UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

LABORATORIO ANALISIS Y DISENIO DE SISTEMAS 2

SECCIÓN P

DOCUMENTACIÓN TÉCNICA AYDISCO

Grupo #2

Carne:	Nombre
201403541	Uzzi Libni Aarón Pineda Solórzano
202004765	Javier Alejandro Gutierrez de León
201709311	Edin Emanuel Montenegro Vásquez
202000194	Alvaro Emmanuel Socop Perez
202003919	Fabian Esteban Reyna Juárez

Guatemala, 15 de Junio del 2023

Índice

Introducción

En el mundo actual de desarrollo de software, la arquitectura basada en servicios (SOA) y los contenedores de Docker se han convertido en herramientas fundamentales para crear aplicaciones modernas y escalables. La SOA permite descomponer una aplicación en servicios más pequeños e independientes entre sí, lo que facilita la modularidad y la reutilización del código. Por otro lado, Docker proporciona una forma eficiente y portátil de empaquetar y ejecutar aplicaciones en diferentes entornos.

En esta fase del proyecto buscamos desarrollar una plataforma web para el almacenamiento de canciones en la nube. Para lograrlo, se aplicará la arquitectura basada en servicios, donde la aplicación se dividirá en elementos más pequeños y autónomos llamados servicios. Estos servicios trabajarán juntos para realizar las tareas necesarias y ofrecer una experiencia fluida a los usuarios.

Además, se emplea una implementación aún más granular de la SOA conocida como microservicios. Los microservicios permiten una separación e independencia aún mayor al dividir la aplicación en elementos más pequeños. Esto facilita la gestión y escalabilidad de los servicios, así como el mantenimiento y la evolución independiente de cada uno de ellos.

Para garantizar el despliegue de los servicios y microservicios, se utiliza Docker. Mediante el uso de contenedores, se crea una infraestructura ligera y flexible en la que los servicios pueden ser empaquetados y ejecutados de manera consistente en cualquier máquina con Docker instalado, independientemente del sistema operativo subyacente.

Objetivos

General:

- Introducir al estudiante en el uso de la arquitectura basada en servicios en la nube y microservicios con tecnologías de contenerización como Docker.

Específicos:

- Que el estudiante aprenda a diseñar e implementar arquitecturas de software.
- Que el estudiante aprenda la estructura del archivo DockerFile para describir contenedores y entornos de ejecución.
- Que el estudiante aprenda la estructura de un archivo de configuración de Docker Compose y describa satisfactoriamente un entorno de ejecución multi-contenedor.
- Que el estudiante aprenda a utilizar herramientas de gestión de proyectos e implementación de control de versiones.

Descripción de Hardware y Software

Máquina Virtual: Standard de Google Cloud (GCP)

Sistema Operativo: Ubuntu 20.04 LTS

Frontend: React 18.2.0

Backend: Node JS

Servidor de desarrollo

Proyecto:

AyDisco es una aplicación de almacenamiento de música en la nube que permite a los usuarios registrar una cuenta, cargar canciones, crear y administrar álbumes, así como reproducir su música en cualquier momento y lugar. La arquitectura del sistema está basada en servicios y microservicios, lo que permite una mayor flexibilidad y escalabilidad.

Explicación de implementación con Docker-compose

El sistema "AyDisco" se implementó utilizando Docker Compose siguiendo los siguientes pasos:

Estructura del proyecto:
Se creó una estructura de directorios para el proyecto, que incluye los archivos de configuración de Docker Compose y los servicios y microservicios del sistema como el frontend.
Definición de servicios en Docker Compose:

Se creó un archivo llamado docker-compose.yml en la raíz del proyecto. Dentro de este archivo, se definieron los servicios necesarios para el sistema, como la base de datos, el backend (servicios y microservicios) y el frontend. Cada servicio se definió como un servicio separado con sus propias configuraciones, incluyendo la imagen Docker utilizada, los puertos expuestos, las variables de entorno y las dependencias.

Configuración de redes y volúmenes:

Se configuraron las redes y volúmenes en Docker Compose.
Se definieron redes para permitir la comunicación entre los servicios y se configuraron volúmenes para persistir los datos en contenedores específicos. Por ejemplo, se configuró un volumen para almacenar los archivos de música cargados por los usuarios.

Construcción y despliegue:

Una vez definidos los servicios en el archivo docker-compose.yml, se ejecutó el comando docker-compose up para construir y desplegar el sistema. Docker Compose se encargó de descargar las imágenes Docker necesarias, crear los contenedores correspondientes para cada servicio y establecer las conexiones entre ellos según la configuración definida.
Configuración adicional:

Se realizaron configuraciones adicionales según las necesidades del sistema, como la configuración de:

- variables de entorno,
- la conexión con la base de datos,
- la instalación de dependencias, etc.

Estas configuraciones se realizaron dentro de los archivos de Dockerfile de cada servicio o a través de las variables de entorno definidas en Docker Compose.

La implementación del sistema utilizando Docker Compose proporcionó ventajas como la fácil gestión de los servicios y su configuración, la capacidad de escalar y replicar los contenedores según sea necesario, y la portabilidad del sistema en diferentes entornos.

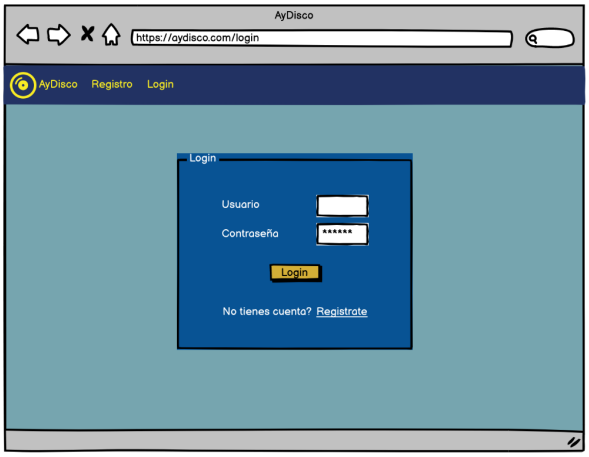
PROTOTIPOS DE FRONTEND:

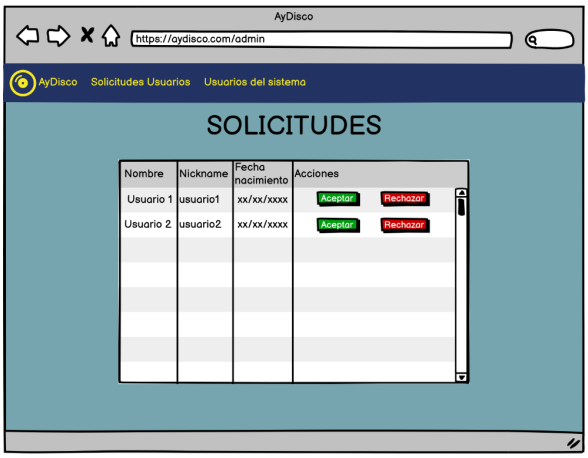
- **Mockups**

Modulo de Administrador

Permite a los administradores gestionar las solicitudes de los usuarios y tomar decisiones sobre su aceptación o rechazo. Una vez que los administradores hayan iniciado sesión, serán dirigidos al panel de administración.

Dentro del panel de administración, los administradores tendrán acceso a una sección dedicada a la gestión de las solicitudes de los usuarios. Aquí se mostrarán las solicitudes pendientes de los usuarios que desean acceder al sistema.



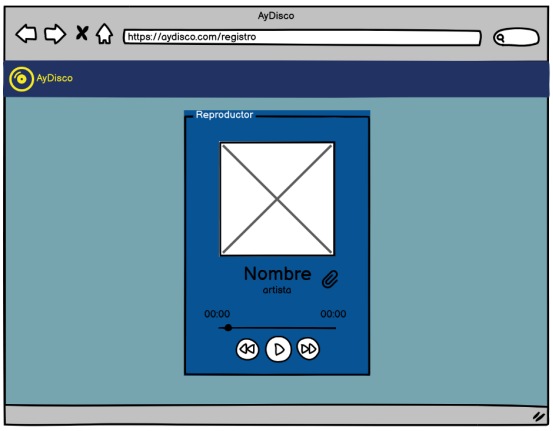


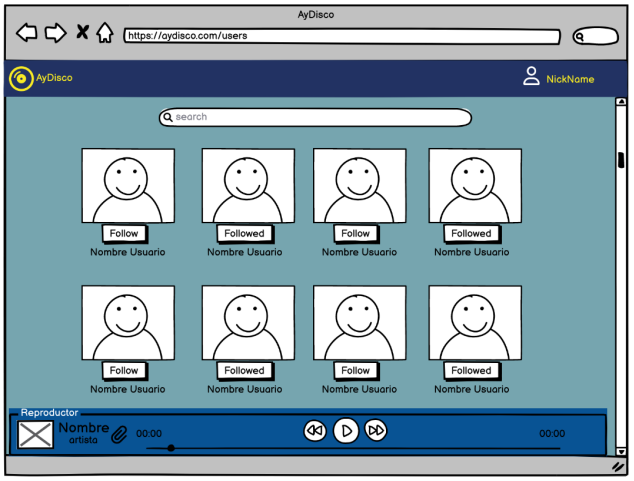
Los administradores tendrán la capacidad de revisar las solicitudes de los usuarios y tomar decisiones sobre su aceptación o rechazo.

Modulo de Usuario

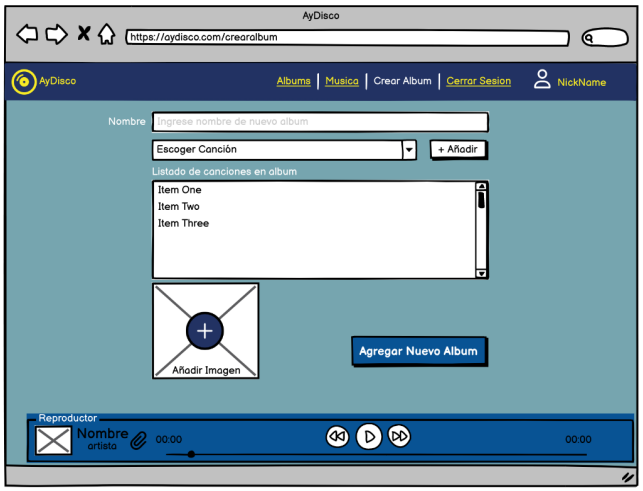
Dentro del módulo de usuario, se incluye un reproductor de música. Este reproductor permite a los usuarios reproducir música,

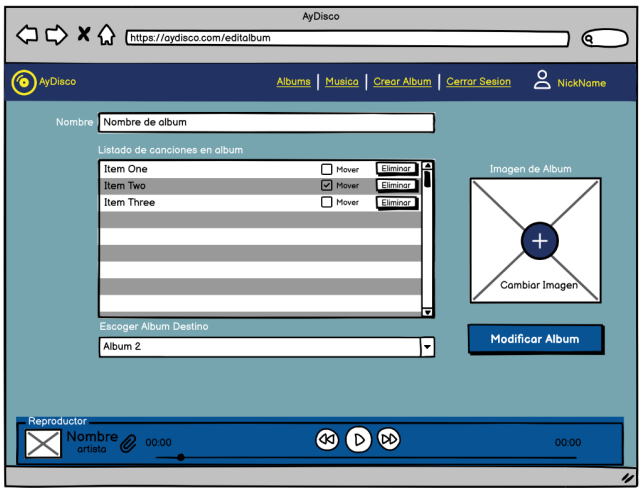
El módulo de usuario puede incluir una sección donde los usuarios pueden ver a otros usuarios en el sistema.





