

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

Proyecto Software

*5 de junio de 2020*

*Álvaro García Díaz, Óscar Baselga Lahoz, Alberto Calvo Rubió,*

*Saúl Flores Benavente, Luis García Garcés, Alejandro Facorro Loscos, Germán Garcés Latre*

*Grupo 10. Karen Sparck Jones*



## Tabla de contenido

1	Introducción .....	5
2	Organización del proyecto .....	6
3	Plan de gestión del proyecto.....	7
3.1	Procesos .....	7
3.1.1	Procesos de inicio del proyecto.....	7
3.1.2	Procesos de ejecución y control del proyecto .....	7
3.1.3	Procesos técnicos .....	9
3.2	Planes .....	11
3.2.1	Plan de gestión de configuraciones.....	11
3.2.2	Plan de construcción y despliegue del software .....	12
3.2.3	Plan de aseguramiento de la calidad .....	13
3.2.4	Calendario del proyecto y división del trabajo.....	16
4	Análisis y diseño del sistema .....	20
4.1	Análisis de requisitos.....	20
4.1.1	Requisitos funcionales.....	20
4.1.2	Requisitos no funcionales .....	21
4.2	Diseño del sistema .....	22
4.2.1	Diagrama de despliegue.....	22
4.2.2	Diagrama de clases.....	23
4.2.3	Componentes y conectores.....	29
4.2.4	Interfaces.....	30
4.2.5	Mapa de navegación .....	41
4.2.6	Casos de uso .....	42
5	Memoria del proyecto.....	48
5.1	Inicio del proyecto.....	48
5.2	Ejecución y control del proyecto.....	48
5.3	Cierre del proyecto.....	52
6	Conclusiones.....	54
7	Anexo 1: Actas de reunión .....	58
8	Bibliografía .....	99

## Tabla de figuras

<i>Ilustración 1: Logo fondo blanco .....</i>	10
<i>Ilustración 2: Logo con nombre horizontal .....</i>	13
<i>Ilustración 3: Logo con nombre cuadrado .....</i>	14
<i>Ilustración 4: Logo sin nombre cuadrado .....</i>	14
<i>Ilustración 5: Diagrama de despliegue .....</i>	22
<i>Ilustración 6: Diagrama de clases web .....</i>	23
<i>Ilustración 7: Diagrama de clases Android .....</i>	24
<i>Ilustración 8: Diagrama de paquetes Android .....</i>	25
<i>Ilustración 9: Modelo de datos Backend .....</i>	26
<i>Ilustración 10: Diagrama Entidad-Relación .....</i>	27
<i>Ilustración 11: Diagrama de clases servidor .....</i>	28
<i>Ilustración 12: Diagrama de paquetes servidor .....</i>	28
<i>Ilustración 13: Diagrama de componentes y conectores .....</i>	29
<i>Ilustración 14: Pantalla inicial Android .....</i>	30
<i>Ilustración 15: Pantalla inicio sesión Android .....</i>	30
<i>Ilustración 16: Pantalla registro Android .....</i>	31
<i>Ilustración 17: Pantalla principal Android .....</i>	31
<i>Ilustración 18: Pantalla listas de reproducción Android .....</i>	31
<i>Ilustración 19: Pantalla podcast Android .....</i>	32
<i>Ilustración 20: Pantalla lista de canciones/podcast Android .....</i>	32
<i>Ilustración 21: Pantalla desplegable lateral Android .....</i>	32
<i>Ilustración 22: Pantalla artista Android .....</i>	33
<i>Ilustración 23: Pantalla configuración Android .....</i>	33
<i>Ilustración 24: Pantalla búsqueda amigos Android .....</i>	33
<i>Ilustración 25: Pantalla lista de amigos Android .....</i>	34
<i>Ilustración 26: Pantalla notificaciones Android .....</i>	34
<i>Ilustración 27: Pantalla reproductor Android .....</i>	34
<i>Ilustración 28: Pantalla inicial Web .....</i>	35
<i>Ilustración 29: Pantalla inicio de sesión Web .....</i>	35
<i>Ilustración 30: Pantalla perfil Web .....</i>	36
<i>Ilustración 31: Pantalla principal Web .....</i>	36
<i>Ilustración 32: Pantalla listas de reproducción Web .....</i>	37
<i>Ilustración 33: Pantalla podcast Web .....</i>	37
<i>Ilustración 34: Pantalla lista de canciones/podcast Web .....</i>	38
<i>Ilustración 35: Pantalla artista Web .....</i>	38
<i>Ilustración 36: Pantalla configuración Web .....</i>	39
<i>Ilustración 37: Pantalla álbum Web .....</i>	39
<i>Ilustración 38: Pantalla notificaciones Web .....</i>	40
<i>Ilustración 39: Pantalla reproductor Web .....</i>	40
<i>Ilustración 40: Mapa de navegación Android .....</i>	41
<i>Ilustración 41: Mapa de navegación Web .....</i>	42
<i>Ilustración 42: Caso de uso "Escuchar una canción de una lista de reproducción" Android .....</i>	43
<i>Ilustración 43: Caso de uso "Escuchar una canción de una lista de reproducción" Web .....</i>	43
<i>Ilustración 44: Caso de uso "Reproducir canción de un artista" Android .....</i>	44
<i>Ilustración 45: Caso de uso "Reproducir canción de un artista" Web .....</i>	44
<i>Ilustración 46: Caso de uso "Suscribirse a un podcast" Android .....</i>	45
<i>Ilustración 47: Caso de uso "Suscribirse a un podcast" Web .....</i>	45
<i>Ilustración 48: Caso de uso "Añadir amigo" Android .....</i>	46
<i>Ilustración 49: Caso de uso "Añadir amigo" Web .....</i>	46
<i>Ilustración 50: Caso de uso "Consultar notificaciones" Android .....</i>	47
<i>Ilustración 51: Caso de uso "Consultar notificaciones" Web .....</i>	47

<i>Ilustración 52: Control requisitos Web .....</i>	49
<i>Ilustración 53: Control requisitos Android .....</i>	50
<i>Ilustración 54: Control requisitos Global .....</i>	51
<i>Ilustración 55: Control issues.....</i>	51
<i>Ilustración 56: Métricas totales.....</i>	51
<i>Ilustración 57: Gráfica de requisitos e issues.....</i>	52
<i>Ilustración 58: Gráfica de líneas de código.....</i>	52

## 1 Introducción

En este proyecto se va a desarrollar una aplicación que ofrece un servicio enfocado a la reproducción de música y podcasts en streaming.

Se trata de una aplicación que permite a los usuarios disfrutar de las obras de artistas, lo que hace que sea una forma rápida y sencilla de que los músicos que están empezando en el mundillo puedan abrirse un hueco en él.

Los usuarios consumidores disfrutan sin límites del contenido que es publicado, consiguiendo así horas y horas de entretenimiento.

La aplicación permite a los usuarios crear sus listas de reproducción (privadas o públicas) y compartirlas con sus amigos, reproducir canciones o listas de otros usuarios, seguir a sus artistas favoritos, y muchas posibilidades más. También cuenta con un ámbito social, dado que cada usuario puede compartir contenido con amigos desde la propia aplicación o a través de otras redes sociales, como puede ser Whatsapp.

El sistema es accesible desde la aplicación **Android** o desde un **navegador Web**, como Google Chrome o Mozilla Firefox. Así, el usuario tiene la opción de conectarse desde distintos dispositivos sin importar su sistema operativo. Existen detalles que consiguen mejorar la usabilidad del sistema, como por ejemplo mantener el estado de la última canción escuchada al terminar la sesión, pudiendo así continuar la próxima vez desde donde se dejó.

El proyecto va a contar con una serie de hitos. El primero consiste en la finalización de una primera versión del ‘Plan de gestión, análisis, diseño y memoria’ del proyecto, programado para el 14 de marzo.

El siguiente hito trata de una segunda versión del plan anterior y de la primera propuesta de código, en la cual se tendrá implementado un reproductor de música capaz de comunicarse con la base de datos. Está programada para el 15 de abril. El último hito corresponde a la entrega de la memoria (actual documento) y de la aplicación final. Se encuentra programada para el 8 de junio.

Por último, se va a realizar un resumen del resto de los apartados de este documento. Primero se presentan a los miembros del equipo (punto 2) que van a desarrollar el proyecto; después, se va a explicar el llamado plan de gestión del proyecto (punto 3), dividido en dos partes: los procesos (punto 3.1), que se encargan de describir cómo se van a llevar a cabo las distintas tareas que hay que ir realizando a lo largo del proyecto, y los planes (punto 3.2), que establecen un objetivo y qué es lo que se necesita para conseguir alcanzarlo; el apartado que sigue consiste en el análisis y diseño del sistema (punto 4), en él se van a plantear los requisitos que el sistema deberá cumplir en su versión final y cómo han sido diseñados para su incorporación; seguidamente aparece la memoria del proyecto (punto 5), en la cual se realiza una descripción de cómo ha ido evolucionando el proyecto, incluyendo cambios respecto a la versión inicial, imprevistos surgidos, etc.; se termina con un apartado de conclusiones (punto 6) que engloba tanto personales como grupales.

## 2 Organización del proyecto

El equipo está formado por 7 alumnos de Ingeniería Informática de la Universidad de Zaragoza:

Nombre	Tareas	Nombre Github
<b>Saúl Flores Benavente</b>	Líder backend, servidor y API	<i>saul205</i>
<b>Alberto Calvo Rubió</b>	Director de Proyecto, base de Datos, despliegue local y cloud	<i>AlbertoCalvoRubio</i>
<b>Luis García Garcés</b>	Líder frontend Android, aplicación Android	<i>luisgg98</i>
<b>Alejandro Facorro Loscos</b>	Aplicación Android	<i>Greyscarr</i>
<b>Óscar Baselga Lahoz</b>	Aplicación Android	<i>oscarbaselga</i>
<b>Álvaro García Díaz</b>	Líder frontend web, aplicación web	<i>Alvarogd6</i>
<b>Germán Garcés Latre</b>	Aplicación Web	<i>fntkg</i>

Para gestionar el proyecto, se han establecido diferentes roles:

- **Director de Proyecto:** Alberto Calvo. Se encarga de liderar y coordinar a todos los miembros, velar por el progreso del proyecto, responsable de la documentación entregable (revisión y asignación de apartados), investigar y plantear estrategias y metodologías de trabajo para el proyecto. Tiene una visión global del proyecto y sirve de enlace con el cliente.
- **Líderes de las diferentes partes del sistema:** coordinan un pequeño equipo (reparto de tareas y control de horas), mantienen el repositorio específico, revisan su código y son encargados de crear, revisar y cerrar los issues de su equipo. Permite que en algunas reuniones sólo acudan los distintos líderes.
  - Líder Backend: Saúl Flores
  - Líder Frontend Android: Luis García
  - Líder Frontend Web: Álvaro García
- **Encargados de actas:** Alberto Calvo, Luis García y Germán Garcés

Para el desarrollo, algunos conocimientos que van a ayudar a realizar este proyecto que los integrantes poseen son:

- Conocimientos sobre programación backend y frontend en el desarrollo de un sistema de información con Java
- Experiencia en el desarrollo de aplicaciones Android
- Conocimientos de bases de datos (Oracle, MySQL y PostgreSQL)

### 3 Plan de gestión del proyecto

#### 3.1 Procesos

##### 3.1.1 Procesos de inicio del proyecto

El proyecto se iniciará con la división en dos grupos, **Front-End y Back-End**, también se decidirá a qué serie de dispositivos irá nuestra aplicación y con qué **lenguaje de programación y frameworks** se trabajará, una vez decidido, se creará la organización de cada grupo y del proyecto mediante la votación de los diferentes **organizadores** y la asignación de cada miembro del proyecto a cada grupo. Por último, se crearán las distintas vías de **comunicación** entre los miembros del proyecto.

Respecto a la creación de **repositorios en Github**, cada organizador del proyecto creará el propio repositorio siguiendo las directrices del proyecto y dando permiso a cada uno de los integrantes de dicho grupo en concreto, los repositorios de los diferentes grupos que conforman el proyecto son los siguientes:

- **Back-End** [1]
- **Front-End Web** [2]
- **Front-End Móvil** [3]
- **Documentación** [4]

En cuanto a la **formación** de cada uno de los grupos, cada grupo sigue una lista de tutoriales para cada uno de los frameworks y lenguajes de programación elegidos.

- **Flask[Episodios 1-5 y 8] [ [5]**
- **SQLAlchemy y API** [6]
- **Angular:** [7]
- **TypeScript y HTML** [8]
- **Android Studio** [9]
- **Flutter:** [10]

Para la **formación de usuarios nuevos en la plataforma**, se tendría que comunicar con el Director del Proyecto para unirlo a los diferentes chat de comunicación del proyecto e indicar en cuál de los diferentes grupos se comenzaría a trabajar, en dicho grupo se le indicaría que tutoriales seguir, qué herramientas instalar y la asignación tanto de tareas como de permisos en diferentes plataformas.

Por último, nos decantamos por utilizar **Heroku** como servidor cloud de la aplicación, para ello se utiliza la versión gratuita, la cual nos brinda: 10.000 filas para la base de datos, un proceso para ejecutar, 512MB de RAM y 550h de límite de tiempo de los dynos(contenedores). Para la obtención de dicho servidor, solo se tuvo que realizar el registro y elegir la opción gratuita por defecto; la persona que tiene acceso a dicho servidor mediante la contraseña y nombre de usuario es el propio Director del Proyecto y el líder de backend, también se tuvo que instalar el add-on **Heroku Postgres** en la aplicación creada en Heroku.

##### 3.1.2 Procesos de ejecución y control del proyecto

Las comunicaciones internas se llevarán a cabo mediante distintos medios. El encargado de administrar y dar de alta al personal es el Director de Proyecto.

- **Grupo de Whatsapp:** conversaciones acerca de dudas, avisos y temas de menor importancia, evitando que las decisiones queden reflejadas aquí ya que se suelen perder con el tiempo.
- **Canal de Discord:** tras los sucesos ocurridos (cuarentena por COVID-19) se ha necesitado un medio por el que realizar reuniones mediante llamadas de voz. Una vez empezadas las reuniones en Discord, el Director de Proyecto guía la reunión para evitar que se alarguen y que se cumpla el objetivo. Todos los participantes deben guardar silencio e intervenir de forma ordenada para llevar a cabo la reunión de forma correcta. También se puede utilizar como medio de conversación informal y resolución de dudas cuando no se sea el momento de reunión. Para acceder se puede utilizar el siguiente enlace [11]
- **Github issues:** cada líder de repositorio (web, android, backend y documentación) crea un issue por cada tarea en la que se pueda dividir el trabajo, modularizando basándose en los requisitos y en la planificación, de forma que se independice lo máximo posible una tarea de otra. Se asignan a cada uno de los miembros de cada equipo según la valoración del líder basándose en los conocimientos del asignado, su dedicación de horas y se tendrá en cuenta dentro de lo posible el horario de cada uno. Se priorizará el equilibrio de horas dedicadas de cada uno. Quedará constancia del progreso mediante comentarios cada vez que se realice un progreso o se termine la tarea, en cuyo caso el líder correspondiente revisará la tarea/issue y determinará si cerrarlo o no.
- **Google Calendar:** herramienta utilizada para mostrar el calendario del proyecto como entregas y reuniones. Utilizando funciones de recordatorio mediante notificaciones y correos.

Además, se realizarán **actas de reuniones formales** (ya sean de grupo o con el profesor/cliente) en los que quedan reflejados: participantes, ausencias, duración, orden del día, acciones tomadas, objetivos para la siguiente reunión y fecha de la siguiente reunión. Se utiliza la plantilla estándar de Google Docs "*Notas de la reunión - Escritor moderno*". En cada reunión los **encargados de actas** crearán el acta, tomarán notas y revisarán el acta al final de la reunión.

Mediante estos elementos se espera un correcto funcionamiento de grupo de forma que todos estén notificados de todas las actividades y se mantenga una serie de **registros formales** en los que poder identificar problemas y sacar conclusiones, evitando los problemas que puedan surgir y corrigiéndolos lo antes posible.

Respecto a la **resolución de disputas** se solucionará mediante la conversación entre las personas que tienen la disputa. Si fuese necesario, se recurrirá al líder del correspondiente grupo (backend, android, web) y en todo caso, al Director de Proyecto. Si se debe tomar una decisión que involucra a todos los miembros del grupo se realizará una votación entre todos ellos. El ganador será el que obtenga mayoría de votos. En caso de empate, recaerá la decisión sobre el Director de Proyecto.

Las **medidas de progreso** a tener en cuenta son los **issues** cerrados respecto a los totales, los **requisitos cumplidos** frente al total y las **líneas de código**. Se utilizará una hoja de cálculo en la que anotar las métricas y se calculen datos y gráficas de forma automática (Disponible en el capítulo 5). Estos progresos serán medidos semanalmente en la reunión semanal (normalmente sábado o domingo) por el director de proyecto y se revisarán para obtener una visión global del progreso. Para valorar cuantitativamente la dificultad de los issues y requisitos se les asigna un valor 1(Fácil), 2(Medio) y 3(Difícil) consiguiendo unos cálculos más próximos a la realidad. Los issues los valoran los líderes teniendo en cuenta tiempo aproximado, desconocimiento de la solución y experiencia propia. Se les añadirá una etiqueta con un determinado color

correspondiente en Github Issues (**low difficult/#ffff000**, **medium difficult/#ff7000** y **hard difficult/#ff0000**). A los requisitos se les asigna la valoración en conjunto por todos los miembros. Para contar las líneas de código de todo el proyecto se han de descargar en una carpeta los tres repositorios y eliminar el entorno virtual de Flask(no queremos contar sus librerías). Se utiliza la herramienta cloc en la carpeta y se recoge el total de líneas de código mostrado por pantalla. Además, se guardará una copia de todo el desglose de análisis del código en un fichero y se guardará en Google Drive en la carpeta de Control de proyecto por si se necesita su revisión posteriormente.

En el caso de que se detecten problemas de rendimiento, avance insuficiente o desviaciones respecto al plan inicial, se hará un reajuste de tareas al personal para solucionar el problema de progreso de forma más ajustada, intentando aprovechar el máximo de horas posibles que cada miembro disponga. Incluso se puede mover personal de un grupo a otro para intentar ayudar en tareas que no sean específicas de la tecnología propia (ya que probablemente se desconozca), intentando ayudar en documentación u otras tareas. Se prioriza la entrega de todos los requisitos pactados, pero en caso de no poder llegar a la entrega final con todo desarrollado con buena calidad, se recortará en requisitos prevaleciendo la calidad de los demás.

### 3.1.3 Procesos técnicos

#### Proceso de Configuración y Construcción:

En cada uno de los pasos siguientes se trabaja siempre desde la rama master de cada repositorio como se indica en el apartado 3.2.1.

También, dentro del fichero “**README.TXT**” está indicado las versiones necesarias para trabajar en cada grupo y que instalaciones y configuraciones han de hacerse, dicho fichero será **actualizado** conforme a los cambios que se realicen en cada uno de los diferentes grupos del proyecto. Este un documento en el cual se explica de mejor manera este punto se puede encontrar en el repositorio “**Documentación**” en Github [4].

En la parte de **frontend web**, al trabajar con Angular, las dependencias se indicarán en el fichero “**package.json**”. También, existe una carpeta denominada “**node\_modules**” en la que se descargaran todas las dependencias, este fichero tiene **dos dependencias** que hacen uso de ficheros javascript (control de volumen) las cuales son “**jquery**” y “**bootstrap**”.

Para la parte de **frontend móvil**, utiliza Flutter, siendo este un plugin para Android Studio. También es necesario haberse instalado una serie de bibliotecas, las cuales se recopilan en el fichero “**pubspec.yaml**”, dentro del apartado “**dependencies**”, siendo las más importantes: **http**, **flutter explayer**, **encrypt** y **flappy search bar**. Además, **pubspec.yaml** no solo indica las dependencias, esta aplicación utiliza imágenes por defecto, por ejemplo:



Ilustración 1: Logo fondo blanco

Esta imagen **es LogoApp.png**, se encuentran en la carpeta de “**assets**” dentro de nuestro proyecto.

Flask dispone del fichero “**requirements.txt**” en el que se indican todas las dependencias. Estas se instalan en un entorno virtual de python ven, (por actualizar posteriormente el fichero sin introducir otras personales) que no se subirá al repositorio. Además, son necesarias las librerías “**libcurl4-openssl-dev libssl-dev libpq-dev**” instalables en debian y derivados.

#### Proceso de Despliegue:

Para **desplegar** la interfaz web, hay que ejecutar "ng build --prod", que generará una carpeta denominada "dist" y subir dicha carpeta. En este proyecto, hay un fichero JSON denominado "package.json" que especifica todas las versiones de las distintas dependencias empleadas y del código, por lo tanto, ambos integrantes tienen las mismas versiones al tener todo el proyecto en un repositorio de GitHub común.

Para realizar el despliegue en cloud de las aplicaciones de Angular y Flask es necesario primero que se pueden desplegar localmente mediante el fichero **docker-compose** del repositorio de backend.

1. Construir la imagen (*docker-compose build*) y despliegue (*docker-compose up*)
2. En caso de no surgir errores de construcción se realizarán las pruebas automáticas descritas en los procesos de pruebas.
3. Tras pasar las pruebas satisfactoriamente se procede a desplegar los contenedores en Heroku mediante **Heroku CLI** (instalación [12]). Para cada contenedor:
  - a. Login de Heroku (*heroku login*). Credenciales personales no reflejadas aquí (Contactar con Director de Proyecto)
  - b. Logueo y actualización de contenedor (*heroku container:login*, *heroku container:push web*)
  - c. Liberar nueva versión (*heroku container:release web*)

Para utilizar la **aplicación** en el móvil se ha de instalar un **archivo .apk**. Este archivo se genera desde la terminal de Android Studio. **Se ejecuta** “flutter build apk” en el directorio de nuestro proyecto. Tras ejecutar esto se habrá **creado** un **ejecutable .apk** que se copiará en los dispositivos compatibles y se instalará.

#### Proceso de Pruebas:

Las pruebas se **realizarán** de forma específica para cada **servidor** y la **base de datos** mediante scripts/ficheros que ejecutarán **las pruebas automáticamente**. En algunos casos se realizarán pruebas de forma manual para casos puntuales.

A la hora de **comprobarlo localmente**, se podrá probarlo en cada uno de los **diferentes entornos de trabajo** respectivamente a los grupos de aplicación Web y Móvil. Para probar la aplicación de forma conjunta se utilizará tanto **los navegadores web y dispositivos móviles compatibles** según los requisitos no funcionales y dichos dispositivos se comunican con el servidor web que estará desplegado en Heroku.

Las pruebas en el **servidor** consisten en **tests unitarios** que se ejecutan con la base de datos vacía, ya que los datos necesarios son introducidos durante los tests y luego se borran. Para ejecutarlos se usan los comandos de python específicos para tests unitarios, en este caso se usará:

```
$ python -m unittest discover -s test
```

Este comando buscará ficheros que importen la librería de tests unitarios de python, encontrando nuestro fichero de pruebas `test_server.py` y ejecutando su contenido.

Tras la ejecución, se muestra en pantalla el número de tests ejecutados y el número de tests exitosos y fallidos. El objetivo es que todos los tests sean correctos, lo que corresponde a que todas las peticiones realizadas al servidor han devuelto el resultado esperado.

## 3.2 Planes

### 3.2.1 Plan de gestión de configuraciones

Para el código que se desarrolla en el proyecto se han creado **3 repositorios en GitHub**. Dos de estos para la parte de Frontend de la aplicación y otra para el Backend y lógica del servidor. Los administradores de dichos repositorios son los encargados de coordinar cada apartado. En el caso del Frontend para **móvil el administrador será Luis García**, para frontend **web Álvaro García** y para administrar el repositorio de **Backend será Saúl Flores**. Para la documentación también se ha creado un repositorio de GitHub y utilizaremos las incidencias para llevar un control del proceso de **documentación**. El administrador de este repositorio es el **coordinador general** del proyecto, **Alberto Calvo**. Los administradores de dichos repositorios serán los encargados de gestionar las incidencias y llevar el control del desarrollo. Esto se hará semanalmente, una vez a la semana los coordinadores mirarán sus respectivos repositorios y comprobarán el avance o la actualización de las incidencias.

Los ficheros correspondientes a Frontend, tanto web como móvil, y Backend, código del servidor y la base de datos, serán nombrados según el **estándar SnakeCase**. La documentación que acompaña al proyecto también seguirá este estándar. Además, para la escritura de código se seguirá el **estándar** recomendado por **Google** en el caso de **Android** [13] y **Backend**, y el recomendado por la **W3Schools** [14] para **Frontend**.

Se va a utilizar la gestión de incidencias de GitHub para organizar y llevar un seguimiento del trabajo. **Semanalmente** los coordinadores repartirán el trabajo, indicando qué ha de hacer cada miembro esa semana, todos **los domingos**. Cuando se haya **completado** una incidencia, la persona que la haya terminado **adjuntará un comentario a la incidencia**, indicando que la ha terminado, se pondrá en contacto con el coordinador y este se encargará de valorar si se ha terminado y la cerrará.

