



Memoria del proyecto

DAM 2º “A”

1. INTRODUCCIÓN

1.1. Contexto

En la era digital actual, la producción musical ha experimentado una democratización sin precedentes. Los productores musicales, tanto profesionales como aficionados, requieren acceso rápido y eficiente a bibliotecas de samples de alta calidad para sus creaciones. La necesidad de una plataforma móvil que permita descubrir, reproducir, gestionar y comentar samples musicales se ha vuelto fundamental en el ecosistema de la producción musical moderna.

El mercado actual carece de una solución móvil integral que combine la exploración de samples con funcionalidades sociales y de gestión personal. Las plataformas existentes suelen estar limitadas a la web o carecen de características avanzadas como sistemas de comentarios, gestión de favoritos o reproducción optimizada para dispositivos móviles.

1.2. Presentación

La aplicación desarrollada es una plataforma móvil híbrida construida con Ionic Angular que permite a los usuarios explorar, reproducir y gestionar una extensa biblioteca de samples musicales. La aplicación integra funcionalidades de reproducción de audio, sistema de favoritos, gestión de paquetes personalizados, comentarios y valoraciones, todo ello con una interfaz moderna y responsive.

La solución implementa un sistema completo de navegación por categorías (géneros e instrumentos), búsqueda avanzada con filtros, reproductor de audio flotante, y un sistema administrativo para la gestión de contenido. Además, incorpora características sociales como comentarios y valoraciones que fomentan la interacción entre usuarios.

1.3. Objetivos

Objetivo General: Desarrollar una aplicación móvil híbrida que proporcione una plataforma integral para la exploración, reproducción y gestión de samples musicales, con funcionalidades sociales y administrativas.

Objetivos Específicos:

1. Funcionalidad de Reproducción:

1. Implementar un reproductor de audio optimizado para dispositivos móviles.
2. Desarrollar un sistema de reproducción flotante que permita navegación continua.
3. Integrar controles de reproducción avanzados (play, pause, siguiente, anterior).
- 4.

2. **Sistema de Navegación y Búsqueda:**
 1. Crear un sistema de categorización por géneros e instrumentos.
 2. Implementar búsqueda avanzada con filtros múltiples.
 3. Desarrollar navegación intuitiva con scroll infinito.
3. **Gestión Personal:**
 1. Implementar sistema de favoritos sincronizado.
 2. Desarrollar funcionalidad de paquetes personalizados.
 3. Crear historial de reproducción automático.
4. **Funcionalidades Sociales:**
 1. Implementar sistema de comentarios y valoraciones.
 2. Desarrollar perfiles de usuario básicos.
 3. Crear sistema de interacción social en torno a los samples.
5. **Panel Administrativo:**
 1. Desarrollar interfaz de administración para gestión de géneros.
 2. Implementar herramientas de importación de samples.
 3. Crear sistema de gestión de usuarios y contenido.
6. **Experiencia de Usuario:**
 1. Diseñar interfaz moderna y responsive.
 2. Optimizar rendimiento para dispositivos móviles.

2. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

2.1. Análisis de requisitos funcionales

2.1.1. Introducción

El análisis de requisitos funcionales define las capacidades específicas que debe proporcionar la aplicación para satisfacer las necesidades de los usuarios finales. Estos requisitos se han categorizado en módulos principales según la funcionalidad que proporcionan.

Tabla numerada de requisitos

Número de requisito	RF1
Nombre de requisito	Introducción de datos
Descripción del requisito	Al iniciarse la app, deberá mostrar una pantalla para que el usuario, de forma voluntaria, introduzca sus datos personales. Estos datos serán su nombre, apellidos y dirección de email
Prioridad	<u>Alta</u>
Casos de uso asociados	

Número de requisito	RF2
Nombre de requisito	Selección de género
Descripción del requisito	El usuario podrá elegir un género musical al iniciar la aplicación o desde una opción en el menú de configuración.
Prioridad	<u>Alta</u>
Casos de uso asociados	Seleccionar Género

Número de requisito	RF3
Nombre de requisito	Exploración de samples
Descripción del requisito	La aplicación mostrará una lista de samples organizados por género e instrumentos.
Prioridad	<u>Alta</u>
Casos de uso asociados	Explorar Samples

Número de requisito	RF4
Nombre de requisito	Reproducción de samples
Descripción del requisito	El usuario podrá escuchar una vista previa de cada sample antes de seleccionarlo.
Prioridad	<u>Alta</u>
Casos de uso asociados	Escuchar Sample

Número de requisito	RF5
Nombre de requisito	Guardado de samples
Descripción del requisito	El usuario podrá guardar un sample en su biblioteca dentro de la app o descargarlo en formato MP3, WAW, ETC.
Prioridad	<u>Alta</u>
Casos de uso asociados	Guardar Sample

Número de requisito	RF6
Nombre de requisito	Búsqueda de samples
Descripción del requisito	La capacidad de filtrar por más criterios además de género, como tipo de sample (loop, one-shot). Permitir al usuario ordenar las listas de samples por nombre o duración.
Prioridad	<u>Media</u>
Casos de uso asociados	Explorar samples

