Escuela Politécnica Superior de Jaén
Grado en Ingeniería Informática

Universidad de Jaén
Escuela Politécnica Superior
de Jaén

TRABAJO FIN DE GRADO

PROTOTIPO DE PLATAFORMA PARA JUEGOS DE CARTAS TRADICIONALES

Alumno

Sergio Martos Pariente

Tutores

Juan Roberto Jiménez Pérez
(Departamento de Informática)

Pedro Jose Sanchez Sanchez
(Departamento de Informática)

Feberero, 2024

(Página intencionalmente en blanco)

Universidad de Jaén

Departamento de Informática

Don Juan Roberto Jiménez Pérez y Don Pedro Jose Sanchez Sanchez, tutores del Trabajo Fin de Grado titulado: '**Prototipo de plataforma para juegos de cartas tradicionales**', que presenta Don Sergio Martos Pariente, otorgan el visto bueno para su entrega y defensa en la Escuela Politécnica Superior de Jaén.

Jaén, Febrero de 2024

El alumno: Los tutores:

JIMENEZ PEREZ JUAN ROBERTO - 44259364G
Firmado digitalmente por JIMENEZ PEREZ JUAN ROBERTO - 44259364G
Fecha: 2024.02.20 12:17:46 +01'00'

SANCHEZ SANCHEZ PEDRO JOSE - 26007525E
Firmado digitalmente por SANCHEZ SANCHEZ PEDRO JOSE - 26007525E
Fecha: 2024.02.20 12:27:52 +01'00'

Sergio Martos Pariente

Juan Roberto Jiménez Pérez

Pedro Jose Sanchez Sanchez

(Página intencionalmente en blanco)

Agradecimientos

En primer lugar a Roberto, mi tutor, por las facilidades ofrecidas desde el primer momento y por sus aportaciones que han contribuido al trabajo.

A Pedro, cotutor del trabajo, por las ayudas recibidas y las dudas resueltas para desarrollar una plataforma de agentes consistente y estable.

También a aquellos profesores que han influido en mi formación, desde que comencé en el mundo de la informática con el ciclo superior de sistemas informáticos en red.

A mi padre y mi madre que me han facilitado y dado la oportunidad de poder dedicarme a lo que quiero haciendo sus propios sacrificios.

Por ultimo a todas las personas, tanto familiares como amigos, personas que comenzaron el camino conmigo y no lo han podido acabar, todas ellas forman parte de lo que he conseguido y siempre quedaran reflejadas junto a mis logros.

FICHA DEL TRABAJO FIN DE TÍTULO	
Titulación	Grado en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	Tecnologías de la Información
Mención (solo TFG)	Sistemas Gráficos
Idioma	Español
Tipo	Específico
TFT en equipo	No
Autor/a	Sergio Martos Pariente
Fecha de asignación	02/11/2022
Descripción corta	En este Trabajo de Fin de Grado se ha llevado a cabo el desarrollo de un prototipo de plataforma para juegos de cartas, dicha plataforma estará destinada a albergar múltiples juegos de cartas, los cuales se pueden jugar tanto de forma online como de forma offline.

NORMAS APLICADAS EN ESTE DOCUMENTO	
LOCALES	
TFT-UJA:2017	Normativa de Trabajos Fin de Grado, Fin de Máster y otros Trabajos Fin de Título de la Universidad de Jaén (Normativa marco UJA aprobada en Consejo de Gobierno)
TFT-EPSJ:2017	Normativa sobre Trabajos Fin de Grado y Fin de Máster en la Escuela Politécnica Superior de Jaén (Normativa EPSJ aprobada en Junta de Escuela)
TFT-EPSJ	Criterios de evaluación y normas de estilo para TFG y TFM de la Escuela Politécnica Superior de Jaén
NACIONALES E INTERNACIONALES	
ISO 2145:1978	Documentación - Numeración de divisiones y subdivisiones en documentos escritos
UNE 50132:1994	Traducción de la ISO 2145
APA 6ª edición	Estilo de referencias y citas de APA (American Psychological Association)

NORMAS UTILIZADAS COMO BASE O REFERENCIA	
NACIONALES	
UNE 157001:2014	Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico
UNE 157801:2007	Criterios generales para la elaboración de proyectos de sistemas de información
*Estas normas se han utilizado **como base o referencia** para la inclusión de algunos contenidos y definiciones sobre elaboración de proyectos, entendiendo como **proyecto** la documentación consensuada entre una empresa y un cliente, que da lugar al perfeccionamiento de un contrato para la elaboración de una obra o la prestación de un servicio. Por consiguiente, no debe esperarse la aplicación de estas normas en cuanto a la completitud de los contenidos ni a la organización de los mismos.*	

Contenido

Índice de ilustraciones

Índice de tablas

1 ESPECIFICACIÓN DEL TRABAJO

En este capítulo se presenta la especificación del trabajo, con una estructura y contenidos **inspirados** en los criterios y recomendaciones que establece la norma UNE 157801:2007 - "*Criterios Generales para la elaboración de proyectos de Sistemas de Información*".

A lo largo del documento se utilizarán términos y acrónimos cuya descripción aparecen en el apartado 7 (Definiciones y abreviaturas).

1.1 Introducción

Los videojuegos son una forma de entretenimiento y una plataforma cultural que ha transformado la sociedad. Además de proporcionar diversión, fomentar habilidades cognitivas, sociales y narrativas, su influencia se extiende a la educación, terapia y comunidades en línea. Los videojuegos son una manifestación artística que refleja la creatividad y la diversión de hoy en día.

La industria del desarrollo de videojuegos ha sufrido una recesión en los últimos años, causada por los retrasos y cuellos de botella causados por la pandemia de Covid y por el impacto de la inflación y los tipos de interés en la economía de los consumidores (Desarrollo Español de Videojuegos, 2022, pág. 14). Aun así, en dicha industria se sigue facturando una cantidad considerable de dinero, en el pasado año 2022 fueron 184.400 millones de dólares, de hecho, se estima los videojuegos vuelvan a regresar a la senda del crecimiento y se estima una facturación de 211.000 millones de dólares para el año 2025 (Desarrollo Español de Videojuegos, 2022, pág. 14).

En lo que se refiere a España, en el año 2021 se facturo un total de 1.218 millones de euros, con un total de 455 empresas dedicadas al ámbito del desarrollo de videojuegos, lo que se traduce en que de media cada empresa facturo un total de 3 millones de euros (Desarrollo Español de Videojuegos, 2022, pág. 22). No podemos dudar de que no es una buena cifra, la cual coloca a España quinta en el ranking europeo y decimotercera del mundo, pero se encuentra muy lejos de los principales países como Estados Unidos o Finlandia.

El empleo generado en España en el año 2021 por parte de la industria de los videojuegos es de 8.833 personas en empleos directos, 1.634 profesionales

freelancers y 4.183 empleos indirectos, lo cual hizo que el desarrollo de videojuegos generara un total de 14.646 puestos de trabajo en este año (Desarrollo Español de Videojuegos, 2022, pág. 27). Si hablamos solamente de los puestos directos de trabajo, podemos afirmar que tuvo un incremento del 10% respecto al año anterior a pesar de los efectos recesivos que se comentaron anteriormente. En la ilustración 1.1 podemos observar la previsión de empleo que se espera con las mejoras fiscales y ayudas que reclama el sector, se puede vaticinar que en 2025 habrá alrededor de 14.000 empleos directos en el desarrollo de videojuegos.

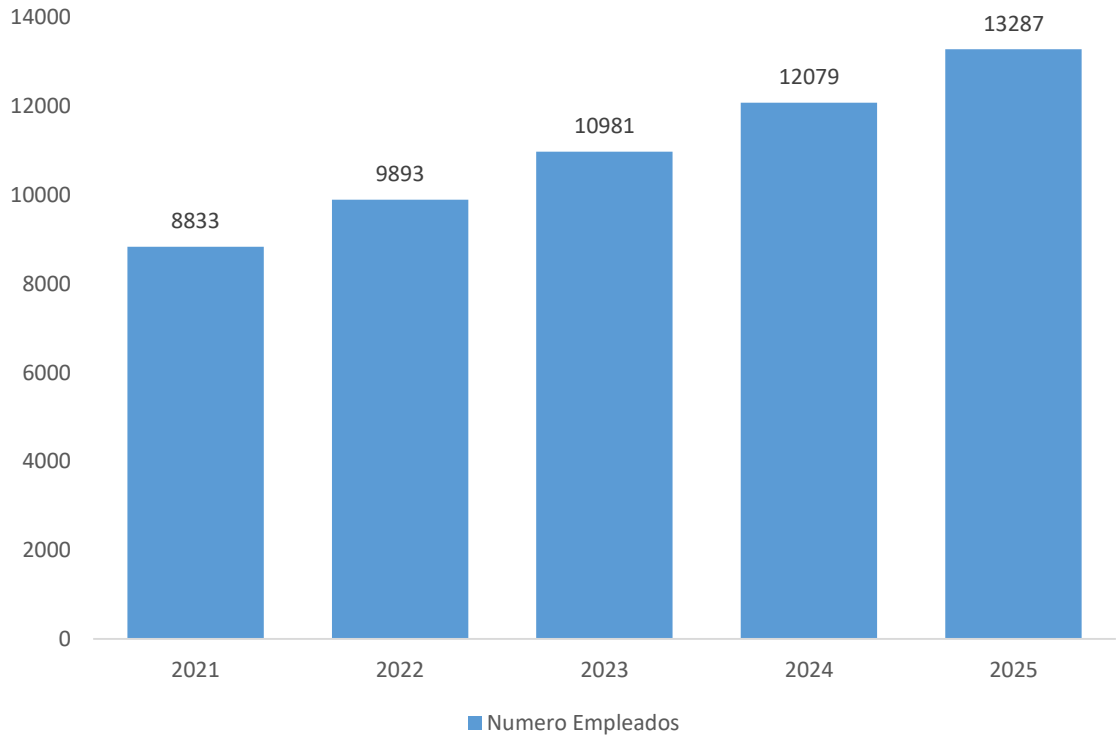

Ilustración 1.1 Evolución esperada del empleo en la industria de videojuegos

Si hablamos de la tipología de actividades, plataformas, tiendas y herramientas, el desarrollo de videojuegos es la principal actividad de los estudios españoles con un 94%, seguido de la autopublicación con un 48% y el desarrollo para terceros con un 41% (Desarrollo Español de Videojuegos, 2022, pág. 32). Esto nos indica que el prototipo que vamos a desarrollar compite contra la mayoría de las empresas, pero también indica que es el camino a seguir para obtener rentabilidad para el juego, ya que es a lo que mayormente dedican su actividad las mismas.

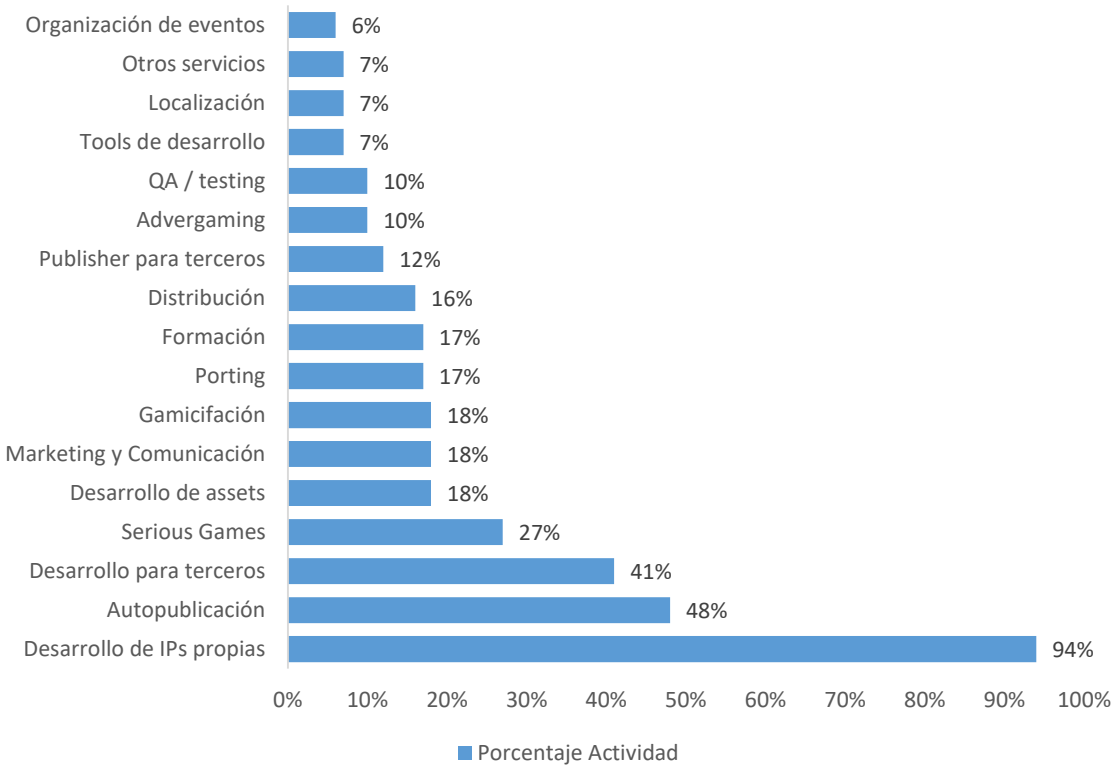

Ilustración 1.2 Tipología de actividades de las empresas

En cuanto al contexto actual de la industria se ha observado que los videojuegos más consumidos son los Free To Play (F2P), por lo que cada vez es más usual introducir publicidad en los juegos para poder sustentar la financiación del mismo. Otro aspecto a tener en cuenta es que cada vez es más común desarrollar videojuegos en los que no existe una diferenciación tan clara entre las plataformas en las que juega cada persona, el juego cruzado será un aspecto que marcará tendencia en los próximos años (Desarrollo Español de Videojuegos, 2022, págs. 20,21).

En este trabajo de fin de grado se ha llevado a cabo el desarrollo de un prototipo de juegos de cartas tradicionales, en el que se pretende aprovechar la tecnología actual y el avance en los motores de videojuegos para desarrollar una plataforma que permita jugar a juegos de cartas tradicionales en línea y promover tanto su difusión como su recuerdo, de esta forma las futuras generaciones aprenderán sobre este tipo de juego y a las generaciones de mayor edad les servirá como una nueva forma de entretenimiento.

En esta memoria se detalla el proceso para el desarrollo del prototipo de videojuego de cartas tradicionales.

1.2 Objetivos del trabajo

El objetivo principal de este proyecto es desarrollar un prototipo funcional de un juego de cartas para promover la difusión de los mismos, así como la validación del videojuego con la implementación de los juegos de cartas.

Para garantizar el correcto desarrollo y alcance del proyecto se ha de definir una serie de objetivos. A continuación, se mostrarán los objetivos generales:

- Desarrollar una plataforma de juegos de cartas en linea.
- Desarrollar un juego de cartas tradicional para validar la plataforma.

Tras definir los objetivos generales, surgen una serie de objetivos secundarios que derivan directamente de los anteriores, estos se detallan en la siguiente lista:

- Elaborar un estudio sobre los motores de videojuegos disponibles para su posterior eleccion para el desarrollo.
- Analizar y diseñar la interfaz de interacción con el usuario.
- Modelar animaciones para la interaccion con las cartas.
- Definir las mecanicas del juego.
- Seleccionar las herramientas complementarias requeridas en el desarrollo que mas se ajusten a las necesidades del proyecto.
- Implementación de elementos visuales y de sonidos para el juego de cartas.
- Elaborar un estudio sobre las posibles plataformas que proporcionan servicios de juegos en linea disponibles para su posterior eleccion para el desarrollo.
- Implementar un sistema de Inteligencia Artificial para que los jugadores puedan jugar a los juegos de cartas sin necesidad de que el juego sea en linea.

1.3 Antecedentes y estado del arte

El prototipo de videojuegos de cartas se ha visto inspirado en otros videojuegos para la realización de ciertos elementos gráficos.

Al querer “remasterizar” los juegos de cartas y querer darle una imagen nueva a este tipo de juegos, se ha considerado que la opción de 3D era esencial para que resulte más atractiva a los jugadores, por ello hemos basado el posicionamiento de las cartas en el videojuego UNO Mobile (UNO, s.f.), ya que a día de hoy es uno de los juegos de cartas más utilizado y es por pequeños detalles como este. La ilustración 1.3 muestra una forma de colocar las cartas parecida a como se hace en el mundo real lo que ayuda a obtener una mayor aceptación.

Ilustración 1.3 Juego de cartas UNO Mobile

Otro videojuego que queremos tomar como referencia en nuestra interfaz de usuario es Brawl Stars (Supercell, s.f.), la característica que comparte este juego con el nuestro es que dentro del mismo juego te permite jugar a distintos modos de juego, lo cual encaja perfectamente con la idea de nuestro prototipo de poder jugar distintos juegos de cartas, ya puede ser Cinquillo, Tute… etc. Por lo tanto, se puede asemejar en la forma que Brawl Starts utiliza para informar al usuario sobre qué modo de juego va a jugar, en la ilustración 1.4 se puede ver como indica el modo de juego

seleccionado, que en este caso es Gem Grab y el usuario ya sabe que jugara a este modo de juego cuando presione el botón de jugar.

Ilustración 1.4 Captura del juego Brawl Starts

1.4 Descripción de la situación de partida

En este apartado describiremos las condiciones iniciales del trabajo de fin de grado, para ello estudiaremos las propias condiciones de desarrollo del trabajo y como se verá afectado.

1.4.1 Descripción del entorno actual

En lo que se refiere al entorno actual podemos divisar distintos ámbitos, los cuales serán tratados de forma independiente.

1.4.1.1 Entorno social

Lo primero que debemos de hacer es realizar un estudio sobre la cantidad de personas que juegan a videojuegos para saber el impacto que tendrá el mismo y a cuanta gente podemos llegar.

La cantidad de “videojugadores” que existen en España representan un alto porcentaje de la población, por lo que a la hora de desarrollar un videojuego tenemos una gran cantidad de clientes potenciales. En nuestro país se estima que juegan a videojuegos 18.2 millones de españoles, repartidos de forma muy equitativa entre hombres y mujeres, donde los hombres representan el 53% de videojugadores

mientras que las mujeres representan el 47% del total de videojugadores (Asociación Española de Videojuegos, 2022, pág. 17). Estos se encuentran repartidos en diferentes rangos de edad, como muestra la ilustración 1.5. En el caso del juego de cartas, se puede afirmar que gran parte del público español lo jugaría, sobre todo a los mayores de 25, ya que normalmente jugar a las cartas es un hobby para las personas de estos rangos de edad. Además, al hacer el juego con gráficos actualizados también permite acceder a publico menos común, como podrían ser los menores de 25 años (Asociación Española de Videojuegos, 2022, pág. 24)

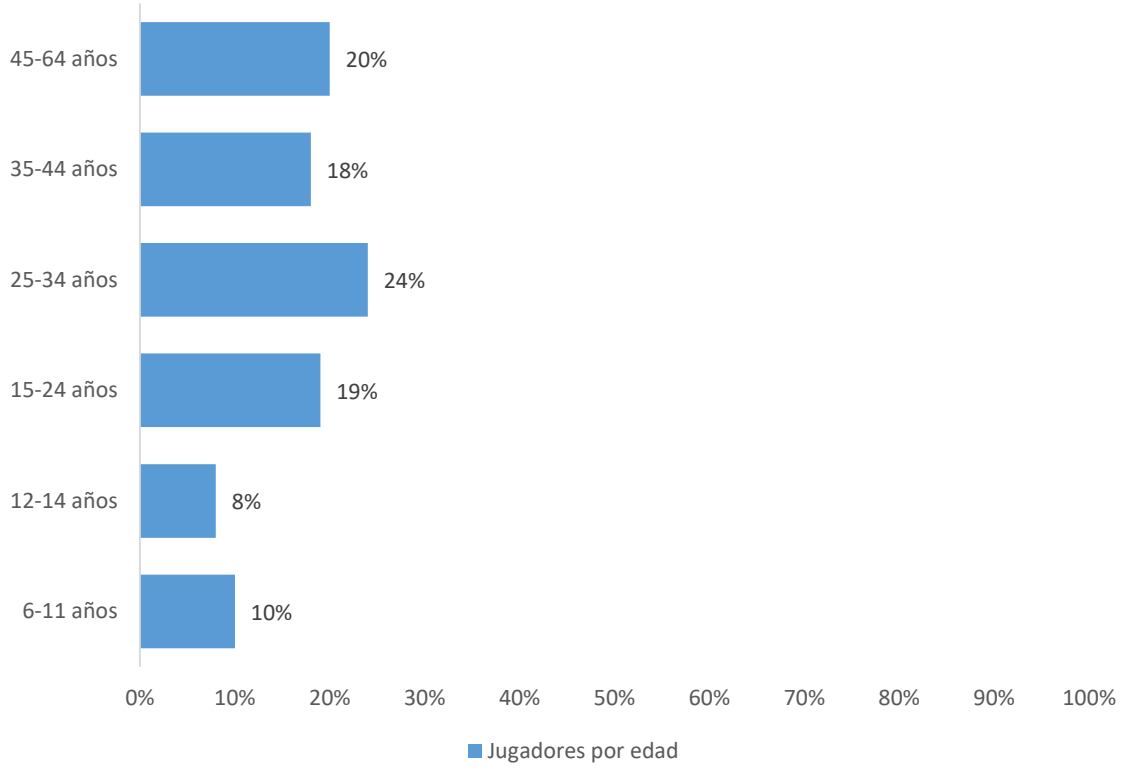

Ilustración 1.5 Porcentaje de videojugadores por rango de edad

En cuanto a las horas de uso de uso de los videojuegos se han obtenido los resultados de la ilustración 1.6. Estos resultados muestran que los videojuegos son un hábito importante en el día a día de las personas

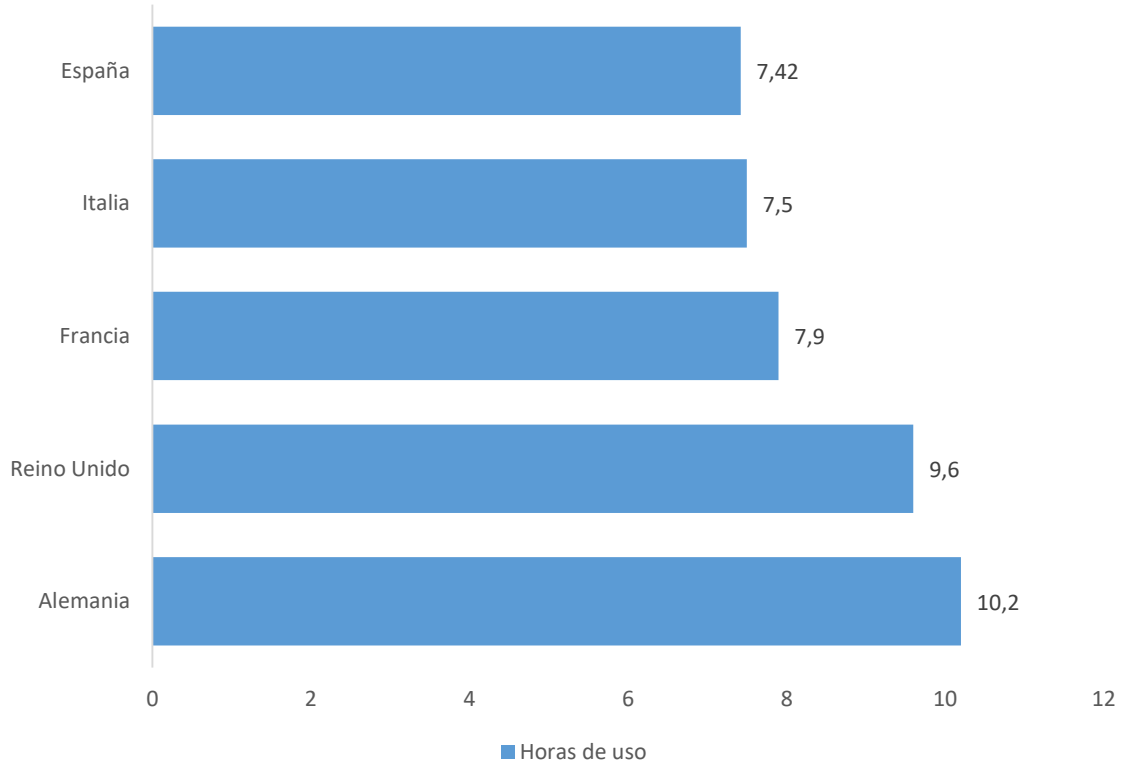

Ilustración 1.6 Horas de uso semanales de videojuegos

Si comparamos estos datos actuales con respecto a los de otros años, los españoles han bajado la media respecto al año 2021 en un 8% (Asociación Española de Videojuegos, 2022, pág. 17).

Otro elemento a tener en cuenta, es el formato que los españoles prefieren utilizar a la hora de jugar a los videojuegos, a continuación, en la ilustración 1.7 se muestra los porcentajes de la facturación total durante el año 2022 que han ido destinados al formato digital y al formato físico.

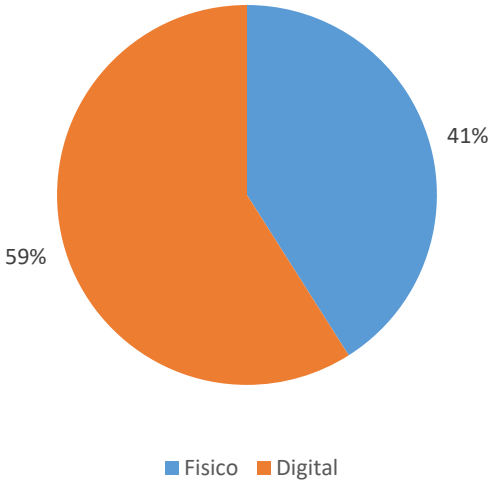

Ilustración 1.7 Porcentaje de facturación de los formatos de los videojuegos

Además, también se ha estudiado la comparación respecto al año pasado y el formato digital está haciéndose cada vez más importante, ya que ha crecido un 30% respecto al año 2021 (Asociación Española de Videojuegos, 2022, pág. 18).

1.4.1.2 Entorno de la empresa

En lo que se refiere a la forma de encajar el desarrollo del software actual dentro de la empresa, al no disponer de ningún proyecto en funcionamiento, no tendrá ningún impacto en lo que a métricas y configuraciones se refiere, esto ocurre al no tener implementado ningún elemento interno de la empresa que suministre datos a la aplicación, ni tampoco disponer de servidores para almacenar la información de los usuarios.

1.4.2 Resumen de las deficiencias y carencias identificadas

Realizando un estudio sobre los distintos juegos de cartas que están disponibles actualmente se han identificado las siguientes carencias y deficiencias.

1.4.2.1 Exceso de fragmentación

Si hablamos de dispositivos móviles, no ha sido posible encontrar ninguna app que englobe todos estos juegos de cartas, se han buscado las compañías con más repercusión y reseñas positivas dentro de la Play Store de Google y de entre todas las compañías destacan dos, que son DonNaipe y Quarzo Apps (Google Play Store, s.f.), como podemos observar en la ilustración 1.8, estas compañías ofrecen varios juegos de cartas pero separados en distintas apps, esto tiene sus ventajas en cuanto almacenamiento, pero a la hora de la experiencia del jugador deja mucho que desear. Esta fragmentación obliga a una gran cantidad de usuarios a utilizar varias aplicaciones si desea jugar a varios juegos de cartas, sin embargo si todos los juegos estuvieran en una misma aplicación se optimizaría el almacenamiento, ya que solamente se necesitaría una app, además también se ahorraría tiempo y batería al no tener que estar constantemente cerrando y abriendo aplicaciones. Por lo que hacer una aplicación con todos los juegos unificados podría resolver esta necesidad.

Ilustración 1.8 Suite de juegos ofrecidos por las compañías DonNaipe y QuarzoApps

1.4.2.2 Falta de especificación de las plataformas

En este apartado se hablara sobre las plataformas que incluyen una gran cantidad de juegos, en dichas aplicaciones no solo se limitan a juegos de cartas, ya que también te dan las opciones de juegos como ajedrez, damas, parchís, incluso juegos que no tienen nada que ver con juegos de mesa. Esto no significa que sea una mala clasificación de los tipos de juegos, ya que cada sitio organiza sus juegos de la forma más conveniente para la empresa, Lo que ocurre es que esto puede hacer huir a gran cantidad de personas que solamente buscan jugar a las cartas y no tienen una plataforma específica para ello.

Para las personas que solo quieren jugar a juegos de cartas, sin tener en cuenta el resto, pueden llegar a confundirse a la hora de escoger el juego de cartas deseado, dado la gran cantidad de juegos disponibles. En la ilustración 1.9 se muestra la cantidad de juegos que tiene la página de minijuegos solamente mostrando los juegos de mesa y similares (Minijuegos, s.f.).

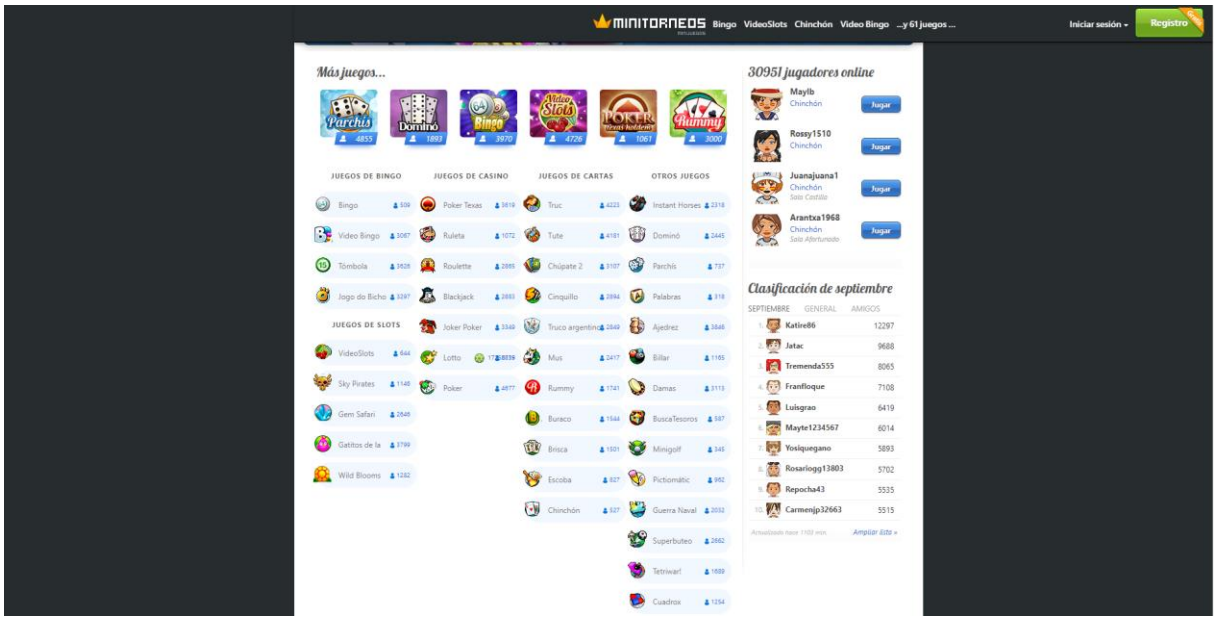

Ilustración 1.9 Selección de juego en la plataforma de minijuegos

1.4.2.3 Gráficos desfasados

Después de probar varios videojuegos de cartas tradicionales, se ha llegado a la conclusión de que todos tienen un factor común, los juegos de cartas se muestran en vista cenital, simulando un 2D. Aunque haya videojuegos que estén implementados en 3D, al usuario solamente se le muestra visión desde este 2D. Esta vista es simple

y muchos jugadores están acostumbrada a ella y no tienen problema a usarla, pero se ha visto a través de juegos como el UNO, que es posible darle un aire nuevo a la forma de jugar a las cartas, si se implementa la opción de dejar al usuario seleccionar su tipo de vista se podrán acaparar más clientes en un futuro y de esta forma que más personas puedan disfrutar de los juegos de cartas. Como podemos observar en la ilustración 1.10 esta es de las mejores interfaces graficas que nos hemos encontrado en un juego de cartas de naipes, y aun así da la sensación de juego antiguo (Ludoteka, s.f.).

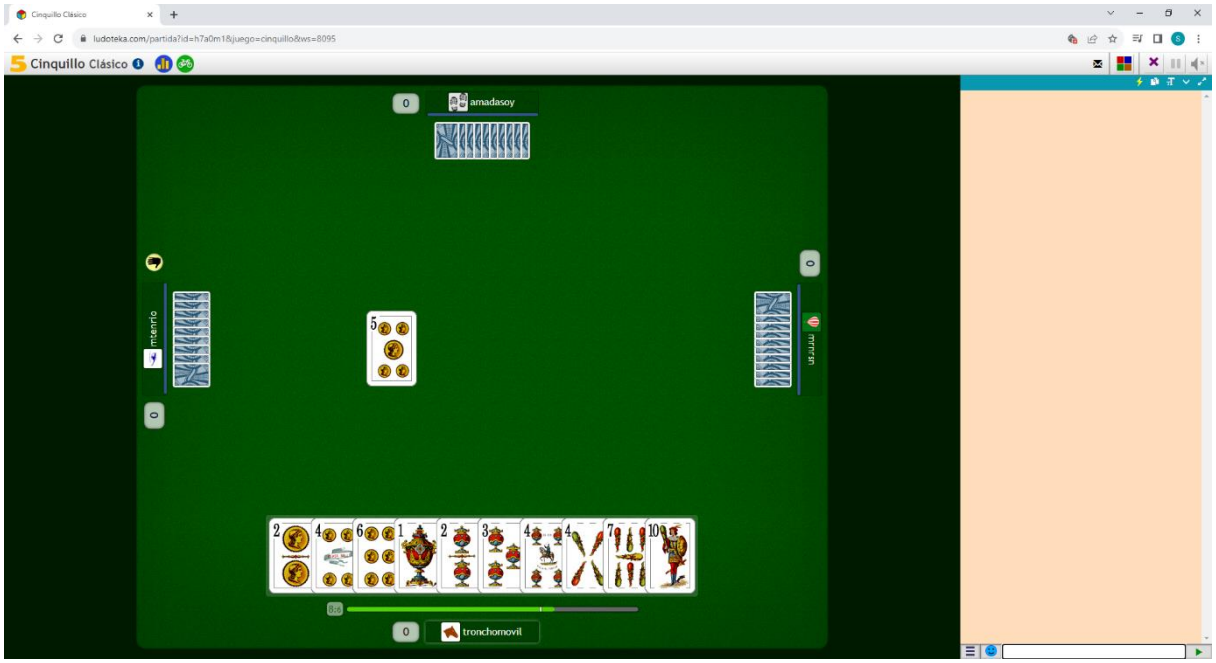

Ilustración 1.10 Plataforma de juegos de mesa Ludoteka

1.5 Requisitos iniciales

En este apartado se especificarán los requisitos principales a alto nivel, para el prototipo de juego de cartas se pueden identificar los siguientes requisitos:

- El usuario debe entender en que modo de juego se encuentra, por lo que antes de iniciar una partida el usuario debe saber el tipo de partida (En linea, local) y el juego al que va a jugar (Cinquillo, Tute…etc).
- El usuario debe poder jugar partidas contra la Central Proccessing Unit (CPU).
- El usuario debe poder jugar partidas en linea contra otras personas del resto del mundo.

- Cuando el usuario se encuentre en una partida debe saber en todo momento el turno del jugador que le corresponde realizar un movimiento.
- El usuario debe tener la opcion de volver a seleccionar partida y jugar tantas partidas como el desee despues de finalizar la partida en la que se encuentra.
- El usuario debe comprender correctamente las mecanicas del juego por lo que estas deben ser lo mas parecidas a las metaforas de la vida real.
- El usuario debería poder modificar su informacion en el perfil de usuario.
- El usuario debería tener una introduccion para que entienda correctamente el funcionamiento de la aplicacion.
- El usuario debería poder cambiar ciertos elementos de la interfaz, como el idioma o el tipo de carta con la que jugar la partida.

1.6 Alcance

Para definir hasta dónde llega el trabajo, se expondrá un marco con el que se delimitara claramente todo lo que se incluye entre los resultados. Por lo cual se mostrará una enumeración con los entregables del trabajo con su contenido y características.

- Entregable con los objetivos del proyecto: En el se informara sobre los elementos que se piensan incluir en el software a realizar.
- Entregable con las mecanicas del jugador: En el se entregara un prototipo en donde se puede arrastrar las cartas, y seleccionarlas, para que el usuario vea como seran los movimientos de la mano del jugador
- Entregable con la interfaz de usuario: En este entregable se incluira la propia interfaz de usuario, en la que se experimentara si la colocacion de los botones es la idonea, si es lo suficientemente clara para que el usuario entienda lo que esta haciendo y si se ajusta a las necesidades del propio usuario.
- Entregable final: Este sera el ultimo entregable y contendra un prototipo demostrativo de una plataforma que puede utilizar varios juegos de cartas,

tanto en red como contra el propio ordenador en el que se crean varios agentes que son capaces de jugar de manera autónoma.

1.7 Hipótesis y restricciones

Para este apartado se enumerarán las hipótesis y restricciones que tiene nuestro proyecto a la hora de aplicar los objetivos del proyecto

1.7.1 Restricciones

A continuación, se mostrarán las restricciones a la que se enfrenta nuestro prototipo de juego de cartas.

- La duracion maxima de dedicacion al es de 300 horas, ya que el TFG es una asignatura de 12 creditos y es la duracuion que se le debe dedicar a este trabajo.
- La duracion maxima para la entrega del proyecto es de 3 meses, ya que es el tiempo disponible para entregar el TFG.
- Los entregables deben estar dispnibles un dia antes de su exposicion.
- El presupuesto no puede excederse el presupuesto previsto para este trabajo, ya que se ha contemplado esta estimacion y no podemos sobrepasarnos de la misma.
- La calidad del trabajo debe dejar satisfecho al 70% de personas que lo utilicen, si no es asi se debe considerar de nuevo los factores implementados en la entrega.

1.8 Estudio de alternativas y viabilidad

En este apartado se explicaran las alternativas que se han tenido en cuenta para desarrollar el proyecto y por qué se ha decidido escoger una determinada opción entre todas las alternativas posibles.

1.8.1 Elección del motor de videojuegos

En el mercado actualmente se encuentran una gran cantidad de motores de videojuegos, los aspectos más importantes a la hora de escoger un motor de

videojuegos dependen en gran medida de la experiencia y de las preferencias personales de los desarrolladores. Aun teniendo estos factores en cuenta, existen una gran cantidad de criterios para seleccionar el motor, algunos criterios generales podrían ser los siguientes (Superprof, s.f.):

- El desarrollo del progama: La eleccion de un motor de juego suele ser un compromiso de varios meses o incluso años de desarrollo. Por eso es importate que el motor que elijamos se adapte a las necesidades y objetivos, que sea estable, actualizado y compatible con las plataformas deseadas.
- El aprendizaje: Algunos motores de videojuegos son mas faciles de usar que otros, esta “facilidad” depende de factores como documentación, lenguaje de programacion utilizado y la comunidad que el motor de videojuegos tenga para proporcionar informacion sobre los posibles errores.
- La reputación: La popularidad y el prestigio pueden influir notablemente en la calidad del proyecto a realizar, ya que a partir de esta podemos hacernos la idea de si el motor esta o no preparado para las necesidades del videojuego, como por ejemplo a traves de reseñas o viendo problemas que previamente han experimentado otros desarrolladores con su videojuego.
- Las caracteristicas: Cada motor de videojuegos tiene sus propias caracteristicas, como ofrecer soporte exclusivo para juegos 2D, o soporte tanto para 2D como 3D, sistema de fisicas utilizado, costes del motor de videojuego, etcétera (etc). Es importante conocer estos aspectos del motor para escoger el que mas se ajuste al proyeto.

Una vez expuestos los principales criterios a la hora de seleccionar un motor de videojuegos veremos mas a fondo las caracteristicas de los principales motores de videojuegos.

1.8.1.1 Unity

Unity es una plataforma de desarrollo en tiempo real, la cual nos permite crear y hacer crecer juegos, aplicaciones y experiencias 3D en tiempo real, que pueden ser utilizadas para entrenamiento, cine, automoción, arquitectura y más funcionalidades (Unity, s.f.). Con Unity podemos desarrollar tantos proyectos en 3D como en 2D por lo que nos ofrece una gran adaptabilidad a nuestro tipo de proyecto.

Uno de los puntos fuertes de Unity es su interfaz gráfica, ya que posee una interfaz intuitiva, lo que hace que sea fácil de usar y permite a los desarrolladores centrarse en la creación del juego en vez de perder el tiempo intentando comprender como funciona el motor.

Este motor de videojuegos cuenta también con una documentación completa y bien explicada (Unity | Manual, s.f.), con buen soporte técnico que ayudan a los desarrolladores a resolver sus problemas y poder continuar con la realización del proyecto. Si esto fuera poco cuenta con una gran comunidad de desarrolladores para ayudarse entre sí, teniendo disponible un foro (Unity | Forum, s.f.) y paginas como Unity Answers (Unity | Disscusions, s.f.)

Se dice que Unity tiene una curva de aprendizaje bastante baja, ya que, con poco puedes tener un videojuego bastante completo sin adentrarte mucho en el motor. Por este motivo la mayoría de desarrolladores que están comenzando en el mundo de desarrollo de videojuegos comienzan por Unity.

Otro aspecto a tener en cuenta es que Unity es compatible con múltiples plataformas, esto es otro punto a favor del motor ya que permite a los desarrolladores llegar a una audiencia más amplia. A día de hoy soporta el desarrollo en plataformas móviles, tanto Android como IOS, plataformas de escritorio como Windows, Mac y Linux, plataformas web, como WebGL, todo el catálogo de consolas actuales, incluso para plataformas de realdad virtual extendida (Wikipedia | Unity, s.f.).

En lo que se refiere a los costes de Unity, tiene una gran variedad de planes y precios para ajustarse a las necesidades de los usuarios, entre estos planes podemos diferenciar los siguientes (Unity | Pricing, s.f.):

- Unity Personal: Es la version gratuita de Unity, para la mayoria de desarrolladores es suficiente, el unico “inconveniente” de esta version es que tienes que agregar al inicio de tu proyecto una pantalla con el logo de Unity, la cual no se puede borrar.

- Unity Pro: Es una version avanzada de la version gratuita, la cual tiene un coste de 1.800€ al año, en esta ya no existe la pantalla de carga y tenemos algunas ventajas como servicio al cliente prioritario. Esta tambien orientado a desarrolladoores indie, pero que ya tienen cierta experiencia con los videojuegos.

- Unity Industry: Esta enfocada a la realidad virtual y tiene un coste de 4.500€ al año.
- Unity Enterprise: Esta modalidad esta enfocada a empresas y no dispone de un precio oficial ya que depende del acuerdo que llege la empresa con Unity. Este plan esta enfocado para empresas que se dediquen a los videojuegos.

Algunos juegos desarrollados en Unity son Among Us (Steam | Among Us, s.f.), Cuphead (Cuphead, s.f.), Hollow Knight (Hollow Knight, s.f.), Pokémon Diamante Brillante y Perla Reluciente (Pokemon Diamante Brillante y Perla Reluciente, s.f.)

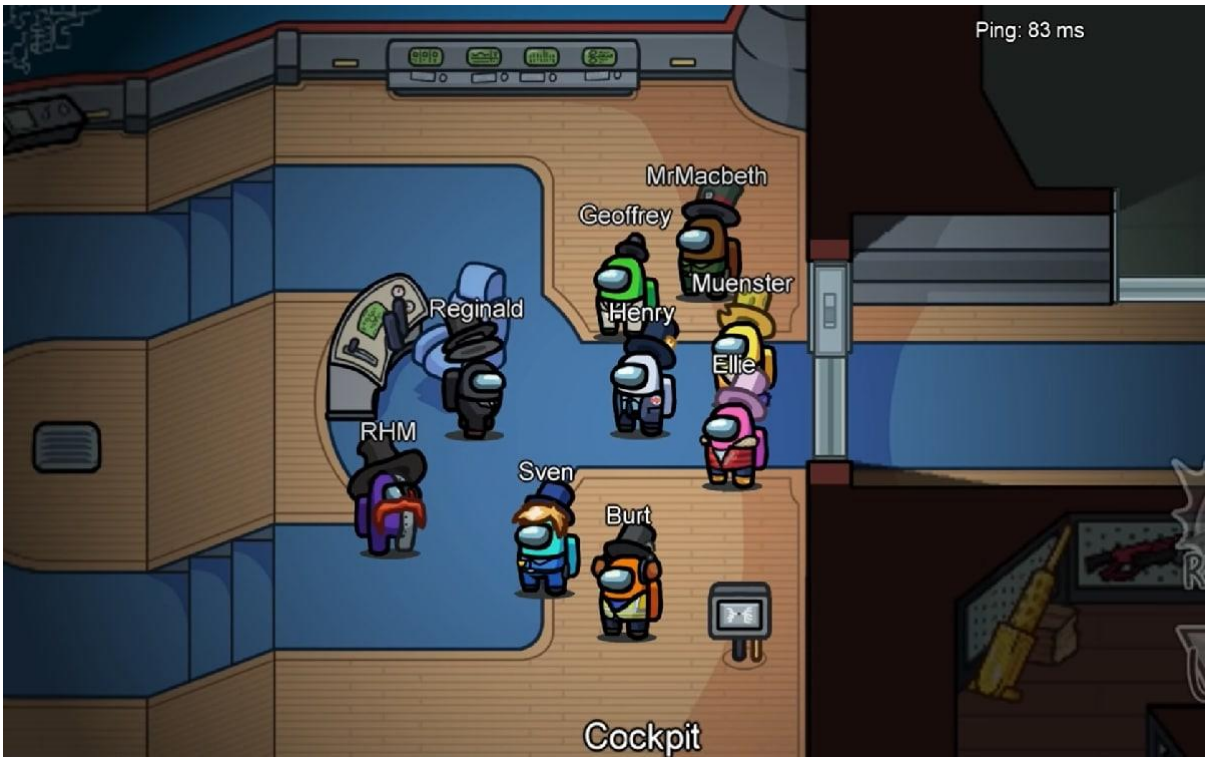

Ilustración 1.11 Captura de videojuego. Among Us

1.8.1.2 Unreal Engine

Unreal Engine es un motor de videojuegos creado por la compañía de Epic Games, su primera publicación fue un shooter en primera persona en el año 1998 y estaba pensando para desarrollar juegos solamente de este tipo, pero con el tiempo ha ido evolucionando para desarrollar cualquier tipo de juego (Wikipedia | Unreal Engine, s.f.). A diferencia de Unity no se presenta una forma nativa de desarrollar

videojuegos en 2D, pero eso no significa que no pueda hacerse tanto juegos 2D como 3D (Softzone, s.f.)

En lo que se refiere a la interfaz gráfica tiene una interfaz gráfica intuitiva lo cual ayuda a los usuarios a que se centren en el desarrollo del videojuego en vez de aprender sobre la interfaz.

Hablando de la documentación, en Unreal Engine no existe tanto nivel de detalle como en Unity, aunque nos muestra ejemplos sobre posibles implementaciones y como llevarlas a cabo (Unreal Engine | Documentation, s.f.). También cuenta con un foro donde se puede debatir y encontrar información sobre soluciones a errores a través de las respuestas de otros desarrolladores (Unreal Engine | Forum, s.f.)

Por toda la comunidad de desarrolladores es sabido que Unreal Engine tiene una curva de aprendizaje bastante elevada, según la comunidad esto en gran parte es causado por las constantes actualizaciones con nuevas funcionalidades, aunque todo depende del tipo de juego que se quiera desarrollar (Developer Comunity Unreal Engine, s.f.)

Si hablamos de plataformas para las cuales podemos programar nuestros juegos Ureal Engine permite la programación en cualquier plataforma que no sea web, es decir permite plataformas como consolas, escritorio y dispositivos móviles (Unreal Engine | Features, s.f.).

Si hablamos de costes Unreal Engine tiene distintos planes para ajustarse a las necesidades de los usuarios (Unreal Engine | Pricing, s.f.):

- Standard License: Es gratuita y esta enfocada a creadores y editores que no requieren soporte premium ni terminos personalizados.
- Enterprise Program: Esta licencia esta pensada para profesionales ajenos a los juegos que buscan soporte premium y terminos de licencia personalizados, el coste es de 1500$ por año.
- Custom license: Esta modalidad esta enfocada para estudios profesionales de desarrollo de juegos que buscan soporte premium, no dispone de tarifa de preciones ya que depende del acuerdo que llegue la empresa con Unreal.

Algunos de los videojuegos desarrollados bajo el motor de Unreal Engine son, Star Wars: Jedi Fallen Order (EA, s.f.), Fortnite (Fortnite, s.f.), Dragon Ball Fighter Z (Bandai, s.f.).

Ilustración 1.12 Captura de videojuego. Dragon Ball Fighter Z

1.8.1.3 Conclusión

A continuación, se abordará un estudio sobre los motores de videojuegos más utilizados en España durante el año 2022 por parte de las empresas, de esta forma podremos hacernos una idea de que motores tienen más prestigio o mejor reputación, ya que las empresas depositan su confianza en ellos.

Según la encuesta DEV (Desarrollo Español de Videojuegos, 2022, pág. 35) el motor de desarrollo de videojuegos más utilizado sigue siendo Unity, al igual que años anteriores, seguido por Unreal Engine. Estos dos son los más utilizados y están muy distanciados de las demás opciones como muestra la ilustración 1.13

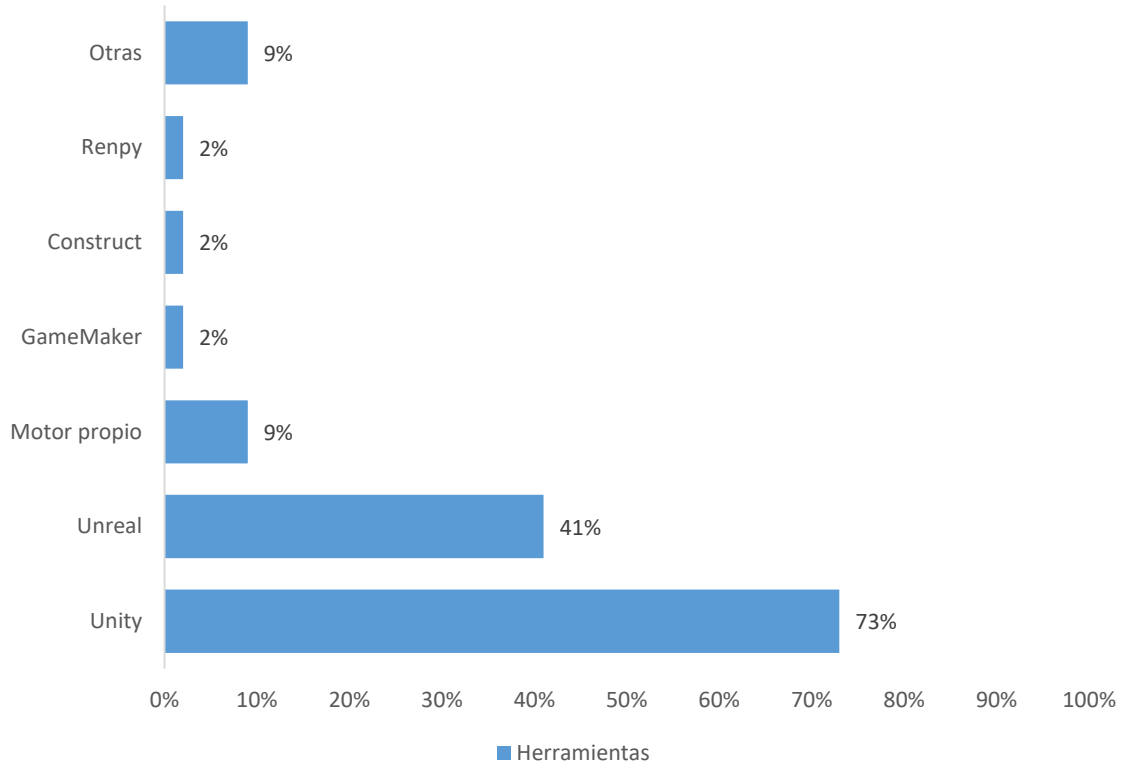

Ilustración 1.13 Motores de videojuegos más utilizados por las compañías españolas en 2022

Por lo tanto, para este proyecto se utilizara el motor de desarrollo de videojuegos de Unity, ya que ha sido una herramienta con una curva de aprendizaje mas suave que la de Unreal Engine, permite el soporte web, que aunque no sea prioritario en este momento, pensando en posibles actualizaciones podría ser una buena idea crear una build del proyecto para web. Como podemos observar en la ilustración 1.13, es el más usado por las empresas españolas lo que nos indica que es una plataforma estable para desarrollar nuestro proyecto. Otra razón de peso para utilizar este motor de videojuegos es que ya se ha trabajado con él en el grado por lo que costara mucho menos esfuerzo crear el juego en esta plataforma que mudarse a otra nueva y aprender todo desde 0.

Al ser el primer proyecto y no tener una estructura de empresa se utilizará la versión gratuita que cubre nuestras necesidades para completar el trabajo de fin de grado.

1.8.2 Topología de los agentes

La decisión anterior sobre el uso de Unity como motor de desarrollo de videojuegos nos abre nuevos elementos entre los cuales debemos tomar una decisión, uno de ellos es escoger el tipo de agentes que vamos a realizar para el desarrollo de la toma de decisiones de lo que denominaremos como inteligencia artificial.

1.8.2.1 Unity Machine Learning Agents Toolkit

Estos agentes que nos proporciona Unity son agentes de aprendizaje por refuerzo, lo cual nos proporciona uno muy buenos resultados de los agentes a largo plazo.

Uno de los problemas de este tipo de agentes es que los modelos de los agentes deben de ser entrenados en Unity, durante un largo periodo de tiempo y es bastante difícil de escalar al no ser que se utilice otro software que ayuden a su entrenamiento (Amazon, s.f.).

Algo que cabe destacar es que para este caso también tenemos una documentación muy extensa y detallada donde podemos poner en marcha este tipo de agentes (Unity, Unity | MLAgents, s.f.).

El uso de este tipo de agentes es bastante interesante, pero realmente no se necesitara esto para el videojuego de cartas, ya que no se busca tener una inteligencia artificial que siempre gane al jugador, se podrían entrenar modelos con distintos niveles de dificultad pero la idea del proyecto no es centrar el esfuerzo en la inteligencia de los agentes que actuaran como adversario.

