TecnoCampus
Escola Superior Politècnica

Centre adscrit a la
upf. Universitat Pompeu Fabra Barcelona

Diseño de una APP per mecanizar el proceso de innovación

Memoria Final

Grado en Ingeniería Informática de Gestión y

Sistemas de Información

Pau López Méndez

Tutor: Dr. Jaume Teodoro i Sadurní

2021-2022

TecnoCampus
Mataró-Maresme

Resumen

Este proyecto pretende plantear, diseñar y desarrollar un 'serious game' con la idea de divulgar información sobre el método TOOLBOARD, un modelo de diseño emprendedor desarrollado por Jaume Teodoro. Dicho juego está diseñado alrededor de las cartas, para ser accesible y educativo, a la vez que entretenido y divertido. El proyecto ha sido implementado usando el modelo Cliente-Servidor, utilizando Node.Js y Unity, y comprende un aplicativo web en el que un usuario puede acceder sencillamente.

Resum

Aquest projecte pretén plantejar, dissenyar i desenvolupar un 'serious game' amb la idea de divulgar informació sobre el mètode TOOLBOARD, un model de disseny emprenedor desenvolupat per en Jaume Teodoro. Aquest joc està dissenyat al voltant de les cartes, per a ser accessible i educatiu, alhora que entretingut i divertit. El projecte ha sigut implementat fent server el model Client-Servidor, utilitzant Node.Js i Unity, i comprén una aplicació web en la qual l'usuari pot accedir senzillament.

Abstract

This project aims to design and develop a serious game, with the idea of spread information about the TOOLBOARD method, an entrepreneur design model developed by Jaume Teodoro. The game is designed around cards, to be accessible and educative, and, at the same time, entertaining and fun. The project has been implemented using the Client-Server model, using Node.Js and Unity, and comprehends a web app where the user can easily access.

Índice

Índice de figuras

Glosario de términos

TFG Trabajo de Final de Grado

MVP Minimum Viable Product

RNG Random Number Generators

1 Objeto del proyecto

Los videojuegos educativos permiten complementar el aprendizaje y reforzar la memorización de los usuarios puesto que éstos están inmersos en la actividad a la vez que disfrutándola [1]. El término *serious games* se usa para llamar a esos juegos que tienen al menos una intención más allá de entretener [2].

El objetivo del proyecto es diseñar un *serious game* en forma de juego de cartas y, posteriormente, llevarlo a la plataforma digital en forma de videojuego online.

El proyecto pretende crear una actividad interactiva y divertida para que los jóvenes emprendedores aprendan conocimientos sobre TOOLBOARD, el modelo de diseño emprendedor creado por Jaume Teodoro, PhD, el cual les ofrece un método que seguir para llegar a cumplir sus objetivos [3].

A su vez, el videojuego es una puerta de entrada a los conceptos básicos y, si uno quiere saber más al respecto, tendrá accesible en el mercado el libro de TOOLBOARD.

2 Estudio Previo

2.1 Marco teórico

2.1.1 Serious Games

Según Platón, el acto de jugar se puede usar como guía en el desarrollo de un infante y, por ende, el juego puede tener un propósito educativo. Aun así, hasta finales del siglo XVIII el juego fue visto como algo que se tenía que limitar en los niños. Fue a través del trabajo de los filósofos de la Ilustración Friedrich Schiller y Jean-Jacques Rousseau que comenzamos a reconocer un encuadre contemporáneo del juego como una actividad intrínsecamente útil [4].

Clark Abt fue un investigador estadounidense que se le acredita establecer las bases de los s*erious games* en su libro *"Serious Games"* (1970). Abt los define como juegos cuyo propósito es educativo y la intención principal no es la diversión del jugador; pero aun así eso no significa que los *serious games* no tengan que ser entretenidos [5].

A medida que el mercado de los videojuegos fue creciendo, también los títulos dedicados a educación, medicina, militares… Al llegar a un público mayor y cambiar la connotación negativa de los videojuegos, a estos se les empezó a llamar *Serious Games* 40 años después de su creación, para nombrar a esta nueva generación de juegos con propósitos serios [5].

Hay distintas razones para enseñar sobre un tema a través de un juego, pero hay una característica en concreto que se cita con frecuencia puesto que se considera esencial: La **motivación**, pues permite que los jugadores absorbidos en largas sesiones de juego, y de aprendizaje en este caso. Otra característica mencionada es el ***engagement***, pues se considera que el jugador debe sentirse emocionalmente involucrado en la actividad para activar el desarrollo cognitivo hacia ese aprendizaje [6], [7].

2.1.2 Metodología TOOLBOARD

TOOLBOARD es la metodología de diseño emprendedor diseñado por Jaume Teodoro, PhD, y el tema que quiere llegar a enseñar el producto final de este proyecto [3].

2.1.2.1 Objetivo

El objetivo de esta metodología es diseñar un proyecto emprendedor de inicio a fin con la que definir la iniciativa de emprendimiento e innovación de forma clara y asequible. El resultado es la

entrega de un proyecto innovador o emprendedor que llame la atención de un inversor y la tenga en cuenta.

2.1.2.2 Target

El target a quien va dirigido es el de emprendedores que quieren convertir ideas u oportunidades de negocio en una iniciativa o que buscan una solución a un problema o necesidad de la comunidad.

2.1.2.3 Enfoque

TOOLBOARD tiene un enfoque centrado en las siguientes características:

- Emprendimiento por diseño
    - La metodología tiene una base de Design Thinking. Ésta tiene una filosofía que dice que la innovación viene de dos formas:
        - Descubrir una oportunidad a partir de un cliente e investigar para llegar a una solución (crear algo nuevo).
        - Identificar una oportunidad para cubrir una necesidad y adaptar la solución al cliente (mejorar lo que ya existe).
    - Para evaluar el diseño, hay que fijarse en los tres pilares del éxito de una solución:
        - Deseable → Lo que la gente necesita y desea.
        - Viable → Capturar valor en forma de ingresos.
        - Factible → Lo funcional que soluciona el problema.

- Proceso iterativo y centrado en la persona
    - El proceso tiene un enfoque de Human Centered Design, que establece un proceso iterativo entre tres pilares:
        - Hear → Entender el problema o necesidad.
        - Create → Generar ideas de solución y realizar oferta.
        - Deliver → Prototipar y testear.

2.1.2.4 Metodología

TOOLBOARD divide el proceso de la idea emprendedora en tres grandes fases:

1. **Fase de Investigación** (Zona de problema)
    - Se parte de un reto para identificar una oportunidad concreta.
    - Solución basada a partir de insights de cliente.

- Concluye con una síntesis de los principios de diseño.

2. **Fase de Ideación** (Zona de solución)
   - Se parte de los resultados de la investigación.
   - Se quiere buscar ideas de una solución que cree valor al cliente a la vez que sea factible y viable para la organización.
   - Concluye con una teoría para ser validada.
3. **Fase de Validación** (Zona de demanda)
   - Prueba empírica de la aplicación práctica.
   - Experimentación con clientes reales con un MVP.
   - Concluye con un proyecto listo para ser invertido.

A su vez, cada una de estas fases están compuestas de tres escenas, sumando un total de las nueve escenas que conlleva la metodología y son la clave del proceso (más una escena inicial de crear el equipo emprendedor). Las escenas contempladas en TOOLBOARD se muestran en la figura 1.

Figura 1. Escenas de TOOLBOARD. Fuente: [3]

Cada escena enseña un esquema de trabajo que hace de guía, y está compuesta por:

- Tareas → Conjunto de acciones a realizar a partir del objetivo de cada escena.
- Métodos → Los procedimientos que efectúan las tareas y conducen a resultados.

- Resultados → Conocimiento extraído a partir del diseño.
- Herramientas de acción → Representación visual de los resultados que facilita el trabajo con ciclos de iteración.

Un resumen de cómo funciona la metodología se muestra en la figura 2.

