Grupo 02: Mary Allen Wikes

Proyecto PlayStack

Plan de gestión, análisis, diseño y memoria del proyecto

Enrique Ruiz Flores	766685@unizar.es
Daniel Subías Sarrato	759533@unizar.es
Alberto Martínez Rodriguez	764900@unizar.es
Fernando Peña Bes	756012@unizar.es
Pedro José Pérez García	756642@unizar.es
Alba Vallés Esteban	760099@unizar.es
Andrés Otero García	757755@unizar.es

Fecha de presentación de la propuesta:
16 de Marzo de 2020

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

1. Introducción

En este proyecto se va a abordar la creación de un reproductor multiplataforma de música y podcasts con un gran potencial, puesto que no solo permitirá a los usuarios realizar las acciones típicas de cualquier otro reproductor de música, como puede ser la creación de listas de reproducción personalizadas, confeccionar una lista de canciones favoritas o reproducir canciones a voluntad,sino que además destaca por ofrecer numerosas posibilidades muy atractivas como por ejemplo la incorporación de diferentes aspectos sociales, y mucho más, todo ello desde cualquier dispositivo. Y es que el propósito es crear una aplicación que el usuario pueda utilizar en absolutamente cualquier plataforma con acceso a internet, ya sea el móvil, el ordenador o una tableta y de esa manera permitir llevarse su música y otro contenido a donde quiera que vaya.

Se busca entre otras cosas que la aplicación sirva como mejor alternativa al reproductor por defecto de cualquier dispositivo móvil, ya sea Android o iPhone. Para ello, esta aplicación ofrecerá tanto las opciones básicas de cualquier reproductor de música integrado por defecto, como la reproducción de canciones almacenadas en el dispositivo o confección de listas de reproducción con éstas,como otras cuantas funcionalidades de las que suelen carecer muchos de estos reproductores por defecto como por ejemplo la reproducción de música en streaming, sin necesidad de tener el contenido almacenado en el dispositivo del usuario. Además, para lograr un mayor alcance se crearán también una versión de escritorio para sistema operativo Windows por ser este el más extendido y una versión web permitiendo escuchar las listas de reproducción previamente creadas, crear nuevas listas y todo lo demás que ofrece la aplicación desde cualquier navegador web o desde la aplicación para escritorio.

Se pretende alcanzar la máxima interoperabilidad entre dispositivos llegando a permitir continuar la reproducción que se inició en un dispositivo en otro diferente a voluntad además de ver reflejados cambios que se hagan desde cualquier sitio en todos los demás lugares donde se acceda con la misma cuenta, permitiendo ,por ejemplo, crear una lista de reproducción desde la aplicación móvil y ver la lista en la aplicación de escritorio para poder reproducirla.

Tras el desarrollo del proyecto se entregarán tanto el código en lenguaje *Dart* para la compilación e instalación de las aplicaciones Android e iOS, como el correspondiente ejecutable `.exe` para Windows y archivo `.war` para el despliegue de la aplicación en un servidor web.

Más adelante en este mismo documento se incluyen la descripción de como se ha organizado el proyecto, cuál ha sido el plan de gestión del proyecto detallando tanto los procesos como los planes seguidos y por último también se incluye una sección dedicada al análisis y diseño del sistema.

2. Organización del proyecto

Antes de detallar cómo se ha organizado el proyecto se describirá el equipo del proyecto. El equipo tiene siete integrantes, y para una mejor distribución del trabajo se han dividido en tres grupos principales:

- El equipo encargado de la **capa de acceso a datos** (gestión del backend) que se encargará de la creación y despliegue en un servidor de la base de datos que emplearán todas las aplicaciones, tanto las móviles como la web y la de escritorio para su correcto funcionamiento. Los integrantes de este grupo y sus roles son los siguientes:
  - Daniel Subías Sarrato: administrador de la base de datos, responsable de diseñar, implementar, mejorar y mantener el sistema de base de datos
  - Alberto Martínez Rodriguez: desarrollador de backend responsable de la lógica del negocio y de crear las APIs para que el Frontend pueda hacer uso de ellas .
  - Enrique Ruiz Flores: Director del proyecto y administrador del servidor, responsable de gestionar la instalación, soporte y mantenimiento de el servidor en donde se aloja la bases de datos y la app, además de organizar y supervisar el progreso del equipo entero.
- El equipo encargado de la creación de la **aplicación de escritorio y la aplicación web**. Los integrantes de este grupo son:
  - Pedro José Pérez: encargado de la producción, modificación y mantenimiento de la la interfaz de la página web y la aplicación de escritorio para Windows.
  - Fernando Peña Bes: junto con el otro integrante de este equipo también será encargado de la producción, modificación y mantenimiento de la la interfaz de la página web y la aplicación de escritorio para Windows.
  - Alba Vallés Esteban: también encargada del desarrollo y gestión de la aplicación de escritorio.
- El equipo encargado de la creación de las dos **aplicaciones móviles** cuyos integrantes son:
  - Andrés Otero García: encargado del desarrollo, modificación y mantenimiento de las aplicaciones móviles Android e iOS.
  - Enrique Ruiz: también colaborará en el desarrollo y mantenimiento de las aplicaciones móviles.

3. Plan de gestión del proyecto

3.1 Procesos

3.1.1 Procesos de inicio del proyecto

Se va a utilizar el servicio de almacenamiento en Cloud de Azure para desplegar la aplicación, aprovechando la bonificación de 100 euros por registro que ofrecen a los estudiantes. Para realizar pruebas en dispositivos móviles se emplearán diferentes simuladores de Android Studio y móviles iPhone de los componentes del grupo.

