TecnoCampus

Centres universitaris adscrits a la

upf. Universitat Pompeu Fabra Barcelona

Grado en Diseño y Producción de Videojuegos

Creación de un videojuego de mánager online de un club deportivo
MEMÓRIA FINAL

Xavier Gesa Bages
Tutor: Jorge Arnal Montoya

TecnoCampus Mataró-Maresme

Agradecimientos

Al Club Handbol Palautordera-Salicru por su colaboración en el desarrollo del videojuego.

Abstract

In this work, it is explored the PlayFab platform that allows the creation of an online video game in the Unity engine. For this reason, research is carried out on the development of different sports management games in order to develop one of their own. This will consist of an online manager video game about an existing handball sports club.

Resum

En aquest treball s'explora la plataforma PlayFab que permet la creació d'un videojoc en línia en el motor de Unity. Per això, es duu a terme la investigació del desenvolupament de diferents jocs de gestió esportiva per tal de poder elaborar-ne un de propi. Aquest consistirà en un videojoc de mànager en línia sobre un club esportiu d'handbol existent.

Resumen

En este trabajo se explora la plataforma PlayFab que permite la creación de un videojuego online en el motor de Unity. Por ello, se lleva a cabo la investigación del desarrollo de diferentes juegos de gestión deportiva para poder elaborar uno propio. Éste consistirá en un videojuego de mánager online sobre un club deportivo de balonmano existente.

Índice

Índice de ilustraciones

1. Introducción

Hoy en día todo el mundo ha jugado a videojuegos online porque a los usuarios les encanta tener que competir con jugadores desconocidos de otras partes del mundo o amigos. Sin embargo, durante la carrera no se ha tenido la oportunidad de aprender mucho sobre como poder crear uno. En cambio, sí se ha tenido la posibilidad de hacer juegos del tipo multijugador local, los cuales podrían asemejarse. De aquí surge la motivación para elaborar este proyecto, ya que es algo novedoso y se requiere de una intensa investigación para llevarlo a cabo.

En los videojuegos se conoce un género que son los del tipo mánager o de gestión deportiva. Estos se basan normalmente en el control de una entidad, club o equipo deportivo en los cuales hay una simulación de la gestión de sus recursos. En los juegos de mánager online, la competitividad reside en ver qué jugador es el que hace una mejor gestión del equipo. Dentro de este, existen otros géneros, como, por ejemplo, unos donde el juego está relacionado con lo que sucede en partidos de la vida real u otros donde los partidos son simulados por el juego.

Los juegos en donde se requiere de datos de la vida real para poder seguir funcionando y de un sistema de obtención de ellos, manejan muchos datos. Esto hace que prácticamente sea imposible que sea manejado por una sola persona. Es por eso por lo que para el proyecto se añade otro reto que es el de crear una inteligencia artificial (IA) que pueda simular y crear resultados con un mínimo de sentido y sin que todo sea un factor de aleatoriedad. Aunque este apartado sí que ha sido trabajado en la carrera, no deja de ser algo a tener en cuenta.

En cuanto al proyecto, este consistirá en la creación de un videojuego del tipo mencionado, es decir, un videojuego de mánager online pero centrado en un club deportivo. Básicamente, los usuarios tendrán que formar una alineación con los diferentes jugadores de las distintas categorías de un club en cada jornada, y los que sean definidos como titulares van a recibir una puntuación que se va a definir dependiendo de su actuación en los partidos. La actuación de estos dependerá de factores como la valoración del jugador, su posición en el campo, el rival… De esta

forma los enfrentamientos de cada jornada serán entre los usuarios que formen parte de la liga que haya entre ellos. Los usuarios van a poder pujar o vender jugadores para mejorar su plantilla.

El club elegido para el proyecto es el Club Handbol Palautordera-Salicru (1956), por lo tanto, este juego tratará de balonmano.

Para llevarlo a cabo y antes de empezar, se deberá pensar en el diseño del juego y de la IA que va a simular la actuación de los jugadores. Esto provoca que se necesite, aparte de conocimiento para desarrollar videojuegos online, también sobre el deporte y del club, y/o extraer información sobre los jugadores del club y de cómo abundan normalmente las estadísticas en el balonmano.

2. Objetivos

Para definir los objetivos, primero cabe aclarar que se va a utilizar el motor de Unity (Unity Technologies, 2005) y la plataforma PlayFab (Microsoft Corporation, 2014) que permite la conexión entre el usuario y servidor.

El objetivo principal es, con la ayuda de PlayFab, desarrollar un videojuego online (APK para móvil) de Mánager sobre el Club Handbol Palautordera-Salicru, elaborar una IA capaz de simular los datos de los partidos y así determinar la puntuación de los usuarios. Para ello, hará falta una investigación de ejemplos de desarrollo de juegos pertenecientes de igual o similar forma al género, un pequeño estudio de las estadísticas del balonmano u otros deportes y también explorar los conceptos principales de los videojuegos online.

3. Marco teórico

3.1. Videojuegos de mánager

Los juegos de mánager son juegos virtuales en los que cada usuario puede administrar a un equipo como él crea conveniente mediante fichajes, alineaciones y muchos otros aspectos con el objetivo de llegar a ser campeón en la competición del juego. Muchos de ellos ofrecen la opción de competir con otros usuarios o contra sus propios amigos.

Así pues, después de analizar diferentes juegos de mánager que más adelante se podrán ver algunos en el apartado de 4 (Referentes), se puede determinar que un juego forma parte de esta categoría si contiene las siguientes características:

- Gestión de un equipo o club: Esta función es vital que aparezca con el máximo nivel de opciones para poder dar al usuario una experiencia que haga sentirle como el entrenador que está teniendo el control absoluto del equipo. Cuando se habla de gestión se hace referencia a alineación, venta y adquisición de jugadores, entrenamientos…

Ilustración 3.1: La gestión del equipo antes de un partido en el juego de PC Fútbol 18 (IDC Games, 2013)

- Competición contra otros usuarios o contra la IA: El usuario necesita competir contra alguien, ya sean amigos, usuarios desconocidos o una IA.

Dentro de esta categoría de juegos de mánager se han encontrado tres géneros diferentes:

- Juegos de mánager online basados en hechos que suceden en la realidad: Este género consiste en la gestión de un equipo que el usuario deberá formar con jugadores que formen parte de una liga de la vida real. Los usuarios pueden jugar una liga privada con amigos o unirse a una liga con jugadores desconocidos. Estos pueden sumar puntos en cada jornada que se juegue en la liga. La cantidad de puntos que consigan en esa jornada dependerá de la actuación de sus jugadores de la plantilla en la vida real. En estos juegos se suele generar una competición entre usuarios por quién tiene más conocimiento de la liga y por quién sabe llevar a cabo una mayor gestión.
- Juegos de mánager online: Estos juegos conllevan la gestión de un club donde se compite contra otros usuarios. En este género los otros usuarios también se encuentran gestionando su club.

  En este género existen dos tipos de juegos:
  - Los que, en el partido contra los usuarios, el usuario lo puede jugar y gestionar.
  - Los que, en el partido contra los usuarios, el usuario no puede jugar el partido, solo puede gestionarlo. Por lo tanto, este tipo cuenta con una IA que se encarga de simular el partido.

  Esta gestión del partido sucede mediante cambios de jugadores o cambios de estrategia como intentar aumentar la presión, cambiar la formación, el estilo de juego…

- Juegos de mánager sin online: Este tipo de juegos es igual al anterior, pero sin el factor online. El jugador juega contra equipos controlados por la IA.

3.1.1. Desarrollo de videojuegos de mánager

Para descubrir cómo es el desarrollo de un videojuego de mánager, se han encontrado entrevistas donde desarrolladores hablan sobre su propio juego y dan información importante sobre su núcleo, tecnología, producción, organización, estadísticas…

3.1.1.1. Contenido de los videojuegos de mánager

En este apartado se puede se encuentra información que dan desarrolladores sobre el contenido de los juegos de mánager.

3.1.1.1.1. Añadir características al juego

El núcleo de los juegos de mánager como bien se ha visto es hacer las alineaciones, las tácticas, los fichajes, preparar entrenamientos… Los juegos evolucionan y hay que introducir nuevos sistemas o modos que hagan aumentar el realismo de este. Abril (2021), programador de PC Fútbol (Dinamic Multimedia, 1994) de los años 90, contaba en una entrevista del 2018 la presión que había para innovar en este juego, ya que había que darle un poco la vuelta porque hacer alineaciones, hacer las tácticas, entrenar un poco a los jugadores para poder subir las estadísticas ya no estaban dentro de lo que sería el "core" del juego. Había que salirse de ello y empezar a introducir cosas nuevas.

Evidentemente, la base de un juego de mánager debe siempre ser la misma para mantener su esencia y que el resto sea accesorio. De hecho, Abril (2021) cuenta que una vez tuvieron la parte principal del juego bien definida, luego empezaban a introducir aspectos accesorios. Y que, respecto a esta parte, reconoció que la parte de "merchandising" del juego existía porque la pedían los usuarios. Este "feedback" con los usuarios es el que le llevó a ser exitoso. Además de que también cuidaron mucho el marketing del juego.

Ilustración 3.2: Gameplay de PC Fútbol 3.0 (Dinamic Multimedia, 1994)

De hecho, Abril (2021) cuenta que el “feedback” con los usuarios fue tan bueno, que cuando él recibió a Rafa Benítez (entrenador de fútbol) en sus estudios del juego, este le preguntó si podía añadir en la base de datos de la ficha de los jugadores un botón de "Notas" porque él lo utilizaba como referencia para buscar jugadores. Y así se añadió esta opción.

Ruíz (2018), cofundador de Dinamic, explica en una entrevista que el juego se inventó cuando dos personas vieron en un quiosco una base de datos de equipos y futbolistas; y de ahí les surgió la idea de mezclar la base de datos con el juego.

Así pues, se puede observar que el juego se creó con esta idea y evolucionó con los pedidos de los usuarios.

Pero, PC Fútbol (Dinamic Multimedia, 1994) no ha sido el único en fijarse en los comentarios de la comunidad, según Jacobson (2020), director de Sports Interactive y hablando de una nueva entrega de Football Manager (Sports Interactive, 1994) en una entrevista en Meristation dijo:

> Tenemos cientos de ideas sobre funciones a añadir que entran en nuestro proceso de aprobación que todavía no han llegado al juego. Y tenemos más ideas cada año, ya sea de foros, del equipo interno, de futbolistas, de los análisis que leemos, de la gente redes sociales… Cada año tenemos más ideas de las que podemos hacer en un mismo ciclo de desarrollo.

Rivera (2018), uno de los productores de FIFA (EA, 1982), en una entrevista en Eurogamer contó que:

> Hemos recibido muchísimo *feedback* y un montón es realmente bueno. Antes incluso de la beta cerrada tuvimos sesiones con bastantes jugadores, y realmente notaron que hay una gran diferencia entre FIFA 18 y FIFA 19, y eso es una parte muy importante. Obviamente cada año hay cosas que retocamos. El principal propósito de la beta cerrada es obtener feedback y seguir arreglando cosas, porque no es la versión final.

En este caso, la empresa EA cuenta con mucho más potencial económico permitiéndose disponer de una beta cerrada para obtener “feedback”. Lo más

importante a destacar es que muchos retoques que se producen en el juego son a partir de comentarios de usuarios.

También, Milutinovic (2020), CEO de Nordeus, hablando sobre Top Eleven (Nordeus, 2010), en una entrevista en Applicantes dijo:

> Creo que una de nuestras fortalezas es el reconocimiento de marca y la longevidad. Creo que eso cuenta en cualquier industria del mundo. Esa estabilidad y el éxito continuo muestran que debemos estar haciendo algo bien. Tenemos y debemos buscar desarrollar nuevas funciones, escuchar a nuestros jugadores y seguir innovando.
>
> […]
>
> Todavía hay muchas cosas que podemos hacer (nuevas funciones, actualizaciones, “loops” principales): se trata de encontrar el equilibrio y lo que es correcto para los jugadores, pero también para el juego y el equipo.

Así que ya son varios los ejemplos de juegos que dan esa importancia al usuario y que usan una estrategia para seguir innovando en el contenido ofrecido. Las desarrolladoras buscan aspectos nuevos y mejoras que le puedan dar al juego para evitar una monotonía que dé la sensación de que este siempre sea lo mismo.

A parte de que los desarrolladores se respaldaran en las opiniones de los usuarios, también es habitual crear contenido inspirándose en otros juegos como en el caso de Sociable Soccer (Tower Studios, 1990). Hedenborg (2021) dijo sobre su juego:

> Hemos tomado cierta influencia de la parte de MyClub / FIFA Ultimate Team de los juegos al crear tus propios equipos: por ejemplo, cómo agregar jugadores legendarios a tu equipo.
>
> También verás diferentes desafíos en el mundo del fútbol mientras juegas. Hemos analizado un poco cómo Nintendo crea juegos deportivos porque son unos maestros en hacer juegos divertidos y encantadores sin renunciar a la sensación de competitividad.

De esta forma, se puede observar cómo empresas hacen estudios de mercado e investigan lo que hacen otras competencias para aprender sobre ellas, algo que no es nuevo en este sector y que es muy útil.

Ilustración 3.3: Gameplay de Sociable Soccer ’21 (Tower Studios, 2021)

3.1.1.1.2. Dificultades para las desarrolladoras

Sin embargo, pese que parezca todo muy fácil también han aparecido grandes dificultades en este género que se tratarán a continuación. Algunas son típicas del sector y otras más en el género de juegos deportivos. Milutinovic (2020) afirmaba con su juego:

> Llegamos a un punto con Top Eleven en el que no le dedicábamos el cariño que requería, ya que estábamos distraídos creando algo brillante y nuevo y olvidamos nuestra esencia central y las cosas sobre las que construimos Nordeus.
>
> Así que, en 2018, pasamos mucho tiempo echando la vista hacia atrás y mirando los fundamentos del juego, profundizando en los principales problemas y *pain points* (puntos de dolor) que tenían los jugadores, pero también en las razones técnicas que estaban conduciendo a que produjeran esos problemas.

Milutinovic daba a entender con estas palabras que, al estar enfocados en aspectos nuevos, se olvidaron de cuidar las partes principales del juego que al final son las más importantes. Es decir, pese a que es importante añadir características accesorias para renovar el juego, su base no debe ser olvidada y dejar de ser cuidada, ya que es la parte que lo sostiene. Por esta razón, Nordeus en 2018 se dedicó a observar cuáles eren los principales problemas que había en los fundamentos del juego.

Ilustración 3.4: Gameplay de Top Eleven (Nordeus, 2010)

En Football Manager (Sports Interactive, 1994) también hubo problemas para añadir nuevas características. Jacobson (2020) habló sobre el año cuando surgió la pandemia:

> Este año ha sido duro. Hemos cortado algunas características alrededor del final del ciclo del juego porque no íbamos a poder hacerlo de manera que les hiciera justicia. Pero hubo otras que características que hemos traído de versiones futuras, como las reuniones para transferencias o la función de poder preguntar la disponibilidad de los agentes; esas funciones no estaban pensadas para Football Manager 21. Originalmente una de ellas estaba pensada para Football Manager 22, otra estuvo para Football Manager 23. Pero encajaron bien con las otras características que estábamos pensando y haciendo, y pudimos lograr hacerlas en este ciclo. Mientras había otras cosas que no podíamos lograr en el mismo ciclo. Así que hemos movido esas para futuros juegos.

Aunque estas dificultades fuesen más dirigidas a las consecuencias de la pandemia, no deja de ser interesante la redistribución en la que esta empresa decidió analizar cuáles eran las características que se podían desarrollar para el juego y si estas podían encajar bien pese a estar pensadas para otras entregas.

Otra dificultad existente actual para Hare (2021), diseñador del juego Sensible Soccer ’22 (Tower Studios, 1990), aseguraba en una entrevista:

> Era mucho más fácil lanzar un juego de fútbol en los días en que las licencias de jugadores y equipos no era un requisito. El mayor problema ahora es la cantidad de dinero que se debe pagar solo para usar los nombres e insignias correctos de jugadores y equipos, etc.

Antiguamente, esta cuestión era una ventaja para PC Fútbol (Dinamic Multimedia, 1994) y actualmente es una desventaja para los juegos deportivos en general.

Hare (2021) añadió:

> Tanto en Sociable Soccer como en Sensible Soccer hay más de 1.000 equipos y más de 25.000 jugadores, por lo que en estos días el costo de la licencia es de millones de euros, incluso si tu nuevo juego no es un fracaso, este es un gran riesgo para muchos desarrolladores y editores más modestos.

