Empresa Barbara Liskov SL

Contacto: Abel 718997@unizar.es



Proyecto **SpreadYourMusic**

**Plan de gestión, análisis, diseño y memoria del proyecto**



31 de Mayo de 2018

**ÍNDICE**

[1. Introducción 3](#_Toc515655227)

[2. Organización del proyecto 4](#_Toc515655228)

[ Equipo 4](#_Toc515655229)

[3. Procesos 5](#_Toc515655230)

[ Procesos de inicio de proyecto 5](#_Toc515655231)

[ Procesos de ejecución y control del proyecto 5](#_Toc515655232)

[ Procesos técnicos 7](#_Toc515655233)

[ Planes 8](#_Toc515655234)

[ Plan de gestión de configuraciones 8](#_Toc515655235)

[ Plan de construcción y despliegue del software 9](#_Toc515655236)

[ Diagrama de Gantt de la aplicación Android 11](#_Toc515655237)

[ Diagrama de Gantt del Backend 13](#_Toc515655238)

[ Diagrama de Gantt de la aplicación Web 14](#_Toc515655239)

[4. Análisis y Diseño del sistema 15](#_Toc515655240)

[ Análisis de requisitos 15](#_Toc515655241)

[ Sistema: Requisitos no funcionales 15](#_Toc515655242)

[ Backend: Requisitos funcionales 16](#_Toc515655243)

[ Backend: Requisitos no funcionales 18](#_Toc515655244)

[ Android: Requisitos funcionales 19](#_Toc515655245)

[ Web : Requisitos funcionales 21](#_Toc515655246)

[ Web: Requisitos no funcionales 22](#_Toc515655247)

[ Diseño del sistema 23](#_Toc515655248)

[ Diagramas de distribución o despliegue: 23](#_Toc515655249)

[ Arquitectura del Backend 24](#_Toc515655250)

[ Diagrama de paquetes del Backend: 25](#_Toc515655251)

[ Compendio de la arquitectura de la aplicación Android 26](#_Toc515655252)

[ Versionado de la aplicación Android 27](#_Toc515655253)

[ Diagrama estructural de la aplicación Android 28](#_Toc515655254)

[ Módulo: Persistencia datos local de la aplicación Android: 31](#_Toc515655255)

[ Módulo: Controlador de la aplicación Android: 33](#_Toc515655256)

[ Módulo: Infraestructura UI de la aplicación Android: 35](#_Toc515655257)

[ Modulo: Music Service de la aplicación Android: 38](#_Toc515655258)

[ Modelo de datos de la aplicación Android:: 40](#_Toc515655259)

[ Diagrama de paquetes de la aplicación Android: 41](#_Toc515655260)

[ Pantallas de la aplicación Android 42](#_Toc515655261)

[ Mapa de navegación Web 86](#_Toc515655262)

[ Tecnologías elegidas: 88](#_Toc515655263)

[ Otros aspectos técnicos de interés: 88](#_Toc515655264)

[5. Memoria del proyecto 89](#_Toc515655265)

[ Inicio del proyecto 89](#_Toc515655266)

[ Ejecución y control del proyecto 89](#_Toc515655267)

[ Cierre del proyecto 91](#_Toc515655268)

[ Jorge Aznar: 91](#_Toc515655269)

[ Ángel Cañal: 92](#_Toc515655270)

[ Abel Chils: 93](#_Toc515655271)

[ Oscar Fraca: 96](#_Toc515655272)

[ Alex Oarga: 97](#_Toc515655273)

[ Jorge Pinilla: 98](#_Toc515655274)

[ Esfuerzo realizado 99](#_Toc515655275)

[6. ANEXO I 100](#_Toc515655276)

[ Bibliografía: 100](#_Toc515655277)

# Introducción

## Resumen

La aplicación a desarrollar consistirá en un reproductor de música en streaming inspirado en Soundcloud y Spotify.

Es una aplicación orientada a todo tipo de usuario, tanto a los músicos que están empezando su carrera en el mundo de la música como a cualquier persona aficionada a ella. Nuestra aplicación permitirá a los usuarios subir canciones, crear listas de reproducción o escuchar canciones utilizando un reproductor propio entre otras funcionalidades. La versión Android permitirá a los usuarios descargar las canciones para poder escucharlas sin necesidad de estar conectado a Internet. También incluirá características sociales, permitiendo a los usuarios seguir a sus artistas favoritos para ver las novedades que publican o subscribirse a listas de reproducción creadas por otros usuarios para enterarse de cambios en esta. El sistema también poseerá integración con redes sociales, así como la posibilidad de autentificación mediante cuenta de Google.

El usuario tendrá recomendaciones personalizadas para el usuario tanto basadas en su historial de reproducción como en geolocalización y facilitará al usuario encontrar canciones, pudiendo buscar canciones por categorías, autor, nombre, así como también mostrando canciones populares dentro de la aplicación.

Nuestra aplicación tendrá soporte web (en navegadores Chrome y Firefox) y como aplicación Android, de manera que el usuario pueda usarla en web, en el móvil o en ambas ya que incorpora un sistema de sincronización de manera que el usuario puede seguir escuchando en cualquier dispositivo la misma canción justa en el momento en el que la dejó.

# Organización del proyecto

## Equipo

El equipo está formado por 7 estudiantes de Ingeniería Informática. Estos poseen conocimientos sobre programación del frontend y backend de una aplicación Web. También se tiene experiencia en el diseño de base de datos así como con la programación sobre la plataforma Android.

Se han desarrollado varias aplicaciones Android entre las que destacan una aplicación para la gestión de notas y otra enfocada al sector veterinario. También se han desarrollado diversos sistemas de información web como son un portal de venta de libros. Por otro lado también posee experiencia en la programación 3d con la creación de varias modificaciones sobre videojuegos. Por último el equipo ha participado en concursos como Google HashCode y uCode.

Por otro lado los integrantes del grupo ya poseen experiencia trabajando juntos. Para desarrollar esta aplicación se han creado 3 grupos dentro del equipo según la experiencia de los integrantes en las áreas de backend, interfaz Web(frontend) y desarrollo de la aplicación Android. Para gestionar el proyecto se ha designado un coordinador general así como coordinadores específicos dentro de cada área.

# Procesos

## Procesos de inicio de proyecto

La aplicación móvil funcionará en dispositivos con Android 5.0 como mínimo, la aplicación web funcionará en los navegadores Firefox y Chrome en su última versión disponible para Windows y Mac. Para realizar las pruebas de la aplicación móvil se va a usar un dispositivo físico con Android5.0. Por otro lado, para las pruebas de la versión web se va a usar un ordenador con el navegador instalado y actualizado (Chrome y Firefox). El sistema inicialmente funcionará sobre un clúster, que contará con un almacenamiento bruto de 20GB. Se ha estimado este tamaño para el clúster ya que presenta un espacio aceptable para almacenar canciones de prueba en esta arquitectura. Este espacio se usará como base en las pruebas iniciales ya que cuando el software se entregue al cliente este podrá poner el tamaño que desee en su clúster puesto que el sistema es fácilmente escalable.

## Procesos de ejecución y control del proyecto

Las comunicaciones del grupo se van a realizar mediante un grupo en la aplicación de mensajería instantánea WhatsApp para tratar temas y comunicaciones poco importantes y eventuales. Sin embargo, para temas que deban ser permanentes y/o deba quedar constancia de esta comunicación, se usarán las Issues o incidencias de la plataforma de alojamiento de los proyectos GitHub. En esta plataforma de alojamiento se almacenará el código fuente del sistema desarrollado y todos los documentos generados durante el desarrollo. Entre estos documentos se encuentra, por ejemplo, las actas de las reuniones con los clientes, que serán redactadas por al menos un miembro del equipo durante dicha reunión. De forma similar se registrarán las reuniones del equipo, los contenidos y las decisiones que puedan tomarse en esas reuniones.

En todas las actas, tanto de reuniones del equipo como reuniones con los clientes se incluirá al menos la fecha y hora de la reunión, la duración de la reunión, los miembros presentes en la reunión y los temas y decisiones que se tomen en la reunión.

Todas las semanas tienen un conjunto de tareas asociadas. Al final de cada semana, el responsable de cada proyecto revisa las tareas que se han realizado esa semana y se realiza una reunión conjunta de todos los miembros del equipo a través de la plataforma Skype, si hay tareas que no se han cumplido, se asigna automáticamente para la siguiente semana, siendo estas tareas las primeras que se deberán hacer. Además, cada semana el responsable del sub-proyecto revisará cuantas tareas se han realizado para la siguiente iteración como métrica de monitorización de la desviación según el plan original. Tras esta revisión el responsable del sub-proyecto asignará las tareas de la semana a todos los miembros del equipo que puedan trabajar esa semana. Durante el desarrollo del proyecto puede haber problemas y disputas entre los miembros del equipo. Para tratar de resolverlos el responsable del sub-proyecto será el primero en mediar entre los miembros en disputa y, si hay alguna razón que haga imposible esta mediación será el resto del equipo quien deberá mediar.

## Procesos técnicos

Las herramientas utilizadas tanto para desarrollo del software (construcción, pruebas y despliegue) serán IntelliJ Idea y Android Studio debido a que ambas dan soporte para los lenguajes de programación Java y Kotlin además de su buena integración con Git, herramienta que se utilizará para el control de versiones. En el caso especial del Frontend se usará como herramienta WebStorm por razones similares a los anteriores.

Para asegurar la calidad del software, al mismo tiempo que se vaya desarrollando se irán creando pruebas unitarias sobre el mismo con JUnit.

Siguiendo una metodología de Diseño incremental, cada dos semanas se tiene planificada la entrega interna de una versión nueva del software. Estas versiones se basan en la versión anterior añadiéndole ciertas funcionalidades y no serán lanzadas al público.

## Planes

### Plan de gestión de configuraciones

El código desarrollado deberá seguir una serie de estándares, siendo los siguientes los aceptados. Para el nombrado de métodos, variables, clases... se utilizará la filosofía CamelCase en los proyectos de Android y Backend y la filosofía SnakeCase en el caso del Frontend. Además, para la escritura de código se seguirá el estándar recomendado por Google en el caso de Android y Backend y el recomendado por la W3Schools para Frontend. Como medida adicional, para permitir que un mayor número de dispositivos puedan visualizar correctamente la página web, se va a desarrollar la web de forma Responsive.

Para asegurar que el desarrollo sigue los estándares de calidad y de nombrado, el coordinador de cada proyecto se encargará de la revisión de los commits de su proyecto y la aceptación o no aceptación de código nuevo. Del despliegue y la puesta en marcha del sistema se encargará Jorge Pinilla y de la correcta administración del sistema de control de versiones (VCS) Git Á ngel Cañal.

Este Sistema de Control de Versiones está compuesto por 4 repositorios, uno en el que se alojan todos los documentos del sistema (Ejmeplo: ”Propuesta Económica” o ”Plan de Gestión”) y 3 repositorios más, uno por cada uno de los proyectos (Android, Backend y Frontend). Para intentar reducir conflictos o problemas, sólo los miembros más experimentados tendrán acceso total a los repositorios (estos son Ángel Cañal y Abel Chils). El resto del equipo sólo tendrá acceso al repositorio de Documentos y al del proyecto en el que esté trabajando.

Para reducir problemas causados por modificación concurrente del mismo fichero de código fuente, se va a usar un flujo de trabajo en Git denominado Git Flow por el que hay dos ramas principales, una para la última versión estable (master) y otra para la versión en desarrollo (develop). Además, para cualquier nueva característica a añadir habrá que crear una nueva rama en Git y, cuando se haya terminado, se volcará esa rama a la rama de desarrollo. No se permite subir código directamente a las ramas develop ni master. Como este flujo de trabajo puede resultar complejo, se simplifica haciendo uso del comando ”git flow”.

Como el sistema Git soporta la gestión de incidencias, se va a aprovechar este sistema para gestionar todas las situaciones inesperadas que pudieran suceder. Además, con la gestión de incidencias se va a utilizar una incidencia especial, llamada ”tarjeta” por la que se va a poder gestionar las tareas pendientes por hacer. Para ello, el responsable de cada equipo creará una incidencia por tarea a realizar (lo más pequeña posible). Estas incidencias podrán estar en 3 estados, ”Abierto”, ”En progreso” y ”Terminado” además de estar asignado a un punto en el tiempo en el que deberán estar terminadas por completo. Para pasar de ”En progreso” a ”Terminado” el coordinador del equipo debe dar el visto bueno e incorporar el nuevo código al resto (rama develop).

### Plan de construcción y despliegue del software

Para la compilación del backend utilizamos Maven que facilita las dependencias entre los paquetes y garantiza que todos compilen con las mismas dependencias.

Para la compilación de Android se utiliza Grandle de una manera similar a Maven.

Para la compilación de la Web se realiza una compilación manual de Jekyll.

Para realizar los test, se realizan de manera manual, primero en los ordenadores personales de cada integrante y luego integrandolo con el servidor de producción.

Los modulos son independientes entre ellos y se comunican principalmente mediante peticiones HTTP, esto permite simular las peticiones antes de probarlo con el entorno real de producción.

No existe una construcción automática del software, sino que se realiza una construcción cuando se ha terminado alguna feature. Se van realizando compilaciones locales y unicamente cuando hay una release se realiza una compilación en el servidor.

Debido a presupuesto hemos simulado todo el entorno de pruebas en un solo servidor mediante máquinas virtuales y virtualización anidada. El servidor host únicamente es el encargado de redirigir las peticiones mediante un HAProxy a modo de router y se utilizan 4 máquinas virtuales que emulan los servicios de la aplicación.

3 máquinas virtuales constituyen el servicio de almacenamiento, encargado de servir los datos, almacenar la información y garantizar la consistencia y seguridad de los datos. Se utiliza ceph (<https://ceph.com/>) como sistema de almacenamiento distribuido definido por software sobre debian como sistema operativo, para desplegarlo se utiliza la herramienta semi-automática de ceph-deploy.

Se utiliza RadosGateway (<http://docs.ceph.com/docs/master/radosgw/>) como interfaz para exportar los datos mediante HTTP, de una manera compatible con Amazon S3, las canciones se almacenan en Buckets y disponen de un URL único. Es posible añadir servidores de almacenamiento para aumentar prestaciones de manera sencilla.

El cuarto servidor contiene Proxmox (<https://www.proxmox.com/en/>) y kvm. Se encarga de virtualizar tanto la base de datos como el backend y su integración con ceph permite una gestión unificada del servicio de almacenamiento mediante una interfaz sencilla.

La idea de utilizar máquinas virtuales es la gran escalabilidad que permite y la posibilidad de configurar resistencia a fallos automática, el servicio de almacenamiento tolere un 1 fallo de un servidor completo sin caida de servicio.

La separación entre backend y el servicio de almacenamiento permite redirigir las peticiones sin sobrecargar ninguno de los dos servicios, un solo servidor de backend atenderá a pequeñas peticiones de consultas mientras que los 3 servidores de almacenamiento atienden a las peticiones de datos de canciones. Ademas, dicha separación y el almacenamiento compartido abre la posibilidad de añadir facilmente más nodos de proxmox gestionarlos de manera unificada y permitir así escalabilidad horizontal.

*Puertos:*

ceph01 192.168.200.10 2000 RGW(HTTP) 7480

ceph02 192.168.200.11 2001 RGW(HTTP) 7480

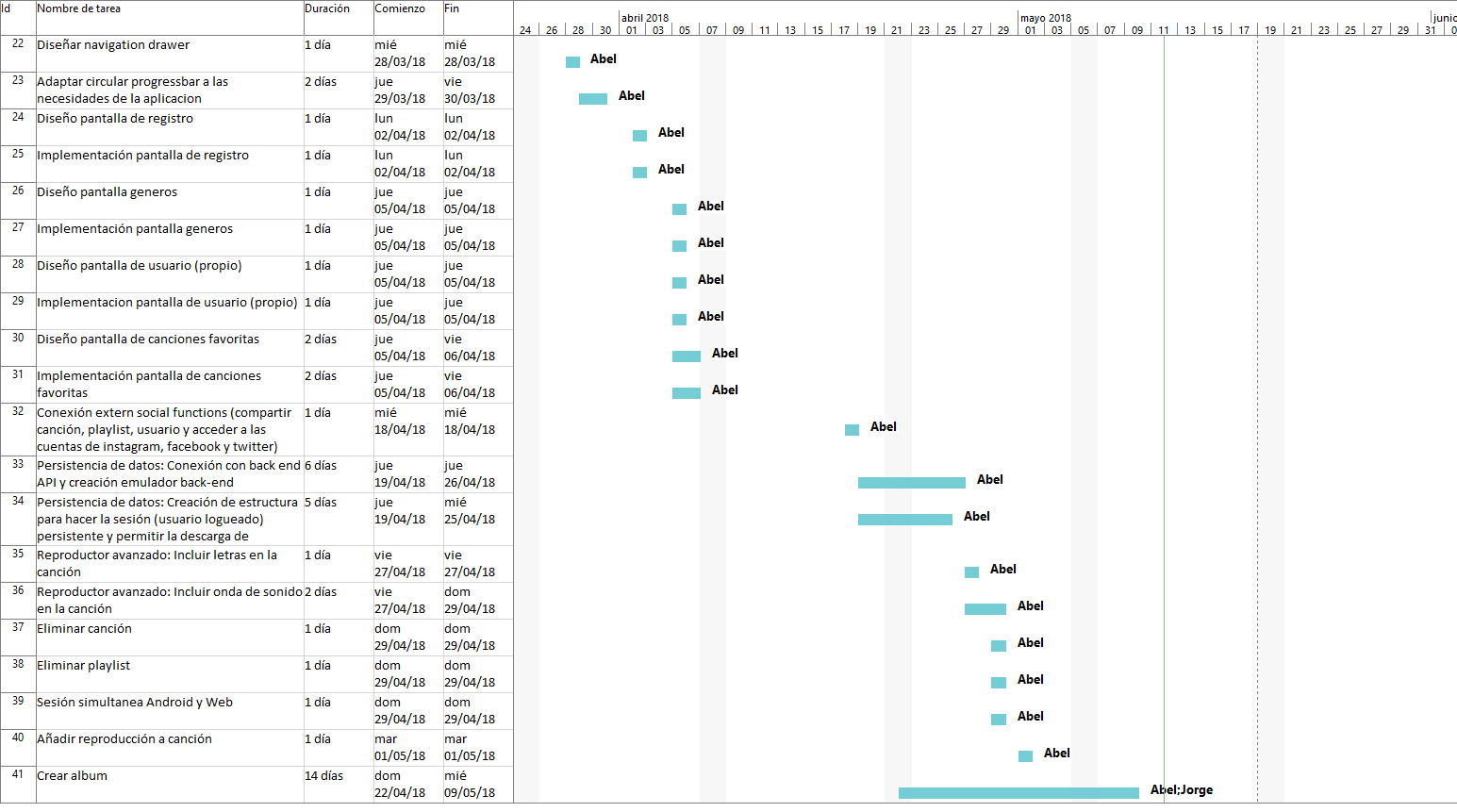
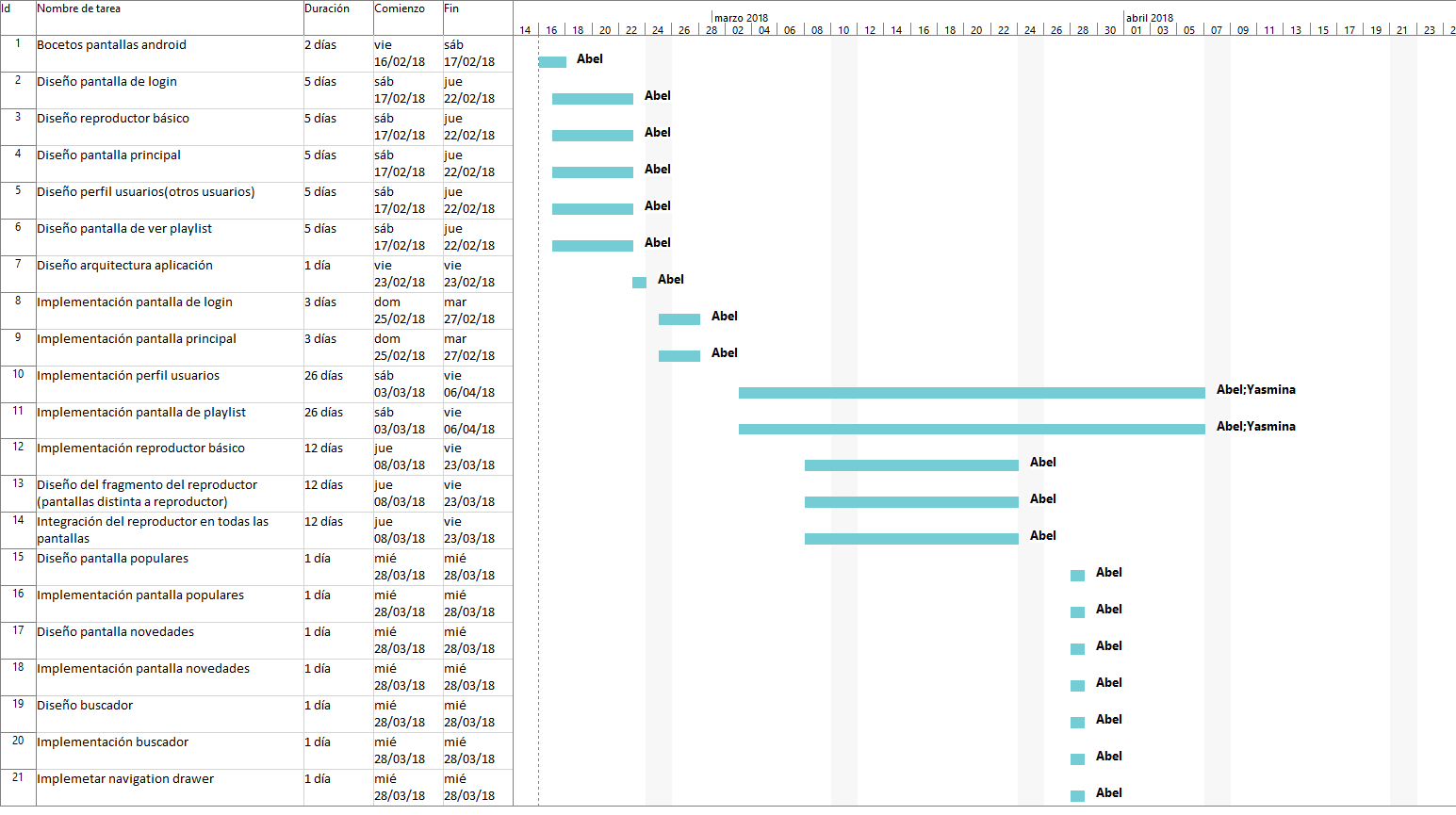
ceph03 192.168.200.12 2002 RGW(HTTP) 7480

