

DUNGEON REVENANT

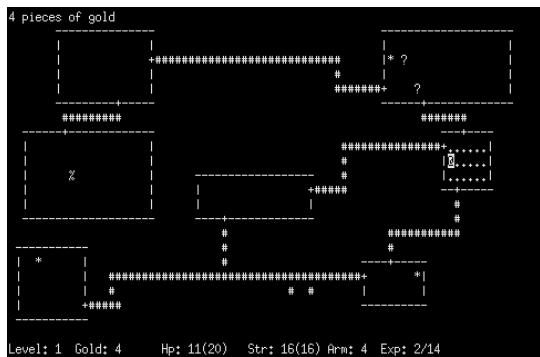


INDICE

INDICE.....	2
1. Descripción del proyecto.....	3
2. Justificación del proyecto.....	5
3. Alcance.....	6
4. Stack tecnológico.....	7
5. Objetivos.....	8
6. Requisitos del sistema.....	8
Requisitos funcionales.....	8
Requisitos no funcionales.....	8
Requisitos de Interfaz.....	8
9. Diseño del prototipo inicial.....	12
10. Diseño del prototipo final.....	13
11. Flujo de navegación.....	16
12. Casos de Uso.....	17
CASO DE USO - 01.....	17
CASO DE USO - 02.....	17
CASO DE USO - 03.....	18
CASO DE USO - 04.....	18
13. BBDD.....	19
14. Muestra de código.....	20
15. Guia de estilos.....	28
16. Manual de usuario.....	29
1. Introducción a la aplicación.....	29
2. Requerimientos.....	29
3. Acceso a la aplicación.....	29
4. Funcionalidad según el Rol.....	30
4.1 Registrarse.....	30
4.2 Jugar.....	30
4.3 Clasificación.....	31
4.3 Mute / Activar sonido.....	31
4.3 Cerrar sesión.....	32
4.3 Usuarios.....	32
17. Manual de configuración.....	34
1. Requisitos del sistema.....	34
2. Instalación y preparación del entorno.....	34
2.1 Acceso al contenedor.....	34
2.2 Encendido del contenedor.....	34
18. Manual de instalación.....	35
1. Requisitos previos.....	35
2. Instalación de la aplicación.....	35
19. Evaluación del resultado final.....	37
20. Posibles mejoras.....	37
21. Conclusión.....	38

1. Descripción del proyecto

La denominación Roguelike proviene del videojuego “Rogue”, el primer juego que se había creado de este estilo en los años 80, realizado con caracteres. Rogue nos ponía en los zapatos de un aventurero que entraba en un calabozo para recuperar el amuleto de Yendor, en el que el personaje podría recoger loot y mejorar sus características. También en los años 80 aparecieron Hack y NetHack, que mejoró la idea que tenía Rogue, añadiendo la idea procedural, tienda, más monstruos y habilidades.



Dungeon Revenant hace combinación de un par de conceptos que reflejan la esencia del juego.

- Dungeon: Hace referencia a las mazmorras que se van a generar, que es el apartado más importante del juego. Cada partida lleva al jugador a explorar un territorio hostil lleno de enemigos, trampas y desafíos.
- Revenant: Hace referencia a un ser que regresa con un propósito. Simboliza al jugador que, al morir, no desaparece por completo. En lugar de un reinicio total, le permite mantener las mejoras adquiridas permanentes, haciendo que el jugador vuelva más fuerte para la siguiente partida.

Dungeon Revenant es un juego roguelike en 2D que pone al jugador frente a mazmorras, donde cada partida ofrece una experiencia distinta. El proyecto se basa en la idea de los juegos roguelike, además de tener enemigos y objetos, lo que asegura que cada vez que se inicie una nueva partida, el jugador se enfrente a un reto distinto. El sistema de muerte permanente es de lo más importante del juego, ya que al morir, el jugador debe comenzar de nuevo. Sin embargo, a lo largo de las partidas, el jugador podrá desbloquear mejoras permanentes que lo ayudarán a avanzar con más facilidad en su siguiente intento, lo que añade un poco de estrategia y de progresión a la jugabilidad.

En este tipo de juegos la variabilidad es clave, ya que nos permite garantizar que los jugadores no se cansen de los mismos escenarios ni enemigos. El juego contendrá muchas mazmorras, serán seleccionadas de forma aleatoria en cada partida, permitiendo que los jugadores sientan una experiencia desafiante, con una jugabilidad que cambia constantemente. La propuesta incluye un sistema de logueo que permitirá a los jugadores

guardar su progreso además de acceder a un ranking online basado en su rendimiento, lo que añade una experiencia competitiva.

Con el proyecto se busca capturar la esencia de algunos de los mejores roguelikes del mercado, pero con un enfoque renovado, en cuanto a variedad de enemigos, jefes y mecánicas. A diferencia de otros juegos del género como *Hades*, *Dead Cells* o *The Binding of Isaac*, Dungeon Revenant no solo se enfocará en la jugabilidad de acción con exploración, sino también en una progresión de mejoras temporales y permanentes. El jugador podrá recolectar “monedas”, puntos o habilidades que le permitirán mejorar su personaje, adquirir nuevos objetos, armas y de esta manera aumentar sus probabilidades de supervivencia en las mazmorras. Algunos objetos se perderán al morir, mientras que otros se conservan para futuras partidas, de esta forma se mantiene un equilibrio entre el tener que arriesgar y el recibir una recompensa por lograrlo.

A parte de la generación de mazmorras, el sistema de combate será otro aspecto fundamental del juego. El jugador tendrá la opción de realizar ataques cuerpo a cuerpo, a distancia, bloquear o esquivar ataques de los enemigos, dependiendo de las armas u objetos que lleve el jugador. Existirá una amplia variedad de enemigos que patrullan, persiguen al jugador o disparan proyectiles, lo que va a hacer que el jugador tenga una exigencia a la hora de adaptar su estilo de combate según la situación. Los jefes, tendrán habilidades especiales y patrones de ataques únicos que presentarán un reto en los niveles más altos.

Otra parte importante del juego será la muerte permanente, un sistema que intenta asegurar que cada vida cuente. Sin embargo, eso no significa que el progreso se elimine, a lo largo de las partidas podrá desbloquear mejoras permanentes que permitirán facilitar las partidas futuras. Teniendo este enfoque permite tener una mezcla entre progreso a largo plazo y desafío constante, lo que hace aumentar la sensación de logro.

El sistema de logeo de usuarios junto a un ranking online permitirán que los jugadores puedan comparar sus trayectorias, pero también crear una competición por posicionarse en posiciones mas altas, lo que genera una competitividad para seguir jugando. Se recibirán premios o recompensas por logros alcanzados, incentivando de esta forma a los jugadores a mejorar sus habilidades.

El mercado de los videojuegos ha demostrado que existe un interés creciente por los títulos roguelike, sobre todo aquellos que ofrecen una buena rejugabilidad y un nivel desafiante. Títulos como *Dead Cells*, *Hades*, *The Binding of Isaac* han ganado un lugar importante dentro de la industria, ofreciendo experiencias que no se olvidan con facilidad. Sin embargo este proyecto, busca ir más allá al incorporar elementos únicos que hacen de cada partida una experiencia totalmente distinta. Mientras algunos se enfocan en aspectos, como pueden ser la narrativa o la fluidez de la acción, Dungeon Revenant se centrará en una experiencia de exploración, un combate desafiante y una progresión estratégica que motive a los jugadores a seguir mejorando.

Gráficamente el juego se va a encontrar basado en sprites 2D, lo que proporciona un estilo visual claro.

