Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

GENERALITAT VALENCIANA
Conselleria d'Educació, Investigació, Cultura i Esport

IES Mare Nostrum

Rythm Rush!

Proyecto de fin de grado

Luis Escolano Piquer

Tutor: Jesús García Navarro
Curso: 2º DAM
IES Mare Nostrum

Índice

1. Introducción

En este apartado haremos una breve descripción tanto de nuestra idea de negocio como del contenido de la memoria.

1.1. Nuestra idea

Nuestra empresa ofrece un videojuego arcade en dos dimensiones donde simularemos tocar un instrumento al ritmo de la música.
Lo que buscamos es que nuestros usuarios tengan una experiencia que los traslade a un punto similar al de la época de las máquinas recreativas, donde podías ser un jugador más casual y echar unas partidas con tus amigos o incluso competir por ser el mejor en ese videojuego.

1.2 ¿Por qué elegirnos?

Gracias a nuestro sistema de creación de niveles, nuestros usuarios podrán disfrutar de una gran variedad dentro del juego, lo que hará que la experiencia no resulte monótona y dará pie a ser creativos y competitivos.

1.3 Contenido de la memoria

Podemos decir que esta memoria constará de dos partes diferenciadas: la parte empresarial y la parte técnica.
En la parte empresarial, nuestro objetivo será analizar el proyecto desde un punto de vista comercial, haciendo un análisis de varias partes entre las cuales constaran aspectos como el estudio de mercado, marketing, forma jurídica, recursos humanos, análisis de costes, inversión inicial, fuentes de financiación y viabilidad económica del proyecto.

En cuanto a la parte técnica, analizaremos el proyecto desde un punto de vista técnico. Esto requerirá un análisis de varias áreas, que definirán cómo debe ser hecho el proyecto y nos servirá como guía a la hora de desarrollarlo. Este incluirá apartados como el análisis de requisitos, diseño de la aplicación, implementación y pruebas.

Finalmente haremos un resumen con nuestras conclusiones sobre el proyecto, haciendo énfasis en los puntos más importantes e impresiones personales. También se incluirán posibles mejoras sobre el proyecto ya que, la memoria en sí misma está planteada sobre el mínimo producto viable.

2. Estudio de mercado

En este punto analizaremos los aspectos más importantes sobre nuestra empresa, como dónde nos posicionamos en el mercado, posibles competencias, puntos fuertes y débiles, tamaño, estructura, etc...

Analizar todos estos puntos es muy importante de cara a ser realistas con los resultados que esperamos obtener, ya que nos permitirán hacernos una idea de cómo estamos posicionados en el mercado y tomar decisiones acordes a nuestra posición.

2.1. Contexto del sector productivo

En este apartado, haremos un ejercicio en el que posicionaremos nuestra empresa en su sector correspondiente, analizando el macroentorno de este.

En primer lugar, partimos de que pertenecemos a un sector muy definido, que será el sector de los videojuegos. Este sector, pese a lo que pueda parecer, es muy amplio y tiene muchas ramas dentro de él.

El sector de los videojuegos es uno de los más dinámicos y en rápido crecimiento dentro de la industria del entretenimiento. Podemos ver que desde 2001, donde la industria mundial de videojuegos a penas valía 20.000 millones de dólares, ha crecido en valor hasta alcanzar los casi 250.000 millones de dólares. Asimismo, la previsión apunta a que, en 2030, alcanzará un valor de 680.000 millones.

En el siguiente gráfico podemos visualizar cómo ha transcurrido el crecimiento del sector de los videojuegos con respecto a otros similares dentro del sector del entretenimiento, como pueden ser el sector de la música y el sector de las películas.

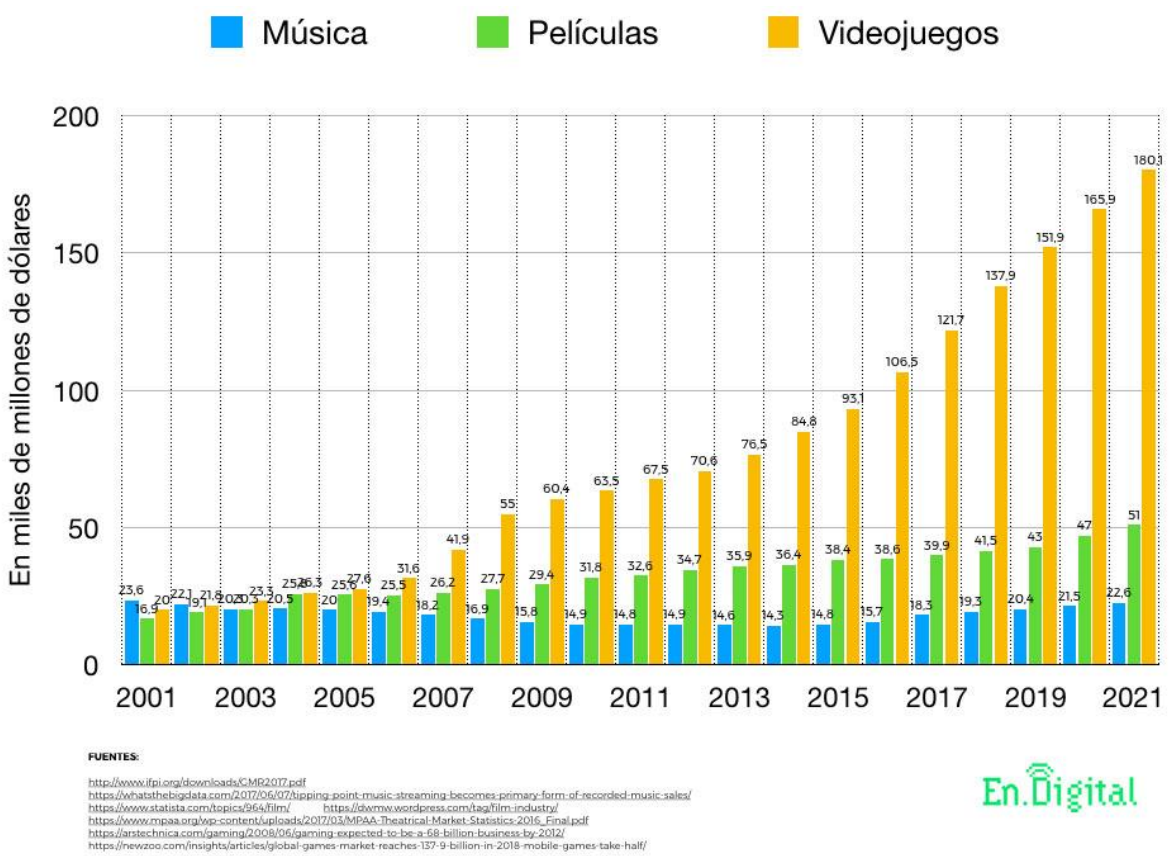


Figura 1. Crecimiento del sector de los videojuegos (Quora, 2021)

La estructura del mercado está dominada por grandes compañías como Electronic Arts, Ubisoft o Activision, aunque hay una vertiente donde un creciente número de desarrolladores independientes están ganando reconocimiento gracias a plataformas de distribución digital como Steam y consolas de nueva generación.

En cuanto a las diferentes tendencias que están afectando al sector, podemos encontrar varios datos interesantes. Para poder diferenciar bien los sectores, enumeraremos algunos puntos de cada tendencia.

Tendencias tecnológicas:

- La reciente adopción de la realidad virtual y aumentada está creando nuevas oportunidades para experiencias inmersivas.
- El crecimiento del juego en la nube está permitiendo a los jugadores acceder a juegos de alta calidad sin necesidad de hardware costoso.
- El crecimiento exponencial del videojuego en smartphones representa una parte significativa del sector, donde se ha conseguido una audiencia global junto con nuevos modelos de negocio.

Tendencias demográficas:

- La audiencia de los videojuegos cada vez es más diversa. No sólo los jóvenes juegan, sino que hay un aumento significativo de su uso en todas las edades y géneros.

Tendencias económicas:

- Con la expansión del sector, la oferta ya no incluye juegos con precios prohibitivos, lo que nos permite disfrutar de muchos juegos a precios competitivos e incluso gratuitos.
- Los nuevos modelos de negocio, como las microtransacciones, benefician tanto al usuario como al desarrollador, dando la posibilidad al mismo de abaratar el precio del producto.

Tendencias sociales:

- Con el crecimiento de nuevas formas de entretenimiento como Youtube o Twitch.tv, donde los creadores de contenido llegan a audiencias masivas, se ha aumentado la visibilidad y el interés sobre el sector de los videojuegos.
- Las comunidades de jugadores en internet han transformado la interacción de las personas en los videojuegos. Mediante estas comunidades la gente puede compartir contenido que les gusta, formar grupos para jugar online o enterarse de eventos relacionados con videojuegos, entre otras.

2.2 Análisis de competencia y DAFO

Centrándonos más en lo que nos afecta a nosotros como empresa, es importante analizar cómo podemos competir en un mercado tan amplio como es el de los videojuegos. Para ello, analizaremos posibles competidores, nuestras debilidades, fortalezas y oportunidades con respecto a otras empresas similares.

Competidores

Como hemos comentado, hay multitud de competidores en la industria del videojuego. Para ello, analizaremos en qué punto exacto nos encontramos y cómo podemos competir.
Si miramos de lejos, veremos que cada día se lanzan una media de 39 juegos en Steam, la plataforma que usaremos para distribuir nuestro producto.

Ésta cifra puede parecer impresionante, pero vamos a segmentar un poco los datos y veremos que es completamente viable entrar a esta competencia.
En primer lugar, hay que diferenciar entre tipos de desarrolladores. A grandes rasgos, podemos definirlos en dos campos: grandes empresas y desarrolladores independientes.
Por lo general, las grandes empresas se centran en desarrollos más grandes y costosos, que llegan al público general y generan impacto en el sector.

Estas empresas no pueden ser consideradas nuestra competencia, ya que ni tenemos recursos para ello ni nuestro producto está orientado de esa forma.

Por otro lado, están los desarrolladores independientes, donde entraríamos nosotros. En este lado, encontramos que el impacto de estos pequeños desarrolladores es cada vez mayor en la industria.

El éxito de estos desarrolladores independientes, por lo general, es debido a que no compiten con los juegos "triple A" de las grandes empresas. Por lo general, son juegos de un género específico, donde implementan mecánicas innovadoras o simplemente hacen bien su trabajo.

La tendencia de las grandes empresas es no arriesgar demasiado en los desarrollos, por lo que este peso recae en los desarrolladores independientes, que suelen tener mucho menos que perder y pueden ser más creativos en el desarrollo y aportar algo de frescura a un mercado que está saturado de calcos.

Por otro lado, está el gusto personal de los jugadores. Dentro de los videojuegos existen muchos géneros y mezclas de géneros. Dentro de estos géneros, podemos encontrar infinidad de estos, como pueden ser los juegos de plataformas, arcade, shooter, etc. Cada uno de los géneros tiene un nicho marcado de jugadores.

Alguien a quien le gusten los juegos de plataformas no tiene por qué disfrutar de un juego de disparos. En nuestro caso, nuestro nicho son los juegos arcade. En concreto, el "nicho dentro del nicho", serían los juegos rítmicos.

Algunos de los juegos rítmicos más populares actualmente, y que podríamos considerar competencia directa serían:
- OSU! / OSU! Manía (desarrollado por Dean Lewis Herbert)
- SOUND VOLTEX (desarrollado por Bemani)
- Muse Dash (desarrollado por PeroPeroGames)

¿Podrían sustituir nuestro producto?

En el mundo de los videojuegos siempre corremos el riesgo de que alguien haga una copia de nuestro juego. Por eso, intentamos tener sistemas o apartados diferenciadores del resto, como diferentes mecánicas o apartado gráfico.

En nuestro caso, intentamos marcar la diferencia con nuestro creador de niveles, ranking online, mecánica donde se introduce el ratón como elemento principal y apartado gráfico.

¿ Quién nos lo compraría?

Por un lado, tenemos a jugadores casuales que realmente no entran en este nicho. Estos jugadores buscan un juego donde entretenerse un rato sin demasiadas pretensiones y jugar partidas cortas. El género arcade y en concreto, lo juegos rítmicos, suele ser una buena salida para este tipo de jugadores.

Por otro lado, como hemos comentado anteriormente, apuntamos a un nicho muy concreto, que es el de los jugadores que juegan a juegos rítmicos.

Las comunidades dedicadas a estos juegos, pese a no ser las más grandes, son muy activas y suelen ver con muy buenos ojos los nuevos lanzamientos. Además, suele ser gente muy proactiva y creativa, lo que encaja a la perfección con el concepto del creador de niveles.

Por otro lado, son jugadores que disfrutan de competir contra otros, lo que también encaja a la perfección con el concepto de nuestro ranking, además de asegurarnos que van a jugar asiduamente.

También cabe destacar que los juegos arcade no tienen una meta final, más que la que se marquen los propios jugadores, por lo que podemos decir que “no se acaban nunca”. Esto sumado al creador de niveles hará que nuestro juego tenga una vida larga y sea un valor seguro para los jugadores.

