Plan de gestio´n, ana´lisis, Diseño y memoria del proyecto



SpreadYourMusic

**Índice**

1. [Introduccio´n 2](#_TOC_250014)
   1. [Resumen 2](#_TOC_250013)
2. [Organizacio´n del proyecto 2](#_TOC_250012)
   1. [Equipo 2](#_TOC_250011)
3. [Plan de gestio´n del proyecto 3](#_TOC_250010)
   1. [Procesos 3](#_TOC_250009)
      1. [Procesos de inicio de proyecto 3](#_TOC_250008)
      2. [Procesos de ejecucio´n y control del proyecto 3](#_TOC_250007)
      3. [Procesos te´cnicos 4](#_TOC_250006)
      4. [Plan de gestio´n de configuraciones 4](#_TOC_250004)
      5. [Plan de construccio´n y despliegue del software 5](#_TOC_250003)
4. [Ana´lisis y Diseño del sistema 8](#_TOC_250002)
   1. [Ana´lisis de requisitos 8](#_TOC_250001)
      1. Sistema 8
      2. Backend 8
      3. Android 10
      4. Web 11
   2. [Diseño del sistema 13](#_TOC_250000)

5 **Memoria del proyecto**

* + 1. Inicio del proyecto
    2. Ejecución y control del proyecto
    3. Cierre del proyecto

# Introduccio´n

## Resumen

La aplicación a desarrollar consistirá en un reproductor de mu´sica en *streaming* inspirado en *Soundcloud* y *Spotify*.

Es una aplicación orientada a todo tipo de usuario, tanto a los mu´sicos que están empezando su carrera en el mundo de la mu´sica como a cualquier persona aficionada a ella. Nuestra aplicación permitira´ a los usuarios subir canciones, crear listas de reproducción o escuchar canciones utilizando un reproductor propio entre otras funcionalidades. La versio´n Android permitira´ a los usuarios descargar las canciones para poder escucharlas sin necesidad de estar conectado a Internet. Tambie´n incluira´ caracter´ısticas 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 tambie´n poseera´ integracio´n con redes sociales, as´ı como la posibilidad de autentificacio´n mediante cuenta de Google.

El usuario tendrá recomendaciones personalizadas para el usuario tanto basadas en su historial de reproducción como en geolocalizacio´n y facilitara´ al usuario encontrar canciones, pudiendo buscar canciones por categor´ıas, autor, nombre, as´ı como tambie´n mostrando canciones populares dentro de la aplicacio´n.

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 mo´vil o en ambas ya que incorpora un sistema de sincronizacio´n de manera que el usuario puede seguir escuchando en cualquier dispositivo la misma canción justa en el momento en el que la dejo´.

# Organizacio´n del proyecto

## Equipo

El equipo esta´ formado por 7 estudiantes de Ingenier´ıa Informa´tica: Jorge Aznar Lo´pez, A´ngel Can˜al Muniesa, Abel Chils Trabanco, Alicia Yasmina Albero Escudero, O´scar Fraca Fernández, Alexandru Ioan Oarga Hategan y Jorge Pinilla Lo´pez.

De estos, cinco poseen conocimientos sobre programacio´n Web (*Frontend*) as´ı como pra´ctica en el desarrollo del *Backend* de una aplicación Web. Por otro lado, todos tienen experiencia en el Diseño de base de datos, as´ı como experiencia con la programacio´n sobre la plataforma Android. Adema´s, los integrantes del grupo ya poseen experiencia trabajando juntos, lo cual facilitara´ la comunicacio´n entre ellos.

Para desarrollar esta aplicación se han creado 3 grupos y cada uno se centrara´ en una parte del desarrollo. Un equipo se encargara´ del *Backend*, otro de la interfaz Web (o *Frontend*) y por u´ltimo otro de la aplicación Android. La asignacio´n de los integrantes del grupo a cada una de las partes se ha realizado en base a la experiencia que posea el integrante en cuestio´n en dicha a´rea

El equipo de *Backend* esta´ formado por O´ scar, A´ngel y Jorge Aznar. El equipo de la plataforma Android esta´ formado por Abel y Yasmina. Por u´ltimo, el equipo de *Frontend* esta´ formado por Alexandru y Jorge Pinilla.

Para gestionar el proyecto se ha designado un director de proyecto y coordinadores específicos dentro de cada a´rea. El director es Abel, el coordinador de *Backend* es A´ngel, el de Android es Abel y el de Frontend es Jorge Pinilla.

# Plan de gestio´n del proyecto

## Procesos

### Procesos de inicio de proyecto

La aplicación mo´vil funcionara´ en dispositivos con Android 5.0 como m´ınimo, la aplicación web funcionara´ en los navegadores Firefox y Chrome en su u´ltima versio´n disponible para Windows y Mac. Para realizar las pruebas de la aplicación mo´vil se va a usar un dispositivo f´ısico con Android5.0. Por otro lado, para las pruebas de la versio´n web se va a usar un ordenador con el navegador instalado y actualizado (Chrome y Firefox). El sistema inicialmente funcionara´ sobre un clu´ster, que contara´ con un almacenamiento bruto de 20GB. Se ha estimado este taman˜o para el clu´ster ya que presenta un espacio aceptable para almacenar canciones de prueba en esta arquitectura. Este espacio se usara como base en las pruebas iniciales ya que cuando el software se entregue al cliente este podra´ poner el taman˜o que desee en su clu´ster puesto que el sistema es fa´cilmente escalable.

### Procesos de ejecucio´n y control del proyecto

Las comunicaciones del grupo se van a realizar mediante un grupo en la aplicación de mensajer´ıa instanta´nea *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 comunicacio´n, se usara´n las *Issues* o incidencias de la plataforma de alojamiento de los proyectos *GitHub*. En esta plataforma de alojamiento se almacenara´ el co´digo 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 sera´n redactadas por al menos un miembro del equipo durante dicha reunio´n. De forma similar se registrara´n 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 reunio´n, la duracio´n de la reunio´n, los miembros presentes en la reunio´n y los temas y decisiones que se tomen en la reunio´n.

