UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de un Videojuego de Realidad Virtual basado en Puzle Boxes

TRABAJO FIN DE GRADO

Julio 2025

AUTOR: Enzo Agustín Hernández Reynoso

DIRECTOR: Manuel Quesada Martínez

RESUMEN

Los *Puzzle Boxes*, como su nombre indica, son cajas normalmente de madera con patrones intricados que solo pueden ser abiertas resolviendo un puzle. Estos puzles pueden variar en complejidad, yendo desde algo simple como pulsar un botón oculto hasta algo complejo como la obtención de herramientas ocultas dentro de la caja, y su uso para la apertura de esta. Este proyecto se enfoca en el desarrollo de un videojuego en Realidad Virtual (RV) con la capacidad de crear, compartir y jugar niveles de *Puzzle Boxes* personalizados.

El estudio inicial de videojuegos y herramientas de desarrollo relacionadas permite definir un marco de trabajo para crear un prototipo funcional de un videojuego de Realidad Virtual que permita jugar y distribuir puzles creados por los usuarios mediante un apartado de comunidad con un sistema de valoraciones y comentarios. El editor usado para crear los puzles será personalizable, así que se ajustará la colocación de los bloques de manera más precisa dependiendo de las necesidades del usuario.

La implementación está dividida en dos partes: un videojuego de RV y un conjunto de servicios web que proporcionan acceso a la base de datos compartida que aloja información de la comunidad. Para el desarrollo del videojuego se explora y hace uso de las funcionalidades de Unreal Engine (UE) que permiten: el diseño y la interacción de interfaces, la implementación de un editor de puzles, y la interacción con objetos virtuales. Para la base de datos se usa MySQL y el servidor utiliza tecnología ASP.NET Core Web API. En cuanto al servidor se usa una plantilla de Visual Studio (VS) modificada y dividida en controladores y repositorios, añadiendo también la conexión con la base de datos y la autenticación mediante JWT.

Se ha podido desplegar el prototipo elaborado en un entorno de desarrollo local por medio de un casco Valve Index, lo que ha permitido realizar diversas pruebas con usuarios a lo largo del desarrollo que han llevado a mejoras tanto en la experiencia de usuario como en los aspectos visuales del juego.

Palabras clave: Realidad Virtual, Videojuego, Unreal Engine, Puzles, Interfaz, Comunidad, Editor, Visual Studio, ASP.NET Core Web API, MySQL.

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mi tutor Manuel Quesada, no solo por ser uno de los mejores profesores que he tenido hasta el momento, sino también por apoyar este proyecto desde el inicio, y por su confianza en mí capacidad como Ingeniero en Informática a lo largo del desarrollo de este trabajo.

En segundo lugar, me gustaría agradecer a mis amigos por apoyarme durante todo el proceso de diferentes formas: expresando su interés por la idea, ofreciéndose a realizar test y dando *feedback* constructivo, y proporcionando perspectivas e ideas nuevas. Me gustaría hacer también una mención especial a mi mejor amigo Christian por proporcionarme no solo un ordenador más potente, lo que me ha permitido trabajar de manera más cómoda, sino también por ayudarme haciendo sugerencias de herramientas y otros recursos que han terminado siendo claves para el proyecto.

Finalmente, me gustaría agradecer a mis padres por apoyarme todos estos años mientras estudiaba, a pesar de tener mis altibajos, y por no perder su esperanza hasta que conseguí encontrar mi vocación en la Ingeniería Informática. A mis hermanos, quienes me dieron apoyo cuando más lo necesitaba, y más aún durante este trabajo, proporcionándome el casco de Realidad Virtual utilizado para las pruebas, y ayudándome con el diseño 3D de los bloques y los materiales usados en el juego.

Contenido

1 Introducción

Ya en el 3100 A.C., la civilización egipcia disfrutaba de juegos de puzles como el antiguo *Senet*, que es un juego de mesa de estrategia para dos personas cuyo objetivo es sacar las piezas de un jugador de un tablero y capturar las del contrincante [1]. En la **Figura 1.1:** Senet se muestra un ejemplo de este juego.

Figura 1.1: Senet.

1.1 Introducción a una *Puzzle Box*

En línea con el argumento de *Senet*, las *Puzzle Boxes*, en su definición más simple, son cajas cuyo objetivo es abrirlas [2]. Para ello se ha de resolver un puzle, su complejidad y temática puede variar mucho, siendo algo tan sencillo como una caja de madera donde hay que deslizar las caras del objeto, como en el ejemplo de la **Figura 1.2:** a) *Puzzle Box* básico cerrado b) *Puzzle Box* básico resuelto.

En sus versiones más complejas, este tipo de puzzles pueden ser tan complejos como teclados numéricos, contraseñas, uso de herramientas, cartas, videos u otros elementos que le proporcionen información al usuario, como en el ejemplo de la **Figura 1.3**.

Figura 1.2: a) *Puzzle Box* básico cerrado b) *Puzzle Box* básico resuelto.

Figura 1.3: Puzzle Box con temática de Batman.

1.2 La Realidad Virtual y su contribución a una Puzzle Box

La Realidad Virtual, abreviado RV, es una tecnología que permite la creación de mundos inmersivos e interactivos creados por ordenador. El equipamiento suele consistir en un casco con unas pantallas estereoscópicas, sensores de dirección y altavoces. También se incluyen unos controladores que permiten al usuario interactuar con el entorno. En algunos casos el dispositivo es autosuficiente, por lo que contiene todo el hardware necesario para procesar las imágenes y la lógica, mientras que otros hacen uso de un ordenador para dicho propósito. El uso más común son los videojuegos, aunque también se utilizan en otras áreas como la educación o el diseño 3D, entre otros [3].

Un análisis de octubre de 2022 estimó que había alrededor de 27.7 millones de dispositivos de 6 grados de libertad (6DoF) en el mercado, teniendo en cuenta un crecimiento anual del 51% se espera que esta cifra alcance los 46 millones para finales de 2024, sugiriendo una adopción cada vez mayor de tecnologías más inmersivas y avanzadas [4].

Lo que puede aportar la RV a las *Puzzle Boxes* es la facilidad de creación, puesto que actualmente las forma en que se crean son artesanalmente y requieren de mucho tiempo y conocimiento, lo que encarece el precio. Así mismo el acceso para los usuarios sería mucho más sencillo y el coste de distribución se reduce. Y, por último, se pueden proporcionar opciones de accesibilidad para aquellas personas que tienen dificultades haciendo uso de las *Puzzle Boxes* normales.

1.3 Justificación y objetivos

La motivación principal detrás de este proyecto se encuentra en mi pasión por los videojuegos y mi fascinación por la complejidad de su desarrollo, que empezó desde que era muy pequeño. Desde el código y herramientas que dan vida a estos mundos virtuales, hasta la evolución gráfica y los sistemas hápticos que han llevado a la RV y Realidad Aumentada (RA), he seguido con interés la evolución de esta forma de entretenimiento a lo largo de los años.

Además, mi afinidad por las Matemáticas y la Informática, junto con la satisfacción de aprender algo nuevo, han impulsado mi deseo de emprender el desafío de crear un videojuego. A lo largo de mi carrera, he adquirido conocimientos en diversos lenguajes para abordar prácticamente cualquier proyecto que me proponga, y emprender uno que me permita aprender mientras cumplo uno de mis sueños es una oportunidad que no podía dejar pasar.

En cuanto a los objetivos, el proyecto se centra en la creación de un videojuego de Realidad Virtual, dividiendo el objetivo principal en metas específicas:

1. **Desarrollo de un juego de Realidad Virtual:**
    - Implementación de controles del personaje.
    - Creación de interfaces interactivas.
    - Establecimiento de almacenamiento de datos locales.
    - Definición de metas para los usuarios.

2. **Servicios de almacenamiento y compartición:**
    - Implementación de una base de datos.
    - Desarrollo de un servidor (API) para gestionar solicitudes y respuestas del cliente.
    - Manejo de cuentas de usuario, incluyendo registro, inicio de sesión, y verificación.

3. **Aprendizaje y desarrollo:**
    - Adquirir conocimientos en desarrollo de videojuegos y familiarizarse con las herramientas disponibles.

4. **Reafirmar conocimientos:**
    - Aplicar estándares, tecnologías y metodologías aprendidas a lo largo de la carrera.

1.4 Metodología

Las etapas del proyecto comprenden:

- **Investigación del estado de la cuestión:**
    - Análisis de los videojuegos de puzles en RV más populares actualmente y herramientas disponibles para el desarrollo de videojuegos.
- **Análisis del problema:**
    - Descripción detallada de los requisitos del proyecto.
    - En esta y las siguientes etapas del proyecto, se seguirá una metodología ágil con reuniones semanales entre el alumno y el tutor para evaluar el progreso y ajustar los requisitos y tareas, y para ello se hará uso de Trello.
- **Diseño de la solución:**
    - Selección de las herramientas disponibles para el desarrollo del proyecto, planificación de modelos, y prototipos.
- **Implementación de la solución:**
    - Creación del videojuego.
- **Implantación:**
    - Despliegue del servidor.
    - Despliegue del videojuego.
- **Retrospectiva y trabajo futuro:**
    - Evaluación de lo aprendido y planificación futura.

1.5 Colaboraciones externas

Dada la envergadura y diversidad del proyecto, se optó por una colaboración externa con una estudiante de otra universidad especializada en diseño 3D. Esta

colaboración ha sido fundamental para adquirir elementos esenciales que van más allá de las competencias habituales de la ingeniería informática, agilizando el desarrollo y permitiéndome centrarme en otros aspectos del proyecto.

2 Materiales y Métodos

En esta sección, se realizará un análisis del estado actual de la industria de los videojuegos de RV. Tras ello, se ofrecerá una enumeración detallada de herramientas de desarrollo para videojuegos de RV describiendo sus capacidades, y se realizará una comparativa considerando factores como la flexibilidad, la escalabilidad y la integración con tecnologías específicas de RV.

2.1 Estado del arte

Antes de comenzar a analizar soluciones de RV centradas en el tema central de este trabajo, las Puzzle Boxes, nos gustaría dedicar una breve introducción a la evolución de la industria de los videojuegos de RV.

2.1.1 Historia de la industria de los videojuegos de RV

La transición al mundo virtual tuvo un hito en 1958 con la llegada de *Tennis for Two* (ver **Figura 2.1**), considerado el primer videojuego de la historia. Este juego, visualizado en un osciloscopio, marcó el inicio de una nueva era en la que los jugadores podían interactuar electrónicamente en un espacio virtual [5].

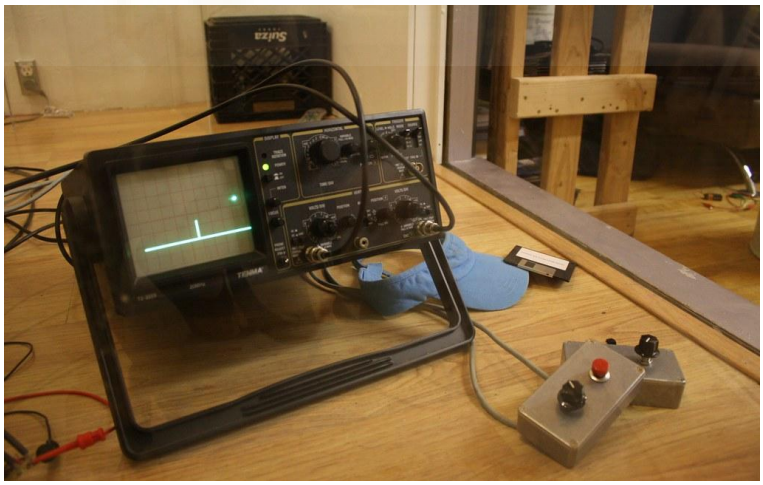

Figura 2.1: Tenis for Two.

Avanzando rápidamente a 1971, se produjo el nacimiento del primer juego de arcade, *Computer Space* (ver **Figura 2.2**), que difería significativamente de sus predecesores en términos de controles e inmersión. Este juego, creado por futuros fundadores de Atari, jugó un papel crucial en el ascenso de los videojuegos arcade como un modelo de negocio rentable [6].

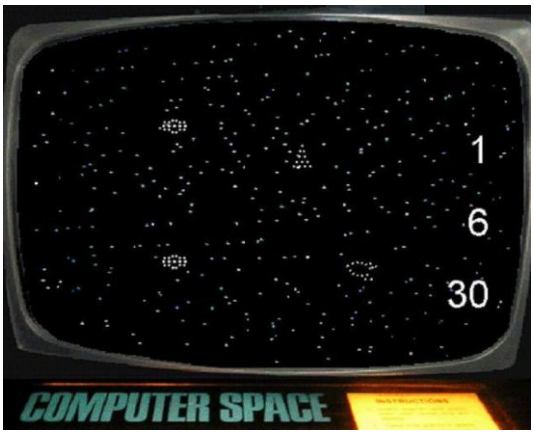

Figura 2.2: Compture Space.

El salto a los hogares se materializó en 1972 con el lanzamiento de la primera consola, la *Magnavox Odyssey* (ver **Figura 2.3**), que incluía un juego similar a *Pong*. La popularidad de esta consola, que contaba con el respaldo del juego de arcade, contribuyó al crecimiento de la industria [7].

Figura 2.3: Magnavox Odyssey.

Figura 2.4: Guantes Sayre de Sandin y Defanti

El resto de la década de 1970 y la de 1980 vieron el desarrollo de dispositivos pioneros, como los guantes de reconocimiento de gestos de Daniel J. Sandín y Thomas A. DeFanti en 1977 [8] (ver **Figura 2.4**). Mientras que VPL, fundada en 1985, creó dispositivos como *DataGlove* y *EyePhone HMD* [9] (ver **Figura 2.5**).

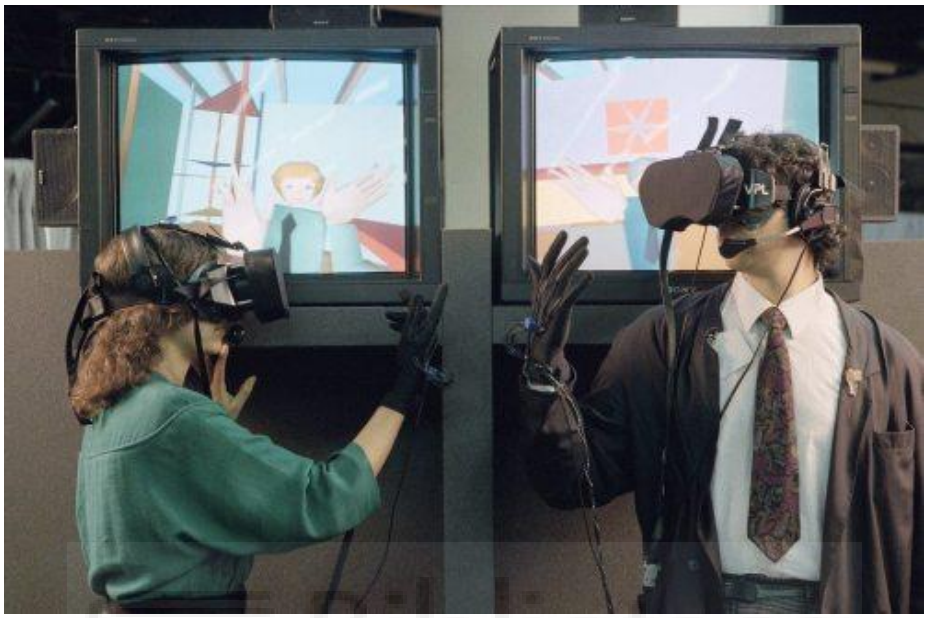

Figura 2.5: EyePhone y *DataGloves* de VPL

Sin embargo, la industria también se enfrentó a una crisis debido a la saturación de videojuegos en el mercado, la baja calidad de estos, y la falta de reseñas confiables. El colapso resultante se superó en parte gracias a Nintendo, que lideró un resurgimiento de las consolas entre 1985 y 1988 [10].

En 1988, Sega lanzó la Mega Drive (ver **Figura 2.6**), también conocida como Mega Genesis, acompañada por el primer intento de Realidad Virtual con *Space Harrier 3D*. Aunque el módulo de VR se canceló (ver **Figura 2.7**), este evento marcó la entrada de la VR en la escena de los videojuegos [11].

Figura 2.6: Mega Drive.

Figura 2.7: Módulo VR para la Mega Drive/Genesis.

El período entre 1990 y 1995 fue testigo de eventos notables, como la exhibición de la máquina arcade de Realidad Virtual *Virtuality* [9] (ver **Figura 2.8**).

Figura 2.8: Máquina VR *Virtuality*.

En 1995, Nintendo presentó la *Virtual Boy* (ver **Figura 2.9** y **Figura 2.10**), la primera consola de Realidad Virtual comercial. Sin embargo, debido a varios inconvenientes, como la pantalla monocromática y la falta de juegos atractivos, la consola fracasó en el mercado [12], [13], [14].

Figura 2.9: Virtual Boy

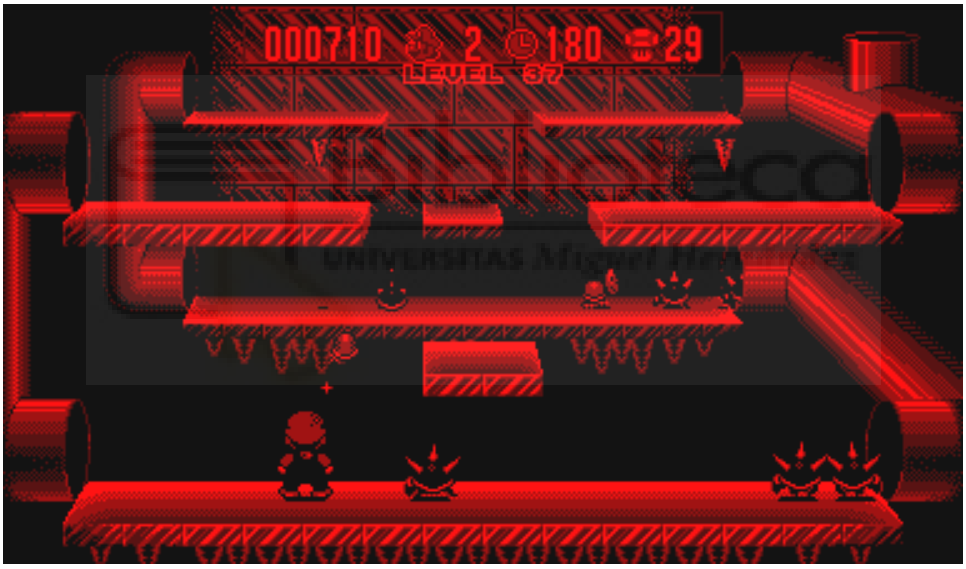

Figura 2.10: Juego para la VB: Mario Clash.

Figura 2.11: Versión en desarrollo de Oculus Rift.

Ingresando al siglo XXI, Palmer Luckey desató un resurgimiento del interés en 2010 con el prototipo de Oculus Rift (ver **Figura 2.11**), financiadas inicialmente a través de Kickstarter y posteriormente adquiridas por Facebook/Meta, marcando el inicio de la nueva era de los cascos de VR. En 2016, HTC lanzó el VIVE Steam VR (ver **Figura 2.12**), permitiendo a los usuarios moverse libremente con sensores. En 2018, Facebook reveló un prototipo con lentes de foco variable y un ángulo de visión de 140 grados [9].

Figura 2.12: Kit de VIVE Steam VR.

El año 2019 trajo innovaciones significativas con el lanzamiento del Valve Index por parte de Valve (ver **Figura 2.13**). Este casco mantuvo la conexión por cable, pero introdujo mandos capaces de seguir cada dedo individualmente, detectando la presión y permitiendo una interacción más intuitiva y rica, así mismo, Valve respaldó la tecnología VR con su juego *Team Fortress 2*, allanando el camino para la compatibilidad de otros títulos y el surgimiento de nuevos juegos independientes financiados por Facebook [15][16].

Figura 2.13: Kit de Valve Index VR.

La última década ha sido testigo de una rápida evolución en la tecnología de Realidad Virtual, con desarrollos continuos y avances que mejoran la inmersión y la interactividad para los usuarios. A medida que la industria avanza, se espera que nuevas innovaciones, como los rumoreados Valve Index 2: *Deckard*, continúen llevando la Realidad Virtual a nuevas alturas [17][18].

Así mismo, los videojuegos tampoco se quedan atrás, con lanzamientos constantes de nuevos títulos. Entre los más destacados se encuentran *Beat Saber*, *Half-Life: Alyx* (ver **Figura 2.14** y **Figura 2.15**), *Horizon: Call of the Mountain* y muchos otros, que ofrecen experiencias inmersivas y avanzadas en el emocionante mundo de la Realidad Virtual [19].

Figura 2.14: Gameplay de Half Life Alyx (I).

Figura 2.15: Gameplay de Half Life Alyx (II).

2.1.2 Análisis de juegos de RV de puzles

A continuación, se identificará ejemplos representativos del subgénero de los videojuegos de puzles. También se llevará a cabo un análisis comparativo de juegos de RV de puzles populares entre los jugadores, y que, o bien comparten similitudes o serán de inspiración de este trabajo.

2.1.2.1 Keep Talking and Nobody Explodes

Keep Talking and Nobody Explodes [20] (ver **Figura 2.16**) ofrecen una experiencia única y cooperativa donde un jugador intenta desactivar una bomba mientras el resto de los jugadores proporcionan instrucciones en un tiempo limitado. Este juego destaca por la importancia de la comunicación, el trabajo en equipo y la resolución de patrones. Categorizado como juego de puzles, requiere encontrar la combinación correcta de acciones para desactivar la bomba antes de que explote, añadiendo una capa de tensión y estrategia al género.

Es importante señalar que parte de los niveles es aleatoria, lo que implica que la solución no será la misma en partidas consecutivas, proporcionando así una experiencia dinámica y desafiante.

Figura 2.16: Keep Talking and Nobody Explodes VR

2.1.2.2 Moss

Moss [21] (ver **Figura 2.17**) se presenta como un juego de aventura en tercera persona, donde los jugadores deben ayudar al protagonista a avanzar manipulando elementos del entorno. La peculiaridad de *Moss* radica en su perspectiva teatral, ofreciendo una visión única de los niveles como si el jugador fuera un espectador. Este enfoque aporta una dinámica diferente al género, brindando una experiencia envolvente y visualmente atractiva.

Figura 2.17: Moss

Figura 2.18: The Room VR: A Dark Matter

2.1.2.3 The Room VR: A Dark Matter

The Room VR: A Dark Matter [22] (ver **Figura 2.18**) presenta una experiencia de aventura en primera persona tipo *scaperoom*, sumergiendo al jugador en diversos entornos donde se deben resolver puzles para descubrir los misterios de un arqueólogo desaparecido. La manipulación de elementos en el mundo tridimensional y la interacción con el entorno son elementos fundamentales para avanzar en el juego. La combinación de narrativa intrigante y desafíos de puzles enriquece la experiencia del jugador.

2.1.2.4 Puzzling Places

Puzzling Places [23] (ver **Figura 2.19**) redefine el concepto tradicional de puzle al llevarlo a la tercera dimensión en un entorno de realidad virtual. Este juego en primera persona permite a los jugadores ensamblar puzles pieza a pieza en un espacio tridimensional, ofreciendo una perspectiva única y desafiante. La experiencia inmersiva y la innovadora mecánica de puzles añaden un nuevo nivel de complejidad y entretenimiento al género de los puzles en VR.

Figura 2.19: Puzzling Places

	Puzles	RV	Niveles aleatorios	Multijugador	Tabla de Clasificación	Creación de niveles
Keep Talking and Nobody Explodes	✓	✓	✓	**Parcial**	✓	✓
Moss	✓	✓	✗	✗	✗	✗
The Room VR: A Dark Matter	✓	✓	✗	✗	✗	✗
Puzzling Places	✓	✓	✗	✓	**Parcial**	✗

Tabla 1: Comparativa de características útiles en juegos relacionados

Las características identificadas en cada juego se pueden ver resumidas en la

Tabla 1. En general, estos juegos comparten la premisa central de desafiar a los jugadores con rompecabezas diversos y estimulantes en un entorno de Realidad Virtual, algunos de ellos adoptan la idea de incluir una tabla de clasificación.

Usando estos juegos como referencia para el desarrollo de este trabajo, se han extraído las siguientes ideas:

- Keep Talking: se extrae la idea de niveles cambiantes hechos por los usuarios que den lugar a crear una comunidad activa.
- The Room VR: en esta *scaperoom* se hace uso de puzle boxes, por lo que la forma de integrarlos y manipularlos se tomó aquí como referencia.
- Puzzling Places: sirvió como inspiración para implementar un editor de niveles, así como la forma de manipular las piezas que propone.

2.1.3 Análisis de herramientas para desarrollar el videojuego

Una vez analizados referentes desde el punto de vista de jugabilidad, dirigimos nuestro análisis a la parte más técnica con el fin de identificar las herramientas y tecnologías disponibles para desarrollar nuestra solución.

Merece la pena destacar que el desarrollo de un video juego de RV requiere el desarrollo de distintos tipos de artefactos. Por ejemplo, el *front-end* es el videojuego donde el usuario realiza sus funciones e interactúa con el sistema. A su vez, para la implementación de este *front-end* tenemos tres grandes categorías de herramientas a utilizar: los motores de juegos, los diseñadores de *assets*, y los lanzadores de juegos.

2.1.3.1 Motores de juegos: Unity y Unreal

Los motores de juegos son *frameworks* que permiten construir videojuegos de una manera más sencilla. Entre las opciones más destacadas se encuentran:

- **Unity Game Engine** [24]**:** este motor de juegos multiplataforma, basado en C# y C++, es una elección popular para el desarrollo de juegos en 2D y 3D. Unity no solo proporciona un entorno robusto para la creación de mundos virtuales, sino que también destaca por su activa comunidad, una

amplia variedad de recursos educativos, y un plan especial para estudiantes que facilita su aprendizaje y uso. En la **Figura 2.20** se muestra un ejemplo de la interfaz de la herramienta.

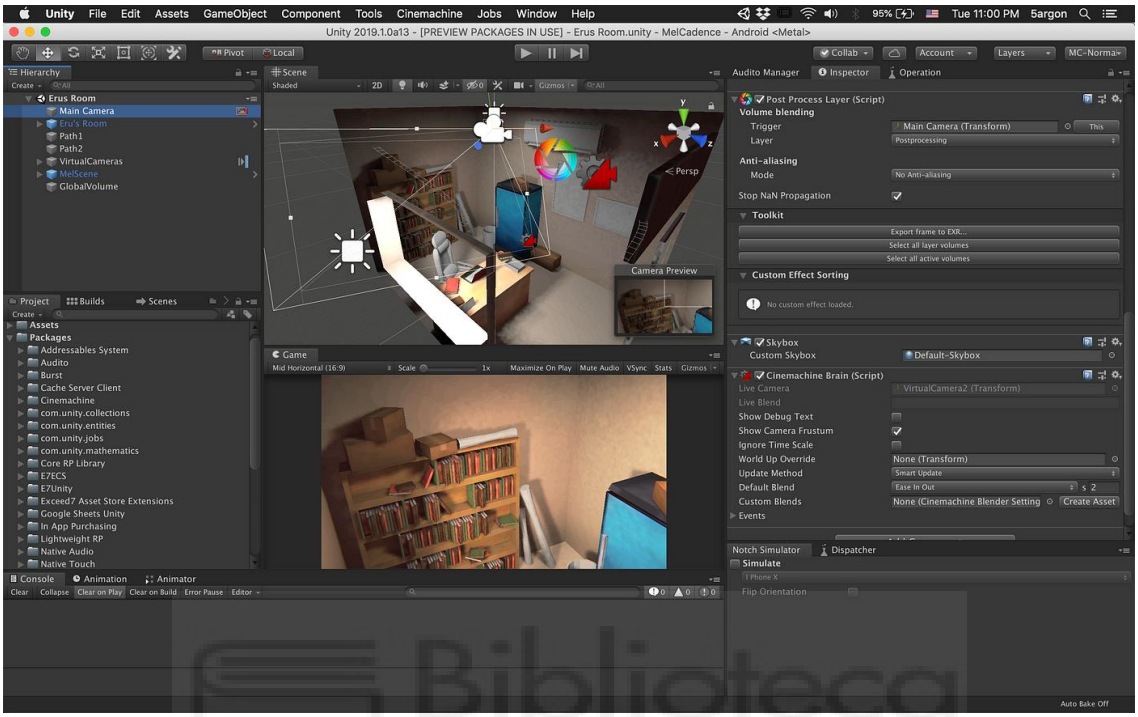

Figura 2.20: Diseñador de niveles de Unity

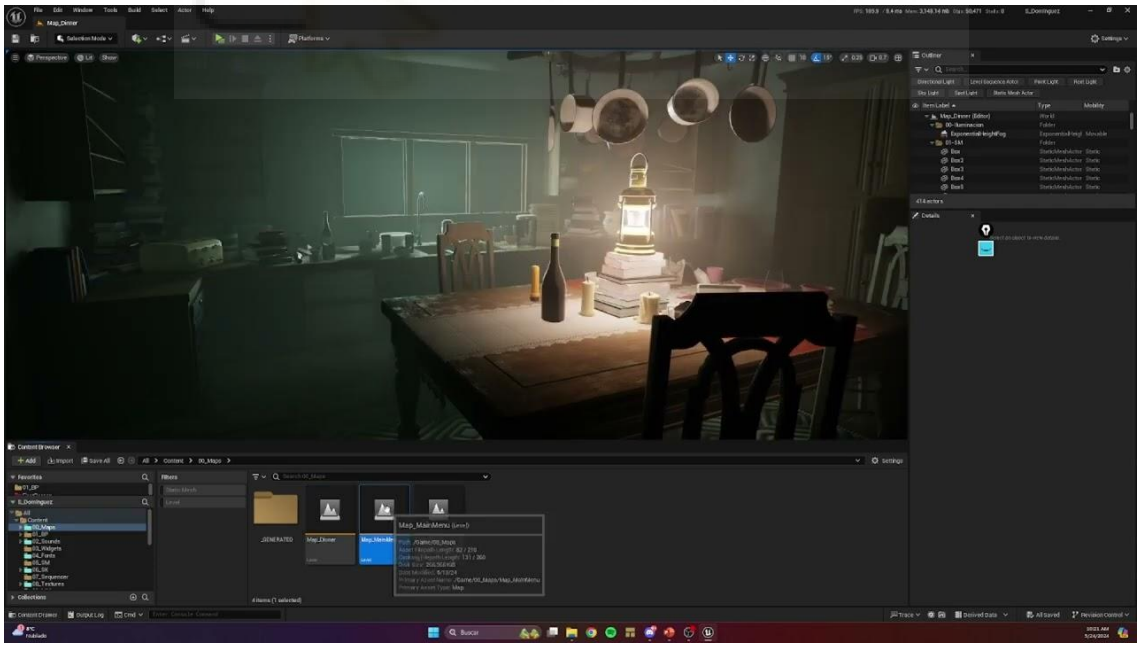

Figura 2.21: Renderizado en Unreal Engine

- **Unreal Engine 5** [24]**:** es otro motor de renombre en la industria, basado en C++. Con características como Nanite y Lumen, ahora disponibles

también para experiencias de realidad virtual. La inclusión de tutoriales y documentación específicos para el desarrollo de VR hace de Unreal Engine 5 una opción atractiva para los desarrolladores que buscan llevar sus creaciones al siguiente nivel. En la **Figura 2.21** se muestra un ejemplo de la interfaz de la herramienta.

Cuando se evalúan estas dos principales herramientas de desarrollo, Unity y Unreal, resulta esencial analizar varios aspectos clave para determinar cuál se adapta mejor a los objetivos del proyecto que es desarrollar un videojuego de Realidad Virtual.

- **Lenguaje de programación:** Unity utiliza C#, tanto para el editor estándar como para cualquier plugin importado. Por otro lado, Unreal emplea C++. Ambos motores comparten la característica de incluir un lenguaje visual basado en nodos; Unity utiliza Bolt, mientras que Unreal utiliza Blueprint o Visual Scripting, ambos lenguajes propietarios.

- **Dispositivos de Realidad Virtual compatibles:** Unity, al basarse en C#, ofrece una alta compatibilidad con diversas plataformas VR, incluyendo Oculus, Playstation VR, Google ARCore, WMR, Magic Leap, y mediante Unity XR SDK, soporta SteamVR y Google Cardboard. En cambio, Unreal es compatible con Oculus, SteamVR, HTC Vive, WMR, Samsung Gear VR y Google VR. Para este proyecto, el uso de SteamVR como plataforma intermedia permite la compatibilidad con cualquier dispositivo que cumpla con el estándar OpenXR.

- **Facilidad de aprendizaje:** Unity es elogiada por su facilidad de uso, especialmente para principiantes, gracias a su uso de C#, un lenguaje relativamente fácil de aprender. Por otro lado, Unreal, al hacer uso de C++ y/o Blueprint Visual Scripting, se percibe como teniendo una curva de aprendizaje más pronunciada.

- **Disponibilidad de recursos y documentación** en términos de documentación, Unity ofrece Unity Learn, una plataforma con recursos en vivo o bajo demanda, videotutoriales de la comunidad en YouTube, un foro con más de 7,138 hilos sobre VR/AR, y un manual de usuario

exhaustivo. Por su parte, Unreal proporciona seminarios abiertos a todos, videotutoriales de la comunidad en YouTube, un foro con más de tres mil hilos, un foro comunitario categorizado y documentación detallada sobre el motor y el desarrollo en VR. Ambas herramientas cuentan con recursos que respaldan el aprendizaje y el desarrollo continuo del proyecto.

En la **Tabla 2** se resumen de las características más relevantes identificadas en cada motor:

	Unity	**Unreal**
Lenguaje de programación	C#	C++
Lenguaje visual de nodos	✓	✓
Dispositivos de VR compatibles	Oculus, Playstation VR, Google ARCore, WMR, Magic Leap, y mediante Unity XR SDK, soporta SteamVR y Google Cardboard	Oculus, SteamVR, HTC Vive, WMR, Samsung Gear VR y Google VR
Facilidad de aprendizaje	Media	Alta
Documentación y recursos para el aprendizaje	✓	✓

Tabla 2: Comparativa de características de Unity y Unreal.

2.1.3.2 Diseñadores de assets: Blender y Autodesk Maya

Una vez analizados los motores, debemos analizar los diseñadores de assets. Éstos son programas dedicados a la creación de elementos 3D incluyendo modelos, animaciones, texturas, y efectos visuales, entre otros. Aunque hay un gran rango de programas disponibles, en este análisis nos hemos centrado en dos de los más populares:

- **Blender** [25]: este programa de código abierto se está convirtiendo poco a poco en uno de los estándares de la industria indie. Fue creado para ser capaz de manejar casi todo el proceso de desarrollo 3D, intentando reducir el uso de diferentes programas. Además, su atractivo reside tres factores calve, (1) su constante desarrollo, (2) su rápido crecimiento

gracias a las aportaciones de la comunidad y (3) que es gratuito tanto para usos personales y comerciales.

- **Autodesk Maya** [26]: Maya es el estándar de las grandes producciones desde hace más de una década, centrándose sobre todo en su uso para modelado, animación, iluminación y renderizado. Es una herramienta poderosa que, al contrario que Blender, fue pensada para ser usada en conjunto con otros programas en el desarrollo 3D. Aunque su alto coste limita su accesibilidad, existe una licencia gratuita para estudiantes.

Tanto Blender como Autodesk Maya ofrecen los recursos necesarios para la creación de assets del videojuego. Estos assets pueden ser importados al proyecto directamente y utilizados en los escenarios o como objectos interactuables sin ningún tipo de alteración. En la **Figura 2.22** se muestra como ejemplo la interfaz ofrecida por Blender para diseñar un asset 3D.