En lo relacionado al código, las incidencias serán añadidas al repositorio por los administradores y se corresponderá al trabajo relacionado con el proyecto que se desarrollará esa semana, esto se realizará **todos los domingos**. Las incidencias estarán en su estado inicial antes de comenzar

el trabajo, este estado inicial es “To Do”. Cuando un miembro del grupo comience dicha incidencia, este mismo actualizará su estado a “In progress”. Una vez la haya terminado dicho trabajo, el coordinador pondrá dicha incidencia como finalizada, pasando esta incidencia al estado final “Done”, en el caso de cumplir los requisitos establecidos.

Los requisitos que ha de cumplir una incidencia para darse por finalizada se acordarán previamente entre el coordinador y los miembros de grupo. Estos serán requisitos de funcionalidad y estética, el código entregado ha de compilar y no interrumpir el desarrollo que el resto de los compañeros están llevando a cabo. Cada **issue** tendrá sus propios requisitos, estos se anotarán en la **descripción del issue** una vez haya sido creada por parte del coordinador.

En lo relacionado a la **documentación** el control de issues será similar, aunque en este caso sí que será necesaria una **revisión del coordinador**. Lo cual implica que las incidencias tengan un estado anterior al estado “Done”, este estado será “Revisión pendiente”. Para que una de estas incidencias pase al estado de “Done” ha de ser aprobada por el coordinador del proyecto. El director de proyecto es el encargado de actualizar los documentos del repositorio una vez estén terminados, ya que su modificación se lleva a cabo en Google Drive en un doc de Google.

Vamos a utilizar un **flujo de trabajo centralizado** para los cuatro repositorios de GitHub que empleamos en este proyecto. Trabajaremos sobre una sola rama, la **rama master** y todos los cambios se confirman en dicha rama. Se ha llevado a cabo esta decisión ya que los repositorios de GitHub cuentan con un máximo de 3 integrantes y se parten las tareas de forma que el número de conflictos se minimice.

### 3.2.2 Plan de construcción y despliegue del software

El software se construye y se prueba mediante los procesos indicados en el 3.1.3. En la etapa temprana del proyecto se planea hacerla de **forma unitaria** de cada repositorio sin interactuar con las demás partes. Una vez se han alcanzado las etapas intermedia y final del proyecto se espera poder construir y desplegar localmente de forma conjunta mediante varios contenedores. De esta forma, cada desarrollador puede desplegar mediante **docker-compose** su versión que esté modificando y **simultáneamente** la última versión estable de las otras partes incluyendo su propia base de datos. Con este método se asemeja más al entorno de producción en el que se despliega con contenedores, **Heroku**.

En Heroku se dispone de una cuenta gratuita creada por el Director de proyecto con un límite en cada contenedor de 500 MB de RAM y 1 worker. Se disponen de 450h de funcionamiento, contando que cada vez que se despliega, la aplicación sigue activa durante 30 min hasta que se suspende. Además, se dispone del addon Heroku Postgres en la versión “hobby” que permite almacenar 10000 filas y realiza mantenimientos eventuales no programables en los que se desactiva temporalmente (5 min aproximadamente).

Para almacenar las canciones se dispone de una cuenta de estudiante de **AWS S3** durante 12 meses con 5GB de almacenamiento, 20000 solicitudes GET y 2000 solicitudes POST. En ella se almacenarán las canciones mediante **buckets**.

Los despliegues e integración de todo el software están **programados semanalmente**. Se deberá haber probado localmente antes con la configuración de los contenedores establecidos. En cada reunión semanal se realizará una presentación de las nuevas funcionalidades desplegadas. Se valorará si la aplicación en ese momento es funcional respecto a los demás repositorios (para

poder desarrollar de forma estable) y en caso positivo será la que se mantendrá desplegada también en Heroku para realizar pruebas. De esta forma, se plantea una especie de integración continua un poco más relajada sin realizar entregas semanales.

### 3.2.3 Plan de aseguramiento de la calidad

Para atraer usuarios y tener un **diseño común** entre las dos interfaces (web y móvil) se han consultado varias guías de estilo y/o pautas para desarrolladores de diferentes aplicaciones relacionadas con la música, tales como Spotify, Shazam o Deezer. Con el fin de conseguir usuarios, se han seguido las guías de estilo para facilitar a que el cambio no sea tan brusco respecto a una aplicación de música habitual y sea más fácil identificar dónde se encuentra cada función disponible.

Una de las guías de estilo más empleada es la de **Spotify** [15] debido a que se trata de la mayor aplicación de reproducción de música y es la que posee una mayor base de usuarios. En este caso, se han seguido tanto la guía de estilo y algunas **pautas para desarrolladores** [16]. Al seguir sus recomendaciones, se han llegado a las siguientes conclusiones junto a las pantallas afectadas:

Si se muestra la imagen de un álbum, la **imagen** ha de ser **cuadrada** (pantallas de álbum, artista, listas de reproducción y todas aquellas que muestren como mínimo una canción)

En la barra superior, en la parte izquierda se mostrará el logo de la aplicación seguido de su nombre (presente en todas las pantallas)

El logo ha de encontrarse en un fondo blanco (#FFFFFF), negro (#000000) o que no esté en solo dos tonos diferentes (presente en todas las pantallas, destacando la de iniciar sesión y registro)

El **logo** sólo puede encontrarse de tres maneras: el logo con el nombre de la aplicación a la derecha (barra superior), con el nombre de la aplicación abajo (inicio sesión y registro) o solo el logo (imagen de la aplicación móvil o ícono de la pestaña en interfaz web)

Los colores predominantes serán el blanco (#FFFFFF), negro (#000000), los colores del logo (azul (#214B8B), morado (#833A8E)) y una escala de grises (presente en todas las pantallas)



Ilustración 2: Logo con nombre horizontal



Ilustración 3: Logo con nombre cuadrado



Ilustración 4: Logo sin nombre cuadrado

Otra guía de estilo consultada ha sido la de **Shazam** [17], con la que se han obtenido nuevas conclusiones:

La **tipografía** será ligeramente mayor para el nombre de la canción y menor para el nombre del artista cuando se muestren en forma vertical, es decir, el nombre de la canción y debajo el nombre del artista (barra inferior y pantallas de canción y álbum). También, será mayor cuando se muestre el título de una lista de reproducción en comparación con las canciones de esa lista

La tipografía predominante será *Roboto Mono* [18] y podrá presentarse en negrita (pantalla de canción en el nombre de la canción)

Para consultar varios de los **bocetos** de las pantallas, en el punto 4.2 se muestran antes de seguir las decisiones tomadas en este punto, por lo que las pantallas se verán modificadas en el resultado final, aunque la estética será parecida.

En cuanto a la **usabilidad**, todas las páginas principales de la aplicación se pueden acceder desde el menú de la izquierda o desde la barra superior, reduciendo la búsqueda del usuario al querer acceder a una página en concreto al encontrarse todo en dos sitios. Cuando no se ha iniciado sesión, en la pantalla información en el caso de la interfaz web, los botones que redirigen a las páginas de inicio de sesión y registro se encuentran en el centro de la pantalla para que sea fácil identificarlas y cueste menor tiempo encontrarlas. En el caso de la interfaz móvil, se accede directamente a la pantalla de iniciar sesión, a través de la cual se puede acceder a la página registro.

Ambas interfaces (web y móvil) presentan una **estética común**, permitiendo que el cambio de interfaz no sea tan brusco. Esta estética común se compone de una barra superior con páginas

y el logo con el nombre de la aplicación, a la izquierda un menú desplegable con las páginas principales y una barra inferior con la reproducción actual y permitiendo gestionar el audio, como parar la canción o pasar a la siguiente. En la barra superior se puede encontrar la página “Ayuda”, la cual contiene una serie de preguntas frecuentes que pueden ayudar al usuario a resolver la duda o problema surgido. Esta estética no se encuentra presente en las pantallas que se muestran cuando no se ha iniciado sesión, presentando solamente la barra superior.

En cada parte del proyecto, se harán una serie de **tests automáticos**, que pueden tratarse de acciones realizadas por un usuario o probando datos que podrían ser inválidos (introducir en la pantalla de registro valores que sobrepasen el límite o valores ya repetidos en la base de datos) y de pruebas **manuales**, que consistirán en la prueba de cada funcionalidad de forma más detallada por parte de los encargados de esa parte y explicándoles al resto del grupo qué es lo que han hecho, cómo funciona y si van a implementar alguna mejora. Si al grupo le ha convencido lo propuesto y no ha surgido alguna duda o mejora, se considera como finalizada. La función de dichos tests es para comprobar el correcto funcionamiento de las peticiones de la API y de si las entidades y sus relaciones en la base de datos son correctas y se pueden añadir o eliminar elementos. En el caso del Frontend, los tests se realizan para comprobar si los diferentes componentes/clases/pantallas funcionan como se espera (como el control del reproductor o el cerrar sesión del usuario).

En la parte de backend tienen una serie de pruebas para comprobar que las peticiones producen los resultados esperados. Todas las pruebas siguen el mismo formato, se añade un elemento a la base de datos, se ejecuta la petición a probar, se comprueba el resultado y se elimina ese elemento para restaurar la base de datos.

En cuanto a frontend, se comprueba que cada funcionalidad realiza lo esperado, como es la reproducción en bucle o aleatoriedad. Para comprobar las diferentes funcionalidades relacionadas con el reproductor, se precisa, como mínimo, de una canción. Como en backend, las pruebas siguen un formato, se prueba la funcionalidad, se comprueba y se prueba con un diferente número de canciones.

Cuando un miembro no líder da por **terminada una funcionalidad**, el líder de la parte correspondiente, se asegura de que funciona correctamente e informa de que dicha función ha sido completada o que falta alguna parte para finalizar. Para considerarla como finalizada, se emplean las **issues** de GitHub en el que se añade una descripción para enfatizar las acciones que hay que realizar para completarla y cerrarla. Aunque se haya aceptado, quedará pendiente la demostración al resto del grupo explicada en el apartado anterior.

Un código limpio y ordenado es más fácil de comprender y de encontrar fallos, por lo tanto, para cada parte se empleará un **analizador estático** y **formateadores de código**, asegurándose de tener un código con la menor cantidad de errores o avisos. En la parte de Frontend web se va a emplear Prettier [19] como formateador de código y codelyzer [20] como analizador estático, en Frontend Android se ha escogido dartanalyzer [21] como analizador estático y en cuanto al formateador de código se usa el comando “dartfmt” y en el backend el propio IDE PyCharm reorganiza el código y como analizador estático se emplea el plugin Pylint [22].

Para llevar la cuenta de la cantidad de líneas de cada tipo de lenguaje, teniendo en cuenta líneas en blanco y comentarios, se emplea la herramienta “cloc” [23] con la carpeta que contiene los ficheros como parámetro.

Otra parte importante del proyecto es la **documentación**, por lo que hay que tener en cuenta el diseño de esta memoria. Para ello, cuando cada miembro ha finalizado sus partes asignadas, el líder del proyecto revisa cada apartado, saca defectos y se los comunica a las personas implicadas. Cuando no encuentre ningún defecto, se pasa toda la memoria a LaTeX para conseguir un diseño más formal.

### 3.2.4 Calendario del proyecto y división del trabajo

Para la organización del trabajo del equipo, se ha realizado un diagrama de Gantt. En este diagrama se planifican las *issues* publicadas en GitHub para cada una de las partes del proyecto: frontend web, frontend móvil y backend. Por ello, si se quiere buscar una explicación/descripción de la *issue* para conocerla más en detalle, simplemente hay que buscarla en el repositorio correspondiente del proyecto.

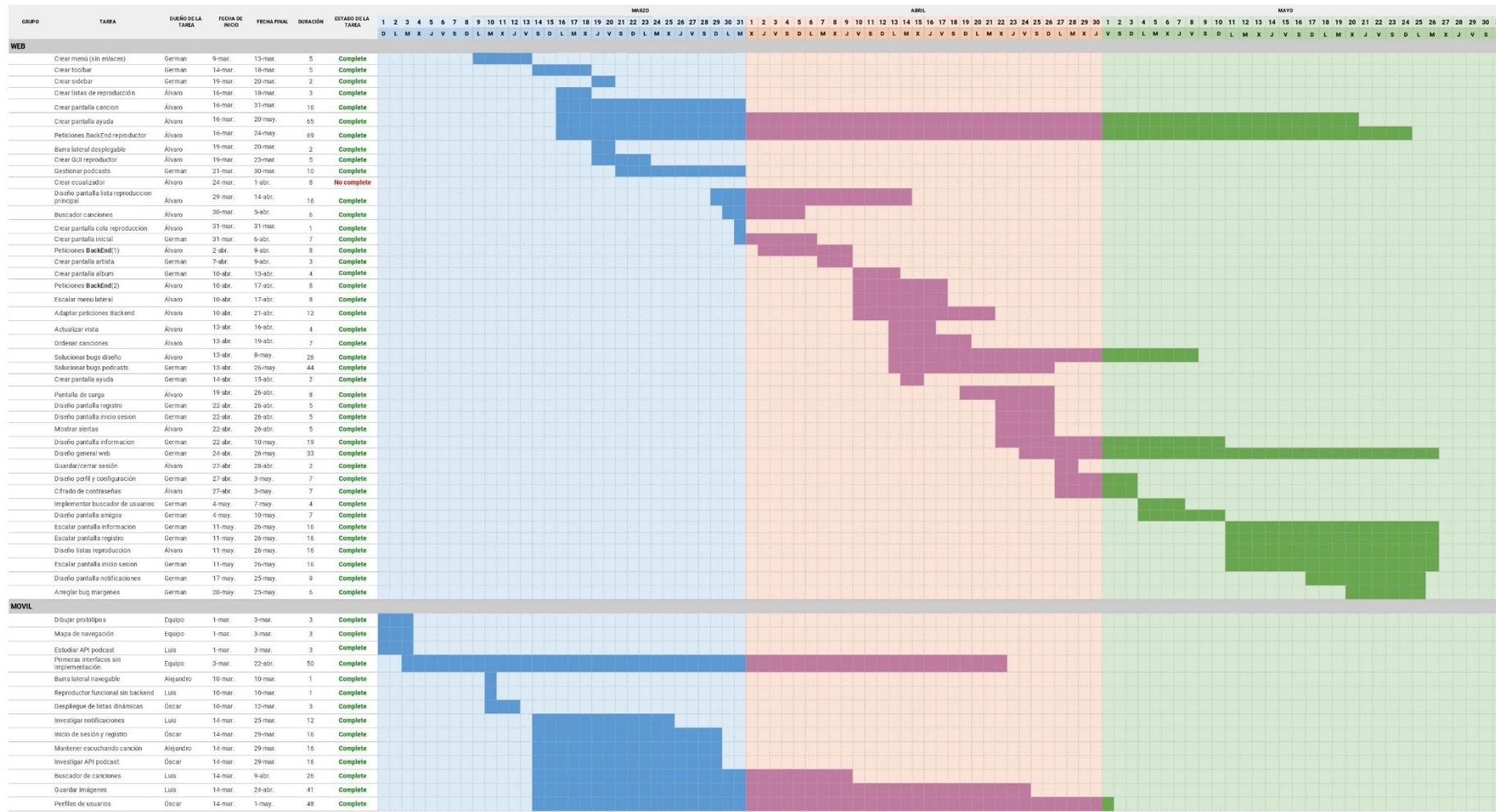
Se ha establecido que, para la primera entrega, deben estar realizados todos los requisitos relacionados con un reproductor de música o podcasts y sus equivalentes, por ejemplo, el crear listas de reproducción, el poder buscar podcasts... Mientras que para la segunda entrega se ha decidido que se completarán los requisitos restantes, es decir, aquellos requisitos relacionados con los usuarios y las interacciones entre ellas.

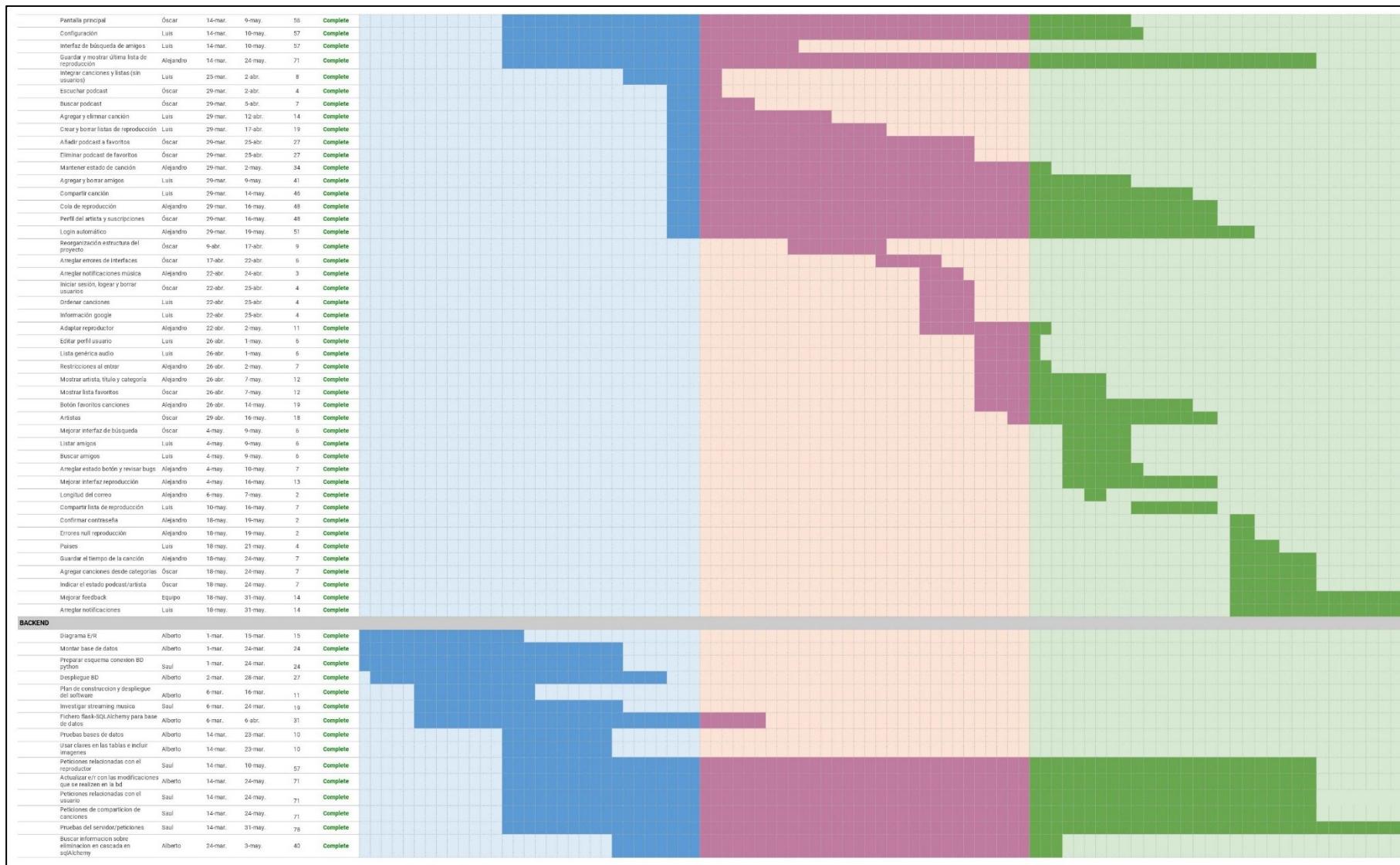
En términos de requisitos, para la **primera entrega** se tiene previsto tener completados del RNF1 al RNF4, del RNF8 al RNF 10, del RF1 al RF5, RF13 y del RF32 al RF34. Y para la **segunda entrega**: del RF6 al RF12 y del RF14 al RF31.

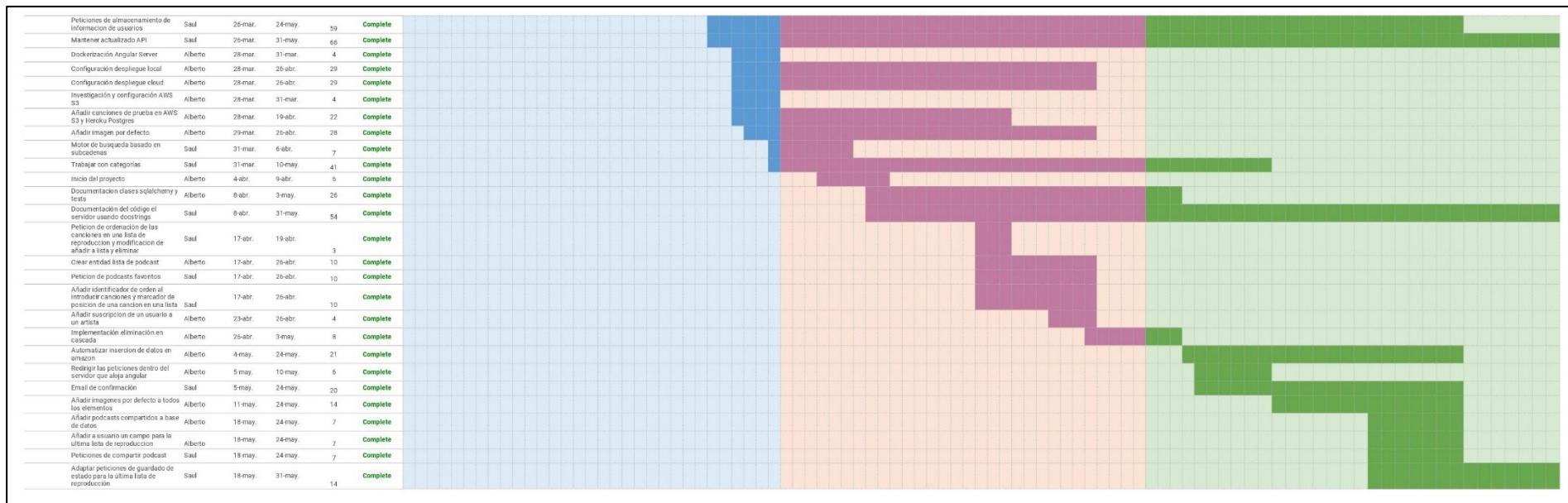
## DIAGRAMA DE GANTT

TÍTULO DEL PROYECTO Tunefit  
PROJECT MANAGER Alberto Calvo

NOMBRE COMPAÑIA Karen Spanck Jones  
FECHA 10/2/20







## 4 Análisis y diseño del sistema

### 4.1 Análisis de requisitos

#### 4.1.1 Requisitos funcionales

- RF1.* Se permite reproducir canciones y podcasts
- RF2.* Se permite la reproducción en bucle, aleatoria, empezar, pausar y avanzar a la anterior o siguiente canción/podcast
- RF3.* En una lista de reproducción se puede filtrar por categoría
- RF4.* Una canción/podcast se compone de un audio, un título, un artista, un álbum y una categoría
- RF5.* Una lista de reproducción está compuesta por una serie de canciones en un orden específico, creadas por un usuario o por el sistema, que contiene un título y una descripción
- RF6.* Los usuarios pueden crear o eliminar listas de reproducción
- RF7.* Los usuarios pueden añadir o eliminar una canción de una lista de reproducción propia previamente creada
- RF8.* Los usuarios pueden ordenar las canciones presentes en una lista de reproducción propia manualmente y por artista y título
- RF9.* Los usuarios poseen una lista de canciones/podcasts favoritos
- RF10.* Los usuarios pueden añadir o eliminar canciones/podcasts de favoritos
- RF11.* Los usuarios pueden buscar canciones/podcasts
- RF12.* Los podcasts se agrupan en series de distintos episodios
- RF13.* Un usuario está conformado por un nombre de usuario, un correo electrónico único, una contraseña para acceder, una fecha de nacimiento, su país y foto de perfil
- RF14.* Los usuarios se pueden registrar en el sistema
- RF15.* Los usuarios registrados pueden iniciar sesión
- RF16.* Los usuarios pueden buscar a otros usuarios
- RF17.* Los usuarios pueden agregar a otros usuarios como amigos
- RF18.* El sistema permite consultar los usuarios amigos
- RF19.* El sistema permite eliminar a otros usuarios amigos
- RF20.* El sistema permite consultar las peticiones de amistad recibidas
- RF21.* Los usuarios pueden compartir canciones/podcasts o listas de reproducción con otro usuario amigo
- RF22.* Los usuarios pueden agregar a sus listas de reproducción otras listas creadas por otros usuarios
- RF23.* Los usuarios pueden modificar su nombre de usuario, su contraseña, su país y su foto de perfil
- RF24.* Los usuarios pueden eliminar su cuenta
- RF25.* Los usuarios se pueden suscribir a los artistas
- RF26.* Se guarda el tiempo en el que se ha cerrado la aplicación de la última canción
- RF27.* Se guarda la última lista reproducida
- RF28.* La reproducción de canciones se sincronizará entre dispositivos web/android de un mismo usuario
- RF29.* Se recupera el tiempo de la última canción escuchada cuando se abre el sistema
- RF30.* Se recupera la última lista de reproducción cuando se abre el sistema

- RF31.* El sistema permite buscar información externa asociada a una canción/podcast mediante Google
- RF32.* La aplicación permite listar canciones/podcasts
- RF33.* La aplicación mostrará el estado de la canción/podcast
- RF34.* El usuario al registrarse debe confirmar un correo para poder iniciar sesión

#### 4.1.2 Requisitos no funcionales

- RNF1.* El sistema tendrá una versión para Android y otra para Web
- RNF2.* El sistema soportará la versión de Android 6.0 y de la Web los navegadores Chrome (78.0.3904.108) y Firefox (75.0)
- RNF3.* El sistema se conectará a un servidor para obtener las canciones/podcasts
- RNF4.* El sistema necesita conexión a Internet
- RNF5.* El nombre del usuario debe tener entre 3 y 50 caracteres
- RNF6.* Las contraseñas del usuario se encuentran cifradas
- RNF7.* Los usuarios inician sesión mediante el correo electrónico y la contraseña
- RNF8.* Se muestran las últimas canciones añadidas al sistema
- RNF9.* El reproductor en la parte inferior es fijo. Cuando no se ha seleccionado una canción/podcast no deja interactuar
- RNF10.* La búsqueda de canciones/podcasts se realiza mediante palabras clave con un mínimo de una palabra y una longitud mayor o igual a 3 caracteres
- RNF11.* La contraseña de los usuarios debe tener 7 o más caracteres como mínimo, 1 número mínimo y no debe tener caracteres especiales (rangos ASCII 48-57 números, 65-90 mayúsculas, 97-122 minúsculas)
- RNF12.* El correo electrónico no debe sobrepasar los 50 caracteres

## 4.2 Diseño del sistema

### 4.2.1 Diagrama de despliegue

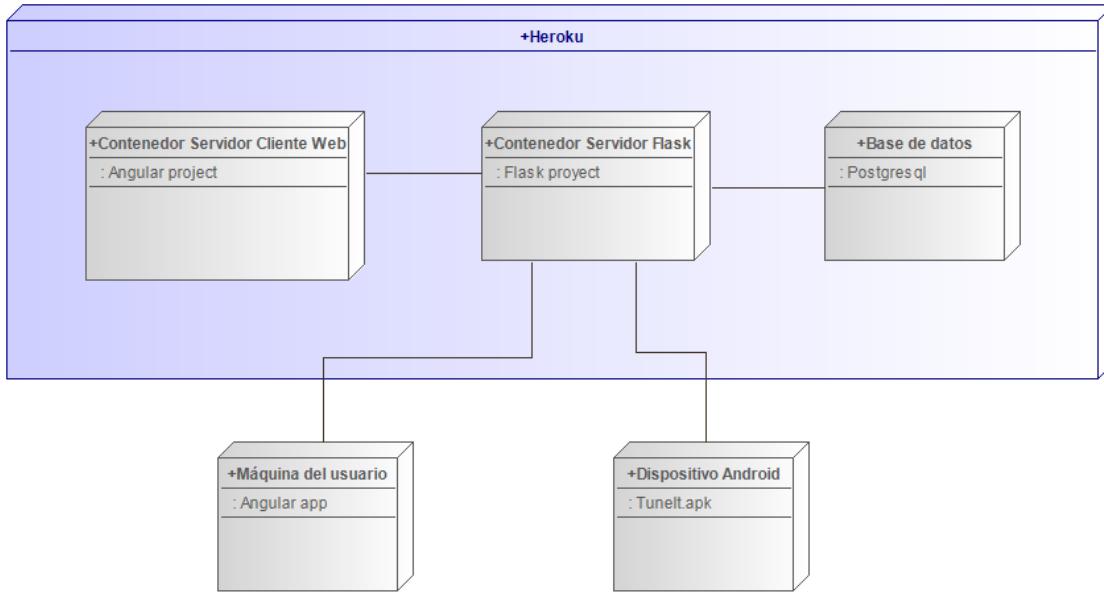


Ilustración 5: Diagrama de despliegue

El protocolo empleado para la comunicación entre Backend y Frontend será HTTP, usando los métodos estándar de HTTP.