Dentro del módulo de usuario, se incluye una sección dedicada a los álbumes. Los usuarios pueden ver una lista de álbumes disponibles en el sistema
los usuarios pueden seleccionar un álbum específico y acceder a su lista de canciones. Al seleccionar una canción, se reproduce a través del reproductor multimedia mencionado anteriormente.





El módulo de usuario permite a los usuarios subir sus propias canciones para crear nuevos álbumes. El módulo de usuario proporciona funcionalidades de CRUD para álbumes y canciones.

Dentro del módulo de usuario, los usuarios tienen la capacidad de ver su propio perfil, que muestra información personal

PERFIL DEL USUARIO

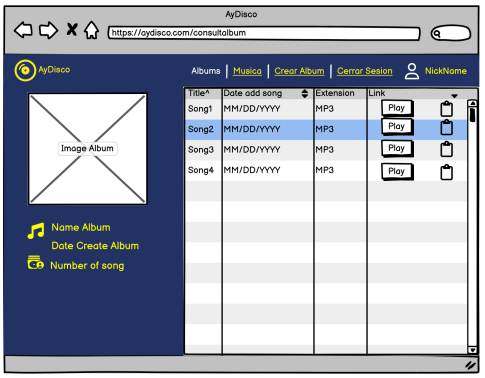


Diagrama de Arquitectura

Descripción:

Capa de presentación:
En la capa de presentación, se utiliza React como tecnología para desarrollar la interfaz de usuario. React es una biblioteca de JavaScript ampliamente utilizada para crear interfaces de usuario interactivas y de respuesta rápida.

Middleware:
El middleware actúa como una capa intermedia entre los microservicios y la capa de presentación. Node.js se utiliza como tecnología de servidor en este caso. Node.js es un entorno de tiempo de ejecución de JavaScript del lado del servidor que permite ejecutar código JavaScript en el servidor.

Integración con GCP:
Google Cloud Platform (GCP) se utiliza para proporcionar servicios en la nube, como almacenamiento, computación, bases de datos y servicios de análisis.

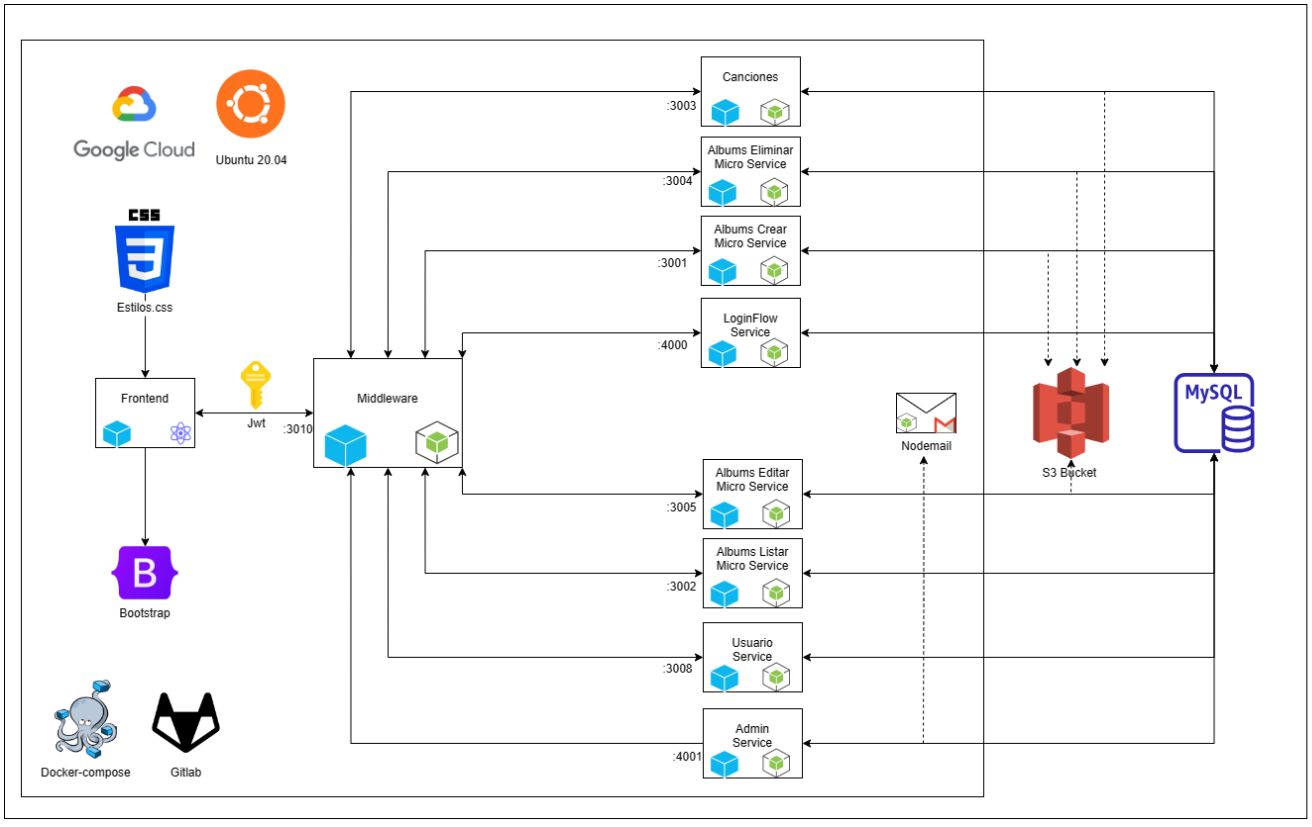
Microservicios:
El enfoque de arquitectura SOA se basa en dividir la funcionalidad en microservicios independientes y especializados. Aquí hay varios microservicios identificados en el diagrama:

petición post a middleware: **localhost:3010?api=xxx&id=xxx**

API	ID	equivale en sus endpoints
albums	crear	localhost:3001/crearalbum
	listar	localhost:3002/listaralbums
	eliminar	localhost:3004/eliminaralbums
	editar	localhost:3005/editaralbum
usuario	profile	localhost:3008/verperfil
	editprofile	localhost:3008/editarperfil
	follow	localhost:3008/follow
	usuarios	localhost:3008/usuarios
cancion	canciones	localhost:3003/getcanciones
	cancion	localhost:3003/getcancion
	sinalbum	localhost:3003/getcancionessinalbum

	infocanciones	localhost:3003/infocanciones
loginflow	registro	localhost:4000/registro
	login	localhost:4000/login
admin	solicitudes	localhost:4001/solicitudes
	usuarios	localhost:4001/usuarios
	aceptar	localhost:4001/aceptar
	rechazar	localhost:4001/rechazar
	dar_baja	localhost:4001/dar_baja
	topsongs	localhost:3009/vertopsongs
	topfollowers	localhost:3009/vertopfollowers
	toptimelisten	localhost:3009/vertoptime

Tabla de servicios y microservicios en modelo Arquitectonico SOA
fuente: Elaboracion Propia



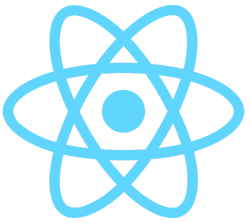
- **tecnologías usadas en el desarrollo del sistema**

Además de la estrategia de branching, en el desarrollo del sistema se utilizan las siguientes tecnologías:

- NodeJS: Node.js se utiliza para construir aplicaciones web y servidores del lado del servidor de alto rendimiento y escalables. Se destaca por su capacidad de manejar grandes volúmenes de solicitudes concurrentes de forma eficiente y su enfoque en la programación basada en eventos y en tiempo real. Node.js es popular en el desarrollo de microservicios debido a su capacidad para modularizar y desplegar componentes individuales de manera independiente, lo que permite una arquitectura de microservicios flexible y fácil de mantener.

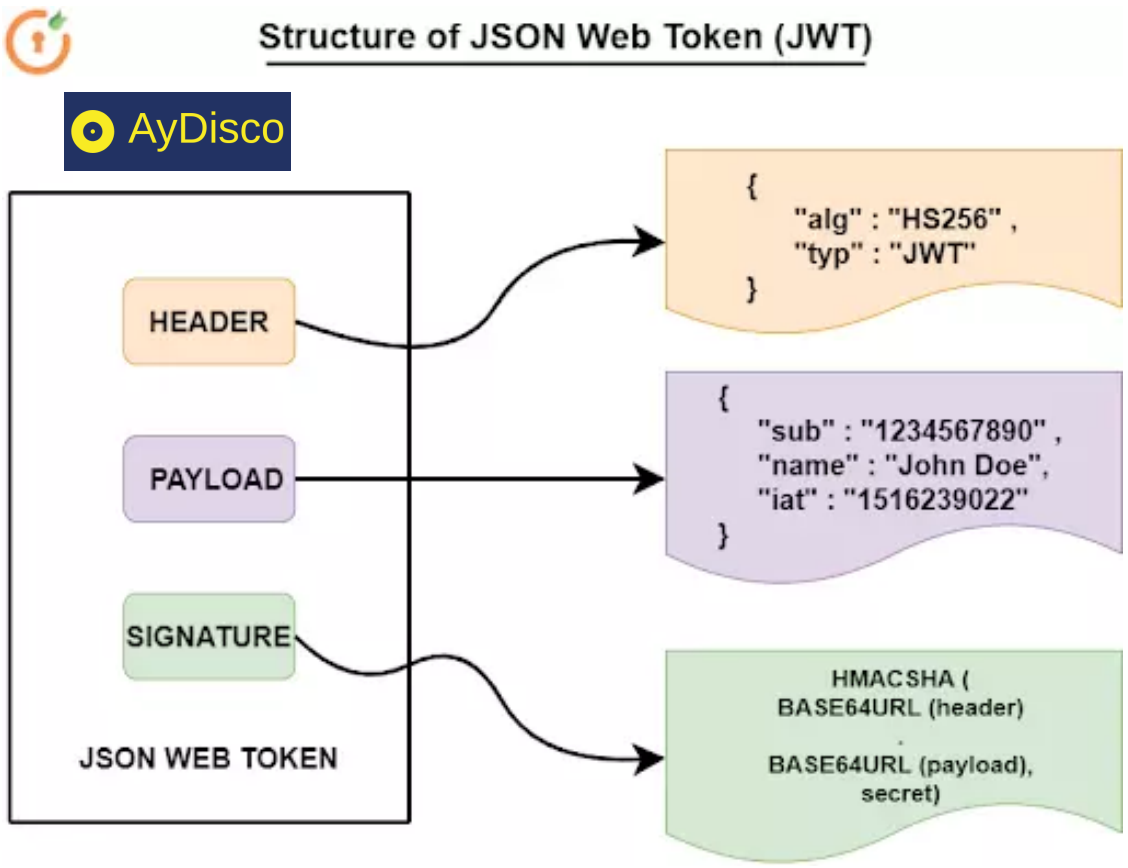
- MySQL : MySQL en AyDisco está alojado en la nube de Google Cloud Platform, específicamente en el servicio de Cloud SQL. Esto permite que la empresa tenga una base de datos escalable, disponible y segura para el almacenamiento y gestión de la información del sistema.