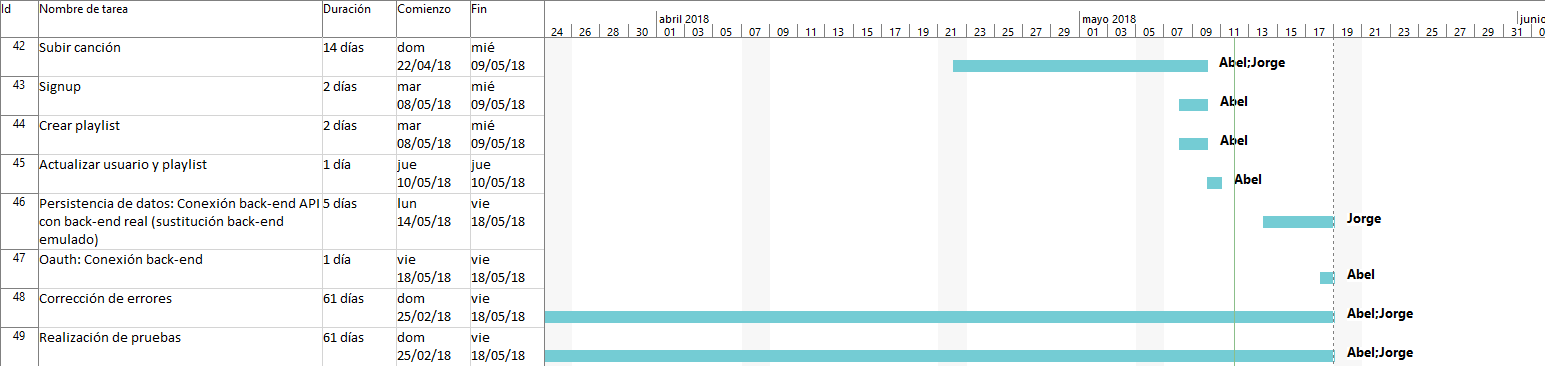
proxmox 192.168.200.20 2005 WEB 8006

backend 192.168.200.16 2006 HTTP 7800 HTTPS 7880

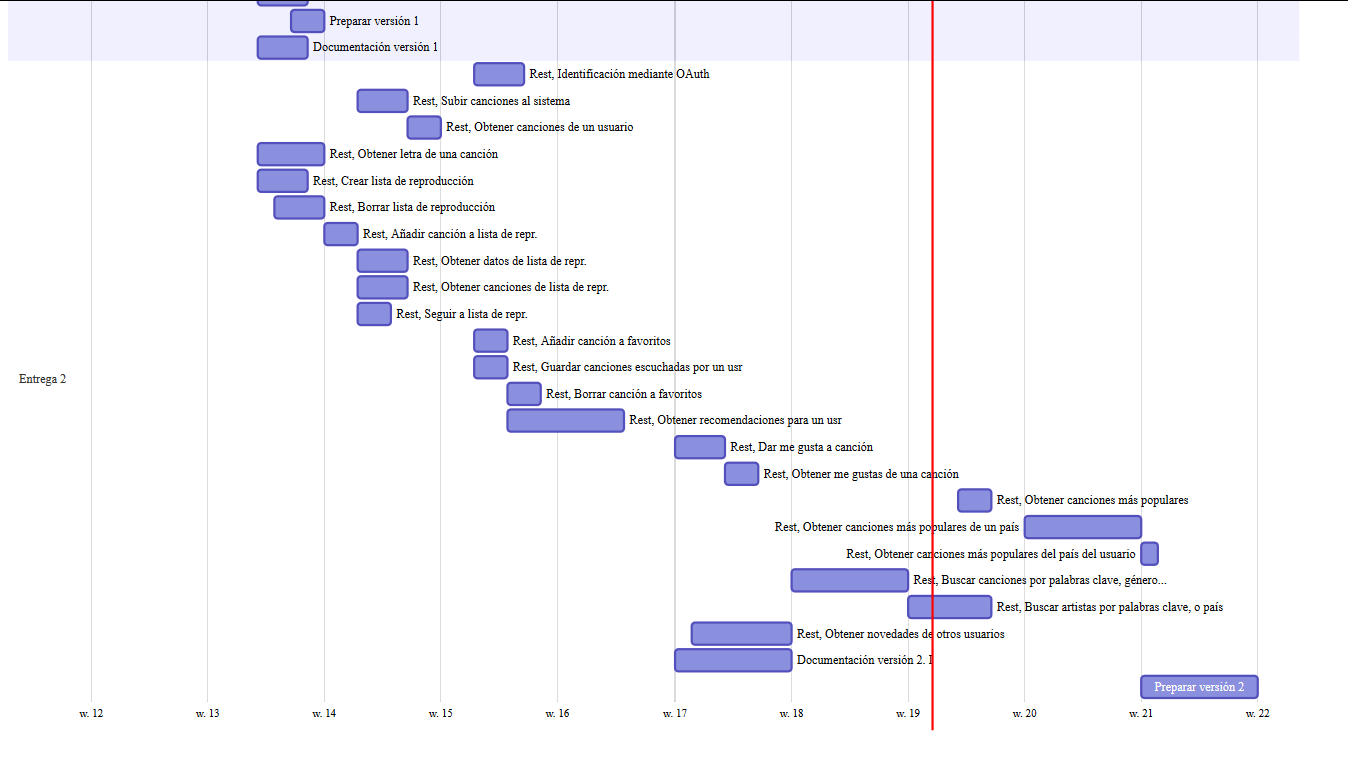
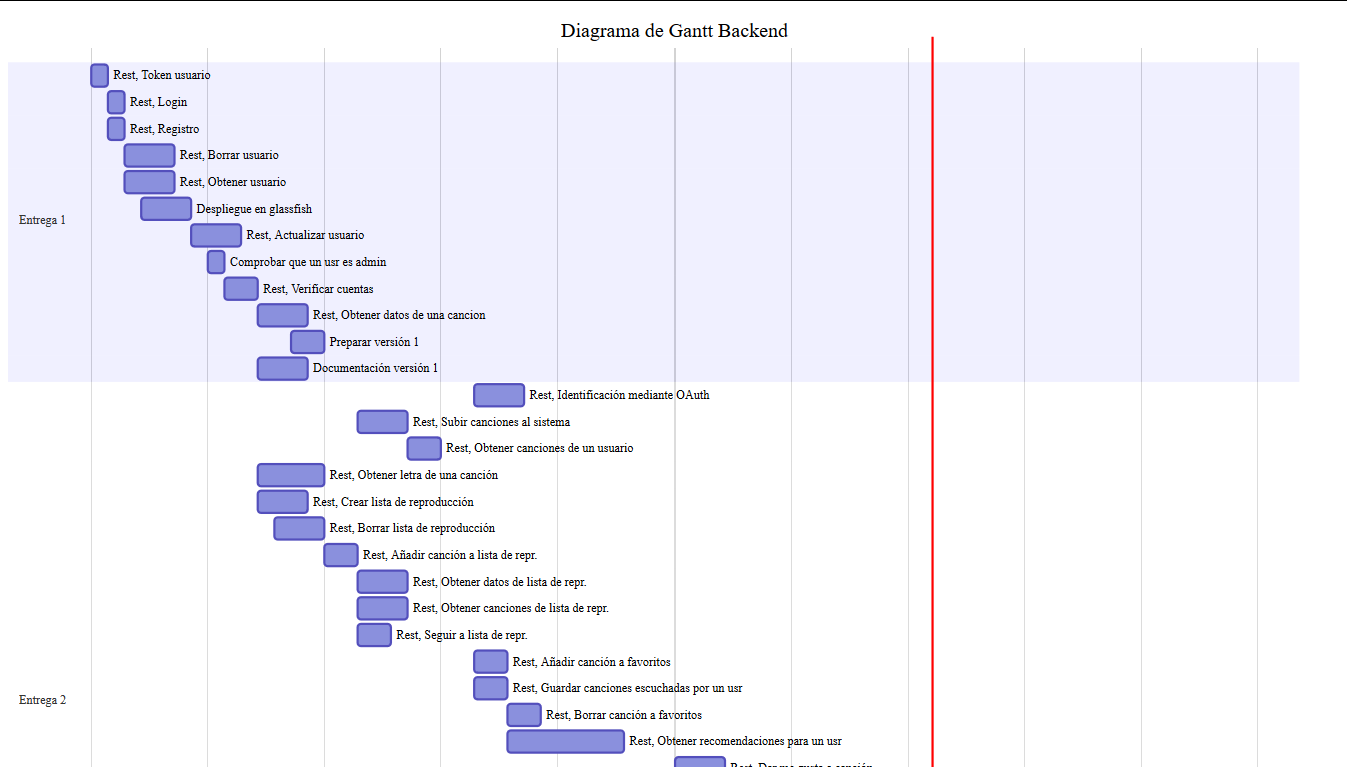
postgres 192.168.200.17 2007 POSTGRES 7850

### Diagrama de Gantt de la aplicación Android

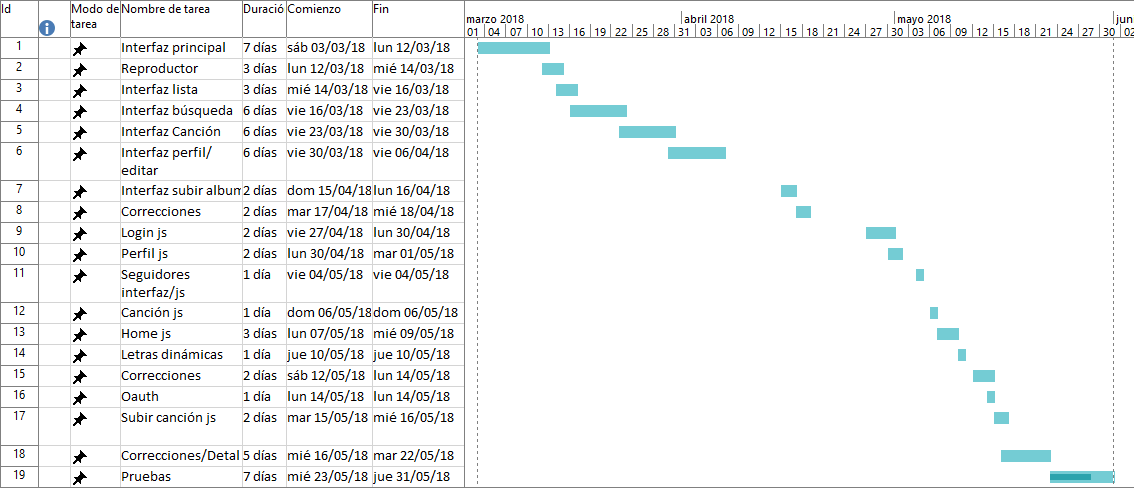
****



### Diagrama de Gantt del Backend

* + 1. 

### Diagrama de Gantt de la aplicación Web

* + 1. 

# Análisis y Diseño del sistema

## Análisis de requisitos

### Sistema: Requisitos no funcionales

El sistema tendrá una versión para Android y otra para Web.

El sistema soporta los ficheros MP3, WAV, OGG.

El sistema dispone de un servidor para el almacenamiento de canciones.

La aplicación móvil soportará Android 5.0 y la Web la última versión de Firefox y Chrome.

### Backend: Requisitos funcionales

1. El sistema se compone de Usuarios, Canciones y Listas de reproducción.
2. El sistema permite 3 tipos de usuarios:

* Usuario registrado (usuario que esté registrado en el sistema)
* Usuario no registrado (que es el cual el sistema no puede identificar con ninguna cuenta)
* Usuario administrador (aquel que es como un usuario registrado pero unos privilegios especiales, como es verificar cuentas)

1. Una canción se compone de un título, un audio, la transcripción de dicho audio, el país de la canción, cuántos me gusta tiene y el número de reproducciones.
2. Una lista de reproducción o categoría es una lista de canciones generadas por el sistema agrupadas por género, éxito, país, situación para la que es propicia o generada por el usuario.
3. Un usuario registrado se compone de un nombre, un nick único (es decir, que dos usuarios no pueden tener el mismo nick), un correo electrónico, una contraseña para acceder a la aplicación, la fecha de nacimiento, una biografía, una foto de perfil, qué canciones ha reproducido, cuántos me gusta han recibido las canciones que ha subido, cuántos me gusta ha dado, sus canciones, el país y las redes sociales que quiera añadir de forma opcional (Twitter, Facebook e Instagram).
4. El sistema permite obtener todos los datos de una canción.
5. El sistema permite que los usuarios se registren en el sistema.
6. El sistema permite que los usuarios de identifiquen tanto con usuario y contraseña así como con una cuenta de Google.
7. El sistema permite que los usuarios registrado registren nuevas canciones en el sistema.
8. El sistema permite que los usuarios registrados creen y borren listas de reproducción formadas por canciones que pueden ser o no del propio usuario, las cuales serán públicas.
9. El sistema permite que los usuarios registrados añadan una canción a una lista de reproducción que haya creado el mismo.
10. El sistema permite obtener todas las canciones de una lista de reproducción.
11. El sistema permite que los usuarios registrados tengan una lista de canciones favoritas, la cual será privada.
12. El sistema permite que los usuarios registrados añadan o eliminen canciones de favoritos.
13. El sistema permite que los usuarios registrados obtengan la lista de sus canciones favoritas.
14. El sistema permite obtener las letras de una canción.
15. El sistema permite obtener recomendaciones (canciones, listas de reproducción y usuarios), canciones más populares, novedades sobre los usuarios o listas de reproducción que un usuario sigue.
16. El sistema permite que los usuarios registrados modifiquen su nombre, su correo electrónico, su contraseña, su fecha de nacimiento, su biografía, su foto de perfil, su país y sus redes sociales.
17. El sistema permite que los usuarios registrados sigan a una lista de reproducción.
18. El sistema permite buscar canciones mediante palabras clave (título), género, país, me gusta, reproducciones o una combinación de varios.
19. El sistema permite ver las canciones más populares de un país.
20. El sistema permite ver las canciones más populares del país de un usuario.
21. El sistema permite buscar artistas mediante palabras clave (nick, nombre y biografía), país o una combinación de varios.
22. Las búsquedas por palabras clave deberán contener al menos una palabra y, si contiene sólo una palabra, que sea de longitud mayor o igual a 3 caracteres.