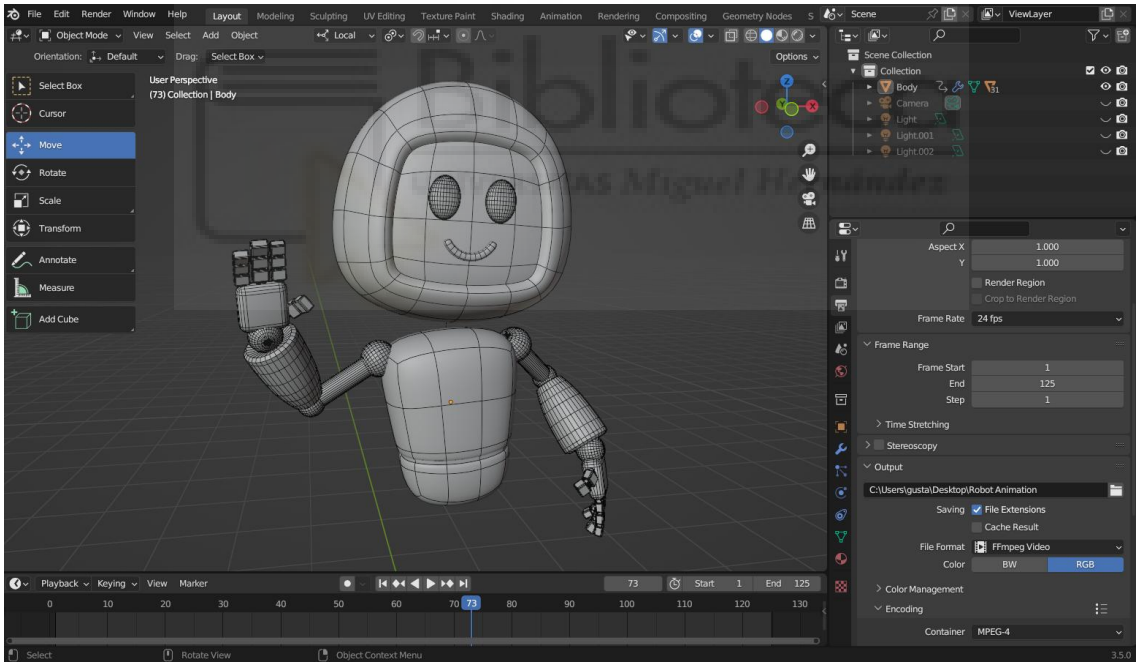

Figura 2.22: Ejemplo de la interfaz de Blender para diseñar un asset 3D

2.1.3.3 Lanzadores: Steam, Epic Games Store y Itch.io

Finalmente, debemos dirigir nuestro análisis a los lanzadores. Los lanzadores son plataformas diseñadas para la distribución digital de videojuegos, en ellos se puede comprar, descargar y ejecutar juegos. Las tiendas de los lanzadores

ofrecen tanto juegos de pago como gratuitos. Dentro de las opciones más populares se encuentran los siguientes lanzadores:

- **Steam** [27]: es una plataforma desarrollada por Valve Corporation para la distribución de juegos propios de la compañía y de terceros. Actualmente es la tienda para PC gaming de más renombre, con una comunidad muy establecida.
- **Epic Games Store** [28]: otra plataforma muy conocida desarrollada por Epic Games, cuyo punto de interés es el porcentaje de comisión que la compañía se lleva por cada venta, el cual es muy bajo en comparación a otras plataformas, actualmente un 12% vs un 30% que ofrece Steam.
- **Itch.io** [29]: el catálogo de itch.io es muy diferente al de Steam o Epic, dado que se centra completamente en la venta de contenido indie. Ya sean videojuegos, comics, música o assets. El beneficio que se lleva la plataforma lo decide el propio desarrollador, pudiendo incluso elegir que la plataforma se lleve un 0% de comisión por las ventas de un juego.

Figura 2.23: Logotipos de los lanzadores analizados. En orden, Steam, Itch.io y Epi Games Store

2.1.4 Análisis de herramientas para desarrollar el API

Como hemos comentado anteriormente, se pretende permitir al usuario subir sus propios niveles y al resto de jugadores a acceder a ellos a través de una funcionalidad consumida por el videojuego desde un servidor. En este apartado vamos a explorar las tecnologías disponibles para realizarlo.

2.1.4.1 Servidor: Django, Express.js, Spring y ASP.NET Core

Una forma muy sencilla de enviar y recibir los datos del servidor es por medio de una API, por lo que se tendrán que crear *endpoints* que puedan ser llamados por medio de peticiones HTTP/HTTPS y que devuelvan la información necesaria para ser mostrada en el juego.

Para conseguir esto valoraremos los lenguajes y *frameworks* más predominantes actualmente. Entre las opciones más destacadas se encuentran:

- **Django (Python):** Framework web para Python que destaca por su enfoque rápido y práctico en el desarrollo de APIs además de una estructura sólida y herramientas integradas. Ofrece un abordaje *"baterías incluidas",* incluyendo *Object Relational Maping* (ORM) o Mapeo Objeto-Relacional (MOR) para facilitar la interacción con bases de datos. Por último, su orientación a la seguridad y medidas incorporadas contra vulnerabilidades comunes lo convierten en una buena opción.

- **Express.js (JavaScript - Node.js):** Framework para Node.js basado en JavaScript, es conocido por su sencillez y velocidad en el desarrollo de APIs. Cuenta con una arquitectura minimalista, permite construir APIs eficientes, respaldadas por una comunidad activa y documentación completa.

- **Spring (Java):** Frameworks para el desarrollo Java, ofrece herramientas completas para construir APIs. Su modularidad, inversión de control (IoC) y módulos especializados como Spring Boot facilitan el desarrollo rápido y robusto de APIs escalables.

- **ASP.NET Core (C#):** Hace uso de C# para construir aplicaciones web y APIs. Su versatilidad, eficiencia en proyectos complejos y escalabilidad, junto con una integración eficiente con herramientas de desarrollo como Visual Studio, hacen de este lenguaje una gran opción.

Cuando se evalúan lenguajes de programación, es fundamental considerar la velocidad de desarrollo de aplicaciones, la calidad de la documentación, la

disponibilidad de librerías y otros recursos, y, en el caso de las APIs, aspectos como escalabilidad, seguridad y rendimiento, entre otros.

Dado que el enfoque principal del proyecto reside en la creación del videojuego y no en el servidor, la elección del lenguaje para este último se ha basado mayormente en la familiaridad. Por ende, se descartaron tanto Django como Express.js, ya que Python no se ha utilizado de manera extensa a lo largo de la carrera y nunca se ha explorado Node.js. En cambio, se tiene experiencia con Spring (Springboot) durante prácticas extracurriculares realizadas el último año y con ASP.NET Core durante la carrera para el desarrollo de aplicaciones web.

Al analizar las diferencias clave entre ambos lenguajes, se observa que Spring cuenta con una comunidad muy activa, mientras que ASP.NET dispone de una documentación detallada. Spring se integra eficientemente con otras tecnologías de Java, mientras que ASP.NET tiene una gran compatibilidad con las tecnologías de Microsoft y es multiplataforma. A pesar de que ambas herramientas ofrecen buen rendimiento y escalabilidad, la diferencia principal radica en que Spring requiere una configuración inicial más compleja que la de ASP.NET.

2.1.4.2 Base de datos: MySQL y PosgreSQL

Para la persistencia de datos es crucial decidir el tipo de base de datos que se utilizará. En este contexto, se pueden identificar dos categorías principales:

- **Relacionales:** estas bases de datos establecen conexiones entre las tablas a través de relaciones, utilizando claves primarias y foráneas. Este enfoque facilita la integridad referencial y la consistencia de los datos.

- **No relacionales:** a diferencia de las anteriores, estas bases de datos prescinden de relaciones para conectar las tablas y emplean otros métodos para almacenar la información.

Dado que el proyecto se basa en estructuras bien definidas y no variables, y existe una relación clara entre estas estructuras, se opta por una base de datos relacional. Esto garantiza la eliminación de todos los datos relacionados al eliminar un registro, preservando así la integridad referencial y la consistencia de los datos, aspectos esenciales para el proyecto.

Entre las herramientas y tecnologías disponibles para utilizar una base de datos relacionales, las opciones más destacadas son las siguientes:

- **MySQL:** Esta base de datos de código abierto es compatible con varios lenguajes de programación, operativa tanto localmente como en un servidor, fácil de instalar, y de alto rendimiento y escalabilidad.

- **PostgreSQL:** Otra base de datos de código abierto, reconocida por su robustez, fiabilidad de datos y alto rendimiento. Además, incluye siempre características avanzadas como ACID y control de concurrencia, entre otros.

En la comparación entre ambas bases de datos, el enfoque primario para este proyecto es la velocidad de desarrollo. Aunque PostgreSQL ofrece una gama más amplia de funcionalidades que MySQL, se ha decidido no considerarlas debido al alcance del proyecto. En términos de robustez, rendimiento y escalabilidad, no se observan diferencias significativas que afecten al proyecto.

2.1.5 Herramientas seleccionadas y su justificación

La elección de las herramientas para el desarrollo del proyecto se ha basado en criterios específicos orientados a las necesidades particulares del juego a desarrollar en este trabajo. Considerando que el proyecto se pretende lanzar en la plataforma Steam y se centrará en la manipulación de objetos cercanos, la herramienta debe ser compatible con OpenXR y permitir el desarrollo de ambientes fotorrealistas.

- **Motores de juegos:** se ha tomado la decisión de emplear Unreal Engine 5. Esta elección se fundamenta en las nuevas tecnologías exclusivas incluidas en la última versión por Epic Games, el creador de Unreal Engine. Además, al utilizar Steam como plataforma de lanzamiento, la capacidad de adaptación y remapeado de controles para distintos dispositivos VR elimina la ventaja que Unity podría tener en términos de soporte de plataformas.

  El rendimiento también ha sido un factor determinante en la elección, ya que Unreal Engine 5 integra la tecnología Nanite, lo que implica un notable aumento de rendimiento en dispositivos de gama baja. Aunque se

reconoce la diferencia en la curva de dificultad entre Unity y Unreal, se confía en que los conocimientos previos en C, C++, Java y otros lenguajes adquiridos durante la carrera contribuirán a nivelar la curva de aprendizaje.

Otra característica importante a tener en cuenta es el motor de físicas de Unreal Engine, que proporciona un comportamiento realista en las interacciones objeto-objeto y jugador-objeto en el juego. Gracias a este motor se espera incorporar gravedad, colisiones y movimientos dinámicos a los objetos, lo que proporcionará una mayor inmersión e interacción por parte del usuario.

- **Diseñadores de assets:** dado que actualmente se cuenta con una licencia de Maya, se ha decidido hacer uso de ambos diseñadores para los propósitos que se consideren oportunos en cada momento.

- **Lanzador:** por razones de compatibilidad con el motor elegido y el dispositivo VR disponible para las pruebas se ha decidido seleccionar Steam como lanzador para el proyecto. No obstante, también se hará uso de Epic Games ya que es el lanzador donde se encuentra de manera exclusiva Unreal Engine.

- **Lenguaje de programación y servidor:** ante la duda de usar Spring o ASP NET CORE, Spring cuenta con una gran cantidad de librerías, una comunidad muy activa e inyección de dependencias, aunque eso a su vez hace que el código pueda acabar siendo muy complejo. Por otro lado, se ha realizado múltiples proyectos con ASP.NET Core, tiene una gran cantidad de documentación, es multiplataforma, y tiene un alto rendimiento y escalabilidad. Mirando ambos lenguajes se ha decidido hacer uso de ASP.NET Core dado que se tiene una mayor profundidad de conocimiento y para el tamaño del proyecto es más ideal.

- **Motor de base de datos:** se ha decidido usar MySQL por su previo uso en varios proyectos, a diferencia de PostgreSQL que no se ha utilizado anteriormente. Además, la infraestructura y configuración necesarias para implementar MySQL en el proyecto ya están disponibles, mientras que

para PostgreSQL requeriría investigaciones y pruebas de implementación adicionales.

2.1.6 Otras herramientas utilizadas

Además de la herramienta principal para el desarrollo del proyecto, se han empleado varias herramientas complementarias. Estas herramientas (ver logotipos en la **Figura 2.24**) han sido seleccionadas para cubrir diferentes aspectos del desarrollo, desde la gestión de referencias bibliográficas hasta el manejo eficiente del código y las pruebas del servidor. Su integración ha contribuido a un flujo de trabajo más organizado y ha facilitado la consecución de los objetivos del proyecto.

- **Mendeley:** herramienta integral para la gestión y citación de referencias bibliográficas. Mendeley ha facilitado el almacenamiento ordenado de información bibliográfica y asegura la correcta referencia de fuentes en el proyecto.
- **SteamVR:** espacio virtual proporcionado por Steam que actúa como controlador central para los dispositivos VR utilizados en el desarrollo del proyecto.
- **Bitbucket:** plataforma de almacenamiento del proyecto, se utiliza para el control de versiones y como lugar centralizado para enviar y almacenar datos mediante SourceTree.

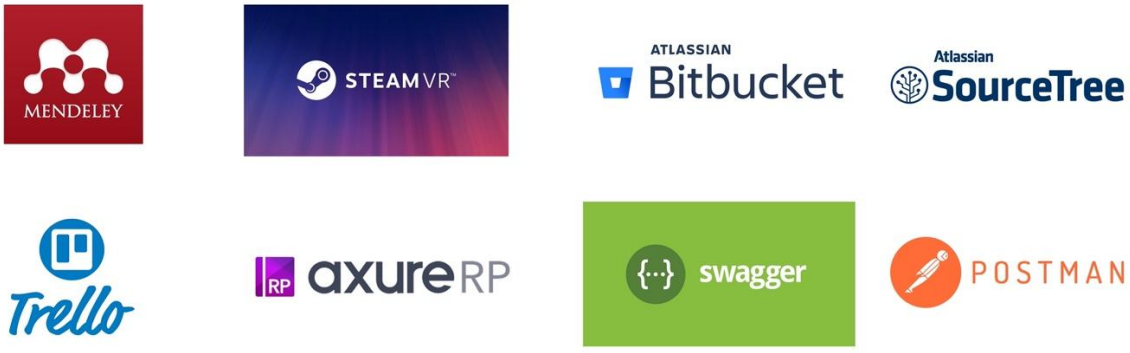

Figura 2.24: Resumen de los logotipos de otras herramientas utilizadas

- **Sourcetree:** interfaz gráfica de usuario (GUI) para Git, permite un manejo intuitivo del repositorio y el control de versiones del proyecto, y facilita la gestión eficiente de cambios en el código fuente.
- **Trello:** herramienta de gestión de proyectos basada en tableros, Trello se usa para el seguimiento y planificación de tareas a lo largo del desarrollo del proyecto. Facilita una gestión ágil y eficaz del flujo de trabajo.
- **Axure RP:** herramienta de diseño y prototipado. Esto permite crear wireframes, prototipos y documentación para visualizar y validar los conceptos de diseño antes de la implementación final.
- **Swagger:** herramienta que simplifica y automatiza el proceso de documentación de APIs. Proporciona una interfaz visual que permite describir, consumir y visualizar servicios web sin la necesidad de acceder directamente al código fuente. Genera documentación dinámica para la API utilizada en el proyecto, lo que facilita la comprensión y el uso de los diferentes puntos de acceso.
- **Postman:** Herramienta colaborativa de desarrollo de APIs que se usa para realizar pruebas del servidor cuando aún no se dispone de las interfaces necesarias. Y ha permitido realizar pruebas y validar el funcionamiento del servidor antes de la implementación completa.

2.2 Propuesta de solución

A modo de resumen, en la **Figura 2.25** se muestran la arquitectura del videojuego a desarrollar, junto con los lenguajes y tecnologías elegidos. Para el desarrollo del proyecto, el *hosteo* del servidor y base de datos, y el uso del juego se dispone de una portátil XMG CORE 15, que es un portátil de gama media-alta. Las gafas de VR que se usarán son unas Valve Index.

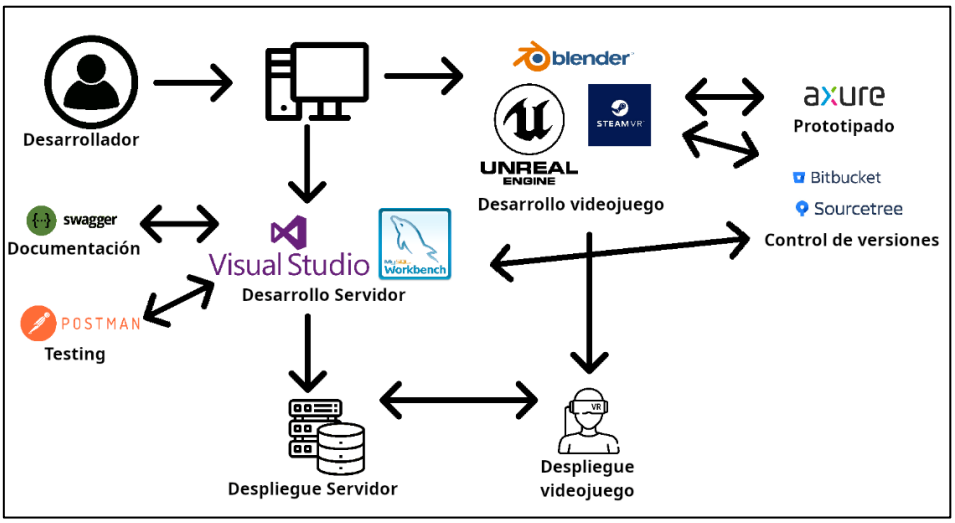

Figura 2.25: Resumen de las herramientas a utilizar y arquitectura hardware y software de la propuesta

3 Metodología y Resultados

En este capítulo se van a describir los aspectos relacionados con la planificación, el diseño y el desarrollo de software. Para finalizar se mostrará una descripción de la funcionalidad del videojuego implementado.

3.1 Metodología de desarrollo de software

Para el desarrollo del video juego se ha utilizado una metodología ágil. Una metodología ágil es un método de desarrollo basado en ciclos de trabajo cortos e iterativos donde se revisa y mejora continuamente el producto. Esto fomenta la entrega frecuente de avances, lo que permite un mejor desarrollo.

El proceso se ha organizado en las siguientes fases:

1. **Documentación:** el proceso de documentación ha sido algo continuo, ya que los requisitos, aunque en su mayoría ya definidos, han ido cambiando conforme ha avanzado el proyecto. Aquí se engloba la investigación del estado de la cuestión, las herramientas disponibles de desarrollo, los videotutoriales para el aprendizaje del uso de Unreal Engine, etc. Se ha hecho uso de posts en foros, publicaciones de revistas online, páginas oficiales de las aplicaciones que se querían usar y videos de YouTube o páginas similares.

2. **Backlog:** el backlog ha sido algo que ha ido creciendo con el tiempo, con cada revisión se ha hecho una planificación del siguiente "*sprint*" donde se han puesto los objetivos a cumplir y en caso de que los hubiere, se ha mantenido los que no se cumplieron del "*sprint*" anterior, siempre intentando mantener los objetivos como algo posible dentro del periodo establecido. Esto ha continuado hasta que se ha dado por cumplido los objetivos del TFG.

3. **Mockups:** los mockups es de las primeras cosas que se ha hecho, aunque también es cierto que se ha ido añadiendo más contenido conforme se avanzaba en el proyecto, como las interfaces, o el logo del juego.

Para la generación de mockups se ha planteado una idea general de lo que se quería conseguir y después de analizarlo, teniendo en cuenta lo aprendido en la carrera y haciendo un poco de investigación si fuese necesario, se ha plasmado un boceto en papel. En algunos casos se ha pasado también a digital más adelante.

4. **Materiales:** el desarrollo de materiales es algo que considero de los aspectos más interesantes de este proyecto, ya que se pueden hacer tan simples o complejos como se pueda o se crea necesario.

   Lo bueno es que si se sabe lo que se hace se pueden crear materiales que emitan luz, que parezcan tener profundidad sin que sea cierto. También se consigue que desaparezcan parcialmente en función de la posición de la cámara, o que al juntar múltiples bloques con el mismo material la textura se vea como algo continuo en vez de que sea algo que se repita como un patrón. Todo ello da una mejor sensación visual.

   Para la implementación se ha seguido múltiples tutoriales de YouTube, y después de varias pruebas y la asistencia de la colaboradora externa, se ha podido crear varios materiales que se usaron tanto para el diseño del nivel de la campaña como para los materiales usados en los bloques de los puzles.

5. **Controles:** para los controles de los mandos se dispone de un código base que viene en el tutorial de RV de Unreal Engine. Para la implementación se ha hecho uso de múltiples videotutoriales y refactorización continua cuando se ha querido añadir funcionalidades nuevas, ya que, al no tener mucha experiencia, no se hizo inicialmente de una manera que permitiese una reutilización adecuada del código, esto es algo que se fue arreglando conforme se avanzaba en el desarrollo.

6. **Interfaces:** para las interfaces se ha hecho uso de los conocimientos adquiridos en el desarrollo web, ya que los elementos de la interfaz de usuario (UI) son presentados y organizados de una manera muy similar. A pesar de ello, sí que se ha encontrado ciertas limitaciones con los

elementos de UI de los que se dispone, pero gracias a múltiples tutoriales y muchas pruebas, se ha podido conseguir todo lo que se pretendía.

7. **Actores:** los actores son todo aquello que está presente en una escena, es decir, en un escenario/nivel. En general podemos decir que el avatar que usa el usuario, el suelo, el punto de iluminación, los bloques y muchos otros son todos actores. Cada uno ha seguido se propio desarrollo, y ha tenido sus problemas a la hora de implementarlo con el resto de las cosas. Para la implementación del avatar se ha tenido que buscar información e implementar el uso de los mandos, la cámara, el movimiento del personaje, las llamadas de funciones al realizar acciones para no sobrecargar la CPU…

   Para los bloques se ha tenido que realizar múltiples revisiones por la colisión de objetos, al igual que la implementación individual del movimiento de algunos bloques, y las funcionalidades de finalización de nivel.

   Para el grid se ha creado uno propio desde cero, dando soporte al cambio de color, tamaño, transparencia, opción de ajuste de rejilla o libre, y la rotación por incremento o libre.

   Las interfaces también se consideran actores, y su uso en un espacio tridimensional ha supuesto un desafío ya que no eran algo que se encontraba en un punto estático del nivel, sino que tenían que ser generados en función de la cámara y el ángulo en el que se encuentra.

8. **Editor de niveles:** en su mayoría el editor de niveles se ha desarrollado a la vez que las interfaces, el grid, y el uso de los bloques ya que sin uno no existe el otro. Para su implementación se ha comenzado con lo básico, crear una interfaz para hacer aparecer los bloques. Luego se ha añadido la funcionalidad de manipular dichos bloques, y se ha creado el grid para colocarlos, seguido del resto de aspectos del grid. Una vez implementado eso se ha añadido la interfaz de texturas para los bloques y se ha terminado dando unos retoques al resto de interfaces del editor.

9. **Comunidad:** cuando se empezó con el apartado de la comunidad ya se había realizado la mayoría del proyecto por lo que se entendía mucho mejor el uso de los elementos de UI y como manipularlos adecuadamente. Esto ha permitido una implementación rápida, aunque compleja, de las interfaces que dan como resultado lo que se verá más adelante.
   Este paso se ha hecho en gran medida a la vez que el desarrollo del servidor, ya que ha sido necesario realizar múltiples pruebas para comprobar que la aplicación estaba preparada para enviar y recibir correctamente los datos del servidor.
   Como se ha mencionado anteriormente, debido a las limitaciones del sistema de nodos, se ha tenido que invertir mucho más tiempo del necesario.

10. **Servidor:** Para el servidor se ha hecho uso de una API RESTful. Como ya había trabajado anteriormente con una implementación similar la estructura general no fue muy complicada de hacer, el único problema ha sido a la hora de introducir los JWTokens, ya que es algo que no había trabajado anteriormente, pero después de leer documentación y ver tutoriales conseguí que funcione adecuadamente.

11. **Demo:** A la hora de producir la demo lo que se hizo fue generar los pasos expuestos del apartado **3.2** y proceder a mostrar los múltiples apartados del juego como se muestra en el apartado **3.3**.

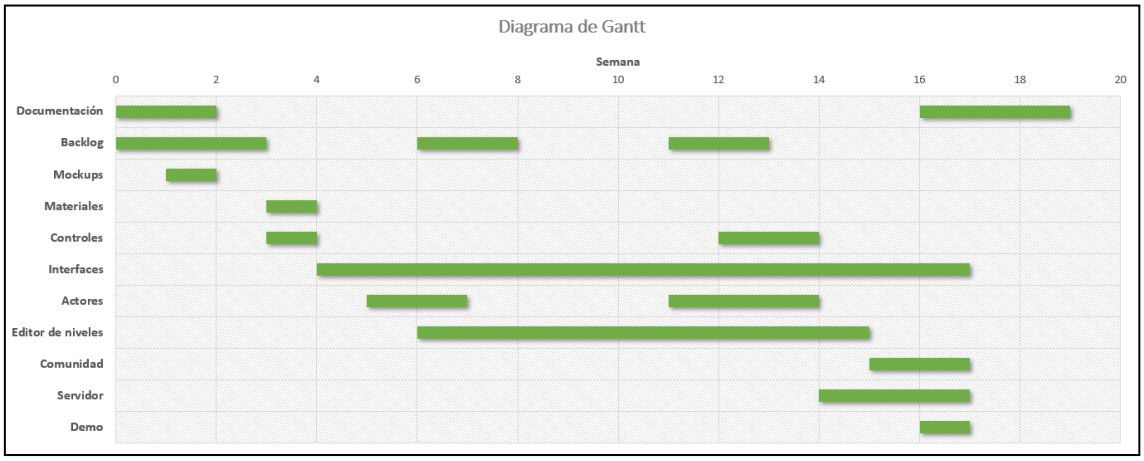

Figura 3.1: Diagrama de Gantt

En la **Figura 3.1** se muestra el diagrama de Gantt resultado de la planificación del proyecto. Además, en la **Figura 3.2** se muestra el uso de Trello.

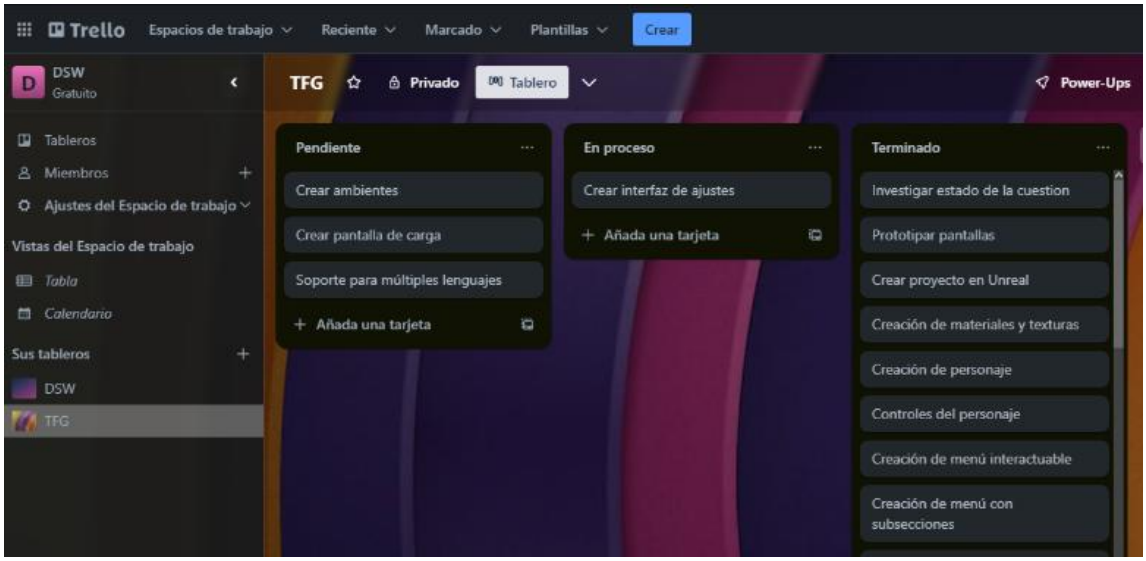

Figura 3.2: Trello

3.1.1 Análisis y diseño de software

En este apartado se resumirán algunos de los resultados consecuencia de la fase de diseño y desarrollo del videojuego.

3.1.1.1 Requisitos funcionales y casos de uso

Dentro del sistema se han definido un total de 8 requisitos funcionales y un requisito no funcional: la accesibilidad. En las Tablas 3-11 se describen cada uno de estos requisitos:

RF – 1	**Jugabilidad**
Descripción	Los usuarios deben poder jugar los niveles creados con los bloques proporcionados.

Tabla 3: Requisito Funcional: Jugabilidad

RF - 2	**Diseño de niveles**
Descripción	Los usuarios deben poder diseñar sus propios niveles, jugarlos, y compartirlos una vez validados. También se permite asignar múltiples etiquetas que describan aspectos del nivel.

Tabla 4: Requisito Funcional: Diseño de niveles

RF - 3	**Cuenta de usuario**
Descripción	Se requieren cuentas de usuario para aquellos que deseen subir niveles. El acceso al videojuego solo estará disponible después de iniciar sesión.

Tabla 5: Requisito Funcional: Cuenta de usuario

RF - 4	**Ratings**
Descripción	Los usuarios deben poder valorar los niveles del apartado de la comunidad.

Tabla 6: Requisito Funcional: Ratings

RF - 5	**Comentarios**
Descripción	Lo usuarios deben poder añadir comentarios a los niveles del apartado de la comunidad

Tabla 7: Requisito Funcional: Comentarios

RF - 6	**Likes**
Descripción	Los usuarios deben poder expresar su preferencia (Me gusta o No me gusta) en cualquier comentario de un nivel de la comunidad.

Tabla 8: Requisito Funcional: Likes

RF - 7	**Calificaciones**
Descripción	Cuando un usuario completa un nivel de la comunidad, el tiempo empleado se registrará y se mostrará en una tabla de resultados, ordenada de menor a mayor tiempo.

Tabla 9: Requisito Funcional: Calificaciones

RF - 8	**Datos**

Descripción	Los usuarios solo pueden modificar datos relacionados consigo mismos. Sin embargo, podrán acceder a listados y datos generales sin necesidad de iniciar sesión (API pública).

Tabla 10: Requisito Funcional: Datos

RNF - 1	**Accesibilidad**
Descripción	Se deben proporcionar opciones para que los usuarios adapten varios aspectos del juego, como colores o controles, según sus necesidades.

Tabla 11: Requisito No Funcional: Accesibilidad

En cuanto a los casos de uso, para este sistema se distinguen dos tipos de actores, los usuarios no registrados (ver **Tabla 12**), que son aquellos que no tienen una cuenta en el sistema, y los usuarios registrados (ver **Tabla 13**), que son aquellos que sí que tienen una cuenta en el sistema.

Nombre	**Usuario no registrado**
Descripción	Usuario que no dispone de una cuenta en el sistema. Lo único a lo que tiene acceso es al apartado de inicio de sesión y al de registro.
Casos de uso	C.U.1, C.U.2

Tabla 12: Rol de Usuario: Usuario no registrado

Nombre	**Usuario registrado**
Descripción	Usuario que sí dispone de una cuenta en el sistema. Este tipo de actor puede acceder a todas las opciones del sistema.
Casos de uso	C.U.1, C.U.2, C.U.3, C.U.4, C.U.5, C.U.6, C.U.7, C.U.8, C.U.9, C.U.10, C.U.11, C.U.12, C.U.13, C.U.14, C.U.15, C.U.16, C.U.17, C.U.18, C.U.19, C.U.20, C.U.21, C.U.22, C.U.23, C.U.24, C.U.25, C.U.26, C.U.27

Tabla 13: Rol de Usuario: Usuario registrado

Finalmente, en la **Figura 3.3** se muestra un diagrama general de los casos de uso, que pueden ser consultados con mayor detalle en el ANEXO I.

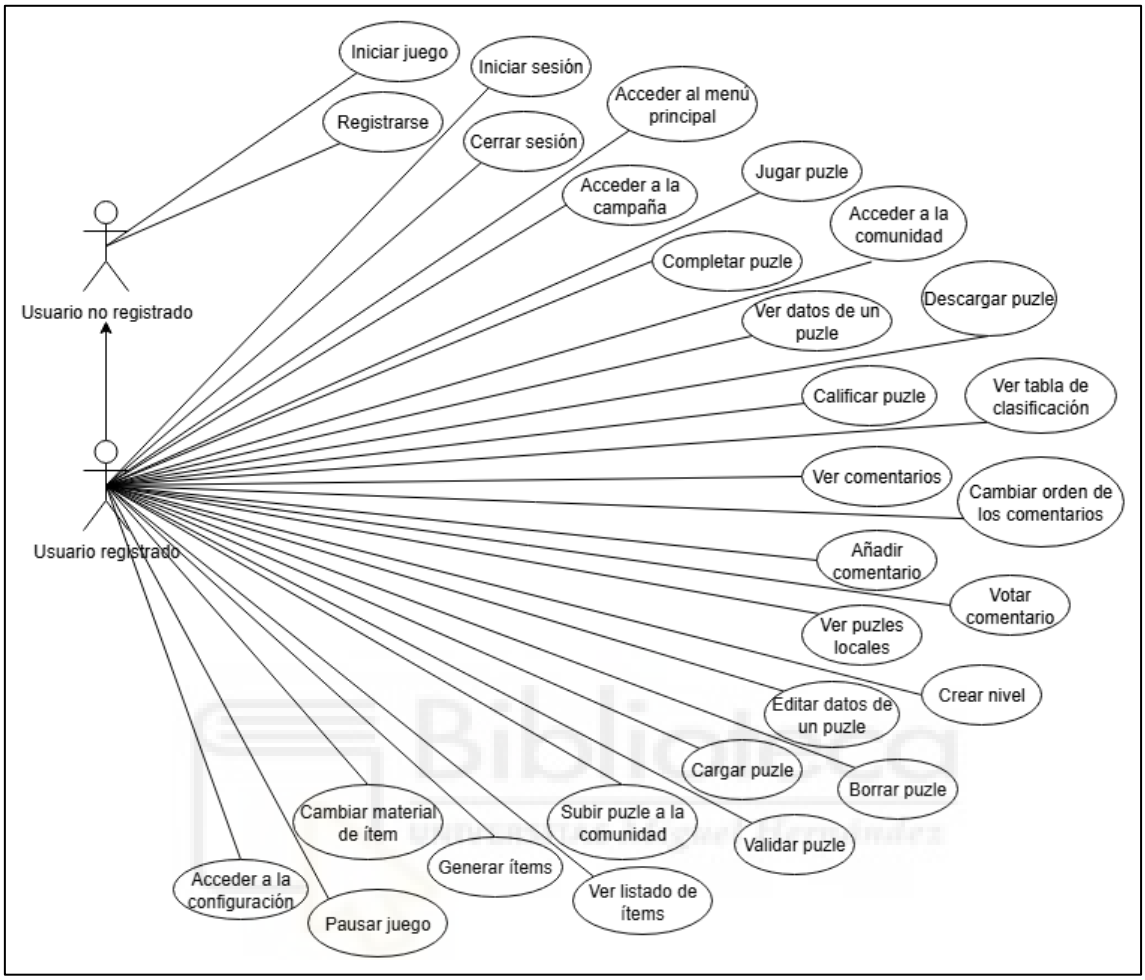

Figura 3.3: Diagrama de casos de uso definidos en el ANEXO I

3.1.1.2 Diagrama del sistema

El juego se ha dividido en 5 subsistemas mostrados en la **Figura 3.4**. A modo de resumen, el subsistema de Autenticación se encargará de gestionar el registro y autenticación de usuarios. El subsistema de Diseño de puzles se encarga de gestionar la creación de los puzles por parte del usuario. El subsistema de Gestión de Puzles se encarga de permitir la administración y manipulación de los puzles creados por el usuario. El subsistema de Completación de Niveles se encarga de gestionar lo relacionado a la finalización de un nivel, ya sea de campaña, comunidad o validación de un nivel. El subsistema de Comunidad se encarga de gestionar los aspectos online, como dar me gusta a un comentario, comentar, puntuar un nivel, etc.

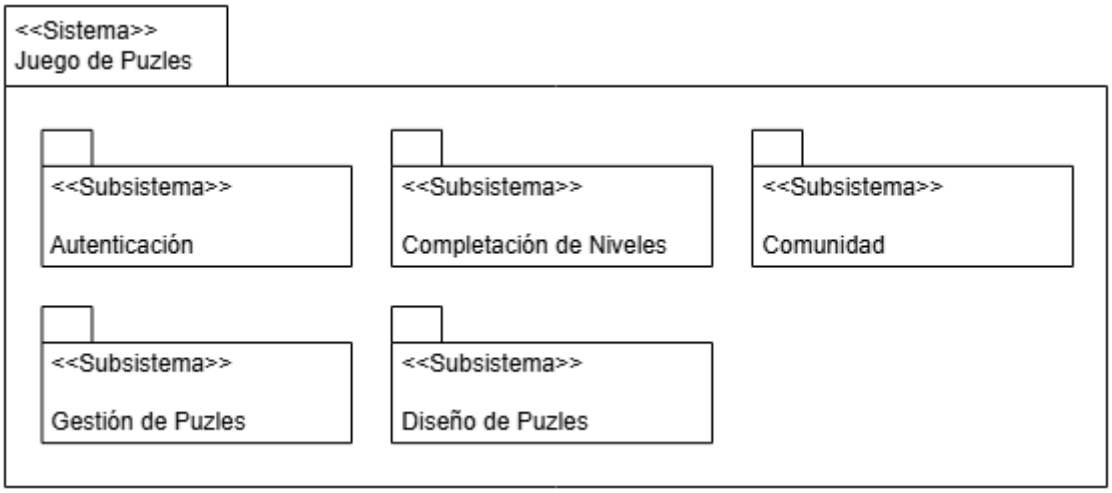

Figura 3.4: Diagrama de sistema

3.1.1.3 Mapa de interfaces del videojuego

A continuación, se presenta un mapa comprensible de la relación entre las interfaces implementadas.

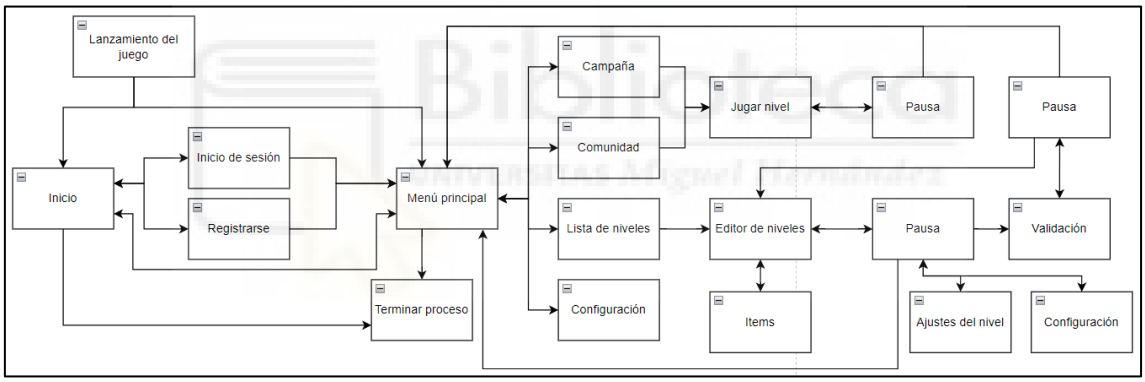

Figura 3.5: Mapa de Interfaces

3.1.1.4 Prototipado

En la fase inicial del desarrollo del videojuego, se han creado prototipos para visualizar las interfaces que estarán presentes en el juego. Estos prototipos abarcan diversas pantallas y elementos que se implementarán posteriormente, como se puede ver en las **Figura 3.6** y **Figura 3.7**.

