

I.E.S Fuengirola N°1

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO INTEGRADO

Halaman Tower

Departamento de Informática

Manual Técnico

Autor: Juan Carlos Lucena Martinez

ÍNDICE DE CONTENIDOS

1 - Sobre éste proyecto	2
1.1 - Control de versiones	2
1.2 - Licencia de uso	2
2 - Análisis del problema	2
2.1 - Introducción al problema	2
2.2 - Antecedentes	2
2.3 - Objetivos	2
2.4 - Requisitos	2
2.4.1 - Funcionales	2
2.4.2 - No funcionales	2
2.5 - Recursos	2
2.5.1 - Software	2
2.5.2 - Hardware	2
3 - Diseño de la solución software	2
3.1 - Modelados	2
3.1.1 - Casos de uso	2
3.1.2 - [Interacción]	2
3.1.3 - [Estado]	2
3.1.4 - [Actividad]	2
3.2 - Prototipado gráfico	2
3.2.1 - [Escritorio]	2
3.2.2 - [Tablets / Smartphones]	2
3.3 - Base de datos	2
3.3.1 - Diseño Conceptual (ER)	2
3.3.2 - Diseño lógico (tablas normalizadas)	2
4 - Implementación	2
4.1 - Codificación	2
4.1.1 - [Usabilidad]	2
4.1.2 - Backend	2
4.1.3 - Frontend	2
4.2 - [Pruebas]	2
5 - Documentación	2
5.1 - Empaquetado / Distribución	2
5.2 - Instalación	2
5.3 - Manual de Usuario / Referencia	2
6 - Conclusiones	2
7 - Bibliografía	2

1 - Sobre éste proyecto

1.1 - Control de versiones

El control de versiones utilizado será git junto a github

1.2 - Licencia de uso

Tendría una licencia de uso Creative Commons (CC), del tipo CC BY-SA-NC, esto implica que otros usuarios puedan usar, modificar y distribuir el trabajo, siempre que den crédito, las obras derivadas han de compartir la misma licencia y finalmente el impedimento de compartir y modificar el trabajo con fines comerciales

2 - Análisis del problema

2.1 - Introducción al problema

En la actualidad, muchos juegos carecen de una experiencia compacta que combine simplicidad, desafío estratégico y rejugabilidad. Halaman Tower surge para atender esta necesidad, ofreciendo un RPG por turnos con mecánicas de rogue-like, donde el jugador se enfrenta a una serie ininterrumpida de enemigos, eligiendo cómo mejorar sus estadísticas y buscando maximizar su puntuación. Este videojuego busca brindar una experiencia rápida, estratégica y competitiva, ideal para quienes buscan partidas dinámicas pero con profundidad táctica. Además, incorpora un sistema de registro en línea que fomenta la competitividad al permitir a los jugadores comparar sus resultados, integrando de forma innovadora componentes locales y en línea en un formato accesible y rejugable.

2.2 – Antecedentes

En el género de los videojuegos RPG y rogue-like, han existido numerosos títulos que han explorado mecánicas de combate por turnos y progresión aleatoria. Juegos como Slay the Spire, Darkest Dungeon y Hades han demostrado el atractivo de estos sistemas, combinando estrategia y rejugabilidad con un diseño desafiante. Sin embargo, muchos de estos juegos requieren sesiones de juego extensas o presentan una curva de aprendizaje elevada, lo que puede desalentar a ciertos jugadores. Halaman Tower busca ofrecer una alternativa accesible y dinámica, manteniendo la profundidad táctica y la aleatoriedad característica del género, pero con partidas más rápidas y un sistema de progresión sencillo pero desafiante.

