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 — 3 de julio de 2025
Tutor: Álvar 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 8

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 37

Apéndice D Documentación técnica de programación 41
D.1. Introducción 41
D.2. Estructura de directorios 41
D.3. Manual del programador 43

D.4. Compilación y ejecución del proyecto 46
D.5. Pruebas del sistema 47

Apéndice E Documentación de usuario **51**
E.1. Introducción 51
E.2. Requisitos de usuarios 51
E.3. Instalación 53
E.4. Manual del usuario 53

Apéndice F Anexo de sostenibilización curricular **55**
F.1. Introducción 55
F.2. ODS 3: Salud y Bienestar 55
F.3. ODS 9: Industria, Innovación e Infraestructura 56
F.4. ODS 10: Reducción de las desigualdades 56
F.5. ODS 13: Acción por el clima 57
F.6. Conclusión 57

Bibliografía **59**

Índice de figuras

B.1. Diagrama: casos de uso 19

C.1. Ejemplo de singleton 35
C.2. Ejemplo de command 36
C.3. Diagrama de secuencia de registro 38
C.4. Diagrama de secuencia de nueva partida 39
C.5. Diagrama de secuencia cargado de partida 39

D.1. Estructura del proyecto 42
D.2. Ejemplo de telemetria 48

E.1. Pantalla de registro 54

Índice de tablas

A.1. Resumen de costes estimados del proyecto 10
A.2. Licencias de dependencias y recursos del proyecto 14

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 puzle seleccionado. 25
B.7. CU-7 Exportar a CSV. 26
B.8. CU-8 Realizar sesión de juego. 27
B.9. CU-9 Jugar partida: 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

E.1. Requisitos mínimos PC para Meta Quest Link 52

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 esta siguiendo la metodologia Scrum, sera 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 puzles:** Leer y analizar toda la documentación relacionada con los puzles 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 puzles.
  3. **Selección de gafas de realidad virtual:** Acordar la versión de las gafas de *VR* 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 es 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 marco 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 utilizo 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 VR, 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 del*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 VR. Debido a que comencé las practicas y el horario establecido para que pudiese utilizar las gafas fue muy reducido, tome la decisión de pedirlas 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. **Obtención de la lámpara mágica:** Integrar el objeto 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 habian 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 opto 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 En un principio se tenía estimado realizar otro puzle más, pero después de hablarlo y valorarlo se decidió no seguir ese plan y dedicar las últimas semanas al pulido de detalles y terminar la documentación.

Sprint 7: Documentación final y fase de pulido

- *Planning meeting*: Durante la reunión de planificación del último *sprint*, centrado en la documentación y el pulido final del proyecto, se establecieron los siguientes objetivos clave:

  1. **Mejora de la interfaz de usuario:** Se acordó realizar ajustes visuales finales que aumenten la claridad, accesibilidad y atractivo de la experiencia, como la reorganización de elementos gráficos, tamaños de fuente o disposición de los menús en *VR*.
  2. **Determinación definitiva de datos exportables:** Se propuso cerrar la definición de las métricas que serán registradas en el archivo *CSV*, incluyendo aquellos datos que hasta ahora estaban pendientes.
  3. **Empaquetado del proyecto:** Garantizar que la versión final del sistema esté correctamente empaquetada como ejecutable (`.exe`) para *Windows*, con compatibilidad completa con *Meta Quest Link* y sin dependencias externas no incluidas.
  4. **Realización de pruebas funcionales y corrección de errores:** Llevar a cabo una batería final de pruebas sobre las funcionalidades principales —interacción con el entorno, guardado de progreso, exportación de métricas, desplazamiento del jugador— e identificar y corregir errores remanentes.
  5. **Finalización de la documentación:** Redactar y estructurar adecuadamente todos los apartados pendientes de la memoria del *TFG*

- Duración del sprint: El *sprint* se realizó entre el 11 de junio de 2025 y el 7 de julio de 2025, con una duración de 26 días.

- Burndown Report En este *sprint* se cumplió con todos los objetivos en el plazo determinado. Fue una fase larga donde se realizaron múltiple test de la aplicación, se termino de realizar la toma y exportado de datos a CSV y la correcta implementación del guardado y cargado, donde había varios *bugs*.

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:

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

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

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

- FOGASA: 0,20 % = 0,002
- Formación profesional: 0,60 % = 0,006

Aplicando estos porcentajes:

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

Multiplicando por los 4 meses de duración estimada del proyecto:

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

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:

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

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:

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

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:

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

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:

$$18\ € \times 4 = 72\ €$$

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

$$28\ € \times 4 = 112\ €$$

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 [6].

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 [5].
- **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.

Licencia de distribución de software

Tras el análisis de compatibilidad de licencias con las herramientas utilizadas en el desarrollo, se ha optado por distribuir el código fuente del proyecto bajo la licencia **MIT License**. Esta elección permite una

reutilización amplia y flexible del código, incluso en proyectos comerciales o cerrados, siempre que se mantenga la atribución al autor original.

La *MIT License* resulta especialmente adecuada para proyectos académicos y de investigación, ya que fomenta la difusión abierta del conocimiento sin imponer restricciones adicionales. Asimismo, es compatible con el uso del motor *Unreal Engine*, cuyo entorno de desarrollo no impide la aplicación de licencias permisivas en el código propio del desarrollador.

