



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**VRRecoveryGym: entorno de
realidad virtual para personas
con trastornos motores
Documentación Técnica**



Presentado por Rodrigo Grande Muñoz
en Universidad de Burgos — 30 de junio
de 2025

Tutor: Álgvar Arnaiz González, Héctor Royo
Concepción y Luis Fuente Ibáñez

Índice general

Índice general	i
Índice de figuras	iii
Índice de tablas	iv
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
A.3. Estudio de viabilidad	7
Apéndice B Especificación de Requisitos	15
B.1. Introducción	15
B.2. Objetivos generales	16
B.3. Catálogo de requisitos	16
B.4. Especificación de requisitos	18
Apéndice C Especificación de diseño	31
C.1. Introducción	31
C.2. Diseño de datos	31
C.3. Diseño arquitectónico	34
C.4. Diseño procedimental	35
Apéndice D Documentación técnica de programación	37
D.1. Introducción	37
D.2. Estructura de directorios	37
D.3. Manual del programador	37

D.4. Compilación, instalación y ejecución del proyecto	37
D.5. Pruebas del sistema	37
Apéndice E Documentación de usuario	39
E.1. Introducción	39
E.2. Requisitos de usuarios	39
E.3. Instalación	39
E.4. Manual del usuario	39
Apéndice F Anexo de sostenibilización curricular	41
F.1. Introducción	41
Bibliografía	43

Índice de figuras

B.1. Diagrama: casos de uso	19
C.1. Diagrama de secuencia de registro	36

Índice de tablas

A.1. Resumen de costes estimados del proyecto	10
A.2. Licencias de dependencias y recursos del proyecto	13
B.1. CU-1 Listar partidas.	20
B.2. CU-2 Cargar partida guardada.	21
B.3. CU-3 Borrar partida.	22
B.4. CU-4 Mostrar pantalla de créditos y agradecimientos.	23
B.5. CU-5 Elegir juego (nivel).	24
B.6. CU-6 Mostrar resultados históricos de sesión asociados al puzzle seleccionado.	25
B.7. CU-7 Exportar a CSV.	26
B.8. CU-8 Realizar sesión de juego.	27
B.9. CU-9 Realizar sesión de juego: Palacio de Shahriar	28
B.10. CU-10 Consulta de datos.	29
C.1. Diccionario de datos: BP_PlayerSave	33
C.2. Diccionario de datos: BP_PalaceSave	33

Apéndice A

Plan de Proyecto Software

A.1. Introducción

El presente anexo recoge los aspectos fundamentales del plan de desarrollo seguido durante la realización del proyecto. En él se detalla la planificación temporal adoptada, basada en la metodología ágil *Scrum*, así como un análisis de la viabilidad económica y legal del sistema desarrollado.

La planificación con *Scrum* ha permitido dividir el trabajo en iteraciones controladas, fomentar la adaptación continua a los cambios propuestos por el cliente, y mantener una visión clara del avance del proyecto a lo largo del tiempo. Se presentan los *sprints* realizados, el *product backlog*, y los artefactos generados durante las distintas fases de desarrollo.

Por otro lado, se incluye un estudio de la viabilidad económica, donde se estiman los costes asociados al desarrollo, licencias, recursos y herramientas utilizadas. También se expone la viabilidad legal, analizando las licencias de software empleadas y definiendo un marco legal que permita el uso, distribución o ampliación futura del sistema respetando los derechos de terceros.

Este anexo sirve como complemento técnico y organizativo a la memoria principal, aportando una visión estructurada de la gestión del proyecto desde una perspectiva profesional.

A.2. Planificación temporal

La planificación temporal de este proyecto, debido a que se está siguiendo la metodología Scrum, será realizada mediante *sprints*.

Sprint 1: Planificación del proyecto y estructura del juego

- Planning meeting: Durante la reunión se marcaron los siguientes objetivos:
 1. **Revisión de documentación de puzzles:** Leer y analizar toda la documentación relacionada con los puzzles para comprender los requisitos, mecánicas y desafíos que deberán implementarse.
 2. **Planificación de la estructura del juego:** Definir la arquitectura general del juego, estableciendo los módulos principales, el flujo de navegación y la integración de los puzzles.
 3. **Selección de gafas de realidad virtual:** Acordar la versión de las gafas de *RV* a utilizar y definir la preferencia basada en la compatibilidad, rendimiento y experiencia de usuario.
- Duración del sprint: El *sprint* se realizó entre el 12 de marzo de 2025 y el 17 de marzo de 2025, con una duración de 7 días.
- Burndown Report: En este *sprint* se cumplió con casi todos los objetivos en el plazo determinado. La selección de las gafas de realidad virtual se veía retrasada debido a que la persona que iba a prestarlas para este proyecto no se encontraba disponible en ese momento. Cabe destacar que durante la duración de este *sprint* también se realizaron tareas de aprendizaje sobre Unreal Engine ya que los conocimientos eran nulos sobre este motor gráfico.
- Sprint review meeting Durante esta reunión, se resolvieron dudas sobre el sistema de registro que debía tener la aplicación, como se querían estructurar las partidas y el requisito de toma de algunos datos que pedía la organización ASPAYM que eran inviables en el nivel de este proyecto. Los *issues* fueron creados de forma correcta, pero se marcó que había que planificar los *sprints* correspondientes al proyecto.

