Plan de gestión, análisis, diseño y memoria del proyecto

Proyecto software

Álvaro García Díaz, Óscar Baselga Lahoz, Alberto Calvo Rubió,
Saúl Flores Benavente, Luis García Garcés, Alejandro Facorro Loscos,
Germán Garcés Latre

Universidad de Zaragoza

16 de marzo de 2020

Índice

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 sus 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:
Backend:

- Saúl Flores Benavente: Servidor y API
- Alberto Calvo Rubió: Base de Datos, despliegue (Docker y Heroku) y apoyo al servidor

Frontend Android:

- Luis García Garcés: aplicación Android
- Alejandro Facorro Loscos: aplicación Android
- Óscar Baselga Lahoz: aplicación Android

Frontend Web:

- Álvaro García Díaz: aplicación Web
- Germán Garcés Latre: aplicación Web

Para gestionar el proyecto, se han establecido diferentes roles:

- Director de Proyecto: Alberto Calvo
- Líder Backend: Saúl Flores
- Líder Frontend Android: Luis García
- Líder Frontend Web: Álvaro García

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 y 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

La aplicación será compatible en Google Chrome y Firefox en cuanto a interfaz web, ambos navegadores utilizarán la última versión, 80.0.3987.132 para Google Chrome y 74.0 para Firefox las cuales están disponibles en Windows y Mac, en cuanto a la aplicación móvil, será compatible para

dispositivos android que tengan como mínimo la versión 9. Las pruebas de nuestra aplicación se harán utilizando un navegador web o un APK que se comunicarán con el servidor web.

Para almacenar y probar la información referente a la aplicación se utilizará contenedores tanto para la base de datos como el servidor web. Dichos contenedores se desplegarán en Heroku siendo este un servicio de computación en la nube, Heroku brinda 10.000 filas para la base de datos, un proceso para la ejecución y 500MB de RAM; en cuanto al tamaño del contenedor no existe ninguna limitación a cambio de solo poder utilizarlo durante un cierto periodo de tiempo.

La formación inicial de los miembros del equipo consistirá en revisar tutoriales escritos o por video de los lenguajes de programación establecidos para cada uno de los componentes del proyecto, además de descargar los IDE y los frameworks correspondientes y compatibles con estos, por ejemplo, Flutter para Android Studio en cuanto a Front-end móvil y Angular para Visual Studio Code para Front-end web .

3.1.2. Procesos de ejecución y control del proyecto

Las comunicaciones internas se llevarán a cabo mediante distintos medios:

- Grupo de Whastapp: 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.
- Github issues: en la plataforma de github se crearán issues por los líderes de cada grupo que se asignan a los diferentes miembros, quedando constancia del progreso mediante comentarios y también reflejando el estado mediante el apartado de projects. De esta forma se podrá observar el progreso del proyecto respecto a medidas como issues abiertos frente a cerrados.
- 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 en todas las reuniones formales (ya sean de grupo completo o con el profesor/cliente) en los que quedan reflejados: participantes, ausencias, duración, acciones tomadas, objetivos para la siguiente reunión y fecha de la siguiente 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 serán lo issues creados en cada repositorio. Mediante estos se debe reflejar todas las tareas para cumplir un objetivo. De forma que contando los issues

cerrados respecto al total de issues se pueda medir el número de pasos que quedan para finalizar el proyecto (hay que tener en cuenta de forma subjetiva la dificultad de cada uno de ellos). Estos progresos serán medidos por el líder de cada repositorio/grupo (en caso de la documentación será el Director de Proyecto). Y finalmente, estos se reportarán al Director para obtener una visión global del progreso.

3.1.3. Procesos técnicos

Las herramientas que se utilizarán en este proyecto serán Android Studio utilizando el framework Flutter que da soporte al lenguaje Dart, Visual Studio Code que utiliza el framework Angular junto a los lenguajes de programación HTML, Typescript y CSS y por último, Pycharm que usa el lenguaje de programación Python, todo el proyecto será integrado en Git como herramienta para el control de versiones dentro de la plataforma Github.

