UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Muschain, un juego colaborativo para compositores

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carlos Caballer Chacón

Tutor: Vicente Pelechano Ferragud

Curso 2019/2020

Resumen

El objetivo de este TFG es el desarrollo de un juego para músicos y compositores, con su correspondiente estudio de mercado y aceptación por parte de los usuarios. Muschain es un juego colaborativo, en el cual se quiere explorar la creatividad y la colaboración entre artistas. El objetivo es que diferentes compositores participen en el desarrollo de una canción, cada uno realizando una pequeña porción de la misma, para luego juntarlos todos y obtener una pieza completa. Además esto permite descubrir nuevas personas con quien compartir su afición por la música, de una forma interactiva y divertida.

El desarrollo de la aplicación se realizará mediante el uso de metodologías ágiles y Lean Startup. Para ello, se usarán herramientas como Git, PyCharm y Back4App. Se realizarán dos MVP y su correspondiente experimento con un conjunto de *early adopters*, obteniendo los resultados mediante una encuesta a través de un formulario online. Tras el análisis de los resultados se procederá a la creación de una segunda iteración del producto, que incluirá nuevas características y mejoras de las ya presentadas, junto con un nuevo experimento con una población más grande. La repetición de este proceso nos permitirá obtener una versión cercana al producto final.

Palabras clave: MIDI, Python, QT, Redes sociales, Audio, Música, Juego.

Abstract

The goal of this TFG will be to develop a game for musicians and composers alike, doing a market research and studying the user reception. Muschain is a collaborative game where the player is invited to be creative and interact with other artists. Each player is prompted to write a little piece of music that afterwards it's *chained* with other player's pieces to form a whole song. This also allows players to meet new people with similar likings and hobbies in a funny and interactive way.

Development will be guided by agile methods and Lean Startup, using software tools like Git, PyCharm and Back4App. There will be two MVPs or demo releases with early adopters to obtain user reception via an online form. The second version of the product will be made after the analysis of the data obtained from the first demo, adding new features and improving those already present in the prior version. Repeating this process several times will lead us to a working version of the product close to the final product.

Keywords : MIDI, Python, QT, SNS, Audio, Music, Game.

Tabla de contenidos

Tabla de figuras

1. Introducción

Existen infinidad de redes sociales en la actualidad, cada una con un formato diferente para las publicaciones y un *target* de usuarios distinto. Algunas están centradas en publicar entradas de texto o *posts*, pudiendo ir acompañadas de imágenes o vídeo. Estas redes sociales pertenecen a la categoría de *blogging*, ya que se basan en publicaciones con un formato similar a las entradas de un blog. El referente más destacado es Facebook [1], que cuenta actualmente con una base de unos 2.500 millones de usuarios activos al mes [2].

A partir de la popularidad de este formato nacieron otras redes sociales como Twitter [3], cuyo atractivo reside en la limitación del número de caracteres por entrada. Así, a diferencia de Facebook donde es posible publicar *posts* muy largos con todo lujo de detalles, en Twitter es necesario resumir y concentrar la información en la menor longitud posible. A este formato se le concibió el nombre de *microblogging*.

También surgieron otras redes sociales con un formato completamente distinto, como es el caso de Instagram [4] y sus *posts* basados en imágenes (más tarde también vídeos), YouTube [5] para vídeos, y SoundCloud [6] para música.

Con todo el auge de las redes sociales, la tecnología y el uso de Internet en general, también empezaron a aparecer adaptaciones de diversos juegos de mesa en forma de páginas web o aplicaciones software. Así ya no era necesario reunirse el grupo de amigos o familiares en un mismo lugar para poder disfrutar de este tipo de juegos. Algunos ejemplos son Apalabrados [7] (una adaptación del juego de palabras cruzadas Scrabble [8]), Pinturillo [9] (adaptación del juego de dibujo y preguntas Pictionary [10]), entre otros.

Por otro lado, también las herramientas que usamos para el trabajo se vieron obligadas a adaptarse ante la popularización de Internet. Tenemos la aparición de *suites* de ofimática donde es posible redactar documentos entre varias personas al mismo tiempo, como es el caso de Google Docs [11] o Graphite Docs [12]. O la función que ofrecen muchos programas de *guardado en la nube*, lo cual nos permite disponer de una copia en Internet de nuestro documento/proyecto a salvo de cualquier catástrofe que pueda ocurrir en nuestro dispositivo.

Cada uno de los productos mencionados anteriormente tienen un objetivo más o menos distinto: las redes sociales sirven para compartir ideas, mantenerse en contacto con nuestros conocidos y ampliar nuestro círculo de contactos. Los juegos sociales, pese a que es posible conocer gente nueva a través de ellos, están pensados para ser usados con un grupo de personas que ya se conoce previamente. Y las *suites* de ofimática o cualquier otro software de productividad están ideadas para facilitar la labor y reducir el tiempo dedicado a maquetación o corrección de errores, para poder dedicarlo a desarrollar nuestro proyecto.

Sin embargo, ¿qué ocurriría si intentáramos juntar estos conceptos en un solo producto? Un producto que nos permitiese relacionarnos con los demás, pasar un buen rato y además servirnos para desarrollar un proyecto. Con esta idea nace el experimento de Muschain.

1.1. Motivación

La idea de Muschain surgió por mi afición a la música y la composición, así como por las ganas de desarrollar un producto distinto que no existiese en la actualidad. Al principio contemplé muchas otras alternativas que juntaran la música y la informática de distintas formas, entre las cuales destacaban:

- El desarrollo de un videojuego en colaboración con otros compañeros (encargándome de la parte de audio y la banda sonora, junto con parte de la programación)
- El desarrollo de un plugin para el reproductor VLC [13] que permitiese sincronizar la librería de música entre el PC y el smartphone mediante el protocolo MTP [14], utilizando un conversor de audio para comprimir los archivos en el proceso
- Una plataforma alternativa a SoundCloud que permitiese a los artistas publicar sus canciones, y a su vez hacer uso del protocolo ActivityPub [15] para poder interoperar con otros servidores dentro del Fediverse [16]

Comencé participando en el desarrollo del videojuego, ya que era una idea que nos atraía a todos los participantes y nuestra intención era presentarlo como un TFG conjunto. Sin embargo, debido a la alta carga de trabajo artístico (tanto visual como sonoro), decidimos apartar el proyecto y buscar otra alternativa. Sobre lo anterior se sumó que, personalmente, no me encontraba demasiado cómodo trabajando con el motor de videojuegos Unity3D [17], así que decidí buscar otro proyecto por cuenta ajena.

Sobre los otros dos proyectos, me encontré con que la documentación por parte de VLC para el desarrollo de plugins no era excesivamente accesible, y desarrollar la plataforma alternativa a SoundCloud conllevaría el manejo de tecnologías web para el diseño tanto del frontend como del backend, sobre las cuales apenas tenía conocimientos. Por ello descarté ambas ideas y me decanté por lo que terminaría siendo Muschain.

Muschain es un software para PC escrito en Python, el cual hace uso del framework QT [18] para la interfaz gráfica. Para simular la parte del servidor y almacenar las canciones de los usuarios utiliza la API REST de Back4App [19], una base de datos online al estilo de Firebase [20] de Google basada en el proyecto open-source Parse [21]. La elección de estas tecnologías está más relacionada con algunas preferencias personales:

- Siempre me llamó la atención aprender a programar en Python, por lo que me pareció un buen momento para profundizar mis conocimientos.
- QT es un framework bastante extendido y compatible con distintos sistemas operativos como Windows, Mac y Linux. Se adapta al estilo nativo de la interfaz del sistema, además de ser el framework que utiliza el entorno de escritorio Plasma [22] de KDE, el cual tengo instalado en mi PC.

1.2. Objetivos del TFG

Este proyecto tiene como principal objetivo estudiar la viabilidad de Muschain como un producto y servicio en el mercado, así como desarrollar una versión funcional, pero no completa. Además, sería conveniente hacer una separación entre objetivos funcionales y no funcionales, los cuales estarán englobados y amparados bajo el objetivo principal mencionado anteriormente:

Objetivos funcionales

- Realizar un estudio de mercado, en el cual se compararán distintos productos y servicios existentes con Muschain, resaltando similitudes y diferencias, características únicas y posibles mejoras de aquellas que sean compartidas.
- Desarrollar una versión funcional de Muschain, que pueda ser usada por usuarios objetivo y comprobar su nivel de aceptación a través de varios experimentos con encuestas.
- A largo plazo, implementar una versión completa de Muschain con la infrastructura web necesaria para ello (parte del servidor o backend, aplicación web o frontend, y la posibilidad de complementarlo con clientes nativos para escritorio o móvil). El motivo por el cual no se ha optado por esta vía directamente radica en los costes asociados al *hosting* de la aplicación, ya que prácticamente la totalidad de los servicios de alojamiento gratuito que hay disponibles se limitan a páginas web, no a aplicaciones completas. Para ello, sería necesario la obtención de un dominio (existen algunas opciones gratuitas) y la contratación de un VPS [23]. En su defecto, sería posible también alojar la aplicación de forma local, pero no sería viable más allá del primer experimento del MVP por la imposibilidad de dejar el ordenador personal encendido de forma continua.

Objetivos no funcionales

- Siendo la primera aplicación compleja que desarrollo utilizando Python, un objetivo personal es aprender y comprender mejor el lenguaje, el cual goza de un alto interés por parte de la comunidad de desarrolladores [24] para distintos ámbitos como web, inteligencia artificial y manejo de datos científicos.
- Encontrar una nueva conexión entre el mundo de la informática como desarrollador y mi afición por la música y la composición. El manejo de audio de forma programática resulta de una complejidad algo superior al de imágenes, debido a que son necesarios algunos conocimientos técnicos específicos de sonido, e incluso de matemáticas y física si se quiere indagar en la síntesis de sonido.

1.3. Estructura

Esta memoria para el TFG consta de seis secciones principales. A continuación realizaremos un pequeño resumen de cada una de ellas, donde indicaremos qué se espera encontrar en cada una de ellas:

- En la primera sección encontramos la **Introducción** (*sección actual*), donde ha puesto en contexto la idea del proyecto, los motivos que han determinado el desarrollo de esta idea, así como los objetivos del TFG

- En la segunda sección, **Estado del arte**, se hablará del contexto tecnológico actual, productos reales que han servido de fuente de inspiración para el proyecto y su relación con él, así como algunos detalles de los mismos

- En la tercera sección, **Evaluación de la idea de negocio**, se realizarán diversos análisis y métricas para estudiar la viabilidad y la competencia de mercado para el proyecto. Para ello, se mencionarán algunas de las características que tendrá nuestro producto, con el objetivo de compararlo con otros productos comerciales existentes. Para finalizar, se realizará una proyección económica para simular la puesta en el mercado del producto y su evolución.

- En la cuarta sección, **Desarrollo de la plataforma**, se hablará sobre los pasos seguidos para la planificación del proyecto y se profundizará en las características que tendrá nuestro producto, completando la información presentada en la sección anterior. Se realizarán dos demostraciones del producto en diferentes fases del desarrollo con un grupo reducido de usuarios, para que lo prueben y expresen su valoración mediante un formulario.

- En la quinta sección, **Aspectos técnicos**, se detallarán las herramientas utilizadas para desarrollar el proyecto, la arquitectura utilizada y también los problemas y desafíos técnicos encontrados antes y durante su desarrollo.

- Para terminar, en la sexta sección **Conclusiones** se valorarán los resultados obtenidos a través de los dos experimentos. También se hablará sobre los planes de futuro para continuar con el desarrollo del proyecto, así como la experiencia personal obtenida.

Para facilitar la comprensión de algunos párrafos, debido al uso de siglas y términos específicos relacionados con la música, se incluye un **Glosario** al final de este documento con una breve definición de estos términos. En caso de necesitar información más detallada, se incluyen como referencias bibliográficas enlaces a páginas web donde se habla sobre el tema con mayor profundidad.

2. Estado del arte

Debido a que Muschain es un experimento, el cual busca ofrecer un producto que no existe como tal en la actualidad, resulta un poco más complejo compararlo con otros competidores. No obstante, sí podemos hacer una comparativa con otros productos si dividimos Muschain en varios componentes, según su funcionalidad:

2.1. Componente social:

Muschain posee algunas características que lo convierten en una especie de red social, gracias a los perfiles de usuario, *followers*, lista de favoritos, entre otras cosas que veremos más detalladamente en el mapa de características presentado más adelante. Así pues, analicemos una de las redes sociales más conocidas en la actualidad, con un nicho de usuarios parecido:

2.1.1. SoundCloud

SoundCloud [6] comenzó en el año 2007 como un servicio de publicación para artistas y discográficas que, con el tiempo, se convirtió en una de las redes sociales orientadas a música más conocidas y utilizadas con 175 millones de usuarios en todo el mundo [25]. No obstante, su popularidad en la actualidad se ha visto disminuida ante la llegada de nuevas plataformas competidoras como Spotify [26], o el auge de la popularidad de YouTube [5] para todo tipo de contenido, desde *vlogs*[1] hasta historias interactivas [27].

Sus principales características como red social y plataforma de publicación son las siguientes:

- Subida de archivos de música en distintos formatos (MP3, OGG, FLAC, …) [28]
- Estadísticas semanales y mensuales sobre reproducciones de tus pistas (nº de reproducciones, ubicación del oyente, nº de veces compartidas, …)
- Posibilidad de ofrecer comentarios sobre una pista en un momento concreto, útil para hacer referencia a una parte específica de la canción
- Sistema de mensajes personales, equivalente a los emails. Estos mensajes son privados y sólo los pueden ver los implicados en la conversación
- Lista de favoritos, seguidores, historial de reproducción y sección de destacados (sólo usuarios PRO)
- Posibilidad de seleccionar el tipo de licencia con la que se quiere publicar la obra: Copyright (todos los derechos reservados) o Creative Commons [29]
- Editor de metadatos y opción de compartir tu actividad mediante RSS [30]

1 *Vlog* es la adaptación de los blogs tradicionales al formato de vídeo, cada vez más popular

El modelo de negocio de SoundCloud es el conocido como *freemium*, donde el uso de la plataforma es completamente gratuito, pero éste se encuentra limitado de alguna forma, o reservando funcionalidades exclusivas para los usuarios de pago. Obtener acceso a todas las funcionalidades como usuario PRO conllevará el pago de una cuota annual de 99€ (distribuida en pagos de 8’25€/mes), o cuotas mensuales de 11€/mes. [31]

Las principales diferencias entre un usuario PRO y uno gratuito son las siguientes:

- Sin límite de publicaciones (los usuarios gratuitos tienen un máximo de 3h de audio entre todas las pistas).
- Posibilidad de reemplazar el archivo de audio de una publicación.
- Sección de Destacados en el perfil de usuario, donde exponer los 5 trabajos más representativos del artista.
- Monetización por la reproducción de tus pistas.
- Varios descuentos en algunos productos orientados a la edición de audio y estudio de música profesional.
- Descuento del 50% en la contratación del servicio SoundCloud Go+ (*streamming* de música *offline*, sin anuncios, etc)

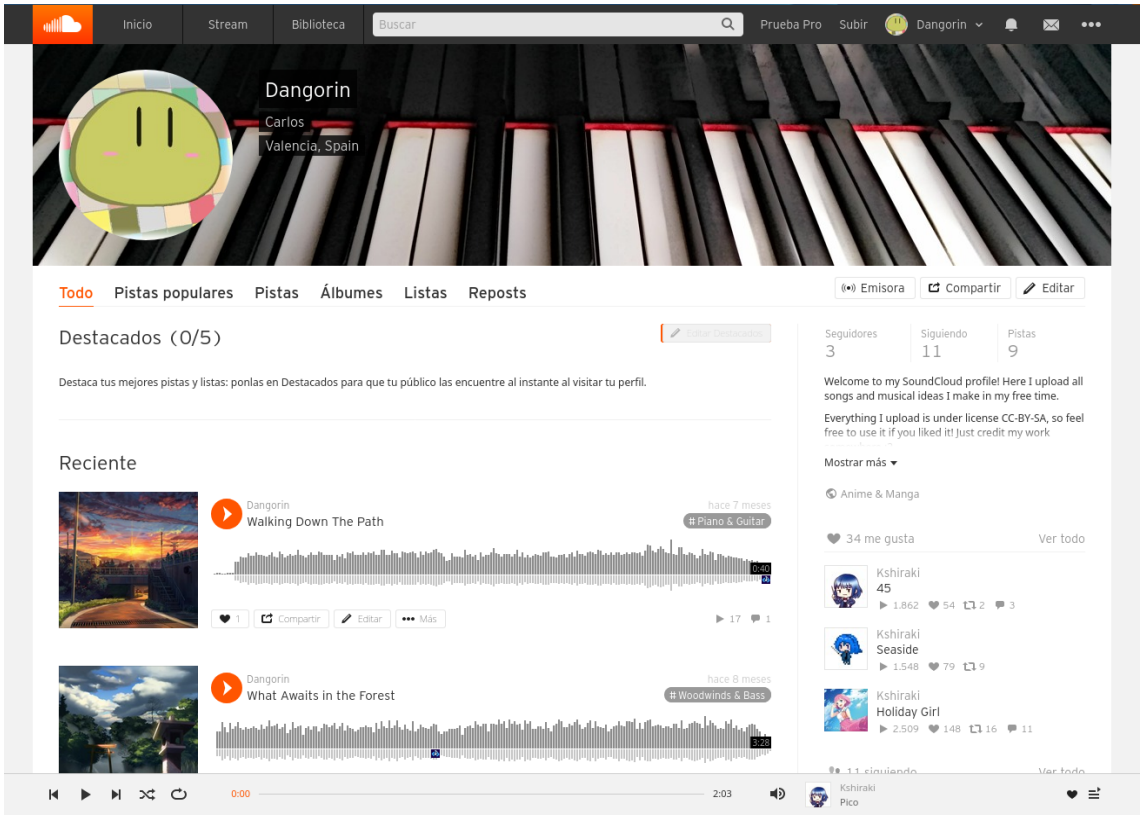

Figura 1: Ejemplo perfil de SoundCloud

2.2. Componente de edición:

Muschain incluye también una sección dedicada a la edición y composición de fragmentos musicales. Para ello, hace uso de un tipo de secuenciador conocido como *tracker* [32] en el cual es posible editar una canción como una matriz de notas musicales.

A continuación analizaremos una de las aplicaciones software más conocidas en el mundo de la edición de música estilo *chiptune* que utiliza el formato *tracker*:

2.2.1. Renoise

Renoise [33] es un software de edición de audio profesional lanzado inicialmente en el año 2002 que utiliza el formato de *tracker*, el cual se hizo muy popular en la década de los 90 gracias a MS-DOS, las tarjetas de sonido SoundBlaster [34] y el formato SoundFont [35] para dotar a los juegos con música basada en MIDI [36] de una mayor calidad sonora.

Este tipo de programas de edición de audio entran dentro de la categoría conocida como DAW (Digital Audio Workstation) [37], cuyo objetivo es ofrecer todas las características que podría ofrecer un estudio de grabación: secuenciador, mixer, instrumentos virtuales, filtros, … En la actualidad la mayoría de DAWs utilizan un formato linear y horizontal, donde cada pista se extiende a lo ancho del espacio de trabajo, a diferencia del *tracker* que sigue un formato vertical conformado por bloques o patrones.

Algunas de las características más relevantes de Renoise son las siguientes:

- Multiplataforma (Windows, MacOS, Linux)
- Soporte de múltiples formatos de audio (MP3, WAV, OGG, FLAC, …) [28]
- Edición en tiempo real gracias al soporte de drivers de audio de baja latencia (ASIO, CoreAudio, JACK) [38][39][40]
- Soporte para plugins de instrumentos virtuales y efectos de sonido (VST, AU, …) [41] [42]
- Permite trabajar y exportar audio de alta fidelidad con frecuencias de muestreo de hasta 96 kHz y profundidad de 32 bits.

El modelo de negocio de Renoise es el de producto software comercial. Una licencia completa del programa tiene un coste de 68€ + IVA, lo que nos dará acceso a todas las funcionalidades, la posibilidad de instalarlo en todos los dispositivos compatibles que tengamos, y actualizaciones gratuitas hasta la siguiente versión completa (si adquirimos la versión 2.6 tendremos actualizaciones hasta alcanzar la versión 3.6) [43]

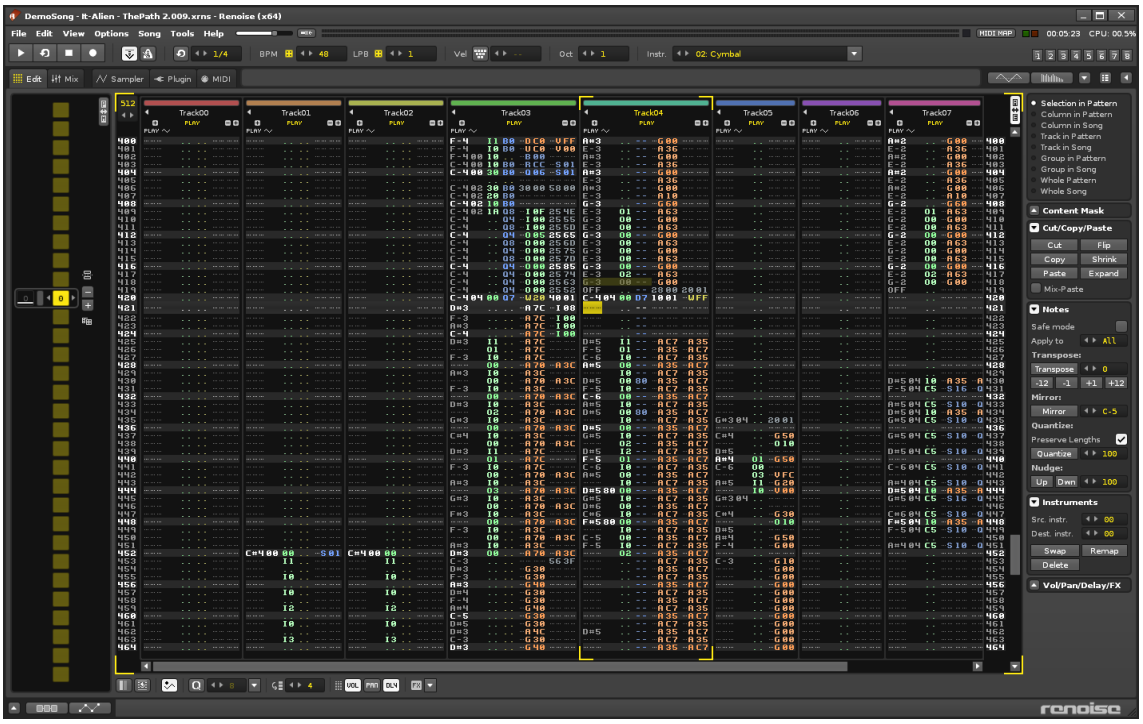

Figura 2: Ventana del modo Editor de Renoise

2.3. Componente lúdico:

Para terminar, hablaremos de la parte lúdica de Muschain, o aquella por la cual está catalogado como “juego”. Uno de los objetivos de Muschain es conseguir que sus usuarios se entretengan y pasen un buen rato participando en un juego similar al juego popular de las “palabras encadenadas”. La mecánica es bastante sencilla:

Se decide un tema concreto o una característica común a todas las palabras (verbos, sustantivos, …) y los jugadores deberán ir diciendo por turnos una palabra que cumpla con las condiciones anteriores, y que además comience por la última letra o sílaba de la palabra que haya dicho el anterior jugador. El primero que se quede sin ideas o repita una palabra que se haya dicho anteriormente, pierde.

