UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

MusicApp: software de gestión para escuelas de música

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Diéguez Bens, Javier

Tutor/a: Letelier Torres, Patricio Orlando

CURSO ACADÉMICO: 2023/2024

A mis padres por darme apoyo y hacer posible que pueda estudiar lo que me gusta

Agradecimientos

Me gustaría agradecer a todos los profesores que he tenido a lo largo de mi vida y que me han permitido los conocimientos que tengo ahora.

Gracias a mi familia y amigos por estar ahí siempre por su comprensión y apoyo.

Y gracias al Centre Estudi Musical, Vicent Rioja y a todos los participantes de los experimentos por su contribución a este proyecto.

Resumen

En este Trabajo de Fin de Grado se aborda el desarrollo de un software de gestión dirigido a escuelas de música, con el objetivo de optimizar sus procesos administrativos y mejorar la calidad de la enseñanza musical. La transformación digital está revolucionando todos los sectores, y la educación no es la excepción. Este proyecto surge de la necesidad de dotar a las escuelas de música de herramientas modernas diseñadas específicamente para las peculiaridades de estos centros educativos, que faciliten la labor de los docentes y permitan una gestión más eficiente y precisa. Este proyecto se enmarca en Star.tinf, el espacio de emprendimiento de la ETSINF y se tratará como un proyecto de emprendimiento. Se busca involucrar a docentes y trabajadores del Centre Estudi Musical de Almàssera (CEM) durante el desarrollo del proyecto, asegurando que el producto final responda a las necesidades reales de los usuarios y mejore significativamente la experiencia educativa y administrativa en las escuelas de música.

Palabras clave: desarrollo de aplicación móvil, desarrollo de aplicación web, escuelas de música, software empresarial, software de gestión.

Abstract

This Final Degree Project deals with the development of a management software aimed at music schools, with the aim of optimizing their administrative processes and improving the quality of music teaching. Digital transformation is revolutionizing all sectors, and education is no exception. This project arises from the need to provide music schools with modern tools designed specifically for the peculiarities of these educational centers, which facilitate the work of teachers and enable more efficient and accurate management. This project is part of Star.tinf, the ETSINF's entrepreneurship space, and will be treated as an entrepreneurship project. The aim is to involve teachers and employees of the Centre Estudi Musical de Almàssera (CEM) during the development of the project, ensuring that the final product responds to the real needs of users and significantly improves the educational and administrative experience in music schools.

Keywords: mobile application development, web application development, music schools, business software, management software.

Índice de contenidos

Índice de figuras

Índice de tablas

CAPÍTULO 1

Introducción

1.1 Motivación

En un mundo cada vez más tecnológico, la transformación digital es un hecho que está revolucionando todos los sectores, desde la industria hasta los servicios, pasando por la educación. Cada vez más organizaciones y empresas se adaptan a esta nueva realidad para mantenerse relevantes y competitivas en un panorama tecnológico en constante cambio.

La adopción de estas herramientas en el ámbito de las instituciones educativas, especialmente para las escuelas de enseñanza musical, no es solo una tendencia, sino una necesidad para mejorar la eficiencia, la gestión y la calidad de la enseñanza. La tecnología puede optimizar los procesos administrativos, facilitando la labor de los docentes, liberándoles de tareas rutinarias y permitiéndoles concentrarse en lo que realmente importa, la educación y el desarrollo de sus estudiantes.

Desde mi propia experiencia como músico, puedo constatar que las escuelas de música no solo son espacios donde se adquieren habilidades técnicas y teóricas musicales, sino que también son lugares donde se cultivan talentos y se estimula la creatividad. Estos centros ofrecen un entorno enriquecedor para los estudiantes, permitiéndoles desarrollar su expresión artística, mejorar sus habilidades sociales y construir una base sólida para su crecimiento personal. La educación musical fomenta valores fundamentales como la disciplina, la perseverancia y el trabajo en equipo, aspectos cruciales para el desarrollo personal y académico de los estudiantes.

El sistema de enseñanza musical en España es amplio y diverso, en la ley Orgánica 2/2006, de 3 de mayo, de educación [1] se recoge que las enseñanzas artísticas están divididas en 3 niveles. Las enseñanzas elementales, que engloban los conocimientos fundamentales sobre teoría musical y práctica instrumental. Las enseñanzas profesionales corresponden a un nivel medio y a conocimientos más avanzados. Por último, las enseñanzas superiores pueden considerarse equivalentes a un grado universitario. La duración de cada nivel es de 4 a 6 años aproximadamente.

Los centros de enseñanza musical son considerados centros de enseñanza de régimen especial, vamos a distinguir entre ellos a los conservatorios, donde se pueden impartir los tres niveles de enseñanza nombrados, los conservatorios públicos están reglados por el Ministerio de Educación y Cultura. Otro centro que vamos a considerar son las escuelas municipales, gestionadas por los ayuntamientos de dicho municipio. En ellas solo se pueden impartir las enseñanzas elementales.

Según datos de la plataforma Educateca, que podemos observar en la figura 1, en España existen más de 2.000 centros de enseñanza de música, con una concentración significativa en la Comunidad Valenciana otras fuentes como la FSMCV[1] o Federación de sociedades musicales de la comunidad valenciana, quienes afirman en su web [2] que cuentan con alrededor de 600 centros miembros. Muchas de estas escuelas aún dependen de métodos tradicionales para gestionar sus actividades diarias, lo que puede resultar ineficiente y propenso a errores. La falta de herramientas de gestión modernas y específicas limita su capacidad para crecer y ofrecer un mejor servicio y una experiencia educativa más enriquecedora a sus estudiantes.

CENTROS POR COMUNIDAD AUTÓNOMA A 03-06-2024

Comunidad Autónoma	Públicos	Privados	Total Centros
TOTAL CENTROS	1.200	945	2.145
Andalucía	185	39	224
Aragón	45	22	67
Cantabria	7	4	11
Castilla y León	93	26	119
Castilla-La Mancha	70	18	88
Cataluña	206	118	324
Ceuta	1		1
Comunidad de Madrid	128	99	227
Comunidad Foral de Navarra	55	9	64
Comunidad Valenciana	74	406	480
Extremadura	39	10	49
Galicia	106	74	180
Islas Baleares	45	21	66
Islas Canarias	38	7	45
La Rioja	15	4	19
Melilla	2		2
País Vasco	50	60	110
Principado de Asturias	26	12	38
Región de Murcia	15	16	31

Figura 1- Centros de enseñanza musical actualmente en España [5]

[1] FSMCV: https://fsmcv.org/red-de-centros-educativos/.

Por todo esto que se decidió enfocar este TFG al desarrollo de un software de gestión personalizado para escuelas de música. un software que sea eficiente, intuitivo y fácil de usar, y que, a su vez, esté específicamente diseñado para las peculiaridades de las escuelas de música, este software facilitará las tareas administrativas y permitirá a los docentes centrarse más en la enseñanza y menos en la burocracia, beneficiando en última instancia a los estudiantes.

Se abordó este desafío identificando las necesidades tanto de clientes como de los usuarios, prototipando soluciones y obteniendo retroalimentación para iterar y mejorar continuamente el software hasta alcanzar una solución óptima logrando una gestión más simple más precisa y cercana.

Para ello se contó con la participación durante el desarrollo de docentes y trabajadores del Centre Estudi Musical de Almàssera CEM, es una escuela municipal donde se imparten las enseñanzas elementales, y es en este centro donde comencé mi trayectoria musical. Ellos por lo tanto se convierten en un *early adopter*[2] clave para este proyecto.

1.2 Objetivos

El objetivo principal de este trabajo de fin de grado consiste en desarrollar MusicApp, un software de gestión para escuelas de música que optimice procesos administrativos y mejore la eficiencia en la gestión académica.

Al tratarse de un enfoque de TFG de emprendimiento se busca establecer un punto de partida sólido para el despliegue la comercialización y monetización del software en un futuro. Para ello será clave:

- Desarrollar y presentar, al menos, un Producto Mínimo Viable (MVP[3]) para validar el modelo de negocio, obteniendo retroalimentación y una valoración directa de usuarios.

- Alcanzar una tasa de aceptación del 70% entre los usuarios participantes en el experimento, para medir dicha aceptación se emplearán métodos como encuestas anónimas a dichos participantes.

- Asegurar un desarrollo limpio y que siga buenas prácticas para garantizar su fácil mantenimiento y capacidad de escalabilidad y así adaptarse sin problemas a las crecientes demandas y evolución que pueda enfrentar el software en un futuro.

[2] **Early adopter**: primeros consumidores que compran un producto o adquieren un servicio recién lanzado al mercado.
[3] **Minimum viable product(MVP)**: versión mínima de un producto para obtener valoraciones de los clientes antes de invertir más recursos en este.

En cuanto a los objetivos relacionados con las funcionalidades del software a desarrollar encontramos:

- Desarrollar un software multiplataforma, funcional en iOS, Android y web
- Garantizar un software rápido, intuitivo y fácil de usar para usuarios con diferentes niveles de experiencia
- Brindar a los usuarios la posibilidad de consultar registros de asistencia y notas.
- Implementar medidas de seguridad robustas para proteger los datos y la privacidad de los usuarios.
- Garantizar a los administradores de la escuela la monitorización y gestión de los usuarios del software de dicha escuela. Respetando la separación de responsabilidades y privilegios con una buena gestión de roles y permisos.
- Brindar a los profesores la posibilidad de poner faltas y evaluar a los alumnos.

1.3 Estructura de la memoria

El proceso de desarrollo del proyecto y por ende la memoria se han estructurado de la siguiente manera, se presentan 6 capítulos principales, detallando una fase del desarrollo en cada uno de ellos, posteriormente se adjunta el apartado de los anexos y las referencias bibliográficas.

Capítulo 2: Evaluación de la idea de negocio

Para llevar a cabo esta evaluación, se estudia si existe una demanda real para el servicio que ofrecemos, determinando quiénes son los clientes potenciales. Posteriormente, se realiza un estudio exhaustivo de las soluciones de nuestros competidores, culminando en una comparación de las características que ofrece cada una frente a las ofrecidas por nuestra solución. Se realiza además un análisis DAFO para reconocer los aspectos positivos y negativos de la idea de negocio. También se lleva a cabo una proyección de los ingresos y los gastos que generará la plataforma una vez esté puesta en marcha. Por último, se realiza un Lean Canvas seguido de las conclusiones de la evaluación.

Capítulo 3: Tecnologías Utilizadas

En este capítulo veremos las principales tecnologías que se emplearon durante el desarrollo de este proyecto. En cada punto entraremos en detalle de cómo funcionan estas herramientas, algunos ejemplos de cómo han sido empleadas y el porqué de su uso.

Capítulo 4: Desarrollo de la idea de negocio

Para este punto detallaremos la metodología que se siguió para el desarrollo de este proyecto. Se abordan los requisitos del sistema, tanto funcionales como no funcionales, y se explica el diseño de la interfaz de usuario, así como la arquitectura del sistema. También se cubren los aspectos relacionados con la programación concreta realizada, destacando los desafíos que se han presentado, los patrones utilizados y las pruebas realizadas.

Capítulo 5: Cronología del TFG

Aquí veremos la cronología del proyecto, de manera detallada se presentarán los hitos acontecidos durante el desarrollo, acompañados de las fechas y documentando dichos hechos.

Capítulo 6: Conclusiones y trabajo futuro

Por último, en este capítulo se reflexiona sobre el trabajo realizado, si se han cumplido con los objetivos previamente detallados y se pone la mirada en el futuro, estudiando posibles nuevos pasos en el desarrollo de la plataforma y analizando la posible evolución que deberá afrontar en adelante el software.

Referencias

En este punto se enumeran de manera detallada y ordenada las referencias bibliográficas que se han consultado y han sido necesarias para la realización de este proyecto.

Anexo A: Descripción de la aplicación

En este anexo se detalla una guía de uso de la aplicación presentado todas las funcionalidades de la plataforma, tanto para la versión web como la versión móvil.

Anexo B: Alineación con los ODS

Se estudia el grado de cercanía de la solución desarrollada con los objetivos de desarrollo sostenible, analizando la alineación de las características ofrecidas con cada uno de ellos**.**

CAPÍTULO 2

Evaluación de la idea de negocio

En este capítulo analizaremos y determinaremos la viabilidad, el potencial y la rentabilidad de la idea de negocio antes de invertir tiempo y recursos significativos en su desarrollo.

2.1 Clientes

Antes de concretar quien sería nuestro cliente final hay que hacer varias consideraciones. Emplearemos el modelo TAM-SAM-SOM, que es una estrategia de análisis del mercado que facilita la definición del tamaño potencial de mercado que podría tener un proyecto [3][4]. Siguiendo esta técnica el primer paso es la identificación del mercado objetivo, en este caso el mercado de software empresarial y de administración. En la figura 2 podemos observar un gráfico que ilustra el modelo TAM-SAM-SOM.

El TAM o (*Total Available Market*), se define como el mercado total disponible, es decir, la cantidad de personas que podrían estar interesadas en adquirir nuestro producto o servicio. Para MusicApp el TAM correspondería no solo a todos los conservatorios y escuelas municipales de España, sino a todos los que existen en el mundo que deseen emplearlo.

El siguiente paso es la definición del SAM o (*Serviceable Available Market*) que hace referencia a la parte del mercado que podría cubrir con el producto o servicio actual, al tratarse de un software habría que considerar la capacidad de este de ser escalable y su accesibilidad. Tomando como referencia el producto final al que se aspira, el software podría dar servicio a un gran número de escuelas, el SAM correspondería al total de escuelas de España, alrededor de 2000 centros escuelas.

Por último, falta definir el SOM o (*Serviceable Obtainable Market*) que haría referencia al volumen del mercado que se puede conseguir a corto o medio plazo, podríamos incluir en este grupo a las escuelas de la Comunidad Valenciana siendo alrededor de 500 escuelas.

En nuestro caso tenemos la suerte de contar ya con un early adopter que más adelante se podría considerar un cliente. Normalmente los profesores de estas escuelas, sobre todo los que imparten clases de un determinado instrumento, no suelen trabajar únicamente en una escuela, debido a que su capacidad de trabajo puede en algunos casos no verse completa, debido a que la carga de alumnos para ese instrumento no es numerosa y pueden impartir en otros centros. Esto se da en el CEM, de esta manera también se cuenta con una red de contactos con otras escuelas municipales o conservatorios hacia los que se podría expandir este volumen de mercado obtenible, contando así con una buena base de posibles clientes.

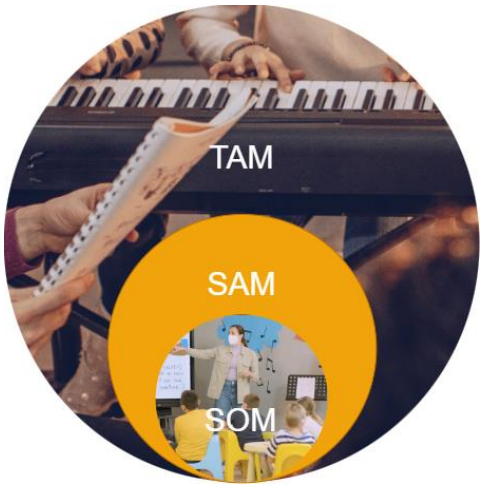

Figura 2 - Modelo TAM-SAM-SOM

MusicApp es un software dirigido a centros de enseñanza musical, sus usuarios, serán tanto los trabajadores como los padres de estudiantes de dichos centros e incluso los propios estudiantes llegada cierta edad. En ningún caso se pretende que el usuario sea el cliente, el cliente podrá variar según el ámbito de la escuela. Para aplicar el software en una escuela o conservatorio privado, sería responsable del pago la propia institución. En cambio, para centros públicos, esta gestión se llevaría a cabo junto con la entidad administrativa correspondiente.

Para determinar el servicio que se le brindará al cliente, es necesario considerar para cada uno, el tipo de centro que es y las enseñanzas que imparte, porque será proporcional a la cantidad de usuarios e información que deberá soportar el software.

2.2 Estudio de mercado y competidores

Para evaluar la viabilidad de la idea de negocio, se llevará a cabo un estudio de mercado y análisis de competidores. Se analizarán las siguientes soluciones competidoras: Ender, Academity, Additio App y Glissandoo

Dado que el software administrativo a menudo no está disponible para pruebas completas, la evaluación de estos productos competidores se basará principalmente en información obtenida de sus sitios web oficiales y en experiencias compartidas por usuarios que han tenido la oportunidad de utilizarlos.

Para realizar el estudio de mercado, se recopilará información detallada sobre cada uno de estos productos, enfocándose en aspectos clave como las funcionalidades ofrecidas, la facilidad de uso, el soporte técnico, y las opiniones de los usuarios. Junto con nuestra propuesta, esta información se sintetizará en una tabla comparativa.

Ender

Figura 3- Ender Atenea

En su página oficial, Ender[4] ofrece diversos servicios, entre los que destacamos un software de gestión denominado Atenea que va dirigido a centros de formación [6]. Podemos ver en la figura 3 una imagen corporativa de la web de la empresa. Esta web incluye una demo de Atenea, permitiendo a los usuarios explorar sus funcionalidades. El enfoque de este servicio es proporcionar una serie de características principales en su versión básica, como la gestión de profesorado, alumnos, evaluaciones, entre otras.

Además, se ofrecen funcionalidades adicionales que no están incluidos en el software base y se adquieren por separado. Entre estos módulos adicionales se encuentran la integración con la web del centro, el Aula virtual para realización de clases online, la aplicación móvil para los usuarios de los centros, disponible para Android e iOS, el control de jornadas laborales del personal o la gestión de ventas de material dentro del centro.

[4] Ender: Ender - Software de gestión para academias.

Después de poder utilizar el software empleando la demo, se puede decir con seguridad que no es un software orientado a la facilidad de uso. Debido a la diversa funcionalidad que presenta no es fácil de usar en un comienzo y es necesario familiarizarse o consultar un guía, ellos mismo adjuntan en su web un enlace al *Manual Atenea*, dicho manual consta de 10 capítulos con hasta 20 temas en cada uno.

Tiene un modelo de negocio de suscripción, existen 4 planes: ERP, CAMPUS, AULA y ECOMM. Los precios de estos planes se pueden apreciar en la figura 4. Dan la posibilidad de hacer estos pagos mensuales o anuales aplicando un pequeño descuento, al exigir una permanencia de 12 meses esta distinción solo sirve para dar la falsa ilusión de descuento, además los precios indicados no incluyen IVA. Cada uno de estos planes incluye la funcionalidad del anterior y algunas funcionalidades añadidas, a parte de estos planes, se contratan los módulos adicionales comentados anteriormente, para configurar Atenea a medida del consumidor.

Figura 4 - Pricing Ender

En Ender también contemplan la posibilidad de emplear el programa Digital Toolkit o Kit Digital para contratar sus servicios. Este programa es una iniciativa promovida por el Gobierno de España, orientada a fomentar la digitalización de las pequeñas y medianas empresas (pymes) y de los autónomos. Este forma parte del Plan de Recuperación, Transformación y Resiliencia de la economía española, el cual es financiado a través de los fondos Next Generation EU[7].

Se proporcionan estas ayudas para la adopción de soluciones tecnológicas destinadas a la implementación de diversas herramientas digitales y corresponden a pagos únicos para la implementación inicial. Para que una empresa desarrolladora pueda considerarse un Agente digitalizador y ofrecer sus servicios a través del Kit Digital es crucial que sus soluciones tecnológicas estén alineadas con los requisitos y categorías especificadas por el programa. Estas categorías son las siguientes: sitio web y presencia en Internet, comercio electrónico, gestión de redes sociales, gestión de clientes, inteligencia empresarial y analítica, servicios y herramientas de oficina virtual, gestión de procesos, factura electrónica, comunicaciones seguras y ciberseguridad [8] [9]. La empresa desarrolladora debe asegurarse de que sus productos o servicios son elegibles para recibir las subvenciones del Kit Digital.

En el caso de modelos de negocio basados en suscripción, la empresa desarrolladora podría adaptarse ofreciendo un modelo de pago único inicial. Esto permite que el cliente pueda utilizar la subvención del Kit Digital para cubrir el coste inicial y luego continuar pagando las cuotas de suscripción según lo acordado. Lo que se hace en Ender es establecer un rango de precios para cada uno de sus servicios, en el caso de Atenea el precio base es de 1.350€ + IVA, es importante destacar que estos servicios tienen una duración de 12 meses, después podemos deducir que ya no contemplaran este servicio para el mismo centro cliente. También clasifican sus servicios según las soluciones digitales subvencionadas por el programa Kit Digital. Atenea entraría en el marco de los servicios de *gestión de clientes* y *gestión de procesos*.

Academity

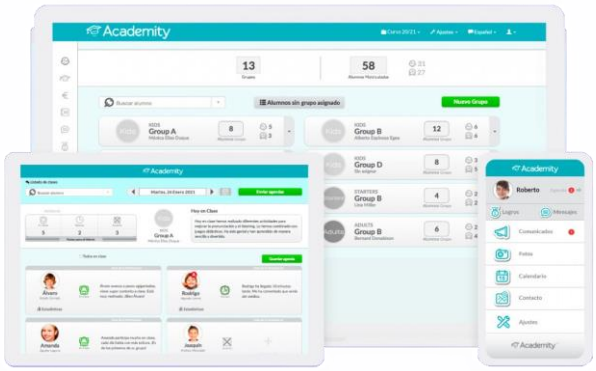

Figura 5 - Academity

The Long Finger Company[5] es una empresa desarrolladora que ofrece diversos productos, entre ellos Academity[6], una aplicación web destinada a centros de formación. Esta solución incluye una aplicación para móviles y tablets para el alumnado y sus familias. En la figura 5 podemos observar algunas de las interfaces de la plataforma y como se puede observar existe compatibilidad para dispositivos móviles y web.

Academity fue utilizada durante el curso 2022-2023 en el CEM, y según nos han comentado, en abril de 2024, los usuarios aún pueden acceder a la herramienta a pesar de haber cancelado el servicio casi un año antes. Por lo que se pudo ver con mayor detalle sus funcionalidades. Una de las características que destacan en su web sobre Academity es la personalización de la app móvil, permitiendo a las instituciones fortalecer su imagen de marca mediante la personalización del ícono y el nombre de la aplicación. Otra función destacable que ofrece Academity es la gestión administrativa, permitiendo la creación de servicios con las distintas ofertas del centro, la generación de recibos y facturas, así como la posibilidad de utilizar diferentes métodos de pago.

[5] **The long finger company:** https://thelongfingercompany.com/..
[6] **Academity:** https://academity.es/.

Durante una reunión con los docentes del CEM, se indicó que dejaron de usar el servicio porque no se adaptaba a las necesidades específicas de la escuela y no lo consideraban práctico. En cuanto a la información de precios, la página web de Academity no proporciona detalles explícitos, pero se sabe por la experiencia del CEM que el costo de la suscripción mensual era aproximadamente 60€, sin compromiso de permanencia.

Academity también es compatible con el Kit Digital, pero no hace una distinción por producto, sino que clasifica por tipos de servicios de los establecidos por el programa. Los precios se estructuran de la siguiente manera:

- Gestión de procesos: Precio fijo de 1.500€ más impuestos.
- Servicios y herramientas de oficina virtual: 250€ por usuario, sin incluir impuestos.
- Gestión de clientes: Precio fijo de 1.500€ más impuestos.

Academity se menciona en las tres categorías, lo que genera incertidumbre sobre si existe un precio final que abarque todas las funcionalidades o si se contratan de manera modular. Al igual que otros competidores, los servicios ofrecidos bajo el Kit Digital tienen una validez de 12 meses.

Additio App

Figura 6 - Additio App

Additio App[7] es una plataforma integral de gestión educativa diseñada para mejorar la administración escolar y la dinámica en el aula [10]. Dirigida tanto a profesores individuales como a instituciones educativas completas, esta herramienta ofrece funcionalidades que incrementan la eficiencia y facilitan la comunicación en el entorno educativo. En la figura 6 se muestra una imagen de la web de Additio.

[7] **Additio App:** https://additioapp.com/.

Para los profesores, Additio proporciona herramientas avanzadas que simplifican la evaluación formativa, la gestión del aula y la planificación de clases. Esto incluye la personalización de criterios de evaluación, seguimiento de asistencia y generación de informes de progreso y rendimiento de los estudiantes.

A nivel institucional, Additio no establece un precio y ofrece funcionalidades para la gestión académica, financiera y administrativa, tales como gestión de horarios, pagos, matrículas e informes. Además, facilita la comunicación interna y entre el personal escolar y las familias.

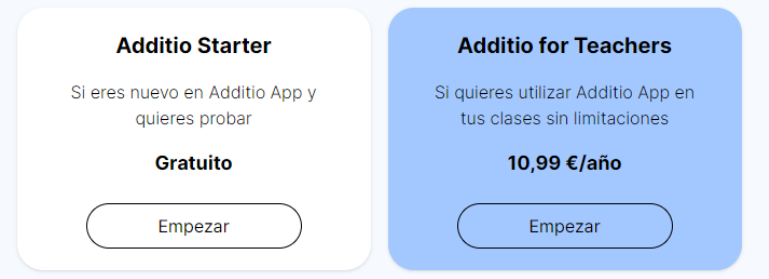

Figura 7 - Additio App planes individuales

Figura 8 - Additio App planes para centros

Disponible en dispositivos web, Android e iOS, Additio permite a los docentes acceder a sus datos y herramientas en cualquier momento y lugar. Además, se integra con plataformas educativas populares como Google Classroom[8], Moodle[9] y Microsoft Teams[10], optimizando la experiencia educativa al sincronizar datos y actividades.

El modelo de negocio se basa en suscripciones con opciones gratuitas y premium. Los planes gratuitos ofrecen funcionalidades básicas, mientras que los planes premium están diseñados para instituciones más grandes. Esta estructura hace que Additio sea accesible tanto para profesores individuales como para grandes instituciones educativas. En las figuras 7 y 8 podemos ver sus planes de suscripción.

[8] **Google Classroom:** https://classroom.google.com/.
[9] **Moodle:** https://moodle.org/?lang=es.
[10] **Teams:** https://www.microsoft.com/es-es/microsoft-teams/group-chat-software.

Glissandoo

Figura 9 - Glissandoo

Este último competidor a analizar es una *Startup* valenciana que se encuentra en la Universitat Politècnica de València, no es un software para centros educativos, sino que es una aplicación móvil diseñada para facilitar la gestión de grupos musicales. Glissandoo [11]una variedad de herramientas para coordinar ensayos, conciertos y la distribución del repertorio musical de manera eficiente y organizada. La idea principal detrás de Glissandoo es simplificar la vida de los directores y miembros de bandas, coros y orquestas. Aunque no se trate de una app para gestión de escuelas de música, se centra en este sector y es realmente popular, por lo que cabe tenerla en consideración. En la figura 9 podemos ver una ilustración de la web de Glissandoo.

Una de las características destacadas de Glissandoo es su capacidad para gestionar los miembros de un grupo. La aplicación permite registrar la asistencia, enviar comunicaciones específicas y mantener un seguimiento de la participación de cada integrante. Además, ofrece una función para la distribución digital del repertorio, lo que facilita que todos los miembros tengan acceso a las partituras y materiales necesarios en cualquier momento y desde cualquier lugar.

