la enciclopedia libre*, 2024).

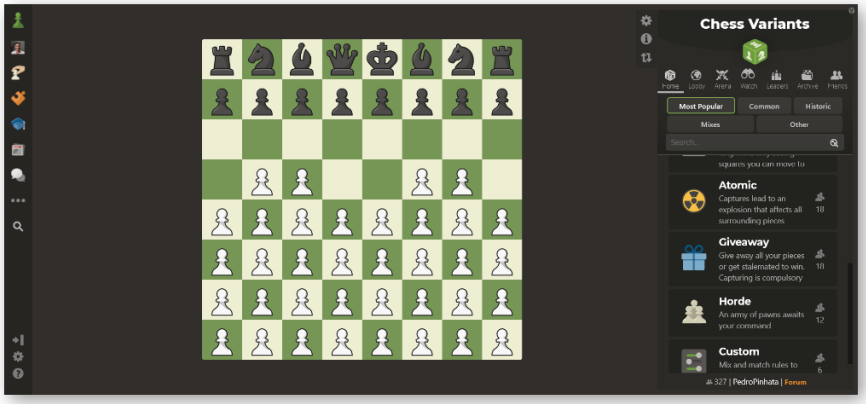


Figura 2.3 Página de inicio de Chess.com

Fuente: Wikipedia (Chess.com - Wikipedia, la enciclopedia libre, 2024)

2.3.3.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este sistema:

- Popularidad y comunidad: Tiene una gran comunidad de usuarios, lo que facilita encontrar oponentes y compartir conocimientos.
- Análisis de partidas: Utiliza motores avanzados para proporcionar análisis detallados y sugerencias de mejora.
- Variedad de herramientas: Ofrece múltiples herramientas de análisis, puzzles y lecciones de ajedrez.
- Contenido educativo: Dispone de una amplia gama de videos instructivos y artículos.

2.3.3.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este sistema:

- Interfaz publicitaria: La versión gratuita puede tener anuncios, lo cual puede resultar molesto para algunos usuarios.
- Dependencia de internet: Al igual que Lichess, necesita una conexión a internet para acceder a la mayoría de sus funcionalidades.
- Costo de la membresía: Aunque hay una versión gratuita, las funcionalidades más avanzadas requieren una suscripción de pago.
- Limitación de análisis: Solo permite un análisis por día, lo cual puede ser insuficiente para usuarios que necesiten revisar más de una partida diariamente.

2.3.4 Arena Chess GUI

Arena Chess GUI es un programa gratuito y de código abierto para análisis de ajedrez, compatible con numerosos motores. Fue desarrollado por Martin Blume y es popular entre los entusiastas que buscan una herramienta flexible y personalizable (*Arena Chess GUI*, s. f.).

Figura 2.4 Interfaz de Arena Chess GUI

Fuente: Arena Chess GUI (Arena Chess GUI, s. f.)

2.3.4.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este sistema:

- Análisis detallado: Permite realizar análisis detallados y personalizables de las partidas.
- Gratuito y de código abierto: No tiene costo y permite modificaciones por parte de la comunidad.
- Compatibilidad con múltiples motores: Soporta una amplia variedad de motores de ajedrez, incluyendo Stockfish, Komodo y otros.
- Personalización: Ofrece muchas opciones de personalización para ajustar la interfaz y las configuraciones de análisis según las necesidades del usuario.

2.3.4.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este sistema:

- Requiere instalación: Necesita ser instalado en un ordenador, lo que puede ser una limitación para algunos usuarios.
- Interfaz menos intuitiva: La interfaz puede ser menos amigable en comparación con plataformas comerciales como ChessBase.
- Menos soporte técnico: Al ser un proyecto de código abierto, el soporte técnico puede ser menos consistente que en soluciones comerciales.

2.3.5 Evaluación de Alternativas

Esta sección se divide en tres partes: la investigación de tecnologías para el desarrollo de la interfaz, la consideración de bases de datos locales para almacenar partidas y análisis, y el estudio de frameworks para la implementación de pruebas unitarias.

2.3.5.1 Interfaz de usuario

A continuación, se presentan las alternativas investigadas para implementar la interfaz, basándonos en el uso del framework Electron junto con JavaScript, TypeScript, HTML y CSS para el desarrollo de la aplicación.

2.3.5.1.1 Vue.js

Vue.js es un framework de desarrollo para crear interfaces de usuario. Se enfoca en la parte visual de las aplicaciones web, lo que lo hace ideal para proyectos que necesitan una experiencia de usuario ágil.

2.3.5.1.1.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este framework:

- Fácil de aprender: Vue.js tiene una curva de aprendizaje suave, lo que lo hace atractivo para desarrolladores de todos los niveles.
- Buen rendimiento: Ofrece un buen rendimiento en aplicaciones de una sola página.
- Flexibilidad: Permite integrar fácilmente con otros proyectos y bibliotecas.

2.3.5.1.1.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este framework:

- Ecosistema en crecimiento: Aunque está creciendo rápidamente, el ecosistema de Vue.js aún es más pequeño que el de otras opciones.
- Documentación menos extensa: Aunque la documentación es buena, no es tan completa como la de otros marcos.

2.3.5.1.2 Angular

Angular es un framework completo desarrollado por Google para crear aplicaciones web. Proporciona una solución integral, incluyendo enrutamiento y manejo de estado.

2.3.5.1.2.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este framework:

- Tipado fuerte: Escrito en TypeScript, proporciona una experiencia de desarrollo más segura.
- Gran soporte: Respaldado por Google, tiene una comunidad activa y sólida documentación.
- Completo: Angular ofrece todo lo necesario para desarrollar una aplicación web, lo que lo hace conveniente para proyectos grandes.

2.3.5.1.2.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este framework:

- Curva de aprendizaje pronunciada: Puede ser complicado para los principiantes debido a su complejidad.
- Exceso de funcionalidades: Puede ser demasiado robusto para proyectos más pequeños.

2.3.5.1.3 React

React es una biblioteca de JavaScript para construir interfaces de usuario. Es mantenida por Facebook y tiene una gran base de usuarios y empresas que lo respaldan.

2.3.5.1.3.1 Ventajas

En esta sección se discutirán las ventajas que ofrece esta biblioteca:

- Componentes reutilizables: Fomenta la creación de componentes que se pueden reutilizar fácilmente, lo que facilita el mantenimiento y la escalabilidad.
- Gran ecosistema: Tiene una amplia gama de bibliotecas y herramientas complementarias, así como una documentación extensa.
- Virtual DOM: Utiliza un Virtual DOM para mejorar el rendimiento, lo que permite actualizaciones eficientes de la interfaz de usuario.

2.3.5.1.3.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece esta biblioteca:

- Curva de aprendizaje: Puede ser empinada para algunos desarrolladores, especialmente para aquellos nuevos en la programación declarativa.
- Necesita herramientas adicionales: Para construir una aplicación completa, es posible que necesites integrarlo con otras herramientas y bibliotecas.

2.3.5.2 Bases de datos

A continuación, se presentan las alternativas evaluadas para almacenar las partidas y análisis. Es importante que esta solución sea local, para evitar la dependencia de internet, y que sea compatible con el resto de las tecnologías seleccionadas.

2.3.5.2.1 SQLite

SQLite es una base de datos ligera y fácil de usar que viene en forma de una biblioteca. Es perfecta para aplicaciones locales y móviles porque no requiere un servidor separado.

2.3.5.2.1.1 Ventajas

En esta sección se discutirán las ventajas que ofrece esta base de datos:

- Ligera: Es pequeña y rápida, ideal para aplicaciones con recursos limitados.
- Sin configuración: No necesitas configurar un servidor, lo cual simplifica su uso.
- Integración fácil: Funciona bien con muchos lenguajes de programación y plataformas.

2.3.5.2.1.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece esta base de datos:

- Concurrencia limitada: No maneja bien muchas escrituras simultáneas.
- Capacidad: No está diseñada para manejar bases de datos muy grandes.
- Funcionalidades avanzadas: Le faltan algunas características avanzadas que tienen otras bases de datos más grandes como PostgreSQL o MySQL.

2.3.5.2.2 NeDB

NeDB es una base de datos No-SQL para Node.js, diseñada para ser simple y fácil de usar en aplicaciones locales. Se parece a MongoDB, pero funciona completamente de manera local.

2.3.5.2.2.1 Ventajas

En esta sección se discutirán las ventajas que ofrece esta base de datos:

- Flexible: Permite consultas avanzadas y tiene un esquema flexible.
- Fácil de usar: Su sintaxis es similar a MongoDB, lo que la hace fácil de aprender.
- Integración con Node.js: Está hecha para trabajar bien con aplicaciones de Node.js.

2.3.5.2.2.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece esta base de datos:

- Rendimiento: No es tan rápida para grandes volúmenes de datos.
- Concurrencia: Aunque soporta concurrencia, no es tan robusta como bases de datos más grandes.
- Capacidades limitadas: Carece de algunas características avanzadas que tienen sistemas de bases de datos más completos.

2.3.5.2.3 IndexedDB

IndexedDB es una base de datos para aplicaciones web que permite almacenar grandes cantidades de datos directamente en el navegador. Es ideal para aplicaciones que necesitan guardar datos localmente.

2.3.5.2.3.1 Ventajas

En esta sección se discutirán las ventajas que ofrece esta base de datos:

- Asíncrona: Las operaciones son asíncronas, lo que ayuda a mantener la aplicación rápida y responsiva.
- Capacidad: Puede manejar grandes cantidades de datos, perfecto para aplicaciones web.
- No-SQL: Su modelo No-SQL permite almacenar datos complejos de manera flexible.

2.3.5.2.3.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece esta base de datos:

- Compatibilidad: No todos los navegadores implementan IndexedDB de la misma manera, lo que puede generar problemas.
- Soporte limitado: Tiene menos capacidades para consultas avanzadas en comparación con bases de datos SQL.
- Complejidad: Su API puede ser complicada y difícil de usar.

2.3.5.3 Pruebas unitarias

A continuación, se presentan las alternativas estudiadas para la implementación de pruebas unitarias. Es esencial que esta solución sea compatible y cuente con documentación que explique su uso en conjunto con las demás tecnologías seleccionadas.

2.3.5.3.1 Jest

Jest es un framework de pruebas desarrollado por Facebook. Es ampliamente utilizado en el ecosistema de React debido a su facilidad de uso y a sus características integradas que facilitan la configuración y ejecución de pruebas.

2.3.5.3.1.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este framework:

- Fácil de configurar: Jest viene con configuraciones predeterminadas que funcionan bien para la mayoría de los proyectos, lo que reduce el tiempo de configuración.
- Snapshots: Permite la creación de snapshots, una forma poderosa de verificar que la interfaz de usuario no ha cambiado inesperadamente.
- Soporte para mocks: Tiene un robusto sistema de mocks integrado que facilita la simulación de dependencias durante las pruebas.
- Pruebas en paralelo: Ejecuta pruebas en paralelo, lo que puede reducir significativamente el tiempo de ejecución.

2.3.5.3.1.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este framework:

- Curva de aprendizaje para principiantes: Aunque es fácil de configurar, los principiantes pueden encontrar algunas de sus características avanzadas complicadas al inicio.
- Rendimiento en proyectos muy grandes: En algunos casos, el rendimiento puede disminuir en proyectos extremadamente grandes si no se configura adecuadamente.

2.3.5.3.2 Mocha

Mocha es un framework de pruebas para JavaScript que se ejecuta en Node.js y en el navegador, lo que lo hace versátil para diferentes entornos de prueba.

2.3.5.3.2.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este framework:

- Flexibilidad: Mocha es altamente configurable y puede ser adaptado para diferentes necesidades y estilos de pruebas.
- Popularidad y comunidad: Tiene una amplia comunidad y muchos recursos disponibles para aprender y solucionar problemas.
- Compatibilidad con otras bibliotecas: Funciona bien con otras bibliotecas de aserción como Chai y Sinon, permitiendo un enfoque modular en las pruebas.

2.3.5.3.2.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este framework:

- Configuración inicial: Requiere más configuración inicial en comparación con Jest, especialmente cuando se integra con otras herramientas como Chai.
- Paralelización manual: No ejecuta pruebas en paralelo por defecto, lo que puede resultar en tiempos de prueba más largos si no se configura adecuadamente.

2.3.5.3.3 Jasmine

Jasmine es un framework de pruebas para JavaScript, diseñado para ser sencillo y fácil de usar. Es uno de los más antiguos y maduros en el ecosistema de pruebas JavaScript.

2.3.5.3.3.1 Ventajas

En esta sección se discutirán las ventajas que ofrece este framework:

- Soporte para pruebas asíncronas: Maneja de manera efectiva las pruebas asíncronas, lo que es crucial para aplicaciones modernas.
- Sin dependencias externas: No requiere dependencias externas, lo que simplifica su uso y configuración inicial.
- Sintaxis intuitiva: Ofrece una sintaxis clara y fácil de entender, lo que facilita la escritura de pruebas.

2.3.5.3.3.2 Inconvenientes

En esta sección se discutirán los inconvenientes que ofrece este framework:

- Menos integraciones modernas: Aunque es robusto, puede carecer de algunas de las integraciones más modernas y convenientes que ofrecen Jest y Mocha.
- Configuración adicional para mocks: Aunque se puede utilizar con bibliotecas de mocks, esto requiere configuraciones adicionales, lo que puede aumentar la complejidad de la configuración inicial.

Capítulo 3. Aspectos Teóricos

En este capítulo se comentarán los conceptos, herramientas y tecnologías más importantes usados en este proyecto, justificando su elección cuando sea necesario.

3.1 MÉTRICA Versión 3

En este apartado se describirá la metodología MÉTRICA Versión 3.

3.1.1 Descripción

Métrica V3 es una metodología del Gobierno de España para planificar, desarrollar y mantener sistemas de información en las administraciones públicas. Basada en estándares internacionales, busca:

- Mejorar la productividad de los departamentos de TI.
- Optimizar la operación y mantenimiento del software.
- Definir sistemas que apoyen los objetivos organizacionales.
- Crear software que cumpla con las necesidades de los usuarios.
- Facilitar la comunicación entre todos los participantes del proyecto.

Organiza las actividades en procesos específicos y es compatible con diversas tecnologías y métodos de desarrollo, además de permitir la automatización de tareas mediante herramientas disponibles en el mercado (*PAe - Métrica v.3*, 2001).

Figura 3.1 Procesos de Métrica V3

Fuente: El mínimo viable (Sánchez García, 2020)

Este documento está desarrollado usando esta metodología, pero de forma adaptada, contemplando solo aquellos apartados que se han considerado adecuados para el proyecto.

3.2 Visual Studio Code

En este apartado se describirá el editor de código fuente Visual Studio Code y se justificará su uso en el proyecto.

3.2.1 Descripción

Visual Studio Code es un editor de código fuente desarrollado por Microsoft con soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes y tiempos de ejecución (*Documentation for Visual Studio Code*, s. f.; *Visual Studio Code - Wikipedia, la enciclopedia libre*, 2024).

Figura 3.2 Logo de VS Code

3.2.2 Justificación de uso

Además de ser un entorno que ya he utilizado anteriormente, es gratuito y cuenta con extensiones muy conocidas y bien mantenidas para el framework y los lenguajes usados en el proyecto, lo que facilitará el proceso de desarrollo.

3.3 Electron

En este apartado se describirá el framework Electron y se justificará su uso en el proyecto.

3.3.1 Descripción

Electron es un framework de código abierto creado por Cheng Zhao, y ahora desarrollado por GitHub. Permite el desarrollo de aplicaciones gráficas de escritorio usando componentes del lado del cliente y del servidor originalmente desarrolladas para aplicaciones web: Node.js del lado del servidor y Chromium como interfaz. Electron es el framework gráfico detrás de muchos proyectos de código abierto importantes, incluyendo a Microsoft Visual Studio Code y el cliente de escritorio freeware del servicio de mensajería instantánea Discord (*Electron (software) - Wikipedia, la enciclopedia libre*, 2024).

Figura 3.3 Logo de Electron

3.3.2 Justificación de uso

El uso de este framework está estipulado en la propuesta original del proyecto.

3.4 React

En este apartado se describirá la biblioteca React y se justificará su uso en el proyecto.

3.4.1 Descripción

React es una biblioteca de código abierto en JavaScript, creada para simplificar el desarrollo de interfaces de usuario en aplicaciones de página única. Es mantenida por Facebook y la comunidad de software libre (*React - Wikipedia, la enciclopedia libre*, 2024). Actualmente, se utiliza en las páginas principales de sitios como Netflix, Instagram y Uber, entre otros.

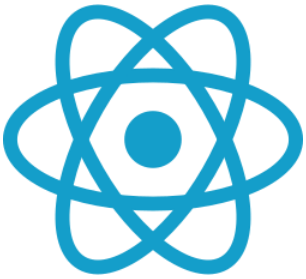


Figura 3.4 Logo de React

Esta biblioteca se utilizará para implementar la interfaz de usuario del proyecto, junto con varias bibliotecas como MUI, lo que facilitará la visualización de elementos específicos, como el tablero de ajedrez.

3.4.2 Justificación de uso

Como se mencionó anteriormente, React facilita la escalabilidad del código debido a sus componentes reutilizables y dispone de un gran ecosistema de bibliotecas, lo cual me permitirá ahorrar tiempo de desarrollo en aspectos de menor importancia.

Además, tengo una amplia experiencia previa con esta biblioteca y acceso a una extensa documentación y numerosos ejemplos sobre su uso junto con Electron.

3.5 JavaScript y TypeScript

En este apartado se describirán los lenguajes de programación JavaScript y TypeScript, y se justificará su uso en el proyecto.

3.5.1 Descripción

JavaScript es un lenguaje de programación que se caracteriza por ser ligero, interpretado o compilado justo a tiempo, y cuenta con funciones de primera clase. Se considera un dialecto del estándar ECMAScript y es mantenido por la Fundación Mozilla (*JavaScript | MDN*, s. f.).

Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas y JavaScript del lado del servidor. Su uso en aplicaciones externas a la web, como en documentos PDF o aplicaciones de escritorio (mayoritariamente widgets) es también significativo (*JavaScript - Wikipedia, la enciclopedia libre*, 2024).

Figura 3.5 Logo de JavaScript

TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases. TypeScript es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor, o extensiones para programas (Node.js y Deno) (*TypeScript - Wikipedia*, 2024).

Figura 3.6 Logo de TypeScript

La mayor parte del proyecto se desarrollará utilizando TypeScript. Sin embargo, para las partes relacionadas con el motor de análisis Stockfish se utilizará JavaScript, ya que estas se escribieron originalmente en ese lenguaje y se prefirió mantenerlas así para evitar posibles complicaciones con el tipado.

3.5.2 Justificación de uso

El uso de estos lenguajes está estipulado en la propuesta original del proyecto.

3.6 SQLite

En este apartado se describirá el sistema de gestión de bases de datos SQLite y se justificará su uso en el proyecto.

3.6.1 Descripción

SQLite es una biblioteca en lenguaje C que implementa un motor de base de datos SQL pequeño, rápido, autónomo, de alta fiabilidad y completo. SQLite es el motor de bases de datos más utilizado del mundo: está integrado en todos los teléfonos móviles y en la mayoría de los ordenadores, y viene incluido en innumerables aplicaciones que se utilizan a diario. El código fuente de SQLite es de dominio público y su uso es libre para todo el mundo (*SQLite Home Page*, 2024) .

Figura 3.7 Logo de SQLite

Esta base de datos se utilizará tanto para almacenar las partidas descargadas como para guardar los análisis realizados, evitando así el malgasto de recursos.

3.6.2 Justificación de uso

Como se mencionó anteriormente, SQLite es bastante ligera y fácil de configurar con Electron, y los problemas de concurrencia limitada se pueden solucionar mediante el uso de transacciones. Se optó por una base de datos relacional debido a su soporte nativo para transacciones ACID y su optimización para escalabilidad vertical, características ideales para aplicaciones que requieren procesamiento transaccional rápido. Además, tengo una amplia experiencia previa con esta biblioteca, lo que facilitará su implementación.

3.7 Stockfish

En este apartado se describirá el motor Stockfish y se justificará su uso en el proyecto.

3.7.1 Descripción

Stockfish, creado por Tord Romstad, Marco Costalba y Joona Kiiski, es un motor de ajedrez ampliamente reconocido por su potencia y popularidad. Surgió de un proyecto llamado Glaurung y ha sido desarrollado colaborativamente por una comunidad de programadores y amantes del ajedrez. Destaca por su capacidad para analizar posiciones complejas con una profundidad impresionante y es utilizado por jugadores profesionales, entrenadores y aficionados para diversos fines, como analizar partidas, estudiar aperturas y mejorar tácticas.

En torneos de ajedrez para ordenadores, Stockfish ha demostrado su superioridad en múltiples ocasiones. Su código abierto ha facilitado su integración en diversas plataformas y aplicaciones de ajedrez, lo que lo hace accesible para una amplia variedad de usuarios (Fuentes citadas).

Recientemente, Stockfish ha sido optimizado para trabajar en conjunto con NNUE (Redes Neuronales de Uso Eficiente en Nodos), una tecnología que combina la potencia de cálculo clásica con el aprendizaje profundo de las redes neuronales. Esta integración ha mejorado aún más la fuerza de juego de Stockfish, permitiéndole evaluar posiciones de manera más precisa y jugar de forma más sólida y creativa. Esto lo convierte en una herramienta aún más valiosa para jugadores y analistas de ajedrez en todo el mundo (*Introducing NNUE Evaluation - Stockfish - Strong open-source chess engine*, 2020).

Figura 3.8 Logo de Stockfish

En el proyecto, se empleará para analizar cada posición de la partida de forma dinámica, calculando la ventaja del jugador blanco en centipawns. Los centipawns son una unidad de medida que indica la ventaja o desventaja de una posición en el ajedrez. Cada centipawn equivale a una centésima parte de un peón. Por ejemplo, una diferencia de 100 centipawns indica una ventaja de un peón para uno de los jugadores.

Este valor se utilizará para evaluar las imperfecciones y errores cometidos por ambos jugadores a lo largo de la partida. Además, se determinará la precisión de los movimientos realizados durante el juego, utilizando una fórmula matemática propuesta por Lichess:

La precisión se calcula mediante la comparación de las posiciones antes y después de cada movimiento. Primero, se determina la probabilidad de ganar la partida a partir de la evaluación de la posición en centipawns por Stockfish. Luego, se utiliza una fórmula para calcular la precisión del movimiento realizado, teniendo en cuenta cómo cambió la probabilidad de ganar después de ese movimiento. Esta precisión se expresa como un porcentaje, donde un 100% significa que se jugaron los movimientos preferidos por Stockfish, mientras que un 0% indica que solo se realizaron movimientos terribles (*Lichess Accuracy metric • lichess.org*, s. f.).

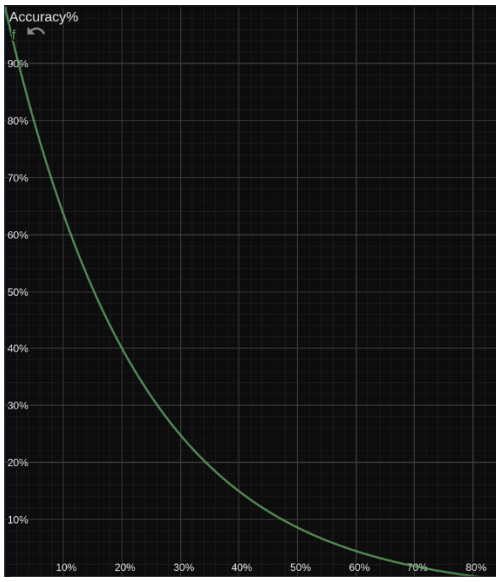


Figura 3.9 Precisión% por diferencia de Victoria% de una posición a la siguiente

Fuente: Lichess (Lichess Accuracy metric • lichess.org, s. f.)

3.7.2 Justificación de uso

El uso de este motor está estipulado en la propuesta original del proyecto.

3.7.3 Fuentes

Página de Stockfish

Artículo de Wikipedia

Glosario de Chess.com

Métrica de precisión de Lichess

Introducción de evaluación NNUE en Stockfish

3.8 Jest

En este apartado se describirá el framework de pruebas Jest y se justificará su uso en el proyecto.

3.8.1 Descripción

Jest es un framework de pruebas para JavaScript creado por Facebook, diseñado principalmente para proyectos basados en React y otras bibliotecas similares. Facilita la escritura y ejecución de pruebas unitarias e integración con características como mocks automáticos y snapshots (Jest · Delightful JavaScript Testing, s. f.).

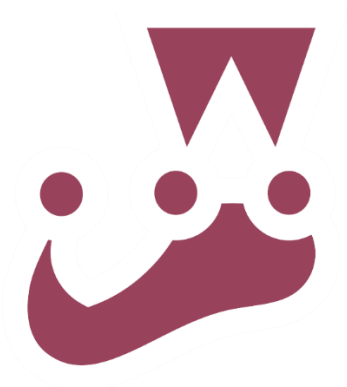


Figura 3.10 Logo de Jest

Este framework se usará para implementar las pruebas unitarias del sistema, junto con React Testing Library.

3.8.2 Justificación de uso

Jest es conocido por su facilidad de uso con React y por contar con una gran comunidad, lo que significa que hay muchos recursos y documentación disponibles. Además, ofrece soporte para mocks, lo cual es particularmente útil en el contexto de Electron, y lo he utilizado en varios proyectos anteriores, lo que facilitará el desarrollo de las pruebas.

Capítulo 4. Planificación del Proyecto y Presupuesto Iniciales

En este capítulo se desarrollarán la planificación y presupuestos iniciales del proyecto.

4.1 Planificación Inicial

Una vez establecidos el alcance y los objetivos del proyecto, se ha elaborado una planificación inicial para su ejecución. Esta planificación se seguirá durante el desarrollo del trabajo y al final se reflejarán los cambios realizados en una versión actualizada.

La estimación de la duración de los módulos se realizó, antes de comenzar el proyecto, basándose en la experiencia previa. En la figura 4.1 se muestra un cronograma indicando las fechas previstas de comienzo y final de cada una de las tareas, y en la figura 4.2 se presenta dicho cronograma utilizando un diagrama de Gantt:

Nombre de tarea	Duración	Comienzo	Fin
Documentación inicial	**3,88 días**	**lun 29/01/24**	**dom 11/02/24**
Reunión de arranque	2 hrs	lun 29/01/24	lun 29/01/24
Planteamiento de la motivación y los objetivos	2 hrs	vie 02/02/24	vie 02/02/24
Estudio de la situación actual	6 hrs	sáb 03/02/24	sáb 03/02/24
Estudio de los conceptos, herramientas y tecnologías a usar	6 hrs	dom 04/02/24	lun 05/02/24
Elaboración de la planificación inicial	6 hrs	vie 09/02/24	sáb 10/02/24
Elaboración del presupuesto inicial	6 hrs	sáb 10/02/24	dom 11/02/24
Análisis	**3 días**	**lun 12/02/24**	**sáb 24/02/24**
Determinación del alcance del sistema	2 hrs	lun 12/02/24	lun 12/02/24
Elaboración de requisitos y casos de uso	12 hrs	vie 16/02/24	dom 18/02/24
Definición de clases	4 hrs	lun 19/02/24	vie 23/02/24
Diseño de interfaces de usuario	4 hrs	sáb 24/02/24	sáb 24/02/24
Especificación del plan de pruebas	2 hrs	sáb 24/02/24	sáb 24/02/24
Diseño	**2,63 días**	**dom 25/02/24**	**dom 03/03/24**
Definición de la arquitectura del sistema y elaboración de diagramas	12 hrs	dom 25/02/24	sáb 02/03/24
Diseño de la base de datos	2 hrs	sáb 02/03/24	sáb 02/03/24
Especificación técnica del plan de pruebas	4 hrs	dom 03/03/24	dom 03/03/24
Implementación	**23,13 días**	**lun 04/03/24**	**sáb 01/06/24**
Elaboración de la base del programa	2 hrs	lun 04/03/24	lun 04/03/24
Front-end	**6,13 días**	**vie 08/03/24**	**sáb 30/03/24**
Desarrollo de la página de inicio	8 hrs	vie 08/03/24	sáb 09/03/24
Desarrollo de la página con la lista de partidas	16 hrs	dom 10/03/24	dom 17/03/24
Desarrollo de la página de análisis	16 hrs	dom 17/03/24	dom 24/03/24
Desarrollo de la página de estadísticas	7 hrs	lun 25/03/24	sáb 30/03/24
Back-end	**15,88 días**	**sáb 30/03/24**	**sáb 01/06/24**
Desarrollo del servicio de importación de partidas	16 hrs	sáb 30/03/24	sáb 06/04/24
Integración con Stockfish	64 hrs	dom 07/04/24	vie 10/05/24
Integración con la base de datos	32 hrs	sáb 11/05/24	sáb 25/05/24
Desarrollo del servicio de cálculo de estadísticas	15 hrs	sáb 25/05/24	sáb 01/06/24
Pruebas	32 hrs	dom 02/06/24	dom 16/06/24
Reuniones de seguimiento	**17,13 días**	**sáb 30/03/24**	**sáb 01/06/24**
Muestra del prototipo inicial	1 hr	sáb 30/03/24	sáb 30/03/24
Muestra de la aplicación final	1 hr	sáb 01/06/24	sáb 01/06/24
Elaboración de los manuales del sistema	4 hrs	lun 17/06/24	vie 21/06/24
Finalización de la documentación	4 hrs	sáb 22/06/24	sáb 22/06/24
Revisión de la documentación	2 hrs	sáb 22/06/24	sáb 22/06/24

Figura 4.1 Cronograma de la planificación inicial

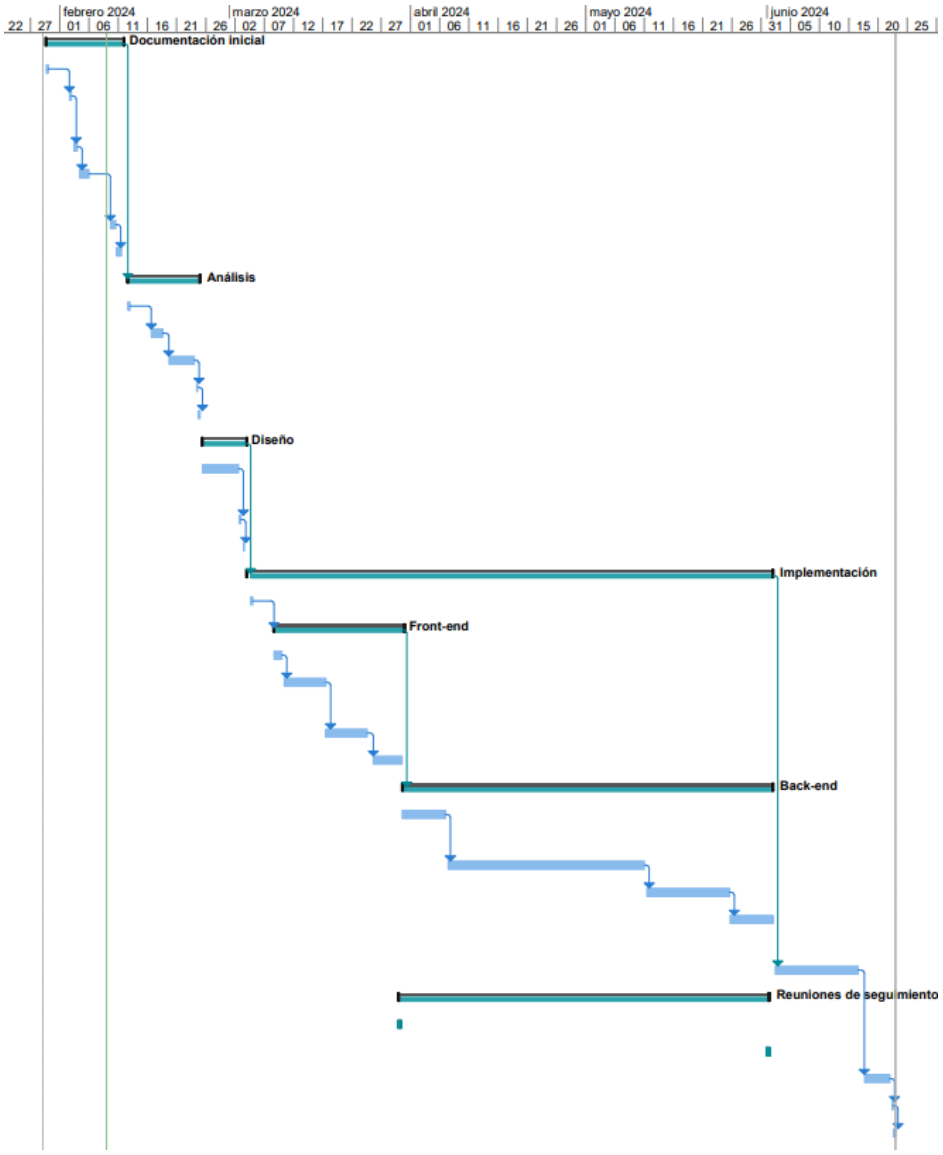


Figura 4.2 Diagrama de Gantt de la planificación inicial

Como se puede observar en la figura 4.1, hay un total de 36 tareas por completar, de las cuales 7 son tareas resumen. Se comenzó llevando a cabo una reunión con el director de este proyecto, con el fin de comprender la propuesta y sus objetivos, seguida de un estudio de sistemas similares y de las tecnologías o herramientas a usar en el proyecto.