DAFO

El análisis DAFO nos permitirá detectar con antelación las principales debilidades, amenazas, fortalezas y oportunidades de nuestro proyecto y, por lo tanto, en qué deberíamos centrarnos a la hora del desarrollo.

DAFO: Rythm Rush!

Descripción del DAFO:

Matriz de factores

Figura 2. Análisis DAFO. Realizado por el alumno

2.3 Segmentación del mercado

En este apartado explicaremos los criterios de segmentación que hemos utilizado y qué modelo de negocio utilizaremos. De esta manera podremos saber cuál es nuestro cliente ideal y como debemos enfocarnos en él.

Como hemos mencionado con anterioridad, nuestro videojuego va dirigido a un nicho muy concreto, que es el de los jugadores de juegos de ritmo, así que utilizaremos la estrategia de segmentación concentrada.

Esta estrategia nos permitirá captar la atención de nuestros potenciales clientes de la forma que menos recursos requiera.

Partiendo de nuestro estudio de mercado, tenemos varios tipos de segmentación:

- Segmentación demográfica: personas entre 15 y 25 años. Por lo general, son estudiantes.
- Segmentación psicográfica: personas con tendencia a ser competitivas.

- Segmentación por comportamiento: jugadores habituales, que sean activos en la comunidad.

Modelo de negocio

Nuestro modelo de negocio es el de BTC (business to consumer), lo que implica que nosotros(empresa) vendemos un bien o servicio a un cliente.

En nuestro caso hemos optado por un modelo de juego free-to-play, lo que permitirá a los clientes poder jugar gratuitamente, con algunas limitaciones.

Por otra parte, venderemos un software como servicio (SaaS), que brindará mejoras sobre la versión gratuita del juego.

De esta manera, los jugadores más comprometidos verán mejorada su experiencia y tendrán ventajas sobre los más casuales.

2.4 Ubicación

Debido a que en un principio habría un solo trabajador, la ubicación de trabajo elegida es un coworking en el centro de Alicante.
Es una opción muy económica (90€ al mes) que cubre las necesidades que podamos tener.
El precio incluye:

- Espacio de trabajo
- Conexón a internet
- Horario de 8:30 a 20:30
- Sala de reuniones
- Coffe break
- Baño
- Patio

3. Marketing

Es importante tener un buen plan de marketing, donde definiremos qué es lo que estamos vendiendo y cómo lo vamos a vender.

3.1. Producto o servicio

En nuestro caso, podríamos decir que tenemos ambas.
Por un lado, nuestro producto, el juego en sí, será gratuito. Esto permitirá eliminar la barrera de entrada de nuestro cliente ideal.
Por otro lado, nuestro servicio, que sí venderemos, estará centrado en mejorar el producto.
La versión free-to-play contendrá un paquete de canciones y un personaje. Además, sus descargas de niveles estarán limitadas diariamente, y su uso del ranking será limitado a la vista personal y global.

Para aquellos jugadores que deseen disfrutar de estas ventajas, ofreceremos una suscripción que brindará las siguientes mejoras:

- Descargas de niveles ilimitadas.
- Uso completo del ranking (ej. Entre amigos de Steam o a nivel nacional).
- Desbloqueo progresivo de nuevos personajes.

3.2. Precio

Como hemos indicado, el producto constará de una parte gratuita y un servicio de pago.

Los jugadores que deseen obtener las mejoras del servicio de pago podrán hacerlo mediante una suscripción. Dicha suscripción tendrá dos opciones de pago:

- 4€ de forma mensual.
- 40€ de forma anual.

Consideramos estos unos precios razonables y asequibles para la gente que realmente le de uso al juego.

De esta forma, quien quiera probar el juego podrá hacerlo de manera gratuita y si realmente va a hacer uso de este, puede pagar la mensualidad.

Cada cierto tiempo se incluirán actualizaciones, como diferentes personajes, que incentiven a la gente a comprar, al menos, una mensualidad del mismo para poder ser obtenidos.

3.3. Promoción

En nuestro caso, tenemos varias vías por las que promocionarnos:

- Comunidades de otros juegos.
- Página de Steam.
- Redes sociales.
- Youtubers / creadores de contenido.

Tras investigar los diferentes canales de promoción, hay que aclarar que hay algunas acciones que deben tomarse antes del lanzamiento del juego. Todas las siguientes acciones pueden empezar desde antes del lanzamiento, así que siempre habrá que aclarar la fecha de salida. También, habrá que hacer lo posible para que la gente ponga nuestro juego en su lista de deseados, en pos de tener más repercusión.

Básicamente, la plataforma donde vamos a lanzar el juego tiene un algoritmo en el cual recomienda el juego en base a una "lista de deseados" de la misma plataforma. Cuanta más gente tenga el juego en su lista de deseados, mayor será el apoyo de la plataforma el día de salida.

Esto nos lleva a que debemos empezar la promoción del juego como mínimo dos meses antes del lanzamiento.

Las comunidades de otros juegos de ritmo son, a parte de gratuitas, una forma genial de captar la atención de nuevos clientes. En estas comunidades encontraremos miles de personas interesadas en el producto ofrecido, así que no podemos dejar pasar la oportunidad de promocionarnos por estas vías.

En cuanto a las redes sociales, es gratuito crear un perfil en ellas y publicar actualizaciones con respecto al juego. La gente podrá seguirnos y podremos tener un feedback directo de los usuarios e interactuar con ellos.

En cuanto a los youtubers y similares, nos vamos a centrar en los que centren su contenido en juegos rítmicos. Apuntar a personas con millones de seguidores solo porque tienen millones de seguidores lo único que hará será quemar nuestro presupuesto.

Por contrario, si apuntamos a creadores del nicho en el que nos queremos centrar, tenemos dos beneficios:

- Promoción mucho más barata o incluso gratuita.
- Los consumidores de ese tipo de contenido estarán potencialmente interesados en nuestro juego.

En cuanto al presupuesto para hacer marketing, hemos decidido destinar unos 2000€ en el mismo, ya que únicamente nos costará dinero el posible pago a youtubers e influencers.

3.4 Distribución

En nuestro caso, sólo vamos a vender un producto digital, que será vendido a través de Steam. Para poder publicar nuestro juego en Steam, deberemos pagar 100$, que recuperaríamos en caso de llegar a los 1000$ en ventas. En nuestro caso, no llegaremos ya que nuestro producto será gratuito y las compras estarán dentro del propio juego.
Del mismo modo, no tendremos que preocuparnos por el 30% que nos cobra Steam por cada venta de nuestro juego.

Para nuestras mensualidades, utilizaremos PayPal como pasarela de pagos. La tarifa de paypal en estas circunstancias (considerado micropagos) es del 5% + 0.05€ por transacción. En caso de micropagos internacionales será del 6% + tarifa fija dependiendo de la moneda de cada país.
Para futuros cálculos usaremos como referencia el 6% + 0.05€. En nuestro caso, por una venta de 4€, nos cobrarían 0.29€ por venta, dejandonos a nosotros con 3.71€.

4. Forma jurídica

Dadas las circunstancias en las que nos encontramos (un solo trabajador, sin una gran inversión), consideramos que la mejor opción para arrancar es la de empresario individual, es decir, trabajador autónomo.

Debido a que somos autónomos, el único trámite que debemos llevar a cabo es el de darnos de alta en la seguridad social.

Teniendo todo esto en cuenta, los gastos asociados a la constitución y puesta en marcha de nuestra empresa serían:

- 80€ Tarifa plana para autónomos abonado a la Seguridad Social.
- IRPF correspondiente.

5. Recursos humanos

Tal como hemos comentado en el apartado anterior, no dispondremos de más trabajadores que el propio empresario, que tendrá condición de autónomo.

Para el empresario hemos estimado una remuneración mensual mínima con la que poder salir adelante (7000€/año). El mismo tendrá que pagar la tarifa plana de autónomo, que constaría de 960€/año. También deberá hacerse cargo del correspondiente pago del IRPF.

6. Análisis de costes

En este apartado analizaremos los diferentes costes de nuestro producto. Todos los costes serán considerados como fijos, aunque en la realidad no sea así. Todos los costes serán representados de manera anual.

Primero los analizaremos uno a uno, dando detalles sobre cada uno. Después mostraremos una tabla en la que podremos verlos de manera rápida.

- Tarifa plana autónomo: 960€. La suma de la tarifa plana de autónomo de 80€/mes.
- Sueldo propio: 7000€. Es el sueldo propio del empresario. La retribución es mínima debido a la incertidumbre y de los fondos de los que partimos.
- Coworking: 1080€/año.
- Servidores S3: 78,48€/año IVA incl. Tráfico ilimitado, 4GB RAM, Intel ATOM N2800. Consideramos suficientes estos servidores, ya que no esperamos un gran tráfico continuo.
- Promoción: 2000€
- Distribución juego Steam: 100$. Unos 92,16€. Un solo pago.

Costes fijos (total anual)	Euros (total anual)
Tarifa plana autónomo	960€
Retribución empresario	7000€
Coworking	1080€
Servidores S3	78,48€
Distribución Steam	92,16€
Promoción	2000€
TOTAL	11.210,64€

Para calcular el umbral de rentabilidad, debemos tener en cuenta que tenemos dos suscripciones diferentes, la de 4€ y la de 40€. Vamos a calcularlo en base a que de cada 20 suscripciones, 18 son mensuales y 2 anuales. Tenemos que tener en cuenta los costes de la pasarela de pago.

Suscripción de 4€: 4 – 0.29 = 3.71€

Suscripción de 40€: 40 – 2,45 = 37,55€

Quedaría una media por venta de:

((3,71 * 18) + (37,55*2)) / 20 = 7,09€ por venta.

Considerando el precio medio por venta, el umbral de rentabilidad sería:

11210,64 / 7,09€ = 1.582 Unidades vendidas

7. Inversión inicial

El empresario dispone del equipo necesario para realizar las tareas requeridas, como puede ser un ordenador portátil, periféricos, etc. Por lo tanto, concluimos que no es necesario contar con una inversión inicial.

8. Fuentes de financiación

En nuestro caso, no requeriremos de fuentes de financiación externas. Esto es debido a que, gracias a la vida laboral del empresario, este ha podido beneficiarse de la capitalización por la prestación de desempleo. Gracias a estas circunstancias, disponemos de 20.175,25€ con los que podremos financiar nuestro proyecto.

9. Viabilidad económica: Plan económico-financiero

En este apartado, se detallará el plan de tesorería y la cuenta de resultados proyectada para asegurar la viabilidad económica del proyecto. El plan de tesorería incluirá las proyecciones de entradas y salidas de efectivo, mientras que la cuenta de resultados proyectada mostrará los ingresos, gastos y el beneficio neto esperado.

9.1. Plan de tesorería

En este apartado, veremos el total de entradas y salidas de dinero de nuestra cuenta, indicándonos cómo quedará nuestra cuenta después de un año.

ENTRADAS	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE	TOTAL
Aportaciones del empresario	€ 20.175,25	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ 20.175,25
Ventas	€ 4.112,20	€ 3.091,24	€ 1.666,15	€ 2.481,50	€ 1.985,20	€ 2.049,01	€ 1.949,75	€ 2.056,10	€ 2.552,40	€ 2.127,00	€ 1.432,18	€ 1.722,87	€ 27.225,60
TOTAL	€ 24.287,45	€ 3.091,24	€ 1.666,15	€ 2.481,50	€ 1.985,20	€ 2.049,01	€ 1.949,75	€ 2.056,10	€ 2.552,40	€ 2.127,00	€ 1.432,18	€ 1.722,87	**€ 47.400,85**

SALIDAS	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE	TOTAL
Seg. social	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 80,00	€ 960,00
Retribución empresario	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 583,33	€ 6.999,96
Alquiler coworking	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 90,00	€ 1.080,00
Servidores S3	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 6,54	€ 78,48
Distribución Steam	€ 92,16	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ -	€ 92,16
Promoción	€ 1.000,00	€ -	€ -	€ 500,00	€ -	€ -	€ -	€ -	€ 500,00	€ -	€ -	€ -	€ 2.000,00
TOTAL	€ 1.852,03	€ 759,87	€ 759,87	€ 1.259,87	€ 759,87	€ 759,87	€ 759,87	€ 759,87	€ 1.259,87	€ 759,87	€ 759,87	€ 759,87	**€ 11.210,60**

Entradas menos salidas	€ 22.435,42	€ 2.331,37	€ 906,28	€ 1.221,63	€ 1.225,33	€ 1.289,14	€ 1.189,88	€ 1.296,23	€ 1.292,53	€ 1.367,13	€ 672,31	€ 963,00	**€ 36.190,25**
Saldo en el banco c.c.	€ 22.435,42	€ 24.766,79	€ 25.673,07	€ 26.894,70	€ 28.120,03	€ 29.409,17	€ 30.599,05	€ 31.895,28	€ 33.187,81	€ 34.554,94	€ 35.227,25	€ 36.190,25	

9.2. Cuenta de resultados

A continuación, haremos una tabla representando la cuenta de resultados, donde podremos ver si realmente nuestro proyecto es viable o no.