Otra funcionalidad importante de Glissandoo es la planificación y organización de eventos. Los directores pueden programar ensayos y conciertos, y la aplicación permite prever la asistencia de los miembros, lo que ayuda a coordinar mejor las actividades del grupo. También ofrece la posibilidad de subir videos y audios relacionados con el repertorio.

Como podemos observar estas funciones no se alejan tanto de lo que sería necesario en una herramienta de gestión para un escuela de música y por eso es importante tomar a Glissandoo como ejemplo.

[11] **Glissandoo:** https://glissandoo.com/.

Figura 10 - Pricing Glissandoo

El modelo de negocio de Glissandoo puede verse en la figura 10, se basa en ofrecer una plataforma gratuita con funcionalidades esenciales para grupos musicales pequeños y opciones de suscripción para grupos más grandes o instituciones con múltiples conjuntos. También da la opción de realizar un pago anual, aplicando un descuento a la suscripción. La aplicación está disponible tanto en la App Store para dispositivos Apple como en Google Play Store para dispositivos Android.

Además de esto, Glissandoo tiene acuerdos de colaboración con la Confederación Española de Sociedades Musicales (COESSM[12]) para fomentar la digitalización en el sector [11], por lo cual las distintas federaciones que la componen cuentan con un descuento en el precio al contratar este software. En algunos casos, como con la Federación de Bandas de Música de Navarra (FBMN[13]), la propia federación se ofrece a abonar hasta 95 € de ayuda para cubrir el gasto a los miembros interesados en emplear Glissandoo [12].

Tabla comparativa

A continuación, se presenta la tabla 1 con la comparativa de los productos software competidores nombrados en esta sección, estudiando para cada uno de ellos, sus características funcionales concretas y algunos aspectos sobre el modelo de negocio que siguen las empresas para estas soluciones.

[12] **COESSM:** https://coessm.org/.
[13] **FBMN:** https://www.bandasdenavarra.com/.

Aspectos a comparar	**Academity**	**Ender**	**Additio App**	**Glissandoo**	**MusicApp**
gestión de grupos	✓	✓	✓	—	✓
Ficha de alumno	✓	✓	✓	—	✓
Ficha de profesor	✓	✓	✓	—	✓
Registro de jornada laboral	✓	✓	—	—	✓
Registros de cursos pasados	✓	✓	✓	—	✓
Gestión administrativa y económica	✓	✓	✓	—	—
Mensajería interna	✓	✓	✓	—	✓
Mensajería por correo externo	—	✓	—	—	✓
Comunicaciones generales del centro	✓	✓	—	—	✓
Repositorios de contenido multimedia	✓	✓	✓	✓	✓
Calendario de actividades y horario	✓	✓	✓	—	✓
Clases online	✓	✓	✓	—	—
Reuniones online	—	—	✓	—	—
Agenda, gestión diaria de asistencia	✓	✓	✓	✓	✓
Gestión de criterios de evaluación	✓	✓	✓	—	✓
Logros y gamificación	✓	—	—	—	✓
Información de asistencia en tiempo real	✓	✓	✓	—	✓
Matriculación online	—	✓	✓	—	—
Herramienta web	✓	✓	✓	—	✓
Aplicación móvil	✓	✓	✓	✓	✓
Distribución de partituras	—	—	—	✓	—
Previsión de asistencia	—	—	—	✓	✓
Sin Permanencia	✓	12 meses	✓	✓	✓
Agente digitalizador del programa Kit Digital	✓	✓	—	—	—

	Academity	Ender	Additio App	Glissandoo	MusicApp
Exportación de información a XLS/ CSV/ TXT/ XML	✓	—	✓	—	✓
Curva de aprendizaje	media	alta	media	baja	baja
Interfaz intuitiva	—	—	—	✓	✓
Estadísticas e informes de usuarios	—	—	✓	✓	✓
Informes de gastos y pagos	—	✓	✓	—	—
Segmentación por instrumentos	—	—	—	✓	✓
Precio independiente del número de alumnos	—	✓	✓	—	—
Versión gratuita limitada	—	—	✓	✓	—
Integración con otras entornos de aprendizaje	—	✓	✓	—	—
Integración con firma digital	—	—	✓	—	—
Datos compartidos de los grupos entre profesores	—	—	✓	—	✓
Planificación de clases	—	—	✓	—	—
Planes de pago mensuales	✓	✓	—	✓	✓
Planes de pago anuales	✓	✓	✓	✓	✓
Venta de productos	—	✓	—	—	—
Inventario y préstamo de instrumentos	—	—	—	—	✓

Tabla 1 - Comparativa competidores

Podemos apreciar que MusicApp es un digno competidor respecto de estas otras solucionas, a pesar de no incurrir en el ámbito administrativo cuanto con numerosas funcionalidades de gestión como la gestión de asistencia en tiempo real, las comunicaciones internas , la disponibilidad de calendarios o el registro de la jornada laboral. También hay características distintivas de MusicApp como la gestión de inventario y préstamo de instrumentos los logros y la gamificación y el contar con una plataforma disponible tanto en móvil como en versión web.

2.3 Modelo de negocio

Basándose en las soluciones analizadas en el punto anterior además de otras herramientas del sector y las necesidades propias de software MusicApp, se ha decido aplicar un modelo de negocio mediante pagos por suscripción.

Modelos de negocio no aplicables

Se estudiado y descartado otros modelos de negocio para nuestra solución, a continuación, se concretarás más sobra cada opción descartada y los motivos que han llevado a esta decisión.

Modelo Freemium

Al tratarse un software empresarial orientado a los centros, no tiene sentido ofrecer una versión gratuita limitada debido a que al limitar la funcionalidad para poder ofrecer esta versión gratuita podría dejar de ser un software funcional, además también podría ocasionar poca rentabilización del producto, debido a que poco cliente de esta versión gratuita tuviera la necesidad pasar a un plan de pago, al serle suficiente este plan gratuito.

Modelo de publicidad

El modelo de negocio de publicidad se basa en ofrecer contenido o servicios gratuitos a cambio de la exposición de contenido publicitario que generan ingresos. Para poder emplear esta dinámica sería necesario un volumen muy elevado de usuarios, garantizando así un beneficio adecuado por el uso de anuncios [13] . Al no tener garantía de esta gran cantidad de usuarios, se ha optado por descartar esta opción.

Modelo híbrido entre venta directa y suscripción

Hemos visto que algunos de los competidores anteriores, empleaban un modelo de negocio hibrido, integrando el de venta directa al asignar un precio fijo a sus productos o servicios y así poder adecuarse a los requerimientos que tenía el Kit Digital. También se aclaraba que este tipo de servicios tenían una validez establecida y pasada la fecha el cliente debía volver al modelo de suscripción. Después de estudiarlo detenidamente y teniendo principalmente en cuenta los requisitos para poder ofrecer servicios mediante este programa de ayudas, se ha considerado que esta opción no es viable actualmente y ni para un futuro próximo.

Modelo de negocio final de MusicApp: modelo por suscripción

Después de analizar las opciones comentadas, nos decantamos por que MusicApp siga un modelo de negocio por suscripción, veamos algunas de las ventajas que nos ofrece este modelo:

- **Ingresos Predecibles:** Las suscripciones mensuales o anuales proporcionan un flujo de ingresos estable y predecible, lo cual facilita la planificación financiera y la sostenibilidad a largo plazo de MusicApp.
- **Fidelización de Clientes:** Los clientes suscritos tienen un compromiso continuo con el servicio, lo que puede resultar en una menor tasa de cancelación y una base de usuarios más leal.
- **Escalabilidad:** Se permite adaptar los planes de suscripción según las necesidades y el tamaño de las academias musicales.

Para definir los planes de suscripción se tendrá en cuenta el número de usuarios del servicio, se entenderá esta cifra como la cantidad de alumnos más la cantidad de trabajadores del centro, además de los usuarios extra que se requieran para familias con padres separados. Se ofrecerá un plan básico predefinido, con un precio de 55€ al mes y con limitación de hasta 100 usuarios. Este plan básico, incluirá limitaciones en cuanto al almacenamiento, 8 GB por centro, afectando principalmente a la cantidad de materiales multimedia que puedan subirse a la plataforma. Esta limitación surge de la necesidad y se impone debido a las restricciones inherentes a la tecnología utilizada, y las limitaciones propias de sus planes de suscripción. Se entrará más en detalle sobre esto en el siguiente apartado *2.4 proyección de ingresos y gastos*.

Además de este plan habrá un plan personalizable para los centros que no quieran el plan básico o no cumplan con sus requisitos. El precio de este plan personalizado vendrá dado por el volumen de usuarios que se pretende comprender para ese centro. Además, se dará la posibilidad de realizar pagos mensuales o pagos anuales, aplicando para este último un descuento del 16%. En la figura 11 se muestra una representación de los planes establecidos.

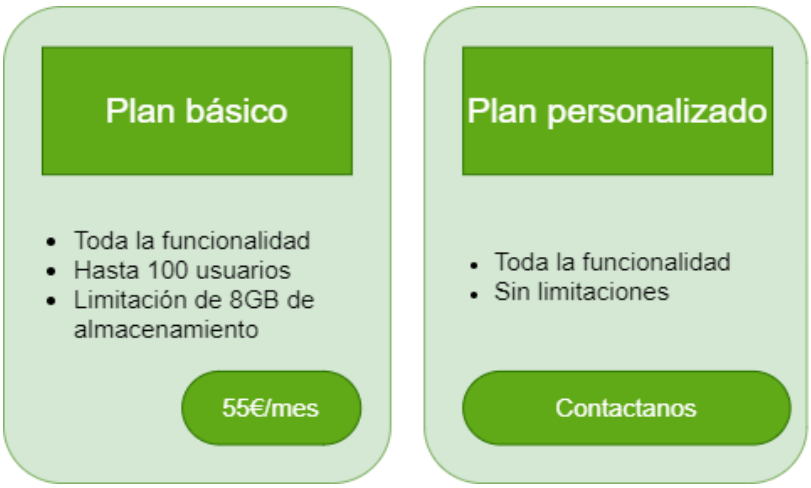

Figura 11 - Resumen Pricing MusicApp

2.4 Análisis DAFO

Para poder visualizar de manera clara los puntos a favor y en contra de esta idea, y así poder tener una base sólida sobre la que desarrollar vamos a realizar un análisis DAFO [14].

	ORIGEN INTERNO	ORIGEN EXTERNO
	DEBILIDADES	**AMENAZAS**
NEGATIVOS	• Sin experienza en el sector empresarial • Sin reconocimiento de marca • No se contempla la gestión económica	• Existen otras soluciones de software para la gestión de instituciones educativas. • Las escuelas pueden ser reacias a adoptar nuevas tecnologías si están utilizando otros sistemas o no cuentan con personal capacitado.
	FORTALEZAS	**OPORTUNIDADES**
POSITIVOS	• Solución específicamente diseñada y con funcionalidades concretas para escuelas de música. • La solución actual sería funcional aún sin ningún coste de mantenimiento.	• Nicho de mercado con un volumen elevado. • Demanda creciente de este tipo de plataformas debido a la tendencia hacia la digitalización.

Tabla 2 - Análisis DAFO

Como se observa en el análisis de la tabla 2, A pesar de que la solución no cuenta con experiencia en el sector empresarial ni reconocimiento de marca, y no contempla la gestión económica, presenta oportunidades significativas en un nicho de mercado con un volumen elevado. La creciente demanda de plataformas digitales impulsa esta tendencia, lo que puede favorecer su adopción.

Además, la competencia con aplicaciones similares representa una amenaza. Sin embargo, la solución destaca por estar específicamente diseñada para academias de música, ofreciendo funcionalidades concretas. La primera versión de MusicApp podría ser funcional para un número reducido de academias antes de que se necesitara dedicar más recursos a la infraestructura porque lo que tempranamente estaría operando de manera funcional sin incurrir en costes de mantenimiento adicionales.

2.5 Proyección de ingresos y gastos

La proyección de ingresos y gastos es una herramienta financiera esencial que permite anticipar los flujos de caja de una empresa, proyecto o producto en el futuro. De esta manera nos ayudará a determinar la viabilidad de nuestro proyecto MusicApp además de ofrecer información crucial para la planificación estratégica y la toma de decisiones empresariales [15].

Para esta proyección se ha planteado estudiar los 5 primeros años de funcionamiento de la plataforma, además supondremos el escenario ideal, donde se contará con financiación y los clientes esperados. Como se ha comentado en el apartado 2.3 MusicApp presenta 2 planes de suscripción: el plan básico y el personalizado, el precio del primero será de 55€ al mes. Para la suscripción personalizada se va a suponer una media de precio de 90€ para la que realizaremos los cálculos. También se supondrá que las contrataciones se han realizado con pagos anuales por lo que se aplicará un 5% de descuento a las cuotas. Podemos ver estos precios en la tabla 3.

Precio de suscripción base anual (5% descuento)	627,00 €
Media de precio de suscripción personalizada anual (5% descuento)	1.026,00 €

Tabla 3 - Pricing para la proyección de ingresos y gatos

Basándose en los competidores estudiados y en la cantidad de escuelas que empleaban sus servicios se ha establecido la siguiente proyección esperada de clientes. Este análisis nos permite estimar el número de nuevas suscripciones que se espera que se contraten cada año. Además se supondrá porcentaje de fidelización con los clientes del 85%. Según esta estimación de ventas de suscripciones, el quinto año contaríamos con 8 escuelas con planes básicos y 6 con planes personalizados contratados.

Licencias vendidas	**Año 1**	**Año 2**	**Año 3**	**Año 4**	**Año 5**
Suscripciones básicas vendidas	15	27	49	87	157
Suscripciones básicas renovadas	0	13	23	41	74
Suscripciones personalizadas vendidas	6	11	19	35	63
Suscripciones personalizadas renovadas	0	5	9	17	30
Total de suscripciones basicas	15	40	72	129	232
Total de suscripciones personalizadas	6	16	29	52	93

Tabla 4 - Proyección de venta de suscripciones

Durante el primer año, se contratará a un desarrollador para mejorar el software y así lograr una solución comercializable. También se contrataría a un equipo de diseño para realizar un rebranding y un diseño completo de la plataforma.

A partir del segundo año, se llevará a cabo una campaña de marketing, invirtiendo en publicidad en redes sociales, SEO y marketing digital. Otro gasto a destacar sería el dominio de la plataforma y los servicios SMTP para el envío de correos. Estos incurrirían en costos a partir del tercer año, cuando se comenzará a utilizar proveedores SMTP para satisfacer las necesidades de la plataforma y gestionar un número más elevado de clientes.

Uno de los gastos principales surge del mantenimiento de la infraestructura de la solución, para esto se ha empleado Supabase[14], una plataforma BaaS (Backend as a Service) que facilita una amplia gama de herramientas para la gestión y la creación del Backend, se concretará más información sobre el funcionamiento de Supabase y las funcionalidades que ofrece y que se han empleado para el desarrollo en el capítulo 3 Tecnologías utilizadas.

Este servicio presenta suscripciones de uso, durante el desarrollo se ha empleado la licencia gratuita, que ofrece toda la funcionalidad de Supabase con limitaciones. Estas limitaciones hacen que la suscripción gratuita no sea suficiente para un aplicación en producción, por lo que sería necesario cambiar a un plan superior.

Licencias vendidas	Año 1	Año 2	Año 3	Año 4	Año 5
Total de suscripciones básicas	15	40	72	129	232
Total de suscripciones personalizadas	6	16	29	52	93
Ingresos Anuales					
Ingresos por suscripciones básicas	6.156,00 €	16.313 €	29.364 €	52.855 €	95.140 €
Ingresos por suscripciones personalizadas	15.561 €	41.237 €	74.226 €	133.607 €	240.492 €
Total Ingresos	21.717 €	57.550 €	103.590 €	186.462 €	335.632 €
Gastos Anuales					
Infraestructura Cloud Supabase	300 €	300 €	300 €	300 €	300 €
Dominio	10 €	10 €	10 €	10 €	10 €
Proveedores SMTP	- €	130 €	130 €	130 €	130 €
Publicidad en redes sociales	300 €	300 €	500 €	500 €	800 €
SEO y Marketing Digital	- €	1.200 €	850 €	2.000 €	2.000 €
Diseño y rebranding	8.000 €	5.000 €	1.000 €	1.000 €	1.000 €
CEO	15.000 €	18.000 €	21.600 €	25.920 €	31.104 €
Desarrollador	10.000 €	12.000 €	14.400 €	17.280 €	20.736 €
Total Gastos	33.610 €	36.940 €	38.790 €	47.140 €	56.080 €
Resultado Anual	-18.049 €	4.297 €	35.436 €	86.467 €	184.412 €
Resultado Anual Acumulado	-18.049 €	-13.752 €	21.684 €	108.150 €	292.563 €

Tabla 5 – Proyección de ingresos y gastos

[14] **Supabase:** https://supabase.com/.

Podemos observar que, a partir del segundo año, los beneficios generados serán mayores que a los gastos de ese año, superando los resultados negativos del primer año. A partir del tercer año como se puede observar en la figura 12 se comienzan a generar beneficios.

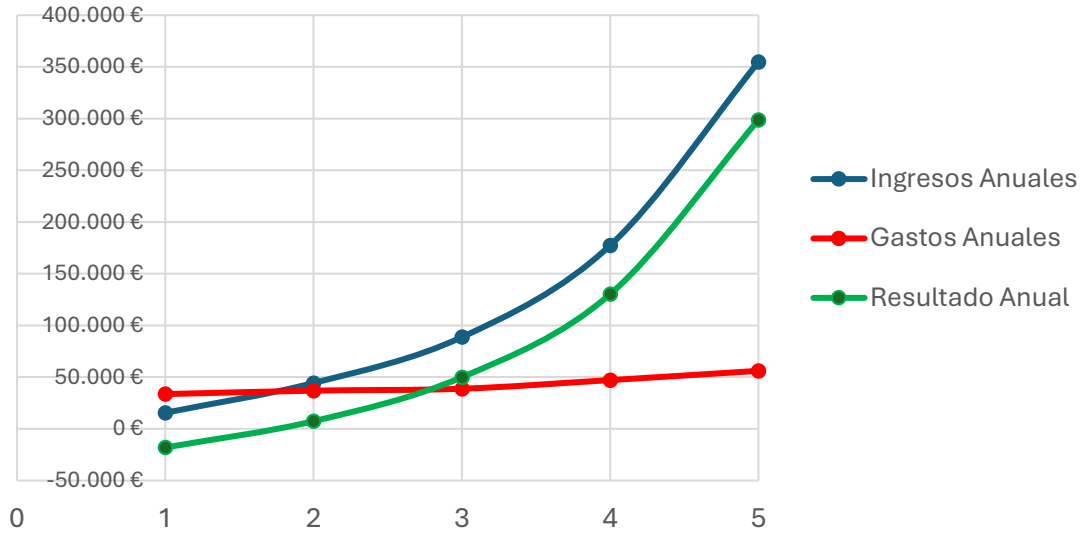

Figura 12 - Ingresos y gastos anuales

2.6 Lean Canvas

El Lean Canvas es una herramienta de gestión estratégica con un enfoque más ágil y orientado hacia la validación rápida de ideas utilizada principalmente en el ámbito de startups y emprendimientos para visualizar y desarrollar modelos de negocio de manera concisa y efectiva [16].

En la Figura 13 se presenta el Lean Canvas completo, donde se han detallado los puntos clave: segmentos de clientes, propuesta de valor única, canales, problemas, soluciones, métricas clave, ventaja injusta, early adopters, estructura de costes y flujo de ingresos.

Problemas

Las escuelas de música carecen de herramientas digitales personalizadas que faciliten sus tareas diarias y relacionadas con los servicios que ofrecen. Las soluciones de software actuales están enfocadas en centros educativos en general, incluyendo funcionalidades innecesarias para las academias de música y careciendo de otras que sí son necesarias.

Alternativas existentes

Actualmente, las escuelas realizan estas tareas a papel o mediante el uso de extensas hojas de cálculo en Excel, lo cual puede resultar engorroso e incómodo.

Soluciones

Desarrollar un software fácil de usar e intuitivo que permita a estas escuelas realizar y gestionar sus tareas de manera cómoda y eficaz.

Canales

Su crearan cuentas en redes sociales como Instagram, Facebook o Twitter y se publicitará la plataforma.

Se crearía una landing page para presentar el software, sus funcionalidades y los planes de suscripción.

Propuesta única de valor

Aplicación multiplataforma que ofrece a las escuelas de música una herramienta de gestión integral, diseñada específicamente para cubrir todas sus necesidades operativas. Esta solución garantiza una experiencia cómoda de usar y eficaz, permitiendo a las escuelas realizar sus tareas de manera fluida, optimizando procesos y mejorando la experiencia tanto de los adminitradores y trabajadores como de estudiantes

Ventaja injusta

El desarrollo de la aplicación enfocado de manera exclusiva a las escuelas de música nos permite diferenciarnos de otras soluciones genéricas, ofrecienddo funcionalidades que realmente cubren las necesidades específicas de este sector

Métricas clave

- Número de usuarios activos
- Número de suscripciones activa
- Tasa de retención de las suscripciones

Segmentos de clientes

Nuestros clientes son escuelas de música que buscan una solución integral y especializada para gestionar eficientemente sus actividades educativas y administrativas.
Están interesados en una plataforma fácil de usar y que ofreazca herrramientas avanzadas que mejoren la ezperiencia tanto para los administradores como para los estudiantes

Early adopters

Trabajadores y estudiantes del Centre Estudi Musical de Almássera

Estrucutura de costes

- Mantenimiento de la infraestructura en Supabase
- Publicaciones de la app móvil en App Store y en Google Play store.
- Dominio de la plataforma
- Proveedores SMTP
- Publicidad en redes sociales y SEO

Flujos de ingresos

- Suscripciones básicas contratadas
- Suscripciones personalizadas contratadas
- Posible inversión inicial

Figura 13 - Lean Canvas

2.7 Conclusiones de la evaluación

La elección de un modelo de negocio por suscripción de MusicApp ofrece una estructura sólida para garantizar ingresos estables y sostenibles a través de pagos recurrentes de suscripción. Este enfoque facilita la gestión financiera al prever ingresos a largo plazo y promueve la fidelización y retención de clientes mediante una relación continua y beneficiosa para ambas partes. La capacidad de ajustar los planes de suscripción según el tamaño y las necesidades específicas de cada academia de música mejora la satisfacción del cliente y también permite a MusicApp adaptarse ágilmente a las demandas del mercado.

La propuesta de valor de MusicApp se centra en resolver las necesidades específicas de las academias de música, como la gestión de alumnos, materiales multimedia y comunicación efectiva entre el personal y los estudiantes. Esto no solo podría mejorar la eficiencia operativa de las academias, sino también mejorar la experiencia educativa para los alumnos y aumentar la satisfacción de los clientes.

La idea también enfrenta varios desafíos. La falta de reconocimiento de marca y la competencia en el mercado de software educativo podrían dificultar la adquisición inicial de clientes. Además, la limitación en el almacenamiento y otras funcionalidades debido a las restricciones tecnológicas podría afectar a la aceptación y utilidad percibida por parte de los usuarios.

La proyección financiera indica una fase inicial de inversión con resultados financieros negativos esperados durante los primeros años. Esto refleja la necesidad de una inversión inicial considerable en desarrollo de producto, marketing y expansión de infraestructura. Sin embargo, con una estrategia clara y un enfoque en la mejora continua, se anticipa que MusicApp alcance la rentabilidad en el quinto año, marcando el comienzo de una trayectoria hacia el éxito económico sostenido.

En conclusión, la idea de negocio de MusicApp presenta un potencial significativo al abordar un nicho específico con necesidades no completamente cubiertas por soluciones existentes. Sin embargo, para aprovechar esta oportunidad, será crucial desarrollar una estrategia sólida de marketing y mejora continua del producto, asegurando al mismo tiempo una experiencia de usuario robusta y escalable que pueda adaptarse al crecimiento del negocio y las demandas del mercado.

CAPÍTULO 3

Tecnologías utilizadas

3.1 TypeScript 

TypeScript[15] es un lenguaje de programación desarrollado por Microsoft que se construye sobre JavaScript añadiendo funcionalidades de tipado estático opcional. Esto significa que TypeScript permite especificar tipos para variables, parámetros de funciones, y valores de retorno, entre otros [17]. A continuación, se verán algunas de las ventajas que trae consigo el uso de este lenguaje:

- Poder definir tipos estáticos facilita la detección de errores durante la compilación reduciendo significativamente los errores en tiempo de ejecución.

- TypeScript es un *superset* de JavaScript, lo que significa que todo el código JavaScript existente es válido en TypeScript. Esto permite a los equipos adoptar gradualmente TypeScript en proyectos ya en marcha.

- TypeScript sigue de cerca las últimas especificaciones de ECMAScript[16], asegurando compatibilidad con las nuevas características del lenguaje.

- Cuenta con un ecosistema robusto de herramientas de desarrollo y una comunidad activa que contribuye con la integración de TypeScript con bibliotecas y frameworks[17] actuales.

- Las firmas de funciones en TypeScript actúan como documentación, ayudando a comprender rápidamente cómo utilizar una función.

Se ha decidido emplear este lenguaje desde el comienzo del desarrollo. De esta manera MusicApp está desarrollado por completo en TypeScript lo que facilita la mantenibilidad a largo plazo, y una base robusta sobre la que realizar pruebas gracias a este tipado.

---

[15] **TypeScript:** https://www.typescriptlang.org/.
[16] **ECMAScript:** especificación de lenguaje de programación publicada por Ecma International, https://ecma-international.org/technical-committees/tc39/.
[17] **Framework:** marco de trabajo para desarrollar softwares de manera más rápida y eficiente.

3.2 React y React Native

En primer lugar, hablaremos sobre React [18]y a partir de este como surge React Native[19]. React es una biblioteca de JavaScript utilizada para construir interfaces de usuario de manera eficiente y escalable. Desarrollada por Facebook[20], se destaca por su eficiencia y capacidad para gestionar actualizaciones eficientes del DOM. Utiliza un enfoque basado en componentes, donde cada componente encapsula una parte específica de la interfaz de usuario, lo que facilita la reutilización y el mantenimiento del código [18].

Componentes

Constituyen la unidad básica en React. Los componentes son piezas reutilizables de código que representan una parte de la interfaz de usuario. Pueden ser componentes funcionales o componentes de clase. En la figura 14 vemos un ejemplo de cada uno.

```
// Ejemplo de componente funcional
function Greeting(props) {
  return <h1>Hello, {props.name}</h1>;
}

// Ejemplo de componente de clase
class Greeting extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Figura 14 - Código 1 ejemplo React

JSX y TSX

JavaScript XML (JSX) es una extensión de sintaxis para JavaScript que permite escribir código similar a HTML[21] dentro de archivos JavaScript. JSX facilita la creación de componentes React de forma declarativa y más legible. En la figura 15 podemos ver un ejemplo muy simple de un componente de React. En nuestro caso, en lugar de JSX

[18] **React:** https://es.react.dev/.
[19] **React Native:** https://reactnative.dev/.
[20] **FaceBook:** https://www.facebook.com/.
[21] **HTML:** lenguaje que se utiliza para estructurar y una página web.

emplearemos archivos TSX lo que facilita a TypeScript a identificar que archivos contienen JSX [19].

