Proyecto UPBEAT
Grupo 03. Barbara Liskov

Plan de gestión, análisis y memoria del proyecto

Alejandro Ruiz Sumelzo
Alejandro Piedrafita Barrantes
Álvaro Santamaría De la Fuente
Fernando Navarro Zarralanga
José Félix Yagüe Royo
Víctor Pérez Sanmartín
Sergio Torres Castillo

16 de marzo de 2020

Introducción

El proyecto *Upbeat* consiste en el desarrollo e implementación de un reproductor de música en streaming.

La aplicación está diseñada para todo tipo de público, incluidos artistas. Permitirá a los usuarios escuchar canciones o podcasts, crear listas de reproducción, añadir canciones o podcasts a su lista de favoritos para después acceder a sus contenidos preferidos de una forma más rápida o seguir a otros usuarios entre otras funcionalidades.

De manera similar, los artistas tendrán las mismas características que cualquier usuario además de poder subir canciones y/o crear álbumes. Esto será algo exclusivo de las cuentas de clientes registradas como artistas.
Añadirá aspectos de la 'red social', tales como seguir a otros usuarios o sus playlist creadas.

Otras de las características diferenciales de nuestra aplicación será el uso de banners de publicidad o la implementación de un ecualizador de sonido que modificará la manera de reproducir las canciones.

Por último, el reproductor de música *Upbeat* permitirá realizar una búsqueda por título de canción, género, artista y país de la misma.

Se permitirá usar la aplicación de tres maneras diferentes: Mediante una aplicación web (navegador Chrome), una aplicación para dispositivos*Android* y una aplicación para dispositivos *iOS*.

Las 3 versiones de la aplicación serán muy similares, salvo que un pequeño número de características solo estarán disponibles en la versión web, como el ecualizador de sonido o los banners de publicidad.

Usará también un sistema de sincronización, permitiendo que el usuario retome cualquier canción donde la había dejado, en todos los dispositivos.

Índice

Página

1. Organización del proyecto **4**

2. Plan de gestión del proyecto **6**
2.1. Procesos 6
2.1.1. Procesos de inicio del proyecto 6
2.1.2. Procesos de ejecución y control del proyecto 6
2.1.3. Procesos técnicos 8
2.2. Planes 9
2.2.1. Plan de gestión de configuraciones 9
2.2.2. Plan de construcción y despliegue del software 10
2.2.3. Plan de aseguramiento de la calidad 11
2.2.4. Calendario del proyecto y división del trabajo 11
2.2.5. Diagramas de Gantt de la aplicación 14

3. Análisis y diseño del sistema **16**
3.1. Análisis de requisitos 16
3.2. Diseño del sistema 18

1. Organización del proyecto

Integrante	Puesto	Responsabilidades
Alejandro Ruiz Sumelzo	Director del proyecto. Coordinador y desarrollador del grupo de back-end. Encargado de la documentación del análisis y diseño del sistema.	Responsable de redactar algunas actas en reuniones con el profesor. Control de la distribución de trabajo (elaboración de calendario) y revisión de esfuerzos. Desarrollador de modelos, repositorios y controladores de la API. Encargado del despliegue del back end sobre el servidor.
Alejandro Piedrafita Barrantes	Desarrollador de apoyo para el grupo de back-end	Realización de tareas de gestión (edición de memoria y otros documentos). Desarrollador de modelos, repositorios y controladores de la API. Diseño del sistema mediante diagramas.
Víctor Pérez Sanmartín	Desarrollador de apoyo para el grupo de back-end	Responsable de redactar algunas actas en reuniones con el profesor. Realización de tareas de gestión (edición de memoria y otros documentos). Desarrollador de modelos, repositorios y controladores de la API. Encargado del diseño e implementación de la base de datos.
Álvaro Santamaría de la Fuente	Desarrollador del grupo de front-end móvil.	Realización de tareas de gestión (edición de memoria y otros documentos). Desarrollo e implementación del front-end de la aplicación móvil. Implementar la lógica de la aplicación de *Android* e *iOS*. Encargado de unificar las partes de la aplicación móvil y llevar a cabo el despliegue en *Android* e *iOS*.
José Félix Yagüe Royo	Desarrollador del grupo de front-end móvil. Encargado de la gestión de la parte front móvil del proyecto.	Realización de tareas de gestión (edición de memoria y otros documentos). Desarrollo e implementación del front end de la aplicación móvil. Encargado de la conexión con el API REST. Control de la distribución de trabajo y coordinación dentro del grupo front-end de móvil.

Integrante	Puesto	Responsabilidades
Fernando Navarro Zarralanga	Coordinador y desarrollador del grupo de front-end web. Encargado de la gestión de la parte front web del proyecto.	Realización de tareas de gestión (edición de memoria y otros documentos). Diseñar pantallas de la aplicación de la web. Implementar la lógica de la aplicación web. Encargado de llevar a cabo el despliegue de la web.
Sergio Torres Castillo	Desarrollador de apoyo para el grupo de front-end	Realización de tareas de gestión (edición de memoria y otros documentos). Diseñar pantallas de la aplicación de la web. Implementar parte de la lógica de la aplicación web.

2. Plan de gestión del proyecto

2.1. Procesos

2.1.1. Procesos de inicio del proyecto

En el equipo de front-end de movil se va a utilizar *Android Studio* como entorno de desarrollo, y *Flutter* como *SDK*, el cual utiliza *Dart* como lenguaje de programación. Para la realización de las pruebas, se van a utilizar tanto un emulador, como un móvil real con *Android 10*. Para *iOS* se va a instalar *XCode* en *Mac* ya que es el IDE oficial de *Apple*, pero se va a mantener tanto *Flutter* como *Dart* para su desarrollo, y un emulador para las pruebas. La aplicación requerirá tener instalado *Android* 5.0 mínimo para funcionar.

