Taller de Proyecto 1

JULIAN ESTEBAN CADENA ROJAS

ROGER FABIAN BONILLA CARO

MATEO BECERRA PORRAS JACEL

THOMÁS ENCISO PINZÓN

Universidad Nacional De Colombia – Sede Bogotá

Facultad de Ingeniería

Grupo: 9

Curso: Ingeniería de software I

Punto #2:

Levantamiento de requerimientos y contexto del proyecto.

El ajedrez ha sido un juego estratégico que, a lo largo del tiempo, ha evolucionado en múltiples formas de juego, adaptándose a las necesidades de jugadores de distintos niveles. Con la digitalización del ajedrez, han surgido diversas plataformas que permiten a los usuarios jugar en línea, sin embargo, muchas de ellas presentan problemas recurrentes que afectan la experiencia de juego. Desde interfaces poco intuitivas hasta latencias que impactan la fluidez de las partidas, estos obstáculos dificultan el disfrute de un juego tan meticuloso y estratégico como el ajedrez.

Uno de los principales inconvenientes que enfrentan los jugadores en plataformas existentes es la falta de precisión en la ejecución de los movimientos, lo que puede generar errores en la validación de jugadas y afectar la equidad en una partida. Asimismo, la ausencia de opciones personalizadas de juego limita la diversidad de experiencias, reduciendo las posibilidades de adaptación a los diferentes estilos y niveles de habilidad de los jugadores. Además, el acceso a partidas contra inteligencia artificial en muchos sistemas actuales es restrictivo o poco desafiante, ya que carecen de un motor eficiente que se ajuste al nivel del jugador.

Para solucionar estos problemas, se propone el desarrollo de una plataforma web de ajedrez en línea que garantice partidas dinámicas, precisas y accesibles para jugadores de todos los niveles. Esta plataforma deberá priorizar un **diseño eficiente** que optimice el rendimiento sin comprometer la jugabilidad, asegurando que cada movimiento se ejecute de manera instantánea y sin retrasos perceptibles. La propuesta busca ofrecer una interfaz intuitiva que facilite la navegación, permitiendo a los usuarios encontrar partidas de manera rápida y sin complicaciones.

Uno de los aspectos clave del proyecto será la implementación de **un motor de juego avanzado que valide todos los movimientos conforme a las reglas oficiales del ajedrez**, incluyendo enroque, promoción de peón, tablas y jaque mate. La precisión en la aplicación de estas reglas garantizará que los jugadores experimenten partidas justas y equilibradas. Además, se habilitará la posibilidad de jugar contra **una inteligencia artificial configurable en distintos niveles de dificultad**, brindando a los usuarios la oportunidad de entrenar y mejorar su desempeño progresivamente.

Dado que el ajedrez es un juego de estrategia y aprendizaje, la plataforma también incluirá **un historial de partidas** con un análisis básico de movimientos, permitiendo a los jugadores revisar y estudiar sus jugadas previas para identificar errores y mejorar su técnica. Esta funcionalidad añadirá un valor educativo a la plataforma, convirtiéndola en una herramienta de aprendizaje además de un espacio de entretenimiento.

Otro elemento crucial será la incorporación de modos de juego personalizados, como **partidas blitz y bullet**, para ofrecer opciones dinámicas y variadas según la preferencia del usuario. Asimismo, se añadirá una opción de juego local que permita a dos jugadores compartir un mismo dispositivo o que un usuario pueda jugar contra sí mismo para analizar posiciones estratégicas.

En términos de interacción social, la plataforma contará con un sistema de comunicación dentro de las partidas, donde los jugadores podrán intercambiar mensajes en **un chat**

habilitado durante el juego. Para mayor comodidad, se incluirá la opción de desactivar esta función si el usuario lo prefiere, evitando distracciones innecesarias.

La propuesta de esta plataforma no solo busca ofrecer una solución a los problemas actuales del ajedrez en línea, sino que pretende revolucionar la manera en que los jugadores experimentan el juego en un entorno digital. Al desarrollar un sistema eficiente, accesible y dinámico, se garantizará una experiencia óptima para todos los usuarios, sin importar su nivel de habilidad.