Esta decisión está alineada con el objetivo de facilitar la adopción, mejora y extensión del sistema por parte de terceros, incluyendo comunidades educativas, clínicas y entidades colaboradoras como ASPAYM. Se incluye el texto completo de la licencia en el repositorio del proyecto, junto con la atribución correspondiente en los archivos fuente principales.

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 ente 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 puzles 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.

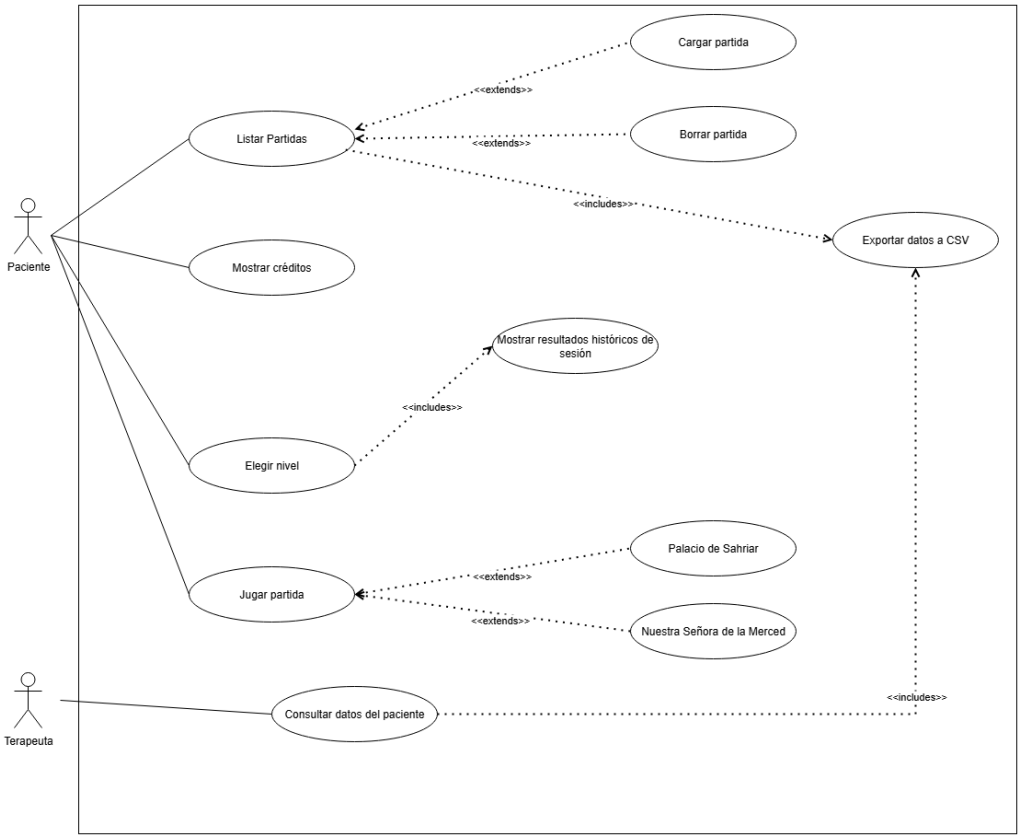
B.4. Especificación de requisitos

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	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	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	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	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	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	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	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	**Jugar partida**
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	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	**Jugar partida: 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	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 Jugar partida: 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	1. El terapeuta consulta el CSV.
Postcondición	El usuario recibe un *feedback* 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.

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 puzles.

BP_PlayerSave

Este fichero guarda la información general del jugador y un resumen del estado de cada puzle 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	FDateTime	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.

Patrones de diseño

Patrón *Singleton*

Se implementa el patrón de diseño *Singleton* mediante la clase `UInstance`, la cual hereda de `UGameInstance` en *Unreal Engine*. Esta clase actúa como una instancia global accesible desde cualquier nivel o clase dentro del motor.

El patrón *Singleton* garantiza que sólo exista una única instancia de esta clase durante toda la ejecución del juego, y que todas las clases del sistema puedan acceder a ella de manera controlada. Esta instancia persiste entre cambios de nivel, lo que permite compartir datos sin pérdida de información.

Responsabilidades de la clase `UInstance` en el proyecto:

- Gestionar el acceso centralizado al *SaveGame* cargado, conteniendo los datos del usuario.
- Almacenar y compartir variables globales (por ejemplo, puntuaciones, tiempo total de actividad, nivel de dificultad, estadísticas de movimiento, etc.).
- Servir de punto de comunicación entre niveles y sistemas desacoplados dentro del entorno VR.