La primera semana, tras decidir las tecnologías y acordar los requisitos con el cliente, se dedicará a la formación en cada tecnología y lenguaje necesarios para el desarrollo de las distintas partes que componen el proyecto. Visto que ningún miembro que compone el equipo de creación de la interfaz móvil tiene experiencia en el uso de la tecnología que se va a usar para tal fin, Flutter en este caso, se consultará la documentación de la misma y se realizarán tutoriales online que los propios desarrolladores ponen a disposición del público. De igual forma, las personas encargadas de la versión de escritorio consultarán la información correspondiente de Ionic y realizarán los cursos que se ofertan en la página oficial.

En lo que respecta al desarrollo del Back-End, los miembros del equipo consultarán la documentación del framework Django y realizarán tutoriales online visuales para comprender el funcionamiento de esta tecnología. No será necesaria la formación en la creación de bases de datos relacionales y lenguaje SQL pues todos los miembros participantes en el Back-End tienen experiencia en este campo. Sin embargo a pesar de dominar el diseño de bases de datos, se deberá aprender a desplegar una base de datos en la plataforma cloud Azure ya mencionada, consultando la documentación proporcionada por Microsoft y aprovechar las facilidades que esta plataforma ofrece.

Equipo BackEnd: Alberto Martinez Rodriguez, Daniel Subías Sarrato y Enrique Ruiz Flores

- Almacenamiento en la nube con Azure (PaaS)
- Gestor de la Base de Datos: PostgreSQL
- Django (Framework Python)

Equipo FrontEnd: Fernando Peña Bes, Pedro José Pérez García, Andrés Otero García , Alba Vallés Esteban y y Enrique Ruiz Flores

- JavaScript (tutorial [17]) (lenguaje más utilizado en tecnologías FrontEnd).
- Ionic (tutoriales[16][11] y curso de 7 días [12]) para la aplicación de escritorio.
- Flutter (tutoriales [18]) para dispositivos móviles.
- Android Studio para la interfaz de usuario en móvil.

Modelio para la realización de diagramas en la fase de Análisis y Diseño.

3.1.2 Procesos de ejecución y control del proyecto

Se van a llevar a cabo reuniones semanales todos los miércoles en horario de 12:00 a 14:00 para revisar el trabajo elaborado hasta el momento y tomar nuevas decisiones sobre el trabajo futuro a realizar.

Se ha creado un grupo en la aplicación WhatsApp para la comunicación interna relativa a la organización de reuniones, cuestiones generales del proyecto y detalles que requieran respuestas instantáneas y consenso de todos los miembros.

NOTA: Con los recientes eventos relacionados con el coronavirus COVID-19, las recomendaciones del Ministerio de Sanidad del Gobierno de España, así como las medidas preventivas para frenar la expansión del virus, se realizarán reuniones en un horario más flexible, y se utilizarán distintas plataformas para la comunicación mediante videoconferencia, llamada de voz y mensajes de texto, como Discord, Skype, Google Meet o Whatsapp. Ya se dispone de un servidor en la aplicación Discord para poder coordinar el grupo ante la situación de emergencia sanitaria.

Las decisiones importantes así como la redacción de las actas se escribirán en un borrador preliminar en un documento Word, para posteriormente ser redactadas correctamente en este documento.

Las tareas a realizar progresivamente se decidirán en las reuniones semanales y se repartirán utilizando GitHub para mantener el registro de las mismas. El líder será el responsable de repartir dichas tareas teniendo en cuenta las características técnicas de cada miembro.

La gestión del equipo humano y las disputas que surjan se abordarán con una votación entre todos los miembros del grupo. La elección o posición que obtenga la mayoría será la que se lleve a cabo.

Se monitorizará el estado de cada tarea para medir el progreso del proyecto cada semana en las reuniones internas. Si se detecta algún problema de rendimiento se realizarán reuniones adicionales para asegurar el trabajo continuo de cada componente del grupo y el avance del proyecto. Si esta medida no fuese suficiente, se consensuaría una modificación de los requisitos con el cliente.

Los resultados obtenidos a lo largo del desarrollo del proyecto se le entregarán al cliente a través de las entregas planificadas en la plataforma Moodle2, intentando cumplir en todo momento las fechas de entrega establecidas.

3.1.3 Procesos técnicos

Para el desarrollo del software que constituirá el producto final a implementar se utilizarán distintos IDEs para generar el código con el objetivo de facilitar esta labor. La creación de la aplicación móvil se hará con el entorno AndroidStudio, pues este proporciona un simulador móvil para poder realizar pruebas y comprobar el correcto funcionamiento del software. Una vez probada la aplicación de forma simulada, esta se ejecutará en los móviles de los integrantes del equipo para comprobar que puede correr de forma adecuada en los dispositivos de los usuarios finales.

En el desarrollo del software para la versión de escritorio de la aplicación, se pretende utilizar Angular como framework, de este modo se usará tecnología web para la creación del interfaz gráfico. Con el objetivo de hacer mas amena la implementación se utilizará el SDK Ionic para la edición del código TypeScript de Angular. Una vez terminada la versión de escritorio se harán pruebas manualmente en los ordenadores del equipo, comprobando que el funcionamiento de la aplicación es el esperado. Cuando tanto los interfaces gráficas de móvil y escritorio estén terminadas se procederá al acoplamiento de estas con el nivel de modelo lógico realizar pruebas de conexión y funcionamiento.

En lo que respecta al Back-End, el modelo de la aplicación va a ser implementado en el lenguaje de programación Python debido a que al usar Django como framework, este está escrito en dicho lenguaje, por esta razón se usará el IDE Pycharm para la edición del código. Django también ofrece la posibilidad de crear las tablas SQL en Python por lo que se aprovechará esta funcionalidad para la generación del código de creación de la base de datos, la cual será posteriormente desplegada en el servidor de Azure adquirido configurando en él el gestor Postgre para poder tener permisos de acceso. Una vez creada la base de datos se poblará con datos falsos (los cuales se eliminarán antes de presentar la versión final) para poder realizar consultas de prueba desde las maquinas del equipo de desarrollo y comprobar que el despliegue se ha realizado con éxito. Disponiendo ya de la capa del nivel de datos del sistema, se procederá a la implementación del modelo lógico mediante Pycharm diseñando un conjunto de pruebas para comprobar las correcta conexión con la base de datos, todo esto desde uno de los ordenadores del equipo de Back-End.