Como se puede observar, el prototipado se ha realizado usando tanto el papel como la herramienta Axure RP. Merece la pena destacar que en ocasiones los prototipos iniciales han experimentado rediseños. El conjunto completo de los prototipos desarrollados se puede consultar en el **ANEXO II.**

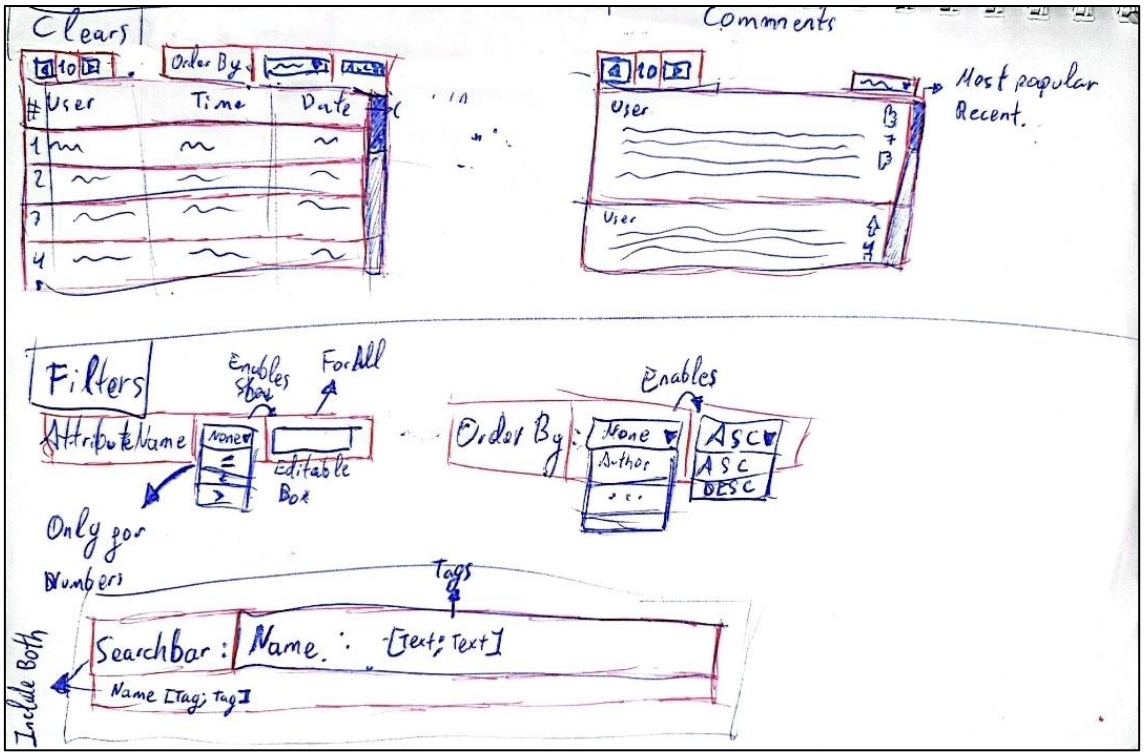

Figura 3.6: Prototipo a papel del menú de la comunidad

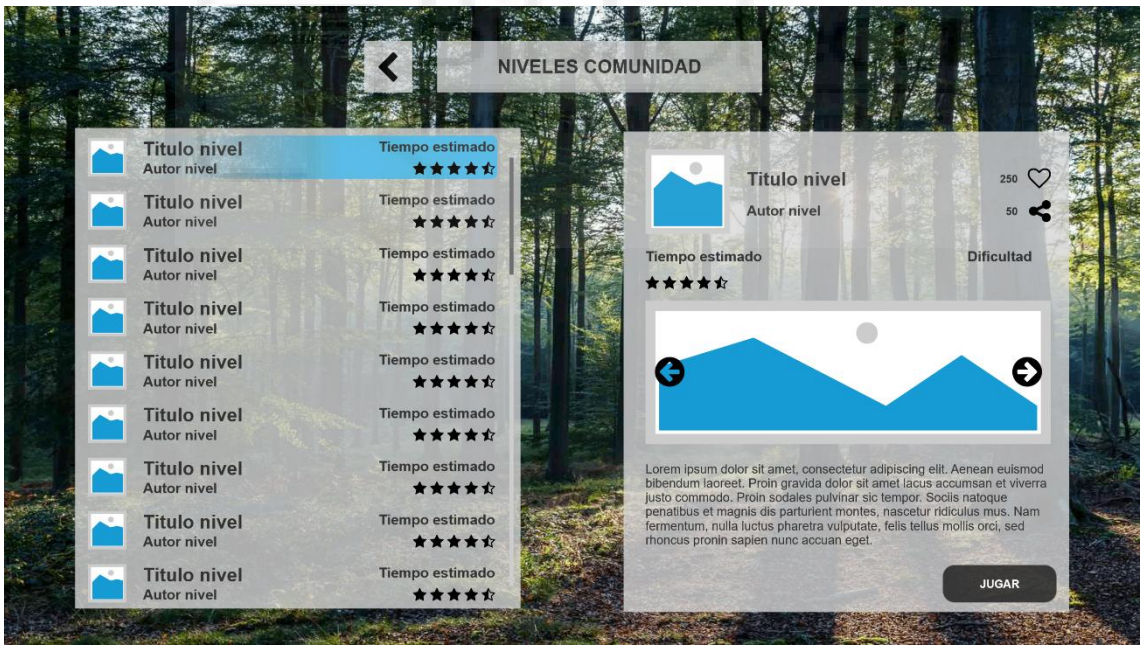

Figura 3.7: Prototipo digital del menú de la comunidad

3.1.2 Desarrollo del front-end (Videojuego de RV)

El diseño e implementación de la solución planteada se dividió en dos partes: el front-end (el videojuego) donde el usuario realiza sus funciones e interactúa con el sistema, y el back-end compuesto por la base de datos donde se almacenan

los datos generados por los usuarios y el servidor que es el sistema por el cual el usuario interactúa con la base de datos. En este apartado destacaremos algunos aspectos relacionados con la implementación del videojuego de RV.

Los videojuegos de VR ofrecen aspectos de gameplay que van más allá de la norma. Sitúan al jugador en un ambiente completamente inmersivo, con una percepción del mundo virtual por encima que cualquier otra plataforma, haciendo que el jugador se sienta transportado a otro mundo.

Muchos juegos de realidad virtual requieren movimiento físico, lo cual lo hace una experiencia única al transmitir los movimientos reales del jugador directamente al mundo virtual, es una interactividad total en todas las dimensiones. Esto, además, ofrece al jugador una movilidad sana mientras disfruta del ocio.

La implementación del videojuego ha comenzado con la exploración de los materiales, escenarios y controles proporcionados en el nivel de ejemplo de Unreal Engine. Este paso ha permitido familiarizarse con los aspectos clave del motor. Se ha analizado también implementaciones existentes como referencia para la construcción del proyecto.

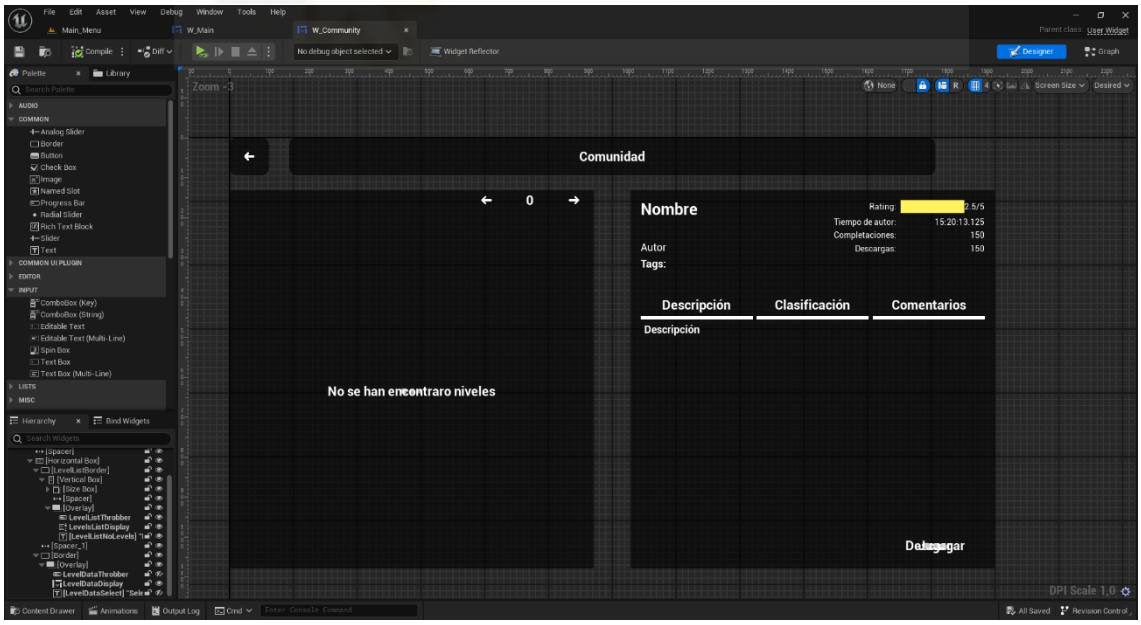

Figura 3.8: Menú de Comunidad, editor de interfaces de Unreal Engine

3.1.2.1 Implementación de interfaces gráficas

Se ha llevado a cabo la implementación de interfaces que previamente se habían prototipado. Por ejemplo, en las **Figura 3.8** y **Figura 3.9** se muestra el resultado de la vista de diseño del menú comunidad, así como su visualización en un entorno de RV. Se puede apreciar la similitud con las figuras del prototipado.

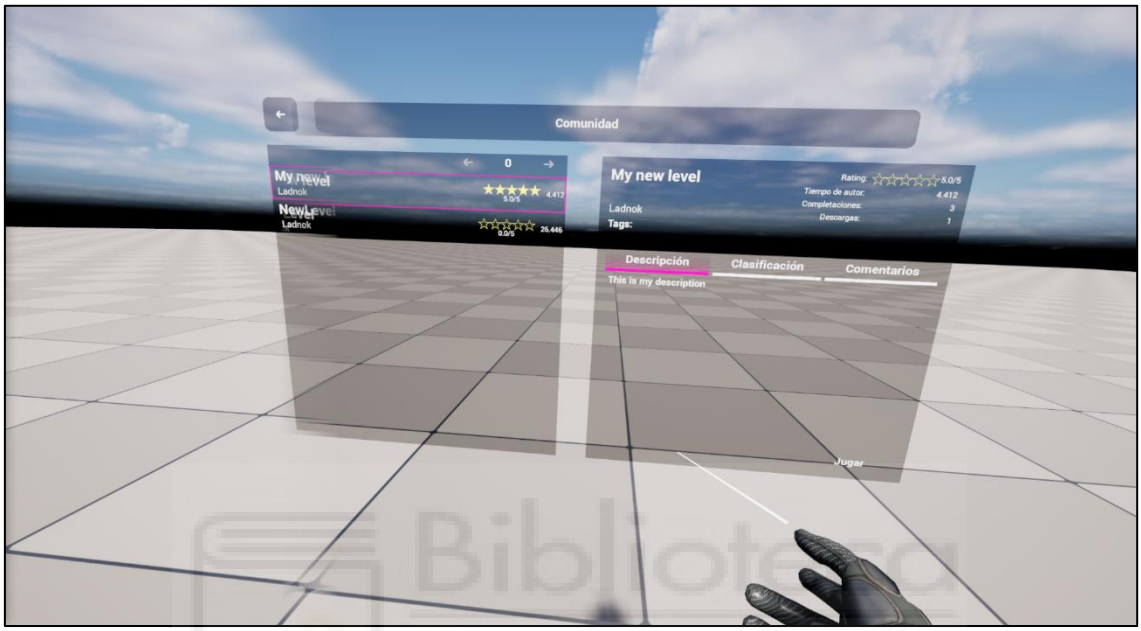

Figura 3.9: Menú de la Comunidad, Datos de un Nivel In Game

3.1.2.2 Diseño de assets

Para el diseño de assets se ha hecho uso principalmente de Maya, esto ha sido en colaboración con una estudiante de la Universidad de La Salle con estudios en Diseño y Animación en 3D. Estos assets han sido todos originales, aunque inspirados en diferentes juegos de puzles jugados a lo largo del tiempo.

El diseño inicial ha sido a papel, y este luego se ha pasado a la compañera que ha hecho la implementación en Maya. Una vez finalizado los assets, se han importado al proyecto y se ha probado si la escala es correcta y si cumplen con su funcionalidad o si hace falta hacer una revisión del objeto. En la **Figura 3.10** se puede apreciar algunos de los bloques dentro de Maya.

Cabe destacar que para algo más simple como una esfera se ha hecho uso de Blender, que cuenta con objetos predeterminados que son fáciles de crear y exportar, algo que se ha podido hacer por cuenta propia, como se puede ver en la **Figura 3.11**.

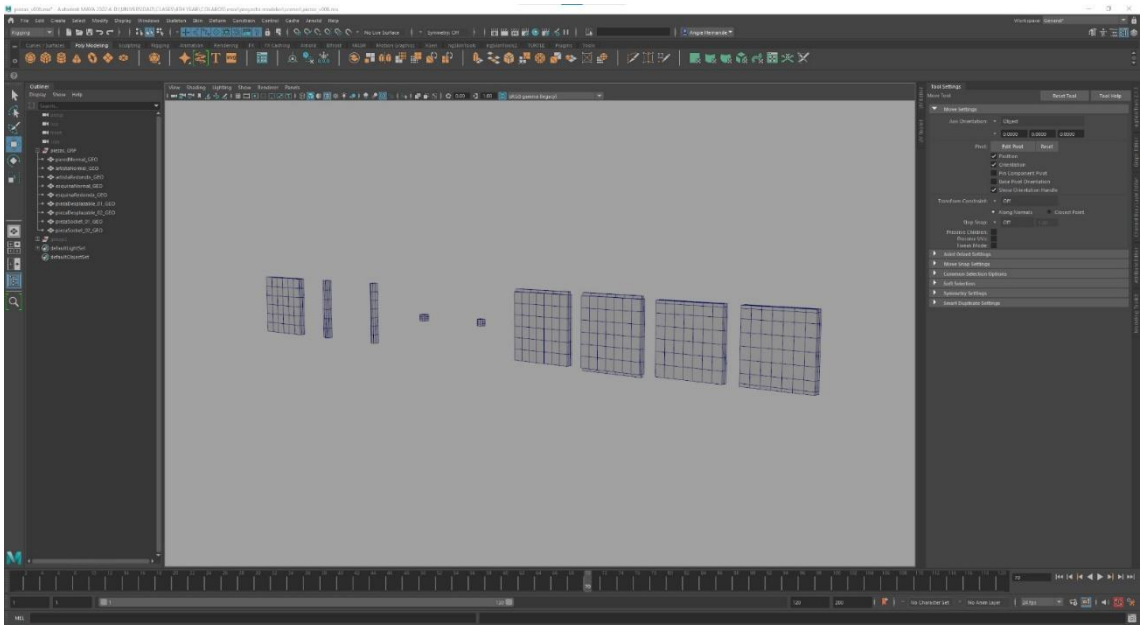

Figura 3.10: Implementación de los bloques estáticos en Maya

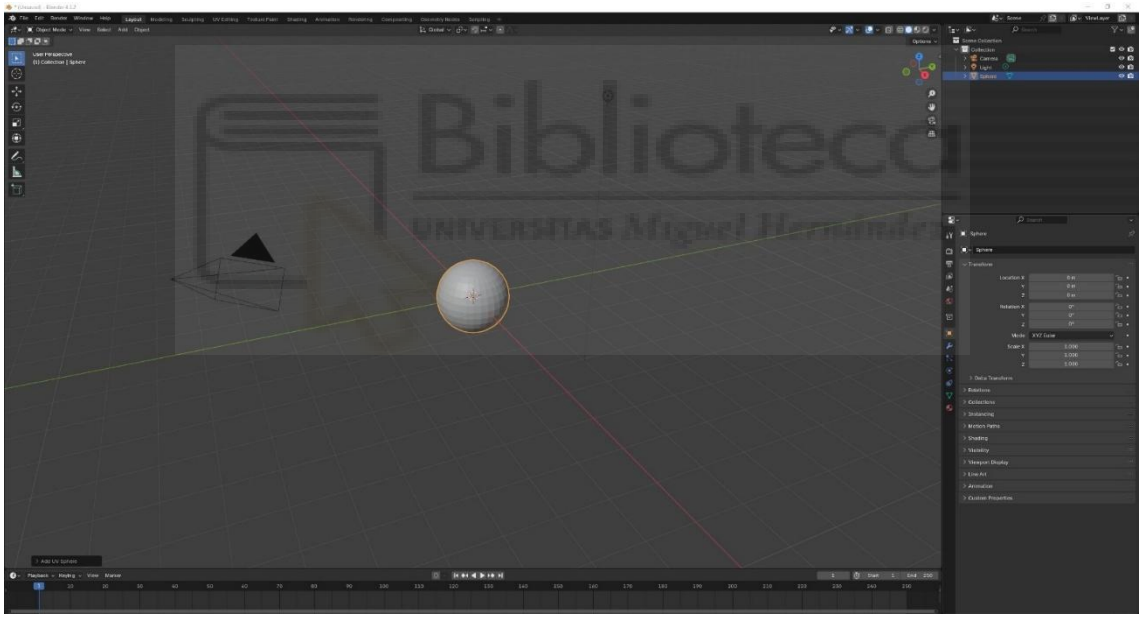

Figura 3.11: Implementación de esfera en Blender

3.1.2.3 Generador de niveles y comunidad

Como se ha comentado anteriormente, uno de los aspectos diferenciales de nuestra propuesta es dotar al usuario de la posibilidad de definir sus propios puzles boxes.

Se ha implementado un generador de niveles que da el poder de creatividad a los jugadores para generar escenarios personalizados. Si quieren algo nuevo, lo

generan ellos sin ningún límite de creatividad, con miles de posibles puzles al alcance de su mano. Los puzles pueden ser tan complejos o simples como ellos busquen, por lo cual no existe casi ninguna barrera de dificultad para ningún jugador.

Podría decirse que la comunidad de un juego es el factor más importante para su prosperidad. Cuando la comunidad es activa, los usuarios se motivan entre ellos, surgen más ideas, y hay más interactividad con el juego. El juego prospera gracias a una base de jugadores fieles que crean juntos, juegan juntos y entablan relaciones, sintiéndose parte de algo importante, una comunidad.

Para el desarrollo del editor de niveles, se ha optado por la creación de un grid visual que define la zona de juego. Se han implementado elementos visuales para indicar la posibilidad de colocar o eliminar bloques, y se ha finalizado incorporando las funcionalidades específicas de cada bloque. Por ejemplo, en la **Figura 3.12** se pueden apreciar bloques.

El inferior de todos es un bloque estático, su función es la de una pared, con ella se pueden construir diferentes formar que limitan la visibilidad o el espacio disponible del puzle, como se puede apreciar en la **Figura 3.14**, que se trata de un puzle hecho enteramente con este bloque y un material de cristal aplicado.

La esfera dorada es un bloque de finalización, se trata de un bloque que se ve afectado por las físicas del juego por lo que, si se encuentra en presencia de una pendiente, por ejemplo, esta esfera rodara por ella. Su función principal es la de “terminar” el nivel, es decir, cuando el usuario agarra dicho bloque, se considera el nivel como completado. Todos los niveles requieren de al menos un bloque de finalización para poder ser considerados como válidos.

Por último, el bloque que tiene el usuario en la mano se trata de un bloque móvil. Su función principal es deslizarse en un plano horizontal siempre y cuando se cumplan unas condiciones, como por ejemplo que este en contacto con otro bloque. Esto permite que si se libera el bloque del resto este puede ser “descartado”. También hace función de pared, ya que el resto de los bloques, móviles o de finalización, pueden interactuar con este.

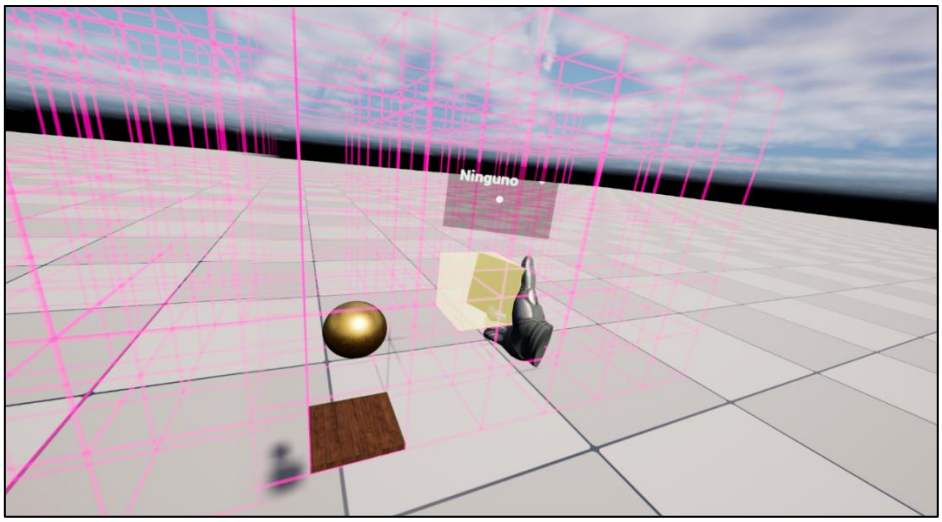

Figura 3.12: Usuario Colocando Bloque en el Grid

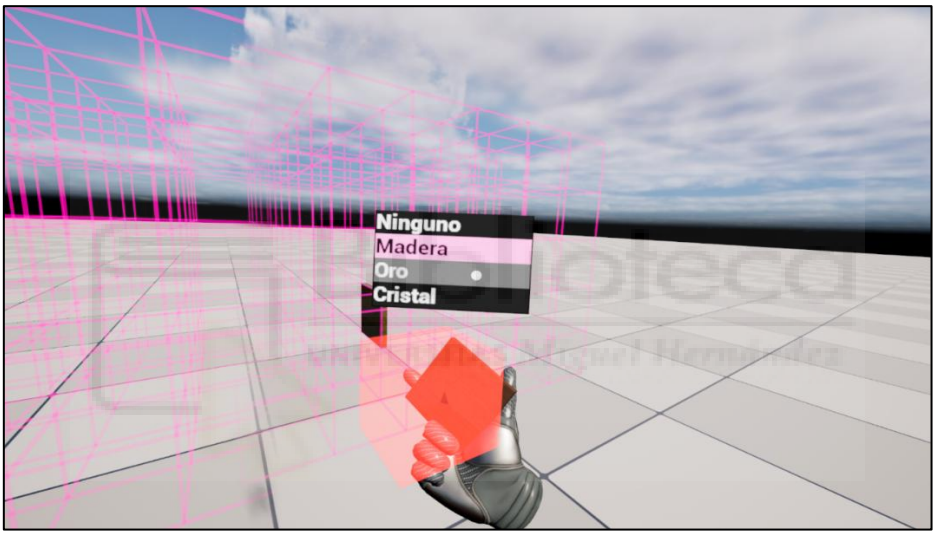

Figura 3.13: Usuario Editando Material de un Bloque

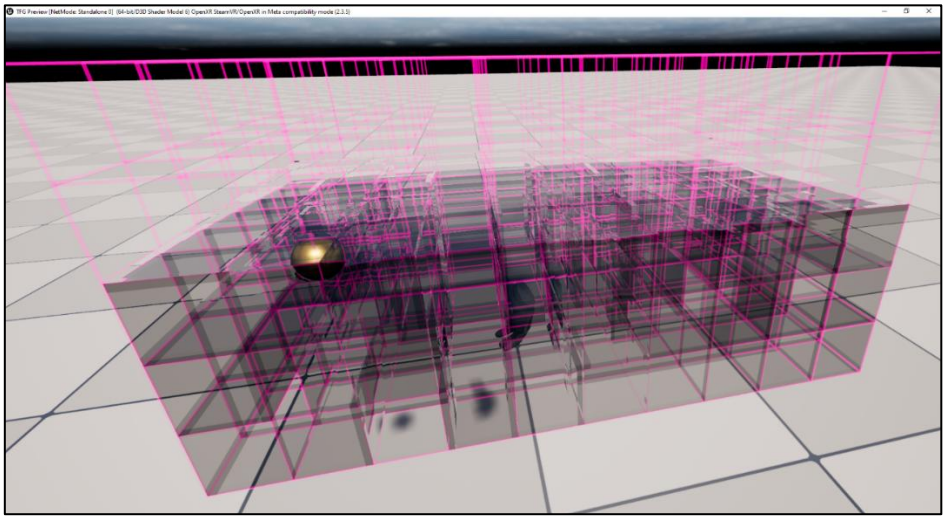

Figura 3.14: Ejemplo de Puzle Completado en el Editor

3.1.3 Desarrollo del back-end (API RESTful y BBDD)

3.1.3.1 Diseño e implementación de la base de datos

En el proceso de diseño de la base de datos ha sido incremental al requerir de algunos ajustes para cumplir con los requisitos. En la **Figura 3.15** se muestra la versión final de la base de datos que explicaremos a continuación.

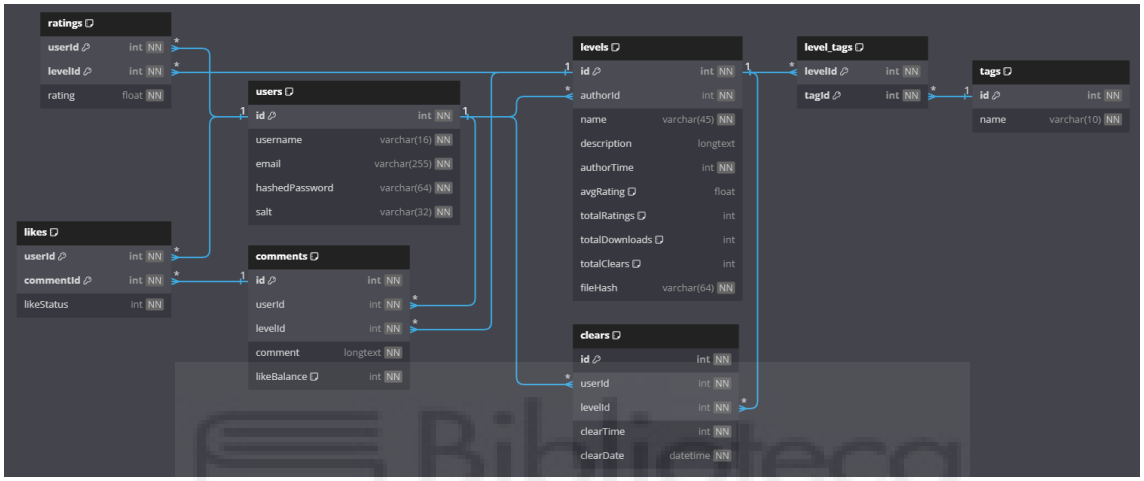

Figura 3.15: Modelo de la base de datos usada en el videojuego

En primer lugar, cada valor en la base de datos es NOT NULL, lo que significa que debe tener un valor cuando se inserta en la tabla. La única excepción es para algunos datos de la tabla Levels. Para el caso de la descripción, es por preferencia, mientras que para el resto es debido a que tendrán un valor predeterminado de 0.

Además, las eliminaciones en la base de datos se manejan en cascada. Esto implica que, si se elimina un usuario, todos sus niveles, comentarios, likes, ratings, etc., serán eliminados también, al igual que con cualquier tabla relacionada. La única excepción es la tabla Tags, ya que debido a su naturaleza de almacenar todos los tags creados por los usuarios estos podrían estar asociados a otro nivel.

Otro aspecto que destacar es la decisión de romper las formas normales con las tablas niveles y comentarios. Se almacenaron directamente valores como la media de calificación y el balance de likes, al igual que contadores. Esto se hizo

para mejorar la eficiencia y reducir el uso de recursos, especialmente a medida que la base de datos crece con una gran cantidad de datos.

3.1.3.1.1 Listado de tablas

A continuación, se listan y explica el contenido de cada una de las tablas contenidas en la base de datos desarrollada:

1. **Users:** tabla principal de la base de datos donde se almacenan los datos del usuario. Incluye id, nombre, correo electrónico, contraseña encriptada y la salt utilizada como parte de la encriptación de la contraseña. Tanto el nombre de usuario como el correo electrónico deben ser únicos.

2. **Levels:** almacena los datos de los niveles subidos por los usuarios. Incluye id, id del autor, nombre, descripción, tiempo que ha tardado el autor en validar el nivel, media de la calificación, calificaciones totales, descargas totales, completaciones totales y el hash del fichero del nivel. El hash se utiliza para verificar la integridad del nivel cuando un usuario lo juega. Todos los niveles de un usuario han de tener nombres distintos.

3. **Tags:** almacena todos los tags creados por los usuarios. No se pueden repetir nunca, y las entradas nuevas no se eliminan.

4. **Level_Tags:** asocia tags a niveles.

5. **Clears:** almacena todas las completaciones de los niveles. Incluye id, id del usuario, id del nivel completado, tiempo de completación y el día de completación.

6. **Ratings: a**lmacena las calificaciones de los usuarios a los niveles. Incluye el id del usuario, id del nivel y el rating, que va desde 0 a 5 en intervalos de 0.5. Se aplica un algoritmo simple para actualizar la media en la tabla de niveles cuando se cambia el rating.

7. **Comments:** almacena los comentarios de los usuarios a los niveles. Incluye id, id del usuario, id del nivel, el texto y el balance de likes, que puede ser positivo o negativo.

8. **Likes:** almacena los me gusta que reciban los comentarios. Incluye el id de usuario, id de comentario y el estado del like, que puede ser -1, 0 o 1. Se actualiza el balance en la tabla de comentarios al cambiar el like.

3.1.3.1.2 Uso de MySQL Workbench

Se ha empleado MySQL Workbench como herramienta que proporciona una interfaz efectiva para gestionar la información contenida en la base de datos. Por ejemplo, en la parte izquierda de la **Figura 3.16** se puede ver el esquema implementado, y en la parte derecha la interfaz de consultas que hemos utilizado durante la fase de desarrollo.

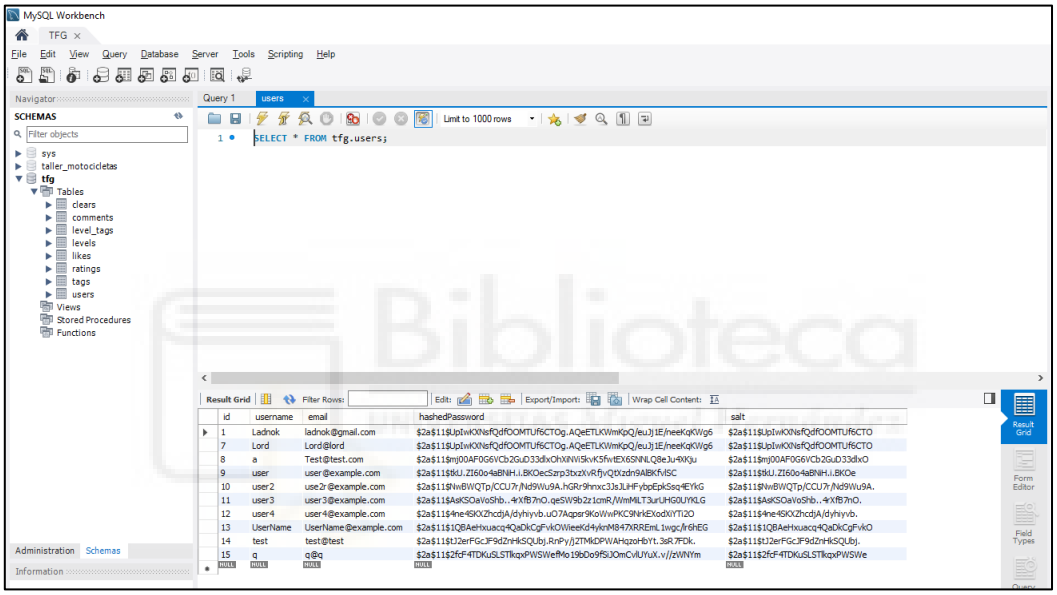

Figura 3.16: Ejemplo de Consulta en MySQL Workbench

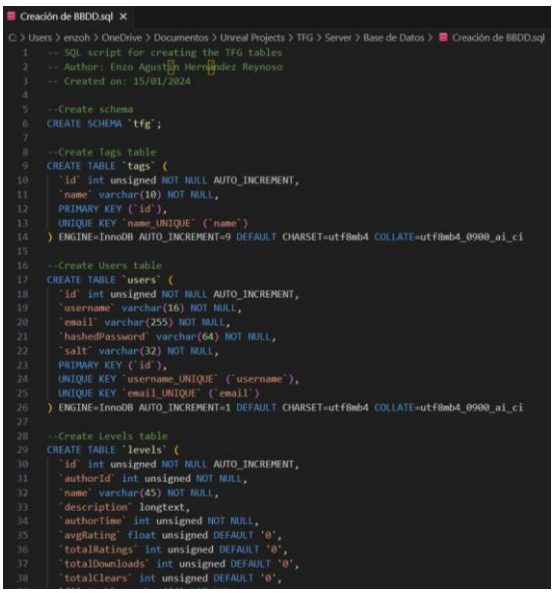

Figura 3.17: Fichero de Creación de la Base de Datos

3.1.3.1.3 Script SQL de creación de la BBDD

Finalmente, a partir de la interfaz gráfica se ha creado un fichero (ver **Figura 3.17**) que contiene la información necesaria para el esquema y todas las tablas con sus relaciones y restricciones.

3.1.4 Diseño e implementación de la API RESTful

Para la arquitectura del servidor, se plantea el uso de una API RESTful que se divida en múltiples capas:

1. **Controladores:** en esta capa se definen los endpoints, es decir, la URL a la que hay que enviar la petición HTTP. Se encargan de la lógica como autenticación, procesamiento de datos, envío de respuestas, entre otros.
2. **Repositorios:** en esta capa se realiza la lógica de datos, tratando directamente con la base de datos.
3. **Modelos:** en esta capa se definen las entidades y DTOs utilizados en el servidor para la recepción de peticiones y el envío de respuestas.

3.1.4.1 Uso de plantilla base

Para la implementación del servidor, se ha partido de una plantilla proporcionada por Visual Studio para ASP.NET Core Web API. A partir de esta plantilla, se han realizado diversas modificaciones como la conexión con la base de datos, la implementación de JWT para la autenticación y la subdivisión del proyecto en las partes mencionadas durante el diseño.

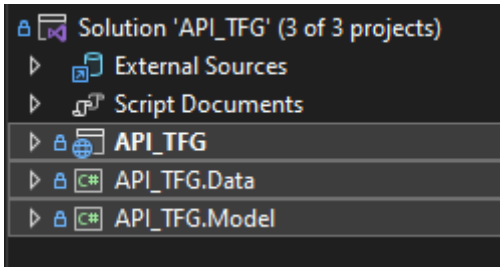

Figura 3.18: Estructura de la Solución en VS

3.1.4.2 Controladores

En este apartado se ha creado casi un controlador por cada tabla de la base de datos, y también se han incluido DTOs que derivan de las Entidades definidas más adelante (ver **Figura 3.19**). La razón es que en algunos casos se requiere

de datos de múltiples tablas, o solo datos parciales de las Entidades, lo que facilita la comprobación de que los datos recibidos cumplen con el modelo esperado.

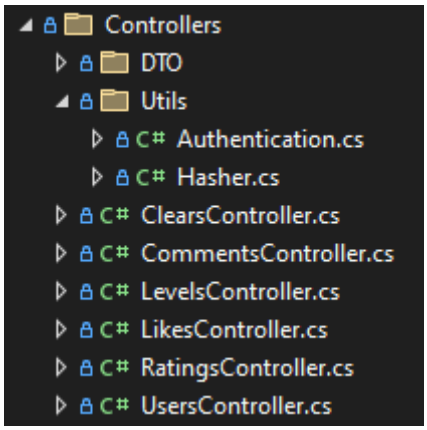

Figura 3.19: Estructura del apartado de los Controladores

Todos los controladores siguen una estructura similar. En primer lugar, se importan las librerías necesarias, se asigna el nombre de la ruta base, se especifica que se trata de un controlador de tipo API y, por último, se definen las referencias a la configuración del servidor y los repositorios que se utilizarán en el controlador. Todo esto se puede ver en la **Figura 3.20**.

```
namespace API_TFG.Controllers
{
    [Route("api/[controller]")]
    [ApiController]
    public class UsersController : ControllerBase
    {
        private readonly IUserRepository _userRepository;
        private readonly IConfiguration _configuration;

        public UsersController(IUserRepository userRepository, IConfiguration configuration)
        {
            _userRepository = userRepository;
            _configuration = configuration;
        }
```

Figura 3.20: Configuración Común de los Controladores

Luego se definen las rutas asíncronas del controlador. Se determina el tipo de petición (GET, POST, PUT, DELETE), se especifica el nombre completo de la ruta y, finalmente, se indican los parámetros que se encuentran en el cuerpo de la solicitud o en la ruta misma.

