Universitat Oberta de Catalunya

Autor: Victor Manuel Quesada Cobos

Tutor: Jordi Duch Gavaldà.

Profesor: Joan Arnedo Moreno

Máster universitario de Diseño y Programación de Videojuegos

Diseño de experiencias de juego

18/06/2023

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Kingdoms of Lunite*
Nombre del autor:	*Víctor Manuel Quesada Cobos*
Nombre del colaborador/a docente:	*Jordi Duch Gavaldà*
Nombre del PRA:	*Joan Arnedo Moreno*
Fecha de entrega:	*06/2023*
Titulación o programa:	*Máster universitario de Diseño y Programación de Videojuegos*
Área del Trabajo Final:	*Diseño de experiencias de juego*
Idioma del trabajo:	*Español*
Palabras clave	*MMO, Gestión de recursos, Combates tácticos por turnos RPG.*

Resumen del Trabajo

Este proyecto tiene como finalidad el desarrollo de un videojuego de estrategia multijugador masivo online en tiempo real con elementos de gestión de recursos y rol. El objetivo del jugador será desarrollar y gestionar su imperio, mientras compite por recursos y se enfrenta a otros jugadores en batallas tácticas por turnos.

Las mecánicas principales del juego incluyen la gestión de la aldea, donde se podrán crear edificios e investigar nuevas tecnologías, así como el combate táctico por turnos en el que los jugadores controlan a un equipo de personajes y deben utilizar una variedad de habilidades y tácticas para vencer a sus rivales.

Para el desarrollo del videojuego, se ha utilizado la herramienta Unity para la creación del cliente y se ha implementado un servidor web para gestionar el estado de las cuentas de los jugadores. La metodología de desarrollo empleada ha sido Agile, con iteraciones para ir construyendo el juego de forma incremental y evolutiva.

Los resultados del proyecto incluyen la creación de una versión funcional del juego, con una interfaz de usuario intuitiva y un sistema de combate táctico bien equilibrado. También se ha logrado integrar eficientemente los elementos de gestión de recursos y combate táctico por turnos, proporcionando una experiencia de juego equilibrada y satisfactoria para el jugador.

Esta memoria incluye la fase de planificación, diseño y desarrollo de todo el proyecto, así como la historia y descripción del videojuego.

Abstract

This project aims to develop a real-time multiplayer online strategy game with resource management and role-playing elements. The player's objective is to develop and manage their empire while competing for resources and engaging in turn-based tactical battles against other players.

The core mechanics of the game include village management, where players can construct buildings and research new technologies, as well as turn-based tactical combat where players control a team of characters and must utilize a variety of skills and tactics to defeat their opponents.

Unity has been used as the development tool for creating the client, and a web server has been implemented to manage player account states. The Agile development methodology has been employed, with iterative processes to build the game incrementally and iteratively.

The project's outcomes include the creation of a functional version of the game, featuring an intuitive user interface and a well-balanced turn-based combat system. Efficient integration of resource management and turn-based tactical combat elements has been achieved, providing a balanced and satisfying gaming experience for the player.

This report covers the planning, design, and development phases of the entire project, as well as the story and description of the video game.

Índice

1. Introducción 10
1.1. Introducción/Prefacio 10
1.1.1. Motivación 10
1.2. Descripción/Definición 11
1.3. Objetivos generales 12
1.3.1. Objetivos principales 12
1.3.2. Objetivos secundarios 13
1.4. Metodología y proceso de trabajo 14
1.5. Planificación 14
1.5.1. Riesgos 17
1.5.2. Presupuesto 17
2. Análisis de mercado 19
2.1. Público objetivo y perfiles de usuario 21
2.2. Competencia 22
2.3. Análisis DAFO 25
3. Propuesta 25
3.1. Definición de objetivos/especificaciones del producto 26
3.2. Conceptualización 27
3.2.1. Universo 27
3.2.2. Aldeas 28
3.2.3. Recursos 28
3.2.4. Edificios 30
3.2.5. Investigaciones 33
3.2.6. Unidades 37
3.2.7. Héroes 39
3.2.8. Misiones 39
3.2.9. Combate 41
3.3. Modelo de negocio 42
3.4. Estrategia de marketing 42

4. Diseño técnico 43
4.1. Entorno 44
4.2. Requisitos técnicos entorno desarrollo 45
4.3. Inventario herramientas usadas 45
4.3.1. Unity Editor 2021.3.20f1 LTS 45
4.3.2. Unity Multipurpose Avatar 2 46
4.3.3. Visual Studio 2022 46
4.3.4. Notepad++ 46
4.3.5. SQL Server Management Studio 46
4.3.6. Mixamo.com 46
4.3.7. GitHub 47
4.3.8. Sourcetree 47
4.3.9. DALL·E 47
4.4. Inventario assets usados 47
4.4.1. Assets UI 47
4.4.2. Assets VFX 48
4.4.3. Assets SFX 49
4.4.4. Assets 3D 49
4.4.5. Plugins 49
4.5. Arquitectura general del sistema 50
4.5.1. Arquitectura del cliente 50
4.5.2. Arquitectura del servidor 52
4.5.3. Comunicación cliente-servidor 53
4.5.4. Bases de datos 55
4.5.5. Seguridad 58
4.6. Diseño de niveles 59
5. Implementación 63
5.1. Implementación técnica 63
5.1.1. Control del tiempo 63
5.1.2. Recursos 63
5.1.3. Construcción de elementos 64
5.1.4. Misiones 65

5.1.5. Recursos de texto 66
5.1.6. Interfaz usuario Aldea 67
5.1.7. Creación de personajes 68
5.1.8. Combate 69
5.1.9. Sonido 70
5.2. Requisitos de instalación 71
5.2.1. Requisitos de Software 71
5.2.2. Requisitos de Hardware 72
5.2.3. Formación/Conocimientos 72
5.3. Instrucciones de instalación 73
5.3.1. Base de datos 73
5.3.2. Servidor web 73
6. Demostración 74
6.1. Prototipos 74
6.1.1. Prototipos Lo-Fi 74
6.1.2. Prototipos Hi-Fi 75
6.2. Tests 76
6.3. Ejemplos de uso del producto 77
6.3.1. Empezar a jugar 77
6.3.2. Mejora de elemento 79
7. Conclusiones y líneas de futuro 81
7.1. Conclusiones 81
7.1.1. Lecciones aprendidas 81
7.1.2. Reflexión objetivos conseguidos 82
7.1.3. Análisis seguimiento de la planificación 83
7.2. Líneas de futuro 83
Bibliografía 85
Anexos 88

Índice de figuras

Figura 1: imagen pantalla de carga inicial 10
Figura 2: Planificación diseño 14
Figura 3: Planificación plan de proyecto 15
Figura 4: Planificación primera versión 15
Figura 5: Planificación versión jugable 16
Figura 6: Planificación producto final 16
Figura 7: Recurso Hierro 29
Figura 8: Recurso Madera 29
Figura 9: Recurso Lunite 29
Figura 10: Recurso Comida 29
Figura 11: Mina de Hiero 29
Figura 12: Almacén de Hierro 29
Figura 13: Bóveda de Hierro 29
Figura 14: Aserradero 29
Figura 15: Almacén de Madera 29
Figura 16: Bóveda de Madera 29
Figura 17: Mina de Lunita 30
Figura 18: Almacén de Lunita 30
Figura 19: Bóveda de Lunita 30
Figura 20: Granja 30
Figura 21: Cuartel 31
Figura 22: Universidad 31
Figura 23: Academia 31
Figura 24: Hospital 31
Figura 25: Taller 32
Figura 26: Taller de Constructores 32
Figura 27: Taller de Ingenieros 32
Figura 28: Oasificadora 32
Figura 29: Matemáticas 33
Figura 30: Metalurgia 33
Figura 31: Herrería 33
Figura 32: Mecánica 33
Figura 33: Hidrología 34
Figura 34: Exploración 34
Figura 35: Logística 34
Figura 36: Oasificación 34
Figura 37: Piscicultura 35
Figura 38: Red Académica 35
Figura 39: Sanación 35
Figura 40: Artes Marciales 35
Figura 41: Artes Arcanas 36

Figura 42: Fortaleza 36
Figura 43: Alquimia 36
Figura 44: Brujería 36
Figura 45: Conjuración 37
Figura 46: Destrucción 37
Figura 47: Restauración 37
Figura 48: Alteración 37
Figura 49: Carro de carga pequeño 38
Figura 50: Carro de Carga Grande 38
Figura 51: Trampa para Peces 38
Figura 52: Especto Espía 38
Figura 53: estructura solución servidor 52
Figura 54: swagger 53
Figura 55: Comunicación cliente-servidor 54
Figura 56: esquema identity 55
Figura 57: esquema datos de juego 56
Figura 58: esquema jwt 58
Figura 59: vista área mapa completo 60
Figura 60: puntos aparición héroes 60
Figura 61: mapa con mesh obstáculos 61
Figura 62: ayudas visuales con elementos UI del tipo espaciales 61
Figura 63: vista inicial desnivel desde puntos de aparición 62
Figura 64: canvas aldea 67
Figura 65: modelo datos habilidades 69
Figura 66: audiomixer 71
Figura 67: opciones publicación servidor 74
Figura 68: prototipo lo-fi vista aldea 75
Figura 69: prototipo lo-fi vista combate 75
Figura 70: prototipo hi-fi vista aldea 76
Figura 71: prototipo hi-fi vista combate 76
Figura 72: panel registro 77
Figura 73: visor recursos 78
Figura 74: selector aldea 78
Figura 75: menú principal 79
Figura 76: menú opciones 79
Figura 77: panel universidad 80
Figura 78: panel requisitos 80
Figura 79: cola construcción unidades 80

Índice de tablas

Tabla 1: tabla de costes 18
Tabla 2: competidores 24
Tabla 3: análisis DAFO 25

1. Introducción

1.1. Introducción/Prefacio

El trabajo propuesto es un juego de fantasía medieval que combina elementos de gestión de recursos, estrategia en tiempo real y combates tácticos por turnos con RPG. Los jugadores una vez crean su cuenta en el juego, toman el control de una pequeña aldea y deberán ir evolucionando las distintas características que el juego les presenta y combatiendo con los otros jugadores para obtener recursos.

Figura 1: imagen pantalla de carga inicial

Los combates serán por turnos tácticos de máximo 3 personajes por jugador en los que si el atacante gana consigue robar recursos. A medida que los personajes van compitiendo irán ganando experiencia e irán subiendo de nivel y mejorando sus atributos.
Los combates tendrán lugar en escenarios en 3D donde los jugadores deberán tener en cuenta el terreno y la posición de sus personajes al planear sus estrategias de combate.

1.1.1. Motivación

La motivación detrás de la creación de este juego es personal. Como aficionado a los juegos de gestión MMO y a los combates por turnos con roles de personajes, siempre he sentido que faltaba algo en los juegos de gestión. Si bien la gestión de recursos y la estrategia son fundamentales, las batallas a menudo se reducen a simples combates automáticos en los que el jugador tiene poco control.

Por lo tanto, mi objetivo con este proyecto es combinar lo mejor de ambos mundos y crear un juego que me gustaría jugar: uno en el que la gestión de recursos sea importante y

estratégica, pero también con batallas tácticas por turnos en las que el jugador tenga un control más directo sobre el resultado.
Se pretende realizar un juego que permita al jugador vivir una experiencia completa, desde la gestión de recursos y la evolución de su aldea hasta el combate táctico y la obtención de recompensas.
Esta combinación puede ser una adición interesante al género y puede ofrecer una experiencia única para los jugadores que buscan un desafío tanto en la gestión de recursos como en los combates tácticos.

1.2. Descripción/Definición

El punto de partida del trabajo de creación de Kingdoms of Lunite es el proceso de diseño de un videojuego de estrategia multijugador masivo en línea que combina mecánicas de gestión de recursos y combate táctico RPG. Este punto de partida implica la definición de los elementos esenciales del juego, tales como su ambientación, mecánicas de juego, objetivos y retos, así como la creación de un prototipo que permita evaluar su viabilidad y funcionamiento.

En este proceso de diseño se deben considerar tanto los aspectos técnicos como los creativos y artísticos, tales como la elección de la plataforma y tecnologías adecuadas para el juego, la creación de personajes, edificios y paisajes que formen parte de la ambientación, y la elaboración de una historia coherente y atractiva que permita a los usuarios sumergirse en el mundo de Kingdoms of Lunite.

Además, el punto de partida también implica la investigación de la competencia y el análisis del mercado existente para identificar oportunidades y desafíos en el lanzamiento y promoción del juego. De esta forma, se podrá diseñar una estrategia de marketing y comercialización efectiva y adaptada al público objetivo del juego.

Kingdoms of Lunite aborda la necesidad de un juego de estrategia en tiempo real con un enfoque en la gestión de recursos y los combates tácticos RPG en una ambientación de fantasía medieval. Este tipo de juego es popular en el mercado de los videojuegos y tiene una gran base de fans. Por lo tanto, es un tema relevante porque hay una demanda para este tipo de juego y los usuarios buscan experiencias nuevas y emocionantes.

En la actualidad, hay muchos juegos de estrategia en tiempo real en el mercado, como Ogame (Gameforge, 2002), Rise of Kingdoms (Lilith Games, 2018), Clash of Clans (Supercell, 2012), entre otros, pero no todos tienen un enfoque en la gestión de recursos y los combates tácticos RPG en una ambientación de fantasía medieval. Por lo tanto, Kingdoms of Lunite resuelve este problema al ofrecer una experiencia única e innovadora combinando varias mecánicas principales.

Las características principales de Kingdoms of Lunite incluyen:

- Ambientación de fantasía medieval.
- Mecánicas de juego de gestión de recursos y combates tácticos RPG
- Personalización de personajes
- Multijugador online
- Interfaz de usuario intuitiva y fácil de usar.

1.3. Objetivos generales

Se procede a crear una lista de objetivos para cada una de las siguientes áreas: vista imperio y vista combate.

1.3.1. Objetivos principales

Lista objetivos Escena de combate:

- Crear mapa único.
- Movimiento de los personajes libre dentro del rango del personaje.
- Cambiar de personaje controlado durante la partida.
- Realizar ataques y defensas.
- Realizar habilidad que intercambia vida por puntos de magia.
- Controlar estado del combate.
- Minimapa donde aparecen la información de los personajes del jugador y las del enemigo una vez que los ha descubierto.
- Adaptar animaciones mixamo (Adobe, 2008) a avatares UMA (UMA Steering Group, 2009).
- Animación al atacar y recibir daño.
- Sistema de partículas y efectos para las habilidades y al recibir daño.
- Habilidades genéricas y especiales por razas, con animaciones propias.

- Posibilidad de ver los movimientos realizados por el rival en el turno anterior cuando el jugador inicia turno.

Lista objetivos escena de imperio:

- Vista registro usuario.
- Vista acceso al juego.
- Vista general donde visualizar el estado actual de la aldea, todo lo que se está construyendo/mejorando y el resumen de la información de la aldea.
- Vista de recursos donde gestionar los edificios que producen éstos.
- Creación de personajes.
- Vista edificios donde mejorar los edificios.
- Vista universidad donde poder seleccionar las tecnologías a mejorar y donde poder visualizar árbol de dependencias.
- Vista continente donde explorar el mundo y ver las aldeas de los rivales.
- Vista misiones donde se crean las misiones a otras aldeas.
- Vista personajes donde gestionar el aspecto visual y la evolución de los personajes.
- Vista ranking.
- Recibir actualizaciones tanto de misiones como eventos del sistema sin tener que cambiar de sección o refrescar.

1.3.2. Objetivos secundarios

Lista objetivos secundarios escena combate:

- Recibir movimientos del rival sin tener que salir del combate y volver a cargar.
- Movimiento de los personajes haciendo clic donde se quiere desplazar dentro del rango del personaje.
- Historial del combate.
- Limitación de tiempo para finalizar turno.
- Pantalla de finalización con resultados.
- Añadir mapas distintos.

Lista objetivos secundarios escena de imperio:

- Vista academia donde gestionar las habilidades de los personajes.

- Vista hospital donde ver los personajes que se encuentran recuperando salud.
- Sistema de mensajes (resultado espionaje).
- Vista taller donde crear unidades de transporte.
- Creación de escena donde aparecen los edificios que el jugador ha construido (vista RTS).
- Edificios seleccionables que funcionan como acceso directo.
- Edificios en el mapa reaccionan a su estado

1.4. Metodología y proceso de trabajo

El propósito de las listas de objetivos es servir como backlog priorizado de las tareas necesarias para realizar el juego propuesto de manera completa. La priorización se basa en la idea de identificar lo mínimo necesario para llegar a tener un MVP. El objetivo de la metodología elegida, Agile, es tener un producto ejecutable, aunque no tenga todavía todas las características ni la misma forma que el producto final, durante cada iteración.

1.5. Planificación

Es imprescindible llevar a cabo un proceso de diseño sólido, que abarque desde la conceptualización del juego hasta la creación de un documento de diseño, antes de proceder a la implementación de la lista de objetivos priorizadas descritas a continuación con el fin de garantizar la coherencia y cohesión del resultado final.

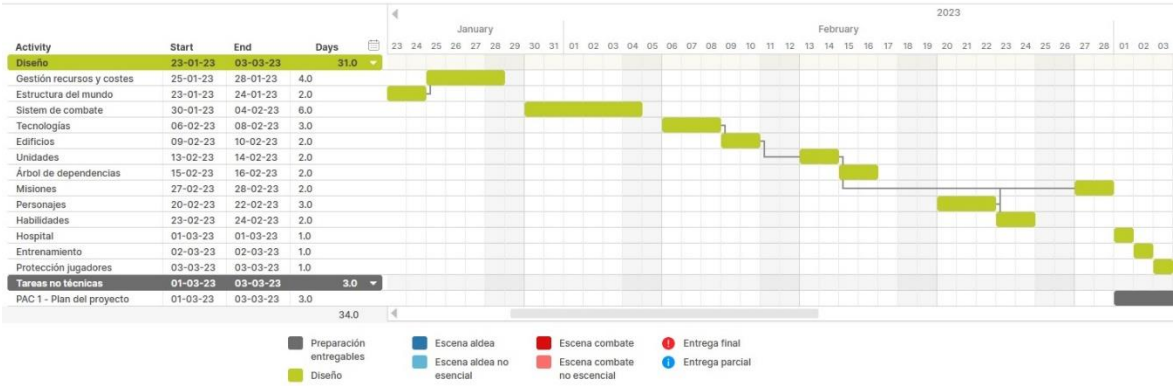

Figura 2: Planificación diseño

En primer lugar, la gestión de recursos y costes es crucial para garantizar el equilibrio del juego. Se debe permitir que los jugadores tomen decisiones estratégicas informadas sobre cómo asignar sus recursos limitados. Con el objetivo de proporcionar un primer marco para la jugabilidad donde los jugadores puedan construir y desarrollar sus propias colonias y territorios, se define la estructura del mundo y el sistema de aldeas.

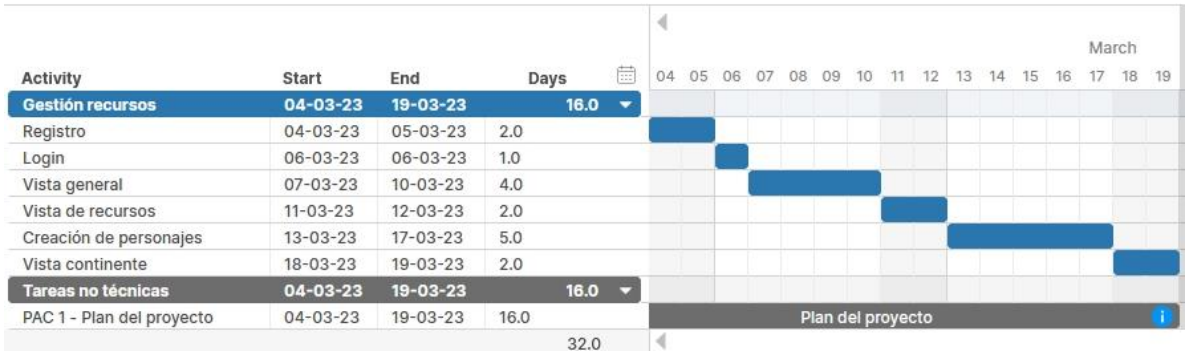

Figura 3: Planificación plan de proyecto

Posteriormente a estos aspectos claves del juego respecto a las mecánicas de gestión de recursos y multijugador, el sistema de combate, tipos de habilidades y distancias son esenciales para la otra parte del núcleo del juego, ya que proporcionan la mecánica central para las interacciones entre jugadores.

A continuación, los siguientes puntos se basan tanto en la economía como en el combate y son una evolución de los sistemas previamente diseñados en las fases anteriores, como lo son el tipo de edificios e investigaciones y las funcionalidades que aportan al jugador, el árbol de dependencias y las misiones.

