PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL CON INTELIGENCIA ARTIFICIAL Y DESPLIEGUE EN UN ENTORNO MULTICLOUD PARA LA CONTRATACIÓN DE MÚSICOS EN LIMA METROPOLITANA, PERÚ

Tesis para obtener el título profesional de Ingeniero de las Telecomunicaciones

AUTOR:

Loammi Jezreel Ugaz Manayay

ASESOR:

Cesar Stuardo Lucho Romero

Lima, Junio, 2025

Informe de Similitud

Yo, **CESAR STUARDO LUCHO ROMERO**, docente de la Facultad de **CIENCIAS E INGENIERÍA** de la Pontificia Universidad Católica del Perú, asesor(a) de la tesis titulada **DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL CON INTELIGENCIA ARTIFICIAL Y DESPLIEGUE EN UN ENTORNO MULTICLOUD PARA LA CONTRATACIÓN DE MÚSICOS EN LIMA METROPOLITANA, PERÚ**, del autor **LOAMMI JEZREEL UGAZ MANAYAY**, dejo constancia de lo siguiente:

- El mencionado documento tiene un índice de puntuación de similitud de **8%**. Así lo consigna el reporte de similitud emitido por el software *Turnitin* el **16/06/2025**.
- He revisado con detalle dicho reporte y la Tesis o Trabajo de Suficiencia Profesional, y no se advierte indicios de plagio.
- Las citas a otros autores y sus respectivas referencias cumplen con las pautas académicas.

Lugar y fecha: **San Miguel, 16 de JUNIO de 2025**

Apellidos y nombres del asesor: **LUCHO ROMERO, CESAR STUARDO**	
DNI: 70326504	Firma
ORCID: **https://orcid.org/0000-0002-5749-2689**	

DEDICATORIA

A mis padres, Anthony y Beatriz, a quienes amo con toda el alma y de quienes estoy orgulloso de ser hijo. Me inculcaron valores y me dieron los consejos necesarios para enfrentar la vida.

A mi novia, Thaiz, quien presenció el desarrollo de mi tesis y me apoyó moralmente en los momentos difíciles.

A mis abuelos, Angélica y José, a quienes amo profundamente.

AGRADECIMIENTOS

Agradezco a Dios por brindarme la oportunidad de estudiar una carrera profesional y por darme la fuerza e inteligencia para llegar hasta este momento.

Agradezco a mi asesor, Mg. César Stuardo Lucho Romero, quien despertó mi interés en el mundo de la tecnología y me brindó apoyo constante en el desarrollo de mi tesis. Su ayuda fue crucial.

Agradezco a mis padres por su apoyo y preocupación por mi vida profesional, y por brindarme el apoyo moral necesario.

Agradezco a mi novia, Thaiz, por su apoyo incondicional y sus consejos en el desarrollo de la tesis.

Finalmente, agradezco a mis amigos Armando y Atilio por el ánimo para seguir adelante y no decaer ante las dificultades.

RESUMEN

La contratación de músicos en vivo en Lima presenta desafíos como la informalidad en los acuerdos y la desconfianza provocada por estafas. Estos problemas afectan tanto a músicos, que buscan oportunidades justas, como a clientes que requieren un servicio confiable y de calidad. Ante esta situación, la presente investigación propone el diseño e implementación de un aplicativo móvil con inteligencia artificial, orientado a mejorar la seguridad y facilitar la contratación de músicos en Lima, utilizando servicios en la nube y una API conectada a una base de datos.

El método de desarrollo se basa en una plataforma móvil que, mediante servicios en la nube (AWS, Google-Cloud-Platform), garantiza alta disponibilidad, seguridad y escalabilidad. Además, se integran funciones como chat en tiempo real y recomendaciones personalizadas, que mejoran la experiencia de músicos y clientes.

Como resultado, la aplicación ofrece una plataforma confiable para la contratación de músicos, garantizando la seguridad de las comunicaciones mediante un certificado SSL y permitiendo la integración con una pasarela de pago segura. Finalmente, el uso de tecnologías avanzadas y servicios en la nube facilita la resolución de problemas clave de la industria musical limeña, mejorando la transparencia y la confianza entre músicos y clientes.

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

Introducción

La tecnología y las aplicaciones móviles ofrecen una solución viable para brindar un puente factible entre músicos y clientes interesados a contratar. El objetivo principal de esta tesis es desarrollar una aplicación móvil que utilice servicios en la nube para mejorar la contratación de músicos en Lima, creando un entorno más seguro, confiable y eficiente. La solución propuesta busca facilitar la contratación de músicos y proporcionar confianza al usuario.

Entre los logros más importantes de esta tesis se encuentran el diseño e implementación de una aplicación móvil que integra inteligencia artificial para ofrecer recomendaciones personalizadas, la creación de un sistema de chat en tiempo real que mejora la comunicación entre músicos y clientes, la integración de una pasarela de pagos segura mediante un proveedor certificado, y el despliegue de la solución en un entorno *multicloud* que garantiza alta disponibilidad y escalabilidad.

En el Capítulo 1, se aborda la problemática actual de la contratación de músicos en vivo en Lima, incluyendo la formulación del problema y los objetivos. En el Capítulo 2, se desarrollan el fundamento teórico y el estado del arte relacionados con la tecnología y la industria musical. El Capítulo 3 presenta los requerimientos del sistema, la arquitectura y el diseño de la aplicación móvil. Finalmente, el Capítulo 4 describe el proceso de implementación, pruebas de funcionamiento y análisis de costos.

La tesis finaliza con la presentación de las conclusiones, orientadas al cumplimiento del objetivo general y los objetivos específicos, y con recomendaciones finales para futuras mejoras y desarrollos.

Capítulo1. Problemática de la contratación de músicos en vivo en Lima

En este primer capítulo se analiza la problemática de la contratación de músicos en vivo en Lima, partiendo de la importancia cultural de la música peruana y su diversidad. Se examinan los retos que enfrentan los músicos, incluyendo la informalidad contractual y la desconfianza generada por fraudes y estafas en el sector. Para contextualizar, se presentan casos de bandas peruanas afectadas y se describe el impacto de la transformación digital en la industria musical local. Finalmente, se formula el problema central de la tesis, se establecen los objetivos, se identifican los beneficiarios, y se expone la justificación desde la perspectiva de las telecomunicaciones y la motivación personal que impulsa esta investigación.

1.1 La música en el Perú

La música, según la Real Academia Española (RAE), se define como el arte de combinar sonidos, ya sea de la voz humana o de instrumentos, para generar melodía, ritmo y armonía combinados en busca del deleite auditivo [1]. A lo largo de la historia, la música ha sido característica fundamental en cada cultura, ya que se utiliza no solo para compartir historias, sino también para expresar su identidad cultural y una actividad social intrínseca. Por tanto, la

música puede ser más que solo notas musicales ordenadas sistemáticamente para generar un sonido agradable, pues también es un vehículo para la construcción de identidades culturales, expresión de emociones y conexión entre personas de diversas comunidades [2][3].

La música peruana se distingue por la expresión de una alta gama de emociones, como alegría, melancolía e ilusión, a través de una variada instrumentación que incluye instrumentos de viento, cuerdas y percusión. Esta riqueza musical es el fruto del encuentro entre la herencia del imperio Incaico, la influencia española y la contribución de los esclavos africanos. En un inicio, la música peruana fue estudiada y clasificada por sus géneros denominados como "tradicionalmente representativos", de los cuales forman parte la música criolla, andina y afroperuana. Estos géneros reflejaban la diversidad cultural y étnica del Perú, fusionando elementos indígenas, europeos y africanos en una expresión musical única [4].

Sin embargo, a medida que la globalización impactó a la música peruana y se produjeron nuevos intereses influenciados por música extranjera, surgieron nuevos estilos musicales representativos. Algunos de los principales estilos que surgieron en este contexto fueron el rock peruano, la salsa, la cumbia, la música chicha, el hip-hop y el reguetón.

1.2 El mercado musical en la era digital

En la actualidad, la música experimenta una revolución marcada por avances tecnológicos y la creciente multiculturalidad de todas las regiones del mundo. La manera en que la música se crea, comparte y disfruta ha evolucionado significativamente en las últimas décadas. Cada año, la industria musical mueve miles de millones de dólares. Según la Federación Internacional de la Industria Fonográfica (IFPI), el valor de los ingresos globales de la música grabada alcanzó la asombrosa cifra de US$26.2 mil millones en el 2022, marcando el octavo año consecutivo de crecimiento, donde las plataformas *streaming* de música representan el 67% de estos ingresos [5]. Ejemplos populares de estas plataformas *streaming* son aplicaciones móviles

como Apple Music o Spotify, donde esta última actualmente superó los 500 millones de usuarios [6].

1.2.1 La música en vivo en el mundo

La innegable presencia de la música en la vida de las personas se observa en los reportes mundiales de música en vivo de cada año. Pollstar, fuente conocida por la industria de la música por dedicarse a la recopilación y el análisis de datos relacionados con conciertos y eventos a nivel global, genera sus reportes de ingresos basados en los tipos de lugares, donde se realizan los eventos en vivo (anfiteatros, arenas, clubes, estadios y teatros) [7].

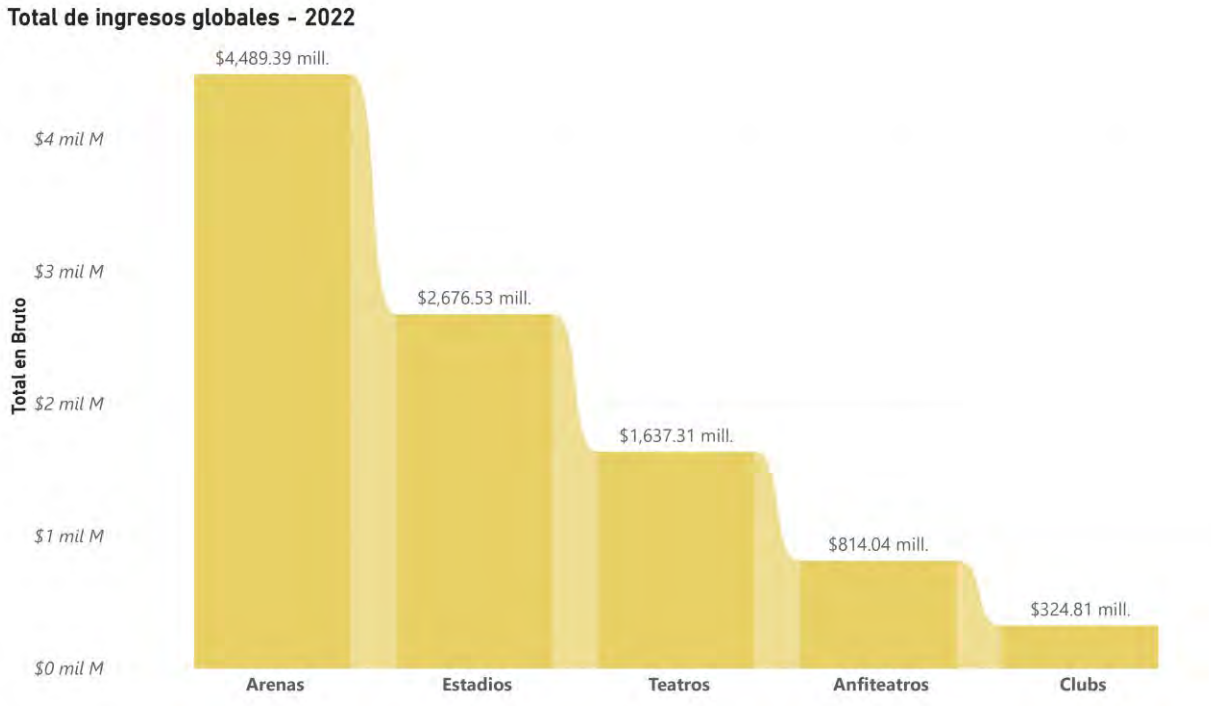


Figura 1. Total de ingresos globales por música en vivo según el tipo de lugar
Fuente: Elaboración propia con datos tomados de [7]

Como se observa en la anterior figura, la Arena produjo las mayores fuentes de ingreso con US$4.5 mil millones en 2022, seguido de los Estadios con US$2.7 mil millones y los Teatros con US$1.6 mil millones.

Según datos de *Billboard Boxscore*, una de las fuentes más reconocidas por el seguimiento de ingresos de giras musicales a nivel global, la industria de la música en vivo continúa siendo una de las más rentables del entretenimiento. En 2022, artistas como Bad Bunny, Elton John, Ed Sheeran, Harry Styles y Coldplay lideraron el ranking de giras con mayores ingresos, alcanzando cifras que superan los 200 millones de dólares cada uno [8]. Destaca el caso de Bad Bunny, quien encabezó la lista con más de 373 millones de dólares en ingresos y más de 1.8 millones de asistentes en tan solo 65 conciertos.

Como muestran los datos revisados, los eventos en vivo representan una fuente significativa y en crecimiento dentro del modelo económico musical. Más allá del sustento que brindan a artistas y promotores, estos espectáculos impulsan el dinamismo del mercado musical en cada país, aportando no solo ingresos directos, sino también valor cultural, empleo y desarrollo económico local.

1.2.2 La música en vivo en el Perú

Según una publicación del 14 de enero de 2021 de Price-Waterhouse-Coopers (PwC) Perú [9], los ingresos de la industria musical en el país han sido significativamente inferiores en comparación con otros mercados de la región. En 2019 se registraron 37 millones de dólares, cifra que apenas varió en 2021 con 38 millones debido al impacto de la pandemia. Para 2024, los ingresos aumentaron a aproximadamente 54 millones de dólares, aunque el Perú sigue ubicándose como el de menor recaudación dentro de los países considerados en esta comparación.

País	2019	2021	2024
Perú	$37,000,000	$38,000,000	$54,000,000
Chile	$101,000,000	$101,000,000	$133,000,000
Argentina	$105,000,000	$121,000,000	$188,000,000
Colombia	$119,000,000	$102,000,000	$148,000,000
México	$443,000,000	$404,000,000	$546,000,000
Brasil	$578,000,000	$686,000,000	$872,000,000
Total	**$1,383,000,000**	**$1,452,000,000**	**$1,941,000,000**

Figura 2. Ingresos por la música de Brasil, México, Colombia, Argentina, Chile y Perú durante el 2019, 2021 y 2024
Fuente: Elaboración propia con datos tomados de [9]

No obstante, a pesar de estas cifras generales aún modestas, se ha evidenciado un repunte importante en la actividad de música en vivo en el país. El 11 de octubre del 2022, el diario El Comercio publicó que la Asociación Peruana de Autores y Compositores (APDAYC) reportó un notable crecimiento en los mega eventos musicales, con ingresos superiores a los S/300 000 por evento a nivel nacional. Hasta septiembre de ese año, se registraron 56 eventos, donde artistas internacionales como Rauw Alejandro y Karol G reunieron entre 20 000 y 30 000 asistentes por presentación. Asimismo, el cantautor peruano Gian Marco superó los 40 000 asistentes en el concierto por sus 30 años de trayectoria. Marina Marcovich Adrianzén, directora nacional de recaudación de APDAYC, atribuye este aumento a la fuerte demanda contenida por parte del público tras dos años de restricciones por la pandemia [10].

1.3 Músicos en Lima

Las metas de los músicos son variadas: desde alcanzar el reconocimiento local o internacional, hasta simplemente disfrutar del arte sin aspirar a la fama. Sin embargo, todos enfrentan un reto común: abrirse camino en un entorno que muchas veces ofrece pocas garantías. En Lima, como en muchas otras ciudades, los músicos lidian con barreras para consolidar su carrera y encontrar espacios donde compartir su arte.

1.3.1 Inicios de bandas exitosas en el Perú: De los bares a la Fama

El camino hacia el reconocimiento musical ha estado tradicionalmente vinculado a la exposición en escenarios en vivo. Un ejemplo clásico es The Beatles, una de las bandas más influyentes en la historia de la música desde los años 60. Su éxito no fue inmediato: comenzaron tocando en pequeños bares de Liverpool, donde desarrollaron su estilo y ganaron visibilidad hasta consolidarse como un fenómeno mundial [11].

Estas experiencias iniciales de bandas icónicas sirven como inspiración para muchas agrupaciones emergentes. En Lima, por ejemplo, bandas como Los Saicos, considerados pioneros del punk en Perú, también comenzaron ensayando en casas familiares, hasta que una presentación en un festival atrajo la atención de los medios [12][13]. De forma similar, Amén surgió como una comunidad de músicos que se reunía regularmente hasta consolidarse en 1995. Su proyección despegó tras participar en concursos televisivos que les permitieron grabar su primer álbum en 1997 [14].

Estos casos muestran que el desarrollo artístico y la formación de seguidores dependen de la música en vivo y el uso de espacios físicos, tanto en el extranjero como en el Perú. Sin embargo, en Lima, esta actividad se enfrenta a un alto nivel de informalidad que limita las oportunidades y condiciones laborales de los músicos.

1.3.2 La informalidad para los músicos en Lima

En su tesis “El músico clásico en el Perú: entre la vocación y la profesión”, Diana Luz Montes Álvarez distingue entre trabajos fijos y “chivos” para los músicos clásicos [15]. Los “chivos” son contratos temporales, típicos en eventos con música en vivo como bodas, fiestas patronales, funerales, y aniversarios.

Aunque el término surge en el ámbito clásico, estas prácticas también aplican a otros músicos, como The Beatles o Los Saicos, quienes empezaron con presentaciones no fijas en bares o eventos pequeños, equivalentes a “chivos”. Montes, a partir de entrevistas en Lima, señala que estas oportunidades no se obtienen mediante convocatorias públicas, sino principalmente por recomendaciones personales.

A pesar de la dinámica que ofrece el mercado de música en vivo, la mayoría de músicos opera en un entorno informal. Esta informalidad genera obstáculos para quienes buscan estabilidad y crecimiento profesional, debido a la falta de estructura y la dependencia de redes personales para acceder a contratos.

1.4 Desafíos de confianza en la contratación de servicios

Los músicos en Lima enfrentan múltiples desafíos que afectan su desarrollo profesional, entre los cuales destacan las estafas, una amenaza creciente en el país. Este fenómeno impacta no solo a los artistas, sino también a quienes buscan contratar servicios musicales, generando un clima de desconfianza que dificulta las relaciones laborales y comerciales en el sector.

1.4.1 Estafas en el Perú

La estafa, definida como un acto de engaño o fraude, tiene implicancias legales significativas. Según la Real Academia Española (RAE), la estafa constituye un delito que busca causar perjuicio patrimonial a una persona mediante el uso de engaños con el fin de obtener beneficios económicos [16]. En el año 2021, en el Perú, se detallan las denuncias registradas por la Policía Nacional del Perú (PNP) a nivel nacional [17], destacando Lima como la región con el mayor volumen según el Anuario Estadístico Policial - 2021.

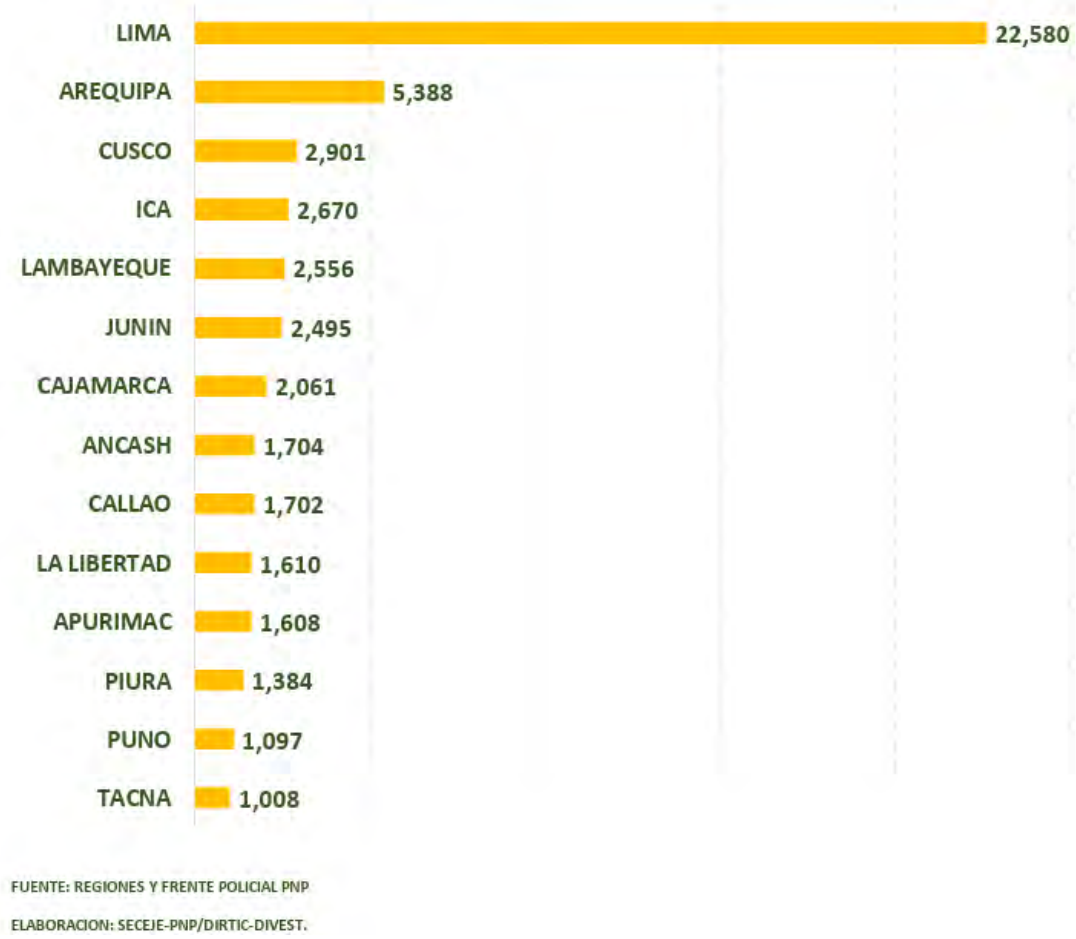


Figura 3. Departamentos con mayor número de denuncias a nivel nacional. Año 2021
Fuente: Imagen tomada de [17]

Asimismo, se destaca que se registraron un total de 224 291 denuncias por delitos contra el patrimonio, dentro de las cuales 15 183 corresponden a casos de estafas y otras defraudaciones durante el año 2021. Si comparamos estas 15 183 denuncias con las 8 915 del año 2020, se evidencia un crecimiento de aproximadamente un 70.5%. Este significativo aumento es una señal alarmante de que las estafas y defraudaciones están experimentando un incremento preocupante en el Perú.

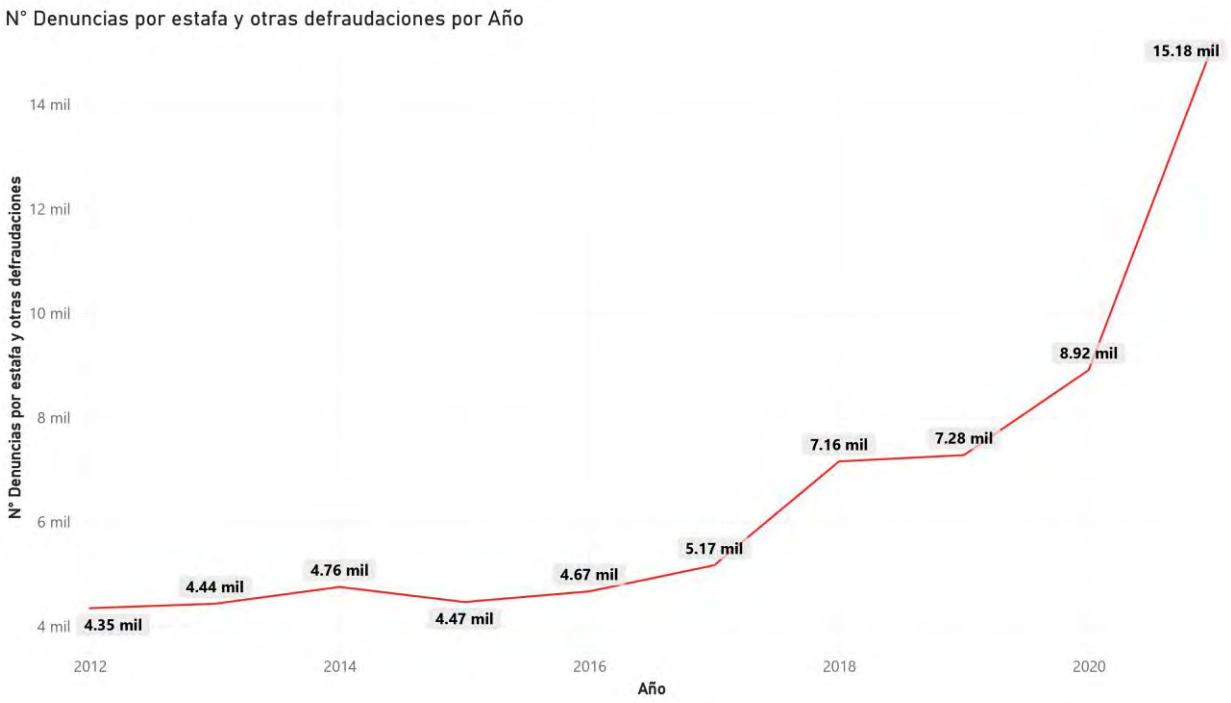


Figura 4. *N° de denuncias por estafa y otras defraudaciones. Año 2012 - 2021*
Fuente: Elaboración propia con datos tomados de [17]

1.4.2 Estafas en el ámbito musical

En la sección anterior, se examinaron las estafas en el ámbito general en el Perú, evidenciando un crecimiento alarmante en los delitos contra el patrimonio, especialmente en Lima. Ahora, nos centraremos en un aspecto más específico de este problema: las estafas que han permeado el ámbito musical.

Entre enero y mayo de 2023, la Policía Nacional del Perú (PNP) registró 3 410 denuncias por estafas en Lima Metropolitana, según informes de la Agencia Peruana de Noticias ANDINA [18]. Estas denuncias abarcan diversas modalidades de estafa, como la falsificación de entradas para partidos de fútbol [19], esquemas piramidales [20], eventos falsos o contratos de shows fraudulentos [21], entre otros. Este tipo de estafas no solo afectan a sectores específicos, sino que también se extienden al ámbito musical, un fenómeno que se explorará en esta sección.

En 2020, Grupo 5, reconocida banda de cumbia y merengue peruana, fue víctima de una estafa cuando cuentas falsas en Facebook promocionaron la venta de entradas fraudulentas para un concierto junto a Josimar. Fanáticos denunciaron públicamente el engaño con mensajes como: “Me parece una estafa total que aún no me acepten en el grupo [...] ni siquiera contestan los mensajes” o “¿Cómo puedo ingresar? ya hice el pago” [22]. Este caso no fue aislado. En 2016, la banda Ráfaga denunció que promotores ofrecían conciertos falsos en Huaral usando su nombre, mientras ellos se encontraban de gira en Argentina [23]. Más recientemente, en 2022, varias bandas se retiraron del Festival Perú Central alegando fraude y desorganización, y anunciaron su decisión en redes sociales [24]. Estos casos evidencian un patrón de estafas en eventos musicales, que perjudica tanto a artistas como al público.

Adicionalmente, las redes sociales se han convertido en un escenario propicio para las estafas en el ámbito musical. Publicaciones fraudulentas que prometen eventos, contratos y colaboraciones han aumentado, generando insatisfacción por parte de la comunidad musical en Lima. Músicos y clientes afectados utilizan estas mismas plataformas para denunciar públicamente tales prácticas, como se observa en la siguiente figura, evidenciando una red de alerta entre la comunidad.

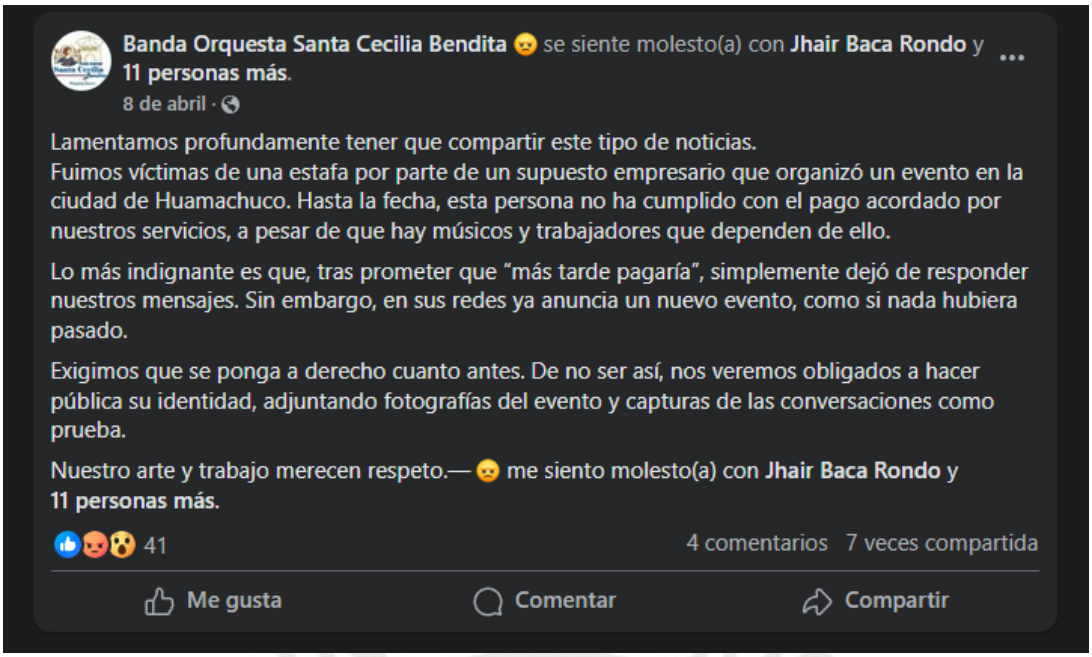


Figura 5. *Ejemplo de denuncia pública sobre estafa en el ámbito musical*
Fuente: [25]

La repetición de estos incidentes ha provocado un aumento de la desconfianza en el proceso de contratación de servicios musicales, afectando tanto a músicos como a clientes. Este fenómeno genera consecuencias económicas y daña la reputación y la integridad del sector musical en las localidades afectadas. Estas situaciones reflejan problemas persistentes en la confianza y formalidad dentro de la industria musical en Lima, que afectan las relaciones entre quienes ofrecen y quienes buscan servicios musicales.

1.5 Formulación del Problema

En el complejo escenario de la contratación de músicos en Lima, se presentan desafíos significativos tanto para los artistas, que buscan oportunidades justas y reconocimiento, como para los clientes, que enfrentan dificultades para encontrar servicios musicales confiables y de calidad. La informalidad en los acuerdos, frecuentemente basados en mensajes de texto o acuerdos verbales sin estructura formal, sumada a la amenaza constante de estafas en entornos

digitales y presenciales, genera un clima de desconfianza que afecta directamente la trayectoria profesional de los músicos y la experiencia de los contratantes.

Esta problemática se ve agravada por antecedentes de fraudes en eventos musicales, donde tanto músicos como clientes han sido víctimas de prácticas engañosas. Aunque las redes sociales son herramientas clave para la promoción, también han facilitado la proliferación de estafadores, incrementando la insatisfacción y desconfianza dentro de la comunidad musical limeña.

Por ello, se identifica la necesidad de desarrollar un aplicativo especializado para la contratación de servicios musicales, que permita establecer un entorno más seguro, confiable y eficiente en Lima Metropolitana.

1.6 Objetivos

La presente tesis los siguientes objetivos:

1.6.1 Objetivo General

Diseñar e implementar un aplicativo móvil con inteligencia artificial destinado a brindar mayor seguridad y facilitar la contratación de músicos en vivo en Lima, aprovechando servicios en la nube y utilizando una API (*Application Programing Interface*) conectada a una base de datos.

1.6.2 Objetivos Específicos

- Diseñar e implementar una aplicación móvil para la contratación de músicos en Lima.
- Implementar servicios *backend* seguros y eficientes en un entorno *multicloud,* incluyendo la gestión de pagos.
- Incorporar un sistema de chat en tiempo real para facilitar la comunicación entre usuarios y músicos.

- Implementar un sistema de recomendaciones personalizado para cada cliente, utilizando inteligencia artificial.

1.7 Beneficiarios

El presente proyecto beneficiará a distintos actores dentro del ecosistema musical en Lima.

- **Beneficiarios directos:** Los músicos que buscan oportunidades laborales seguras y transparentes, ya que podrán acceder a una plataforma confiable que facilite la contratación de sus servicios.
- **Beneficiarios indirectos:** Los organizadores podrán contar con un sistema que simplifique la contratación de artistas, mientras que el público disfrutará de eventos con mayor calidad y profesionalismo.

1.8 Justificación desde la perspectiva de las telecomunicaciones

Desde la perspectiva de Ingeniería de las Telecomunicaciones, este proyecto aborda diversos desafíos tecnológicos mediante la integración de servicios en la nube, la optimización de la comunicación entre músicos y clientes, y la gestión eficiente de recursos digitales.

- **Infraestructura de redes y comunicación:** Se emplean plataformas *Platform as a Service* (PaaS) y *Backend as a Service* (BaaS) para garantizar escalabilidad, alta disponibilidad y tiempos de respuesta óptimos, asegurando así una comunicación fluida entre los usuarios del sistema.
- **Sistemas de distribución de contenidos y servicios digitales:** Implementación de inteligencia artificial para recomendaciones, repositorio de videos y fotos, chat en tiempo real y pasarela de pagos segura.
- **Gestión de proyectos:** Análisis de costos, planificación estratégica y administración de recursos para asegurar la viabilidad técnica y económica del sistema.

1.9 Motivación personal

La presente tesis se centra en el desarrollo de un aplicativo móvil especializado para la contratación de músicos en vivo, un proyecto motivado por la pasión por la música y las vivencias acumuladas en este ámbito. Diversos retos, como la informalidad prevaleciente en las contrataciones y la dificultad de conectar músicos con potenciales clientes de manera segura y eficiente, subrayan la necesidad de una solución innovadora. Desde la perspectiva del autor, este aplicativo no solo es necesario, sino que representa una oportunidad para revolucionar la interacción entre artistas y clientes, proporcionando un marco más profesional y estructurado para la industria.

Esta propuesta no se limita a responder a una necesidad local; su diseño y funcionalidad tienen el potencial de ser adaptados y aplicados en otras ciudades o incluso países que enfrentan desafíos similares, fomentando así el profesionalismo en la carrera musical a un nivel más amplio. Con este aplicativo, se aspira a impactar significativamente en la industria musical de Lima, ofreciendo soluciones tecnológicas que no solo faciliten el acceso a la música en vivo, sino que también brinden una capa adicional de seguridad y confianza para los involucrados.

