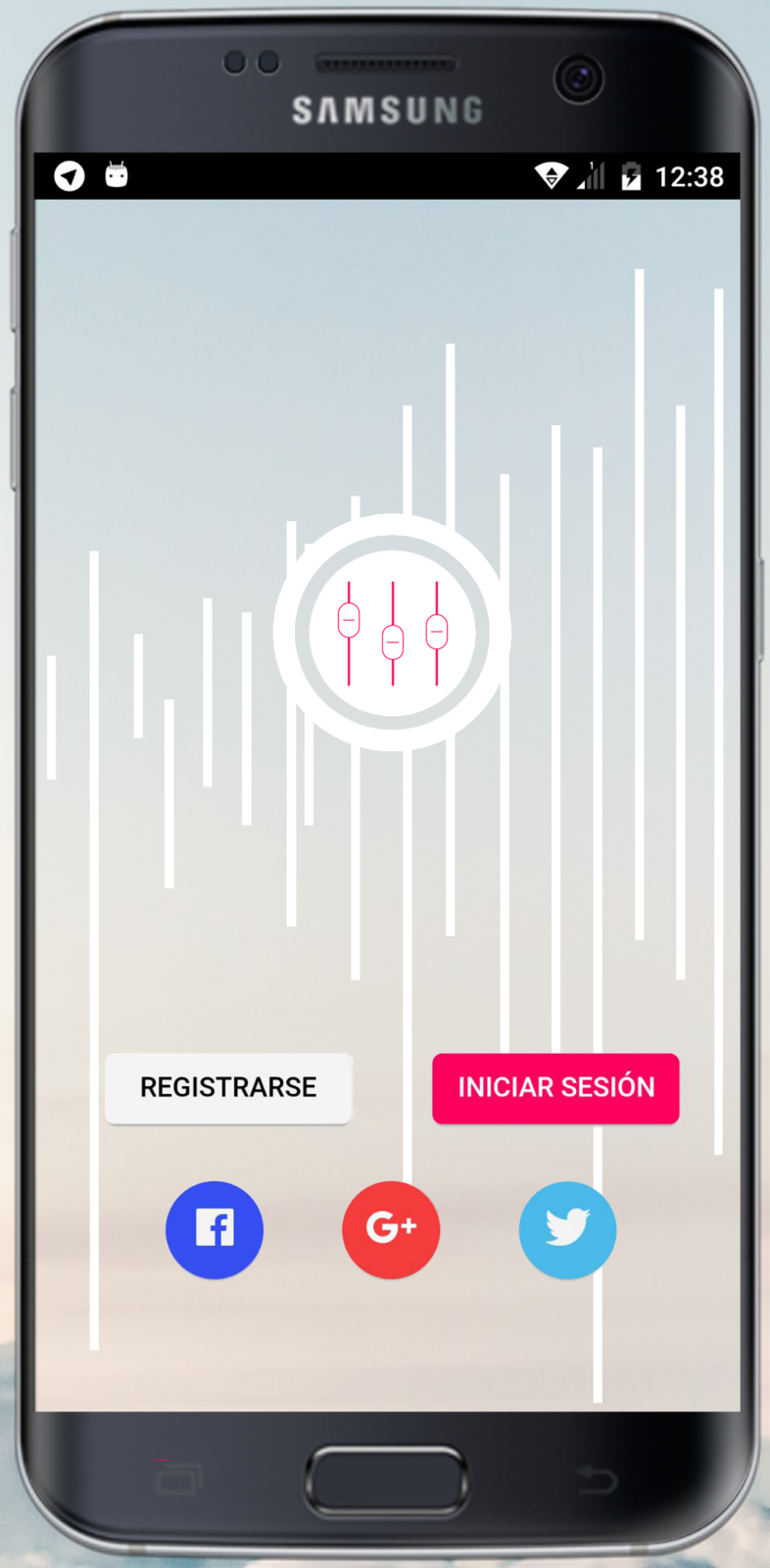
UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Escuela de Ingeniería Informática

MUSICFACT



DESARROLLO DE UN PROTOTIPO DE APLICACIÓN MÓVIL

Plataforma de distribución de audio

Trabajo de Fin de Grado - Grado en Ingeniería Informática

Alexander Reyes Sánchez - Tutor: Abraham Rodríguez Rodríguez - Julio 2017

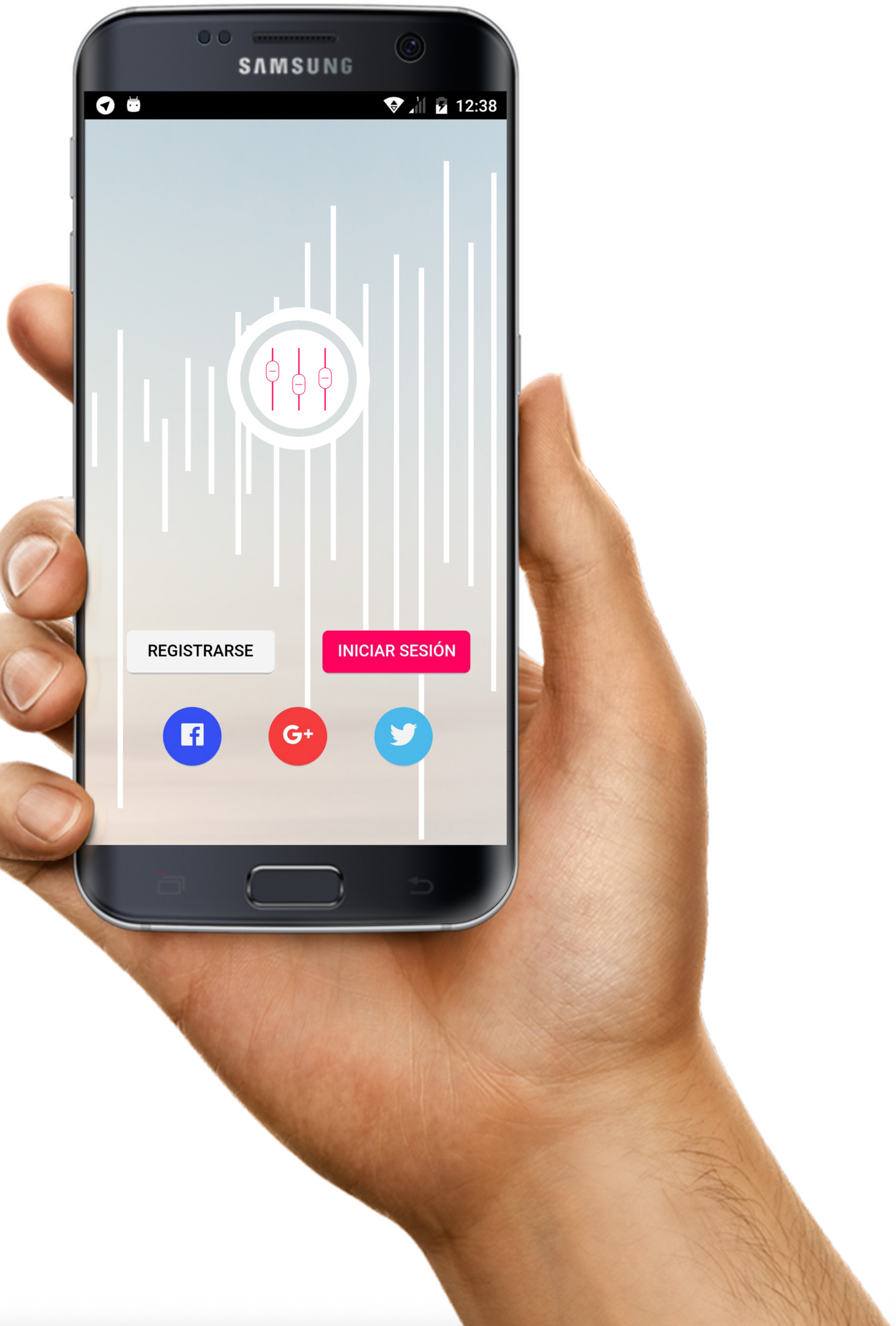
M U S I C F A C T

DESARROLLO DE UN PROTOTIPO DE APLICACIÓN MÓVIL

Plataforma de distribución de audio

Trabajo de Fin de Grado - Grado en Ingeniería Informática

Alexander Reyes Sánchez - Tutor: Abraham Rodríguez Rodríguez - Julio 2017



Índice

1| Introducción 1

1.1 Contexto 1

1.2 Objetivos 2

1.3 Metodología 3

1.4 Competencias específicas cubiertas 4

1.5 Aportaciones al entorno socioeconómico, técnico o científico 5

1.6 Estructura de la memoria 5

2| Análisis 6

2.1 Estudio de aplicaciones similares 6

2.2 Requisitos 18

2.3 Diagramas de casos de uso 22

2.4 Mockups, diseño de la interfaz y paleta de colores 24

2.5 Modelo de negocio 25

2.6 Normativa y legislación 27

3| Diseño e implementación 30

3.1 Alcance de la implementación 30

3.2 Diseño arquitectónico 33

3.2.1 Módulos 35

3.2.2 Componentes 35

3.2.3 Plantillas 36

3.2.4 Metadatos 37

3.2.5 Data binding 37

3.2.6 Directivas 37

3.2.7 Servicios 38

3.2.8 Inyección de dependencias 38

3.3 Modelo de la base de datos 39

3.3.1 Realtime Database 39

3.3.2 Autenticación 45

3.3.3 Storage 46

3.4 Tecnologías 47

3.4.1 HTML5, JavaScript, CSS3 y Sass 47

3.4.2 TypeScript *48*
3.4.3 Ionic 2 ..... *49*
3.4.4 Ionic Native y Apache Cordova/PhoneGap *50*
3.4.5 Firebase Notifications *53*
3.4.6 Ionic Storage *54*
3.4.7 Ionic Creator *54*
3.4.8 AngularFire2 ..... *55*
3.4.9 Lodash *55*
3.4.10 Angular2-Charts *56*
3.4.11 GitHub *56*
3.4.12 Waffle.io *56*
3.4.13 Zedge *57*
3.5 Pruebas 57
3.6 Acceso al código y despliegue 59
4| Conclusiones 60
Fuentes de información 61
Anexos 63
A) Manual de usuario 63
B) Aplicaciones Híbridas 76
C) Camera 77

1 | Introducción

1.1 CONTEXTO

Música/Audio

El contexto en el que se basa nuestra aplicación parte de la sociedad actual que busca un mecanismo de desconexión durante algunas horas del día para mantener la mente ocupada, amenizar las horas de trabajo, trayectos en coche o a modo ambiental en celebraciones.

Las aplicaciones móviles han tomado importancia desde el surgimiento de los smartphones, ya que la mayor parte de las personas eligen sus dispositivos móviles como vía principal para interactuar con sus redes sociales. Por ello, el desarrollo de aplicaciones móviles ha ido incrementando exponencialmente y, con este crecimiento, surgiendo nuevas tecnologías.

Con las nuevas tecnologías, cambia la manera de desarrollar ciertas tareas que probablemente antes costaban más tiempo y esfuerzo. De esta forma, conseguimos aumentar la productividad de los desarrolladores.

Ilustración 1 - Incremento exponencial de aplicaciones móviles [1]

[1] Incremento exponencial de aplicaciones móviles - Julio 2017

Centrándonos en nuestro trabajo, las redes sociales tienen muchas utilidades, te dan la oportunidad de mantenerte en contacto con tus amigos, ver fotos de tus familiares en otros países o ciudades, ver vídeos originales, escuchar canciones, etc. Sin embargo, ¿qué opciones tienen aquellas personas creativas que se dedican a grabar música individualmente o con un grupo/banda, pero que no cuentan con el respaldo de una compañía discográfica o un sello multinacional? Hoy en día, no es nada fácil darse a conocer y para ello normalmente es necesaria una inversión.

Nuestra red social está destinada principalmente para esos músicos, de tal manera que puedan subir, registrar, promover y difundir sus creaciones a través de canales, promocionando su música de forma simple sin la necesidad de utilizar otros servicios de alojamiento de archivos.

Por otro lado, tenemos a los usuarios que eligen la plataforma como reproductor de música, álbum, seguimiento de sus artistas favoritos, ... Son estas personas de diferentes lugares de todo el mundo las que promueven las canciones.

1.2 OBJETIVOS

Para esta parte de la memoria, estableceremos los diferentes objetivos que se pretenden alcanzar con el desarrollo de la aplicación. Dichos objetivos los podemos dividir en tres categorías diferentes:

- Sociales: Intensificando en los usuarios que eligen la plataforma como reproductor de música, biblioteca de audio, etc.

- Económicos: Para los usuarios/músicos que pretenden comercializar sus creaciones musicales.

- Estadísticos: Para los usuarios/músicos que están interesados en estudiar los gustos musicales predominantes en la sociedad actual.

En resumen, el objetivo principal de este proyecto es ayudar a las personas creativas a sacar a la luz sus creaciones musicales para que puedan llegar en algún momento a comercializase. A parte de proporcionar al resto de usuarios una plataforma músico-social donde puedan realizar un seguimiento de sus artistas favoritos, escuchar sus canciones y/o conocer gente con los mismos gustos musicales.

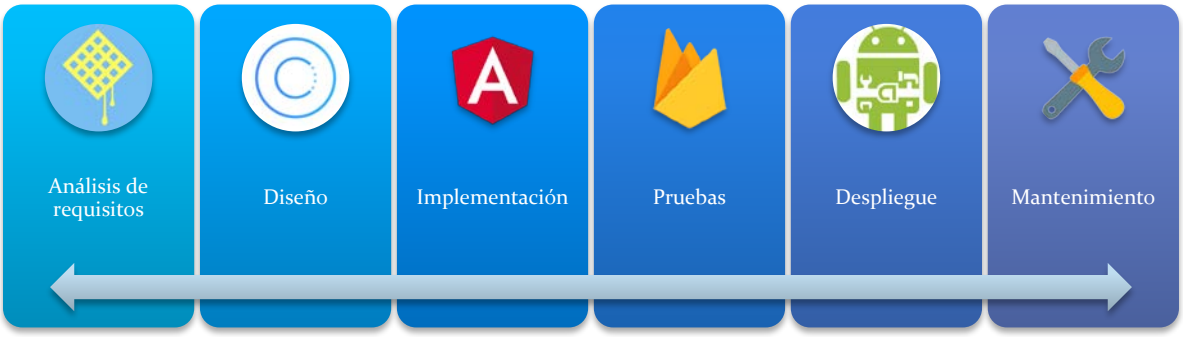
Como resultado del trabajo obtendremos un enorme tesoro de contenidos, entretejiéndolo todo en la red y haciendo que sea más fácil compartir nuestro ingenio.

1.3 METODOLOGÍA

Para el desarrollo de este proyecto se ha seguido el modelo clásico en cascada. Este modelo usa un ciclo de vida lineal secuencial, es decir, cada fase debe ser completada antes de que la siguiente fase pueda comenzar, por lo que normalmente el resultado de una fase actúa como entrada para la siguiente.

Dentro de cada fase, utilizaremos diferentes herramientas y tecnologías. De entre ellas, cabe destacar algunas como **Waffle.io** para la fase de requisitos, **Ionic Creator** para el diseño de la aplicación, **Angular** junto a otras tecnologías para la implementación, **Firebase** como base de datos en la nube y entorno para verificar los datos de las pruebas, o **Android SDK** para el despliegue de la aplicación en un dispositivo móvil.

A continuación, se presenta un esquema con las diferentes fases y herramientas mencionadas anteriormente:



Esquema 1 - Fases y herramientas para el modelo clásico en cascada

Entre las ventajas que tiene esta metodología de software destacamos el control. En otras palabras, podemos ajustar plazos para cada etapa del desarrollo. Sin embargo, una de las grandes desventajas del modelo en cascada es que no permite mucha reflexión o revisión. Por ejemplo, una vez que una aplicación se encuentra en fase de pruebas, es muy difícil volver atrás y modificar algún contenido.

Si comparamos esta metodología con la planificación prevista en la propuesta de trabajo de fin de grado, no existen grandes desviaciones.

Fases	***Duración Estimada*** *(horas)*	***Tareas*** *(nombre y descripción, obligatorio al menos una por fase)*
Estudio previo / Análisis	50	Tarea 1.1: Estudio de plan inicial con el tutor
		Tarea 1.2: Análisis de herramientas y tecnologías a usar
		Tarea 1.3: Estudio de mercado de aplicaciones semejantes
Diseño / Desarrollo / Implementación	180	Tarea 2.1: Diseño de formato para intercambio de datos
		Tarea 2.2: Desarrollo de funcionalidades
		Tarea 2.3: Diseño e implementación lógica del lado del servidor
		Tarea 2.4: Diseño e implementación interfaz frontend
Evaluación / Validación / Prueba	40	Tarea 3.1: Evaluación/Validación de funcionalidades implementadas
		Tarea 3.2: Pruebas de rendimiento en dispositivo móvil
Documentación / Presentación	30	Tarea 4.1: Redacción de la memoria final del proyecto
		Tarea 4.2: Desarrollo de la presentación para la defensa

Tabla 1 - Planificación prevista en la propuesta de trabajo

1.4 COMPETENCIAS ESPECÍFICAS CUBIERTAS

Con el presente trabajo, de acuerdo con la intensificación *Tecnologías de la Información* del *Grado en Ingeniería Informática*, se han cubierto las siguientes competencias específicas:

- Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.

- Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.

1.5 APORTACIONES AL ENTORNO SOCIOECONÓMICO, TÉCNICO O CIENTÍFICO

[2]Como aportación que nos ofrece la aplicación al entorno socioeconómico, distinguimos el impacto que recae en el mundo socio musical. Con este tipo de aplicaciones, se pretende reducir el número de descargas ilegales en la red.

Ilustración 2 - Piratería

El problema de la piratería recae en que el público ha sido entrenado por las grandes compañías de medios para desear y exigir entretenimiento sin fin, ya sea música, series, películas, etc. Una vez que se establece la adicción, depende del negocio mantener satisfecho al público. Si esto no se hace de forma eficiente, los consumidores acudirán a los medios de piratería. Sin embargo, si proporcionas medios parcialmente gratuitos al público con *streaming*, el problema se resuelve solo. Si al usuario le gusta la plataforma y le resulta cómoda (más que molestarse en descargar copias ilegales), probablemente no le importe pagar una cuota mensual que se ajuste a su bolsillo. En la sección 2.5 Modelo de negocio, indagaremos más sobre este tema.

1.6 ESTRUCTURA DE LA MEMORIA

La estructura de la memoria sigue la metodología de desarrollo software descrita anteriormente. En primer lugar, la etapa de análisis consta de un estudio previo de aplicaciones similares al proyecto, para luego poder definir los requisitos que debe cumplir la aplicación. Los requisitos pueden ser tanto funcionales como no funcionales. Para guiarnos de una forma más intuitiva, haremos uso de diagramas de casos de uso y algunos *mockups* para las pantallas más representativas. Finalmente, para terminar con esta fase hablaremos sobre el modelo de negocio a seguir y la normativa/legislación que debe cumplir nuestra aplicación.

En segundo lugar, se definen las características implementadas, junto con la arquitectura software y el modelo de la base de datos usados. Además, hablaremos sobre cada tecnología utilizada en el desarrollo del proyecto partiendo desde las bases.

Al finalizar la etapa de implementación, se describirá la metodología de pruebas seguida para validar el correcto funcionamiento de la aplicación. También se adjuntan varios enlaces para realizar un seguimiento del proyecto en GitHub y Waffle.io. Por último, se añaden las conclusiones que hemos sacado al terminar el desarrollo de la aplicación.

[2] Piratería - Julio 2017

2 | Análisis

En este capítulo se presenta la fase de análisis, parte inicial de todo proyecto software. Se estudia el mercado de aplicaciones similares si las hubiera, se definen los requisitos y se muestra una vista global de la arquitectura pensada para el sistema usando diagramas, *mockups*. Para ello, también se tiene en cuenta el modelo de negocio que se va a seguir y la normativa/legislación vigente.

Como aspecto más importante del presente capítulo, se expondrá el catálogo de requisitos como base para el diseño de todos los aspectos de la aplicación. Es por ello, por lo que la fase de análisis es de suma importancia para el devenir de todo producto software. Es en este capítulo donde se deben asentar las bases, a modo de cimientos, del proyecto, y a partir de las cuales se construirá todo lo demás.

2.1 ESTUDIO DE APLICACIONES SIMILARES

A continuación, desglosaremos tres aplicaciones '*top*' basadas en el contexto de la aplicación objeto del presente trabajo.

[3]Spotify

- Desarrollador: Spotify Ltd.
- Versión 8.4.1.846
- Sistema operativo: Android
- Fecha: 05/05/2017
- Más de 100 millones de descargas
- Puntuación de 4,5/5 con un número de votantes de 8.270.130
- Género de la aplicación: Música y audio

Ilustración 3 - Logo Spotify

[3] Logo Spotify - Julio 2017

Al iniciar la aplicación nos encontramos con una tarjeta-vídeo de bienvenida donde podemos crear una cuenta o iniciar sesión. Para crear una cuenta o iniciar sesión disponemos de dos opciones:

- Conectarnos/Iniciar sesión con Facebook.
- Registrarnos/Iniciar sesión con nuestra dirección de email: La contraseña debe ser de al menos 4 caracteres, un nombre de usuario, fecha de nacimiento, sexo (mujer | hombre).

