ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

DESARROLLO DE SISTEMA WEB Y APLICACIÓN MÓVIL INFORMATIVA SOBRE MÚSICA NACIONAL ECUATORIANA

DESARROLLO DE UN BACKEND

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO SUPERIOR EN DESARROLLO DE SOFTWARE

EDWIN IVÁN FRAGA TANA

DIRECTOR: LOARTE CAJAMARCA BYRON GUSTAVO

Quito, septiembre 2022

CERTIFICACIONES

Yo, Edwin Iván Fraga Tana declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Iván Fraga

edwin.fraga@epn.edu.ec

fragaivan@outlook.com

Certifico que el presente trabajo de integración curricular fue desarrollado por Edwin Iván Fraga Tana, bajo mi supervisión.

Ing. Byron Loarte, MSc.
DIRECTOR

byron.loarteb@epn.edu.ec

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Edwin Iván Fraga Tana

DEDICATORIA

Este trabajo está dedicado a Dios, que me ha ayudado con el entendimiento para poder comprender los conocimientos que me han ofrecido los profesores y segundo a mi mamá, que siempre me apoyó y me animo por el logro más mínimo que obtenía.

Edwin Iván Fraga Tana

AGRADECIMIENTO

Agradezco a Dios, por el entendimiento que siempre me brinda a mi mamá, que sin su guía y apoyo probablemente ni siquiera hubiese podido ingresar a la universidad. Ella me mostró que la constancia y disciplina son fundamentales para el logro de cualquier meta.

Agradezco a los profesores y en especial al Ing. Byron Loarte, quien se ha esforzado con mucha paciencia para poder compartir un poco de su conocimiento conmigo y finalmente agradezco a los tutoriales y recursos en la web ya que la mayor parte de este trabajo está basado en su implementación.

Edwin Iván Fraga Tana

ÍNDICE DE CONTENIDO

RESUMEN

Actualmente, la emisión de contenido musical se encuentra en su mayoría difundido por plataformas y aplicaciones *web*, las cuales se enfocan en recomendar contenido de artistas populares, omitiendo a nuevos artistas del medio, esto resulta para los nuevos artistas un impedimento para poder darse a conocer. Un claro ejemplo de esto es *Spotify* en donde al ser una plataforma internacional se centra en ofrecerle a sus usuarios principalmente canciones del momento. Es por ello, que las plataformas de música nacional se han convertido en un medio apropiado para dar a conocer a artistas ecuatorianos.

Para poder difundir el contenido musical de artistas nacionales, en el presente trabajo se ha desarrollado un *backend* sobre música nacional ecuatoriana, el cual da la posibilidad a artistas ecuatorianos de subir géneros, álbumes y canciones al sistema “TakiTri”, con el objetivo de que la información pueda ser consumida por cualquier aplicación del lado del cliente o móvil y por otra parte, que la ciudadanía conozca el contenido musical de estos artistas de manera interactiva gracias al uso de la tecnología.

La estructura de este documento es la siguiente: En la primera sección se describen los antecedentes, objetivos, alcance y marco metodológico. La segunda sección detalla cómo está puesto en práctica la metodología ágil *Scrum*, modelo de datos, diseño de arquitectura y herramientas en el desarrollo del presente *backend*. La tercera sección, detalla cada actividad que se ha realizado en cada iteración, resultados, conclusiones y recomendaciones extraídas en el desarrollo de este trabajo de integración curricular.

PALABRAS CLAVE: *backend, Angular, Firebase, Scrum*.

ABSTRACT

At present, the dissemination of musical content is mainly carried out through web platforms and applications, focusing on recommending the content of popular artists, omitting new artists from the medium, which is an obstacle for new artists to become famous. A clear example is Spotify, an international platform that focuses on providing users with the songs of the moment. Therefore, national music platforms have become an appropriate means to make national artists known.

In order to spread the musical content of national artists, in the present work an Ecuadorian national music *backend* is developed, which offers Ecuadorian artists the possibility of uploading genres, albums and songs to the "TakiTri" system. The information can be used by any application on the client or mobile side, on the other hand, thanks to the use of technology, the public interactively knows the musical content of these artists.

The structure of this document is as follows: The first section describes the background, objectives, scope and methodological framework. The second part details how the agile Scrum methodology, data model, architectural design, and tools are put into practice in back-end development. The third section, details each activity carried out in each iteration, results, conclusions and recommendations extracted in the development of this integration work.

KEYWORDS: *backend, Angular, Firebase, Scrum*.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

Hoy por hoy, las aplicaciones en relación con la música se enfocan principalmente en presentar contenido musical de artistas del momento a sus usuarios, pero debido a la popularidad de estos artistas, ocasiona que el contenido de artistas nacionales sea opacado y por tanto ignorado por los usuarios de dichas aplicaciones [1].

Según Guzmán [2] la internacionalización de grupos o artistas musicales es cada vez menor debido a los altos recursos que implica realizarla. Las entidades encargadas del apoyo financiero a estos grupos son cada vez más selectivas y debido a que la venta de *CDs* ya no es viable, estas han optado por basarse en plataformas digitales.

Es importante tomar en cuenta que, en Ecuador a causa de la falta de patrocinio de empresas dedicadas a la industria musical, el músico ecuatoriano se ve forzado a producir, publicar y gestionar su contenido musical, y aun cuando al principio no perciban ningún ingreso, lo hacen con el fin de difundir su trabajo [3].

Las plataformas que utilizan los pequeños artistas para difundir su trabajo que, si bien no lo muestran de forma regular, tales como *YouTube Music, Spotify, Amazon Music,* etc. Cumplen con su cometido de ofrecer contenido musical de manera eficaz y eficiente hacia sus usuarios. Eso es debido a que estas plataformas de música usan otras plataformas de almacenamiento en la nube y la presentación de la información en tiempo real para la carga y descarga de contenido multimedia [4].

Actualmente una de las plataformas digitales más populares para el almacenamiento en la nube es *Firebase*, cuyo objetivo principal no solo es al almacenamiento en la nube, sino el apoyo para desarrollar aplicaciones *web* y móviles. Entre las principales ventajas de *Firebase* se encuentra el soporte en varios sistemas operativos y lenguajes de programación tales como *Android*, *iOS* y la *Web*. Su uso es gratuito al principio, pero si se requiere servicios en alta escala se puede solicitar planes según las necesidades del desarrollo [5].

Por lo anteriormente citado en el presente proyecto de Integración Curricular se ha desarrollado un *backend* utilizando la plataforma *Firebase* el cual permite acceder a una serie de servicios para que sean consumidos por parte de aplicaciones de lado del cliente o móviles. Logrando de esta manera que los artistas ecuatorianos puedan subir su contenido musical por medio de un *frontend* y que la ciudadanía pueda escuchar sus canciones por medio de una aplicación móvil. El enfoque principal de este proyecto es

proporcionar un medio digital que ofrezca géneros, álbumes y canciones propias de artistas 100% ecuatorianos que comúnmente no aparecen en otras aplicaciones o plataformas musicales.

1.1 Objetivo general

Desarrollar un *backend* informativo sobre música nacional ecuatoriana.

1.2 Objetivos específicos

1. Levantar requisitos correspondientes al *backend*.
2. Diseñar la arquitectura de datos para el *backend* en base a los requerimientos que se han obtenido.
3. Configurar y otorgar acceso a la Base de Datos.
4. Codificar el acceso a las diferentes funciones y servicios de *Firebase*.
5. Verificar el funcionamiento del *backend* y despliegue a producción.

1.3 Alcance

Hoy en día las aplicaciones móviles y *web* son programas que funcionan a través de Internet, es decir, trabajan, procesan y almacenan información en tiempo real. El concepto de sistemas *web* se relaciona con la gestión de contenido en la nube y el de las aplicaciones móviles con el consumo de estos [6]. Es por esta razón, que se ha optado por la implementación de la plataforma *Firebase* la cual otorga los siguientes servicios:

- *Firebase Authentication*, el cual permite registrar, iniciar sesión y cambiar la contraseña de una cuenta de usuario, consiguiendo así una mejor seguridad y protección de los datos tanto para el usuario como para la aplicación misma [7].
- *Firebase Firestore*, el cual permite establecer una Base de Datos en tiempo real mediante la implementación de colecciones y documentos [7].
- *Cloud Storage*, el cual permite el almacenamiento de archivos generados por los usuarios a través de las aplicaciones que se han desarrollado [7].

por consiguiente, estos servicios de *Firebase* junto con sus métodos respectivos permiten tener un *backend* totalmente disponible para que los artistas ecuatorianos puedan subir su contenido por medio de una aplicación del lado del cliente que se ha desarrollado con Angular y que el consumo de dicha información se lo realice libremente por medio de una aplicación móvil que se ha desarrollado con *React Native*. Además, cada artista puede

gestionar su información de manera detallada, eficiente, personalizada, en tiempo real y con una serie de roles y perfiles garantizando de esta manera una capa de seguridad robusta. El desarrollo de este *backend* permite fortalecer los géneros musicales que son únicos en el Ecuador y que necesitan ser difundidos día a día gracias al uso de la tecnología y aplicaciones de software [8].

A continuación, se presenta a cada uno de los usuarios y las acciones que pueden realizar una vez que el *backend* se integre con un *frontend*.

Usuario invitado

- Implementación de un método para registrarse a través de un formulario.

Usuario administrador

- Implementación de varios métodos para iniciar sesión, cerrar sesión y recuperar contraseña.
- Implementación de varios métodos para modificar el perfil de usuario.
- Implementación de varios métodos para gestionar las solicitudes de artistas.

Usuario artista

- Implementación de varios métodos para iniciar sesión, cerrar sesión y recuperar contraseña.
- Implementación de varios métodos para modificar el perfil de usuario.
- Implementación de varios métodos para gestionar géneros musicales.
- Implementación de varios métodos para gestionar álbumes.
- Implementación de varios métodos para gestionar canciones.

A continuación, se presenta a cada uno de los usuarios y las acciones que pueden realizar una vez que el *backend* se integre con una aplicación móvil.

Usuario invitado

- Implementación de un método para registrarse a través de un formulario.

Usuario ciudadano

- Implementación de varios métodos para iniciar sesión, cerrar sesión y recuperar contraseña.
- Implementación de un método para modificar el perfil de usuario.
- Implementación de un método para visualizar géneros musicales.

- Implementación de un método para visualizar canciones.
- Implementación de varios métodos para gestionar favoritos.
- Implementación de varios métodos para gestionar *playlist*.
- Implementación de varios métodos para gestionar canciones al *playlist*.

1.4 Marco teórico

A lo largo del tiempo la industria musical ha cambiado de forma notable y las personas consumen este tipo de contenido según su época, de manera que ha ido variando desde casets, discos de vinilo, *CD* y ahora en formato digital. Es por ello por lo que en la actualidad el formato MP3 se ha convertido en el más popular en la industria musical [9].

Las plataformas o aplicaciones de música son aquellas que posibilitan la reproducción de audio de forma continua, sin la necesidad de descargar localmente los archivos; ofrecen muchos beneficios al usuario ya que puede disfrutar de un acceso fácil y en algunas ocasiones de manera gratuita [10].

Las plataformas digitales han tomado un lugar trascendental en la industria musical dejando así en segundo plano a los antiguos formatos tales como el vinilo y *CD*. Las plataformas ofrecen un modelo fácil, ordenado, simple y accesible. Es por esta razón, que para su uso únicamente se requiere de dispositivos inteligentes con acceso a Internet, con ello el usuario puede gozar de un contenido musical personalizado [9].

Para el desarrollo de este tipo de plataformas se requiere de dos capas: *frontend* y *backend*. La capa *frontend* es la encargada del diseño y desarrollo que se representa en el navegador, es decir la capa que interactúa directamente con los usuarios, mientras que la capa *backend* es la encargada de la lógica de la plataforma y del acceso a la base de datos, dicho en otras palabras, es la capa responsable del correcto funcionamiento de la plataforma y del acceso a los datos tanto de usuarios como de la plataforma misma [11].

El *backend*, por lo anteriormente citado y varias otras funcionalidades como: gestión de contenido multimedia, autenticación, acceso a una base de datos en tiempo real y despliegue a producción, entre otros se ha optado por el uso de la plataforma *Firebase*. *Firebase* es una plataforma en la nube enfocada en ser un apoyo para el desarrollo de aplicaciones *web* y móviles gracias a una serie de servicios [12]. Uno de ellos es *Cloud Firestore*, el cual es una base de datos de tipo *NoSQL* multifuncional que permite almacenar, consultar y sincronizar datos de manera eficiente en aplicaciones tanto *web* como móviles a nivel mundial [13]. Su funcionamiento radica en el uso de colecciones y

documentos para la estructuración de datos. A medida que se crean colecciones dentro de documentos se crean jerarquías, las cuales contienen datos relacionados y que pueden recuperarse por medio de consultas expresivas. La seguridad que implementa es sólida debido a su lenguaje de seguridad declarativo en el que se puede limitar el acceso de datos en relación con la de identidad del usuario. Por otra parte, otro de los servicios que *Firebase* proporciona es *Firebase Authentication* el cual ofrece una autenticación de usuarios de manera sencilla e intuitiva y con varios proveedores de terceros [14]. De igual manera, *Cloud Storage* está diseñado para almacenar, consultar y sincronizar contenido multimedia subido por usuarios. Los archivos pueden ser tales como: imágenes, canciones, videos, documentos, etc. La carga y descarga de estos archivos es robusta, debido a que en vista de que los usuarios no siempre están conectados el *SDK* de *Firebase Storage* permite detener o reanudar procesos de transferencia automáticamente dependiendo de la conectividad del usuario [15]. Para el lanzamiento de la aplicación, *Firebase Hosting* proporciona un servicio de *hosting* en el que se pueden implementar aplicaciones *web*, aplicaciones de destino a dispositivos móviles, aplicaciones progresivas, etc. Así mismo *Firebase Hosting* proporciona un certificado *SSL* para que cada sitio implementado establezca una conexión cifrada y por lo tanto sea segura para el usuario [16].

Angular es un *Framework* enfocado en el desarrollo de aplicaciones *Single Page Application (SPA) o Progresive Web App (PWA).* Angular presenta una base para el desarrollo de aplicaciones robustas escalables y optimizadas, aunque esta base es principalmente para el desarrollo de la parte *frontend* aborda técnicas para el desarrollo e integración con aplicaciones del lado de *backend* [17].

Las aplicaciones *SPA* son aquellas en las que se muestra todo el contenido en una sola página y sus pantallas son vistas que se van intercambiando de manera progresiva, para obtener esto *Angular* ofrece un sistema de *routing* que gracias al uso de *Lazy Loading* permiten obtener aplicaciones optimizadas y eficientes [18].

Angular usa *TypeScript* como lenguaje de programación el cual es una extensión de *JavaScript,* pero con la diferencia que posee adicionales características tales como el tipado de datos o decoradores. Otra de las razones por las que se usa *TypeScript* en *Angular* es su utilidad al mantener el código y la detección temprana de errores [17].

Visual Studio Code (VSC) es un editor de código fuente que puede ser ejecutado en Sistemas Operativos tales como *Linux, macOS* y *Windows*. Por otra parte, se caracteriza principalmente por ser gratuito, de código abierto y adaptable con múltiples lenguajes de programación tales como *Python, JavaScript, Java, TypeScript*, etc. [19] Permite

personalizar su interfaz de forma que se puede tener más de un código visible al mismo tiempo ayudando así al desarrollador en la optimización de tiempo y un gran soporte debido a su frecuente uso por parte de los desarrolladores [20].

2 METODOLOGÍA

Un estudio de casos es la investigación exhaustiva y detallada sobre un tema o problema en específico. Este tipo de investigación se diferencia de las demás debido a que en su realización e implementación se hace uso en su mayoría de metodologías cualitativas, por lo que sus resultados son comúnmente usados para poder describir, comparar y comprender los diversos aspectos de un problema o explorar las características, significados e implicaciones sobre un tema concreto a profundidad [21].