Las tecnologías elegidas son variadas. Para el Backend se han elegido Flask , SQLAlchemy y ,para la base de datos, PostgreSQL. Flask ha sido elegido ya que se trata de un framework en python que permite el desarrollo de proyectos de forma sencilla, rápida y en muy pocas líneas de código. SQLAlchemy ha sido elegido ya que permite interactuar con la base de datos de forma mucho más simple. Evita utilizar lenguaje SQL y ayuda a pasar las relaciones y clases de la base de datos a Python, de esta forma se simplifica la interacción con Flask. Para el desarrollo del Frontend web se va a utilizar el framework Angular, desarrollado en TypeScript, de código abierto y mantenido por Google. Los lenguajes con los que trabajaremos en Angular son principalmente TypeScript junto a HTML y SCSS.

Finalmente, para desarrollar el Frontend móvil utilizaremos Flutter. Es un SDK código fuente abierto de desarrollo de aplicaciones móviles creado por Google que emplea el lenguaje Dart. Haremos uso de una API web externa con la que se conectará nuestro sistema. Utilizaremos la API que ofrece Listen Notes [6], esta API nos permite realizar hasta 5000 consultas de forma totalmente gratuita, de esta forma tendremos acceso a los últimos podcasts que se hayan publicado.

En cuanto a la base de datos, se va a utilizar una base de datos Postgresql, de tipo relacional, ya que no se va a trabajar con grandes cantidades de datos, ni se tiene que escalar horizontalmente ni se necesitan todas las otras ventajas que ofrece una base de datos NoSQL frente a las SQL.

A la hora de almacenar la información, la mayoría de los datos se almacenarán en la base de datos, mientras que las canciones, que son los elementos más pesados, se almacenarán en un servicio cloud de almacenamiento de Amazon S3, guardando en la base de datos únicamente la

ruta necesaria para poder alcanzarlas, que es además lo que se necesita para poder realizar el streaming.

La API no va a ser de tipo RESTful, es suficientemente sencilla como para que no sea necesario, aunque sí contará con algunas de sus características, como ser cliente-servidor y no guardar estado, sino que las peticiones contengan la información necesaria para llevarlas a cabo.

La seguridad de la información de los usuarios que se va a almacenar se asegurará cifrando las contraseñas de estos, de forma que nunca se tratan en la base de datos las contraseñas planas, sino que se trabajará con los hashes.

El despliegue de la aplicación se realizará en Heroku [24], donde se alojarán 2 contenedores, uno donde se instala el servidor Flask y otro para el servidor que se usará de host para el cliente web en angular. La base de datos Postgresql se despliega mediante el addon Heroku Postgres.

Se accede a la aplicación mediante los navegadores web instalados en los equipos a través de la url que Heroku proporcionará al servidor web. Para la aplicación Android, se simulará el cliente en los equipos de los desarrolladores mediante las herramientas que ofrece Android studio y una vez haya madurado lo suficiente, se instale en algún dispositivo de forma que no provoque errores graves.

#### 4.2.2 Diagrama de clases

##### 4.2.2.1 Cliente Web

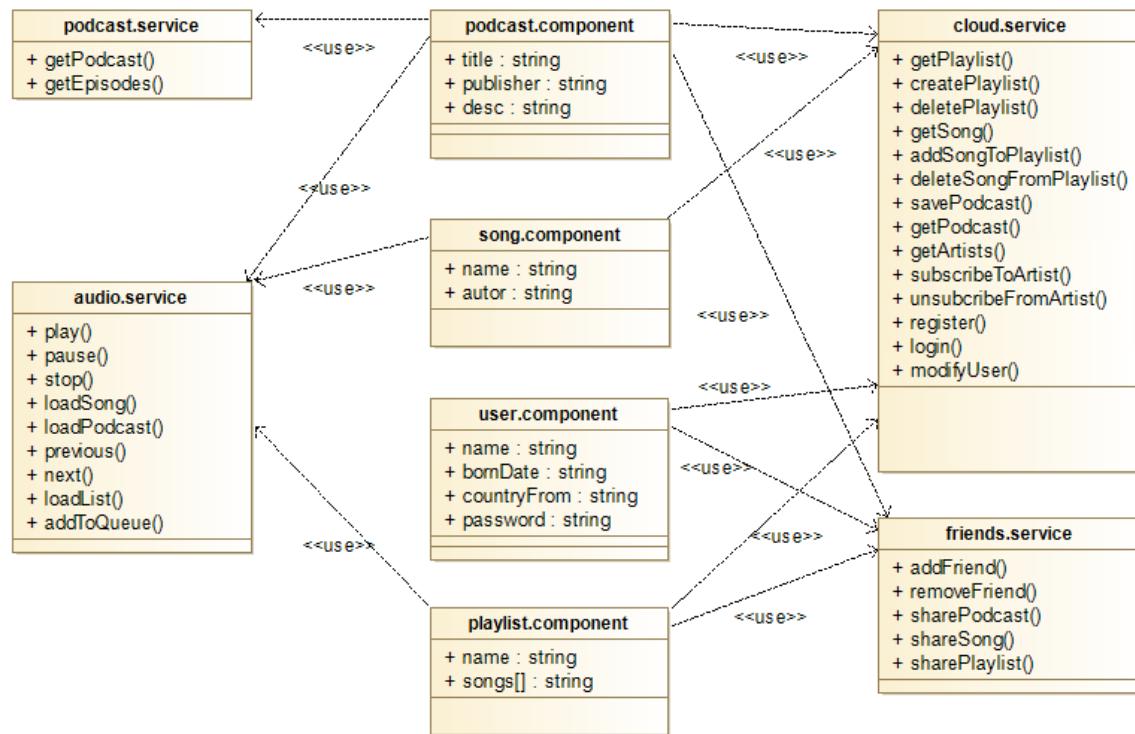


Ilustración 6: Diagrama de clases web

Observando el diagrama se puede comprobar que el proyecto se basa en “componentes” y “servicios”. Estos “servicios” se encargan ofrecer la interactividad con la API de BackEnd y la API de ListenNotes. Por otro lado, los “componentes” se encargan de mostrar la información al usuario de la web.

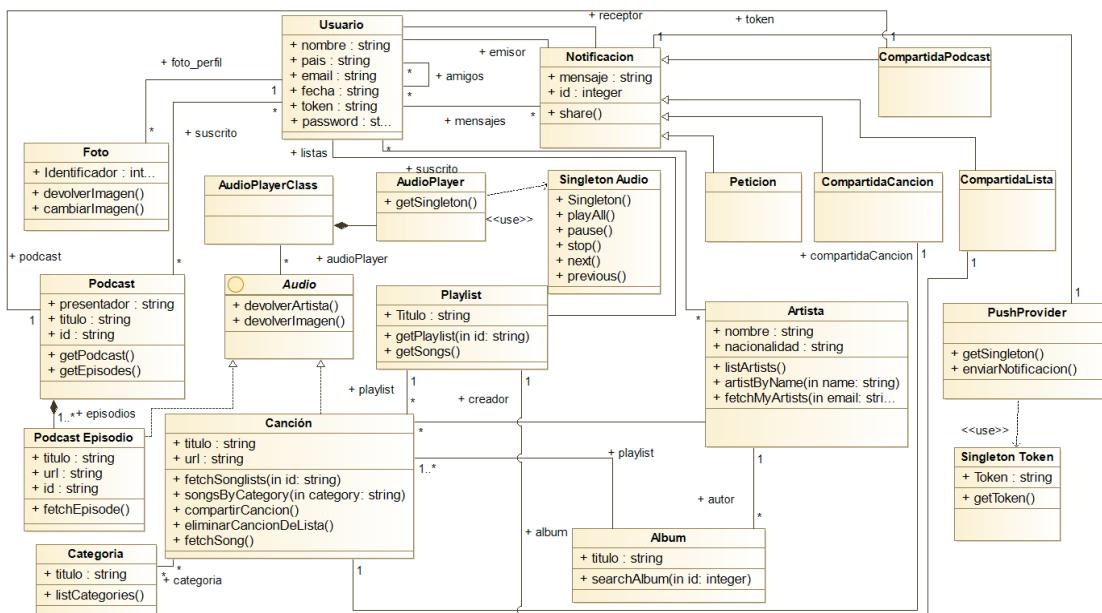
“podcast.service” es la clase que se encarga de comunicarse con la API de ListenNotes y ofrecer los resultados de la comunicación a “podcast.component” el cual se encarga de mostrar al usuario el resultado de la petición realizada.

“audio.service” es la clase encargada de reproducir las canciones, podcasts o playlists, en este caso no existe ninguna comunicación externa.

“friends.service” enlaza web con backend respecto al tema de las relaciones entre usuarios, ofrece las diferentes interacciones entre usuarios (añadirlos, borrarlos, compartir cosas con ellos...).

“cloud.service” es la clase que usa el resto de funciones de la API de back end, se encarga de gestionar toda la creación y modificación de playlists, de elementos favoritos, de suscripciones a artistas y de gestionar el login, el registro y los usuarios.

#### **4.2.2.2 Cliente Android**



*Ilustración 7: Diagrama de clases Android*

El diagrama mostrado se corresponde al diagrama de clases utilizado en el frontend Android. La clase Usuario se corresponde a la clase que guarda los valores que identifican al usuario que está usando la aplicación en el teléfono móvil en ese momento. Cada usuario puede recibir notificaciones de otros usuarios, estas notificaciones pueden ser peticiones de amistad, listas compartidas, canciones compartidas o Podcasts. Esto se ve reflejado en la clase Notificacion y sus especializaciones Peticion, CompartidaCancion, CompartidaLista y CompartidaPodcast.

Cada usuario cuenta con una lista de la clase Notificacion y la clase Notificacion tiene un emisor y un receptor que son una instancia de la clase Usuario. Para la reproducción de música y podcast ha sido necesario aplicar el patrón Singleton a nuestro diseño. Este patrón se ha utilizado con la clase AudioPlayer que es la clase que nos permite reproducir audio en Flutter. De esta forma en toda la aplicación se utiliza la misma instancia de la clase AudioPlayer así podemos reproducir audio en cualquier pantalla.

La clase `AudioPlayerClass` contiene una lista de instancias de la clase `Audio` y la instancia única de la clase `AudioPlayer`. La clase `Audio` es la generalización de `Canción` y `Podcast Episodio` que

son el contenido que podemos reproducir en el móvil. La clase Canción se encuentra relacionada con las clases Categoría, Álbum y Artista. La clase Álbum identifica a qué álbum pertenece la canción y la clase Artista identifica a cuál es el artista de esta. Cada usuario tiene sus propias listas de reproducción que este ha creado para escuchar su música y cuenta con una lista de podcasts a los que se ha suscrito. Las listas de reproducción del usuario están representadas en la clase Playlist. Finalmente, los podcasts asociados a un usuario se encuentran representados en la clase Podcast.

En la clase Podcast se encuentran las funciones que interaccionan con la API externa proveedora de podcasts, en este caso Listen Notes [25]. En esta clase tenemos las funciones que nos permiten buscar y mostrar los podcasts disponibles, esta API nos devuelve la información asociada a un podcast. Esta información es el título, los autores, una descripción, un identificador único y la lista de sus episodios. La clase Notificacion interacciona con nuestra API para obtener la información de los mensajes enviados al usuario y utiliza la clase PushProvider para enviar una notificación a tiempo real al dispositivo móvil del usuario receptor. PushProvider interacciona con la API externa de Firebase [26]. El funcionamiento es el siguiente, cada usuario tiene un token que se presenta al dispositivo móvil que está utilizando a través de la API de Firebase podemos hacer que a ese dispositivo se le envíe una notificación a tiempo real gracias al uso del token. Para tener acceso al token en cualquier parte de nuestra aplicación se ha aplicado el patrón **Singleton** a la clase Pushprovider.

A continuación, se muestra la agrupación de estas clases en paquetes.

#### 4.2.2.3 Diagrama de paquetes

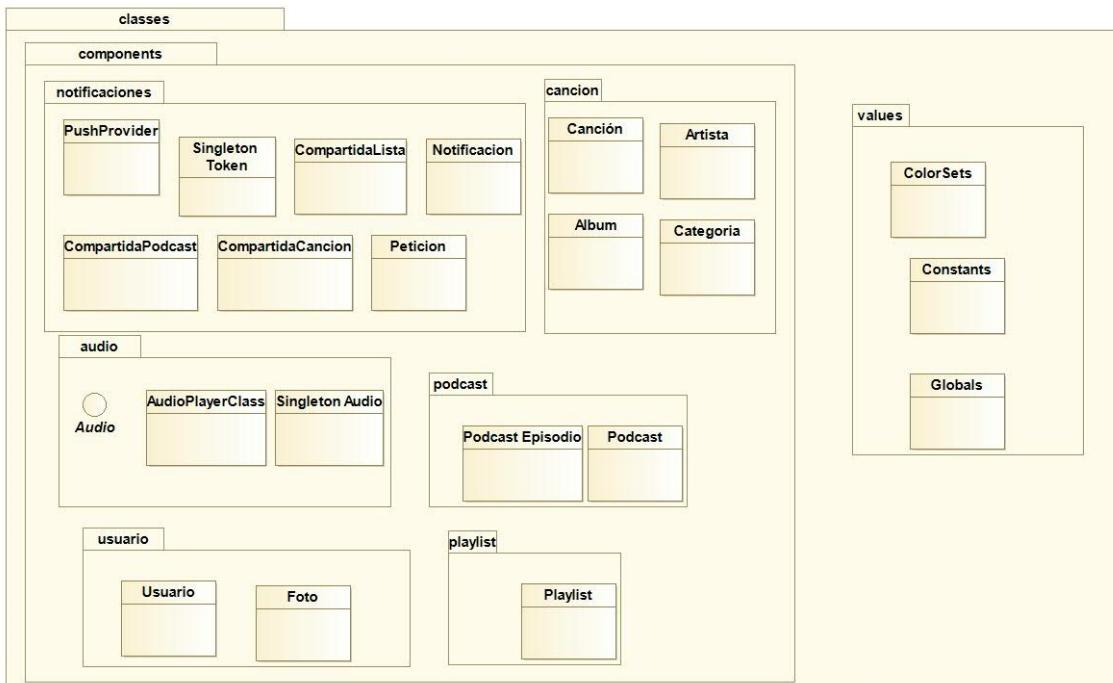


Ilustración 8: Diagrama de paquetes Android

El paquete values contiene las clases ColorSets, Constants y Globals. Estas clases se utilizan en toda la aplicación. ColorSets es una clase que tiene contiene los códigos de los colores que empleamos en las interfaces. Constants contiene el texto que aparece varias veces repetido en la aplicación. Globals cuenta con variables globales que se mantienen durante toda la aplicación.

#### 4.2.2.4 Servidor

Los datos de la aplicación se almacenan en una base de datos PostgreSQL, para acceder y tratar estos datos se ha usado un mapeador objeto-relacional SQLAlchemy. El modelo de datos que se usa en el servidor es una traducción a clases del esquema entidad-relación (Ilustración 10: Diagrama Entidad-Relación) presente en la base de datos.

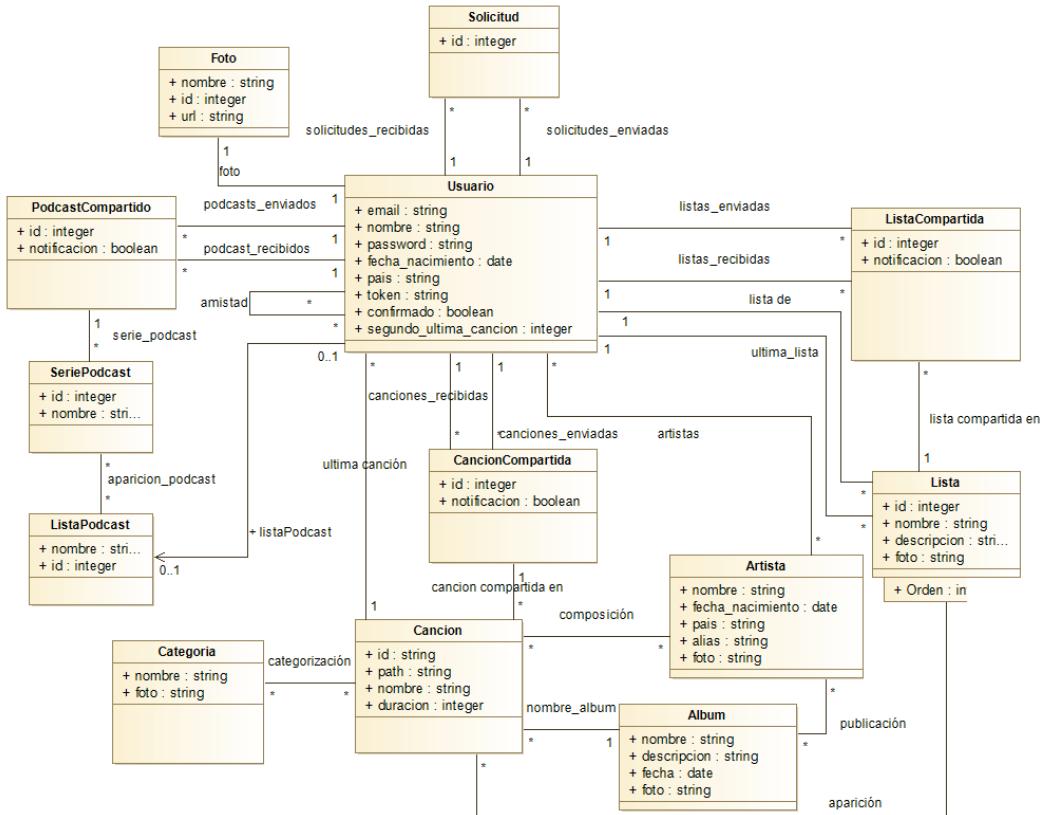


Ilustración 9: Modelo de datos Backend

Estas clases son utilizadas por una instancia de la clase SQLAlchemy a través de una “session”. En una “session”, todos los cambios en la base de datos se aplican al hacer commit y, en caso de error durante el commit, se hace rollback para mantener la consistencia.

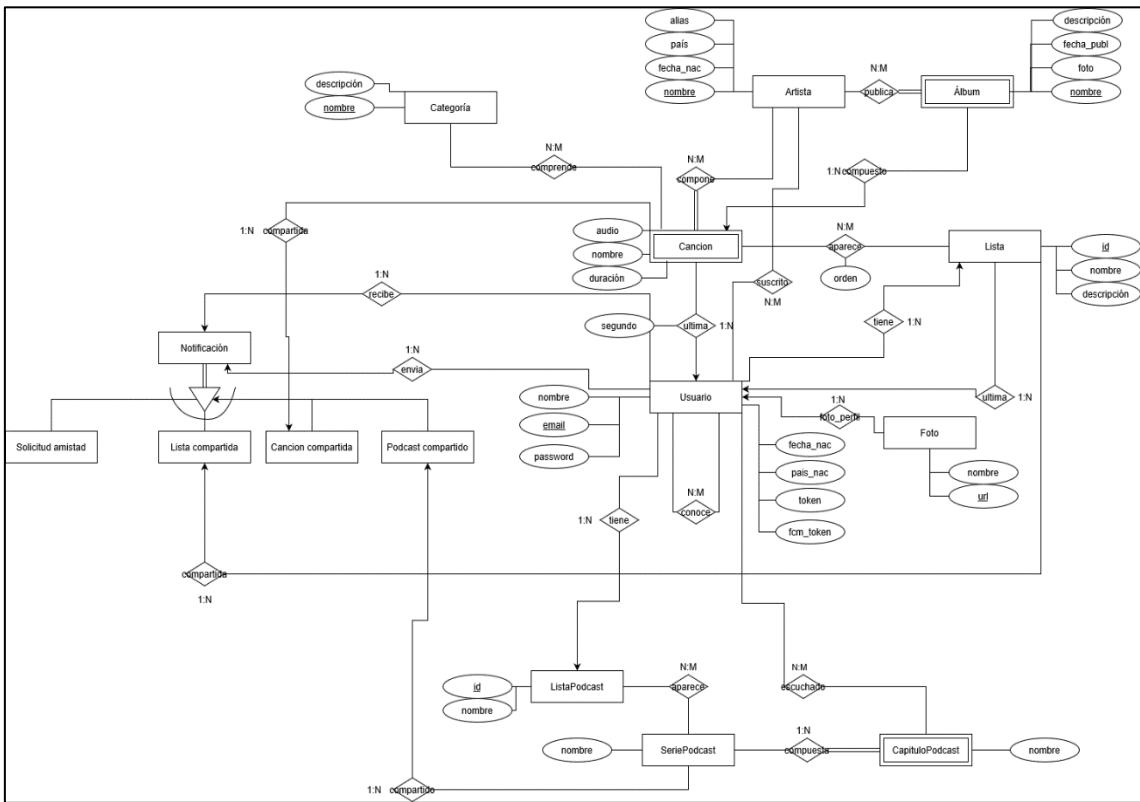


Ilustración 10: Diagrama Entidad-Relación

La clase principal de la aplicación es APP, una instancia de la clase Flask que contiene la aplicación y es la encargada de recibir y responder las peticiones. Se lanza el servidor flask con el método run, y route permite enlazar url con funciones en python que tratan la petición a dicha url.

La clase CORS es la que permite las peticiones cruzadas para angular, en nuestro caso solo hace falta aplicar el constructor a la clase APP para que esto suceda.

La clase Mail es la que se utiliza para enviar correos a través de Flask. Se crea al inicializar el servidor una instancia de correo que, posteriormente, enviará el correo de confirmación a los usuarios que se registran.