Todas las semanas tienen un conjunto de tareas asociadas. Al final de cada semana, el responsable de cada proyecto revisará las tareas que se han realizado esa semana y, si hay tareas que no se han cumplido, se asignara´n automa´ticamente para la siguiente semana, siendo estas tareas las primeras que se debera´n hacer. Adema´s, cada semana el responsable del sub-proyecto revisara´ cuantas tareas se han realizado para la siguiente iteracio´n como me´trica de monitorizacio´n de la desviacio´n segu´n el plan original. Tras esta revisio´n el responsable del sub-proyecto asignara´ 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 razo´n que haga imposible esta mediacio´n sera´ el resto del equipo quien deberá mediar.

### Procesos te´cnicos

Las herramientas utilizadas tanto para desarrollo del software (construccio´n, pruebas y despliegue) sera´n IntelliJ Idea y Android Studio debido a que ambas dan soporte para los lenguajes de programacio´n Java y Kotlin además de su buena integracio´n con Git, herramienta que se utilizara´ 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 ira´n creando pruebas unitarias sobre el mismo con JUnit.

Siguiendo una metodolog´ıa de Diseño incremental, cada dos semanas se tiene planificada la entrega interna de una versio´n nueva del software. Estas versiones se basa en la versio´n anterior an˜adie´ndole ciertas funcionalidades y no sera´n lanzadas al pu´blico.

## Planes

### Plan de gestio´n de configuraciones

El co´digo desarrollado debera´ seguir una serie de estándares, siendo los siguientes los aceptados. Para el nombrado de me´todos, variables, clases... se utilizara´ la filosof´ıa **CamelCase** en los proyectos de Android y *Backend* y la filosof´ıa **SnakeCase** en el caso del *Frontend*. Adema´s, para la escritura de co´digo 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 nu´mero de dispositivos puedan visualizar correctamente la pa´gina 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 royecto se encargara´ de la revisio´n de los commits de su proyecto y la aceptacio´n o no aceptacio´n de co´digo nuevo. Del despliegue y la puesta en marcha del sistema se encargara´ Jorge Pinilla y de la correcta administracio´n del sistema de control de versiones (VCS) Git A´ ngel Can˜al.

Este Sistema de Control de Versiones está compuesto por 4 repositorios, uno en el que se alojan todos los documentos del sistema (p.e. ”Propuesta Econo´mica” o ”Plan de Gestio´n”) y 3 repositorios ma´s, uno por cada uno de los proyectos (Android, *Backend* y *Frontend*). Para intentar reducir conflictos o problemas, so´lo los miembros ma´s experimentados tendra´n acceso total a los reposito- rios (estos son A´ ngel Can˜al y Abel Chils). El resto del equipo so´lo tendrá acceso al repositorio de Documentos y al del proyecto en el que esté trabajando.

Para reducir problemas causados por modificacio´n concurrente del mismo fichero de co´digo fuente, se va a usar un flujo de trabajo en Git denominado Git Flow por el que hay dos ramas principales, una para la u´ltima versio´n estable (*master*) y otra para la versio´n en desarrollo (*develop*). Adema´s, para cualquier nueva caracter´ıstica a an˜adir 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 env´ıar co´digo 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 gestio´n de incidencias, se va a aprovechar este sistema para gestionar todas las situaciones inesperadas que pudieran suceder. Adema´s, con la gestio´n 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 ma´s pequen˜a posible). Estas incidencias podra´n estar en 3 estados, ”Abierto”, ”En progreso” y ”Terminado” además de estar asignado a un punto en el tiempo en el que debera´n estar terminadas por completo. Para pasar de ”En progreso” a ”Terminado” el coordinador del equipo debe dar el visto bueno e incorporar el nuevo co´digo al resto (rama *develop*).

### Plan de construccio´n 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.

Despliegue de software.

Debido a presupuesto hemos simulado todo el entorno de pruebas en un solo servidor mediante máquinas virtuales y virtualización anidada.

El servidor host unicamente es el encargado de redirigir las peticiones mediante un HAProxy a modo de router.

Se utilizan 4 maquinas virtuales:

3 maquinas 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 semiautomatica de ceph-deploy.

Se utiliza RadosGateway (http://docs.ceph.com/docs/master/radosgw/) como interfaz para exportar los datos mediante HTTP, de una manera similar a Amazon S3, las canciones se almacenan en Buckets y disponen de un URL único.

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 gestion unificada del servicio de almacenamiento.

La idea de utilizar maquinas virtuales es la gran escalabilidad que permite y la posibilidad de configurar resistencia a fallos automática.

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.