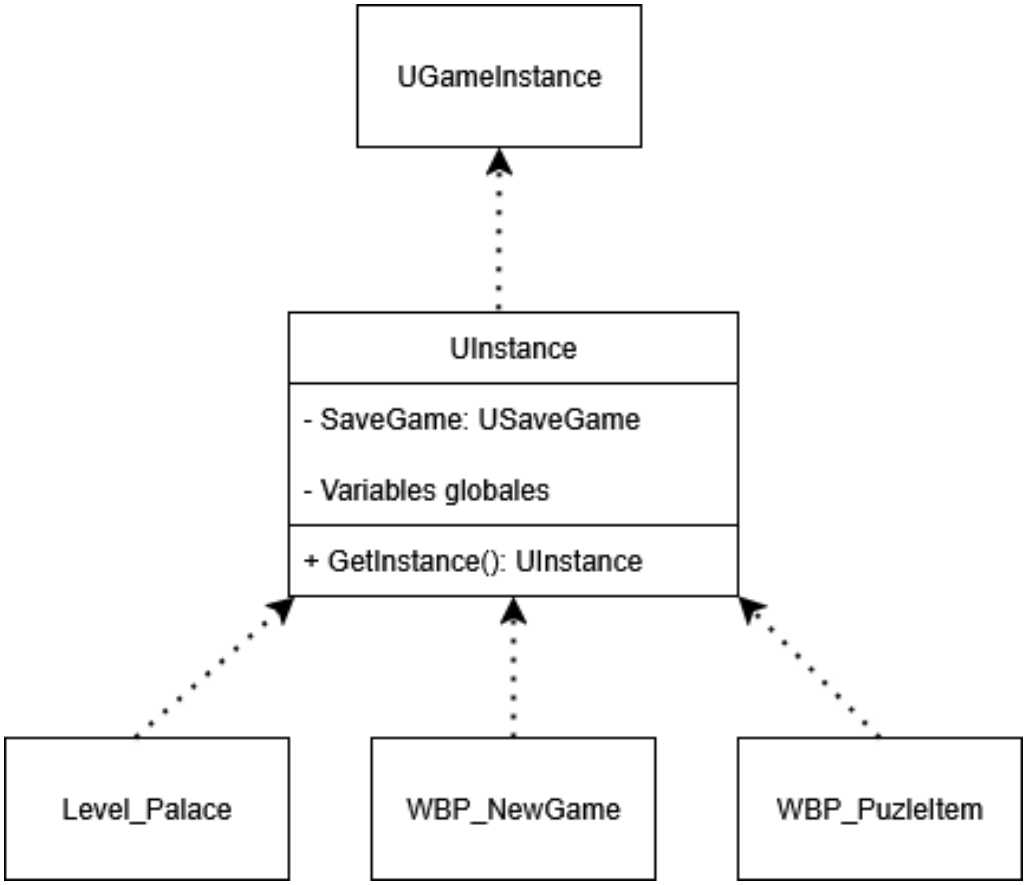
Ventajas del uso del *Singleton* en este contexto:

- Evita la necesidad de pasar referencias entre niveles o clases.
- Mejora la organización y consistencia del estado global del juego.

- Asegura la integridad de los datos compartidos entre escenas.

Este uso del patrón *Singleton* es coherente con las mejores prácticas de diseño en *Unreal Engine*, y resulta especialmente útil en aplicaciones donde la continuidad del estado entre escenarios es esencial para la experiencia del usuario.

Figura C.1: Ejemplo del patrón singleton en el proyecto.



Patrón *Command*

El patrón de diseño *Command* se aplica para gestionar acciones que deben ejecutarse como respuesta a condiciones específicas del entorno de juego, manteniendo bajo acoplamiento entre los objetos emisores y receptores.

Este patrón encapsula una petición (acción) como un objeto, permitiendo su almacenamiento, programación o ejecución diferida. En Unreal Engine,

esta lógica se implementa comúnmente mediante *event dispatchers* o *delegates*, herramientas que permiten vincular eventos con funciones de manera flexible.

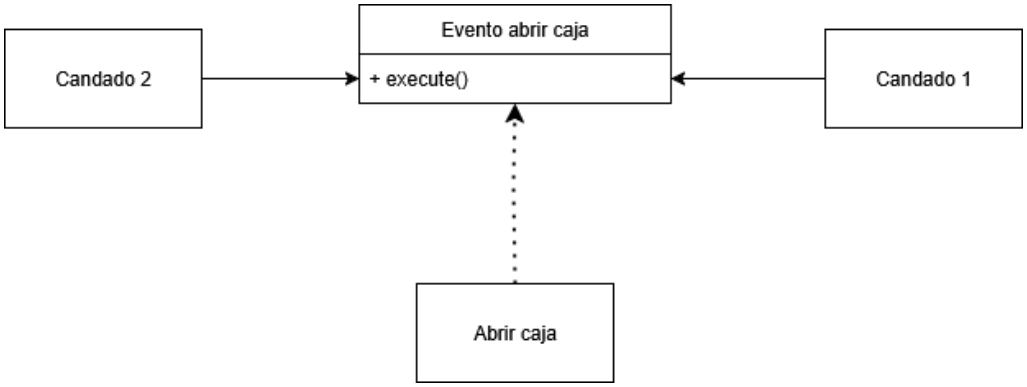
Ejemplos concretos en el proyecto:

- **Evento de apertura de caja:** Cuando todos los candados están resueltos, se emite un evento que ejecuta la animación de apertura de la caja. El objeto çandado"no necesita conocer directamente la lógica interna de la caja, sólo invocar el comando asociado.
- **Evento de finalización del juego:** Al cumplirse los criterios de éxito de la tarea motriz, se lanza un comando que activa la animación final, la recolección de métricas y la transición de nivel. Todo ello está encapsulado en un conjunto de acciones invocadas como comandos.