De manera similar, para la realización de la aplicación web, solamente es necesario tener *Google Chrome* para hacerlo funcionar. El desarrollo se realizará sobre *VSCode*.
Se ha elegido un tamaño de 15 GB en la BD, suficiente para llevar a cabo pruebas con diversas canciones y configuraciones.

2.1.2. Procesos de ejecución y control del proyecto

Las comunicaciones internas se llevarán a cabo mediante un grupo de *WhatsApp* estimado para ello, para cualquier otra duda o comunicación entre componentes del equipo se hará de forma individual. La resolución de tareas será totalmente independiente y completa en el entorno de *GitHub*.

Los responsables de realizar la puesta en marcha serán los encargados de la parte front-end y de la parte back-end. La creación de copias de seguridad y semejantes se realizarían de manera automática gracias a *GitHub*.
El repositorio que se creará con todos los archivos referentes al proyecto se encontrará en *GitHub*, para que todos los integrantes del proyecto puedan acceder fácilmente a los archivos. Además, se usará el *Issue Tracker* de *GitHub* para la gestión de incidencias; el director del mismo se encargará de crear las incidencias principales, y cada uno de los encargados de cada parte las completarán en cada uno de sus ámbitos.

El proyecto estará dividido en varios repositorios: uno específico para front-end, otro para back-end, y la memoria. Para conseguir que no se modifique el mismo fichero por dos personas al mismo tiempo y evitar problemas, cada equipo tendrá más sub-ramas de desarrollo, por ejemplo, una para cada miembro del equipo, que serán actualizadas con cambios no siempre funcionales y cuando sean más estables se volcarán a la rama de desarrollo principal.
En la rama principal de cada uno de los repositorios, sólo podrá haber una versión funcional del sistema, que antes de ser subida será sometida a diferentes test automáticos, entre los que se incluirán test para comprobar la estabilidad del sistema (pruebas de sobrecarga) y test que revisarán las acciones disponibles

para comprobar los requisitos que se han resuelto, además de ser testeado por varios miembros del equipo.
Para que lo desarrollado en cada uno de estos repositorios pase al repositorio funcional, cada líder de las respectivas partes revisará el código actualizado y si todo está correcto se considerará válido.

Todas las semanas habrá una serie de tareas asociadas mediante el *Issue Tracker*. Al final de cada semana, el responsable de cada parte del proyecto revisa las tareas que se han realizado esa semana y se realiza un seguimiento de las mismas: si hay tareas que no se han cumplido, se asigna automáticamente para la siguiente semana, siendo estas tareas las primeras que se deberán hacer.
Además, cada semana el responsable de cada una de las partes revisará cuántas tareas se han realizado para la siguiente iteración, controlando si han sido completadas o no y cuánto tiempo de retraso acumula respecto a la planificación original. Tras esta revisión, este responsable asignará las tareas de la semana a todos los miembros del equipo que puedan trabajar esa semana.

Durante el desarrollo del proyecto puede haber problemas y disputas entre los miembros del equipo. Para tratar de resolverlos los coordinadores serán los primeros en mediar entre los miembros en disputa y, si hay alguna razón que haga imposible esta mediación, será el resto del equipo quien deberá votar en consecuencia.
Todos los componentes del equipo son capaces de modificar los ficheros de los repositorios, excepto en el de las versiones, el cual solo podrán subir archivos y modificarlos los líderes del front-end y el back-end.

2.1.3. Procesos técnicos

En primer lugar, en el front-end web se ha utilizado la herramienta *Angular* que utiliza *JavaScript*, *HTML* y *CSS*. Se han utilizado librerías ya predefinidas en la guía de estilo *Angular Material* para los elementos de diseño. Para desplegar y probar la aplicación hemos empleado el navegador *Google Chrome*. Para usar esta herramienta ha sido necesario utilizar el manual de *Angular* y algunos tutoriales online de *JavaScript* y *CSS*.

Para el desarrollo del software de la parte del Front-end de la aplicación móvil se va a utilizar el entorno de *Android Studio* haciendo uso de su integración con *git* para facilitar el control de versiones y así, gestionar de forma uniforme entre los dos integrantes del grupo, el repositorio *Flutter* en el que se irá desarrollando el software correspondiente a la aplicación móvil. Para las pruebas en el dispositivo de *iOS* se utilizará el entorno de *VSCode*.

Para el proceso de prueba de funcionalidades de la aplicación móvil se seguirá el siguiente guión:

- Primero el desarrollador se asegurará de tener la última versión del repositorio *Flutter*, efectuando un pull de la rama que desea testear.
- A continuación se abre el entorno de desarrollo (VSCode en caso de que se vaya a probar la aplicación en iOS o Android Studio en el caso de un dispositivo Android) y se procede a buildear el código de la app. En caso de encontrar errores se notificarán al encargado de haber realizado esa parte del código.
- Se ejecuta la aplicación libre de errores de compilación en el dispositivo deseado (iOS o Android)
- Posteriormente se realiza la prueba de la nueva funcionalidad implementada, comprobando que no se dan situaciones de error ni resultados inesperados.
- Si las pruebas realizados han sido satisfactorias, se informará al coordinador del grupo de front-end móvil y realizará un merge de esa rama con la rama máster.

2.2. Planes

2.2.1. Plan de gestión de configuraciones

La convención de nombres utilizadas para nombrar los distintos archivos sería la siguiente:

Las versiones solo se modificarán cada vez que se produzcan cambios suficientemente importantes, como por ejemplo la implementación de una nueva funcionalidad. Cada vez que se cree una nueva versión, pero sus cambios sean menores, como resolución de errores, se modificará su número de revisión, pero no de versión. Se crearán ficheros de documentación que permita ir recopilando toda la información referente a los cambios. Además, en los ficheros de documentación en los que se expliquen las diversas funcionalidades que tiene la aplicación y que errores se han ido resolviendo, cuando estos sean de una nueva versión o revisión solo se ofrecerá la información sobre los cambios que existan entre esta y la versión o revisión anterior, pero siempre que se cambie la versión se documentarán los cambios respecto a la primera revisión de la versión anterior (p.ej. La versión 2.1 solo contendrá las novedades respecto a la versión 2.0, pero la versión 3.0 contendrá todos los cambios que hayan sucedido desde la versión 2.0 aunque la mayoría se hayan documentado ya en las revisiones).

