

Grado Universitario en Ingeniería Informática  
2023-2024

*Trabajo Fin de Grado*

“Desarrollo de un videojuego Roguelike multijugador”

Álvaro Morata Hontanaya

Tutor

Alejandro Rey López

Leganés, 2024



Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento – No Comercial – Sin Obra Derivada**

Resumen

ABSTRACT

Dedicatoria

Índice de contenidos

[1. INTRODUCCIÓN 1](#_Toc167904667)

[1.1. Motivación 1](#_Toc167904668)

[1.2. Objetivos 3](#_Toc167904669)

[1.3. Estructura del documento 4](#_Toc167904670)

[2. ESTADO DEL ARTE 6](#_Toc167904671)

[2.1. Videojuegos y el modo multijugador 6](#_Toc167904672)

[2.1.1. Definición 6](#_Toc167904673)

[2.1.2. Comienzos y evolución 6](#_Toc167904674)

[2.1.3. Topologías de red en el diseño de juegos multijugador 7](#_Toc167904675)

[2.1.3.1. Topologías de área local 7](#_Toc167904676)

[2.1.3.2. Arquitecturas de área extensa 8](#_Toc167904677)

[2.1.4. Propiedad y autoridad sobre los objetos 13](#_Toc167904678)

[2.1.4.1. Autoridad del servidor vs. Autoridad del cliente 13](#_Toc167904679)

[2.1.5. Soluciones para implementar el multijugador en Unity 14](#_Toc167904680)

[2.1.6. Proceso de conexión de la arquitectura Cliente-Anfitrión 16](#_Toc167904681)

[2.2. El género *roguelike* 17](#_Toc167904682)

[2.2.1. Juegos del género *roguelike* similares a la solución propuesta 19](#_Toc167904683)

[2.3. Motores de desarrollo de videojuegos 21](#_Toc167904684)

[2.3.1. Comparativa de los tres motores de videojuegos 23](#_Toc167904685)

[3. ANÁLISIS 25](#_Toc167904686)

[3.1. Casos de uso 25](#_Toc167904687)

[3.1.1. Diagramas de los casos de uso 25](#_Toc167904688)

[3.1.1.1. Diagrama del escenario 1 26](#_Toc167904689)

[3.1.1.2. Diagrama del escenario 2 28](#_Toc167904690)

[3.1.1.3. Diagrama del escenario 3 29](#_Toc167904691)

[3.1.2. Descripción de los casos de uso 30](#_Toc167904692)

[3.2. Definición de los requisitos 30](#_Toc167904693)

[3.2.1. Requisitos funcionales 31](#_Toc167904694)

[3.2.2. Requisitos no funcionales 31](#_Toc167904695)

[3.3. Matriz de trazabilidad 31](#_Toc167904696)

[3.4. Tecnologías utilizadas 31](#_Toc167904697)

[3.5. Metodología de desarrollo utilizada 31](#_Toc167904698)

[4. DISEÑO 32](#_Toc167904699)

[5. EVALUACIÓN 33](#_Toc167904700)

[6. PLANIFICACIÓN 34](#_Toc167904701)

[7. MARCO REGULADOR 35](#_Toc167904702)

[8. ENTORNO SOCIOECONÓMICO 36](#_Toc167904703)

[8.1. Presupuesto 36](#_Toc167904704)

[8.2. Impacto socioeconómico 36](#_Toc167904705)

[9. CONCLUSIONES 37](#_Toc167904706)

[BIBLIOGRAFÍA 38](#_Toc167904707)

[DEVELOPMENT OF A ROGUELIKE MULTIPLAYER VIDEOGAME 43](#_Toc167904708)

Índice de figuras

[Fig. 1.1. 50 años de ingresos por videojuegos [1] 1](#_Toc167908619)

[Fig. 2.1. Recreación del videojuego Tennis for Two [13] 6](#_Toc167908620)

[Fig. 2.2. Overcooked 2, juego que implementa el multijugador de sofá con un modo cooperativo [22] 8](#_Toc167908621)

[Fig. 2.3. Esquema de una red P2P 9](#_Toc167908622)

[Fig. 2.4. Arquitectura Cliente-Servidor 10](#_Toc167908623)

[Fig. 2.5. Topología Cliente-Anfitrión 11](#_Toc167908624)

[Fig. 2.6. Arquitectura de autoridad distribuida 13](#_Toc167908625)

[Fig. 2.7. Captura de pantalla de Rogue en su versión de 1980 [41] 18](#_Toc167908626)

[Fig. 2.8. Captura de pantalla de Enter the Gungeon 20](#_Toc167908627)

[Fig. 2.9. Captura de pantalla de Nuclear Throne 21](#_Toc167908628)

[Fig. 2.10. El juego de VR Beat Saber, desarrollado en Unity [59] 22](#_Toc167908629)

[Fig. 3.1. Diagrama de casos de uso del escenario 1 26](#_Toc167908630)

[Fig. 3.2. Diagrama de casos de uso del escenario 2 28](#_Toc167908631)

[Fig. 3.3. Diagrama de casos de uso del escenario 3 29](#_Toc167908632)

Índice de tablas

[Tabla 1.1. Plataformas Objetivo de Unity [6] 2](#_Toc167908963)

[Tabla 2.1. Diferencias entre Autoridad del Servidor y Autoridad del Cliente 14](#_Toc167908964)

[Tabla 2.2. Diferencias entre NGO y Netcode for Entities 15](#_Toc167908965)

[Tabla 2.3. Unity vs Unreal vs Godot 23](#_Toc167908966)

[Tabla 3.1. Casos de uso del escenario 1 27](#_Toc167908967)

[Tabla 3.2. Casos de uso del escenario 2 28](#_Toc167908968)

[Tabla 3.3. Casos de uso del escenario 3 29](#_Toc167908969)

[Tabla 3.4. Plantilla de la descripción de los casos de uso 30](#_Toc167908970)

[Tabla 3.5. Caso de uso CU01 30](#_Toc167908971)

[Tabla 3.6. Caso de uso CU02 30](#_Toc167908972)

[Tabla 3.7. Caso de uso CU03 30](#_Toc167908973)

[Tabla 3.8. Caso de uso CU04 31](#_Toc167908974)

[Tabla 3.9. Caso de uso CU05 31](#_Toc167908975)

[Tabla 3.10. Caso de uso CU06 31](#_Toc167908976)

[Tabla 3.11. Caso de uso CU07 31](#_Toc167908977)

[Tabla 3.12. Caso de uso CU08 31](#_Toc167908978)

[Tabla 3.13. Caso de uso CU09 32](#_Toc167908979)

[Tabla 3.14. Caso de uso CU10 32](#_Toc167908980)

[Tabla 3.15. Caso de uso CU11 32](#_Toc167908981)

[Tabla 3.16. Caso de uso CU12 32](#_Toc167908982)

[Tabla 3.17. Caso de uso CU13 33](#_Toc167908983)

[Tabla 3.18. Caso de uso CU14 33](#_Toc167908984)

[Tabla 3.19. Caso de uso CU15 33](#_Toc167908985)

[Tabla 3.20. Caso de uso CU16 33](#_Toc167908986)

[Tabla 3.21. Caso de uso CU17 33](#_Toc167908987)

[Tabla 3.22. Caso de uso CU18 34](#_Toc167908988)

[Tabla 3.23. Caso de uso CU19 34](#_Toc167908989)

[Tabla 3.24. Caso de uso CU20 34](#_Toc167908990)

[Tabla 3.25. Plantilla de los Requisitos 34](#_Toc167908991)

1. INTRODUCCIÓN

Este apartado introductorio trata de definir las motivaciones por las que se ha realizado el trabajo de fin de grado sobre esta temática en concreto, además de los objetivos que se busca conseguir mediante su desarrollo.

Asimismo, se incluye también un apartado en el que se explica la estructura que sigue este documento, ofreciendo una breve descripción de cada una de las partes que lo componen.

* 1. Motivación

En la actualidad, la industria de los videojuegos es el sector líder en el mercado del entretenimiento, con aproximadamente 3 mil millones de jugadores alrededor del mundo y generando ingresos superiores a los 180 mil millones de dólares en el año 2022 [1], [2], [3].

Gráfico

Descripción generada automáticamente

Fig. 1.1. 50 años de ingresos por videojuegos [1]

Como se puede observar en la figura anterior, con datos ofrecidos por la firma anglosajona Pelham Smithers, los ingresos generados por la industria de los videojuegos han seguido un crecimiento exponencial, empezando a estancarse a partir del año 2022. No obstante, este estancamiento no ha supuesto que esta forma de entretenimiento pierda su liderazgo en el ámbito del ocio. En 2023, este sector generó unos ingresos globales de 184 mil millones de dólares, una subida del 0,6% respecto a 2022. Además, se estima que para 2026, esta cifra supere los 205,4 mil millones de dólares [4].

Adicionalmente, se puede apreciar que la mayor parte de la cuota de mercado de este sector se corresponde con las plataformas móviles, siendo responsable de alrededor de un 50% de los ingresos totales. Los juegos desarrollados para ordenadores personales están detrás de los anteriores, con una cuota de mercado aproximada del 25%, siguiendo de cerca las consolas de videojuegos.

Con este trasfondo se puede empezar a hablar de Unity. Unity Engine es un motor de desarrollo 3D, en tiempo real y multiplataforma, disponible para Microsoft Windows, Linux y macOS [5]. Esta plataforma ha sido la elegida para el desarrollo de este proyecto, por diversas motivaciones que se exponen a continuación.

En primer lugar, Unity permite a los diseñadores y desarrolladores lanzar sus productos en las siguientes plataformas:

Tabla 1.1.  
Plataformas Objetivo de Unity [6]

| **Plataforma objetivo** | **Sistema Operativo/Plataforma** |
| --- | --- |
| Escritorio | Microsoft Windows  Mac  Linux  Universal Windows Platform |
| Móviles | iOS  Android |
| Realidad extendida | ARKit  ARCore  Microsoft HoloLens  Windows Mixed Reality  Magic Leap  Oculus  PlayStation VR |
| Consolas | PlayStation 4 y PlayStation 5  Xbox One y Xbox Series X|S  Nintendo Switch  Google Stadia |
| Plataformas embebidas | Embedded Linux  QNX |
| Web | WebGL |
| TV | tvOS  Android TV |

Esta extensa lista de plataformas permite al equipo de desarrollo realizar juegos multiplataforma de una manera sencilla y eficaz.

Otra de las motivaciones que han llevado a escoger esta plataforma de desarrollo es que incorpora un plan personal gratuito, por lo que no es necesario pagar una licencia para poder empezar a desarrollar en este motor.

Además, según un estudio realizado por SlashData en 2017, Unity es el motor de videojuegos más popular entre los desarrolladores, teniendo una cuota de mercado del 38% [7]. Esta popularidad se traduce en una mayor documentación disponible en Internet, además de la extensa documentación de Unity, como su manual de usuario y API de *scripting* [8].

La última razón por la que se ha decidido utilizar Unity es porque no se ha utilizado previamente en el grado universitario, lo que proporciona un componente de aprendizaje nuevo y un reto, al tener que aprender a utilizar una nueva herramienta profesional desde cero y aprender un nuevo lenguaje de programación, ya que Unity utiliza como lenguaje de programación C#, un lenguaje multiparadigma de código abierto desarrollado por Microsoft [9].