Existe la opción "¿Se te ha olvidado la contraseña?" para poder recuperar la contraseña en caso de que la hayamos olvidado a través de nuestro correo electrónico/nombre de usuario.

Una vez iniciada la sesión, nos encontramos con varias secciones en la parte inferior. Vamos a describir dichas secciones una por una. En 'Inicio' aparecen varias listas de canciones divididas por categorías, de entre las cuales destacamos 'Basada en lo que has escuchado', 'Listas de éxitos' y 'Estado de ánimo'.

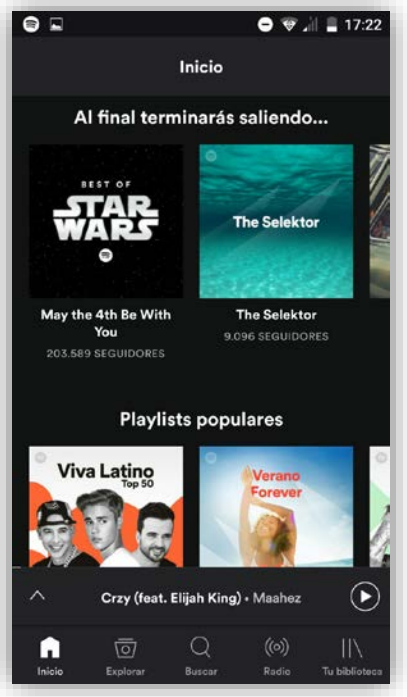


Ilustración 4 - Inicio Spotify

Cuando abrimos una lista, se muestra la imagen de portada junto con el título y número de seguidores. Dentro de las funcionalidades aquí presentes tenemos: Ordenar por título, artista, álbum o de forma personalizada a través de un filtro de texto; Seguir; Compartir lista; Ir a la radio de la lista; Modo aleatorio (única forma de escuchar las canciones de la lista si tenemos activada la cuenta gratis); Descargar la lista de canciones (sólo disponible con Spotify Premium); Lista de las canciones que incluye con diversas opciones como guardar canción, añadir a una lista, añadir a la cola, ir al álbum, ir al artista, compartir o ir a la radio de la canción. En cada canción hay un desplegable para poder visualizar información relevante sobre ésta.

En la sección 'Explorar', disponemos de una serie de listas de canciones y varias categorías: Listas de éxitos; Novedades; Vídeos; Podcasts; Descubrir; Conciertos (buscar conciertos populares cerca de una ubicación concreta); Géneros y estados de ánimo.

En 'Buscar' aparece una funcionalidad poco habitual. Es posible buscar por un código Spotify a través de la cámara de nuestro dispositivo o seleccionarlo en las fotos de nuestra galería. Además del buscador por defecto para encontrar canciones, álbumes, artistas, amigos, listas, podcasts y vídeos.

En 'Radio' nos encontramos con emisoras recomendadas y una lista de emisoras organizadas por género. También es posible buscar emisoras al igual que en el apartado anterior.

Por último, en 'Tu Biblioteca' observamos la lista de favoritos separados por categorías: Listas (posibilidad de crearlas); Emisoras; Canciones; Álbumes; Artistas; Podcasts y vídeos; Escuchado recientemente. En la barra superior tenemos dos iconos:

- A la izquierda un icono que nos lleva a nuestro perfil de usuario. Incorpora función para encontrar amigos en Facebook o seguir artistas famosos.

- A la derecha se encuentra la configuración de nuestra cuenta, donde podemos pasarnos a Premium (posibilidad de probarlo gratis durante 7 días). Además, existen diferentes apartados de configuración como, por ejemplo: Modo sin conexión (sólo para canciones o vídeos/podcasts ya descargados); *Crossfade*/difuminación entre una canción y otra; Conectar a un dispositivo; Sesión privada para no compartir lo que se escucha; Calidad de *Streaming* (a más calidad más consumo de datos); Ecualizador; Notificaciones; Información de versión, software de terceros, política de privacidad; Eliminar caché y los datos guardados; Cerrar sesión.

Cabe destacar que todas las pestañas incluyen, encima de las secciones, el reproductor con la canción actual. Al abrir el reproductor, éste ocupa toda la pantalla y aparecen botones con diferentes funcionalidades, de entre las cuales destacamos los subtítulos de la canción mejorando la experiencia del usuario (no disponible en todas las canciones).

Como gran desventaja podemos añadir los anuncios al terminar algunas canciones si estamos usando una cuenta gratuita.

[4]SoundCloud

Ilustración 5 - Logo SoundCloud

- Desarrollador: SoundCloud
- Versión 2017.04.19-release
- Sistema operativo: Android
- Fecha: 05/05/2017
- Más de 100 millones de descargas
- Puntuación de 4,4/5 con un número de votantes de 2.879.143
- Género de la aplicación: Música y audio
- Novedad: *The Upload* (lista personalizada que se actualiza a diario)

Al iniciar la aplicación nos encontramos con una tarjeta de bienvenida parecida a la que describimos en el análisis de Spotify con alguna diferencia como la posibilidad de crear una cuenta/iniciar sesión a través de Google Plus. Nos avisan de que al iniciar sesión estamos aceptando las condiciones de uso y la política de privacidad de SoundCloud (algo importante que más adelante abordaremos en el apartado sobre 2.6 Normativa y legislación).

En primer lugar, disponemos con cuatro secciones en la parte superior. Una de ellas es 'Home' donde aparece una lista de canciones. Estas canciones son las que publican los usuarios a los que seguimos o también canciones que han compartido dichos usuarios de otros usuarios.

En la siguiente sección, disponemos de un buscador que filtra por pistas de audio, gente, álbumes, listas o de forma general. Además, nos ofrecen pistas de canciones recomendadas y emisoras sugeridas en función de los 'me gusta' que hayamos dado. Seguimos con una serie de listas con novedades, top 50, listas de éxitos por género o incluso buscar listas populares por etiqueta (#hip hop, #rap, #techno, …).

[4] Logo SoundCloud - Julio 2017

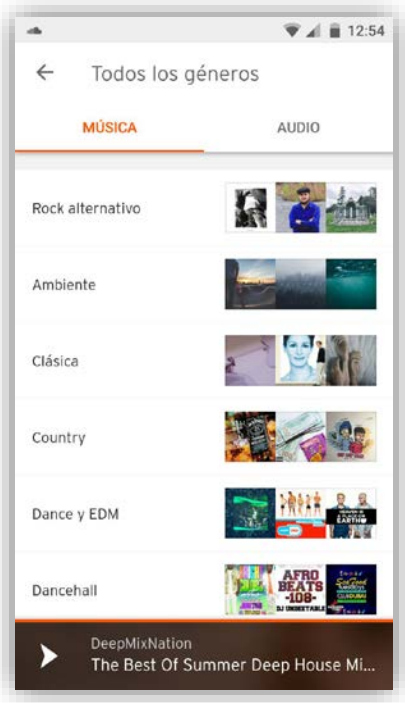


Ilustración 6 - Etiquetas SoundCloud

Luego, tenemos nuestra biblioteca de 'me gusta' divida en tres categorías:

- Pistas que te gustan
- Listas y álbumes
- Emisoras

A parte, podemos ver canciones escuchadas recientemente junto con un historial de reproducción en la parte inferior.

La última sección recoge nuestro perfil de usuario. En ella, observamos la imagen de perfil y el nombre de usuario junto a una opción para ver nuestro perfil público. Aquí podemos ver dos botones de navegación:

- Pistas: Aparecen las canciones que hemos compartido a nuestros seguidores, añadido a 'me gusta' y canciones propias subidas.
- Info: Aparecen nuestros seguidores y los usuarios a los que estamos siguiendo.

Luego, seguimos con un conjunto de varios apartados:

- Notificaciones: actividad de usuarios que han comenzado a seguirte, ...
- Grabar: iniciar una grabación sobre algún prototipo y publicarla con portada, título de la pista, permiso de visibilidad (posibilidad de recortarla, difuminado de entrada o salida).
- Configuración básica: actualizar datos sólo con wifi, borrar caché de la aplicación, privacidad.
- Configuración de notificaciones: notificaciones emergentes.
- Ayuda: link a la URL oficial de SoundCloud.
- Legal: información de derechos de autor, condiciones de uso, política de privacidad, información de la empresa.
- Cerrar sesión.

Todas las pestañas incluyen en la zona inferior el reproductor con la canción actual. Al abrirlo, ocupa toda la pantalla y se muestran botones con diferentes funcionalidades: Me gusta; Compartir; Lista de reproducción (aleatoria, repetición); Información de la canción; Comentar en 'minuto:segundo'; *Repostear*/Compartir; Añadir a una lista de canciones creada previamente o crearla en el momento (pública - privada); Emisora de pistas basada en la canción actual.

[5]Deezer

Ilustración 7 - Logo Deezer

- Desarrollador: Deezer Mobile
- Versión 5.4.4.88
- Sistema operativo: Android
- Fecha: 05/05/2017
- Más de 100 millones de descargas
- Puntuación de 4,1/5 con un número de votantes de 1.193.320
- Género de la aplicación: Música y audio

[5] Logo Deezer - Julio 2017

Al iniciar la aplicación nos encontramos con una tarjeta de bienvenida donde podemos crear una cuenta o iniciar sesión prácticamente con las mismas características que SoundCloud. Aviso al registrarnos para conocer mejor nuestros gustos musicales (novedad con respecto a otras aplicaciones del mismo género).

En primer lugar, disponemos de cuatro secciones para navegar en la parte superior. En 'Inicio' tenemos la opción de escuchar canciones sugeridas en función de los gustos que hayamos indicado al registrarnos. Posteriormente, aparecen varias transiciones o comúnmente llamadas '*slides*':

- Explorar: lista de etiquetas (fiesta, electrónica, popular, pop, deporte, niños, …)
- Selección de listas
- Popular
- Top Spain
- Nuevos lanzamientos
- (NOVEDAD) Fútbol: podemos ver próximos partidos, competiciones, canales, podcasts de fútbol, *playlist* de fútbol.

En la sección 'Mi Música' podemos ver nuestro perfil de usuario incluyendo diferentes opciones como, por ejemplo: Suscripción a Premium+ con 30 días gratis (escuchar música sin conexión y sin anuncios); Ver usuarios seguidos/seguidores; Mis canciones favoritas; Ajustes donde podemos, entre otros, suscribirnos a *Deezer Family* (14,99 €/mes), gestionar nuestra cuenta, ajustes de audio, modo sin conexión, *Deezer Labs* (uso de nuevas funcionalidades en fase de prueba).

Dentro de 'Notificaciones' disponemos de una lista con diferentes tipos de notificaciones: Álbum, nuevo *single*, nuevo álbum, listas. Cada notificación muestra a su vez la fecha de publicación, título, artista/s. Aquí, podemos añadir canciones a una lista, descargar, agregar a favoritos, ver la ficha del artista, compartir.

En 'Buscar' simplemente existe un buscador que filtra por texto, de forma que nos aparecen los resultados más buscados, artistas, álbumes, canciones, *playlists*, perfiles.

Por último, el reproductor de música que, desde mi punto de vista uno de los mejores, cumple las siguientes funcionalidades:

- Añadir a una lista creada o añadir una nueva
- Agregar a favoritos
- Ficha del artista
- Ficha del álbum
- Compartir
- Letras (estilo karaoke sólo algunas canciones)
- Lista de reproducción
- Ajuste de audio vistos anteriormente
- Reproducción aleatoria o repetición de canción actual

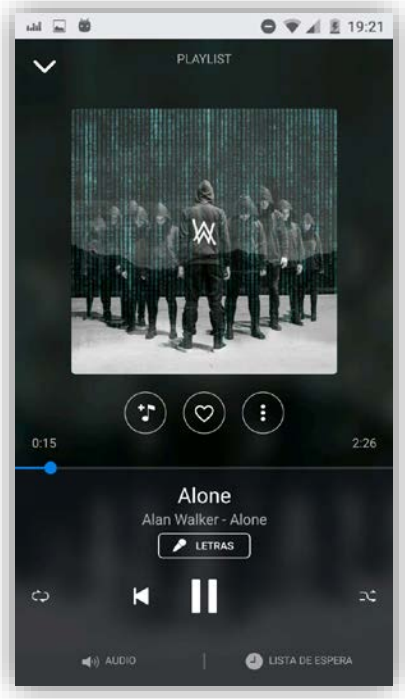


Ilustración 8 - Reproductor Deezer

Tabla comparativa de las aplicaciones

	SoundCloud	Deezer	Spotify	MusicFact
Desarrollador	SoundCloud	Deezer Mobile	Spotify Ltd.	Alexander Reyes
Versión	2017.04.19-release	5.4.4.88	8.4.1.846	2017.07
Descargas	Más de 100 millones	Más de 100 millones	Más de 100 millones	---
Puntuación-Votantes	4,4/5 - 2.879.143	4,1/5 - 1.193.320	4,5/5 - 8.270.130	---
Género	Música y audio	Música y audio	Música y audio	Música y audio
Novedades	- The Upload: lista personalizada que se actualiza a diario	- Funcionalidad al iniciar por primera vez para conocer gustos musicales - Fútbol	- Buscar conciertos populares cerca de una ubicación concreta - *Crossfade*	- Tendencias musicales
Registrarse/Iniciar sesión - Google+	✓	✓		✓
Registrarse/Iniciar sesión - Facebook	✓	✓	✓	✓
Registrarse/Iniciar sesión - Email/Pass	✓	✓	✓	✓
Opción ¿Has olvidado la contraseña?	✓	✓	✓	✓
Emisora/Radio (pistas similares)	✓		✓	
Visualización lista de canciones	✓	✓	✓	✓

	SoundCloud	Deezer	Spotify	MusicFact
Buscador	✓	✓	✓	✓
Canciones/listas recomendadas	✓	✓	✓	
Lista de canciones con novedades	✓	✓	✓	
Lista de canciones Top	✓	✓	✓	✓
Lista de éxitos por género	✓	✓	✓	
Lista de canciones por estado de ánimo			✓	
Buscar por etiqueta	✓	✓	✓	✓
***Likes*/Guardar en biblioteca**	✓	✓	✓	✓
Quitar *like* de una canción	✓	✓	✓	✓
***Repost*/Compartir**	✓			✓
Historial de reproducción	✓	✓	✓	✓
Perfil de usuario	✓	✓	✓	✓
Seguidores	✓	✓	✓	✓
Seguidos	✓	✓	✓	✓
Notificaciones	✓	✓	✓	✓
Grabar pista/Subir pista	✓			✓

	SoundCloud	Deezer	Spotify	MusicFact
Configuración básica	✓	✓	✓	
Configuración de notificaciones	✓	✓		✓
Cerrar sesión	✓	✓	✓	✓
Reproductor	✓			✓
Compartir en redes sociales	✓	✓	✓	✓
Mostrar información de la canción	✓	✓	✓	✓
Comentar canción	✓			✓
Crear listas de canciones	✓	✓	✓	✓
Agregar canciones a una lista	✓	✓	✓	✓
Escuchar canciones sugeridas	✓	✓		
Selección *playlists*	✓	✓	✓	
Suscripción	✓	✓	✓	
Ajustes de audio		✓	✓	
Dispositivos conectados		✓	✓	
Modo sin conexión (ya descargado)		✓	✓	

	SoundCloud	Deezer	Spotify	MusicFact
Uso funcionalidades en fase de prueba		✓		
Descargas		✓	✓ (Premium)	
Álbumes	✓	✓	✓	
Artistas	✓	✓	✓	
Tus creaciones	✓			✓
Publicidad otras aplicaciones		✓	✓	
Letras de canciones (algunas)		✓	✓	
Reproducción aleatoria	✓	✓	✓ (obligatoria con cuenta gratis)	
Ordenar lista		✓	✓	
Lista de canciones de la *playlist*	✓	✓	✓	✓
Vídeos			✓	
Podcasts	✓		✓	
Buscar por código (uso de la cámara del dispositivo/ galería)			✓	
Buscar amigos en Facebook		✓	✓	

	SoundCloud	Deezer	Spotify	MusicFact
Sesión privada			✓	
Añadir canciones/listas a la cola de reproducción	✓		✓ (Premium)	
Estudio estadístico para artistas				✓

Tabla 2 - Comparativa de aplicaciones

2.2 REQUISITOS

La aplicación es igual para cada perfil de usuario, depende de cada uno el uso que le quiera dar. Es decir, habrá usuarios que se limiten a usar la aplicación como una red social para escuchar música o aquellos que la usarán para difundir su música y poder ver el grado de aceptación entre los usuarios registrados en la aplicación.

A continuación, se desglosan las funcionalidades de la aplicación a modo de catálogo de requisitos, teniendo en cuenta tanto requisitos funcionales como no funcionales.

El catálogo de requisitos es la especificación del comportamiento que se espera de cualquier proyecto software. Estudiando el mercado de aplicaciones similares en la sección anterior, se ha predefinido una serie de requisitos que se consideran indispensables para el proyecto. Seguidamente, se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

Requisitos funcionales

Los requisitos funcionales describen todas las interacciones que tendrán los usuarios con el software.

- **Registro**
  1. La aplicación debe permitir al usuario introducir sus datos en el formulario de registro.
  2. Opcionalmente, acceder a la cámara o galería para seleccionarla una imagen de perfil y mostrarla en el formulario.
  3. El sistema se encargará de validar los datos.
  4. El sistema mostrará un mensaje de error si alguno de los datos no cumple las validaciones especificadas para el formulario de registro.

5. En el caso de que la validación sea correcta, se habilitará el botón de registro y el sistema se encargará de guardar los datos del usuario en la base de datos.
6. El sistema mostrará una confirmación al usuario en el caso de que el registro se haya efectuado correctamente.
7. La aplicación redirigirá a la portada para que el usuario inicie sesión.
8. Se permite al usuario registrarse con su cuenta de Facebook redirigiéndolo a su aplicación nativa para autenticarse.

- **Iniciar sesión**
  1. Para iniciar sesión el usuario deberá identificarse con su email y contraseña correspondiente. En el caso de ya haber iniciado sesión anteriormente, el inicio de sesión será automático al iniciar la aplicación (siempre y cuando el usuario no haya cerrado su sesión).
  2. El sistema se encargará de validar y permitir o denegar el acceso a la aplicación.
  3. El sistema mostrará un mensaje de error en el caso de que la validación no sea correcta.
  4. En el caso de que la validación sea correcta, se mostrará un mensaje de bienvenida al usuario y se le redirigirá a la pantalla principal.
  5. Se permite al usuario iniciar sesión con su cuenta de Facebook redirigiéndolo a su aplicación nativa para autenticarse.
  6. Iniciar sesión a través de una cuenta de Google Plus.
  7. Iniciar sesión a través de una cuenta de Twitter.
  8. Si no se dispone de conexión a internet y existe una sesión abierta previamente, al iniciar la aplicación después de 15 segundos se mostrará un mensaje de 'Sin conexión'.

- **Solicitud de cambio de contraseña**
  1. Se permite al usuario cambiar la contraseña suministrando la dirección email vinculada.
  2. En el caso de ser una dirección email válida registrada, se notificará al usuario a través de un mensaje para que comprueba su correo electrónico.
  3. El sistema mostrará un mensaje de error en el caso de introducir una dirección email inválida.

- **Cierre de sesión**
  1. Cualquier usuario de la aplicación debe poder finalizar su sesión en la aplicación mediante un botón que indique "Cerrar sesión".
  2. En el caso de que el usuario pulse el botón de cerrar sesión, el sistema mostrará un mensaje para asegurarse de que el usuario realmente quiere cerrar sesión, permitiendo aceptar o cancelar.
  3. El usuario será redirigido a la portada sin estar logueado con un mensaje que le indique que ha cerrado su sesión.

- **Navegación aplicación**
  1. Visualizar la lista de canciones compartidas por usuarios a los que sigues.
  2. Añadir a '*like*' y guardar la canción en nuestra biblioteca de 'me gusta'.
  3. Deshacer '*like*' en una canción y eliminarla de la biblioteca de 'me gusta'.
  4. *Repostear* o compartir una pista de audio a tus seguidores.
  5. Deshacer el '*repost*' de una pista de audio.
  6. Comentar una pista concreta.
  7. Eliminar comentarios propios (deslizar hacia la izquierda).
  8. Crear listas de canciones públicas o privadas.
  9. Agregar canciones a una lista ya existente.
  10. Mostrar mensaje de error en caso de que se quiera agregar una canción privada a una lista pública.
  11. Compartir el título y audio de la canción a través de las redes sociales.
  12. Reproducir canción.
  13. Usar reproductor en segundo plano (*media notifications*).
  14. Ver historial de reproducción.
  15. Buscar pistas/usuarios.
  16. Visualizar lista de canciones que nos gustan.
  17. Ver perfiles de usuarios incluido el nuestro.
  18. Seguir a un usuario.
  19. Enviar notificación de seguimiento al usuario seguido.
  20. Dejar de seguir a un usuario.
  21. Ver lista de canciones subidas por el usuario en su perfil (sólo públicas).
  22. Las pistas privadas que hayamos subido sólo podrán ser visualizadas en nuestro perfil.
  23. Ver listas creadas por el usuario en su perfil (sólo públicas).
  24. Las listas privadas que hayamos creado sólo podrán ser visualizadas en nuestro perfil.
  25. Ver canciones dentro de cada lista.
  26. Ver usuarios seguidos/seguidores.
  27. Mostrar mensaje de aviso/confirmación al compartir una canción declarada como privada.
  28. Subir pista de audio a la nube.
  29. Cancelar progreso al subir pista de audio
  30. Eliminar pista de audio propia en cascada (borrar dependencias)
  31. Validar formato de la pista de audio.
  32. Ver tendencias según estilo de música.
  33. Ver notificaciones de quién ha comenzado a seguirte.
  34. Configurar las notificaciones emergentes.