- React: Es una librería de JavaScript para construir interfaces de usuario. Se utiliza principalmente en el desarrollo del front-end o interfaz de usuario del sistema.



- JWT: Es un estándar de autenticación y autorización basado en tokens. Se utiliza para controlar el acceso a las diferentes partes del sistema, asegurando que los usuarios tengan los permisos necesarios para acceder a ellas.

AyDisco Front



AyDisco Back

Diagrama de componentes

El proyecto consta de los siguientes componentes: Frontend, Middleware, Backend, Servidor base de datos y Cloud storage.

Cmponente Frontend contiene la vista web, es decir representa la interfaz de usuario en forma de aplicación web la cual recibe y muestra los datos de otros componentes.

Componente Middleware, maneja la seguridad y la comunicación de la aplicación utilizando JWT el cual genera un token de acceso que a su vez se valida en cada parte del frontend, también sirve como comunicador entre el backend y el frontend.

Componente Backend, tiene almacenado las api's utilizadas en el proyecto, siendo estas las rutas, conexión db, autenticación y nodemailer.

Componente Servidor Base de Datos, representa donde se almacenan los datos del sistema como información de usuarios, álbumes y canciones.

Componente Cloud Storage, servicio utilizado para el almacenamiento de portadas de álbumes y canciones.

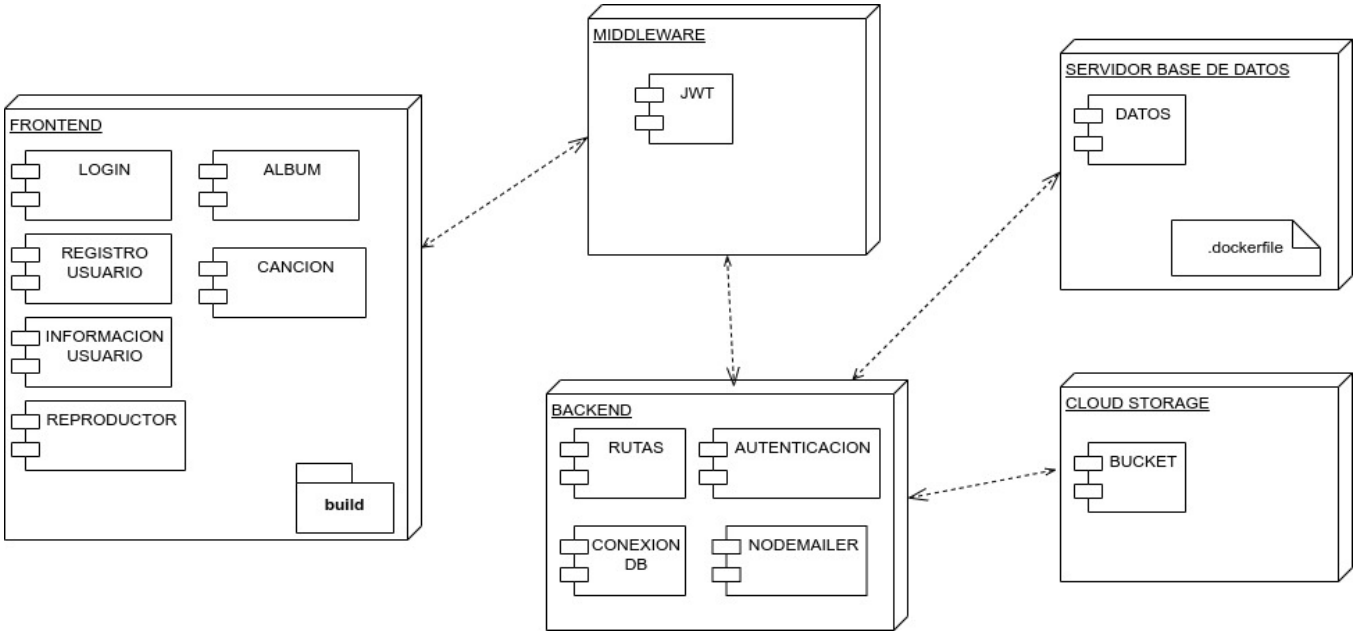


Diagrama de Despliegue

Servidor de Despliegue:

En el centro del diagrama se encuentra el servidor de despliegue, que es el entorno donde se ejecuta Docker Compose.

Este servidor puede ser un servidor dedicado, una máquina virtual o un entorno en la nube, según las necesidades del proyecto.

Contenedores de Servicios:

Los servicios del sistema, como la base de datos, el backend y el frontend, se implementan como contenedores Docker.

Cada servicio se ejecuta en un contenedor independiente, que está aislado y encapsula todas las dependencias y configuraciones necesarias.

Base de Datos:

El contenedor de la base de datos almacena y gestiona los datos del sistema.

Backend:

El contenedor del backend contiene la lógica de negocio y proporciona las API y funcionalidades necesarias para la plataforma web y móvil.

Se comunica con la base de datos para realizar operaciones de lectura y escritura de datos.

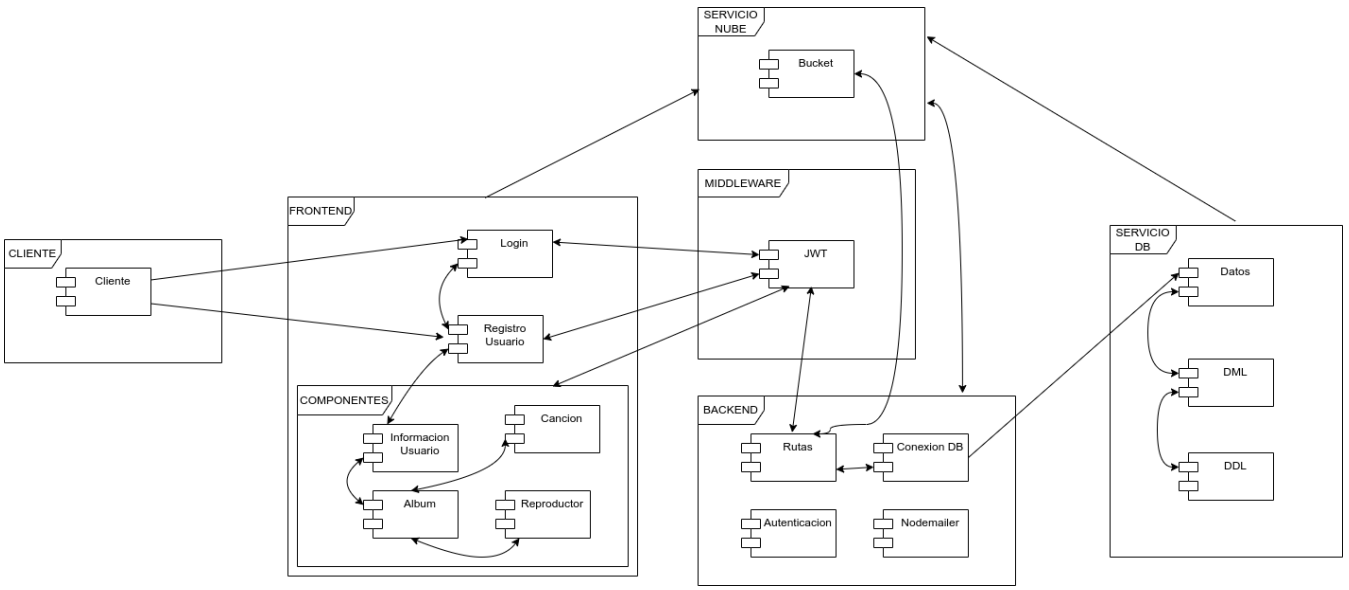
Frontend:

El contenedor del frontend aloja la interfaz de usuario de la plataforma web.

Los usuarios interactúan con el sistema a través del frontend, accediendo a las funcionalidades y visualizando la información.

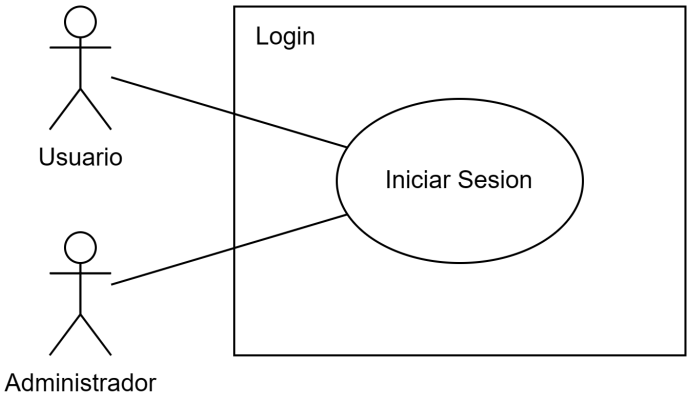
Comunicación entre Servicios:

Los contenedores se comunican entre sí a través de la red definida en Docker Compose. Por ejemplo, el frontend realiza solicitudes HTTP al backend para obtener datos y realizar operaciones.

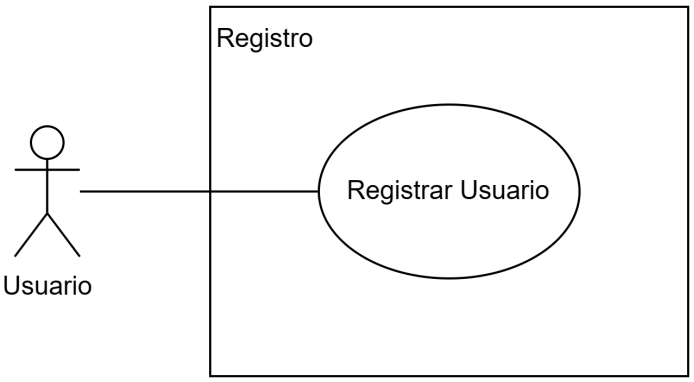


Diagramas de casos de uso

Login
Caso de Uso: Login
Actores: Usuario, Administrador
Tipo: Primario
Descripción: El usuario o administrador inicia sesión en el sistema llenando con el nickname y la contraseña de los usuarios. El sistema comprueba si el usuario existe y se encuentra aprobado por el administrador.

