UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo e implementación del backend y comportamiento de los enemigos en un videojuego

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Payá Poveda, Juan

Tutor/a: Abad Cerdá, Francisco José

CURSO ACADÉMICO: 2022/2023

Agradecimientos

Gracias a mis padres: Consuelo y Juan Salvador, por guiarme en este largo camino llamado vida.

A mi hermana Victoria y a toda la familia por su apoyo.

Gracias a mi novia Laura por estar a mi lado brindándome su confianza, apoyo y cariño siempre que lo he necesitado.

Gracias a mi tutor Paco Abad por su apoyo y dedicación durante todos estos meses de duro trabajo, ya que ha sido fundamental para que el proyecto salga adelante.

Gracias a mi compañero Sergio por su ayuda para desarrollar este videojuego.

Resumen

El proyecto consiste en el desarrollo de una IA que se encargará de implementar el comportamiento complejo de los enemigos controlados por la máquina en videojuego y de la creación de un back-end que se encargará de gestionar la lógica del juego.

Se basa en un juego de tipo rol estratégico inspirado en la saga de juegos Fire Emblem que se ha desarrollado de forma previa en Unity para la asignatura de Diseño de Videojuegos en 2D. El juego consiste en una serie de batallas en mapas configurados de forma similar a un tablero de ajedrez por el que se pueden mover las unidades.

El back-end contendrá la lógica necesaria para desarrollar juegos del mismo tipo independientemente del entorno en el que se quiera construir el juego, haciendo que sea fácilmente migrable entre motores de desarrollo.

En cuanto a la IA el objetivo es crear una máquina de estados para dotar de un comportamiento complejo a los enemigos controlados por la máquina.

El objetivo de este trabajo es mostrar los conocimientos adquiridos en la rama de Ingeniería del Software para la realización de proyectos informáticos, siendo un videojuego el tipo de proyecto que se ha escogido para ello. Por ello el proyecto se desarrollará siguiendo las metodologías y estándares estudiados durante el Grado.

Palabras clave: backend;RPG in turns;IA;Unity.

Abstract

The project consists of the development of an AI that will be in charge of implementing the complex behaviour of the enemies controlled by the machine in the videogame and the creation of a back-end that will be in charge of managing the logic of the game.

It is based on a strategic role-playing game inspired by the Fire Emblem game saga that has been previously developed in Unity for the 2D Game Design course. The game consists of a series of battles on maps configured in a similar way to a chessboard where units can move around.

The back-end will contain the logic necessary to develop games of the same type regardless of the environment in which you want to build the game, making it easily migratable between development engines.

As for the AI the goal is to create a state machine to provide complex behaviour to the enemies controlled by the machine.

The aim of this work is to show the knowledge acquired in the branch of Software Engineering for the realization of computer projects, being a video game the type of project that has been chosen for it. Therefore, the project will be developed following the methodologies and standards studied during the Degree.

Key words: back-end;RPG por turnos;IA;Unity.

Índice General

Índice de figuras

Índice de Tablas

Capítulo 1

Introducción

El proyecto software a desarrollar para este trabajo de fin de grado es un videojuego. Los juegos tal y como los conocemos nacieron en la década de los 70s con títulos como "Pong" considerado el producto que inició la industria de los videojuegos. Desde entonces los videojuegos han avanzado enormemente. La década de los 80s trajo la popularidad de los arcades y el icónico "Super Mario Bros." En 1985 la introducción del CD-ROM en los años 90 permitió narrativas más profundas, evidenciado por "Final Fantasy VII" en 1997. Y desde entonces los videojuegos no han hecho más que crecer hasta el día de hoy.

Debido a la gran diversidad de juegos que existen, estos se clasifican en diversos géneros. Realm Of Warriors pertenece al género de rol táctico (TRPG), siendo este una mezcla de los géneros de rol (RPG) y estrategia. Su característica fundamental reside en su sistema de combate, que fusiona elementos característicos de los juegos de rol, tales como una trama y la evolución de los protagonistas, con la táctica. El usuario dirige un conjunto extenso de individuos, todos ellos con capacidades y características únicas, y se le encomienda la tarea de desplazarlos de manera intercalada en un campo de lucha repleto de adversarios, empleando las estrategias y disposiciones que estime convenientes.

Realm Of Warriors busca ser una aproximación amigable a este género de videojuegos imitando varios de los aspectos que los caracterizan, pero.

eliminando mecánicas cuya dificultad implica una experiencia frustrante para los jugadores que acaban de iniciarse

1.1 Motivación

De entre los diferentes productos software, los videojuegos es uno de los que más atractivo tienen para los consumidores. En general, se podría decir que los videojuegos son uno de los primeros contactos que muchos de los usuarios tienen con la informática, dada su popularidad actual.

Durante los últimos años se han lanzado al mercado nuevos juegos del género de rol táctico, haciendo que este se hiciera conocido fuera de Japón. No obstante, todos estos videojuegos suelen tener una complejidad muy elevada por la gran cantidad de mecánicas que hay en ellos y la dificultad del juego suele ser demasiado elevada para aquellos jugadores que nunca los han jugado y buscan iniciarse.

El tipo de jugabilidad de este videojuego está basado en juegos de mesa como el Estratego y el ajedrez. Estos juegos buscan emular una batalla mediante un tablero y piezas que se mueven por él, siguiendo la estrategia que decida el jugador que actúa como comandante de las tropas. El ajedrez, tal como se conoce actualmente, surgió en Europa durante el siglo XV,como evolución del juego persa Shatranj, que a su vez surgió a partir del más antiguo Chaturanga, que se practicaba en la India en el siglo VI [1]. Se puede ver un tablero de ajedrez en la Figura 1.

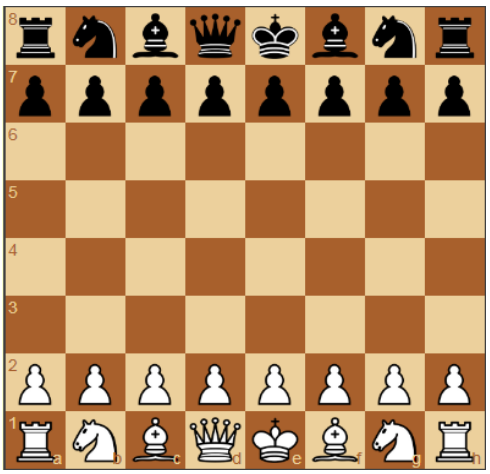


Figura 1. Tablero de ajedrez

No obstante, la idea principal para este juego surgió de la saga Fire Emblem, cuyo primer juego fue de los pioneros en el género, ya que para poder completar los diferentes retos propuestos en el videojuego es necesario hacer uso del ingenio y la estrategia a la hora de manejar a las tropas de las que se dispone para conseguir superarlos.

Este juego inspiró Realm Of Warriors, videojuego en el que se basa mi TFG. Nuestro videojuego adapta Fire Emblem de forma más amigable para aquellos que quieran iniciarse en el género, ya que contiene la esencia de los juegos de rol táctico, pero elimina mecánicas que pueden hacer de esta una experiencia frustrante para jugadores novatos.

1.2 Objetivos

A continuación, se enumeran los distintos objetivos de este trabajo de final de grado.

- Desarrollar un proyecto software a partir de un videojuego desarrollado para la asignatura de Videojuegos en 2D, empezando por el análisis de requisitos, diseño y planificación del proyecto, la implementación de la solución, el control de versiones y finalizando con el mantenimiento o cierre del proyecto.

- Aprender a desarrollar un proyecto utilizando las metodologías ágiles.

- Aprender a manejar las tecnologías de Swagger y HTTP para la comunicación entre aplicaciones y la base de datos.

1.3 Metodología

En este proyecto se han empleado metodologías ágiles para mantener un ritmo de desarrollo constante y controlado. Durante las fases iniciales del proyecto se analizaron los requisitos que debía cumplir el proyecto. Una vez se tenían claras las facetas que debían ser cubiertas, se procedió a establecer las distintas tareas que conformaban el producto final que se deseaba obtener. Dichas tareas estaban sujetas a modificaciones durante el proyecto, ya que, al ser empleada una metodología ágil, el proyecto podría ver alterado su flujo de desarrollo para adecuarse a los sprints establecidos.

Un sprint, en este tipo de metodologías, es un periodo de tiempo en el que se asignan tareas a desarrollar. En cada sprint, se pueden consultar las tareas para ver qué tareas están en proceso, cuáles finalizadas y cuáles están en espera de iniciarse. De esta forma, en todo momento se tiene un control sobre el trabajo realizado y las tareas pendientes.

El proyecto se realizó en sprints de dos semanas, durante los cuales se realizaban las tareas establecidas para ese periodo de tiempo.

1.4 Estructura de la memoria

La memoria consta de varias secciones. Comienza con una introducción, que es la parte actual del documento. A continuación, se presenta el estado del arte, donde se exploran diferentes productos similares al desarrollado en este proyecto y se establece una comparación entre sus características y las de nuestro videojuego. También se analizan diversas alternativas consideradas para implementar varias mecánicas del juego, junto con las razones que respaldaron la elección de las implementaciones finales.

Seguidamente, se realiza un análisis del problema a resolver y del proyecto a desarrollar, lo que implica recopilar los requisitos tanto funcionales como no funcionales. Posteriormente, se procede al diseño del videojuego y sus componentes, y se proporciona información sobre las tecnologías empleadas en este proyecto. El siguiente paso es el desarrollo del proyecto, que describe cómo se ha llevado a cabo y detalla su implementación.

El documento continúa con un análisis de los resultados de las pruebas realizadas con usuarios, las pruebas de funcionamiento aplicadas al proyecto y las conclusiones obtenidas. Finalmente, se incluye una bibliografía con las referencias utilizadas en la redacción de esta memoria, así como varios documentos anexos que comprenden el Game Design Document (GDD) del videojuego y el documento sobre los Objetivos de Desarrollo Sostenible (ODS). Además, se proporciona un glosario con todas las abreviaciones empleadas en el documento.

1.5 Colaboraciones

La primera versión del juego fue desarrollada junto con Javier Calatayud Ferre y Sergio Pérez Gascon, compañeros de la asignatura de desarrollo de videojuegos 2D.

Sergio Pérez Gascon se ha encargado de adaptar un modo multijugador para el juego, la información sobre su proyecto estará en RiuNet.

Capítulo 2

Estado del arte

En este capítulo, se abordará el contexto tecnológico de nuestro videojuego. Se llevará a cabo un análisis de los juegos presentes en el mercado que comparten similitudes en términos de características y género con el objetivo de identificar los elementos que distinguen a Realm Of Warriors. Además, se investigará el origen de las mecánicas clave que sustentan este juego. Por último, se realizará una evaluación de las posibles alternativas a estas mecánicas y se ofrecerán razones que respalden la elección de las mecánicas finales.

Esta sección permitirá una comprensión más profunda de cómo nuestro videojuego se posiciona en el mercado y cómo se diferencia de otros títulos similares, así como proporcionará información valiosa sobre las decisiones detrás de las mecánicas implementadas.

2.1 Juegos de rol táctico

El rol táctico es un subgénero inspirado en juegos de mesa que se centra especialmente en la estrategia posicional en los combates por turnos. También se les conoce como TRPG por sus siglas en inglés (tactical role-playing game) y se desarrollan en escenarios con cuadrículas como si fuera una partida de ajedrez. Eso sí, cada personaje tiene su propio rango de acción, motivo por el que es importante mantener lejos del combate a las unidades más débiles y acercar a las más fuertes [2].

A lo largo de los años se han publicado decenas de juegos RPG de estrategia. Posiblemente uno de los que ha ganado mayor popularidad sea Fire Emblem.

A continuación, se enumeran las características principales que definen el género de juegos de rol tácticos:

Sistema de Batalla Tácticas: El núcleo del juego gira en torno a combates estratégicos por turnos en mapas hexagonales o cuadrados. Los jugadores deben

planificar cuidadosamente sus movimientos, posiciones y acciones para lograr la victoria

Personajes Jugables: Los personajes únicos son un elemento crucial en los juegos de rol táctico. Cada personaje tiene habilidades, clases, estadísticas y arquetipos específicos que influyen en su papel en el campo de batalla.

Permadeath: Una característica distintiva de muchos juegos de rol táctico es la opción de permadeath, donde los personajes caídos en batalla mueren permanentemente. Esto agrega un elemento emocional y estratégico a las decisiones del jugador.

Desarrollo de Personajes: Los jugadores pueden mejorar a sus personajes a lo largo del juego, ya sea mediante la asignación de puntos de experiencia, cambios de clase, mejoras de equipo o desarrollo de habilidades.

Historia y Narrativa: Los juegos de rol táctico a menudo presentan tramas ricas y personajes profundos que interactúan en un mundo de fantasía o medieval. Las decisiones pueden afectar la historia y las relaciones entre los personajes.

Clases y Habilidades: Los personajes pueden pertenecer a diversas clases, cada una con sus propias habilidades y ventajas. La elección de clases y la planificación de habilidades son esenciales para el éxito táctico.

2.1.1 Fire Emblem

Fire Emblem: Ankoku Ryū to Hikari no Tsurugi es un videojuego de rol táctico para la NES, desarrollado por Intelligent Systems y Nintendo R&D1 y publicado por Nintendo. El juego gira en torno a batallas tácticas en mapas basados en cuadrículas, con las unidades derrotadas sujetas a muerte permanente [3].

El primer juego de la serie titulado simplemente "Fire Emblem" en Occidente, fue lanzado en 1990 para la consola Famicom de Nintendo. Dirigido por Shouzou Kaga, este juego sentó las bases para lo que se convertiría en una franquicia de estrategia y rol aclamada en todo el mundo.

Se desarrolla en el continente de Archanea y sigue la historia de Marth, el príncipe exiliado de Altea. La trama comienza cuando su reino es atacado por el ejército de Dolhr, dirigido por el malvado Dragón Divino Medeus. Marth huye y busca aliados para reclamar su trono y liberar su tierra natal de la opresión de Medeus. En cuanto a la ambientación, el videojuego se desarrolla en un mundo medieval de fantasía lleno de reinos en conflicto, dragones, magia y héroes legendarios. La historia se desarrolla a lo largo de una serie de capítulos y misiones, cada una con su propia trama y personajes.

La jugabilidad de este título asentó las bases de lo que sería la franquicia de videojuegos Fire Emblem, esta consiste en batallas por turnos donde el jugador puede mover sus tropas a través de un mapa con forma de tablero cuadriculado siguiendo un sistema de combate por turnos. A lo largo de los diferentes niveles las condiciones para completar el nivel pueden variar, desde tener que vencer al general enemigo hasta tener que escapar de un ejército que nos supera con creces. Cada unidad pertenece a una clase con habilidades propias y queda a juicio del jugador el cómo utilizarlas. En la Figura 2 se puede ver una imagen del juego en cuestión.

Figura 2. Mapa de Fire Emblem

Debido al éxito de este juego la saga Fire Emblem prosperó y dió lugar a una gran cantidad de juegos que heredan las mecánicas de este primer título y añaden nuevas mecánicas para enriquecer la experiencia.

A continuación, un listado de los juegos de la saga:

1. Fire Emblem: Ankoku Ryū to Hikari no Tsurugi (1990) - NES
2. Fire Emblem Gaiden (1992) - NES
3. Fire Emblem: Monshou no Nazo (1994) - SNES
4. Fire Emblem: Seisen no Keifu (1996) - SNES
5. Fire Emblem: Thracia 776 (1999) - SNES
6. Fire Emblem: Fuuin no Tsurugi (2002) - Game Boy Advance
7. Fire Emblem (2003) - Game Boy Advance (Conocido como Fire Emblem: Blazing Sword en América)
8. Fire Emblem: The Sacred Stones (2004) - Game Boy Advance
9. Fire Emblem: Path of Radiance (2005) - GameCube
10. Fire Emblem: Radiant Dawn (2007) - Wii
11. Fire Emblem: Shadow Dragon (2008) - Nintendo DS
12. Fire Emblem: New Mystery of the Emblem (2010) - Nintendo DS
13. Fire Emblem Awakening (2012) - Nintendo 3DS

14. Fire Emblem Fates (2015) - Nintendo 3DS (Incluye tres versiones: Birthright, Conquest y Revelation)
15. Fire Emblem Echoes: Shadows of Valentia (2017) - Nintendo 3DS
16. Fire Emblem: Three Houses (2019) - Nintendo Switch
17. Fire Emblem Engage (2023) – Nintendo Switch

2.1.2 Triangle Strategy

Triangle Strategy es un videojuego de rol táctico desarrollado por Square Enix y Artdink para la Nintendo Switch. El juego fue publicado por Square Enix en Japón y por Nintendo intencionalmente para la Nintendo Switch el 4 de marzo de 2022 [4].

Figura 3. Partida Triangle Strategy

Este título cautivó a los jugadores con su impresionante estilo visual similar a una pintura y una narrativa rica y compleja. El juego destaca por su enfoque en la toma de decisiones morales que afectan el desarrollo de la trama y el destino de los personajes. Los jugadores se enfrentan a dilemas éticos a lo largo de la historia y deben elegir entre tres facciones en conflicto, lo que da lugar a múltiples finales posibles. Además, "Triangle Strategy" mantiene las raíces de los juegos de estrategia táctica, desafiando a los jugadores a planificar cuidadosamente sus movimientos en el campo de batalla para obtener la victoria, las mecánicas incluyen la gestión de puntos de acción para realizar acciones como moverse, atacar o usar habilidades especiales. Los jugadores también pueden personalizar a sus personajes, equipándolos con diferentes armas y objetos, y explorar un mundo detallado mientras avanzan en la historia.

Este juego, con su combinación de elementos de rol profundo y jugabilidad estratégica, se ha convertido en una adición emocionante a la biblioteca de títulos de juegos tácticos y ha sido bien recibido por los fanáticos del género. Se puede ver una captura de una partida en la Figura 3.

2.1.3 Final Fantasy Tactics

Final Fantasy Tactics es un juego de rol táctico basado en combates por turnos desarrollado por Squaresoft para PlayStation. El juego se desarrolla en un reino medieval ficticio llamado Ivalice y sigue la historia de Ramza Beoulve, un cadete que se ve envuelto en medio de un conflicto militar conocido como La Guerra de León. Los personajes se mueven en escenarios llenos de cuadrados y el rango de acción está determinado por las estadísticas de cada personaje y clase de trabajo. Final Fantasy Tactics presenta un sistema de clases de personajes visto en juegos anteriores de la franquicia [5].