2.2.2. Plan de construcción y despliegue del software

Respecto a la parte front-end web, la manera utiliza para desplegar el software para probarlo se ejecuta mediante la línea de comandos y un servidor virtual accesible desde un navegador local (como *Chrome*) mediante la ruta *localhost* y el puerto *4200*. No se requiere ningún tipo de autentificación para ello. Algo a destacar es la omisión por parte de *GitHub* de la carpeta de módulos de Angular (librerías que utiliza el proyecto), ya que ocupa demasiado, de tal forma que cada uno tiene que tener su propia carpeta de librerías.

Para la parte móvil, el uso de *Android Studio* y *Gradle* permite la instalación automática de las dependencias necesarias del proyecto, además de compilar el mismo cada vez que se prueba una parte del código.

Para comprobar cada una de las funcionalidades que se van añadiendo, se realizan de manera manual, primero en los ordenadores personales de cada integrante y luego integrándolo con el servidor de producción. Los módulos son independientes entre ellos y se comunican principalmente mediante peticiones *HTTP*, esto permite simular las peticiones antes de probarlo con el entorno real de producción. Este servidor de producción será *Heroku*, pero aún no se ha implementado todo para poder ir subiendo las diferentes versiones de manera correcta.

En la parte referente al back-end, la construcción y despliegue se realizan de la siguiente manera:

1. Se lanza la base de datos *upbeat* en *PostgreSQL* (de momento únicamente en local) con usuario: *postgres* y contraseña: *root* en el puerto *5432*.
2. Establecer puerto en las propiedades del proyecto *Spring* para las conexiones *HTTP* (por defecto se usará el *8080* pero podría llegar a cambiarse en caso de puertos ya ocupados/escuchando).
3. Compilar el proyecto utilizando *Maven*, el cual facilita las dependencias de nuestro proyecto asegurándose que todo quede compilado con las mismas dependencias y funcione correctamente.
4. Lanzar el proyecto de *Spring* (*Run*)*.
5. Para la comprobación del correcto funcionamiento de la parte referente al back-end, antes de conectar con los respectivos proyectos de *Angular* y *Flutter* (front-end) se utilizará la aplicación *Postman*. Esta herramienta nos permite hacer peticiones *GET* y *POST* y poder observar su resultado.

*En caso de encontrar errores, notificar al responsable de dicha parte o, si es posible, solucionar errores cuando sean mínimos y se sepa cómo corregirlos. Para el despliegue final del servidor y la base de datos en la entrega final se buscará utilizar plataformas como *Heroku*.

2.2.3. Plan de aseguramiento de la calidad

Para asegurar la calidad, en el caso de la aplicación *Android* se usará la guía de estilo de *Android* [1], siguiendo algunos consejos que ahí se comentan. En la aplicación web, se usará determinados ejemplos de la página de *Angular* [2] creada por *Google*, la cual permite usar y conocer conceptos importantes a la hora de hacer un código limpio y eficiente.

Además, siempre se realizarán las pruebas necesarias antes de realizar un *commit* con la última actualización del proyecto validado. Por lo que la versión que residirá en *GitHub* será la correcta.
Una vez que la base de datos, aplicación web y móvil, estén terminadas y funcionen con el servidor, se seguirán unas determinadas pautas para comprobar que la aplicación funciona correctamente en todos ellos. Unido a todo esto, es imprescindible el uso de ambas aplicaciones por personas ajenas al proyecto, recogiendo opiniones sobre la usabilidad del sistema y detectar posibles errores.

2.2.4. Calendario del proyecto y división del trabajo

En la primera iteración del proceso de diseño nos centraremos en desarrollar las funcionalidades principales del sistema, mientras que en la segunda iteración se corregirán todos los errores encontrados en la primera, se implementarán las funcionalidades secundarias y se afinara el diseño de la página web y de las aplicaciones móviles para que sean más agradables al usuario.
Para la primera iteración, se planea permitir la creación, edición y borrado de clientes con sus credenciales básicos: nombre de usuario, nombre real, correo, contraseña. Unido a esto, comprobar si las entidades Artista y Usuario se crean y borrar correctamente. También se permitirá la subida de canciones por parte de los artistas; estas canciones serán visibles en la aplicación y podrán ser reproducidas (al igual que los podcasts). Los álbumes estarán disponibles con su descripción y podrán ser consultados, reproduciendo cada una de sus canciones.
Para la segunda iteración se finalizarán los requisitos que, por falta de tiempo, no pudieron ser completados en la primera y se añadirán funcionalidades al sistema. Estas funcionalidades son: añadir canciones a la lista de reproducción de un usuario, permitir información adicional en los perfiles de usuario (como puede ser una foto de perfil, una descripción, etc.), además de poder seguirse entre dos usuarios. Se permitirá la búsqueda y filtrado de determinadas canciones y/o álbumes por unos determinados parámetros, al igual que utilizar un ecualizador en la aplicación web con el uso de *banners*.