Las pruebas de la aplicación durante el desarrollo se realizarán de manera local en los frameworks correspondientes de cada equipo. Para probar la aplicación de forma conjunta se utilizará tanto los navegadores web y dispositivos móviles compatibles y dichos dispositivos se comunicarán con el servidor web que estará desplegado en Heroku

Para desplegar el servidor flask se necesita copiar los archivos del proyecto en el equipo/cloud/contenedor correspondiente. Una vez están los archivos necesarios, se deben instalar los paquetes y librerías necesarias, en este caso, especificadas en un fichero requisitos.txt que se usará para este propósito mediante el comando pip: pip -r requisitos.txt. Adicionalmente se necesitan librerías de postgresql para poder usar el paquete de acceso a esta base de datos, lo cual no es un problema si está instalado en el mismo equipo. Es necesario que la base de datos esté funcionando con los parámetros establecidos.

Para instalar la interfaz web, es preciso copiar el proyecto en el equipo/contenedor/servidor correspondiente y ejecutar “npm install” para instalar las dependencias necesarias para su ejecución. A continuación, para hacer accesible la web, hay que ejecutar “ng serve” y entonces se puede acceder a la web mediante los navegadores web introduciendo la URL correspondiente (para pruebas sería localhost:4200). Para que funcione correctamente, es necesario que la base de datos esté lanzada para las peticiones de canciones y que la API de podcast funcione (al ser externa al proyecto no se puede manejar su estado). El proyecto Angular se puede abrir mediante VIsual Studio Code gracias a una serie de plugins o extensiones instaladas [4], después se pueden ejecutar los comandos anteriores por medio de dichas extensiones.

Para utilizar la aplicación en el móvil se ha de instalar un archivo .apk . Este archivo se genera desde la terminal de Android Studio. Ejecutamos “flutter build apk” en el directorio de nuestro proyecto. Tras ejecutar esto se habrá creado un ejecutable .apk que copiaremos en nuestros dispositivos e instalaremos. 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.

El despliegue del sistema se realizará en Heroku mediante contenedores. Se deberá ejecutar el docker-compose correspondiente que mediante Dockerfile creará las imágenes y resolverá las dependencias necesarias indicadas en los ficheros de cada servidor.

3.2. Planes

3.2.1. Plan de gestión de configuraciones

Para el código que se desarrolla en el proyecto se ha 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, 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.

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 acompañe al proyecto también seguirá este estándar. Además, para la escritura de código se seguirá el estándar recomendado por Google en el caso de Android y Backend y el recomendado por la W3Schools para Frontend.

Como el sistema Git soporta la gestión de incidencias, se va a aprovechar este sistema para gestionar todas las situaciones inesperadas que pudieran suceder. Cuando se haya completado una incidencia la persona que la haya terminado se pondrá en contacto con el coordinador y este se encargará de ponerla como terminada.

En lo relacionado al código, las incidencias serán añadidas al repositorio por lo administradores y se corresponderán al trabajo relacionado con el proyecto que se desarrollará esa semana. 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, se lo comunicará al coordinador para que este cambie 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 mínimos. 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 compañeros están llevando a cabo.

En lo relacionado a la documentación el control de versiones 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.

Aunque cada apartado cuente con un coordinador, no hace falta que esté de permisos al resto de los miembros del grupo para hacer commit, lo que ha de hacer es informar al resto de que ha realizado ese commit, pero no necesita solicitar autorización.

3.2.2. Plan de construcción y despliegue del software

El equipo de frontend web trabaja con un proyecto Angular en Visual Studio Code con la ayuda de una serie de extensiones[1] que facilitan el trabajo. Todo el proyecto se encuentra en un repositorio de Github, permitiendo que todos tengan el mismo código. Las dependencias necesarias se pueden instalar simplemente ejecutando "npm install" en el proyecto Angular.

