Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de una aplicación móvil híbrida para la solicitud de canciones

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Carlos Fernández Samper

Tutor: Manuela Albert Albiol
Cotutor: María Victoria Torres Bosch

2020-2021

“Algo puede suceder si primero has determinado que es posible que suceda”

Elon Musk

Resumen

Este TFG consiste en el desarrollo de una **aplicación híbrida**, cuyo nombre provisional es **Mefis**, que tiene como objetivo ofrecer la posibilidad de solicitar canciones concretas al responsable de la música de un local. El modelo de negocio consiste en que dichas peticiones serán monetizadas. Un usuario solicitará una canción ofreciendo una cantidad que considere pertinente. El responsable de la música de dicho local aceptará o rechazará dicha petición. En caso de aceptar esta solicitud la transacción será repartida entre el propio responsable de la música, el local en el que se encuentre y el desarrollador de la app. La aplicación se desarrollará empleando el framework de desarrollo **Ionic** de Angular, de forma que con un único desarrollo esta sea válida tanto para **iOS** como **Android**.

Palabras clave: app, móvil, iOS, Android, Ionic, Angular.

Abstract

This FDP consists of the development of a hybrid application with Mefis as provisional name, wich aims to offer the possibility of requesting specific songs from the person in charge of the music of a pub. The business model is that these requests will be monetized. A user will request a song offering a certain ammount and the person in charge of the music will acept or reject that petition. In case of accepting this request, the transaction amount will be divided between the same person in charge of the music, the pub and the developer of the app. The application will be developed using the Angular mobile development framework, Ionic, so that with a single development it will be valid for both iOS and Andorid.

Keywords: app, mobile, iOS, Android, Ionic, Angular.

Tabla de contenidos

Contenido

Índice de tablas

Índice de ilustraciones

1. Introducción

El ocio nocturno es una de las actividades más famosas que existen en la actualidad. Grupos de personas se reúnen en locales de música con el fin de socializar. Bailar y cantar son las principales tareas que realizan en dicha actividad. Ambas tienen una cosa en común: la música.

Y es que la música de un local, en adelante, discoteca, determina en gran medida el aforo de esta. Una discoteca con buena música garantiza, en la mayoría de ocasiones, una gran cantidad de clientes.

Es bastante común que, estos clientes, en un momento determinado deseen que suene un tipo de música en específico y, en particular, una canción. Sin embargo, para que esto se produzca, el cliente tiene que pedir personalmente la canción al DJ.

Llevar a cabo esta tarea actualmente resulta costosa ya que, o bien la distancia entre la cabina del DJ y la pista de baile es demasiado grande, o bien no es posible comunicarse con este.

Por ello, en este documento planteo una solución a este problema mediante el uso de la tecnología móvil, más concretamente mediante el uso de una aplicación.

1.1 Motivación

Antes de elegir un tema para la realización del trabajo de fin de grado, ya tenía claro que quería realizar una aplicación móvil. De hecho, el desarrollo de Software es uno de los principales motivos por el que decidí cursar la carrera de Ingeniería Informática. Es más, actualmente me dedico a ello profesionalmente en una empresa valenciana.

La idea sobre una aplicación para solicitar canciones surgió cuando me encontraba viendo una película inspirada en los años 80. En ella, un hombre entró a un bar y se acercó a un especie de tocadiscos. Una vez allí, introdujo una moneda, seleccionó un disco y al cabo de unos segundos, este comenzó a sonar por todo el bar.

Esta idea se vió reforzada cuando, una noche en una discoteca con un grupo de amigos, nos situamos muy cerca de la cabina del DJ de aquel local. Para nuestro asombro, cada vez era mayor la frecuencia y el número de personas que se acercaban a dicha cabina con el fin de solicitar una canción en concreto.

1.2 Objetivos

El principal objetivo de este TFG es el desarrollo de una aplicación que permita solicitar canciones a los responsables de la música de un local .