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

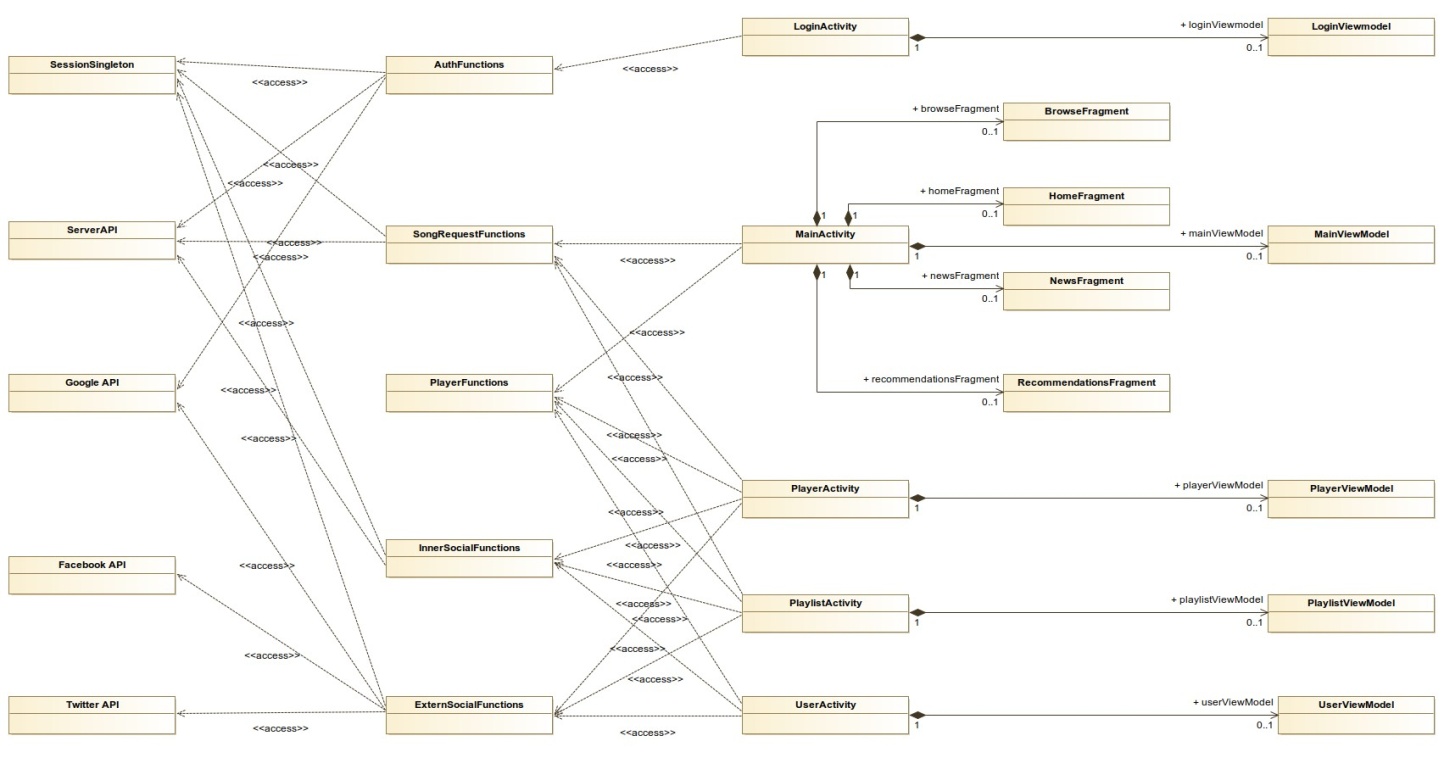
backend 192.168.200.16 2006 HTTP 7800 HTTPS 7880

postgres 192.168.200.17 2007 POSTGRES 7850

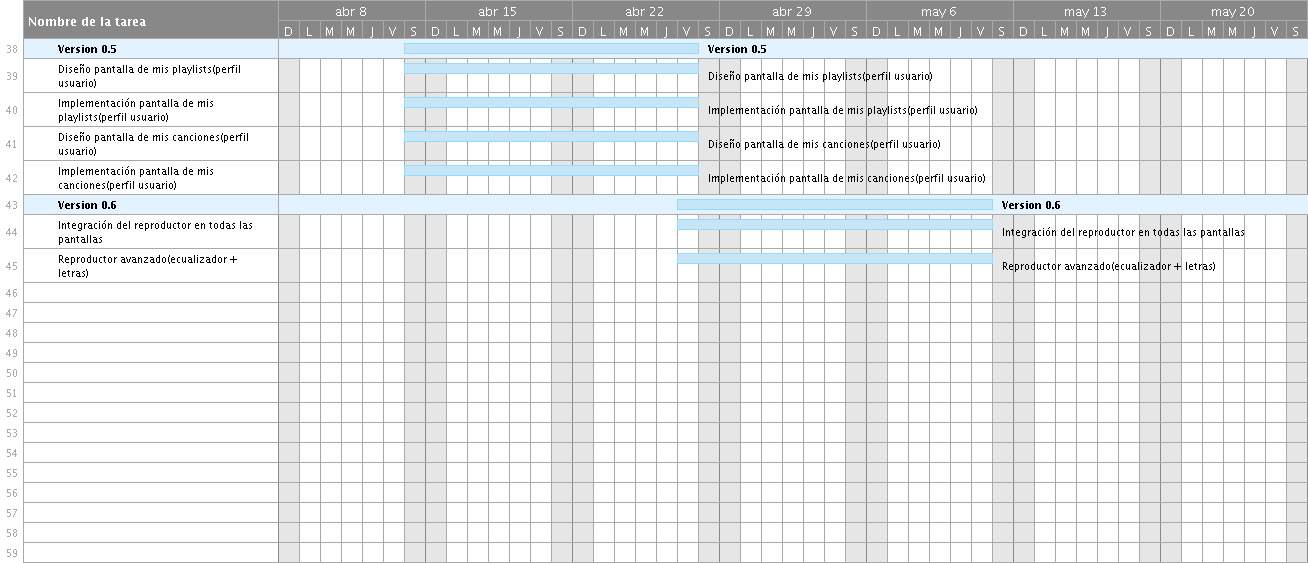
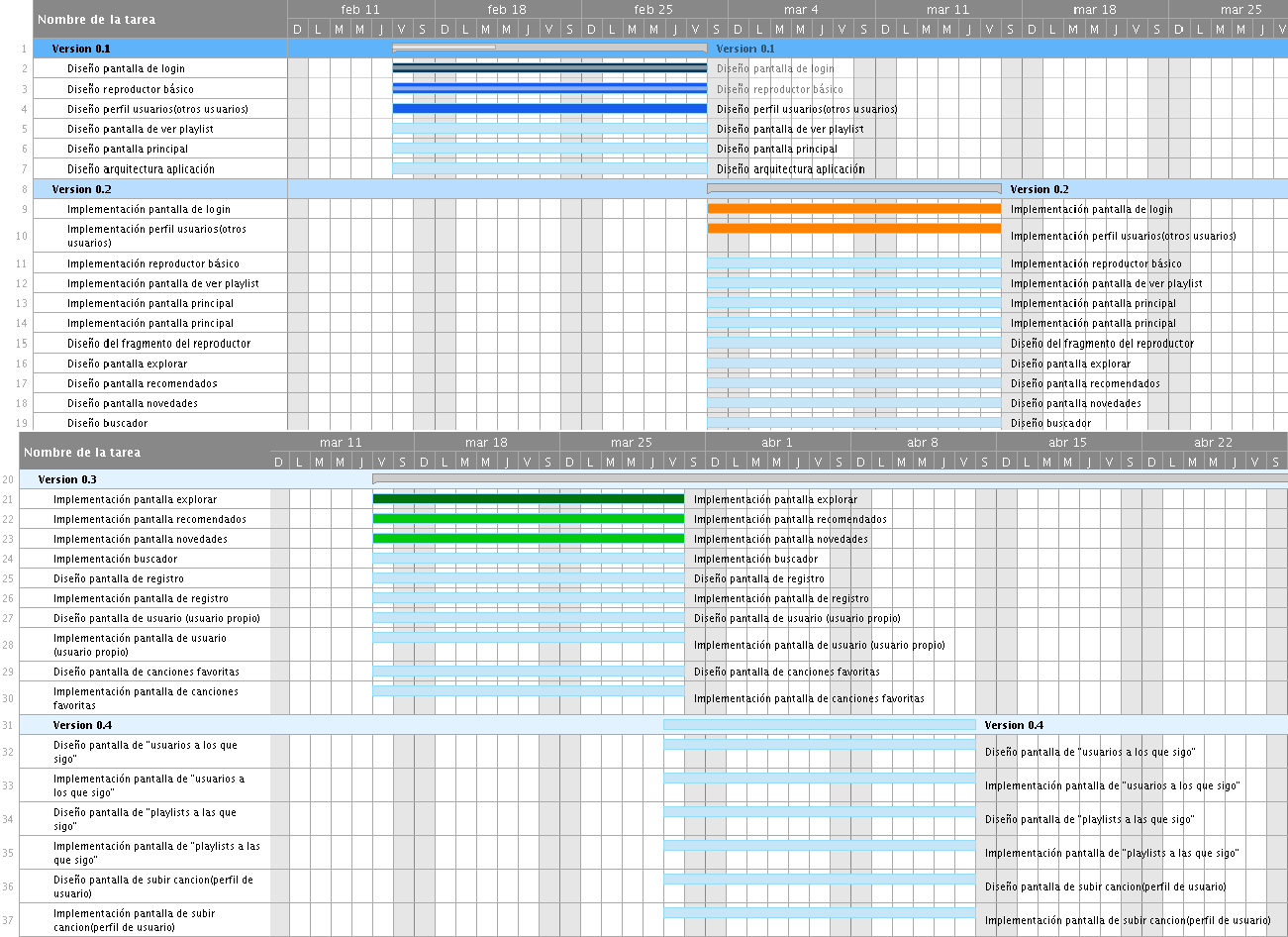
**Arquitectura de la aplicación Android**