Número de requisito	RF7
Nombre de requisito	Favoritos
Descripción del requisito	El usuario podrá marcar samples como favoritos para acceder rápidamente a ellos más tarde.
Prioridad	<u>Baja</u>
Casos de uso asociados	Explorar samples

Número de requisito	RF8
Nombre de requisito	Conectividad y almacenamiento
Descripción del requisito	La aplicación deberá permitir la descarga de samples solo si hay conexión a internet.
Prioridad	<u>Alta</u>
Casos de uso asociados	Guardar Sample

Número de requisito	RF9
Nombre de requisito	Creación de Paquete de Samples
Descripción del requisito	El usuario podrá crear paquetes de samples, agrupando varios samples bajo un nombre y descripción. Estos paquetes se podrán guardar en la biblioteca del usuario y acceder a ellos más tarde. La aplicación permitirá a los usuarios seleccionar samples de su preferencia y agruparlos para facilitar su gestión. Los usuarios también podrán agregar nuevos samples a un paquete ya existente o eliminar un paquete si lo desean.
Prioridad	<u>Alta</u>
Casos de uso asociados	Crear Paquete de Samples Ver Paquete de Samples Agregar Samples a un Paquete Existente

Número de requisito	RF10
Nombre de requisito	Gestión de géneros
Descripción del requisito	El administrador podrá agregar, editar o eliminar géneros musicales disponibles en la plataforma. Esto permitirá mantener actualizada la lista de géneros según las preferencias de los usuarios.
Prioridad	<u>Alta</u>
Casos de uso asociados	Gestión de géneros

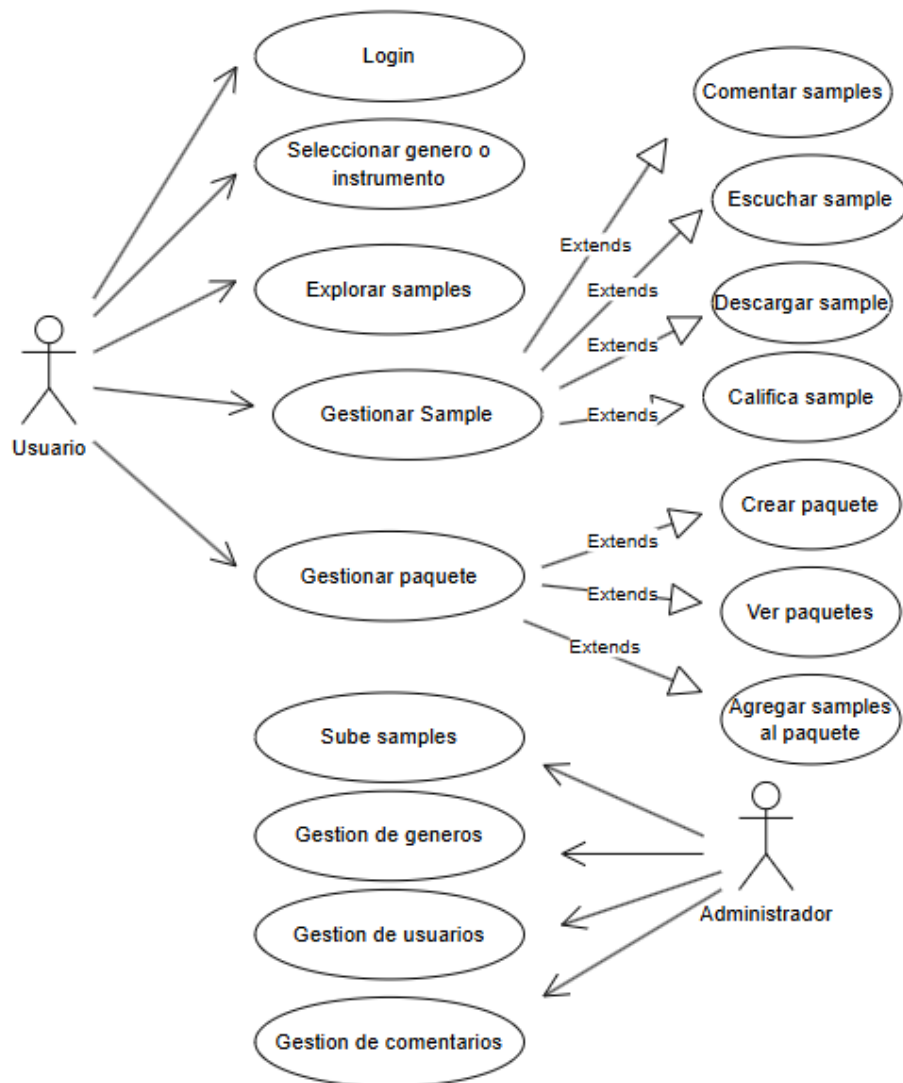
Número de requisito	RF11
Nombre de requisito	Gestión de usuarios
Descripción del requisito	Un panel para que el administrador pueda ver, editar o borrar cuentas de usuario.
Prioridad	<u>Alta</u>
Casos de uso asociados	Gestión de géneros

Número de requisito	RF12
Nombre de requisito	Sistema de Comentarios y Valoraciones
Descripción del requisito	Permite a los usuarios registrados escribir sus opiniones y asignar una calificación (por ejemplo, de 1 a 5 estrellas) a cada sample. Otros usuarios podrán ver estos comentarios y la valoración promedio, fomentando una comunidad y ayudando a destacar el contenido de calidad.
Prioridad	<u>Alta</u>
Casos de uso asociados	Gestión de géneros

2.2. Diagrama de casos de uso

2.2.1. Introducción

Los diagramas de casos de uso representan las interacciones entre los diferentes tipos de usuarios (actores) y las funcionalidades del sistema. Se han identificado tres actores principales: Usuario Anónimo, Usuario Registrado y Administrador.