Gameplay \ Design	Gameplay Purist	Gameplay Neutral	Gameplay Rebel
	Must be a singleplayer, turn based game focused on exploring areas, defeating NPC enemies, and managing stats and inventory.	Having only a few of the gameplay characteristics of a "pure" roguelike is also allowed.	Gameplay can be based on literally anything.
Design Purist	Must have all key roguelike elements such as random generation, permadeath and high complexity. Past runs having an effect on future runs (metaprogression) is strictly prohibited.	 "HyperRogue is a roguelike"	 "Minecraft Hardcore is a roguelike"
Design Neutral	Having only a few of the design characteristics of a "pure" roguelike is also allowed.	 "Darkest Dungeon is a roguelike"	 "Dead Cells is a roguelike"
Design Rebel	Design choices can be literally anything.	 "Pokémon is a roguelike"	 "Zelda is a roguelike"

2.3 – Objetivos

Un juego capaz de entretener al público, ofreciendo una posibilidad al estilo de juego de cada persona, con capacidad de aprendizaje adaptándose a pequeñas mecánicas que hará mejor o peor al personaje en algunos aspectos, anteponiendo siempre la diversión del jugador

2.4 - Requisitos

2.4.1 – Funcionales

- Creación de cuenta y uso de la misma.
- Implementación de un combate por turnos.
- Generación de enemigos de forma aleatoria (entre los existentes en el juego).
- Permitir la mejora de los atributos seleccionados del personaje.
- Subida de nivel.
- Ser capaz de registrar puntuaciones y mostrarse, así como las mejores puntuaciones.

2.4.2 - No funcionales

- Creación de una interfaz sencilla
- Evitar problemas de rendimiento, buscando que funcione sin necesidad de dispositivos potentes.
- Permitir que el juego pueda escalar para que sea actualizable en el futuro.
- Conexión con la base de datos segura.

2.5 - Recursos

2.5.1 – Software

- Visual Studio Code: Editor de código para escribir scripts en C# y editar configuraciones.
- Unity Editor: Entorno de desarrollo integrado (IDE) para crear el juego.

Software online:

- PythonAnywhere: Servicio web utilizado para alojar el servidor y la base de datos.
- Piskel: Herramienta en línea para la creación de sprites y pixel art.
- MyEdit: Plataforma web para edición de imágenes y audio.
- 123Apps: Recorte de audio online.
- Convertio: Conversión de audio mp3 a ogg.



2.5.2 - Hardware

Se estima que un ordenador con 4GB de RAM cerca de 200mg de espacio en disco. Resolución de pantalla 1280x720 o mismas proporciones. Intel core i5 o superior. Windows 10. Aún que puede funcionar en condiciones menores, estas aseguran un funcionamiento correcto

3 - Diseño de la solución software

3.1 - Modelados

3.1.1 - Casos de uso

Actor: Jugador.

- Iniciar partida.
- Seleccionar habilidad.
- Realizar acción.
- Avazar nivel.
- Seleccionar objeto
- Guardar puntuación.

3.1.2 - [Interacción]

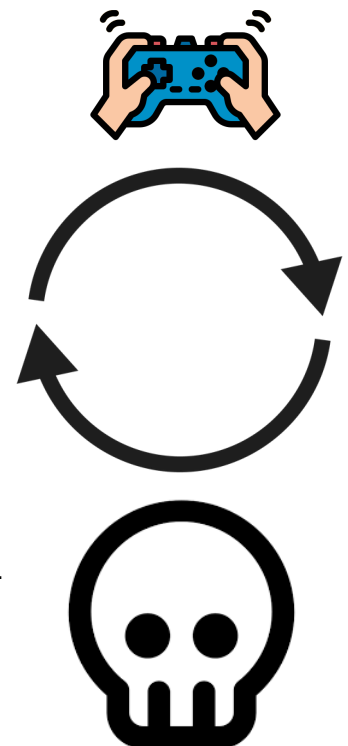
- Se elige una habilidad.
- El juego determina el tipo de habilidad.
- Interactúa con el jugador o enemigo según la habilidad.
- Se actualiza la vida.
- Se eligen objetos para mejorar.
- Se pierde
- Se vuelve a empezar

3.1.3 - [Estado]

Vivo: el personaje / enemigo tiene vida y puede realizar acciones.
Muerto: el personaje/ enemigo ha reducido su vida a 0.