Punto #3:

Análisis de requerimientos con el método MoSCoW

Requerimiento	Importancia	Estimación de esfuerzo	Justificación
Jugar ajedrez en línea	Must Have	5	Jugar ajedrez en línea es esencial para el proyecto, aunque requiere WebSockets y gestión de sesiones. Existen herramientas para facilitarte, pero su integración es compleja. Su fluidez impacta directamente la experiencia del usuario y define el éxito de la plataforma.
Reglas y validación del juego	Must Have	1	Las reglas y validación del juego son fundamentales, pero librerías como chess.js reducen su complejidad. Su correcta implementación garantiza partidas justas y precisas, impactando la experiencia del usuario y asegurando la coherencia con los objetivos del proyecto.
Jugar ajedrez en modo un jugador	Must Have	3	El modo de un jugador con IA es crucial para ofrecer una experiencia completa. Integrar el motor Stockfish facilita esta funcionalidad, aunque requiere configuración técnica. Su inclusión mejora la experiencia del usuario y se alinea con el objetivo de ofrecer desafíos diversos.
Registro de usuario e inicio de sesión	Should Have	5	El registro de usuario e inicio de sesión son esenciales para personalizar la experiencia y gestionar partidas. Aunque existen librerías que facilitan su

			implementación, requiere atención en seguridad. Su correcta integración mejora la experiencia del usuario y cumple con los objetivos de ofrecer un servicio personalizado y seguro.
Chat en línea	Should Have	5	Un chat en vivo es una funcionalidad valiosa para la interacción entre jugadores, pero requiere la integración de sistemas de mensajería en tiempo real. Aunque existen herramientas disponibles, su implementación debe garantizar baja latencia y seguridad. Mejora la experiencia del usuario y fomenta la comunidad, alineándose con los objetivos de ofrecer una plataforma interactiva.
Historial de partidas	Could Have	3	El historial de partidas es clave para ofrecer a los usuarios una forma de revisar su progreso y mejorar su juego. Su implementación es relativamente sencilla con bases de datos, pero debe ser eficiente para manejar grandes volúmenes de datos. Mejora la experiencia del usuario y cumple con el objetivo de proporcionar una plataforma que fomente el aprendizaje y la mejora continua.

Partida Local	Could Have	3	La implementación de un modo de juego local es una tarea moderadamente sencilla en términos técnicos, ya que reutiliza muchas de las estructuras existentes del sistema. No requiere comunicación en red ni un motor de inteligencia artificial, pero sí necesita ajustes en la gestión de turnos, la interfaz de usuario y el almacenamiento temporal de la partida.

Modos de juego	Won´t Have	8	Ofrecer distintas modalidades y estilos de juego en ajedrez amplía la experiencia del usuario, permitiendo partidas rápidas, clásicas o personalizadas. Su implementación es compleja por la necesidad de ajustes en las reglas y la interfaz del sistema.

Punto #4:

Fase de Desarrollo	***Funcionalidad***	***Tiempo***	***Actividad y Herramientas***
Planificación y Diseño	Diseño de interfaz de inicio del juego	1 día	Realizar los diseños visuales del juego, incluyendo la interfaz de inicio, el tablero y los elementos interactivos. **Herramientas:** - Figma
	Diseño del juego en funcionamiento	2 días	
	Definición de reglas y validación del juego	1 día	
Desarrollo	Registro de usuario e inicio de sesión (Backend Login + UI)	3 días	Implementar el sistema de autenticación y la pantalla de inicio. Implementar la lógica principal del juego en sus diferentes modos. **Herramientas:** - React (Frontend) - NodeJS (Backend) - Express (Creación de APIs) - PostgreSQL (Base de datos) - StockFish (Ajedrez IA) - Socket.io (Conexión tiempo real).
	Interfaz de inicio	1 día	
	Programación de reglas y validación del juego	5 días	
	Jugar ajedrez en línea	4 días	
	Jugar ajedrez en modo un jugador (vs IA)	2 días	
Funciones Adicionales	Partida Local	2 días	Implementar modulos de funcionalidades adicionales del sistema. **Herramientas:** - React (Frontend) - NodeJS (Backend) - Express (Creación de APIs) - PostgreSQL (Base de datos)
	Chat en línea	2 días	
	Historial de partidas	2 días	
Pruebas y Ajustes Finales	Testing (Pruebas de UI, backend y lógica del juego)	4 días	- Realizar testing - Integración del sistema - Corrección de errores - Optimización

Alcance del proyecto:

El alcance de este proyecto abarca el desarrollo completo de una plataforma de ajedrez en línea con todas las funcionalidades esenciales para su correcto funcionamiento. Se garantizará la implementación de un motor de juego que valide y ejecute las reglas del ajedrez, un sistema de autenticación seguro para el registro e inicio de sesión de los usuarios, y una interfaz intuitiva.