Todos los integrantes del grupo, tanto de Front-end como Back-end, subirán a GitHub el código que creen o modifiquen para que el resto del equipo pueda ver las actualizaciones realizadas y nadie quede desinformado de los cambios intentando así trabajar de la manera mas síncrona posible. Tanto en el análisis como en el diseño del software, los diagramas que reflejan el comportamiento de este se harán utilizando la herramienta CASE Modelio pues todo el equipo tiene una dilatada experiencia con el entorno.

3.2 Planes

3.2.1 Plan de gestión de configuraciones

En la parte correspondiente al Back-End, se utilizará la Guía de Estilo para código Python "PEP8"(Python Enhancement Proposal) [10].

Si bien es cierto que la aplicación web va a ser escrita en TypeScript, no existe un estándar generalizado para este lenguaje de programación, por lo que se seguirá la guía de estilo de JavaScript en el estándar de Google [8].

Finalmente, en el caso de la aplicación móvil, se va a seguir la guía de estilo recomendada por los desarrolladores de Dart para escribir código en su lenguaje [9].

En cuanto a los responsables de las distintas tareas del proyecto, Enrique, al ser el director del proyecto, se encarga de las tareas de puesta en marcha y apoyo al equipo. En cambio las tareas de revisión de commits, copias de seguridad, y control de versiones, serán ejecutadas por los diferentes miembros de los equipos de backend y frontend, siendo Enrique el máximo responsable y el coordinador.

Para hacer un control de versiones correcto, se ha creado una organización en GitHub que incluye a todos los miembros del proyecto. Con ésta, se han creado tres repositorios destinados a la Documentación, el Front-end y el Back-end.

Dentro del repositorio de Documentación, se creará una carpeta que contendrá los enlaces a los documentos de Overleaf como este mismo reporte. Además poseerá una carpeta en la que se almacenarán las versiones finales de las actas en formato docx y pdf.

En el repositorio de Back-end existirá una carpeta donde se almacenarán tanto los ficheros sql de creación, poblado y/o disparadores de la base de datos como ficheros que describen ciertas normalizaciones aplicadas y modelo relacional y como el esquema Entidad Relación. Por otro lado habrá otra carpeta en la que se añadirán los ficheros correspondientes al sistema desarrollado en Django.

El repositorio de Front-end tendrá dos carpetas, una para los ficheros de la aplicación desarrollada con Flutter, y otra para la aplicación desarrollada en Ionic.

En cada repositorio se utilizará el gestor de incidencias de GitHub (issue tracker) para controlar el estado del proyecto mediante incidencias. Estas describen las tareas a realizar, y podrán tener 3 estados diferentes: tareas pendientes (si nadie esta asignado a la tarea), tareas en curso (con miembros asignados) y tareas completadas. Las incidencias podrán referenciarse entre ellas para mostrar dependencias y jerarquías, todas estarán asociadas a una de las dos entregas que se realizarán a lo largo del proceso, todo esto con el objetivo de llevar control del orden de las tareas a desarrollar, y el correcto progreso del proyecto.

El flujo de trabajo que se utilizará en GitHub es el centralizado, consiste en tener una rama master principal para cada repositorio contral, donde se subirán todos los cambios importantes. Al disponer de un equipo relativamente pequeño cada integrante tiene una gran cantidad de responsabilidad, por ello los commits se podrán subir directamente al repositorio principal(sin la necesidad de pull request), y cada miembro tendrá la capacidad de crear, cerrar y modificar el estado de las incidencias.

3.2.2 Plan de se construcción y despliegue de software

No se emplearán scripts de automatización para realizar la compilación, pero se asegurará que la compilación sea igual en todos los casos documentándolo con un archivo `README.md` en los repositorios de GitHub del proyecto. La instalación de cada IDE y framework se ha realizado de la misma forma en todos los equipos que se van a utilizar para desarrollar, con el objetivo de que se usen las mismas dependencias.

Se realizará una compilación progresiva, es decir, se compilarán exclusivamente las secciones donde se hayan realizado cambios. Por ejemplo, si se finalizase un módulo antes de tiempo, sólo se recompilará en el momento en el que se haya realizado un cambio a dicho módulo o a otro del que éste dependa.

Como se ha descrito en la sección de procesos, el servicio de almacenamiento para la base de datos se encontrará en la nube con el sistema de Azure Database for PostgreSQL de Microsoft [4]. Además se utilizará el servicio de Function App de la misma plataforma [2] para el proceso de back-end (proceso que maneja la base de datos) desarrollado en Python y el servicio Web App [3] para desplegar la aplicación web.

Se seguirán los asistentes de configuración de los servicios de Azure para configurarlos.

3.2.3 Plan de aseguramiento de la calidad

Se emplearán la guía de diseño de aplicaciones de Google para el desarrollo de la GUI de la aplicación móvil [6].

Para el diseño de la UI de la aplicación web, se utilizará la guía que proporciona Google para desarrolladores [7].

En cuanto a la calidad del código, esta será garantizada mediante revisión de código por pares, ya que en cada subequipo hay por lo menos dos integrantes. Además, una vez el código este desarrollado, se realizarán una serie de pruebas manuales sobre la aplicación en funcionamiento que permitan garantizar el correcto funcionamiento de todas sus funcionalidades.

Los requisitos y diagramas de diseño realizados a lo largo del proyecto son otro indicador de la calidad del proyecto, ya que permiten a los desarrolladores un entendimiento equivalente de los problemas a desarrollar. Los requisitos del proyecto han sido revisados y aprobados por todos los integrantes del equipo, y los diagramas de diseño serán revisados por pares, igual que el código.