De forma similar, en Muschain se invita al jugador a continuar una canción empezada por otra persona, pudiendo escuchar únicamente un fragmento inmediatamente anterior. Encadenando varios de estos fragmentos obtendremos una canción que se compartirá entre todos los usuarios participantes.

A continuación analizaremos un juego comercial basado también en una idea parecida:

2.3.1. Érase una vez

El juego de cartas “Érase una vez” [44] (su nombre original es *Once Upon a Time*) es un juego de mesa publicado por la compañía Atlas Games [45] en el año 1994, en el cual los jugadores deben interactuar para narrar una historia y cerrarla con un “final feliz”.

Cada jugador recibe una serie de cartas en la mano representando objetos variados, acciones, lugares, etc., y una carta especial de “Final feliz”. Se escoge al azar un jugador que será quien comience a narrar la historia y, usando las cartas que tiene en la mano para que formen parte del relato, deberá deshacerse de todas ellas para concluir el relato con la carta de Final. No obstante, el resto de jugadores podrán interrumpir al narrador si éste menciona alguna de las cartas que tengan en la mano, o si disponen de una carta especial Interrupción. Así, el turno de narrador pasará al otro jugador, quien deberá continuar la historia con las cartas que tiene en su mano. También el antiguo narrador deberá robar nuevas cartas de la baraja. El primer jugador que consiga quedarse sin cartas, y por tanto que consiga concluir el relato con la carta de Final, ganará la partida.

El modelo de negocio de *Érase una vez* se basa en la venta de un *producto base* completamente funcional, el cual puede ampliarse mediante distintas expansiones que le aportan nuevas mecánicas, mayor número de jugadores y otros elementos variados. El juego base puede adquirirse en España a través de la distribuidora EDGE Entertainment [46] a un PVP de 24’95€. Cada expansión tiene un precio que oscila entre los 9’95€ y los 12’95€.

Figura 3: Ejemplo mano de jugador Érase una vez

2.4. *Web 2.0* y el *Software as a Service*

Desde que en 1999 Darcy DiNucci hablara sobre el concepto de *Web 2.0*, y que más adelante otras personas como Tim O'Reilly lo volviesen más popular [47], muchas cosas han cambiado. Hace ya bastantes años que existen muchas alternativas a distintas aplicaciones de escritorio en forma de SaaS [48], como el caso de los ya mencionados Google Docs y Graphite Docs para las suites de ofimática al estilo de Microsoft Office [49] o LibreOffice [50].

La ventaja de los productos SaaS es que no requieren de distribución, ya que se accede a ellos a través del navegador de Internet. Por este mismo motivo, a este tipo de productos también se los conoce como *web app* [51]. Debido al éxito que presentaron las plataformas sociales, muchos de estos productos han ido incorporando más y más funcionalidades que permiten la interacción entre usuarios, llegando incluso a la posibilidad de trabajar conjuntamente en un mismo documento o proyecto en tiempo real.A su vez, representan una ventaja para el propietario del software, ya que se accede al producto de forma totalmente centralizada, y también permite actualizarlo y desplegarlo más fácilmente.

Como no podía ser de otra forma, con el paso de los años y las mejoras tecnológicas han terminado apareciendo distintas alternativas a los DAWs tradicionales en forma de *web app*. La mayoría de ellos tienen en común que su uso es completamente gratuito, limitando algunas funcionalidades para los usuarios que se subscriban a alguno de los planes *premium*. También incorporan todas las funcionalidades que cabe esperar de una red social, cumpliendo con el objetivo de disponer de *un único lugar para trabajar y compartir tus creaciones desde donde quieras, cuando quieras*. Esto pone en entredicho la supervivencia de DAWs tradicionales, además de asertar un duro golpe a plataformas como SoundCloud, que son utilizadas principalmente por artistas *amateur* o no profesionales.

A continuación analizaremos una de estas "DAW 2.0", las cuales incorporan la funcionalidad de red social además del editor, de forma similar al caso de Muschain:

2.4.1. SoundTrap

SoundTrap [52] es un estudio de música digital o DAW colaborativo basado en la nube, creado por una empresa con el mismo nombre con sede en Estocolmo, Suecia. Fue presentado públicamente como un servicio en fase *beta* en el año 2013 y en 2017 fue adquirido por Spotify, la principal plataforma de streamming de música en la actualidad. [53][54]

Sus principales características, y por las que destaca frente a otros productos, son las siguientes:

- Aplicación web basada completamente en la nube, por lo que es posible usarla desde casi cualquier dispositivo con acceso a Internet y un navegador web (ordenador, smartphone, tablet, …)
- Su uso es completamente gratuito. Permite usar el estudio sin restricciones salvo exportar el proyecto (requiere subscripción de pago)

- Dispone de 210 instrumentos virtuales, 900 *samples* y más de 150.000 efectos de sonido en su versión gratuita (los planes de pago añaden todavía más)
- Almacenamiento ilimitado de proyectos en la nube, así como sin restricciones de número de pistas/instrumentos por proyecto

El modelo de negocio de SoundTrap es el *freemium*, con varias modalidades de subscripción que oscilan entre los 95’88€ anuales (9’98€ si lo queremos mensualmente) y los 167’88€/año (17’99€/mes) para el pack más completo. [55]

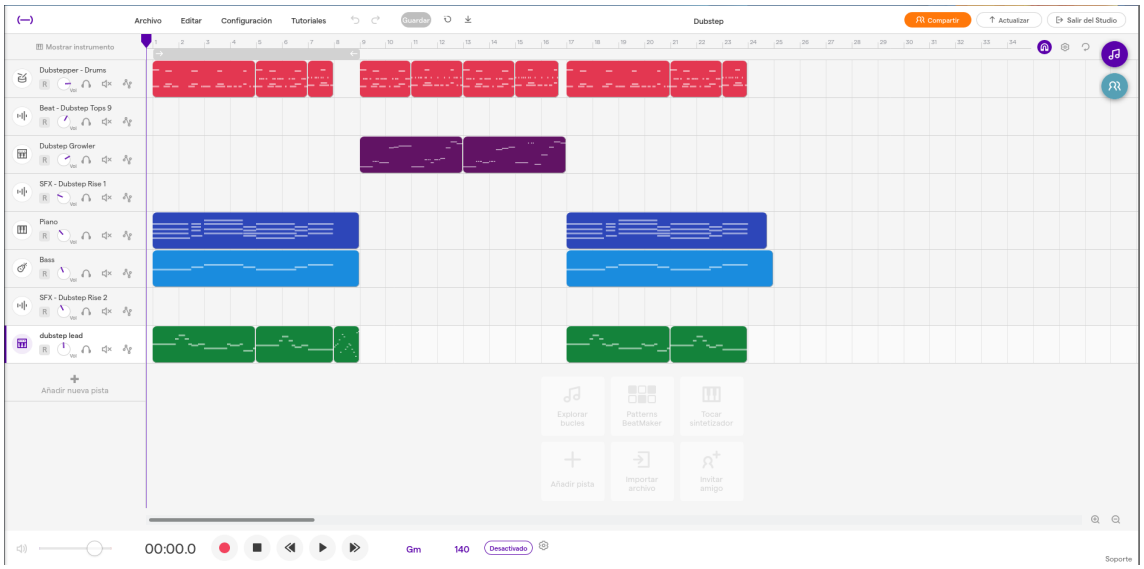

Figura 4: Vista de Editor de SoundTrap

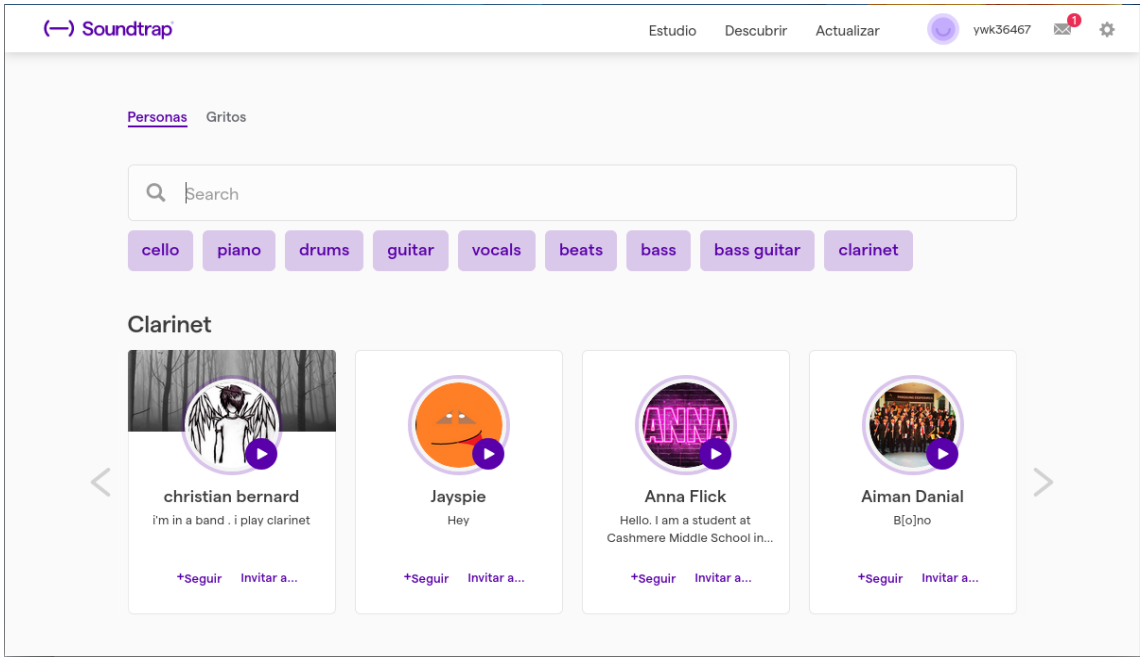

Figura 5: Apartado Descubrir de SoundTrap

3. Evaluación de la idea de negocio

Como parte de este TFG se pretende también realizar un estudio de mercado, para comprobar la viabilidad y la competitividad de Muschain. Para ello se usarán diferentes técnicas para analizar las características que debería tener Muschain, su objetivo y posibles factores externos que deberían tenerse en cuenta a la hora de su desarrollo.

Empezaremos con un estudio mediante la herramienta de Lean Canvas, seguido de un análisis DAFO, una comparativa de características con otros productos competidores, y terminaremos con una simulación de la viabilidad económica usando una proyección a 3 años.

3.1. Lean canvas:

El Lean Canvas es una herramienta para analizar diversos aspectos a tener en cuenta a la hora de empezar un nuevo proyecto, perteneciente a la metodología Lean Startup. Consiste en un lienzo dividido en 9 secciones concretas que nos permitirá sintetizar los elementos claves para el desarrollo de nuestro proyecto. Está especialmente orientada para su uso en proyectos de emprendimiento, ya que conllevan un riesgo y una incertidumbre bastante alta. A continuación se muestra el Lean Canvas aplicado al proyecto de Muschain:

2. Problemas	**4. Solución**	**7. Costes**
Existen muchas redes sociales, de muchos tipos y con audiencias diferentes. Sin embargo, en el ámbito musical la gran mayoría de ellas se resumen a compartir gustos comunes o a publicitar las canciones de los creativos. Se encuentra por tanto cierta dificultad para conocer nuevas personas con quienes poder colaborar para crear una obra conjunta.	Una plataforma en la que poder interactuar con otros músicos y compositores a través de un pequeño juego. Encadenando fragmentos musicales, se consigue obtener una obra completa, siguiendo algunas normas. Disponibilidad de perfiles de usuario, mensajes públicos y privados, lista de seguidores y favoritos… todo ello lo convertirá en una red social completamente funcional.	El principal coste sería el alquiler del servidor para alojar la plataforma. Inicialmente estaríamos hablando de unos 55€/mes, que aumentaría en función de los usuarios registrados y activos. Otros gastos secundarios serían las tarifas de publicación en la Microsoft Store y la App Store, además de los gastos asociados a marketing y el registro de un dominio. En total sumaría alrededor de unos 20.000€ para el primer año.

3. Proposición de valor Muschain busca ofrecer una nueva forma de conocer y relacionarse con músicos y compositores por todo el mundo. Más allá de escuchar sus obras, comentarlas y darlas a conocer, se pretende fomentar las colaboraciones entre usuarios a través de la creación de canciones a partir de fragmentos encadenados. Ofrece la posibilidad de compartir el resultado obtenido con el resto de los usuarios de Muschain, así como exportarlo para su uso en herramientas externas[2].	**1. Clientes** La plataforma se centra en los compositores musicales, o gente aficionada a la composición. No obstante, cualquier persona puede registrarse en la plataforma y hacer uso de ella. El perfil de usuario modelo son personas entre los 20 y los 40 años, los cuales están más acostumbrados al uso de las redes sociales. La edad mínima para registrarse es de 14 años, como en el resto de redes sociales, sin una edad máxima.	**6. Ingresos** La aplicación incluirá anuncios no intrusivos, los cuales podrán ser eliminados si el usuario se suscribe al plan anual de 9’99€. Este plan también incluye algunas funciones exclusivas como deshabilitar el límite de canciones en favoritos y la opción de exportar las secciones individuales para usarlas en herramientas externas[2]. La previsión es llegar a los 100.000 usuarios registrados en el primer año, de los cuales se espera que unos 1.000 se acojan al plan de suscripción. Con ello, los ingresos ascenderían a 10.000€ duramente el primer año, con un crecimiento futuro.
5. Canales Podrá accederse a la plataforma a través de la página web, o descargar la aplicación de escritorio que se distribuirá a través de la Microsoft Store para Windows, la App Store de Mac, y como paquete Flatpak para Linux.	**8. Métricas** Se contabilizará el número de descargas, usuarios activos, canciones registradas y/o completadas, número de mensajes publicados, ...	**9. Ventaja competitiva** La principal arma y baza de una red social es su comunidad de usuarios. Si Muschain consigue una base de usuarios antes de que aparezcan otras copias, es probable que éstos se queden por no perder el contacto con el resto de usuarios.

3.2. Análisis DAFO

El análisis DAFO es una herramienta que permite estudiar la situación en la que se encuentra un proyecto u organización, analizando distintas variables del entorno (Amenazas, Oportunidades) así como las características internas (Debilidades, Fortalezas). A continuación se presenta una matriz DAFO aplicada al proyecto de Muschain:

Debilidades	**Fortalezas**
• Orientada a un nicho de usuarios bastante concreto • El mercado de las redes sociales está bastante saturado, con muchas alternativas para escoger • Falta de personal para el desarrollo, mantenimiento y moderación de la plataforma	• La plataforma se basa en un concepto novedoso que no se encuentra en la actualidad • Su uso es completamente gratuito, y el registro es abierto •

2 Requiere de una suscripción. Sólo válido para las secciones compuestas por el usuario.

Amenazas	Oportunidades
• Poca visibilidad frente a las redes sociales mucho más grandes, dirigidas a un público más general • Apuesta arriesgada por un producto nuevo en el mercado. El éxito depende de la base de usuarios que se llegue a conseguir	• Cada vez más gente hace uso de las redes sociales en su día a día • Las personas tienden a rodearse e interactuar más con aquellos con quienes comparten su principal afición. Centrando la red social en ello, puede atraer a un buen número de gente

En la tabla anterior podemos apreciar los aspectos que pueden favorecer el proyecto, y también aquellos que juegan en su contra. Al ser un producto nuevo, no podemos beneficiarnos de la experiencia previa de otros productos similares en el mercado. Sin embargo, esa misma contra puede volverse a su favor si consigue llamar la suficiente atención: al ser un producto distinto, tardarán en aparecer apuestas similares que intenten competir con nuetra plataforma. Por ello es de vital importancia captar la atención de los usuarios potenciales desde el primer momento.

El volumen de usuarios activos nunca va a ser comparable con otras plataformas más generales, pero tampoco es lo que se busca con Muschain. El objetivo es que los usuarios utilizen la plataforma por ese factor que la hace única: el juego basado en encadenar fragmentos musicales escritos por los propios usuarios. Se incentiva a los usuarios para que compartan sus creaciones dentro de la misma red social, pero en ningún momento se les impide compartirlo en otras redes. En parte, esto mismo serviría de publicidad para la plataforma, ya que otras personas podrían sentir curiosidad y terminaran probándola.

Para mantener el interés por la plataforma puede usarse una mecánica muy recurrida en los juegos de móvil casuales, como es incluir eventos temporales que inviten a los usuarios a cumplir ciertos objetivos a cambio de recompensas. Éstas podrían ser desde elementos de personalización dentro de la plataforma, hasta cupones descuento de algún tipo. Existen posibilidades muy variadas dentro de esta técnica, ya que tanto los objetivos como sus recompensas se adecúan según el público y la naturaleza del producto.

3.3. Productos competidores

Siguiendo con el análisis de mercado para Muschain, a continuación realizamos una comparativa entre algunos de los productos y plataformas que presentan una posición de “productos competidores”. En el apartado de Estado del arte se habla sobre algunos de ellos, pero aquí realizaremos una comparativa más en profundidad con Muschain, para estudiar las características comunes y únicas de cada uno de ellos.

	AudioTool [56]	Ohm Studio [57]	Soundation [58]	SoundTrap	**Muschain**
Uso gratuito	Sí	Sí	Sí	Sí	**Sí**
Plataforma	Web	Windows. Mac	Web	Web	**Windows, Mac*, Linux**
Actualizaciones	Sí	Sí	Sí	Sí	**Sí**

constantes					
Publicación de canciones	Sí	Sí	Sí	Sí	**Sí**
Comentarios a publicaciones	Sí	Sí	Sí	Sí	**Sí**
Mensajes privados	Sí	Sí	Sí	Sí	**Sí**
Mensajes públicos	Sí	No	No	Sí	**Sí**
Grupos	Sí	No	Sí	No	**No**
Lista seguidores y seguidos	Sí	Sí	Sí	Sí	**Sí**
Editor de música	Sí	Sí	Sí	Sí	**Sí**
Guardado en la nube	Sí	Sí	Sí	Sí	**Sí**
Nº de proyectos	Ilimitado	200	10	Ilimitado	**Ilimitado**[3]
Nº pistas / instrumentos	Ilimitado	Ilimitado	Ilimitado	Ilimitado	**5**
Colaboraciones	Tiempo real, previo contacto	Tiempo real, previo contacto	Tiempo real, previo contacto	Tiempo real, previo contacto	**Sí, anónimas, emparejamiento por puntos**
Videoconferencia	Sí	No	No	Sí	**No**
Función de exportar	No	Audio[4]	Sí[4]	Sí[4]	**Sí**[4]
Sistema de recompensas	No	No	No	No	**Sí**
Eventos de temporada	No	No	No	No	**Sí**
Ingresos	Donaciones	Pago inicial, donaciones	Plan subscripción	Plan subscripción	**Micropagos, plan subscripción**

En la tabla anterior podemos apreciar que, por características y según la implementación actual de Muschain, nuestro principal competidor sería Ohm Studio, ya que se basa también en una aplicación de escritorio. No obstante, uno de los objetivos a largo plazo de la plataforma sería adaptarlo a una aplicación web, además de la posibilidad de incluir una versión para dispositivos móviles. En este caso, nuestro competidor más directo sería la plataforma de SoundTrap, la cual además sigue un modelo de negocio por subscripción, similar al de nuestro producto.

Muschain comparte con SoundTrap muchas características, de las cuales SoundTrap sólo incluye las videoconferencias como “exclusivo” respecto a Muschain. Esto sin embargo no es algo extremadamente decisivo, ya que puede hacerse uso de una plataforma externa como Discord [59], Skype [60] o productos similares, los cuales son completamente gratuitos para un uso básico de videollamadas. Otro aspecto en el que SoundTrap destaca es en la cantidad de pistas e instrumentos que incluye las versiones gratuitas y de pago (tal como analizamos en la sección 2.4.1) frente al límite de 5 pistas/instrumentos por proyecto que tiene Muschain, junto a la capacidad de edición en tiempo real.

No obstante no sería justo hacer la comparativa con números simplemente, ya que tanto el modo de trabajo en ambas aplicaciones como el objetivo es completamente distinto. El editor de Muschain es de estilo simple y minimalista, pensado para que sea fácil de utilizar. Así el usuario

3 Máximo 3 proyectos al día, acumulable

4 Requiere subscripción

se puede centrar más en la parte creativa y en la mecánica de juego, que consiste en continuar el fragmento musical anterior. En una futura sección se explicará con mayor detalle la mecánica de juego Muschain. En cambio el editor de SoundTrap es un secuenciador mucho más complejo, con infinidad de opciones y herramientas para poder realizar un proyecto musical de forma profesional.