Los tests automáticos propuestos para la parte de frontend (web y móvil) son las siguientes: valores nulos, vacíos o con muchos caracteres (más de un millón) en los campos de registro y configuración, un correo electrónico ya presente en la base de datos y una petición inválida al solicitar una canción o podcast. El equipo de backend trabaja con un proyecto flask en Pycharm usando un entorno virtual para desarrollar. Las dependencias se anotan en un fichero requirements.txt que se actualiza en el repositorio y cada vez que se modifica los miembros instalan los paquetes con la misma versión, asegurando que todos los miembros trabajan en el mismo entorno.

Para las pruebas se va a preparar un fichero que compruebe que el servidor funciona correctamente respondiendo a las peticiones. Para cada petición que puede realizar el frontend, se añaden elementos a la base de datos que permitan recrear la situación, se comprueba que funciona y luego se borran. En un principio se ha propuesto una construcción conjunta de todo el sistema y ejecución de todas las pruebas disponibles cada 2 semanas. Conforme se vaya avanzando en el calendario, se prevé realizarlo semanalmente. Todo ello se realizará en Heroku mediante la herramienta docker-compose desplegando cada servidor y la base de datos en un contenedor distinto, permitiendo una gran movilidad, por si fuese necesario cambiar de plataforma.

3.2.3. Plan de aseguramiento de la calidad

La aplicación a realizar es un reproductor de música, y 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 y sea más fácil identificar dónde se encuentra cada función disponible.

Una de las guías de estilo más empleada es la de Spotify[2] 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[3]. 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, negro o que no esté en solo dos tonos diferentes (presente en todas las pantallas, destacando la de iniciar sesión y registro)

[1] `https://marketplace.visualstudio.com/items?itemName=johnpapa.angular-essentials`
[2] `https://gcase.files.wordpress.com/2015/10/spotify-developer-brand-standards-guide-02jr.pdf`
[3] `https://developer.spotify.com/branding-guidelines/`

- El logo sólo puede encontrarse de tres maneras: el logo con el nombre de la aplicación a la derecha (barra superior) o con el nombre de la aplicación abajo (inicio sesión y registro) o solo el logo (imagen de la aplicación móvil o icono de la pestaña en interfaz web)
- Los colores predominantes serán el blanco, negro, los colores del logo (azul, morado) y una escala de grises (presente en todas las pantallas)

Otra guia de estilo consultada ha sido la de Shazam[4], 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 Sans y podrá presentarse en negrita (pantalla de canción en el nombre de la canción)

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.

Para las diferentes partes del proyecto, se van a utilizar programas o entornos de programación que ayudan a asegurarse de la calidad del código:

- Angular: Este framework va a ser usado en el editor Visual Studio Code con la instalación de una serie de plugins para trabajar con Angular. Como se requiere trabajar en grupo, se ha instalado otra extensión para facilitar la sincronización con el repositorio de Github, GitLens[5].
- Flutter: Para trabajar con Flutter, se ha empleado el entorno de desarrollo Android Studio. Ha sido necesario la instalación de una extensión para que fuera posible la compatibilidad con el código Dart.

[4] `https://issuu.com/duncanriley/docs/01_sz_brand_guidelines_lite`
[5] `https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens`

- Flask: El entorno de desarrollo escogido ha sido PyCharm al desarrollarse la parte de backend en el lenguaje de programación Python. Para interactuar con la base de datos, se ha empleado SQLAlchemy, que facilita la creación de clases y relaciones en Python, evitando utilizar SQL. En cuanto a la base de datos, se ha decidido utilizar PostgreSQL, que se trata de una base de datos relacional orientada a objetos.

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 del personal no implicado de esa parte, ya que al no saber los límites del sistema y no estar condicionados, pueden probar una mayor cantidad de posibilidades.