Cuenta de resultados	Año 1	Cuenta de resultados	Año 1
Ingresos de explotación		**Gastos de explotación**	
Ventas/ingresos	€ 27.225,60	Seg. Social	€ 960,00
		Retribución empresario	€ 6.999,96
		Coworking	€ 1.080,00
		Servidores S3	€ 78,48
		Distribución Steam	€ 92,16
		Promoción	€ 2.000,00
TOTAL INGRESOS DE EXPLOTACIÓN	**€ 27.225,60**	**TOTAL GASTOS EXPLOTACIÓN**	**€ 11.210,60**
INGRESOS FINANCIEROS		**GASTOS FINANCIEROS**	
Ingresos financieros	€ -	Intereses préstamo	€ -
TOTAL INGRESOS FINANCIEROS	**€ -**	**TOTAL GASTOS FINANCIEROS**	**€ -**
Resultados de explotación (ingresos de la explotación - gastos de la explotación)	**€ 16.015,00**		
Resultados financieros (ingresos financieros - gastos financieros)	**€ -**		
Resultados antes de impuestos (resultado de la explotación + resultado financiero)	**€ 16.015,00**		

10. Análisis de requisitos

El análisis de requisitos es una parte fundamental del desarrollo de software. Gracias a este análisis podemos plasmar qué queremos que haga nuestro programa y cómo debería comportarse.

Si lo hacemos correctamente, podremos sacar esos pensamientos, que solo son ideas, a una definición real de la aplicación. Esto nos ayudará a darnos cuenta de si lo que tenemos en mente es viable o solo un conjunto de ideas sueltas.

10.1. Requisitos funcionales

Los requisitos funcionales describen acciones concretas, función a función. Los requisitos funcionales que tendrá la aplicación propuesta son los siguientes:

Requisito	Descripción
Opciones	- El usuario puede modificar las opciones del juego
Selección de nivel	- Se muestra una lista con los niveles disponibles - Se muestra una tabla de puntuaciones de ese nivel - Da paso a la pantalla de juego
Jugar	- Se muestra una interfaz de usuario con información relevante - Se muestran varios puntos que sirven para indicar cuándo hay que pulsar una tecla - Aparecen notas que se aproximan a los puntos indicados anteriormente - Pulsando la tecla correspondiente, se evalúa si el intento es correcto y si lo es, el nivel de precisión que se ha tenido - Suena la canción que corresponde con el nivel - Si se falla un determinado número de veces, se pierde el nivel - Se acumula puntuación en base a la precisión - Se puede pausar el juego - Se puede cancelar el nivel y volver a la pantalla de selección de nivel
Crear nivel	- Permite crear niveles de juego - El usuario debe aportar mínimo una canción

	- El usuario puede aportar una imagen de fondo - Permite ajustar el offset en milisegundos del inicio de la canción, de forma que puede ser sincronizada - Permite introducir el BPM de la canción - Permite separar el nivel en dificultades - Hay una pista que indica el tempo de la canción en BPM - Se puede desplazar la pista de tempo - Se pueden poner notas sobre la pista de tempo - Se puede guardar el progreso (local) - Se puede subir el nivel a la base de datos
Obtener puntuación	- Se obtiene y muestra los resultados de la última partida.
Enviar puntuación	- Se envía la puntuación a la base de datos para su almacenamiento.
Personalizador	- Permite elegir personajes
Descargar niveles	- Muestra los niveles disponibles para descargar - Permite descargar niveles y añadirlos a tu colección
Ver ranking	- Muestra el ranking de usuarios - Se puede filtrar por canciones/tipos de usuario
Desbloquear personaje	- Desbloquea el uso de un personaje

10.2. Requisitos no funcionales

Los requisitos no funcionales son los necesarios para que el software pueda operar, pero que el usuario no los ve. En nuestro caso, podríamos decir que los requisitos no funcionales de nuestra aplicación serán los siguientes:

Requisito	Descripción
API Rest	- Permite el CRUD de los datos de jugador y niveles
Servidor de almacenamiento	- Almacenamiento masivo - Permite almacenar/descargar los niveles
Base de datos de juego	- Gestiona la información de lo relativo al juego, como información de niveles, puntuaciones, usuarios...

10.3. Metodología de desarrollo. Fases del proyecto. Tareas y plazos de ejecución

En nuestro caso, hemos escogido una metodología incremental tradicional.

Debido a la naturaleza del proyecto, es muy importante que cada cambio esté correctamente integrado en el sistema, ya que cada acción tiene impacto directo con el usuario.

El modelo incremental es una metodología de desarrollo de software en la que el proyecto se divide en pequeños incrementos funcionales. Cada incremento incluye fases de análisis de requisitos, diseño, implementación y pruebas, entregando una parte del producto que funciona y se puede utilizar. Estos incrementos se desarrollan secuencialmente, mejorando y ampliando el sistema con cada ciclo. Esto permite adaptarse a cambios y proporcionar valor al cliente de manera continua.

Las diferentes fases del modelo incremental pueden ser resumidas de la siguiente manera:

- **Análisis:** en la fase de análisis identificaremos los requisitos iniciales y determinaremos los requisitos para el incremento.
- **Diseño:** Creamos un diseño detallado que guiará la implementación de la funcionalidad que vamos a implementar.
- **Implementación:** en esta etapa desarrollaremos y codificaremos los componentes diseñados para el incremento. El software se construye en pequeñas partes funcionales y se integran progresivamente.
- **Pruebas:** en este apartado verificamos el incremento desarrollado. Se realizan pruebas para asegurar que se cumple con el objetivo inicial y que funciona correctamente, antes de pasar al siguiente ciclo.

Aunque parezca obvio, cabe destacar que si las pruebas no son correctas, pasaríamos del apartado de pruebas a la implementación de nuevo hasta que la funcionalidad a implementar sea correctamente funcional.

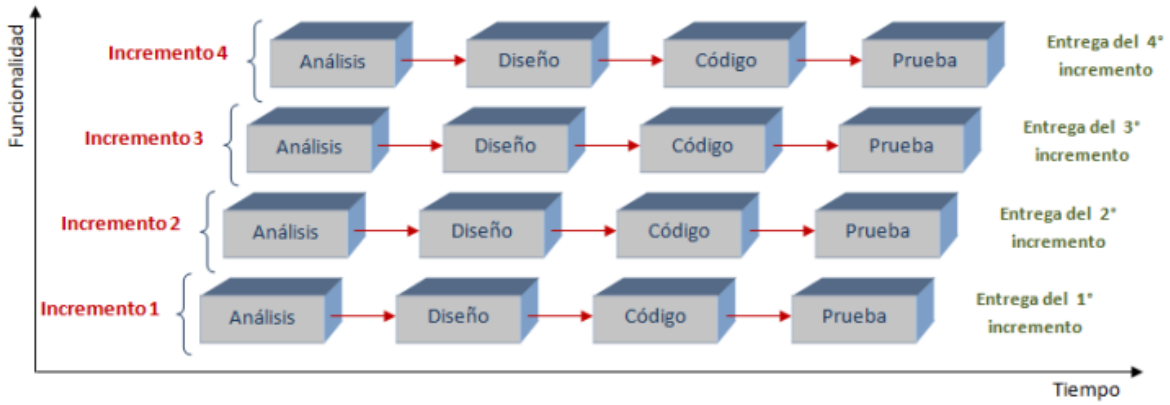


Figura 1: El Modelo Incremental

Figura 3. El modelo incremental (Mwebs, n.d.)

En este tipo de metodología, no se especifica la manera de hacer un seguimiento o de escoger las tareas que debemos hacer. Sin embargo, hay montones de herramientas diferentes para estos fines.

En nuestro caso, hemos decidido usar Trello. Trello es una herramienta que nos permitirá tener un tablero de tareas y hacer un seguimiento de cada tarea, entre otras. Trello es usada tanto por equipos como por desarrolladores individuales. Hay otras herramientas más potentes, pero consideramos que esta se adapta a nuestras características y tipo de proyecto.

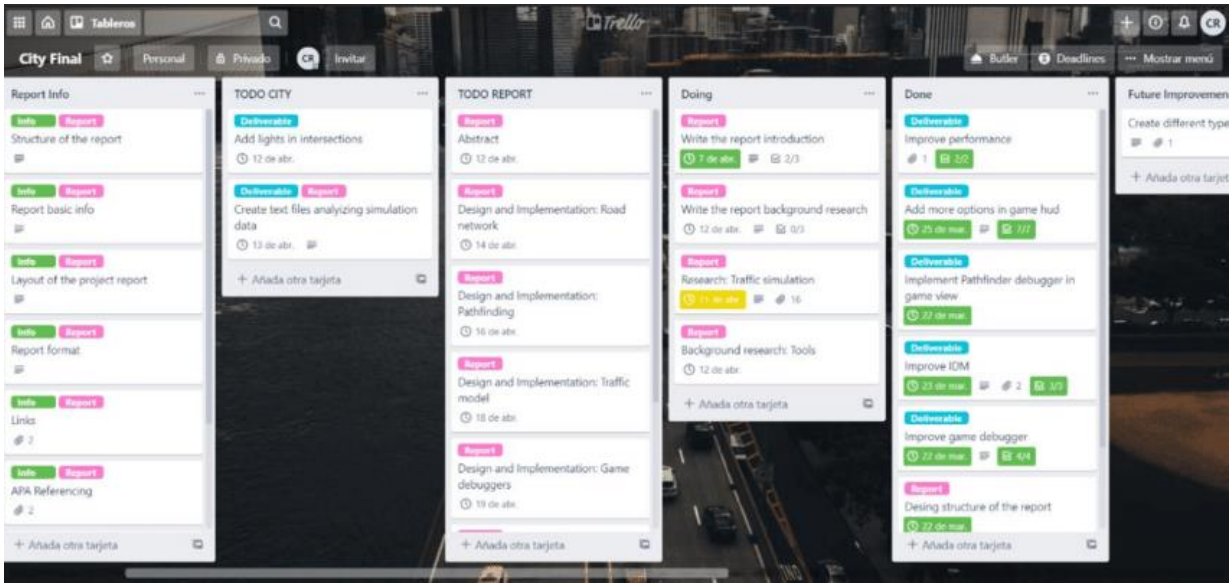


Figura 4. Ejemplo trello. (ResearchGate, 2019)

El uso de esta herramienta permitirá cierta flexibilidad y adaptabilidad a la hora de escoger el siguiente incremento, a la par que podremos ver de forma transparente cómo avanzamos con el proyecto.

A su vez, utilizaremos GitHub como herramienta de control de versiones, lo que nos permitirá gestionar los incrementos de forma individual e integrarlos de manera continua y segura.

Hablando de un mínimo producto viable, que incluya todo lo descrito, creemos que el desarrollo requeriría de un año, contando con que solo trabaja en él una persona a tiempo completo.

En el primer mes, debería desarrollarse un prototipo donde se pueda jugar con las dos mecánicas principales. Esto incluye además de las dos mecánicas, pulir el sistema de lectura de niveles, sistema de puntuaciones, etc... Todo, por su puesto, trabajando en un entorno local.

La herramienta más costosa sin duda es el creador de niveles. Estimamos que para que sea totalmente funcional, harían falta unos tres meses de trabajo, siendo generosos.

A esto le seguiría pulir la parte jugable, el sistema de ranking, y la gestión de niveles.

Llegados a este punto, sería la hora de empezar a integrar nuestros sistemas en la nube. Habría que desarrollar la API REST, el modelo de base de datos y (ya que aún no queremos pagarlo) hacer simulaciones con el creador de niveles (con, por ejemplo, contenedores de docker en su lugar).

Para los últimos meses debemos enfocarnos en encontrar fallos y hacer nuestro juego más “juicy”. Esto quiere decir, mejorar las interacciones con el usuario, añadiendo animaciones, dando feedback con las acciones, añadir sonidos, etc...

11. Diseño

11.1 Diagrama de componentes

Debido a la estructura del proyecto, hemos decidido añadir un pequeño diagrama a modo de "Big Picture" donde se reflejan los componentes relacionados con la aplicación. De esta forma podremos ver de manera más clara la estructura del proyecto antes de entrar en detalles.