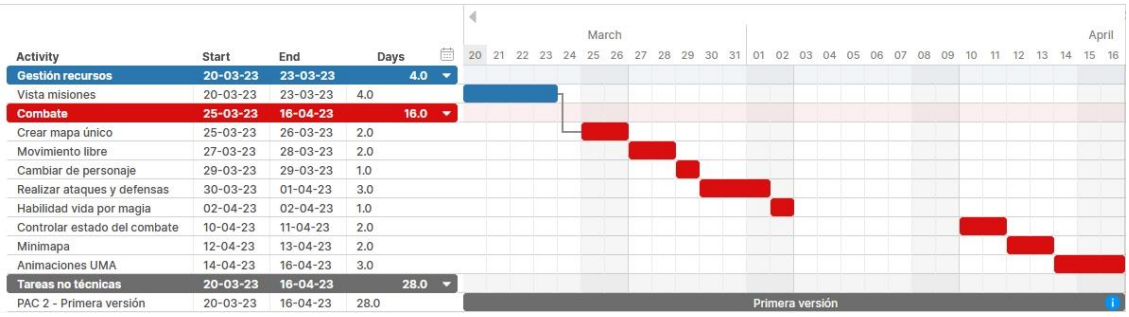

Figura 4: Planificación primera versión

Finalmente, el resto de los puntos, son importantes, pero no son esenciales para el núcleo del juego y pueden ser diseñados en una etapa posterior.

Teniendo en cuenta el objetivo de tener un MVP jugable, se debe priorizar la implementación de las funcionalidades que permitan al usuario registrar y acceder a su cuenta, así como gestionar la aldea y sus recursos, crear personajes, ya que son aspectos fundamentales en cualquier juego de estrategia.

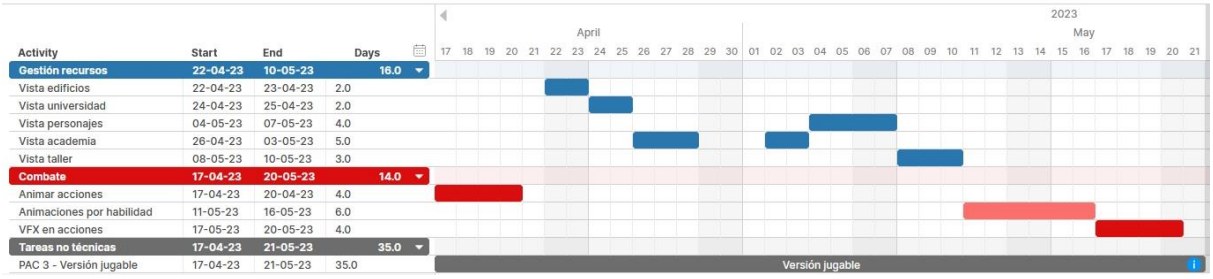

Figura 5: Planificación versión jugable

A continuación, la vista universidad y la exploración del continente son aspectos importantes para el desarrollo a largo plazo del juego y, por lo tanto, deben ser incluidos en el MVP. Las demás características, como la escena adaptada al avance del jugador, la vista de personajes, academia, hospital y taller pueden ser importantes para el desarrollo del juego, pero no son esenciales en una versión inicial.

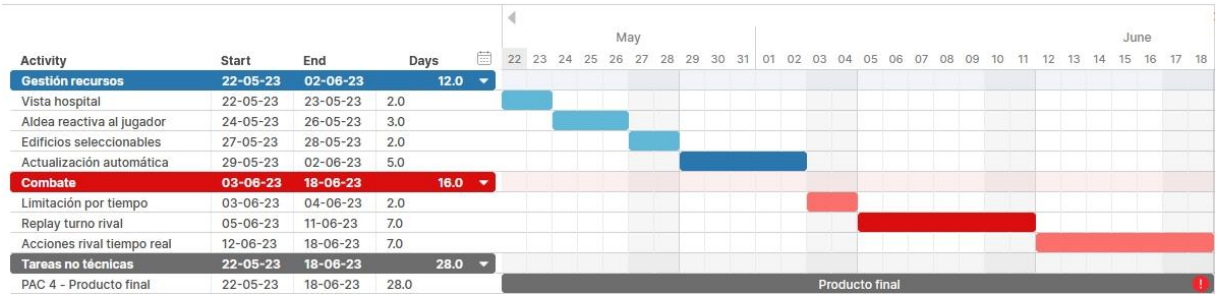

Figura 6: Planificación producto final

También se puede ver en el diagrama que la planificación propuesta se adapta al marco temporal de los hitos del TFM y tiene en cuenta la disponibilidad del estudiante y fechas especiales como festivos.

1.5.1. Riesgos

Cabe destacar que el proyecto propuesto se sale del marco estrictamente visto en los planes de estudio del master, ya que para poder realizar las mecánicas de estrategia en tiempo real multijugador masivo en línea es necesario el desarrollo de la parte servidor web para poder gestionar y respaldar el estado del universo y del progreso de todos los jugadores. Las distintas funcionalidades que se listan en la parte de las escenas de combate y la escena de imperio tienen su equivalente desarrollo en la parte servidor para controlar la partida del jugador y sincronizar sus acciones con el resto del universo.
Teniendo esto en cuenta, la planificación se ha realizado priorizando las funcionalidades para tener una primera versión y posteriormente un producto final para el proyecto en el que hay partes que son no esenciales.

El desarrollo de cualquier proyecto puede estar sujeto a imprevistos que puedan afectar el cumplimiento de los objetivos planificados. Para minimizar estos riesgos y garantizar el éxito del proyecto, se ha elaborado un plan de contingencia que tiene en cuenta la priorización de los objetivos principales y secundarios.

En primer lugar, se ha establecido que objetivos son prioritarios en las escenas de combate y de imperio para el proyecto. Por lo tanto, cualquier retraso o imprevisto que se presente en la implementación de estos objetivos debe ser tratado con la mayor rapidez posible para no afectar el cronograma general del proyecto.

Para los objetivos secundarios, se ha establecido que, si bien son importantes para la experiencia de juego y pueden mejorarla, su retraso o modificación no afectaría significativamente el cumplimiento de los objetivos principales. Por lo tanto, cualquier imprevisto en la implementación de estos objetivos puede ser abordado con un enfoque más flexible y se pueden hacer ajustes en el cronograma.

1.5.2. Presupuesto

El presupuesto es una parte fundamental de cualquier proyecto, ya que permite establecer los recursos necesarios y el coste estimado de cada tarea. En este proyecto, se ha elaborado un presupuesto en base a las horas dedicadas y a las diferentes partidas presupuestarias que lo componen.

Respecto al presupuesto en equipo humano, se ha realizado estimando el precio de 1 desarrollador. En cuanto al equipamiento técnico, se requerirá una serie de equipos informáticos de alta gama y un servidor para la realización de pruebas y para la puesta en marcha del proyecto.

Por otro lado, también se han tenido en cuenta otros recursos necesarios para el proyecto, como por ejemplo software específico, licencias de uso y cualquier otro recurso necesario para el desarrollo del proyecto.
Sobre el coste estimado para el desarrollo completo del proyecto, se ha realizado una estimación teniendo en cuenta el trabajo realizado hasta la fecha y los costes necesarios para la finalización de este.

		A		
		Hora	Precio/Hora	Coste
Partidas presupuestarias	Diseño	120	15€	1800€
	Análisis	80	25€	1000€
	Programación unity	200	15€	3000€
	Programación servidor	120	20€	2400€
	Hardware desarrollo	-	-	1500€
	Servidor	3*	60*	180€
	Asset: Terrain	-	-	285€
	Asset: UI	-	-	150€
	Asset: contenido UMA	-	-	180€
	Asset: edificios 3d	-	-	55€
	Asset: vfx	-	-	50€
	TOTAL	-	-	10.600€

Tabla 1: tabla de costes

El precio del servidor es mensual, durante la fase de desarrollo.

Las horas necesarias para cada partida son extraídas de la planificación realizada.
Los precios por hora son basados en el precio medio de los trabajadores en España durante el año 2023 según la web talen.com.

2. Análisis de mercado

Los juegos de estrategia en línea han sido una parte fundamental del mundo de los videojuegos desde hace muchos años. Estos juegos se caracterizan por su énfasis en la toma de decisiones y la planificación, en lugar de la acción inmediata y la rapidez de reflejos. Los primeros juegos de estrategia en línea se originaron en la década de 1970 con títulos MUD. Estos juegos se basaban en la interacción entre jugadores y tenían unas bases textuales.

Con el avance de la tecnología y la popularización de Internet, los juegos de estrategia en línea se han convertido en un género muy popular. A mediados de los años 90 surgieron juegos como Warcraft (Blizzard, 2004), Command & Conquer (Westwood, 1995) y Age of Empires (Ensemble, 1997), los cuales popularizaron el género de estrategia en tiempo real. A partir de entonces, han surgido muchos otros juegos en línea de este género, cada uno con su propia mecánica y características únicas.

En los últimos años, los juegos multijugador-masivos en línea han experimentado un gran crecimiento gracias a la facilidad de acceso a internet y el aumento en la potencia de procesamiento de los dispositivos móviles. Los juegos de estrategia no son una excepción, y actualmente existe una gran cantidad de juegos multijugador-masivos en línea de estrategia para una amplia gama de plataformas, desde móviles hasta consolas de sobremesa.

En la actualidad, existen muchos juegos de estrategia disponibles en el mercado, algunos de los cuales son muy populares y han sido jugados por millones de personas en todo el mundo. Muchos de estos juegos se basan en la construcción y administración de ciudades, así como en la formación de ejércitos y la estrategia para ganar batallas.

En este contexto, hay algunos juegos muy destacados, como Starcraft (Blizzard, 1998) o EVE online (CCP, 2003). Estos juegos ofrecen diferentes enfoques en cuanto a mecánicas de juego y elementos temáticos, desde la era de la antigüedad hasta el futuro lejano y el espacio exterior.

Concretamente, en el mundo de los juegos multijugador masivo online de estrategia de navegador, existen varios títulos populares, cada uno con su propia mecánica y estilo de juego. Uno de los juegos más famosos es Travian (Travian Games, 2004), un juego de estrategia donde los jugadores compiten por recursos y territorio en un mundo virtual. Ogame es otro juego popular de navegador que se centra en la construcción de imperios espaciales, donde los jugadores pueden construir y mejorar naves espaciales para conquistar nuevos planetas y recursos. Con el auge de la tecnología web, estos juegos están disponibles para cualquier persona con acceso a Internet y un navegador web, lo que los hace accesibles a una gran audiencia de jugadores en todo el mundo.

En los últimos años, además del auge de las tecnologías web, los juegos móviles han cobrado una gran importancia en el mercado de los juegos de estrategia online. Los dispositivos móviles ofrecen una gran comodidad y accesibilidad para jugar en cualquier lugar y en cualquier momento. En esta plataforma se puede encontrar títulos como los mencionados Clash of Clans y Rise of Kingdoms. Estos juegos han adoptado el modelo free-to-play, donde los jugadores pueden descargar el juego de forma gratuita, pero luego se les ofrece la posibilidad de comprar elementos virtuales dentro del juego.

Las mecánicas de juego son un aspecto fundamental en cualquier juego, especialmente en los juegos multijugador-masivos en línea de estrategia. Al analizar las mecánicas de juego, se pueden identificar las principales características que hacen que los juegos sean interesantes y atractivos para los jugadores.

Algunas de las mecánicas más comunes en los juegos de estrategia multijugador en línea incluyen la construcción de bases, la gestión de recursos, la creación y el entrenamiento de unidades, y la investigación de tecnologías. Estas mecánicas son implementadas de diferentes maneras logrando crear experiencias únicas y atractivas para los jugadores.

Además de las mecánicas mencionadas anteriormente, los juegos multijugador-masivos en línea de estrategia también pueden incluir mecánicas sociales, como la formación de alianzas y el trabajo en equipo para lograr objetivos comunes. Estas mecánicas son fundamentales para fomentar la interacción y la cooperación entre los jugadores y mejorar la experiencia de usuario.

La experiencia de usuario es un factor clave en el género. Para que los jugadores se sientan atraídos y comprometidos con el juego, los creadores tienen en cuenta varios factores que influyen en la experiencia, como la interfaz de usuario, la jugabilidad, la personalización y la comunidad.
Kingdoms of Lunite tiene una propuesta única que lo diferencia de otros juegos multijugador masivo en línea. En primer lugar, el enfoque en la gestión de recursos es importante y estratégico, lo que permite a los jugadores tener una experiencia más completa e inmersiva. Además, los combates tácticos por turnos en los que el jugador tiene un mayor control sobre el resultado añaden un nivel de complejidad y estrategia al juego.

La combinación de estos enfoques, gestión de recursos y combates tácticos RPG, lo hace ser distinto sobre juegos de similar temática. Además, el juego ofrece una experiencia completa que permite al jugador crear su aldea, evolucionar sus características, personalizar a sus personajes y combatir con otros jugadores en escenarios en 3D.

En comparación con otros juegos de estrategia en tiempo real en el mercado, Kingdoms of Lunite resuelve la falta de un juego que combine gestión de recursos y combates tácticos RPG en una ambientación de fantasía medieval. Juegos como Ogame, Rise of Kingdoms y Clash of Clans, aunque exitosos, no ofrecen esta combinación única de mecánicas principales. Esta diferencia hace que Kingdoms of Lunite sea atractivo para aquellos que buscan nuevas y emocionantes experiencias en el género de la estrategia MMO.

2.1. Público objetivo y perfiles de usuario

Para realizar un análisis del público objetivo de un juego, es necesario tener en cuenta diversos factores como la edad, género, intereses y hábitos de juego de los potenciales usuarios. En el caso de un juego de estrategia multijugador masivo en línea, el público objetivo puede variar en función de la temática y el estilo del juego.

En general, los jugadores de este tipo de juegos suelen ser jóvenes o adultos jóvenes, con edades comprendidas entre los 18 y los 35 años (Yee, Nick, 2004). También es común que sean jugadores experimentados, que han jugado otros juegos de estrategia en el pasado y buscan un nuevo desafío. En cuanto al género, el público objetivo suele estar compuesto por hombres y mujeres por igual, aunque en algunos casos puede haber una ligera predominancia masculina.

En cuanto a los intereses y hábitos de juego, el público objetivo de un juego de estrategia multijugador masivo en línea puede tener una pasión por los juegos de estrategia y la planificación a largo plazo.

En términos generales, el público objetivo al que se dirige el juego son personas interesadas en los juegos de estrategia en línea y con experiencia previa en juegos similares. El juego está dirigido a usuarios de cualquier género, mayores de 18 años, con acceso a un ordenador de escritorio y una conexión a internet estable.

En cuanto a la descripción de perfiles tipo de usuarios, se pueden considerar los siguientes:

- Jugadores experimentados de juegos de estrategia en línea: Este perfil de usuario tiene experiencia previa en juegos similares y busca nuevos desafíos y experiencias en el género. Suelen dedicar varias horas al día al juego y valoran la complejidad y profundidad de las mecánicas de juego.
- Jugadores ocasionales de juegos de estrategia en línea: Este perfil de usuario tiene interés en los juegos de estrategia en línea, pero no tiene tanta experiencia previa. Buscan juegos que les permitan jugar en su tiempo libre y que no requieran un compromiso excesivo. Suelen valorar la simplicidad y facilidad de uso de los juegos.
- Jugadores interesados en la temática del juego: Este perfil de usuario se siente atraído por la temática de fantasía medieval del juego y busca una experiencia de inmersión en el universo del juego. Pueden tener experiencia previa en otros juegos, pero su principal interés es la ambientación y la historia del juego.

2.2. Competencia

En el mercado de los juegos de estrategia en línea, existe una gran variedad de opciones entre las cuales los usuarios pueden elegir. Desde los clásicos juegos de construcción y combate, hasta los más recientes juegos de rol y gestión de recursos. En este contexto, es fundamental conocer en profundidad a los competidores para poder adaptarse y ofrecer una experiencia de juego única y atractiva para los usuarios.

En este análisis, se comparará el juego desarrollado con algunos de los competidores más populares en el mercado para distintos tipos de plataforma: Ogame, Clash of Clans, Tribal Wars 2 (InnoGames, 2012), X-COM (Mythos Games, 1993), Mario + Rabbids (Nintendo, 2017) y Solasta (Tactical Adventures, 2020).

- Tribal Wars 2: es un juego de navegador de estrategia en tiempo real ambientado en un mundo medieval. Los jugadores construyen su aldea, entrenan a su ejército, y

compiten contra otros jugadores. Tiene una amplia comunidad de jugadores, gráficos atractivos, y es gratuito. Ofrece una alta dificultad para los nuevos jugadores y el juego puede ser lento a veces.

- Ogame: es un juego de navegador de estrategia en tiempo real ambientado en el espacio. Los jugadores construyen y gestionan su imperio y compiten contra otros jugadores. Ofrece una alta complejidad y dificultad, lo que lo hace más adecuado para jugadores experimentados.
- Clash of Clans: es un juego de estrategia en tiempo real para dispositivos móviles en el que los jugadores construyen y mejoran su aldea, entrenan a su ejército y luchan contra otros jugadores. El juego tiene una gran comunidad de jugadores y una mecánica de juego adictiva, lo que lo ha convertido en uno de los juegos móviles más populares.
- XCOM: es una saga de juegos multiplataforma de estrategia por turnos en los que los jugadores lideran una organización militar que se encarga de defender la Tierra de una invasión alienígena. El juego presenta una fuerte mecánica de gestión de recursos y personalización de personajes. Los gráficos y el diseño son de alta calidad y la dificultad puede ser ajustada según las preferencias del jugador. Es una opción atractiva para los amantes de la ciencia ficción y la estrategia.
- Mario + Rabbids: es un juego de estrategia por turnos para Nintendo Switch. Los jugadores controlan a distintos personajes con distintas características. El juego cuenta con un sistema de personalización de personajes, con habilidades y armas únicas para cada uno. La ambientación es colorida y atractiva, y los gráficos son de alta calidad. Es una buena opción para aquellos que buscan un juego de estrategia más ligero y divertido.
- Solasta: es un juego de rol táctico para PC ambientado en un mundo de fantasía. Los jugadores lideran un grupo de cuatro personajes, cada uno con habilidades únicas, a través de una serie de misiones y batallas tácticas. El juego presenta un sistema de personalización de personajes detallado y un sistema de creación de misiones que permite a los jugadores crear sus propias aventuras. Los gráficos son de alta calidad y la ambientación es inmersiva. Es una opción atractiva para los amantes de los juegos de rol y la estrategia táctica.

.

		Plataforma	Tipo de juego	Ambientación	Mecánicas principales	Diseño visual	Público objetivo	Modelo de negocio
Títulos	Kingdoms of Lunite	Pc	MMO Estrategia	Fantasía medieval	Combate táctico RPG, Gestión de recursos	Gráficos 3D realistas	Adultos jóvenes interesados en estrategia, RPG y fantasía medieval	Free-to-play con compras integradas
	Ogame	Web	MMO Estrategia	Ciencia ficción	Gestión de recursos, exploración y combate	Gráficos 2D	Adultos jóvenes interesados en estrategia y ciencia ficción	Free-to-play con compras integradas
	Clash of Clans	Móvil	MMO Estrategia	Fantasía medieval	Combate en tiempo real y gestión de recursos	Gráficos 2.5D estilizados	Adultos jóvenes interesados en estrategia y fantasía medieval	Free-to-play con compras integradas
	Tribal Wars 2	Web	MMO Estrategia	Medieval	Gestión de recursos, combate y construcción de aldeas	Gráficos 2.5D realistas	Adultos jóvenes interesados en estrategia e historia medieval	Free-to-play con compras integradas
	X-COM	Multiplataforma	Estrategia táctica	Ciencia ficción	Gestión de recursos, combate táctico RPG	Gráficos 3D realistas	Adultos jóvenes interesados en estrategia y juegos de rol	Pago único
	Mario + Rabbids	Nintendo Switch	Estrategia táctica	Fantasía	Combate táctico RPG	Gráficos 3D estilizados	Jugadores de todas las edades	Pago único
	Solasta	Pc	Estrategia táctica	Fantasía medieval	Combate táctico RPG	Gráficos 3D realistas	Adultos jóvenes interesados en juegos de rol	Pago único

Tabla 2: competidores

En la anterior tabla comparativa se han analizado las fortalezas y debilidades de cada juego en diferentes aspectos, como plataforma, tipo de juego, ambientación, mecánicas principales, entre otros.

2.3. Análisis DAFO

Análisis interno	Análisis externo
Debilidades • Mecánicas con menor complejidad. • Falta de capital en comparación a proyectos desarrollados por grandes empresas. • Modelo de negocio incierto.	**Amenazas** • Competencia fuerte y consolidada. • Dificultad de atraer jugadores que prefieran mecánicas más complejas. • Dificultad de darse a conocer. • Coste de servidor.
Fortalezas • Combates tácticos en escenario 3D. • Combinación innovadora de mecánicas de gestión de recursos MMO con combates por turnos y estrategia.	**Oportunidades** • Mercado de juegos en crecimiento. • Posibilidad de atraer jugadores interesados en juegos de gestión de recursos y en juegos de combates por turnos.