Una vez tratados dichos puntos, se plantearon esta planificación inicial y su correspondiente presupuesto, comentado en el siguiente apartado.

Más tarde se determinaría el alcance del sistema, se elaborarían los requisitos y casos de uso, se definirían sus clases, se diseñarían las interfaces de usuario y se especificaría el plan de pruebas.

Seguidamente, en la fase de diseño se abordaría la arquitectura del sistema, acompañándola de diagramas necesarios para su comprensión y de la especificación técnica del plan de pruebas.

Durante los próximos tres meses, se implementaría la aplicación de escritorio, basándose en la información recopilada anteriormente. Esta fase se dividiría en dos partes claramente diferenciadas: el desarrollo del front-end y del back-end. Cabe destacar que cada una de estas etapas será seguida por una reunión con el director del proyecto, con el fin de mostrarle el prototipo y el estado final de la aplicación.

Con la base del programa ya preparada, se empezaría a desarrollar la página de inicio, la cual permitiría la importación de partidas, y la página que contendría una lista de estas. Finalmente, se añadirían un par de páginas para visualizar los análisis y estadísticas derivados de dichas partidas.

Para dotar de funcionalidad a dicha interfaz, se comenzaría añadiendo un método que permita la importación masiva de partidas desde un sitio web, garantizando que la aplicación no se cuelgue. Con partidas reales disponibles, se podría iniciar la integración con Stockfish, estimada en aproximadamente un mes (o 64 horas de trabajo), debido a la falta de experiencia previa en la comunicación con binarios y en el tratamiento de los resultados para obtener datos coherentes. Finalmente, se integraría una base de datos para almacenar las partidas descargadas y los análisis realizados, y se construiría una serie de consultas para extraer estadísticas relevantes.

Durante las dos próximas semanas, se agregarían las pruebas mencionadas en la especificación para validar el correcto funcionamiento del programa. Luego, se procedería a la elaboración de manuales, marcando así el fin de la documentación.

4.1.1 Calendario de recursos

Como seré el encargado de asumir todos los roles de este proyecto, se ha creado un único calendario de recursos. Se ha tenido en cuenta la participación en prácticas de empresa y la asistencia a la universidad para poder compaginarlas, resultando en un total de 14 horas laborables a la semana:

- Lunes y viernes de 16:00 a 18:00.
- Sábados de 10:00 a 14:00 y de 16:00 a 18:00.
- Domingos de 10:00 a 14:00.

Se planea completar el proyecto en un total de 290 horas. Sin embargo, la planificación está sujeta a posibles cambios debido a imprevistos, los cuales no deberían afectar la finalización del proyecto dentro del plazo previsto.

4.2 Presupuesto Inicial

Para elaborar el presupuesto inicial, se han considerado las tareas planificadas junto con su duración, así como los recursos utilizados en el proyecto:

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	Recursos Humanos				
1.1	Documentación	40	100%	13,00 €	520,00 €
1.2	Análisis	24	100%	18,00 €	432,00 €
1.3	Diseño	18	100%	30,00 €	540,00 €
1.4	Implementación	176	100%	17,00 €	2992,00 €
1.5	Pruebas	32	100%	15,00 €	480,00 €
2	Recursos Software				
2.1	Microsoft Windows 10 Home	2	8%	145,00 €	24,17 €
2.2	Microsoft Office 2021	1	100%	0,00 €	0,00 €
2.3	Microsoft Project 2016	1	100%	0,00 €	0,00 €
2.4	Visual Studio Code	2	100%	0,00 €	0,00 €
3	Recursos Hardware				
3.1	Ordenador de sobremesa	1	8%	1800,00 €	150,00 €
3.2	Portátil	1	8%	800,00 €	66,67 €
4	Costes indirectos				
4.1	Electricidad	5	100%	55,04 €	275,22 €
4.2	Internet	5	100%	70,00 €	350,00 €
Subtotal					5830,05 €
Beneficio (12%)					699,61 €
TOTAL					**6529,66** €

Tabla 4.1 Presupuesto interno inicial

El precio/hora de los profesionales asignados se estimó después de un breve estudio inicial. Los costos por hora son los siguientes:

- Responsable de la documentación: 13€/h.
- Arquitecto de software: 30€/h.
- Programador: 17€/h.
- Analista: 18€/h.
- Tester: 15€/h.

Para calcular el porcentaje de amortización de los equipos informáticos y sus sistemas operativos, se utilizó el siguiente método: suponiendo que la vida útil de estos equipos es de 5 años, o 60 meses, y considerando que el proyecto tiene una duración aproximada de 5 meses, se puede calcular la amortización como el porcentaje del tiempo total de vida útil que corresponde al tiempo de uso en el proyecto. En este caso, 5 meses de uso sobre un total de 60 meses de vida útil representa un 8,34% de amortización.

El precio de la electricidad se determinó calculando el promedio de los precios de los últimos seis meses. Por otro lado, los precios de los demás recursos ya eran conocidos previamente o se obtuvieron consultando los puntos oficiales de venta.

Las licencias de Microsoft Office y Project fueron proporcionadas por la facultad, mientras que Visual Studio Code está disponible de forma gratuita.

4.2.1 Presupuesto Simplificado (Cliente)

En esta sección, se presenta el presupuesto simplificado que se entrega al cliente, el cual se encuentra disponible en la siguiente tabla:

Concepto	Total (€)
Documentación del proyecto	735,95 €
Análisis y diseño del sistema	1.198,75 €
Implementación	3.942,19 €
Pruebas	652,76 €
Subtotal	6.529,66 €
IVA (21%)	1.371,23 €
TOTAL	**7.900,89 €**

Tabla 4.2 Presupuesto inicial del cliente

Los costos de análisis y diseño se han combinado, y los gastos indirectos y el margen de beneficio se han distribuido entre todos los elementos del proyecto según su proporción en la duración planificada. Además, se ha aplicado el margen del 21% vigente a febrero de 2024.

Concepto	% Duración
Documentación del proyecto	14%
Análisis y diseño del sistema	14%
Implementación	61%
Pruebas	11%

Tabla 4.3 Proporción de cada concepto en la duración

Capítulo 5. Análisis

Este capítulo incluye la especificación completa de los requisitos y la documentación detallada del análisis de la aplicación, a partir de la cual se elaborará posteriormente el diseño.

5.1 Definición del Sistema

En este apartado se definirá el sistema, centrándose en su alcance en lugar de repetir lo mencionado en apartados anteriores.

5.1.1 Determinación del Alcance del Sistema

Como se ha mencionado en numerosas ocasiones, el sistema a desarrollar consistirá en una aplicación de escritorio que permitirá analizar la precisión de los movimientos jugados por un usuario en ajedrez.

Las partidas se podrían importar en formato PGN, ampliamente utilizado en el sector, o desde sitios como Chess.com. No obstante, se ha optado por importarlas usando el formato ND-JSON desde la API de Lichess, ya que es oficial y está bien documentada. Además, este formato permite manejar las partidas de manera más eficiente, descargándolas poco a poco para evitar que la aplicación se cuelgue.

Las partidas descargadas serán ordenadas cronológicamente y solo podrán ser filtradas por el nombre del rival. Esto se debe a que muchas de las opciones de búsqueda proporcionadas por algunos portales podrían confundir a los usuarios menos expertos y son exclusivas de las partidas jugadas en esos portales.

En cuanto a los análisis, nos limitaremos a una configuración específica que proporcione buenos resultados, centrándose en la eficiencia. Esto permitirá el procesamiento masivo de partidas, manteniendo la coherencia entre los valores obtenidos y presentándolos de forma transparente para los usuarios sin conocimientos avanzados.

Por último, se ha decidido mostrar un número limitado de estadísticas, centradas en las condiciones de victoria y los errores cometidos, a través de gráficos que faciliten su visualización. Esto servirá como ejemplo de lo que se podría añadir a futuro.

En resumen, se pretende implementar una aplicación robusta y sencilla que priorice la eficiencia y la escalabilidad sobre el detalle y la variedad.

5.1.2 Diagrama de contexto

Para mostrar los límites del sistema e identificar los actores externos involucrados, se presenta el siguiente diagrama de contexto:

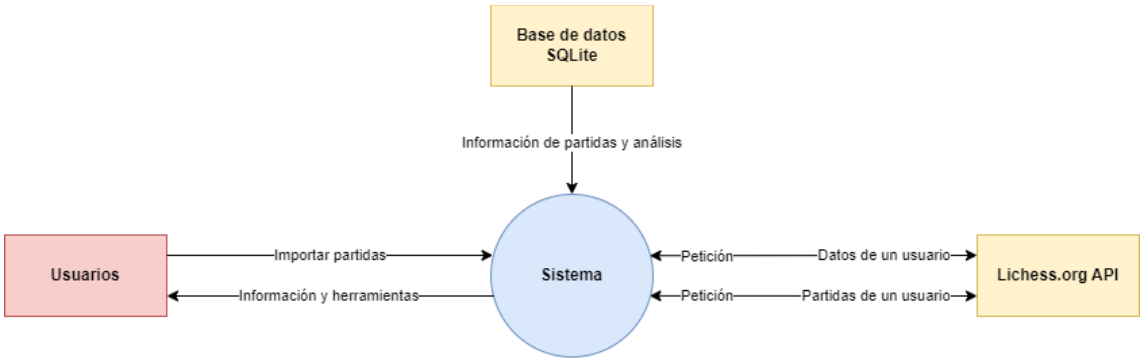


Figura 5.1 Diagrama de contexto

Las interacciones entre el sistema y los actores externos se centran en los usuarios que utilizan la aplicación, en la base de datos SQLite y en la API de Lichess. Aunque en secciones posteriores se detallarán el rol y las capacidades de los usuarios, así como el diseño de la base de datos, es necesario describir la parte de la API de Lichess que se utilizará.

5.1.2.1 Obtener datos públicos de un usuario

Antes de descargar las partidas de un usuario, será necesario verificar si dicho usuario existe en Lichess y si ha jugado partidas en este sitio. Para ello, se realizará una petición GET a su API con el formato "/api/user/{username}", donde "username" es una cadena de texto con el nombre del usuario cuyos datos públicos queremos consultar.

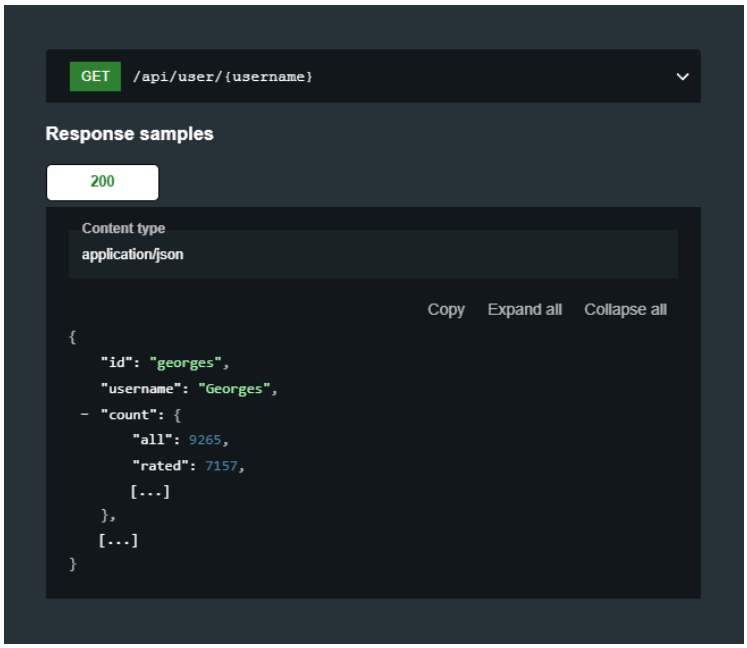


Figura 5.2 Obtener datos públicos de un usuario (Lichess)

Si el usuario no existe, la respuesta devolverá el estado "404". Para determinar cuántas partidas ha jugado, se accederá al campo "all" dentro de "count". Este valor se verificará para asegurarse de que sea distinto de cero y se comparará con el número de partidas de dicho usuario disponibles en la base de datos, a fin de determinar si es necesario descargar partidas.

5.1.2.2 Exportar partidas de un usuario

Para descargar las partidas de un usuario, se realizará una petición GET con el formato "api/games/user/{username}", donde "username" es el nombre del usuario. Siguiendo las recomendaciones de Lichess, las partidas se exportarán en formato ND-JSON utilizando un stream, ya que la respuesta puede ser muy extensa. Como el sistema permitirá importar y analizar las partidas de cualquier usuario, se hará una petición anónima (sin necesidad de autenticarse), descargando hasta 20 partidas por segundo.

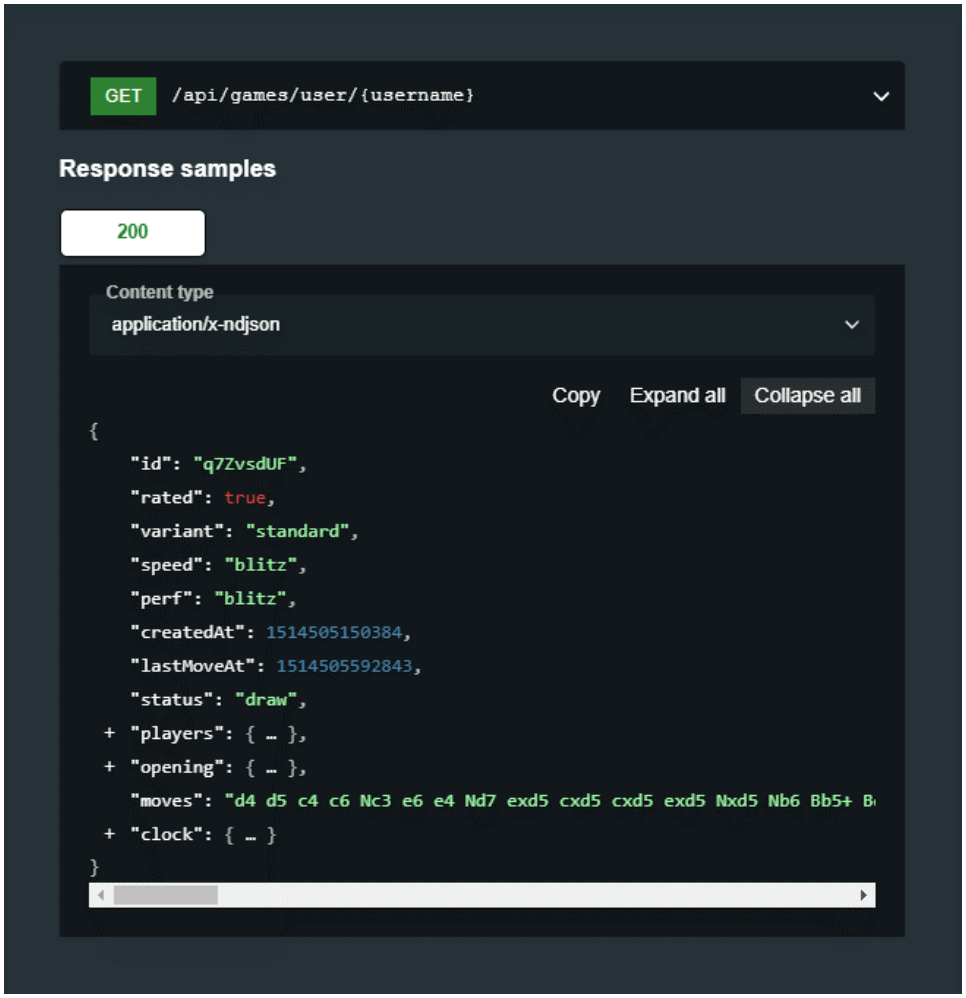


Figura 5.3 Exportar partidas de un usuario (Lichess)

No se explicarán los campos de cada partida en esta sección, ya que serán descritos en detalle en apartados posteriores cuando sea necesario. Para más información, visita Lichess API.

5.2 Requisitos del Sistema

En este apartado se detallan las especificaciones completas y los atributos de calidad del sistema a través de requisitos, describiendo qué debe hacer y cómo debe hacerlo.

5.2.1 Obtención de los Requisitos del Sistema

En esta sección se enumerarán los requisitos funcionales y no funcionales del proyecto.

5.2.1.1 Requisitos funcionales

5.2.1.1.1 Importar partidas

RCarga-1. El sistema debe permitir importar partidas a un usuario
- RCarga-1.1. Solicitará el nombre de usuario para la importación
  - RCarga-1.1.1. Es un dato obligatorio
  - RCarga-1.1.2. El sistema comprobará que dicho usuario haya jugado partidas
- RCarga-1.2. Si el nombre de usuario no es validado correctamente entonces el sistema mostrará un mensaje de error
- RCarga-1.3. Si el nombre de usuario es validado correctamente, el sistema debe comprobar
  - RCarga-1.3.1. Si las partidas de dicho usuario están almacenadas en la base de datos
  - RCarga-1.3.2. Si las partidas de dicho usuario están al día
- RCarga-1.4. Si lo están, el sistema las cargará de la base de datos
- RCarga-1.5. Si no lo están entonces
  - RCarga-1.5.1. El sistema descargará las partidas
  - RCarga-1.5.2. El sistema almacenará las partidas descargadas

5.2.1.1.2 Mostrar partidas

RParti-1. El sistema debe permitir ver las partidas importadas al usuario
- RParti-1.1. El usuario podrá filtrarlas por el nombre del rival
- RParti-1.2. El sistema mostrará información de cada partida
  - RParti-1.2.1. Un tablero que muestra la situación final de la partida
  - RParti-1.2.2. El reloj de la partida
  - RParti-1.2.3. El tipo de rendimiento
  - RParti-1.2.4. Si la partida es amistosa o por puntos
  - RParti-1.2.5. La fecha en que se jugó
  - RParti-1.2.6. Los jugadores enfrentados
  - RParti-1.2.7. El estado de la partida
  - RParti-1.2.8. Si hay un ganador, el sistema lo indicará
  - RParti-1.2.9. Un resumen de los movimientos realizados en la partida

5.2.1.1.3 Analizar partidas

RAnali-1. El sistema debe permitir analizar partidas importadas al usuario
RAnali-1.1. El usuario podrá acceder a dichos análisis, en los que se mostrarán
RAnali-1.1.1. La ventaja de los participantes a lo largo de la partida
RAnali-1.1.2. La precisión de los movimientos jugados por ambos
RAnali-1.1.3. Las imperfecciones, errores y errores graves cometidos por ambos

5.2.1.1.4 Mostrar estadísticas

REstad-1. El sistema debe permitir ver estadísticas de las partidas importadas al usuario
REstad-1.1. El sistema mostrará dos grupos de estadísticas
REstad-1.1.1. Basadas en las victorias
REstad-1.1.2. Basadas en las imperfecciones, errores y errores graves

5.2.1.2 Requisitos no funcionales

RNF-1. La tasa de errores cometidos por el usuario deberá ser menor al 1% de las transacciones totales ejecutadas en el sistema
RNF-2. El sistema debe ser capaz de operar adecuadamente con hasta 10.000 partidas importadas inclusive
RNF-3. La aplicación se ejecutará correctamente en equipos compatibles con Windows, Linux o macOS
RNF-4. El tiempo para cargar 10.000 partidas descargadas o menos no podrá ser mayor a 1 minuto
RNF-5. El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 30 minutos
RNF-6. Las peticiones realizadas por la aplicación utilizarán el protocolo HTTPS

5.2.2 Identificación de Actores del Sistema

Se han reconocido dos actores del sistema: aquellos usuarios que tienen partidas cargadas y aquellos que no.

- El usuario que aún no ha importado partidas ya sea descargándolas de Lichess o cargándolas desde la base de datos, podrá hacerlo.
- El usuario que ha importado partidas podrá navegar a través de ellas, analizarlas y acceder a los análisis y estadísticas relacionadas.

5.2.3 Especificación de Casos de Uso

En esta sección se expondrán los casos de uso del sistema.

5.2.3.1 Casos de uso usuario sin partidas importadas

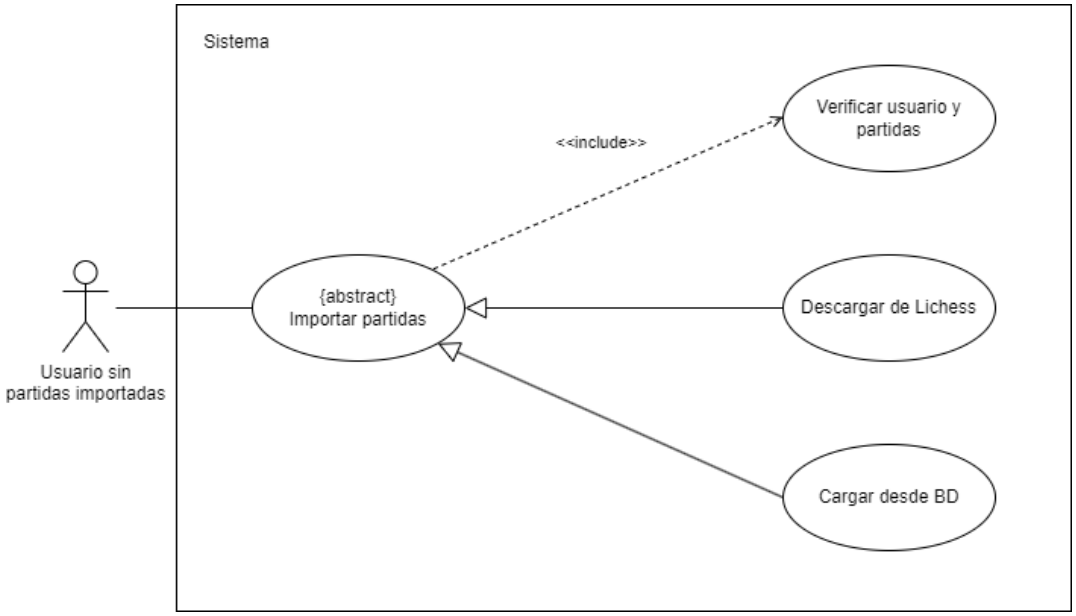


Figura 5.4 Casos de uso de usuario sin partidas importadas

Nombre del Caso de Uso
Verificar usuario y partidas
Descripción
El programa comprobará que el usuario introducido exista, haya participado en partidas y que estas sean compatibles con la aplicación.

Nombre del Caso de Uso
Importar partidas
Descripción
Una vez verificado el usuario y sus partidas, si estas no están en el sistema, se descargarán desde Lichess; de lo contrario, se cargarán desde la base de datos.

Nombre del Caso de Uso
Descargar de Lichess
Descripción
El programa descargará las partidas del usuario introducido desde Lichess.com.

Nombre del Caso de Uso
Cargar desde la base de datos
Descripción
El programa cargará las partidas del usuario introducido desde la base de datos.

5.2.3.2 Casos de uso usuario con partidas importadas

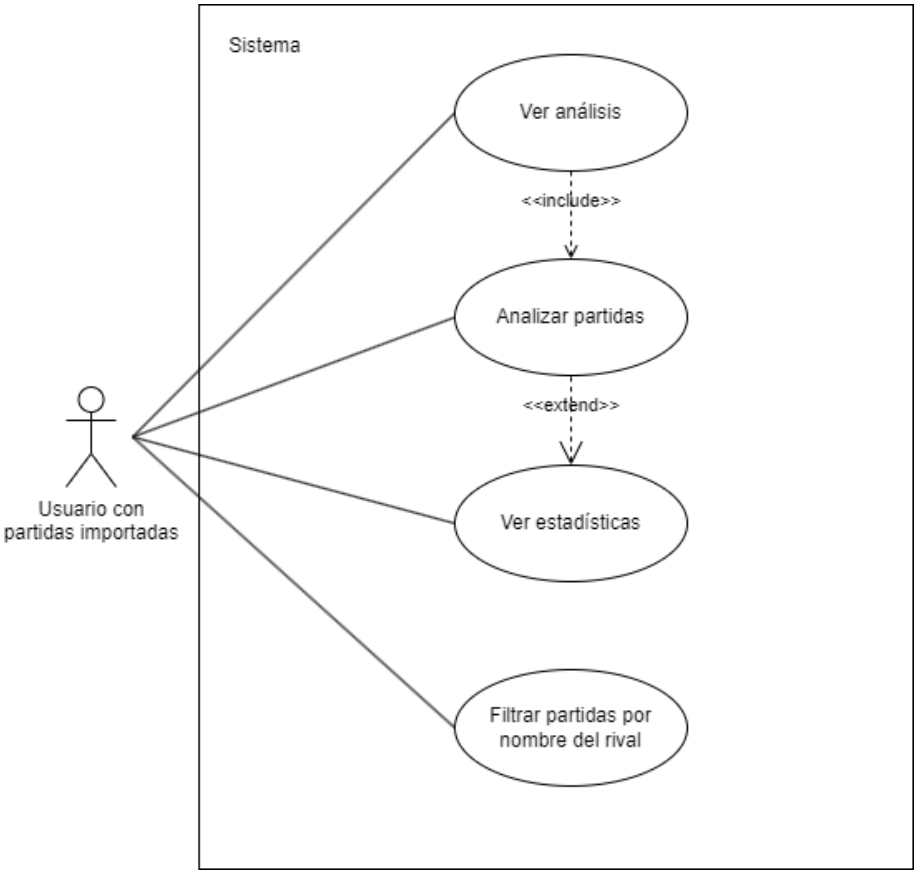


Figura 5.5 Caso de uso de usuario con partidas importadas

Nombre del Caso de Uso
Ver análisis
Descripción
El usuario podrá acceder al análisis de las partidas que lo tengan disponible.

Nombre del Caso de Uso
Analizar partidas
Descripción
El usuario podrá analizar bajo demanda aquellas partidas que aún no hayan sido analizadas.

Nombre del Caso de Uso
Ver estadísticas
Descripción
El usuario podrá acceder a estadísticas de las partidas y/o de sus análisis.

Nombre del Caso de Uso
Filtrar partidas por nombre del rival
Descripción
El usuario podrá filtrar partidas según el nombre del rival.

5.3 Identificación de los Subsistemas en la Fase de Análisis

En este apartado se identificarán los distintos subsistemas que componen el sistema.