Requisitos no funcionales

Requisitos complementarios o atributos de calidad. Especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento (requisitos funcionales).

- **Rendimiento**
  1. Una vez iniciada la aplicación y cargado los datos, se esperan tiempos de respuesta no superiores a un segundo al navegar por la aplicación y para peticiones a la base de datos, dependerá en gran medida de la cantidad de información que se solicite.
  2. Los cálculos que se realicen en la aplicación no suponen demasiada carga para el dispositivo por lo que, en este sentido, el rendimiento será óptimo.

  El tiempo de inicio se verá afectado, puesto que la velocidad al ejecutar la aplicación es uno de los requisitos que no se logra con aplicaciones híbridas. Éstas al iniciar levantan un navegador e inician la aplicación, por ello se demora a diferencia de las aplicaciones nativas.

- **Interfaz y usabilidad**
  1. La aplicación debe constar de una interfaz sencilla, atractiva e intuitiva. De tal forma que su uso no suponga un impedimento o esfuerzo al usuario a la hora de hacer uso de la aplicación.
  2. Se presentará un menú de navegación sencillo donde el usuario encuentre satisfechas sus consultas y cómodo su uso.
  3. Los datos se mostrarán estructurados para que el usuario encuentre fácilmente lo que desea.
  4. Se dará facilidad para leer y rellenar los formularios.

- **Mantenibilidad, disponibilidad y portabilidad**
  1. La aplicación estará capacitada para, bajo determinadas condiciones de uso, conservar o restaurar su estado operativo.
  2. Disponibilidad para dispositivos móviles (smartphones o tablets) con los sistemas operativos comunes (Android, iOS).
  3. Será necesario disponer de una conexión a internet, ya sea por Wifi o por tarifa de datos.

- **Seguridad**
  1. Para poder usar las funcionalidades que ofrece la aplicación es necesario autenticarse.
  2. Si el usuario no cierra su sesión, ésta se mantendrá abierta para futuros usos de la aplicación. De esta manera, el usuario no tendrá que volver a introducir sus credenciales.
  3. Al registrarse un usuario, la contraseña será cifrada.
  4. Al cambiar la contraseña, también se cifrará la nueva contraseña.

- **Escalabilidad y concurrencia**
  1. La aplicación se diseñará para que sea posible incluir nuevas funcionalidades sin perder calidad en los servicios ofrecidos.
  2. Los procesos a la hora de subir datos a la nube se ejecutarán concurrentemente.

2.3 DIAGRAMAS DE CASOS DE USO

Para los diagramas de casos de uso hemos utilizado la herramienta StarUML, y se han dividido en dos. El primero muestra los casos de uso para el usuario que no se encuentra autenticado en la aplicación. Mientras que, en el segundo, se exponen los casos de uso más importantes para un usuario autenticado.

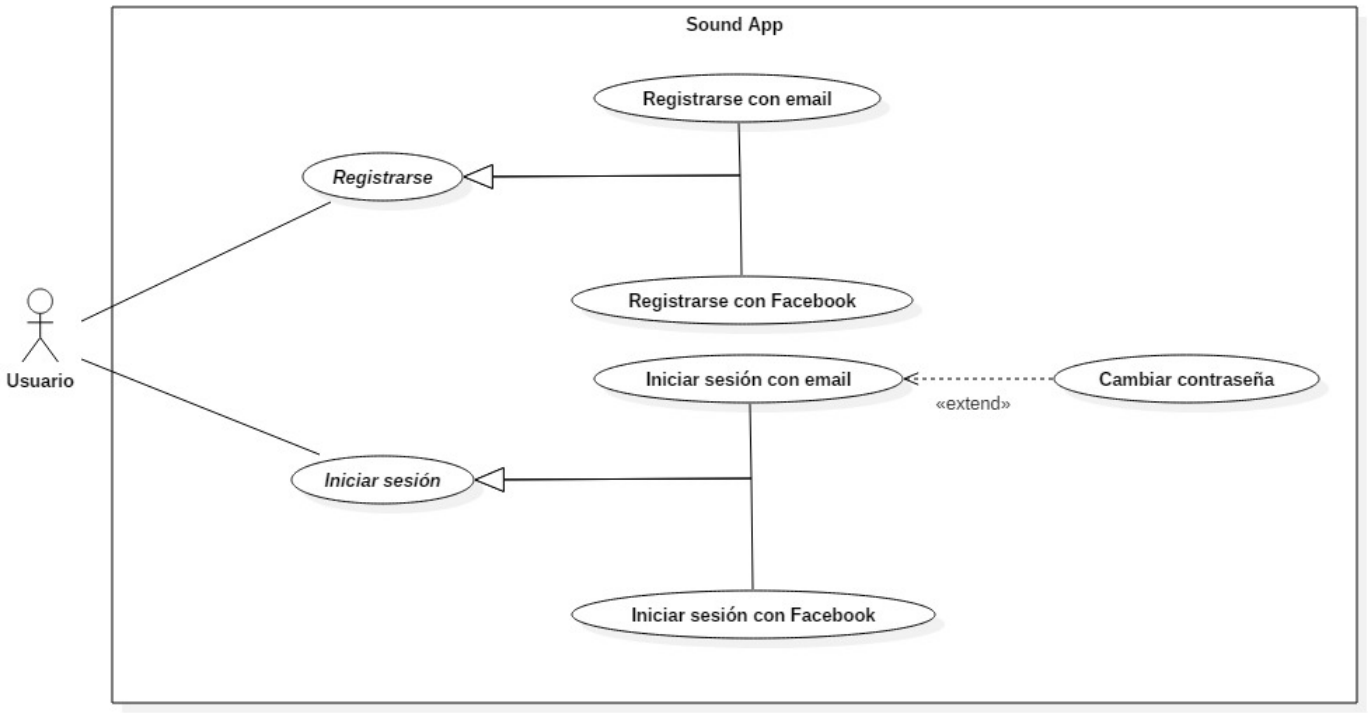


Diagrama 1 - Casos de uso para usuario sin autenticarse

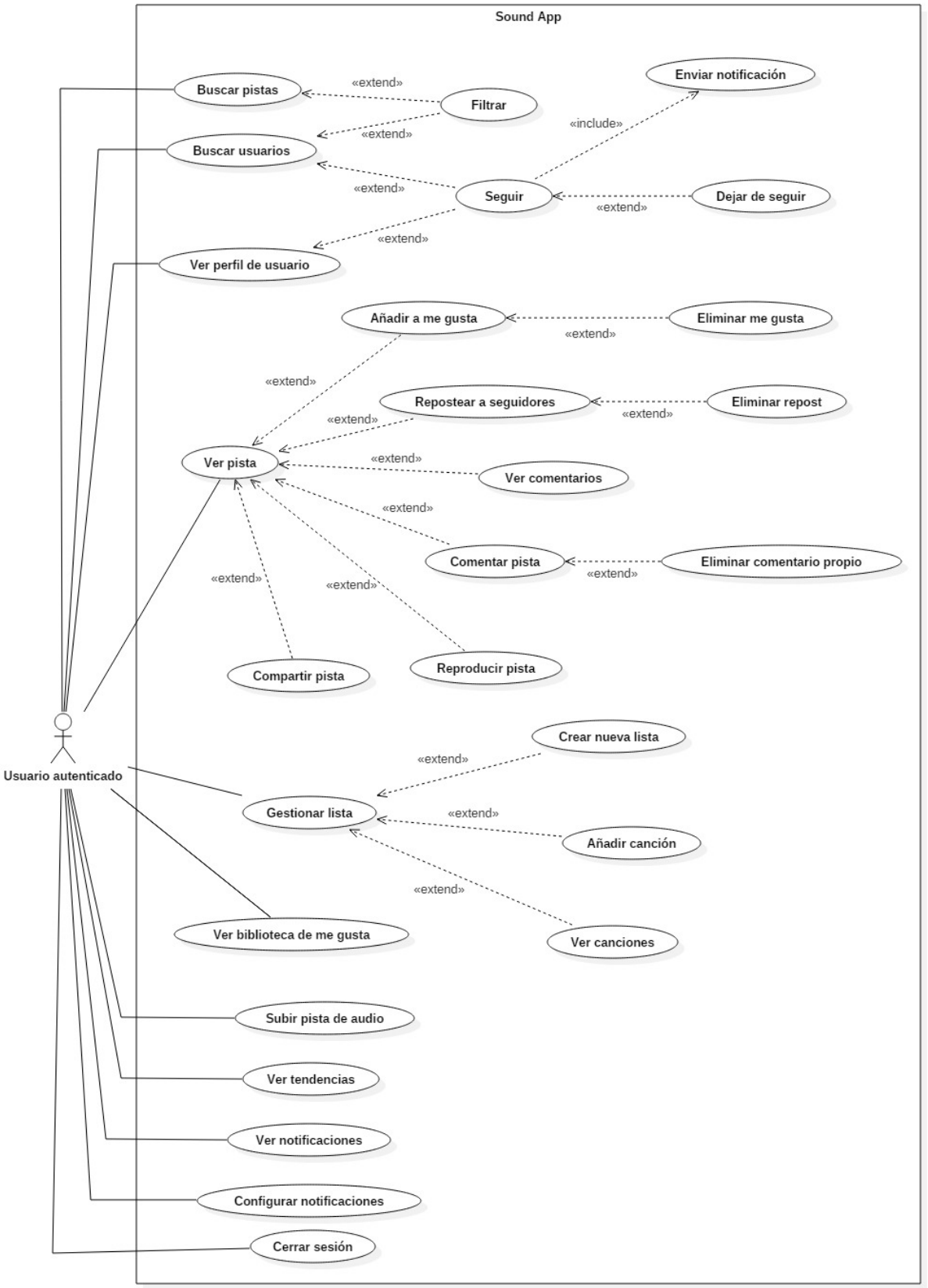


Diagrama 2 - Casos de uso para usuario autenticado

2.4 MOCKUPS, DISEÑO DE LA INTERFAZ Y PALETA DE COLORES

Para elaborar un prototipo de nuestra aplicación vamos a usar la herramienta Ionic Creator, de la cual hablaremos más adelante en el apartado 3.4.6 Ionic Creator.

Con ayuda de Ionic Creator, ha sido posible crear un prototipo de las principales vistas de la aplicación, que sirve de base para establecer los diferentes componentes. En cuanto a la paleta de colores, se ha optado por dejar una por defecto para los *mockups*. Tras observar diferentes combinaciones de colores en las aplicaciones semejantes del mercado y, más adelante, incluir imágenes de prueba, ha destacado la gran combinación de los colores fucsia y blanco, por lo que ha sido nuestra elección como colores base de la aplicación.

Aquí podemos probar la interfaz de prueba tanto en Android como en IPhone (se puede observar que la barra de navegación cambia de posición -superior e inferior- de uno a otro ya que los dispositivos Android suelen tener los botones de retroceder y demás en la pantalla).

A continuación, se pueden ver los *mockups* de las vistas más importantes.

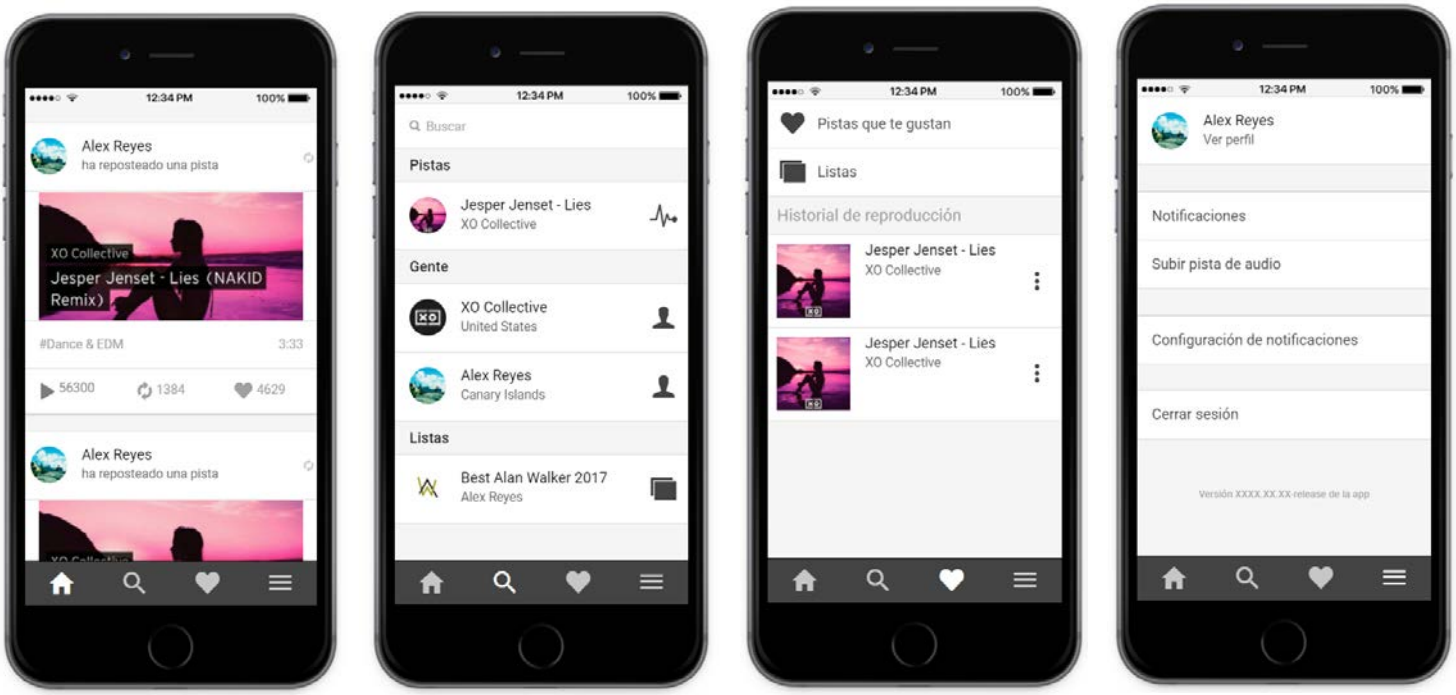


Ilustración 9 - Mockups Iphone

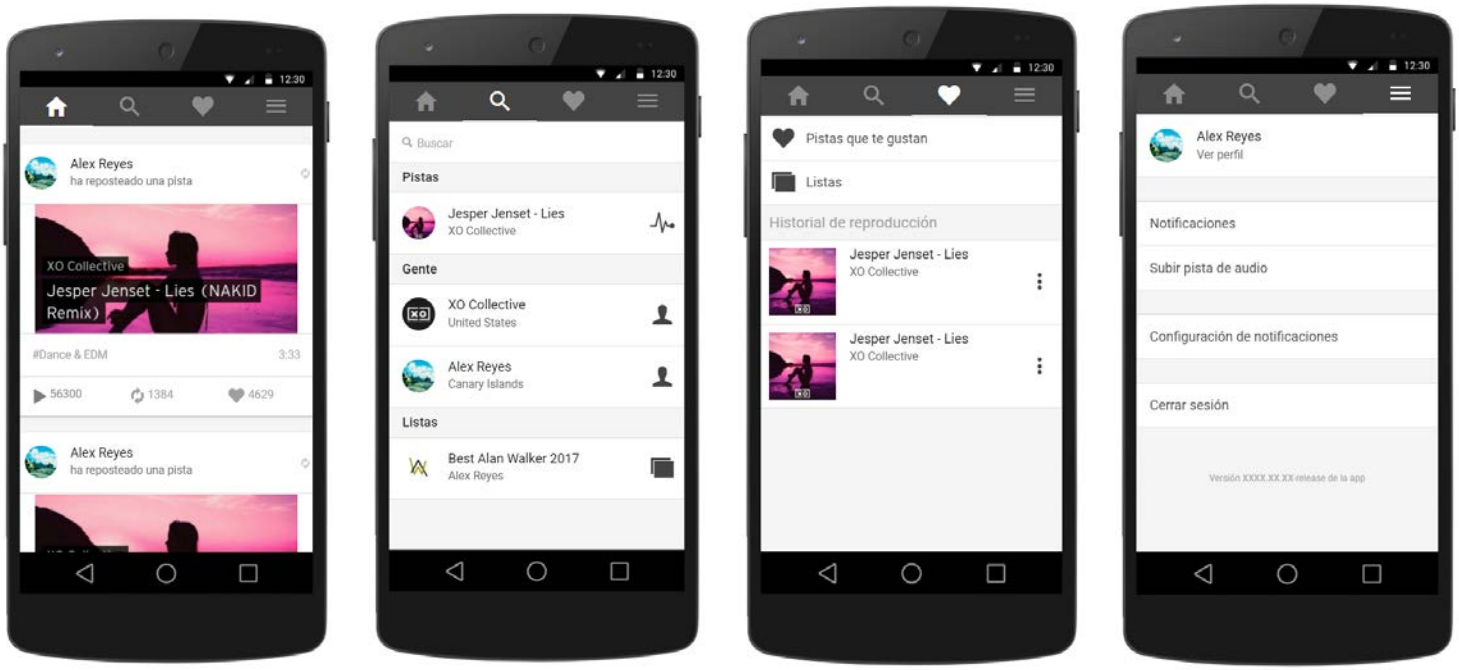


Ilustración 10 - Mockups Android

2.5 MODELO DE NEGOCIO

La rentabilidad de la aplicación no se espera que sea rápida. Se trata de una aplicación escalable que irá incrementando su número de usuarios a través del medio 'boca a boca'. Es decir, se espera que los usuarios la difundan al tratarse de una aplicación cómoda y limpia de anuncios.

En cuanto al modelo de negocio a seguir, se optará por el típico modelo 'Premium' que siguen todas las grandes compañías de este sector socio musical, exceptuando la incorporación de anuncios para los usuarios que optan por una cuenta gratuita, ya que puede llegar a desquiciarlo (experiencia propia).

Hoy en día, teniendo en cuenta el auge de los servicios de entretenimiento en *streaming* para películas, series, música y más recientemente libros, no es un mal modelo de negocio a seguir. Sin embargo, si analizamos la competencia, el modelo de negocio de Spotify es uno de los que más se ha ganado el favor del público incluso usando publicidad.

En Spotify existen dos opciones: usar una cuenta gratuita soportada con publicidad o pagar una suscripción mensual o anual a cambio de eliminar la publicidad y conseguir alguna característica adicional.

Desde el punto de vista del usuario, este servicio es muy cómodo para los menos exigentes. Proporciona una manera legal de escuchar música nueva a cambio de escuchar anuncios cada cierto tiempo o tener acceso sin publicidad mediante una cuenta de pago.

Como principal ventaja para el mundo de la música, elimina algún porcentaje de piratería en las redes P2P o en páginas web de dudosa reputación. Sin embargo, algunos artistas se quejan de la poca cantidad de ingresos que reciben de la compañía en comparación con las ventas de álbumes físicos.

Por lo tanto, podemos dividir el modelo de negocio de Spotify -generalmente el de todas las compañías comunes- en dos principales vías para recaudar dinero:

- Anuncios para los usuarios con cuentas gratuitas.
- Cuotas mensuales o anuales de los usuarios de pago.

Según fuentes indicadas en la bibliografía del presente trabajo, el modelo de negocio de Spotify reparte los *royalties* (pagos al poseedor de derechos de autor) a sus artistas siguiendo el siguiente esquema:

- Se calculan los ingresos mensuales totales.
- Después, se multiplica por el cociente del número de *streamings* del artista entre el total de *streamings*.
- Se le extrae el 70% de los *royalties* que van a parar a las discográficas propietarias de la música.
- Se aplica el porcentaje de *royalty* directo para el artista.

Esquema 2 - Reparto de royalties de Spotify [6]

[6] Reparto de royalties de Spotify - Julio 2017

Como resultado tenemos que el artista se lleva entre 0,006 y 0,0084 dólares por cada reproducción en *streaming* (al menos 30 segundos). Esta cifra ha generado disputas entre los artistas y el modelo de negocio de Spotify.

Si echamos cuentas, un artista que generase 100.000 reproducciones en un mes de una canción se embolsaría entre 600 y 840 dólares.

Según diversos estudios, declaraciones y alguna que otra filtración, el 30% de los ingresos totales de Spotify es para la propia empresa y a partir de ahí, entraría en marcha el sistema de pagos a los artistas.

El modelo de negocio de la empresa sueca no es una fórmula única en todo el mundo ni es sencilla de comprender. Algunos especialistas han podido hacer algunos cálculos dependiendo de varios factores, tales como el tiempo que se escucha un tema o el número de reproducciones, pasando por la relevancia del artista, el país de origen del usuario, si éste es premium o free. Según la empresa, en el 2013 pagó unos 3.300 dólares al mes por un 'disco indie', unos 145.000 dólares para un 'disco del top 10' y unos 425.000 dólares por un 'disco de éxito mundial'.

Lamentablemente en la actualidad grandes discográficas y sellos cobran en promedio un 50% de lo que paga Spotify, lo que significaría que el artista recibiría 0,001606 dólares por reproducción.