La aplicación Android está estructurada en 3 capas, aunque a su vez una de estas está dividida en dos: en la primera de estas se encuentran las APIs con servicios externos (backend, Google, Facebook y Twitter) y una clase que se encargara´ de mantener la sesio´n de la aplicación (incluyendo el manejo de la persistencia de esta sesio´n). Luego existe una segunda capa en la cual se encuentran funciones para manipular la capa citada anteriormente. Por u´ltimo esta´ la capa que se encarga del manejo de la interfaz gra´fica. Esta u´ltima capa se divide a su vez en dos. Una de ellas se encarga del manejo 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 evitar tener que repetir alguna operaciones cuando se cambia la orientación del dispositivo o cuando se apaga la pantalla.

**Diagrama de clases de la aplicación Android**



**Diagrama de Gantt de la aplicación Android**



**Versiones de la aplicación Android**

Versión 0.1 (3 marzo) Diseño pantalla de login, Diseño reproductor básico, Diseño perfil usuarios(otros usuarios), Diseño pantalla de ver playlist, Diseño pantalla principal (Aparecerán recomendaciones, novedades de artistas o playlists a las que se esté suscrito y canciones, playlists o artistas populares), Diseño arquitectura aplicación.

Versio´n 0.2 (17 marzo) Implementacio´n pantalla de login, Implementacio´n perfil usuarios(otros usuarios), Implementacio´n reproductor ba´sico, Implementacio´n pantalla de ver playlist, Implementacio´n pantalla principal (Aparecera´n recomendaciones, novedades de artistas o playlists a las que se este´ suscrito y canciones, playlists o artistas populares), Diseño del fragmento del reproductor que aparecera´ cuando se esta´ escuchando mu´sica en una pantalla que no es la pantalla de reproductor, Diseño pantalla explorar (pantalla en la que aparecen todos los generos, artistas del momento y recomendaciones basadas en la ubicacio´n), Diseño pantalla recomendados, Diseño pantalla novedades, Diseño buscador.

Versio´n 0.3 (31 marzo) Implementacio´n pantalla explorar (pantalla en la que aparecen todos los gneros, artistas del momento y recomendaciones basadas en la ubicacio´n), Implementacio´n pantalla recomendados, Implementacio´n pantalla novedades, Implementacio´n buscador, Diseño pantalla de registro, Implementacio´n pantalla de registro, Diseño pantalla de usuario (usuario propio), Implementacio´n pantalla de usuario (usuario propio), Diseño pantalla de canciones favoritas, Implementacio´n pantalla de canciones favoritas.

Versio´n 0.4 (14 abril) Diseño pantalla de ”usuarios a los que sigo”, Implementacio´n pantalla de ”usuarios a los que sigo”, Diseño pantalla de ”playlists a las que sigo”, Implementacio´n pantalla de ”playlists a las que sigo”, Diseño pantalla de subir cancion(perfil de usuario), Implementacio´n pantalla de subir canción (perfil de usuario).

Versio´n 0.5 (28 abril) Diseño pantalla de mis playlists(perfil usuario), Implementacio´n pantalla de mis playlists(perfil usuario), Diseño pantalla de mis canciones(perfil usuario), Implementacio´n pantalla de mis canciones (perfil usuario).