En cuanto a los motivos por los que se ha decidido diseñar un videojuego del género *roguelike*, se presenta lo siguiente.

El primer motivo es el interés personal. Este género de videojuegos se ha popularizado en los últimos años gracias al auge de los estudios de desarrollo *indie*, quienes han desarrollado videojuegos que han marcado un nuevo estándar en la industria, como Supergiant Games, Motion Twin o Dodge Roll. Esta popularidad se puede observar también en el crecimiento del valor en el mercado de los juegos de este género, que en 2023 consiguieron generar unos ingresos de 23.153 millones de dólares y, según Valuates Reports, se proyecta que para 2030 esta cifra ascienda a los 57.336 millones de dólares [10].

Otro de los motivos es que la mayoría de los juegos del género *roguelike* no disponen de un modo multijugador, si no que se suelen centrar en un modo para un jugador, al menos en el caso de los mayores exponentes de los últimos años, como The Binding of Isaac, Dead Cells, Enter the Gungeon, Hades, etcétera. Esto ofrece la posibilidad de implementar algo relativamente innovador en este género, debido al bajo porcentaje de videojuegos que han implementado esta mecánica.

Por último, el hecho de realizar un juego multijugador permite afianzar y profundizar en conocimientos ya adquiridos durante la carrera, principalmente el conocimiento sobre redes de ordenadores, paso de mensajes, conexión cliente-servidor y sincronización entre equipos.

* 1. Objetivos

Teniendo en cuenta las motivaciones detalladas en el apartado anterior, se pueden enumerar los objetivos que se pretende completar con la ejecución de este trabajo de fin de carrera. Estos objetivos se enumeran a continuación:

1. Diseño y desarrollo desde cero de un prototipo completamente funcional de un videojuego, con elementos del género *roguelike* y con un modo multijugador, en el que al menos dos jugadores puedan actuar simultáneamente.
2. Aprender a utilizar el motor de desarrollo Unity y sus herramientas, desde el punto de vista de una persona sin conocimiento previo sobre motores de videojuegos.
3. Aprender a programar en el lenguaje de programación C#, siguiendo una serie de buenas prácticas, reglas y convenciones de nombramiento y código, como las establecidas por Microsoft [11].
4. Reforzar y profundizar los conocimientos aprendidos en el grado de ingeniería informática, principalmente de asignaturas como redes de computadores, sistemas distribuidos, interfaces de usuario y tecnologías de desarrollo para la web.
5. Documentar adecuadamente el proceso creativo y de desarrollo de todo el trabajo de fin de grado mediante la elaboración de esta memoria.
   1. Estructura del documento

En este epígrafe se detalla la estructura que sigue este documento, además de dar una breve descripción sobre cada uno de los capítulos y el contenido que se puede encontrar en el interior de cada uno de ellos.

1. Introducción. El capítulo actual. En este apartado preparatorio se introducen las motivaciones por las que se ha elegido este tema, además de exponer los objetivos que se busca efectuar con la elaboración de este proyecto final de carrera. Adicionalmente se incluye este epígrafe, en el que se detalla la estructura del documento.
2. Estado del arte. En esta segunda sección se expone la definición, historia y evolución de los videojuegos multijugador, el género *roguelike* y tecnologías relevantes a la elaboración de este proyecto, como los motores de desarrollo de videojuegos y el lenguaje C#. Asimismo, se estudian las alternativas, realizando una síntesis de las ventajas y desventajas que pueden ofrecer las mismas.
3. Análisis. Este capítulo explora el problema planteado, analizando en detalle los requisitos y casos de uso para tener en cuenta, para una correcta ejecución del proyecto, además de estudiar su trazabilidad. También presenta las tecnologías utilizadas y metodología de desarrollo seguida en la ejecución de este trabajo, además de las justificaciones de su elección.
4. Diseño. En esta sección se detalla el apartado técnico del proyecto, así como las decisiones de diseño tomadas durante su ejecución. Se exponen las principales funcionalidades y mecánicas desarrolladas además de los sistemas lógicos más importantes del proyecto.
5. Evaluación. Se corresponde con el plan de pruebas ejecutado para la verificación del correcto funcionamiento del prototipo implementado y los resultados obtenidos. Además, se exponen posibles problemas y soluciones a estos.
6. Planificación. En este capítulo se presenta la planificación seguida durante el desarrollo de este trabajo, apoyándose en un diagrama de Gantt y presentando una estimación del tiempo que ha llevado cada proceso del trabajo de fin de grado.
7. Marco regulador. En él se detalla la legislación aplicable a un proyecto de esta índole. Asimismo, se detallan los estándares seguidos en el proceso y licencias aplicables a los programas y servicios utilizados.
8. Entorno socioeconómico. Sección en la que se detalla el presupuesto de la elaboración de este trabajo de fin de grado. Adicionalmente, en este apartado se detalla el impacto socioeconómico que tiene un proyecto de estas características.
9. Conclusiones. Objetivos cumplidos y justificación de cómo se han satisfecho. En adición, se especifican posibles ampliaciones que se pueden ejecutar sobre este proyecto.
10. Bibliografía. Esta sección contiene todas las referencias bibliográficas que se han utilizado a lo largo de este dosier.
11. Anexos.
12. Development of a roguelike multiplayer videogame. Resumen en inglés del documento.
13. ESTADO DEL ARTE

En este segundo capítulo se presentan y definen las principales bases teóricas sobre las que se construye este proyecto, así como las alternativas estudiadas y proyectos con fines similares.

* 1. Videojuegos y el modo multijugador
     1. Definición

Los videojuegos multijugador son aquellos que permiten a dos o más jugadores interactuar en tiempo real. Esta interacción puede tener lugar en el mismo entorno, como un mismo ordenador o a través de una red, ya sea local o una red de área extensa, como Internet.

* + 1. Comienzos y evolución

El primer proyecto que se considera como el primer videojuego multijugador de la historia es el título Tennis for Two, un juego creado por el físico nuclear William Higinbotham que fue introducido por primera vez en el año 1958. Tennis for Two permitía a dos usuarios jugar simultáneamente a un partido de tenis simulado mediante el uso de un ordenador analógico y utilizando un osciloscopio como pantalla [12].

Una mesa de madera

Descripción generada automáticamente con confianza media

Fig. 2.1. Recreación del videojuego Tennis for Two [13]

Otros videojuegos que se consideran pioneros en este sector por diversas fuentes son el Pong (1972), Astro Race (1973), Spacewar! (1962). Este último se trata del primer videojuego desarrollado para un computador comercial, el Programmed Data Proceessor-1 [14], [15], [16], [17].

El mayor punto de inflexión en los juegos multijugador local se produjo en la década de los ochenta con la aparición de diversas consolas, pero en particular con la Nintendo Entertainment System, con juegos como Contra, Mario Bros. o Donkey Kong [18].

Cuando realmente se popularizaron los juegos multijugador fue en la década de los 90, con la aparición de juegos como Doom en 1993 y los juegos MMO, del inglés *Massively Multiplayer Online games*, que introdujeron la posibilidad de jugar en línea con una cantidad elevada de jugadores, como Ultima Online en 1997 [14], [19].

Alrededor del año 2000 nacieron juegos multijugador que han conseguido perdurar durante más de dos décadas, como es el caso de la saga Counter Strike, que apareció por primera vez en el año 1999 y 25 años después, sigue teniendo más de un millón de jugadores activos a diario [20].

A partir de aquí, la industria del videojuego ha seguido un curso estable. En la actualidad, se pueden observar todo tipo de juegos multijugador con una alta cantidad de jugadores activos, desde MMOs hasta juegos cooperativos de 2 a 4 jugadores simultáneos. Según SteamDB, una plataforma que recopila información de la plataforma de distribución de videojuegos para ordenadores llamada Steam, la cifra de usuarios simultáneos interactuando en algún juego multijugador oscila entre los 5 millones y 11 millones de usuarios, dependiendo del momento del día [21].

* + 1. Topologías de red en el diseño de juegos multijugador

Una topología o arquitectura de red establece cómo se organiza una red de computadores. En este epígrafe se explican las diferentes topologías de red que normalmente se aplican en el diseño de videojuegos multijugador. Se procede a enumerarlas en orden de expansión, es decir, se empieza explicando topologías de área local y se termina con aquellas que permiten implementar un modo multijugador a través de una red de área amplia o WAN (*Wide Area Network*).

* + - 1. Topologías de área local
* Multijugador de sofá. Los multijugadores de sofá, de la expresión en inglés *couch multiplayer games*, son aquellos videojuegos que implementan el multijugador en una misma máquina, por ejemplo, dividiendo la pantalla en partes iguales con un punto de vista para cada jugador o usando una misma cámara para ambos individuos.

Una captura de un videojuego

Descripción generada automáticamente con confianza baja

Fig. 2.2. Overcooked 2, juego que implementa el multijugador de sofá con un modo cooperativo [22]

* Multijugador local. Esta topología permite a varios usuarios desde distintas máquinas conectarse a una misma sesión dentro de un videojuego. Aunque el acceso a Internet no es necesario, puesto que lo único que necesitan los dos equipos es estar conectas a una misma red de área local (LAN), sí que utilizan una arquitectura Anfitrión-Cliente, de modo que una de las máquinas actuará como un servidor, como se verá en el apartado correspondiente.
  + - 1. Arquitecturas de área extensa
* Red punto a punto. Las redes punto a punto, también conocidas como P2P (*peer-to-peer*), son redes descentralizadas en las que varios computadores comparten sus recursos sin necesidad de que haya un servidor intermedio, o que una de las máquinas actúe como anfitrión, si no que cada uno de los ordenadores conectados a la red tiene el mismo nivel de autoridad [23].

Estas topologías de red tienen una ventaja principal respecto a otras, y es que no necesitan un servidor intermedio al que conectarse, lo que abarata costes de manera significativa. No obstante, esto también se convierte en un inconveniente desde el punto de vista de la ciberseguridad, puesto que cualquier usuario de la red puede acceder a otros datos de cualquier equipo conectado a la red si no se hace una implementación correcta, además de ser vulnerable a diversos ciberataques [24].