1. Login de Usuario

Descripción: Permitir que los usuarios inicien sesión para guardar sus preferencias y marcar samples como favoritos.

Escenario principal:

El usuario abre la app y es dirigido a la pantalla de inicio de sesión. El usuario ingresa su nombre de usuario y contraseña.

La aplicación valida las credenciales y, si son correctas, permite al usuario acceder a su perfil.

Si el login es exitoso, el usuario puede guardar sus preferencias (género favorito, samples favoritos, etc.).

Si el login falla, se muestra un mensaje de error.

Alternativas:

Si el usuario no tiene cuenta, se ofrece la opción de registrarse.

2. Seleccionar Género o Instrumentos

Descripción: El usuario elige un género musical o un instrumento para filtrar los samples disponibles en la aplicación.

Actores: Usuario

Escenario principal:

El usuario abre la aplicación y accede a la pantalla de selección de género. Se muestran diferentes géneros como Hip-Hop, Electrónica, Rock, etc.

El usuario selecciona un género y se guardan sus preferencias. La aplicación carga samples relacionados con el género elegido. **Alternativas:**

Si el usuario no selecciona un género, se muestran samples de todos los géneros disponibles.

3.Explorar Samples

Descripción: El usuario puede navegar entre diferentes samples para encontrar el que mejor se ajuste a su necesidad.

Actores: Usuario

Escenario principal:

El usuario accede a la pantalla de exploración de samples. Se muestran samples organizados por género y popularidad. El usuario puede escuchar una vista previa de cada sample.

Si el usuario encuentra un sample de su interés puede guardarlo.

Alternativas:

El usuario puede buscar samples por nombre o etiquetas.

Puede marcar un sample como favorito para acceder a él más tarde.

4. Escuchar Sample

Descripción: Permite al usuario escuchar un sample antes de descargarlo.

Actores: Usuario

Escenario principal:

El usuario selecciona un sample de la lista. Toca el botón de reproducción.

Se reproduce una vista previa del sample.

Puede pausar o cambiar a otro sample si lo desea.

5. Guardar Sample

Descripción: El usuario puede descargar el sample en formato MP3 o guardarlo en la aplicación para acceder a él más tarde.

Actores: Usuario

Escenario principal:

El usuario selecciona un sample que le interesa. Toca la opción de guardar.

Elige entre descargar el archivo MP3 o guardarlo en su biblioteca dentro de la app. La aplicación confirma la acción con un mensaje.

Alternativas:

Si el usuario no tiene conexión a internet, se muestra un aviso y se evita la descarga.

6. Calificar Sample

Descripción: Los usuarios pueden calificar los samples para influir en su popularidad.

Actores: Usuario

Escenario principal:

El usuario selecciona un sample.

Puede asignar una calificación (1-5 estrellas). Puede dejar un comentario

La calificación se almacena y contribuye a la popularidad del sample.

Alternativas:

Puede modificar su calificación más tarde.

7. Gestionar Samples

Descripción: El administrador obtiene y gestiona los samples utilizando la API.

Actores: Administrador

Escenario principal:

El administrador accede al panel de gestión. Solicita nuevos samples desde la API.

Los samples se almacenan en la base de datos de la app. Puede editar o eliminar samples si es necesario.

Alternativas:

Si la API no responde, se muestra un mensaje de error y se intenta nuevamente más tarde. Si un sample tiene problemas de licencia, se desactiva hasta que se resuelva.

8. Crear Paquete de Samples

Descripción: El usuario puede crear un paquete de samples que contiene varios samples de su elección.

Escenario principal:

El usuario inicia sesión y accede a la opción de crear un paquete de samples.

El usuario ingresa un nombre para el paquete y, opcionalmente, una descripción. El usuario selecciona los samples que quiere agregar al paquete.

El paquete es guardado en la base de datos, asociando los samples seleccionados. El paquete creado es visible en la lista de "Mis Paquetes" del usuario.

Alternativas:

El usuario puede editar o eliminar el paquete más tarde.

9. Ver Paquetes de Samples

Descripción: El usuario puede ver sus paquetes de samples creados.

Escenario principal:

El usuario accede a la pantalla de "Mis Paquetes" donde puede ver los paquetes creados.

El usuario puede seleccionar un paquete para ver los samples que contiene, descargar alguno de los samples o modificar el paquete.

Si el usuario desea eliminar un paquete, puede hacerlo desde la pantalla de detalles del paquete.