Por lo tanto, esto prueba la gran dificultad actual existente en términos económicos para hacerse cargo de un juego deportivo que quiera basarse en equipos y ligas reales. Por este motivo, muchos juegos modifican el nombre y el logo de los jugadores y equipos haciendo que este se parezca, pero no sea el mismo para así no tener que pagar la licencia.

3.1.1.2. Producción y organización

En este apartado se observarán detalles de producción y organización que reveló Jacobson hablando sobre su juego Football Manager (Sports Interactive, 1994). Este juego lanza una nueva entrega a finales de cada año y conlleva una gran organización en términos de calendario donde sus trabajadores tienen que cumplir con las fechas para poder seguir con el proceso de desarrollo del juego. Jacobson (2020), después de haber lanzado el Football Manager 21 (Sports Interactive, 1994), explico el proceso resumido en el que trabajarían ese año:

> En estos momentos se está trabajando en un documento de diseño para una de las grandes funciones que buscamos tener el próximo año. Y entonces

> tendremos más reuniones de diseño en enero y febrero, donde la gente desgranará las tareas y estimará cuántas son. Y entonces con suerte para final de febrero, que habitualmente es sobre final de abril (intentamos adelantarlo), tendré una compresión completa sobre cuántas horas tomará a una persona y si está por debajo o sobre el ciclo. Y con eso intentaremos contratar a nuevos empleados para arreglar la disparidad en algunas de las nuevas características.

Es decir, que se intenta saber la cantidad y estimación de tiempo de las tareas a finales de febrero, pero habitualmente acostumbran a tenerlo a finales de abril. Y a partir de ese momento es cuando se empieza a trabajar en producir un juego que acostumbra a ser lanzado en noviembre.

También Jacobson (2020) mencionaba:

> Algunas veces hay algunos programadores que están ligeros, entonces miro en nuestro repositorio de funciones, que son las pequeñas funciones que no han sido diseñadas todavía y muevo algunas de las que no han sido asignadas hacia el juego. Es un proceso largo, y espero que podamos tenerlo completado dentro de poco, ya que eso nos dará más tiempo para trabajar en más funciones y ofrecer todavía más a los consumidores.

Esto, para personas que ya hayan trabajado en el desarrollo de videojuegos les será algo obvio y normal seguramente, pero es una confirmación de que los trabajadores no paran de trabajar una vez han terminado las tareas que tenían asignadas. Como bien dice Jacobson, si por ejemplo un programador termina sus tareas muy rápidamente, se le asigna de otras para seguir avanzando en el desarrollo.

3.1.1.3. Tecnologías

Para el desarrollo de un videojuego, saber elegir o elaborar una tecnología es muy determinante, ya que es la herramienta con la que se va a trabajar. En este apartado se observará las tecnologías usadas por algunos desarrolladores deportivos.

En el caso de PC Fútbol, Abril (2021) mencionó en una entrevista que fue desarrollado con tecnología GFXLib, que era la parte que había programado él con animaciones.

Entre esas rutinas menciona una que convertía las imágenes escaneadas en 256 colores a gráficos de 16 colores. Abril (2021):

> Con el entramado de punto sí, punto no, y los convertía mejor que Photoshop, quedaban genial. Así empezamos a meter los escudos, las fotos de los jugadores… hasta que pasamos a Windows en PC Fútbol 5 usamos esas librerías, que yo iba mejorando cada vez.

En este párrafo Abril estaba hablando de los años anteriores a 1997 y cuando llegó ese año, fue cuando salió el PC Fútbol 5.

Entonces, GFXLib era una librería gráfica creada por Abril que una de sus utilidades fue la de la manipulación de píxeles de bajo nivel. Con este se podían cargar imágenes que se habían escaneado en 256 colores y transformarlo a 16 colores. Esto permitía que el juego no ocupara tanto ya que en esos tiempos era muy importante el peso del juego.

Ilustración 3.5: PC Fútbol 3.0 (Dinamic Multimedia, 1994)

Abril (2021) también expuso que cuando intentaba programar para Windows 95 la diversidad del hardware le daba problemas y por eso usaba la librería Direct X que no tenía problemas de compatibilidad y que facilitaba las complejas tareas relacionadas con multimedia, especialmente programación de juegos y vídeo, en la plataforma Microsoft Windows.

La tecnología es un problema para muchos estudios porque siempre hay que adaptarse a lo que existe en la actualidad y sus tendencias, y dejar atrás la tecnología antigua que se usa menos. Es en el caso de la empresa Sports Interactive, con su juego Football Manager (1992), donde se notó mucho para a empresa los cambios tecnológicos. El director, Jacobson (2020), dijo:

> Cuando empecé a trabajar en Championship Manager (1992) la Premier League ni siquiera existía. Los juegos salían en televisión cada semana. La gente estaba menos interesada en datos y estadísticas. Así que el fútbol ha

> cambiado muchísimo. Desde la perspectiva del juego solo teníamos comentarios, ni siquiera un motor 2D, y mucho menos uno 3D en aquellos momentos. Este año contamos con mejoras gráficas y mejoras en las animaciones que nos permiten alcanzar un mejor nivel.

Como dice Jacobson, las tendencias de la gente han cambiado y con el avance de la tecnología las estadísticas son más interesantes para los aficionados. Por la parte tecnológica hubo una gran evolución en esta empresa ya que pasaron de no contar con un motor 2D a tener unas mejoras gráficas notables.

Ilustración 3.6: Gameplay de Championship Manager (Sports Interactive, 1992)

Jacobson (2020) añadió:

> En aquellos momentos hacíamos avances para ordenadores con disquetes que eran menos potentes que la mayoría de los relojes digitales que hay ahora mismo en los hogares, por no hablar de los smartwatches. Y ahora tenemos un montón de potencia, al poner los sistemas al día se traduce en que necesitas más memoria RAM. Eso fue una de las cosas que, cuando nos sentamos a trabajar en los requisitos del sistema para el juego, alguien se dio la vuelta y dijo: "eh, ¿podemos dejar atrás esta tarjeta gráfica?". Yo estaba como, vale, primero, ¿cuántos años tiene? Y segundo, ¿cuántas personas siguen usándola? Y encontramos que seguíamos dando soporte a una tarjeta gráfica de hace 18 años. Menos de 200 personas seguían usándola. Las cosas han cambiado muchísimo en todo.

En esta cita, cuando se menciona la tarjeta gráfica, se ve claramente la evolución que ha habido y lo importante que es actualizarse. Se estaba dando soporte a una tarjeta gráfica bastante antigua y que poquísimas personas usaban. Sports Interactive estaba

obligada a actualizarse en este aspecto para poder seguir siendo competitiva y tener hueco en el mercado.

Hare (2021), diseñador de Tower Studios, explicó:

> En general, el período 2005-2021 es más fácil y mejor que el período 1995-2005 si tenemos en cuenta todo el enfoque de hacer que los gráficos 3D funcionen y que los editores apuesten por secuelas y licencias vinculadas. Pero hablar desde la perspectiva de los diseñadores de juegos el periodo actual es mucho peor que el período 1985-1995, cuando teníamos mucha más libertad para hacer lo que quisiéramos en máquinas que eran mucho más fáciles de controlar.

Ilustración 3.7: Gameplay de Sensible Soccer (Tower Studios, 1992)

Ilustración 3.8: Gameplay de Sociable Soccer '21 (Tower Studios, 2020)

Hare dividió la historia moderna de los videojuegos de fútbol en dos etapas la de 2005-2021 y la de 1995-2005. Para él es mejor la época actual, aunque como diseñador prefiere el período 1985-1995 cuando las máquinas recreativas estaban en auge. Hare (2021) añadió:

> Desde la perspectiva de los diseñadores, también es mucho más lento avanzar en el juego de desarrollo de títulos donde existe una necesidad constante de desarrollar y mantener el diálogo entre un servidor y el cliente. Esto se debe al juego en línea y a los sistemas de progreso y recompensa en línea, lo que reduce a la mitad la velocidad de progreso, y eso hace que el diseño pueda ser muy frustrante.

> […]
>
> La pérdida del control total es el principal problema para los diseñadores de todo tipo de juegos en las máquinas más modernas. Hay muchos factores, como la integración de redes sociales, los sistemas de monetización, los protocolos en línea, la privacidad de los datos, los hackeos, las regulaciones del titular de cada plataforma, el software de terceros y otras consideraciones que distraen a los programadores de la tarea de hacer el mejor juego posible.

La perspectiva de Hare como diseñador da una importante reflexión donde hace ver que, hoy en día para los juegos, primero existen unas prioridades en programación antes que mejorar el juego en su diseño. Destaca que la mayoría de las prioridades que tiene el programador están relacionadas con el juego en línea. Se podría decir que la aparición del online*,* pese a haber generado grandes ventajas, también ha generado desventajas que se ven reflejadas en las tareas de los programadores.

Por otra parte, uno de los juegos que hizo Tower Studios es Sociable Soccer (2017), desarrollado con el motor de Unity. Para Hare (2021), la aparición de Unity fue un soplo de aire fresco, aunque como diseñador estaba más cómodo en el período 1985-1995 y así lo manifestó con estas palabras:

> La llegada de motores como Unity ha sido una bendición en comparación con los tiempos en que teníamos que hacer nuestros propios motores 3D y realmente ha ayudado con algunos aspectos del desarrollo multiplataforma, sin embargo, desde una perspectiva personal, nunca será tan bueno como poder controlar personalmente todos los aspectos del diseño y los gráficos en el Commodore Amiga, como sucedió cuando creamos el Sensible Soccer original.

Hare es un ejemplo en el que se puede observar que la tarea del diseñador se ve limitada por las capacidades del estudio. En su caso, el surgimiento de nuevas tareas por la evolución tecnológica de los videojuegos ha provocado que el programador tenga más trabajo en otros campos que no sean relacionadas con el diseño del juego.

Derwort (2014), uno de los fundadores de Gamebasics BV que desarrollaron Online Soccer Manager (2012), mostró que la tecnología con la que trabajaban para desarrollar este era:

- MariaDB (MariaDB Foundation, 2009), un sistema de gestión de base de datos.
- Redis (Redis Labs, 2009), un motor de base de datos en memoria.
- Windows Server 2012 (Microsoft, 2012), un sistema operativo destinado a servidores.

Conviene destacar que este estudio en 2014 usaba dos tecnologías para la gestión de base de datos. Además, según Derwort, en Online Soccer Manager se realizan aproximadamente 20.000 operaciones de base de datos por segundo, por lo tanto, se puede observar que este era un factor importante para el juego.

Ilustración 3.9: Gameplay de Online Soccer Manager (Gamebasics BV, 2012)

3.1.1.4. Estadísticas

Las estadísticas, como se ha reflejado en anteriores citas de desarrolladores, cada vez han sido más interesantes para el espectador en el mundo del deporte. Además, para llevar a cabo la simulación de un partido deportivo, es vital tener un buen control y conocimiento de las estadísticas que suelen haber en los partidos.

Antes de nada y como a modo de introducción, se describirá brevemente el videojuego llamado Comunio (comunio, 2000). El objetivo de este es formar el mejor equipo posible con jugadores de la liga española. Los jugadores del usuario en concreto sumarán unos puntos en relación con la actuación que hayan tenido en su partido jugado en la vida real. El usuario puede fichar o vender jugadores. Los usuarios del juego compiten en una liga jornada tras jornada formando la alineación que ellos crean que les dará más puntos. Gana el usuario que haga más puntos.

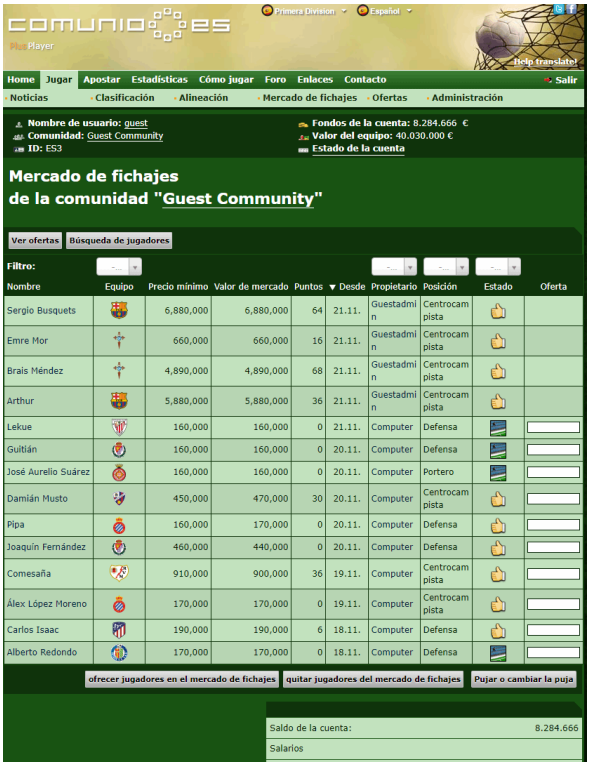

Ilustración 3.10: Gameplay del Comunio (comunio, 2000)

Así pues, Comunio debe contar con un sistema con el que, a partir de la obtención de unas estadísticas, genere los puntos. Las puntuaciones siempre dan mucho que hablar, pero algunos desconocen que en gran parte se basan en las notas de SofaScore (SofaScore, 2010). Esta es una página o aplicación de móvil que permite visualizar datos y estadísticas de muchos deportes. En la mayoría de los partidos de fútbol, este calcula una nota del jugador en función de sus estadísticas.

Letica (2020), uno de los responsables de estas notas, en una entrevista en Comunio Magazine contó:

> Nuestras notas se calculan gracias a un algoritmo especial que tiene en cuenta cerca de 200 estadísticas de cada jugador en cada partido. El algoritmo está configurado de tal forma que da un valor específico a cada estadística dependiendo de varios factores. Por ejemplo, una entrada no tiene la misma repercusión para un delantero que para un defensa, una tarjeta roja será evaluada de forma diferente dependiendo del minuto en que se muestre, etc.

Pues bien, todos esos valores otorgados a cada estadística dan como resultado una nota de 3 a 10 que es lo que se muestra en nuestra plataforma. Las notas no reflejan si un futbolista ha jugado bien o mal, ya que todo el mundo que ve un encuentro de fútbol tiene su propia opinión y hay cosas que no pueden ser medidas con números. Lo que pretenden mostrar es el impacto estadístico del jugador en el partido. Un jugador que consigue un montón de acciones estadísticas positivas como toques, pases precisos, regates o tiros, tendrá siempre una mejor nota que otro jugador que apenas acumule estadísticas positivas.

Comunio aprovecha estos datos proporcionados por SofaScore (SofaScore, 2010) para dar la puntuación de los jugadores en esa jornada y de esta forma no se preocupa de las estadísticas. Es decir, Comunio únicamente obtiene la nota que pone SofaScore sobre el jugador para dar una puntuación. Mientras que SofaScore es la que cuenta con el algoritmo que, a partir de las estadísticas, da una nota del 1 al 10 al jugador.

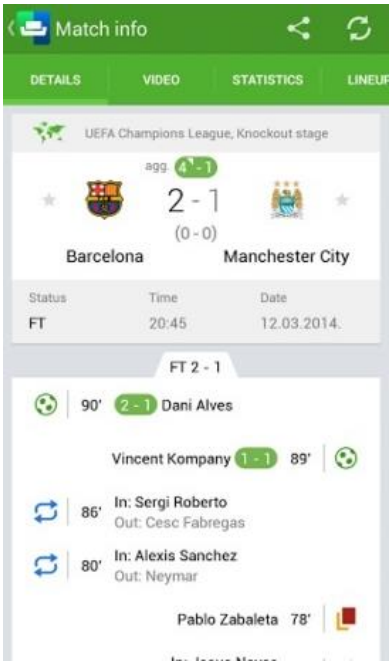

Ilustración 3.11: La aplicación de SofaScore (SofaScore, 2010)

Además, Letica (2020) añade que:

> Cada jugador es evaluado del mismo modo por nuestro algoritmo. Puede haber diferencias por la posición en el campo, ya que un defensa y un centrocampista que tengan las mismas estadísticas no tendrán la misma nota porque sus acciones son ponderadas por el algoritmo de forma distinta.

Este último comentario iba dirigido a los usuarios que les da la sensación de que no todos los jugadores son evaluados de la misma forma. Letica con esto quiere decir que es imposible que esto ocurra porque es el algoritmo quien pone la nota.