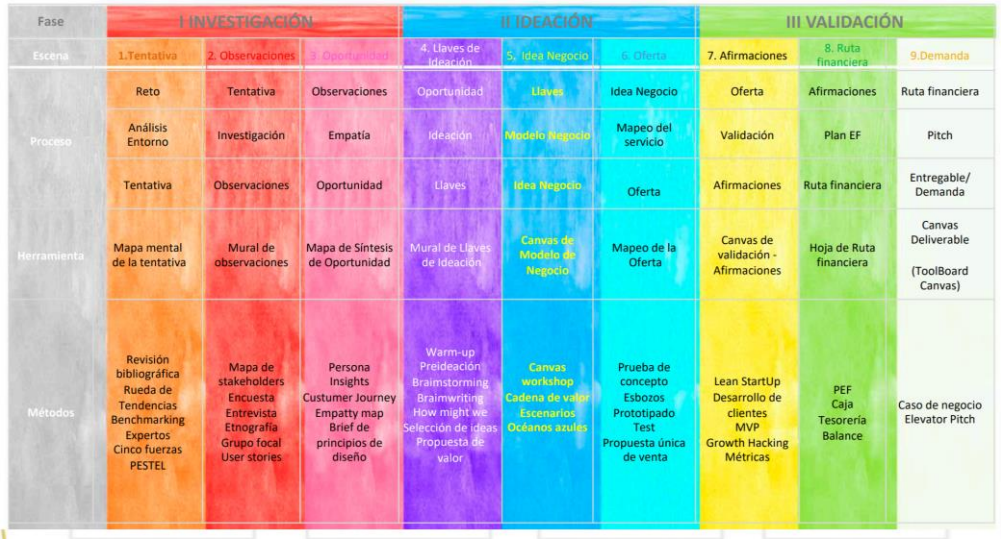

Figura 2. Resumen de métodos y herramientas TOOLBOARD. Fuente: [3]

2.2 Estudio de herramientas del proyecto

Se ha realizado un estudio de las distintas herramientas que darán soporte a crear el producto final y justificar brevemente porqué la elección.

2.2.1 Tabletop Simulator

Tabletop Simulator es el principal sandbox de juegos de mesa del mercado. No existe competidor conocido y es debido a las grandes ventajas que ofrece:

- Ofrece mucha libertad a la hora de añadir y modificar assets de juego.
- Fácil y sencillo de construir un juego personalizado y compartirlo con otros jugadores.
- Al ser un diseño en digital, no es necesario tener gastos en bienes materiales para los prototipos de cartas.
- Mucha documentación sobre como implementar tus propios tipos de assets.

2.2.2 Unity

El motor de juego es una elección importante para crear un videojuego, pues ofrece un framework y unas herramientas de trabajo básicas y avanzadas que son estándar en el desarrollo de un videojuego.

Tres de los grandes juegos de cartas digital coleccionables del mercado han usado Unity para su desarrollo: Hearthstone, Legends of Runeterra y Magic: The Gathering Arena. Estudiando la introspección del equipo de Hearthstone [8] y de Legends of Runeterra [9], destacan aspectos como la facilidad de adaptar el juego a plataformas más allá del PC o la versatilidad de la interfaz del editor de Unity.

Unity es el motor de juegos más popular entre los juegos de Itch.io [10], y eso es debido a que es un motor simple de usar pero que a su vez puede llegar a tener mucha profundidad. Además, es el motor de juegos con la documentación más completa y extensa.

2.2.3 Colyseus

Colyseus es un framework multijugador para Node.js que se centra en proveer estructuras de datos sincronizables para juegos online.

2.2.3.1 Ventajas

- Proporciona facilidad al desarrollo del proyecto, puesto que ofrece una API simple tanto server-side como client-side.
- Facilita el diseño de matchmaking usando sesiones de juego.
- Facilita la replicación de datos entre clientes ya que mantiene un estado de la sesión haciendo que si algún cliente se reconecta después de una perdida de conexión sea automática la sincronización del estado actual.
- Proyecto Free Open Source con licencia MIT.
- Implementación directa y nativa con Unity.

2.2.3.2 Desventajas

Funciona a través de paquetes TCP en vez de UDP. Esto genera una mayor latencia, un mayor uso del ancho de banda y de la necesidad de procesamiento de datos. Al no ser un juego que necesite de una latencia baja no supone un gran problema.

NodeJS tiene un rendimiento bajo comparado con hacer un servidor en lenguajes de alto rendimiento como C++ o Rust, aunque no es un problema ya que la carga del servidor debería ser mínima[11].

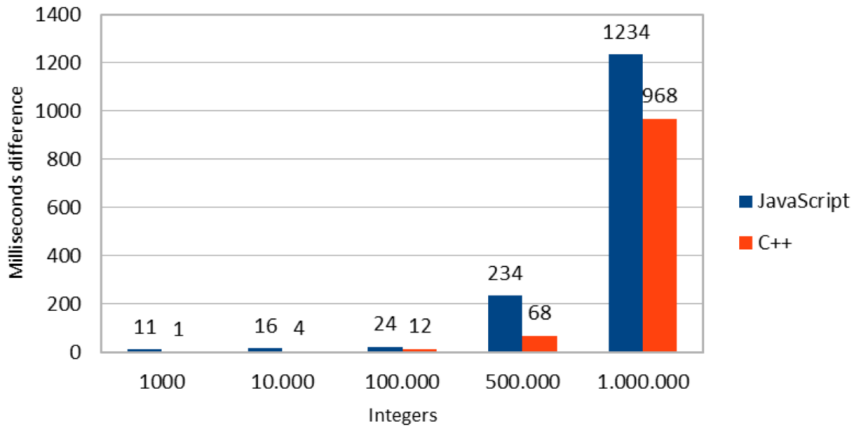

Figura 3. Comparación C++ vs Javascript (Node). Fuente: [11]

3 Objetivos y alcance

El cliente de este proyecto es el Dr. Jaume Teodoro, el cual comparte los siguientes objetivos a cumplir:

- Crear una forma divertida e interactiva que permita enseñar a los jugadores los aspectos más relevantes del TOOLBOARD y el desarrollo de ideas emprendedoras.
- Apoyar y publicitar el libro que publicará sobre el método TOOLBOARD a través de otros medios; en este caso, los videojuegos.
- Crear una idea de producto que se pueda ejecutar tanto de forma física como digital. El cliente especifica que sea un juego de cartas, pues es fácil y ligero de llevar en su versión física.

Para cumplir esos requisitos, se han establecido los siguientes objetivos para el proyecto:

- Investigar sobre el mercado de juegos.
    - Mercado de los juegos de cartas.
    - Mercado de los serious games
- Informarse sobre la teoría tras el diseño tanto de un juego de cartas como de un *serious games*. El último es muy importante, pues es un foco muy importante saber cómo aplicar un diseño cuya misión principal es la educación.
- Diseñar el juego de cartas aplicando los conocimientos aprendidos. Es importante realizar una iteración constante de rediseño y testeo hasta encontrar el producto que se quiera conseguir.
- Desarrollar un prototipo del juego de cartas para poder jugar de forma digital.
    - Desarrollar el backend con todas las funcionalidades requeridas a partir del diseño del juego.
    - Desarrollar un frontend user-friendly con el que interactuará el jugador.
- Establecer una conexión cliente-servidor para poder tener un videojuego online que enfrente dos jugadores.

Objetivos secundarios:

- Crear una interfaz de juego que sea visualmente atractiva.
- Adaptar el juego para ser jugado con dispositivos móviles.

Todos estos objetivos deben tener siempre en mente el público objetivo del proyecto. El producto está pensado para que lo jueguen aquellos a los que va dirigido el método TOOLBOARD: **jóvenes emprendedores (16-25 años)** con grandes ideas y dotes creativas, que sueñan con entrar en el mundo emprendedor.

El alcance del proyecto se centra en las tres partes de éste: investigación, diseño y desarrollo. La fase de desarrollo tiene el mayor peso, pero usando las herramientas correctas el alcance se vuelve de un tamaño acorde para una persona en el tiempo límite.

4 Metodología

El proyecto se dividirá en tres grandes fases:

4.1 Fase de anteproyecto

En esta etapa se realiza la búsqueda del marco teórico y el estado del arte, requerimientos necesarios para realizar un buen diseño del juego de cartas. También se determina de forma específica qué objetivos hay que cumplir, cómo se van a realizar y con qué herramientas trabajar, así como el presupuesto y la viabilidad que conlleva.

4.2 Fase de diseño

Esta fase conlleva el diseño del juego de cartas. Para ello, se realizan iteraciones rápidas de diseño de una semana de duración. Al final de cada ciclo, se presenta el producto al cliente para recoger feedback y seguir iterando. Al terminar la fase, se tiene un producto acabado para implementar de forma digital en forma de videojuego.

En el transcurso de cada ciclo, se realizan cambios y mejoras del diseño (menos la primera semana, en cuyo caso se trabaja puramente en la creación del diseño base) mientras se testea constantemente con una serie de jugadores con sentido crítico en los juegos de cartas.