Por otro lado, Muschain posee algunas características únicas como el sistema de recompensas por cumplir ciertas propuestas o "misiones", las cuales se renuevan cada semana, así como la inclusión de algunos eventos temáticos temporales, que coincidirán con las estaciones del año o festividades representativas de ese periodo del año. Estas mecánicas serían más propias de un juego, destacando sobre todo los sistemas de misiones de los juegos *free2play*, donde se premia al jugador que dedique tiempo a cumplir estos objetivos a cambio de algunas recompensas. Para fomentar el uso de la plataforma como forma de ocio, Muschain incluye este sistema de recompensas, siendo una característica única que lo diferencia entre el resto de competidores, como hemos mencionado anteriormente.

Además, siguiendo la moda que se observa desde hace ya unos años en la industria de los videojuegos [61], Muschain incluye también un sistema de *micropagos* o *microtransacciones*. Este concepto consiste en ofrecer un producto completo de forma gratuita, y habilitar una tienda dentro del juego en la cual los jugadores pueden adquirir distintos elementos para usar dentro del juego. Estos pueden proveer de algún tipo de ventaja para los jugadores, o simplemente cambiar el aspecto estético de un personaje u objeto. A raíz de este sistema nació el concepto de *loot boxes* o cajas de botín, donde los jugadores adquieren el equivalente a un boleto de lotería con premio seguro, pero éste se asigna de forma aleatoria de entre todos los premios posibles dentro del paquete. Esta práctica fue duramente criticada por usuarios (y sus padres, en caso de ser menores) y distintas organizaciones gubernamentales y defensoras de los derechos del consumidor, por incitar al juego y a la ludopatía. [62] Por ello mismo, para no caer en este error, los elementos que se puedan adquirir en Muschain serán mayoritariamente estéticos, salvo una excepción que se explicará con mayor detalle en la siguiente sección.

3.4. Modelo de negocio

Tal y como hemos comentado en apartados anteriores, el modelo de negocio de Muschain sería el *freemium*. La aplicación puede usarse de forma completamente gratuita, salvando algunas características para los usuarios que adquieran un plan de pago, así como limitando su uso en algunos casos. A esto le sumaremos un sistema de microtransacciones y publicidad no invasiva en la versión gratuita, para compensar los altos costes durante el inicio del proyecto.

De esta forma, las principales fuentes de ingresos de Muschain serían la cuota anual que se cobraría al usuario por la subscripción al plan *Unchained (nombre tentativo)*, junto con los ingresos por las microtransacciones. El precio de los elementos dentro de la tienda podría oscilar entre los 0'50€ y los 2€, obteniendo así una mayor probabilidad de adquisición, tal y como se aprecia en el siguiente gráfico procedente de un estudio realizado por SuperData durante septiembre de 2019 [63]:

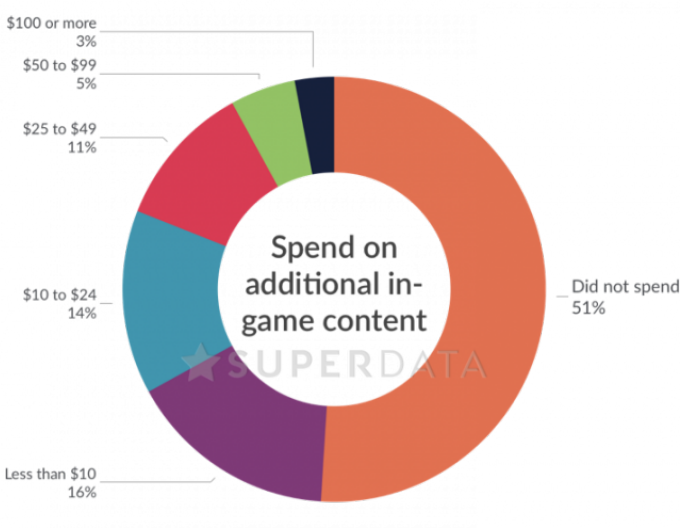

Figura 6: Gasto mensual de los usuarios en microtransacciones

Además, en otro estudio llevado a cabo por la compañía Qutee Gaming Today se observa que, irónicamente, la mayoría de los usuarios no están en contra de las microtransacciones precisamente. Cerca de un 70% de los encuestados respondió que estaban a favor de los micropagos para elementos que fueran puramente estéticos, es decir, que no aportaran una ventaja respecto al resto de jugadores. [64]

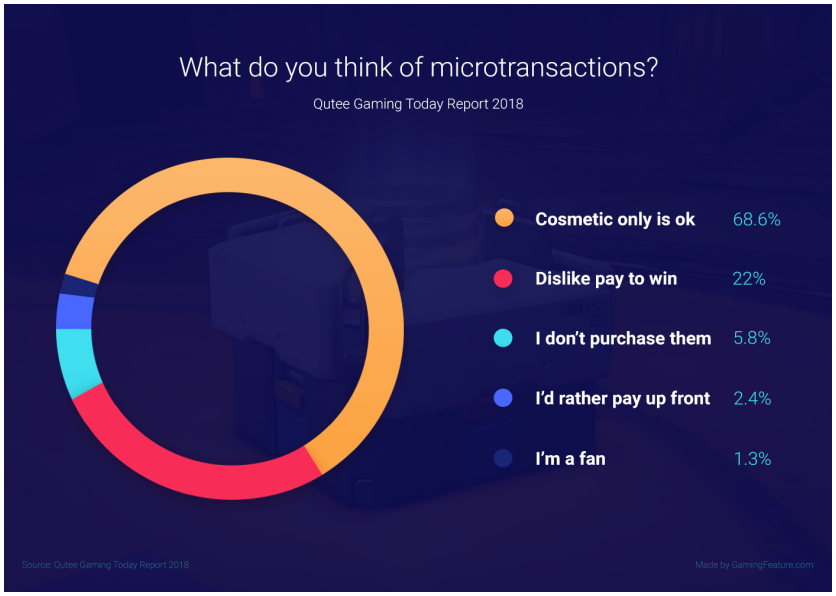

Figura 7: Estudio de Qutee acerca de las microtrasacciones

Teniendo esto en cuenta, posibles adquisiciones dentro de esta tienda serían las siguientes:

- Un pack de 10 cupones para participar en proyectos por 1€, 20 cupones por 1'5€ y 30 cupones por 2€
- Varios temas para cambiar el aspecto a la aplicación por 0'50€ cada uno

- Distintos packs de stickers para usar en el chat y en los mensajes dentro de la aplicación por 0’50€

- Insignias que se muestran junto al nombre de usuario y su avatar al lado de cada publicación que haga el usuario en la plataforma por 0’50€ cada una.

En la tabla comparativa de características entre productos se incluye una nota respecto a la capacidad *ilimitada* que tiene Muschain para los proyectos dentro de la plataforma, donde se menciona un límite de 3 proyectos diarios. Los usuarios de Muschain podrán crear o participar en hasta 3 proyectos al día mediante el uso de unos cupones, los cuales se recibirán cada día que se inicie sesión en la aplicación. Si un día no se consumen todos los cupones obtenidos, se guardarán para poder ser usados la próxima vez. Los usuarios también podrán participar en una de las misiones diarias que consiste en acumular días consecutivos de uso, por la cual podrán aumentar el número de cupones recibidos al inicio de sesión. Estas mecánicas se explicarán con mayor detalle en una futura sección.

Además, como guiño a ese 51% de jugadores que no gastó en microtransacciones y a las más que infames cajas de botín, todos estos elementos se podrían conseguir de forma totalmente gratuita como recompensa por completar distintas misiones u objetivos, o participando en alguno de los eventos de temporada. Al completar alguna misión o evento, se obtendría un elemento de forma aleatoria dentro de su categoría, de entre todos los disponibles para adquirir a través de la tienda. Algunas de las recompensas de los eventos de temporada serían exclusivas, por lo que sólo se podrían obtener participando en el evento, ya que no estarían disponibles a través de la tienda. Esto incentivaría de alguna forma que los usuarios participaran en dichos eventos.

Sobre el plan de subscripción *Unchained (nombre tentativo)*, incluiría los siguientes aspectos y beneficios respecto a un usuario gratuito:

- El número de cupones percibidos cada día que se inicie sesión aumenta de 3 a 10, pudiendo participar igualmente en el sistema de recompensas.

- Se eliminan los anuncios que se encuentran en la versión gratuita del programa.

- Se habilita la función de exportar para los fragmentos compuestos por el usuario, en forma de audio y como fichero MIDI para su uso en otros programas externos.

- Sección de destacados en el perfil de usuario, donde podrá mostrar los proyectos en los que haya participado de los cuales se sienta más orgulloso.

- Posibles descuentos en software o eventos relacionados con la música y audio *(a convenir con los patrocinadores)*

Juntando las subscripciones, las microtransacciones y la publicidad de la versión gratuita de Muschain, se espera que los ingresos de la plataforma sean lo suficientemente altos como para compensar los costes de desarrollo, mantenimiento y adición de nuevo contenido. Estos beneficios escalarían considerablemente con el número de usuarios de la aplicación, por lo que en apariencia sería un proyecto bastante viable y con un margen de beneficio alto.

3.5. Proyección económica

En el siguiente apartado realizaremos una tabla con la estimación de los costes y los ingresos asociados a la actividad de Muschain. Debido a que tanto la App Store de Apple como la Microsoft Store reclaman un 30% de comisión [65] [66] respecto a los ingresos netos que genere la aplicación como condición a la publicación en su plataforma, todos los precios indicados a continuación no tendrán en cuenta el pago de esta comisión. Esto es porque no puede hacerse una estimación correcta de los ingresos procedentes de estas dos plataformas, ya que el producto también estará disponible a través de otros medios de publicación.

	Mes 1	**Mes 6**	**Año 1**	**Año 2**	**Año 3**
Ingresos					
Subscripciones	82,92 €	3.606,88 €	24.750,63 €	83.953,13 €	185.443,13 €
Micropagos	166,67 €	8.291,67 €	62.891,67 €	259.451,67 €	608.891,67 €
Publicidad	45,00 €	2.270,00 €	17.375,00 €	72.773,00 €	171.485,00 €
TOTAL	294,58 €	14.168,54 €	105.017,29 €	416.177,79 €	965.819,79 €
Gastos					
Marketing	-4.000,00 €	-9.000,00 €	-30.000,00 €	-52.000,00 €	-74.000,00 €
Hosting	-70,00 €	-420,00 €	-840,00 €	-1.680,00 €	-2.520,00 €
Personal	-2.000,00 €	-18.000,00 €	-42.000,00 €	-102.000,00 €	-114.000,00 €
TOTAL	-6.070,00 €	-27.420,00 €	-72.840,00 €	-155.680,00 €	-190.520,00 €
Balance					
Beneficios	-5.775,42 €	-13.251,46 €	32.177,29 €	260.497,79 €	775.299,79 €
Estadísticas					
N.º usuarios	1000	9750	54600	98280	174720
N.º subscriptores	100	850	4.250	5.950	10.200
% subscritos	10,00 %	8,72 %	7,78 %	6,05 %	5,84 %
Datos de entrada y otras constantes					
Muschain *Unlimited* (año)		9,95 €	Micropago por usuario (año)		2,00 €
Salario trabajador (mes)		2.000,00 €	N.º de trabajadores		2
Gasto publicidad (mes)		1.000,00 €	Extra gasto publicidad (año)		10.000,00 €
Gasto hosting (mes)		70,00 €	Ingreso publi /usuario (mes)		0,05 €

Figura 8: Proyección económica a 3 años de Muschain

En esta tabla podemos ver una simulación de los gastos y beneficios obtenidos por Muschain durante un periodo de 3 años. Todos los importes son valores acumulados respecto al valor anterior, de esta forma se ilustra mejor la evolución económica del proyecto, y por tanto, su viabilidad en el mercado. Hemos supuesto que la plataforma se desarrolla inicialmente por un equipo de dos personas a jornada completa, cobrando un salario de 2000€ al mes. Al principio los gastos serían mucho mayores que los ingresos, en parte debido al alto coste inicial de las campañas de marketing para dar a conocer la plataforma y el gasto dedicado al personal de desarrollo. Sin embargo, a medida que el número de usuarios creciera, también lo harían los ingresos de forma mucho más pronunciada que los gastos.

En la sección anterior veíamos en el gráfico que el 49% de los usuarios había hecho alguna compra con microtransacciones durante el mes de septiembre de 2019, un 16% con un gasto inferior a $10, y un 14% con un gasto entre $10 y $24. Sin embargo, debido a que el público de Muschain es mucho más reducido, y de que se trata de un experimento, se ha decidido optar por una cifra mucho más comedida: un gasto de 2€ de media por usuario al año.

Con estos datos, vemos cómo durante el primer año los gastos serían superiores a los ingresos, pero sería a partir de entonces cuando los beneficios se incrementarían de forma notable, con una tendencia a seguir creciendo todavía más. Para suplir ese sobrecoste durante el comienzo de la plataforma, se calcula que el proyecto debería conseguir una ronda de financiación inicial de unos 100.000€, que luego podría devolverse a los inversores a partir del segundo año, cuando los beneficios obtenidos por Muschain serían lo suficientemente grandes para devolver el importe inicial junto con la comisión acordada.

4. Desarrollo de la plataforma

Para el desarrollo de Muschain se ha hecho uso de metodologías ágiles y del modelo Lean Startup, debido a la naturaleza del producto y sus condiciones. El modelo Lean Startup es una metodología de desarrollo muy flexible y adaptativa, que resulta de especial interés en proyectos de emprendimiento con un equipo de desarrollo pequeño, como es el caso de Muschain. Esta metodología propone dividir el proceso de desarrollo en una serie de pasos que exponemos a continuación:

- Identificación de las principales características del producto, así como aquellas que constituyan la base de lo que sería un Producto Mínimo Viable (siglas MVP en inglés). Esto es, una versión del producto con las funcionalidades básicas e indispensables para que pueda ser probado por clientes objetivo y que suponga una inversión de recursos mínima.

- Desarrollo del MVP y su correspondiente validación por un grupo de *early adopters*, un grupo reducido y seleccionado de usuarios que probarán el producto bajo unas condiciones concretas, para más tarde obtener sus impresiones y posibles sugerencias. Este proceso de validación determinará si el producto debe seguir su curso de desarrollo como estaba pensado inicialmente, o por el contrario sería necesario hacer algún ajuste o cambio de planes. Es también en este paso donde se determina si el proyecto resulta verdaderamente viable, o debe ser descartado por no cumplir con las expectativas puestas sobre él. Para recoger todos estos datos suele hacerse uso de encuestas, análisis de uso y reacción ante el producto, entre otros métodos.

- Tras el primer acto de evaluación, el proyecto sigue su desarrollo incorporando más funcionalidades que lo acerquen al producto final, así como incorporando los cambios y mejoras detectados durante el análisis. Cuando se alcance esta nueva versión se volverá a repetir el proceso de validación con usuarios, esta vez con un grupo de población todavía mayor. De esta forma se comprueba si los cambios propuestos y las funcionalidades introducidas han tenido una buena aceptación. Este ciclo de *desarrollo-validación* puede repetirse varias veces, cada vez acercándose más al estado final de proyecto.

Tal y como se comentó en el apartado de objetivos durante la Introducción, para este proyecto de TFG se realizarán dos iteraciones del desarrollo del producto, junto con sus correspondientes experimentos o evaluaciones. Tras ello, se analizarán los resultados obtenidos y se plasmarán en un apartado de conclusión, donde también se hablará del trabajo futuro relacionado con el producto.

4.1. Mapa de características

Editor básico	Un instrumento	Reproductor	Función de login y guardado
Mecánica de *encadenamiento*	Más instrumentos	Perfil de usuario	Lista de seguidores y favoritos
Sistema de puntos y *matchmaking*	Mensajes privados	Sistema de publicaciones	Editor más avanzado
Planificación de eventos	Función de exportar	Panel de estadísticas	Sistema de misiones y objetivos
Personalización	Tienda de productos	Cliente web	Cliente móvil

En este mapa de características están representadas las funcionalidades de Muschain, clasificadas y ordenadas por importancia y/o inmediatez de cara al primer MVP. Aquellas con un color azul más oscuro son las que cobran una mayor relevancia, mientras que las que están de color blanco se dejan para el último momento. A continuación se explica con más detalle el contenido y significado de cada una de estas características:

- **Editor básico**: La parte del editor es una de las más importantes. Es donde se va a llevar a cabo la tarea de composición, por lo que se debe prestar especial atención a la usabilidad y la comodidad para trabajar por parte del usuario. Como ya habíamos comentado, el editor de Muschain se basa en el concepto de *tracker*, por lo que se podría entender como "una matriz donde cada celda representa una nota de la canción". Esta primera versión se trata de un editor muy sencillo con una sola pista, donde es posible escoger la nota y la *octava* deseada.

- **Un instrumento**: Para reproducir el contenido del editor, es necesario asignar cada una de las pistas a un instrumento que genere el sonido a partir de las notas existentes. El resultado se emitirá a través del sistema de audio nativo del dispositivo.

- **Reproductor**: El reproductor se encargará de traducir las notas del editor a sus respectivas frecuencias, calculando la duración en función del *tempo* establecido. Tras esto, se envía toda la información al instrumento que generará el sonido.

- **Login y guardado**: Sistema de autentificación de usuario en la plataforma, y función de guardado para indicar que el fragmento musical está terminado y listo para ser compartido. Los datos se serializan y se guardan en el servidor, para que otro usuario pueda acceder a ellos más adelante y continuar la canción.

- **Mecánica de encadenamiento**: Esta es la mecánica sobre la que funciona Muschain. Cuando un usuario termina un fragmento, lo sube al servidor, e inmediatamente está disponible para el resto de usuarios. Cuando se abre el programa y el usuario inicia sesión, puede empezar un nuevo proyecto o continuar uno existente. En este segundo caso, se recupera una canción incompleta del servidor y se carga en el editor. El usuario sólo tiene acceso al fragmento anterior, el cual no podrá ser editado, sino sólo reproducido. Cuando complete el nuevo fragmento, se concatenará a los ya existentes y la canción se subirá de nuevo al servidor. Al completarse la canción, se mandará una notificación a todos los usuarios implicados para que la puedan escuchar. Ahí podrán ver qué usuario ha participado en qué parte, empezar a seguir a alguno de estos usuarios, y guardarse la canción en la lista de favoritos.

- **Más instrumentos**: Hacer melodías con un único sonido y sin *polifonía* resulta un poco limitante. Por ello se añaden más instrumentos con *timbres* diferentes, y se aumenta el número de pistas del *tracker*.

- **Perfil de usuario**: Cada usuario contará con un perfil donde podrá escribir una pequeña biografía, así como indicar links a otras redes sociales o plataformas donde también disponga de una cuenta.

- **Lista de seguidores/favoritos**: Los usuarios tendrán en su perfil la lista de los usuarios a los que han decidido seguir, así como una lista con las canciones en las que han participado y han marcado como favoritas.

- **Puntos y *matchmaking***: Para hacer el proceso de *encadenamiento* un poco más equitativo, las canciones que se asignen al usuario cuando decide continuar un proyecto existente irán en función de la puntuación que tenga el compositor. Cuando se completa la canción, tendrán la opción de otorgar a los participantes +1 puntos si les ha gustado el fragmento que han compuesto. De esta forma, alguien con una alta puntuación recibirá proyectos para continuar de otros usuarios con una puntuación similar.

- **Mensajes privados**: Un sistema de mensajería para que los usuarios puedan comunicarse entre sí dentro de la plataforma.

- **Sistema de publicaciones**: Una vez completada la canción, los usuarios pueden publicarla en el tablón principal, para que el resto de usuarios puedan escucharla. Podrán acompañarla de algún comentario, de la misma forma que el resto de usuarios podrá escribir comentarios en respuesta a la publicación.

- **Editor avanzado**: El editor básico sólo deja introducir las notas y variar el *tempo* de la canción. En una versión posterior, sería posible cambiar el volumen de cada una de las pistas, editar notas de forma individual para añadir efectos (*gain*, *pan*, *pitch*, …), añadir un sistema de historial de acciones que permita deshacer cambios, utilizar herramientas de *time stretching* para cambiar el *tempo* de la canción en un punto determinado, …

- **Planificación eventos**: A lo largo del año, organizar diversos eventos temáticos para incentivar a los usuarios a usar la plataforma. Pueden cambiarse los instrumentos disponibles para que recreen una música ambientada en el evento, recibir recompensas exclusivas ambientadas también en el evento, …

- **Función exportar**: Habilitar una opción para que los usuarios puedan exportar el fragmento en el que están trabajando como un fichero de audio o MIDI. Esto permite que el usuario utilice este fragmento en otro proyecto personal si le ha gustado el resultado o le ha inspirado para hacer otra cosa. Esta funcionalidad estará sólo disponible para los subscriptores.

- **Panel estadísticas**: A través del perfil de usuario se podrá acceder a un panel donde se recogerán diversas estadísticas de interés para el usuario. Algunas pueden ser el tiempo medio invertido para la composición de un fragmento, número de reproducciones de una canción donde se encuentre un fragmento compuesto por el usuario, tiempo total transcurrido dentro de la plataforma, …

- **Misiones y objetivos**: Para hacer la plataforma más dinámica, los usuarios podrán participar en ciertos objetivos a cambio de unos *token* o divisa interna de la plataforma, que más tarde podrán canjear por elementos disponibles en la tienda. Esto permite que todos los elementos de la tienda se puedan adquirir de forma totalmente gratuita. Sin embargo, si un usuario no quiere completar estas misiones, siempre puede adquirir los *token* directamente utilizando dinero real. Esto no producirá que un usuario que no pague se encuentre en desventaja, ya que ningún elemento disponible en la tienda otorgará al usuario un beneficio de cara al resto de usuarios, salvo los cupones de participación. De todas formas, el sistema de puntos busca premiar la calidad, no la cantidad.