3.2.4 Calendario del proyecto y división del trabajo

El desarrollo se dividirá en las siguientes tareas:

Tarea	Descripción	Requisitos
1	Diseño y puesta en marcha de la base de datos y sus funciones (p.e. búsqueda e inserción)	RF-1, RNF-1, 2
2	Creación de la interfaz de usuario en aplicación móvil.	RNF-4, 5, 6
3	Creación de la interfaz de usuario en aplicación web.	RNF-4, 5, 6
4	Puesta en marcha de las funcionalidades de reproducción en streaming.	RF-19, 20, 23, 24, 25, 27, 28
5	Implementación del sistema de episodios de podcasts.	RF-19, 20, 24, 25, 28, 32
6	Puesta en marcha del sistema de inicio de sesión de usuario.	RF-2, 4, 5
7	Implementación del sistema de listas de reproducción, carpetas, favoritos y artistas.	RF-3, 9, 10, 11, 12, 13, 14, 15, 17, 19
8	Implementación del sistema de géneros.	RF-8, 22
9	Puesta en marcha de las funcionalidades de reproducción para archivos locales.	RF-19, 20, 23, 24, 25, 26, 28
10	Implementación de la barra de búsqueda.	RF-7
11	Implementación de la funcionalidad para creadores de contenido.	RF-31, 33, 34, 35, 36, 37, 38, 39, 40
12	Puesta en marcha del sistema de amigos.	RF-29
13	Diseño e implementación de las opciones de personalización del perfil de usuario.	RF-6
14	Implementación del sistema para compartir listas y carpetas (público o privado).	RF-16, 30
15	Implementación de la funcionalidad de continuidad de reproducción en otros dispositivos.	RF-21
16	Implementación del sistema de notificaciones.	RF-41, 42
17	Documentación del código	
18	Prueba del Back-End	
19	Prueba de la aplicación móvil	
20	Prueba de la aplicación web	
21	Memoria del proyecto y cambios realizados	
22	Prueba del sistema completo	
23	Despliegue de los servidores (base de datos, aplicación web)	RNF-8
24	Documentación del despliegue final	

Los miembros del equipo Back-end participarán en las tareas 1, 2, 3, 5, 6, 9, 10, 11, 12, 14, 15, 17, 18, 19, 20, 21 y 24.

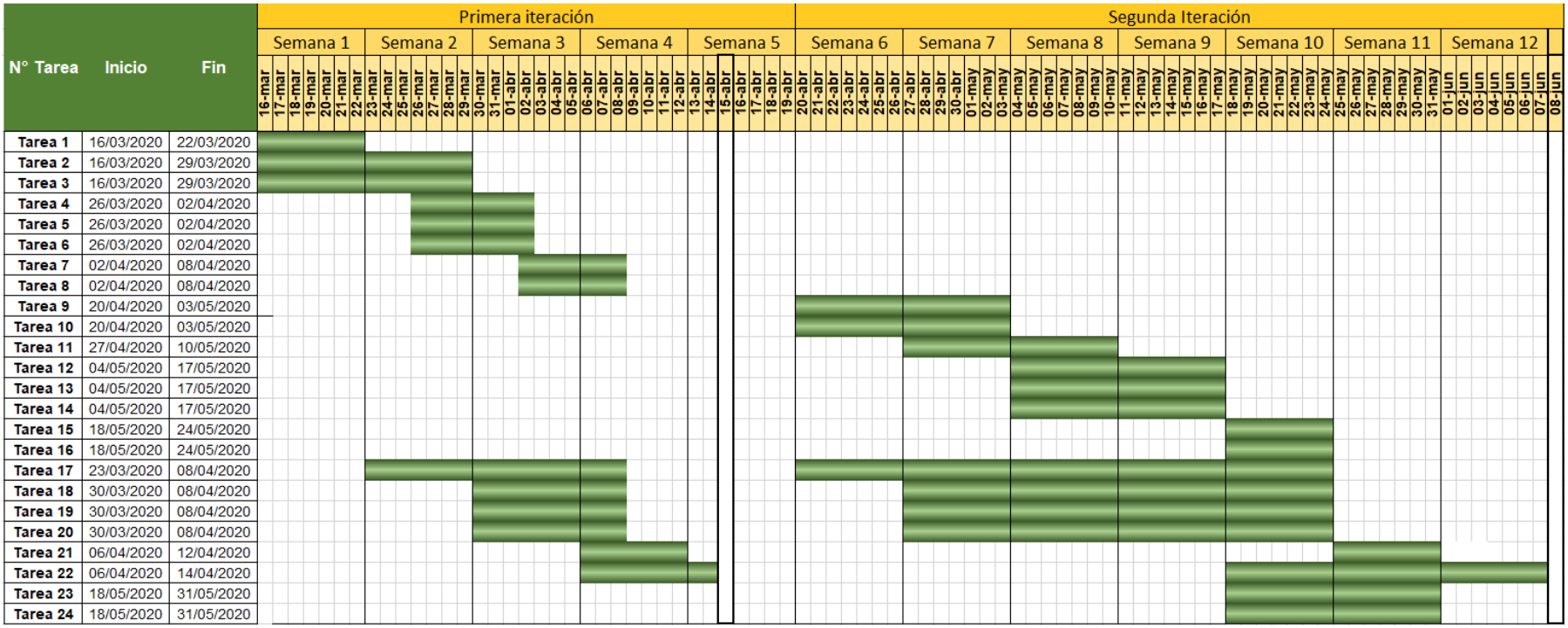
Los miembros encargados de la app móvil participarán en las tareas 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 18, 19, 20, 23 y 24.

Finalmente los miembros encargados de la app web participarán en las tareas 2, 3, 4, 5, 6, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 22 y 24.

Diagrama de Gantt

Las tareas a realizar se han recogido en el siguiente diagrama de Gantt, junto con las fechas previstas para cada una.