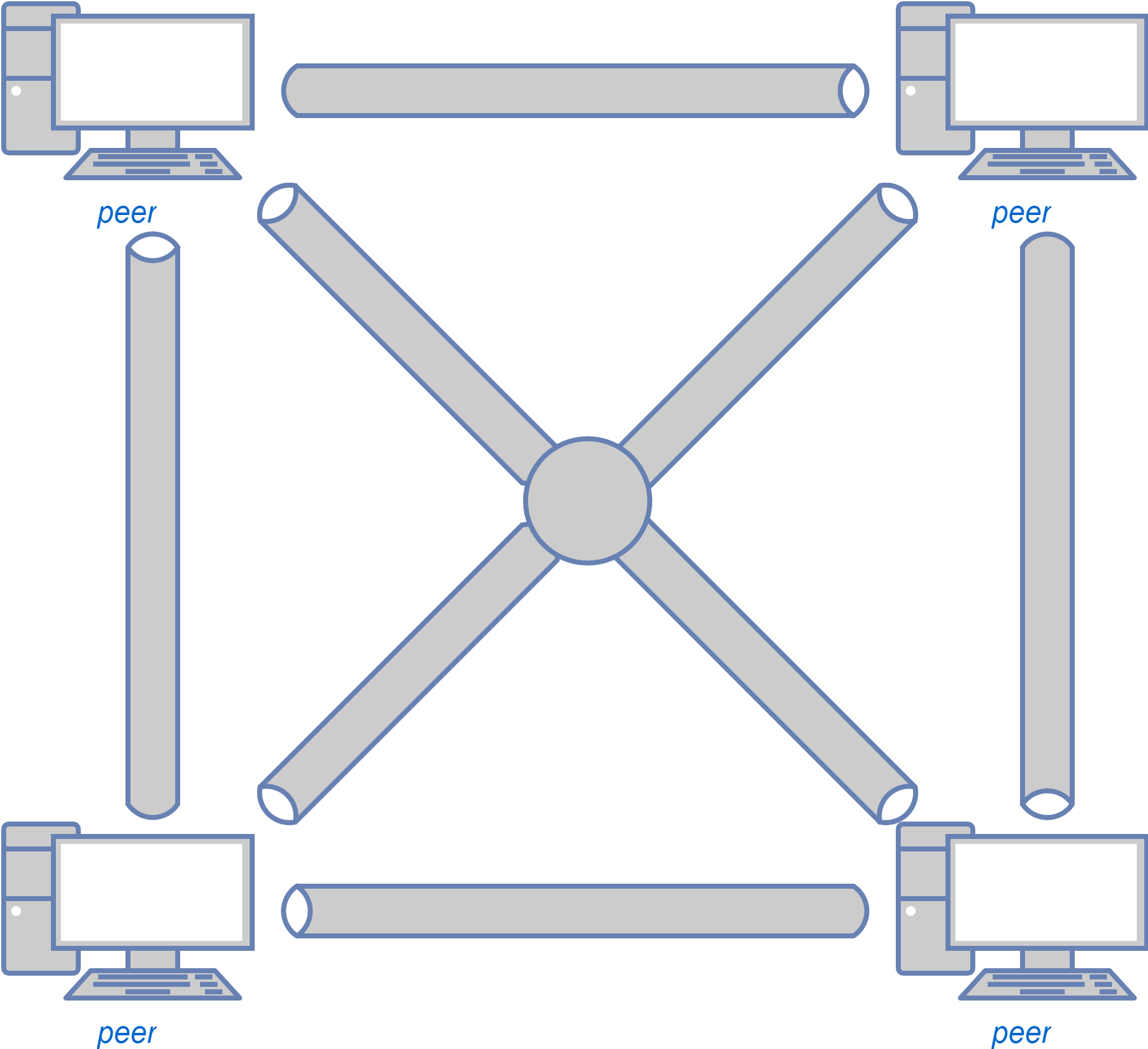


Fig. 2.3. Esquema de una red P2P

* Cliente-Servidor. Las arquitecturas de este tipo se componen de dos tipos de nodos. Por un lado, están los servidores, que son aquellos nodos que se encargan de conectar a los clientes entre sí, además de ocuparse de ejecutar todo el apartado lógico de los videojuegos. En el otro extremo están los clientes, quienes se conectan al servidor a través de peticiones o llamadas remotas para sincronizar los recursos con los otros componentes de la red.

Estas topologías tienen dos desventajas principales. La primera es el coste adicional que supone mantener un servidor dedicado. Otra desventaja que tiene es el aumento de latencia a la hora de sincronizar cambios a través de clientes, ya que todos los datos tienen que pasar previamente por el servidor para ser validados.

En cuanto a las ventajas que ofrece, se pueden encontrar por una parte el rendimiento, pues al usar un servidor para toda la lógica, los clientes pueden gozar de un mayor rendimiento general, al tener una menor carga de trabajo en esas máquinas. El uso de un servidor dedicado también tiene una ventaja significativa desde el punto de vista de la seguridad; al verificarse toda la lógica en el servidor, se puede evitar el uso de modificaciones ilegales por parte de los clientes que afecten al estado del juego o den ventajas significativas a uno de los jugadores en concreto. Esta última ventaja es la razón esencial por la que todos los juegos de la escena competitiva, también denominados *esports*, utilizan esta arquitectura de red [25], [26].

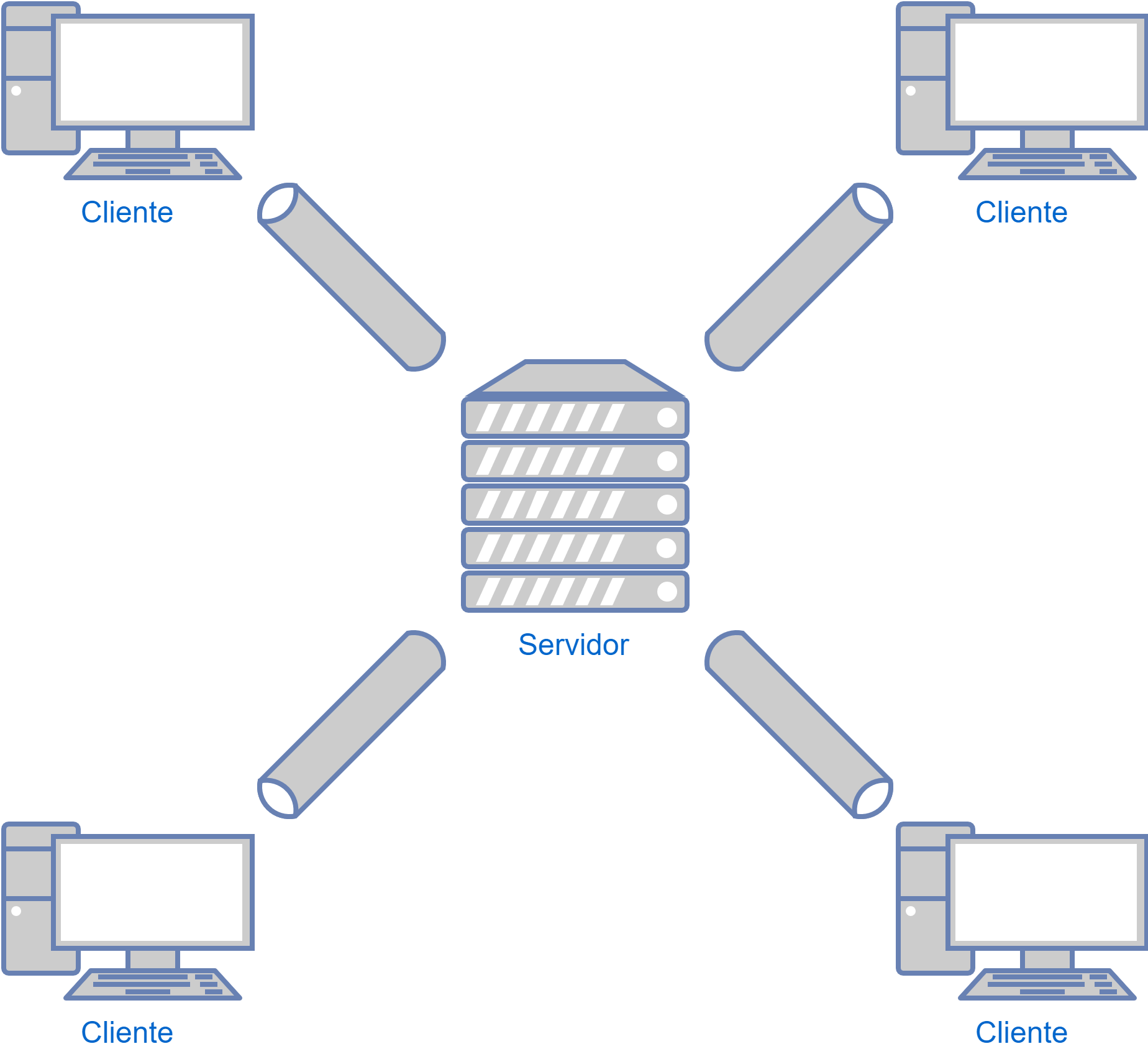


Fig. 2.4. Arquitectura Cliente-Servidor

* Cliente-Anfitrión. A diferencia de las redes de la topología Cliente-Servidor, en esta arquitectura no existe un servidor dedicado, si no que uno de los clientes actúa como cliente y servidor, como anfitrión. Por una parte, en la máquina del anfitrión se inicia un servidor de escucha. Este servidor de escucha tiene la función, al igual que en la topología anterior, de procesar las peticiones de los clientes conectados a la red y de ejecutar toda la lógica del programa. Asimismo, el anfitrión también ejecuta el código que procesa cualquier otro cliente [25].

Esta topología tiene una ventaja fundamental, siendo esta que se abaratan los costes al no necesitar un servidor dedicado, lo que permite centrar los costes de desarrollo en otras áreas del proyecto. Además, no necesitar un servidor significa que no es necesario utilizar ningún tipo de infraestructura especializada, por lo que agiliza el despliegue y permite al videojuego funcionar independientemente de que un nodo raíz, como es un servidor, esté en ejecución. Esta topología de red es, además, la recomendada cuando se quiere añadir una opción de multijugador a un videojuego que originalmente era de un solo jugador, puesto que facilita su desarrollo. Adicionalmente, es ideal para videojuegos donde no se requiere un mundo persistente y el número de jugadores en una misma sesión de juego es menor de 12 jugadores [27].

No obstante, esta arquitectura presenta varios problemas:

* La máquina del anfitrión tiene una mayor carga de trabajo, por lo que el rendimiento puede verse degradado si este equipo no posee unos componentes que cumplan unos requisitos mínimos, lo que puede afectar no solo al anfitrión, sino también a los clientes.
* El rendimiento de la red depende de la conexión a Internet del anfitrión, siendo esta en la mayoría de los casos residencial. Esto se puede traducir a un aumento de la latencia o tiempo de respuesta en el caso de que el comportamiento de la red no sea estable.
* Al ejecutar el anfitrión toda la lógica y tener acceso al servidor, es más fácil para el anfitrión cometer trampas, haciendo que pueda llegar a ser injusto el juego para los clientes en el caso de que se presente esta casuística. Por esta razón, esta arquitectura es inadecuada por norma general en juegos que tienen un aspecto competitivo.
* En el caso de que el anfitrión, es decir, el servidor, deje de funcionar o salga del juego, todos los clientes perderán la conexión a esa sesión de juego.