  Un ejemplo de estas misiones sería premiar los días consecutivos en los que el usuario inicie sesión en la plataforma, como ya se mencionó anteriormente. Esto consistiría en aumentar el número de cupones para proyectos recibidos durante el primer inicio de sesión del día, según el número de días consecutivos que se haya hecho. De forma habitual, el usuario recibe 3 cupones cada día que inicie sesión. Si consigue encadenar 3 días seguidos, el tercer día recibirá +1 cupones; y si consigue encadenar 7 días o más, recibirá +2 cupones en cada inicio de sesión. Esta mecánica se encuentra en multitud de juegos casuales que buscan que el usuario juegue un poco cada día. Un ejemplo actual sería el sistema de “millas” del Animal Crossing: New Horinzons [67], una divisa alternativa dentro del juego que el jugador puede canjear por distintos elementos dentro del juego. La única forma de obtenerlas es participando en diferentes misiones que implican realizar algunas acciones concretas, que ya se realizarían de normal pero si está la misión activa te incentivan a hacerlo de nuevo.

- **Personalización**: La gran mayoría de los elementos que conformen la tienda y los obsequios por los eventos serán componentes que cambien el aspecto visual. Pueden ser *skins* para personalizar la interfaz de usuario, fondo para los mensajes, packs de *stickers*, avatares, insignias, firmas, …

- **Tienda**: Una tienda virtual en la que se podrán comprar *token* y canjearlos por distintos artículos. No se incluyen cajas de botín ni ningún elemento que dependa de la probabilidad. De esta forma, el usuario sabe en todo momento en qué está invirtiendo los *token* que ha obtenido.

- **Cliente web/móvil**: En última instancia, adaptar la plataforma como una *webapp* y/o una versión móvil, para así llegar a un público todavía más amplio que como aplicación de escritorio.

4.2. Elaboración del primer MVP

En el mapa de características del apartado anterior vemos cómo se encuentran ya marcadas según su prioridad y peso dentro de proyecto global. Para poder comprobar si la mecánica en la que se basa Muschain funciona, es necesario por tanto desarrollar el editor básico y todo lo relacionado con él. De esta forma, asignaríamos las siguientes características para el primer experimento:

Editor básico	Un instrumento	Reproductor	Función de guardado (local)

Debido a la falta de familiaridad con el framework gráfico y el desarrollo orientado al audio, se decidió estimar una duración de alrededor de un mes para desarrollar esta primera iteración. También esto nos invitaba a reflexionar si de verdad hubiese sido buena idea introducir todavía más características a nuestra primera versión, siendo que el desarrollo del editor + reproductor ya iba a suponer una cantidad de trabajo considerable.

Usando las características elegidas como base, se descompusieron en tareas un poco más pequeñas que quedaron reflejadas en el sistema de *Issues* y *Milestones* de GitLab, el repositorio remoto elegido para almacenar el sistema de versiones del proyecto. Este no es el verdadero cometido de las *issues*, pero servía para poder planificar las tareas que estaban pendientes de realizar, las que estaban en proceso de desarrollo y las que ya habían sido completadas. Así quedaría el listado de las *issues* en GitLab:

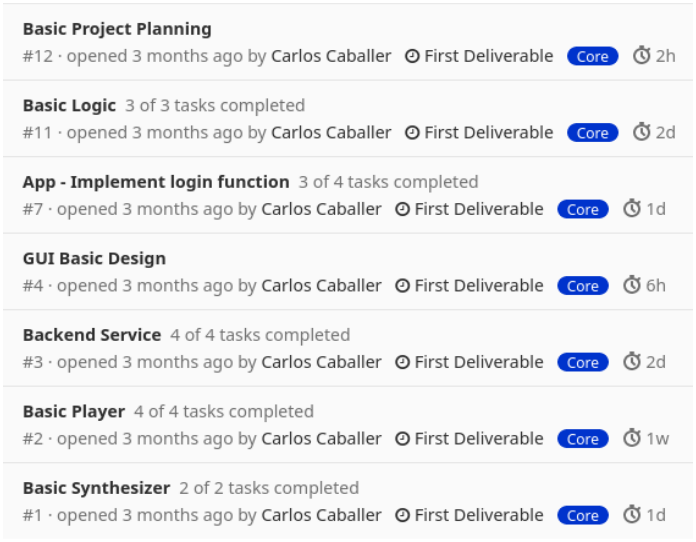

Figura 9: Backlog del primer MVP

Cada una de estas *issues* luego estaba dividida en diferentes tareas, para de esta forma poder llevar mejor el progreso del proyecto. Una vez completadas todas las tareas, se cierra la *issue*, se marca como completado el *milestone* y a continuación quedaría realizar nuestro primer experimento con un grupo de *early adopters*, para comprobar su aceptación.

4.2.1. Primer experimento

Tras completar la implementación de las características citadas en el apartado anterior, dio comienzo el primer experimento con un grupo de early adopters. Se les facilitó el ejecutable de la aplicación, junto con un formulario que debían rellenar después de haber probado un poco la aplicación de Muschain. Para este primer experimento se eligió como grupo de usuarios personas cercanas y conocidas como familiares y amigos, cuyas edades estaban comprendidas principalmente entre los 22 y los 25 años, salvando alguna excepción.

El formulario, el cual es completamente anónimo, se compone de 10 preguntas de distintos estilos:

- Respuesta binaria (sí/no)
- Selección múltiple, para señalar varios valores
- Rango numérico, para indicar el grado de satisfacción o aceptación (1 mucho, 5 poco)
- Respuesta abierta, un campo de texto para expresar una opinión o sugerencia

Además, se realizaron otras 9 pequeñas preguntas antes del formulario principal para poder separar las respuestas en diversos perfiles de usuario. Esto nos permitía conocer la opnión y el grado de aceptación tanto de usuarios objetivo como de cualquier otro tipo de usuario. Los resultados obtenidos fueron los siguientes:

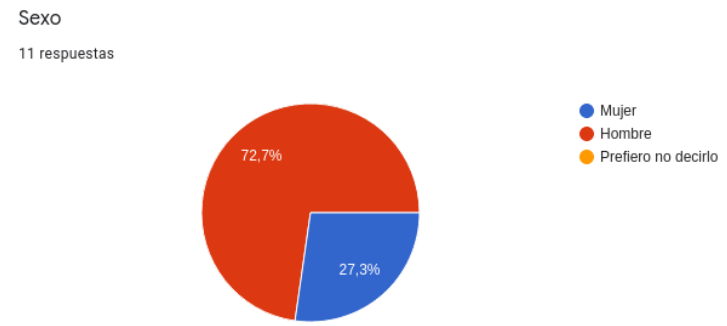

Figura 10: Sexo early adopters 1er MVP

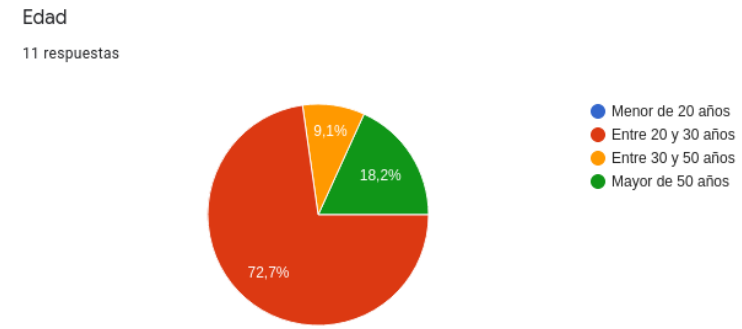

Figura 11: Edad early adopters 1er MVP

Tal y como indicábamos, la mayor parte de la población que participó en la encuesta eran hombres cuya edad estaba comprendida entre los 22 y 25 años, correspondiendo a amigos cercanos y familiares. A continuación se realizaron algunas preguntas sobre distintos hábitos:

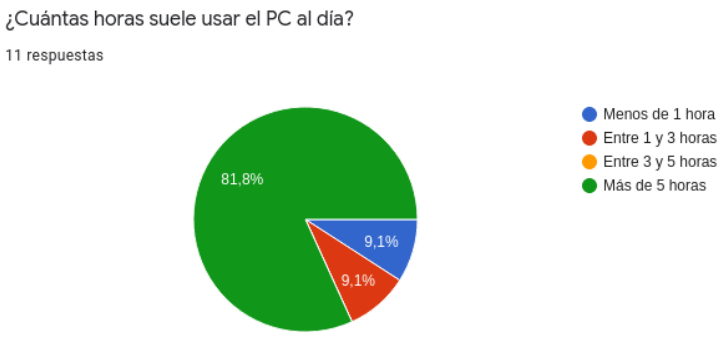

Figura 12: Horas de uso PC early adopters 1er MVP

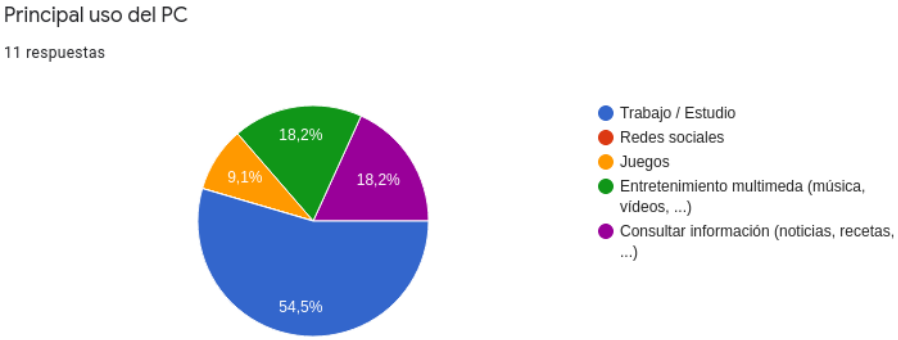

Figura 13: Motivo uso PC early adopters 1er MVP

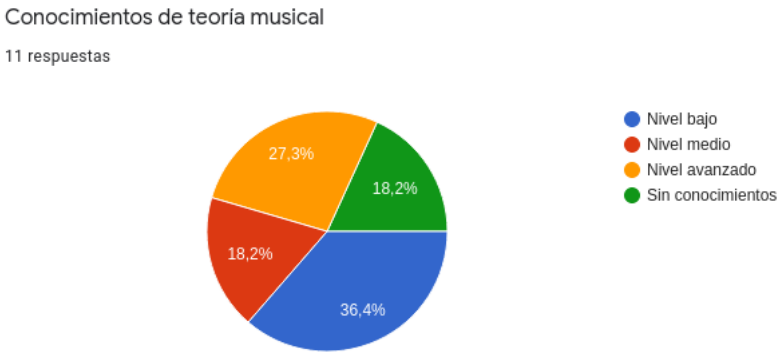

Figura 14: Conocimiento musical early adopters 1er MVP

Como resultado a estas preguntas se obtuvieron unos datos acordes a lo esperado, salvo en el uso que se le daba habitualmente al ordenador, donde pensaba que los juegos cobrarían una mayor relevancia.

Sobre los conocimientos de teoría musical, esta pregunta estaba pensada para filtrar con mayor precisión los usuarios potenciales de Muschain. Debido a que se trata de “un juego para compositores”, se esperaba que la mayor parte de los usuarios de Muschain tuvieran un mínimo de conocimientos sobre composición y teoría musical como concepto. Por ello, se preparó una segunda sección de preguntas que debían contestar **únicamente** aquellas personas que hubiesen indicado tener algunos conocimientos básicos sobre teoría musical. Los resultados fueron los siguientes:

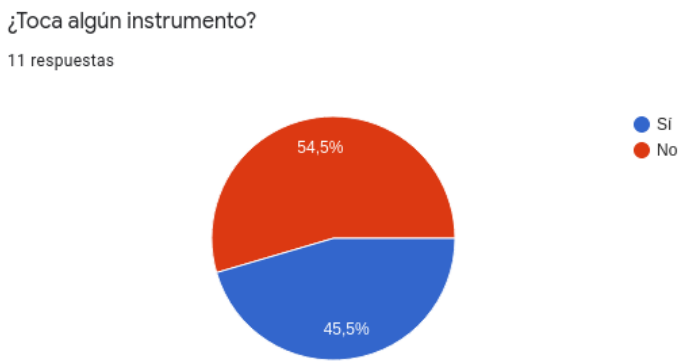

Figura 15: Intérpretes entre early adopters 1er MVP

Figura 16: Compositores entre early adopters 1er MVP

El objetivo de estas dos preguntas era separar aquellas personas que habían indicado tener conocimientos de teoría musical entre *compositores* e *intérpretes*. Sin embargo, pese a que en la cabecera de la sección estaba claramente indicado que sólo aquellas personas que tuvieran algunos conocimientos sobre teoría musical debían contestar estas preguntas, todos los encuestados marcaron alguna opción. De todas formas, estos datos nos sirven para ver que casi la mitad de los encuestados tocaba algún instrumento musical, pero sólo una pequeña parte se dedicaba a componer canciones habitualmente.

De entre las personas que contestaron que componían canciones, ahora sí, respondieron a la siguiente pregunta acerca de hábitos de composición:

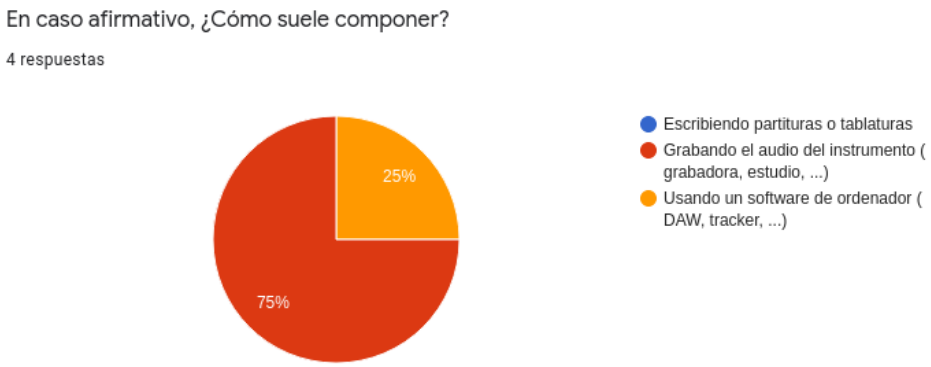

Figura 17: Modo composición early adopters 1er MVP

Como vemos, la mayoría de ellos prefería grabar el sonido del instrumento directamente.

Para terminar la sección de la encuesta relacionada con los hábitos del público con conocimientos musicales, se planteó la siguiente pregunta para estudiar la familiaridad que tenían con algunos de los productos software para edición de audio y partituas más reconocidos. Aquí se muestran los resultados:

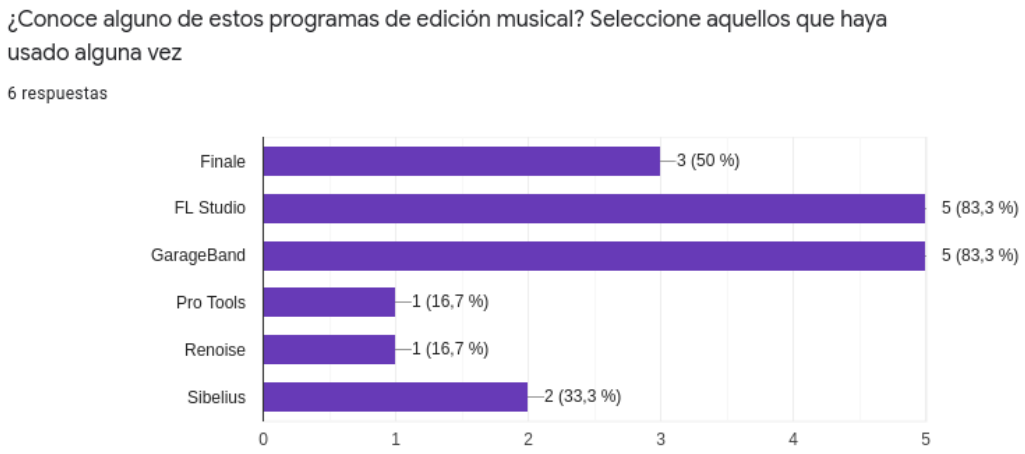

Figura 18: Uso software musical early adopters 1er MVP

De entre todos los que contestaron la pregunta, se aprecia claramente que la mayoría había trabajado con FL Studio [68] o con GarageBand [69], dos aplicaciones de estilo DAW muy populares para Windows (entre otras) y Mac/iOS respectivamente debido a su precio asequible y manejo relativamente sencillo.

Las siguientes preguntas de la encuesta ya estaban completamente relacionadas con Muschain como producto. Así, los encuestados tuvieron que responder acerca de la usabilidad, características y otras opciones sobre el modelo de negocio. Los resultados pueden apreciarse a continuación:

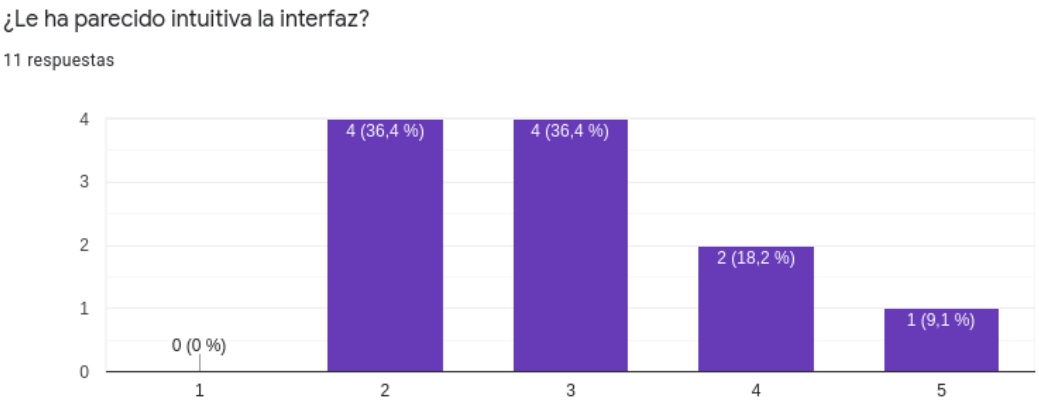

Figura 19: Accesibilidad interfaz Muschain 1er MVP

Figura 20: Manejo controles Muschain 1er MVP

Para estas dos preguntas, se usó una escala del 1 al 5, donde el 1 representaba “totalmente de acuerdo” y el 5 “totalmente en desacuerdo”. De ellas podemos concluir que a la mayoría de la gente le ha resultado bastante sencillo utilizar la aplicación, pero no están del todo satisfechos con los controles, como se aprecia en las respuestas acerca de los controles del editor. Además, se incluyó una pregunta con respuesta abierta para que los encuestados pudiesen expresar mejor su opinión al respecto:

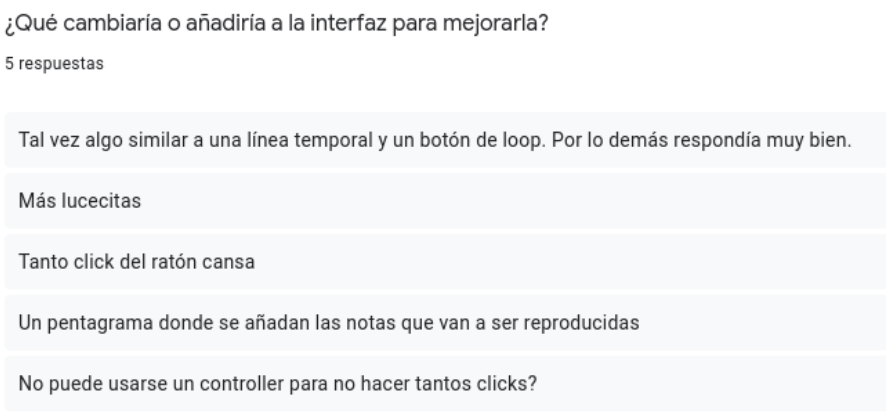

Figura 21: Sugerencias interfaz Muschain 1er MVP

Salvo una respuesta que descartamos por improductiva, el resto parecía coincidir en que la interfaz necesitaba algún retoque, sobre todo en cuanto a la interacción con el usuario. Dos personas mencionaron que se dependía en exceso del uso del ratón, mientras que otras dos personas proponían algunas características que, pese a ser útiles, diferían un poco del concepto conocido como *tracker* en el cual estaba inspirada la interfaz de usuario de Muschain.