### Backend: Requisitos no funcionales

1. El sistema permite registrar usuarios cuyo nick esté comprendido entre 4 y 32 caracteres.

### Android: Requisitos funcionales

1. El sistema permite que los usuarios se registren en la aplicación.
2. El sistema permite un solo tipo de usuario, y es el usuario registrado (usuario que esté registrado en el sistema).
3. El sistema permite que los usuarios de identifiquen tanto con usuario y contraseña así como con una cuenta de Google.
4. El sistema permite modificar los datos de usuario una vez creado.
5. El sistema permite que los usuarios suban canciones a la aplicación.
6. El sistema permite crear listas de reproducción formadas por canciones de las que puedes ser autor o no, las cuales serán públicas.
7. El sistema permite editar una playlist propia.
8. El sistema posee una pantalla específica en la que se puede ver las canciones que posee una lista de reproducción así como seguirla.
9. El sistema permite a los usuarios tener una lista de canciones favoritas, la cual será privada.
10. El sistema posee una pantalla en la que se podrán ver las playlist seguidas.
11. El sistema posee una pantalla en la que se podrán ver los usuarios seguidos.
12. El sistema permite crear álbumes musicales.
13. El sistema posee una pantalla de usuario (usuario propio) en la que se puede ver las canciones del propio usuario y sus listas de reproducción creadas, así como añadir más canciones, listas de reproducción o eliminar alguna de las dos.
14. El sistema posee una pantalla de usuario (otros usuarios) en la que se puede ver su información, canciones, listas de reproducción creadas, su número de seguidores, la opción de seguirlo así como un enlace a sus redes sociales.
15. El sistema permite buscar canciones, artistas y listas de reproducción por su nombre mediante un buscador.
16. El sistema posee una pantalla en la que se muestran las canciones más populares de cada género.
17. El sistema posee una pantalla en la que se muestran las canciones más populares en el país desde el que se conecta el dispositivo, las canciones más populares en el mundo y las canciones que son tendencia.
18. El sistema posee una pantalla en la que se mostrarán recomendaciones ( en las que se incluyen canciones, listas de reproducción y usuarios), canciones nuevas que hayan publicado los usuarios seguidos y canciones más populares en el mundo.
19. El sistema posee una pantalla en la que se podrán ver las playlist seguidas que se hayan actualizado y las nuevas canciones que hayan sacado algún artista seguido.
20. El sistema permite desde cualquier pantalla escuchar canciones y ver los controles básicos de canciones (play, pause y pasar canción)
21. El sistema permite que la musica se siga reproduciendo en segundo plano mienstras se usan otras aplicaciones, para ello contará con una notificación desde la que se podrá parar o relanzar la canción que se está escuchando. Desde esta notificación se podrá abrir la aplicación.
22. El sistema permite desde una pantalla específica acceder a más controles sobre las canciones, donde se incluye cambiar el orden de reproducción(aleatorio o lineal) avanzar o retroceder en la canción a un momento exacto de esta, ver la letra de la canción de forma dinámica (la letra se muestra al ritmo de la canción) en el caso que la posea, así como ver la onda de sonido de la canción. También se permitirá desde esta pantalla descargar la canción así como añadirla a favoritos (me gusta).
23. El sistema almacena la sesión para evitar que el usuario tenga que autentificarse cada vez que accede a la aplicación, solamente lo hará una vez.
24. El sistema permite cerrar una sesión para acceder a la aplicación con otro usuario.
25. El sistema posee una pantalla en la que se podrán ver las canciones descargadas y cualquier canción ejecutada desde esta pantalla no consumirá datos (se ejecutará en local).
26. Las canciones descargadas serán comunes a los diferentes usuarios que accedan desde el mismo dispositivo.

### Web : Requisitos funcionales

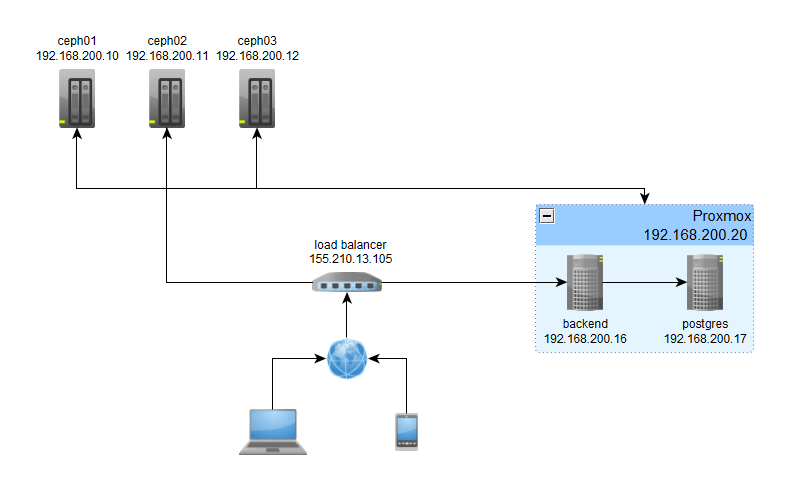
1. El sistema permite registrarse mediante usuario o a través de Google.
2. La reproducción actual de un usuario se sincroniza en todos los dispositivos.
3. Se dispone de 3 perfiles de usuarios: registrados, no registrados y administrador.
4. Los usuarios no registrados pueden acceder a la página principal, páginas de los diferentes usuarios, páginas de canciones además de poder realizar búsquedas y pueden reproducir canciones de usuarios o de listas de reproducción públicas.
5. Los usuarios registrados pueden realizar todas las funcionalidades de un usuario no registrado.
6. El sistema permite a los usuarios registrados identificarse.
7. El sistema permite a los usuarios registrados modificar su información personal.
8. El sistema permite a los usuarios registrados subir canciones.
9. El sistema permite administrar una lista de reproducción privada de favoritos.
10. El sistema permite añadir cualquier canción a favoritos.
11. El sistema permite crear listas de reproducción públicas
12. El sistema permite añadir cualquier canción a una lista de reproducción previamente creada por el usuario.
13. El sistema permite eliminar una canción de sus listas de reproducción o de la lista favoritos.
14. El sistema permite seguir/ dejar de seguir a un usuario.
15. El sistema permite seguir/ dejar de seguir a una lista de reproducción.
16. El sistema permite modificar sus enlaces a redes sociales.
17. El sistema permite al usuario administrador puede verificar la identidad de un usuario.
18. El sistema permite al usuario administrador modificar la información de un usuario, canción o playlist.
19. El usuario administrador se identifica en la aplicación con un usuarios y contraseña predefinidos.

### Web: Requisitos no funcionales

1. El sistema tiene una pantalla principal donde se mostrarán novedades de artistas y canciones, y canciones más populares.
2. La pantalla principal de un usuario registrado contiene recomendaciones de artistas, listas o usuarios en base a sus gusto y novedades de sus artistas/listas seguidos.
3. La pantalla de administrador incluye un buscador para la búsqueda de un usuario, lista o canción y una lista con el resultado de la búsqueda.
4. Cada canción, lista o usuario tiene una pantalla de administrador desde donde el sistema permite al administrador la modificación.
5. El sistema tiene una pantalla de usuario pública para todo el mundo donde aparece su información, canciones, listas de reproducción públicas, número de seguidores y enlaces a sus redes sociales
6. El sistema cuenta en todas las pantallas con un reproductor de audio que incluye información sobre la canción que se está reproduciendo y las opciones de avanzar, retroceder a un momento de la canción; avanzar o retroceder a la siguiente o anterior canción si se estará produciendo una lista parar la reproducción y cambiar el orden de reproducción a aleatorio.
7. El sistema tiene en todas las pantallas un buscador para realizar una búsqueda de canciones, listas y usuarios.
8. El sistema tiene una pantalla para cada canción donde se puede ver la información de la canción, la onda de sonido y si tuviese las letras de forma dinámica en reproducción.
9. El sistema tiene una pantalla para identificarse/ registrarse.
10. El sistema dispone de un sistema de verificación de cuentas. Esto supone un indicador gráfico en el perfil de usuario que anteriormente ha verificado su identidad. La finalidad es garantizar la verdadera identidad del usuario.

## Diseño del sistema

### Diagramas de distribución o despliegue:



### 

### Arquitectura del Backend

El servidor Backend es el servicio que sustenta la funcionalidad del resto de módulos, tanto Web como Android, por lo que deberá ser eficiente y servir las peticiones de la forma más rápida posible. Para ello se hace uso de un complejo sistema de caché que guarde en memoria dos niveles de caché, por un lado una L1 para almacenar los datos de cada transacción en ejecución, y por otro un L2 con los datos más frecuentemente utilizados.

La arquitectura del Backend del sistema está estructurada en 4 capas:

En primer lugar se halla la conexión con la base de datos, la cual será mapeada según el método objeto-relación, haciendo uso de Hibernate.

En segundo lugar, existe un servicio que atiende peticiones REST y permite devolver los datos solicitados de usuarios, canciones y listas en los formatos solicitados (JSON,XML).

En tercer lugar, una capa compuesta por una serie de caches cuya finalidad es optimizar las conexiones con la base de datos.

Por último, una cuarta capa que dispone de las utilidades necesarias para realizar funciones como la generación de cadenas aleatorias, el parseo de datos y comprobaciones sobre los mismos.

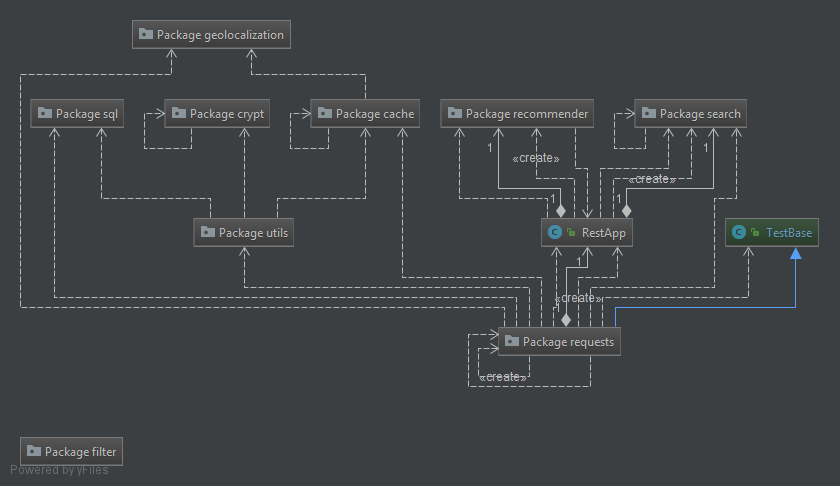
### Resumen de funcionalidades de cada módulo:

El módulo de peticiones Rest está escuchando las peticiones que tanto Web como Frontend le realizan y se comunicará con el resto de módulos para poder servir esas peticiones.

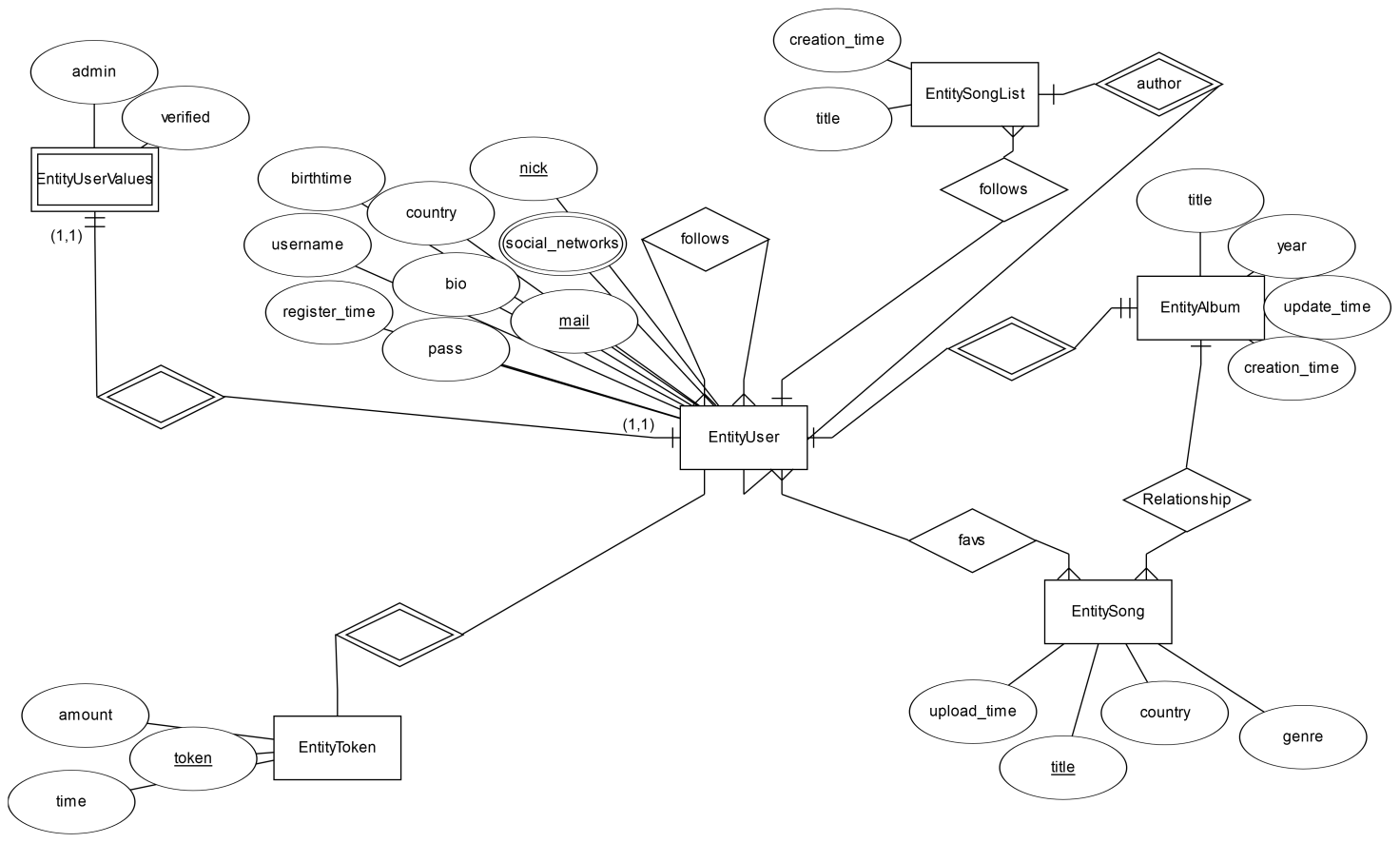
El módulo de Hibernate creará y actualizará tanto la base de datos como el propio contenido de las tablas.

Por último los otros dos módulos simplemente van a realizar operaciones básicas como comprobación de argumentos, generación de cadenas, geolocalización,etcétera.

### Diagrama de paquetes del Backend:



### Diagrama E/R de la base de datos y su mapeo en Hibernate:



### Compendio de la arquitectura de la aplicación Android

La aplicación Android está estructurada en 3 capas (siguiendo una arquitectura modelo-vista-controlador).

En la primera de estas se encuentran las APIs con servicios externos (Backend, Google, Facebook y Twitter) y una clase que se encargará de mantener la persistencia de la sesión (accediendo a ella podríamos obtener información del usuario que está actualmente en sesión). Esta capa permite no depender del desarrollo del backend, creando una interfaz entre ambos que permiten probar funcionalidades no implementadas en el (simulando la respuesta que ofrece el backend).

Existe una segunda capa en la cual se encuentran funciones para manipular la capa citada anteriormente (controlador). En esta tenemos las auth-functions las cuales se encargan de manipular todo lo que tiene que ver con la identificación del usuario (estas funciones son llamadas cuando el usuario inicial sesión). Por otro lado tenemos las player-functions que idealmente sería el punto de acceso al controlador musical, en cambio como se explicará posteriormente actualmente no está operativa. También se poseen las song-request-functions que son las funciones encargadas de solicitar al backend canciones, listas de reproducción (incluyendo listas de recomendados y novedades) e información de usuarios. Por último, tenemos las innner-social-functions que se encargan de las funciones sociales internas de la aplicación (como seguir a un usuario o dar me gusta a una canción) y las extern-social-functions que se encargan del manejo de redes sociales.

Por otro lado nos encontrados la capa de vista, que se encarga del manejo y dibujado de la interfaz gráfica. Esta capa se divide a su vez en dos. Una de ellas se encarga del manejo y dibujado de la interfaz en las pantallas y la otra se encarga de la persistencia de los datos que se están mostrando. Esta segunda capa es requerida para mejorar el rendimiento de la aplicación evitando rehacer determinadas operaciones.

Por último, nos encontramos el controlador musical. Esta capa se encarga de interactuar con el sistema Android para reproducir la música. Se englobaría idealmente en la misma capa en la que se encuentran los servicios externos (comunicación exterior) ya que su funcionamiento es similar y se controlaría mediante el controlador (concretamente mediante las player-functions), pero debido a peculiaridades del diseño en Android y la implementación actual de esta capa, que está muy ligada a la interfaz (en el esquema no se han representado las líneas de comunicación para mejorar su comprensión) hacer esto complicaría la estructura del código a bajo nivel. Aun así esta está sufriendo una reestructuración para hacer posible su encapsulación tras las player-functions.

Asimismo, la aplicación Android es un cliente ligero, ya que necesita conectividad plena para su correcto funcionamiento, aunque para evitar el consumo excesivo de datos se realiza un cacheo de forma automática de las imágenes que aparecen en ella, en esta línea se permite también al usuario descargar las canciones que desee.

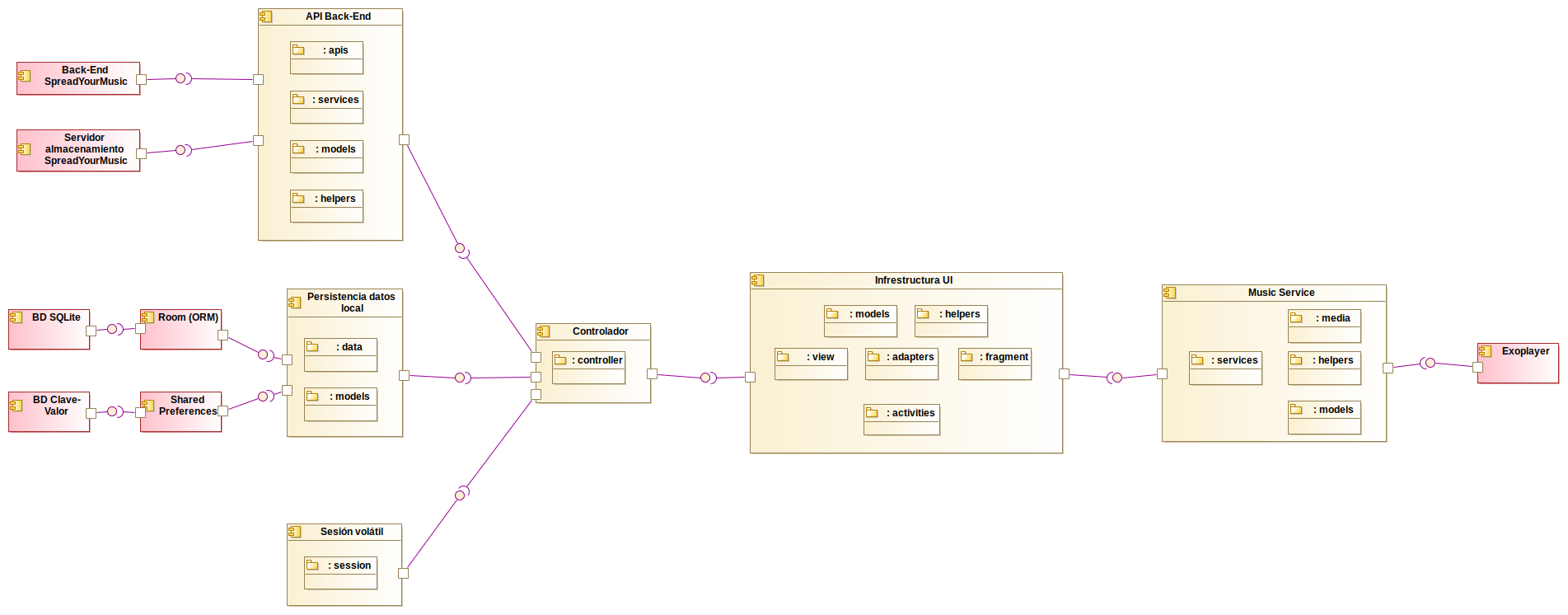
### Versionado de la aplicación Android

La versión 1, primera iteración, es la versión correspondiente a la fecha 31-03 del diagrama de Gantt.

La versión 2, segunda iteración, es la versión correspondiente a la fecha 12-05 del diagrama de Gantt.

Ambas versiones corresponden con los dos 'release' realizados en Github.