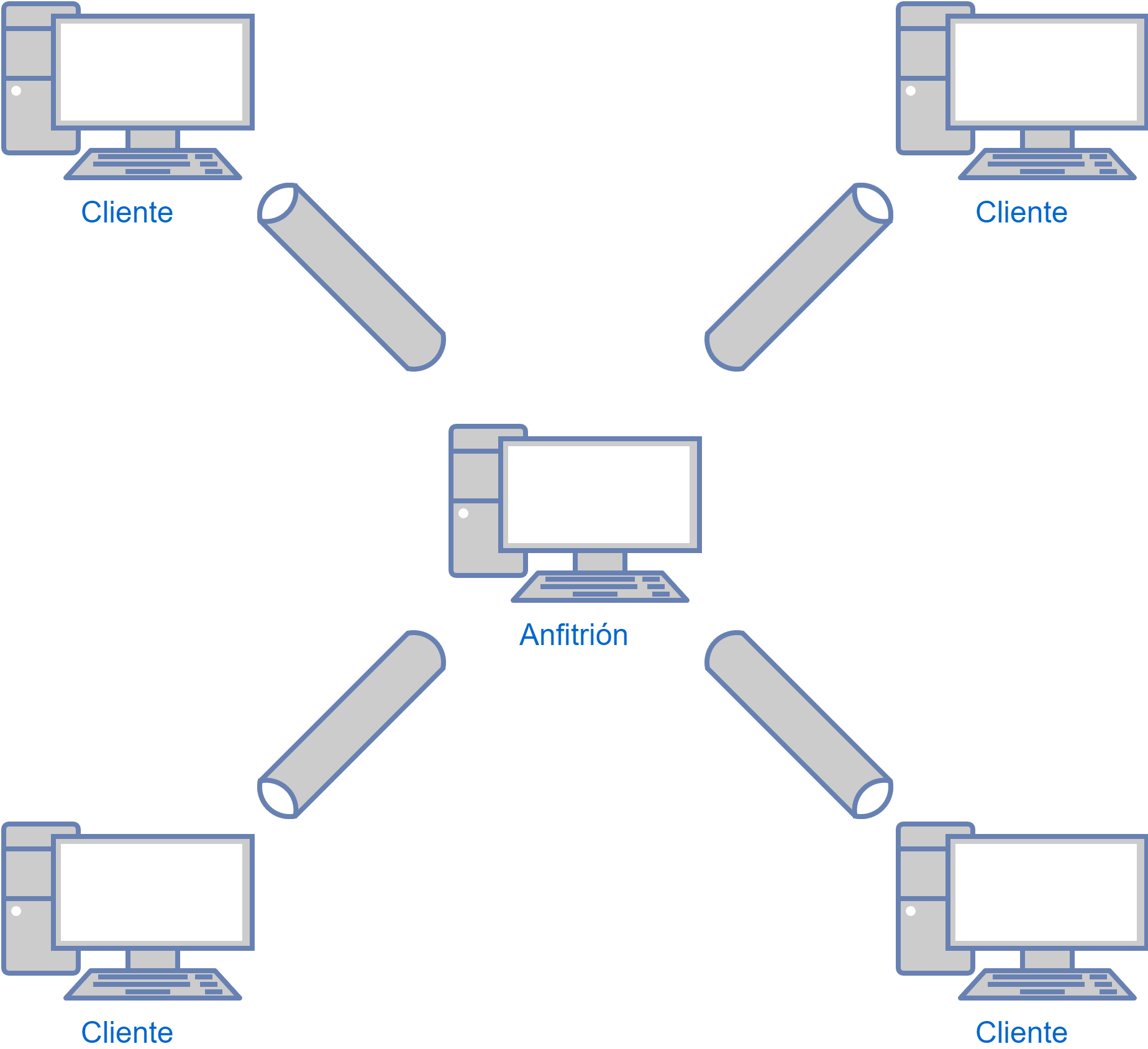


Fig. 2.5. Topología Cliente-Anfitrión

Esta topología es la que se ha utilizado en el desarrollo de este proyecto, por los motivos que se exponen a continuación:

* En cuanto a costes, es más barata que las infraestructuras que necesitan un servidor dedicado o un servicio central en la infraestructura, aunque por lo general es más cara que una red P2P.
* Ofrece un buen balance entre seguridad y rendimiento de la red, aunque la mayoría de la red dependa de un único individuo, el anfitrión, generalmente de origen doméstico.
* Su complejidad es relativamente más baja que en el resto de las arquitecturas, dado que no se necesita gestionar un servidor o un servicio centralizado como en una red de autoridad distribuida. Además, esta última requiere de un enfoque distinto a la hora de implementar las distintas funcionalidades de un proyecto. Por ejemplo, puede llegar a ser necesario desarrollar nuevos sistemas de físicas.

Dado que esta arquitectura de red es la que se emplea en el desarrollo de este trabajo, se explica en detalle, en los tres siguientes epígrafes, cómo funcionan en Unity algunos temas importantes en el contexto de este proyecto, como las diferentes formas de implementar el multijugador en este motor, la conexión de nuevos clientes a una red de este tipo, o cómo funciona la autoridad y apropiación de objetos de la red.

* Red de autoridad distribuida. A diferencia de la topología Cliente-Servidor y su derivada, donde el servidor se encarga de toda la lógica del sistema y es propietario de todos los objetos del entorno, siendo este el único responsable de instanciar nuevos objetos, esta topología propone que los clientes sean capaces de gestionar objetos ellos mismos. Asimismo, al no existir un servidor dedicado que se encargue de procesar la lógica del sistema, cada uno de los clientes tiene que realizar una simulación parcial del estado del juego, sincronizando los cambios al resto de clientes. Sin embargo, aunque no existe un servidor dedicado, sí que es necesario que haya un servicio central que se encargue de realizar un seguimiento de los cambios y enrute el tráfico de red. Esto es principalmente lo que diferencia esta topología de las redes punto a punto [28].

Esta arquitectura tiene una ventaja principal respecto a la topología Cliente-Anfitrión y se trata del concepto de propietario de la sesión. En estas redes, se asigna a uno de los clientes como propietario de la sesión, teniendo permisos similares a los de un anfitrión, siendo el encargado de gestionar tareas que tienen que ver con el estado general del juego. No obstante, siempre y cuando el estado global se sincronice entre todos los clientes de manera adecuada, si el propietario abandona la sesión, uno de los clientes será designado como propietario de la sesión, pasando a encargarse del estado global de la partida. Otras ventajas que incluye esta arquitectura son unos mejores tiempos de respuesta, al no haber un intermediario que se encargue de validar la lógica y una menor carga de trabajo, al distribuirse toda la carga entre todos los clientes conectados.

Las desventajas notables que presenta esta arquitectura son principalmente dos. La primera de ellas se trata de juegos en los que se involucre un alto uso de físicas o la precisión de estas sea importante. Al no haber una sola unidad encargándose de toda la simulación de las físicas, es necesario tomar un enfoque distinto a la hora de implementar la interacción entre objetos. La segunda desventaja que tiene es que es relativamente fácil modificar el estado del juego por parte de los clientes, por lo que el uso de esta arquitectura en juegos de carácter competitivo no está recomendado.

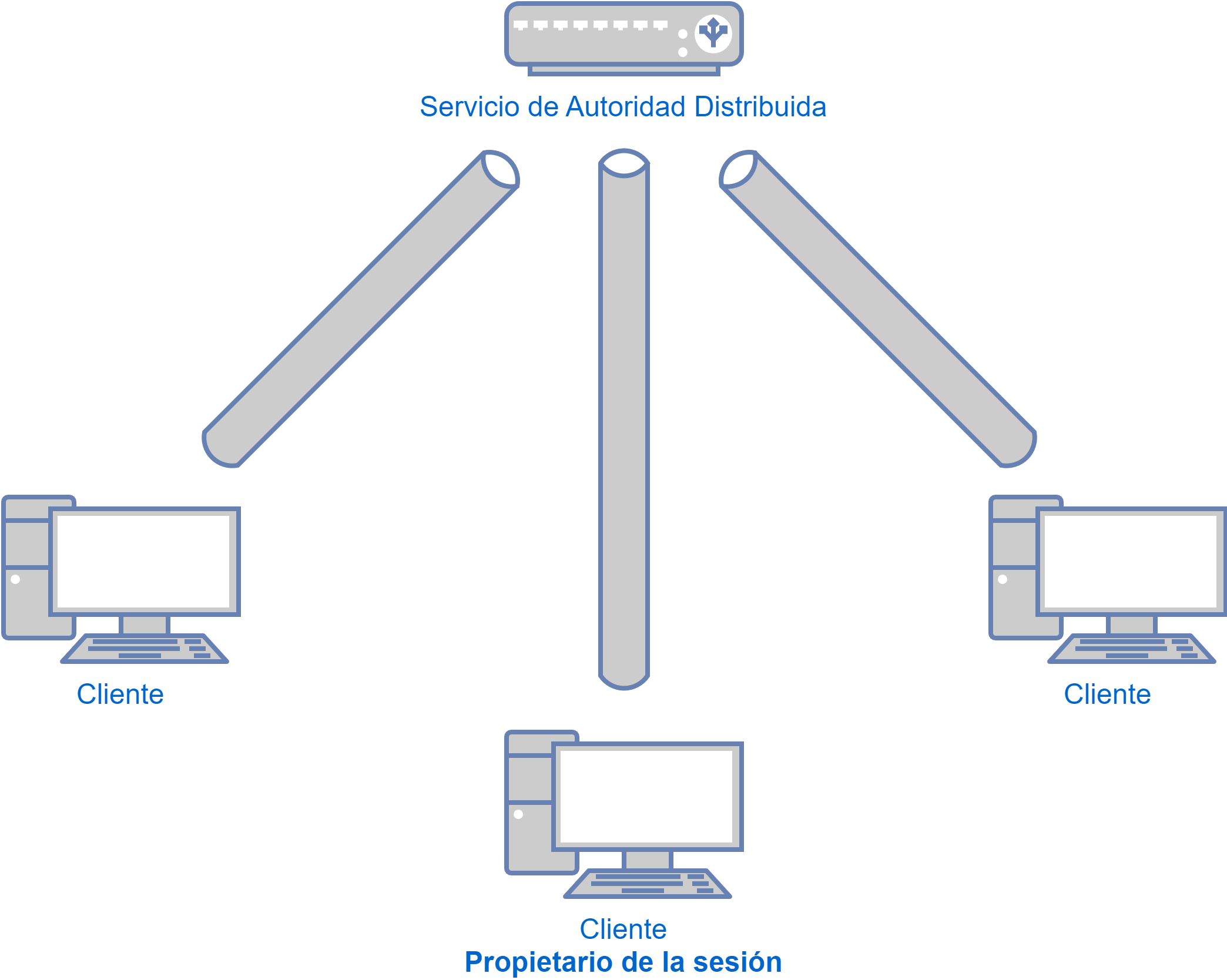


Fig. 2.6. Arquitectura de autoridad distribuida

* + 1. Propiedad y autoridad sobre los objetos

En Unity, la propiedad sobre un objeto de red se refiere a qué actor en específico tiene el control sobre dicho objeto, pudiendo cambiar su estado y propagando estos cambios al resto de clientes de la red. En el caso de la autoridad, se refiere a qué entidad de la red tiene permiso para tomar decisiones finales sobre el objeto de red, como su creación o destrucción, o un cambio de propietario [29].

En las arquitecturas Cliente-Servidor y derivadas, quien tiene autoridad y propiedad sobre todos los objetos de red es el servidor, siendo este el único responsable de realizar cambios sobre ellos, como su instanciación, modificación o destrucción, por lo que en el caso de que un cliente quiera realizar un cambio sobre un objeto de red, es necesario que este envíe una petición al servidor y que este ejecute los cambios. En una topología de autoridad distribuida, la autoridad siempre la tiene el propietario de la sesión, mientras que la propiedad de los objetos de red se puede transferir entre los distintos clientes conectados de manera dinámica.