Sprint 2: Menú principal

- Planning meeting:

Durante la reunión se marcaron los siguientes objetivos:

1. **Registro de usuarios de forma local:** El registro se realizará mediante el *savegame* de la partida, donde, si es la primera vez que se inicia el juego en el dispositivo, solo se pedirá al usuario el nombre. De esta manera, se asegura que el registro sea rápido y sencillo.
2. **Creación de nuevas partidas:** Cada partida será asociada a un nivel, el cual se seleccionará como el puzle específico que se quiere jugar. Esto implica que cada partida estará vinculada a un puzle que servirá como contenido principal de la misma.
3. **Listado de las partidas creadas:** La aplicación deberá permitir visualizar una lista de las partidas y proporcionar información básica sobre cada puzle, como el nombre del puzle, el nivel y el estado del progreso (si corresponde).

- Duración del sprint:

El *sprint* se realizó entre el 26 de marzo de 2025 y el 9 de abril de 2025, con una duración de 14 días.

- Burndown Report:

En este *sprint* se cumplió con los objetivos en el plazo determinado. La estimación de tiempo fue muy extensa debido a que se utilizó este plazo para seguir adquiriendo conocimientos sobre *Unreal Engine* mientras se realizaban los objetivos del sprint.

- Sprint review meeting

Se habló sobre la importancia de agrupar los elementos de la interfaz en *plantillas* como los botones e *items* para la reusabilidad, ya que no tuve en cuenta algunos elementos por lo cual ese aspecto se planteó cambiar para el siguiente *sprint*.

Sprint 3: Plantilla de puzle e integración de menú principal a VR

- Planning meeting:

Durante la reunión se marcaron los siguientes objetivos:

1. **Integración de interfaces a realidad virtual:** Adaptar y conectar todas las pantallas y controles del juego al entorno RV, garantizando la navegación y la interacción fluida.
2. **Contador de tiempo y almacenamiento de nivel:** Implementar un temporizador que comience al iniciar el puzle y guarde el tiempo transcurrido junto con los datos del nivel creado en la sesión.
3. **Cargado de una partida:** Desarrollar la funcionalidad para seleccionar y restaurar el estado de una partida previamente guardada, incluyendo el progreso y el nivel activo.
4. **Visualización de resultados del puzle:** Mostrar al finalizar cada puzle los siguientes datos:
 - Tiempo empleado
 - Fecha de creación de la sesión
 - Nombre del puzle jugado
 - Completado o no completado

- Duración del *sprint*

El *sprint* se realizó entre el 9 de abril de 2025 y el 23 de abril de 2025, con una duración de 14 días.

- Burndown Report

En este *sprint* no se cumplió con los objetivos en el plazo determinado. La estimación de tiempo puede que fuera la correcta pero la falta de *hardware* retrasa las pruebas del software debido a no tener acceso a unas gafas de realidad virtual por la festividad de semana santa. Se focalizo más en la parte de documentación del proyecto.

Sprint 4: Palacio de Shahriar: Preparación e importación inicial

- Planning meeting:

Durante la reunión se marcaron los siguientes objetivos:

1. **Importación de modelos STL:** Importación de todos los modelos 3D del puzle, con su correcta escala y colisiones.
2. **Arreglar el menú de pausa:** Reestructuración del menú de pausa, debido a que era poco practica la implementación realizada.

3. **Implementar lógica de las torres:** Realizar una manera de hacer posible realizar giros de rosca en las torres con diferentes mecánicas como laberinto, giro escalonado o rotación acumulada.

4. **Añadir botón de saltar cinemática**

- Duración del *sprint*:

El *sprint* se realizó entre el 23 de abril de 2025 y el 7 de mayo de 2025, con una duración de 14 días.

- Burndown Report

En este *sprint* no se cumplió con los objetivos en el plazo determinado. La estimación de tiempo puede que fuera la correcta de nuevo, pero una vez más la falta del *hardware* retrasa las pruebas del software debido a no tener acceso a unas gafas de realidad virtual por compatibilidad de horarios con los prestantes de las gafas de RV. Debido a que comencé las practicas y el horario establecido para que pudiese utilizar las gafas fue muy reducido, tome la decisión de pedir las prestadas a un amigo para tener total disponibilidad de ellas y que el proyecto no se viese más atrasado por falta de material para las pruebas del código.

Sprint 5: Palacio de Shahriar: Interacciones avanzadas

- Planning meeting: Durante la reunión se marcaron los siguientes objetivos:

1. **Implementación del sistema de apertura de caja con *Palace_Box_Key*:** Diseñar e integrar la lógica necesaria para permitir la apertura de la caja mediante la llave correspondiente, incluyendo detección de colisiones y validación de acceso.
2. **Mecánica de caña de pescar con efecto magnético:** Desarrollar la funcionalidad de la caña con capacidad de atracción magnética, capaz de detectar objetos a distancias de 300, 60 y 5 unidades, con ajustes de precisión y respuesta visual.
3. **Animación y lógica de apertura de la caja:** Crear la animación correspondiente a la apertura de la caja y programar la lógica asociada para que se ejecute correctamente tras el uso de la llave o resolución del candado.
4. **Implementación del candado direccional:** Diseñar la secuencia de entrada direccional (ej. abajo, izquierda...) y su lógica de

validación para desbloquear el acceso a la caja, con retroalimentación visual y sonora.

5. **Obtención de la lámpara mágica:** Integrar el objeto "lámpara mágica" como recompensa al completar correctamente el puzle de la caja, incluyendo animación y lógica de recogida del objeto.

- Duración del sprint: El *sprint* se realizó entre el 7 de mayo de 2025 y el 21 de mayo de 2025, con una duración de 14 días.
- Burndown Report En este *sprint* se cumplió con casi todos los objetivos en el plazo determinado. La implementación de modelos como la caña de pescar, la caja y los candados no habían sido proporcionados, y el atraso de anteriores *sprints*, influenciaron en que este *sprint* se viese atrasado ligeramente.
- Sprint review meeting Lo más destacable de esta reunión fue profundizar en la facilidad del puzle, focalizando sobre todo en la parte de los *stickmans* para buscar una manera de que el usuario sepa identificar en que posición se deben de poner. Se optó por dejar la parte de la cabeza inmóvil.