### Diagrama estructural de la aplicación Android



La estructura de la aplicación la podríamos ver separada en 4 capas, en las que se sigue un estilo arquitectural modelo-vista-controlador. La capa denominada Infraestructura UI, la cual representa la vista se relaciona con la capa Controlador, la cual representa el controlador y también con la capa Music Service, en esta comunicación el controlador se encuentra incrustado dentro de la capa Infraestructura UI. Por otro lado la capa controlador se relaciona con las capas que ofrecen datos que son API Back-End, Persistencia datos local y Sesión volátil.

Resumen de funcionalidades de cada módulo:

El módulo API Back-End se encarga tanto de la comunicación con los servidores de Back-End en los cuales se realizan peticiones mediante una REST-API, así como con los servidores de almacenamiento en los cuales se suben y eliminan tanto las canciones, sus letras así como las imágenes de perfil de usuario o caratulas.

El módulo persistencia datos local posee dos funciones, la primera es almacenar los datos se la sesión en un almacenamiento persistente, de esta forma tras cerrar la aplicación el usuario sigue manteniendo la sesión iniciada en el dispositivo. Para ello se utiliza un componente que proporciona Android denominado Shared Preferences el cual es una base de datos clave-valor. En esta base de datos se almacena tanto el usuario logueado como el token de sesión que permite identificarse frente al Back-End. Por otro lado, esta capa también se encarga de guardar las canciones descargadas en la memoria física del dispositivo. Para ello se utiliza una memoria no volátil asignada a la aplicación y protegida por el sistema frente a lecturas y escrituras de otras aplicaciones en la cual se guardan los archivos de sonido así como las caratulas de los álbumes a los que pertenecen las canciones. Para guardar la información de las canciones como el nombre, autor, el identificador y otros campos se utiliza un mapeador objeto-relacional llamado Room el cual está implementado sobre la base de datos SQLite que poseen las aplicaciones Android.

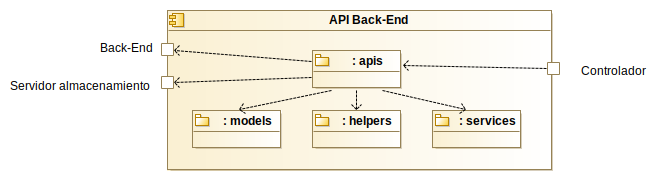
El módulo sesión volátil se encarga de guardar algunos de los datos del usuario para de esta forma evitar tener que hacer llamadas innecesarias al back-end.

El módulo controlador es el nexo entre las capas que ofrecen datos y la capa que controla la interfaz. Las capas que ofrecen datos, todas realizan operaciones muy costosas en tiempo como es comunicarse con otras máquinas o hacer accesos al almacenamiento persistente. Por ello esta capa realiza todas sus operaciones en un hilo de ejecución distinto al principal de forma asíncrona para evitar paralizar el hilo principal que es el que se encarga del control de la pantalla. Para llevar a cabo esto se utiliza el patrón observer, siendo el observador invocado cuando el controlador posee los datos.

El módulo Infraestructura UI es la que se encarga de controlar la vista de la aplicación y la interacción entre todas la pantallas.

Por último el módulo Music Service se encarga de la reproducción de la música así como de la reproducción de las letras de la canción. Esta capa se sustenta sobre la librería Exoplayer la cual ofrece funciones extendidas sobre la librería nativa de audio de Android.

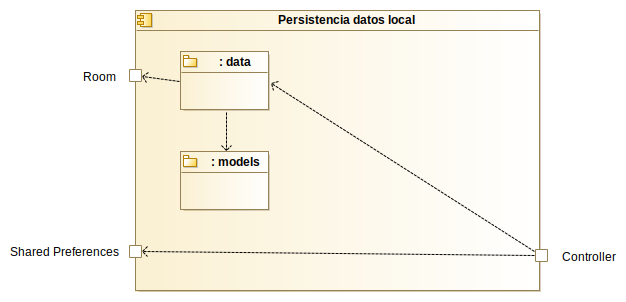
Módulo: API Back-End de la aplicación Android:



El módulo API Back-End recibe peticiones del módulo controlador. Estas peticiones pueden ser de dos tipos. La primera son las que requieren datos, las cuales se resuelven realizando llamadas a la REST-API del Back-End. Para realizar estas llamadas se hace uso del paquete helpers el cual posee unas cuantas utilidades para realizar estas llamadas así como para tratar los datos que estas devuelven. Los resultados de estas llamadas se devuelven como objetos del paquete models, el cual posee el modelo de datos interno de la aplicación.

Por otro lado están las peticiones que requieren subir o eliminar ficheros en los servidores de almacenamiento. Estos servidores utilizan la API de Amazon S3 como forma de comunicación. Para establecer conexiones con esa API, se utiliza una clase auxiliar que se encuentra en el módulo services que se encarga de manejar esta API, debido a que para utilizarla se requiere instanciar objetos que son costosos, encargándose esta clase de hacer esto sólo una vez así como de manejar las conexiones y controlar el estado de las peticiones que se realicen.

### Módulo: Persistencia datos local de la aplicación Android:



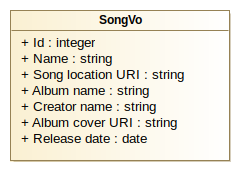
El módulo persistencia datos local posee dos funciones, la primera es almacenar los datos se la sesión en un almacenamiento persistente, de esta forma tras cerrar la aplicación el usuario sigue manteniendo la sesión iniciada en el dispositivo.

Para esta función aunque a alto nivel se realiza dentro del módulo de persistencia de datos, las clases que la implementan se encuentran dentro del paquete controlador. Estas clases controlan el componente denominado Shared Preferences de Android el cual es una base de datos clave-valor.

En ella en el caso de que esté el usuario logueado se almacena tanto el nick de este y el token de sesión que le permite identificarse frente al Back-End.

Por otro lado, para almacenar la información de las canciones descargadas, las que también se gestionan desde este módulo, se utiliza un mapeador objeto-relacional sobre la base de datos SQLite interna de la aplicación llamado Room.

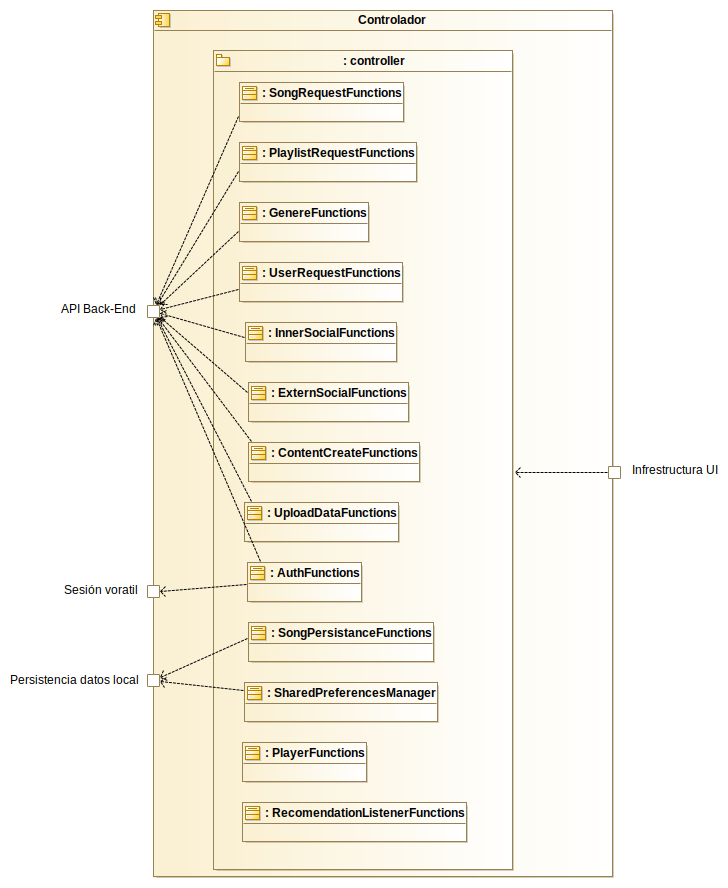
En él se creó una entidad en la cual se almacena toda la información relativa a las canciones y el objeto que la represa es el siguiente:



Además de estos campos de las canciones, se necesita almacenar los ficheros de audio, así como las caratulas de los álbumes a los que pertenecen.

Para ello se hace uso de una memoria no volátil asignada a la aplicación y protegida por el sistema frente a lecturas y escrituras de otras aplicaciones que posee Android denominada Almacenamiento Interno. La diferencia entre guardar las canciones en este almacenamiento y en el almacenamiento externo es que aquí nos aseguramos que los datos de las canciones sólo pueden ser accedidos desde nuestra aplicación forzando al usuario a utilizarla para poder escuchar las canciones descargadas.

### Módulo: Controlador de la aplicación Android:



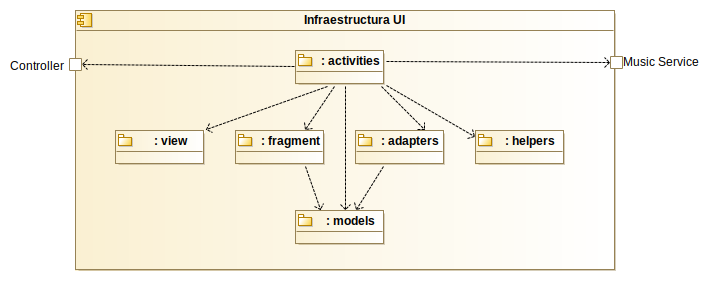
El módulo controlador es el nexo entre las capas que ofrecen datos y la capa que controla la interfaz.

Las capas que ofrecen datos realizan operaciones muy costosas en tiempo debido a que en unos casos realizan peticiones a otras máquinas a través de la red y en otros acceden al almacenamiento persistente del dispositivo. Para gestionar esta situación este módulo realiza todas sus operaciones en un hilo de ejecución distinto al principal de forma asíncrona para evitar ralentizar o incluso paralizar el hilo de ejecución principal que es el que se encarga del control y dibujado de las pantallas. La ralentización o paralización de este hilo produce una mala experiencia de usuario ya que este ve como las pantallas tardan mucho en aparecer y como el teléfono durante unos instantes no responde, además en Android si se mantiene este hilo ocupado durante unos cuantos segundos (actualmente 5), el sistema toma el control de la aplicación mostrando un dialogo en el cual avisa de que la aplicación no es capaz de responder.

Para llevar a cabo esto se utiliza el patrón observer, siendo el observador invocado cuando el controlador posee los datos.

Debido a que para programar la aplicación se ha utilizado el lenguaje Kotlin el cual posee características de programación funcional, el observador que se pasa es una función que será ejecutada cuando se posean los datos. Como esta función es pasada desde la Infraestructura de la UI, y normalmente con estos datos se modificará parte de la interfaz de usuario, aunque la obtención de datos se realiza en un hilo aparte, se fuerza a que esta función se ejecute siempre en el hilo principal de la aplicación que es desde el único que se puede modificar la interfaz de usuario.

### Módulo: Infraestructura UI de la aplicación Android:

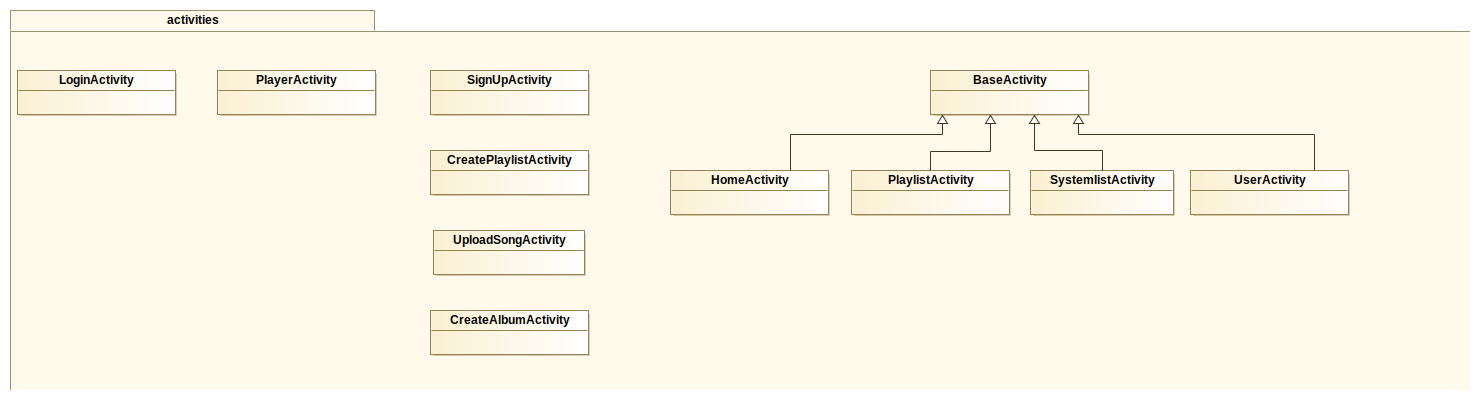


El módulo infraestructura UI es el encargado de controlar la interfaz de usuario y el dibujado de las pantallas. Para ello se utilizan varios paquetes, siendo el principal el paquete activities, en el que cada clase representa al menos una pantalla de Android.

El paquete fragments contiene elementos que permiten dibujar trozos de pantallas de Android. Aquí se han incluido elementos recurrentes en las pantallas.

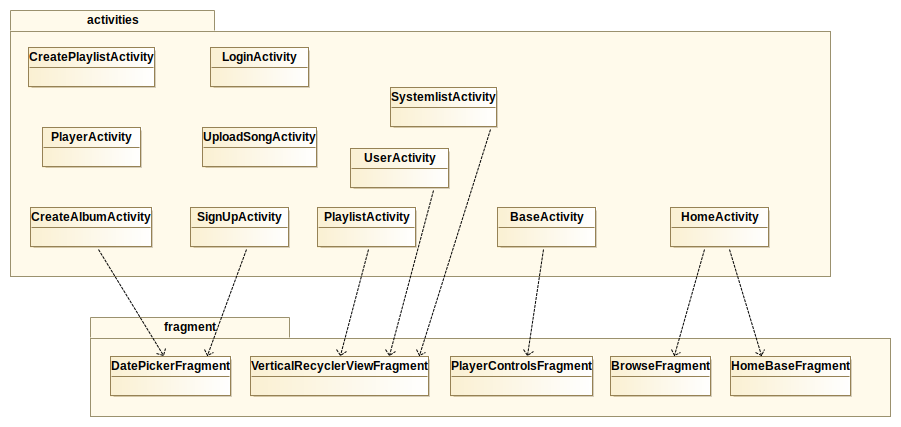
El paquete adapters contiene elementos que permiten dibujar las listas.

Por último el paquete view contiene los elementos que permiten dibujar la onda de sonido circular y la barra de búsqueda circular.

 Las activities las podríamos diferenciar en dos clases, desde las que se puede reproducir música y desde las que no. Desde las que se puede reproducir música todas extienden de BaseActivity salvo PlayerActivity, que es la encargada de controlar el reproductor en pantalla completa.

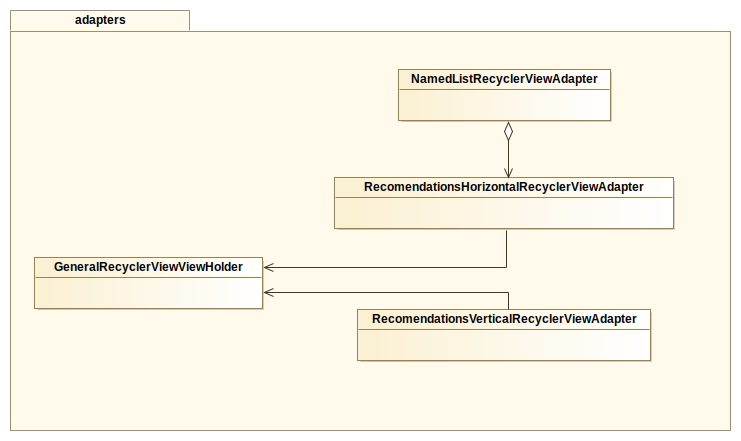
Esto es debido a que en BaseActivity se encuentra incrustado el controlador que permite realizar las comunicaciones con Music Service, de esta forma si se desea reproducir música en una nueva pantalla, solamente ha de extender de BaseActivity y añadir unas pequeñas modificaciones al definir el XML desde el que se dibuja su pantalla.

En el caso de PlayerActivity, el controlador se incrusta dentro de la propia actividad debido a que las comunicaciones que realiza con Music Service son más avanzadas que el resto de pantallas.



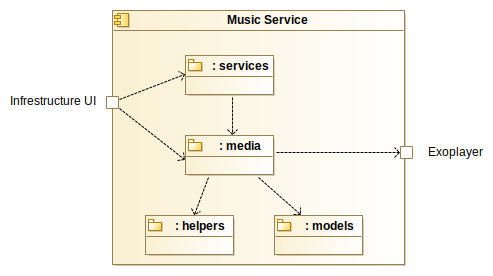
Para simplificar el desarrollo de la interfaz se ideo una estructura de fragments, para construir cada activity. Un fragment permite dibujar un trozo de una pantalla, lo que posibilita en elementos recurrentes en distintas pantallas crear un objeto que pueda ser reusable.

En esta arquitectura el caso que mejor ejemplifica esto es el de HomeBaseFragment. Con ese mismo fragment se consigue crear la pantalla principal, la pantalla de novedades, la pantalla de tendencias y la pantalla de canciones más populares de cada género, siendo creada cada pantalla a partir de la instanciación del objeto y una llamada al módulo controller.

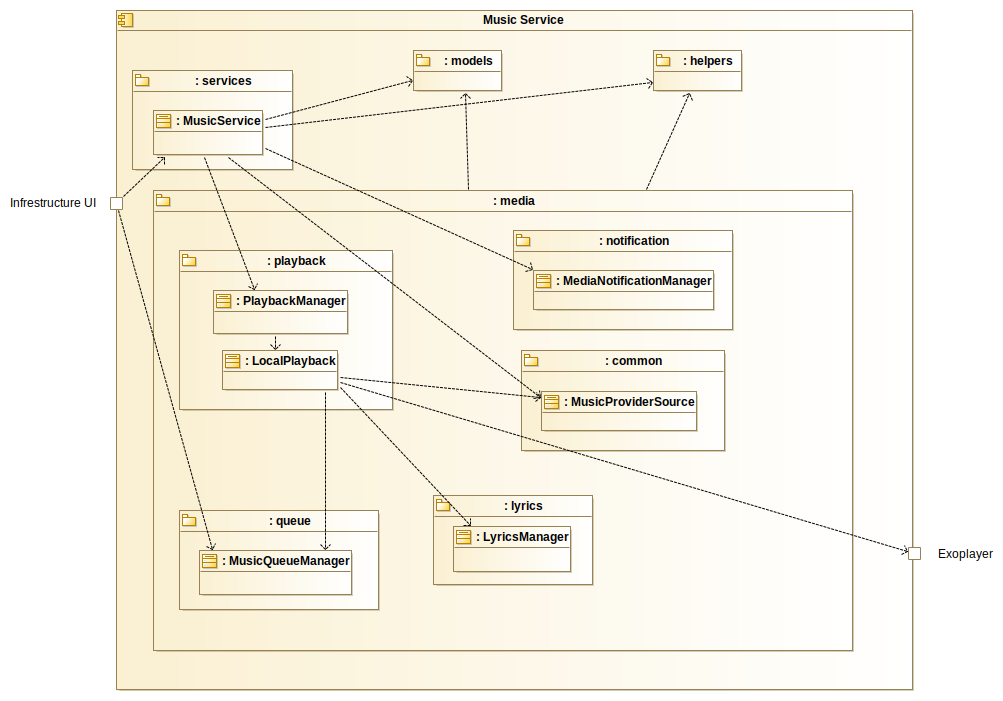


Otro elemento que también permite ser rehusados con los adaptadores para listas, debido a que todas las listas se muestran con el mismo estilo, solamente varía en el caso de que sea lista horizontal o lista vertical.