A continuación se planteaban un par de preguntas sobre algunas características que se pensaban incluir para la siguiente versión de Muschain. Estas fueron las respuestas:

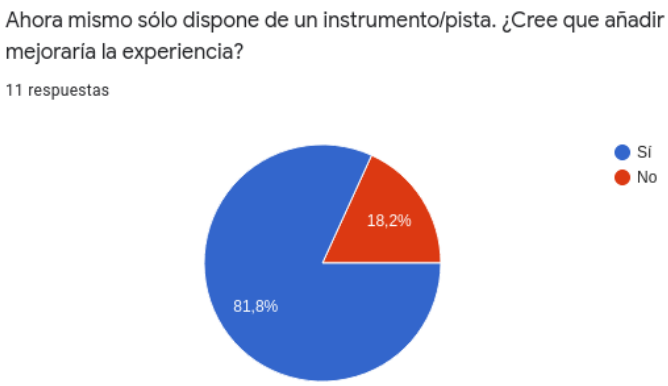

Figura 22: Más instrumentos en Muschain 1er MVP

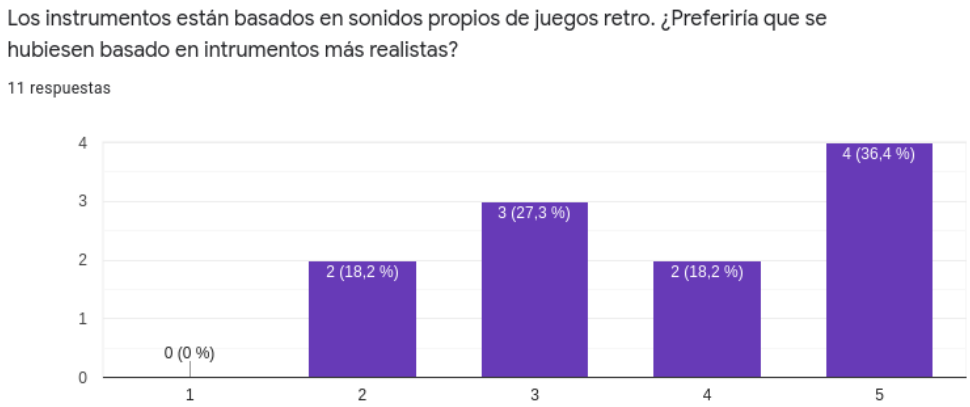

Figura 23: Realismo instrumentos Muschain 1er MVP

La mayor parte de los encuestados consideró que ofrecer una variedad más grande de instrumentos podría favorecer a la experiencia de uso de Muschain. A su vez, también se les preguntó acerca del tipo de instrumentos. En la primera versión, sólo se disponía de un único instrumento de tipo *Pulse*, que consiste en un sintetizador que emite una onda de forma sinusoidal. Como vemos en el resultado de la pregunta acerca del realismo de los instrumentos, casi todos los encuestados parecieron estar contentos con el sintetizador implementado.

El resto de preguntas de la encuesta estaban orientadas a confirmar el modelo de negocio establecido para Muschain. Se hicieron algunas preguntas sobre el precio de la subscripción, los micropagos dentro de la aplicación, entre otros. A continuación mostramos los resultados:

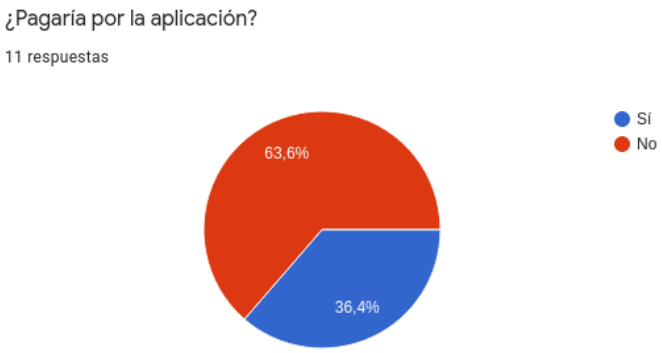

Figura 24: Intención compra Muschain 1er MVP

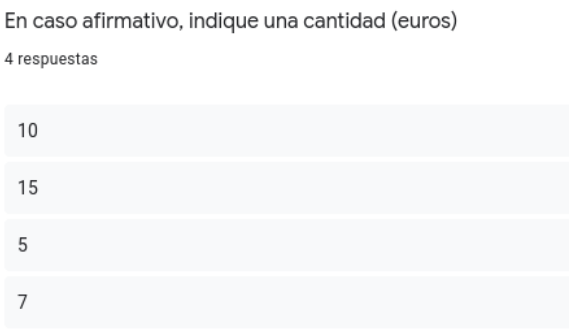

Figura 25: Sugerencia precio Muschain 1er MVP

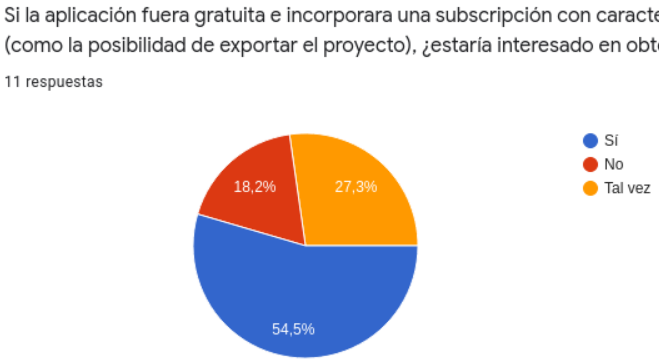

Figura 26: Aceptación subscripción Muschain 1er MVP

Estas tres preguntas tenían como objetivo estudiar si el modelo de negocio *freemium* había sido el acertado. Efectivamente, un mayor porcentaje de los encuestados consideraba que no estaría dispuesto a pagar por la aplicación. Sin embargo este número no fue tan grande como se esperaba. Además, de los usuarios que dijeron estar dispuestos a comprarla, sugirieron unos precios que se acercaban e incluso superaban el precio estimado para la subscripción anual del plan *Unchained (título tentativo)*, valorado en 9’95€/año.

A continuación se realizaron más preguntas acerca del modelo *freemium*, las subscripciones y los micropagos:

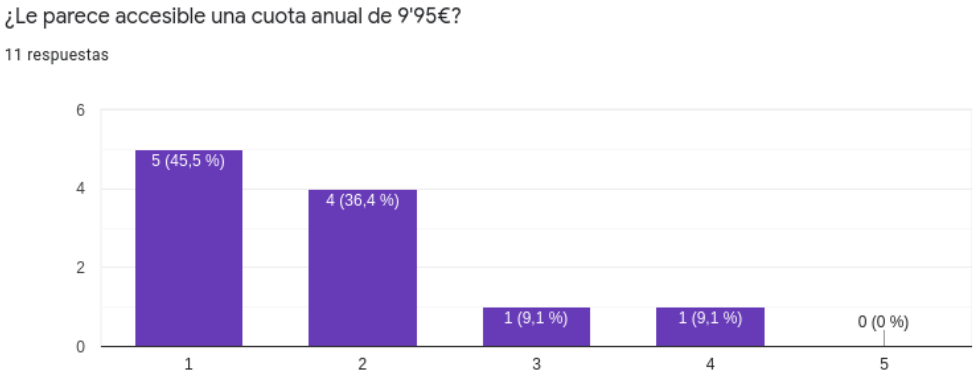

Figura 27: Aceptación precio subscripción Muschain 1er MVP

Casi todos los usuarios encuestados presentaron un interés en el modelo de subscripción. Como es de esperar, todo dependerá del precio de la subscripción y las características que incluya. Acerca del precio de la subscripción una gran mayoría parecía contenta con el precio establecido. Esto nos daba una esperanza en que el modelo de negocio por subscripción podía funcionar.

Para terminar, se realizó una pregunta acerca de los micropagos y su naturaleza dentro de Muschain. Aquí se muestran los resultados:

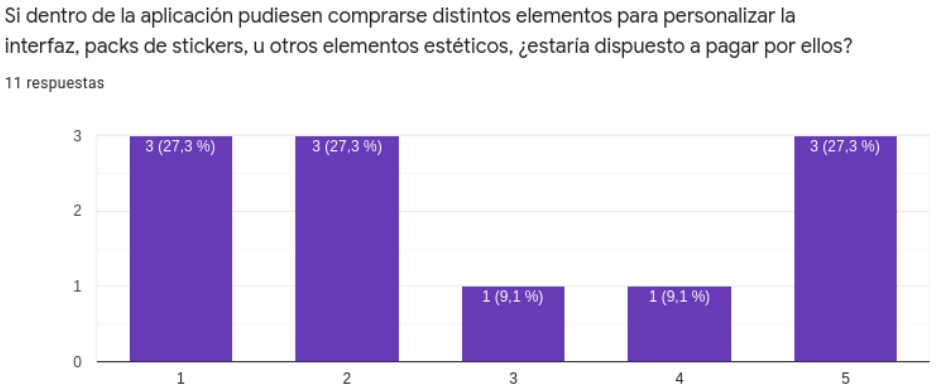

Figura 28: Aceptación micropagos Muschain 1er MVP

En esta pregunta se aprecia un poco de disparidad entre ambos extremos, aquellos que están de acuerdo con las microtransacciones y aquellos que están en contra. No obstance el porcentaje de usuarios encuestados que se mostraba a favor era mayor que los que estaban en contra. Esto corrobora en cierta medida el estudio realizado por Qutee sobre las microtransacciones [64] donde casi el 70% de los encuestados se mostraban conformes con las microtransacciones, siempre que éstas ofrecieran cambios puramente estéticos que no dieran ninguna ventaja respecto a los usuarios que no habían pagado.

4.2.2. Conclusiones

Tras obtener los resultados de la encuesta del primer experimento, podemos destacar algunos puntos para tener en cuenta de cara al desarrollo del segundo MVP:

- Pese a que la interfaz ha resultado ser bastante intuitiva, muchos de los encuestados no parecieron encontrarse excesivamente cómodos trabajando con los controles del editor. Como posible mejora de cara a la siguiente versión de Muschain, podría estar interesante habilitar atajos de teclado para, de esta forma, reducir el número de clicks de ratón necesarios para realizar ciertas acciones.

- Añadir más instrumentos parece ser una opción bien recibida por todos los encuestados. Además, el estilo de los instrumentos basados en formas de onda o *waveforms* básicas producidas por un sintetizador ha tenido buena acogida, por lo que se incluirán otras *waveforms* como la cuadrada (*Square*), triangular (*Triangle*) y la dentada (*Sawtooth*).

- La subscripción *Unchained (nombre tentativo)* ha tenido buena acogida, así como la opción de los micropagos dentro de la aplicación. De todas formas, esto se observará con mayor claridad en el siguiente experimento donde participarán más personas, llegando incluso a salir un poco del círculo de conocidos.

4.2.3. Estado de Muschain durante el primer experimento

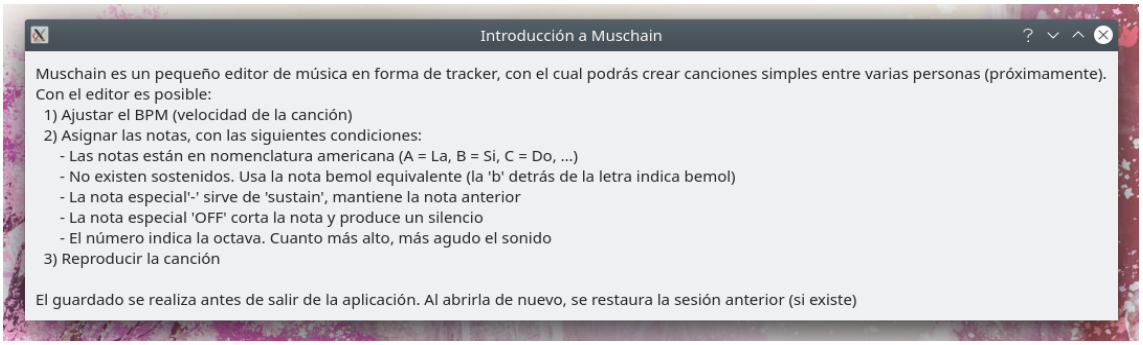

Figura 29: Guía introducción Muschain 1er MVP

Nada más abrir la aplicación de Muschain nos encontraremos con este diálogo que nos explicará un poco el funcionamiento básico del editor. Como en la encuesta iban a participar también algunas personas que no estaban familiarizadas con la notación musical americana ni el funcionamiento de un *tracker*, se optó por incluir esta pequeña guía para agilizar un poco la toma de contacto.

Al mismo tiempo, la ventana principal que contiene el editor aparece en segundo plano, y lo único que debemos hacer para poder empezar a utilizar la aplicación es hacer click sobre ella para cambiar el *focus* con el ratón. Para esta primera versión se incluye un editor muy básico, por lo que no podremos añadir más notas del máximo establecido (16 en este caso), ni tampoco añadir más instrumentos o reemplazar el existente. A continuación se muestra el editor:

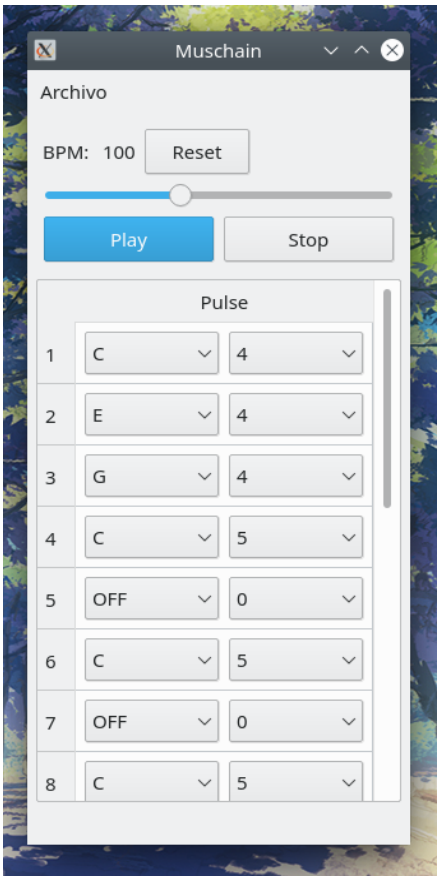

Figura 30: Vista Editor Muschain 1er MVP

Como se aprecia, la interfaz es bastante simple constando únicamente del componente controlador del reproductor (botones *Play/Stop*, selector de BPM y botón de *Reset* para devolverlo a su valor original) y el componente del editor, donde podremos seleccionar las notas que compondrán nuestro fragmento musical.

En la barra de menú tenemos acceso a dos opciones desde el menú 'Archivo'. Una es mostrar de nuevo la ventana de Introducción, para que nos pueda resolver alguna duda que surja acerca del funcionamiento, y otra para cerrar el programa y salir de la aplicación. Como se indica en dicha introducción, la opción de guardado se ofrece justo antes cerrar el programa. Aparecerá una ventana de diálogo donde podremos guardar la sección y terminar, descartar los cambios o cancelar la operación. La siguiente vez que abramos el programa se restaurará el estado en caso de haber elegido la opción de 'Guardar'.

Para la siguiente versión de Muschain, como ya incluirá el servicio de *backend* para comunicarse con el servidor, será posible guardar la sección con la opción de 'Guardar', así como empezar una nueva sección sin tener que reabrir la aplicación.

4.3. Elaboración del segundo MVP

Tras los resultados del primer experimento, continuó el desarrollo de Muschain añadiendo nuevas características según lo previsto, así como mejorando algunos aspectos del programa en base a los resultados obtenidos. Así, los puntos sobre los que se basa esta segunda ronda de desarrollo son los siguientes:

Reproductor (mejora y correción de errores)	Función de login y guardado (en el servidor)	Mecánica de *encadenamiento*	Más instrumentos

Para este segundo experimento se estimó un tiempo de desarrollo de dos semanas, con el objetivo de evitar cualquier error que pudiera provocar un estado no deseado en los datos del servidor, como una canción bloqueada por un usuario que nunca se ha marcado como 'completada', la sobreescritura de una parte de la canción por otra hecha por otro usuario, …

Como la plataforma de Back4App usada para simular la parte del servidor no permite realizar nada más complejo que una *query* compuesta a la base de datos, toda la lógica de bloqueo/liberación de las canciones debía hacerse en la aplicación, lo cual daba pie a muchos errores como los comentados anteriormente. Por ello se dedicó mucho tiempo al *testing* para asegurar que no se produjeran errores. Tras finalizar la fase de *testing* dio comienzo el segundo experimento.

4.3.1. Segundo experimento

Igual que con el primer experimento, se repartió el ejecutable a un grupo de *early adopters* junto a una encuesta que debían rellenar después de haber probado el programa. Esta vez, este grupo de personas fue mayor, incluyendo familiares, amigos, conocidos, e incluso personas de fuera de mi grupo de contactos.

Para que resultara más fácil compartir el programa, se subió a una cuenta de Drive y se compartió un enlace público para que pudieran descargarlo todos aquellos que tuvieran el link de descarga. Sin embargo esto empezó a dar problemas, porque Google tachaba el ejecutable como 'archivo infectado' dificultando que la gente pudiera descargarlo, hasta llegar al punto de que fuera retirado por, según ellos, "violar los términos y condiciones del servicio". Al final se decidió comprimir el ejecutable en un fichero ZIP encriptado con contraseña, para que el servidor de Google no pudiese escanear el contenido del archivo. Luego por otro lado hubo también problemas con Windows 10, ya que también marcaba el archivo como 'infectado' y saltaba el SmartScreen, donde era necesario hacer click en "Más información" para que permitiera ejecutarlo. Pese a esto, un total de 24 personas contestaron la encuesta.

Igual que en el anterior experimento, la encuesta constaba de tres secciones: Una sección general para clasificar a los usuarios según su edad, hábitos o sexo; una sección relacionada con sus conocimientos musicales, y una última sección para valorar la experiencia con Muschain. Se utlizaron preguntas binarias, de selección múltiple, de valoración numérica entre el 1 y el 5 (siendo 1 ‘Totalmente de acuerdo’ y 5 ‘Totalmente en desacuerdo) y de respuesta abierta.

A continuación mostramos los resultados obtenidos:

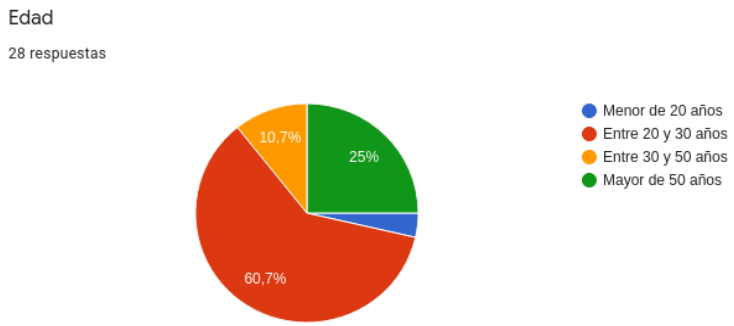

Figura 31: Edad early adopters 2do MVP

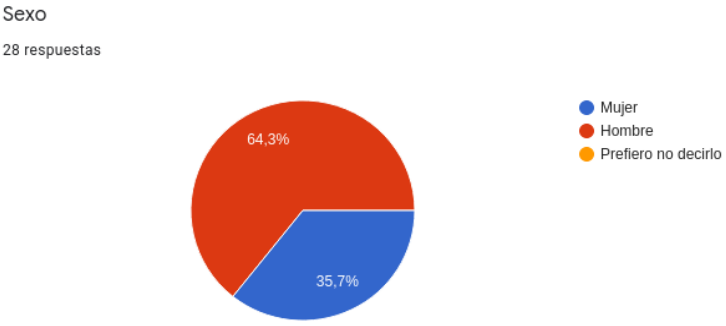

Figura 32: Sexo early adopters 2do MVP

Tal y como se aprecia en estas gráficas, la población que participó en la encuesta siguen siendo mayormente hombres de entre 20 y 30 años, igual que durante el primer experimento. Sí notamos un ligero aumento de personas mayores de 50 años al haber invitado a participar también a familiares lejanos, así como la presencia de una persona menor de 20 años. Seguimos con las preguntas relacionadas con los hábitos de uso del PC y los conocimientos musicales:

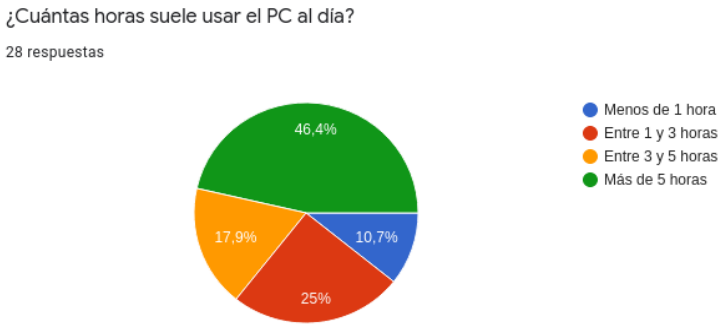

Figura 33: Horas de uso PC early adopters 2do MVP

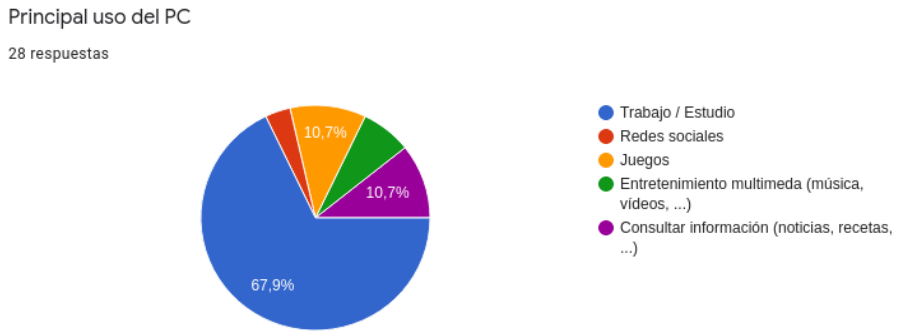

Figura 34: Motivo uso PC early adopters 2do MVP

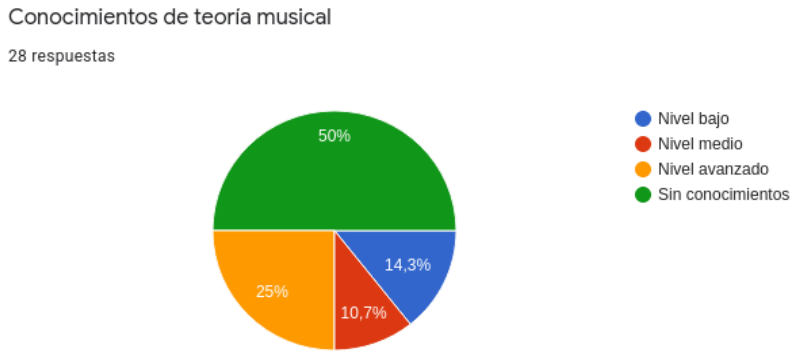

Figura 35: Conocimiento musical early adopters 2do MVP

Si comparamos los resultados con aquellos obtenidos durante el primer experimento, observamos que las horas de uso del PC se reducen. El uso mayoritario del ordenador sigue siendo para trabajar o estudiar, y la proporción de personas sin conocimientos musicales que contestaron la encuesta aumenta. Esto se debe a que se pidió a algunos conocidos que difundieran el programa y la encuesta más allá del círculo de contactos.