5.3.1 Descripción de los Subsistemas

La aplicación estará dividida en 4 grandes subsistemas, disponibles en el cliente y descritos a continuación:

- **Motor de análisis:** Incluye los archivos necesarios para cargar el motor de análisis y las funciones para interactuar con él y procesar sus resultados.
- **Comunicación entre procesos:** Esta sección de Electron facilita la comunicación entre el proceso de renderizado (cliente) y el proceso principal, permitiendo el uso de módulos de Node.js.
- **Capa de persistencia:** Incluye las interfaces necesarias para manipular las partidas y sus análisis, así como las funciones requeridas para descargarlas y almacenarlas en la base de datos SQLite.
- **Cliente:** Incluye los archivos que forman la interfaz de la aplicación, es decir, tanto los ficheros TSX que muestran la información obtenida de la base de datos, como los ficheros auxiliares (CSS, HTML) que afectan al aspecto visual.

5.3.2 Descripción de los Interfaces entre Subsistemas

Todos los subsistemas están disponibles localmente, pero es importante detallar la comunicación entre procesos (IPC) en Electron. En este framework, los procesos intercambian mensajes a través de canales definidos por los desarrolladores utilizando los módulos *ipcMain* e *ipcRenderer*. Estos canales son personalizables y permiten la comunicación bidireccional.

Para importar módulos de Node.js y Electron en un proceso de renderizado aislado del contexto, se expone una API desde el script de precarga usando el módulo *contextbridge*.

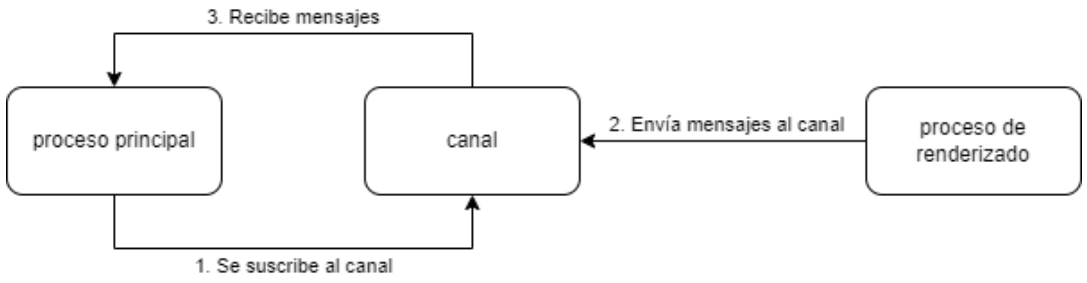


Figura 5.6 Proceso de suscripción de IPC

El aislamiento de contexto garantiza que tanto los scripts de precarga como la lógica interna de Electron se ejecuten en un contexto separado del sitio web cargado. Esto es crucial por motivos de seguridad, ya que ayuda a prevenir que el sitio web acceda a los componentes internos de Electron o a las APIs a las que el script de precarga tiene acceso (*Aislamiento de contexto | Electron*, s. f.).

En esta aplicación, los subsistemas se comunican directamente, aunque algunos utilizan IPC: Por ejemplo, el cliente (el proceso de renderizado) usa IPC para acceder a las operaciones proporcionadas por la capa de persistencia a través del proceso principal. De igual manera, el motor (equivalente al proceso de renderizado en el diagrama) utiliza IPC para interactuar con los módulos de Node.js, también a través del proceso principal.

5.4 Diagrama de Clases Preliminar del Análisis

En este apartado se identificarán las posibles clases del sistema basándose en los casos de uso y los subsistemas previamente revisados. Estas clases son una versión preliminar y simplificada que podría modificarse en etapas posteriores.

5.4.1 Diagrama de Clases

A continuación, se presenta un diagrama de clases global que ilustra las relaciones entre ellas:

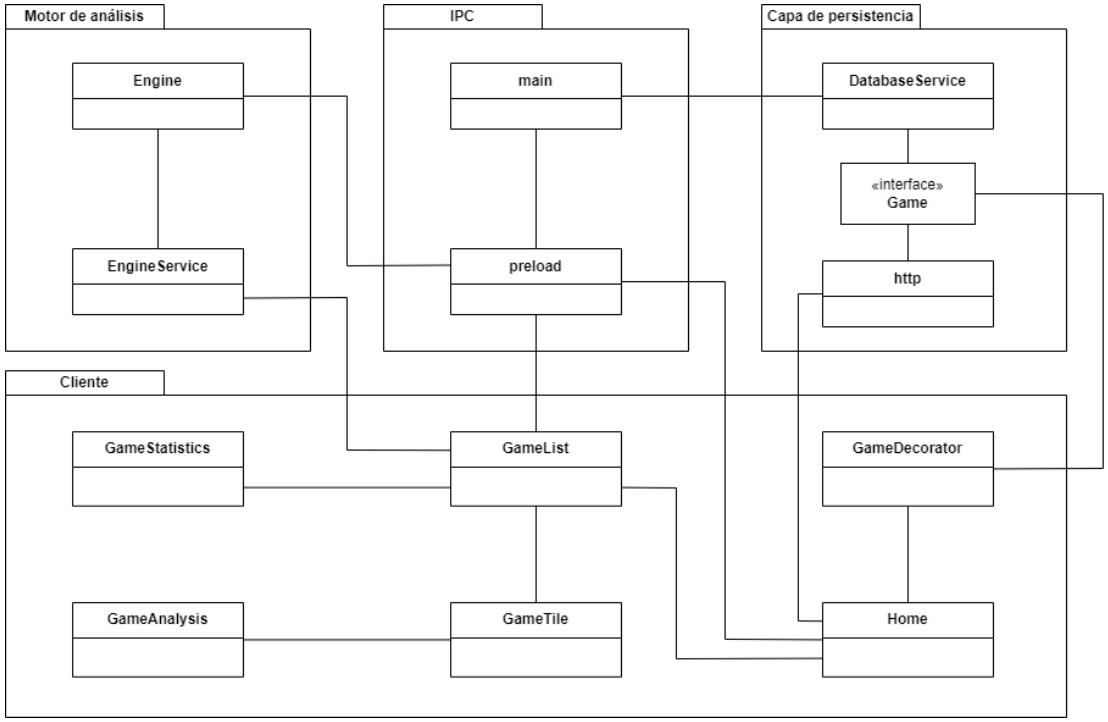


Ilustración 1 Diagrama de clases global

Los módulos y componentes en React se representan mediante clases, aunque no se definan como tales. En el caso de los componentes, las props se traducen a atributos de la clase.

En el subsistema "Motor de análisis", se identificaron dos clases importantes: "Engine", que manejará la lógica de comunicación con Stockfish, y "EngineService", encargada de procesar la salida del motor de análisis para convertirla en información útil.

Dentro del subsistema "IPC", se implementará una clase llamada "preload" que expondrá una API con los canales, y otra clase denominada "main" que se suscribirá a estos canales y proporcionará respuestas según sea necesario.

En la “Capa de persistencia”, se encuentra la clase "DataService", que proporciona todas las operaciones necesarias para consultar y manipular la base de datos. Además, está la interfaz "Game", diseñada para facilitar la gestión de partidas importadas en formato ND-JSON, y la clase "http", que contiene funciones para consultar la API de Lichess y descargar partidas.

Finalmente, en el subsistema "Cliente", cada ventana de la aplicación (excepto "GameTile", que es un componente de "GameList") tiene su propio componente. Estos componentes se comunican entre sí y con los subsistemas mencionados anteriormente para obtener la información a mostrar. Además, se ha implementado la clase "GameDecorator", que proporciona métodos para tratar la información de las partidas.

5.4.2 Descripción de las Clases

En esta sección se describirán las clases definidas anteriormente. Se comentarán sus responsabilidades, junto con los atributos y funciones propuestas.

5.4.2.1 Motor de análisis

Nombre de la Clase
Engine
Descripción
Este módulo facilita las operaciones necesarias para conectar con el motor de análisis, enviar y recibir mensajes, y calcular la ventaja en centipawns a lo largo de la partida, información utilizada por varias métricas.
Responsabilidades
Proporcionar una interfaz para interactuar con Stockfish de manera programática.
Atributos Propuestos
engine: mantiene una referencia al motor de análisis.
Funciones Propuestas
connectEngine: inicializa un Web Worker con el código de Stockfish. **disconnectEngine:** finaliza la conexión con el motor de análisis. **send:** envía un mensaje a Stockfish y devuelve la respuesta. **getGameCentipawns:** calcula la ventaja en centipawns a lo largo de la partida.

Nombre de la Clase
EngineService
Descripción
Este módulo facilita las operaciones utilizadas para calcular los valores mostrados en el análisis de partidas.
Responsabilidades
Proporcionar todas las operaciones necesarias para el análisis de manera centralizada.
Funciones Propuestas
getGameAdvantage: transforma la ventaja en centipawns a valores más comprensibles para el usuario. **calculateWinPercentage:** Estima la probabilidad de victoria basada en la ventaja en centipawns. **calculateAccuracy:** evalúa la precisión de un movimiento según la probabilidad de victoria antes y después de realizarlo. **getGameAccuracy:** obtiene la precisión media de los movimientos realizados por los jugadores en una partida. **calculateErrors:** calcula el número de errores cometidos por ambos jugadores, basándose en dos umbrales.

5.4.2.2 Comunicación entre procesos

Nombre de la Clase
main
Descripción
Este script controla el proceso principal, que se ejecuta en un entorno Node.js completo.
Responsabilidades
Controlar el ciclo de vida de la aplicación, mostrar interfaces nativas, realizar operaciones privilegiadas y gestionar los procesos de renderizado.

Nombre de la Clase
preload
Descripción
Este script contiene el código que se ejecuta antes de que una página web se cargue en la ventana del navegador. Tiene acceso tanto a las API de DOM como al entorno Node.js, y utiliza para exponer APIs privilegiadas al renderizador a través de la API contextBridge.
Responsabilidades
Configurar interfaces de comunicación entre procesos (IPC) para pasar mensajes arbitrarios entre los procesos principal y de renderizado.

5.4.2.3 Capa de persistencia

Nombre de la Clase
DatabaseService
Descripción
Este módulo proporciona operaciones para consultar y operar con la base de datos SQLite.
Responsabilidades
Proporcionar todas las operaciones necesarias para almacenar las partidas y sus análisis.
Funciones Propuestas
connectDatabase: devuelve una conexión con la base de datos SQLite. **insertGames:** inserta las partidas recibidas en la base de datos en una sola transacción para evitar caídas de la aplicación. **getPlayerGames:** devuelve las partidas del jugador cuya ID se pasa como parámetro. **getPlayerGamesCount:** devuelve el número de partidas del jugador cuya ID se pasa como parámetro. **updateGameAnalysis:** actualiza el análisis de la partida cuya ID se pasa como parámetro.

Nombre de la Clase
http
Descripción
Este módulo facilita operaciones para consultar y descargar partidas desde la API de Lichess.
Responsabilidades
Proporcionar una interfaz para interactuar con la API de Lichess.
Funciones Propuestas
checkPlayer: comprueba la existencia del usuario especificado y si ha jugado partidas. **readStream:** ayuda a leer respuestas transmitidas en formato NDJSON. **handleGameStream:** devuelve las partidas jugadas por el jugador especificado y el progreso en la descarga de partidas.

Nombre de la Clase
Game
Descripción
Esta interfaz define los atributos de una partida y sus tipos.
Responsabilidades
Facilitar el manejo de partidas descargadas desde la API de Lichess.
Atributos propuestos

id: identificador único de la partida.
clock (opcional): objeto que describe el reloj utilizado en la partida, con los siguientes atributos:

- **initial:** tiempo inicial en segundos.
- **increment:** incremento de tiempo por jugada en segundos.
- **totalTime:** tiempo total de la partida en segundos.

perf: tipo de rendimiento de la partida.
speed: velocidad o ritmo de la partida (por ejemplo, blitz, rápidas, clásicas).
moves: cadena de texto que representa los movimientos realizados en la partida.
source: origen o fuente de la partida.
rated: indica si la partida fue clasificada o no.
status: estado actual de la partida (por ejemplo, en progreso, terminado).
winner: color del jugador que ganó la partida.
variant: variante de la partida (por ejemplo, estándar, chess960).
createdAt: marca de tiempo Unix que indica cuándo se creó la partida**.**
lastMoveAt: marca de tiempo Unix que indica cuándo se realizó el último movimiento en la partida.
players: objeto que contiene información sobre los jugadores involucrados, con las siguientes propiedades:

- **black:** objeto que describe al jugador que juega con las piezas negras.
- **white:** objeto que describe al jugador que juega con las piezas blancas.
  Cada objeto jugador tiene las siguientes propiedades:
  - **rating (opcional):** clasificación del jugador.
  - **aiLevel (opcional):** nivel de la inteligencia artificial del jugador.
  - **ratingDiff (opcional):** diferencia de clasificación para el jugador.
  - **user:** objeto que contiene la identificación (id) y el nombre (name) del usuario (si el jugador no es anónimo o una IA).

analysis (opcional): objeto que puede contener el análisis relacionado con la partida, representado como un diccionario con claves de tipo cadena y valores de cualquier tipo.

5.4.2.4 Cliente

Nombre de la Clase
GameDecorator
Descripción
Esta clase encapsula objetos de tipo Game y extiende sus funcionalidades con características adicionales.
Responsabilidades
Centralizar todos los métodos relacionados con el manejo de la información de las partidas.
Atributos Propuestos
game: objeto de tipo Game.
Funciones Propuestas
Constructor: inicializa el objeto con una partida proporcionada. **parsePosition:** devuelve la posición actual del tablero en formato FEN. **parseGameClock:** formatea el reloj de la partida si está disponible. **parsePlayerName:** devuelve el nombre del jugador según el lado proporcionado (blanco o negro). Si el jugador es una IA, se devuelve su nivel; si no está registrado, devuelve "Anónimo". **parseGameStatus:** devuelve el estado actual de la partida, indicando quién ha perdido en caso de abandono o tiempo agotado. **parseGameWinner:** devuelve un mensaje indicando quién ganó la partida, considerando si hubo un empate. **parseGameMoves:** construye y devuelve una representación abreviada de las primeras jugadas de la partida, seguida por el número total de movimientos si son más de tres. **getGame:** retorna la partida encapsulada en GameDecorator. **getGameMoves:** retorna un array con los movimientos de la partida. **getSide:** determina el lado del jugador según el ID proporcionado. **getOpponentSide:** determina el lado del oponente del jugador según el ID proporcionado. **getStatusColor:** devuelve una clase CSS basada en si el jugador con el ID proporcionado ganó o perdió. **setAnalysis:** establece el análisis de la partida con los datos proporcionados.

Nombre de la Clase
Home
Descripción
Este componente corresponde a la ventana de inicio de la aplicación.
Atributos propuestos
username: nombre de usuario utilizado para buscar y gestionar partidas. **onGamesUpdate:** función callback que se llama cuando se actualizan las partidas del usuario. **onUsernameUpdate:** función callback que se llama cuando se actualiza el nombre de usuario.
Funciones Propuestas
showHomeError: maneja la visualización de errores en la página principal. **showUserGames:** filtra y muestra las partidas válidas del usuario. Actualiza el estado y utiliza funciones de navegación para redirigir a la página de partidas. **retrieveNewGames:** maneja la recuperación de nuevas partidas del usuario, utilizando un stream de juegos y actualizando el estado de progreso. **retrieveOldGames:** recupera las partidas del usuario almacenadas localmente. **checkForNewGames:** verifica si hay nuevas partidas disponibles comparando el recuento de juegos local con el servidor. Llama a retrieveNewGames si hay diferencias. **verifyPlayer:** verifica la existencia del usuario y el recuento de juegos. Inicia el proceso de importación de partidas nuevas o muestra errores correspondientes. **handleSubmit:** llamada al enviar el formulario de importación de partidas. Verifica si se ha ingresado un nombre de usuario válido y luego llama a verifyPlayer.

Nombre de la Clase
GameList
Descripción
Este componente corresponde a la ventana con la lista de partidas.
Atributos propuestos
username: nombre de usuario utilizado para buscar partidas y realizar análisis. **games:** lista de objetos GameDecorator. **onUsernameUpdate:** función para actualizar el nombre de usuario en el componente padre cuando sea necesario.
Funciones Propuestas
handleToggleAll: alterna la selección de todas las partidas mostradas en la lista. **handleToggle:** permite seleccionar o deseleccionar partidas individuales según el índice. **handleSearchTermChange:** actualiza el término de búsqueda y reinicia la página actual. **handlePageChange:** cambia la página actual de partidas en la lista paginada. **analyseGame:** analiza una partida específica para obtener precisión y ventaja usando servicios asíncronos. **updateEstimatedTime:** calcula y actualiza el tiempo estimado restante para completar el análisis de partidas. **analyseGames:** realiza el análisis de varias partidas de forma secuencial, actualizando el estado de progreso y tiempo estimado. **handleSubmit:** maneja la solicitud de análisis para las partidas seleccionadas, activando el estado de carga y luego iniciando analyseGames. **handleBack:** navega de regreso a la página principal y restablece el nombre de usuario. **handleStatistics:** navega a la página de estadísticas.

Nombre de la Clase
GameTile
Descripción
Este componente corresponde a una de las partidas de la lista.
Atributos propuestos
index: índice de la partida en la lista. **username:** nombre de usuario actual. **checked:** array de índices de juegos seleccionados. **game:** objeto GameDecorator que encapsula la información y métodos relacionados con la partida. **handleToggle:** función para manejar la selección/deselección de partidas.
Funciones Propuestas
handleToggleAll: alterna la selección de todos los juegos mostrados en la lista. **handleToggle:** permite seleccionar o deseleccionar juegos individuales según el índice. **handleSearchTermChange:** actualiza el término de búsqueda y reinicia la página actual. **handlePageChange:** cambia la página actual de juegos en la lista paginada. **analyseGame:** analiza un juego específico para obtener precisión y ventaja usando servicios asíncronos. **updateEstimatedTime:** calcula y actualiza el tiempo estimado restante para completar el análisis de juegos. **analyseGames:** realiza el análisis de varios juegos de forma secuencial, actualizando el estado de progreso y tiempo estimado. **handleSubmit:** maneja la solicitud de análisis para los juegos seleccionados, activando el estado de carga y luego iniciando analyseGames. **handleBack:** navega de regreso a la página principal y restablece el nombre de usuario. **handleStatistics:** navega a la página de estadísticas.

Nombre de la Clase
GameAnalysis
Descripción
Este componente corresponde a la ventana de visualización de análisis.
Atributos propuestos
Extrae el estado de la ubicación para obtener la partida (game) y el nombre de usuario (username) pasados como propiedades de ubicación.
Funciones Propuestas
replayMoves: revisa y actualiza la posición del tablero de ajedrez según el índice especificado. **handleBottom:** permite navegar por los movimientos de la partida hacia el inicio. **handleBack:** permite navegar por los movimientos de la partida hacia atrás. **handleNext:** permite navegar por los movimientos de la partida hacia adelante. **handleTop:** permite navegar por los movimientos de la partida hacia el final. **handleNavigate:** navega de vuelta a la lista de partidas.

Nombre de la Clase
GameStatistics
Descripción
Este componente corresponde a la ventana de visualización de estadísticas.
Atributos propuestos
username: nombre de usuario para el cual se muestran las estadísticas. **games:** array de objetos GameDecorator, con toda la información necesaria para calcular y mostrar las estadísticas.
Funciones Propuestas
updateMistakes: actualiza las estadísticas de errores (blunders, mistakes, inaccuracies) y errores relacionados con la clasificación del jugador. **updateWins:** actualiza las estadísticas de victorias, incluyendo victorias contra oponentes con mayor o menor clasificación Elo. **updateResults:** actualiza las estadísticas generales basadas en los resultados de las partidas (empates, victorias y derrotas). **updateStats:** itera a través de las partidas para actualizar todas las estadísticas utilizando las funciones mencionadas anteriormente.

5.5 Análisis de Casos de Uso y Escenarios

En este apartado se describirán los casos de uso identificados anteriormente de forma detallada, a través de sus escenarios.

5.5.1 Cargar desde la base de datos

Cargar desde la base de datos	
Precondiciones	La base de datos contiene partidas para el usuario especificado.
Poscondiciones	Las partidas importadas se mostrarán al usuario.
Actores	Iniciado por un usuario sin partidas importadas y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la pantalla de inicio. 2. El usuario ingresa un nombre de usuario. 3. El sistema valida la información ingresada. 4. El sistema obtiene las partidas del usuario desde la BD. 5. El sistema presenta las partidas descargadas al usuario.

Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** Falta el nombre de usuario
	  ○ Notificar al usuario sobre el error.
	  ○ Volver al paso 2 del escenario principal.

5.5.2 Descargar de Lichess

Descargar de Lichess	
Precondiciones	La base de datos no contiene partidas para el usuario especificado.
Poscondiciones	Las partidas importadas se mostrarán al usuario.
Actores	Iniciado por un usuario sin partidas importadas y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la pantalla de inicio. 2. El usuario ingresa un nombre de usuario. 3. El sistema valida la información ingresada. 4. El sistema obtiene las partidas del usuario desde Lichess. 5. Las partidas descargadas se almacenan y se presentan al usuario.
Variaciones (escenarios secundarios)	• **Escenario Alternativo 1:** Falta el nombre de usuario o Notificar al usuario sobre el error. o Volver al paso 2 del escenario principal. • **Escenario Alternativo 2:** Usuario no existente o Notificar al usuario sobre el error. o Volver al paso 2 del escenario principal. • **Escenario Alternativo 3:** Usuario sin partidas o Notificar al usuario que no hay partidas disponibles. o Volver al paso 2 del escenario principal. • **Escenario Alternativo 4:** Usuario sin partidas compatibles o Notificar al usuario que no hay partidas compatibles disponibles. o Volver al paso 2 del escenario principal.
Excepciones	• **La API de Lichess no está disponible:** No se puede verificar el usuario ingresado ni descargar sus partidas. o Notificar un error relacionado con el problema encontrado.

5.5.3 Ver análisis

Ver análisis	
Precondiciones	El usuario debe haber importado partidas y haber seleccionado una para analizar.
Poscondiciones	El análisis de la partida seleccionada se mostrará al usuario.
Actores	Iniciado y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la lista de partidas disponibles. 2. El usuario selecciona una partida previamente analizada. 3. El sistema muestra el análisis correspondiente a la partida seleccionada.

5.5.4 Analizar partidas

Analizar partidas	
Precondiciones	El usuario debe haber importado partidas y no haber seleccionado todas para ser analizadas.
Poscondiciones	La partida seleccionada será analizada y mostrada como tal.
Actores	Iniciado y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la lista de partidas disponibles. 2. El usuario selecciona una partida específica para realizar el análisis. 3. El sistema lleva a cabo el análisis de la partida seleccionada y guarda los resultados. 4. El sistema muestra la partida como analizada, actualizando su estado.

5.5.5 Ver estadísticas

Ver estadísticas	
Precondiciones	El usuario debe haber importado partidas.
Poscondiciones	Las estadísticas de las partidas importadas se mostrarán al usuario.
Actores	Iniciado y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la lista de partidas importadas. 2. El usuario accede a la sección de estadísticas. 3. El sistema presenta las estadísticas correspondientes a las partidas importadas.

Excepciones	• **No ha jugado suficientes partidas:** Algunas estadísticas no estarán disponibles.
	  ○ Notificar al usuario sobre esta limitación.
	• **No ha analizado suficientes partidas:** Algunas estadísticas no estarán disponibles.
	  ○ Notificar al usuario sobre esta limitación.

5.5.6 Filtrar partidas por nombre del rival

Filtrar partidas por nombre del rival	
Precondiciones	El usuario debe haber importado partidas.
Poscondiciones	Se mostrarán al usuario las partidas filtradas por el nombre del rival.
Actores	Iniciado y finalizado por un usuario con partidas importadas.
Descripción	1. El sistema muestra la lista de partidas disponibles. 2. El usuario ingresa el nombre del rival para filtrar las partidas. 3. El sistema aplica el filtro y muestra las partidas que coinciden con el nombre del rival introducido.

5.6 Relación Escenarios – Requisitos

Este apartado trata de mostrar de una manera resumida y visual, a través del uso de una tabla, la relación entre los requisitos y los escenarios explicados anteriormente.

Requisitos	5.5.1	5.5.2	5.5.3	5.5.4	5.5.5	5.5.6
RCarga-1	X	X				
RCarga-1.1	X	X				
RCarga-1.1.1	X	X				
RCarga-1.1.2		X				
RCarga-1.2		X				
RCarga-1.3	X					
RCarga-1.3.1	X					
RCarga-1.3.2	X					
RCarga-1.4	X					
RCarga-1.5		X				
RCarga-1.5.1		X				
RCarga-1.5.2		X				
RParti-1			X	X	X	X
RParti-1.1						X
RAnali-1				X		
RAnali-1.1			X	X		
RAnali-1.1.1			X			
RAnali-1.1.2			X			
RAnali-1.1.3			X			
REstad-1					X	
REstad-1.1					X	
REstad-1.1.1					X	
REstad-1.1.2					X	

Tabla 5.1 Relación Escenarios – requisitos

Como se puede observar, todos los requisitos están cubiertos en un escenario, existiendo una trazabilidad directa entre los grupos de requisitos y los casos de uso.

5.7 Análisis de Interfaces de Usuario

Este apartado abordará el diseño de la interfaz de usuario durante la fase de análisis, procurando que esta sea fácil de usar y permita operar el programa de manera eficiente.

5.7.1 Descripción de la Interfaz

En esta sección se presentará el diseño de cada una de las ventanas que formarán parte de este proyecto.

5.7.1.1 Ventana de inicio

Figura 5.7 Ventana de inicio (análisis)

La pantalla de inicio mostrará únicamente un campo de texto para introducir el nombre de usuario cuyas partidas se quieren importar, junto con un botón para enviar el formulario. Cualquier error indicado en secciones anteriores se mostrará justo debajo del campo de texto.

5.7.1.2 Ventana con lista de partidas

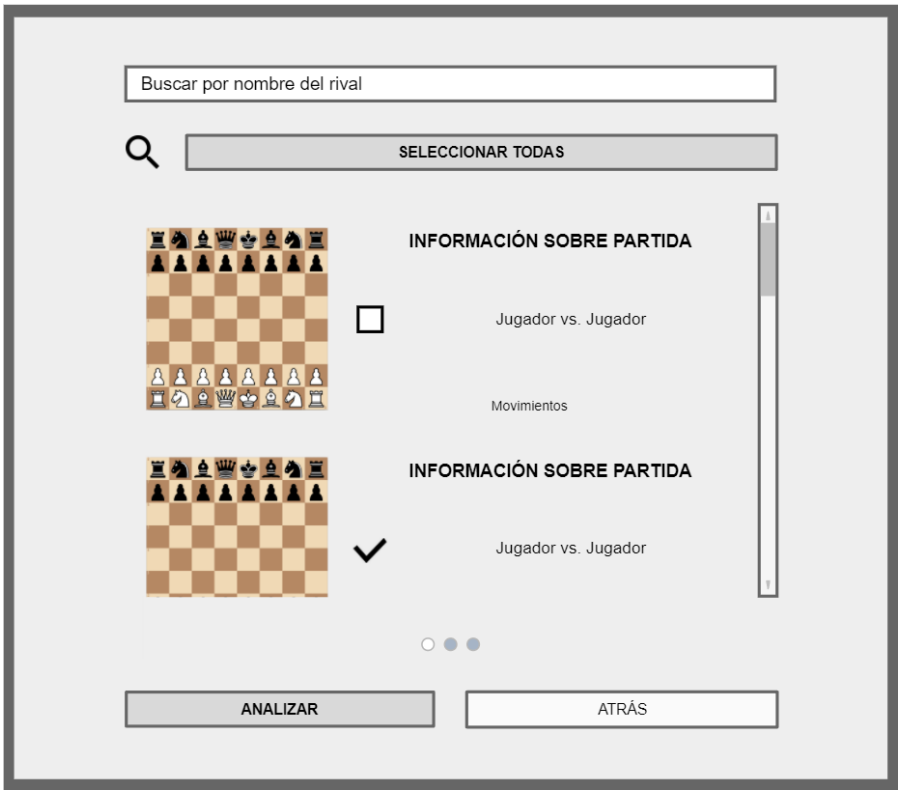


Figura 5.8 Ventana con lista de partidas (análisis)

En la parte superior, hay un campo de texto para introducir el nombre del rival y filtrar las partidas. A continuación, hay un botón con forma de lupa para acceder a las estadísticas, seguido de un botón para seleccionar todas las partidas importadas.

Las partidas se pueden seleccionar en la vista de desplazamiento mediante una casilla de verificación situada entre un tablero que muestra la situación final de la partida y la información identificativa de esta. Las partidas ya analizadas mostrarán un tick y se podrá acceder a su análisis haciendo clic en ellas.

Finalmente, hay un elemento de paginación y dos botones adicionales: uno para solicitar el análisis de las partidas seleccionadas y otro para volver a la ventana de inicio.