Dentro del método, se aplica la lógica necesaria, como la validación de los parámetros recibidos o las llamadas a los repositorios. Finalmente, se construye una respuesta que se devuelve con el código HTTP apropiado. Esto se puede ver con el ejemplo del endpoint de inicio de sesión de la **Figura 3.21**.

```
[HttpPost("LogIn")]
public async Task<IActionResult> LogIn([FromBody] LogInRequest user)
{
    if (user == null)
        return BadRequest();
    if (!ModelState.IsValid)
        return BadRequest();

    Users? DBUser = await _userRepository.GetUserByName(user.UserName);
    if (DBUser != null && Authentication.Authenticate(user, DBUser))
    {
        LogInResponse response = new(Authentication.GenerateToken(DBUser, _configuration), DBUser.Id, DBUser.Username);

        return Ok(response);
    }

    return BadRequest(new LogInErrorResponse("Wrong credentials"));
}
```

Figura 3.21: Ejemplo de Endpoint en Controlador

3.1.4.3 URL de acceso a la API y documentación con Swagger

A continuación, se proporciona una visualización de todas las rutas creadas, separadas por controlador, junto con la documentación de una de las rutas generada por Swagger, y los esquemas utilizados. Por ejemplo, en la **Figura 3.22** se puede observar todos los endpoints del controlador de comentarios, en total se ha creado un GET, dos POST, un PUT y un DELETE.

En la **Figura 3.23** se puede ver más detalles de la información que se espera recibir en el endpoint /api/levels/list, que forma parte del controlador de niveles. En específico se esperan datos de paginación, como página y tamaño de página, una lista de filtros de búsqueda, y por último el orden de la lista devuelta.

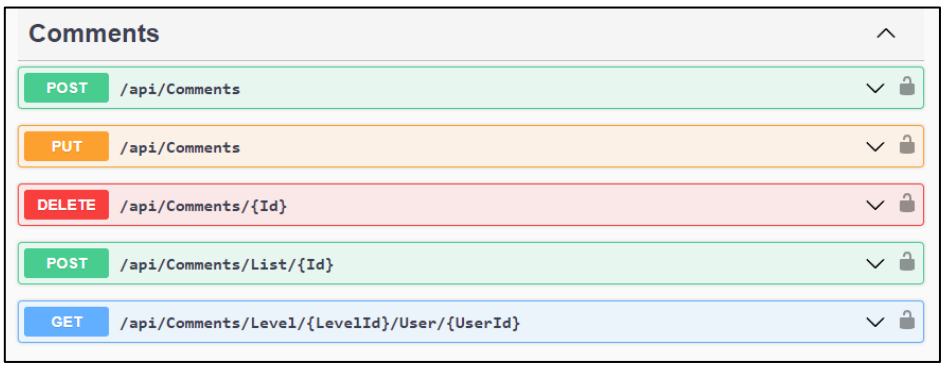

Figura 3.22: Rutas del Controlador "Comments"

Por último, en la **Figura 3.24** se puede ver el schema del ListerDTO, que se trata de un DTO que se usa para determinar qué y cómo devolver un listado de algo. Y dentro de este DTO, en *"Pagination"*, se puede ver que se espera que el atributo *"page"* sea un numero entero.

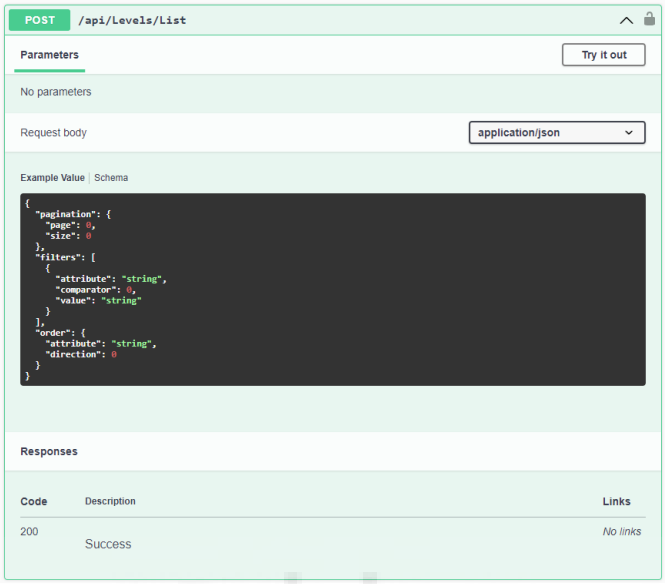

Figura 3.23: Ejemplo de Documentación Generada por Swagger

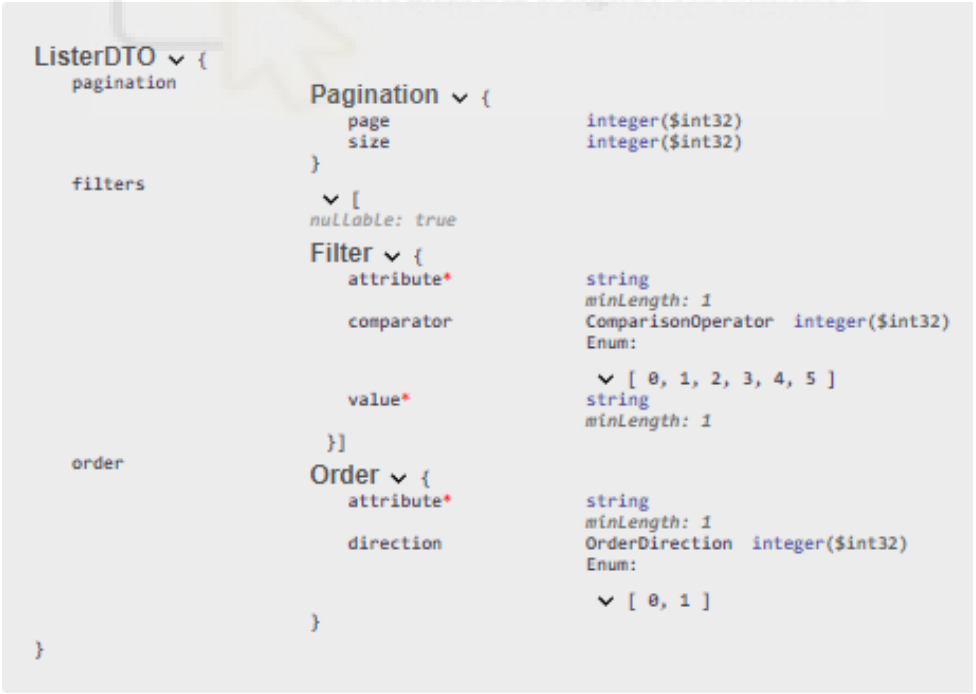

Figura 3.24: Ejemplo de Schema

3.1.4.4 Repositorios

En la sección de repositorios, se ha organizado un archivo por cada tabla de la base de datos, dividiendo cada uno en una interfaz e implementación correspondiente (ver **Figura 3.25**).

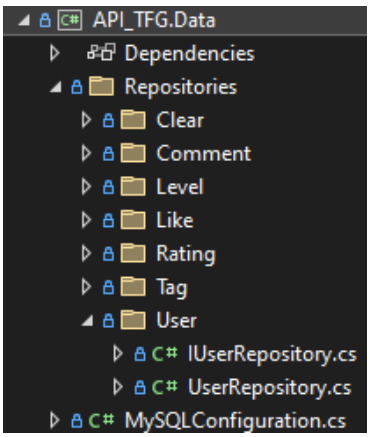

Figura 3.25: Estructura del Apartado de los Repositorios

Al igual que en los controladores, todos los repositorios siguen una estructura similar. En primer lugar, en la interfaz se definen todos los métodos a implementar junto con las entidades importadas, así como los datos que recibe cada método y el resultado que devuelve (ver **Figura 3.26**).

```
using API_TFG.Model.Entities;

namespace API_TFG.Data.Repositories.User
{
    public interface IUserRepository
    {
        Task<Users> GetUserById(uint id);

        Task<Users?> GetUserByName(string userName);

        Task<Users?> GetUserByEmail(string email);

        Task<bool> ExistsByName(string userName);

        Task<bool> ExistsByEmail(string email);

        Task<bool> RegisterUser(Users users);
    }
}
```

Figura 3.26: Ejemplo de Interfaz de Repositorio

En segundo lugar, en la implementación se declara que interfaz que se está implementando, se define la cadena de conexión con la base de datos, que está especificada en el archivo de configuración del servidor, y se crea un método que devuelve una nueva conexión a MySQL (ver **Figura 3.27**).

Finalmente, se implementan los métodos definidos en la interfaz. Se obtiene una conexión con la base de datos, se define la consulta a ejecutar y se lleva a cabo la ejecución de manera asíncrona. Es relevante destacar el uso de parametrización, que previene los ataques de inyección SQL (ver **Figura 3.28**).

```
using API_TFG.Model.Entities;
using Dapper;
using MySql.Data.MySqlClient;

namespace API_TFG.Data.Repositories.User
{
    public class UserRepository : IUserRepository
    {
        private readonly MySQLConfiguration _connectionString;
        public UserRepository(MySQLConfiguration connectionString)
        {
            _connectionString = connectionString;
        }

        protected MySqlConnection DbConnection()
        {
            return new MySqlConnection(_connectionString.ConnectionString);
        }
```

Figura 3.27: Configuración común de los Repositorios

```
public async Task<bool> RegisterUser(Users user)
{
    var db = DbConnection();

    var query = @"INSERT INTO users(username, email, hashedPassword, salt, isAdmin)
                  VALUES(@Username, @Email, @HashedPassword, @Salt, @IsAdmin)";

    var result = await db.ExecuteAsync(query, new
        { user.Username, user.Email, user.HashedPassword, user.Salt, user.IsAdmin });

    return result > 0;
}
```

Figura 3.28: Ejemplo de Método de un Repositorio

3.1.4.5 Modelos

En esta sección se definen las entidades, que se utilizan para asegurar que los datos y su formato son correctos. En su mayoría, estas entidades coinciden con las tablas de la base de datos. Sin embargo, es posible heredar de estas entidades para crear modelos más complejos según sea necesario, como se hace en algunos métodos de los repositorios o en los controladores. Por ejemplo, en la **Figura 3.29** se puede ver la estructura de las carpetas, y la entidad de niveles, que corresponde uno a uno con la tabla de la Base de Datos (BBDD).

También, en la **Figura 3.30** se puede ver un ejemplo del DTO usado para el registro de una cuenta, algo destacable es el uso de *DataAnnotations* para poner restricciones de manera sencilla, como si son requeridos o no, longitud mínima y/o máxima, patrones de cadenas de texto, o incluso anotaciones creadas personalizadas.

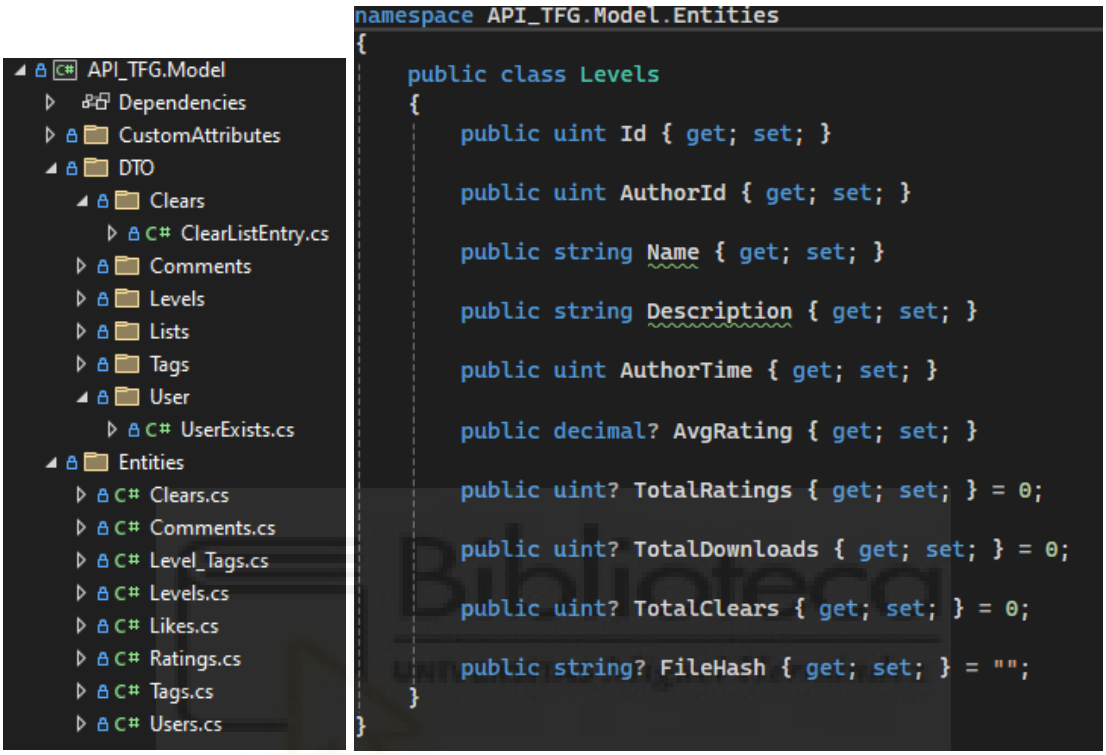

Figura 3.29: Estructura del apartado de modelos y ejemplo de entidad

```
using System.ComponentModel.DataAnnotations;

namespace API_TFG.Controllers.DTO.User
{
    public class RegisterRequest
    {
        [Required]
        [StringLength(16)]
        public string UserName { get; set; }

        [Required]
        [EmailAddress]
        [StringLength(255)]
        public string Email { get; set; }

        [Required]
        [RegularExpression(@"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]{8,25}")]
        public string Password { get; set; }
    }
}
```

Figura 3.30: Ejemplo de DTO

3.2 Implantación y despliegue

La implantación del proyecto se divide en dos partes fundamentales: la del servidor y la del cliente.

3.2.1 Despliegue del servidor

La implantación del servidor es bastante sencilla, ya que solo se necesita un sistema de hosting para la API. En este caso, se utiliza el ordenador personal, y se aprovecha la red local para llevar a cabo todas las pruebas. Para lanzar el servidor, basta con seleccionar el perfil deseado en Visual Studio y hacer clic en "Run". Esto iniciará el proceso y mostrará algunas herramientas de diagnóstico.

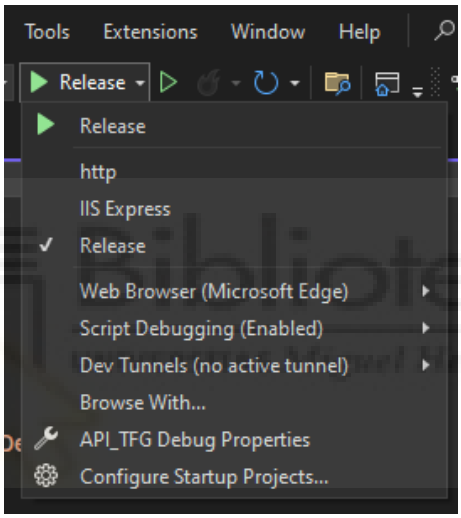

Figura 3.31: Lanzamiento del Servidor

3.2.2 Despliegue del videojuego de RV

La parte del cliente presenta dos opciones para el desarrollador: una para realizar pruebas durante el desarrollo y otra para el lanzamiento final.

3.2.2.1 Entorno de desarrollo

Durante el desarrollo del videojuego, se puede iniciar el juego de manera sencilla utilizando el botón proporcionado por Unreal Engine. Este botón se encarga de compilar y construir los niveles, asegurándose de que no haya errores, y una vez completado, inicia el proceso del juego.

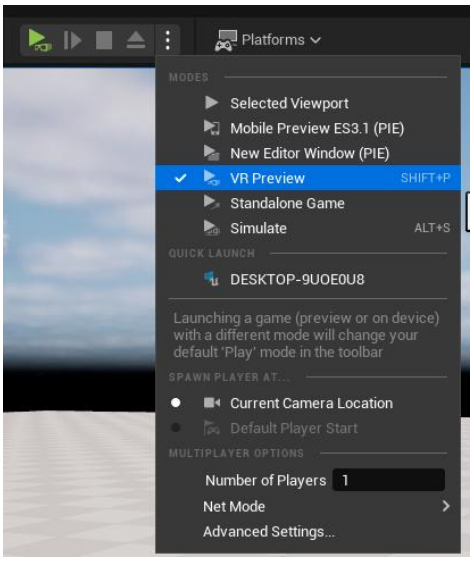

Figura 3.32: Lanzamiento del Videojuego en Desarrollo

Adicionalmente, se empleó Postman para realizar pruebas iniciales de la API, creando una suite de llamadas con valores específicos para validar el correcto funcionamiento de cada ruta. Por ejemplo, en la **Figura 3.33** se muestra un ejemplo del uso de Postman para hacer uso del API y obtener el listado de puzles disponibles.

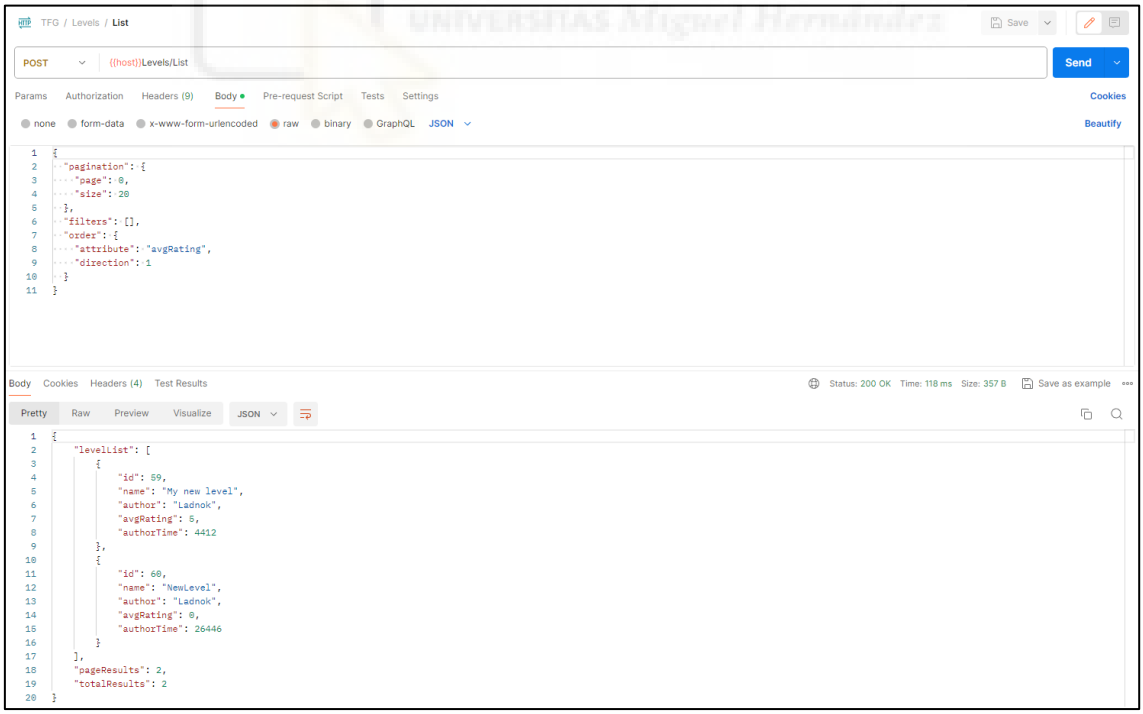

Figura 3.33: Obtención de Listado de Puzles en Postman

3.2.3 Producción

Para la fase de producción, el primer paso es instalar los paquetes de desarrollo de juegos en C++ dentro de Visual Studio (ver **Figura 3.34**).

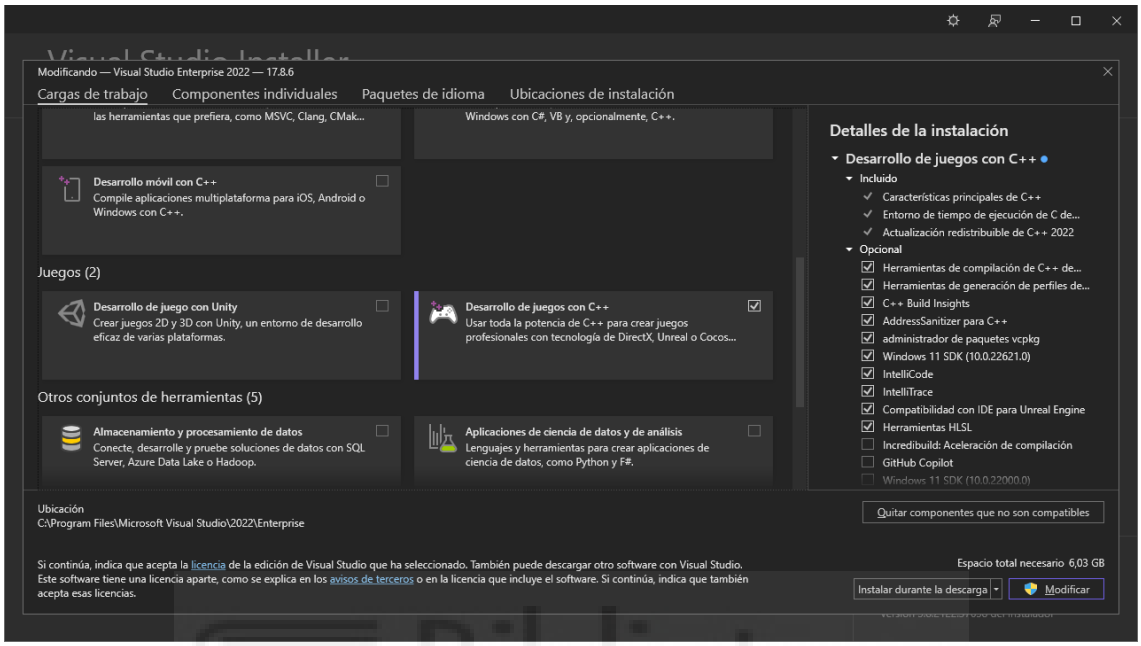

Figura 3.34: Instalación de Paquetes de Desarrollo de Juegos con C++

Una vez instalados, se puede proceder al empaquetamiento del juego (**Figura 3.35** y **Figura 3.36**). La primera vez puede tardar varias horas, mientras que las siguientes veces se reduce a algunos minutos. Una vez completado, se obtiene el ejecutable que puede ser utilizado por cualquier persona (**Figura 3.37**).

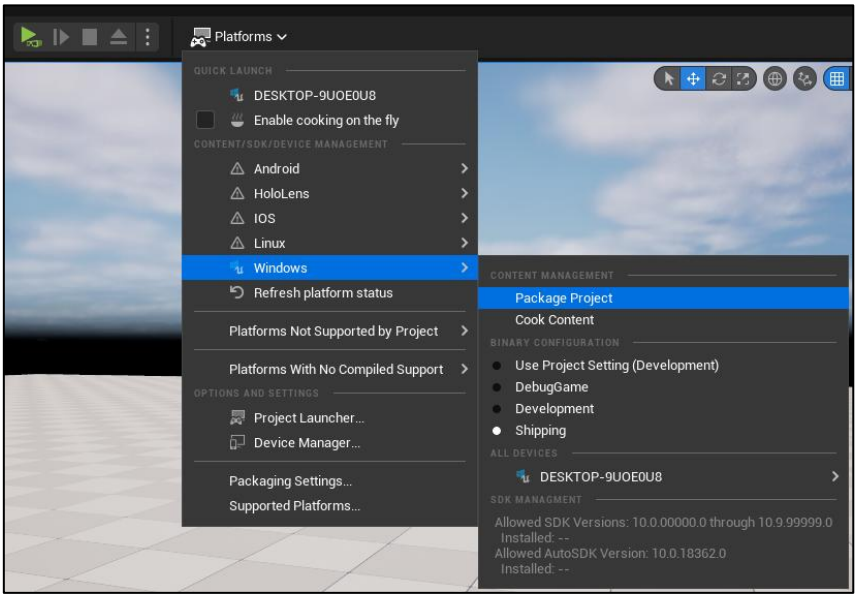

Figura 3.35: Empaquetamiento del Videojuego para Windows

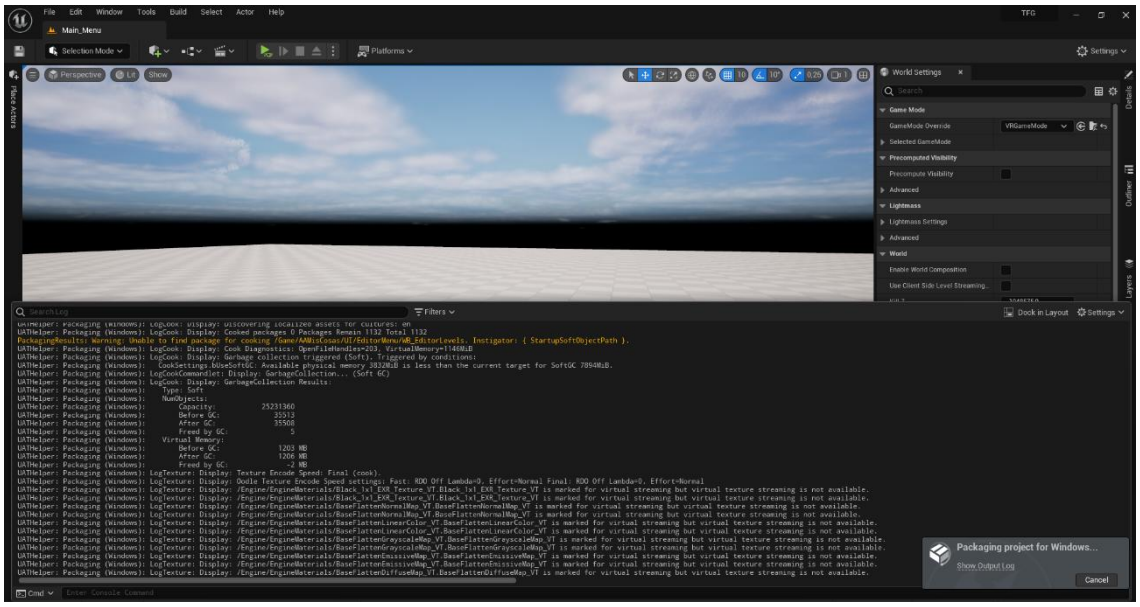

Figura 3.36: Unreal Engine Empaquetando el Videojuego

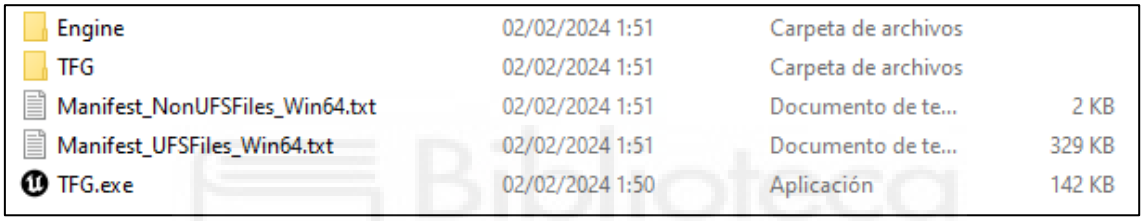

Figura 3.37: Videojuego Empaquetado

3.2.4 Ejecución del videojuego en unas Valve Index

Una vez se tiene el juego empaquetado el proceso de lanzamiento es tan sencillo como hacer doble clic en el ejecutable, en este caso TFG.exe, como se puede ver en la **Figura 3.37**. Dado que se ha hecho uso de las gafas de Valve Index, se tiene acceso al entorno VR de Steam, lo que permite lanzar el juego desde ahí, como se puede ver en la **Figura 3.38**.

Figura 3.38: Entorno virtual de Steam

Aunque esto no es requerido ya que, si se lanza desde Unreal Engine, el juego se abrirá de todas maneras detectando automáticamente las gafas de VR siempre que estas estén encendidas y configuradas adecuadamente. Un ejemplo de una prueba realizada puede ser observada en la **Figura 3.39**.

Como se ha mencionado anteriormente, el ordenador usado para el desarrollo y las pruebas es un XMG CORE 15, que consta de una gráfica GeForce RTX 2060, procesador AMD Ryzen 7 4800H, 2x8GB de RAM DDR4-3200 Crucial y una pantalla de 15.6'' Full HD de 144 Hz.

Por otro lado, las gafas usadas son las Valve Index, que consta de dos pantallas LCD de 1440x1600, un refresco de 80/90/120/144 Hz, además de eso permite el ajuste de los lentes ópticos, cuenta con un soporte intercambiable que absorbe el sudor, un soporte trasero y auriculares estéreo.

Figura 3.39: Foto de una prueba del juego

3.3 Ejemplo de uso de la aplicación

En este apartado se muestra un ejemplo de uso del videojuego desarrollado. Debido a la naturaleza de la RV es difícil mostrar todas las pantallas con claridad, por lo que se dispone de una versión limpia junto con otras pantallas y detalles.

3.3.1 Creación de cuenta e inicio de sesión

La primera vez que el usuario accede al juego se encontrará con las opciones de crear una cuenta o iniciar sesión (ver **Figura 3.40**).

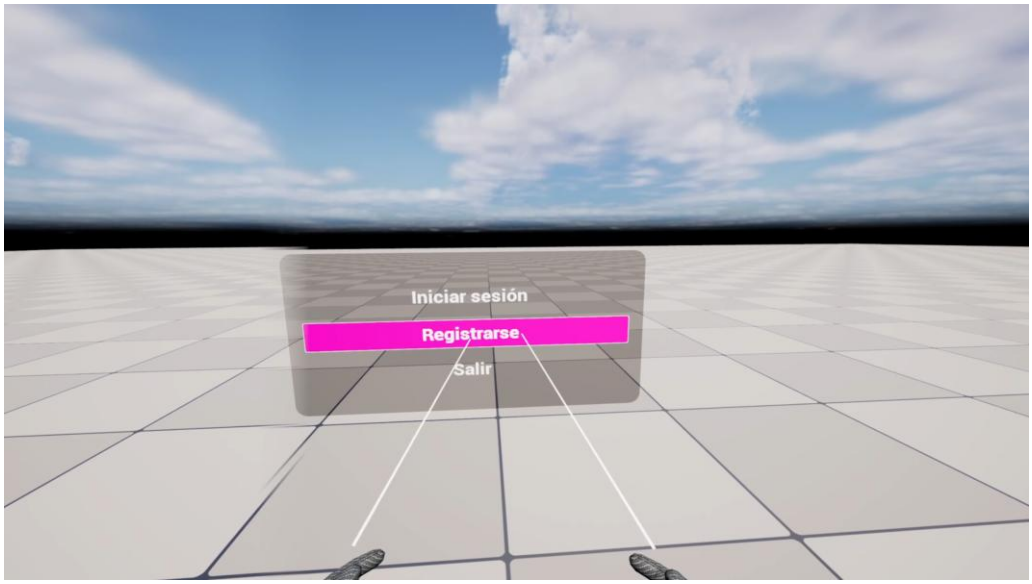

Figura 3.40: Ejemplo menú inicial

En caso de que no se tenga cuenta el usuario tendrá que registrarse, en esta pantalla se le pedirá algunos datos como el nombre de usuario y el correo, que han de ser únicos, y una contraseña, que tiene unos requerimientos mínimos y para la escritura se dispone siempre de un teclado virtual. (ver **Figura 3.41**). En caso de que alguno de los campos no cumpla con los requisitos se mostrará un mensaje de error debajo del mismo como se puede observar en la **Figura 3.42**.

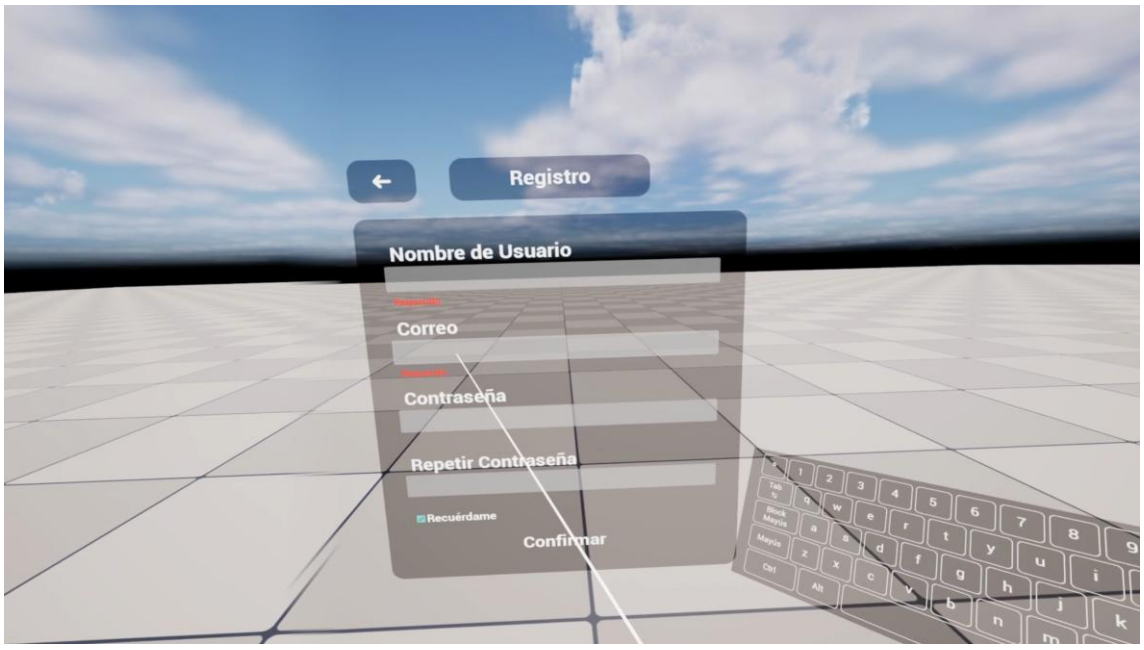

Figura 3.41: Ejemplo de registro

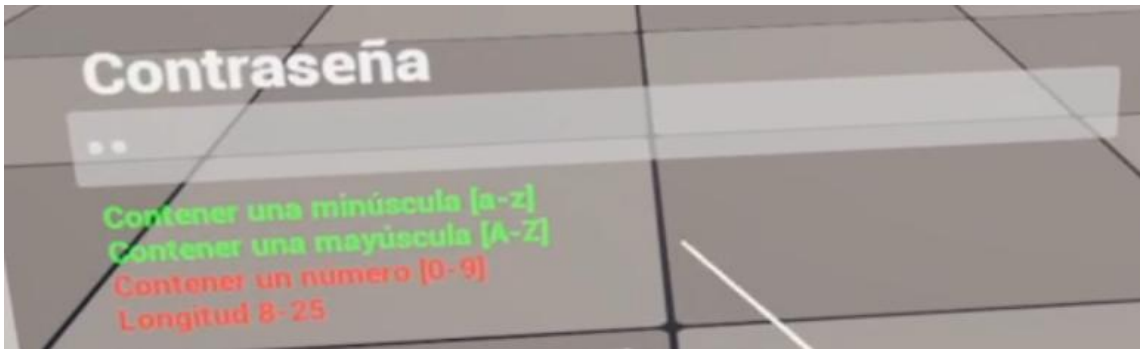

Figura 3.42: Ejemplo de requisitos de contraseña

Si el usuario dispone de una cuenta, este puede ir al menú de acceso, donde se le pedirá el nombre y la contraseña (ver **Figura 3.43**), permitiendo que se mantenga abierta la sesión para futuras partidas, o que se tenga que volver a introducir cada vez que se abre el juego.

En caso de que los datos sean incorrectos se mostrará un mensaje de error.

Figura 3.43: Ejemplo de inicio de sesión

3.3.2 Menú principal

Una vez el usuario ha iniciado sesión se le mostraran las opciones principales del juego, como se puede ver en la **Figura 3.44**. Listadas son:

- **Campaña:** donde están los niveles creados por el desarrollador. Estos están diseñados para introducir el jugador a las mecánicas del juego, así

como ir demostrando los diferentes usos de los objetos del juego, aumentando poco a poco en dificultad conforme se progrese.

- **Comunidad:** donde están los niveles creados por los jugadores.
- **Editor de niveles:** donde están los niveles creados por el usuario, al igual que permite crear nuevos niveles.
- **Configuración:** donde están las opciones para modificar como se ve el juego y como se controla el personaje.
- **Cerrar sesión:** este botón elimina los tokens del usuario del ordenador y cierra la sesión del usuario.
- **Salir:** este botón cierra el juego sin cerrar la sesión del usuario.

Figura 3.44: Ejemplo de menú principal

3.3.3 Campaña

Dentro del menú de campaña se encuentran los distintos niveles como ya se ha mencionado. El usuario es capaza de seleccionar la dificultad, y una vez hecho eso se le mostraran las opciones de niveles disponibles, en la **Figura 3.45** se puede ver el nivel de prueba disponible dentro de la sección de la dificultad fácil.