Por lo anteriormente mencionado, el presente proyecto de Integración Curricular mantiene un estudio de casos, puesto que parte de un problema sobre los conflictos que tienen los artistas ecuatorianos al difundir su contenido musical. Todo esto permite llevar a cabo el desarrollo de un *backend* para para que los artistas ecuatorianos puedan difundir su contenido y darse a conocer a más ecuatorianos por medio de una aplicación móvil y gracias al uso de la tecnología e Internet.

2.1 Metodología de Desarrollo

Se define como metodología de desarrollo de software al conjunto de técnicas y métodos estructurales y organizativas con el fin de desarrollar software. Estas metodologías se desarrollaron con la intención de optimizar el trabajo en equipo. Para ello es necesario tener en cuenta factores como: costes, complejidad, equipo de trabajo, lenguajes a usar, etc. Todo ello, permite organizar y agilizar el proceso de desarrollo con un menor nivel de dificultad [22].

Las metodologías ágiles en software son el conjunto de técnicas y prácticas administrativas desarrolladas con el objetivo de entregar un proyecto de software de manera pronta y progresiva. Estas generalmente están basadas en estructuras de ciclos breves que, al ser implementadas por el equipo de trabajo, posibilitan la entrega de avances periódicos del proyecto al cliente. Además, al aplicar metodologías ágiles, estas contribuyen a que el proyecto sea más flexible y adaptable a posibles cambios en el transcurso de su desarrollo [23]. Es por esta razón, que el desarrollo completo del *backend* se ha optado por la utilización de la metodología *Scrum*.

Scrum es una metodología ágil usada para aminorar el nivel de complejidad de un proyecto software y que el mismo pueda ser entregado con calidad de manera breve. En otras palabras, *Scrum* es una metodología que, si bien no proporciona instrucciones detalladas, pautas o reglas a seguir, es una guía para ser implementada entre los colaboradores del

proyecto [24]. A continuación, se presentan los roles y artefactos para que la metodología *Scrum* pueda llevarse a cabo con éxito.

Roles

En esta metodología existen 3 roles principales los cuales son: *Scrum Master, Product Owner* y *Development Team.* Estos roles contribuyen entre sí a lo largo del proyecto, logrando una participación más activa y organizada para el cumplimiento del objetivo propuesto [25].

Product Owner

Denominado el dueño del producto o el representante de los clientes que requieren el proyecto. Es decir, es el responsable de establecer los requerimientos del sistema software a desarrollar y de trasladar esa visión a todo el equipo de trabajo [25]. En ese sentido, en la **TABLA I** se evidencia la persona encargada para este rol.

Scrum Master

Es el líder del equipo de desarrollo encargado de que estos cumplan con los procesos de la metodología, el *Scrum Master* trabaja de la mano con el *Product Owner* para mantener y manejar las tareas previstas por este. Su principal objetivo es la reducción de las interferencias que ralenticen el desarrollo del proyecto [25]. Por tal razón, en la **TABLA I** se evidencia la persona encargada para este rol.

Development Team

Es el grupo de programadores con las habilidades necesarias para cumplir cada una de las tareas proporcionadas por el *Product Owner*, también son los encargados de presentar entregables funcionales al cliente por cada *Sprint* que se haya planificado [25]. Muestra de ello, en la **TABLA I** se evidencia a la persona encargada para este rol.

TABLA I: Asignación de Roles.

ROLES	NOMBRES
Product Owner	Ing. Byron Loarte, MSc.
Scrum Master	Ing. Byron Loarte, MSc.
Development Team	Iván Fraga.

Artefactos

Los artefactos de *Scrum* son elementos diseñados para registrar información únicamente fundamental del proyecto, mejorando así la productividad y calidad de este, dicho de otra manera, son los recursos que presenta *Scrum* para tomar y organizar toda la información necesaria para el desarrollo del proyecto y así evitar tomar decisiones erróneas en el transcurso del proyecto que provoquen retrasos innecesarios [26]. Es por ese motivo que a continuación, se presentan los artefactos que se han utilizado para el desarrollo del presente *backend*.

Recopilación de requerimientos

La actividad de recopilar los requerimientos define las necesidades que debe cumplir el proyecto. Los requisitos deben ser recabados, analizados y registrados con un alto nivel de detalle, el cual permita que una vez definidos pueda comenzar la etapa de codificación del proyecto sin problemas [27]. En la **TABLA II** se presenta el formato que se ha utilizado para la Recopilación de requerimientos. Sin embargo, la tabla completa con todos los requerimientos se detalla en el **ANEXO II** del presente documento.

TABLA II: Formato para la Recopilación de requerimientos.

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
BACKEND	**RR001**	Como usuarios administrador y artista necesitan implementar varios métodos para: • Iniciar sesión • Cerrar sesión. • Recuperar contraseña.
	RR002	Como usuario administrador necesita implementar varios métodos para: • Modificar perfil de usuario.

Historias de Usuario

Es una presentación corta, simple e informal de una función del proyecto, pero realizada desde la perspectiva del usuario que desea la función. Además, las historias de usuario permiten la asignación ordenada de cada una de las tareas para así evitar una mala

distribución de las tareas con el *Development Team* [28]. Es por esta razón, que en la **TABLA III** se presenta el formato que se ha utilizado para elaborar cada una de las Historias de Usuario. Sin embargo, las demás tablas completas se detallan en el **ANEXO II** del presente documento.

TABLA III: Historia de Usuario Nro. 1.

HISTORIA DE USUARIO	
Identificador (ID): HU001	**Usuario**: Administrador y Artista
Nombre Historia: Iniciar sesión, cerrar sesión y cambiar contraseña.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil administrador y artista permite generar varios métodos para: • Iniciar Sesión. • Cerrar Sesión. • Cambiar contraseña.	
Observación: Los usuarios administrador y artista necesitan estar registrados para acceder a los métodos anteriormente mencionados. Para cambiar la contraseña se envía un correo al e-mail del usuario respectivo.	

Product Backlog

Es una lista que resume las necesidades presentadas en la Recopilación de requerimientos. La lista del *Product Backlog* es una lista dinámica, es decir, puede ir variando a medida que avance el proyecto. Cada elemento o requerimiento de la lista contiene la descripción de tareas a cumplir, estas a su vez se irán desarrollando en función de las prioridades que se vayan presentando [29]. En la **TABLA IV,** se presenta el formato que se ha empleado para listar el *Product Backlog*, mientras que la tabla completa se detalla en el **ANEXO II** del presente documento.

TABLA IV: Formato del *Product Backlog*.

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID – HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU001	Iniciar sesión, cerrar sesión y recuperar contraseña.	1	Finalizada	Media
HU002	Modificar perfil de usuario.	1	Finalizada	Media

Sprint Backlog

Es una lista que se genera después de la creación del *Product Backlog* como se ha mencionado anteriormente, pero con mayor detalle sobre cada una las tareas que se deben realizar, estas son colocadas y clasificadas por medio de *Sprints* o Iteraciones [29]. No obstante, para desarrollar cada una de las tareas estas cuentan con un tiempo determinado para su cumplimiento. Muestra de ello, en la **TABLA V** se presenta el formato que se ha empleado para listar los *Sprints* mientras que la tabla completa con todas las tareas se detalla en el **ANEXO II** del presente documento.

TABLA V: Formato del *Sprint Backlog*.

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID-SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIA DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB000	Configuración del ambiente de desarrollo	-------	-----	----------	• Recopilación y definición de requerimientos. • Elaboración del Modelo de Datos. • Roles de los usuarios. • Elaboración de la Base de datos en *Firebase*.	20 H

2.2 Diseño de la arquitectura

El definir una arquitectura de software es de suma relevancia en el proyecto a realizar, pues de esta depende el correcto funcionamiento y mantenibilidad que tendrá el proyecto luego de que salga a producción. La arquitectura de software consiste en definir los componentes del proyecto y su combinación interna [30]. Por lo anterior citado se ha establecido una arquitectura para la etapa de codificación del *backend* y su integración con otras herramientas y librerías.

Patrón Arquitectónico

El patrón arquitectónico que se ha empleado en este proyecto es Modelo Vista Controlador (MVC), el cual emplea 3 componentes enfocados en: gestionar la entrada y salida de información al sistema, la obtención de datos y la interfaz visual presentada al usuario [31].

- **Modelo**: componente encargado de la creación, actualización y eliminación de los datos, es decir la manipulación de estos. Así mismo el modelo es el encargado de la obtención de datos [31].
- **Vista**: componente responsable de presentar al usuario la parte gráfica e interactúa directamente con el usuario, es decir el resultado gráfico de la comunicación entre el modelo y el controlador [31].
- **Controlador**: componente encargado de la gestión de las peticiones realizadas tanto por el sistema como por los usuarios [31].

La **Fig. 1**, presenta el patrón arquitectónico que se ha usado para el desarrollo del *backend*, este presenta una serie de ventajas tales como: estructura organizada, escalable y de fácil mantenimiento.

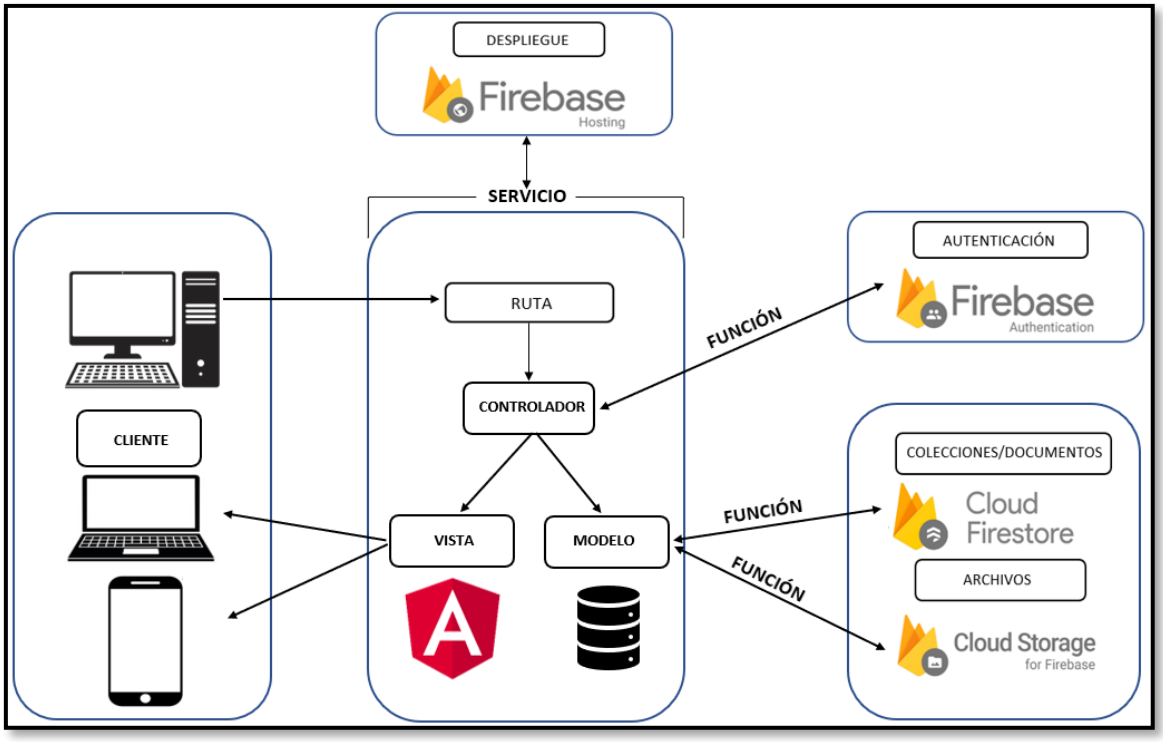

Fig. 1: Patrón Arquitectónico – *backend*.

2.3 Herramientas de desarrollo

Una vez que se ha definido el patrón arquitectónico se procede a la selección de herramientas de desarrollo de software. Estas permiten el modelado, codificación, integración, etc. Por lo que consecuentemente facilitan y agilizan el proceso de desarrollo [32]. Por esta razón la **TABLA VI** presenta las herramientas empleadas para la codificación del *backend*.

TABLA VI: Herramientas para el desarrollo del *backend*.

HERRAMIENTA	JUSTIFICACIÓN
Cloud Firestore	Es una base de datos *NoSQL* multifuncional que permite almacenar, consultar y sincronizar información con aplicaciones *web* y móviles en tiempo real [14].
Firebase Authentication	Es un servicio propio de *Firebase* que facilita autenticación de usuarios mediante correo electrónico u otros medios [14].
Cloud Storage for Firebase	Es un servicio propio de *Firebase* diseñado para almacenar y gestionar con rapidez archivos multimedia generados por usuarios [15].

Firebase Hosting	Es un servicio de *hosting* de *Firebase* diseñado para implementar aplicaciones *web*, móviles, etc. [16].
Angular	Es un *Framework* para el desarrollo de aplicaciones del lado del *frontend*, sin embargo, se utiliza para generar una comunicación con el *backend* y para demostrar el correcto funcionamiento de sus métodos [8].
Visual Studio Code	Es un editor de código personalizable y compatible con múltiples lenguajes de programación [19].

Librerías

La **TABLA VII** muestra las librerías que se han implementado para el desarrollo del *backend* y la incorporación con las herramientas listadas anteriormente.

TABLA VII: Librerías para el desarrollo del *backend*.

LIBRERÍA	DESCRIPCIÓN
AngularFire	Es una librería de *Node.js*, que se encarga de acoplar las funcionalidades *Firebase* a las necesidades de Angular [33].
Bootstrap	Es un Framework CSS que ayuda al diseño ágil y *responsive* de aplicaciones web y móviles [34].

3 RESULTADOS

En esta sección, se detallan cada uno de los resultados que se han obtenido en la implementación de los diferentes métodos para el *backend,* así como el resultado de las pruebas y el proceso de despliegue a producción. No obstante, cada uno de los resultados se presenta por medio de *Sprints.*

3.1 *Sprint* 0. Configuración del ambiente de desarrollo.

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Recopilación y definición de requerimientos.
- Elaboración del Modelo de Datos en *Firebase*.
- Roles de usuarios.

Recopilación y definición de requerimientos

Implementar métodos para el registro de usuarios

El *backend* implementa un método propio de *Firebase* que permite el registro de usuarios mediante campos predefinidos en el sistema *web* y aplicación móvil.

Implementar métodos para iniciar sesión, cerrar sesión y cambiar contraseña.

El *backend* implementa un método que permite el inicio de sesión a usuarios con perfil artista y administrador en el sistema *web* y con perfil ciudadano en la aplicación móvil. Un método para el cierre de sesión de los usuarios con los perfiles mencionados anteriormente y un último método para el cambio de contraseña.

Implementar métodos para gestionar las solicitudes de los artistas

El *backend* implementa varios métodos que permiten gestionar las solicitudes de usuarios artistas por parte del usuario administrador, este puede aprobar o rechazar las solicitudes en función de un formulario enviado por parte del usuario artista, en caso de que el usuario artista sea aprobado, este puede hacer uso de los demás métodos propios del usuario artista para subir su contenido musical.

Implementar métodos para modificar el perfil de usuario

El *backend* implementa varios métodos que permiten la visualización y actualización del perfil de los usuarios artista y administrador para el sistema *web*, y usuario ciudadano en la aplicación móvil.

Implementar métodos para gestionar géneros musicales

El *backend* implementa varios métodos que permiten al usuario artista, previamente aprobado, visualizar, seleccionar o crear un género musical para el proceso de subir contenido musical. El proceso de subir contenido musical consiste en la creación o selección de un género musical, seguido de la creación o selección de un álbum para finalmente subir el contenido musical.