Versio´n 0.6 (u´ltima) (12 mayo) Integracio´n del reproductor en todas las pantallas, Reproductor avanzado(ecualizador + letras).

**Arquitectura del Backend**

La arquitectura del Backend del sistema esta´ estructurada en 4 capas:

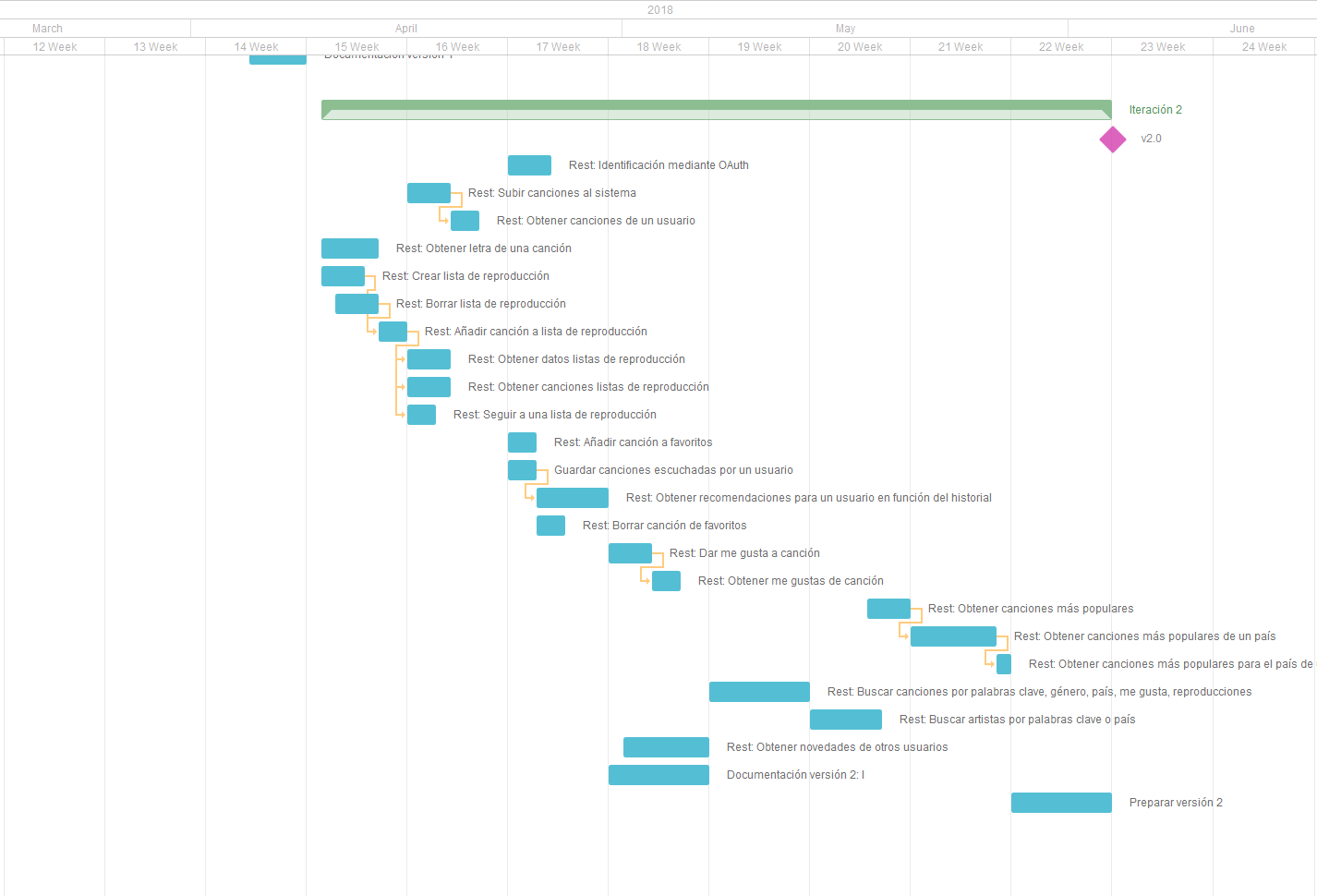
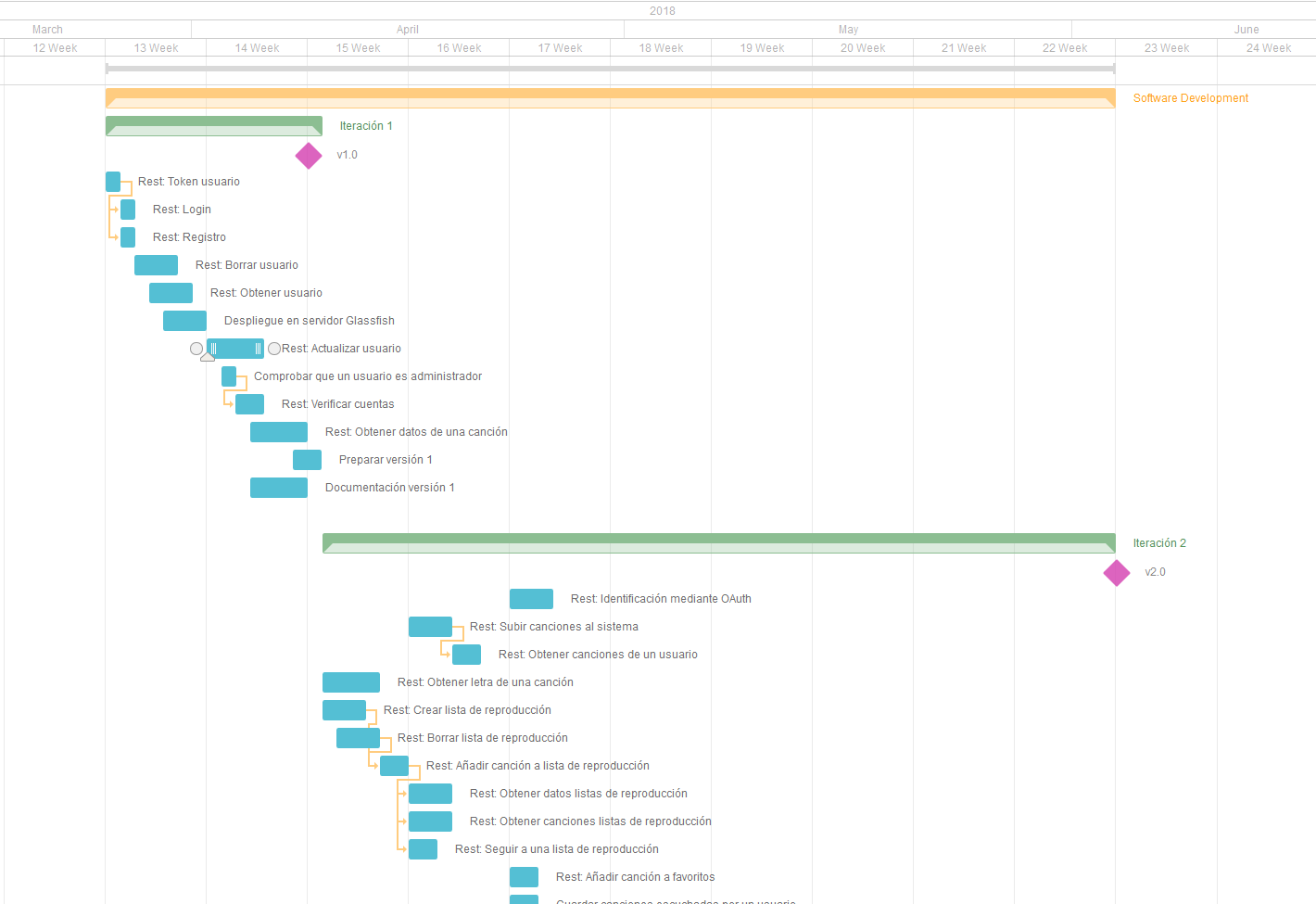
En primer lugar se halla la conexión con la base de datos, la cual será mapeada según el método objeto-relación, haciendo uso de Hibernate.