2.6 NORMATIVA Y LEGISLACIÓN

Para desarrollar aplicaciones móviles, uno de los aspectos para tener en cuenta son los legales. De esta forma, evitamos sanciones y protegemos nuestra aplicación y trabajo como programadores.

En la actualidad, el mercado de las aplicaciones móviles parece que presta menos atención a dichos aspectos legales. La mayoría de los usuarios buscan e instalan aplicaciones gratuitas renunciando a la privacidad sin pararse unos minutos a leer lo que están aceptando.

La instalación de las aplicaciones en nuestros dispositivos móviles puede llegar a recoger una gran cantidad de datos de carácter personal. Por este motivo, en febrero del 2013, las autoridades europeas de protección de datos aprobaron el primer dictamen que clarifica el marco jurídico aplicable al uso de aplicaciones para dispositivos inteligentes, denominado 'Dictamen 2/2013 del Grupo de Trabajo del artículo 29', concluyendo que resulta de plena aplicación la normativa de protección de datos (**LOPD**) para los desarrolladores de aplicaciones.

A continuación, expondremos los diferentes aspectos para tener en cuenta para respetar la normativa y legislación vigente.

Funcionalidades

Dentro de las funcionalidades de nuestra aplicación, hay que reflexionar cuáles podemos desarrollar y cuáles no. Siempre usando medios lícitos, por lo que debemos tener claro que lo que no se puede hacer offline o través de campañas de marketing tradicional, no se podrá hacer desde nuestra aplicación.

Derechos propios y de terceros

Debemos contar con las respectivas licencias de los recursos que utilicemos, ya sean librerías de programación, bases de datos, elementos gráficos, canciones, textos, imágenes. En nuestro caso, debemos hacer hincapié en este aspecto al usar algunas canciones e imágenes para probar nuestra aplicación. También es recomendable leer las condiciones de uso para evitar problemas, ya que en algunas ocasiones (no es el caso para un trabajo de fin de grado), esos recursos excluyen el uso comercial y no podríamos utilizarlos en el desarrollo de aplicaciones. Para terminar, es recomendable proteger tu trabajo una vez hayas finalizado de desarrollar la aplicación para evitar plagios, copias.

Licencia y condiciones de uso

Como hemos señalado anteriormente, debemos añadir condiciones y licencias de uso que el usuario debe aceptar para poder hacer uso de la aplicación. En las condiciones legales deberemos hacer una adecuación a la normativa y, además, poder eximirnos de las responsabilidades que podamos, para así evitar que nos reclamen por el mal uso que se hagan de ellas. Si su adecuación a la realidad de la aplicación y su ajuste a la legislación vigente son los correctos, será la mejor defensa posible en caso de cualquier reclamación. Por ello, la aceptación previa por el usuario es imprescindible.

Información y permisos

Para este apartado, al tratarse de aplicaciones que se instalan y ejecutan en dispositivos móviles, resulta bastante importante ser claros y explícitos al solicitar permisos al usuario. Normalmente, cualquier aplicación móvil va a necesitar acceder a los contactos de la agenda o a contenidos del móvil, ya sea por cesión de datos, compartir contenidos. Más adelante, en el apartado 3.4.4 Ionic Native y Apache Cordova/PhoneGap, hablaremos en profundidad sobre estos permisos a la hora de iniciar nuestra aplicación por primera vez.

Política de Cookies

La necesidad de aceptación de las cookies es tan importante en páginas web como en dispositivos móviles a la hora de descargar aplicaciones. Se deberá establecer un aviso informativo con la información básica sobre qué son las cookies y la finalidad de éstas. Esta breve información redirigirá al usuario a la información completa con todos los aspectos que exige la ley de política de cookies.

Informar al usuario

En general, una gran parte de las aplicaciones móviles pueden ser consideradas como “servicios de la sociedad de la información”. Por ello, hay que cumplir con las obligaciones que la legislación implica para estos servicios. La principal obligación es la de informar a los usuarios de los aspectos marcados por la ley, que se hace comúnmente a través de textos en las secciones sobre las condiciones legales.

Estos apartados proveen al usuario de información de derechos de autor, condiciones de uso, política de privacidad. Además, se incluyen aspectos como información de la empresa, de contacto, sucursales, exención de responsabilidad mencionada anteriormente.

Seguridad

La seguridad en las aplicaciones móviles es un tema prioritario, puesto que la mayor parte de ellas guardan información personal de sus usuarios. Uno de los aspectos para tener en cuenta es el cifrado de las contraseñas. En nuestro caso, esta labor está delegada a Firebase que gestiona las cuentas de usuario.

Publicidad

La monetización de la mayoría de las aplicaciones gratuitas puede provenir de distintas técnicas, algunas más lucrativas que otras. Siempre es recomendable escoger un sistema en función de las utilidades y los tipos de aplicaciones móviles.

Sin embargo, por regla general está cada vez más extendido el uso de publicidad para generar los ingresos. Aunque es totalmente lícito que una aplicación incluya publicidad, ésta deberá aparecer siempre identificada como tal para evitar posibles problemas.

3 Diseño e implementación

En el presente capítulo se describe todo el proceso de diseño de la aplicación. Se ha realizado un diseño que abarca todos los requisitos descritos en el apartado 2.2 Requisitos. El diseño proporciona una idea completa del software desarrollado en el proyecto. Para ello, dividiremos esta sección en:

3.1 Alcance de la implementación.

3.2 Diseño arquitectónico.

3.3 Modelo de la base de datos.

En cuanto a la fase de implementación del diseño, durante el desarrollo del proyecto hemos usado el sistema operativo Ubuntu 16.04 LTS, basado en GNU/Linux y distribuido como software libre. Como editor de código fuente se ha utilizado Visual Studio Code, ligero pero potente que se ejecuta en el escritorio y está disponible para Windows, macOS y Linux. Además, viene con soporte incorporado para Node.js, JavaScript, TypeScript.

3.1 ALCANCE DE LA IMPLEMENTACIÓN

Se ha desarrollado un prototipo de aplicación cuya funcionalidad es validar las principales ideas que hemos especificado en el análisis y diseño expuesto anteriormente. Con el objetivo de adquirir los conocimientos básicos sobre las tecnologías que hemos estado usando, de las cuales hablaremos más adelante en la sección denominada 3.4 Tecnologías.

Dentro de las características implementadas podemos destacar las siguientes, que cumplen con la mayoría de requisitos especificados previamente.

Al iniciar la aplicación nos encontramos con una página de inicio en la que podemos registrarnos o iniciar sesión con una cuenta creada previamente a través de un pequeño formulario protegido con una serie de validaciones. También, existe la posibilidad de registrarnos/iniciar sesión con Facebook de una forma más rápida.

En el caso de que el usuario vaya a iniciar sesión con su cuenta y no recuerde la contraseña, puede solicitar el cambio de ésta introduciendo su correo electrónico en el apartado '¿Has olvidado tu contraseña?'. Posteriormente, será notificado a través de su cliente de email con las instrucciones correspondientes para realizar el cambio de contraseña.

La aplicación está desarrollada para que, si el usuario la cierra en cualquier momento y no ha cerrado su sesión, se guardará dicha sesión al iniciar de nuevo la aplicación para así evitar volver a introducir de nuevo las credenciales.

Una vez dentro de la aplicación, el usuario podrá buscar en el menú correspondiente (intuitivo gracias a los iconos incluidos), artistas y/o canciones que otros usuarios han publicado. De cualquier modo, podemos filtrar para buscar canciones o usuarios. En el caso de que ya conozcamos al artista, podemos seguirlo de forma rápida en el mismo buscador o visitar su perfil y escuchar algunas canciones publicadas por él mismo. Al seguir a un usuario, enviaremos una notificación que se le mostrará al usuario en cuestión si éste tiene la sesión activa. Como hemos indicado antes, también es posible buscar canciones para ver su autor y poder añadirlo a la lista de seguidos. El objetivo de seguir a distintos usuarios/artistas es poder ver de forma más amena sus publicaciones compartidas en nuestro 'Home'. Entiéndase como 'publicación compartida' cualquier canción que haya sido subida como pública y el usuario al que seguimos la ha compartido, no son necesariamente canciones subidas por el mismo autor. De esta forma, favorecemos la distribución de música y aumentamos exponencialmente la rapidez con la que llega a los usuarios registrados en nuestra aplicación. Posteriormente, hablaremos sobre este mecanismo denominado '*reposts*'.

Para apoyar las canciones disponemos del clásico botón de 'me gusta', el cual cumple dos funciones principales en nuestra aplicación:

- Añadir la canción a nuestra lista de 'me gusta' para poder escucharla en cualquier otro momento de manera más directa y ordenada.

- Ayudar a los usuarios artistas que se dedican al mundo de la música a tener una ligera idea de las tendencias musicales en auge.

Al lado del botón '*like*' tenemos el botón '*repost*' que hemos mencionado antes. La función de este botón es compartir la canción a tus seguidores para que la vean automáticamente en su directorio principal 'Home'. A su vez, este usuario seguidor podrá compartirla a sus seguidores y así sucesivamente. Para el caso en el que hayamos subido una canción como privada y queramos luego compartirla, se nos avisará para convertirla en pública.

Otra característica habitual es la de poder realizar comentarios. Para cada canción pública es posible ver los comentarios de los usuarios ordenados por fecha y añadir comentarios propios. En caso de querer eliminar cualquier comentario propio, es posible hacerlo simplemente deslizando el dedo en la pantalla hacia la izquierda para mostrar la opción de eliminar. Conjuntamente, desde aquí podemos navegar a la vista de perfil de cualquier usuario que haya comentado.

Luego, tenemos la opción de añadir la canción a una lista creada previamente o crear una nueva con esa canción. Esta funcionalidad es útil para aquellos usuarios que quieran mantener su biblioteca ordenada según el estilo de música, autor, … Por otro lado, las pistas se ordenarán dentro de cada lista por antigüedad desde la más antigua hasta la más nueva. Y, además, existe la posibilidad de crear listas privadas para nuestro uso y disfrute. De cualquier forma, como es lógico, la aplicación no nos dejará incluir una canción privada a una lista que haya sido creada como pública.

Además, es posible compartir el título y audio de una canción a través de las redes sociales. Al no disponer de un dominio para la aplicación, el enlace compartido redirigirá al usuario a donde está almacenado el audio y así al menos poder escucharlo.

La reproducción de la canción está disponible en la vista amplificada de la pista, donde podemos realizar cualquiera de las acciones expuestas previamente y, también, visitar el perfil público del usuario que ha subido dicha canción.

En los perfiles, incluido el nuestro, podemos ver la foto de perfil o una por defecto junto con la información básica de perfil y, justo al lado, el botón de 'Seguir' o 'Siguiendo'. Posteriormente, podemos ver las canciones propias que hemos o ha subido el usuario en cuestión y, justo al lado, disponemos de las listas creadas tanto públicas como privadas. Lógicamente, las listas privadas solo podrán ser vistas por el usuario que las creó al igual que las canciones. Finalmente, podemos ver los usuarios seguidos y los seguidores de forma que podemos navegar indefinidamente entre perfiles. Sin embargo, suponiendo que el usuario quiera volver al menú principal, dispone de un botón para retroceder automáticamente.

Como característica destacada y fundamental, tenemos el formulario para subir las canciones desde nuestro móvil a la base de datos en la nube que estamos usando para desarrollar la aplicación. Para empezar, disponemos de la opción de establecer una portada para nuestra publicación o dejar la que la aplicación nos ofrece por defecto. Si decidimos añadir una imagen nueva, podemos seleccionar una existente desde la galería o abrir la cámara del móvil para efectuar una actual. De cualquier modo, podemos previsualizarla en el mismo formulario. Por consiguiente, será necesario añadir el archivo de audio guardado en el dispositivo móvil. Para ello, primero tendremos que aceptar los permisos oportunos para poder acceder a los repositorios/carpetas del sistema operativo del smartphone. Una vez seleccionado el archivo de audio, el sistema validará su formato y, en caso de ser aceptado, veremos la ruta específica en el formulario para poder verificar que se ha seleccionado una canción. Sólo nos queda añadir un título de canción, una etiqueta de las preseleccionadas que identifique el tipo de música y si queremos que la canción sea pública (compartirla a nuestros seguidores) o privada. Con todo esto y las validaciones pertinentes, se habilitará el botón para subir la pista a la base de datos. Esto puede tardar varios segundos o incluso minutos, dependiendo en gran medida de la velocidad de nuestra red y del tamaño del archivo de audio ya que, en caso de subir alguna imagen, ésta la habremos redimensionado a un tamaño

práctico y agradable visualmente que no afecte al rendimiento de la aplicación. Un indicador de progreso nos mantendrá informados de la subida pertinente al audio y a la portada en su caso.

En cuanto a las tendencias de las que hablamos en el apartado del botón 'me gusta', podemos ver un gráfico circular en el que se encuentran los porcentajes de 'me gusta' de cada etiqueta de estilo de música disponible en la aplicación. El objetivo de este gráfico es ilustrar a los artistas cómo van variando los gustos musicales, y tomen nota de ello para sus posteriores producciones. De esta forma, es posible observar qué tipo de música se demanda más, tomando como muestra a los usuarios registrados en la aplicación.

También, podemos ver un listado de las notificaciones recibidas por los usuarios que nos han seguido y acceder al perfil. Las notificaciones emergentes las podemos activar/desactivar en la sección de configuración de notificaciones. Una vez guardados los cambios, se nos pedirá que reiniciemos la sesión para establecer la nueva configuración.

Por último, el usuario podrá cerrar su sesión y confirmar dicha acción a través de un mensaje de alerta.

3.2 DISEÑO ARQUITECTÓNICO

Ilustración 11 - Lenguajes que forman la arquitectura de Angular [7]

El diseño arquitectónico de nuestra aplicación sigue un modelo basado en componentes. Esto quiere decir que la aplicación estará compuesta por un árbol de componentes que se utilizan los unos a los otros para la obtención de los objetivos globales que se quieren alcanzar.

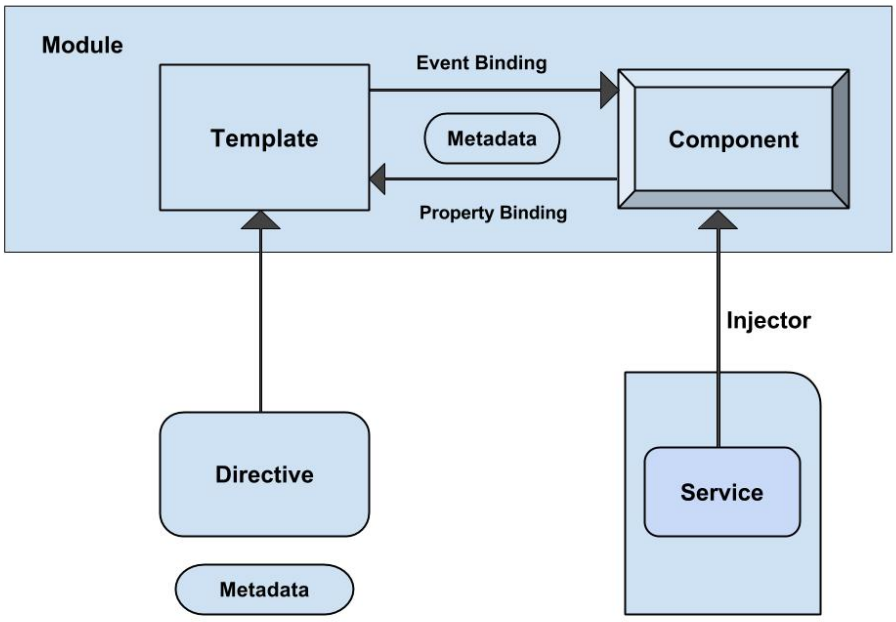
La mayoría de *frameworks* modernos usan esta arquitectura, que se ha demostrado que es la que ofrece un mejor código, más fácilmente escalable y mantenible.

[7] Lenguajes que forman la arquitectura de Angular - Julio 2017

Los componentes están pensados para, de manera modular y encapsulada, resolver pequeños problemas. Por ejemplo, puede haber componentes para implementar un sencillo botón, para selectores de fechas o en nuestro caso, componentes para hacer un sistema de navegación por *tabs*, una barra de progreso, etc. Integrando componentes somos capaces de construir aplicaciones grandes y complejas. El *framework* que estamos usando, Ionic 2 nos ofrece ya de base una cantidad muy grande de componentes que son capaces de trabajar perfectamente en dispositivos móviles con pantallas táctiles, pero obviamente para el desarrollo de las aplicaciones necesitaremos construir nuestros propios componentes que implementen los comportamientos más específicos de nuestro análisis.

Ionic 2 está desarrollado sobre el *framework* de JavaScript Angular. Lo que quiere decir que para el desarrollo con Ionic 2 podemos apoyarnos en todas las ventajas de desarrollo con Angular, lo que nos permitirá contar con una excelente estructura de proyecto, el trabajo con buenas prácticas, uso de patrones de diseño de software variados, principalmente MVC (Model View Controller), y por supuesto como se ha indicado antes una buena gama de componentes y directivas. De esta manera, usando Angular podemos aprovechar muchas utilidades pensadas para un desarrollo más rápido y, además, tenemos la certeza de disponer de un código más optimizado y de mayor rendimiento, adaptado para el momento actual y el futuro.

El siguiente diagrama muestra la arquitectura de Angular:



Esquema 3 - Elementos que conforman la arquitectura de Angular [8]

[8] Elementos que conforman la arquitectura de Angular - Julio 2017

A continuación, enunciaremos y describiremos las principales partes que conforman esta arquitectura:

3.2.1 Módulos

Las aplicaciones en Angular son modulares y se conocen como 'NgModules'. Cada aplicación Angular tiene al menos un módulo, que es el módulo raíz, denominado 'AppModule'. La mayoría de las aplicaciones tienen varios módulos, cada uno es un bloque dedicado a una tarea concreta. En nuestro caso, hemos partido del módulo raíz sin crear módulos nuevos.

Un módulo es una clase con el decorador @NgModule. Los decoradores son funciones que modifican las clases JavaScript. Angular posee varios decoradores que adjuntan metadatos a las clases. Cada módulo deberá estar declarado en el fichero 'app.module.ts' bajo el decorador @NgModule.

Las propiedades más importantes de 'NgModule' son:

- Declarations: Las clases de vista que pertenecen a este módulo. Angular tiene tres tipos de clases de vista: componentes, directivas y pipes.
- Imports: Otros módulos cuyas clases exportadas son necesarias por los componentes de las plantillas declaradas en este módulo.
- Bootstrap: La vista principal de la aplicación, denominada componente raíz, que aloja todos los otros puntos de vista. Sólo el módulo raíz debe establecer esta propiedad.
- EntryComponents: Para componentes añadidos dinámicamente. Al añadirlos, Angular comunica al compilador de plantillas sin conexión que los compile.
- Providers: Creadores de los servicios que este módulo utiliza (son accesibles en toda la aplicación).

3.2.2 Componentes

Un componente controla una parte de lo que se muestra en la pantalla denominado vista. El componente interactúa con la vista a través de una serie de propiedades y métodos, en los que se define la lógica de la aplicación.

Angular es capaz de crear, actualizar y destruir componentes a medida que el usuario navega por la aplicación. La actuación se puede dar en determinados momentos del ciclo de vida como, por ejemplo, al inicializar la vista.

En nuestra aplicación, existen numerosos componentes que interactúan entre sí para el correcto funcionamiento de la misma. A continuación, explicaremos uno de los componentes que conforman la aplicación y que hemos creado desde cero.

El componente denominado 'ProgressBar', lo hemos creado desde cero para aprender cómo se desarrolla un componente personalizado y adaptarlo a la funcionalidad que queremos.

Básicamente, se trata de una barra de progreso que nos indica el estado de subida de un archivo a nuestra base de datos en la nube de Firebase. Cuando subimos un archivo (audio y/o imagen) guardamos el estado actual en una variable que almacena el porcentaje de subida. Este porcentaje lo calculamos a través de un *snapshot* (instantánea) que incluye los bytes subidos y el total de bytes que se pretenden subir.

De esta manera, capturamos el estado cambiante de esa variable y se la pasamos como parámetro de entrada al nuevo componente. El componente tiene su propia plantilla HTML que simplemente muestra una vista con una barra cuya longitud cambia conforme varía la variable captada como parámetro de entrada.

Usando el selector que hemos declarado en el componente, podemos usar esta barra de progreso en cualquier lugar de nuestra de aplicación de una forma escalable y siguiendo el principio fundamental de desarrollo de software **DRY (Don't Repeat Yourself)**.

El resto de componentes se limitan a recoger información de la base de datos, transformarla y mostrarla al usuario para que éste pueda visualizarla e interactuar con ella. De cualquier modo, también se encargan de enviar información a la base de datos para registrar nuevos usuarios o para verificar datos, como puede ser a la hora de iniciar sesión.

3.2.3 Plantillas

Las plantillas son fragmentos de HTML con etiquetas especiales de Angular e Ionic 2. Estas plantillas indicarán cómo se debe representar el componente. Se trata de código HTML al que se le ha añadido determinadas etiquetas y atributos especiales, tanto de Angular como de Ionic en nuestro caso. Es posible añadir etiquetas que representan un componente distinto. Como, por ejemplo, el componente 'ProgressBar' bajo el directorio 'components' descrito previamente, trabaja sobre sus propios datos de entrada y tiene su propia plantilla HTML.