1.8.2.2 Agentes JADE

JADE es una plataforma de código abierto para aplicaciones basadas en sistemas multiagentes, los cuales cumplen con las especificaciones FIPA (FIPA, s.f.). Se basa en un modelo de composición y ejecución de tareas simples con una comunicación a través del paso de mensajes asíncronos, dispone de un servicio de páginas amartillas que permite mecanismos de suscripción y publicación (JADE, s.f.).

Como vemos la estructura de JADE se ajusta mucho más a lo que buscamos para el desarrollo del proyecto. Ya que no necesitamos de un modelo previo, los agentes reaccionaran a las distintas situaciones que se enfrenten dentro de una partida de cartas, lo cual quita complejidad al sistema que aparentemente es

innecesaria. Además no dejan de ser agentes colaborativos por lo que su inteligencia se puede escalar como el desarrollador desee a lo largo del tiempo.

Otro aspecto a destacar es su documentación, que es extensa y bien cuidada, aparte de tener libros muy interesantes que explican cómo sacar el máximo provecho a este tipo de agentes y el cual usaremos a lo largo del desarrollo (Bellifemine, 2007)

1.8.3 Comunicación con la plataforma de agentes

En nuestro juego es necesario que el motor de videojuegos se comunique de alguna forma con la plataforma de agentes. En este apartado se explicaran los métodos que se han estudiado para llevar a cabo este traspaso de información.

1.8.3.1 Escritura / Lectura de ficheros

La comunicación a través de la escritura y lectura de ficheros tiene las siguientes ventajas

- Persistencia: Los datos se mantienen en el archivo.
- Senzillez: La implementacion de la escritura y lectura de ficheros es relativamente simple.
- Independencia de la red: No se requierede conexión en tiempo real para escribir en los ficheros.

En cuanto a los inconvenientes de este modo de comunicación se pueden destacar los siguientes:

- Latencia: Las lecturas y escrituras se van haciendo mas lentas conforme el tamaño del archivo aumenta.
- Sincronización: Se pueden modificar datos de forma simultanea, por lo que no se garantiza que sean seguros a la concurrencia.

Este metodo fue descartado dado que sus inconvenientes, estos nos pueden llevar a problemas como que se desincronicen los turnos y tiempos de espera para que cada jugador realice sus movimientos, lo cual dificularia bastante el desarrollo.

1.8.3.2 Sockets

En lo que se refiere a la comunicación a través de sockets podemos decir que tiene las siguientes ventajas

- Latencia: La comunicación es instantanea y en tiempo real.
- Flexibilidad: Los sockets pueden implementarse para distintos protocolos de comunicación (TCP, UDP, etc…).
- Eficiencia: La cantidad de datos transferida es mas eficiente que la lectura de ficheros, ya que no es necesario leer ficheros completos para obtener la informacion

Si hablamos de los inconvenientes de los sockets podemos destacar los siguientes:

- Complejidad: Es mas dificil de implementar que la lectura / escritura de un sistema de ficheros.
- Dependencia de la red: Las plataformas deben de estar conectadas para que funcione la comunicaion.

Como se puede observar los sockets estan mas preparados para esta comunicación entre plataformas y es por ello, que escogeremos esta opción para implementar la comunicación entre la plataforma de agentes y el motor de desarrollo de videojuegos.

1.8.4 Elección de la topología para de red para el juego en línea

Antes de seleccionar la plataforma para llevar a cabo los juegos en línea, tenemos que establecer una tipología para la conexión de los distintos dispositivos a la red, para llevar a cabo esta funcionalidad. A continuación se explicaran brevemente las diferentes topologías que han sido estudiadas para llevar a cabo el aspecto del multijugador.

1.8.4.1 Topología LAN

Fueron de las primeras redes utilizadas para el multijugador, en la actualidad están prácticamente en desuso ya que para su correcto funcionamiento es necesario que todos los jugadores estén conectados al mismo router. Esta topología se sigue usando para eventos deportivos como los eSports, ya que su gran ventaja es la baja latencia para transmitir la información.

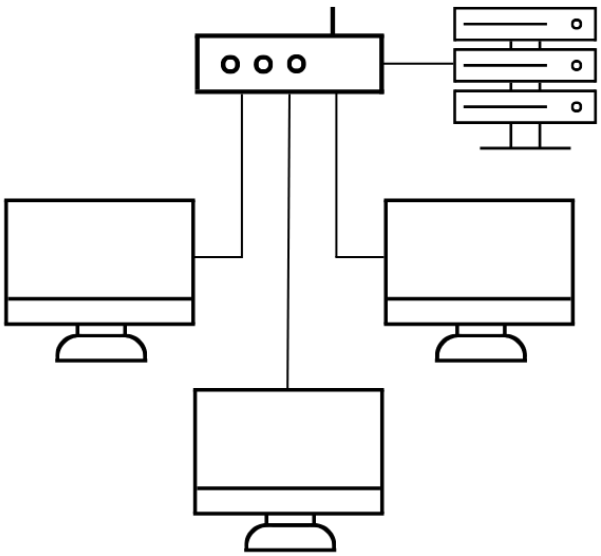

Ilustración 1.14 Topología LAN

1.8.4.2 Topología (Direct) Peer-To-Peer

Esta topología es muy inusual en el desarrollo de videojuegos multijugador, aunque su principal ventaja es que el coste monetario de implementarla, que es nulo, se decidió descartarla por sus grandes inconvenientes.

Para empezar la complejidad para manejar los aspectos multijugador es mucho mayor que el de otras topologías, ya que toda la información se debe mandar muchas veces para que todos los jugadores actualicen sus datos, además la seguridad de las partidas es casi nula, ya que con esta topología se conoce toda los datos de antemano, como el mazo, siguientes movimientos, etc…

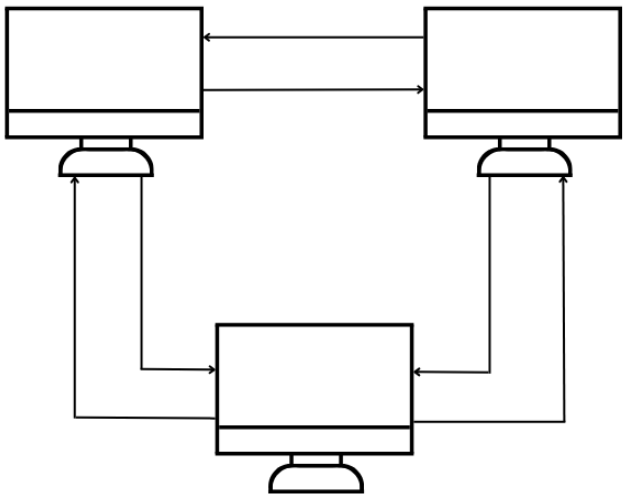

Ilustración 1.15 Topología Direct Peer -To – Peer

1.8.4.3 Topología (Player-Hosted Client/Server) Peer-To-Peer

Esta topología mejora bastante a la anterior, por ello es que si es más común encontrarla en juegos multijugador, en ella solamente tenemos un jugador de toda la partida que actúa como cliente y servidor, el resto son todos clientes. Esta topología mantiene la esencia de la anterior y no necesita servidores para su funcionamiento, lo cual abarata mucho el coste la aplicación.

En cuanto a los inconvenientes de esta topología, se encuentran los siguientes, el jugador que actué como servidor tiene ventaja respecto al resto ya que su latencia es menor y tiene más tiempo para pensar que los demás, sigue siendo compleja la implementación de esta estructura, requiere esfuerzos de programar migración de host si el jugador servidor se desconecta, programar la dualidad cliente / servidor al igual que el anterior… etc.

Por lo tanto también ha sido descartada y con ella plataformas como Mirror (Mirror, s.f.), Fish-Networking (Fish-Networking, 2023), que aunque permite todo tipo de topologías si queremos un servidor externo al final tienes que almacenarlo en una máquina virtual.

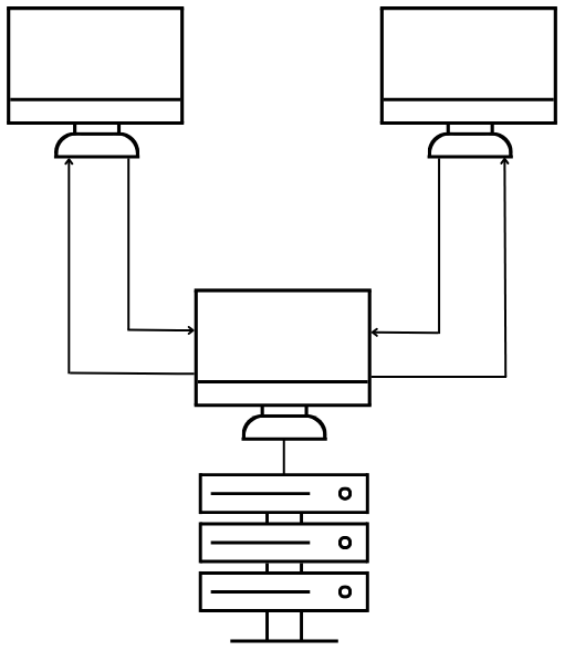

Ilustración 1.16 Topología Player-Hosted Client/Server Peer -To - Peer

1.8.4.4 Topología servidor dedicado

Esta topología es la más escalable y la más segura de todas las estudiadas, además brinda la flexibilidad de alojar los servidores en el lugar en el que el usuario desee, simplifica el código bastante ya que todos los jugadores son clientes que harán peticiones a este servidor. Su mayor inconveniente es que el precio es mayor a cualquier otra implementación.

En nuestro caso hemos decido que esta es la mejor opción por que independientemente de los costes un servidor dedicado es la solución más profesional lo cual va a permitir a nuestro juego proporcionar una mejor experiencia de usuario.

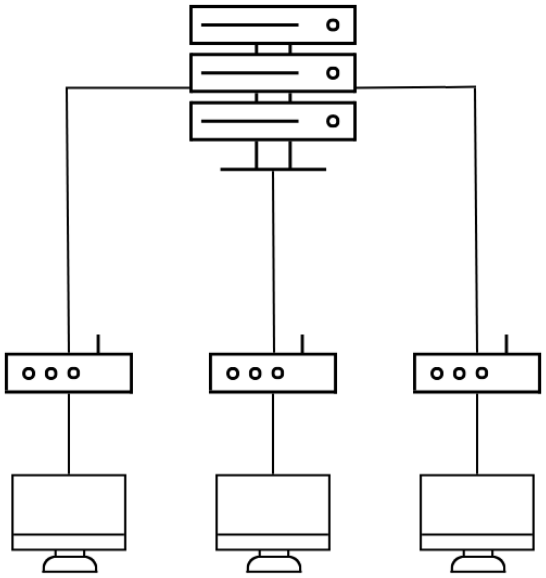

Ilustración 1.17 Topología servidor dedicado

1.8.5 Elección de la plataforma para soporte de partidas en línea

Una vez realizado el filtrado por la topología que se desea para el desarrollo del aspecto de una partida online compararemos las prestaciones que nos ofrecen las distintas empresas para lleva a cabo el videojuego.

1.8.5.1 Photon

Este software es una herramienta que proporciona un servidor dedicado para juegos multijugador, permite juego cruzado entre distintas plataformas. A su vez Photon ofrece un servicio de atención al cliente las 24 horas del día, por si ocurre algún tipo de problemas con el servidor contratado.

Otro aspecto importante de Photon es dispone de una muy buena documentación a la hora de realizar los juegos multijugadores (Photon, Photon Documentacion, s.f.). Este elemento lo hace muy atractivo a los desarrolladores.

Muchas empresas avalan el uso de Photon (Photon, s.f.) como Vodafone, P&P, e incluso existen videojuegos con un gran renombre que lo utilizan o Among Us VR, Stumble Guys. Que estos grandes nombres estén utilizando esta herramienta da confianza al desarrollador.

Lo último a tener en cuenta sobre Photon es su tarifa de precios (Photon, Photon Precios, s.f.), en las que se puede observar que los costes van asociados al número de usuarios simultáneos jugando al juego y estos costes son muy elevados para un número relativamente pequeño de jugadores simultáneos. Estos costos son tan altos porque Photon está enfocado más a juegos que requieran una baja latencia y con requerimientos del modo de juego online mucho más exigente del que se necesita para nuestro proyecto.

1.8.5.2 AppWarp

Esta herramienta no es tan conocida como el resto, pero también proporciona un servidor dedicado para realizar la comunicación entre los distintos jugadores, otro factor es importante es que como la gran mayoría permite un juego cruzado entre plataformas además de muchas características de personalización en la nube (AppWarp, s.f.).

Sin embargo uno de los inconvenientes es que esta herramienta no dispone de grandes videojuegos o empresa que la estén utilizando en la actualidad, por lo tanto seria arriesgado comenzar con un software que no dispone de estos respaldos.

En lo que se refiere a los precios por utilizar este servidor (AppWarp, AppWarp Precios, s.f.), podemos observar que los costos van asociados al número de peticiones que se realice. Dispone de un plan gratuito bastante generoso con dos millones de peticiones gratis, pero a partir de aquí los precios se disparan.

1.8.5.3 Playfab

Playfab es una plataforma de servicios en la nube pensada especialmente para los desarrolladores de videojuegos, actualmente es propiedad de Microsoft por lo que está vinculada a la nube Azure.

El punto fuerte de Playfab es la comodidad para el desarrollador, todo está configurado para que el desarrollador simplemente se preocupe de configurar los parámetros de sus implementaciones como economía interna o chats de comunicación, sistemas de emparejamiento, incluso autenticación de los usuarios (Microsoft, Playfab, s.f.).

Playfab es compatible con los motores de videojuego más importante lo cual hace que sea sencilla su instalación a través de un SDK, además de ello una gran cantidad de juegos importantes lo están utilizando, como Doom, Minecraft.

Al ser propiedad de Microsoft nos proporciona como solución una máquina virtual alojada en la nube de Azure para desarrollar el intercambio de información de los jugadores (Microsoft, Playfab | Servidores Multijador, s.f.), estos costes son independientes a los de PlayFab, por lo que para evaluar el coste total deberíamos de sumar ambos, los de tener una máquina virtual en la nube de Azure (Microsoft, Microsoft | Precios Azure, s.f.) y lo que tiene tener la herramienta de PlayFab (Microsoft, Playfab | Precios, s.f.), lo cual puede ser excesivo para lo que estamos buscando.

1.8.5.4 GameLit

En este apartado hablaremos de GameLit, la solución propuesta por Amazon para la comunicación entre jugadores en las sesiones de juego. Esta herramienta tiene una característica interesante y es que es capaz de apagar las máquinas virtuales si nadie está haciendo uso de ellas, por lo que ahorra en costes para ambas partes dependiendo del plan que se haya contratado. Es por ello que empresas con gran repercusión están utilizando esta herramienta, como es el ejemplo de Ubisoft.

La documentación es escueta y podría llegar a ser un problema con las futuras mejoras que se quieran implementar (Amazon, GameLit Docuemtacion, s.f.).

En lo que se refiere al precio, es de las soluciones que más se ajusta a un juego a las características de nuestro juego ya que ofrece servidores dedicados por precios bastantes económicos (Amazon, GameLit Precios, s.f.).

1.8.5.5 Suite Multiplayer Unity

La propia plataforma de Unity proporciona una suite de elementos para llevar a cabo un juego multijugador, dependiendo de los requerimientos del mismo. En el caso del prototipo de cartas sería excesivo utilizar el NetCode From GameObjects (Unity, Unity NetCode, s.f.), puesto que no sería necesario que todos los objetos de la escena estén sincronizados en sus movimientos, ya que esta actualización se puede realizar de forma independiente en cada instancia del juego con el paso de mensajes.

Por lo tanto de todos los elementos que proporciona el motor de videojuegos nos centraremos en el estudio del paquete Transport (Unity, Unity | Transport, s.f.), el cual permite la comunicación entre distintos jugadores a través de WebSocket, por ende lo hace compatible con cualquier plataforma y permite el juego cruzado. Todo ello conlleva el problema de que no tenemos un servidor dedicado para desarrollar el videojuego, lo cual contrasta con la idea de tener un servidor para la seguridad de las partidas.

Inicialmente integrar esta comunicación en Unity es totalmente gratuita, lo cual es incita a la implementación en nuestro proyecto, pero el problema ocurre cuando queremos escalar nuestro proyecto, dado que nos veremos obligados a utilizar Unity Relay (Unity, Unity | Relay, s.f.), el cual ya si es de pago y los precios son bastantes elevados para agregar otro tipo de funcionalidades como Lobbys en las partidas, economía interna, etc… (Unity, Unity | Gaming Services Precio, s.f.). El precio se basa de nuevo en el número de usuarios concurrentes algo que no interesa en el proyecto a realizar, dado que el número de mensajes por partida no es demasiado grande para los costes esperados.

Por estas razones se descartó realizar el multijugador con la propia solución de Unity, aunque es la más simple y accesible no encaja con los requerimientos que se buscan para la aplicación.

1.8.5.6 Firebase

Esta plataforma en si no está destinada para el desarrollo de videojuegos, pero puede ser adaptada para el mismo. La idea de estudiar este tipo de plataformas surge por la lectura de foros para desarrollar un juego basado en turnos, que es el problema que se afronta en el trabajo de fin de grado.

Sabemos que no necesitamos una latencia baja para la actualización de los movimientos, esta es una característica de los juegos basados en turno, no hay ningún problema en que la interfaz se actualice un segundo o dos más tarde de lo esperado, es por ello que podemos utilizar la base de datos en tiempo real de Firebase (Google, Firebase | Real Time Database, s.f.), como si fuera un servidor dedicado. De esta manera los jugadores podrían publicar sus movimientos en esta base de datos y el resto leerlo y actualizar sus propias instancias en el juego.

Esta característica de lectura de datos hace que se permita el juego cruzado al igual que cualquier servidor dedicado estudiado con anterioridad, encima la comunicación con Firebase se puede realizar con una API Rest (Google, Firebase | API Rest, s.f.), lo cual lo hace accesible a cualquier tipo de dispositivo ya que la comunicación se realiza a través de peticiones HTTPS.

Otro aspecto importante es la empresa que lo respalda, que es Google, lo cual genera mucha confianza a la hora de utilizarla. Como es sabido Firebase es altamente escalable y podemos utilizar muchos servicios de la propia suite de Google para el trabajo, como la autentificación, tipos de verificación y todas las funcionalidades que se puedan imaginar con Google Cloud (Google, Google Cloud, s.f.).

Pero el aspecto más importante de este servicio y por el cual se ha decidido que era el mejor para el proyecto es sus los precios que ofrecen (Google, Firebase | Precios, s.f.), con diferencia los más económicos para nuestro proyecto en específico y es por ello que se ha seleccionado, se ha decidido sacrificar un poco de esfuerzo en la configuración del mismo a cambio de los ahorros que supondrá.

1.9 Descripción de la solución propuesta

En este apartado explicaremos brevemente el esquema final que tendrá este proyecto con las propuestas finales seleccionadas de todas las alternativas estudiadas.

Por lo tanto el esquema final será un proyecto en Unity el cual será desarrollado en lenguaje C#, el cual por un lado se comunicara a través de Sockets con otro proyecto Java el cual almacenara la plataforma de Agentes implementados en JADE.

Por otro lado para las partidas en línea se comunicara con las *Cloud Functions*, las cuales actuaran como ese servidor dedicado que buscamos, las cuales serán capaces de autorizar movimientos permitidos o no, eliminar jugadores si están siguiendo las reglas, etc... Además también se comunicara con la API a través de peticiones HTTPS para recibir la información sobre los otros jugadores de la partida e incluso actualizar datos que no requieran supervisión del "Servidor Dedicado" como podría ser el estado actual del jugador, o apuntarse a una lista de juego. En la siguiente ilustración podemos ver el esquema.

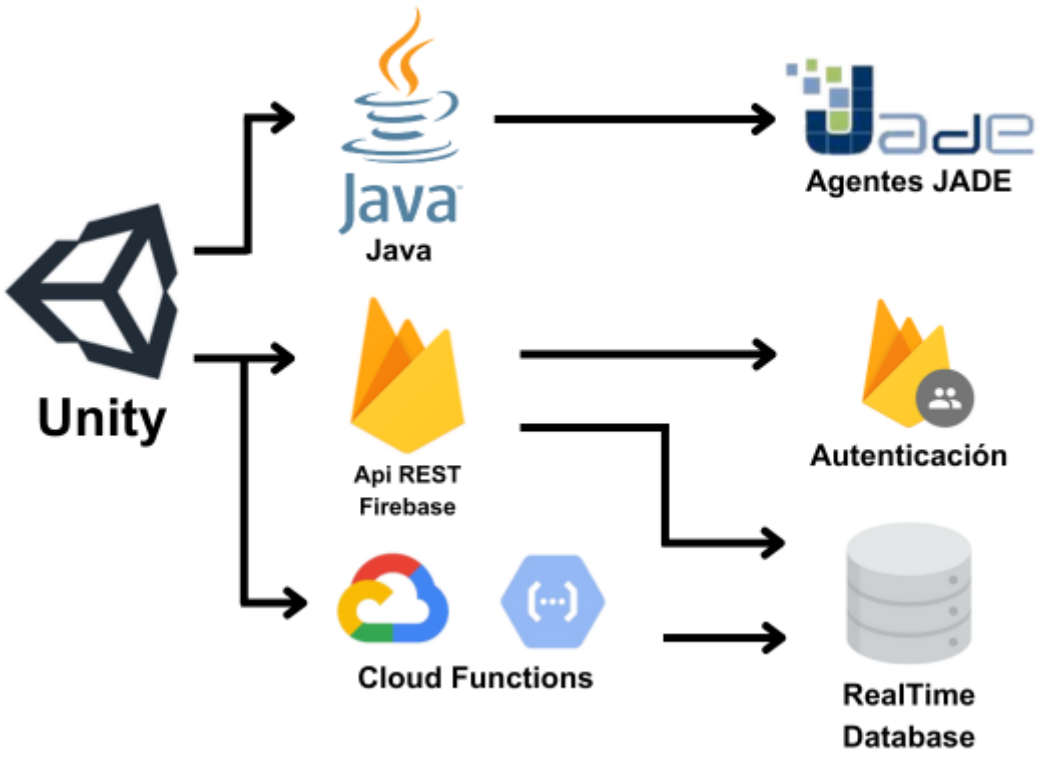

Ilustración 1.18 Esquema de comunicación de la plataforma de juego de cartas

1.10 Tecnologías utilizadas

En esta sección se hará una breve explicación sobre el conjunto de programas que han sido utilizados para el desarrollo del videojuego y las funcionalidades que nos han aportado cada uno de ellos.

1.10.1 Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft, que se distribuye bajo la licencia MIT. Funciona en cualquier sistema operativo, tiene una interfaz intuitiva y personalizable, y soporta una amplia gama de lenguajes de programación que pueden ser utilizados fácilmente con la instalación de plugin, además da soporte para snippets tanto de C# y de Unity de forma muy simple por ello se ha decidido utilizar este editor de código (Microsoft, Visual Studio Code, s.f.). Además de todas estas ventajas también tiene soporte integrado con Git y Azure, lo que permite desarrollar y depurar el código en el mismo lugar. A opinión personal sobre esta herramienta, que es la primera vez que la utilizo, he quedado bastante satisfecho con ella el único inconveniente es a la hora de instalar librerías que es un poco más complejo de lo habitual ya que esta herramienta no está destinada a ello.

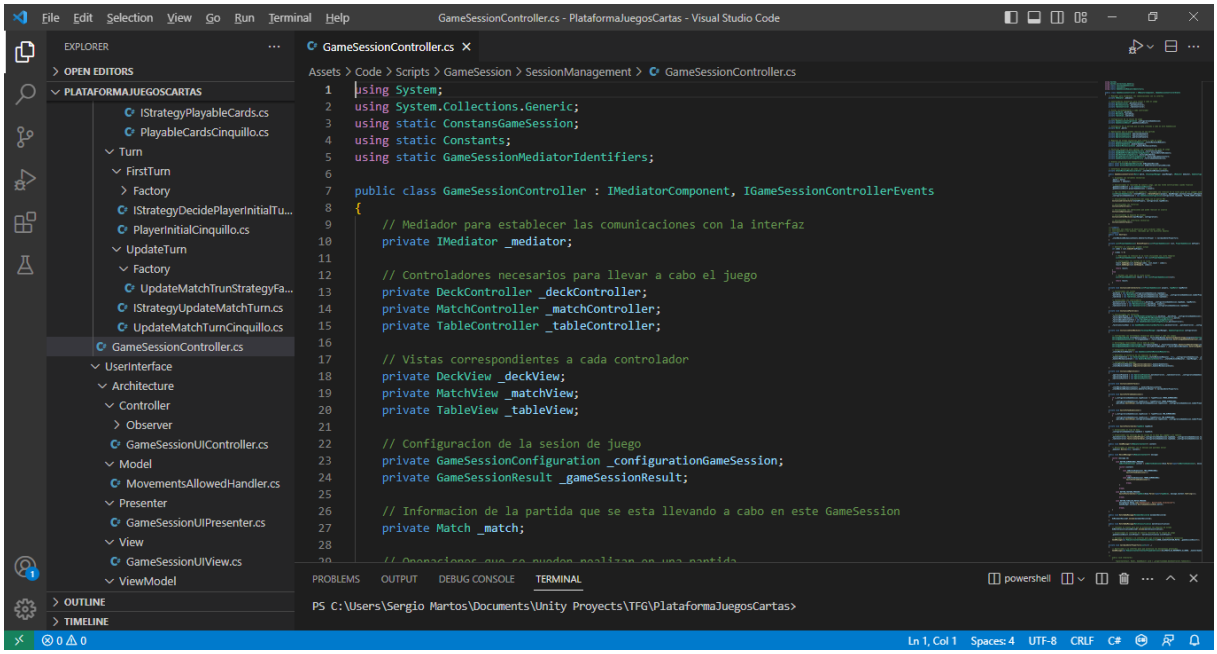

Ilustración 1.19 Captura del IDE Visual Studio Code

1.10.2 NetBeans

Es un IDE fundado por Sun Microsystems, el cual es actualmente controlado por Oracle Corporation (Apache, s.f.), la licencia bajo la cual se distribuye NetBeans es la CDDL y GPL. Al igual que Visual Studio Code funciona en cualquier sistema operativo, pero está más orientado al desarrollo de aplicaciones Java, (Ilustración 1.20). Se ha decidido la utilización de este IDE para poder implementar la plataforma de agentes JADE, ya que en este entorno de desarrollo es bastante fácil la instalación de librerías, y existe bastante documentación sobre cómo utilizar la plataforma JADE en Netbeans. Además, también ha sido un software bastante utilizado durante el grado por lo que no existirán tantos problemas de configuración, instalaciones, etc.

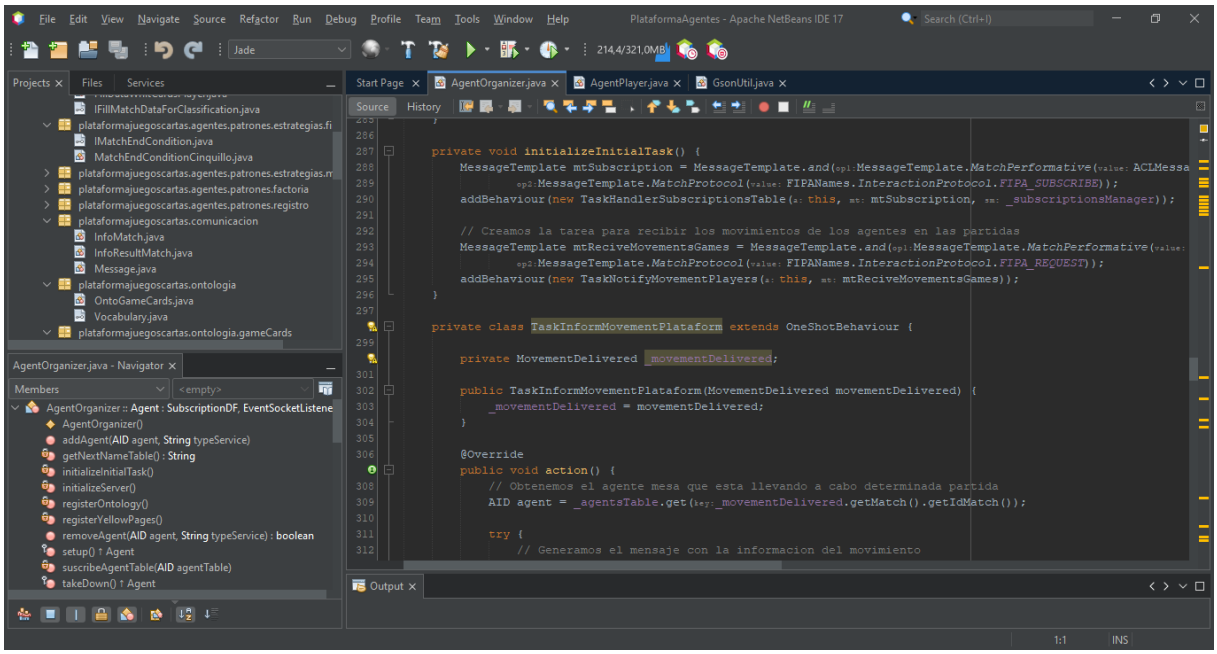

Ilustración 1.20 Captura del IDE Netbeans

1.10.3 UMLet

UMLet es una herramienta gratuita y de código abierto, distribuida bajo la licencia GNU GPL, con una interfaz de usuario amigable y sencilla, la cual permite crear diagramas de diversos tipos (UMLet, s.f.). Nos permite generar diagramas tanto arrastrando y soltando elementos, como a través de markdown. Además, dispone de la opción de incorporarse a ciertos IDEs, como Eclipse o Visual Studio Code. Se ha creído conveniente la utilización de este programa por la simplicidad y personalización que nos ofrece a la hora de crear cualquier tipo de diagrama con cualquier tipo de información.

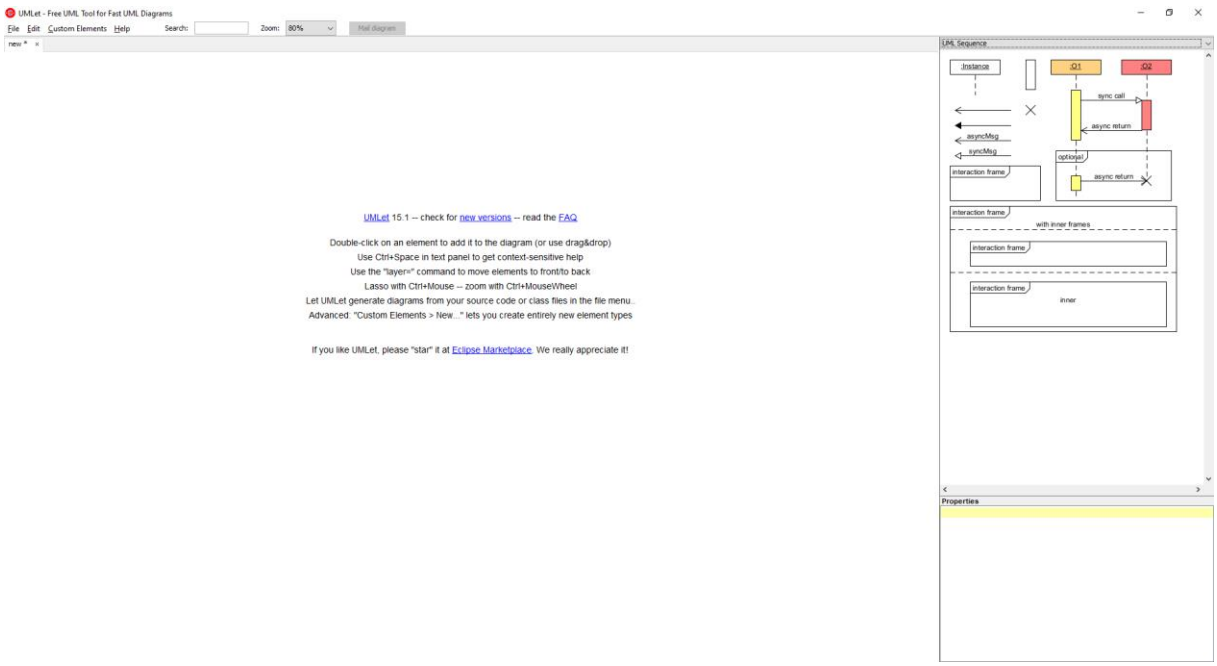

Ilustración 1.21 Captura de la herramienta UMLet

1.10.4 GIMP

GIMP es un software utilizado para la manipulación de imágenes, sus siglas son un acrónimo de GNU Image Manipulation Program. Como su nombre nos anticipaba se distribuye bajo la licencia publica general de GNU (Gimp, 2023). En nuestro proyecto se ha utilizado para mejorar ciertos aspectos de la interfaz. Una característica interesante de GIMP es que es altamente modificable y se le pueden agregar extensiones a través de plugin o script. La decisión de utilizar este programa para la creación de imágenes ha sido por su coste y por el rendimiento que ofrece, aunque si bien es cierto que no es de las mejores herramientas de edición que existen para nuestro uso es más que suficiente.

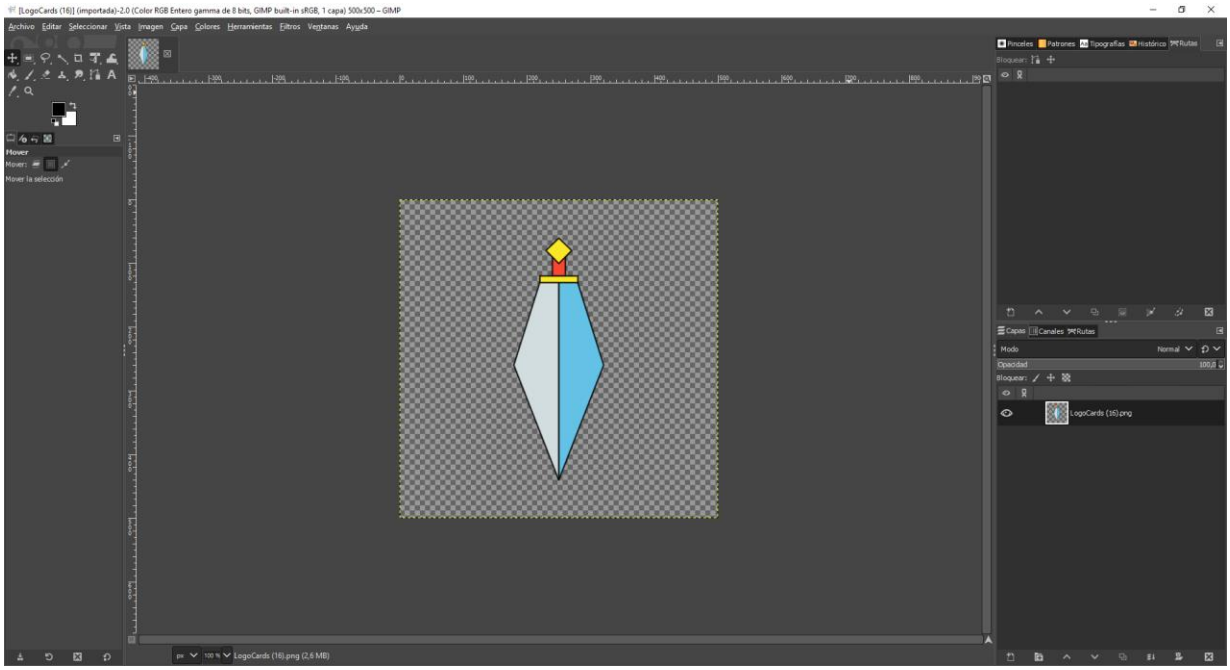

Ilustración 1.22 Captura de la herramienta GIMP

1.10.5 Blender

Blender es una potente suite de software de código abierto diseñada principalmente para la creación de contenido 3D, animaciones, modelado, renderizado y más. Ha sido publicada bajo la licencia de GNU (Blender, Blender, s.f.). En nuestro, caso necesitábamos solamente un modelo, que era la carta en sí, así que Blender era la opción ideal para nuestros requerimientos. De esta forma hemos podido personalizar la texturización de la carta y en un futuro si se considera necesario crear animaciones o cualquier tipo de modificación.

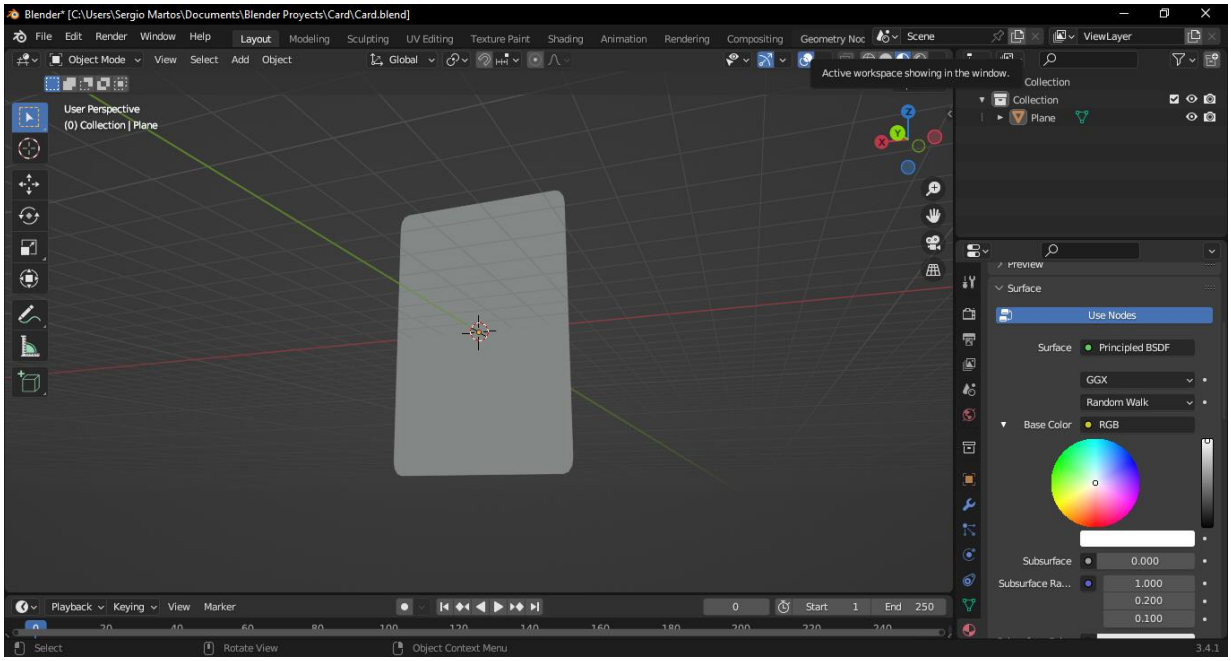

Ilustración 1.23 Captura de la herramienta Blender

1.11 Metodología de desarrollo de software

Dada la naturaleza de un videojuego, que requiere pruebas intermedias durante el desarrollo, se decidido utilizar una metodología incremental general. El modelo incremental, como se conoce, aplica una secuencia lineal de incrementos a lo largo del tiempo. Cada una de estas produce incrementos y consigo resultados del software. A estos incrementos también se les denomina iteraciones y cada incluyen más opciones y funcionalidades. (Pressman, 2010).

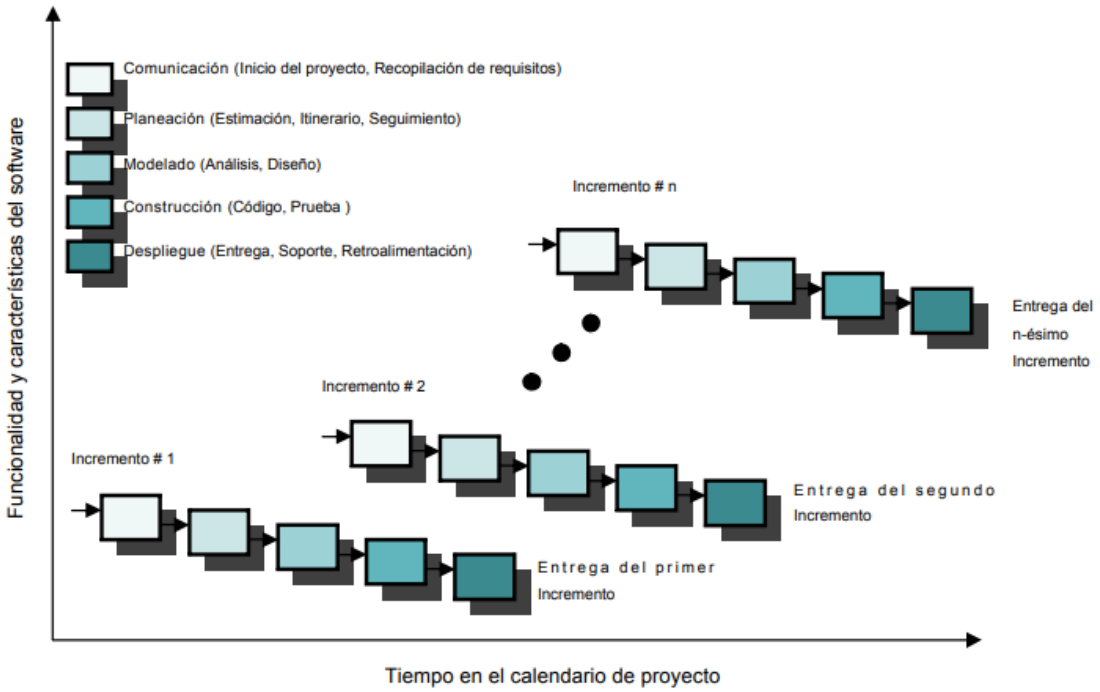

Ilustración 1.24 Esquema del modelo incremental

Cuando se utiliza un modelo incremental como el que podemos observar en la ilustración, las primeras iteraciones se denominan iteraciones núcleo (Pressman, 2010) porque normalmente son la iteraciones que requieren más trabajo además de ser las que centran y dan estructura al proyecto.

Para este caso se han propuesto las siguientes iteraciones:

1. Puesta en marcha del sistema.
2. Configuración de comunicación con plataformas externas.
3. Implementación de la sesión de juego.
4. Implementación de la plataforma de Agentes.

5. Implementación de la plataforma de Firebase.
6. Agregando elementos decorativos

Con cada iteración agregamos nuevos elementos y pulimos todas las iteraciones anteriores para que sigan funcionando como se esperaba con las nuevas actualizaciones.

1.12 Análisis de riesgos

En este apartado se identificarán los riesgos que afectan al trabajo, tanto en su planteamiento como en el desarrollo, para ello nos basaremos en la metodología MAGERIT (Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información). Fue elaborada por el antiguo Consejo Superior de Administración Electrónica y actualmente es mantenida por la Secretaría General de Administración Digital con la colaboración del Centro Criptográfico Nacional (PAE, 2012).

1.12.1 Caracterización de los activos

Esta actividad busca identificar los activos relevantes dentro del sistema a analizar, caracterizándolos por el tipo de activo, identificando las relaciones entre los diferentes activos, determinando en qué dimensiones de seguridad son importantes y valorando esta importancia.

Dentro del ámbito empresarial, los activos representan los bienes, derechos y otros recursos controlados económicamente por la empresa, resultantes de sucesos pasados, de los que se espera que la empresa obtenga beneficios o rendimientos económicos en el futuro. (Sage, s.f.).

Por lo tanto teniendo esta información en cuenta se deberán identificar aquellos aspectos de los cuales se dependen para desarrollar este proyecto de fin de grado.

1.12.1.1 Identificación de activos

La primera fase consiste en identificar los activos, en nuestro caso podemos diferenciar los siguientes:

- Aplicaciones informaticas: Estas seran utilizadas para el desarrollo de nuestro de videojuego.
- Equipos informaticos: Que seran los encargados de hospedar las aplicaciones informaticas.
- Las redes de comunicacion: Las utilizaremos para conexiones a interntet, busqueda de informacion… etc.
- Las personas: En este caso, solamente existe una persona como activo que es el propio desarrollador del trabajo.

1.12.1.2 Dependencias entre activos

Siguiendo con la caracterización de los activos en este apartado se establecerá la referencia entre los propios activos.

Lo primero que vamos a realizar es una referencia de cada activo con un símbolo para que sea más entendible, donde A = Aplicaciones informáticas, B = Equipo informático, C = Redes de comunicación y D = Personal.

Para este caso se realizará de forma cualitativa, y el resultado es el que podemos observar en la ilustración 1.25

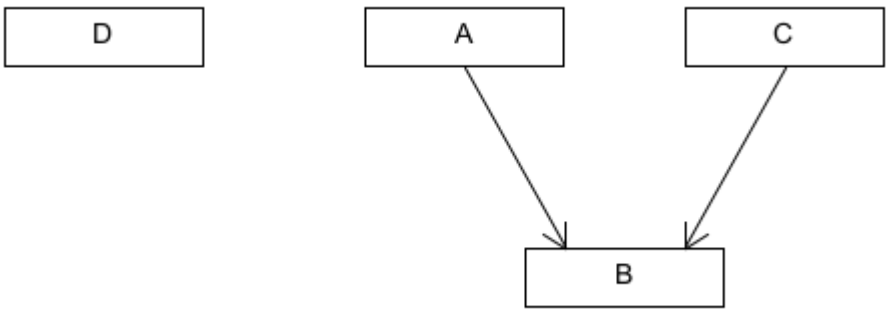

Ilustración 1.25 Esquema de dependencia entre activos

También puede expresarse de forma algebraica, que sería de la siguiente forma:

$$(D) \wedge (A \rightarrow B) \wedge (C \rightarrow B) \wedge (B)$$

1.12.1.3 Valoración de activos

A continuación, se va a proceder a la valoración de activos. En esta metodología nos indican que los activos se valoran por cinco dimensiones, que son:

- Disponibilidad: ¿Qué perjuicio causaría no tenerlo o no poder utilizarlo?
- Integridad de los datos: ¿Qué perjuicio causaría que estuviera dañado o corrupto?
- Confidencialidad de la informacion: ¿Qué daño causaría que lo conociera quien no debe?
- Autenticidad: ¿Qué perjuicio causaría no saber exactamente quien hace o ha hecho cada cosa?
- Trazabilidad: ¿Qué daño causaría no saber a quién se le presta tal servicio, quien accede a los datos y que hace con ellos?

En base a estos atributos le tenemos que asignar un valor a cada activo, para ello vamos a utilizar la tabla que nos proporciona el catálogo de MAGERIT, la cual cuantifica el valor de nuestros activos a efectos de posibles riesgos.

Valor		Criterio
10	Extremo	Daño extremadamente grave
9	Muy alto	Daño muy grave
6-8	Alto	Daño grave
3-5	Medio	Daño importante
1-2	Bajo	Daño menor
0	Despreciable	Importante a efectos prácticos

Tabla 1.1 Criterios de la importancia del riesgo

En base a la tabla 1.1 y a las dimensiones a tener en cuenta por cada activo, obtenemos los siguientes valores para los mismos:

Activo	Valor
Equipo Informático	Extremo
Aplicaciones informáticas	Alto
Redes Comunicación	Medio
Personas	Extremo

Tabla 1.2 Valores de la importancia del riesgo

1.12.2 Caracterización de las amenazas

Esta actividad busca identificar las amenazas relevantes sobre el sistema a analizar, caracterizándolas por las estimaciones de ocurrencia (probabilidad) y daño causado (degradación).

1.12.2.1 Identificación de las amenazas

En este apartado se identificarán las posibles amenazas hacia nuestros activos, en esta metodología se diferencian cinco grupos de posibles amenazas, los cuales son:

- Origen natural
- Origen Industrial
- Defectos en las aplicaciones
- Causadas por personas accidentalmente
- Causadas por personas deliberadamente

Una vez ejemplificado los grupos de posibles amenzas vamos a identificarlas, indicar a que activos afecta y en que dimension, ademas de identificar a que grupo pertenece cada amenaza

- Fuego (Desastres naturales, Origen industrial): Afecta a la disponibilidad de los equipos informaticos.
- Daños por agua (Desastres naturales, Origen industrial): Afecta a la disponibilidad de los equipos informaticos.
- Desastres naturales (Desastres naturales): Engloba otros incidentes como tormentas electricas, terremotos…etc. Afectan a la disponibilidad de los equipos informaticos.
- Desastres industriales (Origen industrial): Engloba otros incidentes como explosiones, accidentes… etc. Afectan a la disponibilidad de los equipos informaticos.
- Contaminación (Origen industrial): Afecta a la disponibilidad de equipos informaticos.

- Avería de origen físico o lógico (Origen industrial): Afecta a la disponibilidad tanto de equipos como de aplicaciones informaticas.
- Corte del suministro eléctrico (Origen industrial): Afecta a la disponibilidad de los equipos informaticos.
- Fallo de servicios de comunicaciones (Origen industrial): Afecta a la disponibilidad de las redes de comunicacion.
- Errores del administrador (Causadas por personas accidentalmente): Afecta a la disponibilidad, integridad y confidencialidad de las aplicaciones, los equipos informaticos y las redes de comunicación.
- Difusión de software dañino (Causadas por personas accidentalmente): Afecta a la disponibilidad, integridad y confidencialidad de las aplicaciones.
- Alteración accidental de la informacion (Causadas por personas accidentalmente): Afecta a la integridad de las aplicaciones y redes de comunicacion.
- Destrucción de la información (Causadas por personas accidentalmente): Afecta a la disponibilidad de las aplicaciones y redes de comunicacion.
- Vulnerabilidades de los programas (Causadas por personas accidentalmente): Afectan a la integridad, disponibilidad y confidencialidad de las aplicaciones informaticas.
- Errores de mantenimiento / actualizaciones de programas software (Causadas por personas accidentalmente): Afecta a la integridad y disponibilidad de las aplicaciones informaticas.
- Caída del sistema por agotamiento de recursos (Causadas por personas accidentalmente): Afecta a la disponibilidad de equipos informaticos y redes de comunicaciones.
- Perdida de equipos (Causadas por personas accidentalmente): Afecta a la disponibilidad y confidencialidad de los equipos informaticos.
- Indisponibilidad del personal (Causadas por personas accidentalmente): Afecta a la disponibilidad del personal.

1.12.2.2 Valoración de las amenazas

Para realizar la valoración de una amenaza debemos de valorar la influencia sobre el activo en dos sentidos:

- Degradacion: Cuán perjudicado resultaria el activo. Esta se calculara dividiendo el valor del activo por la fraccion perjudicada o dañada.
- Probabilidad: Cuán problabe es que se materialice la amenza. Para ello se utiliza una tabla de frecuencias la cual se modelara numericamente como indica la tabla 1.3

Frecuencia		Criterio
MA	100	A diario
A	10	Mensualmente
M	1	Una vez al año
B	1/10	Cada varios años
MB	1/100	Siglos

Tabla 1.3 Criterios de la frecuencia de riesgos

Cuando un activo es víctima de una amenaza, no se ve afectado por igual en todas las dimensiones, ni tampoco es afectado con la misma cuantía para cada tipo de dimensión.

Otro aspecto que se debe tener en cuenta dentro de la valoración de las amenazas es la intencionalidad, si la amenaza ocurre de forma no intencionada, la estimación de riesgos será correcta y se podrá guiar por el análisis que se está realizando en este documento. Sin embargo si se produce una amenaza de forma intencionada no se puede pensar en la proporcionalidad del atacante y la estimación puede diferir bastante de lo ocurrido en realidad.

Teniendo en cuenta la tabla 1.3 y las posibles amenazas identificadas, se le asignara una frecuencia o probabilidad de que ocurra esta amenaza (D representa Disponibilidad, C representa Confidencialidad, I representa Integridad). También se tendrá en cuenta la degradación que causara a cada una de las dimensiones en caso de que se sufra la amenaza, para posteriormente continuar con el análisis de riesgos.

Amenaza	Frecuencia	D	C	I
Fuego	1/10	90%	-	-
Daños por agua	1/10	90%	-	-
Desastres naturales	1/100	90%	-	-
Desastres industriales	1/100	90%	-	-
Contaminación	100	80%	-	-
Avería de origen físico o lógico	1	90%	-	-
Corte del suministro eléctrico	1	90%	-	-
Fallo de servicios de comunicaciones	1	30%	-	-
Errores del administrador	10	20%	50%	95%
Difusión de software dañino	1	95%	95%	95%
Alteración accidental de la información	1	-	-	80%
Destrucción de la información	1	80%	-	-
Vulnerabilidades de los programas	10	20%	60%	90%
Errores de mantenimiento / actualizaciones de programas software	1	50%	-	50%
Caída del sistema por agotamiento de recursos	1/10	95%	-	-
Perdida de equipos	1/10	99%	99%	-
Indisponibilidad del personal	1	99%	-	-

Tabla 1.4 Valores de la frecuencia de riesgos

1.12.3 Estimación del impacto

Se denomina impacto a la medida del daño sobre el activo derivado de la materialización de una amenaza. Conociendo el valor de los activos (en varias dimensiones) y la degradación que causan las amenazas, es directo derivar el impacto que estas tendrían sobre el sistema.

Para realizar una correcta valorización sobre el impacto de una materialización de una determinada amenaza hay que tener en los siguientes tipos de impacto:

- Impacto repercutido: Que es el valor de cada activo junto con las amenzas a las que estan expuestos los activos que depende
- Impacto acumulado: Que es el valor de los activos mas el valor de los activos que depnden de el y los activos de los que depende.

Existen diferentes herramientas para tener en cuenta los posibles impactos sobre los activos. Para el análisis de riesgos implementado en este proyecto utilizaremos esta tabla para relacionar el valor del activo con la degradación del mismo si se materializa dicha amenaza.

Impacto		Degradación del activo		
		0% - 24%	25% - 89%	90% - 100%
Valor	Extremo	Alto (A)	Muy Alto (MA)	Muy Alto (MA)
	Muy alto	Medio (M)	Alto (A)	Muy Alto (MA)
	Alto	Medio (M)	Alto (A)	Alto (A)
	Medio	Bajo (B)	Medio (M)	Medio (M)
	Bajo	Bajo (B)	Bajo (B)	Medio (M)
	Despreciable	Bajo (B)	Bajo (B)	Bajo (B)

Tabla 1.5 Criterios de la estimación del impacto

Teniendo en cuenta la tabla 1.5, obtenemos las siguientes estimaciones de impacto para nuestros activos.