4.3 Fase de desarrollo

Para esta fase se usa la metodología Agile, en concreto una implementación basada en Scrum. Este proceso tiene una serie de ventajas con el proyecto y, aunque su foco está centrado en equipos, se puede adaptar para un “equipo” de una persona. Estudios [12], [13] demuestran que en el mundo del desarrollo del videojuego es una práctica muy usada y tiene buenos resultados.

Algunas de las ventajas que ofrece scrum respecto prácticas tradicionales como Waterfall son [14]:

- Al realizarse de forma iterativa dividido en sprints, es fácil tener una buena visibilidad del estado del proyecto en todo momento.
- El testeo constante es muy bueno para el proyecto. Permite realizar cambios adecuados y fáciles y permite entregar un producto de calidad en menor tiempo.

- Mantiene relación constante con el cliente. Al ser un juego cuyo diseño se centra en la educación es perfecto contar con una metodología tan centrada en las personas, donde el feedback es muy constante.
- No requiere de escribir una extensa documentación, por lo que los costes de tiempo en ese aspecto decrecen para potenciar el desarrollo rápido.

En vez de aplicar el concepto de los sprints tradicionales, el proyecto se desarrolla usando la metodología modificada de scrum *ABC-Sprints*. Ésta divide el proceso de desarrollo en tres sprints de entre 2-4 semanas, donde cada sprint especifica de forma muy concisa en qué punto tiene que encontrarse desarrollo del juego tal como se ve en la figura 4 [15].

ABC - Sprints	***Alpha Sprint***	Basic Functionality, Proof of Concept, Beta Sprint Backlog
	Beta Sprint	Feature-complete Game, Final Assets, Completion Sprint Backlog
	Completion Sprint	Bug-free, Balanced, Polished and finalized Game

Figura 4. ABC-Sprints. Fuente: [14]

Al final del Completion Sprint, se realiza la entrega de la memoria final del TFG y el producto; y se empieza a preparar la defensa ante el tribunal.

5 Requerimientos funcionales y tecnológicos

5.1 Requerimientos funcionales

- Tener las funcionalidades del core game loop básico.
- Tener las funcionalidades de todas las mecánicas secundarias del juego.
- Permitir al jugador interactuar con la interfaz de juego para jugar.
- Permitir que los jugadores puedan crear salas de juego online y unirse en ellas.
- Ver un apartado visual pulido.

5.2 Requerimientos tecnológicos

- Estudiar y entender el funcionamiento del framework Colyseus

6 Desarrollo

En este apartado se expone la evolución del diseño del juego de cartas; desde un inicio en que se establecen los principios de diseño del producto hasta su resultado final.

6.1 Principios de Diseño

Siguiendo la metodología de Design Thinking, como paso previo al propio diseño del juego se han desarrollado una serie de principios de diseño, o *Insights,* que establezcan de forma clara los valores sobre los que se sostendrá el proyecto[3].

El diseño debe tener en cuenta que…

- **…el único elemento que se usa en el juego son cartas.**

  ¿Porqué? Por una parte, se busca la simpleza en la preparación del juego, y un mazo de cartas cumple ese requerimiento mientras a su vez tienen mucha flexibilidad a la hora de diseñar. Por otra parte, el cliente requiere que el producto físico sea fácilmente portable en el bolsillo de los clientes para poder llevarlo cómodamente de casa a la reunión.

  ¿Qué implica? Se añade una restricción al diseño del juego, cosa que permite explotar ese entorno per a su vez no permitiendo añadir elementos extras al diseño. Además, se ha determinado que una caja portacartas del tamaño de las cartas clásicas de UNO es muy adecuado para el producto.

- **…el cliente no tiene o tiene poca experiencia en juegos de mesa.**

  ¿Porqué? El cliente objetivo es alguien que está en constante búsqueda de conocimiento y creación de ideas, por lo que se asume que no ha tenido experiencia en juegos de mesa más allá de los más famosos dentro del entorno casual.

  ¿Qué implica? El diseño debe estar dirigido en todo momento a un público casual, por lo que se restringe la creación de unas mecánicas complicadas de entender, así como la cantidad de éstas, pues el cliente no debe pasar más de cinco minutos entendiendo las reglas del juego.

- **…el jugador tiene que aprender conceptos de TOOLBOARD mientras juega.**

  ¿Porqué? El propósito del producto final este proyecto es ser más que un juego cuya única finalidad sea la de divertirse, a pesar de ser la principal. También se busca que se

ayude al jugador, de una forma u otra, a interiorizar los conocimientos que aporta la metodología TOOLBOARD.

¿Qué implica? Es fácil pensar en que la temática del juego será sobre un equipo emprendedor y las tools de TOOLBOARD que usarán para conseguir una idea emprendedora. Ahora bien, hay que pensar detenidamente cómo el jugador llega a percibir las mecánicas de juego para que realmente llegue a asociarlas al método emprendedor.

6.2 Desarrollo del diseño de juego

6.2.1 Diseño inicial

El primer paso a la hora de desarrollar un juego de mesa es la ideación [16]. Se realizó un *brainstorming* exhaustivo teniendo en cuenta los principios de diseño establecidos previamente. Las ideas más interesantes que salieron fueron:

- En vez de tener cartas que mostrasen un equipo emprendedor, serlo el propio jugador. A partir de esta idea también se planteó realizar el juego por equipos e incluso ser un juego no competitivo, por lo que los jugadores debían plantearse las jugadas juntos para superar los retos.
- Usar los personajes del prototipo fallido, junto a la tabla de tools que establece cuales tiene cada uno de forma equilibrada. Cada tool determina una mecánica en específico, un poder, y el jugador podía reunir personajes con unas tools u otras dependiendo del estilo de juego al que se quería dirigir.

  A partir de la idea de que las tools realizan efectos beneficiosos para el jugador, se planteó cambiar la condición de victoria del prototipo fallido (reunir las nueve tools), pues las tools se usaban como herramientas más que como finalidad. La meta que se pensó era una *Idea*. El concepto de la carta de Idea tuvo dos planteamientos distintos: ser una carta que cada jugador conseguía al principio de la partida y determinaba cual era la meta para ganar, o ser una carta que podías conseguir entre muchas otras, pero podías evolucionarla a lo largo de la partida para poco a poco convertirse en la idea definitiva.
- Construir el juego con las nueve tools como la mecánica principal del juego. En esta idea, el juego consistía en nueve fases, una para cada tool. En cada una, cada jugador realizaba una acción distinta y, dependiendo de tu equipo emprendedor, podías hacerlas mejor o peor que los otros jugadores.

Tras el primer ciclo, se determinó que las ideas hasta el momento no eran realmente buenas. Por sí mismas podían dar juego a un juego interesante, pero el problema era que se iban totalmente de uno de los principios de diseño: "*el cliente no tiene o tiene poca experiencia en juegos de mesa".* La primera idea mencionada anteriormente requiere que los jugadores jueguen en grupo, teniendo así que crear un juego que añade una capa de complejidad sobre un juego individual, mientras que las dos últimas tenían una gran cantidad de mecánicas y se alejaban del diseño casual buscado. Analizando el razonamiento que se tuvo detrás de estas ideas, se puede concluir que el principio de diseño "*el jugador tiene que aprender conceptos de TOOLBOARD mientras juega"* tuvo mucho peso, pues ambas ideas tienden mucho a querer enseñar al jugador cómo funciona cada tool.

Estas ideas fueron en gran parte rechazadas, pues se requería buscar una dirección de diseño que balancease bien el querer enseñar con el querer ser divertido para jugadores casuales.

6.2.2 Primer ciclo

Encontrar un balance perfecto entre aprendizaje y diversión no es sencillo. En reuniones anteriores, el Dr. J. Teodoro afirmó que, en casos así, prefería que el juego fuese entretenido. Teniendo en cuenta esa declaración, se decidió poner más peso en ese principio de diseño. Para ello, se prescindió de intentar enseñar el funcionamiento del método TOOLBOARD en detalle, para construir un diseño de juego que busca educar a través de la memorización. Los jugadores juegan cartas de tools, y esa acción refuerzan su memoria sobre los conocimientos impartidos sobre el método. Recuerdan lo que significa, aunque el juego mecánicamente no lo haga. La mecánica principal del juego es ampliar el equipo emprendedor y hacer que aprendan esas tools, por lo que el núcleo del juego sí se relaciona con los valores del método.