Figura 4. Partida de Final Fantasy Tactics Advance

Lo que hizo que "Final Fantasy Tactics Advance" fuera tan especial fue su sistema de trabajo, que permitía a los personajes cambiar de profesión y aprender nuevas habilidades. Esto proporcionó una gran variedad de opciones estratégicas en el campo de batalla. Además, el juego ofrecía una narrativa rica y una exploración del tema de la escapatoria de la realidad a través de la fantasía. La mezcla de una jugabilidad profunda y una historia conmovedora lo convirtió en un favorito de los fanáticos y una joya dentro de la serie "Final Fantasy" y el género de los juegos tácticos.

2.2 Comparación con la propuesta

A continuación, se describe cómo la propuesta, Realm Of Warriors, implementa las mecánicas vistas en el apartado anterior. Se hará una comparación con los títulos mencionados en el apartado previo.

Si bien en todos los juegos de rol táctico es común que las unidades puedan morir durante el transcurso de la partida, en el juego Fire Emblem las unidades muertas no pueden regresar durante el resto de la partida. No obstante, otros juegos del género como los seleccionados para esta comparativa no tienen dicha mecánica implementada, en el caso de Realm Of Warriors al buscarse un enfoque más amigable para los jugadores que se inician en este género de videojuegos la mecánica de muerte permanente de unidades o *Permadeath* en inglés, no estará implementada.

Por otro lado, una característica muy común que estará presente en el juego y que no puede faltar a la hora de crear un juego de rol táctico es el sistema de clases. No obstante, a diferencia de juegos como Fire Emblem y Final Fantasy Tactics donde los personajes tienen una clase, pero esta puede variar, nos acercaremos más a la forma en la que estas se presentan en Triangle Strategy donde los personajes tienen una clase fija que irá ganando experiencia y progresando dentro de la misma, incluso tal vez alcanzando evoluciones de esta. En la propuesta esta mecánica se simplifica aún más dejando los personajes en clases estables que definen su rol desde el principio.

Otra característica que está presente en los videojuegos elegidos es el sistema de agrupaciones que consiste en hacer que los personajes se vuelvan más fuertes si pelean en casillas adyacentes a sus aliados, mecánica que se ha excluido de nuestro videojuego. Una mecánica que compartirá con todos los videojuegos previamente mencionados será la interacción con el terreno, pues se podrá usar para el beneficio de las tropas aliadas o podrá jugar en su contra.

Otra mecánica que ha sido excluida de nuestro videojuego y que está presente en los demás es la posibilidad de que las acciones fallen, es decir, si realizas un ataque existe una probabilidad de que ese ataque no tenga éxito y la unidad enemiga no sufra ningún daño. Por ello y con el objetivo de hacer que Realm Of Warriors sea un juego sencillo para jugadores poco experimentados, esta mecánica se ha eliminado.

Y, por último, algo que no puede faltar en ningún juego del género es una historia que haga que los jugadores se sumerjan en la experiencia.

En la Tabla 1 que se encuentra a continuación podemos ver las características analizadas anteriormente comparando los títulos que se han descrito en este capítulo con la propuesta realizada.

	Realm Of Warriors	**Fire Emblem**	**Triangle Strategy**	**Final Fantasy Tactics**
Permadeath	**No**	**Sí**	**No**	**No**
Clases y Habilidades	**Sí**	**Sí**	**Sí**	**Sí**
Historia	**Sí**	**Sí**	**Sí**	**Sí**
Efectividades	**No**	**No**	**Sí**	**Sí**
Posibilidad de acción	**No**	**Sí**	**Sí**	**Sí**
Sistema de agrupaciones	**No**	**No**	**Sí**	**No**

	Realm Of Warriors	Fire Emblem	Triangle Strategy	Final Fantasy Tactics
Interacción con el terreno	**Sí**	**Sí**	**Sí**	**Sí**

Tabla 1. Comparación de características

Capítulo 3

Análisis del problema

Este TFG se ha centrado en dos áreas principales del desarrollo del videojuego:

- Crear un back-end con la lógica necesaria para facilitar la migración entre plataformas de desarrollo de videojuegos.
- El desarrollo de la IA de los enemigos.

3.1 Especificación de requisitos

3.1.1 Requisitos funcionales

Los requisitos funcionales recopilados al comienzo del desarrollo del proyecto se agrupan en diversas categorías en función de la funcionalidad a la que se refieren. Un requisito funcional es una descripción precisa de una función o característica que un sistema de software debe cumplir o ejecutar. Estos requisitos funcionales son fundamentales para comprender y definir el comportamiento que el software debe tener para satisfacer las necesidades del usuario y los objetivos del proyecto.

Gestión de Almacenamiento

El almacenamiento de datos en la base de datos. El back-end debe dar soporte al almacenamiento de estructuras de datos que representan los elementos del juego en su base de datos.

Alm1 - Se dispondrá de los métodos para recuperar la información necesaria.

Alm2 - Se guardarán todos los elementos referentes al juego necesarios, mapas, personajes, tiles y relaciones entre ellos.

Alm3 - Debe ser posible agregar información a la base de datos.

Alm4 - Debe ser posible modificar información de la base de datos.

Alm5 - Debe ser posible borrar información de la base de datos.

Alm6 - Desde el juego debe poderse modificar la información de base de datos a través de la API.

Partida

El back-end ha de proporcionar métodos para comunicarse con el entorno de desarrollo en el que se realice el videojuego para llevar a cabo las acciones relacionadas con la jugabilidad.

Part1 - El jugador podrá mover los personajes.

Part2 - Los jugadores podrán utilizar las habilidades de los personajes.

Part3 - El jugador podrá utilizar pociones.

Part4 - El jugador podrá atacar a tropas enemigas.

Part5 - Cada personaje que controle el jugador tendrá habilidades únicas.

Part6 - Cada personaje dentro del juego cumplirá un rol marcado.

Part7 - Cada batalla tendrá un requisito para su finalización.

IAEnemigos

El back ha de tener una máquina de estados que gestione las acciones que realizan las unidades enemigas.

IA1 - Las unidades enemigas podrán atacar.

IA2 - Las unidades enemigas podrán retirarse.

IA3 - Las unidades enemigas podrán agruparse.

IA4 - Las unidades enemigas podrán alternar entre retirarse y atacar.

IA5 - Las unidades enemigas podrán decidir qué acción es mejor en cada ocasión.

3.1.2 Requisitos no funcionales

Un requisito no funcional en proyectos de software se refiere a una característica o restricción que no se relaciona directamente con la funcionalidad específica del software, sino más bien con aspectos de rendimiento, calidad, seguridad, usabilidad y otros criterios que afectan la experiencia general del usuario o el funcionamiento del sistema.

RNF1 - Los tiempos de respuesta no deben ser prolongados.

RNF2 - El back-end debe tener facilidad para ser ampliado.

RNF3 - El juego ha de ser sencillo, el jugador debe poder dominar la jugabilidad de forma fácil e intuitiva.

RNF4 - El juego podrá ser ejecutado en equipos con pocos recursos.

RNF5 - Fácil modificación de la solución.

RNF6 - Los datos de la base de datos deben ser accesibles desde cualquier dispositivo que descargue el juego.

3.2 Modelado conceptual

Una vez que se han detallado los requisitos que el back-end debe cumplir, es momento de especificar cómo será el sistema una vez que esté implementado. Para lograr esto, es común utilizar el estándar UML (Lenguaje Unificado de Modelado), que es el lenguaje de modelado de software más ampliamente reconocido. Se ha creado un diagrama de componentes, que se muestra en la Figura 5. Este diagrama representa cómo se han agrupado las diferentes clases responsables de respaldar la funcionalidad del back-end.

Este enfoque de modelado ayuda a visualizar la estructura y la interacción de los componentes del sistema, lo que facilita la comprensión y el desarrollo de la aplicación.

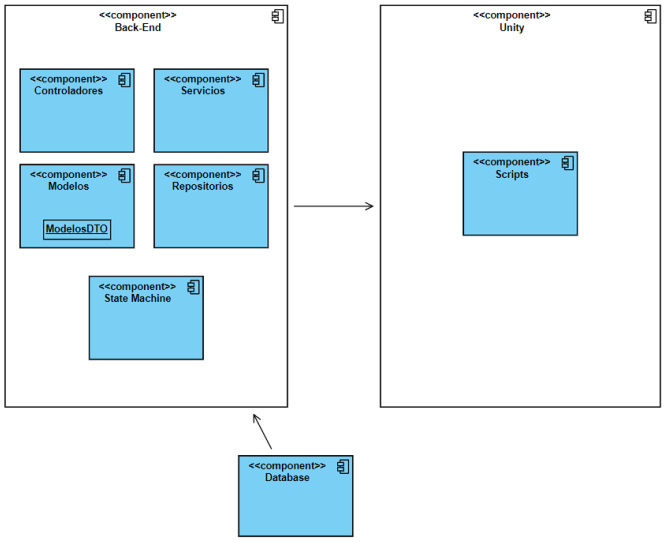


Figura 5. Diagrama de componentes

Por un lado, tenemos un módulo donde se agrupan las clases empleadas para el modelado de los objetos de base de datos y de las clases de modelos Dto, las siglas DTO (Data Transfer Object) hacen referencia a un tipo de objetos que sirve únicamente para transportar datos.

Otro módulo es el módulo de repositorios en el cual se encuentran los métodos específicos para cada modelo con los cuales se accede a la base de datos y se recupera la información que se solicite.

Por último, tenemos otro módulo donde se encuentran los controladores de los modelos estos controladores son componentes que se utilizan para gestionar y procesar las solicitudes entrantes y las respuestas salientes. Los controladores desempeñan un papel fundamental en la organización y la estructura de una API pues se encargan del manejo de solicitudes HTTP enviadas por los clientes, el procesamiento de los datos enviados en las solicitudes y son los encargados de interactuar con la capa de modelos y de repositorios para acceder a la base de datos o con los servicios desarrollados en la API para generar respuestas a las solicitudes.

3.3 Solución propuesta

La solución propuesta se ha desarrollado teniendo como base el videojuego Realm Of Warriors que fue desarrollado para la asignatura de Desarrollo de videojuegos 2D (14101). Durante el videojuego el jugador se sumerge en una historia de ambientación medieval, y trata de vencer a ejércitos enemigos con las tropas que se encuentran a su disposición durante batallas que tienen lugar a lo largo de la historia en mapas dispuestos con forma de tablero cuadriculado por el cual se desplazan las unidades. A medida que avanza la historia se unen nuevos personajes a las unidades que puede controlar el jugador, cada una de estas

con estadísticas y habilidades diferentes, haciendo que el jugador deba planificar una estrategia para sacar el máximo provecho a cada unidad.

Para soportar todas las tareas que se requieren para el funcionamiento del videojuego se ha creado un back-end que contiene una conexión a una base de datos alojada en Azure donde se almacenan los datos de los diferentes personajes, mapas y sus elementos y de las batallas que se llevan a cabo. Dicho back debe ser accesible en todo momento por cualquiera jugador que inicie el juego, por ello se alojará a modo de API en Azure.

3.4 Plan de trabajo

Una vez se tenían claros los requisitos funcionales y no funcionales que se implementarían en el proyecto, se redactaron tareas que se llevarían a cabo. Se asigna prioridades a las diferentes tareas según la urgencia e importancia que tendría para el desarrollo del videojuego y se subdividieron en subtareas para tener un control más precioso de la carga de trabajo restante. En la Figura 6 se pueden ver algunas tareas que se desarrollaron durante el proyecto.

Se puede ver que en la aplicación utilizada había diferentes opciones, se podía establecer una fecha límite para el desarrollo de la tarea, se permite mediante un icono de bandera establecer la prioridad de la tarea que se ha de desarrollar según el color asignado a la misma y cada tarea puede ser dividida en subtareas que desaparecen en cuanto se marcan como completadas.

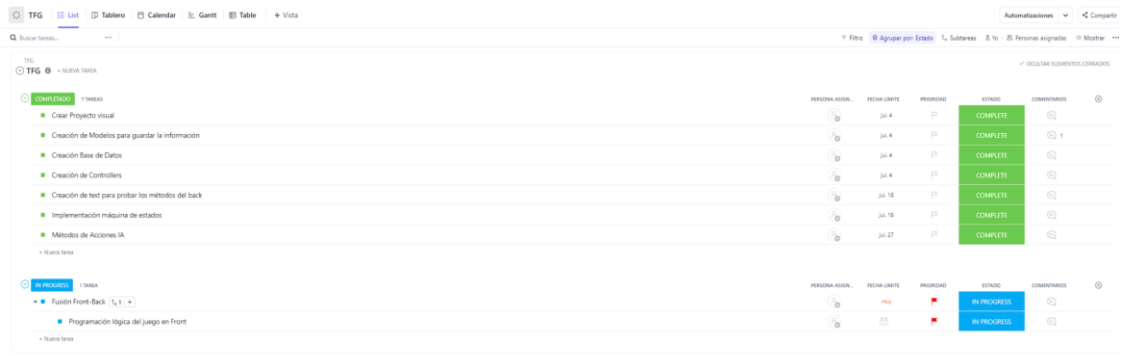


Figura 6. Lista de tareas del proyecto

Capítulo 4

Tecnologías Utilizadas

En este capítulo, se presentarán y describirán todas las tecnologías y herramientas que se han empleado durante el proceso de desarrollo del proyecto.

Al querer dar independencia al back del juego desarrollado en el motor de desarrollo de videojuegos Unity, lo primero en lo que se pensó es en crear una librería escrita en C# que tuviera clases y métodos que se encargan de extraer la funcionalidad del juego y responder a las llamadas que se hicieran desde el videojuego en Unity.

No obstante, teniendo en cuenta la posible proyección que se le daría al videojuego con el desarrollo de un multijugador finalmente se decidió crear una API que se encargará de llevar la lógica del juego y gestionar las llamadas a base de datos.

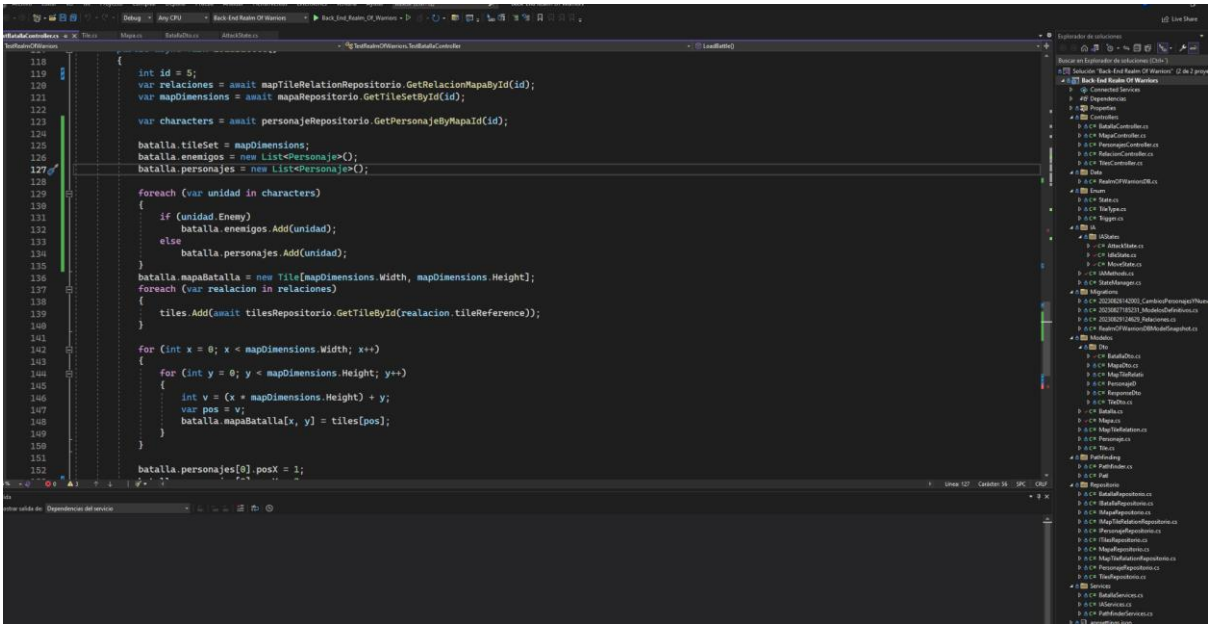
4.1 Visual Studio 2022

En este proceso de desarrollo de una solución informática, es esencial contar con un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) que facilite la programación y gestión del proyecto. En nuestro caso, hemos seleccionado Visual Studio 2022 debido a la familiaridad con él y a las herramientas que proporciona para programar, depurar e implementar soluciones de manera eficiente.

Visual Studio es ampliamente reconocido en la industria de desarrollo de software por su versatilidad y su amplia gama de características que respaldan el desarrollo de aplicaciones.

El principal motivo por el que se decidió la utilización de este IDE es que en anteriores proyectos realizados en grupo ya había dado resultados satisfactorios. Además, la

programación de los scripts realizados para la primera versión del juego Realm Of Warriors habían sido realizados utilizando Visual Studio.



```
{
    int id = 5;
    var relaciones = await mapTileRelationRepositorio.GetRelacionMapaById(id);
    var mapDimensions = await mapaRepositorio.GetTileSetById(id);

    var characters = await personajeRepositorio.GetPersonajeByMapaId(id);

    batalla.tileSet = mapDimensions;
    batalla.enemigos = new List<Personaje>();
    batalla.personajes = new List<Personaje>();

    foreach (var unidad in characters)
    {
        if (unidad.Enemy)
            batalla.enemigos.Add(unidad);
        else
            batalla.personajes.Add(unidad);
    }
    batalla.mapaBatalla = new Tile[mapDimensions.Width, mapDimensions.Height];
    foreach (var realacion in relaciones)
    {
        tiles.Add(await tilesRepositorio.GetTileById(realacion.tileReference));
    }

    for (int x = 0; x < mapDimensions.Width; x++)
    {
        for (int y = 0; y < mapDimensions.Height; y++)
        {
            int v = (x * mapDimensions.Height) + y;
            var pos = v;
            batalla.mapaBatalla[x, y] = tiles[pos];
        }
    }

    batalla.personajes[0].posX = 1;
```

Figura 7. Clase de Visual Studio

En la Figura 7 podemos ver una clase desarrollada en Visual Studio, a la derecha se pueden ver todas las clases y carpetas pertenecientes al proyecto.