En relación con el algoritmo, Letica (2020) explicó que:

> Ganar el partido no tiene, en teoría, ningún impacto en las notas. No hay una estadística «partido ganado» en nuestro algoritmo. Hay estadísticas por goles concedidos o marcados que sí que pueden tener impacto en todo el equipo, pero puede ocurrir que el equipo perdedor pueda tener mejores notas que el vencedor si su impacto estadístico en el partido ha sido mayor. El estilo de juego sí puede tener efecto en las notas, ya que los equipos que tengan mucha posesión hacen que sus jugadores realicen más toques, pases, ataques y tengan más fácil acumular estadísticas positivas, en especial sus defensas y medios.

Es decir, que los jugadores son evaluados independientemente del resultado del partido. Esto hace que, por ejemplo, si un delantero marca tres goles, pero el otro equipo marca cuatro, este delantero probablemente tenga una nota bastante buena porque los responsables principales de recibir cuatro goles son los defensas y el portero. En otras palabras, la nota de un jugador no puede depender de la de otro porque son notas individuales.

Letica (2020) pone el caso de Messi con el que esto se acaba de respaldar:

> Messi tiene un impacto estadístico brutal en cada partido. Por poner un ejemplo, en el partido en el que el Bayern ganó 8-2 al Barça, Messi no marcó ni asistió, pero aun así remató 3 veces, completó 3 de 5 regates, realizó un disparo al palo, 6 de 7 balones largos, ganó 8 duelos de 15 y recibió 4 faltas.

> Esas estadísticas son increíbles y si el Barcelona no hubiera concedido 8 goles, Messi probablemente habría tenido una nota cercana al 8.0.

En este partido que Letica pone de ejemplo, Messi sacó en SofaScore una nota de 7,2 sobre 10. No fue la nota más alta del Barça porque Luis Suárez sacó un 7,3. El resto del equipo obtuvo notas alrededor del 5 y 6. La media de las notas del Barça fue un 6,08 y las del Bayern un 7,51. Esto prueba que, no hace falta ganar para que un jugador saqué buena nota, pero las medias de las notas del equipo tenderán a ser más bajas en general.

Por otra parte, Letica (2020) reveló que:

> Las notas se basan en estadísticas y éstas no las recogemos nosotros mismos, de eso se encarga nuestro proveedor, OPTA. Esta empresa revisa minuciosamente cada estadística de cada jugador incluso varias horas después de haber finalizado el encuentro, en especial en las grandes ligas. Si finalmente corrigen una estadística mal apuntada o incluyen alguna pasada por alto en la revisión, puede afectar a nuestras notas, de ahí que haya cambios. Normalmente, nuestras notas se pueden considerar definitivas 4-5 horas después de acabar cada partido.

Esto significa que SofaScore no se encarga de calcular las estadísticas de los partidos, es otra empresa llamada OPTA (Stats Perform, 1996) quién lo hace. Por lo tanto, SofaScore únicamente se encarga de mostrarlo en su página o aplicación y de otros procedimientos como el de calcular las notas.

3.1.1.4.1. Estadísticas en el balonmano

Por lo tanto, está claro que, para simular un partido, hay que conocer como acostumbran a ser las estadísticas del deporte, en este caso, el balonmano.

Sáez, Roldán y Feu (2009) hicieron un estudio de las diferencias en las estadísticas de juego entre los equipos ganadores y perdedores de la copa del rey 2008 de balonmano masculino.

Los resultados se pueden observar en la Ilustración 3.12 donde todos los datos están normalizados a 100 posesiones de balón para evitar el efecto contaminante que supone el diferente ritmo de juego (Ibañez, Sampaio, Sáenz-López, Giménez & Janeira, 2003; Sampaio, Ibañez, Feu, 2004; Oliver, 2004). "Esta normalización permite comparar y analizar los datos de diferentes partidos" (Sáez et al., 2009). En la tabla se calculan los siguientes coeficientes:

	Resultado			
	Victoria		Derrota	
	M	*DT*	*M*	*DT*
Goles marcados	57.804	5.833	50.150	3.809
Goles 9m	18.529	5.598	15.886	3.065
Lanz. fallados 9m	16.194	3.920	18.387	3.054
Goles 6m	22.754	3.899	20.577	6.309
Lanz. fallados 6m	7.626	2.7104	13.251	3.546
Goles7m	5.13	3.171	6.35	2.769
Lanz. fallados 7m	3.081	2.126	1.954	1.387
Goles contraataque	10.850	3.225	7.337	2.740
Lanz. fallados contra.	2.127	2.396	2.601	2.780
Lanzamientos totales	86.290	6.2196	86.344	3.730
Pases de gol	9.329	3.296	6.156	3.929
Perdidas	13.710	6.2196	13.656	3.730
Recuperaciones	3.873	3.986	5.697	4.133
Paradas 9m	14.371	3.8810	12.596	4.578
Paradas 6m	9.567	2.683	6.640	1.775
Paradas 7m	1.951	1.497	3.081	2.135
Paradas contraataque	9.533	3.417	13.493	4.626
Paradas totales	27.980	4.015	20.539	9.527
Exclusiones	7.2843	2.199	6.3400	4.602
Desc. directas	.000	.000	.2343	.6199
Expulsiones	.000	.000	.000	.000
Posesiones	58.142	4.598	58.000	5.538
CEO	57.804	5.833	50.150	3.809
CRO	66.939	3.666	58.148	4.848
CED	50.150	3.809	57.804	5.833
CRD	58.148	4.848	66.937	3.665

Ilustración 3.12: Medias y desviaciones típicas de la Copa del Rey 2007/08 (Sáez et al., 2009).

- Coeficiente de eficacia ofensiva (CEO): CEO = Nº de Goles x 100 / Nº de posesiones.
- Coeficiente de resolución ofensiva (CRO): CRO = Nº de Goles x 100 / Nº de lanzamientos.
- Coeficiente de eficacia defensiva (CED): CED = Nº de goles recibidos x 100 / Nº de acciones defensivas.
- Coeficiente de resolución defensiva (CRD): CRD = Nº de Goles recibidos x 100 / Nº de lanzamientos recibidos (Sáez et al., 2009).

Se observa que los equipos ganadores de cada 100 posesiones consiguen anotar en 58, mientras que los perdedores 50, por lo tanto, tienen mejor eficacia ofensiva. Como se puede ver, esto se debe a que en los equipos ganadores los porteros y la defensa tienen una mucha mejor actuación. Además, gracias a una buena defensa, consiguen muchas más anotaciones de contrataque. El contraataque, como bien se sabe, proviene de recuperaciones de balón, paradas del portero, tiros desde zonas desfavorables, faltas en ataque… Todo debido a una buena defensa.

El coeficiente de resolución ofensiva (CRO) es muy diferente entre los equipos ganadores y perdedores. Los ganadores suelen tener mucha más eficacia en situaciones de lanzamiento.

Finalmente, se puede observar que la media de las posesiones de balón por partido es de 58 por equipo. Czerwinski (1994) realizó un análisis descriptivo de los partidos internacionales comprendidos entre el año 1970 hasta 1992 y afirmó que en un partido se producen alrededor de 50 posesiones y que el 17% son acciones cortas (5''-20''), el 61% acciones medias (21''-35'') y 22% acciones (+35''). Este es un factor importante a tener en cuenta para simular los partidos porque será a partir de las posesiones que dispongan los dos equipos con lo que se calculará que sucede en cada posesión.

3.1.1.5. Calendario de partidos

En cualquier liga formada por diferentes equipos deportivos que se deben enfrentar entre ellos existe un calendario de partidos. Este busca que dos equipos solo se enfrenten una única vez por vuelta. Una vuelta es completada cuando todos los equipos se hayan enfrentado una vez en esa misma vuelta. Para crear este calendario de partidos, existe el sistema "Round Robin".

Según Landeta y García (2010):

> La metodología para programar un calendario de partidos bajo el sistema "Round Robin" se basa en escoger un equipo comodín que es justamente el que salva la situación hipotética de que a un equipo le tocase enfrentarse a sí mismo, lo cual simple y sencillamente no es posible. El resto de equipos, sea cual sea su número, deben seguir el rol normal de actividades, es decir el equipo x se enfrenta a un contrincante, el cual debe enfrentar en la siguiente jornada al equipo marcado con el número siguiente, x+1, luego al equipo x+2 y así sucesivamente.

Es decir, consiste en elegir un equipo que se quedará fijo (en la Ilustración 3.13 es el número 1), mientras que todos los demás equipos se irán cambiando de posición.

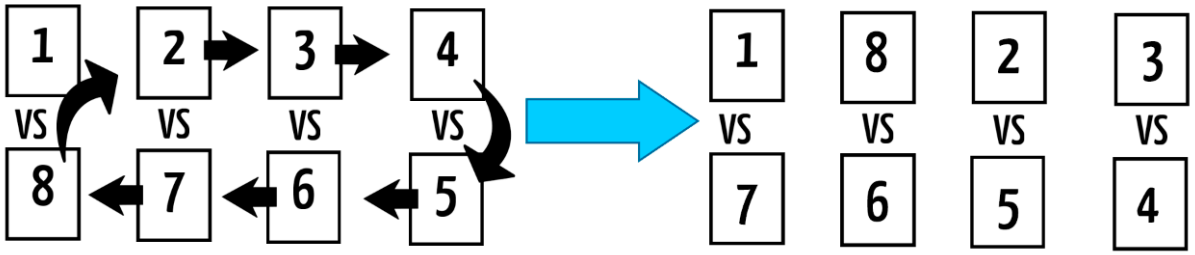

Ilustración 3.13: Esquema del funcionamiento del sistema "Round Robin". Fuente: Elaboración propia.

De esta forma todos los equipos se van enfrentando unos contra otros sin repetirse el enfrentamiento.

3.2. Los videojuegos en línea

Según la RAE, el término “en línea” significa: “Conectado a un sistema central a través de una red de comunicación” (Real Academia Española, 2021). Por lo tanto, un videojuego en línea transmite y accede a una información que es almacenada en un sistema. Entonces, un videojuego en línea no tiene necesidad de estar en conexión a otros usuarios, sino que estos se conectan a un servidor al que le envían o reciben información. Es por eso por lo que un videojuego en línea no tiene por qué estar en comunicación con usuarios que juegan online a la vez para denominarse así

3.2.1. Breve historia de los videojuegos online

Los videojuegos empezaron a viralizarse a partir del año 1972 con las máquinas recreativas que se encontraban en lugares públicos como bares, salones recreativos, aeropuertos… Esto ocurrió con los famosos juegos de Asteroids (Atari) o Space Invaders (Taito). Más tarde, aparecerían las consolas domésticas que buscaban que los usuarios jugarán desde su propia casa. Es en estos dispositivos donde más tarde se implementó el juego en línea, algo totalmente novedoso que permitía a los usuarios jugar con otras personas independientemente de la ubicación de estas (Belli, S., & Raventós, C. L., 2008).

Maestre (2019) explica que:

> "En los 90 y primera década del siglo XXI, con el desarrollo de Internet, se empezó a generar una serie nueva de videojuegos, en donde los jugadores podían interactuar con otros de otras partes del mundo. Incluso algunos de estos productos del entretenimiento eran solamente online".

Y finalmente, esto es lo que ha llevado la industria del videojuego a lo que existe en la actualidad generado esta conexión entre usuarios por un juego, ya sea desde las mismas partidas o en sitios externos como foros, chats, redes…

Para Adamus (2012):

> La evolución de los videojuegos técnica y narrativamente, así como la popularización de los juegos online multijugador con ayuda de Internet y las TIC, ha contribuido a la creación de una cultura del videojuego y de los jugadores sustentada en las comunidades surgidas de la actividad social, tanto dentro del propio juego, como fuera de él (con foros, chats, redes sociales, etc.).

Adamus dejó claro que los videojuegos online traen consigo mismos factores como sociabilidad, diversión, aprendizaje…

3.2.2. Introducción a las comunicaciones red

3.2.2.1. Servidor

Para iniciar cualquier juego online, se debe determinar con qué otras máquinas se debe conectar cada usuario. Para eso se necesita la creación y configuración de un servidor. Un servidor es una "unidad informática que proporciona diversos servicios a computadoras conectadas con ella a través de una red" (Real Academia Española, 2021). Esta definición es muy similar a la de "en línea" y no hace más que reforzar el significado de la definición ya vista. Por lo tanto, un servidor es el elemento que en la definición de "en línea" se llama sistema central.

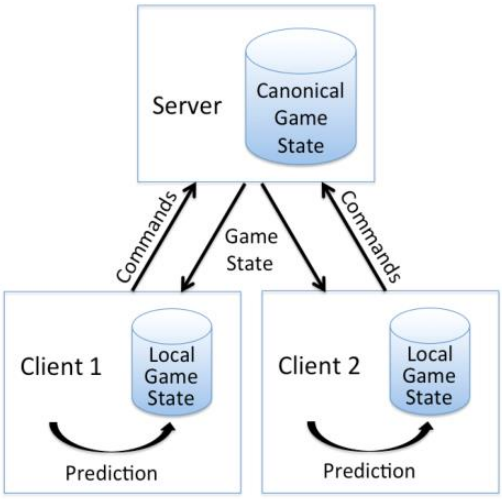

Ilustración 3.14: Interacción red entre usuario y servidor (T.C. Graham, 2018).

Así pues, la interacción del usuario con el servidor es como la Ilustración 3.14 donde el usuario manda una orden y el servidor la recibe, se actualiza el estado del juego y el usuario recibe el estado del juego actualizado.

3.2.2.2. Conceptos principales de las comunicaciones red

En los juegos online existen una serie de conceptos los cuales hacen referencia al rendimiento de la red. La mayoría de los juegos buscan entregar el mejor escenario red posible para que el usuario no noté ningún tipo de problema al jugar y no cause frustración.

3.2.2.3. Latencia

Como cuenta Pérez (2018), la latencia indica un desfase temporal entre dos puntos por los que pasa un paquete de datos y existen dos métodos para medirlo:

- El tiempo que transcurre desde que un paquete sale del nodo cliente hasta que alcanza el servidor.
- El tiempo que transcurre desde que un paquete de datos sale del cliente hacia el servidor y vuelve de nuevo al cliente.

El segundo método es el más común y se acostumbra a medir con el “ping” o respuesta de eco.

Por lo tanto, una latencia alta traerá una peor experiencia para el usuario, ya que tarda más tiempo en enviar y recibir paquetes de datos.

3.2.2.4. Ancho de banda

“Normalmente, cuando se habla de ancho de banda se está haciendo referencia a la banda de frecuencia utilizada por un canal para transmitir información” (Pérez, 2018).

Por lo tanto, este concepto define la capacidad máxima de datos que se puede transmitir a partir de una conexión en un momento determinado. Este normalmente se mide en bit/s.

3.2.2.5. Pérdida de paquetes

Según Pérez (2018), “este fenómeno ocurre cuando un paquete es enviado desde un nodo, pero nunca alcanza su destino”. Entonces, se puede decir que es una pérdida de datos que se produce porque este no llega a su receptor, refiriéndose este último al usuario o el servidor.

3.2.2.6. NAT

“El término NAT (Network Address Translation), es el proceso por el cual se traducen direcciones IP para extender el rango de direcciones disponible de una red” (Pérez, 2018). Por lo tanto, es muy útil para el intercambio de paquetes entre dos redes que tienen direcciones IP incompatibles entre ellas.

3.2.3. Tipos de implementación del modelo de red en juegos

Actualmente, existen diferentes formas de actuación del servidor. Las más importantes son la “Authoritative Server” y la “Non-Authoritative Server”. Estas dos tienen la finalidad de establecer una comunicación entre usuario y servidor, pero actúan de manera distinta. “Ambos modelos ofrecen privacidad para los usuarios finales, ya que, realmente, los clientes nunca conectan directamente entre ellos, ni llegan a revelar sus direcciones IP al resto de jugadores” (Pérez, 2018).

3.2.3.1. “Authoritative Server”

“En esta estructura, los jugadores envían peticiones y el servidor las recoge, actualiza el estado del mundo virtual e informa a cada jugador de los cambios de este. El cliente nunca realiza ningún cambio en el estado del juego” (Pérez, 2018). Por lo tanto, el servidor es el encargado de gestionar todos los cambios del estado del juego y significa que este tendrá una gran labor de procesamiento de información.

3.2.3.2. Non-Authoritative Server

“En este modelo, los clientes procesan localmente la lógica del juego y los inputs para cambios de estado, y envían los resultados al servidor. El servidor se limita a sincronizar todas las acciones con el estado del juego” (Pérez, 2018). Así pues, este modelo es lo contrario al anterior. Si antes todo era procesado por el servidor, ahora es el usuario quién lleva el procesamiento. El servidor solamente se encarga de hacer que todos los elementos coincidan en el tiempo.