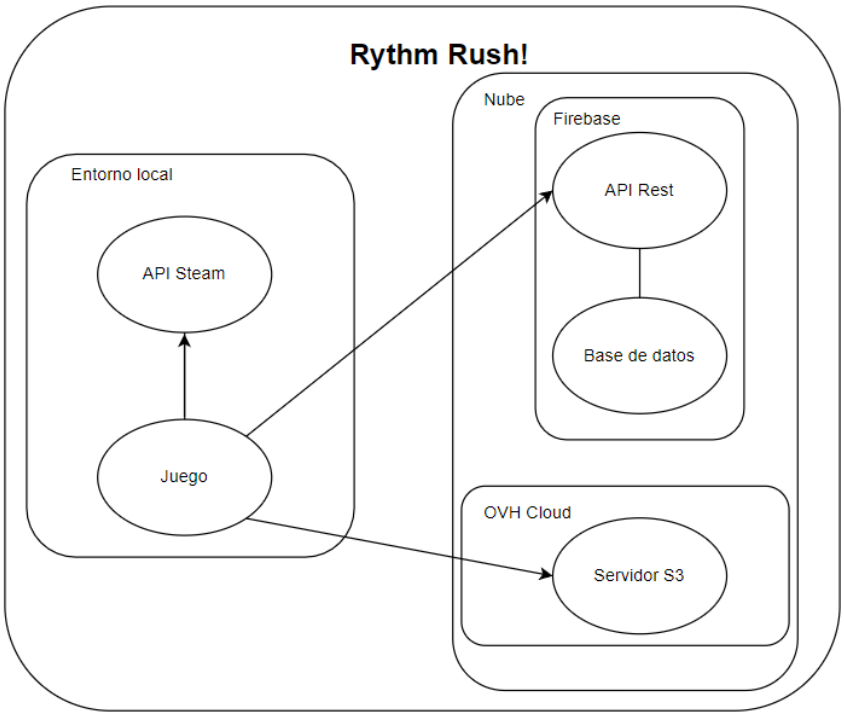


Figura 5. Mapa de componentes. Realizado por el alumno.

En nuestro entorno local estaremos ejecutando steam y el propio juego, mientras que en la nube tendremos varios servicios: nuestra API Rest, la base de datos y el servidor S3.

A modo de resumen:

- El juego hará peticiones a la API de Steam para que nos proporcione los datos del usuario.
- El juego hará consultas a la API Rest y la API rest, a la base de datos. De este modo aseguramos no conectar directamente con la base de datos.
- Desde el juego subiremos y descargaremos los niveles, que están contenidos en el servidor S3.

11.2 Diseño de datos

En este apartado podremos ver tanto el diagrama entidad-relación de las principales entidades como la estructura de los datos en formato JSON en la base de datos (no relacional).

A continuación, definimos una lista con las principales entidades y sus principales atributos, explicando a qué se refiere cada uno, ya que sólo con el nombre podemos caer en un malentendido.

- **Player:**
  - IdPlayer: Id del player, la recibimos desde la API de Steam.
  - Name: Nombre del jugador
  - Premium: Fecha en la que acaba la suscripción premium. Campo en blanco o con fecha anterior a la actual significa que no hay suscripción premium activa.
  - PP: o "Performance Points". Puntuación total del jugador. Es un campo que se calcula en base a las mejores puntuaciones del jugador.
  - Scores: Lista de puntuaciones conseguidas por el jugador.
  - UploadedLevels: Niveles publicados por el jugador.
  - Characters: lista que hace referencia a los personajes desbloqueados.

- **Score:**
  - Misses: número de fallos que ha cometido el jugador en la partida.
  - Date: fecha en la que se ha conseguido la puntuación.
  - Accuracy: precisión del jugador. Es dada según el número de aciertos sobre el total de notas en la partida.
  - MaxCombo: Máximo de notas seguidas pulsadas sin fallar.
  - Player: Jugador que ha conseguido la puntuación.
  - Level: Nivel en el que se ha producido la puntuación.
  - Difficulty: Dificultad concreta en la que se ha conseguido la puntuación.
  - PP: o "Performance Points". Es el valor numérico que damos a los puntos que otorga la puntuación. Es **calculado** en base a la dificultad del nivel y precisión de la puntuación.

- **Level:**
  - Duration: duración del nivel.
  - BPM: o "Beats Per Minute". Indica cuántos "beats" o pulsaciones por minuto tiene la canción. Esto es crucial ya que marcará el ritmo de la canción para poder cuadrarlo con las notas en nuestro nivel.
  - Offset: indica si debemos desplazar en el tiempo las notas puestas en el nivel. Se indica en milisegundos. Nos servirá para cuadrar el sonido con las acciones del jugador.
  - DownloadLink: Link de descarga del nivel. Apuntará a nuestro servidor S3, donde está el contenido del nivel (canción e información del mismo).
  - Name: nombre del nivel.

- **Difficulty:**
  - NJS: o "Note Jump Speed". Este valor determinará cómo de rapido se mueven las notas en la partida.
  - Stars: Valor numérico en el que representamos la dificultad.
  - Level: A qué nivel pertenece esta dificultad.
  - Notes: Lista de notas de esta dificultad.

- **Note:**
  - Position: se refiere al input que hay que pulsar para interactuar con ella (boton del raton o tecla). Irá de 1 a 4 dependiendo de cual input deseemos.
  - BPM: es la referencia en BPM al tiempo en el que debe pulsarse la nota.
  - Duration: En caso de ser 0, la nota será una nota "normal", donde solo habrá que pulsar. En caso de ser más, la nota será un "Slider", donde tendremos que mantener pulsado el input durante un tiempo.

11.2.1 Entidad - Relación

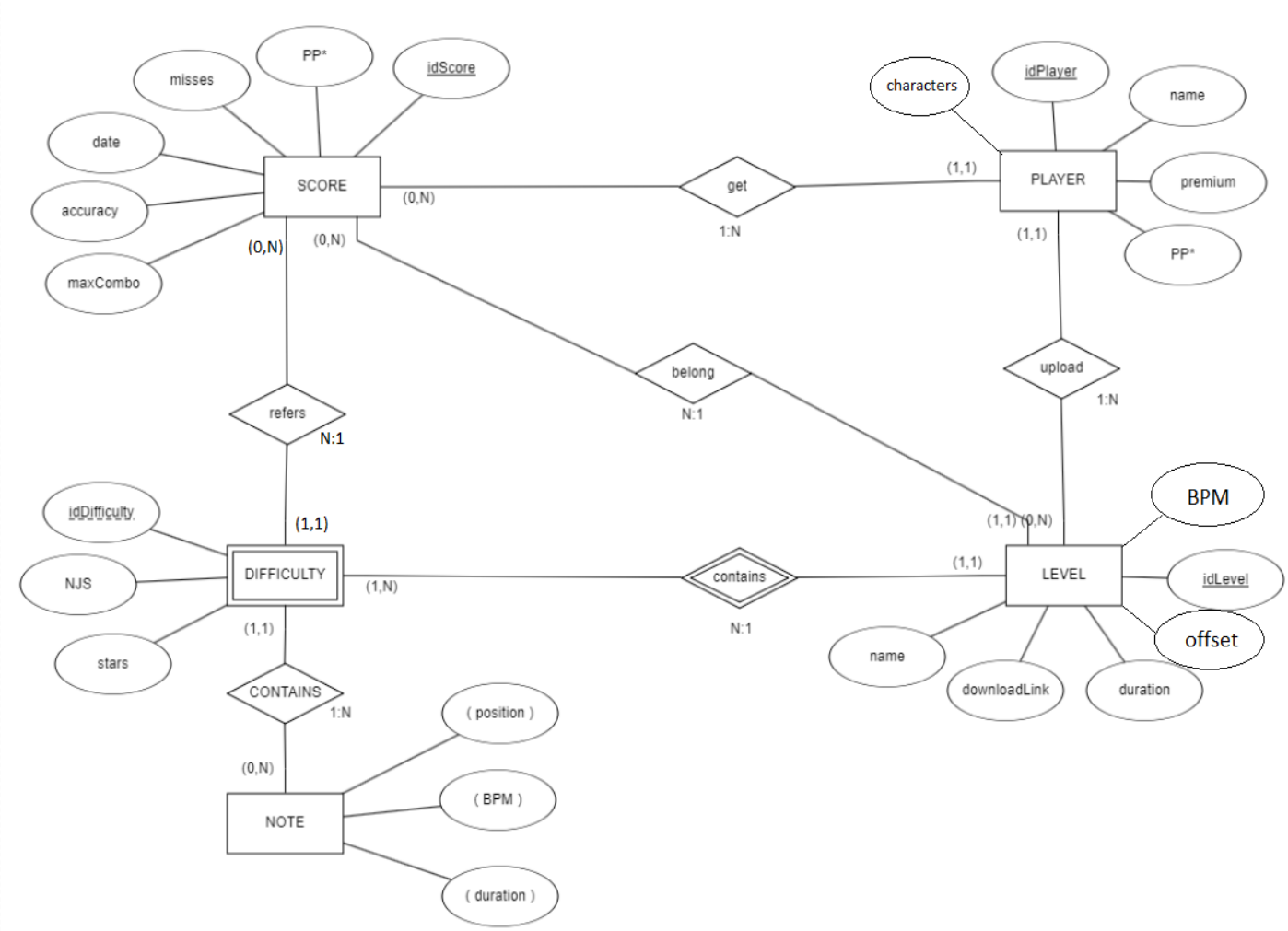


Figura 6. Diagrama entidad-relación. Realizado por el alumno.

11.2.2 Estructura de la base de datos

A continuación, mostraremos la estructura de los archivos JSON de las principales entidades que utilizaremos en nuestra base de datos.

Player

```
{
    "Player": {
        "IdPlayer": string,
        "Name": string,
        "Premium": date,
        "PP": float,
        "UploadedLevels": [string],
        "Scores": [string],
        "Characters": [int]
    }
}
```

Score

```
{
    "Score": {
        "idScore": string,
        "PP": float,
        "Misses": int,
        "Date": date,
        "Accuracy": float,
        "MaxCombo": int,
        "Player": string,
        "Level": string,
        "Difficulty": string
    }
}
```

Level

```
{
    “Level”: {
        “IdLevel”: string,
        “Duration”: float,
        “DownloadLink”: string,
        “Name”: string,
        “Author”: string,
        “Difficulties”: [string]
    }
}
```

Difficulty

```
{
    “Difficulty”: {
        “IdLevel”: string,
        “IdDifficulty”: string
        “Stars”: float
        “NJS”: float,
        “Notes”: [
            [int, float, float]
        ]
    }
}
```

La entidad “Note” no requerirá ser subida a la base de datos, ya que se representa dentro de la entidad “Difficulty”, con una lista que contiene los valores de las diferentes notas del nivel.

11.3 Diseño funcional

11.3.1 Diagrama de clases

Tener un diagrama de clases nos permitirá plantear cómo vamos a gestionar los diversos objetos en nuestra aplicación y cómo se relacionan entre ellos. A continuación, se mostrará un diagrama con los elementos más importantes de nuestro juego:

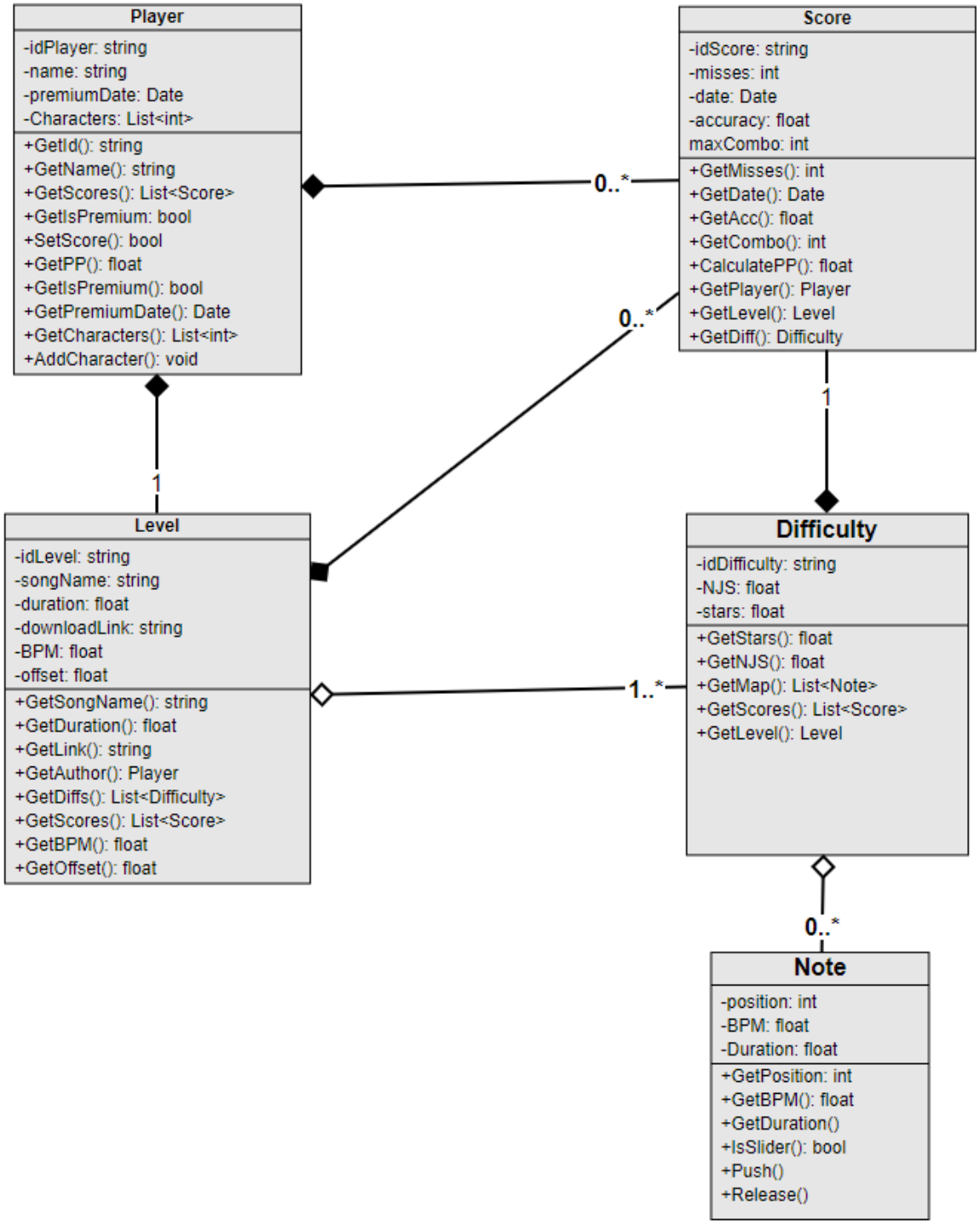


Figura 7. Diagrama de clases. Realizado por el alumno.