Además, se desarrollará un sistema de emparejamiento para partidas en línea, un modo de juego contra inteligencia artificial y la opción de partidas locales. Sin embargo, las funcionalidades adicionales, como modos de juego avanzados, chat en linea, historial de partidas y características extra, no forman parte del MVP, aunque podrían ser consideradas en fases futuras del desarrollo.

Funcionalidades Incluidas en el MVP	
Fase de Desarrollo asociada	***Funcionalidades***
Planificación y Diseño	Diseño de interfaz de inicio del juego
	Diseño del juego en funcionamiento
	Definición de reglas y validación del juego
Desarrollo	Registro de usuario e inicio de sesión (Backend Login + UI)
	Interfaz de inicio
	Programación de reglas y validación del juego
	Jugar ajedrez en línea
	Jugar ajedrez en modo un jugador (vs IA)
Funciones Adicionales	Partida Local

Funcionalidades excluidas del MVP	
Fase de Desarrollo asociada	***Funcionalidades***
Funciones Adicionales	Chat en línea
	Historial de partidas

Costos:

Los costos de cada uno de los actores del programa varían dependiendo de su experiencia y conocimientos, así como de sus funciones dentro del mismo, las siguientes estimaciones se realizaron con base en la locación de Bogotá, Colombia.

Salarios desarrolladores:

Rol / Experiencia / Función	**Estimación Salario mensual (COP)**
Desarrollador Junior	$ 2’200.000
Desarrollador Senior	$ 4’700.000
Desarrollado Back-end	$ 4’100.000
Desarrollador Front-end / UX	$ 2’750.000
Desarrollador Full-Stack	$ 4’500.000
Desarrollador Web	$ 2’200.000
Tester	$ 2’600.000

La información fue obtenida de las siguientes fuentes:

- *https://co.talent.com*
- *https://co.computrabajo.com*
- *https://co.indeed.com/?from=gnav-homepage*

Estimación de costos:

Tipo de Costo	Costo mensual (COP)	Costo Total mensual (COP)	Justificación
Costo de Personal: Desarrollador Junior	$ 2’000.000	$ 8’000.000	Salario para cuatro desarrolladores junior sin experiencia
Herramientas de desarrollo	$ 0	$ 0	Usando planes gratuitos en Firebase y herramientas de código abierto
Servidor en la nube: AWS	$ 50.555	$ 50.555	Dependiendo de los recursos requeridos.
TOTAL	$ 8’050.555		Coste estimado del proyecto, sujeto a la duración del desarrollo.

Las estimaciones fueron planteadas en base a las siguientes fuentes:

- *https://aws.amazon.com/es/amplify/pricing*
- *https://firebase.google.com/pricing?authuser=0&hl=es-419*

Parte 2

Punto #8

Arquitectura del Sistema

Cliente (React y WebSockets):

- **React:** Interfaz de usuario dinámica y responsiva para el tablero de ajedrez.
- **WebSockets:** Sincronización en tiempo real de movimientos entre jugadores.
- **Web Requests:** El backend procesa solicitudes HTTP y gestiona la lógica del juego, como validación de movimientos y gestión de partidas.

Service (Node.js, Express.js, PostgreSQL, WebSockets):

- **Node.js:** Backend eficiente para manejar múltiples conexiones simultáneas.
- **Express.js**: Define rutas y maneja solicitudes HTTP.
- **PostgreSQL**: Almacena datos estructurados como usuarios, partidas y estadísticas.
- **WebSockets:** Comunicación en tiempo real entre backend y clientes.

Firebase Services (Firebase Auth, Realtime DB, FCM, Storage):

- **Firebase Auth:** Autenticación segura de usuarios.
- **Firebase Realtime DB**: Sincronización en tiempo real de datos.
- **Firebase FCM**: Notificaciones push.
- **Firebase Storage**: Almacenamiento de archivos como imágenes de perfil.

SQL Queries / Data Management:

- PostgreSQL gestiona consultas SQL y datos estructurados.

Push Notifications:

- Firebase FCM gestiona notificaciones push.

PostgreSQL DB:

- Tablas para usuarios, partidas y estadísticas.

Justificación de la Arquitectura

Requisitos del Proyecto:
- React y WebSockets para interfaz dinámica y sincronización en tiempo real.
- Firebase para facilidad de uso y capacidades en tiempo real.
- PostgreSQL para datos estructurados.

Recursos Disponibles:
- Node.js y Express.js con amplia comunidad y recursos.
- Firebase para integración sencilla de autenticación y notificaciones.

Escalabilidad:
- Node.js y Firebase ofrecen escalabilidad para múltiples partidas simultáneas.
- Diseño del Esquema de la Base de Datos

Normalización:
- 1NF: Columnas atómicas.
- 2NF: Atributos no clave dependen completamente de la clave primaria.
- 3NF: Atributos no clave son independientes entre sí.