5.7.1.3 Ventana de análisis

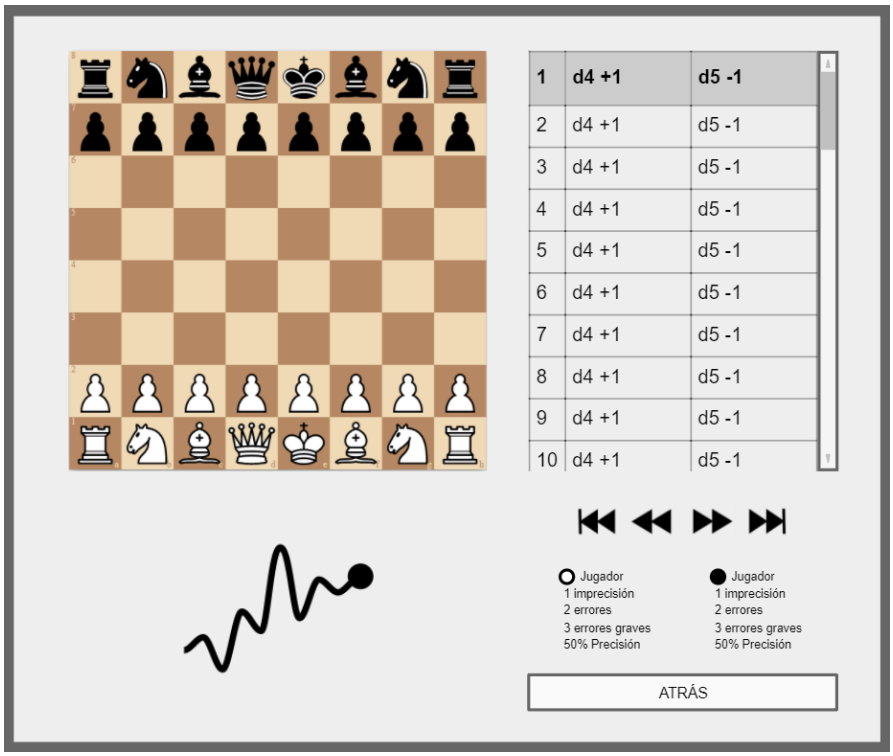


Figura 5.9 Ventana de análisis (análisis)

En esta ventana se distinguen cuatro elementos principales:

En la primera fila, hay un tablero que muestra la posición de las piezas en un momento concreto de la partida. A su derecha, se encuentra una tabla con los movimientos y la ventaja en ese momento. En la parte inferior, hay cuatro botones que permiten avanzar y retroceder los movimientos.

En la segunda fila, hay un gráfico que muestra de forma visual la ventaja a lo largo de la partida. A su derecha, se presenta información sobre los errores cometidos y la precisión media de los movimientos de cada jugador.

Por último, en la esquina inferior derecha, hay un botón para volver a la lista de partidas.

5.7.1.4 Ventana de estadísticas

Figura 5.10 Ventana de estadísticas

En esta ventana se presentarán las estadísticas, organizadas en secciones según su temática. Por ejemplo, las estadísticas de victorias estarán en su sección correspondiente. Para cambiar entre secciones, se podrán utilizar botones situados a la izquierda y derecha del componente.

Este mostrará diferentes gráficos, como gráficos de pastel y gráficos de barras, para cada una de las estadísticas, todos ellos titulados para facilitar su comprensión.

Por último, en la parte inferior derecha hay un botón para volver a la lista de partidas.

5.7.2 Diagrama de Navegabilidad

En el siguiente diagrama, se muestran todas las posibles transiciones entre las ventanas descritas en la sección anterior:

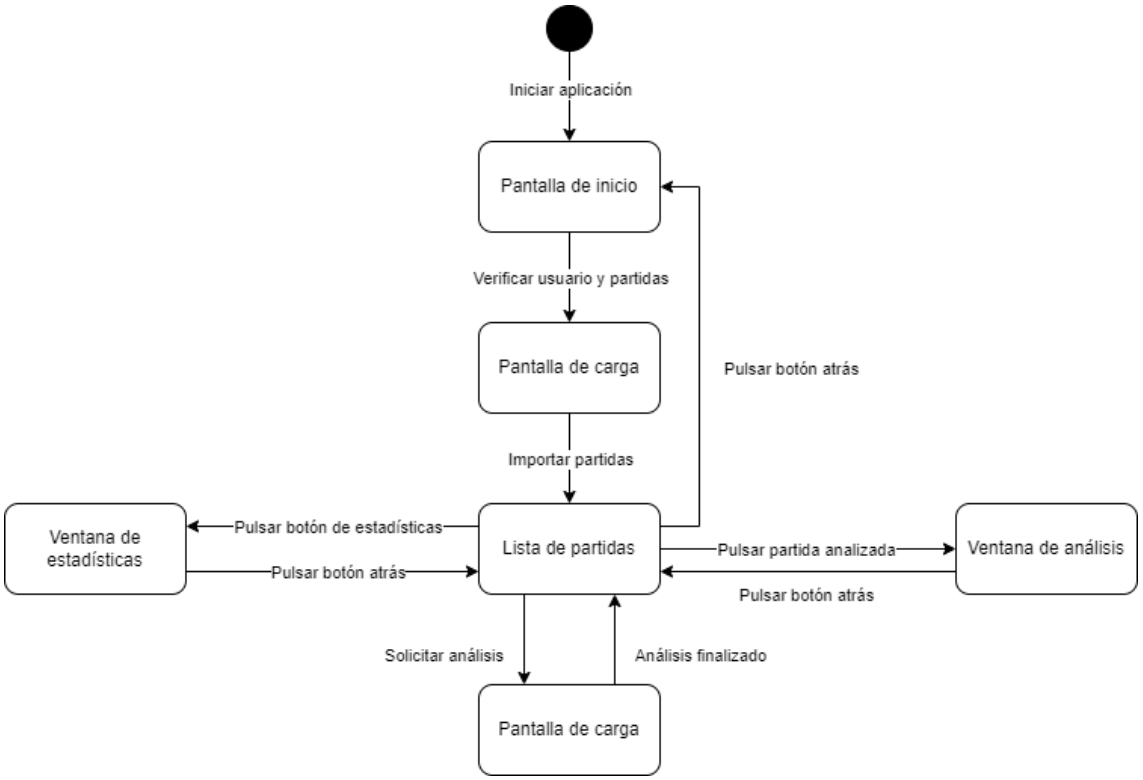


Figura 5.11 Diagrama de navegabilidad

5.8 Especificación del Plan de Pruebas

En este apartado se elaborará el plan de pruebas de la aplicación, así como todos los mecanismos que serán utilizados para detectar errores y corregirlos durante la fase de implementación.

5.8.1 Pruebas unitarias

Las pruebas unitarias verificarán el correcto funcionamiento individual de cada módulo de código. Se utilizará Jest junto con la biblioteca de testing de React para evaluar el comportamiento de los componentes correspondientes a las ventanas mencionadas en la sección 5.7.1. Además, se evaluarán las operaciones del módulo EngineService que actúan sobre los resultados devueltos por el motor, simulando el resto de componentes, funciones y/o los canales de Electron.

No se incluirán pruebas para la base de datos y el motor de análisis debido a las dificultades para acceder a estos en el entorno de pruebas y porque no son parte directa del desarrollo del proyecto. A partir de los casos de uso, escenarios y módulos descritos anteriormente, se desarrollarán las pruebas unitarias necesarias y se especificarán los resultados esperados para cada una de ellas una vez sean ejecutadas.

5.8.2 Pruebas de integración y del sistema

Estas pruebas verificarán que diferentes conjuntos de componentes funcionen correctamente cuando se ensamblan para cumplir una función específica. Cada escenario se probará manualmente con una amplia variedad de entradas para asegurar que el sistema responda adecuadamente ante errores de los usuarios.

No se realizarán de manera programática debido a que la mayoría de frameworks no son compatibles con Electron, y las soluciones disponibles para integrar el motor y los canales con los frameworks que sí lo son no se consideran óptimas.

5.8.3 Pruebas de usabilidad

Para las pruebas de usabilidad, se pedirá a personas con diversos perfiles (diferentes edades y niveles de experiencia en informática) que respondan a una serie de preguntas relacionadas con la ejecución de tareas básicas en el proyecto, como descargar partidas, analizarlas y acceder a diferentes datos. Se les proporcionará un contexto básico sobre el propósito de la aplicación para ayudarles a orientarse.

Además, se incluirá a un jugador avanzado de ajedrez para que evalúe la presentación del análisis y la utilidad de las estadísticas proporcionadas.

Capítulo 6. Diseño del Sistema

En este capítulo se describirán los diagramas y diseños utilizados durante la implementación del proyecto, los cuales están basados en los subsistemas definidos en la fase de análisis.

6.1 Arquitectura del Sistema

En este apartado se describirán los diagramas de paquetes, componentes y arquitectura.

6.1.1 Diagrama de Paquetes

Se agruparán las clases identificadas en paquetes según su funcionalidad. A continuación, se presenta un diagrama que ilustra las relaciones entre estos paquetes, utilizando colores más oscuros para representar niveles más profundos:

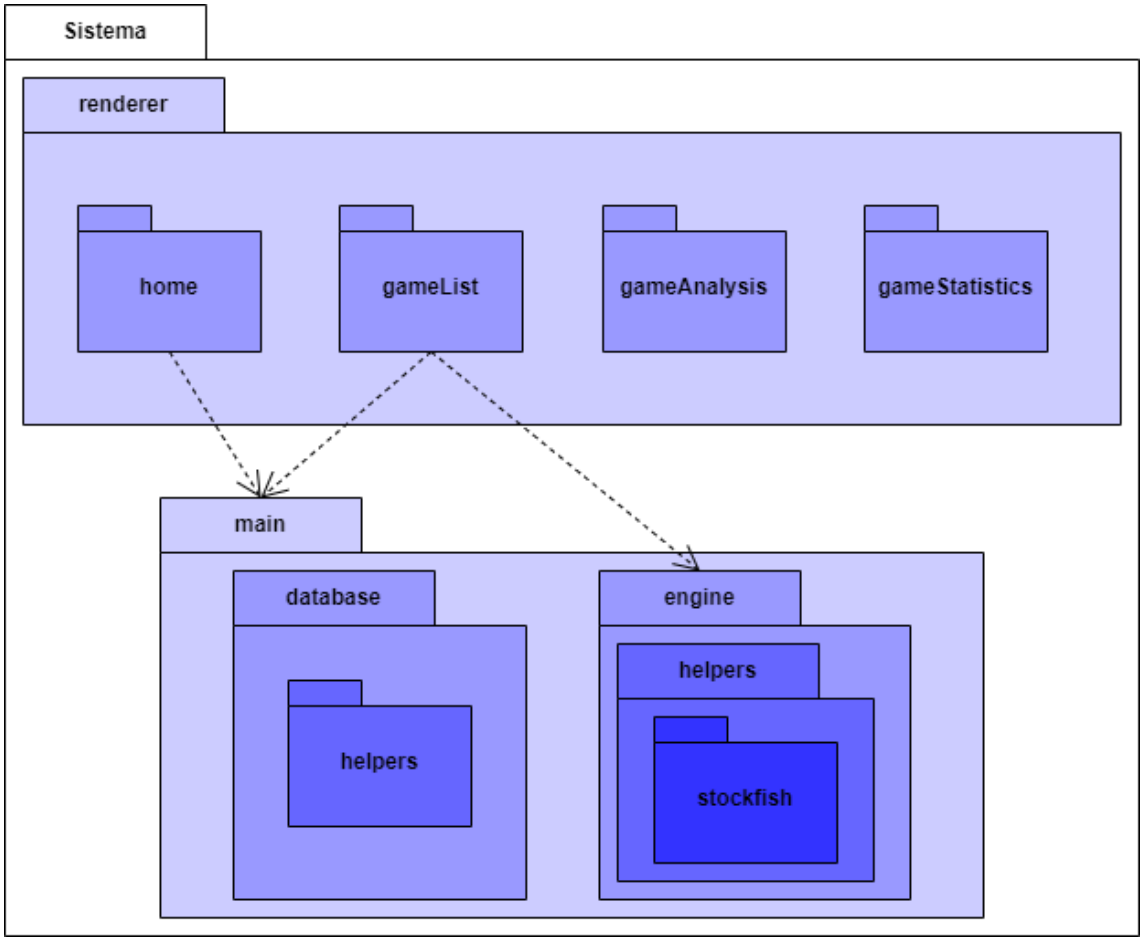


Figura 6.1 Diagrama de paquetes

El sistema se organiza en dos grandes paquetes: "renderer" para el proceso de renderizado y "main" para el proceso principal. Estos, a su vez, se dividen en varios subpaquetes:

6.1.1.1 Sistema

En este nivel, además de los dos paquetes principales, se definen los tipos e interfaces utilizados en diferentes partes de la aplicación, así como el módulo responsable de interactuar con la API de Lichess.

6.1.1.1.1 renderer

En este paquete se encuentra el componente raíz que envuelve al resto, organizados en varios paquetes correspondientes a cada una de las ventanas.

6.1.1.1.1.1 home

Este paquete contiene los archivos responsables de renderizar la pantalla de inicio. Se comunica con la base de datos a través de los canales definidos en el paquete main.

6.1.1.1.1.2 gameList

Este paquete contiene los archivos responsables de renderizar la lista de partidas. Se comunica con la base de datos a través de los canales definidos en el paquete main y con el motor de análisis mediante las funciones del paquete engine.

6.1.1.1.1.3 gameAnalysis

Este paquete contiene los archivos responsables de renderizar la vista de análisis.

6.1.1.1.1.4 gameStatistics

Este paquete contiene los archivos responsables de renderizar la vista de estadísticas.

6.1.1.1.2 main

Este paquete contiene las clases responsables del funcionamiento de Electron (main, preload, etc.), los paquetes relacionados con los servicios de la base de datos y el motor de análisis, además de funciones auxiliares para el funcionamiento de la aplicación y sus pruebas unitarias que no pertenezcan a los otros paquetes.

6.1.1.1.2.1 services

Este paquete agrupa los subpaquetes relacionados con la base de datos y el motor de análisis.

6.1.1.1.2.1.1 database

Este paquete incluye la clase que facilita la operación de la base de datos SQLite y expone sus funciones en la API de Electron. También contiene un subpaquete con funciones auxiliares utilizadas por la clase principal.

6.1.1.1.2.1.1.1 helpers

Este paquete recopila funciones para mapear elementos a insertar o a recuperar de la base de datos, así como otras utilidades, como las consultas preparadas y usadas por el servicio.

6.1.1.1.2.1.2 engine

Este paquete incluye la clase que procesa las salidas del motor de análisis Stockfish, junto con un subpaquete de elementos auxiliares utilizados por esta.

6.1.1.1.2.1.2.1 helpers

Este paquete incluye una clase que define la interfaz utilizada para interactuar con el motor de análisis, además de contener el código en JavaScript de Stockfish y otros archivos necesarios para cargarlo en un trabajador web.

6.1.2 Diagrama de Despliegue

En esta sección se presenta el diagrama de despliegue del sistema, el cual muestra cómo se relacionan los procesos de software en la máquina y con sistemas externos:

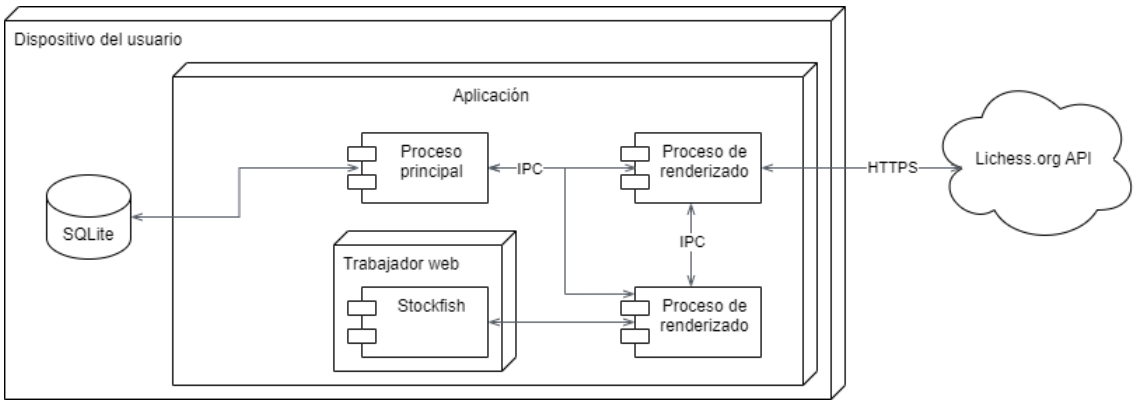


Figura 6.2 Diagrama de despliegue

El sistema se ejecuta en una sola máquina y se comunica con la API de Lichess utilizando el protocolo HTTPS. A continuación, se detallan cada uno de los componentes y sus relaciones con los demás.

6.1.2.1 Dispositivo del usuario

Al usar Electron, esta aplicación será compatible con sistemas Windows, Linux y macOS.

6.1.2.2 Aplicación

Como se ha mencionado repetidamente en capítulos anteriores, las aplicaciones desarrolladas en Electron constan de dos tipos de procesos (principal y de renderizado) que se comunican mediante canales. En este caso específico, los procesos de renderizado se comunican con la base de datos SQLite a través del proceso principal.

6.1.2.3 Trabajador web

Para utilizar Stockfish, se carga su versión JavaScript en un trabajador web, con el cual se comunicarán los procesos de renderizado mediante el envío asíncrono de mensajes. Este trabajador se enciende solo cuando es necesario y luego se apaga.

6.1.2.3.1 Interacción con el motor

A continuación, se detalla la interacción con este motor, comenzando desde un nivel superficial y profundizando:

El módulo "engine" interactúa con un objeto que expone varios métodos a través de una interfaz. Nos centraremos específicamente en dos métodos:

- **send:** recibe una cadena de texto y dos funciones de callback. La cadena de texto representa el mensaje enviado al motor, y las funciones de callback se ejecutan cuando el motor completa su procesamiento y devuelve una respuesta (ya sea completa o parcial).
- **quit:** apaga el trabajador y elimina la referencia para que el recolector de basura pueda limpiarlo.

En este objeto, cada método está definido para interactuar con el trabajador web mediante mensajes "postMessage" y para manejar las respuestas recibidas a través de "onMessage".

El trabajador web se instancia con el script Stockfish.js, que es una implementación en JavaScript y WebAssembly del motor de ajedrez Stockfish. Se crea un nuevo proceso hijo utilizando Node.js, configurado para la comunicación a través de “stdin”, “stdout” y “stderr”. Además, se definen métodos para manejar mensajes entrantes (“echo”), enviar mensajes (“postMessage”) y apagar el trabajador (“terminate”).

6.1.2.3.2 Comandos enviados

Durante la comunicación con el motor, se emplea el protocolo UCI, recomendado para aplicaciones con interfaz gráfica de usuario, mediante el envío de comandos de texto. A continuación, se detallan algunos de estos comandos y su uso específico en el proyecto:

- **go:** Inicia el cálculo en la posición actual establecida con el comando “position”. Hay una serie de parámetros que pueden seguir a este comando y todos serán enviados en la misma cadena. En este sistema, se enviará "go nodes x" para explorar x nodos.
- **setOption:** Se envía al motor cuando el usuario quiere cambiar los parámetros internos del mismo. En este sistema, se enviará "setoption name Use NNUE value true" para asegurar que se utilice NNUE en la evaluación.
- **position:** Establece la posición descrita en la cadena recibida como parámetro. En este sistema, se enviará "position fen x" para fijar la posición descrita en FEN de un punto específico de la partida.

6.1.2.4 Base de datos SQLite

Siguiendo las convenciones de Electron, la base de datos encargada de almacenar partidas y sus análisis se encuentra en el directorio "userData", que por defecto corresponde al directorio "appData" seguido del nombre de la aplicación.

6.1.2.5 API de Lichess

Los procesos de renderizado se comunicarán con la API de Lichess.org mediante el protocolo seguro HTTPS, sin necesidad de autorización, para realizar consultas sobre el estado de un usuario y/o descargar sus partidas transmitidas en formato ND-JSON.

6.2 Diagramas de Interacción

En este apartado se presentarán los diagramas de interacción, abarcando los casos de uso más complejos del proyecto y omitiendo los más simples y repetitivos. Estos diagramas se utilizan para documentar el diseño del sistema, destacando la interacción entre objetos o módulos y omitiendo las llamadas a métodos internos.

Para evitar confusiones, la comunicación entre procesos (IPC) se representará como si los módulos se comunicaran directamente. Para una explicación más detallada, consulta la sección 5.3.2.

6.2.1 Analizar partidas

En este diagrama se detalla el proceso que se desencadena cuando el usuario selecciona una partida y solicita su análisis.

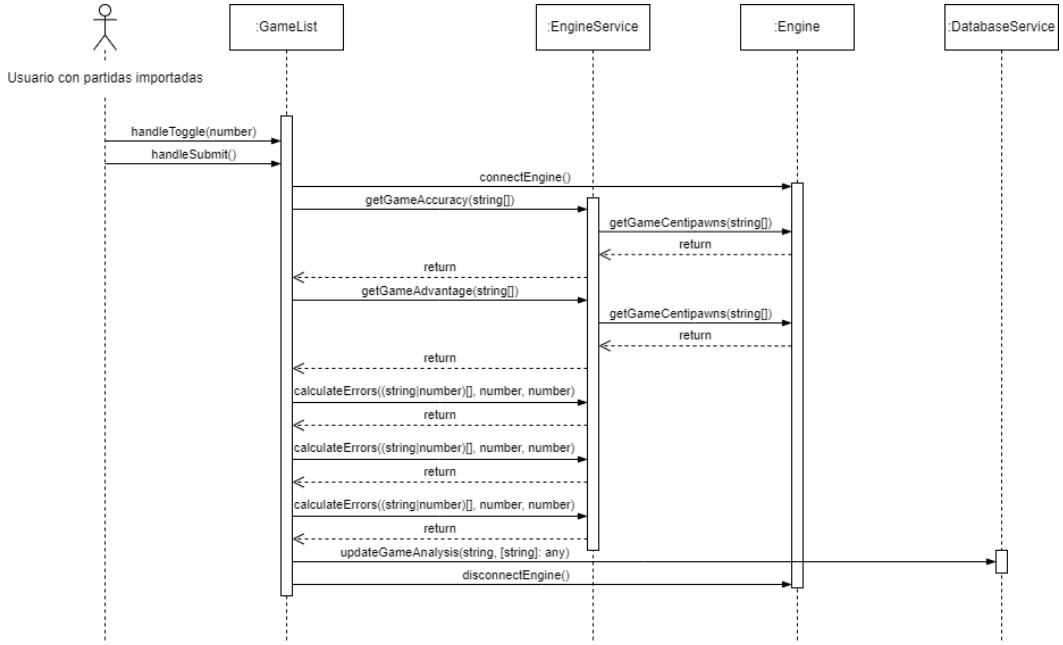


Figura 6.3 Diagrama de interacción sobre analizar partidas

Aunque la secuencia se pueda entender fácilmente viendo el diagrama, hay varios aspectos que requieren mención:

- El usuario solo selecciona una partida. Si seleccionara n juegos, la secuencia desde "getGameAccuracy" hasta "updateGameAnalysis" se repetiría n veces.
- Aunque se podría reutilizar su valor, la función "getGameCentipawns" se llama al calcular la precisión y ventaja, ya que se pretende que estas métricas sean independientes y puedan agregarse o eliminarse según se desee.
- El método "connectEngine" se comunica con el proceso principal para obtener la ruta del código de Stockfish, y el método "updateGameAnalysis" no se llama directamente, sino a través de un canal.

6.2.2 Descargar de Lichess

En este diagrama se detalla el proceso que se desencadena cuando el usuario solicita importar las partidas de un usuario que no están disponibles en la base de datos.

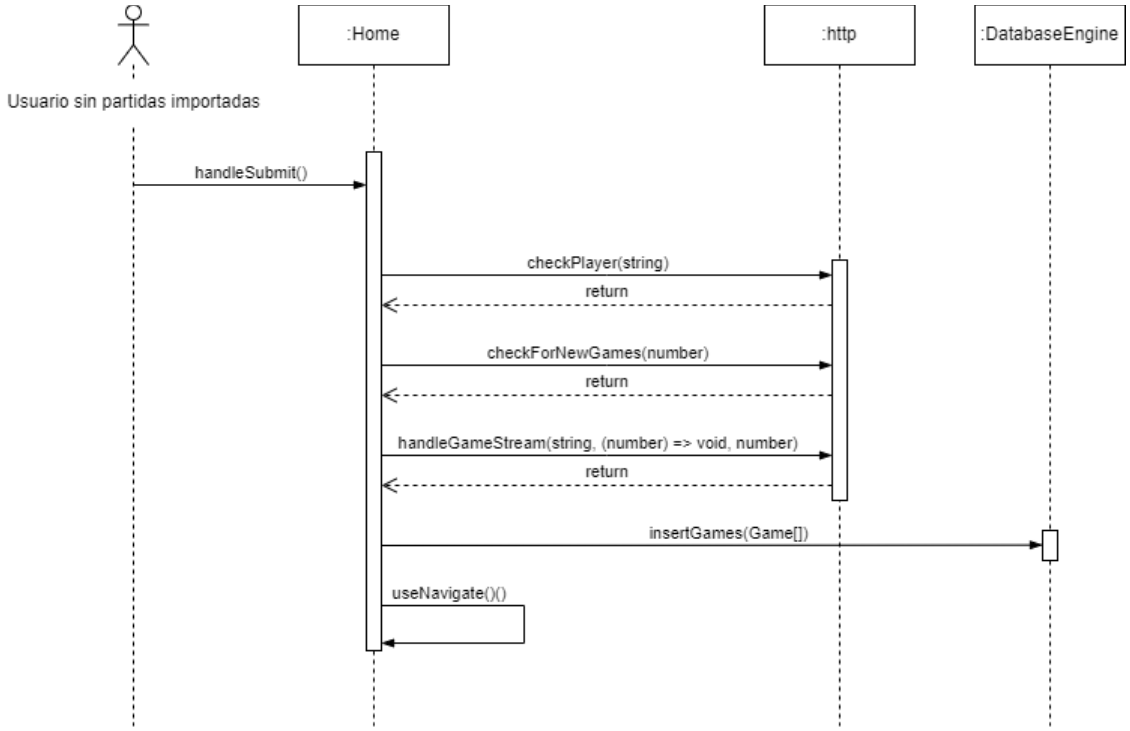


Figura 6.4 Diagrama de interacción sobre descargar de Lichess

Similar al diagrama anterior, es importante tener en cuenta varios factores:

- Si los datos devueltos por la función “checkPlayer” indican que el nombre de usuario introducido es inválido, sucederán los distintos escenarios alternativos y se mostrará un mensaje de error. Si la función “checkForNewGames” confirma que las partidas están disponibles y actualizadas en la base de datos, estas se cargarán desde allí.
- Una vez finalizados la descarga y el almacenamiento de las partidas, se navega a la lista de estas utilizando el Router de React.
- Para insertar juegos, se emplea uno de los canales definidos en el script de precarga.

6.3 Diseño de la Base de Datos

En este apartado, se detallará todo lo relacionado con el sistema de gestión de bases de datos utilizado en la aplicación, incluyendo su integración con el sistema y la tabla creada.

6.3.1 Descripción del SGBD Usado

Como se mencionó en el apartado 3.6, el sistema de gestión de bases de datos utilizado es SQLite3. Para ello, se utiliza la versión 9.4.3 de la librería de Node.js "better-sqlite3", la cual es fiable y más rápida que otras alternativas como "sqlite" y "sqlite3". A continuación, se presenta una tabla comparativa de la velocidad entre estas librerías:

	Recuperar 1 fila	**Recuperar 100 filas**	**Recuperar 100 filas una por una**	**Insertar 1 fila**	**Insertar 100 filas**
better-sqlite3	1x	1x	1x	1x	1x
sqlite and sqlite3	11.7x más lento	2.9x más lento	24.4x más lento	2.8x más lento	15.6x más lento

Tabla 6.1 Comparación de la velocidad entre librerías

6.3.2 Integración del SGBD en Nuestro Sistema

Se utilizarán elementos del subsistema "Capa de persistencia", detallados en la sección 5.4.2.3, para su implementación:

Las partidas descargadas en formato NDJSON se manejarán a través de la interfaz "Game". Estas se mapearán utilizando funciones auxiliares al recuperarlas de la base de datos, y antes de ser insertadas en esta.

La conexión a la base de datos, ubicada en los datos del usuario, se abrirá cuando sea necesario y se cerrará al finalizar su uso. Las nuevas partidas se insertarán en la base de datos una vez completada su descarga.

Por razones de seguridad y eficiencia, todas las funciones relacionadas con la base de datos serán accesibles a través del proceso principal de Electron.

6.3.3 Tabla game

Se optó por no incluir un diagrama entidad-relación debido a que solo habría una entidad: "game". Para simplificar y evitar operaciones innecesarias en el procesamiento de las partidas descargadas de Lichess, se decidió representar su formato mediante una tabla. Esta tabla es esencialmente una traducción directa de la interfaz descrita anteriormente.

Aunque se podría considerar el análisis como una entidad adicional, este proyecto tiene como objetivo facilitar la adición de nuevas métricas sin necesidad de modificar mucho código. Por esta razón, surgió la idea de definir el análisis como un diccionario donde las claves son de tipo “string” y los valores asociados pueden ser de cualquier tipo ("any"), lo cual permite convertirlo en una cadena de texto JSON para su almacenamiento en la base de datos.

Este sistema posibilita añadir nuevos pares de datos sin necesidad de alterar la capa de persistencia. No obstante, si en el futuro se necesitara almacenar una cantidad considerablemente mayor de métricas, sería aconsejable diseñar un nuevo sistema para manejar esta necesidad.

6.4 Diseño de la Interfaz

En este apartado se presentará la interfaz final de la aplicación y sus diferentes componentes. El diseño sigue las pautas establecidas en la fase de análisis, con mejoras específicas para aumentar la usabilidad.

No se incluirá la descripción de cada ventana, ya detallada en la sección 5.7.1. En su lugar, este apartado se centrará en los elementos nuevos o comunes a varias ventanas, explicando su propósito y función.