11.3.2 Diagrama de casos de uso

Tener un diagrama de casos de uso nos permitirá tener claras las interacciones entre el usuario y la aplicación, describiendo las funcionalidades desde el punto de vista del usuario. A continuación, se mostrará dicho diagrama orientado a nuestra aplicación propuesta.

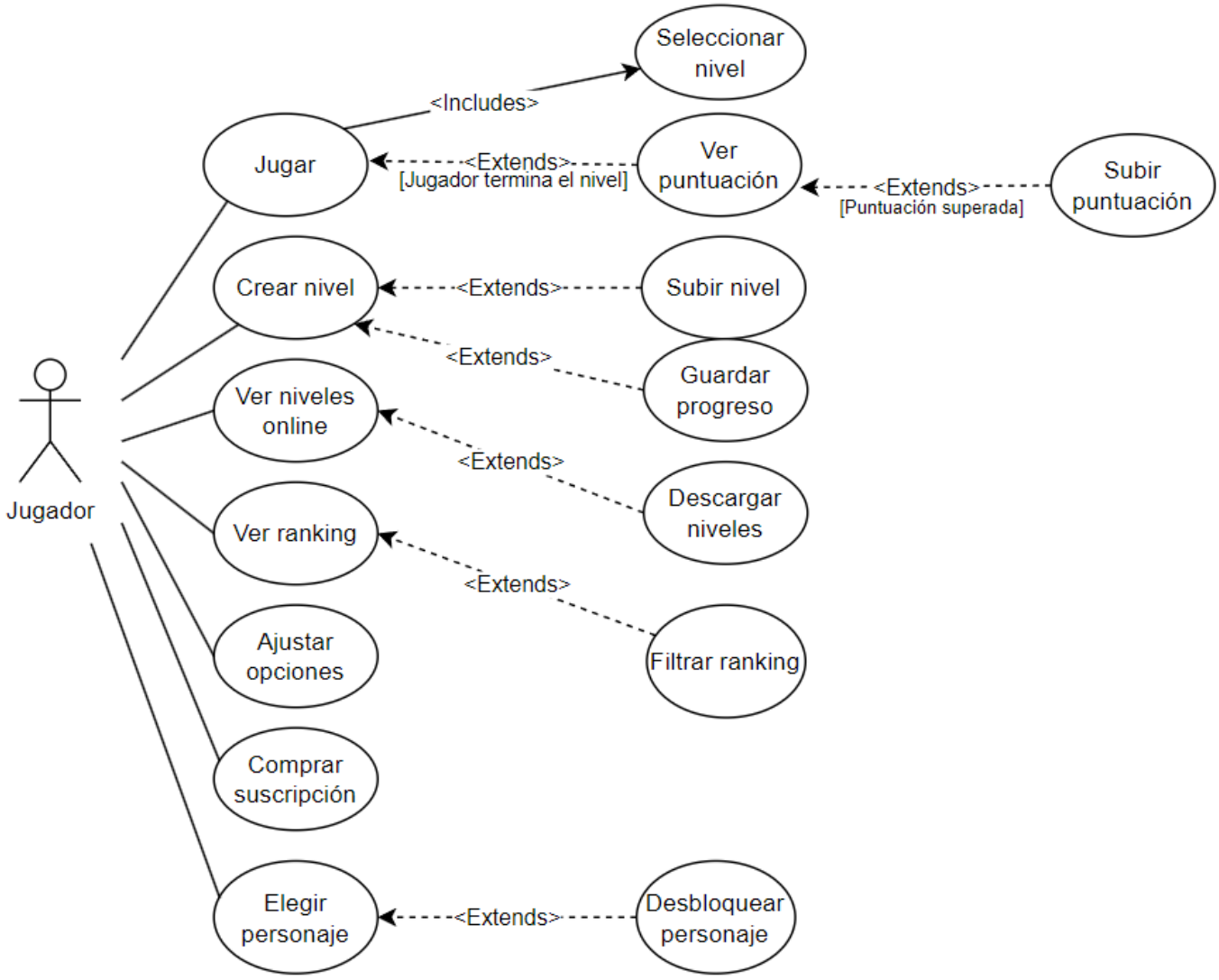


Figura 8. Diagrama de casos de uso. Realizado por el alumno.

11.4 Diseño conceptual

En este apartado veremos el concepto del juego de una manera más profunda, explicando algunos pormenores y mecánicas.

También explicaremos la visión que tenemos al plantear el proyecto, el carácter diferenciador del mismo con respecto de otros juegos y lo que lo hace interesante para algunos jugadores frente a otras opciones del mercado.

11.4.1 Concepto del juego

Rythm Rush es un juego rítmico en 2D, que mezcla el estilo pixel-art e ilustración vectorial. Esto quiere decir que usaremos tanto dibujo vectorial como píxel art. Junto a colores simples, dará una estética retro que acompañe al concepto de juego.

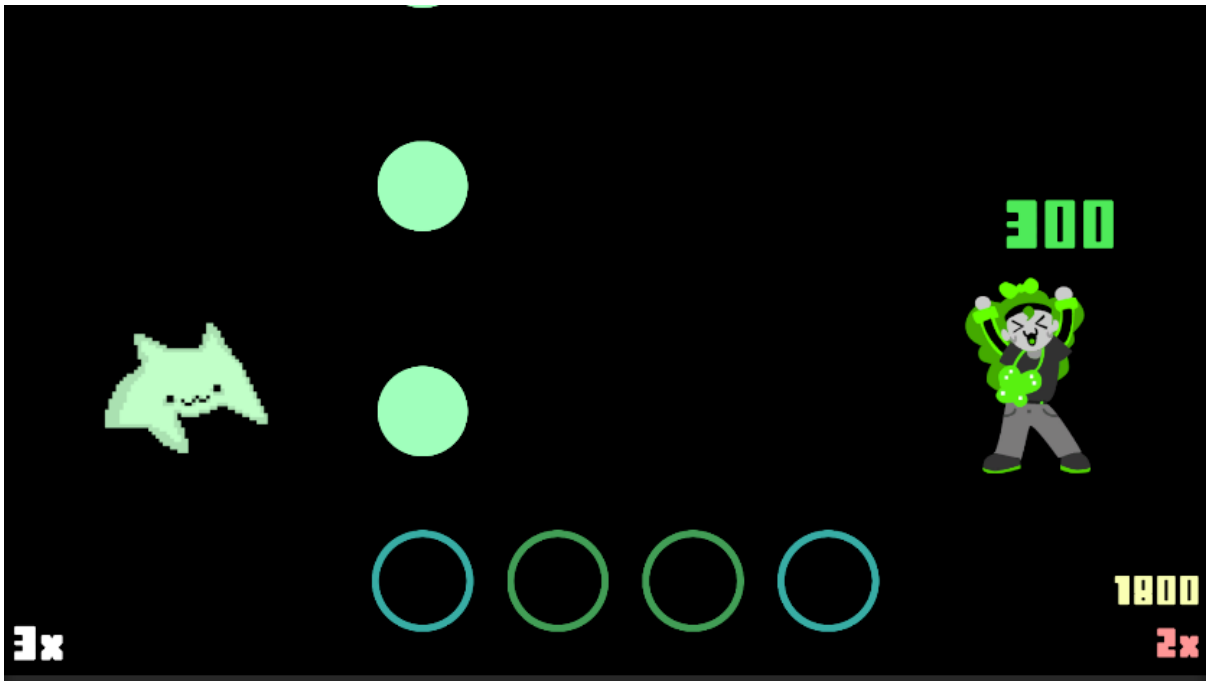


Figura 9. Ejemplo de arte. Realizado por el alumno.

Usando nuestra mano derecha en el teclado y nuestra mano izquierda en el ratón, simularemos los controles de un instrumento musical, que tendremos que ir tocando al ritmo de la música.

En <esta compañía> somos grandes fans del mundo BeMaNi, por lo que queríamos crear nuestro propio título del género, basándonos en grandes títulos del sector como Guitar Hero, Osu!, Dance Dance Revolution, Beat Saber o Muse Dash.

Podemos dividir el contenido del juego en dos partes: los niveles oficiales y los niveles de la comunidad. Los niveles oficiales se lanzarán en forma de pack. Estos packs podrán comprarse y tratarán de ser colaboraciones con artistas que puedan resultar interesantes para los jugadores. Los niveles de la comunidad serán creados por los propios jugadores mediante el creador de niveles. Ambas partes se tendrán en cuenta para el ranking de jugadores.

Una característica que buscamos con el desarrollo de este juego es que llame tanto al jugador casual como al jugador competitivo.

Por su parte, el jugador casual busca un juego donde pasar un rato entretenido, en sesiones cortas de juego. Dado el estilo que hemos escogido, el jugador puede simplemente jugar 20-30 minutos y tener una experiencia amena.

Asimismo, volver a jugar no será una experiencia repetitiva, debido a la variedad de dificultades y canciones disponibles en el juego.

Por otro lado, el jugador competitivo buscará tener las mayores puntuaciones en las dificultades más altas del juego, lo que requiere de mucha práctica y dedicación.

Además, gracias al creador de niveles, tanto la dificultad como el contenido "no acaban nunca". Esto es debido a que las comunidades que se forman alrededor de este género suelen aportar mucha creatividad con este tipo de herramientas.

Todo este contenido, bien acompañado de un buen sistema de rankings, hará las delicias de los jugadores que busquen competir contra otros, e impulsará la creación de comunidades relativas al juego.

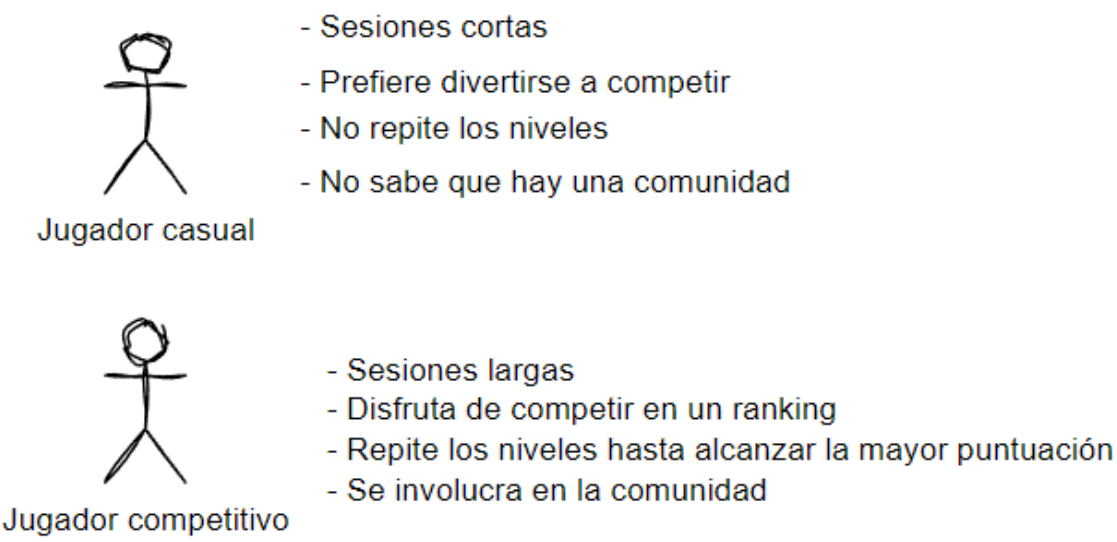


Figura 10. Diferencias entre jugadores. Realizado por el alumno.

11.4.2 Ambientación

El juego carece de historia o narrativa por sí mismo. No buscamos un juego de largas sesiones donde el jugador se vea forzado a jugar durante horas y sin perder detalle.

Nuestra visión con Rythm Rush es transportarnos a esa época de las máquinas recreativas, donde jugábamos juegos arcade y competíamos por ser el mejor en la tabla de puntuación, sin más pretensiones que pasarlo bien durante un rato.