Tabla 3: análisis DAFO

3. Propuesta

Tras el análisis exhaustivo de la competencia en el mercado de los juegos de estrategia en tiempo real, se ha identificado que la mayoría de los productos existentes se centran en batallas épicas y/o campañas militares. Sin embargo, la propuesta de Kingdoms of Lunite, se diferencia de la competencia al ofrecer una experiencia de juego única que combina la gestión de recursos y los combates tácticos por turnos.

3.1. Definición de objetivos/especificaciones del producto

Kingdoms of Lunite se desarrolla en un mundo de fantasía medieval llamado Lunaria. Los jugadores asumen el papel de líderes de aldeas que compiten por recursos y poder. Deben construir y gestionar su aldea para que crezca y se convierta en un gran imperio.

El juego tiene como objetivo principal la construcción y gestión del reino, para lo cual los jugadores deben recolectar recursos, construir estructuras y mejorar la economía de su reino. La investigación y la tecnología son clave para la supervivencia y la prosperidad del reino, y los jugadores pueden investigar nuevas formas de construir y producir recursos.

Además, los jugadores deben defender su reino de los enemigos y pueden atacar a otros reinos para ganar recursos y aumentar su poder. La estrategia es clave para lograr la victoria en los combates, que se desarrollan en un sistema de combate táctico basado en turnos en un entorno 3D.

Los jugadores pueden crear distintos personajes, llamados héroes, de entre varias opciones, cada uno con habilidades y atributos únicos, que serán los personajes que controlan durante el combate.

Los principales objetivos del juego son:

- Construir un reino: Los jugadores tienen que construir y desarrollar un reino desde cero, recolectando recursos, construyendo estructuras y mejorando la economía de su reino.
- Investigación y tecnología: Los jugadores pueden mejorar su tecnología e investigar nuevas formas de construir y producir recursos. La investigación y tecnología son clave para la supervivencia y la prosperidad de su reino.
- Combate y estrategia: Los jugadores tienen que defender su reino de enemigos y atacar a otros reinos para ganar recursos y aumentar su poder. La estrategia es clave para lograr la victoria en los combates.
- Personalización: Los jugadores pueden personalizar sus personajes tanto visualmente como estadísticas. Existen distintas razas para los personajes y habilidades únicas por raza.

Especificaciones del juego:

- Género: estrategia en tiempo real.
- Plataforma: disponible para PC.
- Modo de juego: Un jugador multijugador en línea.
- Gráficos: 2D en vista aldea y 3D en combate.
- Ambientación: fantasía medieval.
- Sistema de combate: táctico basado en turnos.

3.2. Conceptualización

En esta sección se presenta una descripción detallada de los elementos del juego Kingdoms of Lunite, desde una perspectiva de Game Design. Se describen los distintos elementos del juego, como las investigaciones, edificios, héroes, misiones, entre otros, junto con su funcionalidad y relevancia en el contexto del juego.

También, se presenta el contexto del juego en cuanto a la ambientación, el argumento y la historia, para brindar una comprensión más completa del universo en el que se desarrolla.

3.2.1. Universo

El universo del juego es Lunaria, un mundo fantástico donde existe la magia. El universo se compone de un reino con distintas regiones, cada una con sus propios continentes y tierras, que ofrecen posiciones para ser colonizadas por los jugadores.
El universo está habitado por diferentes imperios y reinos, los jugadores, cada uno con su propia estrategia expansionista, y con estados más o menos avanzados a nivel tecnológico y mágico.

La magia es una parte fundamental del universo de Lunaria y se origina a partir de la Lunita, un mineral que se forma a partir de la esencia lunar. La Lunita es un recurso valioso y codiciado, que puede ser extraído y utilizado para desarrollar tecnología mágica y mejorar las habilidades de los héroes.
Los jugadores pueden explorar el universo de Lunaria y pueden construir sus aldeas en los diferentes continentes y mejorar sus tecnologías y habilidades mágicas.

Cada servidor del juego representa un reino. Dependiendo de la configuración aplicada, el reino dispone de más o menos regiones. Cada región dispone de múltiples continentes y asimismo cada continente cuenta con 11 posiciones fijas. Esta división forma la estructura del juego donde los jugadores pueden crear aldeas en posiciones que encuentren libres.

Esta división ofrece diversos beneficios en un juego de fantasía medieval:

- Permite a los jugadores explorar y expandirse a través de diferentes regiones del mundo del juego, lo que daría lugar a nuevos desafíos y oportunidades.
- Facilita la creación de relaciones y conflictos entre los jugadores, ya que podrían formar alianzas o enfrentarse a otros jugadores en diferentes regiones del mundo.
- La posición en el reino de las aldeas determina la distancia entre ellas.

3.2.2. Aldeas

En Kingdoms of Lunite, la aldea es la entidad central sobre la que pivotan todas las mecánicas disponibles. Cada aldea es un asentamiento que es controlado y gestionado por un jugador, y está formada por un conjunto de edificios y unidades que se construyen y desarrollan a lo largo del tiempo.

Las aldeas son el punto de partida para la expansión de los imperios en el juego, y los jugadores deben construir y mejorar sus aldeas para aumentar su producción, reclutar unidades y expandir su territorio. Cada aldea tiene una serie de recursos que se pueden explotar, como minas y canteras, que se utilizan para construir y mejorar los edificios y unidades de la aldea.

Dentro de la estructura del reino, las aldeas se establecen en una de las 11 posiciones disponibles en cada uno de los continentes. Cada posición ofrece diferentes ventajas y desventajas en cuanto a la obtención de recursos, lo que afecta el tamaño y la producción de la aldea.

3.2.3. Recursos

El juego cuenta con cuatro tipos de recursos: hierro, madera, lunita y comida. Cada recurso es necesario para diferentes aspectos del desarrollo del juego.


Figura 7: Recurso Hierro


Figura 8: Recurso Madera


Figura 9: Recurso Lunite


Figura 10: Recurso Comida

- Hierro

Es un recurso esencial para cualquier imperio. Se utiliza para mejorar edificios, realizar investigaciones y producir unidades. La mina de hierro es la estructura responsable de producir dicho metal. A medida que se actualiza, se vuelve más eficiente en la extracción de hierro de la tierra, lo que aumenta su rendimiento por hora. Sin embargo, con cada actualización, la cantidad de comida necesaria para mantener la productividad de la mina también aumenta. El almacenamiento de hierro y la bóveda de hierro son estructuras para almacenar hierro y protegerlo de los ataques enemigos.


Figura 11: Mina de Hiero


Figura 12: Almacén de Hierro


Figura 13: Bóveda de Hierro

- Madera

Es un recurso vital para la mejora y construcción de edificios. La estructura encargada de producir madera es el aserradero, y al igual que la mina de hierro, a medida que se actualiza, se vuelve más eficiente en la producción de madera, pero aumenta su necesidad de comida. El almacenamiento de madera y la bóveda de madera son estructuras para almacenar madera y protegerla de los ataques enemigos.


Figura 14: Aserradero


Figura 15: Almacén de Madera


Figura 16: Bóveda de Madera

- Lunita

Es un recurso raro y mágico obtenido a través de la extracción de piedra lunar. Cada nivel de actualización de la mina de lunita aumenta la tasa de producción de lunita y requiere una cantidad de comida mayor para explotarla. La lunita se utiliza para propósitos mágicos y tiene propiedades beneficiosas, además se utiliza como energía para las unidades y como pago para los héroes. El almacenamiento de lunita y la bóveda de lunita son estructuras para almacenar lunita y protegerla de los ataques enemigos.


Figura 17: Mina de Lunita


Figura 18: Almacén de Lunita


Figura 19: Bóveda de Lunita

- Comida

Es un recurso necesario para mantener la producción de recursos del imperio. La granja es la estructura encargada de producir alimentos. A medida que se actualiza, produce más alimentos por hora. La comida es esencial para mantener un equilibrio en el imperio y no agotar los recursos de los jugadores.


Figura 20: Granja

3.2.4. Edificios

En el juego, la construcción de edificios es esencial para el progreso y la prosperidad de cualquier imperio. Los edificios ofrecen beneficios significativos en términos de recursos, defensa y producción.

- Cuartel

El Cuartel es una piedra angular de cualquier fuerza militar fuerte, permitiendo a los emperadores reclutar héroes para liderar sus ejércitos en la batalla. A medida que el edificio

se actualiza, se vuelve más espacioso, permitiendo el alojamiento de héroes adicionales. Cada nivel del edificio aumenta el número máximo de héroes que pueden ser alojados.

Figura 21: Cuartel

Figura 22: Universidad

- Universidad

La Universidad es el centro de conocimiento y sabiduría en el imperio, donde expertos investigadores se reúnen para empujar los límites de lo posible. Las tecnologías desbloqueadas a través de la investigación en la Universidad conceden acceso a herramientas poderosas, como construcciones más avanzadas, unidades más rápidas y una mejor gestión de recursos.

- Academia

La Academia es un edificio vital para cualquier emperador que quiera mantener a sus héroes actualizados con las últimas habilidades y capacidades. Si un emperador ha investigado una habilidad en particular y la Academia ha alcanzado el nivel requerido, sus héroes pueden aprender nuevas técnicas basadas en su raza y nivel.

Figura 23: Academia

Figura 24: Hospital

- Hospital

El Hospital es un edificio crucial para cualquier imperio que busque mantener su ejército en condiciones óptimas. Cuando los héroes caen en combate, son enviados al hospital para recuperar su salud. No pueden ser utilizados en combate hasta que tengan al menos 1 punto de salud. Con cada nivel del Hospital, se pueden tratar más héroes de forma simultánea.

- Taller

El Taller es responsable de la producción de las unidades, como carros de transporte de recursos, trampas de pesca y fantasmas exploradores para espionaje. A medida que el Taller es mejorado, la velocidad de producción de estas unidades aumenta, lo que permite un transporte y comercio más eficiente entre diferentes aldeas e incluso con otras civilizaciones.

Figura 25: Taller

Figura 26: Taller de Constructores

- Taller de constructores

El Taller de Constructores es una estructura esencial para cualquier constructor aspirante. A medida que el taller es mejorado, los constructores tienen acceso a nuevas tecnologías y técnicas que les permiten construir edificios de manera más rápida y eficiente.

- Taller de ingenieros

El Taller de Ingenieros es una instalación de construcción altamente especializada que emplea a maestros artesanos e ingenieros para diseñar y construir edificios y unidades con una eficiencia increíble. A medida que el taller es mejorado, los artesanos se vuelven más hábiles y los ingenieros desarrollan nuevas técnicas y métodos para construir edificios y unidades a una velocidad más rápida.

Figura 27: Taller de Ingenieros

Figura 28: Oasificadora

- Oasificadora

El Generador de Oasis permite a los jugadores expandir sus territorios convirtiendo la tierra baldía en tierras de cultivo productivas. Cada nivel del generador desbloquea 2 nuevos campos en la aldea.

3.2.5. Investigaciones

En Kingdoms of Lunite, una de las claves para el éxito en la construcción de un reino próspero y poderoso es la investigación. Las investigaciones son esenciales para mejorar la producción, desbloquear nuevas funcionalidades, mejorar a los héroes y aumentar la velocidad de las unidades.

En la sección Universidad del juego, los jugadores pueden descubrir tecnologías únicas y poderosas que les permiten mejorar sus aldeas y fortalezas de una manera significativa. Desde la agricultura hasta la metalurgia y la mecánica avanzada, cada tecnología desbloquea nuevas posibilidades para los jugadores y los ayuda a avanzar en su partida.

Lista investigaciones básicas:

- Matemáticas

La investigación de Matemáticas es esencial para cualquier civilización ya que desbloquea varios avances tecnológicos. Sin las Matemáticas, las civilizaciones pueden tener dificultades para progresar y mantenerse competitivas frente a otras.

Figura 29: Matemáticas

Figura 30: Metalurgia

- Metalurgia

Investigación base que desbloquea distintos árboles de tecnología, además cada nivel de investigación de metalurgia aumenta la resistencia de los carros de carga a los ataques enemigos y mejora la producción de las minas.

- Herrería

Con esta investigación, la civilización puede construir un taller, que sirve para la fabricación de diversas herramientas y equipos. Este avance permite la creación de maquinaria y armas avanzadas que mejoran la producción del Aserradero.

Figura 31: Herrería

Figura 32: Mecánica

- Mecánica

La investigación de Mecánica es fundamental para el desarrollo de cualquier civilización, ya que permite la creación de métodos avanzados de transporte y mejora el rendimiento de las unidades de carga.

- Hidrología

La investigación de Hidrología es fundamental para cualquier civilización que busque desarrollar su industria agrícola. Con cada nivel de investigación de Hidrología, la capacidad de la civilización para administrar los recursos hídricos mejora, lo que incrementa el ritmo de producción de comida por hora.

Figura 33: Hidrología

Figura 34: Exploración

Lista investigaciones avanzadas:

- Exploración

La investigación de Exploración proporciona el conocimiento necesario para colonizar nuevos territorios. Con cada 2 niveles de investigación de Exploración, una civilización adquiere la capacidad de establecer una nueva aldea en un lugar desocupado.

- Logística

La investigación de Logística permite la creación de misiones, que pueden transportar unidades y recursos entre diferentes ubicaciones. Con cada nivel de investigación de Logística, se disponen de más misiones, permitiendo una mayor flexibilidad y eficiencia tanto en el comercio como en las misiones militares.

Figura 35: Logística

Figura 36: Oasificación

- Oasificación

La investigación de Oasificación es un logro notable en el campo de la hidrología y la agricultura. Cada nivel de investigación de Oasificación aumenta la cantidad de tierra que puede ser recuperada mediante la construcción de Oasificadoras, proporcionando una fuente valiosa de alimentos y recursos para la creciente aldea.

- Piscicultura

La investigación de Piscicultura es un avance clave que permite a las civilizaciones optimizar sus Trampas de Peces y aumentar su rendimiento. Al implementar esta investigación, las civilizaciones pueden expandir sus capacidades pesqueras y ser más autosuficientes en su producción de alimentos.

Figura 37: Piscicultura

Figura 38: Red Académica

- Red académica

Esta tecnología de investigación permite que varias universidades colaboren y compartan conocimientos, lo que resulta en un progreso acelerado de la investigación.

Lista investigaciones militares:

- Sanación

Esta investigación es esencial para cualquier civilización que quiera mantener a sus héroes en las mejores condiciones posibles. Cada nivel de investigación de sanidad aumenta la velocidad de la curación de los hospitales, lo que permite que los héroes heridos regresen al campo de batalla más rápidamente.

Figura 39: Sanación

Figura 40: Artes Marciales

- Artes marciales

Con cada mejora, las propiedades de ataque y defensa de los héroes aumentarán un 10%, lo que los hace aún más formidables en el campo de batalla.

- Artes arcanas

Con cada mejora, las propiedades de ataque y defensa mágicos de los héroes aumentarán un 10%.

Figura 41: Artes Arcanas

Figura 42: Fortaleza

- Fortaleza

Con cada nivel de investigación de fortaleza completado, los puntos máximos de salud de los héroes aumentarán un 10%, lo que los hace más duraderos y mejor equipados para sobrevivir a los rigores de la batalla.

- Alquimia

Cada nivel de esta investigación incrementa en 10% los puntos totales de chakra disponibles en los combates por los héroes.

Figura 43: Alquimia

Figura 44: Brujería

Lista investigaciones mágicas:

- Brujería

La brujería es un campo complejo y esotérico que profundiza en los secretos de lo arcano. Es una investigación vital que permite a las civilizaciones aprovechar el poder de la magia y utilizarlo para diversos propósitos. Con el estudio de la brujería, las civilizaciones pueden aprender a lanzar hechizos poderosos, convocar criaturas místicas e incluso controlar los elementos.

- Conjuración

Con cada nivel de Conjuración, los académicos obtienen la capacidad de crear unidades avanzadas de Espectros Espía, capaces de infiltrarse en el territorio enemigo y recopilar información vital.

Figura 45: Conjuración

Figura 46: Destrucción

- Destrucción

Con la investigación de Destrucción, los magos pueden desbloquear los secretos de la magia destructiva y enseñarla a otros en la Academia.

- Restauración

Esta investigación desbloquea una amplia gama de hechizos que se pueden usar para curar lesiones en el campo de batalla.

Figura 47: Restauración

Figura 48: Alteración

- Alteración

La investigación de Alteración permite a los héroes aprovechar el poder de la magia para moverse más rápidamente y de manera más eficiente entre aldeas al dominar las técnicas de alteración de la realidad.

3.2.6. Unidades

En el juego, la interacción pacífica entre aldeas y jugadores se lleva a cabo mediante las unidades de transporte.

- Carro de Carga Pequeño

Los Carros de Carga Pequeños se utilizan para el transporte de recursos entre aldeas. Con una capacidad de carga de 5000 unidades, pueden mover eficientemente los recursos entre colonias, asegurando su distribución y utilización adecuadas.

Figura 49: Carro de carga pequeño

Figura 50: Carro de Carga Grande

- Carro de Carga Grande

El Carro de Carga Grande es un medio confiable y eficiente para transportar recursos entre aldeas. Son una mejora del Carro de Carga Pequeño, con una capacidad de carga de 25000 unidades, permitiendo mover más recursos en un solo viaje. Aunque son más caros de construir, su capacidad los convierte en una inversión valiosa.

- Trampa para Peces

Las trampas para peces se construyen en el taller y no existe limitación a la hora de crearlas más allá de disponer de los recursos suficientes. Pueden ser destruidas cuando la aldea es atacada y derrotada. La productividad de alimento de las trampas para peces depende en gran medida de la ubicación de la aldea en el continente. Las trampas ubicadas en áreas costeras tendrán un alto rendimiento de alimentos debido a la abundante pesca, mientras que las ubicadas en otras áreas tendrán un rendimiento más bajo.

Figura 51: Trampa para Peces

Figura 52: Especto Espía

- Espectro Espía

El Espectro Espía es una entidad fantasmal que puede ser invocada desde el reino espiritual para ayudar en misiones de reconocimiento. Su velocidad y agilidad los convierten en ideales para infiltrarse en el territorio enemigo y recopilar información vital sobre sus defensas, movimientos de tropas y acopio de recursos. Sin embargo, debido a la naturaleza de su existencia, solo se pueden utilizar una vez y regresan al reino espiritual después de completar su misión.

3.2.7. Héroes

Los héroes en el juego se presentan con un enfoque único para cada raza, permitiendo que cada una tenga su propio árbol de habilidades. Esta estrategia orienta a las razas a funcionar como clases controladas, asegurando que cada personaje tenga habilidades únicas y se especialice en diferentes roles.
Cada héroe cuenta con distintos atributos, que incluyen Ataque, Defensa, Ataque Mágico, Defensa Mágica, Puntos de Vitalidad, Puntos de Chakra y Naturaleza.
Cada raza tiene sus habilidades propias, que se pueden desbloquear y mejorar en función del nivel de cada héroe y de las investigaciones en la Academia.

Actualmente el juego cuenta con 2 tipos de razas.

- Caballeros

Los Caballeros son una raza de guerreros altamente disciplinados y bien entrenados en el arte de la lucha con armas a corta distancia. A pesar de su baja movilidad en el campo de batalla, son excelentes defensores gracias a su resistencia y alta capacidad defensiva. Sin embargo, los Caballeros tienen una baja habilidad mágica, lo que los hace vulnerables a los ataques de magia.

- Elfos

Los Elfos son una raza ágil y elegante que se especializan en combate a distancia y magia. Con su movimiento intermedio, son capaces de evadir a los enemigos y lanzar ataques mágicos desde la distancia. Su habilidad mágica es normal, lo que les permite lanzar hechizos con facilidad y precisión. Los Elfos también son competentes en el uso de armas a distancia, pero no son tan hábiles en combate cuerpo a cuerpo como los Caballeros.

3.2.8. Misiones

En el juego existen diferentes tipos de misiones que los jugadores pueden realizar para avanzar y obtener recursos. Estas misiones se dividen en dos categorías: civiles y militares.

Dentro de las misiones civiles se encuentran tres tipos: transporte, despliegue y colonización. La misión de transporte permite a los jugadores mover recursos y unidades entre aldeas propias, pero también entre aldeas de distintos jugadores, lo que puede ser útil para establecer alianzas o comerciar.

Por otro lado, la misión de despliegue permite a los jugadores mover unidades y héroes desde una aldea propia a otra aldea propia, incluyendo los recursos que se carguen en la misión. Por último, la misión de colonización es una forma de crear una aldea nueva en una posición libre, lo que puede ser beneficioso para expandir el territorio del jugador.