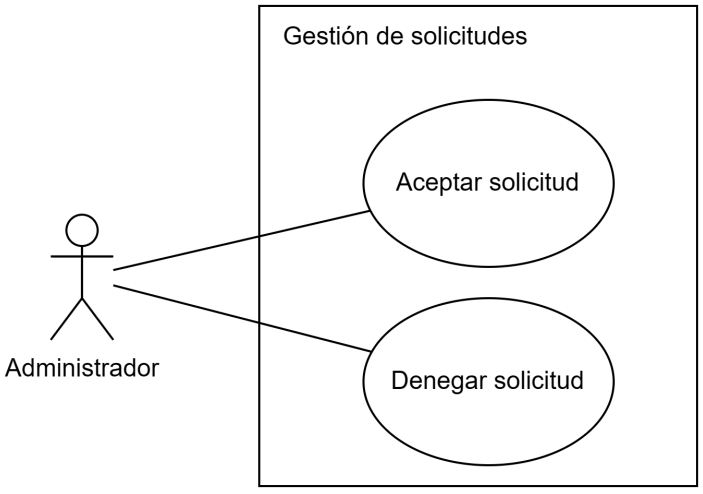


Registro
Caso de Uso: Registro
Actores: Usuario
Tipo: Primario
Descripción: El usuario llena sus datos personales en el formulario de registro y lo envía.

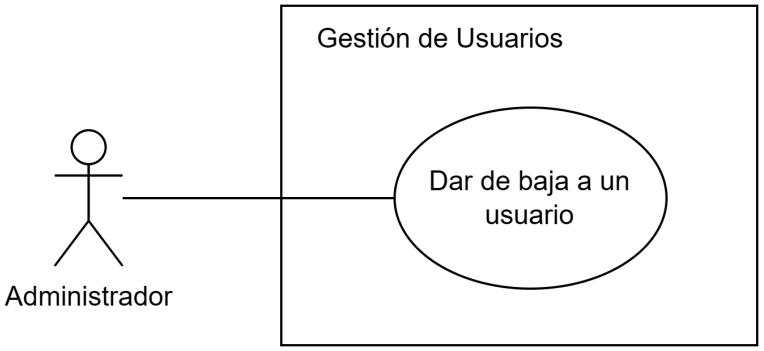


Gestion de solicitudes de usuarios

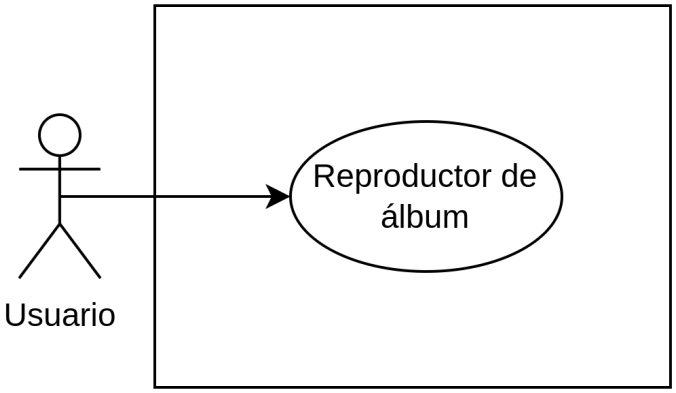
Caso de Uso No.3	
Nombre:	Gestion de solicitudes de usuarios
Autor	Fabian Reyna
Fecha	16 de junio de 2023
Descripción: El administrador puede ver, aceptar o denegar las solicitudes de usuarios desde el panel de solicitudes.	
Actores: Administrador	
Precondiciones:Estar logueado como administrador	
Flujo Normal: 1. El administrador se loguea al sistema 2. El administrador ingresa al módulo de solicitudes de usuarios 3. Procede a aceptar las solicitudes de los usuarios	
Flujo Alternativo: 3. Procede a denegar la solicitudes de los usuarios	
Postcondiciones: El estado del usuario se actualiza en la base de datos en función de la decisión del administrador y se le notifica de la misma al usuario por medio de un correo electrónico.	



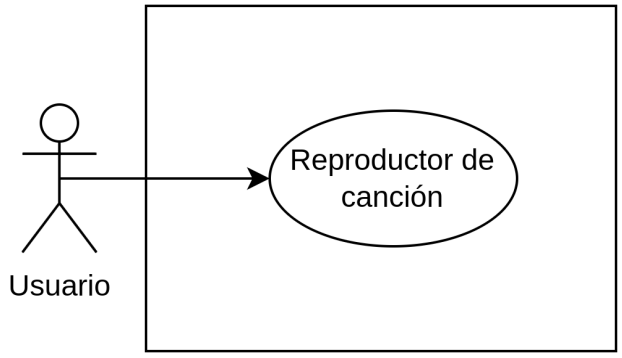
Caso de Uso: Gestión de Usuarios
Actores: Administrador
Tipo: Primario
Descripción: El administrador puede dar de baja a un usuario, indicando el motivo.



Reproducción de álbum
Caso de Uso: Reproducción de álbum
Actores: Usuario
Tipo: Primario
Descripción: El usuario accede al reproductor, donde se cargan las canciones del álbum seleccionado previamente, pudiendo reproducir todas las canciones en una sola vista, con opción de reproducir/pausar, siguiente y anterior canción para poder navegar por la lista de canciones.



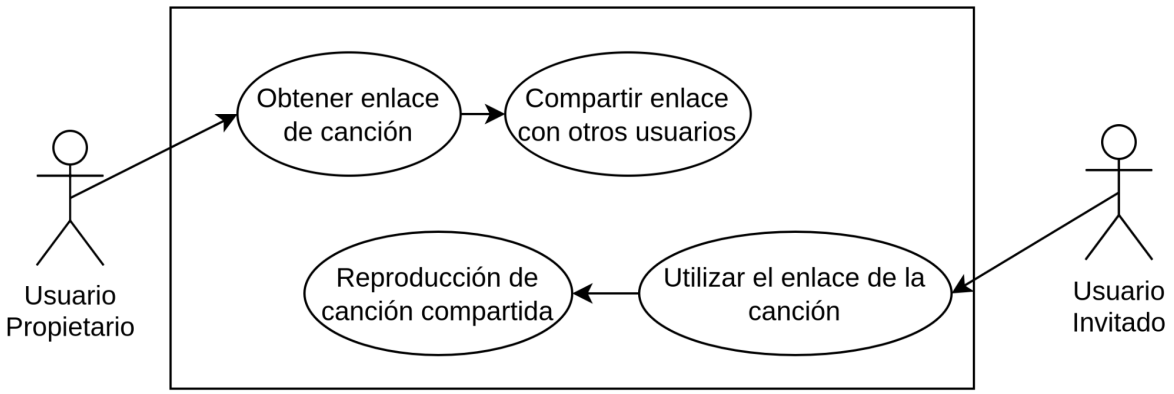
Reproducción de canción
Caso de Uso: Reproducción de canción
Actores: Usuario
Tipo: Primario
Descripción: El usuario accede al reproductor, donde se carga la canción previamente seleccionada, con opción de reproducir/pausar la canción en cualquier momento.



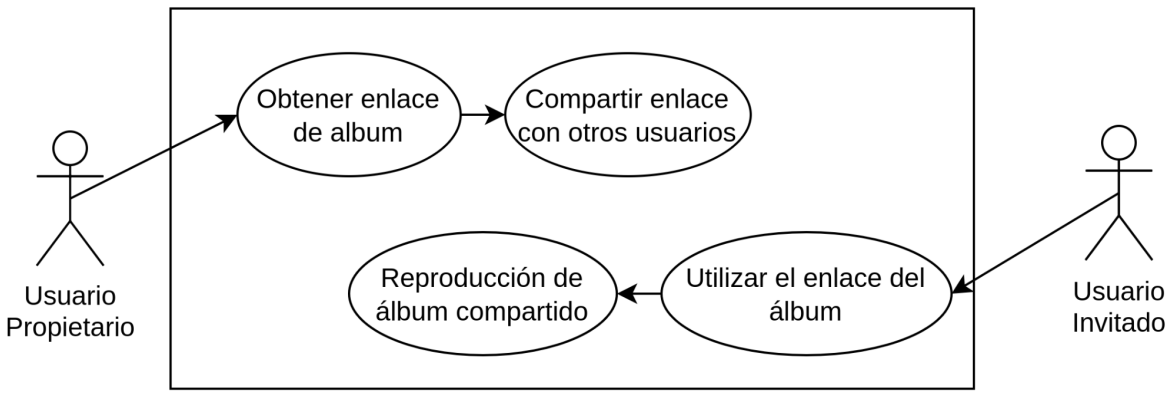
Canciones compartidas
Caso de Uso: Canciones compartidas
Actores: Usuario, usuario secundario o invitado
Tipo: Primario
Descripción: El usuario puede obtener el enlace para compartir con otros usuarios una canción sin importar si el usuario pertenece al sistema. Cuando el usuario secundario o invitado accede al enlace puede reproducir la canción compartida en cualquier momento.

Caso de Uso No.7	
Nombre:	Canciones compartidas
Autor	Javier Gutierrez
Fecha	16 de junio de 2023
Descripción: El usuario puede obtener el enlace para compartir con otros usuarios una canción sin importar si el usuario pertenece al sistema. Cuando el usuario secundario o invitado accede al enlace puede reproducir la canción compartida en cualquier momento.	
Actores: Usuario, usuario secundario o invitado	
Precondiciones: Existencia de la canción	
Flujo Normal: 1. El usuario accede al álbum 2. El usuario presiona el botón de compartir de la canción deseada 3. El usuario le envía el enlace que se copió en su portapapeles a otro usuario 4. El otro usuario accede al enlace (puede hacerlo sin estar logueado en el sistema) 5. El otro usuario reproduce la canción que le compartieron	
Flujo Alternativo: 3. El usuario modifica el enlace por uno incorrecto 4. El otro usuario no puede acceder a ninguna canción	

Postcondiciones:
El usuario secundario o invitado puede acceder a la canción que le compartieron en el momento que desee, siempre y cuando conserve el enlace



Álbumes compartidos
Caso de Uso: Álbumes compartidos
Actores: Usuario, usuario secundario o invitado
Tipo: Primario
Descripción: El usuario puede obtener el enlace para compartir con otros usuarios un álbum sin importar si el usuario pertenece al sistema. Cuando el usuario secundario o invitado accede al enlace puede reproducir el álbum compartido en cualquier momento.