3.3. Tecnología para el desarrollo del proyecto

3.3.1. Unity (Unity Technologies, 2005)

Unity (Unity Technologies, 2005) es una plataforma que permite la creación de videojuegos o experiencias interactivas tanto en 2D como 3D. Una de las funcionalidades que ofrece es implementar un modelo de conexión a la red para el juego (Pérez, 2018).

Ilustración 3.15: Logo de Unity (Unity Technologies, 2005)

3.3.2. Microsoft Azure PlayFab (Microsoft, 2014)

PlayFab (Microsoft, 2014) es una plataforma que proporciona un servidor y permite a los desarrolladores utilizar la nube para crear y operar juegos, analizar datos de juegos y mejorar las experiencias de juego en general.

Ilustración 3.16: Logo de PlayFab (Microsoft, 2014)

El servicio gratuito que ofrece es fácil de comprender y útil para almacenar los datos en el servidor. En el caso que el juego no superé los 100.000 usuarios o si se quiere

usar ciertas funciones se necesita pagar. Pero para el juego que se quiere desarrollar, las funcionalidades de PlayFab, que son de pago, no son necesarias.

Juegos como Minecraft (Mojang, 2011), Roblox (Roblox Corporation, 2006) o Tom Clancy's Rainbow Six Siege (Ubisoft, 2015) son ejecutados en PlayFab.

3.3.2.1. JSON: Serializar y deserializar

“JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo” (ECMA-404, 2017).

JSON está constituido por dos estructuras:

- Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla “hash”, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias (ECMA-404, 2017).

JSON se usa para almacenar datos en PlayFab*,* por lo tanto, es importante conocer como transformar estos datos para poder leerlos en Unity.

“La serialización y deserialización en JSON consiste en convertir objetos al formato JSON o viceversa. Esto suele ser útil cuando se interactúa con servicios web, o simplemente para empacar y desempacar datos a un formato de texto fácil” (Unity Technologies, 2022).

Es decir, en Unity, la serialización sirve para enviar los datos de PlayFab al servidor en el formato JSON, y la deserialización para transformar el paquete de datos que está en formato JSON a un objeto.

En PlayFab, todo el paquete de datos, se le llama “value*”*. Pero este necesita un nombre para poder acceder a él. A este nombre, en PlayFab, se le llama “key”. Es decir, que cada paquete de datos está formado por un nombre en clave y los datos. A este conjunto se le llama “key/value”.

3.3.2.2. Las “Requests” y las “Results”

Una “request” es un mensaje que se transmite del cliente al servidor y una “result” o “response”, es lo contrario, un mensaje que se transmite del servidor al cliente.

> El cuerpo de una “request” es el modelo de objeto que se envía como JSON al servicio API. Estos modelos contendrán propiedades que deben enviarse junto con su API como carga útil. Algunas propiedades se marcarán como obligatorias, mientras que otras son opcionales (Azure PlayFab Documentation, 2021).

Por lo tanto, cuando desde Unity se haga una “request”, este servirá para enviar información al servidor.

Después de una “request”, siempre hay una “result” porque este siempre tiene un mensaje que devolver.

> El servicio API devolverá un 200 OK con una carga JSON que se puede deserializar como modelo. Si la respuesta falla por algún motivo, se recibe una solicitud incorrecta 400. Esto podría significar que el servicio API está devolviendo información sobre una solicitud incorrecta. A menudo, es posible que falten parámetros en la llamada a la API o por otras razones (Azure PlayFab Documentation, 2021).

Es decir, si el “result” es correcto, se recibirá la información de la respuesta, pero si el “result” falla, se mandará un mensaje de error.

3.3.2.3. CloudScript

> CloudScript es una de las características más versátiles de PlayFab. Permite que el código del cliente solicite la ejecución de cualquier funcionalidad personalizada del lado del servidor que pueda implementar, y se puede usar junto con prácticamente cualquier cosa (Azure PlayFab Documentation, 2022).

Básicamente, este permite tener código, con el lenguaje de Javascript, en la nube del servidor y se puede ejecutar des de Unity. Esto puede dar más seguridad al proyecto, ya que se pueden crear funciones únicamente accesibles des del juego y evitan la posible manipulación por parte de los clientes.

3.3.2.4. Scheduled Tasks

“Las Scheduled Tasks son una forma de automatizar las rutinas comunes de operación del juego” (Azure PlayFab Documentation, 2022).

Algunos ejemplos son:

- Actualización de los datos del título para reflejar los cambios de un evento actual.
- Inyectando monedas virtuales en la economía del juego diariamente.
- Modificación de precios en una tienda según la hora del día.
- Llegar a los jugadores inactivos y alentarlos a volver a participar (Azure PlayFab Documentation, 2022).

Las Scheduled Tasks pueden ejecutar trozos de código almacenados en el CloudScript del servidor del juego cada cierto tiempo. La frecuencia con la que se ejecutan las funciones se escribe en PlayFab con una expresión Cron. Este es un programa del sistema operativo Unix (Unix, 1971).

> El formato cron de Unix se utiliza para especificar la hora en el parámetro planificación de los procedimientos ADMIN_TASK_ADD y ADMIN_TASK_UPDATE.
>
> El formato cron tiene cinco fechas de hora y fecha separados como mínimo por un espacio en blanco. No puede haber un espacio en blanco dentro de un valor de campo. Las tareas planificadas se ejecutan cuando los campos “minuto”, “hora” y “mes del año” coincidan con la fecha y hora actual, y como mínimo uno de los dos campos de día (mes del año, o día de la semana) coincidan con la fecha actual (IBM, 2015).

Resumidamente, esta es una expresión para definir cada cuando se ejecutará un trozo de código de CloudScript en PlayFab mediante las Scheduled Tasks.

3.3.2.5. Límites de PlayFab

En PlayFab, por el motivo de usar su versión gratuita, existen ciertas limitaciones a la hora de trabajar con ello. Estas son las más importantes y que pueden afectar en el desarrollo de este proyecto:

- El total de Scheduled Tasks que pueden existir al mismo tiempo son 25.
- El número total de Scheduled Tasks que se pueden ejecutar al mismo tiempo son 5.
- El número de pares “key/value” que se pueden actualizar en una sola “request” es 10.
- El tamaño de los datos que se almacenan en el servidor está limitado. El valor de esta limitación depende de donde se almacenan.

4. Referentes

Los referentes se van a clasificar dependiendo del que es vital remarcar y tener en cuenta para el proyecto. Para ello se ha hecho un análisis de juegos de mánager conocidos que puedan ayudar a elaborar el proyecto con sus ideas.

4.1. LaLiga Fantasy MARCA 2022 (La Liga Nacional de Fútbol Profesional, 2015).

El primer juego es LaLiga Fantasy MARCA (La Liga Nacional de Fútbol Profesional, 2015). Este juego está sincronizado con los sucesos jornada tras jornada de la primera división española de fútbol. Los usuarios tienen que formar una alineación cada jornada con los jugadores que pueden obtener mediante el mercado de transferencias. Los jugadores que en esa jornada formen parte del 11 inicial del usuario son los que puntúan dependiendo de la actuación de la vida real.

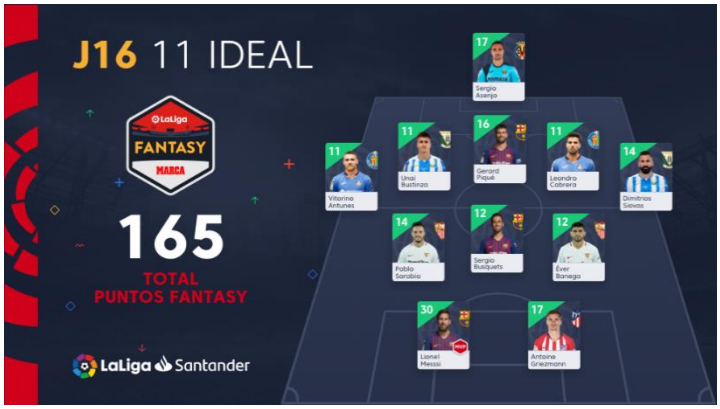

Ilustración 4.1: El 11 ideal con las mejores puntuaciones de la jornada 16 de la temporada 2018-2019 (La Liga Nacional de Fútbol Profesional, 2018).

Por un lado, en este juego cabe destacar su forma de puntuar la actuación de los jugadores. Son un total de 31 parámetros (goles, asistencias, tarjetas…) los que determinan el puntaje de cada jugador. A partir de cada parámetro, el juego le da un valor que se traslada en un número que se suma o resta a la puntación. Finalmente, la puntuación de todos los jugadores se suma y de ahí sale los puntos obtenidos en esa jornada.

El juego elige los factores que se deben de tener en cuenta para determinar la puntuación y como estos deben de afectar en el desarrollo de un partido. De aquí, también se puede aprender que, una estadística a la que es difícil justificar si es un valor positivo o negativo para el equipo, es mejor no tenerla en cuenta. Por ejemplo, la posesión del balón, que no aparece como parámetro puntuable del juego.

También es importante saberles dar un valor, ya que por ejemplo no se le puede dar la misma puntuación a un gol que una asistencia sin gol.

Por otro lado, el juego cuenta con muchas estadísticas de los partidos y las valoraciones del diario MARCA (Unidad Editorial, 1938). Esto le permite poder optar a una amplia variedad de parámetros para determinar los datos en comparación con este proyecto. Esto provoca que, en vez de trabajar con datos reales, lo cual sería muy costoso, se tenga elaborar una IA para simularlos. De hecho, las valoraciones que son reflejadas por las estadísticas de los jugadores son obtenidas por OPTA, la empresa de analítica deportiva mencionada en anteriores apartados.

Además, la liga saca mucho provecho de la aplicación gracias a esta mecánica que pretende buscar que los usuarios del juego estén más pendientes de las novedades y actualidades de la competición. Por lo tanto, se genera más vínculo entre los fans de la competición.

En el caso de este proyecto, la aplicación no trataría de buscar que los usuarios traten de ir a ver los partidos de su club, ya que el juego no va conectado con la realidad y no obtendrían una información que les diese más ventaja frente a los otros jugadores; sino que sería el juego el que daría una información sobre los jugadores del club.

Por el contrario, esto puede ocasionar un problema, ya que la información no tiene por qué concordar con la realidad, por ejemplo, en la vida real se lesiona un jugador que en el videojuego está perfectamente físicamente. Aunque esto también podría ser parte de la gracia del juego.

En cuanto al mercado, ahí se pueden encontrar los jugadores que el juego o los usuarios ponen a la venta. Aquí es donde sucede la interacción entre los usuarios que quieren pujar por los jugadores. Se puede pujar por un jugador hasta que se acabe el tiempo y el valor de las pujas no puede ser visto por el resto. Finalmente, se adueña del jugador el usuario que hizo la puja más alta.

Ilustración 4.2: Dos jugadores en el mercado de transferibles (La Liga Nacional de Fútbol Profesional, 2015).

4.2. Online Soccer Manager (Gamebasics, 2012) y Top Eleven (Nordeus, 2010)

Online Soccer Manager (Gamebasics BV, 2012) y Top Eleven (Nordeus, 2010) son dos videojuegos muy parecidos. Los dos consisten en gestionar un equipo de fútbol para después jugar contra otros usuarios en línea. La gestión se basa en controlar los fichajes, las tácticas, los entrenamientos, las finanzas, los eventos…

Los enfrentamientos entre usuarios son una de las mejores opciones de los dos juegos. Mientras el usuario ve el partido, puede tomar decisiones, cambiar la táctica, la formación o los jugadores en tiempo real. Estas decisiones pueden marcar la diferencia para tener mejores resultados.

Con respecto al proyecto, estos dos juegos son un ejemplo para tener en cuenta por cómo llevan a cabo la simulación de los partidos. Mientras que Top Eleven muestra el partido que se está simulando de una forma más compleja de programar, pero mucho más visual con una representación gráfica de cada acercamiento al área que exista durante el partido; Online Soccer Manager no lo enseña tan visual, pero lo sigue mostrando de una forma que funciona en el juego.

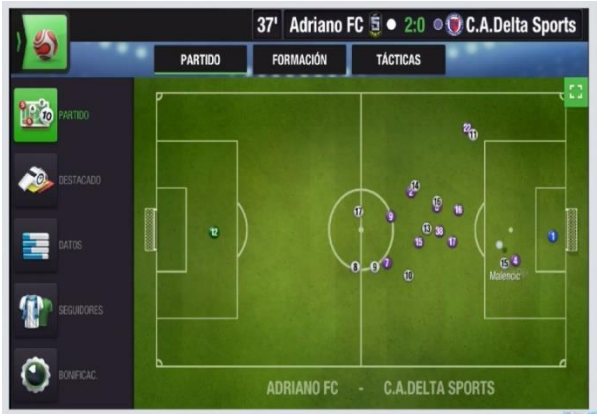

Ilustración 4.4: Partido simulado en Top Eleven (Nordeus, 2010)

Ilustración 4.3: Partido simulado en Online Soccer Manager (Gamebasics BV, 2012)

En general, entre los dos juegos, estos son los factores que se ha encontrado que influyan en la simulación de los partidos:

- Las características de los jugadores como su calidad, su moral, su condición física, su posición en el campo, si sufre lesiones y su edad (que influiría en la condición física). Estas características pueden variar en el transcurso del juego, sobre todo en los partidos, si se aplican entrenamientos a los jugadores o se mejora la calidad de la plantilla mediante fichajes.

Ilustración 4.5: Características de los jugadores en Top Eleven (Nordeus, 2010)

- El rival también influye en el desarrollo de los partidos. Por ejemplo, para el usuario es más fácil ganar a un rival con una plantilla de más baja calidad que uno de calidad alta.
- Las tácticas que se pueden ejecutar antes o durante los partidos. El usuario tiene la posibilidad de sustituir jugadores, cambiar sus posiciones, la formación, aumentar la presión, cambiar el tipo de pases, forzar contraataques, introducir marcajes… Estos cambios pueden determinar el desarrollo del partido. En el caso de Online Soccer Manager, solo es posible hacer estos cambios antes del partido.

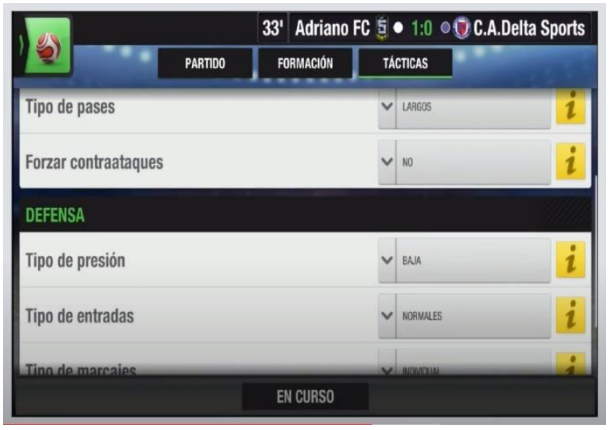

Ilustración 4.6: Tácticas modificables en Top Eleven durante el partido (Nordeus, 2010).

Con respecto al proyecto, van a ser importantes los tres tipos de factores para determinar los sucesos del partido. Para elaborar la IA y realizar la simulación se deberá tomar una decisión de cómo van a afectar todos los factores.

4.3. Handball Manager 2021 (netmin games, 2021)

Handball Manager 2021 (netmin games, 2021) es un juego similar a los dos anteriores, pero sin el factor online. El usuario tiene total control sobre todos los aspectos deportivos y comerciales del club utilizando una infinita variedad de funciones de entrenamiento, tácticas, traspasos, estadio, apoyo de la afición, cantera, merchandising…

Ilustración 4.7: Menú del Handball Manager 2021 (netmin games, 2021)

A pesar de no ser un juego tan relevante a nivel mediático y contar con un presupuesto diferente a los anteriores, cuenta con una similitud muy relevante en comparación al proyecto que es el balonmano. Conviene señalar de este juego todas las características diferentes por ser un juego que trata un deporte diferente. Sobre todo, las tácticas que puede modificar el usuario y que no se pueden observar en los videojuegos de fútbol.

5. Diseño metodológico y cronograma

En este apartado se explica el diseño metodológico que se seguirá para desarrollar el trabajo y su planificación.

5.1. Fases del proyecto

El desarrollo del proyecto se ha dividido en siete fases las cuales harán más fácil el desenvolvimiento y la organización de este trabajo.

5.1.1. Investigación

La primera fase tratará de una investigación sobre el marco del trabajo. Se adentrará en indagar en todos los conceptos de red y de juegos de mánager que abarcan el juego en cuestión. Este servirá para tener una base de conocimientos que ayudará a cumplir con los objetivos del trabajo.