Activo	Amenaza	Impacto
Equipo informático	Fuego	Muy alto
Equipo informático	Daño por agua	Muy alto
Equipo informático	Desastres Naturales	Muy alto
Equipo informático	Desastres Industriales	Muy alto
Equipo informático	Contaminación	Muy alto
Equipo informático	Avería de origen físico o lógico	Muy alto
Aplicaciones informáticas	Avería de origen físico o lógico	Alto
Equipo informático	Corte del suministro eléctrico	Muy alto
Redes de comunicación	Fallo se servicio de comunicaciones	Medio
Equipo informático	Errores del administrador	Muy alto
Aplicaciones informáticas	Errores del administrador	Alto
Redes de comunicación	Errores del administrador	Medio
Aplicaciones informáticas	Difusión de software dañino	Alto
Aplicaciones informáticas	Alteración accidental de la información	Alto
Redes de comunicación	Alteración accidental de la información	Medio
Aplicaciones informáticas	Destrucción de la información	Alto
Redes de comunicación	Destrucción de la información	Medio
Aplicaciones informáticas	Vulnerabilidades de los programas	Alto
Aplicaciones informáticas	Errores de mantenimiento / actualizaciones de programas software	Alto
Equipo informático	Caída del sistema por agotamiento de recursos	Muy alto
Redes de comunicación	Caída del sistema por agotamiento de recursos	Medio
Equipo informático	Perdida de equipos	Muy alto
Personal	Indisponibilidad del personal	Muy alto

Tabla 1.6 Valores de impacto en los activos

1.12.4 Estimación del riesgo

En el siguiente apartado haremos una estimación sobre el cálculo del riesgo. Se denomina riesgo a la medida del daño probable sobre un sistema. Conociendo el impacto de las amenazas sobre los activos, es directo derivar el riesgo sin más que tener en cuenta la probabilidad de ocurrencia.

Este tipo de metodología identifica cuatro posibles zonas para los riesgos:

- Zona 1: Riesgos muy problabes y de muy alto impacto
- Zona 2: Cubre un amplio rango desde situaciones improbables y de impacto medio, hasta situaciones muy probables pero de impacto bajo o muy bajo
- Zona 3: Riesgos improbables y de bajo impacto
- Zona 4: Riesgos improbables pero de muy alto impacto

Teniendo en cuenta las zonas identificadas usaremos la tabla 1.7 la cual muestra la relación entre la frecuencia de la amenaza y el impacto. Con dicha tabla se cubrirán el espectro de todas las zonas y la estimación de riesgos se podrá hacer de simple y entendible.

Riesgo		Frecuencia				
		100	10	1	1/10	1/100
Impacto	Muy alto	Muy alto	Muy alto	Alto	Medio	Medio
	Alto	Muy alto	Alto	Medio	Bajo	Bajo
	Medio	Alto	Alto	Medio	Bajo	Muy bajo
	Bajo	Medio	Medio	Bajo	Muy bajo	Muy bajo

Tabla 1.7 Criterios para los posibles riesgos

Teniendo en cuenta la tabla 1.7, obtenemos las siguientes estimaciones de riesgo para nuestros activos, dependiendo del tipo de amenaza.

Activo	Amenaza	Riesgo
Equipo informático	Fuego	Medio
Equipo informático	Daño por agua	Medio
Equipo informático	Desastres Naturales	Medio
Equipo informático	Desastres Industriales	Medio
Equipo informático	Contaminación	Muy alto
Equipo informático	Avería de origen físico o lógico	Alto
Aplicaciones informáticas	Avería de origen físico o lógico	Medio
Equipo informático	Corte del suministro eléctrico	Alto
Redes de comunicación	Fallo se servicio de comunicaciones	Medio
Equipo informático	Errores del administrador	Muy alto
Aplicaciones informáticas	Errores del administrador	Alto
Redes de comunicación	Errores del administrador	Alto
Aplicaciones informáticas	Difusión de software dañino	Medio
Aplicaciones informáticas	Alteración accidental de la información	Medio
Redes de comunicación	Alteración accidental de la información	Medio
Aplicaciones informáticas	Destrucción de la información	Medio
Redes de comunicación	Destrucción de la información	Medio
Aplicaciones informáticas	Vulnerabilidades de los programas	Alto
Aplicaciones informáticas	Errores de mantenimiento / actualizaciones de programas software	Medio
Equipo informático	Caída del sistema por agotamiento de recursos	Medio
Redes de comunicación	Caída del sistema por agotamiento de recursos	Bajo
Equipo informático	Perdida de equipos	Medio
Personal	Indisponibilidad del personal	Alto

Tabla 1.8 Valores de riesgos para las posibles amenazas

1.12.5 Caracterización de salvaguardas

En este apartado se identificarán las salvaguardas necesarias para minimizar los daños en caso de que se produzca una amenaza. Ante el amplio abanico de posibles salvaguardas a considerar, es necesario hacer una criba inicial para quedarnos con aquellas que son relevantes para lo que hay que proteger. En esta criba se deben tener en cuenta los siguientes aspectos:

1. Tipo de activos a proteger, pues cada tipo se protege de una forma específica.
2. Dimensión o dimensiones de seguridad que requieren protección
3. Amenazas de las que necesitamos protegernos
4. Si existen salvaguardas alternativas

Teniendo en cuenta estos aspectos se han podido identificar las siguientes salvaguardas:

- Copias de seguridad: De esta forma tendremos copias de respaldo para restablecer el proyecto en caso de cualquier tipo de error en cuanto a modificaciones, vulnerabilidades, software dañino o perdida de informacion o equipos.
- Aseguramiento de la disponibilidad: Para ello se utilizara otro equipo con menores prestaciones que el principal, el cual permite continuar con el trabajo en caso de que el principal no funcione correctamente.
- Proteccion contra programas malignos: Para evitar vulnerabilidades y difusion de software dañino se utilizara un antivirus.
- Monitorizacion del sistema: Uso de registros log para controlar que todo esta funcionando como se espera.
- Politica de actualizacion: Implementar una politica de actualizacion, la cual indique que el software no debe ser actualizado hasta que se tenga una version estable y se haya realizado una copia de seguridad previamente.

1.13 Estimación del tamaño y esfuerzo

En este apartado vamos a realizar la estimación de tamaño y esfuerzo del TFG, para ello nos vamos a basar en el modelo de estimación COCOMO, el cual es un modelo de estimación de costes creado por Barry W.Bohem en el que se incluyen tres submodelos cada uno con un nivel de detalle específico (Wikipedia, s.f.).

En dicha estimación se utilizara el modelo intermedio, con la intención de dar una estimación más precisa sobre el esfuerzo. Además para ello es necesario identificar las características del proyecto así como los factores de ajuste que se utilizaran para el mismo (Artemisa, s.f.).

El proyecto tendrá alrededor de una cantidad de 6 KDLC, se implementara dentro de un entorno de desarrollo estable, formado un solo desarrollador.

En cuanto a los atributos conductores del coste necesarios para calcular el factor de ajuste de esfuerzo se ha considerado que la fiabilidad del proyecto debe ser nominal, con una complejidad del producto alta, un analista con una capacidad alta, en el que se tiene una baja experiencia para el desarrollo de este tipo de aplicaciones, aunque la experiencia del programador es alta, tiene una experiencia normal con los lenguajes de programación y el uso de las herramientas software requeridas para el proyecto es alto.

Para este caso en específico solamente se calculara el esfuerzo ya que al tratarse del TFG existen restricciones de tipo temporal, que es la duración del trabajo y personal, que viene dada por el número de alumnos que efectúan el trabajo.

Con todos los requerimientos necesarios ya identificados, lo que se debe hacer es calcular es esfuerzo, para ello se utilizara la siguiente formula:

$$Esfuerzo = a \cdot KLDC^{b} \cdot FAE$$

El elemento FAE viene determinado por los atributos conductores del coste, el cual aplicando la tabla de la referencia nombrada anteriormente obtenemos que el valor de FAE es 0,8746.

Por lo tanto ya sabiendo este valor, que a y b son valores constantes dado por el modo de COCOMO, que para este caso será el orgánico, dada las características del número de líneas de código y las condiciones del grupo de trabajo que realizara el proyecto, y que ya se ha hecho la estimación de las KDLC, podemos estimar que el esfuerzo será de 17.631 personas / mes

1.14 Presupuesto

En este apartado se hablara sobre los costes del proyecto software, para ello se deben tener en cuenta los costes asociados al software, a los recursos materiales y al personal. Dichos costes detallaran a continuación en los siguientes apartados.

Los costes que se van a considerar son ficticios y estimados, por lo que no hay que regirse por la exactitud de los resultados obtenidos.

1.14.1 Costes asociados al software

En este apartado se incluirá todos los costes asociados al software utilizado para el desarrollo del proyecto.

En la tabla 1.9 se muestra el desglose de cada uno de los programas utilizados.

Programa	Objetivo	Coste
NetBeans	Programación	0.00€
Visual Studio Code	Programación	0.00€
Blender	Modelar objetos 3D	0.00€
GIMP	Diseño elementos interfaz	0.00€
Unity	Motor de desarrollo	0.00€
UMLet	Diseño de diagramas	0.00€
Coste total:		0.00€

Tabla 1.9 Costes asociados al software

Los programas desglosados son los necesarios a la vista de la planificación realizada. Sin embargo, es probable que durante el desarrollo pueda ser necesario el uso de algún programa (o de alguna tecnología) cuya licencia o uso pueda no ser gratuito. Dichos gastos entrarían dentro del porcentaje destinado para los gastos imprevistos, desarrollado al final de este punto.

1.14.2 Costes asociados a los recursos materiales

En este apartado se tendrán en cuenta los costes asociados a los materiales utilizados para llevar a cabo el TFG.

Para calcular el costo, tenemos que considerar que los bienes adquiridos no se utilizan exclusivamente para la elaboración de este proyecto, es decir estos pueden

seguir utilizándose después de la finalización del mismo, por lo tanto, debemos realizar la amortización parcial de dichos bienes.

Lo primero que tenemos que hacer es buscar la información referente a los bienes adquiridos en las tablas de amortización lineal que nos ofrece el ministerio (Ministerio de Hacienda y Funcion Publica, 2020). En esta tabla podemos observar dos datos, los cuales son el coeficiente lineal máximo, que estima la vida útil del bien adquirido, expresado en porcentaje máximo por año y el periodo máximo el cual nos determina la horquilla en la que podemos amortizar cada año.

En primer lugar, utilizaremos esta fórmula para calcular el coeficiente mínimo de amortización, de esta forma podremos saber cuál es el mínimo que la empresa puede declarar a la hora de realizar las amortizaciones. La fórmula es la siguiente:

$$coeficeinte\ minimo\ amortizacion = \frac{100}{Periodo\ Maximo}$$

Ecuación 1.1 Coeficiente mínimo de amortización

Donde 100 representa el porcentaje total, divide entre el periodo máximo, que es el tiempo total en el cual el ordenador esta amortizado.

Después de saber cómo se determina la horquilla de amortización debemos realizar el coste dependiendo del coeficiente estimado, para ello utilizaremos la siguiente formula:

$$coste = \left(\frac{AporteInicial \times CoeficienteLineal}{12}\right) \times NumeroMeses$$

Ecuación 1.2 Coste del producto amortizado

Donde el aporte inicial representa el coste del ordenador, el coeficiente inicial viene dado por el valor de estimación que quiera darle el empresario comprendido entre la horquilla calculada, divido por 12, porque son los meses que tiene un año, y multiplicado por el número de meses que ha sido utilizado.

Seguidamente con los datos obtenidos en el sitio web, los cuales vienen dados con “Equipos para procesos de la información” para el equipo de desarrollo necesario para el proyecto y con “Otros Enseres” donde englobaremos todos los gastos relacionados, con folios, bolis, carpetas para organizar la información…etc. Podemos estimar lo siguiente.

La horquilla entre la cual podemos declarar como coeficiente de amortización para el equipo informático oscila entre 12,5% y 25% y para los otros enseres entre el 15% y 7%. Lo cual nos indica que el ordenador puede estar amortizado entre 4 y 8 años y los otros enseres entre 6,6 y 14 años. Para nuestro caso trataremos de amortizarlo lo antes posible por lo que nos ceñiremos al coeficiente del 25% para el ordenador y del 15% para los enseres.

En cuanto al coste, se realizará con una duración en tiempo de 3 meses, que viene dado por la duración del trabajo, además, para el aporte inicial de los equipos se estima un valor de 1000€ y para los elementos de oficina en 50€.

Los gastos totales vienen desglosados en la tabla 1.10

Concepto	Objetivo	Coste
Equipo de desarrollo	Desarrollo del proyecto y redacción de la documentación	62.5€
Otros Enseres	Anotación de ideas, mockups, explicaciones…	1,87€
Coste total:		64.37€

Tabla 1.10 Costes asociados a los recursos materiales

1.14.3 Costes asociados al personal

Para la realización de este proyecto el trabajo reside exclusivamente en el alumno, que asume diferentes roles durante el desarrollo. Los salarios base se han extraído *XVIII Convenio colectivo estatal de empresas de consultoría y estudios de mercado y de la opinión pública de 2023* (BOE, 2023).

Para este proyecto se han requerido los siguientes perfiles de trabajadores:

- Analista / Diseñador: Se encargará de todo el análisis y diseño del sistema. Perteneciente al Área 3, Grupo B, Nivel 2
- Jefe del proyecto: Encargado de supervisar y generar la documentación pertinente, además de transmitir los resultados. Perteneciente al Área 3, Grupo B, Nivel 2
- Supervisor de pruebas: Encargado de la experimentación y pruebas de usuario. Perteneciente al Área 4, Grupo D, Nivel 3.
- Diseñador gráfico: Diseña los elementos gráficos del videojuego. Perteneciente al Área 3, Grupo C, Nivel 3.

- Programador: Encargado de realizar la implementación del sistema, con los requisitos preestablecidos. Perteneciente al Area 3, Grupo C, Nivel 3.

Para realizar un calculo mas aproximado del gasto en sueldo de los trabajadores, se debe tener en cuenta el tiempo de dedicacion de cada rol al proyecto durante los 3 meses de desarrollo de trabajo. Para calcular el coste real de los trabajadores en este proyecto y no el coste de tener a estos tipos de trabajadores en una empresa.

Concepto	Sueldo bruto	Dedicación al proyecto	Coste
Analista/ Diseñador	2262.26 €	30%	2036.03 €
Jefe de proyecto	2262.26 €	20%	1357.36 €
Supervisor de pruebas	1472.66 €	20%	883.59 €
Diseñador gráfico	1875.26 €	35%	1969.02 €
Programador	1875.26 €	100%	5625.78 €
Coste total:			11871.78 €

Tabla 1.11 Costes asociados al personal

1.14.4 Costes finales

Una vez vistos todos los costes iniciales, es necesario considerar todos aquellos gastos indirectos asociados al proyecto. Son costes indirectos todos aquellos gastos extra asociados a cada proyecto y que son necesarios para llevar a cabo su realización.

En estos gastos se incluyen aquellos asociados al funcionamiento de la empresa como bien puede ser el alquiler de las oficinas, facturas de luz y agua, la conexión a internet, etc. Además, también implican aquellos gastos que son considerados como imprevistos.

Para el caso de este presupuesto se ha considerado destinar un 25% del coste inicial como gastos indirectos del proyecto.

Teniendo los gastos anteriormente descritos, el coste total del proyecto se puede desglosar en total en los gastos que podemos observar en la siguiente tabla.

Recurso	Coste total
Recursos software	0.00 €
Recursos materiales y equipamiento	64.37 €
Recursos personal	11871.78 €
Costes indirectos (25%)	2948.04 €
Coste total:	14884.19 €

Tabla 1.12 Costes finales del proyecto

1.14.5 Mecanismos de control de calidad

El aseguramiento de la calidad en la redacción del trabajo implica la verificación de la completitud (falta de omisiones), la integridad de la documentación, así como una redacción clara, concisa y entendible por todos los participantes e interesados en dicho trabajo.

Al estar el trabajo desarrollado por una sola persona y verificado por un/a tutor/a, no es necesario llevar un documento de control de edición y revisión de la documentación. De esta forma, se han utilizado mecanismos básicos para la verificación de la integridad y completitud, que incluyen un control de versiones a nivel de documento y un control sencillo de la trazabilidad de especificaciones y requisitos.

2 DISEÑO INICIAL

En esta sección se especificarán los aspectos referentes a la arquitectura propuesta y los modelos de diseño correspondiente a la funcionalidad, interfaces y datos.

2.1 Especificaciones del sistema

En este apartado se describirán los requisitos a mayor nivel, por lo tanto, se especificarán tanto los requisitos funcionales como los no funcionales, además también se incluirá los casos de uso.

2.1.1 Requisitos funcionales y no funcionales

En esta sección se mostrarán con más detalles los requisitos funcionales, que hablan de lo que hace el sistema y de los requisitos no funcionales que tratan sobre cómo debería hacerse (Medium, 2018).

- El usuario debe disponer de una interfaz en la que se le muestre las opciones necesarias para que pueda decidir si quiere realizar una partida en linea o local.
- El usuario debe disponer de una interfaz en la que se le permita seleccionar el tipo de juego que desee jugar.
- El usuario debe disponer de una interfaz en la partida que le permita realizar el cambio de vista 2D a vista 3D y viceversa.
- El usuario debe poder seleccionar una carta antes de ser lanzada, de esta forma servira para asegurarse que quiere lanzar esa carta y no se ha equivocado en la pulsacion que ha realizado.
- El usuario debe poder arrastrar las cartas hasta la zona indicada para que sea jugada, asi se le agregara dinamismo a la partida.
- Se debe de crear animaciones para que las cartas se muevan de su posicion incial que es el mazo, a la mano del jugador.
- Se debe de crear animaciones para que las cartas se muevan de la mano del jugador a la mesa.

- Se debe indicar una zona donde el usuario puede lanzar su carta, de esta forma sabra si la carta va a ser jugada o no.
- Si se esta arrastrando una carta y se coloca dentro de la zona indicada, la carta debera adoptar la posicion final para indicar al usuario que esta lista para ser colocada.
- Si la carta se encuetra colocada en su posicion final despues de haber sido arrastrada, se debe implementar un mecanismo para que el jugador pueda anular su jugada, el mas conveniente seria seguir arrastrando el mouse o el touch para que la carta vuelva a la mano del jugador.
- Se debe implementar un metodo para ordenar las cartas de la mano del jugador segun el tipo de juego.
- Se debe implmentar un metodo encargado de barajar las cartas.
- Se deben implementar varios metodos encargados de repartir las cartas. Como por ejemplo un reparto aleternativo, un reparto uniforme…etc.

2.2 Análisis y diseño del sistema

En este apartado se describen los distintos elementos del videojuego y como han sido diseñados para su correcto funcionamiento, además de realizar un análisis general sobre el videojuego.

2.2.1 Mecánica del juego

El principal objetivo del videojuego es vencer a los demás oponentes en una partida cartas, ya sean oponentes humanos, u oponentes manejados por la propia inteligencia de la plataforma de agentes.

El juego está compuesto por diferentes escena las cuales le indican al jugador en qué momento se encuentra, puede ser seleccionando el juego que desee jugar, esperando a que comience la partida o jugando una partida de cartas.

Al ser un prototipo solamente se han implementado dos juegos de cartas, que son el Cinquillo y la Brisca, todo ello con la finalidad de demostrar la escalabilidad del juego, en cuanto a la forma de poder desarrollar partidas con distintas reglas, dependiendo el tipo de juego seleccionado.

2.2.1.1 Criterios de éxito y fracaso

Todo videojuego tiene criterios de éxito y fracaso, y este no es menos, aunque si bien es cierto que el éxito o fracaso no es tan determinante como en otro tipo de juegos.

Para el criterio de éxito el jugador deberá ganar la partida a la que ha decidido jugar, ello lo sabrá al finalizar la partida y observando si su nombre se encuentra situado en la parte superior de la clasificación. Los puntos obtenidos dependerán del tipo de juego.

Por otro lado el criterio de fracaso es antagonista al de victoria, si no se ha conseguido ganar la partida el jugador no se encontrara en la parte superior de la clasificación.

2.2.1.2 Conjunto de reglas que dominan el juego

Para establecer una buena experiencia de usuario y diversión dentro del videojuego se ha considerado el establecimiento de las siguientes reglas

- Cada jugador dispone de un tiempo limitado para realizar un movimiento, esto mejorar la experiencia de usuario y hará que las partidas no tengan una duración excesiva, si el movimiento no se ha realizado en el tiempo establecido se realizara un movimiento aleatorio.
- Solamente se podrán realizar movimientos validos dependiendo el tipo de juego con el fin de mantener la integridad y la diversión de las partidas de jugar a un juego que todos conocen sus reglas

2.2.2 Diseño de base de datos

El diseño de base de datos es un aspecto muy importante ya que un buen diseño mejora la eficiencia y velocidad de la aplicación, así como en nuestro caso también ahorro de costes, almacenando solamente la información necesaria.

El diseño se ha realizado sobre una base de datos de tipo NoSQL (Oracle, s.f.), dado que esta es el tipo que utiliza Real Time Firebase. Por lo tanto lo primero que se debe hacer es identificar los requisitos que requiere para el desarrollo de una partida.

- Matchmaking: El jugador puede apuntarse a cualquier juego y permacener a la espera hasta que se cumpla con la cantidad minima de jugadores para desarrollar la partida.
- Reparto Inicial: El jugador debe recibir las cartas con las que comenzara el juego, asi como la informacion necesaria para el reparto inicial, como por ejemplo la carta del triunfo.
- Movimientos: El jugador debe poder publicar los movimientos realizados para informar al resto de jugadores, asi como leer el movimiento de los demas para actualizarlo en su partida
- Resultado: El jugador debe recibir el resultado de Firebase para mostrarlo en su propia instancia.

Una vez identificado los requisitos tenemos que establecer un diseño que estos se lleven a cabo, el diseño se mostrara en formato de documento que es la forma de trabajar de este tipo de bases de datos, los documentos usualmente se suelen trabajar en formato JSON (IBM, s.f.), es por ello que se mostrara la estructura en este formato.

Comenzando por el matchmaking necesitamos tener la información sobre el juego que quiere apuntarse el jugador, una vez sabido esto se le deberá informar al jugador si ha sido seleccionado para comenzar una partida o no. Esta información se

almacenara dentro del apartado *Matchs*, que vemos en la ilustración, aquí se publicaran todas las partidas generadas por el matchmaking, el jugador leerá la información y si se encuentra dentro de esta partida la almacena y comienza la partida. Además se necesita saber los jugadores que se encuentran en la espera para comenzar una partida, estos se almacenan dentro del apartado *Players*, cuando entran a una partida estos jugadores son eliminados de esta sección del documento.

Ilustración 2.1 Jerarquía de documentos para el matchmaking

Como se puede observar el resto de requerimientos todos están enfocados al transcurso de la partida por lo que se englobaran todos dentro de una estructura que identifique unívocamente a todas las partidas que se encuentran en desarrollo. Por lo

tanto el resto de información se almacenara dentro del documento que se muestra en la ilustración

Ilustración 2.2 Jerarquía de documentos para desarrollar un juego

Con la estructura que nos identifica la partida que se está llevando a cabo debemos generar una estructura que contenga todos los datos necesarios para comenzar la partida. En esta sección debemos incluir una lista de cartas asignadas a cada jugador, que será el reparto inicial, la carta del triunfo para los juegos que sean

necesarios, como por ejemplo la Brisca, la información de los jugadores para saber que todos están jugando la partida, y el tipo de juego que se está llevando a cabo, el cual es necesario para actualizar los movimientos en otras secciones del documento. Toda esta información se ha recogido de la forma que vemos en la siguiente ilustración

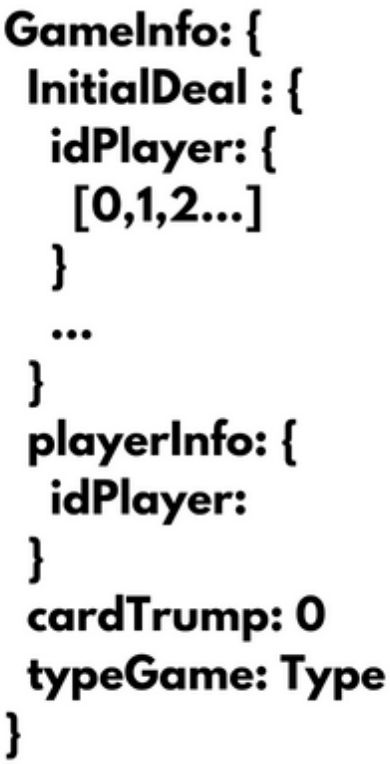

Ilustración 2.3 Jerarquía de documentos con la información del juego

Además de esta información necesitamos tener un lugar donde se publiquen todos los movimientos de la partida, para que cada jugador obtenga únicamente el movimiento, sin necesidad de recibir información innecesaria. En el movimiento se incluirá el tipo de movimiento para actualizarlo en la propia instancia del jugador y en caso de que el movimiento sea de lanzar carta también contiene la información de la

carta que ha lanzado el jugador. Esta estructura de un movimiento se puede ver en la siguiente imagen.

Ilustración 2.4 Jerarquía de documentos para informar de los movimientos

Por ultimo tenemos que informar sobre el resultado de la partida a los jugadores de la misma, pero para ello necesitamos la información de toda la partida. Se ha creído más conveniente almacenar esta información en otro documento dentro de la partida, en vez de utilizar las funciones de Google Cloud como almacenamiento. Por ello se ha generado las siguientes estructuras con las cartas lanzadas por cada jugador, las cartas que ha ganado cada jugador, para juegos como la brisca y el reparto de cartas

a cada jugador, todas ellas tienen la misma estructura en el documento, la información se almacena bajo el identificador del jugador. Además en esta sección también se almacenara el mazo para las operaciones de robar carta que hagan los jugadores, junto con un contador de movimientos realizados por los jugadores, de esta forma se puede saber cuándo debe de comenzar el robo de cartas y marcar las operaciones a realizar dependiendo del tipo de juego.

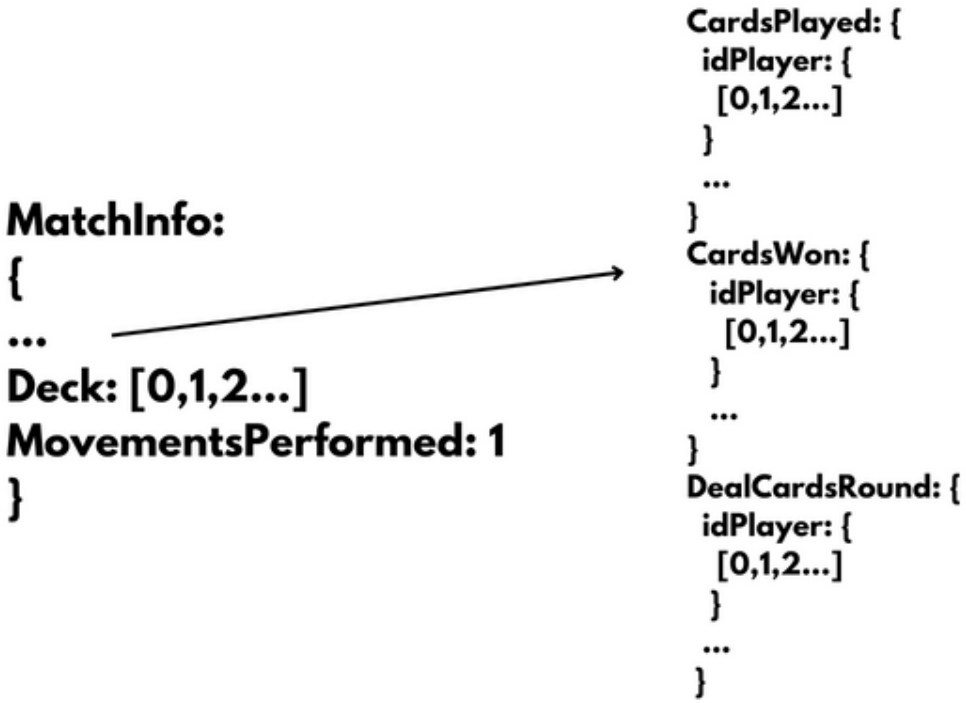

Ilustración 2.5 Jerarquía de documentos para almacenar información sobre la partida

Con esta estructura que se irá actualizando conforme se vaya desarrollando la partida ya podemos generar la clasificación haciendo peticiones desde Google Cloud a dicha estructura, el resultado de la partida seguirá la estructura del documento que vemos en la siguiente ilustración. El cual está conformado por la lista de jugadores de la partida y la puntuación obtenida.

Ilustración 2.6 Jerarquía de documentos para el resultado de la partida

2.2.3 Análisis de la ontología de agentes

La palabra ontología tiene varias definiciones, es un elemento filosófico y significa la descripción del ente. Una ontología debe de ser clara, coherente y extensible. Dichas ontologías pueden ser implementadas en muchos lenguajes, para el caso de los agentes se utilizara el lenguaje Semantic Lenguaje (SL), que es el que utiliza la plataforma JADE. Todo lenguaje tiene un vocabulario, que son los elementos que conforman dicho lenguaje, el símil con la programación es una clase encargada del almacenar conceptos, esta explicación es necesaria para aclarar cuando nos refiramos a estos conceptos dentro del apartado.

2.2.3.1 Análisis de clases

Para realizar las transformaciones la biblioteca Jade proporciona las definiciones presentes en el paquete jade.content. En la siguiente imagen se presenta la jerarquía de los elementos que se debe utilizar para el uso de las ontologías con los agentes Jade.

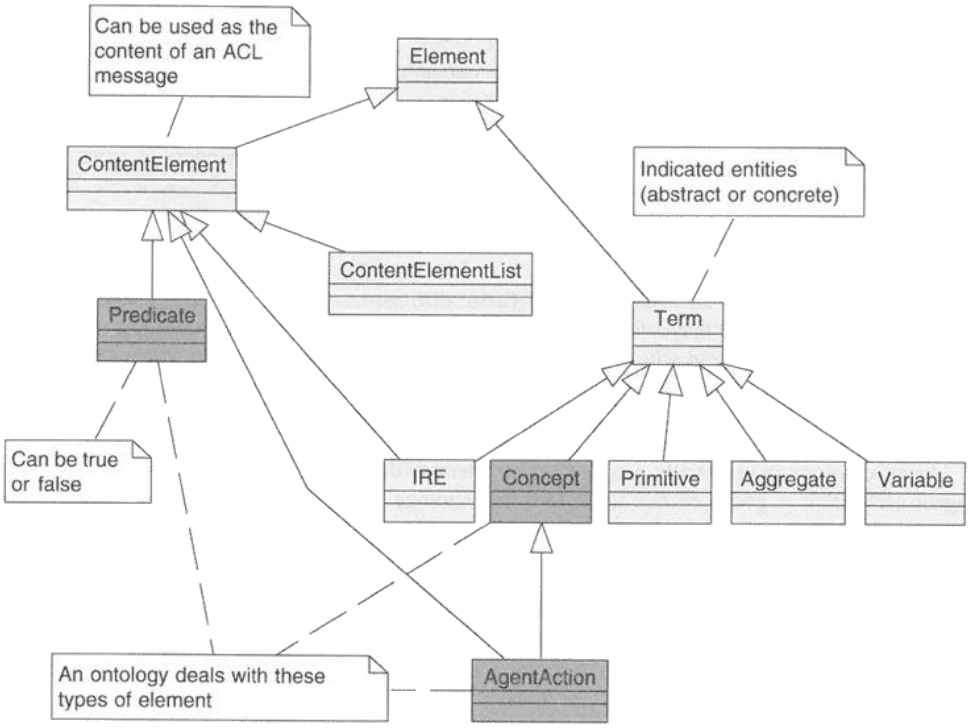

Ilustración 2.7 Esquema estructural del paquete jade.content

2.2.3.1.1 Conceptos de la ontología

Un concepto es una entidad que puede definirse por sus características. En el ámbito del código son un conjunto de atributos almacenados dentro de una clase que aportan información a la ontología.

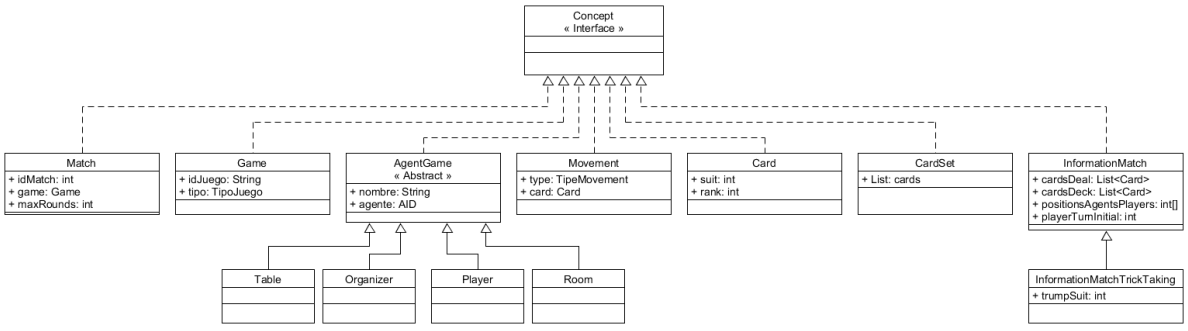

Ilustración 2.8 Diagrama de clases de los conceptos de la ontología

2.2.3.1.2 Predicados de la ontología

Los predicados son expresiones que dan información sobre el estado del mundo, las cuales pueden ser verdaderas o falsas. En el ámbito del código serán las clases que se envíen como respuestas a las peticiones de acción de un agente.

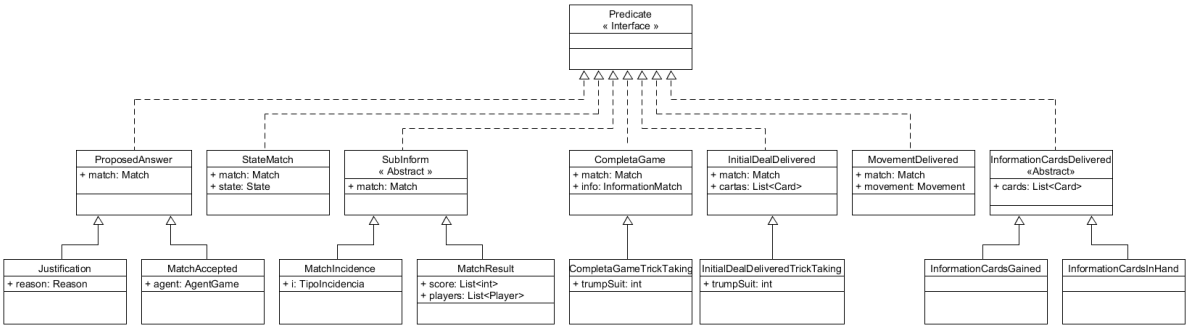

Ilustración 2.9 Diagrama de clases de los predicados de la ontología

2.2.3.1.3 Acciones de la ontología

Las acciones son un tipo de concepto especial. En el ámbito del código serán las clases encargadas de realizar peticiones de acciones al resto de agentes.

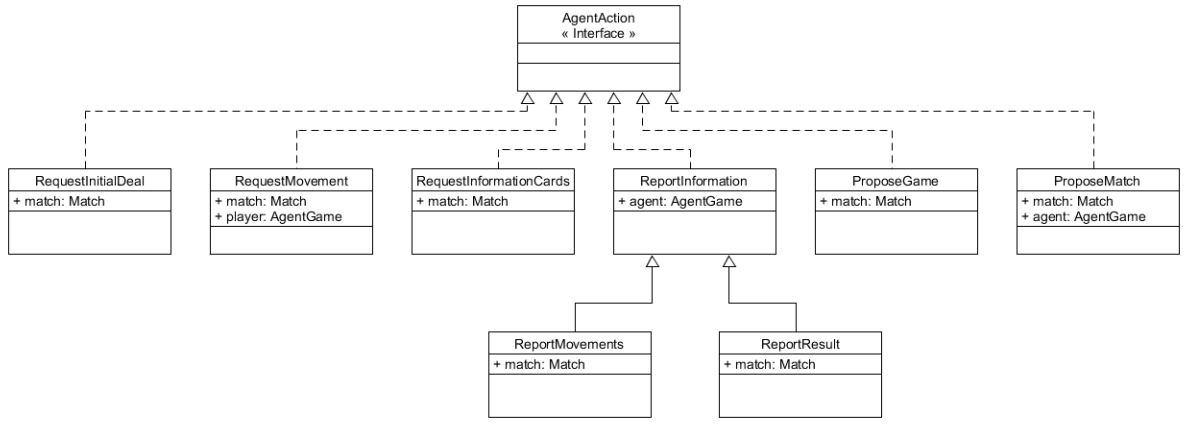

Ilustración 2.10 Diagrama de clases de las acciones de la ontología

2.2.3.2 Análisis de la ontología

La ontología debe responder apropiadamente a las siguientes preguntas:

- ¿Cómo diferenciar a los diferentes agentes especializados?

- ¿Cómo proponer a los diferentes jugadores que participen en un juego?
- ¿Cómo debe completarse un juego?
- ¿Cómo completar un turno de una partida?
- ¿Cómo completar la partida?
- ¿Cómo informar del resultado final de la partida?

Estas preguntas van a necesitar que se intercambie información entre los agentes de la plataforma. Para resolver las diferentes preguntas se presentan los diagramas “AUML” para la secuencia de mensajes que deben intercambiarse entre los agentes implicados. Estos diagramas son una modificación respecto a los diagramas de secuencia habituales ya que están pensados para la representación de agentes.

2.2.3.2.1 ¿Cómo diferenciar a los diferentes agentes especializados?

Para resolver la primera pregunta se utilizara la utilidad del servicio de páginas amarillas que proporciona la plataforma de agentes. De esta manera no será necesario implementar elementos de ontología para este problema de comunicación, sin embargo si será necesario elementos de vocabulario para que los agentes puedan suscribirse al servicio de forma homogénea.

- TypeService: Los agentes especializados tendrán asociado un tipo de servicio y utilizan este elemento del vocabulario. Los agentes especializados serán
  - ORGANIZER: Representa el tipo de servicio para cualquier agente organizador.
  - PLAYER: Representa el tipo de servicio para cualquier agente jugador.

El *AgentOrganizer* especializado necesita ser registrado para que los *AgentTable* cuando sean creados sean capaces de encontrarlo y comunicarse con el directamente para la recepcion de movmientos, por otro lado los *AgentPlayer* deben ser visibles siempre para el resto de agentes sala, para llevar a cabo las partidas.

2.2.3.2.2 ¿Cómo proponer a los diferentes jugadores que participen en un juego?

El *AgentRoom* será el encargado de proponer los juegos a los *AgentPlayer*. El juego debe identificarse de forma univoca y el agente puede tomar la decisión de no participar en el juego si se cumple una determinada condición, todo ello con finalidad de dotar de inteligencia a este tipo de agentes.

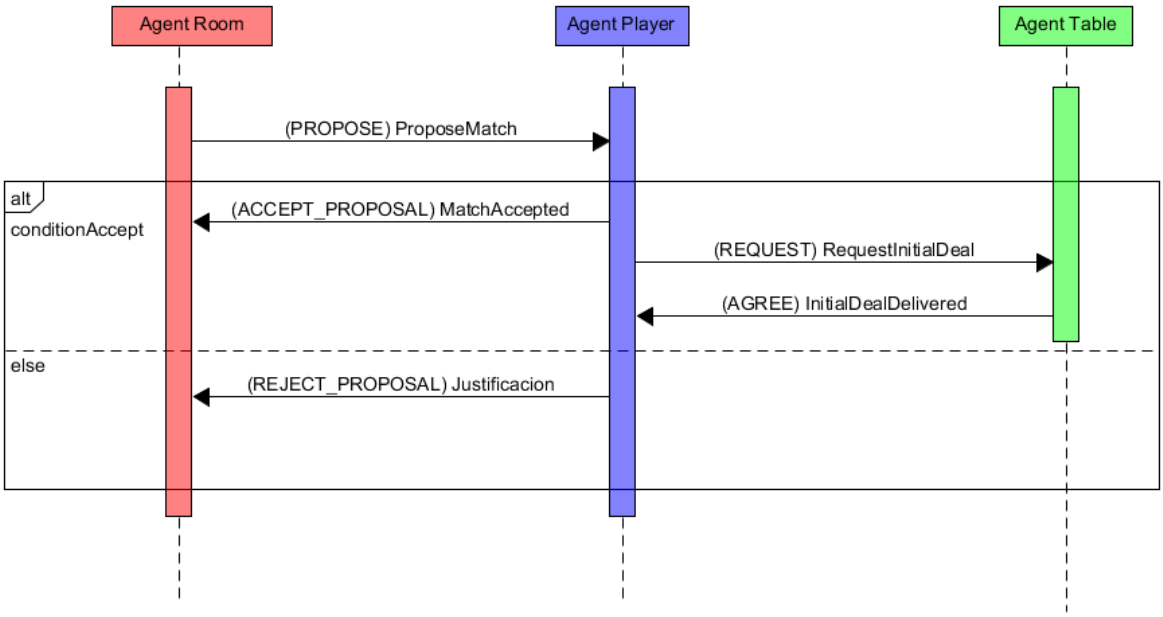

Ilustración 2.11 Esquema de comunicación para la propuesta de un juego

2.2.3.2.3 ¿Cómo debe completarse un juego?

El *AgentOrganizer* debe crear un *AgentRoom* para que le lleve a cabo el juego enviado por la plataforma de Unity, el *AgentOrganizer* almacenara todos los *AgentRoom* que haya creada, por si rechazan la petición de completar el juego deberá crear otro para que lleve a cabo la partida.

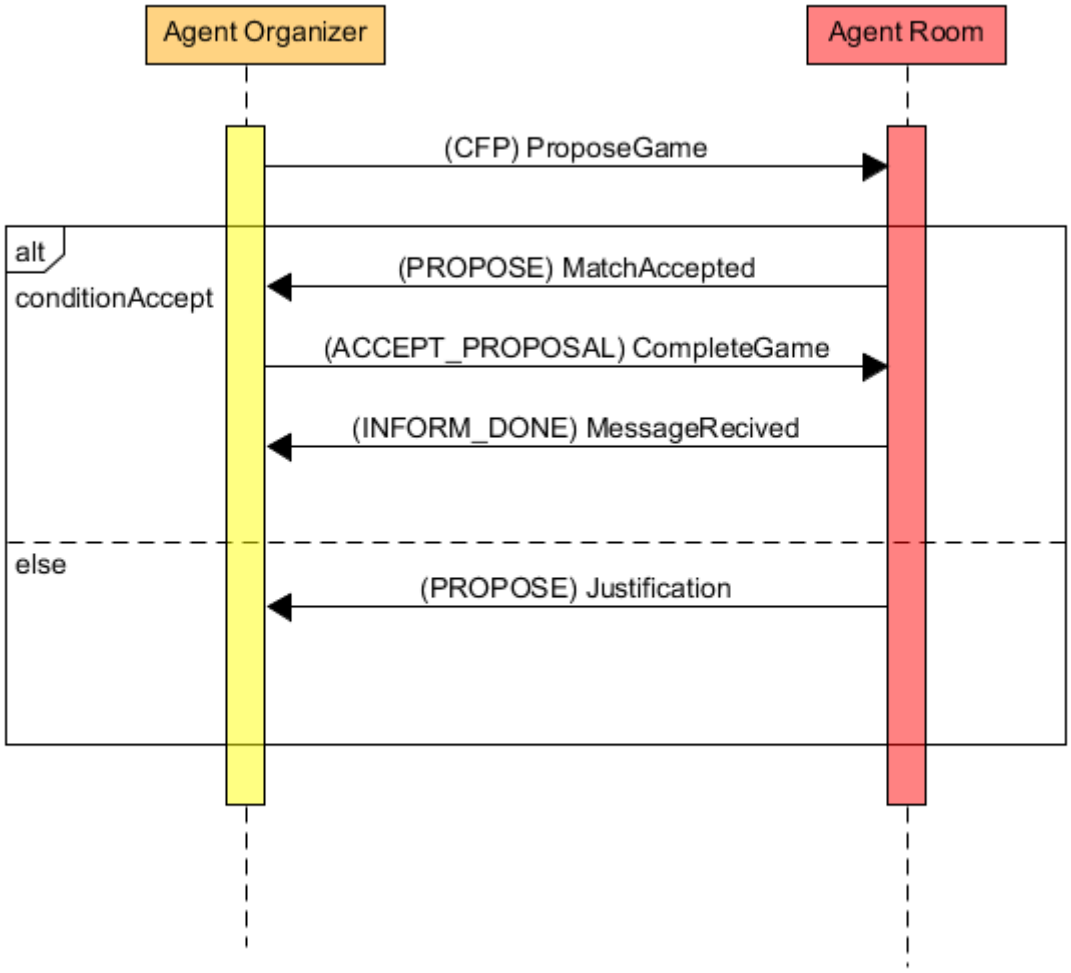

Ilustración 2.12 Esquema de comunicación para completar un juego

2.2.3.2.4 ¿Cómo completar un turno de una partida?

El *AgentTable* es el encargado organizar los turnos que componen una partida. Por lo tanto, este agente será el encargado de realizar peticiones a los *AgentPlayer* por los movimientos de la partida. Dicho esto lo importante es identificar los elementos de información necesario para desarrollar turnos.

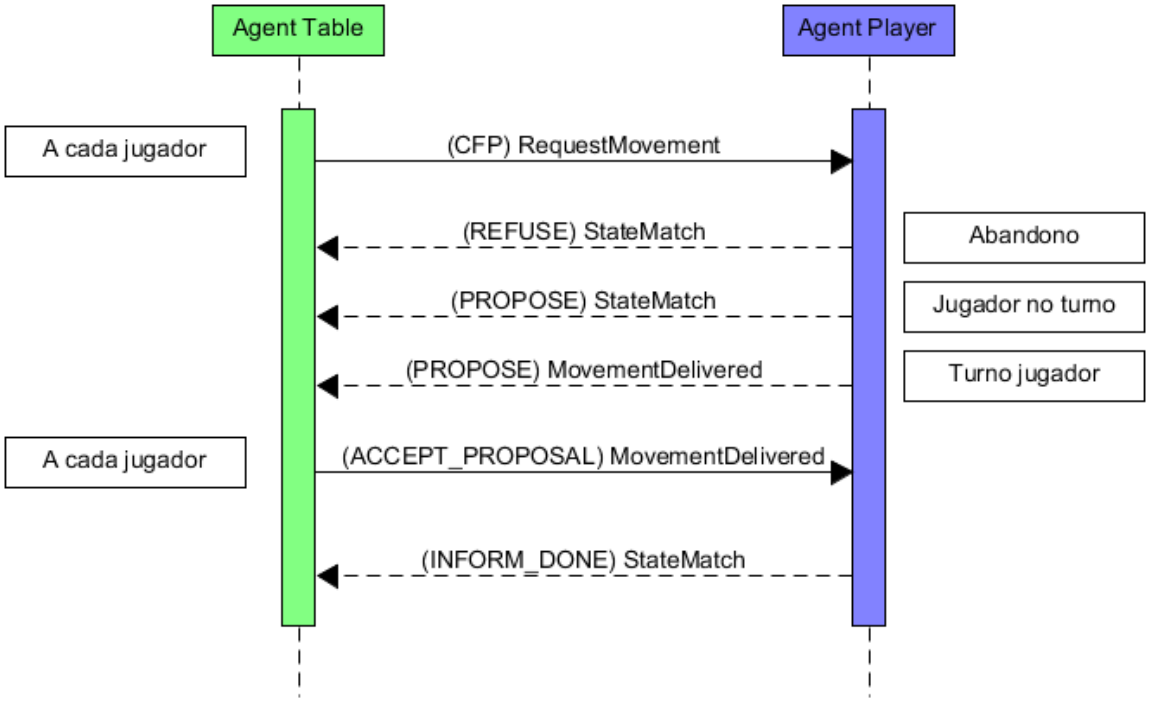

Ilustración 2.13 Esquema de comunicación para desarrollar un turno de juego

2.2.3.2.5 ¿Cómo completar la partida?

Para completar la partida hay que ir generando los turnos necesarios hasta que los *AgentPlayer*, los cuales tienen la inteligencia necesaria para saber cuándo una partida ha finalizado, envíen un mensaje con el estado de la finalización de la partida. Una vez identificado este final de partida se deberá establecer una comunicación para obtener los datos necesarios por parte del *AgentTable* para generar la clasificación.

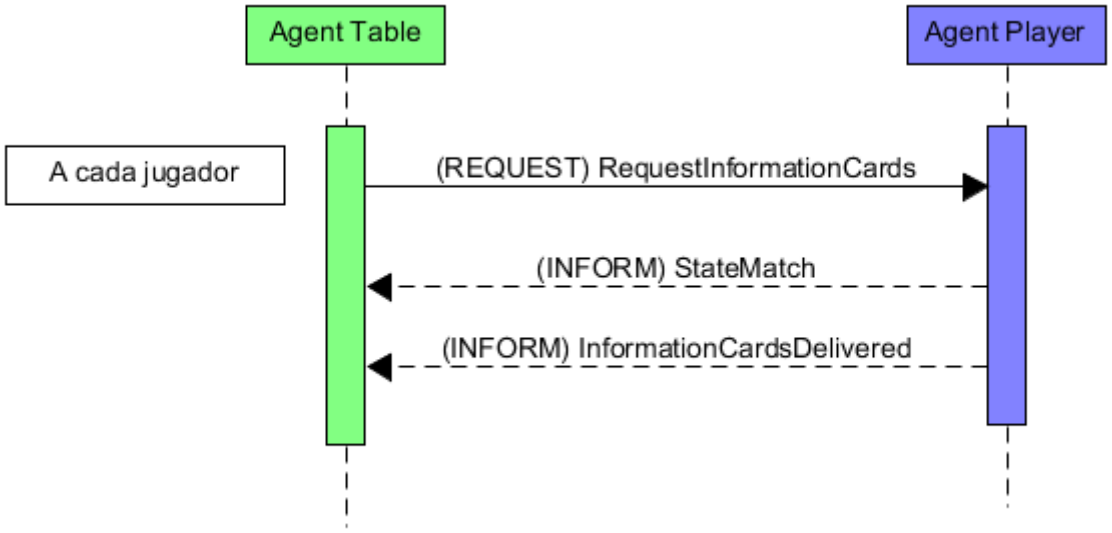

Ilustración 2.14 Esquema de comunicación para completar una partida

2.2.3.2.6 ¿Cómo informar del resultado final de la partida?

Todos los agentes relacionados con el desarrollo de la partida deben ser informados con el resultado de la partida, los *AgentPlayer* deben conocer el resultado con la finalidad de poder mejorar sus heurísticas a la hora de realizar movimientos para la siguiente partida, el *AgentRoom* debe saber cuándo ha finalizado la partida para notificar del resultado al *AgentOrganizer*, el cual mandara la información de la clasificación a la plataforma de Unity.

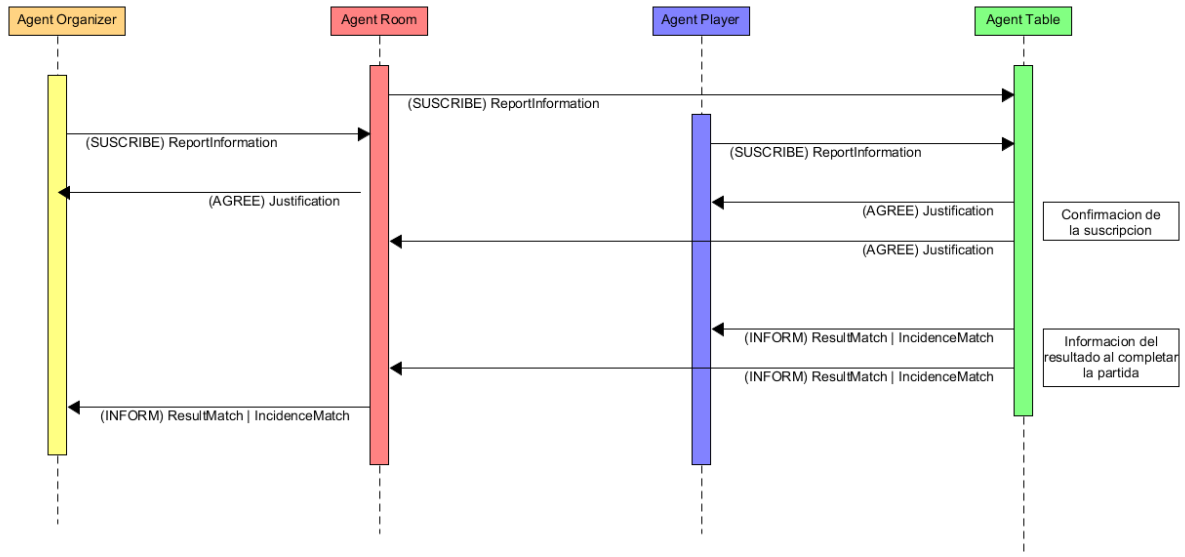

Ilustración 2.15 Esquema de comunicación para informar del resultado del juego

2.2.4 Diseño de la interfaz

Cuando se trata de aplicaciones de software, la interfaz de usuario es una de las partes más importantes y donde pasa la mayor parte del tiempo, ya que puede marcar la diferencia entre el éxito y el fracaso de la propia aplicación. Se está volviendo cada vez más importante en los videojuegos, ya que un diseño deficiente puede romper la inmersión del jugador y hacer que la experiencia sea desagradable.

El primer paso a la hora de desarrollar una interfaz es identificar las pantallas que se mostraran en la aplicación.

- Inicio de sesión.
- Registro.
- Perfil de usuario.

- Seleccion modo de juego.
- Seleccion tipo de juego.
- Sesion de juego.
- Clasificacion del juego

Una vez identificadas las pantallas, el siguiente paso es plasmar la información que contendrán las mismas mediante un boceto.

2.2.4.1 Inicio de sesión

Esta será la primera interfaz que el usuario vea cuando inicie el juego por primera vez, en ella el jugador deberá decidir entre iniciar sesión con una cuenta previamente creada o en entrar como invitado a jugar partidas de cartas. Para ello dispone de dos campos de texto, para introducir los datos de inicio de sesión y de los botones con textos descriptivos para formalizar las acciones.