4.2 Unity

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, Mac OS, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas (Véase la sección Plataformas objetivo). A partir de su versión 5.4.0 ya no permite el desarrollo de contenido para navegador a través de su plugin web, en su lugar se utiliza WebGL. [6]

Se eligió Unity por haber sido el entorno utilizado durante la asignatura de desarrollo en videojuegos 2D pues ya se estaba familiarizado al haber cursado la asignatura, Unity ofrece soporte para realizar a llamadas a la API que se había desarrollado como back-end así como un entorno que se puede asociar con Visual Studio de forma sencilla para la depuración y programación de las clases a las que se corresponden los Scripts.

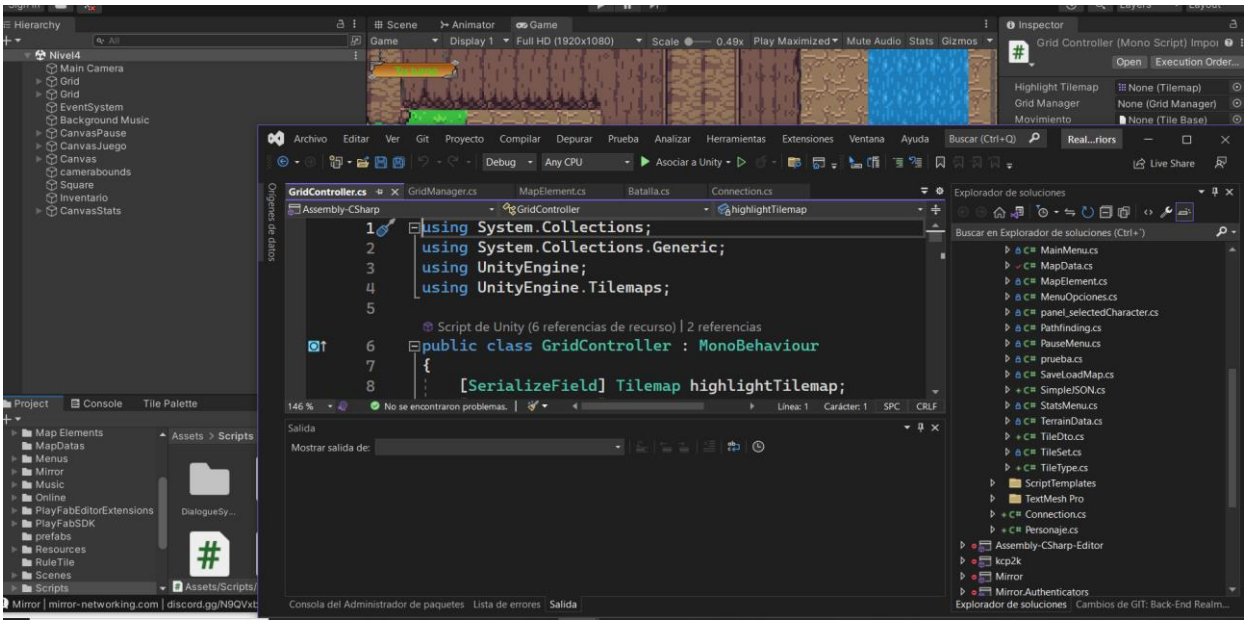


Figura 8. Entorno de Unity vinculado a Visual Studio.

En la Figura 8 podemos ver un como es el entorno de Unity que se encuentra asociado a Visual Studio donde se programan los scripts que hacen que el videojuego sea jugable.

4.3 Azure

Microsoft Azure (anteriormente Windows Azure y Azure Services Platform) es una plataforma de computación en la nube creada por Microsoft para construir, probar, desplegar y administrar aplicaciones y servicios mediante el uso de sus centros de datos. Proporciona software como servicio (SaaS), plataforma como servicio (PaaS) e infraestructura como servicio (IaaS) y es compatible con muchos lenguajes, herramientas y marcos de programación diferentes, incluidos software y sistemas específicos de Microsoft y de terceros.

Azure fue anunciado en octubre de 2008, comenzó con el nombre en clave "Project Red Dog" y publicado el 1 de febrero de 2010 como "Windows Azure" antes de ser rebautizado como "Microsoft Azure" el 25 de marzo de 2014. [7]

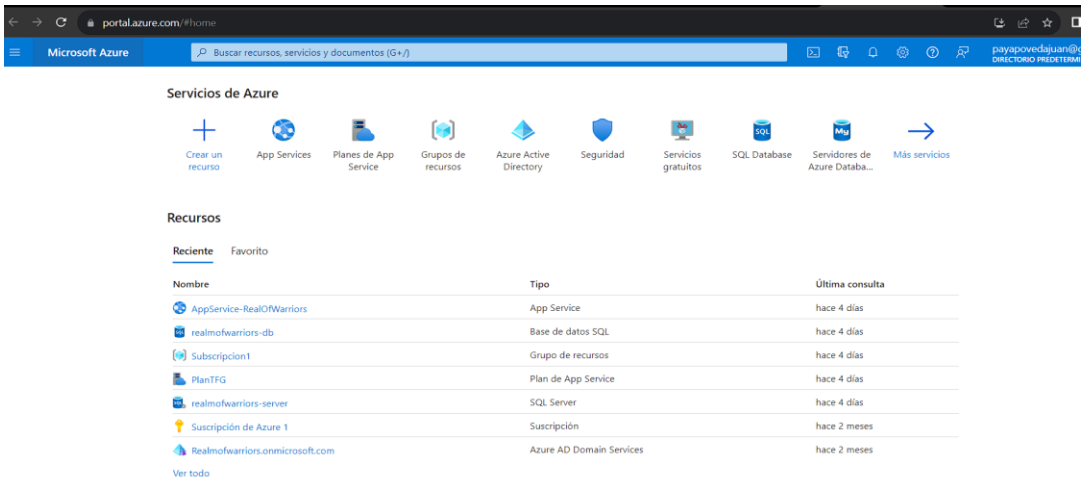


Figura 9. Portal de Azure

En la Figura 9 se puede ver el perfil de azure en el que se han creado los servicios necesarios para mantener la base de datos y la API que está conectada a Unity desde donde se realizan los accesos al servicio para crear los objetos del videojuego durante la ejecución.

4.4 SQL Server Management Studio Management 19

Microsoft SQL Server Management Studio (comúnmente abreviado como SMSS) es una aplicación utilizada para la gestión y administración de los componentes dentro de SQL Server. Es el sucesor de Enterprise Manager de la versión 2000 haciendo su aparición por primera vez con la versión 2005. Entre sus características destacan el Explorador de Objetos, en el cual el usuario puede manipular cualquier objeto dentro del servidor al que se esté conectado, herramientas para generar comandos y scripts para tareas administrativas, monitorear en registros la actividad del servidor y poder registrar y conectarse a estos sea en la ubicación o de forma remota. [8]

Se decidió el uso de Microsoft SQL Management Studio debido a que tiene una gran compatibilidad con el ecosistema de Microsoft ya que para hospedar la base de datos y la API se utiliza el entorno de Azure. Por supuesto, también se eligió por otras características como la amplia funcionalidad que ofrece, su rendimiento y optimización y las opciones de seguridad y administración.

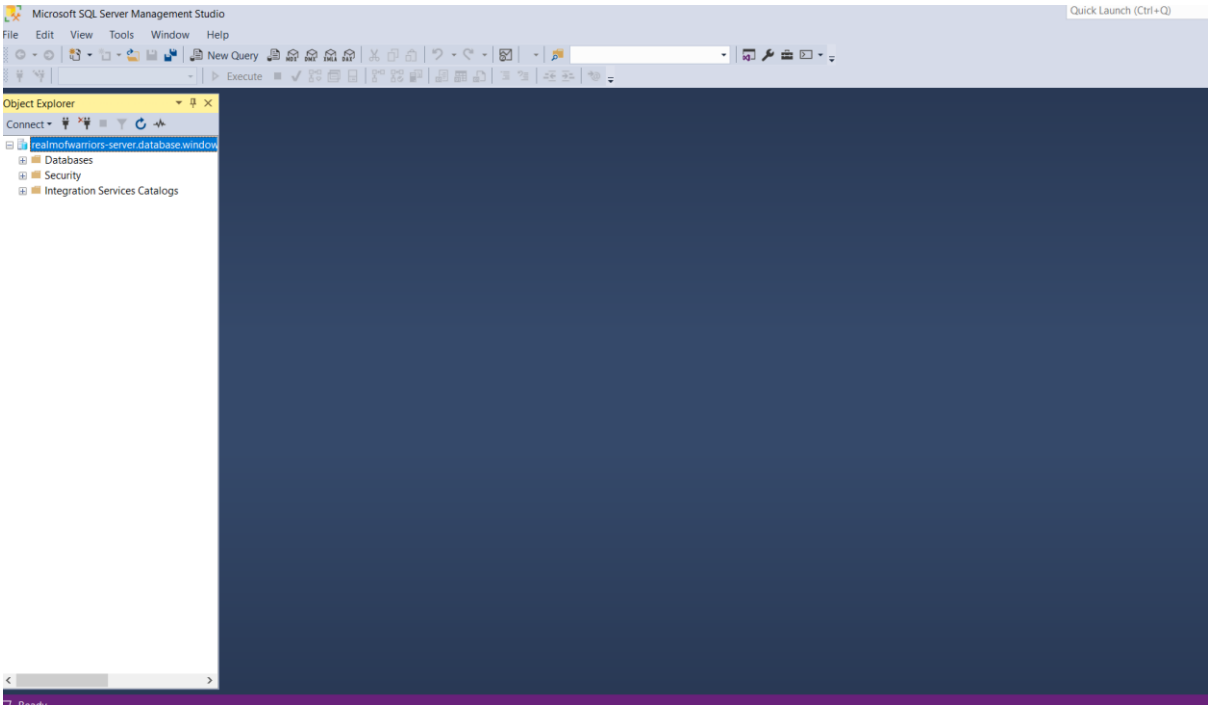


Figura 10. Entorno de SQL Server Management

En la Figura 10 podemos ver el entorno de Microsoft SQL Manager ya vinculado a nuestra base de datos que se puede ver en la barra situada en el lado izquierdo.

4.5 Github

GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador. El software que opera GitHub fue escrito en Ruby on Rails. Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. Anteriormente era conocida como Logical Awesome LLC. El código de los proyectos alojados en GitHub se almacena generalmente de forma pública. [9]

En el caso de nuestro proyecto se utilizó un git que se vinculó al proyecto de back-end en Visual Studio donde se llevaba el control de versiones de la API que se estaba desarrollando, siendo extremadamente útil para saber qué cambios se habían realizado entre las diferentes versiones llevando de esta forma un buen seguimiento del proyecto. También era muy útil en caso de algún mal funcionamiento del proyecto actual pues permite comparar los cambios con una versión que sí era correcta y de esta forma identificar las causas de fallo de manera más sencilla.

4.6 Entity Framework

Entity Framework es un marco de trabajo (framework) de mapeo objeto-relacional (ORM) desarrollado por Microsoft para simplificar el acceso a bases de datos en aplicaciones .NET. Permite a los desarrolladores trabajar con datos relacionales utilizando objetos y clases en lugar de escribir consultas SQL directamente. Entity Framework se utiliza comúnmente en aplicaciones de Microsoft .NET, especialmente aquellas que se basan en ASP.NET y ASP.NET Core.

En el proyecto realizado se ha empleado para poder crear, gestionar y realizar consultas a la base de datos de forma sencilla desde la API que se ha desarrollado. Se generaron los modelos y a partir de estos se generaron las tablas necesarias en la base de datos, todo esto se realizó mediante código utilizando Entity Framework.

Capítulo 5

Diseño

En este capítulo, se presentará y describirá el proceso de desarrollo del back-end de Realm Of Warriors, detallando cómo se implementó la solución propuesta en el análisis.

5.1 Arquitectura del Sistema

En este apartado detallaremos la arquitectura del sistema. El programa está compuesto por distintos módulos que interactúan entre sí. Estos son principalmente El front-end que se corresponde con el videojuego, el back-end que corresponde con la aplicación desarrollada y que interactúa con el front-end y con la base de datos. Esta es la capa donde se ha trasladado toda la lógica del juego. Por último, tenemos la base de datos que está conectada a la API, la cual hace las llamadas para extraer la información que se guarda en la base de datos. Esta base de datos contiene las tablas correspondientes a los modelos generados en el back-end para almacenar la información correspondiente.

En concreto la estructura que se ha seguido es la de una arquitectura en tres niveles que procederemos a explicar a continuación.

La arquitectura de tres niveles es un modelo de diseño de software ampliamente reconocido que estructura las aplicaciones en tres niveles lógicos y físicos: el nivel de presentación o interfaz de usuario, el nivel de aplicación donde se procesan los datos, y el nivel de datos donde se almacenan. A continuación, se muestra un esquema de esta arquitectura en la Figura 11:

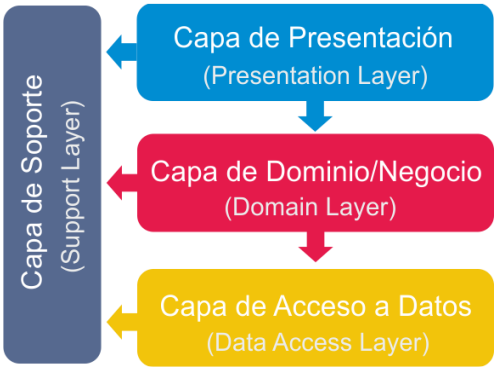


Figura 11. Esquema de arquitectura en tres capas

Esta arquitectura se utiliza comúnmente para separar y organizar las responsabilidades dentro de una aplicación, lo que facilita la escalabilidad, el mantenimiento y la colaboración en el desarrollo de software. Cada uno de los niveles tiene funciones y responsabilidades específicas para asegurar una implementación eficiente y modular de la aplicación.

5.1.1 Capa de presentación

En esta capa se encuentra la interfaz de usuario o cliente que permite a los usuarios interactuar con el sistema. Puede ser una aplicación web, una aplicación móvil, un cliente de escritorio, etc. Su función principal es presentar la información al usuario de manera comprensible y recopilar la entrada del usuario para su procesamiento.

El nivel de presentación se corresponde con el videojuego que en nuestro caso se trata de Realm Of Warriors, en este caso el videojuego recoge las acciones del usuario y se comunica mediante llamadas HTTP con la API para obtener una respuesta y aplicar los cambios en el estado actual de la partida. De esta forma es el servicio el que realiza los cambios pertinentes y devuelve al front-end los cambios que debe efectuar para que aquello que se ve durante la partida se corresponda con el estado real de la partida. En la Figura 12 podemos ver una captura de uno de los mapas de Realm Of Warriors.

Figura 12. Partida de Realm Of Warriors

5.1.2 Capa de lógica de negocios

La capa de lógica de negocios, también conocida como capa intermedia o API (Interfaz de Programación de Aplicaciones), es responsable de procesar las solicitudes del cliente, ejecutar la lógica empresarial y coordinar las operaciones necesarias en la base de datos. Aquí se manejan la validación, el procesamiento de datos y cualquier lógica específica del dominio de la aplicación.

La API actúa como un intermediario entre la capa de presentación y la capa de datos. Proporciona una interfaz estandarizada a través de la cual el cliente puede enviar solicitudes y recibir respuestas. Esto permite separar la lógica del cliente de la lógica de negocio, lo que facilita el mantenimiento y la escalabilidad.

En el contexto del proyecto la capa de lógica se corresponde con la API que actúa como back-end del proyecto en esta se tienen las clases que aplican la lógica del videojuego y simulan una partida devolviendo de esta forma al front-end respuestas para que se construyan GameObjects en Unity, se genere el tablero de la partida o se realicen los diversos cambios que sean necesarios en base a las decisiones que el jugador haya tomado.

5.1.3 Capa de datos

La capa de datos es donde se almacena y gestiona la información del sistema. Puede ser una base de datos relacional (como Microsoft SQL Server, MySQL, PostgreSQL) o no relacional (como MongoDB, Redis). Aquí se almacenan los datos en tablas o estructuras adecuadas según el tipo de base de datos. La capa de datos es la encargada de manejar las operaciones de almacenamiento, recuperación, actualización y eliminación de datos.

Se ha creado una base de datos hospedada en Azure con la finalidad de que recopile la información necesaria para dar soporte a las operaciones que se realicen en el back-end y contenga los modelos que son necesarios para llevar a cabo la lógica de la capa de negocio.

En resumen, la interacción entre estas capas funciona de la siguiente manera: el cliente envía solicitudes a la API a través de protocolos de comunicación como HTTP. La API procesa estas solicitudes, realiza operaciones de lógica y accede a la base de datos según sea necesario. Luego, la API devuelve una respuesta al cliente, que se presenta en la interfaz de usuario.

5.2 Diseño del juego

El jugador parte de una pantalla de inicio donde dará comienzo a la partida, en esta primera pantalla se encuentran las opciones básicas de ver los créditos, configurar las opciones del juego o comenzar nueva partida. Una vez iniciada la partida el jugador accede al nivel 1 que se corresponde con la primera batalla, cada vez que el jugador complete un nivel, aparecerá en medio de la pantalla un letrero que indicará que el nivel ha sido completado e iniciando de forma automática el avance a la siguiente batalla, avanzando hasta llegar al final del juego lo cual lo llevaría a la pantalla de créditos. Si en cualquiera de los niveles el jugador fuera derrotado este sería devuelto a la pantalla de inicio donde debería volver a empezar la aventura. En la siguiente Figura 13 podemos ver el diagrama de navegación de pantallas.