El proyecto Dungeon Revenant a futuro tiene potencial suficiente para evolucionar y

expandirse con nuevas actualizaciones, con más enemigos, jefes, armas y mejoras. Además, el feedback de la comunidad será fundamental para el desarrollo y para la mejora de las mecánicas del juego. La posibilidad de crear eventos temporales y nuevos desafíos asegurará que los jugadores sigan motivados para regresar otra vez.

En resumen, Dungeon Revenant se presenta como una propuesta innovadora dentro del género roguelike, la progresión estratégica y una jugabilidad desafiante que mantendrá a los jugadores enganchados. Gracias a un stack tecnológico sólido y a su orientación hacia una comunidad activa, el proyecto promete una experiencia emocionante para los jugadores.

2. Justificación del proyecto.

Los videojuegos roguelike en los últimos años han conseguido ganarse un lugar destacado dentro de la industria de los videojuegos gracias a su combinación de progresión constante, rejugabilidad y desafíos. Títulos como *Dead Cells*, *The Binding of Isaac* han demostrado que existe una audiencia dispuesta a invertir tiempo y esfuerzo en juegos que recompensen la mejora constante y el dominio del juego.

Dungeon Revenant aparece dentro de este género con la intención de aprovechar la fortalezas que lo han popularizado.

- Muerte permanente, genera un reto constante y eleva la satisfacción de completar logros.
- Mazmorras, garantizando partidas únicas y experiencias siempre frescas.
- Progresión permanente, motivando a los jugadores a intentarlo de nuevo, incluso después de fallar.

Son elementos que sirven para brindar una experiencia divertida y emocionante capaz de atraer a tanto jugadores experimentados en el género como a nuevos adeptos del género, aprovechando de buena forma el creciente interés por juegos desafiantes.

Muchos juegos roguelike destacan por una jugabilidad intensa, en este caso Dungeon Revenant apuesta por un enfoque en la progresión estratégica. Esta característica permite al jugador mejorar a lo largo de múltiples partidas, reduciendo la frustración que se pueda llegar a tener en los roguelike al ofrecer un sentido de avance duradero.

El juego combina:

- Un enfoque social y competitivo.
- Variedad de enemigos
- Patrones únicos.
- Sistema de recompensas al completar logros.

Estas innovaciones aportan frescura y complejidad al género, manteniendo a los jugadores motivados a largo plazo.

Dungeon Revenant es una propuesta innovadora dentro del género roguelike que combina la emoción de la exploración, el desafío constante y una progresión estratégica que recompensa el esfuerzo del jugador. Aprovechando el interés creciente por juegos difíciles pero justos, el proyecto se diferencia por su enfoque competitivo y persistente, con un potencial claro de crecimiento en el futuro.

3. Alcance

Generación procedural de mazmorras

- a. Mazmorras generadas aleatoriamente en cada partida.
- b. Diseño variado de habitaciones con trampas y desafíos especiales.

Sistema de combate

- c. Ataques a distancia, cuerpo a cuerpo, habilidades según las armas y objetos obtenidos.
- d. Enemigos con patrones de ataques distintos y comportamientos específicos.
- e. Jefes con habilidades.

Progresión y muerte permanente.

- f. Muerte permanente con reinicio al principio tras cada vez que se pierde.
- g. Objetos y habilidades que se pierden al morir.
- h. Sistema de mejoras permanentes del jugador.

Variedad de objetos y contenido.

- i. Gama de armas, armaduras, pociones, hechizos y objetos consumibles.
- j. Eventos con posibles encuentros aleatorios.
- k. Enemigos variados con comportamientos y habilidades.

Sistema de Logueo y Ranking Online

- l. Creación de cuentas para guardar el progreso.
- m. Ranking competitivo.
- n. Recompensas por cumplir logros.

Estilo y Diseño.

- o. Estilo 2D.
- p. Interfaz intuitiva y clara.

4. Stack tecnológico

El stack tecnológico elegido es:



- **Motor de juego Unity:** Motor de desarrollo de videojuegos ampliamente utilizado en proyectos 2D y 3D, ideal para la creación de roguelikes dinámicos y complejos. Su capacidad para el manejo de físicas y animaciones 2D permite desarrollar un juego visualmente atractivo y fluido. Contiene una amplia comunidad activa que facilita la solución de problemas y la integración de plugins. Además de tener soporte multiplataforma, con su herramientas de editor visual permite ajustar niveles, objetos y enemigos rápidamente.



- **Lenguaje de Programación C#:** Al tener una integración nativa con Unity, facilita el desarrollo rápido y eficiente. Permite una programación orientada a objetos, ideal para controlar la lógica de la generación, los enemigos, el sistema de combate y la progresión del jugador.



- **Base de datos MySQL:** Se encargará de guardar los datos persistentes del jugador, como pueden ser los logros, estadísticas, clasificaciones, progreso y logueo de los usuarios. De esta forma podremos tener una base de datos relacional confiable, nos va a permitir tener consultas complejas para manejar la tabla de puntuaciones o el progreso de los jugadores.



- **Lenguaje PHP:** Se utiliza como intermediario entre Unity y la base de datos. A través de scripts que se utilizarán para operaciones como el registro de usuarios, validación de credenciales y actualización de puntuaciones. PHP nos permite establecer una comunicación segura entre el cliente (Dungeon Revenant) y el servidor (BBDD).



- **Control de versiones Git:** Se utilizará Git junto a su plataforma de alojamiento de código GitHub para el control de versiones.

5. Objetivos

Desarrollo de un juego roguelike 2D, desafiante que contenga elementos de competitividad y progresión para ofrecer una buena experiencia.

- Implementar un sistema de generación de mazmorras para asegurar partidas variadas.
- Diseño de combate dinámico con habilidades, armas cuerpo a cuerpo y a distancia.
- Sistema de logueo y ranking.

6. Requisitos del sistema

Requisitos funcionales

- RF1: Los jugadores pueden crear cuentas, iniciar sesión y guardar su progreso.
- RF2: El jugador se puede mover, atacar, proteger y usar habilidades.
- RF3: Se pueden obtener objetos y habilidades, tanto temporales para la partida como permanentes.
- RF4: Ranking que muestra a los mejores jugadores.
- RF5: Morir reinicia la partida pero conserva las mejoras permanentes.
- RF6: Sistema de recompensas al completar logros.

Requisitos no funcionales

- RNF1: Intento de que el juego funcione a FPS estables.
- RNF2: Interfaz intuitiva.
- RNF3: Código bien documentado y estructurado para facilitar futuras mejoras.
- RNF4: Las contraseñas de los jugadores deben de estar encriptadas.

Requisitos de Interfaz

- RI1: Menú intuitivo con botones claros.
- RI2: Notificaciones al completar logros.
- RI3: HUD mostrando la salud, habilidades, objetos.
- RI4: Interfaz acorde al estilo que se elija del juego.

7. Estudio del juego

Forma	“Dungeon Revenant” es un juego, que busca que el jugador sienta la dificultad de un juego a la vez de que tenga que aprender y preparar estrategias dependiendo de los enemigos. Cada partida reta al jugador a pensar una estrategia, adaptando el estilo de combate según los enemigos, armas y trampas que vaya encontrando. Se apuesta por una ambientación oscura y misteriosa, con unos gráficos en 2D. Con cada intento, el jugador desbloquea mejoras permanentes, haciendo de eso una fácil mejora en el progreso del jugador.
Mecánica	<ul style="list-style-type: none">● Mejoras en el mapa.● Enemigos y jefes.● Eventos.● Mejoras permanentes.● Sistema de logros.● Sistema de ranking.● Bonificaciones y maldiciones.● Distintos tipos de combates.
Contexto	Dungeon Revenant se sitúa en un mundo decadente, en el que han emergido algún tipo de mazmorras. Las mazmorras emergidas como cicatrices en la tierra. El jugador controla a un Revenant, una persona que ha regresado de la muerte. Cada partida representa un nuevo desafío, aunque la muerte es permanente, parte de las mejoras obtenidas permanecen, reforzando la idea de caída y volver a intentarlo.
Arte	Dungeon Revenant adoptará un estilo visual 2D, creando una atmósfera oscura, opresiva y misteriosa. El arte se basará en temáticas oscuras dándole un toque único, utilizando colores sombríos con momentos en los que se buscará resaltar a los enemigos, trampas y objetos con los que se puedan interactuar. La mazmorra podría tener distintos biomas visuales con elementos que refuerzen la sensación de descubrimiento.
Target	A partir de 10 años, exclusivamente para ordenador.