3.2.4. Calendario del proyecto y división del trabajo

PROJECT NAME	PROJECT MANAGER	START DATE	END DATE	OVERALL PROGRESS
TuneIt	Alberto Calvo	10-feb.	29-may.	11%

TASKS	RESPONSIBLE	START	END	DAYS	STATUS
Desarrollar reproductor móvil	Equipo frontend móvil	3/3	3/15	12	Complete
Desarrollar reproductor web	Equipo frontend web	3/3	3/15	12	Complete
Desarrollar listas por defecto móvil	Equipo frontend móvil	3/3	3/15	12	Complete
Desarrollar listas por defecto web	Equipo frontend web	3/3	3/15	12	Complete
Desarrollar buscador canciones móvil	Equipo frontend móvil	3/15	3/20	5	In Progress
Desarrollar buscador canciones web	Equipo frontend web	3/15	3/20	5	In Progress
Coordinar varias sesiones del usuario móvil	Equipo frontend móvil	3/21	3/25	4	Not Started
Coordinar varias sesiones del usuario web	Equipo frontend web	3/21	3/25	4	Not Started
Implementar creación listas móvil	Equipo frontend móvil	3/26	3/31	5	Not Started
Implementar creación listas web	Equipo frontend web	3/26	3/31	5	Not Started
Diseñar conexión Python con la BD	Equipo backend	3/3	3/8	5	Complete
Fichero Flask-SQLAlchemy para BD	Equipo backend	3/7	3/13	6	In Progress
Tratar peticiones del reproductor	Saúl Flores	3/11	3/15	4	In Progress
Tratar peticiones compartir canciones	Saúl Flores	3/15	3/18	3	Not Started
Tratar peticiones de usuarios	Saúl Flores	3/18	3/21	3	Not Started
Pruebas servidor	Saúl Flores	3/5	3/20	15	In Progress
Montar BD	Alberto Calvo	3/4	3/18	14	In Progress
Desplegar BD	Alberto Calvo	3/17	3/20	3	Not Started
Claves en las tablas e incluir img. en BD	Alberto Calvo	3/19	3/21	2	Not Started
Actualizar model E/R de la BD	Alberto Calvo	3/5	3/25	20	In Progress
Pruebas BD	Alberto Calvo	3/5	3/25	20	In Progress
LAUNCH		2/10	5/29	109	