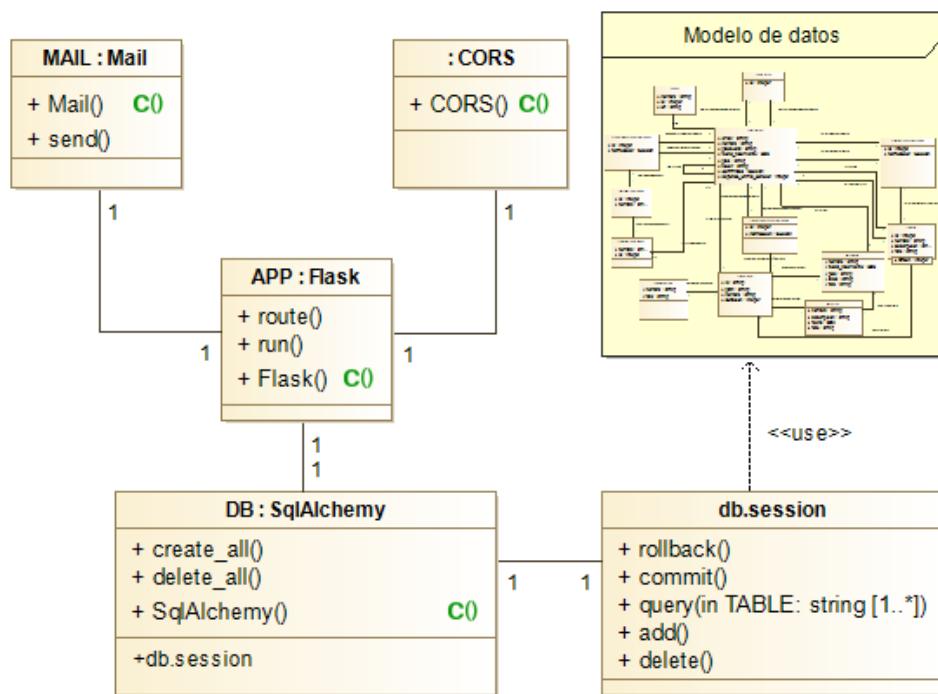


Ilustración 11: Diagrama de clases servidor

Los componentes del servidor se distribuyen en dos paquetes pertenecientes a un proyecto Flask en Python:

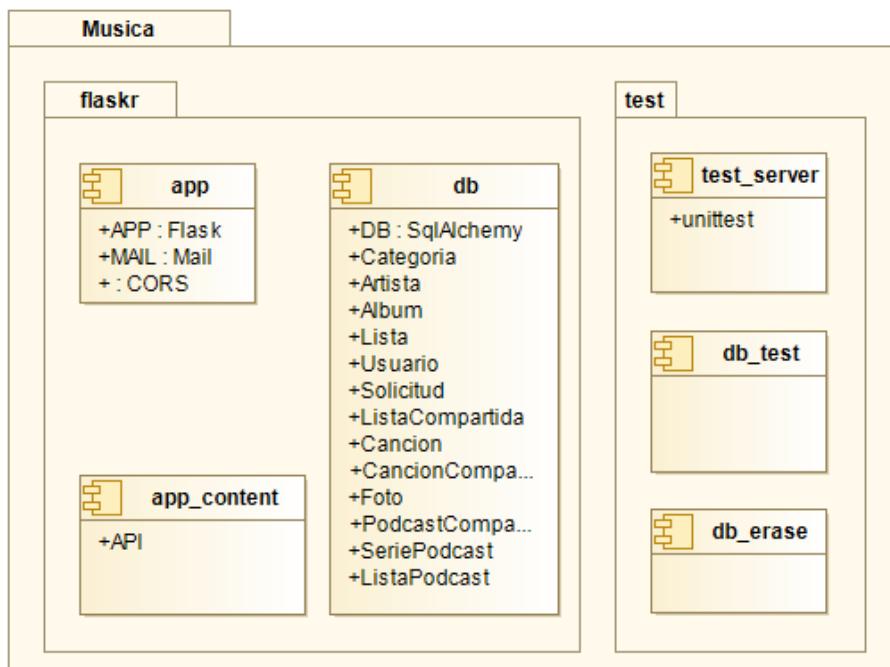


Ilustración 12: Diagrama de paquetes servidor

Música es el nombre del proyecto Flask en python sobre el que se ha trabajado.

Los paquetes son flaskr, que contiene la aplicación flask, y test, que contiene los distintos archivos que permiten probar tanto la base de datos como la aplicación.

En el `__init__` del paquete flaskr se define un constructor de aplicación, en este constructor se establecen los parámetros de configuración de la aplicación y se crean las instancias de CORS y MAIL; posteriormente la aplicación se crea en db donde se crea a su vez la clase DB para la base de datos y se inicializa el modelo de datos.

La API se establece en el fichero app\_content. En este fichero se crean las funciones que tratan las peticiones y se enlazan con las url correspondientes. Se encuentran en este archivo también las funciones auxiliares utilizadas por aquellas que responden a las peticiones.

En el paquete test se encuentran 3 ficheros. El archivo db\_erase es un pequeño script que borra la base de datos y se ha utilizado a la hora de hacer despliegues o solucionar errores. db\_test es un script que se encarga de poblar la base de datos con datos artificiales cuya función es poder probar la base de datos durante el proceso de desarrollo y poder mostrar los resultados a los profesores. Test\_server contiene distintos tests que permiten probar la corrección de las peticiones y su correcto funcionamiento.

#### 4.2.3 Componentes y conectores

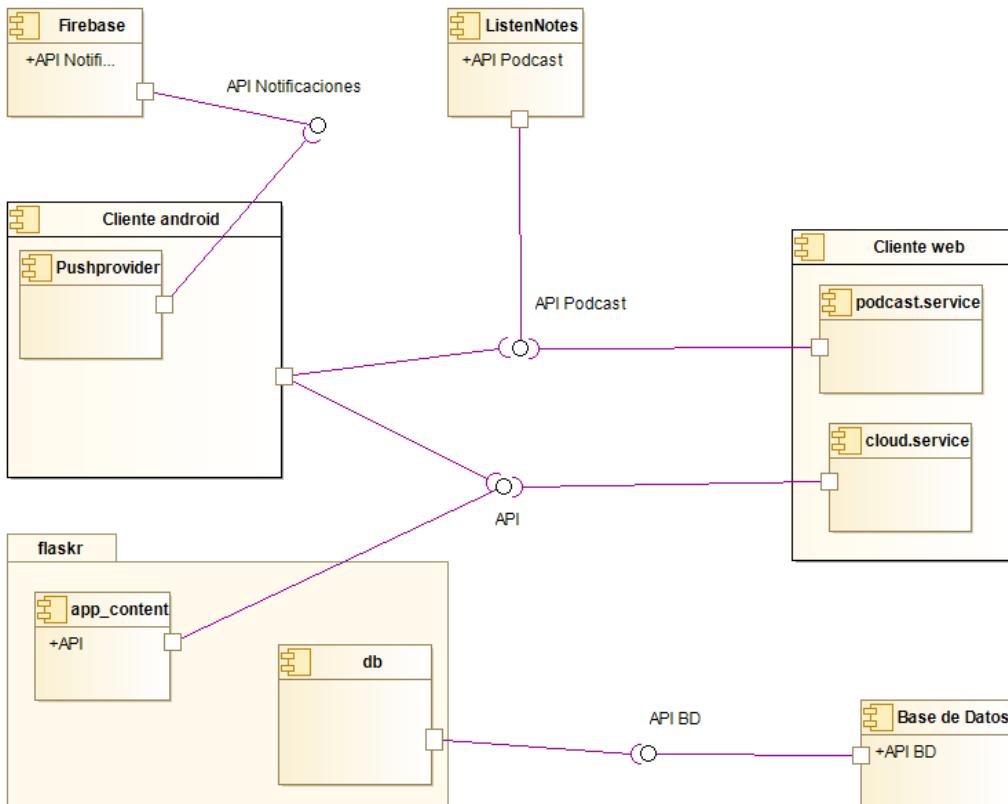


Ilustración 13: Diagrama de componentes y conectores

Los componentes principales de nuestra aplicación son los mostrados en la imagen 4.2.5, donde se visualiza también que existen 4 API en uso. De estas 4 API, 3 son externas, estas son ListenNotes, una api de podcast que es la que nos permite acceder a los podcast para mostrarlos y reproducirlos; Firebase, una api utilizada por la aplicación android para enviar notificaciones a los dispositivos y la API de la base de datos con operaciones de consulta y modificación de los datos.

La última API que aparece es la API web que se ha desarrollado dentro de nuestro proyecto y que contiene las peticiones http que los frontends(web y android) pueden realizar contra el

servidor. La documentación de las peticiones con sus parámetros de entrada y salida y una pequeña descripción se encuentra en una wiki del repositorio de documentación de nuestro proyecto [27]

#### 4.2.4 Interfaces

A continuación, añadimos las pantallas que se diseñaron al iniciar el desarrollo y a su vez las pantallas con el resultado final.

