2023 - 2024

IES San Vicente

Jorge Sánchez González

[ONE CARD]

Memoria del proyecto final de ciclo: One Card

Contenido

1. Introducción

El mundo de los juegos de cartas coleccionables (TCG) ha capturado mi imaginación y la de jugadores de todas las edades desde sus inicios. Con títulos icónicos como Magic, Legends of Runeterra y Hearthstone, los TCG ofrecen una combinación única de estrategia, coleccionismo y narrativa inmersiva. En este proyecto, me baso en estos fundamentos, pero introduzco una nueva y emocionante dimensión al mezclar estos elementos con el vasto y rico universo de One Piece, una de las franquicias de manga y anime más queridas y populares a nivel mundial.

El juego de cartas basado en One Piece que he desarrollado tiene como objetivo proporcionar una experiencia de juego única y envolvente para los fanáticos de la serie, así como para los entusiastas de los TCG. Utilizando tecnologías modernas como NodeJS y ReactJS, junto con sockets para la conexión en tiempo real entre jugadores, he creado una plataforma que permite a los jugadores sumergirse en batallas estratégicas y dinámicas. La elección de estas tecnologías no solo garantiza un buen rendimiento y una experiencia de usuario fluida, sino que también facilita futuras expansiones y mejoras al juego.

En el núcleo de mi juego están las tripulaciones de One Piece. Los jugadores pueden elegir entre la tripulación de los Sombrero de Paja, liderada por Monkey D. Luffy, o la Marina, con figuras emblemáticas como Akainu y Kizaru. Cada tripulación ofrece un conjunto único de cartas de personaje, cartas de campo y cartas mágicas, lo que permite a los jugadores desarrollar estrategias diversas y adaptativas. Las cartas de personaje representan a los miembros principales de cada facción, con atributos de ataque, defensa y energía que reflejan sus habilidades en el anime y manga. Las cartas de campo son cruciales para gestionar los recursos de energía, necesarios para ejecutar ataques. Las cartas mágicas, divididas en acciones sobre jugadores o personajes, añaden una capa adicional de táctica y sorpresa al juego.

El flujo del juego está diseñado para ser accesible y emocionante. Comienza con una fase de espera hasta que se conecten dos jugadores. Cada turno está estructurado en varias fases: desde la fase de robo, donde los jugadores añaden nuevas cartas a su mano, hasta las fases de bajada de campo y no campo, y finalmente las fases de ataque y defensa. Esta secuencia garantiza que cada turno sea una oportunidad para planificar y ejecutar estrategias.

La inspiración para este proyecto surge tanto de mi amor por el universo de One Piece como del deseo de crear una experiencia de TCG que pueda competir con los mejores juegos del género. Magic me enseñó la importancia de una rica mitología y profundidad estratégica. Legends of Runeterra me mostró cómo las mecánicas de juego dinámico y los campeones evolucionables pueden mantener a los jugadores comprometidos, mecánica que me hubiese encantado incorporar a mi proyecto. Hearthstone me inspiró con su accesibilidad y su diseño intuitivo y pulido, así como con sus habilidades especiales de cada personaje. Combinando estos aprendizajes con los personajes de One Piece, aspiro a crear un juego que no solo entretenga, sino que también conecte profundamente con los jugadores a través de su pasión compartida por el universo.

2. Antecedentes

Magic

Magic es un juego de cartas coleccionables que comparte varias similitudes con el juego de cartas que he desarrollado. Ambos juegos involucran el uso de cartas para representar personajes y hechizos, y cada jugador tiene un mazo personalizado. En Magic, los jugadores utilizan tierras (similares a las cartas de campo en el juego de One Piece) para generar maná, que es necesario para jugar cartas de criatura y hechizos. Las criaturas en Magic, análogas a las cartas de personaje en el juego de One Piece, también tienen atributos de ataque y defensa. Sin embargo, Magic tiene una mayor variedad de tipos de cartas y habilidades, ofreciendo una complejidad estratégica más profunda y un meta-juego más amplio debido a su extensa historia y expansión continua.

A diferencia del juego de One Piece, que se basa en tripulaciones específicas y tiene un enfoque temático en el universo de One Piece, Magic tiene cinco colores de maná, cada uno representando los diferentes tipos de elementos que tienen las criaturas, lo que añade una capa adicional de elección estratégica al construir un mazo. Además, Magic tiene un sistema de fases del turno más detallado y estructurado, con múltiples oportunidades para que los jugadores respondan a las acciones del oponente, creando un juego dinámico y a menudo impredecible. En comparación, el juego de One Piece parece tener un flujo de juego más directo y posiblemente más rápido, lo que podría hacerlo más accesible para nuevos jugadores o aquellos menos familiarizados con los juegos de cartas.

Legends of Runeterra (LoR)

Legends of Runeterra (LoR) es otro juego de cartas coleccionables digital que, al igual que mi juego de One Piece, permite a los jugadores construir mazos basados en personajes y hechizos específicos. LoR se centra en el universo de League of Legends, permitiendo a los jugadores utilizar campeones y cartas de diferentes regiones, que es similar a cómo los jugadores eligen entre las tripulaciones de Sombrero de Paja y la Marina. En ambos juegos, los jugadores deben gestionar sus recursos (mana en LoR y cartas de campo en One Piece) para jugar cartas y realizar ataques, con el objetivo de reducir los puntos de vida del oponente a cero.

Una diferencia significativa es que LoR tiene un sistema de turnos alternados donde ambos jugadores tienen la oportunidad de atacar y defender en cada ronda, creando una dinámica de juego muy interactiva y estratégica. Además, incorpora un sistema de progresión de campeones, donde ciertos personajes pueden subir de nivel y desbloquear habilidades más poderosas al cumplir condiciones específicas, añadiendo otra capa de estrategia a la construcción del mazo y el juego. El juego que he desarrollado tiene una estructura de turnos más tradicional y predecible, con fases de ataque y defensa claramente definidas. Esto podría hacer que el juego sea más fácil de aprender y jugar, especialmente para jugadores que tal vez no estén familiarizados con mecánicas de juego de cartas más complejas.

Hearthstone

Hearthstone, desarrollado por Blizzard, es otro popular juego de cartas digital que comparte algunas mecánicas con todos los anteriores. En Hearthstone, los jugadores utilizan mazos construidos alrededor de héroes específicos con poderes únicos. Al igual que los anteriores, Hearthstone utiliza un sistema de maná que aumenta con cada turno para jugar cartas de criaturas (personajes) y hechizos (cartas mágicas), y el objetivo es reducir los puntos de vida del oponente a cero.

Una diferencia clave es que Hearthstone tiene una fuerte integración con su plataforma digital, incluyendo animaciones, efectos de sonido y una interfaz de usuario pulida que mejora la experiencia de juego. En mi caso, esto supone una complejidad técnica y una falta de los recursos necesarios para igualar el nivel de pulido de Hearthstone, sin embargo, una menor carga en niveles de interfaz hace más accesible su uso para jugadores que no dispongan de sistemas potentes.

3. Análisis

La fase de análisis de mi proyecto se centra en desglosar y comprender todos los componentes y funcionalidades necesarias para desarrollar el juego de cartas.

Este análisis incluye la identificación de los requisitos del sistema, tanto funcionales como no funcionales, así como la elaboración de un modelo detallado de cómo interactuarán los diferentes elementos del juego. Al explorar cada aspecto del juego, desde la gestión de las cartas hasta la experiencia del usuario, puedo asegurarme de que todas las partes trabajen de manera cohesiva y eficiente, proporcionando una experiencia de juego fluida y atractiva.

Análisis funcional

Jugadores:

El juego permanecerá en espera hasta que se conecten un total de 2 jugadores.

Tripulaciones:

El jugador tendrá una de las dos tripulaciones definidas: Sombrero de Paja o Marina.

Mazo:

Cada mazo está compuesto por cartas que dependen de la tripulación escogída por el usuario y por unas cartas mágicas comunes para ambas tripulaciones.