Dentro de las misiones militares se encuentran dos tipos: espionaje y ataque. La misión de espionaje permite a los jugadores obtener información sobre la aldea que se quiere atacar, para conocer sus fortalezas y debilidades.

La misión de ataque, en cambio, busca robar los recursos de la aldea atacada. Si la aldea atacada se defiende, se crea un combate en el que los jugadores deben enfrentarse a los héroes defensores. Si la aldea atacada no se defiende, el jugador que ha creado la misión puede saquear sus recursos con éxito.

En Kingdoms of Lunite, y en general en los juegos de estrategia en tiempo real, las misiones entre aldeas no ocurren instantáneamente. Cada unidad tiene una velocidad base que, en conjunto con la distancia entre las aldeas, determina el tiempo de trayecto entre ellas. Este factor es importante para que los jugadores tengan en cuenta al planificar y llevar a cabo sus misiones.

Además, es importante destacar que la velocidad de las unidades no solo influye en las misiones civiles, sino también en las militares. En el caso de los ataques, el tiempo de viaje es crucial, ya que los jugadores pueden sorprender a sus enemigos si llegan a su aldea más rápido de lo que esperan si en la misión de ataque no incluyen unidades de carga que son más lentas que los héroes necesarios para realizar el ataque. En contraprestación, si un ataque se realiza sin unidades de carga, dispondrá de menos capacidad para cargar los recursos saqueados y será menos rentable.

Por lo tanto, es importante que los jugadores tengan en cuenta el tiempo de viaje al planificar sus misiones y estrategias, ya que puede tener un gran impacto en el resultado final. La gestión del tiempo y la velocidad de las unidades son factores clave para lograr el éxito en los juegos de estrategia en tiempo real.

3.2.9. Combate

En Kingdoms of Lunite, los combates se llevan a cabo en turnos tácticos, con un máximo de tres personajes por jugador. Si el atacante gana, puede robar recursos del defensor. Una vez que una misión de ataque llega a su destino, el defensor tiene un breve tiempo para decidir su defensa. Si no está activo, el saqueo se llevará a cabo sin combate.

Dentro del combate, cada personaje controlado por el jugador puede realizar hasta cuatro acciones por turno, que incluyen cambiar de posición, habilidades ofensivas/defensivas y una acción que intercambia puntos de vitalidad por puntos de chakra. Estos puntos de chakra son necesarios para realizar habilidades.

Cada turno, cada personaje puede realizar 2 acciones del tipo habilidad y puede elegir realizar 2 habilidades ofensivas o 2 defensivas. No hay una imposición de 1 ofensiva y 1 defensiva, el jugador puede decidir libremente cómo gestionar las 2 posibles acciones de habilidades.

Una vez que se realiza el turno de un jugador y este jugador realiza ataques al enemigo, el daño generado no se aplica directamente, si no que en el turno siguiente el rival tiene la posibilidad de aplicar movimientos defensivos para contrarrestar el daño recibido. Cuando finaliza el turno del rival es cuando el daño enviado previamente se aplica según las acciones realizadas por el defensor.
Si un personaje tiene pendiente defender un daño que, si no lo logra detener significa que cae derrotado, pero el jugador decide utilizar su turno para generar daño al rival, es válido. Cuando acabe su turno, se debilitará, pero el daño enviado tendrá que ser defendido por el otro jugador en el siguiente turno si es que el jugador cuenta con algún personaje vivo.

Los héroes tienen un rango disponible de movimiento, que varía según su raza. Los jugadores controlan el movimiento libre de su personaje con el teclado y una vez que quieren quedarse en esa posición, lo indican.

Durante el combate, cada habilidad se cataloga con un rango, ya sea corto, medio o largo. Esto indica la distancia a la que el usuario puede utilizarla con mayor efectividad. Si el objetivo de la habilidad no se encuentra dentro del rango adecuado, habrá una penalización al usarla, lo que disminuye su potencia.

Es importante tener en cuenta que también hay un rango máximo en el que no se permite atacar. Si los personajes se encuentran muy lejos fuera de cualquier tipo de rango, no podrán utilizar sus habilidades ofensivas. Esto fomenta la estrategia y el posicionamiento en el campo de batalla, ya que los jugadores tendrán que moverse y posicionarse adecuadamente para poder atacar con éxito.

3.3. Modelo de negocio

El modelo de negocio que se aplicaría para el juego Kingdoms of Lunite sería el freemium, el cual permite a los jugadores descargar y jugar el juego de forma gratuita, pero se les ofrece la opción de comprar elementos virtuales dentro del juego para mejorar su experiencia de juego.

Esta estrategia de negocio se debe a que el mercado de juegos se ha vuelto cada vez más competitivo y es necesario ofrecer una opción atractiva para atraer a los jugadores (Adell E. Ferran, s.f.). Al ofrecer el juego de forma gratuita, se elimina cualquier barrera inicial para que los jugadores lo prueben. Además, este modelo ofrece la posibilidad de generar ingresos constantes a través de las compras integradas en el juego.

Para aplicar este modelo de negocio en Kingdoms of Lunite, se ofrecerían elementos virtuales dentro del juego como mejoras cosméticas para los personajes y las aldeas, habilidades especiales, personajes y otros elementos para mejorar la experiencia de juego de los usuarios. Estos elementos estarían disponibles para su compra con el recurso Lunite del juego y con dinero real a través de transacciones en la tienda.

Además, para hacer que el juego sea más atractivo y atraer a más jugadores, se ofrecerían eventos especiales y promociones para los usuarios. Estos eventos podrían incluir regalos gratuitos de elementos virtuales o bonificaciones especiales por ciertas compras dentro del juego. Esto no solo aumentaría la participación de los usuarios, sino que también aumentaría la posibilidad de que realicen compras dentro del juego.

3.4. Estrategia de marketing

El enfoque del trabajo para el branding y la promoción de Kingdoms of Lunite se basará en la creación de una identidad visual fuerte y coherente que transmita los valores y la experiencia

de juego únicos que ofrece el juego. Para lograr esto, se utilizará una combinación de diseño gráfico, redes sociales y marketing en línea (Moreno, J, s.f.).

El plan de promoción incluirá la creación de contenido atractivo para las redes sociales, como imágenes, vídeos y gifs animados, que muestren la jugabilidad, los personajes y los escenarios del juego. También se crearán campañas publicitarias en línea que llegarán a los jugadores interesados en los juegos de rol y estrategia.

La política de precios de Kingdoms of Lunite, como se ha comentado anteriormente, se centrará en una estrategia freemium, que permitirá a los jugadores descargar y jugar el juego de forma gratuita, pero con la opción de adquirir objetos y ventajas adicionales dentro del juego a través de microtransacciones. Esto permitirá que los jugadores se familiaricen con el juego antes de decidir si desean invertir dinero en él.

Como parte principal de la estrategia de venta se ofrecerán distintos servidores y se irán abriendo nuevos de forma periódica. Esto, además de proporcionar una experiencia de juego justa y equilibrada para todos los jugadores, permite que los jugadores tengan la opción de elegir el servidor que mejor se adapte a su estilo de juego y preferencias.

Además, la existencia de múltiples servidores permite que los jugadores compitan entre sí en un entorno justo y competitivo, ya que todos los jugadores comienzan en igualdad de condiciones en cada servidor. Si no hubiera servidores diferentes, los jugadores experimentados podrían monopolizar el juego y hacer que sea menos divertido para los nuevos jugadores.

4. Diseño técnico

En esta sección se detalla todo lo relacionado con el entorno y el diseño de niveles en el proyecto. Se describen los requisitos necesarios para crear un entorno adecuado, incluyendo las herramientas y assets utilizados en el proceso.

Se presenta una visión general de la arquitectura del sistema, destacando las decisiones de diseño relevantes y las herramientas de programación y APIs utilizadas en la construcción del juego.

Todo esto con el objetivo de proporcionar una comprensión completa de cómo se construyó el entorno del juego y cómo se diseñaron los niveles para garantizar una experiencia de juego satisfactoria para el usuario final.

La fase de diseño es fundamental en cualquier proyecto de software, ya que es la que define cómo será la interacción entre los usuarios y el sistema.

4.1. Entorno

El entorno elegido para el desarrollo de Kingdoms of Lunite consta de varias herramientas y tecnologías.
El cliente del juego está desarrollado usando Unity 2021.3.20f1 LTS, una plataforma de desarrollo de juegos muy popular y versátil. La elección de Unity se basa en su facilidad de uso y su capacidad para crear juegos en 2D y 3D, lo que permite a los programadores crear el mundo del juego y dar vida a los personajes y objetos. Además, Unity es una buena opción para el desarrollo de juegos multiplataforma, lo que significa que el juego puede ser distribuido en diferentes dispositivos y sistemas operativos.

Por otro lado, el servidor web se ha desarrollado utilizando .NET 7, una plataforma de desarrollo web creada por Microsoft. La base de datos del juego es SQL Server y para acceder a los datos se ha elegido Entity Framework como ORM, de nuevo dentro del stack de tecnologías Microsoft.
La elección de estas tecnologías se basa en la capacidad de C# para trabajar tanto en el backend como en la programación del gameplay, lo que facilita la tarea de los programadores y mejora la mantenibilidad de la base de código y permite que la misma lógica de negocio codificada en una biblioteca de clases en C# pueda ser usada tanto desde el cliente como desde el servidor.
Además, SQL Server es una base de datos muy escalable y confiable que ofrece un alto rendimiento y una gran capacidad de almacenamiento.

La elección de Azure como plataforma donde se desplegará el servidor se basa en la capacidad de la plataforma para escalar automáticamente y proporcionar un alto nivel de seguridad y disponibilidad. Azure es la mejor opción para el despliegue de aplicaciones web y bases de datos con tecnología Microsoft, lo que facilita el mantenimiento y la gestión del servidor.

La elección de Unity, .net7, SQL Server y Azure como pipeline completo para el entorno de desarrollo del proyecto se basa en la capacidad para trabajar con C# bajo el mismo IDE (Visual Studio 2022), su escalabilidad, rendimiento, seguridad y disponibilidad. Además, estas tecnologías son muy populares y están ampliamente utilizadas en la industria de los videojuegos (Unity) y el desarrollo web (AspNetCore), lo que facilita la búsqueda de recursos y apoyo en línea.

4.2. Requisitos técnicos entorno desarrollo

- Para Unity 2021.3.20f1 LTS, se recomienda un sistema operativo Windows 10 de 64 bits, un procesador con el set de instrucciones SSE2, 16 GB de RAM y una tarjeta gráfica compatible con DirectX 10,11 y 12 (Unity Technologies, 2021).
- Para Visual Studio 2022, se recomienda un sistema operativo Windows 10 de 64 bits, un procesador de X64 y 16 GB de RAM (Microsoft, 2022).
- Para la Web API en .NET 7, se recomienda un sistema operativo Windows Server 2019 o Windows 10 de 64 bits, un procesador de 2,0 GHz o superior, 4 GB de RAM (se recomiendan 8 GB o más para un mejor rendimiento).
- Para SQL Server, se recomienda un sistema operativo Windows Server 2019 o Windows 10 de 64 bits, un procesador de 2,0 GHz o superior, 4 GB de RAM (se recomiendan 8 GB o más para un mejor rendimiento).
- Para Azure, se requiere una suscripción a Azure y una cuenta de Microsoft. Para una carga inicial limitada de jugadores simultáneos en Azure, se recomienda utilizar un plan de servicio de aplicaciones B1. Este plan ofrece un rendimiento suficiente para manejar la carga esperada, junto con características importantes como la capacidad de escalar verticalmente para aumentar el rendimiento, la capacidad de escalar horizontalmente para aumentar la capacidad de procesamiento y la capacidad de crear instancias de clonación para distribuir la carga de trabajo.

4.3. Inventario herramientas usadas

4.3.1. Unity Editor 2021.3.20f1 LTS

Es un entorno de desarrollo integrado (IDE) que se utiliza para la creación de videojuegos y experiencias interactivas. Con él se pueden diseñar, programar y animar elementos en 2D y 3D, así como añadir efectos de sonido y partículas, entre otras funcionalidades.

4.3.2. Unity Multipurpose Avatar 2

Es una herramienta para crear y personalizar avatares en Unity. Con UMA, los desarrolladores pueden crear personajes con un alto grado de personalización, incluyendo la forma, el tamaño y el color de diversas partes del cuerpo, como el cabello, la piel, los ojos y la ropa. UMA utiliza un sistema basado en slots para permitir a los desarrolladores crear personajes complejos a partir de piezas modulares predefinidas.
Además, UMA permite la integración de animaciones y es compatible con varias plataformas de realidad virtual. UMA es una herramienta de código abierto, lo que permite a los desarrolladores personalizar y extender su funcionalidad según sus necesidades.

4.3.3. Visual Studio 2022

Es un IDE desarrollado por Microsoft que se utiliza principalmente para programar en lenguajes como C#, C++ y Visual Basic .NET. Ofrece una amplia variedad de herramientas y funcionalidades para facilitar el proceso de desarrollo de software.

4.3.4. Notepad++

Es un editor de texto gratuito y de código abierto que se utiliza principalmente para la edición de código fuente. Proporciona funciones avanzadas como resaltado de sintaxis, plegado de código, autocompletado de palabras y la posibilidad de trabajar con múltiples archivos al mismo tiempo.

4.3.5. SQL Server Management Studio

Es una herramienta de administración de bases de datos que se utiliza para gestionar y mantener bases de datos SQL Server. Permite crear, modificar y eliminar objetos de la base de datos, así como realizar consultas y ejecutar scripts.

4.3.6. Mixamo.com

Es una plataforma en línea que ofrece una amplia variedad de modelos y animaciones en 3D para su uso en videojuegos y proyectos de animación. Ofrece herramientas para la personalización de los modelos y animaciones, así como una opción para exportarlos directamente a Unity.

4.3.7. GitHub

Es una plataforma en línea de alojamiento de repositorios de código fuente que utiliza el control de versiones Git. Permite a los desarrolladores colaborar en proyectos de forma remota, realizar un seguimiento de los cambios en el código y mantener una versión actualizada del mismo.

4.3.8. Sourcetree

Es una herramienta de gestión de repositorios de Git que proporciona una interfaz gráfica de usuario para realizar operaciones de Git (Atlassian, 2022). Facilita el proceso de clonar, confirmar y fusionar cambios, así como la visualización de la historia de los cambios en un repositorio.

4.3.9. DALL·E

Es un modelo de inteligencia artificial desarrollado por OpenAI que utiliza redes neuronales para generar imágenes a partir de descripciones de texto (OpenAI, 2021). Puede utilizarse para crear imágenes personalizadas y únicas para su uso en proyectos de diseño gráfico, entre otros. En Kingdoms Of Lunite ha sido utilizado para crear distintas imágenes en la escena de gestión de la aldea, como los edificios, investigaciones, unidades, etc.

4.4. Inventario assets usados

Todos los assets listados pertenecen a sus autores y se dispone de la licencia correspondiente para su uso en este proyecto.

4.4.1. Assets UI

- Medieval Kingdom UI (PONETI, 2022)

Es un conjunto de elementos de interfaz de usuario en 2D diseñados para juegos de fantasía medieval. Incluye botones, iconos, ventanas emergentes y elementos de navegación, entre otros. Se ha usado como base principal para toda la UI del proyecto.

- 4000 Fantasy Icons (PONETI, 2022)

Es un paquete de iconos de fantasía que incluye más de 4000 iconos diferentes en diferentes estilos y tamaños, útiles para juegos y aplicaciones de fantasía. Se ha usado para los iconos de las habilidades y atributos de héroes.

- Procedural Circular Health and Progress Bars Pro (Sam Schiffer, 2023)

Es un sistema de barras de progreso y salud para Unity para crear barras animadas en tiempo real. Se ha usado para el indicador de habilidades disponibles en la pantalla de combate.

- Simple Spinner (Hippo, 2022)

Es un asset para Unity que proporciona un spinner simple y personalizable que se puede utilizar en diferentes situaciones, como por ejemplo durante la carga de una escena. Se ha usado para el spinner que se muestra cuando el cliente se comunica con el servidor.

- UGUI MiniMap (Lovatto Studio, 2022)

Es un asset que permite agregar un minimapa en tiempo real a cualquier juego hecho en Unity. Es altamente personalizable. Se ha usado como minimapa en la escena de combate.

- Simple Tools (Gerard Gascón, 2022)

Paquete de utilidades para Unity que incluye una serie de herramientas pensadas para usar en distintos proyectos. En concreto de este paquete se ha usado el sistema de dialogo como base para el sistema de mensajes de ayuda de Kingdoms of Lunite.

4.4.2. Assets VFX

- AAA Stylized Projectiles Vol.1 (Hovl Studio, 2023)

Es una colección de proyectiles de estilo AAA diseñados para juegos de fantasía y ciencia ficción. Se ha usado para los efectos VFX de las habilidades en combate.

- Particle Ribbon (Moonflower Carnivore, 2018)

Paquete que contiene animaciones y efectos de auras. Se ha usado para los efectos durante el combate.

- Loot Beams (Garrexus, 2021)

Es un asset que agrega efectos visuales de partículas a los objetos que el jugador recolecta en el juego. Se ha usado en la escena de combate para diferenciar a los héroes del jugador de los del rival y para marcar a los personajes seleccionados.

- Mobile Force Field (Isle of Assets, 2023)

Es un asset para Unity que permite crear efectos de campo de fuerza ligeros. Se ha usado para marcar hasta donde pueden realizar distintas acciones cada héroe en la escena de combate.

4.4.3. Assets SFX

- Ultimate SFX Bundle (Sidearm Studios, 2021)

Es un paquete de SFX que incluye más de 5000 efectos de sonido de alta calidad para utilizar en juegos y aplicaciones. Se ha usado para los sonidos al realizar habilidades en la escena de combate y para sonorizar la UI.

- Ultimate Music Bundle (Sidearm Studios, 2021)

Es un paquete de música que incluye más de 300 pistas de música de alta calidad en diferentes estilos y géneros para utilizar en juegos y aplicaciones. Se ha usado como música.

- Dialog Text Sound Effects (Alan Dalcastagne, 2021)

Paquete de sonidos diseñados para ser usados en sistemas de dialogo 2D.

4.4.4. Assets 3D

- Gaia Pro 2021 - Terrain & Scene Generator (Procedural Worlds, 2023)

Es un sistema de generación de terrenos para Unity que permite crear escenarios naturales y realistas de manera rápida y eficiente. Incluye herramientas para la colocación de objetos, la creación de biomas y la edición de terrenos. Se ha usado para la creación de la escena de combate, tanto terreno como iluminación.

- Uma assets (Arteria3D, 2021)

Es un conjunto de assets para crear avatares en Unity utilizando el sistema UMA. Incluye modelos 3D de personajes, ropa, accesorios y animaciones, entre otros. Se ha usado para personalizar a los héroes.

4.4.5. Plugins

- DOTween (Demigiant, 2022)

Es una librería para Unity que permite animar fácilmente objetos y propiedades. Proporciona diferentes tipos de interpolación y opciones de configuración para crear animaciones complejas de forma sencilla.

- RestClient (Proyecto26, 2021)

Es una librería para Unity que proporciona una forma sencilla de realizar peticiones HTTP a través de una API RESTful. Proporciona diferentes métodos para realizar peticiones GET, POST, PUT, DELETE, entre otros.

- SmartFormat (Axuno, 2019)

Es una librería para C# que proporciona una forma sencilla de formatear cadenas de texto. Proporciona diferentes opciones de configuración para ajustar el formato según las necesidades del proyecto. Se ha usado para leer el fichero de recursos de texto.

4.5. Arquitectura general del sistema

En esta sección, se presenta una visión general del diseño de la arquitectura del sistema, cliente y servidor, proporcionando información sobre cómo se organiza el software, cómo se interconectan las diferentes partes y qué tipos de tecnologías se han utilizado para implementarlo.
Además, se analizan los principales objetivos del diseño y se explican las decisiones que se han tomado para conseguirlos, de modo que se pueda entender el porqué de la arquitectura elegida y cómo se ha abordado su implementación.

4.5.1. Arquitectura del cliente

El proyecto del cliente está diseñado para ofrecer una experiencia de usuario modular al programador, escalable y fácil de mantener, gracias al uso de scriptable objects, eventos y una nomenclatura específica para la organización del código.

El juego consta de tres escenas, cada una correspondiente a una funcionalidad específica del juego: la escena de login/registro, la escena de gestión de la aldea y la escena de combate. El código del cliente se estructura mediante el uso de scriptable objects, lo que permite la configuración de distintas partes del juego, como las razas activas, los tipos de misiones disponibles, la ropa disponible por raza, entre otros aspectos.

La interfaz de usuario es muy modular, lo que permite una mayor facilidad de mantenimiento y escalabilidad del sistema. Cada vista del menú lateral corresponde a un panel que, a su vez, dispone de distintos tipos de scripts asociados que controlan la lógica de la aplicación. La comunicación entre los distintos scripts se realiza mediante el uso de eventos, lo que permite la implementación de una arquitectura desacoplada y escalable.