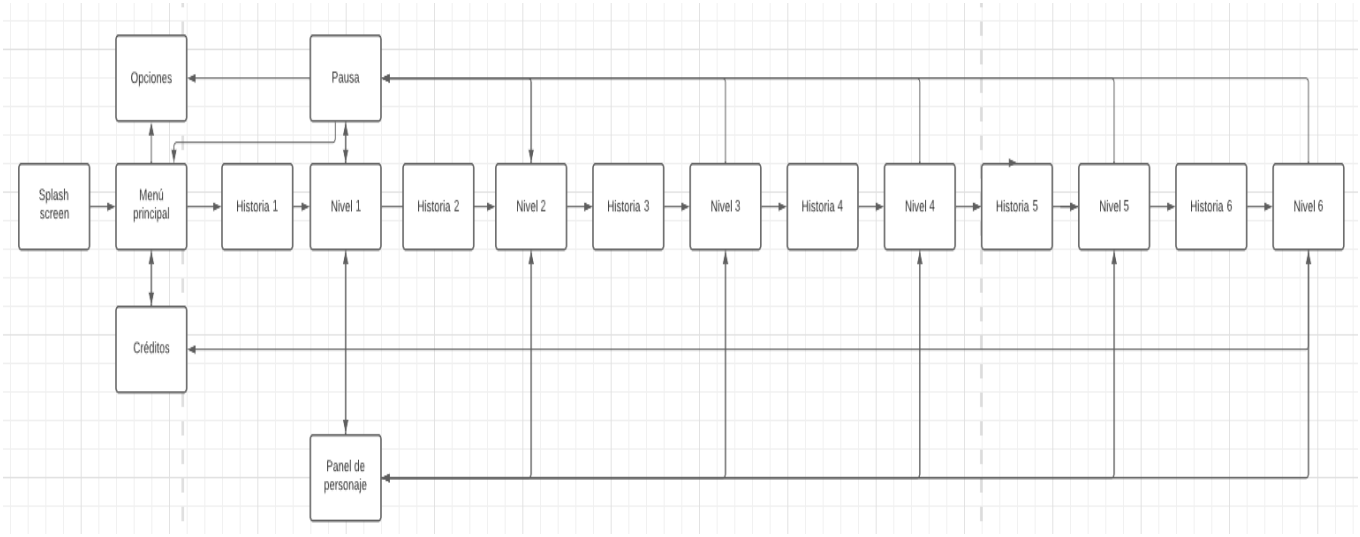


Figura 13.Diagrama de navegación de pantallas.

Es interesante notar que Realm Of Warriors está fuertemente influenciado por la saga de juegos Fire Emblem, que consta de una serie impresionante de 23 videojuegos. La mayoría de estos títulos han sido lanzados en diversas consolas de Nintendo, desde la NES hasta la Nintendo Switch, además de incursiones en dispositivos móviles como Fire Emblem Heroes, lanzado en 2017.

Particularmente, una de las principales fuentes de inspiración dentro de la franquicia Fire Emblem para Realm Of Warriors es el videojuego Fire Emblem Shadow Dragon, que se lanzó para Nintendo DS en 2004. Este juego en particular ha influido en la forma en que se han diseñado los mapas en Realm Of Warriors. En la Figura 14 se puede apreciar una imagen del juego Fire Emblem Shadow Dragon, donde se pueden ver las características de los mapas que han servido de base para la creación de los mapas en Realm Of Warriors.

Figura 14. Mapa Fire Emblem Shadow Dragon

Por ello al ser desarrollado para PC ha visto una adaptación de sus controles para poder ser jugable mediante teclado y ratón, basándose en la versión para PC de un juego del género reciente llamado Triangle Strategy llegado a PC el 13 de octubre de 2022. Se puede ver una imágen de la jugabilidad de dicho juego en la Figura 15.

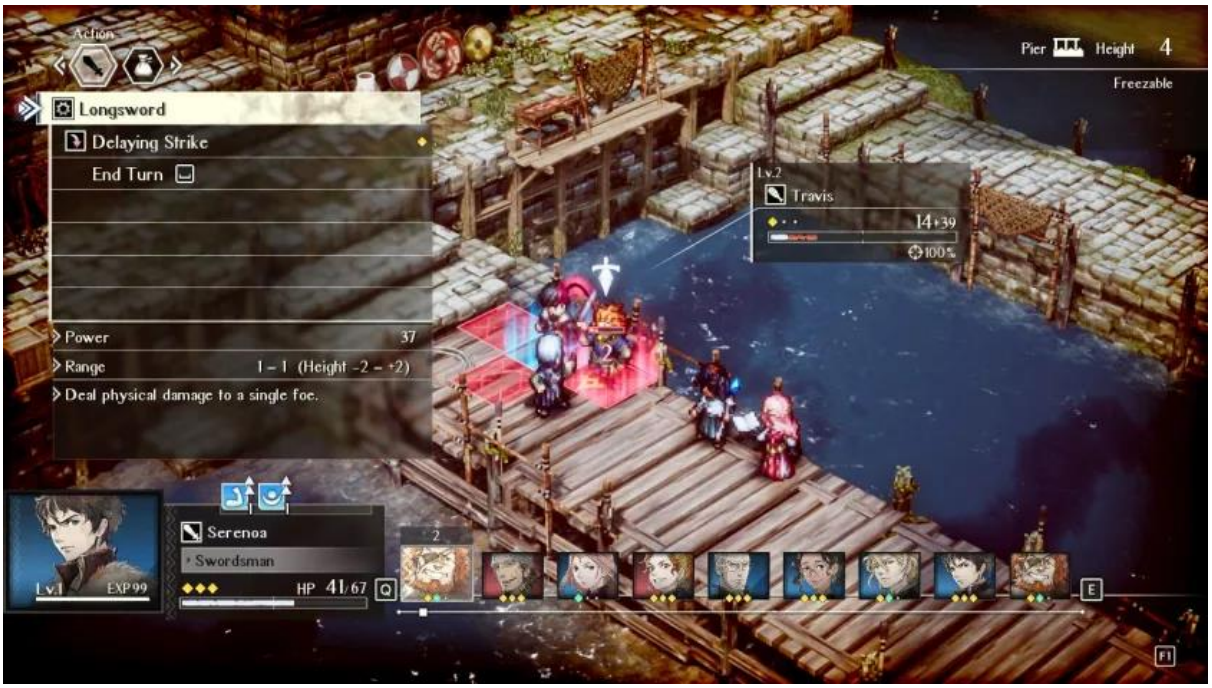


Figura 15. Partida de la versión de PC de Triangle Strategy

5.3 Diseño de las Batallas

El elemento clave dentro del juego son las batallas que tienen lugar, pues es donde el jugador controla las unidades con el objetivo de derrotar a las unidades del rival y es donde transcurre la mayor parte del juego. A la hora de establecer un diseño, se barajaron diversas opciones, pero como se ha mencionado en el apartado anterior hubo una fuerte inspiración en el diseño de mapas por parte del juego Fire Emblem Shadow Dragon (7 de agosto de 2008).

Para el que se puedan desarrollar de forma correcta se diseñó una clase que contuviera los elementos necesarios para simular una batalla en el back-end, de tal forma que el videojuego tan solo debe consultar el estado de esta y representarlo en el videojuego desvinculando de esta forma la lógica del back y del videojuego. El jugador inicia una partida, en ese momento en el back se carga una batalla según cual sea el nivel del juego en el que

se ha entrado, a partir de ese momento cada acción que el jugador realice será transmitida al back-end donde se modifica la batalla según la acción realizada por el jugador y será devuelta al front para que se represente en la interfaz que ve el usuario consiguiendo así un desacoplamiento completo de la lógica del juego y de la interfaz. En la Figura 16 podemos ver una captura donde se muestran los atributos del modelo de Batalla.

```
public Mapa? tileSet { get; set; }
public List<Personaje> personajes = new List<Personaje>();
public List<Personaje> enemigos = new List<Personaje>();
public List<(int, int)> posicion = new List<(int, int)>();
public Tile[,] mapaBatalla;
```

Figura 16. Clase del modelo de Batallas

5.4 Diseño de la IA de los enemigos

Otro elemento de gran importancia es dotar de inteligencia a las unidades enemigas para que supongan un reto.

Con el objetivo de conseguir un comportamiento complejo por parte de las unidades controladas por el videojuego, se ha implementado una máquina de estados. En el videojuego los enemigos controlados por la máquina ejecutarán acciones según el estado en el que se encuentre la máquina y se realizan transiciones entre estados según las necesidades. Las unidades enemigas esperan a que las unidades del jugador entren en el rango de acción establecido para estas y deciden qué acción llevar a cabo dependiendo de en qué estado se encuentre la máquina. Se puede ver un esquema que representa los estados que rigen el comportamiento de las unidades controladas por la máquina en la Figura 17, en esta podemos ver que estados se encuentran comunicados entre sí y las transiciones que existen entre los diferentes estados.

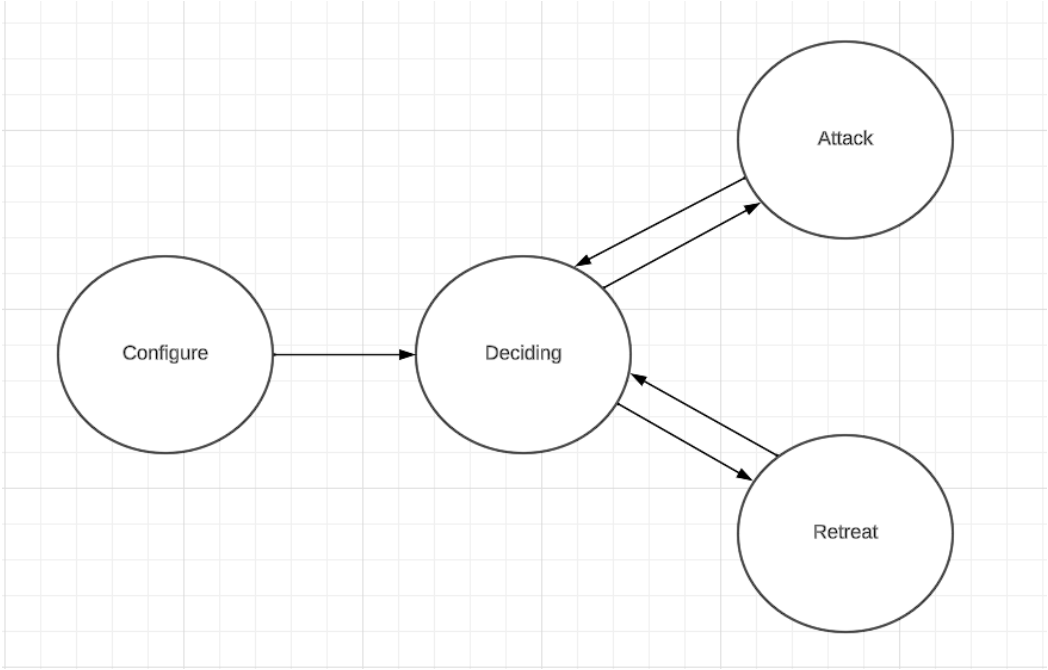


Figura 17. Esquema de estados de la máquina de estados

El funcionamiento de la máquina es el siguiente:

Primero se configura la máquina de estados para crear una instancia de esta con todo lo necesario para llevar a cabo su funcionamiento, una vez la configuración se ha completado la máquina transiciona al estado de Deciding en el cual valorará que opción es mejor si atacar o retirarse haciendo que la máquina cambie al estado que se haya decidido a la salida del estado. En el estado Attack se realizará la acción de ataque, es decir, las unidades que en este estado sean movidas harán una acción ofensiva. Por otro lado, en el estado de Retreat las unidades cuya acción sea jugada mientras la máquina está en este estado lo harán de forma defensiva. Cada vez que la máquina realice una jugada regresa al estado de deciding para volver a deducir cual es la mejor jugada posible en el siguiente turno.

5.5 Diseño de las clases

Una vez se ha definido la arquitectura del sistema, lo siguiente es detallar las clases, usando diagramas de clases UML. En la Figura 18 podemos ver un diagrama de clases correspondiente al componente de batalla. Se puede ver que está compuesta por un mapa, que a su vez está compuesto de una lista de casillas o tiles en inglés y los personajes que participarán en la batalla.

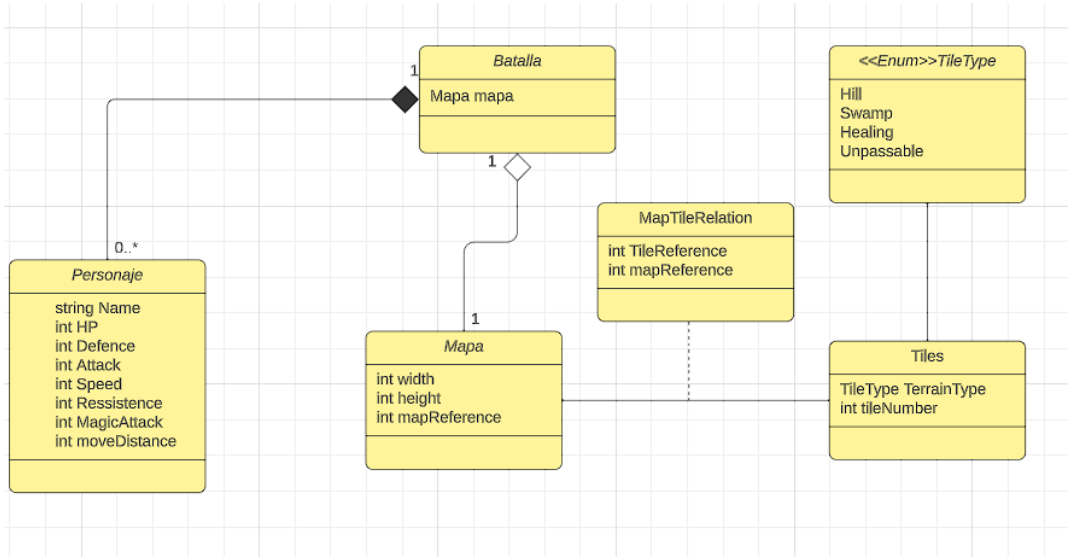


Figura 18. Diagrama de clases de los modelos

La idea principal detrás de esta clase es que se tenga una representación completa de lo que sería una batalla y de esta forma el back-end pueda operar de forma independiente al front-end y así aplicar las operaciones que sean necesarias sin esperar ninguna información extra del videojuego, que se limita a transmitir las acciones que el usuario realiza en el tablero que se encuentra en el front, que no es más que una representación gráfica de la clase Batalla del back.

Por otro lado, tenemos la clase mapa, esta contiene los atributos para la representación correcta de la disposición de las casillas y la referencia a la batalla en la que debe ser utilizado el mapa. Las variables width y height delimitan la anchura y la altura que tendrá el mapa y por último la referencia al mapa permite que se puedan generar diferentes batallas de forma simultánea utilizando la misma plantilla de mapa. En la Figura 19 se puede ver el modelo de la clase Mapa.

```
public class Mapa
{
    [Key]
    2 referencias
    public int Id { get; set; }
    10 referencias
    public int Height { get; set; }
    9 referencias
    public int Width { get; set; }
```

Figura 19. Atributos clase Mapa

En cuanto a la clase tiles tenemos una clase que contiene los atributos necesarios para la representar los diferentes tipos de casillas que se pueden representar en los distintos mapas, tiene un atributo TileType que hace referencia al tipo de terreno que representa esa

casilla y un atributo para identificar el tipo de casilla de forma inequívoca en el front-end cuando sea necesario hacer una representación gráfica del mapa para el usuario. En la Figura 20 podemos ver el modelo que define a la clase Tile.

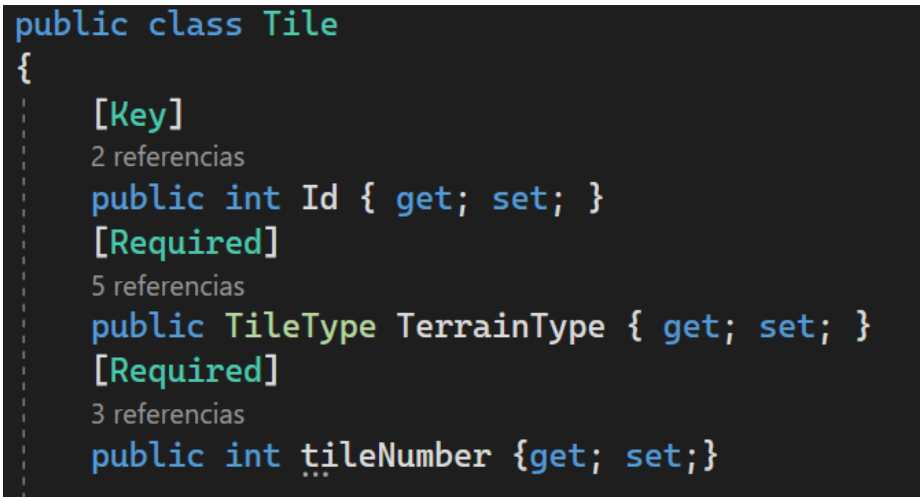


Figura 20. Atributos Clase Tile

Tenemos la clase de personajes, esta clase contiene todos los atributos necesarios para la representación en back de un personaje jugable durante la partida. Los atributos simbolizan sus estadísticas y alguna cualidad referente al personaje. En la Figura 21 que podemos ver parte de los atributos que contiene el modelo de la clase personaje

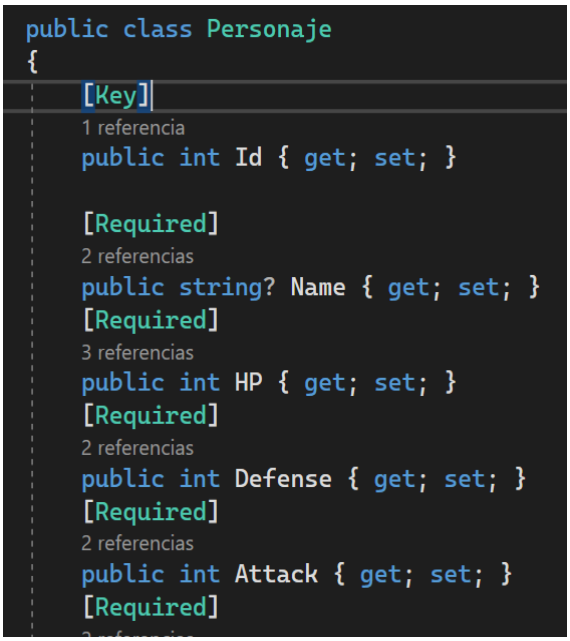


Figura 21. Clase del modelo de personajes

Por último, cabe mencionar la necesidad de crear una clase para relacionar las Tiles que contiene un mapa y de esta forma obtener una lista de las Tiles pertenecientes a un mapa de una batalla.

El conjunto de las clases mencionadas se encarga de que los objetos que son necesarios en el juego se puedan almacenar en la base de datos y luego extraer para realizar las operaciones que sean necesarias con ellos.