##### 4.2.4.1 Android

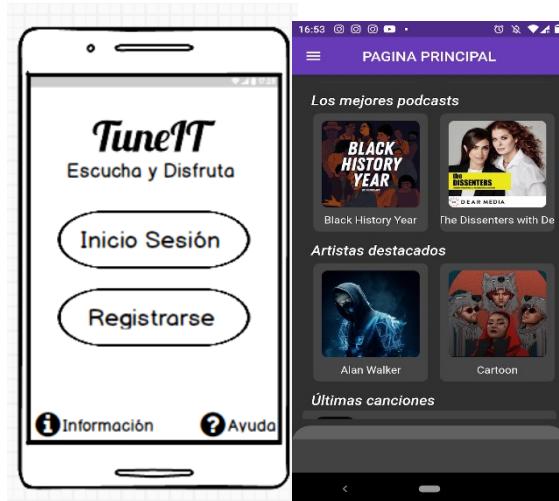


Ilustración 14: Pantalla inicial Android

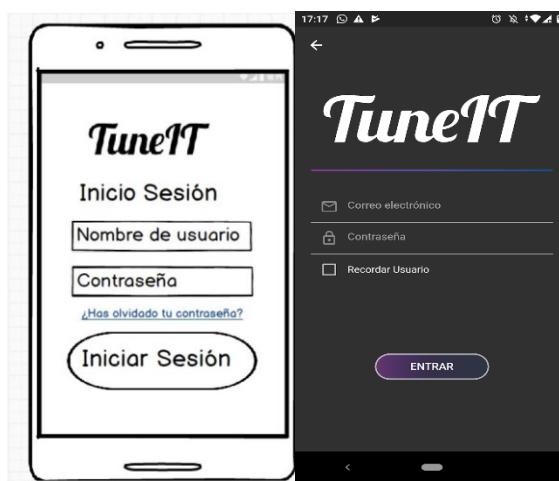


Ilustración 15: Pantalla inicio sesión Android

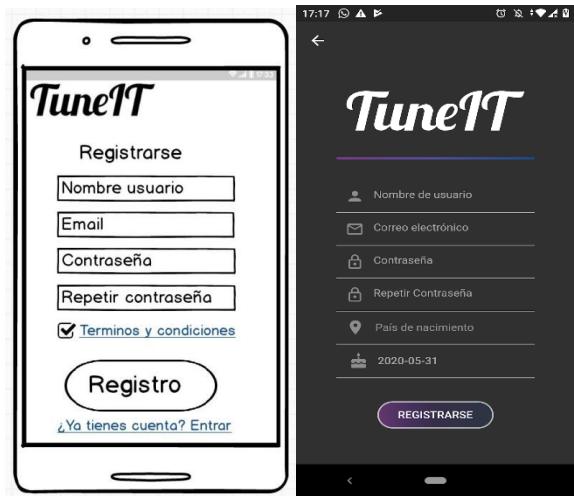


Ilustración 16: Pantalla registro Android



Ilustración 17: Pantalla principal Android



Ilustración 18: Pantalla listas de reproducción Android

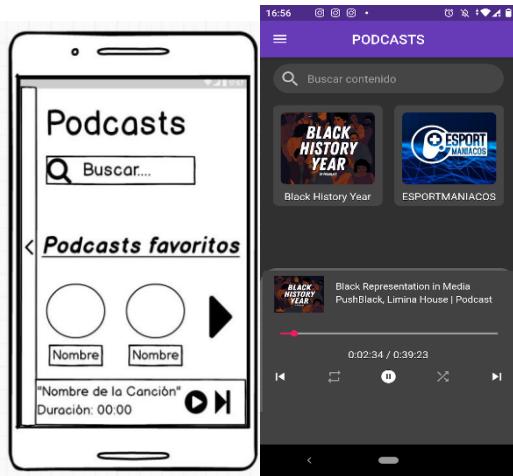


Ilustración 19: Pantalla podcast Android



Ilustración 20: Pantalla lista de canciones/podcast Android

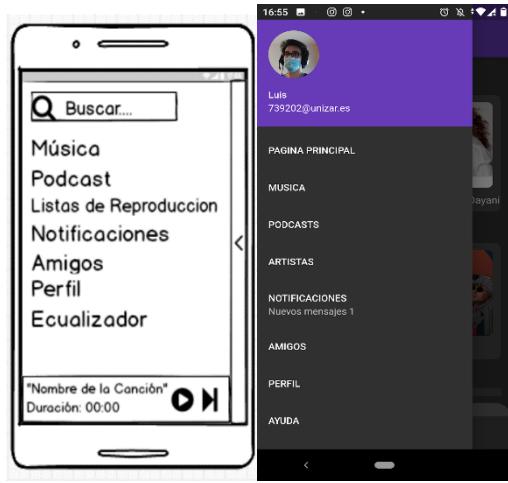


Ilustración 21: Pantalla desplegable lateral Android

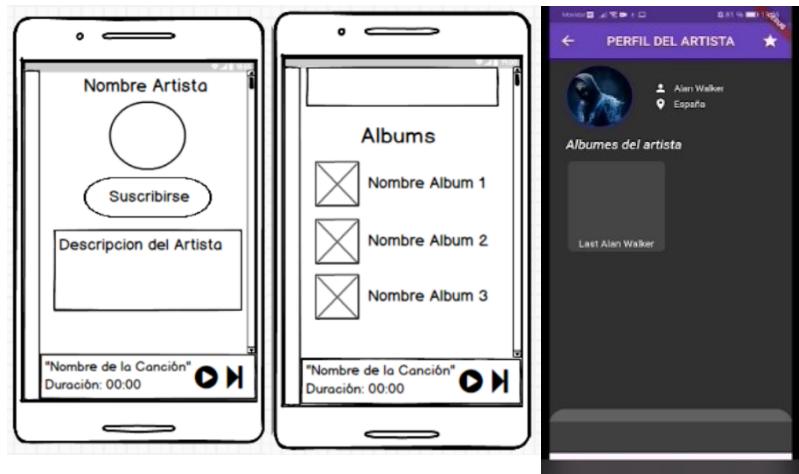


Ilustración 22: Pantalla artista Android



Ilustración 23: Pantalla configuración Android

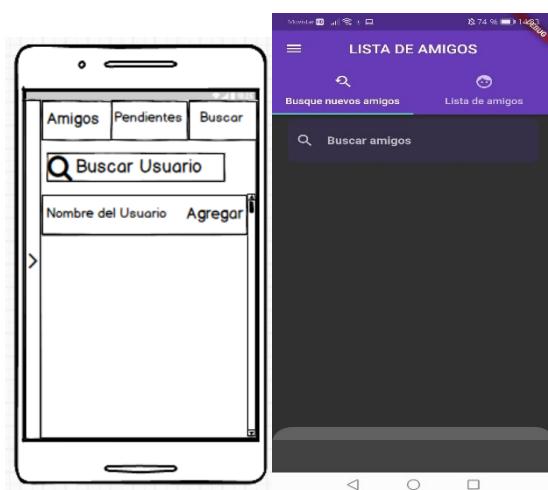


Ilustración 24: Pantalla búsqueda amigos Android

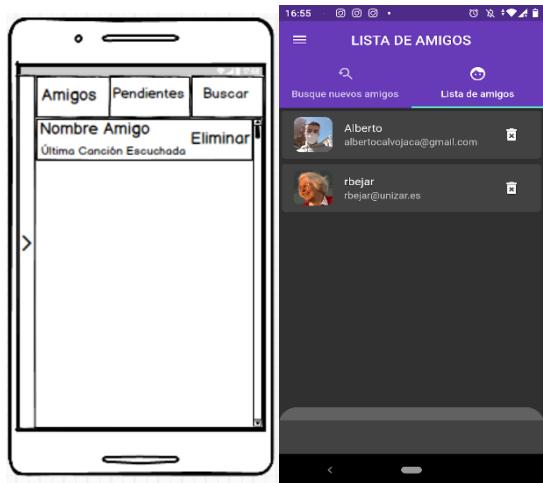


Ilustración 25: Pantalla lista de amigos Android

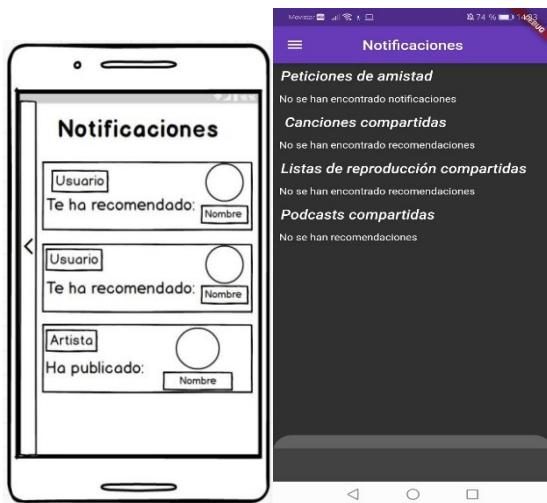


Ilustración 26: Pantalla notificaciones Android

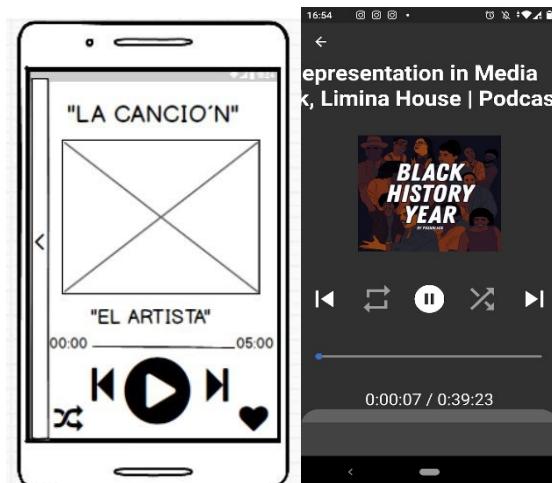


Ilustración 27: Pantalla reproductor Android

#### 4.2.4.2 Web

##### Pantalla inicial

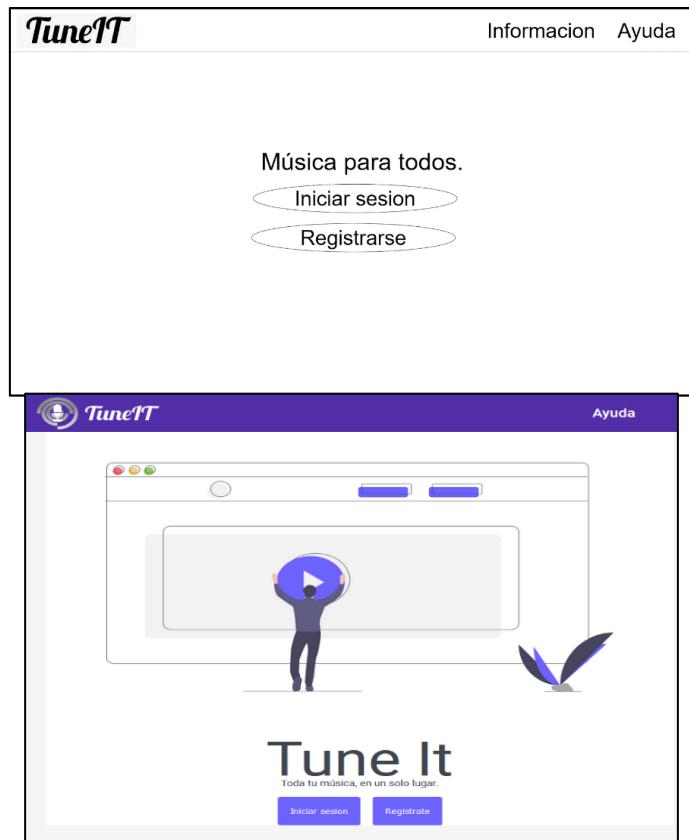


Ilustración 28: Pantalla inicial Web

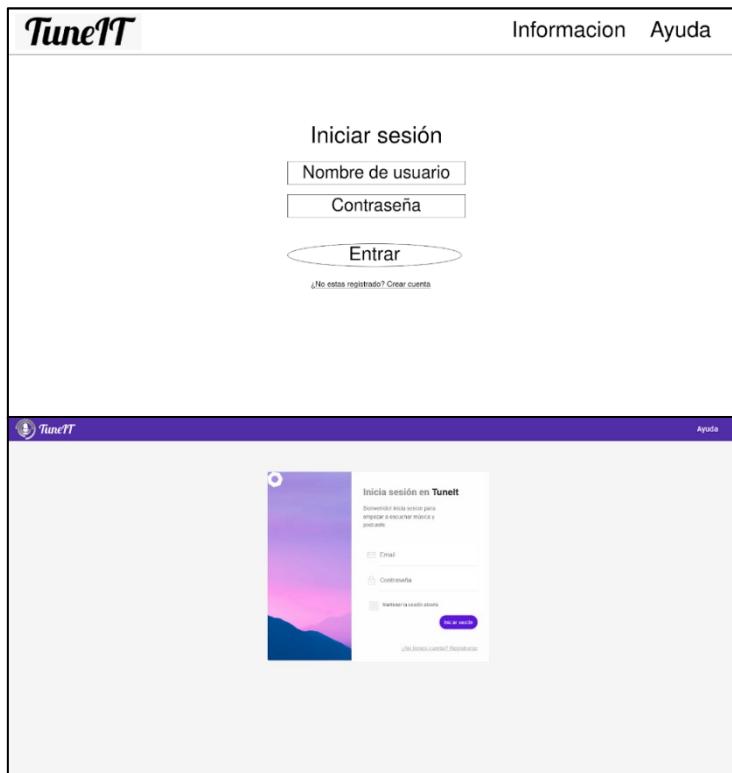


Ilustración 29: Pantalla inicio de sesión Web

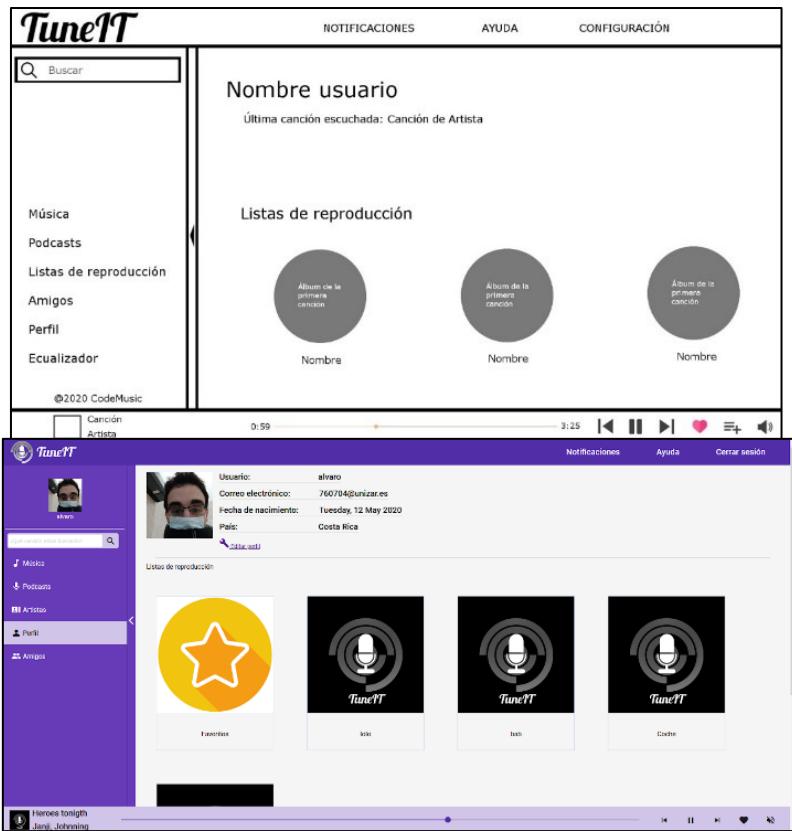


Ilustración 30: Pantalla perfil Web

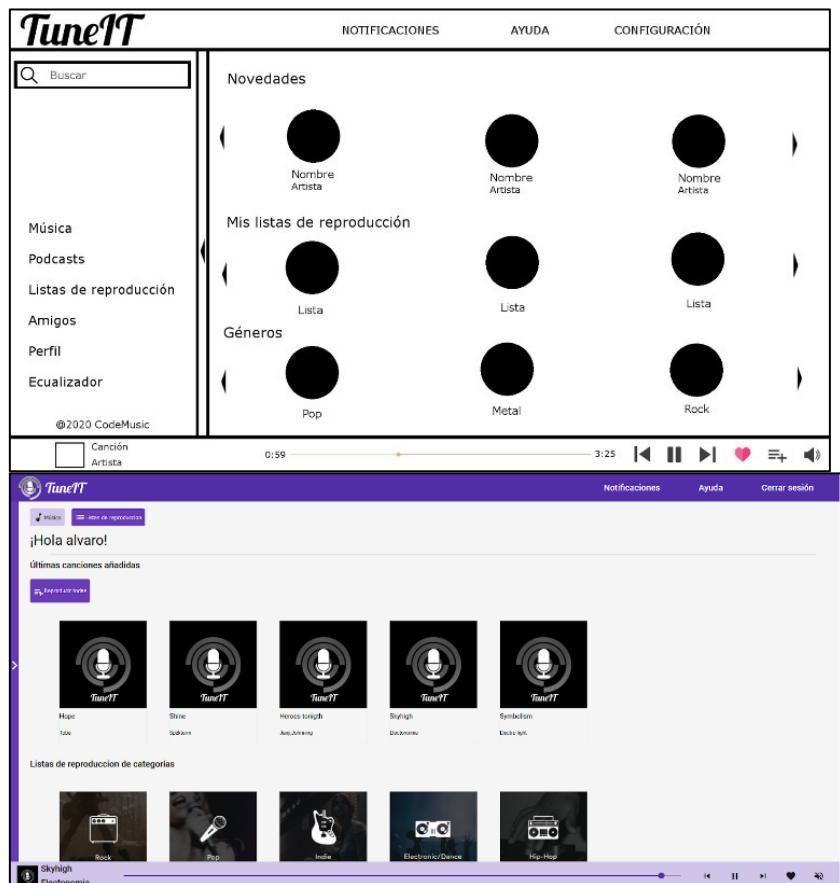


Ilustración 31: Pantalla principal Web

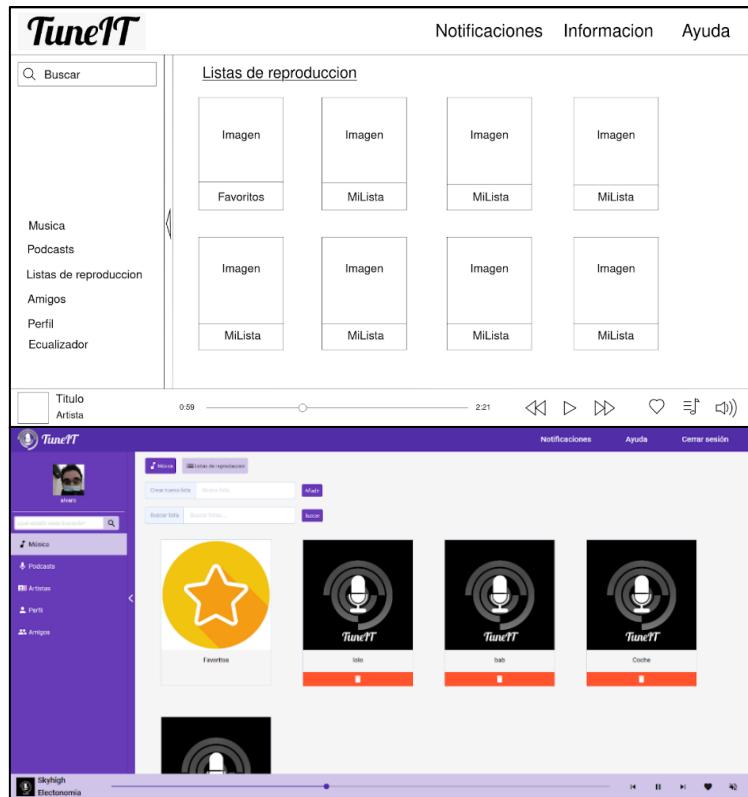


Ilustración 32: Pantalla listas de reproducción Web

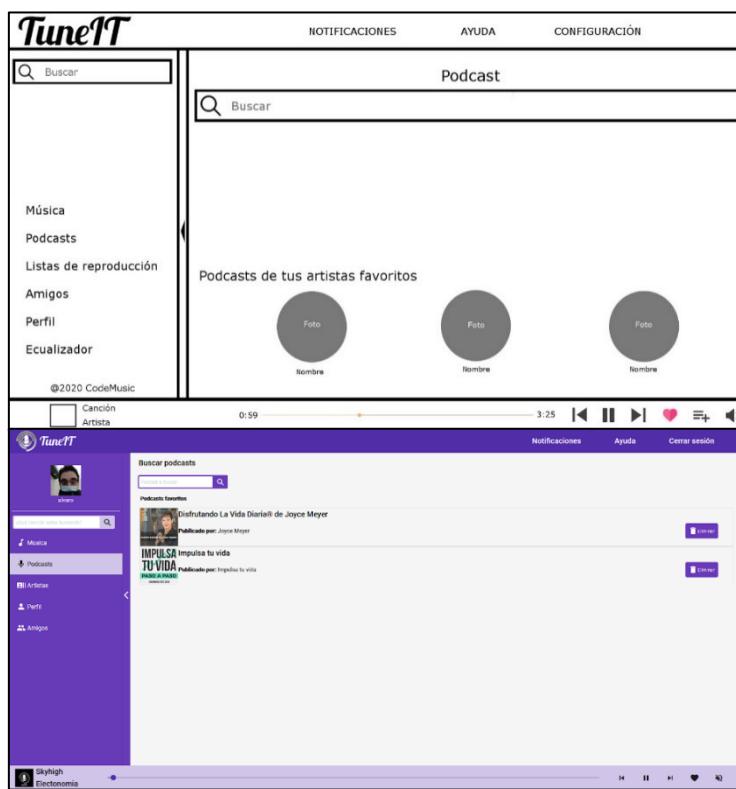


Ilustración 33: Pantalla podcast Web

The screenshot displays two views of the TuneIT website. The top view shows a 'Mi lista' (My List) page with a search bar and a sidebar with links like 'Musica', 'Podcasts', etc. The bottom view shows a 'Favoritos' (Favorites) page with a large yellow star icon, a search bar, and a table listing artists and their tracks.

Nombre	Artistas	Categorías
Heroses tough	Jean, Jötemrg	Hip hop
Skyhigh	Electronomia	Electronic

Ilustración 34: Pantalla lista de canciones/podcast Web

The screenshot shows an artist profile for 'Alan Walker'. The main area displays his name, a 'Suscribirse' (Subscribe) button, a description, and two albums. The sidebar includes a search bar and links for 'Música', 'Podcasts', etc. The bottom view shows a detailed album page for 'Last Alan Walker'.

Ilustración 35: Pantalla artista Web

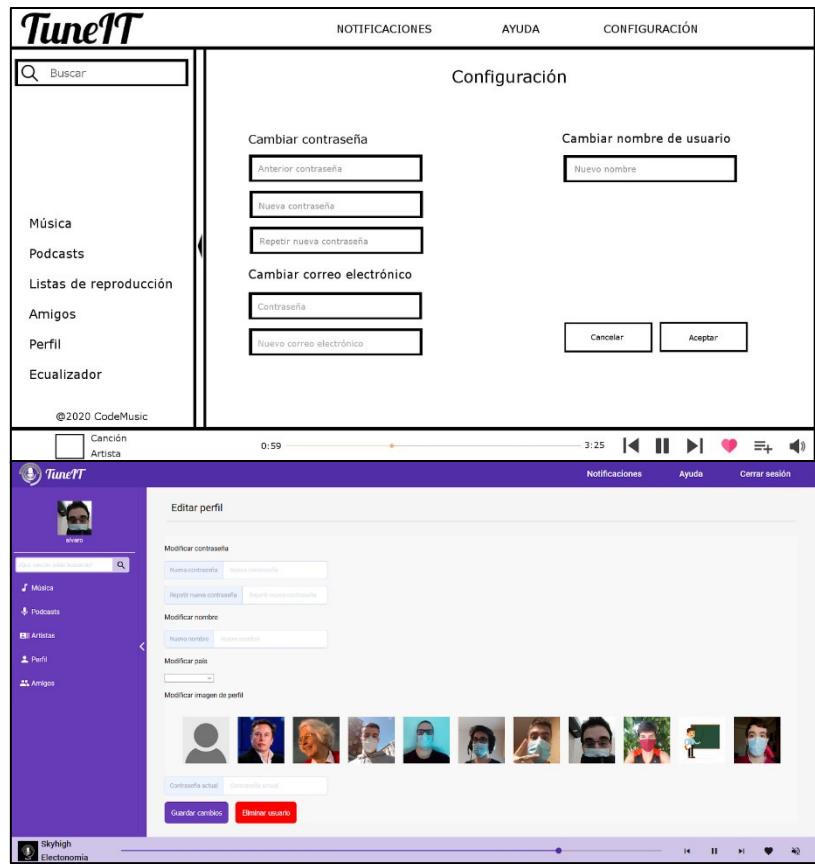


Ilustración 36: Pantalla configuración Web

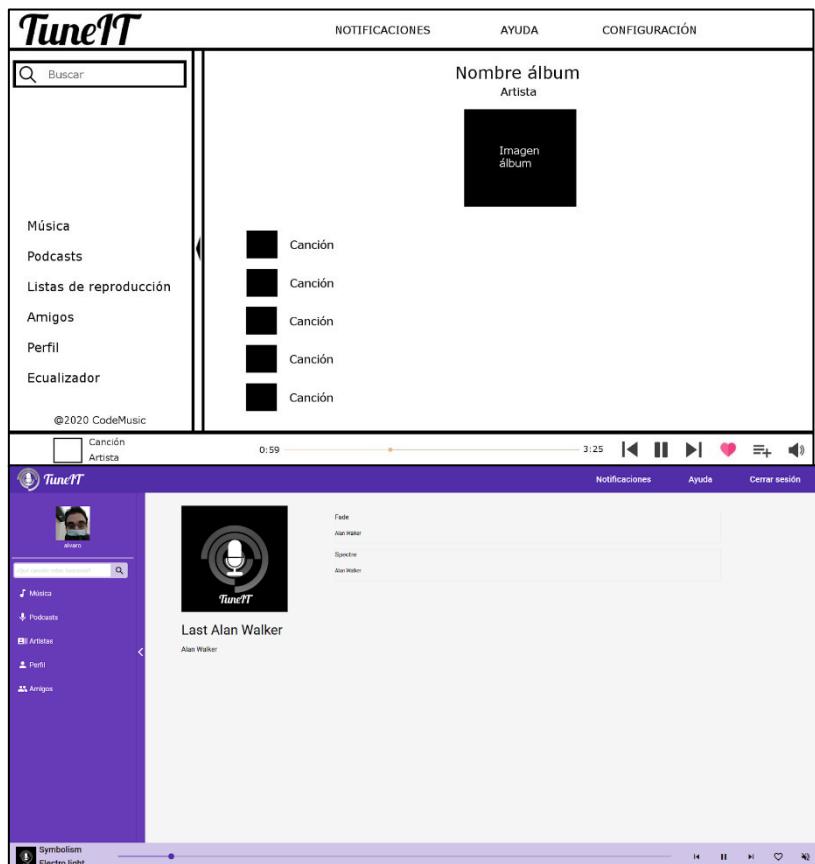


Ilustración 37: Pantalla álbum Web

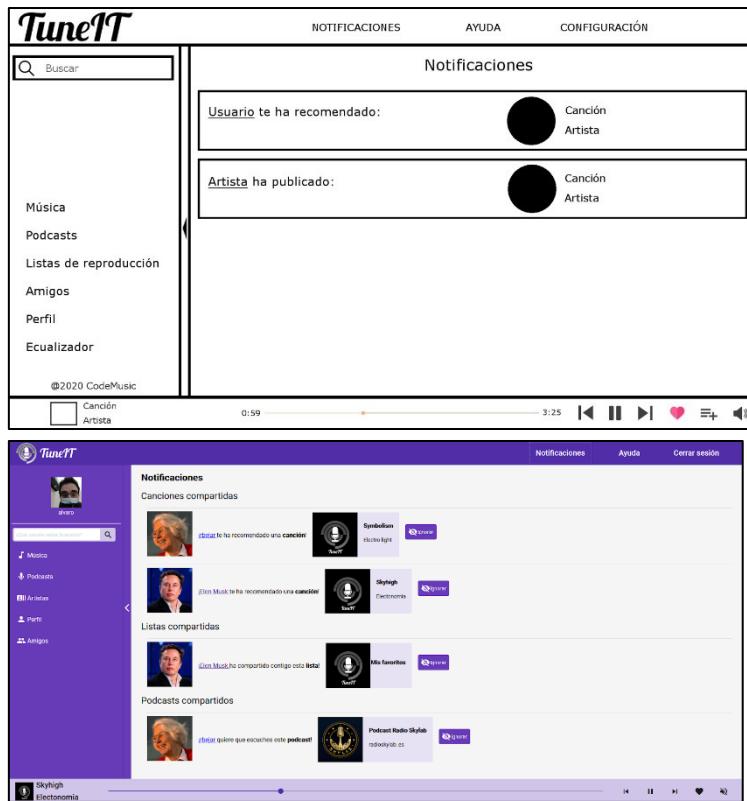


Ilustración 38: Pantalla notificaciones Web

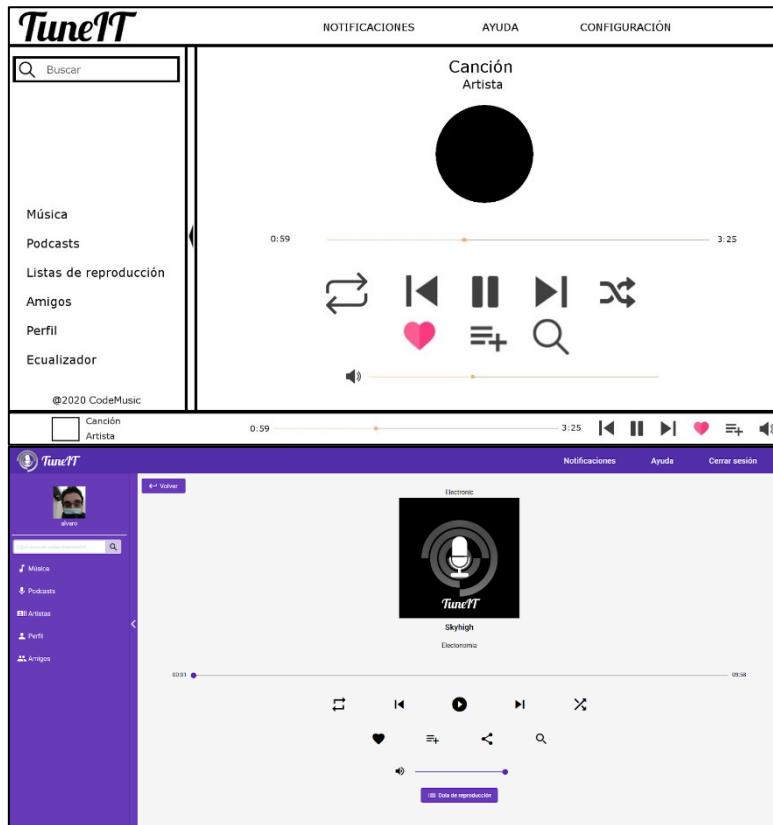


Ilustración 39: Pantalla reproductor Web

#### 4.2.5 Mapa de navegación

Este es el mapa de navegación de toda la aplicación Android, ahora vamos a desglosarlo en unos casos de uso de ejemplo para la aplicación

##### 4.2.5.1 Android

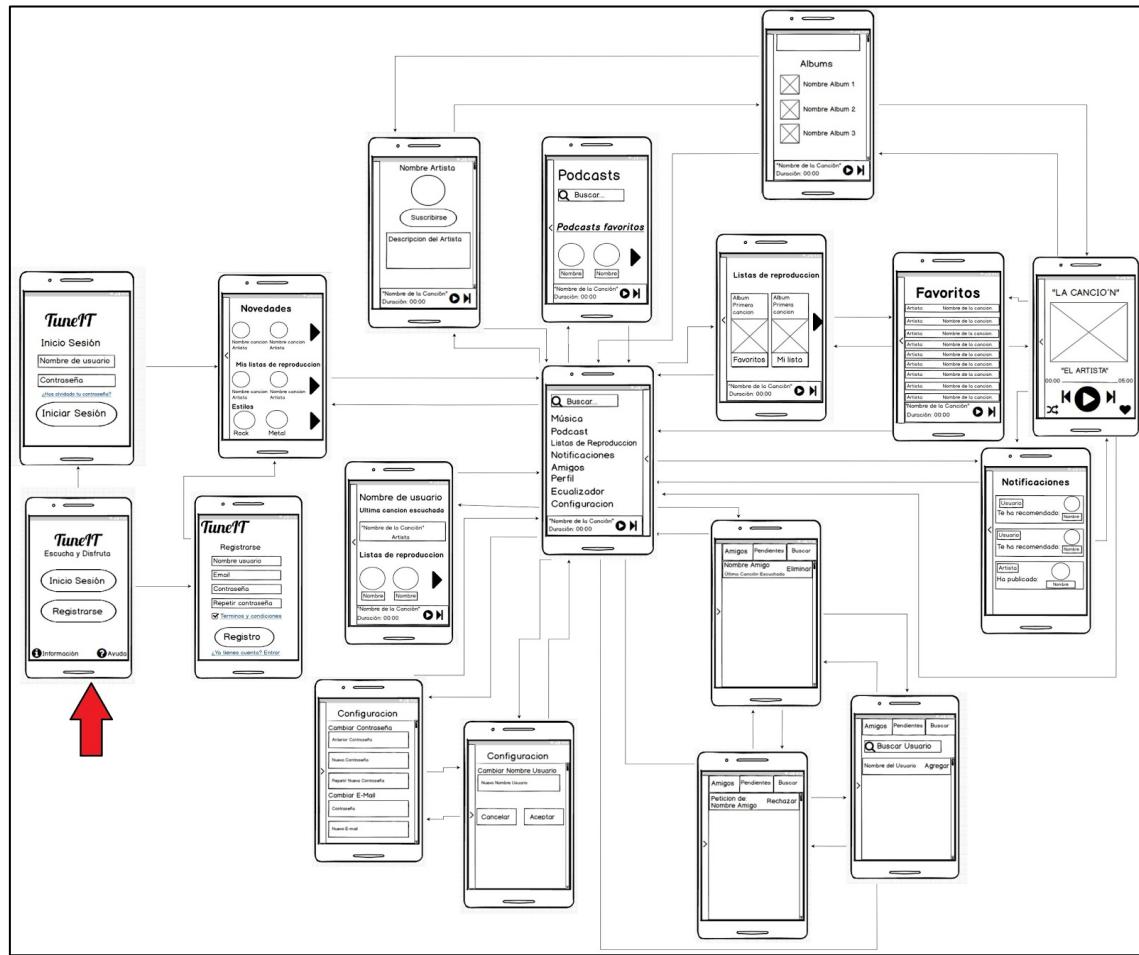


Ilustración 40: Mapa de navegación Android

#### 4.2.5.2 Web

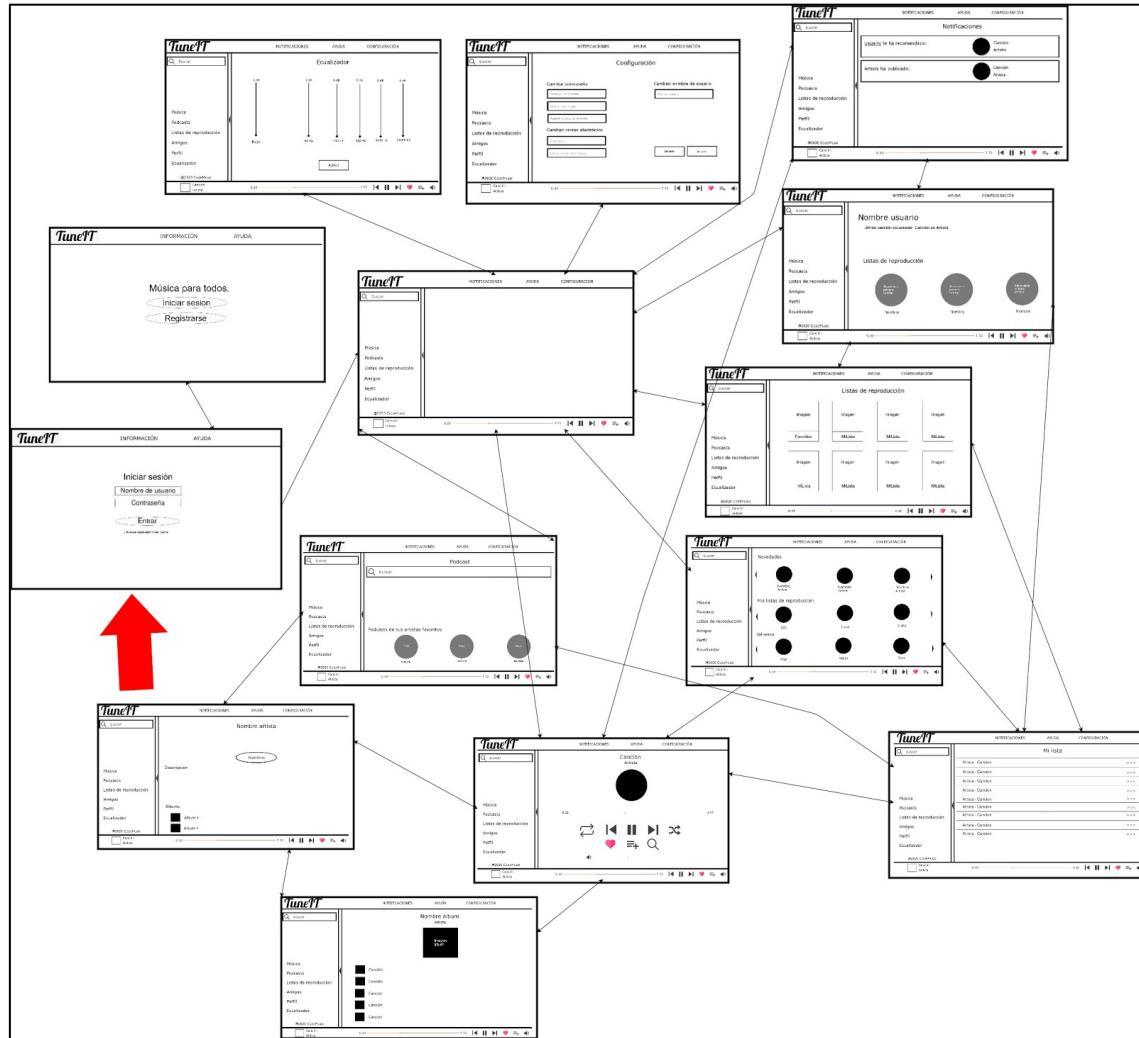


Ilustración 41: Mapa de navegación Web

#### 4.2.6 Casos de uso

- Escuchar una canción con búsqueda:** el usuario tiene que acceder a aplicación y buscar una canción para poder escucharla.
- Reproducir canción de un artista:** el usuario escuchará una canción perteneciente a un artista. Para ello tiene que buscar el artista, acceder a sus álbumes y seleccionar la canción.
- Suscribirse a un podcast:** el usuario va a suscribirse a un podcast. Para ello ha de buscar el podcast, acceder a sus episodios y suscribirse.
- Añadir amigo:** el usuario va a buscar a otro usuario y lo agregar a su lista de amigos.
- Consultar notificaciones:** el usuario va a acceder a sus notificaciones pendientes a través del menú lateral desplegable.

## Escuchar una canción de una lista de reproducción

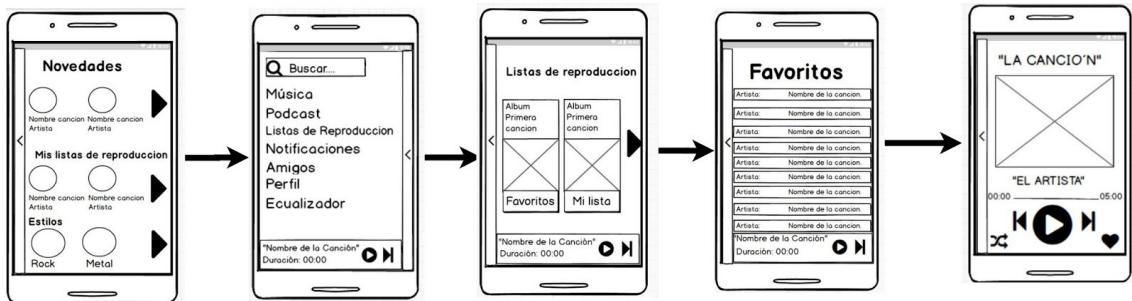
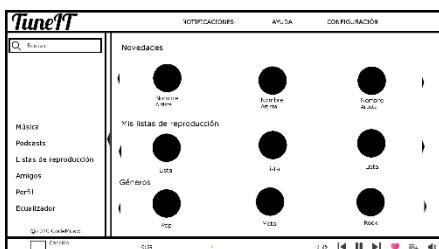


Ilustración 42: Caso de uso "Escuchar una canción de una lista de reproducción" Android



Desde la pantalla principal el usuario desplegaría el menú lateral mostrando sus opciones.

Seleccionaría la opción Listas de Reproducción, una vez hecho esto le aparecerá el menú de listas de reproducción.

En ese menú selecciona una lista de reproducción y pulsa sobre ella.

Se desplegarán las canciones pertenecientes a la lista.

Una vez el usuario haya elegido qué canción quiere escuchar, selecciona esa canción.

Una vez entra en la canción, selecciona el botón central de reproducción.