Ventajas en el contexto del proyecto:

- Desacopla la lógica de condición (candado, éxito, tiempo, etc.) de la lógica de ejecución.
- Permite reutilizar, programar o modificar comandos sin afectar el código fuente de los objetos emisores.
- Facilita la modularidad y mantenimiento del sistema VR, donde múltiples interacciones generan eventos secuenciados.

Figura C.2: Ejemplo del patrón command en el proyecto.



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 muestran los pasos que sigue el programa al registrarse por primera vez:

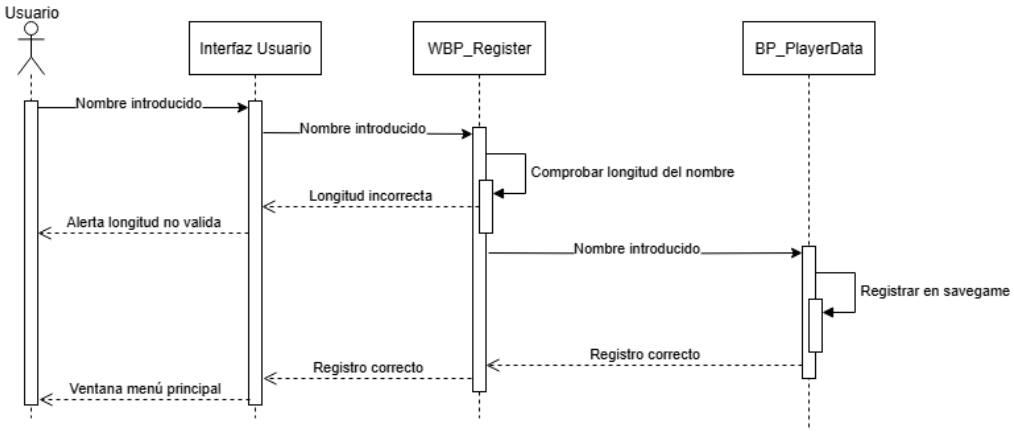


Figura C.3: Diagrama de secuencia del registro por primera vez

Crear nueva partida

En este segundo diagrama de secuencia se muestran los pasos que sigue el programa al crear un nuevo puzle.

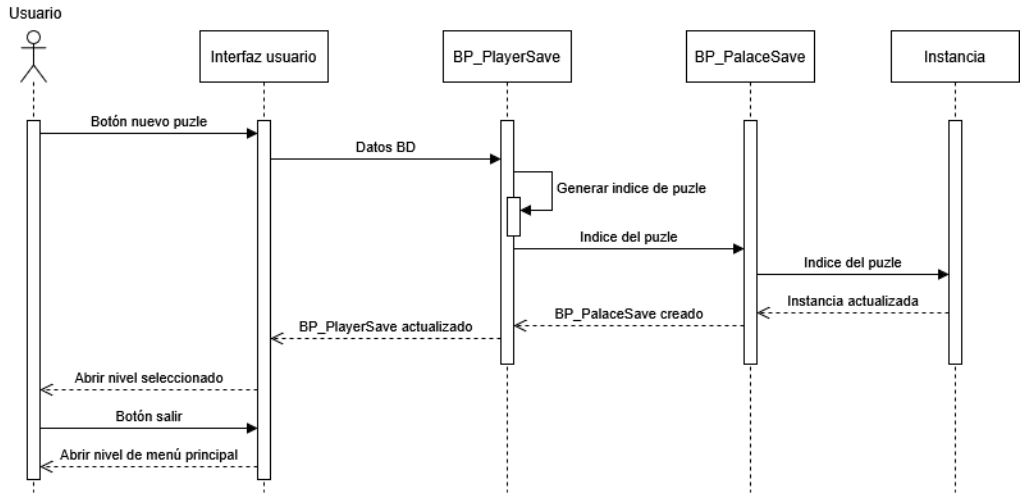


Figura C.4: Diagrama de secuencia de nueva partida

Cargado de partida

En este tercer diagrama de secuencia se muestran los pasos que sigue el programa para cargar un puzle.

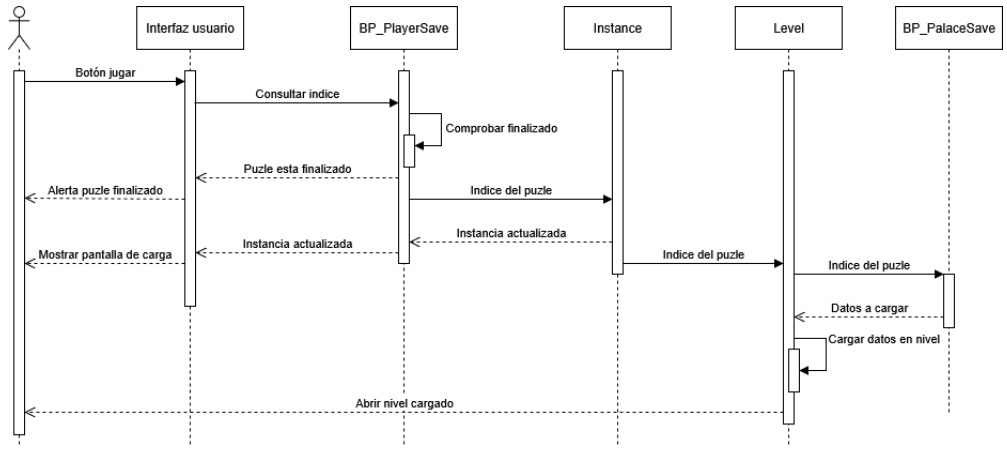


Figura C.5: Diagrama de secuencia de cargado de partida

Apéndice D

Documentación técnica de programación

D.1. Introducción

Este documento técnico proporciona toda la información necesaria para que un desarrollador pueda instalar, ejecutar y mantener el programa.