Además de este objetivo general, estos son los objetivos particulares a conseguir:

- Permitir a los usuarios registrarse en la aplicación, ya sea como DJ, dueños de un local o simplemente usuarios.
- Permitir a un usuario crear una petición de una canción determinada en un local determinado con un importe determinado
- Mostrar la lista de locales disponibles haciendo uso de la geolocalización
- Permitir a los dueños de un local registrar dicho local en la aplicación.
- Permitir a los usuarios DJ crear y borrar sesiones en un local determinado.

1.3 Impacto esperado

En esta aplicación hay tres claros actores, por lo que se va a proceder a explicar el resultado esperado para cada uno de ellos:

- Por un lado, el **usuario** que desea solicitar una canción determinada en un local determinado. Para este usuario, la existencia de una aplicación que permita realizar esto de forma sencilla y rápida, supondrá una gran satisfacción lo que se traducirá en un aumento de la frecuencia con la que el usuario realiza esta actividad.
- Por otro lado, mediante el uso de esta aplicación, un **DJ** podrá obtener ingresos extra en cada una de sus sesiones. Además, podrá publicitarse profesionalmente a través de la aplicación.
- Finalmente, y de forma análoga al segundo actor, el **dueño de un local** obtendrá ingresos extras, así como publicidad para el local.

1.4 Metodología

El desarrollo de este proyecto se ha llevado a cabo empleando una metodología incremental, lo que permite ir aumentando la complejidad de este en cada ciclo o incremento minimizando los riesgos. Esta metodología es ideal para proyectos individuales como este.

Cada ciclo de desarrollo de esta metodología está compuesto por diversas etapas:

- En primer lugar, se realiza un **análisis** del problema a resolver, así como del estado actual del proyecto y como este problema puede afectar a la integridad del proyecto.
- En segundo lugar, se diseña la solución a este problema teniendo en cuenta todo aquello que puda afectar al proyecto.
- En tercer lugar, se implementa dicha solución.

- Finalmente y en cuarto lugar, se valida el resultado obtenido y se compara con los objetivos del análisis inicial. Esto nos permitirá establecer cual será el siguiente ciclo.

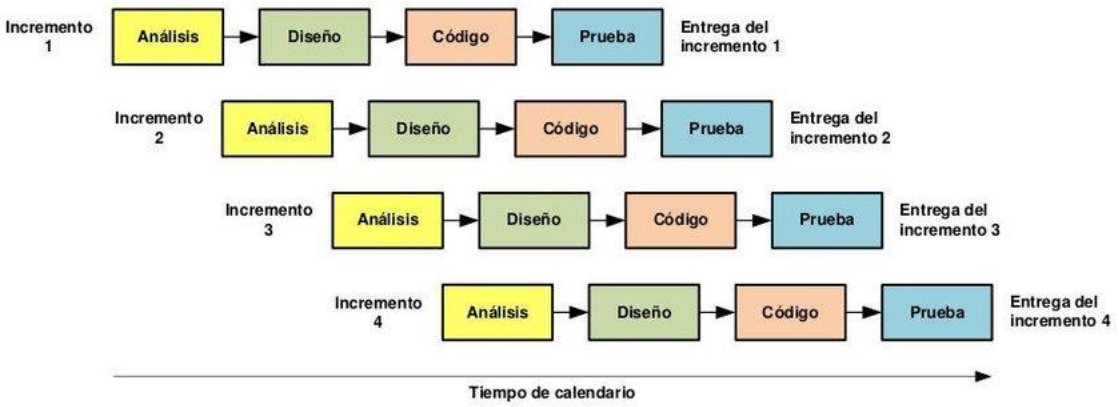


Ilustración 1. Metodología incremental

1.5 Estructura

Con el fin de facilitar la lectura de este documento, se adjunta una breve descripción de cada uno de los capítulos que aparecen en el.