* + - 1. Autoridad del servidor vs. Autoridad del cliente

En Unity, normalmente es el servidor quien posee la autoridad sobre los objetos de la escena. No obstante, es posible cambiar la autoridad de un objeto a un cliente, por ejemplo, para el objeto de red que represente al jugador asociado. Esto tiene ciertas ventajas y desventajas, como se va a observar con la siguiente tabla [30]:

Tabla 2.1.  
Diferencias entre Autoridad del Servidor y Autoridad del Cliente

|  |  |  |
| --- | --- | --- |
|  | **Autoridad del servidor** | **Autoridad del cliente** |
| **Seguridad** | Mayor seguridad. Al ser todas las peticiones de los clientes verificadas por el servidor, se evita cualquier modificación posible que pueda afectar al entorno de juego. | Menor seguridad. No se verifican las acciones del cliente, si no que se sincronizan al servidor directamente los cambios. Esto implica que cualquier cliente puede modificar el estado del juego. Por esta razón, su uso no está aconsejado en juegos donde hay un carácter competitivo o existe el *Player versus Player* (jugador contra jugador). |
| **Respuesta** | Menor capacidad de respuesta. Como todas las acciones tienen que ser verificadas por el servidor, esto implica que, por ejemplo, si un cliente quiere mover su jugador, tiene que realizar una petición al servidor y que este realice el cambio, sincronizando posteriormente los cambios realizados por el servidor. Esto se traduce en un aumento del tiempo de respuesta y si la conexión con el servidor no es estable, puede dar lugar a problemas de sincronización por parte del cliente. No obstante, el resto de los clientes verán los cambios reflejados correctamente. | Mayor capacidad de respuesta. Al no tener que realizar una petición al servidor para realizar una acción, el cliente que la ejecuta observa los cambios realizados de manera instantánea. Esto se traduce en una menor latencia por parte del cliente, lo que es ideal en juegos que no sean competitivos y se desee que los clientes tengan una mayor capacidad de respuesta. |
| **Sincronización** | No existen problemas de sincronización. El servidor es quién ejecuta todas las acciones, por lo que, si un cliente tiene una conexión inestable con el servidor, el resto de los clientes no se verán afectados y se seguirán sincronizando todos los cambios. | Pueden existir problemas de sincronización. En el caso de que un cliente no tenga una conexión estable con el servidor, los cambios que realice este cliente no se sincronizarán adecuadamente con el servidor, por lo que el resto de los clientes no observarán el mismo comportamiento que el cliente local que está sufriendo la inestabilidad de red. |

* + 1. Soluciones para implementar el multijugador en Unity

En la actualidad, Unity dispone de dos soluciones oficiales para la implementación del multijugador, siendo estas Netcode for GameObjects y Netcode for Entities. Además, incluye otras implementaciones desarrolladas por la comunidad o empresas privadas. En este proyecto, solo se han valorado las soluciones de Fish-Networking y Mirror.

Netcode for GameObjects, en adelante NGO, es una biblioteca oficial de alto nivel desarrollada por Unity, que permite al desarrollador enviar datos del mundo y de los objetos de la sesión, en Unity llamados *GameObjects* (en castellano, objetos del juego), a través de una red a todos los jugadores de la sesión. Actualmente, está disponible en todas las plataformas de escritorio, móviles y realidad extendida, además de la mayoría de plataformas cerradas, como las consolas [31]. Esta solución ha sido la elegida en el desarrollo de la práctica, al ser una solución con soporte oficial y una documentación muy detallada.

Por otro lado, Unity ofrece la biblioteca de Netcode for Entities. Esta biblioteca, que forma parte de Unity DOTS (*Data-Oriented Technology Stack*), depende fundamentalmente de la arquitectura ECS (*Entity Component System*) de Unity. Esta arquitectura emplea entidades, o *entities*, como unidades discretas para representar un conjunto de datos, a diferencia de las unidades que utiliza por defecto Unity, los *GameObjects*. Al contrario de estos últimos, una entidad actúa como un identificador agrupando componentes únicos, en vez de contener directamente código o servir como un contenedor asociado a otros componentes, como es en el caso de los *GameObjects* [32], [33].

A continuación, se muestra una tabla resumen con las diferencias entre estas dos soluciones.

Tabla 2.2.  
Diferencias entre NGO y Netcode for Entities

|  | **Netcode for GameObjects** | **Netcode for Entities** |
| --- | --- | --- |
| **Arquitecturas de red soportadas** | Por defecto, NGO trabaja como una topología Cliente-Servidor, pero también permite utilizar topologías Cliente-Anfitrión y arquitecturas de autoridad distribuida. | Solo soporta topologías Cliente-Servidor y Cliente-Anfitrión. |
| **Unidades sobre las que trabaja** | Utiliza los componentes por defecto de Unity, los denominados *GameObjects* y sus componentes, como *Rigidbody*, *SpriteRenderer,* etc. | Emplea las *entities* del ECS y componentes especiales, como sistemas de físicas. |
| **Complejidad** | Funciona exactamente igual que el resto de los componentes estándar de Unity, por lo que no es necesario realizar ningún cambio significativo en la forma de trabajar. | Requiere saber utilizar el ECS de Unity y depende directamente de este, por lo que cambia completamente la manera de trabajar. |
| **Juegos objetivo** | Se recomienda para juegos de pequeña a mediana escala, como juegos cooperativos. No obstante, también es capaz soportar juegos multijugador de una escala mayor. | Es recomendable para juegos de gran escala y juegos competitivos que requieran de métodos de predicción del cliente. |

En cuanto a las bibliotecas desarrolladas por entidades externas a Unity, se ha realizado un estudio de dos de ellas, expuestas a continuación:

* En primer lugar, se habla de Mirror Networking, una solución de código abierto creada por la comunidad que nació a raíz UNET, una solución para la gestión del multijugador lanzada por Unity en 2015 y que en la actualidad está obsoleta. Mirror nació a raíz de la falta de soporte que Unity daba a UNET y al descontento de la comunidad, solucionando diversos problemas que tenía UNET una vez se liberó cierta parte del código, la parte de alto nivel (HLAPI). Una vez Unity dejó de dar soporte a UNET, Mirror creció como una alternativa totalmente distinta, ofreciéndose el día de hoy como una solución alternativa para la implementación del multijugador en videojuegos desarrollados en Unity [34].
* Fish-Networking es una solución desarrollada por la compañía First Gear Games. Es una solución que tiene como objetivo proporcionar flexibilidad, eficacia, facilidad de uso y fiabilidad. Incorpora ciertas soluciones como predicción en la parte del cliente, compensación de latencia y soporte para diversos tipos de transporte y servidores de retransmisión [35].
  + 1. Proceso de conexión de la arquitectura Cliente-Anfitrión

En una topología Cliente-Anfitrión, todos los clientes (incluyendo el anfitrión) se conectan habitualmente desde una red doméstica. Esto presenta un problema, puesto que, salvo en raras excepciones, en estas redes se realiza una traducción de la dirección de red, conocido comúnmente como NAT (*Network Address Translation*). En redes domésticas, se utiliza la traducción de direcciones de red para enmascarar la dirección IP (*Internet Protocol*) privada de un usuario de la red doméstica, protegiéndolo del exterior y además permitiendo que el espacio de direcciones IPv4 no se sature. Esto se traduce a que, por defecto, todos los dispositivos de una red doméstica están asociados a una única IP pública, por lo que tiene que haber una manera de identificarlos desde el exterior, para que el rúter pueda enrutar el tráfico hacia el dispositivo de la red correcto [36].

Para que un cliente pueda conectarse a un anfitrión protegido por un NAT, es necesario encontrar un método que permita identificar al anfitrión desde una red externa a la de este último. En la actualidad, existen tres maneras de solventar esta circunstancia [27]:

* Redirección de puertos. Esta técnica consiste en abrir un puerto público en el rúter del anfitrión, redireccionando el tráfico que llega a ese puerto a la máquina concreta del anfitrión. Sin embargo, esta solución tiene varios inconvenientes. El primero de ellos es que esta solución se tiene que hacer de manera local. Es decir, para que un cliente se pueda unir a la sesión de este anfitrión, este último tiene que abrir un puerto desde su rúter o cortafuegos, por lo que es necesario que el usuario en cuestión tenga ciertos conocimientos técnicos, además de puede llegar a ser posible que esté en una situación en la que no pueda realizar esta acción; por ejemplo, si el anfitrión está utilizando un dispositivo móvil o una red pública. El otro inconveniente que presenta, son los riesgos de ciberseguridad que conlleva abrir un puerto, ya que esto permite a un atacante tener una ruta directa hacia el sistema del anfitrión.
* Servidor de retransmisión. Un servidor de retransmisión es un servidor dedicado en la nube o en un centro de datos, que tiene preparada previamente una redirección de puertos, lo que permite a los usuarios conectarse a ellos directamente. El servidor de retransmisión actúa como un intermediario entre el anfitrión y los clientes, donde los clientes envían paquetes al servidor y este se encarga de que lleguen al destino deseado. Esta opción, en comparación con la anterior, siempre está disponible para los clientes y el anfitrión, además de poder notificar a los clientes si el anfitrión se desconecta de la red o incluso iniciar un proceso de migración. Las desventajas que presenta esta solución es que tienen un coste añadido al utilizar una infraestructura especializada, además de que la latencia entre el anfitrión y los clientes puede ser mayor, puesto que para realizar la comunicación tienen que pasar previamente por el servidor de retransmisión.
* NAT *punch-through*. Esta técnica, que en castellano se podría traducir perforación de NAT, intenta abrir una conexión directa entre cliente y servidor sin hacer uso de una redirección de puertos. En caso de que esta acción resulte exitosa, los clientes estarán directamente conectados. No obstante, debido al tipo de traducción de direcciones de red que suele tener lugar entre los clientes, este proceso suele fallar. Por esto mismo, esta técnica se suele implementar con una alternativa, que en la mayoría de los casos consiste en usar un servidor de retransmisión en el caso de que la perforación de NAT no de resultado. En caso de que los clientes puedan realizar la conexión mediante la perforación de NAT, no se realizará la conexión a través del servidor de retransmisión, lo que abarata costes y carga de trabajo por parte de este último.

En el desarrollo de este proyecto se ha decidido utilizar la opción de un servidor de retransmisión, puesto que Unity ofrece el servicio Unity Relay, un servicio que permite a los desarrolladores utilizar un servidor de retransmisión de manera gratuita, siempre y cuando no se alcance una cuota de uso establecida por el plan mensual gratuito de Unity Gaming Services, como se verá más adelante en el capítulo del marco legal [37]. Como este proyecto durante su desarrollo no va a sobrepasar esa cuota, se ha decidido optar por su utilización.

* 1. El género *roguelike*

El término *roguelike* es un subgénero de videojuegos que recibe su nombre directamente del videojuego Rogue, lanzado en 1980. Este término surgió por primera vez en 1993, a raíz de una discusión en USENET entre usuarios que buscaban un nombre para videojuegos que tenían características parecidas a Rogue, como Adventure, Zork, Moria o Angband. Tras diversas discusiones, la comunidad llegó al consenso de un nombre para este tipo de juegos, que se denominarían como videojuegos del género *roguelike* (*rogue-like*, en castellano, “como Rogue”) [38], [39]. Según estas discusiones, el género se caracterizaba por estas particularidades [40]:

* El jugador controla un único personaje.
* Son juegos de rol de fantasía, que generalmente tienen lugar en una mazmorra.
* Poseen una interfaz gráfica sencilla, basada en caracteres ASCII.
* El objetivo del juego es volverse más fuerte, con el propósito de completar una misión difícil.
* Son juegos distribuidos de manera gratuita.