3.2.4 Metadatos

Los metadatos contienen la información que Angular necesita para procesar una clase. Para adjuntar metadatos a una clase es necesario usar el decorador @Component y pasarle un objeto que lo configure. El decorador @Component irá encima de la clase exportada y como mínimo le tendremos que pasar el nombre que recibirá el selector y qué plantilla debe utilizar. La plantilla, los metadatos y los componentes describen una vista/módulo como se puede observar en la figura al comienzo de esta sección.

3.2.5 Data binding

El 'data binding' es el enlace de la información que tenemos con lo que se mostrará en la plantilla HTML. Angular es capaz de proporcionarnos una forma sencilla y muy limpia de tratar este enlace de información. Hay cuatro formas en las que se produce el 'data binding':

- Desde el componente hacia el DOM (Modelo de Objetos del Documento) mediante {{ valor }} denominado interpolación. En este caso se está recuperando directamente el valor de una variable que existe en el componente.

- Desde el componente hacia el DOM mediante [propiedad] = "valor" denominado 'property binding'. Igual que el anterior, pero ahora se recupera como un atributo de una etiqueta.

- Desde el DOM hacia el componente mediante (evento) = "manipulador" denominado 'event binding'. En este caso se puede llamar a un método del componente a través de un evento determinado y pasar un valor.

- En ambos sentidos con [(ngModel)] = "propiedad", denominado 'Two-way data binding' o coloquialmente 'banana in a box'. Esto asocia una propiedad y un evento de una sola vez. Cualquier cambio que haga el usuario se pasará al componente y viceversa.

El 'data binding' es importante para la comunicación entre la plantilla y el componente, y también es importante para la comunicación entre los componentes principales y los secundarios como usamos en 'ProgressBar'.

3.2.6 Directivas

Las plantillas de Angular son dinámicas. El DOM se va transformando de acuerdo con las instrucciones pasadas en las directivas. De entre los principales tipos de directivas tenemos:

- Estructurales: Son directivas que alteran el diseño mediante la creación, eliminación y sustitución de elementos en el DOM. Por ejemplo, tenemos las más habituales que son la directiva *ngFor (lógica de repetición) y la directiva *ngIf (lógica de condición).

- Atributos: Son directivas que alteran la apariencia o el comportamiento de un elemento existente. En las plantillas se ven como atributos HTML normales de ahí su nombre. Ionic 2 proporciona un conjunto de atributos que se pueden utilizar en cualquier elemento para modificar el texto, ajustar el *padding* (relleno) o el margen, etc.

3.2.7 Servicios

Un servicio es una categoría amplia que abarca cualquier valor, función o característica que necesite la aplicación. Es típicamente una clase con un propósito bien definido, es decir, debe hacer algo específico.

Los servicios son fundamentales para cualquier aplicación Angular, ya que mantienen organizado el código, favorecen su mantenibilidad y testeo. Un caso bastante habitual de uso de los servicios, y que usamos nosotros en nuestra aplicación, es para la obtención de datos del servidor donde se encuentra alojada la base de datos o para administrar registros de nuevos usuarios, controlar las sesiones activas, etc.

Otro posible uso que le damos es para precargar determinados sonidos al iniciar la aplicación y reproducirlos sin retraso alguno. Por ejemplo, al pulsar el botón de 'me gusta'.

3.2.8 Inyección de dependencias

La inyección de dependencias es una manera de proporcionar una nueva instancia de una clase con las dependencias que requiere. La mayoría de las dependencias son servicios.

Angular decide qué servicios necesita un componente examinando los parámetros del constructor. Para que Angular pueda crear dicho componente, pedirá un inyector para los servicios que está requiriendo con su posterior importación.

3.3 MODELO DE LA BASE DE DATOS

Nuestra base de datos no sigue un diagrama del tipo entidad-relación. Se trata de un modelo de datos jerarquizado tipo JSON, al que podremos acceder de forma remota. La base de datos en tiempo real de Firebase (*Firebase Realtime Database*) cuenta con la capacidad de almacenar datos en la nube, uno de los requerimientos de los que pocas aplicaciones actuales pueden escapar.

Ilustración 12 - Logo Firebase [9]

3.3.1 Realtime Database

Firebase nos proporciona un servicio de base de datos con la particularidad de ser en tiempo real. ¿Pero qué significa esto? La sincronización en tiempo real implica que cualquier cambio realizado en los datos por cualquier cliente se sincronizarán automáticamente y de forma inmediata (siempre que la conexión lo permita) en el resto de clientes, sin necesidad de que éstos vuelvan a consultar los datos. ¿Y si se pierde temporalmente la conexión? No hay problema, Firebase está preparado para permitir interactuar con la base de datos cuando el dispositivo no tiene conexión (siempre dentro de unos límites) mediante un sistema de cachés y colas de escritura locales. Cuando el dispositivo vuelve a tener conexión, los cambios locales serán sincronizados automáticamente con la base de datos y con el resto de clientes conectados a ella.

En Firebase, los datos estarán estructurados de forma similar a como se organiza la información en un fichero JSON, es decir, en forma de árbol donde cada nodo puede contener un valor o bien contener nodos hijo, que a su vez podrán contener valores o tener nuevos nodos hijo (hasta un máximo de 32 niveles de anidación). Éste es un aspecto importante para tener en cuenta, Firebase no nos ofrece una base de datos SQL tradicional con sus tablas y sus registros perfectamente estructurados.

[9] Logo Firebase - Julio 2017

Para establecer relaciones entre los diferentes nodos que conforman la aplicación será necesario identificar cada nodo a través de un UID (identificador único). Este identificador se genera automáticamente para cada elemento nuevo con la posibilidad de que varios clientes puedan agregar campos secundarios a la misma ubicación al mismo tiempo sin conflictos de escritura. Este UID generado por la método *push()*, el cual se encarga de agregar nuevos datos a la referencia que hayamos indicado a Firebase, se basa en una marca de tiempo. Por lo tanto, los elementos se ordenan cronológicamente de forma automática.

A través de estos identificadores únicos, es posible relacionar nodos de tal manera que un campo de un elemento contenga como valor el identificador del nodo con el que se quiere crear la relación. Para ilustrar este comportamiento, vamos a fijarnos en una parte de nuestro modelo JSON.

Ilustración 13 - Estructura JSON para los comentarios de las canciones

Cuando un usuario sube una pista de audio a Firebase, se crea como hemos indicado previamente un UID único para esa canción. Al crearse la pista, no tiene comentarios por lo que no debe aparecer bajo el nodo 'comments'. Sin embargo, una vez subida puede ser comentada por varios usuarios y puede tener varios comentarios del mismo usuario. Cuando por primera vez un usuario comenta la canción, se crea bajo el nodo 'comments' un identificador único, pero no cualquiera, el identificador creado cuando se subió la canción, o lo que es lo mismo el UID de la canción que se comenta. Justo debajo como nodo hijo aparece otro UID diferente, este identificador es nuevo y corresponde al comentario/nodo que se acaba de publicar junto con los campos pertinentes. Destacamos el campo 'user_id' que nos indica el UID del usuario que publicó el comentario.

Así es como establecemos las relaciones que dominan la aplicación y que sin ellas no podríamos mantener la coherencia de los datos que se muestran al usuario. A través de la redundancia conseguimos una relación bidireccional permitiendo obtener datos de manera rápida y eficiente, incluso cuando la lista escala a millones.

En cuanto al resto de nodos, siguen una estructura parecida a la anterior.

- **Listas:** La primera vez que un usuario crea una lista, se añade el identificador de usuario bajo el nodo 'lists' y, justo debajo de éste, el nuevo UUID que identifica la lista creada. Luego, tenemos los campos pertinentes junto con los UUIDs de las canciones agregadas a la lista y, además, el instante en el que se ha agregado cada una para poder ordenarlas cuando se muestren al usuario.

Ilustración 14 - Estructura JSON para las listas creadas por los usuarios

- **Canciones:** Al subir canciones, se genera un UUID junto con todos los campos especificados en la siguiente imagen. Destacamos la relación entre canción-artista a través del campo 'artist_id', que hace referencia al UUID del artista que ha subido la canción

Ilustración 15 - Estructura JSON para las canciones subidas por los usuarios

- **Usuarios:** Esta estructura es la más amplia, pero no quiere decir que sea más compleja que las demás. Las relaciones se establecen de la misma manera que las anteriores.

Ilustración 16 - Estructura JSON para los usuarios

3.3.2 Autenticación

En cuanto a la autenticación en Firebase, la mayoría de las aplicaciones deben reconocer la identidad de un usuario. El reconocimiento de la identidad de un usuario permite a una aplicación guardar datos de éste de manera segura en la nube y brindar la misma experiencia personalizada en todos sus dispositivos.

Firebase Authentication proporciona servicios de *backend*, SDK fáciles de usar y bibliotecas ya hechas para autenticar usuarios en la aplicación. Admite autenticación con contraseña, proveedores de identidades federadas populares como Google, Facebook, Twitter y más opciones.

Firebase Authentication se integra estrechamente con otros servicios de Firebase y aprovecha estándares industriales como 'OAuth 2.0'. Éste es un protocolo de autorización que permite que los usuarios autoricen a terceros para acceder a su información. Por lo tanto, se puede integrar fácilmente con nuestro *backend* personalizado.

Para iniciar la sesión de un usuario en nuestra aplicación, primero recibimos las credenciales de autenticación de éste. Estas credenciales pueden ser la dirección de correo electrónico y contraseña del usuario o un token 'OAuth' de un proveedor de identidades federadas. Una vez tenemos las credenciales, tenemos que pasarlas al Firebase Authentication SDK. Luego, se verificarán dichas credenciales y se devolverá una respuesta al cliente.

Después del acceso exitoso, ya es posible acceder a la información de perfil básica del usuario y controlar el acceso por parte del usuario a los datos almacenados en la Firebase Realtime Database. También se puede usar el token de autenticación proporcionado para verificar la identidad de los usuarios en nuestros servicios de *backend*. Aunque a medida que hemos ido avanzando con la aplicación, se ha optado por combinar este acceso con otra herramienta que nos permite almacenar los datos básicos de usuario de una forma más eficiente. Se trata de 'Ionic Storage', en la sección 3.4.6 Ionic Storage hablaremos sobre ella.

De forma predeterminada, los usuarios autenticados pueden leer datos y escribirlos en la Firebase Realtime Database y el Firebase Storage, del cual hablaremos a continuación. Es posible controlar el acceso de dichos usuarios modificando las reglas para cada uno. En cualquier caso, para las funcionalidades que ofrece nuestra aplicación nos bastan las reglas por defecto.

Un usuario de Firebase tiene un conjunto fijo de propiedades básicas (un ID único, una dirección de correo electrónico) almacenado en una base de datos de usuarios. Nuestro formulario de registro solicita más información sobre los usuarios que se registran (por ejemplo, el país/región). No podemos agregar otras propiedades al objeto de usuario de Firebase directamente. En lugar de ello, tendremos que almacenar las propiedades adicionales en nuestra Firebase Realtime Database.

La primera vez que un usuario se registra en la aplicación, los datos de su perfil se completan con la información disponible:

- Si el usuario se registra con una dirección de correo electrónico y una contraseña a través del formulario, sólo se completa la dirección de correo electrónico. Sin embargo, como hemos indicado anteriormente, es posible añadir nuevos campos en nuestra Firebase Realtime Database como son el nombre de usuario, la región o país y la imagen de perfil que se subirá al Firebase Storage.

- Si el usuario se registra con un proveedor de identidades federadas como Facebook o Google, se usa la información de la cuenta disponible a través del proveedor para completar el perfil de usuario de Firebase.

Cuando un usuario se registra o accede, pasa a ser el usuario actual de la instancia de autenticación. La instancia de autenticación de Firebase conserva el estado del usuario. Por lo tanto, al actualizarse la página (en un navegador) o reiniciar la aplicación no se pierde la información del usuario (*remember me*).

Cuando el usuario cierra la sesión, la instancia de autenticación deja de conservar una referencia al objeto de usuario y con ello el estado de éste. Deja de haber usuario actual y la lógica de la aplicación se encarga de redirigir al usuario a la página de inicio.

Para analizar el ciclo de vida del usuario con un seguimiento del estado actual de la instancia de autenticación de Firebase usaremos receptores denominados 'observables' en JavaScript.

3.3.3 Storage

Firebase Storage ofrece la posibilidad de subir y descargar archivos de forma segura, independientemente de la calidad de la red. Puedes usarlo para almacenar imágenes, audio, vídeo y otro contenido generado por el usuario. Firebase Storage está respaldado por Google Cloud Storage, un servicio potente, simple y rentable de almacenamiento de objetos.

Los desarrolladores usan el SDK Firebase Storage para subir y bajar archivos directamente de los clientes. Si la conexión de red no es buena, el cliente puede volver a intentar la operación justo donde la dejó, lo que permite ahorrar tiempo y ancho de banda a los usuarios. En nuestro caso, subimos imágenes al Storage cuando nos vamos a registrar si añadimos una imagen propia o cuando subimos un archivo de audio o imagen de portada para las canciones.

Firebase Storage almacena tus archivos en una cubeta de Google Cloud Storage, lo que significa que puedes acceder a ellos a través de Firebase y de las APIs de Google Cloud. Esto te brinda la flexibilidad necesaria para subir y descargar archivos de clientes móviles a través de Firebase y realizar procesamientos en el servidor, como filtrado de imágenes. Firebase Storage escala automáticamente, lo que significa que no necesitas migrar de Firebase Storage a Google Cloud Storage ni a ningún otro proveedor.

Además, Firebase Storage se integra perfectamente con Firebase Authentication para identificar usuarios y proporciona lenguaje de seguridad comparativo que te permite acceder a controles en archivos individuales o grupos de archivos, de modo que puedas hacer que los archivos sean tan públicos o privados como desees.

3.4 TECNOLOGÍAS

En este apartado se enumeran el resto de las diferentes herramientas y tecnologías utilizadas durante el desarrollo del proyecto. Hablaremos en profundidad sobre el *framework* de aplicaciones híbridas Ionic 2, ya que las otras dos tecnologías fundamentales, Angular y Firebase, las hemos descrito en los apartados anteriores de 3.2 Diseño arquitectónico y 3.3 Modelo de la base de datos, respectivamente.

3.4.1 HTML5, JavaScript, CSS3 y Sass

Una aplicación híbrida está desarrollada en base a las tecnologías web: HTML (como lenguaje para crear y representar una vista) + JavaScript (como lenguaje interpretado para operaciones) + CSS (como lenguaje de estilo). Son como cualquier otra aplicación de las que puedes instalar a través de las tiendas de aplicaciones para cada sistema, por lo que en principio los usuarios finales no percibirán la diferencia con respecto a otros tipos de aproximaciones diferentes como las aplicaciones nativas (ver Anexo B para más información sobre las aplicaciones híbridas).

Ilustración 17 - Logos Lenguajes HTML5, JavaScript y CSS3 [10]

[10] Logos Lenguajes HTML5, JavaScript y CSS3 - Julio 2107

[11]Además, Ionic incorpora Sass, un lenguaje de hoja de estilos que es traducido a CSS, con el que se pueden hacer diseños avanzados para todo tipo de estilos y gustos. En el fichero 'variables.scss' bajo el directorio 'theme', podemos personalizar la apariencia de la aplicación a través de variables que se pueden sobrescribir.

Ilustración 18 - Logo Sass

3.4.2 TypeScript

[12]Otra cosa que viene dada por el desarrollo de Angular es el uso del lenguaje TypeScript. TypeScript es JavaScript, pero con añadidos pensados para mejorar el trabajo por parte de los desarrolladores, incrementando la productividad.

Ilustración 19 - Logo TypeScript

La mayor aportación de TypeScript al lenguaje JavaScript es la posibilidad de definición de tipos para las variables, pero en general aporta mucho más y además nos permite usar todas las mejoras de ES6 y algunas de ES7 en las aplicaciones. ES o ECMAScript es el estándar que define cómo debe ser el lenguaje JavaScript.

No debemos pensar en TypeScript como un problema por ser algo que debemos aprender de nuevo. TypeScript es en realidad JavaScript, por lo que no significa aprender un nuevo lenguaje, sino agregar algunas cosas al que ya conoces, cosas que en definitiva te van a ayudar durante la etapa de desarrollo y a la hora de mantener las aplicaciones en el futuro.

TypeScript requiere una transpilación del código, de modo que aquel código que escribes en TypeScript se convierta en código JavaScript capaz de ejecutarse en cualquier navegador. Esa transpilación la realiza Ionic 2 por debajo, por lo que no representa un problema para el desarrollador.

[11] Logo Sass - Julio 2017
[12] Logo TypeScript - Julio 2017

3.4.3 Ionic 2

Como hemos indicado, estamos usando Ionic 2 como herramienta para desarrollar la aplicación. Pero ¿por qué Ionic? A continuación, justificaremos su elección.

Ilustración 20 - Logo Ionic 2 [13]

Durante el aprendizaje de esta tecnología, hemos tenido que aprender sobre todo Angular que básicamente hemos explicado en la sección anterior sobre el diseño arquitectónico. La parte de Ionic ha sido sencilla gracias a su documentación oficial que incluye infinidad de funcionalidades con varios ejemplos que ilustran bastante bien lo que hace cada componente, api, *plugins* nativos, etc.

Ionic 2 hace uso de un SDK que nos ayuda a generar las vistas para cada sistema operativo sin necesidad de generar un código para cada uno de ellos, lo cual incrementa la eficiencia de un equipo de trabajo al intentar desarrollar aplicaciones móviles para distintas plataformas.

Los componentes de Ionic 2 ya vienen adaptados al dispositivo de manera estética. Quiere decir que, cuando se compila una aplicación para iOS, el componente se visualizará de manera diferente que cuando se compila para Android. En Android usará 'Material Design' mientras que en iOS usará las guías de diseño definidas por Apple. Sin embargo, es posible mantener, por ejemplo, el estilo de iconos que queramos para cada plataforma.

Esto es una ventaja en sí, porque las personas disfrutarán de aplicaciones con una experiencia de usuario cercana a la que están acostumbrados en su teléfono y nos evita a los desarrolladores la necesidad de trabajar más para conseguir este efecto. Sin embargo, como autores de las aplicaciones con Ionic 2 también somos capaces de alterar el diseño de las aplicaciones, proporcionando una experiencia de usuario específica y original para nuestra propia aplicación.

[13] Logo Ionic 2 - Julio 2017

El alto rendimiento de Ionic 2 es una de las diferencias que lo denotan sobre sus competidores, ya que está construido con la mínima manipulación del DOM, con nada de código jQuery y con aceleraciones de transiciones por hardware. Se trata de una estupenda herramienta para la creación de aplicaciones híbridas sorprendentes, pensada para obtener resultados de una manera rápida y con una menor inversión económica.

3.4.4 Ionic Native y Apache Cordova/PhoneGap

[14]Ionic Native es un contenedor TypeScript para los complementos de Apache Cordova/PhoneGap que hacen que la adición de cualquier funcionalidad nativa que necesites para tu aplicación móvil sea fácil.

Ilustración 21 - Logo Apache Cordova

Este contenedor envuelve las respuestas en una 'Promise' o en un 'Observable', proporcionando una interfaz común para todos los complementos y asegurando que los eventos nativos activan la detección de cambios en Angular.

Por lo tanto, para acceder a los componentes nativos del dispositivo, se usan *plugin*s que nos proporciona, principalmente, Apache Cordova y Phonegap en algunos casos. Actualmente, también Ionic es proveedor de algunos *plugins* para trabajar con la parte nativa del teléfono.

Aquí como nativo nos referimos a elementos como la cámara, geolocalización, etc. Todos esos elementos se pueden usar desde las aplicaciones de Ionic a través de los correspondientes *plugins* nativos, que forman una especie de puente entre el desarrollo con JavaScript y el teléfono.

En este apartado, hablaremos de los diferentes *plugins* usados en nuestra aplicación.

Facebook

[15]Para registrarnos/iniciar sesión en la aplicación a través de nuestra cuenta de Facebook desde el dispositivo móvil, será necesario usar el plugin de Cordova 'cordova-plugin-facebook4' para obtener acceso a la aplicación nativa de Facebook en iOS y Android.

Ilustración 22 - Logo Facebook

Antes de todo, primero hay que incluir la aplicación en el portal de desarrolladores de Facebook, en donde seguiremos unos pasos guiados para configurar la aplicación. También será necesario habilitar a Facebook en Firebase como proveedor de autenticación indicando el identificador de la aplicación junto con la clave secreta que nos ha generado Facebook en el paso anterior.

[14] Logo Apache Cordova - Julio 2017
[15] Logo Facebook - Julio 2017

Por último, para completar la configuración, tenemos que añadir una URI de redireccionamiento de 'OAuth' a la configuración de la aplicación en Facebook. Esta URI nos la suministrará Firebase en el apartado de autenticación al habilitar a Facebook como proveedor de inicio de sesión.

Camera

Este complemento define un objeto global que proporciona una API para capturar imágenes con la cámara o elegirlas de la galería de imágenes del dispositivo.

Como respuesta exitosa a la llamada, la imagen se devuelve como una cadena que contiene la imagen de la foto codificada en Base64 o como una cadena que representa la ubicación del archivo de la imagen en el almacenamiento local (ver Anexo C para más información sobre este plugin)

Una vez tenemos la imagen, en nuestro caso en base64, la vamos a publicar en el Storage de Firebase. Para ello, tenemos que transformarla a tipo 'blob' o 'file' que son los formatos que acepta Firebase Storage. Una vez transformada, procedemos a subirla a la nube y, posteriormente, añadir su referencia o URL de descarga a la 'Realtime Database' para acceder a ella a través de las peticiones oportunas.