Durante el primer ciclo se llegó a desarrollar una idea sólida de donde partió el producto final:

- Se reutilizaron las mismas cartas que el prototipo fallido, pues se consideró que el trabajo artístico realizado, así como el tener la lista de personajes con sus tools de forma balanceada era una buena labor que no tenía por qué desecharse.
- El juego tiene dos mazos de cartas: el mazo de personajes, con 33 en total; y el mazo de tools, formado por nueve tools, y donde cada una aparece cuatro veces, dando un total de 36 tools.
- El core game loop era el siguiente:
    - Al principio de cada ronda, se sacan cinco personajes del mazo de personajes y se colocan en el centro de la mesa.

  - Cada jugador roba cinco cartas del mazo de tools. Entonces se desarrolla una toma de cartas estilo *draft*, popular en juegos como *Magic: The Gathering.* Para ello, cada jugador coge una carta de las cinco y pasa las restantes al jugador de la derecha. Este proceso se repite hasta que cada jugador tiene cinco cartas en su posesión. Este método de toma de cartas permite que los jugadores se vean menos afectados por el RNG e interaccionar con la toma de cartas de sus rivales.
  - Por orden, cada jugador puede descartar dos tools que coincidan con un personaje del centro de la mesa más una cantidad de tools para haber descartado, en total, la misma cantidad que las que puede conocer ese personaje (cada personaje los tiene marcados en la carta). De esta forma puede "reclutar" a ese personaje para su equipo.
  - Un jugador también puede tomar dos o más tools de su mano que coincidan con las tools que puede conocer un personaje de su equipo. De esta forma, ese personaje las aprende. El razonamiento detrás de que sean dos es tener que calcular bien qué tools usar en los personajes con un total de tools impares (tres o cinco).
  - Al final de la ronda, se vuelven a colocar personajes en el centro de la mesa hasta que haya cinco y se mueve el orden de turnos a la derecha. Los jugadores descartan sus cartas sobrantes.
- El objetivo del juego era conseguir que el equipo emprendedor del jugador esté versado en las nueve tools establecidas en el método TOOLBOARD.

El juego fue prototipado dentro del entorno de Tabletop Simulator y se testeó con jugadores versados en los juegos de mesa.

La recepción del juego no fue mala, pero tampoco tuvo mucho éxito. Con el feedback de los jugadores, se analizaron qué puntos podían fallar del juego:

- La mecánica de que tanto para comprar como para hacer aprender a los personajes donde el jugador debe usar mínimo dos tools coincidentes se siente rara. Parece un número arbitrario que solo aporta complejidad innecesaria.
- El hecho de tener que adquirir tools que coincidan con un personaje para reclutarlo, y que luego tengas que volver a conseguir esas tools para enseñarle frustra a los jugadores. Sienten que tienen que hacer el doble de esfuerzo de lo que deberían.

- El *draft* toma mucho tiempo de juego, incluso más que la fase de reclutar y aprender. A los jugadores les cuesta hacer decisiones pues deben tener en cuenta información que no ven de forma temporal. Además, las decisiones se sienten arbitrarias.
- A los jugadores les frustra perder la mano sobrante al final de la ronda si contiene cartas que en un futuro les puede ser de utilidad.
- Los jugadores sienten que con cinco cartas en mano no tienen mucha libertad de toma de decisiones, y se siente mal reclutar un personaje de cuatro o cinco tools, pues luego no pueden hacer nada más en su turno. Además, aun tomando las cartas a través de un *draft*, sienten que hay mucho RNG implicado pues requieres de tener tools específicas en cada ronda.

6.2.3 Segundo ciclo

Durante el segundo ciclo se ha seguido testeando el juego para modificar así los errores de diseño que tenía el juego. Los cambios principales fueron los siguientes:

- A la hora de robar cartas de tools, los jugadores mantenían las cartas en su mano en vez de realizar un *draft*. De esta forma, el tiempo de juego se reduce de forma considerable. Es cierto que este método aumenta el RNG, pues las cartas que te tocan en mano no se pueden modificar, pero de esta forma se potencia el hecho de que un jugador lo haga lo mejor posible con lo que tiene, por lo que se potencia la optimización de las cartas. Además, con la nueva mecánica de reclutamiento y aprendizaje, el RNG no afecta tan negativamente a los jugadores como antes.
- La mecánica de reclutamiento solo requiere que descartes un número de tools equivalente a la cantidad de tools que puede conocer ese personaje, sin tener que descartar ninguna tool en específico. De esta forma, los jugadores pueden desechar las cartas que menos le interesan y conseguir beneficios.
- La mecánica de aprendizaje funciona diferente. En vez de tener que jugar mínimo dos tools a ese personaje, se permite cualquier número de tools coincidentes. Ahora bien, el aprendizaje solo se puede realizar una vez por personaje. Una vez un personaje ha aprendido (ya sea una o el máximo posible de tools a la vez), ya no se puede volver a elegir. De esta forma ya no existe un número mínimo, por lo que los jugadores son más libres a la hora de tomar decisiones, pero también es una decisión dura, pues depende de qué tools jueguen en un personaje, quizás otras tools que podría conocer se quedan bloqueadas, por lo que requieren de otro personaje distinto para tenerla.

- Una vez se han robado cartas, en vez de desarrollar la siguiente fase de juego de forma que los jugadores puedan decidir si reclutar o aprender, se ha establecido una primera fase de reclutar. Cuando todos los jugadores acaban la fase, entonces empieza la fase de aprender. De esta forma, se establece un orden de juego donde en cada fase un jugador realiza la misma acción (una o múltiples veces), y así mejorar el juego para jugadores casuales.
- Los jugadores pueden decidir mantener cualquier número de cartas al final de cada ronda. De esta forma, los jugadores tienen la libertad de poder realizar estrategias a largo plazo o simplemente no perder cartas que les interesan.
- Los jugadores roban un mayor número de cartas por ronda. El jugador que va primero en esa ronda roba hasta siete cartas, es decir, si ha mantenido una carta de la ronda anterior robará seis; mientras que el último jugador roba hasta nueve cartas. Los jugadores así tienen mayor libertad de acciones por turno, ya sea para reclutar y luego hacer aprender a ese personaje, reclutar múltiples veces, mantener cartas para futuros turnos… El hecho de que el último y el primero roben distintos números de cartas es para balancear el hecho que el primer jugador tiene mayor ventaja a la hora de reclutar personajes.

Adicionalmente, posteriormente se añadieron nuevas modificaciones de balanceo:

- Se testeó y se llegó a la conclusión de que el juego de cartas se puede jugar sin problemas de dos a cuatro jugadores. Los jugadores que no son ni primeros ni últimos robaban hasta ocho cartas.
- Se redujo el número de cartas de personajes que se colocan en el centro de la mesa de cinco a cuatro para el modo de dos jugadores, pues se determinó que una elección menos daba mayor juego. Por cada jugador que se añade a la partida, se sacan dos cartas de personajes adicionales.
- Se dobló la cantidad de cartas del mazo de tools (de 36 a 72). En una partida de múltiples jugadores, el mazo se acaba rápido y entonces hay que mezclar la pila de descartes para volver a generar un mazo. Tener que hacer esa acción múltiples veces cada poco tiempo puede llegar a ser una molestia.
- El hecho de que el primero robe hasta siete cartas y el último hasta nueve era una capa de dificultad añadida que hacía que los jugadores tuviesen que pensar de más en cada ronda. Para dejar unas normas claras a jugadores casuales, se ha modificado para

que todos los jugadores roben siempre hasta ocho cartas. Este cambio seguirá testeándose para ver si el primer jugador realmente tiene mucha ventaja.

El resultado del tercer ciclo es muy positivo. La recepción de los jugadores ante los cambios ha sido buena y la experiencia de juego ha mejorado mucho. El juego también ha sido testeado en formato físico, con resultados igual de positivos.

Figura 5. Mesa de juego en Tabletop Simulator. Fuente: Elaboración Propia.

6.2.4 Tercer ciclo

Se ha realizado un último ciclo de diseño cuyo foco principal es el test realizado de forma presencial junto al Dr. Carlos González Tardón, doctor en Psicología, Ocio y Desarrollo Humano y con una tesis doctoral centrado en los Serious Games. Tras un debate después de la sesión de juego, se ha determinado realizar unos últimos cambios al diseño:

- **Un jugador puede tener un máximo de tres personajes reclutados**

  Con este cambio se busca arreglar el problema de poder dejar bloqueada alguna tool, pues se hace más difícil que los jugadores lleguen a amontonar suficientes personajes que compartan una tool como para quitar la posibilidad de conseguir esa tool para otros jugadores.