```
1   const element = <h1>Hello, world!</h1>;
```

Figura 15 - Código 2 ejemplo React

DOM, Virtual DOM y React DOM

El DOM o (Document Object Model) es una representación en forma de árbol de los elementos HTML de una página web. Los navegadores utilizan el DOM para renderizar la interfaz de usuario y permitir la interacción con los elementos de la página. Manipular directamente el DOM puede ser ineficiente, ya que cada cambio en el DOM puede desencadenar una re-renderización completa de la página, lo que puede ser costoso en términos de rendimiento [20]. En la figura 16 puede verse una representación del DOM.

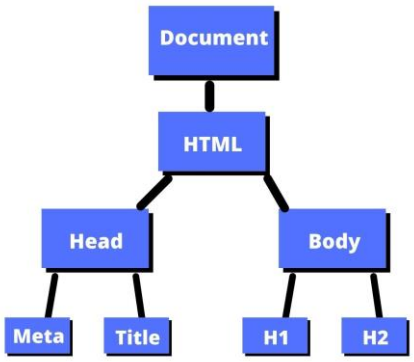

Figura 16 - Representación DOM

El DOM virtual (VDOM) es una técnica clave en React que permite mejorar el rendimiento de las aplicaciones web optimizando las actualizaciones de la interfaz de usuario. En lugar de manipular directamente el DOM del navegador, React mantiene una representación en memoria del DOM y actualiza solo los componentes que han cambiado, minimizando las operaciones costosas del DOM real [21].

React DOM es una librería complementaria a React que se utiliza para renderizar aplicaciones React específicamente en el navegador. Mientras que React se centra en la creación de componentes de interfaz de usuario y la gestión de su estado, React DOM se encarga de actualizar el DOM del navegador para que coincida con el estado de los componentes React [22].

Hooks

Los hooks de React se definen dentro de los componentes funcionales y permiten manejar el estado y otros aspectos del ciclo de vida de estos de una manera simple. A continuación, una explicación de los principales hooks y cómo gestionan el estado:

- **useState()**

```
1  function Counter() {
2    const [count, setCount] = useState(0);
3
4    return (
5      <div>
6        <p>You clicked {count} times</p>
7        <button onClick={() => setCount(count + 1)}>
8          Click me
9        </button>
10     </div>
11   );
12 }
```

Figura 17 - Ejemplo Código useState

En la figura 17 se puede observar cómo funciona el hook *useState*, este nos permite agregar estado local a los componentes funcionales, la constante guarda un par con el valor del estado actual que tiene la variable *count* y una función para actualizarlo *setCount*, suele utilizarse un nombre descriptivo para esta función. También puedes proporcionarle un valor para el estado inicial entre paréntesis, como en este caso (0).

- **useEffect()**

Este hook permite realizar efectos secundarios en componentes funcionales, es útil para fetch de datos, manipulación del DOM, suscripciones con *listeners*, etc. Se ejecuta después de que el componente se renderice.

En algunos casos, se querrá controlar cuándo debe ejecutarse *useEffect*. Esto se logra especificando estados como dependencias en un segundo argumento en *useEffect*. Cuando las dependencias cambian, *useEffect* se ejecutará nuevamente. En la figura 18 podemos ver 2 ejemplo de *useEffect* con dependencias, el primero, con dependencia vacía ([]), se ejecutará solo la primera vez que renderice el componente, el segundo cuando cambie el estado *count*.

```
const [count, setCount] = useState(0);

useEffect(() => {
 ...
}, []);

useEffect(() => {
  ...
 }, [count]);
```

Figura 18 - Ejemplo Código useEffect

- **useContext()**

En React se pueden crear contextos que sirven para compartir estado o lógica entre múltiples componentes sin pasar props manualmente a través de todos los niveles del árbol de componentes., evitando así el *prop drilling*[22]. El hook *useContext* permite consumir el valor de un contexto en componentes funcionales.

Después de crear un contexto se crea un *provider* de este, todo componente envuelto por el *provider* puede usar el hook *useContext* para conocer el contexto. En caso contrario no puede acceder al valor del contexto.

```
const ThemeContext = createContext()

export const ThemeProvider = ({ children }) => {
  const [theme, setTheme] = useState('light')

  const toggleTheme = () => {
    setTheme(theme === 'light' ? 'dark' : 'light')
  }

  return(
  <ThemeContext.Provider value={{ theme, toggleTheme }}>
    {children}
  </ThemeContext.Provider>
  )
}

export const useTheme = () => {
  return useContext(ThemeContext)
}
```

Figura 19 - Ejemplo Código creación de un contexto

[22] **Prop drilling:** los parámetros van pasándose entre componentes por toda las estructura de la aplicación como un taladro.

```jsx
const App = () => {
  const { theme, toggleTheme } = useTheme()

  return (
    <ThemeProvider>
      <div style={{ textAlign: 'center', marginTop: '50px' }}>
        <h1>Theme Toggle Example</h1>
        <button onClick={toggleTheme}>Toggle Theme</button>
      </div>
    </ThemeProvider>
  )
}
```

Figura 20 - Ejemplo Código uso del hook useContext

En las figuras 19 y 20 podemos observar un caso de uso de contextos en React, se crea el contexto del tema, además de un provider, en este caso también se crea un hook personalizado *useTheme* que llama a su vez a *useContext*, esto es considerado buena práctica para no exportar el contexto fuera del archivo donde se crea y se genera el provider. El componente App se llama a *useTheme* y con desestructuración se objectos extraemos *theme* y la función *toggleTheme* para poder usarlas en el componente App, al cambiar el estado desde App el contexto cambiará para todos los componentes suscritos a este.

React Native

React Native es una extensión de React diseñada para construir aplicaciones móviles nativas utilizando JavaScript y React. Aunque comparte muchos conceptos fundamentales con React, está optimizado específicamente para el desarrollo móvil, permitiendo a los desarrolladores crear aplicaciones para iOS y Android con una base de código compartida [23].

A diferencia de React, los componentes en React Native son, como se da a entender, nativos. Estos son un conjunto de componentes predefinidos para desarrollo en móvil. Para iOS se utilizan componentes de UIKit y para Android componentes de Android View. Estos componentes son parte integral del sistema operativo y están optimizados para ofrecer un rendimiento y una experiencia de usuario nativa.

Aunque React Native utiliza una sintaxis similar a JSX para la definición de componentes, el estilo y el diseño de la interfaz de usuario en React Native se gestionan de manera diferente. En lugar de CSS, se utilizan estilos en línea y objetos de estilo que se traducen a propiedades de estilo nativas.

```jsx
import React from 'react';
import { View, Text, Button, StyleSheet, Alert } from 'react-native';

const MyMobileComponent = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.heading}>Hello, Mobile!</Text>
      <Text>This is a simple React Native component for mobile.</Text>
      <Button title="Click me" onPress={() => Alert.alert('Button clicked!')} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    padding: 20,
    backgroundColor: '#f0f0f0',
  },
  heading: {
    color: '#333',
    fontSize: 24,
    fontWeight: 'bold',
  },
});

export default MyMobileComponent;
```

Figura 21 – Ejemplo Código React Native

Como Podemos ver en la figura 21 React Native proporciona una serie de componentes predefinidos que son equivalentes a los elementos HTML en la web. En la tabla 6 podemos ver algunos de los componentes más comunes y sus equivalencias con las librerías de Android e iOS:

El componente de React Native UI	La vista de Android	La vista de iOS	El análogo de web
<View>	ViewGroup	UIView	sin scrolling
<Text>	TextView	UITextView	<p>
<Image>	ImageView	UIImageView	<img>
<ScrollView>	ScrollView	UIScrollView	
<TexInput>	EditText	UITextField	<input type="text">

Tabla 6 - Componentes de React Native

Tanto para React como para React Native es recomendable el uso de *frameworks* como Next.js, Gastby o Remix que son herramientas que proporcionan un entorno preparado con todas las herramientas para el desarrollo, dependiendo del tipo de desarrollo recomiendan uno diferente, algunos ejemplos pueden ser Next.js para sitios web y proyectos que emplean el uso de SSR (Server-Side Rendering) y Gastby para sitios web estáticos [24].

Existen también herramientas como *Create React App* (CRA) y el equivalente *Create React Native App* (CRNA), estas son herramientas diseñadas para simplificar el proceso de inicio y desarrollo de aplicaciones web y móviles [25]. Sin embargo, en la nueva documentación de React y React Native se ha dejado de recomendar su uso en proyectos reales y solo es recomendable en proyectos para el aprendizaje, ya que existen alternativas como los *frameworks* nombrados que ofrecen una experiencia más amplia y mejorada para el desarrollo.

En lugar de estas herramientas, para el desarrollo de MusicApp se ha empleado Expo. Este no es exactamente un framework en el mismo sentido que Gatsby o Next.js. Expo es más bien una plataforma que proporciona un conjunto de herramientas, bibliotecas y servicios para facilitar el desarrollo de aplicaciones móviles utilizando React Native [26].

3.3 Expo

Tal y como dicen en su documentación, *"Expo es una plataforma de código abierto para crear aplicaciones nativas universales para Android, iOS y la web utilizando JavaScript y React."* Cabe destacar que también es compatible con TypeScript. El uso de Expo es muy recomendable y ofrece una gran cantidad de ventajas. La figura 22 muestra una imagen que ilustra expo.

Figura 22 – Expo

Expo[23] nos evita la necesidad de escribir código nativo específico para iOS y Android al proporcionar una capa de abstracción sobre las API nativas del dispositivo. Utilizando JavaScript o TypeScript y React, Expo permite a los desarrolladores crear aplicaciones móviles que funcionan en ambas plataformas de manera uniforme. Esto se logra mediante el uso de bibliotecas y componentes preconstruidos que encapsulan las funcionalidades comunes de iOS y Android, como la cámara, la geolocalización y las notificaciones [26].

EAS (Expo Application Services)

EAS es una plataforma de Expo diseñada para facilitar diversas etapas del ciclo de vida de desarrollo de aplicaciones móviles [27]. Incluye:

- EAS Build: Es un servicio de construcción de aplicaciones móviles que permite compilar y generar archivos binarios para iOS y Android directamente en la nube de Expo. EAS Build optimiza el proceso de compilación al manejar automáticamente dependencias, configuraciones de compilación específicas de plataforma y la firma de aplicaciones.
- EAS Submit: Facilita el proceso de envío de aplicaciones a las tiendas de aplicaciones de iOS y Android. Automatiza tareas como la generación de certificados, la preparación de archivos de subida y otros requisitos específicos de cada tienda de aplicaciones.
- EAS Update: simplifica la gestión de actualizaciones en aplicaciones móviles Expo al permitir a los desarrolladores administrar y asegurar las actualizaciones desde un panel centralizado. Facilita el control sobre la implementación y

[23] **Expo:** https://expo.dev/.

reversión de cambios, integrándose automáticamente para mantener la coherencia con las versiones más recientes de las aplicaciones.

¿Expo Sirve para Desarrollo Web?

Expo está diseñado principalmente para el desarrollo de aplicaciones móviles utilizando React Native. Sin embargo, Expo también proporciona herramientas y servicios para el desarrollo de aplicaciones web. A través de bibliotecas como *expo-web-browser* y *expo-constants*, Expo facilita la integración de funcionalidades web comunes en aplicaciones desarrolladas con Expo, como el acceso al navegador y la obtención de información del dispositivo [28].

Aunque Expo puede ser utilizado para desarrollar aplicaciones web, su enfoque principal y sus características más avanzadas están orientadas hacia el desarrollo de aplicaciones móviles nativas para iOS y Android. Para proyectos web más complejos o específicos, es posible que otras herramientas y frameworks como los nombrados anteriormente Next.js o Gastby sean más adecuados debido a su enfoque específico en el desarrollo web y sus capacidades adicionales como el SSR (*Server-Side Rendering*) y la generación estática de sitios web.

¿Por qué Expo?

Se eligió Expo porque se tenía conocimiento previo en esta tecnología. Esto facilitó su adopción, ya que se podía aprovechar la experiencia existente con React y React Native. Además, Expo soporta TypeScript, lo cual era un punto importante, ya que TypeScript ha ayudado a mantener el código más seguro y organizado. Expo también ofrece un conjunto potente de herramientas que facilitan el desarrollo, junto con una documentación completa y una comunidad activa, lo que garantiza gran cantidad de documentación.

Expo permite desarrollar tanto aplicaciones móviles para iOS y Android como aplicaciones web a partir de un único proyecto en JavaScript o TypeScript, esto era uno de los objetivos planteados para este proyecto y de esta manera se cubre fácilmente, evitando la necesidad de mantener varios desarrollos.

3.4 Jest 

Jest[24] es un marco de pruebas de JavaScript desarrollado y mantenido por Facebook. Está diseñado para trabajar con proyectos de JavaScript, especialmente aquellos que utilizan React, pero puede ser utilizado en cualquier aplicación JavaScript. Jest proporciona un entorno completo para realizar pruebas unitarias, de integración y de extremo a extremo, facilitando la creación de código de alta calidad y sin errores. También funciona con proyectos que utilicen: TypeScript, Node, React, Angular, Vue y más.

```
PS C:\Users\Javi\Desktop\JAVI\repos\MusicAppTS> npm run test

> musicappts@1.0.0 test
> jest --coverage

 PASS  __test__/src/utils/StringUtils.test.ts
 PASS  __test__/src/utils/globals.test.ts
 PASS  __test__/src/utils/formValidations.test.ts
--------------------|---------|----------|---------|---------|-------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
All files           |     100 |      100 |     100 |     100 |
 StringUtils.ts     |     100 |      100 |     100 |     100 |
 formValidations.ts |     100 |      100 |     100 |     100 |
 globals.ts         |     100 |      100 |     100 |     100 |
--------------------|---------|----------|---------|---------|-------------------

Test Suites: 3 passed, 3 total
Tests:       34 passed, 34 total
Snapshots:   0 total
Time:        2.791 s, estimated 4 s
Ran all test suites.
```

Figura 23 - Pruebas Jest con coverage

Principales características y herramientas de Jest

- Simplicidad: Jest es muy fácil de configurar y usar, con una sintaxis clara y sencilla. Funciona bien sin necesidad de mucha configuración adicional.

- Velocidad: Ejecuta las pruebas en paralelo, mejorando el rendimiento, lo cual es especialmente útil en proyectos grandes. Al ejecutar los test se puede observar cómo comienza a ejecutar todas las suites al mismo tiempo, además se completan todos test actuales en menos de 3 segundos.

- Cobertura de Código: Incluye informes de cobertura de código integrados, ayudando a los desarrolladores a identificar qué partes del código no están siendo probadas. En la figura 23 puede observarse como después de ejecutar todas las pruebas hay cobertura del 100 para los archivos que están siendo testeados.

[24] **Jest:** https://jestjs.io/es-ES/.

Organización de Pruebas en Jest

Cuando se escriben pruebas en Jest, es común estructurarlas en bloques lógicos utilizando describe, beforeEach, y afterEach. Estas herramientas ayudan a organizar las pruebas, configurar el estado inicial, y limpiar después de las pruebas. La estructura general de un archivo de pruebas suele incluir:

- *describe*: Agrupa pruebas relacionadas en un bloque lógico. Puedes anidar múltiples bloques describe para organizar las pruebas jerárquicamente.
- *beforeEach*: Ejecuta una función antes de cada prueba en el bloque describe. Es útil para configurar el estado inicial.
- *afterEach*: Ejecuta una función después de cada prueba en el bloque describe. Es útil para limpiar el estado después de cada prueba.

En la figura 24 se puede observar la estructura que se ha empleado durante el desarrollo, organizando todos los test del archivo en un *describe* general y dentro de este, otros secundarios con tests relacionado.

```
describe('formValidations', () => {
  describe('validateEmail', () => {
    it('should return true for a valid email', () => {
      expect(validateEmail('test@example.com')).toBe(true)
    })

    it('should return false for an invalid email', () => {
      expect(validateEmail('test@example')).toBe(false)
    })
  })
  ...
})
```

Figura 24 - Ejemplo de test de MusicApp con Jest

Moks y SpyOn

Un mock es una función simulada que imita el comportamiento de una función real. Los mocks permiten probar cómo interactúa una función con otras funciones o módulos sin ejecutar el código real de esas dependencias. Esto es útil para aislar la función que se está probando y simular diferentes comportamientos de dependencias externas. También puede cambiar la implementación de la función que se *mockea* con las funciones *mockReturnValue o mockReturnValueOnce*.

Un spy en cambio permite rastrear las llamadas a una función existente sin alterar su implementación. Mediante el uso de Mocks y Spies, se puede verificar que una función se haya llamado correctamente, con los argumentos esperados y cuántas veces se ha llamado. Toda la información sobre esta herramienta ha sido extraída de la documentación de esta, se puede encontrar en [29]

```
  beforeEach(() => {
    jest.clearAllMocks() // Limpia todos los mocks antes de cada prueba
  })
  it('should mock validateEmail function', () => {
    const mockValidateEmail = jest.fn()
    const result = mockValidateEmail('mock@example.com')

    expect(result).toBe(true)
    expect(mockValidateEmail).toHaveBeenCalled()
    expect(mockValidateEmail).toHaveBeenCalledWith('mock@example.com')
  })
  it('should spy on validateEmail and check it is called once', () => {
    const spyValidateEmail = jest.spyOn(
      require('../../../src/utils/formValidations'),
     'validateEmail')
    const result = validateEmail('spy@example.com')

    expect(result).toBe(true)
    expect(spyValidateEmail).toHaveBeenCalledTimes(1)
    expect(spyValidateEmail).toHaveBeenCalledWith('spy@example.com')
  })
})
```

Figura 25 - Ejemplo uso de Mocks y SpyOn en tests de Jest

En la figura 25 se ve la utilidad de los mocks que actúan aislando al componente que se quiere probar de los demás, para centrarse únicamente en su comportamiento. En la figura también se presenta el empleo de la sentencia *beforeEach*, para reiniciar los mock antes de pasar los tests, se podría hacer de la misma manera con un *afterEach*.

3.5 Visual Studio Code y extensiones 

Visual Studio Code[25] es el editor de código que se ha empleado para el desarrollo de MusicApp, por su sencillez el conocimiento previo de esta herramienta. Algunas de las extensiones de VS Code que se han utilizado se comentan acontinuación.

Jest Runner

Jest Runner[26] es una extensión diseñada específicamente para proyectos que utilizan Jest como framework de pruebas, como es el caso. Facilita la ejecución y la visualización de los resultados de las pruebas directamente desde la consola de VS Code, lo que mejora la eficiencia al evitar tener que cambiar constantemente entre la terminal y el editor.

Prettier y ESLint

Prettier[27] es un formateador de código que ayuda a mantener un estilo de código consistente y legible. Es compatible con múltiples lenguajes y frameworks, y se integra perfectamente con VS Code. Fui fácil de configurar, se establecían unas normas que este debe seguir a la hora de formatear, también se pueden excluir archivos por extensión o nombre, e incluye una función para que formatee automáticamente el código de un archivo cada vez que se guarda este, lo que es realmente cómodo para mantener la legibilidad y consistencia del código.

ESLint[28] es una herramienta de *linting*[29] para JavaScript que ayuda a mantener un código consistente y libre de errores. Cuando se integra con VS Code, proporciona advertencias y errores en tiempo real directamente en el editor, lo que facilita la corrección y la adhesión a las mejores prácticas de codificación.

Se integra con VS Code mediante una extensión que analiza el código mientras escribes. Detecta patrones problemáticos o errores potenciales basados en reglas preconfiguradas y muestra advertencias o errores en el editor.

El uso combinado de ambas herramientas al principio genera problemas pero después de una correcta configuración, facilita mucho el desarrollo y el flujo de trabajo. La información empleada para configurar tanto Eslint como Prettier se encuentra en [30] y [31] respectivamente.

[25] **Visual Studio Code:** https://code.visualstudio.com/.
[26] **Jest Runner:** https://www.npmjs.com/package/jest-runner.
[27] **Prettier:** https://prettier.io/.
[28] **Eslint:** https://eslint.org/.
[29] **Linting:** proceso de analizar el código fuente para identificar posibles errores.

CodeSnap

Code Snap[30] es una extensión que permite realizar capturas de pantalla del código con formato personalizable. Esto es útil para compartir ejemplos de una manera visualmente atractiva y coherente. Para todas las imágenes de este TFG que contienen código de MusicApp se ha empleado esta herramienta.

3.6 Expo Go 

Es una aplicación móvil desarrollada por Expo que permite a los desarrolladores y usuarios probar aplicaciones creadas con Expo directamente en dispositivos móviles iOS y Android o en emuladores. Expo Go[31] actúa como un entorno de ejecución para las aplicaciones Expo, proporcionando una forma conveniente y rápida de visualizar y probar el funcionamiento de las aplicaciones en un entorno real [32].

Expo Go es especialmente útil durante el desarrollo y la fase de pruebas de una aplicación móvil, ya que elimina la necesidad de compilar y desplegar repetidamente en dispositivos físicos. Permite visualizar cambios en tiempo real y obtener feedback inmediato sobre la experiencia de usuario en dispositivos reales.

Para empezar a usar Expo Go, simplemente hay que instalar la aplicación desde la App Store (para iOS) o Google Play Store (para Android), y luego escanear un código QR generado desde su proyecto Expo en el navegador web. Esto carga automáticamente la aplicación en Expo Go para pruebas inmediatas.

Para no depender siempre de dispositivos reales, también se ha optado por emplear emuladores como el de Android Studio, Actualmente se emplea un script bat el contenido del cual puede verse en la figura 26.

```
1  @echo off
2  emulator -avd Pixel5
3  pause
```

Figura 26 - Script Emulador

[30] **CodeSnap:** https://codesnap.dev/ .
[31] **Expo Go:** https://expo.dev/go.

3.7 Sentry

Sentry[32] es una plataforma de monitoreo de errores diseñada para ayudar a los equipos de desarrollo a detectar, diagnosticar y solucionar problemas en tiempo real en sus aplicaciones. Su función principal es capturar y reportar errores y excepciones que ocurren en el código, lo que permite identificar rápidamente las causas y tomar medidas correctivas [33]. Sentry opera en tiempo real, alerta con notificaciones y correos en el momento en el que ocurre un error o excepción, mediante un *dashboard* se pueden ver de manera clara cuando y que ha fallado. Es una herramienta muy útil para la identificación de errores no controlados, al trabajar en el desarrollo de una aplicación móvil esto puede llegar a ser muy desesperante, y Sentry ayuda mucho en estos casos.

Sentry es un servicio que presenta planes de suscripción, entre ellos existe un plan gratuito con limitaciones, actualmente el servicio gratuito que ofrecen cubre las necesidades del software que se está desarrollando, por lo que no se ha contemplado para la proyección económica como un gasto, pero puede que en un futuro pueda llegar a ser interesante contratar uno de estos planes de pago. En la figura 27 pueden verse los planes de Sentry.

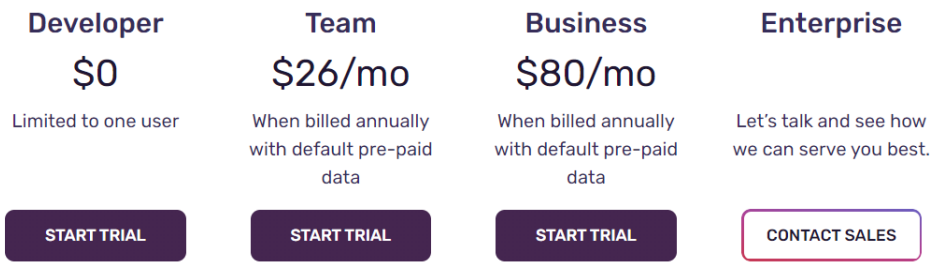

Figura 27 - Pricing Sentry

3.8 I18next

I18next[33] es una biblioteca o librería de internacionalización (i18n) escrita en JavaScript. Esta librería facilita la tarea de integrar el multilenguaje en aplicaciones web y otras aplicaciones que se ejecutan en entornos basados en JavaScript. Ofrece funcionalidades avanzadas para la gestión de textos traducidos, soporte para plurales, entre otros. Proporciona métodos y utilidades para separar y gestionar eficientemente los textos traducibles de las aplicaciones, permitiendo una gestión más estructurada y mantenible de las traducciones [34]. Un ejemplo de empleo se esta herramienta puede observarse en el siguiente código de la figura 28.

[32] **Sentry:** https://sentry.io/.
[33] **I18next:** https://www.i18next.com/.