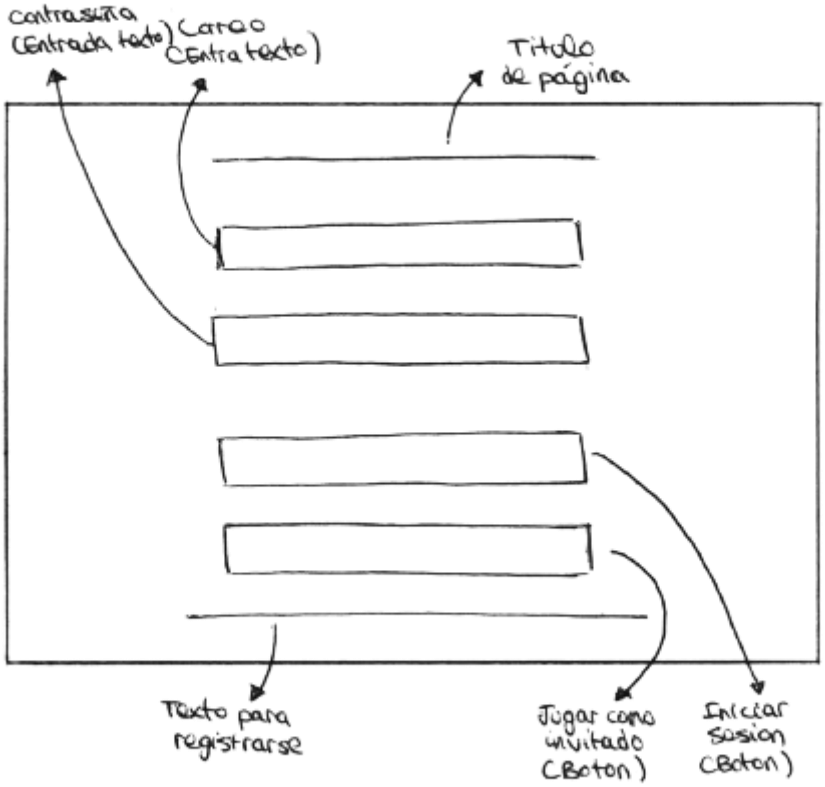

Ilustración 2.16 Diseño de la pantalla de inicio de sesión

2.2.4.2 Registro

Esta interfaz es similar a la de inicio de sesión, la única diferencia es que a esta interfaz se debe acceder si no se tiene ninguna cuenta y queremos crear una. Para agregar más seguridad a la hora de registrarse se le pedirá al usuario que introduzca

su contraseña para el registro dos veces, estos campos de entrada deben de coincidir, si no el registro no se realizara.

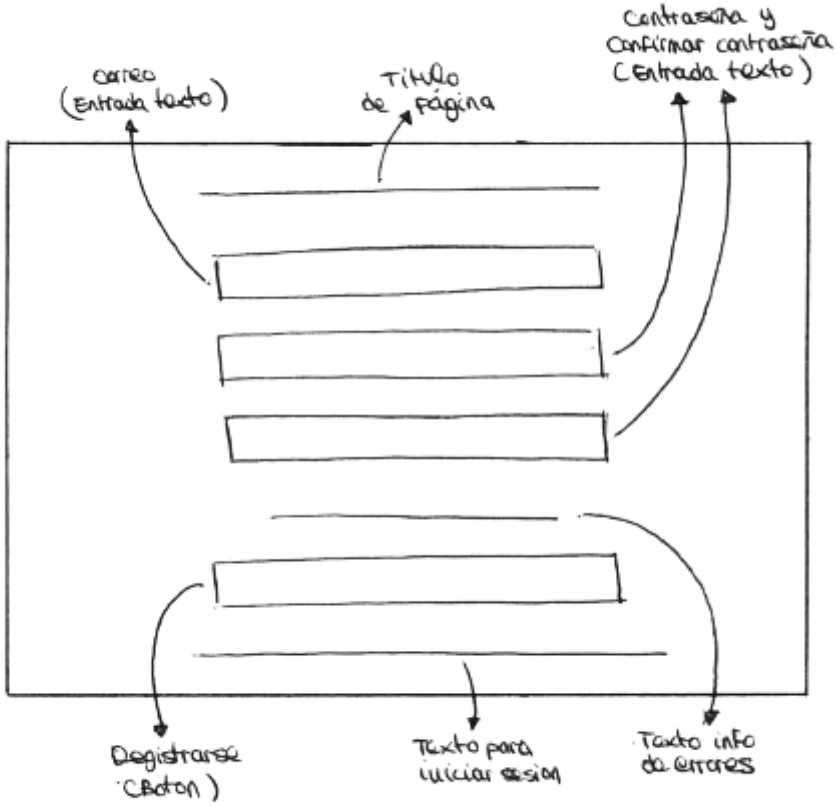

Ilustración 2.17 Diseño de la pantalla de registro

2.2.4.3 Selección modo de juego

Esta es la primera interfaz que vera el usuario después de haber iniciado sesión en el videojuego, en ella se permitirá seleccionar entre las distintas modalidades de las partidas, estos botones ocuparan la mayor parte de la interfaz para incitar al usuario a clicarlos primero, dado que la finalidad es la de jugar a los juegos de cartas.

Por otro lado desde esta página principal se podrá acceder a la interfaz del perfil de usuario a través del botón que se indica en la ilustración 2.18

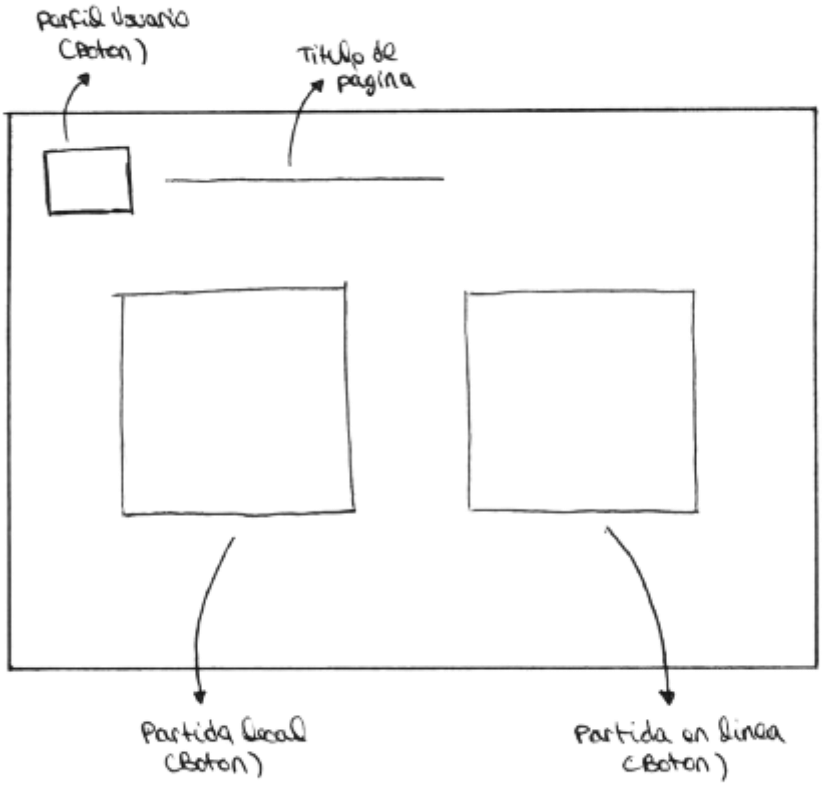

Ilustración 2.18 Diseño de la pantalla para la selección de modo de juego

2.2.4.4 Perfil de usuario

En el perfil de usuario permitirá al jugador ver información sobre su propio jugador en las sesiones de juego. Además esta interfaz dispone de un campo de entrada de texto para permitir al usuario cambiar su nombre en las partidas.

Con la finalidad de dar al usuario conciencia sobre la navegación de páginas se le permitirá volver a través de un botón de la interfaz, de esta forma será mucho más comprensible comprender para el usuario por donde esta navegando

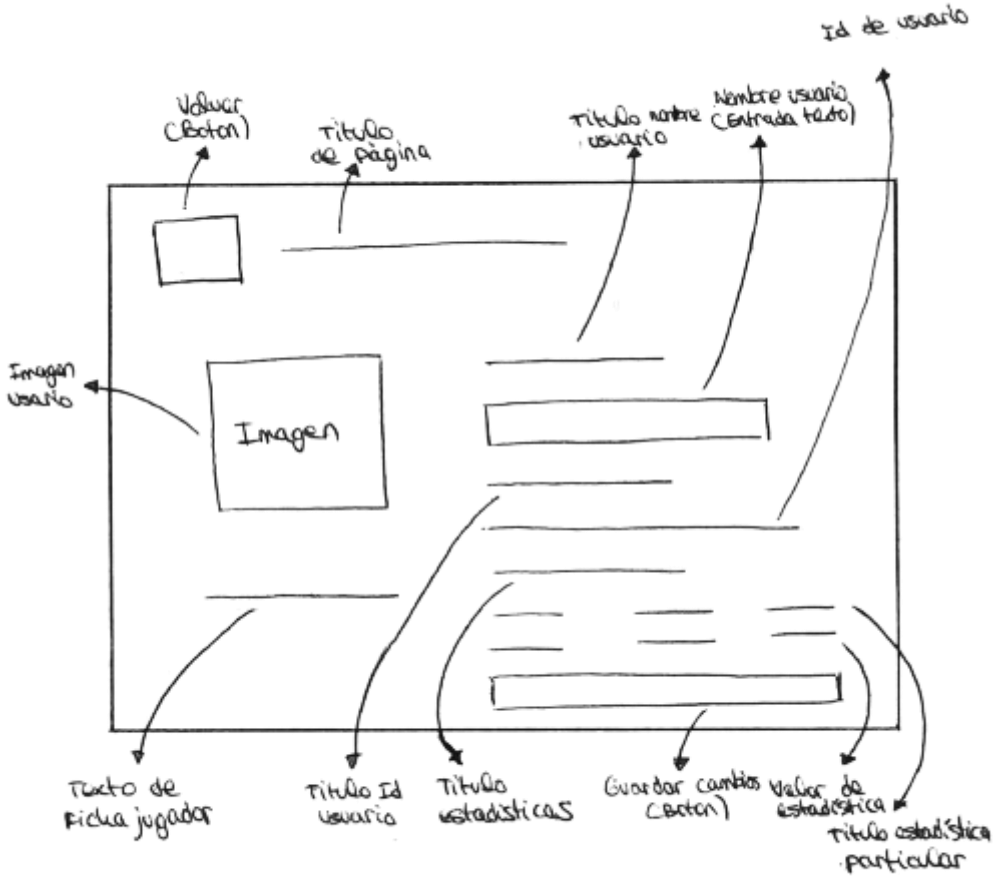

Ilustración 2.19 Diseño de la pantalla del perfil de usuario

2.2.4.5 Selección tipo de juego

En esta pantalla el jugador seleccionara entre los juegos disponibles para comenzar una partida, siempre estará informado del juego que va a entrar gracias al texto inferior.

Al ser accedida desde la interfaz de la selección de modo de juego también se le dará la opción al usuario de volver, por si desea cambiar la modalidad del tipo de juego.

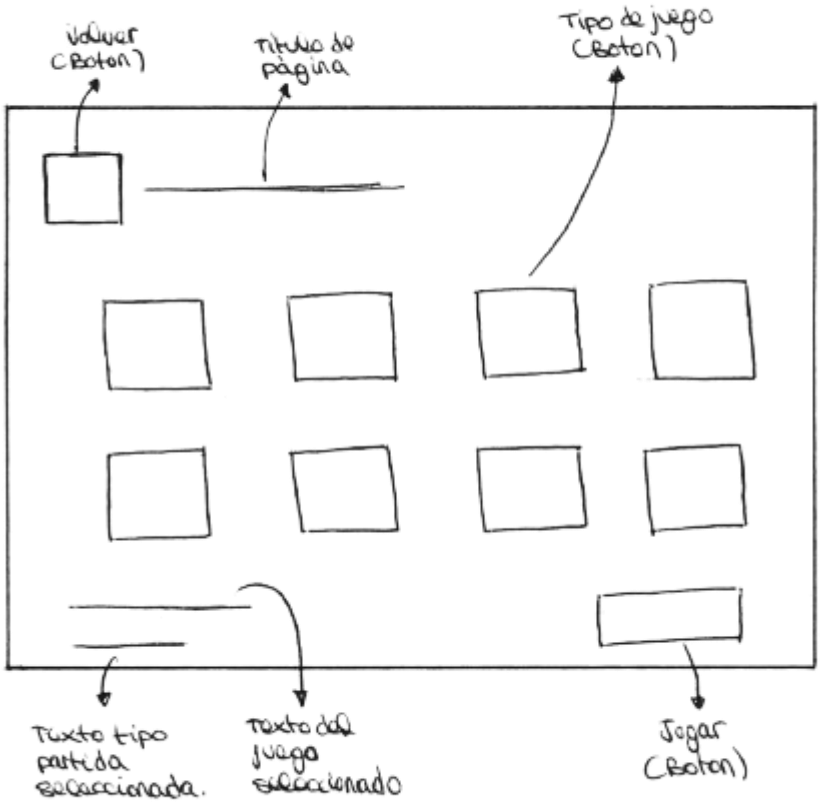

Ilustración 2.20 Diseño de la pantalla de selección de tipo de juego.

2.2.4.6 Sesión de juego

Para la sesión de juego se han realizado bocetos para dos interfaces posibles, ya que como se habló en apartados anteriores, el juego permite modificar de la vista 2D a la vista 3D, y viceversa durante el transcurso de la partida. En la ilustración 2.21 se ha plasmado como se colocaría las cartas dependiendo de la vista que haya seleccionado el usuario.

En la sesión de juego el jugador vera dos botones, uno para realizar estos cambios de vista y otro para cambiar las imágenes de las cartas.

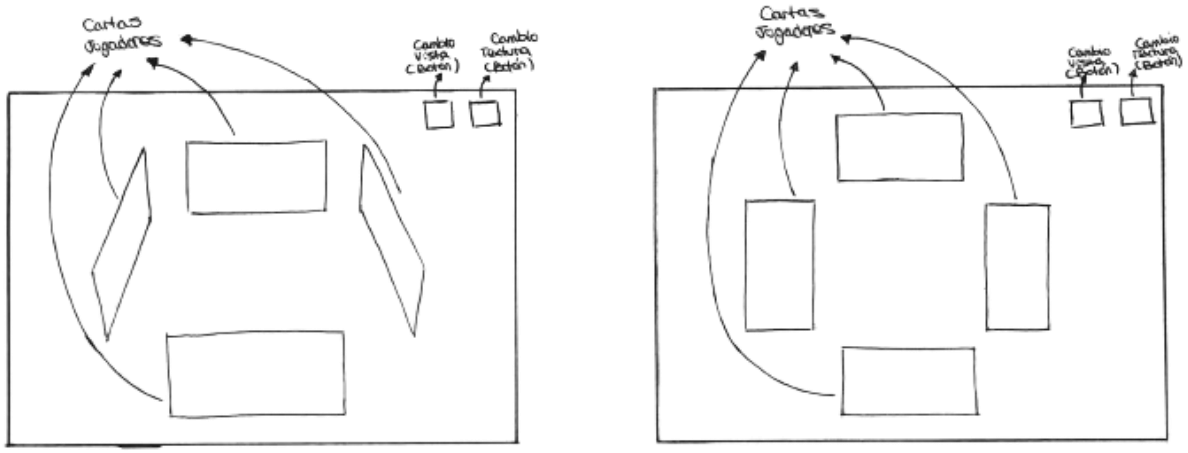

Ilustración 2.21 Diseño de las pantallas de la sesión de juego

2.2.4.7 Clasificación del juego

El jugador deberá de conocer el resultado de la partida a través de la interfaz gráfica, es por ello que el resultado consistirá en un panel que se superponga a la vista de la sesión de juego, cuando finalice el propio juego. El boceto correspondiente a este panel de clasificación lo podemos observar en la siguiente imagen.

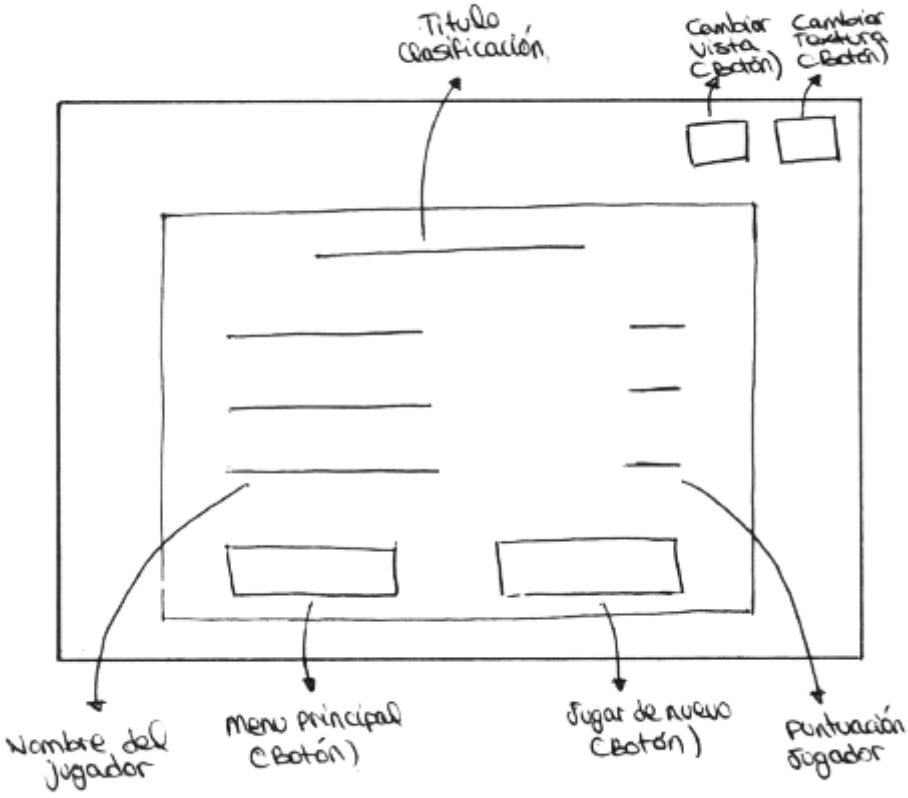

Ilustración 2.22 Diseño de la pantalla de clasificación

Al finalizar la partida la interfaz de usuario dispondrá de dos botones, uno para volver a jugar una partida del mismo tipo y otro para volver al menú principal del juego. De esta forma se cerrara el ciclo de interacción con las interfaces de usuario

3 DESARROLLO

En un modelo iterativo, como se está tratando en este proyecto, las primeras iteraciones suelen ser las que más trabajo incluyen, ya que se dedican a desarrollar la lógica principal de las aplicaciones.

En este apartado se procede a explicar los elementos desarrollados en cada una de las iteraciones. En el caso de este proyecto, se tienen las siguientes iteraciones

3.1 Primera Iteración

En esta primera iteración se expondrá la puesta en marcha e implementación de los elementos troncales del sistema, como pueden ser manejadores e instalaciones de configuraciones iniciales.

3.1.1 Instalando Universal Render Pipeline

Lo primero que se ha realizado en el proyecto es tomar la decisión sobre el pipeline de renderización del videojuego, este aspecto es importante porque determinara para que posibles plataformas a las que puede llegar nuestro proyecto.

Para la elección del pipeline Unity nos proporciona información sobre los distintos pipeline en los cuales podemos implementar y cuáles son sus ventajas e inconvenientes (Unity, Unity - Best Practise Render Pipeline, s.f.). Como buscamos la mayor escalabilidad y eficiencia posible se implementara utilizando URP, el cual está pensado para realizar juegos en sistemas de baja potencia como dispositivos móviles, tabletas y además también nos permite que los juegos se puedan ejecutar en todas las plataformas.

Para instalar Universal Render Pipeline en Unity hemos seguido la documentación que nos proporciona el propio motor de videojuegos (Unity, Universal Render Pipeline, s.f.).

3.1.2 Modelando las cartas

El modelo de carta se ha decido realizarse desde cero, la principal razón es que no había modelos en internet que se ajustaran a las necesidades de escalabilidad que estaban previstas, ya que se buscaba que la parte trasera de la carta fuera independiente a la parte delantera para poder combinar texturas tanto frontales como

traseras de distintos tipos de cartas. El camino a seguir para el modelado de la carta ha sido el siguiente. Primero se ha escalado el cubo inicial que nos proporciona Blender (Blender, Blender, s.f.) hasta dejarlo con la forma de una carta, como se puede ver en la ilustración.

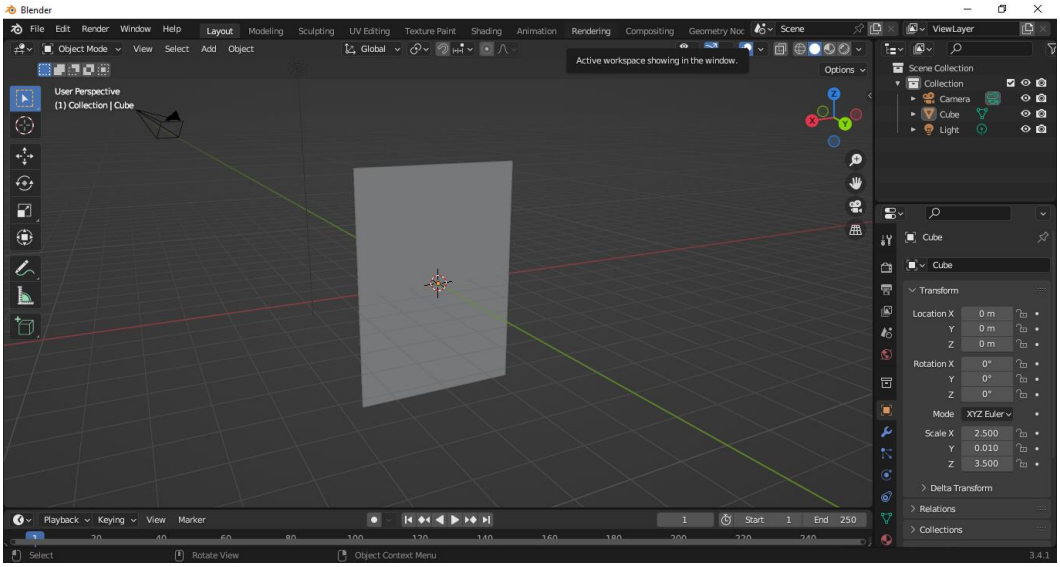

Ilustración 3.1 Captura de la primera forma de la carta

Una vez realizado el escalado redondearemos las esquinas para que no sea un modelo con unos bordes tan bruscos, para ello usaremos la herramienta de *bevel* (Blender, Blender | Bevel, s.f.) dentro de Blender.

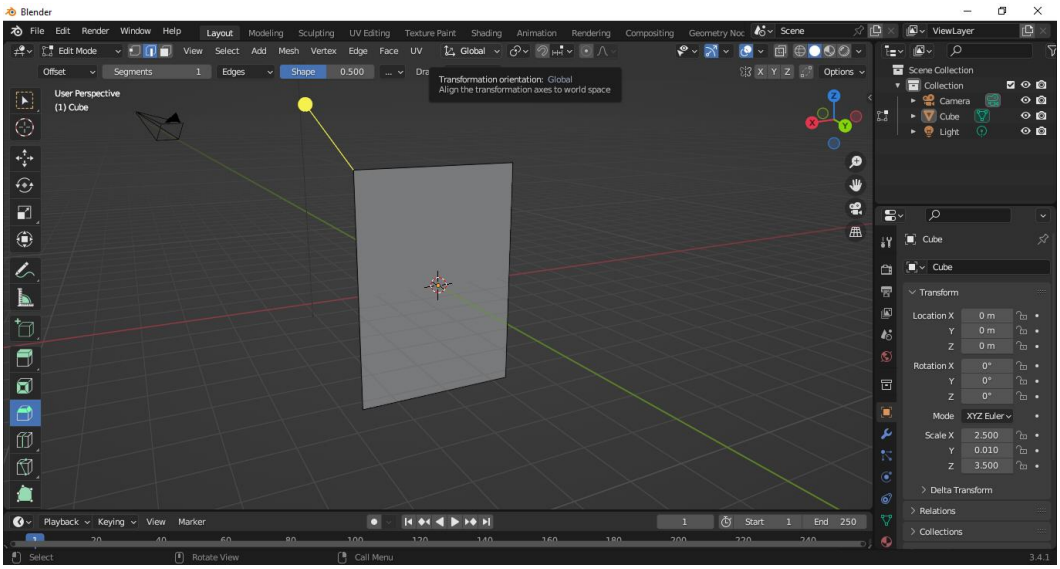

Ilustración 3.2 Captura de la carta con los bordes redondeados

Por ultimo con la forma de la carta ya realizada solamente nos queda asignarle un material a la parte delantera y modificar las coordenadas de textura para que ocupen toda la superficie de la carta, una vez hecho esto ya tendremos el modelo de carta finalizado.

El siguiente paso será importarlo a Unity, lo cual es bastante simple. En este caso hemos creado el modelo en formato FBX, este archivo tendremos que arrastrarlo dentro del inspector del proyecto de Unity y se importara automáticamente.

3.1.3 Agregando texturas a las cartas

Para la optimización en el uso de texturas se ha creído conveniente crear un TextureArray en Unity, el cual permitirá en una sola imagen almacenar todas las texturas de cada carta, ahorrando así en almacenamiento en el proyecto.

La implementación de un TextureArray en Unity se compone de dos pasos, el primero consiste en encontrar una textura que contenga todas las imágenes de la baraja de carta, como podemos ver en la ilustración 3.3.

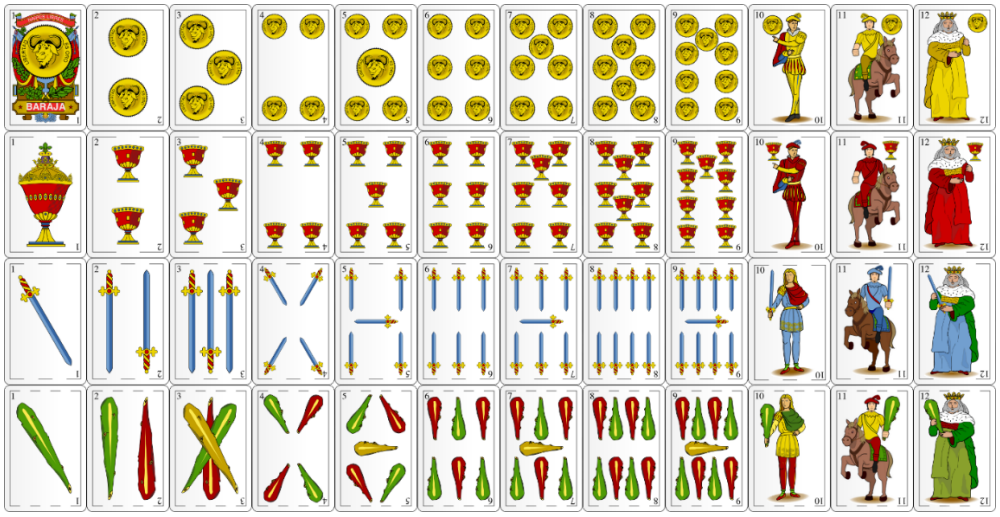

Ilustración 3.3 Texturas de cartas utilizadas

Con la imagen ya seleccionada debemos importarla a Unity indicando que vamos a utilizar esta textura como un TextureArray (Unity, Unity - Texture Array, s.f.), en ella le indicaremos las filas y columnas para que divida las coordenadas de textura correctamente.

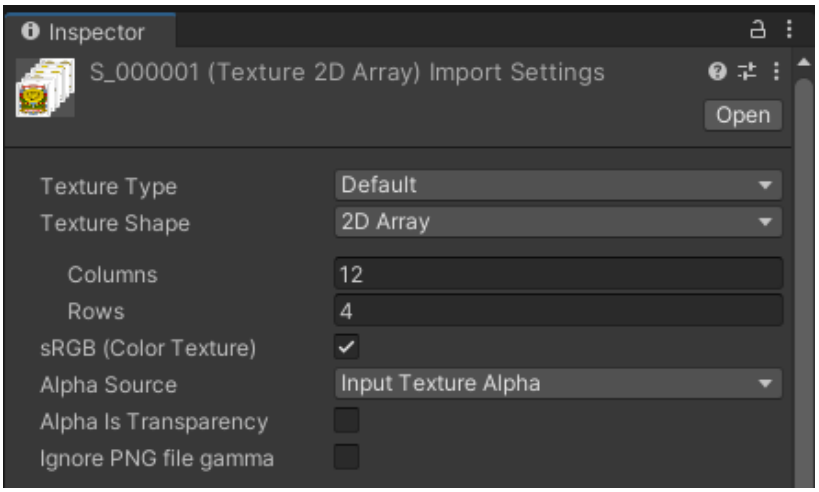

Ilustración 3.4 Captura configuración Texture Array en Unity

Con la textura ya importada para utilizarla tendremos que codificar un shader (Unity, Unity Shaders, s.f.) para que se asignen correctamente las coordenadas de textura sobre el modelo de la carta.

En Unity los shaders se codifican en el lenguaje HLSL (Microsoft, Microsoft - HLSL, s.f.), en la documentación se puede encontrar un simple ejemplo de cómo utilizar este TextureArray en cualquier videojuego (Unity, UnityTextureArrayShader, s.f.). Este deberá adaptar al Pipeline de URP que es el que se está llevando a cabo en el videojuego.

3.1.4 Implementando los Managers

Con el proyecto configurado y las cartas listas para ser utilizadas, se necesita implementar correctamente los managers del proyecto. Estas clases serán las encargadas de manejar un determinado ámbito dentro del videojuego, es por ello que necesariamente necesitamos una instancia única dentro del videojuego ya que van a estar encargados de una tarea específica y no deben existir más de una instancia de la misma, todo ello para asegurar la integridad de los datos.

3.1.4.1 Implementando el patrón Singleton

El patrón Singleton es un patrón de diseño el cual asegura tener una única instancia durante la ejecución del programa, esta única instancia será accesible desde cualquier clase.

Al tener la particularidad de que los managers van ligados indirectamente a este patrón, ya que solamente se requiere de una instancia de los mismos, y que otras clases pueden implementarlo también por otros motivos, como una más fácil accesibilidad o evitar una duplicación de clases innecesaria, se ha pensado en dedicar esfuerzo en programar patrones Singleton genéricos tanto para los elementos MonoBehaviour (GameDevelopment, s.f.), los cuales son clases específicas de Unity pensadas para trabajar dentro del entorno, como para las clases normales de C# (C#Corner, s.f.).

3.1.4.2 Game Manager

Es el manejador del juego y su función principal es la de coordinar todas las acciones que sucedan dentro del propio juego, como cambio de escenas, creación de nuevos juegos y comunicación con las plataformas externas para llevar a cabo el desarrollo de una partida.

Este manejador será el encargado de concordar el resto de manejadores y de proporcionar ciertas funcionalidades para que sean accesible desde todas las clases.

3.1.4.3 Audio Manager

Este manager es el encargado de reproducir la música de fondo y los efectos de sonido, almacenara los clips de audio dentro de esta clase y este los reproducirá con las clases que le hagan peticiones sobre la reproducción de audios.

Los elementos necesarios para reproducir sonidos en Unity son el elementos AudioSouce (Unity, Unity | AudioSource, s.f.), el cual será encargado de emitir los sonidos, y el elemento AudioClip (Unity, Unity | AudioClip, s.f.).

Al tener la necesidad de reproducir la música de fondo y efectos de sonido se han requerido la implementación de dos AudioSource, los cuales serán instanciados en un GameObject dentro de la escena denominado GameManager.

En la siguiente ilustración se puede observar la configuración, de un AudioSource dentro de la plataforma de Unity, este elemento permite realizar leve modificaciones sobre el clip que estas reproducciones, todas estas propiedades son modificables desde los script, por lo que el manejador podrá cambiarlas para crear distintos efectos de sonido.

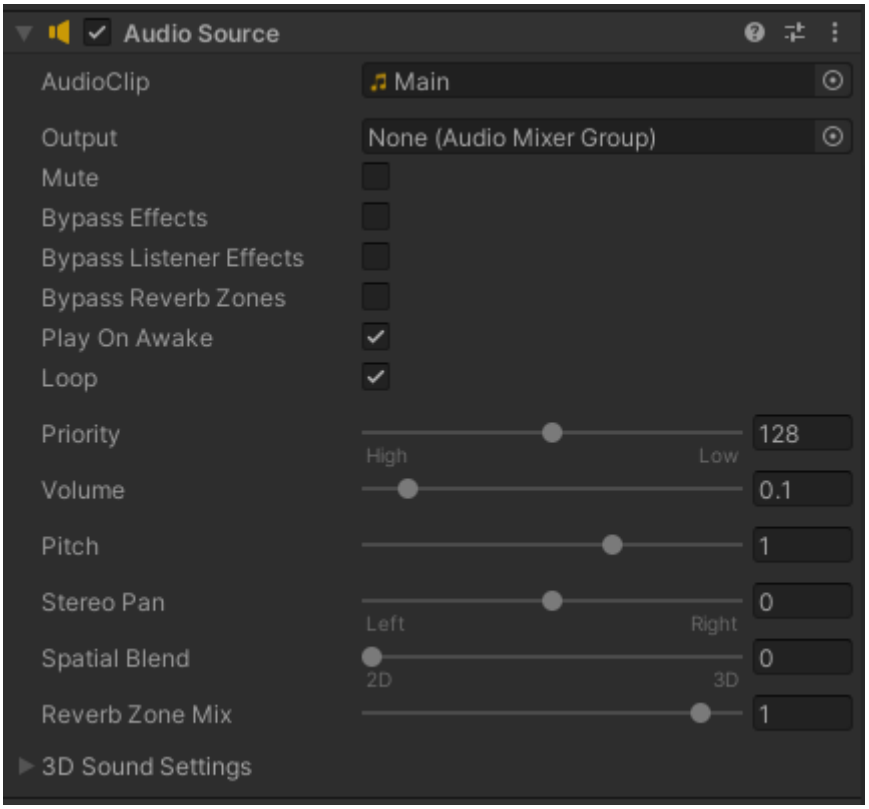

Ilustración 3.5 Captura de la configuración del Audio Source

Para la escena de carga se modificara el Pitch para indicar al jugador que se encuentra en otro momento del juego, indicando que tiene que esperar a que se termine la búsqueda de jugadores.

3.1.4.4 Coroutine Manager

El manager de las corutinas es necesario para tener centralizado la instancias de las coroutinas (Unity, Unity | Coroutines, s.f.), además se han necesitado la personalización de las mismas, por lo que existen clases personalizadas dentro de este *Manager* para tener un control más granular y personalizado sobre las coroutinas instanciadas en el sistema, como su estado, acciones después de finalizar, etc.

3.1.4.5 Writer / Reader Manager

En lo que se refiere a este manager, ha sido necesario para la escritura y lectura de ficheros, la ventaja que aporta tener esta funcionalidad dentro de un manager es que toda la gestión de los mismos está centrada dentro del mismo archivo, junto con flujo de ficheros, por lo tanto asegura la integridad de los datos a la hora de lecturas y escrituras.

El manager de esta clase accederá a dos ficheros, uno será el de la configuración del usuario, en el cual almacenara tanto su nombre, como la autenticación que ha realizado, como el lenguaje que está utilizando actualmente el jugador. Dicho fichero se llamara *configurationUser.csv*

Por otro lado también a través de dicho manager se podrá escribir información en ficheros log, para trazar posibles problemas en los ejecutables los cuales no disponen de una consola como a la hora de ejecutarlo dentro de la plataforma de Unity, aunque esta funcionalidad este deshabilitada también esta implementada por el manejador para utilizarla cuando se considere necesario.

3.1.4.6 Scene Loader Manager

Este manager hará uso del propio SceneManager de Unity (Unity, Unity | SceneManager, s.f.) para realizar los cambios de escena necesarios a través del propio script.

Dentro de este manejador se ha implementado un patrón estrageia, el cual realiza una serie de acciones antes de realizar el cambio de escena y después de realizar el cambio de escena.

De esta forma accediendo a otros managers se realizaran las acciones necesarias entre escenas, como habilitar o deshabilitar los mapas de entrada, ya que en la interfaz de usuario no es útil ni eficiente estar captando continuamente si el usuario ha dejado el botón pulsado o no, además de modificar los clips de audios que se van a reproducir, para cambiar el audio entre la escena del juego y la escena de la interfaz.

3.1.4.7 Input Manager

Este manager accederá a las características de la última versión del *Input System* de Unity (Unity, Unity | Input System, s.f.). Actuará de intermediario entre el nuevo *Input System,* el cual notificara constantemente sobre los eventos de entrada de usuario a dicho manejador y el resto de clases que estén interesadas en saber las acciones del usuario, como pulsaciones de ratón, tipos de pulsaciones (Largas, cortas, doble clic…).

El funcionamiento del nuevo sistema de entrada es mucho más potente y escalable que el que trae por defecto el motor de Unity. Dispone de un configurador, el cual permite agregar distintos mapas para distintos modos de juego, es decir la acción de arrastrar se puede activar para un modo de juego con una pulsación larga, pero esta misma podría tener otra entrada si se encuentra activado otro mapa de movimientos. Para la realización del proyecto solo se ha necesitado uno como se muestra en esta imagen.

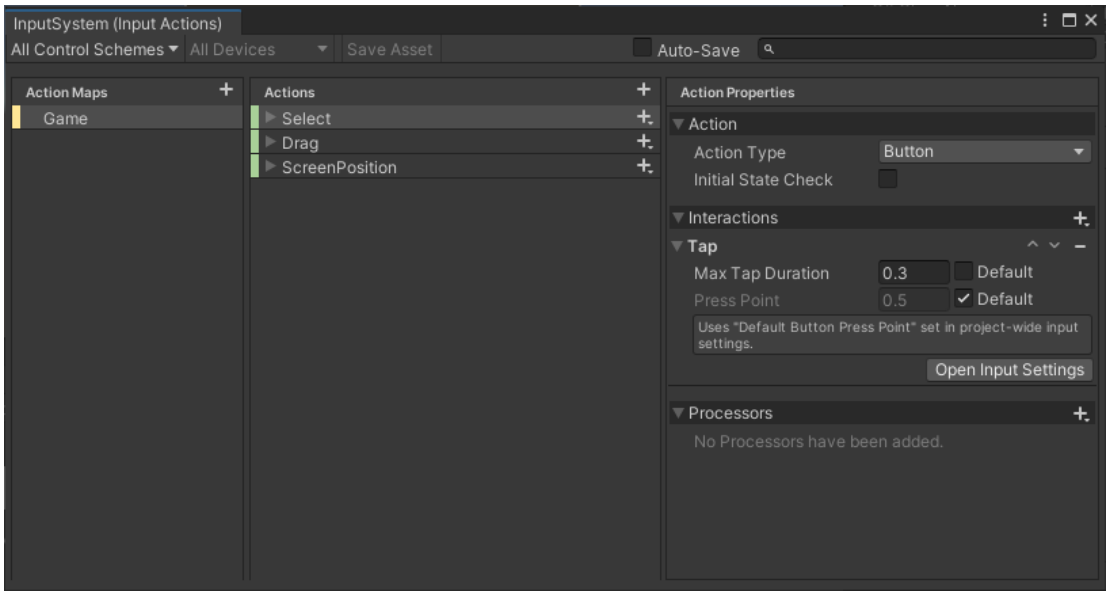

Ilustración 3.6 Captura del mapa configurado para el sistema de entrada

Este mapa puede ser utilizado vía script y es de ello de lo que se encargara este manejador. Cada acción de dicho mapa debe ser creada y configurada en la parte derecha de la ilustración anterior, en el apartado *Action Properties*.

3.1.4.8 Localization Manager

Para este proyecto se ha implementado un sistema de localización, el cual permite a la interfaz de usuario visualizarse en distintos lenguajes. De esta forma el videojuego será accesible para una cantidad mayor de usuarios.

La función de este manager es la de actualizar los datos de la localización y ofrecerlo al resto de clases, que normalmente serán vistas de la interfaz para que asignen los textos correctamente dependiendo del idioma utilizado para el videojuego.

Para dicha localización se utilizaran los *ScriptableObject* (Unity, Unity | ScriptableObjects, s.f.), es una forma útil y sencilla de tener almacenado distintas localizaciones dentro del sistema.

La funcionalidad para la cual están pensada estos objetos es para almacenar distintas propiedades de los enemigos como fuerza, vida, y demás elementos, todo ello sin duplicar código para cada tipo de enemigo. Pero esto puede ser escalable a distintos sistemas de localización en donde se tiene una serie de atributos, que serán los identificadores de los textos de la interfaz y se busca que estos varíen según el idioma seleccionado.

La ventaja que proporciona tener almacenados en *ScriptableObject* es que es más fácil agregar nuevos elementos, ya que todas las “cabeceras” que identifican el texto que se quiere almacenar se encuentran dentro de un script, y para modificar sus datos se puede hacer directamente desde el inspector de Unity, como se puede ver en la siguiente ilustración.

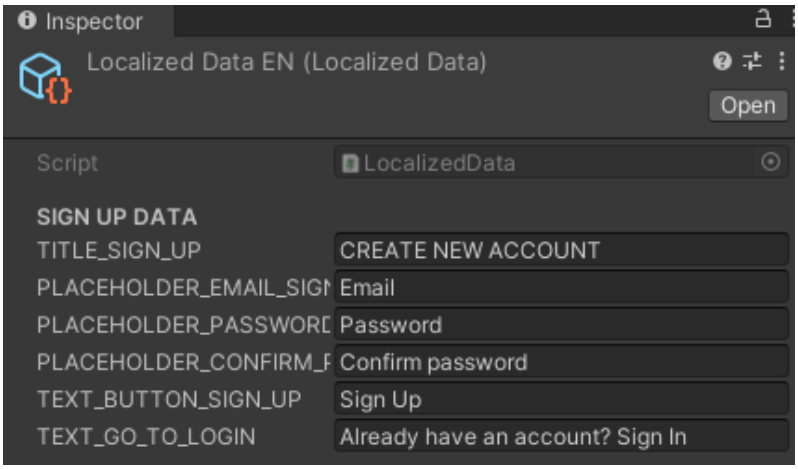

Ilustración 3.7 Captura de los datos de localización en ingles

3.1.4.9 Texture Manager

Esta clase es la encargada de manejar todas las texturas del juego, de echo esta a su vez está conformado de más managers, dependiendo del tipo de texturas o imágenes que trabajen. Por lo tanto la función principal de este Manager es la de dar acceso al resto de managers específicos de cada tipo de textura, de esta forma al centralizarlo en uno solo se hace más simple la utilización.

Con lo explicado anteriormente ya se saben las técnicas utilizadas para la texturización, pero aún no se ha mostrado como se aplica el array de texturas a un propio modelo de carta. De esto se encarga la clase *CardTexturedManager,* dicha clase es la encargada de referenciar un cuadrado pequeño de la imagen con el modelo de una carta.

Para ello primero se tienen que tener almacenados los datos de la carta, estos se hará a través de la clase *Card*, la cual se puede observar en la siguiente ilustración.

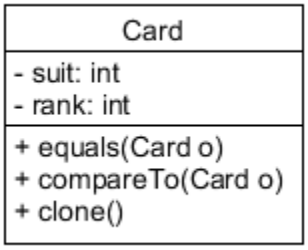

Ilustración 3.8 UML de la clase Card

Cada carta almacena solamente el valor del palo de la carta y el valor del propio número, de esta forma a través de simples operaciones aritméticas se puede pasar una determinada carta al rango de 0 – 47 que es la forma en la que está almacenado la textura. Para realizar este cálculo de textura se utiliza esta ecuación

$$numeroCarta = (suit \times cantidadCartasPalo) + rank$$

Ecuación 3.1 Transformar desde una carta a un numero

CantidadCartasPalo variara según el tipo de mazo que se esté utilizando, ya que la baraja francesa contiene una carta más que la baraja española y con ello el rango de transformación de cartas, que para la baraja francesa es mayor.

3.2 Segunda iteración

Una vez tenemos las cartas preparadas para ser utilizadas dentro del proyecto, se deberá configurar la comunicación con las plataformas externas para que las cartas reaccionen a los mensajes que se reciban de las mismas.

3.2.1 Comunicación con la plataforma de Agentes

Como se dijo anteriormente en el apartado de alternativas y viabilidad la comunicación con la plataforma de agentes se ha realizado a través de sockets, por lo tanto se necesita un servidor socket y un cliente socket para la plataforma de agentes y otro servidor socket y cliente socket para la plataforma de Unity.

El cliente socket de ambas plataformas se conecta a la dirección IP de localhost con el puerto de la plataforma a la cual se quiere enviar el mensaje y lo envía cada vez que lo requiera cualquier plataforma. De esta forma evitaremos que el socket se quede colgado por cualquier motivo.

Por otro lado tenemos los servidores socket, los cuales tienen una particularidad, aunque bien es cierto que existen servidores asíncronos tanto en C# como en Java, en Unity existen problemas con la asincronia que utiliza C#, ya que entraría en conflicto con la asincronia del propio Unity y es por ello que ciertos métodos no funcionan como se esperaban. Para solucionar este inconveniente se han programados los servidores sockets como normalmente se realizarían, aunque estos son bloqueantes, lo cual quiere decir que el hilo de ejecución no continuara hasta que reciba un mensaje. Por lo tanto para solucionar esto los servidores se lanzan en un hilo aparte para que la ejecución continué como espera y se mantendrán a la escucha mientras la partida de modalidad offline siga en juego, de esta forma lo que permanece bloqueado es el hilo que se ha lanzado.

3.2.2 Comunicación con Firebase

En cuanto a la comunicación con Firebase existían pocas alternativas, aunque dispone de un SDK, el cual puede integrarse fácilmente con Unity, pero este no se ha utilizado por que no permite la portabilidad de la Application Programming Interface (API) (Firebase, Firebase | API REST, s.f.), la cual nos da los beneficios de funcionamiento en cualquier tipo de plataforma, aparte de que no requiere configuración alguna dentro del proyecto de Firebase.

Para la comunicación con Firebase necesitaremos 3 identificadores, los cuales serán utilizados con distintas finalidades, a continuación se explicaran cada uno de ellos.

Se necesitara la Uniform Resource Locator (URL) de la base de datos en tiempo real de Firebase (Firebase, Firebase | Instalacion y configuración de la API, s.f.), cada base de datos tiene una URL univoca, se necesitara este identificador para realizar escrituras y lecturas sobre los datos de la misma base de datos, para ello se harán peticiones a través de la API de la plataforma de Firebase.

Para realizar peticiones se debe crear una URL con el contenido que se explica a continuación.

Primero agregamos el identificador del proyecto, seguido de la ruta del documento que queremos modificar, la forma de hacerlo es como si de un sistema de archivos se tratara, por último se agrega la terminación JSON, para indicar que estamos trabajando con ese tipo de datos y se le agregan los datos de autorización, previamente obtenidos.

Otro identificador a tener en cuenta es el de Cloud Functions, este elemento es lo que se conoce como un End Point, consiste en una URL conformada por la dirección de peticiones de Cloud Functions, a la cual se le debe asociar el proyecto de Google Cloud del usuario (Firebase, Firebase | Cloud Functions, s.f.), cabe destacar que el identificador de proyecto de Google Cloud no es el mismo que el identificador de la base de datos.

Este tipo de peticiones se utilizaran para agregar más seguridad a los movimientos realizados por jugadores y que en este caso las Cloud Functions actúen como un servidor dedicado.

Por último se necesita conocer la API_KEY del proyecto, es un identificador que tiene asociado el proyecto de Google Cloud para el envió de peticiones a determinados servicios. Esta es necesaria para enviar peticiones a los End Points de autenticación de Firebase (Firebase, Firebase | Auth API, s.f.). La forma de enviar estas peticiones será con el End Point de autenticación seguido de la API_KEY del proyecto.

3.2.3 Implementación patrón Observer

El patrón Observer será necesario para suscribirse y desuscribirse a la información que recibamos de las plataformas externas, la ventaja que nos aportara será la de no estar haciendo continuamente peticiones para ver si los mensajes que se esperan han llegado o no, de esta forma cunado el mensaje que se está esperando llegue, se notificara a toda la gente que este suscrita a determinado tópico.

Para implementarlo en Unity el propio lenguaje de programación de C# proporciona objetos a los cuales los denomina Actions (Microsoft, Microsoft | Actions, s.f.) o a través de interfaces las cuales ejecutan un método cada vez que se reciba dicha información (Ilustración 3.9). Para el desarrollo del videojuego se han necesitado los dos tipos de patrones Observer, el primero proporciona menos acoplamiento del código mientras que el segundo aporta más especificación y es más simple de llevar a cabo.

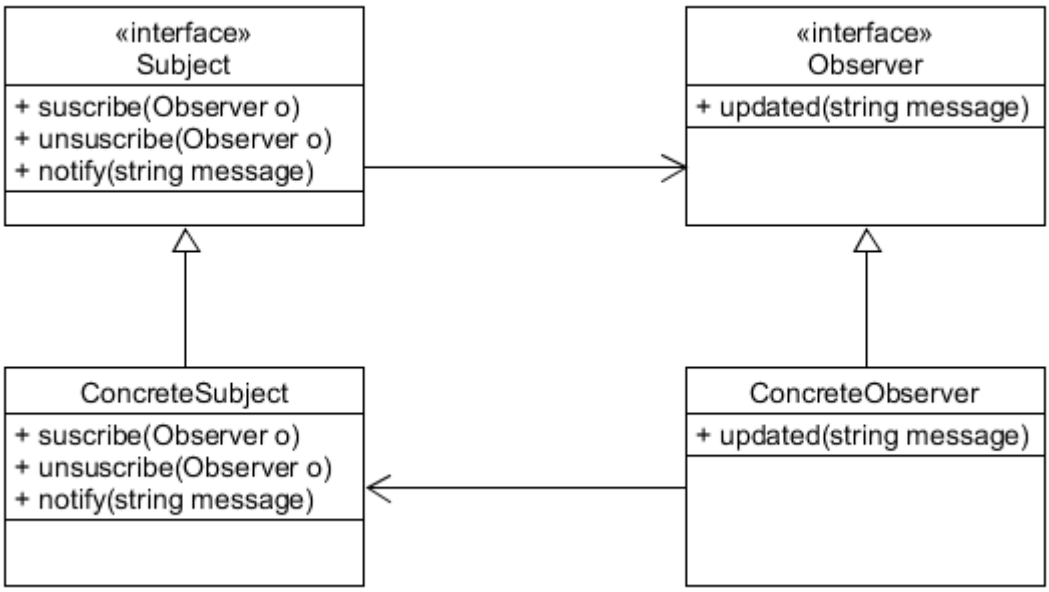

Ilustración 3.9 Diagrama UML del patrón Observer

3.3 Tercera iteración

Esta iteración se centró en realizar la arquitectura de una sesión de juego, lo que permitirá tener una primera versión del videojuego, aunque todavía no sea funcional.

3.3.1 Implementando la estructura

A lo a lo largo de la documentación se le ha denominado sesión de juego a llevar a cabo del desarrollo de una partida de cartas. La propia naturaleza del motor de videojuegos y del desarrollo implica casi “obligatoriamente” la implementación de un patrón MVC, de esta forma se separa correctamente las distintas partes a la hora de llevar a cabo la partida, teniendo por una parte el modelo, con la información de las cartas, por otra parte la vista con los GameObjects correspondientes a las cartas y el controlador para actualizar ambos elementos o no dependiendo de las necesidades.

Una vez identificado el patrón que se ha querido utilizar para representar la sesión de juego se debe identificar qué elementos son necesarios para el desarrollo de dicha sesión. En el caso del proyecto se han identificado los siguientes:

- Deck: Encargado de controlar todos aspectos relativos al mazo de cartas.
- Match: Encargado de controlar todos los aspectos relzativos a los jugadores de la partida, como las cartas de su mano, e informacion asoiciada a los mismos.
- Table: Encargado de controlar los aspectos de las cartas que se encuentran sobre la mesa,como por ejemplo la colocacion de las mismas.

Se ha querido separar en estas tres entidades para que la separacion de resposabilidades sea mayo y reducir el desacoplamiento de una sesion de juego, de esta forma es mas facil identificar en que lugar se deben realizar las modificaciones para incluir nuevas implementaciones.

3.3.1.1 Deck

Los elementos utilizados para la implementación del mazo son los que se pueden observar en la siguiente ilustración.

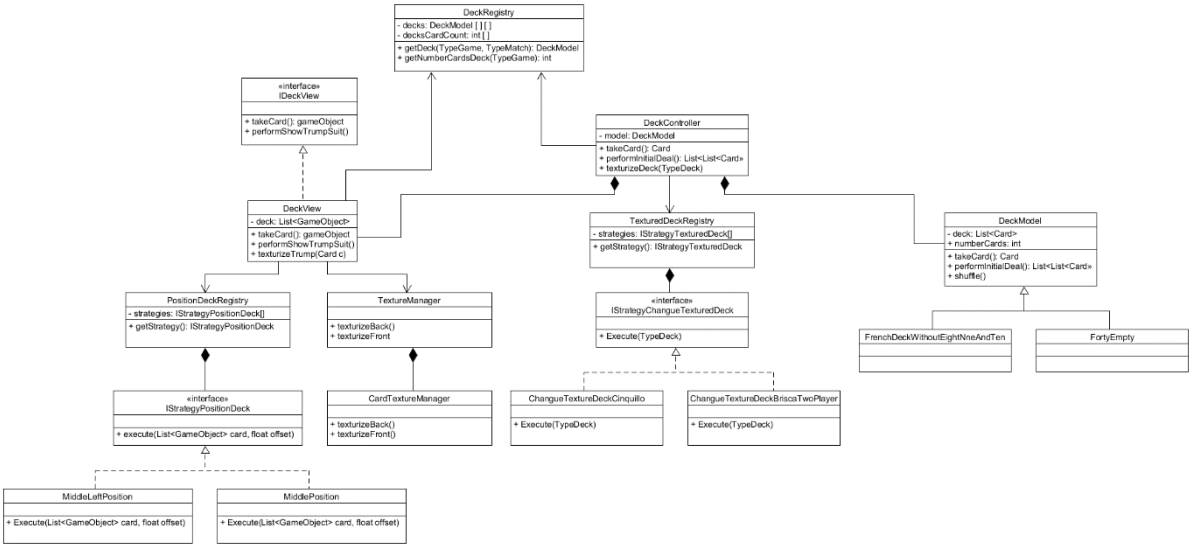

Ilustración 3.10 Diagrama de clases del patrón MVC Deck

En la vista del mazo se ha implementado un patrón estrategia para colocar el mazo de cartas en una determinada posición dependiendo del tipo de juego, esto nos permite que nada más registrando una posición para un juego el mazo se coloque directamente en dicha posición. Además la vista es la encargada de agregar texturas a las cartas, ya que esta responsabilidad recae sobre los GameObjects almacenados en la vista, para ello hará uso del TextureManager, el cual permite texturizar los modelos de las cartas.

La vista del mazo implementa una interfaz para que el resto de clases que necesiten actualizar solamente la vista sin necesidad de modificar u obtener datos del modelo pueden hacerlo a través de dicha interfaz, de esta manera se logra desacoplar más el código.

Por otro lado se ha implementado el modelo del mazo, este consiste en una clase base la cual marca la estructura del resto de modelos. Estos modelos son almacenados dentro de la clase DeckRegistry, dicha clase, almacena las referencias de los modelos de mazo correspondiente al tipo de juego, en el caso del proyecto no han sido necesarios la implementación de distintos modelos de mazo, pero puede darse el caso de que haya juegos que se jueguen con la baraja española completa y

juegos en los que se quiten los números nueve y ocho, de esta forma almacenándolos en esta clase, se hace más simple la implementación de nuevos mazos para nuevos tipos de juego.