8. Valoración de alternativas existentes en el mercado.

Dentro del género, existen títulos reconocidos que han dejado una marca en la industria de los videojuegos. Dead Cells y The Binding of Isaac son dos de los más exitosos, cada uno con unas características distintivas que le han permitido convertirse en referentes. Con la valoración se busca identificar fortalezas y debilidades para poder colocar a Dungeon Revenant de manera más atractiva y competitiva.



Juego roguelike de acción y plataformas en 2D con un combate dinámico, exploración y una progresión mediante células recogidas al derrotar enemigos. Como es normal en los roguelike al morir se reinicia la partida, teniendo la opción de obtener mejoras permanentes.

En cuanto a sus fortalezas podríamos decir que son:

- Combate fluido y rápido combinando armas y habilidades.
- Contiene un diseño de niveles estilo metroidvania.
- Progresión permanente.
- Variedad de armas y habilidades.

Ahora vamos a tener alguna de sus debilidades:

- Una curva de dificultad frustrante para jugadores novatos.
- Progresión limitada al depender sobre todo de armas y habilidades.



The Binding of Isaac



Roguelike de exploración de mazmorras en 2D con elementos de disparo y un tono oscuro con muchos simbolismo religioso. Cada partida otorga objetos y poderes con efectos combinables.

Algunas de sus fortalezas pueden ser:

- Una gran cantidad de objetos y habilidades con sinergias que fomentan la experimentación.
- Mecánicas simples pero una gran profundidad estratégica.
- Narrativa implícita y oscura que suele dejar a los jugadores intrigados.

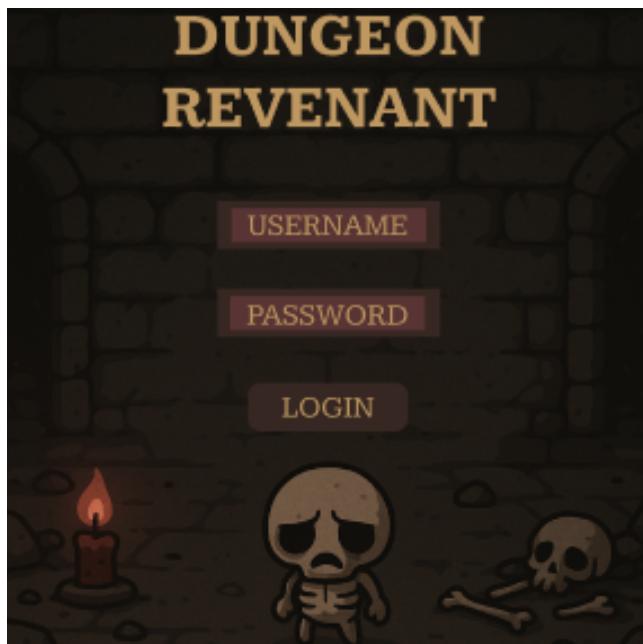
Debilidades de este juego:

- El sistema de progresión depende de la suerte en cada partida.
- La dificultad inicial puede resultar difícil para nuevos jugadores.