Figura 1: Diagrama de Gant

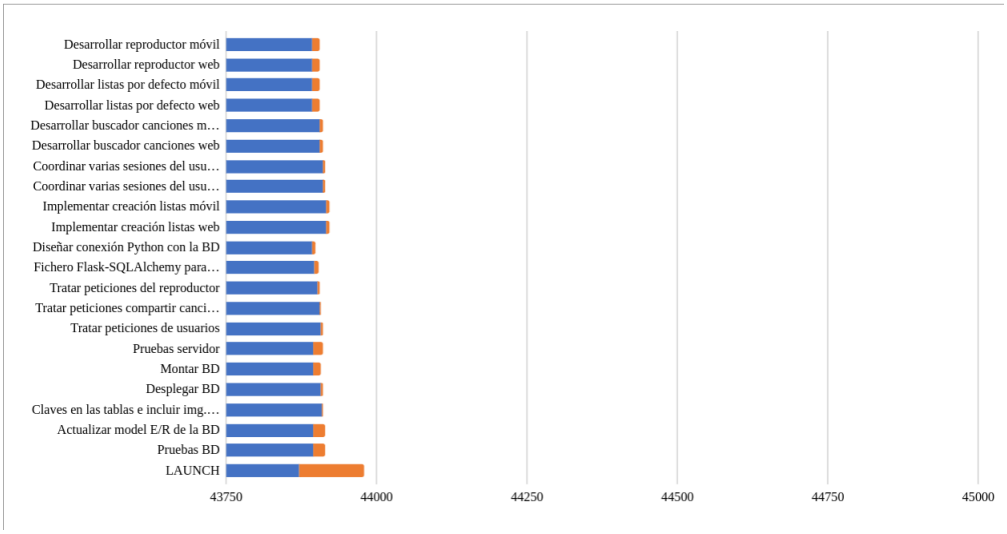

Figura 2: Gráfico

4. Análisis y diseño del sistema

4.1. Análisis de requisitos

4.1.1. Sistema: Requisitos no funcionales

1. El sistema tendrá una versión para Android y otra para Web
2. El sistema soportará la versión de Android 6.0 y de la Web la última versión de los navegadores Chrome y Firefox
3. El sistema se conectará a un servidor para obtener las canciones/podcasts
4. El sistema necesita conexión a Internet

4.1.2. Backend: Requisitos funcionales

1. Una canción/podcast se compone de un audio, un título, un artista y un álbum
2. 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
3. Un usuario está conformado por un nombre de usuario (nick), un correo electrónico único (es decir, no puede haber dos usuarios con el mismo correo), una contraseña para acceder, una fecha de nacimiento y su país
4. Los usuarios se pueden registrar en el sistema

5. Los usuarios pueden agregar a otros usuarios como amigos
6. Los usuarios se pueden identificar mediante el correo electrónico y la contraseña
7. Los usuarios pueden crear o eliminar listas de reproducción formadas por canciones
8. Los usuarios pueden agregar a sus listas de reproducción otras listas creadas por otros usuarios
9. Los usuarios pueden compartir canciones/podcasts o listas de reproducción con otro usuario amigo
10. Los usuarios pueden añadir o eliminar una canción de una lista de reproducción propia previamente creada
11. Los usuarios pueden ordenar las canciones presentes en una lista de reproducción propia
12. Los usuarios poseen una lista de canciones favoritas
13. Los usuarios pueden añadir o eliminar canciones de favoritos
14. Los usuarios pueden ordenar las canciones presentes en favoritos
15. Los usuarios pueden obtener las canciones de una lista de reproducción
16. Los usuarios pueden modificar su nombre de usuario, su correo electrónico y su contraseña
17. Los usuarios se pueden suscribir a los artistas
18. Los usuarios reciben notificaciones de canciones nuevas lanzadas por artistas suscritos
19. Los usuarios pueden buscar canciones mediante palabras clave con un mínimo de una palabra y una longitud mayor o igual a 3 caracteres
20. Se guarda el tiempo en el que se ha pausado o cerrado la aplicación de la última canción
21. Se guarda la última lista reproducida

4.1.3. Backend: Requisitos no funcionales

1. El nick del usuario debe tener entre 3 y 50 caracteres
2. Las contraseñas del usuario se encuentran cifradas

4.1.4. Frontend: Requisitos funcionales

1. La reproducción de canciones/podcast se sincronizará entre dispositivos de un mismo usuario
2. Se recupera el tiempo de la última canción escuchada cuando se abre el sistema
3. Se recupera la última lista de reproducción cuando se abre el sistema
4. El sistema permite a los usuarios registrados iniciar sesión

5. El sistema permite crear listas de reproducción con un nombre
6. El sistema permite organizar las listas de reproducción propias y la de favoritos
7. El sistema permite añadir y eliminar canciones de las listas de reproducción propias previamente creadas y en favoritos
8. El sistema permite consultar los usuarios amigos
9. El sistema permite agregar a otros usuarios como amigos
10. El sistema permite eliminar a otros usuarios amigos
11. El sistema permite consultar las peticiones de amistad recibidas
12. El sistema permite buscar a otros usuarios
13. El sistema permite a los usuarios modificar su información (correo electrónico, nick y contraseña)
14. El sistema permite al usuario eliminar su cuenta
15. El sistema permite recibir notificaciones cuando un artista suscrito publica una nueva canción
16. El sistema permite que los usuarios compartan canciones o listas de reproducción con otros usuarios amigos
17. El sistema permite buscar información externa asociada a una canción/podcast mediante Google