Implementar métodos para gestionar álbumes

El *backend* implementa varios métodos que permiten al usuario artista visualizar, seleccionar o crear un álbum. En caso de que el artista quiera seleccionar o crear un álbum, debe tener en cuenta que previamente a ello debe haber seleccionado un género musical. En el caso de que el género se haya creado obligatoriamente el usuario debe crear un álbum.

Implementar métodos para gestionar canciones

El *backend* implementa varios métodos que permiten al usuario artista visualizar, subir, editar o eliminar contenido musical. En el caso de que el usuario quiera subir contenido musical debe haber seleccionado o creado un álbum previamente, caso contrario no puede subir contenido musical.

Implementar métodos para gestionar favoritos

El *backend* implementa varios métodos que permiten al usuario ciudadano agregar y eliminar canciones a su lista favoritos mientras visualiza canciones.

Implementar métodos para gestionar *playlist*

El *backend* implementa varios métodos que permiten al usuario ciudadano crear y actualizar *playlist*.

Implementar métodos para gestionar las canciones al *playlist*

El *backend* implementa varios métodos que permiten al usuario ciudadano una vez creada la *playlist* agregar o quitar canciones de esta.

A continuación, la **Fig. 2**, **Fig. 3** y **Fig. 4** muestran los diferentes métodos a los que pueden interactuar los usuarios del *backend*.

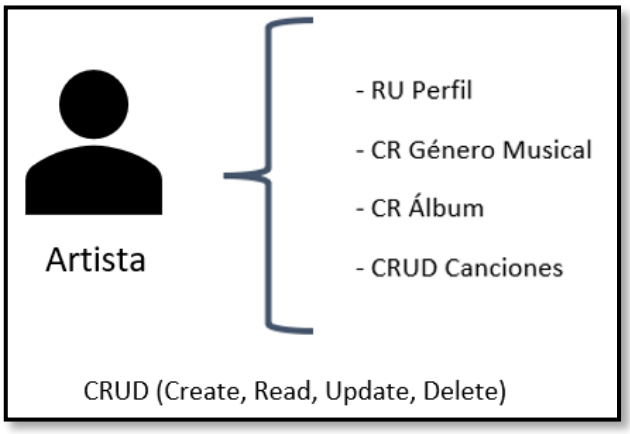

Fig. 2: Usuario con perfil artista.

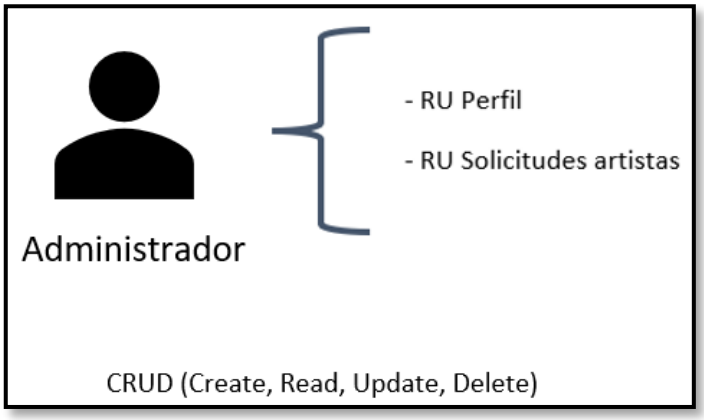

Fig. 3: Usuario con perfil administrador.

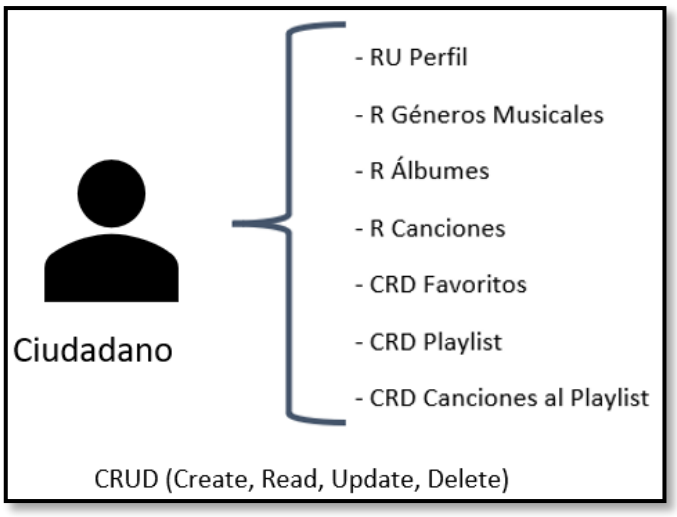

Fig. 4: Usuario con perfil ciudadano.

Elaboración del Modelo de Datos en *Firebase*

Para el almacenamiento de datos que se necesita en este proyecto se ha optado por la plataforma *Firebase*, específicamente por la base de datos *Firebase Firestore*, la cual permite sincronizar y almacenar datos en tiempo real por medio de colecciones y documentos [5]. Además, la **Fig. 5** ilustra el modelo de datos que se ha utilizado para la comunicación con el *backend*, mientras que el esquema completo se presenta en el **ANEXO II** del presente documento.

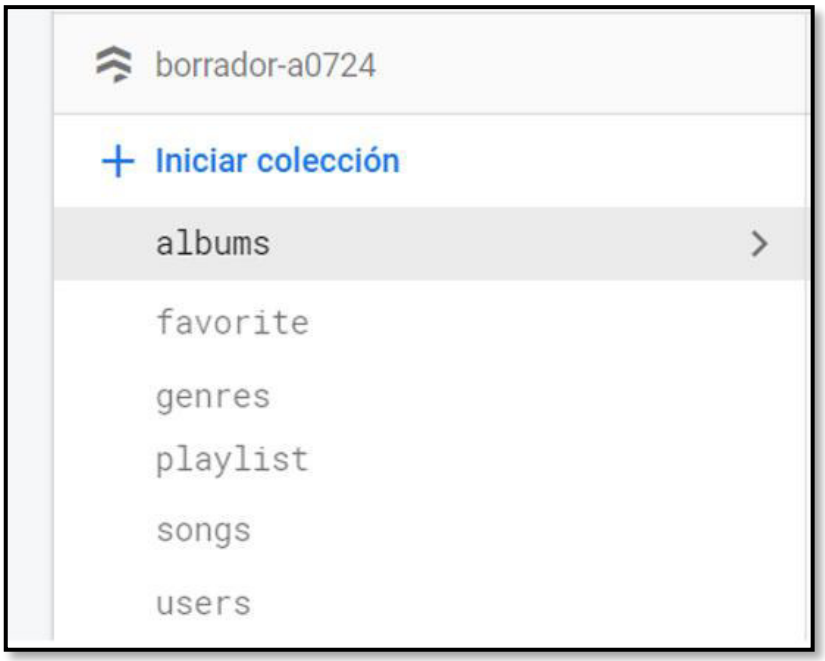

Fig. 5: Modelo de datos en *Firebase.*

Roles de usuarios

A continuación, la **Fig. *6*** ilustra los cuatro usuarios y los módulos a los cuales tienen acceso en función del rol asignado.

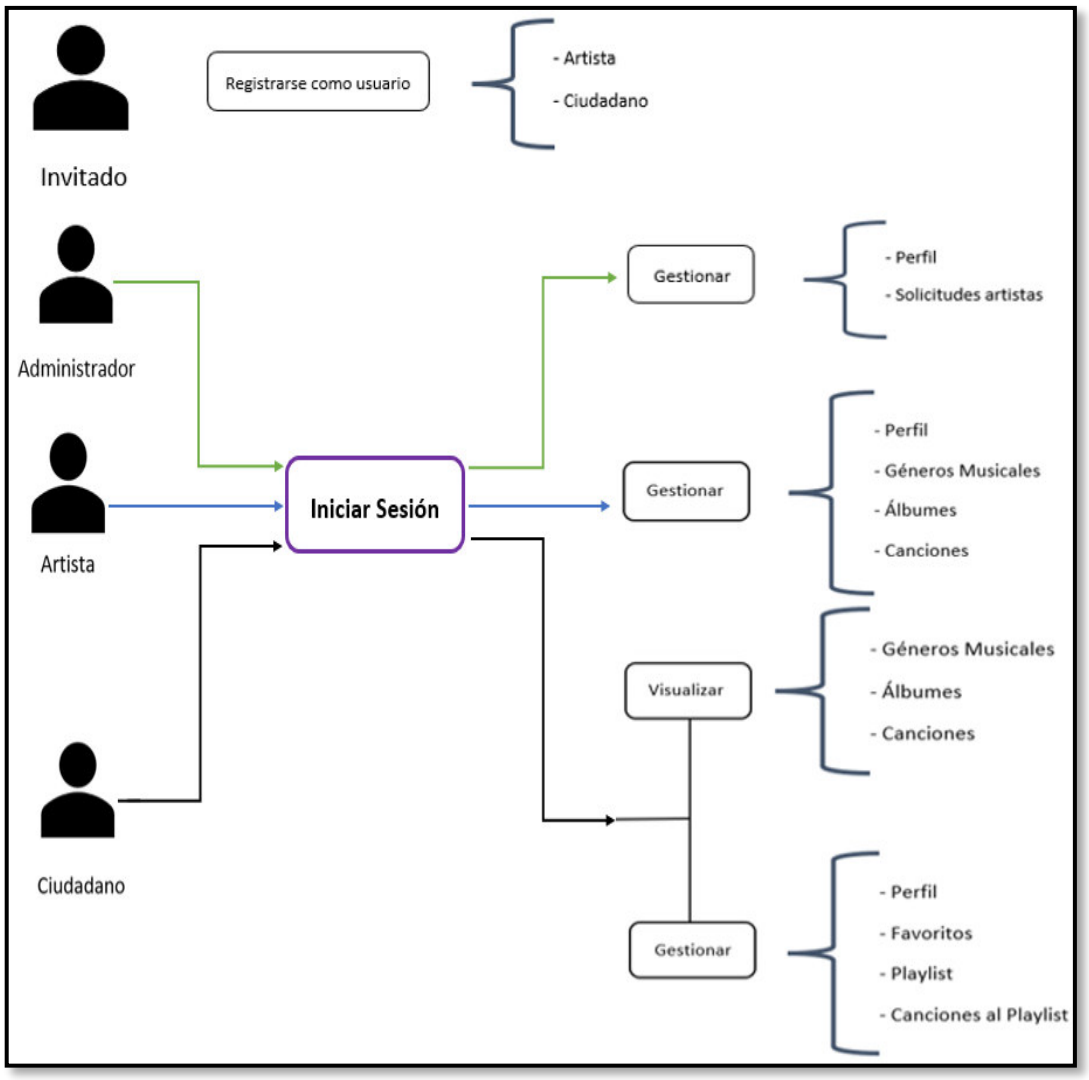

Fig. 6: Roles de usuario.

3.2 *Sprint* 1. Diseño e implementación de métodos para el usuario administrador.

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Implementar varios métodos para iniciar sesión, cerrar sesión y cambiar contraseña.
- Implementar varios métodos para visualizar y editar el perfil de usuario.
- Implementar varios métodos para visualizar, aceptar y rechazar solicitudes de artistas.

Implementar varios métodos para iniciar sesión, cerrar sesión y cambiar contraseña

Para el inicio de sesión, cierre de sesión y cambio de contraseña se han implementado varios métodos que ayudan al usuario artista a tener un mayor control al consumir información de la base de datos. El método para iniciar sesión solicita correo y contraseña para la autenticación como se muestra en la **Fig. 7**. El método para cambiar la contraseña solicita el correo del usuario como se muestra en la **Fig. 8**, consecuentemente envía un enlace al correo del usuario del cual puede reestablecer su contraseña. El método para cerrar sesión ayuda al usuario a desconectase del *backend* como se muestra en la **Fig. 9**. No obstante, los métodos aquí explicados también son implementados en los usuarios con perfil artista y ciudadano. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//login method
//Método para Acceder al sistema
//Necesita parámetro: objeto User
login(user: User) {
  //método de firebase para acceder al sistema mediante el correo y contraseña
  return this.fireauth.signInWithEmailAndPassword(user.mail,user.password);

}
```

Fig. 7: Método para iniciar sesión.

```
//forgot password method
//Método para reestablecer contraseña
//Necesita parámetro: correo
forgotPassword(email : string) {
  //método de firebase para reestablecer contraseña mediante correo
  this.fireauth.sendPasswordResetEmail(email).then(() => {
    //redireccionar al dashboard
    this.router.navigate(['/login']);
  }, err => {
    console.log("error: ", err)
    alert('Something went wrong');
  })
}
```

Fig. 8: Método para cambiar contraseña.

```
//logout method
//Método para Salir del sistema
//No necesita parámetros
logout(){
  //método de firebase para salir del sistema
  this.fireauth.signOut().then(()=>{
    //deshabilitar el token
    localStorage.clear();
    //redieccionar al Inicio de sesión
    this.router.navigate(['/login']);
  }, err=>{
    alert(err.message);

  })
}
```

Fig. 9: Método para cerrar sesión.

Implementar varios métodos para visualizar y editar el perfil de usuario

Para visualizar y editar el perfil de usuario administrador se han implementado varios métodos que ayudan al usuario administrador a personalizar su perfil. El método que se ha utilizado para visualizar y verificar la obtención de los datos en función del ID del usuario se muestra en la **Fig. 10**. El método que se ha utilizado para editar los datos del usuario en base a su ID se muestra en la **Fig. 11** el cual procede a actualizarse en tiempo real en la base de datos NoSQL. No obstante, los métodos aquí explicados también son implementados en los usuarios con perfil artista y ciudadano. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//Obtener documento específico
//necesita parámetros: nombre de la colección, id del documento
getObject(collection: string, id: any){
  return this.angularFirestore//accede a Firestore
  .collection(collection)//especifica la colección
  .doc(id)//especifica el documento
  .valueChanges()//obtiene el documento
}
```

Fig. 10: Método para visualizar perfil de usuario.

```
//función para actualizar perfil
//necesita parámetros: objeto usuario, url, referencia en FireStorage
updateUser(object: any, url: any, path: any, id: any){
  return this.firestore.collection("users")
  //referencia al id del usuario
  .doc(id)
    //actualización de los siguientes campos
    .update({
      name: object.name,
      imageURL : url,
      image_reference: path
  })
}
```

Fig. 11: Método para editar perfil de usuario.

Implementar varios métodos para visualizar, aceptar y rechazar solicitudes de artistas

Para visualizar, aceptar y rechazar las solicitudes de artistas se han implementado varios métodos que ayudan al usuario con perfil administrador a controlar a todos los usuarios que se puedan convertir en perfil artista. El método que se ha utilizado para visualizar los datos de los usuarios con rol de artista y no artista se muestra en la **Fig. 12**. El método que se ha utilizado para aceptar la solicitud y cambiar el rol del usuario de “no artista” a ”artista” se muestra en la **Fig. 13**, mientras que el método que se ha utilizado para rechazar la solicitud se muestra en la **Fig. 14**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//traemos todos los no artistas
getNoArtist(){
  return this.angularFirestore
  .collection("users", ref => ref.where('rol', '==', 'no artist')) //filtrado de usuarios no artistas
  .snapshotChanges()
}
//traemos todos los no artistas
getArtist(){
  return this.angularFirestore
  .collection("users", ref => ref.where('rol', '==', 'artist')) //filtrado de usuarios artistas
  .snapshotChanges()
}
```

Fig. 12: Método para visualizar solicitudes de artistas.

```
//cambiar de rol
//necesita parámetro: objeto artista
artistRol(artist: Artist)
{
  console.log("el rol a actualizar es del artista: ", artist.id )
  const au = this.angularFirestore.doc(`users/${artist.id}`)
  au.update({
    rol: artist.rol= "artist"
    })
}
```

Fig. 13: Método para aceptar solicitudes de artistas.