Cada mazo tendrá:

- Cartas de personaje
- Cartas de campo
- Cartas mágicas

Cartas de personaje:

Se representan con el color rojo.

Representan a los personajes principales de One Piece (por ejemplo Luffy, Nami y Zoro para la tripulacion de los Sombrero de Paja o Akainu y Kizaru para la tripulacion de la Marina)

Tienen atributos de ataque, defensa y energía.

Cartas de campo:

Se representan con el color verde.

Cada carta de campo permite aumentar el energía del jugador en 1 por cada carta de campo que baje al tablero (y que quedará definida en la sección de campos).

Dicho maná será el que necesite para efectuar un ataque.

Por ejemplo, si una carta de personaje tiene 3 puntos de energía pero el jugador tiene solo 1 punto de energía (el equivalente a tener una carta de campo en el tablero), no podrá atacar con ella.

Cartas mágicas:

Se representan con el color morado.

Se divide en dos tipos de cartas mágicas:

- Las cartas que efectuan una acción sobre un jugador
- Las cartas que efectuan una acción sobre una carta de personaje.

La habilidad de esta carta se efectua en el mismo momento en el que se baja al tablero.

Mecánicas del Juego:

Costes de invocación

El número de cartas de campo que tengas en la sección de campo determinará los costes de ataque disponibles.

Invocación de Personajes:

Tienen costos de energía y atributos que determinan su fuerza y resistencia.

Durante el primer turno de partida no podrán atacar.

Cada turno

Se robará una carta del mazo y podrá bajarse al tablero una carta de campo y una carta de personaje o mágica y efectuar un ataque.

Objetivo del Juego:

Reducir los puntos de vida del oponente a cero.

Fases del Turno:

Fase de Robo: El jugador roba una carta al inicio de su turno.

Fase Bajar campo: Baja una carta de campo de la mano al tablero.

Fase Bajar no campo: Baja una carta de personaje o mágica de la mano al tablero.

Fase Efectuar habilidad mágica: En caso de bajar una carta mágica que efectúe una acción sobre otra carta, se elegirá a que carta se le aplicará.

Fase de Ataque: Permite atacar con una carta de personaje del tablero del jugador actual.

Fase de Defensa: En caso de recibir un ataque, permite defenderse con una carta de personaje del tablero del jugador que recibe el ataque.

Diagrama de arquitectura

Los diagramas de arquitectura son el primer recurso al que se accede al iniciar un proyecto porque permiten a los desarrolladores y partes interesadas comprender cómo se conectan y comunican las diferentes partes de la aplicación desde el inicio. Estos diagramas son esenciales, ya que proporcionan una representación visual clara y detallada de la estructura del sistema.

Además de mejorar la comunicación y la documentación, son valiosos para la planificación y la toma de decisiones técnicas. Permiten a los desarrolladores visualizar el flujo de datos, las dependencias y los puntos críticos de la aplicación, lo que ayuda a optimizar el rendimiento y la seguridad. Al ser el primer diagrama elaborado, se asegura que todas las decisiones técnicas posteriores se basen en una comprensión sólida y compartida de la estructura del sistema.

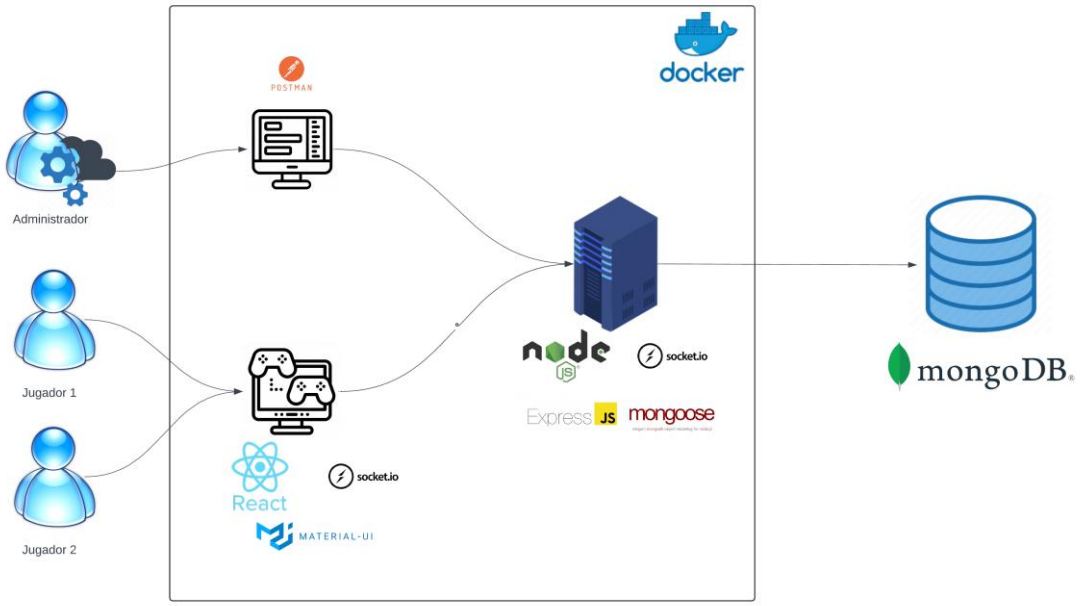
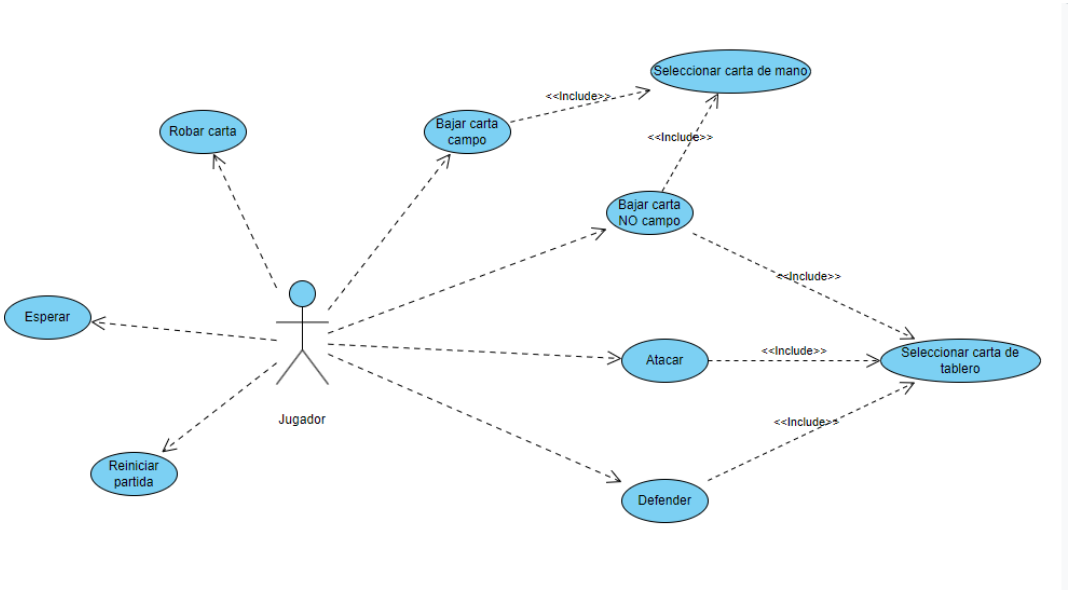


Diagrama de casos de uso

Para representar visualmente estos requisitos y la interacción entre los diferentes componentes del sistema, he creado un diagrama de casos de uso.

Estos diagramas ayudan a identificar las acciones y eventos principales que ocurren en el juego, permitiéndome planificar cómo los jugadores interactuarán con el sistema en diferentes escenarios. Estos diagramas proporcionan una visión clara y estructurada de la funcionalidad del sistema y asegurando que todos los requisitos del usuario sean considerados y atendidos durante el desarrollo del proyecto.



Diagramas de Flujo

En esta sección, presento dos diagramas de flujo que ilustran la funcionalidad del juego. Los diagramas de flujo son herramientas esenciales que me permiten visualizar y planificar la lógica del juego, asegurando que todas las acciones entre diferentes estados se realicen de manera correcta y eficiente. Estos diagramas facilitan la comprensión del comportamiento del sistema tanto a nivel macro, con una visión general del juego, como a nivel micro, detallando las acciones específicas dentro de un turno.