Capítulo 6

Desarrollo de la solución

6.1 Comienzo del proyecto

La idea del proyecto surgió durante la primavera del curso 2022/2023. Se quería crear un videojuego similar a los videojuegos de la saga Fire Emblem, ya mencionada en anteriores capítulos. Para ello se decidió desarrollar un videojuego que recordara a los inicios de la saga, no obstante, este tendría un enfoque más sencillo para nuevos jugadores. El videojuego se desarrolló en un inicio utilizando el motor de Unity como proyecto para la asignatura de Desarrollo de videojuegos 2D (14101). Al finalizar la asignatura se le propuso al profesor que impartía la misma, Paco Abad, si estuviera dispuesto a ser mi tutor de TFG, y aceptó amablemente.

Se le comentó al tutor si se pudiera hacer una versión del juego que fuera independiente del motor de desarrollo y profundizar mejor en la IA que se había hecho de forma muy superficial para la primera versión del juego. A este pareció agradarle la idea pues aceptó y se encargó de realizar la formalización del proyecto. Ya con todo listo nos dispusimos a dar comienzo al proyecto.

6.2 Desarrollo del back-end

Realm Of Warriors fue inicialmente concebido como un videojuego desarrollado en Unity, así pues, para realizar el desarrollo de un back-end independiente se debió tener en cuenta que este debía ser adaptable a otros motores de desarrollo de videojuegos por ello se decidió crear una API que se comunicara con una base de datos haciendo de esta forma que el juego pudiera ser escalado fácilmente dependiendo de las necesidades que se requieran del back-end y fácilmente migrable a otros motores de desarrollo.

6.2.1 Desarrollo de la base de datos y estructura de la API

Tras sopesar diversas opciones se decidió crear y hospedar una base de datos en Azure la cual se encargaría de guardar información sobre los modelos que se crearon de forma previa en el back-end, para crear las tablas en base de datos se utilizó Entity Framework con el cual se definieron los modelos de los objetos necesarios.

Para realizar los accesos a la base de datos y recuperar, modificar o eliminar la información que se requiriese en cada caso, se crearon unas clases que se nombraron Repositorio + nombre del modelo, por ejemplo “MapaRepositorio”, estas clases se encargan de agrupar los métodos necesarios para acceder a la base de datos y realizar las operaciones que sean necesarias, como los son operaciones GET para recuperar información desde la base de datos, operaciones POST para introducir datos, PUT para modificar objetos existentes en la base de datos y DELETE para borrar datos.

Los métodos de estas clases son empleados por los métodos pertenecientes a los controladores que se encargan de procesar las solicitudes que el back-end recibe desde el videojuego mediante llamadas HTTP, estas llamadas especifican una ruta a al cual quieren acceder y el tipo de operación que se quiere realizar. De esta forma cuando el back-end recibe una solicitud es capaz de identificar qué método ha de ejecutar.

Para realizar estas llamadas desde Unity se utiliza UnityWebRequest objeto que proporciona un sistema modular para componer peticiones HTTP y respuestas HTTP. Su objetivo principal es hacer que los juegos de Unity puedan interactuar con back-ends Web modernos.

Las llamadas se realizan desde un script que se denominó como Connectios el cual agrupa todos los métodos necesarios para llevar a cabo las solicitudes. Estos métodos son llamados desde las partes en las que se requiere que el back-end realice operaciones y devuelva una respuesta.

Por último, en lo que se refiere a la conexión del back-end y del videojuego se publicó la API en Azure para que esta pudiera ser accedida en cualquier momento.

Realizando esto se cumplieron los requisitos que se indican en la Tabla 5.1

Alm1

Alm2

Alm3

Alm4

Alm5

RNF2

RNF6

Tabla 2.Requisitos de almacenamiento cumplidos.

Los requisitos que hacen referencia fueron completados pues se consiguió una API con todas las operaciones necesarias para el manejo de información, en cuanto a los requisitos no funcionales completados, el RNF2 fue completado ya que la estructura con la que se realizó el proyecto permite una fácil ampliación y el RNF6 se completó tras publicar la API en Azure haciendo que esté activa en todo momento.

6.2.2 Desarrollo de las funciones de juego

Por otra parte tenemos el desarrollo que afectaba a la jugabilidad, el objetivo principal del Back-End de Realm Of Warriors era ofrecer una API para migrar de forma sencilla el juego de un motor de desarrollo a otro, por ello se crearon diferentes clases que proporcionaban los métodos necesarios para tener una representación de la batalla que está teniendo lugar en el videojuego para de esta forma realizar las operaciones en el servidor y comunicar los resultados de dichas operaciones al videojuego y que represente la información al usuario.

Para realizar los movimientos de las unidades por el mapa, durante el primer desarrollo realizado de Realm Of Warriors, se creó un algoritmo de pathfinding adaptado a los elementos del editor de Unity. Un algoritmo de pathfinding es un algoritmo utilizado en informática y programación para encontrar la ruta más corta o eficiente entre dos puntos en un grafo o red. Se utiliza comúnmente en aplicaciones como sistemas de navegación GPS, videojuegos, simulaciones, robótica y muchas otras áreas donde se necesita determinar una ruta óptima.

El objetivo ahora era desarrollar un algoritmo de pathfinding que se encontrara totalmente desvinculado del motor de Unity y que pudiera realizar los cálculos para hallar los caminos que pueden seguir las unidades utilizando los modelos creados para la API.

Como primer paso se creó una clase llamada PathNode, que podemos ver en la Figura 22, la cual representaba una casilla y su coste de movimiento, gracias a esta clase el algoritmo

puede identificar el coste de mover una unidad a través de las casillas que sean parte la ruta a seguir por la unidad.

```
public class PathNode
{
    public TileType terrainType;
    public int xPos;
    public int yPos;
    public float gValue;
    public float hValue;
    public PathNode parentNode;

    4 referencias
    public float fValue
    {
        get
        {
            return gValue + hValue;
        }
    }
```

Figura 22. Clase PathNode

Como siguiente paso se adaptó el algoritmo de pathfinding que se había creado para la versión iniciar de Realm Of Warriors, fue necesario eliminar todos aquello elementos de las clases que utilizaban objetos de Unity y reemplazarlos para que se adaptaran a las clases de los modelos que se hayan en la API, la cual ya se ha descrito en capítulos anteriores. Dando como resultado un algoritmo de pathfinding capaz de calcular las rutas que pueden seguir las unidades a través del mapa que pertenezca a una batalla. En la Figura 23 podemos ver el constructor de la clase Pathfinder que utiliza un objeto de tipo Batalla para inicializar los atributos de la clase.

```
public class Pathfinder
{
    PathfinderServices mapaServices;
    Batalla gridMap;
    PathNode[,] pathNodes;

    7 referencias
    public Pathfinder(Batalla mapa)
    {
        mapaServices = new PathfinderServices(mapa);
        gridMap = mapa;
        pathNodes = new PathNode[gridMap.tileSet.Width, gridMap.tileSet.Height];

        for (int x = 0; x < gridMap.tileSet.Width; x++)
        {
            for (int y = 0; y < gridMap.tileSet.Height; y++)
            {
                pathNodes[x, y] = new PathNode(x, y, mapaServices.GetTerrainType(x, y));
            }
        }
    }
```

Figura 23. Clase y constructor de Pathfinder.

Una vez se había creado el algoritmo para realizar el movimiento de las unidades desplegadas por el mapa, se creó una clase llamada BatallaServices en la cual se encuentran los métodos que permiten desplazar a un personaje, esta función se basa en el algoritmo

descrito previamente ya que es la forma en la que se consigue averiguar la ruta a seguir. También se requirió desarrollar una forma de detectar las unidades desplegadas por el mapa pues el algoritmo solo basaba sus rutas en base a las casillas que conformaban el mapa sin tener en cuenta si estas estaban ocupadas o no. Para ello uno de los métodos del algoritmo fue adaptado para que pudiera identificar aquellas casillas que se encontraban ocupadas por una unidad y fueran excluidas de la ruta que se estaba trazando.

6.2.3 Desarrollo de la IA de los enemigos

Otra funcionalidad fundamental del juego era la de poder atacar a otro personaje que no perteneciera al bando mismo bando, para ello teniendo completamente adaptado el algoritmo lo único que se requería era implementar un método que detectara si el lugar donde se deseaba mover una unidad estaba ocupado por una unidad enemiga y en caso de que así fuera esta se viera afectada por el ataque de la unidad que se había desplazado. El ataque podría ver modificado su efecto en caso de que el personaje hubiera utilizado su habilidad, esto requirió que se añadieran parámetros al modelo de personaje para poder identificar su habilidad y si esta podía ser activada.

Por último, quedaba poder identificar si la partida debía acabar por ello se crearon listas que contenían las unidades que se encontraban desplegadas en el mapa. Si cualquiera de estas listas llega a estar vacía significa que uno de los bandos se ha quedado sin unidades y no puede seguir la batalla, confiriendo automáticamente la victoria al bando rival.

Todo lo descrito previamente fue realizado con el objetivo de completar los requisitos que se listan en la Tabla 3:

Part1

Part2

Part3

Part4

Part5

Part6

Part7

Tabla 3. Requisitos de partida cumplidos.

Otro desarrollo importante fue la correcta implementación de una máquina de estados que pudiera hacer que los personajes no controlados por un jugador pudieran decidir qué hacer durante la partida.

Con este objetivo en mente se desarrolló una máquina de estados utilizando la librería Stateless esta es una herramienta de código abierto para. NET que nos permite crear máquinas de estado y flujos de trabajo de manera sencilla.

En un principio se trató de crear una gran cantidad de estados cada uno con su acción específica, no obstante, tras comprobar que de esta forma la complejidad de la máquina era demasiado elevada para ser utilizada de forma sencilla, se decidió agrupar las acciones en dos estados, un estado de ataque y otro estado de retirada. Después se diseñó un estado encargado de decidir a cuál de los dos estados anteriores se accedía cuando una unidad controlada por la máquina se disponía a realizar una acción. Y, por último, un estado de configuración donde se le cargaba la información necesaria a la máquina de estados para que pudiera decidir las acciones a realizar.

En cuanto a los dos estados de ataque y retirada, estos separaban las acciones que se podían realizar estando dentro de cada uno. Se diferenció entre acciones de carácter ofensivo como atacar a un personaje o aproximarse en su dirección que se agruparon dentro del estado de ataque. Y las acciones defensivas que se agruparon dentro del estado de retirada como podían ser retirarse, agruparse con compañeros o evitar el combate con unidades enemigas.

Así pues, de esta forma se consiguió una máquina de estados con 4 estados en los cuales se agrupan las diferentes acciones que puede realizar la máquina. El desarrollo de la máquina de estados permitió cumplir los requisitos que se muestran en la Tabla 4:

IA1
IA2
IA3
IA4
IA5

Tabla 4. Requisitos de la IA cumplidos.

Los requisitos que no han sido mencionados en las tablas se consiguieron tras pulir el back-end mediante el análisis de las críticas de los jugadores que rellenaron una encuesta y la creación de un proyecto de pruebas de test, todo esto se detallarán más adelante.

6.2.4 Conexión de la API con el proyecto de Unity

Tras haber realizado las pruebas necesarias, que se detallarán en profundidad en el próximo capítulo, fueron conectados la API y el proyecto de Unity.

Para realizar dicha conexión fue creado un script llamado Connectios que contenía una clase llamada Connections. Este script fue asignado a uno de los objetos presentes en la escena de Unity, el termino escena se refiere a un archivo de proyecto que contiene todos los elementos necesarios para representar un nivel, una pantalla o una parte específica de un juego o aplicación. En este script se crearon los métodos que realizaban las llamadas necesarias a la API utilizando la clase UnityWebRequest, que ya ha sido descrita en apartados anteriores. Gracias a estos métodos se obtienen los objetos representados en la base de datos que con convertidos en GameObjects e introducidos en la escena.

Capítulo 7

Implantación y pruebas

7.1 Implantación

Una vez que el videojuego se ha convertido en un producto viable, se ha procedido a publicarlo en la plataforma web itch.io. Esta plataforma es ampliamente reconocida y utilizada para la distribución y promoción de juegos independientes, proporcionando a los desarrolladores un espacio donde pueden compartir y dar a conocer sus creaciones a una audiencia global. El uso de itch.io permite que el juego esté disponible para jugadores de todo el mundo y brinda la oportunidad de recibir retroalimentación valiosa de la comunidad de jugadores independientes. El objetivo de esto es conseguir feedback de personas cercanas y más adelante de gente ajena a los desarrolladores con el objetivo de implementar mejoras en el proyecto. En la Figura 24 se puede ver la página de descarga que se creó en un principio. Con esta sencilla interfaz se lanzó el juego en la plataforma para que personas que poseyeran la contraseña pudieran descargarlo y probarlo.

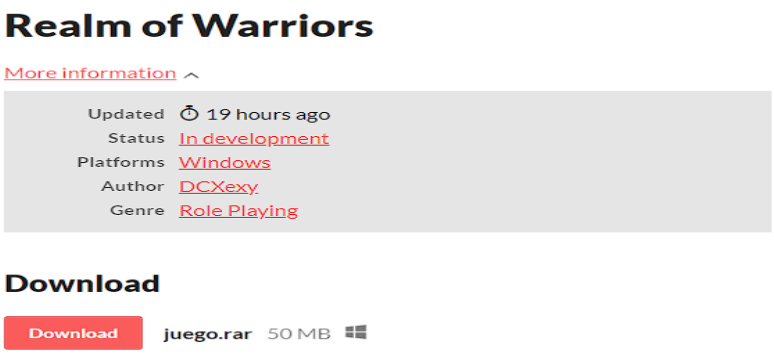


Figura 24. Página de descarga Realm Of Warriors

Junto con la publicación del juego en internet, se creó un formulario de Google Forms donde se encuestaba a los usuarios sobre diferentes aspectos del juego que se creían podían ser más conflictivos y así conocer la opinión para mejorar el proyecto. A continuación, analizaremos los resultados de esta primera encuesta lanzada.

En la Figura 25 se ve el resultado a la afirmación sobre si es un juego entretenido. Con los resultados podemos ver que los usuarios que probaron el juego disfrutaron el rato que lo jugaron, ya que más de un 50% está de acuerdo con que el juego es entretenido. Por lo que el objetivo de brindar un juego entretenido se puede dar como cumplido.

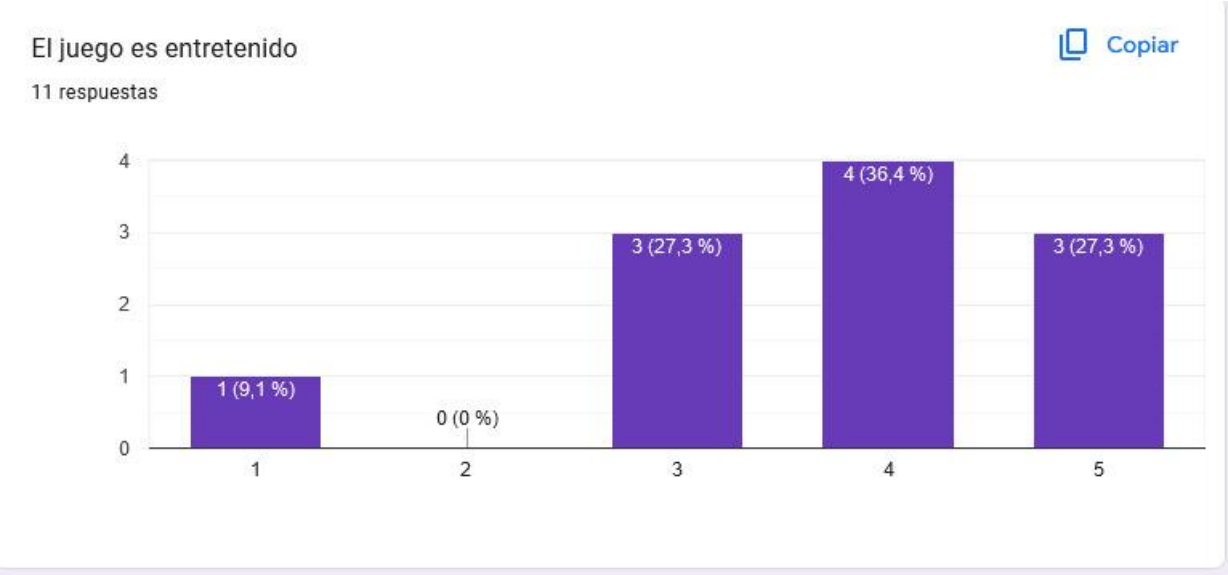


Figura 25. Resultados pregunta juego es entretenido

En la figura 26 podemos ver una pregunta que hace referencia a la fluidez con la que el juego funciona, pues al realizar llamadas a la API esto puede llegar a afectar a la continuidad de la partida. Echando un vistazo a los resultados podemos ver que en su mayoría los usuarios que han probado el juego están contentos con el rendimiento, pero hay algunos casos donde parece que la experiencia puede llegar a ser frustrante, por ello es un área donde se podría mejorar nuestro juego.

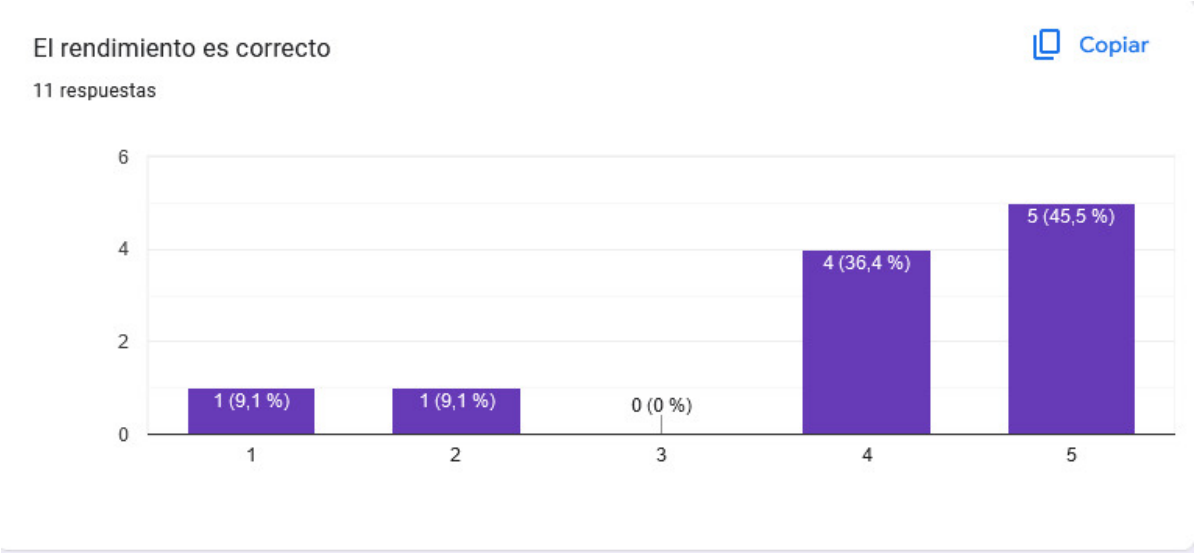


Figura 26. Resultados pregunta el rendimiento es correcto

En cuanto a la dificultad del juego se hizo una pregunta expresamente para conseguir saber la opinión de los usuarios pues es una de las principales preocupaciones que se tuvo durante el desarrollo del juego ya que quería evitar ser frustrante y complicado para aquellos jugadores que estaban iniciándose en el género de videojuegos. Tras analizar los resultados de la figura 27 parece razonable asumir que el objetivo ha sido cumplido.