```
noArtistRol(artist: Artist){
  console.log("el rol a actualizar es: ")
  return this.angularFirestore
    .collection("citizen")
    .doc(artist.id)
    .update({
      rol: artist.rol= "no artist"
    })
}
```

Fig. 14: Método para rechazar solicitudes de artistas.

3.3 *Sprint* 2. Desarrollo e implementación de métodos para el usuario artista.

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Implementar un método para el registro de usuarios a través de un formulario.
- Implementar varios métodos para gestionar géneros musicales.
- Implementar varios métodos para gestionar álbumes.
- Implementar varios métodos para gestionar canciones.

Implementar un método para el registro de usuarios a través de un formulario

Para el registro de usuarios se ha implementado un método que ayuda al usuario con perfil invitado a registrarse con perfil artista. El método que se ha utilizado para el registro de usuarios se lo realiza por medio de un formulario y luego a registrarse en la base de datos

NoSQL como se muestra en la **Fig. 15**. No obstante, el método aquí explicado también es implementado en el usuario ciudadano. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//register method
//Función para Registrarse en el sistema
//Necesita parámetros: objeto usuario
register(user: User) {
  //método de firebase para registrarse en el sistema mediante el correo y contraseña

  console.log("user value: ", user)
  //método de firebase para registrarse en el sistema mediante el correo y contraseña
  return this.fireauth.createUserWithEmailAndPassword(user.mail, user.password)
}
//Función para crear usuario en firestore
//Necesita parámetros: objeto usuario, dirección de la colección
createUser(user: User, path: string){
  //ubica segun la dirección en la colección correspondiente
  this.firestore.collection(path)
  //crea segun la id registrada en direauthentication
  .doc(user.id)
  //establece al usuario segun los campos de usuario
  .set({id: user.id,
  mail: user.mail,
  name: user.name,
  birthdate: user.birthdate,
  rol: user.rol,
  imageURL: user.imageURL,
  image_reference: user.image_reference,
  });
}
```

Fig. 15: Método para el registro de usuarios.

Implementar varios métodos para gestionar géneros musicales

Para gestionar géneros musicales se han implementado varios métodos que ayudan al usuario con perfil artista a visualizar crear o seleccionar géneros musicales. El método que se ha utilizado para visualizar géneros musicales se muestra en la **Fig. 16**. El método que se ha utilizado para crear géneros musicales a la base de datos *NoSQL* se muestra en la **Fig. 17**. El método que se ha utilizado para seleccionar un género musical a partir de los géneros que se han creado y continuar con el proceso de subir contenido musical se muestra en la **Fig. 18**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
getGenres(collection: string){
  return this.angularFirestore
  //Búsqueda de géneros que pertenezcan al artista
  .collection(collection, ref => ref.where('authorId', '==', localStorage.getItem("idUser")))
  .snapshotChanges();
}
```

Fig. 16: Método para visualizar géneros musicales.

```
//función para crear un género
//necesita parámetros: objeto género, url, referencia en FireStorage
genreCreate(genre: Genre, urlImg:any, filePath: any) {
 const id = this.angularFirestore.createId();//crea un ID
 console.log("el id del género es: ", id)
 //crea un documento con los campos especificados
 this.genreCollection
   .doc(id)//especifica el documento
   //establece y crea documento mediante los campos especificados
   .set({id,
     name: genre.name,
     imageURL: urlImg,
     author: localStorage.getItem("nameUser"),
     authorId: localStorage.getItem("idUser"),
     image_reference: filePath });
}
```

Fig. 17: Método para crear géneros musicales.

```
//función para obtener datos del género
getGenreSongProperties(genre: Genre){
  //asignación de datos de género en variables genéricas
  localStorage.setItem("genreName", genre.name);
}
```

Fig. 18: Método para seleccionar un género musical.

Implementar varios métodos para gestionar álbumes

Para gestionar álbumes se han implementado varios métodos que ayudan al usuario con perfil artista a visualizar crear o seleccionar álbumes. El método que se ha utilizado para visualizar álbumes se muestra en la **Fig. 19**. El método que se ha utilizado para crear álbumes a la base de datos *NoSQL* se muestra en la **Fig. 20**. El método que se ha utilizado para seleccionar un álbum a partir de los álbumes que se ha creado y continuar con el proceso de subir contenido musical se muestra en la **Fig. 21**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//función para obtener álbumes correspondientes al género
getAlbums(){
  return this.angularFirestore
  .collection("albums", ref => ref.where('genre_name', '==', localStorage.getItem("genreName")))
  .snapshotChanges();
}
```

Fig. 19: Método para visualizar álbumes.

```
//función para crear un álbum
//necesita parámetros: objeto género, url, referencia en FireStorage
albumCreate(album: Album, urlImg:any, filePath: any) {
  const id = this.angularFirestore.createId();//crea un ID
  localStorage.setItem("imageURL", urlImg);
  //crea un documento con los campos especificados
  this.albumCollection
    .doc(id)//especifica el documento
    //establece y crea documento mediante los campos especificados
    .set({id,
      //nombre del género por parte de variable general
      //genre_name: this.genre_name,
      genre_name: localStorage.getItem("genreName"),
      name: album.name,
      imageURL: urlImg,
      author: localStorage.getItem("nameUser"),
      image_reference: filePath,
    year: album.year });
```

Fig. 20: Método para crear álbumes.

```
//función para obtener datos del álbum
getAlbumSongProperties(album: Album){
  //asignación de datos de álbum en variables genéricas
  localStorage.setItem("album_name", album.name);
  localStorage.setItem("imageURL", album.imageURL);
  localStorage.setItem("year", String(album.year));

}
```

Fig. 21: Método para seleccionar un álbum.

Implementar varios métodos para gestionar canciones

Para gestionar canciones se han implementado varios métodos que ayudan al usuario con perfil artista a visualizar, subir, editar o eliminar contenido musical. El método que se ha utilizado para visualizar canciones se muestra en la **Fig. 22**. El método que se ha utilizado para subir canciones a la base de datos *NoSQL* se muestra en la **Fig. 23**. Para subir

canciones se debe tener en cuenta de haber creado o seleccionado el género y álbum, es decir completar los pasos previos para el proceso de subir contenido musical. El método que se ha utilizado para editar una canción se muestra en la **Fig. 24** el cual procede a actualizarse en la base de datos *NoSQL*. El método que se ha utilizado para eliminar una canción se muestra en la **Fig. 25** para posteriormente quitarla de la base de datos *NoSQL*. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//Función para obtener canciones pertenecientes a album
//necesita parámetro: nombre de álbum
getAlbumSongs(collection: string){
  return this.angularFirestore
  //Búsqueda de canciones que pertenezcan al álbum
  .collection(collection, ref => ref.where('album_name', '==', localStorage.getItem("album_name")))
  .snapshotChanges();
}
```

Fig. 22: Método para visualizar canciones.

```
//función para crear canción
//necesita parámetros: objeto canción, url, referencia en FireStorage
songCreate(song: Song, urlSong:any, filePath: any){
  const id = this.angularFirestore.createId();//crea un ID
  //obtener datos de variables genéicas para asignarlas a campos de canción
  this.song_name= song.song_name;
  localStorage.setItem("song_name", song.song_name);
  this.songURL= urlSong;
  this.song_reference= filePath;
  this.id= id;
  //crea un documento con los campos especificados
  this.songCollection
    .doc(id)
    .set({
      id,
      genre_name: localStorage.getItem("genreName"),
      album_name: localStorage.getItem("album_name"),
      imageURL: localStorage.getItem("imageURL"),
      song_name: song.song_name,
      year: parseInt(localStorage.getItem("year")),
      author: localStorage.getItem("nameUser"),
      songURL: urlSong,
      song_reference: filePath,
      authorId: localStorage.getItem("idUser")
    })
}
```

Fig. 23: Método para subir canciones.

```
//función para actualizar canción
//necesita parámetros: objeto canción, url, referencia en FireStorage
updateSong(song: Song, urlSong:any, filePath: any, id:any){
  //obtener datos de variables genéicas para asignarlas a campos de canción
  this.album_name= song.album_name;
  this.song_name= song.song_name;
  this.songURL= urlSong;
  this.song_reference= filePath;
  localStorage.setItem("song_name", song.song_name);
  return this.songCollection
    .doc(id)//referencia al documento por id
    //actualización de los siguientes campos de la canción
    .update({
      song_name: song.song_name,
      songURL: urlSong,
      song_reference: filePath
  })
}
```

Fig. 24: Método para editar una canción.

```
//función para eliminar canción
//necesita parámetro: objeto canción
deleteSong(song){
  //Eliminar archivo segun la referencia de la canción en FireStorage
  this.storage.ref(song.song_reference).delete();
  //Eliminar documento segun el id de la canción en Firestore
  return this.songCollection
  .doc(song.id)
  .delete();
}
```

Fig. 25: Método para eliminar una canción.

3.4 *Sprint* 3. Diseño e implementación de métodos para el usuario ciudadano.

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Implementar un método para visualizar canciones y sus datos relacionados.
- Implementar varios métodos para gestionar favoritos.
- Implementar un método para visualizar géneros y álbumes musicales.
- Implementar varios métodos para gestionar *playlist*.
- Implementar varios métodos para gestionar canciones al *playlist*.

Implementar un método para visualizar canciones y sus datos relacionados

Para visualizar canciones y sus datos relacionados se ha implementado un método que ayuda al usuario ciudadano a visualizar un detalle sobre la canción que desee. El método que se ha utilizado para visualizar canciones y sus datos relacionados tales como: género musical, álbum, año, nombre de la canción, imagen, entre otros, se muestra en la **Fig. 26**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//Obtener todos los documentos de la colección
//necesita parámetro: nombre de la colección
getList(collection: string){
 return this.angularFirestore//accede a Firestore
  .collection(collection)//especifica la colección
  .snapshotChanges();//obtiene los documentos
}
```

Fig. 26: Método para visualizar canciones y sus datos relacionados.

Implementar varios métodos para gestionar favoritos

Para gestionar favoritos se ha implementado varios métodos que ayudan al usuario ciudadano a seleccionar y visualizar sus canciones favoritas. El método que se ha utilizado para seleccionar canciones favoritas se muestra en la **Fig. 27**. El método que se ha utilizado para visualizar favoritos en una *playlist* se muestra en la **Fig. 28**. Si las canciones favoritas se proceden a deseleccionar, estas se quitan de la lista de favoritos. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
public userID=localStorage.getItem("idUser");//seria el usuario que inicia sesión
//función para añadir canciones a favoritos
//necesita parámetros: objeto canciones
addToFavorite(song: Song){
  //la dirección de la colección se determina por el ID del usuario
  const path="/favorite/"+this.userID+"/songs"
  console.log("La canción guadada es: ", song.song_name,
  "\n Su ID: ", song.id)
  //se procede a guardar el ID de la canción
  return this.angularFirestore
    //se guarda en la dirección anteriomente especificada
   .collection(path)
   //toma el nombre del ID de la canción
   .doc(song.id)
   //establece el documento mediante el ID de la canción
   .set({
    id: song.id
   })
```

Fig. 27: Método para seleccionar canciones favoritas.

```
//dirección de favoritos
collection= "/favorite/"+ this.id + "/songs";
//Obtener todos los documentos de la colección
//necesita parámetro: nombre de la colección
getList(collection: string){
 return this.angularFirestore//accede a Firestore
  .collection(collection)//especifica la colección
  .snapshotChanges();//obtiene los documentos
}
```

Fig. 28: Método para visualizar favoritos.

Implementar varios métodos para visualizar géneros y álbumes musicales.

Para visualizar géneros musicales y álbumes se han implementado varios métodos que ayudan al usuario ciudadano a visualizar el detalle sobre géneros y álbumes musicales. El método que se ha utilizado para visualizar géneros musicales se muestra en la **Fig. 29**. El método que se ha utilizado para visualizar álbumes y las canciones pertenecientes al álbum musical se muestra en la **Fig. 30**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
 collection= "genres"
//Obtener todos los documentos de la colección
//necesita parámetro: nombre de la colección
getList(collection: string){
 return this.angularFirestore//accede a Firestore
  .collection(collection)//especifica la colección
  .snapshotChanges();//obtiene los documentos
}
```

Fig. 29: Método para visualizar géneros musicales.

```
 collection= "albums"
//Obtener todos los documentos de la colección
//necesita parámetro: nombre de la colección
getList(collection: string){
 return this.angularFirestore//accede a Firestore
  .collection(collection)//especifica la colección
  .snapshotChanges();//obtiene los documentos
}
```

Fig. 30: Método para visualizar álbumes.

Implementar varios métodos para gestionar *playlist*

Para gestionar *playlist* se han implementado varios métodos que ayudan al usuario ciudadano a crear y visualizar *playlists*. El método que se ha utilizado para crear un *playlist* en la base de datos NoSQL se muestra en la **Fig. *31***. El método que se ha utilizado para visualizar *playlists* se muestra en la **Fig. *32***. El método que se ha utilizado para seleccionar una *playlists* se muestra en la **Fig. 33**, caso contrario la *playlist* no puede ser eliminada. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//función para crear una playlist
//necesita parámetros: objeto playlist, url, referencia en FireStorage
 playlistCreate(playlist: Playlist) {
  const id = this.angularFirestore.createId();//crea un ID
  playlist.id= id;
  //crea un documento con los campos especificados
  this.getPlaylistProperties(playlist);
  const path = "playlist/"+this.userID+"/playlist";
  this.angularFirestore.collection(path).
    doc(id)
    //establece y crea documento mediante los campos especificados
    .set({
      id,
      playlist_name: playlist.playlist_name,
      playlist_collection: playlist.playlist_collection});

}
```

Fig. 31: Método para crear *playlists*.

```
//función para obtener playlist
//No necesita parámetros
showPlaylists(){
  //dirección de las playlists
  const path =  "playlist/"+this.userID+"/playlist";
  //retorna los documentos de la colección segun la dirección
  return this.angularFirestore.collection(path)
  .snapshotChanges();//obtiene los documentos
}
```

Fig. 32: Método para visualizar *playlists*.

```
//función para obtener los campos de playlist
//Necesita parámetros: objeto playlist
getPlaylistProperties(playlist: Playlist){
  //actualiza el valor de las variables generales con los campos de la playlist
  localStorage.setItem("idPlaylist", playlist.id);
  localStorage.setItem("playlist_name", playlist.playlist_name);

  this.id= localStorage.getItem("idPlaylist");
  this.playlist_name= localStorage.getItem("playlist_name");
  this.playlist_songs= playlist.playlist_collection
  console.log("propiedades de la playlist: ", playlist)
}
```

Fig. 33: Método para seleccionar *playlist*.

Implementar varios métodos para gestionar canciones al *playlist*

Para gestionar canciones al *playlist* se han implementado varios métodos que ayudan al usuario ciudadano agregar y quitar canciones al *playlist.* El método que se ha utilizado para agregar canciones al *playlist* se muestra en la **Fig. 34**. El método que se ha utilizado para visualizar las canciones agregadas al *playlist* se muestra en la **Fig. 35**. El método que se ha utilizado para eliminar canciones de la *playlist* se muestra en la **Fig. 36**. Adicionalmente, el proceso a detalle para visualizar completamente la implementación de estos métodos se lo puede apreciar en el **ANEXO III** del presente documento.

```
//función para añadir canciones al playlist
//Necesita parámetros: lista de ids de canciones
addPlaylistSongs(playlistSongs: string[]){
  //dirección de la playlist
  const path = "playlist/"+this.userID+"/playlist";
  //añadir canciones a las ya existentes en el album
  this.playlist_songs.push(...playlistSongs);
  //actualizar la lista de canciones del album
  return this.angularFirestore.collection(path)
  .doc(this.id)
  .update({
    playlist_collection: this.playlist_songs
  })
}
```

Fig. 34: Método para agregar canciones al *playlist*.