Diagrama de Flujo General del Juego

El diagrama de flujo general del juego proporciona una visión global de cómo se estructura y progresa una partida desde el inicio hasta el final. Comienza con la conexión de dos jugadores y continúa a través de las diversas fases del juego, incluyendo la selección de tripulaciones, la gestión de mazos, y el desarrollo de la partida. Este diagrama es crucial para entender el marco completo en el que se desenvuelven las acciones de los jugadores, desde la configuración inicial hasta la determinación del ganador, asegurando que todas las posibles interacciones y transiciones estén claramente definidas.

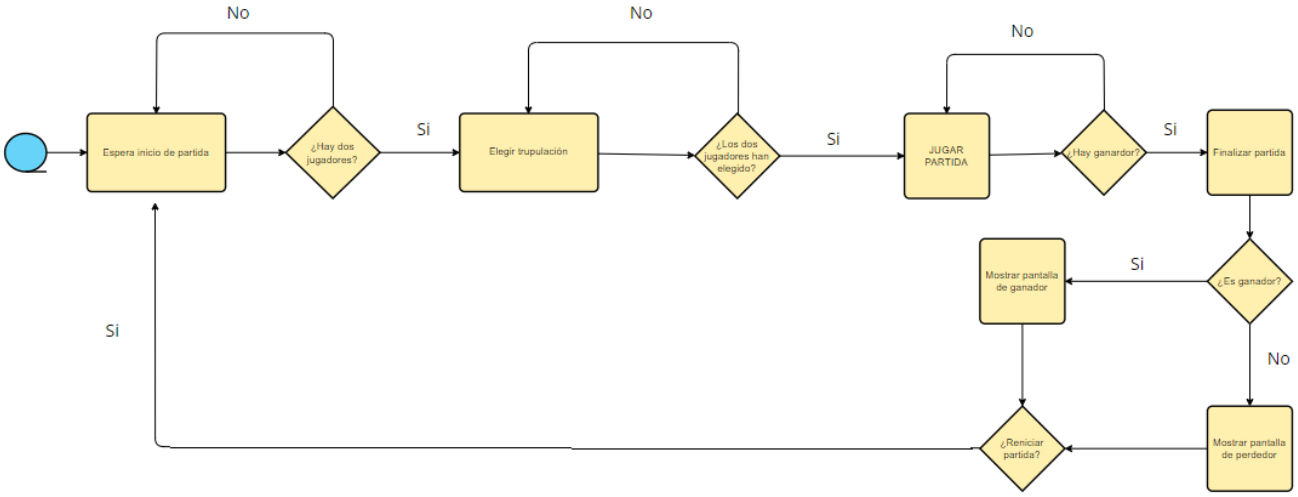
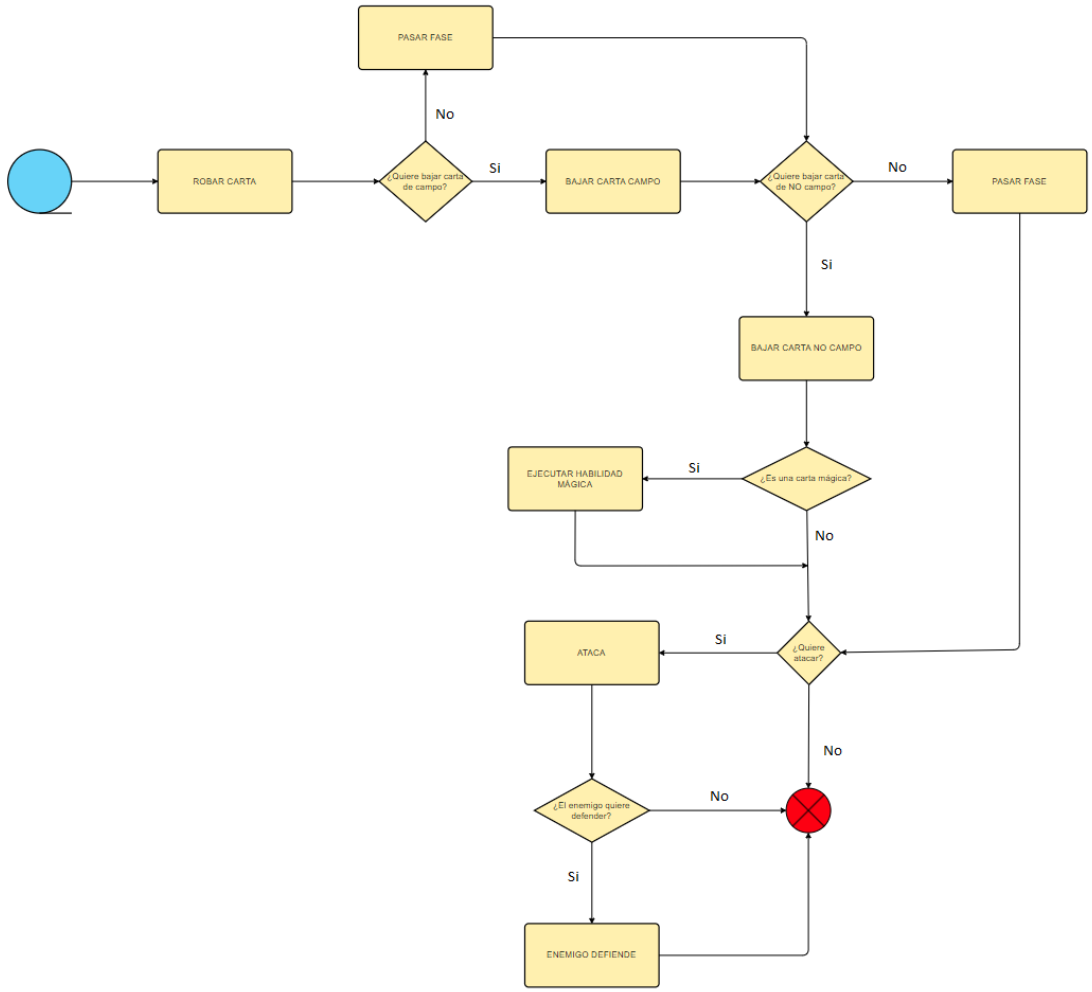


Diagrama de Flujo de un Turno

El diagrama de flujo específico de un turno desglosa las acciones y decisiones que un jugador puede tomar durante su turno. Este flujo incluye fases detalladas como la fase de robo, la fase de bajar cartas de campo y no campo, la fase de habilidad mágica, la fase de ataque y la fase de defensa. Al detallar cada una de estas fases, el diagrama proporciona una guía clara de las opciones y restricciones que enfrenta el jugador en cada etapa de su turno. Este nivel de detalle es fundamental para asegurar que la mecánica del juego se mantenga equilibrada y justa, ofreciendo una experiencia de juego profunda y satisfactoria.



4. Diseño

En el apartado de diseño de mi proyecto, detallo las tecnologías utilizadas y los motivos detrás de su elección, asi como aporto una solución al funcionamiento especificado en la fase de análisis, enfocándome en ofrecer una experiencia de juego accesible y fluida directamente desde el navegador web.

A diferencia de otros juegos similares que pueden requerir la instalación de aplicaciones, mi juego de cartas basado en One Piece se puede jugar sin necesidad de descargas adicionales, lo que facilita el acceso y la participación de los jugadores. Esta accesibilidad se logra mediante el uso de tecnologías modernas y robustas que garantizan un buen rendimiento y una experiencia de usuario intuitiva.

Para el desarrollo del juego, he elegido NodeJS para el backend, ReactJS para el frontend, sockets para la comunicación en tiempo real entre los jugadores y MongoDB para la BBDD que almacenará la información sobre las cartas.

NodeJS es una plataforma de servidor eficiente y escalable que maneja múltiples conexiones simultáneas con gran eficiencia, lo que es esencial para un juego multijugador en línea.