6.4.1 Ventana de inicio

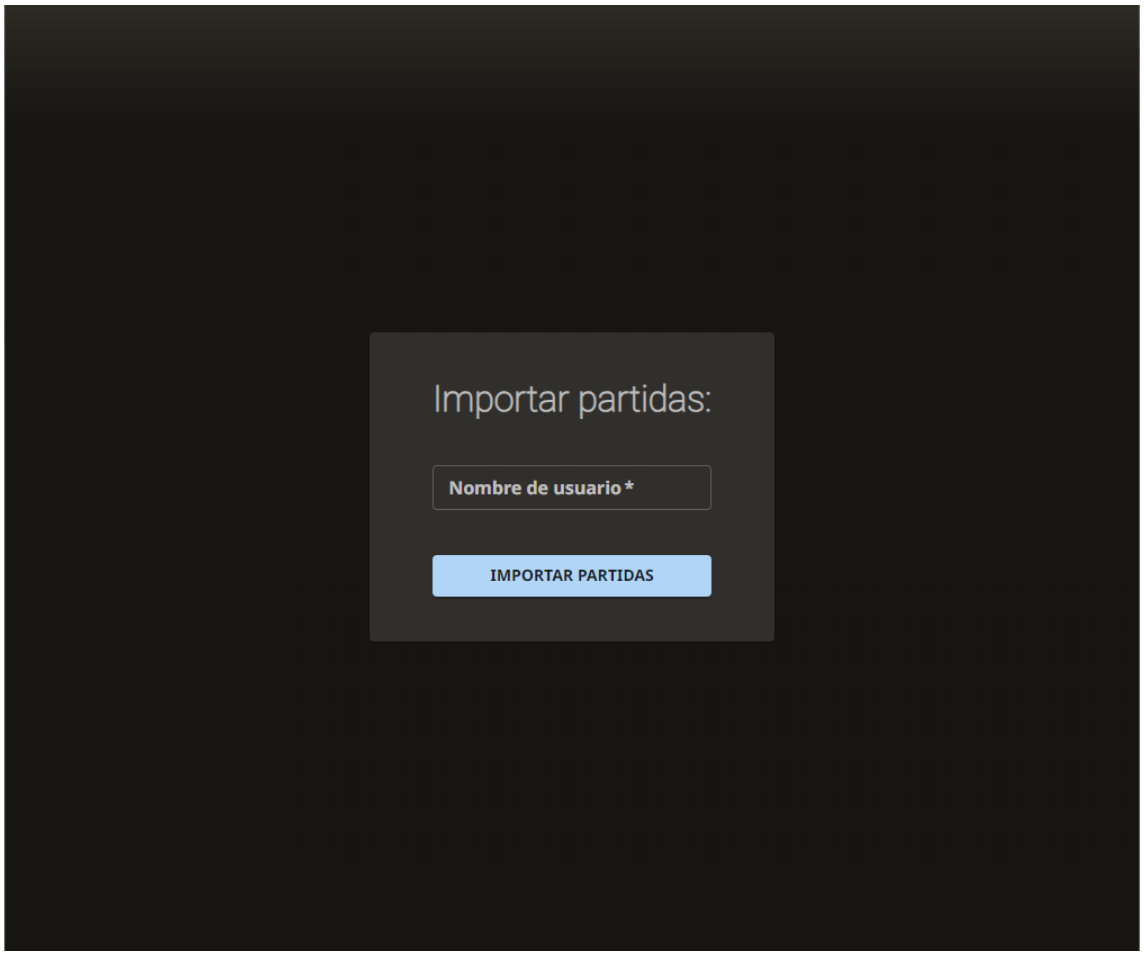


Figura 6.5 Ventana de inicio (diseño)

Como se puede observar, esta página sigue las directrices definidas durante la fase de análisis. Sin embargo, se ha añadido una pantalla de carga entre esta y la lista de partidas. Esta pantalla muestra el progreso de la descarga si se realiza desde Lichess. En el caso de cargar las partidas desde la base de datos, solo se muestra una barra de progreso casi imperceptible, ya que el proceso suele completarse en menos de un segundo.

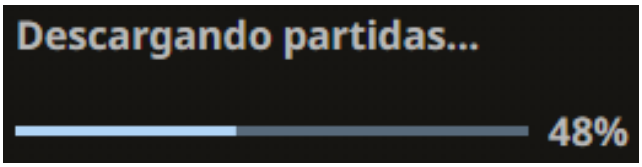


Figura 6.6 Progreso de descarga (diseño)

6.4.2 Ventana con lista de partidas

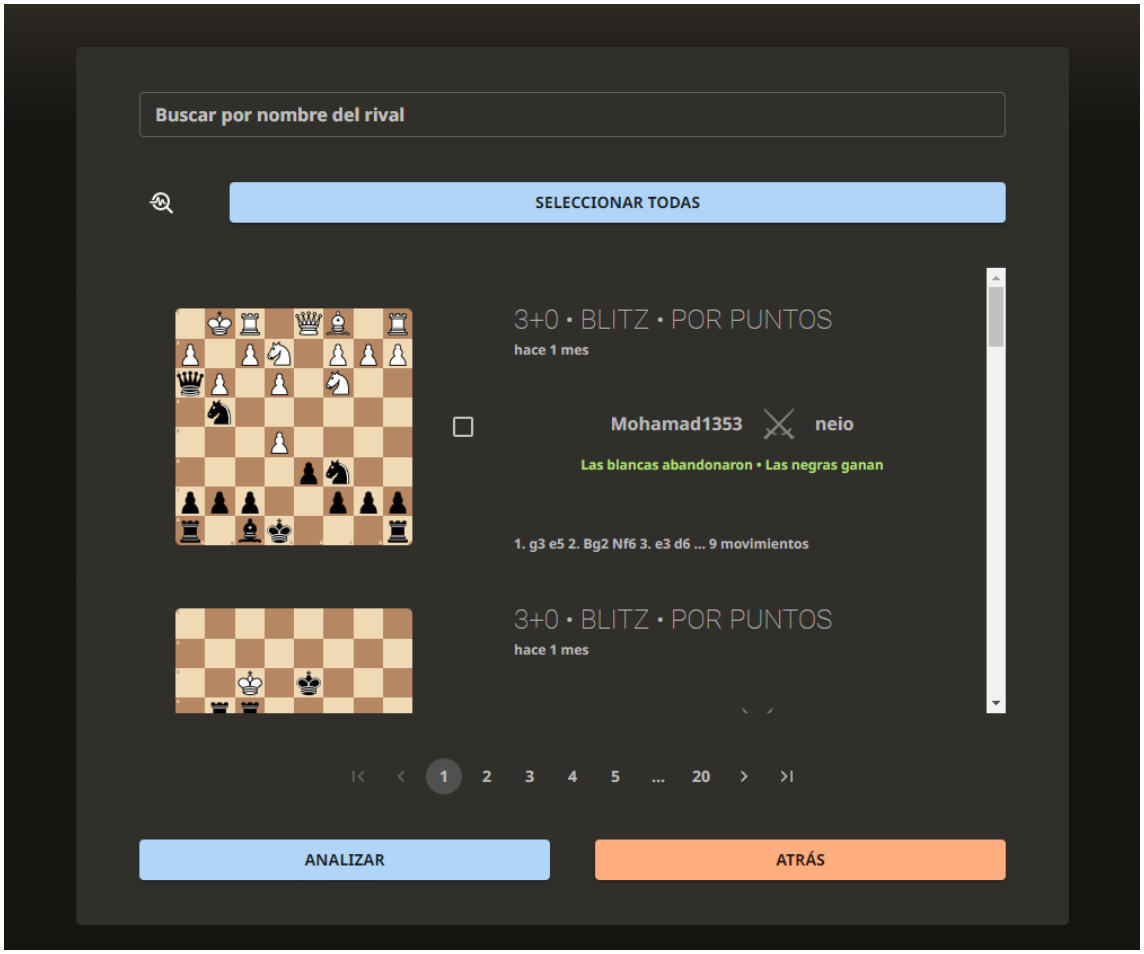


Figura 6.7 Ventana con lista de partidas (diseño)

Para evitar confusiones con una opción de búsqueda, se ha añadido al botón de estadísticas un símbolo con forma de gráfico, facilitando su identificación.

Para cada partida se muestra, de izquierda a derecha y de arriba abajo: el reloj de la partida, el tipo de rendimiento, si es amistosa o por puntos, la fecha en que se jugó, los jugadores enfrentados, el estado de la partida y su ganador (si lo hay), y, por último, un resumen de los movimientos.

Además, se ha añadido una pantalla de carga al solicitar el análisis de partidas. Esta pantalla muestra el progreso del análisis y el tiempo estimado para su finalización, el cual se ajusta con cada iteración.

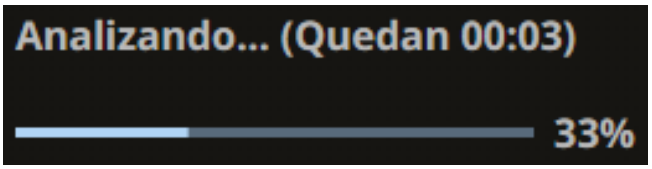


Figura 6.8 Progreso de análisis (diseño)

6.4.3 Ventana de análisis

Figura 6.9 Ventana de análisis (diseño)

En esta ventana, se ha decidido resaltar en color azul el estado actual del tablero, además de utilizar un esquema de colores distinto para cada tipo de error, destacando su importancia.

Al pasar el puntero sobre el gráfico, se indica el estado de ventaja de la partida en ese punto específico.

6.4.4 Ventana de estadísticas

Figura 6.10 Estadísticas en base a victorias (diseño)

Figura 6.11 Estadísticas en base a errores (diseño)ç

Se ha decidido mostrar estadísticas extraídas del número de victorias y de los errores cometidos. Aunque algunas estadísticas puedan tener menos relevancia que otras, como se ha mencionado en varias ocasiones, el propósito de este proyecto es priorizar la facilidad de adición de estas frente a la variedad en primera instancia.

Ambas secciones siguen un diseño similar, mostrando en gráficos el número de victorias, tablas y derrotas, así como las imprecisiones, errores y errores graves en la sección de errores. A continuación, se presentan las victorias y errores con blancas y negras, contra usuarios con mayor y menor ELO, y, por último, un gráfico de barras para los modos de juego con mayor número de victorias y errores.

Se ha optado por que la sección cambie automáticamente tras un tiempo sin posar el ratón sobre ella, y que los botones laterales solo aparezcan al posar el ratón en los laterales, reduciendo el ruido visual y evitando que tapen las estadísticas.

6.4.5 Elementos comunes

La aplicación contará con una barra de herramientas sencilla, que solo proporcionará atajos para cerrar el programa o ampliarlo a pantalla completa. Por esta razón, se admitirán sugerencias de los participantes en las pruebas de usabilidad para mejorar este aspecto, un ajuste que no tomaría más de 5 minutos y se podría realizar en el momento.

El esquema de colores elegido se inspira en la página de Lichess, procurando familiaridad sin imitar directamente. Se intentará escoger elementos que no dificulten la accesibilidad, como los “iframes”, y se utilizan colores con suficiente contraste para cumplir con los estándares.

6.5 Especificación Técnica del Plan de Pruebas

Este apartado explicará cómo se llevarán a cabo las pruebas diseñadas y detallará otros tipos de pruebas que se ejecutarán en el software.

Las pruebas se realizan usando Visual Studio Code en un equipo con Windows 10 Home, equipado con un procesador Ryzen 7700X y 32 GB de RAM DDR5 a 6000 MHz.

6.5.1 Pruebas Unitarias

Como se indicó durante la fase de análisis, se empleará Jest en conjunto con la librería de pruebas de React para crear pruebas unitarias que verifiquen el correcto funcionamiento del cliente y las operaciones de procesamiento de datos del motor de análisis. A continuación, se describe la aplicación de estas pruebas:

6.5.1.1 Cliente

En relación con el cliente, se verificará el correcto funcionamiento de los componentes de cada una de las ventanas descritas anteriormente, simulando el resto de los módulos con los que interactúan. Sin embargo, se excluyen las pruebas relacionadas con gráficos y tableros, debido a la imposibilidad de probarlos. Nos limitaremos, por ejemplo, únicamente a la tabla de movimientos en la ventana de análisis.

Componente raíz	
Prueba	**Resultado esperado**
Debería mostrarse correctamente	La aplicación se muestra correctamente sin ningún error

Componente GameAnalysis	
Prueba	**Resultado esperado**
Debería mostrar el número correcto de filas	El componente muestra tantas filas como la mitad de los movimientos que haya
Prueba	**Resultado esperado**
Debería mostrar los movimientos y ventajas correctos	El componente muestra correctamente los movimientos realizados y la ventaja en ese momento
Prueba	**Resultado esperado**
Debería aplicar los estilos correctos a la celda actual	El componente muestra la celda actual de color azul

Componente Home	
Prueba	**Resultado esperado**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error
Prueba	**Resultado esperado**
Debería mostrar un error si no se ha introducido un nombre de usuario	El componente muestra el mensaje de error "Ingrese un nombre de usuario"
Prueba	**Resultado esperado**
Debería mostrar un error si el usuario introducido no existe	El componente muestra el mensaje de error "El usuario introducido no existe"
Prueba	**Resultado esperado**
Debería mostrar un error si el usuario introducido no ha jugador partidas	El componente muestra el mensaje de error "El usuario no ha jugado partidas"
Prueba	**Resultado esperado**
Debería mostrar un error si el usuario introducido no ha jugador válidas	El componente muestra el mensaje de error "El usuario no ha jugado partidas válidas"
Prueba	**Resultado esperado**
Debería descargar e importar las partidas, si hay nuevas disponibles	Completa el proceso y navega a la lista de partidas
Prueba	**Resultado esperado**
Debería cargar las partidas desde la base de datos, si ya están presentes	Completa el proceso y navega a la lista de partidas

Componente GameList	
Prueba	**Resultado esperado**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error
Prueba	**Resultado esperado**
Debería navegar a la pantalla de inicio al pulsar el botón “Atrás”	El componente limpia el nombre de usuario y redirige a la pantalla de inicio
Prueba	**Resultado esperado**
Debería navegar a las estadísticas al pulsar el botón correspondiente	El componente navega a la ventana de estadísticas
Prueba	**Resultado esperado**
Se deberían poder seleccionar y deseleccionar partidas correctamente	El componente muestra correctamente las partidas seleccionadas y deseleccionadas
Prueba	**Resultado esperado**
Debería analizar las partidas seleccionadas	El componente analiza las partidas seleccionadas y las muestra como tal
Prueba	**Resultado esperado**
Debería filtrar las partidas según el término de búsqueda	El componente muestra únicamente las partidas de “playerC” al buscar ese nombre, y todas las partidas cuando se vacía el término de búsqueda

Componente GameStatistics	
Prueba	**Resultado esperado**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error
Prueba	**Resultado esperado**
Debería mostrar un mensaje cuando no hay partidas o partidas analizadas suficientes	El componente muestra un mensaje indicando que juegues o analices más partidas, según convenga
Prueba	**Resultado esperado**
Debería navegar a la lista de partidas al pulsar el botón “Atrás”	El componente redirige a la lista de partidas

6.5.1.2 Motor de análisis

En relación con el motor de análisis, nos limitaremos a probar únicamente las funciones del módulo EngineService, encargado de procesar las salidas de Stockfish, las cuales serán simuladas. Aunque no se trata de operaciones excesivamente complejas que requieran una prueba exhaustiva, se considera necesario contar con pruebas que verifiquen su adecuado comportamiento, en caso de que se realicen grandes cambios en el futuro.

Módulo EngineService	
Prueba	**Resultado esperado**
La función getGameAccuracy debería calcular la precisión media de forma correcta	La función devuelve la precisión media correspondiente, incluso con valores extremos
Prueba	**Resultado esperado**
La función getGameAdvantage debería calcular la ventaja de forma correcta	La función traduce correctamente la ventaja de centipawns a valores más cercanos al usuario
Prueba	**Resultado esperado**
La función calculateErrors debería calcular las imprecisiones de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 0.5 y 1
Prueba	**Resultado esperado**
La función calculateErrors debería calcular los errores de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 1 y 1.5
Prueba	**Resultado esperado**
La función calculateErrors debería calcular los errores graves de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 1.5 e infinito

6.5.2 Pruebas de Integración y del Sistema

Las pruebas de integración y del sistema se realizarán manualmente para verificar el correcto funcionamiento de la aplicación en su conjunto, utilizando el ejecutable generado tras la fase de desarrollo. A continuación, se detallan las pruebas propuestas:

- Importación de partidas sin introducir nombre de usuario.
- Importación de partidas introduciendo un nombre de usuario inexistente.
- Importación de partidas introduciendo el nombre de un usuario que no haya jugado partidas.
- Importación de partidas introduciendo el nombre de un usuario que no haya jugado partidas compatibles.
- Importación de partidas disponibles en la base de datos.
- Importación de partidas no disponibles en la base de datos.
- Selección, análisis y acceso al análisis de una partida importada.
- Selección y análisis de varias partidas importadas, y acceso a alguno de sus análisis.

6.5.3 Pruebas de Usabilidad y Accesibilidad

Aquí se detallarán los métodos y herramientas utilizados en las pruebas de usabilidad y accesibilidad. Estas pruebas tienen como objetivo verificar la facilidad de uso de las diferentes partes del programa, con el fin de mejorar la experiencia de los usuarios al interactuar con nuestra aplicación y reducir el tiempo requerido para completar diversas tareas en ella.

6.5.3.1 Diseño de Cuestionarios

Como se mencionó anteriormente, las pruebas de usabilidad se llevarán a cabo utilizando cuestionarios, los cuales se describen a continuación.

6.5.3.1.1 Cuestionario de Evaluación

Después de utilizar la aplicación y completar las tareas asignadas, los usuarios deberán completar un cuestionario que abordará los siguientes puntos:

- **1º**: **Preguntas de carácter general** a través de las cuales se determine el tipo de usuario y su nivel de conocimiento informático.
- **2º**: **Actividades guiadas** para hacer con nuestra aplicación.
- **3º**: **Batería de preguntas cortas** con los distintos aspectos de la aplicación que se pretendan evaluar.
- **4º**: **Observaciones**, para que el usuario aporte todo lo que considere oportuno de nuestra aplicación.

Además, el responsable de las pruebas podrá anotar distintos aspectos que observe durante su realización en un formulario adicional.

6.5.3.2 Actividades de las Pruebas de Usabilidad

A continuación, se describen las secciones que contendrán los cuestionarios mencionados.

6.5.3.2.1 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Está familiarizado con el ajedrez, ya sea en su modalidad tradicional o en línea?
1. Sí, juego ajedrez regularmente 2. Sí, aunque no juego frecuentemente 3. No, aunque conozco las reglas básicas 4. No tengo familiaridad con el ajedrez
¿Está familiarizado con los conceptos manejados por la aplicación?
1. Sí, tengo conocimientos avanzados en estos conceptos 2. Sí, tengo conocimientos básicos en estos conceptos 3. No estoy muy familiarizado, pero tengo una comprensión general 4. No tengo familiaridad con estos conceptos

6.5.3.2.2 Actividades guiadas

Los participantes llevarán a cabo las siguientes actividades, durante las cuales discutirán los problemas y dificultades que consideren que tiene la aplicación, si los hay:

- Importar partidas de un usuario
- Buscar una partida específica utilizando el nombre del rival
- Seleccionar y analizar una partida
- Acceder al análisis detallado de una partida
- Mostrar un momento específico de una partida en el tablero
- Ver las estadísticas de un usuario

6.5.3.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?				
¿Existe ayuda para las funciones en caso de que tenga dudas?				
¿Le resulta sencillo el uso de la aplicación?				
¿Conoce dónde dirigirse para realizar las acciones que desea?				
¿Sabe cómo navegar por la aplicación utilizando sus botones?				
¿La aplicación le notifica sobre los resultados de sus acciones?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿Sabe cómo buscar una partida específica de un usuario determinado?				
¿El tiempo de respuesta de la aplicación para los análisis es muy grande?				
¿Considera que la calidad de los análisis obtenidos es adecuada?				
¿Le han sido útiles los análisis obtenidos?				
¿Le han sido útiles las estadísticas extraídas?				
¿Considera que el número de estadísticas extraídas es suficiente?				

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				
Cualquier comentario del usuario				

6.5.3.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
Consultas del usuario	
Dificultades encontradas	

6.5.3.3 Pruebas de Accesibilidad

En cuanto a la accesibilidad, se verificará el contraste de los colores utilizando la herramienta Colour Contrast Analyser (CCA). Además, se completará el checklist proporcionado por WCAG para asegurar el cumplimiento de las pautas de accesibilidad de una aplicación web, el cual está disponible a continuación:

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	**Sí**	**No**	**N/A**
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.			
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.			
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			

6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.			
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.			
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.			
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.			
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.			
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.			
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.			
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda visual de una presentación multimedia.			
1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad) equivalente y sea actualizada tan a menudo como la página (original) inaccesible.			

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	Sí	No	N/A
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].			
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.			
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.			
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.			
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.			
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.			
3.6 Marque correctamente las listas y los ítems de las listas.			
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.			
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.			
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).			
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.			
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.			
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.			
11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean soportadas.			
11.2 Evite características desaconsejadas por las tecnologías *W3C*.			
12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.			
13.1 Identifique claramente el objetivo de cada vínculo.			
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.			
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).			
13.4 Utilice los mecanismos de navegación de forma coherente.			

Y si utiliza tablas (Prioridad 2)	Sí	No	N/A
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).			
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.			
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.			
12.4 Asocie explícitamente las etiquetas con sus controles.			
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.			
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].			
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.			
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	Sí	No	N/A
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			
4.3 Identifique el idioma principal de un documento.			
9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.			
9.5 Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.			
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).			
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.			

13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo hagan, proporcione una manera de evitar el grupo.			
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.			
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.			
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).			
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.			
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.			
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.			
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.			
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.			
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.			
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.			
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.			

6.5.4 Pruebas de Rendimiento

El análisis de partidas requerirá una cantidad específica de recursos que se procurará minimizar, manteniendo un nivel adecuado de calidad. Para ello, se medirá el tiempo de procesamiento necesario para analizar varias partidas y se registrarán los resultados obtenidos con diferentes cantidades de nodos a explorar.

Esta métrica permite mantener la consistencia de los resultados a lo largo del tiempo y entre distintos equipos, a diferencia de la profundidad o el tiempo de exploración. Se seleccionará cuidadosamente, evaluando si la similitud de los valores obtenidos con el análisis de Lichess justifica el tiempo utilizado.

Para este propósito, se empleará la distancia euclidiana, que mide la distancia "recta" entre dos puntos en un espacio multidimensional. Cuanto menor sea este valor, más similares serán las dos listas.

Capítulo 7. Implementación del Sistema

Este capítulo abordará los estándares, lenguajes de programación y herramientas empleados en la implementación del sistema, así como los problemas encontrados durante dicho proceso.

7.1 Estándares y Normas Seguidos

En este apartado se describirán los estándares y normas utilizados en nuestra aplicación durante el desarrollo del código, así como las medidas tomadas para asegurar que dichos estándares se cumplan efectivamente.

7.1.1 CSS3

Si bien algunos estilos de ciertos componentes necesitan estar presentes en el código, se ha procurado ubicar la mayor cantidad posible de estos en las hojas de estilo. Estas hojas siguen los estándares del W3C y se ha confirmado mediante el servicio de validación de CSS del W3C que son documentos CSS versión 3 válidos.

Figura 7.1 CSS válido por W3C

7.1.2 JSON

Se ha empleado ND-JSON para descargar partidas desde la API de Lichess. ND-JSON es un formato de texto donde cada línea contiene un objeto JSON independiente, separados por saltos de línea.

Aunque ND-JSON es ampliamente utilizado, no está estandarizado debido a la existencia de formatos similares como JSON Lines. Por otro lado, JSON sí está estandarizado: ECMA International desarrolló el Estándar ECMA-404, que especifica la estructura de JSON.

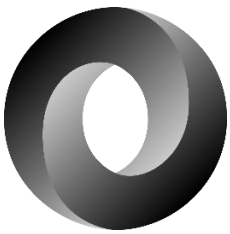


Figura 7.2 Logo de JSON

7.2 Lenguajes de Programación

En este apartado se detallarán los lenguajes de programación utilizados, así como los módulos o complementos asociados a cada uno de ellos.

7.2.1 JavaScript y TypeScript

En el proyecto se utilizan exclusivamente TypeScript y JavaScript como lenguajes de programación. Aunque la mayor parte del desarrollo se realiza en TypeScript, se conserva la implementación en JavaScript del motor y los archivos auxiliares necesarios para cargarlo en un trabajador web. Para obtener más información sobre estos lenguajes, se puede consultar el apartado 3.5.

7.2.2 MUI

La mayoría de los elementos del front-end han sido desarrollados utilizando la versión 5.15.6 de Material-UI, una biblioteca también conocida como MUI, creada por Olivier Tassinari y Matteo Mazzarolo. Esta herramienta está específicamente diseñada para React y sigue las directrices visuales de Material Design de Google.

Material-UI goza de una amplia adopción entre grandes empresas y organizaciones tecnológicas como Google, Apple, Microsoft, Airbnb y Netflix, gracias a su popularidad y su integración fluída con React.

Figura 7.3 Logo de MUI

7.2.3 Chess.js

La simulación de partidas de ajedrez y la traducción de notación algebraica a FEN en este proyecto se llevan a cabo utilizando Chess.js en su versión 1.0.0-beta.7.

Chess.js es una biblioteca de JavaScript desarrollada por Jeff Hlywa, que facilita la gestión de la lógica del juego de ajedrez en aplicaciones web. Esta herramienta permite a los desarrolladores implementar y administrar las reglas de movimiento, validar movimientos y estructurar el juego de ajedrez dentro de sus aplicaciones.

7.3 Herramientas y Programas Usados para el Desarrollo

En este apartado se detallan todas las herramientas de desarrollo, complementos y otros productos de software necesarios para la implementación de nuestro sistema.

7.3.1 Visual Studio Code

Se ha empleado Visual Studio Code versión 1.90.2 como entorno de desarrollo para este proyecto, basándonos en los motivos expuestos en el apartado 3.2.

7.3.2 DB Browser for SQLite

Se utilizó la versión 3.12.2 de DB Browser for SQLite para visualizar la base de datos SQLite utilizada en el proyecto, donde se almacenan las partidas descargadas y sus análisis. Esta herramienta permitió examinar el esquema de la tabla y revisar sus contenidos.

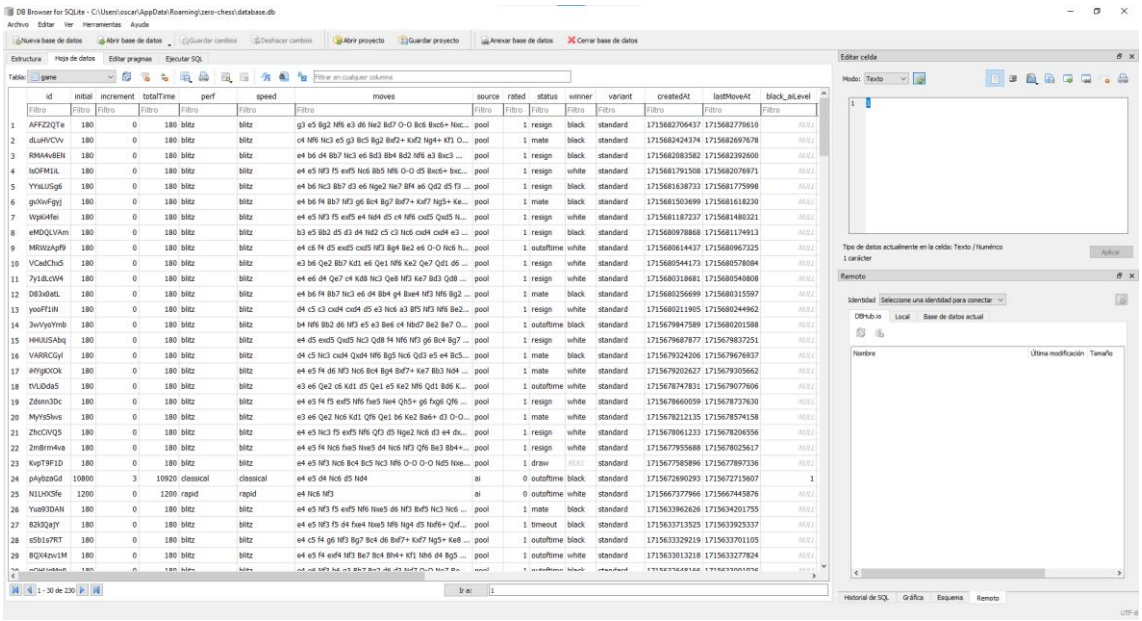


Figura 7.4 Interfaz de DB Browser for SQLite

7.3.3 Electron Builder

Se empleó la versión 24.6.4 de electron-builder para crear el ejecutable de la aplicación. Esta herramienta proporciona una solución completa para empaquetar y construir una aplicación Electron lista para distribuir en MacOS, Windows y Linux, con soporte integrado para actualizaciones automáticas.

```
PS C:\Users\oscar\Documents\Universidad\TFG\tfg> npm run package

> package
> ts-node ./.erb/scripts/clean.js dist && npm run build && electron-builder build --publish never && npm run build:dll


> build
> concurrently "npm run build:main" "npm run build:renderer"

[0]
[0] > build:main
[0] > cross-env NODE_ENV=production TS_NODE_TRANSPILE_ONLY=true webpack --config ./.erb/configs/webpack.config.main.prod.ts
[0]
[1]
[1] > build:renderer
[1] > cross-env NODE_ENV=production TS_NODE_TRANSPILE_ONLY=true webpack --config ./.erb/configs/webpack.config.renderer.prod.ts
[1]
[0] npm run build:main exited with code 0
[1] npm run build:renderer exited with code 0
  • electron-builder  version=24.13.3 os=10.0.19045
  • loaded configuration  file=package.json ("build" field)
  • writing effective config  file=release\build\builder-effective-config.yaml
  • rebuilding native dependencies  dependencies=better-sqlite3@9.4.3 platform=win32 arch=x64
  • install prebuilt binary  name=better-sqlite3 version=9.4.3 platform=win32 arch=x64 napi=
  • packaging       platform=win32 arch=x64 electron=27.0.0 appOutDir=release\build\win-unpacked
  • building        target=nsis file=release\build\ZeroChess Setup 1.0.0.exe archs=x64 oneClick=true perMachine=false
  • building block map  blockMapFile=release\build\ZeroChess Setup 1.0.0.exe.blockmap

> build:dll
> cross-env NODE_ENV=development TS_NODE_TRANSPILE_ONLY=true webpack --config ./.erb/configs/webpack.config.renderer.dev.dll.ts
```

Figura 7.5 Salida del proceso de empaquetado

7.4 Creación del Sistema

En este apartado se detallan todos los aspectos observados durante la implementación del sistema.