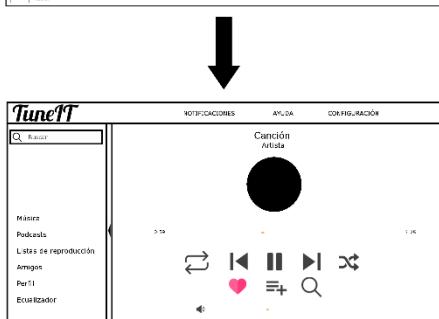
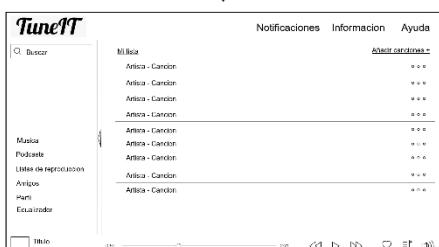
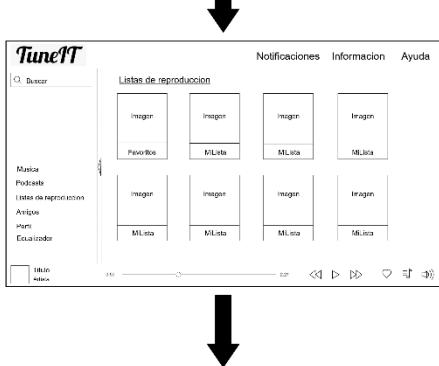


Ilustración 43: Caso de uso "Escuchar una canción de una lista de reproducción" Web

## Reproducir canción de un artista

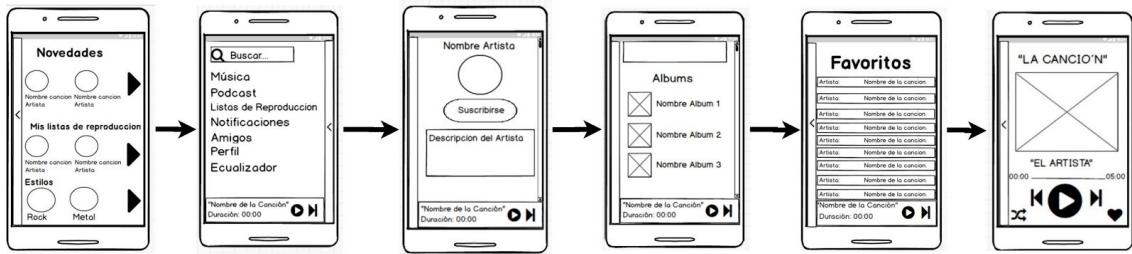
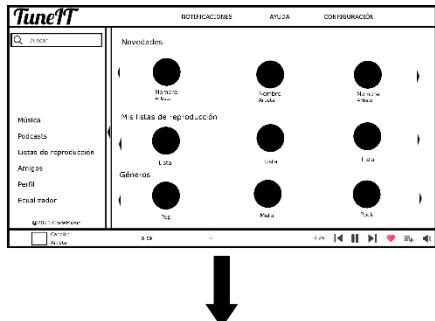


Ilustración 44: Caso de uso "Reproducir canción de un artista" Android



Para empezar, ha de buscar un artista, para ello ha de desplegar el menú lateral y escribir el nombre del artista que quiera encontrar.

Al terminar de escribir el nombre del artista comenzará su búsqueda y se mostrarán sus datos.

El usuario deslizará la pantalla hasta acceder a los álbumes del artista.

Seleccionará un álbum y este desplegará como una lista de reproducción mostrando las canciones que lo compone.

El usuario seleccionará una canción y posteriormente pulsará el botón de central de play.

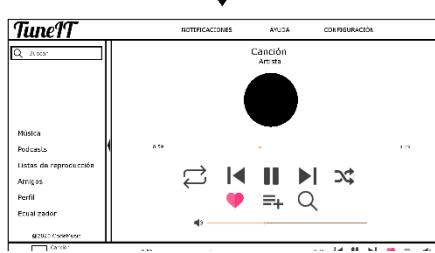
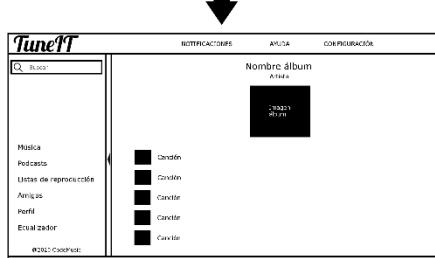
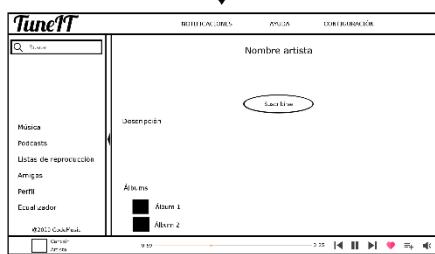


Ilustración 45: Caso de uso "Reproducir canción de un artista" Web

## Suscribirse a un podcast

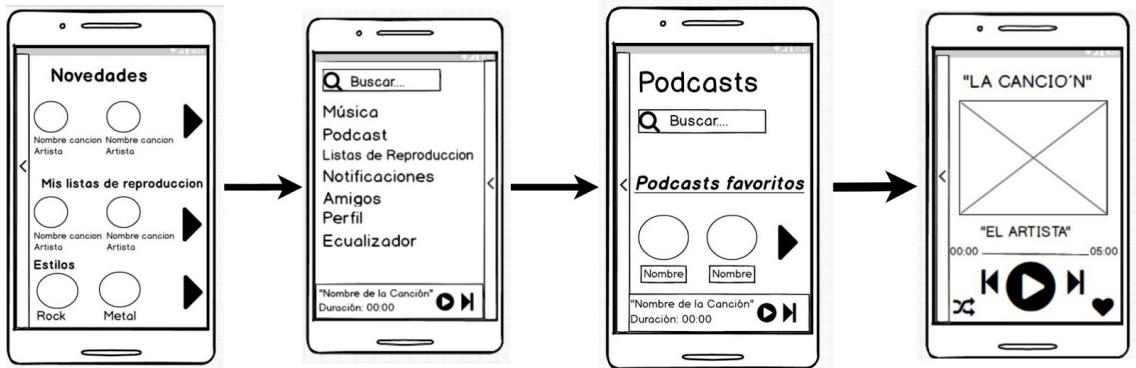
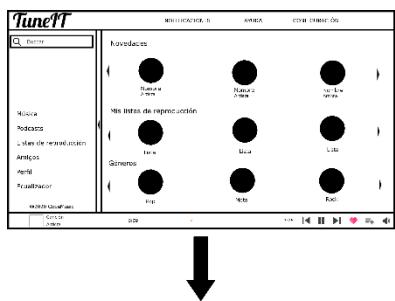


Ilustración 46: Caso de uso "Suscribirse a un podcast" Android



Para suscribirse a un podcast el usuario comenzará buscándolo.

Desplegará el menú lateral y seleccionará la opción podcast.

Aparecerá el menú con los podcasts a los que el usuario se haya suscrito y una opción de búsqueda.

Buscará el podcast que quiera escuchar.

Le aparecerá las opciones que coincidan con su búsqueda.

Entrará en el podcast y marcará el corazón para suscribirse a él.

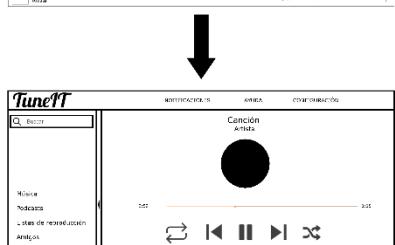
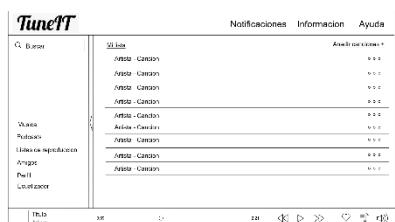
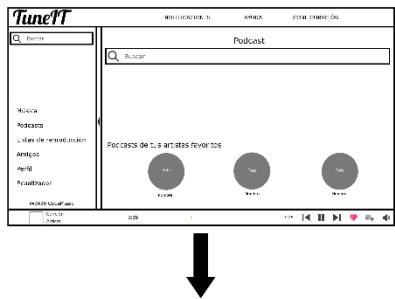


Ilustración 47: Caso de uso "Suscribirse a un podcast" Web

## Añadir amigo

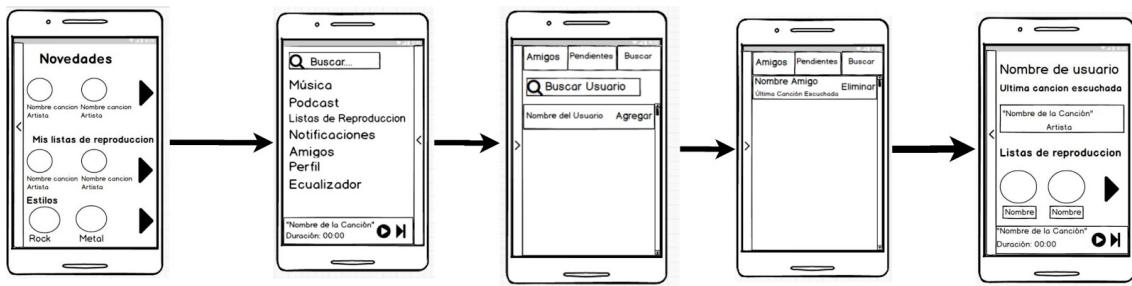
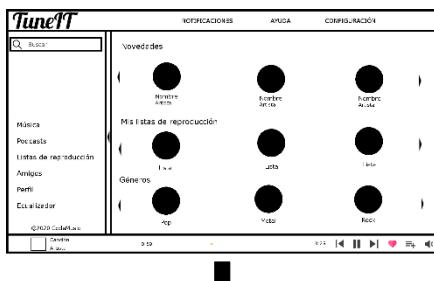


Ilustración 48: Caso de uso "Añadir amigo" Android



El usuario ha de desplegar el menú lateral y acceder a la opción de amigos.

Dentro de las opciones relacionadas con amigos ha de seleccionar la opción Buscar.

Escribirá el nombre del usuario que quiera agregar.

Se iniciará su búsqueda y pulsará sobre la opción de agregar.

Una vez el usuario haya aceptado su petición este aparecerá en la pestaña de amigos como nuevo amigo.

El usuario podrá acceder al perfil de este nuevo usuario y ver su perfil.

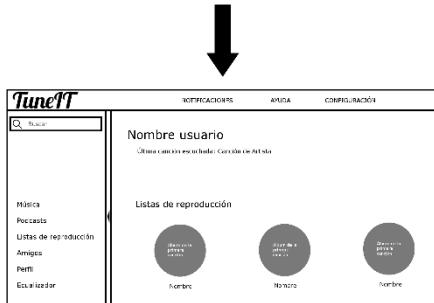
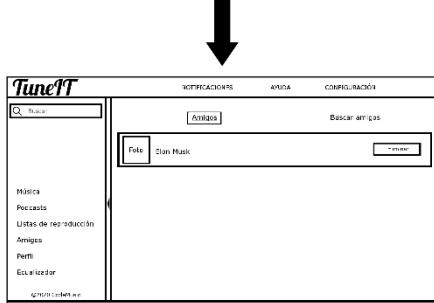
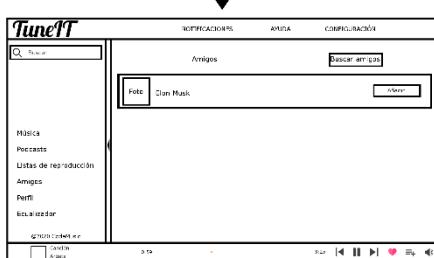


Ilustración 49: Caso de uso "Añadir amigo" Web

## Consultar notificaciones

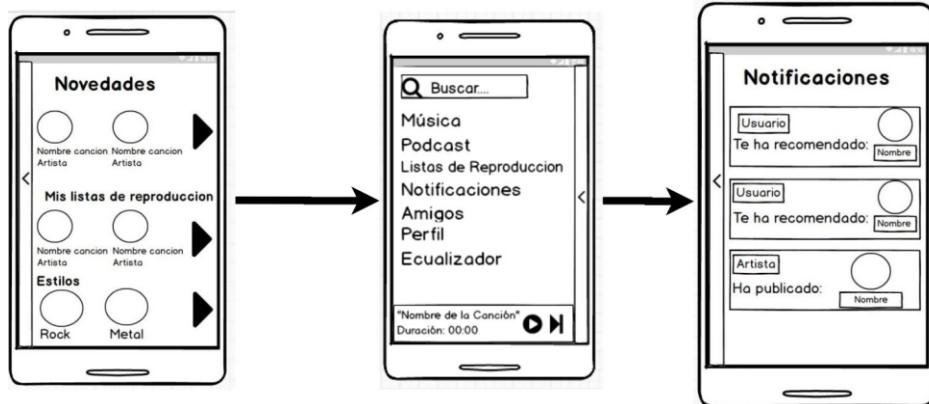
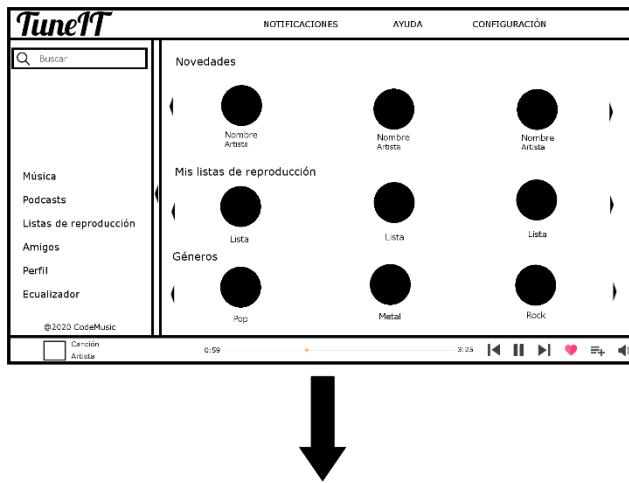


Ilustración 50: Caso de uso "Consultar notificaciones" Android



Con el objetivo de consultar sus notificaciones el usuario tendrá que desplegar el menú lateral.

Una vez desplegado el menú lateral, seleccionará la opción *Notificaciones*.

Al haber seleccionado esta opción se mostrará una nueva pantalla con las nuevas notificaciones pendientes que tenga el usuario.

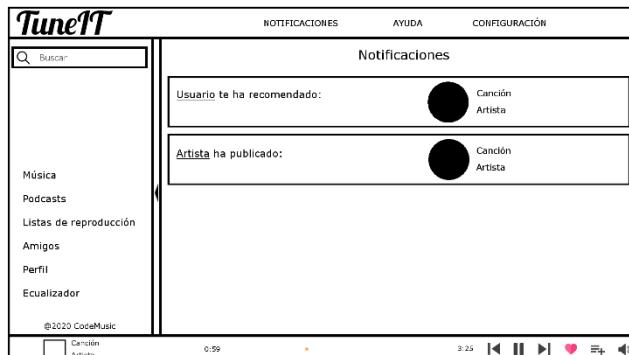


Ilustración 51: Caso de uso "Consultar notificaciones" Web

## 5 Memoria del proyecto

### 5.1 Inicio del proyecto

En el inicio del proyecto se organizaron dos grupos, uno de backend y otro de frontend. Tras la reunión con el profesor se siguió el consejo de dividir frontend en dos: web y android. Rápidamente se eligieron los frameworks con los que se iba a trabajar consultando a otros alumnos y tras la primera reunión con el profesor. De esta forma se inició rápidamente la primera etapa de documentación mediante tutoriales y la documentación oficial de los frameworks elegidos.

La etapa de formación duró aproximadamente dos semanas, aunque se necesitó seguir formándose a lo largo del proyecto ya que casi todas las tecnologías eran nuevas o no se habían utilizado con los objetivos establecidos en este proyecto.

La división de los equipos permitió organizar a los miembros de una mejor forma reduciendo complejidad al trabajar como pequeños equipos. Se permitía que los coordinadores de cada uno se reuniesen entre ellos o con el director general sin tener que estar todos los miembros.

La elección de trabajar en cloud permitió a todos los miembros disponer de los demás componentes del sistema en una época temprana (aun en pruebas) para poder conectarse e interactuar con backend y la base de datos.

### 5.2 Ejecución y control del proyecto

El reparto de las tareas en el grupo de Front End Android empezó con la elección de los miembros del grupo de una tarea de un conjunto de tareas asignada por el propio coordinador, una vez realizada la tarea y revisada por dicho coordinador, este asignó a cada uno de los miembros del grupo una serie de tareas relacionadas con la anteriormente realizada. Mientras que en el reparto de tareas del grupo Front End Web se ha realizado basándose en la dificultad, las tareas que tienen que ver con algo realizado por alguno de los miembros del grupo son asignadas a estos, ya que están experimentados en la materia o están familiarizados con dicha tarea. Por último, en el grupo de Back End, uno de los miembros se ha centrado más en el funcionamiento del servidor mientras que el otro miembro del grupo se ha centrado más en el despliegue del mismo y la base de datos.

La comunicación interna entre los distintos miembros de los diferentes equipos se ha realizado según el apartado de Procesos de ejecución y control del proyecto 3.1.2, en el cual, explica las diferentes vías de comunicación que han utilizado dichos miembros de los grupos.

El progreso se ha medido conforme al cumplimiento de los requisitos tanto en los grupos de Front-End como Back-End, los requisitos se deciden si se cumplen o no durante las reuniones entre todos los miembros del grupo, además de que cada requisito tiene asociado una puntuación entre 1 y 3, que fue decidida por todos los miembros del grupo durante una votación. Los trabajos se saben si han sido realizado mediante los “Issues” en la plataforma Github, explicado anteriormente en el apartado de Procesos de ejecución y control del proyecto 3.1.2

Se han realizado modificaciones frente al calendario inicial si más de la mitad de los miembros del proyecto no han realizado su parte correspondiente, dando como máximo 1 día más.

Durante el desarrollo de integración y de despliegue, los miembros del grupo de Back-End no sabían cómo realizar dichas acciones, esto se ha resuelto mediante la adquisición de conocimiento sobre Heroku y Docker, además los mínimos conocimientos de las tecnologías utilizadas en backend y frontend.

Durante las pruebas del software, se han cumplido las ideas que se tenían al respecto mediante el cumplimiento de los requisitos de la aplicación

Como se ha nombrado en el punto 3.1.2 se ha tomado medidas de progreso a lo largo de la mayor parte del proyecto (desde 05/04).

A continuación, se muestran los distintos gráficos para el control del proyecto:

Primero las tablas de requisitos tanto en Android como en Web y finalmente en Global (se calcula automáticamente respecto a los anteriores). Se colorea según el grado de completitud de cada requisito como se indica en el apartado 3.1.2.

Web	05/04	12/04	19/04	26/04	03/05	10/05	17/05	24/05	31/05	07/06	14/06
RF1	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF2	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF3	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF4	0,5	0,5	0,5	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0
RF5	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF6	0,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF7	0,5	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF8	0,0	0,0	1,0	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF9	0,5	0,5	0,5	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF10	0,0	0,5	0,5	0,5	0,5	0,5	1,0	1,0	1,0	0,0	0,0
RF11	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF12	0,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF13	0,0	0,0	0,0	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF14	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF15	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF16	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF17	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF18	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF19	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF20	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF21	0,0	0,0	0,0	0,0	0,0	0,0	0,7	1,0	1,0	0,0	0,0
RF22	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RF23	0,0	0,0	0,0	0,5	0,5	0,5	0,7	1,0	1,0	0,0	0,0
RF24	0,0	0,0	0,0	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF25	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF26	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF27	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
RF28	0,0	0,0	0,0	0,0	0,0	0,0	0,5	1,0	1,0	0,0	0,0
RF29	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF30	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
RF31	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF32	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF33	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF34	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF1	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF2	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF3	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF4	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
RNF5	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF6	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF7	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF8	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF9	0,0	0,5	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF10	0,0	0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF11	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF12	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0

Ilustración 52: Control requisitos Web

Android	05/04	12/04	19/04	26/04	03/05	10/05	17/05	24/05	31/05	07/06	14/06
RF1	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF2	0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF3	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RF4	0,5	0,5	0,5	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0
RF5	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF6	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF7	0,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF8	0,0	0,0	0,5	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF9	0,5	0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF10	0,0	0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF11	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF12	0,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF13	0,0	0,0	0,0	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF14	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF15	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF16	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF17	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF18	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF19	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF20	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RF21	0,0	0,0	0,0	0,0	0,0	0,0	0,7	1,0	1,0	0,0	0,0
RF22	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RF23	0,0	0,0	0,0	0,0	0,0	1,0	0,7	1,0	1,0	0,0	0,0
RF24	0,0	0,0	0,0	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0
RF25	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RF26	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RF27	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
RF28	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,25	1,0	1,0	0,0
RF29	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0
RF30	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0
RF31	0,0	0,0	0,0	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0
RF32	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF33	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RF34	0,0	0,0	0,0	0,0	0,0	0,5	1,0	1,0	1,0	0,0	0,0
RNF1	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF2	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF3	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0
RNF4	0,0	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	0,0	0,0
RNF5	0,0	0,0	0,0	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0
RNF6	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF7	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF8	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF9	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF10	0,0	0,5	0,5	1,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF11	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0
RNF12	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0

Ilustración 53: Control requisitos Android

Requisito	05/04	12/04	19/04	26/04	03/05	10/05	17/05	24/05	31/05	07/06	14/06	Total	Puntuación
RF1	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF2	2,3	2,3	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF3	0,0	0,0	0,0	0,0	0,0	1,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF4	0,5	0,5	0,5	0,5	0,5	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF5	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF6	0,0	1,5	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF7	0,8	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF8	0,0	0,0	1,5	1,5	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF9	1,0	1,0	1,0	1,5	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF10	0,0	1,0	1,0	1,5	1,5	1,5	2,0	2,0	2,0	0,0	0,0	2,0	2
RF11	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF12	0,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF13	0,0	0,0	0,0	0,5	1,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF14	0,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF15	0,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF16	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF17	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF18	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF19	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF20	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF21	0,0	0,0	0,0	0,0	0,0	0,0	1,4	2,0	2,0	0,0	0,0	2,0	2
RF22	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF23	0,0	0,0	0,0	0,3	0,3	0,3	0,6	0,7	1,0	1,0	0,0	1,0	1
RF24	0,0	0,0	0,0	0,5	0,8	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RF25	0,0	0,0	0,0	0,0	0,0	1,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF26	0,0	0,0	0,0	0,0	0,0	1,5	3,0	3,0	3,0	0,0	0,0	3,0	3
RF27	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	2,0	0,0	0,0	2,0	2
RF28	0,0	0,0	0,0	0,0	0,0	0,0	1,1	3,0	3,0	0,0	0,0	3,0	3
RF29	0,0	0,0	0,0	0,0	0,0	1,0	1,0	2,0	2,0	0,0	0,0	2,0	2
RF30	0,0	0,0	0,0	0,0	0,0	0,0	0,0	2,0	2,0	0,0	0,0	2,0	2
RF31	1,0	1,0	1,0	1,5	1,5	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF32	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RF33	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RF34	0,0	0,0	0,0	0,0	0,0	1,5	2,0	2,0	2,0	0,0	0,0	2,0	2
RNF1	0,0	0,0	0,0	0,0	0,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RNF2	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RNF3	0,0	0,0	0,0	0,0	0,0	0,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RNF4	0,0	0,0	0,0	0,0	0,0	0,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RNF5	0,0	0,0	0,0	0,8	0,8	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RNF6	0,0	0,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RNF7	0,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RNF8	0,0	0,0	0,0	0,0	0,5	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
RNF9	0,0	0,5	0,0	0,0	1,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RNF10	0,0	1,5	1,5	3,0	3,0	3,0	3,0	3,0	3,0	0,0	0,0	3,0	3
RNF11	0,0	0,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2
RNF12	0,0	0,0	0,0	0,0	1,0	1,0	1,0	1,0	1,0	0,0	0,0	1,0	1
Total	16,5	28,3	31,5	42,5	49,8	65,3	81,2	89,0	89,0	0,0	0,0	89	

Ilustración 54: Control requisitos Global

Tabla semanal de porcentaje de cumplimiento de issues. Se busca no tener un bajo porcentaje de issues cerrados frente a el total de issues. Se puede observar cómo conforme se avanza hay más issues cerrados que abiertos aumentando el porcentaje.

	05/04	12/04	19/04	26/04	03/05	10/05	17/05	24/05	31/05	
Web										
Issue	Abiertos	Cerrados								
Facil	4	4	4	4	5	4	6	2	1	10
Medio	4	3	3	4	5	5	9	5	12	14
Difícil	3	2	3	2	4	2	5	3	6	16
Proporción	43,24%	48,65%	60,00%	66,10%	72,31%	77,78%	74,68%	75,61%	100,00%	
Android										
Issue	Abiertos	Cerrados								
Facil	5	7	3	9	3	10	4	11	3	15
Medio	7	3	7	6	5	4	10	4	11	15
Difícil	7	2	6	6	6	6	4	4	12	13
Proporción	32,29%	48,68%	45,90%	59,5%	72,43%	76,79%	81,92%	86,52%	100,00%	
Backend										
Issue	Abiertos	Cerrados								
Facil	5	1	1	4	4	6	6	10	3	13
Medio	5	3	5	4	3	6	5	4	7	11
Difícil	3	2	3	4	4	3	5	3	7	9
Proporción	27,27%	43,49%	44,44%	62,50%	64,07%	68,85%	77,74%	90,91%	100,00%	
Proporción total	34,24%	44,27%	50,32%	62,59%	69,94%	75,14%	78,33%	87,68%	100,00%	

Ilustración 55: Control issues

Gráficas para ver la evolución en el tiempo y hacerse mejor una idea del avance del proyecto.