5.1.2. Busca de referentes

En este apartado se buscan juegos parecidos y así descubrir técnicas o ideas útiles que se usen para luego transportarlas al juego. Sirven todo tipo de juegos de mánager en los que se pueda observar el funcionamiento de la inteligencia artificial que se encarga de la simulación de los partidos y también de los que se pueda aprender acerca de sus características.

5.1.3. Diseño del juego

Aunque este trabajo sea de programación, no significa que no se tenga que diseñar. En este apartado, después de haber investigado conceptos y juegos referentes, se definirá como será el juego. También se tratará de idear la inteligencia artificial que será también importante para el juego.

5.1.4. Desarrollo completo del videojuego

En esta fase, se desarrollará el juego junto todo su apartado red. Al terminar esta fase el juego debe de poder conectarse con el servidor y transmitir y recibir información del estado del juego con el resto de los usuarios. También debe de funcionar a la perfección la IA desarrollada haciendo correctamente lo planteado en la fase del apartado 5.1.3. Solo faltaría lo referente al apartado 5.1.5.

5.1.5. Pulido del juego

El objetivo de esta fase es dejar un período de tiempo lo más amplio posible que sirva para corregir errores del juego y mejorar su visualización. Una vez terminada esta fase, el juego debería ya estar listo para que sea jugado por cualquier persona.

5.1.6. Conclusiones

Por último, el objetivo de esta fase es reflexionar sobre el resultado del trabajo, es decir, un equivalente a un post mortem. Con este, se debería concluir si se han cumplido los objetivos y determinar la satisfacción del resultado.

5.2. Cronograma

Teniendo en cuenta el calendario académico y el tiempo que se dispone se ha realizado la tabla que se observa en Ilustración1 con tal de planificar las fases descritas anteriormente. Por cada fase se ha marcado los meses en los que se llevará a cabo el trabajo de ella. Las líneas rojas marcan los plazos de entrega del trabajo.

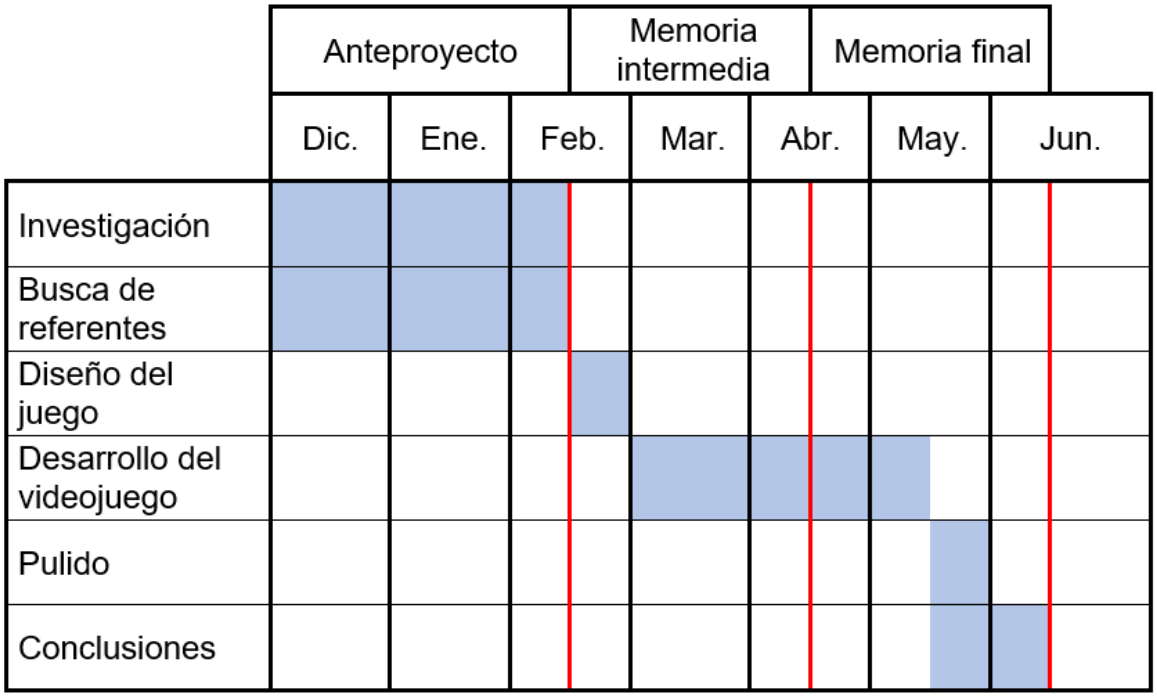

Ilustración 5.1: Cronograma de las fases del proyecto. Fuente: Elaboración propia.

6. Desarrollo y resultados

Este capítulo contiene toda la información relacionada con el proceso y desarrollo del proyecto, mostrando así también los resultados.

6.1. Preparación

Para empezar, se ha iniciado un proyecto en Unity con la versión 2020.3.4f1. No existe ningún tipo de motivo por el cual se haya elegido esta versión, simplemente se ha escogido una con la que el autor del proyecto se sienta más cómodo.

También, se ha necesitado el registro de una cuenta en PlayFab, la instalación de los "packages" para poder usar la API en Unity y el inicio de sesión de PlayFab en Unity. Finalmente, se ha creado un nuevo estudio en la web de PlayFab donde se almacenarán los datos del juego.

6.1.1. "Assets"

En cuanto al apartado artístico del proyecto, exceptuando el fondo y la tipografía, se usarán "assets" que fueron elaborados de forma propia para otras asignaturas de la carrera. En la Ilustración 6.1: "Assets" usados en el proyecto. Fuente: Elaboración propia.Ilustración 6.1 se pueden observar todos ellos.

Ilustración 6.1: “Assets” usados en el proyecto. Fuente: Elaboración propia.

Ilustración 6.2: La tipografia usada en el proyecto: Yanone Kaffeesatz (Matas & del Peral, 2004)

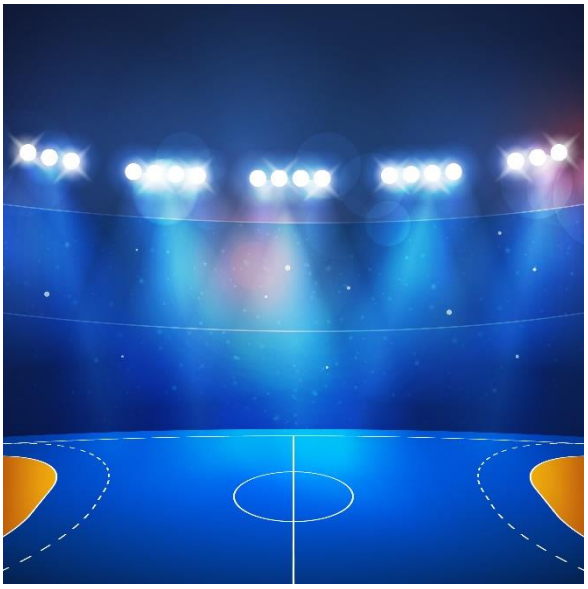

Ilustración 6.3: Fondo usado en el proyecto (Freepik.com, 2010)

6.2. Menú de “login” y registro del usuario

Una vez hecha la preparación, el videojuego ya tiene un servidor y ya se puede proceder a crear una interfaz con la que el usuario pueda crear una cuenta o hacer el “login”. Y así pues todo ello almacenarlo en la base de datos del servidor. Los objetos que deberá tener esta primera interfaz son:

- “Input field” del nombre del usuario*:* Sitio donde el usuario pueda escribir su nombre de usuario.
- “Input field” de la contraseña: Sitio donde el usuario pueda escribir su contraseña para el usuario.
- Botón de “login”
- Botón de registro y “login” a la vez
- Texto de error: Texto que servirá para informar al jugador si hay algún tipo de error.

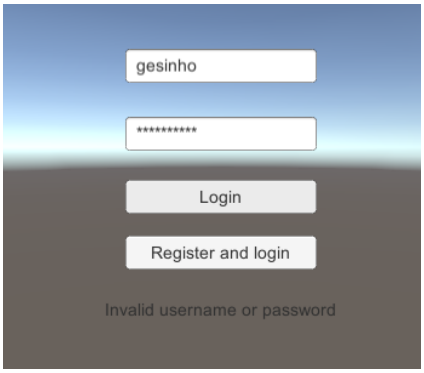

Ilustración 6.4: Interfaz de “login” y registro. Fuente: Elaboración propia

Con esto, está hecha la primera interfaz del juego, ahora falta enviar la información recibida al servidor y que esta permita el “login” o el registro del usuario.

Para registrar al usuario en el servidor se necesita crear la clase *PlayfabManager*, donde se crearán las funciones que se ejecutarán cuando se pulsen los dos botones.

Respecto al registro, se crean tres funciones:

- *OnRegisterSuccess* recibe una variable que será el resultado de la información que se envía al servidor cuando se consigue registrar el usuario. Cuando esta función sea ejecutada, significa que el usuario ha sido registrado en el servidor.
- *OnError* esta función se llamará en caso de que haya un error al hacer la “request”. Contiene un atributo, que es el mensaje del error, y se puede usar para enseñarlo al usuario.
- *RegisterButton* se ejecuta cuando se pulsa el botón. Esta recoge los *strings* del nombre del usuario y la contraseña. Seguidamente, comprueba que la contraseña tenga 6 caracteres o más, ya que PlayFab no acepta contraseñas más cortas. Después crea una variable de tipo *var* y será un objeto nuevo de tipo *RegisterPlayFabUserRequest* donde se guardan los atributos para el

registro (nombre de usuario y contraseña en este caso). Finalmente, se llama a la función *RegisterPlayFabUser* de la clase *PlayFabClientAPI,* se le pasa el objeto creado anteriormente y las dos funciones mencionadas anteriormente (*OnRegisterSuccess* y *OnError*).

Para el botón de “login”, se hace exactamente lo mismo. Se hace una “request” donde se envía el nombre de usuario y contraseña al servidor: se llama a la función *LoginWithPlayFab,* se le pasa la información y las funciones de *OnLoginSuccess* y *OnError.* Para la función OnError, no es necesario crear una de nueva para el *login* porque este siempre recibe el error en cuestión.

6.3. Menú de acceder o crear una liga

Una vez el usuario se ha registrado o ha accedido con una cuenta existente, se abrirá una nueva ventana donde tendrá las siguientes opciones: “Logout”, crear una nueva liga y acceder a una liga donde él ya forma parte. Esta ventana también está controlada por la script *PlayfabManager.*

6.3.1. “Logout”

Para esta opción, es tan sencillo como crear un botón que desactive la ventana actual y active la ventana anterior donde el jugador podía registrarse. Una vez lo haga, PlayFab detecta la nueva cuenta con la que este está iniciando sesión y cierra la que estaba usando.

6.3.2. Crear una nueva liga

Para crear una nueva liga, se ha creado un “Input field” para que el usuario escriba el nombre de esta y un botón para crearla. Cuando el botón es pulsado, se hace una “request” al servidor llamando la función *CreateGroup* de la *PlayfabGroupsAPI* que crea un grupo. Esto más adelante permitirá acceder a todas las ligas en las que el usuario forma parte. Además, también se hace otra “request” donde se llama a la función *CreateSharedGroupData* de la *PlayfabClientAPI*, este crea un grupo que permite almacenar datos.

A estas dos funciones se les pasa el parámetro del nombre del grupo con el que más adelante se podrá acceder a la información del grupo a partir de él.

6.3.3. Acceder a una liga donde el usuario ya forma parte

Para esta opción, se ha creado en la pantalla un “scroll view” en la interfaz para mostrar al usuario los ligas en las que él forma parte. El usuario podrá interactuar con este objeto para verlas.

De este modo, después que el usuario haga el “login”, se ha hace una “request” a la función *ListMembership* de la *PlayfabGroupAPI*. Esta permite acceder a una lista donde están todos los grupos del usuario. A continuación, por cada grupo que hay, se instancia un objeto prefabricado que tiene un texto con el nombre del grupo y se le hace hijo del objeto *Content* que también es hijo del “scroll view”. De esta forma, todos estos objetos se situarán dentro de *Content*.

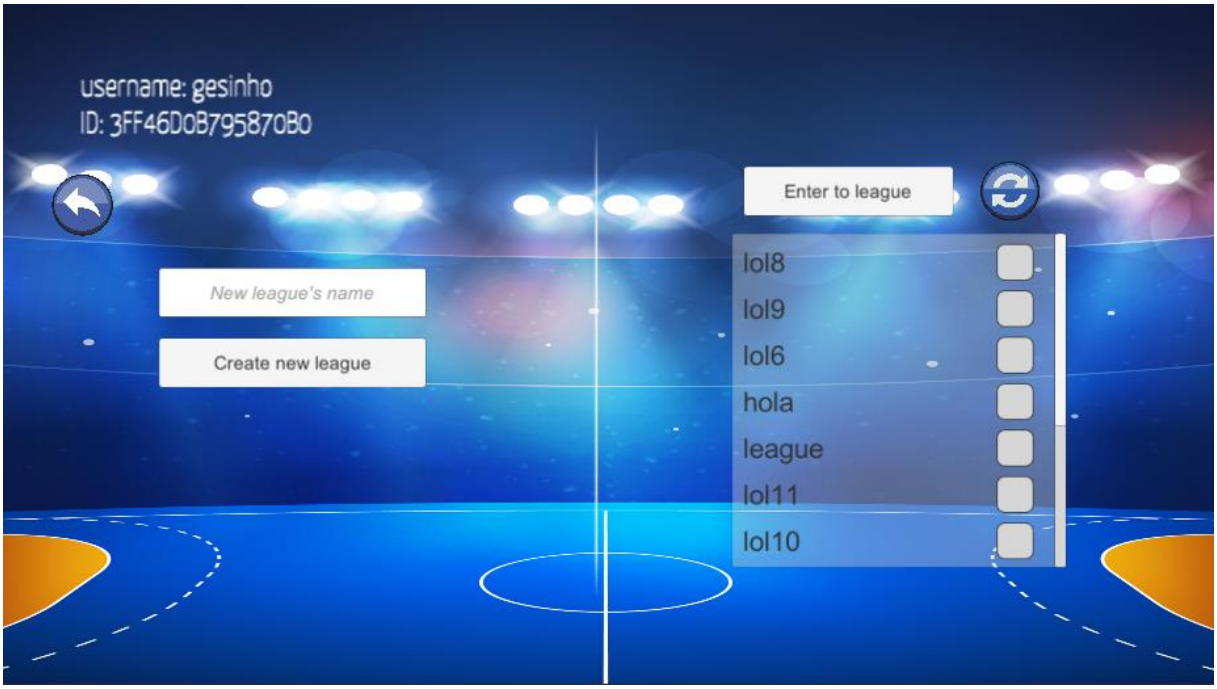

Ilustración 6.5: Menú para entrar en la liga. Fuente: Elaboración propia

En la parte derecha de la Ilustración 6.5 se puede observar cómo se vería este “scroll view”. Cada objeto prefabricado contiene una casilla para que el usuario marque el grupo donde quiere entrar. El objeto *Content* tiene el componente “Toggle Group” que hace que solo pueda estar marcada una casilla de todas las que hay.

Cuando el usuario pulse el botón “Enter to league”, se revisa cuál de todos los objetos es el que está marcado y se guarda en una variable el texto que contiene. Si no hay ninguna liga seleccionada, se muestra un mensaje de error.

6.4. Base de datos

Antes de seguir con las siguientes ventanas, es necesario almacenar todos los datos de los jugadores con los que los usuarios van a jugar y también todas las reglas que definen el juego.

En el servidor del juego PlayFab hay un apartado llamado “Title Data”, allí se pueden almacenar todos los datos principales que serán iguales para todos los usuarios. En este sitio se almacenarán todos los jugadores del club (un identificador, el nombre, la categoría, la posición, la calidad y el precio inicial). Para ello, como hay muchos (164), primero se ha creado un Excel y luego se ha usado un convertidor para transformarlo en el formato que usa PlayFab, el formato JSON.