Android Permissions

Este complemento está diseñado para admitir nuevos mecanismos de control de permisos de Android. Necesario para acceder a contenido alojado en el dispositivo móvil.

File Chooser

Abre el selector de archivos en Android para que el usuario seleccione alguno. Devuelve un archivo URI. Complemento usado para seleccionar algún archivo de audio.

File Path

Este complemento permite resolver la ruta del sistema de archivos nativo para los URI de contenido de Android y se basa en el código de la biblioteca 'File Chooser'. Usado para tratar la ruta del archivo de audio.

Native Audio

Este complemento de Cordova/PhoneGap permite la simultaneidad (reproducción multicanal), la polifonía (reproducción de múltiples voces) y la latencia minimizada (mediante caché) en aplicaciones basadas en audio, aprovechando las API de audio nativas. Está diseñado para juegos multiplataforma basados en HTML5 y aplicaciones de audio híbridas móviles.

Una gran parte para conseguir que una aplicación móvil sea de alta calidad es proporcionar retroalimentación inmediata y obvia para las acciones de usuario. Cuando un usuario realiza alguna acción, querrá comunicarse con ella de manera que perciba que dicha acción ha tenido éxito. Esa información puede comunicarse visualmente, a través del sonido o incluso a través de una sensación física como la vibración.

Si has utilizado la aplicación de Facebook, es probable que hayas notado que al usar la aplicación se reproducen varios efectos de sonido. Por ejemplo, a medida que cambiamos entre las pestañas, al pulsar algún botón, etc.

En nuestra aplicación, hemos agregado efecto de sonido al pulsar un botón de tal manera que se usará audio HTML5 estándar por defecto o audio nativo cuando la plataforma sea Cordova.

Al utilizar audio HTML5, el audio se carga al vuelo por lo que hay un ligero, pero notable retraso antes de que el audio se reproduzca. De lo contrario, cuando se utiliza audio nativo, el sonido está precargado y se puede reproducir al instante.

Action Sheet

El complemento 'ActionSheet' muestra una lista nativa de opciones de las que el usuario puede elegir. Usado para desplegar opciones referentes a una canción.

Social Sharing

Este complemento permite usar la ventana nativa del dispositivo móvil para compartir contenido. Es posible compartir texto, un enlace, imágenes u otros archivos como pdf. El asunto también es compatible, cuando la aplicación de destino lo admite.

Permite compartir archivos desde Internet y el sistema de archivos local. Además, se puede compartir directamente con Twitter, Facebook u otras aplicaciones.

Phonegap Plugin Push

Utilizamos el complemento Push de Phonegap para recibir notificaciones nativas. Al instalar el plugin, es necesario proporcionar una variable SENDER_ID, que es el ID del remitente FCM (*Firebase Cloud Messaging*). Esta variable la podemos obtener desde el apartado '*Cloud Messaging*' de configuración del proyecto en Firebase.

Otro de los requisitos es tener instalado Cloud Client para interactuar con los servicios de Ionic desde nuestra aplicación. Por consiguiente, tenemos que facilitar nuestra clave del servidor FCM a Ionic. Vamos al apartado de certificados en la configuración del proyecto de Ionic y añadimos un nuevo perfil de seguridad e insertamos la clave FCM.

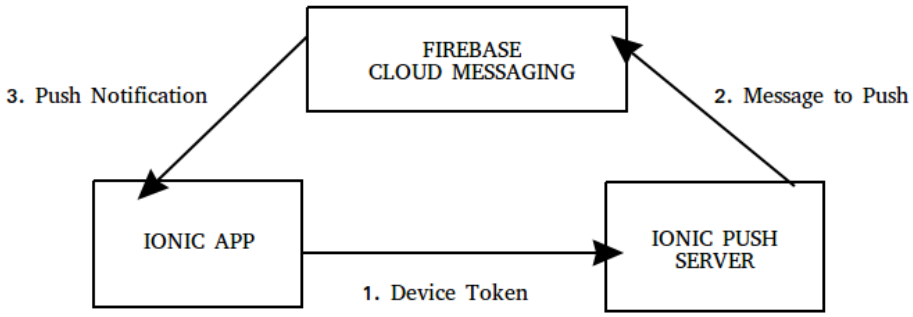
3.4.5 Firebase Notifications

Firebase Notifications es un servicio gratuito para desarrolladores de aplicaciones para dispositivos móviles que permite enviar notificaciones orientadas a los usuarios.

Basada en Firebase Cloud Messaging y en el SDK FCM, Firebase Notifications ofrece una opción para una plataforma de notificación flexible que requiera codificación mínima para comenzar y una consola gráfica para enviar mensajes.

Firebase Cloud Messaging se ocupa del enrutamiento y la entrega a los dispositivos de destino. Cuando la aplicación está en segundo plano en el dispositivo de un usuario, las notificaciones se entregan en la bandeja del sistema. Cuando un usuario hace clic en la notificación, el lanzador abre la aplicación. Si se desea, también es posible agregar administración de mensajes del cliente para recibir notificaciones cuando la aplicación ya esté en primer plano.

En nuestra aplicación, enviamos mensajes de notificación usando el protocolo HTTP a través de una solitud POST a la API de Firebase Cloud Messaging. De tal manera que, tenemos que especificar en la cabecera el tipo de contenido y la clave de autorización que obtenemos en la configuración de nuestro proyecto de Firebase, junto con el cuerpo del mensaje donde irán los datos y los tokens de los clientes a los que queremos enviar la notificación. En el siguiente esquema, se muestra de forma resumida el ciclo de funcionamiento.



[16]

Esquema 4 – Ciclo notificaciones push

[16] Ciclo notificaciones push - Julio 2017

3.4.6 Ionic Storage

El almacenamiento de Ionic es una manera fácil de almacenar pares clave/valor y objetos JSON. Dicho almacenamiento utiliza una variedad de motores por debajo, escogiendo el mejor disponible en función de la plataforma.

Cuando se ejecuta en un contexto de aplicación nativa, Storage priorizará el uso de SQLite, ya que es una de las bases de datos más estables y ampliamente utilizadas. Además, evita algunas de las trampas de 'localstorage' e 'IndexedDB', como el sistema operativo que decide eliminar los datos almacenados en situaciones de bajo espacio en disco.

Cuando se ejecuta en la web o como una aplicación web progresiva, Storage intentará utilizar 'IndexedDB', 'WebSQL' y 'localstorage', en ese orden.

En nuestro caso, usamos esta técnica para al iniciar sesión como usuario, guardar pares clave/valor que nos indiquen el nombre de usuario, la imagen de perfil y la región a la que pertenece dicho usuario. De esta forma es más eficiente, por ejemplo, a la hora de comentar una canción, obtener el nombre de usuario y la foto de perfil del usuario actual.

3.4.7 Ionic Creator

[17]Esta herramienta nos permite diseñar una aplicación base usando algunos componentes del *framework* que vamos a usar durante el desarrollo de la aplicación. Además, nos permite modificar los temas predeterminados e incluye un editor donde podemos incluir clases SCSS para los diferentes componentes que usemos.

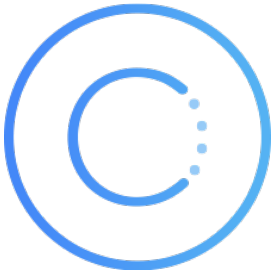


Ilustración 23 - Logo Ionic Creator

Esta herramienta está limitada con la cuenta gratis, es decir, existen numerosos componentes, APIs, *plugins* nativos que Ionic Creator no incluye de forma gratuita. Por lo que simplemente la usaremos para hacernos una idea de cómo puede quedar nuestra aplicación objetivo.

Se trata de una herramienta bastante potente, ya que nos ahorra bastante trabajo para la parte *frontend* de las aplicaciones móviles. Nos da la posibilidad de exportar nuestro prototipo para integrarlo en el proyecto/carpeta donde vamos a trabajar realmente con el *framework*.

Como inconveniente nos encontramos con la privacidad del prototipo, es necesario actualizarse a una cuenta de pago para poder privatizar nuestro proyecto o protegerlo con una contraseña. De lo contrario, cualquiera que obtenga la URL puede verlo (no editarlo ni borrarlo).

[17] Logo Ionic Creator - Julio 2017

3.4.8 AngularFire2

Para facilitar la interacción con Firebase, hemos usado la librería oficial denominada AngularFire2 que nos ayuda bastante para el tráfico de información entre nuestra aplicación y el servicio en la nube de Firebase. Es útil tanto para peticiones de información como para labores de autenticación.

Ilustración 24 - Logo AngularFire2 [18]

3.4.9 Lodash

[19]Una biblioteca de utilidades JavaScript que ofrece modularidad, rendimiento y extras. Lodash hace que JavaScript resulte más fácil al no tener que trabajar con matrices, números, objetos, cadenas, etc. Los métodos modulares de Lodash son excelentes para:

- Arrays, objetos y cadenas de iteración.
- Manipulación y prueba de valores.
- Creación de funciones compuestas.

Ilustración 25 - Logo Lodash

Es un caso familiar al tratar con muchos datos, el tema del procesamiento de la información antes o después de mostrarla al usuario. La biblioteca Lodash nos ahorra ese tiempo que se pierde cuando comenzamos la cascada de ideas para lograr limpiar u obtener la información que queremos.

Todo ese trabajo normalmente no tiene nada que ver con la idea principal, es sólo procesamiento de datos: mover *arrays*, tomar algunos campos, filtrar por condiciones. Al fin y al cabo, son detalles que, aunque necesarios, no debemos gastar nuestro preciado tiempo en ellos. La aplicación que estamos desarrollando usa Lodash para varias funcionalidades como por ejemplo buscar o filtrar por un valor, ordenar una lista en función de un campo como puede ser la fecha de publicación, etc. Todas estas tares son repetitivas en toda la aplicación, por lo que usando Lodash evitamos la repetición de código y un gasto de tiempo considerable.

[18] Logo AngularFire2 - Julio 2017
[19] Logo Lodash - Julio 2017

[20]3.4.10 Angular2-Charts

Esta librería/módulo nos ayuda a mostrar información relevante para el usuario de una forma homogénea y simple como son los gráficos. En nuestra aplicación hemos usado el gráfico tipo "Pie" (gráfico circular) que nos ilustra con las tendencias de estilo de música predominantes.

Ilustración 26 - Logo Chart.js

3.4.11 GitHub

[21]GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Ilustración 27 - Logo GitHub

GitHub aloja tu repositorio de código y te brinda herramientas muy útiles para el trabajo en equipo, dentro de un proyecto.

Básicamente, el uso que le hemos dado a esta herramienta es para salvaguardar nuestro código cada vez que hallamos completado alguna historia de usuario o corregido tareas pendientes, etc. Junto a esta plataforma, hemos usado otra herramienta bastante útil para mantenernos organizados en todo momento durante el desarrollo de la aplicación. A continuación, hablaremos un poco sobre ella.

3.4.12 Waffle.io

[22]Waffle.io es una herramienta de gestión de proyectos para equipos de desarrollo que utiliza los datos de GitHub como fuente principal de registro.

Ilustración 28 - Logo Waffle.io

Dentro de sus funcionalidades destacamos la de añadir tarjetas que identifican historias de usuario, problemas, solicitudes, ... Nos ayuda a planificar, organizar y rastrear el trabajo como un equipo en una visión global.

Estos problemas (issues) se encuentran en el repositorio de GitHub que hemos vinculado al crear un proyecto en Waffle.io y los podemos colocar en diferentes columnas que identifican el estado de cada tarea. Además, podemos especificar en la tarea quién o quiénes se encargarán de ella, el peso de cada una, etiquetas, etc. Por otro lado, tenemos las gráficas que nos indican lo que hemos avanzado cada día al completar 'issues', diagrama

[20] Logo Chart.js - Julio 2017
[21] Logo GitHub - Julio 2017
[22] Logo Waffle.io - Julio 2017

burndown que nos representa una gráfica del trabajo por hacer en un proyecto en el tiempo, en otras palabras, es un diagrama que representa una serie temporal del trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo. Usualmente se usa en el desarrollo ágil de software, especialmente con Scrum.

3.4.13 Zedge

[23]Zedge es una plataforma de contenido, líder mundial en personalización de smartphones, con más de 250 millones de descargas y 33 millones de usuarios activos mensuales. La gente usa Zedge para hacer sus smartphones más personales, para expresar sus emociones, gustos e intereses usando fondos de pantalla, iconos, widgets, tonos y más. La plataforma Zedge permite a las marcas, artistas y creadores compartir el contenido de personalización de sus smartphones con sus fans para ampliar su alcance, reforzar su mensaje y obtener información valiosa sobre cómo interactúan los clientes con su contenido.

Ilustración 29 - Logo Zedge

Simplemente la hemos usado como proveedor de imágenes y audios de prueba para nuestra aplicación.

3.5 PRUEBAS

Además de las pruebas finales, la aplicación se ha ido testeando parcialmente a medida que se iban completando los requisitos. En todas las fases de pruebas se va comprobando la correcta adición, modificación o eliminación de los datos a Firebase. Podemos dividir las pruebas según los medios que se han utilizado para validarlas, ya que algunas de ellas requieren elementos nativos del dispositivo móvil.

La mayoría de las pruebas las hemos confirmado a través del navegador web Chromium gracias a su herramienta para desarrolladores de aplicaciones móviles. Con esta herramienta podemos ver cómo se verá nuestra aplicación en diferentes dispositivos móviles con los sistemas operativos de Android e iOS. Además, disponemos de una consola en la que podemos observar los errores en la aplicación si los hubiera, el flujo de la aplicación una vez puesta en marcha, la cache del navegador, el código HTML y CSS de la página que se está visualizando (sobre todo el código CSS que permite habilitar y deshabilitar los atributos mostrando los cambios inmediatamente).

[23] Logo Zedge - Julio 2017

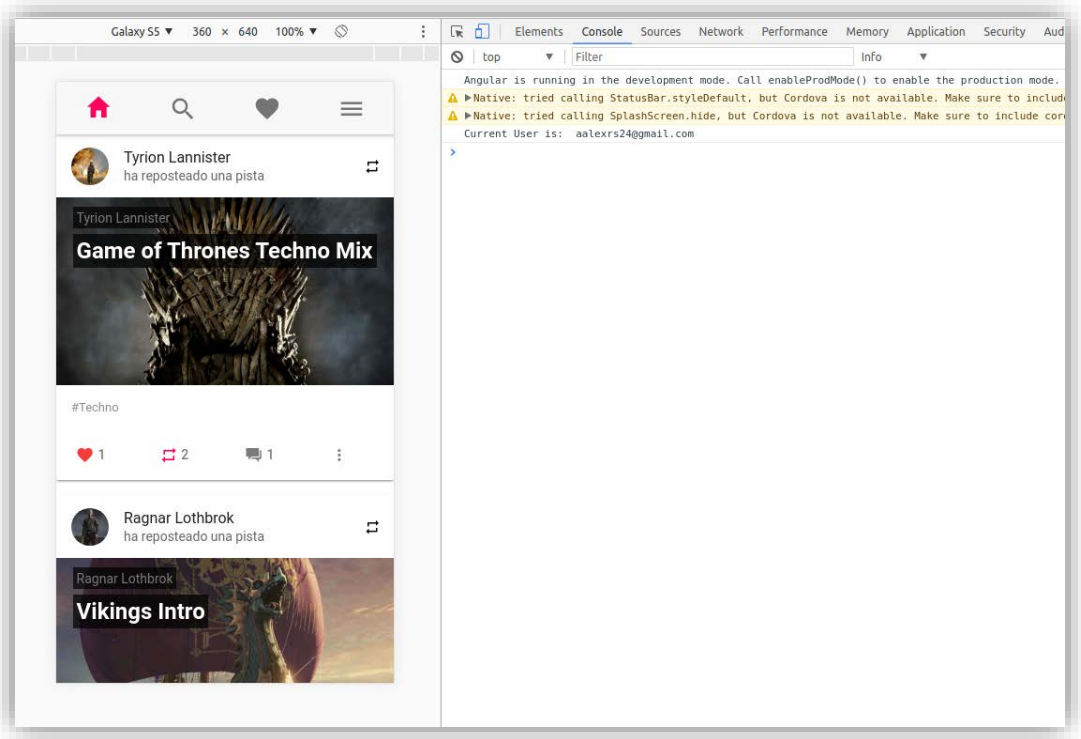


Ilustración 30 - Android Galaxy S5 consola de pruebas

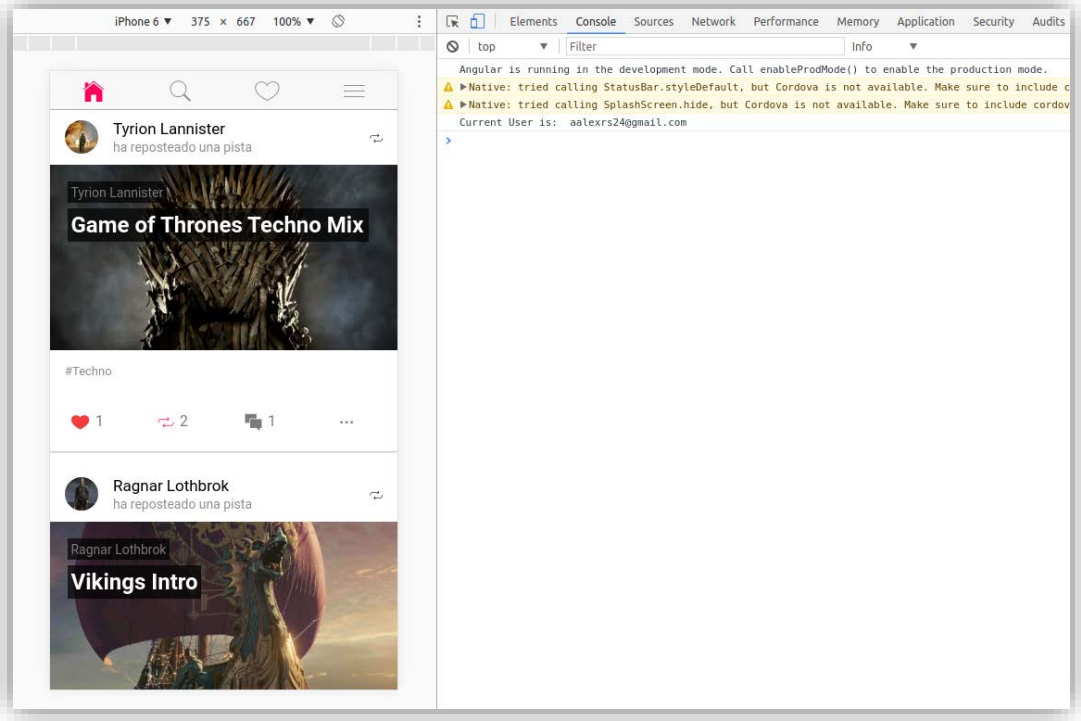


Ilustración 31 - Iphone 6 consola de pruebas

Por otro lado, para probar las funcionalidades que usan complementos de Cordova hemos utilizado una extensión de Google Chrome denominada 'Cordova Mocks'. Sin embargo, los plugins soportados no son suficientes para probar todas las características de la aplicación. Por ello, ha sido necesario instalar las SDK Tools de Android que nos permitirán probar la aplicación con todos los complementos operativos en un dispositivo móvil Android. Además, probaremos el rendimiento de la aplicación usando el hardware para la que se ha creado.

A la hora de probar nuestra aplicación en un dispositivo móvil real, no disponemos de ninguna consola en el smartphone que nos indique el flujo de datos al arrancar la aplicación. Para ello, disponemos de una extensión de Google Chrome llamada 'Inspect Devices' que nos permite, habilitando la depuración USB en las opciones de desarrollo de nuestro smartphone y conectándolo a nuestro equipo de trabajo, percibir los cambios como si estuviéramos emulando la aplicación en el navegador. Sin esta extensión, se complica bastante el trabajo cuando probamos funcionalidades que usan complementos nativos del dispositivo móvil.

Con respecto al contenido de prueba que vamos a utilizar en la aplicación, es decir, imágenes y audio. Vamos a usar una aplicación que podemos descargar desde la 'Play Store' denominada 'Zedge'. Podemos descargar todo tipo de imágenes, tonos, … para incluir como ejemplos de contenido y poder probar la aplicación. Todo el material descargado pertenece a Zedge y será usado solamente con fines académicos.

Para terminar, validaremos globalmente la aplicación con las pruebas finales oportunas para cada funcionalidad implementada, observando el rendimiento de la aplicación.

3.6 ACCESO AL CÓDIGO Y DESPLIEGUE

Para ver el código de la aplicación se adjunta un enlace al repositorio de GitHub y otro para ver cómo se han ido completando algunos 'issues' en Waffle.io (algunos ya se han cerrado y no se ven en la columna 'Done'). Sin embargo, en el apartado Issues de GitHub se guardan todos los movimientos.

Actualmente, el proyecto como prototipo, es funcional y operativo para una primera fase de demostración.

4 Conclusiones

Al finalizar el desarrollo de la aplicación y comprobar los resultados obtenidos, hemos llegado a la conclusión de que todo el trabajo dedicado a este proyecto da sus frutos. Nuestro prototipo cumple con los requisitos del análisis realizado y se alcanzan los objetivos con creces.