Esta tesis es, por tanto, un esfuerzo para enriquecer la cultura musical, mejorando las condiciones de trabajo para los músicos y la calidad de servicio para los aficionados de la música.

Capítulo 2. Marco teórico y estado del arte

En este capítulo, se presentan los fundamentos teóricos esenciales para la formulación y desarrollo de la solución tecnológica propuesta, que incluye tanto la aplicación móvil como su infraestructura en la nube. Además, se analiza el estado del arte de las soluciones disponibles actualmente para enfrentar los desafíos relacionados con la contratación de servicios musicales.

2.1 Marco teórico

2.1.1 Sistemas operativos para móvil

Desde la creación de los primeros dispositivos móviles, la función fundamental de un sistema operativo (OS) ha sido la gestión eficiente de los recursos del dispositivo y la facilitación de la interacción entre el *hardware* y el *software*. Los dispositivos móviles, como *smartphones* y *tablets*, dependen de sistemas operativos especializados [26].

Según la información recopilada por StatCounter en noviembre de 2023, Android presenta una sólida cuota de mercado del 70.19%, consolidándose como el sistema operativo líder a nivel mundial. Por su parte, iOS, la contraparte de Apple, se sitúa en un respetable 29.12%. Este análisis revela la predominancia de estas dos plataformas en el escenario global de sistemas

operativos móviles, dejando un margen del 0.69% destinado a sistemas operativos minoritarios, como KaiOS y Samsung, entre otros [27].

- **Android OS:** Es un sistema operativo de código abierto basado en Linux, utilizando su *kernel* para la gestión de subprocesos y memoria. La Capa de Abstracción de Hardware (HAL) actúa como puente entre el *hardware* del dispositivo y las aplicaciones, facilitando el uso de funciones específicas del *hardware*. Las bibliotecas nativas C/C++ son la base de componentes clave como ART y HAL, accesibles a través de la API de Java. Android incluye aplicaciones del sistema como correo electrónico y SMS [28]. La arquitectura del sistema operativo Android se presenta en el Anexo 1.
- **iOS:** Es el sistema operativo de Apple para dispositivos móviles, basado en Darwin, un sistema operativo Unix desarrollado por Apple. A diferencia de Android, iOS no se basa en Linux. Las aplicaciones en iOS no interactúan directamente con el *hardware*, sino a través de interfaces definidas por el sistema operativo. La capa "Cocoa Touch" proporciona la infraestructura de las aplicaciones, "Media Layer" gestiona gráficos, audio y video, "Core Services" ofrece servicios esenciales como redes y ubicación, y "Core OS" interactúa con el *kernel* de Darwin, incluyendo bibliotecas esenciales [29]. Para más detalle sobre la arquitectura de iOS, consulte el Anexo 2.

2.1.2 Desarrollo de aplicaciones móviles

Según Microsoft, una aplicación móvil es un software diseñado para dispositivos portátiles como *tablets* y *smartphones*. Incluye software nativo, sistemas operativos, plataformas y lenguajes que soportan su funcionamiento [30]. En la clasificación de Microsoft, las aplicaciones se dividen en nativas, multiplataforma e híbridas. En la siguiente tabla, se realiza una comparación sobre estos tipos.

Tabla 1. Comparativa de tipos de aplicaciones móviles

Detalles	Aplicaciones Nativas	Aplicaciones Multiplataforma	Aplicaciones Híbridas
Número de Codebases	Una por plataforma	Una única, pero compilada para cada plataforma	Una para la aplicación, otra para el contenedor
Lenguajes y Frameworks	Native only	Team's choice	Web and native
Acceso a SDKs y APIs	Sí	Sí	Limitado
Rendimiento	La más alta	Alta	Baja
Acceso al Hardware del Dispositivo	Completo	La mayoría	Alguno
Respuesta a la Entrada del Usuario	Buena	Buena	Pobre
Uso de Recursos del Dispositivo	Alto	Alto	Medio
Requiere Conectividad	No	No	Sí
Costo de Construcción y Mantenimiento	El más alto	Alto	Bajo
Ubicación de Almacenamiento de la Aplicación	En el dispositivo	En el dispositivo	En el dispositivo y en el servidor
Desplegado a Través de	Marketplace	Marketplace	Marketplace

Fuente: Elaboración propia con datos tomados de [30]

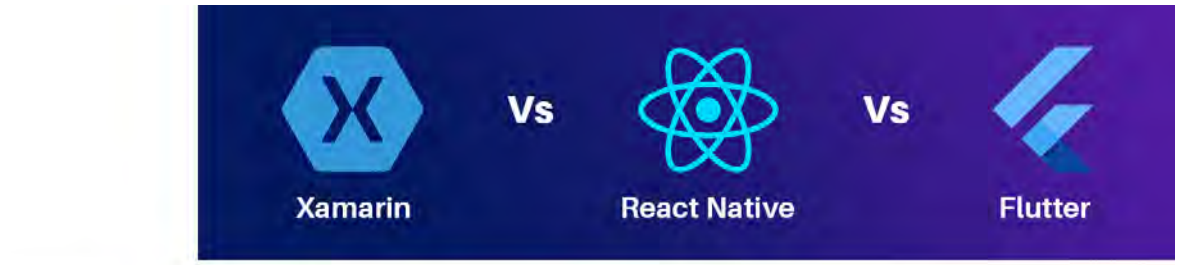
2.1.3 *Frameworks* para el desarrollo móvil

Un *framework* se define como un conjunto predefinido de herramientas, reglas y convenciones que proporciona una estructura para facilitar el desarrollo de *software*. Su función principal es agilizar y simplificar el proceso de codificación al ofrecer soluciones comunes para tareas recurrentes [31].

Esta sección analiza *frameworks* para el desarrollo de aplicaciones móviles multiplataforma, comparando Flutter, Xamarin y React Native.

- **Flutter:** Lanzado por Google en 2017, Flutter es un *framework* multiplataforma compatible con iOS, Android, Web, Windows, MacOS y Linux. Utiliza el lenguaje Dart y ofrece rendimiento casi nativo al compilarse en código máquina. Flutter se destaca por su enfoque en widgets para construir interfaces de usuario coherentes y su motor gráfico Skia para consistencia visual. Herramientas como recarga en caliente y un inspector de widgets facilitan el desarrollo. Flutter cuenta con una comunidad activa y numerosos paquetes adicionales en *pub.dev* [32].
- **Xamarin:** Xamarin es un *framework* de código abierto que permite desarrollar aplicaciones nativas para iOS, Android y Windows mediante .NET y C#, utilizando Visual Studio [33]. Incluye componentes como Xamarin.Android, Xamarin.iOS y Xamarin.Essentials, ofreciendo rendimiento similar a las aplicaciones nativas. Sin embargo, enfrenta desafíos como una comunidad más pequeña comparada con React Native y Flutter, problemas de compatibilidad con algunas librerías y costos elevados para usar Visual Studio profesionalmente [34].
- **React Native**: React Native, desarrollado por Meta en 2015, es un framework de código abierto para crear aplicaciones nativas en iOS y Android usando JavaScript y React [35]. Con un único código base, combina Código Nativo, JavaScriptCore VM y React Native Bridge, permitiendo ejecutar JavaScript y conectarse a las APIs nativas y componentes de cada plataforma. Esta arquitectura asegura una experiencia de desarrollo eficiente y aprovecha las características nativas [36].

Finalmente, un resumen comparativo de estas tres tecnologías se presenta en la siguiente figura.



Year introduced	2011	2015	2018
Backed by	Microsoft	Facebook	Google
Presentation language	XAML and/or xamarin.forms	Proprietary but looks like JSX	Dart
Procedural Language	C#	JavaScript	Dart
Still need to know some truly native development	Very high	High	Low
Recent popularity trend	Slightly decreasing	Increasing	Increasing
Cost	Teams of > 5 must buy a license to Visual Studio	Free	Free

Figura 6. Comparación entre frameworks: Flutter, React Native y Xamarin
Fuente: Imagen tomada de theonetechnologies.com

2.1.4 Bases de datos

Una base de datos es una colección organizada de datos estructurados almacenados electrónicamente. Un Sistema de Gestión de Bases de Datos (DBMS) permite el

almacenamiento, manipulación y acceso eficiente a estos datos [37]. Juntos, los datos, el DBMS y las aplicaciones asociadas forman un sistema de bases de datos.

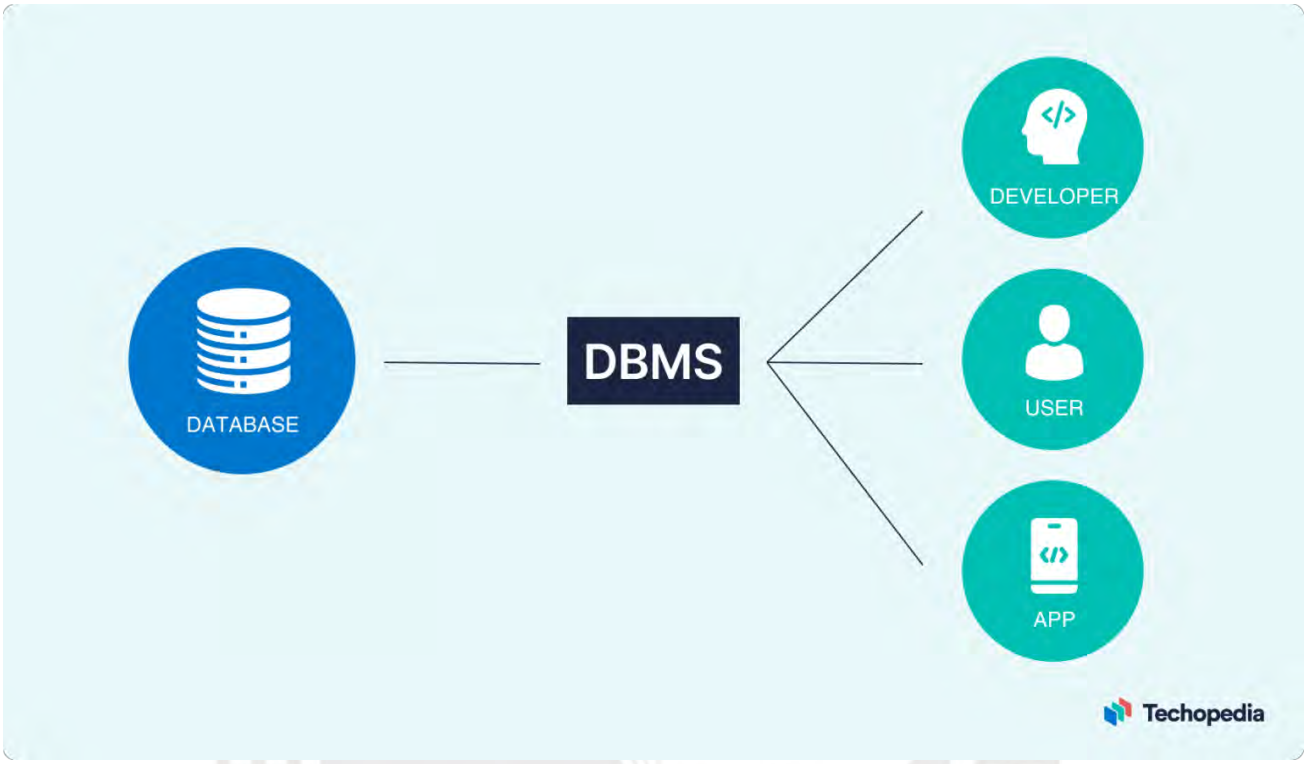


Figura 7. Base de datos, DBMS y usuarios finales
Fuente: Imagen tomada de www.techopedia.com

Base de datos SQL:

Una base de datos SQL *(Structured Query Language)* es un sistema de gestión de bases de datos relacional que utiliza el lenguaje SQL para definir y manipular los datos. En este tipo de bases de datos, la información se organiza en tablas, donde cada tabla consiste en filas y columnas. Cada fila representa un registro individual y cada columna, un atributo específico. Las bases de datos SQL permiten realizar consultas complejas, actualizaciones y operaciones sobre los datos utilizando instrucciones SQL estándar. Ejemplos populares de sistemas de gestión de bases de datos SQL incluyen MySQL, PostgreSQL y SQLite [38].

- **MySQL:** MySQL es una de las bases de datos relacionales de código abierto más utilizadas en el mundo, ampliamente adoptada en aplicaciones web y plataformas de redes sociales. Simplifica la gestión y visualización de relaciones entre datos, facilitando la manipulación eficiente de la información. Su condición de software de código abierto bajo la Licencia Pública General de GNU ha propiciado variantes como MariaDB y Percona Server for MySQL [39].
- **SQLite**: Diseñado por D. Richard Hipp en 2000, SQLite es una base de datos de dominio público, escrita en C, que combina "SQL" con "lite" para indicar su ligereza. Utilizado en sistemas operativos y programas como Android, iOS, Windows Phone, Skype, y navegadores como Chrome, Firefox y Safari, SQLite no requiere software de servidor adicional. Su pequeño tamaño y capacidad para almacenar la base de datos en un único archivo lo hacen único [40].
- **PostgreSQL:** PostgreSQL, es un sistema de gestión de bases de datos objeto-relacional (ORDBMS) de código abierto que destaca por su capacidad para gestionar datos relacionales y no relacionales con eficiencia. Su versatilidad e integridad lo hacen ideal para sectores como servicios financieros, fabricación, comercio minorista y logística. PostgreSQL es una opción robusta y flexible para diversas industrias [41].

Es importante destacar que MySQL y PostgreSQL son opciones más robustas cuando en función de almacenar grandes volúmenes de datos y ejecutar consultas en la base de datos. A continuación, una comparación visual entre estas dos alternativas:

Tabla 2. Comparativa entre MySQL y PostgreSQL

Categoría de Diferencia	MySQL	PostgreSQL
Tipos de Datos	Soporta menos tipos de datos	Soporta más tipos de datos
Sensibilidad a Mayúsculas	No	Sí
UTF-8	Requiere conversión	No requiere conversión
Sentencia Condicional	Funciones IF() y NULLIF()	Sentencia CASE WHEN
DROP CASCADE	No	Sí
DROP TEMPORARY TABLE	Sí	No
TRUNCATE	Sí, solo TRUNCATE	Sí, con opciones mejoradas
FULL OUTER JOIN	No	Sí
INTERSECT y EXCEPT	No	Sí
Funciones de Ventana	Sí, incluyendo algunas funciones agregadas	Sí, incluyendo todas las funciones agregadas

Fuente: Elaboración propia basado en [42]

Base de datos NoSQL

NoSQL almacena y consulta datos de manera diferente a las bases de datos relacionales, destacándose por su capacidad de escalar rápidamente y manejar grandes volúmenes de datos a alta velocidad. Son populares en entornos de nube, *Big Data* y aplicaciones web y móviles, ofreciendo flexibilidad frente a las limitaciones de las bases de datos relacionales en entornos de rápido crecimiento, como el comercio electrónico [43].

Algunos tipos de bases de datos NoSQL se presentan en la siguiente tabla.

Tabla 3. Tipos de base de datos NoSQL

Tipo	Detalle	Ejemplos
Almacén de pares clave - valor	Estructura simple de clave y valor en un diccionario. Ideal para almacenar información de sesión, como carritos de compra	Redis y Memcached
Almacén de documentos	Almacena datos como documentos en formatos como JSON, XML o BSON. Útil para datos semiestructurados y mayor flexibilidad.	MongoDB
Almacén distribuido en columnas	Almacena información en columnas, permitiendo el acceso selectivo. Intenta superar deficiencias de otros modelos.	Apache HBase y Apache Cassandra
Almacén de grafos	Gestiona datos de grafo de conocimiento con nodos, aristas y propiedades. Utilizado para almacenar y gestionar conexiones en una red.	Neo4j
Almacén en memoria	Datos residen en la memoria principal, acelerando el acceso.	IBM solidDB

Fuente: Elaboración propia con datos de [43]

2.1.5 Interfaz de programación de aplicaciones (API)

Un API es un mecanismo que facilita la comunicación entre dos componentes de software al establecer un conjunto de definiciones y protocolos. En el contexto de las API, la palabra "aplicación" se refiere a cualquier software con una función distinta. Se podría considerar que una API es un "contrato de servicio" entre dos aplicaciones, especificando cómo deben

comunicarse mediante solicitudes y respuestas [44]. Un ejemplo de API es la pasarela de pagos de Izypay, que permite a las aplicaciones de comercio electrónico procesar pagos con tarjeta de crédito [45].

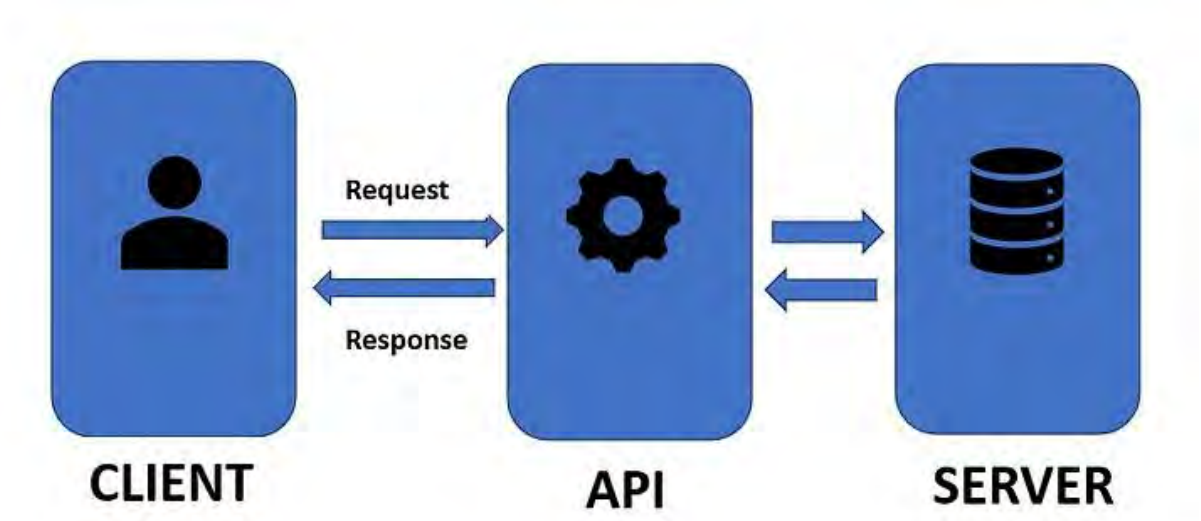


Figura 8. *API (Application Programming Interface)*
Fuente: [46]

2.1.6 Computación en la nube

Las empresas pueden acceder a servicios escalables y bajo demanda a través de internet en lugar de gestionar servidores físicos. El *cloud computing* proporciona recursos de computación bajo demanda [47]. Según el Cuadrante Mágico de Gartner 2022, los principales proveedores son Amazon Web Services (AWS), Google Cloud y Microsoft Azure [48].

Según Google Cloud [49], los beneficios de utilizar *cloud computing* son los siguientes:

- **Agilidad:** Acceso a servicios en la nube desde cualquier lugar con conexión a internet.
- **Elasticidad:** Capacidad para escalar y reducir servicios verticalmente según sea necesario.

- **Rentabilidad:** Pago solo por los recursos de computación utilizados, evitando la necesidad de sobrecargar la capacidad de los centros de datos.
- **Seguridad:** Se resalta la seguridad de *cloud computing* como sólida, con riesgos relativamente bajos en comparación con los centros de datos de las empresas.

Tipos de modelos de servicio

Existen tres tipos principales de modelos de servicio de *cloud computing*, cada uno ofreciendo un nivel diferente de control y gestión [49].

- **Infraestructura como Servicio (IaaS):** IaaS proporciona control avanzado sobre recursos TI como servidores, almacenamiento, redes y virtualización. El cliente puede seleccionar y gestionar su sistema operativo y otros recursos en una Máquina Virtual (VM). Ejemplos de IaaS incluyen AWS (EC2) y Google Compute Engine [49].
- **Plataforma como Servicio (PaaS):** PaaS elimina la necesidad de gestionar infraestructura, proporcionando un entorno completo para desarrollar, implementar y gestionar aplicaciones sin preocuparse por hardware o sistemas operativos. Los desarrolladores pueden centrarse en sus aplicaciones sin gestionar recursos o mantenimiento. Ejemplos de PaaS incluyen Red Hat Openshift y AWS Elastic Beanstalk [49].
- **Software como Servicio (SaaS):** SaaS proporciona a los usuarios acceso a aplicaciones completas gestionadas por el proveedor, eliminando la necesidad de administrar infraestructura o mantenimiento. Los usuarios simplemente utilizan la aplicación sin preocuparse por aspectos técnicos. Ejemplos de SaaS incluyen Gmail, Zoom y Facebook [49].

En el Anexo 10, se muestra gráficamente el nivel de control que el cliente obtiene en cada servicio, destacando cómo varía desde un mayor control en IaaS hasta un menor control en SaaS.

2.1.7 Inteligencia Artificial como Servicio (AIaaS)

Según la ISO, la Inteligencia Artificial (IA) es la capacidad de un sistema para interpretar datos, aprender de ellos y lograr objetivos específicos [50]. IBM la define como el uso de algoritmos y modelos de aprendizaje para realizar tareas cognitivas humanas [51]. Incluye recomendaciones, reconocimiento de voces y rostros, y generación de texto para *chatbots*, mejorando con aprendizaje automático.

Según Google, el *Machine Learning* es una aplicación de la IA que permite a las máquinas aprender y mejorar a partir de la experiencia sin programación explícita, usando algoritmos para analizar grandes volúmenes de datos y realizar predicciones [52].

El *Machine Learning* se divide en varias categorías, entre las que se destacan:

- **Aprendizaje Supervisado:** El aprendizaje supervisado entrena algoritmos con datos etiquetados para crear un modelo que predice resultados en nuevos datos, buscando generalizar y realizar predicciones precisas [53].
- **Aprendizaje No Supervisado:** El aprendizaje no supervisado utiliza datos no etiquetados para identificar patrones y estructuras ocultas. A diferencia del aprendizaje supervisado, no hay respuestas predefinidas; el algoritmo explora datos para encontrar similitudes y diferencias, descubriendo información valiosa sin intervención humana directa [54].

La Inteligencia Artificial como Servicio (AIaaS) permite a las organizaciones acceder a herramientas de IA en la nube sin necesidad de infraestructura local. Proveedores como AWS

y GCP ofrecen soluciones avanzadas de recomendaciones personalizadas mediante *Machine Learning*, permitiendo a las empresas ofrecer experiencias personalizadas basadas en el comportamiento de los usuarios.

- **Amazon Personalize:** Amazon Personalize es un servicio de *Machine Learning* ofrecido por AWS que permite a los desarrolladores crear sistemas de recomendación personalizados en tiempo real. Utiliza técnicas avanzadas de aprendizaje automático desarrolladas por Amazon para analizar datos de interacciones y preferencias de los usuarios, generando recomendaciones precisas y relevantes sin que el desarrollador necesite experiencia previa en *Machine Learning*. Además, Amazon Personalize soporta la ingesta de eventos en tiempo real para actualizar los modelos dinámicamente, y ofrece capacidades de ranking para ordenar resultados según la relevancia, mejorando así la personalización y la experiencia del usuario. [55].
- **Google BigQuery ML:** Google BigQuery ML permite crear y ejecutar modelos de *Machine Learning* directamente en BigQuery usando SQL. Es ideal para usuarios con conocimientos más avanzados, para la construcción de modelos personalizados de regresión, clasificación y otros tipos. [56].

2.1.8 Firebase (BaaS)

Firebase es una plataforma de desarrollo de aplicaciones móviles y web ofrecida por Google. Proporciona servicios integrados que simplifican el desarrollo y gestión de aplicaciones, reduciendo la necesidad de manejar infraestructura o *backend* [57]. A continuación, se presentan sus principales productos:

- **Cloud Firestore:** Cloud Firestore es una base de datos NoSQL flexible y escalable de Google Cloud, diseñada para servidores, móviles y web. Usa colecciones y documentos para almacenar datos, ofreciendo sincronización en tiempo real y soporte sin conexión.

Sus características clave incluyen consultas expresivas, actualizaciones en tiempo real y fácil integración con otros productos de Firebase y Google Cloud [58].

- **Cloud Functions:** Cloud Functions para Firebase es un *framework* sin servidores que ejecuta automáticamente código de *backend* en respuesta a eventos de Firebase o solicitudes HTTPS. Permite a los desarrolladores centrarse en la lógica de la aplicación sin gestionar la infraestructura, respondiendo a eventos como cambios en la base de datos o solicitudes HTTP [58].
- **Firebase Cloud Messaging (FCM):** FCM permite a las aplicaciones cliente enviar mensajes, como confirmaciones o mensajes de chat, desde los dispositivos a los servidores a través del canal de conexión eficiente y de bajo consumo de batería proporcionado por FCM [58].
- **Firebase Authentication:** Es un servicio completo que facilita la identificación segura de usuarios en aplicaciones mediante contraseñas, números de teléfono y proveedores federados, y cumple con estándares como OAuth 2.0 [58].

2.2 Estado del arte

Este apartado analiza las soluciones y tendencias actuales en plataformas digitales y aplicaciones móviles enfocadas en la contratación de músicos, con el fin de identificar enfoques existentes y oportunidades de mejora.

2.2.1 Métodos tradicionales de publicidad

Los carteles “chicha" son una forma popular y tradicional de publicidad informal en Perú, utilizada especialmente para promocionar eventos musicales, fiestas y conciertos en barrios populares. Estos carteles se colocan en postes, paredes, paraderos y otros espacios públicos, presentando un diseño llamativo, pero a menudo poco profesional, con colores vivos y tipografías grandes para captar la atención.

2.2.2 Redes sociales

Actualmente, en Lima Metropolitana, los músicos optan en su mayoría por el uso de las redes sociales para publicitar sus actividades y servicios. Las redes sociales permiten llegar a un mayor volumen de personas que podrían estar interesadas en estos eventos.

Además, estas ofrecen un perfil personalizable donde el músico puede dar a conocer sus trabajos y talento. La persona interesada en contactar al músico puede escribir directamente a este, y el músico puede responder para llegar a un acuerdo para el proceso de pago y el servicio musical. Ejemplos de las redes sociales más utilizadas para esta función son Instagram, WhatsApp y Facebook.

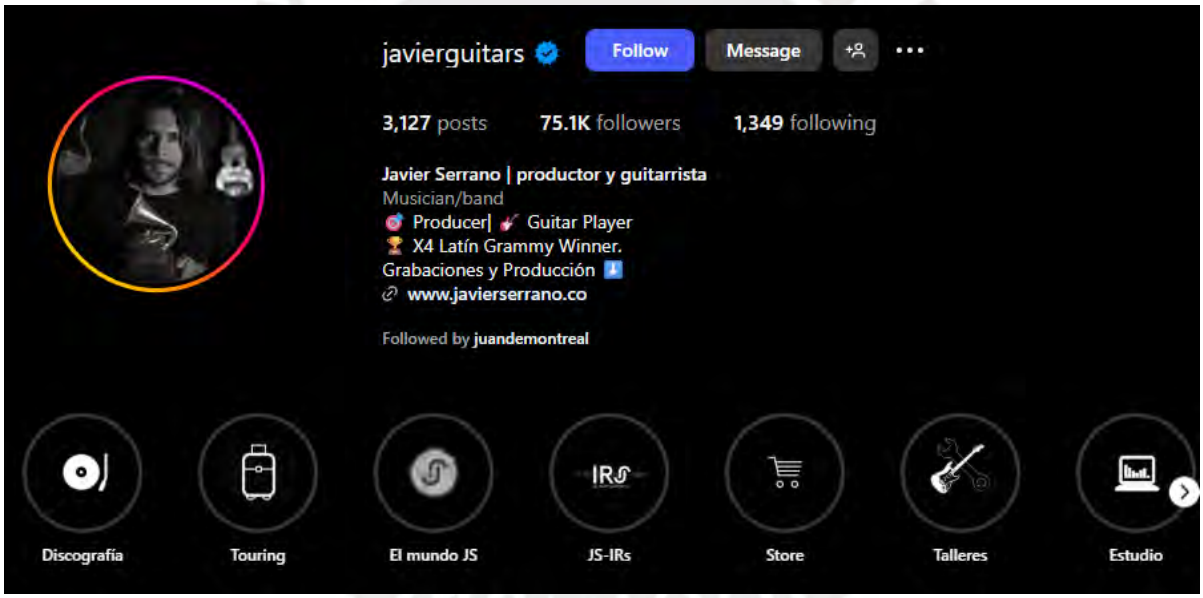


Figura 9. Ejemplo de perfil personalizado de músico en una red social
Fuente: [59]

2.2.3 Tutoque.co

Tutoque.co, fundada por Sebastián Larrañaga en 2018, es un *marketplace* para contratar músicos en vivo en la escena musical colombiana, operando en ciudades como Bogotá, Cartagena, Cali, Barranquilla y Medellín. La plataforma digital simplifica la búsqueda,

contratación y pago, ofreciendo a clientes y artistas una experiencia eficiente y segura para eventos corporativos y personales [60].

Figura 10. Logo de Tutoque.co
Fuente: [60]

2.2.4 ReverbNation

ReverbNation, creada el 31 de octubre de 2006, es una aplicación web que conecta artistas, productores y fanáticos. Los fans pueden descubrir conciertos, escuchar y comprar música, mientras que los artistas pueden crear perfiles profesionales y promocionar sus eventos y talentos musicales [61].

Figura 11. Logo de ReverbNation
Fuente: [62]

2.2.5 GIGSALAD

GIGSALAD, fundada en 2007, es una plataforma que conecta anfitriones con artistas y profesionales para eventos en EE.UU. y Canadá. Permite buscar, comparar y reservar músicos, DJ y animadores con garantía de pago y disponibilidad asegurada [63] [64].

Figura 12. Logo de GIGSALAD
Fuente: [64]

2.3 Comparativo y reflexiones finales

A lo largo de este capítulo, se han explorado diversas tecnologías fundamentales para el desarrollo de aplicaciones móviles. En cuanto a los sistemas operativos móviles, Android e iOS dominan el mercado, lo que confirma la necesidad de desarrollar soluciones compatibles con ambas plataformas.

Las aplicaciones multiplataforma, como Flutter y React Native, permiten crear un solo código base para iOS y Android, optimizando el equilibrio entre rendimiento y costos. En particular, Flutter ofrece interfaces consistentes y de alto rendimiento, gracias a Dart.

El análisis de bases de datos evidencia que la elección entre SQL y NoSQL depende en gran medida de la naturaleza de los datos y la necesidad de escalabilidad. MySQL y PostgreSQL son opciones robustas para almacenamiento estructurado, mientras que soluciones NoSQL como Firebase Firestore o MongoDB ofrecen flexibilidad para manejar datos no estructurados y sincronización en tiempo real, lo cual es clave en aplicaciones con interacción dinámica.

Por otro lado, la computación en la nube y los servicios BaaS como Firebase facilitan el desarrollo ágil al proveer infraestructura escalable sin necesidad de gestión directa de servidores. Además, la Inteligencia Artificial como Servicio (AIaaS) presenta oportunidades para mejorar la personalización dentro de la aplicación, como en el caso de sistemas de recomendación basados en machine learning.

Finalmente, si bien existen plataformas, como Tutoque.co, ReverbNation, y GIGSALAD, que ofrecen soluciones para la contratación de artistas, presentan limitaciones en términos de integración de pagos seguros, enfoque exclusivo en músicos y disponibilidad en aplicaciones móviles. Esto genera una oportunidad para desarrollar una solución innovadora que cubra estas deficiencias y brinde una experiencia optimizada para músicos y contratantes.

Capítulo 3. Arquitectura de la solución y diseño del aplicativo móvil

Este capítulo se centra en el análisis, diseño y arquitectura del aplicativo móvil. Primero, se analizarán los requerimientos funcionales y no funcionales del sistema. Luego, se justificarán las tecnologías seleccionadas para la implementación y se presentará la arquitectura de nube propuesta. Además, se describirán los flujos de interacción dentro de la aplicación. Finalmente, se propondrá el diseño de la interfaz de usuario.

3.1 Análisis de requerimientos

El análisis de los requerimientos de la aplicación es fundamental para su desarrollo. A continuación, se definen los requerimientos funcionales y no funcionales del sistema.

3.1.1 Requerimientos funcionales

Los requerimientos funcionales describen las funciones específicas que el aplicativo debe realizar. Estos se visualizan en la siguiente tabla:

Tabla 4. Requerimientos Funcionales

Requerimientos Funcionales	
RF1	La aplicación debe permitir a los usuarios actuar como músicos y clientes.
RF2	El usuario debe tener la posibilidad de indicar sus categorías de músico, como músico, y categorías de preferencia, como cliente. Las categorías se deben dividir entre **INSTRUMENTOS** y **GÉNEROS** musicales.
RF3	El usuario debe acceder a la aplicación autenticándose con su cuenta de Google en la vista **ACCESO**.
RF4	Obtener la ubicación del usuario durante la edición de su perfil solo debe permitirse si el usuario ha otorgado los permisos correspondientes.
RF5	Al acceder por primera vez, la aplicación solicita al usuario sus categorías de preferencia **(Instrumentos y/o géneros)** en la vista **PREFERENCIAS.**
RF6	Al acceder, el usuario visualizará en la vista **PRINCIPAL** a todos los músicos que recomienda la aplicación con un algoritmo de IA basado en la información almacenada hasta la fecha.
RF7	En la barra de navegación principal se tienen las siguientes opciones **CHAT**, **ALERTAS**, **MÚSICOS** y **EVENTOS**.
RF8	En la vista **PERFIL**, el usuario debe visualizar sus detalles (**email, nombre de usuario, número, foto de perfil, descripción, localización, categorías de músico, categorías musicales de preferencia y estadísticas)** y poder personalizar con la opción de subir 5 fotos y 2 videos de una calidad mínima de 360p para presentarse como músico o cliente.
RF9	En la vista **MÚSICOS**, se debe mostrar las opciones **PERFIL**, **BÚSQUEDA DE MÚSICOS**, **MIS TRABAJOS** y **MIS EVENTOS**.
RF10	En la vista **BÚSQUEDA DE MÚSICOS**, se debe visualizar a los músicos recomendados; proporcionar opciones de **filtro por ubicación, instrumentos, géneros o nombres de usuario** para una búsqueda más personalizada; habilitar un botón que acceda a la ventana **MAPA RESULTADOS**; y, seleccionando a cualquier músico, acceder a la ventana **DETALLES MÚSICO** para más detalles.
RF11	Para visualizar la ventana **MAPA RESULTADOS**, se solicitará al usuario permiso para acceder a su ubicación. Si el usuario no acepta, no se permite el acceso a esta ventana.

