Levantamiento de requerimientos

Desde sus inicios, el proyecto Twordle ha sido una iniciativa con un fuerte sentido de pertenencia por parte del equipo, particularmente de quien propuso la idea original. Su evolución, de un prototipo en Python con malas prácticas a una versión web bien estructurada, refleja la intención de mejorar no solo en términos técnicos, sino también en la experiencia que se ofrece a los jugadores. La meta principal de este proyecto es transformar un juego de palabras en una competencia emocionante.

Wordle, la inspiración para este proyecto, no proporciona una experiencia competitiva para aquellos usuarios que aman demostrar sus habilidades, por lo que Twordle busca satisfacer el lado más competitivo de las personas en un juego de palabras que en principio parece no ser desafiante, pero bajo presión es distinto.

Twordle busca proporcionar una experiencia competitiva en la que los jugadores se enfrenten entre sí en tiempo real. Para lograr esto, se han identificado una serie de funcionalidades esenciales que permitirán ofrecer una experiencia fluida e intuitiva:

I. **Pantalla de inicio:** Debe ser clara y atractiva, presentando las instrucciones del juego de manera sencilla para que los usuarios comprendan rápidamente la mecánica. Se hará uso de colores que faciliten la comprensión sin abrumar al jugador. Desde esta pantalla, los usuarios podrán acceder al juego a través de un botón de "Jugar", que los redireccionará a una partida.

II. **Tutorial de juego:** Para aquellos jugadores que ingresan por primera vez o deseen repasar las reglas, se incluirá un tutorial breve y dinámico que explique las mecánicas de juego.

III. **Textfield para nickname:** Antes de comenzar una partida, los jugadores deberán ingresar un nombre de usuario que los identificará durante el juego.

IV. **Sistema de emparejamiento:** Dado que Twordle es una experiencia multijugador, será necesario un sistema que conecte a los jugadores en función de su disponibilidad. Este sistema emparejará automáticamente a los usuarios en duelos de palabras en tiempo real.

V. **Jugabilidad:** Se establecerá un modo de juego base en el que los jugadores intenten adivinar la palabra oculta antes que su oponente, haciendo uso de pistas y estrategias para lograrlo.

VI. **Opción de nueva partida:** Una vez finalizado un duelo, los jugadores podrán optar por iniciar otra partida y continuar con la experiencia competitiva sin interrupciones.

El desarrollo de estas funcionalidades no solo permitirá cumplir con los objetivos del proyecto, sino que también brindará un entorno de aprendizaje desafiante para el equipo, reflejando situaciones del mundo laboral en las que se requiere aprender nuevas tecnologías y aplicarlas en plazos reducidos. A medida que avanza el desarrollo, se busca mantener una comunicación constante y una gestión eficiente del trabajo, con el fin de garantizar un resultado que cumpla con las expectativas de todos los involucrados.

Análisis de Requerimientos

- MoSCoW 🐄

Must-Have	Should-Have	Could-Have	Won't-Have
Tutorial de juego	Estadísticas al final de la partida	Modo oscuro	
Textfield para nickname	Emparejamiento específico (Sala privada de juego)	Más modos de juego (científico, tilde)	
Sistema de emparejamiento	Botón de revancha		
Modo de juego normal	Página responsive		
Opción de nueva partida			

- Planning Poker ♦

Requerimiento	No. Días	Justificación
Tutorial de Juego	1	Tanto la explicación del juego como los ejemplos serían más que todo texto e imágenes ilustrativas, por lo tanto, no tomará mucho tiempo ni dificultades técnicas.
Textfield para nickname	1	Tener la interfaz y lógica para leer el input del usuario y asignar dicho input al nombre de cada host es algo que no requiere de conocimientos avanzados o acciones adicionales que puedan convertirlo en un proceso complicado, por ello, estaría listo en un día.
Sistema de emparejamiento	8	Este sistema debe soportar la búsqueda y conexión de usuarios activos simultáneamente, cosa que requiere de diversas dependencias y asegurarse de que funcione