Está dirigido a programadores, colaboradores o personal técnico que desee contribuir al proyecto, comprender su estructura interna o realizar mejoras y extensiones futuras.

D.2. Estructura de directorios

En esta sección se describe la estructura jerárquica de carpetas empleada en el desarrollo del entorno, implementado en *Unreal Engine*. La clasificación de los recursos se ha realizado de forma modular, separando claramente los elementos visuales, la lógica del juego, las interfaces de usuario y los activos reutilizables, con el objetivo de facilitar tanto el trabajo colaborativo como la futura evolución del sistema.

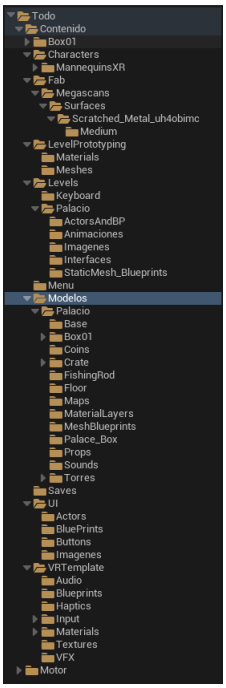


Figura D.1: Estructura del proyecto

Descripción de la estructura

`Todo/` – Carpeta raíz del proyecto, contiene todos los activos y recursos del entorno de desarrollo en *Unreal Engine*.

- `Contenido/` – Carpeta principal de contenido dentro del editor de Unreal Engine.
  - `Characters/` – Personajes utilizados en el entorno.
    - `MannequinsXR/` – Modelos base y esqueletos optimizados para realidad virtual extendida.
  - `Fab/` – Carpeta que agrupa elementos descargados mediante el marketplace de Unreal.
    - `Megascans/` – Repositorio de materiales escaneados de alta calidad.
      - `Surfaces/` – Superficies específicas utilizadas en escenas.
  - `LevelPrototyping/` – Elementos para diseño preliminar de niveles.

        - `Materials/`, `Meshes/` – Materiales y mallas usados para prototipado.
    - `Levels/` – Niveles jugables o en desarrollo.
        - `Keyboard/` – Nivel asociado al uso de teclado.
        - `Palacio/` – Nivel principal del entorno virtual.
            - `ActorsAndBP/` – Actores y *Blueprints* usados en el nivel.
            - `Animaciones/` – Animaciones de personajes u objetos.
            - `Imagenes/` – Recursos gráficos.
            - `Interfaces/` – Interfaces para interacción usuario-juego.
            - `StaticMesh_Blueprints/` – Mallas estáticas con lógica personalizada.
    - `Menu/` – Elementos del menú principal de la aplicación.
    - `Modelos/` – Modelos 3D usados en la aplicación.
        - `Palacio/` – Carpeta con submodelos y utilidades específicas del entorno tipo palacio.
            - `Base/`, `Box01/`, `Coins/`, `Crate/`, `FishingRod/`, `Floor/`, `Maps/`, `MaterialLayers/`, `MeshBlueprints/`, `Palace_Box/`, `Props/`, `Sounds/`, `Torres/` – Subcarpetas con modelos, sonidos y elementos visuales del entorno.
    - `Saves/` – *Savegames* utilizados en el proyecto
    - `UI/` – Interfaz de usuario.
        - `Actors/`, `BluePrints/`, `Buttons/`, `Imagenes/` – Componentes visuales interactivos de la UI.
    - `VRTemplate/` – Plantilla base de *Unreal* para proyectos VR.
        - `Audio/`, `Blueprints/`, `Haptics/`, `Input/`, `Materials/`, `Textures/`, `VFX/` – Recursos multimedia, entradas hápticas, materiales, texturas y efectos visuales usados en VR.
- `Motor/` – Recursos internos del motor Unreal que no deben modificarse directamente.

D.3. Manual del programador

El presente manual está dirigido a los desarrolladores que trabajen en el mantenimiento, ampliación o comprensión del sistema.

Instalaciones necesarias

Para el desarrollo y despliegue del proyecto, se requieren las siguientes herramientas y plataformas:

Unreal Engine 5

Unreal Engine 5 es el motor principal utilizado para desarrollar el entorno de realidad virtual interactivo. Se ha elegido por su potencia gráfica, compatibilidad con *VR* y soporte para programación en C++.

- Página oficial de descarga: Unreal Engine
- Versión utilizada en el proyecto: 5.4

Pasos de instalación:

1. Crear una cuenta en *Epic Games*.
2. Descargar el *Epic Games Launcher*.
3. Instalar *Unreal Engine 5.4* desde el lanzador.

Blender

Blender se ha utilizado para el modelado y exportación de objetos 3D empleados en el puzle.

- Página oficial de descarga: Blender
- Versión utilizada en el proyecto: 3.6 LTS

Pasos de instalación:

1. Descargar el instalador desde la página oficial.
2. Ejecutar el archivo y seguir el asistente de instalación.
3. Verificar la exportación en formato `.fbx` para su compatibilidad con Unreal Engine.