Como se puede ver, la primera iteración está dedicada a crear el esqueleto del sistema e implementar las funcionalidades más básicas, mientras que en la segunda iteración, se implementarán el resto de funcionalidades y se pulirán algunos detalles.



N° Tarea	Inicio	Fin
Tarea 1	16/03/2020	22/03/2020
Tarea 2	16/03/2020	29/03/2020
Tarea 3	16/03/2020	29/03/2020
Tarea 4	26/03/2020	02/04/2020
Tarea 5	26/03/2020	02/04/2020
Tarea 6	26/03/2020	02/04/2020
Tarea 7	02/04/2020	08/04/2020
Tarea 8	02/04/2020	08/04/2020
Tarea 9	20/04/2020	03/05/2020
Tarea 10	20/04/2020	03/05/2020
Tarea 11	27/04/2020	10/05/2020
Tarea 12	04/05/2020	17/05/2020
Tarea 13	04/05/2020	17/05/2020
Tarea 14	04/05/2020	17/05/2020
Tarea 15	18/05/2020	24/05/2020
Tarea 16	18/05/2020	24/05/2020
Tarea 17	23/03/2020	08/04/2020
Tarea 18	30/03/2020	08/04/2020
Tarea 19	30/03/2020	08/04/2020
Tarea 20	30/03/2020	08/04/2020
Tarea 21	06/04/2020	12/04/2020
Tarea 22	06/04/2020	14/04/2020
Tarea 23	18/05/2020	31/05/2020
Tarea 24	18/05/2020	31/05/2020

Figura 1: Diagrama de Gantt con la distribución de tareas.

4. Análisis y diseño del sistema

En este apartado se incluye la documentación sobre los requisitos del sistema a desarrollar.

4.1 Análisis de requisitos

A continuación, se muestran los requisitos funcionales, no funcionales y restricciones sobre el sistema.

4.1.1 Requisitos funcionales

Código	Descripción
	Sistema
RF-1	La aplicación tendrá una base de datos de canciones y podcasts[1] que los usuarios podrán reproducir en streaming.
RF-2	Habrá tres tipos de usuario, estándar, premium y creador de contenido.
RF-3	Las canciones se organizan en el sistema en listas de reproducción y favoritos, por álbum, y por artista.
RF-4	Todos los usuarios deberán registrarse en el sistema utilizando un nombre de usuario, correo y contraseña.
RF-5	Todos los usuarios deberán identificarse en el sistema con su nombre de usuario o correo, y contraseña, previamente registrados.
RF-6	Los usuarios podrán personalizar su perfil cambiando su foto de usuario.
RF-7	El sistema ofrece un buscador de música, podcasts, álbumes, listas de reproducción y usuarios.
RF-8	Un administrador del sistema podrá añadir géneros (como Pop, Rock, Electrónica...) que estarán vacíos inicialmente hasta que un artista añada una canción al mismo (explicado en el RF-39)
	Listas de reproducción
RF-9	Los usuarios estándar podrán crear listas de reproducción o bien con sólo canciones almacenadas localmente o bien con sólo música en streaming.
RF-10	Los usuarios premium podrán crear listas de reproducción con canciones almacenadas localmente y en streaming.
RF-11	Los usuarios premium podrán añadir canciones tanto locales como en streaming a sus listas de reproducción.
RF-12	Los usuarios estándar podrán añadir canciones locales a una lista de reproducción de canciones almacenadas en local, o bien añadir canciones en streaming a una lista de reproducción de música en streaming.
RF-13	Todos los usuarios pueden eliminar canciones de una lista de reproducción.
RF-14	Todos los usuarios pueden eliminar sus listas de reproducción.
RF-15	Todos los usuarios podrán organizar sus listas de reproducción en carpetas.
RF-16	Las listas de reproducción y las carpetas pueden hacerse públicas o privadas. Cualquier otro usuario podrá ver y reproducirlas si es pública.
RF-17	La lista de favoritos será una lista que no podrá ni borrarse ni compartirse.

[1] Un podcast es un programa de audio formado por diferentes episodios.

RF-18	Todos los usuarios podrán marcar como favoritas canciones, que pasarán automáticamente a formar parte de la lista de favoritos del usuario.
	Reproducción
RF-19	Todos los usuarios podrán reproducir música y podcasts que estén almacenados en el dispositivo o en el sistema de streaming.
RF-20	Todos los usuarios podrán pausar o reanudar la reproducción actual en todo momento y en cualquier dispositivo.
RF-21	Un usuario podrá reanudar la reproducción de un podcast desde el momento en el que estaba incluso después de cerrar el programa o pasar a otra reproducción.
RF-22	Toda canción deberá estar incluida al menos en un género (Pop, Rock, Electrónica, Reggaeton...)
RF-23	Todos los usuarios podrán reproducir música de forma aleatoria y/o en bucle dentro de un conjunto de canciones[2].
RF-24	Los usuarios premium podrán cambiar de canción o podcast en todo momento.
RF-25	Los usuarios estándar podrán cambiar de podcast/episodio de podcast en todo momento.
RF-26	En el caso de canciones, los usuarios estándar podrán saltar canciones dentro de un conjunto formado únicamente por archivos de audio locales.
RF-27	Los usuarios estándar podrán saltar a la siguiente o anterior canción dentro de un conjunto que contenga canciones en streaming un máximo de 10 veces por hora.
RF-28	Un usuario premium podrá añadir en todo momento canciones o podcasts a su cola de reproducción, para se reproduzcan a continuación.
	Social
RF-29	Todos los usuarios podrán añadir a otros usuarios como amigos. Si el otro usuario acepta la solicitud de amistad, el primero podrá ver las canciones o podcasts más escuchadas del segundo, géneros más escuchados y las últimas 20 canciones o podcasts escuchados.
RF-30	Se podrán compartir canciones, podcasts o listas de reproducción generando un enlace.
RF-31	Todos los usuarios podrán seguir a creadores de contenido.
RF-32	Todos los usuarios podrán suscribirse a podcasts.
	Creadores de contenido
RF-33	Los creadores de contenido serán usuarios verificados por los administradores de la aplicación.
RF-34	Los creadores de contenido dispondrán de los mismos privilegios que los usuarios premium.
RF-35	Los creadores de contenido podrán crear álbumes de canciones.
RF-36	Los creadores de contenido podrán añadir, modificar, borrar y cambiar de orden canciones en los álbumes.
RF-37	Los creadores de contenido podrán crear podcasts.
RF-38	Los creadores de contenido podrán añadir, modificar, borrar y cambiar de orden episodios en los podcasts.
RF-39	Al añadir una canción, los creadores de contenido deberán añadir autores adicionales que participan en ella, definirla dentro de un género (Pop, Rock, Electrónica...) y definir el idioma.