```
//función para obtener canciones de playlist mediante la lista de ids
//necesita parámetros: id de la playlist
getPlaylistById( id: any){

  return this.angularFirestore
  //Búsqueda de canciones que pertenezcan al álbum
  .collection("songs", ref => ref.where('id', 'in', id))
  .snapshotChanges();
}
```

Fig. 35: Método para visualizar canciones del *playlist.*

```
//Eliminar canción de la playlist
//necesita parámetros: id de la canción
deletePlaylistSong(id: string){
  //dirección de la playlist
  const path = "playlist/"+this.userID+"/playlist";
  //nuevo array que obtiene los ids omitiendo el id a eliminar
  var newArray = this.playlist_songs.filter((item) => item !== id);
  //asignación de la nueva lista de ids
  this.playlist_songs= newArray;
  //actualización de la lista de ids
  return this.angularFirestore.collection(path)
  .doc(this.id)
  .update({
    playlist_collection: this.playlist_songs
  })
}
```

Fig. 36: Método para eliminar canciones de la *playlist.*

3.5 *Sprint* 4. Pruebas del *backend*.

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene las siguientes tareas:

- Ejecución de pruebas unitarias y resultados.
- Ejecución de pruebas de rendimiento y resultados.
- Ejecución de pruebas de aceptación y resultados.

Ejecución de pruebas unitarias y resultados

Una vez que se ha finalizado el proceso de implementación de todos los métodos como parte del *backend* y continuando con la metodología, se procede a ejecutar las pruebas unitarias permitiendo detectar todos los fallos que existan en cada uno de los métodos que se han implementado. En ese sentido para realizar dichas pruebas se ha utilizado *Karma*, la cual es una herramienta que permite la automatización de tareas de *testing* y que a su

vez está desarrollado por el equipo de Angular, logrando de esta manera tener una mayor garantía al momento de testear funcionalidades en Angular [35].

La **Fig. 37** muestra una parte del código que se ha implementado para el inicio de sesión de usuarios, por otro lado, en la **Fig. 38** se muestra el resultado que se ha obtenido una vez que se ha realizado la respectiva prueba. La descripción completa de la ejecución y resultados de las demás pruebas restantes se aprecian de mejor manera en el **ANEXO II** del presente documento.