Para una mejor organización y estructuración del código, se ha adoptado una nomenclatura específica. Los scripts que controlan la interfaz de usuario se denominan "Controladores", mientras que los scripts que gestionan una mecánica específica del juego se denominan "Managers". Por ejemplo, como managers se tiene el HudManager o el MissionPanelManager que controlan uno todos los componentes del Hud y otro la mecánica de crear misiones, mientras que el ResourceElementController controla como se visualiza la cantidad de recursos de los que se dispone en la aldea actualizando una serie de componentes de texto.
Otro aspecto destacable de la arquitectura del cliente es que al iniciar la ejecución y cuando el usuario hace login, se descarga los valores que controlan el juego del servidor específico al que se ha accedido. Estos valores incluyen factores de velocidad para los costes de los edificios, unidades, producción de recursos, etc. Además, los valores de rango y penalización para los combates también se reciben del servidor.

La capacidad de ajustar estos valores para crear universos distintos en el juego es una característica importante del sistema que permite personalizar la experiencia de juego para diferentes servidores. Esto significa que los jugadores pueden elegir participar en un universo donde las unidades se mueven más rápido, los costes son más bajos o donde los combates tienen diferentes reglas.

Desde el punto de vista de los valores que configuran el universo, se ha diseñado la arquitectura para considerarla agnóstica a los datos, para que no dependa de ellos. El objetivo ha sido separar el cliente de los datos específicos del juego, para facilitar la gestión de los cambios en la configuración del juego y permitir una mayor flexibilidad y escalabilidad.

En la arquitectura del cliente, se ha creado una DLL en C# que contiene todas las clases y la lógica de negocio compartida por el proyecto web del backend y el proyecto Unity. Esta librería es esencial para garantizar la coherencia entre el servidor y el cliente en cuanto a las mecánicas del juego y los cálculos de costes y distintas duraciones.
La inclusión de esta DLL permite que tanto el cliente como el servidor realicen las mismas operaciones en el código compartido, lo que asegura una experiencia de juego coherente y equilibrada. Además, la creación de esta DLL también mejora la eficiencia del desarrollo de software, ya que los cambios en la lógica de negocio solo deben realizarse en un lugar y se aplicarán automáticamente tanto en el backend como en el cliente.

4.5.2. Arquitectura del servidor

La arquitectura del servidor del juego está diseñada utilizando diversos patrones de diseño, incluyendo el patrón CQRS, el patrón de Repositorio y el patrón de Unidad de Trabajo. Además, la aplicación utiliza Hangfire para procesamiento de trabajos en segundo plano y Swagger para documentación y pruebas de la API.

El patrón CQRS se ha utilizado para separar las responsabilidades de manejar comandos de escritura y consultas de lectura en diferentes componentes de la aplicación, lo que mejora la escalabilidad, el rendimiento y la flexibilidad (Day, J., 2022). El patrón de Repositorio se ha utilizado para manejar el acceso a los datos y reducir el acoplamiento entre las capas de la aplicación, lo que facilita la evolución y el mantenimiento de la aplicación (Microsoft, 2023). El patrón de Unidad de Trabajo se ha utilizado para manejar la gestión de transacciones y garantizar la consistencia de los datos en la base de datos.

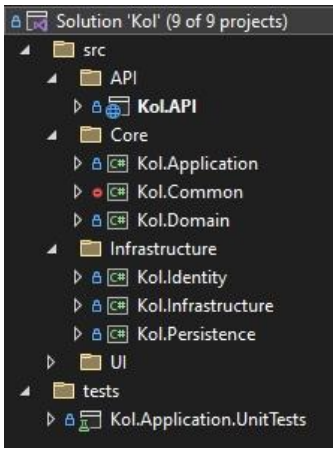

Figura 53: estructura solución servidor

Entity Framework se utiliza como una capa de mapeo de objetos relacionales para interactuar con la base de datos SQL Server. Esto facilita la interacción con la base de datos y reduce la cantidad de código necesario para realizar operaciones CRUD.
Hangfire se ha utilizado para el procesamiento de trabajos en segundo plano, lo que ayuda a mejorar la escalabilidad y el rendimiento de la aplicación, ya que los trabajos se ejecutan de forma asíncrona. Los trabajos en segundo plano que se ejecutan son comprobaciones de si las distintas acciones que los jugadores realizan, como misiones, combates, mejora de edificios, se han completado.

Swagger se utiliza para documentar y probar la API del servidor. Esto ayuda a los desarrolladores a comprender cómo interactuar con la API y a probarla antes de integrarla en el cliente del juego.

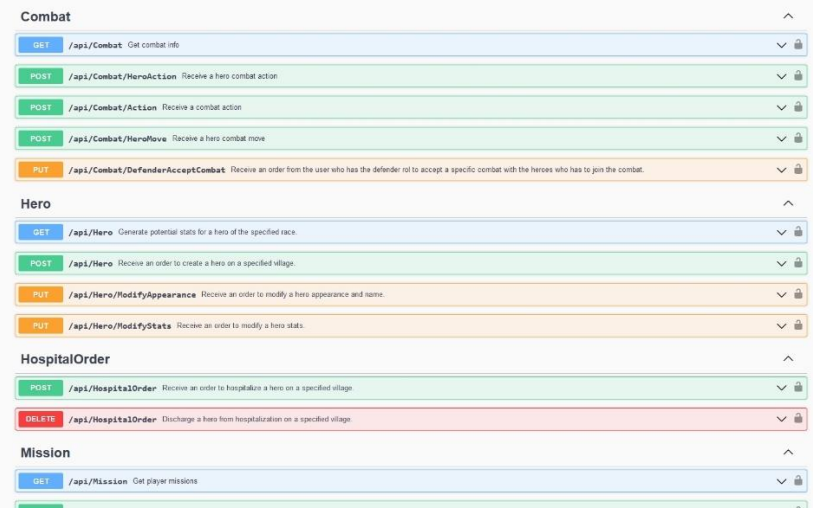

Figura 54: swagger

En el servidor se incluye una capa de caché que mejora el rendimiento al mantener los datos de los jugadores en memoria durante un tiempo determinado mientras están activos, evitando accesos más costosos a datos innecesarios. Esto significa que el servidor puede responder de manera más rápida y eficiente a las solicitudes de los jugadores, ya que no tiene que buscar constantemente en la base de datos para recuperar información (Microsoft, 2023).
La capa de caché también permite una mejor escalabilidad, ya que puede reducir la carga en la base de datos y reducir el tiempo de respuesta para las solicitudes simultáneas.

4.5.3. Comunicación cliente-servidor

La comunicación entre el cliente y el servidor es una parte fundamental en cualquier aplicación que requiera acceso a datos o servicios remotos. En este proyecto, la aplicación cliente es una app Unity y el servidor es un api REST en .NET 7. Ambos se comunican mediante internet, utilizando protocolos estándar para el intercambio de información.

En particular, en el cliente se utiliza un framework llamado proyecto26/RestClient, el cual es una biblioteca de código abierto basada en promesas que ofrece un cliente REST y HTTP

para Unity. Este framework se integra fácilmente en Unity, lo que permite a los desarrolladores realizar solicitudes HTTPS y REST con facilidad.

Internamente, el proyecto26/RestClient utiliza UnityWebRequest y corutinas para realizar las solicitudes al servidor y procesar las respuestas. UnityWebRequest es una clase de Unity que permite realizar solicitudes HTTP y REST de manera eficiente, mientras que las corutinas son una forma de programación asíncrona que permite a los desarrolladores realizar múltiples tareas al mismo tiempo sin bloquear el hilo principal de Unity.

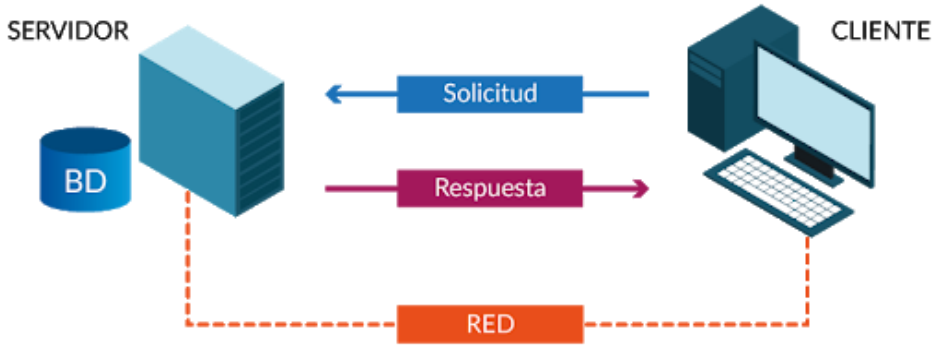

Figura 55: Comunicación cliente-servidor

Por su parte, en el servidor se utilizan los protocolos HTTPS y REST para procesar las peticiones recibidas desde el cliente. Estos protocolos permiten que la comunicación sea escalable, confiable y segura, ya que se basan en estándares ampliamente aceptados en la industria.

En este caso, tanto el cliente (la aplicación Unity) como el servidor (la API REST en .NET7) envían y reciben datos en formato JSON. Cuando se envían datos desde el cliente al servidor, el cliente convierte los datos del juego a formato JSON y los envía a través de una petición HTTPS. El servidor, por su parte, recibe la petición y analiza los datos JSON para extraer la información necesaria.

Cuando el servidor responde con datos al cliente, también lo hace en formato JSON. El servidor procesa la solicitud, extrae los datos necesarios y los convierte en formato JSON antes de enviarlos de vuelta al cliente a través de una respuesta HTTPS. El cliente, por su parte, recibe los datos en formato JSON y los analiza para extraer la información necesaria y la usa en el juego.

4.5.4. Bases de datos

El proyecto, como ya se ha comentado, utiliza SQL Server como sistema gestor de bases de datos.

El servidor se comunica con tres bases de datos distintas para garantizar un rendimiento óptimo y una gestión eficiente de los datos. La primera de ellas se encarga del login de los usuarios, almacenando los datos de autenticación de estos. La segunda es la encargada de almacenar los datos del juego, incluyendo información sobre héroes, aldeas, misiones, entre otros. Por último, se utiliza una base de datos específica para la gestión de las tareas programadas a través de Hangfire.

Cada base de datos tiene una estructura específica, diseñada para cumplir con sus objetivos específicos.

- Datos de usuario

En la base de datos de usuarios, se utilizan tablas para almacenar información de los usuarios, como su nombre de usuario y contraseña, así como también información adicional de autenticación y seguridad.

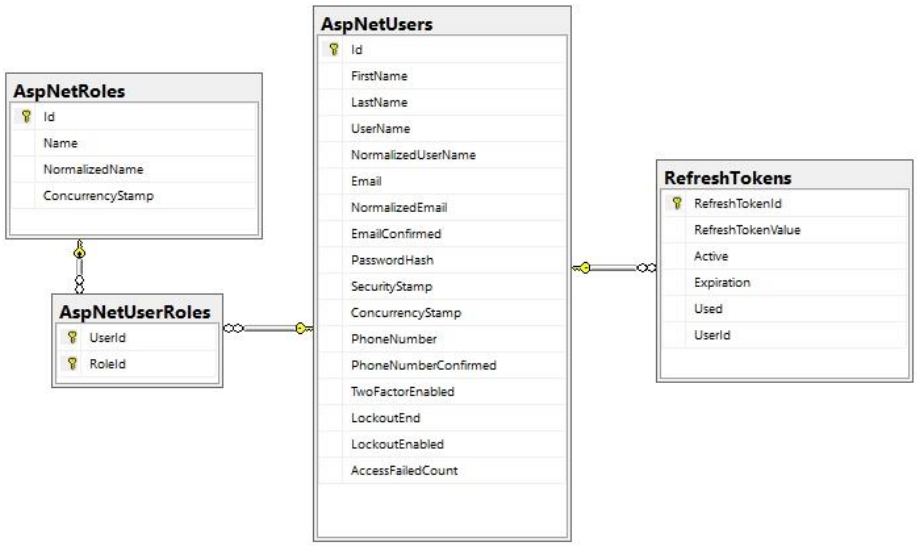

Figura 56: esquema identity

La separación de las bases de datos con los datos del juego y los datos de autenticación de los usuarios es una necesidad importante para el proyecto. Esta solución ofrece múltiples beneficios, como la mayor seguridad de los datos de los usuarios, la organización de los datos del juego por universos, evitar la replicación de datos privados de los jugadores y, además, permite utilizar una base de datos de usuarios común para todos los universos, lo que facilita la administración y gestión de la información (Bigelow Stephen, 2022).

- Datos del juego

En la base de datos de datos del juego, se utilizan tablas para almacenar información sobre los personajes, objetos y misiones, así como también tablas para establecer relaciones entre ellos. Cada una de estas tablas representa una entidad dentro del sistema y están relacionadas entre sí mediante claves primarias y foráneas, lo que permite establecer las relaciones entre las diferentes entidades.

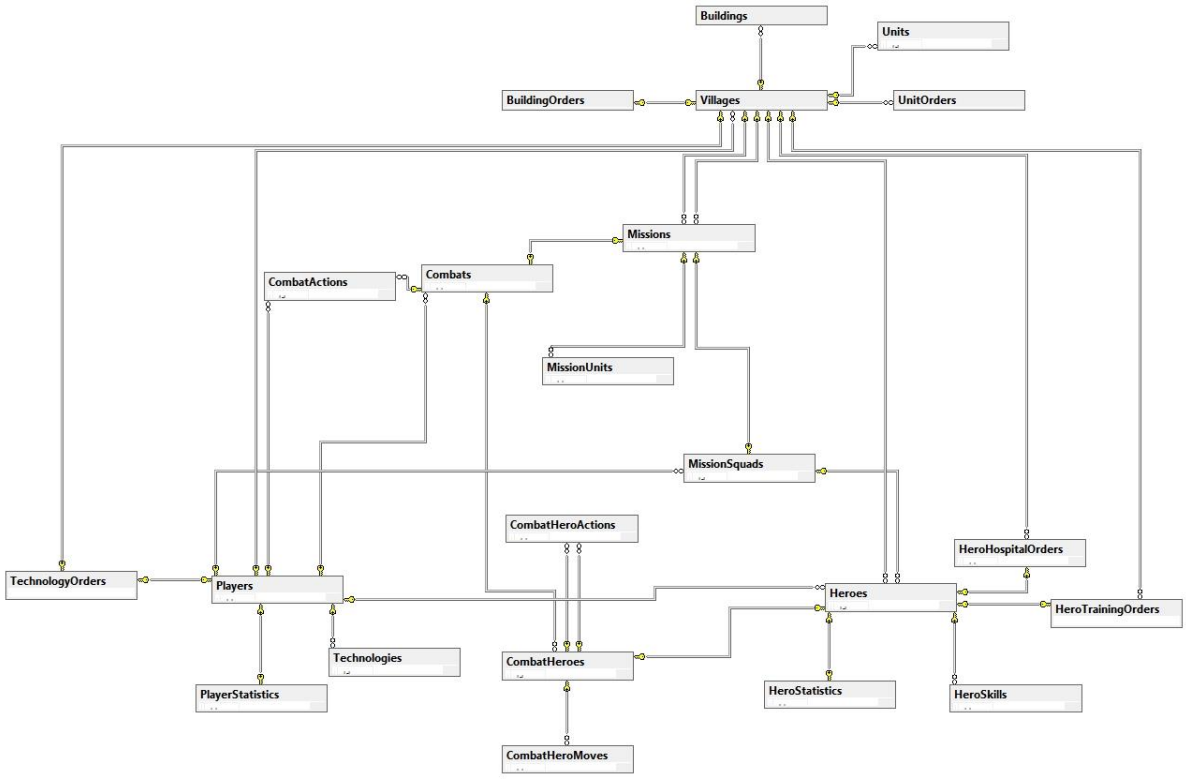

Figura 57: esquema datos de juego

Además, se utilizan índices para mejorar el rendimiento de las consultas en la base de datos, permitiendo realizar búsquedas de manera más eficiente. Se utilizan tanto índices agrupados como no agrupados, en función de las necesidades de cada tabla.

Una técnica comúnmente utilizada para mejorar el rendimiento de las consultas en una base de datos es la desnormalización. Sin embargo, esta técnica debe ser analizada cuidadosamente antes de implementarla, ya que puede presentar problemas de mantenimiento y escalabilidad a largo plazo (kryptonsolid, s.f).

Un ejemplo de desnormalización en los datos del juego podría ser llevado a cabo en la tabla Technologies. En esta tabla, cada registro incluye el identificador del jugador y el identificador del tipo de tecnología. Por lo tanto, para las investigaciones de cada jugador, se generarán múltiples registros, uno por cada tecnología investigada, lo que resultará en una tabla con un gran número de registros.

Si se opta por desnormalizar esta tabla y no se siguen las formas normales, se utilizará un registro por jugador, con 2 campos por cada tipo de tecnología (techtype, techlevel), teniendo 1 registro por cada jugador en lugar de como máximo 20 registros por jugador. Aunque esto podría mejorar el rendimiento de las consultas y, en algunos casos, reducir el espacio en disco, puede resultar en problemas de integridad de datos y escalabilidad a medida que aumente el número de jugadores y las investigaciones.

Por lo tanto, es importante considerar cuidadosamente la desnormalización y evaluar los posibles problemas a largo plazo antes de implementarla en un proyecto. En el caso de Kingdoms of Lunite, se ha decidido no utilizarla debido a los problemas potenciales que puede ocasionar a largo plazo y porque el servidor ya incluye una capa de caché para mejorar el rendimiento de las lecturas.

- Datos tareas

Por su parte, la base de datos específica para Hangfire, creada por el propio framework, se encarga de almacenar y gestionar las tareas programadas en el servidor. Esta base de datos está diseñada específicamente para la gestión de estas tareas, y utiliza procedimientos

almacenados y otras herramientas de SQL Server para garantizar un rendimiento óptimo y una alta disponibilidad.

4.5.5. Seguridad

La seguridad es un aspecto fundamental en cualquier proyecto que implique la gestión de datos sensibles. En el caso de Kingdoms of Lunite, se han implementado diversas medidas de seguridad para proteger la información de los usuarios y garantizar la integridad de los datos.

Una de las medidas de seguridad implementadas en el proyecto es el cifrado de datos. Todas las comunicaciones entre el cliente y el servidor se realizan a través de conexiones seguras mediante el protocolo HTTPS, lo que garantiza que los datos transmitidos están cifrados y protegidos frente a posibles interceptaciones.

Otra medida de seguridad importante es la autenticación de usuarios. Para ello, el servidor utiliza JWT Bearer Tokens con un sistema de refresco automático y el algoritmo de cifrado HS256. JWT es un estándar abierto que define un formato compacto y autónomo para la transmisión de información segura entre partes como un objeto JSON. El uso de estos tokens permite verificar la identidad del usuario y garantizar que solo los usuarios autenticados puedan acceder a los datos del juego. El sistema de refresco, por su parte, permite renovar los tokens de manera automática y segura, evitando que los usuarios tengan que volver a iniciar sesión constantemente.

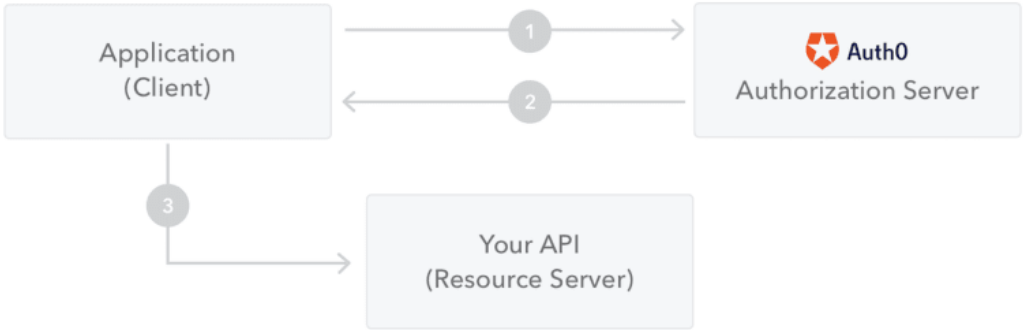

Figura 58: esquema jwt

La autorización es el propósito principal de los tokens JWT, y se utiliza comúnmente después de que un usuario haya iniciado sesión en un sistema. Cada llamada posterior al servicio que el usuario realice deberá incluir el JWT correspondiente, lo que le permitirá acceder a rutas, servicios o recursos que están restringidos únicamente a usuarios con un token válido. En particular, los tokens JWT son ampliamente utilizados en la actualidad en la funcionalidad de inicio de sesión único (SSO, por sus siglas en inglés), debido a su tamaño compacto y capacidad para ser utilizados en diferentes dominios.