***Sprint* 6: Palacio de Shahriar: Exportación de datos a CSV y pruebas**

- Planning meeting: Durante la reunión se marcaron los siguientes objetivos:
 1. **Exportación de métricas de rendimiento e interacción:** Definir e implementar la función que registre y exporte diversas métricas del puzle —tiempos de completado de cada paso, número de intentos por paso y cualquier otro dato relevante— generando un CSV con columnas para:
 - Identificación del paso
 - Timestamp de inicio y fin
 - Duración total
 - Número de intentos
 2. **Registro de movilidad del personaje:** Diseñar un sistema de teletransporte que cambie la posición del personaje a lo largo del puzle para mejorar la accesibilidad.

3. **Pruebas funcionales de exportación:** Elaborar y ejecutar casos de prueba que verifiquen la correcta generación del CSV en distintos escenarios (puzle completado, interrumpido, reiniciado), validando integridad y consistencia de datos.
 4. **Añadir elementos visuales:** Añadir imágenes y elementos que ayuden al usuario a realizar el puzle como controles del mando y pistas.
- Duración del sprint: El *sprint* se realizó entre el 21 de mayo de 2025 y el 11 de junio de 2025, con una duración de 21 días.
 - Burndown Report En este *sprint* se cumplió con todos los objetivos en el plazo determinado.
 - Sprint review meeting

A.3. Estudio de viabilidad

Este apartado tiene como objetivo analizar la viabilidad económica del proyecto, evaluando los costes asociados a su desarrollo y lanzamiento. Dado que el proyecto se encuentra en el ámbito académico, no tiene fines comerciales ni busca obtener beneficios económicos, pero se realiza una estimación teórica de los recursos empleados.

Viabilidad económica

Costes

Coste de personal Se estima que el desarrollo del proyecto ha requerido unas 300 horas de trabajo, distribuidas a lo largo de aproximadamente 4 meses. Esto equivale a unas 18,75 horas semanales.

Considerando que un ingeniero informático recién titulado recibe un salario bruto mensual de aproximadamente 1 900 € por una jornada completa de 40 horas semanales, el salario mensual equivalente para 18,75 horas semanales sería:

$$\frac{1900 \text{ €}}{40 \text{ h}} \times 18,75 \text{ h} = 890,63 \text{ €}$$

Este será el salario bruto del desarrollador. Para obtener el coste real para la empresa, se aplican los siguientes porcentajes según el *Régimen General de la Seguridad Social*¹:

- Contingencias comunes: $23,60\% = 0,236$
- Mecanismo de Equidad Intergeneracional (MEI): $0,67\% = 0,0067$
- Desempleo: $5,50\% = 0,055$
- FOGASA: $0,20\% = 0,002$
- Formación profesional: $0,60\% = 0,006$

Aplicando estos porcentajes:

$$890,63 \text{ €} \times (1 + 0,236 + 0,0067 + 0,055 + 0,002 + 0,006) = 1\,162,22 \text{ €}$$

Multiplicando por los 4 meses de duración estimada del proyecto:

$$1\,162,22 \text{ €} \times 4 = 4\,648,88 \text{ €}$$

Además, se incluyen sesiones de seguimiento con los tutores del TFG. Se han realizado 8 reuniones de aproximadamente 40 minutos con 3 tutores, lo que suma un total de 16 horas de dedicación conjunta. El salario por hora de un docente con experiencia se estima en 35 €/h.

$$16 \text{ h} \times 35 \text{ €/h} \times 3 \text{ tutores} = 1\,680 \text{ € (bruto mensual)}$$

Aplicando los mismos coeficientes para obtener el coste empresarial:

$$1\,680 \text{ €} \times (1 + 0,236 + 0,0067 + 0,055 + 0,002 + 0,006) = 2\,191,94 \text{ €}$$

¹<https://www.seg-social.es/wps/portal/wss/internet/CotizacionRecaudacionTrabajadores/Cotizacion/27243>

Coste de hardware El equipo de desarrollo utilizado ha sido un ordenador de uso personal, cuyo valor estimado es de 1 300 €. Se amortiza a 5 años (60 meses), por lo que el coste atribuido a los 4 meses de duración del proyecto es:

$$1\,300\text{ €} \times \frac{4}{60} = 86,67\text{ €}$$

El visor de realidad virtual *Meta Quest 2*, también de propiedad personal, tiene un valor de mercado estimado de 350 €, amortizable igualmente a 5 años:

$$350\text{ €} \times \frac{4}{60} = 23,33\text{ €}$$

Coste de software Todas las herramientas empleadas son gratuitas para uso académico:

- *Unreal Engine 5*
- *Visual Studio Community*
- *Blender*

La única excepción es la suscripción empresarial a *GitLab*, con un coste de 18 €/mes durante 4 meses:

$$18\text{ €} \times 4 = 72\text{ €}$$

Costes indirectos Se considera una tarifa de conexión a internet de 28 €/mes durante 4 meses:

$$28\text{ €} \times 4 = 112\text{ €}$$

Resumen de costes

Concepto	Duración / Cantidad	Coste (€)
Personal		
Desarrollador	4 meses	4 648,88
Tutores	16 horas	2 191,94
Hardware		
Ordenador de desarrollo (amortización)	4 meses	86,67
Meta Quest 2 (amortización)	4 meses	23,33
Software		
GitLab (suscripción empresarial)	4 meses	72,00
Costes indirectos		
Internet	4 meses	112,00
Total estimado		7 134,82

Tabla A.1: Resumen de costes estimados del proyecto

Beneficios

El desarrollo del proyecto no persigue ningún fin comercial ni beneficio económico directo. Se trata de un proyecto académico realizado en el marco del Trabajo de Fin de Grado, orientado a la mejora de la rehabilitación motriz de personas con discapacidad en las extremidades superiores.