3.1.4 - [Actividad]

1. Inicia el turno del jugador.
2. El jugador usa una habilidad.
3. Se ejecuta la acción y los cambios a la salud que impliquen.
4. Inicia el turno del enemigo.
- 5 Se repite el ciclo hasta que uno caiga



3.2 - Prototipado gráfico

3.2.1 - [Escritorio]



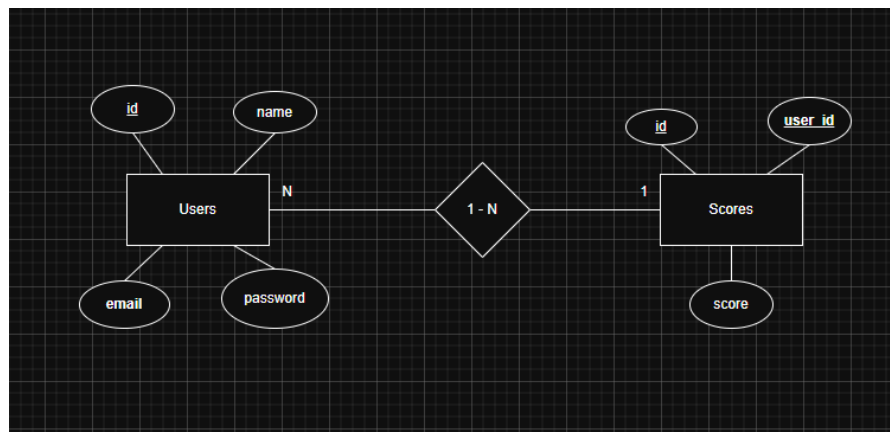
Personaje controlado por el jugador y enemigo controlado por la cpu, un log que indica lo sucedido y que a su vez contiene las habilidades que puede usar el jugador

3.2.2 - [Tablets / Smartphones]

No aplica

3.3 - Base de datos

3.3.1 - Diseño Conceptual (ER)



3.3.2 - Diseño lógico (tablas normalizadas)

Users (id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(50) NOT NULL, email VARCHAR(100) UNIQUE NOT NULL, password VARCHAR(255) NOT NULL)

Scores (id INT PRIMARY KEY AUTO_INCREMENT, user_id INT NOT NULL, score INT NOT NULL, FOREIGN KEY (user_id) REFERENCES Users(id) ON DELETE CASCADE)

4 - Implementación

4.1 - Codificación

4.1.1 - [Usabilidad]

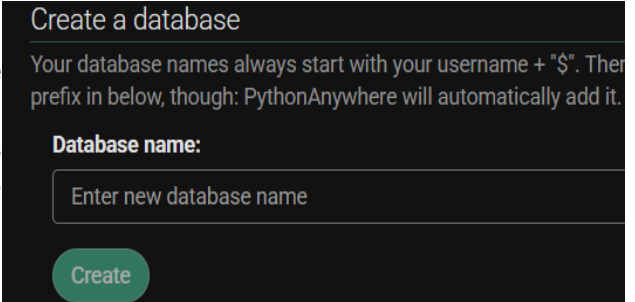
Sencilla e intuitiva de usar, permitiendo su uso a cualquier persona siempre que posea un ordenador con sistema operativo Windows

4.1.2 – Backend

La gestión Backend puede ser dividida en dos facetas distintas, el operativo en el servidor y la usada en Unity tanto para el envío de datos como la funcionalidad.

PythonAnywhere

Comenzando por python anywhere, hemos de crear una cuenta y registrarnos con ella, tras eso, en las diferentes opciones disponibles tendremos “Databases”, en la cual entramos y usamos uno de sus apartados disponibles para crear la base de datos.



Create a database

Your database names always start with your username + "\$". There's a prefix in below, though: PythonAnywhere will automatically add it.