[2]Se considera como conjunto: una lista de reproducción, la lista de favoritos, un álbum, las canciones de un artista, la cola de reproducción o un género.

RF-40	Al añadir un episodio a un podcast, los creadores de contenido deberán añadir colaboradores que aparecen en él, crear una descripción y definir el idioma.
Notificaciones	
RF-41	Los usuarios recibirán notificaciones con la actividad de los creadores de contenido que sigan.
RF-42	Los usuarios recibirán una notificación cada vez que se suba un nuevo programa a un podcast que sigan.

Tabla 2: Requisitos funcionales.

4.1.2 Requisitos no funcionales

Código	**Descripción**
RNF-1	La contraseña se almacenará de forma segura, utilizando cifrado.
RNF-2	El sistema soportará los formatos `mp3` y `ogg` para el almacenamiento y reproducción de contenido.
RNF-3	Para ser usuario premium, se cobrará una cuota mensual.
RNF-4	El sistema funcionará de manera fluida y la experiencia de usuario será satisfactoria.
RNF-5	Las canciones y los podcasts se mostrarán en zonas diferentes de la interfaz.
RNF-6	Para cada podcast se mostrará una resumen de su contenido
RNF-7	Todos los pagos se realizarán de forma segura a través de una pasarela de pagos.
RNF-8	El sistema de reproducción en streaming estará disponible 24/7.

Tabla 3: Requisitos no funcionales.

4.1.3 Restricciones del sistema

- El dispositivo en el que se utilice la aplicación deberá disponer de conexión a internet para poder acceder a los recursos online.
- El pago para ser usuario premium podrá realizarse con tarjeta o a través de PayPal.

4.1.4 Diagrama de Casos de Uso

El catálogo de requisitos anterior se concreta en el siguiente diagrama de Casos de Uso.