ALBUMS
Caso de Uso: Listar Albums
Actores: Usuario
Tipo: Primario
Descripción: El usuario puede ver todos sus álbumes, luego de iniciar sesión en “AyMusica”.

Caso de Uso No. 8	
Nombre:	Crear Álbum
Autor	Edin Montenegro
Fecha	16 de Junio 2023
Descripción: El usuario de “aymusica” puede crear álbumes, para poder organizar su música.	
Actores: Cliente.	
Precondiciones:Usuario con sesión activa.	
Flujo Normal: 1. El usuario ingresa un nombre para el álbum. 2. El usuario puede seleccionar canciones para añadir al álbum. 3. El usuario observa en una tabla las canciones que ha agregado al álbum. 4. El usuario selecciona la imagen de portada para el álbum. 5. El usuario procede a agregar nuevo álbum a su biblioteca de álbumes. 6. El usuario observa el mensaje de que se ha creado satisfactoriamente el álbum deseado.	
Flujo Alternativo: E-1. El nombre del álbum ya existe. E-2. La imagen seleccionada es muy grande.	
Postcondiciones: Ninguna.	

Caso de Uso No. 9	
Nombre:	Editar Álbum
Autor	Edin Montenegro
Fecha	16 de Junio 2023
Descripción: El usuario puede seleccionar cualquier álbum de su biblioteca y puede editar las propiedades del mismo (nombre, portada, canciones).	
Actores: Cliente.	
Precondiciones:Usuario con sesión activa, usuario con álbumes existentes en biblioteca.	
Flujo Normal: 1. El usuario selecciona el álbum a editar. 2. El sistema redirige a una nueva vista.	

3. El usuario en la nueva vista, puede ver el nombre del álbum, listado de canciones, álbum destino e imagen de portada.
4. El usuario puede cambiar nombre.
5. El usuario puede eliminar canciones de álbum.
6. El usuario puede mover canciones, seleccionando las que desea mover.
7. El usuario elige el álbum destino (previamente ha seleccionado canciones a mover)
8. El usuario puede cambiar la imagen del álbum.
9. El usuario confirma la modificación del álbum.

Flujo Alternativo:
E-1. El nombre del álbum ya existe.
E-2. La imagen seleccionada es muy grande.
E-3. No se ha elegido el álbum destino.

Postcondiciones:
Ninguna.

Caso de Uso No. 10	
Nombre:	Eliminar Álbum
Autor	Edin Montenegro
Fecha	16 de Junio 2023

Descripción: Este caso de uso permite al usuario eliminar un álbum existente en el sistema.

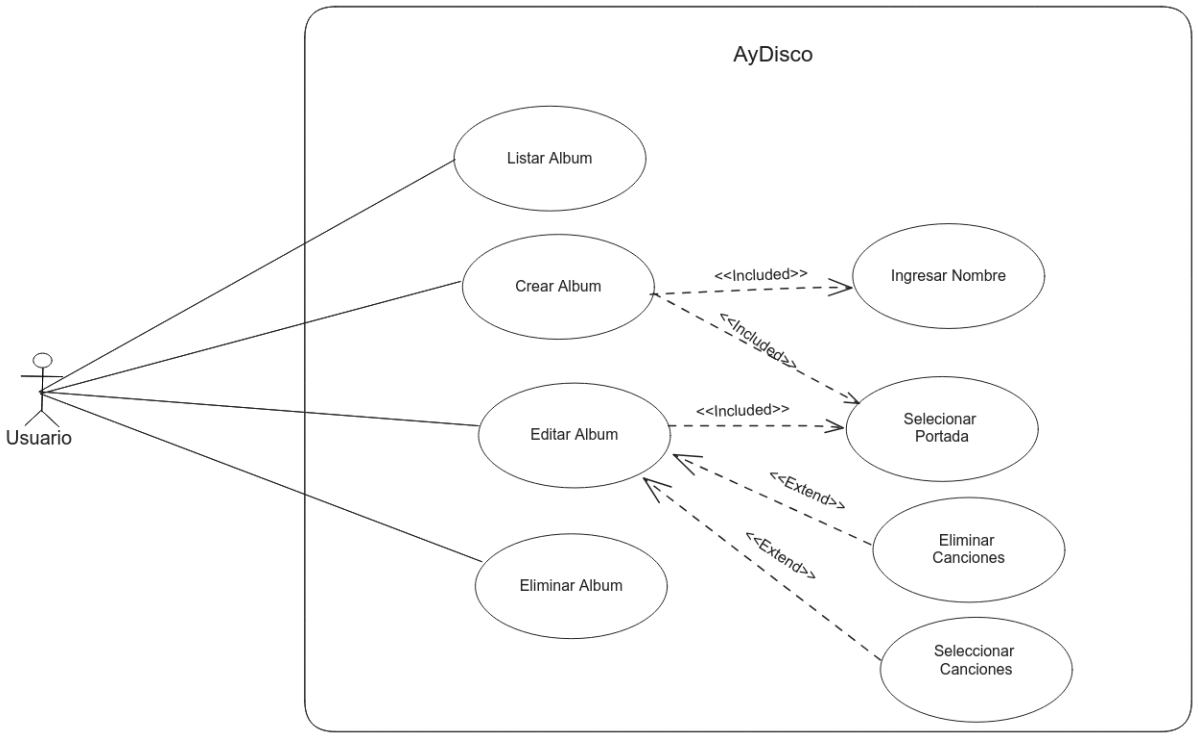
Actores: Usuario.

Precondiciones:Usuario con sesión activa, usuario con álbumes existentes en biblioteca.

Flujo Normal:
1. El sistema muestra al usuario una lista de álbumes disponibles.
2. El usuario selecciona el álbum que desea eliminar.
3. El usuario presiona el botón "Eliminar" correspondiente al álbum seleccionado.
4. El sistema muestra una ventana emergente o una página de confirmación que pregunta al usuario si está seguro de eliminar el álbum.
5. El usuario elige la opción de confirmación para eliminar el álbum.
6. El sistema elimina el álbum seleccionado y muestra una notificación o mensaje confirmando la eliminación exitosa.

Flujo Alternativo:

Postcondiciones: Ninguna.



Caso de Uso: Consulta de álbumes.
Actores: Usuario,Sistema
Tipo: Primario
Descripción: Consultar las propiedades de los álbumes creados por el usuario.

Caso de Uso No. 11	
Nombre:	Consulta de álbumes.
Autor	Libni Pineda
Fecha	16 Junio 2023
Descripción: Consultar las propiedades de los álbumes creados por el usuario.	
Actores: Usuario, Sistema	
Precondiciones: Sesión activa del usuario El usuario debe tener creados álbumes en el sistema.	

Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. El sistema muestra la página principal del usuario. 3. El usuario selecciona el álbum que le interesa. 4. El sistema lo redirecciona a la pagina “consultaAlbum”. 5. El sistema envía un token al middleware. 6. El sistema recupera los atributos del álbum asociado a cada usuario de la base de datos. 7. El sistema muestra los atributos asociados a dicho álbum
Flujo Alternativo: 2.1) El sistema redirecciona a la pagina de login. 2.2) El usuario inicia sesión en el sistema. 7.1) El sistema muestra un mensaje “No es posible obtener los atributos del álbum”
Postcondiciones: Ninguno.

Caso de Uso: Consulta de canciones.
Actores: Usuario,Sistema
Tipo: Primario
Descripción: Consultar las propiedades de las canciones subidas por el usuario.

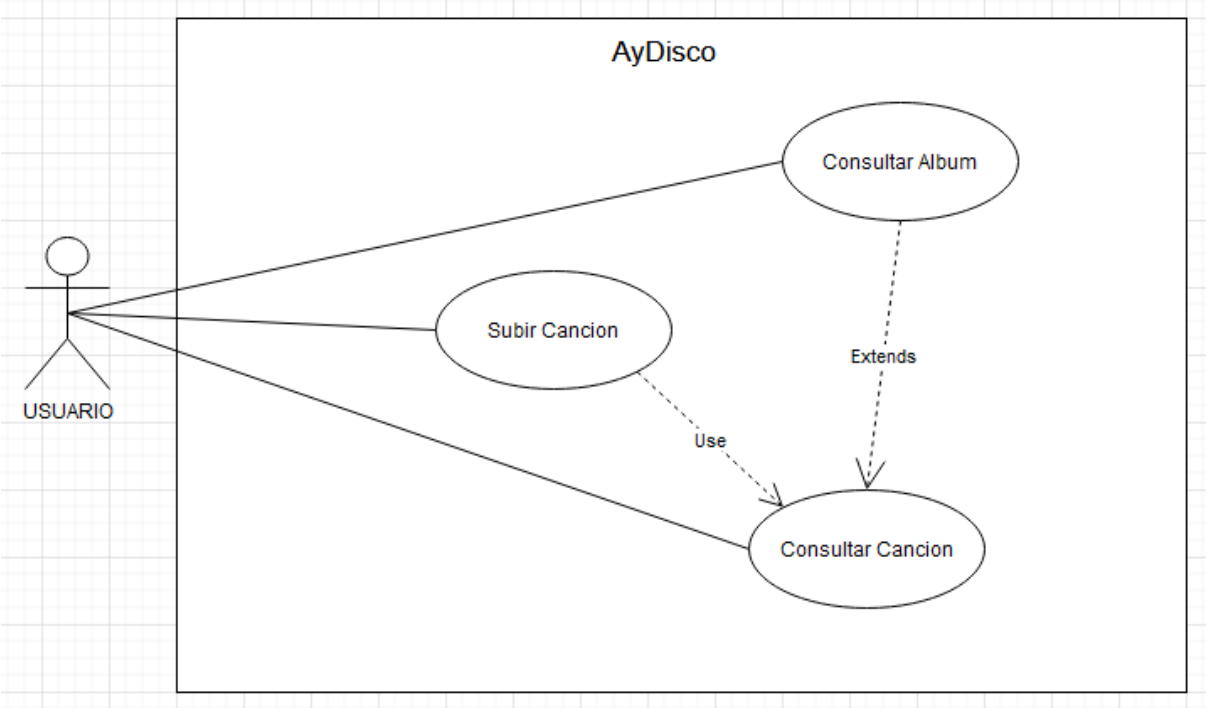
Caso de Uso No. 12	
Nombre:	Propiedad de álbumes.
Autor	Libni Pineda
Fecha	16 Junio 2023
Descripción: Consultar las propiedades de las canciones subidas al sistema por el usuario	
Actores: Usuario, Sistema	
Precondiciones: Sesión activa del usuario El usuario debe tener canciones subidas en el sistema.	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. El sistema muestra la página principal del usuario. 3. El usuario selecciona el álbum que le interesa. 4. El sistema lo redirecciona a la pagina “consultaAlbum”. 5. El sistema envía un token al middleware. 6. El sistema recupera los atributos de las canciones asociado a cada álbum del usuario de la base de datos. 7. El sistema muestra los atributos de cada canción asociados a dicho álbum	

Flujo Alternativo: 2.1) El sistema redirecciona a la página de login. 2.2) El usuario inicia sesión en el sistema. 7.1) El sistema muestra un mensaje “No es posible obtener los atributos del álbum”
Postcondiciones: Ninguno.