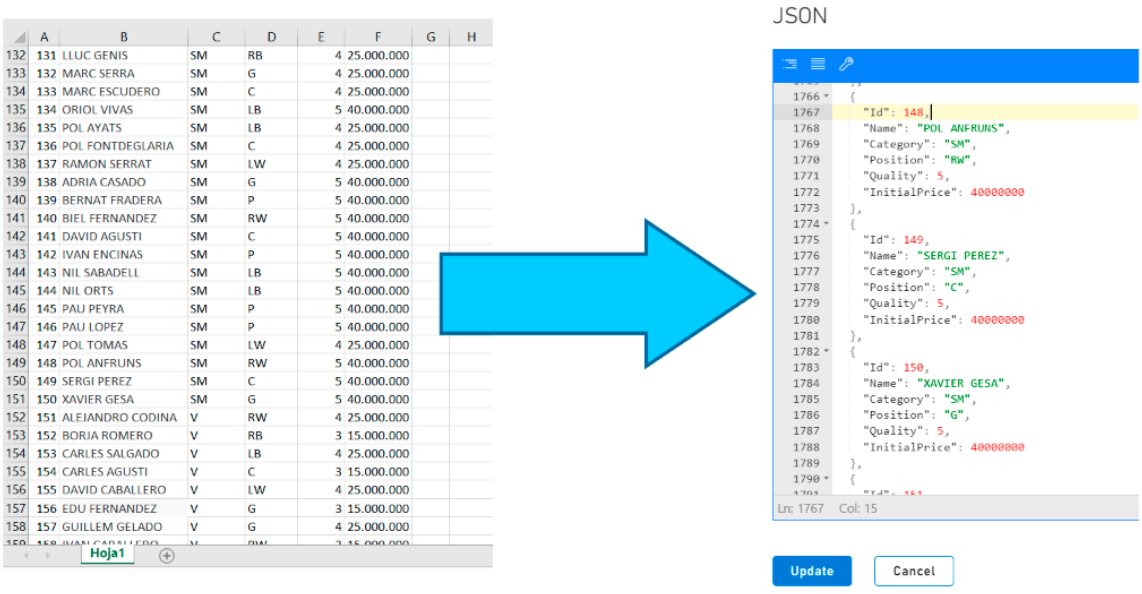

Ilustración 6.6: Conversión de la base de datos de los jugadores en Excel a formato JSON. Fuente: Elaboración propia .

A continuación, allí también se han almacenado las reglas, es decir, el dinero con el que empieza el usuario, algunos parámetros para la creación de cada plantilla, la hora en la que se actualiza el mercado y se simulan los partidos, el máximo de jugadores

en una plantilla, el mínimo y máximo de miembros que puede haber en una liga y el número de rondas que habrá.

En el juego, una vez el usuario haya hecho el “login” o se haya registrado, se hace una “request” para obtener esta información que se encuentra en el “Title Data” del juego. Para almacenar los jugadores y las reglas en Unity, se ha creado la clase *HandballPlayer* y la clase *Rules* que contienen las mismas variables mencionadas anteriormente. Se deserializa toda la secuencia de caracteres que están en formato JSON a una lista del objeto *HandballPlayer* y también se hace lo mismo con las reglas. En la Ilustración 6.7 se puede observar cómo se ve en el código la clase *HandballPlayer*. De esta forma, el juego, una vez se haya iniciado sesión, ya tiene todo el “Title Data” almacenado en dos listas diferentes, una de jugadores y otra de reglas.

```
public class HandballPlayer
{
    public int playerID;
    public string playerName;
    public string category;
    public string position;
    public int quality;
    public int initialPrice;

    public HandballPlayer(int id, string name, string category,
string position, int quality, int initialPrice)
    {
        this.playerID = id;
        this.playerName = name;
        this.category = category;
        this.position = position;
        this.quality = quality;
        this.initialPrice = initialPrice;
    }
}
```

Ilustración 6.7: La clase *HandballPlayer*. Fuente: Elaboración propia

6.5. Creación de los datos de la liga

Una vez el usuario ha creado un grupo, en el *SharedGroupData* de este se quiere almacenar la siguiente información: la plantilla, el resto de los jugadores que no son propiedad de ningún equipo, el dinero del usuario, el estado de la liga (si ha empezado o no, en qué jornada están, cuántas jornadas tiene en total), la clasificación y el *host*.

Para enviar al servidor toda esta información se llama a la función *UpdateSharedGroupData* de la *PlayfabClientAPI*. Esta necesita como parámetros el nombre del grupo y todos los datos que se quieren almacenar. Los datos son una variable *Dictionary<string, string>*. El primer *string*, llamado "key", es el nombre con el que se identifica el paquete de datos que está en formato JSON y el segundo *string,* llamado "value", son los datos en formato JSON. El conjunto de los dos *string* se llaman "Key/Value".

```
var data = new Dictionary<string, string>();

data = new Dictionary<string, string> {
      { playerName + "LineUp", JsonConvert.SerializeObject(team) },
      {"RestPlayers",  JsonConvert.SerializeObject(restPlayers) },
      {playerName + "Money", rules.startPlayerMoney.ToString() },
      {"State", JsonConvert.SerializeObject(new LeagueState("NoStarted",
0, 0)) },
      {"Classification",  JsonConvert.SerializeObject(classification) },
      {"Host", playerName }
};
```

Ilustración 6.8: Los datos enviados al SharedGroupData cuando es creado. Fuente: Elaboración propia

En total se envían seis parejas "Key/Value":

1- *team* es una lista de *TeamHandballPlayerSharedGroup.* Esta es una clase que almacena el identificador del jugador y su posición.
2- *restPlayers* es una lista de *integers*. Es una lista de todos los identificadores de los jugadores que no forman parte de ningún equipo.

El proceso para determinar los valores de la lista *restPlayers* y *team* se explica en el apartado 6.5.1.

3- El dinero del usuario no hace falta serializarlo a formato JSON, simplemente se convierte en *string*.
4- El estado del juego es un objeto de la clase *LeagueState* que tiene las tres variables nombradas anteriormente: el estado, la jornada actual y el número de jornadas que tiene la liga.
5- *classification* es una lista de *LeagueTeam*. Esta es una clase que almacena el nombre del usuario y sus puntos.
6- *Host* es el nombre del usuario que ha creado la liga.

Cuando se añade un miembro, que no sea el primero en unirse a la liga, el “Key/Value” del *host* no se manda al servidor y, previamente al *UpdateSharedGroupData*, se hace una llamada a la función *GetSharedGroupData* para obtener los datos que se tienen que actualizar una vez añadido este nuevo miembro.

6.5.1. Creación de la plantilla del usuario

En este apartado se explica cómo se genera la plantilla de un usuario. El objetivo es obtener una lista de diez *TeamHandballPlayerSharedGroup* para definir el equipo del usuario y otra de *integers* que define los jugadores que no son propiedad de ningún usuario. Para ello, antes que nada, se ha creado un script llamado *PartyDesigner* que va a servir para llevar a cabo todo el proceso. Seguidamente, para enviar los datos de este nuevo usuario que formará parte de la liga, hay que diferenciar si este es el primero en unirse o no. En caso de ser afirmativo, se usa la lista de *HandballPlayer* obtenida del “Title Data” para generar la plantilla.

En el caso de no ser así, primero se comprueba que no se supere el límite máximo de usuarios en la liga y después, se llama a la función *GetSharedGroupData* para obtener la clasificación y la lista de *restPlayers* (los jugadores que no son propiedad de ningún equipo). En vez de usar la lista *HandballPlayer*, se creará una nueva con todos los *HandballPlayer* que tengan su identificador en la lista de *restPlayers*.

Una vez definida la lista de *HandballPlayer*, esta se divide en 5 listas según el valor de la calidad que tenga cada jugador, ya que uno de ellos puede tener entre 1 y 5 de calidad:

- En la lista de los que el valor de calidad es 1, se añaden cinco jugadores aleatorios a la plantilla.
- En la lista de los que el valor de calidad es 2, se añade un jugador aleatorio a la plantilla.
- En la lista de los que el valor de calidad es 3, se añade un jugador aleatorio a la plantilla.
- En la lista de los que el valor de calidad es 4, se añaden dos jugadores aleatorios a la plantilla.

- En la lista de los que el valor de calidad es 5, se añade un jugador aleatorio a la plantilla.

El número de jugadores que se escogen de cada lista para generar la plantilla del usuario se define en las *rules* del “Title Data” del servidor de PlayFab.

Al añadirse el jugador en la plantilla, también se elimina de la lista de *restPlayers*.

Antes de enviar la plantilla al servidor, hay que evitar que haya dos jugadores que tengan la misma posición. En el balonmano existen 7 posiciones de campo y en el código se han añadido 6 más para los jugadores que estan en el banquillo. Estos se guardan con la siguiente nomenclatura en el servidor del juego: “C”, “LB”, “RB”, “LW”, “RW”, “P”, “G”, “S1”, “S2”, “S3”, “S4”, “S5”, “S6”. Entonces, si hay un jugador cuya posición está repetida por otro, se le pone en una posición del banquillo que no esté ocupada; y si el banquillo está ocupado al completo, se le pone en una posición del campo que no esté ocupada.

Así pues, se envían los nuevos “Key/Value” del dinero del usuario, su plantilla, la clasificación y la lista de jugadores que no son propiedad de ningún usuario al *SharedGroupData*.

6.5.2. Inicialización del mercado de jugadores

Por último, a la liga le falta un mercado de jugadores. Este mercado se creará en caso de que el usuario que se está añadiendo al grupo sea el primero en unirse. Esto se hará con una función dentro del CloudScript de PlayFab porque hay parámetros del mercado que se deben poder modificar en un futuro desde el servidor si es necesario. Además, la actualización de los jugadores que saldrán en el mercado se hará de forma diaria desde CloudScript y conviene tenerlo todo en el mismo “script”.

Así que, una vez el usuario ha creado una liga, se llama a la función *ExecuteCloudScript* de la *PlayfabClientAPI*. Esta llama a la función *SetMarketForFirstTime* que se ha creado dentro de CloudScript y se le pasa el nombre del grupo donde tiene que crear el mercado.

En *SetMarketForFirstTime* se hace una “request” al grupo para obtener la lista de *restPlayers*. A partir de esta lista, se consiguen cinco jugadores aleatorios que serán los que se eliminarán de esta lista y formarán parte de la lista del mercado.

Finalmente, se actualiza el *SharedGroupData* con las “key/value” del mercado y de *restPlayers*.

La lista del mercado es una lista de *HandballPlayerMarketSharedGroup* que es un objeto que tiene como variables el identificador del jugador (*integer*), una lista de los jugadores que pujan sobre este (*List<Bid>*) y el propietario del jugador (*string*). *Bid* es un objeto que tiene como variables el nombre del usuario que hace la puja (*string*) y la cantidad de la puja (*integer*).

6.6. Menú previo al inicio de la liga

Después de que el usuario haya creado una liga o haya entrado en uno ya existente se hace una *request* para comprobar si el estado de la liga es “NoStarted”, es decir, no ha empezado. La gestión de este menú se lleva a cabo en un nuevo *script* llamado *PartyMenuManager.*

Si no está empezada, se abre un menú donde el jugador puede añadir miembros a la liga o puede iniciarla. Si el estado de la liga es “Started”, es decir, ha sido iniciada, se abrirá otro menú que se explica en el apartado 6.8.

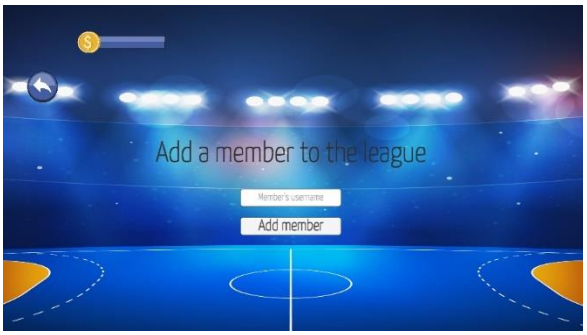

Ilustración 6.9: Menú para añadir miembros a la liga. Fuente: Elaboración propia.

Ilustración 6.10: Menú que se abre cuando el estado de la liga es "NoStarted". Fuente: Elaboración propia.

Cuando el usuario añade un nuevo miembro a la liga se hacen tres “requests”:

1- La primera llama a la función *GetAccountInfo* de la *PlayfabClientAPI* para obtener el valor del *PlayfabID* del usuario a partir de su nombre.
2- Seguidamente, se llama a la función *AddSharedGroupMembers* para añadirlo al *SharedGroupData*.
3- Por último, se le añade también al grupo usando la clase *PlayfabGroupsAPI*.

Una vez el usuario haya añadido a todos los miembros, ya puede iniciar la liga. Solo podrá iniciarla en caso de que haya entre 4 y 8 miembros en el grupo.

6.6.1. Iniciar la liga

El objetivo de iniciar la liga es crear un calendario de partidos donde se enviará al servidor tantas parejas “key/value” como tantas jornadas tenga el calendario. Para ello, se vuelve al *script* de *PartyDesigner* cuya función es generar la liga. Allí se hace una “request” para obtener el valor de la clasificación que permitirá acceder a la lista de miembros.

Para empezar, se crea un nuevo *LeagueState* donde el estado es “Started”, empieza por la jornada uno y tiene un número definido de jornadas. Este último se define, como se ve en la Ilustración 6.11, por el número de miembros menos uno y multiplicado por el número de vueltas (si el número de miembros es par) o por el número de miembros multiplicado por el número de vueltas (si el número de miembros es impar).

```
if (classification.Count % 2 == 0) //Par
{
      numberOfDays = (classification.Count - 1) * rules.numberOfRounds;
}
else //Impar
{
      numberOfDays = (classification.Count) * rules.numberOfRounds;
}
```

Ilustración 6.11: Cálculo del número de jornades de la liga. Fuente: Elaboración pròpia.

A continuación, si el número de miembros es impar, se añade un nuevo *LeagueTeam* a la clasificación llamado “Break” para que el equipo que le toque jugar contra él en esa jornada sea el equipo que descanse.

Seguidamente, se deben determinar los enfrentamientos de cada jornada. Para ello, se utiliza el método “Round Robin” que se puede observar en la Ilustración 6.12. Todos los equipos rotan la posición descrita en el esquema excepto el 1 que se queda fijo. De esta forma se consigue que nunca se repitan los partidos en una vuelta.

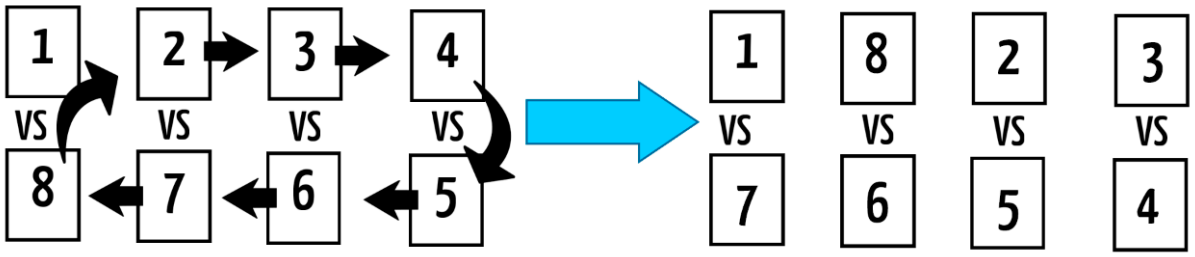

Ilustración 6.12: Esquema de la elaboración del calendario de partidos usando el sistema “Round Robin”. Fuente: Elaboración propia.

Cada jornada está formada por una lista de *Match*. Este objeto contiene las variables del nombre del equipo local y el del visitante, el número de goles del local y el del visitante, y el número de puntos del equipo local y el del visitante.

Finalmente, por cada jornada se envía al *SharedGroupData* una pareja “key/value” con la lista de partidos de esta.

6.7. Programación de las Scheduled Tasks

Una vez se ha iniciado la liga, se necesita tener un fragmento de código que se ejecute cada día a una hora concreta. Para ello, se usan las Scheduled Tasks que ejecutan una función que este en el CloudScript del servidor de Playfab cada cierto tiempo. De modo que, una vez iniciada la liga, también se llama la función *CreateCloudScriptTask* de la clase *PlayFabAdminAPI*. Esta recibe el nombre de la tarea, el parámetro del nombre del grupo, el nombre de la función de CloudScript que debe llamar y cada cuanto tiempo se ejecutará (en formato *Cron*).

6.7.1. Simulación de partidos

La Scheduled Task de la liga en cuestión ejecuta la función que se encuentra en CloudScript llamada *MatchSimulation*. Esta hace una “request” para obtener la lista de jugadores del Title Data, y otra para obtener el estado de la liga y la clasificación. Cuando se obtiene la jornada actual de la liga, se hace otra “request” para tener la lista de partidos que se juegan esa jornada y la lista de todas las plantillas de todos los equipos.

Por cada partido de la jornada se hace una simulación, exceptuando el partido en que el local o el visitante se llame “Break” porque significa que le toca descansar al usuario en cuestión.