		adecuadamente, además consideramos que es el mayor reto técnico del proyecto.
Modo de juego normal	8	Toda la lógica del juego está en este requerimiento, desde seleccionar una palabra al azar del banco de palabras hasta la lógica de colores por letra, por lo que tomará tiempo desarrollar este requerimiento.
Opción de nueva partida	1	Este requerimiento conlleva crear un botón que empareje al usuario con otro, como la lógica de emparejamiento ya existía previamente, desarrollar este botón no tomará tiempo.
Estadísticas al final de la partida	1	Hacer un seguimiento de lo que sucede en las partidas y posteriormente mostrar los datos obtenidos es algo que consiste en almacenar información temporalmente en algunas variables, de tal modo que, no resulta complicado.
Emparejamiento específico (Sala privada de juego)	5	Que los usuarios puedan crear una sala y mantener una identificación única para cada una, implica la implementación de búsqueda y verificación de partidas existentes o expiradas y una infraestructura necesaria para cada host, lo que se convierte en un proceso que requiere de algunos días para ser desarrollado adecuadamente.
Botón de revancha	1	Un botón de revancha entre dos jugadores previamente emparejados no es difícil ya que lo laborioso era emparejarlos. La lógica para este requerimiento simplemente consiste en verificar una flag de Revancha por parte de ambos jugadores.
Página responsive	5	A pesar de que las páginas responsive son un básico de la web, resulta complicado tener estas prácticas para personas poco experimentadas en el desarrollo web. Por lo tanto, desarrollar una página responsive puede ser desafiante y llevar un poco más de lo esperado.
Modo oscuro	1	Al ser un requerimiento could-have, la página principal ya estaría funcional y desarrollar la característica de modo oscuro no debería tomar más de un día.
Más modos de juego (científico, tilde)	2	Los modos de juego extra solo implicaría tomar palabras de otro banco de palabras (científicas y con tildes). El modo de juego con tilde necesitaría un teclado con 5 teclas extras referentes a las vocales con tildes. Por lo anterior, llevar a cabo este requerimiento no es complejo.

Análisis gestión de software

Tiempo:

El proyecto se dividió en 6 etapas: Identificación y levantamiento de requerimientos, diseño de interfaz de usuario, estudio de las tecnologías a usar, planteamiento de la estructura (frontend-backend), desarrollo de la aplicación y por último la etapa de testing y corrección de errores. Para lo anterior, se realizó un análisis del tiempo estimado por etapa:

- **Identificación y levantamiento de requerimientos:** Como este proyecto está inspirado en otro, el tiempo esperado para analizar los requerimientos es corto y no superior a 3 días, ya que consiste únicamente en tomar las características de Wordle y realizar las respectivas modificaciones y adaptaciones para Twordle.
- **Diseño de interfaz de usuario:** Igual que la identificación de los requerimientos, el diseño de interfaz de usuario está basado en Wordle, por lo que diseñar la GUI es solo un proceso de reproducción de la inspiración. Se aproxima un total de 2 días.
- **Estudio de las tecnologías:** Esta etapa junto a la del desarrollo son las más largas. Estudiar las tecnologías lleva tiempo y experimentación para tener las mejores prácticas y no caer en código o arquitecturas sucias. Aunque no se tiene un tiempo esperado para esta etapa, es válido afirmar que aprender y pulir las habilidades en las distintas tecnologías puede tardar lo mismo que la vida del proyecto.
- **Planteamiento de la estructura:** Se espera que la etapa para plantear la estructura no tome mucho tiempo, sin embargo puede resultar tediosa ya que consiste en realizar la ruta de funcionamiento de la aplicación web y el grupo no es muy experimentado en esta esta tarea, por lo que se esperan problemas a la hora de conectar el frontend con el backend si la estructura no queda bien planteada. Se espera por lo menos 5 días en ello.
- **Desarrollo de la aplicación:** Esta es la etapa más larga y crítica, donde se evidenciará de primera mano que tan bien se realizaron las etapas anteriores, sumado a que se verá que tan factible es realizar la aplicación dadas las habilidades del equipo. Se estima un tiempo de 3 a 4 semanas para desarrollar la aplicación.
- **Testing y corrección de errores:** Testear y corregir los errores solo supondrá una parte pequeña del tiempo de vida del proyecto (amén 🙏), se espera que la etapa de desarrollo no conlleve muchos errores y así mismo tardar poco en la corrección de cualquier imprevisto.