ReactJS, por su parte, proporciona una interfaz de usuario dinámica y reactiva, permitiendo actualizaciones rápidas y fluidas de la vista del juego en respuesta a las acciones de los jugadores.

La utilización de sockets permite una comunicación bidireccional en tiempo real, asegurando que las interacciones entre los jugadores sean inmediatas y sin demoras.

En lo referente al modelo de datos, se complementa la aplicación con una base de datos diseñada con MongoDB para el almacenamiento de las cartas, apoyada con una serie de servicios desarrollados en NodeJS con Express para el mantenimiento (CRUD), lo cual nos permite una mayor facilidad a la hora de administrar y modificar las mismas, asi como la posibilidad de añadir nuevas de manera sencilla.

Comunicación servidor-cliente:

La conexión entre el frontend y el backend se establece mediante sockets utilizando la biblioteca Socket.IO.

Este mecanismo permite una comunicación bidireccional en tiempo real entre el cliente y el servidor.

En el backend, en el archivo ***App.js,*** se crea un servidor con http y se inicializa un socket apuntando al mismo servidor que se mantiene a la espera de interacciones por parte del cliente.

Asimismo, en el frontend, se inicializa un socket en el archivo ***App.tsx*** que se conecta al servidor backend (por defecto ubicado en la raíz, ya que se ejecutan en la misma máquina).

Una vez establecida la conexión, el frontend puede enviar eventos al backend, como solicitudes de acción del usuario o actualizaciones del estado del juego.

Por ejemplo, cuando un jugador realiza una acción como jugar una carta o pasar su turno, se emite un evento desde el frontend al backend a través del socket correspondiente. El backend recibe estos eventos y ejecuta la lógica del juego correspondiente, actualizando el estado del juego según sea necesario.

Además, el backend puede enviar eventos de vuelta al frontend, como actualizaciones del estado del juego o notificaciones de eventos importantes.

Por ejemplo, el inicio de la partida, el estado actual de los jugadores o el final del juego.

El frontend recibe estos eventos y actualiza la interfaz de usuario para reflejar los cambios en el estado del juego.

Esta comunicación continua y bidireccional a través de sockets permite que el frontend y el backend interactúen de manera sincronizada y en tiempo real, proporcionando una experiencia de juego fluida y receptiva para los jugadores.

Uso de constantes

El uso de un diseño de constantes compartidas entre el frontend y el backend es fundamental para mejorar la comunicación y la consistencia en el desarrollo del proyecto.

Al definir constantes para elementos como tipos de carta, fases del juego y tripulaciones, se establece una referencia única que garantiza que todos los componentes del sistema estén sincronizados y utilicen la misma terminología, lo que evita posibles errores de comunicación y asegura que las interacciones entre el frontend y el backend sean coherentes y predecibles. Esto promueve una arquitectura más robusta y coherente, facilitando el desarrollo, la depuración y el mantenimiento del sistema en su conjunto.

Base de datos

He elegido MongoDB como base de datos para este proyecto debido a su flexibilidad y facilidad de uso.

MongoDB es una base de datos NoSQL que permite almacenar datos en un formato de documento JSON, lo que es ideal para aplicaciones que requieren manejar objetos. En el contexto de un juego de cartas, donde cada carta puede tener una variedad de atributos (nombre, tipo, ataque, habilidades, etc.), esta flexibilidad permite una mayor agilidad en la definición y actualización de los esquemas de datos sin necesidad de realizar costosas migraciones de esquema.

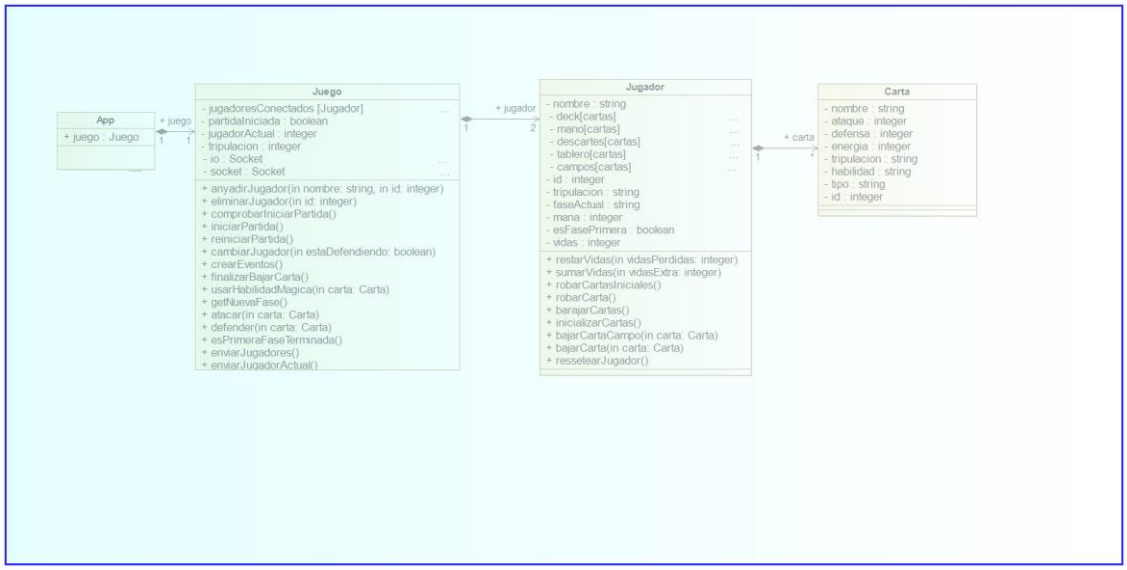
MongoDB se integra perfectamente con Node.js y el ecosistema de JavaScript, que son las tecnologías principales utilizadas en el desarrollo del proyecto. La compatibilidad entre ambas tecnologías facilita el manejo de operaciones asíncronas y el intercambio de datos entre el servidor y la base de datos de manera eficiente.

Además, cuenta con una amplia comunidad de desarrolladores y un robusto soporte técnico. La disponibilidad de documentación extensa, foros comunitarios y servicios de soporte proporciona una red de apoyo confiable que es esencial para resolver problemas técnicos y optimizar el uso de la base de datos.

Estructura del lado del servidor

Para visualizar la estructura en el lado del servidor me he apoyado en el diagrama de clases, el cual ilustra y proporciona una visión clara de cómo las principales clases y relaciones interactúan entre sí: Juego, Jugador y Carta

La clase Juego actúa como el núcleo del sistema, gestionando la lógica de la partida y las interacciones entre los jugadores. *La clase Jugador* representa a cada participante del juego, manteniendo información relevante como sus cartas, vida y estado actual. *La clase Carta* contiene los atributos y comportamientos de las cartas del juego, diferenciadas por tipos y efectos.



Punto de entrada de la aplicación

En el caso del backend, el archivo App.js actúa como el punto de entrada principal para la aplicación. Este fichero configura un servidor utilizando Express y Socket.io para manejar los estados del juego. Se crea una instancia de la clase Juego y se configura el servidor HTTP y Socket.io para escuchar en el puerto 3000.

Cuando un jugador se conecta, se asignan las instancias de io y socket al juego, para que sea capaz de manejar algunos estados, y se mantiene a la espera del evento 'listoParaJugar' que indica que el jugador ha seleccionado una tripulación.

Cuando se lanza este evento, se añade el jugador al Juego y se verifica si hay suficientes jugadores para iniciar la partida. Si se cumplen las condiciones, se inicia el juego. Además, el servidor maneja las desconexiones eliminando al jugador correspondiente y reiniciando la partida si estaba en curso.

Esta configuración garantiza que el juego pueda gestionar dinámicamente las conexiones y desconexiones de los jugadores, manteniendo el flujo y el estado del juego en todo momento.

Además, en este punto de la aplicación se configuran distintos servicios CRUD con los que podemos realizar una administración y mantenimiento de las cartas almacenadas en la BBDD y, en un futuro, proporcionar una interfaz que se alimente de ellos para facilitar su uso.