Por ultimo existe el controlador, el cual es el encargado de realizar las operaciones sobre el modelo, por lo que todas las peticiones pasan por este, ya que es normal que tenga que realizar antes una serie de comprobaciones, y en el caso que sea necesario actualizar también la vista.

3.3.1.2 Match

En este apartado se hablara sobre la estructura que sigue el concepto de partida encajado dentro del marco arquitectónico del patrón MVC. Dicha estructura la podemos observar en la siguiente ilustración.

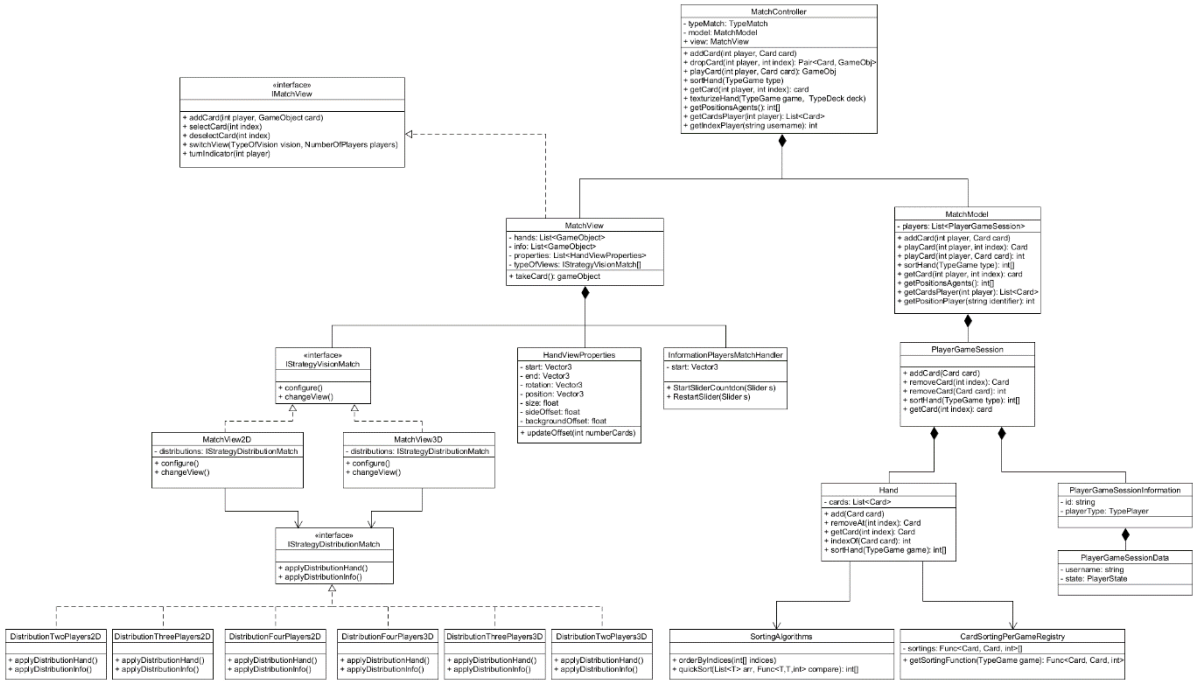

Ilustración 3.11 Diagrama de clases del patrón MVC Match

Comenzando con la explicación del modelo, cada partida almacenara un *PlayerGameSession* por cada jugador que sea necesario, para simplificar esto la información relevante a la propia instancia del jugador estará almacenada en la clase *PlayerGameSessionInfo*, de esta forma cada vez que el jugador comience un juego le pasamos esta información, para que cree una nueva mano de cartas para el jugador (Lo que en la ilustración anterior se puede ver como la clase Hand). Para no enviar

tanta información a la plataforma de Firebase se ha creado la clase *PlayerGameSessionData*, que solamente contiene la información que se necesita para la partida, de dicha manera también tenemos esta información separada en otra clase y no existe la necesidad de ir seleccionándola si estuviera toda almacenada dentro de la propia *PlayerGameSession*. Siguiendo con el modelo, la clase Hand almacena una lista de cartas, concretamente las de la mano del jugador, estas se ordenaran inicialmente según el criterio establecido por el tipo de juego dentro de la clase *CardsSortingPerGameRegistry*, la cual almacena funciones con un criterio de comparación, una vez obtenido el criterio de comparación de cartas ejecutara el algoritmo de ordenación que almacena la clase *SortingAlgorithms,* en el caso del proyecto se ha implementado el QuickSort aunque, para la cantidad de elementos a ordenar la eficiencia no se hace de notar.

Por otra parte tenemos la vista de la partida la cual es bastante más compleja del modelo, el porqué de ello es por la necesidad de calcular posiciones y rotaciones para cada tipo de distribución. A continuación se explicara detalladamente como se realiza las operaciones en la mano del jugador.

Lo primero es almacenar las posiciones y rotaciones de las manos de los jugadores, estas vienen dadas por la estrategia de distribución que podemos ver en la ilustración 1.4, las cuales dan una posición y rotación para un número de jugadores concreto con un tipo de visión concreta, ya sea la visión 2D o la visión 3D. Una vez aplicada dicha estrategia tenemos las 4 manos del jugador representadas con un GameObject cada una colocada en la escena del juego.

Después de la colocación de los GameObject contenedores, se agregaran los GameObject de las cartas, previamente creadas por la clase DeckView, a la mano de los jugadores, es decir se agregaran como hijos dentro de esto GameObject, para que mantengan su posición en el espacio y las cartas no se despilfarren dentro de la escena, pero esto no nos asegura que siempre estén en la posición correcta, por lo tanto debemos calcular dichas posiciones relativas dentro del GameObject contenedor.

Para dar una mejor experiencia de juego las cartas cuando se reparten se van ajustando a un determinado tamaño, lo mismo para cuando se van lanzando y se van eliminando de la mano del jugador. De realizar todas estas operaciones con la mano del jugador se encarga la clase *HandViewProperties,* junto con la propia *MatchView,* a continuación para explicar el funcionamiento de dicha clase, nos apoyaremos sobre la siguiente ilustración.

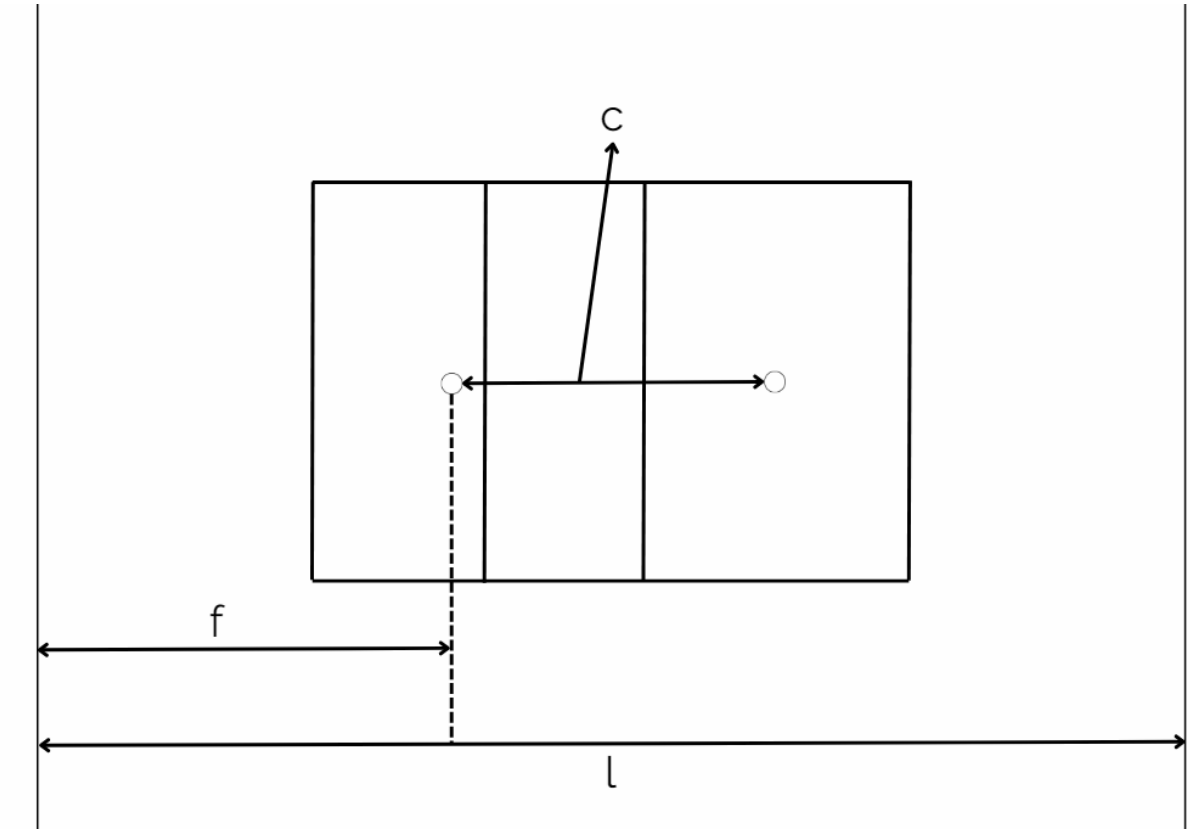

Ilustración 3.12 Esquema de los parametros de las propiedades necesarias para los movimientos en la mano del jugador

El elemento c, es la ocupación actual de la mano del jugador y va desde el centro de la primera carta hasta el centro de la última, para calcularlo utilizamos la siguiente formula:

$$c = numeroCartas * desplazamientoEjeX$$

Ecuación 3.2 Calculo de la ocupación de la mano del jugador

Donde el número de cartas viene dado por el número de hijos del contenedor y el desplazamiento en el eje x es un valor entero dado con anterioridad.

Por otro lado tenemos l, este elemento es la distancia total que como dijimos anteriormente ya ha sido asignada por defecto.

Por ultimo tenemos el valor f, el cual es un valor simétrico, es decir, debe ser el mismo tanto por la parte izquierda como por la parte derecha, para mantener las cartas centradas dentro del GameObject contendor, a este valor f se le denominara distancia con el origen, el cual sigue la siguiente formula.

$$f = \frac{(l - c)}{2}$$

Ecuación 3.3 Calculo del espacio libre en la mano del jugador

Al restar l – c, se obtiene un nuevo valor con la distancia sin ocupar por la mano del jugador, y como hemos dicho que esta distancia debe ser simétrica se divide entre dos para dejar la misma distancia por ambos lados.

Este no es el único aspecto a tener en cuenta cuando estamos agregando las cartas a la mano del jugador, también se debe tener en cuenta que ocurre cuando el valor de c, supera en tamaño al valor de l, ya que de esta forma no nos serviría la ecuación que se está usando actualmente. Cuando ocurra esta situación, lo que se hará será actualizar el desplazamiento en el eje x que se hará más pequeño siguiendo la siguiente formula.

$$desplazamientoEjeX = \frac{l}{numeroCartas}$$

Ecuación 3.4 Calculo de la separación entre cartas

Al dividir el tamaño total por el número de cartas obtenemos un desplazamiento en el eje x el cual se ajusta completamente al taño de la mano del jugador.

El último elemento de la vista es el manejador de la información del usuario, en la actualidad solamente tiene un elemento interactivo que es el slider del tiempo, el cual esta clase será la encargada de incrementarlo y decrementarlo conforme se vayan actualizando los turnos de la partida. Además este elemento servirá como indicador de turno, cambiando de color para cuando el turno es del jugador. La posición de estos elementos también es asignada por la estrategia de distribución que

aplica las posiciones a los GameObject contenedores. Al igual que ocurría con el mazo también implementa una interfaz para ser accesible por otras clases.

Por ultimo tenemos el controlador que su función es muy similar a la del mazo que vimos en el apartado anterior, en él se permite el cambio de visión 2D a 3D en tiempo de ejecución, agregar y borrar cartas de la mano del jugador coordinar los sliders de los jugadores, etc.

3.3.1.3 Table

En cuanto a la mesa esta se ha divido en dos zonas una que contendrá las cartas correspondientes al juego y otra que contendrá las cartas descartadas del juego. El diagrama de clases es el siguiente:

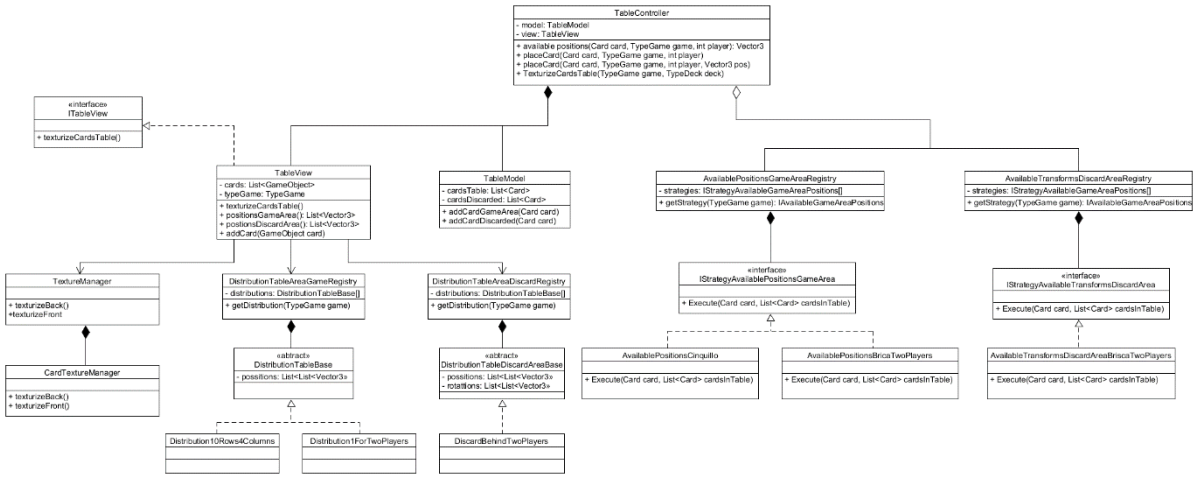

Ilustración 3.13 Diagrama de clases del patron MCV Table

Por un lado tenemos el modelo que es bastante simple, solamente almacena las cartas descartadas y las cartas del tablero.

La vista contiene varias estrategias las cuales almacenan las posibles posiciones en las que se pueden colocar tanto las cartas de descarte como las cartas del juego. Como cada tipo de juego tendrá unas determinadas posiciones o incluso puede que no tenga posiciones de descarte, se ha implementado el patrón estrategia, para almacenar dichas posiciones por cada juego. La vista también hace uso del TextureManager, al igual que las anteriores para texturizar correctamente las cartas y cambiarlas en tiempo de ejecución.

Para acabar, el controlador tiene unas estrategias para las posiciones de descarte y la zona de juego, esta estrategia lo que hace es relacionar una determinada

carta con una determinada posición del tablero, de las cuales se calcularon en la vista, ya que por ejemplo en el juego del cinquillo cada carta debe estar en una posición específica para su correcta visualización. Aunque lo más común es que una carta solamente pueda ser colocada en una posición, en futuras implementaciones una misma carta podría ser colocadas en distintas posiciones posibles, es por ello que existe esta distinción entre estos patrones estrategia.

3.3.2 Movimientos de las cartas

Las cartas deben de tener un movimiento fluido, estas están en constante cambio dentro del patrón MVC, las referencias de los GameObjects se van intercambiando entre las distintas vistas hasta llegar a la mesa, junto con las respectivas actualizaciones del modelo.

Para mover estas cartas se ha decidido utilizar una clase denominada *Lerper,* el cual se basa en el patrón Singleton, para ser accesible por el resto de clases. Básicamente es un interpolador como indica la traducción de su nombre, y lo que hace es realizar traslaciones y rotaciones en el tiempo que se le indique. Se decidió esta propuesta ya que el hecho de realizar animaciones podría llegar a ser más costoso y el resultado que se obtiene en los movimientos realizados con esta clase es más que aceptable. Otro motivo por el que se decidió que era mejor motivo tener esta clase es la alta personalización que nos proporciona a través de los scripts, cosa que las animaciones no.

En lo que se refiere a la explicación del interpolador las operaciones que realizas son de translación, rotación y escalado, a este se le indica sobre que GameObject tiene que actuar y se le indica los parámetros con los que comienza y los parámetros con los que quiere que acabe el propio GameObject, el resto se encarga de realizarlo la corutina a través de funciones de interpolación que proporciona el propio Unity.

3.3.3 Implementando máquinas de estado

Con la estructura anterior no se desarrolla el transcurso de la partida ya que lo único que se ha hecho es marcar la estructura de los distintos elementos de la partida.

Para el desarrollo de la propia partida se ha decidido implementar máquinas de estado para controlar el estado actual de la partida. Esto ha sido posible gracias a la

naturaleza de la aplicación la asincronia y el sistema de turnos deja muy marcado los posibles estados de un jugador en el transcurso de la partida. Es cierto que la implementación de este tipo de patrón es bastante laboriosa, pero a cambio nos ofrece una gran escalabilidad, pudiendo meter estados y transiciones fácilmente para las nuevas funcionalidades en los distintos tipos de juegos.

Para el desarrollo del videojuego se han necesitado dos máquinas de estado, una para llevar a cabo el desarrollo de la partida y otra para llevar a cabo las habilidades del jugador dentro de la partida.

3.3.3.1 Máquina de estados del juego

Esta máquina de estados almacenara el estado actual del juego desde que comienza hasta que finaliza, dentro de estos estados se realizar las acciones correspondientes.

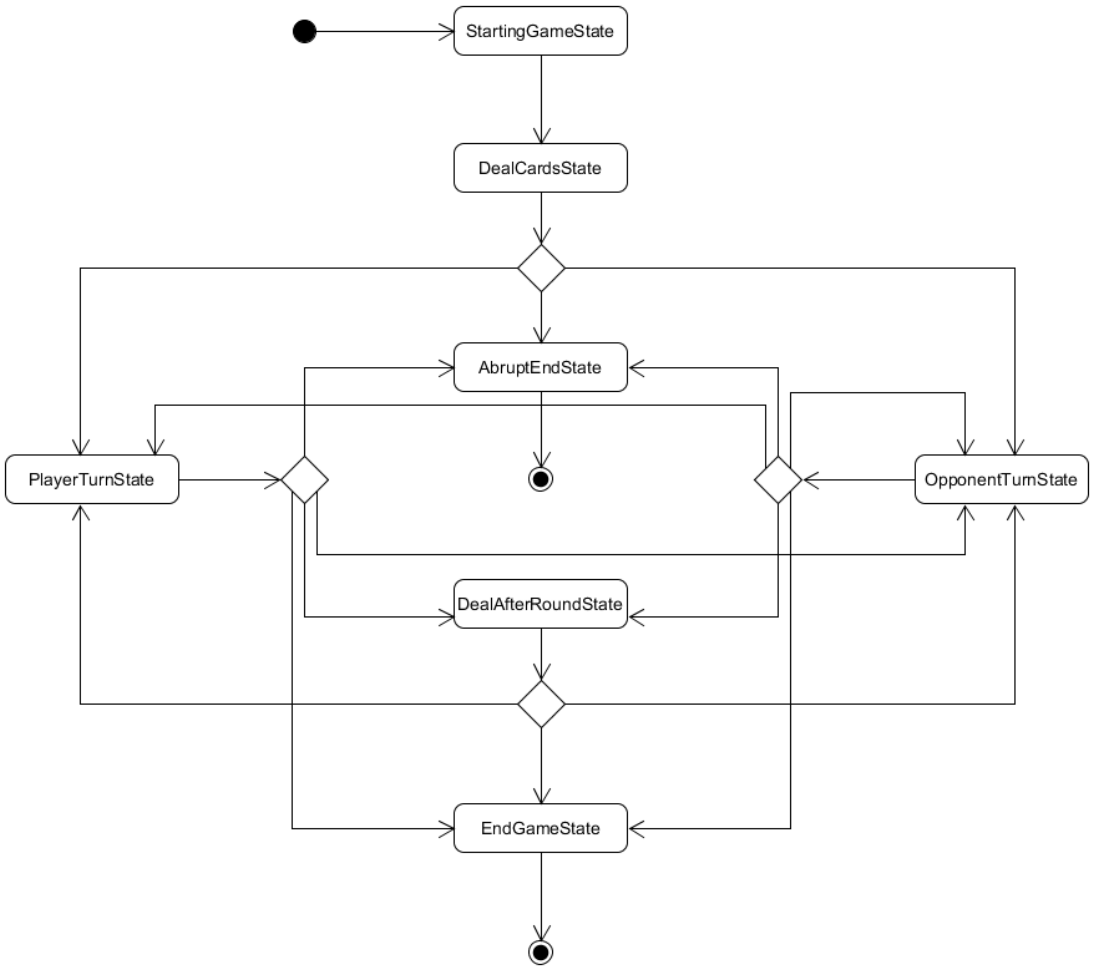

Ilustración 3.14 Diagrama de la máquina de estados para una sesión de juego

Como se puede observar en la imagen anterior para los juegos implementados ha sido necesario identificar los siguientes estados:

- Starting Game: En este estado se pueden realizar algunas operaciones antes de comenzar el reparto, actualmente su unica funcion es transitar al reparto inicial de cartas.
- Deal Cards State: En este estado se realiza el reparto inicial de cartas, asi como la configuracion inicial de turnos de la partida.
- Player Turn State: Es el estado del turno del jugador, en el se mantiene a la espera hasta que el propio jugador realice un movimiento, si no lo realiza dentro del tiempo establecido se realizara un movimiento aleatorio.
- Opponent Turn State: Este estado es el contrario al del turno del jugador, en el se mantiene la espera hasta recibir un movimiento de la plataforma externa, una vez recibido se actualiza la vista y se transita al siguiente estado según el tipo de juego
- Deal After Round State: Este estado es opcional, no todos los juegos deben de pasar por el. Dicho estado es el encargado de realizar el reparto de cartas a los jugadores despues de una ronda de juego, esta operación es necesaria solamente para el juego de la Brisca.
- End Game State: Es el estado final del juego y se accede a el cuando la partida ha finalizado, su funcion es recibir los datos de la clasificacion y actualizar los elementos necesarios para que se muestre en la interfaz.
- Abrupt End State: Este estado es otro estado final, al igual que sucede con el *End Game State*, la diferencia es que a este estado se llega por que ha ocurrido algun error durante el desarrollo de la partida o el jugador a cerrado la aplicación de forma inesperada. Realiza las operaciones necesarias para que se eliminen todos los elementos correctamente.

Durante la explicacion de los estados se ha podido apreciar que los estados necesitan compartir informacion entre ellos para su correcto funcionamiento, es por eso que ha sido necesario la implementacion de contextos, dichos contextos almacenan la informacion comun para todos los estados, los cuales son accedidos y modificados por ellos mismos.

3.3.3.2 Máquina de estado para las mecánicas del juego

También se ha considerado necesario la implementación una máquina de estados para la acciones a realizar del jugador, de esta forma se asegurara que solamente se esté realizando una mecánica en un determinado momento y que no entren en conflictos unas con otras, además al realizar la máquina de estados es más simple modificar una mecánica en específico y establecer transiciones entre mecánicas.

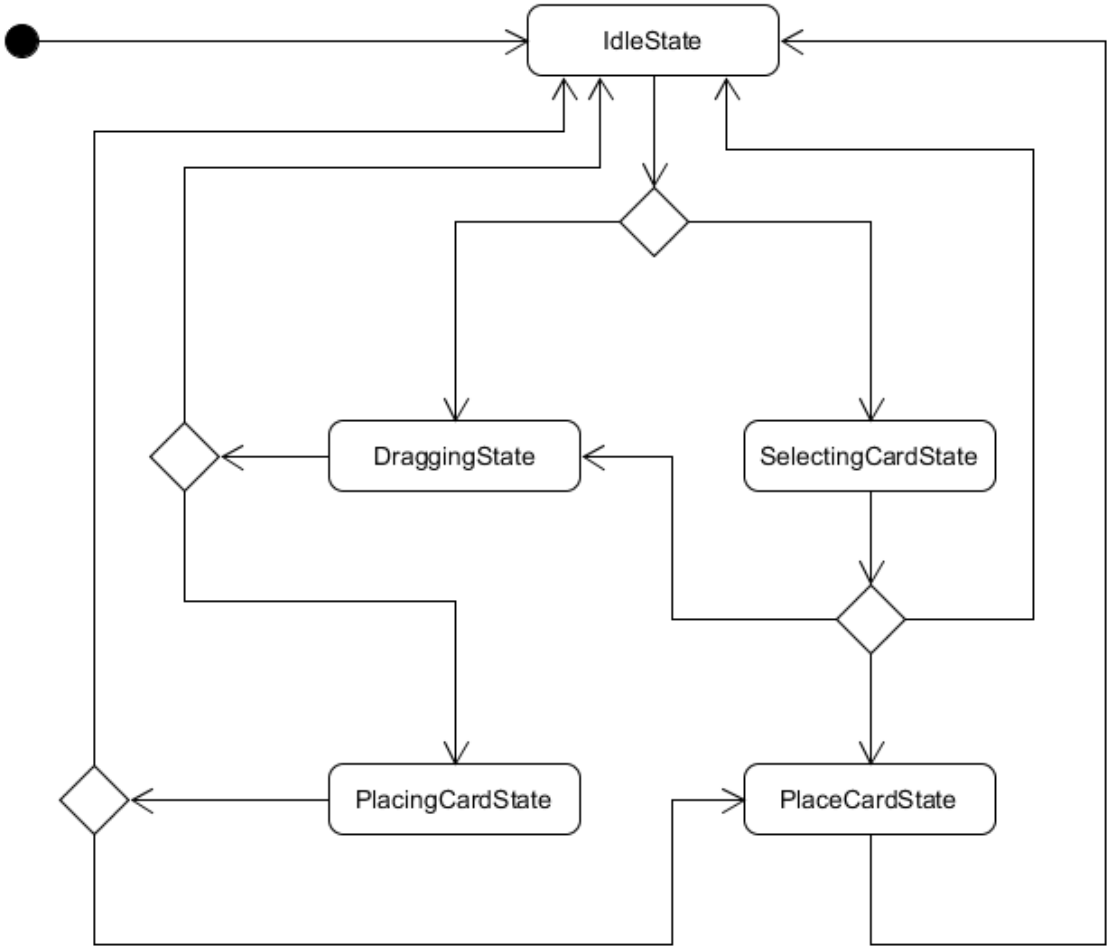

Ilustración 3.15 Diagrama de la máquina de estados para las mecánicas del jugador

Para esta máquina de estados, se han podido identificar los estados que se explicaran a continuación

- Idle State: Es el estado incial de la maquina de estados, en el se encontrara cuando el jugador no este rezando ninguna accion, este estado se mantendra a la espera hasta que se produzca un evento I/O.
- Draggin State: Este estado consiste en el arrastre de las cartas del jugador, ya que estas pueden ser movidas hasta el destino donde tienen que ser lanzadas, para entrar a este estado el jugador debe dejar pulasado el boton sobre la carta.
- Selection Card State: En dicho estado se realiza la accion de seleccionar la carta clicada, si el jugador hace clic sobre una carta, esta no se lanzara automaticamente, se ha pensado esta idea con el fin de no lanzar cartas de forma indeseada, por lo que las carta seleccionada sobresaldra del resto, junto con las de sus alrededores para que el usuario confirme el movimiento.
- Placing Card State: En este estado se llega solo a traves del estado de arrastre de cartas, se mantendra en este estado mientras el jugador no suelte el boton de arrastrar, de esta forma se le da la opcion de cancelar el movimiento al usuario con la carta que comenzo a arrastrar.
- Place Card State: Este es el ultimo estado, el cual coloca la carta sobre la mesa y elimina la carta de la mano del jugador.

Esta maquina de estados necesita estar en contacto con la maquina de la sesion del juego, para que en el estado del jugador no sucedan acciones inesperadas cuando se realice un movimiento aleatorio, es por ello que se debera de notificar a traves del patron Mediator, la entrada o salida a distintos estados de la maquina de estados de mecanicas a la maquina de estados del juego, esta ya se encargara de notificar a sus estados a traves del peatron Observer.

3.3.3.2.1 Detección de colision sobre las cartas

Cuando el jugador realiza determinadas acciones como las pulsaciones de botón se lanza un rayo desde la cámara, realizando las conversiones necesarias al espacio del mundo, para obtener con que elementos ha colisionado dicho rayo lanzado.

Si dicho rayo choca con una carta entonces es cuando comienzan las transiciones de la máquina de estados nombrada anteriormente. Pero para detectar estas colisiones es necesario que las cartas contenga un componente que sea un Collider (Unity, Unity | Colliders, s.f.).

Concretamente se utilizara un BoxCollider (Unity, Unity | BoxCollider, s.f.), por su sencillez y eficiencia la hora de detectar colisiones y también porque la carta no requiere de colliders muy complejos, los cuales tengan una forma muy irregular. Este elemento se agregara dentro de Prefab, normalmente desactivado, ya que no se busca que el jugador pueda interactuar con las cartas de los adversarios, solamente con las suyas propias.

Este BoxCollider ha sido necesario ajustarlo mucho al tamaño de la carta, para que la precisión con los clic sea lo más alta posible, en la siguiente imagen el borde verde marca la zona donde se detectara la colisión al hacer clic por el jugador.

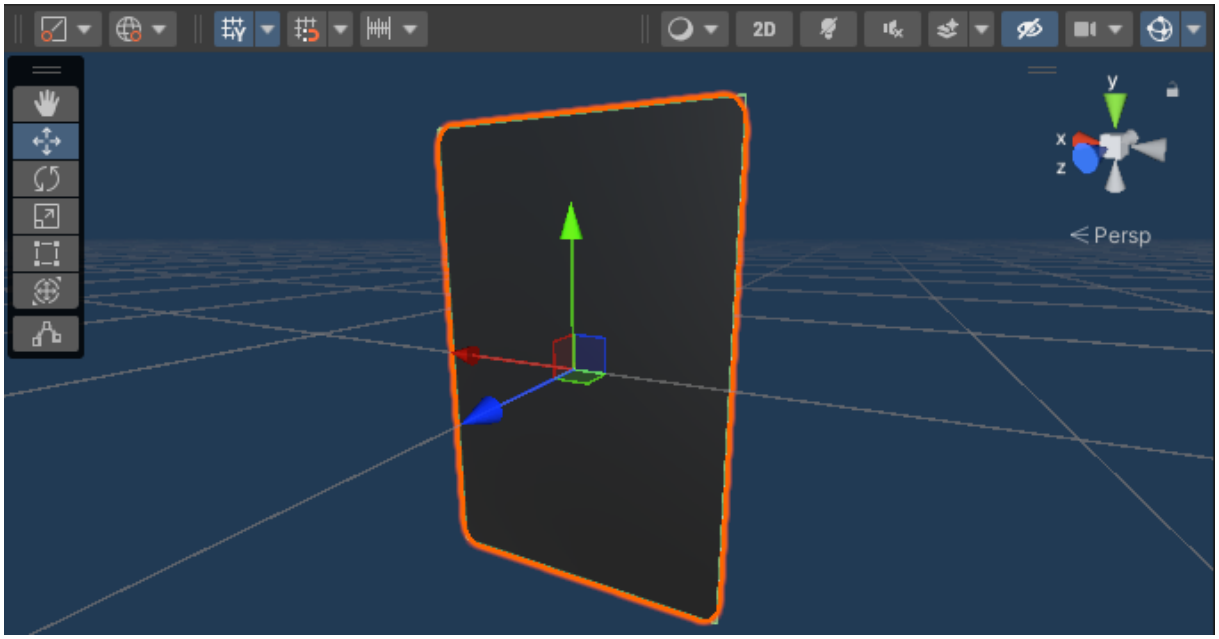

Ilustración 3.16 Captura sobre el Prefab de la carta dentro de Unity

3.3.4 Implementacion del Game Session Controller

Con la estructura explicada anteriormente ya estaría completa la estructura para llevar a cabo una sesión de juego, pero es necesario que una clase las instancie dentro del propio sistema y que coordine el paso de información entre máquinas de estados, interfaces y demás aspectos dentro de una sesión de juego.

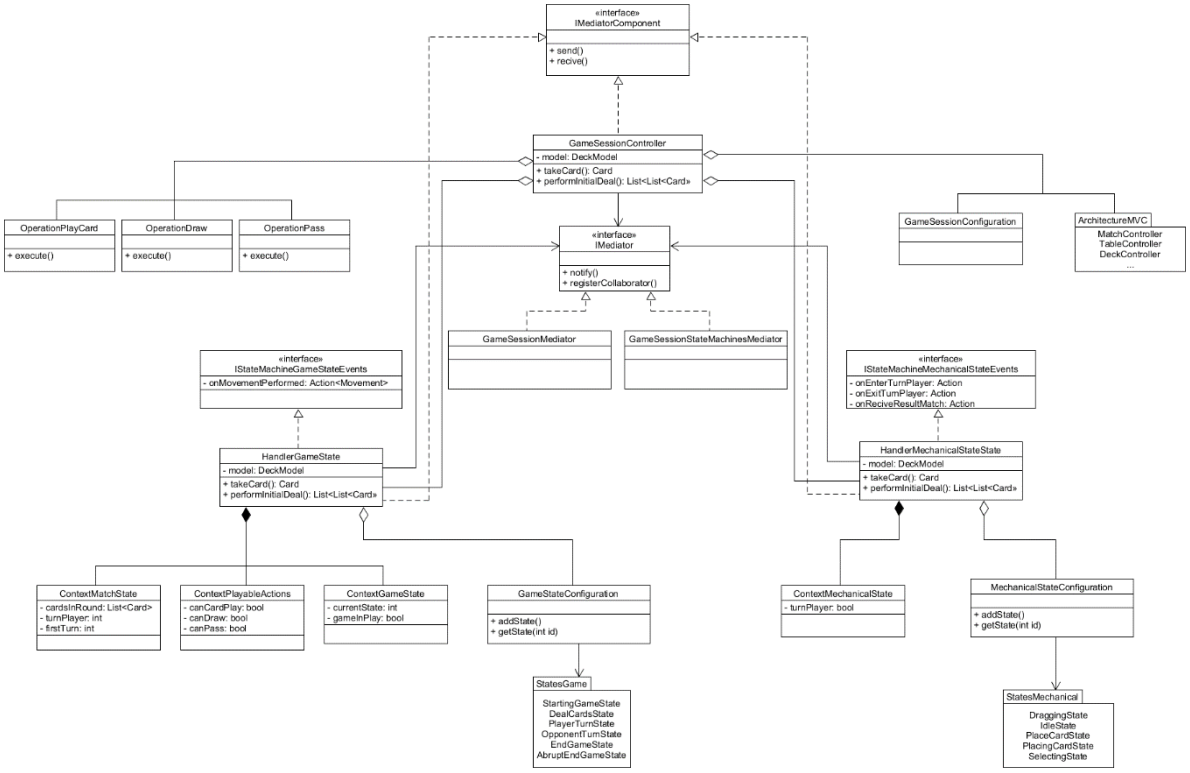

Ilustración 3.17 Diagrama de clases del GameSessionController

Esta clase también será la encargada de instanciar las operaciones que puede realizar un jugador ya que estas se utilizaran en distintas partes de la máquinas de estados y evita la duplicidad de código que se instancie solo a través del controlador.

Otro aspecto del cual se encarga el *GameSessionController* es de colocar al jugador siempre en la primera posición de la lista de jugadores y rotar al resto de jugadores para que la concordancia de los movimientos sea la misma en todas las instancias de la partida. Lo que se consigue con este algoritmo es tener los jugadores ordenados de la forma que muestra la ilustración.

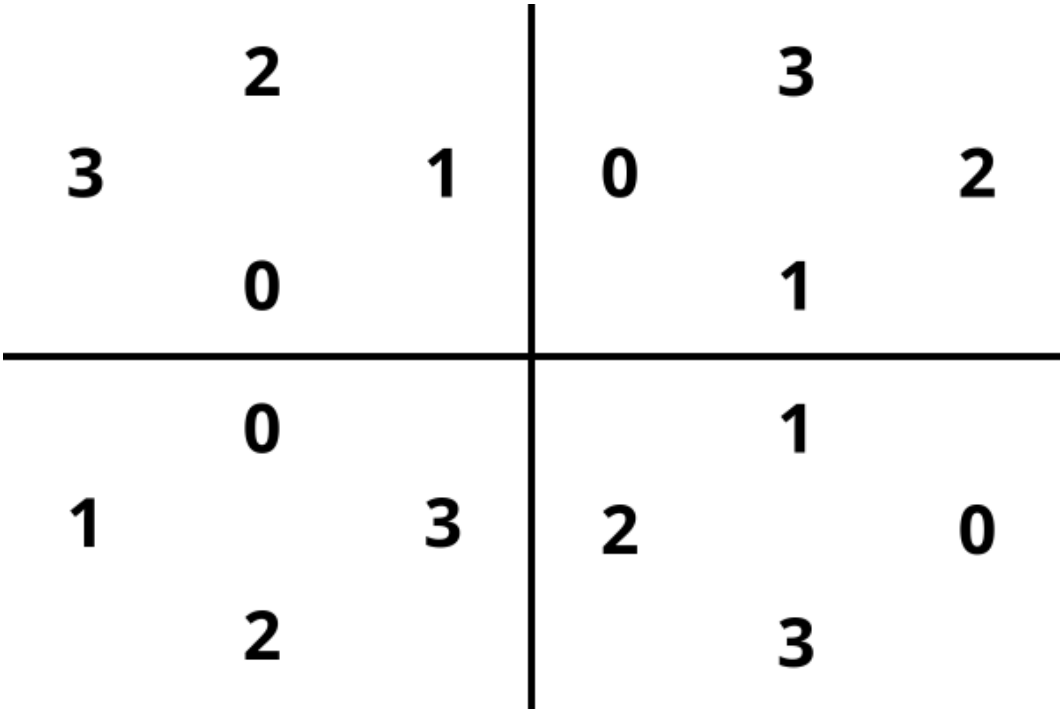

Ilustración 3.18 Posiciones de los jugadores en una partida online

Este algoritmo es de vital importancia para las partidas online, como se observa al realizar estas rotaciones en las listas cada jugador en su propia instancia tiene a los mismos jugadores colindantes, de esta forma los movimientos se sincronizan correctamente. Sin embargo en las partidas locales el jugador por defecto siempre estará en la posición 0, por lo que la aplicación de este algoritmo no es necesaria.

En las siguientes iteraciones se mostrara como se comunica la plataforma de Unity con las plataformas externas, la clase *GameSessionController* instanciara los patrones estrategias necesarios para realizar dichas comunicaciones necesarias en diferentes momentos del desarrollo de una partida. La idea de dichas comunicaciones con las plataformas es que fueran lo más similares posible dentro de sus evidentes diferencias, para evitar estar comprobando continuamente en las máquinas de estado el tipo de juego que se está realizando.

3.4 Cuarta iteración

En dicha iteración se hablara sobre la implementación de la plataforma de agentes, la cual permitirá al jugador desarrollar partida contra lo que comúnmente es conocido como la “maquina”, aunque sería más correcto asimilar este concepto con la inteligencia artificial del sistema. Además de esto se tendrán en cuenta los aspectos más relevantes para llevar a cabo una partida offline y como se solucionan las circunstancias propias de la partida.

3.4.1 Implementando el manejador de la plataforma de agentes

Para la comunicación con la plataforma de agentes, se requerirá de un manejador el cual permita el envío y recepción de mensajes con la propia plataforma.

Como se ha comentado a lo largo de la documentación la comunicación se realizara a través de sockets por lo que también será el encargado de mantenerlos activos y eliminar las conexiones cuando sea necesario.

Otro aspecto del cual se encarga el manejador será de levantar la plataforma de agentes, para ello dispondrá del archivo .jar previamente compilado el cual ejecutara a través de la utilidad de C# Process (Microsoft, Microsoft | Process, s.f.), para ejecutar aplicaciones externas.

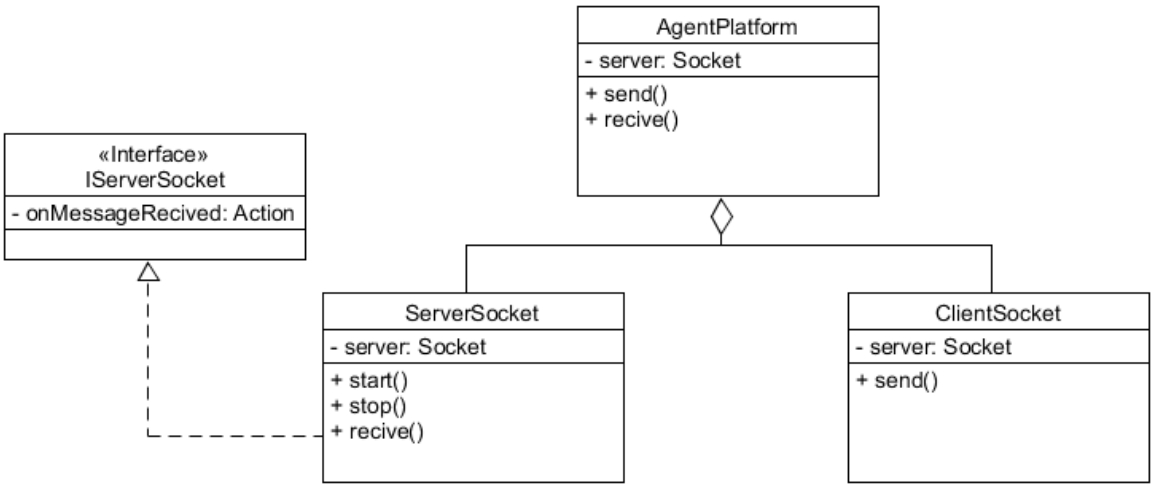

Ilustración 3.19 Diagrama de clases de la plataforma de Agentes en Unity

Una característica del manejador es que implementa un sistema de mensajes en formato JSON para comunicarse con la plataforma de agentes, se ha decidido esta estructura para unificar el formato de comunicación con ambas plataformas, tanto la de agentes como la plataforma de Firebsae además de por su simplicidad a la hora de enviar información a través de los sockets. Cada mensaje que se envié y reciba tendrá un identificador y después una cadena con el contenido del mensaje en formato JSON, de esta forma se deserializara primero con la clase mensaje y a través de este identificador ya se sabrá que clase utilizar para deserializar el contenido del propio mensaje.

3.4.2 Comunicación para iniciar y apagar la plataforma

Posteriormente a la ejecución de la plataforma de agentes, deberá existir una comunicación inicial para saber cuándo es viable el intercambio de información, esta comunicación se puede observar a través de la siguiente imagen.

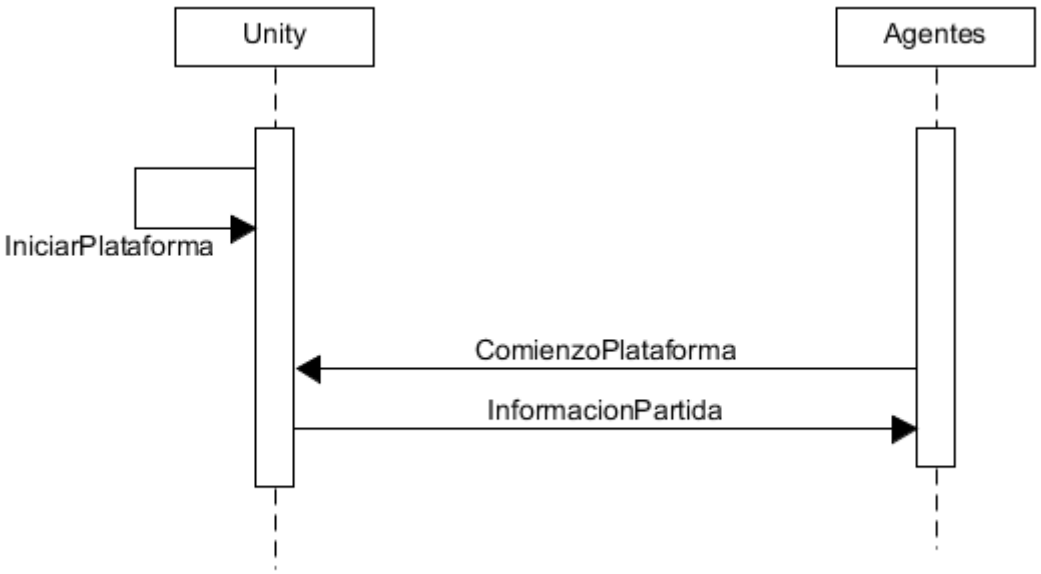

Ilustración 3.20 Diagrama de comunicación inicial para el comienzo de la partida

Esta comunicación es necesaria porque existe una espera mientras comienza la plataforma de agentes, lo cual se puede producir un error al enviar la información de la partida y que se pierda el mensaje, por ello es importante esperarse hasta recibir el primer mensaje de la plataforma de agentes.

Por otro lado para la finalización de la plataforma también debe de existir una comunicación la cual haga que finalice la plataforma de Unity, el esquema de comunicación se muestra en la ilustración 3.21

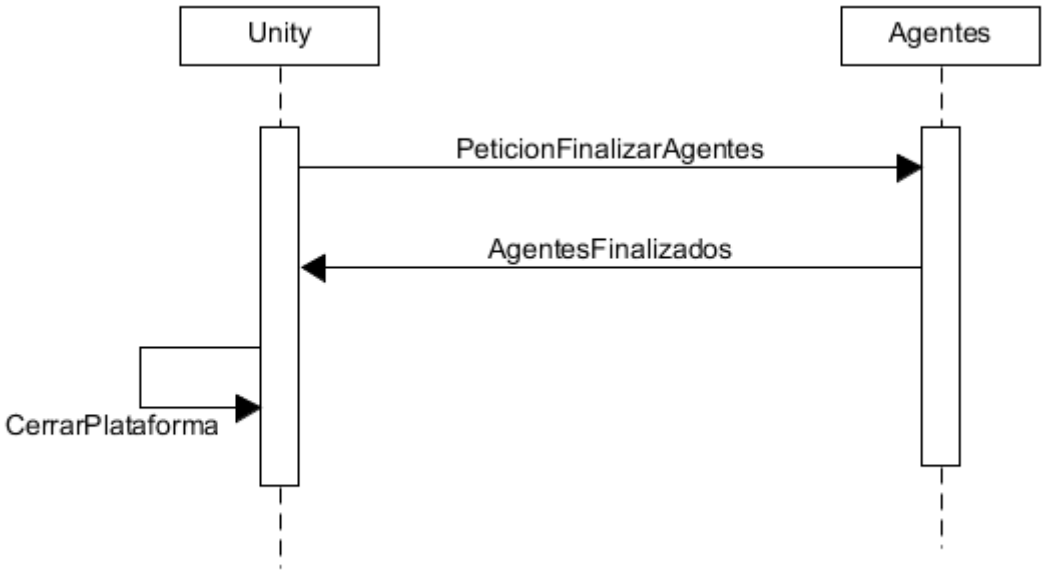

Ilustración 3.21 Diagrama de comunicación para finalizar la plataforma de agentes

La comunicación final también es importante, se deberá esperar el mensaje de finalización de la plataforma, para esperar hasta que finalicen todos los agentes, de esta forma se liberaran los recursos correctamente.

3.4.3 Como apuntarse a la lista del matchmaking

El matchmaking para una partida offline se hace de forma inmediata, la forma de realizarse es crear un *PlayerGameSession* representando al propio jugador y crear el resto de *PlayerGameSession* con el nombre de IA, dependiendo del tipo de juego.

No se necesita establecer comunicación con nadie, por lo tanto después de crear los jugadores que conformaran la partida se pasa a la escena que lleva a cabo las sesiones de juego.

Para controlar los tiempos que tarda la plataforma de Unity en agregar a los jugadores creados, realizar el reparto de cartas e instanciar los GameObject necesarios, dentro de la plataformas de agentes, cuando se reciba la información con la partida que se tiene que llevar a cabo se lanzara una tarea encargada de esperar

una determinada cantidad de tiempo para comenzarla. Así la plataforma de agentes se asegura que todo lo necesario esta instanciado antes de comenzar la comunicación.

3.4.4 Como realizar el reparto inicial

Al tener la información del mazo en la propia instancia de la plataforma de Unity, tampoco se necesita establecer una información para realizar el reparto de cartas, por lo que la forma de hacerlo es realizando el reparto dentro de Unity e informando a la plataforma de agentes sobre la información inicial del juego. En dicha información se encuentra el turno del jugador inicial, que ha sido previamente calculado en la plataforma de Unity

Por lo tanto el esquema básico de comunicación entre plataformas es el de la siguiente ilustración

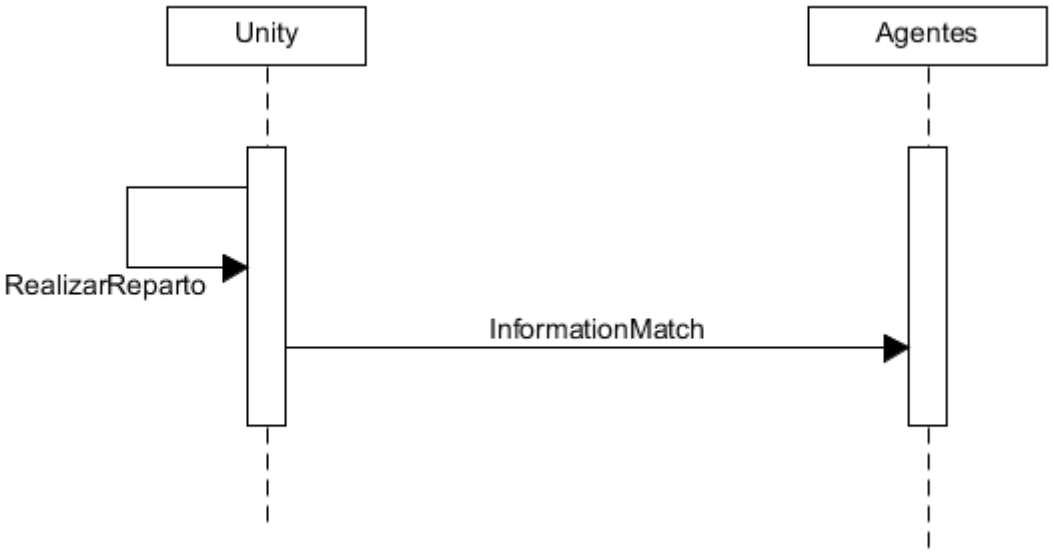

Ilustración 3.22 Comunicación de Unity con la plataforma de agentes para el reparto inicial

3.4.5 Como realizar y recibir movimientos

La forma de notificar los movimientos realizados desde unity consiste únicamente en enviar el movimiento en formato JSON a través de los sockets previamente configurados, siguiendo el esquema de interacción que se muestra a continuación.

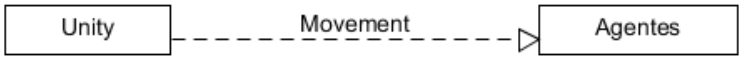

Ilustración 3.23 Comunicación de Unity con la plataforma de agentes para enviar movimientos

Sin embargo a la hora de recibir el movimiento es algo más complejo, la recepción de movimientos se realizara siempre dentro del estado del oponente, en dicho estado se contiene un patrón estrategia encargado de la recepción de los mensajes, cuando el controlador de los agentes reciba el movimiento, notificara a todas las clases que estén suscritas a un tópico identificado por el propio identificador de la partida, la cual dicha información viaja siempre junto con el movimiento realizado. Por lo tanto la interacción para recibir movimientos quedaría de la siguiente manera.

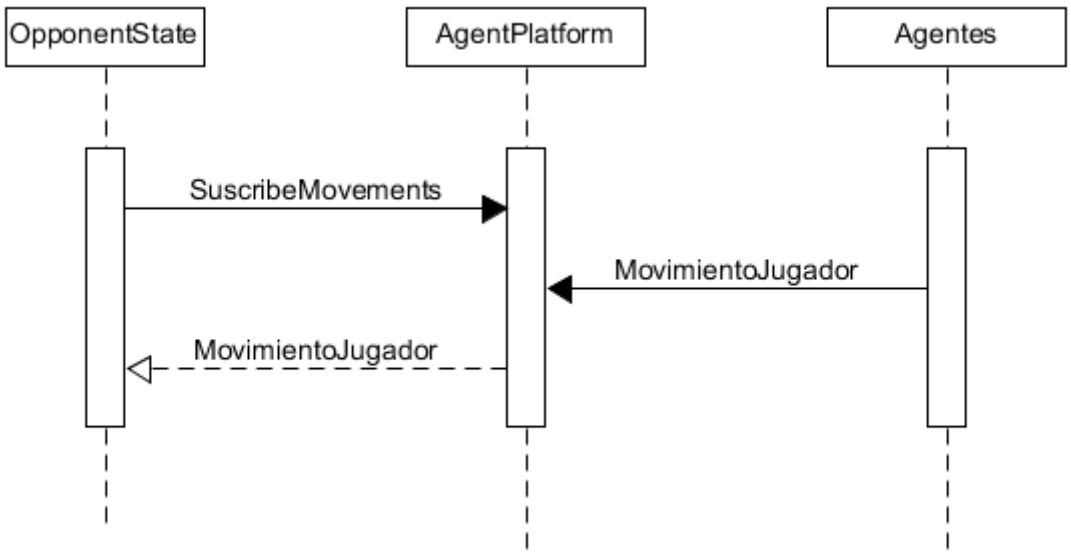

Ilustración 3.24 Comunicación de Unity con la plataforma de agentes para recibir movimientos

3.4.6 Como realizar el reparto de cartas después de una ronda de juego

Para realizar esta operación es necesario realizar las comprobaciones de que todo está funcionando correctamente, por lo tanto todos los movimientos de robar después de ronda que hayan sido ejecutados en la plataforma de agentes deben de ser notificados a la plataforma de Unity, con el fin de continuar con la integridad de la partida, el propio jugado de la plataforma realizara el movimiento de robar de forma automática, por lo tanto el jugador solo debe esperar a que finalice el reparto después de la ronda. La interacción entre las distintas plataformas se puede observar a través del siguiente esquema:

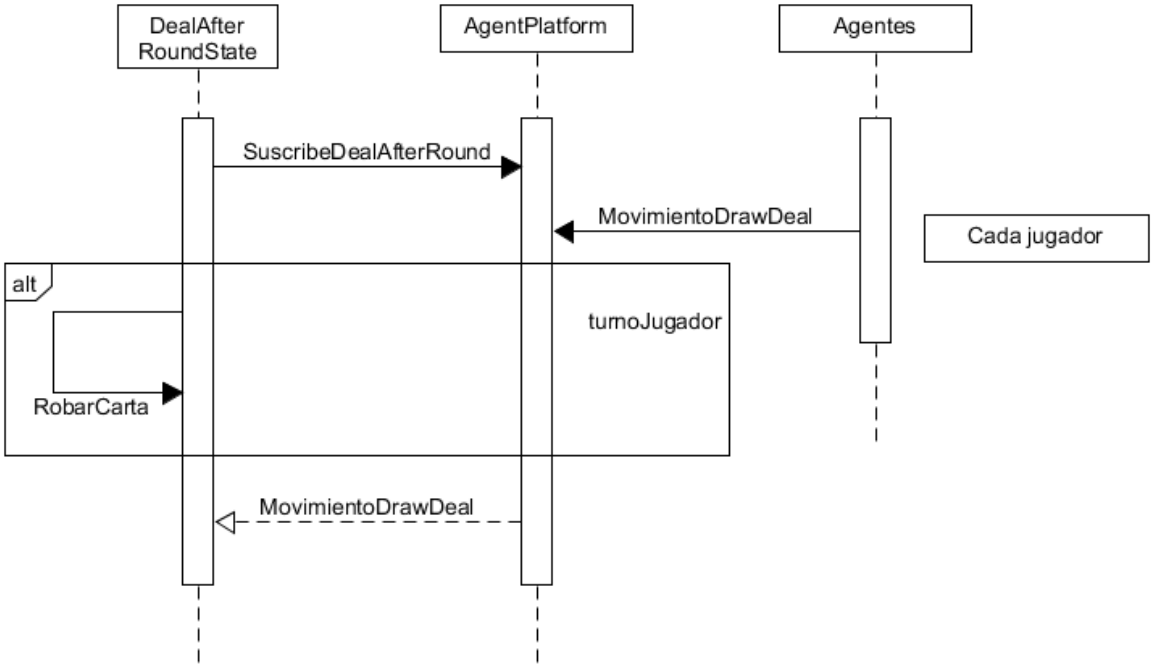

Ilustración 3.25 Comunicación de Unity con la plataforma de agentes para el realizar el reparto de cartas después de una ronda de juego

3.4.7 Como recibir el resultado de la partida

La recepción del resultado de la partida es bastante simple, lo único que hay que hacer es suscribirse al tópico del resultado de la partida y esperar a que se reciba un mensaje con el resultado.