Visual Studio 2022 Community

Visual Studio es el entorno de desarrollo empleado para compilar y depurar el código C++ del proyecto.

- Página oficial de descarga: Visual Studio
- Edición recomendada: *Community Edition*

Componentes requeridos:

- *Desarrollo para escritorio con C++*
- *Herramientas de desarrollo para Unreal Engine*

Meta Quest Link

Para ejecutar el entorno en un dispositivo *Meta Quest 2*, es necesaria la aplicación *Meta Quest Link*, que permite conectar el visor al PC y ejecutar las escenas directamente desde Unreal Engine.

- Página oficial de descarga: Meta Quest Link

GitLab

El código fuente del proyecto se encuentra alojado en un repositorio privado de *GitLab*.

Pasos para clonar el repositorio:

1. Abre una terminal y navega al directorio deseado.
2. Ejecuta:

   ```
   git clone https://gitlab.com/HP-SCDS/Observatorio/2024-2025/vrrecoverygym/ubu-vrrecoverygym
   ```

3. Abre el proyecto desde el *Epic Games Launcher* o desde Unreal Engine directamente.

Hardware utilizado

El hardware mínimo necesario para desarrollar y ejecutar el proyecto incluye:

- **PC de desarrollo** con las siguientes características recomendadas:
  - Procesador: Intel i7 / AMD Ryzen 7 o superior.
  - Memoria RAM: mínimo 16 GB.
  - Tarjeta gráfica: NVIDIA RTX 3060 o superior.
  - Almacenamiento: SSD de al menos 500 GB.

  Estas especificaciones no son las mínimas, pero son las recomendadas para un desarrollo cómodo sobre el proyecto.
- **Dispositivo de realidad virtual:** *Meta Quest 2* con conexión mediante *Meta Quest Link*.
  - Página oficial del producto: Meta Quest 2
  - Requiere cable USB 3.0 compatible (recomendable por rendimiento) o conexión *Wi-Fi* para *Air Link*.

  Existe la posibilidad de probar el proyecto con gafas de versiones superiores como las Meta Quest 3 o 3s.

D.4. Compilación y ejecución del proyecto

Una vez el repositorio esté clonado, abre Unreal Engine y ve a *Archivo* > *Abrir* proyecto, navega hasta la carpeta donde está el proyecto y abre el archivo .uproject.

Verificación de dependencias: Se recomienda verificar desde el menú *Editar* > *Complementos* que los complementos *OpenXR* y *VictoryBPLibrary* estén activados.

Ejecución del proyecto

Para ejecutar el proyecto directamente en el visor VR, es necesario:

- Tener correctamente configurado el entorno de desarrollo con el dispositivo.

- Conectar las gafas al PC mediante *Air Link* o cable USB-C a traves de la aplicación Meta Quest Link.

- Presionar el botón *Play* y seleccionar la opción *VR Preview*.

El nivel por el que empezarás es el que esté abierto, por defecto es *Level_Start* que es el primer nivel del juego, si necesitas hacer pruebas sobre otro nivel tan solo ábrelo en el editor y sigue los mismos pasos.

D.5. Pruebas del sistema

Las pruebas del sistema constituyen una fase esencial para validar el correcto funcionamiento del proyecto, detectar errores y verificar que se cumplen los objetivos definidos. En el caso de un entorno de rehabilitación motriz en realidad virtual, estas pruebas incluyen tanto comprobaciones técnicas como pruebas de experiencia de usuario.

A continuación, se describen los principales tipos de pruebas realizadas:

Pruebas funcionales

Estas pruebas tienen como objetivo verificar que las distintas funcionalidades del sistema operan correctamente, de acuerdo con los requisitos establecidos.

- **Carga del entorno**: Se prueba que el entorno se carga sin errores y todos los elementos visuales aparecen correctamente.

- **Interacción con objetos**: Se verifica que el usuario puede manipular los objetos mediante controladores *VR* (agarre, colocación, rotación, etc.).

- **Seguimiento de manos y controladores**: Se comprueba que el sistema reconoce correctamente los movimientos del usuario, así como la respuesta háptica.

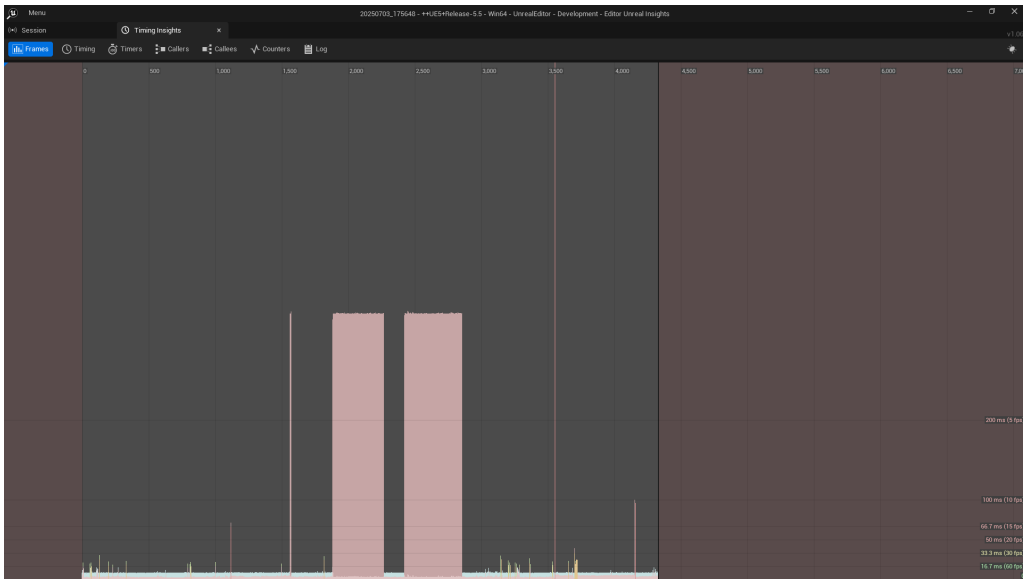
- **Detección de finalización de tareas**: Se prueban los mecanismos de validación de los puzles o ejercicios, asegurando que el sistema reconoce cuándo se ha completado correctamente una tarea.