Caso de Uso: Subir Cancion.
Actores: Usuario,Sistema
Tipo: Primario
Descripción: Proceso de subir una canción al bucket y las propiedades a la base de datos, en el cual se verifica el nombre y extensión del archivo para que sean únicos, con el objetivo de evitar la duplicación en el sistema.

Caso de Uso No. 13	
Nombre:	Subir cancion
Autor	Libni Pineda
Fecha	16 Junio 2023
Descripción: Proceso de subir una canción al bucket y las propiedades a la base de datos, en el cual se verifica el nombre y extensión del archivo para que sean únicos, con el objetivo de evitar la duplicación en el sistema.	
Actores: Usuario, Sistema	
Precondiciones: Sesión activa del usuario	
Flujo Normal: 1. El usuario inicia sesión en el sistema. 2. El sistema muestra la página principal del usuario. 3. El usuario selecciona la opción “Subir canción” en la interfaz del sistema. 4. El sistema muestra un formulario y una ventana de selección de archivos. 5. El usuario elige el archivo de la canción que desea subir desde su dispositivo. 6. El sistema valida el formato del archivo para asegurarse que sea un archivo de audio compatible. 7. El sistema extrae el nombre y la extensión del archivo de la canción. 8. El sistema realiza una consulta a la base de datos para verificar si existe una canción con el mismo nombre y extensión. 9. Si se encuentra coincidencia en la base de datos no se sube el archivo a la base de datos y se despliega un mensaje. 10. Si no se encuentra coincidencia en la base de datos se suben las propiedades de la canción en la base de datos.	

Flujo Alternativo: 1.1) El sistema redirecciona a la página de login. 1.2) El usuario inicia sesión en el sistema. El sistema muestra un mensaje “No es posible obtener los atributos del álbum”
Postcondiciones: Ninguno.



Editar Perfil de los Usuarios

Caso de Uso No. 14	
Nombre:	Editar Perfil de los Usuarios
Autor	Alvaro Socop
Fecha	16 de Junio de 2023
Descripción: Este caso de uso describe el proceso de ver y editar el perfil de los usuarios en el sistema de música. Se toma en cuenta estética y la seguridad de la información tomando en cuenta el diseño y se oculta la información sensible como las contraseñas	
Actores: Usuario: Persona que está registrada en el sistema AyDisco	
Precondiciones: ● El usuario debe estar autenticado en el sistema. ● El usuario debe tener un perfil existente en el sistema.	

Flujo Normal:

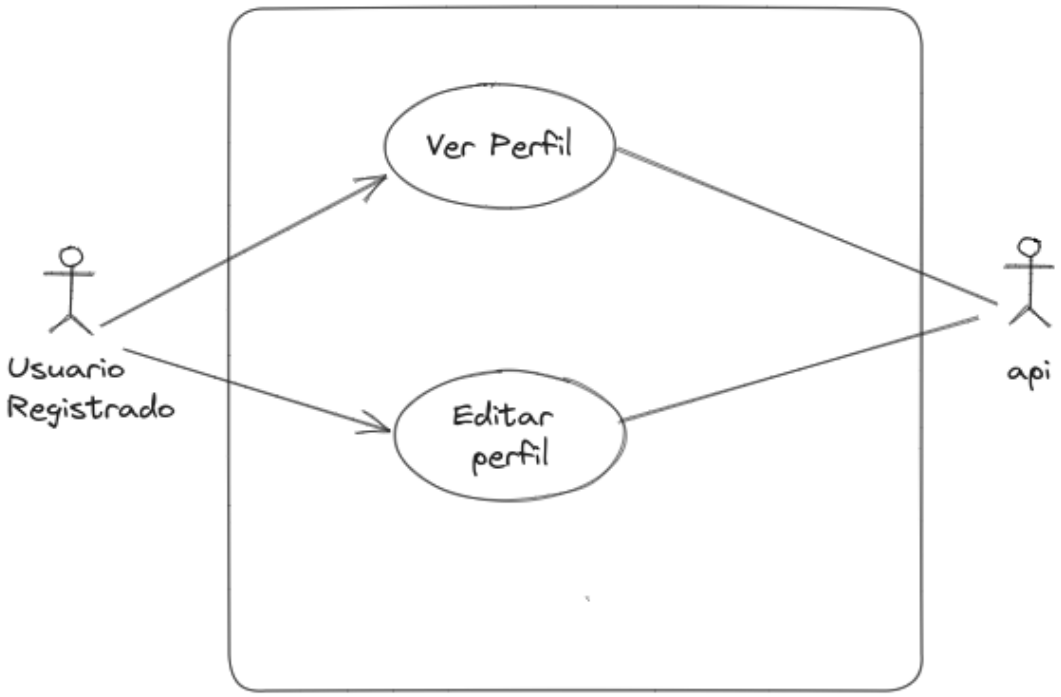
1. Primero el usuario puede acceder a la página de perfil en el sistema de música.
2. El sistema muestra los detalles del perfil actual del usuario, como el nombre, la fecha de nacimiento, el correo.
3. El usuario selecciona la opción de editar perfil.
4. El sistema presenta un formulario con los campos del perfil que pueden ser modificados.
5. Los datos pueden ser cambiados y enviados al sistema solo si la contraseña actual es ingresada dentro del formulario.
6. El usuario realiza los cambios deseados en los campos correspondientes.
7. El usuario confirma los cambios realizados.
8. El sistema valida los datos ingresados por el usuario.
9. El sistema actualiza el perfil del usuario con los nuevos datos proporcionados.
10. El sistema redirige al usuario a la página del perfil, para que pueda visualizar los cambios actualizados.

Flujo Alternativo:

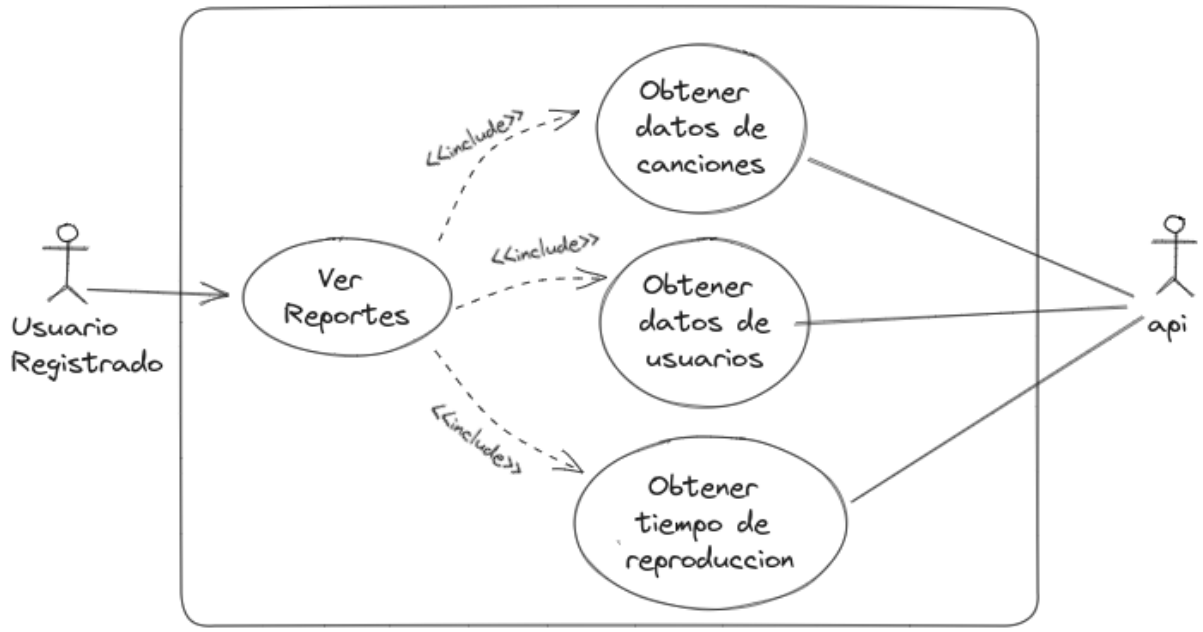
1. El usuario puede no enviar los datos en el paso 7 si no manda la contraseña.

Postcondiciones:

- El perfil del usuario se actualiza con los cambios realizados.
- Los datos modificados por el usuario se reflejan en la página de perfil del sistema de música.



Ver reportes de Administrador
Caso de Uso: Ver reportes de Administrador
Actores: Administrador
Tipo: Primario
Descripción: El administrador puede acceder y visualizar los reportes de las canciones más escuchadas, el Top de los usuarios con mas seguidores, y el Top con los usuarios registrados que tienen más tiempo de reproducciones, osea los que han utilizado con mayor frecuencia el sistema.



Prototipos de interfaces

- **Diagrama de datos o almacenamiento:**

  Esta base de datos está compuesta por cinco tablas: USER, ALBUM, CANCION, Logs y SEGUIDOS.

1 . Tabla USER: Esta tabla almacena información sobre los usuarios del sistema. Contiene las siguientes columnas:

- id: identificador único del usuario (clave primaria).
- nombre: nombre del usuario.
- nickname: apodo o nombre de usuario utilizado para identificar al usuario.
- correo: dirección de correo electrónico del usuario.
- nacimiento: fecha de nacimiento del usuario.
- contraseña: contraseña del usuario.
- tipo_usuario: número que representa el tipo de usuario.
- t_reproduccion: tiempo de reproducción (en minutos) del usuario.
- estado: estado del usuario.

2. Tabla ALBUM: Esta tabla almacena información sobre los álbumes de música.

- idALBUM: identificador único del álbum (clave primaria).
- nombre: nombre del álbum.
- fecha_creacion: fecha de creación del álbum.
- cant_canciones: número de canciones que contiene el álbum.
- imglink: enlace a la imagen del álbum.