Imagen que contiene Escala de tiempo

Descripción generada automáticamente

Fig. 2.7. Captura de pantalla de Rogue en su versión de 1980 [41]

Hoy en día, esta definición ha evolucionado, eliminando algunas de las restricciones que presentaba, como que los juegos para pertenecer a este género tienen que ser de distribución gratuita, o que la interfaz tiene que ser simple, basada en caracteres o mosaicos. En la actualidad, un juego se puede considerar, por lo general, como un *roguelike* si cumple las siguientes características [42], [43]:

* Generación procedimental. La generación del entorno, enemigos y demás elementos de la escena es pseudoaleatoria, por lo que cada partida es distinta a la anterior.
* Jugabilidad por turnos. El transcurso del videojuego es parecido al de un juego de mesa, donde el jugador tiene un tiempo infinito para pensar una estrategia con la que poder completar el juego.
* El jugador controla a un solo personaje. Al igual que en la definición clásica, el jugador únicamente manejará a un individuo durante todo el transcurso de la partida.
* Consecuencias permanentes. Cualquier acción del jugador tiene consecuencias permanentes e inmutables. Esto incluye la muerte del personaje del jugador.
* Jugabilidad basada en nuevas partidas o *runs*. La muerte permanente del jugador implica que cada partida empezará desde el principio, perdiendo cualquier progreso que este haya hecho hacia la meta. Lo único que se conserva es la propia experiencia del jugador, conseguida de partidas anteriores.

Esta definición, bajo el contexto de los videojuegos actuales de este género, es la más popular. No obstante, al no ser un término que se haya formalizado, está en constante cambio. Del mismo modo, existen otras definiciones que han surgido de diferentes discusiones o eventos, como en la International Roguelike Development Conference, en Berlín, Alemania, en el año 2008 [44].

También es necesario mencionar un subgénero del anterior, el término *roguelite*. Este término se utiliza hoy día en aquellos videojuegos cuyas bases y características son las propias de los videojuegos *roguelike*, pero que incorporan cambios críticos a su definición, como puede ser algún tipo de progreso, por ejemplo, con el desbloqueo de nuevas armas, habilidades, o mejora de las estadísticas base del jugador [43].

En el prototipo desarrollado durante la realización de este proyecto, se han incluido ciertos elementos del género *roguelike*. La única característica que no se satisface en su totalidad es la generación aleatoria. En este prototipo, el entorno es siempre el mismo, puesto que la complejidad que supone crear un mecanismo de generación procedimental del escenario se aleja del objetivo de este proyecto, aumentando de manera excesiva la carga de trabajo. Sin embargo, se ha incorporado la generación de enemigos en el escenario pseudoaleatoria, por lo que cada partida es ligeramente distinta a la anterior.

* + 1. Juegos del género *roguelike* similares a la solución propuesta

A continuación, se van a listar algunos de los videojuegos del género *roguelike* que ofrecen características similares al desarrollado en este proyecto.

Enter the Gungeon.

Enter the Gungeon es un videojuego *roguelike* de disparos en el que el jugador, que controla a una serie de personajes marginados y arrepentidos con su pasado, tiene que superar los diversos desafíos impuestos por la Armazmorra, con el objetivo de hacerse con el tesoro que aguarda en esta: el arma que puede matar al pasado [45].

Este videojuego, desarrollado por el estudio independiente Dodge Roll y lanzado en 2016, es uno de los videojuegos *roguelike* mejor valorados de la última década, con un 95% de reseñas positivas en Steam y un 82 sobre 100 en Metacritic [46], [47]. En enero de 2020, Dodge Roll anunció que había vendido más de tres millones de copias juntando las ventas de todas las plataformas [48]. Este juego, al igual que el prototipo desarrollado en este proyecto se creó con el motor Unity.

En el prototipo desarrollado se pueden ver muchas semejanzas a este juego, principalmente porque este ha sido el primero que se ha utilizado como referencia. Algunas de las similitudes que se pueden observar son las siguientes:

* El estilo artístico utilizado es un estilo bidimensional basado en la forma de arte digital *pixel art*. Es utilizado principalmente por su simpleza
* Ambos juegos utilizan un punto de vista cenital.
* Tanto en Enter the Gungeon como en el prototipo desarrollado, el jugador controla a un personaje que puede disparar con armas a distancia a sus enemigos.
* En ambos juegos el escenario es una mazmorra de la que el jugador tiene que escapar.

Imagen de la pantalla de un video juego

Descripción generada automáticamente con confianza media

Fig. 2.8. Captura de pantalla de Enter the Gungeon

No obstante, Enter the Gungeon no posee un modo multijugador, siendo esta la principal diferencia entre este juego y el prototipo desarrollado a lo largo de este trabajo.

The Binding of Isaac.

The Binding of Isaac es otro *roguelike* que salió por primera vez al mercado en el año 2011, desarrollado por Edmund McMillen y Florian Himsl [49]. Este juego está inspirado fundamentalmente por la historia del Sacrificio de Isaac, recogida en la Biblia, como se puede deducir por su nombre o algunas de las referencias que contiene el videojuego.

Este videojuego fue muy bien recibido por el público, teniendo una media de un 91% de valoraciones positivas en Steam. En 2014, Edmund McMillen, junto a la compañía Nicalis Inc., realizaron una adaptación del juego original, The Binding of Isaac: Rebirth, esta versión, que incluye diversas mejoras respecto al juego original, es considerada por muchos críticos como uno de los mejores juegos *roguelike* de la historia, teniendo un 96,45% de reseñas positivas en Steam [50]. Además, entre ambas versiones, se estima que se han vendido alrededor de once millones de copias en todo el mundo [51].

Este juego, al igual que el prototipo, tiene una vista cenital y un estilo en dos dimensiones, donde el jugador controla a un personaje que dispara proyectiles (en este caso lágrimas).

Cabe destacar que en 2017 el juego recibió la expansión The Binding of Isaac: Afterbirth+, que incorpora la posibilidad de jugar en multijugador. No obstante, solo implementa la arquitectura de multijugador de sofá, aunque gracias al servicio de Steam Remote Play, es posible jugar con otros jugadores a través de Internet, pero con el prerrequisito de tener una cuenta en esta plataforma [52].

Nuclear Throne

De estos tres juegos, Nuclear Throne es el menos conocido de ellos, habiendo vendido alrededor de 680.000 copias [53]. No obstante, este juego tiene un 96% de reseñas positivas en Steam y un 89 sobre 100 en Metacritic [54], [55].

Al igual que los otros dos juegos, se trata de un *roguelike* de disparos en vista cenital y en 2D, con un estilo artístico simple y desarrollado por un estudio independiente, Vlambeer, compuesto por una sola persona [56]. Este juego, como ocurría con The Binding of Isaac, tiene un modo de juego multijugador local y la posibilidad de jugar con otra persona en línea a través de la solución Steam Remote Play.

Mapa

Descripción generada automáticamente

Fig. 2.9. Captura de pantalla de Nuclear Throne

* 1. Motores de desarrollo de videojuegos

Los motores de desarrollo de videojuegos son herramientas especializadas que ayudan a los diseñadores y programadores de videojuegos a crear este tipo de aplicaciones, formando un entorno de desarrollo que facilite y optimice todo el proceso de creación y mantenimiento del videojuego. Asimismo, los motores de videojuegos proporcionan un motor gráfico con el que poder renderizar los gráficos del programa, además de uno o más motores físicos que permitan emular las leyes de la física.

Aunque su uso está recomendado por la facilidad y organización que aportan, no son indispensables en la creación de videojuegos, pudiendo crear cada uno de los componentes necesarios o usando aplicaciones que implementen las funcionalidades necesitadas. Aun así, hoy en día su uso se ha normalizado. Según un estudio realizado por SlashData sobre una muestra de 2.325 desarrolladores de videojuegos, en la actualidad un 60% de ellos utiliza un motor de videojuegos [7].

A continuación, se va a hablar de algunos de estos motores, incluyendo el elegido para el desarrollo de este proyecto, Unity. Se incorpora un breve resumen de cada uno de estos motores de videojuegos y una tabla comparativa entre sus funcionalidades.

Unity Engine

Unity Engine, como ya se habló en el primer capítulo, es un motor de desarrollo 3D en tiempo real que permite tanto a desarrolladores como diseñadores crear videojuegos, además de otros tipos de contenidos animados, como simulaciones. Soporta una gran variedad de plataformas, tanto de sobremesa, dispositivos portables y otras plataformas, como las de realidad virtual, como se puede ver en la tabla I de este documento.

Unity Engine, creado por la compañía Over The Edge Entertainment (OTEE), se lanzó por primera vez al mercado en 2005 como un motor de videojuegos para Mac OS X, con el objetivo de crear un motor gráfico que estuviese disponible en varias plataformas, fuese asequible y fácil de utilizar. En 2005 se lanzó el primer videojuego creado en esta plataforma, GooBall, creado por OTEE. Este juego no acabó siendo rentable, pero permitió a OTEE contratar más personal para seguir desarrollando Unity Engine. A partir de aquí Unity fue ganando popularidad, primero en juegos para navegadores y dispositivos móviles, hasta la actualidad, donde es el motor de videojuegos líder del mercado [57].

Algunos ejemplos de títulos populares que se han desarrollado con este motor son Beat Saber (2019), Among Us (2018), Subnautica (2018), Hollow Knight (2017), Rust (2018), Pokémon GO (2016) o Genshin Impact (2020) [58].

Imagen que contiene Texto

Descripción generada automáticamente

Fig. 2.10. El juego de VR Beat Saber, desarrollado en Unity [59]

Unreal Engine

Unreal Engine es un motor de desarrollo de videojuegos 3D, lanzado por la compañía Epic Games en 1998. El primer juego que se desarrolló con este motor gráfico fue el juego Unreal, en el mismo año de su lanzamiento y por la misma compañía que creó el motor gráfico. Este juego fue uno de los primeros en enfatizar la atención al detalle y texturas de mayor resolución a las del resto de videojuegos de su época. Unreal Engine ha seguido evolucionando hasta su última versión principal, Unreal Engine 5, que se utiliza tanto en el desarrollo de videojuegos de todos los presupuestos, simulación y en la cinematografía [60], [61].

Algunos de los juegos más conocidos de los últimos años desarrollados con Unreal Engine son: Fortnite, Senua’s Saga: Hellblade II, Astroneer, Assetto Corsa Competizione, y Ark: Survival Evolved.

Godot Engine

Godot es un motor de videojuegos de código abierto, creado por Juan Linietsky y Ariel Manzur. En 2001 se empezó a desarrollar este motor, siendo un proyecto privativo creado por Juan, Ariel y la empresa en la que trabajaban, OKAM Studio, pero en 2014 se liberó su código bajo la licencia MIT, año en el que además salió su primera versión estable (Godot 1.0). Alrededor de 2016, cuando su versión 2.0 fue lanzada, el proyecto empezó a conseguir popularidad, lo que hizo que cada vez más personas contribuyesen en su desarrollo, agilizándolo y consiguiendo que hoy en día sea uno de los motores de videojuegos más utilizados [62], [63]. Actualmente, el repositorio de GitHub tiene más de 84.400 estrellas y en él han contribuido más de 2.500 personas [64].

Algunos juegos populares creados con Godot Engine son: Brotato (2022), Dome Keeper (2022), Buckshot Roulette (2024) y Casette Beasts (2023).