```
const { t } = useTranslation()
...
  <OptionsIcon
    options={[
      {
        label: t('Delete'),
        icon: require('../../../../assets//icon/papelera.png'),
        onPress: () => {
          setModalDeleteVisible(true)
        },
      },
    ]}
  />
```

Figura 28 - Ejemplo de uso de la librería i18next

3.9 Supabase

Supabase es una herramienta poderosa y versátil para el desarrollo de aplicaciones, está diseñada para facilitar el proceso de construcción de backend robustos y escalables. *Es* un Software as a Service (SaaS) ofrece servicios de base de datos y backend como servicio, permitiendo crear aplicaciones sin preocuparse por la infraestructura subyacente. Algunas de sus funciones integradas incluyen la gestión de la base de datos PostgreSQL, la creación de *API RESTful* y en tiempo real, la autenticación de usuarios, Supabase ofrece interfaces con las que podremos configurar de manera sencilla estos servicios [35].

Base de datos PostgresSQL

Supabase utiliza PostgreSQL como su sistema de gestión de bases de datos relacional (SGBDR). PostgreSQL es conocido por su robustez, capacidad para manejar grandes volúmenes de datos, soporte para consultas complejas y extensiones avanzadas como JSONB. Supabase ofrece una instancia gestionada de PostgreSQL que permite a los desarrolladores almacenar y manipular datos de manera eficiente y segura [36].

Autentication

Supabase proporciona un sistema integrado de autenticación que permite a los desarrolladores gestionar la identidad y el acceso de los usuarios a sus aplicaciones de manera segura. Soporta métodos de autenticación como correo electrónico/contraseña, autenticación social (como Google, GitHub, etc.), y tokens JWT (JSON Web Tokens). Esto facilita la implementación de funciones de registro, inicio de sesión y gestión de usuarios de la plataforma que se está desarrollando.

Storage

Supabase ofrece un servicio de almacenamiento que permite manejar archivos estáticos y multimedia en sus aplicaciones. Utiliza almacenamiento en la nube para alojar y servir archivos de forma escalable y segura. Es posible cargar, descargar y gestionar archivos directamente desde sus aplicaciones utilizando la API de almacenamiento de Supabase.

Real time

Supabase utiliza *WebSockets* para establecer conexiones persistentes entre el cliente y el servidor, permitiendo notificaciones instantáneas, actualizaciones en tiempo real de datos y colaboración en tiempo real en aplicaciones colaborativas como chats, lo que nos es muy útil para MusicApp.

RLS (Row-Level Security)

Las políticas de seguridad a nivel de fila o *Row-Level Security* son una característica clave de PostgreSQL que proporciona un nivel adicional de control de acceso a los datos a nivel de fila en una base de datos. Como ya hemos comentado Supabase presenta un servicio de autenticación, que se ocupa de verificar la identidad de un usuario (quién es), mientras que la autorización se ocupa de qué acciones puede realizar ese usuario en los datos (qué puede hacer), esta autorización se logra con el uso de políticas RLS.

RLS permite a los administradores de bases de datos definir políticas de seguridad en las tablas, lo que determina qué filas de datos son visibles y/o modificables para los usuarios en función de sus roles o condiciones específicas. Esto es especialmente útil en entornos multiusuario donde diferentes roles necesitan diferentes niveles de acceso a los datos. Y precisamente para MusicApp se van a estar tratando 3 roles, el de Administrador de centro, el de los profesores y los usuarios comunes.

Antes de seguir con las RLS es necesario comentar lo que son las funciones en Postgres. Son bloques de código que se pueden definir y ejecutar dentro de la base de datos para realizar tareas específicas. Pueden aceptar parámetros de entrada y opcionalmente devolver un valor como resultado. Permiten encapsular lógica compleja, mejorar la reutilización del código y optimizar el rendimiento al realizar operaciones que no pueden ser manejadas eficientemente solo con consultas SQL estándar. Además, son fundamentales para definir políticas RLS, automatizar tareas y realizar cálculos complejos directamente dentro de la base de datos.

Imaginamos ahora una tabla llamada *profiles* la cual contiene los campos id y nombre. Además, supondremos 2 tipos de usuarios, los administradores y los usuarios comunes. Por último, también se cuenta con la función Postgres isAdmin(). Un ejemplo de política RLS podría ser el siguiente:

- Solo los administradores puedan borrar elementos de la tabla profiles.
- Tanto administradores como usuarios comunes pueden insertar perfiles
- Todos puede ver los perfiles independientemente del rol.
- Cada usuario solo puede editar su propio perfil

```sql
CREATE POLICY select_profiles_policy
ON profiles
FOR SELECT
TO PUBLIC
USING (true);

-- Política de RLS para INSERT
CREATE POLICY insert_profiles_policy
ON profiles
FOR INSERT
TO PUBLIC
USING (true);

-- Política de RLS para UPDATE
CREATE POLICY update_profiles_policy
ON profiles
FOR UPDATE
TO PUBLIC
USING (profile.id = Auth.uid());

-- Política de RLS para DELETE
CREATE POLICY delete_profiles_policy
ON profiles
FOR DELETE
TO PUBLIC
USING (
    CASE
        WHEN (SELECT isAdmin()) THEN true
        ELSE false
    END
);
```

Figura 29 - Ejemplo Código RLS

En la figura 29 podemos apreciar una posible solución para el problema planteado, gracias a la función isAdmin comprobamos el rol del usuario y de esta manera restringimos ciertas operaciones CRUD para tipos concretos de usuarios. Estas políticas quedan almacenadas en las tablas y pueden editarse posteriormente fácilmente gracias a la interfaz que proporciona Supabase.

Para lograr el último requisito "Cada usuario solo puede editar su propio perfil" se necesita emplear el servicio de autenticación de Supabase nombrado antes, se consigue accediendo a la tabla Auth, gestionada por Supabase, para poder determinar quién es quién entre los usuarios que interactúen con la plataforma.

Ventajas y Usos Prácticos de Supabase

Al principio, se consideró utilizar Firebase[34], para el backend de mi aplicación debido a su popularidad y sus amplias capacidades, incluyendo autenticación, base de datos en tiempo real y almacenamiento. Firebase es una solución completa que facilita la implementación rápida de funcionalidades esenciales, lo que lo convierte en una opción atractiva para muchos desarrolladores. Sin embargo, después de investigar más a fondo, encontramos Supabase, una alternativa *open-source* que ofrece una experiencia similar a Firebase, pero con algunas ventajas significativas.

Supabase utiliza PostgreSQL como su base de datos principal, lo que proporciona una mayor flexibilidad y poder en las consultas de datos. A diferencia de Firebase, que utiliza su propia base de datos NoSQL, Supabase permite aprovechar las características avanzadas de SQL y las políticas RLS para un control de acceso más granular. Además, Supabase proporciona una integración más sencilla con otros servicios y una mayor transparencia en la gestión de la infraestructura, lo que nos llevó a decantarnos por esta plataforma.

Elegir una arquitectura serverless con Supabase también ofrece numerosas ventajas. La administración de la infraestructura está completamente gestionada, lo que elimina la necesidad de preocuparse por el mantenimiento de servidores y permite escalar automáticamente según la demanda. Esto no solo reduce los costos operativos, sino que también mejora la eficiencia y la rapidez del desarrollo, permitiéndome centrarme más en el desarrollo de funcionalidades clave y la mejora de la experiencia del usuario.

[34] **Firebase:** https://firebase.google.com/?hl=es.

3.10 Trello

Trello[35] es una herramienta de gestión de proyectos basada en tableros que facilita la organización y colaboración entre equipos. Funciona utilizando un sistema de tableros, listas y tarjetas que representan tareas y actividades [37].

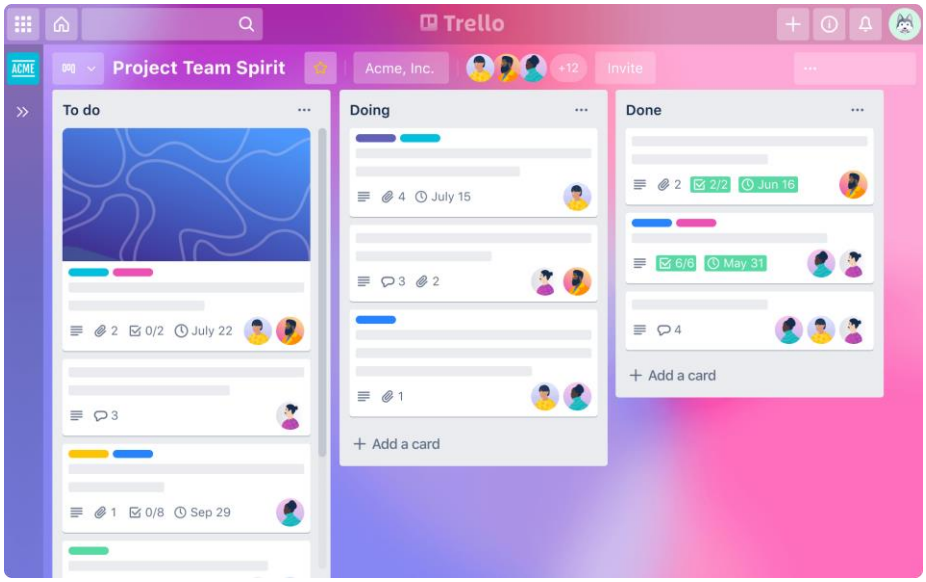

Figura 30 - Trello

Un tablero en Trello representa un proyecto o proceso de trabajo. Se pueden crear múltiples tableros para diferentes proyectos o áreas de trabajo. Dentro de cada tablero, se pueden crear listas. Las listas son columnas verticales que representan diferentes etapas de tu flujo de trabajo, como "Pendiente", "En Progreso" y "Completado".

Cada tarea o actividad se representa mediante una tarjeta que se coloca en una lista. Las tarjetas contienen detalles sobre la tarea, como descripción, fecha de vencimiento, archivos adjuntos, etiquetas y comentarios.

Es posible mover las tarjetas entre listas para reflejar el progreso de la tarea. Por ejemplo, una tarjeta puede comenzar en la lista "Pendiente", luego pasar a "En Progreso" y finalmente a "Completado" una vez finalizada.

Trello es una herramienta que facilita la colaboración entre equipos. Las tareas pueden asignarse a los distintos miembros de un equipo y es posible agregar comentarios, adjuntar archivos y recibir notificaciones sobre actualizaciones y cambios en las tarjetas.

[35] **Trello:** https://trello.com/es.

CAPÍTULO 4

Desarrollo de la idea de negocio

En este capítulo, se detallan las características clave del desarrollo, como se ha estructurado este y la manera de trabajar que se ha seguido.

4.1 Metodología

La metodología que se ha empleado en este proyecto es Lean Startup [38] en combinación con diversas prácticas ágiles. La metodología Lean Startup se ha adoptado ampliamente para el desarrollo de negocios y productos, con el objetivo de acortar los ciclos de desarrollo y maximizar el aprendizaje. Se centra en la creación rápida de un producto mínimo viable (MVP) y la iteración continua en función de la retroalimentación de los usuarios. Su objetivo es reducir el riesgo y aumentar las probabilidades de éxito al validar las ideas de negocio de manera eficiente. La metodología fue popularizada por Eric Ries en su libro The Lean Startup [39].

Principios Clave de Lean Startup

1. Producto Mínimo Viable o *Minimum viable product* (PMV o MVP): es la versión más simple de un producto que permite a un equipo recoger la máxima cantidad de conocimiento validado sobre los clientes con el mínimo esfuerzo.

2. Ciclo Crear-Medir-Aprender: Este ciclo es esencial en Lean Startup. Implica construir un MVP, medir la respuesta del mercado y aprender de los resultados para iterar y mejorar el producto. Es un proceso continuo de transformación de ideas en productos, medición de la reacción del cliente y aprendizaje para decidir si perseverar o pivotar.

3. Iteración Continua: Lean Startup promueve la iteración basada en la retroalimentación del cliente. Las empresas deben estar dispuestas a modificar su producto y cambiar de dirección (pivotar) cuando sea necesario para ajustarse mejor a las necesidades del mercado.

4. Aprendizaje Validado: Utiliza datos y experimentos reales para confirmar suposiciones sobre el producto y el mercado. Este enfoque científico ayuda a evitar el desperdicio de recursos en características que los clientes no desean.

Desarrollo ágil

El desarrollo ágil consiste en un conjunto de prácticas a seguir a la hora de gestionar proyectos de software que se enfoca en la entrega incremental y la flexibilidad para adaptarse a los cambios. Nació como respuesta a las limitaciones de los enfoques tradicionales, como el modelo en cascada, que a menudo resultaban rígidos e ineficaces frente a los requisitos cambiantes de los clientes. El desarrollo ágil se basa en los siguientes valores, basados en los 12 principios del Manifiesto Ágil [40]:

1. Individuos e interacciones sobre procesos y herramientas.
2. Software funcionando sobre documentación extensiva.
3. Colaboración con el cliente sobre negociación contractual.
4. Respuesta ante el cambio sobre seguir un plan.

Las prácticas agiles que se han seguido para el desarrollo del proyecto y que junto con el modelo Lean Startup conforman la metodología empleada se detallan a continuación:

Sprints

Los *sprints* son la piedra angular del desarrollo ágil, comúnmente se ha asociado a la metodología Scrum pero hoy en día se ha generalizado el término sprint en el marco de las metodologías agiles. Un *sprint* es un ciclo de desarrollo corto y recurrente, generalmente de una a cuatro semanas, durante el cual el equipo de desarrollo trabaja para completar una cantidad específica de trabajo.

Antes de comenzar un sprint, se realiza una reunión de planificación donde el equipo selecciona los elementos del backlog que se van a trabajar y establece un objetivo claro para el sprint. Durante el sprint, los miembros del equipo desarrollan las funcionalidades, realizan pruebas y trabajan juntos para completar las tareas asignadas. Al final del sprint, el equipo presenta el trabajo completado al Product Owner y otros interesados para obtener feedback. También es común realizar una reunión de retrospectiva, donde el equipo se reúne para discutir lo que funcionó bien, lo que no y cómo pueden mejorar en futuros sprints.

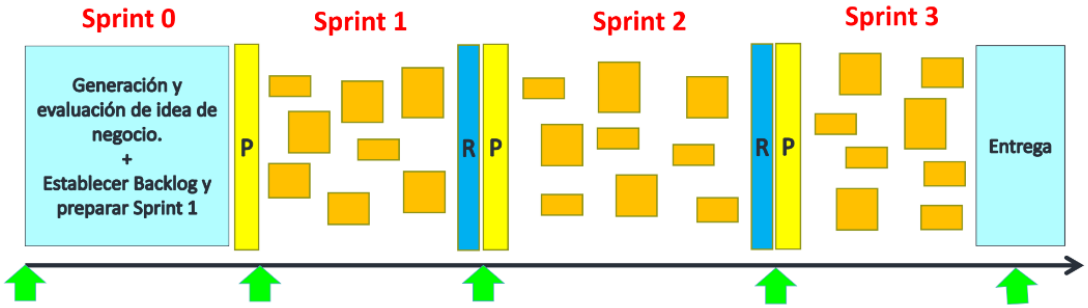

Figura 31 - Representación de un desarrollo con Sprints

Backlog

El *Product Backlog* es una lista ordenada por orden de prioridad de todas las funcionalidades, mejoras, tareas y correcciones necesarias para el producto. Este backlog es dinámico y puede evolucionar a medida que se descubren nuevas necesidades y se recibe feedback.

Cada elemento del backlog es conocido como "historia de usuario" o "item de backlog" también hay herramientas donde se denomina “Unidad de trabajo (UT)”, estos describen una funcionalidad o tarea específica que agrega valor al producto. El orden de priorización que se realiza es en función del valor que aportan al negocio y a los usuarios. Este backlog también recibe un refinamiento durante el cual el equipo revisa y detalla los elementos para asegurar que estén listos para ser trabajados en futuros sprints.

En la figura 32 vemos que la forma con la que se suele representar el Backlog es un triángulo, esta forma es debido a la priorización de las tareas o elementos que lo componen. Esta representación gráfica ayuda a entender cómo se gestionan las diferentes actividades o requisitos del proyecto. En la parte superior encontramos lo elementos con mayor prioridad, los que se deberán desarrollar primero, esto sugiere que las tareas se irán desplazando de abajo hacia arriba en el Backlog.

Figura 32 - Representación proceso de creación y refinamiento del Backlog

Roles comúnmente empleados

- Product Owner (PO): Es responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo. El PO gestiona el Product Backlog, prioriza las tareas y asegura que el equipo entienda claramente los elementos del backlog y los objetivos del sprint.

- Scrum Master: Facilita el proceso Scrum, asegurándose de que el equipo siga las prácticas y principios ágiles. El Scrum Master elimina obstáculos que impiden el progreso del equipo y fomenta la mejora continua.

- Equipo de Desarrollo: Compuesto por profesionales multifuncionales que trabajan juntos para entregar incrementos de producto potencialmente liberables al final de cada sprint.

Incremento

Al final de cada sprint, se debe entregar un incremento de producto potencialmente liberable. Este incremento debe estar completamente desarrollado, probado y listo para ser utilizado. En nuestro caso los incrementos se corresponderán con los MVP.

Ventajas del Desarrollo Ágil

- Flexibilidad: El desarrollo ágil permite adaptarse rápidamente a los cambios en los requisitos del proyecto.
- Entrega Incremental: Las funcionalidades se entregan en pequeñas iteraciones, lo que permite obtener feedback temprano y frecuente.
- Colaboración: Fomenta una estrecha colaboración entre el equipo de desarrollo y los interesados.
- Transparencia: La transparencia en el progreso del proyecto ayuda a mantener a todos los interesados informados y alineados con los objetivos del proyecto.
- Mejora Continua: Las retrospectivas y la cultura de feedback continuo ayudan a mejorar los procesos y la calidad del producto.

Proyectos desarrollados anteriormente

Durante el grado, se ha empleado tanto las metodologías de desarrollo ágil como Lean Startup en proyectos de asignaturas como PIN (Proyecto de Ingeniería de Software) y PSW (Proceso de Software). Lo que ha permitido trabajar con ella y comprenderla mejor.

El proyecto desarrollado en PIN fue una aplicación móvil que conformaba una guia interactiva para el cuidado de plantas en el hogar. El desarrollo de este proyecto se estructuró en 3 sprints, además de un sprint 0 inicial. En la figura 31 puede observarse su distribución. Para este proyecto se conformaron grupos multidisciplinarios con estudiantes de Diseño, quienes participaron durante todo el desarrollo e hicieron un gran trabajo diseñando las interfaces de la aplicación. El proyecto fue desarrollado durante el segundo cuatrimestre del tercer año del grado y el resultado se presentó en la 6ª edición de la feria de proyectos de la ETSINF. Aunque la aplicación fue bien recibida y elogiada, después de este lanzamiento se decidió no continuar el proyecto.

Las figuras 31 y 32 se han obtenido del material de la asignatura de PIN.

4.2 Requisitos

En esta sección se verán los requisitos funcionales y no funcionales de MusicApp.

Requisitos funcionales

Se realiza un Diagrama de Casos de Uso con Lucidchat[36] para poder representar de manera clara las funcionalidades de MusicApp. En la figura 35 vemos este diagrama.

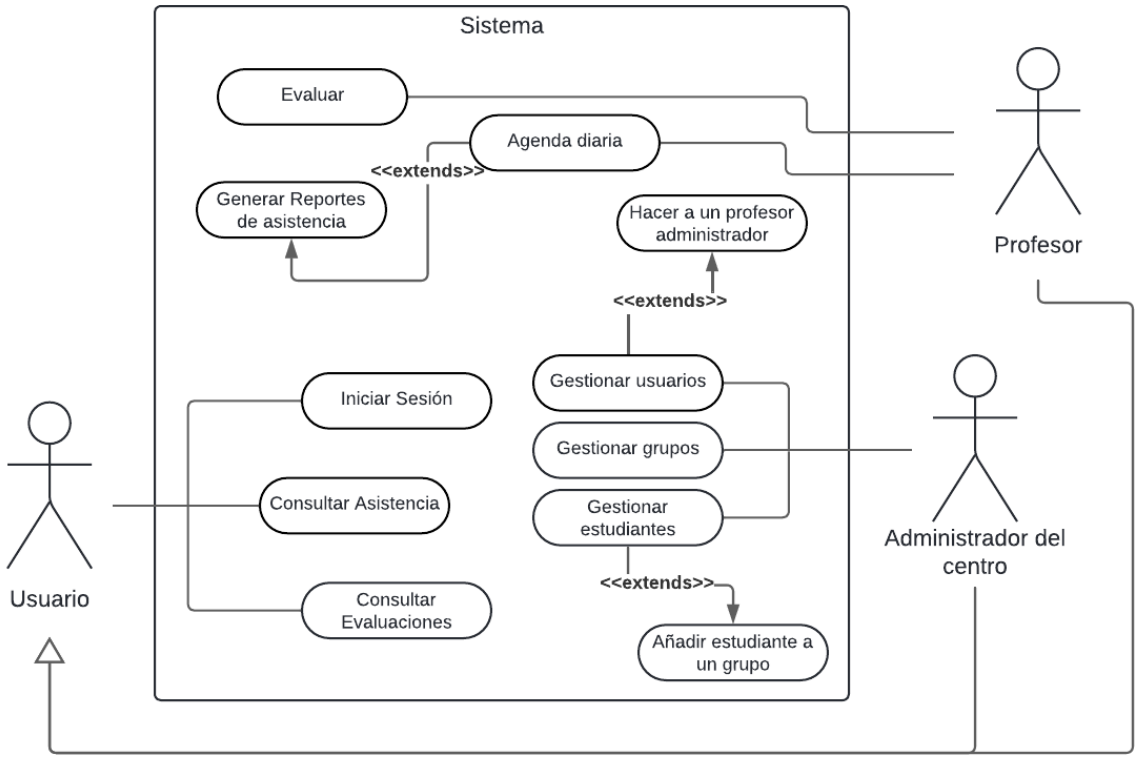

Figura 33 -Diagrama de Casos de Uso de MusicApp

Al entrar en la plataforma, todo usuario debe iniciar sesión, después de esto el usuarios ya puede ser identificado, al igual que su rol, con el que se va a representar el tipo de usuario. Dependiendo del rol del usuario, la plataforma brinda un grupo diferente de funcionalidades.

En la figura 35 podemos apreciar los 3 tipos de usuarios que interactuarán con el sistema, el primero de ellos, corresponde a los padres o personas a cargo de los estudiantes, estos usuarios solo podrán consultar información sobre los estudiantes relacionados a dichos perfiles, esta información incluye los informes de asistencia, las evaluaciones, los horarios y los eventos de estos alumnos. También podrán enviar mensajes a través de los chats de los grupos en los que estén los estudiantes. podrían ser los propios estudiantes quienes manejasen sus propios perfiles.

[36] **Lucidchart:** https://www.lucidchart.com/pages/es .

El segundo tipo de usuarios que se presenta son los profesores, quienes además de poder consultar la información podrán gestionar informes de asistencia, evaluaciones y criterios de evaluación, pudiendo además crear, editar y borrar. Por último, se nos muestra a los Administradores del centro quienes además podrán gestionar los grupos, las asignaturas y los usuarios, además de los anuncios generales del centro. Al igual que los profesores, tendrían todos los privilegios sobre esta información.

Es razonable que tanto usuarios comunes, como profesores y administradores, solo podrán acceder a la información de su propio centro. De esta manera se mantiene asegurada la información privada de cada centro.

Todos los usuarios pueden consultar los reportes de asistencia y las evaluaciones de los estudiantes, los administradores pueden actuar sobre todos los estudiantes de la escuela, los profesores sobre los estudiantes de sus grupos y los demás usuarios solo pueden acceder a los estudiantes a su cargo.

Los profesores son los encargan de evaluar a los estudiantes, creando informes de evaluación, solo pueden evaluar a los estudiantes de sus grupos, además desde la Agenda diaria pueden pasar lista durante las clases para generar los correspondiente reportes de asistencia en el caso de que haya alguna falta o retraso.

Los administradores son quienes gestionan a todos los usuarios del centro, son quienes los registran, pueden crear otros administradores convirtiendo a un profesor en profesor administrador, un administrador también puede actuar como profesor si tiene grupos a su cargo. También generan los grupos y deciden que profesor es el encargado de ese grupo. Por último son responsables de la gestión de estudiantes del centro y de a que grupos pertenecen.

Requisitos no funcionales

Para la evaluación de los requisitos no funcionales de mi plataforma, siguiendo el estándar ISO/IEC 25000[37].

Disponibilidad

La plataforma debe estar disponible y accesible para los usuarios en cualquier momento. Al emplear Supabase para el backend, nos apoyamos de manera indirecta en la disponibilidad de este servicio. En status.supabase[38] se puede consultar esta disponibilidad. Podemos observar que desde octubre de 2023 no se han presentado caídas del servicio por lo que se garantiza una disponibilidad cercana al 100%, adjuntan que, de haber interrupciones, estas tendrán siempre un tiempo inferior a 15 minutos.

Para monitorizar la aplicación y detectar cualquier problema, emplearemos Sentry. Como se ha comentado previamente en el capítulo 3, esta herramienta nos permitiría identificar y resolver fallos de manera rápida y eficiente, minimizando el impacto en los usuarios.

[37] ISO 25000: Sistemas e ingeniería de software — Requisitos de Calidad y Evaluación de Sistemas y Software (SQuaRE) — Guía para SQuaRE[42].
[38] Status.supabase: https://status.supabase.com/.

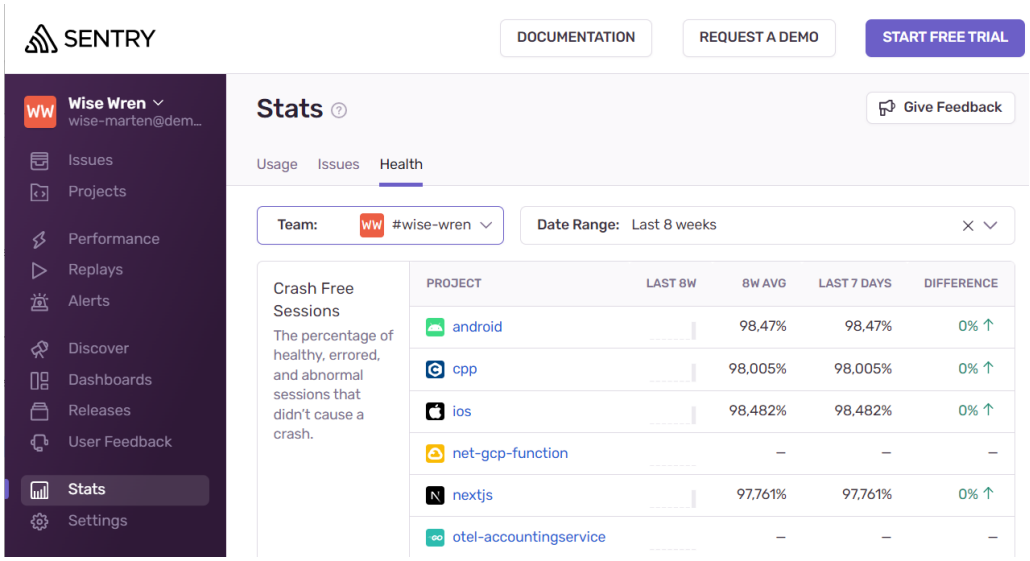

Figura 34 comprobación de la disponibilidad en la demo de Sentry

En la figura 36 podemos ver como se haría esta comprobación de la disponibilidad, en el apartado *Stats* y dentro del subapartado *Heath*, podremos ver el promedio de la disponibilidad de nuestro servicio, pudiendo realizar esta representación tomando información desde 2 hasta 12 semanas. En caso de mantenimiento programado, siempre se informaría a los usuarios con anticipación para evitar interrupciones inesperadas.

MusicApp respaldada por la robustez de Supabase y la monitorización con Sentry, ofrecería una alta disponibilidad, asegurando que tanto administradores, profesores como alumnos puedan acceder a ella prácticamente en cualquier momento.

Mantenibilidad

La mantenibilidad se refiere a la facilidad con la que una aplicación puede ser modificada, ya sea para corregir errores, mejorar su rendimiento o agregar nuevas funcionalidades. Permitiendo así a la aplicación evolucionar y adaptarse a cambios sin contar con altos costos de desarrollo [41].

Para garantizar esta mantenibilidad de nuestra aplicación, se desarrolla MusicApp siguiendo buenas prácticas de programación, como el uso de nombres descriptivos de variables o el estilo y formato constante gracias al uso del formateador Prettier.

Con la implementación pruebas unitarias y de integración se garantiza que cada parte del código funcione correctamente y que los cambios no introduzcan nuevos errores. Además, se podrían ejecutar estas pruebas en cada commit mediante un sistema de Integración Continua, asegurando que cualquier error sea detectado y corregido rápidamente.

Adaptabilidad

MusicApp ha sido desarrollada con el fin de poder ser empleada tanto en sistemas Android e iOS como en la web, todo ello gracias al empleo de tecnologías como expo y React native.

Escalabilidad

La escalabilidad corresponde a la capacidad del producto para gestionar cargas de trabajo crecientes o decrecientes y adaptar su capacidad dichas variaciones, todo ello sin comprometer al rendimiento. Gracias al uso de Supabase se consigue que nuestro servicio se totalmente escalable ya que la API de Supabase es capaz de manejar gran cantidad de solicitudes y sus servicios escalan de manera automática.

Seguridad

Gracias a la gestión de roles de los usuarios se consigue que la información sea accesible para los usuarios solo si estos están autorizados para interactuar con dicha información. Esto se consigue gracias a Postgres y sus políticas RLS, como se comentó en el capítulo 3, estas políticas nos ayudan a definir que usuarios puede hacer que acciones sobre la información de la base de datos.

El *endpoint* que nos brinda Supabase solo es accesible si las solicitudes presentan unas claves en sus parámetros estas claves nos las da Supabase al crear el proyecto y no forman parte del código de la aplicación, sino que se configuran como variables de entorno tanto en el servicio de EAS de Expo como en Vercel.

La primera pantalla que se muestra al iniciar la aplicación es el inicio de sesión, no existe un registro de usuarios, esta acción es realizada por los administradores de los centros. En el caso de que un usuario con intenciones maliciosas consiguiera las claves necesarias de la API de Supabase, podría crear usuarios, pero seguiría sin poder acceder a la información de ningún centro, al no poder pasar a formar parte de un centro por sí mismo, ya que esta acción también puede ser realizada únicamente por un administrador.

4.3 Diseño

La arquitectura de un sistema juega un rol crucial en el desarrollo de cualquier aplicación. Una buena arquitectura no solo determina la eficiencia y el rendimiento de la aplicación, sino que también es clave para su escalabilidad y mantenimiento a largo plazo. Esta estructura establece cómo interactúan los distintos componentes de la aplicación y de qué manera se gestiona el flujo de datos y la información.

La arquitectura de MusicApp sigue un modelo *serverless,* lo que nos ha facilitado la gestión infraestructura de la aplicación. Supabase nos proporciona una solución del backend. En la figura 37 podemos ver la arquitectura global de la aplicación, la aplicación de Android e iOS y la web se crean como clientes de Supabase de nuestro backend y pueden enviar solicitudes a nuestro servidor. Estas solitudes se manejan de manera adecuada por Supabase, gracias a su servicio *Auth*, se identifica las solicitudes autenticadas y sabe en todo momento que usuarios han realizado esas solicitudes.

Aunque Auth es una herramienta de gestión de usuarios muy potente y que brinda muchas opciones de integración con herramientas de autenticación externas, solamente se ha empleado para la autenticación de usuarios mediante correo electrónico. Dentro de la base de datos, existe el esquema de tablas Auth, es ajeno al que contiene las demás tablas del proyecto y contiene información sobre los usuarios autenticados de la aplicación. Existe un identificador único para los usuarios en esta tabla y mediante este identificador se enlaza esta con una nueva tabla *Profiles* donde se gestionará información añadida de los usuarios.

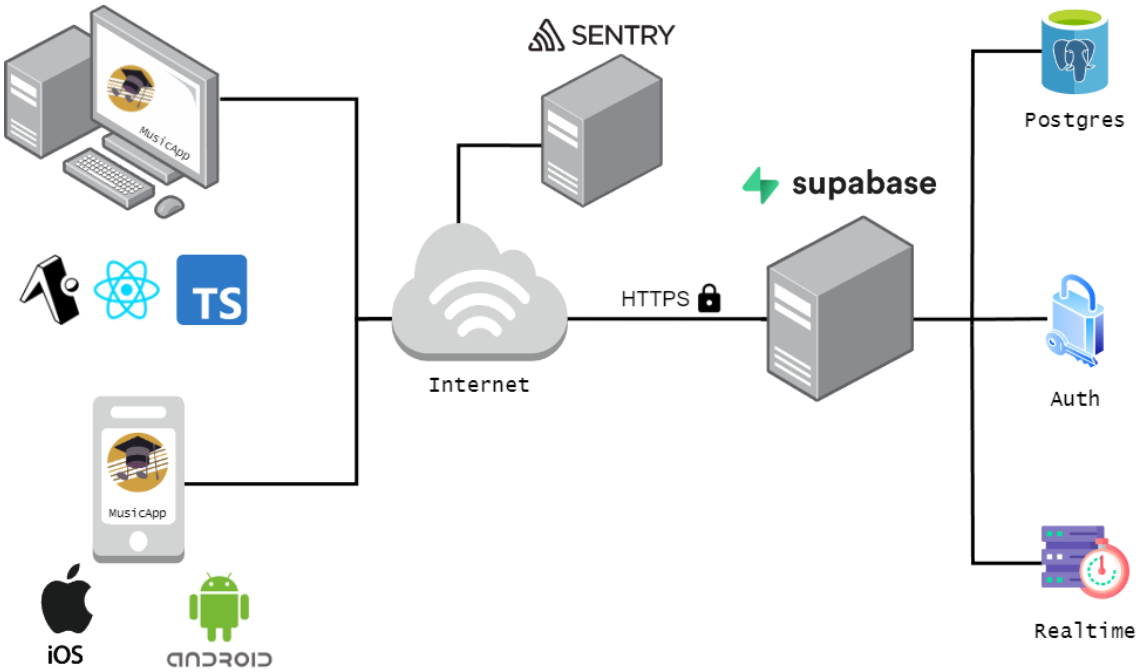

Figura 35 - Arquitectura global de la aplicación

El diagrama de clases es una representación visual que nos va a ayudar a visualizar la estructura del sistema, las interacciones entre los diferentes tipos de componentes y la lógica de negocio que sustenta la aplicación. Podemos ver el diagrama de clases de MusicApp en la figura 38.

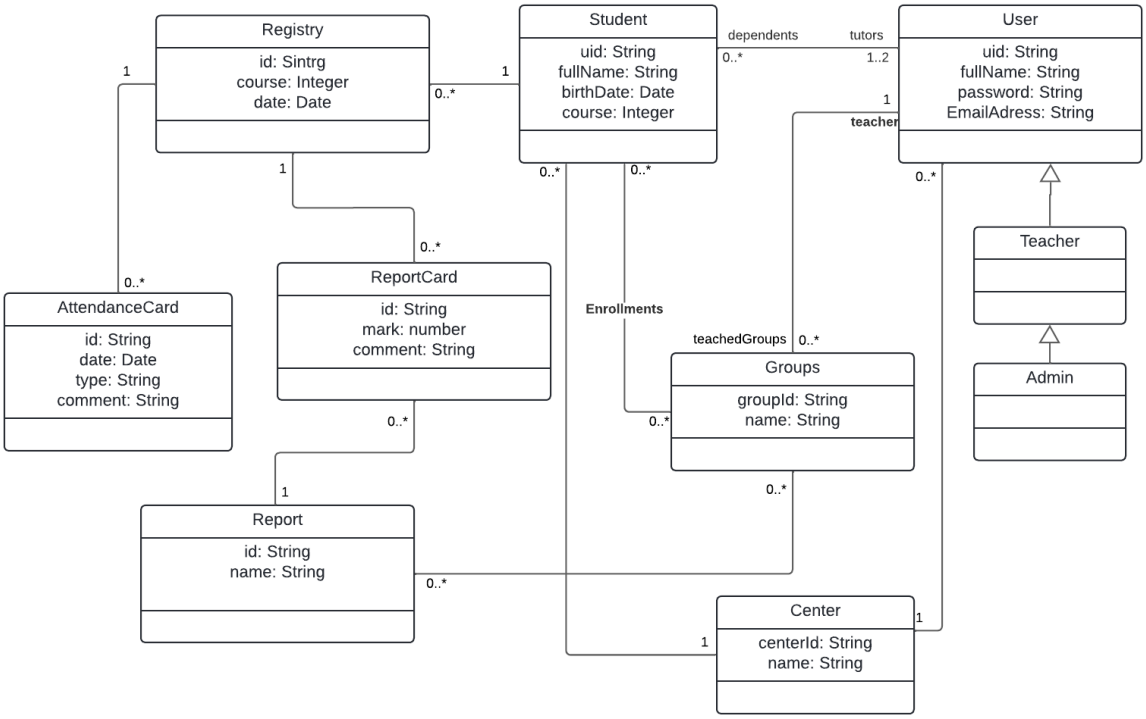

Figura 36 - Diagrama de clases MusicApp

El desarrollo del Front end de la aplicación está enfocado en la componentización, se pretende diseñar los distintos elementos de las interfaces como componentes únicos que se integren en los *layouts* o ventanas de la aplicación, de esta manera se fomenta la reutilización y el no repetir código.

En la figura 39 podemos observar la estructuración por la que se ha optado par esto componentes. Se plantea una división principal en función de los roles de los usuarios, debido a que existen ventanas propias de estos tipos y por lo tanto elementos único por rol. Además de esto, en *basic* encontramos componentes que no tienen que ver con el dominio de nuestra aplicación y por lo tanto podrían ser empleador en cualquier otra aplicación desarrollada con React native en typescript. Por último, en la carpeta *common* encontraremos los componentes que son comunes a las pantallas de todos los roles.

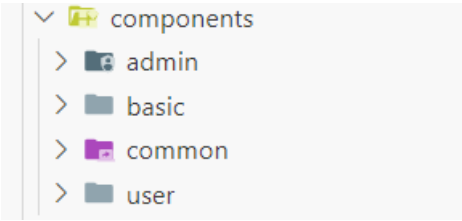

Figura 37 - estructuración de componentes

Para las pantallas o layouts se plantea la misma división por roles, también existe la carpeta common en este caso para elementos tanto de la navegación como pantallas que sean completamente comunes a los tres roles. Podemos ver esto en la figura 40.

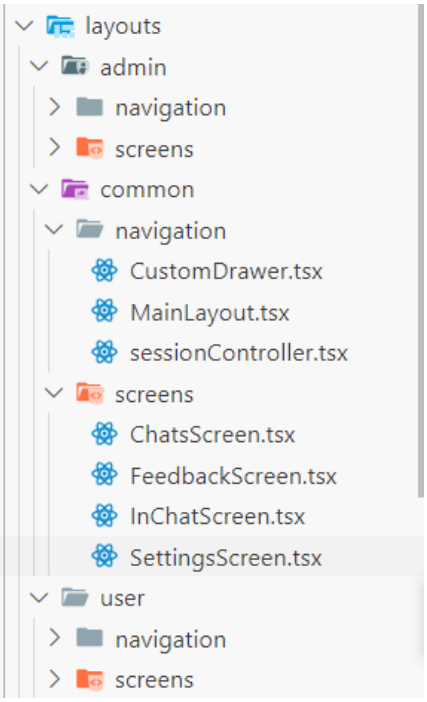

Figura 38 - estructuración de las pantallas

El emplear TypeScript nos permite el uso de clases y tipos, esto es muy comveniente para definir de manera adecuada la lógic de negocio en el Front end de nuestra aplicación, para ello simplemente se han creado tipos para cada instancia. En la figura 41 podemos ver la gestión del tipado, en la carpeta types además de lo comentadotambién se almacenan algunos enumerados útiles y tipos de parámetros para funciones.

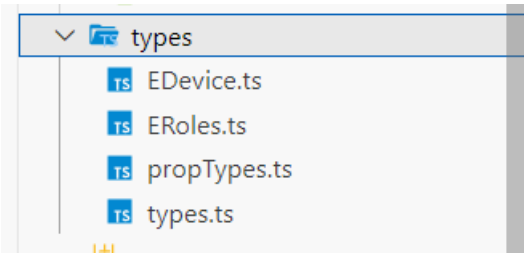

Figura 39 - tipado en MusicApp

Supabase cuenta con interfaces que han facilitado la tarea de creación de la base de datos en nuestro backend. Esto ha resultado sencillo, crear las tablas y establecer sus atributos se conseguir con un par de *clicks*, todo ello sin ser necesario escribir SQL. En la figura 42 se pueden observar todas las tablas que existen actualmente en a base de datos además se brinda información sobre ellas como descripciones, cantidad de filas que tiene cada una y una aproximación del tamaño que ocupan en memoria.

Name	Description	Rows (Estimated)	Size (Estimated)	Realtime Enabled	
AttendanceCards	attendance register of student	2	32 kB	×	6 columns
Centers	verified centers	3	48 kB	×	2 columns
ConfigPreferences	No description	1	32 kB	×	4 columns
Dependents	No description	5	24 kB	×	3 columns
Enrollments	No description	8	24 kB	×	2 columns
Groups	Relation between teachers a...	8	48 kB	×	8 columns
Profiles	people who can use the sof...	10	48 kB	×	4 columns
Registries	Data registries of students	4	56 kB	×	5 columns
ReportCards	No description	4	32 kB	×	6 columns
Reports	No description	3	32 kB	×	4 columns
Roles	No description	3	48 kB	×	3 columns
Students	Verified students in the syst...	6	48 kB	×	7 columns
UserRoles	No description	9	40 kB	×	3 columns

Figura 40 - tablas en la base de datos

Supabase, también nos ofrecen una herramienta de visualización para poder observar el esquema global de las tablas de la base de datos y sus relaciones, en la figura 41 podemos ver esta representación global.

Se puede apreciar cómo se han creado tablas para las relaciones entre clases de muchos a muchos, la taba *Enrollments* permite conocer que estudiantes pertenecen a que grupos y *UserRoles* que permite saber que roles corresponden a cada usuario, la tabla *ConfigPreferences* también se ha añadido, para guardar la preferencias de configuración del usuario y sincronizar entre dispositivos.

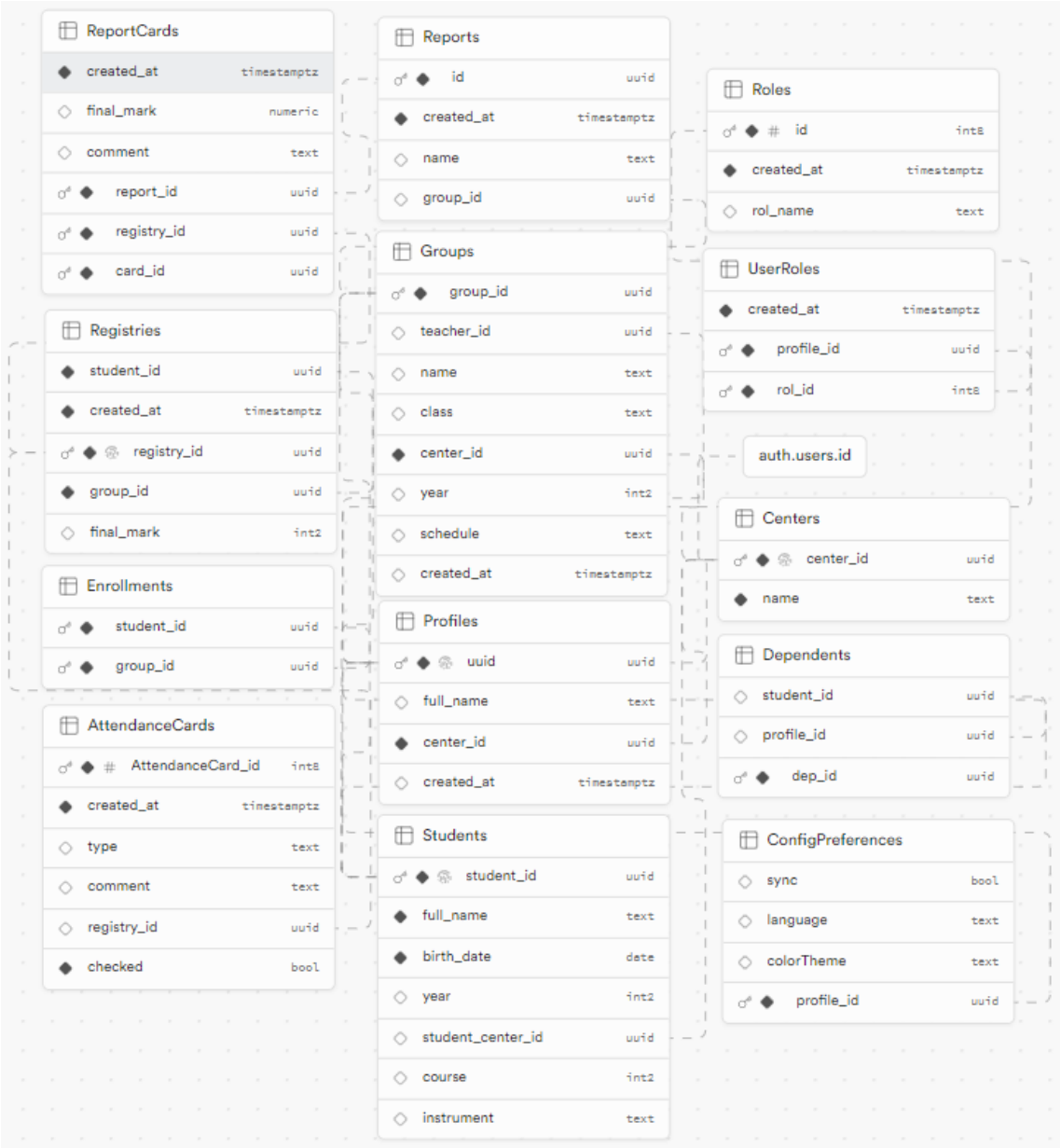

Figura 41 - Esquema global

4.4 Programación

Durante el desarrollo se encontraron dificultades y tareas que requirieron soluciones más costosas de desarrollar, en este apartado se presentan los aspectos más importantes que han surgido durante el desarrollo de MusicApp. También se hace hincapié en los patrones de programación que se han seguido y se explica cada uno de ellos.

Funciones Postgres

Como ya hemos visto las funciones en PostgreSQL son útiles cuando se necesita manejar lógica compleja directamente en la base de datos, evitando duplicar código y optimizando el rendimiento. Supabase ofrece una interfaz útil para crear estas funciones donde solo hay que escribir de manera secuencial, lo que debe hacer la función.

En la figura 42 podemos observar cómo se accede a la tabla *Registries* y se seleccionan todos sus atributos, además de esto se hace un *join* con la tabla *Groups*, para poder obtener el nombre del grupo al que hace referencia este *registry*. Después filtra los resultados para que estén contenidos entre las fechas aportadas y los devuelve ordenados, esta función se emplea en la ventana de consultar reportes de asistencia y en la de consultar evaluaciones ya que el nexo de estos es un objeto *registry*.