7.4.1 Problemas Encontrados

Aunque la documentación generada en las etapas iniciales del proyecto ayudó a establecer la estructura y los límites para el desarrollo de la plataforma, esto no impidió que surgieran problemas durante el proceso. A continuación, se enumeran dichos problemas junto con las soluciones que se implementaron:

7.4.1.1 Electron

Este proyecto implicó la necesidad de aprender un framework hasta entonces desconocido: Electron. Aunque no se trata de un entorno excesivamente complejo, ciertos aspectos como el aislamiento del contexto y la comunicación entre procesos requirieron tiempo de estudio para ser documentados adecuadamente y utilizarse correctamente en la aplicación.

7.4.1.2 Descarga y paginación de partidas

Durante las primeras etapas del desarrollo del backend y su integración con la base de datos, se identificó un problema común: algunos jugadores tenían demasiadas partidas para ser manejadas de forma convencional. Esto provocaba ralentizaciones y caídas de la aplicación al mostrar las partidas en pantalla, ya que renderizar el tablero para cada partida implicaba un alto costo. Además, al guardar las partidas en la base de datos, se planteó inicialmente hacer las inserciones dentro de un bucle, lo que también resultaba ineficiente.

Sin embargo, este problema tuvo una solución sencilla: reducir el número de partidas mostradas por página a una cantidad razonable que no causara dificultades, y realizar las inserciones en lotes y en una sola transacción. Esta estrategia ha demostrado funcionar sorprendentemente bien, incluso para usuarios con miles de partidas.

7.4.1.3 Rutas del motor y la base de datos

Una vez se inició la integración con el motor de análisis Stockfish, se intentó implementarlo en un trabajador web para interactuar y obtener los resultados deseados. Sin embargo, se desconocía que ciertos aspectos relacionados con las rutas solo funcionaban en el proceso principal, utilizando la suscripción a canales explicada en múltiples ocasiones.

Después de resolver este problema, la ruta seguía sin coincidir con la del archivo, lo cual se corrigió utilizando la variable de entorno "__dirname", que indica la ruta absoluta del directorio que contiene el archivo en ejecución.

No obstante, la solución aún no funcionaba debido a restricciones de seguridad que impedían servir archivos al trabajador web: configuraciones CORS que finalmente se ajustaron en el proceso principal.

A partir de esta problemática, se pudo deducir fácilmente la ruta de la base de datos, pero al empaquetar la aplicación, tanto el motor como la base de datos dejaban de funcionar. Como solución definitiva, se decidió ubicar la base de datos en el directorio "userData" y no empaquetar el código del motor, que es público y no me pertenece. Esto permite que la aplicación funcione correctamente tanto en el entorno de desarrollo como en producción.

7.4.1.4 Expresiones regulares del motor

El código utilizado para cargar el motor en un trabajador web se extrajo del repositorio de la implementación de Stockfish usada. Aunque la interfaz inicialmente parecía estar funcionando correctamente, se descubrió que algunas expresiones regulares utilizadas para determinar si el motor había completado una tarea estaban desactualizadas y algunos condicionales eran incorrectos. Corregir este problema fue relativamente sencillo, pero identificar la causa del fallo llevó bastante tiempo.

7.4.1.5 Operaciones de procesado

Para obtener los valores mostrados al usuario en los análisis, nos basamos en el modelo establecido por Lichess.com según lo descrito en la sección 3.7.1. Aunque las funciones matemáticas tenían una traducción directa al código, los valores utilizados por estas funciones no coincidían con los retornados por el motor. Para resolver este problema, se dedicó tiempo a consultar la documentación oficial y desarrollar una solución óptima para la conversión de estos valores.

Además, aspectos como el cálculo de errores y precisión media implican el uso de funciones complejas que consideran diversos parámetros como el modo de juego, el momento en el que ocurren y el Elo de los jugadores involucrados. Estas funciones no están disponibles para consulta externa y su diseño excede el alcance del proyecto. Por lo tanto, se optó por un enfoque simplificado para el cálculo de errores, basado en la diferencia de ventaja entre dos umbrales y en la penalización cuadrática de los errores para la precisión. Este método nos permitió obtener resultados similares y convincentes.

Capítulo 8. Desarrollo de las Pruebas

Este capítulo presentará los resultados obtenidos en las pruebas descritas en el apartado 6.5.

8.1 Pruebas Unitarias

En este apartado se mostrarán los resultados de las pruebas unitarias que ya se han diseñado y descrito previamente.

8.1.1 Cliente

Componente raíz		
Prueba	**Resultado esperado**	**Superada**
Debería mostrarse correctamente	La aplicación se muestra correctamente sin ningún error	Sí

Componente Home		
Prueba	**Resultado esperado**	**Superada**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar un error si no se ha introducido un nombre de usuario	El componente muestra el mensaje de error "Ingrese un nombre de usuario"	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar un error si el usuario introducido no existe	El componente muestra el mensaje de error "El usuario introducido no existe"	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar un error si el usuario introducido no ha jugador partidas	El componente muestra el mensaje de error "El usuario no ha jugado partidas"	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar un error si el usuario introducido no ha jugador válidas	El componente muestra el mensaje de error "El usuario no ha jugado partidas válidas"	Sí
Prueba	**Resultado esperado**	**Superada**
Debería descargar e importar las partidas, si hay nuevas disponibles	Completa el proceso y navega a la lista de partidas	Sí
Prueba	**Resultado esperado**	**Superada**
Debería cargar las partidas desde la base de datos, si ya están presentes	Completa el proceso y navega a la lista de partidas	Sí

Componente GameList		
Prueba	**Resultado esperado**	**Superada**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error	Sí
Prueba	**Resultado esperado**	**Superada**
Debería navegar a la pantalla de inicio al pulsar el botón “Atrás”	El componente limpia el nombre de usuario y redirige a la pantalla de inicio	Sí
Prueba	**Resultado esperado**	**Superada**
Debería navegar a las estadísticas al pulsar el botón correspondiente	El componente navega a la ventana de estadísticas	Sí
Prueba	**Resultado esperado**	**Superada**
Se deberían poder seleccionar y deseleccionar partidas correctamente	El componente muestra correctamente las partidas seleccionadas y deseleccionadas	Sí
Prueba	**Resultado esperado**	**Superada**
Debería analizar las partidas seleccionadas	El componente analiza las partidas seleccionadas y las muestra como tal	Sí
Prueba	**Resultado esperado**	**Superada**
Debería filtrar las partidas según el término de búsqueda	El componente muestra únicamente las partidas de “playerC” al buscar ese nombre, y todas las partidas cuando se vacía el término de búsqueda	Sí

Componente GameAnalysis		
Prueba	**Resultado esperado**	**Superada**
Debería mostrar el número correcto de filas	El componente muestra tantas filas como la mitad de los movimientos que haya	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar los movimientos y ventajas correctos	El componente muestra correctamente los movimientos realizados y la ventaja en ese momento	Sí
Prueba	**Resultado esperado**	**Superada**
Debería aplicar los estilos correctos a la celda actual	El componente muestra la celda actual de color azul	Sí

Componente GameStatistics		
Prueba	**Resultado esperado**	**Superada**
Debería mostrarse correctamente	El componente se muestra correctamente sin ningún error	Sí
Prueba	**Resultado esperado**	**Superada**
Debería mostrar un mensaje cuando no hay partidas o partidas analizadas suficientes	El componente muestra un mensaje indicando que juegues o analices más partidas, según convenga	Sí
Prueba	**Resultado esperado**	**Superada**
Debería navegar a la lista de partidas al pulsar el botón “Atrás”	El componente redirige a la lista de partidas	Sí

8.1.2 Motor de análisis

Módulo EngineService		
Prueba	**Resultado esperado**	**Superada**
La función getGameAccuracy debería calcular la precisión media de forma correcta	La función devuelve la precisión media correspondiente, incluso con valores extremos	Sí
Prueba	**Resultado esperado**	**Superada**
La función getGameAdvantage debería calcular la ventaja de forma correcta	La función traduce correctamente la ventaja de centipawns a valores más cercanos al usuario	Sí
Prueba	**Resultado esperado**	**Superada**
La función calculateErrors debería calcular las imprecisiones de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 0.5 y 1	Sí
Prueba	**Resultado esperado**	**Superada**
La función calculateErrors debería calcular los errores de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 1 y 1.5	Sí
Prueba	**Resultado esperado**	**Superada**
La función calculateErrors debería calcular los errores graves de forma correcta	La función devuelve el número de errores cometidos entre los umbrales 1.5 e infinito	Sí

A continuación, se presentan los resultados de la ejecución de las pruebas unitarias descritas:

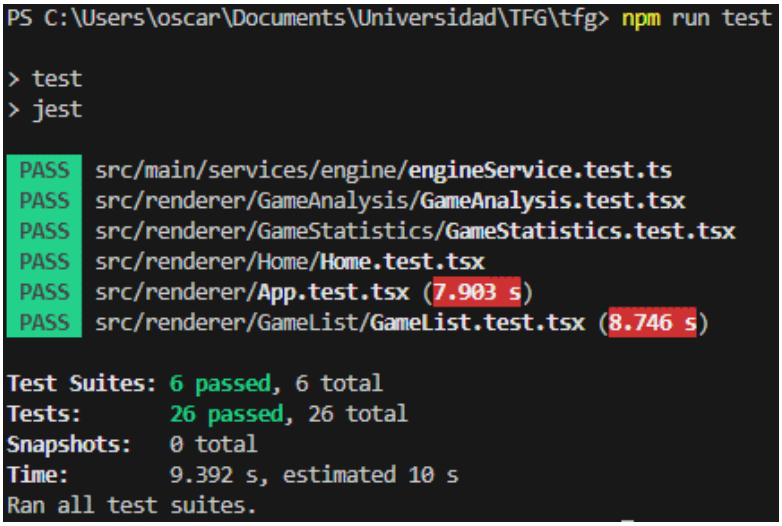


Figura 8.1 Resultados de la ejecución de las pruebas unitarias

8.2 Pruebas de Integración y del Sistema

En este apartado se presentan los resultados de la ejecución de las pruebas funcionales propuestas en la sección 6.5.2.

Importación de partidas		
Prueba	**Resultado esperado**	**Superada**
Se intenta importar partidas sin introducir un nombre de usuario	Se muestra el mensaje de error "Ingrese un nombre de usuario"	Sí
Prueba	**Resultado esperado**	**Superada**
Se intenta importar partidas introduciendo un nombre de usuario inexistente	Se muestra el mensaje de error "El usuario introducido no existe"	Sí
Prueba	**Resultado esperado**	**Superada**
Se intenta importar partidas introduciendo el nombre de un usuario que no haya jugado partidas	Se muestra el mensaje de error "El usuario no ha jugado partidas"	Sí
Prueba	**Resultado esperado**	**Superada**
Se intenta importar partidas introduciendo el nombre de un usuario que no haya jugado partidas compatibles	Se muestra el mensaje de error "El usuario no ha jugado partidas válidas"	Sí
Prueba	**Resultado esperado**	**Superada**
Se importan partidas de un usuario disponibles en la base de datos	Se cargan y se muestran en formato lista	Sí
Prueba	**Resultado esperado**	**Superada**
Se importan partidas de un usuario no disponibles en la base de datos	Se descargan de Lichess y se muestran en formato lista	Sí

Selección, análisis y acceso al análisis de partidas importadas		
Prueba	**Resultado esperado**	**Superada**
Se selecciona una partida importada, se solicita su análisis y se accede a este una vez disponible	La partida se analiza sin producirse errores, se muestra como analizada y al pulsar en ella se accede a su análisis	Sí
Prueba	**Resultado esperado**	**Superada**
Se seleccionan varias partidas importadas, se solicita su análisis y se accede a uno de ellos una vez disponibles	Las partidas se analizan sin errores, mostrando el progreso y tiempo restante; al finalizar, se muestran como analizadas y al pulsar en una de ellas se accede a su análisis	Sí

Como se puede observar, todas las pruebas de integración propuestas se superaron sin problemas.

8.3 Pruebas de Usabilidad y Accesibilidad

En este apartado se mostrarán los resultados obtenidos de las pruebas de usabilidad y accesibilidad.

8.3.1 Pruebas de Usabilidad

Aquí se presentan los resultados de los cuestionarios aplicados a todos los usuarios que participaron en las pruebas de usabilidad. El grupo de usuarios incluyó a tres personas con diferentes rangos de edad, niveles tecnológicos y conocimientos de ajedrez:

- Una persona de 21 años que trabaja en informática y tiene conocimientos básicos de ajedrez.
- Una persona de 48 años que utiliza el ordenador para tareas básicas de ofimática, pero no está familiarizada con las reglas del ajedrez.
- Una persona de 63 años que no suele usar el ordenador, pero es profesor de ajedrez en un instituto.

8.3.1.1 Usuario 1

En esta sección se mostrarán los resultados obtenidos para el primer usuario.

8.3.1.1.1 Datos personales

- Ocupación: Desarrollador junior
- Edad: 21 años

8.3.1.1.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días * 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión * 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares * 3. No, nunca
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias *
¿Está familiarizado con el ajedrez, ya sea en su modalidad tradicional o en línea?
1. Sí, juego ajedrez regularmente 2. Sí, aunque no juego frecuentemente 3. No, aunque conozco las reglas básicas * 4. No tengo familiaridad con el ajedrez
¿Está familiarizado con los conceptos manejados por la aplicación?
1. Sí, tengo conocimientos avanzados en estos conceptos 2. Sí, tengo conocimientos básicos en estos conceptos 3. No estoy muy familiarizado, pero tengo una comprensión general * 4. No tengo familiaridad con estos conceptos

8.3.1.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?	X			
¿Existe ayuda para las funciones en caso de que tenga dudas?		X		
¿Le resulta sencillo el uso de la aplicación?	X			
¿Conoce dónde dirigirse para realizar las acciones que desea?	X			
¿Sabe cómo navegar por la aplicación utilizando sus botones?	X			
¿La aplicación le notifica sobre los resultados de sus acciones?	X			
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?	X			
¿Sabe cómo buscar una partida específica de un usuario determinado?	X			
¿El tiempo de respuesta de la aplicación para los análisis es muy grande?				X
¿Considera que la calidad de los análisis obtenidos es adecuada?		X		
¿Le han sido útiles los análisis obtenidos?		X		
¿Le han sido útiles las estadísticas extraídas?			X	
¿Considera que el número de estadísticas extraídas es suficiente?	X			

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son		X		
Los colores empleados son	X			
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?		X		
¿Cree que el programa está bien estructurado?		X		

Observaciones
Sería bueno añadir más tipos de gráficos y estadísticas en la pestaña correspondiente.

Se tendrá en cuenta la sugerencia para futuras ampliaciones. Sin embargo, en este proyecto se ha optado por extraer estadísticas similares a las propuestas por otros sistemas, sin realizar operaciones adicionales y respetando lo expuesto en la sección 5.1.1.

8.3.1.1.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	Notas
El usuario comienza a trabajar de forma rápida por las tareas	Sí, después de una breve explicación sobre el objetivo y el funcionamiento de la aplicación
Tiempo en realizar cada tarea	Breve
Errores leves cometidos	No
Errores graves cometidos	No
Consultas del usuario	¿Qué unidades se utilizan en el análisis? ¿En qué te basas para calcular la precisión?
Dificultades encontradas	Ninguna

8.3.1.2 Usuario 2

En esta sección se mostrarán los resultados obtenidos para el segundo usuario.

8.3.1.2.1 Datos personales

- Ocupación: Ama de casa
- Edad: 48 años

8.3.1.2.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana * 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office * 4. Únicamente leo el correo y navego ocasionalmente
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca *

¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar 2. Que sea intuitivo * 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Está familiarizado con el ajedrez, ya sea en su modalidad tradicional o en línea?
1. Sí, juego ajedrez regularmente 2. Sí, aunque no juego frecuentemente 3. No, aunque conozco las reglas básicas 4. No tengo familiaridad con el ajedrez *
¿Está familiarizado con los conceptos manejados por la aplicación?
1. Sí, tengo conocimientos avanzados en estos conceptos 2. Sí, tengo conocimientos básicos en estos conceptos 3. No estoy muy familiarizado, pero tengo una comprensión general 4. No tengo familiaridad con estos conceptos *

8.3.1.2.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?		X		
¿Existe ayuda para las funciones en caso de que tenga dudas?			X	
¿Le resulta sencillo el uso de la aplicación?		X		
¿Conoce dónde dirigirse para realizar las acciones que desea?		X		
¿Sabe cómo navegar por la aplicación utilizando sus botones?	X			
¿La aplicación le notifica sobre los resultados de sus acciones?	X			
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		X		
¿Sabe cómo buscar una partida específica de un usuario determinado?	X			
¿El tiempo de respuesta de la aplicación para los análisis es muy grande?				X
¿Considera que la calidad de los análisis obtenidos es adecuada?	X			

¿Le han sido útiles los análisis obtenidos?			X	
¿Le han sido útiles las estadísticas extraídas?			X	
¿Considera que el número de estadísticas extraídas es suficiente?	X			
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es		X		
Los iconos e imágenes usados son	X			
Los colores empleados son	X			
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?				X
¿Cree que el programa está bien estructurado?		X		
Observaciones				
Se podría indicar la relación de los errores con el gráfico y la tabla de movimientos.				

Se tomará en cuenta la sugerencia para futuras ampliaciones. No obstante, los gráficos utilizados (MUI X) no permitirían este cambio. Además, el número de errores calculados a veces es considerablemente alto en comparación con otras páginas debido al método empleado, lo que dificultaría la comprensión si se mostraran en la tabla de movimientos.

8.3.1.2.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	Ha sido necesario proporcionar una explicación detallada sobre la aplicación para el usuario.
Tiempo en realizar cada tarea	Breve, aunque se estanca en ciertas tareas
Errores leves cometidos	No
Errores graves cometidos	Confundir el formulario de importación con uno de registro
Consultas del usuario	¿Qué significan estas letras y números (los movimientos y la ventaja)?
Dificultades encontradas	Ninguna

8.3.1.3 Usuario 3

En esta sección se mostrarán los resultados obtenidos para el tercer usuario.

8.3.1.3.1 Datos personales

- Ocupación: Jubilado, profesor de ajedrez en tiempos libres
- Edad: 63 años

8.3.1.3.2 Preguntas de carácter general

¿Usa un ordenador frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente * 4. Nunca o casi nunca
¿Qué tipo de actividades realiza con el ordenador?
1. Es parte de mi trabajo o profesión 2. Lo uso básicamente para ocio 3. Solo empleo aplicaciones estilo Office 4. Únicamente leo el correo y navego ocasionalmente *
¿Ha usado alguna vez software como el de esta prueba?
1. Sí, he empleado software similar 2. No, aunque si empleo otros programas que me ayudan a realizar tareas similares 3. No, nunca *
¿Qué busca Vd. Principalmente en un programa?
1. Que sea fácil de usar * 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Está familiarizado con el ajedrez, ya sea en su modalidad tradicional o en línea?
1. Sí, juego ajedrez regularmente * 2. Sí, aunque no juego frecuentemente 3. No, aunque conozco las reglas básicas 4. No tengo familiaridad con el ajedrez
¿Está familiarizado con los conceptos manejados por la aplicación?
1. Sí, tengo conocimientos avanzados en estos conceptos 2. Sí, tengo conocimientos básicos en estos conceptos * 3. No estoy muy familiarizado, pero tengo una comprensión general 4. No tengo familiaridad con estos conceptos

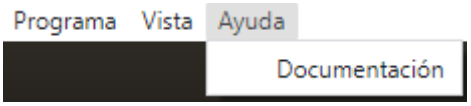
8.3.1.3.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación?			X	
¿Existe ayuda para las funciones en caso de que tenga dudas?			X	
¿Le resulta sencillo el uso de la aplicación?		X		
¿Conoce dónde dirigirse para realizar las acciones que desea?			X	
¿Sabe cómo navegar por la aplicación utilizando sus botones?		X		
¿La aplicación le notifica sobre los resultados de sus acciones?	X			
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		X		
¿Sabe cómo buscar una partida específica de un usuario determinado?				X
¿El tiempo de respuesta de la aplicación para los análisis es muy grande?				X
¿Considera que la calidad de los análisis obtenidos es adecuada?	X			
¿Le han sido útiles los análisis obtenidos?	X			
¿Le han sido útiles las estadísticas extraídas?		X		
¿Considera que el número de estadísticas extraídas es suficiente?	X			

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	X			
Los iconos e imágenes usados son			X	
Los colores empleados son	X			
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		X		
¿El diseño de las pantallas es claro y atractivo?				X
¿Cree que el programa está bien estructurado?		X		

Observaciones
Sería bueno agregar algún tipo de ayuda para explicar la utilidad de la aplicación y cómo realizar las tareas más comunes.

Como se mencionó en la sección 6.4.5, las sugerencias sobre la barra de herramientas serán bienvenidas e implementadas en el momento. Por esta razón, se ha decidido dejar disponible este documento en un apartado denominado "Documentación".



8.3.1.3.4 Cuestionario para el Responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	No ha sido necesario explicar la utilidad de la aplicación, pero sí cómo utilizarla a fondo
Tiempo en realizar cada tarea	Medio, sorprendentemente rápido para su perfil
Errores leves cometidos	No discernir las partidas analizadas de las que no lo están
Errores graves cometidos	Confundir el botón de estadísticas con una función de búsqueda
Consultas del usuario	¿Qué introduzco aquí (nombre de usuario)?
Dificultades encontradas	Ninguna

Este usuario ha comentado que los datos se presentan de manera adecuada y clara en los análisis. Además, sugiere la posibilidad de añadir nuevos tipos de estadísticas en el futuro, aunque encuentra interesantes las que están actualmente disponibles.

8.3.2 Pruebas de Accesibilidad

A continuación, se presentarán los resultados de las pruebas de accesibilidad conforme a lo indicado en el capítulo de diseño.

8.3.2.1 Comprobación de contraste

Como se ha mencionado previamente, se verificará el contraste de los colores en fondos, elementos y textos para cumplir con el Nivel Triple-A de las Directrices de Accesibilidad para el Contenido Web 2.1 (WCAG). En el apéndice, se incluyen capturas de la herramienta utilizada, sin especificar a qué corresponde cada color ya que varios elementos comparten los mismos.

8.3.2.2 Checklist del WCAG

A continuación, se completa el checklist de WCAG que se encuentra en la sección 6.5.3.3.

Puntos de verificación Prioridad 1:

En general (Prioridad 1)	Sí	No	N/A
1.1 Proporcione un texto equivalente para todo elemento no textual (Por ejemplo, a través de "alt", "longdesc" o en el contenido del elemento). *Esto incluye:* imágenes, representaciones gráficas del texto, mapas de imagen, animaciones (Por ejemplo, *GIFs* animados), "applets" y objetos programados, "ascii art", marcos, scripts, imágenes usadas como viñetas en las listas, espaciadores, botones gráficos, sonidos (ejecutados con o sin interacción del usuario), archivos exclusivamente auditivos, banda sonora del vídeo y vídeos.	**X**		
2.1 Asegúrese de que toda la información transmitida a través de los colores también esté disponible sin color, por ejemplo mediante el contexto o por marcadores.	**X**		
4.1 Identifique claramente los cambios en el idioma del texto del documento y en cualquier texto equivalente (por ejemplo, leyendas).			**X**
6.1 Organice el documento de forma que pueda ser leído sin hoja de estilo. Por ejemplo, cuando un documento HTML es interpretado sin asociarlo a una hoja de estilo, tiene que ser posible leerlo.		**X**	
6.2 Asegúrese de que los equivalentes de un contenido dinámico son actualizados cuando cambia el contenido dinámico.	**X**		
7.1 Hasta que las aplicaciones de usuario permitan controlarlo, evite provocar destellos en la pantalla.	**X**		
14.1 Utilice el lenguaje apropiado más claro y simple para el contenido de un sitio.	**X**		
Y si utiliza imágenes y mapas de imagen (Prioridad 1)	**Sí**	**No**	**N/A**
1.2 Proporcione vínculos redundantes en formato texto para cada zona activa de un mapa de imagen del servidor.			**X**
9.1 Proporcione mapas de imagen controlados por el cliente en lugar de por el servidor, excepto donde las zonas sensibles no puedan ser definidas con una forma geométrica.			**X**
Y si utiliza tablas (Prioridad 1)	**Sí**	**No**	**N/A**
5.1 En las tablas de datos, identifique los encabezamientos de fila y columna.	**X**		
5.2 Para las tablas de datos que tienen dos o más niveles lógicos de encabezamientos de fila o columna, utilice marcadores para asociar las celdas de encabezamiento y las celdas de datos.			**X**
Y si utiliza marcos ("frames") (Prioridad 1)	**Sí**	**No**	**N/A**
12.1 Titule cada marco para facilitar su identificación y navegación.			**X**
Y si utiliza "applets" y "scripts" (Prioridad 1)	**Sí**	**No**	**N/A**
6.3 Asegure que las páginas sigan siendo utilizables cuando se desconecten o no se soporten los scripts, *applets* u otros objetos programados. Si esto no es posible, proporcione información equivalente en una página alternativa accesible.	**X**		
Y si utiliza multimedia (Prioridad 1)	**Sí**	**No**	**N/A**
1.3 Hasta que las aplicaciones de usuario puedan leer en voz alta automáticamente el texto equivalente de la banda visual, proporcione una descripción auditiva de la información importante de la banda visual de una presentación multimedia.			**X**

1.4 Para toda presentación multimedia dependiente del tiempo (por ejemplo, una película o animación) sincronice alternativas equivalentes (por ejemplo, subtítulos o descripciones de la banda visual) con la presentación.			**X**
Y si todo lo demás falla (Prioridad 1)	**Sí**	**No**	**N/A**
11.4 Si, después de los mayores esfuerzos, no puede crear una página accesible, proporcione un vínculo a una página alternativa que use tecnologías *W3C*, sea accesible, tenga información (o funcionalidad) equivalente y sea actualizada tan a menudo como la página (original) inaccesible.		**X**	

Puntos de verificación Prioridad 2:

En general (Prioridad 2)	**Sí**	**No**	**N/A**
2.2 Asegúrese de que las combinaciones de los colores de fondo y primer plano tengan el suficiente contraste para que sean percibidas por personas con deficiencias de percepción de color o en pantallas en blanco y negro [Prioridad 2 para las imágenes. Prioridad 3 para los textos].	**X**		
3.1 Cuando exista un marcador apropiado, use marcadores en vez de imágenes para transmitir la información.	**X**		
3.2 Cree documentos que estén validados por las gramáticas formales publicadas.	**X**		
3.3 Utilice hojas de estilo para controlar la maquetación y la presentación.	**X**		
3.4 Utilice unidades relativas en lugar de absolutas al especificar los valores en los atributos de los marcadores de lenguaje y en los valores de las propiedades de las hojas de estilo.	**X**		
3.5 Utilice elementos de encabezado para transmitir la estructura lógica y utilícelos de acuerdo con la especificación.	**X**		
3.6 Marque correctamente las listas y los ítems de las listas.	**X**		
3.7 Marque las citas. No utilice el marcador de citas para efectos de formato tales como sangrías.			**X**
6.5 Asegúrese de que los contenidos dinámicos son accesibles o proporcione una página o presentación alternativa.	**X**		
7.2 Hasta que las aplicaciones de usuario permitan controlarlo, evite el parpadeo del contenido (por ejemplo, cambio de presentación en periodos regulares, así como el encendido y apagado).	**X**		
7.4 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener las actualizaciones, no cree páginas que se actualicen automáticamente de forma periódica.	**X**		
7.5 Hasta que las aplicaciones de usuario proporcionen la posibilidad de detener el redireccionamiento automático, no utilice marcadores para redirigir las páginas automáticamente. En su lugar, configure el servidor para que ejecute esta posibilidad.	**X**		
10.1 Hasta que las aplicaciones de usuario permitan desconectar la apertura de nuevas ventanas, no provoque apariciones repentinas de nuevas ventanas y no cambie la ventana actual sin informar al usuario.	**X**		

11.1 Utilice tecnologías *W3C* cuando estén disponibles y sean apropiadas para la tarea y use las últimas versiones que sean soportadas.	**X**		
11.2 Evite características desaconsejadas por las tecnologías *W3C*.	**X**		
12.3 Divida los bloques largos de información en grupos más manejables cuando sea natural y apropiado.	**X**		
13.1 Identifique claramente el objetivo de cada vínculo.	**X**		
13.2 Proporcione metadatos para añadir información semántica a las páginas y sitios.			**X**
13.3 Proporcione información sobre la maquetación general de un sitio (por ejemplo, mapa del sitio o tabla de contenidos).			**X**
13.4 Utilice los mecanismos de navegación de forma coherente.	**X**		
Y si utiliza tablas (Prioridad 2)	**Sí**	**No**	**N/A**
5.3 No utilice tablas para maquetar, a menos que la tabla tenga sentido cuando se alinee. Por otro lado, si la tabla no tiene sentido, proporcione una alternativa equivalente (la cual debe ser una versión alineada).	**X**		
5.4 Si se utiliza una tabla para maquetar, no utilice marcadores estructurales para realizar un efecto visual de formato.			**X**
Y si utiliza marcos ("frames") (Prioridad 2)	**Sí**	**No**	**N/A**
12.2 Describa el propósito de los marcos y cómo éstos se relacionan entre sí, si no resulta obvio solamente con el título del marco.			**X**
Y si utiliza formularios (Prioridad 2)	**Sí**	**No**	**N/A**
10.2 Hasta que las aplicaciones de usuario soporten explícitamente la asociación entre control de formulario y etiqueta, para todos los controles de formularios con etiquetas asociadas implícitamente, asegúrese de que la etiqueta está colocada adecuadamente.	**X**		
12.4 Asocie explícitamente las etiquetas con sus controles.			**X**
Y si utiliza "applets" y "scripts" (Prioridad 2)	**Sí**	**No**	**N/A**
6.4 Para los *scripts* y *applets*, asegúrese de que los manejadores de eventos sean independientes del dispositivo de entrada.			**X**
7.3 Hasta que las aplicaciones de usuario permitan congelar el movimiento de los contenidos, evite los movimientos en las páginas.	**X**		
8.1 Haga los elementos de programación, tales como *scripts* y *applets*, directamente accesibles o compatibles con las ayudas técnicas [Prioridad 1 si la funcionalidad es importante y no se presenta en otro lugar; de otra manera, Prioridad 2].			**X**
9.2 Asegúrese de que cualquier elemento que tiene su propia interfaz pueda manejarse de forma independiente del dispositivo.	**X**		
9.3 Para los "scripts", especifique manejadores de evento lógicos mejor que manejadores de eventos dependientes de dispositivos.			**X**