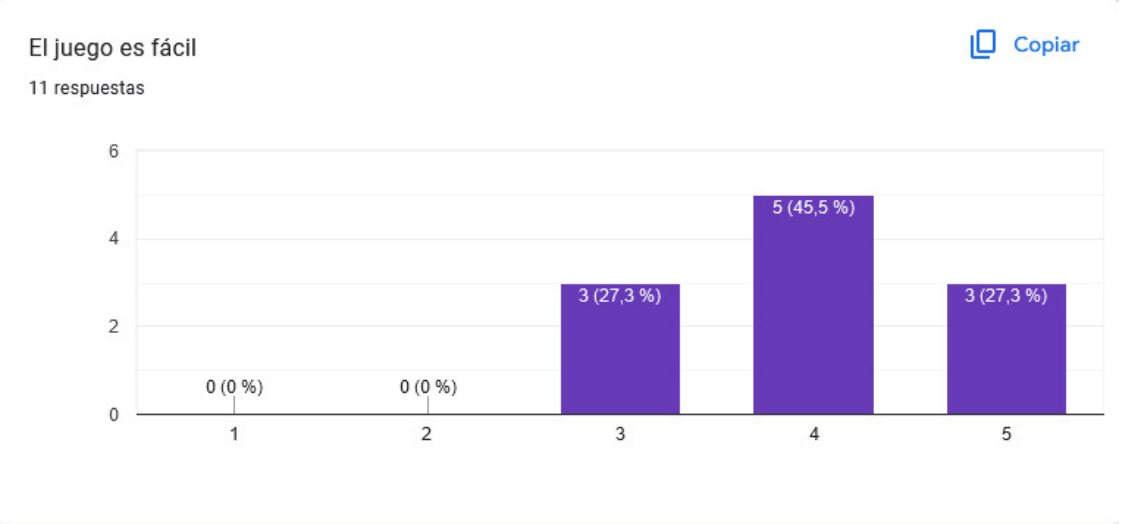


Figura 27. Resultados es un juego fácil

Con la pregunta de si el funcionamiento de la IA buscamos saber si a los usuarios las acciones controladas por el ordenador tienen cierta lógica y hacen del juego un reto entretenido. Como podemos ver en la figura 28 hay gran variedad de opiniones y solo hay un usuario que cree que el funcionamiento de la IA es totalmente correcto. A la vista de estos resultados el funcionamiento de la IA es un punto importante de mejora.

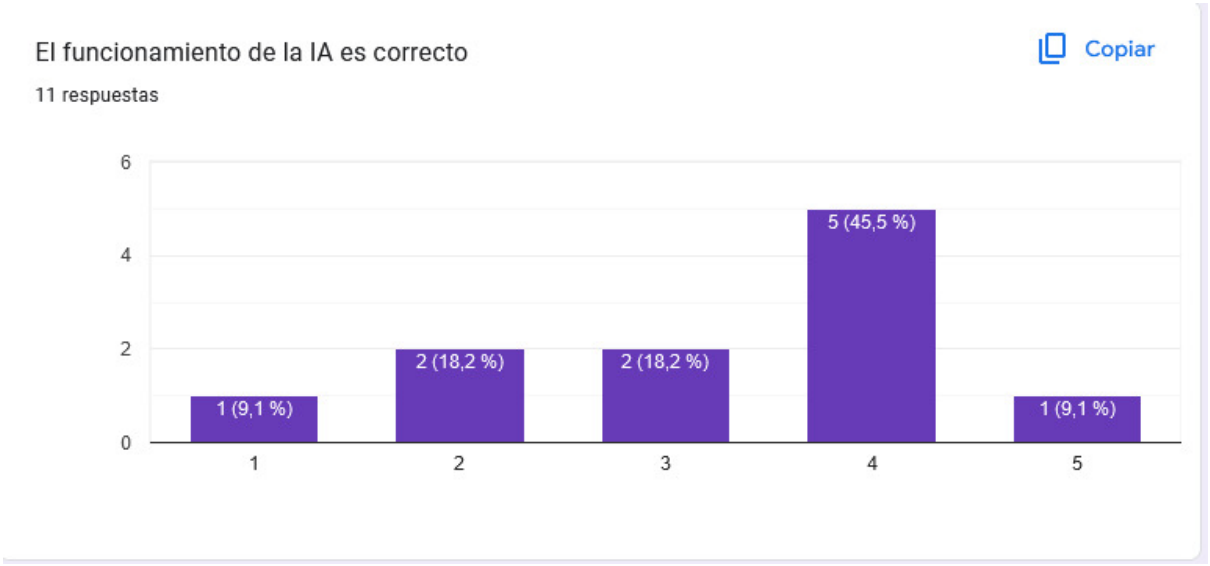


Figura 28. Resultados funcionamiento de la IA

Se preguntó por el balance general del juego y a la vista de los resultados parece que el trabajo de balance del juego está bien logrado. No hay ninguna valoración inferior a 3 con lo cual se cree que se ha hecho un buen trabajo con el balance general del juego pese a que se trabajará para mejorar la sensación de balance general entre los personajes enemigos y aliados. En la figura 29 se pueden ver los resultados de la encuesta en los que se han basado las conclusiones anteriores.

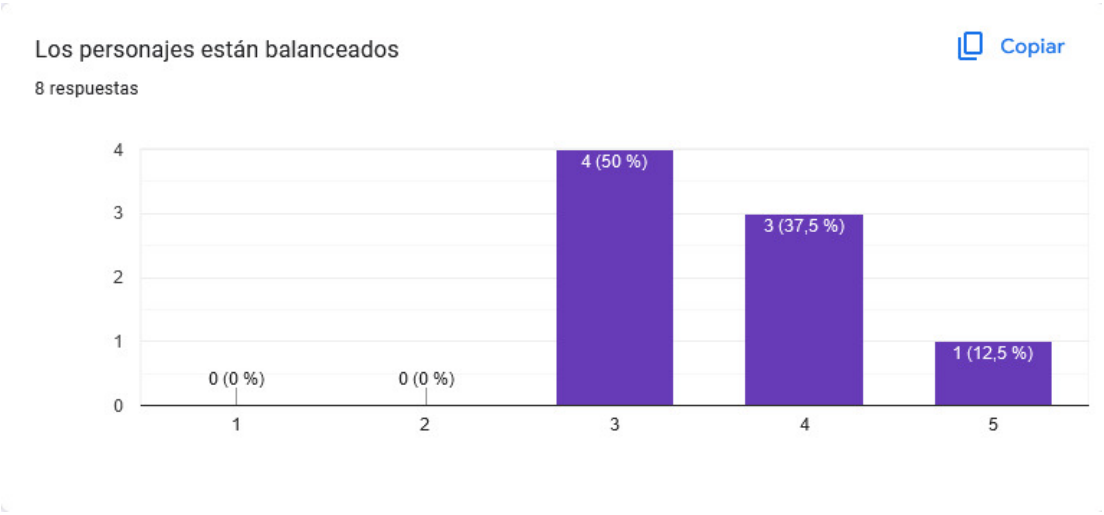


Figura 29. Resultados personajes balanceados

Siguiendo con los resultados de la siguiente pregunta realizada en el cuestionario, en la figura 30 podemos ver que la mayoría de personas que rellenaron la encuesta indicaron que preferían que se añadieran más modos de juego en vez de una variedad de mapas mayor, esto quiere decir que el back debería estar preparado para poder soportar aplicaciones sin comprometer su estructura pues la introducción de modos de juego puede llegar a requerir modificaciones en el código de la misma, por eso es necesario que se cumpla que el back sea fácilmente modificable.

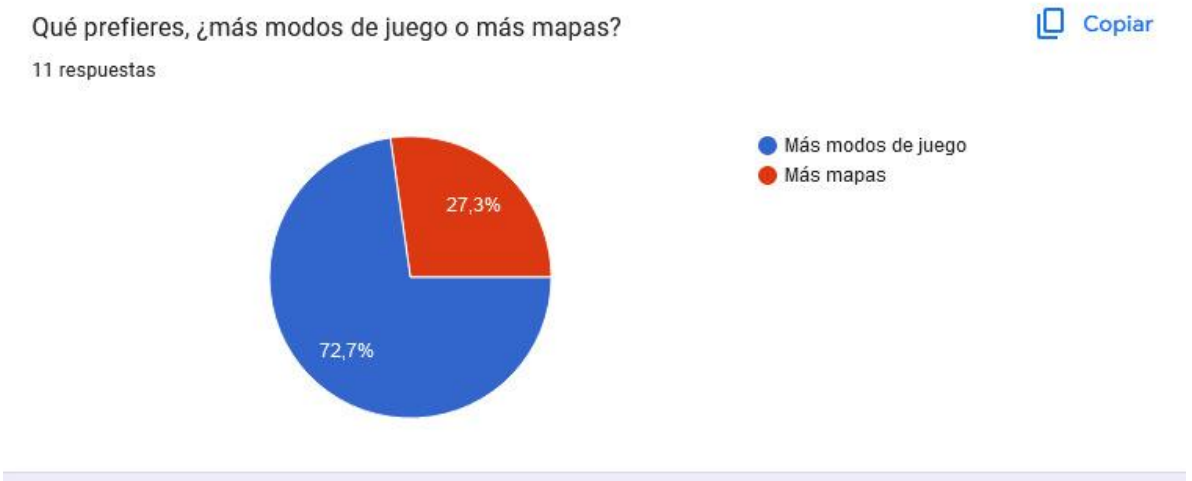


Figura 30. Resultados pregunta más modos de juego

Por último, haciendo referencia a la última pregunta del cuestionario encontramos que gran parte de los encuestados querrían un modo competitivo esto implicaría que la eficiencia de las consultas realizadas a la API creada tienen que ser muy alta pues jugar de forma competitiva contra otros jugadores requiere una mayor concentración y cualquier falla de rendimiento por parte del videojuego puede suponer una experiencia desagradable para los jugadores. Podemos ver los resultados de la pregunta en la Figura 31.

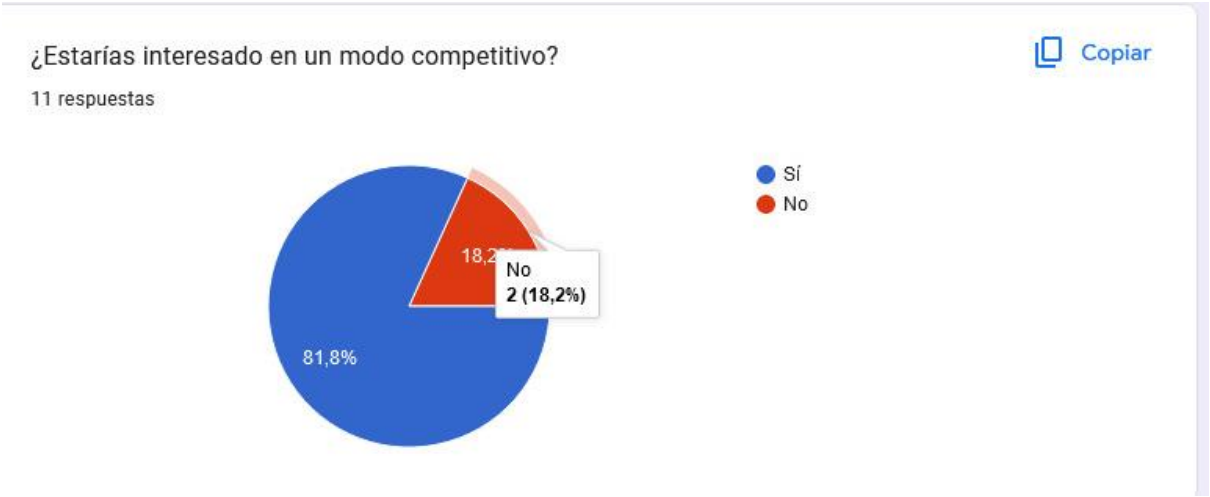


Figura 31. Resultados pregunta competitivo

7.2 Pruebas

Para asegurar el correcto funcionamiento de toda la solución se creó un proyecto de Prueba en NUnit, un framework que trata de facilitar la escritura, ejecución y organización de pruebas unitarias para el lenguaje de programación C# (y otros lenguajes de la plataforma .NET) con el objetivo de ayudar con el desarrollo de proyectos software.

Figura 32. Logo NUnit

Se creó un proyecto de Prueba donde se probaron las diferentes llamadas a los métodos de los controladores de cada modelo, las clases de servicio que proporcionaban la jugabilidad y a la clase de gestión de estados que se encargaba de la IA. En la figura 33 se puede ver una captura con las clases pertenecientes al proyecto.

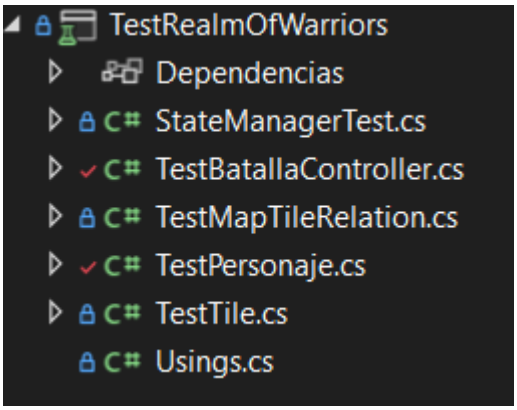


Figura 33. Proyecto de Pruebas de Realm Of Warriors

Se crearon clases de test para cada uno de los modelos, sus controladores y sus repositorios para asegurar el correcto funcionamiento de estas y poder en todo momento cerciorarse de que el proyecto funcionaba de forma correcta, en una primera instancia estos métodos imprimían por pantalla los resultados, por ejemplo si recuperaban los datos de un personaje almacenado en base de datos se imprimía la información correspondiente a sus estadísticas y se comprobaba que coincidían con el objeto almacenado en la base de datos. En la Figura 34 podemos ver un ejemplo de pruebas realizadas a mano.

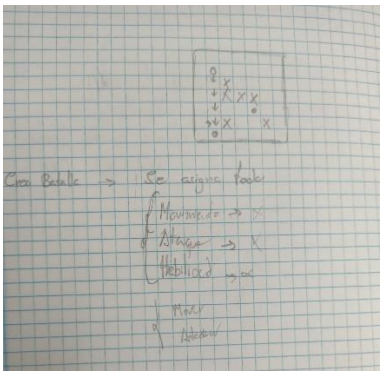


Figura 34. Ejemplo de pruebas hechas a mano

Una vez se aseguró que los resultados eran los esperados, se automatizaron las pruebas para cada una de las acciones posibles, borrar, modificar, añadir y obtener la información para cada uno de los modelos que se buscaba almacenar en la base de datos, así pues, con resultados esperables de realizar estas acciones se crearon métodos que realizaban cualquiera de las acciones previamente mencionadas y se comprobaba que eran coincidentes.

El proceso de las clases de servicio fue similar. En una primera instancia los métodos de prueba mostraban los resultados de las acciones que puede realizar el jugador y la IA sobre la batalla como lo son mover sus unidades, atacar a las unidades rivales o utilizar habilidades. Cuando se comprobó el correcto funcionamiento de todas estas funciones se automatizaron las pruebas al igual que se hicieron con las pruebas referentes a los modelos.

De esta forma se aseguró un correcto funcionamiento de los métodos y clases implementados en el back-end para poder centrarse de forma posterior en la implementación del videojuego.

Podemos ver un ejemplo de método de pruebas en la Figura 35 en la cual se hace una carga completa desde la base de datos de una batalla utilizando los métodos de repositorios para realizar los accesos a la base de datos y conseguir la información necesaria para crear el objeto Batalla.

```
[TestMethod]
0 referencias
public async Task LoadBattle()
{
    int id = 5;
    var relaciones = await mapTileRelationRepositorio.GetRelacionMapaById(id);
    var mapDimensions = await mapaRepositorio.GetTileSetById(id);

    var characters = await personajeRepositorio.GetPersonajeByMapaId(id);

    batalla.tileSet = mapDimensions;
    batalla.enemigos = new List<Personaje>();
    batalla.personajes = new List<Personaje>();

    foreach (var unidad in characters)
    {
        if (unidad.Enemy)
            batalla.enemigos.Add(unidad);
        else
            batalla.personajes.Add(unidad);
    }
    batalla.mapaBatalla = new Tile[mapDimensions.Width, mapDimensions.Height];
    foreach (var realacion in relaciones)
    {
        tiles.Add(await tilesRepositorio.GetTileById(realacion.tileReference));
    }

    for (int x = 0; x < mapDimensions.Width; x++)
    {
        for (int y = 0; y < mapDimensions.Height; y++)
        {
            int v = (x * mapDimensions.Height) + y;
```

Figura 35. Ejemplo método de Prueba

Capítulo 8

Conclusiones y trabajos futuros

8.1 Conclusiones

El desarrollo del Back-End de Realm Of Warriors ha supuesto toda una experiencia y reto en cuanto a lo personal se refiere. El videojuego era una idea surgida del fanatismo por una saga de videojuegos, pero poder desarrollar un juego del mismo género fue realmente gratificante.

Se puede decir que Realm Of Warriors ha cumplido los objetivos que se propusieron en un inicio ya que ha servido para dar una aproximación sencilla de los juegos de rol táctico que tanto afán nos generan.