Pruebas de rendimiento

Se evalúa el comportamiento del sistema bajo condiciones normales de uso, garantizando una experiencia fluida y sin interrupciones:

- **Tasa de fotogramas por segundo (FPS)**: Se mide el rendimiento con herramientas con la herramienta *Unreal Insights*. buscando mantener al menos 72 FPS en el visor VR.
- **Consumo de memoria**: Se evalúa el uso de recursos para asegurar que el sistema no sobrepasa los límites del *hardware*.
- **Tiempos de carga**: Se comprueba que el inicio del entorno y las transiciones entre niveles (si los hubiera) no introducen retardos perceptibles.

Figura D.2: Ejemplo de telemetria del proyecto con *Unreal Insights*



Pruebas de usuario

Se planifican sesiones de prueba con usuarios colaboradores, objetivo de evaluar:

- **Accesibilidad**: Evaluar si el sistema puede ser utilizado de forma sencilla por personas con movilidad reducida en extremidades superiores.

- **Usabilidad**: Identificar barreras de uso, errores de diseño o elementos confusos.
- **Recogida de feedback**: A través de entrevistas, se recoge información cualitativa sobre la experiencia del usuario y posibles mejoras.

Registro de incidencias

Durante todas las fases de prueba, se utiliza un sistema de control de incidencias mediante *GitLab Issues* y documentando su resolución dentro del flujo de trabajo de *SCRUM*.

Automatización de pruebas

En esta fase del desarrollo se han priorizado pruebas manuales debido al carácter interactivo y centrado en la experiencia de usuario del sistema.

Apéndice E

Documentación de usuario

E.1. Introducción

Este manual explica los requisitos necesarios para utilizar correctamente los dispositivos y el programa, así como las instrucciones de instalación y uso.

E.2. Requisitos de usuarios

A continuación se muestra una tabla con los requisitos mínimos necesarios para ejecutar la aplicación a través de *Meta Quest Link*.

Componente	Requisitos mínimos	Comentarios
CPU	Intel i5-4590 o AMD Ryzen 5 1500X o superior	
RAM	8GB o superior	Memoria RAM recomendada para ejecutar entornos en VR.
GPU	Tarjeta gráfica con soporte para codificación H.264/H.265	Recomendado: NVIDIA GTX 970, AMD RX 570 o superior.
Sistema operativo	Windows 10 o Windows 11 (64 bits)	Compatible con la aplicación de escritorio de Meta Quest Link.
USB	Puerto USB-C o USB3.0	.
Almacenamiento	5GB de espacio libre	Necesario para instalar el software de Meta Quest Link y la aplicación VRRecoveryGym

Tabla E.1: Requisitos mínimos del sistema para ejecutar el proyecto mediante *Meta Quest Link*.[4]

Instalación de *Meta Quest Link*

Para poder ejecutar el sistema de rehabilitación en unas gafas *Meta Quest 2* conectadas al ordenador, es necesario utilizar la herramienta *Meta Quest Link*, que permite el uso del visor como pantalla de realidad virtual para aplicaciones instaladas en el PC.

A continuación, se indican los pasos que debe seguir el usuario para instalar y conectar correctamente el dispositivo:

1. **Descargar el software oficial**: Acceder a la página oficial y descargar la aplicación *Meta Quest Link* para Windows.
2. **Instalar la aplicación**: Ejecutar el instalador descargado y seguir las instrucciones en pantalla para completar la instalación.
3. **Conectar las gafas al ordenador**:
   - **Opción por cable**: Conectar las gafas al PC mediante un cable USB-C.
   - **Opción inalámbrica**: Conectarse por Wi-Fi mediante la función *Air Link*. Para ello, tanto el visor como el ordenador deben estar

en la misma red local y con una buena conexión (preferiblemente a través de router 5 GHz).

4. **Aceptar la conexión en el visor**: Al conectar el dispositivo, aparecerá una notificación en las gafas solicitando permiso para utilizar *Link*. El usuario debe aceptar la conexión y activar el modo *Meta Quest Link*.

5. **Acceso al entorno VR**: Una vez establecida la conexión, el visor mostrará la interfaz de *Link*, desde donde se podrá ejecutar la aplicación.

Este proceso solo debe realizarse una vez; en futuras sesiones, las gafas reconocerán el ordenador automáticamente y permitirán acceder al sistema con rapidez.