Durante el proceso nos hemos topado con numerosos problemas que han ralentizado nuestro trabajo, pero con calma y paciencia se han ido solucionando. Al usar tecnologías y lenguajes relativamente recientes, la documentación que se encentra en internet es escasa. Por lo que el tiempo de búsqueda de información ha afectado enormemente al desarrollo de la aplicación.

El proyecto puede seguir mejorando e incorporar nuevas funcionalidades como, por ejemplo, la implementación de un reproductor en segundo plano. En la columna 'Backlog' de Waffle.io, hemos apuntado algunas ideas que podrían llegar a implementarse usando las herramientas oportunas que nos ofrecen las tecnologías de las que nos hemos servido.

La aplicación se ha instalado y testeado en varios dispositivos móviles Android sin ningún tipo de problema. El siguiente paso es probarla en un dispositivo iOS y verificar sus funcionalidades. En este punto, nos damos cuenta de que el uso de tecnologías para desarrollar aplicaciones híbridas para móviles nos ahorra una cantidad de tiempo considerable. No obstante, no es oro todo lo que reluce, el tiempo de inicio de la aplicación puede no agradar a algunos usuarios impacientes. Esto es algo que, desde mi humilde opinión, debería intentar mejorarse para el futuro prometedor de las aplicaciones híbridas.

Por otro lado, la integración con Firebase también ha supuesto un gasto de tiempo notable. Sobre todo, a la hora de tratar con archivos (imágenes y audios). Al usar componentes nativos del dispositivo móvil para seleccionar los archivos, es necesario formatear su salida según las especificaciones de Firebase. Y considero que la documentación de Ionic es algo escasa para esta parte que se dedica al manejo de los *plugins* nativos. No obstante, el resto de documentación de Ionic referente a los componentes y APIs me parece bastante completa con ejemplos ilustrativos.

En definitiva, creo que este trabajo me ha aportado grandes conocimientos que, probablemente, reciclaré en el futuro para otros proyectos.

Fuentes de información

1. SDLC - Modelo en cascada
   http://www.w3ii.com/es/sdlc/sdlc_waterfall_model.html
   [Último acceso Julio 2017]

2. Para reducir la piratería, dale a la gente lo que quiere
   http://latam.pcmag.com/musica-productos/2241/opinion/para-reducir-la-pirateria-dale-a-la-gente-lo-que-quiere
   [Último acceso Julio 2017]

3. Las dos caras ocultas del modelo de negocio de Spotify
   http://elespectadordigital.com/modelo-de-negocio-de-spotify/
   [Último acceso Julio 2017]

4. Cuál es el Modelo de Negocio de Spotify
   http://www.submarinoatomico.com/2017/05/25/cual-es-el-modelo-de-negocio-de-spotify/
   [Último acceso Julio 2017]

5. Decálogo de buenas prácticas: Aspectos legales de las aplicaciones móviles
   https://www.yeeply.com/blog/decalogo-de-buenas-practicas-aspectos-legales-de-las-aplicaciones-moviles/
   [Último acceso Julio 2017]

6. Aplicaciones móviles y Ley de Protección de Datos
   http://www.mundolopd.com/lopd/proteccion-de-datos-regula-las-aplicaciones-moviles/
   [Último acceso Julio 2017]

7. Angular 2 - Architecture
   http://www.w3ii.com/angular2/angular2_architecture.html
   [Último acceso Julio 2017]

8. Firebase para Android: Base de Datos en Tiempo Real
   http://www.sgoliver.net/blog/firebase-android-base-datos-tiempo-real-1/
   [Último acceso Julio 2017]

9. Documentación Firebase
   https://firebase.google.com/docs/
   [Último acceso Julio 2017]

10. Ionic 2 & Xamarin, dos herramientas para crear aplicaciones híbridas
https://medium.com/@juans_gro/ionic-2-xamarin-el-futuro-del-desarrollo-de-aplicaciones-ed653284e4fe
[Último acceso Julio 2017]

11. Documentación Ionic
https://ionicframework.com/docs/
[Último acceso Julio 2017]

12. Ionic Native
https://ionicframework.com/docs/native/
[Último acceso Julio 2017]

13. Joshmorony Tutorials
https://www.joshmorony.com
[Último acceso Julio 2017]

14. GitHub
https://github.com
[Último acceso Julio 2017]

15. Waffle.io
https://github.com/waffleio/waffle.io
[Último acceso Julio 2017]

16. Lodash
https://lodash.com/
[Último acceso Julio 2017]

17. Chrome Cordova Mocks
https://github.com/pbernasconi/chrome-cordova
[Último acceso Julio 2017]

18. Stack Overflow
https://stackoverflow.com/
[Último acceso Julio 2017]

19. Push Notifications
https://docs.ionic.io/services/push/
[Último acceso Julio 2017]

20. Angular Tutorials
https://www.pluralsight.com/
[Último acceso Julio 2017]

Anexos

A) MANUAL DE USUARIO

Al arrancar la aplicación se mostrará una pantalla de carga con un fondo de portada.

Ilustración 32 - Splash Screen

A continuación, podemos elegir entre registrarnos con nuestro email o a través de Facebook (Google+ y Twitter no están disponibles todavía). La otra opción es iniciar sesión con nuestro email/contraseña o, del mismo modo que para registrarse, usar nuestra cuenta de Facebook.

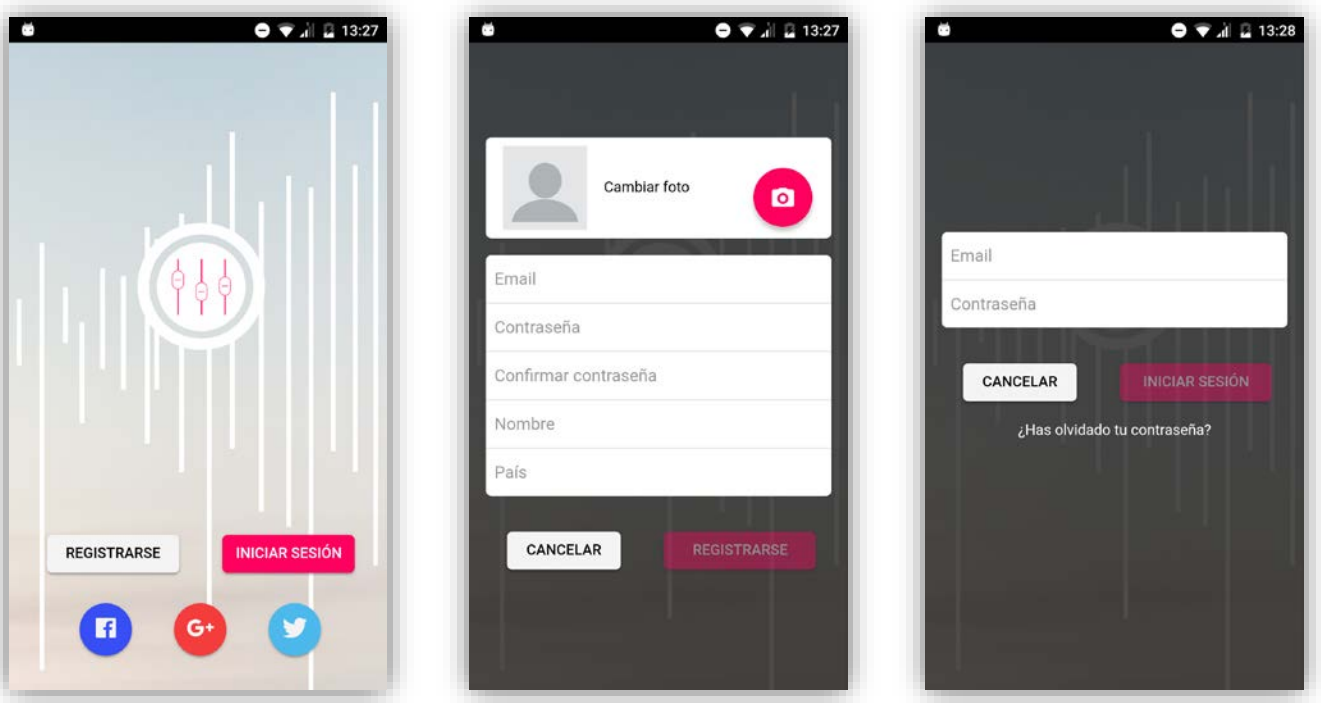


Ilustración 33 - Pantalla de inicio, formulario de registro/inicio de sesión

En caso de que hayamos olvidado la contraseña, no pasa nada, podemos cambiarla mediante la opción "¿Has olvidado tu contraseña?"

Ilustración 34 - Vista para recuperar contraseña

Una vez iniciada la sesión, se nos mostrará un mensaje de bienvenida y accedemos a la pantalla principal. Nada más entrar, podemos observar un mensaje con un icono de una lupa que nos indica y redirige a la vista para buscar canciones o usuarios a los que seguir. Si ya, anteriormente, hemos iniciado sesión en la aplicación y seguido a algún usuario, nos aparecerán las pistas que ha subido y compartido. Aquí, podemos usar la barra de navegación y explorar las diferentes funcionalidades que nos ofrece la aplicación.

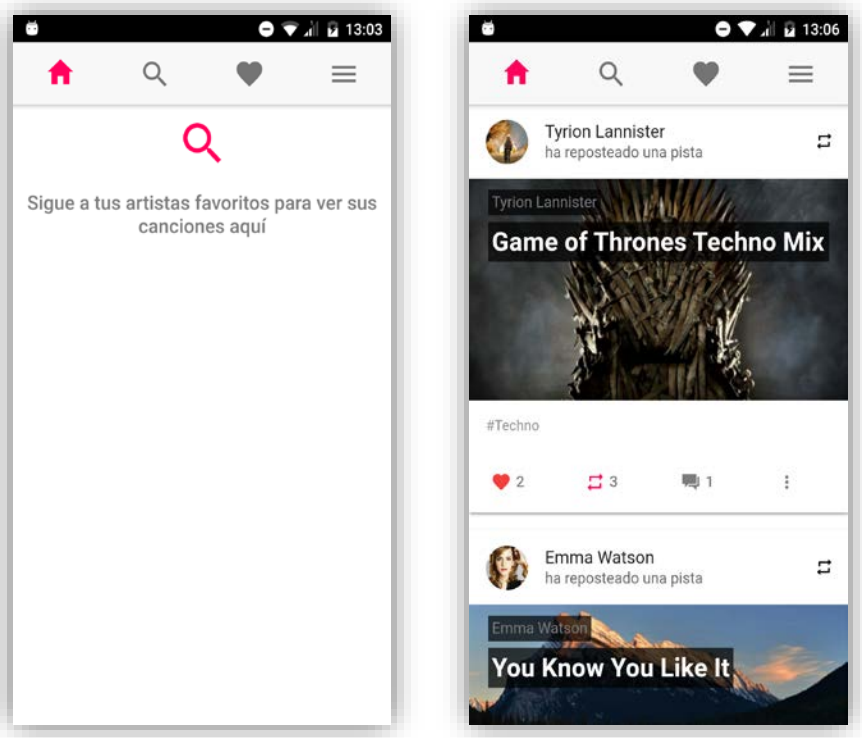


Ilustración 35 - Vista principal ambos casos

El usuario puede ver y filtrar canciones/usuarios por su nombre. En esta vista es posible seguir a los usuarios a través de un botón intuitivo (o dejar de seguir).

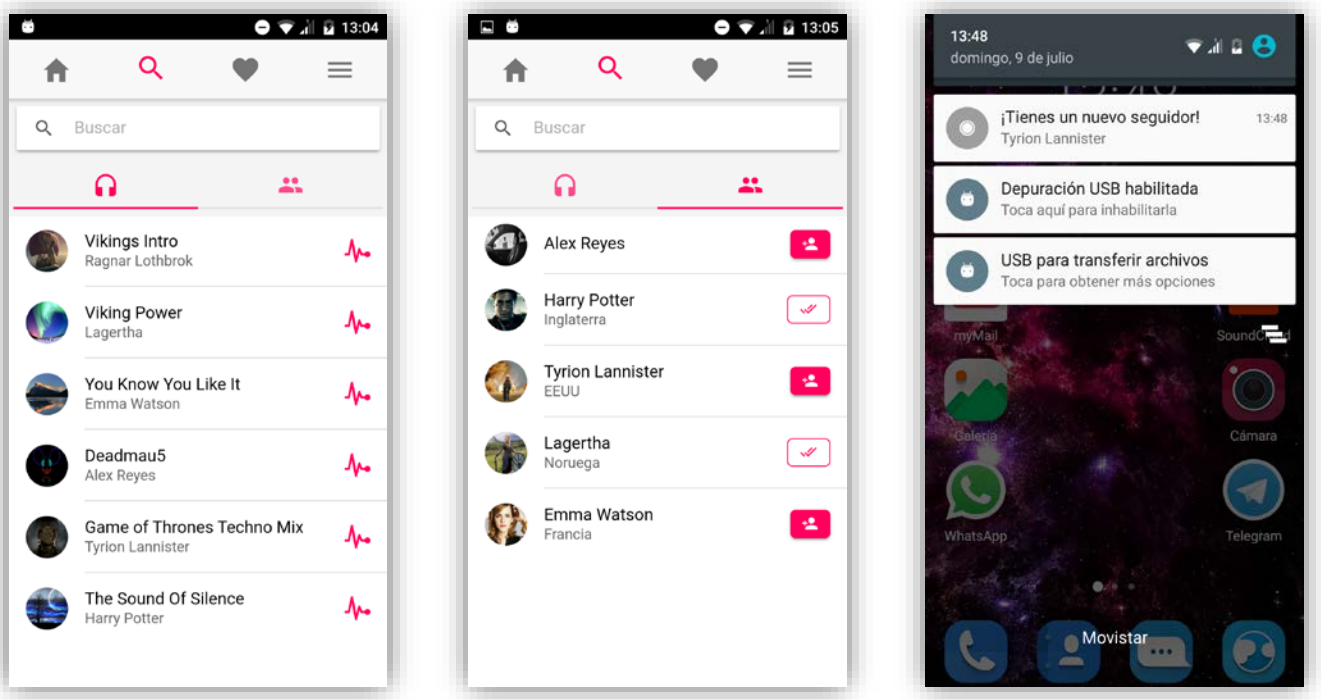


Ilustración 36 - Vista para buscar canciones/usuarios y notificación de nuevo seguidor

A parte, podemos ver en detalle las canciones y el perfil público de los usuarios.

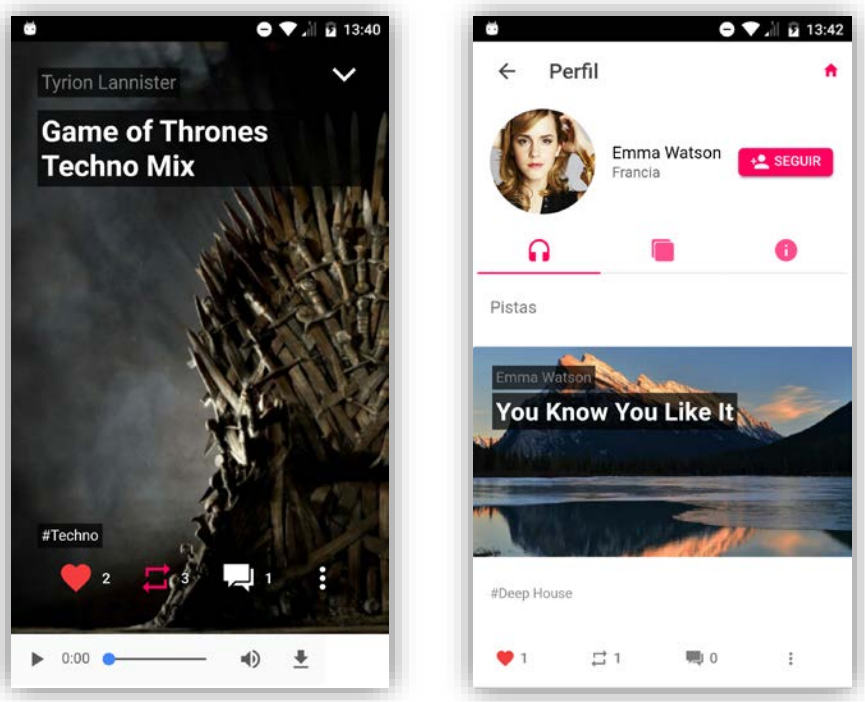


Ilustración 37 - Vista detalle canción y perfil de usuario

Para las canciones, podemos agregarlas a me gusta pulsando el botón del corazón y si queremos eliminar nuestro me gusta, volvemos a pulsar el corazón (observaremos un cambio de color y un sonido característico al agregar a me gusta). Luego, tenemos el botón de 'repost' que sirve para compartir la canción a nuestros seguidores (les aparecerá en su vista principal 'Home'). En la vista de detalle de las canciones, además, podemos reproducir la canción y navegar al perfil del usuario que la ha subido a través de su nombre.

Otra funcionalidad es la de ver los comentarios de una canción y agregar/eliminar comentarios propios. También es posible navegar al perfil de cualquier usuario que ha comentado haciendo click en su foto.

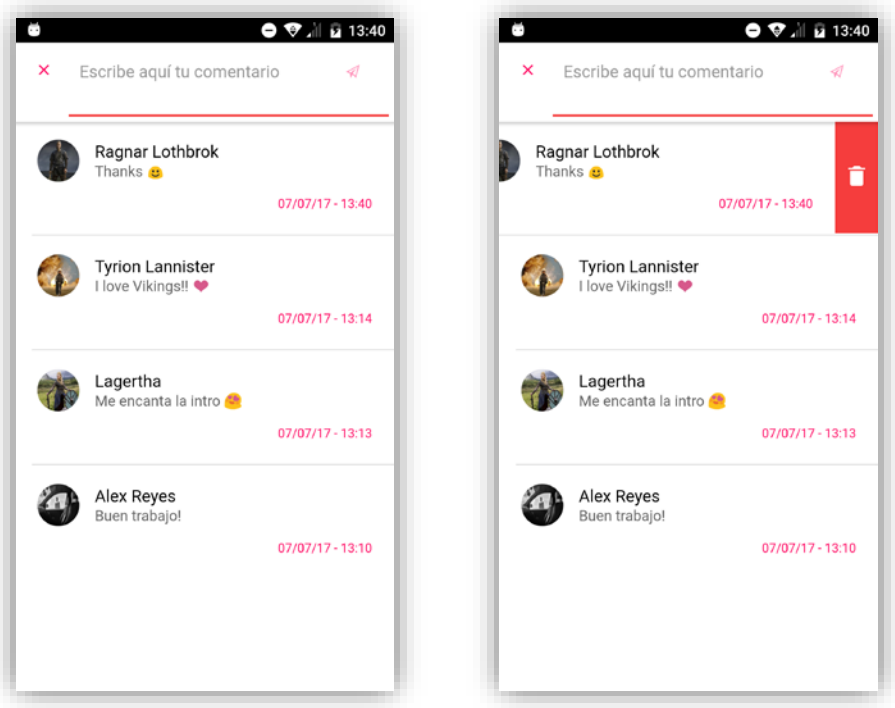


Ilustración 38 - Vista comentarios de una canción

El último botón nos abrirá una pequeña ventana donde podemos agregar la canción a una lista existente o crear una nueva con esa canción. Si seleccionamos 'crear lista', se nos pedirá el título y la privacidad que queremos para la lista. Como última opción, podemos compartir la canción a través de las redes sociales.