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.

**Diagrama de Gantt del Backend**

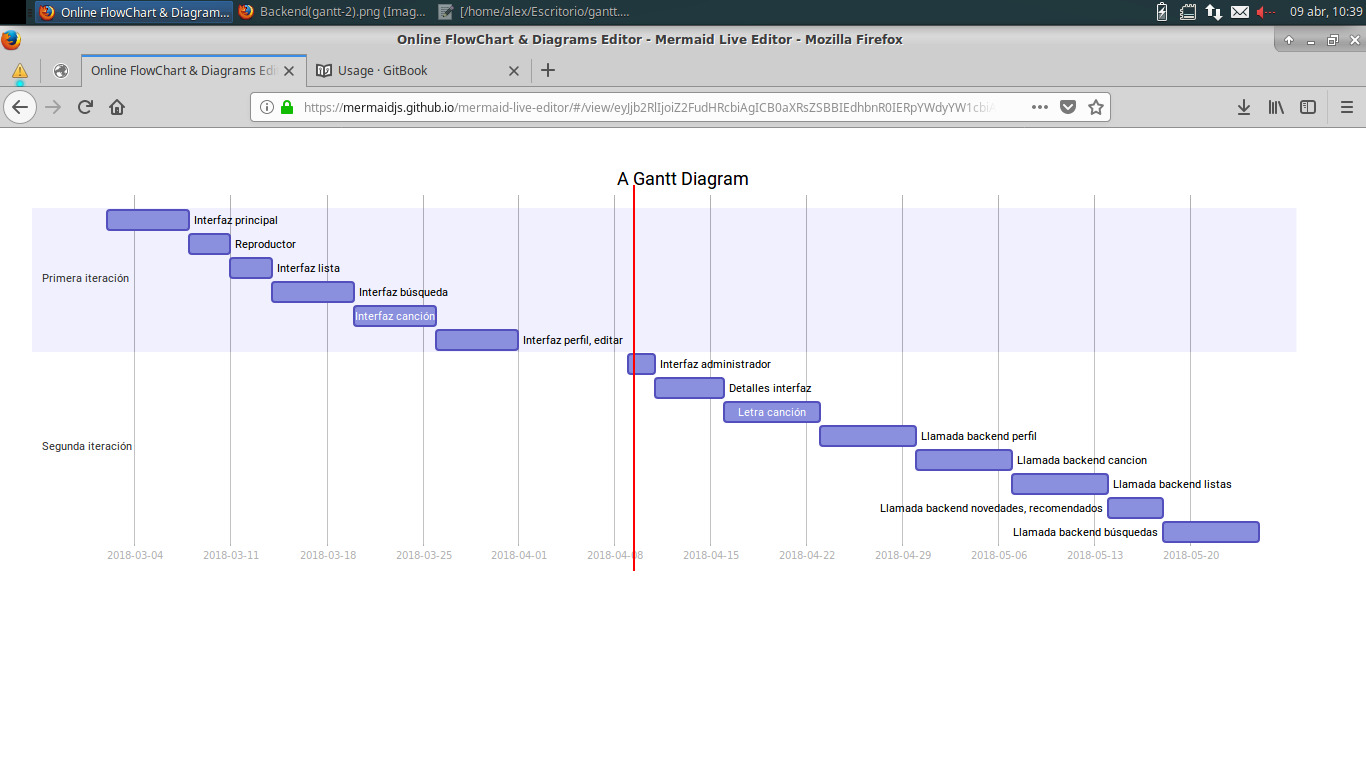


**Arquitectura de la aplicación Web**

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.

**Diagrama de Gantt de la aplicación Web**

****

# Ana´lisis y Diseño del sistema

## Ana´lisis de requisitos

## Sistema: Requisitos no funcionales

* + - 1. El sistema tendrá una versio´n para Android y otra para Web.
      2. El sistema soporta los ficheros MP3, WAV, OGG.
      3. El sistema dispone de un servidor para el almacenamiento de canciones.
      4. La aplicación movil soportara´ Android 5.0 y la Web la u´ltima versio´n de Firefox y Chrome.