- **Cuando un jugador usa las tools para "aprender", ese personaje se pone en una pila de descartes en vez de quitarla del juego**

  Esta norma aparece inmediatamente tras la anterior, pues ambos cambios juntos permiten mejorar mucho la fluidez de las cartas de personajes. El interés es no poder llegar a bloquear tools para que el jugador no se sienta frustrado. Por ello, en visión de cualquier mínimo caso en que, aun con un máximo de tres personajes por jugador, parezca que una tool pueda llegar a ser bloqueada, esta norma complementa la anterior y no permite el bloqueo siempre que los jugadores jueguen a ganar.

  Adicionalmente, estas reglas también tienen la función de mejorar la interfaz de juego (tanto físico como digital). Con un máximo de personajes delante del jugador que, al usarlos, se van a una pila de descartes común, la mesa se ve mucho más limpia.

- **La forma narrativa de explicar el juego se adapta más al loop de juego**

  Hasta entonces la narrativa trataba de que el jugador, como empresario, recluta emprendedores más o menos capacitados para su empresa (fase de reclutamiento) y luego les enseña a usar tools que, por sus capacidades, son capaces de dominar (fase de aprendizaje).

  Como el loop mecánico de juego hace que el jugador coja cartas de personaje para luego descartarlas, al Dr. Carlos González no le llega a cuadrar la relación entre el loop y la narrativa.

  - **Los personajes son emprendedores expertos que ayudan al jugador**

    Los personajes se descartan a una pila común, es decir, que otros jugadores pueden eventualmente reclutar un personaje que ya reclutaste en el pasado. Por ello se estableció que, en vez de considerarlos emprendedores que entran a tu empresa, éstos son expertos emprendedores que ayudan al jugador y luego se marchan.

- **El aprendizaje de tools es del personaje hacia el jugador, en vez de al revés**
  Tras el cambio de narrativa de los personajes, la fase de aprendizaje también se modifica, pues ahora los personajes ayudan al jugador de forma temporal, para luego irse. Por ello se determina que, cuando el jugador usa las tools en un personaje durante la fase de aprendizaje, narrativamente está adquiriendo los conocimientos de esas tools de ese experto.
- **El jugador no es un empresario, sino un emprendedor**
  Con la mecánica de reclutar personajes es fácil caer en la narrativa de que el jugador está montando su propia empresa contratando a esos personajes. Con los cambios anteriores, se puede dar un mayor énfasis a que un emprendedor puede ser o no ser un empresario. Lo que se acaba diciendo con la nueva narrativa es que uno de los puntos más importante de empezar a ser un emprendedor es aprender una metodología, en concreto, el TOOLBOARD.

6.2.5 Diseño final del juego

En este apartado se redactan las normas finales del juego. Para ello, se utiliza unas normas gramaticales apropiadas a la redacción de instrucciones.

"TOOLBOARD Canvas: El juego de cartas te ofrece una experiencia divertida y fácil con la que no solo pasarás un buen rato con tu familia, tus amigos o tus compañeros emprendedores; además aprenderás que el conocimiento de una metodología es clave. Saca la baraja para reclutar expertos, aprender de ellos y llenar tu cabeza de las útiles herramientas que ofrece TOOLBOARD Canvas."

Edad: 8+ (Basado en la edad recomendada del juego de mesa "Virus!", el cual es un referente.)

Jugadores: 2-4 jugadores

Tiempo medio de juego: 10-20 minutos

Contenido

- 33 cartas de emprendedores

Figura 6. Carta de Emprendedor. Fuente: Elaboración Propia.

- 72 cartas de tools

Figura 7. Cartas de Tools. Fuente: Elaboración Propia.

(El total de cartas es 105. El mazo clásico de UNO, el popular juego de cartas casual contiene un total de 112 cartas, por lo que se considera un tamaño de mazo adecuado).

Preparación

Mezcla el mazo de cartas de emprendedores y el mazo de tools. Coloca ambos mazos en el centro de la mesa, al alcance de todos los jugadores. Al lado de cada uno se colocarán boca arriba las cartas descartadas de ese tipo. Cuando un mazo se agota, se mezcla la pila de descartes y se coloca boca abajo, formando una nueva pila de robo.

Objetivo del juego

El primer jugador que adquiera los conocimientos de las 9 tools distintas de la metodología TOOLBOARD gana la partida.

Fases de juego

El juego tiene dos fases que ciclan durante toda la partida: Fase de Reclutamiento y Fase de Aprendizaje. El juego empieza con la Fase de Reclutamiento y empieza el primer jugador que diga una idea emprendedora, como si de un brainstorming se tratara (de esta forma se aprende a desarrollar ideas y no tener miedo de soltarlas al mundo).

Fase de Reclutamiento

- Cuando empieza la fase, todos los jugadores robáis cartas de tools hasta tener 8 en mano. Estas cartas se mantienen en secreto de los otros jugadores.

- Coged 4 cartas del mazo de emprendedores (si jugáis más de dos personas, coged 2 cartas adicionales de emprendedores por cada jugador extra). Colocad esas cartas boca arriba formando una fila, formando así el *Mercado de Emprendedores*.

- Durante tu turno, puedes reclutar un emprendedor del mercado. Para hacerlo, descarta de tu mano tantas cartas de tools como el total de tools que tiene marcado el personaje. Si lo haces, coloca ese emprendedor en tu *Zona de Reclutas* y pasa el turno al jugador de tu izquierda.

  ***Importante**: Puedes tener hasta un máximo de 3 emprendedores reclutados a la vez.

Fase de Aprendizaje

- Al igual que la anterior fase, el jugador que se declaró jugador inicial empieza su turno en primer puesto.

- Durante tu turno, puedes aprender de los expertos emprendedores que has reclutado. Para hacerlo, descarta de tu mano tools que coincidan con las que tiene marcado ese emprendedor. Al hacerlo, coloca ese emprendedor en la pila de descartes y coloca las tools descartadas en tu *Zona Toolboard*. Pasa tu turno al jugador de la izquierda.

  ***Importante**: No puedes descartar más de una copia de una misma tool a la hora de aprender de un emprendedor.

- Si al realizar la acción de aprender consigues tener las 9 tools distintas en la *Zona Toolboard*, ganas la partida.

Pasar turno y Transición de fase

- Si no puedes o no quieres hacer la acción designada de la fase, puedes decidir pasar tu turno.

- Si pasas turno, tu turno será omitido hasta que empiece la siguiente fase.

- Los otros jugadores seguirán recibiendo sus turnos (es decir, que puedes hacer la acción de la fase múltiples veces, pero en turnos distintos).

- Cuando todos los jugadores decidan pasar su turno, se cambiará de fase.

- **Cuando se termina la Fase de Aprendizaje:**
  - Descartad todas las cartas de emprendedores del mercado.
  - Cada jugador puede decidir descartarse cualquier cantidad de cartas de tools de su mano. Como al empezar la Fase de Reclutamiento cada jugador roba **hasta** tener 8 cartas en mano, descartar tools que no te interesan te permite aumentar la probabilidad de robar esas que sí.