- **Capítulo 2: Estado del arte**
  En este apartado se explicará el contexto actual referente a la aplicación desarrollada. Se analizará y valorarán las diferentes alternativas actuales.
- **Capítulo 3: Análisis del problema**
  En este capítulo se detallará la especificación de requisitos del proyecto así como diferentes diagramas que ayudarán a entender la solución que se quiere realizar.
- **Capítulo 4: Diseño de la solución**
  Se detallará la arquitectura y tecnologías utilizadas para el desarrollo del proyecto.
- **Capítulo 5: Desarrollo de la solución propuesta**
  En este apartado se incluirán las diferentes pantallas que componen la aplicación.
- **Capítulo 6: Implantación**
  El capitulo 6 expone como se va a desplegar el sistema para su explotación.
- **Capítulo 7: Conclusiones**
  En este apartado se discutirá si el proyecto desarrollado ha cumplido con los objetivos planteados así como la relación del trabajo desarrollado con los estudios cursados.
- **Capítulo 8: Trabajos futuros**
- **Capítulo 9: Referencias**

2. Estado del arte

En este apartado se explicarán las diferentes alternativas actuales existentes a nuestra aplicación desarrollada. Posteriormente se analizarán cada una de ellas y se compararán a la solución propuesta.

En primer lugar se ha realizado una búsqueda por las diferentes tiendas de aplicaciones de cada sistema operativo móvil disponible.Si atendemos a las aplicaciones existentes en Google Play Store**[2]** y en App Store**[3]**, no existe una aplicación que presente una funcionalidad similar a la descrita en este documento.

Las aplicaciones ofertadas relacionadas con reproducción de música tienen como objetivo crear una sala virtual común entre varias personas con el fin de reproducir la misma música. Permiten enviar nuevas canciones a la cola que acabarán reproduciéndose en dicha sala.

Todas estas aplicaciones se resumen en listas de reproducción compartidas. Ofrecen a los usuarios la posibilidad de añadir nuevas canciones que todos escucharán simultáneamente.

Sin embargo, un antiguo alumno de esta escuela tuvo una idea muy similar. Esta, sumada a la mencionada previamente sinfonola, constituyen las actuales alternativas a nuestra aplicación desarrollada.

2.1 Sinfonola

Como se ha mencionado en el primer capítulo de este documento, en los años 80 ya existían mecanismos para poder reproducir una canción en específico en un local. Estas máquinas se crearon en los años 60 y fueron muy populares durante 3 décadas hasta la creación de los discos compactos (CD).

Su nombre es sinfonola**,** aunque también se conoce como rocola, y su mecanismo es muy sencillo. Se introduce una moneda, se selecciona un disco y al cabo de unos segundos este comienza a sonar.

Ilustración 2. Sinfonola clásica

A partir de los años 90 el disco compacto sobrepasó al disco de vinilo en popularidad y cada vez los DJ eran más demandados en los locales. Por ello, el uso de las sinfonolas fue cada vez a menos.

Hoy en día es raro ver uno de estos aparatos en un local de música o un bar. Los pocos que existen, se han modernizado con respecto a la sinfonola original. Cuentan con pantalla táctil y con una librería de más de 50.000 títulos entre los que elegir. Aún así, no son muy populares entre la población joven.

Ilustración 3. Sinfonola moderna

2.2 DYOU

En el año 2016, Francisco Martí Polo, exalumno de la Escuela Ténica Superior de Ingeniería Informática de la Universidad Politécnica de Valencia desarrolló un proyecto de aplicación móvil**[1]** para enviar recomendaciones de canciones al DJ.

El funcionamiento de esta aplicación es muy similar al funcionamiento definido en este proyecto. Los usuarios podrán seleccionar entre una lista de locales para a continuación hacer una petición de una canción en específico en el local seleccionado.

Estas peticiones podrán ser vistas por el resto de usuarios los cuales, además, podrán votar dichas peticiones para que el DJ tenga constancia de las canciones más solicitadas.