Database name:

Enter new database name

Create

Tras la creación de la base de datos deberemos de irnos a la opción web, donde crearemos una aplicación web, donde crearemos y enlazaremos con el directorio de trabajo, el cual podemos ver justo encima como “Source code”, a su vez será necesaria la creación de un entorno virtual, opción que tendremos justo debajo y con ello instalaremos los requisitos necesarios, como son por ejemplo flask, SQLAlchemy y python, tras ello e inicializar una instancia de SQLAlchemy en el archivo “__init__” tendremos las bases para la creación de las tablas. Donde crearemos en nuestro caso los modelos User y Score, tras ello crearemos las migraciones desde la consola de comandos, finalmente, en app.py configuraremos los datos y crearemos las rutas, como configuración “SQLALCHEMY_DATABASE_URI” para establecer la ruta a la base de datos y conecte y “SQLALCHEMY_TRACK_MODIFICATIONS” junto a “SQLALCHEMY_ENGINE_OPTIONS”(pool_recycle) para aumentar el tiempo necesario hasta que se desconecte automáticamente del servidor.

Seguidamente asociamos “db” con la aplicación, inicializamos “migrate” y establecemos las rutas.

/users: Recoge todos los usuarios que haya en la base de datos, devolviendo su id, usuario y email

/user/create: Verifica que estén todos los datos a la hora de crear un usuario, en caso correcto se intentará crear al usuario en la base de datos, informando con un error si no es posible

/user/login: Usada para el login del usuario, usara el email enviado para buscar coincidencias en la base de datos, informando si no se ha encontrado, en caso de que si exista, se comprobara que la contraseña enviada y la registrada coinciden , en caso correcto se devolverá el id, usuario y email del usuario

/user/edit: Permite la modificación del usuario y el email de un usuario.

/user/delete: Permite la eliminación de una cuenta de usuario.

/scores: Devuelve todos los scores que haya en la base de datos.

/scores/top: Devuelve los 10 scores más altos que haya en la base de datos.

/score/user/<int:user_id>: Devuelve los 10 scores más altos que tenga el usuario.

/score/create: Permite crear un nuevo score, el cual se enlaza al usuario correspondiente.

/score/delete: Permite eliminar un score de la base de datos.

Finalmente con “app.run” iniciamos la ejecución, Con todo esto el servidor esta listo, necesitando unicamente actualizar manualmente en PythonAnywhere la ejecución del servicio una vez cada 3 meses y pulsar recargar la aplicación cuando modifiquemos algo para que se aplique el cambio al servidor.

Unity

Clases:

LoginRequest: Se usa para enviar una solicitud de inicio de sesión al servidor, contiene los datos del usuario: correo electrónico(email) y contraseña(password). Se serializa a JSON antes de enviarla en la solicitud HTTP.

ScoreEntry y ScoreList: Se utiliza al recibir datos del servidor cuando se obtiene el ranking de puntajes para formatearlos estéticamente para el usuario

Combate:

Combat Manager

Start: Genera un enemigo si no existe inicia el loop de combate

CombatLoop: “WAITING_FOR_FIGHTER” mantiene en espera la hasta que el luchador realice su turno.

“FIGHTER_ACTION” Provee de feedback al usuario y realiza la habilidad seleccionada, el estado pasa a comprobar la victoria.

“CHECK_FOR_VICTORY” Si nadie ha sido derrotado continua el combate, si el enemigo es derrotado, el usuario sube de nivel y aparece el siguiente. Si el usuario es derrotado se sube la puntuación al servidor y aparece la pantalla de derrota.

"NEXT_TURN" Pasa el turno al siguiente combatiente.

AddFighter: Agrega un nuevo luchador a la lista.

RemoveFighter: Elimina al luchador enemigo.

GetOppositeFighter: Coge el luchador contrario.

OnFighterSkill: Realiza la acción del usuario.

CheckLevelUp: Comprueba el nivel al que sube el usuario para mostrar el popup de mejora si este es 11.

ShowUpgradePopup: Muestra el popup de mejoras.