Requerimientos Funcionales	
RF12	En la ventana **MAPA RESULTADOS**, se debe visualizar un mapa donde se muestre al cliente y los músicos resultantes de la búsqueda, y presentar la ventana **DETALLES MÚSICO**, al seleccionar a un músico en el mapa.
RF13	En la ventana **DETALLES MÚSICO**, se debe visualizar la presentación del músico, personalizado previamente en su perfil, y proporcionar la opción para invitarlo a un evento activo, previamente creado por el cliente.
RF14	En la vista **EVENTOS**, se debe mostrar las opciones **PERFIL**, **BÚSQUEDA DE EVENTOS**, **MIS TRABAJOS** y **MIS EVENTOS**.
RF15	En la vista **MIS EVENTOS**, se debe permitir el acceso a la vista **CREAR NUEVO EVENTO** y visualizar los eventos creados del cliente, donde cada evento proporcionará la opción de ver sus detalles y abrir la vista **DETALLES EVENTO**.
RF16	En la vista **CREAR NUEVO EVENTO**, se debe indicar el nombre del evento, sus categorías de etiqueta **(Instrumentos y/o géneros)**, tipo de evento, descripción, requisitos, sugerencias, hora de inicio, hora de finalización, ubicación y pago sugerido.
RF17	En la vista **DETALLES EVENTO**, se debe visualizar el detalle del evento, los músicos postulantes con sus respectivas propuestas de pago, proporcionar la opción de seleccionar oficialmente a un músico y un botón que permita acceder a la vista **EDITAR EVENTO**.
RF18	El cliente debe ser capaz de editar un evento en la vista **EDITAR EVENTO.**
RF19	Cada vez que un evento sea editado, la aplicación debe enviar una notificación push a los músicos postulantes sobre los cambios realizados.
RF20	Al ser notificados por cambios en un evento, la aplicación cambiará el estado de todos los músicos postulantes al estado de músicos invitados y, para volver al estado de músico postulante, individualmente cada músico debe aceptar los cambios del evento.
RF21	El cliente debe ser capaz de cancelar cualquiera de sus eventos activos.
RF22	Si es que se seleccionó a un músico, el cliente no puede editar o cancelar el evento hasta 2 horas antes del horario indicado.

Requerimientos Funcionales	
RF23	En la vista **BÚSQUEDA DE EVENTOS**, se debe visualizar a los eventos cercanos a la localización del músico; proporcionar opciones de filtro por ubicación, instrumentos, géneros, tipo de evento o nombre de evento para una búsqueda más personalizada; habilitar un botón que presenta la ventana **MAPA RESULTADOS**; y, al seleccionar cualquier evento, acceder a la ventana **DETALLES EVENTO - MÚSICO** para más detalles.
RF24	En la ventana **DETALLES EVENTO - MÚSICO**, se debe detallar el evento y proporcionar la opción para postular como músico.
RF25	La aplicación debe enviar notificaciones al usuario, adaptándose a su rol según el contexto. Por ejemplo, si el usuario es invitado a un evento, se le notificará como músico; mientras que, si crea un evento y recibe postulaciones de músicos, se le notificará como cliente sobre las postulaciones.
RF26	El sistema debe detectar cuándo ha finalizado el evento y, posteriormente, enviar una notificación al cliente o músico correspondiente, solicitando que califique al músico o evento, respectivamente, y que proporcione un comentario adicional.
RF27	En la vista **CHAT**, se debe visualizar el historial de chats con clientes, donde el usuario como músico postuló a eventos, o chats con músicos, donde el usuario como cliente creó eventos y ellos postularon.
RF28	El usuario debe acceder a la vista **Alertas** para visualizar las notificaciones sobre invitaciones a eventos, cambios en eventos que este postuló y postulaciones de músicos a sus eventos activos.
RF29	La aplicación debe integrar una pasarela de pagos segura para gestionar los pagos entre el cliente y el músico una vez que el músico haya sido contratado para un servicio.
RF30	Al finalizar el pago, la aplicación debe permitir al cliente calificar al músico y al músico calificar el evento basado en su experiencia.

Fuente: Elaboración propia

3.1.2 Requerimientos no funcionales

Los requerimientos no funcionales se refieren a cómo el sistema realiza las funciones requeridas y a través de qué tecnologías. Estos se visualizan en la siguiente tabla.

Tabla 5. Requerimientos No Funcionales

Requerimientos No Funcionales	
RNF1	El sistema debe implementar una arquitectura en la nube de alta disponibilidad para garantizar un tiempo de actividad óptimo y minimizar el riesgo de interrupciones del servicio.
RNF2	El sistema debe implementar una arquitectura en la nube escalable que permita la gestión dinámica de recursos para adaptarse a las variaciones en el tráfico y la carga de trabajo.
RNF3	La comunicación debe ser segura entre la API y la aplicación móvil mediante el uso de HTTPS para proteger la integridad y confidencialidad de los datos transmitidos.
RNF4	La arquitectura del sistema debe incluir medidas de seguridad robustas, como mecanismos de autenticación y autorización, para prevenir accesos no autorizados a los servicios y datos.
RNF5	El sistema debe integrarse con una pasarela de pagos externa certificada con PCI DSS, para garantizar que no se almacenen datos sensibles de tarjetas en nuestros servidores.

Fuente: Elaboración propia

3.2 Arquitectura de la solución

La arquitectura propuesta se presenta en la siguiente figura. Esta incluye a Google Cloud Platform (GCP) y Amazon Web Services (AWS) como proveedores de servicios en la nube; Stripe como pasarela de pagos; y Flutter como *framework* para el desarrollo de la aplicación Android.

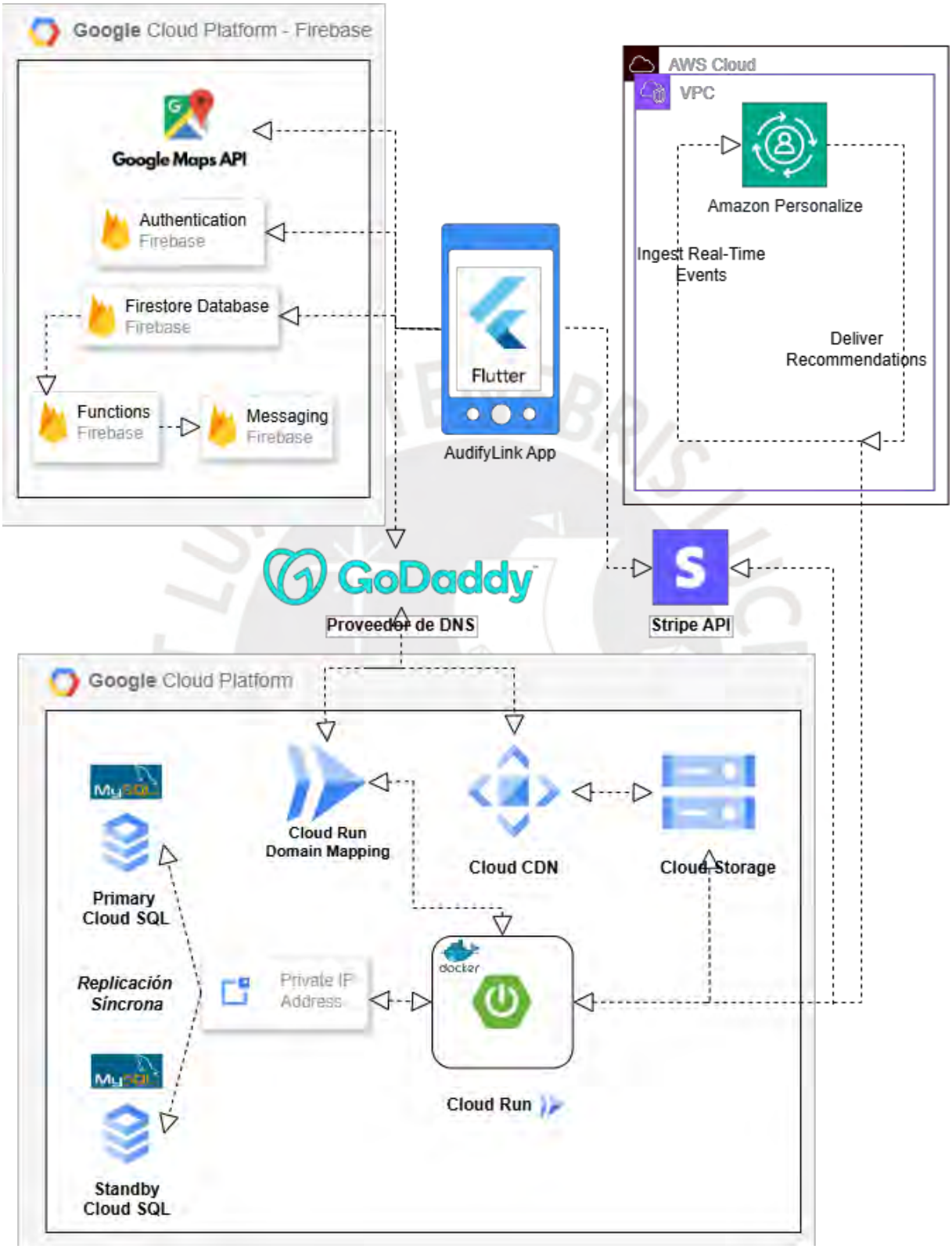


Figura 13. *Arquitectura de aplicación propuesta*
Fuente: Elaboración propia

3.3 Selección de tecnologías y servicios

A continuación, se detallan y justifican las tecnologías y servicios utilizados en la arquitectura de la solución.

3.3.1 Flutter

Para el desarrollo del aplicativo móvil, se decide utilizar Flutter debido a su rendimiento casi nativo, su eficiencia de desarrollo utilizando *hot reload*, y la consistencia visual proporcionada por su enfoque en widgets.

3.3.2 Evaluación y selección de proveedores cloud: GCP, AWS y Azure

Como se mencionó en la sección 2.1.6, los tres principales proveedores de servicios en la nube son Google Cloud Platform (GCP), Amazon Web Services (AWS) y Microsoft Azure. Cada uno ofrece un amplio portafolio de servicios que cubren los requerimientos del presente proyecto. No obstante, al evaluar factores como integración con tecnologías utilizadas, documentación, facilidad de uso y disponibilidad en la región, se establece una comparación técnica centrada en tres servicios clave:

- Base de datos relacional con alta disponibilidad (HA)
- Ejecución de contenedores Docker sin servidor
- Almacenamiento de archivos en la nube

Tabla 6. Servicios GCP, AWS y Azure

Servicio	GCP	AWS	Microsoft Azure
Base de datos relacional con HA	Cloud SQL (con replicación síncrona)	Amazon Aurora (Multi-AZ)	SQL Database (con redundancia geográfica)
Ejecución de contenedores (PaaS)	Cloud Run	App Runner / Fargate	Azure Container Apps
Almacenamiento de archivos	Cloud Storage	Amazon S3	Azure Blob Storage

Fuente: Elaboración propia basado en [65], [66] y [67]

Dado que los tres proveedores ofrecen los servicios necesarios, a continuación, se presenta una tabla de comparación de costos.

Tabla 7. Análisis de costos mensuales AWS, GCP y Azure

Servicio	GCP	AWS	AZURE
Base de datos con HA	Cloud SQL (db-f1-micro, 2 zonas, 10 GB)	Aurora Serverless v2 (Multi-AZ, 10 GB)	SQL Database Business Critical (redundancia geo, 10 GB - JA100)
USD/mes	$43.70	$89.13	$60.59
Ejecución de contenedores (PaaS)	Cloud Run (1M requests + 360 ms/request)	App Runner (1M requests + 360 ms/request)	Azure Container Apps
USD/mes	$0.16	$56.10	$4.32
Almacenamiento de archivos	Cloud Storage (100 GB, clase estándar)	S3 (100 GB, clase estándar)	Blob Storage (100 GB, Hot tier)
USD/mes	$3.26	$4.06	$3.30
Valores estimados calculados en base a precios públicos a mayo de 2025. Los costos pueden variar según región, uso y condiciones comerciales de cada proveedor.			

Fuente: Elaboración propia utilizando [68], [69] y [70]

Para el levantamiento de la API con contenedores, se ha decidido utilizar Cloud Run de GCP debido a su bajo costo y escalabilidad eficiente, lo que lo hace más conveniente en comparación con las opciones de AWS y Azure. En consecuencia, se emplea Cloud SQL como base de datos relacional con alta disponibilidad, ya que también representa la opción más económica y ofrece una integración nativa con Cloud Run. Finalmente, el almacenamiento de archivos será gestionado mediante Cloud Storage de GCP, ya que garantiza una mejor integración con los servicios seleccionados.

3.3.3 Firestore

Aunque Cloud SQL es ideal para datos relacionales con alta consistencia, se utiliza Cloud Firestore para gestionar datos dinámicos y en tiempo real, como chats y notificaciones. Firestore permite sincronización instantánea y escalabilidad automática, además de ofrecer una integración nativa y sencilla con Flutter.

De esta manera, se aprovecha la solidez de Cloud SQL para los datos estructurados, mientras que Firestore facilita el manejo eficiente de información en tiempo real, mejorando el desempeño general y la interacción con la aplicación.

3.3.4 Recomendaciones: Amazon Personalize

Como se mencionó en el inciso 2.1.8 del capítulo 2, Amazon Personalize es una solución desarrollada que facilita la implementación de sistemas de recomendación personalizados sin requerir conocimientos avanzados en *Machine Learning*. Su capacidad para integrarse fácilmente con el sistema mediante datos iniciales y actualizarse dinámicamente a través de eventos (*PUT EVENTS*) la adapta mejor a los objetivos y requerimientos del proyecto, en comparación con otras opciones que implican mayor complejidad técnica.

Por tanto, se opta por Amazon Personalize como servicio de recomendación para garantizar una integración eficiente y un desarrollo más ágil.

3.3.5 Cloud Delivery Network (CDN)

Para mejorar la entrega y visualización de imágenes y videos del repositorio de músicos, se utiliza Cloud CDN *(Content Delivery Network)*.

Cloud CDN es un servicio de red de entrega de contenido de GCP que utiliza la infraestructura global de Google para almacenar en caché contenido cerca de los usuarios finales. Esto reduce la latencia y mejora los tiempos de carga, proporcionando una mejor experiencia para los usuarios [71]. El uso de Cloud CDN junto a Cloud Storage garantiza una experiencia óptima para los usuarios al acceder a imágenes y videos del repositorio de músicos.

3.3.6 Gestión de dominio y redirección de subdominio

Para la gestión del dominio, se plantea utilizar el servicio de DNS proporcionado por GoDaddy, donde se administra el dominio principal del sistema. Mediante la funcionalidad de "Domain Mapping" de Cloud Run [72], se asignará un dominio personalizado al aplicativo, permitiendo su acceso mediante el protocolo HTTPS con certificados SSL gestionados automáticamente por la plataforma.

Asimismo, se considera la configuración de una redirección para el subdominio *repository*, que apuntará a la dirección IP pública del CDN encargado de servir los archivos multimedia.

3.3.7 Pasarela de pagos: Stripe

Stripe es una plataforma tecnológica que permite recibir pagos por Internet y gestiona todo el proceso de forma segura, desde el almacenamiento de datos de tarjetas hasta la transferencia de fondos [73].

Se integrará la API de Stripe en modo desarrollador con cuentas de prueba, lo cual es adecuado para el contexto de la tesis. Esta implementación cumple con el requerimiento funcional RF29 sobre la necesidad de una pasarela de pagos segura entre el cliente y el músico, y con el requerimiento no funcional RNF5, ya que Stripe cumple con el estándar de seguridad PCI DSS [74].

3.3.8 Autenticación con Firebase

En esta tesis, se utiliza Firebase Authentication para gestionar la autenticación de usuarios, permitiendo el inicio de sesión exclusivamente con cuentas de Google. Esta integración delega en Google el manejo del registro, recuperación de contraseña y control de acceso, lo que simplifica la implementación y refuerza la seguridad de la aplicación.

3.3.9 Notificaciones Push con Firebase Cloud Messaging y Functions

Para el envío de notificaciones *push*, se utiliza Firebase Cloud Messaging (FCM) en conjunto con Cloud Functions, las cuales actúan como emisoras. Estas funciones se activan automáticamente mediante un *trigger* asociado a cambios en la base de datos, analizan el evento y envían la notificación al usuario correspondiente.

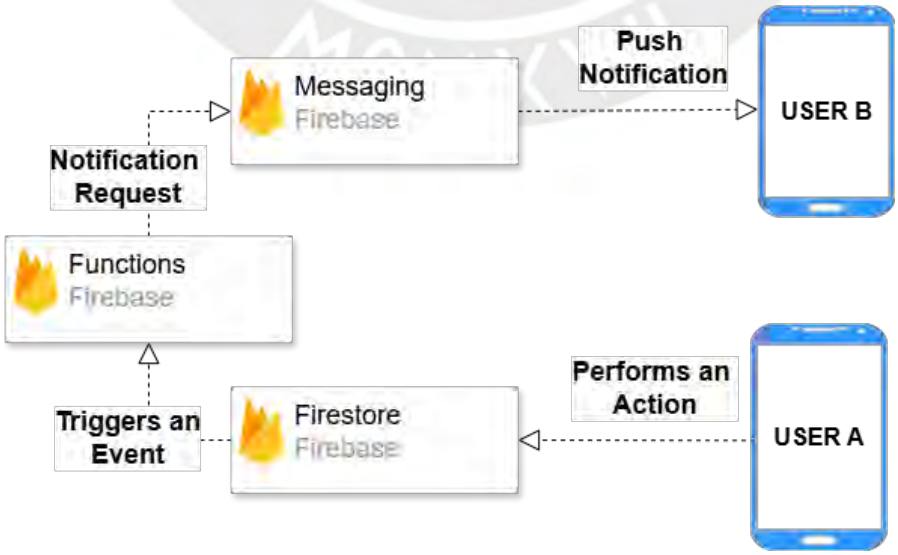


Figura 14. *Flujo de eventos para notificaciones push en tiempo real*
Fuente: Elaboración propia

Cuando se detecta un evento relevante; por ejemplo, la confirmación de un servicio o un nuevo mensaje, la función analiza el tipo de evento, identifica al destinatario correspondiente y solicita el envío de la notificación a través de FCM.

3.4 Diseño de la base de datos

El diseño de la base de datos relacional a implementar en MySQL se encuentra detallado en el Anexo 3, donde se presenta el diagrama entidad-relación correspondiente.

Por su parte, la base de datos en Cloud Firestore utiliza un modelo orientado a documentos. Se define una colección principal de usuarios, donde cada documento representa a un usuario y contiene sub colecciones de notificaciones. Una estructura similar se aplica para los chats y sus respectivos mensajes.

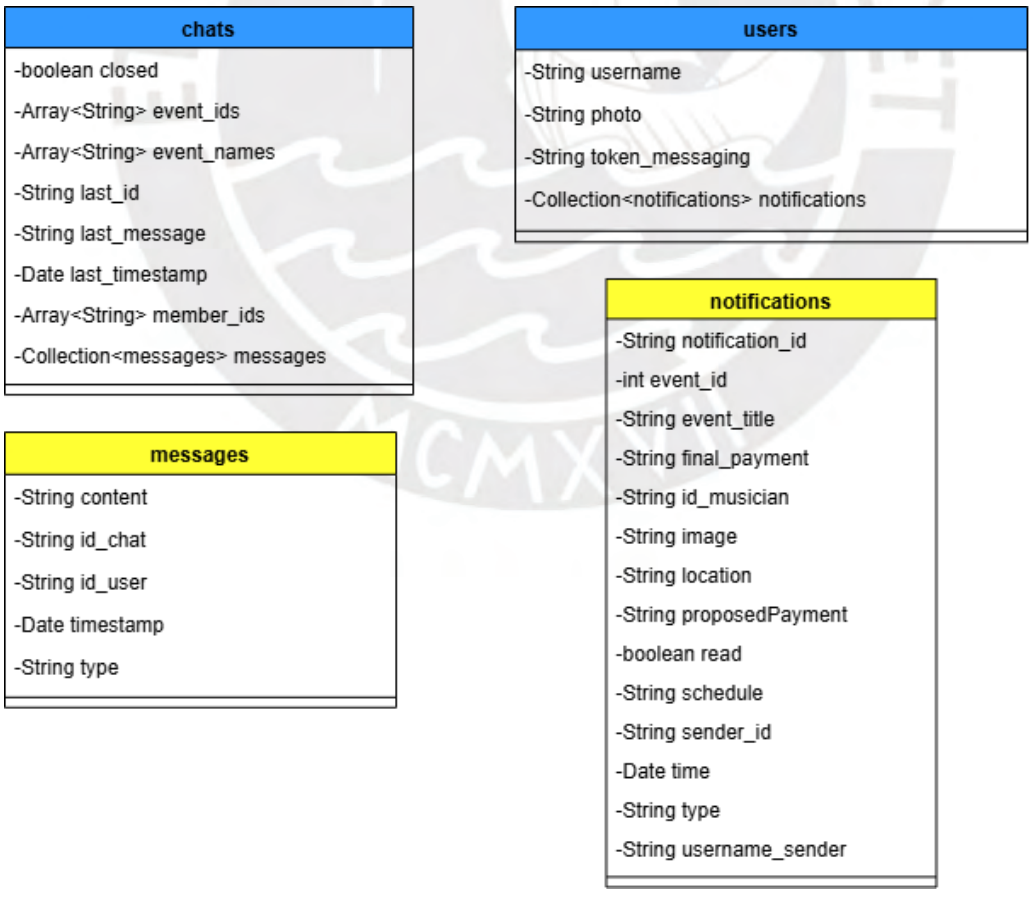


Figura 15. Modelo de datos en Firestore
Fuente: Elaboración propia

3.5 Diagramas de flujo

Los flujos funcionales del aplicativo móvil fueron diseñados para cubrir las principales interacciones entre usuarios y músicos. Estos flujos incluyen: creación de eventos, invitación de músicos, postulación a eventos, selección de músicos y pago de servicios.

Dado que el enfoque principal de esta tesis es el desarrollo e implementación de la arquitectura en la nube que soporta estas funcionalidades, se ha optado por ubicar los diagramas detallados de estos flujos en los Anexos 4 al 8 del documento.

3.6 Diseño de la interfaz de usuario

A continuación, se presentan las interfaces de la aplicación móvil.

3.6.1 Interfaz de autenticación y preferencias iniciales

Las pantallas correspondientes a esta sección se ilustran en el Anexo 9.1.

- **Inicio de sesión:** Pantalla que permite a los usuarios autenticarse mediante Google.
- **Selección de preferencias:** Después de la autenticación, los usuarios nuevos seleccionan sus géneros e instrumentos musicales preferidos para personalizar su experiencia.

3.6.2 Vista principal

La vista principal se compone de cuatro secciones: MÚSICOS, EVENTOS, CHATS y ALERTAS. Sus interfaces se muestran en los Anexos 9.2 y 9.3.

- **Sección de músicos:** Los usuarios pueden consultar su perfil, gestionar “Mis eventos” y “Mis trabajos”. Además, cuentan con una barra de búsqueda para encontrar músicos y una lista de músicos recomendados por la aplicación.

- **Sección de eventos:** Los usuarios pueden acceder a su perfil, gestionar “Mis eventos” y “Mis trabajos”, así como buscar nuevos eventos. La aplicación también muestra eventos recomendados personalizados.
- **Sección de chats:** Muestra el historial de conversaciones del usuario, permitiendo acceder a cada conversación individual.
- **Sección de alertas:** Los usuarios visualizan notificaciones importantes, como invitaciones a eventos, actualizaciones en eventos postulados y nuevas postulaciones de músicos a eventos activos.

3.6.3 Perfil del músico y repositorio multimedia

En esta sección se describen las funcionalidades principales del perfil del usuario y el repositorio multimedia. Las interfaces correspondientes se ilustran en el Anexo 9.4.

La interfaz de perfil permite al usuario visualizar y editar su información personal, incluyendo foto, nombre, calificación promedio, correo, teléfono, ubicación, descripción y preferencias musicales. Además, se muestra un repositorio de imágenes y videos, donde el usuario puede gestionar contenido multimedia que respalde sus habilidades y experiencia como músico.

3.6.4 Interfaz de visualización de detalles de músicos y eventos

En esta sección se describen las interfaces que permiten visualizar en detalle la información de un músico o de un evento. Las correspondientes pantallas se ilustran en los Anexos 9.5 y 9.6, respectivamente.

- **Vista de detalles de un músico:** Esta pantalla muestra información completa del músico, incluyendo su descripción, categorías musicales, correo electrónico, número de contacto y ubicación. Además, permite al usuario invitar al músico a un evento directamente desde esta interfaz.

- **Vista de detalles de un evento:** Esta vista permite a los músicos acceder a toda la información relevante de un evento: horario, estado, tipo, ubicación y pago sugerido. Desde esta pantalla, el músico puede postularse fácilmente al evento si está interesado.

3.6.5 Interfaz de gestión de postulaciones y eventos creados

La aplicación cuenta con dos secciones dedicadas a la organización de las actividades del usuario: “Mis trabajos” y “Mis eventos”. Estas secciones permiten gestionar las postulaciones realizadas y los eventos registrados dentro de la plataforma. Las interfaces correspondientes se muestran en el Anexo 9.7.

- **Sección “Mis trabajos”:** Esta pantalla presenta el listado de actividades del usuario como músico, incluyendo los trabajos realizados, las postulaciones en curso y el estado de cada participación en eventos.
- **Sección “Mis eventos”:** Esta vista muestra los eventos creados por el usuario. Desde esta sección es posible acceder al detalle de cada evento para consultar información más específica, editar datos del evento o registrar uno nuevo.

3.6.6 Creación y edición de eventos

La aplicación proporciona una interfaz para la creación y edición de eventos, mediante la cual el usuario puede registrar o modificar información relevante. Esta pantalla permite ingresar campos como el título del evento, descripción, indicaciones adicionales, horario, tipo de evento, ubicación y el pago sugerido. Además, incorpora un mapa interactivo que facilita la selección geográfica del lugar donde se llevará a cabo el evento. Las interfaces correspondientes se encuentran en el Anexo 9.8.

3.6.7 Detalles de eventos creados y gestión de postulantes

Esta interfaz presenta la información detallada de un evento registrado y la lista de músicos postulantes, con datos relevantes para su evaluación y selección. Las interfaces correspondientes se encuentran en el Anexo 9.9.

3.6.8 Pasarela de pagos

La aplicación integra una pasarela de pagos segura proporcionada por Stripe, que permite realizar transacciones para el pago al músico seleccionado en un evento. El proceso se ejecuta de manera automática y segura a través de la plataforma Stripe, garantizando la protección de los datos. Las interfaces correspondientes se presentan en el Anexo 9.10.

Capítulo 4. Despliegue del aplicativo, pruebas de funcionamiento y análisis de costos

Este capítulo detalla el proceso de despliegue del aplicativo, incluyendo configuraciones técnicas. Además, se presentan pruebas funcionales del aplicativo y un análisis detallado de costos y rentabilidad, abarcando mano de obra y operación.

4.1 Creación de cuenta y uso de Stripe

Para el desarrollo y pruebas de la funcionalidad de pagos, se utilizó una cuenta de prueba en Stripe, adecuada para entornos de desarrollo y sin fines comerciales. A través del panel para desarrolladores, Stripe proporciona una Clave Secreta (*Secret Key*), utilizada para autenticar y autorizar solicitudes desde el *backend* hacia su API. Esta clave permite realizar operaciones como la creación, confirmación y cancelación de pagos.

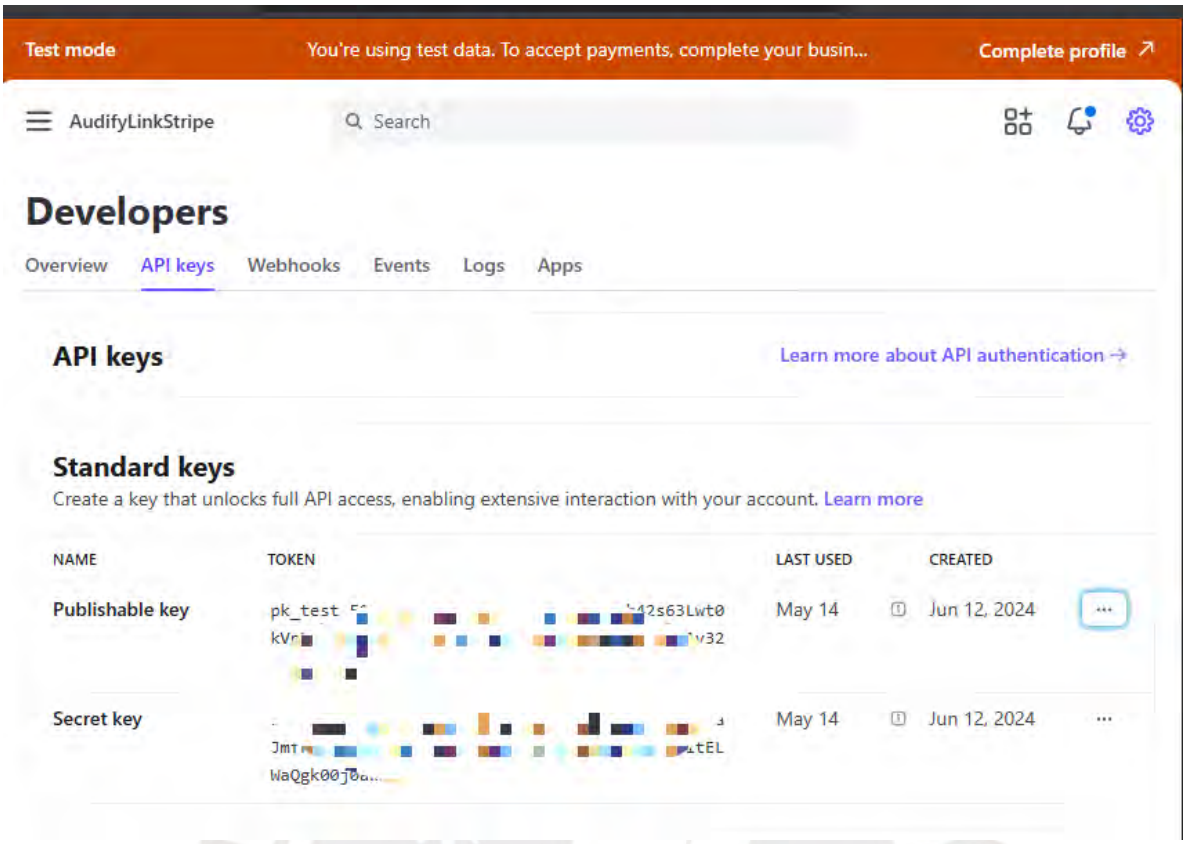


Figura 16. *Cuenta y clave Secreta proporcionada por Stripe*
Fuente: Elaboración propia

4.2 Instancia Cloud SQL con replicación síncrona en GCP

Se implementa una instancia de Cloud SQL con replicación síncrona que garantiza alta disponibilidad y consistencia de datos. Esta sección incluye su creación y configuración, así como la integración con la API.

4.2.1 Creación de instancia

Se crea una instancia de MySQL 8.0 configurada con alta disponibilidad mediante replicación síncrona entre zonas dentro de la región seleccionada. La creación de la instancia se realiza a través de la herramienta de línea de comandos *gcloud*, utilizando el siguiente comando:

```
gcloud sql instances create <NOMBRE_INSTANCIA_SQL> \
  --database-version=MYSQL_8_0 \
  --cpu=<NUM_CPU> \
  --memory=<TAM_MEMORIA> \
  --region=<REGION> \
  --availability-type=REGIONAL \
  --root-password=<CONTRASEÑA> \
  --enable-bin-log
```

Una vez desplegada, se puede verificar que la instancia fue creada correctamente utilizando el siguiente comando, que lista todas las instancias disponibles en el proyecto:

```
gcloud sql instances list
```

Este comando mostrará una tabla con información básica. En dicha tabla, se debe encontrar la instancia creada con el nombre <NOMBRE_INSTANCIA_SQL>. Para este despliegue, se utilizó una configuración con 1 vCPU y 3840MB de memoria, desplegada en la región “us-central1”.

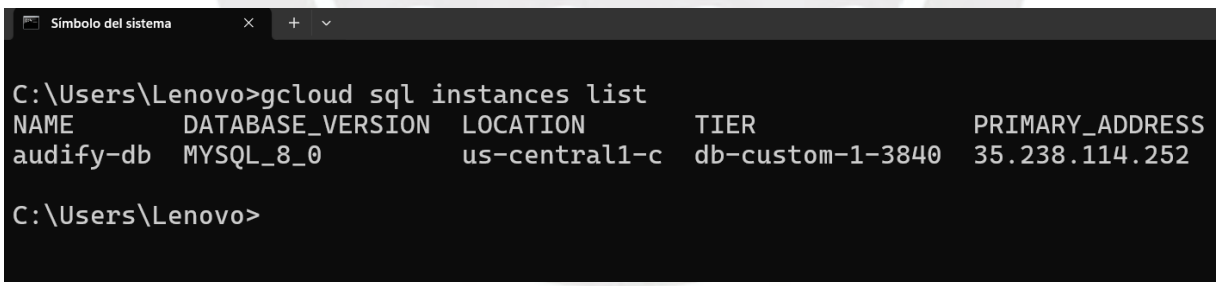


Figura 17. Instancia SQL creada en GCP
Fuente: Elaboración propia

4.2.2 Integración con la API

La API se integra con la instancia de Cloud SQL utilizando la dependencia oficial de Google Cloud SQL Socket Factory para MySQL, que permite establecer conexiones seguras mediante sockets Unix. Esta configuración es especialmente necesaria al desplegar la API en Cloud Run.

La conexión se configura en el archivo “application.properties” de Spring Boot, especificando la instancia Cloud SQL a través del parámetro *cloudSqlInstance* en la URL JDBC, y utilizando el *socket factory* provisto por Google. Además, se activa el perfil *cloud* para ajustar la configuración del entorno.

```
spring.profiles.active=cloud
spring.datasource.url=jdbc:mysql:///beatblink_db?cloudSqlInstan
spring.datasource.username=root
spring.datasource.password=
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

server.servlet.context-path=/api/v1
```

Figura 18. Configuración de conexión a Cloud SQL en el archivo application.properties
Fuente: Elaboración propia

4.3 Almacenamiento de archivos con Cloud Storage

En esta sección se describe el uso de Cloud Storage como servicio de almacenamiento de archivos, incluyendo la creación de un *bucket* y su integración con la API para operaciones como subida y eliminación de archivos.