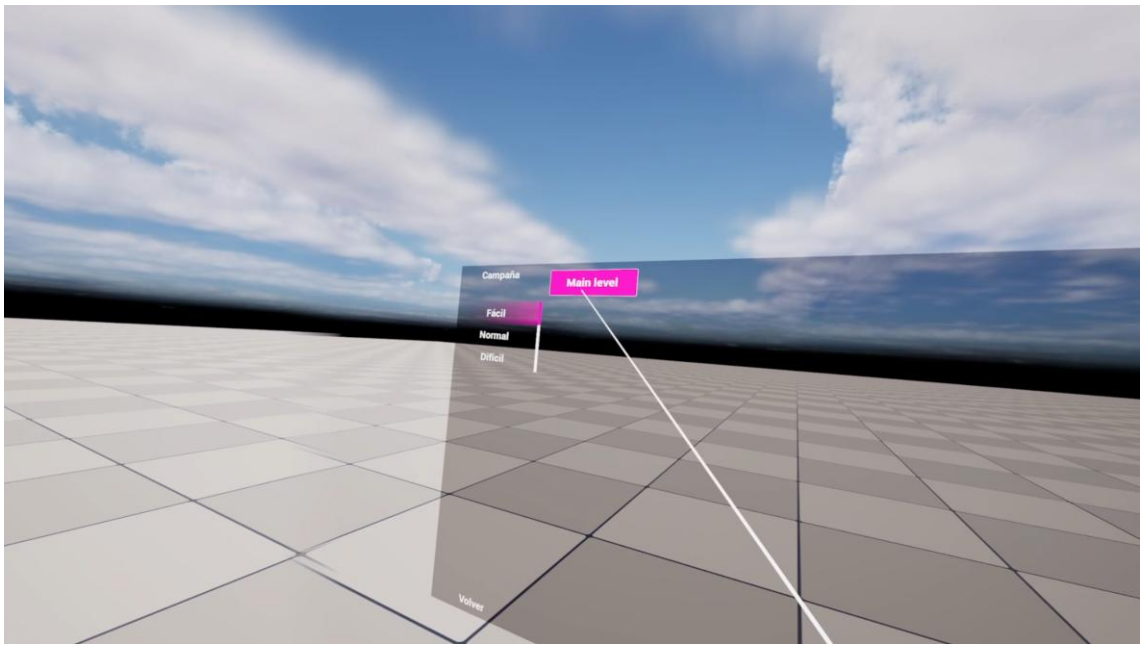

Figura 3.45: Ejemplo de menú de campaña

Una vez seleccionado, el nivel será cargado y el usuario comenzará con la resolución del puzle (ver **Figura 3.46**). Dentro del mismo nivel se puede abrir el menú de pausa que permite volver al menú principal.

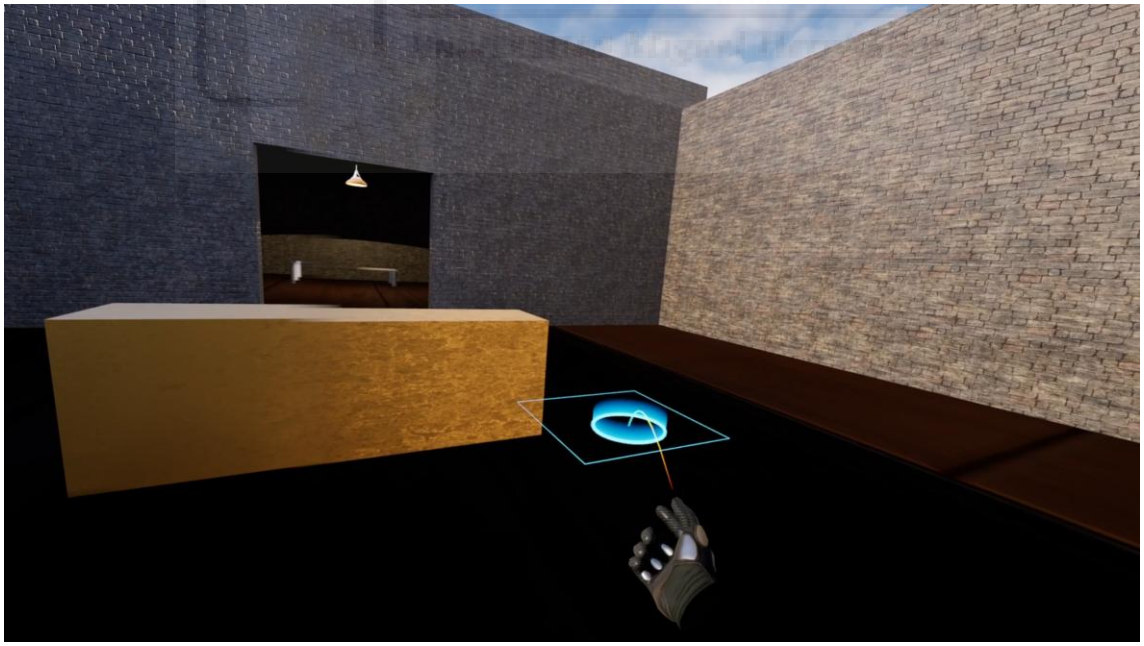

Figura 3.46: Ejemplo de nivel de campaña

3.3.4 Comunidad

Dentro de la comunidad se dispone de una lista ordenada de los niveles. Se puede navegar por distintas páginas, y cuando se selecciona un nivel se muestra

los datos en la sección derecha. En esa sección se incluye el nombre, autor, tags, rating, tiempo que el autor ha tardado en completar el nivel, veces que se ha completado el nivel, veces que se ha descargado. Además de tres pestañas donde se puede ver una descripción puesta por el creador, la clasificación de los usuarios que han completado el nivel, y los comentarios de los usuarios.

Dentro de esta misma pantalla el usuario puede calificar el nivel, escribir comentarios, y dar me gusta a los comentarios de los usuarios, que pueden ser ordenados por recientes o por su calificación.

En la **Figura 3.47** se puede ver el listado de niveles, y los datos del nivel seleccionado, mostrando la descripción puesta por el creador. En la **Figura 3.48** se puede ver la clasificación de los usuarios que han resuelto el nivel. Y en la **Figura 3.49** se puede ver los comentarios puestos por los usuarios.

Figura 3.47: Ejemplo de nivel de comunidad

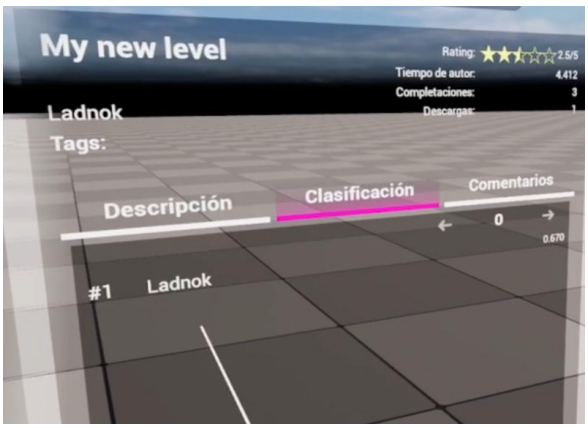

Figura 3.48: Ejemplo de clasificación de nivel

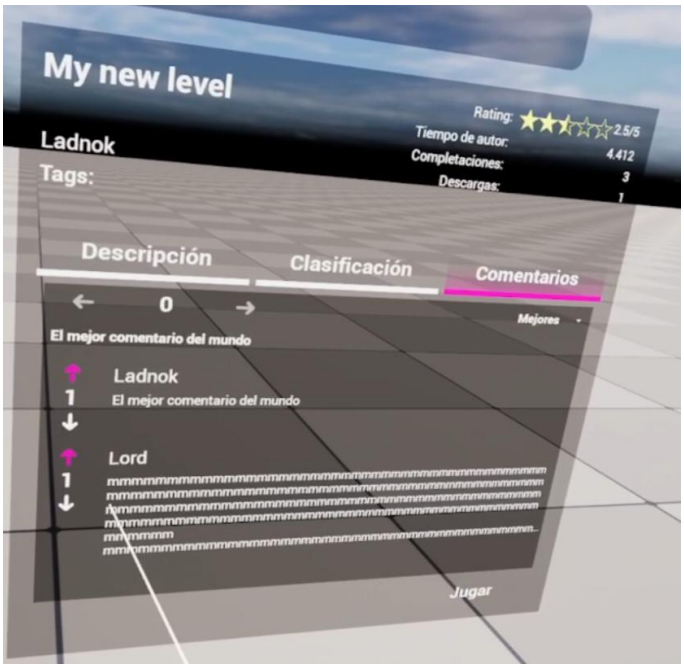

Figura 3.49: Ejemplo de comentarios de nivel

Por último, antes de poder jugar un nivel el usuario lo tiene que descargar (ver **Figura 3.50**). Y una vez se intenta jugar, se hará una comprobación de que los datos del nivel son correctos para mantener la integridad competitiva.

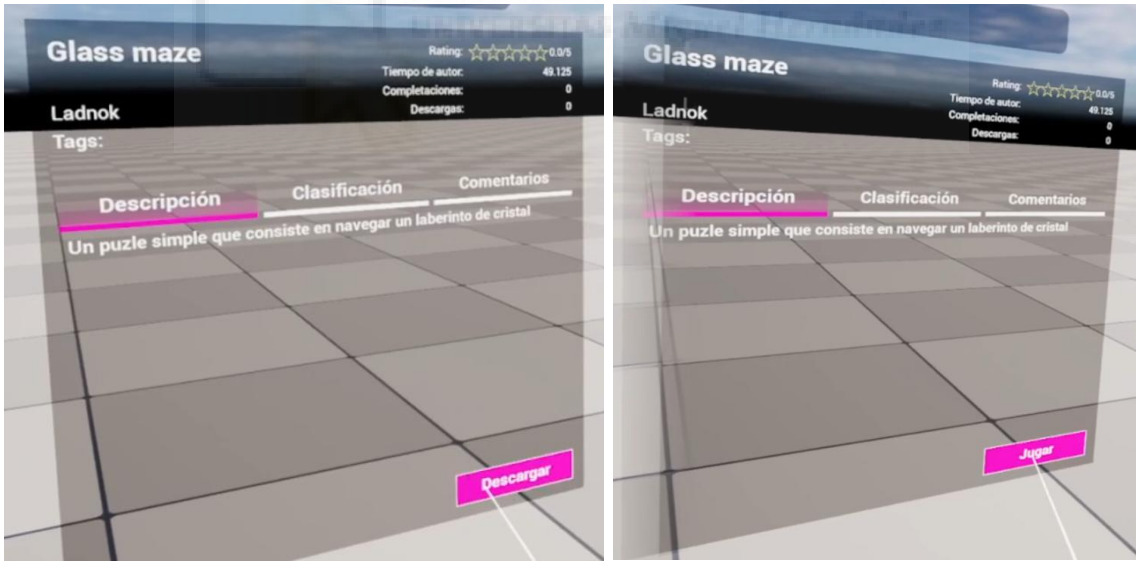

Figura 3.50: Ejemplo de nivel antes y después de ser descargado

Una vez se ha completado el nivel se mostrará una pantalla donde se enseña el tiempo que se ha tardado en completarlo como se muestra en la **Figura 3.51**, y este tiempo es enviado al servidor para que quede registrado y lo pueda ver todo el mundo.

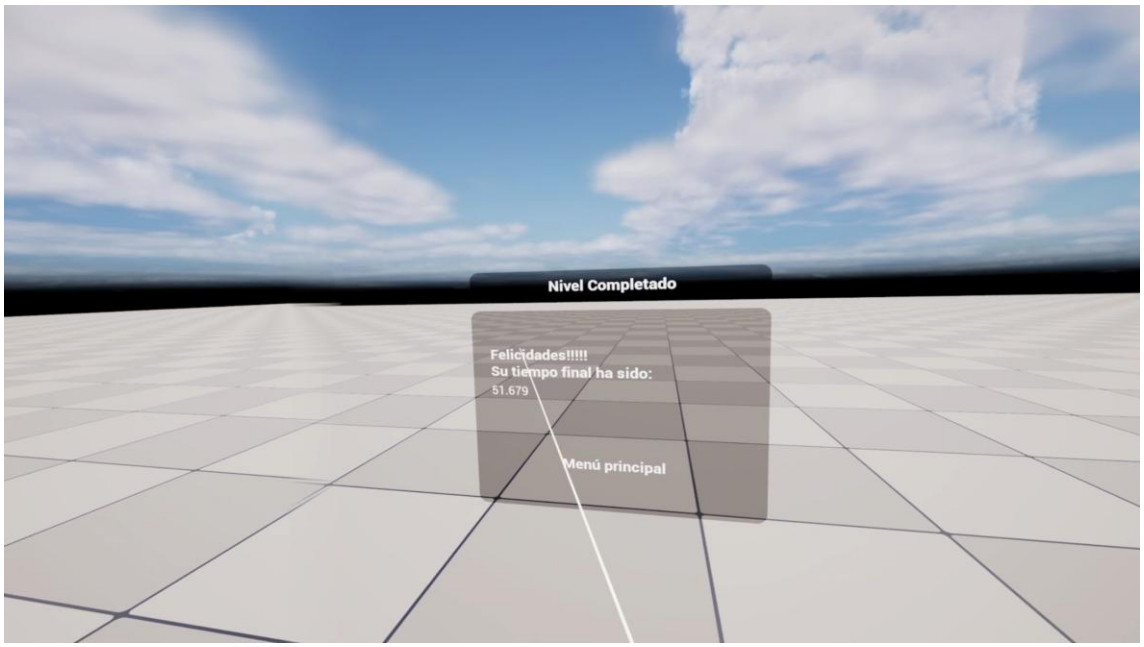

Figura 3.51: Ejemplo de nivel de comunidad completado

3.3.5 Editor de niveles

Dentro del editor de niveles se muestra un menú con todos los niveles creados por el usuario, una vez seleccionado uno se puede guardar, en caso de que se altere el nombre o descripción, cargar, lo que permite al usuario editar y validar el nivel, y borrar, lo que eliminará el nivel del disco local. También se dispone de un botón para volver al menú principal y para crear niveles nuevos (ver **Figura 3.52**).

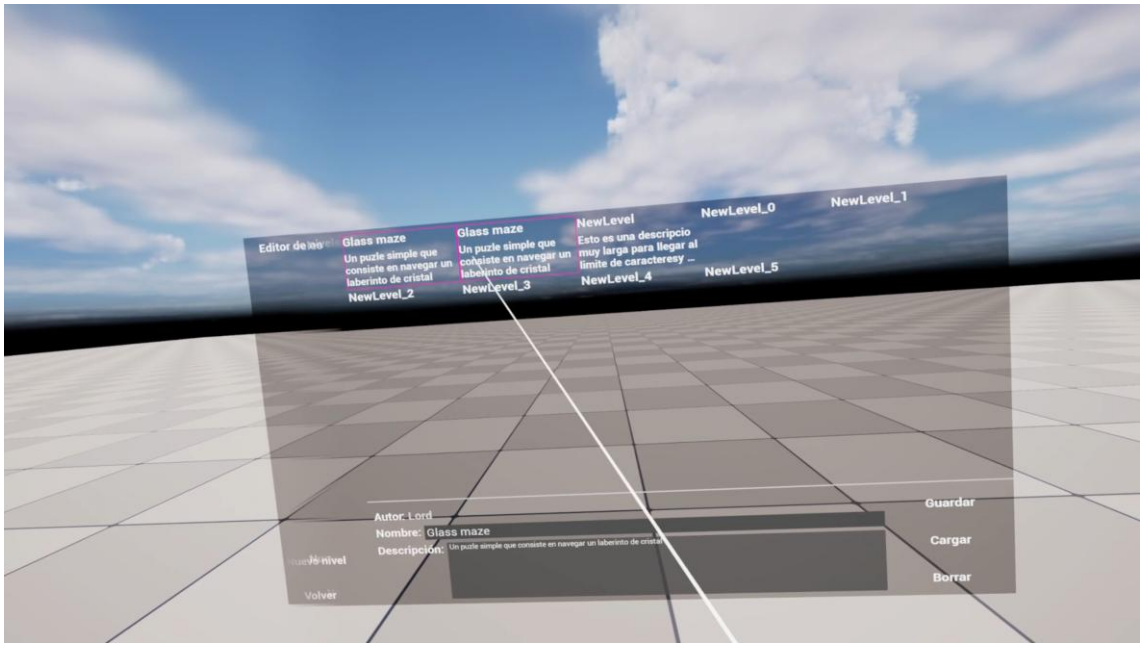

Figura 3.52: Ejemplo de niveles de usuario

Cuando se crea un nuevo nivel o se está editando uno ya existente se muestra un grid donde se pueden colocar los bloques del nivel. Cuando se colocan los bloques, el usuario puede ver indicadores visuales que marcan si esa posición es válida o no. Si lo es, el bloque se colocará al soltarlo como se puede ver en la **Figura 3.53**, en caso contrario este será eliminado.

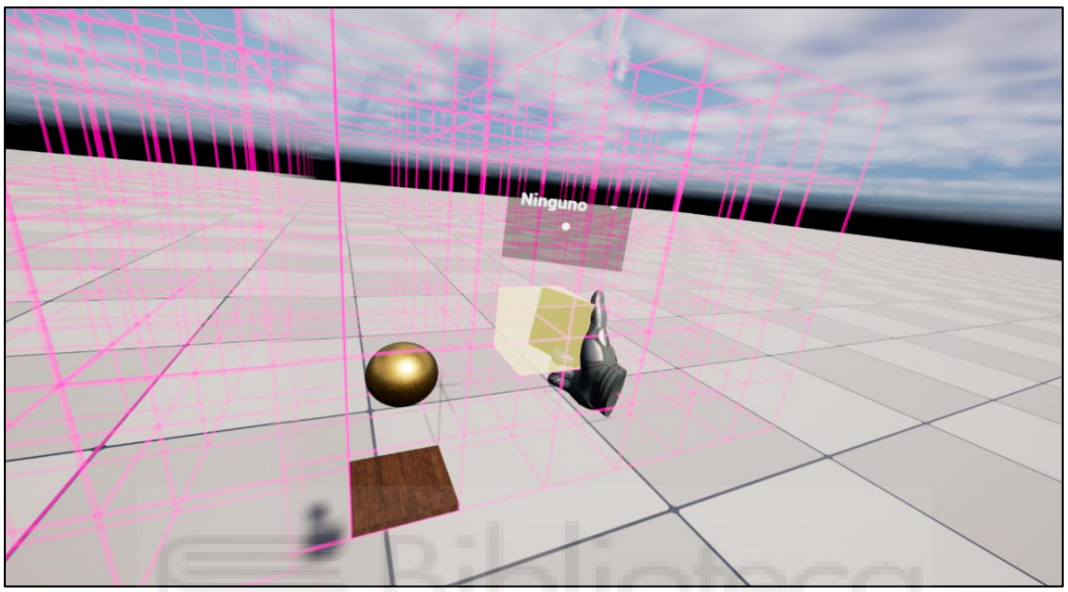

Figura 3.53: Ejemplo de usuario colocando un bloque

Dentro de las opciones disponibles, el usuario puede cambiar el material de los bloques, lo cual le dará un aspecto único a cada nivel, en la **Figura 3.54** se puede apreciar el menú de lista con los materiales disponibles.

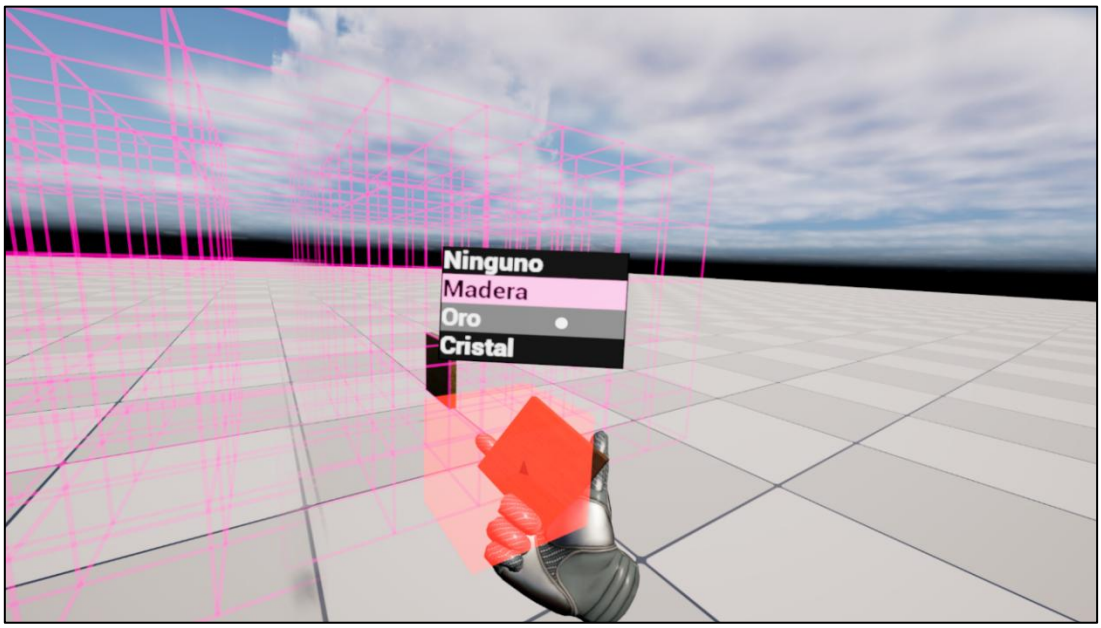

Figura 3.54: Ejemplo de usuario editando material de un bloque

Dentro del editor existen varios menús. El primero es el de bloques, este permite navegar a los distintos tipos de bloques (ver **Figura 3.55**), y al seleccionarlos estos aparecerán en el mundo.

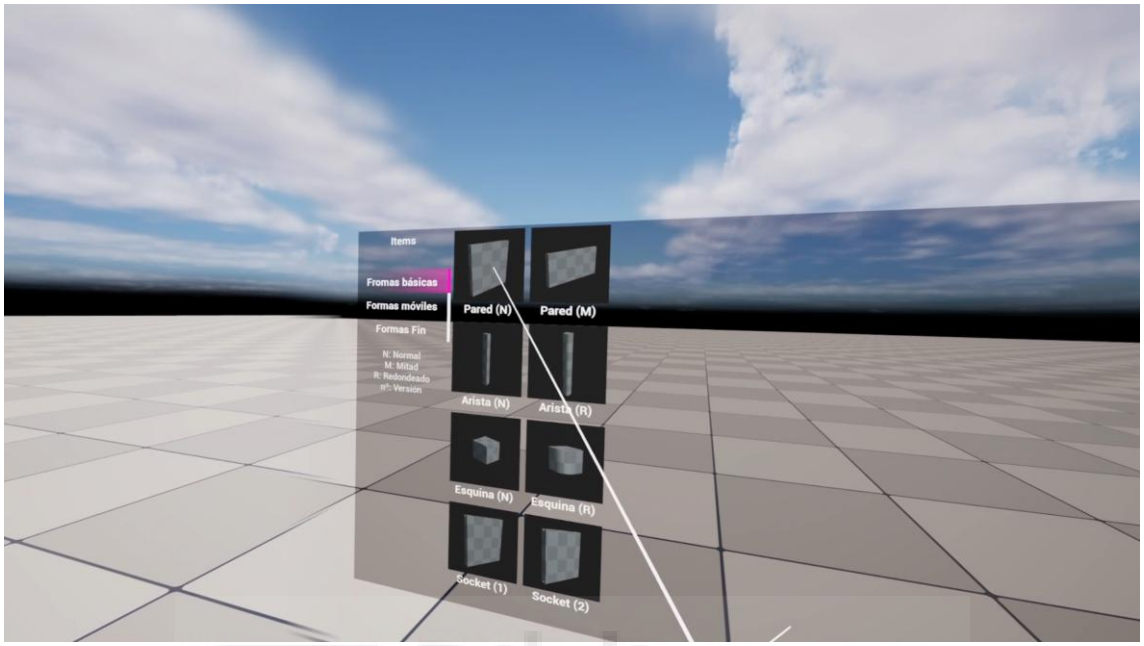

Figura 3.55: Ejemplo de menú de bloques

El segundo es el menú de pausa (ver Figura 3.56), que permite lo siguiente:

- **Guardar el nivel:** guarda el progreso realizado en el nivel.
- **Validar el nivel:** permite al usuario jugar el nivel para ver si se puede completar, y en caso de hacerlo, da la opción de publicarlo para el resto de los usuarios (ver **Figura 3.57**).
- **Ajustes de editor:** abre un segundo menú donde se puede cambiar los datos del nivel, como el nombre, descripción y tags (ver **Figura 3.58**). Además, hay otra pestaña donde se puede cambiar la visibilidad y colores del grid, colores de la colocación de bloques, superposición y colocación libre de bloques, etc. (ver **Figura 3.59**)
- **Configuración:** abre un segundo menú donde se puede alterar la configuración del juego.

- **Menú principal:** sale del editor y vuelve al menú principal, en caso de que se hayan realizado cambios, pero no se ha guardado, saldrá un mensaje de aviso.

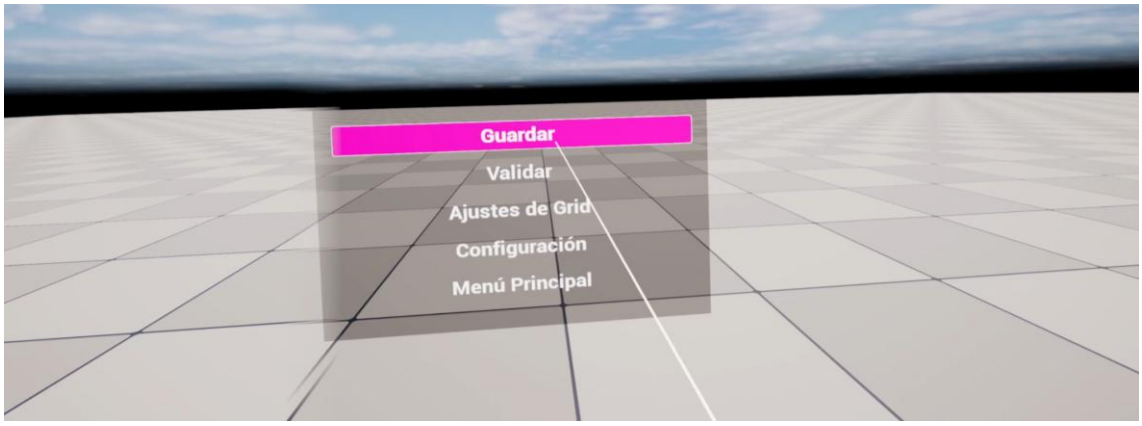

Figura 3.56: Ejemplo de menú de pausa en el editor

Figura 3.57: Ejemplo de menú post validación de nivel

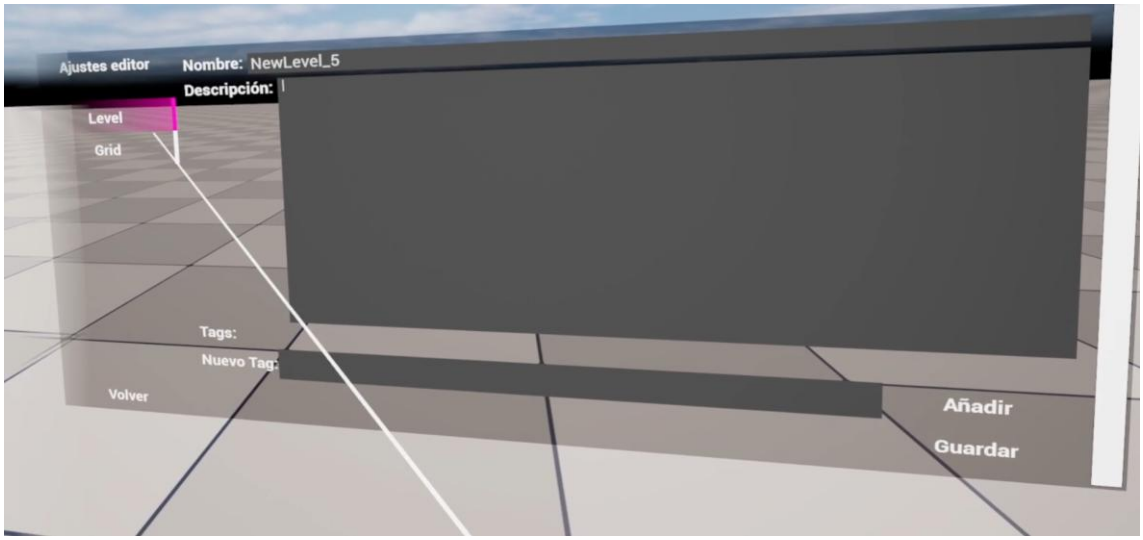

Figura 3.58: Ejemplo de ajustes del nivel

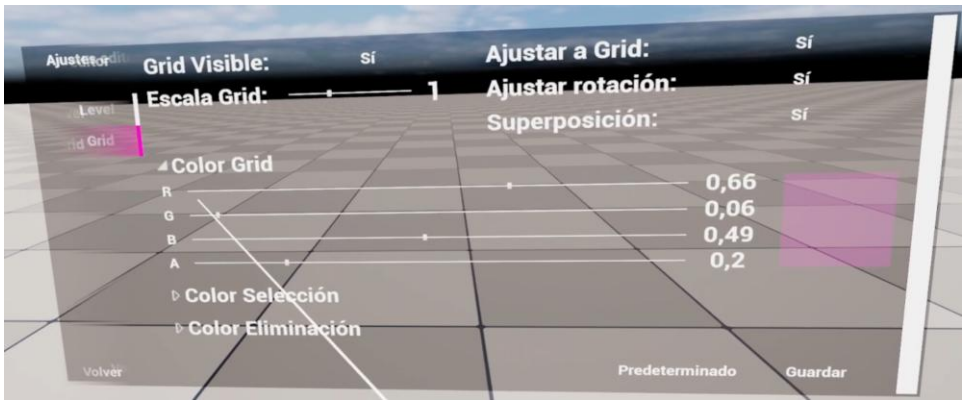

Figura 3.59: Ejemplo de ajustes del grid

3.4 Código fuente y disponibilidad

El código fuente de esta aplicación se encuentra disponible en un repositorio privado de Bitbucket al que se puede acceder a través del siguiente enlace:

https://bitbucket.org/syliconlab/2324_hernandez_reynoso_enzo/src/master/.

En la **Figura 3.60** se muestra una captura de la carpeta principal del repositorio donde se pueden apreciar algunas de las carpetas que componen el proyecto. Si se desea contribuir al mismo se deberá solicitar acceso.

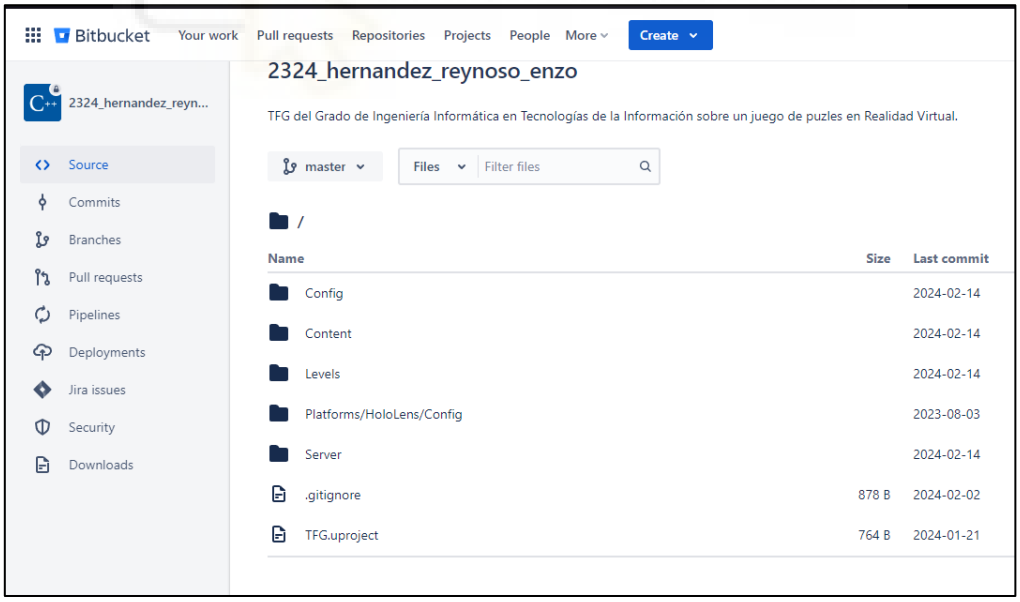

Figura 3.60: Bitbucket

Actualmente, el videojuego se encuentra en un estado de pre-alpha. Esto quiere decir que los menús, assets, imágenes, y modos de juego están sujetos a

cambios constantes y no reflejan el estado del juego final. También es posible que se encuentren bugs que afecten al uso del juego.

Dado que al acceder al juego se necesita de una cuenta, actualmente no se puede hacer uso del juego ya que no se dispone de sistema de hosting público. En caso de que se quiera hacer uso del juego se tendrá que *hostear* un servidor y base de datos privados y editar los archivos de configuración del servidor.

4 Conclusiones y trabajo futuro

Este proyecto comenzó como idea mucho antes de ser planteado como un TFG. La ambición inicial era considerable, especialmente teniendo en cuenta la falta de experiencia en el desarrollo de juegos de Realidad Virtual. Sin embargo, el proyecto tuvo que ajustarse a una visión más conservadora debido a la limitación de tiempo, que ha sido agravado por la incorporación de prácticas durante el desarrollo.

En cuanto al grado de consecución de los objetivos planteados al inicio del proyecto cabe destacar lo siguiente:

- Se puede dar por cumplido el objetivo de desarrollar un videojuego de Realidad Virtual, aunque algunas funcionalidades planteadas durante las revisiones periódicas no han podido implementarse. En general, se logró un juego jugable según lo planeado inicialmente.
- En cuanto a los servicios de almacenamiento y compartición, creemos que también se han alcanzado los objetivos ya que se ha implementado un servidor con una base de datos e interfaces que permiten a los usuarios subir sus niveles, y descargar y jugar los niveles creados por otros usuarios.
- En términos de aprendizaje y desarrollo, se ha aprendido sobre las herramientas disponibles para el desarrollo de juegos de Realidad Virtual, así como a trabajar con niveles, interfaces, almacenamiento local y conexiones remotas, entre otros aspectos. En el apartado del servidor, se aplicaron conocimientos de desarrollo web, prestación de servicios, ciberseguridad y manejo de bases de datos. También se aplicaron conocimientos sobre desarrollo de software para la creación de esta memoria y la documentación del proyecto.

4.1 Valoración personal

Teniendo en cuenta el tiempo disponible para el desarrollo, puedo decir que estoy satisfecho con el trabajo realizado. Considero que hay algunos apartados

implementados que son mejorables, un ejemplo sería el rendimiento en dispositivos de gama media, pero arreglarlo requeriría de un rediseño o implementación completamente distinta de cómo está hecho ahora mismo. El problema radica en el desconocimiento durante el desarrollo, pero esa es una de las facetas de aprender mientras se trabaja con tecnologías nuevas.

4.2 Trabajo futuro

En cuanto al futuro del proyecto, este nunca fue ideado con el objetivo de ser comercializado, sino más bien para el disfrute de los jugadores. Aun así, el proyecto está en una fase muy temprana de desarrollo, y hay mil y un caminos que se pueden tomar de aquí en adelante.

Como se ha mencionado, hay funcionalidades que no han podido ser implementadas, y actualmente se dispone de una multitud de objetivos para mejorar el diseño actual, de los cuales se pueden destacar los siguientes:

- **Interfaces:**
  - Mejora de la personalidad e identidad visual de las interfaces.
  - Reconsideración de la ubicación de las interfaces para mejorar la experiencia del usuario.

- **Editor de niveles:**
  - Rediseño del manejo de bloques para facilitar y agilizar su uso.
  - Ampliación de los controles disponibles en el editor.
  - Adición de más bloques para mayor variedad.

- **Sonido:**
  - Incorporación de música de fondo y otros elementos de audio para mejorar la experiencia del jugador.

- **Escenarios:**
  - Adición de escenarios para hacer la jugabilidad más interesante y visualmente atractiva.

- **Comunidad:**
    - Implementación de perfiles de usuarios, barra de búsqueda con filtros y mejoras en el apartado de comentarios.
    - Posibilidad de que los jugadores incluyan sus propios bloques y escenarios en los niveles.

- **Administración:**
    - Desarrollo de una plataforma web para que los administradores revisen niveles, comentarios y perfiles de usuario.

Como se ha mencionado, estas son las ideas más importantes, hay muchas más que no se han mencionado y que espero poder implementar si continúo trabajando en este proyecto en el futuro.

5 Bibliografía

[1] Carme Mayans, “El senet, un juego de mesa para alcanzar el inframundo,” https://historia.nationalgeographic.com.es/a/senet-juego-mesa-para-alcanzar-inframundo_16348. Accessed: Apr. 26, 2023. [Online]. Available: https://historia.nationalgeographic.com.es/a/senet-juego-mesa-para-alcanzar-inframundo_16348

[2] Vanishing INC., “What Is A Japanese Puzzle Box (Himitsu Bako)?,” https://www.vanishingincmagic.com/puzzles-and-brainteasers/articles/what-is-a-japanese-puzzle-box/.

[3] Brian P., “Virtual Reality,” https://techterms.com/definition/virtualreality.