* + 1. Comparativa de los tres motores de videojuegos

A continuación, se presenta una tabla comparativa de las características de los tres motores gráficos mencionados:

Tabla 2.3.  
Unity vs Unreal vs Godot

| **Característica** | **Unity Engine** | **Unreal Engine** | **Godot Engine** |
| --- | --- | --- | --- |
| **Gráficos objetivo** | 3D y 2D | Solo admite 3D | 3D y 2D |
| **Lanzamiento** | 2005 | 1998 | 2014 |
| **Precio** | Gratis para uso personal, siempre y cuando no se generen unos ingresos superiores a los 100.000 dólares.  Planes de pago Unity Pro, Unity Enterprise y Unity Industry dependiendo del sector y tamaño de los equipos de desarrollo. | Gratis para particulares y pequeñas empresas (con menos de 1 millón de dólares en ingresos anuales).  Basado en regalías o en puestos si se genera más de 1 millón de dólares en ingresos. | Gratuito (licencia MIT) |
| **Plataformas objetivo** | Más de 24 plataformas distintas [6] | 17 plataformas [67] | 6 plataformas [68] |
| **Programación** | C#  Unity Visual Scripting | C++  Visual scripting con Blueprints [65] | GDScript [66]  C#  C++  Visual Scripting  Con extensiones de la comunidad: Rust, Nim, Python y JavaScript |

1. ANÁLISIS

En esta sección se definen los requisitos y casos de uso a partir de los cuales se ha realizado el diseño de este proyecto, además de la trazabilidad de los anteriores. Asimismo, se justifica la metodología de desarrollo utilizada y se detallan el conjunto de tecnologías empleadas.

* 1. Casos de uso

Los casos de uso de este proyecto se han desarrollado en base a tres escenarios o flujos distintos, cada uno con su diagrama correspondiente:

* Escenario 1 - Flujo de la partida. Manifiesta el curso de una partida, representando las acciones que puede tomar el jugador, así como las acciones que pueden ejecutar los enemigos que aparecen por la escena. Como las acciones que pueden realizar todos los jugadores son las mismas, se ha representado dicho escenario con un solo jugador.
* Escenario 2 - Escenario de conexión para el modo un jugador. Representa el flujo de conexión desde que se inicia el juego hasta que se inicia una partida en el modo de un jugador.
* Escenario 3 - Flujo de conexión para el modo multijugador. Expresa el proceso de conexión del cliente y el anfitrión para crear una partida del modo multijugador, hasta el momento en el que se inicia dicha partida.
  + 1. Diagramas de los casos de uso

En este epígrafe se muestran los diagramas de casos de uso de cada uno de los escenarios planteados. Posteriormente, se listan los casos de uso nuevos que aparecen en cada uno de estos diagramas, dándoles un identificador como el siguiente:

*CUXX*

Donde cada CU (Caso de Uso) tiene asociado un número identificativo XX.

* + - 1. Diagrama del escenario 1

Diagrama

Descripción generada automáticamente

Fig. 3.1. Diagrama de casos de uso del escenario 1

En este primer diagrama, aparecen los siguientes casos de uso:

Tabla 3.1.  
Casos de uso del escenario 1

| **Identificador** | **Nombre del caso de uso** |
| --- | --- |
| CU01 | Mover el PJ |
| CU02 | Evadir |
| CU03 | Apuntar el arma del PJ |
| CU04 | Disparar proyectil |
| CU05 | Curar al PJ |
| CU06 | Dañar al PJ |
| CU07 | Matar al PJ |
| CU08 | Perder la partida |
| CU09 | Ganar la partida |
| CU10 | Salir al menú principal |
| CU11 | Matar al enemigo |
| CU12 | Dañar al enemigo |
| CU13 | Atacar al PJ |
| CU14 | Perseguir al PJ |

Nótese que, en estos casos de uso, PJ significa Personaje Jugable, es decir, el jugador.

* + - 1. Diagrama del escenario 2

El diagrama para el segundo escenario es el siguiente, del que surgen dos nuevos casos de uso, como se puede apreciar en la tabla que se encuentra inmediatamente después de este.

Diagrama

Descripción generada automáticamente

Fig. 3.2. Diagrama de casos de uso del escenario 2

Tabla 3.2.  
Casos de uso del escenario 2

| **Identificador** | **Nombre del caso de uso** |
| --- | --- |
| CU15 | Iniciar modo un jugador |
| CU16 | Salir de la aplicación |

* + - 1. Diagrama del escenario 3

Diagrama

Descripción generada automáticamente

Fig. 3.3. Diagrama de casos de uso del escenario 3

En este diagrama se reutiliza el caso de uso *UC16 - Salir de la aplicación*. Así mismo, se incorporan los siguientes casos de uso nuevos:

Tabla 3.3.  
Casos de uso del escenario 3

| **Identificador** | **Nombre del caso de uso** |
| --- | --- |
| CU17 | Iniciar modo multijugador |
| CU18 | Crear una sala |
| CU19 | Unirse a una sala |
| CU20 | Establecerse como listo |

* + 1. Descripción de los casos de uso

A continuación, se va a proporcionar una descripción de alto nivel de cada caso de uso, mediante el uso de una tabla como la siguiente:

Tabla 3.4.  
Plantilla de la descripción de los casos de uso

| **CUXX** | |
| --- | --- |
| **Nombre** |  |
| **Actores** |  |
| **Descripción** |  |

En esta plantilla, se presentan los siguientes campos:

* **Nombre** del caso de uso.
* **Actores** involucrados.
* **Descripción** de alto nivel del caso de uso.

Tabla 3.5.  
Caso de uso CU01

| **CU01** | |
| --- | --- |
| **Nombre** | Mover el PJ |
| **Actores** | Jugador |
| **Descripción** | El jugador es capaz de mover a su personaje jugable por el escenario mediante una señal de entrada. |

Tabla 3.6.  
Caso de uso CU02

| **CU02** | |
| --- | --- |
| **Nombre** | Evadir |
| **Actores** | Jugador |
| **Descripción** | El jugador puede realizar una esquiva en la dirección en la que se está moviendo actualmente. Por ende, es necesario que el jugador esté previamente en movimiento. |

Tabla 3.7.  
Caso de uso CU03

| **CU03** | |
| --- | --- |
| **Nombre** | Apuntar el arma del PJ |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.8.  
Caso de uso CU04

| **CU04** | |
| --- | --- |
| **Nombre** | Disparar proyectil |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.9.  
Caso de uso CU05

| **CU05** | |
| --- | --- |
| **Nombre** | Curar al PJ |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.10.  
Caso de uso CU06

| **CU06** | |
| --- | --- |
| **Nombre** | Dañar al PJ |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.11.  
Caso de uso CU07

| **CU07** | |
| --- | --- |
| **Nombre** | Matar al PJ |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.12.  
Caso de uso CU08

| **CU08** | |
| --- | --- |
| **Nombre** | Perder la partida |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.13.  
Caso de uso CU09

| **CU09** | |
| --- | --- |
| **Nombre** | Ganar la partida |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.14.  
Caso de uso CU10

| **CU10** | |
| --- | --- |
| **Nombre** | Salir al menú principal |
| **Actores** | Jugador |
| **Descripción** |  |

Tabla 3.15.  
Caso de uso CU11

| **CU11** | |
| --- | --- |
| **Nombre** | Matar al enemigo |
| **Actores** | Enemigo |
| **Descripción** |  |

Tabla 3.16.  
Caso de uso CU12

| **CU12** | |
| --- | --- |
| **Nombre** | Dañar al enemigo |
| **Actores** | Enemigo |
| **Descripción** |  |

Tabla 3.17.  
Caso de uso CU13

| **CU13** | |
| --- | --- |
| **Nombre** | Atacar al PJ |
| **Actores** | Enemigo |
| **Descripción** |  |

Tabla 3.18.  
Caso de uso CU14

| **CU14** | |
| --- | --- |
| **Nombre** | Perseguir al PJ |
| **Actores** | Enemigo |
| **Descripción** |  |

Tabla 3.19.  
Caso de uso CU15

| **CU15** | |
| --- | --- |
| **Nombre** | Iniciar modo un jugador |
| **Actores** | Anfitrión |
| **Descripción** |  |

Tabla 3.20.  
Caso de uso CU16

| **CU16** | |
| --- | --- |
| **Nombre** | Salir de la aplicación |
| **Actores** | Anfitrión |
| **Descripción** |  |

Tabla 3.21.  
Caso de uso CU17

| **CU17** | |
| --- | --- |
| **Nombre** | Iniciar modo multijugador |
| **Actores** | Anfitrión, cliente |
| **Descripción** |  |

Tabla 3.22.  
Caso de uso CU18

| **CU18** | |
| --- | --- |
| **Nombre** | Crear una sala |
| **Actores** | Anfitrión |
| **Descripción** |  |

Tabla 3.23.  
Caso de uso CU19

| **CU19** | |
| --- | --- |
| **Nombre** | Unirse a una sala |
| **Actores** | Cliente |
| **Descripción** |  |

Tabla 3.24.  
Caso de uso CU20

| **CU20** | |
| --- | --- |
| **Nombre** | Establecerse como listo |
| **Actores** | Anfitrión, cliente |
| **Descripción** |  |

* 1. Definición de los requisitos

Cada requisito se especifica con una tabla como la que se encuentra a continuación:

Tabla 3.25.  
Plantilla de los Requisitos

|  |  |
| --- | --- |
| **Identificador:** | |
| **Descripción:** | |
| **Prioridad:** | **Necesidad:** |
| **Verificabilidad:** |  |

Donde cada uno de los campos representa lo siguiente:

* **Identificador:** representa la identificación del requisito con una cadena de caracteres con el formato *<F/NF>-<U/S>-<C/R><XXX>*, donde:
  + *<F/NF>*
    - **F:** requisito funcional.
    - **NF:** requisito no funcional.
  + *<U/S>*
    - **U:** requisito de usuario.
    - **S:** requisito de software.
  + *<C/R>*
    - **C:** requisito de capacidad.
    - **R:**requisito de restricción.
  + *<XXX>*
    - Número identificador del requisito.
* **Descripción:** breve explicación del requisito.
* **Prioridad:** indica la preferencia del requisito sobre el resto. Obtiene los siguientes valores:
  + **Alta.** El requerimiento es esencial para el funcionamiento del sistema y otros requisitos dependen de este.
  + **Media.** El requisito depende de otro de mayor prioridad y otros dependen de él.
  + **Baja.** Nivel de prioridad más bajo. La realización del requisito no es fundamental o depende de otros requisitos de mayor prioridad.
* **Necesidad:** representa la obligatoriedad del requisito. Obtiene los siguientes valores:
  + **Obligatorio:** el requisito es indispensable para el sistema.
  + **Opcional:** representa una característica optativa del sistema.
* **Verificabilidad:** indica si el requisito se puede verificar con un plan de pruebas. Puede obtener los siguientes valores:
  + **Total:** se puede verificar en su totalidad mediante un plan de pruebas.
  + **Parcial:** no se puede verificar en su totalidad su correcto funcionamiento.
  + **Nula:** no es posible verificar el requisito en base a las pruebas.
    1. Requisitos funcionales
    2. Requisitos no funcionales
  1. Matriz de trazabilidad
  2. Tecnologías utilizadas
  3. Metodología de desarrollo utilizada

Explicar qué metodología se va a utilizar en el desarrollo del proyecto y por qué (Kanban es posiblemente la que se utilice)

1. DISEÑO
2. EVALUACIÓN
3. PLANIFICACIÓN
4. MARCO REGULADOR

El TFG deberá incorporar el desarrollo de uno o varios de los siguientes apartados:

* Análisis de la legislación aplicable sobre la implementación descrita en el trabajo (riesgos, responsabilidades profesionales, responsabilidades éticas, riesgos laborales, privacidad y seguridad, etc.).
* Estándares técnicos, si son aplicables (sobre tecnología desarrollada, implantada, sobre lenguajes de programación o herramientas utilizados, etc.).
* Estudio de las cuestiones relacionadas con la propiedad intelectual de la idea (patentabilidad, protección…), por ejemplo, si es un trabajo teórico.