```
it('Prueba inicio de sesión exitoso', () =>{
    fixture = TestBed.createComponent(AuthService);
    component = fixture.componentInstance;
    const credentials = {mail: "admin@gmail.com", password: "123abc"};
    const verification = component.login(credentials);
    expect(verification).toBeTruthy();
});
```

Fig. 37: Parte de código para el inicio de sesión.

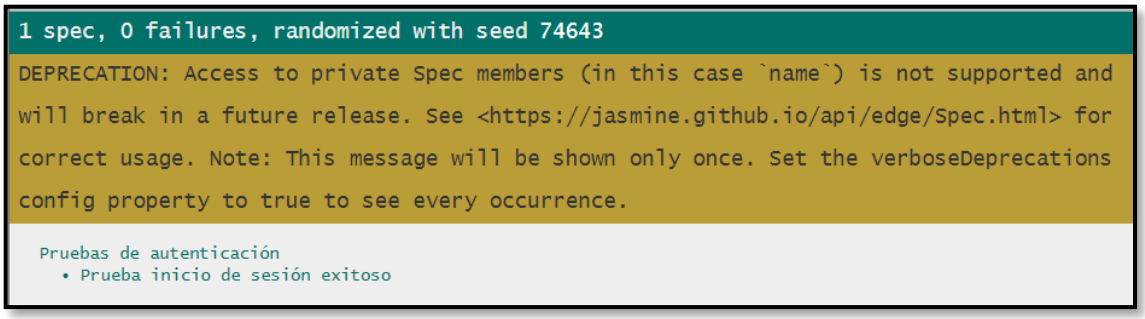

Fig. 38: Resultado de la prueba unitaria de inicio de sesión.

Después de la ejecución de esta prueba y en función de los resultados que se han obtenido, se establece que las funciones que se han implementado como parte del *backend* no presentan conflictos ya sea en ejecución o validación correspondiente y que están listos para que puedan ser consumidos e implementados en aplicaciones del lado del cliente o móviles si se requiere.

Ejecución de pruebas de rendimiento y resultados

La finalidad de este tipo de pruebas por una parte permite verificar el rendimiento de los métodos que se han implementado en el *backend* y por otra parte validan el nivel de estrés máximo que puede soportar un sistema software ante las respuestas de un número de peticiones concurrentes [36], en el caso de este proyecto enfocado en la parte del *backend*

las pruebas de carga han sido ejecutadas por la aplicación *web Pagespeed*, la misma que muestra el tiempo que tarda en cargar toda la página y a su vez muestra los elementos que retardan la carga completa. En ese sentido, la **Fig. 39** presenta los resultados que se han obtenido después aplicar esta prueba de carga en *Pagespeed*.

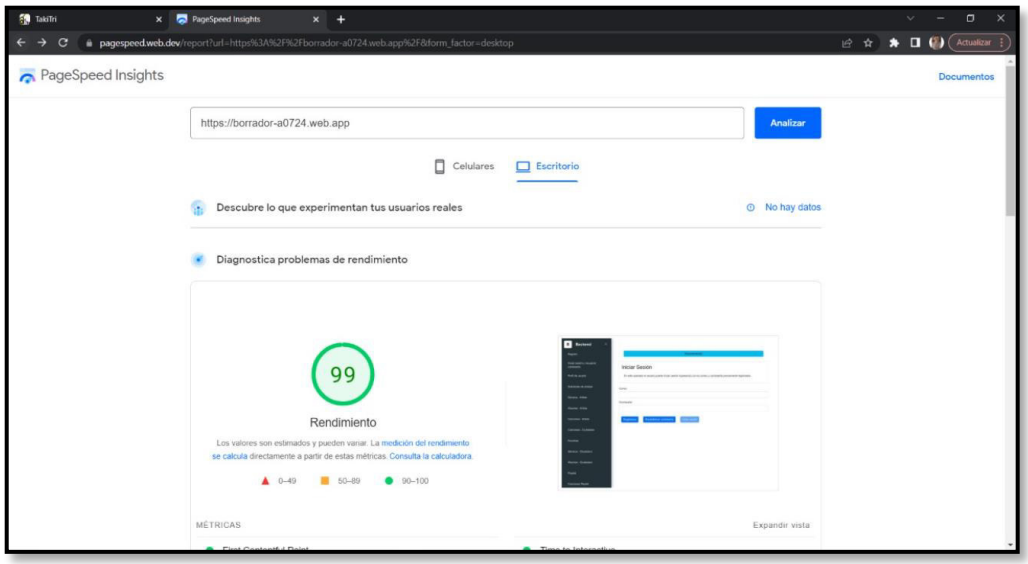

Fig. 39: Resultado de prueba de rendimiento en *PageSpeed.*

Después la ejecución de esta prueba y en función del resultado que se ha obtenido, se establece que el *backend* no presenta conflictos a nivel de tiempo de carga de los métodos que se ha implementado y que la información que se presente sea siempre en tiempo real.

Ejecución de pruebas de aceptación y resultados

El objetivo de estas pruebas es que el usuario pueda testear o probar el software a fin de que compruebe que los requerimientos que se han establecido al principio del proyecto se hayan desarrollado de forma correcta. Se debe tomar en cuenta que este tipo de pruebas se deben realizar previamente antes del despliegue a producción del sistema software [37]. En tal sentido la **TABLA VIII** muestra una prueba de aceptación y el resultado correspondiente, por otra parte, la descripción completa de las otras pruebas y resultados correspondientes se aprecian de mejor manera en el **ANEXO II** del presente documento.

TABLA VIII: Prueba de aceptación Nro. 1.

Prueba de Aceptación	
Identificador: PA001	**Identificador historia de Usuario:** HU001
Nombre: Iniciar sesión, cerrar sesión y cambiar contraseña.	
Descripción: El *backend* por medio del perfil administrador y artista permite generar varios métodos para: • Iniciar Sesión. • Cerrar Sesión. • Cambiar contraseña.	
Pasos de ejecución: • Dirigirse a ruta "*login*". • Ingresar las credenciales: correo y contraseña. • Dar clic en Iniciar sesión, si las credenciales son correctas se ingresa al sistema. • Dar clic en Cerrar sesión, lo que redirige a la ruta "*login*". • Dar clic en "Reestablecer contraseña" e ingresar el correo registrado. • Ingresar al enlace enviado al correo para reestablecer la contraseña. • Volver a la ruta "*login*" e ingresar las credenciales actualizadas.	
Resultado deseado: El *backend* le permite al usuario la implementación de métodos para iniciar sesión, cerrar sesión o cambiar contraseña mediante el uso de sus credenciales.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

Después la ejecución de las pruebas y en función de los resultados que se han obtenido, se establece que se han implementado todas las funcionalidades de forma correcta y ordenada, logrando así la aprobación del *Product Owner* y con ello pasar a la siguiente etapa.

3.6 *Sprint* 4. Despliegue del *backend*

En base a lo estipulado en el *Sprint Backlog* este *Sprint* contiene la siguiente tarea:

Despliegue del *backend* en *Firebase Hosting*

Una vez que se ha culminado el proceso de implementación de todos los métodos y las pruebas respectivas, se efectúa el despliegue del *backend* a producción por medio de *Firebase Hosting*, como se muestra en la **Fig. *40***. Sin embargo, si se requiere acceder de forma visual se puede acceder por medio del siguiente enlace:

https://borrador-a0724.web.app

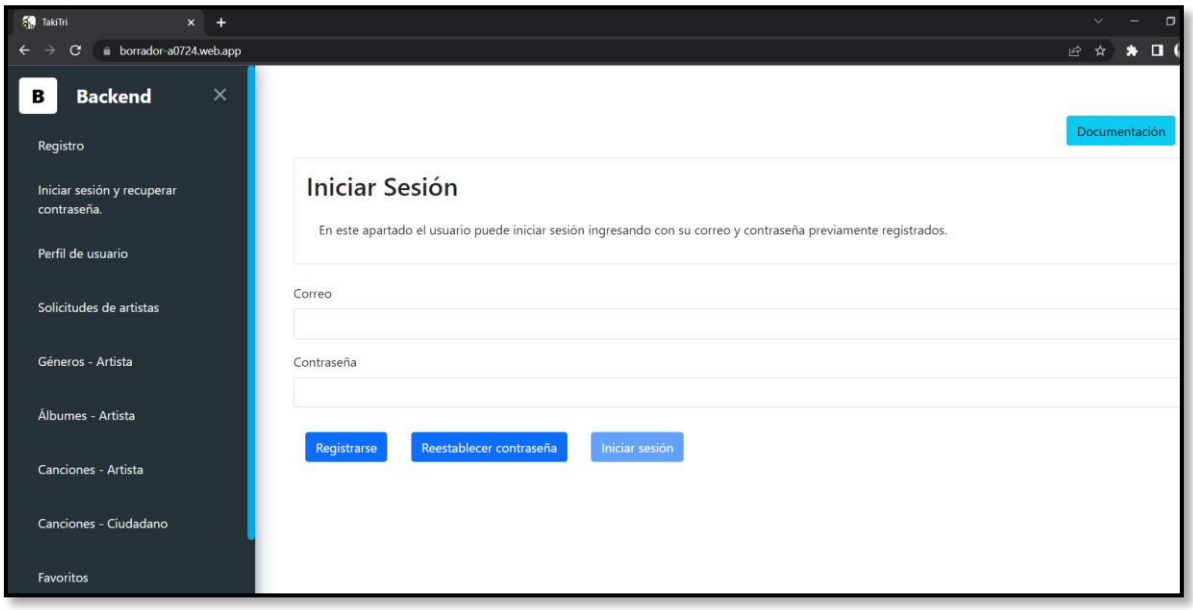

Fig. 40: *backend* que se ha desplegado en *Firebase Hosting.*

Por último, el detalle del despliegue se encuentra en el **ANEXO IV** del actual documento.

4 CONCLUSIONES

En este apartado se presentan las conclusiones que se han obtenido en el transcurso de este trabajo de integración curricular.

- Los requerimientos que se han planteado al inicio del proyecto han hecho que la implementación de cada uno de los métodos de *Firebase* como parte del *backend* se los hayan realizado de una forma organizada y modular, por lo cual si en un futuro se requiere se puede agregar más perfiles y métodos de manera sencilla.

- La integración de la metodología *Scrum* ha permitido que el proyecto se desarrolle eficientemente incluso con las observaciones y correcciones que se fueron presentado en cada *Sprint*, logrando de esta manera desarrollar un software de calidad y el cual este dentro de los límites de tiempo que se han establecido.

- El uso de la arquitectura Modelo-Vista-Controlador ha hecho posible que el *backend* se estructure de una forma correcta y organizada para que de esta manera cada uno de los métodos que se han implementado puedan ser reutilizados en futuros avances del proyecto e integración con más aplicaciones del lado del cliente o móviles.

- La implementación de una Base de datos *NoSQL* específicamente *Cloud Firestore*, ha facilitado que la gestión de la información sea más flexible, organizada y que el acceso a los datos sea más rápido debido al gran soporte que esta maneja y al uso de documentos y colecciones.

- Actualmente el *backend* se encuentra desplegado con una interfaz de fácil acceso e interacción dándole la posibilidad de que el mismo pueda ser empleado o usado como referencia por desarrolladores en proyectos que necesiten de la implementación de los métodos similares.

- Las pruebas que se han realizado en el *backend* como: unitarias, rendimiento y de aceptación garantizan la funcionalidad de cada uno de los métodos que se han implementado y a su vez la calidad del proyecto en general.

5 RECOMENDACIONES

En este apartado se presentan las recomendaciones que se han obtenido en el transcurso de este trabajo de integración curricular.

- Es importante verificar las versiones de Angular que sean estables o mantengan soporte, debido a que puede afectar de manera relevante al realizar el despliegue de cualquier sistema.

- Se recomienda que en caso de que aumente considerablemente los usuarios ciudadanos y artistas, se cambie el plan de *Firebase* para obtener mayor almacenamiento tanto en *Cloud Storage* como en *Cloud Firestore*.

- Se recomienda que en caso de aumentar nuevas funcionalidades independientemente del cliente revisar los servicios, en los cuales se encuentran métodos que pueden ser reutilizados.

- Se recomienda que en caso de aumentar nuevas funcionalidades estas sigan el formato implementado en este proyecto, es decir las nuevas funcionalidades vayan documentándose tanto en código como en las vistas de documentación.

6 BIBLIOGRAFÍA

[1] C. Mateos Casado, M. Pita Asan, M. Veléz Zambrano y Cedeño Mejía, «UN ANÁLISIS DE LA VIOLENCIA Y EL SEXISMO DESDE EL IMAGINARIO MUSICAL,» *Sociedad Española de Estudios de la Comunicación Iberoamericana,* pp. 232-233, 2015.

[2] J. C. GUZMÁN, «La música por internet le gana la batalla a los discos,» 15 Mayo 2016. [En línea]. Available: https://www.eltiempo.com/archivo/documento/CMS-16593353.

[3] L. G. y. Xavier, «A la industria musical todavía le falta ritmo,» 25 Noviembre 2013. [En línea]. Available: https://www.revistalideres.ec/lideres/industria-musical-todavia-le-falta.html.

[4] C. Villarrubia, «Spotify se sube a la nube de Google,» 15 Febrero 2016. [En línea]. Available: https://www.datacenterdynamics.com/es/noticias/spotify-se-sube-a-la-nube-de-google/.

[5] V. Giraldo, «¿Ya conoces Firebase? La herramienta de desarrollo y análisis de aplicaciones mobile,» 16 Abril 2019. [En línea]. Available: https://rockcontent.com/es/blog/que-es-firebase/.

[6] GCFGlobal, «¿Qué son las aplicaciones web?,» 2020. [En línea]. Available: https://edu.gcfglobal.org/es/informatica-basica/que-son-las-aplicaciones-web/1/.

[7] DIGITAL55, «Firebase: qué es, para qué sirve, funcionalidades y ventajas,» 15 Mayo 2020. [En línea]. Available: https://digital55.com/que-es-firebase-funcionalidades-ventajas-conclusiones/.

[8] HostGator, «Angular: qué es, para qué sirve y cuáles son sus ventajas,» 3 Mayo 2021. [En línea]. Available: https://www.hostgator.mx/blog/que-es-y-para-que-sirve-angular/.

[9] J. C. Rosero Villagómez, «Propuesta de una Estrategia de Marketing de un Portal Musical Recomendador de Canciones de Artistas Nacionales.,» Guayaquil, 2019.

[10] J. D. Campo, «Las mejores plataformas de música en streaming,» 1 Febrero 2019. [En línea]. Available: https://wildwildweb.es/es/blog/las-mejores-plataformas-de-musica-en-streaming.

[11] N. Chapaval, «Qué es Frontend y Backend: diferencias y características - Platzi,» 20 Mayo 2018. [En línea]. Available: https://platzi.com/blog/que-es-frontend-y-backend/.

[12] S. L. Mora, «Firebase: qué es, para qué sirve, funcionalidades y ventajas,» 17 Mayo 2020. [En línea]. Available: https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/.

[13] Firebase, «Firebase Documentation,» 12 Noviembre 2021. [En línea]. Available: https://firebase.google.com/docs/database?hl=es-419.

[14] Firebase, «Cloud Firestore,» 2021. [En línea]. Available: https://firebase.google.com/products/firestore?hl=es-419&gclid=CjwKCAjwve2TBhByEiwAaktM1H2ziCw9H1Fo-86sE_ryIPsU59YatDKjjA06niLcEx1AZUbSQdmSQBoCAtsQAvD_BwE&gclsrc=aw.ds.

[15] Firebase, «Cloud Storage for Firebase,» 2021. [En línea]. Available: https://firebase.google.com/products/storage?gclid=CjwKCAjwve2TBhByEiwAaktM1FAnhADg1HcNbI6uV8wREq3ozbt9r4iVNrYF2v-bfCDO8DvAKiyBcRoC9hUQAvD_BwE&gclsrc=aw.ds.

[16] Firebase, «Firebase Hosting,» 2021. [En línea]. Available: https://firebase.google.com/products/hosting?hl=es-419&gclid=CjwKCAjwj42UBhAAEiwACIhADtwirwtGMo1w4XgSAD1mpPbZQrocXE2Fn7gbyQ19fDqhNwGHmIyNNhoCfpMQAvD_BwE&gclsrc=aw.ds.

[17] desarrolloweb.com, «Angular,» 21 Abril 2020. [En línea]. Available: https://desarrolloweb.com/home/angular.

[18] QUALITY DEVS, «QUALITY DEVS,» 16 Septiembre 2019. [En línea]. Available: https://www.qualitydevs.com/2019/09/16/que-es-angular-y-para-que-sirve/.

[19] Wikipedia, «Microsoft Visual Studio,» 10 Mayo 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Microsoft_Visual_Studio.

[20] D. D. Luca, «Visual Studio Code: características principales,» 2020. [En línea]. Available: https://www.ecured.cu/Visual_Studio_Code.

[21] QuestionPro, «¿Qué es un estudio de caso y cómo realizarlo?,» 2022. [En línea]. Available: https://www.questionpro.com/blog/es/que-es-un-estudio-de-caso/.

[22] Santander Universidades, «Metodologías de desarrollo de software,» 21 Diciembre 2020. [En línea]. Available: https://www.becas-santander.com/es/blog/metodologias-desarrollo-software.html.

[23] D. d. Silva, «¿Qué es la metodología ágil? ¿Para qué sirve?,» 19 Febrero 2021. [En línea]. Available: https://www.zendesk.com.mx/blog/metodologia-agil-que-es/.

[24] A. Flores, «Crehana,» 22 Octubre 2021. [En línea]. Available: https://www.crehana.com/ec/blog/diseno-productos/agile-vs-scrum/?utm_source=google&utm_medium=cpc&utm_campaign=search-blog&utm_content=blog-product&utm_term=upper-funnel&gclid=Cj0KCQjw29CRBhCUARIsAOboZbIAmUzXxF8Gi9G1_4ef6S0_JDMnZHMh1M9zsXj1nSP97w1P_4LVEo8.

[25] SOFTENG, «The internet development company,» 2021. [En línea]. Available: https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum/proceso-roles-de-scrum.html.

[26] Viewnext, «Artefactos Scrum ¿Qué son y para qué sirven?,» 27 Noviembre 2019. [En línea]. Available: https://www.viewnext.com/artefactos-scrum/.

[27] O. García, «Recopilación de requisitos,» 1 Mayo 2013. [En línea]. Available: https://www.proyectum.com/sistema/blog/recopilacion-de-requisitos/.

[28] M. Rehkopf, «Historias de usuario con ejemplos y plantilla,» 2022. [En línea]. Available: https://www.atlassian.com/es/agile/project-management/user-stories.

[29] EALDE, «En qué consiste el Product Backlog y el Sprint Backlog,» 27 Agosto 2019. [En línea]. Available: https://www.ealde.es/product-backlog-sprint-backlog/.

[30] Voigtmann, «Arquitectura de software,» 2022. [En línea]. Available: https://www.voigtmann.de/es/desarrollo-de-software/arquitectura-de-software/.

[31] M. García, «MVC (Modelo-Vista-Controlador): ¿qué es y para qué sirve?,» 5 Octubre 2017. [En línea]. Available: https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve.

[32] OKHOSTING, «Herramientas de Desarrollo de Software,» 2021. [En línea]. Available: https://okhosting.com/blog/herramientas-de-desarrollo-de-software/.

[33] A. Vats, «La biblioteca Angular oficial para Firebase.,» 12 Abril 2022. [En línea]. Available: https://github.com/angular/angularfire.

[34] Wikipedia, «Bootstrap (framework),» 14 Mayo 2022. [En línea]. Available: https://es.wikipedia.org/wiki/Bootstrap_(framework).

[35] Admin, «Cómo usar Testing en Angular con Jasmine y Karma,» 20 Abril 2020. [En línea]. Available: https://digital55.com/como-usar-testing-angular-jasmine-karma/#:~:text=%C2%BFQu%C3%A9%20es%20Karma%3F,suites%20de%20testing%2C%20como%20Jasmine..

[36] loadview-testing.com, «Pruebas de carga,» 2022. [En línea]. Available: https://www.loadview-testing.com/es/pruebas-de-carga/.

[37] Digite, «Pruebas de aceptación: el qué y el por qué,» 2022. [En línea]. Available: https://www.digite.com/es/agile/pruebas-de-aceptacion/.

[38] L. Carvajal, Metodología de la Investgación Científica. Curso general y aplicado, 28 ed., Santiago de Cali: U.S.C., 2006, p. 139.

[39] Firebase, «Firebase Realtime Database,» [En línea]. Available: https://firebase.google.com/products/realtime-database?gclid=Cj0KCQiA9OiPBhCOARIsAI0y71DyooA18A5DlgJw2SQF7TTvz6tk3oUkGpFLrIakGAOCdiuVg23e8OoaAm_HEALw_wcB&gclsrc=aw.ds.

[40] Mis Bandas Nacionales Ecuador, «Quienes Somos,» 2021. [En línea]. Available: https://mbnecuador.com/quienes-somos-3/.

[41] R. Arjonilla, «BackEnd,» 2016. [En línea]. Available: https://rafarjonilla.com/que-es/backend/.

[42] T. G. Manuel, «Metodología Scrum,» de *Desarrollo detallado de la fase de aprovación de un proyecto informático mediante el uso de metodologías ágiles.*, Quito, p. 56.

[43] EcuRed, «EcuRed,» [En línea]. Available: https://www.ecured.cu/PowerDesigner.

[44] H. México, «Angular: qué es, para qué sirve y cuáles son sus ventajas,» Mayo 2021. [En línea].

[45] A. d. l. Puente, «Pruebas de compatibilidad,» 2018. [En línea]. Available: Pruebas de compatibilidad.

7 ANEXOS

A continuación, se presentan los ANEXOS que se han implementado para el desarrollo del *backend*, los cuales se encuentran divididos de la siguiente manera:

- **ANEXO I.** Resultado del programa antiplagio *Turnitin*.
- **ANEXO II.** Manual Técnico.
- **ANEXO III.** Manual de Usuario.
- **ANEXO IV.** Manual de Instalación.

ANEXO I

A continuación, se presenta el certificado que el Director de Tesis ha emitido y en donde se evidencia el resultado que se ha obtenido en la herramienta antiplagio *Turnitin*.

ESCUELA POLITÉCNICA NACIONAL
ESCUELA DE FORMACIÓN DE TECNÓLOGOS
CAMPUS POLITÉCNICO "ING. JOSÉ RUBÉN ORELLANA"

CERTIFICADO DE ORIGINALIDAD

Quito, D.M. 22 de agosto de 2022

De mi consideración:

Yo, Loarte Cajamarca Byron Gustavo, en calidad de Director del Trabajo de Integración Curricular titulado Desarrollo de un backend asociado al DESARROLLO DE SISTEMA WEB Y APLICACIÓN MÓVIL INFORMATIVA SOBRE MÚSICA NACIONAL ECUATORIANA elaborado por el estudiante Edwin Iván Fraga Tana de la carrera en Tecnología Superior en Desarrollo de Software, certifico que he empleado la herramienta Turnitin para la revisión de originalidad del documento escrito secciones: Descripción del componente desarrollado, Metodología, Resultados, Conclusiones y Recomendaciones, producto del Trabajo de Integración Curricular indicado.

El documento escrito tiene un índice de similitud del 11%.

Es todo cuanto puedo certificar en honor a la verdad, pudiendo el interesado hacer uso del presente documento para los trámites de titulación.

NOTA: Se adjunta el informe generado por la herramienta Turnitin.

Atentamente,

Loarte Cajamarca Byron Gustavo
Profesor Ocasional a Tiempo Completo
Escuela de Formación de Tecnólogos

ANEXO II

Recopilación de Requerimientos

La **TABLA IX** presenta la Recopilación de requerimientos que se han obtenido al inicio del proyecto de acuerdo con lo solicitado por el *Product Owner.*

TABLA IX: Recopilación de requerimientos.

RECOPILACIÓN DE REQUERIMIENTOS		
TIPO DEL SISTEMA	**ID - RR**	**ENUNCIADO DEL ÍTEM**
BACKEND	**RR003**	Como usuario administrador necesita implementar varios métodos para: ● Gestionar solicitudes de artistas.
	RR004	Como usuario invitado necesita implementar un método para: ● Registrarse a través de un formulario.
	RR005	Como usuario artista necesita implementar varios métodos para: ● Modificar perfil de usuario.
	RR006	Como usuario artista necesita implementar varios métodos para: ● Gestionar géneros musicales.
	RR007	Como usuario artista necesita implementar varios métodos para: ● Gestionar álbumes.
	RR008	Como usuario artista necesita implementar varios métodos para: ● Gestionar canciones.

	RR009	Como usuario invitado necesita implementar un método para que en la aplicación móvil pueda: ● Registrarse a través de un formulario.
	RR010	Como usuario ciudadano necesita implementar varios métodos para: ● Iniciar sesión. ● Cerrar sesión. ● Recuperar contraseña.
	RR011	Como usuario ciudadano necesita implementar varios métodos para: ● Modificar perfil de usuario.
	RR012	Como usuario ciudadano necesita implementar un método para: ● Visualizar canciones.
	RR013	Como usuario ciudadano necesita implementar varios métodos para: ● Gestionar favoritos.
	RR014	Como usuario ciudadano necesita implementar un método para: ● Visualizar géneros y álbumes musicales.
	RR015	Como usuario ciudadano necesita implementar varios métodos para: ● Gestionar *playlist*.
	RR016	Como usuario ciudadano necesita implementar varios métodos para: ● Gestionar canciones al *playlist*.

Historias de Usuario

Posterior a la Recopilación de requerimientos se prosigue el desarrollo de Historias de usuario para el *backend*. En ese sentido, se presentan las 16 Historias de usuario escritas en base a los requerimientos del proyecto que van desde la **TABLA X** a la **TABLA XXIV**

TABLA X: Historia de usuario Nro.2.

HISTORIA DE USUARIO	
Identificador (ID): HU002	**Usuario**: Administrador
Nombre Historia: Modificar perfil de usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 1	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil administrador permite implementar varios métodos para visualizar y editar su perfil por medio de los siguientes campos: • Nombre. • Imagen.	
Observación: El usuario administrador es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para modificar su perfil.	

TABLA XI: Historia de usuario Nro.3.

HISTORIA DE USUARIO	
Identificador (ID): HU003	**Usuario**: Administrador
Nombre Historia: Gestionar las solicitudes de artistas.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 1	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil administrador permite implementar varios métodos para: • Visualizar solicitudes de artistas. • Aprobar solicitudes de artistas. • Rechazar solicitudes de artistas.	

Observación: El usuario administrador es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para poder gestionar las solicitudes de artistas.

TABLA XII: Historia de usuario Nro.4.

HISTORIA DE USUARIO	
Identificador (ID): HU004	**Usuario**: Invitado
Nombre Historia: Registrarse a través de un formulario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Iteración Asignada: 1	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil invitado permite implementar un método para poder registrarse, esto se lo realiza por medio de un formulario llenado los siguientes campos: • E-mail. • Contraseña. • Nombre. • Fecha de nacimiento. • Rol	
Observación: El usuario invitado tras registrarse debe esperar hasta que el usuario administrador lo apruebe, consecuentemente el usuario invitado se convierte en usuario artista.	

TABLA XIII: Historia de usuario Nro.5.

HISTORIA DE USUARIO	
Identificador (ID): HU005	**Usuario**: Artista
Nombre Historia: Modificar el perfil de usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 2	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para visualizar y editar su perfil por medio de los siguientes campos:	

• Nombre. • Imagen.
Observación: El usuario artista es el único que puede acceder los *métodos* mencionados anteriormente, es decir necesita iniciar sesión para poder modificar su perfil.

TABLA XIV: Historia de usuario Nro.6.

HISTORIA DE USUARIO	
Identificador (ID): HU006	**Usuario**: Artista
Nombre Historia: Gestionar géneros musicales.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 2	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para: • Crear géneros musicales. • Visualizar sus géneros musicales. • Seleccionar sus géneros musicales.	
Observación: El usuario artista es el único que puede acceder los métodos mencionados anteriormente, es decir necesita iniciar sesión para poder gestionar géneros musicales.	

TABLA XV: Historia de usuario Nro.7.

HISTORIA DE USUARIO	
Identificador (ID): HU007	**Usuario**: Artista
Nombre Historia: Gestionar álbumes.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta
Iteración Asignada: 2	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil asignado permite implementar varios métodos para: • Crear álbum.	

• Visualizar sus álbumes. • Seleccionar sus álbumes.
Observación: El usuario artista es el único que puede acceder los métodos mencionados anteriormente, es decir necesita iniciar sesión para poder gestionar canciones. Para crear un álbum el artista debe seleccionar o crear un género musical previamente.

TABLA XVI: Historia de usuario Nro.8.

HISTORIA DE USUARIO	
Identificador (ID): HU008	**Usuario**: Artista
Nombre Historia: Gestionar canciones.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 2	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para: • Subir canciones a un álbum. • Visualizar canciones. • Actualizar canciones. • Eliminar canciones.	
Observación: El usuario artista es el único que puede acceder los métodos mencionados anteriormente, es decir necesita iniciar sesión para poder gestionar canciones. Para subir una canción es obligatorio que pertenezca a un álbum.	

TABLA XVII: Historia de usuario Nro.9.

HISTORIA DE USUARIO	
Identificador (ID): HU009	**Usuario**: Invitado
Nombre Historia: Registrarse a través de un formulario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil invitado permite implementar un método	

para poder registrarse, esto se lo realiza por medio de un formulario llenando los siguientes campos: • E-mail. • Contraseña. • Nombre. • Fecha de nacimiento. • Rol
Observación: El usuario invitado tras registrarse se convierte en usuario ciudadano, el usuario invitado también puede registrarse con su cuenta de Google.

TABLA XVIII: Historia de usuario Nro.10.

HISTORIA DE USUARIO	
Identificador (ID): HU010	**Usuario**: Ciudadano
Nombre Historia: Iniciar sesión, cerrar sesión y cambiar contraseña.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Iniciar sesión. • Cerrar sesión. • Recuperar contraseña.	
Observación: Para iniciar sesión el usuario ciudadano debe estar previamente registrado. Al recuperar la contraseña se envía un correo al e-mail del usuario ciudadano.	

TABLA XIX: Historia de usuario Nro.11.

HISTORIA DE USUARIO	
Identificador (ID): HU011	**Usuario**: Ciudadano
Nombre Historia: Modificar perfil de usuario.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	

Responsable (es): Iván Fraga
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para visualizar y editar su perfil a través de los siguientes campos: • Nombre. • Imagen.
Observación: El usuario ciudadano es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para gestionar su perfil.

TABLA XX: Historia de usuario Nro.12.

HISTORIA DE USUARIO	
Identificador (ID): HU012	**Usuario**: Ciudadano
Nombre Historia: Visualizar canciones.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil ciudadano permite implementar un método para visualizar las canciones ingresadas por los usuarios artistas. Además, las canciones presentan la siguiente información: • Título de la canción. • Género de la canción. • Artista de la canción. • Álbum de la canción. • Año del álbum • Canción.	
Observación: El usuario ciudadano es el único que puede acceder al método mencionado anteriormente, es decir necesita iniciar sesión para visualizar canciones.	

TABLA XXI: Historia de usuario Nro.13.

HISTORIA DE USUARIO	
Identificador (ID): HU013	**Usuario**: Ciudadano
Nombre Historia: Gestionar favoritos.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Alta

Iteración Asignada: 3
Responsable (es): Iván Fraga
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Agregar canciones a la lista de favoritos. • Visualizar canciones de la lista de favoritos.
Observación: El usuario ciudadano es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para gestionar la lista de favoritos.

TABLA XXII: Historia de usuario Nro.14.

HISTORIA DE USUARIO	
Identificador (ID): HU014	**Usuario**: Ciudadano
Nombre Historia: Visualizar géneros y álbumes musicales.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 3	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Visualizar géneros musicales. • Visualizar álbumes. En cada género musical se puede visualizar las canciones pertenecientes al mismo.	
Observación: El usuario ciudadano es el único que puede acceder al método mencionado anteriormente, es decir necesita iniciar sesión para visualizar géneros y álbumes musicales.	

TABLA XXIII: Historia de usuario Nro.15.

HISTORIA DE USUARIO	
Identificador (ID): HU015	**Usuario**: Ciudadano
Nombre Historia: Gestionar *playlist*.	
Prioridad en negocio: Media	**Riesgo en desarrollo:** Media
Iteración Asignada: 3	

Responsable (es): Iván Fraga
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Crear *playlist.* • Visualizar *playlist.*
Observación: El usuario ciudadano es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para gestionar la *playlist*.

TABLA XXIV: Historia de usuario Nro.16.

HISTORIA DE USUARIO	
Identificador (ID): HU016	**Usuario**: Ciudadano
Nombre Historia: Gestionar canciones al *playlist*.	
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Iteración Asignada: 3	
Responsable (es): Iván Fraga	
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Agregar canciones al *playlist.* • Visualizar canciones del *playlist.* • Eliminar canciones del *playlist.*	
Observación: El usuario ciudadano es el único que puede acceder a los métodos mencionados anteriormente, es decir necesita iniciar sesión para gestionar canciones al *playlist*.	

Product Backlog

La **TABLA *XXV*** presenta la prioridad de cada requisito que se ha implementado en el *backend*. Los requisitos a continuación se clasifican en base a las necesidades del *Product Owner.*

TABLA XXV: *Product Backlog.*

ELABORACIÓN DEL *PRODUCT BACKLOG*				
ID–HU	**HISTORIA DE USUARIO**	**ITERACIÓN**	**ESTADO**	**PRIORIDAD**
HU003	Gestionar las solicitudes de artistas.	1	Finalizada	Alta
HU004	Registrarse a través de un formulario.	2	Finalizada	Media
HU005	Modificar el perfil de usuario.	2	Finalizada	Media
HU006	Gestionar géneros musicales.	2	Finalizada	Media
HU007	Gestionar álbum.	2	Finalizada	Media
HU008	Gestionar canciones.	2	Finalizada	Alta
HU009	Registrarse a través de un formulario.	3	Finalizada	Media
HU010	Iniciar sesión, cerrar sesión y cambiar contraseña.	3	Finalizada	Media
HU011	Modificar perfil de usuario.	3	Finalizada	Media
HU012	Visualizar canciones.	3	Finalizada	Alta
HU013	Gestionar favoritos.	3	Finalizada	Media
HU014	Visualizar géneros y álbumes musicales.	3	Finalizada	Alta
HU015	Gestionar *playlist*.	3	Finalizada	Media
HU016	Gestionar canciones al *playlist*.	3	Finalizada	Alta

Sprint Backlog

La **TABLA *XXVI*** presenta los *Sprints* que se han desarrollado por parte del *backend*, describiendo las actividades y el tiempo para cumplirlos respectivamente establecidos por el *Product Owner*.

TABLA XXVI: *Sprint Backlog.*

ELABORACIÓN DEL *SPRINT BACKLOG*						
ID – SB	**NOMBRE**	**MÓDULO**	**ID-HU**	**HISTORIAS DE USUARIO**	**TAREAS**	**TIEMPO ESTIMADO**
SB001	Diseño e implementación de métodos para el usuario administrador.	Inicio de sesión	HU002	Iniciar sesión, cerrar sesión y cambiar contraseña.	• Diseño e implementación del método para el inicio de sesión, cierre de sesión y cambio de contraseña. • Consulta a la base de datos y autorización. • Validación de los datos requeridos.	40H
		Módulo perfil.	HU003	Modificar perfil de usuario.	• Diseño e implementación de métodos para visualizar y editar el perfil de usuario. • Consulta en la base de datos. • Validación de los datos requeridos.	

		Gestión de solicitudes.	HU004	Gestionar las solicitudes de artistas.	• Diseño e implementación de métodos para visualizar, aceptar y rechazar las solicitudes de artistas. • Consulta en la base de datos.	
SB002	Diseño e implementación de métodos para el usuario artista.	Registro de artista	HU005	Registrarse a través de un formulario.	• Diseño e implementación de métodos para registrar de usuario artista a través de un formulario. • Validación de los datos requeridos. • Verificación que el registro sea único.	50H
		Módulo perfil	HU006	Modificar el perfil de usuario.	• Diseño e implementación de métodos para visualizar y editar el perfil de usuario. • Consulta en la base de datos. • Validación de los datos requeridos.	
		Módulo géneros musicales	HU0007	Gestionar géneros musicales.	• Diseño e implementación de métodos para crear, visualizar y seleccionar sus géneros musicales. • Consulta en la base de datos. • Validación de los datos requeridos.	

		Módulo álbum	HU0008	Gestionar álbum.	• Diseño e implementación de métodos para crear, visualizar y seleccionar sus álbumes. • Consulta en la base de datos. • Validación de los datos requeridos.	
		Módulo canciones	HU0009	Gestionar canciones.	• Diseño e implementación de métodos para subir, visualizar, actualizar y eliminar canciones a un álbum. • Consulta en la base de datos. • Validación de los datos requeridos.	
SB003	Diseño e implementación de métodos para el usuario ciudadano.	Registro de ciudadano	HU0010	Registrarse a través de un formulario.	• Diseño e implementación del método para registrar de usuario ciudadano a través de un formulario. • Validación de los datos requeridos. • Verificación que el registro sea único.	70H

		Inicio de sesión	HU0011	Iniciar sesión, cerrar sesión y cambiar contraseña.	• Diseño e implementación del método para el inicio de sesión, cierre de sesión y cambiar contraseña. • Consulta a la base de datos y autorización. • Validación de los datos requeridos.	
		Módulo perfil	HU0012	Modificar perfil de usuario.	• Diseño e implementación de métodos para visualizar y editar el perfil de usuario. • Consulta en la base de datos. • Validación de los datos requeridos.	
		Módulo canciones	HU0013	Visualizar canciones.	• Diseño e implementación de un método para la visualización de canciones y sus datos relacionados. • Consulta a la base de datos.	
		Módulo de favoritos	HU0014	Gestionar favoritos.	• Diseño e implementación de métodos para agregar y visualizar canciones de la lista de favoritos. • Consulta a la base de datos.	

		Módulo de géneros musicales	HU0015	Visualizar géneros y álbumes musicales.	• Diseño e implementación de métodos para visualizar información y canciones pertenecientes a géneros y álbumes musicales. • Consulta a la base de datos.	
		Módulo *playlist*	HU0016	Gestionar *playlist*.	• Diseño e implementación de métodos para crear y visualizar *playlist*. • Consulta en la base de datos. • Validación de los datos requeridos.	
		Módulo de canciones al *playlist*.	HU0017	Gestionar canciones al *playlist*.	• Diseño e implementación de métodos para agregar, visualizar y eliminar canciones al *playlist*. • Consulta en la base de datos. • Validación de los datos requeridos.	
SB004	Pruebas del *backend*	• Pruebas unitarias. • Prueba de rendimiento. • Prueba de aceptación.				20H
SB005	Despliegue del *backend*	• Despliegue del *backend* en *Firebase Hosting.*				10H

Documentación	• Trabajo de Integración Curricular. • Anexos.	30H
TOTAL		**240 H**

Diseño de la Base de datos NoSQL

La **Fig. 41** presenta las seis colecciones necesarias para el desarrollo de cada uno métodos implementados del *backend*, con ello se estandariza la información que se maneja y facilita que las consultas sean eficientes.

Fig. 41: Diseño de la Base de datos no relacional.

Pruebas

Una vez que se ha finalizado la etapa de codificación se ha implementado la ejecución de pruebas unitarias, carga y de aceptación para comprobar la calidad del *backend* y cada uno de los módulos respectivamente.

Pruebas unitarias

A continuación, en las figuras que van desde la **Fig. 42** hasta la **Fig. *57*** se muestran las porciones de código en las que se han ejecutado las pruebas unitarias y sus correspondientes resultados que se han obtenido.