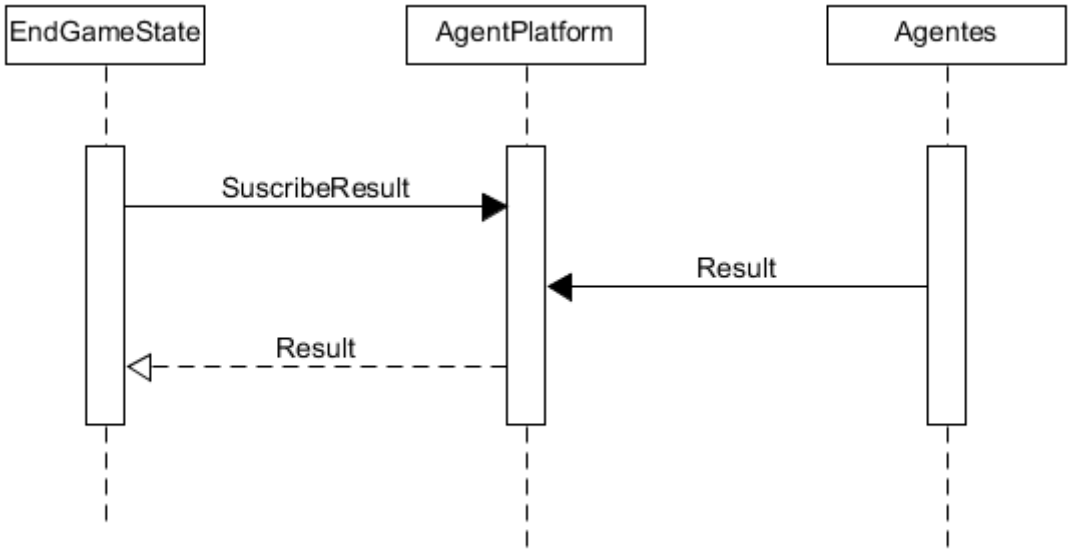

Ilustración 3.26 Comunicación de Unity con la plataforma de agentes para recibir el resultado de la partida

En esta iteración se ha implementado la estructura de comunicación con la plataforma de Firebase para llevar a cabo juegos de forma online, y con ello la creación y configuración de un proyecto Google Cloud para poder llevar a cabo correctamente el desarrollo de dichas partidas.

3.4.8 Implementando la API y los Event Source

Para la comunicación de elementos como ya se contempló en la segunda iteración haría falta hacer uso de una API, dicha api ha sido implementada bajo el paquete de peticiones web para las aplicaciones .NET (Microsoft, Microsoft | System.Net.Http, s.f.). Esta API proporcionará las operaciones básicas para interactuar con la base de datos, a través de los métodos GET, POST, PUT y DELETE

Otro aspecto necesario para la comunicación es la posibilidad de suscribirse a las modificaciones de los datos de la base de datos, para ello se ha utilizado como

base una librería externa (GitHub, s.f.), a la cual se le han realizado modificaciones para adaptarla mejor a las necesidades del proyecto.

3.4.9 Implementando la autenticación

La autenticación es un elemento fundamental para implementar un juego en línea, existen muchas ventajas de tener a los usuarios autenticados, como la integridad del juego, personalización de cuentas, protección de las mismas…

La autenticación es una forma de agregar una capa más de seguridad a los videojuegos. Además Firebase te permite modificar las reglas de seguridad de la base de datos, por lo tanto solo se permitirá la escritura y lectura en la base de datos para los usuarios que estén autenticados. Para ello solamente se deben de modificar las reglas de la base de datos como se ve en la imagen.

Ilustración 3.27 Captura de las reglas de seguridad de Firebase

Para implementarla solamente ha sido necesario instalar el módulo de la autenticación dentro del panel de Firebase y después habilitar los tipos de autenticación que se desean para el videojuego, en este caso, los jugadores solo se podrán autenticar con correo o de forma anónima.

Como ya se estudió en la segunda iteración se necesitaba de una serie de datos para realizar la autenticación, cada una tiene sus propias características y datos de entrada necesarios para realizar la misma, pero todas devuelven unos datos similares los cuales son necesarios para realizar las peticiones autenticadas. Es por ello que el patrón estrategia unifica todas estas características de la autenticación y se usara dependiendo de las entradas del usuario. Las estrategias de autenticación identificadas han sido las siguientes:

- Iniciar sesion con correo electronico y contraseña: En las que se necesitaran como datos de entrada el email y la contraseña.
- Registrarse con correo y contraseña: En la que se necesitaran como datos de entrada el email y la contraseña.
- Iniciar sesion de forma anonima: En la que no se necesitaran datos
- Iniciar sesion a traves de un refresh token: Cada vez que un usuario se autentifica le devuelve un refresh token, se puede registrar iniciar sesion con este dato con una cuenta previamente creada. Este tipo de autenticacion se utilizara con el fin de volver a utilizar los inicio de sesion de anonimos para no saturar la plataforma de usuarios.

Para realizar la autenticación Google ofrece bastantes posibilidades, en el caso de este proyecto se ha optado por la autenticación a través de la API (Firebase, Firebase | Auth API, s.f.)

3.4.10 Implementando el manejador de Firebase

Al igual que con la plataforma de agentes debe de existir un elemento encargado de manejar las los aspectos relacionados con la plataforma para desarrollar juegos de forma online.

Esta clase creara un *EventSource* para cada tópico que reciba de suscripción y utilizara los métodos de la API para realizar operaciones específicas, las cuales si serán accesibles para el resto de clases. Además de esto también proporcionara acceso a los diferentes tipos de autenticación, el diagrama UML de este manejador se puede observar en la ilustración siguiente

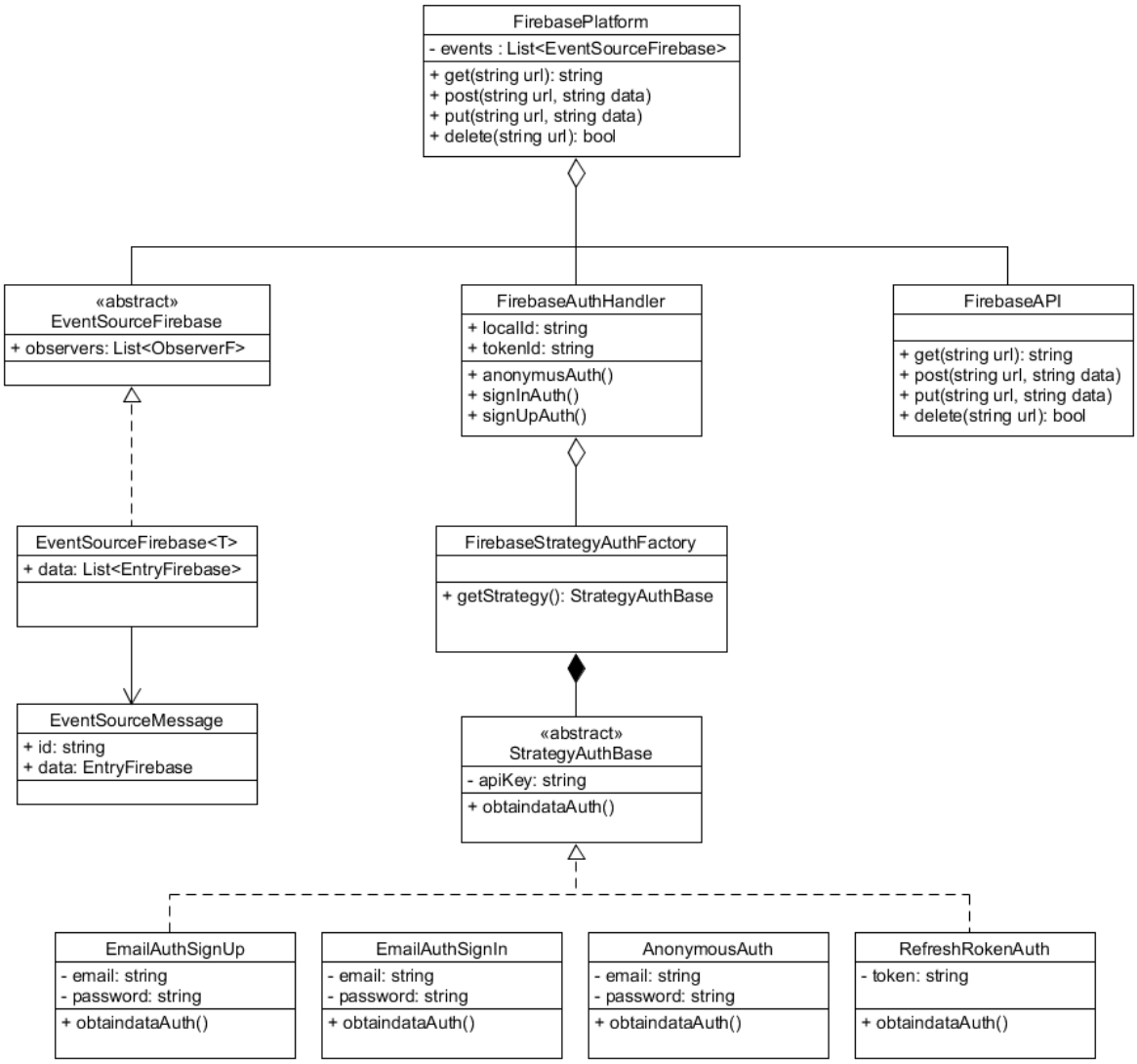

Ilustración 3.28 Diagrama de clases de la plataforma de Firebase en Unity

Para apuntarse a la lista del matchmaking será necesario publicar en la jerarquía del documento referente al matchmaking la información del jugador, después de apuntarse el jugador se mantendrá en la escena de carga hasta que se complete el matchmaking por completo, la estructura de comunicación es la que se puede observar a continuación.

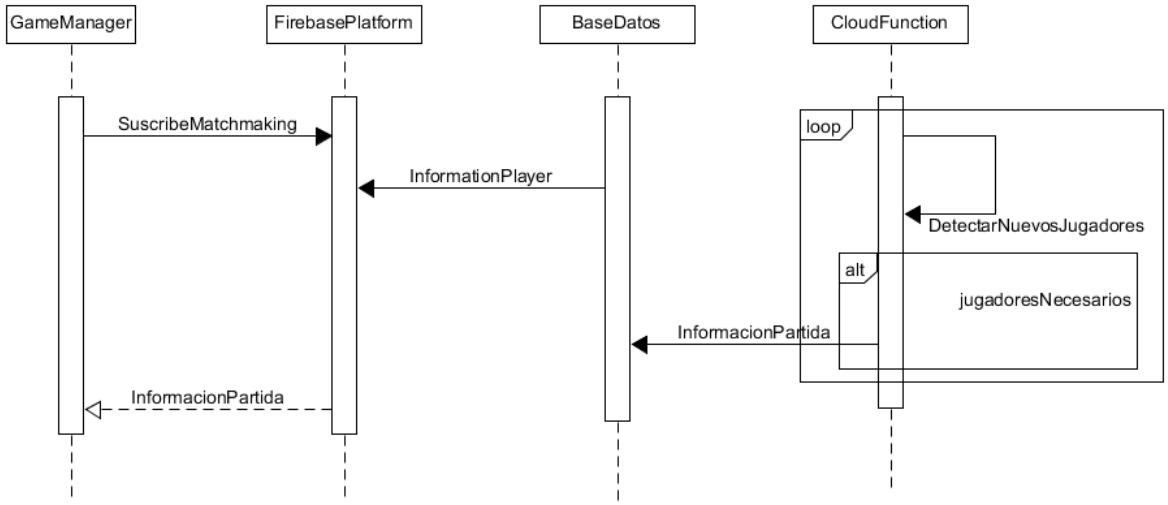

Ilustración 3.29 Comunicación de Unity con la plataforma de firebase para apuntarse al matchmaking

En la imagen anterior se puede observar que quien realmente realiza el matchmaking son las funciones en la nube, la base de datos meramente es un intermediador entre Unity y las funciones de Google.

Una vez se haya completado el matchmaking el jugador deberá modificar los datos de dicha partida publicada en el matchmaking, la finalidad de realizar esta operación es que si cierra la aplicación y vuelve a ejecutarla nuevamente si recibe los datos de la partida y el propio jugador está dentro de la misma sepa que es de una partida anterior y no de una partida nueva.

3.4.11 Como realizar el reparto inicial

En la información de la partida el jugador recibió el identificador de la partida, dentro de este documento se encuentra toda la información necesaria para comenzar una partida, por lo que la interacción se limitara a obtener los datos de la base de datos de Firebase.

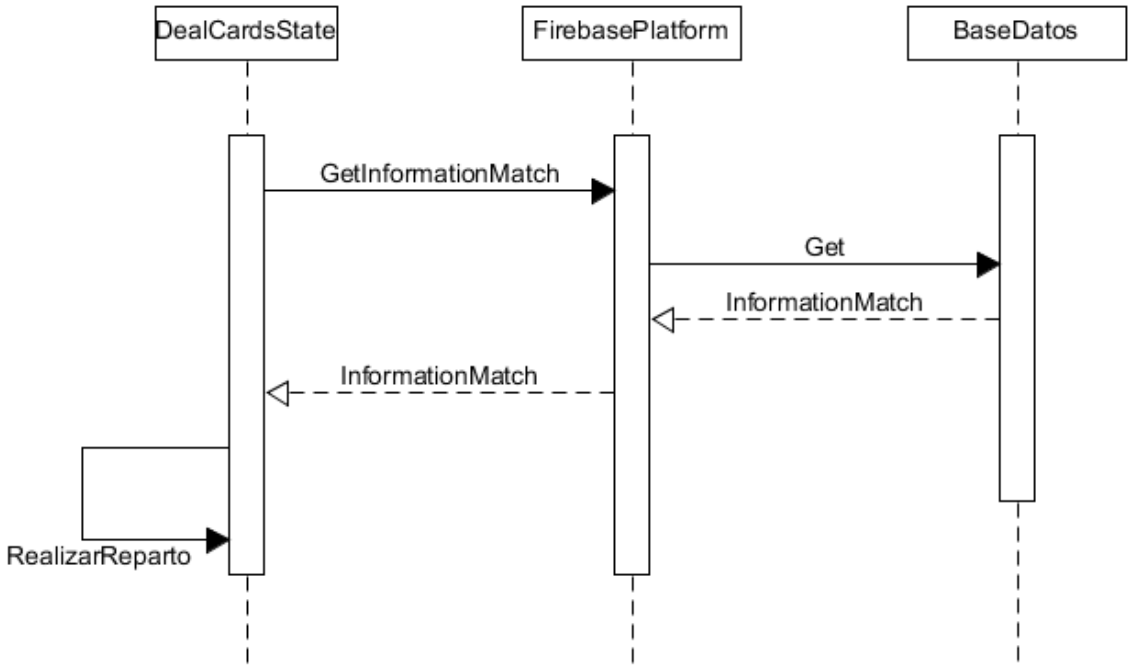

Ilustración 3.30 Comunicación de Unity con la plataforma de firebase para el reparto inicial

Cabe destacar que las cartas en la base de datos no se almacenan por palo y numero como se vio en la primera iteración, estas están almacenadas simplemente con su valor numérico, por lo que para transformarla a cartas se debe de realizar la operación inversa que vimos anteriormente, las operaciones a realizar son las siguientes:

$$rank = numeroCarta \,\%\, cantidadCartasPalo$$

$$suit = \frac{numeroCarta}{cantidadCartasPalo}$$

Ecuación 3.5 Transformaciones para obtener una carta a través de un número

3.4.12 Como realizar y recibir movimientos

La forma en la que se notificara los movimientos tiene una particularidad y es que este movimiento se comunicara a las Cloud Fuction y esta actualizara la base de datos para que el resto de jugadores reciba el movimiento realizado. De esta forma se tiene mucho mayor control sobre los movimientos, así dependiendo del tipo de juego se almacenaran los datos del movimiento realizado en unos documentos u otros dentro de la propia base de datos a través de las Cloud Functions. A continuación se mostrara un esquema simplificado de la comunicación.

.

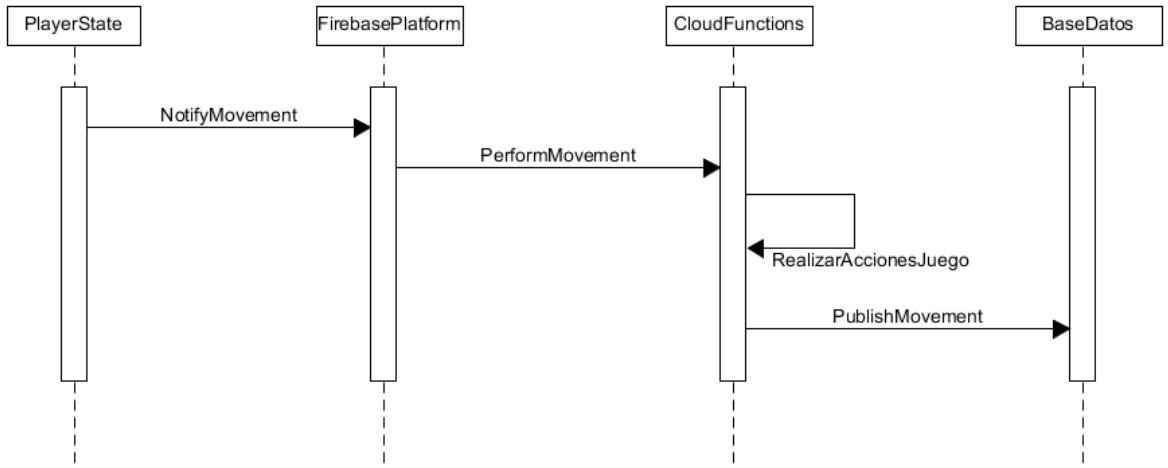

Ilustración 3.31 Comunicación de Unity con la plataforma de firebase para enviar movimientos

Sin embargo la recepción de movimientos sigue una estructura similar a la plataforma de agentes, en el estado del turno del oponente el jugador espera a la recepción de un movimiento de la plataforma al cual está suscrito mediante el patrón observer. La única diferencia es que se deberá de realizar la transformación de la carta como se mostró en el apartado anterior en el caso de que el movimiento sea el de lanzar carta.

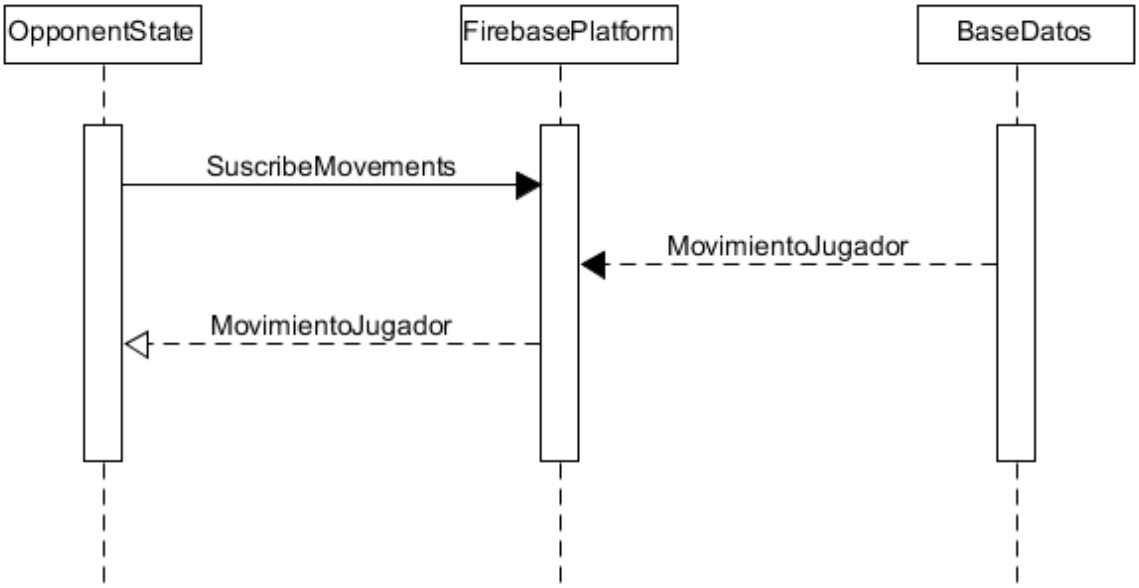

Ilustración 3.32 Comunicación de Unity con la plataforma de firebase para recibir movimientos

3.4.13 Como realizar el reparto de cartas después de una ronda de juego

Para el caso de reparto de cartas después de una ronda, el recibo de cartas es simple, lo que se debe de hacer es estar suscrito a un tópico del documento el cual almacenara las cartas que le corresponden al jugador al finalizar una ronda, este tópico se irá actualizando conforme pase las rondas con las nuevas cartas del mazo.

Dichas cartas se encuentran almacenada en otra parte de la base de datos como ya se vio en el diseño y son las Cloud Funtions las encargadas de actualizar este documento conforme van realizando los movimientos los jugadores de la partida.

Se muestra el esquema de comunicación a la hora de recibir el reparto de cartas

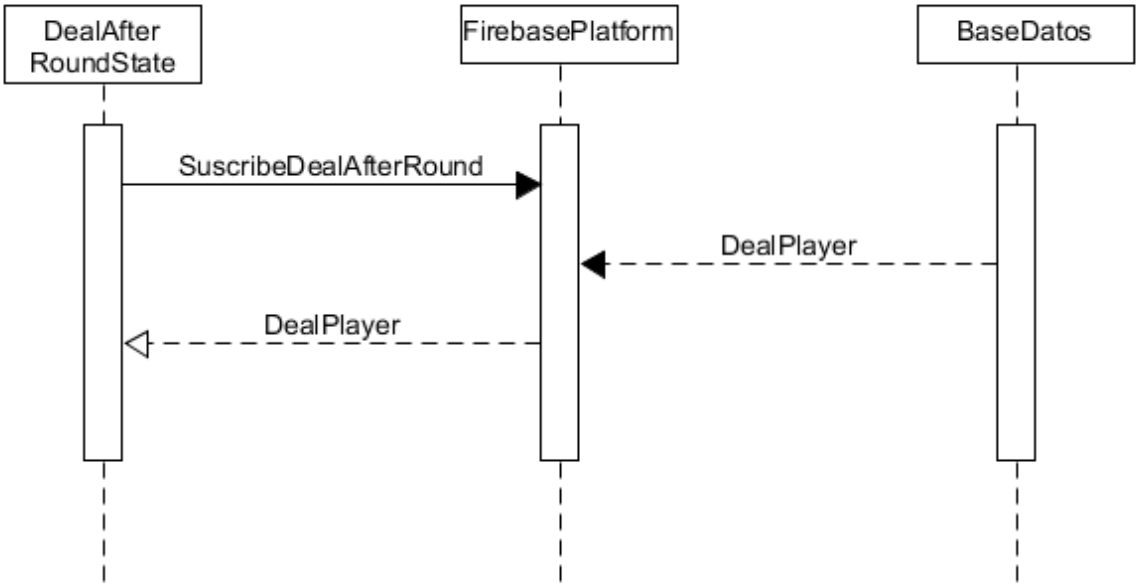

Ilustración 3.33 Comunicación de Unity con la plataforma de firebase para realizar el reparto de cartas después de una ronda de juego

3.4.14 Como recibir el resultado de la partida

Por último se debe recibir los resultados de la partida, como esto se almacena en otra parte del documento, la forma de recibir la información del resultado se realiza exactamente igual que con la plataforma de agentes. La única diferencia radica en las plataformas externas de comunicación. De nuevo el resultado será generado por las funciones de Google, para ello cada vez que un jugador realiza un movimiento comprueba si se cumple la condición para finalizar el juego, si es así publicara el resultado en la base de datos y automáticamente los jugadores suscritos recibirán el resultado.

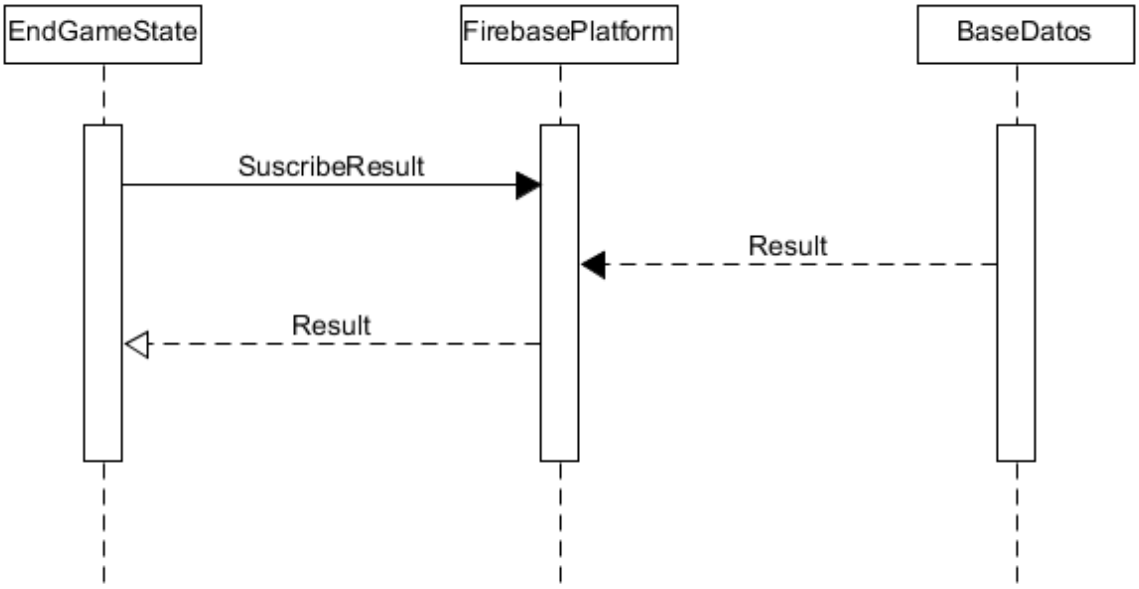

Ilustración 3.34 Comunicación de Unity con la plataforma de firebase para recibir el resultado

3.4.15 Detectando la caída de los jugadores adversarios

Cuando se está desarrollando una partida de forma online, es necesario cerciorarse de que todos los jugadores estén jugando la partida, sino una partida nunca finalizaría, dado que estaría siempre a la espera de que el jugador que abandono realice su movimiento correspondiente. Es por ello que se ha implementado un mecanismo, a través del cual el jugador cuando cierre su aplicación por cualquier motivo y se caiga de la partida, notifique a la propia base de datos de su abandono. Una vez hecho esto los demás agentes recibirán esta información y la partida finalizara como una partida cancelada, en la cual no habrá ganadores ni perdedores, dado que dicha partida no se ha podido finalizar como se esperaba.

Este aspecto solo se debe de tener en cuenta en las partidas en línea, ya que si un jugador cierra la aplicación en una partida offline, de lo único que debe asegurarse es de cerrar la plataforma de agentes correctamente, como se mostró en el apartado anterior en esta documentación.

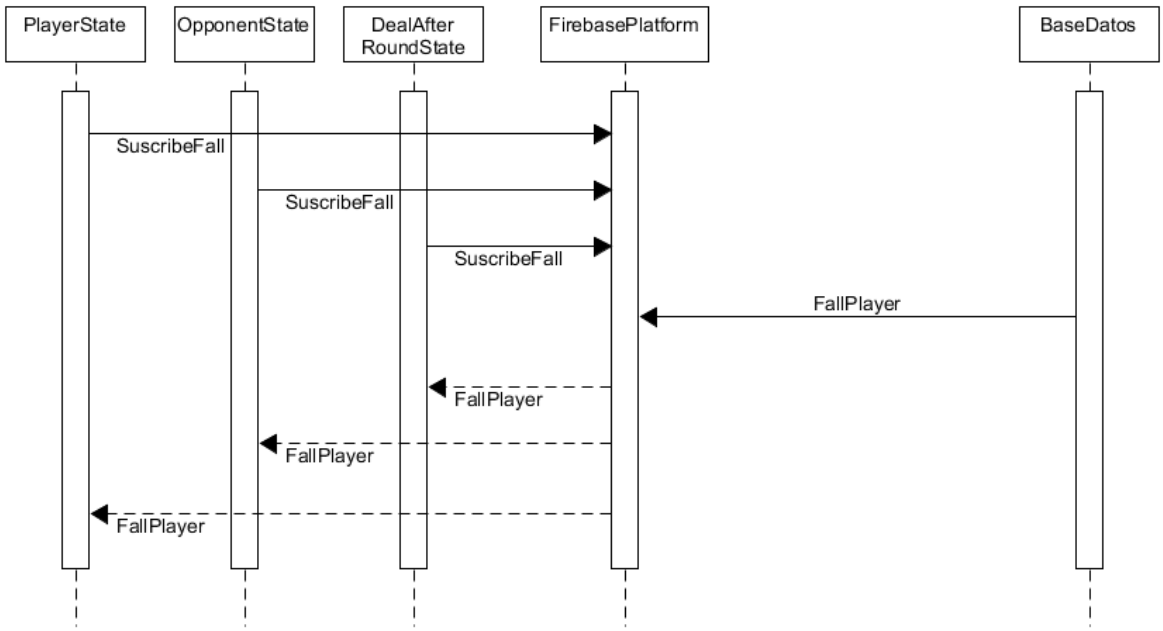

Ilustración 3.35 Esquema de comunicación para la detección de la caída de los jugadores

Es necesario detectar la caída en cualquier estado en el que se está desarrollando la partida, para finalizar correctamente la misma.

3.5 Sexta iteración

En esta iteración se introducirán aspectos referentes a la mejora visual de juego, junto con la interfaz de usuario, la cual es imprescindible para utilizar el juego de forma adecuada.

3.5.1 Implementando diferentes tipos de juego

Cada juego tiene una base de reglas las cuales se irán ejecutando a lo largo de las máquinas de estado, con el fin de poder implementar juegos con mayor facilidad en el futuro y que la claridad del código sea la máxima posible, se han agrupado las posibles diferencias que pueden existir entre distintos tipos de juego, en distintos patrones estrategia.

Se han identificado las siguientes estrategias para poder implementar cualquier tipo de juego:

- IStrategyDraw: Cada juego implementara esta estrategia, indicando si para dicho juego se pueden robar cartas o no
- IStrategyPassTurn: Cada juego implementara esta estrategia, indicando si para dicho juego se puede realizar la operación de pasar o no
- IStrategyPlayableCards: En cada juego se dispone de la información de las cartas de la mesa y cartas de la mano del jugador, en base a esto dicha estrategia calculara cuales son las cartas que pueden ser lanzadas en base a las reglase del juego que se esté llevando a cabo.
- IStrategyGameEndCondition: Cada juego tiene una determinada condición para finalizar la partida, la cual se ejecutara en los estados correspondientes, dicha condición debe ser implementada por cada tipo de juego.
- IStrategyDecidePlayerInitialTurn: Cada juego implementara esta estrategia para decir que jugador debe comenzar en cada turno, dicha estrategia, dicha estrategia solo se ejecutara en partidas offline, ya que si se dispone de toda la información necesaria.
- IStrategyUpdateMatchTurn: Dicha estrategia será la encargada de actualizar los turnos para un determinado tipo de juego, aunque normalmente siempre se pasa al siguiente jugador se podría dar el caso de

que un jugador tuviera que realizar dos movimientos en el mismo turno, por lo tanto para manejar todas estas posibles casuísticas el turno se actualizara mediante dicha estrategia.

Por lo tanto cada vez que se implemente un nuevo juego la incorporación consistirá en agregar nuevas clases que implementen estas estrategias, el resto ya está implementado para que se ejecute con normalidad en el juego.

3.5.2 Implementando la interfaz

Una vez se ha diseñado la interfaz se procede a su realización en el motor del juego y posterior inclusión en el juego. En Unity para la realización de interfaces se dispone del sistema UI (Unity, Unity | UI, s.f.) que permite crear interfaces rápidas e intuitivas. Este sistema permite crear elementos de interfaz como menús, que luego se muestran a los usuarios

3.5.2.1 Patrón Mediator para comunicarse con los controladores

Las acciones que realiza el usuario sobre la interfaz están estrechamente ligadas a las acciones que deben realizar los controladores y manejadores dentro del propio juego. Por esta razón se ha pensado en la implementación del patrón Mediator, el cual permite comunicar los cambios en una parte al resto de componentes mediadores.

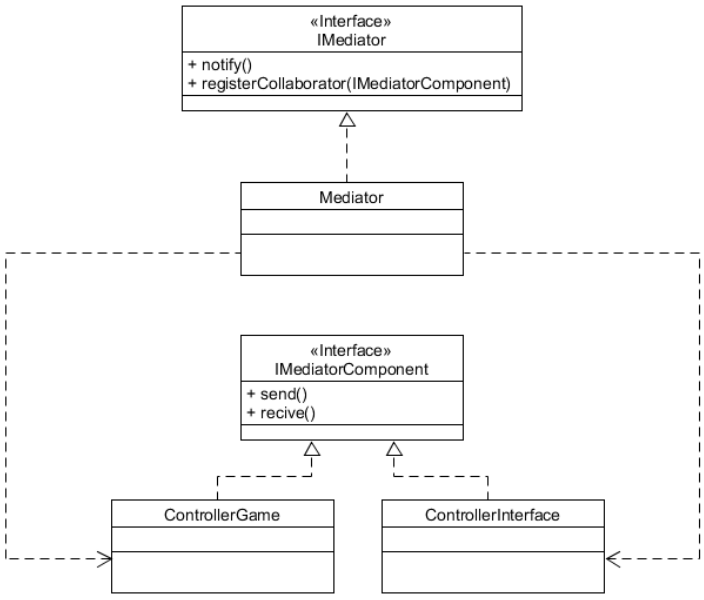

Ilustración 3.36 Diagrama de clases del patrón Mediator para las interfaces

Con esta estructura se actualizarán los datos en cadena dependiendo de las acciones realizadas por el usuario, u otras acciones del juego que requieran cambios en la interfaz. La gran ventaja de este patrón es la centralización de la comunicación, en la cual se puede distinguir el contenido del mensaje y pasar la información a los componentes que la requieran.

3.5.2.2 Arquitectura MVVM – Reactiva

Para el funcionamiento de la interfaz se ha destina un esfuerzo extra para realizarla bajo un patrón arquitectónico, la ventaja de realizarlo de esta forma es que existe una separación de responsabilidades más amplia, y con ello un desacoplamiento mayor del código, lo cual permite introducir nuevos elementos a la interfaz de una forma más simple.

Las interfaces implementadas en este proyecto siguen el patrón arquitectónico MVVM – Reactivo (Ups, s.f.), es una modificación del MVVM habitual el cual combina varios elementos de distintos patrones, la arquitectura de la interfaz es la que se puede observar en la siguiente imagen

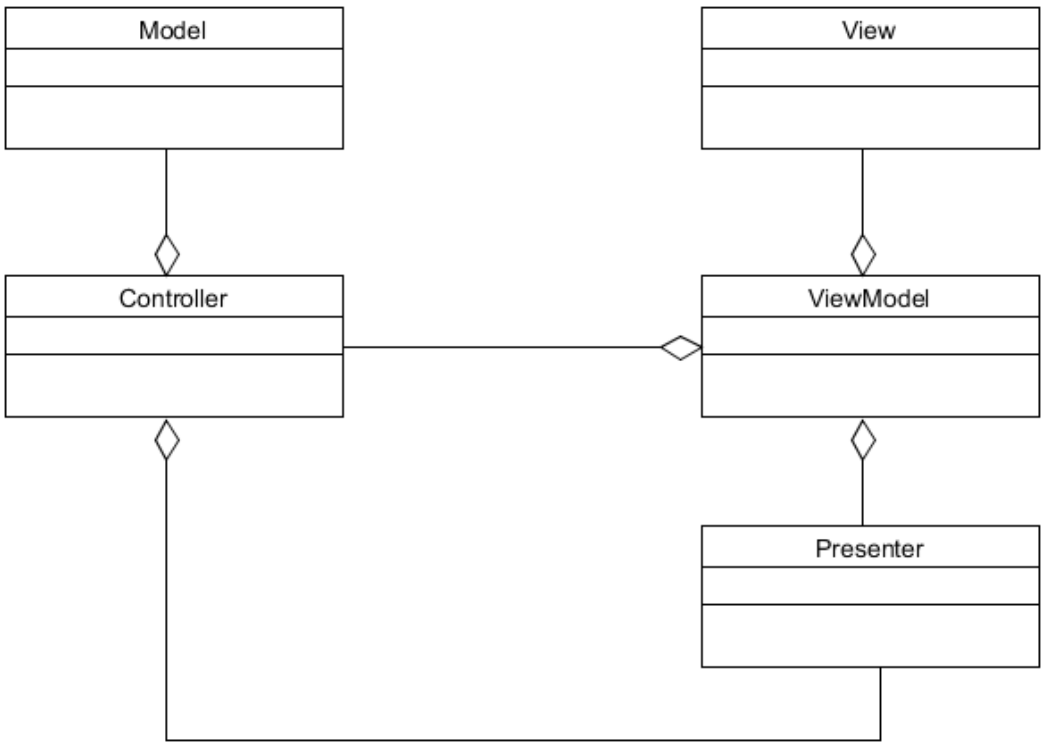

Ilustración 3.37 Diagrama de clases de las clases bases utilizadas para la interfaz

En dicha estructura se puede identificar los siguientes elementos:

- Modelo: En dicho apartado se puede considerar cualquier elemento que almacene los datos sobre una entidad, como podría ser la configuración del juego actual.
- Controller: Es la clase encargada de controlar toda la coordinación entre las distintas clases, la cual invocara los eventos necesarios para que desencadenen en una reacción en la interfaz.
- Presenter: Es la clase encargada de obtener los datos a través del controlador o de la modificación realizada en el modelo, la interpreta y la asigna de la forma correcta al *ViewModel.*
- ViewModel: Almacena los datos sobre la situación actual de la vista, almacena el contenido de los textos, que elementos son visibles y cuales no lo son…, los atributos de estos elementos suelen ser reactivos, lo cual quiere decir que si se modifica su valor se lanza un evento para notificar a los observadores que serán las vistas, sobre el cambio que se ha realizado
- View: Almacena las referencias a los elementos de la interfaz, como botones, paneles y asigna los datos obtenidos del ViewModel

3.5.2.3 Asignando los datos de la localización

Los datos de localización son manejados por el LocalizationManager, cuando los elementos ViewModel de las interfaces son instanciados obtienen los datos a través de este manager, por lo tanto la View, tomas los datos ya localizados, si surge alguna modificación que cambie la localización las clases Presenter, también podrán acceder a dichos datos y modificar los ViewModel para las propias vistas actualicen los datos nuevamente.

3.5.2.4 Imágenes de los botones

Se ha utilizado el software de Gimp (Gimp, 2023), para crear imágenes para los botones, en este apartado se explicara el proceso artístico y el porqué de la utilización de dichas metáforas.

3.5.2.4.1 Ideas desechadas

Los iconos se buscaban que fueran los más claros, simples y concisos posibles, para que de esta forma sea reutilizable en distintos tamaños y más fácil de implementar para el propio alumno. Por lo tanto se desecharon las siguientes ideas que se pueden observar en la siguiente ilustración

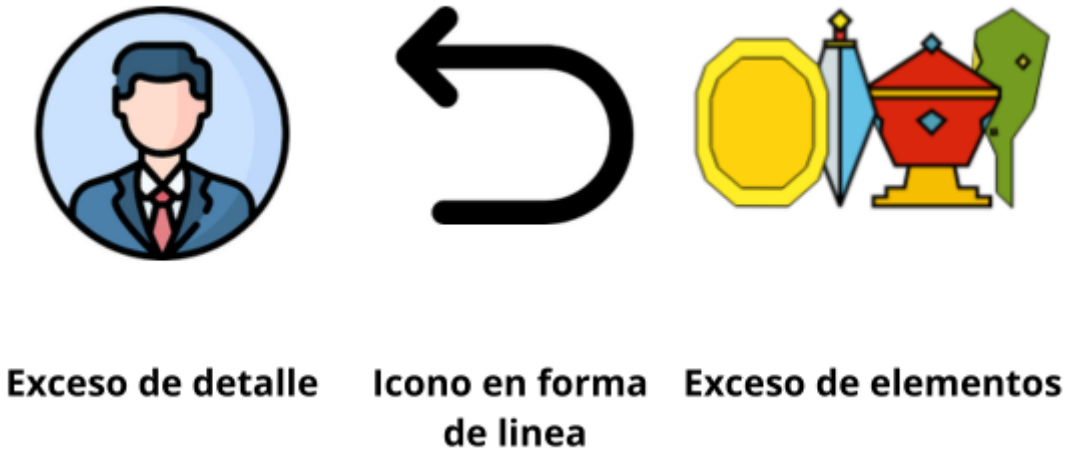

Ilustración 3.38 Ideas desechadas durante el desarrollo de los iconos

3.5.2.4.2 Imagen de perfil

Esta imagen se ha realizado pensando en la metáfora que la mayoría de las asocia al perfil de usuario, se dibujó sin bordes dentro del icono para que no formara ninguna imagen extraña ni produjera efecto de diente de sierra. En la siguiente ilustración se puede observar el resultado final de la imagen utilizada para el botón de perfil

Ilustración 3.39 Imagen del perfil de usuario

3.5.2.4.3 Imagen de volver

Hay distintas metáforas para indicar la vuela al lugar de origen, pero la más común y la más usada es la de una flecha indicando la dirección contraria a la pantalla, de esta forma al hacer clic sobre este botón el usuario sabrá que volver a la pantalla de la que ha llegado al punto actual de la interfaz.

Ilustración 3.40 Imagen de volver a la interfaz anterior

3.5.2.4.4 Imágenes para el cambio de textura

Para este botón se han implementado han asociado los palos de las cartas como identificadores del propio tipo de mazo, de esta forma el jugador asociara la metáfora de una moneda, con el palo de oros, y a su vez con la baraja española, lo mismos ocurre al ver un rombo, que lo asocia con el palo de rombos y a su vez con la baraja francesa.

Se han decidido utilizar estos dos palos porque son los más fáciles de distinguir para distintas resoluciones, además estos dos elementos están bastante más relacionados de lo que puede parecer a simple vista, ya que a las personas que le gusten las cartas y estén informados sobre la historia de las mismas sabrán que cada palo de la española está relacionado con otro palo de la francesa y el palo de rombo y el palo de oros representan el mismo palo en distintas barajas (Fournier, s.f.)

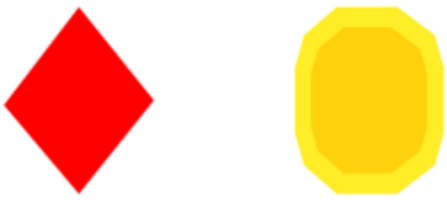

Ilustración 3.41 Imágenes que representan la baraja francesa y española

3.5.2.5 Resultado final de la interfaz

Para la implementación de la interfaz se ha necesitado utilizar dos tipos de Canvas (Unity, Unity | Canvas, s.f.)

El primer tipo de Canvas se renderiza en el espacio de la pantalla. El contenido que se crea puede ser adaptado en función de la resolución del dispositivo y del tamaño de la pantalla en la que se esté reproduciendo el juego. Una vez creado el Canvas, hay que configurarlo para que escale con la resolución (Ilustración 3.38). Esta configuración será utilizada para todos los Canvas presentes en el videojuego que requieran una renderización en el espacio de la pantalla.

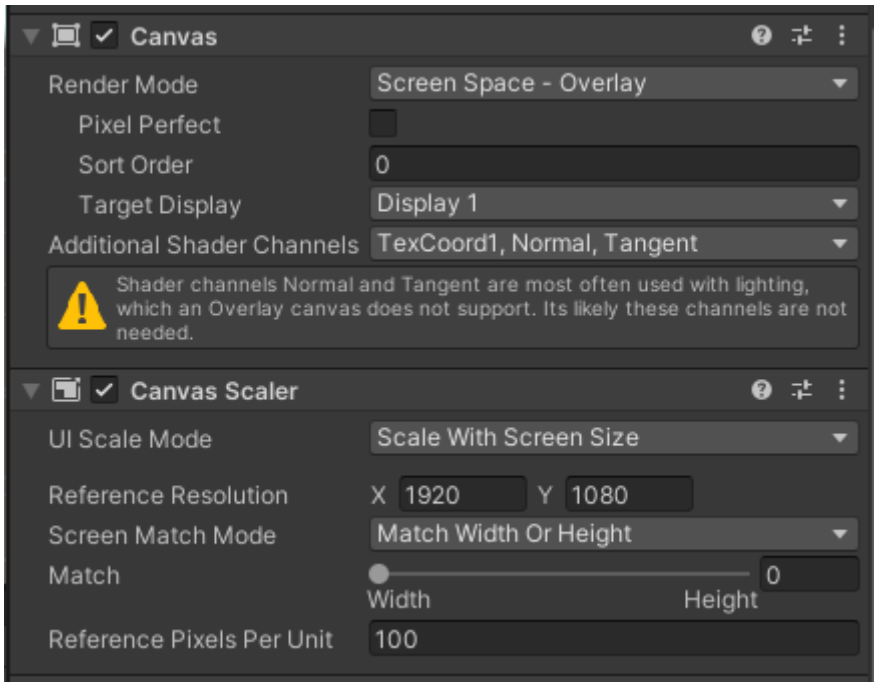

Ilustración 3.42 Captura de la configuración de un Canvas en el espacio de la pantalla

Los Canvas que utilizaran este tipo serán los de la interfaz del menú principal, encargado de realizar la selección del juego y la interfaz de la sesión de juego la cual permite modificar el tipo de vista y las imágenes de las cartas. Dichos Canvas también contendrán un script denominado DependencyInyector, el cual asigna los elementos de la interfaz y los hace accesible a los scripts, que en este caso serán las View, comentadas en el apartado de MVVM las cuales utilizaran dichas clases.

Por otro lado tenemos el tipo de Canvas del espacio del mundo, este se utiliza únicamente para la información que se muestra de los jugadores en una sesión de juego, de este modo los elementos dentro de este Canvas se trataran como cualquier

GameObject interno del juego para darle una posición dentro del mundo en 3D, dicho Canvas también requiere una determinada configuración y es la que muestra en la siguiente imagen.

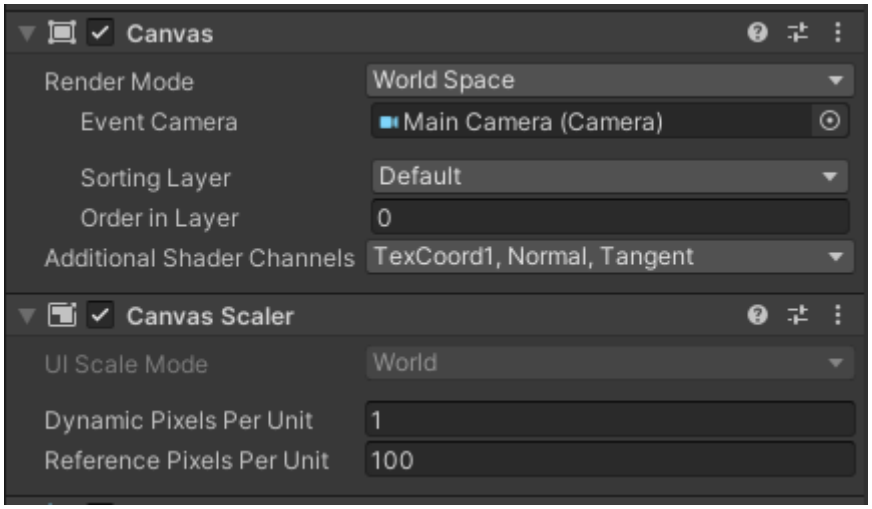

Ilustración 3.43 Captura de la configuración de un Canvas en el espacio del mundo

En la parte de análisis se identificaron una serie de interfaces, seguidamente se mostrara el resultado final de diseño de cada interfaz.

3.5.2.5.1 Inicio sesión

Para la realización de la interfaz ha sido necesario el uso de un *VerticalLayout*, el cual encaja todos los componentes dentro de la propia interfaz

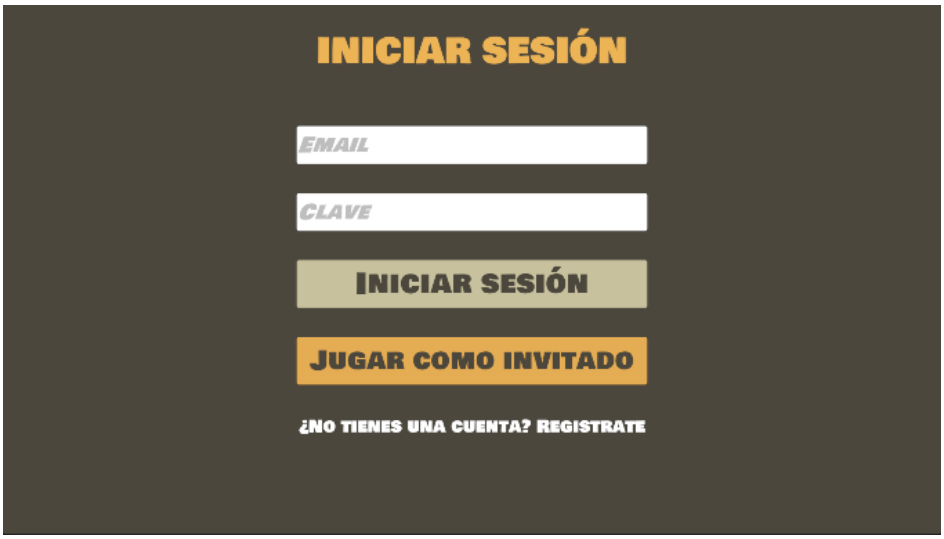

Ilustración 3.44 Resultado interfaz inicio de sesión

3.5.2.5.2 Registro

La implementación de esta interfaz es similar al anterior, con un *VerticalLayout* todos los componentes internos siguen la estructura necesaria.

Ilustración 3.45 Resultado interfaz registro

3.5.2.5.3 Perfil de usuario

Esta interfaz es la más compleja de implementar, dicha interfaz ha necesitado de *Vertical y HorizontalLayout* para diferenciar correctamente las zonas y que todas ellas siga unos márgenes previamente establecidos.

Ilustración 3.46 Resultado interfaz perfil

3.5.2.5.4 Selección modo de juego

Para la selección de modo de juego solamente se ha necesitado un *HorizontalLayout* para colocar correctamente los botones de la partida.

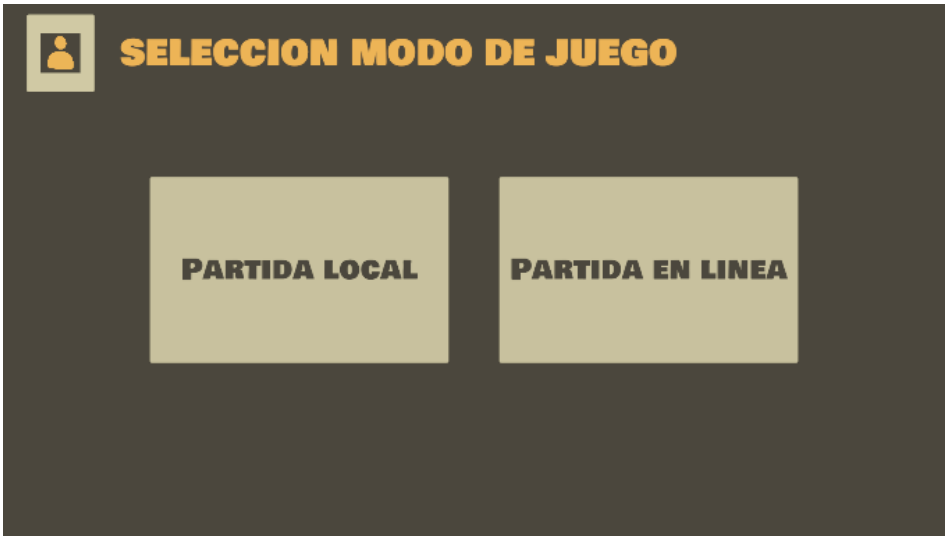

Ilustración 3.47 Resultado interfaz selección modo de juego

3.5.2.5.5 Selección tipo de juego

Para dicha interfaz se ha utilizado un *GridLayout* el cual almacenara los botones en forma de malla y un Horizontal y *VerticalLayout* para la parte inferior de la interfaz, en la cual se muestra la información del juego seleccionado

Ilustración 3.48 Resultado interfaz selección de tipo de juego

3.5.2.5.6 Sesión de juego

Para la implementación de esta interfaz se ha usado el Canvas del mundo para la información de los jugadores y un *HorizontalLayout* para los botones de cambio de visión y textura del mazo.