3. Tabla CANCION: Esta tabla almacena información sobre las canciones.

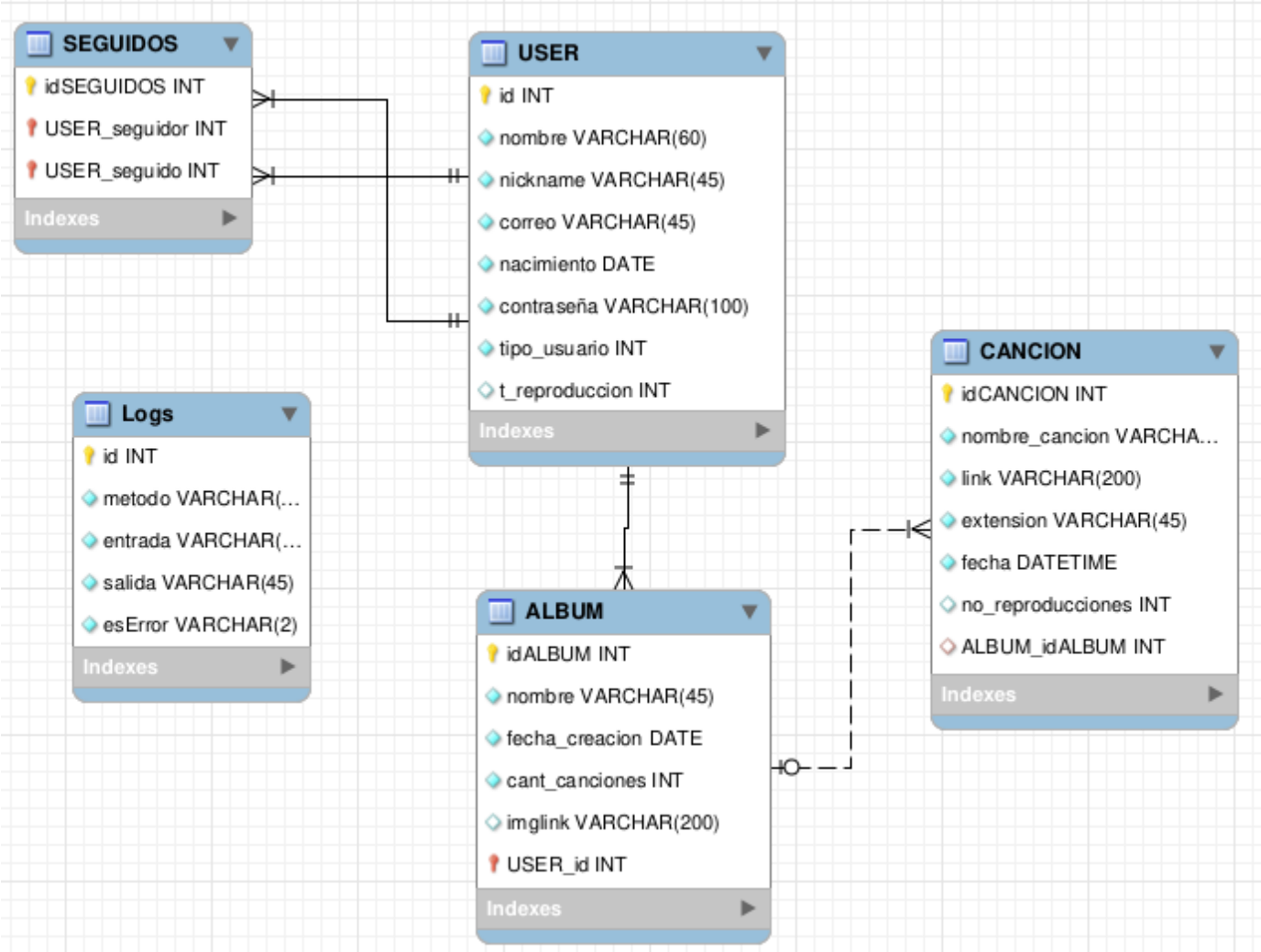
- idCANCION: identificador único de la canción (clave primaria).
- nombre_cancion: nombre de la canción.
- link: enlace a la ubicación de la canción.
- extension: extensión del archivo de la canción.
- fecha: fecha de la canción.
- no_reproducciones: número de reproducciones de la canción.

4. Tabla Logs: Esta tabla almacena registros de logs o registros de actividad.

- id: identificador único del log (clave primaria).
- metodo: nombre del método o función en la que se registra el log.
- entrada: entrada o parámetros de entrada del método.
- salida: salida o resultado del método.
- esError: indica si el log es un error o no.

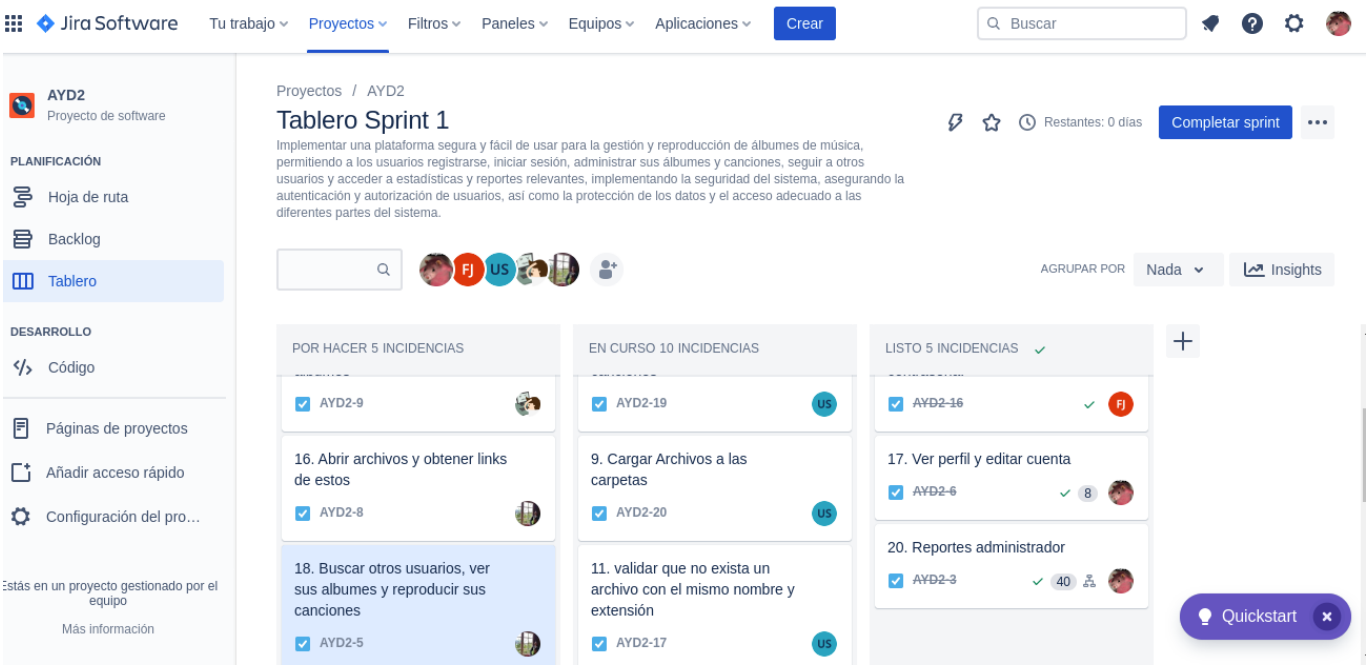
5. Tabla SEGUIDOS: Esta tabla almacena información sobre los usuarios seguidos por otros usuarios. Contiene las siguientes columnas:

- USER_seguidor: identificador del usuario que sigue (clave foránea que hace referencia a la tabla USER).
- USER_seguido: identificador del usuario que es seguido (clave foránea que hace referencia a la tabla USER).



Link de Jira (Gestión de Proyectos)

[JIRA PLANNING LINK]



Toma de requerimientos

La tabla No 1 muestra la lista de requisitos funcionales del sistema: ___AyDisco___
Cada requisito contiene un identificador donde las dos primeras letras indican que es un requerimiento funcional y un número que corresponde a la secuencia de los requisitos. Las columnas nombre y descripción definen el requerimiento, la columna usuario y proceso indican quién debe realizar el requisito y de qué proceso fue derivado dicho requisito. Finalmente, la columna medio indica el "medio" en que se mostrará el requisito (pantalla, papel etc).

FUNCIONALES

ID Requisito	Nombre del Requisito	Descripción del Requisito	Usuario	Medio	Proceso Asociado
RF - 001	Autenticación por correo y contraseña	El sistema debe permitir que los usuarios se autentiquen mediante la introducción de su correo y contraseña	Usuario	Pantalla	Inicio de sesión
RF - 002	Formulario de Solicitud de Registro de Usuario	La plataforma debe contar con un formulario de registro para usuarios para poder interactuar con la plataforma.	Usuario	Pantalla	Registro de usuario
RF - 003	Solicitud de aprobación o rechazo de usuarios	El usuario administrador puede aceptar, denegar y dar de baja a los usuarios indicando el motivo.	Administrador	Pantalla	Gestion solicitudes de usuarios
RF - 004	Reproducción de canciones	El usuario puede reproducir, pausar y cambiar de canción de esta manera poder navegar por la lista de canciones.	Usuario	Pantalla	Reproducción de álbum
RF - 005	Formulário de creación de álbumes	El usuario registrará el nombre, fecha de creación, canciones y una imagen de portada para la creación del álbum.	Usuario	Pantalla	Crear Álbum

RF - 006	Formulario subida de canción	El usuario selecciona el archivo de la computadora y lo carga, también ingresa el nombre con el que desea que se cargue el archivo y fecha de subida.	Usuario	Pantalla	Subir Cancion
RF - 007	Formulario eliminación de álbum	El usuario visualiza la portada de todos los álbumes creados por él y selecciona el álbum a eliminar	Usuario	Pantalla	Eliminar Album
RF - 008	Formulario Edición de álbum	El usuario visualiza la portada de todos los álbumes creados por él, selecciona el álbum y es redireccionado a una página donde podrá efectuar los cambios.	Usuario	Pantalla	Editar Álbum
RF - 009	Formulario Visualización de datos	El usuario visualiza la información que ingresó al momento de efectuar el registro en la plataforma.	Usuario	Pantalla	Visualizar Perfil
RF - 010	Formulario Edición de perfil	El usuario visualiza un formulario en el cual ingresa nueva información o actualización de datos.	Usuario	Pantalla	Editar perfil
RF - 011	Formulario Busqueda de usuarios	El sistema despliega un formulario en el cual puede se visualiza a todos los usuarios de la plataforma o buscar un usuario en concreto.	Usuario	Pantalla	Busqueda de Usuarios
RF - 012	Interacción entre usuarios	El sistema permite que los usuarios aparte de crear álbumes, reproducir canciones, buscar usuarios, también puedan dar follow entre sí.	Usuario	Pantalla	Follow de usuario
RF - 013	Formulario Reporte	El sistema lleva un control de	Administrador	Pantalla	Top canciones

	canción más reproducida.	reproducción de canciones, de esta manera es posible saber que canciones son las más sonadas.			más reproducidas
RF - 014	Formulario Reporte usuarios con mas follow	El sistema lleva un control de la cantidad de follow que tiene un usuario.	Administrador	Pantalla	Top usuarios con más seguidores
RF - 015	Formulario Reporte usuario con más tiempo de canción reproducida.	El sistema lleva un control de tiempo sobre las canciones reproducidas en la plataforma.	Administrador	Pantalla	Top usuarios con más tiempo de canciones reproducidas.
RF - 016	Registro de log y Errores	El sistema lleva un registro de todas las interacciones cliente - servidor que a su vez son almacenadas en la base de datos.	Administrador	Base de Datos	Registro Log y errores.
RF - 017	Visualización de Álbumes	El sistema despliega en interfaz todos los álbumes creados por el usuario en la plataforma.	Usuario	Pantalla	Listar Álbumes
RF - 018	Visualización de Canciones	El sistema despliega en interfaz todas las canciones pertenecientes a un álbum creado por el usuario en la plataforma.	Usuario	Pantalla	Listar Canciones
RF - 019	Compartir Canción	El sistema genera un link para compartir la canción entre usuarios.	Usuario	Pantalla	Link compartir canción
RF - 020	Compartir Álbum	El sistema genera un link para compartir el álbum entre usuarios.	Usuario	Pantalla	Link compartir álbum.