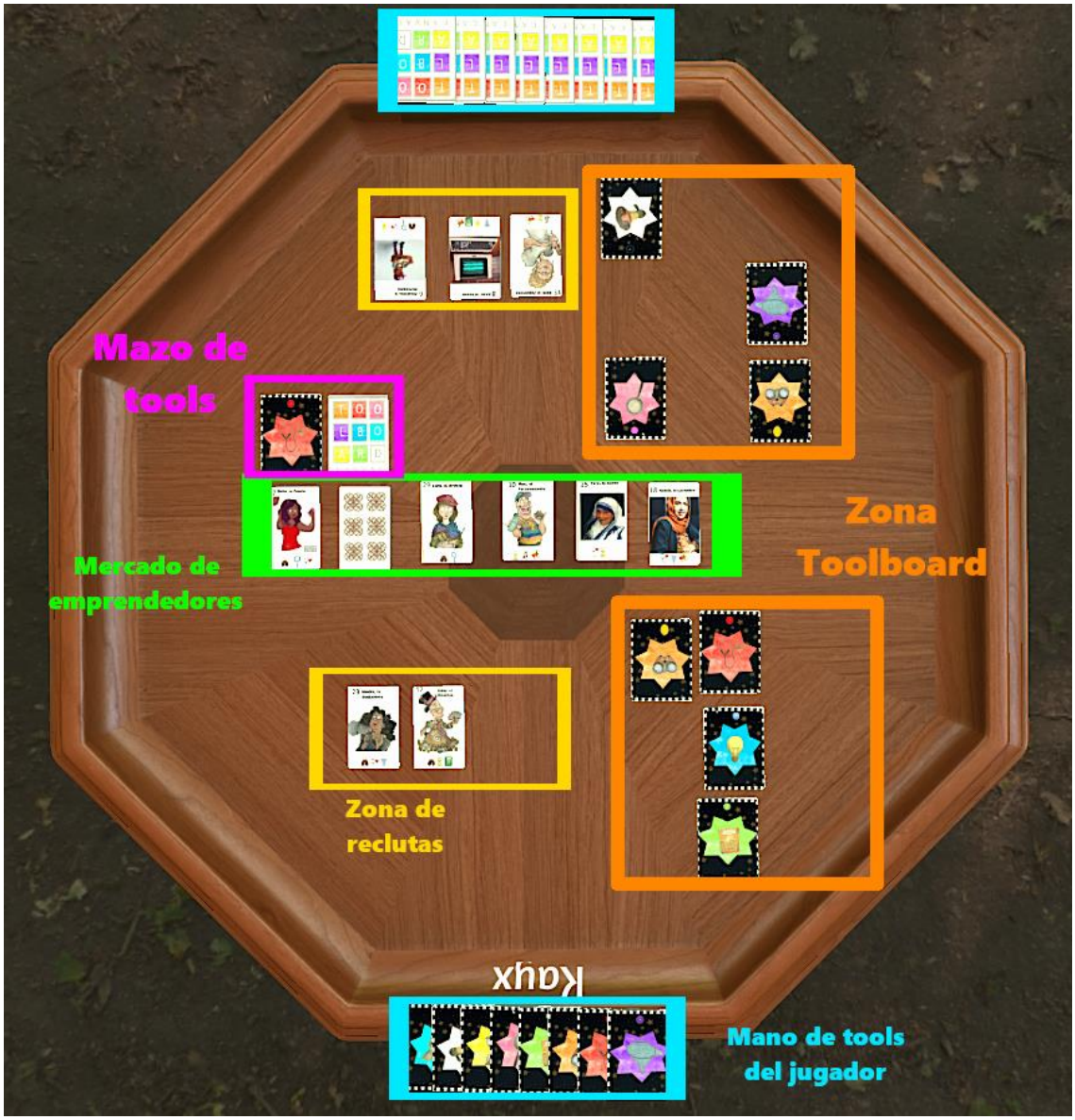

Figura 8. Distribución de la mesa. Fuente: Elaboración Propia.

6.3 Desarrollo del prototipo digital

La segunda mitad del proyecto se centra en el desarrollo del juego de cartas en formato digital, usando Colyseus framework para crear una experiencia multijugador online.

6.3.1 Versión de Colyseus Framework

La versión actual de Colyseus es la 0.14.18 pero el proyecto se ha desarrollado en la versión 0.10.7. La razón principal tras esta decisión es la forma de aprendizaje de realizar código con el framework.

Entre las desventajas del framework se encuentra la documentación. Ésta realmente no está del todo bien estructurada y deja mucha información sin documentar. La versión actual tiene una lista de prototipos que puedes descargar para ver su estructura interna, pero para alguien que aún no entiende el framework se vuelve muy complejo de seguir. Todo este conjunto de factores concluye en que la documentación es densa y poco estructurada para alguien que empieza a usarlo. Hay que recalcar que, sobre todo, la explicación de integración con Unity es escasa[17].

Como la documentación y los ejemplos se han considerado que tienen un alto grado de complejidad para seguir, se ha decidido pasar a la versión 0.10.7, pues de esa versión existe un excelente tutorial que permite tener una mejor visión de qué ocurre en el framework y a partir de ahí adaptarlo a las necesidades del proyecto[18].

El uso de una versión antigua no tiene un impacto negativo real en el proyecto. Las versiones que están por detrás de la 0.10.0 sí se consideran deprecadas, pues el sistema de serialización aun no funcionaba por schemas. En las anteriores versiones se usaba el formato de serialización *"Fossil Delta"*, el cual tiene problemas de cargar la CPU y se consideraba difícil detectar cambios en el estado de la sesión. De la versión 0.10.0 hacia delante siguen consideradas todas las versiones, pues Colyseus tiene documentación de todas esas versiones junto a la última [17].

6.3.2 Estado de la sesión

La sesión de juego debe mantener un seguimiento de los elementos de juego. En el estado de la sesión se declaran las variables públicas que mantiene el servidor. Éstas van a estar sincronizadas con los clientes, por lo que cualquier modificación de esas variables será enviada automáticamente a los clientes conectados a esa sesión.

En el juego, los jugadores tienen toda la información respecto a la situación en la mesa excepto:

- Las cartas que tienen los oponentes, cuyos datos están guardados en local en cada jugador, puesto que solo sirven como elemento de la UI.

- El mazo de cartas de emprendedores. Éste se genera cuando se crea la sesión de juego, se mezcla y se guarda de forma privada en el servidor.

```
import { Schema, type, MapSchema, ArraySchema } from "@colyseus/schema";

export class Player extends Schema {
    @type('int16')
    seat: number; //Asiento asignado al entrar (1 o 2)

    @type('string')
    sessionId: string; //Id de sesión del jugador
}

export class State extends Schema {
    @type({map: Player})
    players: MapSchema<Player> = new MapSchema<Player>(); //Lista de jugadores (la clase de arriba)

    @type('string')
    phase: string = 'waiting'; //Fase de la partida

    @type('int16')
    playerTurn: number = 1; //A que jugador le toca el turno (1 o 2)

    @type('int16')
    winningPlayer: number = -1; //Qué jugador gana la partida (1 o 2)

    @type(['string'])
    cards: ArraySchema<string> = new ArraySchema<string>(); //Qué carta hay en cada posición de la UI
    // 0-2 recruits player 1
    // 3-5 recruits player 2
    // 6-9 shop recruits
    // 10-18 toolboard player 1
    // 19-27 toolboard player 2

    @type('int16')
    playersSkipped: number = 0; //Cuántos jugadores han pasado turno

    @type(['string'])
    recruitsToDestroy: ArraySchema<string> = new ArraySchema<string>(); //Lista de personajes que hay que borrar
                                                                        //(cuando se aprende o se reinicia la tienda)

    @type('boolean')
    firstTurn: boolean = true; //Saber si es el primer turno de la fase de reclutar (para reiniciar la tienda)
}


    @type(['string'])
    recruitsToDestroy: ArraySchema<string> = new ArraySchema<string>(); //Lista de personajes que hay que borrar
                                                                        //(cuando se aprende o se reinicia la tienda)

    @type('boolean')
    firstTurn: boolean = true; //Saber si es el primer turno de la fase de reclutar (para reiniciar la tienda)
}
```

Figura 9. Estado de la sesión. Fuente: Elaboración Propia.

6.3.3 Sesión de juego

Colyseus mantiene las partidas en sesiones de juego, donde cada sesión tiene su propia conexión WebSocket. Una sesión de juego tiene una serie de eventos que se llaman durante su ciclo de vida.

- **OnInit** → Se llama al crear la sesión y le establece el estado inicial de una sesión.
- **OnJoin** → Se llama cuando un jugador entra en la sesión de juego. El servidor crea un jugador y el propio Colyseus le establece una id y un asiento de jugador, que puede ser uno o dos. Lo introduce en el diccionario de jugadores del estado y comprueba cuantos jugadores hay dentro. Cuando se conecta un segundo jugador, la partida empieza.

- **OnLeave** → Se llama cuando un jugador sale de la sesión y lo elimina del diccionario de jugadores del estado.
- **OnMessage** → Se llama cuando un cliente enviar un mensaje al servidor. Este evento se llama cuando el jugador realiza acciones y en ciertas acciones automáticas como reiniciar la tienda o eliminar los personajes usados.

6.3.4 Creación del servidor

Para crear el servidor hay que configurar un servidor express, un pequeño web server que proporciona una infraestructura de aplicaciones web Node.js mínima y flexible. Éste se adjunta a una función de la librería HTTP para crear un servidor de juego.