**Backend: Requisitos funcionales**

* + - 1. El sistema se compone de Usuarios, Canciones y Listas de reproduccio´n.
      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 t´ıtulo, un audio, la transcripcio´n de dicho audio, el pa´ıs de la cancio´n, cua´ntos me gusta tiene y el nu´mero de reproducciones.
      2. Una lista de reproducción o categor´ıa es una lista de canciones generadas por el sistema agrupadas por ge´nero, e´xito, pa´ıs, situacio´n para la que es propicia o generada por el usuario.
      3. Un usuario registrado se compone de un nombre, un nick u´nico (es decir, que dos usuar- ios no pueden tener el mismo nick), un correo electro´nico, una contrasen˜a para acceder a la aplicacio´n, la fecha de nacimiento, una biograf´ıa, una foto de perfil, que´ canciones ha reproducido, cua´ntos me gusta han recibido las canciones que ha subido, cua´ntos me gusta ha dado, sus canciones, el pa´ıs y las redes sociales que quiera an˜adir de forma opcional (Twitter, Facebook e Instagram).
      4. El sistema permite obtener todos los datos de una cancio´n.
      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 sera´n pu´blicas.
      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 reproduccio´n.
      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 cancio´n.
      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 electro´nico, su contraseña, su fecha de nacimiento, su biografía, su foto de perfil, su pa´ıs y sus redes sociales.
      17. El sistema permite que los usuarios registrados sigan a una lista de reproduccio´n.
      18. El sistema permite buscar canciones mediante palabras clave (t´ıtulo), ge´nero, pa´ıs, me gusta, reproducciones o una combinacio´n de varios.
      19. El sistema permite ver las canciones ma´s populares de un pa´ıs.
      20. El sistema permite ver las canciones ma´s populares del pa´ıs de un usuario.
      21. El sistema permite buscar artistas mediante palabras clave (nick, nombre y biografía), país o una combinación de varios.
      22. El sistema permite buscar listas de reproducción mediante palabras clave (título), canciones de la lista, tipo de orden de la lista o una combinación de varios.
      23. 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 funcionales**

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

**Android: Requisitos funcionales**

* + 1. El sistema permite un solo tipo de usuario, y es el usuario registrado (usuario que esté registrado en el sistema).
    2. El sistema permite que los usuarios de identifiquen tanto con usuario y contraseña así como con una cuenta de Google.
    3. El sistema permite que los usuarios suban canciones a la aplicacio´n
    4. El sistema permite que los usuarios se registren en la aplicacio´n.
    5. El sistema permite a los usuarios tener una lista de canciones favoritas, la cual será privada.
    6. El sistema permite an˜adir a favorito una canción que se esta´ escuchando.
    7. El sistema permite desde cualquier pantalla escuchar canciones y ver los controles ba´sicos de canciones (play, pause y pasar cancio´n)
    8. El sistema permite desde una pantalla específica acceder a ma´s controles sobre las canciones.
* Desde la pantalla específica el sistema permite avanzar o retroceder en la canción a un momento exacto de esta.
* Desde la pantalla específica el sistema permite ver la letra de la canción dinámica (va al ritmo de la canción) en el caso que la posea.
* Desde la pantalla específica el sistema permite ver la onda de sonido de la canción.
* Desde la pantalla específica el sistema permite cambiar el orden de muestra de las canciones aleatorio o lineal).
  + 1. El sistema posee una pantalla principal en la que se mostrara´n recomendaciones (canciones, listas de reproducción y usuarios), novedades sobre los usuarios y listas de reproducción seguidos y canciones ma´s populares.
    2. 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.
    3. 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, as´ı como an˜adir ma´s canciones, listas de reproducción o eliminar alguna de las dos.
    4. El sistema permite crear listas de reproducción formadas por canciones de las que puedes ser autor o no, las cuales sera´n publicas.
    5. El sistema permite modificar los datos de usuario una vez creado.
    6. El sistema posee una pantalla específica en la que se puede ver las canciones que posee una lista de reproducción as´ı como seguirla.
    7. El sistema permite buscar las canciones ma´s populares de un ge´nero.
    8. El sistema permite ver las canciones ma´s populares en el pa´ıs desde el que se conecta el dispositivo.
    9. El sistema permite buscar canciones, artistas y listas de reproducción por su nombre mediante un buscador.
    10. El sistema posee una pantalla principal en la que se mostrara´n recomendaciones (canciones, listas de reproducción y usuarios), novedades sobre los usuarios y listas de reproducción seguidos y canciones ma´s populares.
    11. 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 c omo un enlace a sus redes sociales.
    12. 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, as´ı como an˜adir más canciones, listas de reproducción o eliminar alguna de las dos.
    13. **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.

1. **No Registrados**
   1. Los usuarios no registrados pueden acceder a la pa´gina principal, páginas de los diferentes usuarios, páginas de canciones además de poder realizar bu´squedas.
   2. Los usuarios no registrados pueden reproducir canciones de usuarios o de listas de reproducción públicas.
   3. Registrados
   4. Los usuarios registrados pueden realizar todas las funcionalidades de un usuario no registrado.
   5. El sistema permite a los usuarios registrados identificarse.
   6. El sistema permite a los usuarios registrados modificar su información personal.
   7. El sistema permite a los usuarios registrados subir canciones.
   8. El sistema permite administrar una lista de reproducción privada de favoritos.
   9. El sistema permite añadir cualquier canción a favoritos.
   10. El sistema permite crear listas de reproducción públicas
   11. El sistema permite añadir cualquier canción a una lista de reproducción previamente creada por el usuario.
   12. El sistema permite eliminar una canción de sus listas de reproducción o de la lista favoritos.
   13. El sistema permite seguir/ dejar de seguir a un usuario.
   14. El sistema permite seguir/ dejar de seguir a una lista de reproducción.
   15. El sistema permite modificar sus enlaces a redes sociales.
   16. (RNF) 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.