```
BEGIN
    RETURN QUERY
    SELECT
        r.registry_id,
        r.student_id,
        r.created_at,
        r.group_id,
        r.final_mark,
        g.name
    FROM
        public."Registries" r
        JOIN public."Groups" g ON r.group_id = g.group_id
    WHERE
        r.created_at >= start_date
        AND r.created_at <= end_date
        AND r.student_id = studentId
    ORDER BY
        r.created_at;  -- Ordenar por fecha de creación
END;

```

Figura 42 - Función Postgres get_registries_between_dates.

En la aplicación, esta y las demás funciones Postgres se han reunido en una clase llama *ApiService,* se insiste más adelante en esta clase en el apartado de patrones. Un ejemplo de cómo se puede llamar a esta función desde la aplicación se muestra en la figura 43.

```
async getRegistriesBetweenDatesAndStudentId(
    session: supabaseSession,
    start_date: Date,
    end_date: Date,
    studentid: string,
  ) {
    try {
      if (!session?.user) throw new Error('No user on the session!')
      const { data, error } = await supabase.rpc('get_registries_between_dates', {
        end_date,
        start_date,
        studentid,
      })
      if (error) {
        throw error
      }
      if (data !== null && data.length > 0) {
        const Registries: RegistriesGroupName[] = data.map((item: any) => ({
          registry_id: item.registry_id,
          student_id: item.student_id,
          group_id: item.group_id,
          course: item.course,
          created_at: item.created_at,
          finalMark: item.final_mark,
          group_name: item.groupname,
        }))
        return Registries
      }
    } catch (error) {
      if (error instanceof Error) {
        console.error(error.message)
      }
    }
  }
```

Figura 43 - Función Postgres llamada desde el Front end

Trasladar la lógica a las funciones Postgres en lugar de a la aplicación ayuda a la hora de tener tiempos de respuesta menores y menor carga de trabajo para los dispositivos. Aun así algunas operaciones se han realizado directamente en la aplicación debido a su baja complejidad. En la figura 44 puede observarse un ejemplo.

```
async deleteDependent(dep_id: string) {
    const { data, error } = await supabase
      .from('Dependents')
      .update({ student_id: null, profile_id: null })
      .match({ dep_id })
    if (error) {
      console.error('Error deleting data:', error.message)
      return
    }
  }
```

Figura 44 -Función que llama a Supabase

Políticas a nivel de fila RLS

Las políticas RLS han sido muy importantes para el desarrollo de MusicApp a la par que costosas. Para definir estas políticas se han empleado funciones Postgres ya que su uso facilita la reutilización de código y mejora la eficiencia general del sistema. Algunas de estas funciones se presentan a continuación.

```
BEGIN
  RETURN EXISTS (
    SELECT 1
    FROM public."UserRoles" ur
    JOIN public."Roles" r ON ur.rol_id = r.id
    WHERE ur.profile_id = user_id
      AND r.rol_name = 'Teacher'
  );
END;
```

Figura 45 - Función isTeacher()

```
DECLARE
    is_admin BOOLEAN;
BEGIN
    -- Verificar si el usuario actual es administrador
    SELECT EXISTS (
        SELECT 1
        FROM public."UserRoles" ur
        JOIN public."Roles" r ON ur.rol_id = r.id
        WHERE ur.profile_id = auth.uid()
          AND r.rol_name = 'Center_admin'
    ) INTO is_admin;

    RETURN is_admin;
END;
```

Figura 46 - Función isCenterAdmin()

Las funciones *isTeacher* de la figura 45 y *isCenterAdmin* de la figura 46 son muy similares, accede a la tabla *Roles* y a la tabla *UserRoles* para saber que rol corresponde, en el caso de *isTeacher*, el usuario que se estudia es pasado por parámetro con un *profile_id* y en el caso de *isCenterAdmin* se emplea la función *auth.uid()* que nos brinda *Supabase* y devuelve el id del usuario que está ejecutando la función en ese momento.

Las funciones descritas se han empleado en numerosas políticas RLS para determinar que funciones CRUD podía hacer cada tipo de usuario pero también cómo es que se hacían esas funciones.

Pongamos el siguiente ejemplo, los tres tipos de usuario pueden hacer *select* de la tabla *Students*, pero el profesor solo recibirá los estudiantes de sus grupos el administrador los de todo el centro y el usuario recibirá la información de los estudiantes a su cargo, esto se consigue con las políticas mostradas en la figura 47.

```
alter policy "select_students_for_center_admin"
on "public"."Students"
to authenticated
using(  (EXISTS ( SELECT 1
    FROM (("UserRoles"
      JOIN "Roles" ON (("UserRoles".rol_id = "Roles".id)))
      JOIN "Profiles" ON (("UserRoles".profile_id = "Profiles".uuid)))
   WHERE (("Profiles".uuid = auth.uid()) AND ("Roles".rol_name = 'Center_admin'::text) AND
   ("Profiles".center_id = "Students".student_center_id))))
);

alter policy "select_students_for_users"
on "public"."Students"
to authenticated
using (
  (EXISTS ( SELECT 1
   FROM ("Dependents"
     JOIN "Profiles" ON (("Dependents".profile_id = "Profiles".uuid)))
  WHERE (("Dependents".student_id = "Students".student_id) AND
  ("Profiles".uuid = auth.uid()))))

);

alter policy "select_students_for_teacher"
on "public"."Students"
to authenticated
using (
  (EXISTS ( SELECT 1
   FROM ((("Enrollments" e
     JOIN "Groups" g ON ((e.group_id = g.group_id)))
     JOIN "UserRoles" ur ON ((ur.profile_id = auth.uid())))
     JOIN "Roles" r ON ((ur.rol_id = r.id)))
  WHERE ((e.student_id = "Students".student_id) AND
  (g.teacher_id = auth.uid()) AND (r.rol_name = 'Teacher'::text))))

);
```

Figura 47 - Políticas RLS para select Students

Existen otras políticas mucho más simple como las de actualizar un estudiante, la cual solo emplea la función *isCenterAdmin()*. Puede verse en la figura 48.

```
alter policy "edit studnets for center adsmins"
on "public"."Students"
to public
using (
  is_center_admin()
);
```

Figura 48 - Política RLS update Students

Patrones

Durante este desarrollo se han aplicado 2 patrones de programación, el primero de ellos es el patrón Singleton es un patrón de diseño de software que asegura que una clase tenga solo una instancia y proporciona un punto de acceso global a dicha instancia. Este patrón es útil en situaciones donde se necesita un control único sobre un recurso compartido, como una conexión de base de datos o una configuración de red [42].

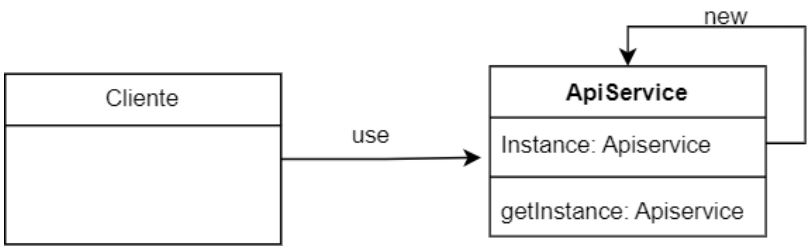

Figura 49 - Diagrama patrón Singleton

En nuestro caso nos interesa que solo exista una instancia de la clase ApiService y que esa instancia se emplee en las distintas ventanas de la aplicación. Esto es crucial para gestionar las solicitudes a la API de manera eficiente y evitar problemas de estado inconsistente. Como podemos ver en el código de la figura 46 y en el diagrama de la figura 45, la misma clase es responsable de crear y mantener su única instancia.

```
export class ApiService {
  private static instance: ApiService | null = null

  private constructor() {}

  public static getInstance(): ApiService {
    if (ApiService.instance === null) {
      ApiService.instance = new ApiService()
    }
    return ApiService.instance
  }
  ...
}
```

Figura 50 - Ejemplo código patrón singleton en Apiservice

El segundo patrón implementado es el patrón *Facade* o Fachada, mediante el cual se gestiona de manera sencilla la lógica de guardado de datos, en este caso se ha desarrollado para gestionar el guardado de las preferecias de configuración de los usuarios. A tratarse de una aplicación multiplataforma se debe conocer el tipo de dispositivo ya para implementar este almacenamiento de datos en memoria se ha empleado *localstorage* para la versión navegador y *asyncstorage* para la versión móvil, existe una tercera implementación para guardar la información en la tabla ConfigPreferences de nuestro backend, útil en los casos en los que se quiere sincronizar la configuración entre dispositivos.

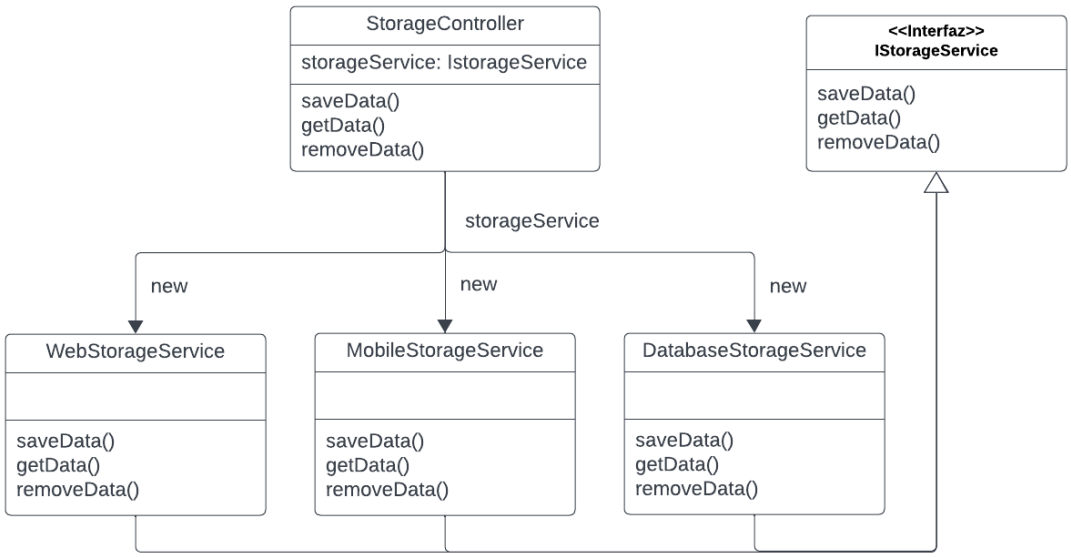

Figura 51 - representación gráfica del patrón fachada implementado

En la figura 51 se ve como la clase *StorageController* corresponde a la fachada, que se encarga de todo este proceso detrás de una interfaz de tres acciones, guardar, obtener y eliminar información, gestionar de donde es la labor de StorageController. En la figura 52 podemos observar cómo se consigue esto.