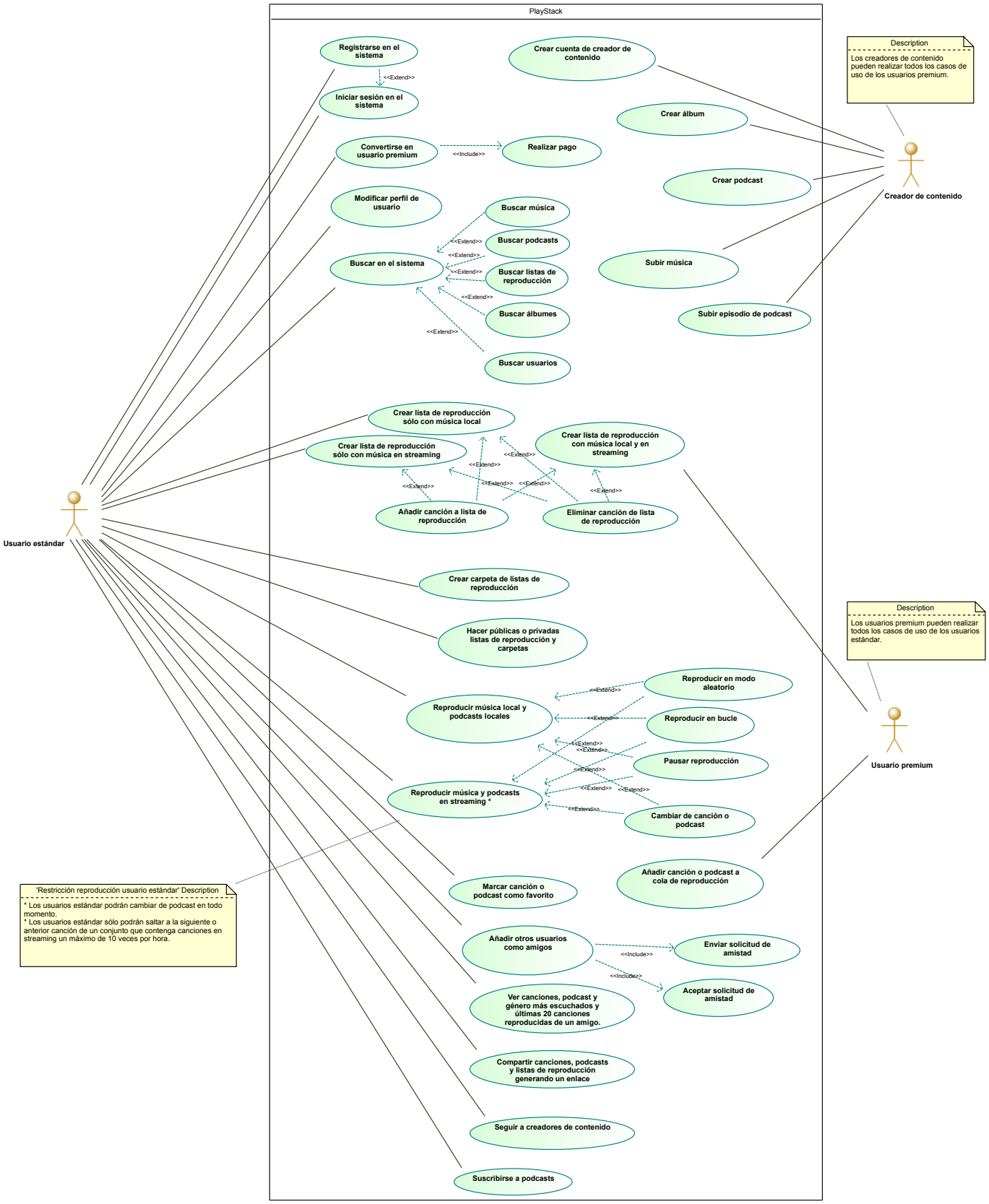


Figura 2: Diagrama de módulos

4.2 Diseño del sistema

4.2.1 Diagramas arquitecturales

A continuación se muestran los diagramas de módulos, componentes y despliegue, que, a falta de concretar más las decisiones del diseño, permiten esbozar cómo van a ser las interacciones y las comunicaciones entre los diferentes componentes de la aplicación.

El diagrama de módulos se centra en mostrar los diferentes paquetes y componentes diferenciables del código, así como las estructuras del sistema, basándose en cómo se estructuraría el código. En el momento de la redacción de este documento, aún faltan determinados aspectos clave por debatir y decidir en consenso, por lo que el nivel de detalle no es todo el que se podría haber conseguido.

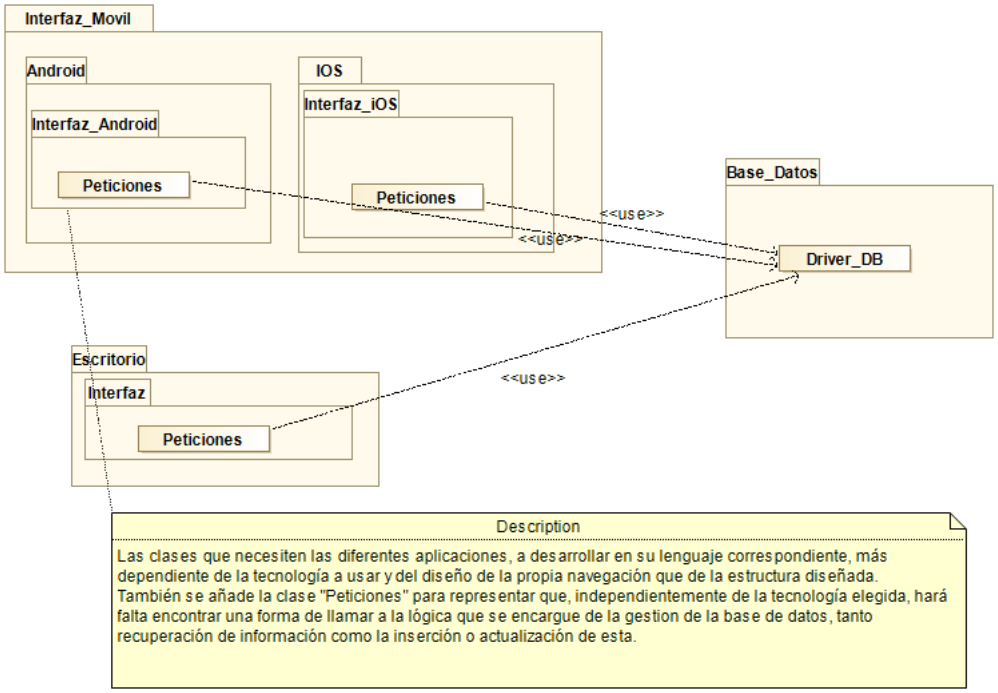


Figura 3: Diagrama de módulos

En el diagrama de componentes se ilustra la idea de que sea la aplicación quien realice diferentes peticiones a los componentes de la API del backend como respuesta a las acciones de los usuarios. No obstante y como ya se ha comentado, aún se tienen que decidir ciertos aspectos del diseño final que se va a implementar.

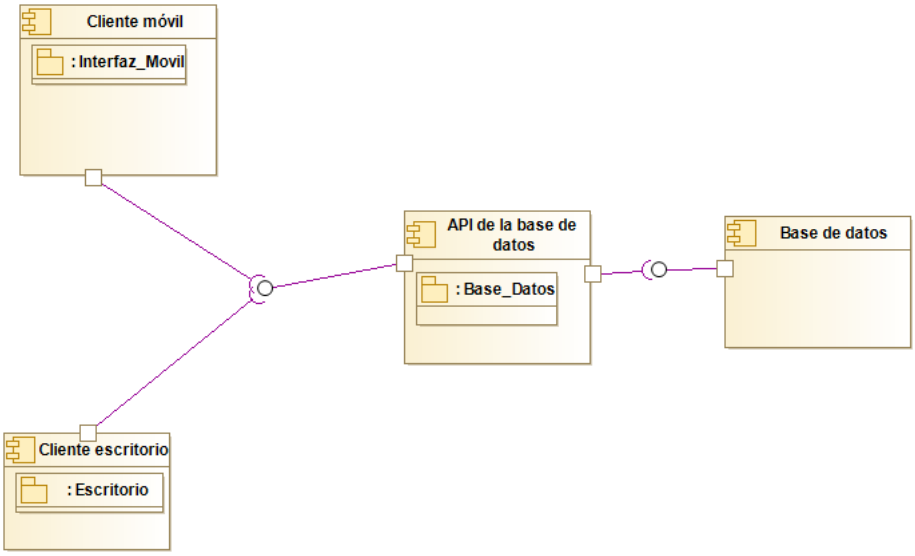


Figura 4: Diagrama de componentes

Por último, en el diagrama de despliegue se muestra cómo la aplicación dentro de los terminales de los usuarios se comunica con la máquina alojada en los servidores cloud de Azure, que alojará la base de datos como tal. Todo ello mediante la API desarrollada por el equipo de Back-End.