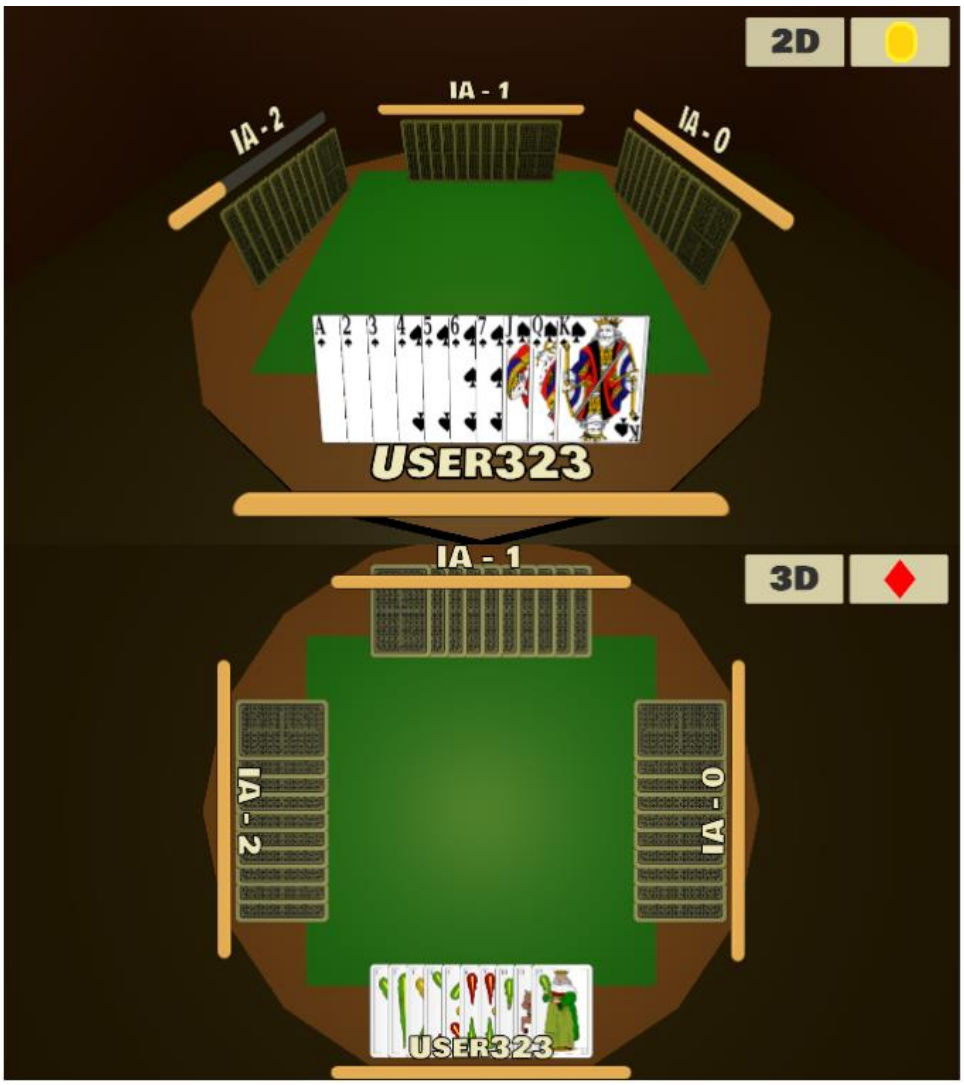

Ilustración 3.49 Resultado interfaz sesión de juego

En dicha interfaz se han agregado nuevos elementos, como los nombres de los jugadores o un indicador de tiempo para informar sobre el tiempo del que disponen los jugadores para realizar un movimiento. Este elemento será bastante importante para las partidas en línea, para que no se produzcan esperas demasiado largas mientras el resto de jugadores piensa su movimiento.

Para implementar este aspecto se ha utilizado un Slider (Unity, Unity | Slider, s.f.), al cual se le ira en decremento en base al tiempo que pase, para tener esa

presentación se ha eliminado el botón que trae por defecto el slider y se ha modificado el color.

3.5.2.5.7 Clasificación del juego

Esta interfaz se ha realizado nuevamente con *Horizontal y VerticalLayout* para mantener unos márgenes dentro de la propia clasificación.

Ilustración 3.50 Resultado interfaz clasificación del juego

Además dicha interfaz se mostrara con un formato distinto cuando el juego finalice por la caída de alguno de los jugadores de la partida, los colores cambiaran y la apariencia será como se muestra en la siguiente imagen.

Ilustración 3.51 Resultado de la interfaz de la clasificación de un juego cancelado

Este panel se hará visible cuando se reciba los datos de la clasificación, por lo que tiene requerimientos de creación dinámica, lo cual quiere decir que es diferente al resto de interfaces que siempre se esperan la misma cantidad de datos. Para implementar esta necesidad los componentes encargados de los layout permiten agregar una opción que es expandir los componentes en ancho y alto (Child Force Expand), de esta forma se puede agregar más o menos elementos a las clasificaciones de los juegos (Ilustración 3.52)

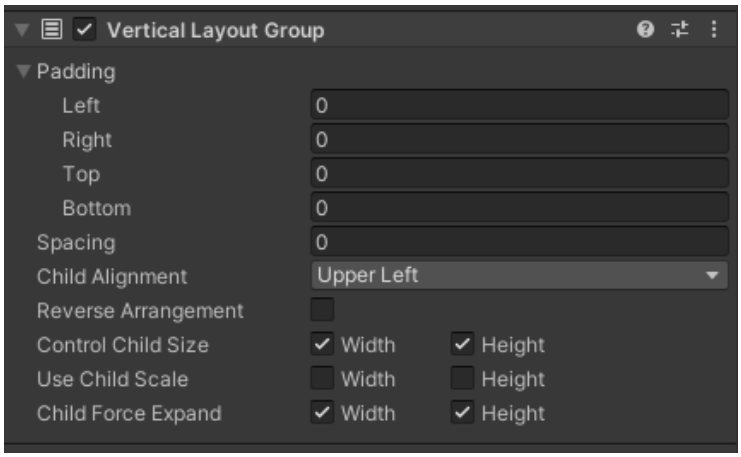

Ilustración 3.52 Captura configuración vertical layout para la clasificación del juego

También otra característica particular de este panel es que es el único que modifica sus colores en tiempo de ejecución dada las necesidades del proyecto. Por lo tanto para llevar a cabo esta situación se comprueba si la clasificación es nula, que en este caso será que la clasificación este conformada por puntuaciones que sean igual a 0 para todos los jugadores, algo que no será posible en partidas normales y si es así aplica este color distinto que se vio en la ilustración 3.51

3.5.3 Implementando las luces

Las luces serán utilizadas en la escena de la sesión de juego, para el desarrollo del videojuego han bastado con dos luces, para dar el ambiente de una pachanga en un bar.

La primera luz que se ha utilizado ha sido una luz puntual, esta luz focalizara sobre el tablero del juego, aunque indirectamente también emitirá luz sobre el resto de la escena.

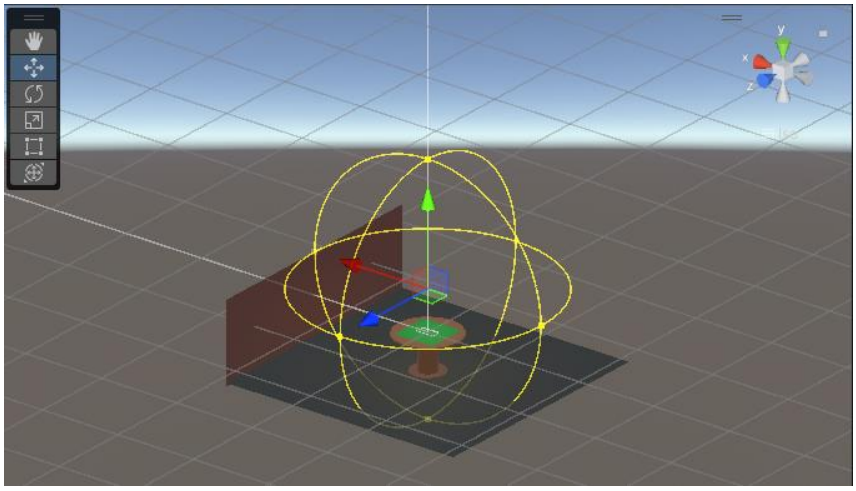

Ilustración 3.53 Luz puntual

Por otro lado se tiene la luz focal, esta luz simula una lámpara situada el techo del lugar donde se está desarrollando el juego de cartas, dando más luz sobre el centro, para que el juego se vea más iluminado e iluminando levemente paredes y elementos que se encuentran retirados del centro.

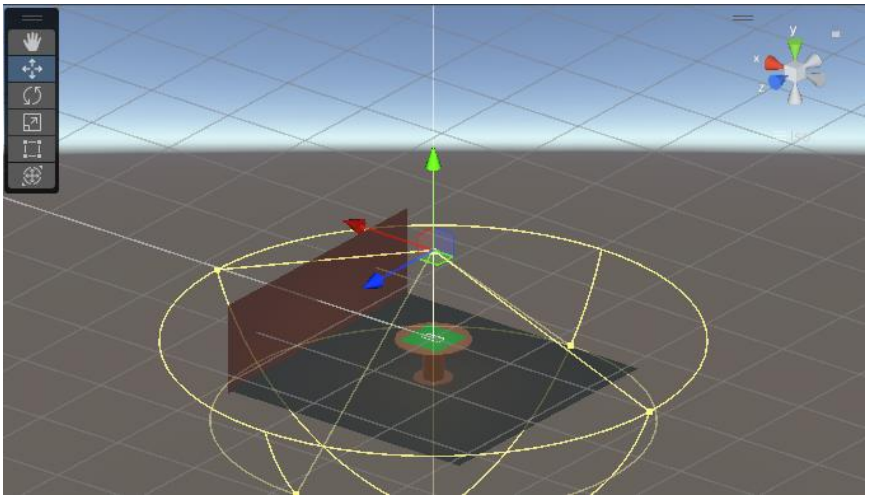

Ilustración 3.54 Luz focal

4 MODELO DE NEGOCIO

En este apartado se desarrollará la creación de una empresa que comercialice el producto desarrollado. La empresa se dedicará al desarrollo de videojuegos de cualquier índole.

Al tratarse de una empresa recién creada, esta se centrará principalmente en el desarrollo del presente proyecto.

4.1 Objetivos

Todo emprendedor que desee crear una nueva empresa, debe tener en cuenta los distintos tipos de objetivos, estos le servirán para saber cuál es el rumbo de la empresa y saber si la empresa está funcionando correctamente. Estos objetivos son los cualitativos y los cuantitativos:

4.1.1 Objetivos Cualitativos

Estos objetivos no pueden medirse discretamente, pero son bastante importantes para hacer que la empresa sea más atractiva a los clientes, en nuestra supuesta empresa implantaremos estos objetivos cualitativos:

- Desarrollo de videojuegos de calidad: De esta forma nuestra clientela estara satisfecha con el dinero invertido en nuestra empresa y no dudara en volver a utilizar nuestros serivicios.
- Prestigio y reconocimiento elevado: Cuando las personas vean que el videojuego ha sido desarrollado por nuestra empresa, esperan un videojuego de calidad, por ello este objetivo esta relacionado directamente con el anterior.
- Compromiso con responsabilidades sociales: Para acaparar mas numero de clientes las responsabilidades sociales son muy importantes, por ello se utilizaran equipos que cuiden el medio ambiente y paneles solares para gastar la menos luz posible.
- Satisfacion de los trabajadores: Este objetivo es indispensable para que la empresa funcione correctamente, por ello se dara la opcion de teletrabajo y se implantaran zonas de ocio en la empresa.

4.1.2 Objetivos Cuantitativos

Como su propio nombre indica son aquellos que se pueden medir de forma discreta, para la empresa vamos a establecer los siguientes objetivos cuantitativos:

- Maximizar los beneficios: Ya que uno de los aspectos mas importantes sera ganar dinero con el desarrollo de videojuegos.
- Minimizar los costes: Este objetivo va relacionado con el anteriror, si se consigue minimizar los costes, se conseguira ganar mas dinero, el uso de software libre es una opción para minimizar estos costes.
- Fidelizar los clientes existentes y lograr un aumento constante de los mismos: Esto ira relacionado con los objetivos cualitativos, por lo que para lograrlo también debemos centrarnos en ellos.
- Aumentar el valor de mercado de la empresa: De esta forma tendremos más dinero para invertir en bienes para la empresa.

4.2 Análisis del entorno

El entorno son todos los factores que rodean a la empresa, los cuales pueden ser tanto beneficiosos como perjudiciales para la misma, por lo que son decisivos para nuestra propia empresa y deben tenerse en cuenta desde la creación de la misma.

Para estudiar el entorno en el que la empresa desarrollara su actividad, hay que realizar el estudio a dos escalas.

- Macroentorno / Entorno General: En los que se tratan factores que afectan por igual a todas las empresas de una determinada región.
- Microentorno / Entorno Especifico: En los que se tratan factores más cercanos e influyen a un conjunto de empresas con características comunes.

4.2.1 Macroentorno

El primer concepto que tenemos que abordar es como encajaría nuestra empresa en España, para ello es necesario conocer el concepto de PYME (Pequeñas y Medianas Empresas), que es la forma en la que conocemos a todo aquello que no

son grandes empresas. Esta definición fue expuesta por la Unión Europea y España acogió estas reglas para aplicarlas (BOE, 2014).

Las condiciones para ser una PYME es tener menos de 250 trabajadores y que el volumen de negocio anual no exceda de los 50 millones de euros, o que el balance general no exceda de 43 millones de euros (IPYME, s.f.), por lo tanto, al crear nuestra propia empresa y no tener previsto esta cantidad de ingresos ni trabajadores, se puede afirmar que seremos una PYME.

En el entorno general se pueden diferenciar cuatro factores, los cuales se van a analizar a continuación, posteriormente se utilizará la técnica PEST, para valorar la situación actual a la hora de crear una empresa en España.

4.2.1.1 Factores políticos

La Dirección General de Industria y de las PYME (DGIPYME) proporciona programas para facilitar el emprendimiento, mediante financiación directa o acceso a las fuentes de financiación. Estas ayudas pueden ser tanto para la creación de nuevas empresas como para acceder a incentivos fiscales y ayudas para la seguridad social, además la plataforma IPYME te ofrece un buscador sobre las ayudas que puedes obtener para tu empresa (IPYME, s.f.).

Otro aspecto a tener en cuenta son los tipos impositivos que han de pagar las PYMES, estos son elevados, aunque este año las nuevas empresas pagaran el 23% en vez del 25% (Agencia Tributaria, s.f.). Este no es el único impuesto que pagan las empresas ya que para obtener bienes también existe el IVA que puede ser del 21%, 10% o 4% dependiendo del tipo de adquisición (Agencia Tributaria, s.f.).

4.2.1.2 Factores económicos

En lo que a factores económicos se refiere, España ha experimentado una tasa de crecimiento del PIB del 5,8% durante el año 2022 (Direccion General de Economia y Estadistica, s.f., pág. 4) y se espera que crezca un 2,3% para el año 2023 (Direccion General de Economia y Estadistica, s.f., pág. 6). Si hablamos de la tasa de inflación, durante el pasado se experimentó una tasa del 3.2% de inflación y para los próximos años esta tasa media se revisa al alza, en 0.4 puntos porcentuales más en el año 2023, llevando la inflación hasta el 3.6% y pudiendo alcanzar tasas de inflación del 4.3% en el año 2024 (Direccion General de Economia y Estadistica, s.f., pág. 8). Otro factor económico a tener en cuenta es la deuda pública que el año pasado se

estableció en el 133,2%, son cifras bastante altas que se esperan que vayan decreciendo con el paso de los años, para el año 2024 se espera que obtenga un valor del 106,9% sobre el PIB de deuda (Direccion General de Economia y Estadistica, s.f., pág. 35). Si nos centramos específicamente en el déficit público, España tampoco atraviesa por sus mejores momentos ya que en el año 2022 se estableció la cifra de un - 4,8%, lo cual indica que hubo más gastos que ingresos, y según los estudios estadísticos, parece que la cifra será negativa hasta el año 2025 en el cual se estima un déficit público del - 4,1% (Direccion General de Economia y Estadistica, s.f., pág. 35).

4.2.1.3 Factores socio-culturales

España tiene una población de 48 millones de habitantes (Instituto Nacional Estadistica, s.f.), con un índice coyuntural de fecundidad de 1,2 hijos por mujer y una esperanza de vida de 83 años (Instituto Nacional de Estadística, 2022). El hecho de que España tenga un índice de fecundidad inferior a 2,1 hijos por mujer supone que no se garantiza una pirámide de población estable (Datosmacro, s.f.).

Nuestro país se enfrenta a un reto de envejecimiento demográfico, lo cual implica un número mayor de pensiones, actualmente España cuenta con un total de 10 millones de pensionistas, lo que supone más del 20% de la población (Instituto Nacional de la Seguridad Social, s.f.), esta cantidad tan elevada de pensionistas representa un gasto del 11,7% del PIB (La Moncloa, s.f.). El envejecimiento no solo afecta al número de pensiones, si no que indirectamente también causa presión sobre el sistema sanitario, en el año 2021 se gastaron 88 millones de euros en sanidad (Ministerio de Sanidad, s.f., pág. 4).

En lo que respecta a la educación España tiene una tasa de alfabetización del 98,6%, lo cual la coloca en la posición número 39 de los países con mayor tasa de alfabetización (Datosmacro, s.f.). Incluso teniendo estos datos tan favorables las tasas de ingresos a estudios han aumentados en todas las etapas educativas, menos en educación infantil (Ministerio de Educacion y Formacion Profesional, 2023)

Otro factor a destacar es el estilo de vida de los españoles, según el estudio, mas del 70% valoran su satisfacción con un 7 o más, lo que supone un promedio final de 7,10. A esto también se le ha de sumar el grado de felicidad que es bastante elevado, alrededor de un 75% de españoles se consideran felices (AEGON, s.f.).

4.2.1.4 Factores tecnológicos

España es un país avanzado en materia de innovación y desarrollo tecnológico, en el año 2021, se destinó 1,43% del PIB en I+D, lo que supuso un total de 17.249 millones de euros (Instituto Nacional de Estadística, s.f.).

El compromiso de España con la tecnología no se centra exclusivamente en el capital aportado a investigación, también se busca que la tecnología llegue al hogar de los españoles, España es el cuarto país de la Unión Europea que más ha aumentado el acceso a internet desde 2011, con un incremento de 32 puntos porcentuales desde este año, se ha conseguido que el 89% de la población tenga una conexión mayor de 100 Mbps (red.es, s.f.).

A estos datos hay que aportarle los dispositivos tecnológicos de los cuales los españoles disponen en sus hogares, en la ilustración 4.1 podemos observar que la mayoría de los españoles disponen de dispositivos móviles, con un 99.5% seguido de ordenadores con un 81,4% (ONTSI, s.f., pág. 14).

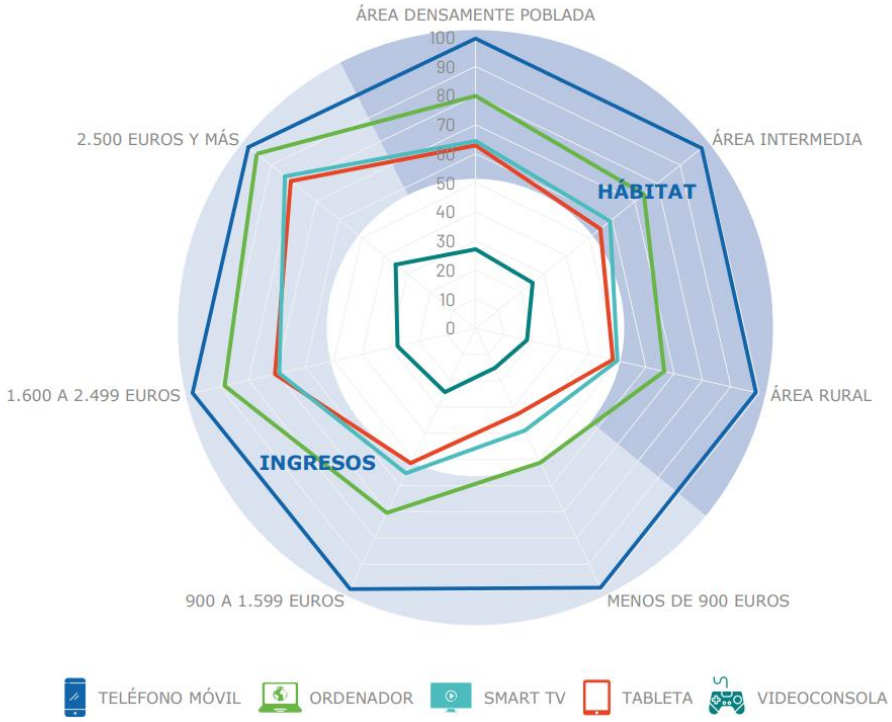

Ilustración 4.1 Acceso a tecnología en hogares por tipo de hábitat y nivel de ingresos

4.2.1.5 Matriz análisis PEST

Perfil Pest	Factor	Impacto		
		Positivo	Neutro	Negativo
Político	Ayudas a la financiación	X		
	Impuestos			X
	Inestabilidad política			X
Económico	Tasa de inflación			X
	Nivel de desarrollo		X	
	Deuda			X
Sociocultural	Crecimiento de la población			X
	Nivel educativo	X		
	Estilo de vida	X		
Tecnológico	Gasto en investigación	X		
	Acceso a la tecnología	X		

Tabla 4.1 Análisis PEST

4.2.2 Microentorno

En el estudio del microentorno trataremos sobre los potenciales clientes, la competencia que existe dentro de nuestro ámbito del desarrollo de videojuegos y con la situación actual de España en el sector específico del desarrollo de videojuegos.

Como ya vimos en el apartado 1.4, existe una gran cantidad de personas que juegan a videojuegos normalmente, por lo que tenemos una gran cantidad de público potencial, pero para hacer un estudio más exhaustivo vamos a segmentar estos posibles clientes.

Como se muestra en la ilustración 4.2 podemos ver que los géneros que más éxito tienen entre las personas que juegan a videojuegos son el género de acción, los videojuegos de rol, y los juegos de deporte que están bastante distanciado de los demás en lo que a número de unidades se refiere (Asociación Española de Videojuegos, 2022, pág. 20).

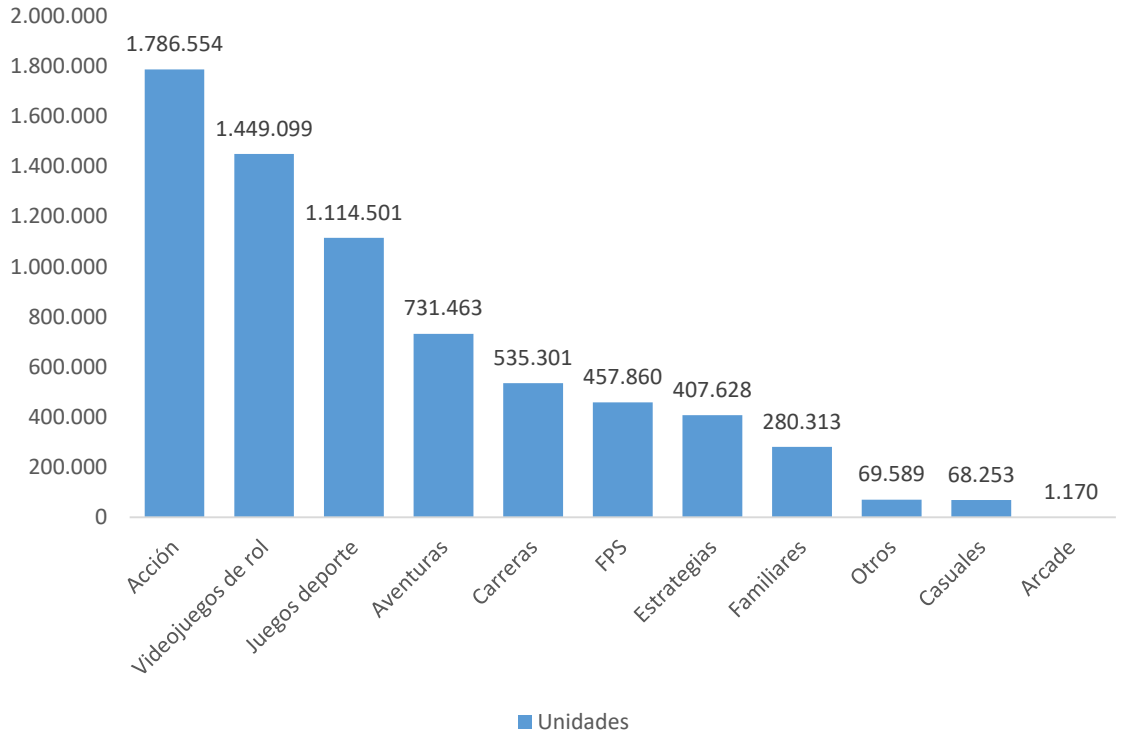

Ilustración 4.2 Géneros más populares en unidades

Otro aspecto a tener en cuenta es saber que dispositivos utilizan mas los españoles a la hora de jugar a videojuegos. Aunque nuestro videojuego está pensado para que sea multiplataforma, gracias a la portabilidad que nos ofrece Unity para generar nuestro juego en distintas plataformas, es importante saber a que plataforma juegan para dedicarle una atención mas especial a dicha plataforma para llegar a más público. En la ilustración 4.3 podemos observar las plataformas usadas por los españoles durante el año 2022 (Asociación Española de Videojuegos, 2022, pág. 26).

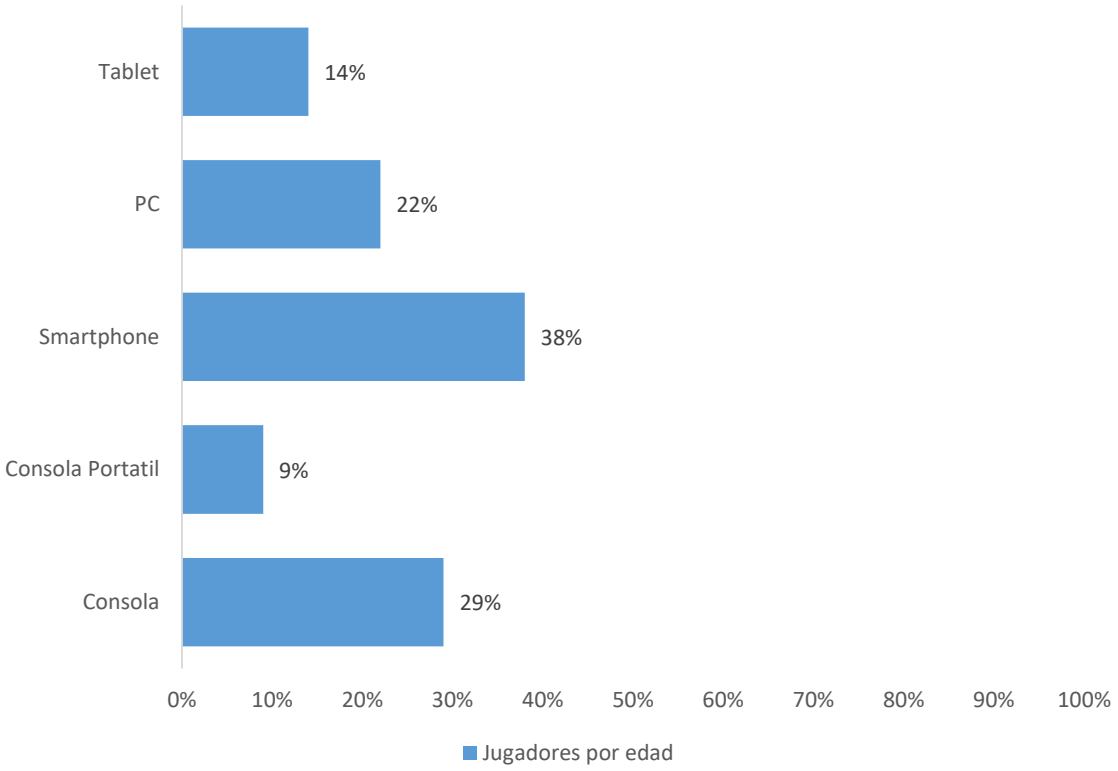

Ilustración 4.3 Plataformas utilizadas por los españoles durante el año 2022

En cuanto a la competencia no existen empresas de desarrollo de videojuegos en la provincia de Jaén (Devuego, s.f.). Sin embargo, en la comunidad autónoma de Andalucía si existe bastante competencia, ya que esta conforma alrededor del 16% de estudios de todo el país (Desarrollo Español de Videojuegos, 2022, pág. 25).

Si hablamos de las ayudas recibidas en el sector durante los últimos años, podemos ver en la ilustración 4.4 que cada vez se dan menos ayudas al desarrollo de videojuegos.

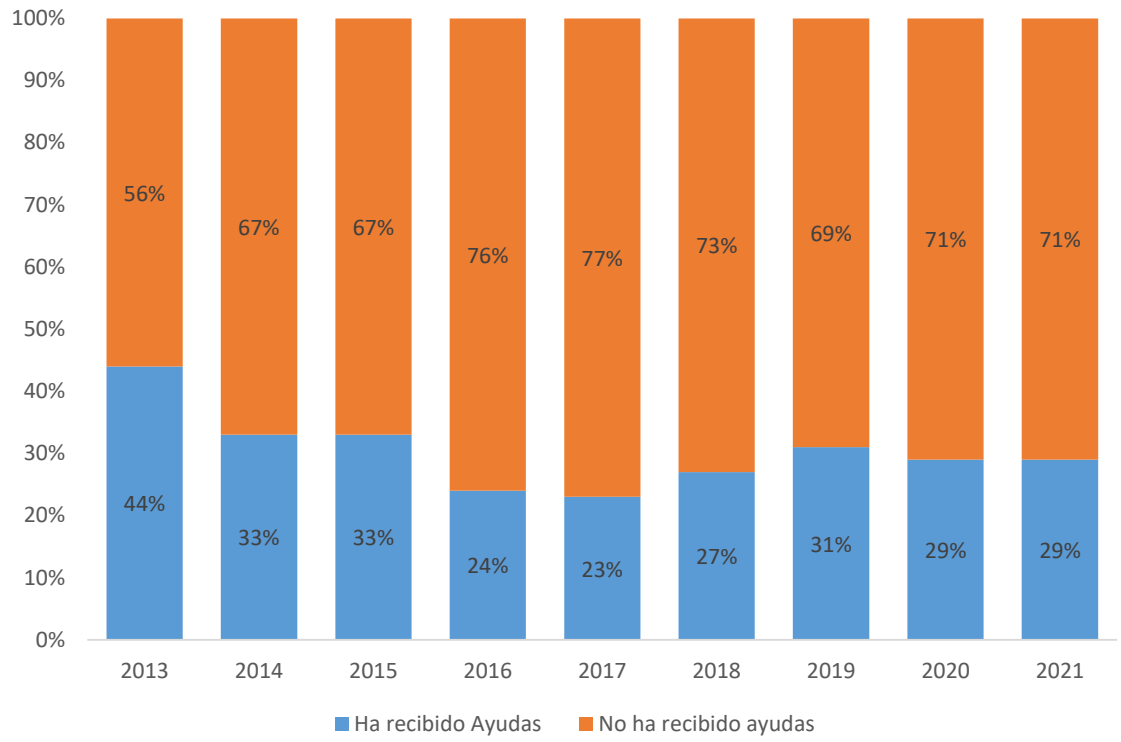

Ilustración 4.4 Porcentaje de empresas que han recibido ayudas públicas (2013-2021)

Sin embargo, la subvención media en 2021 ha ascendido a 55.777 euros, el préstamo medio a 130.000 euros y la deducción fiscal media a 138.833, lo cual nos indica que el desarrollo de videojuegos es algo que importa al gobierno español y está comprometido con el mismo. En lo que respecta al lugar de procedencia de las ayudas a los estudios de videojuegos se ha observado que la principal fuente han sido las comunidades autónomas con un 35%, seguidas por las deducciones por actividades de investigación y desarrollo e innovación tecnológica (30%) y los programas nacionales I+D+i (22%) (Desarrollo Español de Videojuegos, 2022, pág. 46). Se espera que entre el 50% y el 42% opte por reclamar ayudas públicas para su próximo proyecto, aunque también el 56% de los mismos optara por la autofinanciación, lo cual

quiere decir que obtendrá el dinero para el siguiente proyecto de sus propios ingresos (Desarrollo Español de Videojuegos, 2022, pág. 43).

4.2.3 Conclusión

Una herramienta para la evaluación de una idea emprendedora es la evaluación de la matriz DAFO (Debilidades, amenazas, fortalezas y oportunidades), que comprende los factores del microentorno y macroentorno.

	Interno	Externo
Negativo	Debilidades	Amenazas
	Nueva creación de empresa, sin reconocimiento alguno. Promotores de la empresa con poca experiencia.	Gran déficit publico Situación inestable en el país
Positivo	Fortalezas	Oportunidades
	Calidad del producto. Equipo cualificado para el desarrollo de videojuegos.	Aumento en la inversión del sector. Gran ventana en el mercado.

Tabla 4.2 Análisis DAFO

4.3 Plan de mercadotecnia

En este apartado se establecerá un plan de marketing, en el cual se describirá como se llevará a cabo la comercialización del producto, en nuestro caso el proyecto realizado para el TFG.

4.3.1 Política de producto

En cuanto a la política del producto hablaremos sobre la obtención del mismo, y que posibles cambios puede sufrir el mismo.

El producto que vamos a vender al público, es el software que hemos desarrollado durante este proyecto, el cual a lo largo de esta memoria ha sido detallado. Cada cliente obtendrá su aplicación en el dispositivo que la haya y podrá utilizar la aplicación sin restricciones algunas, este archivo ya estará preparado para su instalación y no necesitara ningún despliegue inicialmente.

A continuación lo que debemos hacer es colocar nuestro producto en la parte correspondiente al ciclo de vida y asignar una estrategia a realizar sobre este.

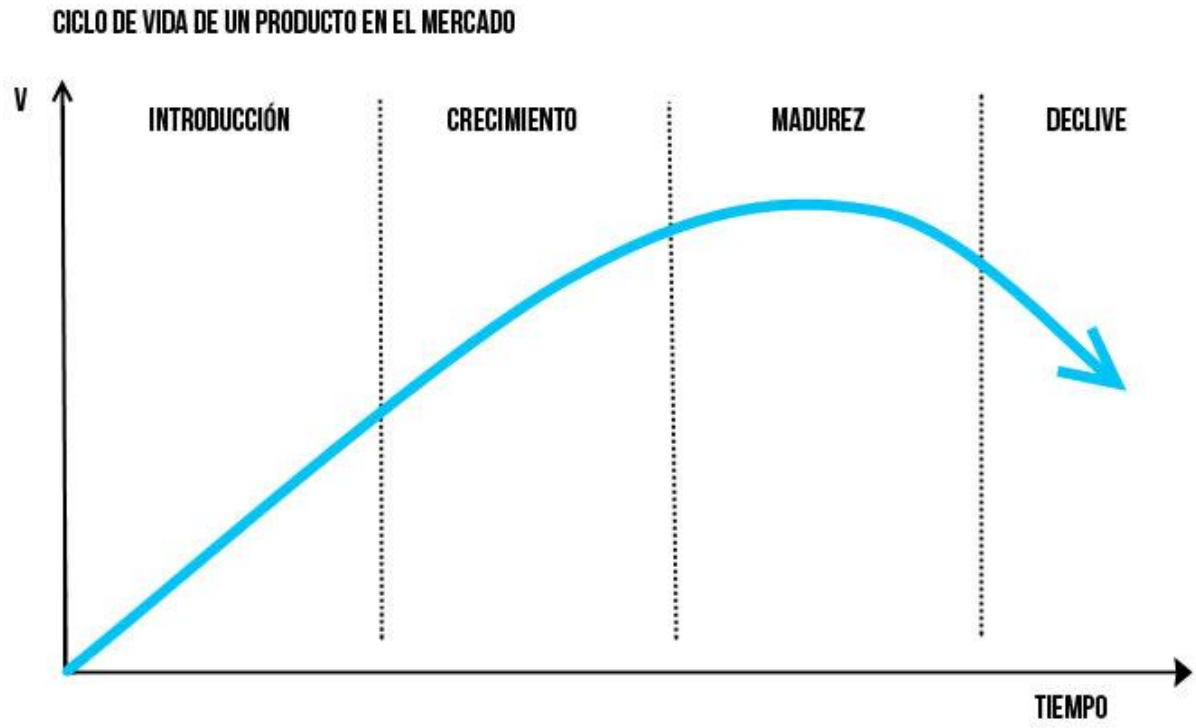

Ilustración 4.5 Ciclo de vida de un producto

4.3.1.1 Introducción

Es el momento en el que acabamos de publicar el juego, por lo tanto lo que haremos será lanzar el juego a un precio bastante bajo y realizar una campaña publicitaria importante, para que todo el mundo que esté interesado en los juegos de cartas este esperando el juego.

4.3.1.2 Crecimiento

Después la etapa de introducción llega el crecimiento, para esta etapa debemos de introducir nuevas funcionalidades al videojuego, como podría ser permitir el uso de diferentes texturas para las cartas, introducir nuevos juegos de cartas todo ello con el fin de seguir manteniendo al público que comenzó con el juego y seguir captando la atención de nuevos clientes.

4.3.1.3 Madurez

En la fase de madurez se produce un estancamiento, por ello nuestra estrategia va a centrarse principalmente en mantener a los clientes que están haciendo uso de nuestro juego aunque también debemos tener en cuenta que debemos seguir intentando captar la atención de nuevos clientes, por ello sería una buena idea agregar nuevos modos de juego, para que los cambios sean más notables que con las nuevas funcionalidades de la etapa anterior. Un buen ejemplo de ello sería agregar un modo que permita competir en torneos sobre un juego de cartas específico, para los jugadores que ganen el torneo se le otorgaran insignias únicas que podrán ser mostradas al resto de jugadores. Esta nueva modalidad dará un aire al juego que parezca que están utilizando otro juego desde el principio.

4.3.1.4 Declive

Esta es la última etapa de nuestro videojuego y se distingue por la baja de ventas de nuestro videojuego. En este punto la estrategia va a ser modificar la forma de adquisición del juego para que siga llegando a más público. Por lo tanto lo que hará será sacar una versión que los clientes pueden obtener gratuitamente pero se conseguirá dinero a través de anuncios y transacciones dentro de la propia aplicación. Además le daremos la opción al usuario de comprar el juego por el precio de salida el cual le permita eliminar los anuncios del juego para poder seguir obteniendo ganancias.

4.3.2 Política de precios

Para establecer una política de precios tendremos que estimar los gastos que realiza un jugador por cada partida y estimar el número de partidas que jugara mensualmente de forma online, que es la que puede generar gastos adicionales. Para realizar este análisis interno haremos una sobrestimación de los datos, lo cual quiere decir que siempre asumiremos las peores condiciones sin tener en cuenta los elementos gratuitos que proporciona Firebase.

Suponemos que la media diaria es de 15 partidas por jugador, lo que significa que de media al mes son 450 partidas.

Una partida está compuesta por una media de 44 peticiones, teniendo en cuenta las peticiones al matchamkings y las peticiones a la realización de movimientos.

Por lo tanto el número de mensajes mensuales enviados por cada jugador será de 19.800 mensajes.

Una vez calculados el número de mensajes, que es uno de los motivos por el cual se realizaran pagos a Firebase, se debe de observar cuanto almacenamiento ocupa una partida dentro de la base de datos, lo cual supone 2 Kb.

Con todas estas estimaciones podemos decir que un jugador con estas estimaciones estaría generando un gasto de alrededor de 1 céntimo.

Por lo que teniendo en cuenta este estudio se estima que el juego podría tener un valor 1.5€, lo cual asegura ganar dinero con cualquier jugador que utilice la aplicación independientemente de su uso.

4.3.3 Política de comunicación

Hablando sobre la política de comunicación tenemos que destacar los distintos ámbitos en los que vamos a publicitar el juego para que llegue al mayor número de personas posibles.

Una buena forma de publicitarse es a través de las redes sociales, por esta razón creamos un perfil de X, Instagram, Facebook y TikTok, con estas cuatro redes sociales se cubre la gran mayoría del rango de clientes potenciales.

Las redes sociales no son suficientes para captar la atención, ya que cada vez más la gente tiende a desconfiar de todo lo que ve en redes sociales, por lo que una buena forma de dar validez al nuevo producto y mostrarlo al mundo es a través de promociones con personajes famosos, como actores, cómicos, influencers, etc.

4.3.4 Política de distribución

La distribución del videojuego se realizara a través de plataformas destinadas a este propósito, inicialmente el videojuego solo estará disponible para ordenador, por lo que las plataformas que usaremos serán Steam (Steam, s.f.), que es la plataforma por excelencia para juegos de ordenador, además de Itch.io (Itch.io, s.f.), que está más orientada a juegos individuales, pero cada vez se está dando más a conocer.

En un futuro gracias a la escalabilidad del proyecto se podría crear las versiones para móvil, tabletas…etc. En consecuencia a ello subirlo a las propias aplicaciones de los sistemas operativos de los mismos.

Es por ello que se subiría la aplicación en Google Play Store (Google Play Store, s.f.) para los dispositivos con sistema operativo android y para los dispositivos con sistema operativo IOS, en la App Store (Apple, s.f.).

4.4 Forma jurídica de la empresa

En este apartado trataremos sobre la forma jurídica de la empresa, la cual establece los derechos y responsabilidades de los propietarios y gestores de la empresa, además de dictar como tributara la empresa y como recaudara el dinero (Indeed, s.f.).

En España existen multitud de estructuras jurídicas de empresa, por lo que para decidir qué tipo será nuestra empresa haremos uso de la herramienta que proporciona el ministerio de Economía, Industria, Comercio y Turismo (Ministerio de Economia, Industria, Comercio y Turismo, s.f.). Con esta herramienta filtraremos los tipos de empresas disponibles dependiendo de las condiciones iniciales para crearlas. Como podemos ver en la ilustración 4.6 gracias a esta plataforma se ha restringido bastante la elección de nuestra forma jurídica y hemos obtenidos las siguientes formas jurídicas.

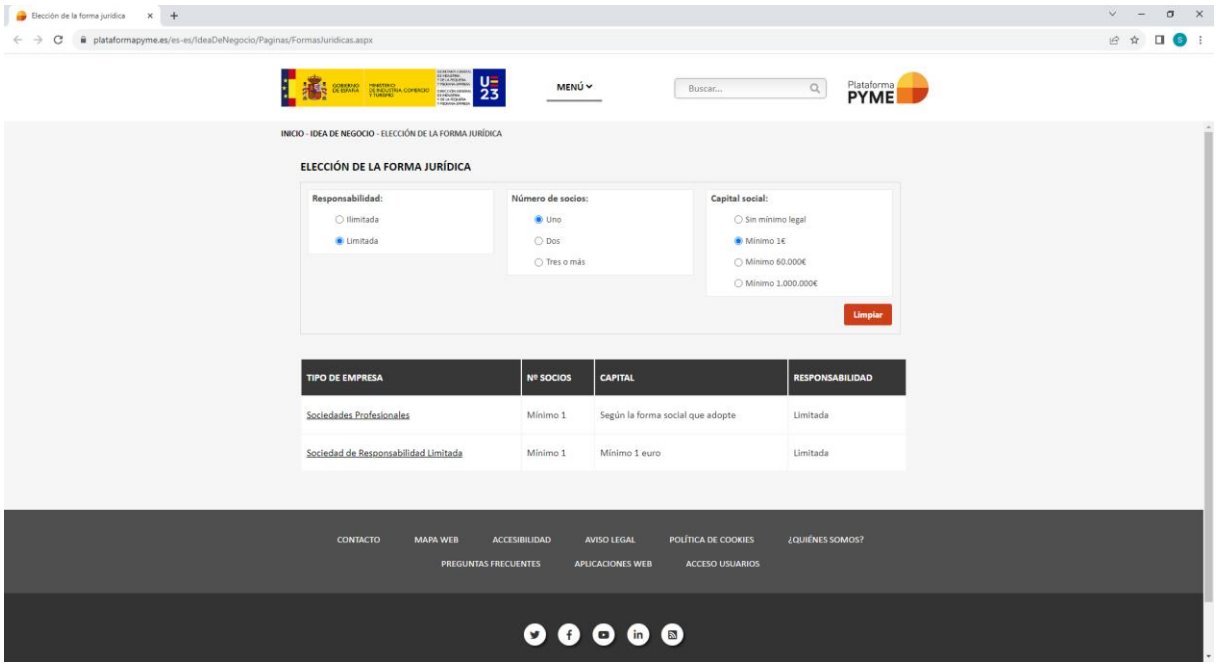

Ilustración 4.6 Captura de la herramienta para la clasificación de una nueva empresa

En nuestro caso la empresa tendrá como forma jurídica la Sociedad de Responsabilidad Limitada (S.R.L), o en su forma abreviada Sociedad Limitada (S.L). La elección es univoca ya que, para ser una Sociedad Profesional, un requerimiento es que se ejerzan actividades reguladas por colegios profesionales, el cual no es nuestro caso.

Este tipo de empresas la responsabilidad de los socios por las deudas sociales está limitada a las aportaciones a capital, lo cual quiere decir que no ponen en riesgo su capital personal, como persona física frente a la empresa. Este capital es mínimo de 1€ lo cual es bastante asequible, aunque cabe destacar que si el capital aportado es inferior a 3000 euros se añaden dos reglas las cuales nos obligan a destinar el 20% de los beneficios a reserva legal hasta llegar la cifra de los 3000 euros y que en caso de liquidación de la empresa los socios deberán aportar la diferencia entre el capital inicial y los 3000 euros. Entre las muchas ventajas de este tipo de empresa podemos discernir como más importantes que no existe un número máximo de socios para conformarla ni un número mínimo de socios trabajadores. Sin embargo, en esta empresa los socios son siempre identificables y no puede cotizar en bolsa si así se deseara en un futuro (Plataforma Pyme, s.f.).

La normativa a aplicar para este tipo de empresa se ve reflejada en los siguientes documentos (Plataforma Pyme, s.f.).

- Ley 18/2022, de 28 de septiembre, de creación y crecimiento de empresas (BOE, 2022).
- Real Decreto Legislativo 1/2010 por el que se aprueba el texto refundido de la Ley de Sociedades de Capital (BOE, 2010).
- Real Decreto 421/2015, de 29 de mayo, por el que se regulan los modelos de estatutos-tipo y de escritura pública estandarizados de las sociedades de responsabilidad limitada, se aprueba modelo de estatutos-tipo, se regula la Agenda Electrónica Notarial y la Bolsa de denominaciones sociales con reserva (BOE, 2015).
- Orden JUS/1840/2015, por la que se aprueba el modelo de escritura pública en formato estandarizado y campos codificados de las sociedades de responsabilidad limitada, así como la relación de actividades que pueden formar parte del objeto social (BOE, 2015).

- Real Decreto-ley 13/2010, de actuaciones en el ámbito fiscal, laboral y liberalizadoras para fomentar la inversión y la creación de empleo (BOE, 2010).
- Ley 14/2013 de apoyo a los emprendedores y su internacionalización (BOE, 2013).

En cuanto a la creación de la empresa, esta se puede constituir tanto de forma presencial como de forma telemática. Para hacerlo de forma telemática, el gobierno pone a nuestra disposición la plataforma CIRCE (Gobierno de España, s.f.). De esta forma se evitan desplazamientos y se produce un ahorro sustancial en tiempo y costes (Plataforma Pyme, s.f.). Para crear una empresa por internet, nosotros como emprendedor por nosotros mismos o acudiendo a un Punto de Atención al Emprendedor (PAE, s.f.), deberemos cumplimentar el Documento Único Electrónico.

Antes de cumplimentar el Documento Único Electrónico, se deberán realizar las siguientes actuaciones (Plataforma Pyme, s.f.):

- Reserva de la Denominación Social: Consiste en solicitar al registro mercantil central (RMC, s.f.) una certificación sobre que el nombre de la empresa no se encuentre asignado ni reservado por otra empresa.
- Aportación del Capital Social: Consiste en indicar el importe de capital, participación de cada socio y, si se trata de aportaciones no dinerarias, una breve descripción del bien aportado y su valor

5 CONCLUSIONES Y TRABAJOS FUTUROS

Este proyecto ha servido para complementar muchas de las capacidades del alumno, así como sobre todo el trabajo autónomo ya que la gestión del desarrollo ha sido propia y no se sigue ningún “guion” como ocurre en las prácticas de las diferentes asignaturas. Además, ha ayudado a aumentar la visión del alumno respecto a la importancia de ciertas partes del desarrollo de un videojuego como por ejemplo puede ser toda la parte relacionada con el diseño de personajes, animaciones y fondos; en resumen, todo lo relacionado con el arte de un videojuego.

El uso y compresión de los patrones arquitectónicos han dado una madurez a la forma de realizar código de ahora en adelante para el alumno, siendo capaz de realizar aplicaciones mucho más consistentes y entendibles, ese ha sido el aspecto más importante de dicho trabajo.

Se han afrontado bastantes problemas durante el desarrollo, como a la hora de mandar información a través de los sockets se utilizaban clases templates, las cuales no podían ser interpretadas simplemente con los atributos, configuraciones con la plataforma de Firebase, bastantes problemas con la concurrencia en Unity, ya que ciertas operaciones necesitaban que se ejecutaran en el hilo principal, pero el gestor de hilos no lo hacía en dicho lugar. Pero a pesar de tantos problemas a los que se ha enfrentado en dicho proyecto, se ha conseguido un prototipo de cartas funcional.

Algunas tareas han quedado pendientes y aunque se algunas se hayan planificado no ha sido posible que llevarse a cabo.

- Implementar la operación de cambiar la carta del triunfo.
- Implementar las caidas de los jugadores en una partida online
- Agregar una interfaz para configurar los aspectos de sonido del juego
- Implementar la funcionalidad de Lobbys, para que se pueda jugar partidas entre amigos
- Implementar niveles de dificultad para jugar contra la CPU

6 APÉNDICES

6.1 Instalación y configuración del sistema

Se necesitara instalar el código fuente dentro de dos programas, ya que como se ha visto a lo largo de la memoria se ha utilizado Unity, para la plataforma que llevara a cabo el juego y Netbeans para la plataforma de agentes.

Con el fin de evitar problemas de compatibilidad se recomienda que se utilice las mismas versiones para ambos programas, las cuales son

- Unity 2021.3.17 LTS
- NetBeans IDE 17

6.1.1 Puesta en marcha del proyecto en Unity

Lo primero que se deberá de hacer será crear un proyecto en 3D, para ello, lo que se debe hacer es abrir UnityHub, pulsar sobre el botón de *NewProject*, esto nos llevara a una nueva interfaz y en ella haremos clic sobre 3D, después se deberá pulsar el botón de *Create Project.*

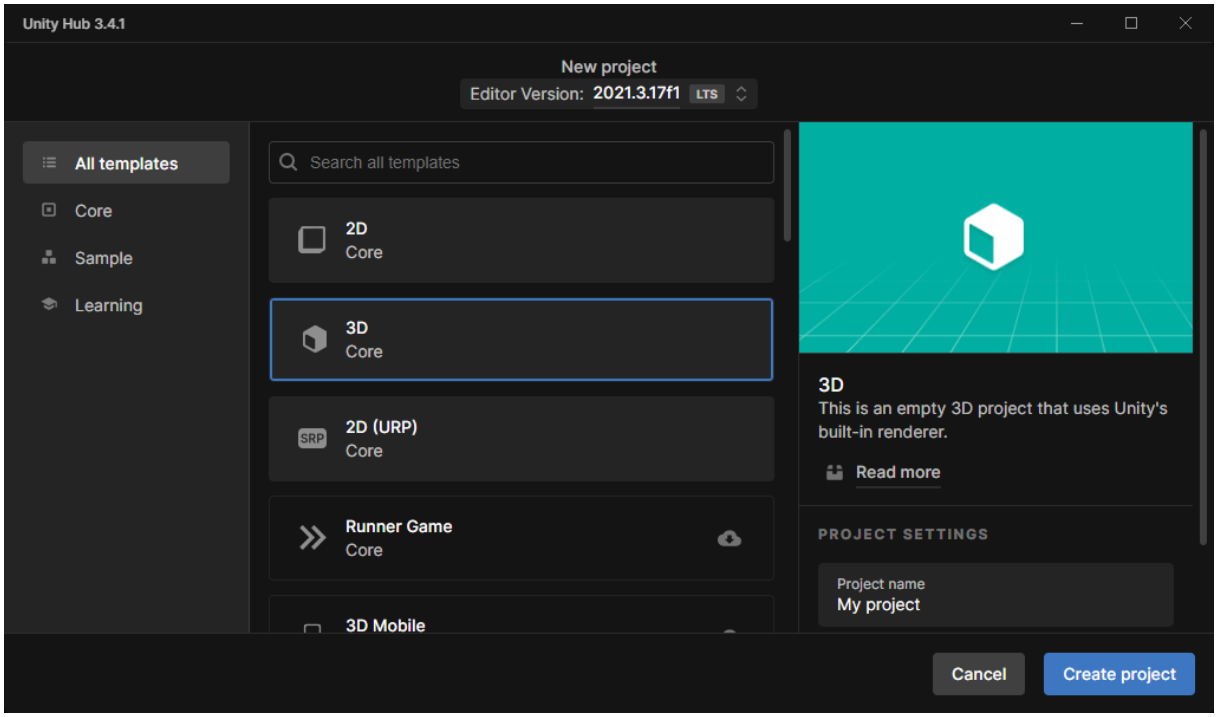

Ilustración 6.1 Captura creación de un proyecto en Unity

Una vez se ha creado el proyecto se deberán instalar los paquetes necesarios para el correcto funcionamiento del mismo. Para instalar dichos paquetes se debe seguir la ruta Window > Package Manager, en el menú superior del proyecto de Unity. Seguidamente en la nueva ventana deberemos seleccionar en el menú deplegable superior Package: Unity Registry e instalar los paquetes que se muestran en la siguiente imagen.