Clase Jugador

La clase Jugador representa a un jugador en el juego de cartas. Almacena información esencial como el ID del jugador, su nombre, la tripulación a la que pertenece, la fase actual del juego, la cantidad de mana y puntos de vida. También gestiona las cartas que el jugador tiene en su mano, en su mazo (deck), en el tablero (tanto cartas de campo como no de campo), y en su pila de descartes. La clase incluye métodos para manipular estas cartas, como robar cartas del mazo, barajar el mazo, y bajar cartas al tablero. Además, permite la inicialización de cartas específicas según la tripulación seleccionada y la gestión de cambios en los puntos de vida del jugador. También maneja el progreso de las fases del juego y puede resetear el estado del jugador para un nuevo juego.

Clase Carta

La clase Carta representa una carta individual en el juego. Cada carta tiene atributos como su ID, nombre, tipo (por ejemplo, personaje, mágica, campo), valores de ataque, defensa, y energía, así como una descripción de sus habilidades y su afiliación a una tripulación específica. La clase proporciona un modelo para crear instancias de diferentes tipos de cartas que los jugadores pueden utilizar durante el juego. Las cartas pueden tener efectos específicos y atributos que influyen en cómo se juegan y cómo interactúan con otras cartas en el tablero. La clase Carta es fundamental para definir las propiedades y comportamientos de cada carta utilizada en el juego, permitiendo la personalización y variabilidad en las estrategias de los jugadores.

Clase Juego

La clase Juego sirve como el eje central del sistema de juego, coordinando la partida entre múltiples jugadores y las reglas del juego, asegurando que las acciones se realicen correctamente y en el orden adecuado.

Se encarga de gestionar la partida en curso, incluyendo acciones como:

- La conexión y desconexión de jugadores
- El inicio y reinicio de la partida
- La ejecución de eventos durante el juego (como robar cartas, pasar turnos, bajar cartas al campo y realizar acciones específicas de las cartas).
- La lógica detrás de cada uno de los eventos anteriores
- El flujo completo de la partida (desde el inicio hasta la determinación de un ganador)

Estructura del lado del cliente

Punto de entrada de la aplicación

En el caso del frontend, el archivo App.tsx actúa como el punto de entrada principal para la aplicación.

Este fichero se encarga de definir y gestionar las rutas a las distintas pantallas del juego:

- Principal
- Juego
- Ganador/Perdedor
- Información sobre el juego

Además, establece un Layout genérico para todas las pantallas que es el que se encarga de mostrar la imagen del fondo.

Principal

El componente Principal es el punto de acceso del usuario una vez accede a la aplicación desde el navegador.

En esta pantalla se muestra:

- Un logo con el nombre de la aplicación
- Un botón para acceder a la pantalla de información sobre el juego
- Dos botones que darán al usuario la oportunidad de elegir con que tripulación desean jugar, pudiendo tener ambos la misma. Dependiendo de la elección, el mazo del jugador variará entre un tipo de cartas u otras.

Información

Este componente muestra una descripción del juego y de sus reglas para que el usuario pueda conocerlas antes de comenzar la partida.

Componente Juego

El componente Juego gestiona la lógica y la interfaz de la partida en curso.

Utiliza varios estados (useState) para mantener actualizada en todo momento la información del jugador, del enemigo, el turno actual y otros datos necesarios para el flujo de la partida.

A través del hook useEffect, se mantiene a la escucha de eventos provenientes del servidor mediante sockets, actualizando los estados del juego en respuesta a distintas acciones como: el inicio de la partida, cambios de turno y habilidades especiales. A través de estos eventos y cambios en los estados, el componente se encarga de actualizar la interfaz del usuario para que el jugador sea consciente en todo momento del punto de la partida en el que se encuentra y poder hacer sus estrategias en base a él.

Además, el componente manejará la navegación según el resultado del juego una vez la partida finalice (ganar o perder).

En cuanto a la interfaz del componente, se estructura con el uso de componentes y estilos tanto de la librería Material-UI como algunos personalizados, organizando la información del jugador, el enemigo y sus respectivas cartas.

Funciones como robarCarta, pasarTurno, bajarCarta, entre otras, emiten eventos al servidor para comunicar las acciones del jugador y que este se encargue de ejecutar la lógica necesaria para el correcto funcionamiento del juego.

Componentes Ganador y perdedor

Una vez finalizada la partida, cuando el servidor nos envíe un evento de tipo 'comprobarFinPartida', el jugador será redireccionado a una de estas dos pantallas, en las que se le informará si ha ganado o perdido y se le mostrará un botón con el que tendrá la opción de volver a iniciar una nueva partida.

Despliegue de la aplicación

Para asegurar una experiencia de instalación y uso fluida para los usuarios de mi aplicación, he priorizado la creación de un archivo README exhaustivo y detallado. Este archivo no solo proporciona instrucciones paso a paso para la instalación y configuración de la aplicación, sino que también incluye soluciones comunes a posibles problemas que los usuarios podrían encontrar. La estructura clara y la inclusión de capturas de pantalla y ejemplos de comandos garantizan que incluso los usuarios con menos experiencia técnica puedan seguir los pasos sin dificultades.

No obstante, reconozco que, en entornos de desarrollo y producción modernos, la simple instalación local puede no ser suficiente. Por ello, he adoptado la tecnología de contenedores mediante Docker para desplegar mi aplicación.

Docker me permite encapsular la aplicación y todas sus dependencias en contenedores ligeros y portátiles, asegurando que se ejecute de manera consistente en cualquier entorno.

Para facilitar el despliegue con Docker, he desarrollado archivos Dockerfile tanto para el frontend como para el backend de mi aplicación. Estos archivos definen las instrucciones necesarias para construir las imágenes de Docker, incluyendo la instalación de las dependencias, la configuración del entorno y la puesta en marcha de los servicios necesarios. Además, he creado un archivo docker-compose.yml que gestiona el despliegue de múltiples contenedores.

Esta configuración permite a los usuarios desplegar la aplicación completa con un solo comando, simplificando enormemente el proceso.

Sin embargo, debido a diferentes problemas técnicos que no he podido resolver, no he logrado un despliegue exitoso de la base de datos utilizando Docker. Como solución temporal, he implementado un sistema de carga estática de datos que se activa cuando la base de datos no está disponible. Este sistema permite que la aplicación continúe funcionando con datos precargados, asegurando que los usuarios puedan seguir interactuando con la aplicación mientras trabajo en una solución más permanente.

5. Resultados

Después de completar la implementación de la aplicación, es fundamental llevar a cabo una evaluación tanto del proceso como de los resultados obtenidos. Durante esta revisión, se verificará que todas las funcionalidades planificadas y deseadas en la fase de diseño se hayan implementado correctamente.

Es importante destacar que todas las características inicialmente previstas han sido incorporadas con éxito en la aplicación. No obstante, se podrían añadir funcionalidades adicionales que enriquezcan la experiencia del usuario o que optimicen el rendimiento general del sistema.

Algunos puntos de mejora o funcionalidades extra que se podrían implementar, serían:

- Aunque la interfaz actual es funcional, hay margen para mejorarla en términos de diseño y usabilidad. Una interfaz más intuitiva y visualmente atractiva podría mejorar significativamente la experiencia del usuario.
- Actualmente, el sistema solo permite una partida a la vez entre dos jugadores. Implementar soporte para múltiples partidas simultáneas permitiría a más jugadores disfrutar del juego sin tiempos de espera, mejorando la accesibilidad y la satisfacción del usuario.
- Por otra parte, aunque los personajes principales están bien representados con sus atributos de ataque, defensa y energía, no he implementado la mecánica de evolución de personajes. Esta característica, inspirada en juegos como Legends of Runeterra, añadiría una capa adicional de estrategia y profundidad al juego, permitiendo que los personajes evolucionen y mejoren sus habilidades a lo largo de la partida.
- Las habilidades únicas que cada personaje podría tener, inspiradas en sus habilidades en One Piece, no están implementadas. Estas habilidades proporcionarían mayor diversidad y opciones tácticas durante las partidas, haciendo que cada personaje se sienta único y valioso.
- Actualmente, los servicios CRUD para añadir, modificar y borrar cartas están implementados, pero se carece de una interfaz frontend que facilite este trabajo. Desarrollar una interfaz de administración intuitiva permitiría a los administradores del juego gestionar las cartas de manera más eficiente. Aunque no afecte directamente al usuario final, mejorará significativamente la mantenibilidad y operatividad del sistema.
- Implementar la opción para que los mazos de cartas sean personalizables permitiría a los jugadores elegir qué cartas incluir en su mazo, incluso mezclando cartas de distintas tripulaciones. Esta funcionalidad añadiría una capa adicional de personalización y estrategia, permitiendo a los jugadores crear mazos únicos y experimentar con diferentes combinaciones y tácticas.
- La implementación de realizar varios ataques en un mismo turno aportaría un nivel extra de complejidad al juego.