[1] https://developer.android.com/design
[2] https://angular.io/

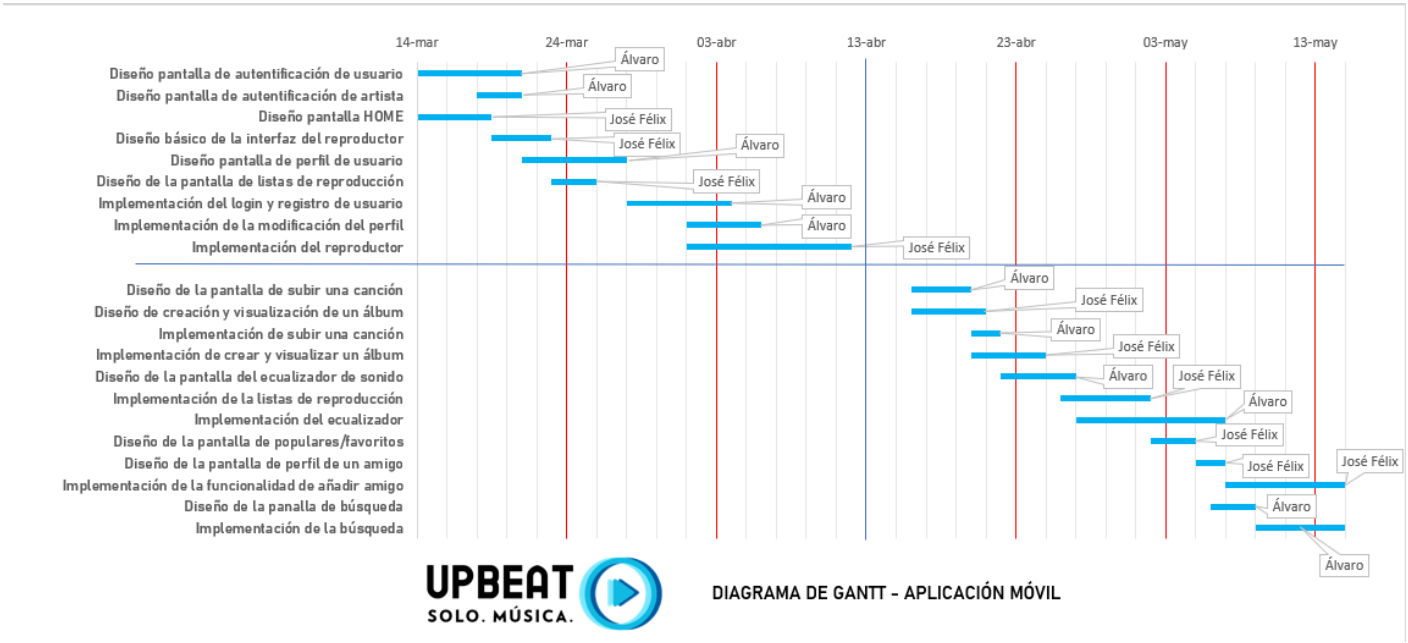
ABRIL
2020

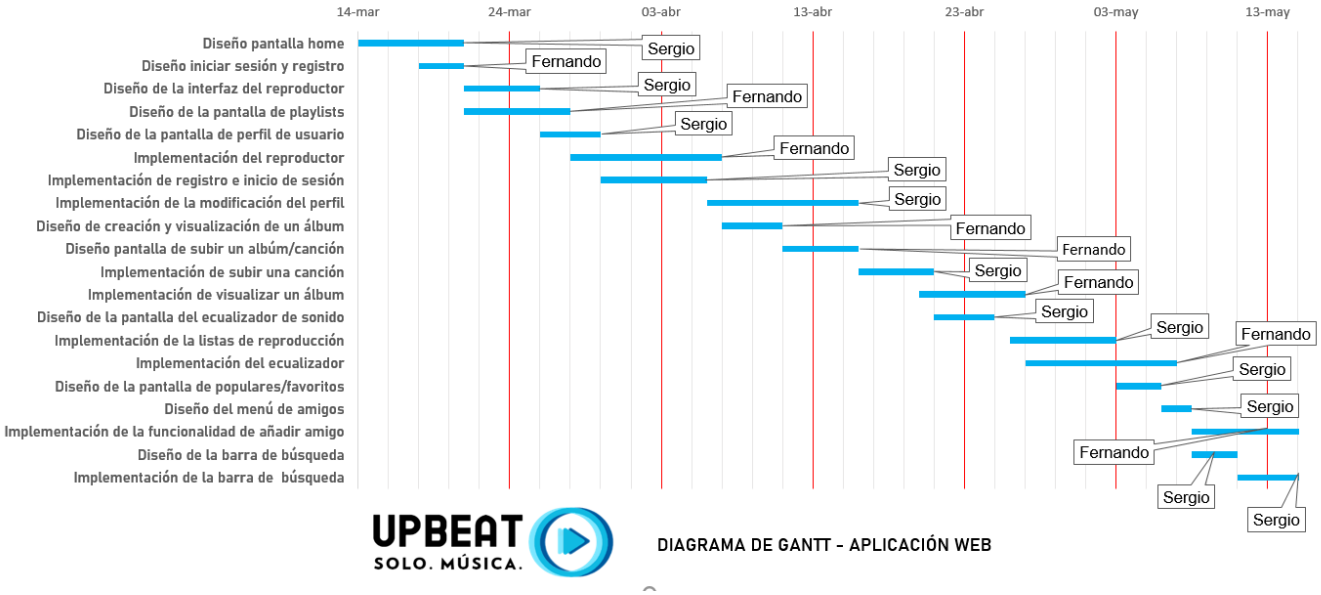
SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
			1	*2*	*Proyecto Upbeat* *3* *Back: lógica del login y registro, modificación del perfil.*	*Proyecto Upbeat* *4* *Web y móvil: vistas de login y registro, modificación del perfil.*
5	*Proyecto Upbeat* *6* *Primera versión de la web y la aplicación, que permita a un cliente registrarse, iniciar sesión y modificar el perfil.*	*7*	*8*	*9*	*Proyecto Upbeat* *10* *Back: lógica para reproducir una canción.*	*Proyecto Upbeat* *11* *Web y móvil: ver y reproducir una canción, pantalla de inicio.*
12	*Proyecto Upbeat* *13* *Pruebas conjuntas de las funcionalidades y corrección de errores.*	*14*	*Proyecto Upbeat* *15* *Entrega de la segunda versión de la memoria.*	*16*	*Proyecto Upbeat* *17* *Back: lógica para crear álbumes y subir canciones.*	*Proyecto Upbeat* *18* *Web y móvil: vistas para crear álbumes y subir canciones.*
19	*Proyecto Upbeat* *20* *Pruebas conjuntas de las funcionalidades y corrección de errores.*	*21*	*22*	*23*	*Proyecto Upbeat* *24* *Web y móvil: vistas para ver y escuchar un álbum con una o varias canciones, además de la información del mismo.*	*Proyecto Upbeat* *25* *Back: lógica de un álbum con una o varias canciones, además de la información del mismo.*
26	*Proyecto Upbeat* *27* *Pruebas conjuntas de las funcionalidades y corrección de errores.*	*28*	*29*	*Proyecto Upbeat* *30* *Despliegue de la lógica y BD en servidor en la nube.*		