En la segunda sección de la encuesta estaban las preguntas relacionadas con los conocimientos musicales. Los resultados fueron los siguientes:

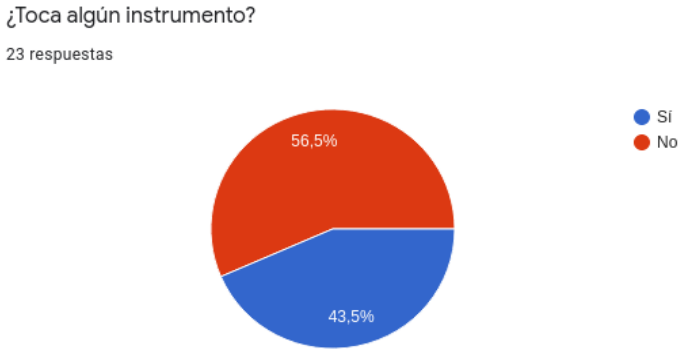

Figura 36: Intérpretes entre early adopters 2do MVP

Figura 37: Compositores entre early adopters 2do MVP

Como ya ocurrió durante el primer experimento, muchas personas que habían seleccionado 'Sin conocimientos' en la cuestión relacionada con el nivel de conocimiento teórico musical también contestaron estas dos preguntas, pese a que en la cabecera de la sección del cuestionario se indica que **sólo deben hacerlo aquellos que hayan indicado tener conocimientos musicales**.

Las siguientes cuestiones buscaban conocer un poco los hábitos de los músicos participantes. Aquí se muestran los resultados:

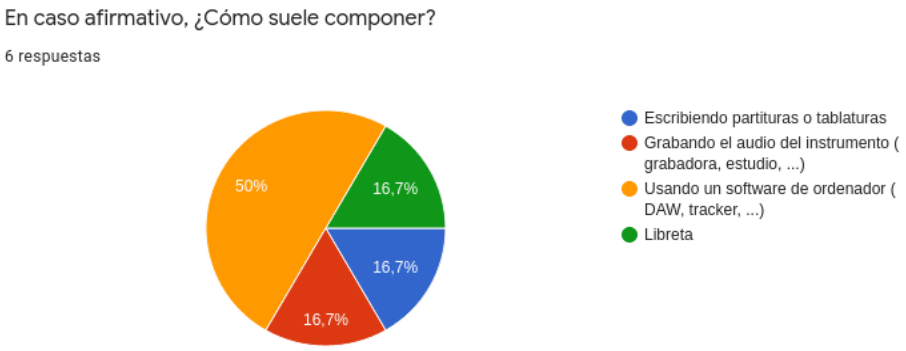

Figura 38: Modo composición early adopters 2do MVP

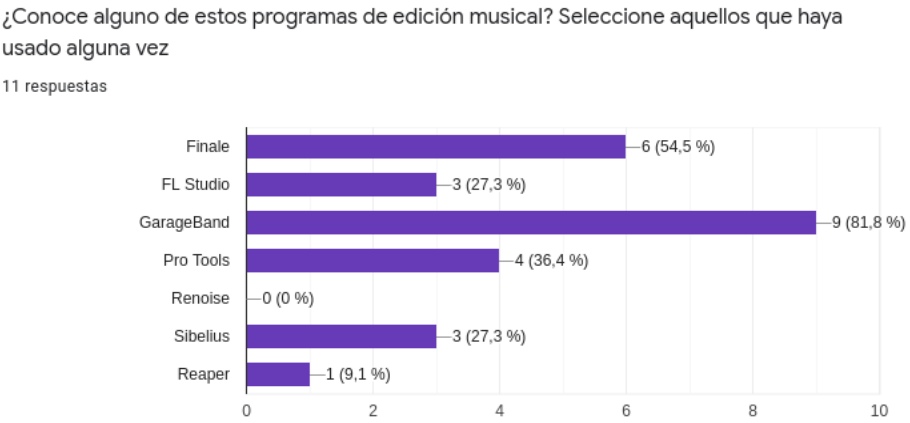

Figura 39: Uso software musical early adopters 2do MVP

A diferencia del primer experimento, esta vez más gente contestó que usaba DAWs y otro software de edición de audio digital como principal método de composición. También curiosamente una persona añadió una opción 'Libreta', por lo que entraría dentro de la opción 'Escribiendo partituras', siendo la segunda opción más votada. El software GarageBand de Apple sigue siendo el más conocido de todos, y también aquí tenemos una nueva opción añadida por un participante: el software DAW Reaper [70].

La sección final de la encuesta estaba dedicada a valorar la experiencia de uso de Muschain. Se preguntó a los usuarios acerca de la interfaz gráfica, los controles, características y modelo económico. Los resultados fueron los siguientes:

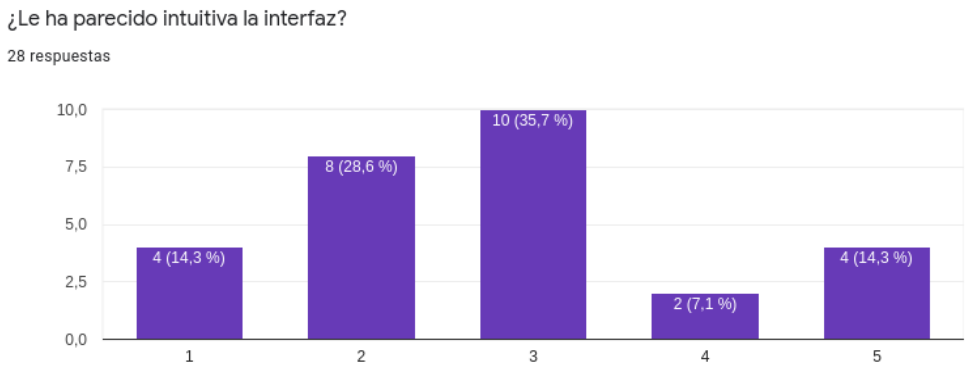

Figura 40: Accesibilidad interfaz Muschain 2do MVP

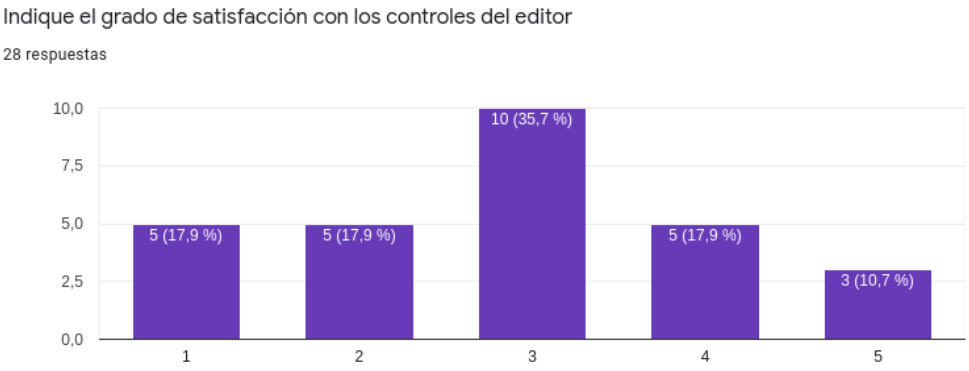

Figura 41: Manejo controles Muschain 2do MVP

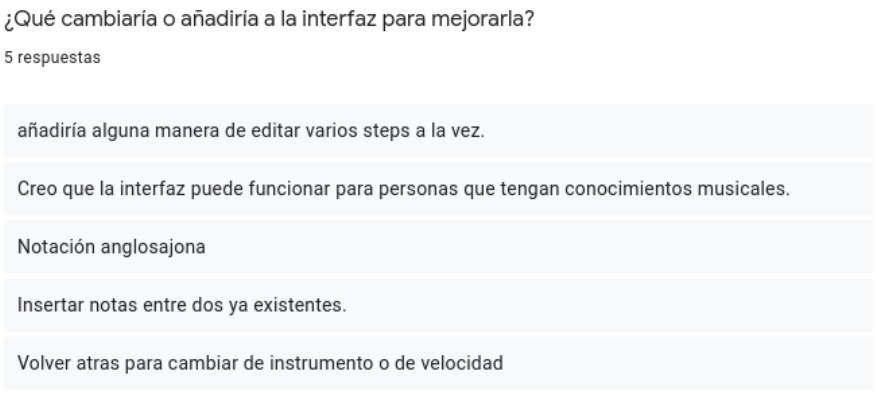

Figura 42: Sugerencias interfaz Muschain 2do MVP

Pese a que las opiniones sobre los controles de la interfaz son mayormente positivos, el número de opiniones negativas aumentó con respecto al primer experimento. Esto puede deberse también a una mayor complejidad dada por la introducción de hasta 4 pistas simultáneas, y una sección para previsualizar y escuchar el fragmento anterior de la canción, con el fin de tener un punto de partida para continuarla. También esta vez algunas personas dejaron comentarios para mejorar la interfaz. Funciones como 'Deshacer y rehacer', o la 'Selección múltiple' pueden ser muy interesantes para agilizar el proceso de composición. Otra mejora podría ser dejar elegir al usuario la nomenclatura de las notas entre el sistema europeo (Do, Re, Mi, …) y el americano (A, B, C, …).

Siguiendo con la valoración de la experiencia en Muschain, se realizaron un par de preguntas para confirmar la aceptación de las nuevas características introducidas para este segundo experimento:

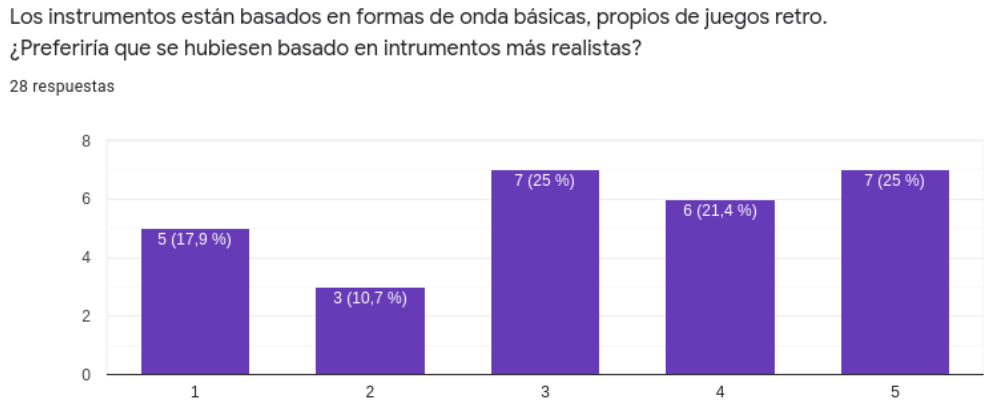

Figura 43: Realismo instrumentos Muschain 2do MVP

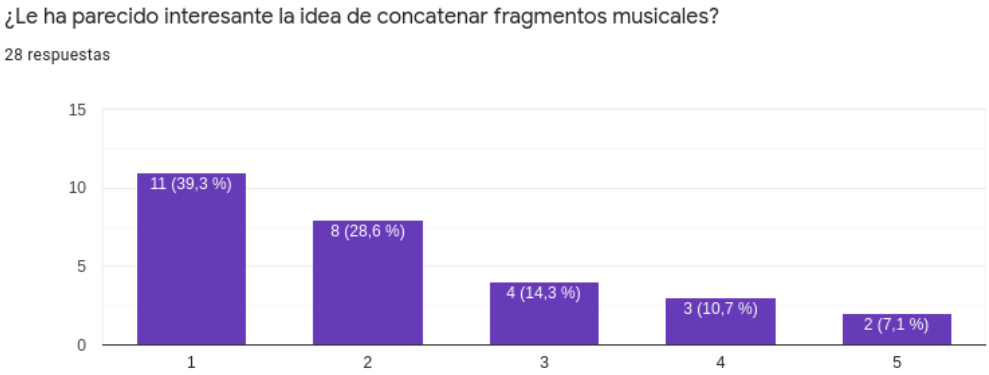

Figura 44: Aceptación encadenamiento Muschain 2do MVP

Como vemos en las gráficas, la mayoría de los usuarios encuestados quedó contento con el tipo de instrumentos que se introdujeron para la segunda versión de Muschain, y la mecánica de *encadenamiento* pareció gustar entre los participantes. También se aprovechó la ocasión para realizar una tercera pregunta, esta vez sobre un par de características que se introducirían para la siguiente versión del programa:

Figura 45: Aceptación sistema puntos Muschain 2do MVP

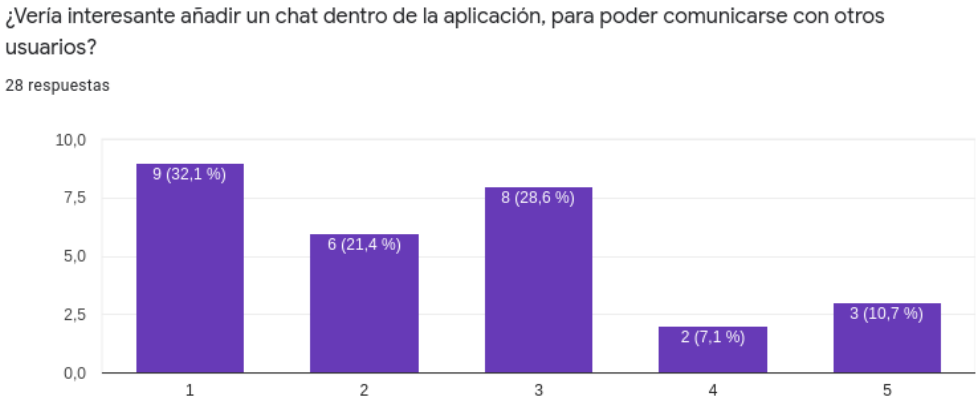

Figura 46: Aceptación servicio mensajería Muschain 2do MVP

Ambas características fueron bastante bien recibidas, sobre todo el sistema de puntos para emparejamiento. Esto permitiría que las canciones resultantes fueran un poco más coherentes y los usuarios menos experimentados en la composición no se sintieran 'intimidados' si les toca continuar una pieza compleja.

Las cuestiones restantes buscaban consolidar la idea de negocio de la aplicación, por lo que se utilizaron las mismas cuestiones que en el anterior experimento. A continuación mostramos los resultados obtenidos:

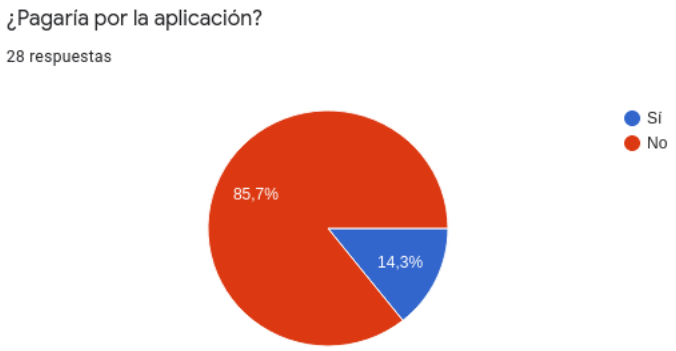

Figura 47: Intención compra Muschain 2do MVP

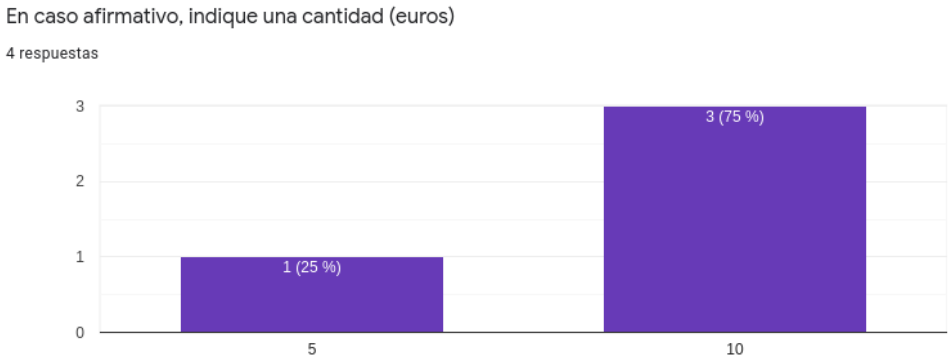

Figura 48: Sugerencia precio Muschain 2do MVP

Igual que en el primer experimento, la mayoría de los usuarios no estaba dispuestos a comprar Muschain. Tan solo 4 personas indicaron su intención de compra, sugiriendo un precio de unos 10 euros.

Para terminar la encuesta, se realizaron un par de preguntas sobre la subscripción *Unchained (nombre tentativo)* y sobre las microtransacciones dentro de la aplicación. Estos fueron los resultados:

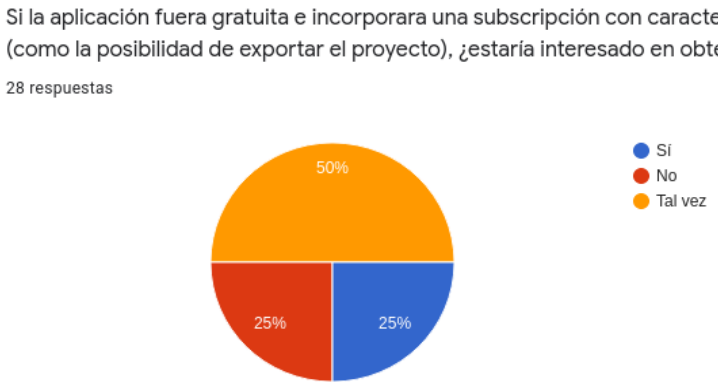

Figura 49: Aceptación subscripción Muschain 2do MVP

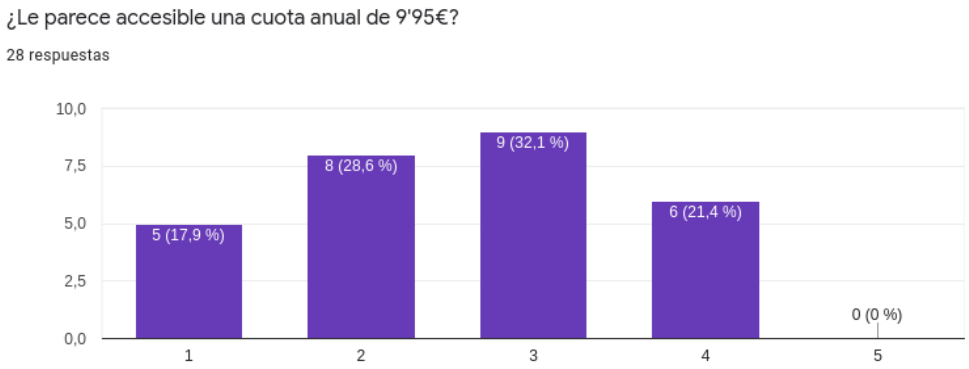

Figura 50: Aceptación precio subscripción Muschain 2do MVP

En contraste con el primer experimento, pese a que la respuesta seguía siendo positiva mayormente, no hubo tantas personas que mostraran un completo interés en obtener la subscripción. De igual forma, la valoración sobre el precio de la subscripción también cayó aunque las opiniones seguían siendo más positivas que negativas.

Donde se produjo un cambio radical fue con el tema de las microtransacciones, como vemos en el siguiente gráfico:

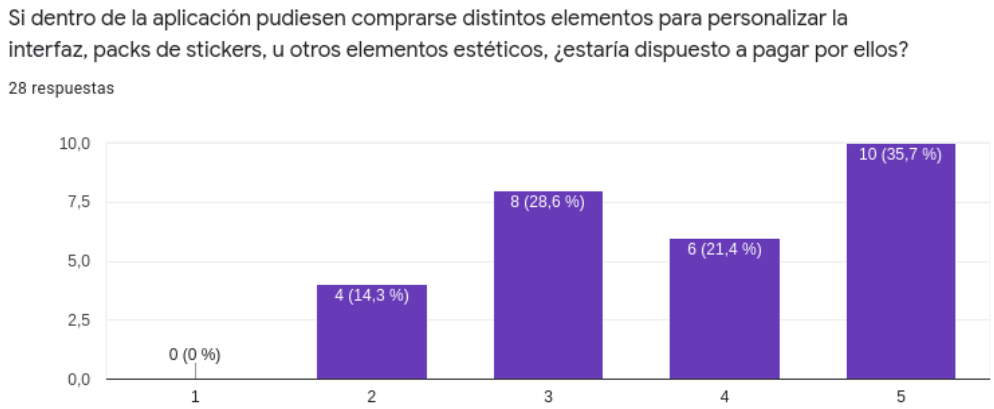

Figura 51: Aceptación micropagos Muschain 2do MVP

A diferencia de la anterior encuesta donde los resultados fueron un poco más equilibrados, ligeramente a favor de este modelo de negocio, esta vez la aceptación fue bastante más negativa. La mayor parte de los usuarios indicó no estar interesado en estos elementos, lo cual será un tema para analizar en la conclusión de este segundo experimento.

4.3.2. Conclusiones

Después de analizar los resultados obtenidos en la encuesta del segundo experimento, podemos destacar algunos puntos clave para tener en cuenta de cara al futuro desarrollo de Muschain:

- Tanto la interfaz como los controles del editor necesitan pulir muchos detalles para hacer que su uso sea más simple, cómodo y a fin de cuentas más agradable para los usuarios. El uso de atajos de teclado, opciones de ‘deshacer y rehacer’ y la personalización de los ajustes de los controles contribuirán a que el usuario se sienta más cómodo usando Muschain, y con ello la valoración general del programa aumentará.

- Añadir nuevas características como el sistema de *matchmaking* y el servicio de mensajería conseguirá que la interacción de los usuarios dentro de la plataforma sea más rica, contribuyendo también a darle vida a la plataforma.

- Del último apartado de la encuesta referente al modelo de negocio podemos extraer la siguiente hipótesis: Debido a que en esta segunda encuesta participaron más personas que no tenían relación con la composición y la teoría musical, es posible que el programa resultara más difícil de comprender y manejar para ellos y careciera de interés real. Con esto se nos recuerda la importancia de identificar de forma clara y precisa el *target* de usuarios al cual va dirigido nuestro producto.