Los videojuegos tipo arcade se caracterizan por su jugabilidad simple, repetitiva y de acción rápida. Pese a que la jugabilidad es simple, la habilidad y reflejos requeridos para jugarlos a alto nivel los hacen perfectos tanto para jugadores casuales que solo buscan entretenerse como para jugadores competitivos que busquen competir contra los demás (y contra sí mismos).

Figura 11. Dance Dance Revolution. (Wikipedia, 2017)

Debido a estos factores, se pretende que estilo artístico del juego sea simple a la vez que llamativo, evocando la estética de los clásicos arcade de los años 80 y 90. Los efectos visuales sincronizados con la música y los elementos de diseño retro crean una experiencia inmersiva y nostálgica para los jugadores.

11.4.3 Mecánicas

En la simpleza radica la elegancia. Las mecánicas de Rythm Rush pueden reducirse a dos: pulsar y mantener.

Por un lado, nuestra mano izquierda, usará el teclado para pulsar dos teclas. El juego requerirá que pulsemos las teclas en un momento concreto, aunque también podrá pedirnos que además de pulsarla, la mantengamos durante un tiempo.
Será representado en el juego con dos círculos, a los que se aproximarán las "notas" que tenemos que pulsar o mantener.

Por otro lado, nuestra mano derecha usará el ratón. Tendremos las mismas mecánicas: pulsar y mantener.

Por lo general, usaremos este control para representar sonidos que no encajen con los de la mano del teclado, como sonidos de fondo o instrumentos diferentes.
Será representado en el juego por unas barras a los lados de los círculos anteriormente mencionados. Dichas barras indicarán el tiempo que debemos mantener pulsado cada botón.

Juntando las mecánicas de las dos manos, el sentimiento que esperamos transmitir será el de estar tocando un instrumento musical, acompañando a la canción.

A priori es un sistema simple, donde usamos cuatro inputs para realizar dos acciones. Cualquier persona puede entrar y ponerse a jugar en cuestión de poco tiempo y práctica.

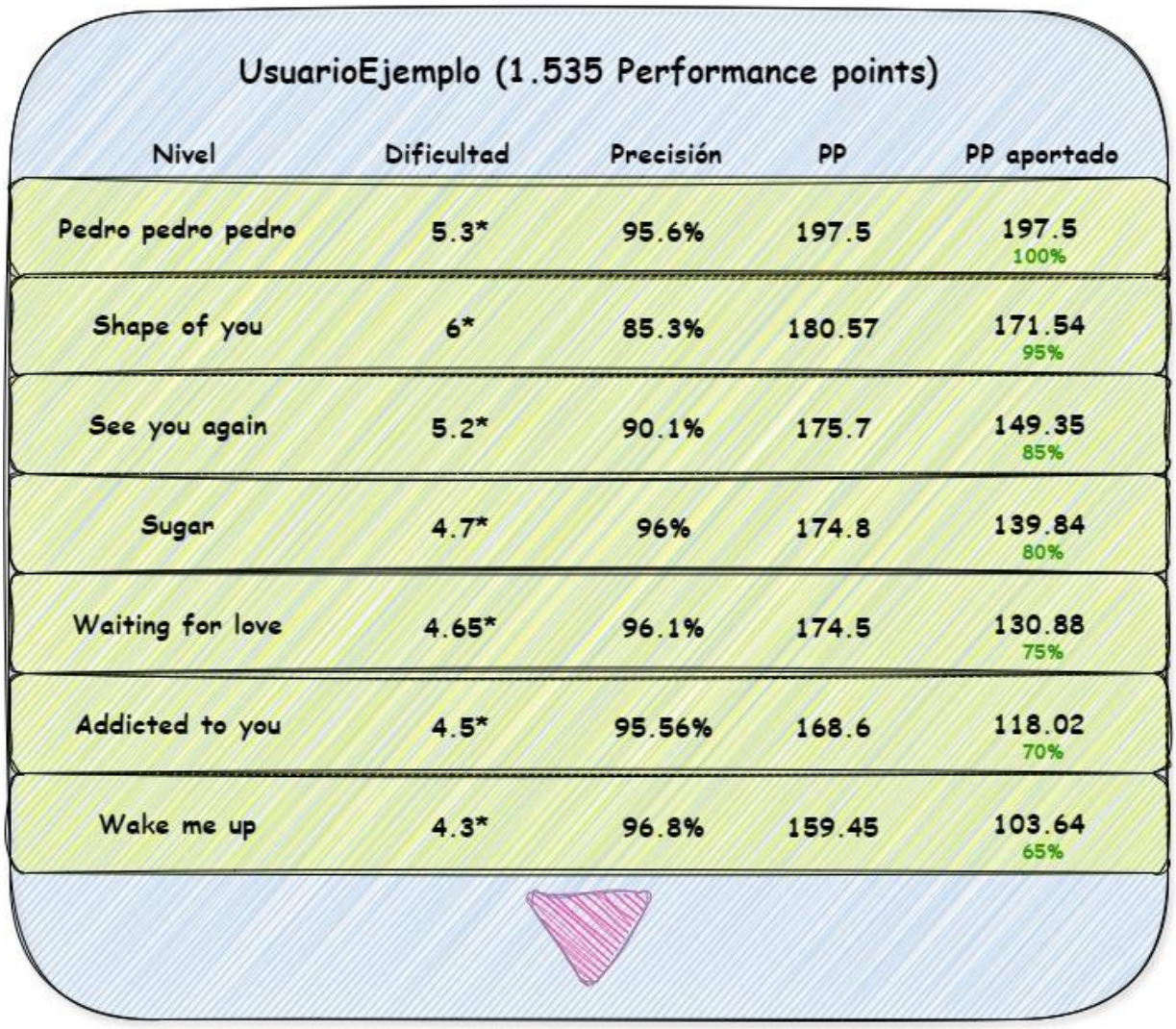
Sin embargo, el sistema de juego se complica con el número y velocidad de notas, así como los patrones de notas escogidos para representar la música. Esto hace que realmente no haya un techo en la habilidad del jugador.

En cuanto al sistema de ranking, estará hecho de tal forma que incite a los jugadores a jugar niveles cada vez más difíciles, haciendo un balance entre precisión y dificultad.

Cada puntuación en la lista de nuestro ranking tendrá menos peso sobre la puntuación total como jugador.

Esto implica que, si por ejemplo nuestras primeras 10 puntuaciones son en canciones con dificultad 5, y luego completamos una canción con dificultad 3, la de dificultad 3 tendrá muy poco peso sobre nuestra puntuación como jugador, incitando a los jugadores a mejorar en el juego (y por lo tanto, jugar más).

A continuación, podemos ver un ejemplo de tabla de puntuación, representando el sistema que hemos definido.



UsuarioEjemplo (1.535 Performance points)

Nivel	Dificultad	Precisión	PP	PP aportado
Pedro pedro pedro	5.3*	95.6%	197.5	197.5 100%
Shape of you	6*	85.3%	180.57	171.54 95%
See you again	5.2*	90.1%	175.7	149.35 85%
Sugar	4.7*	96%	174.8	139.84 80%
Waiting for love	4.65*	96.1%	174.5	130.88 75%
Addicted to you	4.5*	95.56%	168.6	118.02 70%
Wake me up	4.3*	96.8%	159.45	103.64 65%

Figura 12. Ejemplo de tabla de puntuaciones. Realizado por el alumno.

11.4.4 Personajes y enemigos

En nuestro caso, no dispondremos de personajes jugables ni enemigos.

Sí habrá personajes dentro del juego. Sin embargo, la única función de éstos es la de la decoración mientras jugamos. Digamos que nos acompañarán mientras jugamos. Dispondremos de algunos personajes con la compra base del juego, y habrá una funcionalidad donde podremos desbloquear más.

Algunos ejemplos podrían ser, un chico que baile al ritmo de la música, una animadora, un DJ...

Asimismo, en futuras ampliaciones donde hagamos colaboración con ciertos artistas podremos añadir personajes con temática de estos.

Figura 13. Personaje estilo vectorial. Realizado por el alumno.

Figura 14. Bongo cat píxel art. Realizado por el alumno.

Diseño de interfaces

A continuación, mostraremos mockups de las diferentes interfaces y la navegación entre ellas. Esto nos ayudará a plantear el diseño cuando estemos haciendo la aplicación. Estos mockups no tienen por qué coincidir al 100% con el resultado final, pero nos servirán como punto de partida. En la siguiente imagen definimos las escenas con mayor peso a la hora de jugar.

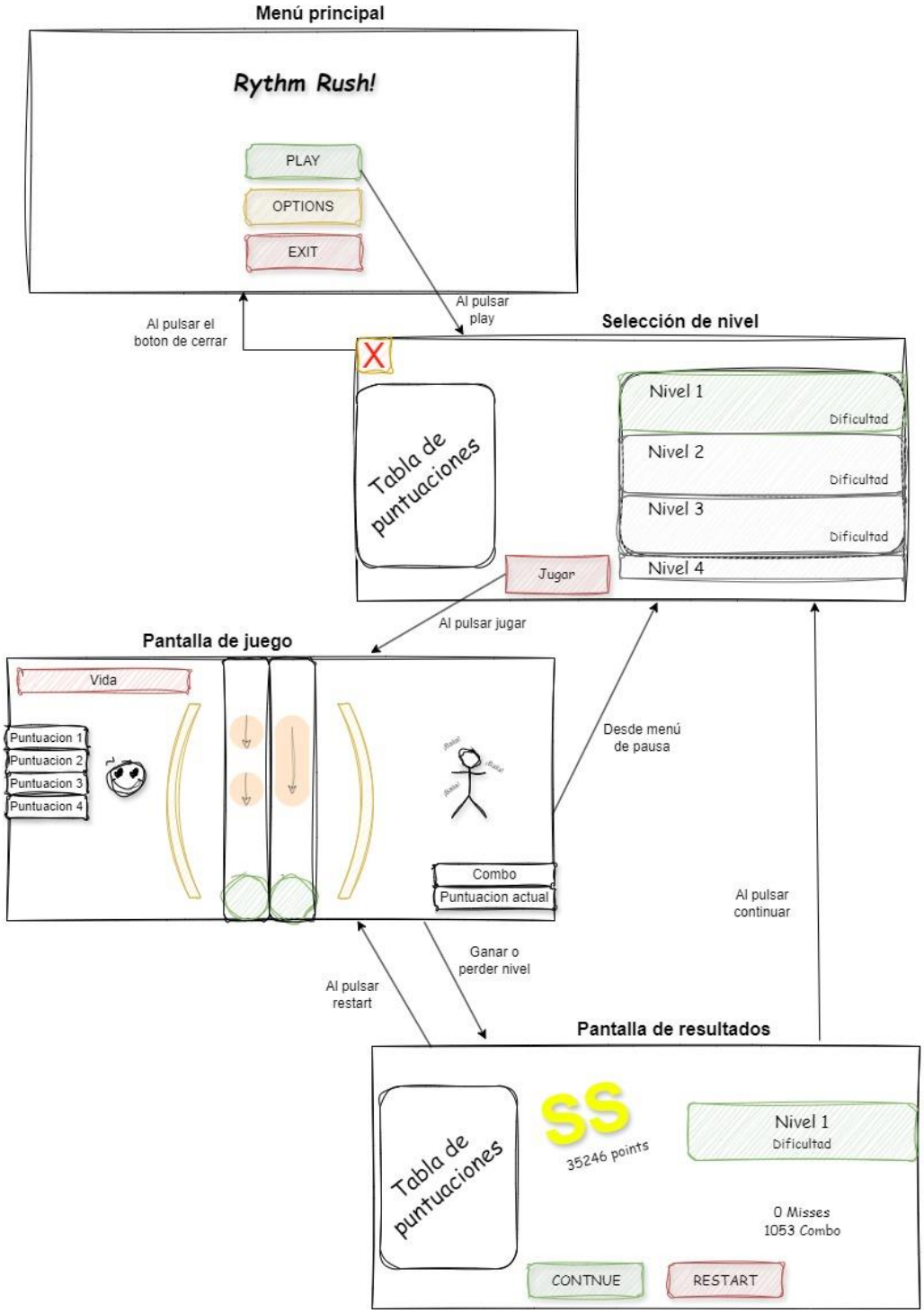


Figura 15. Mockup. Realizado por el alumno.

Las pantallas planteadas para el producto final serían:

- **Menú principal:** Es lo primero que vemos al abrir el juego y lo último que vemos al salir. Podemos acceder al selector de niveles, al menú de opciones y a salir del juego.

- **Selección de niveles:** Desde aquí podremos ver los diferentes niveles y seleccionar uno para jugar. Podremos ver la tabla de puntuaciones del nivel seleccionado. También podremos volver al menú principal.

- **Pantalla de juego:** Pantalla principal del juego. Podremos ver diferentes elementos relacionados con la partida. Cuando acabamos un nivel (ganando o perdiendo) accedemos a la pantalla de resultados. Podremos abrir el menú de pausa, desde el cual podremos salir directamente a la selección de nivel.

- **Pantalla de resultados:** Podremos ver los resultados de la última partida jugada. Podremos reiniciar el nivel o ir a la pantalla de resultados.

- **Pantalla de créditos:** aquí incluiremos los assets y sus autores, respetando las licencias requeridas para su uso.