[4] Tomofumi Kuzuhara, “Virtual Reality’s Second Wind: VR Game Revenues Will Hit $3.2B by 2024 as Active Headset Numbers Hit 46.0M,” *https://newzoo.com/resources/blog/vr-game-revenues-will-hit-3-2b-by-2024-as-active-headset-numbers-hit-46m*, Nov. 2022, Accessed: Apr. 26, 2023. [Online]. Available: https://newzoo.com/resources/blog/vr-game-revenues-will-hit-3-2b-by-2024-as-active-headset-numbers-hit-46m

[5] “The First Video Game?,” https://www.bnl.gov/about/history/firstvideo.php. Accessed: Apr. 26, 2023. [Online]. Available: https://www.bnl.gov/about/history/firstvideo.php

[6] Noah Wardrip-Fruin, “Before Pong, There Was Computer Space,” https://thereader.mitpress.mit.edu/before-pong-there-was-computer-space/. Accessed: May 02, 2023. [Online]. Available: https://thereader.mitpress.mit.edu/before-pong-there-was-computer-space/

[7] “Magnavox Odyssey,” https://retromaquinitas.com/consolas/consolas-philips/odyssey/. Accessed: Apr. 26, 2023. [Online]. Available: https://retromaquinitas.com/consolas/consolas-philips/odyssey/

[8] Electronic Visualization Library (EVL), “Sayre Glove (first wired data glove),” https://www.evl.uic.edu/research/2162.

[9] DOM BARNARD, “History of VR - Timeline of Events and Tech Development,” https://virtualspeech.com/blog/history-of-vr. Accessed: May 02, 2023. [Online]. Available: https://virtualspeech.com/blog/history-of-vr

[10] David Beren, “What Was the Video Game Crash of 1983 and Why Did It Happen?,” https://history-computer.com/what-was-the-video-game-crash-of-1983-and-why-did-it-happen/. Accessed: Apr. 26, 2023. [Online]. Available: https://history-computer.com/what-was-the-video-game-crash-of-1983-and-why-did-it-happen/

[11] “VR Games History,” *https://educationecosystem.com/guides/game-development/vr-games/history*, Accessed: Apr. 26, 2023. [Online]. Available: https://educationecosystem.com/guides/game-development/vr-games/history

[12] “Nintendo pins hopes on Virtual Boy,” *NEXT Generation Issue #3*, Mar. 1995. Accessed: Apr. 26, 2023. [Online]. Available: https://archive.org/details/nextgen-issue-003/page/n21/mode/2up

[13] Patrick Kolan, “IGN Retro: Virtual Boy’s Best Games,” https://web.archive.org/web/20090123101614/http://retro.ign.com/articles/845/845487p1.html. Accessed: Apr. 26, 2023. [Online]. Available:

https://web.archive.org/web/20090123101614/http://retro.ign.com/articles/845/845487p1.html

[14] Patrick Kolan, “IGN Retro: Virtual Boy Revisited,” https://web.archive.org/web/20090527180857/http://retro.ign.com/articles/845/845419p1.html. Accessed: Apr. 26, 2023. [Online]. Available: https://web.archive.org/web/20090527180857/http://retro.ign.com/articles/845/845419p1.html

[15] Eddie Makuch, “Team Fortress 2 is Oculus Rift’s first game,” https://www.gamespot.com/articles/team-fortress-2-is-oculus-rifts-first-game/1100-6405520/. Accessed: Apr. 28, 2023. [Online]. Available: https://www.gamespot.com/articles/team-fortress-2-is-oculus-rifts-first-game/1100-6405520/

[16] Andrew Webster, “These are the first Oculus Rift games,” https://www.theverge.com/2015/6/11/8766703/oculus-rift-launch-games. Accessed: Apr. 28, 2023. [Online]. Available: https://www.theverge.com/2015/6/11/8766703/oculus-rift-launch-games

[17] DOM BARNARD, “History of VR - Timeline of Events and Tech Development,” https://virtualspeech.com/blog/history-of-vr. Accessed: May 02, 2023. [Online]. Available: https://virtualspeech.com/blog/history-of-vr

[18] Michael Higham, “Valve Index Impressions - A Necessary But Incremental Step For PC VR,” https://www.gamespot.com/articles/valve-index-impressions-a-necessary-but-incrementa/1100-6467377/. Accessed: May 02, 2023. [Online]. Available: https://www.gamespot.com/articles/valve-index-impressions-a-necessary-but-incrementa/1100-6467377/

[19] Scott Stein and Andrew Lanxon, “The 41 Best VR Games,” https://www.cnet.com/pictures/best-vr-games/22/. Accessed:

Apr. 28, 2023. [Online]. Available: https://www.cnet.com/pictures/best-vr-games/22/

[20] Steel Crate Games®, “Keep Talking and Nobody Explodes,” https://keeptalkinggame.com. Accessed: Jun. 17, 2025. [Online]. Available: https://keeptalkinggame.com

[21] Polyarc, “Moss,” https://www.polyarcgames.com/games/moss. Accessed: Jun. 17, 2025. [Online]. Available: https://www.polyarcgames.com/games/moss

[22] Fireproof Games, “The Rom VR: A Dark Matter,” https://www.fireproofgames.com/games/the-room-vr-a-dark-matter. Accessed: Jun. 17, 2025. [Online]. Available: https://www.fireproofgames.com/games/the-room-vr-a-dark-matter

[23] Realities.io Inc, “Puzzling places,” https://www.realities.io/puzzling-places. Accessed: Jun. 17, 2025. [Online]. Available: https://www.realities.io/puzzling-places

[24] Unity Technologies, “Unity Engine,” https://unity.com/es. Accessed: Jun. 17, 2025. [Online]. Available: https://unity.com/es

[25] Ton Roosendaal and Blender Foundation, “Blender,” https://www.blender.org. Accessed: Jun. 17, 2025. [Online]. Available: https://www.blender.org

[26] Alias Systems Corporation and Inc. Autodesk, “Autodesk Maya,” https://www.autodesk.com/products/maya/overview. Accessed: Jun. 17, 2025. [Online]. Available: https://www.autodesk.com/products/maya/overview

[27] Valve, “Steam,” https://store.steampowered.com. Accessed: Jun. 17, 2025. [Online]. Available: https://store.steampowered.com

[28] Epic Games, “Epic Games Store.”

[29] Leaf Corcoran, “Itch.io,” https://itch.io. Accessed: Jun. 17, 2025. [Online]. Available: https://itch.io

ANEXO I: Definición de los casos de uso

En este anexo se encuentran detallados lo casos de uso definidos:

- **Tabla 14:** Caso de Uso: Iniciar juego
- **Tabla 15:** Caso de Uso: Registrarse
- **Tabla 16:** Caso de Uso: Iniciar sesión
- **Tabla 17:** Caso de Uso: Acceder al menú principal
- **Tabla 18:** Caso de Uso: Cerrar sesión
- **Tabla 19:** Caso de Uso: Acceder a la campaña
- **Tabla 20:** Caso de Uso: Jugar puzle
- **Tabla 21:** Caso de Uso: Completar puzle
- **Tabla 22:** Caso de Uso: Acceder a la comunidad
- **Tabla 23:** Caso de Uso: Ver datos de un puzle
- **Tabla 24:** Caso de Uso: Descargar puzle
- **Tabla 25:** Caso de Uso: Calificar puzle
- **Tabla 26:** Caso de Uso: Ver tabla de clasificación
- **Tabla 27:** Caso de Uso: Ver comentarios
- **Tabla 28:** Caso de Uso: Cambiar orden de los comentarios
- **Tabla 29:** Caso de Uso: Añadir comentario
- **Tabla 30:** Caso de Uso: Votar comentario
- **Tabla 31:** Caso de Uso: Ver puzles locales
- **Tabla 32:** Caso de Uso: Crear nivel
- **Tabla 33:** Caso de Uso: Editar datos de un puzle
- **Tabla 34:** Caso de Uso: Borrar puzle
- **Tabla 35:** Caso de Uso: Cargar puzle
- **Tabla 36:** Caso de Uso: Validar puzle
- **Tabla 37:** Caso de Uso: Subir puzle a la comunidad
- **Tabla 38:** Caso de Uso: Ver listado de ítems
- **Tabla 39**: Caso de Uso: Generar ítems)
- **Tabla 40:** Caso de Uso: Cambiar material de ítem)
- **Tabla 41:** Caso de Uso: Pausar juego
- **Tabla 42:** Caso de Uso: Acceder a la configuración

C.U.1	Iniciar juego
Actores	Usuario no registrado, Usuario registrado.
Descripción	Proceso por el cual un usuario inicia el juego de Realidad Virtual.
Precondición	• El juego debe estar instalado. • El usuario debe de disponer de un dispositivo de Realidad Virtual compatible.
Secuencia normal	1. Hacer doble clic o presionar la tecla “Enter” con el juego seleccionado. 2. Esperar a que se inicie el juego.
Postcondición	• Una vez finalizado la inicialización se mostrará la interfaz de inicio del juego en el casco de realidad virtual del usuario.
Excepciones	• Si ya existe una sesión iniciada se mostrará directamente el menú principal.
Frecuencia	Alta.
Importancia	Alta.

Tabla 14: Caso de Uso: Iniciar juego

C.U.2	Registrarse
Actores	Usuario no registrado, Usuario registrado.
Descripción	Proceso por el cual un usuario se registra en la base de datos.
Precondición	• C.U.1.
Secuencia normal	1. Se pulsa en el botón de “Registrarse”. 2. Se introduce el nombre de usuario, correo, contraseña y repetir la contraseña. 3. Se envía formulario.
Postcondición	• El usuario queda registrado. • Se inicia sesión automáticamente. • El token de identificación JWT se guarda localmente. • Se muestra el menú principal.

Excepciones	• Si ya existe un usuario con ese nombre o correo se muestra un mensaje de error. • Si las contraseñas no coinciden o no se cumplen unos requisitos mínimos (longitud y caracteres requeridos), se muestra un mensaje de error. • El usuario puede pulsar un botón para regresar al menú anterior (C.U.1)
Frecuencia	Alta.
Importancia	Alta.

Tabla 15: Caso de Uso: Registrarse

C.U.3	**Iniciar sesión**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado inicia sesión.
Precondición	• C.U.1.
Secuencia normal	1. Se pulsa en el botón de “Iniciar sesión”. 2. Se introduce el nombre de usuario y la contraseña. 3. Se envía el formulario.
Postcondición	• Se inicia sesión. • El token de identificación JWT se guarda localmente. • Se muestra el menú principal.
Excepciones	• Si las credenciales son incorrectas se muestra un mensaje de error genérico. • El usuario puede pulsar un botón para regresar al menú anterior (C.U.1).
Frecuencia	Alta.
Importancia	Alta.

Tabla 16: Caso de Uso: Iniciar sesión

C.U.4	**Acceder al menú principal**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado accede al menú principal.
Precondición	• Se ha de tener una sesión actualmente.
Secuencia normal	1. Realizar los C.U.1, C.U.2 o C.U.3

Postcondición	• Se muestra el menú principal.
Excepciones	-
Frecuencia	Alta.
Importancia	Alta.

Tabla 17: Caso de Uso: Acceder al menú principal

C.U.5	**Cerrar sesión**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado cierra sesión.
Precondición	• C.U.4.
Secuencia normal	1. Se pulsa en el botón de “Cerrar sesión”.
Postcondición	• Se borran las credenciales localmente. • Se muestra el menú de inicio.
Excepciones	-
Frecuencia	Baja.
Importancia	Alta.

Tabla 18: Caso de Uso: Cerrar sesión

C.U.6	**Acceder a la campaña**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado accede al apartado de la campaña.
Precondición	• C.U.4.
Secuencia normal	1. Se pulsa en el botón “Campaña”.
Postcondición	• Se muestra el menú de campaña.
Excepciones	-
Frecuencia	Media.
Importancia	Media.

Tabla 19: Caso de Uso: Acceder a la campaña

C.U.7	Jugar puzle
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario juega un puzle.
Precondición 1	• C.U.6.
Secuencia normal 1	1. Se pulsa en uno de los puzles.
Precondición 2	• El puzle de la comunidad ha de estar descargado (C.U.11).
Secuencia normal 2	1. Se pulsa en el botón “Jugar”.
Postcondición	• Se cargan los datos necesarios y se cambia al nivel de juego. • Se muestra en pantalla el puzle cargado.
Excepciones	• Sí hay un error durante la carga de datos se cancela el proceso.
Frecuencia	Alta.
Importancia	Alta.

Tabla 20: Caso de Uso: Jugar puzle

C.U.8	Completar puzle
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario completa un puzle.
Precondición	• C.U.7.
Secuencia normal	1. Se juega el puzle. 2. Se agarra uno de los bloques de finalización. 3. Se envía una petición para añadir una completación al puzle.
Postcondición	• Se muestra un mensaje de felicitación. • Se actualizan las completaciones del puzle.
Excepciones	• Si es un puzle de campaña el tiempo no se almacena ni se actualizan las completaciones.
Frecuencia	Alta.
Importancia	Alta.

Tabla 21: Caso de Uso: Completar puzle

C.U.9	Acceder a la comunidad
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado accede a la sección de la comunidad.
Precondición	• C.U.4.
Secuencia normal	1. Se pulsa el botón “Comunidad”. 2. Se envía una petición y se comprueba la respuesta.
Postcondición	• Se muestra el menú de la comunidad con el listado de puzles actualmente presentes en la base de datos, ordenados por su calificación de mayor a menor.
Excepciones	• Si no se recibe respuesta, hay un error o la respuesta está vacía, se muestra una tabla vacía.
Frecuencia	Media.
Importancia	Alta.

Tabla 22: Caso de Uso: Acceder a la comunidad

C.U.10	Ver datos de un puzle
Actores	Usuario registrado.
Descripción	Proceso por el un usuario registrado visualiza los datos de un puzle de la comunidad.
Precondición	• C.U.9
Secuencia normal	1. Se pulsa uno de los puzles del listado. 2. Se envía una petición al servidor y se comprueba la respuesta.
Postcondición	• Se muestran los datos del puzle.
Excepciones	• Si no se recibe respuesta o hay un error se muestra un mensaje de que se ha de seleccionar un puzle.
Frecuencia	Media.
Importancia	Alta.

Tabla 23: Caso de Uso: Ver datos de un puzle

C.U.11	Descargar puzle
Actores	Usuario registrado.

Descripción	Proceso por el cual un usuario registrado descarga un puzle de la comunidad.
Precondición	• C.U.10
Secuencia normal	1. Se pulsa el botón “Descargar”.
Postcondición	• Se envía una petición al servidor y se almacena el puzle localmente. • Se actualiza la cantidad de descargas del puzle. • Se actualiza el botón para mostrar el mensaje “Jugar”. • El puzle está disponible para ser jugado.
Excepciones	• Si ya se tenía descargado el puzle no se muestra el botón. • Si no se recibe respuesta o hay un error se vuelve a mostrar el botón “Descargar”.
Frecuencia	Alta.
Importancia	Alta.

Tabla 24: Caso de Uso: Descargar puzle

C.U.12	**Calificar puzle**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado califica un puzle de la comunidad.
Precondición	• C.U.10
Secuencia normal	1. Se pulsa una de las estrellas mostradas en los datos del puzle.
Postcondición	• Se envía una petición y se actualiza la calificación localmente.
Excepciones	-
Frecuencia	Baja.
Importancia	Media.

Tabla 25: Caso de Uso: Calificar puzle

C.U.13	**Ver tabla de clasificación**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado visualiza la tabla de clasificación de un puzle de la comunidad.

Precondición	• C.U.10
Secuencia normal	1. Se pulsa el botón “Clasificación”. 2. Se envía una petición al servidor y se comprueba la respuesta. 3. Se remplaza la pestaña a la de clasificación.
Postcondición	• Se visualiza la tabla de clasificación.
Excepciones	• Si no se recibe respuesta, hay un error o la respuesta está vacía, se muestra una tabla vacía.
Frecuencia	Media.
Importancia	Media.

Tabla 26: Caso de Uso: Ver tabla de clasificación

C.U.14	**Ver comentarios**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado puede visualizar los comentarios de un puzle de la comunidad.
Precondición	• C.U.10
Secuencia normal	1. Se pulsa el botón “Comentarios”. 2. Se envía una petición al servidor y se comprueba la respuesta. 3. Se remplaza la pestaña a la de comentarios y se muestra los resultados. 4. Se envía una petición para obtener si el usuario ha creado un comentario para el nivel y se muestra el resultado. 5. Por cada comentario se envía una petición para comprobar el estado del “me gusta” asociado al usuario y, cuando se recibe la respuesta, se actualiza visualmente.
Postcondición	• Se visualiza los comentarios del puzle. • Se actualiza visualmente los estados de los "me gusta" asociados a cada comentario y al usuario.
Excepciones	• Si no se recibe respuesta, hay un error o la respuesta está vacía, se muestra una tabla vacía.
Frecuencia	Media.
Importancia	Baja.

Tabla 27: Caso de Uso: Ver comentarios

C.U.15	**Cambiar orden de los comentarios**
Actores	Usuario registrado.

Descripción	Proceso por el cual un usuario registrado cambial el orden de visualización de los comentarios de un puzle de la comunidad.
Precondición	• C.U.14
Secuencia normal	1. Se pulsa el desplegable. 2. Se selecciona una de las opciones. 3. Se envía una petición con el nuevo orden y se procesa la respuesta. 4. Por cada comentario se envía una petición para comprobar el estado del “me gusta” asociado al usuario y, cuando se recibe la respuesta, se actualiza visualmente.
Postcondición	• Se visualiza los comentarios del puzle con el nuevo orden. • Se actualiza visualmente los estados de los "me gusta" asociados a cada comentario y al usuario.
Excepciones	• Si no se recibe respuesta, hay un error o la respuesta está vacía, se muestra una tabla vacía.
Frecuencia	Media.
Importancia	Baja.

Tabla 28: Caso de Uso: Cambiar orden de los comentarios

C.U.16	**Añadir comentario**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado añade un comentario nuevo a un puzle de la comunidad.
Precondición	• C.U.14
Secuencia normal	1. Se escribe el comentario en el campo designado. 2. Se pulsa el botón “Aplicar”. 3. Se envía una petición y se procesa la respuesta. 4. Se oculta el formulario y se muestra el comentario realizado.
Postcondición	• Se visualiza el comentario añadido.
Excepciones	• Si no se recibe respuesta o hay un error se vuelve a mostrar el formulario.
Frecuencia	Media.
Importancia	Baja.

Tabla 29: Caso de Uso: Añadir comentario

C.U.17	Votar comentario
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado actualiza el estado de “me gusta” de un comentario.
Precondición	• C.U.14
Secuencia normal	1. Se pulsa el botón de “Me gusta” o “No me gusta” 2. Se envía una petición con el nuevo valor y se actualiza el botón visualmente junto con el balance.
Postcondición	• Se visualiza el nuevo balance de likes y el estado de los botones.
Excepciones	-
Frecuencia	Media.
Importancia	Baja.

Tabla 30: Caso de Uso: Votar comentario

C.U.18	Ver puzles locales
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado accede al listado de puzles locales.
Precondición	• C.U.4
Secuencia normal	1. Se pulsa el botón “Editor de niveles”.
Postcondición	• Se muestra el listado de niveles almacenados localmente, mostrando sus títulos y descripciones.
Excepciones	• Si no se encuentra puzles locales o hay un error en su carga, se muestra un listado vacío.
Frecuencia	Alt.
Importancia	Alta.

Tabla 31: Caso de Uso: Ver puzles locales

C.U.19	Crear nivel
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado Crea un puzle local nuevo.
Precondición	• C.U.18

Secuencia normal	1. Se pulsa el botón “Nuevo nivel”.
Postcondición	• Se vacían los datos cargados en caso de que los haya, y se cambia el nivel al editor.
Excepciones	-
Frecuencia	Alta.
Importancia	Alta.

Tabla 32: Caso de Uso: Crear nivel

C.U.20	**Editar datos de un puzle**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado puede editar los datos de un puzle local.
Precondición	• C.U.18 • C.U.27
Secuencia normal	1. Se pulsa en el puzle que se quiere editar. 2. Se selecciona el campo que se quiere editar. 3. Se cambia el contenido. 4. Se pulsa el botón “Guardar”.
Postcondición	• Se actualiza el fichero del puzle con los nuevos datos.
Excepciones	• Si hay un error al cargar los datos no se puede realizar la acción. • En caso de estar en el editor de puzles, primero se ha de pausar el juego (C.U.27) y acceder a las opciones del puzle.
Frecuencia	Alta.
Importancia	Alta.

Tabla 33: Caso de Uso: Editar datos de un puzle

C.U.21	**Borrar puzle**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado borra un puzle local.
Precondición	• C.U.18
Secuencia normal	1. Se pulsa en el puzle que se quiere borrar. 2. Se pulsa el botón “Borrar”.

Postcondición	• Se elimina el puzle localmente.
Excepciones	• Si hay un error al cargar los datos no se puede realizar la acción.
Frecuencia	Media.
Importancia	Alta.

Tabla 34: Caso de Uso: Borrar puzle

C.U.22	**Cargar puzle**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado carga los datos de un puzle y lo edita.
Precondición	• C.U.18
Secuencia normal	1. Se pulsa en el puzle que se quiere cargar. 2. Se pulsa en el botón ”Cargar”.
Postcondición	• Se cargan los datos del puzle en memoria y se cambia de nivel al editor.
Excepciones	• Si hay un error al cargar los datos no se puede realizar la acción. • Después de validar un puzle se puede volver al editor directamente sin subirlo a la comunidad.
Frecuencia	Alta.
Importancia	Alta.

Tabla 35: Caso de Uso: Cargar puzle

C.U.23	**Validar puzle**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado valida un puzle local.
Precondición	• C.U.22. • C.U.27. • Tener al menos una pieza de finalización.
Secuencia normal	1. Se pulsa el botón “Validar”.
Postcondición	• Se guarda los datos del puzle. • Se cambia al nivel de validación.

Excepciones	-
Frecuencia	Alta.
Importancia	Alta.

Tabla 36: Caso de Uso: Validar puzle

C.U.24	**Subir puzle a la comunidad**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado sube un puzle a la comunidad
Precondición	• C.U.23
Secuencia normal	1. Se juega el puzle. 2. Se agarra una piza de finalización. 3. Se muestra el menú de felicitación. 4. Se pulsa el botón “Publicar nivel”.
Postcondición	• Se envían los datos al servidor y se almacena en la base de datos. • Se devuelve al usuario al menú principal.
Excepciones	• Si el puzle ya se había subido, simplemente se actualizan los datos.
Frecuencia	Media.
Importancia	Alta.

Tabla 37: Caso de Uso: Subir puzle a la comunidad

C.U.25	**Ver listado de ítems**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado puede visualizar el listado de ítems disponibles.
Precondición	• C.U.22
Secuencia normal	1. Se pulsa el botón de ítems en el mando de preferencia.
Postcondición	• Se pausa el juego. • Se ocultan los elementos del juego. • Se muestra el menú de ítems, separados por categorías.
Excepciones	• Si ya se está en el menú de ítems se reanuda el juego.
Frecuencia	Alta.
Importancia	Alta.

Tabla 38: Caso de Uso: Ver listado de ítems

C.U.26	Generar ítems
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado genera un ítem nuevo en el editor de puzles.
Precondición	• C.U.25
Secuencia normal	1. Se pulsa en el botón del ítem de preferencia.
Postcondición	• Se genera el ítem 3D en el nivel del editor. • Se puede visualizar e interactuar con el ítem.
Excepciones	-
Frecuencia	Alta.
Importancia	Alta.

Tabla 39: Caso de Uso: Generar ítems

C.U.27	Cambiar material de ítem
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado cambia el material de un ítem
Precondición	• C.U.26
Secuencia normal	1. Se agarra uno de los ítems. 2. Se muestra un menú encima de la mano con un desplegable mostrando el material actual del ítem. 3. Se pulsa sobre el desplegable. 4. Se selecciona el nuevo material.
Postcondición	• Se actualiza el material del bloque. • Se actualiza la interfaz del material.
Excepciones	-
Frecuencia	Alta.
Importancia	Alta.

Tabla 40: Caso de Uso: Cambiar material de ítem

C.U.28	Pausar juego
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado pausa el juego.

Precondición	• C.U.7 • C.U.22 • C.U.23
Secuencia normal	1. Pulsar el botón de pause en el mando de preferencia.
Postcondición	• Se pausa el juego. • Se ocultan los elementos del juego. • Se muestra el menú de pausa.
Excepciones	• Si ya se está en el menú de pausa se reanuda el juego. • Dependiendo del nivel en el que se encuentre el usuario se mostrarán opciones distintas en el menú de pausa.
Frecuencia	Alta.
Importancia	Alta.

Tabla 41: Caso de Uso: Pausar juego

C.U.29	**Acceder a la configuración**
Actores	Usuario registrado.
Descripción	Proceso por el cual un usuario registrado accede a la configuración del juego.
Precondición	• C.U.4 • C.U.28
Secuencia normal	1. Pulsar el botón “Configuración”.
Postcondición	• Se muestra el menú de configuración, separado en distintas secciones.
Excepciones	-
Frecuencia	Media.
Importancia	Alta

Tabla 42: Caso de Uso: Acceder a la configuración

ANEXO II: Prototipos

En este apartado se encuentran los distintos prototipados realizados a lo largo del proyecto:

- **Figura II.1:** Para el menú principal no se había planteado el uso de usuarios de la manera en la que se ha implementado finalmente, por lo que se puede ver que el botón de cierre de sesión no existe.

- **Figura II.2:** El menú de la campaña inicialmente se propuso con una lista desplazable para cada dificultad, mostrando todo en el mismo sito, pero cuando se hizo la implementación se vio que estaba la opción de separar las interfaces en múltiples pestañas, por lo que se acabó optando por eso.

- **Figura II.3:** Para el editor de niveles, como aún no se entendía muy bien como implementar las interfaces, se planteó una interfaz más clásica basada en múltiples programas de edición como GIMP, Unreal Engie, Blender, etc.

- **Figura II.4:** El menú de ajustes era algo que sí que se tenía muy claro cómo iba a ser, después de tantos años jugando juegos era algo claro que apartados como mínimo tiene que haber.

- **Figura II.5:** Estos menús fueron diseñados más adelante, y se intentó hacer de manera que se viese que elementos serían necesarios a la hora de implementarlo dentro de Unreal Engine, de ahí el uso de color rojo para delimitar los tamaños.

- **Figura II.6:** Del mismo modo que la figura anterior, se pretendía mostrar cómo se vería con todo el contenido de la comunidad, y visualizar si sería mucha información lo que llevo a usar una fuente más pequeña y dar más espacio en el producto final.

- **Figura II.7:** La idea original, y algo que se plantea implementar en un futuro es el uso de elementos interactuables, lo que da mayor inversión, pero dada la complejidad se optó por un menú clásico.

- **Figura II.8:** La idea de los bloques estáticos era que permitiese crear un cubo, o realmente cualquier forma con paredes, de forma que no haya esquinas vacías, algo que considero que se ha cumplido. Del mismo modo, las piezas desplazables dan oportunidad a un puzle cambiante, lo que permite crear puzles más complejos.

- **Figura II.9:** Para estos prototipos, se esperaba implementar funcionalidades más complejas haciendo uso de las físicas de Unreal Engine, pero también se incluye la moneda de finalización, que, si fue implementada, y el logo representativo del juego.

- **Figura II.10:** Estos prototipos siguen la misma idea que la figura anterior, aunque sí que se pudo implementar el cajón, aunque con algunos cambios a su diseño.

- **Figura II.11:** Inicialmente se planteaba añadir una pantalla de carga que mostrase un logo del juego, algo que finalmente no se pudo añadir, pero sí que se trabajó como se ve en la siguiente figura.

- **Figura II.12:** A pesar de que se abandonó la idea por falta de tiempo, sí que se hizo un prototipo digital y se trabajó la idea de usar múltiples materiales e incluso un render siguiendo las transiciones mostradas en la **Figura II.11** que se puede encontrar en el siguiente enlace:

https://drive.google.com/file/d/1Q_jfRZOF4dJ2vrRnuhMNkzSAsJbEhq5a/view?usp=sharing

- **Figura II.13:** Como no se tenía del todo claro que información sería necesaria almacenar en la base de datos se hizo varias revisiones como

se puede ver por los tachones y apuntes, y el diseño final fue pasado a digital (ver **Figura 3.15**).