Fecha	Líneas de código	Requisitos	Issues
05/04	5663	20,79%	34,24%
12/04	6119	31,74%	44,27%
19/04	8679	35,39%	50,12%
26/04	10397	47,75%	62,59%
03/05	12467	55,90%	69,94%
10/05	15685	73,31%	75,14%
17/05	17747	91,26%	78,11%
24/05	19933	100,00%	87,68%
31/05	20312	100,00%	100,00%

Ilustración 56: Métricas totales

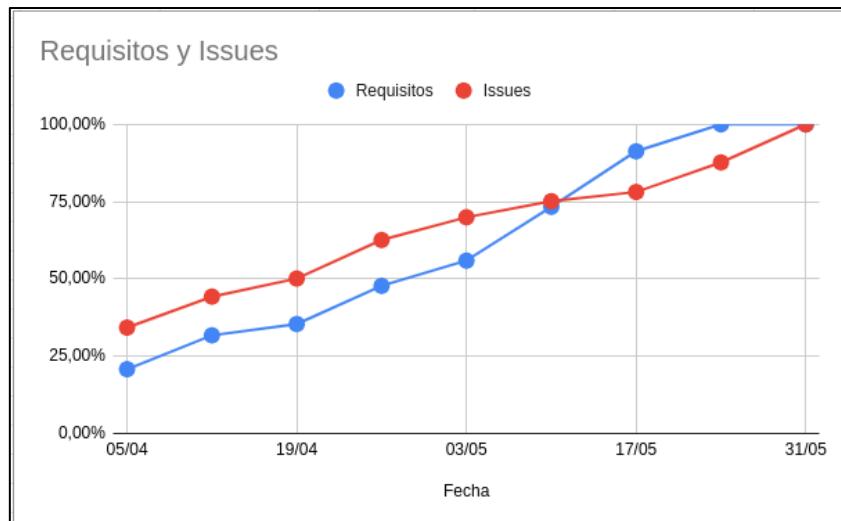


Ilustración 57: Gráfica de requisitos e issues

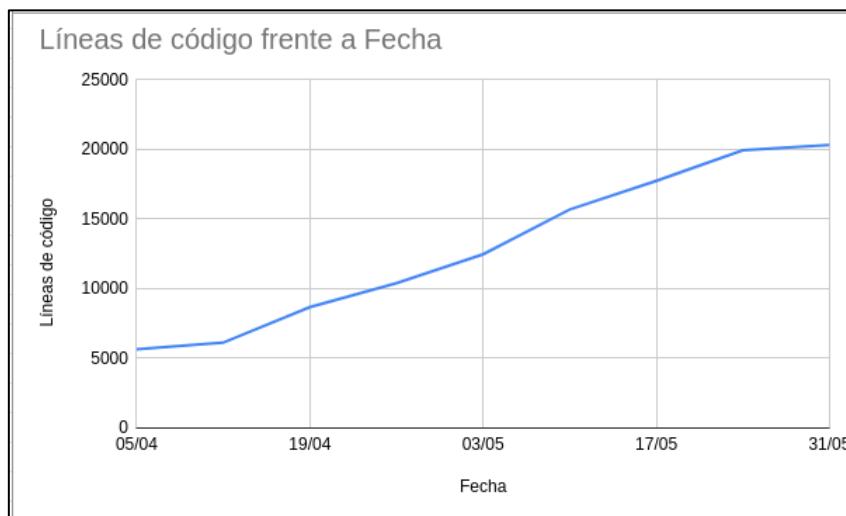


Ilustración 58: Gráfica de líneas de código

### 5.3 Cierre del proyecto

Para cerrar el proyecto, se van a llevar a cabo una serie de análisis sobre aspectos importantes, como son: una comparación entre las estimaciones iniciales planteadas, un análisis que se llevó a cabo a mitad de proyecto y los resultados obtenidos finalmente; un resumen de las lecciones aprendidas sobre las herramientas y tecnologías empleadas; y una recopilación del trabajo realizado por cada uno de los miembros

#### Comparativa de estimaciones iniciales, análisis intermedio y resultados finales:

En la *propuesta técnica y económica* del proyecto se plantearon unas estimaciones sobre las horas de esfuerzo enfocadas a realizar las tareas establecidas y el coste económico que éstas implicarían. Tal y como aparece se presupuestaron **760 horas** con un costo total de 14.060€ (18.5€ por hora), a lo que hay que sumar un incremento de 2810€ por gestión y 1.040€ por costes adicionales convirtiéndolos en **17.910€ sin IVA**. Se calculó un segundo presupuesto que venía acompañado de funcionalidades adicionales que se estimó de 2330€ más, esto es **20.240€ sin IVA**.

A día 10 de abril, se habían invertido un total de 271 horas en desarrollo y documentación, lo que significaba aproximadamente 200 horas mensuales. Además, hay que tener en cuenta las reuniones que suman alrededor de 20 horas por miembro, por lo que se podría decir que hasta esa fecha el total de horas invertidas es de **411**.

A ese ritmo se pensó que se llegaría a la entrega final del 8 de junio con un total de **800 horas**, es decir, 14800€ contabilizando únicamente las horas de trabajo. Los resultados finales que se esperaban eran superiores a los estimados inicialmente, ya que surgieron numerosos imprevistos que requirieron de muchas horas para solucionarlos.

También es cierto que las estimaciones iniciales se calcularon sin dejar un margen de tiempo extra (un error que no se puede cometer) planteando los tiempos de manera optimista. Como consecuencia, el presupuesto final puede ser algo mayor que el estimado para el cliente.

Ya terminado el proyecto, se puede decir con certeza que el total de horas invertidas por el equipo es de 690 horas sin contabilizar las reuniones. Si se contabilizan salen un total de **977 horas** (41 horas por miembro). Tal y como se explicaba antes, cada hora se cotizaba a 18.5€, por lo que esas horas se capitalizan en 18.074,5€ sin IVA, y que, además incrementando el precio de gestión y los costes adicionales nombrados en la estimación inicial, se obtiene un valor de **21.924,5 € sin IVA**.

La cifra final es superior a la estimada en los primeros análisis (tanto al inicio como a mediados de proyecto), así que las estimaciones no fueron muy acertadas.

#### **Lecciones aprendidas sobre herramientas y tecnologías:**

Respecto a **SQLAlchemy**, se ha visto que proporciona un ORM (Object Relational Mapper) el cual permite crear y manipular una base de datos como si fuera POO (Programación orientada a objetos) lo cual es una ventaja dada nuestra experiencia con bases de datos relacionales comunes y POO.

En el servidor **Heroku**, la curva de aprendizaje ha sido lenta y costosa debido a que no se sabía cómo se iba a desplegar el proyecto, ya sea con contenedores, con manifiestos que indiquen cómo descargar heroku, o haciendo push desde los repositorios de git... Otro factor que contribuyó a este lento aprendizaje fue que en la documentación no aparecen casos de error y sus posibles causas o soluciones por lo que se tenía que investigar cada pequeño fallo. A partir de esos 2 factores principales, la curva crece y se hace sencillo el manejo de Heroku y sus utilidades.

En el lado de front end, se ha visto que el proyecto en la parte web, en **Angular**, se comenzó con los tutoriales y se aprendió los básicos de dicha tecnología de manera muy rápida. El problema comenzó cuando se tuvo que empezar a aplicar dichos conocimientos al proyecto. La curva de aprendizaje dejó de crecer y cada pequeña funcionalidad, costaba mucho más tiempo del ideal debido a la diferencia entre la realidad y los tutoriales. El único momento donde se vió un aumento de la efectividad fue al realizar tareas semejantes ya que su "modus operandi" era muy parecido, por ejemplo, a la hora de crear un buscador de podcasts, se tardó 17 horas, muchas más de las esperadas, pero al crear un buscador de canciones apenas se tardaron 3 horas debido a que el proceso era muy similar.

Por otra parte, desde el lado de móvil, se ha visto que la tecnología **Flutter** contiene un problema principal y es que está en constante desarrollo por lo que muchas cosas cambian y los tutoriales dejan de ser útiles, esto hace que la curva de aprendizaje sea muy pronunciada. También se ha

visto que es muy útil para diseñar e implementar interfaces, pero para implementar herramientas en concreto apenas tiene documentación y el equipo de desarrollo tiene que descender a Java para poder crear dicha herramienta, por ejemplo el ecualizador.

Por último, en el uso de **Flask** se ha visto que es una herramienta muy sencilla, con una curva de aprendizaje muy pronunciada y muy bien documentada.

**Recopilación de horas de trabajo y tareas de cada uno de los miembros del equipo:**

**Atencion: las horas de reuniones están por separado**

Miembro	Horas	Tareas
Alberto	92,32	Dirección de proyecto, coordinador de Documentación, encargado de integración y despliegue (recursos cloud) y administración de base de datos
Alejandro	75,67	Desarrollador Android
Álvaro	114,75	Coordinador de frontend web y desarrollador web
Germán	97,25	Desarrollador web y responsable actas
Luis	117,00	Coordinador de frontend Andrid y desarrollador Android
Óscar	88,08	Desarrollador Android
Saúl	104,25	Coordinador backend, desarrollador web y API

**Total horas 689,32**

## 6 Conclusiones

**Óscar:**

Como conclusión final, hay que destacar las dificultades surgidas del trabajo en equipo. La coordinación, aunque pueda parecer poco importante desde fuera, es esencial en este tipo de proyectos. Dividir al equipo en grupos (con cada uno una serie de tareas de las que responsabilizarse) y elegir bien a los líderes/coordinadores de cada uno, así como al general, va a facilitar la resolución de problemas que vayan apareciendo durante el desarrollo.

Durante el proyecto, se han cometido gran cantidad de errores. El más destacado, desde mi punto de vista, se puede describir con la siguiente frase: "Esto lo dejamos para más adelante". Se ha repetido en varias de las tareas de implementación, creyendo que realmente se cumpliría, pero llegados los días finales, se ha visto que era imposible. Sin duda, es algo que hay que evitar. Para ello, es necesario que antes de terminar una tarea se busque conseguir una versión que (si se requiriera) pudiera ser entregada en ese mismo momento.

### Alejandro:

Este proyecto no ha sido una práctica, ha sido más bien un acercamiento a experiencia real de trabajo, ya que la carga de trabajo y la mentalidad son totalmente distintas, de por sí, el grupo de personas es mayor al de las demás prácticas de las asignaturas y que este proyecto se toma mucho más en serio, ya sea por los aportes indicados por el profesor conforme a la memoria o como nos reuníamos entre nosotros e indicábamos que fallos teníamos o que debíamos modificar para que se cumplieran totalmente los requisitos.

Al ser nuestro primer proyecto, hemos tenido fallos, algo que es normal, tanto a la hora de organizarnos entre los diferentes grupos como a la hora de comunicarnos entre nosotros. Desde el primer momento se tendría que haber organizado mejor el tema de los issues a realizar y ver si eran posibles o no de hacer, también se tendría que haber cambiado cuando se hacían las reuniones.

### Alberto:

Este se trata del primer proyecto de software con un cierto tamaño a tomar en serio. En él he desempeñado dos papeles y me he dado cuenta de cosas que nunca me había planteado nunca.

Como Director de Proyecto, se dedican bastantes horas simplemente a coordinación, toma de decisiones, reuniones, revisión de memorias y se nota un extra de responsabilidad al realizar la asignatura. La carga de trabajo fue bastante importante al principio, ya que organizar un grupo de 7 personas en los que ninguno ha hecho algo parecido supone tomar decisiones que si no se estudian bien pueden perjudicar en el proyecto.

Como encargado de Base de Datos, despliegue y recursos cloud también se tiene bastante carga de trabajo al principio del proyecto. Es necesario preparar el modelo de datos y los recursos para poner los servidores para que todo el mundo pueda trabajar e integrar sus partes.

Possiblemente, si tuviera que repetir el proyecto sin ningún conocimiento de nuevo no cogería ambos roles a la vez. El inicio del proyecto resulta más cargado y el final menos. En cambio, una vez teniendo esta experiencia sí que repetiría, ya que, si se trabaja bien al inicio de proyecto, el resto de las tareas suele ser más livianas, ya que se trata de hacer pequeños cambios, añadir entidades, controlar recursos cloud y controlar el proyecto.

### Luis:

Este trabajo ha puesto un reto tanto en lo personal como en lo técnico para mí. En lo personal yo nunca había coordinado a un grupo de compañeros para sacar adelante un proyecto. Esto ha implicado aprender a organizarme y a distribuir el trabajo entre mis compañeros.

Nunca había trabajado en un proyecto que se acerque a un trabajo real. En lo técnico también ha supuesto un reto ya nunca había programado para Android y he tenido que aprender de cero. Respecto a la carga de trabajo ha sido casi todas las semanas constante y ha estado presente en mi día a día. Ha sido complicado en cierta medida, ya que, al hacer cosas que nunca había hecho y al aprender de cero en muchas ocasiones, no sabías cuánto tiempo iba a costar hacer un apartado o cuándo ibas a tener terminado un requisito. Esto último ha generado un poco de ansiedad y varias noches sin dormir tranquilo. Hay objetivos que no hemos logrado cumplir, como el subir imágenes a la aplicación para que los usuarios puedan cambiar su foto de perfil. Aunque en general estoy satisfecho con el proyecto.

### Álvaro:

Con la realización de este proyecto y al ser coordinador en uno de los frontend, me he dado cuenta de la importancia que posee la persona que coordina y de sus funciones, como la asignación de tareas o asegurarse de la correcta finalización de estas. Personalmente, no he sido coordinador en ninguna práctica y ha sido un reto conocer las aptitudes de mi grupo para asignarles las tareas en las que iban a tardar menos tiempo.

En cuanto al trabajo realizado para completar la web, ambos miembros del grupo tenían nula o poca experiencia en el uso de TypeScript y ya habían trabajado anteriormente con HTML y CSS, pero no en un proyecto de esta magnitud, por eso considero acertada el seguimiento de unos tutoriales completados al principio para tener una primera toma de contacto con el framework Angular. Cada semana, se repartía una serie de tareas cada miembro que tendría que completar antes del final de la semana. Si alguna vez no se terminaba alguna, se incluía para la siguiente semana. En algunos casos, la tarea ha resultado más compleja de lo esperado y surgieron problemas con políticas (CORS) que derivaron en la eliminación de ciertos requisitos o tareas como el ecualizador o la subida de imágenes a Amazon para permitir al usuario tener la imagen de perfil que deseé.

Una parte negativa fue la decisión de dejar el diseño y la escalabilidad para el final, ya que, si se hubiera realizado en el mismo momento que la funcionalidad o pantalla, habría costado menos tiempo, por lo que en siguientes proyectos se realizaría al mismo tiempo que la funcionalidad/pantalla.

En general, me ha gustado la experiencia del proyecto ya que se trata de la primera cercanía a un proyecto en la vida real y nos prepara para lo que vamos a realizar en un futuro.

### Germán:

Este proyecto sabía que iba a requerir de muchas horas y esfuerzo por parte de todo el equipo y he de confesar que estoy gratamente sorprendido, a pesar de haber tenido muchos problemas debido a nuestra inexperiencia en proyectos tan grandes y con tecnologías que hemos manejado tan poco, siempre hemos conseguido sacar una solución adecuada.

El tema de dividirnos en grupos (móvil, web, backend) me ha parecido acertadísimo, sin embargo, no nos pusimos roles dentro de cada grupo (al menos en web, Alvaro y yo) por lo que al principio hacíamos los 2 todas las cosas, con el tiempo nos dimos cuenta de que yo era mucho más eficiente con todo el tema de visualización de datos (html y css) y Álvaro era mejor en trabajar con el backend y TypeScript. Si nos hubiéramos dado cuenta de esto al principio, me parece que hubiéramos tenido resultados de mayor calidad, especialmente en el tema de la escalabilidad de la página.

También he agradecido nuestra planificación previa de crear “wireframes” en conjunto antes de comenzar a implementar algo, ya que al ponerse a implementar no había que, además, pensar como tenía que cuadrar todo, simplemente había que tratar de seguir los diseños previos.

También comentar los imprevistos que pueden surgir en proyectos así, por ejemplo, el cambio de políticas de la API de podcasts nos hizo tener que reajustar varias funcionalidades de la web en el último momento y quitar otra. Esto genera bastante estrés ya que además se junta con otras asignaturas, y no me gusta, a nadie le gusta.

### Saúl:

Este ha sido un proyecto más complejo que el resto de las prácticas que hemos realizado, y eso se debe al tener que adaptarnos y aprender nuevas tecnologías desde 0, lo cual provocaba situaciones en las que se descubre como hacer cosas de una manera mejor tras haberlas acabado, y también debido al trabajo en equipo. Los grupos son de mayor tamaño y hay más tareas a repartir, lo cual necesita de una cuidada planificación que, al principio, no tuvimos en cuenta.

La inexperiencia ha formado parte de muchos de los problemas que hemos tenido, sea al no saber cómo llevar a cabo una tarea o a la hora de planificar o establecer requisitos que realmente no sabíamos cuánto trabajo nos requerían. Sin embargo, al ir avanzando, todo el proceso de implementación se facilitaba al conocer mejor las tecnologías, y ver cómo todo iba encajando y funcionando provoca una gran satisfacción.

En mi opinión, cambiaría la organización inicial del proyecto, ya que no le dimos la importancia que requería, debimos establecer mejor las tareas y las fechas en las que deberían estar hechas, una parte que me afectaba activamente siendo líder del grupo de backend, donde quizás no supe desempeñar mi rol adecuadamente en un inicio.

La decisión de tener coordinadores como representantes de cada grupo me pareció acertada, ya que nos permite reunirnos y tratar temas solo entre los coordinadores facilitando la comunicación.

Finalmente, me habría gustado que se hubieran realizado más pruebas, tanto desde los frontends, como desde backend, donde personalmente no pude acabar con el objetivo que tenía pensado, posiblemente por una incorrecta medición del esfuerzo u organización del tiempo.

## 7 Anexo 1: Actas de reunión

Actas de reuniones sin profesor

Proyecto software

# Acta reunión

**13 FEBRERO 2020 / 3:30 PM / Duración: 1h30min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés.

Falta: Oscar Baselga.

## AGENDA

### Última reunión

1. Es primera reunión

#### Objetivos de la reunión

- Elección del proyecto a realizar
- Toma de requisitos

## Acciones tomadas

- Se ha elegido el proyecto de “Reproductor de música y/o podcasts”
- Se ha creado un documento para ir listando los requisitos que va a tener nuestro proyecto: [link al documento con los requisitos](#)

## Objetivos para la siguiente reunión

- Pensar más requisitos
- Pensar tecnologías a usar.
- Habrá que hacer la propuesta técnica y económica en la siguiente reunión.

## Próxima reunión

18 febrero / 10:00 AM

# Acta reunión

**17 FEBRERO 2020 / 17:00 PM / Duración: 1h**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés.

Falta: Oscar Baselga.

## AGENDA

### Objetivos de la reunión

- Discutir distintas tecnologías

### Acciones tomadas

- Opciones planteadas:
  - JHipster: Angular(Front-end web), Spring (Back-end) y BD
  - Angular(Front-end web), DJango (Back-end)
  - Ionic o Flutter para móvil.

Base de datos SQL

### Objetivos para la siguiente reunión

- Preparar la reunión con el tutor

### Próxima reunión

18 febrero / 9:00 AM

Proyecto software

# Acta reunión

18 SEPTIEMBRE 2020 / 9:00 AM / Duración: 2h + 1h tras reunión con profesor

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés.

Falta: Oscar Baselga.

## Agenda

### Continuación de la reunión anterior

1. Preparar la reunión con el tutor (tener listos los presupuestos y los resúmenes ejecutivos y técnicos).

### Objetivos de la reunión

- Preparar la reunión con el tutor
- Creación análisis de costes para el proyecto
- Creación del resumen ejecutivo

## Acciones tomadas

Saul Flores, Alejandro Facorro y Alberto Calvo se han encargado de usar la plantilla que se nos dió para crear el análisis de costes del proyecto.

Germán Garcés, Álvaro García y Luis García se encargan de la propuesta técnica y económica.

Primera idea de división:

Front-end:

- Web: Álvaro, Germán
- Android: Facorro, Luis (Líder)

Back-end:

- Servidor: Saúl (Líder)
- BD: Alberto (Coordinador)
- API: Óscar

## Objetivos para la siguiente reunión

Cada uno investigar tecnología de su parte

Terminar propuesta técnica y económica:

- Facorro, Saúl y Alberto análisis de costes y presupuesto
- Álvaro, Luis, Germán terminar propuesta (documento de google)

Próxima reunión

Viernes 21 febrero, 10:00 AM

Proyecto software

# Acta reunión

21 Febrero 2020 / 10:00 AM / Duración: 1h

## Asistencia

Alberto

Calvo

, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés.

Falta: Oscar Baselga.

## Agenda

### Objetivos de la reunión

- Terminar propuesta técnica y económica.
- Comentar tecnologías investigadas
- Repartir trabajo de la siguiente entrega.

### Acciones tomadas

- Terminada propuesta técnica y económica
- Front end: Angular (aplicación web), Flutter (móvil)
- Back end: Flask (servidor), mySQL o MariaDB (base de datos)

## Objetivos para la siguiente reunión

- Traer bocetos equipo front-end.
- Alberto base de datos, primera tablas, prueba de canciones.

## Próxima reunión

Lunes, 24 febrero, 12:00 AM

# Acta reunión

---

24 FEBRERO 2020 / 12:10 / Duración:

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Seguimiento de la última reunión

1. Se tenían que crear unos bocetos de la parte cliente.
2. Alberto tenía que echar un ojo al tema de la base de datos

### Objetivos de la reunión

- Decidir donde alojar la bbdd, que bbdd usar.
- **Entregar propuesta técnica y económica**
- Comentar la interfaz propuesta

## Acciones tomadas

1. **Reunión con el profesor:** Martes 3 de Marzo a las 11 AM
2. Se ha hablado sobre las diferentes tecnologías que se van a usar y sus ventajas sobre otras.
3. Alberto probará Heroku para ver hasta dónde puede llegar.

## Objetivos para la siguiente reunión

- Equipo back-end (Saúl y Oscar) van a preparar una pequeña propuesta de diagramas de clases.
- Alberto testea BD y su integración en Heroku.
- Equipos front-end (móvil y web) desarrollarán los interfaces para poder en la siguiente reunión finalizar el diagrama de clases.

## Próxima reunión

Viernes 28 febrero a las 4:00 PM

# Acta reunión

---

28 FEBRERO 2020 / 4 PM / Duración: 1h 40min

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

Ausencias: Saul Flores (Baja médica)

## ORDEN DEL DÍA

### Seguimiento de la última reunión

1. Dar confirmación de las interfaces creadas
2. Alberto tiene que explicar lo que ha hecho con la bbdd
3. Back end tenía que hacer una propuesta de entidad relación

### Objetivos de la reunión

- Preparar la reunión con el profesor
- Acabar el entidad relación

## Acciones tomadas

1. Interfaces no completadas ya que no se sabe cómo va a ir el tema de los podcasts
2. Discutir diferentes versiones entidad relación

## Objetivos para la siguiente reunión

- Front end: realizar mapa de navegación
- Front end: estudiar funcionamiento API Podcasts
- Back end: estudiar cómo se van a gestionar las notificaciones (modelo E/R): solicitudes, peticiones y noticias (Alberto)
- Oscar aprenderá Flask y se coordinará con Saúl para repartir lógica de negocio del back-end.

## Próxima reunión

Martes 3 de marzo, 9 AM.

# Acta reunión

---

3 Marzo 2020 / 9 AM / Duración: 2h

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

Ausencias: Saul Flores (Baja médica)

## ORDEN DEL DÍA

### Seguimiento de la última reunión

1. Dar confirmación de los mapas de navegación creados para web y móvil
2. Estudio del funcionamiento de la API de podcasts
3. Estudiar la propuesta del esquema E/R
4. Estudio de Flask

### Objetivos de la reunión

- Preparar la reunión con el profesor
- Empezar a redactar la memoria técnica

## Acciones tomadas

1. Revisión del trabajo para esta reunión (Base de datos, interfaces y proyecto de Flask comentado)
2. Añadir la posibilidad de eliminar la cuenta

## Objetivos para la siguiente reunión

- Crear proyecto de Flask
- Crear la base de datos en SQL
- El equipo de Front-end se prepara para empezar a crear la interfaz, empezando por el reproductor de música (pantalla de canción)

## Próxima reunión

Viernes 13 de marzo, 17:00.

Proyecto software

# Acta reunión

---

15 Marzo 2020 / 17:00 / Duración:

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga, Saul Flores

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisión de tareas de documentación
- Próxima reunión con el profesor (fecha y medio)
- Estado del proyecto y objetivos a futuro
  - Metodología de trabajo a distancia
  - API
  - Integración y pruebas

### Acciones tomadas

- Documentación revisada y se han repartido las tareas para terminar en el mismo día
- Probado Google Meets y Discord. Posible reunión antes de quedar con el profe
- En todos los grupos existe un reproductor funcionando y algunas funcionalidades más
- La API está medio preparada pero falta investigar más
- La integración sigue pendiente y se propone para la semana siguiente

### Objetivos para la siguiente reunión

- Integrado todo el proyecto
- Preparar reunión con el profe

### Próxima reunión

Proyecto software

# Acta reunión

---

24 Marzo 2020 / 20:00 / Duración:

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga, Saul Flores

## ORDEN DEL DÍA

### Objetivos de la reunión

- Aplicar correcciones de la reunión con el tutor.