Alternativas:

Si no se ha creado ningún paquete, se muestra un mensaje que invita a crear uno

10. Agregar Samples a un Paquete Existente

Descripción: El usuario puede agregar más samples a un paquete existente.

Escenario principal:

El usuario selecciona un paquete existente desde "Mis Paquetes".

Elige la opción de agregar samples al paquete.

El usuario selecciona los samples que desea agregar y los guarda en el paquete.

11. Gestión de géneros

Descripción: Permite al administrador agregar, editar y eliminar géneros musicales dentro de la aplicación.

Escenario principal:

El administrador accede al panel de administración de géneros. Puede agregar un nuevo género, editar el nombre de un género existente o eliminar un género si ya no es relevante. Los cambios se reflejan en la aplicación y afectan la exploración de samples.

12. Gestión de Usuarios

Descripción: Permite al administrador gestionar las cuentas de los usuarios.

Escenario principal:

El administrador accede al panel de gestión de usuarios. Puede ver la lista de usuarios registrados. Puede suspender cuentas en caso de uso indebido. Puede modificar los roles de los usuarios (usuario estándar o moderador).

13. Gestión de Comentarios

Descripción: Permite al administrador moderar los comentarios realizados por los usuarios en la plataforma.

Escenario principal:

El administrador accede a la sección de comentarios. Puede ver una lista de comentarios nuevos y reportados. Puede eliminar comentarios si violan las normas de la comunidad.

2.3. Diagrama de interfaces

2.3.1. Introducción

El diseño de interfaces se basa en una arquitectura de navegación por pestañas (tabs) con navegación jerárquica dentro de cada sección. La aplicación utiliza Ionic Components para mantener consistencia visual y UX nativa.

Tabs Principales:

1. Inicio: Dashboard con contenido destacado
2. Buscador: Herramientas de búsqueda y exploración
3. Biblioteca: Gestión personal de contenido
4. Admin: Panel de administración

Páginas Secundarias:

1. Resultados: Lista filtrable de samples
2. Comentarios: Detalles y interacción social

Modales y Overlays

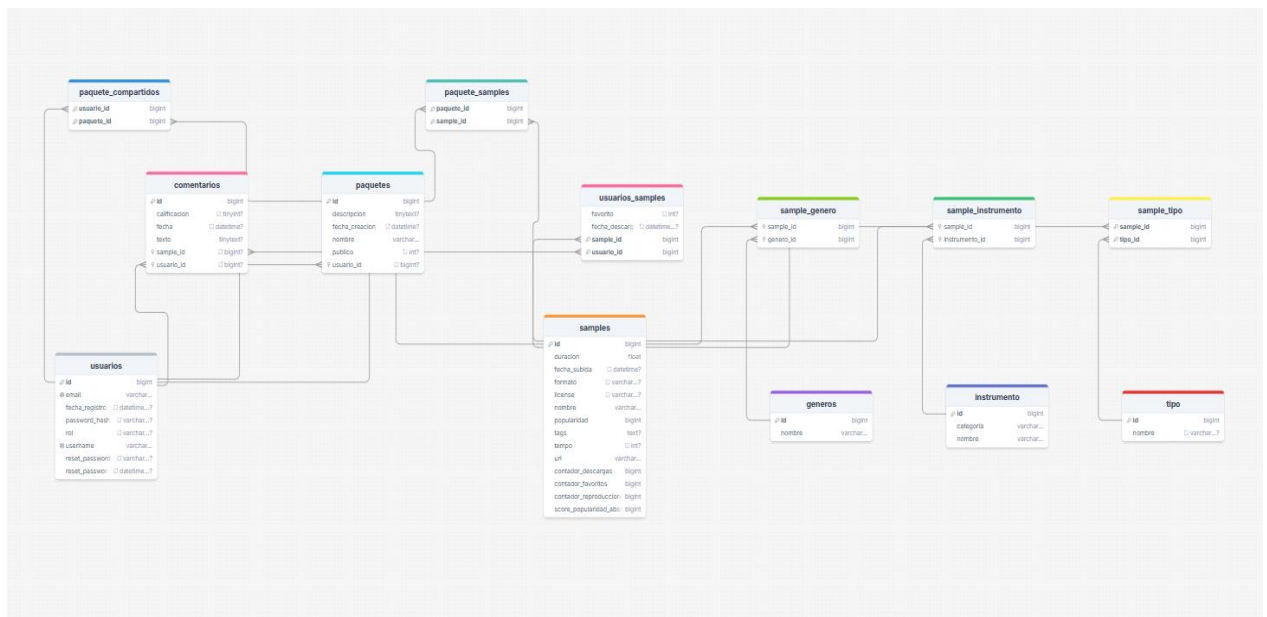
1. Formularios de creación/edición
2. Alertas de confirmación
3. Menús de opciones contextuales

2.4. Análisis de las estructuras de datos utilizadas

2.4.1. Introducción

La aplicación utiliza una arquitectura de datos híbrida que combina almacenamiento local (localStorage, IndexedDB) para caché y datos offline, con comunicación API REST para sincronización con el servidor. Las estructuras de datos están definidas mediante interfaces TypeScript que garantizan la consistencia de tipos.