- **Creador de niveles:** Nos proporcionará las herramientas necesarias para poder crear un nivel, guardar el progreso localmente y publicarlo cuando hayamos terminado con el mismo.

- **Pantalla de rankings:** Podremos ver y filtrar el ranking de los mejores jugadores.

12. Implementación

A continuación, haremos un análisis de los diferentes recursos que necesitaremos para el desarrollo, así como un diario de desarrollo donde apuntaremos cosas relevantes sobre el desarrollo.

12.1 Tecnologías a emplear

A continuación, haremos un breve análisis de las tecnologías empleadas para el desarrollo:

- **Unity:** Motor gráfico, ampliamente usado en el mundo de los videojuegos. Será nuestra principal herramienta en el desarrollo. Usaremos una licencia personal, que no tiene costes hasta llegados unos requisitos mínimos bastante elevados.

Elegimos Unity ya que lo consideramos la mejor opción frente a sus competidores, Godot y Unreal Engine.

En el caso de Godot, es menos estable, menos potente gráficamente y proporciona menos rendimiento, del que no podemos prescindir en este tipo de juegos donde los milisegundos importan.

En cuanto a Unreal Engine, pese a que tiene soporte para juegos en dos dimensiones, las herramientas y oferta de assets son más limitadas ya que se centra más en el desarrollo de juegos 3D “triple A”.

Además, tenemos que añadir que nuestro equipo es especialista y tiene experiencia con Unity, lo que nos ahorrará un proceso de aprendizaje.

- **Lenguaje de programación C#:** Lenguaje de programación orientado a objetos.

Realmente no tenemos elección aquí. Si usamos Unity, usamos C#. Es un lenguaje en el que tenemos experiencia y nos proporciona muchas facilidades con librerías como LINQ.

- **Node-js:** Entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos.

Lo usaremos para construir nuestra API Rest. Tenemos experiencia usándolo y nos proporciona una forma fácil y rápida de implementar la API.

- **Audacity:** Software de edición de audio. Lo usaremos para retocar principalmente el offset de las canciones y que puedan coincidir con el ritmo en nuestros niveles. También lo usaremos para retocar algunos efectos de sonido incluidos en el juego.

  Elegimos Audacity frente a otras opciones ya que es un software gratuito y muy extendido entre la comunidad. Nos brindará las herramientas necesarias para poder desarrollar nuestro juego sin problema.

- **Firebase:** Base de datos no relacional. Pertenece a Google, nos permite almacenar datos en la nube con facilidad.

  Elegimos Firebase por varios factores. Por un lado, al trabajar con el formato JSON, nos ahorrará dolores de cabeza al trabajar con nuestra API Rest.

  Por otro lado, nos permite despreocuparnos del apartado de seguridad, ya que incluye su propio sistema de autenticación, gestión de usuarios, alertas, definir reglas y encriptación. Además, tiene unos planes de precio interesantes, que incluyen unos límites donde es gratuito.

  A todo esto, hay que añadir que es una plataforma que se encuentra en la nube, por lo que también podremos despreocuparnos de contratar un servidor para esta base de datos.

- **Gimp:** Herramienta de edición de imágenes. La usaremos para construir/retocar assets del juego.

  Lo escogemos frente a su alternativa más evidente (Photoshop) ya que es gratuito, por lo que no requerimos de licencia para usarlo. Nos proporciona herramientas suficientes para el uso que le vamos a dar y una gran comunidad en la que apoyarnos en caso de tener dudas.

- **Inkscape:** Herramienta de edición de imágenes. Lo usaremos para hacer los assets que requieran de dibujo vectorial. Es gratuito.

- **Visual Studio 2022:** IDE de Microsoft en el que escribiremos nuestro código para Unity. Debido a nuestra condición de autónomos, no nos afectan las restricciones y lo podremos usar sin problema de forma gratuita.

Usaremos este IDE para la parte en que usemos C# ya que nos ofrece, con diferencia, la mejor integración con Unity.

- **Visual Studio Code:** Editor de código abierto simple y ligero.

  Usaremos este editor para la parte en la que usemos Node.js. Gracias a su sistema de extensiones podremos personalizarlo a gusto e integrarlo con las herramientas necesarias.

- **Servidores S3:** servidores cuyo propósito es almacenar una gran cantidad de datos en la nube. Sacrificamos un poco de velocidad en las transmisiones de datos en virtud del almacenamiento disponible.

  Dadas las características del proyecto, necesitaremos almacenar archivos de bastante peso en la nube (canción + archivos correspondientes por cada nivel), unos 4MB-8MB por nivel. Esto crea la necesidad de usar este tipo de servicios, donde podemos encontrar precios muy asequibles para el almacenamiento.

  La desventaja de este tipo de servidores es, en gran parte, que no están diseñados para una gran carga de transmisión de datos. Para nosotros no debe ser un problema, ya que los niveles solo se suben una vez, y no esperamos que los usuarios descarguen más de una vez cada nivel, ni que los descarguen todos (solo los que les interesen).

- **GitHub:** es una plataforma de desarrollo colaborativo que utiliza control de versiones Git, permitiendo a los desarrolladores gestionar, compartir y colaborar en código de manera eficiente. Lo elegimos porque facilita el seguimiento de cambios, y la integración continua, asegurando un desarrollo organizado y eficiente.

12.2 Assets

En cuanto a nuestros assets, hay dos grandes grupos que podemos separar: el arte gráfico y la música.

En cuanto al arte gráfico usaremos tanto assets con licencia libre como assets propios. Podremos encontrar los assets con sus licencias en un apartado de los anexos.

En cuanto a la música del juego, usaremos música con licencias compradas, esta es nuestra inversión más grande y un punto crítico en cuanto a que el juego sea viable.

Es importante que el primer pack de canciones contenga artistas que sean conocidos en el sector. Por un lado, tienen experiencia en hacer música para este tipo de juegos, lo que hará que el juego tenga calidad en su ámbito más importante. Por otro lado, esto llamará la atención de los jugadores del nicho, los cuales representan unas ventas seguras y usuarios a largo plazo.

Dado que tenemos un presupuesto limitado, tenemos que ser realistas y no apuntar a artistas de talla mundial, por lo que hemos hecho una investigación sobre el nicho de los juegos de ritmo.

Tras esta investigación, hemos encontrado que muchos de los artistas más reconocidos en el mundo de los juegos de ritmo son de origen japonés, muy reconocidos por la comunidad, pero poco reconocidos fuera de su ámbito.

Hay varios nombres que se repiten, pero el patrón común entre ellos es el sello discográfico, “Tano*C”. Este sello discográfico es especialista en el mundo de los juegos rítmicos y se dedica a reunir autores de música electrónica.

Muchos de los autores que trabajan bajo este sello tienen política de no-copyright para las canciones que no se vayan a usar comercialmente. En nuestro caso, por la información que hemos podido recopilar, no podríamos acogernos a esta política, pero nos viene genial para que nuestros usuarios amplíen el contenido en el creador de niveles.

Asimismo, pudimos contactar al algunos de los autores de dicha marca y resultó que no toda su música tiene copyright, y podemos usarla de manera libre en nuestros juegos rítmicos. Incluso algunos lo anuncian abiertamente en sus redes sociales.

Figura 16. Tuit T+Pazolite. (Twitter, 2020, abril 23)

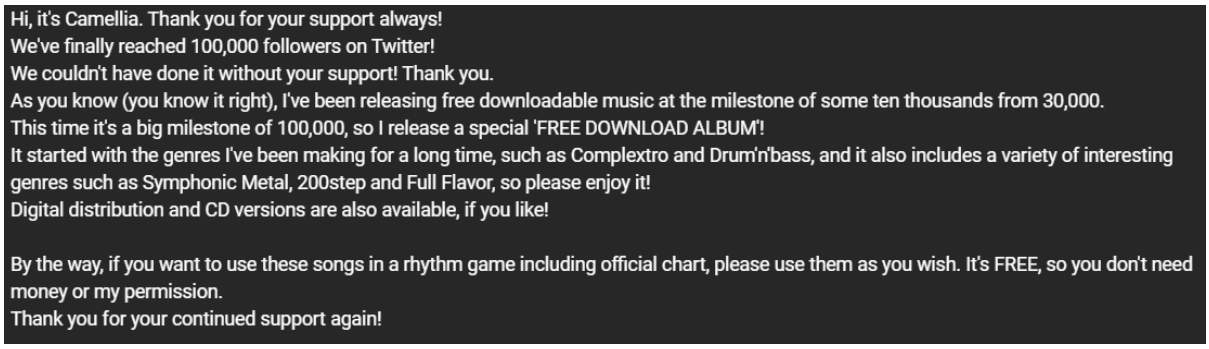


Figura 17. Comentario Camellia. (Youtube, 2020, septiembre 26)

Usar el nombre de estos artistas hará que tengamos un sello de calidad en la música de nuestro juego y despertará el interés entre los jugadores del nicho.

Asimismo, pudimos contactar con artistas emergentes del sector como “Ludicin”, que permitirá que usemos su música en nuestro juego. Este artista, a pesar de ser poco conocido, está tomando fuerza a través de aportaciones en juegos como OSU! o Beat Saber.

Por supuesto, deberemos incluir a los artistas en un apartado de créditos dentro del juego a cambio de usar su contenido.

El resto de assets serán de licencia libre o de creación propia. En caso de necesitar licencia, serán añadidos en un anexo.

12.3 Diario de desarrollo

En este apartado, haremos un resumen de los puntos clave en el desarrollo del prototipo, como las dificultades que hemos tenido al implementar alguna de las partes o detalles que puedan resultar interesantes o relevantes.

- Uno de los primeros puntos que se nos viene a la cabeza al pensar en la implementación de un videojuego de este tipo es, ¿cómo coordinamos la música con los eventos de nuestro juego?

  Este punto es muy importante ya que de él depende que las mecánicas tengan sentido y sean justas. Realmente es un punto que se suele dejar al creador del nivel, ya que también implica tener algo de conocimiento de teoría musical.

  A grandes rasgos, nuestra herramienta permite poner eventos (notas) en base al número de BPM en el que nos encontremos.

  ¿Qué es un BPM?
  ¿Recordamos cuando estudiábamos las blancas, las negras, las corcheas, etc..? Un BPM sería la representación de una negra. Cada canción tiene marcado un ritmo, en BPM, por lo que nuestro trabajo como creador de nivel es encontrar ese número y trabajar en base a él. Algunas veces tendremos que adaptar el comienzo de la canción con herramientas como Audacity para que cuadre el sonido con el ritmo.

  Por otro lado, como desarrollador, es nuestro papel incorporar herramientas para trabajar con estos datos en el creador de niveles, de forma que podamos abstraernos del tiempo en segundos para trabajar con BPM.

- Otro de los puntos más costoso es cómo trabajar con las diferentes canciones con el creador de niveles. Cada canción podemos dividirla en tres partes: la canción, la definición y la información de las dificultades.

Lo hemos planteado de la siguiente manera: Cada nivel tendra su propia carpeta. Esta carpeta contiene una pista de audio, un archivo que define las propiedades de la canción (como la ID, número de dificultades, nombre del nivel...) y un archivo por cada dificultad que contendrá la información propia, como cada nota que debe aparecer en qué momento.

En el desarrollo del prototipo contamos con muy poco tiempo, que a penas da para montar una pequeña maqueta para hacerse una idea. Contamos con 40 horas para poder hacer esto, de modo que nos hemos centrado en la parte jugable del proyecto, y no tanto en la infraestructura que hay detrás.

Por un lado, no queríamos usar demasiados assets externos, así que dibujamos el sprite sheet de la chica que baila. Esto tomó unas 6 horas, por lo que hubo que tomar la decisión de usar más arte de terceros.

Por otro lado, como mencionamos anteriormente, la mayor dificultad vino con la parte de cuadrar la música con nuestras acciones, haciéndonos perder mucho tiempo. Surgieron cantidad de problemas, como que la reproducción de la canción causaba un tirón que descuadraba todos los tiempos y tardaba en empezar. Después, el sonido parecía cuadrar, pero la parte visual no. Luego dejó de funcionar el sistema de notas...

Debido a esto sólo hemos podido implementar una de las dos mecánicas que esperábamos tener en el prototipo, aunque sigue siendo suficiente para poder hacer niveles jugables.

En este punto hubo que elegir, ya que con todo lo anterior habíamos consumido unas 25 horas.

Ya que la parte de base de datos no tiene tanta relevancia en nuestro proyecto (es opcional), decidimos dedicar el tiempo a mejorar el prototipo trabajando en local.

Como podemos imaginarnos, había que hacer algún nivel para poder jugar, y no tenemos un creador de niveles, el cual ni por asomo íbamos a poder tener. De modo que tuvimos que escribir a mano cada nota y ajustarla repitiendo una y otra vez hasta que el resultado fuera aceptable.

Por suerte, contamos con experiencia de otros juegos y pudimos ayudarnos del creador de niveles de Beat Saber. Aun así, esto es un proceso largo y costoso que llevó bastante esfuerzo.