El desarrollo del proyecto estuvo plagado de inconvenientes en sus inicios, el objetivo de conseguir desarrollar un back funcional para el videojuego supuso un reto mayor de lo esperado pues no se estaba suficientemente familiarizado con las tecnologías que se escogieron para cumplir los diferentes objetivos que se plantearon desde un inicio. Hubo que replantear en diversas ocasiones la estructura de la API, los métodos y diferentes componentes del producto final. A pesar de todo esto se logró un resultado satisfactorio.

A nivel personal, el desarrollo de este proyecto supuso un reto autoimpuesto para poder aprender diversas cosas nuevas, pues si bien durante el estudio del grado había muchas asignaturas que se encargan de hacernos experimentar el desarrollo de diferentes proyectos, el hacerlos en grupo ocasiona que en muchas ocasiones no se aprenda como se debe el correcto empleo de todas las tecnologías utilizadas. Por ello, haber conseguido desarrollar el back-end de Realm Of Warriors de forma independiente y haber logrado su finalización ha sido una experiencia enriquecedora y gratificante.

8.2 Trabajos futuros

Como se ha mencionado en apartados anteriores el desarrollo de Realm Of Warriors ha sido un medio de aprendizaje para poder crecer de forma personal y profesional. No se descarta por completo una continuación del proyecto con un enfoque más profesional. Pero el entusiasmo de desarrollar este proyecto fue con el objetivo de pulir las habilidades necesarias y afianzar los conocimientos para entrar en el mundo laboral y poder ser un profesional a la altura.

Si se decidiera continuar con el proyecto como ya se dijo al principio este es una aproximación sencilla a los videojuegos de rol táctico, por ello la solución implementada tiene soporte para las mecánicas más sencillas, no obstante, si fueran necesarias mecánicas más pulidas y conseguir un videojuego que esté a la altura de un videojuego comercial el back permite una sencilla modificación e inclusión de las clases para conseguir un mejor producto final. También se podría migrar la plataforma de desarrollo pues la independencia del back permite este cambio con sencillez.

Pero como se ha querido dejar ver, con el trabajo realizado durante el desarrollo de este proyecto, se espera que la experiencia ganada en sirva para tener una buena base cuando se dé el salto al mundo laboral.

Bibliografía

[1] Ajedrez - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Ajedrez

[2] Mejores juegos de rol táctico - Mundo Deportivo. Consultado en https://www.mundodeportivo.com/alfabeta/listas/mejores-juegos-rol-tactico

[3] Fire Emblem - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Fire_Emblem

[4] Triangle Strategy - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Triangle_Strategy

[5] Final Fantasy Tactics - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Final_Fantasy_Tactics

[6] Unity - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Unity_(motor_de_videojuego)

[7] Azure - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Microsoft_Azure

[8] Microsoft SQL Server Management Studio - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/Microsoft_SQL_Server_Management_Studio

[9] GitHub - Wikipedia la enciclopedia libre. Consultado en https://es.wikipedia.org/wiki/GitHub

APÉNDICE A

Objetivos desarrollo sostenible

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				x
ODS 2. **Hambre cero.**				x
ODS 3. **Salud y bienestar.**		x		
ODS 4. **Educación de calidad.**			x	
ODS 5. **Igualdad de género.**				x
ODS 6. **Agua limpia y saneamiento.**				x
ODS 7. **Energía asequible y no contaminante.**				x
ODS 8. **Trabajo decente y crecimiento económico.**				x
ODS 9. **Industria, innovación e infraestructuras.**	x			
ODS 10. **Reducción de las desigualdades**.			x	
ODS 11. **Ciudades y comunidades sostenibles**.				x
ODS 12. **Producción y consumo responsables.**				x
ODS 13. **Acción por el clima.**				x
ODS 14. **Vida submarina.**				x
ODS 15. **Vida de ecosistemas terrestres.**				x
ODS 16. **Paz, justicia e instituciones sólidas.**				x
ODS 17. **Alianzas para lograr objetivos.**				x

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

Para este proyecto se procuró cumplir tantos objetivos de desarrollo sostenible como fuera posible, pero al tratarse de un videojuego, esto no resultaba tarea fácil. Muchos de estos objetivos son muy complicados de cumplir en un videojuego o directamente imposible, ya que no se encuentran relacionados con este campo. Por este motivo muchos de los ODS han sido marcados como 'no procede', por ejemplo y como es obvio un videojuego no puede acabar con el hambre en el mundo.

Este videojuego tiene relación con el objetivo de salud y bienestar, pues los videojuegos pueden tener un impacto positivo en la salud mental al proporcionar una vía de escape y entretenimiento. Además de que, al ser un juego de estrategia, puede llegar a mejorar la concentración, la toma de decisiones, la planificación estratégica y la memoria. Todo esto ayuda a tener una mente sana y ejercitada gracias a la necesidad de desarrollar todas estas habilidades para superar los diferentes niveles del videojuego.

Siguiendo con los objetivos, también se puede relacionar, aunque en menor medida con el objetivo de educación de calidad, pues los videojuegos pueden ayudar en el desarrollo de habilidades cognitivas, como la resolución de problemas o la toma de decisiones, lo que puede ser fundamental para una educación de calidad ya que prepara para enfrentar desafíos reales. Este videojuego también ofrece un aprendizaje continuo y autodirigido pues al tener una historia lineal donde los desafíos a superar aumentan poco a poco a lo largo de la misma.

Otro objetivo de desarrollo sostenible que se ha intentado relacionar con el proyecto el ods número 9 pues El desarrollo de un back-end independiente de la plataforma implica la creación de herramientas y sistemas que permiten a los desarrolladores de videojuegos acceder a funciones y recursos esenciales de manera más eficiente y versátil. Esto promueve la innovación tecnológica al simplificar el proceso de desarrollo y permitir a los desarrolladores centrarse en la creatividad y la calidad del juego en lugar de preocuparse por las limitaciones de la plataforma. A su vez también puede reducir los costos de desarrollo al proporcionar una base sólida y reutilizable para la creación de juegos en diversas plataformas.

El proyecto también puede llegar a ser relacionado con el objetivo de reducción de las desigualdades, pues al haber sido publicado en la página web itch.io de forma gratuita, se da un acceso equitativo a todas aquellas personas que posean un equipo para descargar el juego, proporcionando de esta forma accesibilidad para personas bajo cualquier condición.

En resumen, se ha intentado implantar los ODS a este proyecto en la medida de lo posible, si bien esto ha resultado complicado ya que el desarrollo de un videojuego no siempre está relacionado con estos objetivos el haber desarrollado un back-end, ha permitido encontrar relación con varios de ellos que de otra forma hubiera sido imposible.


UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA


etsinf

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APÉNDICE B

Realm Of Warriors – Documento de diseño de juego

Game Design Document

Desarrollado por Juan Payá Poveda, Javier Calatayud Ferre y Sergio Pérez Gascón

Realm of Warriors

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Índice

1. Planificación

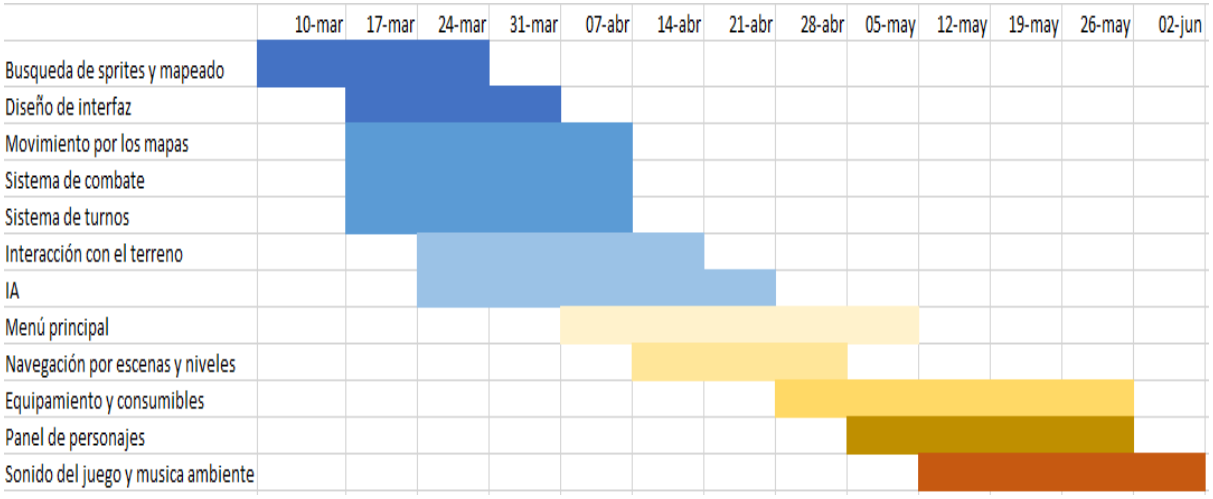
1.1Equipo de trabajo

Realm of Warriors es un proyecto desarrollado siguiendo un flujo de trabajo interdisciplinar. Nuestro equipo está formado por tres personas del Grado de Ingeniería Informática.

Grado de Ingeniería Informática	
Nombre	**Funciones**
Javier Calatayud Ferre	Inclusión de sonidos, funcionalidad del menú y elección de bando.
Juan Payá Poveda	Funcionalidad del recorrido por el mapa.
Sergio Pérez Gascón	HUD e interfaces

1.2 Diagrama de Gantt

A continuación, se presenta el diagrama de Gantt de las tareas que se está siguiendo durante el desarrollo. Este flujo de trabajo ha sido diseñado utilizando la plataforma *HacknPlan*.



2.General

2.1 Género

Realm of warriors es un videojuego de rol táctico.

2.2 Plataformas

PC.

3.3 Referentes

Gameplay de Fire Emblem

La saga Fire Emblem es el principal referente, tanto en gameplay como estética. Nos hemos basado en ella para los diseños de niveles, personajes y tipo de pelea.

Gameplay Triangle Strategy

Triangle Strategy es otro juego en el que nos hemos basado pues tiene mecánicas similares a Fire Emblem, pero aportando mecánicas nuevas a la hora de las batallas.

2.4 Público objetivo

El juego está especialmente dirigido hacia los usuarios que busquen experimentar una experiencia bélica a través de un juego de rol táctico en el que la estratégia será la clave, no obstante también podrá ser disfrutado por todo tipo de jugadores que busquen un juego de estratégia y acción.

2.5 Clasificación PEGI

El juego queda clasificado dentro de la categoría PEGI 12, al verse afectado por el siguiente criterio de clasificación: *"En esta categoría pueden incluirse los videojuegos que muestren violencia de una naturaleza algo más gráfica hacia personajes de fantasía y/o violencia no gráfica hacia personajes de aspecto humano o hacia animales reconocibles"*.

2.6 Diferenciación comercial

El juego tratará de ofrecer una jugabilidad completa con una historia diferente para los dos reinos que existen en el mundo del juego, tratando de hacer que el jugador se sienta como el comandante de un ejército mientras planea estratégias para acabar con los ejércitos rivales sea cual sea su elección de bando.

3. Gameplay

El planteamiento es sencillo. El jugador podrá mover a los personajes de su bando por un mapa diseñado como un tablero, en el cual estarán dispuestas las unidades de ambos bandos.
Saca provecho de tus unidades. Cada unidad poseerá atributos, armas y habilidades únicas que le permitirán destacar en diferentes situaciones. Contarán con estadísticas, como pueden ser salud, defensa, ataque físico, velocidad, etc. que afectarán a distintos factores como el daño que realizan o reciben o la cantidad de casillas que se pueden desplazar.

Plantea tu estrategia. Se trata de un combate por turnos, por lo que las unidades sólo podrán realizar una acción por turno del jugador, lo mismo aplica a las tropas enemigas.
Cuida a tus tropas. Una unidad podrá atacar a un enemigo que se encuentre dentro de su alcance moviéndose hacia él, pero puede acabar rodeada y siendo eliminada.

Las unidades tanto enemigas como aliadas pertenecen a una clase u oficio, por lo que cada unidad contará con habilidades específicas de la clase correspondiente, así como distintas estadísticas que se adapten. Cuando una unidad aliada derrote a un enemigo, obtendrá puntos de experiencia que le permitirán subir de nivel para aumentar sus estadísticas y obtener nuevas habilidades.

3.1 Gameplay básico

El jugador será capaz de mover la unidad seleccionada a la casilla deseada dentro de su rango de movimiento, marcado por las casillas que se encuentran resaltadas en verde. Además, podrá ver las unidades enemigas a su alcance (marcadas en rojo) y las unidades aliadas (marcadas en azul). En la parte superior izquierda de la pantalla podemos ver un panel que nos indicará si es nuestro turno de atacar, o el de la IA.

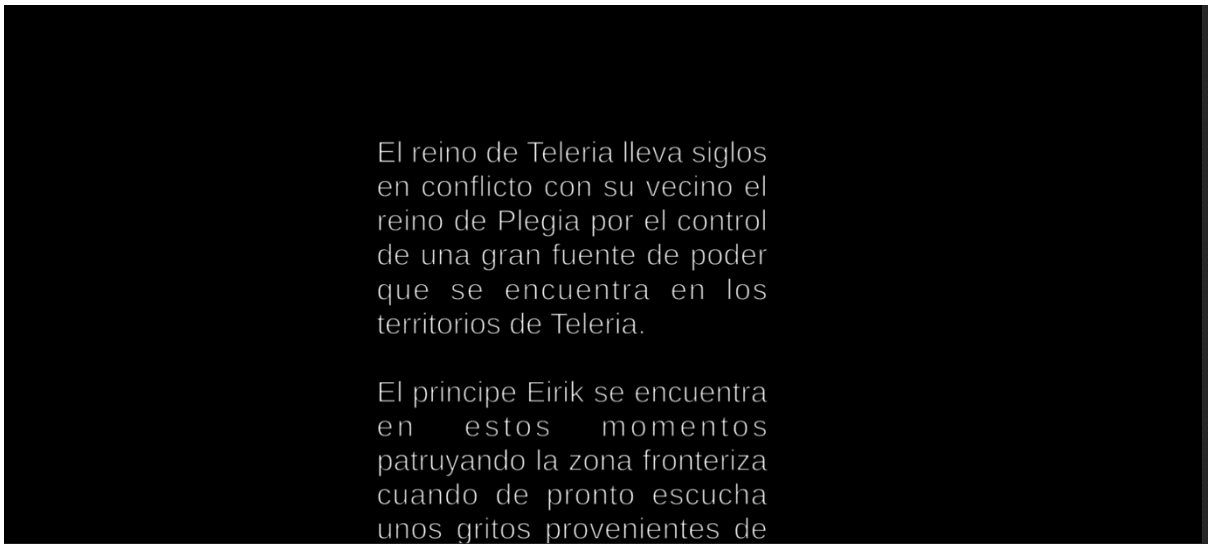
Además, si colocamos el cursor sobre un enemigo podremos ver su rango de movimiento, marcado por las casillas en rojo.

4.Historia

La historia se desarrolla en un mundo de fantasía donde dos grandes naciones están en guerra por el control de una fuente de energía mágica. La nación del este, busca obtener la fuente de energía mágica para poder someter a su reino vecino, mientras que el reino del oeste busca mantener la paz y evitar que su vecino inicie una espiral de destrucción que consumirá todo a su paso. En el bando del reino del oeste el jugador verá como un reino con en vías de desarrollo, será asediada por otro reino buscando obtener su valiosa fuente de energía viéndose obligada a tomar las armas y defender su territorio, con lo que las batallas se centrarán en la defensa de las fortalezas que circundan el territorio.

Imagen de un nivel de asalto a fortaleza Fire Emblem

Antes y después de cada nivel habrá una pantalla en la que se mostrará un pequeño fragmento de la historia.



5. Niveles de juego

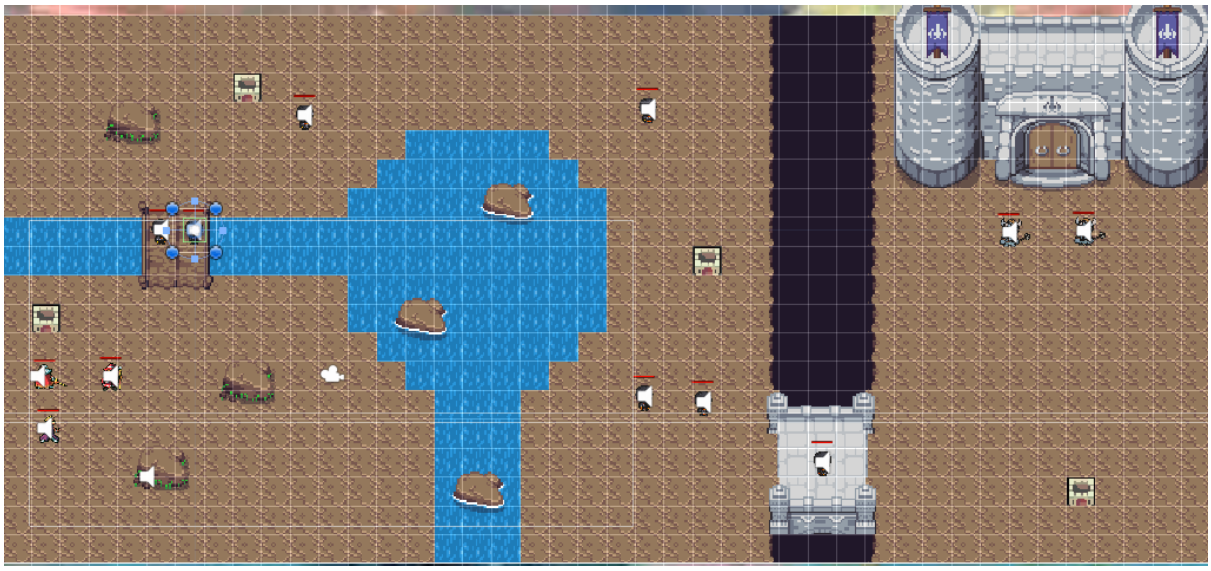
Cada nivel consistirá en un tablero que representará una zona donde se llevará a cabo la batalla, el objetivo será derrotar al comandante enemigo o completar determinadas misiones. Según avancen los niveles se unirán nuevas tropas a nuestro ejército aumentando así las posibilidades a la hora de planificar estratégias.