Claves Primarias y Foráneas:
- Claves primarias únicas en cada tabla.
- Claves foráneas para relaciones entre tablas.

Índices:
- Mejoran el rendimiento de las consultas.

Relaciones:
- Uno a Muchos: Un usuario puede tener muchas partidas.
- Muchos a Muchos: Amistades entre usuarios.

Justificación de la Elección de una Base de Datos Relacional (SQL)

Estructura de Datos:

Los datos en una aplicación de ajedrez en línea son altamente estructurados, con relaciones claras entre entidades como usuarios, partidas y amistades. Las bases de datos relacionales son ideales para manejar este tipo de datos estructurados y relaciones complejas.

Integridad Referencial:

Las bases de datos relacionales ofrecen integridad referencial a través de claves primarias y foráneas, lo que asegura que las relaciones entre tablas sean consistentes. Por ejemplo, no se puede eliminar un usuario que tenga partidas asociadas sin antes manejar esas partidas.

Consultas Complejas:

SQL permite realizar consultas complejas y transacciones ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad), lo que es crucial para operaciones como registrar movimientos, actualizar

estadísticas y gestionar amistades.

Normalización:

La normalización reduce la redundancia de datos y mejora la integridad de los datos. En una aplicación de ajedrez, donde los datos como el historial de partidas y las estadísticas de los usuarios son críticos, la normalización ayuda a mantener la consistencia y precisión de los datos.

Escalabilidad:

Aunque las bases de datos NoSQL son conocidas por su escalabilidad horizontal, las bases de datos relacionales también pueden escalar verticalmente y, con técnicas como el sharding, pueden manejar grandes volúmenes de datos y usuarios.

Justificación de la Elección de una Base de Datos No Relacional (NoSQL)

Además de la base de datos relacional (SQL), se ha optado por utilizar una base de datos no relacional (NoSQL), específicamente Firebase Realtime Database, para ciertos aspectos de la aplicación de ajedrez en línea.

Datos en Tiempo Real:

Firebase Realtime Database es ideal para manejar datos que requieren sincronización en tiempo real, como el estado actual de las partidas y las salas de espera. Esto es crucial para una aplicación de ajedrez, donde los movimientos de las piezas deben reflejarse instantáneamente en ambos clientes.

Escalabilidad Horizontal:

Las bases de datos NoSQL, como Firebase, están diseñadas para escalar horizontalmente, lo que significa que pueden manejar un gran número de conexiones simultáneas y un alto volumen de datos. Esto es beneficioso para una aplicación de ajedrez en línea que puede experimentar picos de usuarios durante torneos o eventos especiales.

Flexibilidad en el Esquema:

NoSQL ofrece un esquema flexible, lo que permite almacenar datos no estructurados o semi-estructurados. Esto es útil para manejar datos como notificaciones push, mensajes en tiempo real y configuraciones de partidas personalizadas, que pueden variar en estructura y no necesitan seguir un esquema rígido.

Integración con Servicios de Firebase:

Firebase ofrece una suite de servicios integrados, como Firebase Auth para autenticación, Firebase Cloud Messaging (FCM) para notificaciones push, y Firebase Storage para almacenamiento de archivos. Utilizar Firebase Realtime Database permite una integración fluida con estos servicios, simplificando el desarrollo y la gestión de la aplicación.

Rendimiento y Latencia:

Firebase Realtime Database está optimizado para ofrecer baja latencia y alto rendimiento en operaciones de lectura y escritura. Esto es esencial para garantizar una experiencia de usuario fluida en una aplicación de ajedrez en línea, donde los retrasos pueden afectar negativamente la experiencia de juego.

Manejo de Datos No Relacionales:

Algunos datos, como las notificaciones push y los mensajes en tiempo real, no tienen una estructura relacional clara y no se benefician de las características de una base de datos SQL. Firebase Realtime Database es más adecuado para manejar este tipo de datos de manera eficiente.

Punto #9
Patrones de diseño:

Para el desarrollo del proyecto, utilizamos principalmente el patrón de diseño Observer, ya que permite que múltiples componentes del código reaccionen automáticamente ante cambios en el estado del juego. En nuestro caso, es fundamental que tanto el servidor como los jugadores estén constantemente notificados sobre el desarrollo de la partida. Este patrón nos permite mantener sincronizados la interfaz de usuario, la IA, los jugadores en línea y Firebase con cada movimiento realizado en el tablero.

Diagrama UML