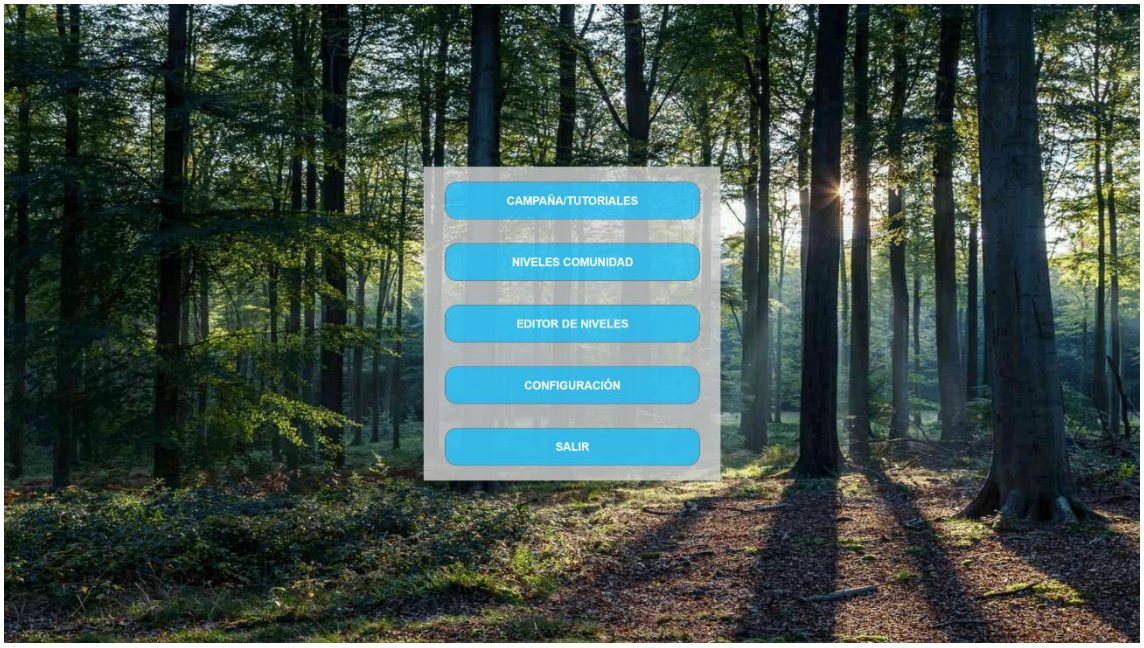

Figura II.1: Prototipo del Menú Principal

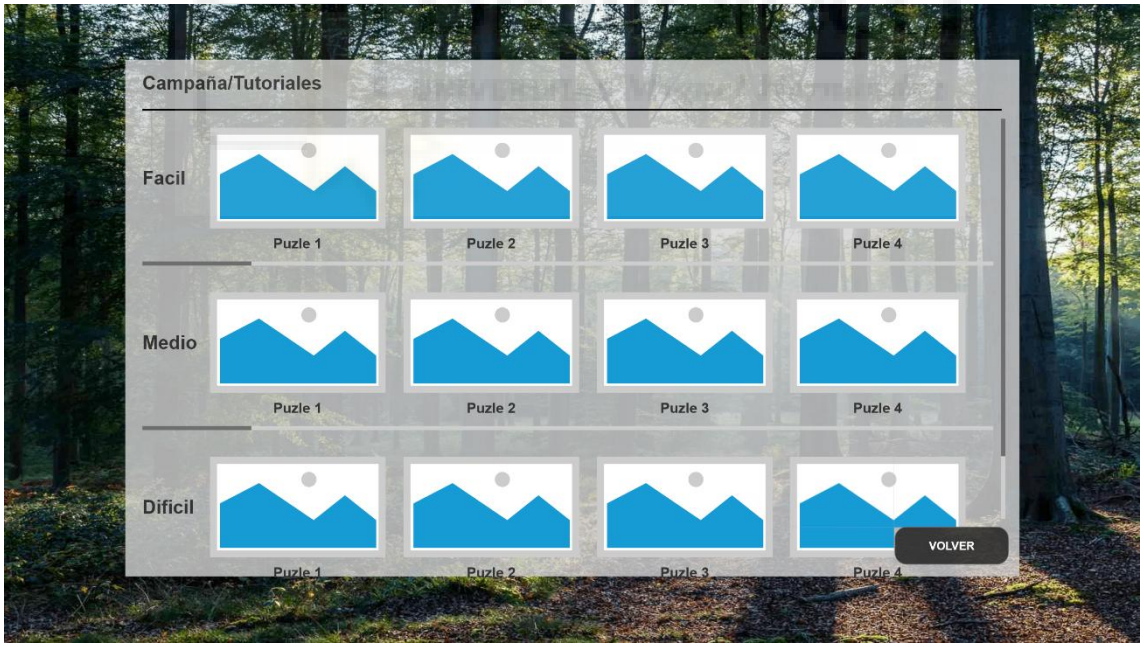

Figura II.2: Prototipo del Menú para el Modo Campaña

Figura II.3: Prototipo del Editor de Niveles

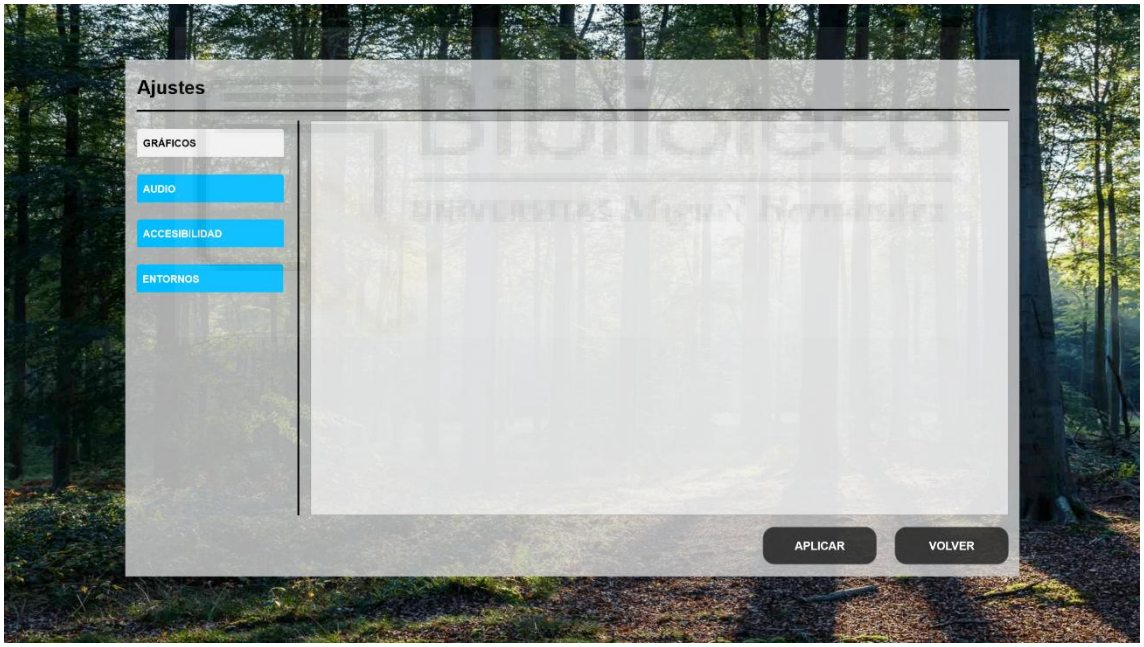

Figura II.4: Prototipo del Menú de Ajustes

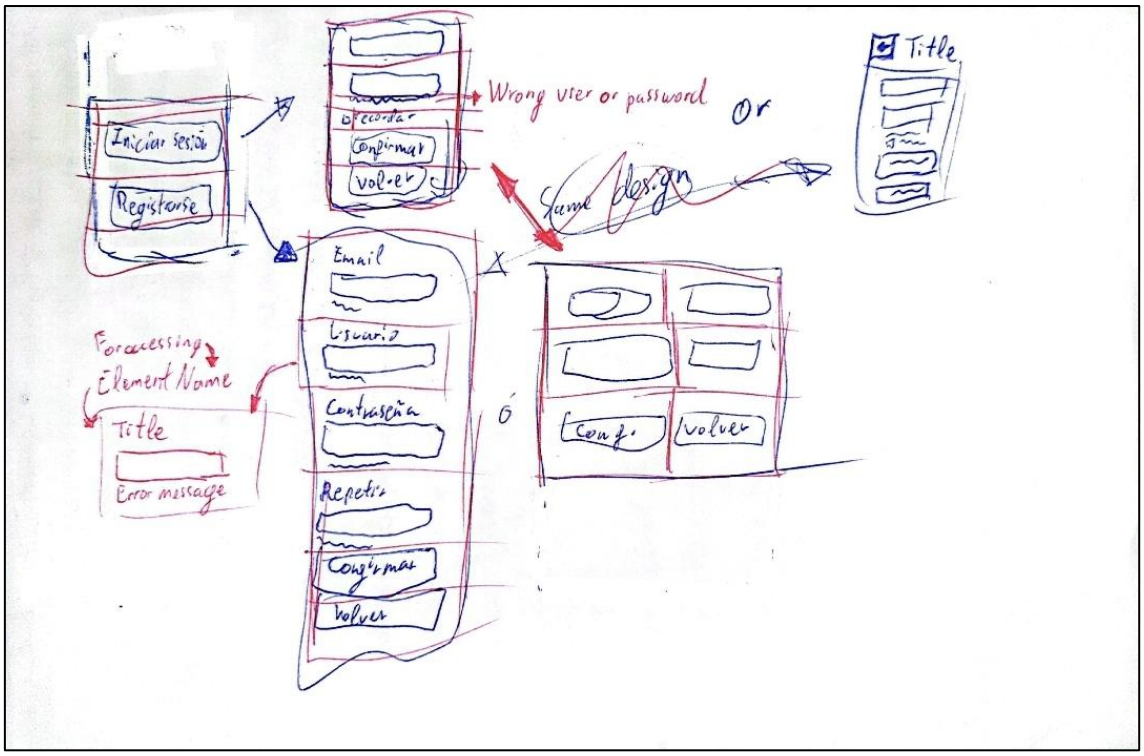

Figura II.5: Prototipos del Menú de Inicio, Registro, e Inicio de Sesión

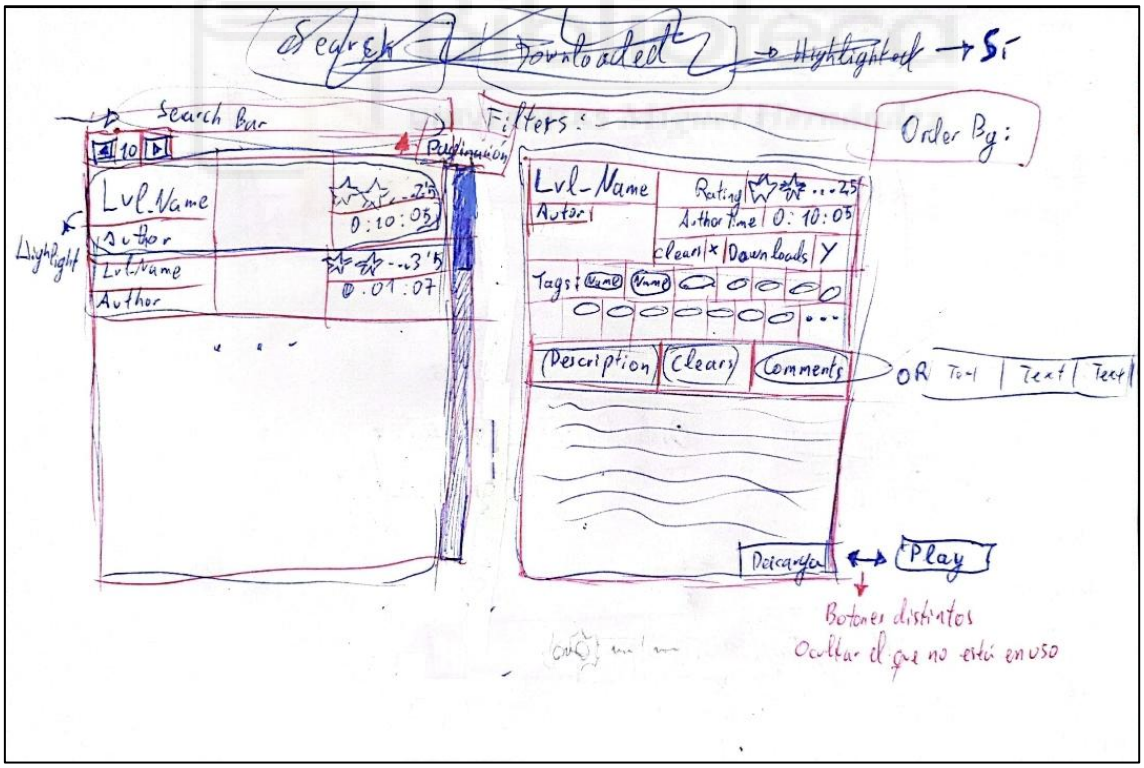

Figura II.6: Prototipo de Menú de la Comunidad

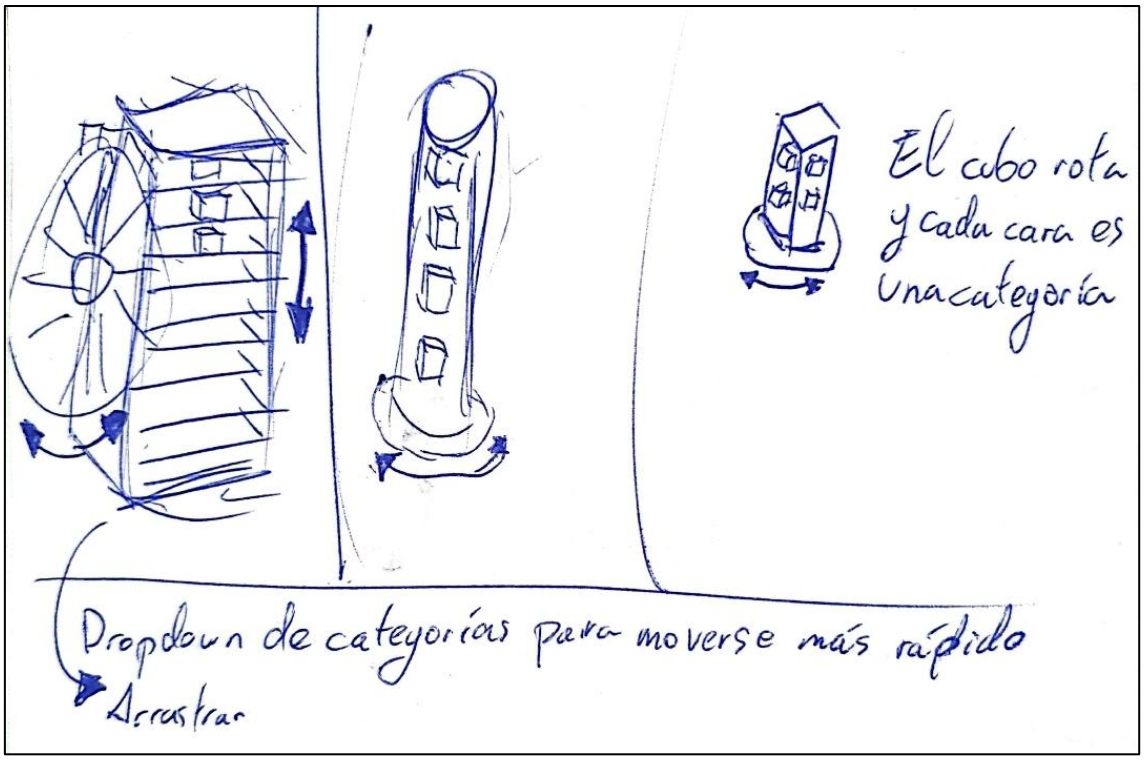

Figura II.7: Prototipo del Menú de Ítems

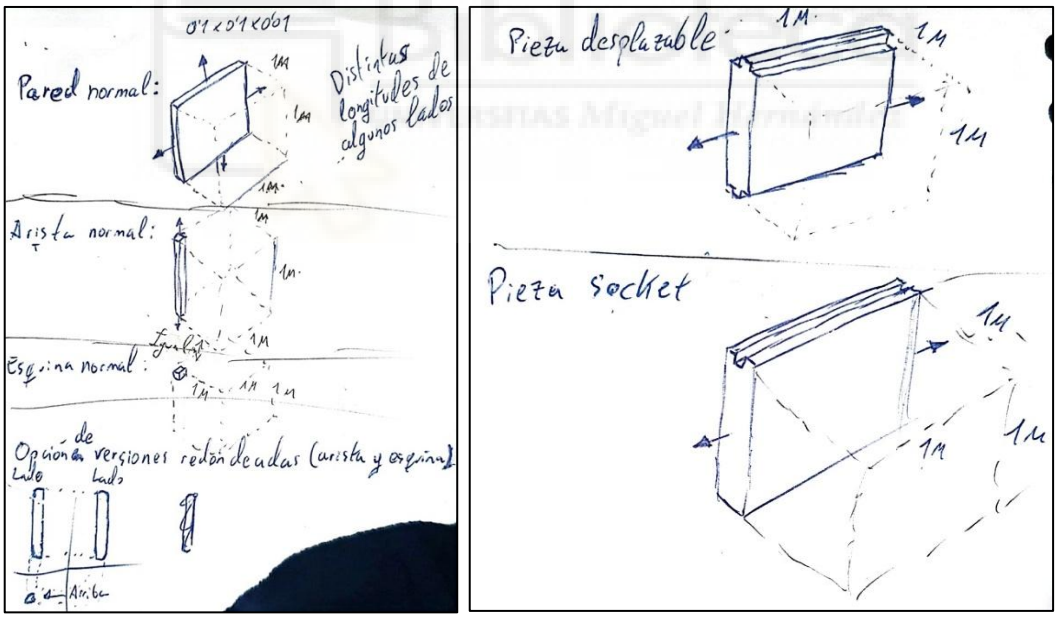

Figura II.8: Prototipo de Ítems Estáticos y Móviles para el Diseño de Niveles

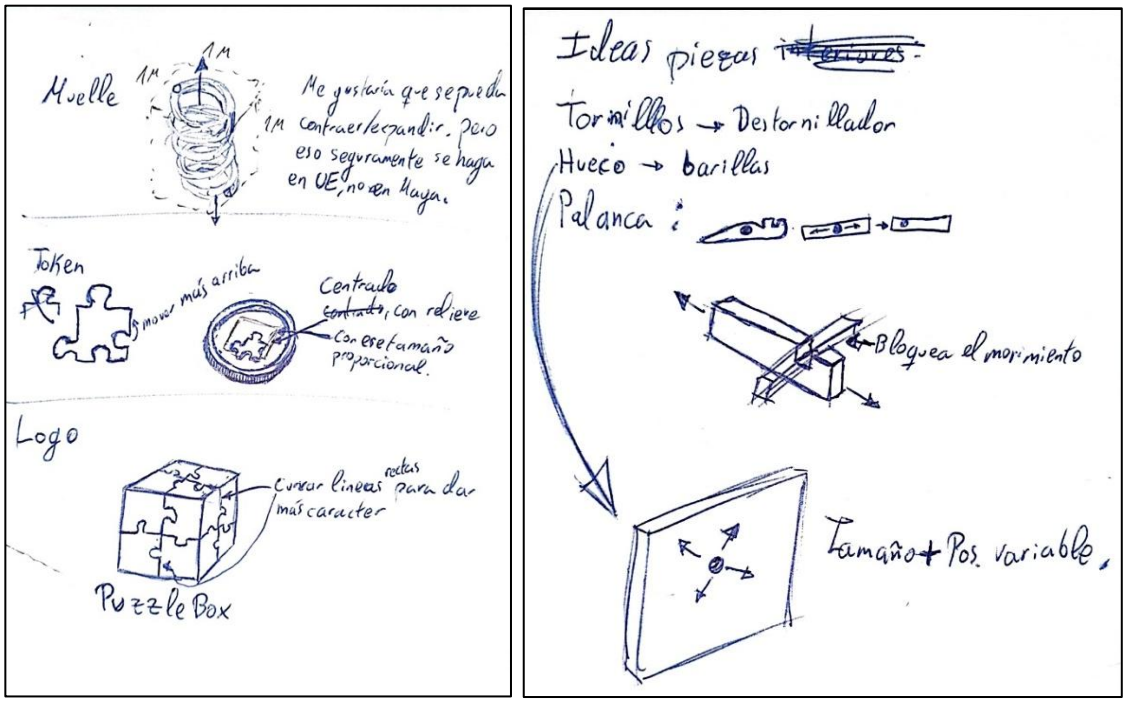

Figura II.9: Prototipo de Ítems Móviles y Logo del Juego

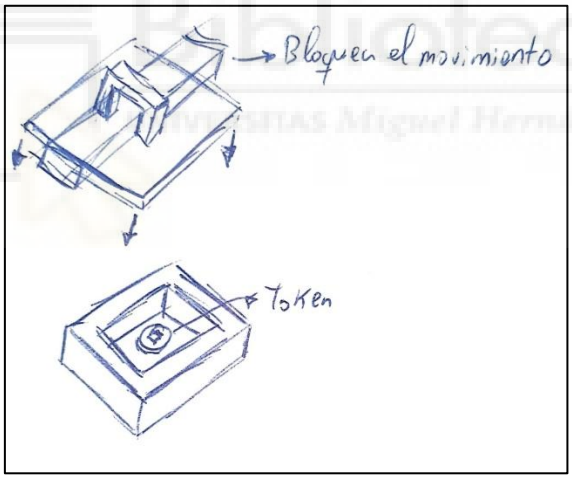

Figura II.10: Prototipo de Objetos Móviles para el Diseño de Niveles

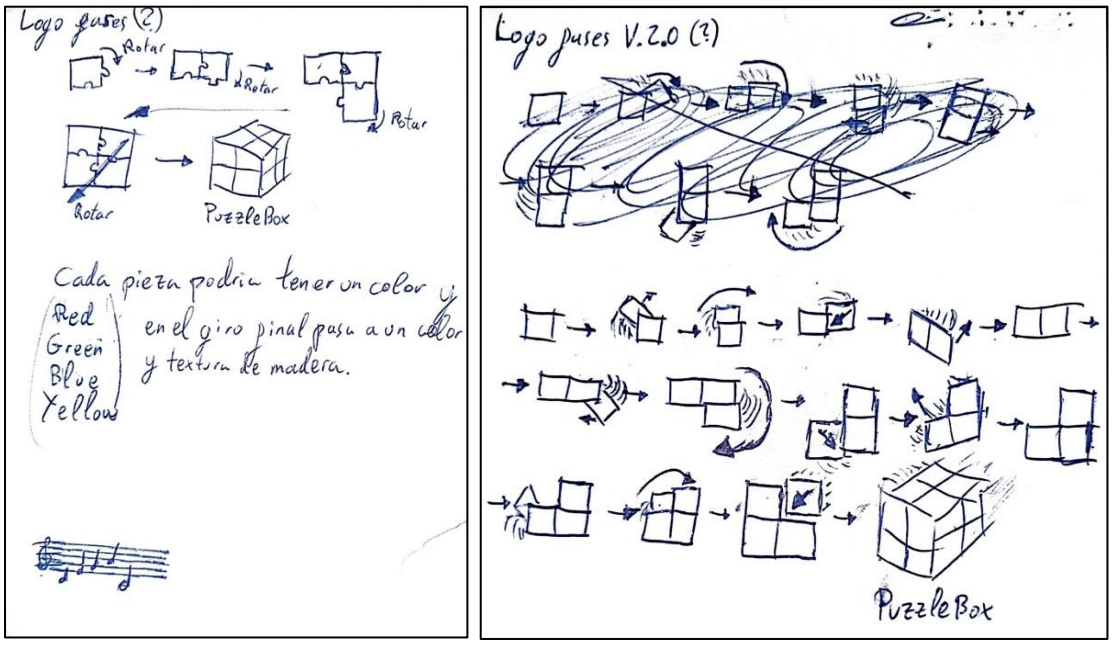

Figura II.11: Prototipo de Secuencia de Lanzamiento del Juego

Figura II.12: Prototipos Digitales del Logo

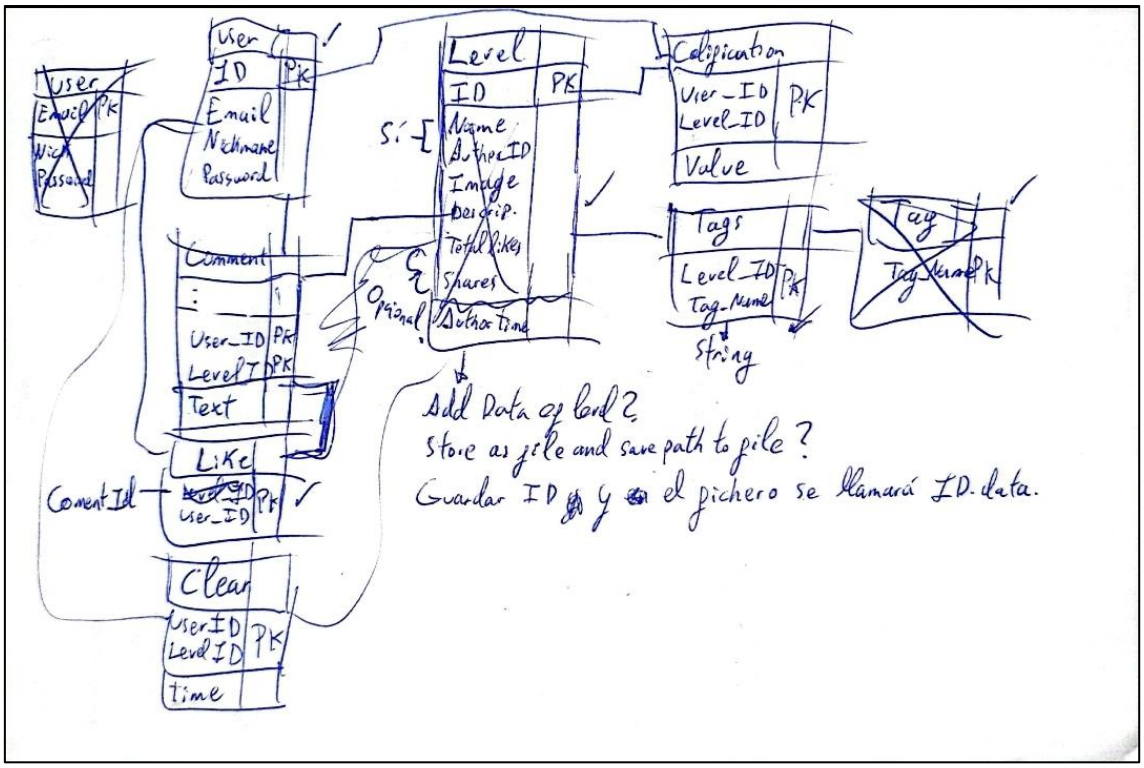

Figura II.13: Prototipo de Base de Datos a Papel

ANEXO III: Diseño de menús en UE

- **Figura III.1:** Este es el menú de inicio que aparece cuando se inicia el juego sin haber iniciado sesión.

- **Figura III.2:** Este es el menú que se usa para registrar una cuenta.

- **Figura III.3:** Este es el menú que se usa para iniciar sesión.

- **Figura III.4:** Este es el menú principal que se ve cuando se tiene sesión iniciada.

- **Figura III.5:** Este es el menú de campaña, donde están los niveles creados por el desarrollador.

- **Figura III.6:** Este es el menú de la comunidad, donde se pueden visualizar los niveles creados y validados por los jugadores, en esta figura se puede ver la clasificación de los usuarios que han completado el nivel.

- **Figura III.7:** Esta figura es igual que la anterior, pero muestra los comentarios puestos por los usuarios.

- **Figura III.8:** Este menú muestra los niveles creados por el usuario, y que están almacenados de manera local.

- **Figura III.9:** Este menú muestra los bloques no móviles que se usan en el editor de niveles.

- **Figura III.10:** Este menú muestra los bloques móviles que se usan en el editor de niveles.

- **Figura III.11:** Este menú muestra los bloques de finalización de nivel que se usan en el editor de niveles, estos bloques son los que marcan el nivel como completado.

- **Figura III.12:** Este es el menú que se usa para editar el material de los bloques en el editor de niveles.

- **Figura III.13:** Este es el menú que se usa para editar el nombre, la descripción y los tags del nivel.

- **Figura III.14:** Este es el menú que se usa para editar la configuración del grid del editor de niveles.

- **Figura III.15:** Este es el menú donde se muestra la configuración del juego, como la resolución, volumen del sonido y música, asignación de los botones de los controles, etc.

- **Figura III.16:** Este es el menú de pausa que ve un usuario mientras juega un nivel.

- **Figura III.17:** Este es el menú de pausa que ve un usuario mientras está editando un nivel.

- **Figura III.18:** Este es el menú de pausa que ve un usuario mientras está validando un nivel.

- **Figura III.19:** Este es el menú que ve el usuario cuando ha terminado de validar un nivel.

- **Figura III.20:** Este es el menú que ve un usuario cuando intenta salir del editor sin haber guardado los cambios.

- **Figura III.21:** Este es el menú que hace de teclado para escribir en las interfaces que lo permiten.