4.3.1 Creación de *bucket*

En esta etapa, se configura un *bucket* de GCS que servirá como repositorio para almacenar los archivos multimedia generados por los usuarios de la aplicación, tales como fotos de perfil, imágenes y videos. La creación del *bucket* se realiza mediante la herramienta de línea de comandos de Google Cloud, utilizando el siguiente comando:

```
gcloud storage buckets create gs://<BUCKET_NAME> --location=<LOCATION> --uniform-bucket-level-access
```

Este comando crea un *bucket* con control de acceso uniforme a nivel de *bucket*, lo que facilita la administración de permisos de forma centralizada.

Dado que la aplicación permite que cualquier usuario pueda visualizar el contenido multimedia de otros usuarios, el *bucket* se configura como público para facilitar el acceso sin necesidad de autenticación. Para esto, se otorga el permiso de lectura a todos los usuarios mediante el siguiente comando:

```
gcloud storage buckets add-iam-policy-binding gs://<BUCKET_NAME> --member=allUsers
--role=roles/storage.objectViewer
```

La estructura del *bucket* está organizada en carpetas segmentadas mediante identificadores de usuario codificados con funciones hash, lo que permite mejorar la seguridad al evitar la exposición directa de los identificadores reales. Dentro de cada carpeta correspondiente a un usuario se almacenan:

- La foto de perfil del usuario.
- El repositorio con las fotos y videos subidos.
- Las miniaturas (*frames*) de los videos, que se utilizan como *previews* antes de la reproducción.

4.3.2 Distribución y optimización del contenido público con CDN

Se configura una CDN de Google Cloud que sirve como capa de caché y distribución global, lo que permite que los archivos se sirvan de manera eficiente y cercana a los usuarios.

Para habilitar esta funcionalidad, es necesario realizar una serie de configuraciones en GCP mediante la línea de comandos.

- Crear un *"backend bucket"* que conecta al *bucket* de almacenamiento con la CDN.

- Configurar un *"URL map"* que define la ruta de las solicitudes hacia el *"backend bucket"*.
- Crear un certificado SSL gestionado para el dominio que usará la CDN.
- Crear un *"proxy HTTPS"* que manejará las solicitudes seguras y direccionará según el *"URL map"*.
- Asignar una dirección IP externa global que será el punto de entrada del tráfico.
- Definir un *"forwarding rule"* que asocia la IP y el proxy HTTPS para enrutar el tráfico por el puerto 443.

Los comandos utilizados son los siguientes:

```
gcloud compute backend-buckets create <BACKEND_BUCKET_NAME> --gcs-bucket-name=<BUCKET_NAME> --enable-cdn


gcloud compute url-maps create <URL_MAP_NAME> --default-backend-bucket=<BACKEND_BUCKET_NAME>


gcloud compute ssl-certificates create <SSL_CERT_NAME> --domains=<YOUR_DOMAIN>


gcloud compute target-https-proxies create <HTTPS_PROXY_NAME> --ssl-certificates=<SSL_CERT_NAME> --url-map=<URL_MAP_NAME>


gcloud compute addresses create <GLOBAL_IP_NAME> --global


gcloud compute forwarding-rules create <FORWARDING_RULE_NAME> \
  --global --target-https-proxy=<HTTPS_PROXY_NAME>\
  --ports=443 --address=<GLOBAL_IP_NAME>
```

La descripción de la regla de *forwarding HTTPS* creada, que asocia la dirección IP global con el proxy HTTPS para enrutar el tráfico por el puerto 443, se presenta en la siguiente figura:

```
C:\Users\Lenovo>gcloud compute forwarding-rules describe images-audify-https-rule
 --global
IPAddress: 34.149.216.27
IPProtocol: TCP
creationTimestamp: '2025-05-21T10:56:02.078-07:00'
description: ''
fingerprint: xndSTE58jGQ=
id: '5795064925898648925'
kind: compute#forwardingRule
labelFingerprint: 42WmSpB8rSM=
loadBalancingScheme: EXTERNAL
name: images-audify-https-rule
networkTier: PREMIUM
portRange: 443-443
```

Figura 19. Descripción de la regla de forwarding HTTPS configurada en GCP
Fuente: Elaboración propia

Una vez completadas las configuraciones en GCP, es necesario vincular el dominio con la infraestructura creada. En este caso, se utiliza un subdominio específico (repository.audifylink.xyz) para servir contenido estático alojado en el *bucket*. Para ello, se debe acceder al panel de administración del proveedor DNS y añadir un registro A que apunte al subdominio "repository" a la dirección IP global reservada en GCP, asociada al proxy HTTPS.

Esta configuración se realiza desde el panel de GoDaddy, como se muestra en la siguiente figura.

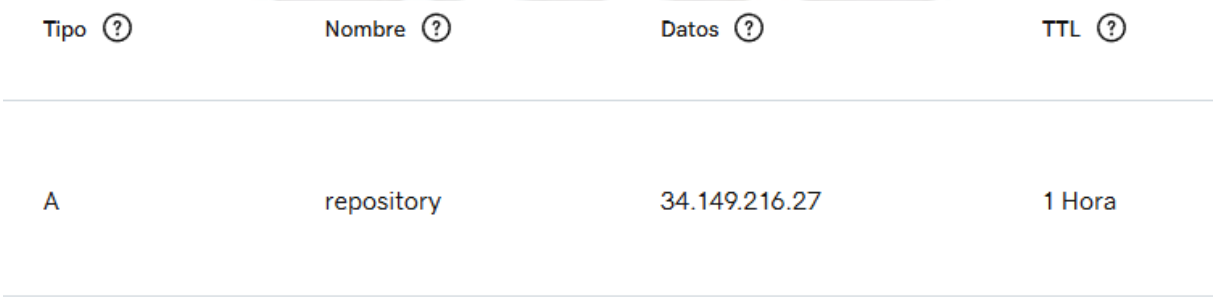


Figura 20. Configuración del registro A para el subdominio repository en GoDaddy
Fuente: Elaboración propia

4.3.3 Integración con la API

La API desarrollada en Spring Boot accede directamente al *bucket* utilizando la librería oficial "google-cloud-storage" (versión 2.50.0), incluida como dependencia de Maven. Gracias a las credenciales de servicio, la API puede subir, eliminar y gestionar archivos de forma segura.

Cuando un usuario sube una imagen, esta se guarda en el *bucket* y su URL pública (basada en el subdominio repository.audifylink.xyz) se registra en la base de datos SQL. Del mismo modo, una solicitud de eliminación remueve tanto el archivo en almacenamiento como su referencia en la base de datos.

4.4 Integración del aplicativo con Firebase

El aplicativo móvil utiliza servicios de Firebase para habilitar funcionalidades como almacenamiento en tiempo real, autenticación de usuarios y envío de notificaciones *push*. Esta sección describe cómo se integran estos servicios en Flutter mediante las dependencias oficiales disponibles.

4.4.1 Integración con Firestore

Se incorpora el paquete oficial "cloud_firestore" (versión 5.6.6) para acceder a colecciones y documentos directamente desde Flutter.

Como se observa en la siguiente figura, las colecciones utilizadas por la aplicación se visualizan correctamente en el *dashboard* durante su ejecución.

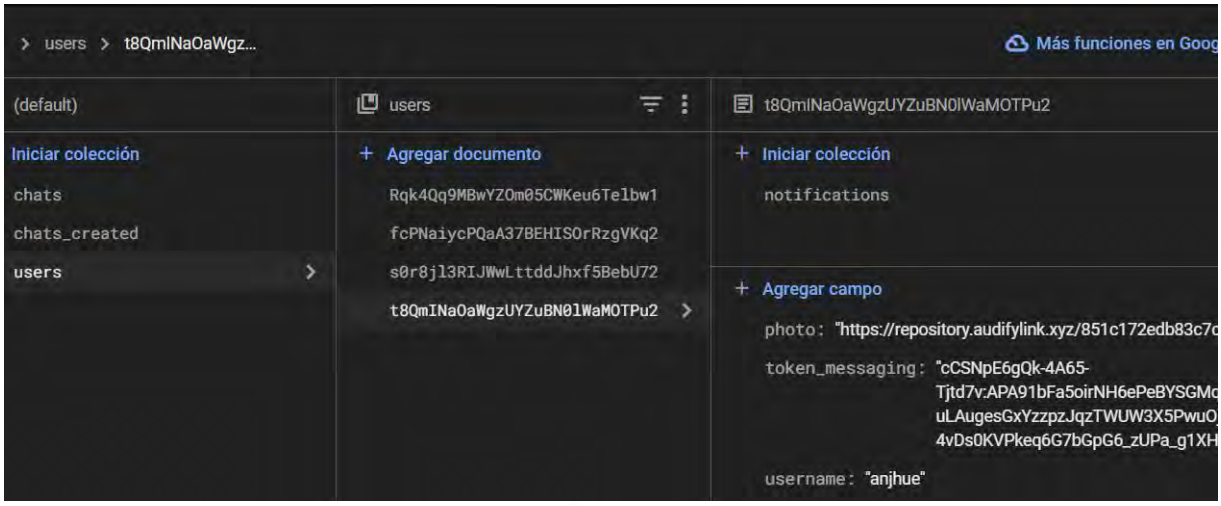


Figura 21. *Visualización de colecciones en Firestore desde el dashboard de Firebase*
Fuente: Elaboración propia

4.4.2 Integración con Firebase Authentication

Para la gestión de usuarios, se integra el paquete oficial “firebase_auth” (versión 5.5.2), que permite registrar, autenticar y mantener sesiones de usuario directamente desde Flutter.

Como se muestra en la siguiente figura, el *dashboard* de Firebase permite visualizar los usuarios registrados durante el uso del aplicativo.

Figura 22. *Visualización de usuarios autenticados en el dashboard de Firebase*
Fuente: Elaboración propia

4.4.3 Integración con Firebase Cloud Messaging

Se integró el paquete oficial “firebase_messaging” (versión 15.2.5), que permite que la aplicación reciba mensajes incluso cuando se encuentra en segundo plano.

El envío de notificaciones se realiza automáticamente mediante funciones definidas en Firebase Functions, las cuales son ejecutadas a partir de eventos en la plataforma (por ejemplo, cambios en la base de datos o acciones del usuario). Como se observa en la siguiente figura, el *dashboard* permite visualizar y gestionar dichas funciones.

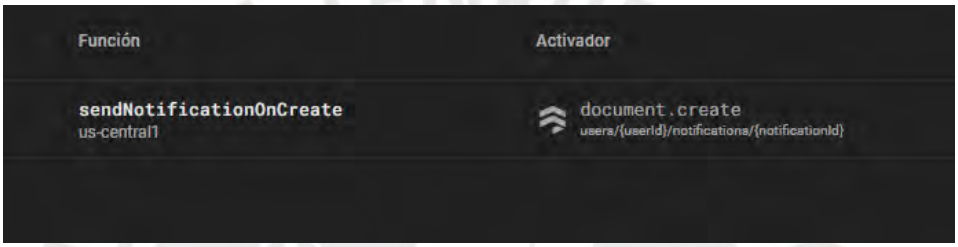


Figura 23. Dashboard de Firebase Functions con funciones para notificaciones push
Fuente: Elaboración propia

4.5 Integración del sistema de recomendaciones con AWS Personalize

Esta sección describe el proceso seguido para integrar el sistema de recomendaciones con el servicio Amazon Personalize, detallando desde la preparación de los datos hasta la configuración del modelo y su actualización en tiempo real.

4.5.1 Requisitos mínimos de datos para entrenamiento

Según la documentación oficial de AWS Personalize [75], para entrenar un modelo es necesario cumplir con ciertos requisitos mínimos de datos:

- Al menos 1 000 registros de interacciones entre usuarios y elementos.
- Un mínimo de 25 usuarios únicos, con al menos dos interacciones.

- Para recomendaciones de calidad, se recomienda un mínimo de 50 000 interacciones, 1 000 usuarios con dos o más interacciones cada uno.
- Es obligatorio contar con un conjunto de datos de interacciones, además de *datasets* complementarios para usuarios y elementos.

4.5.2 Generación y almacenamiento de datos de entrenamiento

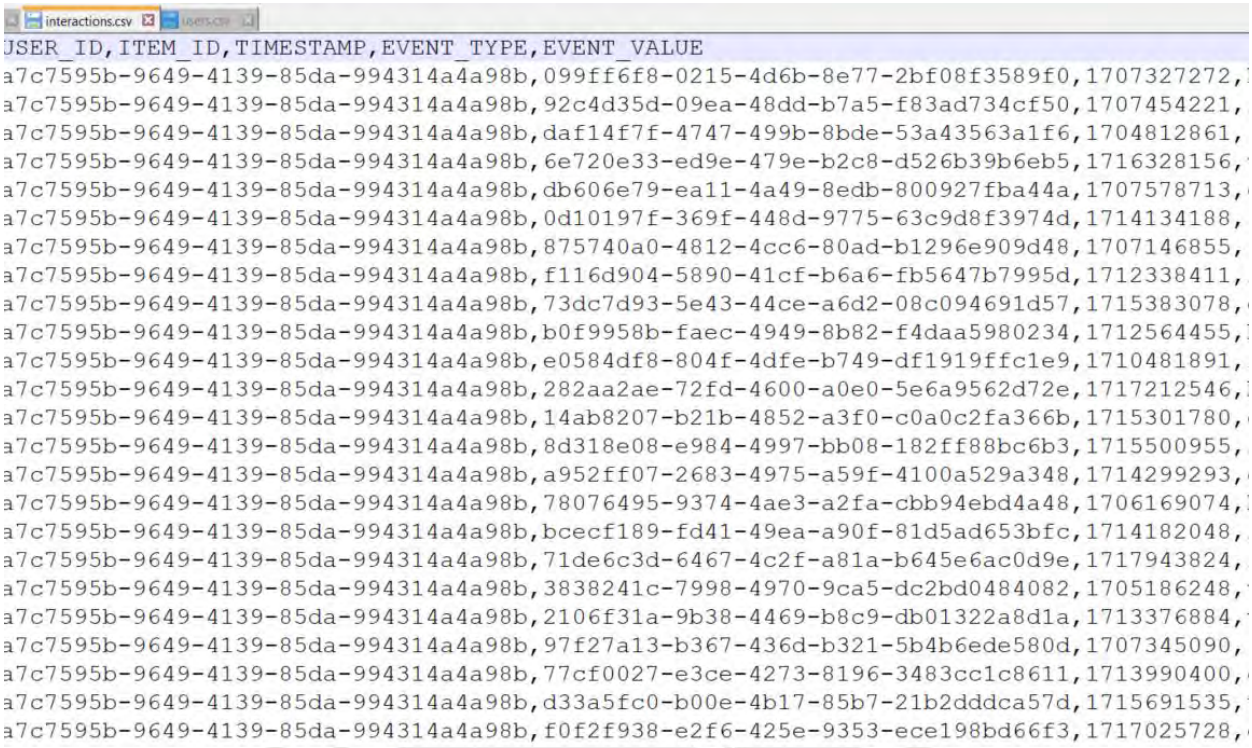
Para cumplir con estos requisitos, se generan datos ficticios que simulan interacciones entre usuarios y eventos o músicos. Esta base inicial permite alimentar el modelo sin necesidad de contar con datos reales desde el inicio.

Se elabora un script SQL para poblar la base de datos con usuarios, elementos e interacciones. Posteriormente, la información es exportada en archivos CSV con la estructura exigida por AWS Personalize para los *datasets*:

- Interacciones
- Usuarios
- Elementos (eventos o músicos)

Como se observa en la siguiente figura, el archivo CSV de interacciones contiene los campos requeridos para iniciar el proceso de entrenamiento.

Los archivos CSV se almacenan en un *bucket* de Amazon S3, donde AWS Personalize puede acceder para la creación de *datasets*.


interactions.csv

```
USER_ID,ITEM_ID,TIMESTAMP,EVENT_TYPE,EVENT_VALUE
a7c7595b-9649-4139-85da-994314a4a98b,099ff6f8-0215-4d6b-8e77-2bf08f3589f0,1707327272,
a7c7595b-9649-4139-85da-994314a4a98b,92c4d35d-09ea-48dd-b7a5-f83ad734cf50,1707454221,
a7c7595b-9649-4139-85da-994314a4a98b,daf14f7f-4747-499b-8bde-53a43563a1f6,1704812861,
a7c7595b-9649-4139-85da-994314a4a98b,6e720e33-ed9e-479e-b2c8-d526b39b6eb5,1716328156,
a7c7595b-9649-4139-85da-994314a4a98b,db606e79-ea11-4a49-8edb-800927fba44a,1707578713,
a7c7595b-9649-4139-85da-994314a4a98b,0d10197f-369f-448d-9775-63c9d8f3974d,1714134188,
a7c7595b-9649-4139-85da-994314a4a98b,875740a0-4812-4cc6-80ad-b1296e909d48,1707146855,
a7c7595b-9649-4139-85da-994314a4a98b,f116d904-5890-41cf-b6a6-fb5647b7995d,1712338411,
a7c7595b-9649-4139-85da-994314a4a98b,73dc7d93-5e43-44ce-a6d2-08c094691d57,1715383078,
a7c7595b-9649-4139-85da-994314a4a98b,b0f9958b-faec-4949-8b82-f4daa5980234,1712564455,
a7c7595b-9649-4139-85da-994314a4a98b,e0584df8-804f-4dfe-b749-df1919ffc1e9,1710481891,
a7c7595b-9649-4139-85da-994314a4a98b,282aa2ae-72fd-4600-a0e0-5e6a9562d72e,1717212546,
a7c7595b-9649-4139-85da-994314a4a98b,14ab8207-b21b-4852-a3f0-c0a0c2fa366b,1715301780,
a7c7595b-9649-4139-85da-994314a4a98b,8d318e08-e984-4997-bb08-182ff88bc6b3,1715500955,
a7c7595b-9649-4139-85da-994314a4a98b,a952ff07-2683-4975-a59f-4100a529a348,1714299293,
a7c7595b-9649-4139-85da-994314a4a98b,78076495-9374-4ae3-a2fa-cbb94ebd4a48,1706169074,
a7c7595b-9649-4139-85da-994314a4a98b,bcecf189-fd41-49ea-a90f-81d5ad653bfc,1714182048,
a7c7595b-9649-4139-85da-994314a4a98b,71de6c3d-6467-4c2f-a81a-b645e6ac0d9e,1717943824,
a7c7595b-9649-4139-85da-994314a4a98b,3838241c-7998-4970-9ca5-dc2bd0484082,1705186248,
a7c7595b-9649-4139-85da-994314a4a98b,2106f31a-9b38-4469-b8c9-db01322a8d1a,1713376884,
a7c7595b-9649-4139-85da-994314a4a98b,97f27a13-b367-436d-b321-5b4b6ede580d,1707345090,
a7c7595b-9649-4139-85da-994314a4a98b,77cf0027-e3ce-4273-8196-3483cc1c8611,1713990400,
a7c7595b-9649-4139-85da-994314a4a98b,d33a5fc0-b00e-4b17-85b7-21b2dddca57d,1715691535,
a7c7595b-9649-4139-85da-994314a4a98b,f0f2f938-e2f6-425e-9353-ece198bd66f3,1717025728,
```

Figura 24. Ejemplo de archivo CSV con datos de interacciones para AWS Personalize
Fuente: Elaboración propia

4.5.3 Creación del dataset group, datasets y solución en AWS Personalize

A continuación, el proceso de configuración del sistema de recomendaciones:

- Creación del “Dataset Group”, que agrupa los *datasets* relacionados a un caso de uso específico.
- Carga de los tres *datasets* (interacciones, usuarios y elementos) a partir de los archivos CSV ubicados en S3.
- Creación de una “Solución” utilizando la receta aws-user-personalization-v2, adecuada para generar recomendaciones personalizadas.
- Implementación de una “Campaña”, que permite realizar inferencias en tiempo real sobre el modelo entrenado.

En la siguiente figura se evidencia la creación exitosa de la campaña, lo que confirma que el modelo entrenado quedó habilitado para generar recomendaciones.

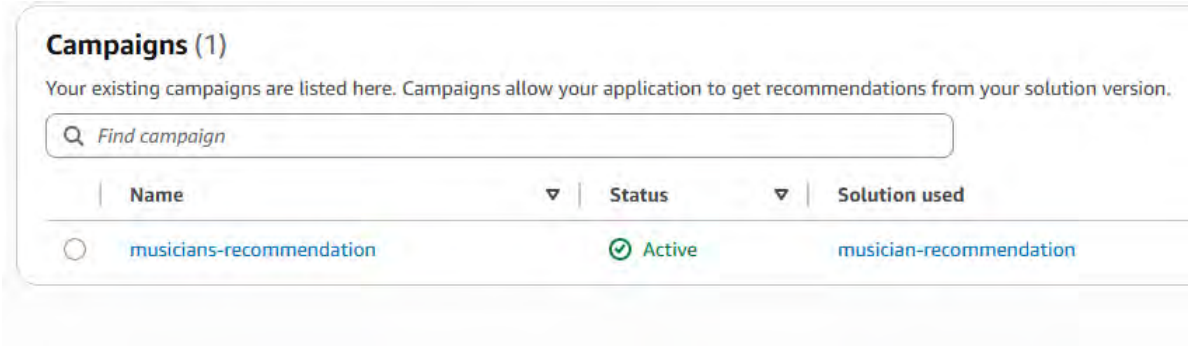


Figura 25. Creación exitosa de la campaña en AWS Personalize
Fuente: Elaboración propia

4.5.4 Actualización en tiempo real mediante eventos de interacción

Una vez desplegada la campaña de recomendaciones, se habilita la capacidad de actualizar dinámicamente el modelo mediante el registro de eventos en tiempo real. Esto permite incorporar nuevas interacciones de los usuarios sin requerir procesos de reentrenamiento periódico.

Para ello, se configura un "Event Tracker" en la consola de Amazon Personalize, recurso que vincula las interacciones en línea con los datos previamente entrenados. Al crear el *tracker* se genera un identificador único (Tracking ID), utilizado por el *backend* para registrar correctamente los eventos, como se muestra en la figura.

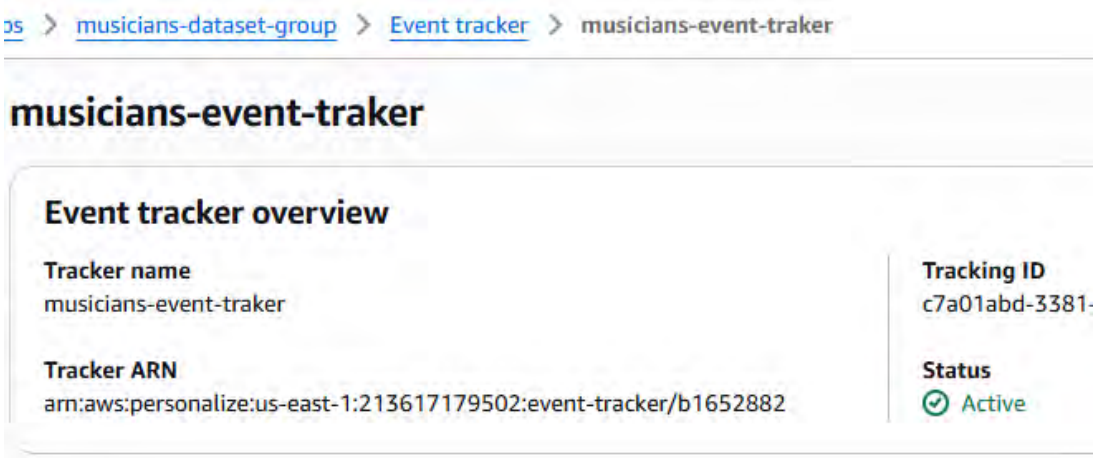


Figura 26. Tracker configurado para el registro de eventos dinámicos
Fuente: Elaboración propia

En la capa *backend* del aplicativo, se integran las bibliotecas oficiales del SDK de AWS para los módulos PersonalizeRuntime y PersonalizeEvents. Estas permiten consumir recomendaciones en tiempo real y registrar nuevos eventos de interacción desde la lógica del sistema.

Cada vez que un usuario realiza una acción sobre un elemento, ya sea un evento o un músico, esta se registra y se transmite a través de la API correspondiente. Las acciones consideradas incluyen:

- HIRE (contratación de un músico)
- INVITE (invitación a un evento)
- CLICK (clic en detalles de eventos o perfiles)
- VIEW (visualización de contenido)
- RATE (valoración asignada por el usuario)

Estas interacciones se jerarquizan según su grado de compromiso, de modo que las acciones como HIRE o INVITE, que reflejan mayor intención del usuario, poseen un peso superior frente a otras de menor impacto como VIEW. Esta priorización mejora la precisión del sistema de recomendaciones, permitiendo adaptar los resultados de manera continua según el comportamiento más reciente de los usuarios dentro de la aplicación.

4.6 Contenerización y despliegue de la API en GCP

La API se desarrolla utilizando Spring Boot y para su despliegue en la nube se genera una imagen Docker a partir de un archivo *Dockerfile*, el cual define el entorno y las dependencias necesarias para ejecutar la aplicación.

Con este archivo, se construye una imagen local de la aplicación. Luego, la imagen se etiqueta y se sube (*push*) al servicio Artifact Registry de Google Cloud Platform (GCP) con los siguientes comandos:

```
docker tag <nombre-local-imagen> <REGION>-
docker.pkg.dev/<NOMBRE_PROYECTO>/<REPO>/<NOMBRE_IMAGEN>:<TAG>
```

```
docker push <REGION>-docker.pkg.dev/<NOMBRE_PROYECTO>/<REPO>/<NOMBRE_IMAGEN>:<TAG>
```

Finalmente, se despliega la imagen en Cloud Run con el siguiente comando, que especifica el servicio, la imagen, la región, el puerto, permite acceso no autenticado y configura la conexión con una instancia de Cloud SQL previamente creada y configurada:

```
gcloud run deploy <NOMBRE_SERVICIO> \
  --image=<REGION>-docker.pkg.dev/<NOMBRE_PROYECTO>/<REPO>/<NOMBRE_IMAGEN>:<TAG> \
  --platform=managed \
  --region=<REGION> \
  --allow-unauthenticated \
  --port=<PUERTO> \
  --add-cloudsql-instances=<NOMBRE_PROYECTO>:<REGION>:<NOMBRE_INSTANCIA_SQL>
```

Una vez desplegada la imagen en Cloud Run, se puede verificar que el servicio fue creado correctamente con el siguiente comando:

```
gcloud run services list --platform=managed --region=<REGION>
```

Este comando mostrará una lista de servicios activos en Cloud Run dentro de la región especificada.

Figura 27. Servicio desplegado en Cloud Run
Fuente: Elaboración propia

Para asociar un dominio personalizado, primero se verifica la propiedad del dominio y se crea el mapeo del dominio al servicio desplegado mediante los siguientes comandos, siguiendo la documentación oficial.

```
gcloud domains verify <TU_DOMINIO>

gcloud beta run domain-mappings create --service=<NOMBRE_SERVICIO> --
domain=<TU_DOMINIO> --region=<REGION_DEL_SERVICIO>
```

Finalmente, se confirma el mapeo correcto del dominio, como se evidencia en la siguiente figura.

```
C:\Users\Lenovo>gcloud beta run domain-mappings list --region=us-central1
   DOMAIN          SERVICE        REGION
+  audifylink.xyz  beatblink-api  us-central1

C:\Users\Lenovo>gcloud domains list-user-verified
ID
audifylink.xyz
```

Figura 28. Confirmación del mapeo del dominio personalizado en Cloud Run
Fuente: Elaboración propia

4.8 Pruebas del aplicativo

En esta sección se presentan las pruebas funcionales realizadas al aplicativo, con el objetivo de verificar su correcto funcionamiento.

Las pruebas abarcan los principales flujos de trabajo y funcionalidades esenciales de la aplicación móvil, como la creación y edición de eventos, la postulación e invitación de músicos, la comunicación por chat, la selección de músico, el proceso de pago mediante la pasarela Stripe, y el almacenamiento de contenido multimedia (imágenes y videos).

4.8.1 Almacenamiento de imágenes y videos

Se valida el correcto funcionamiento del almacenamiento de archivos multimedia en Cloud Storage. Conforme a la estructura establecida previamente, cada usuario posee una carpeta individual que contiene su foto de perfil y los archivos de su repositorio. Durante las pruebas, se comprobó que esta estructura se respeta y que los archivos se almacenan correctamente.

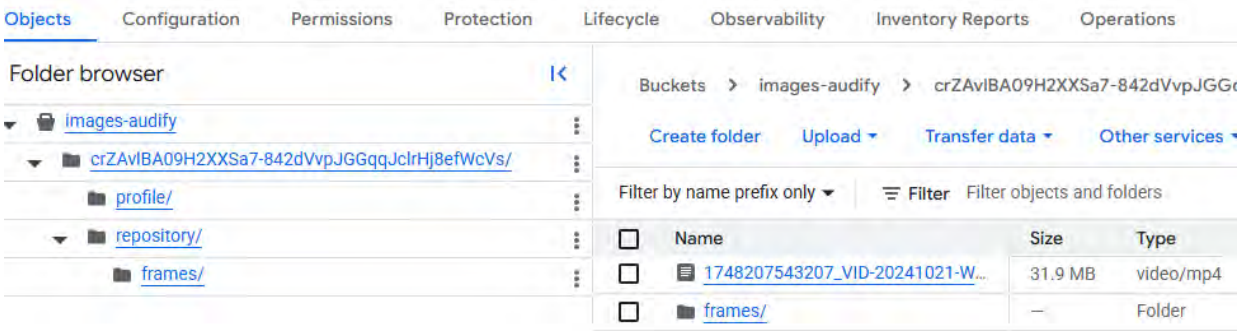


Figura 29. Repositorio y foto de perfil de usuario en Cloud Storage
Fuente: Elaboración propia

En primer lugar, se realizó la prueba de subida de una foto de perfil. El archivo fue almacenado en la carpeta correspondiente del usuario y se generó correctamente el enlace público, lo cual permitió visualizar la imagen directamente en el perfil.

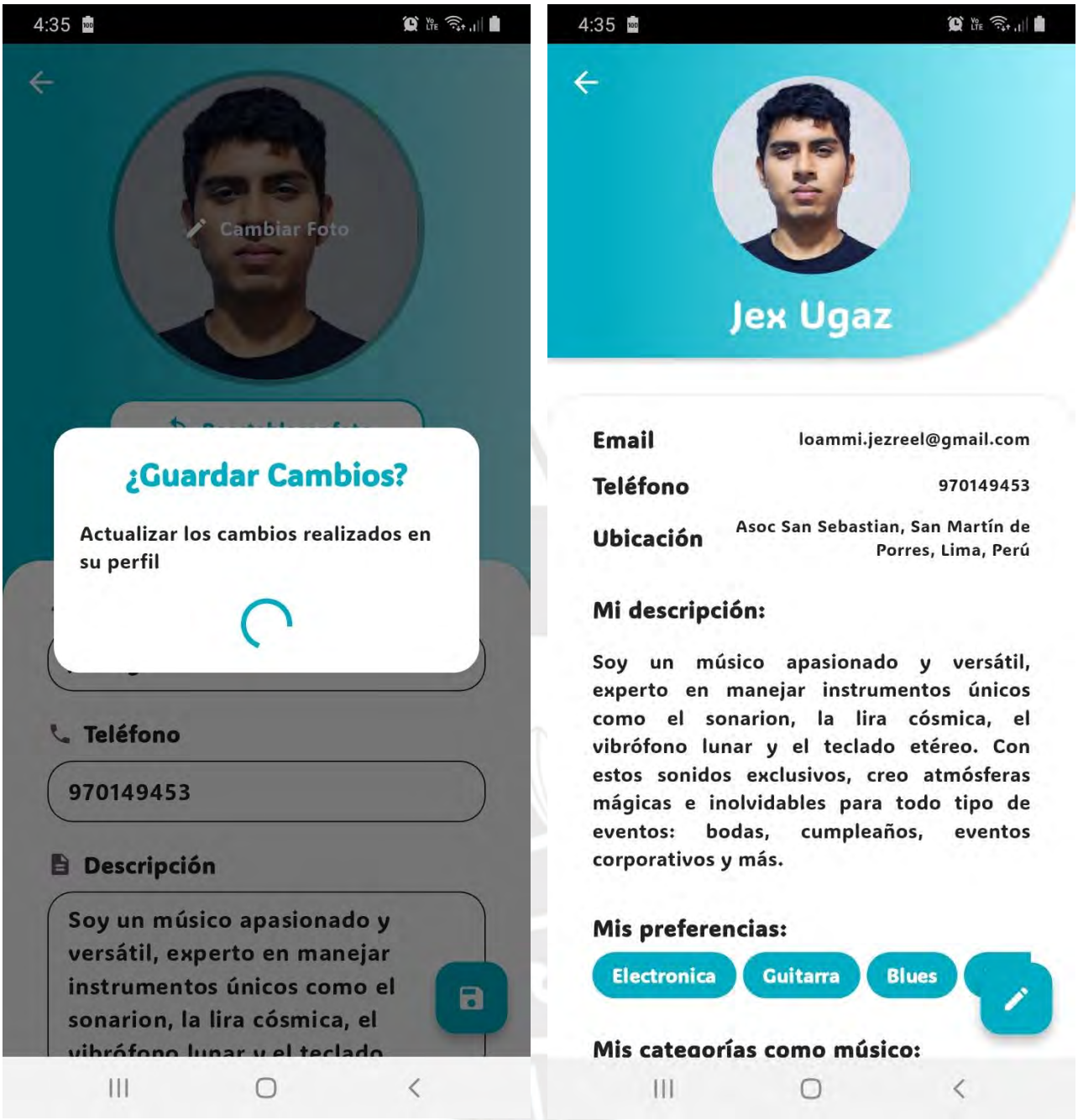


Figura 30. *Actualización de foto de perfil del usuario*
Fuente: Elaboración propia

A continuación, se evaluó la subida de un video al repositorio del mismo usuario. Para esta prueba, se añadió un video con una descripción aleatoria. El contenido se muestra correctamente en la sección de repositorio, junto con su descripción.