Además, se utiliza Identity Core de Microsoft para la gestión de usuarios y roles. Este framework proporciona un conjunto de funcionalidades y herramientas para la autenticación y autorización de usuarios en aplicaciones web, lo que permite garantizar la seguridad en la gestión de usuarios y roles (Microsoft, 2022).

4.6. Diseño de niveles

El diseño de nivel es una parte fundamental en la creación de un videojuego, ya que tiene una gran influencia en la experiencia del jugador. En el caso de un juego de estrategia con combate por turnos, el mapa debe ser diseñado cuidadosamente para que potencie las mecánicas del combate y que ofrezca una experiencia de juego desafiante y satisfactoria.

El objetivo principal de diseño al crear el mapa único del juego ha sido crear un mapa lo suficientemente grande para que los jugadores tengan la oportunidad de esconderse y cubrirse durante el combate. Además, el mapa debe ofrecer posibilidades de juego distintas y variadas, para que los jugadores puedan utilizar diferentes estrategias y tácticas en función de sus habilidades y preferencias (Craig Stern, 2013).

En este sentido, el diseño del nivel se ha realizado de forma que permita el desarrollo de un combate por turnos de 3 contra 3 personajes. Se ha tenido en cuenta la disposición de los elementos del mapa, como obstáculos y terreno, para que puedan influir en el combate y en las decisiones de los jugadores, ya que una de las mecánicas es que no se permiten los ataques sino hay una visión libre en línea recta entre el atacante y el objetivo.

Figura 59: vista área mapa completo

El tamaño del mapa ha sido rigurosamente evaluado con la finalidad de garantizar que ofrezca una experiencia de juego enriquecedora, que permita a los jugadores implementar distintas estrategias y desarrollar un enfoque táctico adecuado teniendo en cuenta la distancia que pueden recorrer las distintas razas de héroes durante cada turno.

En particular, se ha procurado que el mapa sea lo suficientemente grande como para permitir un rápido avance de los jugadores, así como para posibilitar el ocultamiento de personajes en distintas partes del escenario en caso de que se elija una estrategia más cautelosa y defensiva.

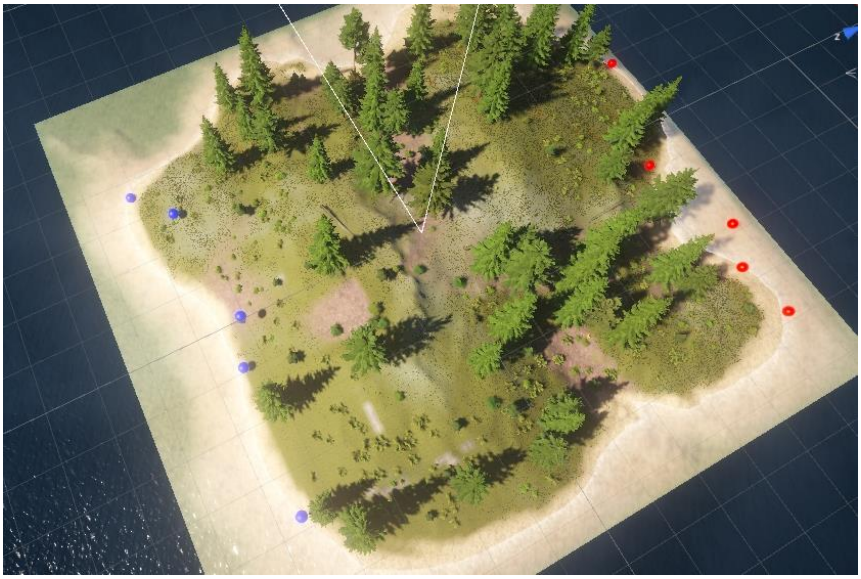

Figura 60: puntos aparición héroes

Uno de los elementos clave ha sido la introducción de vegetación que funciona como puntos de cobertura para los jugadores, permitiendo la planificación de tácticas y estrategias de ocultamiento para sorprender al oponente. Para lograr este objetivo, se ha incluido una gran cantidad de vegetación en el mapa, como árboles y arbustos, que funcionan como puntos de cobertura para los personajes de los jugadores. Asimismo, se ha jugado con una paleta de colores que dificulte la detección de los personajes enemigos a simple vista durante la primera vez que se juega en el mapa.

Figura 61: mapa con mesh obstáculos

Sin embargo, una vez que el jugador haya descubierto la ubicación de los personajes enemigos, se ha implementado un sistema para facilitar su detección. El enemigo se muestra en el minimapa y se le agrega un aura que lo hace más fácil de detectar visualmente en el mapa. De esta manera, una vez que se ha descubierto la ubicación del enemigo, el jugador no tendrá problemas para detectarlo en el futuro.

Figura 62: ayudas visuales con elementos UI del tipo espaciales

Este enfoque busca proporcionar una experiencia de juego equilibrada, en la que, al principio, el jugador tendrá que esforzarse para encontrar a los personajes enemigos, lo que añade un mayor grado de incertidumbre al combate y fomenta la exploración del mapa. Una vez que se han descubierto las ubicaciones, el enfoque cambia a uno más estratégico, en el que el jugador debe adaptarse a las nuevas circunstancias y a las posibilidades de combate que ofrece el mapa.

Además, se ha trabajado en la creación de desniveles en el terreno, situando los puntos de aparición de los héroes a diferentes alturas que la parte central del mapa. Esta disposición permite que al inicio del combate los jugadores no tengan una visión clara de los movimientos del enemigo, añadiendo un mayor grado de incertidumbre al combate y fomentando la exploración del mapa e invitando al jugador a explorar sus posibilidades y a adaptarse a las circunstancias del combate.

Figura 63: vista inicial desnivel desde puntos de aparición

5. Implementación

El presente apartado tiene como objetivo proporcionar información detallada sobre la implementación técnica y los requisitos e instrucciones necesarias para la correcta instalación e implementación de la aplicación desarrollada en el marco del proyecto.

5.1. Implementación técnica

5.1.1. Control del tiempo

Esta capacidad del sistema cliente desempeña un papel fundamental en el juego, ya que proporciona la funcionalidad necesaria para simular eventos en tiempo real en la escena de la Aldea. Su implementación permite que los elementos de la interfaz de usuario respondan de manera dinámica y se actualicen de forma continua, brindando una experiencia más inmersiva y realista para los jugadores.

Una de las características destacadas de este componente es su capacidad para generar eventos de manera periódica a través del uso de suscripciones y eventos. Los diferentes componentes de la interfaz de usuario se suscriben a estos eventos, lo que les permite realizar acciones específicas en función del tiempo transcurrido. Por ejemplo, se actualizan barras de progreso, se gestiona la generación de recursos y se mantienen actualizadas las misiones en tiempo real.

Para lograr esta funcionalidad se ha implementado una instancia única de este componente mediante el uso del patrón de diseño Singleton en el script TimeController. Esto garantiza que solo exista una instancia activa en todo momento, lo que facilita el acceso y el control del tiempo en el juego. Además, proporciona una estructura organizada y eficiente para ejecutar tareas que requieren actualizaciones constantes. Esto evita la necesidad de realizar verificaciones continuas en cada frame y optimiza el rendimiento del juego.

5.1.2. Recursos

Para la generación de recursos, en aparente tiempo real, se implementa un sistema que simula dicha generación continua de recursos en base a las acciones y configuraciones realizadas por el jugador en su aldea. Esta simulación se lleva a cabo de manera interna y se refleja visualmente en el juego mediante eventos periódicos.

Para calcular la cantidad de recursos generados por segundo, se emplean diversas fórmulas en las que las variables son configuradas para cada jugador y su aldea específica. Por ejemplo, al mejorar una mina o modificar el factor de producción, se recalcula la cantidad de recursos por segundo que esa aldea es capaz de generar y se almacena este valor en la base de datos del servidor como la tasa por segundo de cada recurso.

Una vez obtenido el resultado del cálculo y registrado en la base de datos, el juego en Unity simula la generación continua de recursos a través de un evento que se activa cada segundo. Este evento realiza una actualización visual que muestra el incremento de recursos en el equipo del jugador, dando de esta forma una apariencia de actualización en tiempo real.

Cuando el jugador emite una orden o realiza cualquier acción que afecta a los recursos, se lleva a cabo la materialización de los recursos generados desde la última marca de tiempo hasta el momento actual, siguiendo la tasa de generación establecida por segundo. Esta materialización implica la actualización y registro de los recursos disponibles en la aldea correspondiente, lo cual permite verificar si la orden recibida cumple con las reglas del juego y si existen suficientes recursos para su ejecución.

Este sistema presenta la ventaja de liberar al servidor de la tarea de ejecutar un bucle continuo para actualizar los recursos de todos los usuarios. Gracias a la asignación de una cantidad inicial de recursos, una marca de tiempo inicial y una tasa de generación por segundo, es posible calcular los recursos generados en cualquier momento requerido mediante una operación sencilla. Esto evita la necesidad de llevar a cabo actualizaciones recurrentes y permite optimizar el rendimiento del servidor.

5.1.3. Construcción de elementos

La parte de gestión de recursos en el juego está estructurada de manera que el cliente Unity actúa como el frontend, mientras que el servidor web se encarga de almacenar el estado de todos los elementos del juego, como el nivel de construcciones, órdenes de mejoras, investigaciones, misiones, entre otros. Esta arquitectura garantiza la consistencia y la centralización de la información en el servidor.

El almacenamiento del estado interno de los elementos de construcción y otras entidades del juego se realiza en el servidor web. Para gestionar las órdenes y las tareas programadas, se

utiliza un sistema basado en timestamps. Cada orden o tarea tiene un timestamp asociado que indica cuándo debe completarse según las reglas del juego. Utilizando esta información, se calcula el tiempo necesario para llevar a cabo una orden específica.

Durante el período en el que el jugador da órdenes y el tiempo requerido para que se cumplan esas órdenes, el cliente actúa como un simulador que muestra una barra de progreso al jugador. Esta barra muestra visualmente el avance de la tarea en el juego, pero internamente, el sistema comprueba si el timestamp actual ha alcanzado o superado los timestamps asociados a las órdenes en curso, como órdenes de unidades en movimiento o misiones en progreso.

Es importante destacar que todo el estado del juego, incluyendo las órdenes y misiones iniciadas por los usuarios con sus respectivas fechas de finalización, se almacena en la base de datos del servidor. La información se guarda en una estructura de datos adecuada en una base de datos SQL asegurando la integridad de los datos.

Esta arquitectura garantiza un control centralizado de las acciones y eventos del juego, permitiendo un seguimiento preciso del progreso de las órdenes y misiones, así como la sincronización adecuada entre el cliente y el servidor. Además, al almacenar todos los datos en el servidor, se facilita el mantenimiento, la escalabilidad y la seguridad del juego.

5.1.4. Misiones

Las misiones no sólo pueden ser generadas por el propio jugador si no que otros jugadores pueden crear misiones donde el objetivo es una aldea del jugador. En este contexto, nace la necesidad de tener que informar al usuario cuando otro jugador ha creado una misión que interactúa con él sin que el usuario realice ninguna acción.

Con este fin, se ha implementado una funcionalidad que permite la actualización automática de la interfaz de usuario en segundo plano, sin requerir ninguna acción por parte del jugador. Esta característica se basa en consultar periódicamente al servidor para verificar si se han generado nuevas misiones por parte de otros jugadores.

De este modo, se logra mantener al jugador informado y actualizado en tiempo real sobre eventos relevantes en el juego. La consulta al servidor se realiza en intervalos regulares, lo

que garantiza que el sistema esté constantemente sincronizado con los datos más recientes generados por otros jugadores.
Al recibir los nuevos datos, el sistema puede reaccionar de manera apropiada y presentar al usuario la información relevante de acuerdo con el Lore del juego. Esto se puede llevar a cabo mediante la visualización de notificaciones, avisos u otros elementos en la interfaz de usuario.

Esta funcionalidad ejemplifica cómo, a través de la comunicación con el servidor y la recepción de datos que incluyen únicamente marcas de tiempo, se logra brindar al jugador la impresión de que el juego está ocurriendo en tiempo real proporcionando una experiencia de juego más inmersiva y coherente.

5.1.5. Recursos de texto

Se ha implementado un enfoque eficiente y flexible para gestionar los recursos de texto. Con este fin, se ha creado un archivo JSON estructurado con claves y valores, donde cada texto del juego se almacena bajo una clave única. Esta metodología permite una gestión eficaz de los textos, facilitando su recuperación y manipulación en diferentes contextos del juego.

Una de las ventajas clave de este enfoque es la capacidad de incorporar funcionalidades avanzadas en los textos del juego. Por ejemplo, se ha introducido la funcionalidad de texto enriquecido, que permite la inclusión de formatos y estilos especiales para resaltar ciertos fragmentos de texto.

Además, se ha implementado la funcionalidad de reemplazo de parámetros en los textos. Esto significa que ciertas partes del texto se pueden configurar para que sean dinámicas y se actualicen en tiempo real en función de variables o eventos específicos en el juego. Esto brinda una mayor personalización y adaptabilidad en la presentación de la información al jugador, permitiendo una experiencia más inmersiva y personalizada.

El juego solo se encuentra en inglés, pero gracias a esta funcionalidad es fácilmente traducible a otros idiomas.

5.1.6. Interfaz usuario Aldea

El diseño de la escena de la aldea en este proyecto se ha llevado a cabo utilizando elementos creados sobre un objeto canvas. Se ha empleado un enfoque que hace un uso intensivo de prefabs para reutilizar elementos, lo que permite ahorrar tiempo y esfuerzo al evitar tener que recrear elementos similares desde cero en diferentes partes de la escena.

Un componente clave en el sistema es el script denominado HudManager. Este script desempeña un papel central, ya que está suscrito al evento de recepción de datos nuevos del servidor. Su responsabilidad principal es informar a todos los componentes de la interfaz de usuario sobre la llegada de estos nuevos datos.

Los componentes de la interfaz, que son los scripts que controlan los paneles que el usuario ve en pantalla, se comunican entre sí mediante el uso de eventos. Este enfoque se ha adoptado para lograr un desacoplamiento óptimo y mejorar la eficiencia del sistema. Al emplear eventos para la comunicación, se evita que los componentes dependan directamente unos de otros, lo que permite una mayor flexibilidad y mantenibilidad del código. Además, dado que la interfaz cuenta con numerosos pequeños scripts que se actualizan constantemente, esta estrategia de eventos contribuye a mejorar la eficiencia global del sistema.

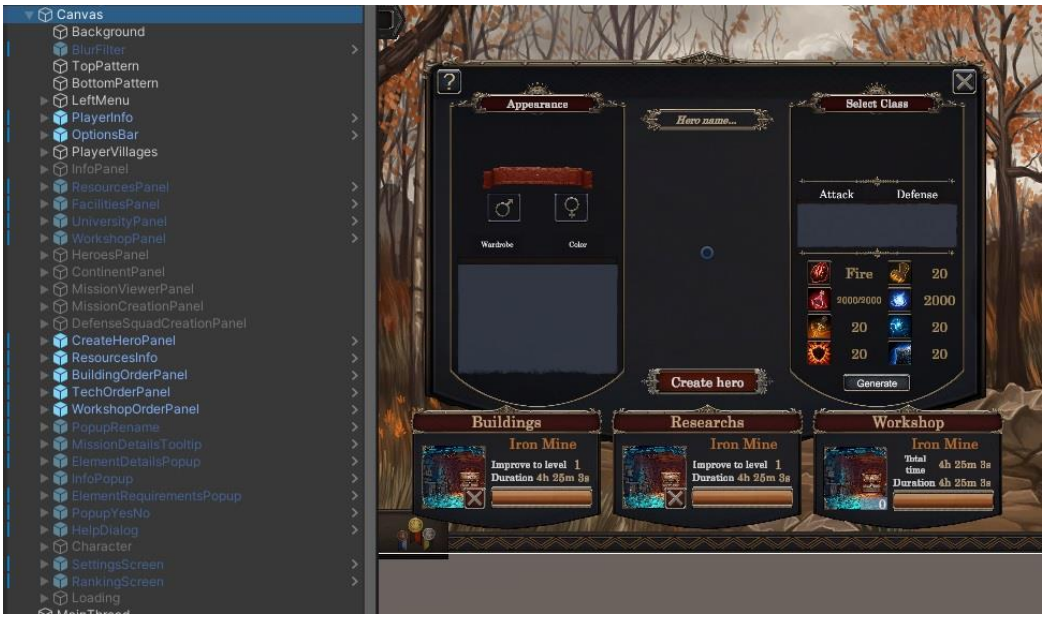

Figura 64: canvas aldea

5.1.7. Creación de personajes

Para permitir la creación de héroes personalizados, se ha implementado un sistema que hace uso del sistema de avatares UMA. Este sistema ha sido utilizado para las razas de Caballeros y Elfos, donde cada una cuenta con animaciones distintas que reflejan sus características y movimientos específicos.

En cuanto a las habilidades, se han diseñado habilidades únicas para cada raza, las cuales incluyen animaciones y efectos personalizados. Esto proporciona una experiencia visualmente atractiva y coherente con el estilo de juego de cada raza.

Con UMA, se ha logrado ofrecer versiones masculinas y femeninas de cada raza, permitiendo a los jugadores elegir el género de su personaje al crearlo. Además, se ha seleccionado ropa adecuada y personalizada para cada raza y género, brindando una amplia variedad de opciones para la apariencia de los personajes.

Para diferenciar claramente las razas, se han preconfigurado opciones de colores específicos para cada una. Esto ayuda a los jugadores a distinguir visualmente las diferentes razas en el juego, tanto en términos de apariencia como de identificación.

Una vez que los jugadores han personalizado su personaje, los datos de personalización se envían al servidor en formato de cadena JSON. Esta cadena de caracteres funciona como una "receta" que contiene todos los detalles de la apariencia y configuración del personaje. Al ingresar a la escena de combate, el servidor utiliza esta receta para construir y mostrar correctamente el avatar personalizado del jugador.

El sistema de avatares UMA en Unity permite una creación y personalización detallada de personajes en el juego. Utiliza la configuración de componentes y características específicas para cada avatar, permitiendo ajustes en tiempo real para lograr una experiencia de personalización fluida y visualmente atractiva.

Toda la configuración comentada para el sistema se ha llevado a cabo mediante Scriptable Objects. Se han creado objetos para representar cada raza en el juego, con sus propias

características y atributos distintivos. Estos objetos contienen la información necesaria para definir las habilidades, animaciones, efectos visuales y sonidos asociados a cada raza.

5.1.8. Combate

En la escena de combate del juego, se implementa un sistema donde por cada héroe vivo se crea un personaje que se vuelve controlable al ser seleccionado. Este sistema permite controlar el flujo del juego y gestionar las acciones que los personajes pueden realizar, siguiendo las reglas establecidas por el juego.

El combate funciona como una interfaz visual para el jugador, mostrando las interacciones y acciones que realiza en tiempo real. Sin embargo, estas acciones son enviadas al servidor para su cómputo y gestión, y se respaldan en la base de datos del juego.

Cada personaje en el combate tiene distintos componentes que se activan o desactivan según si pertenecen al jugador activo o son personajes rivales. Por ejemplo, se utilizan partículas para mostrar la posición del personaje aliado o componentes de navegación por IA para acciones de combate cuerpo a cuerpo que requieren que el personaje se posicione a una distancia específica del objetivo.

Cada acción y habilidad de los personajes, específicas de su raza, están asociadas a un objeto de tipo Scriptable Object que define su comportamiento. Estos Scriptable Objects contienen la información necesaria para ejecutar las acciones de manera adecuada, como los pasos a seguir, las animaciones correspondientes, los sonidos asociados y los efectos de partículas.

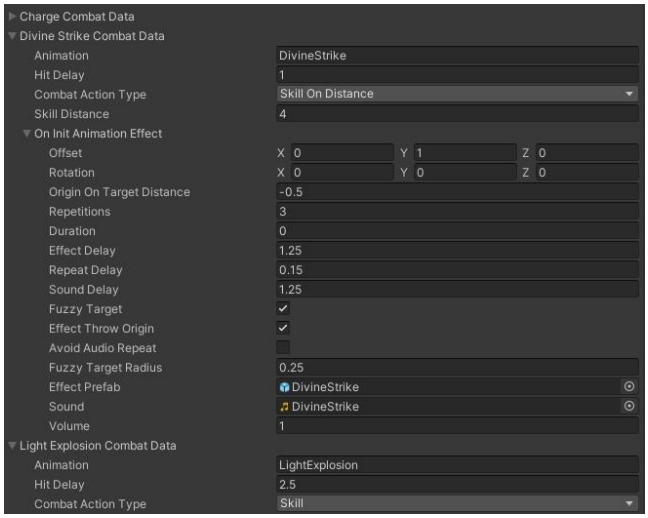

Figura 65: modelo datos habilidades

Cuando el jugador selecciona una acción o habilidad, el modelo de datos asociado se envía al sistema encargado de llevar a cabo dicha acción. Este sistema ejecuta los pasos necesarios para completar la acción, como el desplazamiento del personaje mediante IA hacia el objetivo, la reproducción de animaciones para el lanzamiento o activación de la habilidad, la reproducción de sonidos y la visualización de efectos de partículas. Cada uno de estos elementos puede repetirse según la configuración, como en el caso de efectos de partículas o sonidos que se repiten durante la acción.