El juego cuenta con un nivel a modo de tutorial que servirá para introducir al jugador las mecánicas básicas del juego. El jugador contará con una unidad capaz de realizar acciones básicas como moverse y atacar y con una habilidad especial la cual hace más daño al golpear. Deberá enfrentarse a una serie de enemigos con el objetivo de comprender el estilo de juego. El tutorial presenta una serie de textos que indicarán al jugador como ha de proceder.

Hay un total de 6 niveles, a cada cual más extenso que el anterior. Además, en cada nivel se incorporará una nueva unidad a nuestro equipo. A medida que vayamos avanzando por los niveles también encontraremos nuevos enemigos, los cuales serán cada vez más poderosos.

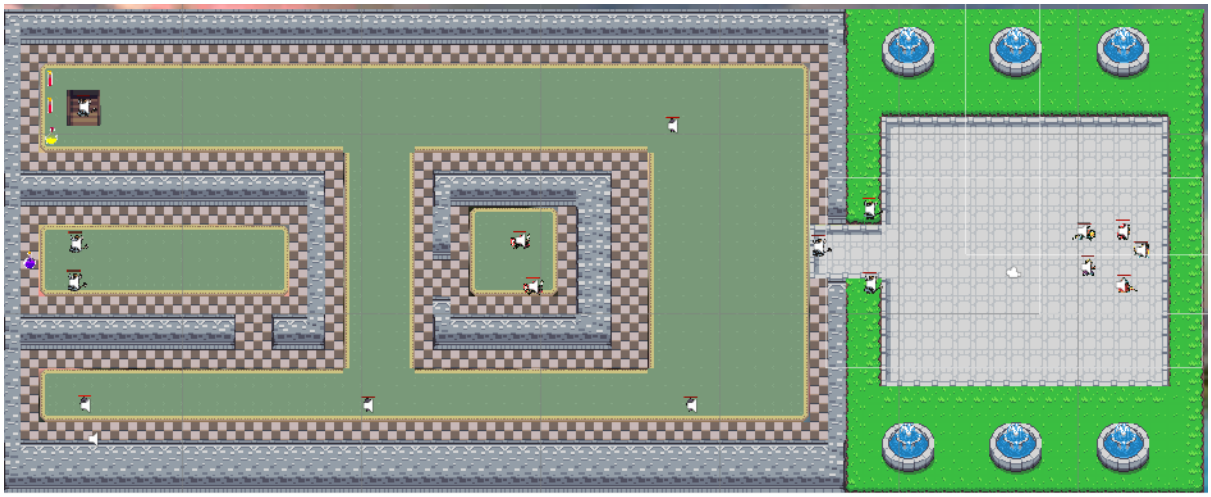
El segundo nivel sucederá en una fortaleza junto al mar, y presentará un nuevo tipo de enemigo: el caballero.
Este enemigo contará con gran resistencia, a cambio de una menor distancia de movimiento.
Además, obtendremos un nuevo aliado, el mago, el cual puede lanzar hechizos a distancia.

En el tercer nivel nos encontraremos en un páramo a las afueras de un castillo, el cual estará protegido por dos nuevos tipos de enemigos: los golems y los minotauros.
En este nivel obtendremos a nuestro tercer personaje, el mago blanco, el cual dispone de una habilidad para curar a los aliados.



En el cuarto nivel habremos localizado nuestro objetivo: el castillo en el cual se encuentra el líder enemigo. Pero para entrar, primero deberemos derrotar a los enemigos que se encuentran protegiéndolo gracias a la ayuda del cuarto integrante de nuestro equipo, el arquero, el cual dispara a distancia y tiene una habilidad de tirar una flecha mágica que hace más daño que las anteriores y va dirigida al enemigo sin importar el rango.
En este nivel deberemos enfrentarnos a caballeros que protegen la entrada y a otro tipo de enemigo nuevo que se encuentra como avanzadilla a las afueras del castillo: los wraiths.

En el quinto nivel nos encontraremos en la entrada del castillo, la cual está protegida por unos caballeros además de algunos minotauros.
Para este nivel, contaremos con nuestra nueva y última incorporación: el lancero, el cual cuenta con una gran movilidad y cuando usa su habilidad los ataque le hacen la mitad de daño, en cambio tiene poco ataque.



Finalmente, llegaremos al último nivel, en el cual deberemos derrotar al jefe final, el cual es uno de nuestros antiguos aliados. Éste constará con una gran cantidad de vida y daño, y estará acompañado de sus secuaces golems.

6. Personajes

6.1 Protagonista

El protagonista es Eirik. Joven príncipe del reino de Arvandor, dotado de una gran habilidad con la espada. Aunque al principio se muestra arrogante, un trágico acontecimiento hace que se cuestione su papel en la guerra que asola su reino. Erik es valiente y leal a sus amigos, pero también tiene un lado más vulnerable que lo hace más humano y empático.

6.2 Antagonista

El antagonista es Zephyr. Un poderoso mago oscuro, antiguo enemigo del reino Arvandor. Zephyr es un hombre frío y calculador, sus súbditos lo temen y le admiran a partes iguales. Su objetivo es conquistar todo el continente y establecer un nuevo orden bajo su mandato. A medida que avanza la historia, se revela que su obsesión por poder está motivada por un pasado traumático.

6.3 Aliados

Mago

Compañero de nuestro protagonista y poderoso mago que será capaz de lanzar diferentes hechizos a los enemigos desde una larga distancia.

Arquero

Personaje que se une a nuestros compañeros para vengar a su aldea, que fue conquistada por el ejército enemigo. Tiene la capacidad de realizar ataques desde una cierto rango y la habilidad de disparar una flecha máica sin importar el rango

Lancero

Valiente guerrero que lucha por la libertad de su reino. Cuenta con una gran fuerza física y es capaz de resistir una gran cantidad de golpes. A pesar de su poco daño es un gran corredor y su habilidad hace que reciba la mitad del daño que se le aplica

Hechicero

Un mago blanco que es capaz de curar a sus aliados gracias a sus habilidades sanadoras.

6.4 Enemigos

Seta

Enemigo básico encontrado principalmente en los bosques del reino. Es bastante débil, aunque a veces se juntan en manada y atacan aldeas indefensas.

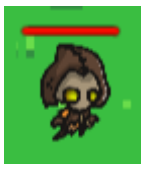
Caballero

Primer enemigo que encontraremos perteneciente al reino rival. Cuenta con una gran cantidad de puntos de salud, aunque su movilidad se ve reducida debido a su pesada armadura.

Golem

Otro de los enemigos básicos del juego. Se trata de un enemigo bastante similar al caballero en cuanto a estadísticas y función.

Wraith



Enemigo muy poco resistente, aunque muy veloz, por lo que cuenta con una elevada distancia de movimiento.

Minotauro

La última línea de defensa del ejército enemigo. Cuenta con una gran cantidad de ataque, por lo que hay que ser cuidadoso a la hora de enfrentarse a él.

7. Elementos de juego

7.1 Escenarios

Los escenarios consistirán en una serie de mapas con diferentes terrenos sobre los que tendrá lugar la batalla, cada mapa tendrá sus respectivos obstáculos, terrenos que confieren ventajas a las unidades y terrenos que afectarán a las unidades de forma diferente.
Se podrá interactuar con el terreno de distintas formas durante el combate, dando lugar a la posibilidad de plantear distintas estrategias.

7.2 Unidades

Las unidades poseerán un panel similar al de la imagen donde se podrá ver la información relativa al personaje seleccionado.
Se pueden ver las estadísticas de cada personaje, las cuales son el nombre, nivel, salud actual y máxima, daño físico, daño mágico y defensa.

Cada unidad pertenece a una clase distinta, por lo cual tiene una habilidad única que lo diferencia del resto. Cada una de estas habilidades contará con un tiempo de enfriamiento, por lo que habrá que esperar cierta cantidad de turnos para volverla a utilizar.

Paladín

Blande su espada y realiza una gran cantidad de daño a un enemigo.

Mago

Lanza una bola de fuego a distancia a un enemigo.

Arquero

Dispara una flecha con daño aumentado a un enemigo.

Lancero

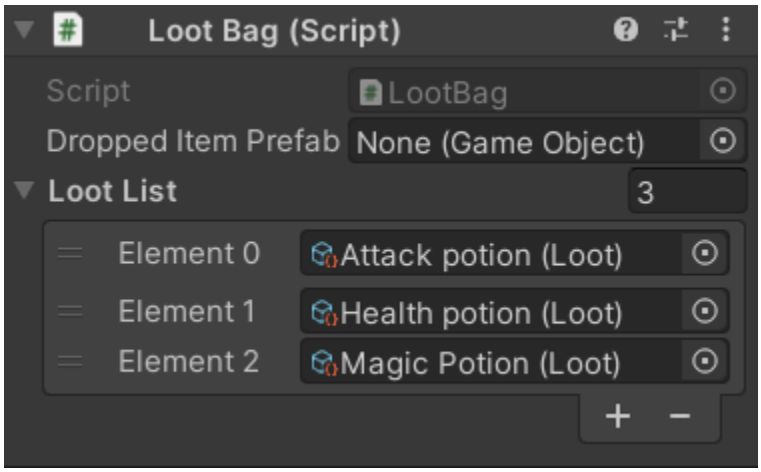
Reduce el daño recibido a la mitad.

Hechicero

Cura a un aliado seleccionado.

7.3 Sistema de loot e inventario

Cada enemigo cuenta con una Loot Bag, a la cual se pueden añadir distintos consumibles con una probabilidad de ser dropeados al morir. Cuando un enemigo dropee un item, el jugador podrá obtenerlo al pasar por encima y éste se añadirá al inventario



El inventario cuenta con 5 slots a los cuales se añadirán los diferentes objetos. Si ya tenemos un objeto de ese tipo, se incrementará el stack.

Poción de daño (Amarilla): Aumenta el daño físico de un aliado en 20 durante 3 turnos.

Poción de salud (Roja). Cura a un aliado 20 puntos de salud.

Poción de magia (Morada). Aumenta el daño mágico de un aliado en 20 durante 3 turnos.

7.4 Sistema de turnos y combate

El combate está basado en un sistema de turnos, por lo cual una vez todos los aliados hayan realizado una acción será el turno del enemigo. Durante el turno un personaje podrá moverse o utilizar una habilidad si se encuentra a rango. Utilizar un consumible no gastará tu turno, aunque no podrás utilizarlo si ya tienes el efecto de un consumible activo.

Al atacar, se calcula el daño realizado en función de varios factores. Primero, todos los ataques y habilidades cuentan con un daño base. De esta forma, si hubiera algún efecto que redujera el daño de un personaje, éste nunca podría realizar daño nulo. A este daño se le suma el daño físico o mágico del personaje, dependiendo de qué habilidad se utilice. Finalmente, este daño se restaría a la defensa del personaje objetivo para obtener el daño total realizado.

8. Interacción

8.1 Cámara

La cámara será estilo “top-down”, también conocida como vista de pájaro, por lo que podremos ver todo el campo de batalla desde arriba.

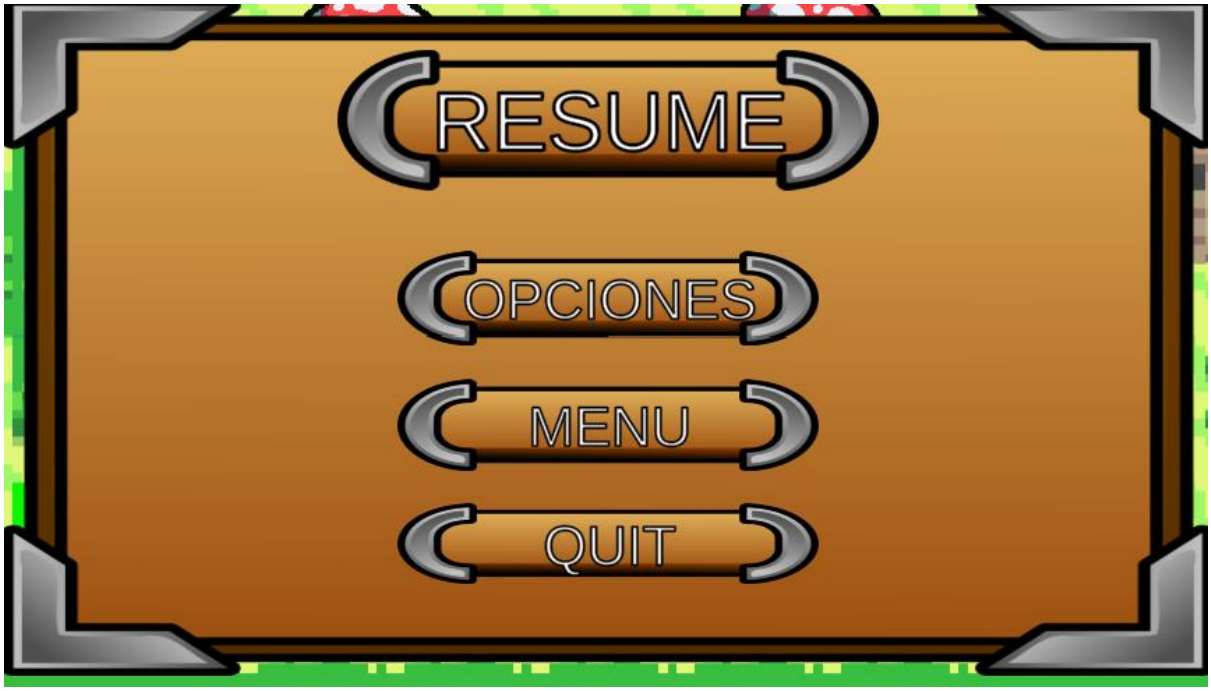
El juego trataría de ofrecer una completa aventura para cualquiera de los dos bandos que el jugador escoja, tratar de que el jugador sienta realmente que sus decisiones durante las batallas repercuten en el desenlace de las mismas.

El proyecto será complejo puesto que el desarrollo de una historia y jugabilidad para 2 bandos será largo y tedioso. Tendrá que tener una historia coherente en ambos casos y unos niveles que ofrezcan nuevas experiencias para ambas rutas de juego.

8.2 HUD

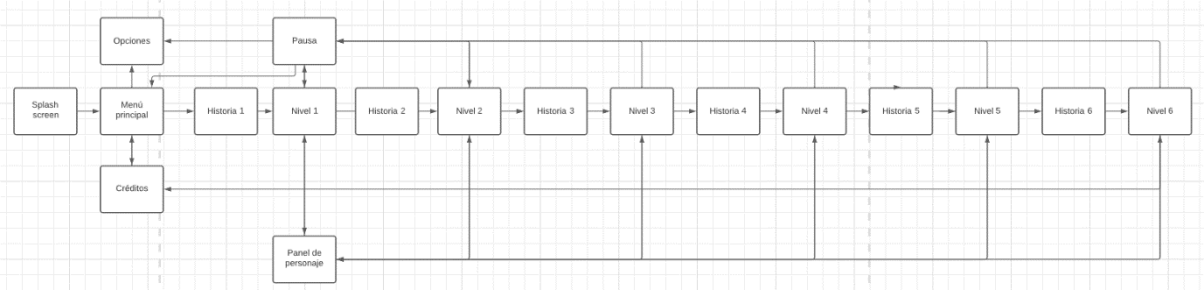
El HUD presenta una estética típica de juegos de fantasía basados en la época medieval, con el objetivo de adaptarse al concepto que busca representar el juego. Este patrón se puede ver representado en distintos elementos como pueden ser los botones de los menús o los propios mapas. Actualmente el juego cuenta con un menú principal desde el cual se puede acceder tanto al primer nivel como a las

pantallas de opciones y créditos. Desde el primer nivel, podremos acceder a un menú de pausa con distintas opciones, así como el panel de estadísticas del personaje que tengamos seleccionado. Cada vez que superemos un nivel, aparecerá una pantalla que nos cuenta el progreso de la historia, tras la cual accederemos al siguiente nivel.

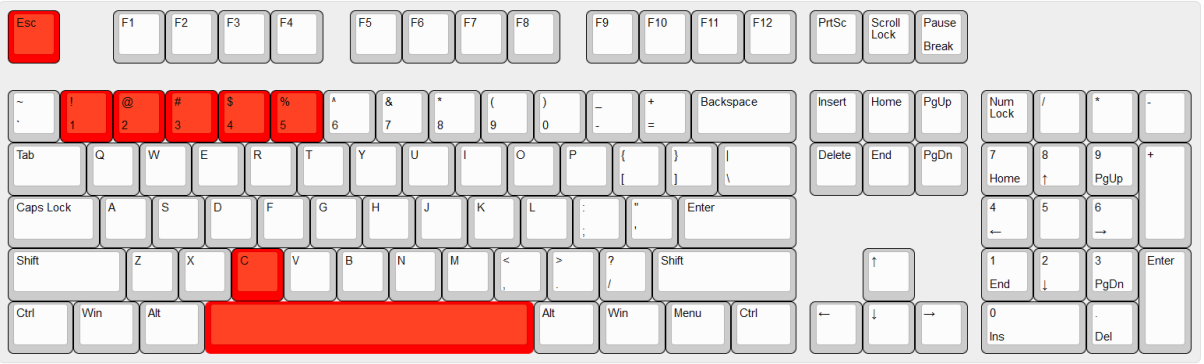


8.3 Diagrama de navegación

En el siguiente diagrama se muestra la navegación actual entre las distintas pantallas del juego.



8.4 Mapa de teclas



Escape - Abrir menú de pausa/saltar escena de historia.
C - Acceder al panel del personaje seleccionado.
Espacio - Avanzar durante un diálogo
Números 1-5 - Utilizar los slots del inventario.
Click izquierdo - Cualquier otra acción que se desee realizar, como seleccionar un personaje, atacar, o pulsar un botón del menú.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

APÉNDICE C

Glosario

API - (del inglés, application programming interfaz) en español, interfaz de programación de aplicaciones.

HTTP - Hypertext Transfer Protocol (o Protocolo de Transferencia de Hipertexto en español)

GDD - Game desing document (o documento de diseño de juego en español)

ODS – Objetivos de desarrollo sostenible

TRPG – Tactical role-palying game (o juego de rol estratégico en español)

RPG - role-palying game (o juego de rol en español)

UML - Unified Modeling Language (o lenguaje unificado de modelado en español)

DTO – Data Transfer Object (u Objeto de transferencia de datos en español)