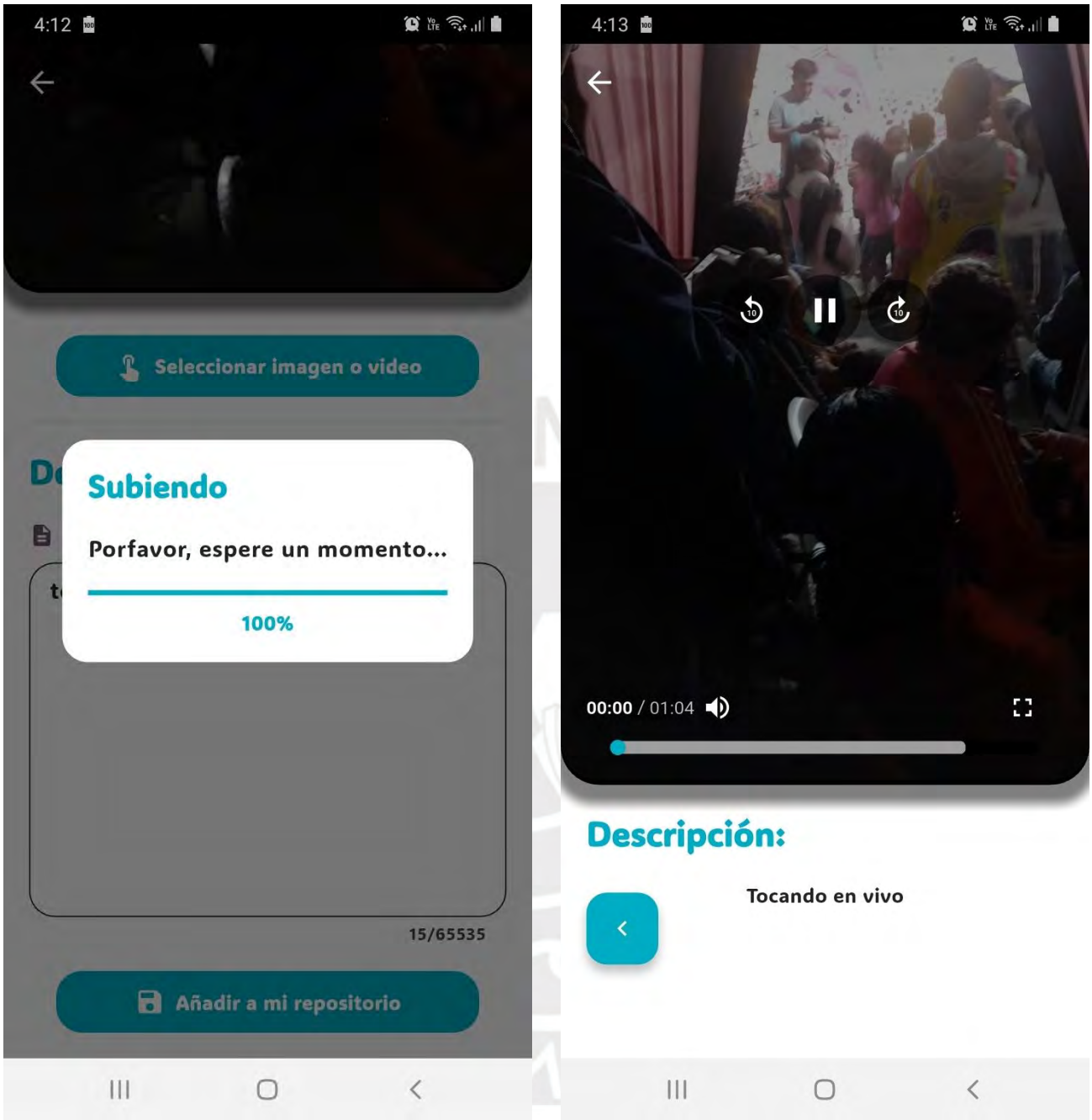


Figura 31. *Subida de video al repositorio del usuario*
Fuente: Elaboración propia

4.8.2 Creación y de edición de eventos

Se validó la creación exitosa de un evento con el título "Graduación de Telecos" y fecha programada para el mes de junio. Tras su creación, el sistema redirige al usuario a la vista de detalle del evento, donde se confirma la correcta visualización de la información.

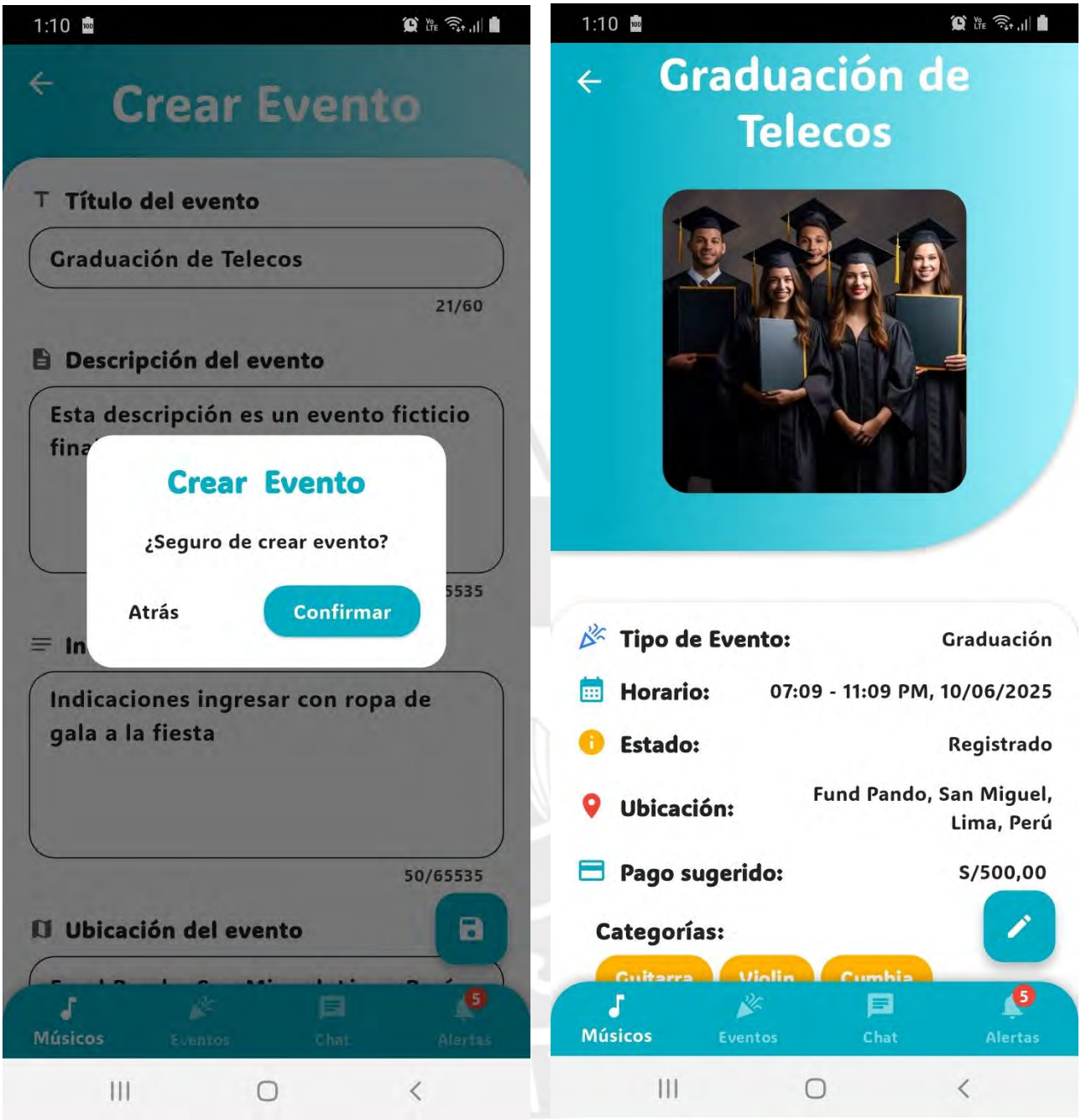


Figura 32. *Prueba Creación de Evento*
Fuente: Elaboración propia

Posteriormente, se verificó la edición del evento, actualizando la fecha del 10 de junio al 11 de julio y modificando la ubicación. Los cambios guardados se reflejaron correctamente en la vista de detalle.

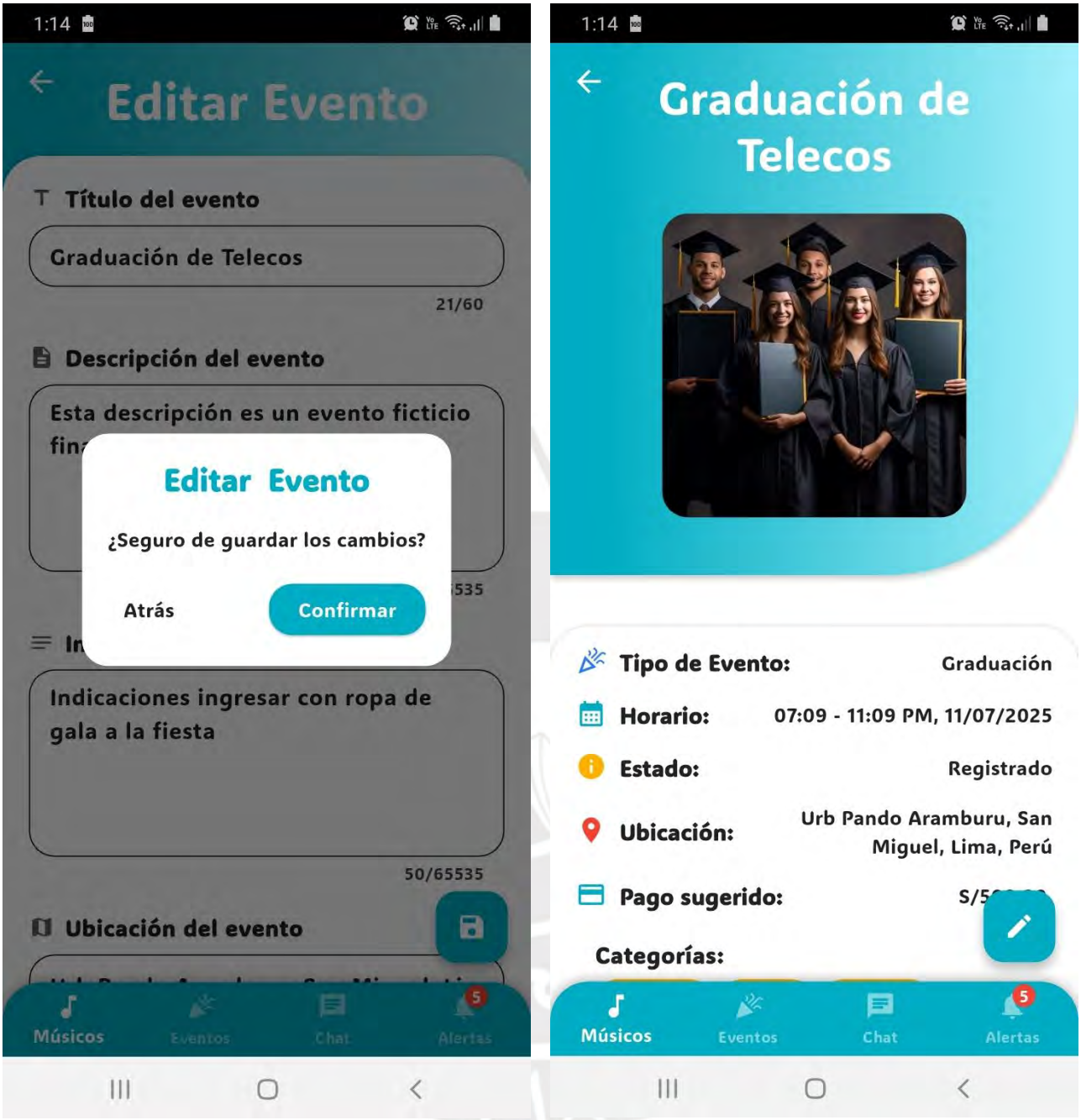


Figura 33. Prueba Edición de Evento
Fuente: Elaboración propia

4.8.3 Invitación a músico

Se probó el envío de invitaciones a músicos desde la vista de detalles, verificando que el proceso se realiza correctamente.

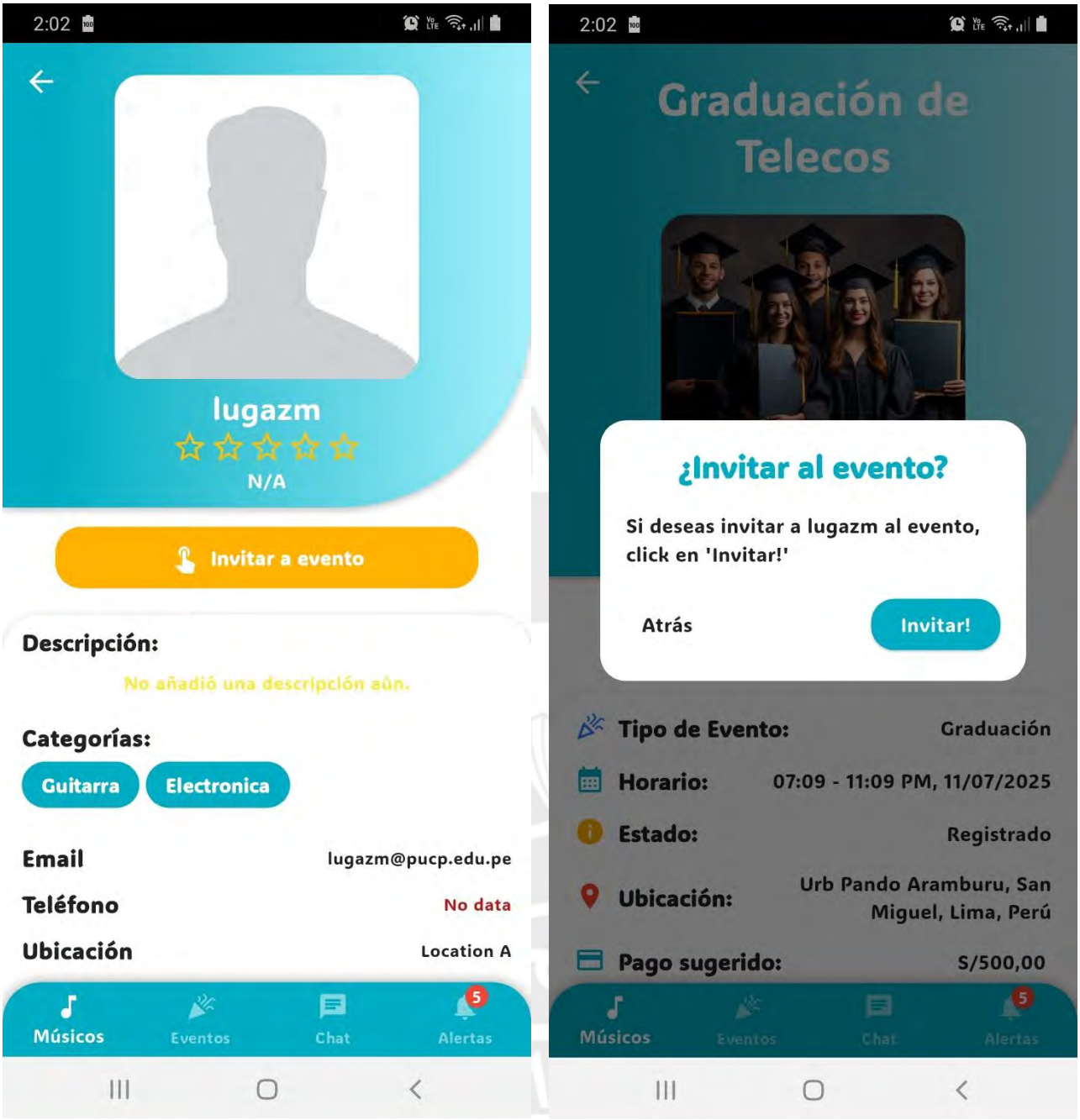


Figura 34. *Envío de invitación a músico para un evento*
Fuente: Elaboración propia

Asimismo, se comprobó que el músico recibe la notificación inmediata a través del sistema de alertas dentro de la aplicación.

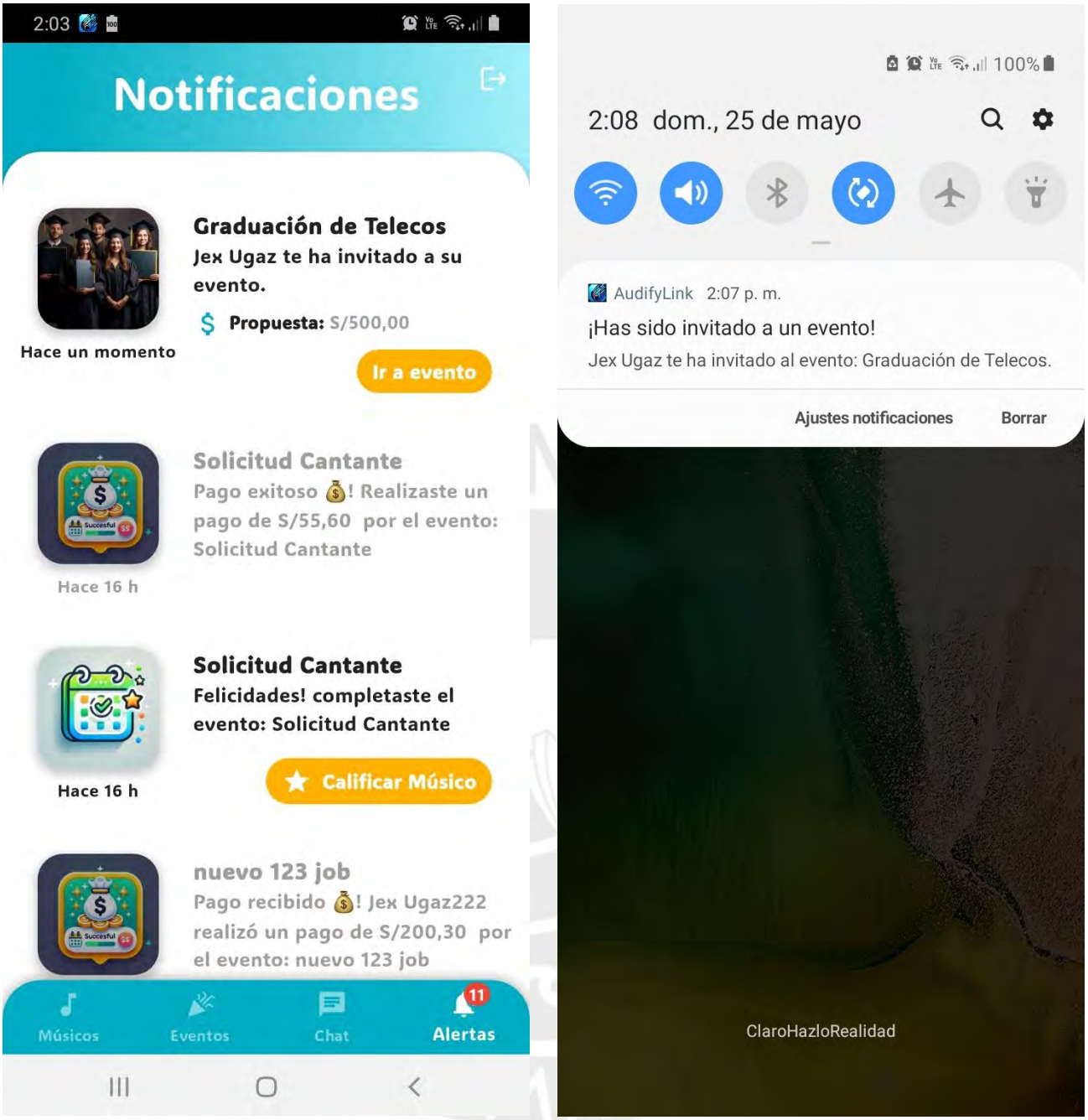


Figura 35. Músico recibe notificación de invitación a evento
Fuente: Elaboración propia

4.8.4 Postulación de músico a evento

Se comprobó que el músico puede postularse a un evento, ya sea desde una invitación recibida o al explorar eventos disponibles. Durante el proceso, se permite ingresar una propuesta de pago dirigida al creador del evento.

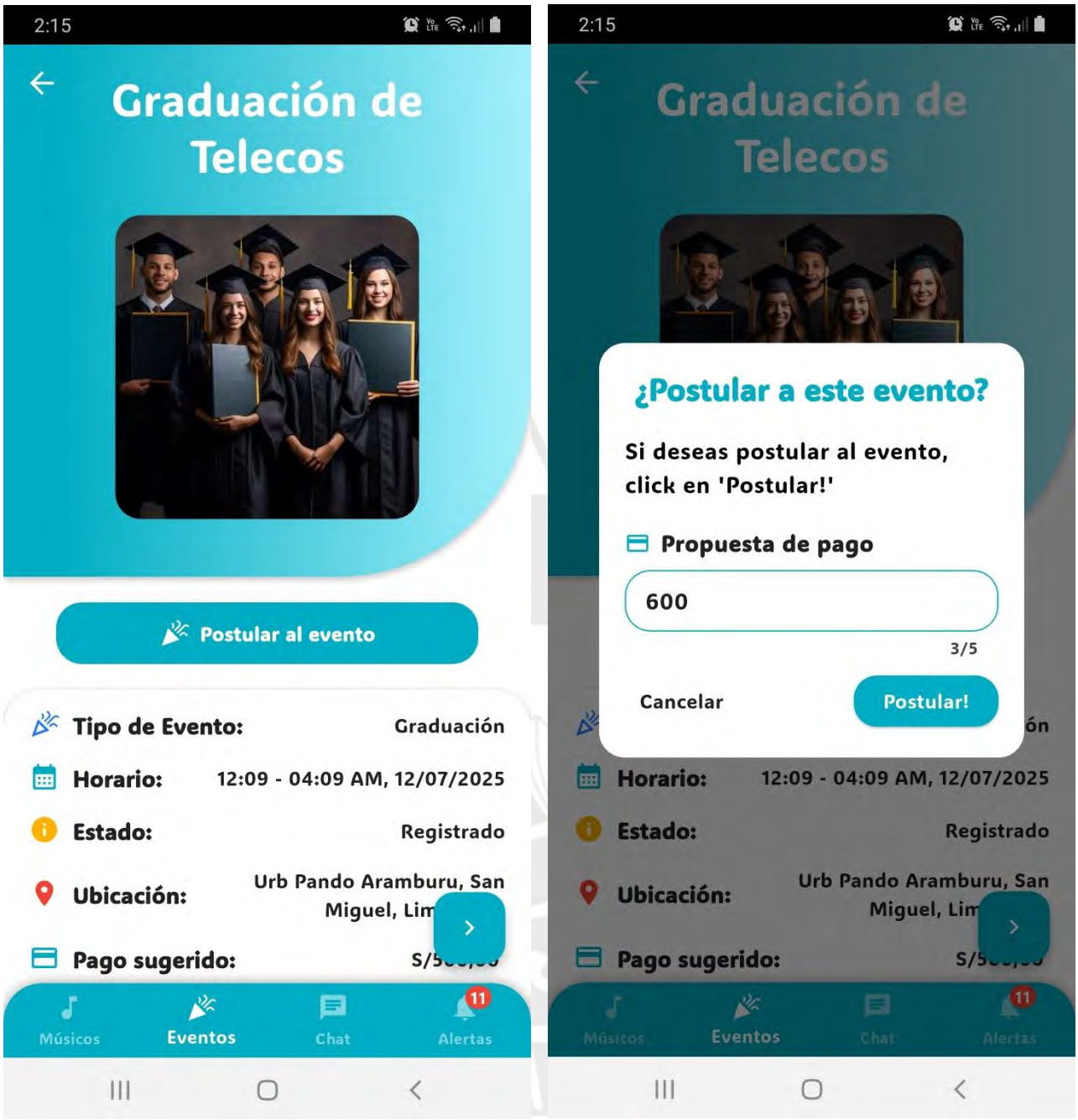


Figura 36. Postulación de músico a un evento
Fuente: Elaboración propia

Como resultado de una postulación exitosa, el creador del evento recibe una notificación informando sobre la nueva postulación.

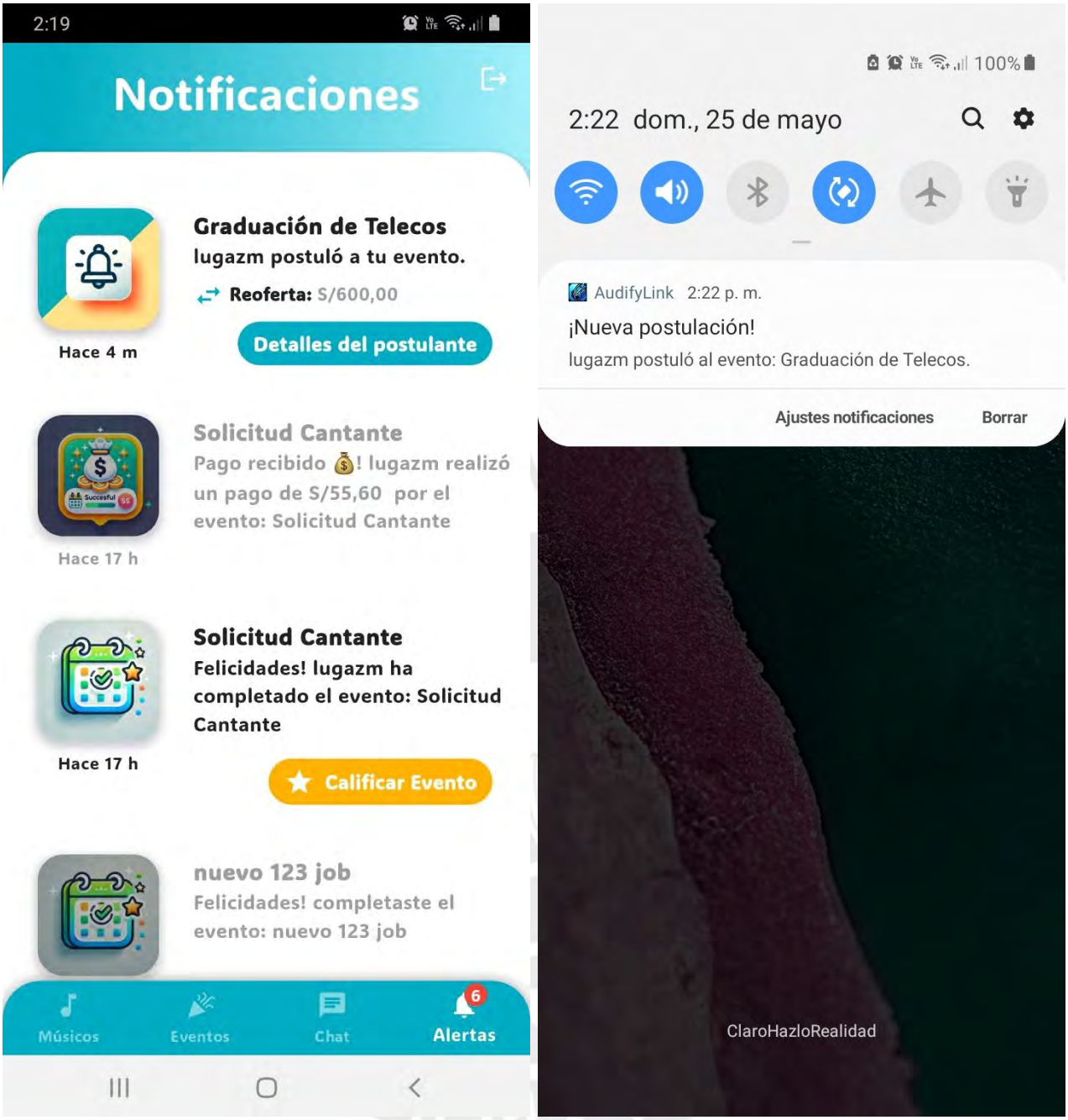


Figura 37. Creador de evento recibe notificación de postulación
Fuente: Elaboración propia

4.8.5 Chat entre músico y creador de evento

Se validó el correcto funcionamiento del chat en tiempo real entre músico y creador del evento, así como la recepción de notificaciones por nuevos mensajes, lo que garantiza una comunicación efectiva entre ambas partes.

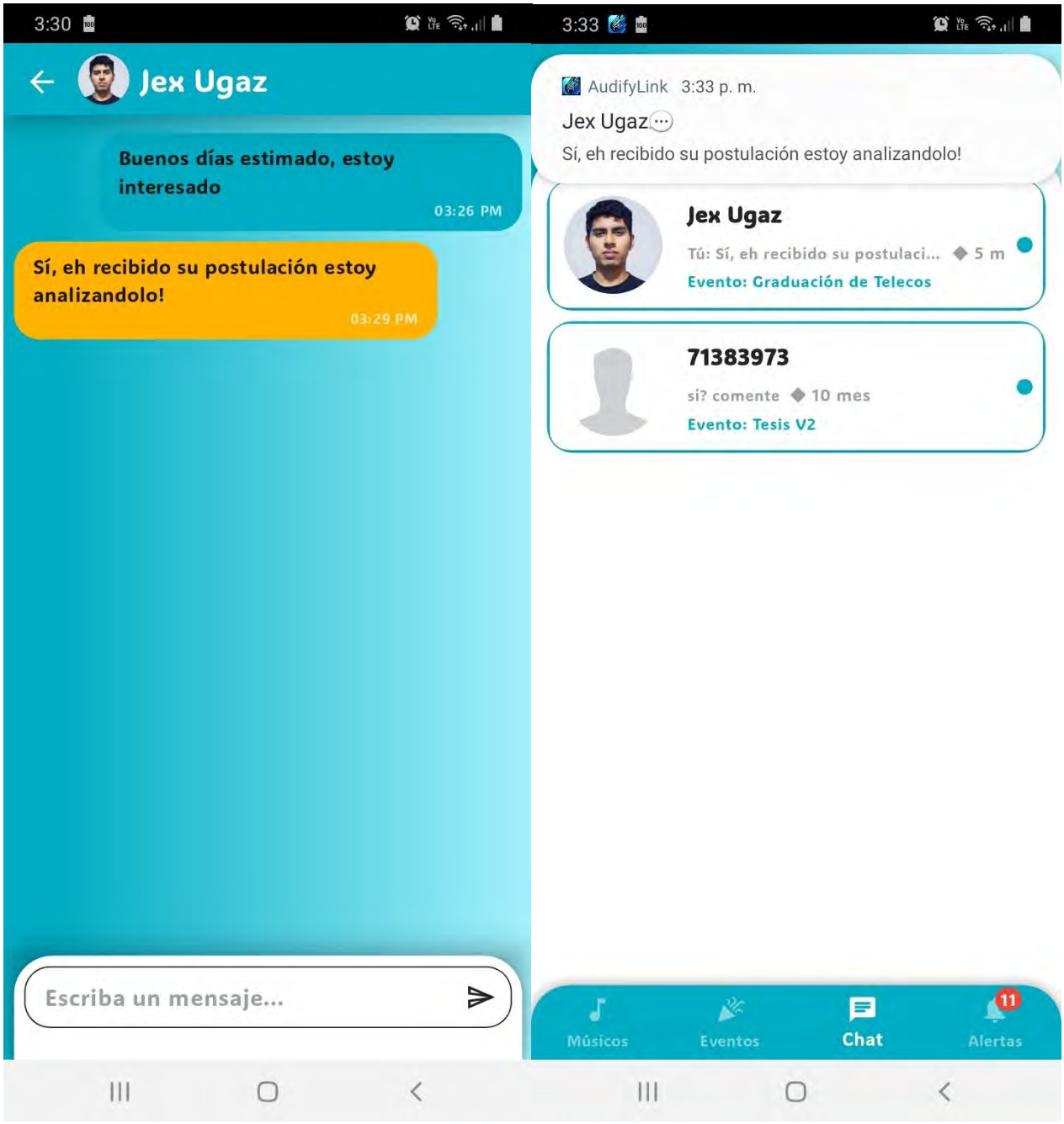


Figura 38. *Chat entre creador de evento y músico*
Fuente: Elaboración propia

4.8.6 Selección de músico

Se comprobó que la selección del músico puede realizarse desde el detalle del postulante o desde la lista de postulantes. Al seleccionar al músico, se asume aceptación del monto propuesto durante la postulación.

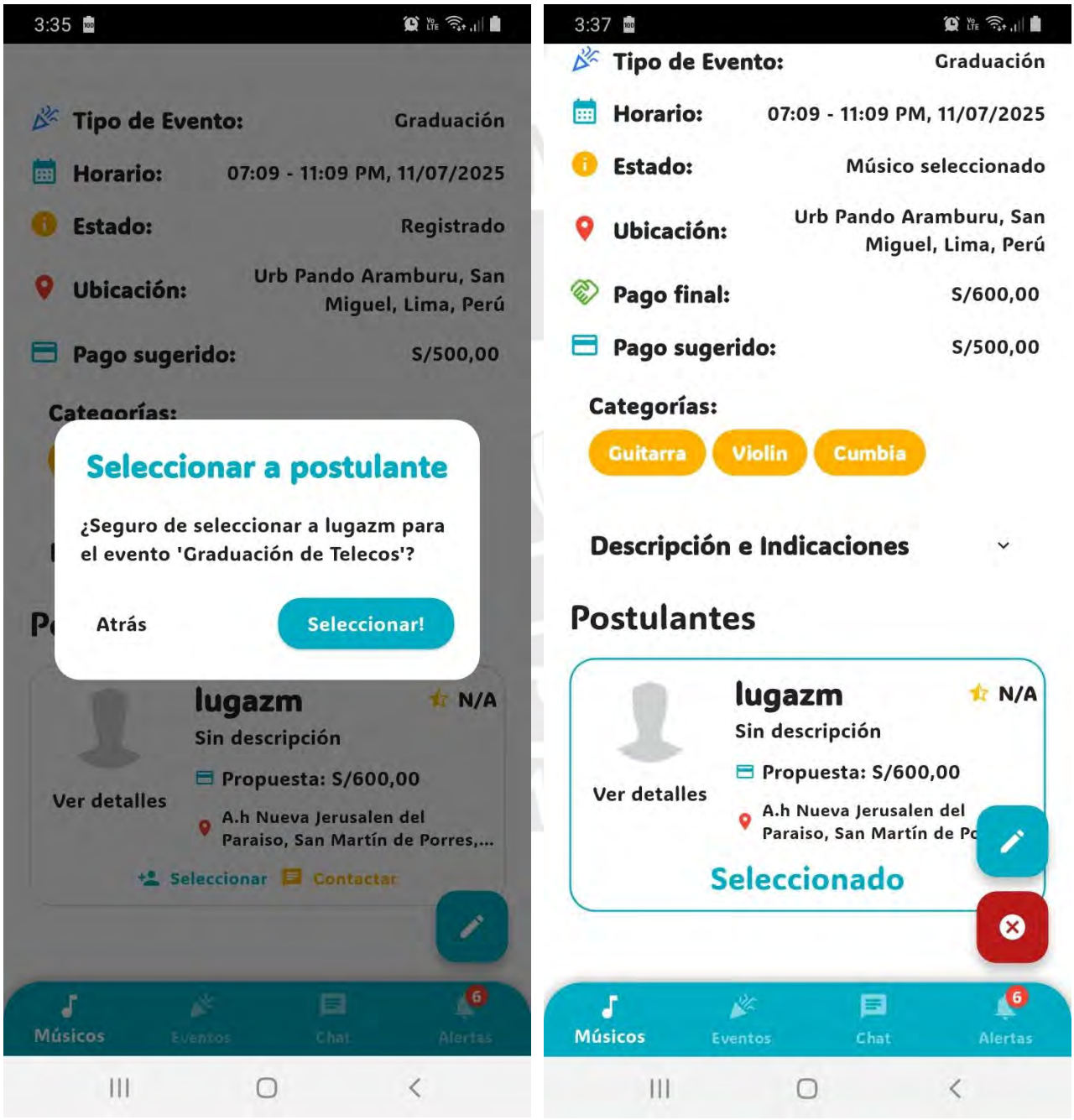


Figura 39. Selección de músico para un evento
Fuente: Elaboración propia

Durante las pruebas, la selección se ejecutó correctamente, mostrando al músico seleccionado con el estado "Seleccionado" en la lista de postulantes. Asimismo, el músico recibió la notificación de su selección, confirmando el funcionamiento adecuado del sistema de notificaciones.