Según las estadísticas que se mostraron en Ilustración 3.12, se podía observar que los partidos de balonmano de la Copa del Rey había unas 58 posesiones en total de media por partido con una desviación típica de 4,5. Esto se ha usado para que, antes de simular el partido, elegir un número aleatorio entre 53 y el 63. Este número serán las posesiones que habrá en el partido.

Por cada posesión, si este es impar atacará el local, si es par atacará el visitante. Seguidamente, en esa posesión se elige una de las seis posiciones de campo (sin contar el portero) y se produce un enfrentamiento entre los dos jugadores de cada equipo. El que le toca defender cuenta con la ayuda del portero en ese enfrentamiento. Los jugadores del banquillo no juegan el partido.

Este enfrentamiento consiste en comparar la calidad del jugador o jugadores locales y la del jugador o jugadores visitantes. Como se puede ver en Ilustración 6.13, esas son las fórmulas para la elegir la calidad local y la visitante. Aunque se tienen en cuenta dos excepciones importantes:

- Si la posición del jugador que está en el enfrentamiento no es la misma que su posición habitual, se le resta un punto a la calidad del jugador.
- Si la posición habitual del portero que está en el enfrentamiento no coincide con la de portero, su calidad será 1.

Ilustración 6.13: Esquema para elegir la calidad de cada equipo. Fuente: Elaboración propia.

A continuación, se suman la calidad del local y la del visitante y se genera un número aleatorio. Si este está entre el 1 y el total de la calidad local, este último gana el enfrentamiento. Si está entre el total de la calidad local y la del visitante, este último gana el enfrentamiento.

Una vez definido el ganador del enfrentamiento se debe asignar los puntos a cada jugador y el gol (en caso de que lo haya):

- Si el jugador que atacaba gana, recibe 4 puntos y se suma un gol al equipo local.
- Si el jugador que atacaba pierde, se le resta 1 punto.
- Si el jugador que defendía gana, recibe un punto.
- Si el jugador que defendía pierde, se le resta un punto.
- Si el portero gana, recibe dos puntos.
- Si el portero pierde, se le resta un punto.

También, a cada usuario se le suma dinero dependiendo de los puntos totales que haya hecho en esa jornada. Al dinero se le suma el número de puntos multiplicado por una variable *int* que es el multiplicador de dinero.

Finalmente, se actualiza llamando la función *UpdateSharedGroupData* y enviando la clasificación actualizada, el nuevo estado de la liga, el dinero de cada jugador y una lista de *integers* donde el índice es el identificador del jugador y su valor son los puntos que ha generado en su último partido.

6.7.2. Actualización del mercado

Al terminar la simulación de los partidos, se llama a otra función que se encargue de actualizar el mercado. Allí se hace una “request” para obtener la siguiente información: la lista de *restPlayers*, el mercado, la lista *assignPlayers,* que es una lista de *string* donde el índice es el identificador del jugador y el valor es el nombre del usuario que se le debe de asignar al jugador a la plantilla; y *returnBids*, una lista de *Bid* que contiene la cantidad de dinero que se le debe retornar al usuario y el nombre del usuario.

Primero de todo, se revisa la lista del mercado. Se introduce a la lista de *assignPlayers*, en el índice del identificador del jugador, el nombre del usuario de la puja más alta. Si no tiene ninguna puja se vuelve a añadir a la lista de *restPlayers*. También, se devuelve el dinero a los usuarios que no ganaron la puja en la lista *returnBids.*

A continuación, se crea el nuevo mercado con cinco jugadores aleatorios de la lista *restPlayers*. Y finalmente, se actualizan las cuatro listas del *SharedGroupData* que se obtuvieron en la “request”.

6.8. Menú de la liga

El menú de la liga es la ventana que se abre cuando su estado es “Started”, es decir, ha sido iniciada. Está controlado por el *script PartyMenuManager*. Esta tiene cinco botones: el de volver atrás, el que muestra la información del último partido, el que permite cambiar la alineación, el del mercado y el de la clasificación. También muestra la siguiente información: el dinero del usuario, el nombre de la liga, el siguiente partido, el número de la jornada y la hora que se simula el partido. Todo ello se puede observar en la Ilustración 6.14.

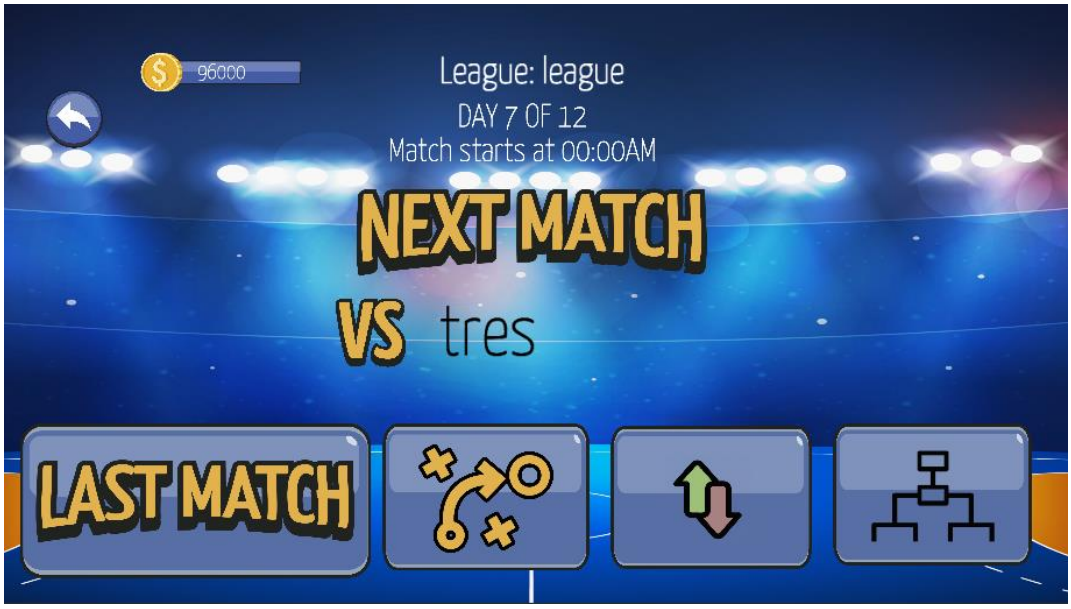

Ilustración 6.14: Menú de la liga. Fuente: Elaboración propia

Al abrirse, primero de todo comprueba si el usuario tiene algún jugador que haya comprado del mercado de la lista *playersToAssign* y que deba ser añadido en su plantilla. Seguidamente, comprueba si al usuario se le tiene que devolver el dinero de alguna puja que no haya ganado de la lista *returnBids*. Y finalmente, muestra toda la información obtenida mediante la llamada de la función *GetSharedGroupData* de la clase *PlayfabClientAPI.*

6.8.1. Menú del último partido

Este menú es una ventana que muestra toda la información del último partido que ha jugado el equipo del usuario, es decir, el resultado, los puntos de cada equipo y los puntos que ha obtenido cada jugador. Se puede observar el menú en la Ilustración 6.15.

Ilustración 6.15: Menú del último partido. Fuente: Elaboración propia.

Todo este menú se gestiona des de un nuevo “script” que tiene el nombre de *LastMatch*. Cuando este se abre, se hace una “request” llamando a la función *GetSharedGroupData* de la clase *PlayfabClientAPI*. De esta se obtiene el estado de la liga, la plantilla del jugador, la lista de puntos que hicieron los jugadores y la lista de partidos de la jornada anterior con respecto a la actual.

Una vez obtenida la información, se pinta el equipo en pantalla. Cada circulo naranja del menú representa a un jugador que, a su vez, también es un *GameObject* que tiene como hijos las imágenes de las estrellas que representan su calidad, un texto con el nombre del jugador y otro texto con los puntos.

6.8.2. Menú para modificar la alineación

Este menú es importante para el usuario porque des de aquí podrá gestionar el siete inicial de la plantilla y decidir cuáles son los jugadores que van a puntuar cuando se juegue el partido. Este menú se asemeja al anterior con la excepción de que se ha introducido:

- Un botón para guardar la alineación.
- Que cada círculo naranja que representa a un jugador también tenga un botón para poder seleccionarlo.

- Que en la parte izquierda se muestre los dos jugadores que tiene el usuario seleccionado con su respectiva información y un botón para poderlos cambiar de posición.

Ilustración 6.16: Menú para cambiar la alineación. Fuente: Elaboración propia.

Para ello, se ha creado un nuevo *script* llamado *LineUp*. Este al abrirse hace una "request" llamando a la función *GetSharedGroupData* de la *PlayfabClientAPI* para saber la plantilla del usuario, que es una lista de *HandballPlayerTeamSharedGroup*. Seguidamente se pinta el equipo en pantalla.

En cuanto a la selección de jugadores, se ha creado un *enum* con el nombre de *Selected* para saber cuál es la posición que tiene seleccionada el usuario. Después, se han instanciado dos *Selected* para identificar el primer y segundo jugador seleccionado para el usuario.

Entonces, cada vez que el usuario pulse un jugador, según el estado de las selecciones se hacen las siguientes comprobaciones:

- El jugador sobre el que se ha pulsado se convierte en la primera selección si esta no tiene nada seleccionado.

- La primera y la segunda selección se deseleccionan si la primera era la misma que el jugador sobre el que se había pulsado.
- El jugador sobre el que se ha pulsado se convierte en la segunda selección si la primera ya tiene un jugador seleccionado.
- La segunda selección se deselecciona si este era el mismo que el jugador sobre el que se había pulsado.
- Aparece el botón para cambiar los dos jugadores de posición si la primera y segunda selección tienen un jugador seleccionado.

Finalmente, una vez el usuario tenga formada la nueva alineación con la que quiere jugar, este puede pulsar el botón de guardar. Entonces, se ejecuta una “request” para actualizar el *SharedGroupData* con la nueva información.

6.8.3. Menú del mercado

El menú del mercado cuenta con dos menús dentro de él: el mercado donde el usuario va a pujar por los jugadores y un sitio para vender jugadores de su plantilla. Los dos son controlados por el *script* de *Market*. Al inicio se hace una “request” para obtener la lista de *restPlayers* (los jugadores que no tienen propietario), la lista de *HandballPlayerMarketSharedGroup* (los jugadores del mercado) y la lista de *HandballPlayerTeamSharedGroup* (la plantilla del jugador). Toda esta información servirá para gestionar los dos menús.

6.8.3.1. Pujar por jugadores

Este es el primer menú que se abre al entrar en el mercado. Está formado por un “scroll view” donde aparecen todos los jugadores del mercado y un botón para ir a la ventana de vender jugadores. Por cada jugador se muestra información sobre él, un “input field” para introducir la puja y un botón para pujar. Además, si hay una puja activa en un jugador, aparece otro botón que le permite al usuario anularla.

Ilustración 6.17: Menú del mercado de jugadores. Fuente: Elaboración propia.

Por cada jugador del mercado se instancia un objeto en el “scroll view” donde a su vez comprueba si existen pujas existentes del propio usuario en el jugador.

Una vez el jugador puja sobre un jugador, se hacen las siguientes comprobaciones:

- Que el valor sea numérico.
- Que no se supere el límite de jugadores que pueda tener el usuario con el fichaje de ese jugador.
- Que se supere el precio inicial del jugador.
- Que el usuario tenga dinero para poder hacer la puja.

En caso de que ya existiese una puja del usuario sobre el jugador en cuestión, se actualiza la puja anterior con el nuevo valor.

Una vez hechas estas comprobaciones, se hacen dos “request” para enviar la puja al *SharedGroupData*. En la primera se obtiene el mercado para saber si esta ha sufrido alguna actualización y en la segunda se actualiza el mercado.

6.8.3.2. Vender jugadores

Este menú se abre cuando se pulsa el botón de “SELL PLAYERS” de la Ilustración 6.17. Este es muy similar al menú para gestionar la alineación del equipo. A diferencia de este último, solo se puede tener seleccionado un jugador y en la parte izquierda de la pantalla aparece su información, un botón para venderlo directamente por su precio inicial y otro botón para enviarlo al mercado.

Ilustración 6.18: Menú para vender jugadores de la plantilla. Fuente: Elaboración propia.

Cuando se pulsa el botón para venderlo por el precio mínimo de compra, se hace una “request” para actualizar el dinero del usuario, su plantilla y la lista de *restPlayers*. En cambio, si se pulsa el botón para enviar el jugador al mercado, se hace una “request” para actualizar el mercado y su plantilla.

6.8.4. Menú de clasificación

Este menú está dividido en dos menús: el primero muestra la clasificación de la liga y el segundo el calendario de los partidos. Todo ello está controlado por el “script” de *ClassificationMenu*. Cuando se abre, este hace una “request” para obtener la lista de *LeagueTeam* (la lista de equipos con sus respectivos puntos), el estado de la liga y la lista de partidos de todas las jornadas que tiene la liga.

6.8.4.1. Clasificación

Este menú está formado por una imagen que a su vez tiene encima un texto que muestra cada equipo de la liga y sus puntos, y por un botón para ir al menú del calendario de partidos.

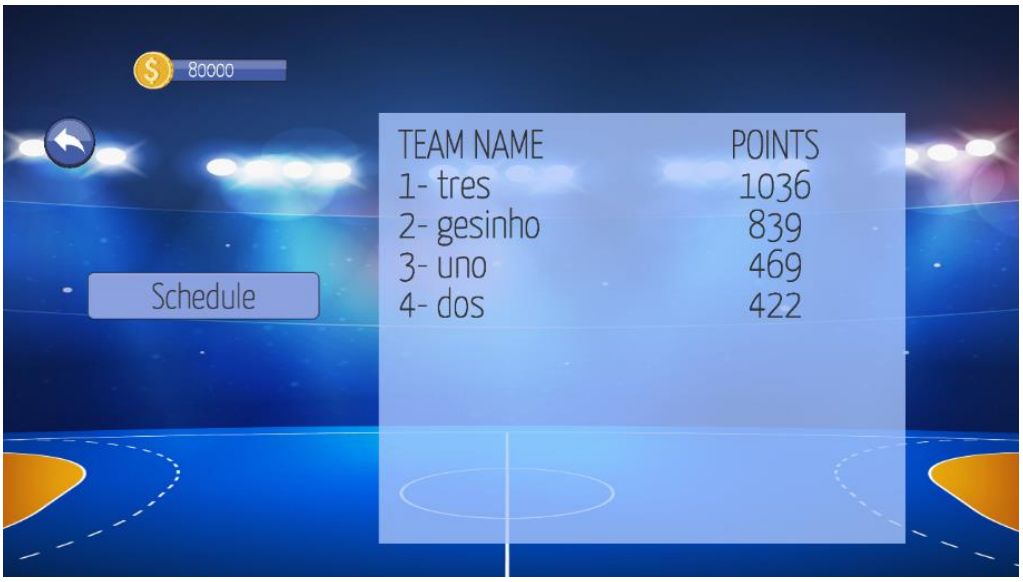

Ilustración 6.19: Menú de clasificación. Fuente: Elaboración propia.

Primero, se ordena la lista de la clasificación con la función *Sort* según los puntos de cada equipo y luego, por cada uno de ellos, se pinta su posición, el nombre del equipo y sus puntos.

6.8.4.2. Calendario de partidos

El menú del calendario está formado por un “scroll view” que, por cada jornada que tenga la liga, instanciará un texto con el partido que juegue el usuario. Los partidos que sean inferiores al número de la jornada actual de la liga mostrarán el resultado del partido, en caso contrario, este no se mostrará.

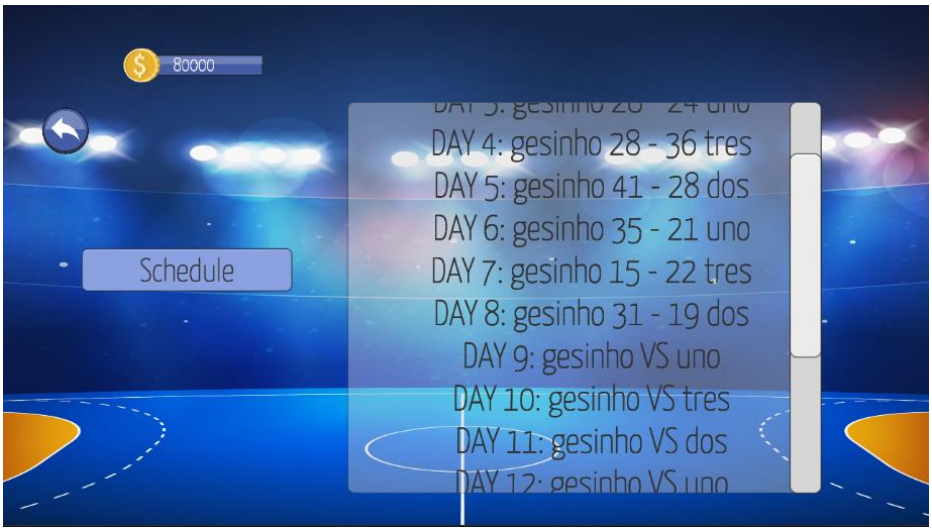

Ilustración 6.20: Menú del calendario de partidos. Fuente: Elaboración propia.