El objetivo principal es contribuir con una solución accesible y funcional que pueda ser utilizada, evaluada o adaptada por entidades sin ánimo de lucro como la fundación ASPAYM. La motivación del proyecto es principalmente social, centrada en aportar valor a través de la tecnología, sin que ello suponga una explotación económica del producto desarrollado.

Por tanto, los beneficios derivados del proyecto son de carácter formativo, social y tecnológico, no existiendo intenciones de monetización ni comercialización del software.

Viabilidad legal

La viabilidad legal de un proyecto software consiste en evaluar todos los aspectos jurídicos vinculados con el desarrollo y la distribución del producto. En este caso, se consideran:

Licencias de *software*

Una licencia de *software* es un acuerdo legal entre el titular de los derechos de autor y el usuario final, en el que se establecen los términos bajo los cuales se permite el uso del producto [5].

El desarrollo del proyecto ha empleado herramientas y recursos cuya licencia es de carácter abierto o gratuito, y compatible con el uso académico. Las principales herramientas empleadas son:

- **Unreal Engine 5**: Licencia de uso gratuita para fines educativos, con posibilidad de publicación sin coste mientras no se superen los umbrales de ingresos establecidos por Epic Games [3].
- **Visual Studio Community**: Licencia gratuita para estudiantes, proyectos personales y académicos [4].
- **Blender**: Licenciado bajo GNU GPLv2+, permite libre uso, distribución y modificación [2].
- **GitLab**: Licenciado bajo MIT.

Dado que el desarrollo se ha realizado en un entorno académico, y no se contempla su distribución comercial, el cumplimiento legal respecto a las licencias se mantiene dentro de los márgenes establecidos por los proveedores de las herramientas utilizadas.

Licencias de imágenes y contenido gráfico

Para el diseño de interfaces o materiales visuales incluidos en el entorno de realidad virtual, se han empleado únicamente recursos libres de derechos o generados específicamente para este proyecto.

Todas las fuentes de iconos o modelos utilizados se han documentado debidamente en el repositorio del proyecto, siguiendo los términos de uso de cada proveedor.

Licencia del proyecto desarrollado

El software desarrollado será distribuido bajo una licencia libre. En este caso, se ha optado por aplicar la licencia **GNU GPLv3**, ya que garantiza que cualquier adaptación o distribución futura del software mantenga las mismas condiciones de libertad y apertura que el original. Esta decisión se alinea con el carácter académico, social y sin ánimo de lucro del proyecto.

Licencia de la documentación

La documentación generada para el proyecto será publicada bajo la licencia **Creative Commons CC BY-NC-ND 4.0**, lo que permite su libre distribución siempre que se mencione la autoría, no se utilice con fines comerciales y no se generen obras derivadas [1].

Cumplimiento de normativa de protección de datos

El proyecto no trata datos personales reales, ya que la aplicación está diseñada para no recoger ningún tipo de dato personal, tan solo datos sobre el progreso realizado en los puzles. Por tanto, no aplica el RGPD ni la LOPDGDD en este contexto.

Recurso	Licencia
Modelos ASPAYM	CC BY-NC-SA 4.0
Código propio (GitLab)	MIT License
Unreal Engine	EULA
Blender	GNU GPL v2 o superior

Tabla A.2: Licencias de dependencias y recursos del proyecto

Apéndice *B*

Especificación de Requisitos

B.1. Introducción

La especificación de requisitos establece el marco de trabajo para entender y documentar las necesidades del cliente y las características que debe cumplir el sistema. En esta fase, el equipo de desarrollo colabora con el *product owner* mediante entrevistas y revisiones iterativas, con el fin de recopilar tanto los requisitos funcionales (las acciones que el sistema debe realizar) como los no funcionales (aspectos de calidad como seguridad, rendimiento y escalabilidad).

Históricamente, las metodologías tradicionales proponen una captura temprana y fija de requisitos, empleando estándares como IEEE 830 para garantizar que sean claros, completos y verificables. Por el contrario, las metodologías ágiles optan por la flexibilidad, recogiendo las necesidades en forma de historias de usuario dentro de un *product backlog* vivo y priorizable.

Dado el carácter monodesarrollador de este TFG y la obligación de presentar una especificación detallada, se ha adoptado un enfoque híbrido que combina:

- Una especificación clásica de requisitos funcionales y no funcionales, estructurada según estándares.
- Algunas historias de usuario representativas, derivadas de las entrevistas con el *product owner*, para ilustrar la perspectiva ágil.

Una muestra de cómo podría ser una tabla de casos de uso:

B.2. Objetivos generales

Los objetivos definidos en este proyecto son los siguientes:

1. **Desarrollar un entorno virtual para la rehabilitación motriz mediante Unreal Engine:** que permita a los usuarios realizar ejercicios de rehabilitación de las extremidades superiores, a través de puzzles adaptados a la tecnología VR.
2. **Monitorear el progreso de los usuarios durante las sesiones de rehabilitación:** Implementar un sistema que registre datos sobre el rendimiento de cada usuario, como el tiempo invertido, el número de movimientos realizados, la duración de las sesiones, y otros parámetros.