1. ENTORNO SOCIOECONÓMICO
   1. Presupuesto

Presupuesto de la elaboración del TFG.

* 1. Impacto socioeconómico

Impacto socioeconómico (impacto económico, social, medioambiental, ético, etc.) esperado de la aplicación del resultado del proyecto, plan de explotación de este, o consideraciones sobre aspectos económicos de la temática del trabajo. Para trabajos teóricos, se debe detallar en qué aplicaciones prácticas podría utilizarse y qué impacto socioeconómico podría generar en el sector de aplicación.

1. CONCLUSIONES

BIBLIOGRAFÍA

[1] P. Rao, «50 Years of Video Game Industry Revenues, by Platform», Visual Capitalist. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://www.visualcapitalist.com/video-game-industry-revenues-by-platform/

[2] «LÍDERES DEL ENTRETENIMIENTO: “¿QUIÉN MERECE SER REY?” | LinkedIn». Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://www.linkedin.com/pulse/l%C3%ADderes-del-entretenimiento-qui%C3%A9n-merece-ser-rey-emilio-hurtado-ruiz/

[3] «Estudio mercado videojuegos: ¿Cuánto dinero mueve? (infografía)», Bankinter. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://www.bankinter.com/blog/finanzas-personales/mercado-videojuegos-dinero-estudio-espana-mundo

[4] «Newzoo’s games market revenue estimates and forecasts by region and segment for 2023», Newzoo. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://newzoo.com/resources/blog/games-market-estimates-and-forecasts-2023

[5] «Plataforma y Editor de desarrollo 3D en tiempo real», Unity. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://unity.com/products/unity-engine

[6] «What platforms are supported by Unity?», Unity. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity

[7] S. Team, «Did you know that 60% of game developers use game engines?», SlashData. Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://www.slashdata.co/post/did-you-know-that-60-of-game-developers-use-game-engines

[8] U. Technologies, «Unity - Manual: Unity User Manual 2022.3 (LTS)». Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://docs.unity3d.com/Manual/index.html

[9] BillWagner, «Un paseo por C#: información general - C#». Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://learn.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/overview

[10] V. Reports, «Roguelike Game Market Size to Grow USD 57,336 Million in 2030 by 2030 at a CAGR of 12.3% | Valuates Reports». Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://www.prnewswire.com/news-releases/roguelike-game-market-size-to-grow-usd-57-336-million-in-2030-by-2030-at-a-cagr-of-12-3--valuates-reports-302079781.html

[11] BillWagner, «C# identifier names - rules and conventions - C#». Accedido: 15 de mayo de 2024. [En línea]. Disponible en: https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/identifier-names

[12] «BNL | History:The First Video Game?» Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.bnl.gov/about/history/firstvideo.php

[13] H. Labs, *Tennis For Two*. 2011. Accedido: 16 de mayo de 2024. [Photo]. Disponible en: https://www.flickr.com/photos/hslphotosync/5941685811/

[14] Jamal\_Aladdin, «The Evolution of Multiplayer Gaming: A Journey Through Time», Medium. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://medium.com/@Jamal\_Aladdin/the-evolution-of-multiplayer-gaming-a-journey-through-time-e34ef59294c2

[15] «Pong - Videogame by Atari», Museum of the Game. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.arcade-museum.com/Videogame/pong

[16] «Astro Race - Videogame by Taito», Museum of the Game. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.arcade-museum.com/Videogame/astro-race

[17] «Introduction | PDP-1 Restoration Project | Computer History Museum». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.computerhistory.org/pdp-1/introduction/

[18] «Nintendo Entertainment System», Nintendo of Europe AG. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.nintendo.com/es-es/Hardware/La-historia-de-Nintendo/Nintendo-Entertainment-System/Nintendo-Entertainment-System-627024.html

[19] «Dictionary.com | Meanings & Definitions of English Words», Dictionary.com. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://www.dictionary.com/browse/mmo

[20] «Counter-Strike 2 Steam Charts», SteamDB. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://steamdb.info/app/730/charts/

[21] «Most played Multiplayer Games Steam Charts», SteamDB. Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://steamdb.info/charts/?tagid=3859

[22] «Overcooked! 2 en Steam». Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/728880/Overcooked\_2/

[23] «Client-server and peer-to-peer networks - Computer networks and topologies - OCR - GCSE Computer Science Revision - OCR», BBC Bitesize. Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://www.bbc.co.uk/bitesize/guides/zvspfcw/revision/4

[24] «¿Cuáles son las ventajas y desventajas de las arquitecturas peer-to-peer y cliente-servidor?» Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://www.linkedin.com/advice/0/what-advantages-disadvantages-peer-to-peer

[25] «Network topologies | Unity Multiplayer Networking». Accedido: 16 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/current/terms-concepts/network-topologies/

[26] «Dictionary.com | Meanings & Definitions of English Words», Dictionary.com. Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://www.dictionary.com/browse/eSports

[27] «Create a game with a listen server and host architecture | Unity Multiplayer Networking». Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/current/learn/listen-server-host-architecture/

[28] «Distributed authority topologies | Unity Multiplayer Networking». Accedido: 17 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/current/terms-concepts/distributed-authority/

[29] «Understanding ownership and authority | Unity Multiplayer Networking». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/current/basics/ownership/

[30] «Tricks and patterns to deal with latency | Unity Multiplayer Networking». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/current/learn/dealing-with-latency/

[31] «About Netcode for GameObjects | Unity Multiplayer Networking». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://docs-multiplayer.unity3d.com/netcode/1.5.2/about/

[32] «Entities overview | Entities | 1.2.1». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://docs.unity3d.com/Packages/com.unity.entities@1.2/manual/index.html

[33] «Entity concepts | Entities | 1.2.1». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://docs.unity3d.com/Packages/com.unity.entities@1.2/manual/concepts-entities.html

[34] «A Brief History of Mirror | Mirror». Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://mirror-networking.gitbook.io/docs/trivia/a-history-of-mirror

[35] «First Gear Games, a game design company and education resource. Home of Fish-Networking.» Accedido: 19 de mayo de 2024. [En línea]. Disponible en: https://firstgeargames.com/

[36] «¿Qué es la traducción de direcciones de red (NAT)? - Software Check Point», Check Point Software. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://www.checkpoint.com/cyber-hub/network-security/what-is-network-address-translation-nat/

[37] «Relay Solución gratuita de conexión y redes P2P | Unidad», Unity. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://unity.com/products/relay

[38] «On the Historical Origin of the “Roguelike” Term», Slashie’s Journal. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://blog.slashie.net/on-the-historical-origin-of-the-roguelike-term/

[39] «RFD: rec.games.dungeon.\* hierarchy». Accedido: 21 de mayo de 2024. [En línea]. Disponible en: https://groups.google.com/g/news.groups/c/CdWOd-M6g-w/m/cgNn2b9uU2sJ

[40] «Roguelike Games Info and FTP Sites (FAQ)». Accedido: 21 de mayo de 2024. [En línea]. Disponible en: https://groups.google.com/g/rec.games.roguelike.misc/c/o66P9tiyxwI/m/EBIpBDSCzOsJ

[41] Thedarkb, *English: A screenshot of the BSD version of Rogue.* 2021. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://commons.wikimedia.org/wiki/File:Rogue\_Screenshot.png

[42] Z. Rogue, «What “Roguelike” Meant», Medium. Accedido: 20 de mayo de 2024. [En línea]. Disponible en: https://zenorogue.medium.com/what-roguelike-meant-fb8b0e1601a

[43] «What a roguelike is - RogueBasin». Accedido: 21 de mayo de 2024. [En línea]. Disponible en: https://roguebasin.com/index.php/What\_a\_roguelike\_is

[44] «Roguelike - Fanlore». Accedido: 21 de mayo de 2024. [En línea]. Disponible en: https://fanlore.org/wiki/Roguelike

[45] «Enter the Gungeon», Enter the Gungeon. Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://enterthegungeon.com/

[46] «Ahorra un 70% en Enter the Gungeon en Steam». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/311690/Enter\_the\_Gungeon/

[47] «Enter the Gungeon». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.metacritic.com/game/enter-the-gungeon/

[48] R. Cowley y Editor, «Enter the Gungeon shoots past three million units sold», pocketgamer.biz. Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.pocketgamer.biz/news/72336/enter-the-gungeon-3-million-units/

[49] «The Binding of Isaac en Steam». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/113200/The\_Binding\_of\_Isaac/

[50] «The Binding of Isaac: Rebirth Price history», SteamDB. Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://steamdb.info/app/250900/charts/

[51] M. Media, «About The Creator», Maestro Media. Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://maestromedia.com/pages/about-the-creator

[52] «The Binding of Isaac: Afterbirth+ en Steam». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/570660/The\_Binding\_of\_Isaac\_Afterbirth/

[53] «Nuclear Throne Steam stats | Gamalytic». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://gamalytic.com/game/242680

[54] «Nuclear Throne en Steam». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/242680/Nuclear\_Throne/

[55] «Nuclear Throne». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.metacritic.com/game/nuclear-throne/

[56] «Vlambeer». Accedido: 22 de mayo de 2024. [En línea]. Disponible en: https://www.vlambeer.com/

[57] okky, «History of Unity Game Engine», Agate. Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://agate.id/history-of-unity-game-engine/

[58] J. Drake, «24 Great Games That Use The Unity Game Engine», TheGamer. Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://www.thegamer.com/unity-game-engine-great-games/

[59] «Beat Saber en Steam». Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://store.steampowered.com/app/620980/Beat\_Saber/

[60] G. M. Y. D. with a keen interest in G. Development y V. Reality!, «Unreal Engine and its Evolution | Extern Labs Inc». Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://externlabs.com/blogs/unreal-engine-and-its-evolution/

[61] «Unreal Engine potencia la producción de cine y televisión», Unreal Engine. Accedido: 23 de mayo de 2024. [En línea]. Disponible en: https://www.unrealengine.com/es-ES/uses/film-television

[62] G. Engine, «A decade in retrospective and future», Godot Engine. Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://godotengine.org/article/retrospective-and-future/

[63] «Google Trends», Google Trends. Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://trends.google.es/trends/explore?date=2014-01-01%202024-05-24&q=%2Fm%2F0\_x6l2s&hl=es

[64] «godotengine/godot: Godot Engine – Multi-platform 2D and 3D game engine». Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://github.com/godotengine/godot

[65] «Introduction to Blueprints». Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://dev.epicgames.com/documentation/en-us/unreal-engine/introduction-to-blueprints-visual-scripting-in-unreal-engine

[66] «GDScript reference», Godot Engine documentation. Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://docs.godotengine.org/es/4.x/tutorials/scripting/gdscript/tutorials/scripting/gdscript/gdscript\_basics.html

[67] «Unreal Engine 5», Unreal Engine. Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://www.unrealengine.com/es-ES/unreal-engine-5

[68] «Preguntas Frecuentes», Godot Engine documentation. Accedido: 24 de mayo de 2024. [En línea]. Disponible en: https://docs.godotengine.org/es/4.x/about/about/faq.html

DEVELOPMENT OF A ROGUELIKE MULTIPLAYER VIDEOGAME