5.1.9. Sonido

La sonorización en un juego desempeña un papel crucial para mejorar la experiencia del jugador (Erich Cárdenas, 2009) y se ha prestado especial atención a la implementación de efectos de sonido y música adecuada en diversas áreas.

En primer lugar, se han añadido efectos de sonido a todos los botones presentes en las diferentes escenas del juego. Estos efectos se activan tanto al pasar el cursor sobre ellos como al pulsarlos, lo que proporciona un feedback inmediato al jugador y mejora la interacción con el juego.

Además, se han introducido sonidos específicos para acciones importantes, como el botón de envío de misiones o la creación y cancelación de mejoras. Estos sonidos realzan la importancia de dichas acciones y generan una sensación de logro o consecuencia.

En cuanto a la música, se ha incluido música ambiental en la pantalla de la aldea, la cual se ha seleccionado cuidadosamente para adecuarse al tipo de juego y crear una atmósfera envolvente. Por otro lado, en la pantalla de combate, se ha decidido prescindir de música y dejar el sonido ambiente propio del escenario. Esto contribuye a crear una experiencia más realista y permite que los efectos sonoros de las acciones tomen protagonismo.

El control del sonido se ha implementado utilizando un Audio Mixer, que permite ajustar los niveles de volumen tanto de la música como de los efectos de sonido de manera global. Además, en las opciones de configuración del juego el jugador puede personalizar sus preferencias de sonido según sus necesidades. Estas configuraciones se guardan entre partidas, asegurando que el jugador pueda mantener sus ajustes personalizados.

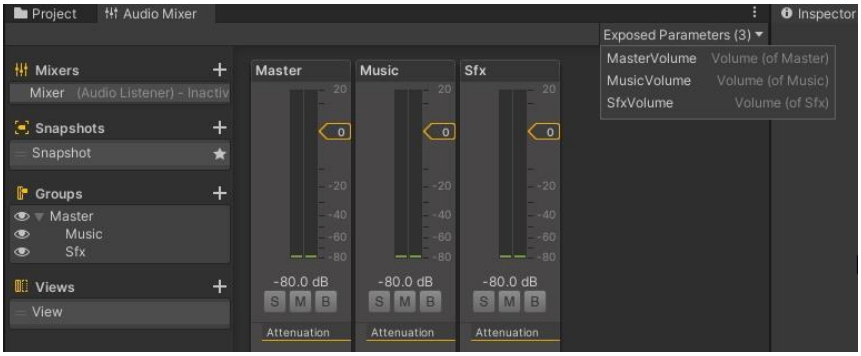

Figura 66: audiomixer

5.2. Requisitos de instalación

Para garantizar un despliegue exitoso tanto en el entorno del cliente como en el servidor, es crucial contar con una comprensión clara de los recursos necesarios, así como seguir los pasos adecuados durante el proceso de instalación.

5.2.1. Requisitos de Software

- Cliente

La aplicación cliente está diseñada para funcionar en el sistema operativo Windows 10 64 bits o compatible. Además, se ha implementado la capacidad de adaptar las exigencias gráficas para garantizar la compatibilidad con diferentes tipos de unidades de procesamiento gráfico (GPUs) de escritorio. Esto permite a los usuarios ajustar la calidad visual según las especificaciones de su GPU, lo que contribuye a una experiencia de juego óptima.

- Servidor

El software del servidor es multiplataforma y puede ejecutarse en cualquier sistema compatible con .NET 7. Esto proporciona flexibilidad en la elección del sistema operativo para el despliegue. Ya sea en la nube o en un entorno local, se requiere un servidor compatible con .NET 7 para garantizar un rendimiento adecuado y una funcionalidad completa del servidor. Durante la fase de desarrollo el servidor ha sido desplegado como una aplicación 32 bits en Azure sobre el servidor web IIS de Microsoft.

5.2.2. Requisitos de Hardware

- Cliente

No se requieren especificaciones de hardware adicionales para el cliente de la aplicación más allá de los requisitos básicos del sistema operativo Windows 10. Sin embargo, para una experiencia de juego óptima, se recomienda contar con un hardware compatible con los estándares mínimos de la plataforma Unity, como un procesador de velocidad adecuada, una cantidad suficiente de memoria RAM y una tarjeta gráfica compatible con los requisitos gráficos deseados.

- Servidor

Los requisitos de hardware del servidor van relacionados con el número máximo de jugadores simultáneos que se quiera poder satisfacer. La capacidad de un App Service en Azure para manejar peticiones simultáneas puede variar dependiendo del plan de App Service seleccionado. En los planes más bajos, como el plan Free utilizado para el desarrollo, la capacidad de procesamiento es limitada y puede soportar un número reducido de peticiones simultáneas. Estos planes están diseñados principalmente para aplicaciones de baja escala o desarrollo y pruebas.
Sin embargo, existe la posibilidad de escalar verticalmente un App Service en Azure y de esta forma es posible mejorar su capacidad de procesamiento y aumentar el número de peticiones simultáneas que puede manejar.

Por otro lado, en cuanto a la escalabilidad horizontal, es importante mencionar que la arquitectura del servidor no está diseñada para admitir este tipo de escalado, pero el juego si está diseñado en distintos universos que cada uno podría estar ejecutándose en un App Service distinto.

5.2.3. Formación/Conocimientos

- Cliente

No se requieren conocimientos o formación específica para utilizar el cliente de la aplicación destinado a los jugadores. Los usuarios solo necesitan descargar el archivo ejecutable (.exe) y ejecutarlo en su sistema. Luego, deberán crear una cuenta utilizando una dirección de correo electrónico y una contraseña para acceder al juego. No se necesita ninguna otra formación adicional.

- Servidor

Dado que la instalación y configuración del servidor requieren un nivel de experiencia más avanzado, se recomienda que un experto en administración de sistemas se encargue de esta tarea. Los conocimientos necesarios para el despliegue y mantenimiento del servidor incluyen la configuración de entornos de nube, la gestión de servidores compatibles con .NET 7 y la administración de bases de datos SQL Server.

5.3. Instrucciones de instalación

A continuación, se detallan los pasos para la instalación de la aplicación, abordando específicamente la creación de una base de datos en SQL Server Azure utilizando migraciones de Entity Framework y el proceso de despliegue desde Visual Studio a un App Service en Azure.

5.3.1. Base de datos

Creación de una base de datos en SQL Server Azure usando migraciones de Entity Framework:

1. Es necesaria una suscripción de Azure y una cuenta de SQL Server Azure válidas.
2. Verificar la cadena de conexión en el archivo de configuración de la base de datos apunte a la instancia de SQL Server Azure.
3. Abrir la consola del Administrador de paquetes en Visual Studio y ejecutar los siguientes comandos:
    a. Enable-Migrations > Habilitar las migraciones en el proyecto.
    b. Update-Database > Aplicar los cambios en la base de datos de SQL Server Azure.

5.3.2. Servidor web

Despliegue desde Visual Studio a un App Service Azure:

1. Es necesario tener acceso a una suscripción de Azure y haber creado un App Service en Azure.
2. Hacer clic derecho en el proyecto de la aplicación y seleccionar "Publicar" para abrir el asistente de publicación.
3. Elegir la opción de implementación en Azure y seleccionar el App Service como destino de implementación.

4. Configurar las opciones de implementación, como el nombre del App Service y el grupo de recursos.

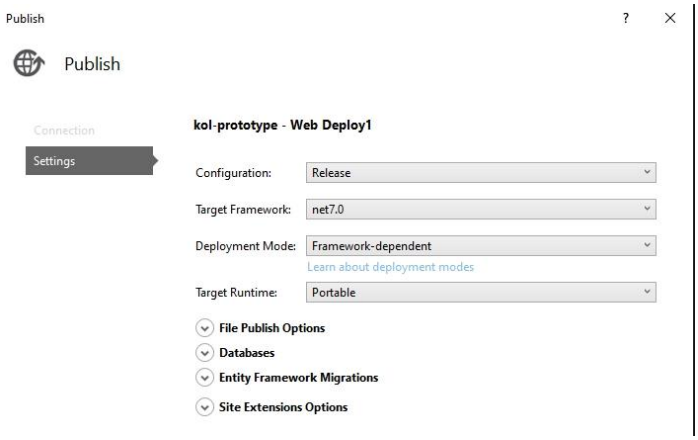

Figura 67: opciones publicación servidor

5. Hacer clic en "Publicar" para iniciar el proceso de implementación y esperar a que se publique.
6. Acceder a la aplicación a través de la URL proporcionada por el App Service en Azure.

6. Demostración

6.1. Prototipos

En este apartado, se presentan los prototipos desarrollados a lo largo del proceso de diseño del proyecto. Los prototipos desempeñan un papel fundamental en la conceptualización, iteración y validación de las ideas y conceptos clave del juego (uxpin.com, s.f).

6.1.1. Prototipos Lo-Fi

Los prototipos Lo-Fi permiten capturar rápidamente ideas y conceptos iniciales, esbozando la estructura y las interacciones básicas del juego. Estos prototipos se centran en la funcionalidad y el flujo de la experiencia del usuario, sin detallar aspectos visuales o gráficos.

Figura 68: prototipo lo-fi vista aldea

Figura 69: prototipo lo-fi vista combate

6.1.2. Prototipos Hi-Fi

Por otro lado, los prototipos Hi-Fi permiten refinar y visualizar de manera más precisa la apariencia visual y la interacción del juego. Estos prototipos incluyen elementos visuales más detallados, como diseños de interfaz de usuario y representaciones visuales de escenas y personajes.

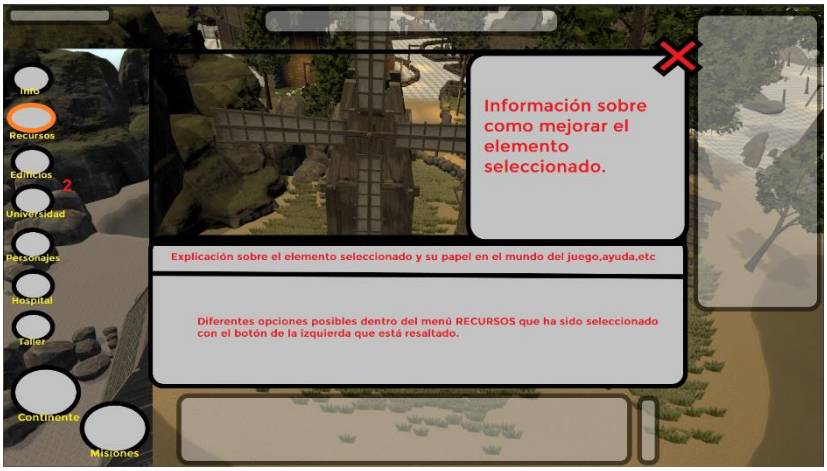

Figura 70: prototipo hi-fi vista aldea

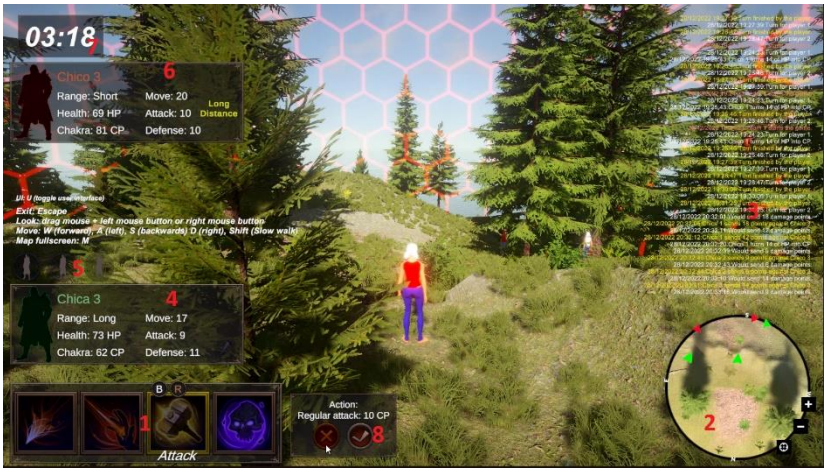

Figura 71: prototipo hi-fi vista combate

6.2. Tests

Durante el proceso de desarrollo, se han llevado a cabo varias pruebas para validar diversos aspectos del juego. Se han realizado pruebas para evaluar el tamaño del escenario de combate y asegurar que se adapta adecuadamente a las necesidades del juego y a la experiencia del jugador. Estas pruebas se centraron en aspectos como la navegación, la exploración y la interacción con el entorno del juego.

Además, se llevaron a cabo pruebas específicas para evaluar la generación de dinámicas esperadas en el combate, especialmente en jugadores nuevos. Estas pruebas permitieron identificar posibles desafíos o problemas relacionados con la curva de aprendizaje, la accesibilidad y la jugabilidad del combate. A partir de los resultados obtenidos, se aplicaron

mejoras tanto en el tamaño como en el diseño del nivel, con el objetivo de proporcionar una experiencia más equilibrada y satisfactoria para los jugadores.

6.3. Ejemplos de uso del producto

6.3.1. Empezar a jugar

1. Abrir la aplicación: El jugador inicia la aplicación del juego.
2. Pantalla de login: En la pantalla de inicio, se muestra el menú de login. Si el jugador no tiene un usuario registrado, debe hacer clic en el botón "Register" para acceder al formulario de registro.
3. Registro de usuario: El jugador completa el formulario de registro proporcionando la información necesaria, como nombre de usuario, contraseña y dirección de correo electrónico.

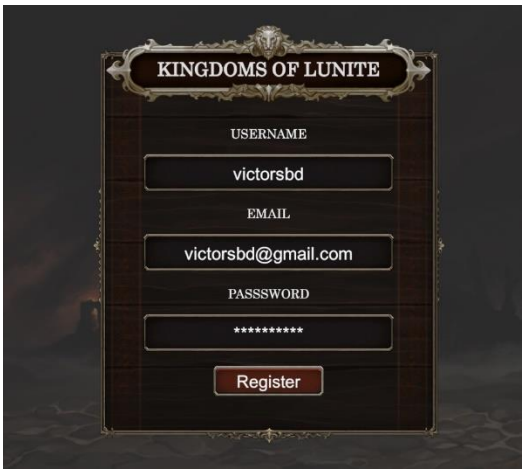

Figura 72: panel registro

4. Iniciar sesión: Una vez que el jugador tiene un usuario registrado, al iniciar el juego debe ingresar su login (email y contraseña) en el menú de login.
5. Autologin: El sistema guarda un token de refresco del login durante 7 días, lo que permite al jugador acceder automáticamente al juego sin tener que ingresar el login cada vez que inicia sesión durante ese período.
6. Escena de la aldea: Después de iniciar sesión, el jugador es dirigido a la escena de la aldea del juego. Esta escena muestra la representación visual de la aldea del jugador a través de los distintos elementos de UI.

7. Visor de recursos: En la parte superior de la pantalla de la escena de la aldea, se encuentra el visor de recursos, que muestra la cantidad de recursos disponibles en la aldea del jugador.

Figura 73: visor recursos

8. Visor de aldeas disponibles: A la derecha de la pantalla de la escena de la aldea, se encuentra el visor de aldeas disponibles. Aquí el jugador puede cambiar entre las diferentes aldeas de las que disponga.

Figura 74: selector aldea

9. Menú principal: A la izquierda de la pantalla de la escena de la aldea, se encuentra el menú principal del juego. El jugador puede abrir los paneles de funcionalidades, como recursos, universidad, misiones, entre otros, a través de este menú.

Figura 75: menú principal

10. Menú de opciones: En la parte inferior de la pantalla de la escena de la aldea, se encuentra el menú de opciones. Aquí el jugador puede encontrar botones para salir del juego, silenciar el audio y abrir la pantalla de opciones, por un lado, mientras que por el otro puede acceder a la vista de los distintos rankings y al sistema de notificaciones, donde dispone de la información de todos los movimientos de tropas, espionajes e informes de combate.

Figura 76: menú opciones

6.3.2. Mejora de elemento

1. Accede al menú correspondiente: Dentro de la aldea, selecciona, por ejemplo, la opción de universidad en el menú principal.

Figura 77: panel universidad

2. Selecciona el elemento: Explora las opciones disponibles y elige el tipo de elemento que deseas construir, como un edificio, investigación o unidad.
3. Verifica los requisitos: Revisa los recursos y condiciones necesarios para mejorar el elemento. Asegúrate de tener los recursos suficientes y cumplir con los requisitos previos, como un nivel mínimo de una construcción.

Figura 78: panel requisitos

4. Confirma la mejora: Una vez que hayas verificado los requisitos, confirma la mejora del elemento.
5. Espera la finalización: El proceso de mejora llevará un tiempo determinado. Durante este período, podrás monitorizar el progreso y visualizar una barra de progreso.

Figura 79: cola construcción unidades

7. Conclusiones y líneas de futuro

7.1. Conclusiones

7.1.1. Lecciones aprendidas

En el desarrollo de este proyecto, se han obtenido diversas conclusiones que han contribuido a enriquecer la experiencia y conocimiento en el ámbito del diseño y programación de videojuegos con Unity del equipo. A continuación, se describen las principales conclusiones derivadas del trabajo realizado:

- Aprendizaje de las mecánicas de desarrollo de videojuegos.

Durante el proyecto, se ha adquirido un profundo conocimiento sobre las diferentes mecánicas que conforman un videojuego, como el diseño de niveles, la implementación y de los distintos sistemas y como se comunican entre sí, la gestión de personajes y la interacción con el entorno. Esta experiencia ha permitido comprender la complejidad y los desafíos involucrados en el desarrollo de un juego completo.

- Importancia de la planificación y organización.

El proyecto ha puesto de manifiesto la necesidad de una planificación adecuada y una organización eficiente para lograr los objetivos establecidos. La creación de un cronograma detallado, la asignación de tareas y el seguimiento del progreso han sido fundamentales para mantener el proyecto en curso y evitar desviaciones significativas.

- Desarrollo de habilidades técnicas.

A lo largo del proyecto, se ha fortalecido y ampliado el conocimiento técnico en el uso de Unity y otras herramientas relacionadas con el desarrollo del servidor web. Se han adquirido habilidades en la programación de scripts, el diseño de niveles, la creación de web api y la resolución de problemas técnicos de alta complejidad como el manejo de la concurrencia en sistemas web. Estas habilidades son transferibles y pueden aplicarse en proyectos futuros de desarrollo de videojuegos e incluso fuera del sector.

- Comprensión de la importancia de la iteración y el feedback.

Durante el desarrollo del proyecto, se ha experimentado la necesidad de realizar iteraciones constantes y recibir feedback de los usuarios o evaluadores. Estas iteraciones han permitido mejorar y optimizar el juego, corregir errores y ajustar las mecánicas para lograr una experiencia de juego más satisfactoria. Se ha aprendido a valorar la retroalimentación como una herramienta fundamental para el crecimiento y la evolución del proyecto.

7.1.2. Reflexión objetivos conseguidos

En el proceso de desarrollo de este proyecto, se plantearon una serie de objetivos iniciales con el fin de guiar y orientar el trabajo realizado. A continuación, se reflexionará críticamente sobre el logro de estos objetivos y se analizarán las razones que han llevado a no alcanzar todos ellos.

En primer lugar, es importante destacar que, si bien se han logrado la mayoría de los objetivos planteados, no se ha conseguido alcanzar la totalidad de ellos. La principal razón detrás de esto ha sido la limitación de tiempo. Durante la fase de planificación del proyecto, se tuvieron en cuenta posibles desviaciones y se estableció un plan de riesgos que contemplaba las funcionalidades que podrían ser descartadas en caso de que surgieran contratiempos. Por lo tanto, aquellos objetivos que no se han alcanzado estaban identificados previamente como posibles descartes en situaciones de desviación temporal.

Sin embargo, es importante destacar que, a pesar de no lograr algunos objetivos iniciales, se han añadido otras funcionalidades que no estaban contempladas en la planificación inicial. Estas incorporaciones surgieron a raíz del feedback recibido por parte del tutor, quien señaló la necesidad de implementar un sistema para ayudar al jugador a comprender mejor el juego. Esta sugerencia se consideró relevante y se decidió incluir en el desarrollo del proyecto, a pesar de no estar inicialmente contemplada. También se ha incluido un sistema de notificaciones que no estaba planeado inicialmente pero se detectó su necesidad gracias a las pruebas realizadas con jugadores.