B.3. Catálogo de requisitos

El presente Catálogo de Requisitos Funcionales recoge de manera estructurada todas las funcionalidades que el software debe ofrecer, conforme a las necesidades expuestas por el *Product Owner* durante las sucesivas reuniones de definición del proyecto. Cada requisito ha sido validado y consensuado con el cliente para asegurar que el sistema cumple con sus expectativas y objetivos. A continuación se detallan las dos categorías de requisitos:

Requisitos funcionales

Este tipo de requerimiento (de ahora en adelante RF) detalla las funciones que debe realizar el software para cumplir con las expectativas del usuario. Es decir, se centra en qué debe hacer el sistema y cómo. En esta categoría se encuentran:

RF-1 Gestión de partidas: el sistema debe permitir al usuario gestionar sus partidas guardadas.

- **RF-1.1 Listar partidas:** el sistema mostrará todas las partidas asociadas al usuario.
- **RF-1.2 Cargar partida:** el sistema cargará la partida seleccionada por el usuario.
- **RF-1.3 Borrar partida:** el sistema eliminará de forma permanente la partida seleccionada por el usuario.

RF-2 Pantalla de créditos y agradecimientos: el sistema debe mostrar una pantalla con los créditos de desarrollo y mensajes de agradecimiento cuando el usuario seleccione dicha opción desde el menú principal.

RF-3 Selección de nivel (puzle): el sistema debe permitir al usuario seleccionar y preparar un nuevo puzle para jugar.

- **RF-3.1 Mostrar lista de puzles:** el sistema listará todos los puzles disponibles.
- **RF-3.2 Mostrar información del puzle:** al seleccionar un puzle, el sistema mostrará detalles y descripción del mismo.
- **RF-3.3 Iniciar puzle:** el sistema cargará el puzle seleccionado y preparará el entorno de juego.
- **RF-3.4 Mostrar resultados históricos:** el sistema mostrará estadísticas de sesiones anteriores asociadas al puzle seleccionado.
- **RF-3.5 Exportar resultados a CSV:** el sistema permitirá al usuario exportar en formato CSV los datos históricos mostrados.

RF-4 Sesión de juego: el sistema debe gestionar la ejecución de la sesión de juego completa.

- **RF-4.1 Introducción narrativa:** al iniciar la sesión, el sistema mostrará una breve narrativa introductoria.
- **RF-4.2 Ejecución del puzle:** el sistema permitirá al usuario realizar la tarea o resolver el puzle.
- **RF-4.3 Mostrar estadísticas de la sesión:** al finalizar, el sistema calculará y mostrará la duración de la sesión y otras estadísticas básicas.

RF-5 Consulta de datos: el terapeuta será capaz de consultar los datos contenidos en el CSV previamente exportado.

Requisitos no funcionales

Además de los requisitos funcionales, es fundamental definir las condiciones de calidad, rendimiento y restricciones bajo las cuales el sistema debe operar. A continuación se incluyen los Requisitos No Funcionales identificados:

- **RNF-1 Rendimiento:** Mantener al menos 72 FPS estables en Meta Quest 2 durante toda la sesión de juego.
- **RNF-2 Usabilidad:** La interfaz debe permitir completar las acciones principales (listar, cargar, borrar partidas; seleccionar e iniciar puzles; ver/exportar resultados) en un máximo de 2 gestos de controlador dentro de un alcance de brazo.
- **RNF-3 Portabilidad:** El software debe ser compatible con Meta Quest.
- **RNF-4 Fiabilidad:** La tasa de errores críticos (*crash* o pérdida de datos) debe ser casi nula.
- **RNF-5 Accesibilidad VR:**
 - Soporte exclusivo de interacción sentado, sin requerir desplazamiento físico del usuario.
 - Todas las acciones deben realizarse mediante movimientos de extremidades superiores (controladores de mano), sin exigir movimientos de torso o piernas.
 - Elementos de interfaz e interactivos a una distancia máxima de 0,6 m (alcance del brazo) y a una altura entre 0,8 m y 1,5 m del suelo aproximadamente.

B.4. Especificación de requisitos

A continuación se presenta la especificación detallada de requisitos de la aplicación *VR* desarrollada en Unreal Engine 5.5 para Meta Quest 2, basada en el Catálogo de Requisitos. En esta sección se describen en profundidad tanto los requisitos funcionales como los no funcionales, incluyendo sus precondiciones, criterios de aceptación y prioridades, con el fin de asegurar que el sistema satisfaga las necesidades de usuarios con movilidad reducida y de los terapeutas encargados de su seguimiento.

Diagrama de casos de uso

Tomando como base las historias de usuario definidas junto al *Product Owner*, se ha construido el diagrama general de casos de uso de la aplicación. A continuación se presenta dicho diagrama, seguido de las tablas detalladas de cada caso de uso.

Figura B.1: Diagrama general de casos de uso de la aplicación.



CU-1	Listar partidas
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-1 (RF-1.1)
Descripción	Permite al usuario visualizar la lista de partidas existentes asociadas a su cuenta.
Precondición	El usuario debe haber iniciado sesión y tener al menos una partida guardada.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona «Listar partidas».2. El sistema recupera y muestra todas las partidas disponibles para el usuario.
Postcondición	Se presenta al usuario la lista actualizada de partidas.
Excepciones	Si no existen partidas, el sistema muestra un mensaje “No hay partidas disponibles”.
Importancia	Alta

Tabla B.1: CU-1 Listar partidas.

CU-2	Cargar partida guardada
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-1 (RF-1.2)
Descripción	Permite al usuario cargar una de sus partidas previamente guardadas.
Precondición	El usuario debe haber iniciado sesión y existir la partida seleccionada.
Acciones	<ol style="list-style-type: none"> 1. El usuario elige «Lista de partidas». 2. El sistema muestra la lista de partidas disponibles. 3. El usuario selecciona la partida deseada. 4. El sistema carga los datos de la partida seleccionada.
Postcondición	La partida guardada se carga y el usuario puede continuar jugando.
Excepciones	Si la partida está corrupta o no existe, se muestra un mensaje de error y se mantiene la sesión anterior.
Importancia	Alta

Tabla B.2: CU-2 Cargar partida guardada.