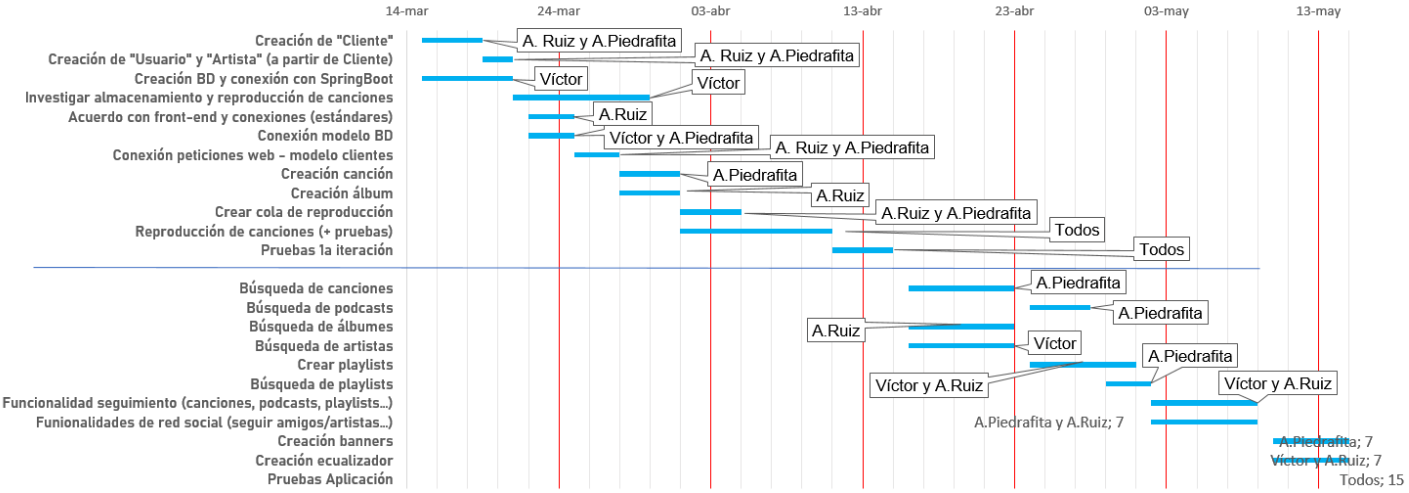
MAYO
2020

SUNDAY	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY
					Proyecto Upbeat 1 *Web y móvil: vistas para crear listas de reproducción y canciones favoritas.*	*Proyecto Upbeat* 2 *Back: lógica de creación de las listas de reproducción y canciones favoritas.*
3	*Proyecto Upbeat* 4 *Pruebas conjuntas de las funcionalidades y corrección de errores.*	5	6	*Proyecto Upbeat* 7 *Despliegue de la lógica y BD en servidor en la nube.*	*Proyecto Upbeat* 8 *Web y móvil: vistas para seguir a otros usuarios y listas de reproducción.*	*Proyecto Upbeat* 9 *Back: lógica para seguir a otros usuarios y listas de reproducción.*
10	*Proyecto Upbeat* 11 *Pruebas conjuntas de las funcionalidades y corrección de errores.*	12	13	*Proyecto Upbeat* 14 *Despliegue de la lógica y BD en servidor en la nube.*	*Proyecto Upbeat* 15 *Web: equalizador y banners, aplicación final.*	*Proyecto Upbeat* 16 *Back: banners, lógica y BD final.*
Proyecto Upbeat 17 *Despliegue de la lógica y BD en servidor en la nube. Pruebas y depuración.*	*Proyecto Upbeat* 18 *Pruebas y depuración.*	*Proyecto Upbeat* 19 *Pruebas y depuración.*	*Proyecto Upbeat* 20 *Pruebas y depuración.*	*Proyecto Upbeat* 21 *Pruebas y depuración.*	*Proyecto Upbeat* 22 *Pruebas y depuración.*	*Proyecto Upbeat* 23 *Pruebas y depuración.*
Proyecto Upbeat 24 *Pruebas y depuración.*	*Proyecto Upbeat* 25 *Pruebas y depuración.*	*Proyecto Upbeat* 26 *Pruebas y depuración.*	*Proyecto Upbeat* 27 *Pruebas y depuración.*	*Proyecto Upbeat* 28 *Pruebas y depuración.*	*Proyecto Upbeat* 29 *Pruebas y depuración.*	*Proyecto Upbeat* 30 *Pruebas y depuración.*

2.2.5. Diagramas de Gantt de la aplicación







UPBEAT SOLO. MÚSICA.