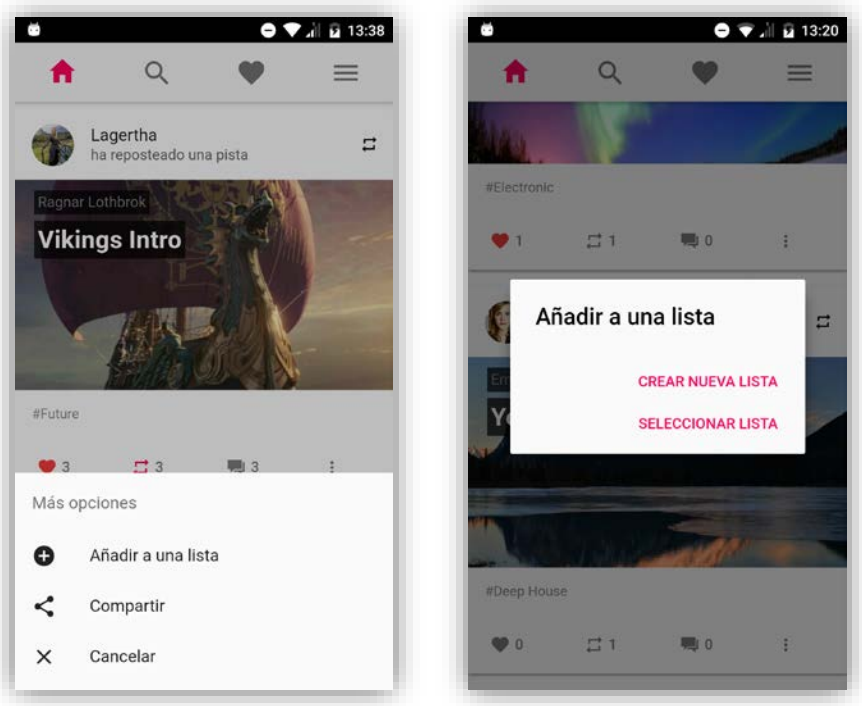


Ilustración 39 - Vista más opciones - Añadir a una lista

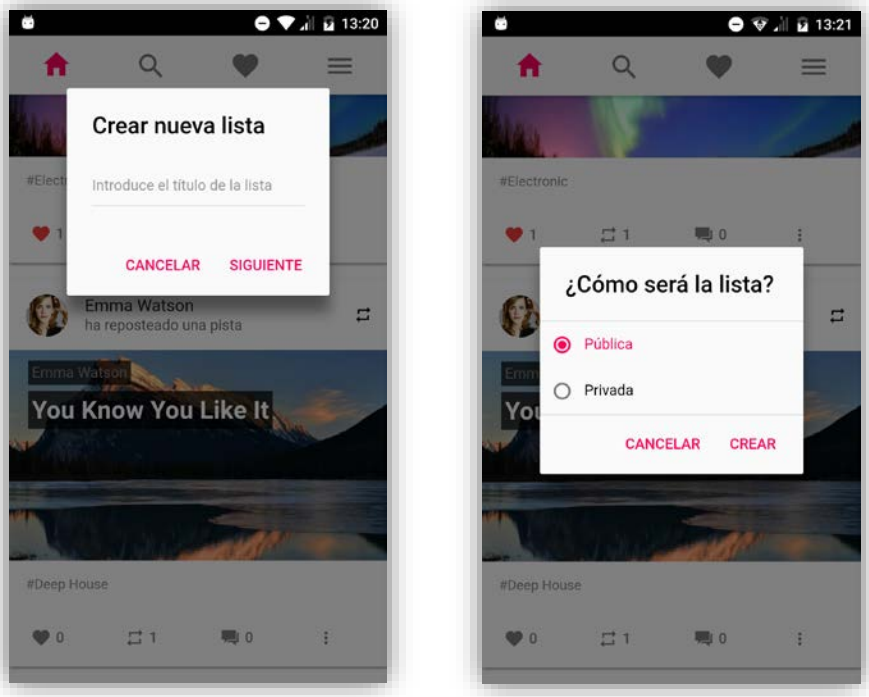


Ilustración 40 - Vista crear nueva lista

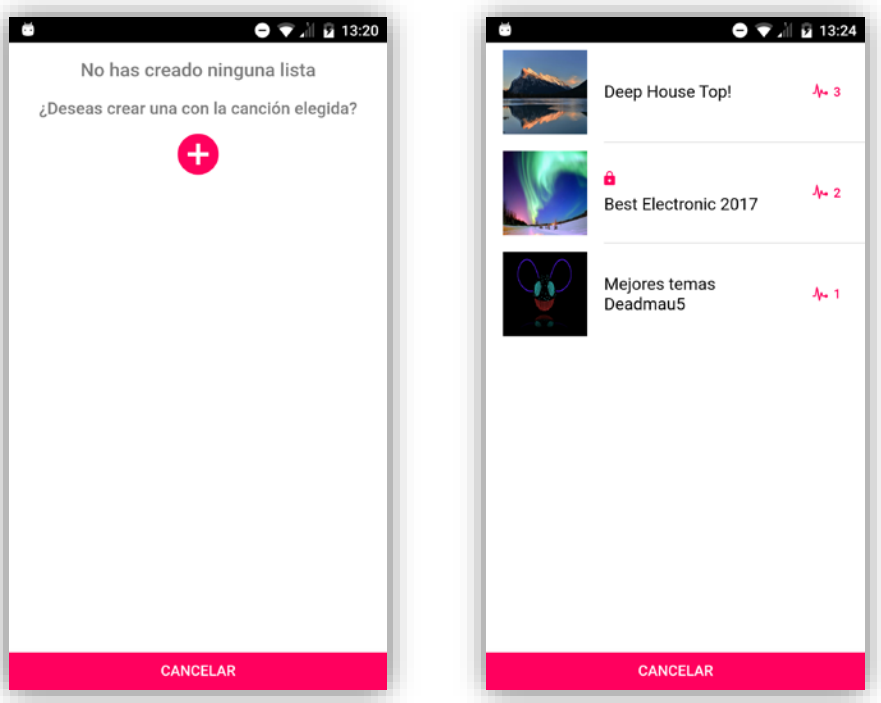


Ilustración 41 - Vista seleccionar una lista (ambos casos)

En la sección del corazón, se encuentran las canciones que hemos agregado a me gusta.

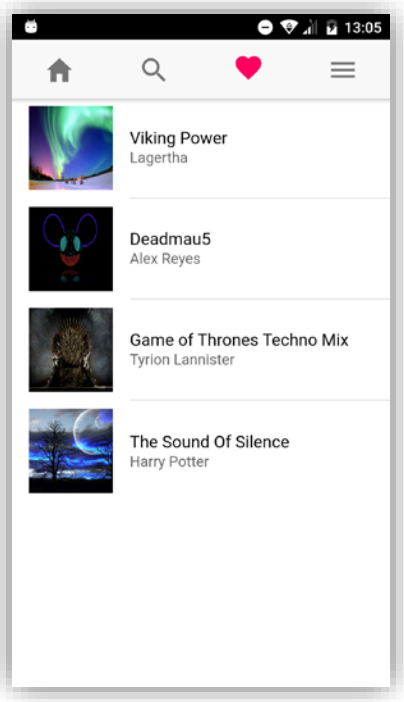


Ilustración 42 - Vista biblioteca de me gusta

En nuestro menú, disponemos de las siguientes funcionalidades.

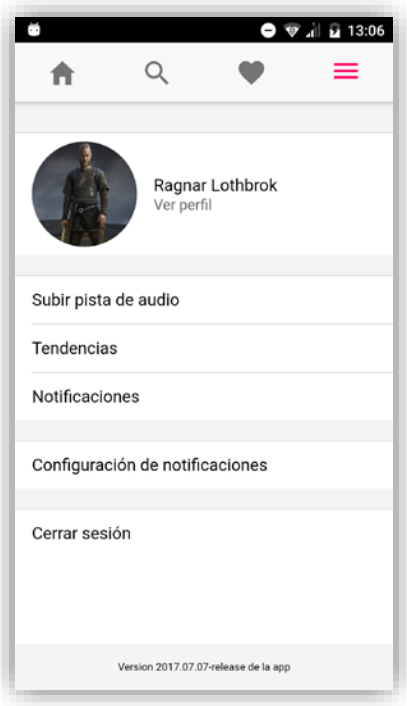


Ilustración 43 - Vista opciones de menú

La primera es para ver nuestro perfil de usuario. Aquí aparecen nuestros datos, las canciones que hemos subido (públicas y privadas), listas creadas (públicas y privadas) y los usuarios seguidos/seguidores.

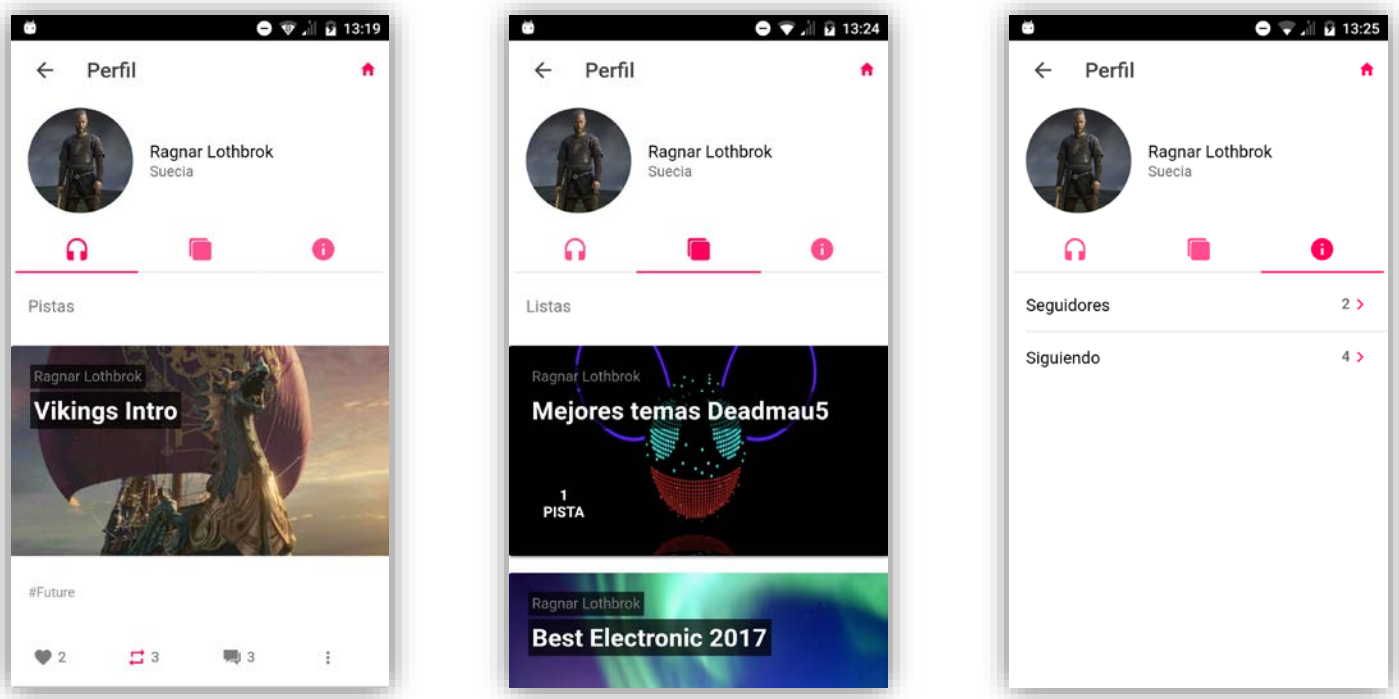


Ilustración 44 - Vista perfil de usuario

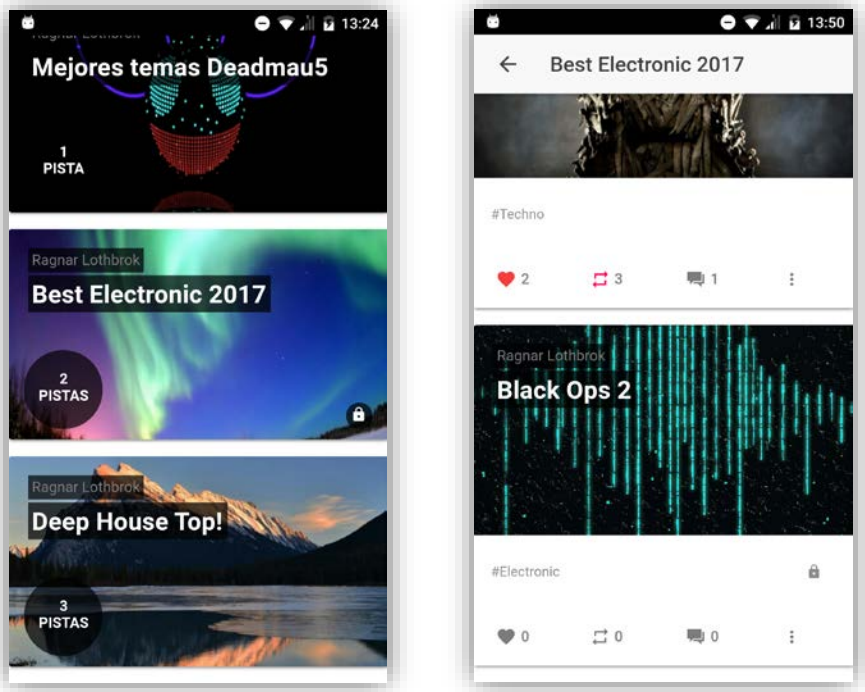


Ilustración 45 - Vista listas creadas y canciones dentro de una lista

Disponemos de un formulario para subir una canción a la nube.

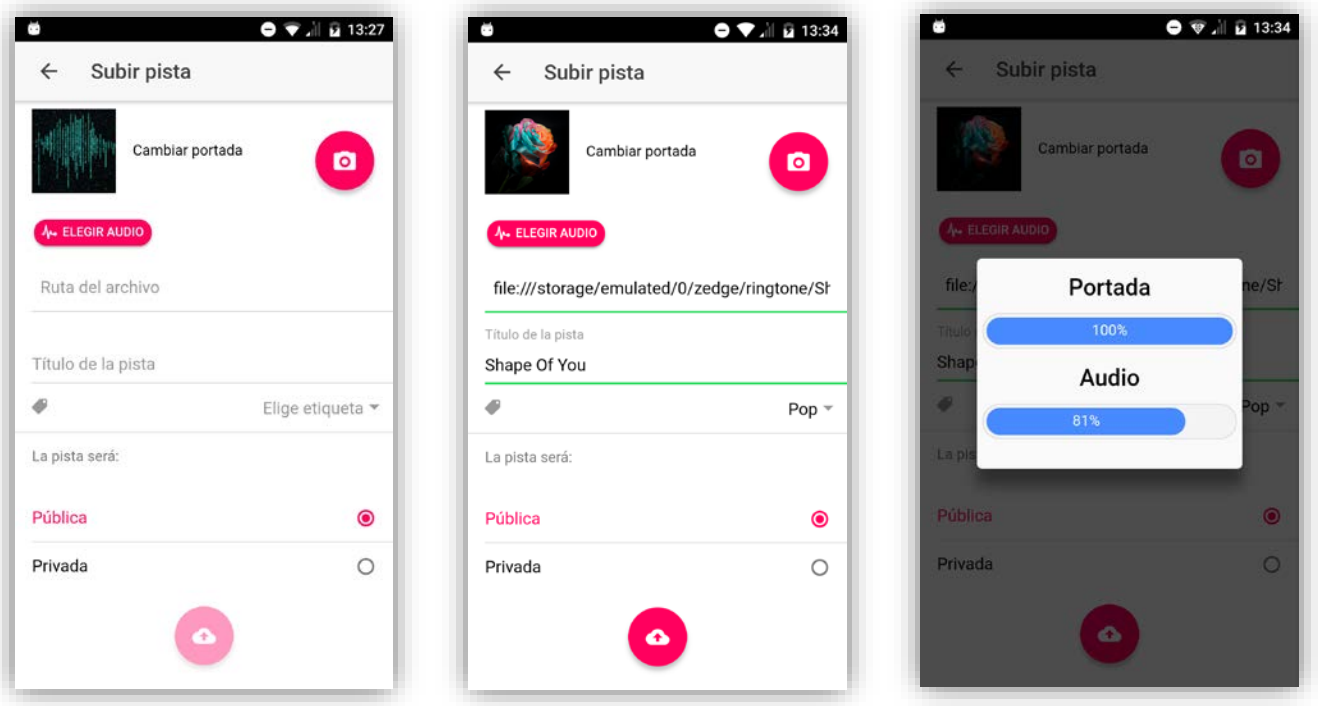


Ilustración 46 - Vista formulario para subir canción

Es posible ver las tendencias actuales según el estilo de música a nivel de los usuarios de la aplicación.

Ilustración 33 - Vista gráfico tendencias musicales

También disponemos de una lista con las notificaciones de los usuarios que nos han seguido. Las notificaciones emergentes se pueden activar/desactivar en la sección de 'Configuración de notificaciones'.

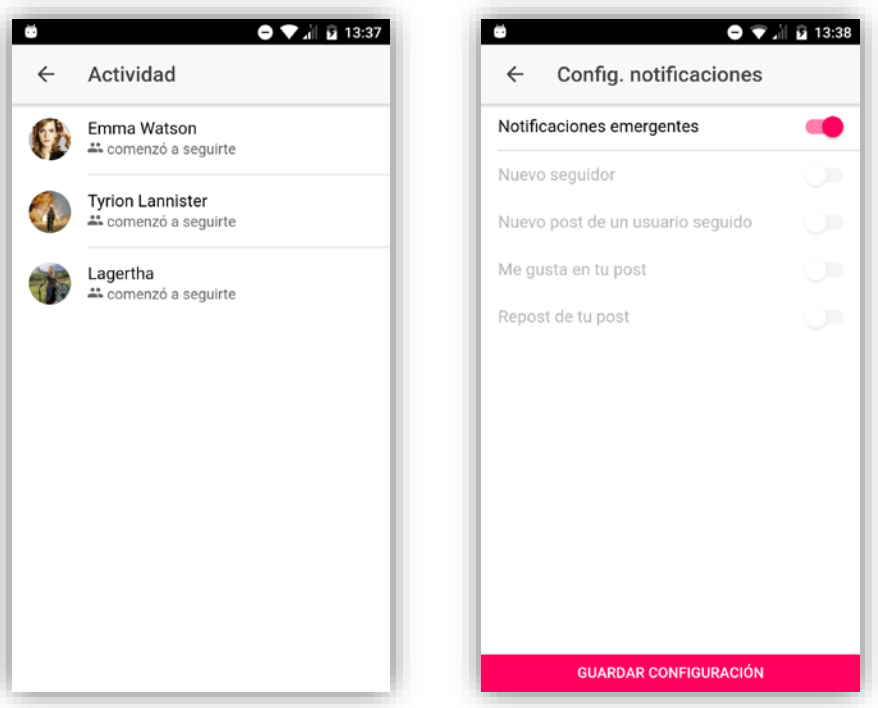


Ilustración 48 - Vista notificaciones y configuración de notificaciones

Para cerrar nuestra sesión, se nos pedirá una confirmación.

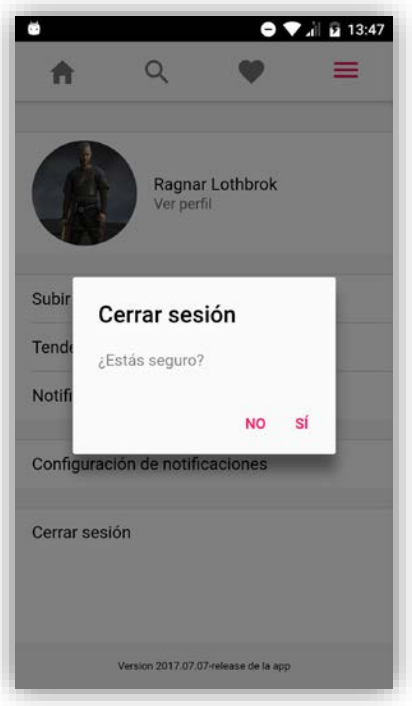


Ilustración 49 - Vista para cerrar la sesión

B) APLICACIONES HÍBRIDAS

Al ejecutarse con tecnologías web, los desarrolladores que ya tienen experiencia en el desarrollo en este medio pueden aprovechar sus conocimientos para lanzarse de una manera más rápida en el desarrollo de aplicaciones para móviles. Para hacer esto posible, las aplicaciones híbridas se ejecutan en lo que se denomina un 'web view', que no es más que una especie de navegador integrado en el móvil y en el que solamente se ejecuta la aplicación híbrida.

Las aplicaciones híbridas son interesantes por diversos motivos:

- Con una misma base de código serán capaces de compilar aplicaciones para funcionar correctamente en una gran cantidad de sistemas operativos de móviles o tablets. Generalmente nos será suficiente que nuestra aplicación funcione en iOS y Android, pero Ionic es capaz de compilar a otros sistemas como Windows Phone.

- El coste del desarrollo es sensiblemente menor, ya que no es necesario contar con varios equipos de desarrollo para cada lenguaje concreto de cada plataforma.

- El tiempo de desarrollo también es menor, ya que solo es necesario construir la aplicación una vez e inmediatamente la tendremos en todas las plataformas a las que nos dirigimos.

- Es de más fácil adaptación para los desarrolladores que vienen de programación web.

Pero todo no son ventajas, también podemos enumerar algunas desventajas en relación con las aplicaciones nativas (aquellas construidas con los lenguajes propios de cada sistema, Java para Android y Swift/Objective-C para iOS).

El rendimiento de una aplicación nativa suele ser mejor que el de una híbrida, aunque las híbridas han mejorado mucho en este sentido.

Al ejecutarse en un 'web view' dependemos de las tecnologías disponibles para el desarrollo web, que pueden ser menos ricas y potentes que las disponibles en nativo.

En las aplicaciones nativas trabajamos directamente con el hardware del teléfono, mientras que en las híbridas dependemos de *plugins* para su acceso. Esto puede derivar en problemas, pero afortunadamente cada vez son menores.

C) CAMERA

La función 'Camera.getPicture()' abre por defecto la aplicación de la cámara del dispositivo que permite a los usuarios tomar imágenes de forma predeterminada. Este comportamiento se produce cuando el parámetro 'sourceType' es igual a 'Camera.PictureSourceType.CAMERA'. Una vez que el usuario confirme la foto, la aplicación de la cámara se cierra y la aplicación se restaura como estaba anteriormente.

Si el parámetro 'sourceType' es 'Camera.PictureSourceType.PHOTOLIBRARY' o 'Camera.PictureSourceType.SAVEDPHOTOALBUM', aparece un cuadro de diálogo que permite a los usuarios seleccionar una imagen existente.

Existen más parámetros opcionales para personalizar los ajustes de la cámara. Dos de ellos, bastante importantes a la hora de tratar con imágenes en la nube, son 'targetWidth' y 'targetHeight' que limitan las dimensiones de la imagen para que a la hora de subirlas o descargarlas al navegar por la aplicación, el usuario no experimente retardos excesivos. En la documentación de Ionic Native Camera se pueden ver detenidamente.