E.3. Instalación

Para instalar la aplicación VRRecoveryGym en Windows se ha de seguir los siguientes pasos:

1. Acceda al repositorio para descargar la carpeta `VRRecoveryGym1.0`.

2. Asegúrese de que sus gafas están conectadas mediante *Meta Quest Link*. Acceda a la carpeta y ejecute TFG.exe

E.4. Manual del usuario

En este apartado se explican todas las acciones que el usuario puede realizar dentro de la aplicación.

Registro

El registro se hace nada más iniciar la aplicación por primera vez y hay que seguir estos pasos:

1. Introducir un nombre mediante el teclado de realidad virtual, con longitud mínima de 3 caracteres y máxima de 20.

2. Presionar el botón de enter.

Figura E.1: Pantalla de registro.

Apéndice F

Anexo de sostenibilización curricular

F.1. Introducción

La sostenibilidad es un eje fundamental para afrontar los retos sociales, ambientales y económicos del presente y del futuro. Su integración en el contexto universitario permite al estudiantado desarrollar competencias clave para promover un desarrollo más justo, eficiente y respetuoso con el entorno.

El Trabajo de Fin de Grado desarrollado, contribuye a varios de los Objetivos de Desarrollo Sostenible (ODS) establecidos por la ONU en la Agenda 2030. Esta aplicación de realidad virtual no solo busca mejorar la rehabilitación motriz de personas con discapacidad en extremidades superiores, sino que lo hace desde una perspectiva inclusiva, accesible y digitalmente eficiente.

A continuación, se exponen los ODS más relevantes relacionados con el proyecto.

F.2. ODS 3: Salud y Bienestar

El objetivo principal de *VRRecoveryGym* es facilitar la rehabilitación de usuarios con dificultades de movilidad, contribuyendo directamente a su bienestar físico y emocional. El sistema permite realizar ejercicios terapéuticos a través de puzles interactivos en un entorno virtual accesible desde el propio domicilio.

Impacto en la práctica:

- **Rehabilitación accesible desde casa:** Los usuarios pueden realizar sus sesiones sin necesidad de desplazarse a las instalaciones físicas de la fundación.
- **Bienestar emocional:** El hecho de poder elegir cuándo y dónde realizar los ejercicios reduce la carga psicológica asociada al tratamiento.
- **Estímulo cognitivo y motor:** El entorno gamificado incentiva la participación y la repetición terapéutica.

F.3. ODS 9: Industria, Innovación e Infraestructura

El proyecto incorpora tecnologías emergentes como la realidad virtual y el desarrollo en Unreal Engine para proponer una solución innovadora a un problema tradicional del ámbito sanitario. Esto permite democratizar el acceso a recursos terapéuticos de calidad.

Impacto en la práctica:

- **Digitalización del tratamiento:** Eliminación del soporte en papel para el seguimiento y registro de datos, que ahora se almacenan digitalmente mediante *SaveGames*.
- **Uso eficiente de recursos:** No se requiere infraestructura física adicional, ya que se ejecuta en un entorno doméstico con equipamiento común.

F.4. ODS 10: Reducción de las desigualdades

Al permitir que cualquier persona con un visor VR y un ordenador compatible pueda acceder al tratamiento, se reducen barreras geográficas y económicas. Esto facilita la inclusión de personas con movilidad reducida o con acceso limitado a centros especializados.

Impacto en la práctica:

- **Accesibilidad:** Solución adaptable a diferentes perfiles de usuario sin necesidad de asistencia externa.
- **Independencia:** Empodera al usuario en su propio proceso terapéutico.

F.5. ODS 13: Acción por el clima

Aunque de forma indirecta,el proyecto también contribuye al ODS 13 al reducir la huella ambiental asociada a los desplazamientos físicos y al uso de materiales impresos. Cada sesión realizada desde casa implica una menor emisión de gases de efecto invernadero, al evitar el uso de transporte privado o colectivo hacia instalaciones especializadas.

Además, al prescindir completamente del papel en el registro y seguimiento de la actividad, el sistema evita el consumo de recursos naturales y la energía empleada en los procesos de impresión, archivo y distribución.

Impacto en la práctica:

- **Reducción de desplazamientos:** El usuario no necesita acudir presencialmente a la fundación, lo que implica una menor dependencia de transporte motorizado.
- **Disminución del uso de papel:** Toda la información relativa al progreso del jugador y al estado del entorno se guarda digitalmente, eliminando la necesidad de formatos físicos.
- **Sostenibilidad digital:** El modelo propuesto apuesta por tecnologías limpias para gestionar la rehabilitación sin generar residuos materiales.

F.6. Conclusión

VRRecoveryGym representa una aproximación tecnológica al servicio de la sostenibilidad humana y social. Permite transformar un proceso tradicionalmente presencial y costoso en una experiencia accesible, digital y centrada en el bienestar del usuario.

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] Meta Platforms Inc. Requisitos del sistema para meta quest link. https://www.meta.com/es-mx/help/quest/140991407990979, 2024. Accedido el 30 de junio de 2025.

[5] Microsoft. Visual studio licensing terms, 2024. [Consulta: mayo 2025].

[6] Red.es. ¿qué es una licencia de software?, 2020. [Consulta: mayo 2025].