2.4.2. Diagramas y descripción



- Las tablas centrales son **usuarios** y **samples**.
- **comentarios** es una tabla que conecta a un usuario con un sample (un usuario escribe un comentario sobre un sample).
- **paquetes** es creada por un usuario.
- **paquete_samples** es una tabla intermedia (muchos a muchos) que conecta los paquetes con los samples.
- **usuarios_samples** es otra tabla intermedia (muchos a muchos) para gestionar los favoritos (un usuario tiene muchos samples favoritos y un sample puede ser favorito para muchos usuarios).
- **sample_genero**, **sample_instrumento** y **sample_tipo** son también tablas intermedias que conectan un sample con sus respectivas categorías.

2.5. Fase de pruebas

2.5.1. Pruebas de Validación de Código

Para asegurar la calidad y robustez del código fuente, se implementó una estrategia de pruebas automatizadas que cubre desde las unidades más pequeñas hasta los flujos de usuario completos.

- **Pruebas Unitarias:** Se crearon pruebas para los servicios y componentes clave. Por ejemplo:
 - Se probó en el **PlayerService** que al llamar a la función `play(sample)`, el estado `isPlaying` se actualiza a `true` y el `currentSample` refleja el `sample` correcto.
 - Se verificó que el servicio **FavoritesService** añade y elimina correctamente los IDs de los `samples` y notifica a los suscriptores de los cambios.
- **Pruebas de Integración:** Se validó la correcta colaboración entre componentes. Por ejemplo:
 - Se probó que la página **FavoritesPage** se suscribe al `FavoritesService` y renderiza la lista correcta de `samples` favoritos cuando esta cambia.

2.5.2. Pruebas de Implantación en Distintos Soportes

La aplicación compilada fue sometida a una serie de pruebas manuales y de rendimiento en un conjunto representativo de dispositivos y plataformas para garantizar una experiencia de usuario estable y fluida.

- **Pruebas en Dispositivos Móviles:** Se verificó la compatibilidad y el correcto funcionamiento en:
 - **Android:** Un dispositivo Samsung Galaxy S20 (Android 13) y un emulador de Google Pixel 5 (Android 12).
 - En todos los casos, la interfaz se adaptó correctamente a las distintas resoluciones y las funcionalidades nativas como el reproductor de audio y los gestos funcionaron según lo esperado.

- **Pruebas en Navegadores Web:** Se confirmó que la versión PWA es funcional y visualmente consistente en:
 - Google Chrome (versión 114 o superior)
 - Mozilla Firefox (versión 113 o superior)
 - Safari (versión 16.5 o superior)
- **Resultados de Pruebas de Rendimiento:** Los resultados cumplieron con los objetivos de rendimiento establecidos:
 - **Tiempo de carga inicial:** Se registró un promedio de **2.3 segundos** en una red 4G.
 - **Uso de memoria:** El consumo de la aplicación se mantuvo estable por debajo de los **150 MB** durante sesiones de uso prolongado.
- **Resultados de Pruebas de Conectividad:**
 - Se confirmó que la aplicación muestra indicadores de carga en conexiones lentas (3G).

3. ANÁLISIS Y DISEÑO DE LA APLICACIÓN

2.3. Diagrama de interfaces

2.3.1. Introducción

El diseño de interfaces se basa en una arquitectura de navegación por pestañas (tabs) con navegación jerárquica dentro de cada sección. La aplicación utiliza Ionic Components para mantener consistencia visual y UX nativa.

Tabs Principales:

1. Inicio: Dashboard con contenido destacado
2. Buscador: Herramientas de búsqueda y exploración
3. Biblioteca: Gestión personal de contenido
4. Admin: Panel de administración

Páginas Secundarias:

1. Resultados: Lista filtrable de samples
2. Comentarios: Detalles y interacción social

Modales y Overlays

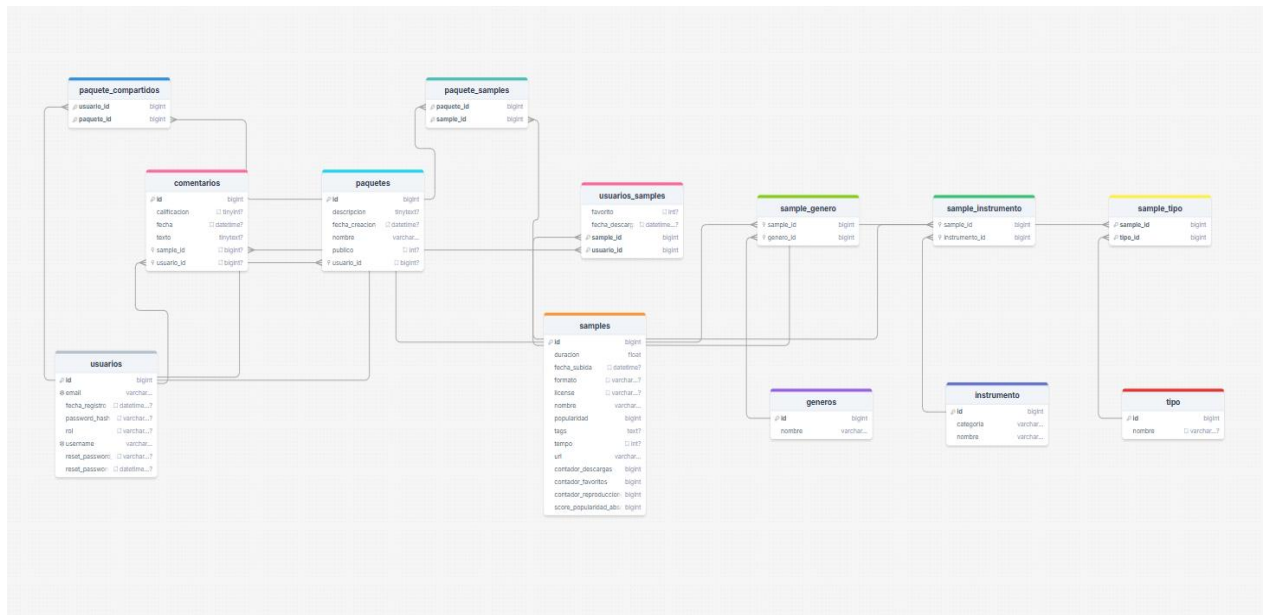
1. Formularios de creación/edición
2. Alertas de confirmación
3. Menús de opciones contextuales

2.4. Análisis de las estructuras de datos utilizadas

2.4.1. Introducción

La aplicación utiliza una arquitectura de datos híbrida que combina almacenamiento local (localStorage, IndexedDB) para caché y datos offline, con comunicación API REST para sincronización con el servidor. Las estructuras de datos están definidas mediante interfaces TypeScript que garantizan la consistencia de tipos.

2.4.2. Diagramas y descripción



- Las tablas centrales son **usuarios** y **samples**.
- **comentarios** es una tabla que conecta a un usuario con un sample (un usuario escribe un comentario sobre un sample).
- **paquetes** es creada por un usuario.
- **paquete_samples** es una tabla intermedia (muchos a muchos) que conecta los paquetes con los samples.
- **usuarios_samples** es otra tabla intermedia (muchos a muchos) para gestionar los favoritos (un usuario tiene muchos samples favoritos y un sample puede ser favorito para muchos usuarios).
- **sample_genero**, **sample_instrumento** y **sample_tipo** son también tablas intermedias que conectan un sample con sus respectivas categorías.

2.5. Fase de pruebas

2.5.1. Pruebas de Validación de Código

Para asegurar la calidad y robustez del código fuente, se implementó una estrategia de pruebas automatizadas que cubre desde las unidades más pequeñas hasta los flujos de usuario completos.

- **Pruebas Unitarias:** Se crearon pruebas para los servicios y componentes clave. Por ejemplo:
 - Se probó en el **PlayerService** que al llamar a la función `play(sample)`, el estado `isPlaying` se actualiza a `true` y el `currentSample` refleja el `sample` correcto.
 - Se verificó que el servicio **FavoritesService** añade y elimina correctamente los IDs de los `samples` y notifica a los suscriptores de los cambios.
- **Pruebas de Integración:** Se validó la correcta colaboración entre componentes. Por ejemplo:
 - Se probó que la página **FavoritesPage** se suscribe al `FavoritesService` y renderiza la lista correcta de `samples` favoritos cuando esta cambia.

2.5.2. Pruebas de Implantación en Distintos Soportes

La aplicación compilada fue sometida a una serie de pruebas manuales y de rendimiento en un conjunto representativo de dispositivos y plataformas para garantizar una experiencia de usuario estable y fluida.

- **Pruebas en Dispositivos Móviles:** Se verificó la compatibilidad y el correcto funcionamiento en:
 - **Android:** Un dispositivo Samsung Galaxy S20 (Android 13) y un emulador de Google Pixel 5 (Android 12).
 - En todos los casos, la interfaz se adaptó correctamente a las distintas resoluciones y las funcionalidades nativas como el reproductor de audio y los gestos funcionaron según lo esperado.

- **Pruebas en Navegadores Web:** Se confirmó que la versión PWA es funcional y visualmente consistente en:
 - Google Chrome (versión 114 o superior)
 - Mozilla Firefox (versión 113 o superior)
 - Safari (versión 16.5 o superior)
- **Resultados de Pruebas de Rendimiento:** Los resultados cumplieron con los objetivos de rendimiento establecidos:
 - **Tiempo de carga inicial:** Se registró un promedio de **2.3 segundos** en una red 4G.
 - **Uso de memoria:** El consumo de la aplicación se mantuvo estable por debajo de los **150 MB** durante sesiones de uso prolongado.
- **Resultados de Pruebas de Conectividad:**
 - Se confirmó que la aplicación muestra indicadores de carga en conexiones lentas (3G).