```
it('Prueba de reestablecer contraseña exitoso', () =>{
    fixture = TestBed.createComponent(AuthService);
    component = fixture.componentInstance;
    const mail = "correoprueba@gmail.com";
    const verification = component.forgotPassword(mail);
    expect(verification).toBeTruthy();
});
```

Fig. 42: Parte de código para cambiar de contraseña.

```
1 spec, 0 failures, randomized with seed 45795
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas de autenticación
  • Prueba de reestablecer contraseña exitoso
```

Fig. 43: Resultado de la prueba unitaria cambiar de contraseña.

```
it('Prueba de visualizar perfil de usuario', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const collection = "users";
    const id = "L61fM2MDn5ekTLWHuWBdqDfGmKD3";
    const verification = component.getObject(collection, id)
    expect(verification).toBeTruthy();
});
```

Fig. 44: Parte de código para visualizar el perfil de usuario.

```
1 spec, 0 failures, randomized with seed 87213
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas songservice
  • Prueba de visualizar perfil de usuario
```

Fig. 45: Resultado de la prueba unitaria de visualizar el perfil de usuario.

```
it('Prueba de visualizar canciones', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const collection = "songs";
    const verification = component.getAlbumSongs(collection);
    expect(verification).toBeTruthy();
});
```

Fig. 46: Parte de código para visualizar de canciones.

```
1 spec, 0 failures, randomized with seed 60901
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas songservice
  • Prueba de visualizar canciones
```

Fig. 47: Resultado de la prueba unitaria de visualizar de canciones.

```
it('Prueba de visualizar géneros', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const collection = "genres";
    const verification = component.getList(collection);
    expect(verification).toBeTruthy();
});
```

Fig. 48: Parte de código para visualizar géneros musicales.

```
1 spec, 0 failures, randomized with seed 47614
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas songservice
  • Prueba de visualizar géneros
```

Fig. 49: Resultado de la prueba unitaria de visualizar géneros musicales.

```
it('Prueba de visualizar albumes', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const collection = "albums";
    const verification = component.getList(collection);
    expect(verification).toBeTruthy();
});
```

Fig. 50: Parte de código para visualizar álbumes.

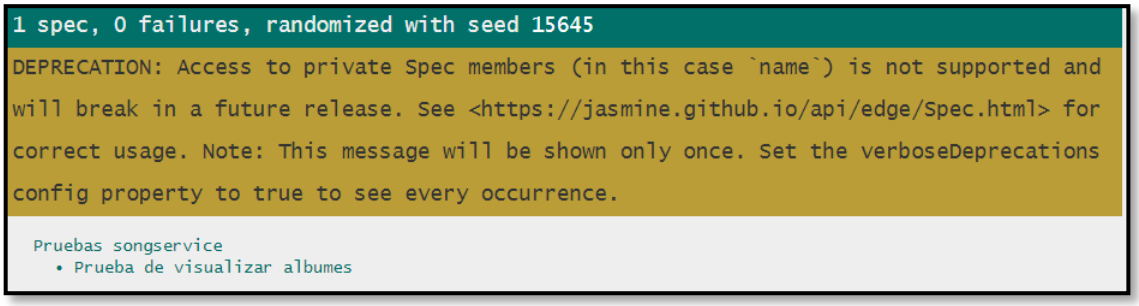

Fig. 51: Resultado de la prueba unitaria de visualizar álbumes.

```
it('Prueba de subir canciones', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const songURL =
    "https://firebasestorage.googleapis.com/v0/b/borrador-a0724.appspot.co
    const path = "songs/pasillo";
    let song: Song = {
        genre_name: "pasillo",
        album_name: "nuevo album",
        imageURL:
        "https://firebasestorage.googleapis.com/v0/b/borrador-a0724.appspo
        song_name: "new song",
        year: 2022,
        author: "Julio Jaramillo",
        songURL: "",
        song_reference: "",
        id:"",
        authorId: ""
    }
    const verification = component.songCreate(song, songURL,path);
    expect(verification).toBeTruthy();
});
```

Fig. 52: Parte de código para subir canciones.

```
1 spec, 0 failures, randomized with seed 16413
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas songservice
  • Prueba de subir canciones
```

Fig. 53: Resultado de la prueba unitaria de subir canciones.

```
it('Prueba de actualizar canciones', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    const songURL =
    "https://firebasestorage.googleapis.com/v0/b/borrador-a0724.appspot.com/o
    const path = "songs/pasillo";
    const id = "L61fM2MDn5ekTLWHuWBdqDfGmKD3"
    let song: Song = {
        genre_name: "pasillo",
        album_name: "nuevo album",
        imageURL:
        "https://firebasestorage.googleapis.com/v0/b/borrador-a0724.appspot.c
        song_name: "new song",
        year: 2022,
        author: "Julio Jaramillo",
        songURL: "",
        song_reference: "",
        id:"",
        authorId: ""
    }
    const verification = component.updateSong(song, songURL, path, id);
    expect(verification).toBeTruthy();
});
```

Fig. 54: Parte de código para editar una canción.

```
1 spec, 0 failures, randomized with seed 54914
DEPRECATION: Access to private Spec members (in this case `name`) is not supported and
will break in a future release. See <https://jasmine.github.io/api/edge/Spec.html> for
correct usage. Note: This message will be shown only once. Set the verboseDeprecations
config property to true to see every occurrence.

Pruebas songservice
  • Prueba de actualizar canciones
```

Fig. 55: Resultado de la prueba unitaria de editar una canción.