9. Diseño del prototipo inicial

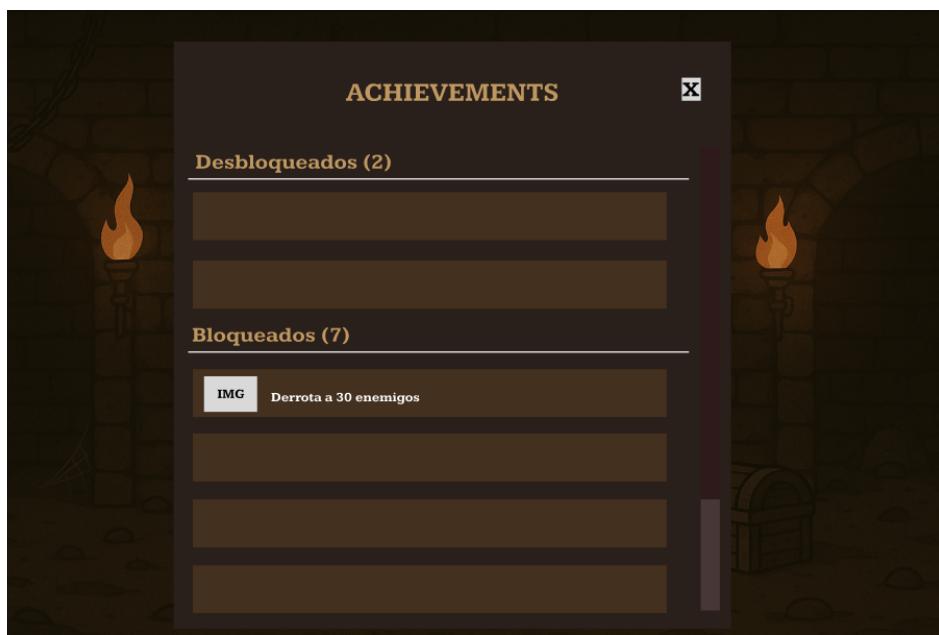
- Pantalla de login



- Pantalla de menú principal

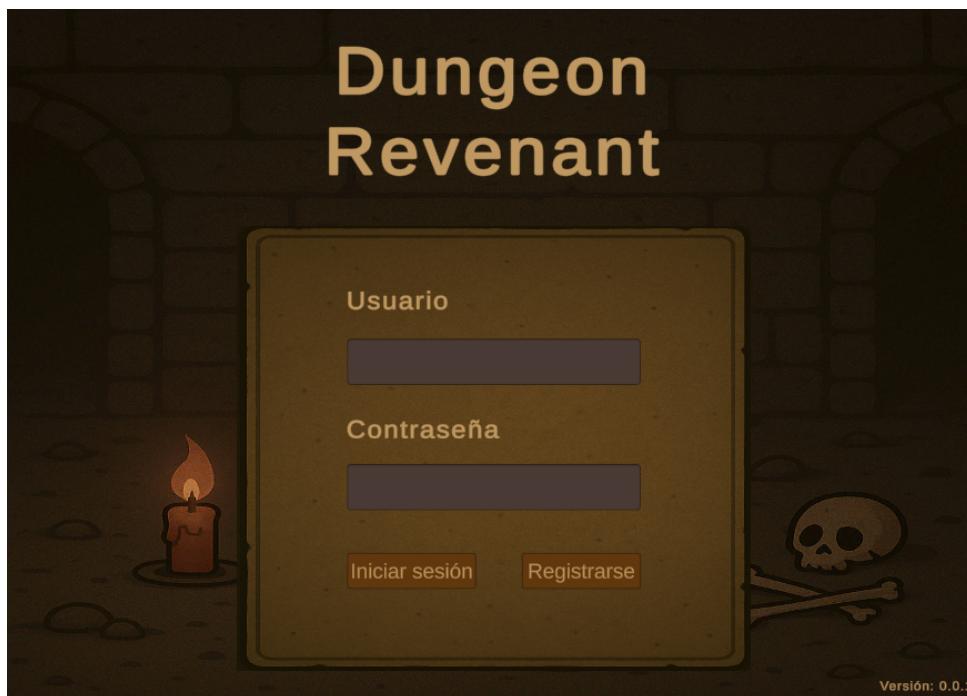


- Pantalla de logros



10. Diseño del prototipo final

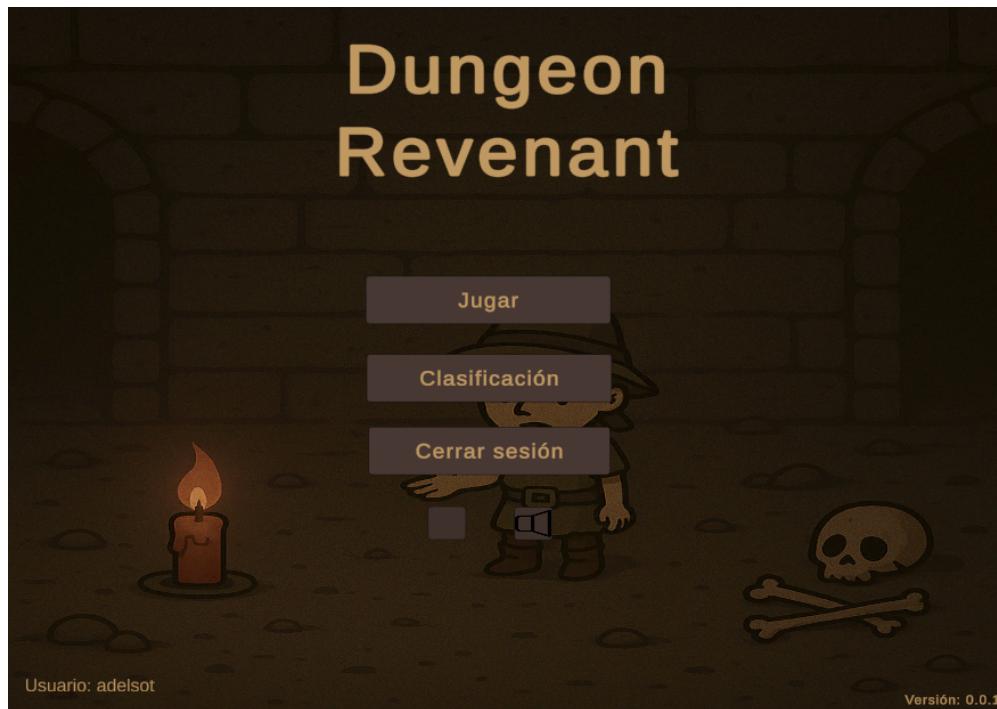
- Pantalla de login



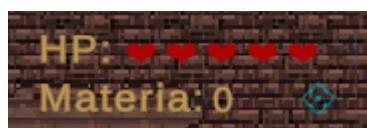
- Pantalla registro



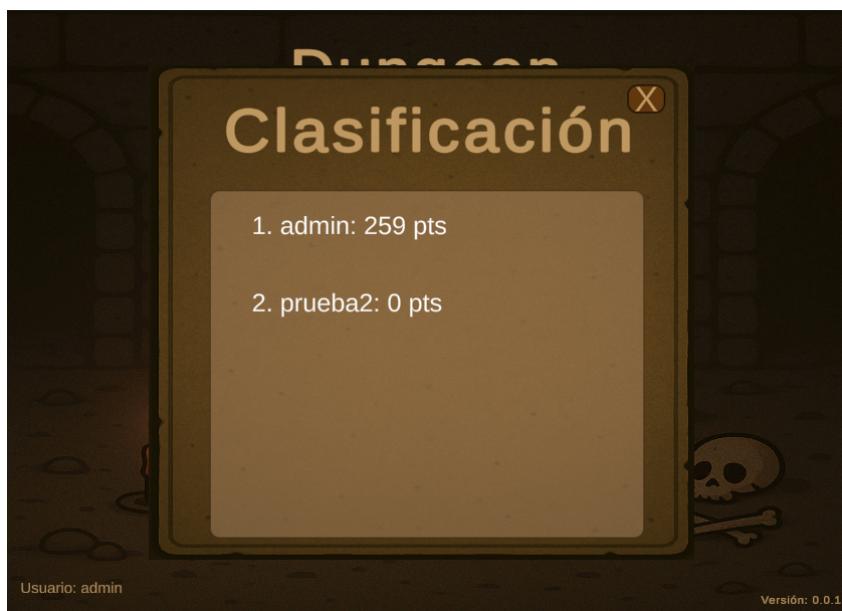
- Menú principal



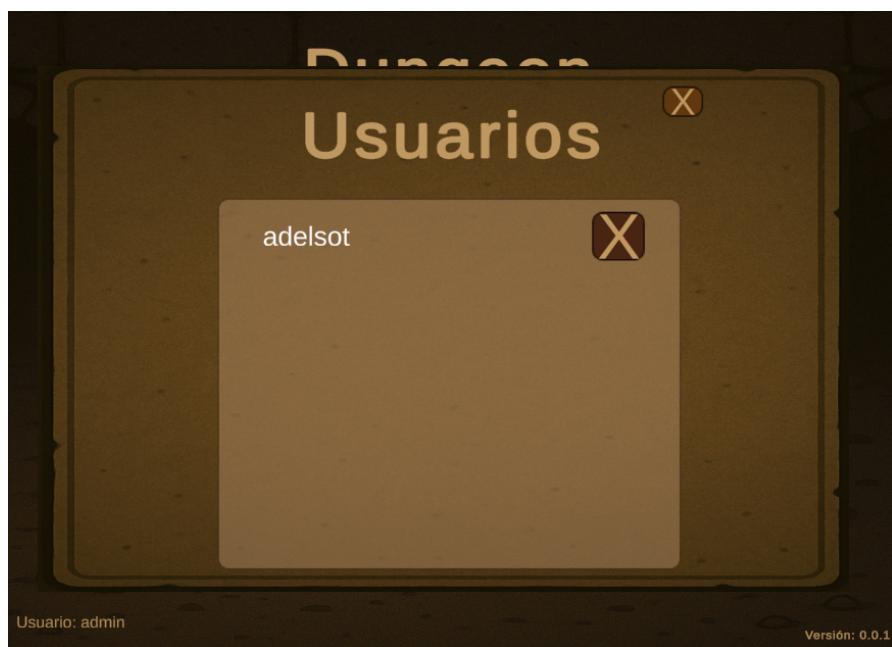
- HUD



- Clasificación



- Usuarios (Admin)



11. Flujo de navegación



- El usuario al principio en la navegación va a tener 2 opciones, iniciar sesión o registrarse, dependiendo de cuál pulse mostrará un recuadro u otro. Una vez haya creado la cuenta e iniciado sesión o directamente iniciado sesión, se le envía al menú principal.
- En el menú principal va a tener varias opciones.
 - Jugar: Para comenzar la partida.
 - Clasificación: Para ver el ranking online de puntuaciones.
 - Cerrar sesión: Botón para cerrar la cuenta.
 - Mute: Botón que activa y silencia la música del menú.
 - Botón admin: Botón que únicamente pueden visualizar los administradores, en el que se activa un panel con una lista de usuarios y tiene los permisos para eliminarlos, el usuario que elimine el administrador de esa lista, también es eliminado en la base de datos.

12. Casos de Uso

CASO DE USO - 01	
Nombre del caso de uso	Iniciar sesión.
Descripción	El jugador escribe su nombre de usuario y contraseña para iniciar sesión en el juego.
Actor	Jugador.
Precondición	Es necesario que el jugador tenga una cuenta creada.
Flujo	<ol style="list-style-type: none">1. El jugador abre el juego e intenta iniciar sesión2. Ingresa su nombre de usuario y clave.3. Se valida si existe en la BBDD ese usuario.4. El jugador accede en caso de ser los datos correctos.
Possible error	En caso de que los datos no sean correctos se muestra un mensaje de error.
Postcondición	El jugador accede al menú principal del juego.