CU-3	Borrar partida
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-1 (RF-1.3)
Descripción	Permite al usuario eliminar una partida de su cuenta de forma permanente.
Precondición	El usuario debe haber iniciado sesión y existir la partida seleccionada.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona «Borrar partida».2. El sistema elimina la partida y actualiza la lista.
Postcondición	La partida seleccionada queda eliminada de la cuenta del usuario.
Excepciones	Si la eliminación falla, se informa al usuario y no se modifica la lista.
Importancia	Alta

Tabla B.3: CU-3 Borrar partida.

CU-4	Mostrar pantalla de créditos y agradecimientos
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-2
Descripción	Muestra una pantalla con los créditos del desarrollo y mensajes de agradecimiento.
Precondición	El usuario accede a la sección de créditos desde el menú principal.
Acciones	<ol style="list-style-type: none">1. El usuario selecciona «Créditos» en el menú.2. El sistema presenta la pantalla con la información de los desarrolladores y colaboradores.
Postcondición	El usuario visualiza la pantalla de créditos.
Excepciones	Ninguna esperada; si falla la carga, se muestra un mensaje genérico.
Importancia	Media

Tabla B.4: CU-4 Mostrar pantalla de créditos y agradecimientos.

CU-5	Elegir nivel
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-3 (RF-3.1,RF-3.2,RF-3.3)
Descripción	Permite al usuario seleccionar un puzle para iniciar una nueva partida.
Precondición	El usuario debe haber iniciado sesión.
Acciones	<ol style="list-style-type: none"> 1. El usuario elige «Nueva partida». 2. El sistema muestra la lista de puzles disponibles. 3. El usuario selecciona el puzle deseado. 4. El sistema muestra información detallada del puzle. 5. El usuario pulsa «Botón con el nombre de puzle». 6. El sistema carga el puzle seleccionado.
Postcondición	El puzle está listo para jugarse.
Excepciones	Fallo de carga de nivel, se informa al usuario con un mensaje.
Importancia	Alta

Tabla B.5: CU-5 Elegir juego (nivel).

CU-6	Mostrar resultados históricos de sesión
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-3 (RF-3.4)
Descripción	Extiende CU-5 para mostrar estadísticas de sesiones anteriores del puzle seleccionado.
Precondición	Se ha seleccionado un puzle (CU-5).
Acciones	<ol style="list-style-type: none">1. El usuario elige «Ver resultados históricos».2. El sistema recupera los datos de sesiones previas de ese puzle.3. Se muestran las estadísticas (duración, puntuación) en pantalla.
Postcondición	Se visualizan los datos históricos.
Excepciones	Si no existen datos, se muestra “No hay resultados anteriores”.
Importancia	Media

Tabla B.6: CU-6 Mostrar resultados históricos de sesión asociados al puzle seleccionado.

CU-7	Exportar a CSV
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-3 (RF-3.5)
Descripción	Extiende CU-6 para exportar los resultados históricos a un archivo CSV.
Precondición	Se están mostrando resultados históricos (CU-6).
Acciones	<ol style="list-style-type: none">1. El usuario selecciona «Exportar a CSV».2. El sistema genera un fichero CSV con los datos mostrados.3. Se inicia la descarga del CSV al dispositivo del usuario.
Postcondición	El fichero CSV con resultados históricos está descargado.
Excepciones	Si hay un error en la generación, se informa al usuario.
Importancia	Media

Tabla B.7: CU-7 Exportar a CSV.

CU-8	Realizar sesión de juego
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-4 (RF-4.1,RF-4.2,RF-4.3)
Descripción	Actividad principal del videojuego: incluye una introducción narrativa y la ejecución del puzle, finalizando con la presentación resumida de estadísticas de la sesión.
Precondición	El puzle debe estar cargado y listo para jugarse (CU-5).
Acciones	<ol style="list-style-type: none">1. El sistema muestra la narrativa introductoria.2. El usuario realiza la tarea o resuelve el puzle.3. Al finalizar, el sistema calcula la duración de la sesión.4. Se muestran las estadísticas básicas de la sesión.
Postcondición	La sesión queda registrada con su duración y resultados.
Excepciones	Si hay fallo al guardar estadísticas, se notifica al usuario y se vuelve a intentar.
Importancia	Alta

Tabla B.8: CU-8 Realizar sesión de juego.

CU-9	Realizar sesión de juego: Palacio de Shahriar
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-4 (RF-4.1,RF-4.2,RF-4.3))
Descripción	Especialización de CU-8 para el escenario del Palacio de Shahriar, con narrativa y desafíos temáticos.
Precondición	El usuario ha seleccionado el puzle “Palacio de Shahriar” (CU-5).
Acciones	<ol style="list-style-type: none"> 1. El sistema presenta la historia del Palacio de Shahriar. 2. El usuario resuelve el enigma específico de este escenario. 3. Se calculan y muestran las estadísticas de la sesión.
Postcondición	La sesión temática queda registrada con sus estadísticas.
Excepciones	Si falla la carga del escenario, se informa al usuario y se vuelve al menú de selección.
Importancia	Alta

Tabla B.9: CU-9 Realizar sesión de juego: Palacio de Shahriar

CU-10	Consulta de datos
Versión	1.0
Autor	Rodrigo Grande
Requisitos asociados	RF-5
Descripción	El terapeuta sera capaz de consultar los datos exportados por el usuario.
Precondición	El usuario ha exportado los datos a CSV y este es facilitado al terapeuta.
Acciones	<ol style="list-style-type: none">1. El terapeuta consulta el CSV.
Postcondición	El usuario recibe un <i>feedback</i> por parte del terapeuta.
Excepciones	El CSV no ha sido exportado correctamente.
Importancia	Alta

Tabla B.10: CU-10 Consulta de datos.