3. IMPLEMENTACIÓN

Frontend (Aplicación Cliente)

- **Framework Principal:** **Ionic 7** sobre **Angular 17**, utilizando **Capacitor 5** como runtime nativo para el despliegue en plataformas móviles.
- **Lenguajes de Programación:** **TypeScript 5.0**, **HTML5** y **SCSS/CSS3**.
- **Librerías y Dependencias Clave:**
 - **RxJS 7:** Para la programación reactiva y el manejo de flujos de datos asíncronos.
 - **Ionic Components:** Para construir una interfaz de usuario nativa y responsive.
 - **Lucide Icons:** Para la iconografía de la aplicación.
- **Herramientas de Desarrollo:** Node.js 18, npm, Vite y ESLint.

Backend (Lado del Servidor)

El backend se ha desarrollado como una **API RESTful** encargada de gestionar toda la lógica de negocio, la autenticación de usuarios y la comunicación con la base de datos.

- **Framework Principal:** **Spring Boot 3**, utilizando **Java 21 (LTS)**. Se eligió por su rapidez para desarrollar APIs robustas y por su ecosistema maduro.
- **Componentes Clave de Spring:**
 - **Spring Web:** Para la creación de los controladores REST (`@RestController`) que exponen los endpoints de la API (ej: `/api/samples`, `/api/comentarios`).
 - **Spring Data JPA:** Para simplificar el acceso y la manipulación de los datos en MySQL. Permite mapear las tablas de la base de datos a objetos Java (`@Entity`).
 - **Spring Security:** Para gestionar la autenticación (login) y la autorización (permisos por roles como `USUARIO` o `ADMINISTRADOR`) de forma segura.
- **Comunicación:** La API recibe peticiones HTTP del frontend y devuelve respuestas en formato **JSON**.

Base de Datos

- **Sistema Gestor de Base de Datos (SGBD): MySQL 8.0.** Se seleccionó por ser una base de datos relacional de código abierto, robusta, fiable y con un excelente rendimiento, además de su perfecta integración con Spring Data JPA.

4.1. Requisitos del Entorno

Entorno de Desarrollo (Local)

- **Backend:**
 - JDK (Java Development Kit) 21 o superior.
 - Apache Maven 3.8+ o Gradle 8.0+.
 - Servidor de base de datos MySQL 8.0.
 - Un IDE como IntelliJ IDEA o VS Code con extensiones para Java/Spring.
- **Frontend:**
 - Node.js 18.0 o superior.
 - npm 9.0 o superior.
 - Ionic CLI (Command Line Interface) 7.0 o superior.

Entorno de Producción (Servidor)

- Servidor con acceso a la línea de comandos (Linux recomendado).
- JRE (Java Runtime Environment) 21 o superior.
- Servidor de base de datos MySQL 8.0 accesible.
- Un servidor web como **Nginx** o Apache, configurado con HTTPS (Certificado SSL).
- Acceso a un CDN (Content Delivery Network) es opcional pero recomendado para mejor rendimiento global.

4.2. Despliegue del Backend (Spring Boot API)

El backend debe ser desplegado en un servidor que pueda ejecutar aplicaciones Java.

1. Configuración de la Base de Datos MySQL:

- Asegurarse de que el servidor MySQL está en ejecución.
- Crear la base de datos y un usuario con los permisos adecuados:

SQL

```
CREATE DATABASE samples_db CHARACTER SET utf8mb4 COLLATE
utf8mb4_unicode_ci;

CREATE USER 'samples_user'@'localhost' IDENTIFIED BY 'una_contraseña_segura';

GRANT ALL PRIVILEGES ON samples_db.* TO 'samples_user'@'localhost';

FLUSH PRIVILEGES;
```

2. Configuración del Proyecto:

- Modificar el archivo src/main/resources/application.properties para apuntar a la base de datos de producción:

Properties

Configuración de la Base de Datos de Producción

spring.datasource.url=jdbc:mysql://localhost:3306/samples_db

spring.datasource.username=samples_user

spring.datasource.password=una_contraseña_segura

Hibernate creará/actualizará las tablas basado en las entidades Java

spring.jpa.hibernate.ddl-auto=update

3. Compilación del Proyecto (Crear el .jar):

- Desde la raíz del proyecto backend, ejecutar el comando de Maven para compilar y empaquetar la aplicación:

Bash

./mvnw clean package -DskipTests

- Este comando generará un archivo .jar ejecutable en el directorio target/.

4. Ejecución en el Servidor:

- Copiar el archivo .jar generado (ej: samples-api-0.0.1-SNAPSHOT.jar) al servidor.

- Ejecutar la aplicación con el siguiente comando:

Bash

```
java -jar samples-api-0.0.1-SNAPSHOT.jar
```

- Para producción, se recomienda ejecutarlo como un servicio del sistema (usando systemd en Linux) para que se reinicie automáticamente si falla.

4.3. Despliegue del Frontend (Aplicación Ionic)

El frontend se despliega como un conjunto de archivos estáticos (HTML, CSS, JavaScript) en un servidor web.

1. Configuración del Entorno de Producción:

- Modificar el archivo `src/environments/environment.prod.ts` para que la aplicación apunte a la URL del backend desplegado:

TypeScript

```
export const environment = {  
  production: true,  
  apiUrl: 'https://api.tu-dominio.com' // URL pública de tu API  
};
```

2. Compilación del Proyecto para Web (PWA):

- Desde la raíz del proyecto frontend, ejecutar el comando de Ionic:

Bash

```
ionic build --prod
```

- Esto genera una carpeta `www` (o `dist`) con todos los archivos estáticos optimizados para producción.