6.9. Creación del APK

Una vez se ha creado el juego, se ha revisado y se han pulido los errores; se tiene que obtener el ejecutable para poder jugar des del móvil.

Des del menú de “Build Settings” de Unity, se puede crear este ejecutable APK.

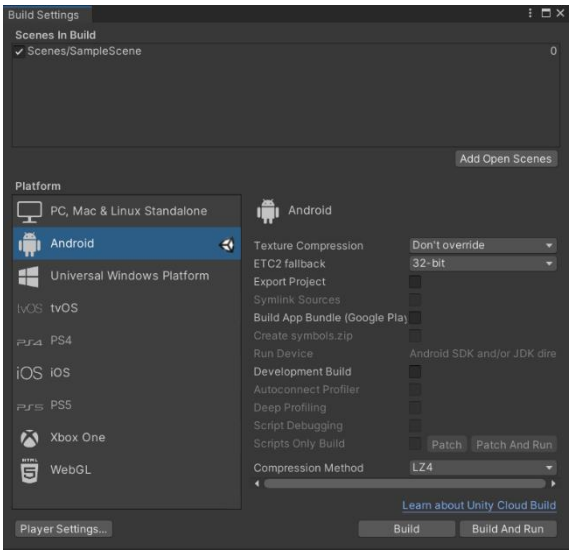

Ilustración 6.21: Imagen del menú de Build Setting de Unity (Unity Technologies, 2022).

7. Conclusiones y valoraciones finales

Este apartado consta de una valoración de la metodología con la que se ha trabajado y otra de los resultados finales obtenidos teniendo en cuenta los objetivos marcados des del inicio.

7.1. Valoración de la metodología

En cuanto a la metodología, ha sido vital cumplir con las datas del calendario, pero lo más importante aún, era que las dos últimas semanas se basarán en el pulido del juego. Esto ha permitido que el juego sufra poco en términos de errores.

Des del inicio ha habido problemas de comprensión para saber utilizar PlayFab. El problema más grande fue la función de crear grupos. Este apartado fue más complejo de lo que uno podría pensar, ya que los grupos no contaban con una sección para almacenar datos. Entonces fue cuando se tuvo que experimentar con el *SharedGroupData* que era un apartado separado a los grupos. Finalmente, y para no dedicar demasiado tiempo en este apartado, se decidió utilizar los grupos para buscar en el usuario en qué ligas formaba parte, y el *SharedGroupData* para almacenar la información. Por este motivo también se decidió que, para añadir a un amigo a la liga, no fuese por invitación, sino que simplemente se escribiera el nombre y de este modo se añadiera.

Por otra parte, las limitaciones que tenía PlayFab por haberse utilizado la versión gratuita ralentizaban el desarrollo del proyecto. Factores como el número de “key/values” que se podían actualizar en el *SharedGroupData*, el tamaño que no podían superar los datos almacenados y el número de Scheduled Tasks activas que podía haber. Todo ello provocaba que se buscasen formas alternativas para llevar hacía adelante el proyecto.

Aun así, se llegó a falta de dos semanas con un videojuego jugable y con la mayoría de las funciones esenciales de los videojuegos de mánager introducidas.

7.2. Valoración de los resultados finales

A nivel de objetivos, se ha logrado crear un proyecto jugable donde el usuario pueda disputar una liga con otros amigos. El juego contiene las funciones más recurrentes de los juegos habituales de gestión deportiva, como el mercado, la gestión de la plantilla, poder ver qué tal ha sido la actuación de un jugador, la clasificación, el calendario y poder vender jugadores. Todo ello era fundamental para que se asemejase lo máximo posible a un videojuego de este género.

Además, se ha usado otro nuevo lenguaje de programación (JavaScript), aparte de C# para así poder ejecutar código des del servidor. Esto permitió que todos los días a cierta hora se ejecute una función que actualizará el estado de la liga. Esta función era un algoritmo que, a partir del nombre del grupo, se buscaba la jornada actual de la liga y generaba los puntos de los jugadores, los puntos de cada equipo y el resultado final. Esta tiene la peculiaridad de estar basada en datos de partidos de balonmano reales y que, con las características que se le asignaron a cada jugador de la base de datos, se obtengan resultados que se acercan bastante a la realidad.

También, el juego cuenta con su propia base de datos de todos los jugadores del club almacenada en el servidor del juego en formato JSON. Este formato sirvió para almacenar todo tipo de datos en el servidor y también para después poder leerlos dentro del juego. Pero lo más importante, es que todos los datos guardados en el servidor facilitaban mucho al desarrollador la tarea de cambiar los valores de forma muy sencilla.

A nivel personal, este trabajo ha sido el primer contacto con la creación de un videojuego *online*. Se ha conocido PlayFab, una plataforma que utilizan videojuegos famosos como Minecraft (Mojang, 2011), Roblox (Roblox Corporation, 2006) o Tom Clancy's Rainbow Six Siege (Ubisoft, 2015). Aunque es limitado en el sentido de querer usarlo para desarrollar un videojuego de grandes características de forma gratuita.

Se ha aprendido a programar con JavaScript y a almacenar datos en formato JSON, y junto a ello la serialización y la deserialización que sirvió para poder transformar la información para así poder leerla y almacenarla.

En cuanto a los videojuegos de gestión deportiva, se observó que las estadísticas individuales de cada jugador son muy importantes, ya que con ellas se puede generar una puntuación sobre su actuación en un partido. También lo son las estadísticas de un partido real de cualquier deporte, ya que ellas pueden ser la base para crear un algoritmo que simule un partido y que este se asemeje lo máximo a la realidad.

Las entrevistas que se buscaron sobre programadores y diseñadores fueron vitales para entender cómo funcionan los juegos de este tipo, las características más importantes, la organización y producción, las estadísticas…

También, fue importante conocer el sistema “Round Robin” que se sigue usando actualmente en competiciones deportivas reales para crear un calendario de partidos con enfrentamientos que no se repitan.

En cuanto a una futura mejora del juego, todavía se le podrían añadir algunas características. Por ejemplo, una opción para poder ver los equipos de los otros usuarios y enviarles ofertas a los jugadores, una función de “login” automático o que las Scheduled Tasks se llamen siempre a la hora que se creó el grupo y no siempre a las 00:00.

Para terminar, una mejora que ayudaría mucho a mejorar el juego, pero es bastante costosa; consiste en pagar por las versiones de pago de PlayFab y así no depender tanto de las limitaciones que ofrece su versión gratuita. Aunque viendo que es un juego muy especializado y que no dispondría de muchos usuarios, es una opción poco viable.

8. Bibliografía/Referencias

Adamus T. (2012) Playing Computer Games as Electronic Sport: In Search of a Theoretical Framework for a New Research Field. In: Fromme J., Unger A. (eds) Computer Games and New Media Cultures. Springer, Dordrecht. https://doi.org/10.1007/978-94-007-2777-9_30

Belli, S., & Raventós, C. L. (2008). Breve historia de los videojuegos. Athenea Digital. Revista de pensamiento e investigación social, (14), 159-179.

Benito Robles, S. (2019). Juegos de tipo mánager: una aproximación basada en soluciones heurísticas y metaheurísticas.

Borondo, S. (2021, 12 de noviembre). Carlos Abril, el programador que creó PC Fútbol. Vandal. Recuperado el 19 april 2022, de https://vandal.elespanol.com/noticia/1350749053/carlos-abril-el-programador-que-creo-pc-futbol/

Caballero, D. (2018, 5 de febrero). Carlos Abril y PC Fútbol en el mundo y en el tiempo. Gamereactor España. Recuperado el 19 april 2022, de https://www.gamereactor.es/carlos-abril-y-pc-futbol-en-el-mundo-y-en-el-tiempo/

Castillo, A. (2020, 9 de noviembre). Football Manager 2021, impresiones y entrevista exclusiva con Miles Jacobson. MeriStation. Recuperado 19 de abril de 2022, de https://as.com/meristation/2020/11/09/avances/1604925396_173104.html

Comunio (2020, 15 de octubre). Entrevista a SofaScore: «el impacto estadístico de Messi es brutal». ComunioMagazine. Recuperado 19 de abril de 2022, de https://magazine.comunio.es/entrevista-a-sofascore/

Comunio (Versión para móvil) [Videojuego]. (2000). comunio. www.comunio.es

Derwort, J. (2014, 8 diciembre). Game Design & Scaling for Online Soccer Manager (OSM). Jeroen Derwort. Recuperado 19 de abril de 2022, de https://www.slideshare.net/Randam/presentatie-vua

ECMA - 404. (2017). JSON. Recuperado el 11 de junio, 2022, de www.json.org

FIFA 22 (Versión para consola) [Videojuego]. (2021). EA SPORTS. www.ea.com

Frankie MB. (2021, 14 de noviembre). ¿Cómo es lanzar un videojuego de fútbol en 2022? Hablamos con el equipo de Sociable Soccer, el heredero... Vidaextra. Recuperado el 19 de abril de 2022, de www.vidaextra.com

Freepik. Recursos Gráficos Para Todos. (2010). Recuperado el 5 de junio, 2022, de https://www.freepik.es/

MariaDB Foundation. (2019, November 13). MariaDB.Org. https://mariadb.org/

GARRO, A. (2014). Html5. licencia Creative Commons Recuperado el, 20.

Handball Manager 2021 (Versión para computadora) [Videojuego]. (2021). netmin games. store.steampowered.com/app/1262930/Handball_Manager_2021/

IBM. (2015). IBM Documentation. Recuperado del 11 de junio, 2022, de www.ibm.com

Kelly S., Kumar K. (2022) RESTful APIs. In: Unity Networking Fundamentals. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7358-6_3

Kelly S., Kumar K. (2022) TCP Connections. In: Unity Networking Fundamentals. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7358-6_4

Kent, E. (2018, 3 de septiembre). Entrevista con los desarrolladores de FIFA 19. Eurogamer. Recuperado 19 de abril de 2022, de https://www.eurogamer.es/fifa-19-entrevista

LaLiga Fantasy MARCA 21-22 (Versión para móvil) [Videojuego]. (2021). La Liga Nacional de Fútbol Profesional. www.laligafantasymarca.com

Landeta, J. M. I., & García, V. H. (2010). Metodología para elaborar un calendario deportivo bajo el sistema “round robin”. Tlatemoani: revista académica de investigación, (1), 6.

Maestre, M. T. G. TEMAS DE ACTUALIDAD EN ADICCIONES.

Marca. (1938). Marca.com. https://www.marca.com/

Mason, E. (2021, 20 de julio). FIFA 22 finalmente debería sentirse más justo: entrevista con el desarrollador Sam Rivera. Recuperado 19 de abril de 2022, de https://dlprivateserver.com/fifa-22-finalmente-deberia-sentirse-mas-justo-entrevista-con-el-desarrollador-sam-rivera/

Matas, S., & del Peral, J. P. (2019). Yanone Kaffeesatz. Google fonts. Recuperado el 5 de junio, 2022, de https://fonts.google.com/specimen/Yanone+Kaffeesatz

Merino A. (2018). Entrevista: Nacho Ruiz. Un pasado mejor. . . Recuperado 19 de abril de 2022, de http://videojuegosretro-upm.blogspot.com/2018/06/entrevista-nacho-ruiz.html

Microsoft Corporation. (2020). Microsoft Azure PlayFab. Recuperado el 18 abril 2022, de www.playfab.com

OSM 21-22 – Football Game (Versión para móvil) [Videojuego]. (2021). Gamebasics BV. es.onlinesoccermanager.com

Payo, A. (2020, 11 de mayo). Nordeus: “Llegamos a un punto con Top Eleven en el que no le dedicábamos el cariño que requería”. Applicantes. Recuperado 19 de abril de 2022, de https://applicantes.com/nordeus-top-eleven-10-aniversario/

Pérez Clemente, A. (2018). Desarrollo de un juego basado en red mediante Unity.

Pérez, J. E. (2019). introduccion a JavaScript.

Playfab. (2014). Azure Playfab Documentation - Playfab. PlayFab | Microsoft Docs. Recuperado el 11 de junio, 2022, de https://docs.microsoft.com/

Redis Labs. (2009). Redis. https://redis.io/

Ribeiro Caçador, A. (2016). Prototipo de juego para dispositivos iOS en el que se dirige un equipo de baloncesto colegial (Bachelor's thesis, Universitat Politècnica de Catalunya).

Sáez Blázquez, F. J., Roldán Romero, A., & Feu Molina, S. (2009). Diferencias en las estadísticas de juego entre los equipos ganadores y perdedores de la copa del rey 2008 de balonmano masculino.

Stats Perform. (2022). Opta data from. https://www.statsperform.com/opta/

The UNIX® Standard. (2019, 4 de junio). The Open Group Website. https://www.opengroup.org/membership/forums/platform/unix

Top Eleven Be Football Manager (Versión para móvil) [Videojuego]. (2010). Nordeus. www.topeleven.com

Unity Technologies. (2022, 29 de enero). Unity - Manual: Unity User Manual 2020.3 (LTS). Unity Documentation. https://docs.unity3d.com/Manual/index.html

Vera, J. A. C. (2015). La dimensión social de los videojuegos' online': de las comunidades de jugadores a los 'e-sports'. Index. comunicación: Revista científica en el ámbito de la Comunicación Aplicada, 5(1), 39-51.

Villanueva, R. (2013, 29 de agosto). Entrevista con desarrolladores de Madden NFL 25. LevelUp. Recuperado 19 de abril de 2022, de https://www.levelup.com/

Zhang, Y. (2021, febrero). Design of Remote Control System for Smart Home Based on Unity and the Internet of Thing. In Journal of Physics: Conference Series (Vol. 1744, No. 2, p. 022099). IOP Publishing.

TecnoCampus

Centres universitaris adscrits a la

upf. Universitat Pompeu Fabra Barcelona

Grado en Diseño y Producción de Videojuegos

Creación de un videojuego de mánager online de un club deportivo
Anexos

Xavier Gesa Bages
Tutor: Jorge Arnal Montoya

TecnoCampus Mataró-Maresme

1. Localización de anexos

A continuación, se muestra el contenido digital anexado y su ruta en la carpeta compartida de Google Drive:

- **Código fuente (Proyecto de Unity):**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\ProyectoUnity.rar
- **Ejecutable:**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\CHPalautorderaManager.apk
- **Vídeo demostrativo:**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\Vídeo.mp4
- **Código fuente (Código que se encuentra en el *CloudScript* del servidor de PlayFab):**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\Código de PlayFab\CódigoCloudScript.js
- **Base de datos de los jugadores almacenada en el servidor de PlayFab:**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\Código de PlayFab\BaseDatosJugadores.json
- **Base de datos de las variables modificables de las normas del juego en el servidor de PlayFab:**
  Gesa_Bages_CreaciónVideojuegoOnline_TFG\Anexos\Código de PlayFab\BaseDatosReglas.json

2. Material de terceros

Para la creación del prototipo se han utilizado los siguientes *assets*:

- Fondo de un campo de balonmano, obtenido de Freepik.
  Enlace: Fondo de campo de balonmano
- Newtonsoft Json Unity Package, obtenido de Unity – Manual.
  Enlace: Newtonsoft Json Unity Package

- PlayFab SDK Unity Package, obtenido de Microsoft Azure PlayFab Documentation.
  Enlace: PlayFab SDK
- Microsoft Azure PlayFab, web donde se encuentra el servidor del juego.
  Enlace: Microsoft Azure Playfab

3. Obtención del ejecutable

En este apartado se explica cómo obtener el ejecutable des del proyecto de Unity cuya versión es la 2020.3.4f1. Se recomienda usar esta versión para asegurar un funcionamiento correcto del proyecto. Para obtener el ejecutable, los pasos a seguir son los siguientes:

- Descargar la versión de Unity mencionada anteriormente.
- Descargar el proyecto y abrirlo con esa versión de Unity.
- Clicar en la opción de “File” y luego “Build Settings”.
- Asegurarse que la opción “Scenes/SampleScene” de “Scenes in Build” está marcada.
- Seleccionar la plataforma de Android.
- Clicar el botón de Build.