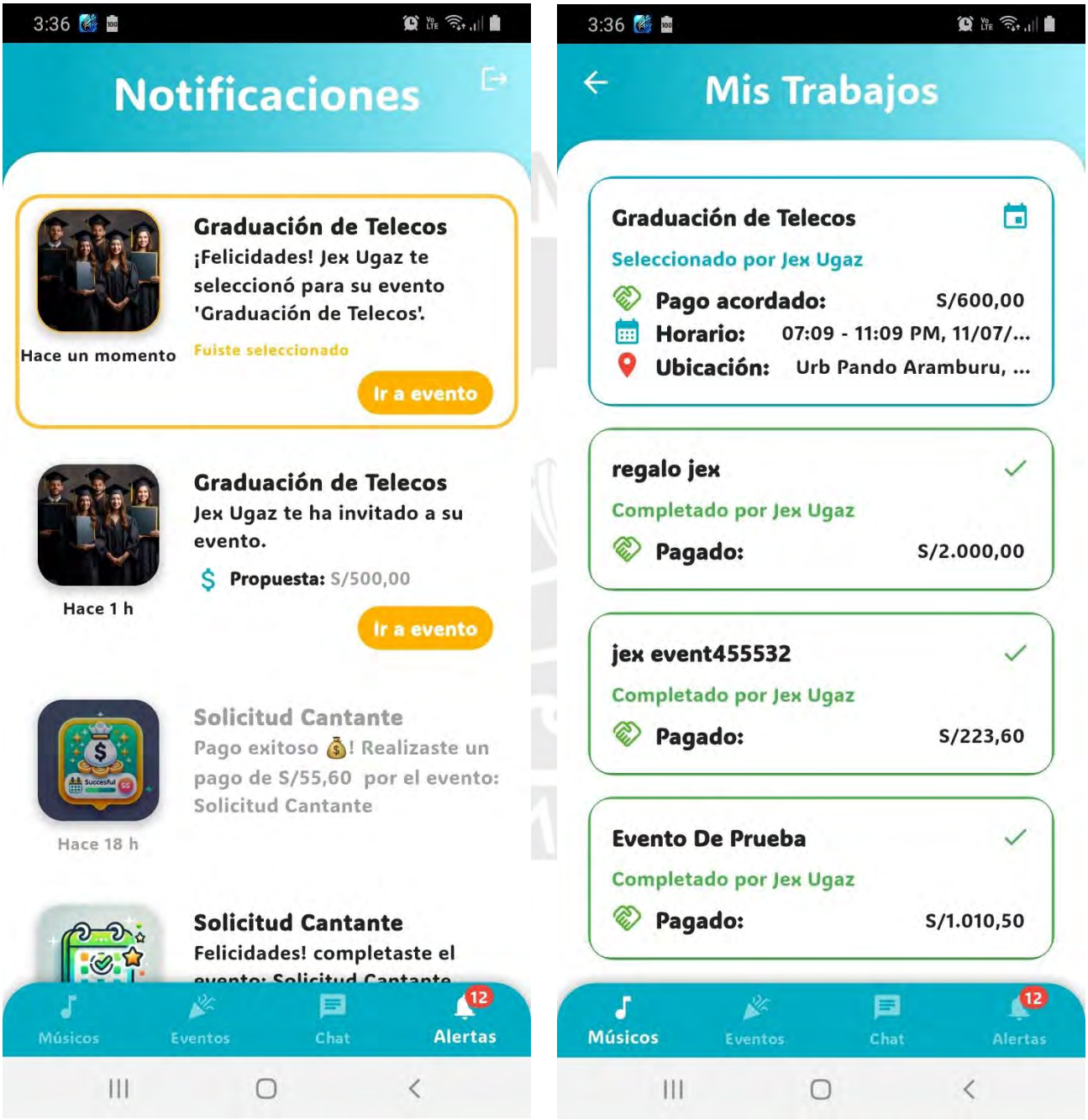


Figura 40. Notificación por selección del músico para un evento y vista de "Mis Trabajos"
Fuente: Elaboración propia

4.8.7 Proceso de pago mediante pasarela de pagos

En esta sección se comprueba el correcto funcionamiento del proceso de pago mediante la pasarela Stripe. Durante las pruebas, el pago se completó correctamente y el estado del evento se actualizó a "Completado" en los detalles del aplicativo.

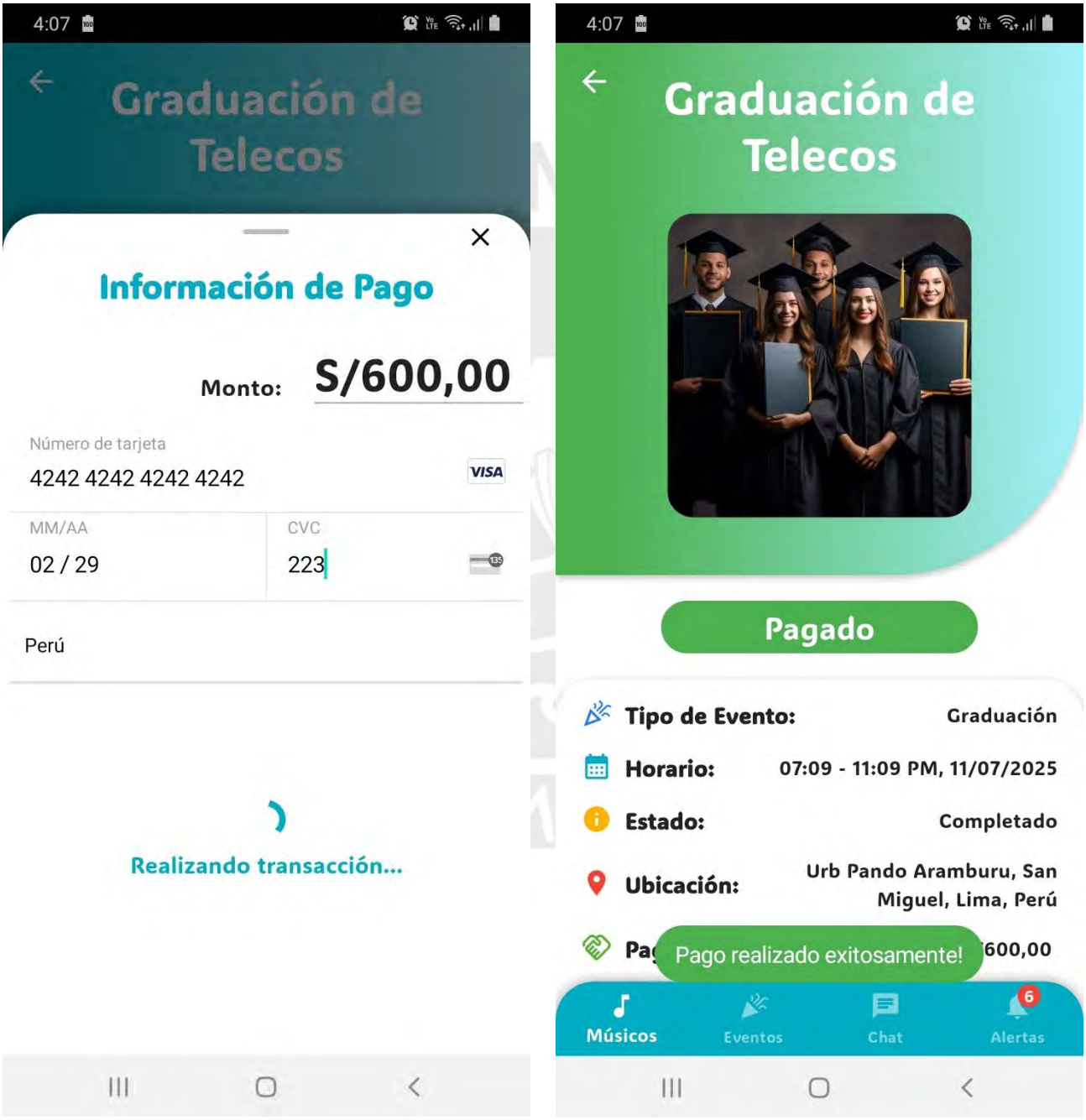


Figura 41. Pago realizado desde la pasarela de pagos de la aplicación
Fuente: Elaboración propia

Asimismo, se verificó que el músico recibe una notificación inmediata informando que se le ha realizado el pago correspondiente.

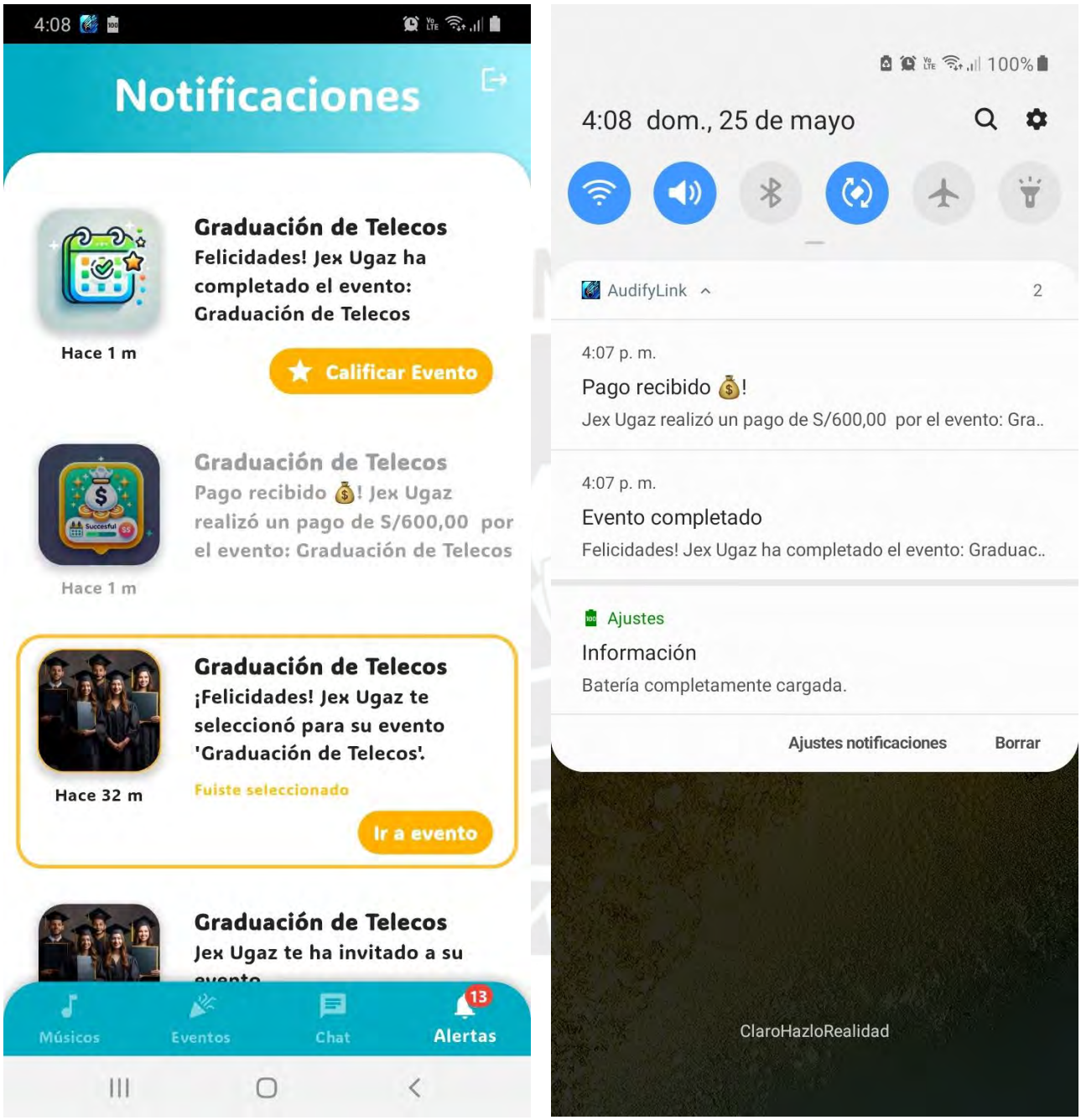


Figura 42. Confirmación de pago por notificación
Fuente: Elaboración propia

Finalmente, en la siguiente figura se muestra el registro exitoso del último pago en el *dashboard* de la pasarela de pagos Stripe, correspondiente al monto de 600 soles por el evento "Graduación de Telecos".

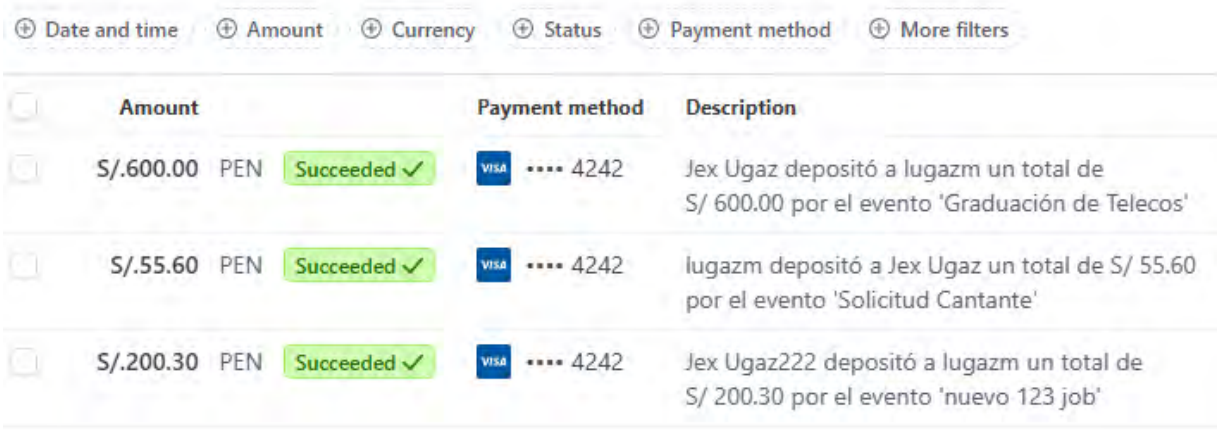


Figura 43. Registro de pago en el dashboard de Stripe
Fuente: Elaboración propia

4.9 Análisis de costos

El análisis de costos se divide en dos partes: CAPEX (gasto de capital) y OPEX (gasto operativo). El CAPEX incluye la inversión en horas de trabajo dedicadas al diseño, construcción y despliegue de la solución. Por otro lado, el OPEX abarca los costos de mantener la aplicación en la nube después de su despliegue.

Para efectos del análisis operativo (OPEX), se proyecta un crecimiento de usuarios de la siguiente manera: 10 usuarios en el primer año, 50 en el segundo, 100 en el tercero y 200 en el cuarto año. Esta proyección será usada como base para estimar el consumo en los servicios en la nube (Firebase, Google Cloud Platform, AWS).

4.9.1 Costos de mano de obra

Se considera un sueldo de S/ 1 025.00 para un practicante (condición del tesista responsable), con una jornada laboral de 30 horas semanales (120 horas al mes). El tipo de cambio usado es de 3.65 soles por dólar. Con ello, el salario por hora es de S/ 8.54, equivalente a S/ 51.25 por día (6 horas diarias).

Tabla 8. Cotización de mano de obra

PARTE	OBRA	Responsable	Días Trabajados	Salario por Día (S/)	Total (S/)
NUBE	Diseño de arquitectura de nube	Tesista	5	51.25	256.25
	Despliegue de la arquitectura en la nube		5	51.25	256.25
	SUBTOTAL		10		512.5
APP MÓVIL	Diseño y construcción del aplicativo móvil	Tesista	45	51.25	2306.25
	SUBTOTAL		45		2306.25
API	Construcción y despliegue de la API	Tesista	15	51.25	768.75
	SUBTOTAL		15		768.75
BASE DE DATOS	Diseño y construcción de la base de datos en Firestore y MySQL	Tesista	10	51.25	512.5
	SUBTOTAL		10		512.5
QA	Trabajo de QA	Tesista	10	51.25	512.5
	SUBTOTAL		10		512.5
TOTAL			90		**4612.5**

Fuente: Elaboración propia

En la tabla anterior, se detalla la estimación de costos asociada al trabajo realizado por el tesista en las distintas fases del desarrollo del sistema, incluyendo nube, aplicación móvil, API, base de datos y pruebas de calidad (QA).

En conclusión, el costo total del CAPEX es S/4 612.5.

4.9.2 Costos de servicios Firebase

En esta sección se analizan los costos operativos asociados al uso de los servicios de Firebase en la solución propuesta. Se consideran los siguientes componentes: Authentication, Functions, Messaging y Firestore.

La estimación se basa en los límites del plan gratuito de Firebase y el consumo proyectado de cada servicio en función del crecimiento de usuarios planteado. A continuación, se presenta el detalle de las condiciones de cada servicio:

Tabla 9. Límites y precios estimados de servicios de Firebase

FIREBASE	
Servicio	**Detalle**
Authentication	Hasta 50 000 usuarios activos gratis
Functions (Memoria usada)	Hasta 400 000 GB-segundo gratis, Costo adicional: $0.0000025 por GB-segundo (estimado)
Functions (Invocaciones)	Hasta 2 000 000 invocaciones gratis Costo adicional: $0.40 por millón de invocaciones (estimado)
Messaging	Gratuito
Firestore (Datos almacenados)	Límite gratuito: 1 GiB
Firestore (Escritura)	Límite gratuito: 600 000 escrituras por mes
Firestore (Lectura)	Límite gratuito: 1 500 000 lecturas por mes

Fuente: Elaboración propia

En los cálculos realizados no se consideran los servicios Firebase Authentication ni Messaging, dado que Messaging es un servicio gratuito y el crecimiento proyectado de usuarios no supera el límite gratuito establecido para Authentication.

A continuación, se presentan los costos estimados para Firebase Functions:

Tabla 10. Costos estimados para Firebase Functions

Firebase Functions					
Año	**Usuarios**	**Invocaciones/mes**	**GB-segundo/mes**	**Total Mensual ($)**	**Total Anual ($)**
1	10	500	30	0.00	0.00
2	50	2 500	150	0.00	0.00
3	100	5 000	300	0.00	0.00
4	200	10 000	600	0.00	0.00

Fuente: Elaboración propia

Asimismo, se presentan los costos estimados para Firebase Firestore:

Tabla 11. Costos estimados para Firebase Firestore

Firebase Firestore						
Año	**Usuarios**	**Lecturas/mes**	**Escrituras/mes**	**Datos Almacenados (MB)**	**Total Mensual ($)**	**Total Anual ($)**
1	10	4 000	1 500	0.5	0.00	0.00
2	50	20 000	7 500	2.5	0.00	0.00
3	100	40 000	15 000	5	0.00	0.00
4	200	80 000	30 000	10	0.00	0.00

Fuente: Elaboración propia

Los resultados del análisis de costos indican que, bajo el crecimiento de usuarios proyectado y las condiciones establecidas, los servicios de Firebase Functions y Firestore no generarán costos adicionales durante los cuatro años.

4.9.3 Costos en Google Cloud Platform

Los componentes considerados en la estimación de costos son Cloud Run, Cloud SQL, Cloud Storage y Cloud CDN.

La replicación síncrona en Cloud SQL se habilita a partir del tercer año, dado que durante los dos primeros años el número de usuarios es reducido. La aplicación se encuentra desplegada en Cloud Run, lo que permite escalar horizontalmente de forma automática en función del

volumen de solicitudes recibidas. De manera complementaria, Cloud SQL se ajusta verticalmente aumentando su capacidad de cómputo, memoria y almacenamiento conforme se incremente la base de usuarios.

A continuación, se presenta la configuración estimada y el costo mensual proyectado para Cloud Run durante los cuatro años, permaneciendo dentro del nivel gratuito debido al bajo uso:

Tabla 12. Estimación de configuración y costos mensuales en App Run

App Run				
Año	**Usuarios**	**RAM/vCPU**	**Invocaciones/mes**	**Costo Mensual ($)**
1	10	512MB / 0.25	1 000	0.00
2	50	512MB / 0.25	5 000	0.00
3	100	1GB / 0.5	10 000	0.00
4	200	1GB / 0.5	20 000	0.00

Fuente: Elaboración propia

En cuanto a Cloud SQL, los costos mensuales se calcularon considerando un uso continuo de 730 horas al mes. La replicación síncrona se habilita desde el tercer año, con un incremento notable en el costo debido al aumento de recursos necesarios.

Tabla 13. Estimación de costos en Cloud SQL según escalabilidad y replicación

Cloud SQL						
Año	**Usuarios**	**Tipo**	**Almacenamiento**	**Replicación Síncrona**	**Costo Mensual ($)**	**Costo Anual ($)**
1	10	db-f1-micro	10 GB	No	9.25	111.00
2	50	db-g1-small	10 GB	No	16.91	202.92
3	100	db-standard-2	10 GB	Sí	200.42	2 405.04
4	200	db-standard-2	20 GB	Sí	203.02	2 436.24

Fuente: Elaboración propia

Para Cloud Storage, se estima que cada usuario cuenta con hasta siete archivos, con un tamaño promedio de 25 MB por archivo. El almacenamiento total se obtiene multiplicando el número de usuarios, la cantidad de archivos y el tamaño promedio, como se muestra en la siguiente tabla. El costo estimado por GB es aproximadamente $0.03, reflejándose en los valores mensuales y anuales indicados.

Tabla 14. Estimación de almacenamiento y costos en Cloud Storage

Cloud Storage						
Año	Usuarios	Archivos por Usuario	Almacenamiento Total (GB)	Costo por GB ($)	Costo Total Mensual ($)	Costo Total Anual ($)
1	10	7	1.58	0.03	0.04	0.48
2	50	7	7.9	0.03	0.21	2.52
3	100	7	15.8	0.03	0.41	4.92
4	200	7	31.6	0.03	0.82	9.84

Fuente: Elaboración propia

Finalmente, la siguiente tabla presenta la suma anual estimada para todos los servicios de GCP, mostrando un incremento progresivo acorde con la escalabilidad y crecimiento del número de usuarios.

Tabla 15. Costo anual total estimado de servicios de GCP

Costos de GCP	
Año	Costo Anual Total ($)
1	111.48
2	205.44
3	2 409.96
4	2 446.08

Fuente: Elaboración propia

4.9.4 Costos del servicio de personalización con AWS Personalize

Aunque en pruebas se empleó el modo en tiempo real, este no resulta rentable para la cantidad proyectada de usuarios (hasta 200 en el cuarto año), ya que requiere mantener una capacidad mínima de 1 transacción por segundo (1 TPS) activa las 24 horas, generando un costo fijo elevado incluso sin solicitudes [76].

Por ello, se optará por *batch recommendations*, que permite generar recomendaciones de forma programada, almacenarlas y entregarlas luego sin costos por disponibilidad continua.

Cabe destacar que AWS Personalize requiere mínimo 25 usuarios únicos, cada uno con al menos 2 interacciones, y un total de 1,000 interacciones para entrenar el modelo. Por lo tanto, en el primer año (10 usuarios estimados), el servicio no podrá utilizarse.

La siguiente tabla estima los costos anuales asociados al uso del servicio. Se considera un promedio de 5 solicitudes diarias por usuario.

Tabla 16. Estimación de costos por recomendaciones *batch* con AWS Personalize

AWS Personalize				
Año	**Usuarios estimados**	**Solicitudes diarias**	**Solicitudes anuales**	**Costo anual estimado ($)**
1	10	-	-	-
2	50	250	91 250	13.69
3	100	500	182 500	27.38
4	200	1 000	365 000	54.75

Fuente: Elaboración propia

4.9.5 Costo del dominio web

En la tesis, el dominio es adquirido a un costo fijo de S/80.05 anuales, lo cual representa un gasto constante durante los cuatro años de operación considerados. Al tratarse de un servicio esencial y sin variaciones en el precio, no se proyectan incrementos ni cargos adicionales por este concepto.

4.9.6 Resumen consolidado de costos

Finalmente, se presenta un resumen del cálculo de los costos de capital (CAPEX) y los costos operativos (OPEX) asociados al desarrollo y operación del aplicativo durante los cuatro años proyectados.

El único costo de capital considerado es la mano de obra, correspondiente al tiempo de desarrollo e implementación del aplicativo:

Tabla 17. Resumen de costos CAPEX

Costos de CAPEX	
Mano de Obra	4 612.50
Costo Total (S/)	**4 612.50**

A continuación, se resumen los costos operativos anuales (OPEX) relacionados con los principales servicios utilizados en la operación del aplicativo: Google Cloud Platform (GCP), AWS Personalize y el dominio web. Para el cálculo de los servicios cobrados en dólares, se ha considerado un tipo de cambio referencial de S/ 3.65 por dólar.

Tabla 18. Resumen de costos OPEX

Costos de OPEX				
Año	**GCP ($)**	**AWS Personalize ($)**	**Dominio (S/)**	**Total Anual (S/)**
1	111.48	0.00	80.05	**486.95**
2	205.44	13.69	80.05	**879.87**
3	2 409.96	27.38	80.05	**8 976.33**
4	2 446.08	54.75	80.05	**9 208.09**

Fuente: Elaboración propia

4.9.7 Análisis de rentabilidad

Para el lanzamiento de la solución al mercado, se proyecta la creación de una empresa propietaria del aplicativo, encargada de su operación, mantenimiento y evolución. Esta empresa generará ingresos mediante una comisión del 15% por cada evento completado a través de la plataforma.

Para evaluar la rentabilidad de la solución, se establece una tasa de descuento (WACC) del 7.9%, basada en el estudio de KPMG sobre el costo de capital 2023 [77].

De acuerdo con fuentes como La República [78], los ingresos de un músico en Perú son altamente variables. Para este análisis se adopta un escenario conservador, considerando que cada músico obtiene en promedio S/ 200 mensuales a través de eventos gestionados en la aplicación.

Tabla 19. Parámetros para el análisis de rentabilidad

Parámetros	
WACC (%)	7.90%
Comisión a músico por evento completado	15.00%
Promedio ganancia músico/mes	200

Fuente: Elaboración propia

Tabla 20. Análisis de rentabilidad proyectado

N° Usuarios	N° Músicos supuestos	Ganancias Totales/Mes (S/)	Comisión de Ganancias/Mes (S/)	Egresos/Año (S/)	Ingresos/Año (S/)	Flujo de Caja (S/)
				4 612.50	S/ -	-S/ 4 612.50
10	6	1 200.00	180.00	486.95	1 673.05	1 186.10
50	30	6 000.00	900.00	879.87	9 920.13	9 040.26
100	60	12 000.00	1 800.00	8 976.33	12 623.67	3 647.34
200	120	24 000.00	3 600.00	9 208.09	33 991.91	24 783.82
					TIR (%)	104.03%
					VAN (S/)	25 439.60
					WACC (%)	7.90%

Fuente: Elaboración propia

Los resultados indican lo siguiente:

- **VAN (Valor Actual Neto):** S/ 25 439.60. Un VAN positivo indica que el proyecto es rentable, ya que supera la tasa de descuento aplicada.
- **TIR (Tasa Interna de Retorno):** 104.03%. Esta tasa es considerablemente superior al WACC del 7.9%, lo que demuestra un alto nivel de rentabilidad respecto al capital invertido.

En resumen, el análisis financiero confirma que el proyecto es económicamente viable y rentable. Según las proyecciones, la inversión se recupera desde el primer año operativo, generando beneficios sostenibles a medida que crece la base de usuarios.

4.10 Análisis Ambiental y de Sostenibilidad

En el contexto de la presente tesis, no se considera necesario un análisis detallado sobre el impacto ambiental y la sostenibilidad, ya que la solución propuesta es de naturaleza digital y no involucra procesos industriales, emisiones, uso de recursos naturales ni generación de residuos físicos que afecten directamente el entorno ambiental.

4.11 Análisis relacionado a la Salud Pública y Seguridad

Para el contexto de esta tesis, no se considera necesario realizar un análisis específico relacionado con la salud pública y la seguridad, ya que la naturaleza del proyecto no implica riesgos directos en estos ámbitos.

4.12 Análisis relacionado al Bienestar y el Orden Social

La aplicación desarrollada en esta tesis formaliza la interacción entre clientes y músicos, proporcionando seguridad y confiabilidad en el consumo de servicios musicales en Lima Metropolitana. Al reducir el riesgo de fraudes, mencionados en la sección 1.4.2, y promover un mercado estructurado, contribuye al bienestar y orden social.

CONCLUSIONES

1. Se diseñó e implementó un aplicativo móvil con enfoque en la experiencia del usuario y la eficiencia del proceso de contratación de músicos en Lima Metropolitana. La aplicación permite a los clientes explorar perfiles musicales, agendar eventos y compartir contenido multimedia, brindando una solución tecnológica integral que responde a una necesidad real del mercado, conforme al objetivo general planteado.
2. Se utilizó una arquitectura *multicloud* basada en servicios de AWS y Google Cloud Platform para desplegar los servicios *backend* de forma distribuida, lo que permitió alcanzar alta disponibilidad, escalabilidad automática y tolerancia a fallos. Este enfoque *multicloud* refuerza la resiliencia del sistema y permite una gestión eficiente de los recursos computacionales.
3. Se integró un sistema de mensajería en tiempo real utilizando Firebase, que abstrae la complejidad de los protocolos de comunicación como WebSockets, permitiendo una interacción instantánea y confiable entre músicos y usuarios. Esta decisión técnica se orientó a la eficiencia del desarrollo, manteniendo al mismo tiempo la capacidad de escalar a grandes volúmenes de usuarios concurrentes.
4. Se desarrolló un sistema de recomendaciones basado en inteligencia artificial, el cual utiliza datos de interacción y preferencias del usuario para ofrecer sugerencias personalizadas. Esta funcionalidad no solo mejora la experiencia de navegación dentro del aplicativo, sino que también representa una aplicación concreta de algoritmos de *machine learning* en el contexto de plataformas móviles.
5. Se implementaron mecanismos de seguridad orientados a proteger la integridad y confidencialidad de las comunicaciones y de los datos del sistema. Para ello, se utilizó cifrado en tránsito mediante HTTPS (SSL/TLS), autenticación de usuarios mediante JWT y servicios de identidad federada como Google, así como una configuración

restringida de la base de datos en la nube, accesible únicamente desde servicios internos. Además, se integró una pasarela de pagos que cumple con el estándar PCI-DSS, garantizando la seguridad de las transacciones financieras. Estas medidas permiten establecer una arquitectura segura y coherente con los principios fundamentales de la ciberseguridad en sistemas distribuidos

RECOMENDACIONES Y OBSERVACIONES

1. Incluir herramientas de análisis básico que permitan a los músicos ver cuántas visitas recibe su perfil o cuántas veces han sido contratados.
2. Agregar una sección de preguntas frecuentes (FAQ) para resolver dudas comunes de nuevos usuarios sin requerir soporte externo.
3. Incorporar una sección de tutorial interactivo, que se muestre la primera vez que un usuario abre la app.
4. Permitir la personalización de notificaciones, para que el usuario pueda elegir si desea recibir recordatorios de eventos 24h o 1h antes.
5. Ofrecer la opción de mostrar disponibilidad semanal en el perfil del músico, usando un calendario simple, para agilizar la coordinación con los contratantes.

REFERENCIAS BIBLIOGRÁFICAS

[1] Real Academia Española. "Música." Diccionario de la lengua española. https://www.rae.es/drae2001/m%C3%BAsica (acceso oct. 28, 2023).

[2] R. Byron, Ed., Music, Culture and Experience: Selected Papers of John Blacking. Chicago, IL, USA: University of Chicago Press, 1995.

[3] C. Small, Musicking. Middletown, CT: Wesleyan University Press, 1998.

[4] R. Romero, "Panorama de los estudios sobre la música tradicional en el Perú," Lic. tesis, Inst. Riva Agüero, PUCP, Lima, Perú, 1987.

[5] Federación Internacional de la Industria Fonográfica. "Global Music Report 2023." IFPI. Informe anual sobre la industria musical global. https://www.ifpi.org/wp-content/uploads/2020/03/Global_Music_Report_2023_State_of_the_Industry.pdf (acceso oct. 28, 2023)

[6] Tecnología. "Spotify supera el hito de los 500 millones de usuarios." Expansión. https://expansion.mx/tecnologia/2023/04/25/spotify-supera-el-hito-de-los-500-millones-de-usuarios (acceso oct. 28, 2023).

[7] Pollstar. "2022 Year-End Biz Analysis: Record-Setting Year Marked By Bad Bunny, Ed Sheeran & Stadiums." Pollstar. https://news.pollstar.com/2022/12/12/2022-year-end-biz-analysis-record-setting-year-marked-by-bad-bunny-ed-sheeran-stadiums/ (acceso oct. 30, 2023).

[8] Billboard. "2022 Year-End Boxscore Charts." Billboard. https://www.billboard.com/2022-year-end-boxscore-charts/ (acceso oct. 30, 2023).

[9] PwC Perú. "PwC Perú: industria musical y radial generarán USD 201 millones en el 2024 pese al COVID-19." AmCham Perú. https://amcham.org.pe/news/pwc-peru-industria-musical-y-radial-generaran-usd-201-millones-en-el-2024-pese-al-covid-19/ (acceso oct. 30, 2023).

[10] El Comercio. "Megaeventos resurgen en el Perú: Estos son los conciertos con mayor público y recaudación hasta setiembre." El Comercio. https://elcomercio.pe/economia/peru/conciertos-resurgen-en-el-peru-estos-son-los-espectaculos-con-mas-publico-y-mayor-recaudacion-hasta-setiembre-2022-rmmn-noticia/ (acceso oct. 30, 2023).

[11] B. Miles, *Los Beatles: Día a Día*. Barcelona: MA NON TROPPO, 2003.