4.3.3. Estado de Muschain durante el segundo experimento

Debido a que para esta segunda versión se incorporaba el servicio de *backend* para guardar las canciones en el servidor y habilitar la mecánica de *encadenamiento*, la primera ventana que nos encontramos al ejecutar la aplicación es un pequeño cuadro de *login*, para poder iniciar sesión en Muschain o crear una cuenta nueva en caso de no haberse registrado todavía.

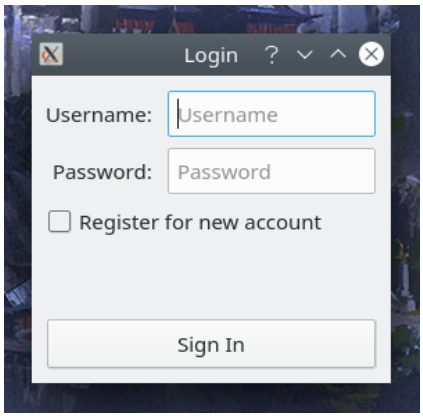

Figura 52: Login Muschain 2do MVP

Tras iniciar sesión o crear una nueva cuenta marcando la casilla justo debajo del campo de contraseña, aparecerá la ventana principal del editor. Al mismo tiempo se abre una ventana de bienvenida que nos permitirá realizar tres acciones:

- Configurar y comenzar una nueva canción
- Continuar una canción existente creada por otro usuario, asignada de forma aleatoria
- Escuchar las canciones completadas

En el caso de cerrar la ventana de bienvenida, el editor mantendrá el estado inicial, configurado para crear una nueva canción básica con una sola pista de 16 *steps* y el instrumento por defecto *Pulse*, igual que durante la primera versión de Muschain.

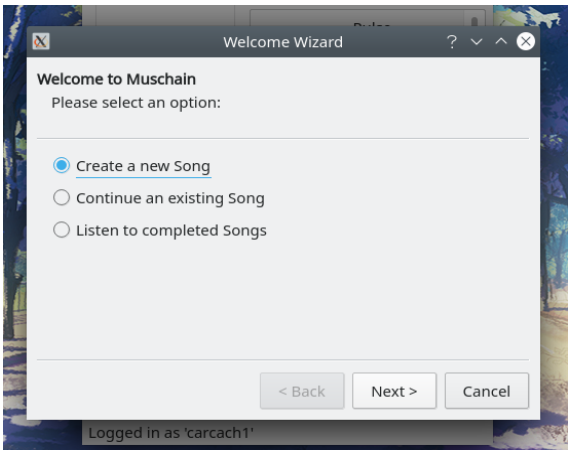

Figura 53: Ventana bienvenida Muschain 2do MVP

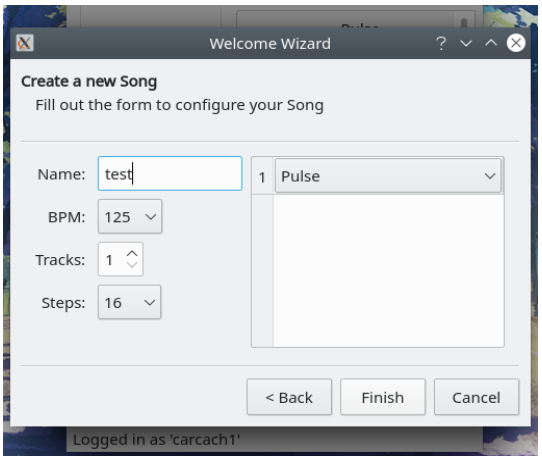

Figura 54: Crear nueva canción Muschain 2do MVP

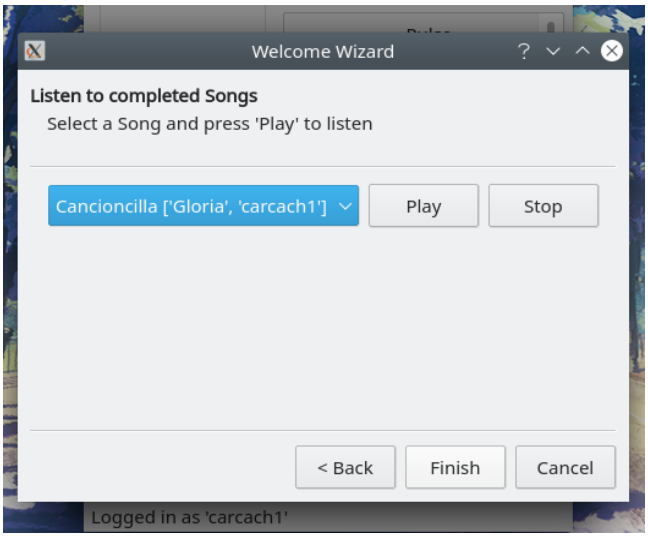

Figura 55: Ventana canciones completadas Muschain 2do MVP

El principal cambio en el editor lo vemos cuando creamos una canción con hasta cuatro instrumentos, o seleccionamos la opción de “Continuar una canción existente”. La ventana principal se actualizará para crear las nuevas pistas con los instrumentos seleccionados, y en el caso de continuar una canción se habilitará el componente de previsualización de la sección anterior. En este componente podremos ver las notas que componen el fragmento musical anterior, así como reproducirlo, para que nos sirva de guía y como punto de partida para poder continuar la canción. En este modo quedará deshabilitado el regulador de BPM para impedir al usuario que continue la canción cambiar la velocidad de reproducción.

Para evitar que las canciones se quedaran en el limbo si una persona quería continuar una canción, pero nunca marcaba su sección como ‘completada’, sólo se permitía realizar una acción de guardado. Al darle a ‘Guardar’, se avisaba al usuario de que esta acción no puede deshacerse, y la sección quedará bloqueada para que otro usuario pueda seguir con la canción.

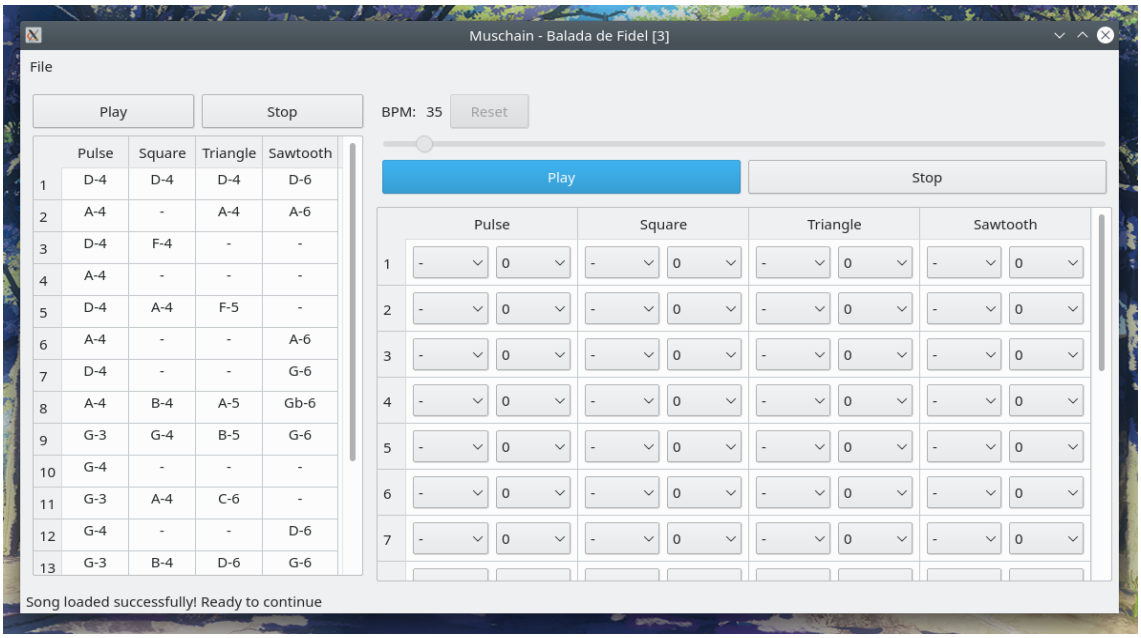

Figura 56: Versión mejorada editor Muschain 2do MVP

5. Aspectos técnicos

En esta sección se detallarán las herramientas utilizadas para el desarrollo de Muschain, así como su aplicación en el proyecto y algunos inconvenientes técnicos que surgieron durante el desarrollo. Comentar también que el orden en que se presentan las herramientas no lleva asociado ningún significado.

5.1. Herramientas utilizadas

5.1.1. Python

Python [71] es un lenguaje de programación interpretado, creado por Guido van Rossum en el año 1991. Su objetivo es ser un lenguaje intuitivo, de código abierto e igual de potente que cualquiera de sus competidores, tal y como recoge la propuesta “Programación para todos” [72] publicada en agosto de 1999.

Muschain se ha desarrollado con este lenguaje de programación, utilizando un estilo orientado a objetos. Con el lanzamiento de Python 3, toda clase que se declare extiende el tipo *object* de Python, por lo que todo es considerado como un objeto completamente mutable [73]

5.1.2. QT for Python

QT [74] es un *framework* gráfico para diseño de interfaces de usuario escrito en C++ y publicado originalmente en 1995 por la empresa Trolltech con licencia de código abierto. El framework es multiplataforma y funciona en varios sistemas operativos como Windows, Mac y Linux, entre otros. Es posible utilizarlo con otros lenguajes de programación a través de distintos *bindings*, como es el caso de QT for Python.

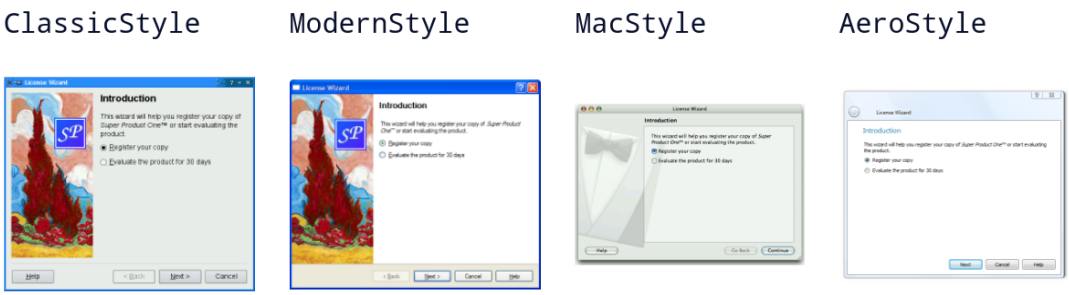

Figura 57: Adaptación de QT al estilo nativo del SO

Se eligió este *framework* para Muschain por su integración con el estilo nativo del sistema operativo anfitrión, entre otros motivos mencionados durante la motivación en la sección de Introducción.

5.1.3. Git

Git [75] es un sistema de control de versiones de código abierto creado por Linus Torvalds en el año 2005. En la actualidad es usado por infinidad de desarrolladores y empresas como herramienta de gestión colaborativa, que permite a varios programadores trabajar en la misma pieza de código cómodamente. También sirve para mantener un registro de la evolución del proyecto, así como separar la implementación de funcionalidades en distintas ramas o *branches*, de forma en que afecte lo menos posible al trabajo de otros desarrolladores que estén implementando otras partes del proyecto. Cada modificación de este registro recibe el nombre de *commit*, que actúa a modo de instantánea del proyecto. Si se realiza algún cambio no deseado en el proyecto y desea deshacerse, basta con recuperar un *commit* anterior donde este cambio no haya sido aplicado todavía. La interfaz de control de Git es la consola de comandos, pero se han desarrollado muchas herramientas con interfaz gráfica para facilitar la tarea a desarrolladores que no estén habituados a trabajar con la consola, como es el caso de Gitkraken [76].

En Muschain se ha hecho uso de Git para gestionar todo el progreso del proyecto, aprovechando las funcionalidades de instantáneas para poder revertir cambios en caso de que fuera necesario.

5.1.4. GitLab

GitLab [77] es un servicio web basado en Git que ofrece varios utilidades de interés para los desarrolladores, como un sistema de control de repositorios, seguimiento de errores o incidencias (*issues*), alojamiento para *wikis* o documentación para el proyecto, herramientas de CI/CD [78], entre otras. GitLab está licenciado como software *open source*, y es posible crear una instancia propia para tener una experiencia más personalizada. La instancia principal puede usarse de forma totalmente gratuita, pero algunas opciones están reservadas para los usuarios que adquieran un plan de subscripción. Su principal competidor y referente en el mundo del desarrollo es GitHub [79], el cual también es de uso gratuito para proyectos *open source* usando un repositorio público. [80]

Para el proyecto de Muschain se ha optado por usar GitLab por la posibilidad de almacenarlo como un repositorio privado de forma totalmente gratuita, y así también poder acceder al proyecto desde cualquier ordenador para continuar su desarrollo indistintamente del equipo desde el cual se trabaje. También se ha hecho uso del sistema de *issues* y las estimaciones de tiempo, para establecer las *milestones* y organizar el proyecto.

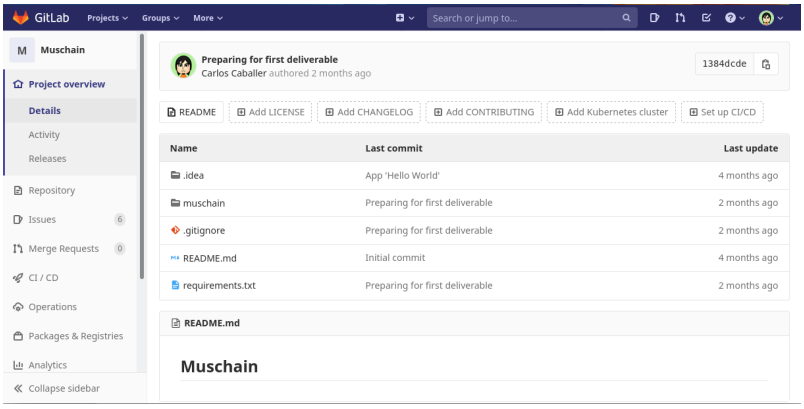

Figura 58: Vista principal de proyecto en GitLab

5.1.5. PyCharm

PyCharm [81] es un IDE o Entorno de Desarrollo Integrado para Python publicado por la empresa JetBrains en el año 2010. Dispone de una versión gratuita y *open source* apodada como Community Edition, la cual nos permite su uso para proyectos no comerciales o con licencia *open source*. Incluye herramientas muy útiles como análisis estático del código, detección de errores de sintaxis, *highlighting* para mejorar la legibilidad del código, herrmientas de *refactoring*, *debugger* y una suite para hacer *testing*, entre otras. También es multiplataforma y funciona tanto en Windows como en Mac o Linux.

Para el desarrollo de Muschain se ha optado por usar esta herramienta por las facilidades que ofrece a la hora de depurar el código, así como las de *refactoring*, muy útiles si se quiere reestructurar el proyecto o renombrar alguna clase o variable.

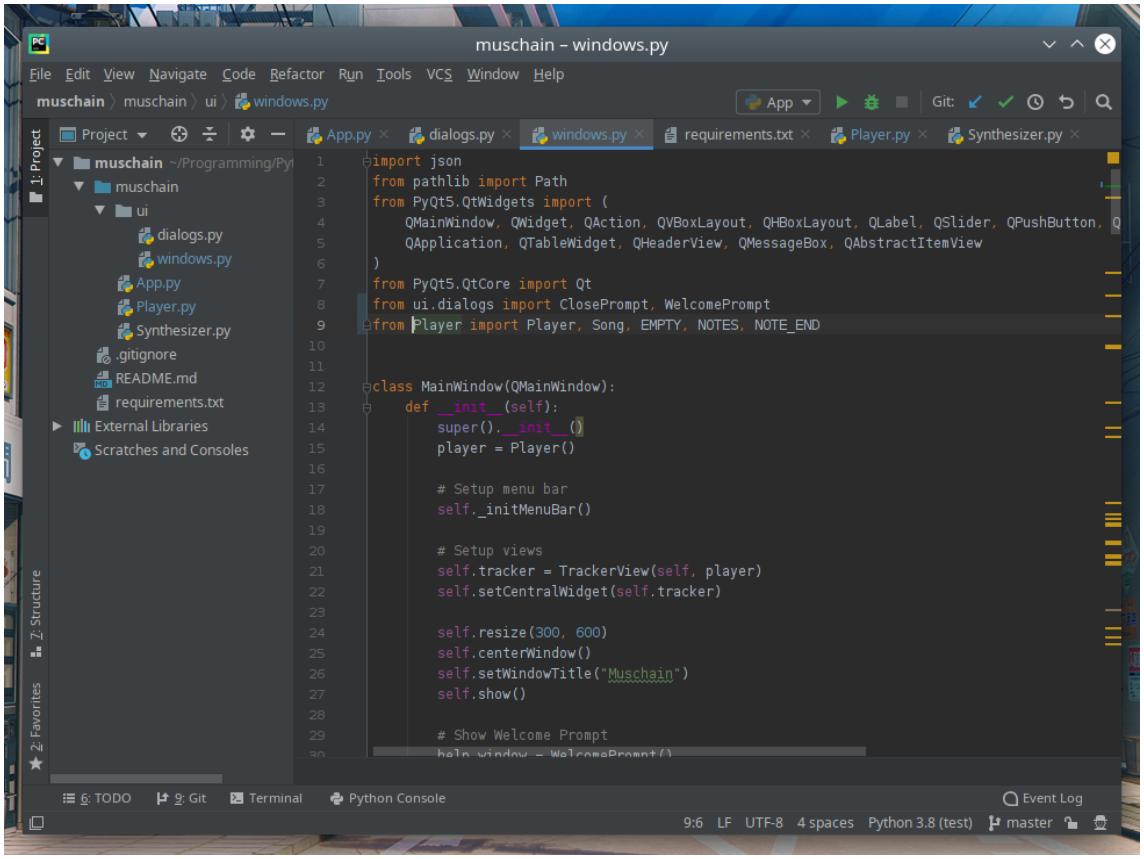

Figura 59: Vista principal de proyecto en PyCharm CE

5.1.6. Back4App

Back4App [19] es un servicio de *backend* con licencia *open source* basado en el framework Parse [21], el cual busca ofrecer a los desarrolladores la posibilidad de configurar una API para sus aplicaciones sin necesidad de programar. Tiene un plan gratuito que permite configurar una API por cuenta, realizar un máximo de 10.000 peticiones al mes, y almacenar un total de 1GB de datos. Los planes de subscripción se separan entre *hosting* compartido y dedicado, con un precio de partida de $5/mes, hasta los $1000/mes para el plan más avanzado. [82]

Se decidió usar Back4App en Muschain para simplificar la parte del servidor, entre otros.

5.2. Arquitectura utilizada

La plataforma de Muschain cuenta actualmente con un diseño relativamente sencillo. Se compone principalmente de un editor, un reproductor, una capa de negocio con la base de datos, y los distintos elementos que conforman lo que sería el concepto de “canción”. A continuación mostramos un diagrama representando su estructura:

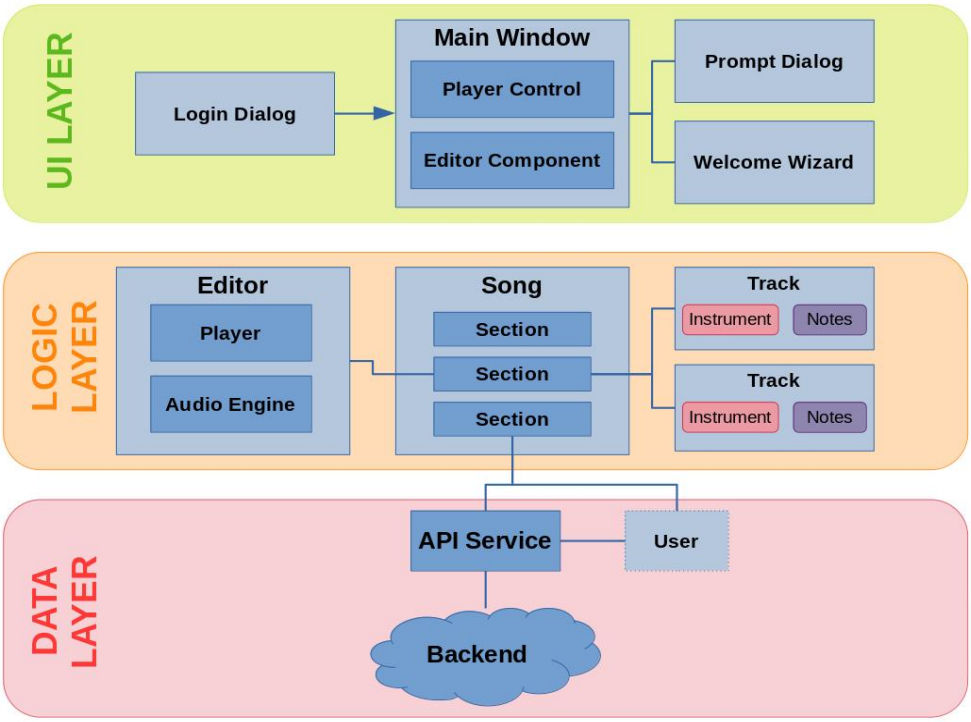

Figura 60: Estructura actual de Muschain

Como podemos apreciar en el diagrama, la mayor parte del peso de Muschain se concentra en la lógica relacionada con el editor, las canciones con sus correspondientes secciones/pistas y la parte de la interfaz gráfica. De la gestión de datos se encarga íntegramente el *backend*, junto con el servicio que se comunica entre el servidor y la aplicación. En la actualidad no existe una entidad de “usuario” dentro de la aplicación, debido a que únicamente se utiliza para indicar el autor de una sección musical. Por ello se relaciona la sección con el ID del usuario de Back4App que ha iniciado sesión utilizando el *endpoint* de *login* proporcionado por el API Service. Cuando en un futuro se implementen los perfiles de usuario, las publicaciones y los mensajes privados, se creará una entidad User en la aplicación para poder administrar todas las relaciones de forma más sencilla.