### Modulo: Music Service de la aplicación Android:



El módulo Music Service se encarga de la reproducción de música en el dispositivo, para ello hace uso de la biblioteca 'Exoplayer' en su versión 2.7, además también aporta el mecanismo que permite mostrar las letras de forma dinámica.



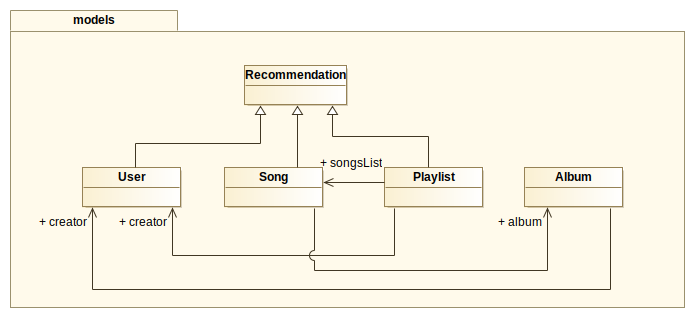
Este módulo posee dos interfaces, la primera es la que se accede mediante el paquete queue. Esta se utilizará para añadir o eliminar canciones en la lista de reproducción en curso y para activar o desactivar la reproducción aleatoria, con esta interfaz se interactúa de forma síncrona.

La otra interfaz se encuentra en el paquete 'services' con la cual se interactúa de forma asíncrona mediante un modelo cliente-servidor. Esta otra interfaz es la que permite obtener las letras de forma dinámica y usar los controles de manejo sobre las canciones como son reanudar canción , parar canción, siguiente canción, canción anterior y avanzar en una canción hasta un momento exacto.

Esta interfaz internamente se comunica con el paquete playback que es el que sirve de interfaz para realizar el manejo del sonido y las letras, siendo esta la que se comunica con 'Exoplayer'.

Por otro lado, desde este mismo módulo también se controla la creación y gestión de la notificación que aparece cuando se está reproduciendo música con la aplicación en segundo plano mediante el paquete 'notification'.

### Modelo de datos de la aplicación Android::



El modelo de datos interno de la aplicación está compuesto por 4 clases que representan cada uno de los elementos básicos sobre los que está asentada la aplicación que son Usuarios, Canciones, Playlist y Albums. Los 3 primeros por cuestiones de implementación extienden de la clase Recommendation, esto permite que muchos elementos como son los adaptadores de las listas se puedan crear de forma genérica y no haga falta crear un adaptador para cada tipo de dato, solamente dentro del adaptador en este caso se comprobaría a qué tipo de dato hay que adaptar la lista, esto complica un poco la lógica de estos elementos pero reduce drásticamente el número clases, así mejorando el mantenimiento del código.

### Diagrama de paquetes de la aplicación Android:



Estructura vista aplicación:

Además del módulo Infraestructura UI, la vista de la aplicación se genera a partir de unos ficheros XML donde se definen los elementos que aparecen en cada pantalla. Estos ficheros han sido desarrollados siguiendo las guías de diseño establecidas por material design, además de haber sido creados de forma responsive para que puedan adaptarse a cualquier tamaño de pantalla. En ella las imágenes e iconos utilizados han sido los proporcionados por Google, los cuales cumplen este estilo de diseño.

### Pantallas de la aplicación Android

LogIn

|  |  |
| --- | --- |
|  |  |

A la izquierda se puede ver la pantalla de LogIn, esta es la primera pantalla que se abre en caso de que el usuario no esté logueado ya, y a la derecha la misma pantalla tras pulsar en la 'G' (LogIn con cuenta personal de Google+), para realizar identificación mediante OAuth.

Con esta pantalla se satisfacen los requisitos nº2 y nº3.

Registrarse

|  |  |
| --- | --- |
|  |  |

La pantalla de registro está compuesta de dos pasos.

Con esta pantalla se satisface el requisito nº1.

Principal

|  |  |
| --- | --- |
|  |  |
|  |  |

A esta pantalla es la primera que se accede después de identificarse el usuario o en el caso de que el usuario se haya identificado anteriormente la aplicación se abrirá en esta pantalla.

La primera de las imágenes es la pantalla en sí. En la segunda se muestra como el reproductor se puede controlar desde cualquier pantalla y en la tercera se muestra un panel lateral de navegación que se usa para desplazarse entre distintas pantallas.

Con esta pantalla se satisface el requisito nº18.

El requisito nº20 se satisface al introducir el reproductor en cualquier pantalla (en el resto de pantallas se muestra de la misma forma).

Canciones más populares de cada género

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº26.

Novedades

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº19.

Tendencias

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº17.

Buscador

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº15.

Playlist seguidas

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº16.

Usuarios seguidos

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº11.

Canciones que me gustan

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº9.

Canciones descargadas

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº25.

Mi perfil

|  |  |
| --- | --- |
|  |  |
|  |  |

En la primera pantalla se muestra el perfil de usuario, en la segunda se muestra la misma pantalla con el menú de opciones expandido, en la tercera se muestra la opción de eliminar cuenta y en la última se muestra la opción de eliminar canción o playlist (esta se produce al mantener pulsado una canción o una playlist).

Con esta pantalla se satisface el requisito nº13.

Reproductor

|  |  |
| --- | --- |
|  |  |

En la primera pantalla se puede ver el reproductor, desde al cual se pueden llevar a cabo varias acciones así como también se puede visualizar la onda de sonido sobre este. En la segunda pantalla se muestra la letra de la canción generada de forma dinámica.

Con esta pantalla se satisface el requisito nº22.

Ver playlist

|  |  |
| --- | --- |
|  |  |
|  |  |

Con esta pantalla se satisface el requisito nº8.

Ver playlist (propia)

|  |  |
| --- | --- |
|  |  |

Ver usuario

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

En las 4 primeras pantallas se muestra un usuario no verificado, en la quinta se muestra un usuario verificado y en las dos últimas se muestra un usuario no verificado que posee cuentas de Twitter, Instagram y Facebook asociadas a la aplicación.

Con esta pantalla se satisface el requisito nº14.

Editar perfil

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº4.

Crear álbum

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº12.

Subir canción

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº5.

Crear playlist

|  |  |
| --- | --- |
|  |  |

Con esta pantalla se satisface el requisito nº6.

Editar playlist

|  |  |
| --- | --- |
|  |  |

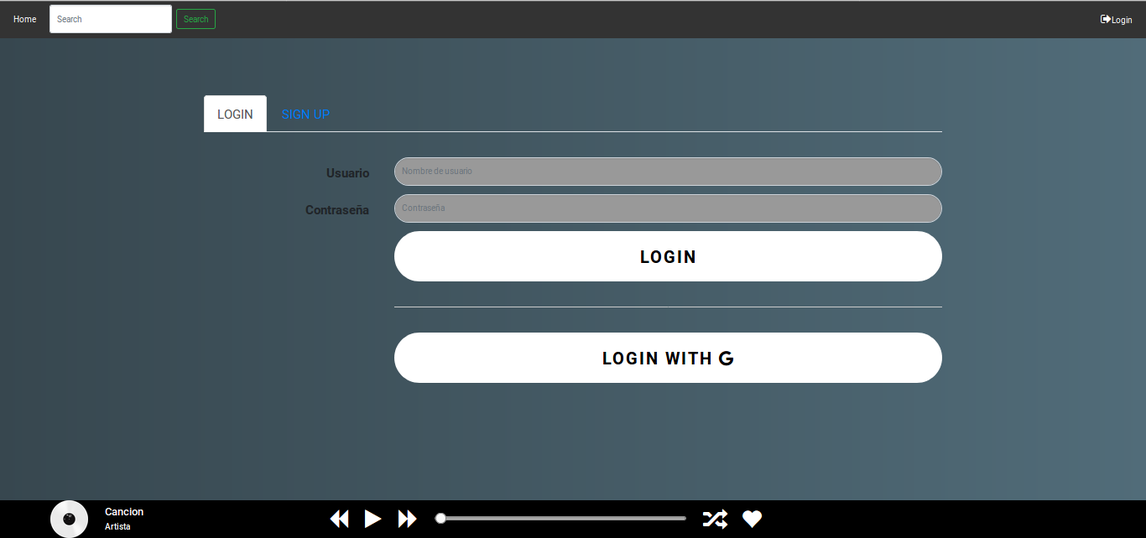
Con esta pantalla se satisface el requisito nº7.

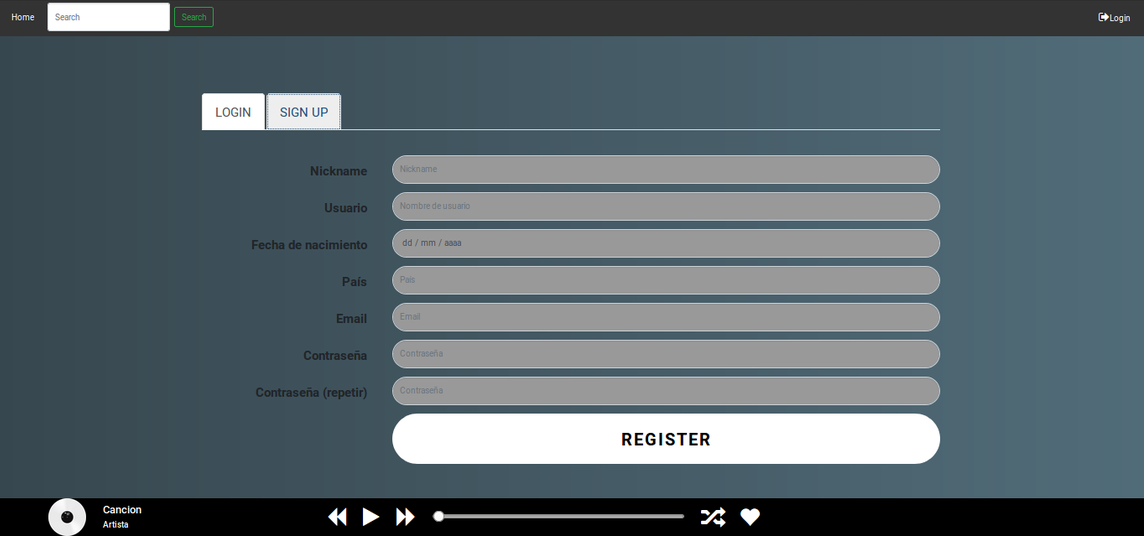
### Arquitectura y pantallas de la aplicación Web

La página Web está estructurada de la siguiente manera:

Existe una página 'Home' desde la cual podemos ubicarnos para acceder al resto de las pantallas web, donde encontraremos las diferentes búsquedas (canciones, listas, artistas), las diferentes listas de reproducción del usuario, el perfil del artista, las canciones para reproducir, la opción de editar el perfil del artista, de editar sus propias listas, subirlas, y en el caso de ser el administrador del sistema, una página especial desde la cual se puede acceder a las diferentes opciones de administrador.

PANTALLAS LOGIN / REGISTRO

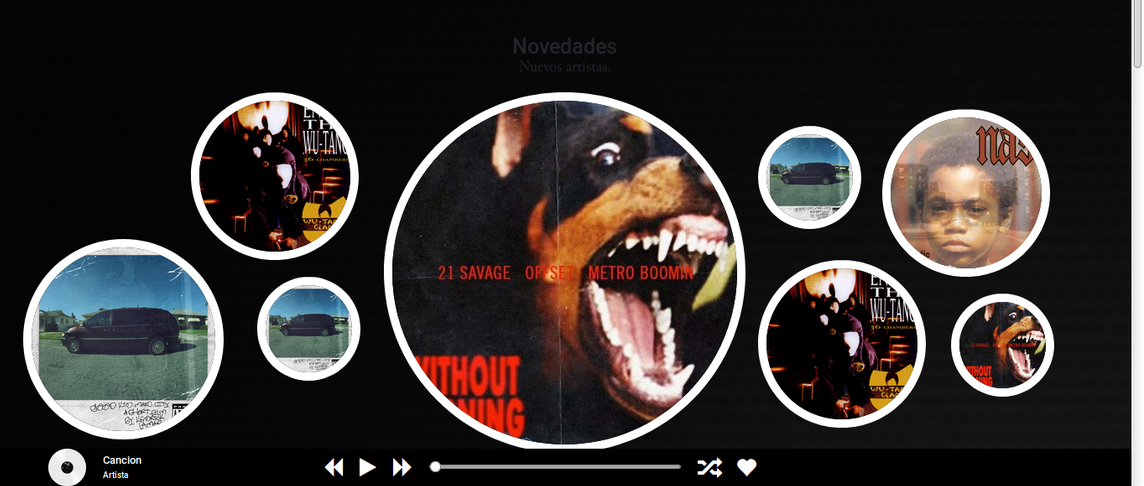


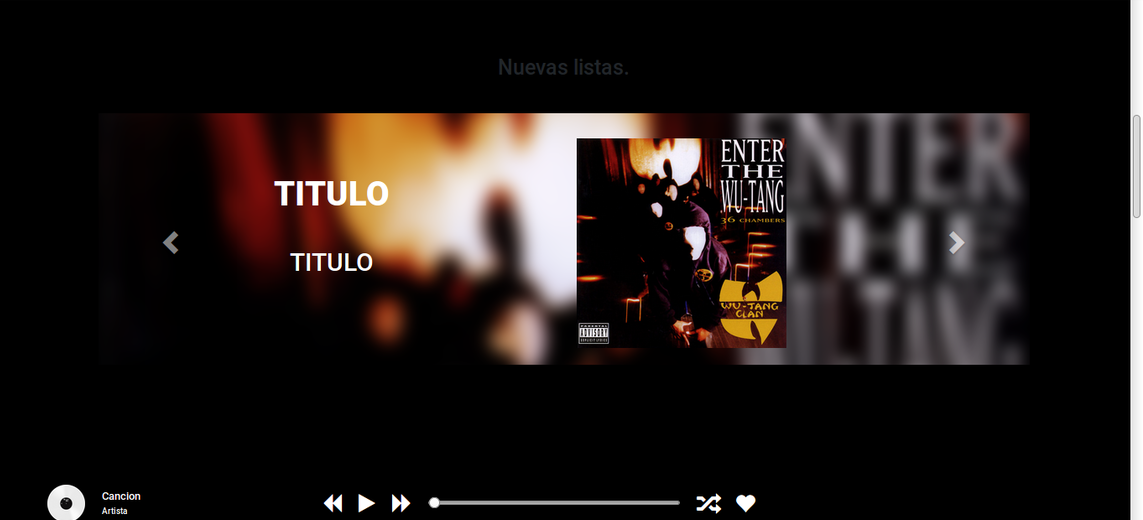


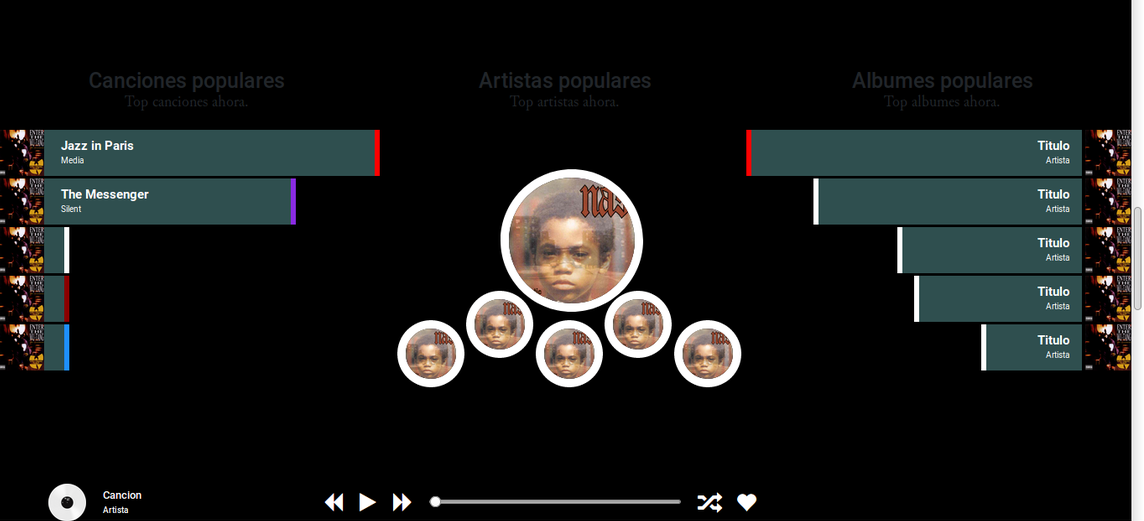
La primera pantalla permite hacer login con el usuario registrado o mediante Google. La segunda es la pantalla de registro (requisito 1, requisito 6).