DIAGRAMA DE GANTT - BACK-END

3. Análisis y diseño del sistema

3.1. Análisis de requisitos

Se exponen los siguientes requisitos funcionales de la aplicación:

Requisito funcional	Descripción
RF1	El sistema permite la existencia de clientes.
RF2	Los clientes podrán ser usuarios normales o artistas.
RF3	Los clientes se registrarán mediante usuario y contraseña.
RF4	Un usuario es un cliente registrado que puede acceder a las funciones de la aplicación como escuchar canciones/podcasts, crear y seguir listas de reproducción, seguir a otros usuarios, escuchar álbumes y ver las canciones subidas por los artistas.
RF5	Un artista es un cliente registrado que puede realizar las mismas funciones que un usuario, además de crear álbumes y subir canciones. Crear un álbum consiste en la creación de un grupo de canciones que pertenecen al mismo artista.
RF6	Un cliente se compone de un nombre único, nombre personal y apellidos, un correo electrónico, una contraseña para acceder a la aplicación, una foto de perfil y las redes sociales que posee (si quiere añadirlas de forma opcional).
RF7	El sistema permite que los clientes modifiquen los datos de su perfil, es decir, su nombre, correo, contraseña, foto de perfil e información de las redes sociales.
RF8	Los clientes accederán a la aplicación mediante una aplicación móvil o una aplicación web.
RF9	El sistema permite reproducir y pausar una canción y/o podcasts. También permite saltar a la siguiente (si la hubiera), retroceder a la anterior, y elegir un bucle de la misma o reproducir aleatoriamente varias canciones y/o podcasts.
RF10	Una canción se compone de un título, un audio, artista/s que la han creado, género de la canción, país de la canción y veces que ha sido reproducida.
RF11	Una lista de reproducción es una lista de canciones generadas por un cliente.
RF12	El sistema permite que un cliente cree y borren listas de reproducción creados por ellos mismos, las cuales serán públicas.
RF13	El sistema permite que los clientes añadan canciones a las listas de reproducción creadas por ellos mismos.
RF14	El sistema permite que los clientes sigan listas de reproducción creadas por otros usuarios.
RF15	El sistema permite que solo los artistas puedan subir canciones.
RF16	El sistema permite buscar una determinada canción por nombre o género, además de un artista por nombre.

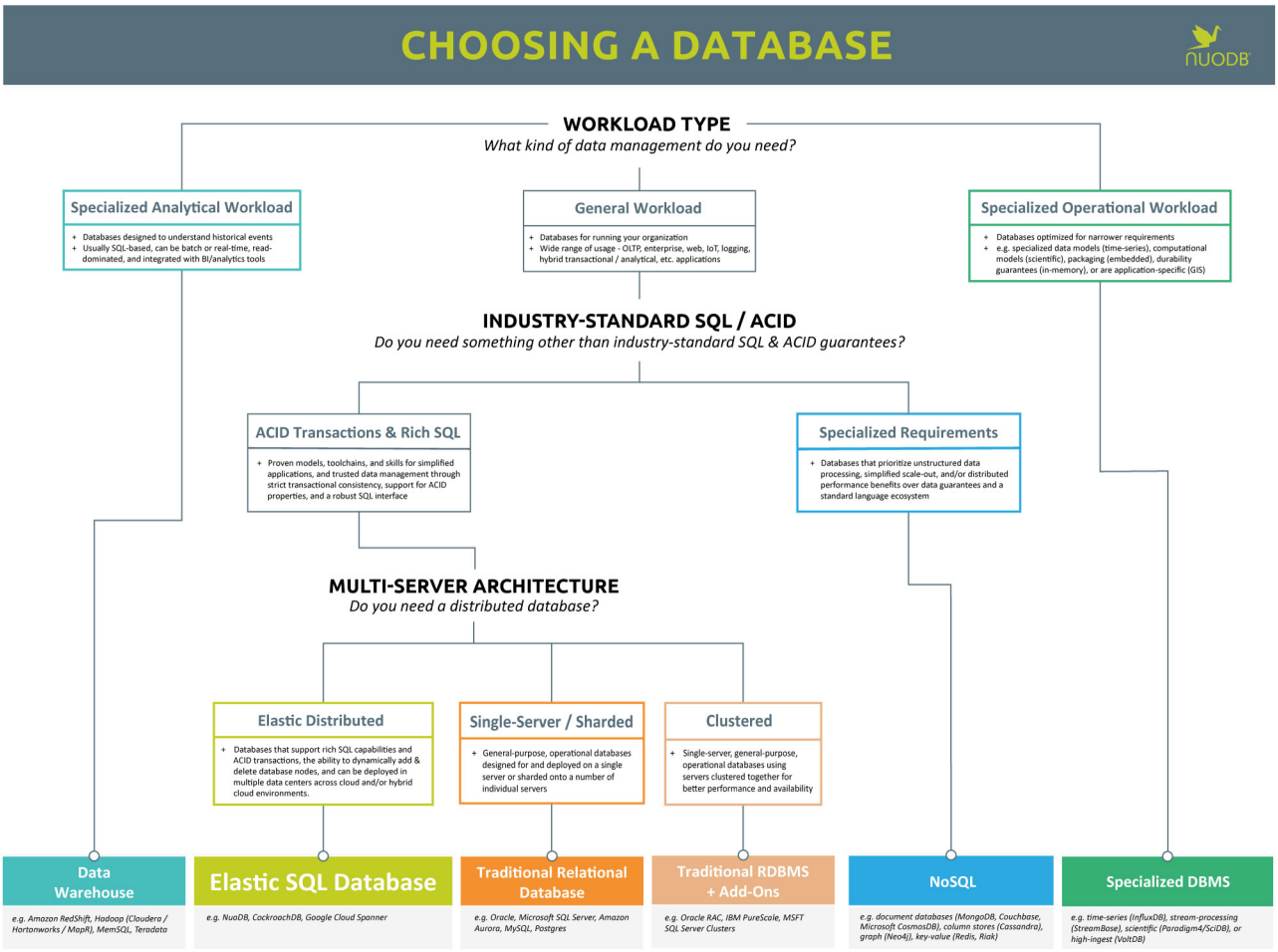
Requisito funcional	Descripción
RF17	Las búsquedas por palabras clave deberán al menos contener una palabra, de mínimo cuatro caracteres.
RF18	El sistema permite ver las canciones más populares de un país.
RF19	El sistema permite ver las canciones más populares de un artista.
RF20	El sistema permitirá usar un equalizador al escuchar las canciones.
RF21	El sistema permitirá el uso de *banners* para ofrecer publicidad.