Además, es importante mencionar que hubo un objetivo específico que no se desarrolló intencionalmente, a pesar de estar marcado como importante y no como posible descarte en el plan de riesgos. Este objetivo se refiere a la funcionalidad de Replay del turno rival. La decisión de no incluir esta funcionalidad fue tomada de manera deliberada, ya que se consideró que iría en contra del diseño del nivel y de las mecánicas del juego, las cuales buscan resaltar las dinámicas estratégicas y mantener la incertidumbre sobre la ubicación del enemigo hasta que se descubra o el enemigo revele su posición.

7.1.3. Análisis seguimiento de la planificación

Es importante destacar que se ha dedicado un considerable esfuerzo para seguir la planificación establecida en el proyecto. Se realizó un minucioso trabajo de planificación con el objetivo de anticiparse a posibles obstáculos y asegurar un desarrollo fluido y eficiente. En gran medida, se ha logrado seguir la planificación en un alto grado, cumpliendo los hitos y plazos establecidos en su mayoría.

En cuanto a la metodología empleada, se considera que ha sido la adecuada para el desarrollo del proyecto. Se optó por una metodología ágil que permitió una mayor flexibilidad y adaptabilidad a medida que se iban identificando nuevas necesidades y oportunidades de mejora. La metodología ágil facilitó la rápida toma de decisiones y la implementación de cambios cuando fue necesario.

En relación con los cambios introducidos a lo largo del proyecto, es importante destacar que estos no fueron realizados con el objetivo de garantizar el éxito del trabajo, sino más bien como resultado de la naturaleza dinámica de cualquier proyecto. A pesar de que se realizó una exhaustiva planificación, siempre existe la posibilidad de que surjan nuevas ideas o se presenten oportunidades que puedan enriquecer el proyecto. Gracias a la metodología ágil empleada, se pudo evaluar rápidamente la viabilidad y aceptación de estas ideas, lo que permitió introducir cambios de manera ágil y eficiente, siempre teniendo en cuenta el impacto en el alcance y los plazos establecidos.

7.2. Líneas de futuro

Es importante recordar que este proyecto nació como un reto personal y no se tuvo en cuenta inicialmente la idea de que pudiera llegar a ser un producto comercial. Mantener la infraestructura necesaria para un juego de este tipo, como un MMO, es costoso y arriesgado sin el respaldo del capital humano y financiero de una empresa. Por tanto, las posibles ampliaciones y mejoras futuras dependerán de la dirección que se decida tomar.

Si se decide apostar por seguir desarrollando el juego como un MMO, se pueden considerar varias mejoras. Por ejemplo, se podría implementar una vista en 3D detrás de la interfaz de usuario en la escena de la aldea, similar a la vista de un juego de estrategia en tiempo real (RTS), para ofrecer una experiencia visual más inmersiva. Además, sería interesante hacer

que los edificios reaccionen a las acciones del jugador, lo que agregaría un nivel de interactividad y realismo al juego. También se podrían introducir nuevas razas y funcionalidades para mejorar la progresión y personalización de los personajes, enfocándose en aspectos de juego de rol (RPG). Además de estas líneas continuistas con el trabajo realizado, con el fin de aumentar las posibilidades de éxito para un juego de este tipo se podrían introducir funcionalidades sociales, como alianzas, amigos y sistema de mensajes.

Otra funcionalidad que no llegó a plantearse inicialmente pero que de cara a una posible evolución a producto comercial podría tener sentido es la inclusión de un modo Arena. La implementación de este modo en Kingdoms of Lunite podría ofrecer varias ventajas que mejoran la jugabilidad, aumentan la participación de los jugadores y fomentan la interacción comunitaria. Al proporcionar una plataforma competitiva para el desarrollo de habilidades, la participación social y contenido adicional para la etapa avanzada del juego, el modo Arena tiene el potencial de enriquecer la experiencia de juego en general y contribuir al éxito a largo plazo del juego.

Por otro lado, una línea alternativa sería aprovechar al máximo el código desarrollado y pivotar el producto hacia un enfoque más accesible para desarrolladores independientes y orientado a un jugador individual, en lugar de un MMO. En este caso, sería necesario analizar qué mecánicas de gestión de recursos y combates tácticos podrían mantenerse, y seguramente se requeriría agregar un componente narrativo más fuerte y un sistema de inteligencia artificial para los combates. Esto permitiría adaptar el juego a un mercado más amplio y ofrecer una experiencia más inmersiva y enriquecedora para los jugadores.

Las líneas de futuro para el trabajo realizado abarcan dos posibles direcciones. Por un lado, se puede seguir desarrollando el juego como un MMO, implementando mejoras visuales, nuevas razas y mejorando las funcionalidades RPG. Por otro lado, se puede pivotar el producto hacia un enfoque más accesible y dirigido a un jugador individual, aprovechando el código existente y agregando componentes narrativos y de inteligencia artificial. La elección dependerá de los recursos disponibles, las perspectivas comerciales y las metas a largo plazo del proyecto.

Bibliografía

Gameforge. (2002). [Videojuego]. **Ogame**.

Lilith Games. (2018). [Videojuego]. **Rise of Kingdoms**.

Supercell. (2012). [Videojuego]. **Clash of Clans**.

Adobe. (2008) **Mixamo**. [Software] https://www.mixamo.com/

UMA Steering Group. (2009) **Unity Multipurpose Avatar**. [Asset]. https://assetstore.unity.com/packages/3d/characters/uma-unity-multipurpose-avatar-35611

Blizzard Entertainment. (2004). **World of Warcraft.** [Videojuego].

Westwood Studios. (1995). **Command & Conquer**. [Videojuego].

Ensemble Studios. (1997). **Age of Empires**. [Videojuego].

Blizzard Entertainment. (1998). **StarCraft**. [Videojuego].

CCP Games. (2003). **EVE Online.** [Videojuego].

Travian Games GmbH. (2004). **Travian.** [Videojuego].

Yee, Nick. (2004). **Unmasking the Avatar**: The Demographics of MMO Player Motivations. https://www.gamedeveloper.com/disciplines/unmasking-the-avatar-the-demographics-of-mmo-player-motivations-in-game-preferences-and-attrition consultado 25/03/2023

InnoGames. (2012). **Tribal Wars 2**. [Videojuego].

Mythos Games. (1993). **X-Com: Ufo Defense.** [Videojuego].

Nintendo. (2017). **Mario + Rabbids Kingdom Battle**. [Videojuego].

Tactical Adventures. (2020). **Solasta: Crown of the Magister.** [Videojuego].

Adell E., Ferran y Casacuberta S., David, **Modelo freemium**. Recursos UOC. http://disseny.recursos.uoc.edu/materials/prod-digital/es/7-3-2-modelo-freemium/ consultado 29/03/2023

Moreno, J. (s.f.). **Estrategia de branding**. Hubspot. https://blog.hubspot.es/marketing/estrategia-branding-elementos-esenciales-marca-solida consultado 25/03/2023

Unity Technologies. (2021). **Unity User Manual**: System Requirements. https://docs.unity.cn/2021.3/Documentation/Manual/system-requirements consultado 19/04/2023

Microsoft. (2022). **Visual Studio 2022 System Requirements**. https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements consultado 19/04/2023

OpenAI. (2021). **DALL·E**. [Software]. https://openai.com/research/dall-e

Atlassian. (2022). **SourceTree**. [Software]. https://www.sourcetreeapp.com/

PONETI. (2022). **Medieval Kingdom UI**. [Asset]. https://assetstore.unity.com/packages/2d/gui/medieval-kingdom-ui-180688

PONETI. (2022). **4000 Fantasy Icons**. [Asset]. https://assetstore.unity.com/packages/2d/gui/icons/4000-fantasy-icons-163280

Sam Schiffer. (2023). **Procedural Circular Health and Progress Bars Pro**. [Asset]. https://assetstore.unity.com/packages/tools/gui/procedural-circular-health-and-progress-bars-pro-187016

Hippo. (2022). **Simple Spinner**. [Asset]. https://assetstore.unity.com/packages/2d/gui/icons/simple-spinner-progress-indicators-for-ui-237500

Lovatto Studio. (2022). **UGUI MiniMap**. [Asset]. https://assetstore.unity.com/packages/tools/gui/ugui-minimap-32874

Gerard Gascón. (2022). **Simple Tools**. [Asset]. https://github.com/GerardGascon/SimpleTools

Hovl Studio. (2023**). AAA Stylized Projectiles Vol.1**. [Asset]. https://assetstore.unity.com/packages/vfx/particles/aaa-stylized-projectiles-vol-1-130378

Moonflower Carnivore. (2018). **Particle Ribbon**. [Asset]. https://assetstore.unity.com/packages/vfx/particles/spells/particle-ribbon-42866

Garrexus. (2021). **Loot Beams**. [Asset]. https://assetstore.unity.com/packages/vfx/loot-beams-162532

Isle of Assets. (2023**). Mobile Force Field**. [Asset]. https://assetstore.unity.com/packages/vfx/shaders/mobile-force-field-226186

Sidearm Studios. (2021). **Ultimate SFX Bundle**. [Asset]. https://www.unrealengine.com/marketplace/en-US/product/ultimate-sfx-bundle

Sidearm Studios. (2021). **Ultimate Music Bundle**. [Asset]. https://www.unrealengine.com/marketplace/en-US/product/ultimate-music-bundle

Alan Dalcastagne. (2021). **Dialog Text Sound Effects**. [Asset]. https://alan-dalcastagne.itch.io/dialog-text-sound-effects

Procedural Worlds. (2023). **Gaia Pro 2021 - Terrain & Scene Generator**. [Asset]. https://assetstore.unity.com/packages/tools/terrain/gaia-pro-2021-terrain-scene-generator-193476

Arteria3D. (2021). **UMA assets**. [Asset]. https://arteria3d.myshopify.com/

Demigiant. (2022). **DOTween**. [Asset]. https://assetstore.unity.com/packages/tools/animation/dotween-hotween-v2-27676

Proyecto26. (2021). **RestClient**. [Asset]. https://github.com/proyecto26/RestClient

Axuno. (2019). **SmartFormat**. [Asset]. https://github.com/axuno/SmartFormat

Day, J. (2022). **Comprehensive Guide to CQRS Data Pattern**. Computer.org. https://www.computer.org/publications/tech-news/trends/comprehensive-guide-to-cqrs-data-pattern consultado 24/04/2023

Microsoft. (2023). **Design the infrastructure persistence layer**. learn.microsoft.com. https://learn.microsoft.com/en-us/dotnet/architecture/microservices/microservice-ddd-cqrs-patterns/infrastructure-persistence-layer-design consultado 24/04/2023

Microsoft. (2023). **Caching in .NET**. learn.microsoft.com. https://learn.microsoft.com/en-us/dotnet/core/extensions/caching 25/04/2023

Bigelow, Stephen. (2022). **What is data separation and why is it important in the cloud?** techtarget.com. https://www.techtarget.com/searchcloudcomputing/answer/What-is-data-separation-and-why-is-it-important-in-the-cloud 26/04/2023

kryptonsolid. (s.f). **¿Qué es la desnormalización y cómo funciona?**. kryptonsolid.com. https://kryptonsolid.com/que-es-la-desnormalizacion-y-como-funciona/ consultado 27/03/2023

Microsoft. (2022). **Role-based authorization in ASP.NET Core**. learn.microsoft.com. https://learn.microsoft.com/en-us/aspnet/core/security/authorization/roles?view=aspnetcore-7.0 consultado 15/03/2023

Craig Stern. (2013). **12 ways to improve turn-based RPG combat systems**. gamedeveloper.com. https://www.gamedeveloper.com/design/12-ways-to-improve-turn-based-rpg-combat-systems consultado 02/03/2023

Erich Cárdenas. (2009). **La importancia del sonido en los videojuegos**. levelup.com. https://www.levelup.com/articulos/161305/La-importancia-del-sonido-en-los-videojuegos consultado 23/05/2023

uxpin. (s.f). **Video Game Prototyping – How To Do It and Why You Should!**. uxpin.com. https://www.uxpin.com/studio/blog/why-and-how-to-use-video-game-prototyping/ consultado 05/03/2023

Anexos

Anexo A: Glosario

Agile: enfoque de desarrollo de software que se basa en la flexibilidad, la colaboración y la adaptación continua. Se centra en la entrega de valor de manera rápida y frecuente, a través de iteraciones cortas.

API (Interfaz de Programación de Aplicaciones): conjunto de reglas y protocolos que permiten a distintos programas y aplicaciones comunicarse e intercambiar datos entre sí de manera eficiente y estandarizada.

ASP.NET Core: framework de desarrollo de aplicaciones web de código abierto y multiplataforma, desarrollado por Microsoft. Proporciona un conjunto de herramientas y librerías para construir aplicaciones web eficientes, escalables y seguras.

Audio Mixer: herramienta que permite controlar y mezclar varios canales de audio de manera simultánea. Proporciona funcionalidades para ajustar el volumen, la panorámica, los efectos y otras propiedades de cada canal de audio, permitiendo crear una mezcla balanceada y coherente de los diferentes elementos sonoros del juego.

Backend: parte de un sistema de software que se encarga de procesar y gestionar la lógica y los datos detrás de escena. Es responsable de manejar las solicitudes del usuario, procesar la lógica empresarial, acceder y manipular la base de datos, y proporcionar los datos necesarios para la interfaz de usuario.

Backlog: lista de tareas, funcionalidades o requisitos pendientes que aún no han sido abordados en el proyecto. Se utiliza para mantener un registro organizado de las tareas pendientes y proporciona una visión general de las funcionalidades o mejoras que se deben implementar en el futuro.

Canvas: es un componente gráfico en el que se representan y se organizan los elementos visuales de una interfaz de usuario. Actúa como una superficie en la que se colocan los elementos gráficos.

Combate por turnos táctico: sistema de batalla en los videojuegos donde los jugadores toman turnos alternados para realizar acciones estratégicas en un campo de batalla. Se caracteriza por la planificación cuidadosa de cada movimiento, la consideración del terreno y la posición de los personajes, y la utilización de habilidades y tácticas específicas para derrotar al oponente.

CQRS (Command Query Responsibility Segregation): patrón de arquitectura de software que separa las operaciones de escritura (comandos) de las operaciones de lectura (consultas) en una aplicación.

CRUD: acrónimo que representa las operaciones básicas de persistencia de datos: Create (crear), Read (leer), Update (actualizar) y Delete (eliminar).

DLL (Dynamic Link Library): archivos compartidos que contienen código y datos que pueden ser utilizados por múltiples programas o aplicaciones. Estas bibliotecas dinámicas se cargan en tiempo de ejecución y permiten la reutilización de funciones y recursos en diferentes aplicaciones.

Entity Framework: ORM de Microsoft. Entity Framework simplifica el acceso y manipulación de datos, al tiempo que ofrece características como el seguimiento de cambios, consultas LINQ, generación de código y migraciones de base de datos, facilitando el desarrollo de aplicaciones escalables y mantenibles.

Freemium: modelo de negocio utilizado en la industria de los videojuegos y aplicaciones móviles, en el cual el producto base se ofrece de forma gratuita, pero se incluyen características o contenido adicional que requieren un pago para acceder a ellos.

Free-to-play: modelo de negocio en la industria de los videojuegos donde los jugadores pueden descargar y jugar un juego de forma gratuita, pero se les ofrece la opción de realizar compras dentro del juego para obtener ventajas o contenido adicional.

Frontend: parte de un sistema de software que interactúa directamente con el usuario. Es la interfaz visible y accesible para el usuario final, a través de la cual se presenta la información y se reciben las interacciones.

Hangfire: biblioteca de programación que permite la programación y ejecución de tareas en segundo plano de manera sencilla y confiable. Proporciona un mecanismo para programar y administrar trabajos diferidos, permitiendo ejecutar tareas de manera asíncrona en un entorno controlado y supervisado.

IDE (Integrated Development Environment): herramienta de software que proporciona un entorno integrado para el desarrollo, prueba y depuración de aplicaciones. Combina un editor de código, herramientas de compilación, depuración y gestión de proyectos en una sola interfaz.

JSON (JavaScript Object Notation): formato ligero de intercambio de datos que se utiliza comúnmente en aplicaciones web y servicios web.

Lore: se refiere al conjunto de historias, mitos, leyendas y conocimiento que se desarrolla en un mundo o universo creado para un videojuego, película, libro u otra forma de narrativa. Es la base narrativa que proporciona contexto, trasfondo y coherencia a la historia y a los elementos presentes en el universo ficticio.

Microtransacciones: transacciones de bajo valor realizadas dentro de un videojuego o aplicación, en las cuales los usuarios pueden adquirir contenido virtual, mejoras o elementos estéticos utilizando dinero real o una moneda virtual del juego.

Minimapa: representación gráfica reducida de un área o entorno más grande dentro de un videojuego. Proporciona una referencia rápida para que los jugadores puedan orientarse y planificar sus acciones.

MMO (Massively Multiplayer Online): acrónimo que hace referencia a los videojuegos en línea de gran escala, donde un gran número de jugadores pueden interactuar simultáneamente en un mundo virtual compartido.

MUD (Multi-User Dungeon): tipo de juego en línea basado en texto que permite a múltiples jugadores interactuar en un mundo virtual compartido.

MVP (Minimum Viable Product): versión mínima y funcional de un producto que se desarrolla con el propósito de validar su viabilidad y recopilar retroalimentación temprana de los usuarios.

NET 7: versión más reciente de la plataforma de desarrollo de software de Microsoft, que proporciona un entorno para crear y ejecutar aplicaciones en diversos sistemas operativos y dispositivos.

ORM (Object-Relational Mapping): técnica que permite mapear objetos de una aplicación orientada a objetos a tablas en una base de datos relacional y viceversa. Proporciona una capa de abstracción que simplifica la interacción con la base de datos, permitiendo a los desarrolladores utilizar código orientado a objetos en lugar de consultas SQL.

Patrón de Unidad de Trabajo: patrón de diseño que se utiliza para gestionar las operaciones de escritura en un repositorio. Proporciona una abstracción de alto nivel que agrupa múltiples operaciones de escritura en una única transacción coherente.

Patrón Repositorio: patrón de diseño que se utiliza para abstraer y encapsular la lógica de acceso a datos en una capa separada de la lógica de negocio.

Prefab: los prefabs son elementos o entidades predefinidas que contienen una configuración y estructura específica. Un prefab es una instancia clonable de un objeto que puede ser reutilizado en múltiples ocasiones en una escena o en diferentes escenas del juego.

Redes neuronales: modelos computacionales inspirados en el funcionamiento del cerebro humano, compuestos por un conjunto de neuronas interconectadas. Estas redes son capaces de aprender y reconocer patrones a partir de un conjunto de datos, lo que les permite realizar tareas como clasificación, reconocimiento de imágenes y procesamiento de lenguaje natural.

REST (Representational State Transfer): estilo arquitectónico para el diseño de sistemas distribuidos basados en la comunicación a través del protocolo HTTP.

RESTful (Representational State Transfer): estilo arquitectónico utilizado en el diseño de sistemas web que se basa en principios y estándares para el intercambio de información entre sistemas distribuidos.

RPG (Role-Playing Game): género de videojuego en el que los jugadores asumen el papel de un personaje ficticio y participan en una historia interactiva. Se caracteriza por la toma de decisiones, la personalización del personaje y el desarrollo de habilidades a lo largo del juego.

RTS (Real-Time Strategy): género de videojuegos en el que los jugadores deben tomar decisiones estratégicas en un entorno en tiempo real. En este tipo de juegos, los jugadores controlan y gestionan unidades y recursos y participan en batallas contra enemigos.

Scriptable Objects: característica de Unity que permite crear objetos personalizados en el editor de Unity para almacenar y administrar datos. Son utilizados para crear elementos reutilizables y configurables.

SFX: sonidos pregrabados o generados digitalmente que se utilizan en producciones audiovisuales, como películas, programas de televisión y videojuegos, para complementar la acción visual y crear una experiencia auditiva inmersiva.

Singleton: patrón de diseño de software que garantiza la existencia de una única instancia de una clase en todo el sistema. Se utiliza cuando se necesita tener un único punto de acceso a un objeto compartido en múltiples partes del código.

SQL Server: sistema de gestión de bases de datos relacional desarrollado por Microsoft. Es ampliamente utilizado en entornos empresariales y proporciona un conjunto completo de herramientas para administrar, almacenar y consultar datos.

Swagger: herramienta de código abierto utilizada para describir, documentar y probar APIs de forma automatizada. Proporciona una interfaz interactiva que permite explorar y probar las diferentes operaciones.

Sistema de partículas: técnica utilizada en los videojuegos para simular y representar efectos visuales como fuego, humo, explosiones, lluvia, entre otros.

Timestamp: marca de tiempo que se utiliza para registrar y representar el momento específico en el que ocurrió un evento. Es una representación numérica de la fecha y hora en la que se realizó una determinada acción o se generó un dato.

VFX: técnicas utilizadas en la industria del cine, la televisión y los videojuegos para crear imágenes o secuencias que no pueden ser capturadas con métodos convencionales.