**(c) Administrador**

* 1. El sistema permite al usuario administrador puede verificar la identidad de un usuario.
  2. El sistema permite al usuario administrador modificar la información de un usuario, canción o playlist.
  3. El usuario administrador se identifica en la aplicación con un usuarios y contraseña predefinidos.
  4. (RNF) 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.
  5. (RNF) Cada canción, lista o usuario tiene una pantalla de administrador desde donde el sistema permite al administrador la modificación.

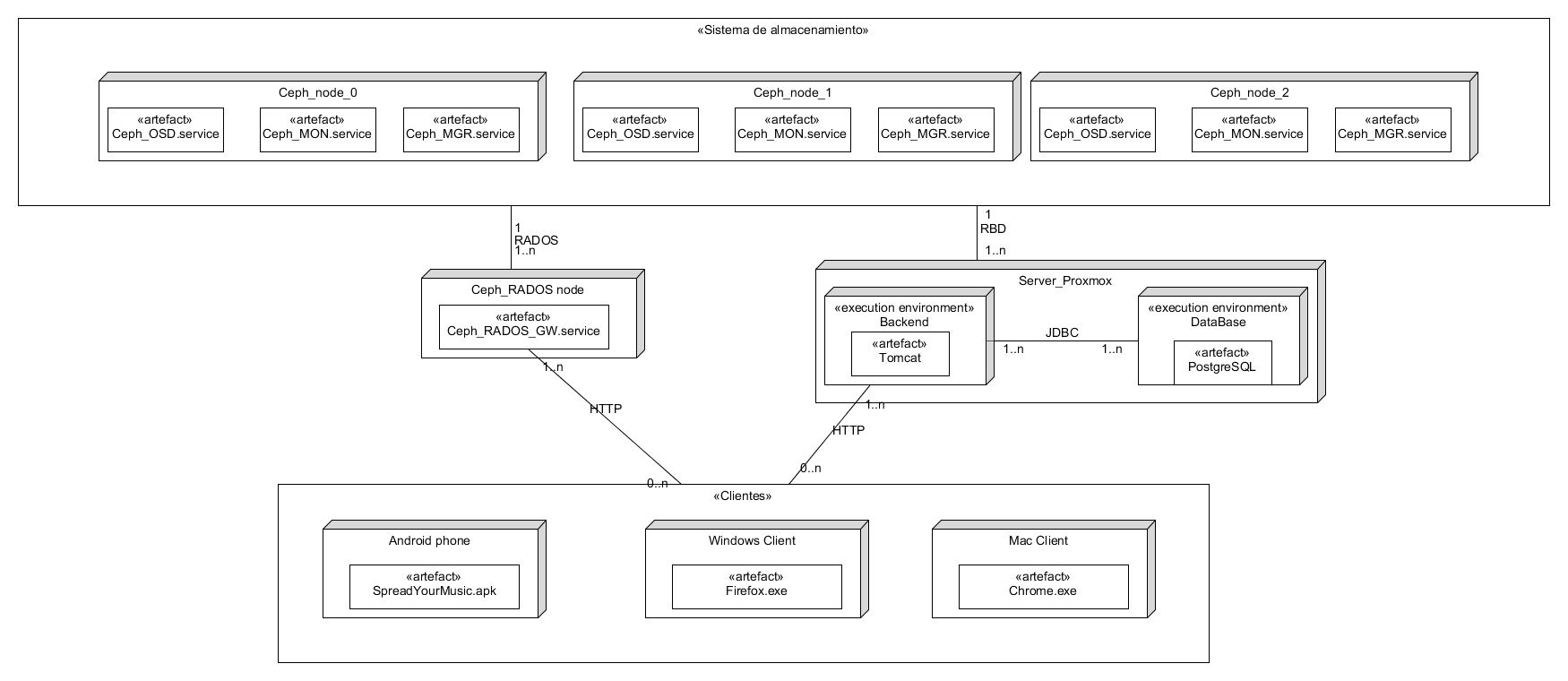
**Web: Requisitos no funcionales**

1. El sistema tiene una pantalla principal donde se mostraran novedades de artistas y canciones, y canciones más populares.
2. El sistema tiene una pantalla de usuario pública para todo el mundo donde aparece su informacio´n, canciones, listas de reproducción pu´blicas, nu´mero de seguidores y enlaces a sus redes sociales
3. El sistema cuenta en todas las pantallas con un reproductor de audio que incluye informacio´n sobre la canción que se está reproduciendo y las opciones de avanzar, retroceder a un momento de la cancio´n; avanzar o retroceder a la siguiente o anterior canción si se está reproduciendo una lista; parar la reproducción y cambiar el orden de reproducción a aleatorio.
4. El sistema tiene en todas las pantallas un buscador para realizar una bu´squeda de canciones, listas y usuarios.
5. El sistema tiene una pantalla para cada canción donde se puede ver la informacio´n de la cancio´n, la onda de sonido y si tuviese las letras de forma dina´mica en reproducción.
6. El sistema tiene una pantalla para identificarse/ registrarse.
7. 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

## Diagrama de módulos:

* Diagrama de distribución o despliegue:



* Tecnolog´ıas 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

## 5.1 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 maquina 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.

## 5.2 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 esta haciendo cada grupo y persona, sin embargo en estas reuniones no se redacta ningún acta por ser mas 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 esta operativo para conectarse a él.

## 5.3 Cierre del proyecto

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 |
| Yasmina Albero | 8h | Desarollo de la aplicación Android |  |
| Jorge Aznar | 21,5h | Desarollo del software para el Backend | Desarrollo memoria técnica v2 |
| Ángel Cañal | 38,25h | Desarollo del software para el Backend | Líder sub-grupo Backend y desarrollo memoria técnica v1 |
| Abel Chils | 71h | Desarollo de la aplicación Android | Líder sub-grupo Android y director del proyecto |
| Oscar Fraca | 12h | Desarollo del software para el Backend | Desarrollo memoria técnica v2 |
| Alex Oarga | 27h | Desarollo de la aplicación Web |  |
| Jorge Pinilla | 28h | Desarollo 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/>)