CASO DE USO - 02	
Nombre del caso de uso	Registro.
Descripción	El jugador escribe el nombre de usuario, su correo, la contraseña y confirmación de contraseña. Permitiendo de esta forma crear una cuenta y poder acceder al juego.
Actor	Jugador.
Precondición	Es necesario que los datos del jugador no existan en la BBDD
Flujo	<ol style="list-style-type: none">1. El jugador abre el juego y pulsa el botón registrarse2. Ingresa su nombre de usuario, email y clave.3. Se valida si existe en la BBDD ese usuario y en caso de no existir se crea el usuario.4. El jugador ahora puede iniciar sesión
Possible error	En caso de que existe el usuario, email o la contraseña no sea la misma que la de confirmación, no se le permite crear la cuenta.
Postcondición	El jugador podrá acceder al juego iniciando sesión

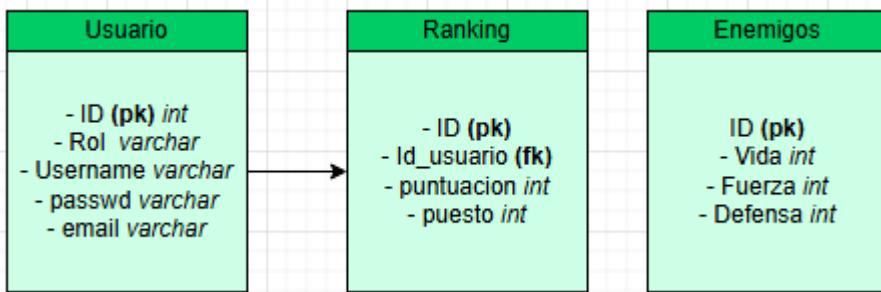
CASO DE USO - 03

Nombre del caso de uso	Combate.
Descripción	El jugador se enfrenta a enemigos usando armas o habilidades.
Actor	Jugador.
Precondición	El jugador se debe encontrar en una partida.
Flujo	<ol style="list-style-type: none"> 1. El jugador ataca a un enemigo. 2. El enemigo reacciona. 3. Si la salud del enemigo llega a 0, muere y deja caer objetos. 4. Si la salud del jugador llega a 0, pierde la partida.
Possible error	En caso de que los datos no sean correctos se muestra un mensaje de error.
Postcondición	Se elimina al enemigo o el jugador pierde la partida.

CASO DE USO - 04

Nombre del caso de uso	Recogida del suelo.
Descripción	El jugador debe poder recoger objetos del suelo dentro de la partida.
Actor	Jugador.
Precondición	El jugador se debe encontrar en una partida.
Flujo	<ol style="list-style-type: none"> 1. El jugador se acerca a un objeto. 2. Pasando por encima lo recoge. 3. El objeto se guarda en un inventario y desaparece del suelo. 4. Se actualiza el HUD.
Possible error	En caso de tener inventario lleno, no se puede recoger.
Postcondición	El objeto se encuentra listo para ser usado.

13. BBDD



```
CREATE DATABASE DungeonRevenantDB;

CREATE TABLE usuarios (
    id_usuario INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    rol VARCHAR(20) NOT NULL,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE
);

CREATE TABLE rankings (
    id_ranking INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    puntuacion INT NOT NULL,
    puesto INT NOT NULL,
    usuario_id INT NOT NULL,
    CONSTRAINT FK_RANKING_USUARIO FOREIGN KEY (usuario_id)
    REFERENCES usuarios(id_usuario)
    ON DELETE CASCADE
);

INSERT INTO usuarios (rol, username, password, email)
VALUES ('admin', 'admin',
'8c6976e5b5410415bde908bd4dee15dfb167a9c873fc4bb8a81f6f2ab448a918',
'admin@admin.com');
```

14. Muestra de código

```
public class EnemigoController : MonoBehaviour
{
    public Transform objetivo;
    public float velocidad = 2f;
    public float rangoDeteccion = 5f;
    public float distanciaAtaque = 1f;

    private Rigidbody2D rb;
    private Animator anim;

    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();
        AsignarObjetivo();
    }

    void FixedUpdate()
    {
        if (objetivo == null) return;

        float distancia = Vector2.Distance(transform.position,
objetivo.position);

        Vector2 direccion = (objetivo.position -
transform.position).normalized;

        if (distancia < rangoDeteccion && distancia > distanciaAtaque)
        {
            // Movimiento hacia el objetivo
            rb.MovePosition(rb.position + direccion * velocidad *
Time.fixedDeltaTime);

            if (anim)
            {
                anim.SetBool("Run", true);
                anim.SetBool("Attack", false);
            }

            // Flip horizontal
            if (direccion.x != 0)
            {
                Vector3 escala = transform.localScale;
                escala.x = Mathf.Sign(direccion.x) * Mathf.Abs(escala.x);
                transform.localScale = escala;
            }
            // Reproducir animación de correr
            if (anim && !anim.GetCurrentAnimatorStateInfo(0).IsName("Enemy
Run"))
            {

```

```

        anim.Play("Enemy Run");
    }
}
else if (distancia <= distanciaAtaque)
{
    // Reproducir animación de ataque
    if (anim && !anim.GetCurrentAnimatorStateInfo(0).IsName("Enemy
Attack 1"))
    {
        anim.Play("Enemy Attack 1");
    }
}
else
{
    // Reproducir animación de idle
    if (anim && !anim.GetCurrentAnimatorStateInfo(0).IsName("Enemy
Idle"))
    {
        anim.Play("Enemy Idle");
    }
}
}

// Metodo especial utlizado para cuando se cambie de escena decirle a
los enemigos que el objetivo es el jugador
void OnEnable()
{
    AsignarObjetivo();
}

// Metodo para asignar el jugador como objetivo a los enemigos
void AsignarObjetivo()
{
    if (objetivo == null)
    {
        GameObject jugador =
GameObject.FindGameObjectWithTag("Player");
        if (jugador != null)
        {
            objetivo = jugador.transform;
        }
    }
}
}

```

```

public class RecogerMateria : MonoBehaviour
{
    private int cantidadMateria;

    public Sprite spriteAzul;
    public Sprite spriteRojo;

    private SpriteRenderer sr;

    void Start()
    {
        sr = GetComponent<SpriteRenderer>();
        // Número aleatorio entre 10 y 80
        cantidadMateria = Random.Range(10, 61);

        // Cambiar sprite segun la cantidad
        if (cantidadMateria >= 10 && cantidadMateria <= 40)
        {
            sr.sprite = spriteAzul;
        } else if (cantidadMateria > 40)
        {
            sr.sprite = spriteRojo;
        }
    }

    // Al tocar el jugador la materia, añadir la puntuación.
    void OnTriggerEnter2D(Collider2D other)
    {
        PlayerController jugador =
        other.GetComponentInParent<PlayerController>();
        if (jugador != null)
        {
            jugador.AñadirMateria(cantidadMateria);
            Debug.Log("Jugador recogió materia: " + cantidadMateria);
            Destroy(gameObject);
        }
    }
}

public class SceneManager : MonoBehaviour
{
    [Header("Login")]
    [SerializeField] private TMP_InputField usuarioLogin = null;
    [SerializeField] private TMP_InputField passwordLogin = null;

    [Header("Registro")]
    [SerializeField] private TMP_InputField usuarioInput = null;
    [SerializeField] private TMP_InputField emailInput = null;
    [SerializeField] private TMP_InputField passwordInput = null;
}