NO FUNCIONALES

La tabla No 2 muestra la lista de la definición de requisitos no funcionales en el sistema, las cuales restringen o condicionan el desarrollo o implementación del sistema.

ID requisito	Descripción
RNF - 001	El sistema debe ser seguro y proteger la privacidad en este caso de la información personal del usuario.
RNF - 002	El sistema debe ser compatible con diferentes plataformas y dispositivos.
RNF - 003	El sistema debe ser rápido y responder rápidamente a las solicitudes del usuario por ejemplo al guardar las canciones en un álbum, o al actualizar el perfil.
RNF -004	La aplicación debe ser intuitiva y fácil de usar, permitiendo una curva de aprendizaje rápida para el usuario.
RNF - 005	La aplicación debe ser escalable y capaz de manejar grandes cantidades de canciones sin afectar el rendimiento.
RNF - 006	El sistema debe ser seguro y confiable para garantizar que las canciones confirmadas por los usuarios se procesen correctamente y sin errores.
RNF - 007	Los datos personales de los clientes, repartidores y empresas deben ser manejados de forma segura y protegidos de posibles ataques informáticos. La plataforma debe contar con medidas de seguridad como JWT para evitar accesos no autorizados y asegurar la integridad de la información.
RNF - 008	El sistema debe ser fácil de usar e intuitivo, para los usuarios. Los formularios y campos deben ser claros y fáciles de entender, y la navegación debe ser sencilla.
RNF - 009	La plataforma debe ser fácil de mantener y actualizar, permitiendo a los desarrolladores agregar nuevas características o corregir errores sin afectar la funcionalidad de la plataforma.

RNF-010	El sistema debe de ser amigable con el usuario indicando a través de alertas los posibles problemas de la aplicación o errores cometidos por los usuarios

GLOSARIO

Glosario con algunos términos utilizados dentro del sistema.

1. Arquitectura de software: Es el conjunto de decisiones y principios que rigen el diseño y desarrollo de un sistema de software. La arquitectura define la estructura, componentes y relaciones de un sistema, y se utiliza para asegurar la calidad, mantenibilidad y escalabilidad del mismo.

2. Prototipado rápido: Es una técnica utilizada en el diseño de sistemas para desarrollar rápidamente prototipos funcionales de un sistema. Permite obtener retroalimentación temprana de los usuarios y detectar problemas en el diseño y funcionamiento del sistema.

3. Diagrama de actividad: Es una herramienta utilizada en el diseño de sistemas para modelar el flujo de actividades en un proceso o tarea. Permite visualizar las diferentes actividades que se llevan a cabo en un proceso, y cómo se relacionan entre sí.

4. Especificación de requisitos: Es un documento que define los requerimientos y funcionalidades que debe tener un sistema. La especificación de requisitos se utiliza como base para el diseño y desarrollo del sistema, y permite asegurar que el sistema cumpla con las necesidades y expectativas de los usuarios.

5. Sprint: Es un ciclo de trabajo en el marco de la metodología Scrum. Durante un sprint se planifican, diseñan, desarrollan y prueban nuevas funcionalidades o características del sistema.

6. Historia de usuario: Es una descripción detallada de una funcionalidad o característica del sistema, desde el punto de vista del usuario. Las historias de usuario se utilizan como base para el diseño y desarrollo de nuevas funcionalidades del sistema.

7. Caso de prueba: Es un conjunto de pasos o acciones que se llevan a cabo para probar una funcionalidad o característica del sistema. Los casos de prueba se utilizan para asegurar la calidad y correcto funcionamiento del sistema.

8. Ciclo de vida del software: Es el conjunto de fases o etapas que atraviesa un sistema de software, desde su concepción hasta su obsolescencia. El ciclo de vida del software incluye etapas como la planificación, análisis, diseño, desarrollo, pruebas, implementación y mantenimiento del sistema.

9. Interfaz de usuario: Es la capa de interacción entre el usuario y el sistema. La interfaz de usuario se encarga de presentar la información y funcionalidades del sistema de manera clara y accesible para el usuario.

10. Análisis de impacto: Es la evaluación de los efectos y consecuencias que tendrá una modificación o cambio en el sistema. El análisis de impacto se utiliza para evaluar los riesgos y beneficios de una modificación en el sistema, y para tomar decisiones informadas en cuanto a su implementación.

11. Caso de uso: Es una técnica utilizada en el análisis de sistemas para identificar los diferentes roles o actores que interactúan con el sistema, y las diferentes acciones que pueden llevar a cabo.

12. Diagrama de flujo de datos: Es una herramienta utilizada en el análisis de sistemas para modelar el flujo de datos en un sistema. Permite identificar los diferentes procesos, entidades y flujos de datos que intervienen en un sistema.

13. Diagrama de clases: Es una herramienta utilizada en el diseño de sistemas orientados a objetos para representar las clases y sus relaciones en un sistema. Permite visualizar las diferentes entidades y objetos que interactúan en el sistema, y cómo se relacionan entre sí.

14. Diagrama de componentes: Es una herramienta utilizada en el diseño de sistemas para identificar los diferentes componentes de un sistema, sus interacciones y relaciones. Permite visualizar la estructura de un sistema y cómo se relacionan los diferentes componentes entre sí.

15. Modelo de datos: Es una representación visual de la estructura de datos en un sistema. Permite definir las tablas, atributos, relaciones y restricciones de una base de datos.

16. Patrón de diseño: Es una solución recurrente a un problema específico en el diseño de sistemas. Los patrones de diseño se utilizan para resolver problemas comunes en el diseño de sistemas, y permiten mejorar la calidad, mantenibilidad y escalabilidad de un sistema.

17. Requerimiento: Es una necesidad o función que debe ser cumplida por un sistema. Los requerimientos se utilizan para definir las funcionalidades y características que debe tener un sistema, y se utilizan como base para el diseño y desarrollo del mismo.

18. Modelo de casos de uso: Es una representación visual de los diferentes casos de uso y actores en un sistema. Permite identificar los diferentes roles o actores que interactúan con el sistema, y las diferentes acciones que pueden llevar a cabo.

19. Prototipo: Es una versión preliminar de un sistema que se utiliza para probar y validar su funcionamiento antes de su lanzamiento. Los prototipos se utilizan para identificar y corregir errores en el diseño y funcionamiento de un sistema.

20. Diagrama de secuencia: Es una herramienta utilizada en el diseño de sistemas para modelar la interacción entre los diferentes objetos y componentes de un sistema. Permite visualizar el orden y secuencia de las acciones que se llevan a cabo en un sistema.

21. JWT: JSON Web Tokens. Es un estándar de seguridad para la transmisión de información entre dos partes. Se utiliza para autenticar y autorizar a los usuarios en la aplicación.

22. ORM: Mapeo objeto relacional. Es una técnica de programación que permite la conversión entre objetos de una aplicación y las tablas de una base de datos relacional.

23. REST: Transferencia de estado representacional. Es un estilo arquitectónico para sistemas web que se basa en la utilización de métodos HTTP para la manipulación de recursos.

Recomendaciones

- Se incluyen recomendaciones para el mantenimiento y la mejora continua del sistema de música, como realizar copias de seguridad periódicas de la base de datos, actualizar las versiones de las librerías y frameworks utilizados.

- Se recomienda utilizar herramientas de gestión de proyectos, como Trello o Jira, para mantener un seguimiento adecuado de las tareas y fomentar la colaboración entre los miembros del equipo. Es importante asignar y distribuir equitativamente las tareas para asegurar la participación equitativa de todos los miembros del equipo.

- Se sugiere seguir los principios de la metodología ágil, como SCRUM, para mejorar la eficiencia y la comunicación dentro de los equipos. Utilizar elementos como el Product Backlog, el Sprint Backlog, la Sprint Planning y las retrospectivas y revisiones de Sprint para monitorear el progreso y realizar mejoras continuas en los proyectos.

- Se recomienda utilizar GitLab como repositorio de código y aplicar una estrategia de branching basada en Git-Flow. Esto facilitará un mejor control de versiones y promoverá la colaboración entre los miembros del equipo.

- Se aconseja aprovechar las herramientas de contenerización como Docker, Dockerfile y Docker Compose para facilitar el desarrollo y despliegue de los servicios y microservicios. Estas herramientas permiten una implementación más rápida, eficiente y escalable, además de una mejor gestión de las dependencias y del entorno de ejecución.
- Es fundamental documentar cada Sprint de manera detallada, incluyendo el Sprint Retrospective y el Sprint Review. Esto permitirá capturar lecciones aprendidas, hacer un seguimiento adecuado de los avances y mejorar la eficacia del equipo en cada iteración.

Conclusiones

- La utilización de herramientas de gestión de proyectos y metodologías ágiles ha demostrado ser efectiva para mantener un seguimiento adecuado de las tareas, promover la colaboración y mejorar la eficiencia en el desarrollo del proyecto.

- La utilización de Docker, Dockerfile y Docker Compose ha brindado beneficios significativos al permitir la contenerización y orquestación de los servicios y microservicios. Esto ha facilitado la implementación, escalabilidad y gestión del entorno de ejecución de la aplicación.

- La documentación exhaustiva de cada Sprint, incluyendo las retrospectivas y revisiones, ha sido fundamental para capturar lecciones aprendidas y mejorar continuamente el proyecto. Esto ha permitido realizar ajustes, corregir errores y optimizar el desarrollo en cada iteración.

- El uso de GitLab como repositorio de código y la implementación de una estrategia de branching basada en Git-Flow han facilitado el control de versiones y la colaboración entre los miembros del equipo, permitiendo un desarrollo más fluido y evitando conflictos en el código.