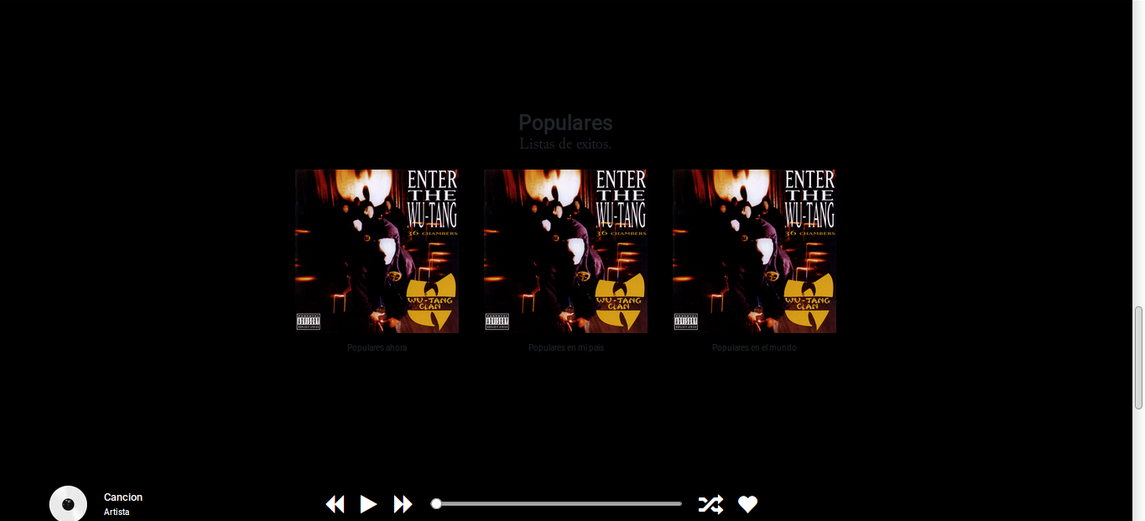
PANTALLA INICIO

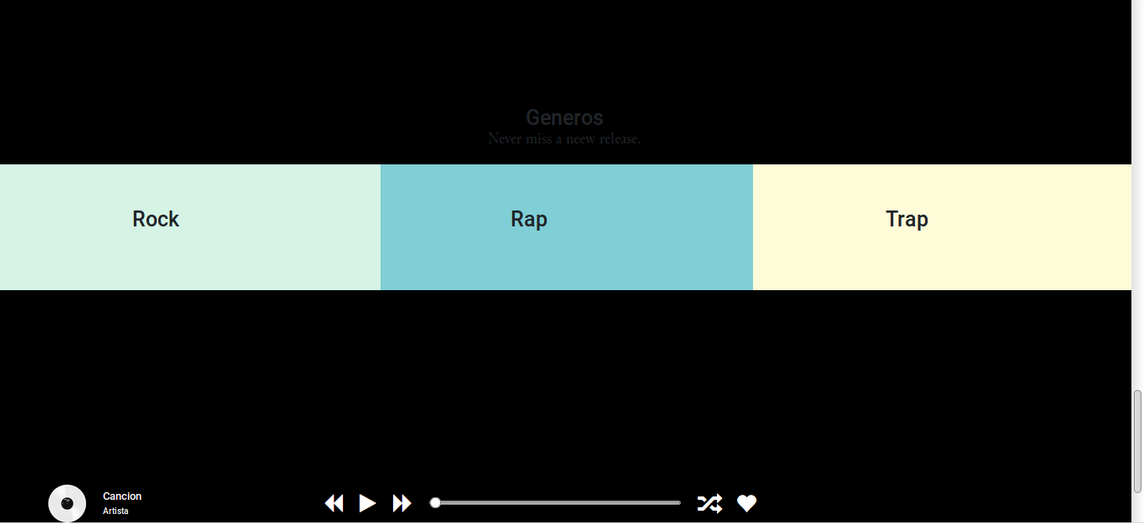
(Capturas de pantalla de las secciones de una misma página)

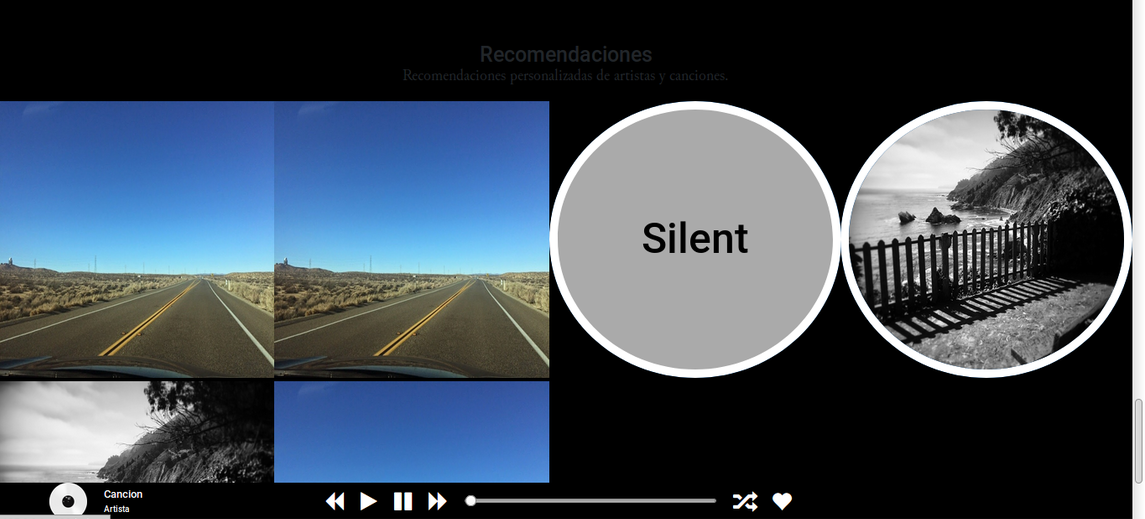








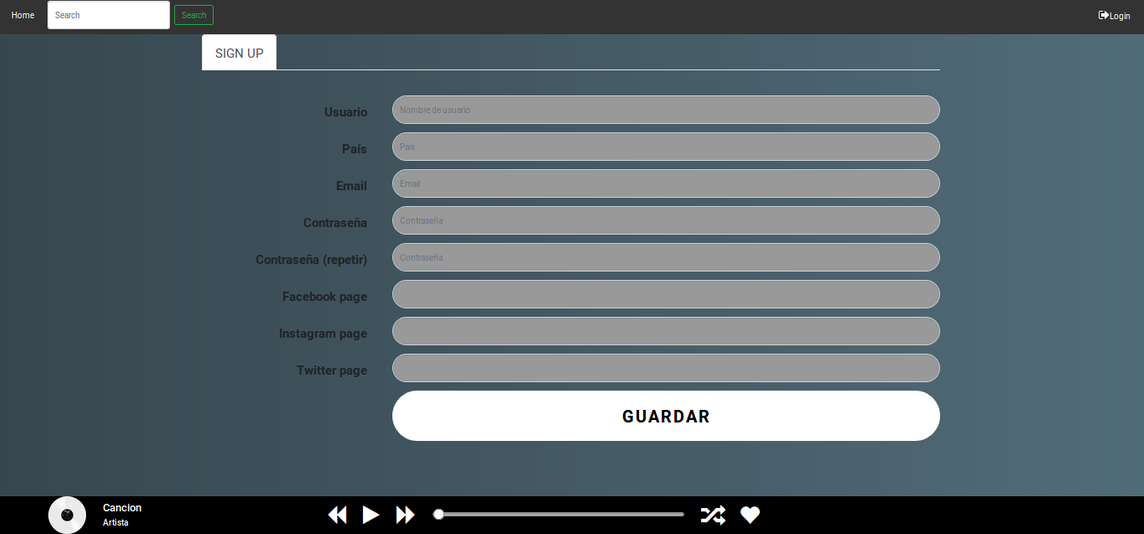




A la página de inicio pueden acceder todos los usuarios. Los usuarios registrados tienen además una sección “Recomendaciones” en la página principal (requisito 4, requisito NF2).

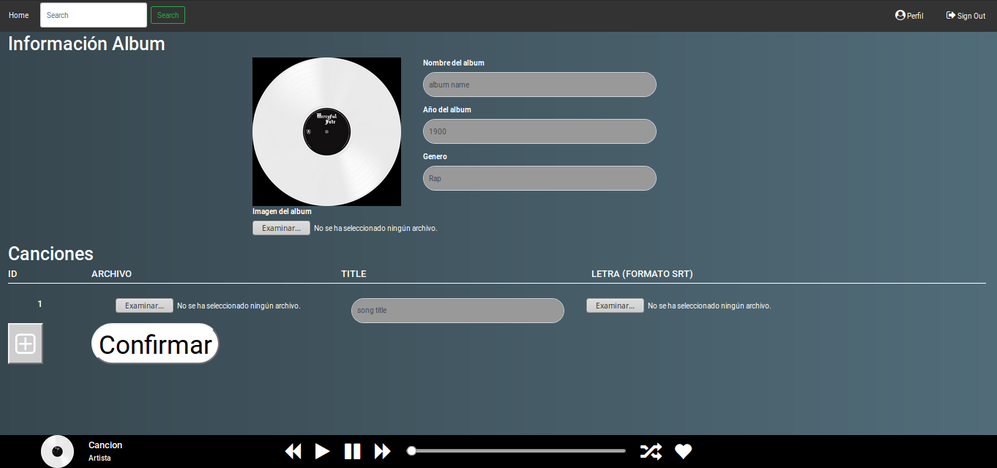
Esta página incluye novedades; canciones, álbumes y artistas populares; listas de éxitos actuales o del país; listas de éxitos según genero y recomendaciones. (requisito NF1)

PÁGINA EDITAR INFORMACIÓN USUARIO



Esta página permite a los usuarios registrados editar su información de perfil (requisito 7). Entre la información que se puede editar se encuentran los enlaces a las redes sociales (requisito 16).

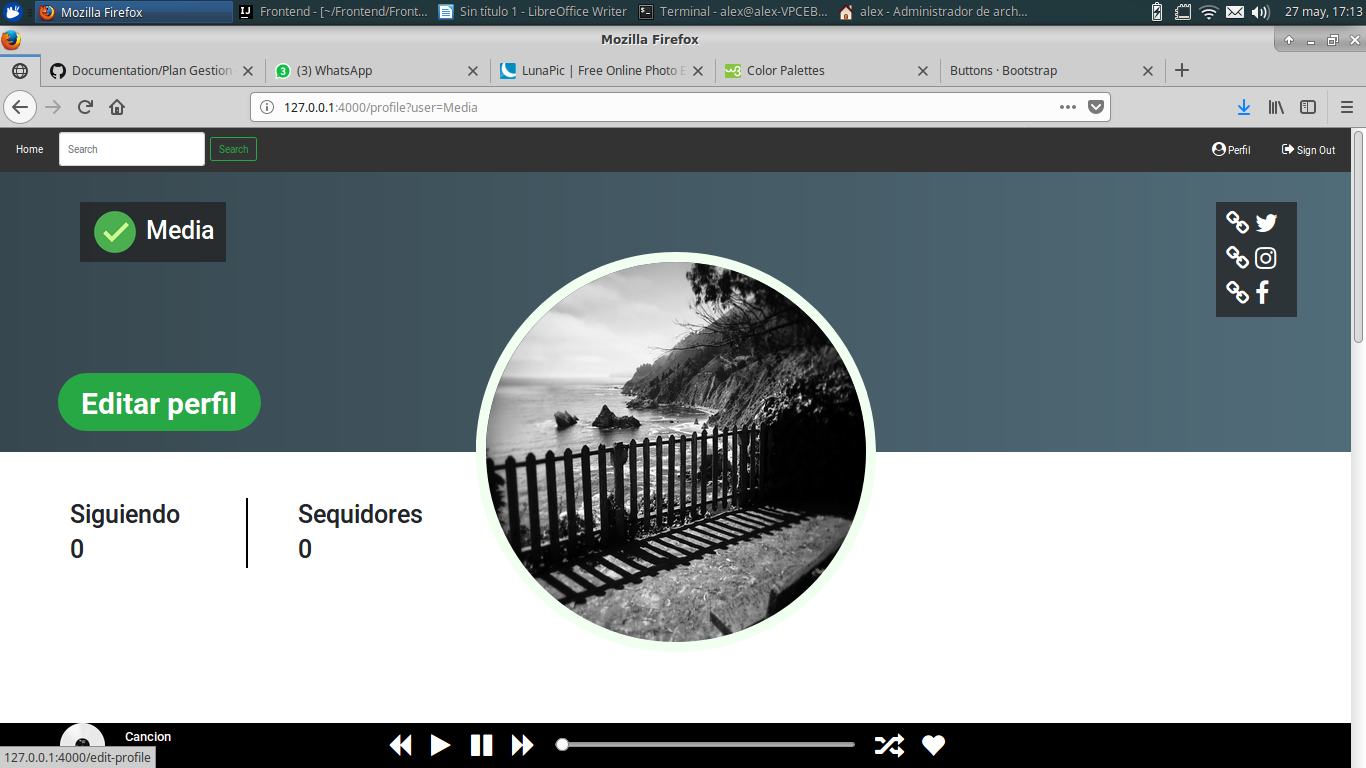
PÁGINA SUBIR ALBUM

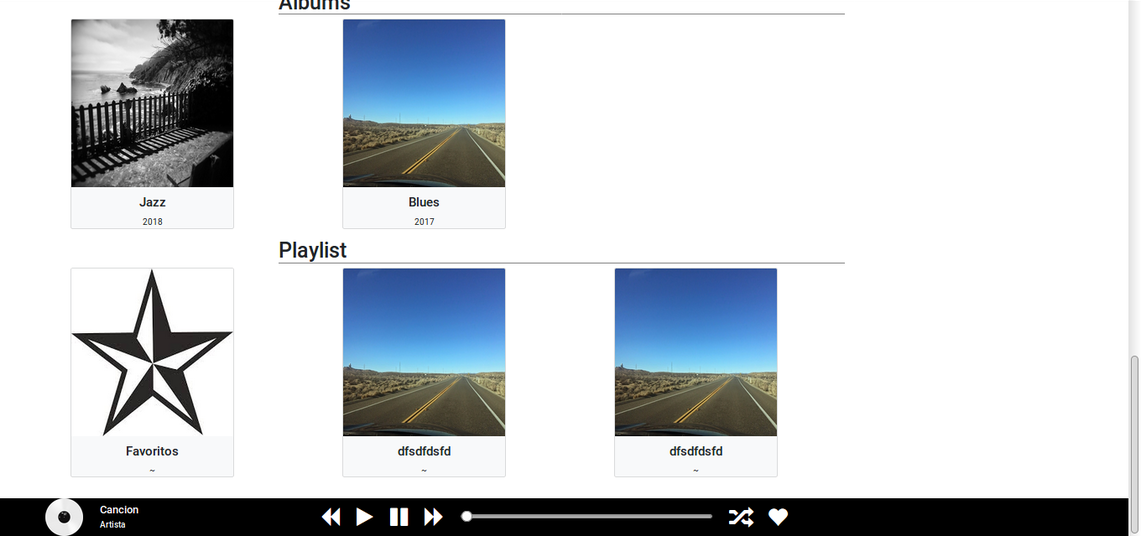


Esta página permite a los usuario registrados subir un álbum con diferentes canciones. (requisito 8).

PÁGINA PERFIL

(Capturas de pantalla de las secciones de una misma página)



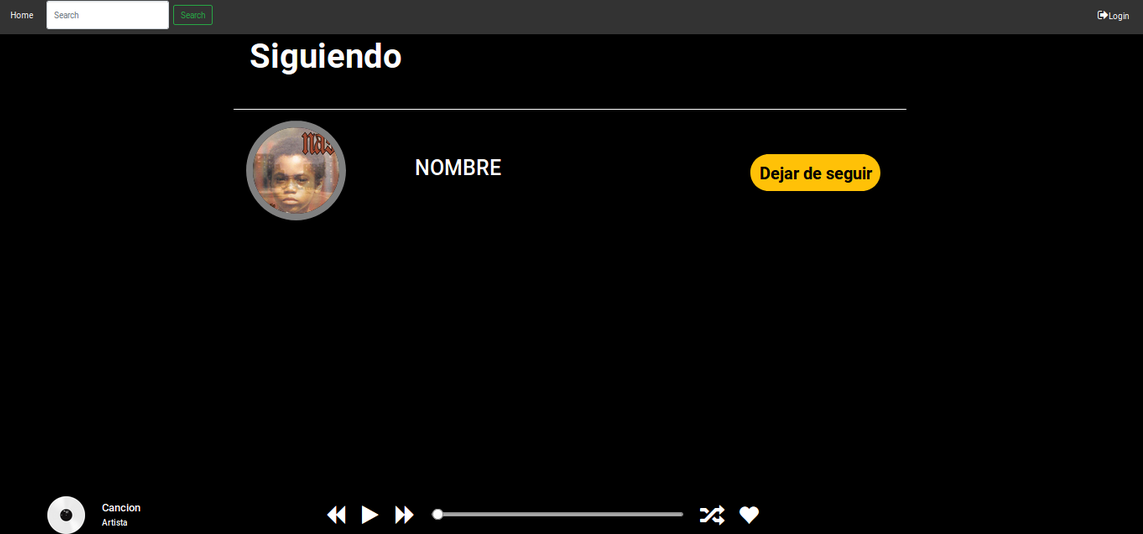


La página incluye un enlace a la lista de favoritos (requisito 9).

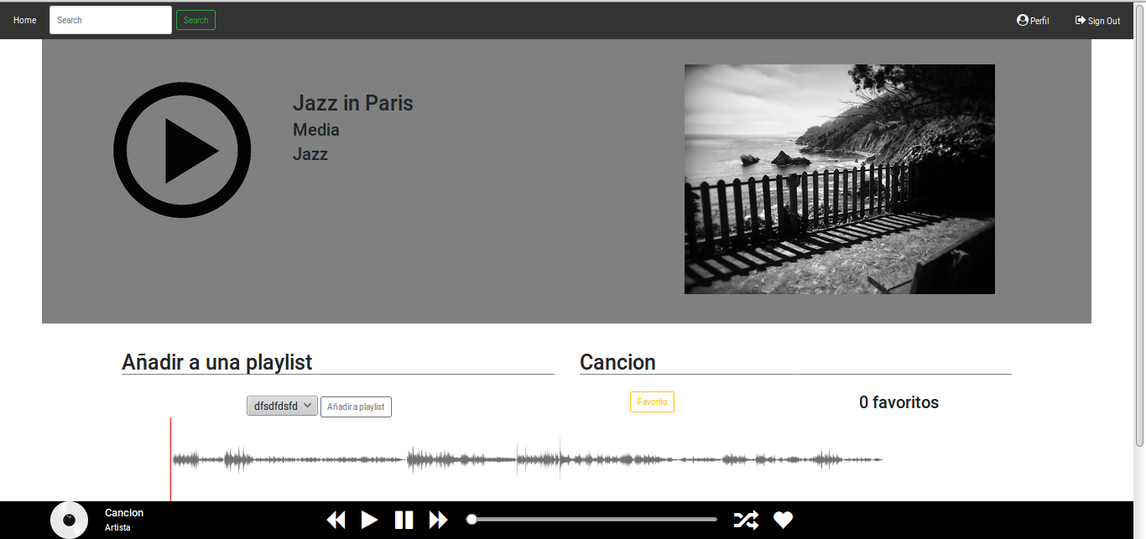
Cuando el perfil no es el del propio usuario el botón con la opción “Editar perfil” desaparece. Por otra parte se hace visible el botón “Seguir” que permite seguir a dicho usuario (requisito 14).

Esta pantalla es pública para todo el mundo (requisito NF5).