En resumen, aunque se ha logrado cumplir con todos los objetivos establecidos en la fase de diseño, la evaluación post-implementación revela oportunidades para futuras mejoras y ampliaciones que podrían potenciar aún más la funcionalidad y el rendimiento de la aplicación.

Pantallas de la aplicación

A continuación, se muestran algunas de las pantallas diseñadas para el juego de cartas basado en One Piece.

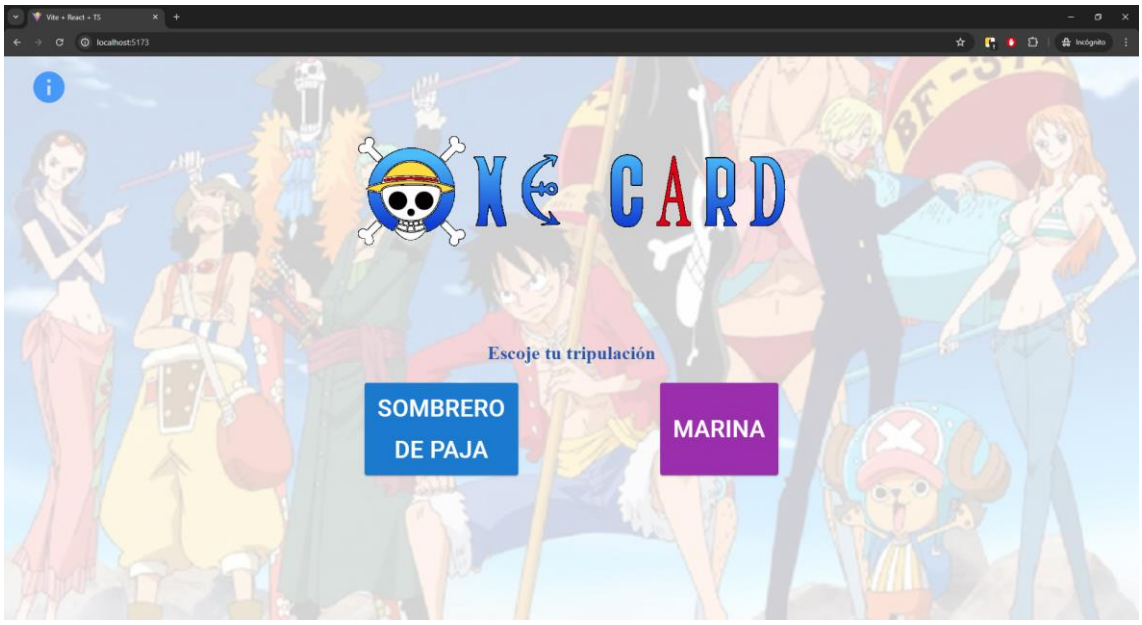


Ilustración 1- Pantalla de inicio

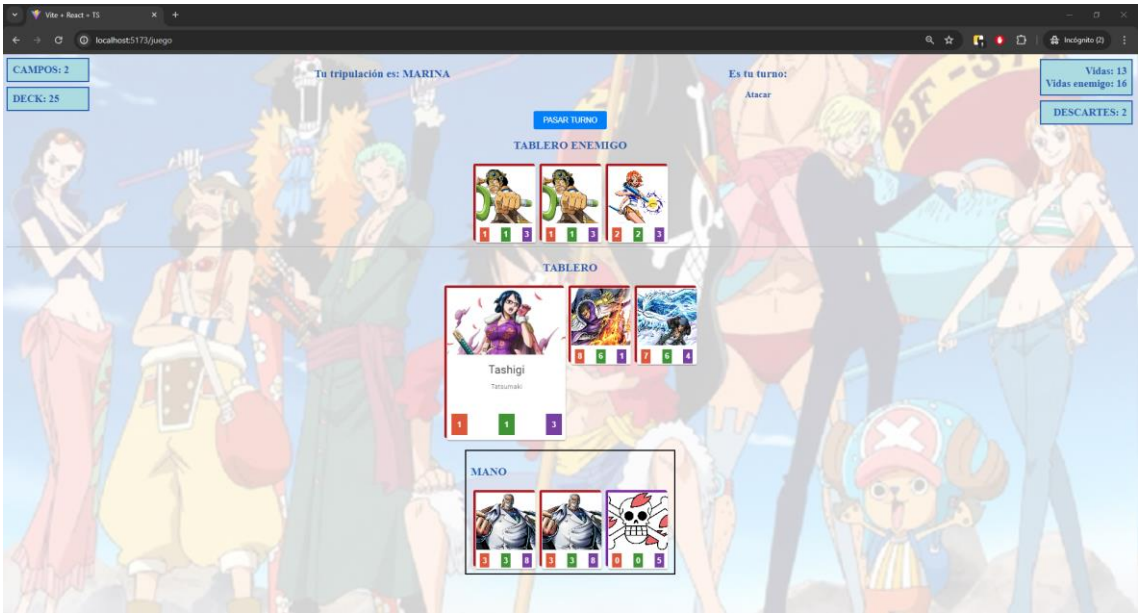


Ilustración 2- Pantalla de juego durante el ataque

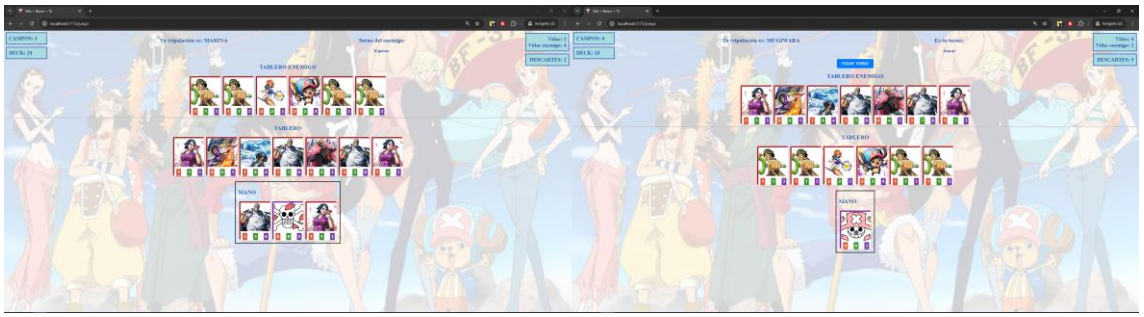


Ilustración 3- Pantalla de juego - Vista de los dos jugadores a la vez

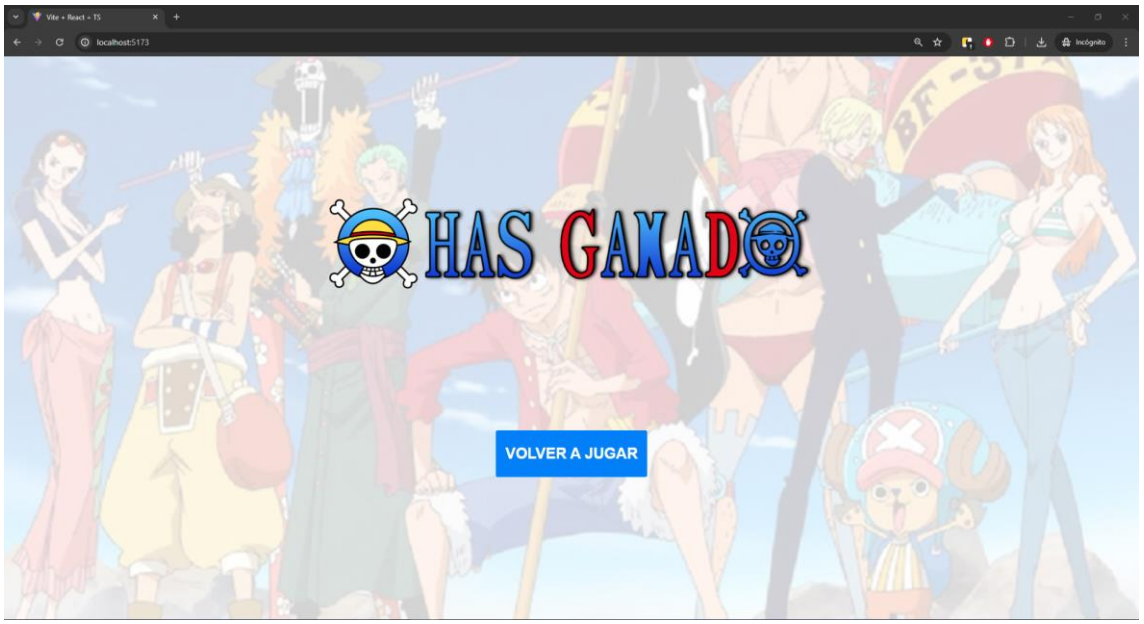


Ilustración 4- Pantalla de jugador ganador

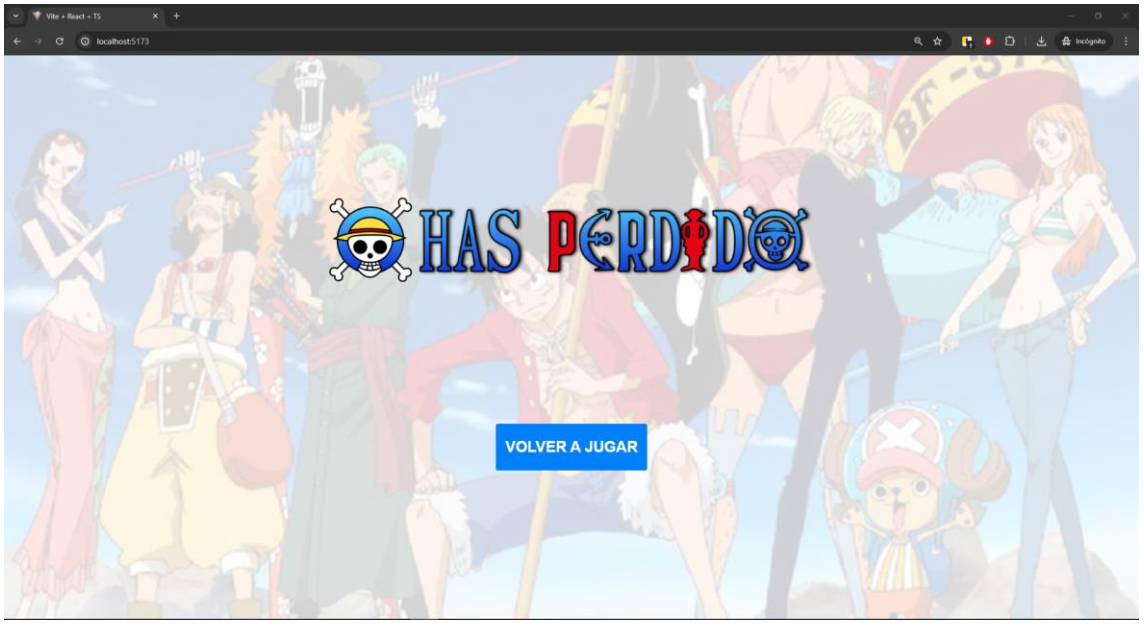


Ilustración 5 - Pantalla jugador perdedor

6. Conclusiones

El desarrollo de este juego de cartas basado en el universo de One Piece ha sido una experiencia enriquecedora y educativa. El tiempo de desarrollo ha sido conforme a lo esperado, permitiendo la implementación de la mayoría de las funcionalidades previstas.

Desde el inicio hasta la implementación final, cada componente del juego se ha desarrollado de manera eficiente, asegurando que se cumplieran los objetivos establecidos en la fase de diseño. La implementación ha permitido crear una experiencia de juego única, donde los jugadores pueden sumergirse en batallas con sus personajes favoritos de One Piece.

Los objetivos del proyecto se han cumplido con éxito. El juego proporciona una experiencia estratégica, permitiendo a los jugadores elegir entre dos tripulaciones icónicas de One Piece y utilizar cartas de personaje, campo y mágicas para planificar y ejecutar sus estrategias. Las mecánicas del juego, han sido implementadas de manera funcional, ofreciendo una base sólida para futuras expansiones y mejoras.

Durante el desarrollo, se localizaron algunos problemas técnicos, especialmente relacionados con el uso de APIs para la comunicación entre el frontend y el backend. Este enfoque inicial resultó en una gran sobrecarga, afectando el rendimiento del juego ya que cada interacción entre los jugadores requería múltiples llamadas a la API. Sin embargo, la adopción de Socket.io para el tratamiento de la información en forma de eventos en tiempo real solucionó eficazmente este problema. Socket.io permitió establecer una comunicación bidireccional constante entre el cliente y el servidor, reduciendo significativamente la latencia y mejorando la fluidez del juego. Esta solución no solo mejoró el rendimiento, sino que también enriqueció la experiencia del usuario, asegurando que las acciones de los jugadores se reflejen instantáneamente en el tablero del oponente.