Apéndice C

Especificación de diseño

C.1. Introducción

La especificación de diseño es una descripción detallada de la estructura, el comportamiento y la interacción de los distintos componentes que conforman el producto *software*. Su propósito es proporcionar una guía clara y precisa que permita comprender cómo se organiza internamente la aplicación, facilitando su desarrollo, mantenimiento y futuras ampliaciones.

A continuación, se presentan los apartados que conforman la especificación de diseño de la aplicación desarrollada en el presente proyecto.

C.2. Diseño de datos

El sistema de guardado del proyecto se basa en el uso de dos *blueprints* derivados de la clase *SaveGame* de *Unreal Engine*: **BP_PlayerSave** y **BP_PalaceSave**. Estos permiten almacenar y recuperar el progreso tanto del jugador como del estado interno de cada entorno de puzzles.

BP_PlayerSave

Este fichero guarda la información general del jugador y un resumen del estado de cada puzzle registrado. Esta organización permite una gestión escalable y ordenada de múltiples partidas o sesiones por jugador, así como una rápida vinculación con el estado persistente de cada entorno.

BP_PalaceSave

Este fichero almacena el estado detallado de un puzle concreto, permitiendo restaurarlo exactamente al punto donde se dejó. Cada fichero de este tipo se corresponde con un índice único que lo vincula a una entrada de BP_PlayerSave.

Esta estructura permite una recuperación precisa del estado del puzle, incluyendo progresos parciales, posiciones intermedias de actores, y lógicas de desbloqueo complejas.

Relación entre archivos de guardado

Cada elemento del array `PuzzleInfoArray` en BP_PlayerSave se vincula con una única instancia de BP_PalaceSave mediante el campo `Index`. De este modo, se mantiene una separación lógica entre los datos personales del jugador y el estado particular de cada entorno de juego, favoreciendo la modularidad y la mantenibilidad del sistema de persistencia.

Diccionario de datos

A continuación, se presentan los diccionarios de datos correspondientes a las estructuras BP_PlayerSave y BP_PalaceSave. Estas tablas describen los atributos almacenados en los archivos de guardado utilizados por la aplicación, especificando su tipo, posibles restricciones y una breve descripción de su función dentro del sistema.

El objetivo de este esquema es facilitar la comprensión de la persistencia de datos implementada mediante *blueprints* en *Unreal Engine*, así como establecer una documentación clara de los elementos clave que intervienen en el proceso de guardado y carga de estado.

Nombre	Tipo	Restricción	Descripción
PlayerName	FString	NOT NULL	Nombre introducido por el usuario en el registro.
PuzzleInfoArray	TArray<FPuzzleInfo>	NOT NULL	Array de estructuras que almacena la información resumida de cada puzle del jugador.
Estructura FPuzzleInfo (elementos del array PuzzleInfoArray):			
PuzzleName	FString	NOT NULL	Nombre del puzle.
CreationDateTime	DateTime	NOT NULL	Fecha y hora de creación del puzle.
TimePlayed	float		Tiempo jugado acumulado.
Solved	bool		Indica si el puzle ha sido completado.
Index	int32	UNIQUE	Índice que enlaza con el correspondiente BP_PalaceSave.

Tabla C.1: Diccionario de datos correspondiente a la clase BP_PlayerSave.

Nombre	Tipo	Restricción	Descripción
Index	int32	UNIQUE NOT NULL	Índice de enlace con BP_PlayerSave.
StepsMade	int32		Número de pasos realizados por el jugador en las torres.
Tower3Finished	bool		Indica si la torre 3 está completada.
ActorsDestroyed	TArray<FName>		Lista de nombres de actores eliminados.
ActualTime	float		Tiempo acumulado que lleva el puzle activo.
ActorsHidden	TArray<FName>		Lista de actores ocultos en el entorno.
Lock1Unlocked	bool		Estado del candado 1 (desbloqueado o no).
Lock2Unlocked	bool		Estado del candado 2.
OnlyRotationsActor	TMap<FName, FRotator>		Rotación de actores con etiqueta SaveRotation.
StepsData	TArray<FStepData>		Información por torre: nombre y tiempo empleado.
Estructura FStepData (elementos del array StepsData):			
Name	FString	NOT NULL	Nombre de la torre.
TimeToComplete	float		Tiempo requerido para completar la torre.

Tabla C.2: Diccionario de datos correspondiente a la clase BP_PalaceSave.

C.3. Diseño arquitectónico

El diseño arquitectónico del sistema establece la estructura general del proyecto, identificando los principales módulos funcionales, sus relaciones y la forma en la que interactúan para lograr los objetivos definidos. En este proyecto, la arquitectura se ha diseñado con un enfoque modular y escalable, facilitando la integración de nuevas funcionalidades y el mantenimiento del código.

El sistema se ha desarrollado utilizando *Unreal Engine 5*, por lo que la arquitectura se organiza en torno a clases y *blueprints*, respetando el paradigma de programación orientada a objetos. Las clases clave se dividen en tres niveles: lógica de usuario, lógica de entorno y control de interacción.