OnUpgradeSelected: Según la mejora seleccionada aumenta la vida, el ataque o la defensa en el porcentaje correspondiente.

SendScoreToServer: Envía una puntuación para agregarla a la base de datos, registrada al usuario que jugó.

ShowGameOver: Pantalla de derrota con botón para reinicio de partida.

ForceGameOver: Fuerza la derrota y envía la puntuación al servidor.

RestartGame: Reinicia el juego.

EnemySpawner

SpawnEnemy: Hace aparecer un enemigo, las estadísticas del mismo varía según el nivel, al nivel diez aparece un enemigo de mayor poder al resto. Además lo agrega a al administrador de combate.

RemoveEnemy: Elimina al enemigo de la lista de luchadores.

RespawnEnemy: Hace aparecer el enemigo y administra el nivel que tendrá.

Login:

ApiManager

SendPostRequest: Administra el envío de peticiones post al servidor.

LoginManager

ShowPanel: Muestra el panel.

HidePanel: Oculta el panel.

HandleLogin: Maneja el login recoge los datos puestos por el usuario y manda el login al servidor.

HandleRegister: Maneja el registro de un usuario.

HideLoginRegisterButtons: Oculta los botones iniciales.

ShowLoginRegisterButtons: Muestra los botones iniciales

RegisterUser: Registra a un usuario en el servidor.

LoginUser: Loguea al usuario.

SaveUserSession: Guarda la sesión del usuario.

LoadUserSession: Carga la sesión del usuario

Logout: Cierra la sesión

Personajes:

Fighter

ModifyHeath: Modifica la vida del luchador.

LevelUp: Permite la subida de nivel, la cual sube las estadísticas de formas aleatorias.

ShowStatChanges: Muestra e cambio de estadísticas.

GetCurrentStats: Obtiene los stats actuales.

InitTurn: Método abstracto para cuando inicien el turno los luchadores.

Enemy

InitTurn: Realiza la acción de su turno, en este caso el de la IA creada.

IA: Una IA simple que elige al azar una habilidad del enemigo

Player

Awake: Se establecen las estadísticas base y el nombre del jugador según el usuario del jugador.

InitTurn: muestra el panel de opciones y se sitúan las habilidades en sus respectivas posiciones.

ExecuteSkill: Ejecuta a habilidad seleccionada

Skills:**HealthModSkill**

OnRun: Cogemos la cantidad de vida modificada y la aplicamos a los luchadores

GetModification: Devuelve la modificación de vida según de que tipo sea, "FIXED" aplica una cantidad fija. "PERCENTAGE" Afecta según el porcentaje de vida establecido. "STAT_BASE" modifica la vida según la relación de ataque y defensa de los luchadores.

Skill

Animate: Aplica la animación de partículas seleccionada.

Run: Se ejecuta la habilidad.

SetEmitterAndReciver: Establece quien realiza y recibe la acción.

UI:**LogPanel:**

Write: Permite escribir en el panel

lvlUpInfo: Opción alterna de mostrar subida de nivel, finalmente descartada para la versión 1.0.

MusicManager1

Awake: Hacer sonar la música que no se destruya.

PlayMusic: Al ejecutarse detiene la canción actual y ejecuta la nueva canción preparada.

Options:

OpenOptionsMenu: Abre el menú de opciones.

OptionsMenu

GetUserScores: Muestra las puntuaciones del usuario, formateandolas de forma que sea legible para el usuario.

GetTopScores: Obtiene las puntuaciones más altas del servidor.

PlayersSkillsPanel

ConfigureButtons: Asignamos los botones y el nombre

StatusPanel

SetStats: Establece las estadísticas.

SetHealth: Establece la vida y cambia el color según la cantidad restante.