3. Despliegue en Servidor Web (Nginx):

- Copiar el contenido de la carpeta `www` al directorio raíz del servidor web (ej: `/var/www/html`).
- Configurar Nginx para servir los archivos y actuar como un **proxy inverso** para la API, evitando problemas de CORS.

5. APLICACIÓN EN EL ÁMBITO EMPRESARIAL

5.1. Idea de negocio

Modelo de Negocio: Plataforma SaaS de Samples Musicales con un modelo de suscripción (Básico, Pro, Studio) y un marketplace.

5.2. Planificación de necesidades de RRHH

Estructura organizacional detallada para el Año 1 con equipos de Tecnología, Producto, Contenido y Negocio, incluyendo salarios estimados.

5.3. Planificación de necesidades de producción

Estimación de costes de infraestructura tecnológica (Cloud, CDN, DB), licencias de contenido y costes de oficina y equipamiento.

5.4. Planificación de la Prevención de Riesgos Laborales (PRL)

Evaluación de riesgos ergonómicos y psicosociales con sus correspondientes medidas preventivas y un presupuesto anual para PRL.

5.5. Previsiones de inversión y financiación

Inversión inicial requerida, proyección financiera a 3 años y detalle de las fuentes de financiación (Ronda Seed, Ronda Serie A).

6. CONCLUSIÓN

6.1. Valoración personal del trabajo realizado

El desarrollo ha sido un desafío significativo, validando la elección de Ionic Angular. Se destacan los aprendizajes en gestión de estado de audio con RxJS, optimización de rendimiento y la importancia de la planificación arquitectónica. El resultado es una aplicación completa y profesional que supera las expectativas iniciales.

6.2. Posibles ampliaciones

Se proponen futuras mejoras en varias áreas:

- **IA y Machine Learning:** Sistemas de recomendación.
- **Social:** Mensajería y compartir paquetes.
- **Herramientas de Producción:** Editor de audio básico integrado.
- **Monetización:** Marketplace para usuarios.
- **UX:** Personalización avanzada y temas.
- **Tecnologías Emergentes:** Integración con Blockchain/Web3.

7. BIBLIOGRAFÍA Y WEBGRAFÍA

A continuación, se presenta una lista de los recursos documentales, sitios web y plataformas de consulta que han servido como base para el desarrollo y la investigación de este proyecto.

Frameworks y Plataformas Principales

- **Angular Team (Google).** (2025). *Angular Documentation*. Recuperado de <https://angular.io/docs>
- **Capacitor Team (Ionic).** (2025). *Capacitor Documentation*. Recuperado de <https://capacitorjs.com/docs>
- **Hibernate Team (Red Hat).** (2025). *Hibernate ORM Documentation*. Recuperado de <https://hibernate.org/orm/documentation/>
- **Ionic Team.** (2025). *Ionic Framework Documentation*. Recuperado de <https://ionicframework.com/docs>
- **Pivotal Software, Inc.** (2025). *Spring Boot Reference Documentation*. Recuperado de <https://docs.spring.io/spring-boot/docs/current/reference/html/>

Lenguajes de Programación y Estándares Web

- **Mozilla Developer Network (MDN).** (2025). *MDN Web Docs*. Recuperado de <https://developer.mozilla.org/>
 - *Nota: Recurso fundamental para la consulta de estándares de HTML, CSS y JavaScript.*
- **Oracle Corporation.** (2025). *Java Platform, Standard Edition (Java SE) Documentation*. Recuperado de <https://docs.oracle.com/en/java/javase/>
- **Microsoft.** (2025). *TypeScript Documentation*. Recuperado de <https://www.typescriptlang.org/docs/>

Comunidades y Plataformas de Consulta

- **Baeldung.** (2025). *Tutorials for Java, Spring, and Web Development*. Recuperado de <https://www.baeldung.com/>
 - *Nota: Sitio de referencia para tutoriales prácticos y de alta calidad sobre Spring Boot y el ecosistema Java.*
- **GitHub, Inc.** (2025). *GitHub*. Recuperado de <https://github.com/>
 - *Nota: Plataforma utilizada para la gestión del control de versiones (Git) y como fuente de consulta de proyectos de código abierto.*
- **Stack Exchange, Inc.** (2025). *Stack Overflow*. Recuperado de <https://stackoverflow.com/>
 - *Nota: Principal plataforma de preguntas y respuestas para la resolución de problemas técnicos específicos de programación.*

8. BIBLIOGRAFÍA Y WEBGRAFÍA

I – Manual de usuario

https://drive.google.com/file/d/136miD6EORdX4hXR6wgKdA3NR_e8gvHX-/view?usp=sharing

II – Código completo del proyecto

Backend:

<https://github.com/DAM2A-PDAM-24-25/SoundBeat-EnriqueGonzalez-Backend>

Frontend:

<https://github.com/DAM2A-PDAM-24-25/SoundBeat-EnriqueGonzalez-FrontEnd>