Puntos de verificación Prioridad 3:

En general (Prioridad 3)	**Sí**	**No**	**N/A**
4.2 Especifique la expansión de cada abreviatura o acrónimo cuando aparezcan por primera vez en el documento.			**X**
4.3 Identifique el idioma principal de un documento.	**X**		

9.4 Cree un orden lógico para navegar con el tabulador a través de vínculos, controles de formulario y objetos.	**X**		
9.5 Proporcione atajos de teclado para los vínculos más importantes (incluidos los de los mapas de imagen de cliente), los controles de formulario y los grupos de controles de formulario.	**X**		
10.5 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten claramente los vínculos contiguos, incluya caracteres imprimibles (rodeados de espacios), que no sirvan como vínculo, entre los vínculos contiguos.			**X**
11.3 Proporcione la información de modo que los usuarios puedan recibir los documentos según sus preferencias (por ejemplo, idioma, tipo de contenido, etc.).		**X**	
13.5 Proporcione barras de navegación para destacar y dar acceso al mecanismo de navegación.		**X**	
13.6 Agrupe los vínculos relacionados, identifique el grupo (para las aplicaciones de usuario) y, hasta que las aplicaciones de usuario lo hagan, proporcione una manera de evitar el grupo.			**X**
13.7 Si proporciona funciones de búsqueda, permita diferentes tipos de búsquedas para diversos niveles de habilidad y preferencias.		**X**	
13.8 Localice la información destacada al principio de los encabezamientos, párrafos, listas, etc.			**X**
13.9 Proporcione información sobre las colecciones de documentos (por ejemplo, los documentos que comprendan múltiples páginas).			**X**
13.10 Proporcione un medio para saltar sobre un *ASCII art* de varias líneas.			**X**
14.2 Complemente el texto con presentaciones gráficas o auditivas cuando ello facilite la comprensión de la página.	**X**		
14.3 Cree un estilo de presentación que sea coherente para todas las páginas.	**X**		
Y si utiliza imágenes o mapas de imagen (Prioridad 3)	**Sí**	**No**	**N/A**
1.5 Hasta que las aplicaciones de usuario interpreten el texto equivalente para los vínculos de los mapas de imagen de cliente, proporcione vínculos de texto redundantes para cada zona activa del mapa de imagen de cliente.			**X**
Y si utiliza tablas (Prioridad 3)	**Sí**	**No**	**N/A**
5.5 Proporcione resúmenes de las tablas.		**X**	
5.6 Proporcione abreviaturas para las etiquetas de encabezamiento.			**X**
10.3 Hasta que las aplicaciones de usuario (incluidas las ayudas técnicas) interpreten correctamente los textos contiguos, proporcione un texto lineal alternativo (en la página actual o en alguna otra) para *todas* las tablas que maquetan texto en paralelo, en columnas de palabras.		**X**	
Y si utiliza formularios (Prioridad 3)	**Sí**	**No**	**N/A**
10.4 Hasta que las aplicaciones de usuario manejen correctamente los controles vacíos, incluya caracteres por defecto en los cuadros de edición y áreas de texto.			**X**

8.3.3 Conclusiones sobre las pruebas de usabilidad y accesibilidad

En base a los resultados presentados, podemos concluir que la aplicación es suficientemente usable y accesible para distintos tipos de usuarios. No obstante, es necesario revisar los errores de accesibilidad identificados en la checklist, ya que podrían ocasionar problemas en el futuro. Además, se han recibido las siguientes sugerencias:

- Sería bueno añadir más tipos de gráficos y estadísticas en la pestaña correspondiente.
- Se podría indicar la relación de los errores con el gráfico y la tabla de movimientos.
- Sería bueno agregar algún tipo de ayuda para explicar la utilidad de la aplicación y cómo realizar las tareas más comunes.

Se ha implementado el punto relacionado con la ayuda y se ha justificado la no implementación del resto en sus respectivas secciones. Sin embargo, estas sugerencias se considerarán para futuras ampliaciones.

8.4 Pruebas de Rendimiento

Durante la ejecución de las pruebas de rendimiento diseñadas, se determinó que no era necesario analizar múltiples partidas, lo cual complicaría el proceso. En cambio, se seleccionó una partida con suficientes movimientos para garantizar una muestra significativa, pero no tantos como para aumentar innecesariamente el tiempo de análisis.

Inicialmente, se fijó el número de nodos a explorar en 10.000 y se registraron el tiempo y la ventaja obtenidos al analizar en incrementos de 10.000 nodos hasta alcanzar 50.000. Una vez obtenidas estas ventajas, se calculó la distancia euclídea entre la lista proporcionada por Lichess y cada una de estas muestras, utilizando la fórmula:

$$d(A, B) = \sqrt{\sum_{i=1}^{n} (A[i] - B[i])^2}$$

Figura 8.2 Fórmula de la distancia euclídea

donde $A[i]$ y $B[i]$ son los elementos en la posición i de las listas A (ventaja de Lichess) y B (ventaja del sistema), respectivamente.

A continuación, se presentan dos gráficos: uno que muestra el aumento en el tiempo de análisis y otro que muestra la similitud con los resultados de Lichess, ambos en función de los nodos explorados.

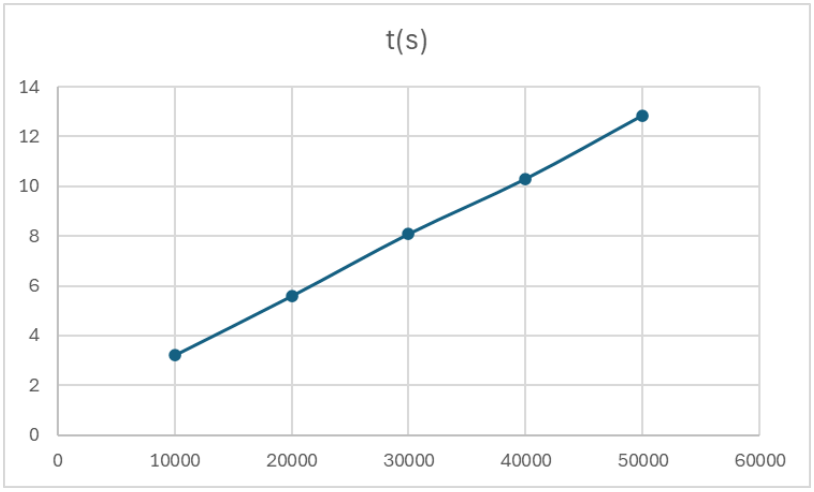


Figura 8.3 Tiempo (en segundos) en función a los nodos explorados

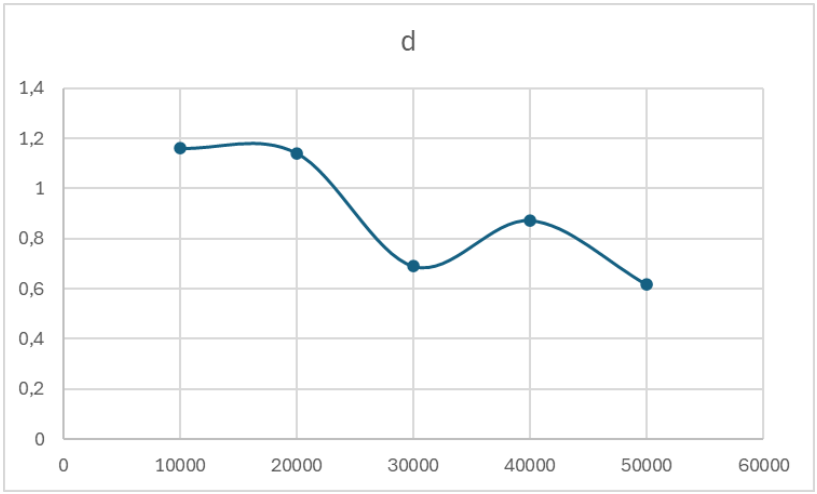


Figura 8.4 Distancia euclídea en función a los nodos explorados

Se puede observar que el tiempo aumenta de manera lineal, incrementándose en 2 segundos, y la ventaja calculada se asemeja más a la proporcionada por Lichess a medida que se incrementa el número de nodos a explorar.

En base a esto, se pueden extraer las siguientes conclusiones: mientras que el tiempo adicional no influye tanto en el análisis de una partida, al analizar varias partidas el tiempo total aumenta considerablemente. Además, aunque nos aproximamos más a los valores del sitio web al aumentar el número de nodos (aunque no de manera directamente proporcional, ya que los valores obtenidos con 40000 nodos difieren más que los obtenidos con 30000), la discrepancia es mínima y, aunque las cifras no coincidan exactamente, la relación entre ellas suele ser prácticamente la misma, variando solo en escala.

Por todas estas razones, se mantendrá el número de nodos en 10000, lo que proporciona un análisis rápido y prácticamente idéntico al proporcionado por Lichess.

Capítulo 9. Manuales del Sistema

9.1 Manual de Instalación

Para instalar el sistema, basta con ejecutar el instalador disponible en el directorio raíz del archivo adjunto. Este está diseñado específicamente para Windows y la aplicación se abrirá automáticamente al completarse la instalación.

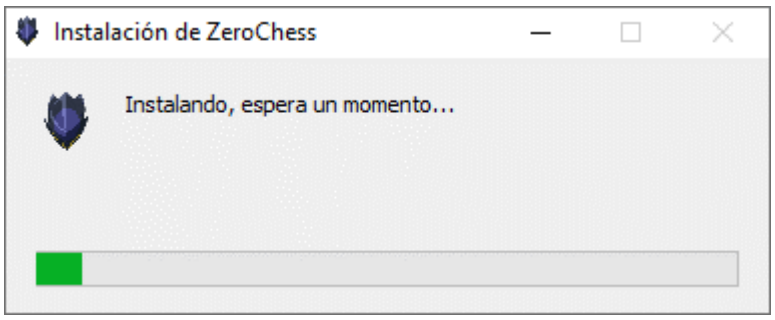


Figura 9.1 Proceso de instalación

Para crear un ejecutable para otra plataforma desde un equipo con el sistema operativo correspondiente, es necesario seguir los siguientes pasos:

- Se debe instalar Node.js, utilizando la versión 20.15.0 (LTS).
- Luego, es necesario abrir una terminal o el símbolo del sistema y navegar hasta la carpeta "./zero-chess/src", que está ubicada en los archivos adjuntos.
- Posteriormente, se debe ejecutar el comando "npm install". Una vez que todas las dependencias estén instaladas, ejecutar el comando "npm run package".
- El instalador generado estará disponible en el directorio "./zero-chess/release/build".

9.2 Manual de Ejecución

Para iniciar el sistema, basta con ejecutar la aplicación una vez instalada. Para iniciar en modo de producción, después de haber seguido los pasos mencionados en el manual de instalación (incluyendo la instalación de dependencias), se debe ejecutar el comando "npm start".

9.3 Manual de Usuario

En este apartado se describen las principales funciones que los usuarios pueden realizar en la aplicación.

9.3.1 Importar partidas

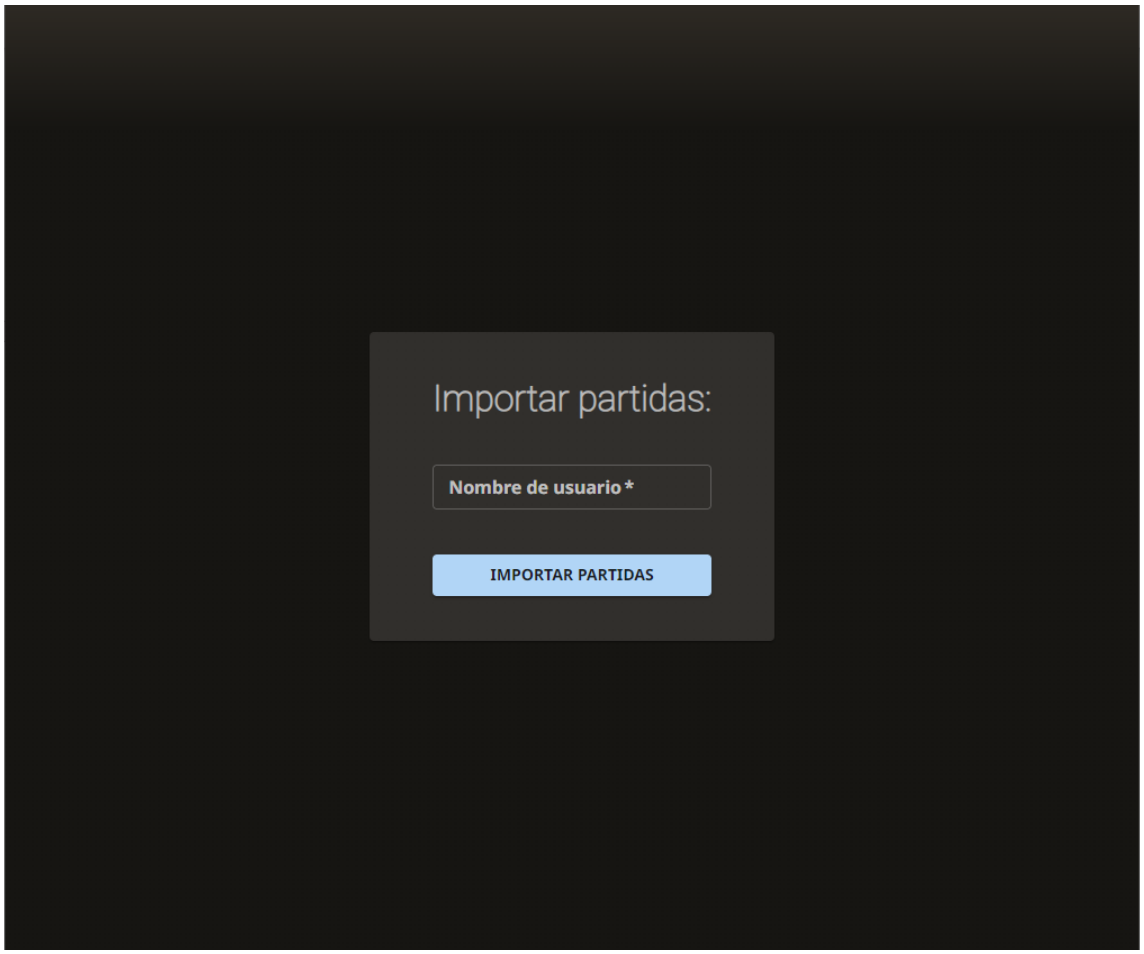


Figura 9.2 Ventana de inicio (manual de usuario)

Al iniciar la aplicación, la primera pantalla que aparece es la de importación de partidas. Para importar las partidas de un usuario de Lichess, se debe ingresar su nombre y luego pulsar el botón correspondiente en el formulario.

Es importante destacar que el usuario debe haber jugado partidas en Lichess y al menos una de estas debe ser compatible con la aplicación. Las partidas incompatibles, como las pertenecientes a la variante “From Position” o aquellas en las que no se ha realizado ningún movimiento, no serán mostradas.

Una vez enviado el formulario, comenzará la descarga de las partidas. Si el nombre de usuario ya fue ingresado anteriormente, las partidas ya habrán sido descargadas y se mostrarán de inmediato.

9.3.2 Navegar por las partidas

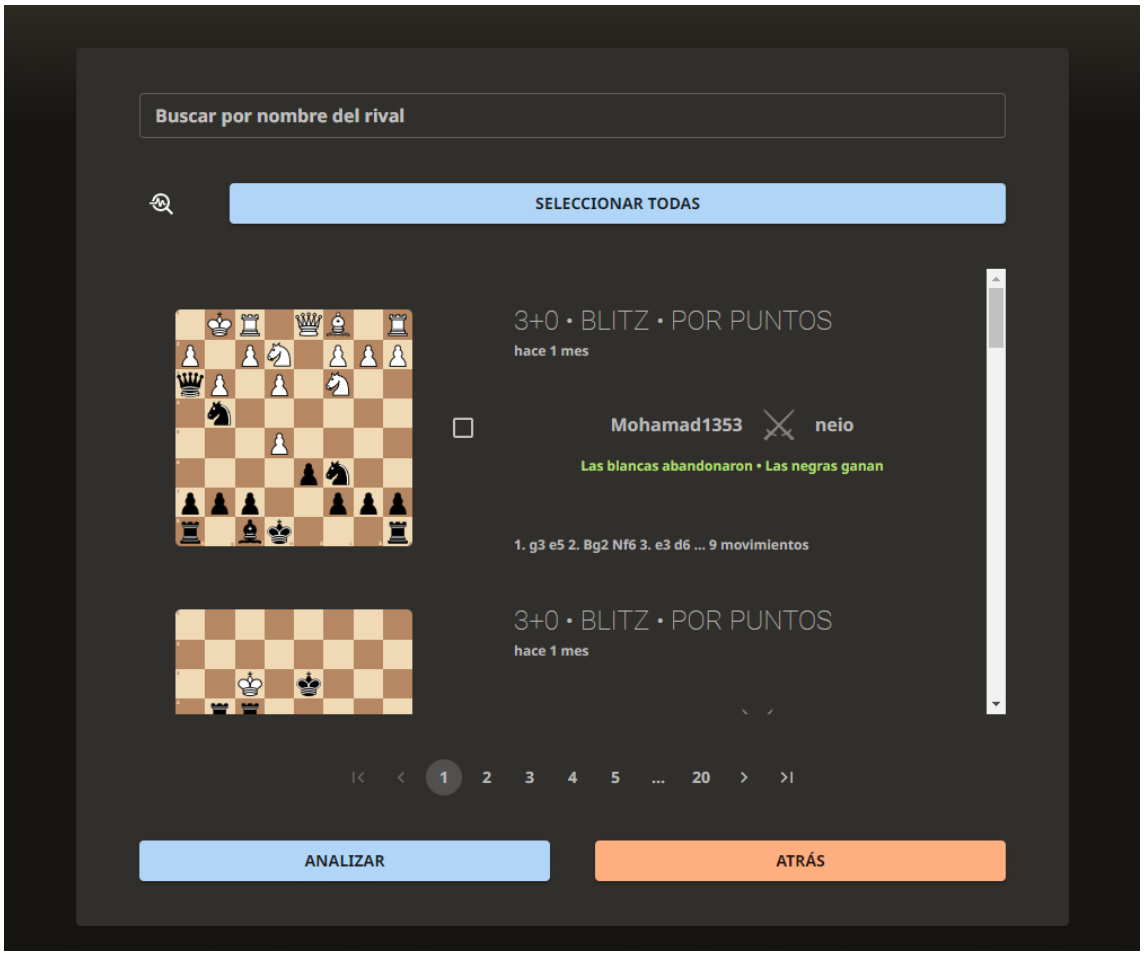


Figura 9.3 Ventana con lista de partidas (manual de usuario)

Una vez importadas, es posible navegar por las partidas jugadas por el usuario introducido. Estas están ordenadas cronológicamente y se cuenta con varios elementos que facilitan la navegación:

- A la derecha de la lista de partidas, se encuentra una barra de desplazamiento que permite navegar por la página actual, donde se muestran hasta diez partidas.
- En la parte superior de la pantalla, se encuentra un campo de búsqueda donde se puede introducir parcial o completamente el nombre de un usuario. Esto permite filtrar las partidas mostradas para incluir únicamente aquellas jugadas contra dicho usuario como rival.
- En la parte inferior de la lista, se puede seleccionar la página de partidas que se desea visualizar.

Cada partida muestra diversos datos como la configuración, la fecha en que se jugó, los participantes, el estado actual de la partida y los movimientos realizados durante esta.

9.3.3 Analizar partidas

Una vez encontradas las partidas que se desean analizar, se debe hacer clic sobre cada una de ellas para seleccionarlas o deseleccionarlas. Para seleccionar todas las partidas importadas, se puede utilizar el botón "Seleccionar todas", disponible en la parte superior de la lista.

Una vez seleccionadas las partidas deseadas, se puede solicitar su análisis pulsando el botón "Analizar", ubicado en la esquina inferior izquierda. Las partidas que tienen disponible el análisis estarán marcadas con un tick.

Figura 9.4 Partida analizada (manual de usuario)

9.3.4 Acceder al análisis

Figura 9.5 Ventana de análisis (manual de usuario)

Cuando se selecciona una partida que tiene disponible el análisis, se accede a él. En la esquina superior derecha, se muestra una tabla con todos los movimientos realizados, comenzando por el jugador blanco. Junto a cada movimiento, hay un número que indica la ventaja en peones a favor del jugador blanco en ese momento de la partida. Si este número es negativo, indica una ventaja para el jugador negro.

Justo debajo de la tabla de movimientos, se encuentran los controles para ejecutar los movimientos en el tablero. El último movimiento jugado se resalta en azul. Para retroceder en la partida hasta el inicio o a un movimiento anterior, se pueden utilizar los botones a la izquierda, mientras que para avanzar en la partida se utilizan los botones ubicados a la derecha.

En la esquina inferior izquierda, hay un gráfico que representa de manera clara la ventaja mencionada durante el transcurso de la partida.

En la esquina inferior derecha, se muestra el color de cada jugador y los tipos de errores cometidos, ordenados de menos a más graves de arriba hacia abajo. También se proporciona la precisión media de los movimientos realizados por cada jugador.

9.3.5 Acceder a las estadísticas

En la lista de partidas, en la esquina superior izquierda, hay un botón en forma de lupa con un gráfico. Al hacer clic sobre él, se accede a las estadísticas del usuario.

Figura 9.6 Ventana de estadísticas (manual de usuario)

Las estadísticas están organizadas en dos secciones: una relativa al número de victorias y otra a los errores cometidos. Para alternar entre ellas, simplemente se puede esperar un momento sin mover el ratón sobre ellas o acercarlo a los laterales donde aparecerán botones con ese propósito.

Es importante mencionar que, si el usuario introducido no ha jugado muchas partidas o no se han analizado las suficientes, algunas estadísticas podrían no mostrarse debido a la falta de datos.

9.4 Manual del Programador

Este manual abarcará todos los aspectos que puedan facilitar la ampliación, modificación o comprensión de la construcción del sistema.

9.4.1 Incluir nuevas funciones de análisis

En esta sección, se detallará paso a paso cómo añadir nuevas funciones de análisis al sistema desde la perspectiva de un desarrollador.

Bajo el directorio “src\main\services\engine” del proyecto, se encuentran disponibles los ficheros engine.ts (en la subcarpeta “helpers”) y engineService.ts.

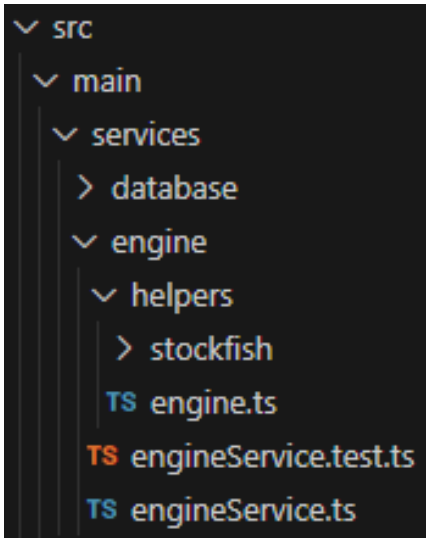


Figura 9.7 Árbol de archivos del motor

Para definir una nueva función de interacción con el motor en el fichero engine.ts, se puede hacer uso de la función “send” definida en la interfaz “Engine”. Esta función recibe el mensaje a enviar al motor y dos funciones callback: una que se ejecuta al finalizar la respuesta del motor y otra que se llama a medida que el motor envía mensajes. Ambas funciones reciben dichas respuestas como parámetro.

```
return new Promise<number | string>((resolve) => {
  engine.send(
    `go nodes ${nodes}`,
    function onDone() {
      if (invert) {
        if (typeof latestScore === 'number') {
          latestScore = -latestScore;
        } else {
          latestScore = `#${-parseInt(latestScore.substring(1), 10)}`;
        }
      }

      resolve(latestScore);
    },
    function onStream(data) {
      const cpRegex = /score cp (-?\d+)/;
      const mateRegex = /score mate (-?\d+)/;

      let match = data.match(mateRegex);
      if (match && match[1]) {
        latestScore = `#${match[1]}`;
      } else {
        match = data.match(cpRegex);
        if (match && match[1]) {
          latestScore = parseFloat(match[1]);
        }
      }
    },
  );
});
```

Figura 9.8 Ejemplo de uso de la función "send"

Para definir una nueva función de procesamiento en el fichero engineService.ts, se puede hacer uso de la función "getGameCentipawns", la cual recibe un array con los movimientos de una partida y devuelve otro array con la ventaja en centipawns en cada momento.

```
const centipawnsPerNode: (number | string)[] = await getGameCentipawns(moves);
```

Figura 9.9 Ejemplo de uso de la función "getGameCentipawns"

Estas nuevas funciones deberán ser llamadas en la función "analyseGame", definida en el fichero GameList.tsx, disponible bajo el directorio "src\renderer\GameList". El resultado de estas funciones deberá añadirse al objeto devuelto por esta.

```
async function analyseGame(game: GameDecorator) {
  const moves = game.getGameMoves();

  const accuracy = await getGameAccuracy(moves);
  const advantage = await getGameAdvantage(moves);
  const mistakes = calculateErrors(advantage, 1, 1.5);
  const inaccuracies = calculateErrors(advantage, 0.5, 1);
  const blunders = calculateErrors(advantage, 1.5, Infinity);

  return { advantage, accuracy, mistakes, inaccuracies, blunders };
}
```

Figura 9.10 Implementación de la función "analyseGame"

De esta manera, las funciones se ejecutarán cuando se analice una partida y su resultado se almacenará en la base de datos sin necesidad de alterar la lógica existente. Se puede acceder a los resultados a través de las partidas, las cuales son accesibles desde cualquier componente.

9.4.2 Incluir nuevas estadísticas

En esta sección, se detallará paso a paso cómo añadir nuevas estadísticas al sistema desde la perspectiva de un desarrollador.

Bajo el directorio "src\renderer\GameStatistics" del proyecto, se encuentran disponibles los ficheros GameStatistics.tsx, StatsBarChart.tsx y StatsPieChart.tsx.

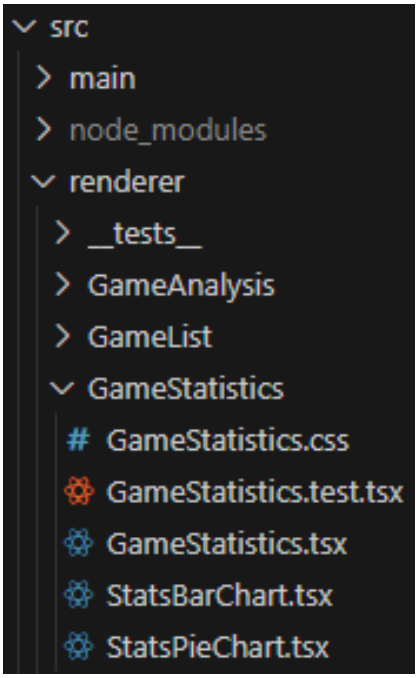


Figura 9.11 Árbol de archivos de las estadísticas

Para facilitar el acceso de una nueva estadística a los gráficos, debemos definir los atributos necesarios en la constante "stats" dentro del componente "GameStatistics".

```
const stats = {
  draws: 0,
  losses: 0,
  blunders: 0,
  mistakes: 0,
  whiteWins: 0,
  blackWins: 0,
  whiteErrors: 0,
  blackErrors: 0,
  inaccuracies: 0,
  winsAgainstLowerRating: 0,
  winsAgainstHigherRating: 0,
  errorsAgainstLowerRating: 0,
  errorsAgainstHigherRating: 0,
  modeWins: new Map<string, number>(),
  modeErrors: new Map<string, number>(),
};
```

Figura 9.12 Definición de la constante "stats"

El cálculo de la nueva estadística debería incluirse en la función “updateGameStats” o alguna de las funciones que esta llama, las cuales tienen acceso a diferentes aspectos de cada partida.

```
const updateStats = (gameDecorator: GameDecorator) => {
  const game = gameDecorator.getGame();
  const playerSide = gameDecorator.getSide(username);
  const opponentSide = gameDecorator.getOpponentSide(username);

  const playerRating = game.players[playerSide].rating;
  const opponentRating = game.players[opponentSide].rating;

  if (game.analysis) {
    updateMistakes(game, playerSide, playerRating, opponentRating);
  }
  updateResults(game, playerSide, playerRating, opponentRating);
};
```

Figura 9.13 Implementación de la función "updateStats"

Una vez definida y calculada la nueva estadística, basta con crear una sección dentro del componente “Carousel” y mostrarla utilizando, por ejemplo, los gráficos “StatsBarChart” y “StatsPieChart”, que son capaces de manejar valores vacíos.

```
<Carousel
  swipe={false}
  interval={8000}
  animation="slide"
  stopAutoPlayOnHover
  className="carousel"
>
  <Box>
    <h2>Victorias</h2>
    <Stack spacing={2} direction="row" justifyContent="space-evenly">
      <StatsPieChart
        title="V/T/D"
        data={[
          {
            value: stats.whiteWins + stats.blackWins,
            color: '#C1E1C1',
            label: 'Victorias',
          },
          { value: stats.draws, color: '#A7C7E7', label: 'Tablas' },
          {
            value: stats.losses,
            color: '#FAA0A0',
            label: 'Derrotas',
          },
        ]}
      />
```

Figura 9.14 Porción de una sección del carrusel

Capítulo 10. Conclusiones y Ampliaciones

En este capítulo, se describirán las conclusiones y posibles ampliaciones del proyecto.