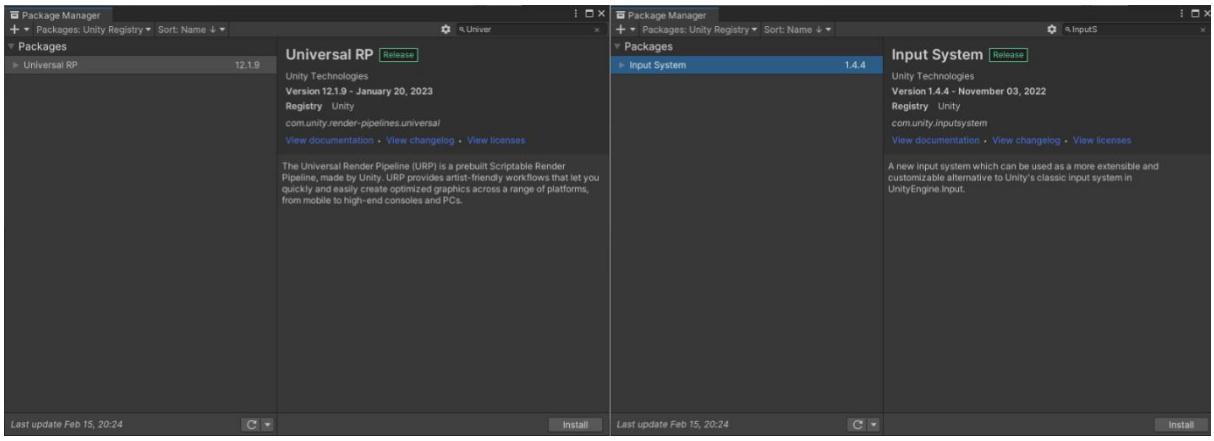

Ilustración 6.2 Captura paquetes necesarios para la instalación

Una vez que se han instalados los paquetes se deberá comenzar con la configuración del proyecto. Primero se configurara el nuevo sistema de renderizado, que instalamos con el paquete de Universal RP. Lo primero que se debe hacer será crear un Asset el cual contendrá los elementos necesarios para implementar dicha renderizacion. Para crearlo se hace clic derecho dentro de la carpeta Asset, y después se seguirá la ruta de *Create > Rendering > URP Asset (Whit uniersal render)*. Se deberá obtener un resultado como el que se muestra en la siguiente imagen.

Ilustración 6.3 Captura de la creación del Asset para URP

Seguidamente se deberá asignar el Asset al proyecto para que lo utilice de ahora en adelante, para ello se dirigirá al menú superior y seguirá *Edit* > *Project Settings*. En la nueva interfaz que se abrirá se debe buscar el apartado *Graphics*. En la parte superior hay una entrada con el nombre de *Scriptabe Render Pipeline Settings*, en dicho elemento se asignara el Asset que se creó anteriormente, el resultado final debe ser el que se muestra en la siguiente imagen.

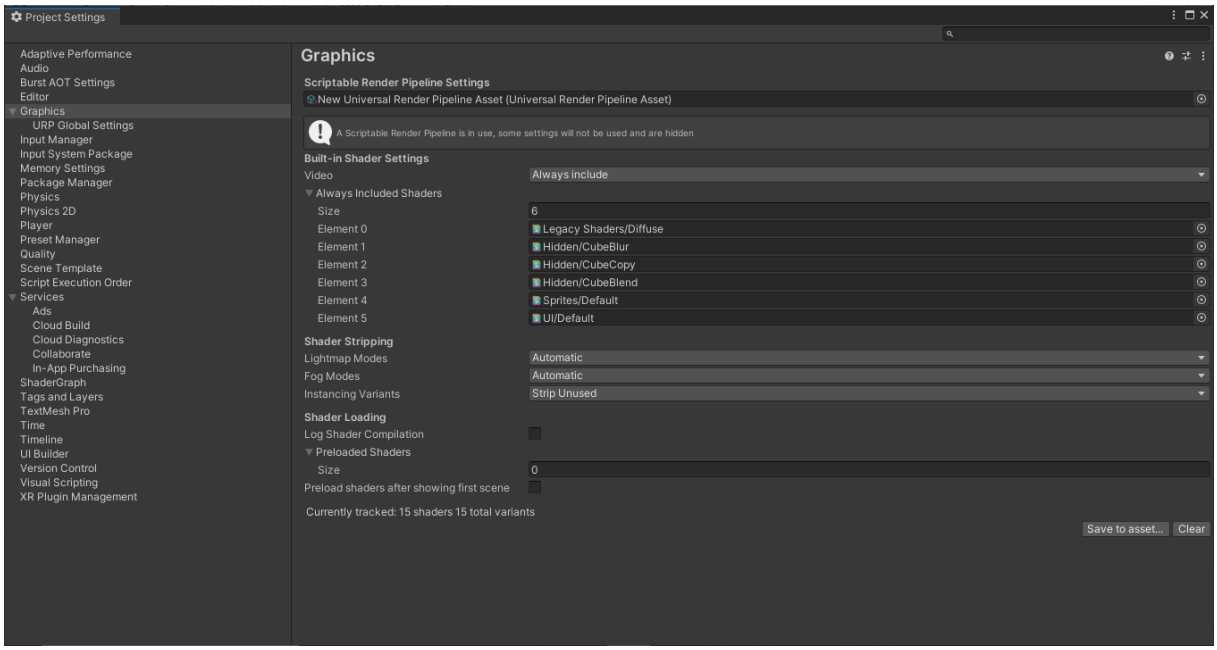

Ilustración 6.4 Captura de la asignación del URP al proyecto

Con este paso realizado ya estaría configurado el primer paquete instalado, que sería el de Universal RP. El siguiente paso será configurar el paquete del Input System, aunque para este no se necesitara realizar cambios en ninguna configuración.

A continuación se deberá copiar la carpeta InputSystem, que se encuentra dentro de la carpta de Unity > Source, en la entrega del proyecto, dentro de la carpeta Asset de Unity, se puede hacer arrastrándola directamente. Una vez hecho esto abrimos dicha carpeta y haremos clic sobre el elemento que está contenido en ella, se abrirá el inspector en la parte derecha del proyecto, en el cual se deberá marcar la casilla Generate C# class (Ilustración 6.5), una vez marcado le damos a aplicar.

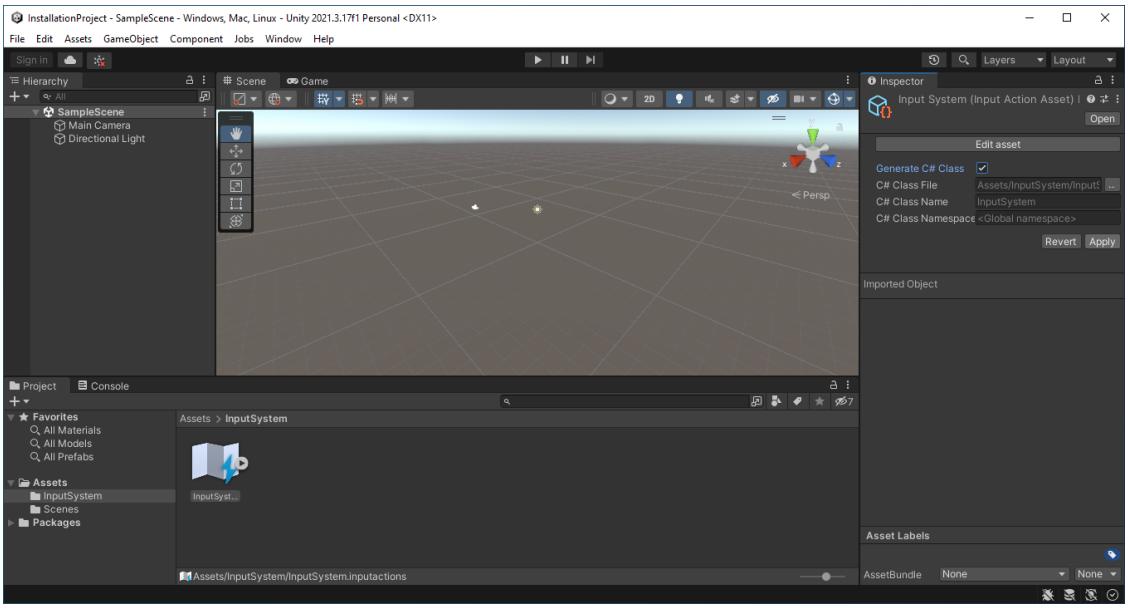

Ilustración 6.5 Captura configuración Input System

El siguiente paso será copiar la carpeta Code, situada dentro de Unity > Source en la entrega del proyecto y la carpeta Resources, la cual se encuentra en la misma ruta, primero se debe copiar la carpeta Code para evitar problemas a la hora de importar. Después de copiar ambas se deberá abrir las escenas para que se importe correctamente. En la apertura de alguna escena se abrirá la ventana emergente que se muestra en la siguiente ilustración. En la cual se importara los dos TMP disponibles.

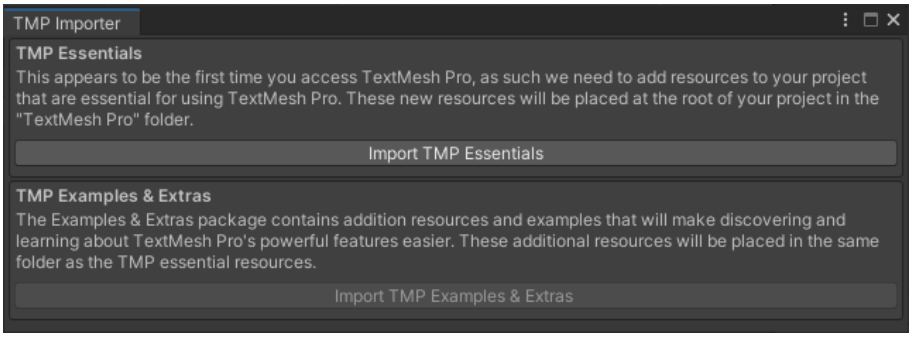

Ilustración 6.6 Captura de importación de TextMeshPro

Para el correcto funcionamiento de la plataforma de agentes se deberá copiar el contenido la carpeta Agents incluida en la entrega Unity > Source, dentro de la carpeta Asset del proyecto, solo puede copiarse aquí ya que esta codificado para que coja el directorio de trabajo como carpeta raíz, y esta es la carpeta Assset.

El último paso será la agregación de las escenas dentro de la propia configuración del proyecto para que el propio motor de videojuegos las reconozca y funcione correctamente el código implementado. Para ello se deberá seguir la siguiente ruta en el menú superior de Unity, File > Build Settings. En la nueva ventana se deberán agregar las escenas y que queden en el orden que se muestra a continuación, para que los índices de las mismas coincidan y no se produzcan transiciones no deseadas.

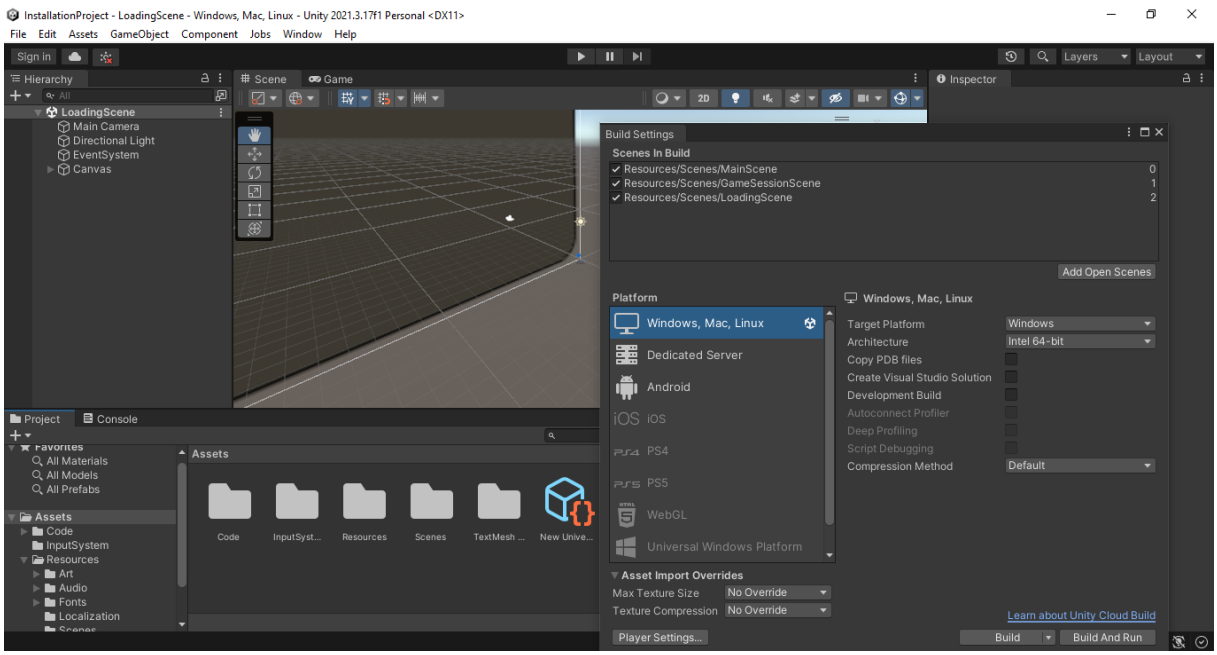

Ilustración 6.7 Captura configuración de escenas

Para agregar una escena se deberá tener dicha escena abierta y una vez se haya realizado esta acción se deberá hacer clic con el botón de *Add Open Scenes,* este paso se debe repetir con las 3 escenas del proyecto.

6.1.2 Puesta en marcha del proyecto en Netbeans

El primer paso a realizar será la creación de un proyecto Maven, dentro del IDE de NetBeans (Ilustración 6.8)

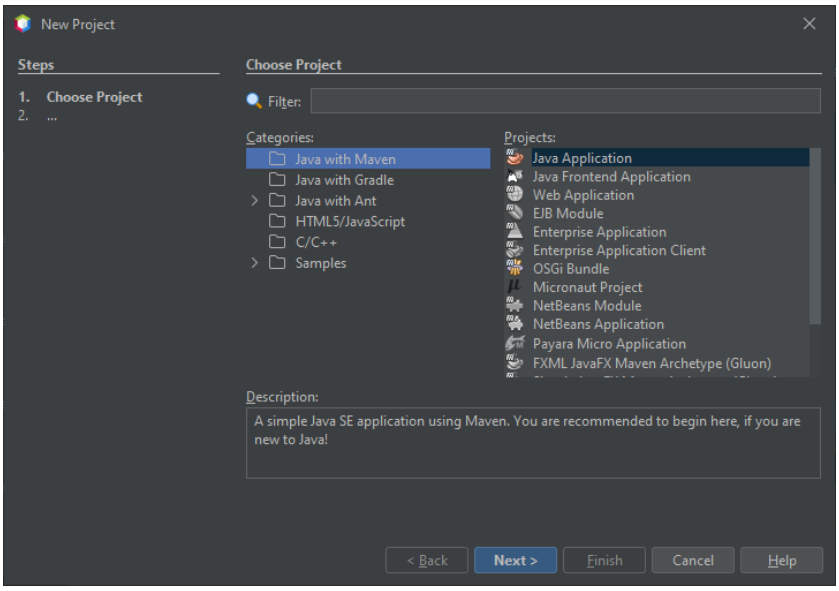

Ilustración 6.8 Captura creación proyecto Maven

En la configuración del proyecto, a la hora de creación del mismo, en el apartado Group ID, se deberá asignar el valor de *plataformajuegoscartas,* para la posterior copia del código dentro del proyecto, tal como se puede observar en la siguiente imagen.

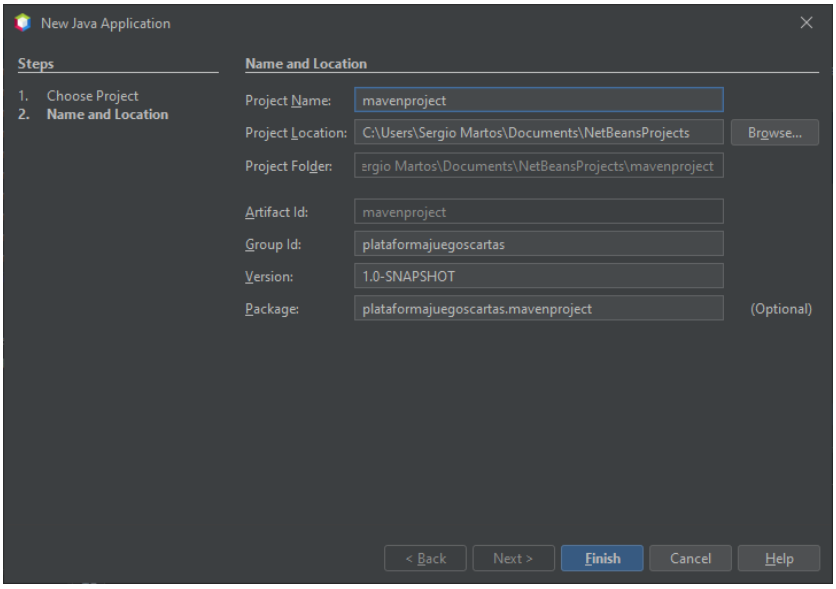

Ilustración 6.9 Captura configuración inicial del proyecto maven

Próximamente se deberá realizar la configuración del proyecto, para acceder a dicha configuración se deberá hacer clic derecho sobre el proyecto creado y seguidamente accedemos a la sección de *Properties*. En dicha sección accederemos al apartado *Configurations*, con el fin de agregar unas directivas para la descarga de la biblioteca. Dentro de dicha sección se hará clic sobre el botón *Add* y se escribirán las directivas que se pueden apreciar en la ilustración 6.10

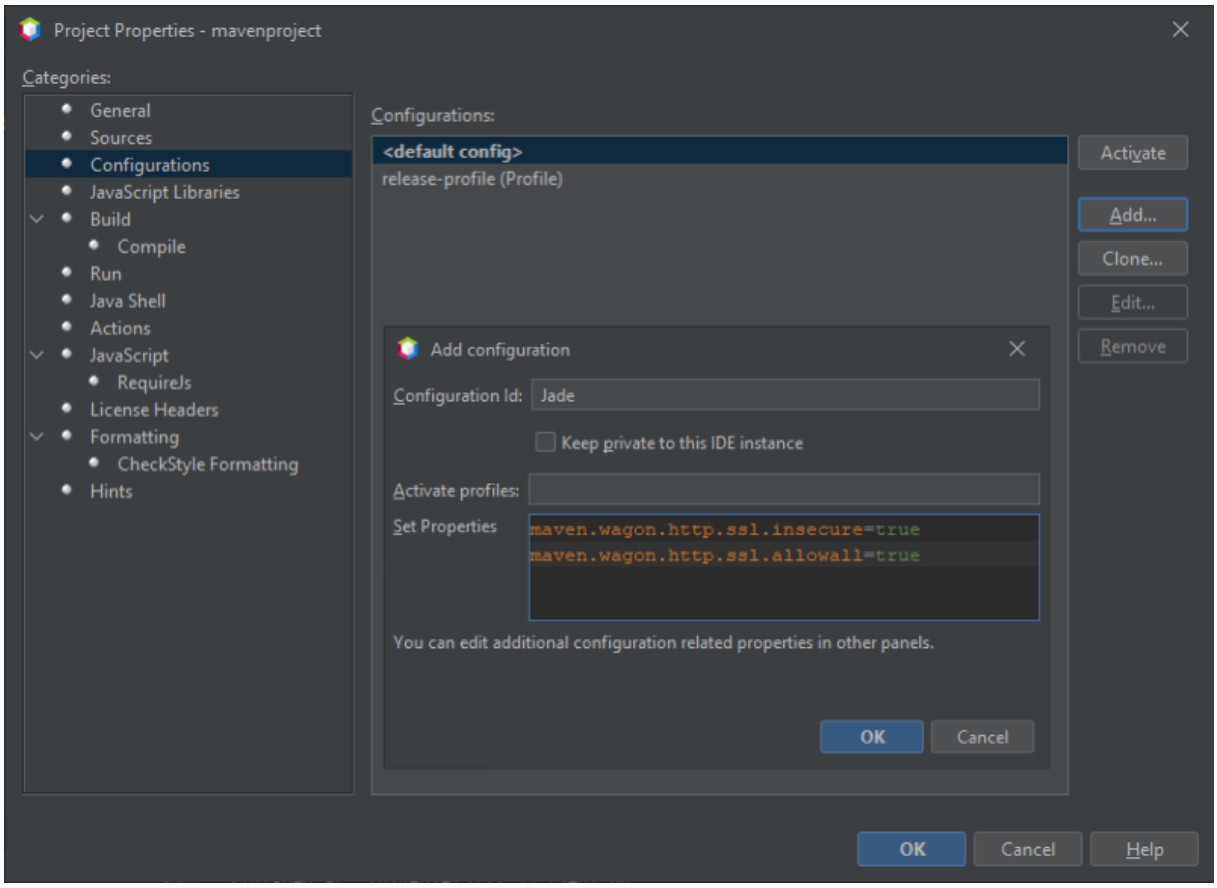

Ilustración 6.10 Captura directivas configuración

Una vez agregada la nueva configuración asegurarse de activarla, para ello se hará un clic sobre la misma y seguidamente se hará clic sobre el botón de *Activate.*

El siguiente paso será configurar la clase inicial del proyecto con la cual comenzara su ejecución, para ello nuevamente deberemos acceder a las propiedades del proyecto, como se explicó anteriormente y dirigirse a la sección de *Run,* aquí se asignara como *Main Class*, la clase jade.Boot, para iniciar la plataforma de agentes y además se agregara como argumentos de creación la siguiente línea *-agents organizador:plataformajuegoscartas.agentes.AgentOrganizer*, para que comience la plataforma con un agente organizador.

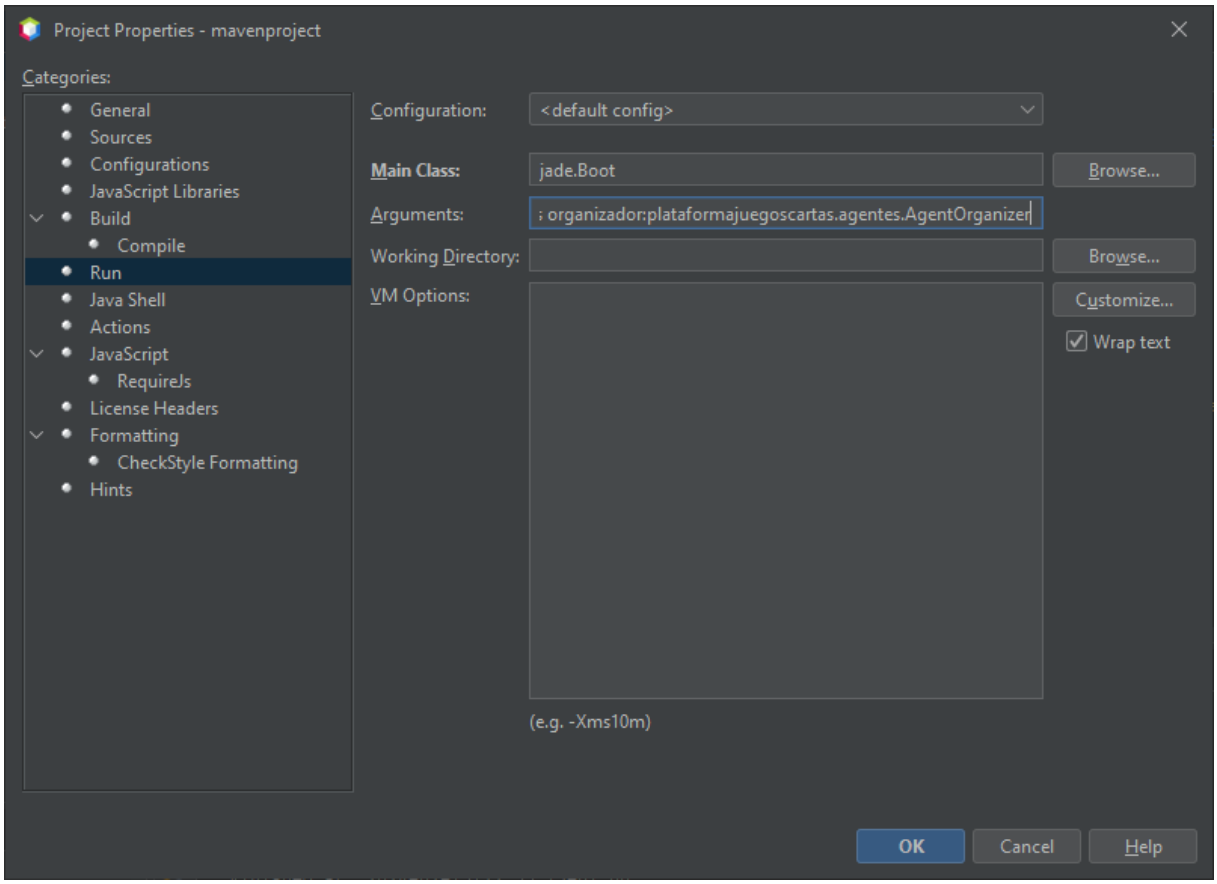

Ilustración 6.11 Captura configuración de la sección Run

Una vez hecho esto se cerrara el proyecto de Netbeans y deberá dirigirse a la carpeta donde se haya almacenado el nuevo proyecto generado, en ella se deberá copiar el contenido de la carpeta Java > Code, entregado en el proyecto, dentro de la carpeta *src* > *main* > *java* > *plataformajuegoscartas.* Además también deberá remplazar el archivo pom.xml por el de la entrega del proyecto.

En dicho archivo hay que tener en cuenta dichos aspectos que se deben remplazar dependiendo de cada proyecto generado

Si se ha utilizado una versión de JDK distinta a la versión 11, se deberá remplazar la versión utilizada en las propiedades que se muestran en la siguiente ilustración.

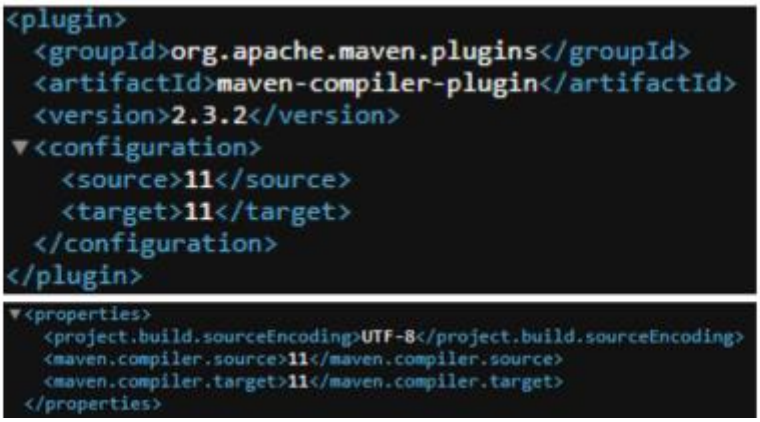

Ilustración 6.12 Captura modificación JDK en el fichero pom.xml

Otro aspecto a tener en cuenta es que si no se modifica la línea de *artifactId*, en la ilustración 6.13, dentro del fichero pom.xml se modificara el nombre del proyecto asignado.

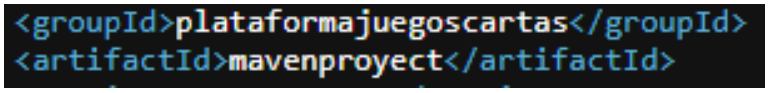

Ilustración 6.13 Captura de la información del proyecto pom.xml

6.2 Manual de usuario

Se mostrara un breve manual de usuario con el fin de mostrar la utilización del software desarrollado.

Al comenzar la ejecución de la aplicación pedirá al usuario que se registre o inicie sesión de alguna forma, este registro es obligatorio, no se podrá jugar al juego hasta que no se realice el registro completo. Si existe alguna duda sobre el registro se puede mirar el video del funcionamiento de la plataforma.

Seguidamente se deberá seleccionar un juego y una modalidad, el juego por defecto será el Cinquillo si no se ha seleccionado ninguno. Nuevamente en el video se muestra los pasos a seguir para unirse a un nuevo juego.

Después de la selección de juego se mostrara la pantalla de carga, es muy importante no cerrar la aplicación cuando se muestra esta pantalla, ya que puede dar lugar a caídas y desincronización de los datos en las partidas en línea, si por cualquier motivo el programa no responde y no queda más remedio que cerrar la aplicación en este punto, se deberá de borrar el contenido de la carpeta Files, que se generara cuando se ejecute el proyecto. A continuación se volverá a abrir de nuevo la aplicación, se debe tener en cuenta que el usuario anterior quedo registrado el matchmaking, por lo que habrá que generar tantos usuarios como sea necesario para que la partida se complete se genere la propia partida y ya vuelva a funcionar la plataforma con normalidad.

Por ultimo destacar que la propia aplicación tiene funcionalidades de cambio de nombre de usuario y cambio de idioma, para ver cómo se implementan mirar nuevamente en el video de las funcionalidades. Además también se le dará la opción al usuario de volver a jugar el tipo de partida jugada o de volver al menú principal, estas funcionalidades son inmediatas ya que solamente se tendrá que pulsar en los botones que se indica.

7 DEFINICIONES Y ABREVIATURAS

- Uniform Resource Locator: Es conocido como la direccion que identifica un determinado sitio dentro de internet.
- Free To Play: Son videojuegos por los cuales no es necesario pagar dinero para jugar al mismo.
- Central Proccessing Unit: Es la unidad central de procesamiento de un ordenador, la parte mas importante, por ello es conocido el termino jugar contra la CPU para referirse a jugar contra el ordenador.
- Etcétera: Es una terminologia del latin, la cual signfica y demas, o y otros, normalmente sirve para indicar que la lista podria continuar con mas ejemplos.
- Application Programming Interface: Es una pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades

8 BIBLIOGRAFÍA

AEGON. (s.f.). Recuperado el 31 de Octubre de 2023, de AEGON | IV Estudio Salud y Vida: https://fr.zone-secure.net/149562/1404875/#page=133

Agencia Tributaria. (s.f.). Recuperado el 31 de Octubre de 2023, de Agencia Tributaria | Tipo Impositivo: https://sede.agenciatributaria.gob.es/Sede/impuesto-sobre-sociedades/que-base-imponible-se-determina-sociedades/tipo-impositivo.html

Agencia Tributaria. (s.f.). Recuperado el 31 de Octubre de 2023, de Agencia Tributaria | Tipos impositvos en el IVA 2023: https://sede.agenciatributaria.gob.es/static_files/Sede/Tema/IVA/Novedades/Empresas/2023/Tipos_IVA%20_2023.pdf

Amazon. (s.f.). *Amazon Sage Maker*. Recuperado el 31 de Octubre de 2023, de https://aws.amazon.com/es/blogs/aws-spanish/formacion-de-un-agente-de-aprendizaje-de-refuerzo-con-unity-y-amazon-sagemaker-rl/#:~:text=mlagents%2Denvs%20%E2%80%94%20Paquete%20que%20proporciona,el%20motor%20de%20juego%20Unity

Amazon. (s.f.). *GameLit Docuemtacion*. Recuperado el 31 de Octubre de 2023, de https://aws.amazon.com/es/solutions/guidance/gamelift-testing-on-aws/

Amazon. (s.f.). *GameLit Precios*. Recuperado el 31 de Octubre de 2023, de https://aws.amazon.com/es/gamelift/pricing/

Apache. (s.f.). *Netbeans*. Recuperado el 8 de Febrero de 2024, de https://netbeans.apache.org/front/main/

Apple. (s.f.). *App Store*. Recuperado el 31 de Octubre de 2024, de https://www.apple.com/es/app-store/

AppWarp. (s.f.). *AppWarp*. Recuperado el 31 de Octubre de 2023, de https://appwarp.shephertz.com/

AppWarp. (s.f.). *AppWarp Precios*. Recuperado el 31 de Octubre de 2023, de https://appwarp.shephertz.com/pricing/

Artemisa. (s.f.). *Artemisa*. Recuperado el 31 de Octubre de 2023, de http://artemisa.unicauca.edu.co/~cardila/LAB_IS_1__03a__Estimaciones__Cocomo.pdf

Asociación Española de Videojuegos. (2022). *La Industria del videojuego en España*. Recuperado el 31 de Octubre de 2023, de AEVI: http://www.aevi.org.es/web/wp-content/uploads/2023/05/Anuario-AEVI-2022.pdf

Bandai. (s.f.). Recuperado el 31 de Octubre de 2023, de Dragon Ball Fighter Z: https://es.bandainamcoent.eu/dragon-ball/dragon-ball-fighterz

Bellifemine, F. (2007). *Developing multi-agent systems whit JADE.* Wiley-Blackwell.

Blender. (s.f.). *Blender*. Recuperado el 31 de Octubre de 2023, de https://www.blender.org/

Blender. (s.f.). *Blender | Bevel*. Recuperado el 10 de Febrero de 2024, de https://docs.blender.org/manual/es/latest/modeling/modifiers/generate/bevel.html

BOE. (3 de Julio de 2010). Recuperado el 31 de Octubre de 2023, de Real Decreto Legislativo 1/2010, de 2 de julio, por el que se aprueba el texto refundido de la Ley de Sociedades de Capital.: https://www.boe.es/buscar/act.php?id=BOE-A-2010-10544

BOE. (3 de Diciembre de 2010). Recuperado el 31 de Octubre de 2023, de Real Decreto-ley 13/2010, de 3 de diciembre, de actuaciones en el ámbito fiscal, laboral y liberalizadoras para fomentar la inversión y la creación de empleo: https://www.boe.es/buscar/act.php?id=BOE-A-2010-18651

BOE. (28 de Septiembre de 2013). Recuperado el 31 de Octubre de 2023, de Ley 14/2013, de 27 de septiembre, de apoyo a los emprendedores y su internacionalización.: https://www.boe.es/buscar/act.php?id=BOE-A-2013-10074

BOE. (2014). Recuperado el 31 de Octubre de 2023, de Reglamento (UE) nº 651/2014: https://www.boe.es/doue/2014/187/L00001-00078.pdf

BOE. (12 de Septiembre de 2015). Recuperado el 31 de Octubre de 2023, de Orden JUS/1840/2015: https://www.boe.es/buscar/act.php?id=BOE-A-2015-9803

BOE. (13 de Junio de 2015). Recuperado el 31 de Octubre de 2023, de Real Decreto 421/2015, de 29 de mayo: https://www.boe.es/buscar/act.php?id=BOE-A-2015-6520

BOE. (29 de Septiembre de 2022). Recuperado el 31 de Octubre de 2023, de Ley 18/2022, de 28 de septiembre, de creación y crecimiento de empresas.: https://www.boe.es/eli/es/l/2022/09/28/18/con

BOE. (13 de Julio de 2023). *MINISTERIO DE TRABAJO Y ECONOMÍA SOCIAL*. Recuperado el 31 de Octubre de 2023, de XVIII CONVENIO COLECTIVO ESTATAL DE EMPRESAS DE CONSULTORÍA, TECNOLOGÍAS DE LA INFORMACIÓN Y ESTUDIOS DE MERCADO Y DE LA OPINIÓN PÚBLICA: https://www.boe.es/boe/dias/2023/07/26/pdfs/BOE-A-2023-17238.pdf

C#Corner. (s.f.). *C#Corner | Singleton*. Recuperado el 8 de Febrero de 2024, de https://www.c-sharpcorner.com/UploadFile/snorrebaard/the-quest-for-the-generic-singleton-in-C-Sharp/

Cuphead. (s.f.). Recuperado el 31 de Octubre de 2023, de Cuphead: https://cupheadgame.com/

Datosmacro. (s.f.). Recuperado el 31 de Octubre de 2023, de España-Natalidad: https://datosmacro.expansion.com/demografia/natalidad/espana

Datosmacro. (s.f.). Recuperado el 31 de Octubre de 2023, de España-Tasa de alfabetización: https://datosmacro.expansion.com/demografia/natalidad/espana

Desarrollo Español de Videojuegos. (2022). *Ministerio de Asuntos Economicos y Transformación Digital*. Recuperado el 31 de Octubre de 2023, de Libro blanco

de desarrollo de videojuegos 2022: https://dev.org.es/images/stories/docs/libro%20blanco%20del%20desarrollo%20espanol%20de%20videojuegos%202022.pdf

Developer Comunity Unreal Engine. (s.f.). *How much is learning curve for Unreal engine?* Recuperado el 31 de Octubre de 2023, de Dev Community: https://forums.unrealengine.com/t/how-much-is-learning-curve-for-unreal-engine/122656

Devuego. (s.f.). Recuperado el 31 de Octubre de 2023, de Devuego | Base de datos del videojuego español: https://www.devuego.es/bd/mapa-estudios/

Direccion General de Economia y Estadistica. (s.f.). Recuperado el 31 de Octubre de 2023, de BANCO DE ESPAÑA: https://www.bde.es/f/webbe/GAP/Secciones/SalaPrensa/IntervencionesPublicas/DirectoresGenerales/economia/IIPP-2023-09-19-gavilan.pdf

EA. (s.f.). Recuperado el 31 de Octubre de 2023, de EA | Star Wars: Jedi Fallen Orded: https://www.ea.com/es-es/games/starwars/jedi/jedi-fallen-order

FIPA. (s.f.). *FIPA*. Recuperado el 5 de Febrero de 2024, de http://fipa.org/

Firebase. (s.f.). *Firebase | API REST*. Recuperado el 12 de Febrero de 2024, de https://firebase.google.com/docs/reference/rest/database?hl=es

Firebase. (s.f.). *Firebase | Auth API*. Recuperado el 10 de Febrero de 2024, de https://firebase.google.com/docs/reference/rest/auth?hl=es

Firebase. (s.f.). *Firebase | Cloud Functions*. Recuperado el 12 de Febrero de 2024, de https://firebase.google.com/docs/functions/get-started?gen=1st&hl=es

Firebase. (s.f.). *Firebase | Instalacion y configuración de la API*. Recuperado el 12 de Febrero de 2024, de https://firebase.google.com/docs/database/rest/start?hl=es#next_steps

Fish-Networking. (31 de Octubre de 2023). *Fish-Networking*. Obtenido de https://fish-networking.gitbook.io/docs/manual/

Fortnite. (s.f.). Recuperado el 31 de Octubre de 2023, de Fortnite: https://www.fortnite.com/?lang=en-US

Fournier. (s.f.). *Fournier*. Recuperado el 10 de Febrero de 2024, de https://www.nhfournier.es/blog/que-relacion-tienen-los-palos-de-la-baraja-francesa-y-la-baraja-espanola/

GameDevelopment. (s.f.). *GameDevelopment | MonoBehaviour Singleton*. Recuperado el 8 de Febrero de 2024, de https://gamedev.stackexchange.com/questions/116009/in-unity-how-do-i-correctly-implement-the-singleton-pattern

Gimp. (31 de Octubre de 2023). *Gimp*. Obtenido de http://www.gimp.org.es/

GitHub. (s.f.). *EvtSource*. Recuperado el 2 de Febrero de 2024, de https://github.com/3ventic/EvtSource

Gobierno de España. (s.f.). Recuperado el 31 de Octubre de 2023, de CIRCE: https://paeelectronico.es/es-es/Servicios/Paginas/Creacion-y-cese-de-empresas.aspx

Google. (s.f.). *Firebase | API Rest*. Recuperado el 5 de Febrero de 2024, de https://firebase.google.com/docs/reference/rest/database?hl=es

Google. (s.f.). *Firebase | Precios*. Recuperado el 31 de Octubre de 2023, de https://firebase.google.com/pricing?hl=es-419

Google. (s.f.). *Firebase | Real Time Database*. Recuperado el 5 de Febrero de 2024, de https://firebase.google.com/docs/database?hl=es

Google. (s.f.). *Google Cloud*. Recuperado el 5 de Febrero de 2024, de https://cloud.google.com/?hl=es

Google Play Store. (s.f.). Recuperado el 31 de Octubre de 2023, de Google Play Store: https://play.google.com/store/games

Hollow Knight. (s.f.). Recuperado el 31 de Octubre de 2023, de Hollow Knight: https://www.hollowknight.com/

IBM. (s.f.). *IBM | JSON*. Recuperado el 12 de Febrero de 2024, de https://www.ibm.com/docs/es/baw/20.x?topic=formats-javascript-object-notation-json-format

Indeed. (s.f.). Recuperado el 31 de Octubre de 2023, de Qué es la forma jurídica de una empresa: https://mx.indeed.com/orientacion-profesional/desarrollo-profesional/forma-juridica-empresa

Instituto Nacional de Estadística. (s.f.). Recuperado el 31 de Octubre de 2023, de Estadisticas sobre actividades I+D: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176754&menu=ultiDatos&idp=1254735576669#:~:text=%C3%9Altima%20Nota%20de%20prensa&text=El%20gasto%20en%20I%2BD,363%2C66%20euros%20por%20habitante

Instituto Nacional de Estadística. (16 de Noviembre de 2022). Recuperado el 31 de Octubre de 2023, de Indicadores demograficos: https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177003&menu=ultiDatos&idp=1254735573002

Instituto Nacional de la Seguridad Social. (s.f.). Recuperado el 31 de Octubre de 2023, de Pensiones contributivas del Sistema de la Seguridad Social en vigor a 1 de agosto de 2023: https://www.seg-social.es/wps/wcm/connect/wss/845d205d-c125-4ee7-b46c-8afeacc6d140/CA202308.pdf?MOD=AJPERES&CONVERT_TO=linktext&CACHEID=ROOTWORKSPACE.Z18_2G50H38209D640QTQ57OVB2000-845d205d-c125-4ee7-b46c-8afeacc6d140-oFaCRXI

Instituto Nacional Estadistica. (s.f.). Recuperado el 31 de Octubre de 2023, de Estadistica Continua de Poblacion: https://www.ine.es/daco/daco42/ecp/ecp0223.pdf

IPYME. (s.f.). Recuperado el 31 de Octubre de 2023, de Portal IPYME: https://ipyme.org/es-es/AyudasIncentivos/Paginas/buscador-de-ayudas.aspx

IPYME. (s.f.). *Preguntas Frecuentes*. Recuperado el 31 de Octubre de 2023, de IPYME: https://ipyme.org/es-es/queespyme/Paginas/preguntas-frecuentes-pyme.aspx

Itch.io. (s.f.). *Itch.io*. Recuperado el 2 de Febrero de 2024, de https://itch.io/

JADE. (s.f.). *JADE*. Recuperado el 31 de Octubre de 2023, de https://jade-project.gitlab.io/

La Moncloa. (s.f.). *El gasto en pensiones contributivas se sitúa en el 11,7% del PIB*. Recuperado el 31 de Octubre de 2023, de Inclusion, Seguridad Social y Migraciones: https://www.lamoncloa.gob.es/serviciosdeprensa/notasprensa/inclusion/paginas/2023/240223-gasto-pensiones-contributivas.aspx#:~:text=24%2F02%2F2023.,Inclusi%C3%B3n%2C%20Seguridad%20Social%20y%20Migraciones%5D

Ludoteka. (s.f.). Recuperado el 31 de Octubre de 2023, de Ludoteka: https://www.ludoteka.com/

Medium. (20 de Abril de 2018). Recuperado el 31 de Octubre de 2023, de Requerimientos funcionales y no funcionales, ejemplos y tips: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a

Microsoft. (s.f.). *Microsoft | Actions*. Recuperado el 31 de Enero de 2024, de https://learn.microsoft.com/en-us/dotnet/api/system.action-1?view=net-8.0

Microsoft. (s.f.). *Microsoft | Precios Azure*. Recuperado el 31 de Octubre de 2023, de https://azure.microsoft.com/en-us/pricing/

Microsoft. (s.f.). *Microsoft | Process*. Recuperado el 31 de Enero de 2024, de https://learn.microsoft.com/es-es/dotnet/api/system.diagnostics.process?view=net-8.0

Microsoft. (s.f.). *Microsoft | System.Net.Http*. Recuperado el 31 de Enero de 2024, de https://learn.microsoft.com/es-es/dotnet/api/system.net.http?view=net-8.0

Microsoft. (s.f.). *Microsoft -HLSL*. Recuperado el 15 de Enero de 2024, de https://learn.microsoft.com/en-us/windows/win32/direct3dhlsl/dx-graphics-hlsl

Microsoft. (s.f.). *Playfab*. Recuperado el 31 de Octubre de 2023, de https://azure.microsoft.com/es-es/products/playfab

Microsoft. (s.f.). *Playfab | Precios*. Recuperado el 31 de Octubre de 2023, de https://playfab.com/pricing/

Microsoft. (s.f.). *Playfab | Servidores Multijador*. Recuperado el 31 de Octubre de 2023, de https://learn.microsoft.com/en-us/gaming/playfab/features/multiplayer/servers/

Microsoft. (s.f.). *Visual Studio Code*. Recuperado el 8 de Febrero de 2024, de https://code.visualstudio.com/

Minijuegos. (s.f.). Recuperado el 31 de Octubre de 2023, de Minijuegos: https://www.minijuegos.com/

Ministerio de Economia, Industria, Comercio y Turismo. (s.f.). Recuperado el 31 de Octubre de 2023, de Plataforma Pyme: https://mx.indeed.com/orientacion-profesional/desarrollo-profesional/forma-juridica-empresa

Ministerio de Educacion y Formacion Profesional. (18 de Julio de 2023). Recuperado el 31 de Octubre de 2023, de Nota Avance 22-23: https://www.educacionyfp.gob.es/dam/jcr:7c2f9b4c-28f0-4b75-af56-ee1a55be0373/nota-avance-2022-23.pdf

Ministerio de Hacienda y Funcion Publica. (2020). Recuperado el 31 de Octubre de 2023, de Manual práctico de Renta 2020 | Por coeficientes de amortización lineal: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/

Ministerio de Sanidad. (s.f.). Recuperado el 31 de Octubre de 2023, de Estadística de Gasto Sanitario Público: https://www.sanidad.gob.es/estadEstudios/estadisticas/docs/EGSP2008/egspPrincipalesResultados.pdf

Mirror. (s.f.). *Mirror*. Recuperado el 1 de Febrero de 2024, de https://mirror-networking.com/

Nintendo. (s.f.). Recuperado el 31 de Octubre de 2023, de Pokemon Diamante Brillante y Perla Reluciente: https://www.nintendo.es/Juegos/Juegos-de-Nintendo-Switch/Pokemon-Diamante-Brillante-1930566.html

ONTSI. (s.f.). Recuperado el 31 de Octubre de 2023, de Tecnologia + Sociedad en España: https://www.ontsi.es/sites/ontsi/files/2022-01/tecnologiasociedadespa%C3%B1a2021_0.pdf

Oracle. (s.f.). *Oracle*. Recuperado el 5 de Febrero de 2024, de https://www.oracle.com/es/database/nosql/what-is-nosql/

PAE. (s.f.). Recuperado el 31 de Octubre de 2023, de Buscador de Puntos de Atención al Emprendimiento: https://paeelectronico.es/es-es/Servicios/Paginas/BuscadorPAE.aspx

PAE. (2012). Recuperado el 31 de Octubre de 2023, de MAGERIT v.3 : Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información: https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Magerit.html

Photon. (s.f.). *Photon*. Recuperado el 31 de Octubre de 2023, de https://www.photonengine.com/

Photon. (s.f.). *Photon Documentacion*. Recuperado el 31 de Octubre de 2023, de https://doc.photonengine.com/fusion/current/fusion-intro

Photon. (s.f.). *Photon Precios*. Recuperado el 31 de Octubre de 2023, de https://www.photonengine.com/pun/pricing

Plataforma Pyme. (s.f.). Recuperado el 31 de Octubre de 2023, de Sociedad de Responsabilidad Limitada | Plataforma Pyme: https://plataformapyme.es/es-

es/IdeaDeNegocio/Paginas/FormasJuridicas-Descripcion.aspx?cod=SRL&nombre=Sociedad%20de%20Responsabilidad%20Limitada&idioma=es-ES

Pressman, R. (2010). *Ingeniería del Software.* McGraw-Hill.

red.es. (s.f.). Recuperado el 31 de Octubre de 2023, de España es el cuarto país de la UE que más ha aumentado el acceso a internet desde 2011: https://www.red.es/es/actualidad/noticias/espana-es-el-cuarto-pais-de-la-ue-que-mas-ha-aumentado-el-acceso-internet-desde

RMC. (s.f.). Recuperado el 31 de Octubre de 2023, de Registro Mercantil Central: https://www.rmc.es/privado/CertificacionesDenominaciones.aspx

Sage. (s.f.). *Activos*. Recuperado el 10 de Febrero de 2024, de https://www.sage.com/es-es/blog/diccionario-empresarial/activo/#:~:text=Los%20activos%2C%20desde%20el%20punto,rendimientos%20econ%C3%B3micos%20en%20el%20futuro.

Softzone. (s.f.). *¿Puedo crear juegos 2D en Unreal Engine?* Recuperado el 31 de Octubre de 2023, de https://www.softzone.es/noticias/programas/puedo-crear-juegos-2d-unreal-engine/

Steam. (s.f.). Recuperado el 31 de Octubre de 2023, de Steam | Among Us: https://store.steampowered.com/app/945360/Among_Us/

Steam. (s.f.). *Steam*. Recuperado el 4 de Febrero de 2024, de https://store.steampowered.com/?l=spanish

Supercell. (s.f.). Recuperado el 31 de Octubre de 2023, de Supercell BrawlStarts: https://supercell.com/en/games/brawlstars/

Superprof. (s.f.). Recuperado el 31 de Octubre de 2023, de Superprof | Eleccion Motor de Videojuegos: https://www.superprof.com.ar/blog/eleccion-motores-juego-creacion/

UMLet. (s.f.). *UMLet*. Recuperado el 31 de Octubre de 2023, de https://www.umlet.com/

Unity. (s.f.). Recuperado el 15 de Enero de 2024, de Unity: https://unity.com/es

Unity. (s.f.). Recuperado el 31 de Octubre de 2023, de Unity | Pricing: https://unity.com/compare-plans

Unity. (s.f.). Recuperado el 15 de Enero de 2023, de Unity | Disscusions: https://discussions.unity.com/

Unity. (s.f.). Recuperado el 15 de Enero de 2024, de Unity | Forum: https://forum.unity.com/

Unity. (s.f.). Recuperado el 15 de Enero de 2024, de Unity | Manual: https://docs.unity3d.com/es/530/Manual/UnityManual.html

Unity. (s.f.). *Unity - Best Practise Render Pipeline*. Recuperado el 1 de Febrero de 2024, de https://docs.unity3d.com/es/2021.1/Manual/BestPracticeLightingPipelines.html

Unity. (s.f.). *Unity - Texture Array*. Recuperado el 31 de Enero de 2024, de https://docs.unity3d.com/es/2019.4/Manual/class-Texture2DArray.html

Unity. (s.f.). *Unity | AudioClip*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/es/2021.1/Manual/class-AudioClip.html

Unity. (s.f.). *Unity | AudioSource*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/es/2021.1/Manual/class-AudioSource.html

Unity. (s.f.). *Unity | BoxCollider*. Recuperado el 10 de 2 de 2024, de https://docs.unity3d.com/es/2018.4/Manual/class-BoxCollider.html

Unity. (s.f.). *Unity | Canvas*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/es/2021.1/Manual/UICanvas.html

Unity. (s.f.). *Unity | Colliders*. Recuperado el 11 de Febrero de 2024, de https://docs.unity3d.com/ScriptReference/Collider.html

Unity. (s.f.). *Unity | Coroutines*. Recuperado el 11 de Febrero de 2024, de https://docs.unity3d.com/Manual/Coroutines.html

Unity. (s.f.). *Unity | Gaming Services Precio*. Recuperado el 31 de Octubre de 2023, de https://unity.com/solutions/gaming-services/pricing

Unity. (s.f.). *Unity | Input System*. Recuperado el 9 de Febrero de 2024, de https://docs.unity3d.com/Packages/com.unity.inputsystem@1.7/manual/index.html

Unity. (s.f.). *Unity | MLAgents*. Recuperado el 15 de Enero de 2024, de https://docs.unity3d.com/Packages/com.unity.ml-agents@1.0/api/Unity.MLAgents.Agent.html

Unity. (s.f.). *Unity | Relay*. Recuperado el 31 de Octubre de 2023, de https://docs.unity.com/ugs/manual/relay/manual/introduction

Unity. (s.f.). *Unity | SceneManager*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/ScriptReference/SceneManagement.SceneManager.html

Unity. (s.f.). *Unity | ScriptableObjects*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/Manual/class-ScriptableObject.html

Unity. (s.f.). *Unity | Slider*. Recuperado el 12 de Febrero de 2024, de https://docs.unity3d.com/2018.2/Documentation/ScriptReference/UI.Slider.html

Unity. (s.f.). *Unity | Transport*. Recuperado el 5 de Febrero de 2024, de https://docs-multiplayer.unity3d.com/transport/current/about/

Unity. (s.f.). *Unity | UI*. Recuperado el 10 de Febrero de 2024, de https://docs.unity3d.com/Packages/com.unity.ugui@1.0/manual/index.html

Unity. (s.f.). *Unity NetCode*. Recuperado el 31 de Octubre de 2023, de https://docs-multiplayer.unity3d.com/netcode/current/about/

Unity. (s.f.). *Unity Shaders*. Recuperado el 3 de Febrero de 2024, de https://docs.unity3d.com/Manual/Shaders.html

Unity. (s.f.). *UnityTextureArrayShader*. Recuperado el 4 de Febrero de 2024, de https://docs.unity3d.com/es/2019.4/Manual/SL-TextureArrays.html

Unity. (s.f.). *Universal Render Pipeline*. Recuperado el 3 de Febrero de 2024, de https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@13.1/manual/InstallURPIntoAProject.html

UNO. (s.f.). Recuperado el 31 de Octubre de 2023, de UNO Mobile: https://www.letsplayuno.com/

Unreal Engine. (s.f.). Recuperado el 31 de Octubre de 2023, de Unreal Engine | Documentation: https://docs.unrealengine.com/5.3/en-US/

Unreal Engine. (s.f.). Recuperado el 31 de Octubre de 2023, de Unreal Engine | Features: https://www.unrealengine.com/en-US/features

Unreal Engine. (s.f.). Recuperado el 31 de Octubre de 2023, de Unreal Engine | Forum: https://forums.unrealengine.com/

Unreal Engine. (s.f.). Recuperado el 31 de Octubre de 2023, de Unreal Engine | Pricing: https://www.unrealengine.com/en-US/license

Ups, T. P. (s.f.). *The Power Ups*. Recuperado el 10 de Febrero de 2024, de https://thepowerups-learning.com/descargar-ejemplo-patron-mvvm-reactivo/

Wikipedia. (s.f.). Recuperado el 31 de Octubre de 2023, de Wikipedia | Unity: https://en.wikipedia.org/wiki/Unity_%28game_engine%29

Wikipedia. (s.f.). Recuperado el 31 de Octubre de 2023, de Wikipedia | Unreal Engine: https://es.wikipedia.org/wiki/Unreal_Engine

Wikipedia. (s.f.). *Modelo COCOMO*. Recuperado el 31 de Octubre de 2023, de https://es.wikipedia.org/wiki/COCOMO