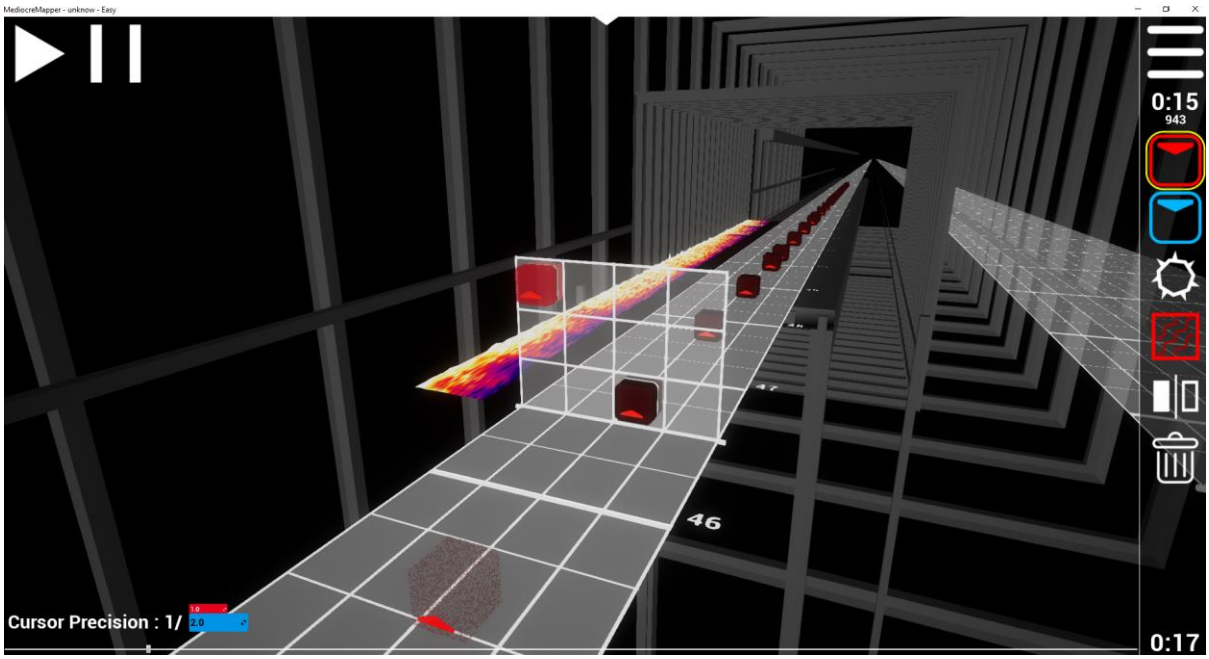


Figura 18. Creador de niveles de Beat Saber. Creado por el alumno.

El resto del tiempo se invirtió en hacer el juego más “juicy”. Este término hace referencia a técnicas de diseño que hacen que el juego sea más gratificante, como añadir efectos visuales, animaciones, feedback inmediato, sonidos...

13. Pruebas

Las pruebas son un factor importante de cara al lanzamiento y desarrollo de nuestra aplicación. A continuación, pondremos la información relevante a las pruebas en el desarrollo de nuestro proyecto.

13.1. Procedimiento de evaluación, seguimiento y control del proyecto.

Debido a la naturaleza del proyecto, es importante decir que prácticamente la totalidad de las pruebas que se han realizado a la hora del desarrollo han sido hechas de forma manual.

Dado que estamos trabajando con un sistema a tiempo real, donde muchas veces los resultados son mostrados de forma gráfica, la realización de tests unitarios es una tarea complicada y muchas veces irrelevante.

Lo más parecido a tests unitarios que tenemos, es una clase que nos permite hacer pruebas manuales a tiempo real del guardado/cargado de niveles en JSON.

Cabe mencionar que no se ha implementado la parte donde subimos contenido o lo descargamos, donde sí tendría sentido hacer una suite de tests unitarios relacionados con esas peticiones.

Por nuestra parte, Unity nos proporciona algunas herramientas que podremos usar para hacer pruebas manuales. Las dos que hemos utilizado principalmente son: el sistema de logs y el sistema de interfaz de usuario.

El sistema de logs de Unity nos permite enviar mensajes a una consola dentro del mismo programa, que a tiempo real irá mostrando información que podamos necesitar.

A parte de sus propios errores, warnings y logs, Unity permite crear los nuestros propios, por lo que es una herramienta ideal para revisar lo que está pasando en ciertos momentos.

Por otra parte, en algunos momentos preferimos usar algún texto que cambie a tiempo real ya que los logs son una herramienta, a veces, inapropiada. Algunos ejemplos de esto son: contador de FPS para revisar cierto punto en el que el rendimiento no era óptimo, o también un contador de tiempo-fps para controlar que nuestro sistema estaba funcionando correctamente.

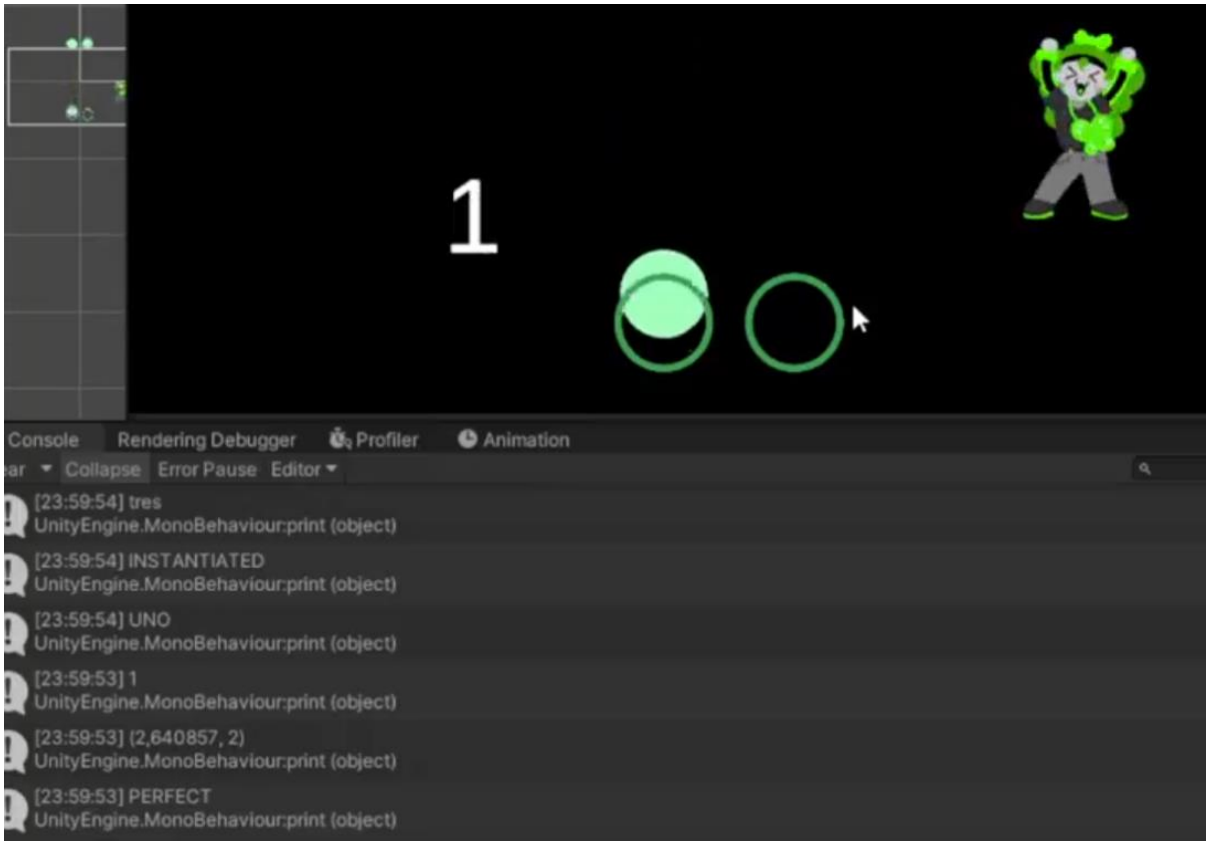


Figura 19. Ejemplo de logs. Creado por el alumno.

Además, en las últimas versiones de Unity han añadido la posibilidad de debuguear el código, aunque no suele usarse con frecuencia debido a la naturaleza de los sistemas a tiempo real. De hecho, al no estar muy pulido, nos trajo más dolores de cabeza que ayuda como tal.

Fuera de estas herramientas, es una práctica común entre desarrolladores de videojuegos tener la rutina de probar el juego después de cada cambio hecho en el código (pruebas manuales).

Esto quiere decir que después de cada cambio, nos aseguramos de ir al punto concreto donde afecta dicho cambio y probarlo de todas las maneras posibles para asegurar su correcto funcionamiento, entrando en una rutina en la que haremos cambio-prueba-cambio-prueba de manera manual hasta que todo esté correcto.

13.2. Procedimientos para la participación de los usuarios en la evaluación del proyecto

Recibir feedback de usuarios es una de las mejores maneras de encontrar fallos o planteamientos nuevos que podamos ir adaptando a tiempo antes de lanzar nuestro producto.

En nuestro caso, hemos dejado probar el prototipo a cinco personas, las cuales están familiarizadas con este tipo de juego.

El feedback recibido ha sido positivo en su mayoría, entendiendo los testers que es un prototipo muy temprano. Esto refuerza la idea de que nuestra idea tiene potencial.

Sumado a todo, hemos recibido grandes ideas por su parte, que podrían ser introducidas en un futuro como contenido jugable, por lo que concluimos que la participación de los usuarios ha sido muy beneficiosa en el desarrollo del proyecto.

14. Conclusiones

A lo largo de la memoria, hemos podido establecer un plan de empresa bastante sólido, planteando un producto viable para que una sola persona, en condición de autónomo pueda desarrollarlo. No hemos necesitado de una gran inversión y podemos llevarlo a cabo financiándolo de manera autónoma.

También hemos podido analizar la parte técnica, incluyendo un pequeño prototipo que permite hacernos una idea de si el producto tiene un potencial real, llegando a la conclusión de que sí lo tiene.

Consideramos que el proyecto puede ser llevado a cabo por una sola persona, pese a necesitar assets de terceros. Eso sí, aun apuntando a un mínimo producto viable, es un proyecto bastante exigente debido a la estructura de los componentes y al conocimiento específico que requiere.

A título personal, me gustaría llevarlo fuera del ámbito escolar y publicarlo, ya que es un proyecto que creo que merece la pena y que puedo llevarlo a cabo de la manera correcta, con mis conocimientos en el sector.

En cuanto al proyecto en sí considero que es una oportunidad genial para desarrollar "esa idea que siempre has querido hacer", recibiendo un feedback relevante. Sin embargo, es muy complicado de llevar junto a las FCT y los plazos propuestos.

Por supuesto, más allá del mínimo producto viable, hay muchas mejoras e ideas que pueden realizarse.

Algunas de ellas serían:

- Mejorar la personalización en las canciones añadiendo un sistema donde los creadores puedan decorar el nivel con elementos y animaciones.
- Añadir un sistema para poder jugar junto a otras personas.
- Añadir un modo torneo
- Si miramos el mundo del arcade, este tipo de juegos son una opción genial de cara a crear máquinas recreativas donde poder jugarlos en locales físicos (sobre todo en el mercado asiático), por lo que podríamos considerar la opción de expandirnos a este mercado en un futuro.

15. Bibliografia

- OVH Cloud.(2024).Servidores s3 https://eco.ovhcloud.com/es-es/?display=list&storage=SATA
- Tpazolite. (2020, Abril 23). Tuit t+Pazolite. X. https://x.com/tpazolite/status/1253262594451763201
- Camellia. (2020, Septiembre 26). Comentario Camellia https://www.youtube.com/watch?v=M3npCLBbg-s
- Ludicin. (n.d.). Perfil Ludicin. Soundcloud. https://soundcloud.com/ludicin/tracks
- Coworking Spain.(2024). Coworking. CoworkingSpain. https://coworkingspain.es/espacios/coworking/alicante/coworking12
- Quora. (2021). Crecimiento del sector de los videojuegos. Quora. https://es.quora.com/
- Mwebs. (N.d.). El modelo incremental. Mwebs. https://mwebs.com.uy/blog
- Researchgate. (2019). Ejemplo trello. ResearchGate. https://www.researchgate.net/
- Wikipedia.(2017). Dance Dance Revolution. Wikipedia. https://es.wikipedia.org/

16. Anexos

16.1 Lista de assets de terceros

- Retro font:
https://www.dafont.com/es/retro.font
- Sci-fi vector elements:
https://assetstore.unity.com/packages/2d/textures-materials/scifi-vector-elements-205002
- Srav3R – RAV#GIRL:
https://www.youtube.com/watch?v=eDfHlrSdB8w&list=RDeDfHlrSdB8w&start_radio=1
- Musica menú: www.suno.ai
- Free UI Click Sounds Effects: https://assetstore.unity.com/packages/audio/sound-fx/free-ui-click-sound-pack-244644