Se exponen los siguientes requisitos no funcionales de la aplicación:

Requisito no funcional	Descripción
RNF1	El sistema permitirá ser utilizar un diseño modular, un lenguaje fácil de entender, usar y mantener.
RNF2	El cliente tendrá el desarrollo móvil en formato *Android* e *iOS*.
RNF3	El sistema permite almacenar canciones y podcasts en formato *.mp3*, *.WAV* y *.OGG*.

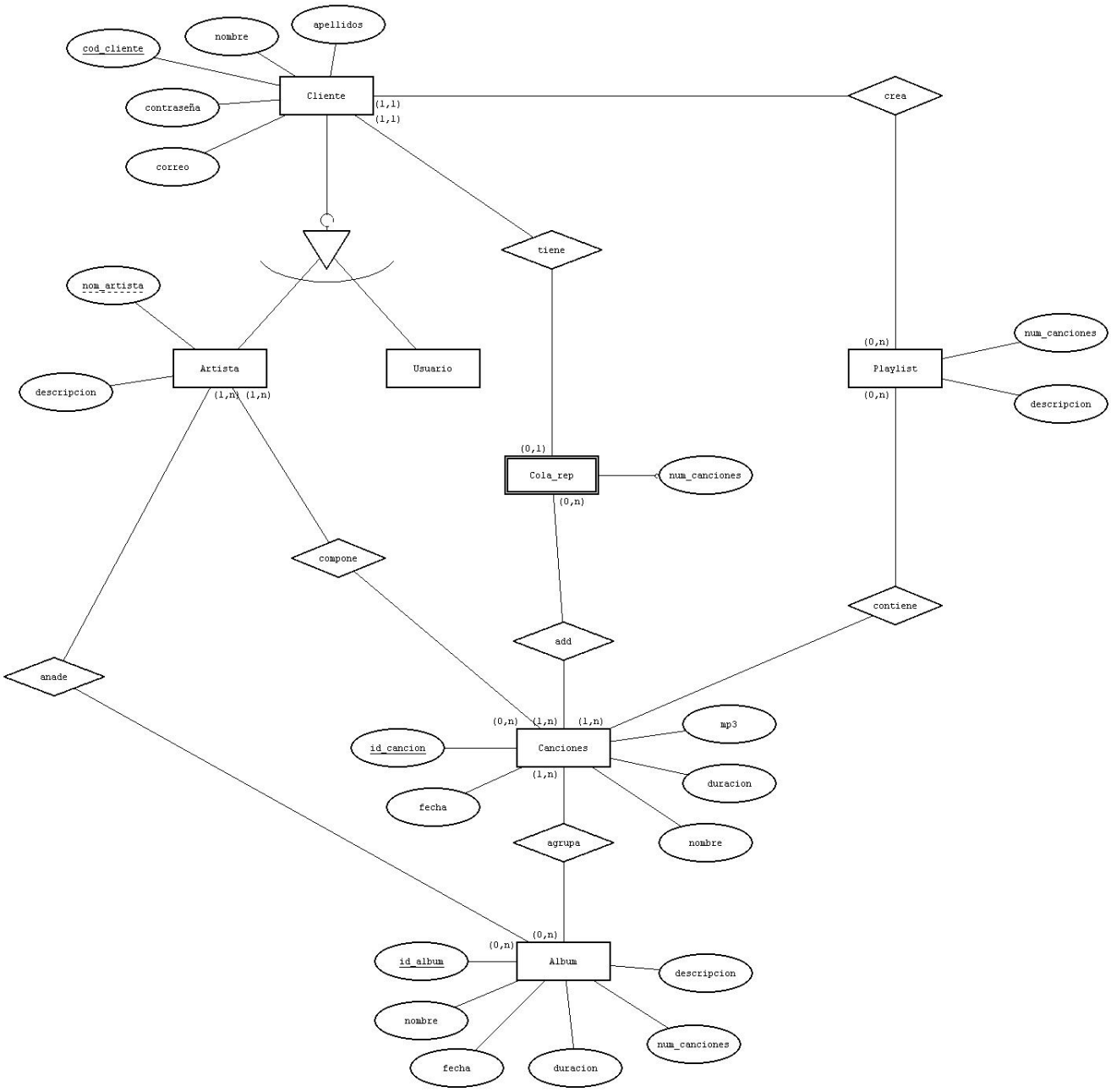
3.2. Diseño del sistema

Un aspecto esencial que se ha considerado desde el principio ha sido la realización de la base de datos relacional en PostgreSQL, siguiendo el siguiente esquema:



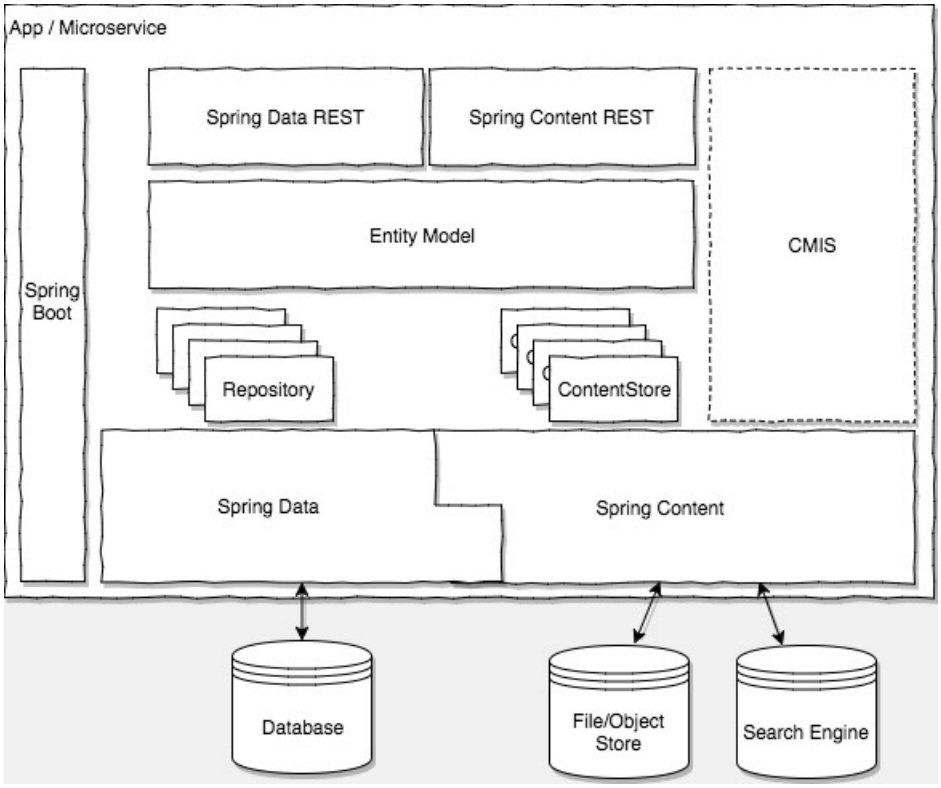
Fue elegido este gestor de datos porque, en comparación con los demás, permite de una manera simple almacenar los datos de las canciones, unido a una alta fiabilidad.

El modelo *Entidad - Relación* que se ha llevado a cabo ha sido el planteado a continuación, implementado en el back-end con esta estructura:

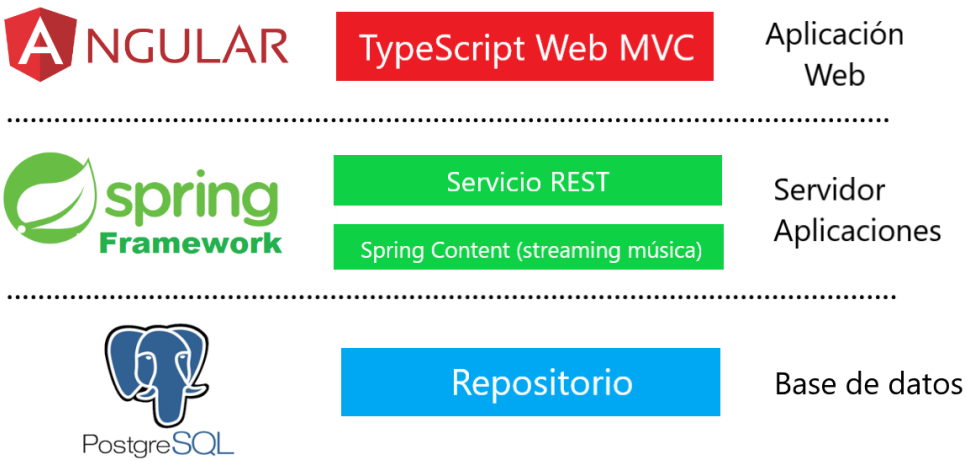


El diseño del back-end será *REST* para todas las consultas y peticiones relacionadas con la aplicación, excepto la parte de las canciones en streaming. Para ello, se usará la siguiente tecnología CRUD: *Spring Content(Cloud-Native Content Services for Spring)*.

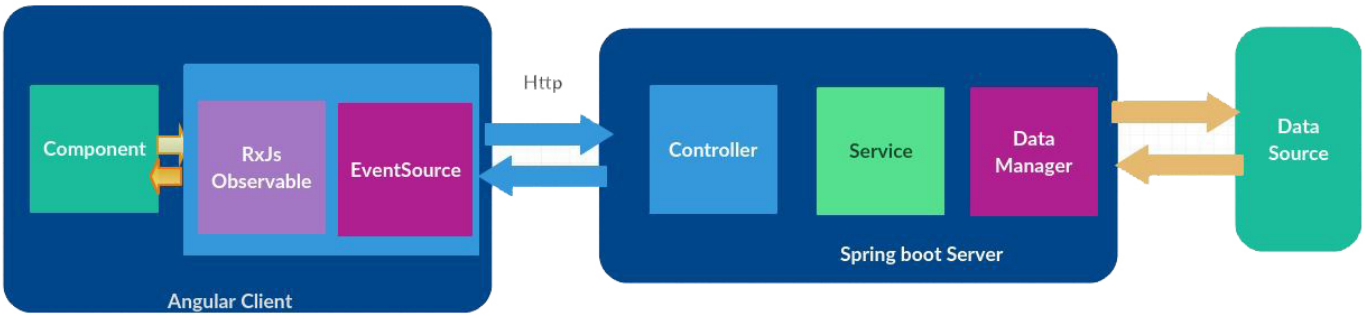
Esto es debido a que no hay que preocuparse de la implementación del código del controlador del reproductor para que soporte *request* parciales (rangos de bits).
Spring Content nos permite crear un servicio *REST-based mp3* que soporta streaming. Además, también aporta completamente la funcionalidad *CRUD* (*Create == POST*, *Read == GET* (incluyendo rangos de bits), *Update == PUT*, *Delete == DELETE*) en caso de ser necesaria.



Por todo esto, la estructura del proyecto con la parte front-end incluida será lo siguiente:



A la izquierda se denotan las tecnologías usadas para cada parte junto a la función que realiza cada una.
La manera de comunicarse entre APIs sigue este patrón:



Event Source between Angular And Java