[12] J. F. Melgar Wong, “La construcción de lo punk en el discurso historiográfico sobre la música de la banda de Los Saicos,” M.A. tesis, Dept. Humanid., Pontificia Univ. Católica del Perú, Lima, Perú, 2020.

[13] Los Saicos Oficial. Los Saicos en 2 a la N - Entrevista. (Ene. 29, 2016). Acceso: Oct. 28, 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=WxWo3pva8xQ

[14] Ventana ROCK. AMEN - ENTREVISTA 2020 (MARCELO MOTTA) (ACÚSTICO). (Mar. 11, 2020). Acceso: Oct. 28, 2023. [Video en línea]. Disponible: https://www.youtube.com/watch?v=j1FDpkJEX5I

[15] D. L. Montes Alvarez, “El músico clásico en el Perú: entre la vocación y la profesión,” tesis de licenciatura, Dept. Humanid., Pontificia Univ. Católica del Perú, Lima, Perú, 2017.

[16] Real Academia Española. "Estafa." Diccionario de la lengua española. https://dle.rae.es/estafa (acceso dic. 1, 2023).

[17] Dirección de Tecnologías de la Información y Comunicaciones. "Anuario Estadístico Policial 2021." Policía Nacional del Perú. https://www.policia.gob.pe/estadisticopnp/documentos/anuario-2021/anuario-estadistico-policial-2021.pdf (acceso dic. 1, 2023).

[18] Agencia Peruana de Noticias (ANDINA). "Policía brinda cinco recomendaciones para ahuyentar a los estafadores en línea." Andina. https://andina.pe/agencia/noticia-policia-brinda-cinco-recomendaciones-para-ahuyentar-a-los-estafadores-linea-953989.aspx (acceso dic. 1, 2023).

[19] La República. "Perú vs. Argentina: intervienen a 30 personas por estafar con entradas adulteradas para partido." La República. https://larepublica.pe/sociedad/2023/10/17/peru-vs-argentina-intervienen-a-30-personas-por-estafar-con-entradas-adulteradas-para-partido-pnp-futbol-en-vivo-eliminatorias-mundial-2026-estafas-1288906 (acceso dic. 1, 2023).

[20] Gestión. "Dirincri alerta sobre nueva modalidad de estafa piramidal por redes sociales." Gestión. https://gestion.pe/peru/dirincri-alerta-sobre-nueva-modalidad-de-estafa-piramidal-por-redes-sociales-noticia/ (acceso dic. 1, 2023).

[21] La República. "Mario Hart cae en una estafa y queda varado en la ciudad de Trujillo." La República. https://larepublica.pe/espectaculos/2023/06/12/mario-hart-cae-en-una-estafa-y-queda-varado-en-la-ciudad-de-trujillo-estafan-a-mario-hart-dejan-varado-a-mario-hart-mdga-1031208 (acceso dic. 1, 2023).

[22] La República. "Grupo 5 denunció que cuentas de Facebook vendieron entradas falsas para su concierto con Josimar." La República.

https://larepublica.pe/espectaculos/2020/11/01/grupo-5-denuncio-que-cuentas-de-facebook-vendieron-entradas-falsas-para-su-concierto-con-josimar (acceso dic. 1, 2023).

[23] Peru.com. "Ráfaga denuncia a falso grupo que estafa a peruanos." Peru.com. https://peru.com/entretenimiento/musica/rafaga-anuncia-falso-grupo-que-estafa-peruanos-noticia-454834/ (acceso dic. 1, 2023).

[24] La República. "Festival Perú Central: bandas locales y extranjeras denuncian irregularidades en el evento." La República. https://larepublica.pe/espectaculos/musica/2022/07/28/festival-peru-central-bandas-locales-afirman-haber-sido-estafadas-por-los-organizadores-del-evento-fotos-instagram (acceso dic. 1, 2023).

[25] Banda Orquesta Santa Cecilia Bendita, "Lamentamos profundamente tener que compartir este tipo de noticias. Fuimos víctimas de una estafa…", Facebook. https://www.facebook.com/permalink.php?story_fbid=pfbid02GC1pbvSLL9X7qB6ny7pV2vLh7DbxxpsJ4Ji19qoTVYuSDZBXTbfvM2z4mzwRnq6bl&id=100010842254233 (acceso Jun. 11, 2025).

[26] De Tecnología y Otras Cosas. "Sistemas Operativos Móviles." De Tecnología y Otras Cosas. https://dtyoc.com/2016/10/03/sistemas-operativos-moviles/ (acceso dic. 1, 2023).

[27] StatCounter. "Mobile Operating System Market Share Worldwide - November 2023." StatCounter. https://gs.statcounter.com/os-market-share/mobile/worldwide (acceso dic. 1, 2023).

[28] Android Developers. "Arquitectura de la plataforma." Android Developers. https://developer.android.com/guide/platform?hl=es-419#art (acceso dic. 1, 2023).

[29] Medium. "[iOS] Architecture of iOS Operating System." Medium. https://medium.com/@ganeshrajugalla/ios-ios-introduction-and-structure-fdd7ecf08c4c (acceso dic. 3, 2023).

[30] Microsoft. "What is mobile application development?" Microsoft. https://azure.microsoft.com/en-ca/resources/cloud-computing-dictionary/what-is-mobile-app-development (acceso dic. 3, 2023).

[31] Techopedia. "Software Framework." Techopedia. https://www.techopedia.com/definition/14384/software-framework (acceso dic. 3, 2023).

[32] Amazon Web Services. "¿Qué es Flutter?" Amazon Web Services. https://aws.amazon.com/es/what-is/flutter/ (acceso dic. 3, 2023).

[33] Microsoft. "¿Qué es Xamarin?" Microsoft Learn. https://learn.microsoft.com/es-es/xamarin/get-started/what-is-xamarin (acceso dic. 3, 2023).

[34] Inmediatum. "Ventajas y desventajas de apps desarrolladas en Xamarin." Inmediatum. https://inmediatum.com/blog/ingenieria/ventajas-y-desventajas-de-apps-desarrolladas-en-xamarin/ (acceso dic. 3, 2023).

[35] S. Lewis. "What is React Native?" TechTarget. https://www.techtarget.com/searchapparchitecture/definition/React-Native (acceso dic. 3, 2023).

[36] Digitality. "¿Qué es React Native y para qué sirve?" Blog de Digitality. Explicación general sobre el framework de desarrollo móvil. https://www.digitality.es/blog/que-es-react-native-y-para-que-sirve (acceso dic. 3, 2023).

[37] Oracle. "What is Database?" Oracle. Explicación general sobre bases de datos y sus funciones principales. https://www.oracle.com/pe/database/what-is-database/ (acceso dic. 3, 2023).

[38] DB-Engines. "DB-Engines Ranking." DB-Engines. Ranking actualizado de sistemas de gestión de bases de datos. https://db-engines.com/en/ranking (acceso dic. 3, 2023).

[39] Google Cloud. "¿Qué es MySQL?" Google Cloud. Descripción general del servicio de base de datos relacional MySQL. https://cloud.google.com/mysql?hl=es (acceso dic. 3, 2023).

[40] Digital Guide Ionos. "SQLite: la famosa biblioteca en detalle." Ionos Digital Guide. Explicación detallada del motor de base de datos SQLite. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/sqlite/ (acceso dic. 3, 2023).

[41] Microsoft Azure. "¿Qué es PostgreSQL?" Diccionario de computación en la nube de Microsoft. https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-postgresql (acceso dic. 3, 2023).

[42] Stratascratch. "Postgres vs MySQL: Which is Better for Analytics?" https://www.stratascratch.com/blog/postgres-vs-mysql-which-is-better-for-analytics/ (acceso dic. 3, 2023).

[43] IBM. "¿Qué son las bases de datos NoSQL?" https://www.ibm.com/es-es/topics/nosql-databases (acceso dic. 3, 2023).

[44] Amazon Web Services. "¿Qué es una API?" https://aws.amazon.com/es/what-is/api/ (acceso dic. 3, 2023).

[45] Izipay, Lima, Perú. Manual de Integración Samishop (Documento PDF). (2023). Acceso: dic. 3, 2023. [En línea]. Disponible: https://www.izipay.pe/pdf/Manual-integracion-Samishop

[46] F. Inspires, "REST API Communication: Navigating API Components from Request to Response". Medium. https://medium.com/@fatimainspires/rest-api-communication-navigating-api-components-from-request-to-response-ce2fbebb3739 (acceso dic. 3, 2023).

[47] Amazon Web Services. "¿Qué es la informática en la nube?". Amazon Web Services. https://aws.amazon.com/es/what-is-cloud-computing/ (acceso dic. 3, 2023).

[48] CX Today. "Gartner Magic Quadrant for Cloud Infrastructure and Platform Services 2022". CX Today. https://www.cxtoday.com/data-analytics/gartner-magic-quadrant-cloud-infrastructure-platform-services-2022/ (acceso dic. 3, 2023).

[49] Google Cloud. "¿Qué es cloud computing?". Google Cloud. https://cloud.google.com/learn/what-is-cloud-computing?hl=es (acceso dic. 3, 2023).

[50] Organización Internacional de Normalización. "What is artificial intelligence (AI)?". ISO. https://www.iso.org/artificial-intelligence/what-is-ai (acceso jul. 10, 2024).

[51] IBM. "What is Artificial Intelligence (AI)?". IBM. https://www.ibm.com/topics/artificial-intelligence (acceso jul. 10, 2024).

[52] Google Cloud. "What is Machine Learning?". Google Cloud. https://cloud.google.com/learn/what-is-machine-learning (acceso jul. 10, 2024).

[53] MathWorks. "Introducción a aprendizaje supervisado". MathWorks. https://la.mathworks.com/discovery/supervised-learning.html (acceso jul. 10, 2024).

[54] IBM. "¿Qué es el aprendizaje no supervisado?". IBM. https://www.ibm.com/es-es/topics/unsupervised-learning (acceso jul. 10, 2024).

[55] Amazon Web Services. "Amazon Personalize". Amazon Web Services. https://aws.amazon.com/es/pm/personalize/ (acceso jul. 10, 2024).

[56] Google Cloud. "BigQuery ML". Google Cloud. https://cloud.google.com/bigquery (acceso jul. 10, 2024).

[57] Oracle. "¿Qué es Firebase?". Oracle Developer. https://developer.oracle.com/es/learn/technical-articles/what-is-firebase (acceso dic. 3, 2023).

[58] Google. "Acelera y escala el desarrollo de apps sin administrar infraestructura". Firebase. https://firebase.google.com/products-build?hl=es-419 (acceso dic. 3, 2023).

[59] J. Serrano, Perfil de Instagram de Javier Serrano, productor y guitarrista. Instagram. https://www.instagram.com/javierguitars/ (acceso jul. 10, 2024).

[60] tuToque.co. "Reporte de Gestión BIC - 2022". tuToque.co. https://tutoque.co/home/bic (acceso dic. 3, 2023).

[61] WWWhat's new. "ReverbNation – Plataforma de Marketing Musical Social". WWWhat's new. https://wwwhatsnew.com/2010/07/31/reverbnation-plataforma-de-marketing-musical-social/ (acceso dic. 8, 2023).

[62] ReverbNation. "ReverbNation: Artists First". ReverbNation. https://www.reverbnation.com/ (acceso dic. 8, 2023).

[63] GigSalad. "About Us". GigSalad. https://www.gigsalad.com/about (acceso dic. 10, 2023).

[64] GigSalad. "How It Works". GigSalad. https://www.gigsalad.com/how-it-works (acceso dic. 10, 2023).

[65] Google Cloud. "Productos de Google Cloud". Google Cloud. https://cloud.google.com/products (acceso jul. 10, 2024).

[66] Amazon Web Services. "Productos de la nube de AWS". Amazon Web Services. https://aws.amazon.com/es/products/ (acceso jul. 10, 2024).

[67] Microsoft Azure. "Azure products". Microsoft Azure. https://azure.microsoft.com/en-us/products/ (acceso jul. 10, 2024).

[68] Amazon Web Services. "Calculadora de precios de AWS". Amazon Web Services. https://calculator.aws/ (acceso jul. 10, 2024).

[69] Google Cloud. "Welcome to Google Cloud's pricing calculator". Google Cloud. https://cloud.google.com/products/calculator (acceso jul. 10, 2024).

[70] Microsoft Azure. "Calculadora de precios". Microsoft Azure. https://azure.microsoft.com/es-es/pricing/calculator/ (acceso jul. 10, 2024).

[71] Google Cloud. "Cloud CDN y Media CDN". Google Cloud. https://cloud.google.com/cdn (acceso jul. 10, 2024).

[72] Google Cloud. "Mapping custom domains". Google Cloud. https://cloud.google.com/run/docs/mapping-custom-domains (acceso ene. 10, 2025).

[73] Stripe. "¿Qué es Stripe?". Stripe. https://support.stripe.com/questions/what-is-stripe (acceso ene. 10, 2025).

[74] Stripe. "A guide to PCI compliance". Stripe. https://stripe.com/guides/pci-compliance (acceso ene. 10, 2025).

[75] Amazon Web Services. "Preparación de los datos de interacción de elemento para el entrenamiento". Amazon Web Services. https://docs.aws.amazon.com/es_es/personalize/latest/dg/interactions-datasets.html (acceso ene. 10, 2025).

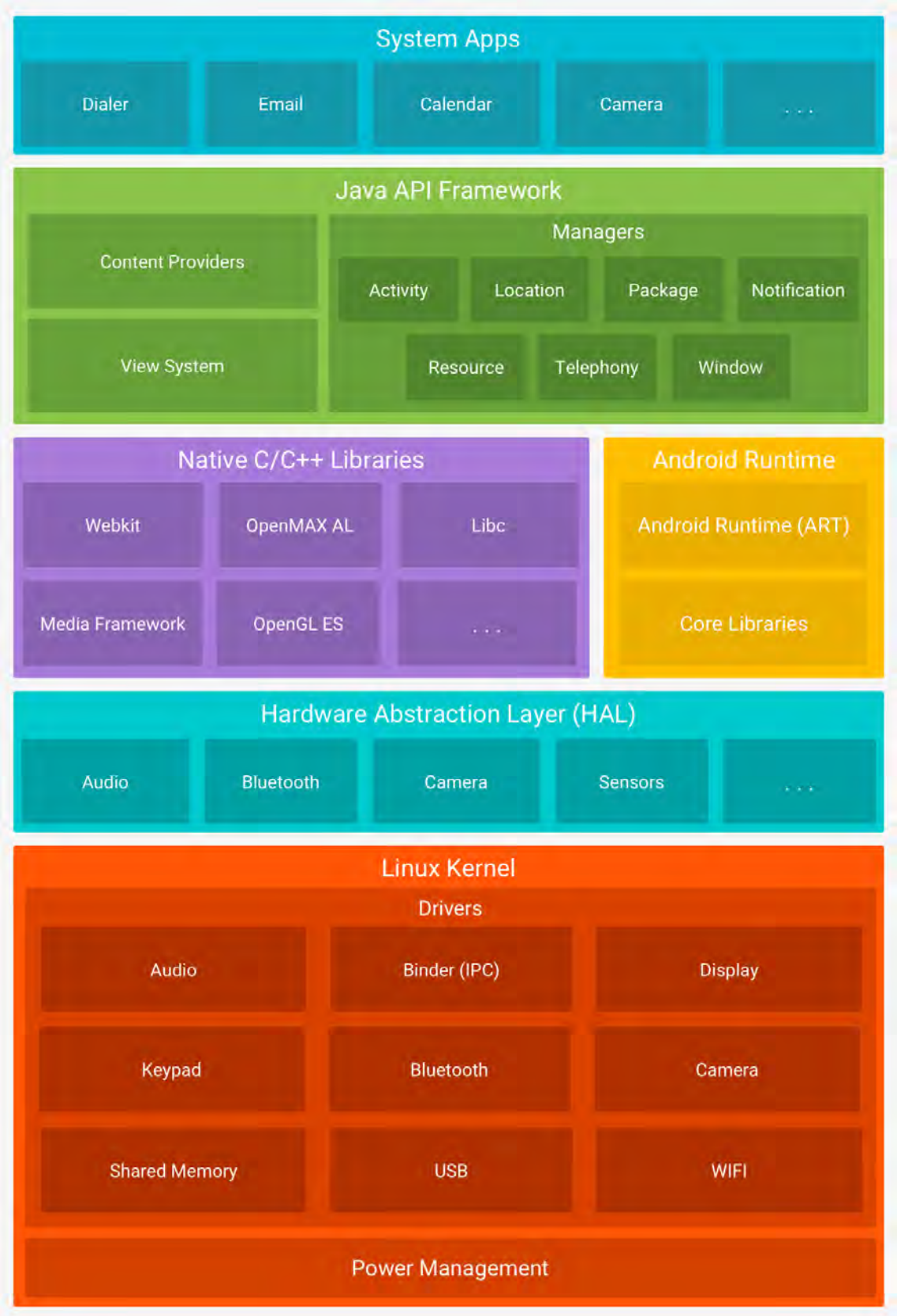
[76] Amazon Web Services. "Amazon Personalize pricing". Amazon Web Services. https://aws.amazon.com/personalize/pricing/ (acceso jun. 15, 2025).

[77] KPMG. "Cost of Capital Study". KPMG. https://kpmg.com/de/en/home/insights/overview/cost-of-capital.study.html (acceso ene. 10, 2025).

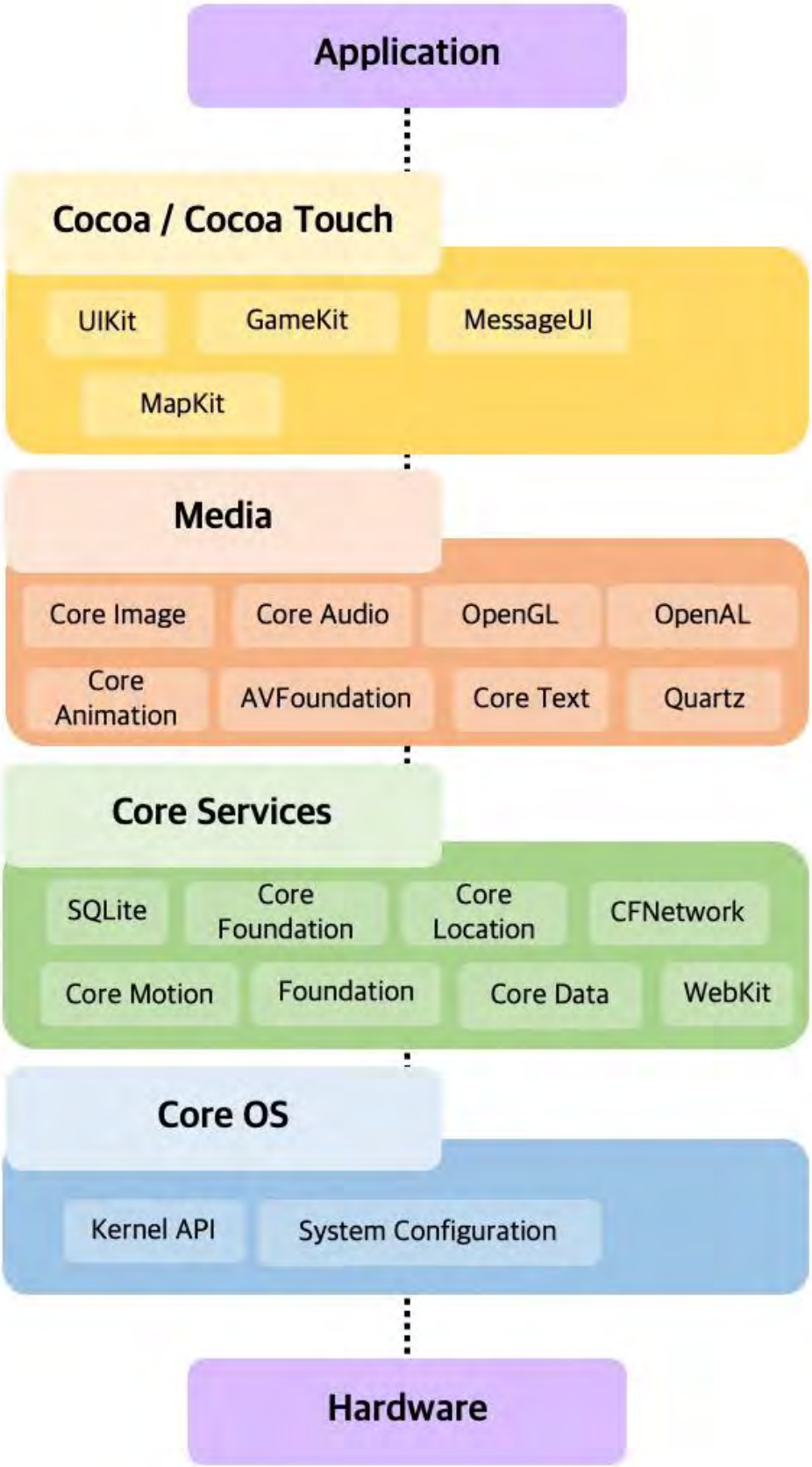
[78] La República. "¿Cuánto gana en promedio un músico profesional en Perú?". La República. https://larepublica.pe/datos-lr/respuestas/2022/10/09/cuanto-gana-un-musico-en-el-peru-en-promedio-conservatorio-nacional-de-musica-arte-sueldos-atmp (acceso ene. 10, 2025).

ANEXOS

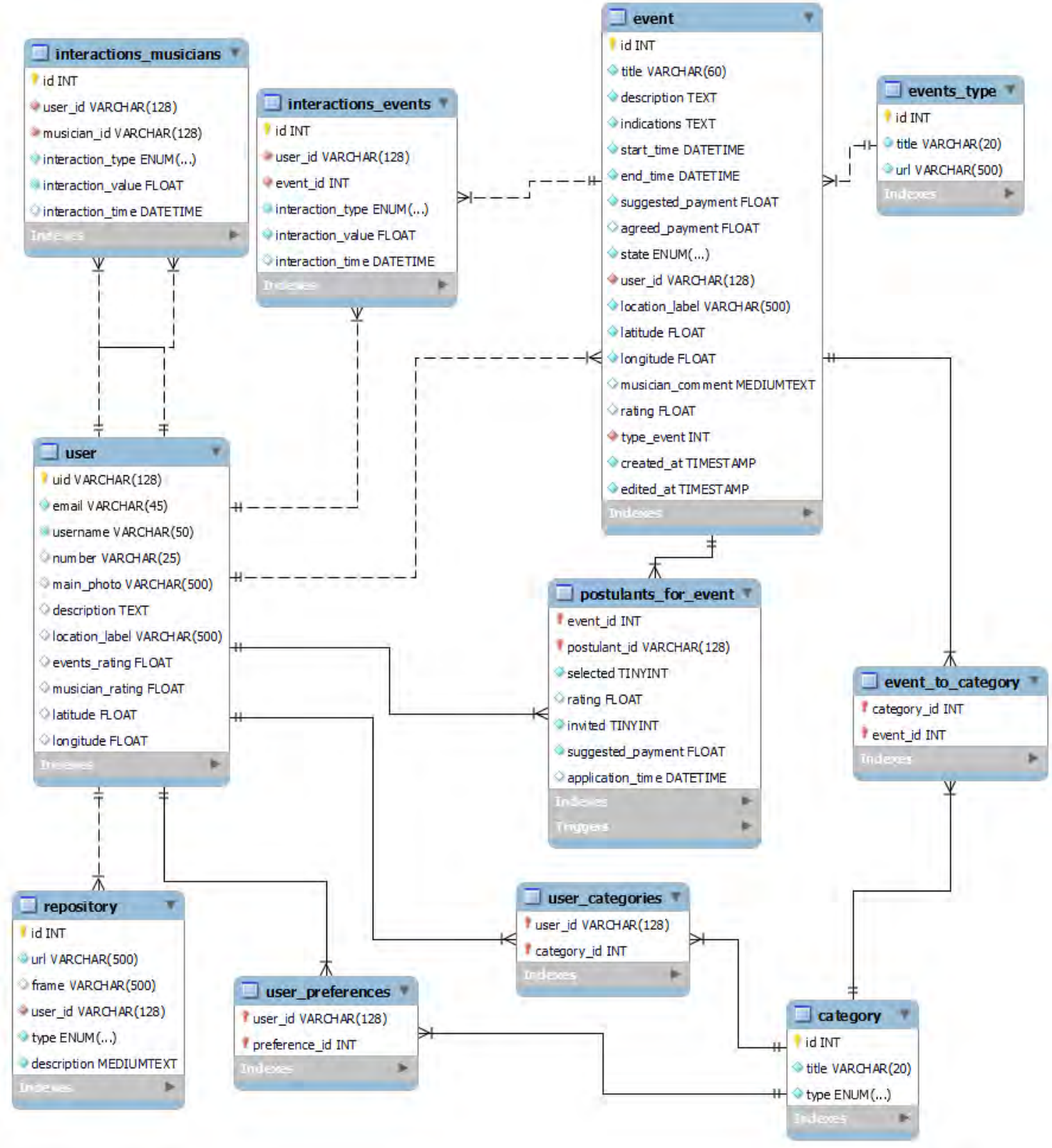
Anexo 1. Arquitectura de Android OS



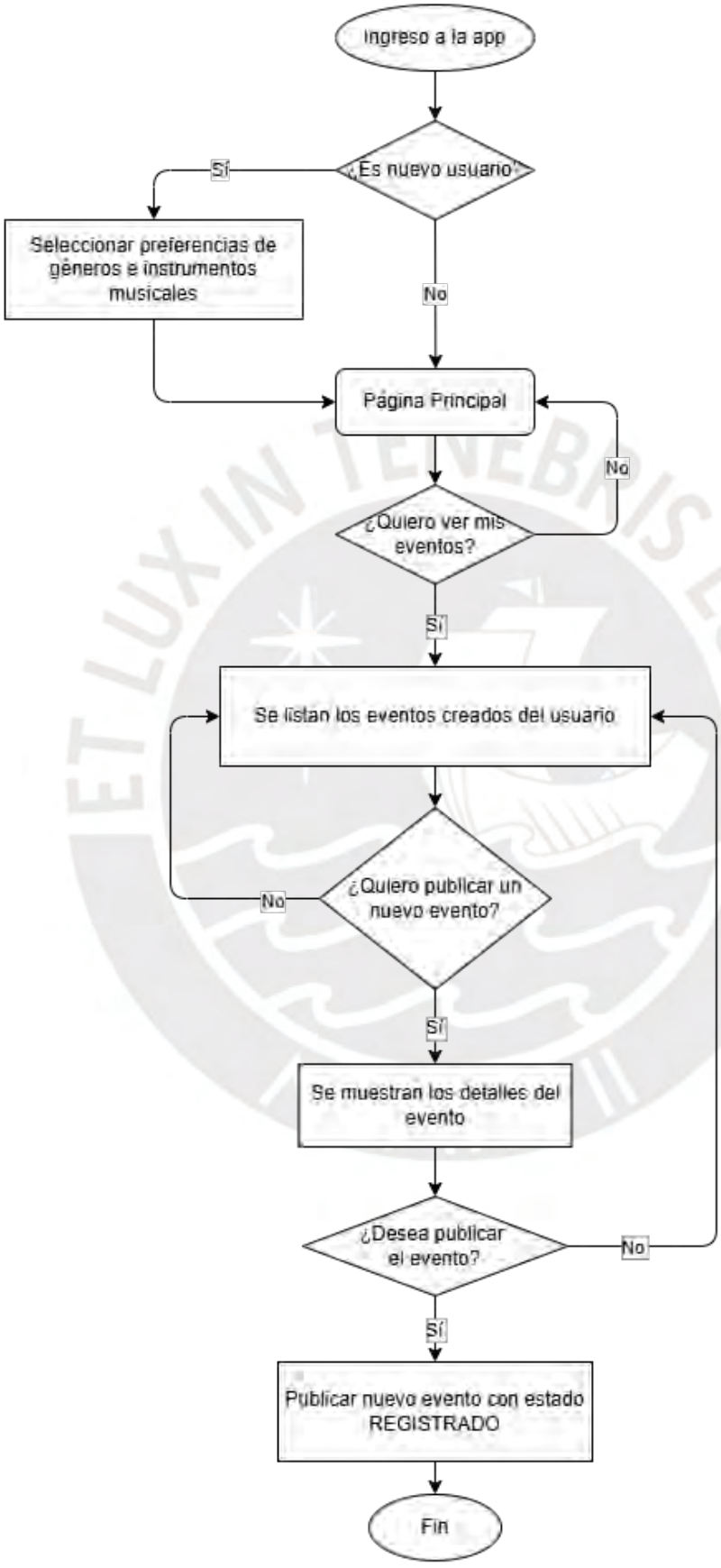
Anexo 2. Arquitectura de iOS



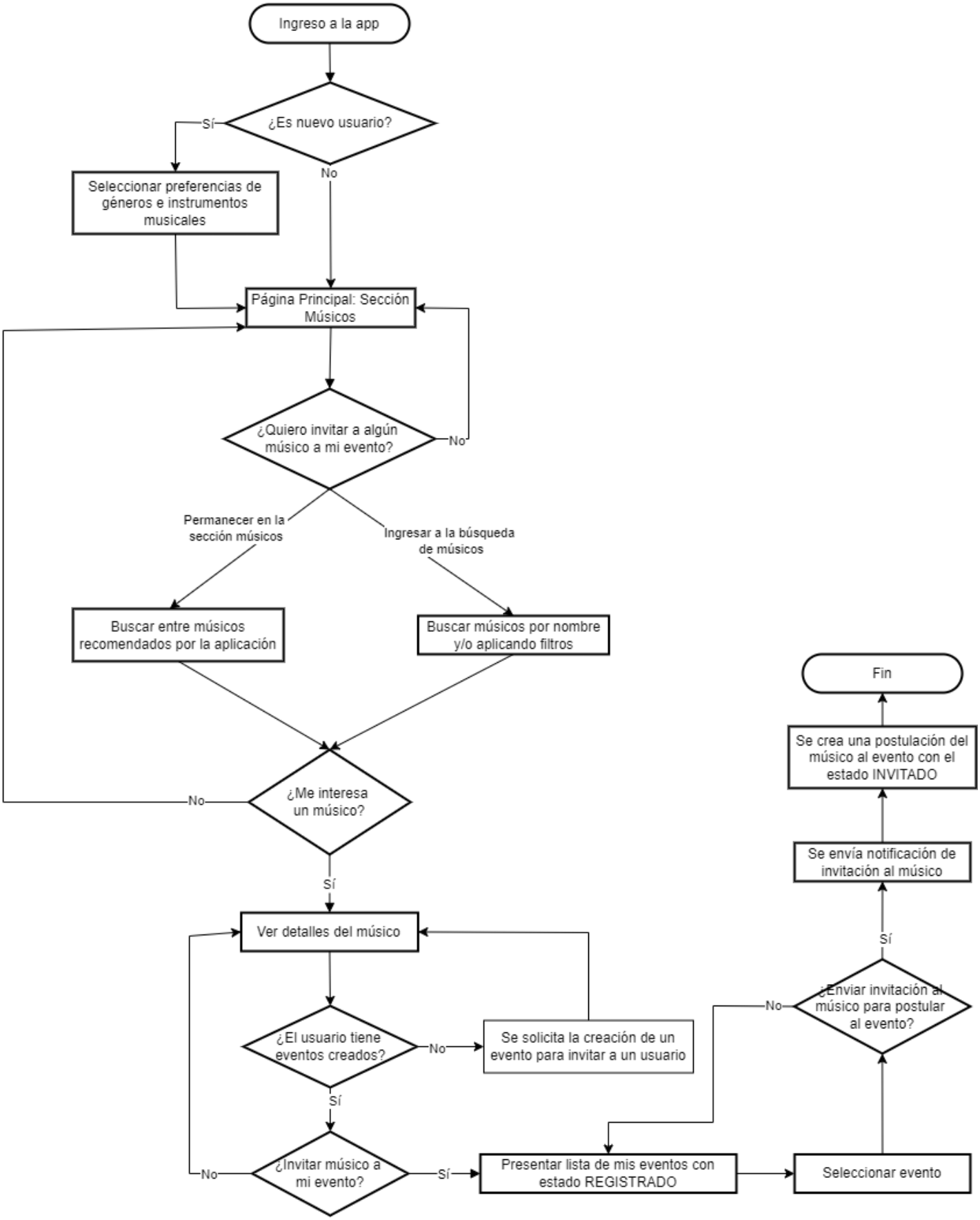
Anexo 3. Diagrama Entidad-Relación de la Base de Datos



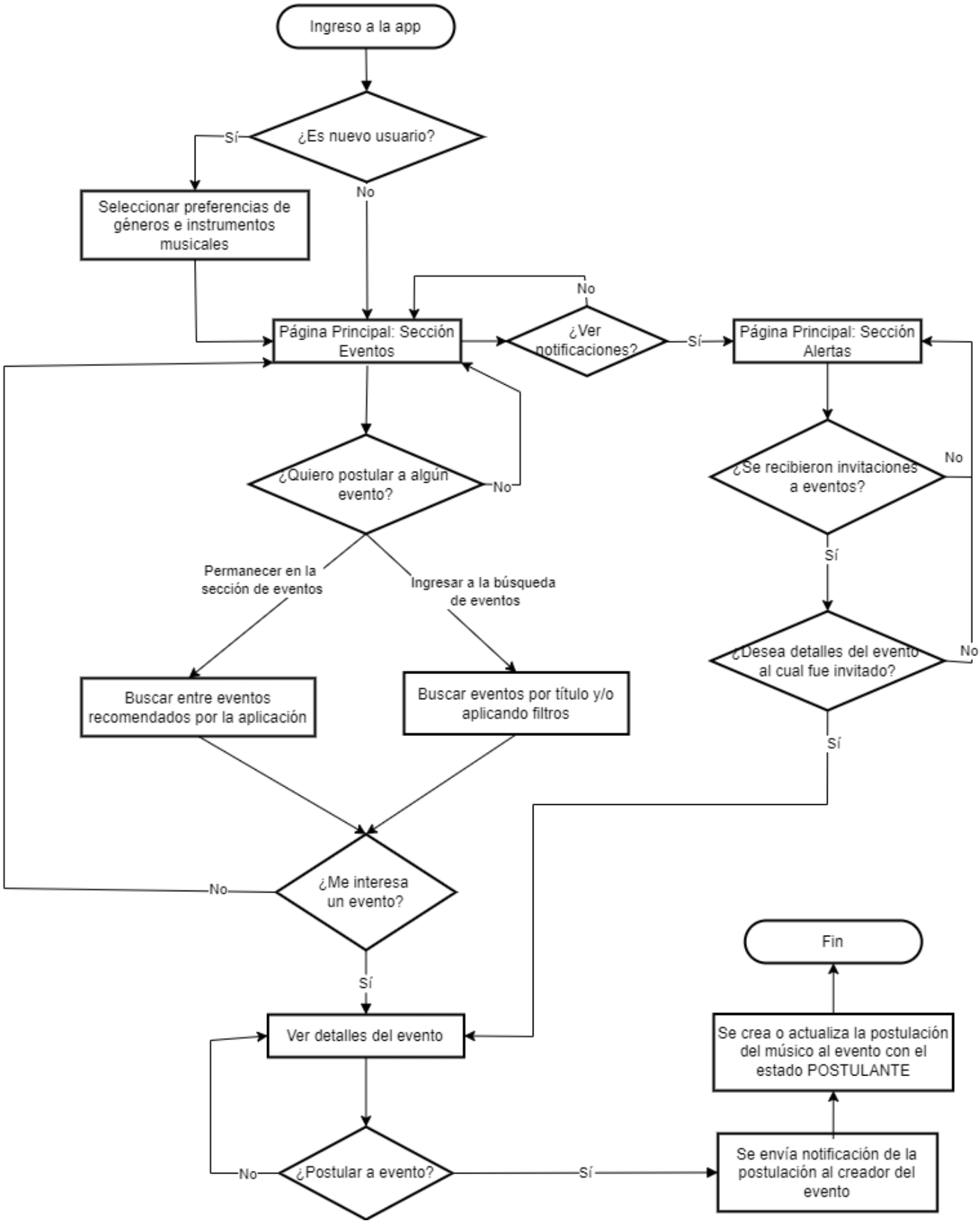
Anexo 4. Diagrama de Flujo: Publicación de evento