PANTALLA SEGUIDORES / SEGUIDOS

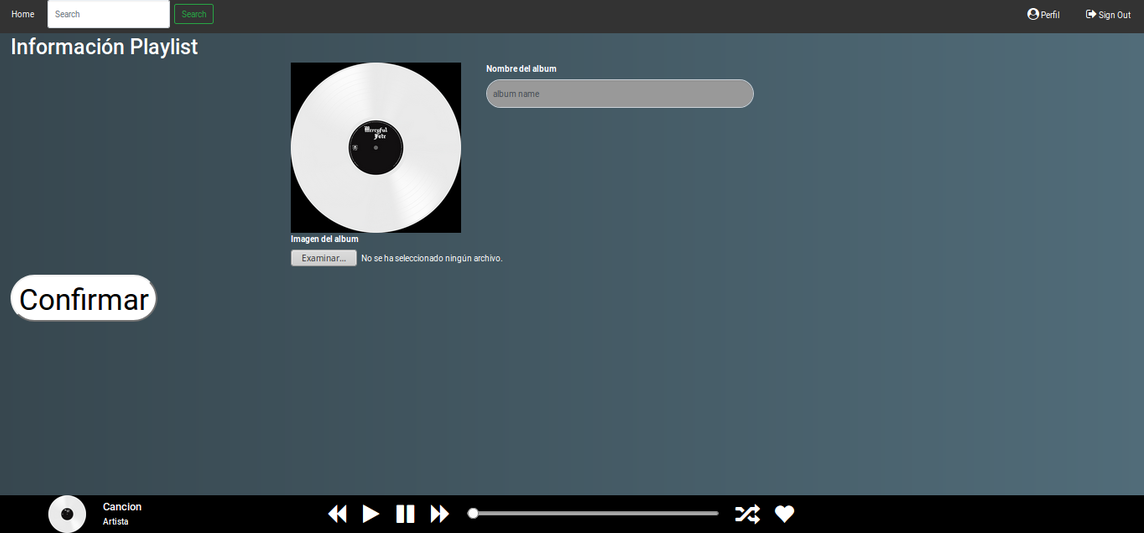


PÁGINA CANCIÓN



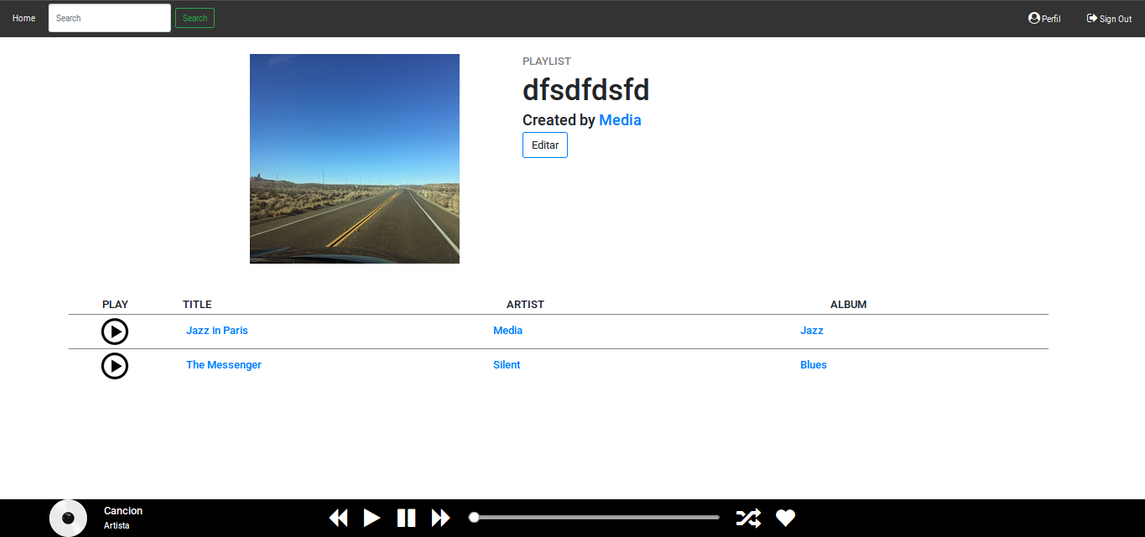
La página de canción permite añadir la canción a favoritos. (requisito 10). Se permite también añadir una canción a una lista de reproducción ya creada (requisito 12). La parte inferior incluye el espectro de onda de la canción. En caso de tener un fichero de letras, la canción al reproducirse las muestra de forma dinámica. (requisito NF8).

PÁGINA CREAR PLAYLIST



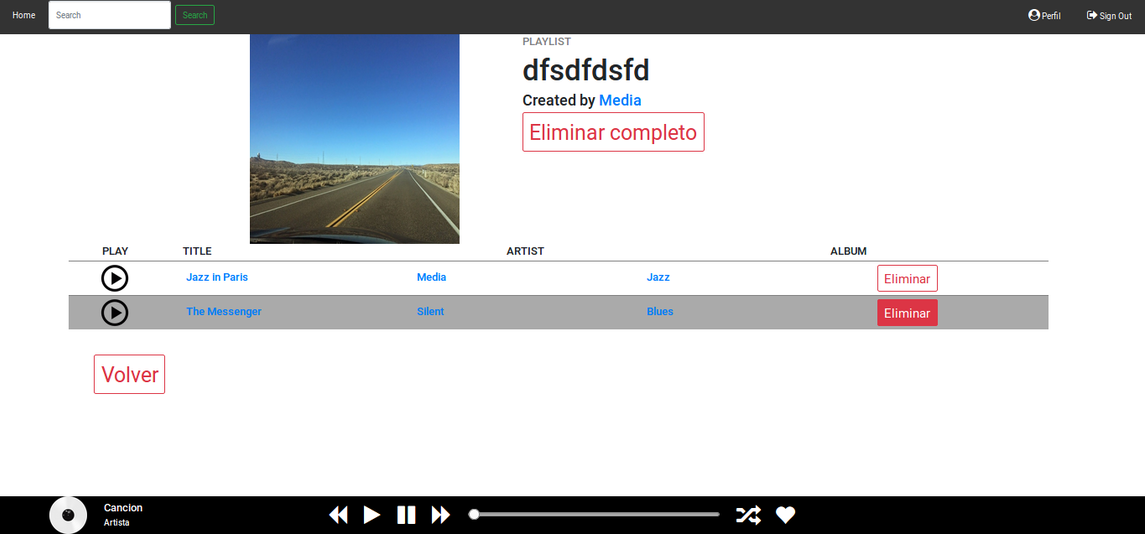
La página permite crear una playlist pública (requisito 11).

PÁGINA LISTA DE REPRODUCCION (ALBUM/ GENERO/ FAVORITOS)



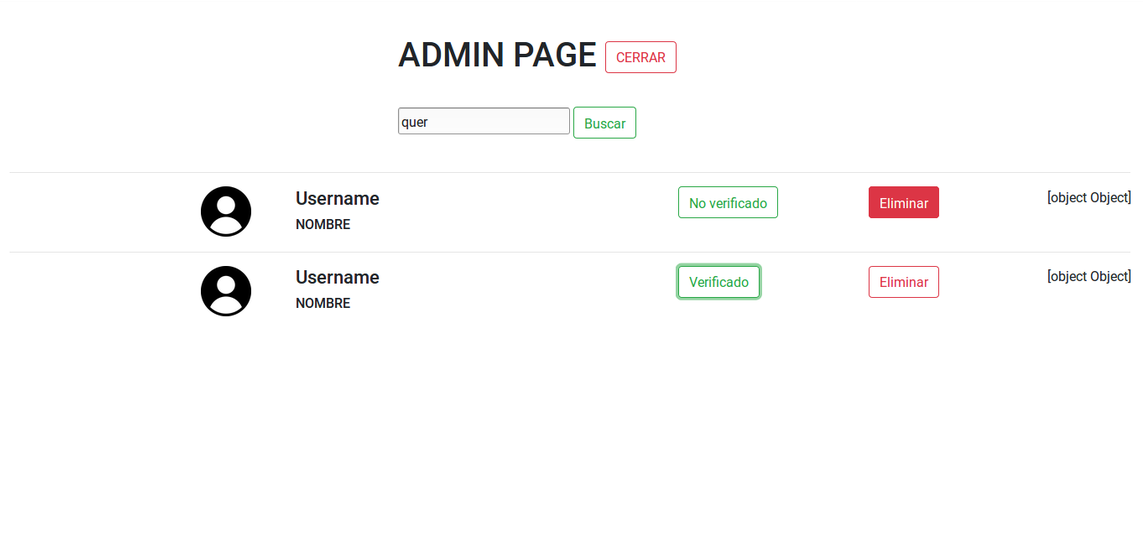
La lista tiene un botón “Seguir” o “Editar Lista” según se es el propietario de la lista o no (requisito 15).

PÁGINA EDITAR LISTA



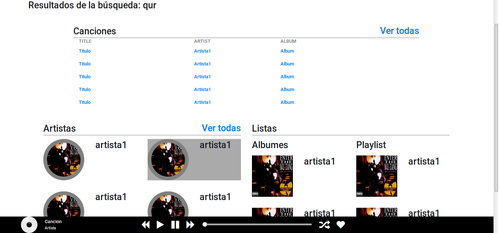
La página permite eliminar canciones de una lista o eliminar la lista entera (requisito 13).

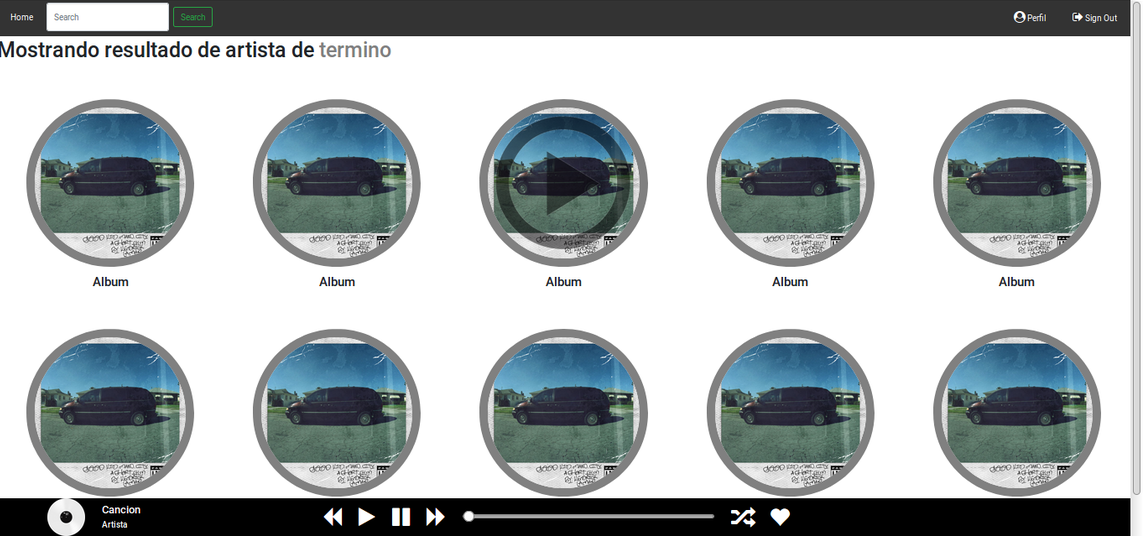
PÁGINA ADMINISTRADOR

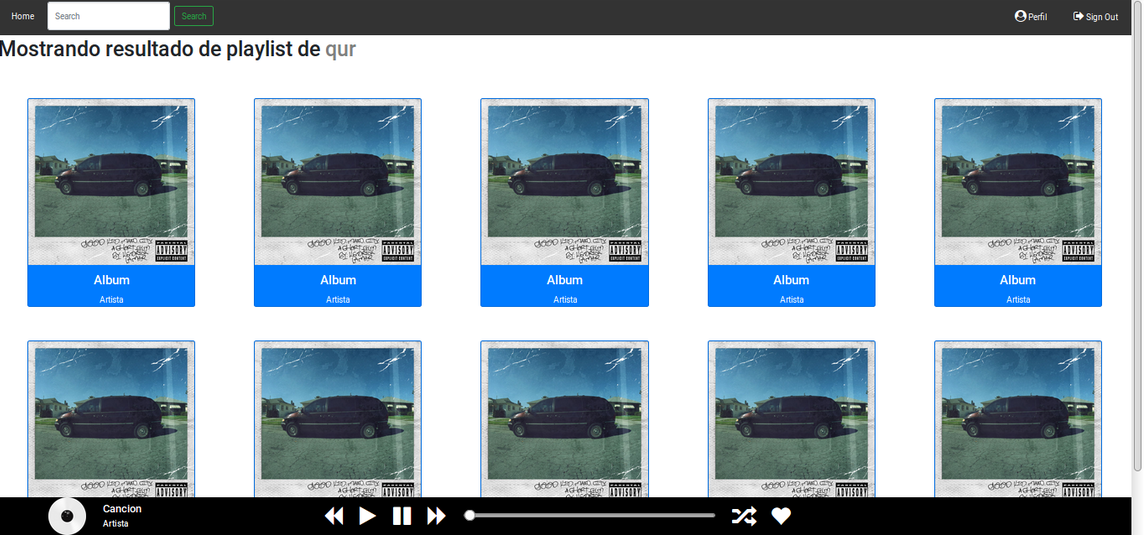


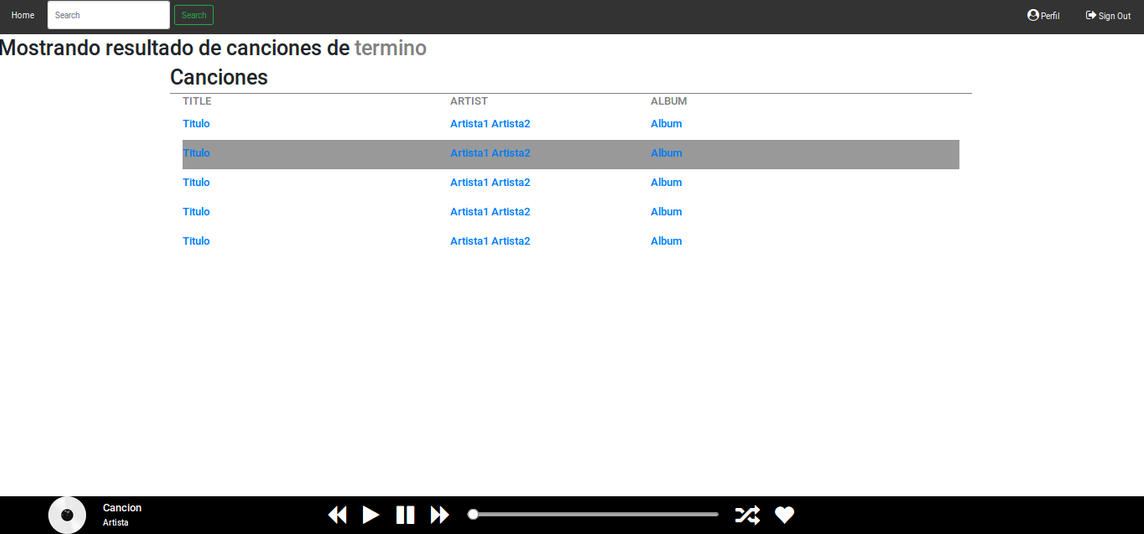
La página permite verificar la identidad de los usuarios (requisito 17, requisito NF3).

PANTALLAS DE BÚSQUEDA









Las siguientes pantallas son los resultados de hacer una búsqueda realizada a través de la barra de navegación. (requisito NF7)

### Mapa de navegación Web

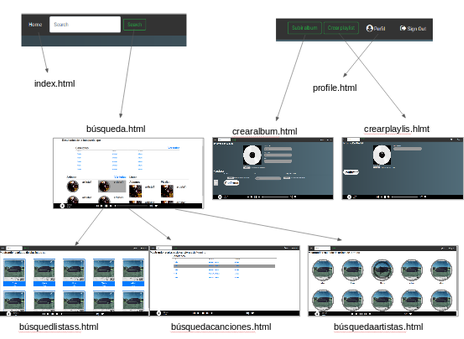
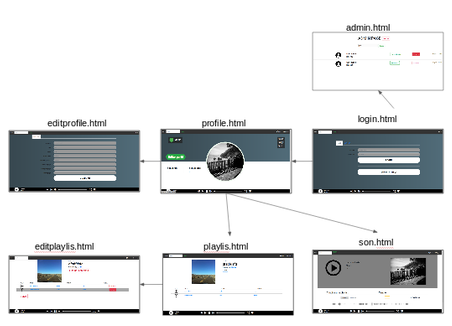
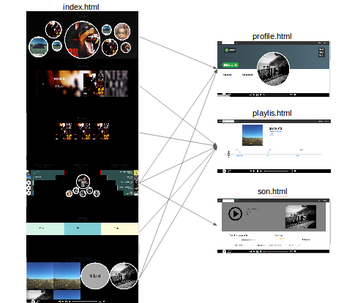
Composición del mapa de navegación Web:

* Pantalla 1: Índice (desplegado en seis distintas, 1.1, 1.2, 1.3,1.4, 1.5 y 1.6)
* Pantalla 2: Perfil del usuario Web
* Pantalla 3: Playlist
* Pantalla 4: Información canción
* Pantalla 5: Página del administrador
* Pantalla 6: Editor de perfil personal de usuario
* Pantalla 7: LogIn del cliente Web
* Pantalla 8: Editor de contenido de una playlist
* Pantalla 9: Buscador del índice
* Pantalla 10: Barra superior
* Pantalla 11: Resultados de la búsqueda
* Pantalla 12: Crear álbum
* Pantalla 13: Crear playlist
* Pantalla 14: Búsqueda de playlists
* Pantalla 15: Búsqueda de canciones
* Pantalla 16: Búsqueda de usuarios

Conexión de las pantallas:

*Se ha realizado de manera decreciente, una vez especificada la conexión entre dos pantallas no se vuelve a especificar la misma conexión (son bilaterales, pues todas permiten volver atrás), a menos que dicha pantalla conecte únicamente con la señalada pantalla de menor índice. P.e., véase que la pantalla 3 conecta con la 2 pero no está re-especificado.*

* Pantalla 1.1: 2,9
* Pantalla 1.2: 3,9
* Pantalla 1.3: 3,9
* Pantalla 1.4: 2,3,4,9
* Pantalla 1.5: 3,9
* Pantalla 1.6: 2,3,9
* Pantalla 2: 3,4,6,7,10
* Pantalla 3: 4,8
* Pantalla 4: 2
* Pantalla 5: 7
* Pantalla 6: 2
* Pantalla 7: 5
* Pantalla 8: 3
* Pantalla 9: 1
* Pantalla 10: 12,13
* Pantalla 11: 14,15,16
* Pantalla 12: 10
* Pantalla 13: 10
* Pantalla 14: 11
* Pantalla 15: 11
* Pantalla 16: 11



### Tecnologías elegidas:

Durante la programación de la aplicación Android se decidió usar Kotlin y Java, en código heredado de otras aplicaciones (ejemplos de Google sobre uso de las bibliotecas de sonido y biblioteca para crear un seekbar circular), Kotlin interopera con código Java para el funcionamiento de la misma y XML para las vistas de la aplicación.

Durante la programación de la aplicación Web se decidió usar HTML, CSS y Javascript junto con las bibliotecas Jekyll, Jquery, Bootstrap y Wavesurfer para facilitar el desarrollo de la aplicación.

### Otros aspectos técnicos de interés:

La base de datos será tipo SQL.

Existe hay una API Web y será RESTful.

La aplicación de los dispositivos Android es nativa y la aplicación de los dispositivos de sobremesa es Web.

# Memoria del proyecto

## Inicio del proyecto

Se planteó el proyecto diferenciando las partes principales de este, que son la aplicación Android, la versión Web para navegadores y el motor que sustenta a ambos, por ello se creó tres equipos para que cada uno trabajase en una parte del proyecto, estos grupos son Grupo Android, Grupo Front-End y Grupo Back-End. Se planteó como lenguaje principal Java y se optó por usar Kotlin (un lenguaje que corre en máquina virtual de Java) para Android por ser más sencillo, ser interoperable con Java y ser un lenguaje oficial de Android. En Web se usaría HTML, CSS y JavaScript. Se procedió entonces a realizar el análisis del sistema, durante esta etapa, uno de los componentes del equipo fue dado de baja médica por dos semanas, tras su vuelta y con el análisis ya realizado se decidió reorganizar los equipos para ajustar horas estimadas de trabajo con la carga de trabajo esperada en cada grupo.

También se procedió a buscar y configurar un clúster con el que trabajar.

## Ejecución y control del proyecto

El reparto de tareas se ha realizado, como ya se ha explicado, separando el grupo de trabajo en tres subgrupos de desarrollo enfocado (App Android, Front-end Web y Backend).