UpgradePopup

Show: Muestra el popup de mejoras

SelectUpgrade: Selecciona la mejora

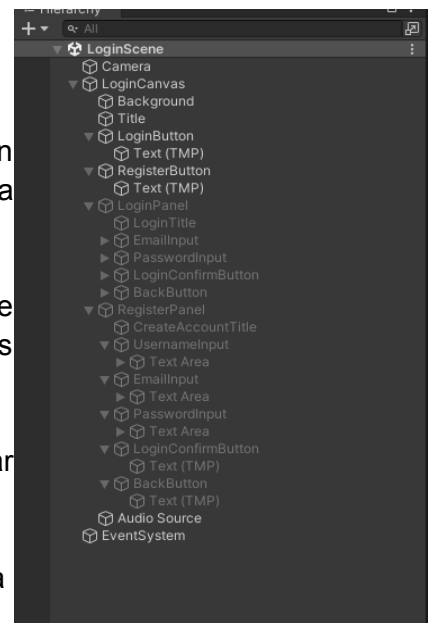
4.1.3 – Frontend**Login**

La pantalla de login contiene dos botones que muestran las opciones de registro o inicio de sesión respectivamente, a la vez que se ocultan a si mismos.

El loginPanel contiene dos campos tipo input en los que poner los datos necesarios para inicio de sesión junto , además de permitir volver atrás.

RegisterPanel contiene los campos necesarios para crear una cuenta, volviendo al menu de login al crearse

Audio Source se encarga de que se reproduzca la música

**Juego**

La pantalla de juego contiene una gran variedad de elementos, empezando por el canvas, el cual contiene el LogPanel en el cual se realiza feedback escrito al usuario.

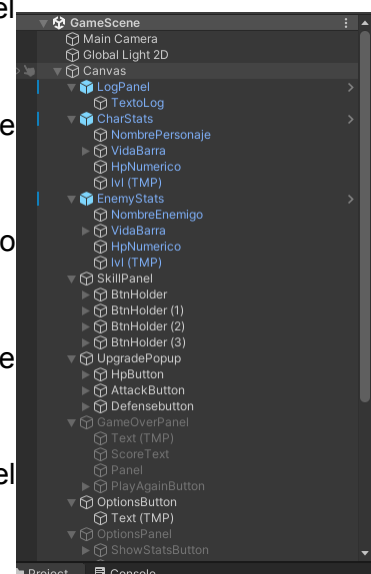
CharStats y EnemyStats, que muestran las estadísticas del jugador y el enemigo, respectivamente

Skill panel contiene los botones con las habilidades que puede llevar a cabo el usuario.

UpgradePopup muestra un panel con tres botones, siendo cada una, una mejora distinta.

GameOverPanel es el panel el cual contiene la pantalla de derrota, el score logrado y un botón que permite volver a jugar

OptionsButton es un boton cuya función es mostrar el panel de opciones



OptionsPanel, a diferencia del botón, muestra todas las opciones que tiene el jugador, las cuales son; ver sus 10 mejores puntuaciones, ver las 10 mejores puntuaciones en el juego, volver al menú, rendirse, ver sus stats y finalmente cerrar el menú.

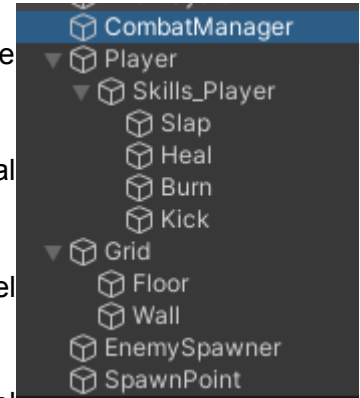
Finalmente, el canvas contiene el AudioSource encargado de reproducir la música.

Una vez fuera del canvas, encontramos el “Player”, el cual contiene las habilidades.

El combatManager será el encargado de administrar el combate.