Al servidor de juego se le atribuye un nombre y un tipo de sesión de juego. En este caso solo es necesario registrar un tipo de sesión, pues solo existe el tipo de la mesa de juego. Por último, al servidor se le atribuye el puerto.

```
import express from 'express';
import { createServer } from 'http';
import { Server } from 'colyseus';
import { GameRoom } from './rooms/game-room';

const port = Number(process.env.PORT || 2567); //Asignamos un puerto
const app = express(); //Configuramos un "servidor express"
                       //(back end web application framework for Node.js)

//Creamos el servidor (Attach WebSocket Server on HTTP Server)
const server = createServer(app);
const gameServer = new Server({
  server,
});

//Registrar el tipo de sesión GameRoom con el nombre 'game' que enviará el cliente
gameServer.register('game', GameRoom);

//Establecer el puerto al servidor
gameServer.listen(port);

console.log(`Listening on http://localhost:${ port }`);
```

Figura 10. Creación del servidor. Fuente: Elaboración Propia.

6.3.5 Conexión Cliente-Servidor

Cuando el cliente quiere realizar una petición al servidor, éste envía un mensaje a través de la sesión de juego, el cual llamará al evento de OnMessage. En estos mensajes también se pueden pasar valores como parámetro para que el servidor los trate.

```
public void RefreshShop (){
        room.Send(new { command = "refreshShop"});
    }

    public void DestroyRecruits(string[][] idCardsDestroy){
        room.Send(new { command = "destroyRecruits", idCardsDestroy});
    }

    public void NullRecruit(string idToNull){
        room.Send(new { command = "nullRecruit", idToNull});
    }

    public void SendRecruited (string cardId,int oldPos, int newPos){
        room.Send(new { command = "recruited", cardId, oldPos, newPos});
    }

    public void SendLearn (string cardId, int posCard, string[] tools){
        room.Send(new { command = "learn", cardId, posCard, tools});
    }
    public void SendSkip (){
        room.Send(new { command = "skip"});
    }
```

Figura 11. Mensajes de cliente a servidor. Fuente: Elaboración Propia.

El servidor, en cambio, tiene dos formas distintas de comunicarse con el cliente, ambas de forma indirecta:

- Cuando un valor del estado de la sesión cambia, el servidor envía automáticamente ese cambio a los clientes.
- Asignando EventHandlers al cliente. Colyseus permite añadir una función al cliente que se ejecutará cuando el estado de la sesión (o algún elemento en específico de éste) sufra alguna modificación.

```
//Se ejectua cuando el estado cambia
state.OnChange += StateChangeHandler

//Se ejecuta cuando se modifica la tabla que ve en
que posición va cada carta
state.cards.OnChange += CardHandling
state.cards.OnAdd += CardHandling

//Se ejecuta cuando se modifica la tabla que ve qué
cartas de personaje hay que borrar de la mesa
state.recruitsToDestroy.OnChange +=
DestroyCardsHandling
state.recruitsToDestroy.OnAdd += DestroyCardsHandling
```

Figura 12. EventHandlers. Fuente: Elaboración Propia.

6.3.6 Acciones del jugador

El jugador puede realizar las siguientes acciones durante su turno:

- Reclutar una carta de personaje (en fase de reclutamiento)
- Aprender de un personaje (en fase de aprendizaje)
- Pasar turno

6.3.6.1 Reclutar carta de personaje

El proceso en que el jugador recluta el personaje es:

0. Condición previa: Estar en fase de reclutamiento y en el turno del jugador.
1. Pulsar sobre un personaje en la tienda.
2. El personaje se marca en rojo, indicando que está seleccionado.
    a. Aparece el botón de OK.

Figura 13. Reclutamiento 1. Fuente: Elaboración Propia.

3. El jugador pulsa sobre una tool de su mano.
4. La tool se marca en rojo, indicando que está seleccionada.
5. Repetir el paso 3-4 hasta que el número de tools coincidan con las del personaje escogido.

Figura 14. Reclutamiento 2. Fuente: Elaboración Propia.

6. Pulsar el botón de OK.
7. El personaje se coloca en la primera posición disponible en la zona de reclutamiento del jugador.
    a. Se eliminan las tools usadas.
8. El turno pasa al siguiente jugador.

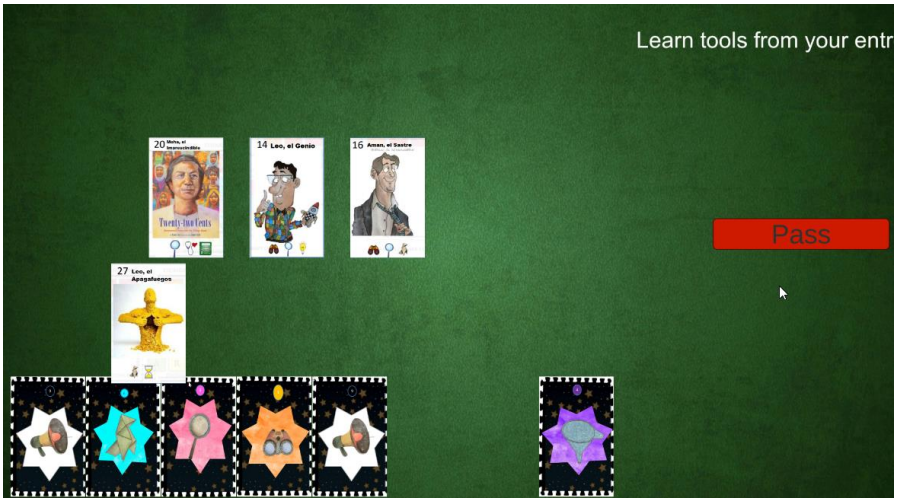

Figura 15. Reclutamiento 3. Fuente: Elaboración Propia.

6.3.6.2 Aprender de un personaje

El proceso en que el jugador aprende de un personaje es:

0. Condición previa: Estar en fase de reclutamiento y en el turno del jugador.
1. Pulsar sobre un personaje de tu zona de reclutamiento.
2. El personaje se marca en rojo, indicando que está seleccionado.
    a. Aparece el botón de OK.
3. El jugador pulsa sobre una tool de su mano que coincide con las tools del personaje.
4. La tool se marca en rojo, indicando que está seleccionada.

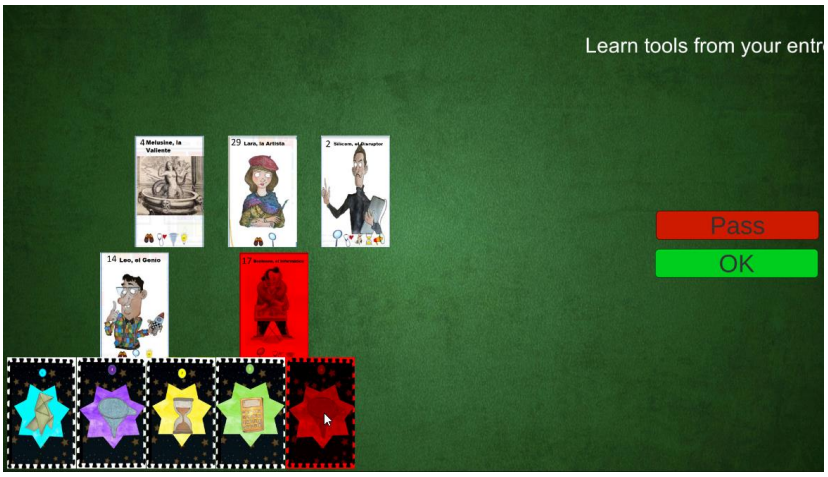

Figura 16. Aprendizaje 1. Fuente: Elaboración Propia.

5. El jugador puede decidir
    a. Pulsar el botón de OK
    b. Volver al paso 3

6. Al pulsar el botón OK, esas tools se mueven a la zona de TOOLBOARD en la casilla correspondiente.
    a. El personaje se elimina.
7. Se pasa al siguiente turno.