```
export class StorageController {
  private storageService: IStorageService

  constructor() {
    if (Platform.OS === 'web') {
      this.storageService = new WebStorageService()
    } else {
      this.storageService = new MobileStorageService()
    }
  }

  useDatabaseService() {
    this.storageService = new DatabaseStorageService()
  }

  async saveData(
    key: string,
    value: { sync: boolean; colorTheme: string; language: string },
    syncWithDatabase: boolean = false,
  ): Promise<void> {
    if (syncWithDatabase) {
      this.useDatabaseService()
    }

    await this.storageService.saveData(key, value)
  }
  ...
}
```

Figura 52 - StorageController

Según el tipo de dispositivo establece el *storageService* mediante un constructor y de manera independiente si se detecta la preferencia de *syncWithDatabase*, es decir, la preferencia de sincronizar con otros dispositivos se guardará en la base de datos las configuración. Al realizar una de las operaciones posibles se llamará a esa misma función del objeto storageService y en cada escenario se ejecutará la función que corresponda.

4.5 Pruebas

Las pruebas de código son fundamentales para asegurar la calidad y fiabilidad de una aplicación. Actúan como una protección ante errores imprevistos, permitiendo identificar y resolver problemas antes de que se presenten en el entorno de producción. Además, contar con un conjunto sólido de pruebas automáticas facilita el mantenimiento continuo del proyecto, al brindar seguridad al realizar modificaciones en el código.

Pruebas unitarias

Las pruebas unitarias se diseñan para verificar el correcto funcionamiento de las unidades individuales de código. Se han llevado a cabo numerosas pruebas unitarias, empleando la herramienta Jest se han construido un total de 101 pruebas. Estas se ejecutan de manera automática y evaluaban el comportamiento de funciones, teniendo en cuenta diferentes casos plateados verificando los valores de los resultado obtenidos y buscado siempre una cobertura[39] del código testeado del 100%. Se ha probado principalmente valores de inputs para todas las ventanas de la aplicación, en la figura podemos observar el resultado de pasar las pruebas unitarias de Jest.

```
--------------------|---------|----------|---------|---------|-------------------
File                | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
--------------------|---------|----------|---------|---------|-------------------
All files           |     100 |      100 |     100 |     100 |
 formValidations.ts |     100 |      100 |     100 |     100 |
 utils.ts           |     100 |      100 |     100 |     100 |
--------------------|---------|----------|---------|---------|-------------------

Test Suites: 3 passed, 3 total
Tests:       101 passed, 101 total
```

Figura 53 - Pruebas unitarias con Jest

A continuación se presentan ejemplos de algunas de las pruebas realizadas .

```
1  describe('validatePassword', () => {
2    it('should return true for a valid password', () => {
3      expect(validatePassword('Password123!', 8)).toBe(true)
4    })
5
6    it('should return false for a password with length less than minimum length', () => {
7      expect(validatePassword('Pass123!', 10)).toBe(false)
8    })
```

Figura 54 - tests de la función validatePassword()

En este sencillo test podemos ver que se está probando la función *validatePassword* empleada en las ventanas de creación de usuarios del *layout* de los administradores. Esta función comprueba que la contraseña proporcionada tiene por lo menos una mayúscula, un número y un carácter especial, además también verifica si cumple con el mínimo de caracteres establecido, que también se pasa por parámetro.

[39] **Cobertura:** porcentaje de líneas de código, sentencias, funciones por las que se pasa al testear un archivo, 100% equivale a todos los caminos posibles.

```
describe('formValidations', () => {
  beforeEach(() => {
    jest.clearAllMocks()
  })
  describe('validateEmail', () => {
    it('should return true for a valid email', () => {
      expect(validateEmail('test@example.com')).toBe(true)
    })

    it('should return false for an invalid email', () => {
      expect(validateEmail('test@example')).toBe(false)
    })
  }
}
```

Figura 55 - tests de la función validateEmail()

Esta otra prueba verifica si un correo electrónico proporcionado tiene el formato correcto, esto se ha logrado gracias al uso de expresiones Regex[40], contenidas en la función *validateEmail*.

Pruebas de integración

Las pruebas de integración son fundamentales para garantizar que los diferentes componentes de un sistema funcionen correctamente al combinarse y comunicarse entre sí. A diferencia de las pruebas unitarias, que evalúan unidades individuales de código en aislamiento, las pruebas de integración se centran en probar la interacción entre módulos, servicios, o sistemas externos.

Permiten identificar posibles errores en los puntos de interacción entre los componentes. Al verificar la correcta cooperación entre los distintos elementos del sistema, las pruebas de integración aseguran que el flujo general de la aplicación funcione como se espera, abordando posibles fallos que pueden no darse cuando los módulos se prueban individualmente.

En el caso de Supabase las pruebas de integración no han podido realizarse de manera automática, ya que con Jest no es posible generar pruebas que accedan a funciones de supabase sin estar ejecutando la aplicación, ya que es necesaria una sesión activa para ello. Pero se han creado pruebas manuales con componentes que verifican la integración con Supabase, en la figura 56 puede verse una de las pruebas realizadas.

[40] **Funciones Regex:** sirven para localizar trozos de texto dentro de otro texto mayor.

```
import { supabase } from "../../../src/lib/supabase";

async function pruebaConexionBaseDatos() {
  try {
    const { data, error, status } = await supabase
      .from('Students').select('*').limit(1);
    if (status === 200) {console.log("Conexión exitosa: Código de estado 200");}
    else {console.log(`Código de estado devuelto: ${status}`);}
    if (error) {console.error(`Error al conectar a la base de datos: ${error.message}`);}
    else {console.log("Datos obtenidos:", data);}
    return data;
  } catch (e) {
    console.error("Error en la prueba de conexión:", e);
  }
}
pruebaConexionBaseDatos();
```

Figura 56 - prueba de integración con Supabase

Este tipo de pruebas, realmente han sido útiles en los momentos en los que se necesitaba conocer si una función Postgres estaba funcionando correctamente o si por el contrario tenía un comportamiento inesperado, de esta manera se podían observar los errores provenientes de Supabase y también era eficaz en los casos que había un problema con las políticas. Estas pruebas han servido durante el desarrollo y para una publicación de MusicApp deberían quedar fuera de la aplicación y así no generar posible ruido en la ejecución ni añadir carga de trabajo al dispositivo ni a la API de supabase.

Pruebas de aceptación

Las pruebas de aceptación son cruciales para asegurar que un sistema o aplicación cumple con los requisitos y expectativas del cliente o usuario final. Con estas pruebas nos centraremos en validar que las funcionalidades cumplen con los casos de uso y los criterios de aceptación definidos.

Para todas la tareas desarrolladas relacionadas con alguna ventana de la aplicación se han creado pruebas de aceptación, estas pruebas validaban el comportamiento de estas ventanas y sus componentes al responder a las acciones del usuario. Estas pruebas se han pasado de manera manual, al finalizar el desarrollo de cada tarea se pasaban las pruebas de aceptación para comprobar si su funcionamiento final era el previamente requerido. Seguidamente nos centraremos en una tarea concreta y analizaremos las pruebas de aceptación de esta en la figura 57.

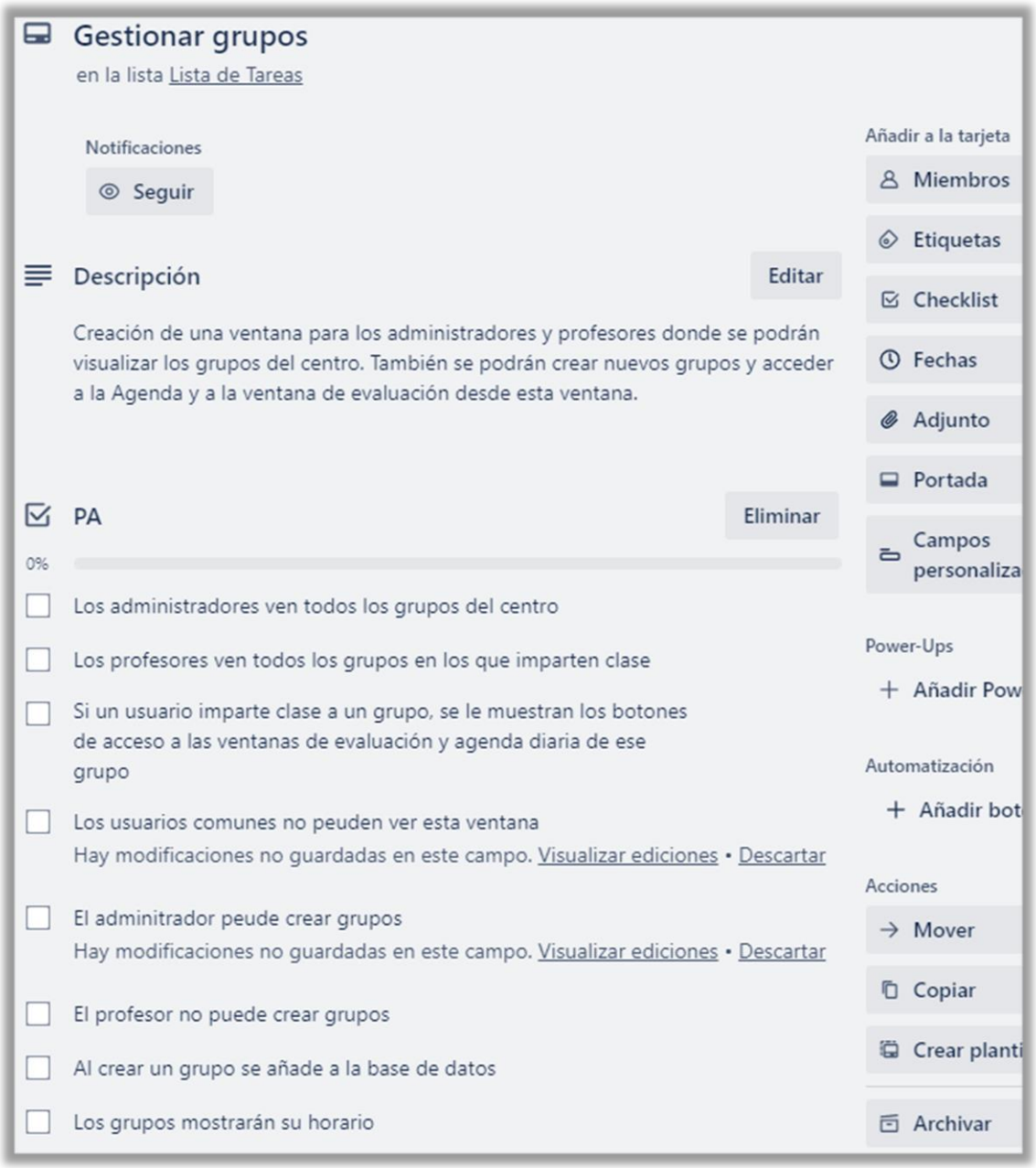

Figura 57 - pruebas de aceptación tarea Gestionar grupos.

De esta manera al comprobar de nuevo las pruebas de aceptación después de haber realizado cambios en la funcionalidad probada, estas pruebas pasan a ser pruebas de regresión, que comprueban que el funcionamiento no ha cambiado a pesar de lo añadido.

Como se observa en la figura 57 muchas de la pruebas hacen referencia al sistema de roles, validando quién puede hacer que tareas en la ventana de grupos. Al pasar las pruebas las marcaríamos y daríamos por finalizada la tarea.

Pruebas de requisitos no funcionales

- Disponibilidad

No se han realizado pruebas concretas para garantizar la disponibilidad de MusicApp pero como se ha comentado, se cuenta con el empleo de la herramienta Sentry , que monitoriza las sesiones de uso de la app y notifica rápidamente si es que ocurre un Error en alguna de estas sesiones. Además Supabase garantiza una disponibilidad elevada por lo que nos apoyamos firmemente en ello.

En caso de que la aplicación fallara a causa de un error no controlado y provocara una falla en la disponibilidad de MusicApp, Sentry notificaría del error por lo que podría ser rápidamente solucionado y podría publicarse una nueva versión en ese momento.

- Adaptabilidad

Expo Go ha permitido comprobar que MusicApp es altamente adaptable, siendo posible generar una versión funcional de la aplicación en versión web, Android e iOS.

- Seguridad

La seguridad de MusicApp se logra con varios factores, en primer lugar el servicio de Auth junto con la navegación de la aplicación gestionan integralmente la Autenticación en la aplicación.

```
1  const SessionController = () => {
2    const session = useSession()
3
4    return (
5      <>
6        <View style={{ flex: 1 }}>
7          {session && session.user && session.user.confirmed_at ? <MainLayout /> : <Auth />}
8        </View>
9      </>
10   )
11 }
```

Figura 58 - Navegación de Autorización

Como se ve en la figura 58, gracias a un hook recuperamos la sesión, guardada en un provider como una variable global. Si esta sesión deja de existir la app redirige al usuario a la ventana *Auth* donde debe autenticarse nuevamente. Supabase emplea JWTs (JSON Web Tokens) de manera que cuando se haga un inicio de sesión, Supabase devolverá un *AuthToken*. Solo con este token el usuario será capaz de mandar solicitudes a la API de Supabase. En la figura 59 se puede apreciar la función que devuleve el *token* de autenticación, *singWithPassword*().

```
async function handleSingIn() {
  setClick(!click)
  setLoading(true)
  const { error } = await supabase.auth.signInWithPassword({
    email: email,
    password: password,
  })
  if (error) Alert.alert(error.message)
  setLoading(false)
}
```

Figura 59 - Función de inicio de sesión

En cambio la Autorización en MusicApp se garantiza gracias a las Políticas RLS, se han realizado pruebas manuales, con ayuda de la interfaz de Supabase, que ofrece una consola SQL con la que interactuar con la base de datos, interpretando a un usuario de los del sistema. En la figura 60 y 61 podemos ver las pruebas realizadas y sus resultados.

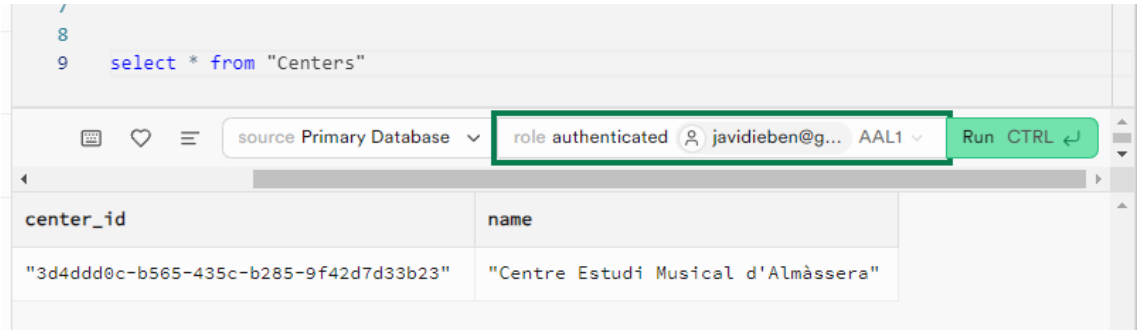

Figura 60 - select Centros para un usuario

Podemos observar que este usuario solo puede visualizar la información de los centros a los que pertenece y no está autorizado a operar sobre información de otros centros. La RLS con la que se hace esto posible se presenta en la figura 61.

```
alter policy "select_centers"
on "public"."Centers"
to public
using (
  (EXISTS ( SELECT 1
   FROM "Profiles"
  WHERE (("Profiles".center_id = "Centers".center_id)
  AND ("Profiles".uuid = auth.uid()))))