Dado el análisis anterior, se espera que todo el proyecto se lleve a cabo en un mínimo de 5 semanas y un máximo de 7 semanas. Por lo tanto, ahora se realizará una exploración de los posibles costos asociados con el desarrollo de una aplicación de este tipo, basándonos en diversas fuentes de Internet.

Costos:

El servicio de hosting para alojar la página web aún no se ha decidido, además de que los costos de operación pueden variar dependiendo del uso que se le de al alojamiento durante las etapas de desarrollo y testing; por lo anterior, se supondrá un costo total de cero. Suponiendo un tiempo de un mes y medio, y teniendo en cuenta que roles como el de diseñador o tester no van a estar durante todo el proceso de creación, se plantea la siguiente tabla de costos:

Rol	Costo mensual (COP)	Duración (meses)	Costo total
Desarrollador web junior (Frontend)	2’400.000	1.5	3’600.000
Desarrollador web junior (Backend)	3’800.000	1.5	5’700.000
Diseñador UX (Super mega junior del SENA)	2’800.000	0.5	1’400.000
Firebase	Desde 0 hasta 201’852.107 (dependiendo de la infraestructura consumida)	1.5	0 (en nuestro caso)

Alcance:

El alcance mínimo viable del proyecto serán las funcionalidades Must-Have definidas en el análisis MoSCoW, es decir: el tutorial, soporte para nicknames, emparejamiento entre jugadores, modo de juego básico (el juego como tal) y una opción de nueva partida. Las demás funcionalidades establecidas en MoSCoW (Should-Have y Could-Have) podrían mejorar la experiencia del usuario, sin embargo, por cuestiones de tiempo y debido a que el alcance mínimo viable es, en efecto, un producto terminado, consideramos que las funcionalidades que no se encuentran en Must-Have no las vamos a tener en cuenta.

Diseño y Arquitectura

Arquitectura del sistema

La arquitectura del sistema dentro del proyecto se basa en el modelo Modelo-Vista-Controlador (MVC), elegido tanto por comodidad como por su conexión con las tecnologías utilizadas. En este caso, el Modelo vendría siendo representado por Firebase, ya que en él se gestionan tanto la lógica de la aplicación web como el almacenamiento de datos. La Vista corresponde a la interfaz proporcionada por React, y el Controlador es la lógica general del juego, incluyendo su funcionamiento, reglas y restricciones.

Esta arquitectura responde a las necesidades y requerimientos previamente analizados mediante la metodología MoSCoW. Además, al tratarse de un juego de sencillo, su despliegue en Firebase resulta una opción eficiente, permitiendo una integración simple facilitando tanto el mantenimiento como futuras actualizaciones del sistema.

Diseño de bases de datos

Se optó por una base de datos NoSQL (Firestore) debido a que el juego requiere actualizaciones en directo entre jugadores. Firestore proporciona sincronización en tiempo real de forma nativa. Aunque el juego es sencillo, Firestore permite manejar múltiples partidas simultáneas, además la flexibilidad del esquema permite añadir nuevos campos sin afectar la estructura existente. También permite definir reglas de acceso y aislamiento de datos.

Se organizó por:

I. **GAMEROOM**: Almacena información que permite gestionar las partidas y el estado del juego según el jugador.
II. **PLAYERS** (Subcolección de GAMEROOM): Mantiene información temporal de los jugadores para el seguimiento del resultado de cada sesión de juego.
III. **WORD** (Subcolección de GAMEROOM): Almacena la cantidad de palabras y la última actualización del diccionario, para mantener un registro organizado.
IV. **GAMEMODE** (Subcolección de WORD): Guarda la palabra y su longitud para establecer la matriz adecuada de la partida.