```

```

[SerializeField] private TMP_InputField rePasswordInput = null;

[Header("Mensajes emergentes")]
[SerializeField] private TMP_Text textoMostrar = null;

[Header("UI")]
[SerializeField] private GameObject registerUI = null;
[SerializeField] private GameObject loginUI = null;
[SerializeField] private GameObject menuPrincipalUI = null;

[Header("UI ADMIN")]
[SerializeField] private GameObject botonAdmin;
[SerializeField] private GameObject panelUsuarios;

[Header("Ranking")]
[SerializeField] private GameObject panelRanking;
[SerializeField] private Transform contenidoRanking;
[SerializeField] private GameObject rankingItemPrefab;
[SerializeField] private TMP_Text textoRanking = null;

private NetworkManager networkManager;

private void Awake()
{
    networkManager = GameObject.FindObjectOfType<NetworkManager>();
}

void Start()
{
    // Comprobar si el GameManager quiere mostrar menú principal tras GameOver
    var gameManager = FindObjectOfType<ControladorGame.GameManager>();
    if (gameManager != null && gameManager.volverAlMenu)
    {
        Debug.Log("Volviendo del GameOver: mostrando menú principal");

        gameManager.ResetGame();

        if (loginUI != null) loginUI.SetActive(false);
        if (registerUI != null) registerUI.SetActive(false);
        if (menuPrincipalUI != null) menuPrincipalUI.SetActive(true);

        gameManager.volverAlMenu = false;
    }
    else
    {
        if (loginUI != null) loginUI.SetActive(true);
        if (registerUI != null) registerUI.SetActive(false);
        if (menuPrincipalUI != null) menuPrincipalUI.SetActive(false);
    }
}

```

```

//-----  

//      LOGIN, REGISTRO Y JUGAR  

//-----  
  

public void SubmitLogin()  

{  

    // Control de errores por si no ha rellenado toda la información.  

    if (string.IsNullOrEmpty(usuarioLogin.text) ||  

string.IsNullOrEmpty(passwordLogin.text))  

    {  

        textoMostrar.text = "Todos los campos deben de encontrarse  

rellenos.";  

        return;  

    }  
  

    networkManager.CheckUser(usuarioLogin.text, passwordLogin.text,  

delegate (Response response)  

{  

    if (response != null)  

    {  

        textoMostrar.text = response.mensaje;  
  

        // Cambiar la ventana a Menu principal si coincide.  

        if (response.mensaje.StartsWith("Usuario:"))  

        {  

            // Guardamos el usuario.  

            networkManager.SetUsuarioActual(usuarioLogin.text);  
  

            loginUI.SetActive(false);  

            menuPrincipalUI.SetActive(true);  
  

            // Comprobar si es un administrador  

            if (botonAdmin != null)  

            {  

                botonAdmin.SetActive(response.esAdmin == 1);  

            }  

        }  

    }  

    else  

    {  

        textoMostrar.text = "Error al conectar con el servidor.  

Inténtalo más tarde.";  

        Debug.LogError("La respuesta del servidor fue nula.");  

    }  

});  

}  
  

public void SubmitRegister()  

{  

    // Control de errores por si no ha rellenado toda la información  

    if (string.IsNullOrEmpty(usuarioInput.text) ||  

string.IsNullOrEmpty(emailInput.text) ||  

string.IsNullOrEmpty(passwordInput.text) ||
```

```

string.IsNullOrEmpty(rePasswordInput.text)) {
    textoMostrar.text = "Todos los campos deben de encontrarse
rellenos.";
}
// Control para saber si ha escrito las mismas contraseñas.
if (passwordInput.text.Equals(rePasswordInput.text)) {
    textoMostrar.text = "Procesando...";
    networkManager.CrearUsuario(usuarioInput.text, emailInput.text,
passwordInput.text, delegate (Response response)
{
    if (response != null)
    {
        textoMostrar.text = response.mensaje;
    }
    else
    {
        textoMostrar.text = "Error al conectar con el servidor.
Inténtalo más tarde.";
        Debug.LogError("La respuesta del servidor fue nula.");
    }
});
} else
{
    textoMostrar.text = "Las contraseñas no son iguales.";
}
}

public void ShowLogin()
{
    // Activamos y desactivamos pantallas
    registerUI.SetActive(false);
    loginUI.SetActive(true);
}

public void ShowRegister()
{
    // Activamos y desactivamos pantallas
    registerUI.SetActive(true);
    loginUI.SetActive(false);
}

public void Jugar()
{
    // Cargar la escena del primer mapa
    UnityEngine.SceneManagement.SceneManager.LoadScene("Mapa01");
}

//-----
//      RANKING
//-----


public void MostrarRanking()
{

```

```

        Debug.Log("Intentando mostrar el panel de ranking...");

        panelRanking.SetActive(true);
    }

    private IEnumerator CargarRankingDesdePHP()
    {
        UnityWebRequest www =
UnityWebRequest.Get("http://172.22.229.23/Game/obtenerRanking.php");
        yield return www.SendWebRequest();

        if (www.result != UnityWebRequest.Result.Success)
        {
            textoMostrar.text = "Error al cargar el ranking.";
            Debug.LogError(www.error);
            yield break;
        }
        Debug.Log("Respuesta JSON del servidor: " +
www.downloadHandler.text);
        // Limpiar contenido anterior
        foreach (Transform hijo in contenidoRanking)
            Destroy(hijo.gameObject);

        // Convertimos la respuesta JSON a una lista básica
        RankingSimple[] ranking =
JsonConvert.DeserializeObject<RankingSimple[]>(www.downloadHandler.text);
        Debug.Log("Actualizando textoRanking con: " + textoRanking.name);

        // Comprobar longitud del array
        if (ranking.Length == 0)
        {
            textoRanking.text = "No hay datos en el ranking aún.";
        }
        else
        {
            textoRanking.text = "";
            int puesto = 1;
            foreach (RankingSimple fila in ranking)
            {
                GameObject rankingItem = Instantiate(rankingItemPrefab,
contenidoRanking);
                TMP_Text texto =
rankingItem.GetComponentInChildren<TMP_Text>();
                texto.text = puesto + ". " + fila.usuario + ":" + +
fila.puntuacion + " pts";
                puesto++;
            }
        }
    }

    public void CerrarRanking()
    {
        panelRanking.SetActive(false);
    }
}

```

```
//-----
//      LISTAR USUARIOS
//-----
```

```
public void AbrirPanelUsuarios()
{
    panelUsuarios.SetActive(true);
    var panelScript =
panelUsuarios.GetComponent<PanelUsuariosManager>();
    if (panelScript != null)
    {
        panelScript.AbrirPanel();
    }
}

public void CerrarUsuarios()
{
    panelUsuarios.SetActive(false);
}

//-----
//      Cerrar sesion
//-----
```

```
public void CerrarSesion()
{
    Debug.Log("Cerrando sesión...");

    // Limpiar los campos de login
    usuarioLogin.text = "";
    passwordLogin.text = "";

    // Cambiar a la ventana de login
    menuPrincipalUI.SetActive(false);
    loginUI.SetActive(true);
}
```

15. Guia de estilos

GUIA DE ESTILOS - DUNGEON REVENANT

Logotipo



Colores principales



Colores secundarios



Tipografía

H1 - Títulos - Kameron - 96 px

H2 - Títulos - kameron - 48 px

Texto del cuerpo - Alice - 20 px

Iconografía



Botones



Imagenes



16. Manual de usuario

1. Introducción a la aplicación

La aplicación es un juego roguelike en 2D que pone al jugador frente a mazmorras, donde cada partida ofrece una experiencia distinta. Permitiendo tener un apartado de ranking para poder compararse competitivamente con los demás jugadores. En cambio los administradores son los encargados de supervisar que todo se cumpla correctamente.