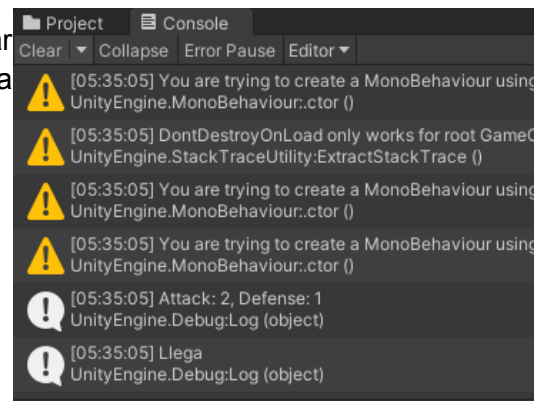
Encontramos a su vez un grid el cual es usado para el escenario del juego.

Finalmente encontramos el “SpawnPoint” el cual indica donde aparecerá el enemigo



4.2 - [Pruebas]

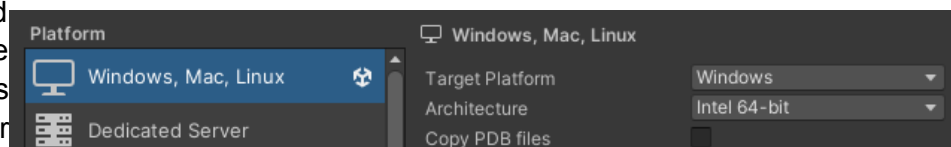
Realizadas en el motor de Unity para verificar el funcionamiento completo y postman para llamadas a la API REST en etapas tempranas



5 - Documentación

5.1 - Empaquetado / Distribución

Será usado unity build settings, lo que permitirá crearse en distintos formatos, entre estos incluye para ordenador (ejecutable windows, mac y linux), dispositivo móvil y WebGL y más, actualmente solo será creada la versión de windows, queriendo una versión WebGL a futuro para plataformas de distribución de juegos indie como “itch.io”.



5.2 – Instalación

No es requerida, debido al formato de unity y al tamaño del juego, ejecutar el programa sería suficiente

Nombre	Fecha de modificación	Tipo	Tamaño
Halaman Tower_BurstDebugInformation_...	09/03/2025 1:55	Carpeta de archivos	
Halaman Tower_Data	08/03/2025 22:51	Carpeta de archivos	
MonoBleedingEdge	08/03/2025 22:51	Carpeta de archivos	
Halaman Tower.exe	08/03/2025 22:51	Aplicación	639 KB
UnityCrashHandler64.exe	08/03/2025 22:51	Aplicación	1.098 KB
UnityPlayer.dll	08/03/2025 22:51	Extensión de la ap...	28.764 KB

5.3 - Manual de Usuario / Referencia

Solo se requiere de ratón.

Ejecute el archivo HalamanTower.exe, Regístrese si no tiene cuenta e inicie sesión. Si tiene cuenta, solo será necesario iniciar sesión.

Seleccione la habilidad que desee, tras ejecutarse su acción, el enemigo realizará la suya, cada 10 enemigos vencidos, aparecerá un objeto para que el usuario se fortalezca

Tras eso, se repetirá el ciclo hasta que uno venza, en caso de ganar el usuario, su personaje se fortalecerá y avanzará a la siguiente dificultad, el objetivo es llegar lo más lejos posible sin perecer.

La puntuación puede ser consultada en el menú de opciones situado dentro del juego, arriba en el centro

6 – Funcionamiento completo del programa

6 – Conclusiones

Halaman Tower ha sido diseñado con la intención de ofrecer una experiencia de juego entretenida, desafiante y accesible. La combinación de mecánicas de RPG y rogue-like en un formato compacto permite que los jugadores disfruten de partidas rápidas pero estratégicas, lo que mejora la rejugabilidad.

Además, posee buenas capacidades de escalabilidad para un crecimiento futuro.

7 – Bibliografía

Unity Technologies. (2024). Unity Manual.
<https://docs.unity3d.com/Manual/>.

Canal de YouTube utilizado como apoyo en la creación:
<https://www.youtube.com/@DanielWayota>

Documentación oficial Python Anywhere:
<https://help.pythonanywhere.com/pages/>

Foros PythonAnywhere:
<https://www.pythonanywhere.com/forums/>

ChatGPT:
<https://chatgpt.com>