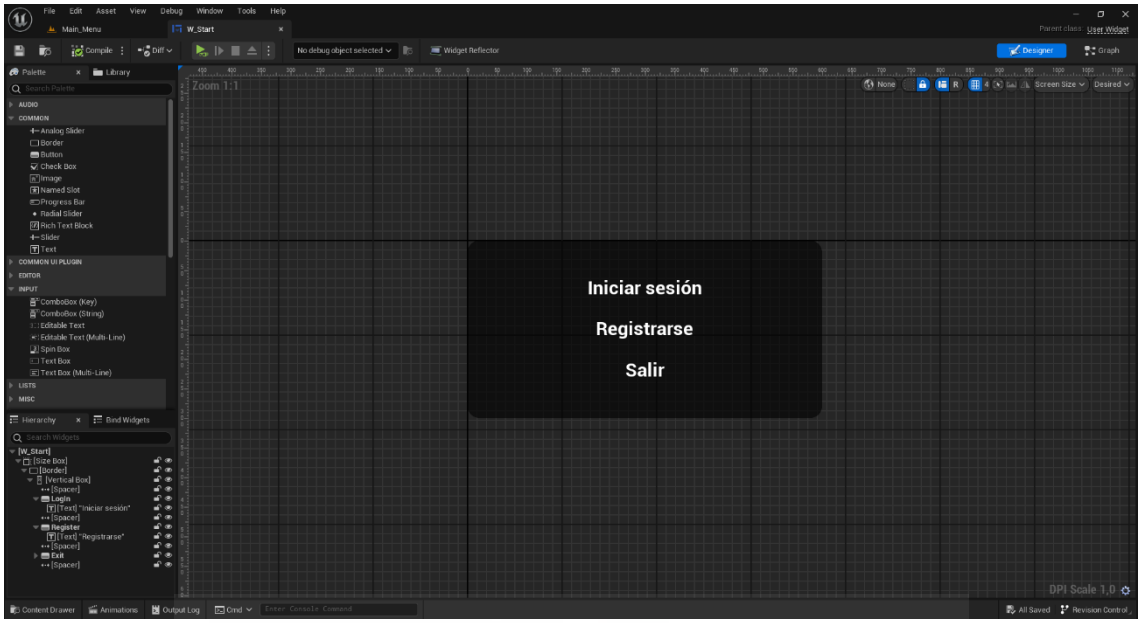

Figura III.1: Menú de Inicio

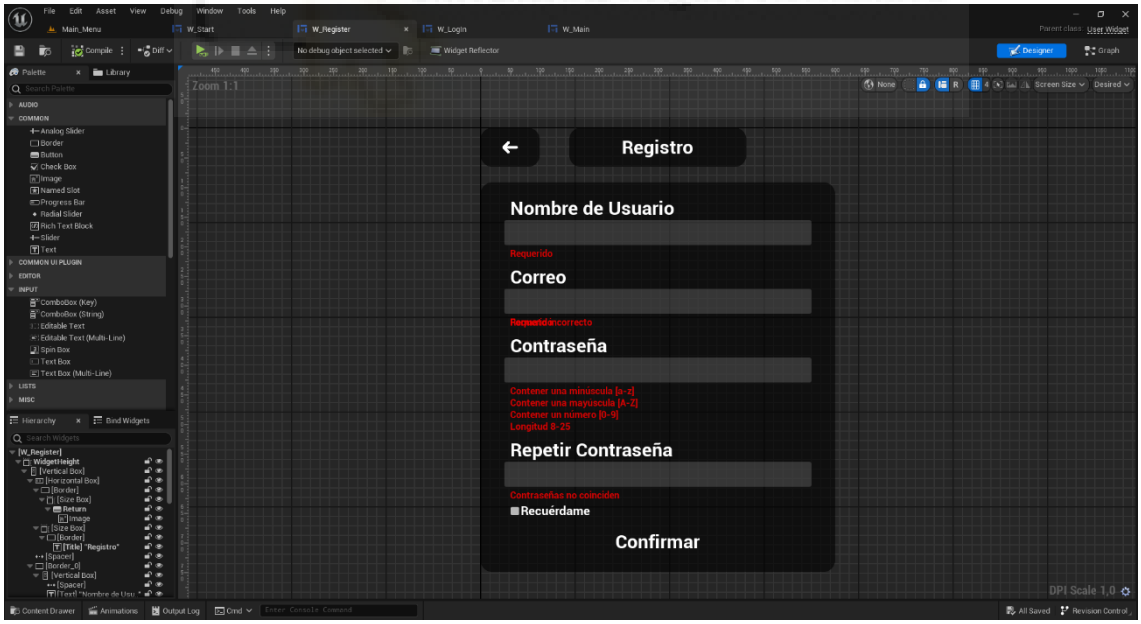

Figura III.2: Menú de Registro

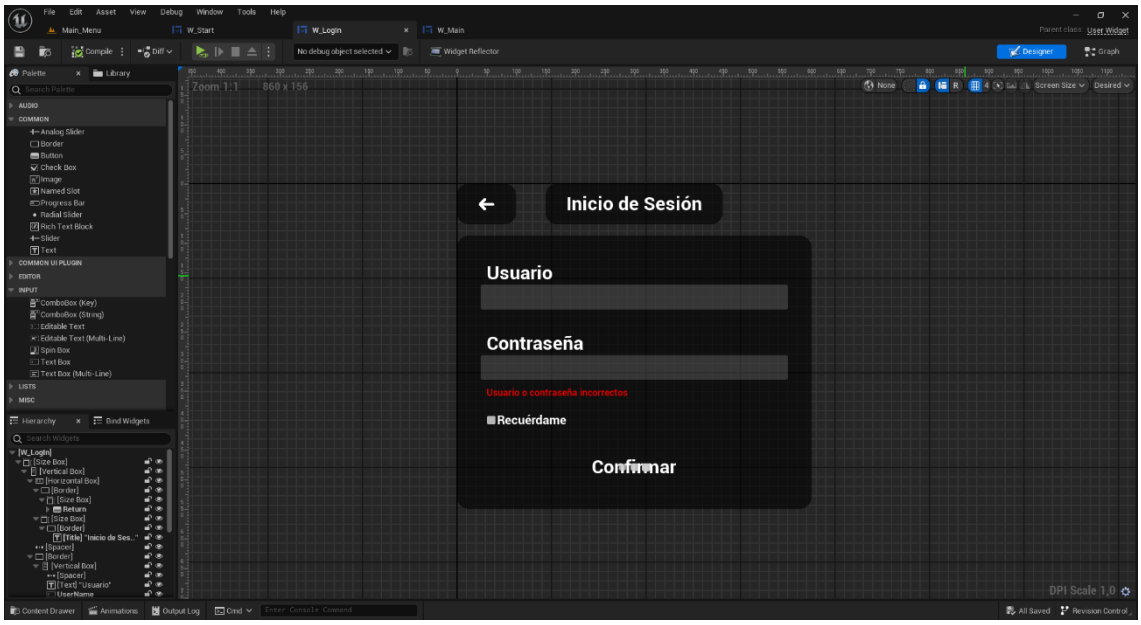

Figura III.3: Menú de Inicio de Sesión

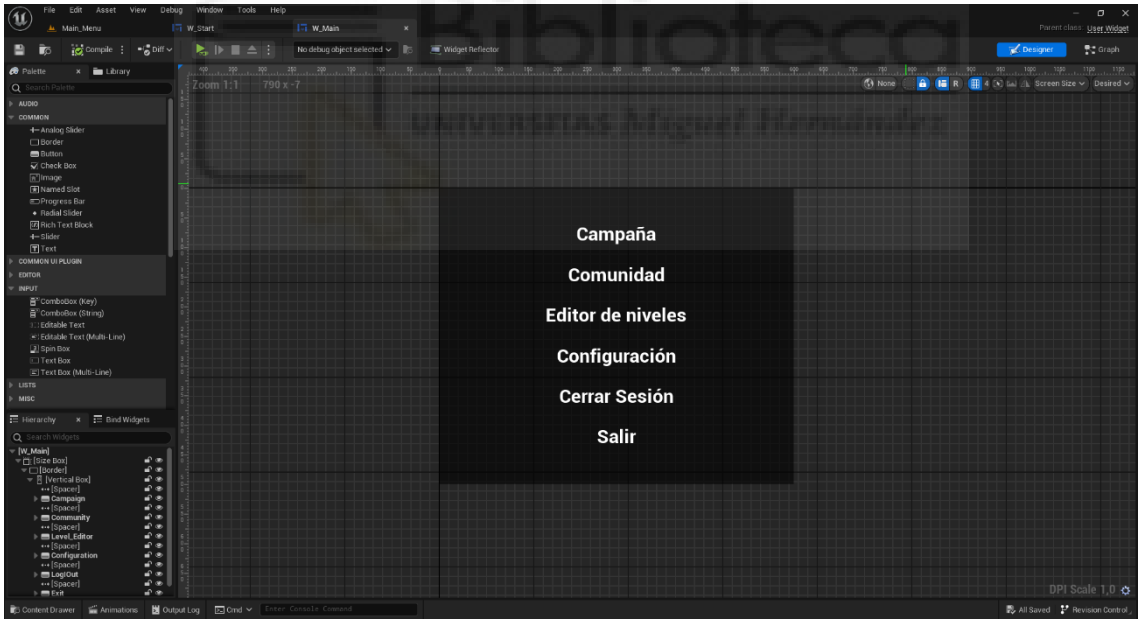

Figura III.4: Menú Principal

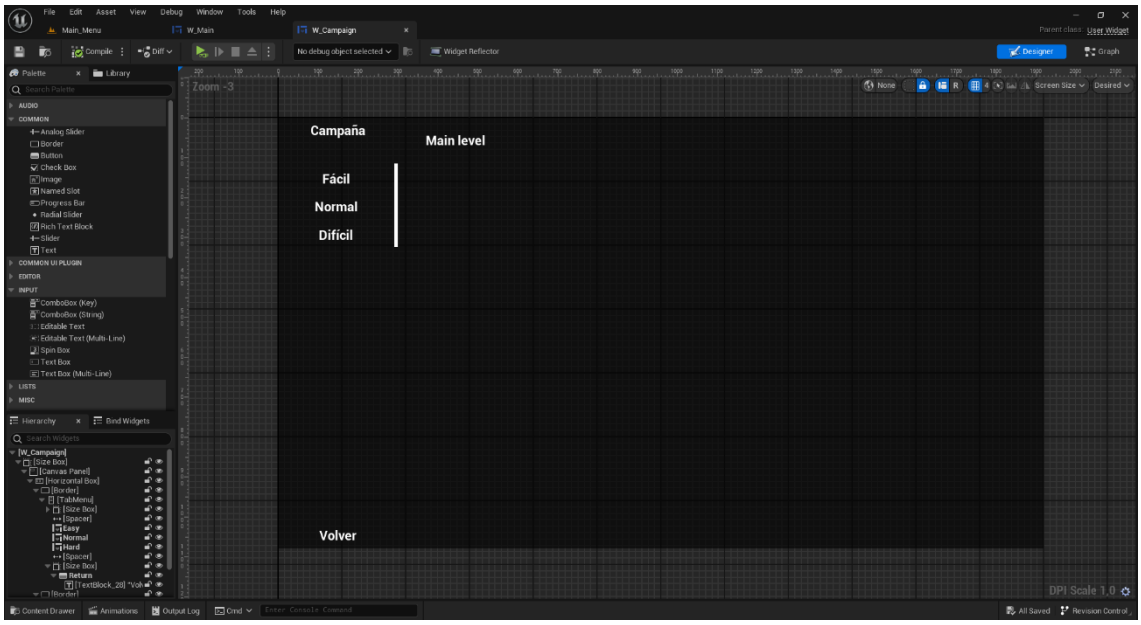

Figura III.5: Menú de Campaña

Figura III.6: Menú de Comunidad, Clasificaciones en un Nivel

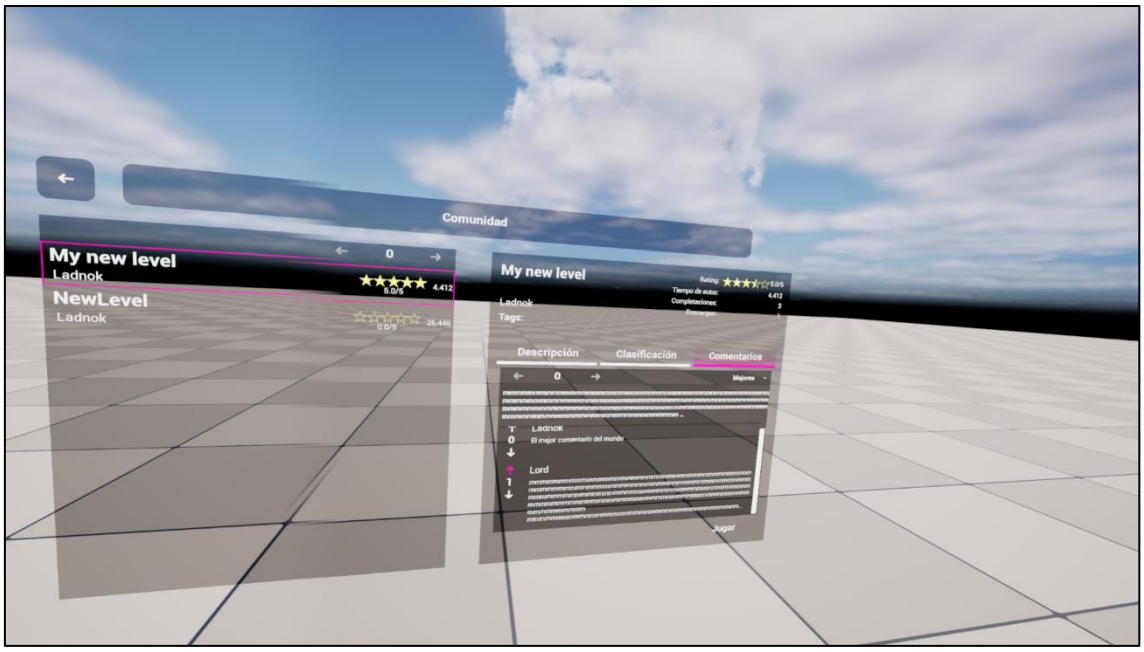

Figura III.7: Menú de Comunidad, Comentarios en un Nivel

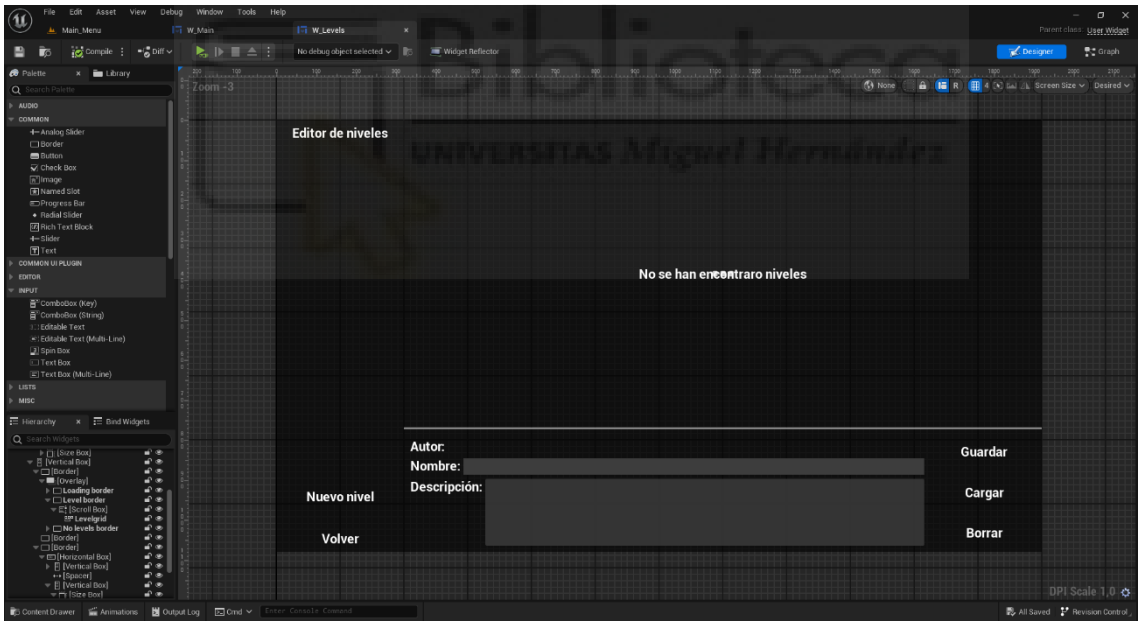

Figura III.8: Menú de Puzles Locales

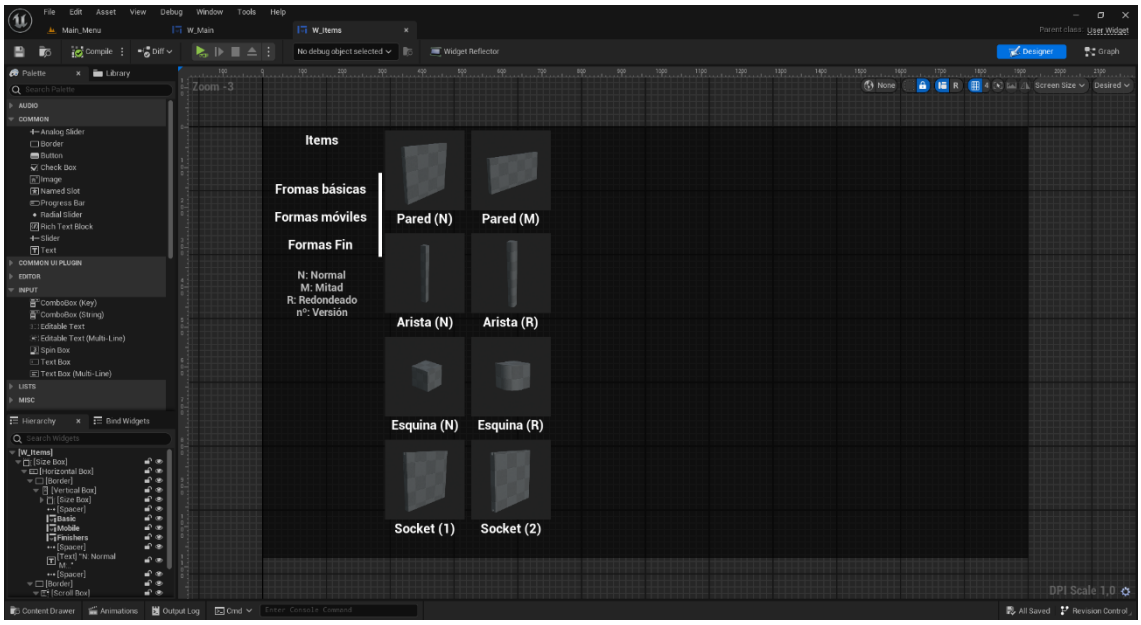

Figura III.9: Menú de Ítems no Móviles

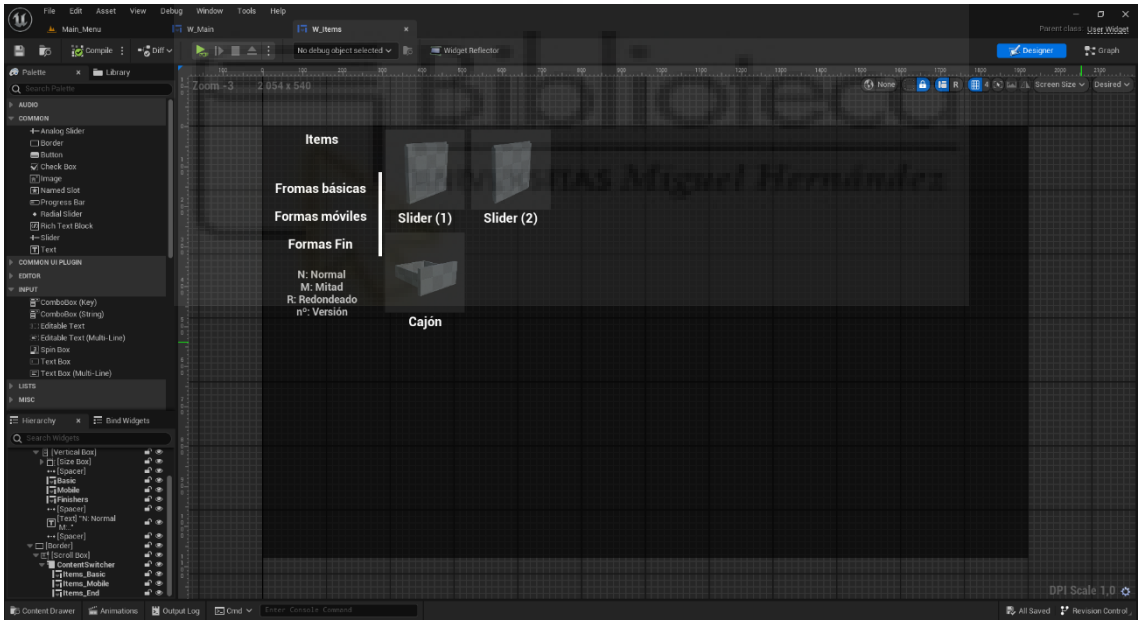

Figura III.10: Menú de Ítems Móviles

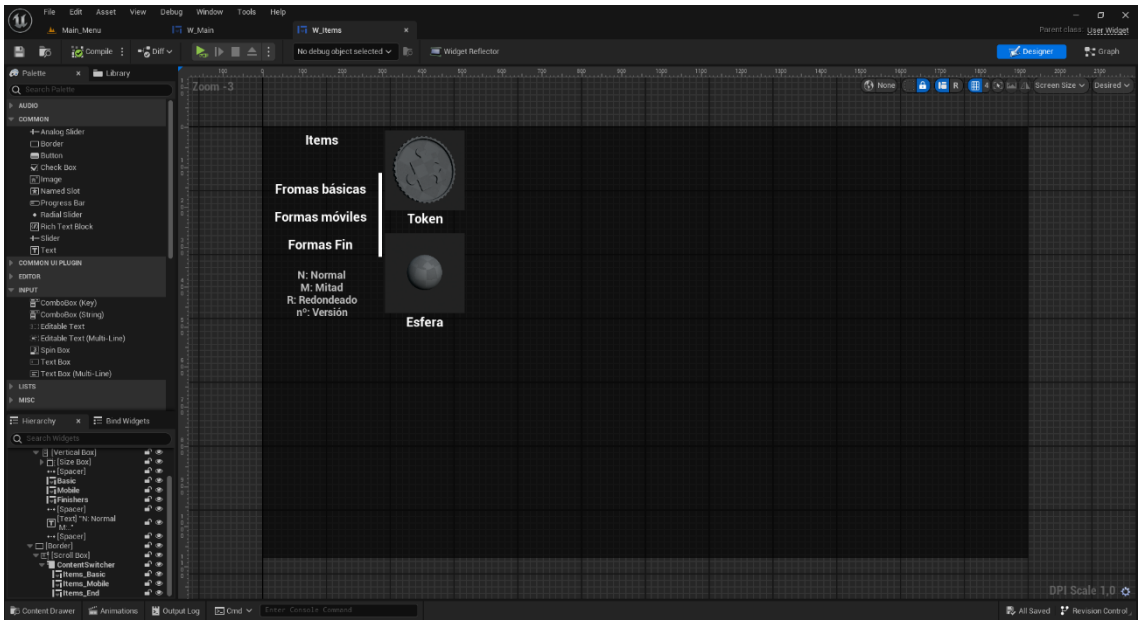

Figura III.11: Menú de Ítems de Finalización de Nivel

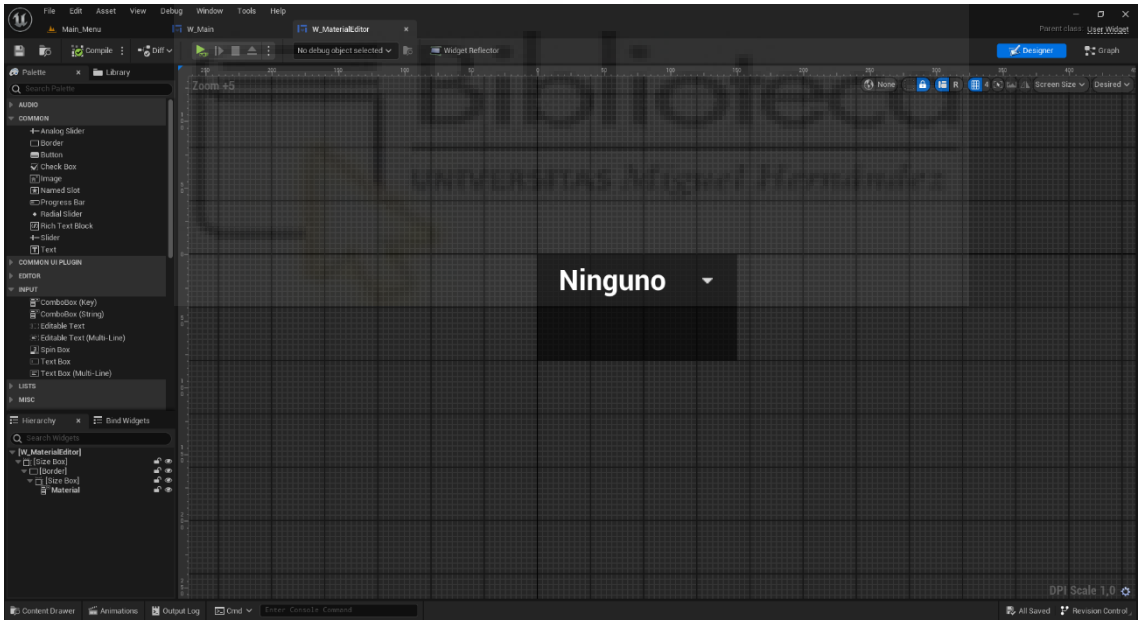

Figura III.12: Menú de Material Aplicable a un Ítem

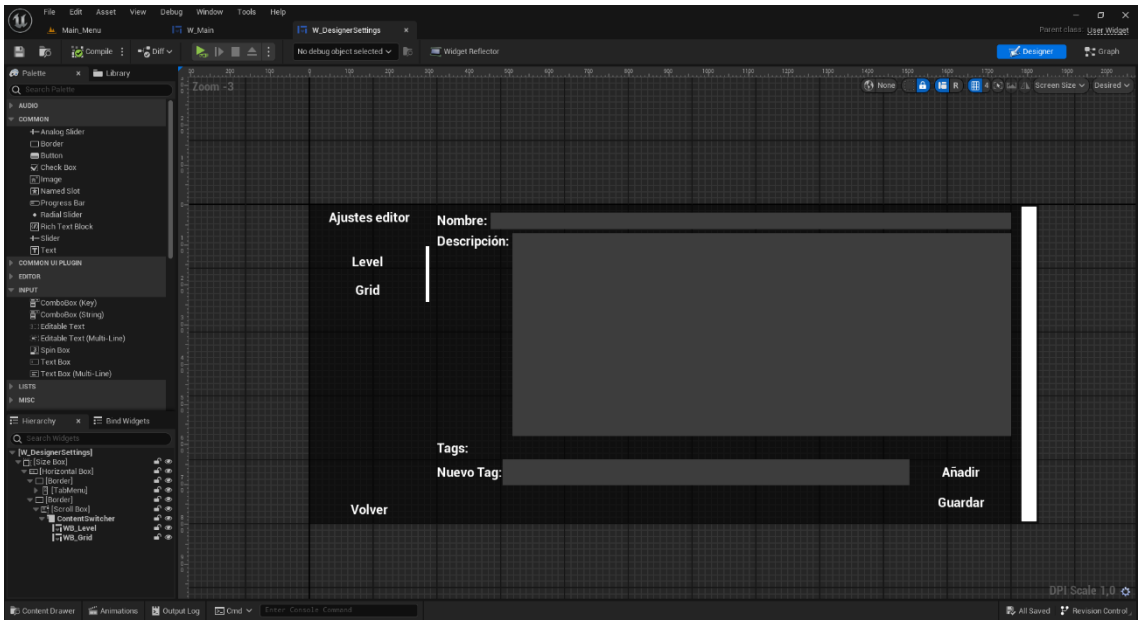

Figura III.13: Menú de Datos del Puzle

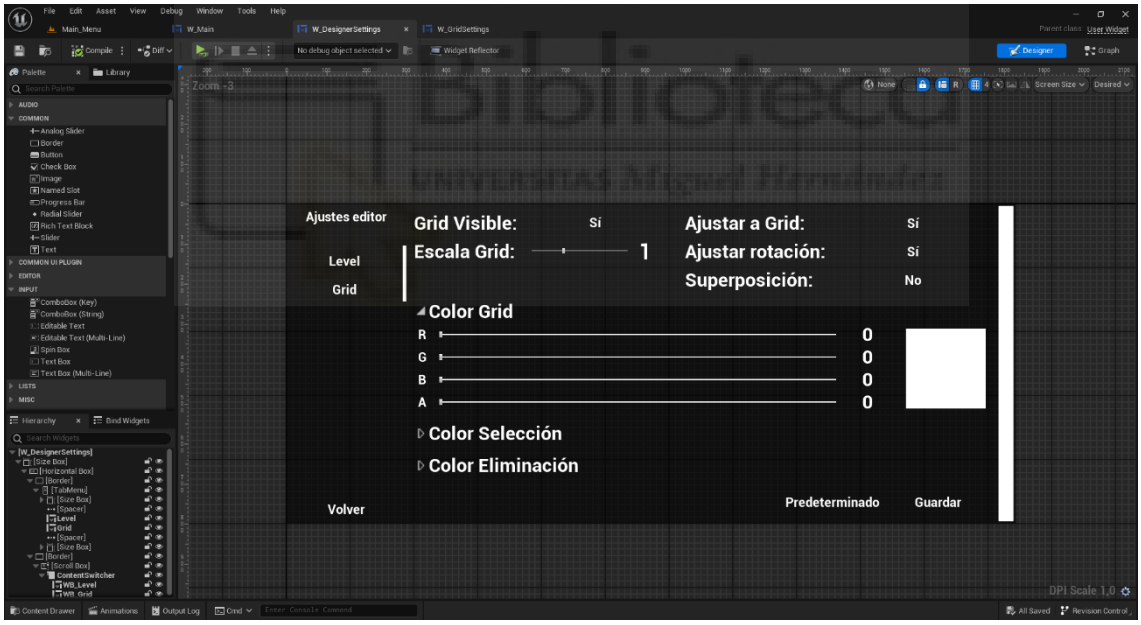

Figura III.14: Menú de Ajustes del Grid

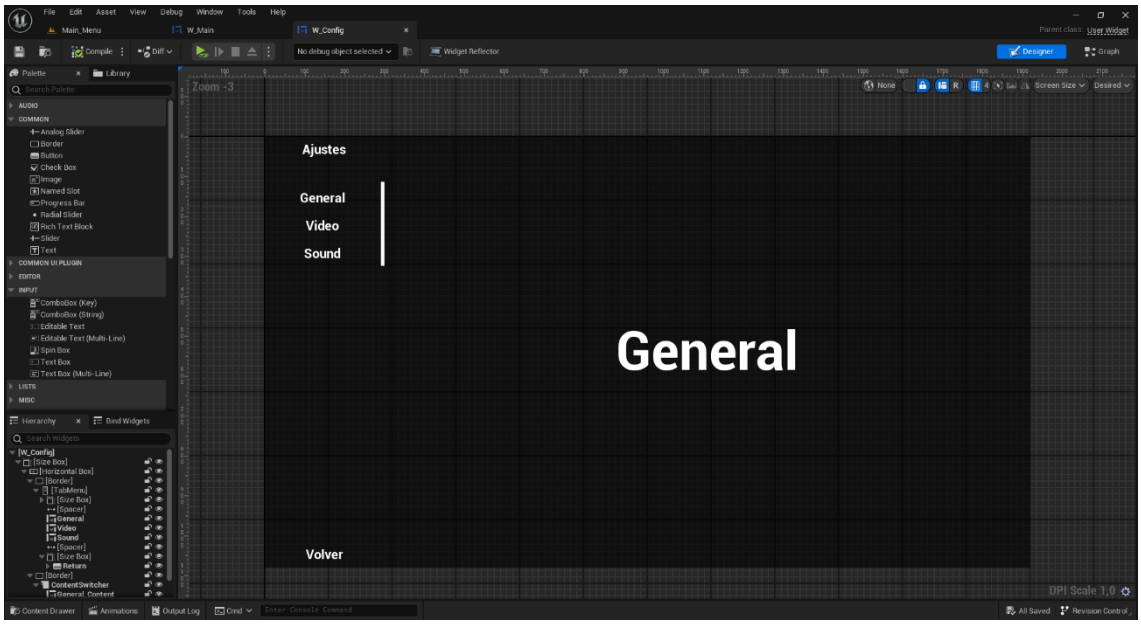

Figura III.15: Menú de Configuración

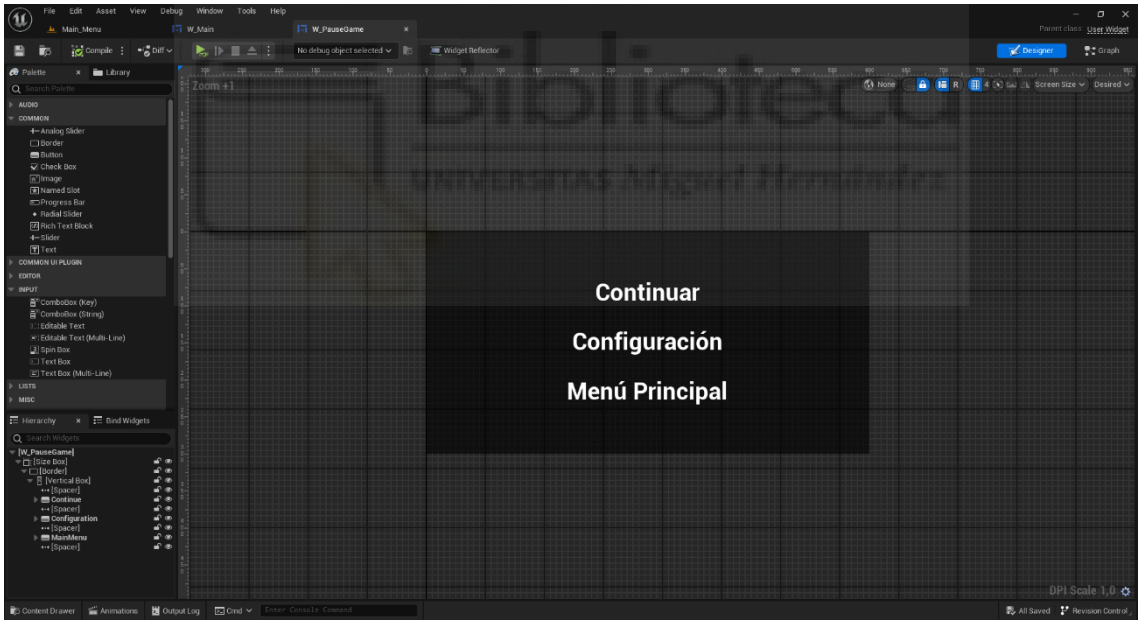

Figura III.16: Menú de Pausa Durante Partida

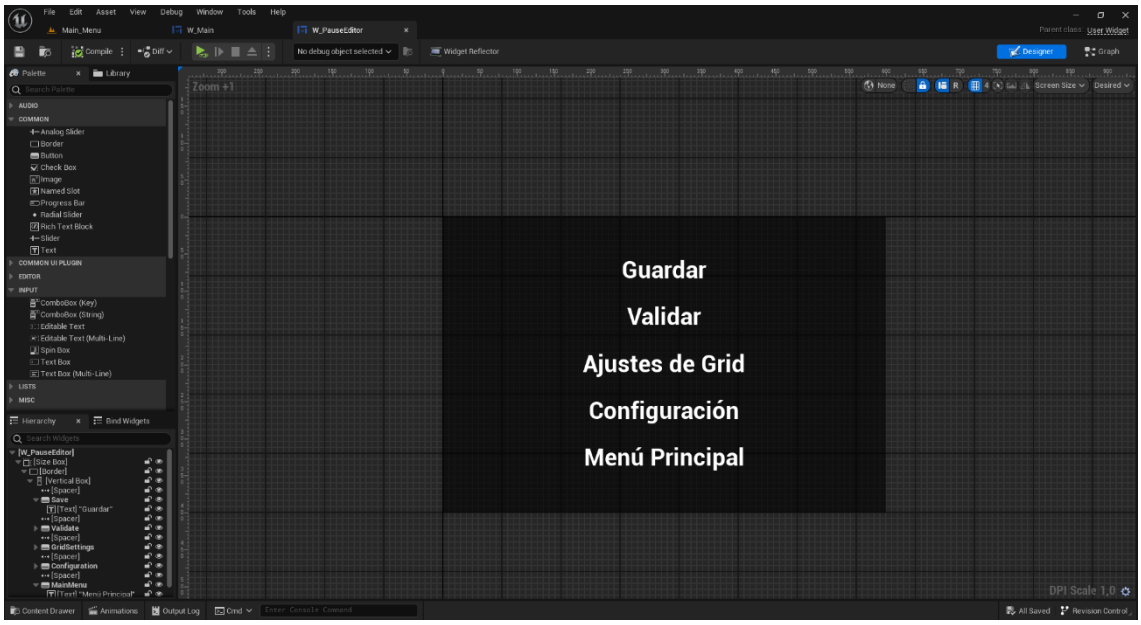

Figura III.17: Menú de Pausa Durante Creación de un Nivel

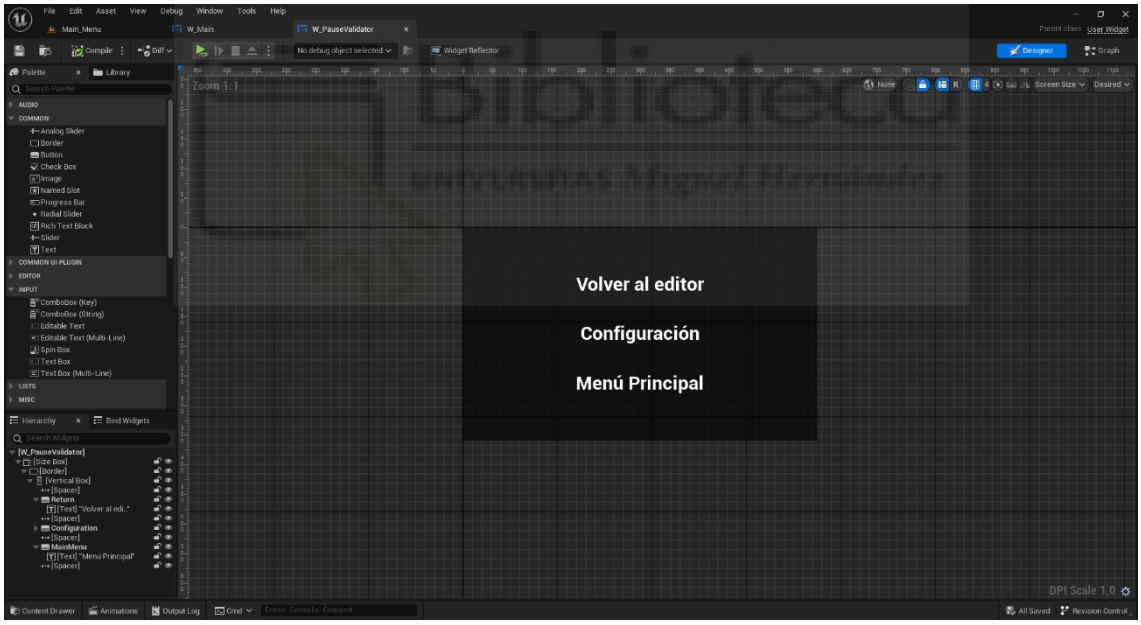

Figura III.18: Menú de Pausa Durante Validación de un Nivel

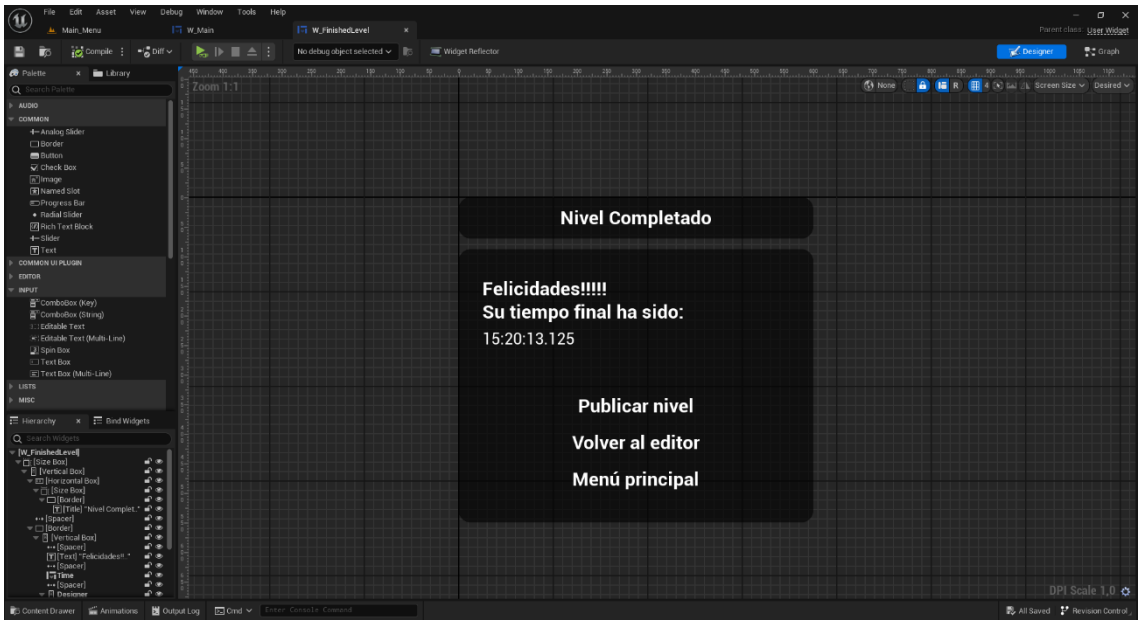

Figura III.19: Menú Post Validación de un Puzle

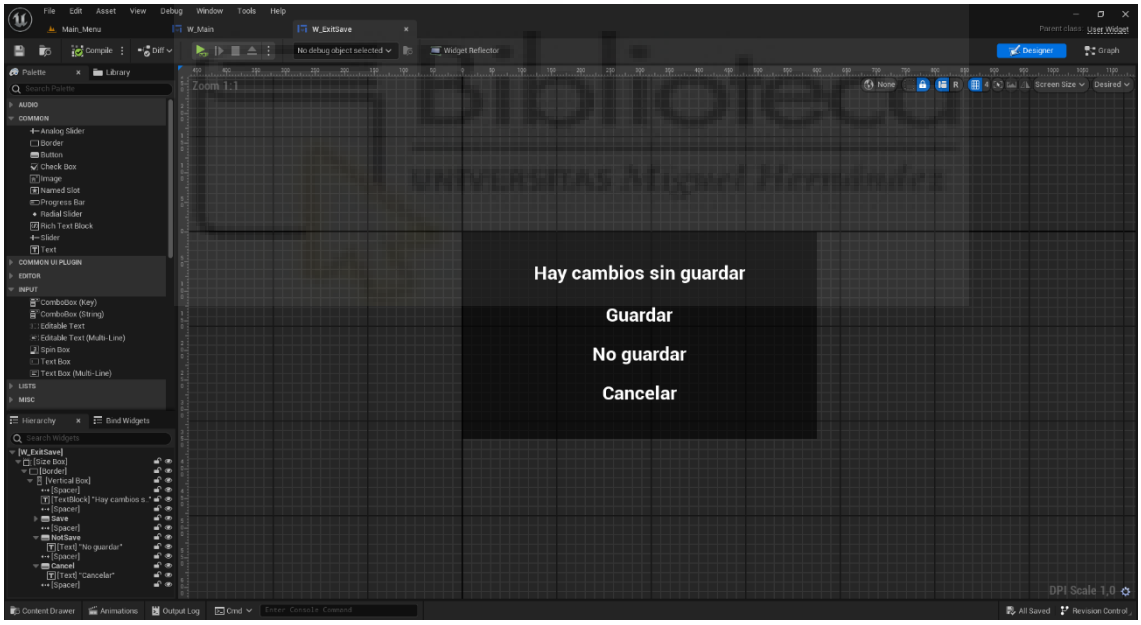

Figura III.20: Menú de Salir del Editor sin Guardar

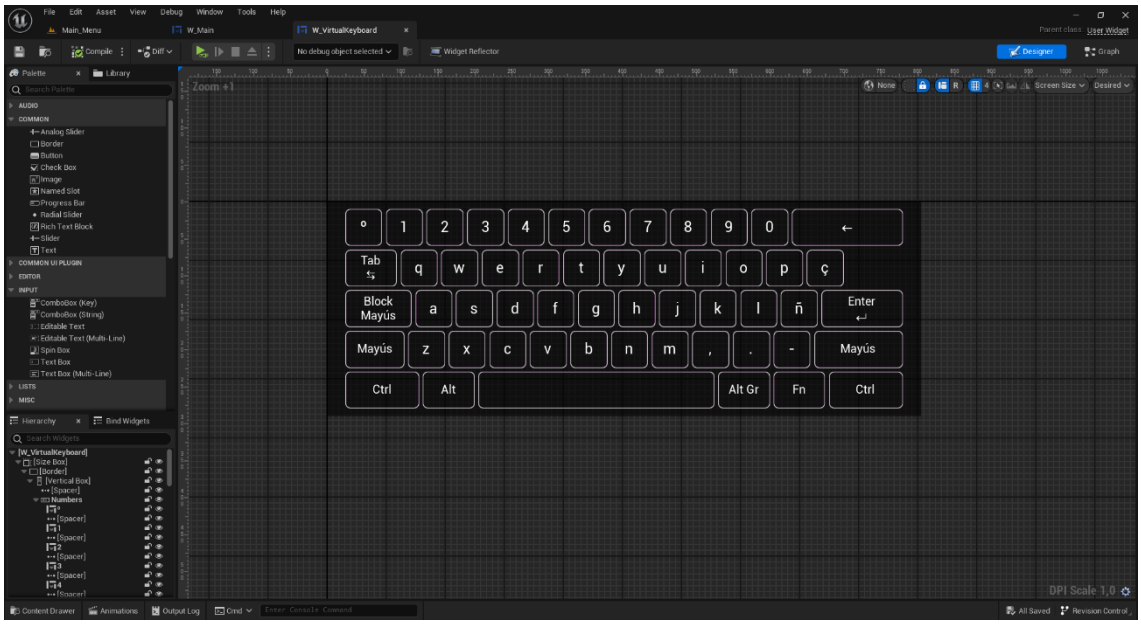

Figura III.21: Teclado Virtual

ANEXO IV: Documentación Swagger

Figura IV.1: Esta figura muestra los endpoints disponibles en el controlador “Clears”, estos hacen lo siguiente:

1. Añade una completación nueva en la base de datos, asociándola al usuario que envía la petición.
2. Devuelve una lista de completaciones realizadas en un nivel, filtrando por paginación.

Figura IV.2: Esta figura muestra los endpoints disponibles en el controlador “Levels”, estos hacen lo siguiente:

1. Devuelve todos los datos de un nivel.
3. Añade un nivel nuevo a la base de datos, asociándolo al usuario que envía la petición.
4. Actualiza los datos de un nivel ya existente.
5. Devuelve una lista de niveles, en función de los filtros enviados.
6. Devuelve el fichero de un nivel, que luego es usado por el juego para cargar la partida.
7. Compara el hash de un nivel con el almacenado en la base de datos, lo que permite validar si un nivel no ha sido alterado antes de comenzar la partida.
8. Comprueba si un nivel ya existe para un usuario especifico.

Figura IV.3: Esta figura muestra los endpoints disponibles en el controlador “Likes”, estos hacen lo siguiente:

1. Añade un like nuevo a la base de datos, asociándolo al usuario que envía la petición.
2. Actualiza el estado de un like ya existente.
3. Devuelve los likes realizados por un usuario sobre los comentarios de un nivel.

Figura IV.4: Esta figura muestra los endpoints disponibles en el controlador “Ratings”, estos hacen lo siguiente:

1. Añade un nuevo rating a un nivel, asociándolo al usuario que envía la petición.
2. Actualiza el estado de un rating ya existente.
3. Devuelve el rating dado por un usuario sobre un nivel.

Figura IV.5: Esta figura muestra los endpoints disponibles en el controlador “Users”, estos hacen lo siguiente:

1. Añade un nuevo usuario a la base de datos.
2. Comprueba los datos de inicio de sesión y genera un JWT para la sesión.
3. Comprueba la validez y estado del JWT.

Figura IV.6: Esta figura muestra todos los schemas usados para recibir datos en los endpoints, los más destacables son los siguientes:

1. Blocks: Contiene toda la información de los bloques de un nivel, como su posición, orientación, material, escala, etc.
2. ListerDTO: Se utiliza para filtrar y ordenar las listas que son enviadas en la respuesta, hace uso de un paginador, un listado de filtros y el orden a usar (ASC, DESC y que se usa como parametro).

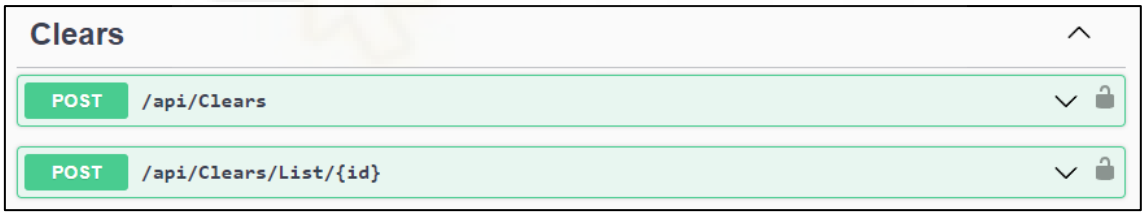

Figura IV.1: Rutas del Controlador "Clears"

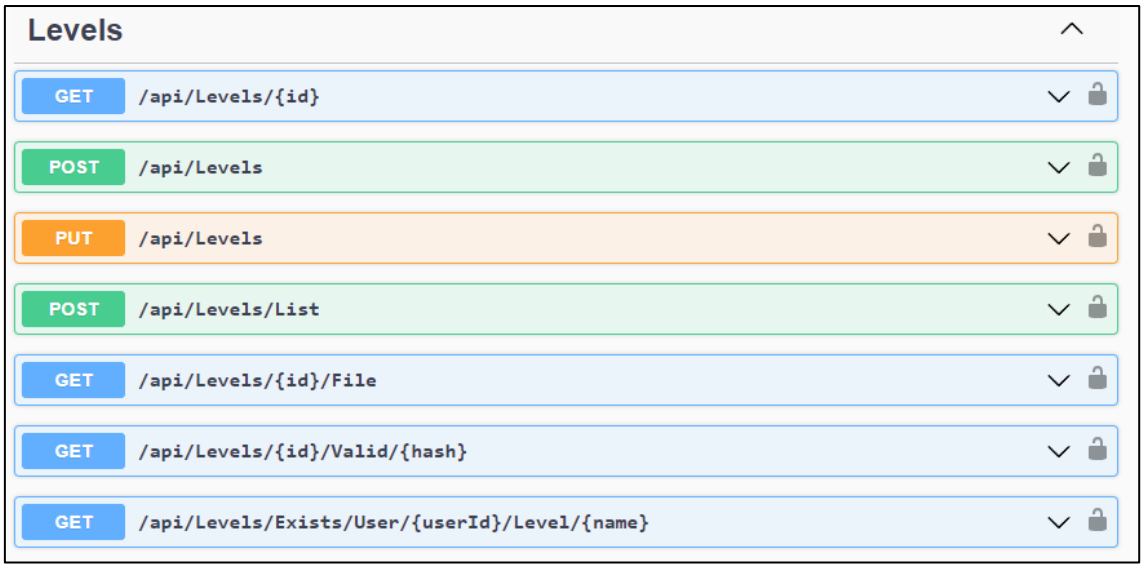

Figura IV.2: Rutas del Controlador "Levels"

Figura IV.3: Rutas del Controlador "Likes"

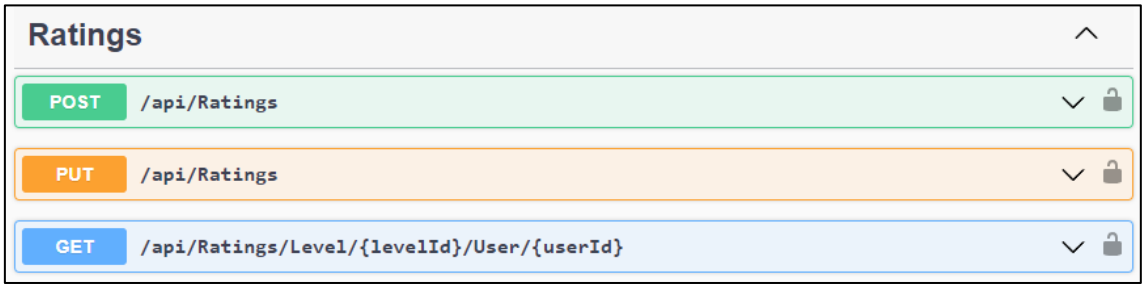

Figura IV.4: Rutas del Controlador "Ratings"

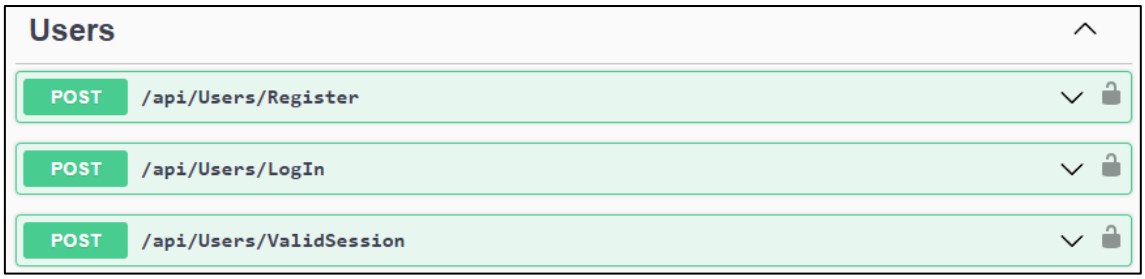

Figura IV.5: Rutas del Controlador "Users"

Figura IV.6: Documentación de Schemas Generada por Swagger