);
```

Figura 61 - RLS de selección de centros

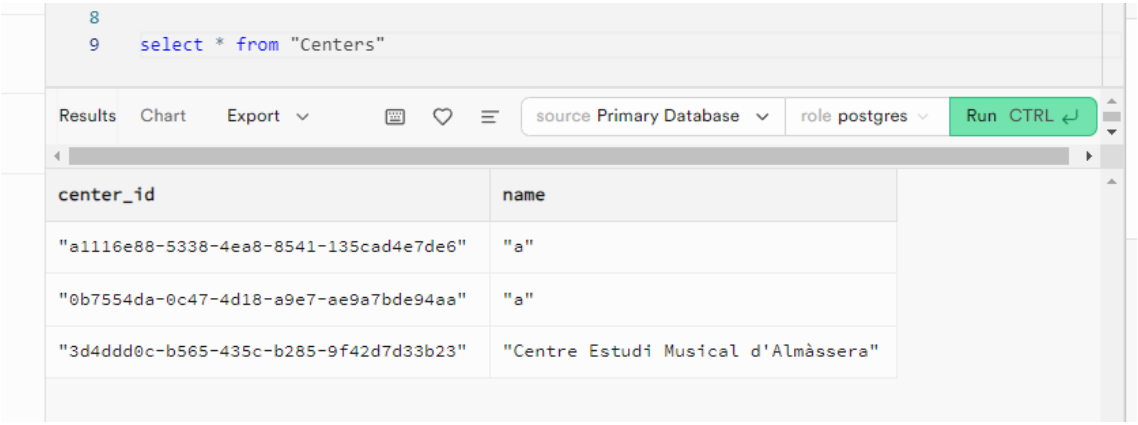

Figura 62 - select Centros para rol Postgres

Podemos observar que existen mas centros gracias a que al hacer la misma petición con el rol Postgres, que es el rol administrados de la base de datos, nos devuelve ahora si toda la información de la tabla.

Supabase también aplica el cifrado de datos, las contraseñas de los usuarios no pueden visualizarse ya que se encuentran encriptadas, de esta manera también se protege a los usuarios de la plataforma. Esto puede apreciarse en la figura 63.

ır	encrypted_password varchar
gmail.com	$2a$10$1sn8pzk22C0mebvZHkNvK.T/WpDNt2J.5gAbjrdKVCuKVPZnCIXqK
com	$2a$10$vo2r2doBBeb3gHLraunSmOQr4kcpsVMsGF7QZRWipCg/LTE8a71rK
com	$2a$10$RH1.DXR.IF.ubVNS6tBcWOIUBntV3nXXjA/RHVm//oqgnQpV3M/Fm
mail.com	$2a$10$2r2c/Zckregu5s6vVDZYuuwaWDdfT75aGBhKOEhgNk.cwBWluP1IO
ens@gmail.com	$2a$10$dA40TpcoVgSNX/MC4OkH7u7rfaWbVK/3ALvegsqKySY2qNGB61GEG
l.com	$2a$10$/jJBtFQwu/3Pixq.GdTXQOMHbZNBXvmwzgS97WDjfXTzKXaXIZ7Oq
mail.com	$2a$10$IzlWAnR9Xd6DAE.9LbgREOIlGufaN/kalHsQyZQkEpVqEilijjPuK
zbens@gmail.com	$2a$10$QUNXh0JNVMORF1ZZsXbW8.ydFe0j3h2v1OsXOvnC3Lff16pcXvgMG

Figura 63 - contraseñas encriptadas

- Escalabilidad

Gracias a Supabase, se garantiza la escalabilidad en la gestión de datos y autenticación de manera robusta y eficiente. Supabase, al emplear PostgreSQL, aprovecha las capacidades de este sistema de gestión de bases de datos, conocido por su capacidad para manejar grandes volúmenes de datos y operaciones simultáneas sin comprometer el rendimiento.

Además la arquitectura *serverless* de Supabase permite que los recursos se ajusten automáticamente según la demanda, eliminando la necesidad de gestionar infraestructura adicional y permitiendo que la aplicación escale sin problemas a medida que crecen los usuarios y los datos.

No se han podido realizar pruebas con grandes volúmenes de usuarios y por lo tanto no se ha probado empíricamente la capacidad de escalabilidad de MusicApp.

- Mantenibilidad

La constante creación de pruebas unitarias durante el desarrollo es una prueba de la mantenibilidad del código, además supabase genera documentación de manera automática de su API para comunicarse con nuestro backend. En la figura 64 puede verse dicha documentación.

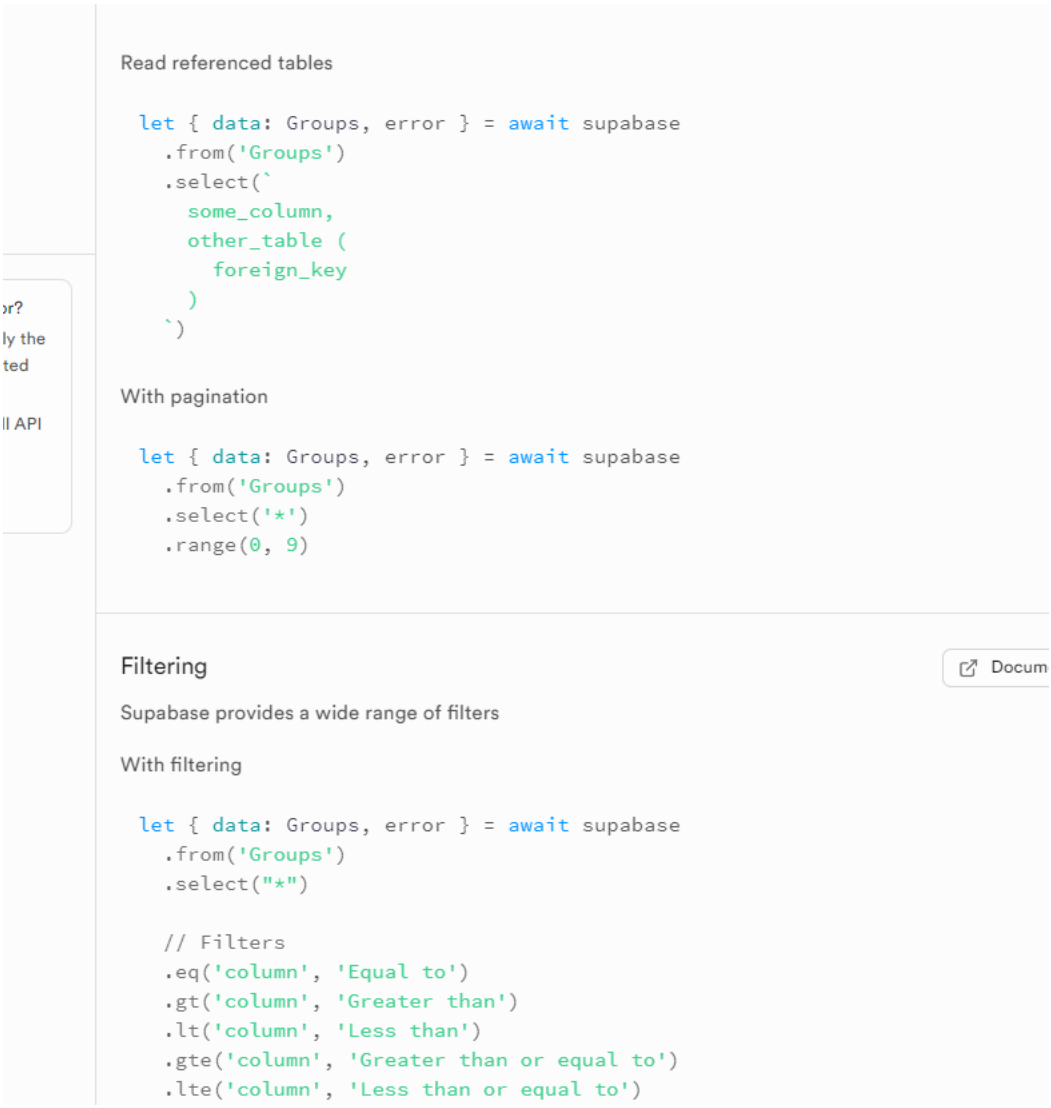

Figura 64 - Documentación Automática

CAPÍTULO 5

Cronología del TFG

En este capítulo se verán los hitos acontecidos durante el desarrollo del proyecto y se detallará de manera clara como se ha organizado el trabajo. En la imagen 41 podemos observar una línea temporal con las fechas de los Sprints y el experimento realizados durante el desarrollo del proyecto.

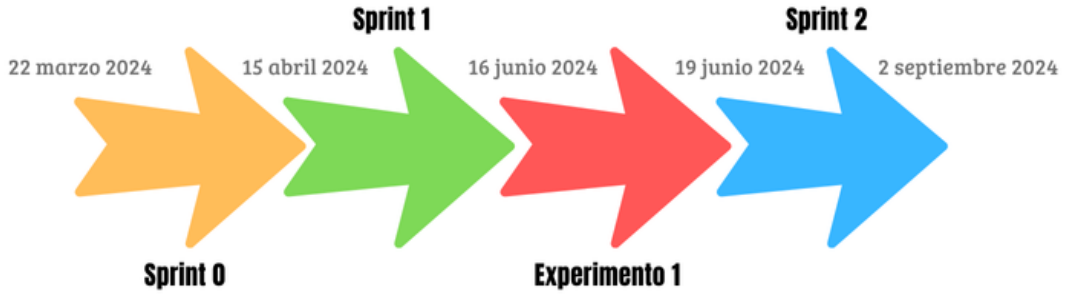

Figura 65 - Registro de hitos

5.1 Backlog inicial

Después de realizar el análisis de la idea de negocio se quiso plantear las funcionalidades que deberían incluirse en el desarrollo de la solución Para ello se realizó un mapa de características para ordenar según el orden de prioridad, las características funcionales del software.

Mapa de características

Teniendo en cuenta las funcionalidades que se tomaron para la comparación con los competidores del punto 2.2, se ha definido la prioridad que tendrían estas mediante un mapa de características, estableciéndolas de menor a mayor prioridad, en la figura 33 puede verse el resultado.

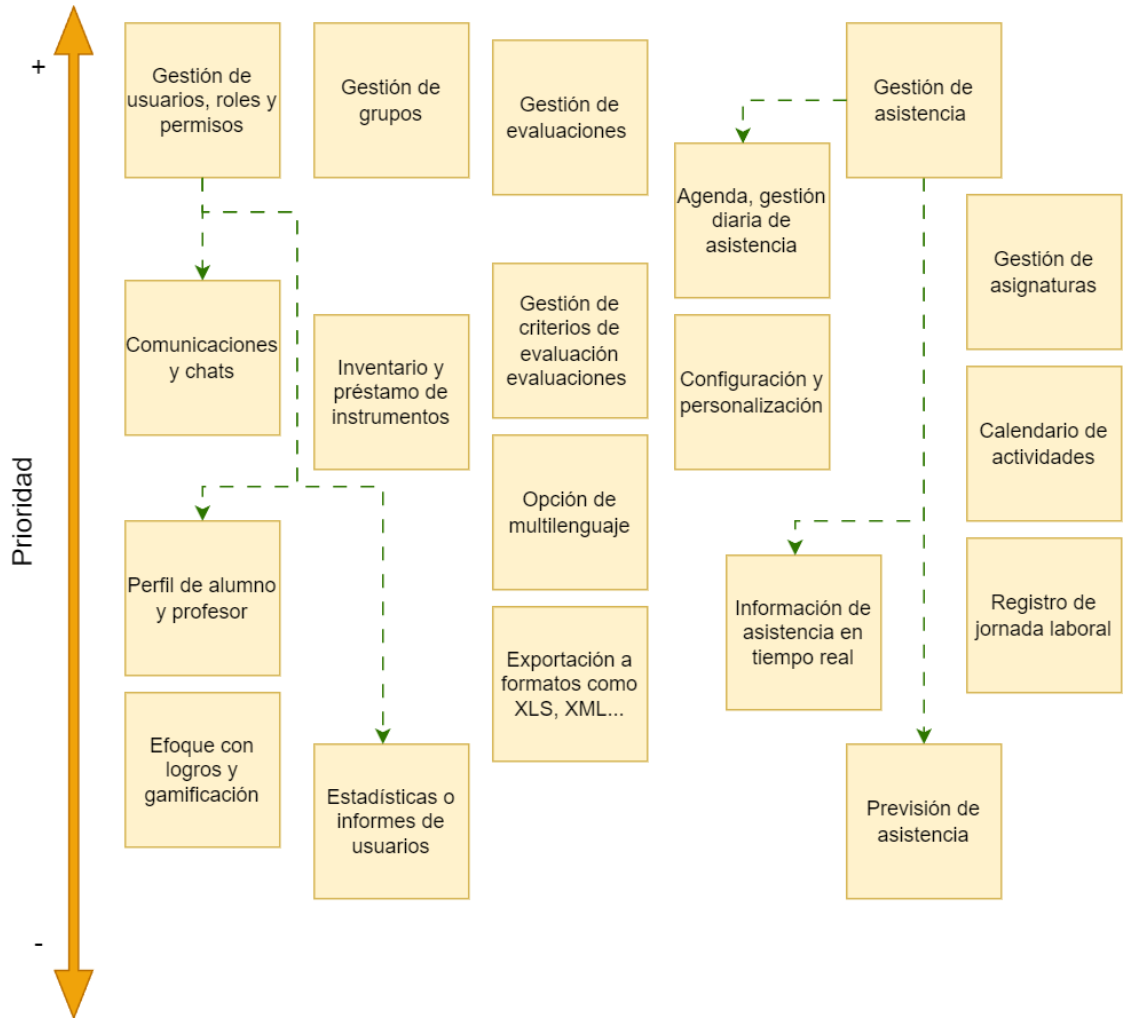

Figura 66 - Mapa de características

Con este mapa de característica se definen las tareas que se van a desarrollar ordenadas por prioridad. La estrategia seguida para realizar esta ordenación se basó en considerar como prioritarias las funciones de gestión como la gestión de usuarios, grupos, asistencia o evaluaciones. Como se puede ver en la figura 47 las flechas verdes indican relaciones de dependencia entre tareas, y es que ha sido necesario desarrollar el sistema de roles y permisos y la gestión de usuarios para poder desarrollar gran parte de la aplicación. Gracias a esta representación organizada y priorizada de las funcionalidades, se facilita la toma de decisiones respecto a lo que debía o no incluirse en el primer MVP.

Después de realizar el mapa de características el siguiente paso consistió en descomponerlas en tareas más pequeñas y manejables. Cada funcionalidad identificada en el mapa se dividió en tareas específicas y concretas. Esto se realizó con el fin de que pudieran ser completadas en un período de tiempo relativamente corto, como parte de un sprint. Con las tareas identificadas, se creó el backlog, para representar el backlog y organizar las tareas durante el desarrollo se ha empleado la herramienta Trello. En la figura 34 pueden observarse algunas tareas.

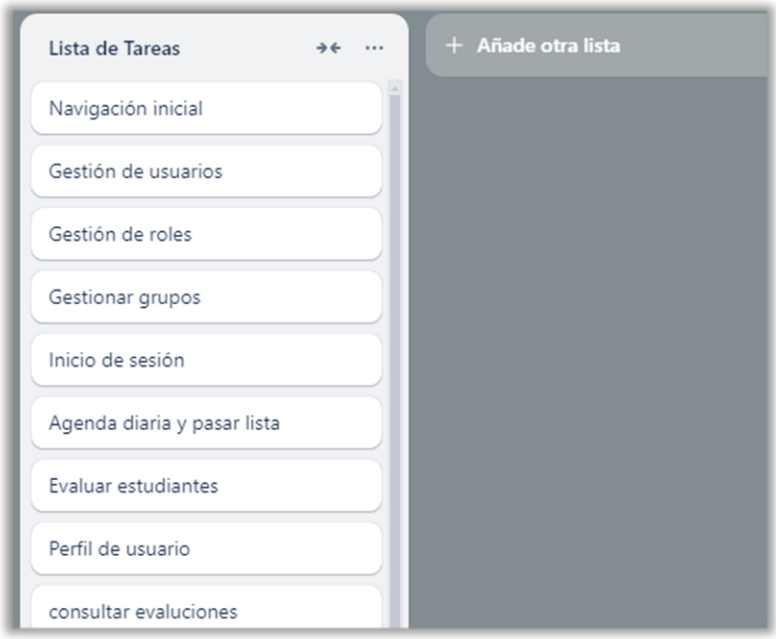

Figura 67 - Backlog inicial

Una tareas de las creadas se puede observar en la figura 35, esta tareas constan de una breve descripción con que se puede identificar la tarea y el trabajo a realizar y una seria de pruebas de aceptación que debería superarse adecuadamente una vez finalizada la tarea, y posteriormente cada vez que se realicen cambios o una tarea similar, considerando de la misma manera pruebas de regresión sobre estas funcionalidades.

Figura 68 - Ejemplo de tarea del backlog

El backlog es dinámico y puede ajustarse en función de lo que ocurra durante el desarrollo, cambios en los requisitos del usuario o nuevas prioridades del negocio. Una tarea que apareció al poco de realizar algunas interfaces fue la refactorización de algunas pantallas con un enfoque basado en componentes, ya que para hacer las primeras pantallas no se tuvo en cuenta que algunos componentes podían reutilizarse y posteriormente se sacaron de las pantallas a componentes aislados, esta tarea puede verse en la figura 36.

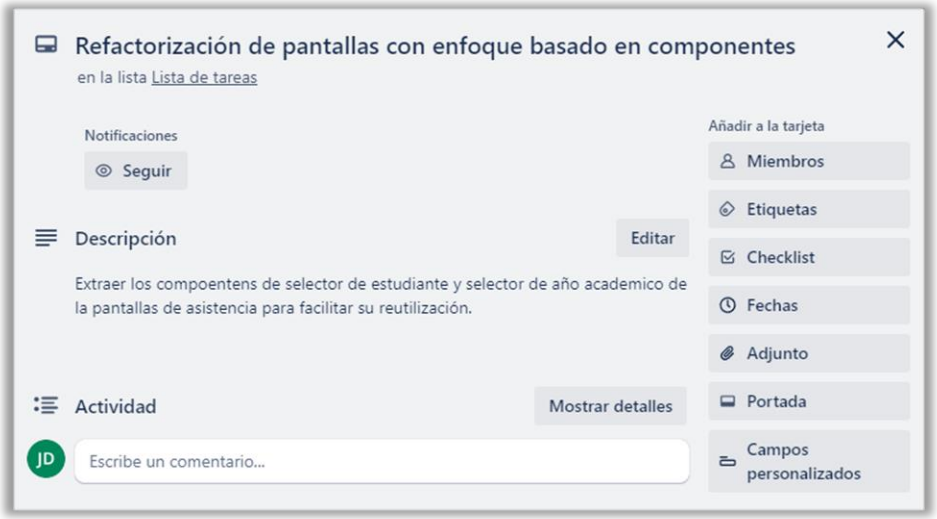

Figura 69 -Tarea de refactorización

Antes de entrar en detalle sobre las tareas a realizar en este Sprint 1, cabe mencionar que se ha decidido realizar también un Sprint 0, en él se pretenden agrupar tareas que pueden realizarse previamente al desarrollo y que facilitarán el comienzo de este. Las tareas realizadas se presentan a continuación.

Formación en las tecnologías utilizadas: Esta formación se realizó mediante el uso de videos, cursos y la propia documentación de las tecnologías utilizadas. Se ha trabajado especialmente Supabase, cuya influencia en el proyecto es notoria, también se han reforzado los conocimientos sobre React Native, empleado previamente en otros proyectos.

Puesta en marcha del entorno de desarrollo: Se preparó Visual Studio y se configuraron las extensiones empleadas. El formateador Prettier y el Linter ESLint requerían una configuración de reglas a seguir para poder aplicarlas. En la figura 38 podemos ver las reglas que aplica el formateador, intercambiará comillas dobles por simples, empleará identación de dos espacios, entre otros.

Creación del proyecto expo: Crear un proyecto de Expo fue una tarea sencilla. Mediante un comando, mostrado en la Figura 39, se genera rápidamente un proyecto de Expo con una estructura base. Al ejecutar el comando, se creará una carpeta con el nombre especificado que contendrá el proyecto.

Con las tareas priorizadas y el backlog creado podían establecerse las tareas para el primer sprint, para ello se creó otro panel concreto para el sprint 1 y a este se desplazaron las tareas seleccionadas, estas se presentan a continuación, pueden observarse en la figura 37. Posteriormente se añadió al Sprint la tarea de Multilenguaje, debido a que requeriría una carga de trabajo menor si se realizaba desde el principio del desarrollo, se añadió el valenciano como lenguaje a la aplicación. También se añadió al Sprint la tarea de visualización de reportes de asistencia.

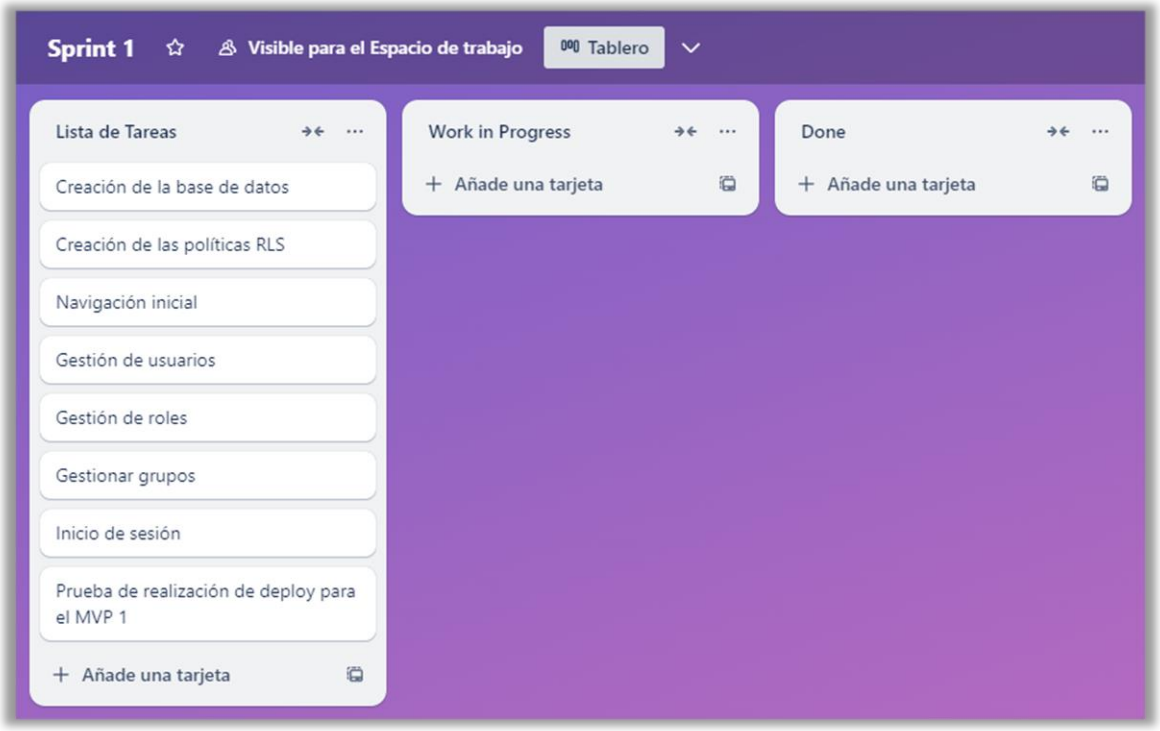

Figura 70 - Panel de Trello sprint 1

Experimento I

Después de la finalización de las tareas del Sprint 1 se llevó a cabo el experimento, se realizó el día 15 de junio de 2024. Este consistió en una reunión en el CEM con 10 participantes, entre ellos, el director de la escuela, un administrador, una profesora y los 7 restantes fueron integrantes de la Banda que se ofrecieron a interpretar el papel de usuarios comunes, siendo algunos padres de estudiantes y otros propios estudiantes del centro.

La funcionalidad presentada en este experimento consistió en:

- Inicio de sesión
- Creación de usuarios
- Creación de administradores
- Creación de profesores
- Visualización de la asistencia de un estudiante
- Creación de grupos
- Eliminación de usuarios
- Eliminación de grupos

Se había realizado previamente una publicación con EAS el servicio de publicación de aplicaciones de Expo y de esta manera se contaba con una APK[41] de la aplicación, disponible únicamente para Android, se llevaron dos ordenadores portátiles a la prueba y los usuarios que no pudieron emplear sus móviles porque poseían un dispositivo de Apple pudieron emplear los ordenadores, en dichos ordenadores se ejecutaba con expo una versión de la aplicación.

Este fue el momento de obtener retroalimentación de los usuarios, se realizaron pruebas, simulando escenarios ficticios para las funcionalidades presentadas. Después de realizarlos se organizó un ejercicio de planteamiento y resolución de cuestiones y sugerencias.0020

Entre las sugerencias plateadas surgieron nuevas ideas para funcionalidades de la plataforma: el préstamo e inventario de instrumentos, la exportación de los datos de la app a formatos de almacenamiento como XLS, la posibilidad de subida de documentos a los perfiles de los usuarios. Todas estas sugerencias se valoraron y algunas se añadieron al backlog posteriormente.

Parte de los apuntes que hicieron los usuarios estaban centrados en que las interfaces, debido a que no eran especialmente estéticas, y es cierto que este sprint no se centró el trabajo en el diseño de las interfaces.

[41] **APK:** acrónimo de Android Package Kit. Es el formato de archivo utilizado por el sistema operativo Android para distribuir e instalar aplicaciones.

Para concluir, se llevó a cabo una encuesta anónima utilizando Google Forms[42] para evaluar el experimento. Esta encuesta permitió recopilar feedback detallado de los participantes sobre su experiencia.

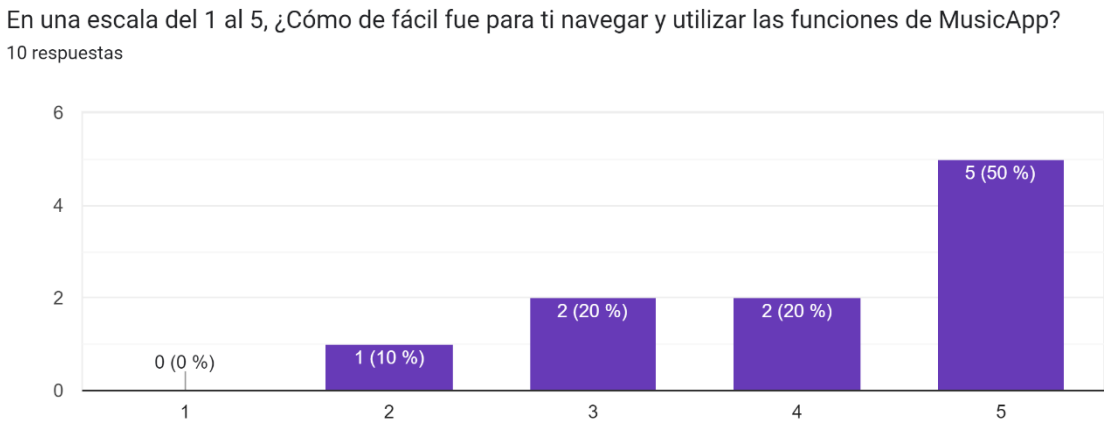

Figura 71 – pregunta 1 del experimento 1

Figura 72 – pregunta 2 del experimento 1

[42] **Google Forms:** https://www.google.es/intl/es/forms/about/.

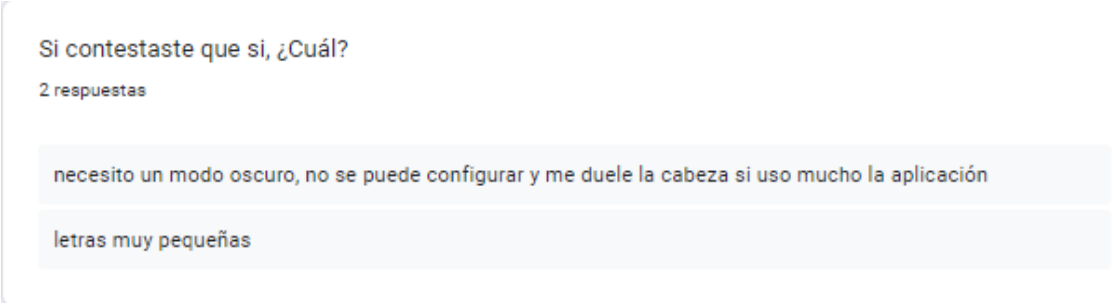

Figura 73 - sugerencias de asccesibilidad experimento 1

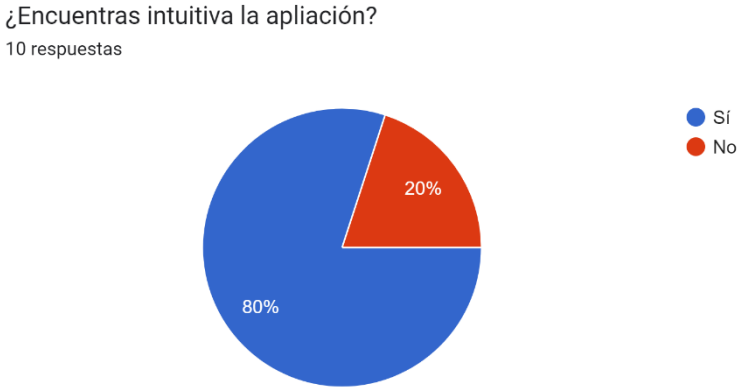

Figura 74 – pregunta 3 del experimento 1

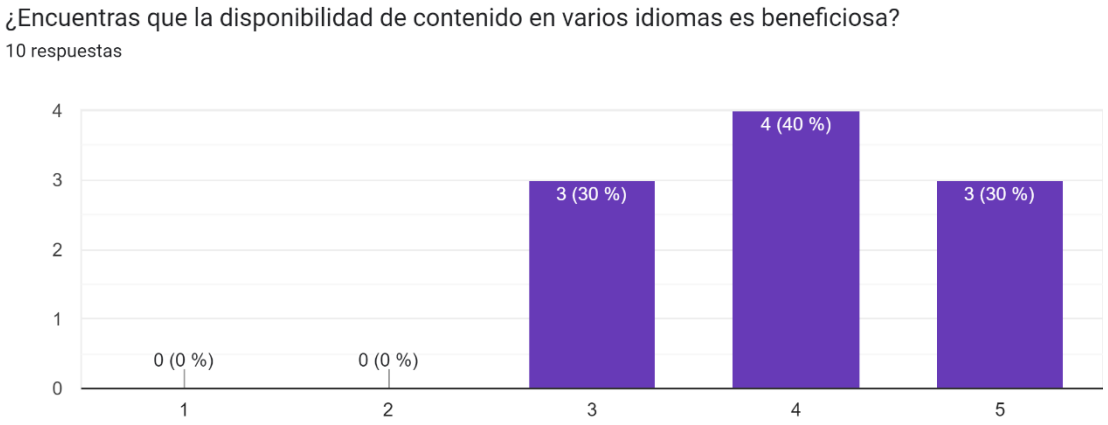

Figura 75 – pregunta 4 del experimento 1

Se observo que la mayoría de los usuarios valoro positivamente la navegación y la intuitiva que era la aplicación, se recogieron las sugerencias de accesibilidad y también se trataron como una nueva tarea del backlog.

Después de este experimento no hubo un segundo, si se realizó in segundo Sprint en el que se incluyeron siguientes tareas:

- Visualización de evaluaciones
- Agenda diaria
- Evaluar
- Configuración y preferencias
- Gestión de profesores

Debido a la falta de tiempo entere el desarrollo y la entrega del proyecto y a que este sprint se realizó en temporada de verano donde la escuela permanece cerrada no se realizó un segundo experimento. En la figura 76 puede apreciarse el panel de trello con las tareas del segundo sprint.

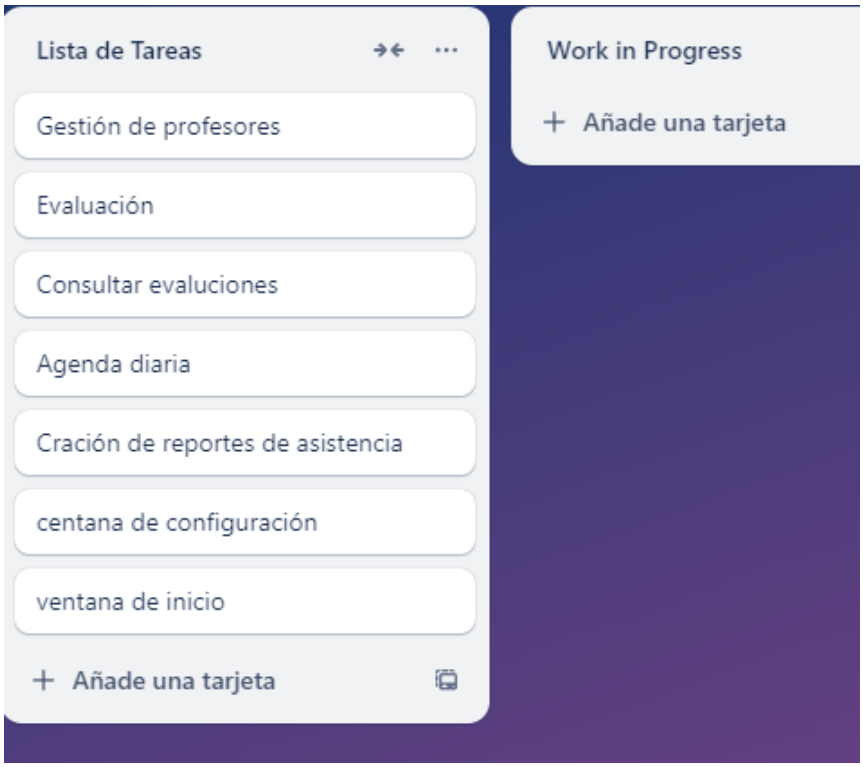

Figura 76 - tablero del Sprint 2

CAPÍTULO 6

Conclusiones y trabajo futuro

Se ha logrado cumplir con los objetivos de manera exitosa y dentro del plazo establecido. Se ha desarrollado un Producto Mínimo Viable (MVP) validado a través de un experimento, y se ha creado una aplicación capaz de presentar tanto una versión web como versiones móviles para Android e iOS.

Se ha alcanzado una tasa de aceptación del 80% según el experimento realizado, superando el objetivo inicial del 70%. El desarrollo de MusicApp ha seguido buenas prácticas, garantizando un código limpio y una arquitectura escalable capaz de adaptarse a futuras demandas. Además, las encuestas realizadas han confirmado que se ha logrado una interfaz rápida, intuitiva y fácil de usar.

Los administradores de la escuela cuentan ahora con herramientas efectivas para la monitorización y gestión de usuarios, mientras que los profesores pueden gestionar faltas y evaluaciones de manera eficiente. El desarrollo de MusicApp ha sido una experiencia enriquecedora tanto a nivel profesional como personal. La experiencia práctica en el diseño y desarrollo de una aplicación multiplataforma ha mejorado significativamente mis habilidades técnicas, lo que he podido comprobar durante las prácticas en empresa realizadas simultáneamente con el desarrollo del proyecto.