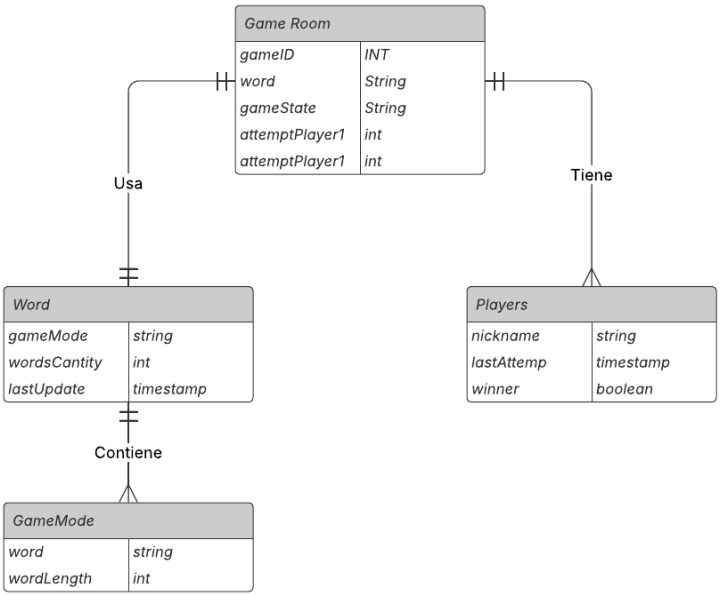

Diagrama Entidad-Relación

Patrones de diseño

Para el desarrollo de este proyecto se utilizó principalmente el diseño Observer, ya que la lógica de juego de Twordle es muy sencilla: un jugador escribe una palabra y la envía, automáticamente el “servidor” realiza una comprobación de la palabra y envía los datos de vuelta a ambos jugadores para saber cuáles letras son correctas. Lo anterior coincide con este patrón, ya que el jugador está constantemente notificando al servidor y al otro jugador sobre sus acciones. Este patrón se repite en todo el ciclo de juego entre dos jugadores ya que constantemente están recibiendo los intentos del otro, además se actualiza el tablero y el teclado.

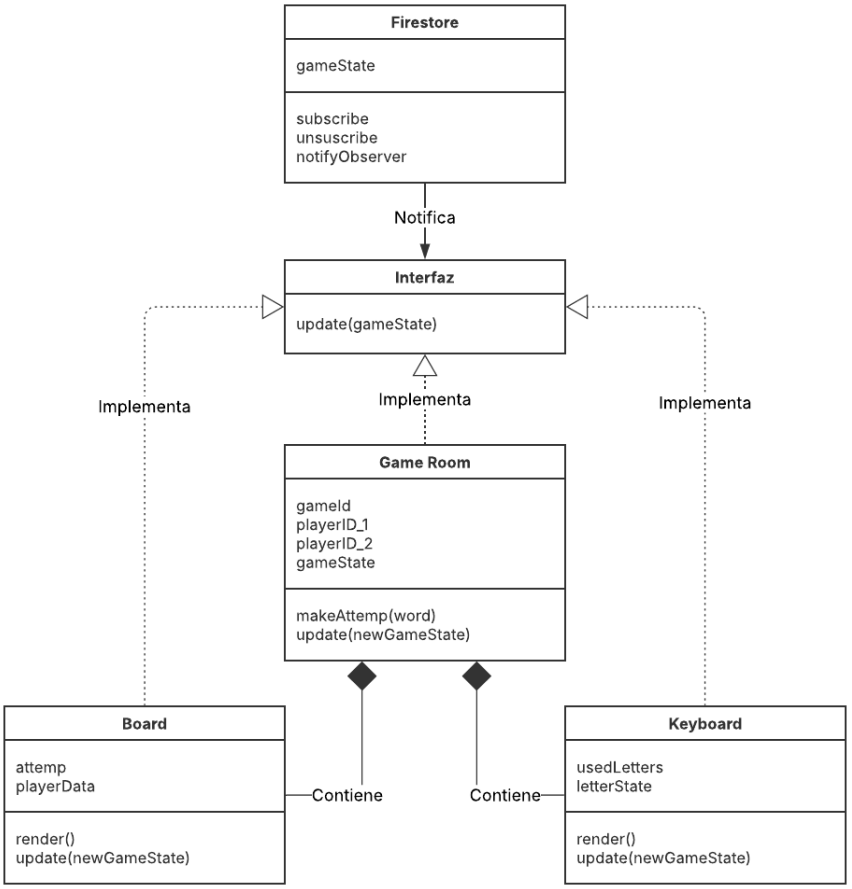

Diagrama UML