Otro gran problema localizado fue la incapacidad de lograr un despliegue exitoso de la base de datos con Docker. Debido a la falta de tiempo y a una baja prioridad respecto a otros problemas funcionales se implementó un sistema de carga predefinida de cartas cuando la base de datos no se encuentra disponible, pero dicha solución solo aporta un resultado temporal ya que implica también la imposibilidad de acceder a un mantenimiento de las cartas, lo cual genera una disminución de la vida útil de la aplicación.

La elección de tecnologías ha sido adecuada y ha contribuido significativamente al éxito del proyecto. NodeJS ha proporcionado un entorno de desarrollo rápido y eficiente para el backend, mientras que ReactJS ha facilitado la creación de una interfaz de usuario dinámica y responsiva. Socket.io ha sido fundamental para implementar la comunicación en tiempo real, asegurando una experiencia de juego fluida. Estas tecnologías no solo han sido efectivas para cumplir con los objetivos actuales del proyecto, sino que también ofrecen una base sólida para futuras mejoras y expansiones, permitiendo que el juego evolucione y se adapte a las necesidades y expectativas de los usuarios.

Anexo 1 – Proyecto de empresa

Macro entorno

El mercado global de juegos de cartas coleccionables está valorado en aproximadamente 2.000 millones de dólares en 2023, con un crecimiento anual esperado del 6% hasta 2027.

Entre las tendencias más notables del mercado se encuentra la creciente digitalización, que facilita el acceso y la jugabilidad en línea.

Los juegos basados en franquicias populares de anime y manga, como One Piece, están ganando popularidad debido a su base de fans establecida.

La competencia en este mercado es muy alta. Magic sigue siendo el líder del mercado con una base de jugadores global y torneos frecuentes. Yu-Gi-Oh! mantiene una popularidad significativa tanto en formato físico como digital, y Pokémon TCG sigue siendo muy popular, especialmente entre los jóvenes, además de otros títulos muy conocidos como Hearthstone y Legends of Runeterra, entre otros.

En términos de factores económicos, aunque las fluctuaciones pueden afectar el gasto en entretenimiento, la pandemia ha incrementado el consumo de juegos digitales.

La expansión del acceso a Internet y la mejora de las velocidades de conexión también están facilitando el crecimiento del mercado de juegos en línea.