10.1 Conclusiones

Con este proyecto se ha logrado desarrollar una alternativa real a otros sistemas de análisis de partidas de ajedrez. Las metas propuestas se han alcanzado con la creación de una aplicación multiplataforma que permite analizar partidas sin depender de un servidor.

El resultado se presenta con una interfaz simple e intuitiva, accesible para cualquier usuario, independientemente de su nivel de conocimiento en la disciplina. Además, se facilita la incorporación de nuevas métricas y estadísticas en el futuro.

Este proyecto ha representado una oportunidad para adquirir conocimientos técnicos tanto en ajedrez como en el análisis de partidas, especialmente útiles en entornos competitivos. Además, he aprendido el proceso completo de desarrollo de aplicaciones de escritorio utilizando Electron, una tecnología hasta ahora desconocida para mí, pero cada vez más influyente en sistemas reales.

A pesar de los desafíos encontrados, como la escasez de referencias sobre la integración programática de binarios, las restricciones del framework para llevarla a cabo, y mi falta de experiencia previa en el manejo de datos a gran escala, estoy seguro de que estas experiencias serán valiosas en el futuro, incluso en contextos no relacionados con este proyecto.

Este proyecto está ideado para seguir evolucionando y adaptándose a las necesidades individuales a través de contribuciones externas, con el objetivo de destacar sobre las plataformas limitadas y conocidas en la actualidad.

10.2 Ampliaciones

En este apartado se describirán las labores de ampliación previstas en el sistema, las cuales no se han realizado debido a razones justificadas o por falta de tiempo.

10.2.1 Importación de partidas en formato PGN

El sistema actual permite importar partidas únicamente descargándolas desde la API de Lichess, y no se ha considerado la descarga desde otras APIs, ya que no son oficiales ni están mantenidas, a diferencia de la mencionada.

Sin embargo, se podría implementar la importación a través de archivos en notación portátil de partida (.PGN), un formato reconocido por la mayoría de los programas de ajedrez para ordenador.

10.2.2 Más filtros en la búsqueda

El sistema actual solo permite filtrar partidas mediante el nombre de usuario del rival. Inicialmente, no se implementaron más opciones porque muchas de ellas son exclusivas de los sitios donde se jugaron las partidas, lo que dificulta su extensibilidad.

No obstante, se podrían identificar términos comunes a todos estos sistemas para facilitar la búsqueda de partidas.

10.2.3 Errores reflejados en el gráfico y la tabla de movimientos

Como sugirió uno de los usuarios durante las pruebas de accesibilidad, se podrían marcar los diferentes tipos de errores cometidos por ambos jugadores en el gráfico o la tabla de movimientos de la ventana de análisis.

Inicialmente, esto no se implementó debido a las diferencias en el cálculo de errores con respecto al sistema de referencia (Lichess), lo que resultaba en una cantidad considerablemente elevada de estos. Sin embargo, afinando este proceso, sería posible mostrar los errores en la interfaz sin saturarla.

10.2.4 Opciones de análisis

El sistema fue diseñado para ser fácil de usar, limitándose a un solo tipo de análisis para evitar confundir a usuarios inexpertos y mantener la coherencia.

Sin embargo, si se destinara el sistema a un público experto, podrían ofrecerse más opciones de análisis, como la capacidad de elegir el número de nodos a explorar, la profundidad de análisis (aunque esta última se desaconseja), o la red neuronal utilizada por el motor.

10.2.5 Más tipos de estadísticas

En el sistema se priorizó la facilidad de añadir nuevas estadísticas sobre la complejidad y la cantidad desde el principio. Aunque actualmente no hay muchas, esta decisión permitirá incorporar fácilmente nuevos tipos de estadísticas en el futuro modificando un único archivo cuando haya más tiempo disponible.

Capítulo 11. Planificación del Proyecto y Presupuesto finales

En este capítulo se abordarán la planificación y los presupuestos finales del proyecto, además de los cambios ocurridos con respecto a la planificación inicial presentada en el capítulo 4.

11.1 Planificación Final

El horario se diseñó con la intención de poder conciliar la ejecución del proyecto con los estudios universitarios y las prácticas en la empresa, sin que ninguno de estos factores afectara a la planificación inicial. No obstante, diversos imprevistos surgidos durante el desarrollo del proyecto provocaron retrasos en ciertas tareas respecto a los plazos acordados inicialmente, tal como se detalla en la planificación final:

Nombre de tarea	Duración	Comienzo	Fin
Documentación inicial	**3,88 días**	**lun 29/01/24**	**dom 11/02/24**
Reunión de arranque	2 hrs	lun 29/01/24	lun 29/01/24
Planteamiento de la motivación y los objetivos	2 hrs	vie 02/02/24	vie 02/02/24
Estudio de la situación actual	6 hrs	sáb 03/02/24	sáb 03/02/24
Estudio de los conceptos, herramientas y tecnologías a usar	6 hrs	dom 04/02/24	lun 05/02/24
Elaboración de la planificación inicial	6 hrs	vie 09/02/24	sáb 10/02/24
Elaboración del presupuesto inicial	6 hrs	sáb 10/02/24	dom 11/02/24
Análisis	**3 días**	**lun 12/02/24**	**sáb 24/02/24**
Determinación del alcance del sistema	2 hrs	lun 12/02/24	lun 12/02/24
Elaboración de requisitos y casos de uso	8 hrs	vie 16/02/24	sáb 17/02/24
Definición de clases	8 hrs	dom 18/02/24	vie 23/02/24
Diseño de interfaces de usuario	4 hrs	sáb 24/02/24	sáb 24/02/24
Especificación del plan de pruebas	2 hrs	sáb 24/02/24	sáb 24/02/24
Diseño	**2,5 días**	**dom 25/02/24**	**lun 04/03/24**
Definición de la arquitectura del sistema y elaboración de diagramas	12 hrs	dom 25/02/24	sáb 02/03/24
Diseño de la base de datos	2 hrs	sáb 02/03/24	sáb 02/03/24
Especificación técnica del plan de pruebas	6 hrs	dom 03/03/24	lun 04/03/24
Implementación	**24,63 días**	**vie 08/03/24**	**sáb 08/06/24**
Elaboración de la base del programa	2 hrs	vie 08/03/24	vie 08/03/24
Front-end	**5,88 días**	**sáb 09/03/24**	**sáb 30/03/24**
Desarrollo de la página de inicio	8 hrs	sáb 09/03/24	dom 10/03/24
Desarrollo de la página con la lista de partidas	18 hrs	dom 10/03/24	lun 18/03/24
Desarrollo de la página de análisis	16 hrs	vie 22/03/24	vie 29/03/24
Desarrollo de la página de estadísticas	4 hrs	sáb 30/03/24	sáb 30/03/24
Back-end	**17,5 días**	**sáb 30/03/24**	**sáb 08/06/24**
Desarrollo del servicio de importación de partidas	18 hrs	sáb 30/03/24	dom 07/04/24
Integración con Stockfish	74 hrs	dom 07/04/24	vie 17/05/24
Integración con la base de datos	36 hrs	vie 17/05/24	dom 02/06/24
Desarrollo del servicio de cálculo de estadísticas	11 hrs	dom 02/06/24	sáb 08/06/24
Pruebas	40 hrs	sáb 08/06/24	sáb 29/06/24
Reuniones de seguimiento	**18,88 días**	**sáb 30/03/24**	**sáb 08/06/24**
Muestra del prototipo inicial	1 hr	sáb 30/03/24	sáb 30/03/24
Muestra de la aplicación final	1 hr	sáb 08/06/24	sáb 08/06/24
Elaboración de los manuales del sistema	2 hrs	sáb 29/06/24	sáb 29/06/24
Finalización de la documentación	4 hrs	sáb 29/06/24	dom 30/06/24
Revisión de la documentación	2 hrs	dom 30/06/24	lun 01/07/24

Figura 11.1 Cronograma de planificación final

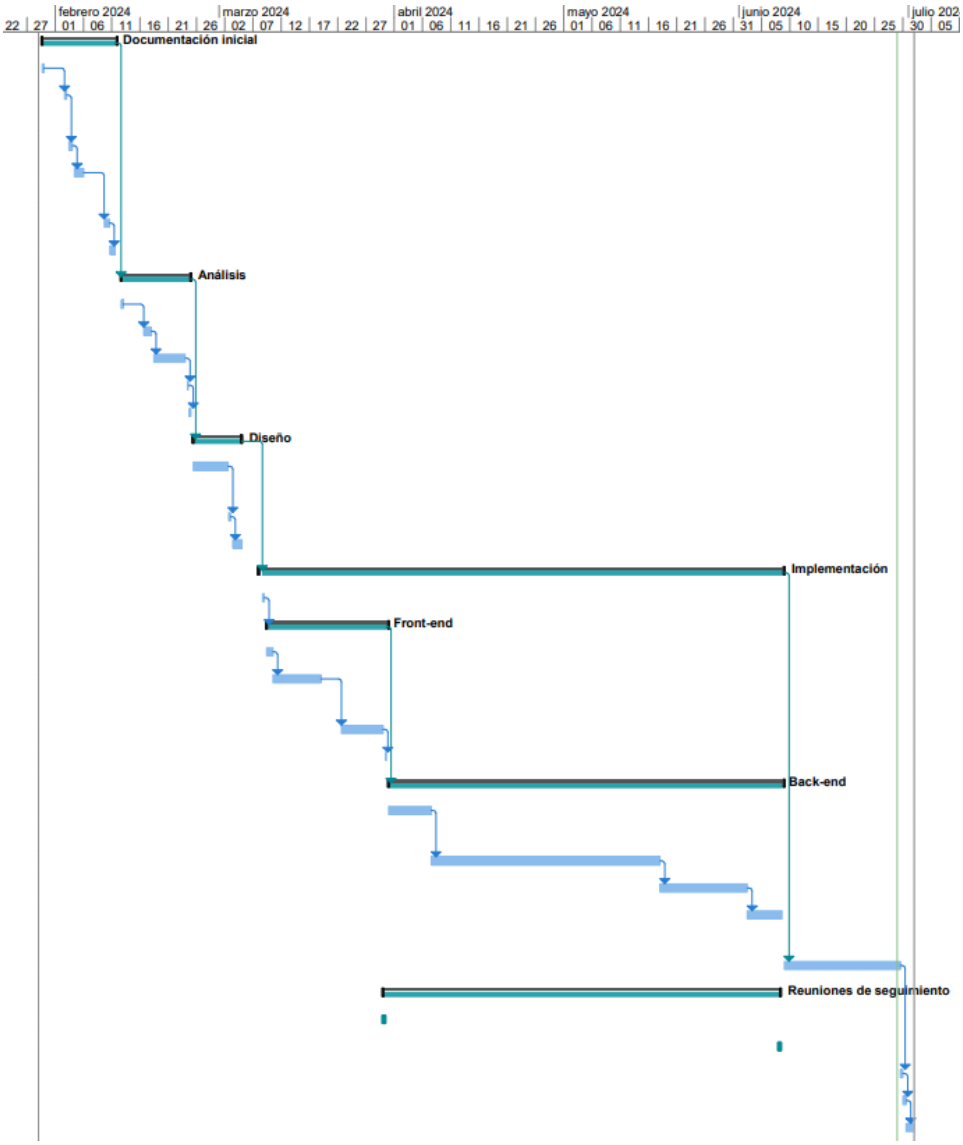


Figura 11.2 Diagrama de Gantt de la planificación final

Aunque algunas tareas tomaron menos o más tiempo del estimado inicialmente, la cantidad y el orden de estas se mantuvieron constantes. Además, la mayoría de los retrasos tienen una trazabilidad directa hacia los problemas encontrados durante la implementación, descritos en la sección 7.4.1.

El primer cambio ocurrió durante la fase de análisis: la elaboración de requisitos y casos de uso llevó menos tiempo del previsto inicialmente, mientras que la definición de clases tomó más. Este último retraso se debió a la necesidad de estudiar el framework para su correcta documentación. Sin embargo, las horas ahorradas en una tarea se utilizaron en la otra, por lo que la duración final de la tarea resumen permaneció invariable.

Además, durante la fase de diseño, la especificación técnica del plan de pruebas llevó más tiempo de lo previsto debido a una infraestimación. Tanto el análisis como el diseño contribuyeron a reducir el tiempo planeado para las tareas relacionadas con el cálculo y la presentación de estadísticas.

No obstante, los problemas encontrados durante la implementación del sistema, expuestos anteriormente, resultaron en un aumento de la duración de varias tareas relacionadas con el desarrollo del back-end. Esto afectó especialmente a aquellas relacionadas con la integración con Stockfish: el framework no facilitó esta integración, y fueron necesarias varias correcciones y operaciones adicionales que no estaban contempladas inicialmente.

El desarrollo de las pruebas tomó un poco más de lo previsto, en parte debido a la falta de experiencia en la simulación de elementos exclusivos de Electron. En contraste, la elaboración de manuales fue más rápida, ya que abordaba conceptos previamente estudiados durante el diseño e implementación.

Al momento de redactar este apartado, se está finalizando la documentación, faltando completar esta tarea y su revisión. Se estima que, una vez terminadas, se habrán utilizado alrededor de 310 horas, 20 horas más de lo inicialmente planeado, pero aún dentro del margen establecido para su finalización.

11.2 Presupuesto Final

En esta parte se llevará a cabo una revisión del presupuesto inicial. Luego, se compararán estos presupuestos con los originales y se analizarán las razones detrás de sus variaciones.

11.2.1 Presupuesto Interno

Para elaborar el presupuesto interno final, se emplearán los mismos cálculos utilizados en el presupuesto inicial, pero considerando las 310 horas finales.

Ítem	Concepto	Cantidad	Amortización	Precio Unitario (€)	Total (€)
1	Recursos Humanos				
1.1	Documentación	38	100%	13,00 €	494,00 €
1.2	Análisis	24	100%	18,00 €	432,00 €
1.3	Diseño	20	100%	30,00 €	600,00 €
1.4	Implementación	187	100%	17,00 €	3.179,00 €
1.5	Pruebas	40	100%	15,00 €	600,00 €
2	Recursos Software				
2.1	Microsoft Windows 10 Home	2	8%	145,00 €	24,17 €
2.2	Microsoft Office 2021	1	100%	0,00 €	0,00 €
2.3	Microsoft Project 2016	1	100%	0,00 €	0,00 €
2.4	Visual Studio Code	2	100%	0,00 €	0,00 €
3	Recursos Hardware				
3.1	Ordenador de sobremesa	1	8%	1800,00 €	150,00 €
3.2	Portátil	1	8%	800,00 €	66,67 €
4	Costes indirectos				
4.1	Electricidad	5	100%	55,04 €	275,22 €
4.2	Internet	5	100%	70,00 €	350,00 €
Subtotal					6.171,05 €
Beneficio (12%)					740,53 €
TOTAL					**6.911,58 €**

Tabla 11.1 Presupuesto interno final

La discrepancia en la duración real del proyecto respecto a la planificada inicialmente resultó en un sobrecoste de 381,92 €.

11.2.2 Presupuesto Simplificado (Cliente)

De manera análoga al presupuesto interno, el presupuesto simplificado final seguirá el mismo proceso descrito para la elaboración del presupuesto simplificado inicial. Este, se encuentra disponible en la siguiente tabla:

Concepto	Total (€)
Documentación del proyecto	691,57 €
Análisis y diseño del sistema	1.260,77 €
Implementación	4.151,27 €
Pruebas	807,97 €
Subtotal	6.911,58 €
IVA (21%)	1.451,43 €
TOTAL	**8.363,01 €**

Tabla 11.2 Presupuesto final del cliente

Se puede concluir que hay un sobrecoste de 462,12€ debido a la infraestimación de algunas tareas. No obstante, esta información será útil para definir de manera más precisa los recursos necesarios en proyectos futuros.

Capítulo 12. Referencias Bibliográficas

Aislamiento de contexto | Electron. (s. f.). https://www.electronjs.org/es/docs/latest/tutorial/context-isolation

Angular (framework) - Wikipedia, la enciclopedia libre. (2023, agosto 18). https://es.wikipedia.org/wiki/Angular_(framework)

Arena Chess GUI. (s. f.). http://www.playwitharena.de/

ChessBase - Wikipedia, la enciclopedia libre. (2024, enero 22). https://es.wikipedia.org/wiki/ChessBase

Chess.com - Wikipedia, la enciclopedia libre. (2024, enero 25). https://es.wikipedia.org/wiki/Chess.com

Comunicación entre procesos | Electron. (s. f.). https://www.electronjs.org/es/docs/latest/tutorial/ipc

Documentation for Visual Studio Code. (s. f.). https://code.visualstudio.com/docs

Electron (software) - Wikipedia, la enciclopedia libre. (2024, marzo 7). https://es.wikipedia.org/wiki/Electron_%28software%29

electron-builder. (s. f.). https://www.electron.build/index.html

Friedel, F. (2022, noviembre 22). *El mundo magnífico de ChessBase 17 | ChessBase*. https://es.chessbase.com/post/el-mundo-magnifico-de-chessbase-17

IndexedDB key characteristics and basic terminology - Web APIs | MDN. (s. f.). https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API/Basic_Terminology

Introducing NNUE Evaluation - Stockfish - Strong open-source chess engine. (2020, agosto 7). https://stockfishchess.org/blog/2020/introducing-nnue-evaluation/

Jasmine Documentation. (s. f.). https://jasmine.github.io/index.html

JavaScript | MDN. (s. f.). https://developer.mozilla.org/es/docs/Web/JavaScript

JavaScript - Wikipedia, la enciclopedia libre. (2024, junio 29). https://es.wikipedia.org/wiki/JavaScript

Jest · Delightful JavaScript Testing. (s. f.). https://jestjs.io/

Lichess - Wikipedia, la enciclopedia libre. (2024, junio 14). https://es.wikipedia.org/wiki/Lichess#

Lichess Accuracy metric • lichess.org. (s. f.). https://lichess.org/page/accuracy

Mocha - the fun, simple, flexible JavaScript test framework. (s. f.). https://mochajs.org/

NeDB - Database of Databases. (s. f.). https://dbdb.io/db/nedb

Newline Delimited JSON (ndjson) Format | MuleSoft Documentation. (s. f.). https://docs.mulesoft.com/dataweave/latest/dataweave-formats-ndjson

Overview - Material UI. (s. f.). https://mui.com/material-ui/getting-started/

PAe - Métrica v.3. (2001). https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html#.U7KmJUDfiSo

Quick Start | Electron. (s. f.). https://www.electronjs.org/docs/latest/tutorial/quick-start#run-the-main-process

React - Wikipedia, la enciclopedia libre. (2024, abril 11). https://es.wikipedia.org/wiki/React

React Charts - MUI X. (s. f.). Recuperado 1 de julio de 2024, de https://mui.com/x/react-charts/

Sánchez García, R. (2020, mayo 11). *Métrica V3: Qué es y cuál es la estructuración de los procesos*. https://elminimoviable.es/metodologia-metrica-v3-que-es-y-cual-es-la-estructuracion-de-los-procesos/

SQLite Home Page. (2024, abril 15). https://sqlite.org/

Testori, E. (2024, abril 16). *Newline-delimited JSON: an unstandardized standard · HyperTesto*. https://www.hypertesto.me/en/blog/jsonl-and-ndjson/

TypeScript - Wikipedia. (2024, mayo 4). https://en.wikipedia.org/wiki/TypeScript

Using Preload Scripts | Electron. (s. f.). https://www.electronjs.org/docs/latest/tutorial/tutorial-preload

Visual Studio Code - Wikipedia, la enciclopedia libre. (2024, junio 22). https://es.wikipedia.org/wiki/Visual_Studio_Code

Vue.js - Wikipedia, la enciclopedia libre. (2024, febrero 2). https://es.wikipedia.org/wiki/Vue.js

Capítulo 13. Apéndices

13.1 Glosario y Diccionario de Datos

En este apartado, se proporciona una breve descripción de todos los términos relevantes en la aplicación, organizados en orden alfabético.

- **Centipawn**: Unidad de medida utilizada en ajedrez para cuantificar el valor de una posición en términos de la ventaja posicional o material. Representa una centésima parte del valor de un peón.
- **Elo:** Sistema de puntuación utilizado para calcular el nivel relativo de habilidad de los jugadores de ajedrez y de otros juegos de habilidad.
- **FEN (Forsyth-Edwards Notation):** Notación para describir el estado de un tablero de ajedrez en un momento específico de una partida. Contiene información sobre la posición de las piezas, el turno de juego, el enroque, la posibilidad de captura al paso y el número de medio-movimientos desde la última jugada que pudo haber realizado una captura o el avance de un peón.
- **Mate:** Situación en la que el rey de un jugador está amenazado de ser capturado y no hay manera de evitarlo, lo que termina la partida.
- **NNUE (Efficiently updatable neural network)**: Técnica utilizada en motores de ajedrez que significa "Redes Neuronales en Utilidad de Evaluación". Utiliza redes neuronales para evaluar posiciones de ajedrez de manera más eficiente y precisa que los métodos tradicionales.
- **PGN (Portable Game Notation):** Formato estándar para representar partidas de ajedrez en texto plano. Permite guardar y compartir partidas de ajedrez de manera legible por humanos y también puede incluir anotaciones, variantes y otros metadatos.
- **UCI (Universal Chess Interface)**: Estándar de comunicación utilizado para la interfaz entre motores de ajedrez y programas de ajedrez. Permite que diferentes programas de ajedrez puedan comunicarse entre sí de manera estandarizada.

13.2 Contenido Entregado en el Archivo adjunto

En este apartado se describirá de manera concisa el contenido del archivo adjunto entregado.

13.2.1 Contenidos

Directorio	Contenido
./ Directorio raíz del archivo adjunto	Contiene un instalador de la aplicación diseñado para Windows.
./zero-chess	Contiene la estructura completa de directorios del proyecto, que se detalla a continuación.

13.2.1.1 Estructura de la carpera "zero-chess"

En esta sección se describirá la estructura de la carpeta "zero-chess", que se encuentra en el archivo adjunto proporcionado:

Directorio	Contenido
./ Directorio raíz de la aplicación	Contiene los archivos necesarios para instalar las dependencias de la aplicación y configurar el editor, ESLint y Git.
./assets	Contiene las imágenes e iconos que se utilizan en la aplicación.
./release	Contiene los ejecutables de la aplicación y su instalador.
./src	Contiene los archivos fuente de la aplicación.

13.2.2 Código Ejecutable e Instalación

Esta sección proporciona una breve descripción sobre cómo instalar y ejecutar el proyecto. Para obtener más información sobre estos procesos, consulta los apartados 9.1 y 9.2.

13.2.2.1 Instalación

Para instalar el sistema, basta con ejecutar el instalador disponible en el directorio raíz del archivo adjunto.

13.2.2.2 Ejecución

Para iniciar el sistema, basta con ejecutar la aplicación una vez instalada.

13.3 Índice Alfabético

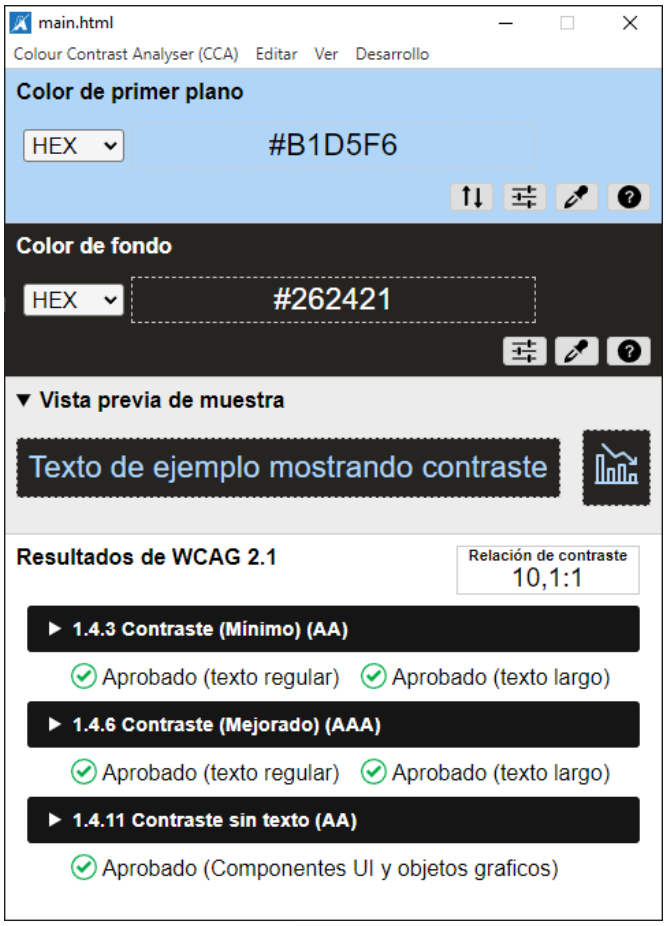
13.4 Código Fuente

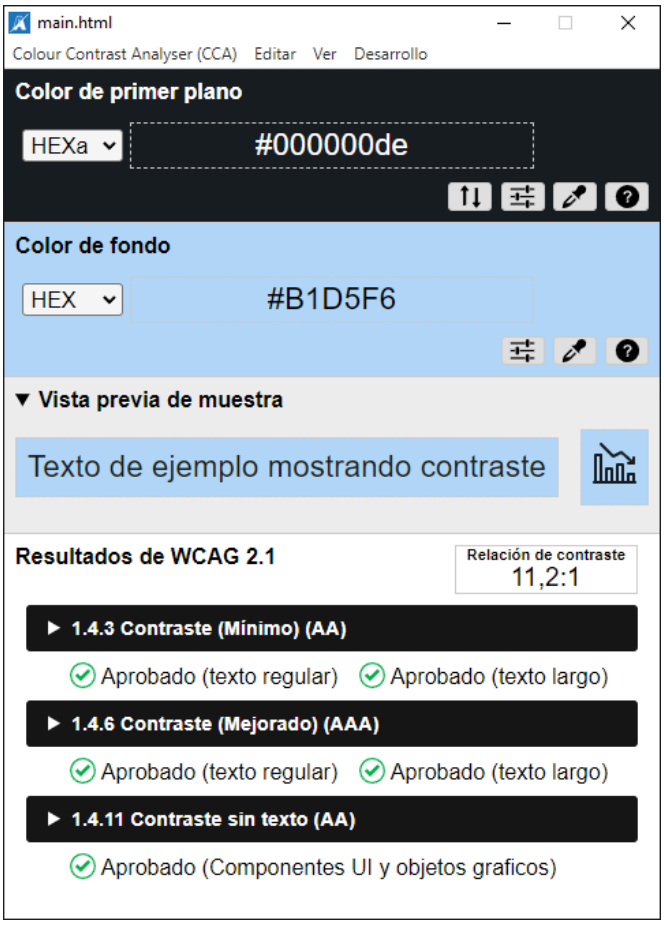
En este apartado se describirá el código fuente proporcionado, detallando cada uno de los directorios del proyecto y su contenido específico.

Directorio	Contenido
./src	Contiene varios subdirectorios que albergan el código fuente, además del código relacionado con la comunicación con Lichess, las interfaces y clases utilizadas, y la configuración de las pruebas.
./src/main	Contiene el código fuente de la lógica de Electron y de los servicios utilizados.
./src/main/database	Contiene el código fuente utilizado para manejar la base de datos.
./src/main/engine	Contiene el código fuente del motor, así como el código para cargarlo en un trabajador web y para interactuar con él. También incluye las funciones encargadas del procesamiento de sus salidas
./renderer	Contiene el código fuente del componente raíz, junto con sus pruebas y estilos, además de directorios equivalentes para cada ventana.
./renderer/GameAnalysis	Contiene el código fuente del componente de la ventana de análisis, junto con sus pruebas y estilos.
./renderer/GameList	Contiene el código fuente del componente de la ventana con la lista de partidas, así como sus pruebas y estilos.
./renderer/GameStatistics	Contiene el código fuente del componente de la ventana de estadísticas, junto con sus pruebas y estilos.
./renderer/Home	Contiene el código fuente del componente de la ventana de inicio, junto con sus pruebas y estilos.

13.5 Resultados de la herramienta CCA

En este apartado se incluyen los resultados de la herramienta Colour Contrast Analyzer (CCA), utilizada para verificar el contraste en las pruebas de accesibilidad:


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#B1D5F6
Color de fondo
HEX
#262421
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
10,1:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEXa
#000000de
Color de fondo
HEX
#B1D5F6
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
11,2:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#E79C9C
Color de fondo
HEX
#262421
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
7,1:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#FFAE80
Color de fondo
HEX
#262421
Vista previa de muestra
Resultados de WCAG 2.1
Relación de contraste
8,6:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEXa
#000000de
Color de fondo
HEX
#FFAE80
Vista previa de muestra
Resultados de WCAG 2.1
Relación de contraste
9,7:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#AEDF77
Color de fondo
HEX
#262421
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
10:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#A4D7F4
Color de fondo
HEX
#262421
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
10:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)


main.html
Colour Contrast Analyser (CCA) Editar Ver Desarrollo
Color de primer plano
HEX
#FFC852
Color de fondo
HEX
#262421
Vista previa de muestra
Texto de ejemplo mostrando contraste
Resultados de WCAG 2.1
Relación de contraste
10,1:1
1.4.3 Contraste (Mínimo) (AA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.6 Contraste (Mejorado) (AAA)
Aprobado (texto regular)
Aprobado (texto largo)
1.4.11 Contraste sin texto (AA)
Aprobado (Componentes UI y objetos graficos)