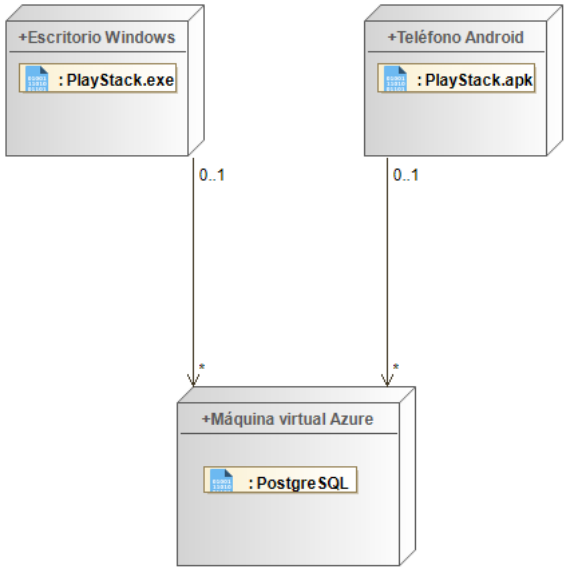


Figura 5: Diagrama de despliegue de PlayStack en diferentes máquinas

4.2.2 Tecnologías elegidas

Como se ha comentado anteriormente, el desarrollo del sistema se va a dividir en Back-End y Front-End. A continuación se detallan las tecnologías escogidas para cada nivel.

- **Front-End**: Se ha escogido el framework Django, que trabaja sobre Python.

- **Back-End**: Las aplicaciones escritorio y web se van a desarrollar en Ionic sobre Angular, que utiliza el lenguaje TypeScript. Para la aplicación móvil se ha escogido Flutter, que utiliza Dart.

En el Back-End se ha elegido Django, principalmente porque uno de los desarrolladores del Back-End (Alberto Martínez) tenía experiencia previa con este framework.

La elección de Flutter para la aplicación móvil, fue también por un motivo similar, el desarrollador Enrique Ruiz ya había realizado varios proyectos con esta herramienta. Además, es muy adecuada para este sistema porque permite un desarrollo rápido y alcanzar rendimiento nativo.

Debido a la similitud entre la aplicación de escritorio y la web, se van a desarrollar de forma conjunta, en forma de aplicación híbrida. Se ha elegido Ionic con Angular porque, a pesar de que ningún desarrollador tenga experiencia previa con esta tecnología, dispone de una amplia documentación y tiene librerías de componentes de gran calidad.

4.2.3 Otros aspectos técnicos

En la base de datos, se va a utilizar el modelo relacional relacional. No se va a almacenar grandes cantidades de información ni de tipos diferentes. Además, ningún miembro del equipo tiene experiencias con bases de datos NoSQL, por lo que es la opción más adecuada. Como SGBD, se ha escogido PostgreSQL, es SGBD relacional más usado en la comunidad de Django y el que mejor se integra actualmente con el framework.

Las respuestas de las peticiones al Back-End se enviarán en formato JSON a las aplicaciones del Front-End, ya que TypeScript y Dart facilitan el procesado de este formato.

Referencias

[1] *Apple Music*. URL: https://www.apple.com/es/music/ (visitado 21-02-2020).

[2] *Descripción de la función Function App de Microsoft Azure*. URL: https://azuremarketplace.microsoft.com/es-es/marketplace/apps/Microsoft.FunctionApp?tab=Overview (visitado 12-03-2020).

[3] *Descripción de la función Web App de Microsoft Azure*. URL: https://azuremarketplace.microsoft.com/es-es/marketplace/apps/Microsoft.WebSite?tab=Overview (visitado 12-03-2020).

[4] *Descripción del servicio Azure Database for PostgreSQL*. URL: https://azuremarketplace.microsoft.com/es-es/marketplace/apps/Microsoft.PostgreSQLServer?tab=Overview (visitado 12-03-2020).

[5] *Google Music*. URL: https://play.google.com/music (visitado 21-02-2020).

[6] *Guía de diseño de aplicaciones móvil de Google*. URL: https://www.thinkwithgoogle.com/intl/es-419/recursos-y-herramientas/aplicaciones/principles-of-mobile-app-design-download-complete-guide/ (visitado 12-03-2020).

[7] *Guía de diseño de aplicaciones web de Google*. URL: https://developers.google.com/web/fundamentals (visitado 12-03-2020).

[8] *Guía de estilo de Google para JavaScript*. URL: https://google.github.io/styleguide/jsguide.html (visitado 12-03-2020).

[9] *Guía de estilo para Dart*. URL: https://dart.dev/guides/language/effective-dart/style (visitado 12-03-2020).

[10] *Guía de estilo para Python PEP8*. URL: https://www.python.org/dev/peps/pep-0008/ (visitado 11-03-2020).

[11] *Ionic 4 & Angular Tutorial For Beginners*. URL: https://youtu.be/r2ga-iXS5i4 (visitado 12-03-2020).

[12] *Ionic Crash Course*. URL: https://ionicacademy.com/ionic-crash-course/ (visitado 12-03-2020).

[13] *Material de la asignatura Proyecto Software – Curso 2019-2020*.

[14] *Spotify*. URL: https://www.spotify.com/es/ (visitado 21-02-2020).

[15] *Tidal*. URL: https://tidal.com (visitado 21-02-2020).

[16] *Tutorial de Ionic*. URL: https://ionicframework.com/docs/angular/your-first-app (visitado 27-02-2020).

[17] *Tutorial de JavaScript*. URL: https://developer.mozilla.org/es/docs/Web/JavaScript/Una_re-introducci%C3%B3n_a_JavaScript (visitado 27-02-2020).

[18] *Tutoriales de Flutter*. URL: https://flutter.dev/docs/reference/tutorials (visitado 03-03-2020).