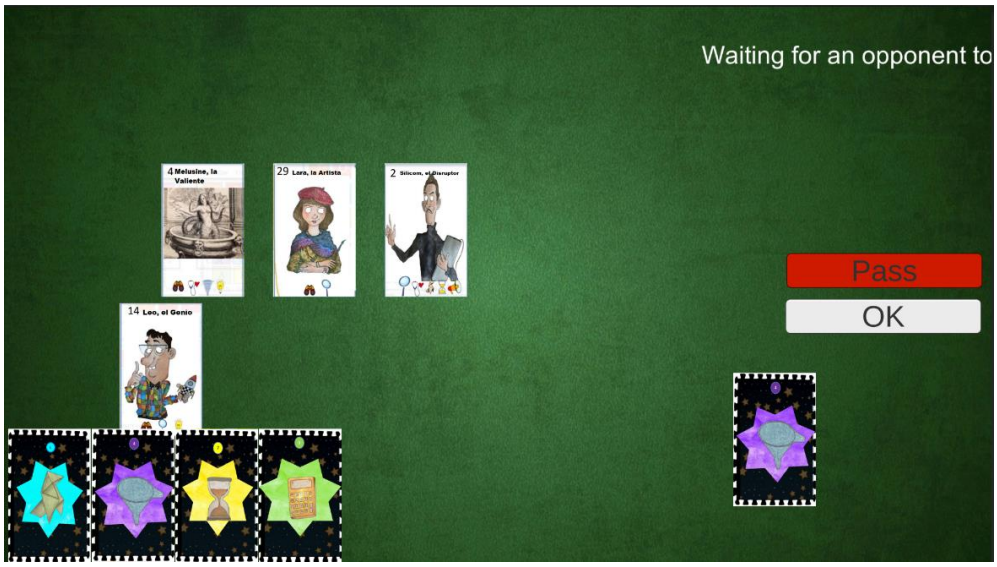

Figura 17. Aprendizaje 2. Fuente: Elaboración Propia.

6.3.6.3 Pasar turno

El proceso en que el jugador aprende de un personaje es:

0. Condición previa: Estar en el turno del jugador.
1. Pulsar sobre el botón de PASS
2. Se pasa al siguiente turno.
3. Si el otro jugador también pasó turno, se cambia de una fase a otra.

6.3.7 Eventos automáticos

6.3.7.1 Empezar fase de reclutamiento

Cada vez que un jugador entra en la fase de reclutamiento ocurre lo siguiente:

1. Se establece que la fase es “Recruit”.
2. El cursor se activa.
3. El cliente instancia hasta ocho cartas aleatorias en los huecos de mano vacíos.

4. Si es el primer turno de reclutamiento, es decir, es el primer turno de la partida o es el primer turno de reclutamiento tras una fase de aprendizaje, se realiza una petición al servidor para que elimine las cartas de personaje que han quedado en la tienda desde la última vez que se sacaron. Acto seguido, envía una petición al servidor para que saque cuatro cartas nuevas y las posicione en la mesa.

6.3.7.2 Esperar al otro jugador

Mientras un jugador espera el turno del otro, su ratón se bloquea y se hace invisible. De esta forma evitamos errores que pueda causar ese jugador.

6.3.7.3 Acabar la partida

Cuando un jugador envía una petición de aprender, el servidor, tras comprobar que las tools a descartar son correctas y colocarlas en la zona de TOOLBOARD, revisa si ese jugador tiene un total de nueve tools en la zona. De ser así, el servidor cambia la fase de la partida a “Result”, el cual muestra un mensaje de victoria o derrota al jugador correspondiente y termina la partida.

6.3.7.4 Entrar en una partida

Cuando un jugador pretende conectarse al servidor, en cliente ocurre lo siguiente:

1. Se crea una instancia singleton del objeto que conecta cliente y servidor
2. Se realiza una conexión al servidor a través del puerto establecido. Para ello, se crea un campo *client* otorgado por Colyseus.
3. Se realiza una conexión a la sesión de juego. Para ello, el cliente debe enviar el nombre del tipo de sala a crear o unirse.
4. El cliente se queda en la escena de espera hasta que se conecte otro jugador. Para ello, tiene un handler que, cuando el servidor modifica el estado de la sesión, le avisa para comprobar si ha entrado otro jugador. Si es así, la escena se cambia a la mesa de juego.

7 Posibles ampliaciones

El diseño de juego siempre está abierto a posibles ampliaciones. Ahora mismo se considera un diseño sólido, pero se seguirá realizando testing hasta la salida del producto. Todo comentario de los testers es considerado para mejorar el producto.

También se plantea una ampliación de diseño como tal. Ha habido ideas para añadir un modo más hardcore para jugadores experimentados o añadir un modo de juego por parejas.

Para la salida del producto también es necesario tener en cuenta que hay que trabajar en el diseño visual de las cartas y el diseño y maquetación del manual de instrucciones.

Para la parte digital del producto, mencionar que se requiere aún de ampliaciones necesarias que no se han podido llevar a cabo en el tiempo establecido. En el momento de entrega de esta memoria el prototipo digital tiene errores que no permiten la finalización de la partida.

8 Conclusiones

La parte de diseño se considera que se ha finalizado de forma exitosa, puesto que cumple con los principios de diseño establecidos y, junto al feedback de los testers, se ha llegado a un producto final de gran calidad.

La parte de prototipado digital, aunque no está del todo finalizada, se considera mayormente un punto positivo. A lo largo del proyecto se ha ido aprendiendo sobre un framework no usado anteriormente, hasta llegar a un punto de entendimiento alto. Además, también se ha seguido aprendiendo sobre Unity, C# y TypeScript.

Colyseus framework ha dado resultados mayormente positivos. Una vez que uno sabe lo que hace, es fácil de configurar y permite realizar conexiones cliente-servidor sin problemas. Aun así, recalcar que la documentación es bastante pobre, por lo que es una difícil barrera de entrada. Aun así, es recomendable 100% y se ha considerado para futuros proyectos.

9 Bibliografía

[1] F. J. Gallego, C. J. Villagrá, R. Satorre, P. Compañ, R. Molina, and F. Llorens, "Panorámica: serious games, gamification y mucho más," vol. 7, no. 2, 2014, [Online]. Available: http://www.byterealms.com/proyectos/gamelearning

[2] R. Dörner, S. Göbel, W. Effelsberg, and J. Wiemeyer, "Serious Games Foundations, Concepts and Practice," 2016.

[3] J. Teodoro, "Manual de trabajo práctico TOOLBOARD."

[4] P. Wilkinson, "Brief history of serious games," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9970 LNCS, pp. 17–41. doi: 10.1007/978-3-319-46152-6_2.

[5] D. Djaouti, J. Alvarez, J.-P. Jessel, and O. Rampnoux, "Origins of Serious Games," 2011.

[6] W. Westera, "How people learn while playing serious games: A computational modelling approach," *Journal of Computational Science*, vol. 18, pp. 32–45, Jan. 2017, doi: 10.1016/j.jocs.2016.12.002.

[7] J. L. Plass, B. D. Homer, and C. K. Kinzer, "Foundations of Game-Based Learning," *Educational Psychologist*, vol. 50, no. 4, pp. 258–283, Oct. 2015, doi: 10.1080/00461520.2015.1122533.

[8] "Card life," Jun. 2014, Accessed: Feb. 01, 2022. [Online]. Available: https://unity.com/es/case-study/hearthstone

[9] J. Glegg, "BRINGING FEATURES TO LIFE IN LEGENDS OF RUNETERRA," Nov. 2019, Accessed: Feb. 01, 2022. [Online]. Available: https://technology.riotgames.com/news/bringing-features-life-legends-runeterra

[10] "Itch.io Most used Engines." https://itch.io/game-development/engines/most-projects (accessed Feb. 03, 2022).

[11] L. P. Chitra and R. Satapathy, "Performance Comparison and Evaluation Of Node.Js And Traditional Web Server (IIS)".

[12] C. Politowski, L. Fontoura, F. Petrilo, and Y. Guéhéneuc, "'Are the Old Days Gone? A Survey on Actual Software Engineering Processes in Video Game Industry', the 5th International Workshop on Games and Software Engineering," pp. 22–28, May 2016.

[13] J. Koutonen and M. Leppänen, "'How are Agile Methods and Practices Deployed in Video Game Development? A Survey into Finnish Game Studios', International Conference on Agile Software Development," pp. 135–149, Jun. 2013.

[14] A. Mathur and A. Acharya, "A Comparative Study on Utilization of Scrum and Spiral Software Development Methodologies: A Review," Nov. 2015. [Online]. Available: www.ijert.org

[15] D. Prasetya Kristiadi, F. Sudarto, D. Sugiarto Fakultas Sains dan Teknologi, R. Sambera, H. Leslie Hendric Spits Warnars, and K. Hashimoto, "Game Development with Scrum methodology," 2019.

[16] E. Ham, "Tabletop Game Design for Video Game Designers," 2016.

[17] " Colyseus & Arena Cloud Documentation."

[18] Moby, "Let's Code | Online Multiplayer Web Game (Part 2 of 2 : Game Server)," 2019.