## Acciones tomadas

- Establecer fechas de final para cada `issue` en github.
  - Comentada la manera de hacerlo de manera correcta.
- Se ha propuesto hacer **cada semana** un despliegue de la aplicación.
- Cada **dos semanas** realizar una prueba más seria, como si se fuera a entregar algo.
- Definidos los roles de los coordinadores.
- Se ha definido un workflow centralizado como forma de trabajo en github.
- Creados encargados de actas: Germán y Luis
- El documento de requisitos se indica como referencia de los requisitos terminados o parcialmente terminados

## Objetivos para la siguiente reunión

- Alberto tendrá para el miércoles la BDD desplegada.
- Continuar con las `issues` en progreso
- Cumplir los requisitos funcionales 1, 2, 3, 4, 5 y 6.

## Próxima reunión

Domingo, 29 de marzo 19:00

# Acta reunión

---

30 Marzo 2020 / 18:00 / Duración: 1 hora

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga, Saul Flores.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Organizar primer despliegue
- Aplicaciones tienen que poder reproducir canciones y listas

## Acciones tomadas

- Se muestran las dos aplicaciones de front-end para ver si se cumplen los requisitos establecidos para hoy
- Requisitos en Android:
  - Cumplidos: RF2
  - Parciales: RF1, RF4, RF5
  - Nada: RF3, RF6
- Requisitos en Web:
  - Cumplidos: RF1, RF2, RF5, RF6
  - Parciales: RF3, RF4
  - Nada
- Se ha establecido la fecha de entrega de documentación al miércoles 8 de abril

## Objetivos para la siguiente reunión

- Cumplir totalmente los requisitos funcionales del 1 al 6 en ambas plataformas.

## Próxima reunión

Domingo 5 de abril 2020

# Acta reunión

---

05 Abril 2020 / 19:00 / Duración: 2h30min

## Asistencia

Alberto Calvo, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga, Saul Flores.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Valoración requisitos con número.
- Valoración de issues con etiqueta/número
- Revisión de requisitos terminados
- Contabilización semanal de requisitos, issues y líneas de código
- Documentación
  - Añadir analizador estático de código -> medida de output

### Acciones tomadas

- **Semanalmente**, coger las 3 medidas de cada repositorio y pasarlas a Alberto.
- Issues valorados con un número del 1 al 3 en función de la importancia
- Requisitos en Android:
  - Cumplidos: RF1, RF5, RF12, RF33, RF34
  - Parciales: RF2, RF4, RF9
- Requisitos en Web:
  - Cumplidos: RF1, RF2, RF5, RF12, RF32, RF33, RF34
  - Parciales: RF4, RF7, RF9

### Objetivos para la siguiente reunión

- Tener implementados los 8 primeros requisitos.
- Terminar la documentación

### Próxima reunión

Domingo, 12 de Abril, 19h.

# Acta reunión

---

12 Abril 2020 / 19:00 / Duración: 2h30min

## Asistencia

Alberto Calvo, Álvaro García, Luis García, Alejandro Facorro, Oscar Baselga, Saul Flores.

Faltas: Germán Garcés.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Valoración requisitos con número.
- Concertar cita con el profesor.
- Revisión de requisitos terminados

## Acciones tomadas

- **Cita concertada:** 22 de abril a las 11:00.
- Tener implementados los 8 primeros requisitos.
- Terminar la documentación
- Revisión de requisitos:
  - Web:
    - Cumplidos: RF1, RF2, RF5, RF6, RF7, RF12, RF13, RF32, RF33, RF34
    - A medias: RF4, RF9, RF10, RNF9, RNF10
  - Android:
    - Cumplidos: RF1, RF5, RF7, RF12, RF13, RF33, RF34
    - A medias: RF2, RF4, RF9, RF10, RNF10

## Próxima reunión

Domingo, 19 de Abril, 19h.

Proyecto software

# Acta reunión

**19 ABRIL 2020 / 7:05 PM / Duración: 1h30min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisión de requisitos
- Preparación de la presentación
- Punto de vista general del proyecto
- Decidir cuándo se vuelve a quedar antes de la presentación y para qué

## Acciones tomadas

- Revisión de requisitos:
  - Web:
    - Cumplidos: RF8, RF32
    - A medias: RF4, RF9, RF10, RF11, RNF9, RNF10
  - Android:
    - Cumplidos: RF2, RF6
    - A medias: RF4, RF8, RF9, RF10, RF11, RF32, RNF9, RNF10
- Ha habido problemas con la presentación, horas antes de realizar la presentación con el profesor.
- Se ha revisado el porcentaje de requisitos.
- El día martes 21, se realizará una presentación para ver qué fallos se han arreglado y que fallos siguen existiendo.
- El día miércoles 22 se realizará la reunión con el profesor.

## Próxima reunión

Martes, 21 de Abril, 19h.

Miércoles, 22 de Abril, 11h.

Proyecto software

# Acta reunión

**26 ABRIL 2020 / 7:00 PM / Duración:**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisión de requisitos
- Revisar RF duplicados(listas)
- Crear RNF de seguridad de contraseña
- Establecer requisitos objetivos para la siguiente semana
- Elegir reunión con el profe

## Acciones tomadas

- Web revisado
- Android revisado
- RNF11 creado
- Objetivo requisitos: RF4, RF8, RF9, RF10, RF11, RF14, RF24, RF32, RNF5, RNF6, **RNF8**, RNF11

## Próxima reunión

- 3 de mayo 19:00
- 6 de mayo 12:00 con profesor

Proyecto software

# Acta reunión

**3 de mayo de 2020 / 5:00 PM / Duración: 2h**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisión de requisitos
- Establecer requisitos objetivos para la siguiente semana

## Acciones tomadas

- Cambios de requisitos, eliminado requisito 11 y actualizar requisito 24, no cambiar el correo de usuario
- Objetivo requisitos:
  - Requisitos para Android
    - 4,24,32(a medias),5 RNF(a medias), 8 RNF
  - Requisitos para Web
    - 4,24(a medias)
  - Requisitos funcionales 16,17,18,19,20 y terminar
- Alberto:
  - Cambiar BD Favoritas -> Favoritos
  - Que tenga sentido las relaciones de álbumes, canciones y artistas

## Próxima reunión

- Reunión profe: 6 de mayo, 12:00
  - 11:40
- Reunión grupo: 10 de mayo

Proyecto software

# Acta reunión

10 de mayo de 2020 / 5:00 PM / Duración: 2h

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisar requisitos estrictamente
- Comentar cosas a mejorar
- Decidir qué requisitos/partes se hacen para la semana siguiente
  - ¿Se mantiene Ecualizador?
    - Se quita requisito
  - ¿Se añaden/quitan requisitos?
    - RF3 filtro categorías
- Estimar cuánto nos queda
- Cerrar issues
- Control horas
- Recogida de métricas

## Acciones tomadas

- Falta decidir el requisito de filtrar por categorías.
- Hablar que aparece en el perfil
  - Listas de reproducción

## Próxima reunión

17 de mayo

# Acta reunión

17 de mayo de 2020 / 7:00 PM / Duración: 2h

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisar todos los requisitos estrictamente
- Comentar diseño y pulir fallos
- Cerrar issues
- Control horas
- Recogida de métricas

## Acciones tomadas

- Se mantiene poder compartir un podcast
- Falta decidir qué países meter como opciones
- Falta decidir el contenido de la página “AYUDA”
- ANDROID: El tiempo de la canción no se guarda
- ANDROID: Falta guardar la lista
- ANDROID: Mantener el estado de la sesión
- ANDROID: Agregar canciones desde categorías
- ANDROID: Confirmar contraseña
- ANDROID: Revisar errores null
- ANDROID: Indicar número notificaciones
- ANDROID: Mejorar el feedback
- ANDROID: Añadir imagen
- WEB DISEÑO: Icono de arriba
- WEB DISEÑO: Barra de arriba y abajo en escala de grises
- WEB DISEÑO: Color verde del botón de reproducir
- WEB DISEÑO: Añadir imagen
- WEB DISEÑO: Color de la cabecera en las tablas
- WEB DISEÑO: Cambiar artistas suscritos al menú

## **Próxima reunión**

24 de mayo

Proyecto software

# Acta reunión

24 de mayo de 2020 / 7:00 PM / Duración: 2h

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisar todos los requisitos estrictamente
- Comentar diseño y pulir fallos
- Cerrar issues
- Control horas
- Recogida de métricas

### Acciones tomadas

- Se han revisado que estén completos todos los requisitos de tal manera que solo queden pinceladas para el día de la entrega.

### Próxima reunión

25 de mayo 19:00

# Acta reunión

25 de mayo de 2020 / 7:00 PM / Duración: 2h

## Asistencia

Alberto Calvo, Saul Flores, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

## ORDEN DEL DÍA

### Objetivos de la reunión

- Revisar últimos requisitos
- Ensayar la presentación
- Comentar últimos detalles
  - Apuntar errores que no se han podido solucionar
- Guardar versión estable

## Acciones tomadas

- Presentación
  - Web: Álvaro, Germán(suplente)
  - Android: Oscar
  - Guión:
    - Pantalla inicial
      - Registro
        - Restricciones
        - Contraseña cifrada
        - Email de confirmación
      - Ayuda
      - Inicio sesión
    - Reproductor de Música
      - Play, saltar canción, aleatorio
      - Estado canción
    - Listas
      - Favoritos (Imborrable y especial)
      - Crear y eliminar
      - Filtros
      - Búsqueda (Listas, Artistas, Álbum)
      - Ordenar a mano

- Buscar canción Google
  - Buscar en el Buscador: alan (salen Álbum, Artista y canciones)
- Artistas
  - Suscripción
  - Listas de artistas
- Repetir Podcast
- Amigos
  - Buscar y añadir
  - Notificación de amistad
  - Ver perfil
  - Compartir canción, lista y podcast (Añadirlos a listas y favoritos)
  - Eliminar un amigo
- Sincronización
  - Cerrar sesión
  - Mantener sesión
  - Ver última canción
- Configuración
  - Cambiar datos
  - Cambiar foto
  - Borrar cuenta
  
- Últimos detalles a pulir:
  - Android:
    - Al iniciar sesión la canción debe estar en pause - No se ha podido hacer
    - Eliminar notificación de canción compartida (no deja por permiso, cambiar cabecera)
    - Tres puntitos de canción compartida: quitar “Eliminar”
    - Poder eliminar notificaciones desde la cruz roja
    - Cambiar colores de las notificaciones para que se diferencien
    - Al meterse en el perfil de otro usuario, en sus listas, en una canción de esa lista, que no salga la opción de eliminar
  - Web:
    - Arreglar pantalla artista suscritos
    - No mover canción en una lista al mantener pulsado, solo si se pulsa el botón
    - En configuración, cambiar la posición de la contraseña actual al final
  - Backend:
    - Alberto rellenar datos de los artistas
    - Arreglar álbumes

- Versión estable(commit):
  - Web:
  - Android:

## Próxima reunión

Reunión grupo antes de la presentación:

Presentación con profesor: 27 de mayo 12:00

**Actas de reuniones con profesor/cliente:**

# Acta reunión

**3 MARZO 2020 / 11:00 AM / Duración: 65 min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

Profesor: Rubén Bejar.

## Agenda

### Objetivos de la reunión

- Presentar el trabajo realizado hasta el momento
- Resolver dudas

## Acciones tomadas

- Notificado el coordinador
- Cambiar ‘gestiones y material’ en el presupuesto. Porque el cliente no es consciente del todo de qué van las cosas. Especificar más.
- Amortización un porcentaje fijo respecto del total del proyecto.
- Requisito adicional: Diferenciar a nivel de usuario los podcasts y las canciones. A los podcasts te suscribes, ha de llegar una notificación de cada podcasts al que estás suscrito, con las canciones no es necesario, son menos frecuentes sus publicaciones.
- Rellenar una vez a la semana el formulario en Moodle para controlar el trabajo.
- Reflejar la distribución detallada del trabajo.
- Procesos de inicio. Recursos que vamos a necesitar durante el proyecto, crear el repositorio de github es un proceso de inicio por ejemplo.
- Explicar cómo utilizaremos Heroku para el despliegue y para el desarrollo.

- Formación en los procesos de inicio, requiere reserva de tiempo.
- Decidir cómo se llevan a cabo las decisiones internas, cómo se hacen las comunicaciones de grupo.
- Mantener un registro escrito de aquello que hacemos. Control del proyecto para determinar si cumplimos los objetivos establecidos.
- Estructura de toma de decisiones.
- Definir cómo será el proceso de entrega. Concretar su despliegue, documenta y qué vamos a hacer.
- Indicar cada cuánto vamos a realizar pruebas y tests.
- Responsable o responsables de las distintas actividades, indicar cómo vamos a hacer los backups.
- Hacer integraciones lo más frecuentemente posible.
- Definir cómo controlar la calidad del código.
- Tener claro qué ha de estar en cada iteración del desarrollo del proyecto.
- En las entregas no se entregan código.
- Abrir 2 repositorios de front-end.

## **Objetivos para la siguiente reunión**

- Redacción del plan de gestión, análisis y diseño del proyecto

## **Próxima reunión**

Reunión de grupo sin profesor:

Reunión con profesor: elegir en Google

# Acta reunión

**18 FEBRERO 2020 / 11:00 AM / Duración: 40 min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés.

Profesor: Rubén Bejar.

Falta: Oscar Baselga.

## Agenda

### Objetivos de la reunión

- Presentar el trabajo realizado hasta el momento
- Resolver dudas

## Acciones tomadas

- Se ha realizado la presentación
- Se han tomado notas acerca de sugerencias y correcciones:
  - Sustituir “Proyecto XX” por Reproductor de música/podcast
  - Realizar el resumen ejecutivo en principio al final (se tiene una mejor idea)
  - En la propuesta técnica generalizar acerca de las funciones de la app
  - Diferenciar aplicación móvil y web en apartado de “funciones”
  - Indicar que es lo que nos diferencia de otras app/empresas
  - Intentar “vender” cosas como “requiere lo mínimos requisitos”
  - El ecualizador es asequible para añadirlo a la propuesta
  - Muro/red social dejar claramente como muy opcional
  - RF de búsqueda de información en Youtube, Wikipedia, etc sí
  - En la propuesta hablar de las tecnologías que le importen al cliente (nada demasiado técnico)
  - Mostrar solvencia técnica, algún diagrama

- Se podría añadir un boceto de la GUI
- Análisis de costes != presupuesto (análisis de coste va aparte)
- Guiarse con las horas de asignatura para en análisis de costes
- Horas de aprendizaje de nuevas tecnologías no mostrarlo en presupuesto
- Añadir amortización lineal del hardware
- JHipster no aconsejado, posible combinación de frameworks -> Angular y Spring/Django
- Podcast -> se pueden integrar elementos como Ibox
- La BD puede ser SQL perfectamente

## Objetivos para la siguiente reunión

- Propuesta técnica y económica entregada
- Redacción del plan de gestión, análisis y diseño del proyecto

## Próxima reunión

Reunión de grupo sin profesor: Viernes 21/02

Reunión con profesor: sin establecer aún

- Apoyo en la redacción del plan de gestión, análisis y diseño del proyecto de cara la entrega 1 del mismo

# Acta reunión

**24 MARZO 2020 / 11:00 AM / Duración: 60 min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

Profesor: Rubén Béjar.

## Agenda

### Objetivos de la reunión

- Presentar el trabajo realizado hasta el momento
- Resolver dudas

## Acciones tomadas

- Modificación de la primera versión del Plan de gestión, análisis y diseño y memoria del proyecto según los comentarios y apuntes del profesor.
- Algunos puntos relevantes de corrección han sido los siguientes:
- Requisitos no funcionales de compatibilidad de la aplicación con dispositivos y cifrado de contraseñas
- Qué información y procesos debe seguir un nuevo miembro del proyecto
- Criterios, medidas y dificultades de las Issues
- Condiciones de finalización de Issues
- Modificación del Diagrama de Gantt y Diagrama de Despliegue
- Indicación de instalación únicas de nuestro despliegue
- Incluir varios bocetos de nuestra aplicación
- Indicar que tipo de test se va a utilizar y el objetivo de cumplimiento de esos test
- Mayor comunicación entre los miembros del grupo
- Unión de requisitos front end móvil y web, de forma sea desde el punto de vista del usuario

- También se ha decidido que las preguntas citadas por el profesor se tendrán en cuenta a partir de la reunión, de manera que se cumplirán para futuras reuniones tanto con el profesorado como los miembros del proyecto.

## Objetivos para la siguiente reunión

- Corregir las indicaciones del profesor
- Preparar la presentación

## Próxima reunión

Reunión con profesor:

Dia 21 o 22 de abril

# Acta reunión

**22 de abril 2020 / 11:00 AM / Duración: 50 min**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Germán Garcés, Oscar Baselga.

Profesor: Rubén Béjar.

## Agenda

### Objetivos de la reunión

- Presentar el trabajo realizado hasta el momento

## Acciones tomadas

- Se ha mostrado las dos aplicaciones al profesor

### Objetivos para la siguiente reunión

- Corregir las indicaciones del profesor
  - Android:
    - Mostrar la cola de reproducción cuando está activada la opción aleatorio.
    - Poder ordenar las canciones de una lista por nombre o por cualquier otro aspecto.
    - Mejorar presentación, vender el producto
  - Web:
    - Realimentación en lo que se puede pulsar
    - Poder ordenar por nombre o por algún criterio
  - Documentación:
    - Agrupar los requisitos que estén muy relacionados
    - Especificar que tipo de ecualizador en requisitos

Próxima reunión

Con profesor: en 2 semanas aproximadamente (pendiente por reservar)

De grupo: domingo 26 19:00

# Acta reunión

**6 de mayo 2020 / 12:00 AM / Duración: 1h**

## Asistencia

Alberto Calvo, Saul Flores, Alvaro García, Luis García, Alejandro Facorro, Oscar Baselga, Germán Garcés

Profesor: Rubén Béjar.

## Agenda

### Objetivos de la reunión

- Comentar la memoria v2

## Acciones tomadas

- Se ha comentado la memoria con las correcciones necesarias

### Objetivos para la siguiente reunión

- Corregir las indicaciones del profesor
- Documentación:
  - 2. Organización del proyecto
    - Nombres en github
  - 3.1.1 Procesos de inicio del proyecto
    - Poner urls repos que hay
  - 3.1.2 Procesos de ejecución y control del proyecto
    - Url discord
    - Definir rol de redactor de actas y asignarlo. Sustituir los nombres por los roles al nombrarlos después
    - Enseñar Métricas del proyecto (Gráficos y tablas) en la versión final.
    - Poner las referencias de los colores en esos colores
    - Cambiar “intentando modularizar” por modularizando basándose en los requisitos y la planificación
  - 3.1.3 Procesos técnicos
    - Quitar versiones concretas, pero sí nombrar algunas bibliotecas (las más importantes) y en qué ficheros se

- encuentran anotadas (Backend: requirements.txt, Android... Web...)

  - README.md Explicar procesos de instalación en el README.md y desde la memoria explicar primero un poco todo y que hay más detalle en el fichero readme.md por posibles cambios
- 3.2.3 Plan de aseguramiento de la calidad
  - Indicar objetivos de los test (Backend pj toda la API), pensar si poner porcentajes (Backend 100% no? por peticiones de test a toda la API)
- 3.2.4
  - Alargar los días del diagrama de Gantt, hasta junio
  - Falta la idea de PLANIFICACIÓN de futuro en el diagrama de Gantt, no solo lo que se ha ido haciendo
- 4.1 Análisis de requisitos
  - No diferenciar RNF entre los de sistema y los de backend y frontend.
  - Asegurarse de que la última versión tiene bien los requisitos con código RF y RNF
- 4.2 Diseño del sistema
  - Diagrama de despliegue:
    - Angular en el navegador del usuario cuando se hace el despliegue final.
    - Cambiar diagrama de despliegue con la aplicación web del cliente está fuera de Heroku
  - Diagrama de clases de cliente distinto para cada frontend
  - Añadir el patrón arquitectural Singleton y los patrones que estén incluidos en los frameworks
  - **Documentar TODOS los diagramas (explicarlos)**
  - Bibliotecas en los diagramas de paquetes y módulos
  - Añadir clases de **conexión y modelo de datos** de Flask a la base de datos (*Domain modeling* en Google)
  - El diagrama de objetos de la BD en el servidor son las tablas
  - Cambiar los diagramas de clases de la BD por el E/R e indicar que se está utilizando un ORM (SQLAlchemy)
  - Diagrama de componentes y conectores: profundizar en los clientes indicando ¿qué componentes hacen las peticiones

[http?](http://) Si hay un patrón indicar con un ejemplo (Proxy?).  
Añadir en Servidor una cajita que ponga modelo de datos  
(y luego se explica)

- Documentar la API con un link a la wiki de github
- Añadir referencias y descripción a las figuras del documento (Alberto Latex)
- Comentar un poco las interfaces en las imágenes de los prototipos
- Explicar el mapa de navegación de web y android
  - Añadir un caso de uso para registro
  - Añadir texto o títulos de pantalla (central, registro) o por secciones
- 5.2 Ejecución y control del proyecto
  - Poner gráficas y métricas del proyecto
  - Quitar el término ‘divergencias’ del calendario del proyecto
- Memoria global:
  - Corregir referencias
  - Se puede usar la primera persona gramatical para que quede más claro. Es posible decir “Nos decantamos por usar Heroku”.
- Enviar apk y url el día de antes de la entrega final

## Próxima reunión

Con profesor: Demostración final 27 de mayo 12:00

Grupo: 10 de mayo 19:00

## 8 Anexo 2: Bibliografía

- [1] «Repositorio backend,» [En línea]. Available: <https://github.com/UNIZAR-30226-2020-10/back-end>.
- [2] «Repositorio frontend web,» [En línea]. Available: [https://github.com/UNIZAR-30226-2020-10/front-end\\_web](https://github.com/UNIZAR-30226-2020-10/front-end_web).
- [3] «Repositorio frontend android,» [En línea]. Available: [https://github.com/UNIZAR-30226-2020-10/front-end\\_android](https://github.com/UNIZAR-30226-2020-10/front-end_android).
- [4] «Repositorio documentación,» [En línea]. Available: <https://github.com/UNIZAR-30226-2020-10/documentacion>.
- [5] «Tutoriales Flask,» [En línea]. Available: <https://www.youtube.com/playlist?list=PLosiE80TeTs4UjLw5MM6OjgkjFeUxCYH>.
- [6] «Tutoriales SQLAlchemy y API,» [En línea]. Available: <https://www.youtube.com/watch?v=PTZiDnuC86g>.
- [7] «Angular,» [En línea]. Available: <https://angular.io/stat>.
- [8] «Tutorial TypeScript y HTML,» [En línea]. Available: <https://auth0.com/blog/amp/building-an-audio-player-app-with-angular-and-rxjs/>.
- [9] «Guía Android Studio,» [En línea]. Available: <https://developer.android.com/guide>.
- [10] «Tutorial Flutter,» [En línea]. Available: <https://www.youtube.com/watch?v=1ukSR1GRtMU&list=PL4cUxeGkcC9jLYyp2Aoh6hcWuxFDX6PBJ>.
- [11] «Discord Grupo 10,» [En línea]. Available: <https://discord.gg/bTTxhZu>.
- [12] «Instalación Heroku CLI,» [En línea]. Available: <https://devcenter.heroku.com/articles/heroku-cli>.
- [13] «Android Developer,» [En línea]. Available: <https://developer.android.com/design>.
- [14] «W3 Standards,» [En línea]. Available: <https://www.w3.org/standards/>.
- [15] «Spotify GCase standards guide,» [En línea]. Available: <https://gcase.files.wordpress.com/2015/10/spotify-developer-brand-standards-guide-02jr.pdf>.
- [16] «Spotify branding guidelines,» [En línea]. Available: <https://developer.spotify.com/branding-guidelines/>.
- [17] «Shazam brand guidelines,» [En línea]. Available: [https://issuu.com/duncanriley/docs/01\\_sz\\_brand\\_guidelines\\_lite](https://issuu.com/duncanriley/docs/01_sz_brand_guidelines_lite).

- [18] «Roboto Mono,» [En línea]. Available: <https://fonts.google.com/specimen/Roboto+Mono>.
- [19] «Prettier,» [En línea]. Available: <https://prettier.io/>.
- [20] «Codelyzer,» [En línea]. Available: <https://www.npmjs.com/package/codelyzer>.
- [21] «Dartanalyzer,» [En línea]. Available: <https://dart.dev/tools/dartanalyzer>.
- [22] «Pylint,» [En línea]. Available: <https://pypi.org/project/pylint-flask-sqlalchemy/>.
- [23] «Cloc,» [En línea]. Available: <http://cloc.sourceforge.net/>.
- [24] «Heroku: Cloud Applicattion,» [En línea]. Available: <https://www.heroku.com>.
- [25] «Listen Notes,» [En línea]. Available: <https://www.listennotes.com/>.
- [26] «Firebase,» [En línea]. Available: [https://firebase.google.com/](https://firebase.google.com).
- [27] «Wiki Documentación: API,» [En línea]. Available: <https://github.com/UNIZAR-30226-2020-10/documentacion/wiki/API>.