A continuación se detallarán algunos inconvenientes que surgieron durante el proceso de desarrollo, las distintas alternativas para solventarlos y las medidas tomadas, junto con una explicación de por qué se adoptaron estas medidas:

5.3. Problemas y desafíos técnicos

Durante el planteamiento del proyecto surgieron algunos problemas que pusieron en peligro su desarrollo, debido a algunos imprevistos. También aparecieron algunos problemas que dificultaron su desarrollo, suponiendo una carga de trabajo mayor de la esperada. Todo ello se detalla a continuación, empezando por el principal problema:

5.3.1. El problema del editor

La idea original cuando comenzó a desarrollarse lo que sería Muschain era utilizar una librería sencilla que recibiese como *input* un array con las notas musicales en nomenclatura americana (A, B, C, …) y las reproduciera acorde a un *tempo* establecido. El sonido se encargaría de generarlo un sintetizador que utilizaría el instrumento que estuviese asignado a la pista en concreto. Un ejemplo de este tipo de librería sería JFugue [83], basada en el lenguaje de programación Java. De esta forma, se hubiese podido dedicar más tiempo a lo que sería el desarrollo de la plataforma como red social, centrando los esfuerzos en la mecánica de juego, las publicaciones y la mensajería dentro de la aplicación. Sin embargo, tras algún tiempo de búsqueda e investigación, no se llegó a dar con una librería similar para usarla con Python. [84] La mayoría de las librerías existentes estaban destinadas a trabajar con ficheros MIDI para reproducirlos con un sintetizador externo, a través de un puerto MIDI real o virtual, o estaban más orientadas al trabajo con audio.

La opción que más se acercaba a lo que se buscaba era un proyecto llamado Synthesizer [85], el cual es un pequeño sintetizador escrito en Python capaz de reproducir una nota o acorde durante un tiempo determinado, dada una frecuencia concreta (o lista de frecuencias, en el caso del acorde). Con esto se solucionaba la parte de la generación del sonido, pero quedaba toda la parte de la reproducción de varias notas seguidas en función del *tempo*, la tarea básica que realiza cualquier secuenciador. Se hicieron varias pruebas para intentar concatenar el audio de salida de Synthesizer, pero no tuvo éxito por problemas con la librería de PyAudio [86] que utiliza el proyecto internamente.

Esto cambió por completo el foco de trabajo de lo que se presentaría como trabajo realizado durante el desarrollo del TFG, que terminó centrándose en el desarrollo del secuenciador. Éste sería a su vez la base sobre la que se desarrollaría la plataforma de Muschain, pero debido a las limitaciones de tiempo, toda la parte social de la plataforma con excepción de la mecánica de encadenamiento quedó fuera del alcance del proyecto de TFG. Tampoco el desarrollo del secuenciador quedó libre de incidencias y algunos retos que se detallarán más adelante.

Para el módulo de sintetizador se usó como base el proyecto de Synthesizer, y se reemplazó el motor de audio por la librería SimpleAudio [87], la cual tenía una documentación más clara y un uso mucho más simple. Se decidió separar la librería de audio de la entidad de “sintetizador”, para de esta forma poder crear varias instancias, cada una que generara una forma de onda distinta, todas ellas utilizando la misma referencia a la librería de audio, que quedaría alojada en el módulo de “reproductor”. A continuación se muestra el módulo resultante del sintetizador:

```python
import numpy as np
from scipy import signal


class Synthesizer(object):
    class WaveForm(Enum):
        PULSE = 'Pulse'
        SQUARE = 'Square'
        TRIANGLE = 'Triangle'
        SAWTOOTH = 'Sawtooth'

    def __init__(self, waveform):
        self.waveform = waveform.value

    def generateWave(self, frequency, rate, duration):
        size = rate*duration
        if frequency:
            phase = 2 * np.pi * np.arange(size)*frequency/rate
            if self.waveform is Synthesizer.WaveForm.PULSE.value:
                return np.sin(phase)
            elif self.waveform is Synthesizer.WaveForm.SQUARE.value:
                return signal.square(phase)*0.5  # Volume too loud
            elif self.waveform is Synthesizer.WaveForm.TRIANGLE.value:
                return signal.sawtooth(phase, width=0.5)
            elif self.waveform is Synthesizer.WaveForm.SAWTOOTH.value:
                return signal.sawtooth(phase)*0.75  # Volume too loud
            else:
                raise TypeError("Unknown waveform: {}".format(self.waveform))
        else:
            return np.zeros(int(size))
```

Figura 61: Módulo sintetizador de Muschain

5.3.2. La conversión entre notas y frecuencias

Para conseguir un flujo de trabajo similar al de JFugue, era necesario convertir las notas en formato “A-3” (nombre de la nota y octava) en sus correspondientes frecuencias, para que luego el sintetizador se encargara de generar el sonido. Utilizando una fórmula para averiguar la frecuencia de una tecla de piano determinada [88] y una tabla de conversión entre “nota-octava” y “tecla de piano”, se consigue traducir la nomenclatura utilizada en el editor en frecuencias. A continuación se muestra un fragmento de código con las funciones para realizar la conversión:

```python
NOTES = ['A', 'Bb', 'B', 'C', 'Db', 'D', 'Eb', 'E', 'F', 'Gb', 'G', 'Ab']
EMPTY, NOTE_END = "-", "OFF"


def _getKeyNumber(self, note, octave):
    base_key = NOTES.index(note) + 1  # Add 1 since Array starts at 0
    if base_key > 3:  # Octave '1' starts with C
        return base_key + 12 * (octave - 1)
    else:
        return base_key + 12 * octave


def _getNoteFrequency(self, input_note):
    if input_note in (EMPTY, NOTE_END):
        return 0
    else:
        note, octave = input_note.split("-")
        key = self._getKeyNumber(note, int(octave))
        freq = pow(2, float(key - 49) / 12) * 440  # A4 is Key 49, octave has 12 keys
        return freq
```

Figura 62: Funciones de conversión nota --> frecuencia en Muschain

5.3.3. La polifonía y los desfases entre pistas según el BPM

El *tracker* calcula la duración "real" que deben tener las notas en función del BPM, el TPB y la frecuencia de muestreo. Sin embargo, debido a la relación existente entre estos parámetros es posible que la duración de la nota termine siendo un valor decimal. Esto crea un inconveniente, y es que de la forma en que trabaja el sintetizador se genera un array con los valores que representan la onda de sonido, para que luego el motor de audio se encargue de convertirlo en sonido real. No obstante, el tamaño de este array no puede ser decimal, por lo que su comportamiento por defecto es redondearlo a la baja para obtener una longitud entera. Esto no supone un problema cuando sólo se quiere trabajar con una pista/instrumento, pero genera inconsistencias cuando se aplica con varios instrumentos, como vemos en el siguiente fragmento de código que representa la generación de una onda polifónica:

```
# class Section ...
def toAudio(self, rate, tick):
    output = self.tracks[0].toAudio(rate, tick)
    for track in self.tracks[1:]:
        output += track.toAudio(rate, tick)
    output /= len(self.tracks)
    return output


# class Player ...
def _getBpmList(self):
    bpm_list = []
    # Find BPM from 30 to 300 within 5 distance
    for i in range(30, 301, 5):
        # Check if 'tick' got decimal
        if not (60 * self._rate) % (i * self._tpb):
            bpm_list.append(i)
    return bpm_list
```

Figura 63: Funciones de polifonía y cálculo BPM en Muschain

La función *toAudio* dentro de la clase *Section* se encarga de generar los arrays con la forma de la onda para cada una de las pistas, para luego "fusionarlos" y crear una onda polifónica. Para que esto funcione, todos los arrays deben tener la misma longitud, pero esto no ocurre cuando es posible que la suma de la duración de cada una de las notas tenga un valor decimal.

El valor de BPM "por defecto" siempre ha sido 120, esto es dos golpes por segundo. Pero por motivos históricos relacionados con el origen de los *trackers* y su dependencia con la tasa de refresco de los monitores de los PCs de la época [89], su valor estándar para los *trackers* es de 125. Teniendo esto en cuenta, se optó por limitar los posibles valores para el BPM en aquellos que garantizaran que la longitud del array resultante iba a ser siempre un entero. Siguiendo la siguiente fórmula matemática podemos calcular la longitud del array:

$$length = rows \times rate \times \left(\frac{60}{BPM \times TPB}\right)$$

siendo *rows* la longitud en notas de la pista y *rate* la frecuencia de muestreo (usando el valor por defecto de 44,1kHz que se encuentra en los CDs [90]). Para que este valor dé un número entero, debemos asegurar que la división dentro del paréntesis sea entera también. Por ello, se optó por restringir los valores posibles para el BPM, ya que el TPB es también una constante.

6. Conclusiones

Tras los dos experimentos realizados con un grupo de *early adopters*, podemos concluir que el resultado del proyecto ha sido satisfactorio. Pese a no haber podido tratar el aspecto social de Muschain con mayor profundidad por limitaciones de tiempo, gran parte de los usuarios encuestados han mostrado interés por el proyecto y estarían dispuestos a seguir utilizándolo si se corrigen ciertos aspectos como los controles del editor, las limitaciones que tiene actualmente en cuanto a las operaciones que se pueden realizar, configuración de los instrumentos, regulación del volumen, …

No obstante, debido a las estrictas medidas de seguridad de las grandes compañías tecnológicas, la distribución del programa para el segundo experimento se complicó más de la cuenta, obligando a los participantes en el experimento a seguir unas determinadas instrucciones para poder llegar a ejecutar el programa y de esta forma probarlo. Este es un tema bastante problemático, no puede publicarse un producto bajo estas condiciones. Por ello sería necesario investigar más este aspecto para obtener un certificado de confianza que permita ejecutar el programa sin problemas en Windows, así como publicarlo mediante una plataforma de distribución adecuada, en lugar de un enlace compartido a un fichero en Drive.

Como experiencia personal, ha sido un reto bastante interesante investigar cómo funcionan internamente los *trackers*, comprender su estructura y reimplementar una versión simplificada. Debido a que el funcionamiento de un *tracker* o secuenciador de musica real es mucho más complejo, implementar todas las características que cabe esperar de este tipo de programas hubiese sido una tarea titánica, quedando completamente fuera del alcance del TFG. Esto me ha servido para valorar el enorme esfuerzo y cantidad de recursos que se necesitan para desarrollar este tipo de software.

6.1. Trabajo futuro

El proyecto de Muschain se encuentra actualmente tal y como se presentó a los *early adopters* durante el segundo experimento. Puede verse el programa en acción en el siguiente vídeo de demostración[5] *(El vídeo no está narrado, no subir mucho el volumen)*.Todavía quedan muchas características por implementar según el plan de desarrollo, y también es necesario corregir y replantear algunos puntos de la interfaz gráfica. Por ello, el primer trabajo a realizar será rediseñar la interacción con el editor, para conseguir que los usuarios se sientan más cómodos trabajando con él. Luego se seguirán añadiendo más características y se probarán nuevas ideas que enriquezcan la mecánica sobre la que se basa el programa para fortalecer la parte lúdica.

Tras varias iteraciones de desarrollo y testeo, se estudiará publicar el software en una plataforma de distribución con un gran alcance de usuarios en forma de *beta* gratuita con anuncios, para que vaya dándose a conocer. Cuando terminen de implementarse y pulirse todas las características principales del programa, arrancará completamente con su modelo *freemium* y las microtransacciones en la tienda dentro de la aplicación.

5 Demo Muschain: https://drive.google.com/file/d/16ih3aLgQfCT07WHZxoullMfjlpjZ8lZs/view?usp=sharing

7. Referencias bibliográficas

[1] Página principal de Facebook: https://facebook.com/

[2] Lista de redes sociales más usadas en 2020: https://www.dreamgrow.com/top-15-most-popular-social-networking-sites/

[3] Página principal de Twitter: https://twitter.com/

[4] Página oficial de Instagram: https://instagram.com/

[5] Página principal de YouTube: https://youtube.com/

[6] Página principal de SoundCloud: https://soundcloud.com/

[7] Página oficial de Apalabrados: https://apalabrados.com/

[8] Página oficial de Scrabble: https://scrabble.hasbro.com/

[9] Página oficial de Pinturillo: https://www.pinturillo2.com/

[10] Página oficial de Pictionary: https://www.mattelgames.com/en/family/pictionary

[11] Página principal de Google Docs: https://docs.google.com/

[12] Página principal de Graphite Docs: https://www.graphitedocs.com/

[13] Página oficial de VLC: https://www.videolan.org/vlc/

[14] Información sobre MTP: https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_medios

[15] Página oficial de ActivityPub: https://activitypub.rocks/

[16] Información sobre el Fediverse: https://es.wikipedia.org/wiki/Fediverso

[17] Página oficial de Unity3D: https://unity.com/es

[18] Página oficial de QT: https://www.qt.io/

[19] Página principal de Back4App: https://www.back4app.com/

[20] Página principal de Firebase: https://firebase.google.com/

[21] Página oficial de Parse: https://parseplatform.org/

[22] Página oficial de KDE Plasma: https://kde.org/plasma-desktop

[23] Virtual Private Server: https://es.wikipedia.org/wiki/Servidor_virtual_privado

[24] Resultados de la encuesta 2019 de StackOverflow: https://insights.stackoverflow.com/survey/2019#most-loved-dreaded-and-wanted

[25] Historia y detalles sobre SoundCloud: https://en.wikipedia.org/wiki/SoundCloud

[26] Página oficial de Spotify: https://www.spotify.com/es/

[27] Algunos vídeos interactivos de YouTube (inglés): https://mashable.com/2011/01/30/interactive-youtube-videos/?europe=true

[28] Comparativa de varios códecs de audio: https://en.wikipedia.org/wiki/Comparison_of_audio_coding_formats

[29] Página oficial de Creative Commons: https://creativecommons.org/licenses/?lang=es

[30] Información sobre RSS: https://es.wikipedia.org/wiki/RSS

[31] Comparativa planes SoundCloud: https://checkout.soundcloud.com/pro

[32] Información sobre trackers de música: https://es.wikipedia.org/wiki/Tracker_(software_de_musica)

[33] Página oficial de Renoise: https://www.renoise.com/

[34] Información sobre las tarjetas SoundBlaster: https://es.wikipedia.org/wiki/Sound_Blaster

[35] Información sobre el formato SoundFont: https://es.wikipedia.org/wiki/SoundFont

[36] Información acerca del protocolo MIDI: https://es.wikipedia.org/wiki/MIDI

[37] Información sobre el software DAW: https://es.wikipedia.org/wiki/Estaci%C3%B3n_de_trabajo_de_audio_digital

[38] Información sobre drivers ASIO: https://es.wikipedia.org/wiki/Audio_Stream_Input/Output

[39] Información sobre los drivers CoreAudio: https://es.wikipedia.org/wiki/Core_Audio

[40] Información sobre el servidor JACK: https://es.wikipedia.org/wiki/JACK_Audio_Connection_Kit

[41] Información sobre el estándar VST: https://es.wikipedia.org/wiki/Virtual_Studio_Technology

[42] Información sobre la arquitectura AU: https://es.wikipedia.org/wiki/Audio_Units

[43] FAQ de Renoise: https://www.renoise.com/registration_faq

[44] Información sobre el juego Érase una vez: http://www.edgeent.com/juegos/coleccion/erase_una_vez

[45] Página oficial de Atlas Games: https://atlas-games.com/

[46] Página oficial de EDGE Entertainment: http://www.edgeent.com

[47] Información acerca de la Web 2.0: https://es.wikipedia.org/wiki/Web_2.0

[48] Información sobre productos SaaS: https://es.wikipedia.org/wiki/Software_como_servicio

[49] Página web oficial de Office 365: https://www.microsoft.com/es-es/microsoft-365

[50] Página web oficial de LibreOffice: https://es.libreoffice.org/

[51] Información sobre web apps: https://es.wikipedia.org/wiki/Aplicacion_web

[52] Página oficial de SoundTrap: https://www.soundtrap.com/

[53] Historia de SoundTrap: https://www.soundtrap.com/about

[54] Spotify adquiere SoundTrap: https://blog.soundtrap.com/soundtrap-acquired-by-spotify/

[55] Planes de precio de SoundTrap: https://www.soundtrap.com/pricing

[56] Página web oficial de AudioTool: https://www.audiotool.com/

[57] Página web oficial de Ohm Studio : https://www.ohmstudio.com/home

[58] Página web oficial de Soundation: https://soundation.com/

[59] Página web oficial de Discord: https://discord.com/

[60] Página web oficial de Skype: https://www.skype.com/es/

[61] Tendencia de las microtransacciones en PC: https://www.pcgamer.com/revenue-from-pc-free-to-play-microtransactions-has-doubled-since-2012/

[62] Información acerca de las cajas de botín: https://en.wikipedia.org/wiki/Loot_box

[63] Estudio sobre microtransacciones en 2019: https://www.playstationlifestyle.net/2019/11/08/microtransaction-spending-in-september/

[64] Estudio naturaleza de microtransacciones por Qutee en 2018: https://s3.amazonaws.com/qutee-reports/Qutee-Gaming-Today-Report.pdf

[65] Condiciones del servicio de la Microsoft Store: https://query.prod.cms.rt.microsoft.com/cms/api/am/binary/RE4o4bF

[66] Detalles sobre el Apple Developer Program: https://developer.apple.com/programs/whats-included/

[67] Página web oficial de Animal Crossing New Horizons: https://animal-crossing.com/new-horizons/

[68] Página web oficial de FL Studio: https://www.image-line.com/flstudio/

[69] Página web oficial de GarageBand (iOS): https://www.apple.com/es/ios/garageband/

[70] Página web oficial de Reaper: https://www.reaper.fm/

[71] Información acerca de Python: https://es.wikipedia.org/wiki/Python

[72] Propuesta a DARPA sobre Python: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.123.6836&rep=rep1&type=pdf

[73] Clases new-style y clases old-style en Python: https://riptutorial.com/python/example/5124/all-classes-are--new-style-classes--in-python-3-

[74] Información sobre el framework QT: https://es.wikipedia.org/wiki/Qt_(biblioteca)

[75] Información acerca de Git: https://es.wikipedia.org/wiki/Git

[76] Página web oficial de Gitkraken: https://www.gitkraken.com/

[77] Página principal de GitLab: https://gitlab.com/

[78] Información sobre técnicas CI/CD: https://en.wikipedia.org/wiki/CI/CD

[79] Página web oficial de GitHub: https://github.com/

[80] Comparativa entre GitHub y GitLab: https://usersnap.com/blog/gitlab-github/

[81] Página web oficial de PyCharm: https://www.jetbrains.com/pycharm/

[82] Comparativa planes Back4App: https://www.back4app.com/compare-all-plans

[83] Página web oficial de JFugue: http://www.jfugue.org/

[84] Lista con algunas librerías de audio para Python: https://wiki.python.org/moin/PythonInMusic

[85] Repositorio del proyecto Synthesizer:
https://github.com/yuma-m/synthesizer

[86] Página web oficial de PyAudio:
http://people.csail.mit.edu/hubert/pyaudio/

[87] Repositorio del proyecto de SimpleAudio:
https://github.com/hamiltron/py-simple-audio

[88] Fórmula para calcular la frecuencia de una tecla de piano:
https://en.wikipedia.org/wiki/Piano_key_frequencies

[89] Relación entre el BPM y TPB en un tracker:
https://modarchive.org/forums/index.php?topic=2709.0

[90] Frecuencia de muestreo de los CDs:
https://en.wikipedia.org/wiki/Compact_Disc_Digital_Audio#Audio_format

Glosario

- BPM / Tempo: *(Beats Per Minute)* Unidad de medida utilizada en la música para indicar la velocidad con la que se deben reproducir las notas de una canción.
- Chiptune: Estilo de música caracterizado por el uso de sonidos generados con chips de audio, muy populares durante las décadas de los 80 y los 90.
- Frecuencia de muestreo: Número de muestras por segundo que se toman de una forma de onda analógica para conseguir una representación digital de la misma.
- Gain: Efecto de sonido que permite aumentar el volumen de una muestra de audio.
- MIDI: Protocolo utilizado por los instrumentos musicales electrónicos para plasmar y comunicar los eventos musicales que componen una canción.
- Octava: En teoría musical, la octava representa la distancia que separa dos notas que "aparentemente suenan iguales", sólo cambiando que es más aguda o grave.
- Pan: Parámetro utilizado para indicar la 'ubicación' de la fuente de sonido en un sistema estéreo o multicanal.
- Pitch: Parámetro que mide el desvío de la frecuencia de referencia para una nota dada. Su alteración supone una distorsión de la nota, hasta llegar a la frecuencia de una nota colindante.
- Polifonía: Capacidad de un instrumento para reproducir más de un sonido de forma simultánea.
- Sample: En música, el término *sample* hace referencia a la grabación de corta duración de un instrumento o sonido, para luego utilizarlo como base para producir las notas o efecto de sonido deseado.
- Secuenciador: Dispositivo físico o software que permite crear y reproducir música. Puede tratarse de un dispositivo analógico, como el rodillo con protuberancias de una caja de música, o de un sistema electrónico como el que se utiliza para componer con instrumentos de música digitales.
- Timbre: Característica del sonido que emite un instrumento, la cual permite distinguirlo de otros incluso reproduciendo las mismas notas.
- Time stretching: Técnica utilizada para alargar o recortar la duración de una muestra de audio sin alterar su *pitch*.
- TPB: *(Ticks Per Beat)* En un *tracker*, número de subdivisiones realizadas a la figura musical *negra* para poder representar notas con una longitud inferior a ésta.
- Tracker: Una variante del secuenciador (software) que se caracteriza por su representación de la línea temporal de la canción como una tabla vertical, donde el cursor avanza hacia abajo reproduciendo las notas a su paso.