Clases principales

En la figura REFERENCIA A FIGURA se muestra el diagrama general que representa las clases y *blueprints* principales de la lógica del proyecto:

- **VRGameInstance**: clase persistente que gestiona el guardado y carga de partidas, así como la transición entre niveles. Controla los **SaveGames**.
- **BP_PlayerSave** y **BP_PalaceSave**: clases derivadas de **SaveGame** que contienen los datos persistentes del jugador y el entorno, respectivamente (descritas en el diseño de datos).

Despliegue

La aplicación está diseñada para ejecutarse como una aplicación de escritorio en sistemas operativos Windows. El entorno de realidad virtual se experimenta a través del visor *Meta Quest 2*, utilizando la funcionalidad *Meta Quest Link*, que permite al dispositivo actuar como pantalla y controlador de entrada para el equipo principal.

El proceso de despliegue y ejecución sigue los siguientes pasos:

1. El proyecto se compila en **Unreal Engine 5** en formato ejecutable **.exe** para Windows, utilizando el modo **VR Preview** para pruebas o el empaquetado estándar para distribución.

2. El visor *Meta Quest 2* se conecta al ordenador mediante cable USB o por conexión inalámbrica compatible (*Air Link*), utilizando la funcionalidad *Meta Quest Link* para emular una pantalla secundaria en realidad virtual.
3. Al ejecutar la aplicación en el equipo, el usuario puede visualizar e interactuar con el entorno virtual desde el visor, con seguimiento completo de movimiento y controladores hápticos.

Este enfoque de despliegue permite aprovechar toda la potencia gráfica del equipo de desarrollo, eliminando las limitaciones de hardware del visor autónomo. Además, facilita el desarrollo iterativo, la depuración en tiempo real y el uso de recursos más exigentes sin afectar al rendimiento.

C.4. Diseño procedimental

En el desarrollo de *software*, el diseño procedimental hace referencia a la definición estructurada y detallada de los pasos necesarios para llevar a cabo funcionalidades concretas dentro de la aplicación. Este enfoque se basa en la descomposición del problema en tareas elementales y secuenciales que, en conjunto, permiten alcanzar un objetivo funcional específico.

En el presente proyecto, el diseño procedimental se ha representado mediante diagramas de secuencia siguiendo la notación de *UML 2*. Estos diagramas permiten visualizar la interacción entre los distintos componentes del sistema —clases, *blueprints*, interfaces de usuario y controladores de entorno— durante la ejecución de los distintos procesos clave de la aplicación.

A continuación, se muestran los diagramas correspondientes a algunas de las funcionalidades principales del sistema, como el proceso de guardado de estado, la carga de partidas, o la gestión de eventos dentro de los puzles, con el objetivo de facilitar la comprensión de su comportamiento dinámico.

Registro

En este primer diagrama de secuencia se muestra los pasos que sigue el programa al registrarse por primera vez:

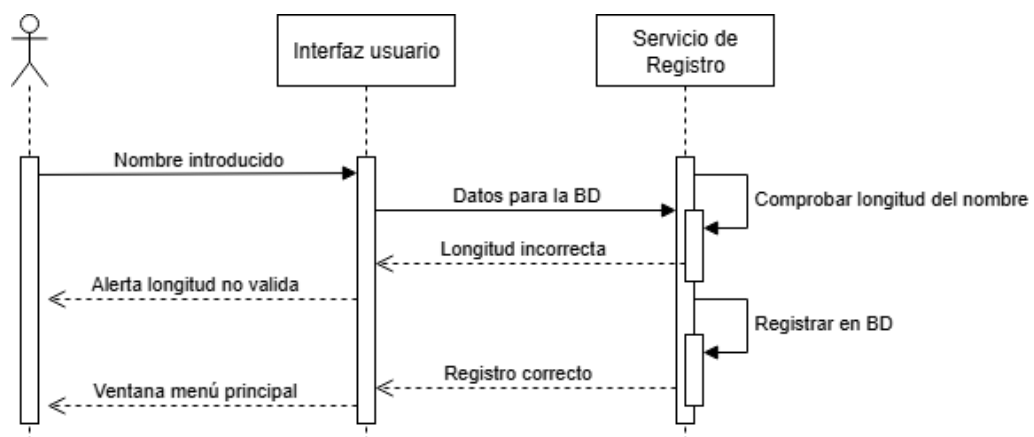


Figura C.1: Diagrama de secuencia del registro por primera vez

Apéndice D

Documentación técnica de programación

- D.1. Introducción
- D.2. Estructura de directorios
- D.3. Manual del programador
- D.4. Compilación, instalación y ejecución del proyecto
- D.5. Pruebas del sistema

Apéndice E

Documentación de usuario

- E.1. Introducción
- E.2. Requisitos de usuarios
- E.3. Instalación
- E.4. Manual del usuario

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

Este anexo incluirá una reflexión personal del alumnado sobre los aspectos de la sostenibilidad que se abordan en el trabajo. Se pueden incluir tantas subsecciones como sean necesarias con la intención de explicar las competencias de sostenibilidad adquiridas durante el alumnado y aplicadas al Trabajo de Fin de Grado.

Más información en el documento de la CRUE https://www.crue.org/wp-content/uploads/2020/02/Directrices_Sostenibilidad_Crue2012.pdf.

Este anexo tendrá una extensión comprendida entre 600 y 800 palabras.

Bibliografía

- [1] Creative Commons. About the licenses - creative commons, 2024. [Consulta: mayo 2025].
- [2] Blender Foundation. Blender license, 2024. [Consulta: mayo 2025].
- [3] Epic Games. Licencia de unreal engine, 2024. [Consulta: mayo 2025].
- [4] Microsoft. Visual studio licensing terms, 2024. [Consulta: mayo 2025].
- [5] Red.es. ¿qué es una licencia de software?, 2020. [Consulta: mayo 2025].