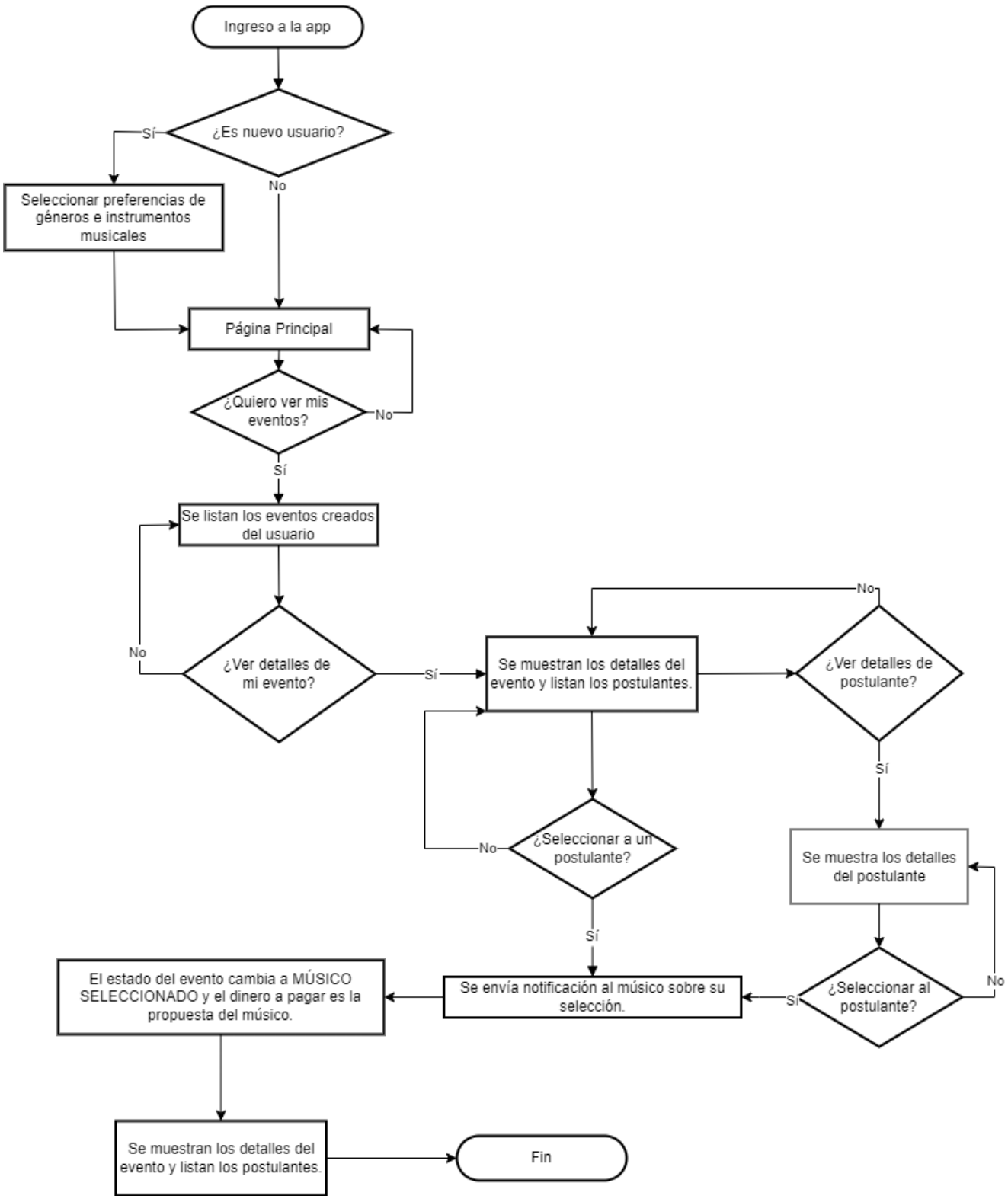
Anexo 5. Diagrama de Flujo: Invitar músico a un evento



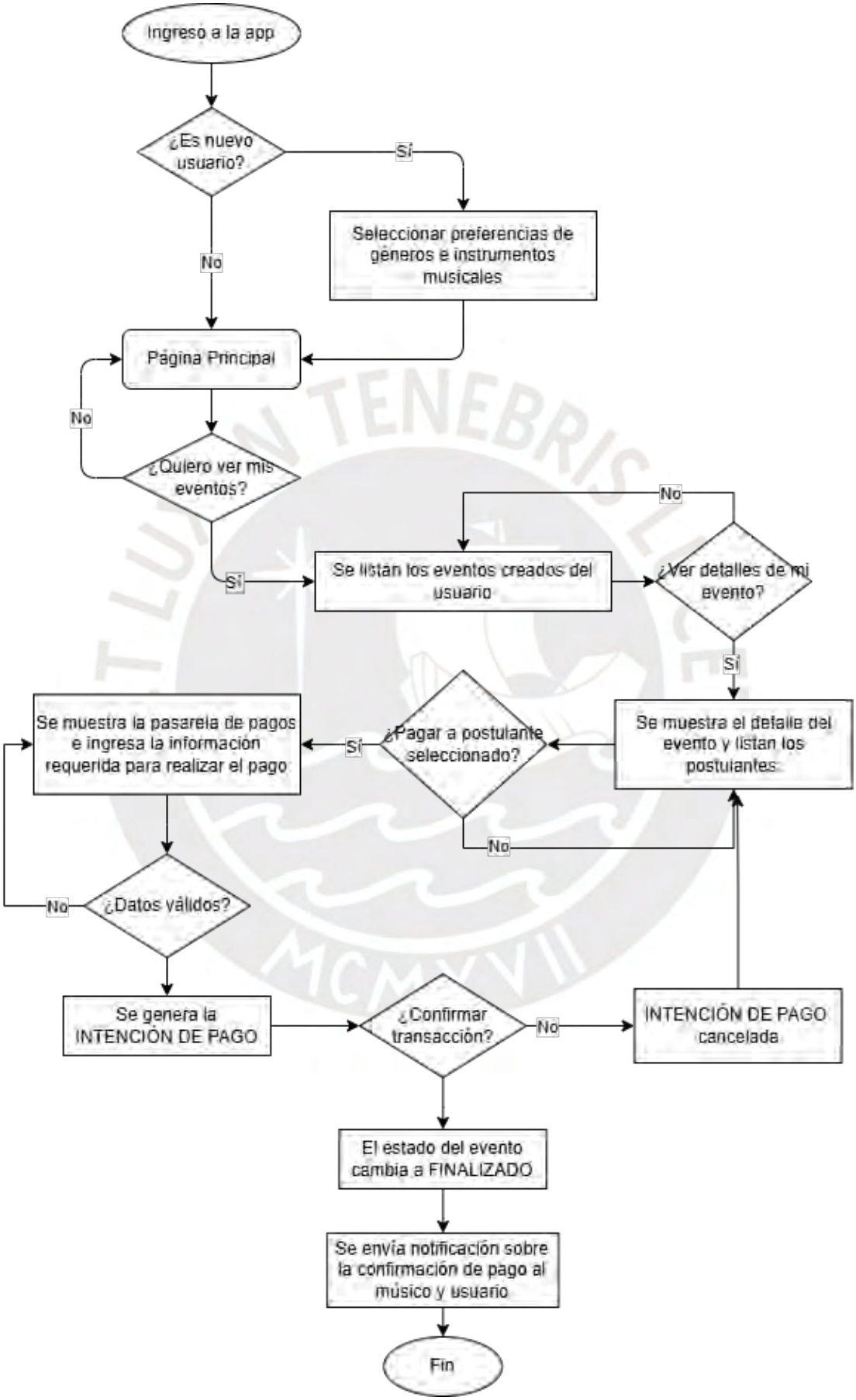
Anexo 6. Diagrama de Flujo: Postular a un evento



Anexo 7. Diagrama de Flujo: Selección de músico a evento

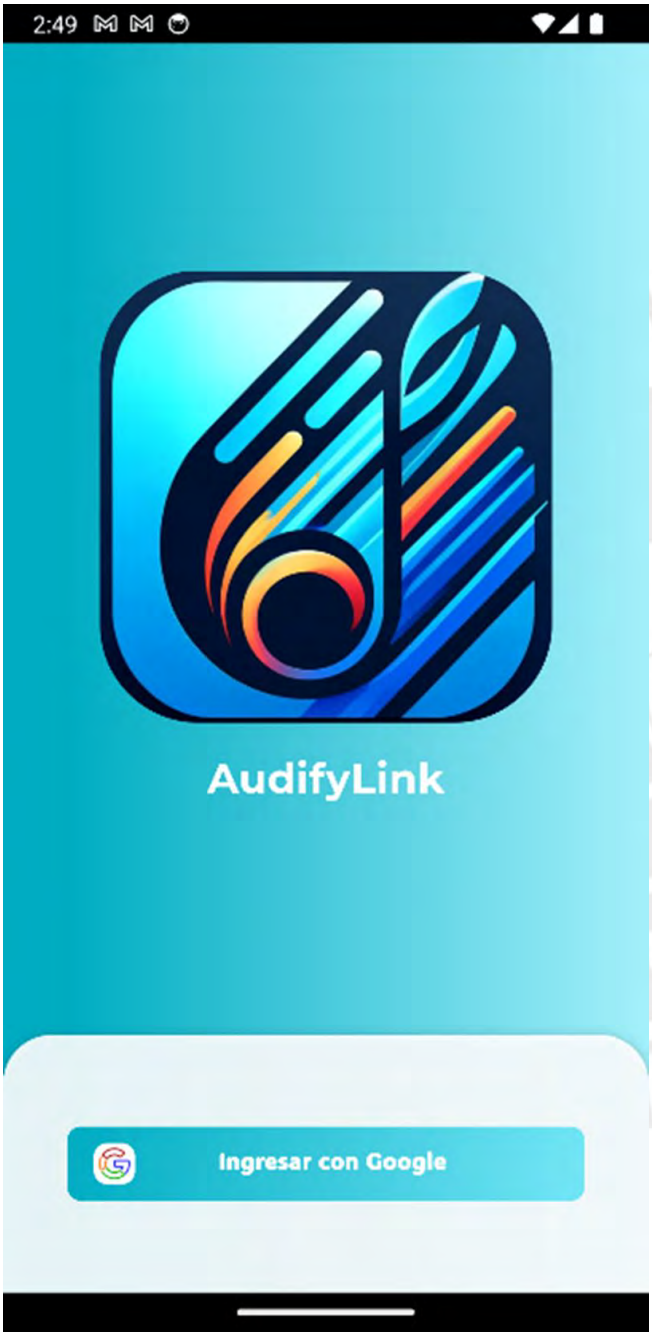


Anexo 8. Diagrama de Flujo: Pagar a músico

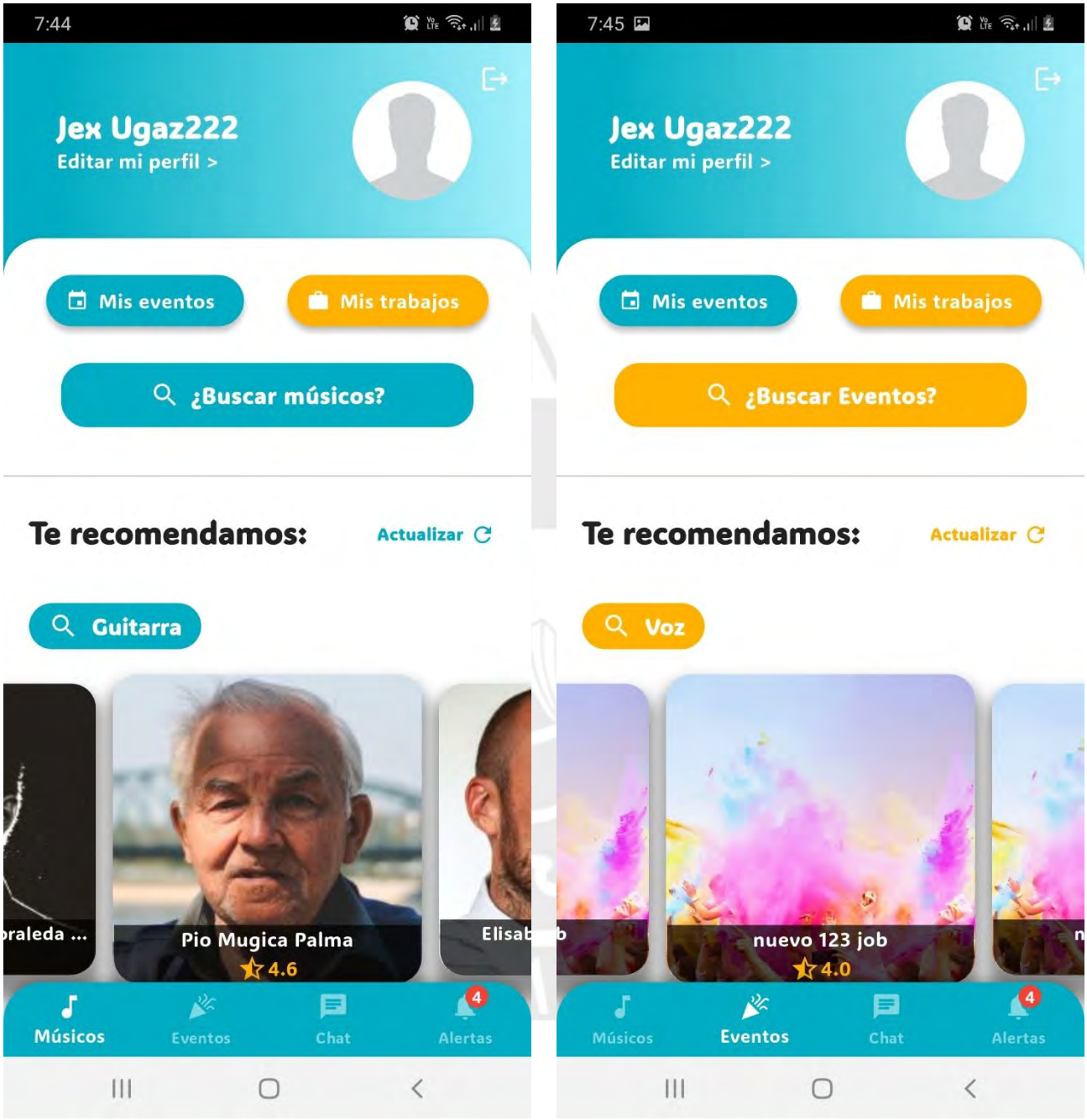


Anexo 9. Interfaces del aplicativo

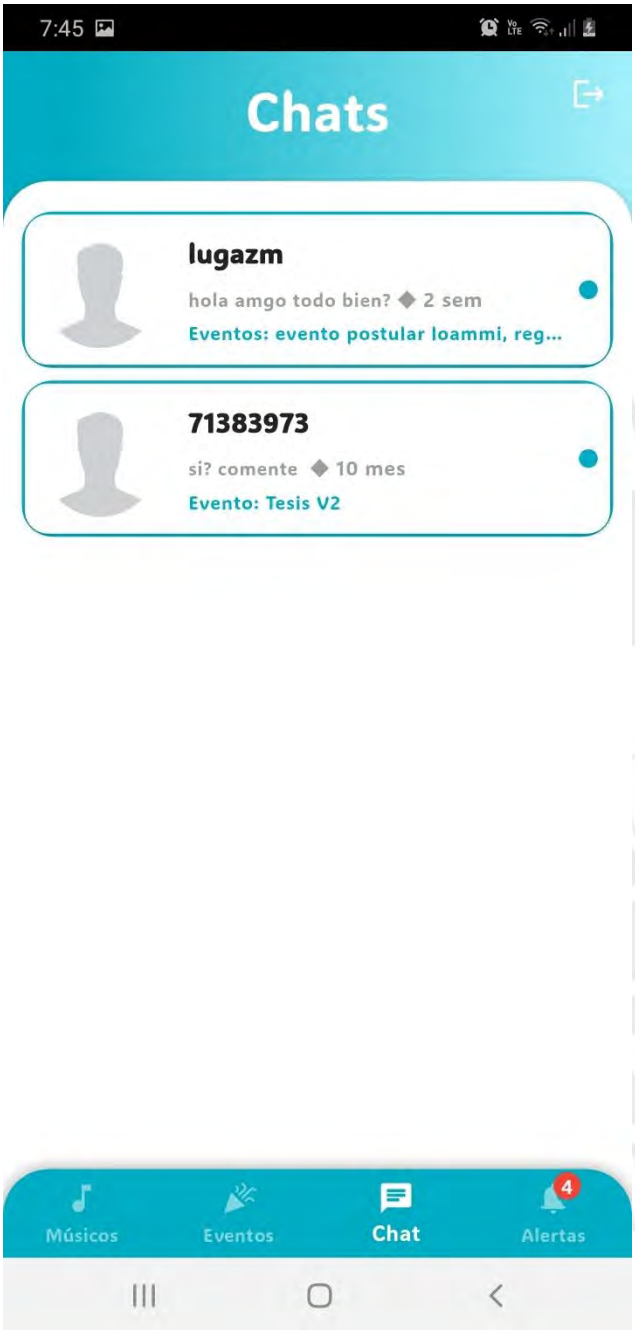
9.1 Interfaz de autenticación y preferencias iniciales



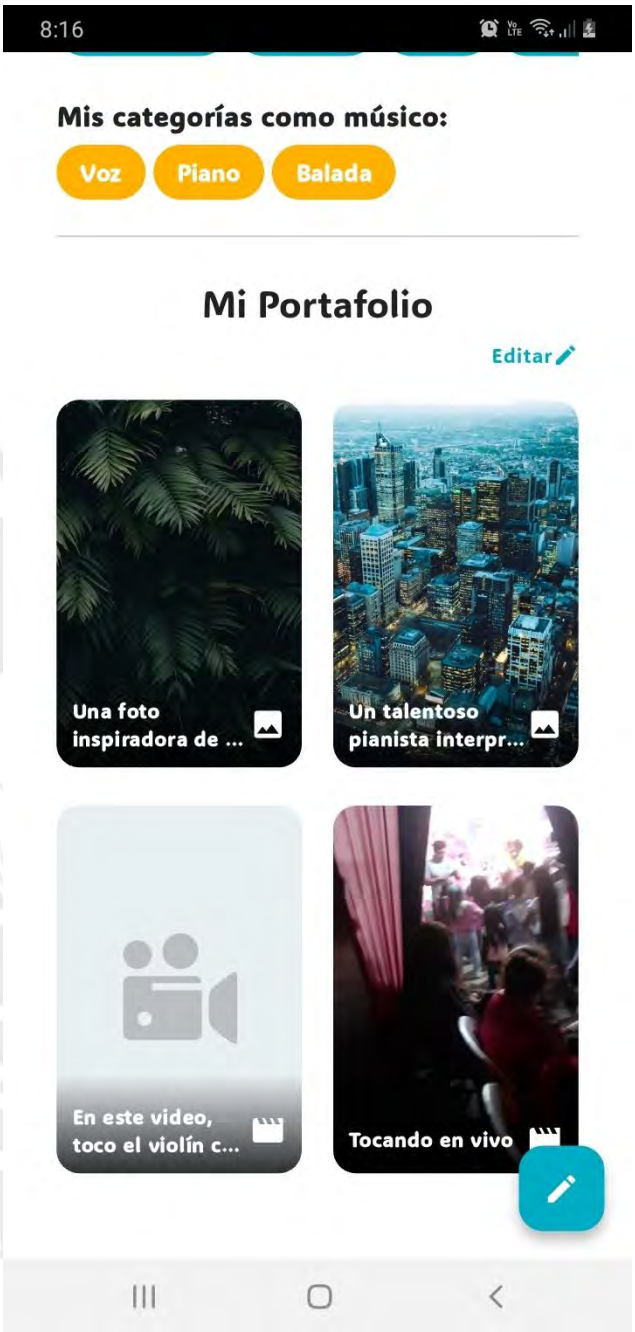
9.2 Vista principal de recomendaciones



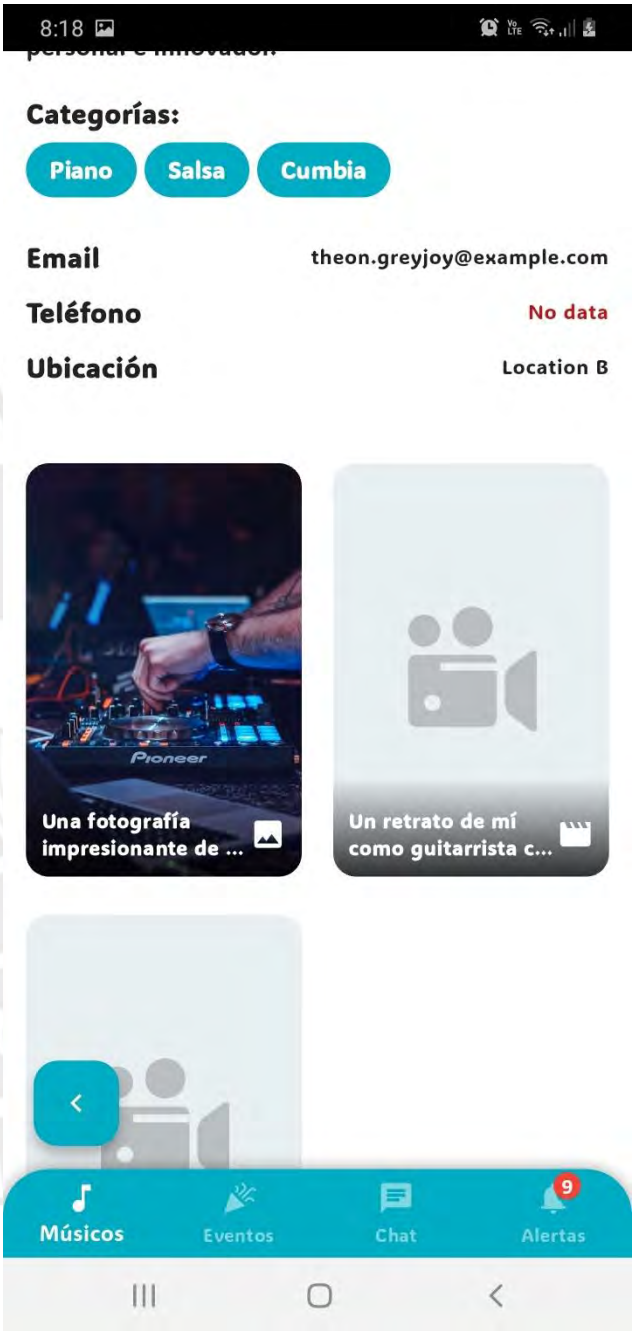
9.3 Vista principal de comunicación y notificaciones



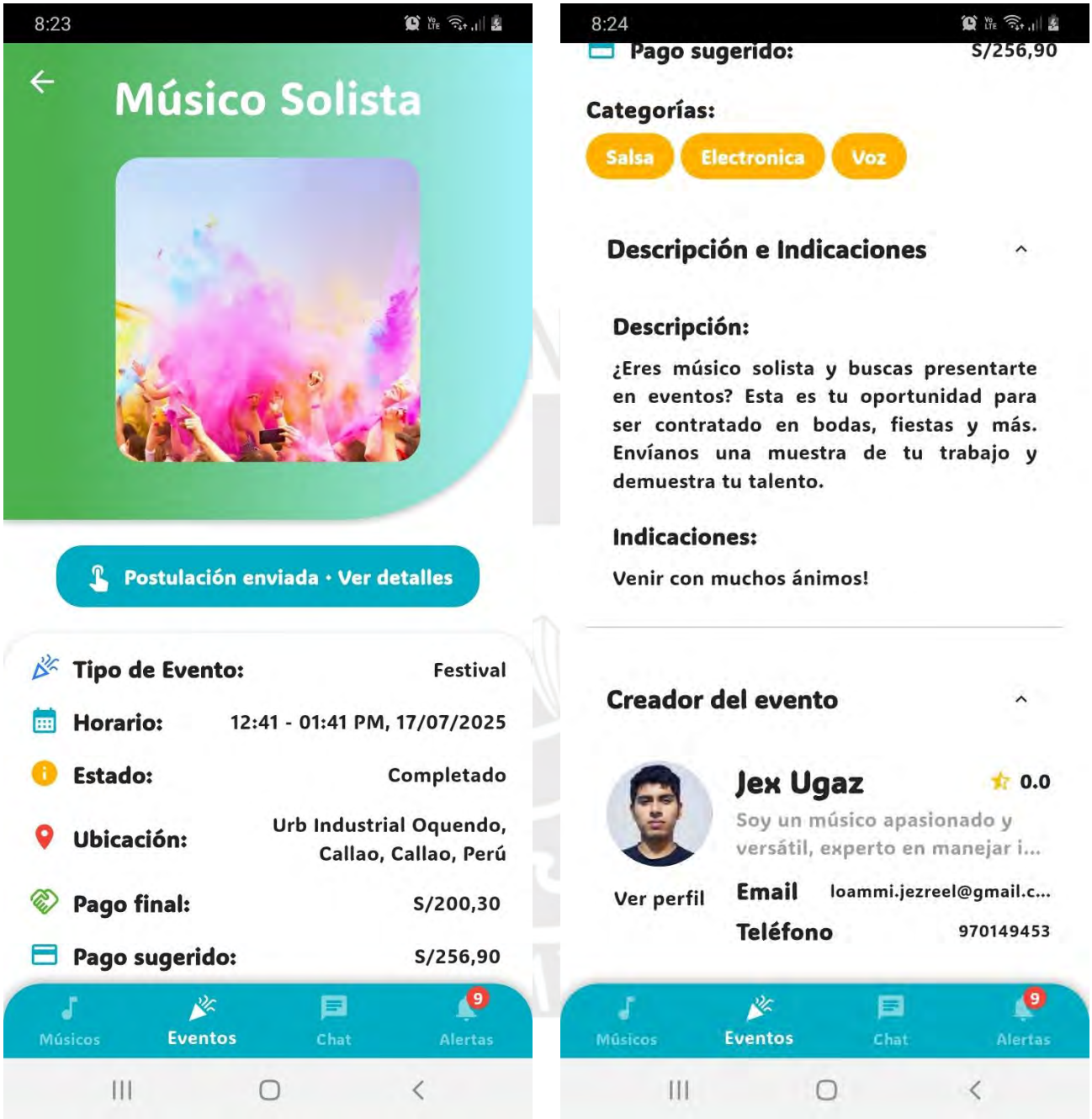
9.4 Perfil del músico y repositorio multimedia



9.5 Interfaz de detalles del músico

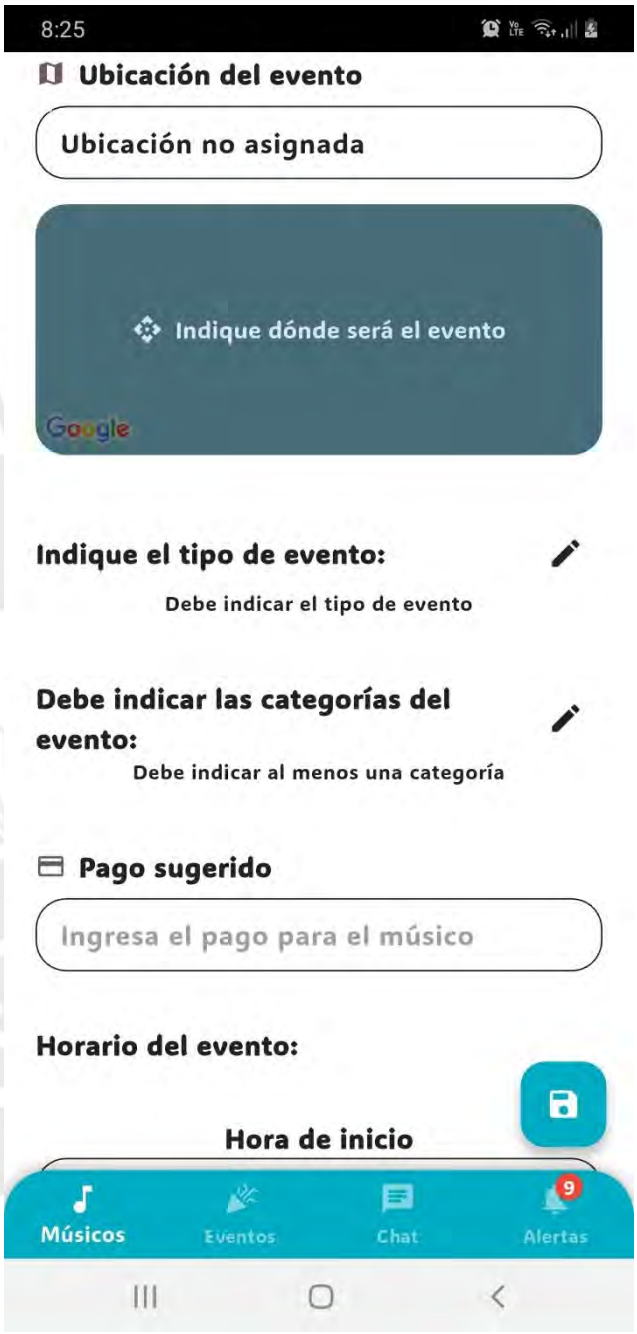


9.6 Interfaz de detalles del evento



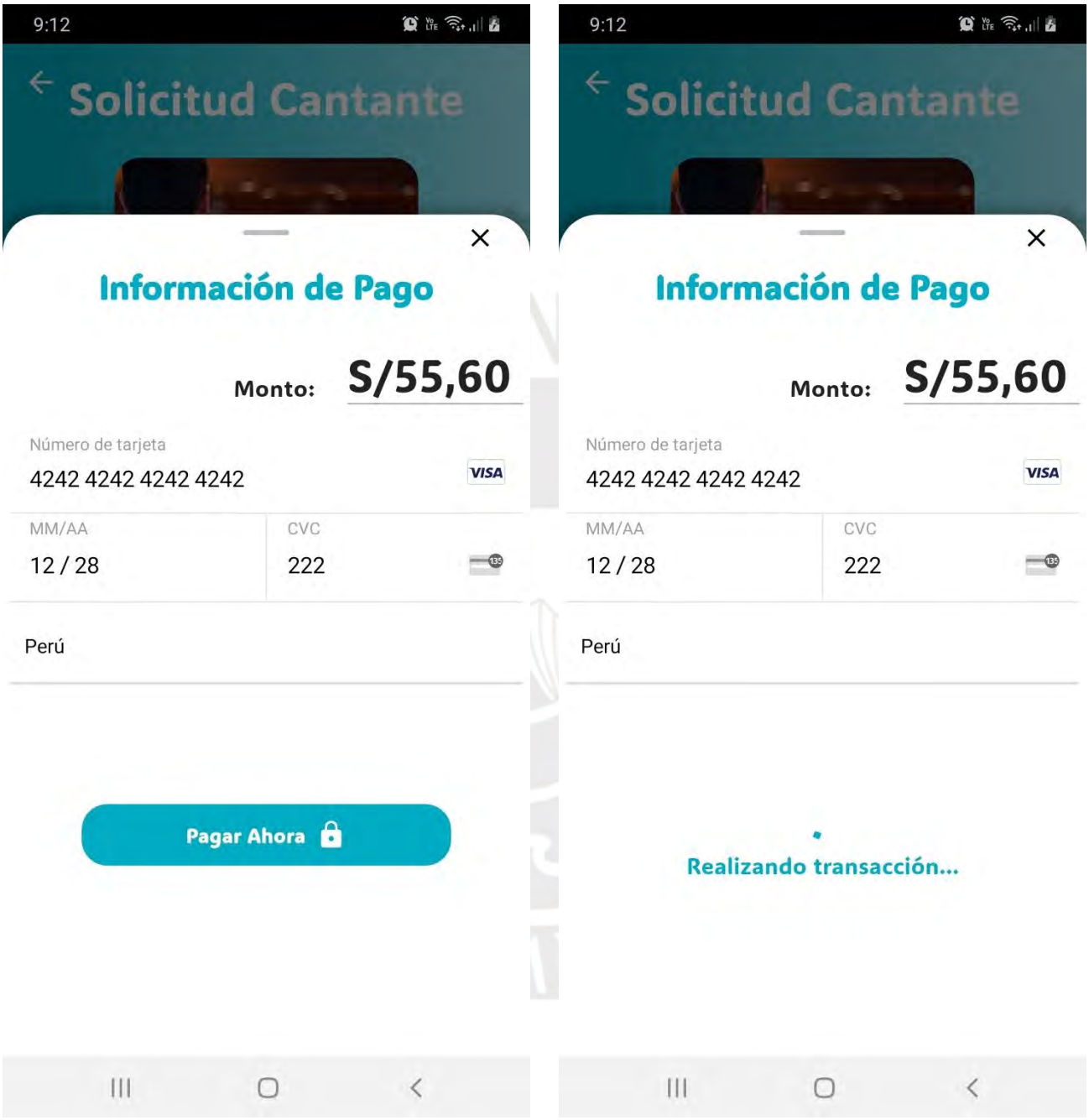
9.7 Interfaz de gestión de trabajos y eventos

9.8 Interfaz de creación y edición de eventos



9.9 Interfaz de detalles de mi evento y gestión de postulantes

9.10 Interfaz de pasarela de pagos



Anexo 10. Diferencias entre IaaS, PaaS y SaaS