Es crucial cumplir con las regulaciones relacionadas con los juegos de azar y el comercio en línea, además de las normativas de protección de datos como el GDPR en Europa.

Micro entorno

Mi mercado objetivo se segmenta en fans de One Piece, jugadores de juegos de cartas coleccionables y jugadores de juegos en línea.

El perfil típico del consumidor de mi juego se compone principalmente de personas de 15 a 35 años, mayoritariamente hombres, aunque el interés entre mujeres está en aumento.

Estos consumidores están interesados en anime, manga, juegos de cartas y la cultura japonesa, y están dispuestos a gastar en productos relacionados con sus intereses, incluyendo microtransacciones dentro de los juegos.

La demanda potencial es alta debido a la popularidad global de One Piece, que cuenta con millones de seguidores. Además, los juegos de cartas combinan elementos estratégicos y coleccionables que atraen a una audiencia amplia y dedicada.

Mis competidores directos utilizan estrategias de marketing intensivas en redes sociales, colaboraciones con influencers y campañas de crowdfunding para financiar el desarrollo inicial y generar interés antes del lanzamiento.

Estas tácticas son efectivas para atraer y retener a una base de jugadores leal.

Análisis DAFO

Debilidades:

Cuento con recursos financieros limitados y dependo de la financiación externa para el desarrollo y marketing.

También tengo experiencia limitada en el desarrollo de juegos de cartas y dependo de la obtención y mantenimiento de los derechos de uso de la franquicia One Piece.

Amenazas:

Enfrento una competencia intensa de competidores bien establecidos con bases de jugadores leales. Además, existen riesgos legales relacionados con la propiedad intelectual y el uso de la franquicia.

Fortalezas:

Usar la franquicia One Piece me proporciona una base de fans global y leal. Implementar tecnología moderna como Node.js, React y sockets permite una experiencia de juego en tiempo real e interactiva.

Además, estoy diseñando una experiencia de usuario atractiva y fácil de usar que mantendrá a los jugadores comprometidos.

Oportunidades:

El mercado de juegos de cartas en línea está en expansión, ofreciendo oportunidades de monetización digital. Existen posibilidades de alianzas con otras marcas y productos relacionados con One Piece.

Plan Financiero

Plan de Tesorería

	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic	Totales
Entradas													
Aportaciones de socios	2.000,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	2.000,00 €
Préstamos	5.000,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	5.000,00 €
Ventas	0,00 €	0,00 €	500,00 €	1.000,00 €	1.500,00 €	2.000,00 €	2.500,00 €	3.000,00 €	3.500,00 €	4.000,00 €	4.500,00 €	5.000,00 €	29.500,00 €
Total entradas	7.000,00 €	0,00 €	500,00 €	1.000,00 €	1.500,00 €	2.000,00 €	2.500,00 €	3.000,00 €	3.500,00 €	4.000,00 €	4.500,00 €	5.000,00 €	42.000,00 €
Salidas													
Gastos generales	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	200,00 €	2.400,00 €
Sueldos y seguridad social	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	1.000,00 €	12.000,00 €
Gastos de distribución	0,00 €	200,00 €	300,00 €	400,00 €	500,00 €	600,00 €	700,00 €	800,00 €	900,00 €	1.000,00 €	1.100,00 €	1.200,00 €	7.700,00 €
Devolución préstamo + intereses	0,00 €	0,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	150,00 €	1.800,00 €
Total salidas	1.200,00 €	1.400,00 €	1.650,00 €	1.750,00 €	1.850,00 €	1.950,00 €	2.050,00 €	2.150,00 €	2.250,00 €	2.350,00 €	2.450,00 €	2.550,00 €	23.550,00 €
Resumen													
Entradas menos salidas	5.800,00 €	-1.400,00 €	-1.150,00 €	-750,00 €	-350,00 €	50,00 €	450,00 €	850,00 €	1.250,00 €	1.650,00 €	2.050,00 €	2.450,00 €	18.400,00 €
Saldo en el banco	5.800,00 €	4.400,00 €	3.250,00 €	2.500,00 €	2.150,00 €	2.200,00 €	2.650,00 €	3.500,00 €	4.750,00 €	6.400,00 €	8.450,00 €	10.900,00 €	
Saldo crédito	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €	0,00 €

Cuenta de Resultados

Ingresos de explotación	
Ventas	29.500,00 €
Total	29.500,00 €

Gastos de explotación	
Gastos generales	2.400,00 €
Gastos de distribución	7.700,00 €
Total	10.100,00 €

		TOTAL
Resultado explotación	29.500€ - 10.100€	19.400,00 €
Resultado financiero	0€ - 412,50€	-412,50 €
Resultado antes de impuestos		18.987,50 €
Resultado neto	25%	14.240,63 €

Ingresos financieros	
Total	0,00 €

Gastos financieros	
Intereses del banco	412,50 €
Total	412,50 €

Balance

Activo		Pasivo	
No corriente		**No corriente**	
		Prestamo banco	5.000,00 €
Corriente		**Corriente**	
Saldo banco	10.900,00 €		
		Patrimonio neto	
		Capital	2.000,00 €
		Cuenta de resultado	14.240,63 €
Total activo	10.900,00 €	Total pasivo	21.240,63 €

Interpretación de los resultados

Al analizar el balance de previsión del primer año, observamos que el total de activos es inferior al total de pasivos, lo que refleja un déficit significativo y pérdidas iniciales. Este escenario es típico para una empresa indie en sus etapas iniciales debido a los altos costos de desarrollo y marketing frente a unos ingresos tardíos.

Sin embargo, se prevé que las inversiones realizadas en el primer año comenzarán a dar sus frutos en los años siguientes, con un aumento en las ventas y una estabilización de los gastos, lo que permitirá a la empresa alcanzar la rentabilidad y generar ganancias sostenibles a largo plazo.

Referencias

Durante la elaboración de la memoria técnica y el desarrollo del proyecto, se consultaron diversas fuentes de información que proporcionaron conocimientos y orientación sobre las tecnologías utilizadas, las mecánicas de juegos similares y las mejores prácticas de desarrollo de juegos web en tiempo real.

A continuación, se detallan algunas de las principales fuentes de referencia:

- Webs oficiales de otros juegos de referencia (Magic, Legends of Runeterra, Hearthstone…)
- Documentación Oficial de Node.js: Recurso fundamental para comprender el funcionamiento de Node.js y sus capacidades para construir aplicaciones del lado del servidor.
- Documentación Oficial de React.js: Referencia esencial para aprender sobre los conceptos fundamentales de React.js, como componentes, estado, props y ciclos de vida, que fueron aplicados en el desarrollo del frontend del juego.
- Documentación de Socket.io: Guía completa sobre el uso de Socket.io para la comunicación en tiempo real entre clientes y servidor mediante WebSockets.
- Documentación de MaterialUI: Guía de componentes de la librería MaterialUI, con los cuales se ha diseñado gran parte de la interfaz del juego.
- Stack Overflow: Plataforma de preguntas y respuestas que proporcionó soluciones a problemas específicos relacionados el desarrollo de la aplicación.
- Tutoriales en línea y blogs especializados: Recursos educativos en forma de tutoriales y blogs que ofrecieron ejemplos prácticos y casos de uso relacionados con el desarrollo de juegos web multiplataforma, con Node.js, React.js y otras tecnologías utilizadas.

Además, se hizo uso de las siguientes herramientas tanto para el desarrollo del proyecto como para la realización de esta memoria:

- **lucid.app:** Para la realización del diagrama de arquitectura.
- **Visual Paradigm:** Para la realización del diagrama de casos de uso.
- **Modelio:** Para la realización del diagrama de clases .
- **Visual Studio Code:** Como editor de código para desarrollar la aplicación.
- **MongoDB:** Como base de datos para el almacenamiento de las cartas.
- **Docker Desktop**: Para gestionar los despliegues de la aplicación con Docker.
- **Word y Excel:** Para la redacción de la memoria
- **Postman:** Para el uso de los servicios creados para manipular la BBDD