Trabajar en MusicApp ha fomentado el desarrollo de habilidades como la gestión del tiempo y la resolución de problemas. Este proyecto ha reforzado mi pasión por el desarrollo de aplicaciones y ha proporcionado una valiosa introducción al mundo laboral.

Durante el desarrollo de este proyecto, he apreciado la influencia de las asignaturas cursadas a lo largo de mi carrera. Estructuras de Datos y Algoritmos (EDA) desempeñó un papel crucial, ya que fue la primera materia que despertó mi interés por la programación. Diseño de Software (DDS) también me brindó conocimientos sobre patrones y buenas prácticas. Fundamentos de la Programación e Ingeniería del Software resultaron igualmente esenciales. Además, las asignaturas relacionadas con redes, como Tecnología de Sistemas de Información en la Red, proporcionaron conocimientos sobre arquitecturas de red cruciales para la integración con servicios como Supabase.

El estudio de Bases de Datos fue fundamental para comprender la estructura y funcionamiento de las consultas SQL. Las asignaturas de Proyecto de Ingeniería de Software y Gestión de Proyectos me ofrecieron herramientas valiosas para aplicar metodologías ágiles y gestionar el proyecto de manera efectiva, mejorando la organización y ejecución.

En el futuro, se pretende desplegar MusicApp en tiendas de aplicaciones, como Google Play Store y Apple App Store. Además, se lanzará la versión web en una plataforma como Vercel. Se planea continuar con los sprints de desarrollo, incluyendo tareas pendientes, y se pretende implementar una estrategia de pruebas exhaustiva. Esto incluirá el uso de librerías especializadas en pruebas, como React Testing Library, para verificar el comportamiento individual de los componentes y realizar pruebas de integración que aseguren que los componentes funcionen correctamente en conjunto.

Referencias

[1] Boletín Oficial del Estado, Ley Orgánica 2/2006, de 3 de mayo, de Educación. Consultado en (marzo-2024) [En línea] disponible en BOE-A-2006-7899 Ley Orgánica 2/2006, de 3 de mayo, de Educación.

[2] FSMCV, Federación de sociedades musicales de la Comunidad Valenciana. Consultado en (marzo-2024) [En línea] disponible en https://fsmcv.org/red-de-centros-educativos/.

[3] CEEI Centro Europeo de Empresas e Innovación de Valencia. Consultado en (marzo-2024) [En línea] disponible en https://ceeivalencia.emprenemjunts.es/%20?op=8&n=30029.

[4] Santander Open Academy. Consultado en (marzo-2024) [En línea] disponible en https://www.santanderopenacademy.com/es/blog/tam-sam-som.html.

[5] EDUCATECA, Base de Datos de Conservatorios y Escuelas de Música en España. Consultado en (marzo-2024) [En línea] disponible en http://www.educateca.com/guia/base-datos-conservatorios-escuelas-musica.asp.

[6] Ender la factoría del software, Atenea. Consultado en (marzo-2024) [En línea] Disponible en https://www.ender.es/.

[7] Acelera pyme, Programa kit Digital de ayudas y subvenciones a pequeñas y medianas empresas. Consultado en (abril-2024) [En línea] disponible en https://www.acelerapyme.gob.es/kit-digital.

[8] Telefónica, Noticias, Digital Toolkit. Consultado en (abril-2024) [En línea] disponible en https://fondoseuropeos.telefonica.es/noticias/digital-toolkit-todo-lo-que-necesitas-saber/.

[9] Kit Digital online, blog. Consultado en (abril-2024) [En línea] disponible en https://kitdigital.online/blog/que-es-agente-digitalizador-quien-puede-serlo/.

[10] Web oficial de Additio App. Consultado en (abril-2024) [En línea] disponible en https://additioapp.com/planes-centros/.

[11] FEDERABAND, web oficial de la Federación Andaluza de Bandas de Música. Consultado en (abril-2024) [En línea] disponible en https://federband.org/glissandoo-com-y-la-confederacion-espanola-de-sociedades-musicales-firman-un-acuerdo-de-colaboracion.

[12] Web oficial de la Federación de Bandas de Música de Navarra. Consultado en (abril-2024) [En línea] disponible en https://www.bandasdenavarra.com/glissando/

[13] ¿Cuánto Pagan por Publicidad en Páginas Web? Descubre los Ingresos. Consultado en (abril-2024) [En línea] disponible en https://adsterra.com/blog/es/cuanto-se-paga-por-publicidad/.

[14] DAFO. Página web de ipyme [Consultado en (mayo-2024) [En línea] disponible en https://dafo.ipyme.org/Home.

[15] Cómo hacer una proyección financiera. Consultado en (mayo-2024) [En línea] disponible en https://www.linkedin.com/pulse/c%C3%B3mo-hacer-proyecciones-financieras-para-tu-pyme-nuboxchile/.

[16] Elmeunebot, Define tu producto digital utilizando lean Canvas. Consultado en (mayo-2024) [En línea] disponible en https://www.linkedin.com/pulse/define-tu-producto-digital-utilizando-lean-canvas-el-meu-nebot/.

[17] Documentación de TypeScript. Consultado en (mayo-2024) [En línea] disponible en https://www.typescriptlang.org/.

[18] Documentación de React. Consultado en (mayo-2024) [En línea] disponible en https://es.react.dev/.

[19] Documentación de React, Usar TypeScript. Consultado en (mayo-2024) [En línea] disponible en https://es.react.dev/learn/typescript.

[20] FreeCodeCamp, ¿Qué es el DOM? Consultado en (mayo-2024) [En línea] disponible en https://www.freecodecamp.org/espanol/news/que-es-el-dom-el-significado-del-modelo-de-objeto-de-documento-en-javascript/.

[21] React wiki, DOM Virtual. Consultado en (mayo-2024) [En línea] disponible en https://www.reactjs.wiki/.

[22] React wiki, React DOM. Consultado en (mayo-2024) [En línea] disponible en https://www.reactjs.wiki/que-es-react-dom.

[23] Documentación de React Native. Consultado en (mayo-2024) [En línea] disponible en https://reactnative.dev/.

[24] Aplyca, NextJS: ¿el futuro de la web? Consultado en (junio-2024) [En línea] disponible en https://www.aplyca.com/blog/nextjs-el-futuro-web-que-es-nextjs.

[25] Youtube, CodelyTv, Por qué React dejó de recomendar Create React App. Consultado en (junio-2024) [En línea] disponible en youtube.com/watch?v=m-1X04v8jKo.

[26] Expo Docs. Documentación de Expo Consultado en (junio-2024) [En línea] disponible en https://docs.expo.dev/.

[27] Expo Docs. Documentación de EAS Consultado en (junio-2024) [En línea] disponible en https://docs.expo.dev/eas/.

[28] Expo Docs. Desarrollo de sitios web con Expo. Consultado en (junio-2024) [En línea] disponible en https://docs.expo.dev/workflow/web/.

[29] Jest Docs. Documentación de Jest. Consultado en (junio-2024) [En línea] disponible en https://jestjs.io/es-ES/docs/getting-started.

[30] Eslint Docs. Documentación de Eslint. Consultado en (junio-2024) [En línea] disponible en https://eslint.org/docs/latest/.

[31] Prettier Docs. Documentación de Prettier. Consultado en (junio-2024) [En línea] disponible en https://prettier.io/docs/en/.

[32] Documentación de Expo, Expo Go. Consultado en (junio-2024) [En línea] disponible en https://docs.expo.dev/develop/tools/#expo-go.

[33] Documentación de Sentry. Consultado en (julio-2024) [En línea] disponible en https://docs.sentry.io/.

[34] Documentación de i18next. Consultado en (julio-2024) [En línea] disponible en https://www.i18next.com/.

[35] Documentación de Supabase. Consultado en (marzo-2024) [En línea] disponible en https://supabase.com/docs.

[36] Documentación sobre Postgresql. Consultado en (abril-2024) [En línea] disponible en https://www.postgresql.org/docs/.

[37] Trello, página web oficial de Atlassian Trello. Consultado en (marzo-2024) [En línea] disponible en https://trello.com/es/tour.

[38] Pautas TFG emprendimiento, Metodología para un TFG de emprendimiento Consultado en (marzo-2024) [En línea] disponible en Teams como parte del material brindado por el tutor Patricio Letelier.

[39] The Lean Startup, Cómo crear empresas de éxito utilizando la innovación continua. Eric Ries [1 abril 2013] Editorial Deusto.

[40] Manifiesto Agile. Consultado en (julio-2024) [En línea] disponible en https://agilemanifesto.org/iso/es/manifesto.html.

[41] Normas de la ISO 25000, 25010. Consultado en (julio-2024) [En línea] disponible en https://iso25000.com/index.php/normas-iso-25000/iso-25010.

[42] Dessing Patterns. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. [31 octubre 1994] Editorial Addison Wesley.

Anexo A: Guía de la aplicación

- Inicio de sesión

Al abrir la aplicación se encontrará la ventana de inicio de sesión para acceder habrá que poseer una cuenta previamente e introducir contraseña y correo.

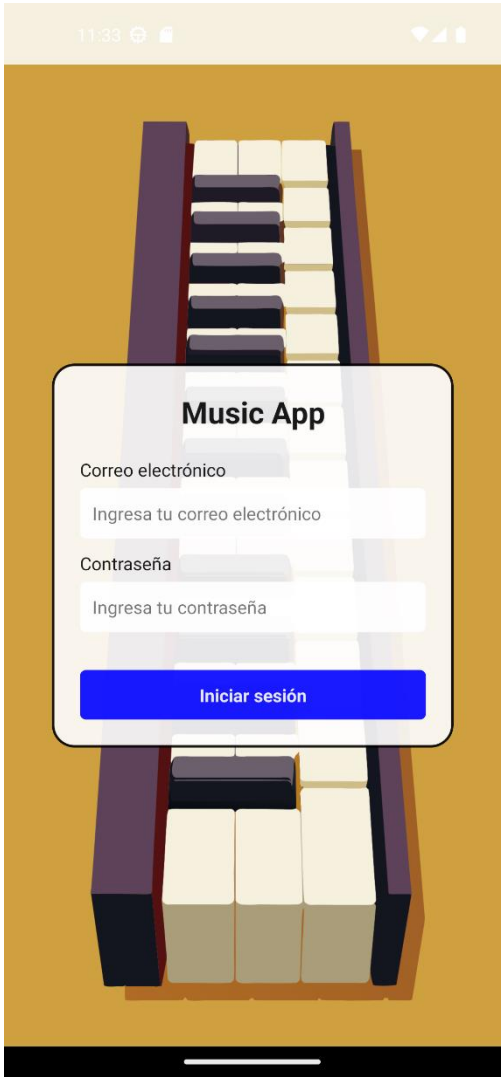

Figura 77 - pantalla de inicio de sesión

- Ventana de inicio

En esta ventana los administradores pueden observar una serie de accesos directos a funciones de la aplicación además puede observar los análisis de los datos sobre su suscripción a MusicApp. Los profesores podrán ver igualmente los accesos directos pero no los análisis, por ultimo los usuarios comunes solo puede ver las fichas de los estudiantes a su cargo.

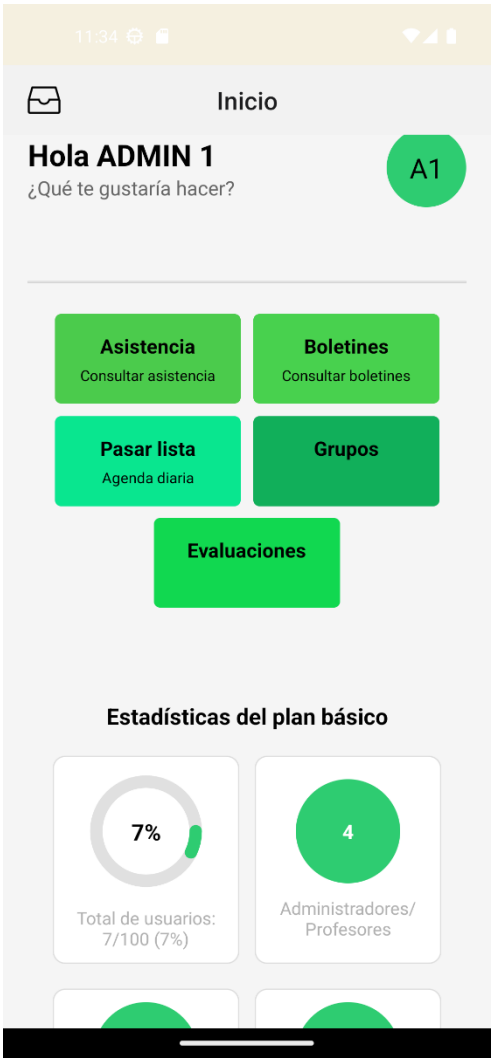

Figura 78 - pantalla de inicio

- Ventana de Gestión de usuarios

Desde esta ventana los administradores podrán crear y eliminar profesores, usuarios y otros administradores

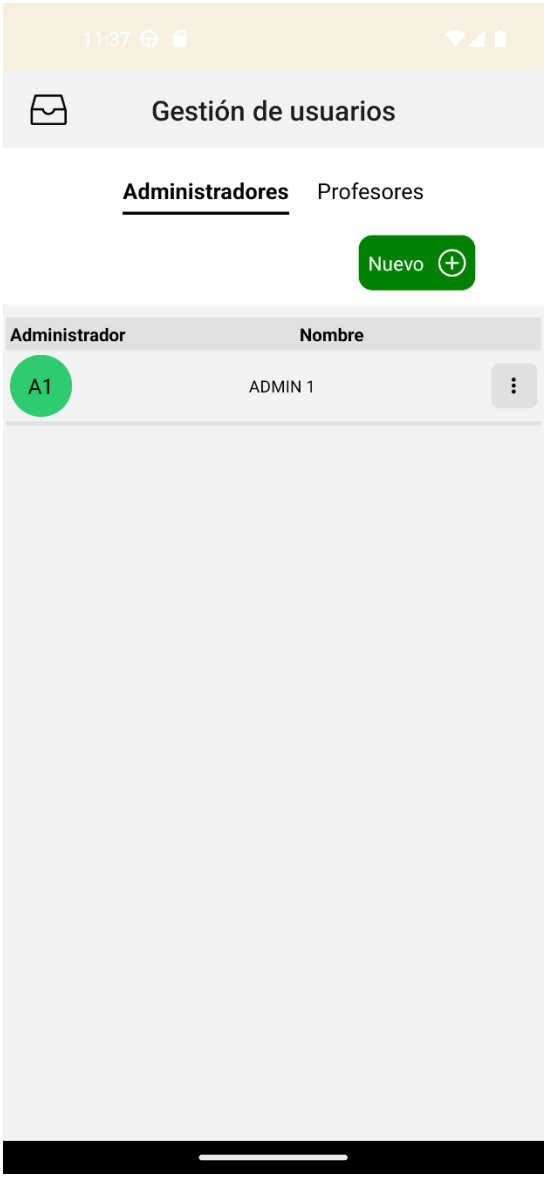

Figura 79 - ventana de gestión de usuarios

- Ventana de grupos

Desde esta ventana los administradores y profesores podrán acceder a las evaluaciones y a la agenda de los grupos a los que imparten, además los administradores podrán borrar grupos. Los grupos deben informar del horario en esta ventana.

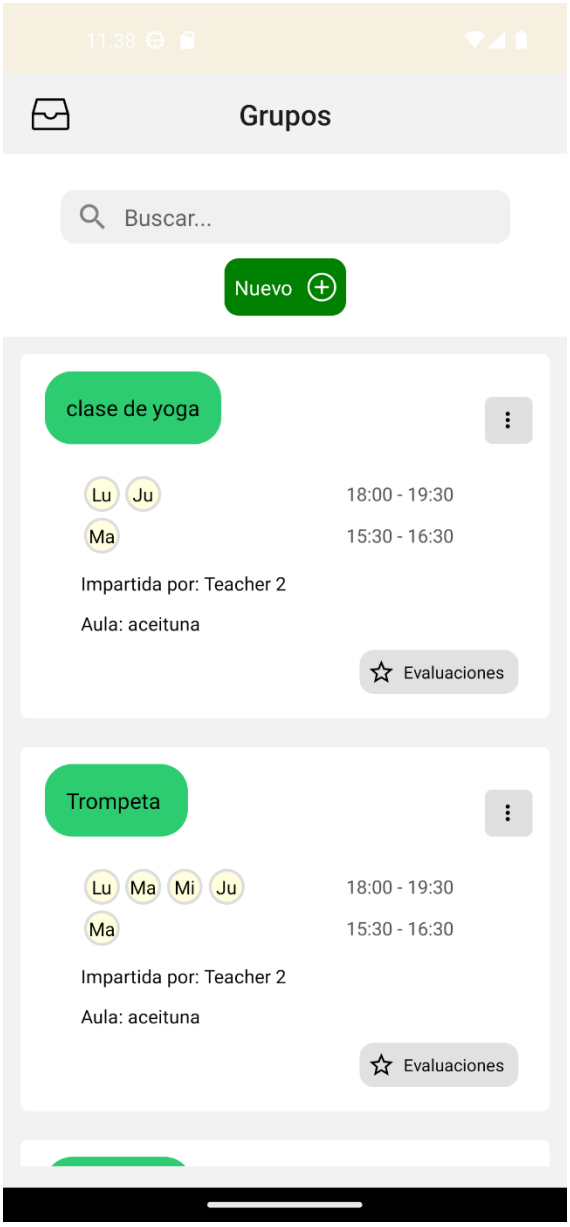

Figura 80 - ventana de gestión de grupos

- Gestión de estudiantes

Desde esta ventana se podrá consultar los estudiantes del centro, acceder a sus evaluaciones y a sus reportes de asistencia, además de gestionar sus grupos, esto últimos solo puede ser realizado por un administrador.

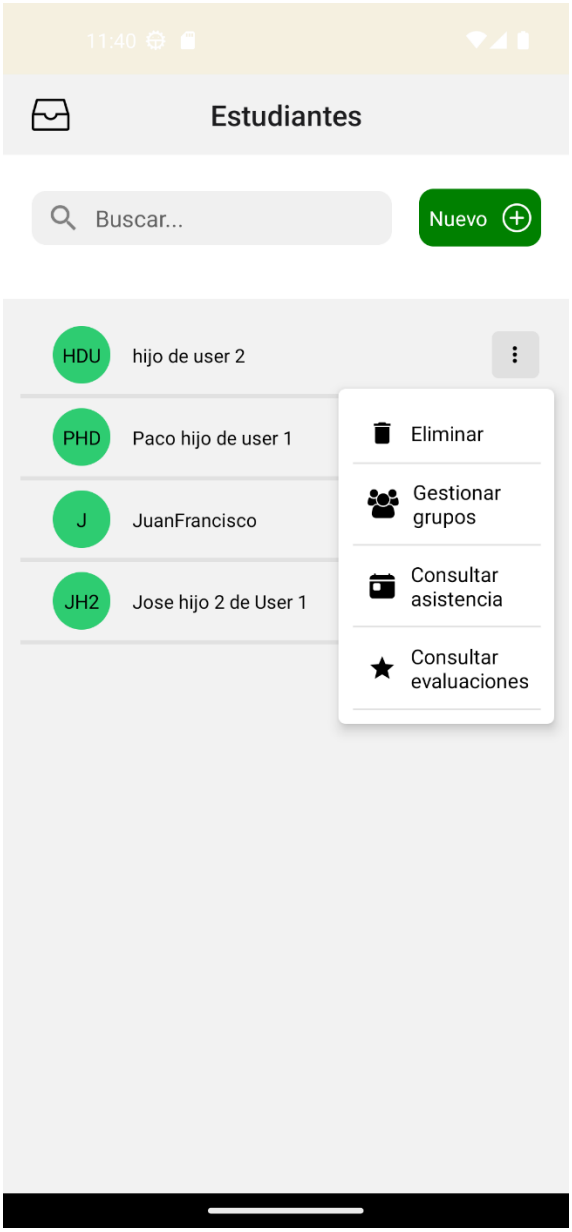

Figura 81 - gestión de estudiantes

- Ventana de asistencia

Desde esta ventana se podrá consultar la asistencia de los alumnos

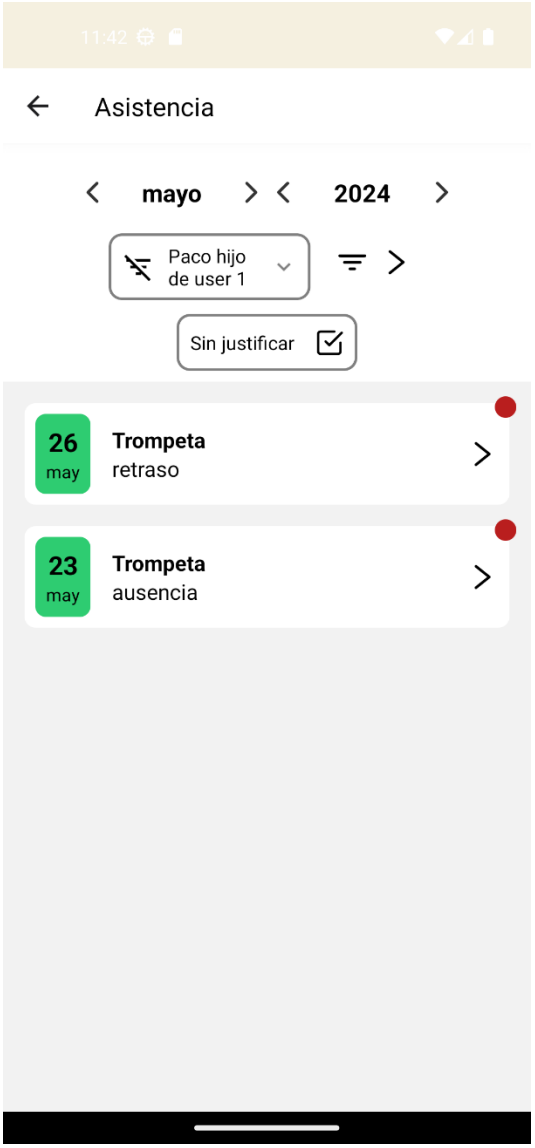

Figura 82 - asistencia de alumnos

- Ventana de evaluaciones

Desde esta ventana se podrán consultar la evaluaciones de los alumnos

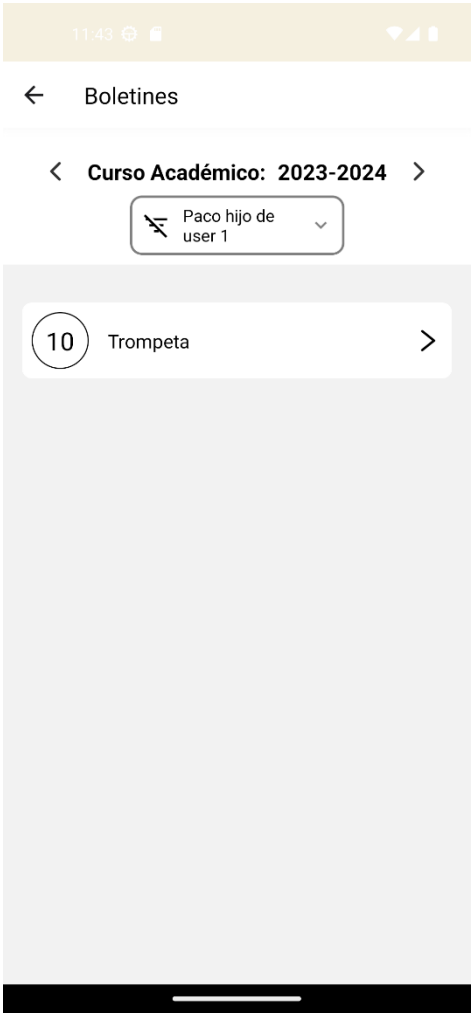

Figura 83 -evaluaciones de los alumnos

- Ventana de agenda

Desde esta venta se gestionan las faltas

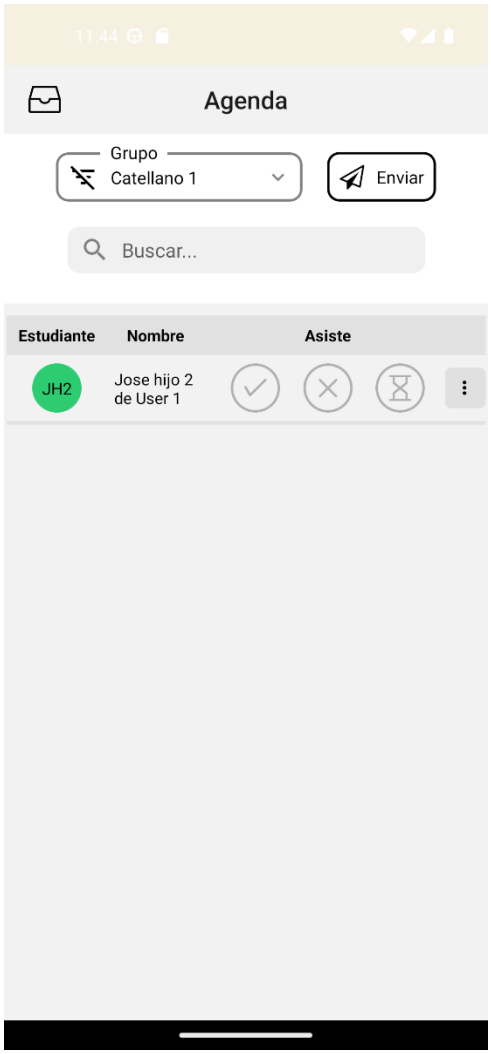

Figura 84 - ventana de agenda

- Ventana de configuración
  Desde esta venta se cambia la configuración

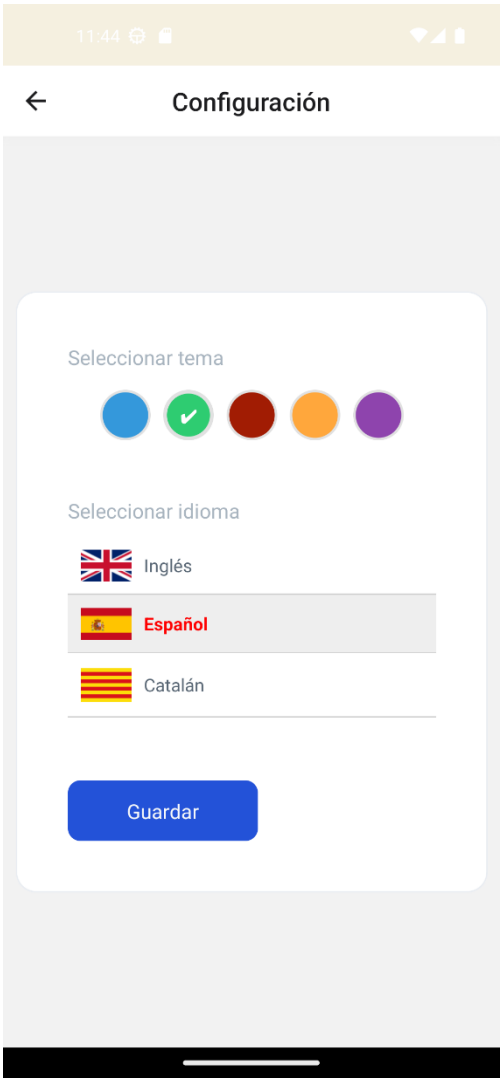

Figura 85 - ventana de configuración

Anexo B: ODS

En este anexo se presenta un tabla con el grado de relación de este proyecto con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	Alto	Medio	Bajo	No corresponde

ODS1. Fin de la pobreza				X
ODS2. Hambre cero				X
ODS3. Salud y bienestar				X
ODS4. Educación de calidad	X			
ODS5. Igualdad de género			X	
ODS6. Agua limpia y saneamiento				X
ODS7. Energía asequible y no contaminante				X
ODS8. Trabajo decente y crecimiento económico		X		
ODS9. Industria, innovación e infraestructura	X			
ODS10. Reducción de las desigualdades				X
ODS11. Ciudades y comunidades sostenibles				X
ODS12. Producción y consumo responsables				X
ODS13. Acción por el clima				X
ODS14. Vida submarina				X
ODS15. Vida de ecosistemas terrestres				X
ODS16. Paz, justicia e instituciones sólidas				X
ODS17. Alianzas para lograr los objetivos				X

Tabla 7 - Grado de relación del proyecto con los ODS

A continuación, una reflexión sobre lo presentado en la tabla 7, detallando cómo contribuye MusicApp a cada uno de los ODS con los que presenta relación.

ODS4. Educación de calidad:

MusicApp está directamente enfocada en mejorar la educación en el ámbito de las escuelas de música. Al proporcionar una plataforma digital que facilita la gestión de estas escuelas, MusicApp contribuye significativamente a la calidad de la educación impartida, haciendo que sea más accesible, organizada y efectiva.

ODS5. Igualdad de género:

Aunque la aplicación no se centra específicamente en la igualdad de género, puede contribuir de manera indirecta al brindar a todos los usuarios un uso equitativo de la plataforma, independientemente de su género. De esta manera podría ayudar a reducir las disparidades de género en el acceso a oportunidades educativas en el ámbito musical.

ODS8. Trabajo decente y crecimiento económico:

La aplicación puede impulsar el crecimiento económico en el sector de la educación musical al facilitar la gestión de las escuelas de música y crear empleos en áreas como la administración educativa, la enseñanza y el desarrollo de software. Además, al hacer más eficiente la gestión de las escuelas, estas pueden crecer y ofrecer más oportunidades laborales, contribuyendo al desarrollo económico del sector cultural.

ODS9. Industria, innovación e infraestructura:

El desarrollo de una plataforma móvil y web para la gestión de escuelas de música es un claro ejemplo de innovación tecnológica aplicada a la educación. MusicApp no solo moderniza la infraestructura educativa de las escuelas de música, sino que también promueve la digitalización y la adopción de nuevas tecnologías en la industria educativa.