```
it('Prueba de eliminar canciones', () =>{
    fixture = TestBed.createComponent(SongService);
    component = fixture.componentInstance;
    let song: Song = {
        genre_name: "pasillo",
        album_name: "nuevo album",
        imageURL:
        "https://firebasestorage.googleapis.com/v0/b/borrador-a0724.appspot.c
        song_name: "new song",
        year: 2022,
        author: "Julio Jaramillo",
        songURL: "",
        song_reference: "",
        id:"",
        authorId: ""
    }
    const verification = component.deleteSong(song);
    expect(verification).toBeTruthy();
});
```

Fig. 56: Parte de código para eliminar una canción.

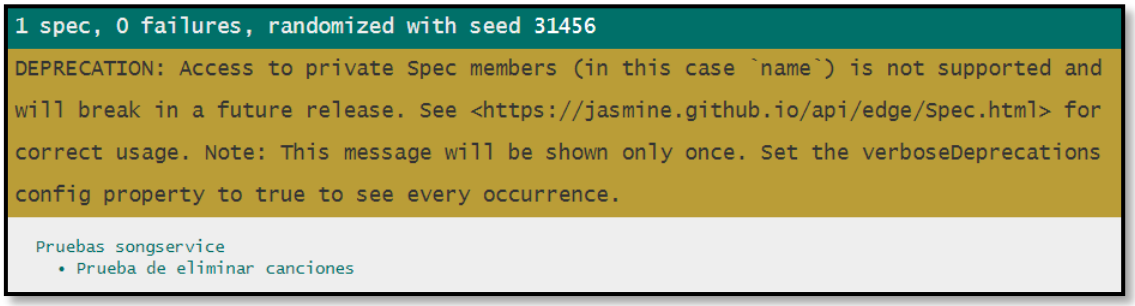

Fig. 57: Resultado de la prueba unitaria de eliminar una canción.

Pruebas de aceptación

A continuación, se muestran las 16 pruebas de aceptación que van desde la **TABLA XXVII** hasta la **TABLA XLI**. Adicionalmente, cada prueba describe el proceso asignado a los tipos de usuario para la ejecución, comprobación y aceptación correspondiente.

TABLA XXVII: Prueba de aceptación 2- Modificar perfil de usuario.

	Prueba de Aceptación
Identificador: PA002	**Identificador historia de Usuario:** HU002
Nombre: Modificar perfil de usuario.	

Descripción: El *backend* por medio del perfil administrador permite implementar varios métodos para visualizar y editar su perfil por medio de los siguientes campos: • Nombre. • Imagen.
Pasos de ejecución: • Autenticarse en el Sistema Web del *backend* • Visualización del perfil de usuario • Clic en “Editar Perfil” • Se puede actualizar el nombre y o la foto de perfil. • Clic en “Guardar”, redirige a la visualización del perfil de usuario con los campos actualizados.
Resultado deseado: El *backend* le permite al usuario administrador la implementación de métodos para visualizar y editar su perfil.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXVIII: Prueba de aceptación 3 - Gestionar las solicitudes de artistas.

	Prueba de Aceptación
Identificador: PA003	**Identificador historia de Usuario:** HU003
Nombre: Gestionar las solicitudes de artistas.	
Descripción: El *backend* por medio del perfil administrador permite implementar varios métodos para: • Visualizar solicitudes de artistas. • Aprobar solicitudes de artistas. • Rechazar solicitudes de artistas.	
Pasos de ejecución:	

• Autenticarse en el Sistema *Web* del *backend*. • Clic en “Gestionar artistas”. • El administrador puede aprobar la solicitud de los artistas registrados dando clic en “Aprobar solicitud”. • Se actualiza el rol de “*no artist*” a “*artist*” del usuario seleccionado.
Resultado deseado: El *backend* le permite al usuario administrador la implementación de métodos para visualizar y aprobar solicitudes de artistas. En caso de que no se aprueben el usuario es considerado rechazado.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXIX: Prueba de aceptación 4 - Registrarse a través de un formulario.

	Prueba de Aceptación
Identificador: PA004	**Identificador historia de Usuario:** HU004
Nombre: Registrarse a través de un formulario.	
Descripción: El *backend* por medio del perfil invitado permite implementar un método para poder registrarse, esto se lo realiza por medio de un formulario llenado los siguientes campos: • E-mail. • Contraseña. • Nombre. • Fecha de nacimiento. • Rol	
Pasos de ejecución: • Dirigirse a ruta “*register*” del Sistema *Web* del *backend*. • Ingresar los campos requeridos. • En caso de que el rol seleccionado sea artista, este deberá esperar hasta la aceptación por parte del usuario administrador para acceder a las funcionalidades propias de un usuario artista	

Resultado deseado: El *backend* le permite al usuario invitado la implementación de métodos para registrarse como usuario artista.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXX: Prueba de aceptación 5 - Modificar el perfil de usuario.

	Prueba de Aceptación
Identificador: PA005	**Identificador historia de Usuario:** HU005
Nombre: Modificar perfil de usuario.	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para visualizar y editar su perfil por medio de los siguientes campos: • Nombre. • Imagen.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en “Editar Perfil” • Se puede actualizar el nombre y o la foto de perfil. • Clic en “Guardar”, redirige a la visualización del perfil de usuario con los campos actualizados.	
Resultado deseado: El *backend* le permite al usuario artista la implementación de métodos para visualizar y editar su perfil.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXXI: Prueba de aceptación 6 - Gestionar géneros musicales.

Prueba de Aceptación	
Identificador: PA006	**Identificador historia de Usuario:** HU006
Nombre: Gestionar géneros musicales.	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para: • Crear géneros musicales. • Visualizar sus géneros musicales. • Seleccionar sus géneros musicales.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Clic en "gestionar géneros" y "visualizar géneros", con ello se puede visualizar los géneros del usuario • En esa vista se puede, o crear un género nuevo o seleccionar uno ya existente. • En caso de que se quiera crear un género este debe completar los campos de nombre e imagen del nuevo género. • En caso de que se quiera seleccionar un género, se debe seleccionar uno de los presentes en dicha vista. • Crear o seleccionar un género se lo realiza con el fin de realizar el proceso de subir canciones.	
Resultado deseado: El *backend* le permite al usuario artista la implementación de métodos para visualizar crear o seleccionar géneros musicales como parte del proceso para subir canciones.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXXII: Prueba de aceptación 7 - Gestionar álbum.

Prueba de Aceptación	
Identificador: PA007	**Identificador historia de Usuario:** HU007

Nombre: Gestionar álbumes.
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para: • Crear álbumes. • Visualizar sus álbumes. • Seleccionar sus álbumes.
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Clic en "gestionar géneros", "visualizar géneros" y seleccionar un género, con ello se puede visualizar los álbumes creados en ese género • En esa vista se puede, o crear un álbum nuevo o seleccionar uno ya existente. • En caso de que se quiera crear un álbum este debe completar los campos de nombre e imagen del nuevo álbum. • En caso de que se quiera seleccionar un álbum, se debe seleccionar uno de los presentes en dicha vista. • Crear o seleccionar un álbum se lo realiza con el fin de seguir el proceso de subir canciones.
Resultado deseado: El *backend* le permite al usuario artista la implementación de métodos para visualizar crear o seleccionar álbumes como parte del proceso para subir canciones.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXIII: Prueba de aceptación 8 - Gestionar canciones.

	Prueba de Aceptación
Identificador: PA008	**Identificador historia de Usuario:** HU008
Nombre: Gestionar canciones.	
Descripción: El *backend* por medio del perfil artista permite implementar varios métodos para: • Subir canciones a un álbum.	

• Visualizar canciones. • Actualizar canciones. • Eliminar canciones.
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Clic en “gestionar géneros”, “visualizar géneros”, seleccionar un género y seleccionar un álbum. • En esa vista se puede subir una canción ingresando los campos de nombre y archivo audio de la canción. • Al completar el proceso anterior se presentan las canciones pertenecientes al álbum seleccionado, junto con la canción subida. • En esa vista se puede editar o eliminar canciones mediante botones que lo indican. • En caso de seleccionar editar una canción, esta puede ser actualizadas en los campos de su nombre y o el archivo audio de la canción. • También se puede visualizar todas las canciones del artista independientemente del género o álbum.
Resultado deseado: El *backend* le permite al usuario artista la implementación de métodos para subir, visualizar, actualizar y eliminar canciones.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXIV: Prueba de aceptación 9 - Registrarse a través de un formulario.

	Prueba de Aceptación
Identificador: PA009	**Identificador historia de Usuario:** HU009
Nombre: Registrarse a través de un formulario.	
Descripción: El *backend* por medio del perfil invitado permite implementar un método para poder registrarse, esto se lo realiza por medio de un formulario llenado los siguientes campos: • E-mail.	

• Contraseña. • Nombre. • Fecha de nacimiento. • Rol
Pasos de ejecución: • Dirigirse a ruta "*register*" del Sistema *Web* del *backend*. • Ingresar los campos requeridos. • En caso de que el rol seleccionado sea ciudadano, este será redireccionado directo hacia su perfil.
Resultado deseado: El *backend* le permite al usuario invitado la implementación de métodos para registrarse como usuario ciudadano.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXV: Prueba de aceptación 10 - Iniciar sesión, cerrar sesión y cambiar contraseña.

	Prueba de Aceptación
Identificador: PA010	**Identificador historia de Usuario:** HU010
Nombre: Iniciar sesión, cerrar sesión y cambiar contraseña.	
Descripción: El *backend* por medio del perfil ciudadano permite generar varios métodos para: • Iniciar Sesión. • Cerrar Sesión. • Cambiar contraseña.	
Pasos de ejecución: • Dirigirse a ruta "*login*" del Sistema *Web* del *backend*. • Ingresar las credenciales: correo y contraseña.	

• Dar clic en Iniciar Sesión, si las credenciales son correctas se ingresa al sistema. • Dar clic en Cerrar Sesión, lo que redirige a la ruta "*login*". • Dar clic en "Reestablecer contraseña" e ingresar el correo registrado. • Ingresar al enlace enviado al correo para reestablecer la contraseña. • Volver a la ruta "*login*" e ingresar las credenciales actualizadas.
Resultado deseado: El *backend* le permite al usuario la implementación de métodos para iniciar sesión, cerrar sesión o cambiar contraseña mediante el uso de sus credenciales.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXVI: Prueba de aceptación 11 - Modificar perfil de usuario.

	Prueba de Aceptación
Identificador: PA011	**Identificador historia de Usuario:** HU011
Nombre: Modificar perfil de usuario.	
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para visualizar y editar su perfil por medio de los siguientes campos: • Nombre. • Imagen.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en "Editar Perfil" • Se puede actualizar el nombre y o la foto de perfil. • Clic en "Guardar", redirige a la visualización del perfil de usuario con los campos actualizados.	
Resultado deseado:	

El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar y editar su perfil.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXVII: Prueba de aceptación 12 - Visualizar canciones.

	Prueba de Aceptación
Identificador: PA012	**Identificador historia de Usuario:** HU012
Nombre: Visualizar canciones.	
Descripción: El *backend* por medio del perfil ciudadano permite implementar un método para visualizar las canciones ingresadas por los usuarios artistas. Además, las canciones presentan la siguiente información: • Título de la canción. • Género de la canción. • Artista de la canción. • Álbum de la canción. • Año del álbum • Canción.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en “Visualizar canciones” • Se puede visualizar todas las canciones subidas por los artistas con todos los datos referentes a estas.	
Resultado deseado: El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar canciones.	
Evaluación de la prueba:	

El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XXXVIII: Prueba de aceptación 13 - Gestionar favoritos.

	Prueba de Aceptación
Identificador: PA013	**Identificador historia de Usuario:** HU013
Nombre: Gestionar favoritos.	
Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Agregar canciones a la lista de favoritos. • Visualizar canciones de la lista de favoritos.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en “Seleccionar favoritos” • Se puede visualizar todas las canciones las cuales puede seleccionarse como favoritas. • Clic en “Visualizar Favoritos” • Se puede visualizar todas las canciones seleccionadas como favoritas.	
Resultado deseado: El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar y seleccionar sus canciones favoritas.	
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.	

TABLA XXXIX: Prueba de aceptación 14 - Visualizar géneros y álbumes musicales.

	Prueba de Aceptación
Identificador: PA014	**Identificador historia de Usuario:** HU014
Nombre: Visualizar géneros y álbumes musicales.	

Descripción: El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Visualizar géneros musicales. • Visualizar álbumes. • En cada género musical se puede visualizar las canciones pertenecientes al mismo.
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en “Visualizar géneros” y “Visualizar Géneros” • Se puede visualizar todos los géneros registrados • Al lado de cada género hay un botón de “Canciones”, al hacer clic • Se puede visualizar todas las canciones pertenecientes a ese género. • Clic en “Visualizar álbumes ciudadanos” y “Visualizar Álbum” • Se puede visualizar todos los álbumes registrados • Al lado de cada álbum hay un botón de “Canciones”, al hacer clic • Se puede visualizar todas las canciones pertenecientes a ese álbum.
Resultado deseado: El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar géneros y álbumes musicales, de los cuales también se puede apreciar las canciones correspondientes a cada uno.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XL: Prueba de aceptación 15 - Gestionar *playlist*.

	Prueba de Aceptación
Identificador: PA015	**Identificador historia de Usuario:** HU015
Nombre: Gestionar *playlist*.	
Descripción:	

El *backend* por medio del perfil ciudadano permite implementar varios métodos para: • Crear *playlist*. • Visualizar *playlist*.
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario • Clic en "Gestionar *Playlist*" y "Visualizar *Playlist*" • Se puede visualizar los *playlist* creados y ofrece la opción de crear nuevos. • En caso de seleccionar crear, se debe ingresar el nombre de la *playlist*.
Resultado deseado: El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar y crear *playlist*.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

TABLA XLI: Prueba de aceptación 16 - Gestionar canciones al *playlist*.

	Prueba de Aceptación
Identificador: PA016	**Identificador historia de Usuario:** HU016
Nombre: Gestionar canciones al *playlist*.	
Descripción: El backend por medio del perfil ciudadano permite implementar varios métodos para: • Agregar canciones al *playlist*. • Visualizar canciones del *playlist*. • Eliminar canciones del *playlist*.	
Pasos de ejecución: • Autenticarse en el Sistema *Web* del *backend* • Visualización del perfil de usuario	

• Clic en “Gestionar *Playlist*”, “Visualizar *Playlist*” y seleccionar “Canciones” de una *playlist*. • Se puede visualizar las canciones propias de la *playlist*. • Alado de cada canción aparece una opción para eliminarla de la *playlist*. • Clic en “Gestionar *Playlist*”, “Visualizar *Playlist*” y seleccionar “Agregar Canciones” de una *playlist*. • Se pueden agregar canciones a la *playlist* correspondientemente.
Resultado deseado: El *backend* le permite al usuario ciudadano la implementación de métodos para visualizar y crear *playlist*.
Evaluación de la prueba: El resultado es el esperado y se tiene la aceptación completa del cliente.

ANEXO III

A continuación, se presenta el enlace al Manual de Usuario

https://www.youtube.com/watch?v=24b0ip_U0pg

En este enlace se menciona de manera clara y precisa cada método implementado en el *backend*, a su vez con los perfiles que intervienen en la misma.

ANEXO IV

A continuación, se procede a definir las credenciales de acceso para el *backend*, así como el enlace al repositorio en *GitHub* en donde se encuentra el código fuente y en el apartado de *README* los pasos para realizar la instalación de forma local.

Para acceder al *backend* en producción, ingresar al siguiente enlace:

https://borrador-a0724.web.app/

Repositorio del código fuente del *backend*

El código fuente de todo el proyecto, se encuentra alojado en el repositorio *GitHub*, el cual se puede acceder a través de la siguiente URL:

https://github.com/ivanfraga/generos_albumes.git