Dentro de estos subgrupos el coordinador de ese grupo creaba tareas de trabajo para que cada desarrollador fuera eligiendo para hacer, se detectó que este modelo de trabajo era bastante ineficiente ya que no había fechas ni obligaciones. Con el objetivo de llevar el proyecto al día según los diagramas de Gantt se corrigió el modelo de trabajo de forma que cada tarea a realizar tiene como límite una semana desde su inicio. Las causas de este cambio incluyen: dejadez de trabajo, aferrarse a una tarea, olvido de trabajo, sobrecarga de trabajo externo.

La comunicación interna ha seguido siendo a través de WhatsApp aunque se ha demostrado un poco ineficiente debido a que las conversaciones se mezclan y pierden un poco, por ello y para realizar un control periódico se ha procedido a realizar conversaciones grupales de voz semanales a través de las aplicaciones Skype y Discord en las que se discute y comenta que ha hecho o está haciendo cada grupo y persona, sin embargo en estas reuniones no se redacta ningún acta por ser más relajadas, no estar los miembros en persona y sobre todo no tomar decisiones nuevas o cambios representativos en el proyecto.

El progreso del proyecto se va midiendo con tarjetas de tareas en GitHub tal y como se planeó. Las reuniones con el cliente han sido recogidas en actas en las cuales se ha escrito temas hablados, errores cometidos para su arreglo y planteamiento de mejoras; dentro de estas se incluyen las fechas, duraciones y personal presente.

El proyecto continúa con las tecnologías y lenguajes pensados en un inicio. No ha habido incidencias en integración de código debido al uso de Git-Flow y a la decisión de no realizar pruebas unitarias automatizadas a este. A fecha de realización de este documento todavía no se han realizado despliegues. Sin embargo, el clúster que se usará como centro del Back-End ya está operativo para conectarse a él.

## Cierre del proyecto

### Jorge Aznar:

Apartado 1:

1.1.Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales ,analizar los resultados y tratar de expresar algunas lecciones aprendidas.

La estimación fue correcta, pero una mala gestión por parte de todos los integrantes produjo una sucesión de retrasos que causaron una mala distribución del tiempo.

1.2.Lecciones aprendidas sobre herramientas y tecnologías:

Se ha aprendido a usar Kotlin junto con algunas de sus bibliotecas para la realización de este trabajo así como un manejo mejor del que se tenía de la herramienta Git.

Apartado 2:

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

Evitar pensar que un miembro que parte de cero sería capaz de ponerse al día en pocas horas con el código desarrollado por otro integrante , eso consumió muchas horas y fue escasamente productivo.

2.2.¿Qué cosas está claro que haríais de otra forma?

La gestión, el proyecto requiere de una persona que invierta 100 horas a la gestión para asegurar que todos los trabajos, memorias y tareas se realizan a tiempo y revisarlas.

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

Mantener un control de que se tiene hecho y que no, lo demás ha resultado en una no muy buena experiencia para la mayoría de los miembros.

### Ángel Cañal:

Apartado 1:

1.1.Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales, analizar los resultados y tratar de expresar algunas lecciones aprendidas.

Al inicio del proyecto sobre todo, se subestimó bastante la cantidad de horas que se iba a tener que dedicar a aprender a manejar las herramientas usadas en Backend (Hibernate por ejemplo), y sin embargo, se sobreestimo las horas necesarias para Lucene por que ya se había usado en otra asignatura y se estimo una cantidad de esfuerzo similar, aunque no ha sido así.

1.2.Lecciones aprendidas sobre herramientas y tecnologías.

Se ha aprendido a trabajar con una herramienta ORM y se ha cambiado la mentalidad con la que se empezó en el proyecto. Al inicio se pensaba de forma más clásica (al estilo C, el programador hace todo), pero al usar estas herramientas se ha descubierto que no es necesario y se pueden delegar muchísimas operaciones (sobre todo las más tediosas y automatizables) a una biblioteca.

Apartado 2:

Además de conclusiones personales (razonadas) sobre el transcurso del proyecto realizado, es importante plantear ideas para mejorar los procesos llevados a cabo: si hubiera que iniciar un nuevo proyecto inmediatamente usando una metodología de gestión basada en procesos:

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

El proceso de asignación de tareas no nos ha funcionado (culpa de todos los integrantes) por lo que replantearía ese modelo al que se ha seguido en la parte final del proyecto. Con deadlines pequeños y pocas tareas en cada, reuniones semanales para controlar cómo avanza el proyecto...

2.2.¿Qué cosas está claro que haríais de otra forma?

Pondría una manera de controlar que cada uno ha hecho su parte (e.g. reuniones semanales), para que al final nadie se encuentre con que no está hecho. Además, todo lo que necesitase un equipo se pondría en una issue del proyecto del que lo necesita, y no en un grupo de Telegram/WhatsApp/Slack en el que se pierde el mensaje.

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

El "workflow" y las tarjetas/issues de Git nos ha servido para poder llevar un control interno de qué estaba hecho y considero que, bien llevado, es una herramienta muy potente que sirve a los grupos para coordinarse.

### Abel Chils:

Apartado 1

1.1 Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales, analizar los resultados y tratar de expresar algunas lecciones aprendidas.

Análisis de la sección de Android

El número de horas necesarias para el proyecto de Android se subestimó inicialmente. Aunque el número de horas finales dedicadas a esa sección sea cercano a las 200 (inicialmente se estimó un número parecido de horas), esta sección se ideo para ser hecha por un programador con experiencia en Android y otro sin ella, el cual aprovecharía el proyecto para aprender a programar para el sistema Android.

En cambio el programador con experiencia en Android tuvo que hacer la gran parte de esas horas debido a que de otra forma el proyecto no hubiera sido completado a tiempo. Una estimación a posteriori de horas necesitadas para el equipo que inicialmente se ideó sería de 270 horas, ya que alguien que se inicia tardaría la mitad del proyecto hasta tener un rendimiento adecuado.

A nivel tecnológico, el coste tan alto de esta sección se debe a varios motivos, el primero consiste en el sistema de reproducción de música el cual se subestimó tanto su coste en número de horas como su complejidad. Esto se debió a que inicialmente se hicieron pruebas con el sonido en Android (reproducir canciones desde una pantalla) lo cual fue muy sencillo, esto llevó a la idea errónea de que crear el sistema al que nos habíamos comprometido sería igual de sencillo, que solamente deberíamos de re-aprovechar eso y añadirle unas pequeñas funcionalidades más, en cambio crear un sistema de sonido capaz de sobrevivir al cambiar de pantalla posee una complejidad mucho mayor, pasando de ser unas pocas líneas de código a ser una capa completa dentro de las 4 capas que posee la arquitectura de nuestra aplicación.

En segundo lugar otro punto en el que se subestimó mucho el coste fue en la realización de pruebas y corrección de errores de la aplicación.

Por otro lado también ha habido un gran coste debido a los retrasos en la sección del servidor Back-End. Al ser este el punto sobre el que se sustenta el sistema (el sistema Android es un cliente de esa aplicación) no disponer de los servicios que iba a ofrecer hasta muy avanzado el proyecto (API que ofrece el Back-End), implicó una reestructuración del proyecto para poder adaptarse una vez se tuvo esta.

Por otro lado, la implementación de la capa de llamadas a esta sección, la cual se empezó a realizar al final del proyecto, se realizó en paralelo mientras el propio Back-End aún estaba en desarrollo lo que causó una gran pérdida de horas debido a errores que existían en él y a la falta de la especificación de las peticiones necesarias para hacer determinadas llamadas, ya que aunque se disponía de una API (la cual fue propuesta desde la sección de Android a mitad de proyecto), esta era más bien orientativa sobre que llamadas ofrecería, para tanto en Android como en Front-End poder estructurar nuestros proyectos en base a algo. La especificación final de las llamadas quedaba en manos del Back-End.

Por otro lado la sección Android también se vio afectada por la pérdida de un miembro en el grupo el cual perteneció a esta sección hasta la mitad del proyecto cuando fue movido a otro grupo al ver que no se adaptaba a la forma de programar para Android. Esto causó que el nuevo miembro que entró al proyecto tuviese que empezar desde cero, tanto a nivel de programación en Android como a nivel de proyecto.

Las lecciones aprendidas son en primer lugar, que las secciones en las que se sustentan los sistemas son las primeras que se tienen que terminar. Posiblemente hubiera sido mejor idea empezar todos los miembros del equipo en la sección Back-End, y una vez terminada empezar con las otras, de esta forma ninguna sección hubiera tenido que esperar por ninguna.

Análisis proyecto

En los costes del proyecto en general también se puede ver que un gran número de horas se ha perdido por la comunicación dentro del grupo. Por un lado por la falta de respuesta ante peticiones ( tener que pedir las cosas de forma reiterada es algo que consume mucho tiempo). En otros casos por la falta de comunicación entre miembros del grupo. Y por último por el incumplimiento reiterado de los plazos marcados (que una entrega de la que se depende para avanzar con una tarea no se entrega a tiempo, implica que la segunda tendrá que esperar).

Estos problemas de organización se podrían haber mejorado cambiando la jerarquía organizativa que marcamos al principio del proyecto. Nuestra jerarquía consistía en 3 líderes de sección y un coordinador entre estos 3. Inicialmente no se ideó la figura de jefe de proyecto aunque el coordinador del las 3 secciones tuvo que asumir este rol para intentar acotar el descontrol.

A posteriori creo que inicialmente se debía de haber colocado un integrante del equipo como jefe de proyecto, el cual solamente hubiese tuviese la función de organizar el proyecto, asignar tareas y realizar las memorias. Esto hubiera mejorado notablemente la organización dentro del grupo y hubiera hecho que el proyecto se terminara en menos tiempo.

1.2.Lecciones aprendidas sobre herramientas y tecnologías.

Durante la realización del proyecto Android se usó Android Studio y el lenguaje de programación Kotlin y se aprendieron muchas funcionalidades de Android como la gestión del sonido y la utilización de servicios web, siendo esta parte bastante satisfactoria. Por otro lado hubieron herramientas que dificultaron el proyecto. Primero tendríamos git-flow, herramienta que empezamos a usar sin conocer adecuadamente, la cual mientras funciona ayuda y hace el trabajo con git más cómodo, cuando aparecen errores o cuando miembros que no saben usarlo entran al proyecto crea mucho descontrol y provoca mucha pérdida de tiempo.

Por otro lado tendríamos la herramienta LaTeX, la cual se empezó a usar para realizar las memorias sin saberla usar adecuadamente, lo que nos hizo perder muchas horas durante la realización las primeras, hasta ser sustituido por Microsoft Word.

Apartado 2:

Además de conclusiones personales (razonadas) sobre el transcurso del proyecto realizado, es importante plantear ideas para mejorar los procesos llevados a cabo: si hubiera que iniciar un nuevo proyecto inmediatamente usando una metodología de gestión basada en procesos:

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

Internamente en el equipo de Android el primer error que cometimos fue a la hora de planificar las tareas que se tenían, ya que para a partir de unos requisitos obtener que tareas hay que hacer, primero se necesita realizar un buen diseño del sistema final y para eso se necesita experiencia que no teníamos, lo que provocó que el diseño inicial no se pareciese nada al sistema final, este diseño después de replantearse se debió de volver a convertir en tareas con las nuevas tareas que debían de hacerse, aunque debido a los tantos cambios de tareas que hubo y que la sección la estaba haciendo una sola persona se dejo de invertir esfuerzo en eso.

2.2.¿Qué cosas está claro que haríais de otra forma?

El primer cambio que haría sería colocar la figura de director general del proyecto, el cual se encargaría de ir repartiendo las tareas y de llevar una visión global de cómo va todo. Otro tema que habría que cambiar sería en la mentalidad de los integrantes del grupo los cuales tendrían que empezar a asumir responsabilidades, el cumplimiento de los compromisos y los plazos así como el esfuerzo continuado desde el principio del proyecto, evitando la dejadez que han mostrado algunos miembros del grupo.

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

A nivel de organización no seguiría haciendo nada igual, aunque al final del proyecto una idea que salió adelante y mejoró el final fue recopilar todas la tareas que quedaban para poder llevar un control de cuales se iban terminando y cuáles no.

Por otro lado otra idea con la que se experimentó a mitad de proyecto fue realizar reuniones semanales en las cuales los miembros mostraban sus progresos. Se hicieron varias reuniones de este estilo aunque se dejaron de hacer por dos motivos, el primero es que al vernos todos los días en clase, los problemas ya se comentaban ahí, la segunda es que debido a que entre semanas no habían progresos suficientes de todas las secciones, estas reuniones perdieron sentido ya que eran horas perdidas.

Si todos los miembros hubieran llevado un trabajo regular durante todo el proyecto estas reuniones hubiesen tenido sentido y hubieran sido aprovechadas.

### Oscar Fraca:

Apartado 1:

1.1.Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales, analizar los resultados y tratar de expresar algunas lecciones aprendidas.

Se subestimó bastante las horas tanto de aprendizaje de nuevas tecnologías como las de dedicación a implementación. También creo que la forma de separar el trabajo ha resultado en un aumento de horas de trabajo innecesario, añadir a esto las esperas entre mensajes de comunicación necesaria.

1.2.Lecciones aprendidas sobre herramientas y tecnologías.

Se ha aprendido a usar lenguajes de documentación tales como LaTex y Markdown y la forma de tratar con una base de datos en una aplicación. También el uso de un programa de control de versiones.

Apartado 2:

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

Teniendo en cuenta que no todos los componentes sabíamos de todo creo que habría sido más productivo no separarnos en equipos sin dedicarnos todos a un mismo bloque (Back-End) buscando tener terminada la base de la aplicación lo primero y poder apoyarnos entre todos con cualquier duda o problema y tener más flexibilidad de horarios. Una vez terminada esta parte, se podría seguir el esquema con las otras dos partes o repartir estas partes entre dos grupos más numerosos. También haría pequeñas explicaciones de un miembro al resto sobre que hace o como usar/probar cada parte del programa que ese miembro ha realizado.

2.2.¿Qué cosas está claro que haríais de otra forma?

Dedicar un poco más de tiempo a analizar totalmente el problema capturando todos los requisitos necesarios y no añadir requisitos que el cliente no ha pedido.

Realizar el diseño inicial de la aplicación total al inicio de acuerdo a los requisitos en vez de ir ampliándolo según se va implementando. Realizar una pequeña reunión presencial de resolución de dudas/observación de progresos cada semana en vez de ser una reunión de videoconferencia donde 7 personas no pueden hablar a la vez o ver documentos de manera sincronizada. Tener en cuenta los distintos horarios que tiene cada integrante cada semana para la asignación de tareas con fecha. Establecer unas herramientas de trabajo desde un inicio para evitar fallos, errores, problemas y cambios a mitad.

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

El uso de Git y las tareas de Github ayudan bastante a tener un objetivo y comunicación así como la gran ayuda que proporciona un programa de control de versiones.

### Alex Oarga:

1.1.Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales, analizar los resultados y tratar de expresar algunas lecciones aprendidas.

1.2.Lecciones aprendidas sobre herramientas y tecnologías.

A continuación se detallan tecnologías con las que no estaba familiarizado y otras a destacar.

Para la web se ha utilizado HTML5, CSS y Javascript además de algunos elementos de CSS3. En algunas ocasiones se ha utilizado también jQUery. Para elementos del diseño se ha utilizado Bootstrap. Se ha recurrido también a elementos de diseño como las animaciones con CSS o CSS Grid. Para la onda de sonido se utilizó en principio la libreria Wavesurfer aunque esta finalmente se desechó por su alta latencia y se sustituyó por una onda dibujada en un canvas de Javascript. Para incluir fragmentos de código en todas las páginas se ha utilizado Jekyll. Un aspecto importante ha sido la conexión al Backend. Esta se ha realizado en código Kotlin compilado a Javascript a través de Intellij. Esto ha forzado una toma de contacto con el lenguaje Kotlin a la hora de depurar.

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

Tener el diseño final más claro antes de pasar a la implementación. Seguir un desarrollo más estructurado. El desconocimiento de las tecnologías a llevado en ocasiones a errores o un mal planteamiento que se podrían haber evitado.

2.2.¿Qué cosas está claro que haríais de otra forma?

Llevar una organización mas documentada del proyecto. Procurar una mayor calidad y legibilidad del código y asegurarse de que la implementación es correcta. Esto último se debe a que al desconocer la tecnología algunos resultados obtenidos no son profesionales(en el caso de la web). Plantear mejor el reparto de tareas desde un primer momento.

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

Mantener una comunicación abierta sobre el estado del proyecto, ya sea a través de reuniones o cara a cara (en nuestro caso en clase). Utilizar metodologías de desarrollo destinadas a proyectos como GitFlow, workflows o issues.

### Jorge Pinilla:

1.1.Comparar las estimaciones al inicio del proyecto (tamaño, esfuerzos, costes) con los resultados finales, analizar los resultados y tratar de expresar algunas lecciones aprendidas.

Las estimaciones realizadas para montar la infraestructura ha sido bastante acertada, en cambio, la estimación de desarrollar el backend ha sido mucho menor que el trabajo real.

1.2.Lecciones aprendidas sobre herramientas y tecnologías.

He aprendido a utilizar:

* Infrastructura: Ceph, KVM, HA-Proxy, Glassfish
* Backend: Hibernate
* Frontend: Amazon SDK, Kotlin

2.1.¿Qué cambios haríais respecto a los procesos que habéis seguido durante este proyecto?

Tener a un líder organizador del proyecto y fijar mucho mejor las interfaces comunes, poder tener más comunicación entre frontend y backend

2.2.¿Qué cosas está claro que haríais de otra forma?

Proceso de dirección del proyecto, es necesario un líder coordinador que se ocupe tanto de definir las interfaces entre varios gropos como de asegurarse de que se cumplen los plazos y realizar su seguimiento

2.3.¿Qué cosas seguiríais haciendo más o menos igual?

Actualmente seguiría usando las tecnologías que hemos empleados a pesar del esfuerzo que ha llevado aprenderlas son realmente útiles.

### Esfuerzo realizado

La siguiente tabla mide el esfuerzo realizado por cada miembro del proyecto en su respectiva tarea, así como una serie de observaciones/aclaraciones.

|  |  |  |  |
| --- | --- | --- | --- |
| Miembro del equipo | Horas dedicadas | Tarea | Observaciones |
| Jorge Aznar | h | Desarrollo del software para el Backend | Desarrollo memoria técnica v2, final y algo de Android |
| Ángel Cañal | h | Desarrollo del software para el Backend | Líder sub-grupo Backend y desarrollo |
| Abel Chils | h | Desarrollo de la aplicación Android | Líder sub-grupo Android y director del proyecto |
| Oscar Fraca | h | Desarrollo del software para el Backend | Desarrollo memoria técnica v2, final |
| Alex Oarga | h | Desarrollo de la aplicación Web |  |
| Jorge Pinilla | h | Desarrollo del Backend de la aplicación Web | Líder sub-grupo Frontend y gestor servidores Backend |

# ANEXO I

## Bibliografía:

Estándar para Android y Web (<https://google.github.io/styleguide/javaguide.html>)

Estándar recomendado por la W3Schools para Frontend [(https://www.w3schools.com/html/html5](http://www.w3schools.com/html/html5) syntax.asp)

Git Flow (<https://danielkummer.github.io/git-flow-cheatsheet/>)