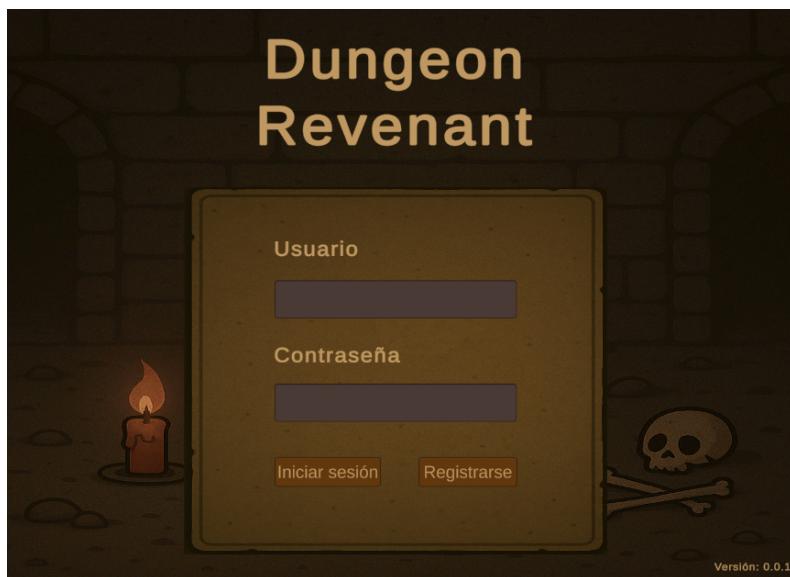
2. Requerimientos

Los requisitos mínimos que debemos tener en cuenta son:

- Sistema operativo Windows.
- Conexión a internet.

3. Acceso a la aplicación

1. Abrir la aplicación.
2. Introducir el nombre de usuario y contraseña.
3. Hacer clic en el botón de “iniciar sesión”.
4. Dependiendo de su rol, el usuario será mandado a una pantalla u otra.



4. Funcionalidad según el Rol

Todos los usuarios → Tienen acceso a la siguiente funcionalidad.

4.1 Registrarse

Si cuando se quiere iniciar sesión no se tiene creada una cuenta, le damos al botón registrarse, nos manda a la siguiente pantalla.

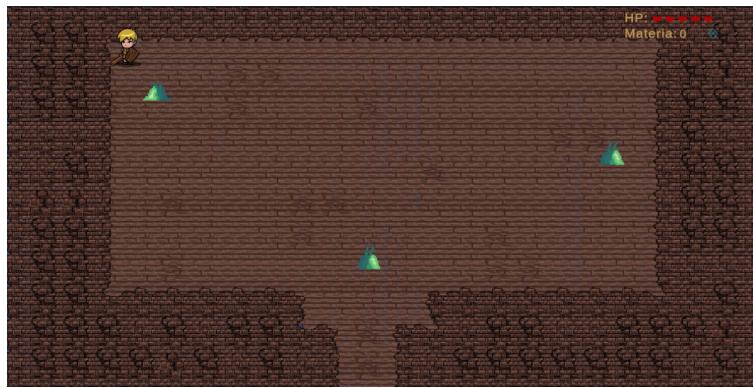


Una vez en la pantalla, rellenamos los datos que nos piden y hacemos clic en “Registrarse”, el sistema comprueba si existe algún usuario con esos datos y si no existe, aparecerá un mensaje de cuenta creada. Ya podemos darle al botón de Iniciar sesión y llenar el formulario con los datos para iniciar sesión.

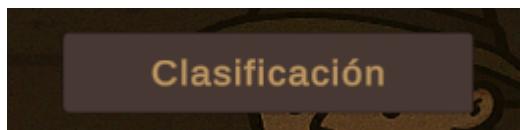
4.2 Jugar



Una vez hemos iniciado sesión nos manda a la pantalla principal, nos aparecen varios botones y uno es el de jugar, si le damos, comienza la partida del juego.



4.3 Clasificación



Si entramos en la opción de clasificación, nos aparecerá un listado de los usuarios que tengan puntuación y en orden del que más tiene a menos.

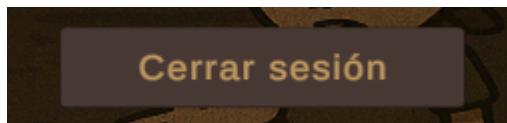


4.3 Mute / Activar sonido



Tenemos la opción de mutear el sonido del juego y activarlo haciendo clic en el botón.

4.3 Cerrar sesión



Para cerrar sesión tenemos el botón que al pulsarlo nos manda al login de nuevo.

Rol administrador → Acceso exclusiva a la siguiente funcionalidad

4.3 Usuarios



A los administradores en el menú principal, les aparece un botón extra con la imagen de un usuario, sirve para mostrar los usuarios registrados en el juego, además los administradores dentro de ese listado tienen la opción de eliminarlos.



5. Preguntas frecuentes

1. ¿Puedo añadir usuarios con el mismo nombre y otros datos?

No, el nombre de usuario y email deben ser únicos para cada cuenta.

2. ¿Puedo modificar el identificador de mi cuenta?

No, los id de los usuarios se realizan de forma auto genérica cada vez que creamos un usuario.

3. ¿Una vez creada mi cuenta puedo cambiar el nombre de usuario?

No, una vez se ha creado un usuario no se puede cambiar el nombre de usuario.

17. Manual de configuración

1. Requisitos del sistema

Cliente

- Sistema operativo: Windows 10 o superior
- Memoria RAM: 8GB mínimo
- Acceso a la red local con permiso para conectarse al servidor

Servidor

- Contenedor LXC Ubuntu
- Base de datos
- Usuario y contraseña configurados para acceso remoto

2. Instalación y preparación del entorno.

2.1 Acceso al contenedor

1. Acceder a la interfaz web de **Proxmox** desde la siguiente dirección:
<https://172.22.0.11:8006>

2. En caso de no encontrarse en la misma red local que el servidor, es necesario conectarse primero mediante **OpenVPN**. Asegúrese de tener instalado el cliente y el archivo de configuración.



2.2 Encendido del contenedor

1. Una vez dentro de la interfaz de Proxmox, localizar el contenedor correspondiente a la base de datos.



2. Encenderlo haciendo clic en el botón "**Start**"
3. Desde el cliente, ejecutar el juego e intentar iniciar sesión como usuario administrador o realizar alguna acción que implique tener la conexión con la base de datos.

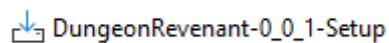
18. Manual de instalación

1. Requisitos previos

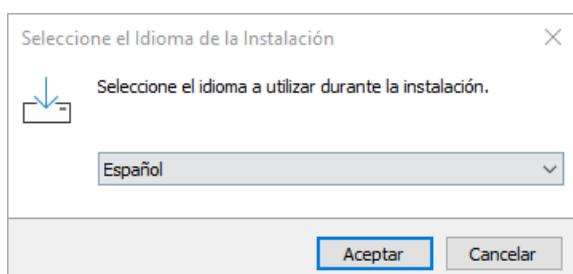
- Tener acceso a la red donde está el servidor o utilizar VPN.
- Contar con permisos de administrador en el equipo por si se necesitan.