4.1.5. Frontend: Requisitos no funcionales

1. En la pantalla principal, se muestran las últimas canciones añadidas al sistema
2. El sistema posee un menú desplegable accesible desde cualquier pantalla, que contiene un buscador, la pantalla principal, el perfil del usuario, los podcasts, las listas de reproducción, el ecualizador y los amigos
3. En la pantalla de perfil, aparece el nick del usuario y sus listas de reproducción
4. En la pantalla de amigos, se muestra una lista de los usuarios que han aceptado una solicitud de amistad. También permite consultar las solicitudes pendientes de aceptar y buscar otros usuarios para agregarlos como amigos
5. El sistema posee una barra inferior con la canción que se está reproduciendo o la última escuchada y permite gestionarla (pausar o iniciar, avanzar a la siguiente canción, retroceder a la anterior canción, añadir a favoritos, añadir a una lista de reproducción y controlar el volumen)
6. El sistema posee una pantalla para cada canción donde se muestra la imagen del álbum al que pertenece, su nombre, el artista y los controles para gestionar la canción

4.2. Diseño del sistema

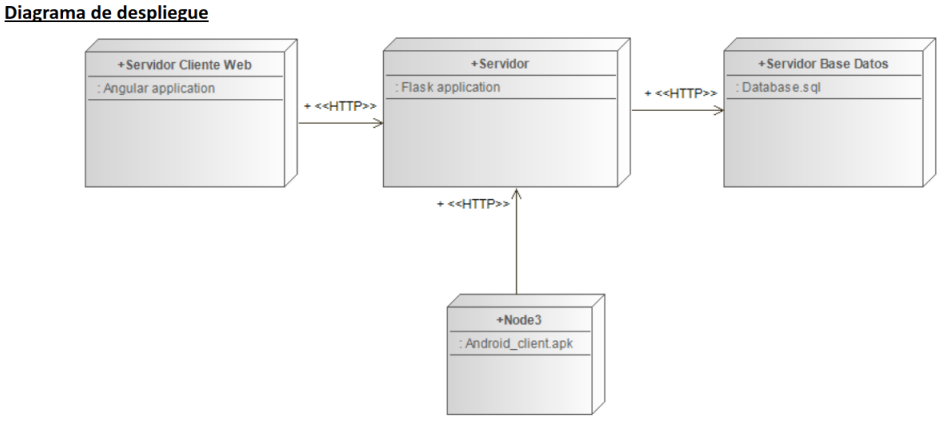

Figura 3: Diagrama de despliegue

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 escrito 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 es 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.

El resultado final será una aplicación web y una aplicación móvil (Android), para web se ha decidido usar tecnologías web para facilitar todo el proceso de implementación de los requisitos funcionales. Para la web se ha decidido usar Angular, un framework para aplicaciones web desarrollado en TypeScript mientras que para móvil se ha usado Flutter, un SDK para la generación de código nativo. 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 no SQL frente a las SQL.

[6] `https://www.listennotes.com/es/api/`

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 mas pesados, se almacenarán en el servidor, 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 seguridad intentara proteger los datos que se poseen de los usuarios, cifrando las contraseñas que se almacenan en la base de datos.

Las instalaciones y despliegue se realizaran en heroku(https://www.heroku.com), donde se alojará el servidor web y la base de datos, a los que se accede desde los navegadores de los equipos de los componentes del equipo. Para la aplicación Android, se simulara el cliente en los equipos de los desarrolladores mediante las herramientas que ofrece Android studio y puede que en alguna ocasión se instale en algún dispositivo.

5. Memoria del proyecto

ESTE CAPÍTULO NO SE RELLENA EN LA PRIMERA ENTREGA

5.1. Inicio del proyecto

5.2. Ejecución y control del proyecto

5.3. Cierre del proyecto

6. Conclusiones

7. Anexo I. Glosario

8. Anexo II. Actas de todas las reuniones realizadas

9. Anexo III..Otros anexos que se consideren necesarios