2. Instalación de la aplicación

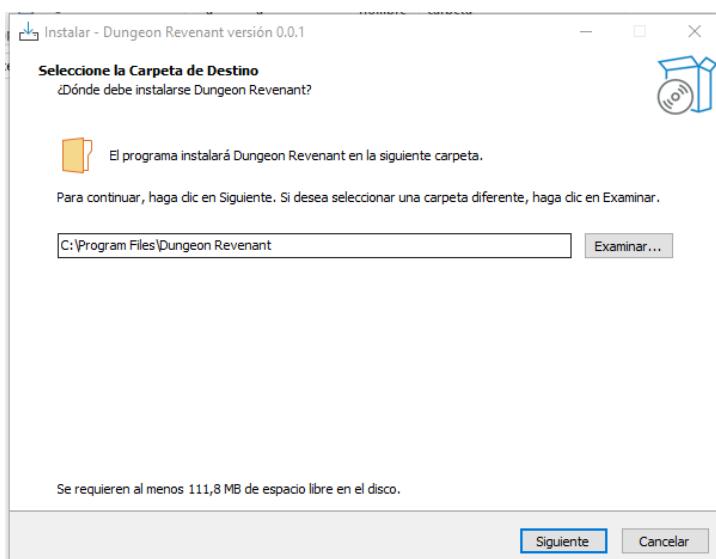
1. Ejecutamos el instalador



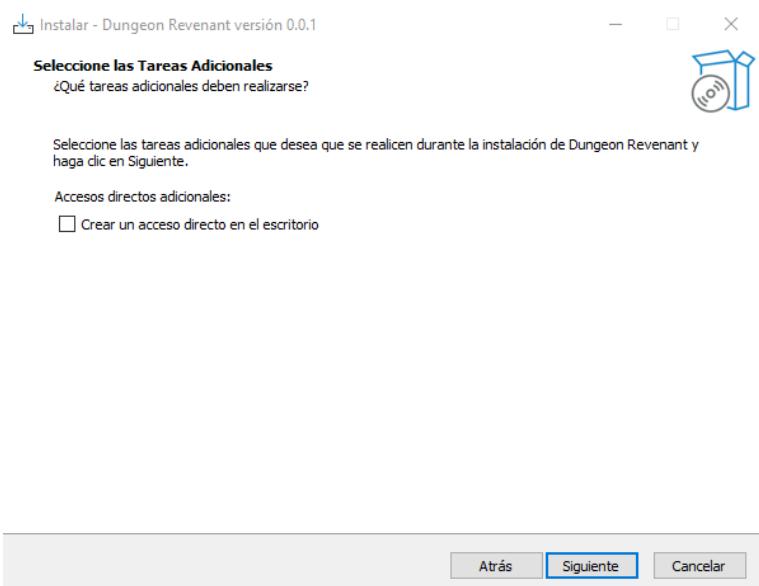
2. Seleccionamos el idioma



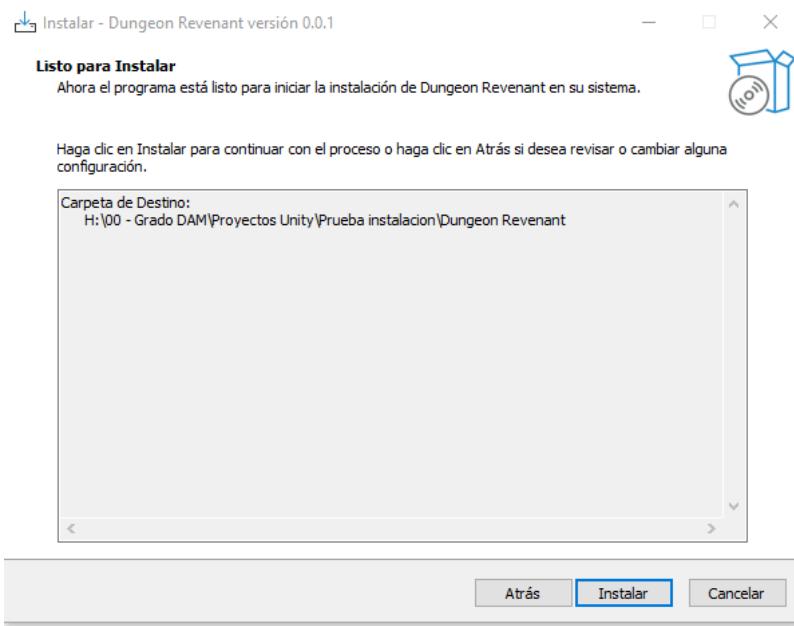
3. Elegimos la carpeta de destino para la instalación.



4. Elegir si queremos tener un acceso directo en el escritorio o no.



5. Por último, instalar.



19. Evaluación del resultado final

El objetivo principal del proyecto se ha logrado, dando como resultado un videojuego 2D roguelike con muerte permanente. Dungeon Revenant permite al jugador registrarse, iniciar sesión, competir a través de un sistema de rankings y enfrentarse a enemigos con diferentes mecánicas de combate. Se han implementado diversos tipos de ataques, enemigos con patrones únicos, así como un sistema de puntuación que mantiene la experiencia de juego desafiante y progresiva. Todo ello se ha logrado manteniendo una estética visual propia usando sprites 2D.

A nivel técnico, el desarrollo ha sido un reto importante, especialmente la parte de código al ser totalmente nuevo, temas como los teletransporte del jugador entre mapas o darle al boss particularidades únicas. Se ha conseguido una estructura de juego sólida y expandible, lo que nos facilita futuras actualizaciones y mejoras. El sistema de logueo es sencilla, lo que facilita una gestión de usuarios rápida.

20. Posibles mejoras

Aunque el proyecto haya alcanzado la mayoría de sus objetivos iniciales, existen varios temas con potencial de mejora que se podrían considerar para futuras versiones del juego.

- Implementación de generación procedural de mazmorras: Actualmente, las mazmorras son seleccionadas entre diseños predefinidos. Un sistema de gestión procedural permitiría crear mapas únicos en cada partida, aumentando la rejugabilidad y acercándose más a los estándares del género roguelike.
- Ampliación de la base de datos con una tabla de enemigos escalables: Ahora mismo los enemigos se encuentran integrados de forma estática. Como mejora, se podría crear una tabla específica en la base de datos para almacenar información detallada de cada enemigo (nombre, nivel recomendado, estadísticas...) De esta forma se nos permitirá escalar la dificultad del enemigo dependiendo de los avances del usuario. Además facilita la implementación de un bestiario, donde los jugadores podrían visualizar qué enemigos han descubierto.
- Mayor variedad de contenido: Ampliar el número de enemigos, armas, objetos y añadir habilidades podría ayudar a mantener el interés a largo plazo.
- Mejora en los enemigos: Aumentar la complejidad de las reacciones de los enemigos para que tuvieran reacciones más inteligentes o patrones más adaptados.
- Sistema de logros y recompensas: Añadir logros desbloqueables al completar ciertos desafíos, y recompensar al jugador por completarlos daría al jugador ese interés en

la rejugabilidad y la constancia.

21. Conclusión

Dungeon Revenant ha demostrado ser un proyecto sólido que cumple con los objetivos propuestos desde el inicio. Se ha logrado desarrollar un videojuego funcional en 2D, dentro del género roguelike, que incluye elementos clave como la muerte permanente, combates variados y una experiencia visual 2D propia.

El proceso de desarrollo ha permitido aplicar conocimientos técnicos adquiridos durante el grado superior, desde la lógica de programación, el entendimiento del funcionamiento de hilos hasta la gestión de bases de datos. Además, ha sido un gran aprendizaje en cuanto a la organización, toma de decisiones, resolución de problemas y cómo funciona la programación en Unity, como relacionarlo con php y una base de datos.

A parte de los logros, también se ha podido encontrar aspectos con posibles mejoras, como la generación procedural de mazmorras, la incorporación de una tabla de enemigos en la base de datos y el aumento de contenido.

El desarrollo del proyecto Dungeon Revenant, ha sido una experiencia enriquecedora tanto a nivel técnico como personal. El proyecto tiene potencial para seguir creciendo y mejorando mediante nuevas actualizaciones, adaptándose a nuevas tendencias del mercado. Con una base ya establecida, se abre la puerta a futuras versiones más completas, capaces de competir con títulos consolidados del género roguelike, pero siempre con una identidad propia.