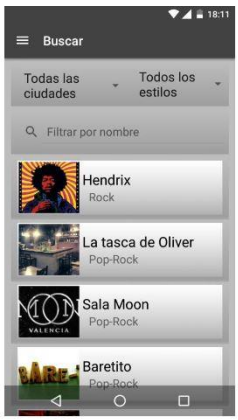


Ilustración 4. Aplicación DYOU

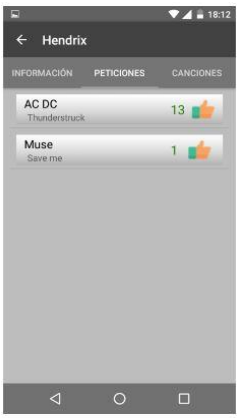


Ilustración 5. Aplicación DYOU

2.3 Análisis alternativas estudiadas

Por un lado, las sinfonolas ofrecen la posibilidad de reproducir una canción en específico en un local en específico. Sin embargo, el principal problema que presentan es que no otorgan una medida de control sobre la música que se reproduce en dicho local.

Un usuario cualquiera del sistema puede monopolizar la elección de canciones de dicho local pues no existe ningún mecanismo que controle esto. Basta con introducir el importe requerido de forma reiterada y todas tus peticiones automáticamente serán efectivas.

Por el otro lado, la aplicación que plantea Francisco Martí Polo es prácticamente idéntica a la aquí desarrollada. Permite buscar un local en específico y ver las diferentes peticiones que se han realizado en él.

El principal inconveniente es que no proporciona ningún sistema de gratificación al DJ de dicho local. Esto puede traducirse en que este DJ no tenga interés alguno en reproducir las canciones solicitadas por los usuarios, lo que puede provocar una baja aceptación de la aplicación.

3. Análisis del problema

Una vez conocidas las diferentes características que debe tener nuestra aplicación, en este apartado se detallará la información necesaria para llevar a cabo este proyecto. Para ello, utilizaremos el estándar IEEE 29148:2018**[4]** que define la estructura de especificación y gestión de requisitos.

3.1. Especificación de requisitos

3.1.1. Contexto de la aplicación

Antes de explicar el modelo de dominio, en este apartado se recogen todos los términos y conceptos relevantes en el contexto de la aplicación que se va a desarrollar. Se explicará la ontología del sistema así como la terminología técnica utilizada.

Ontología del sistema:

Término	Descripción
Usuario	Persona que accede a la aplicación mediante una cuenta creada
DJ	Persona responsable de la música de un local
Dueño de un local	Persona responsable de un local de música
Petición	Solicitud de una canción por parte de un usuario hacia un DJ
Local	Espacio físico donde se reproduce música
Sesión	Elemento que vincula a un DJ con un local durante un tiempo determinado y permite a los usuarios realizar peticiones a ese DJ durante ese tiempo
Canción	Canción asociada a una petición

Tabla 1. Ontología del sistema

Terminología técnica

Término	Descripción
Email	Correo electrónico utilizado por un usuario para registrarse en el sistema
Contraseña	Código secreto formado por caracteres para iniciar sesión en el sistema
Género musical	Categoría que reúne composiciones musicales que comparten distintos criterios de afinidad
Nombre artístico	Nombre que utiliza un DJ para identificarse públicamente
Nombre del local	Nombre asociado a un local de música para identificarse públicamente
Importe	Cantidad de dinero asociada a una petición
Url	Enlace único a la canción solicitada
Pinchar	Acción del DJ de ser el responsable de la música de un local determinado en una sesión determinada

Tabla 2. Terminología técnica

3.1.2. Modelo de dominio

Introducida la terminología a utilizar en el desarrollo del proyecto, a continuación se mostrarán las relaciones existentes entre dichos conceptos. Para ello, se hará uso del diagrama de clases:

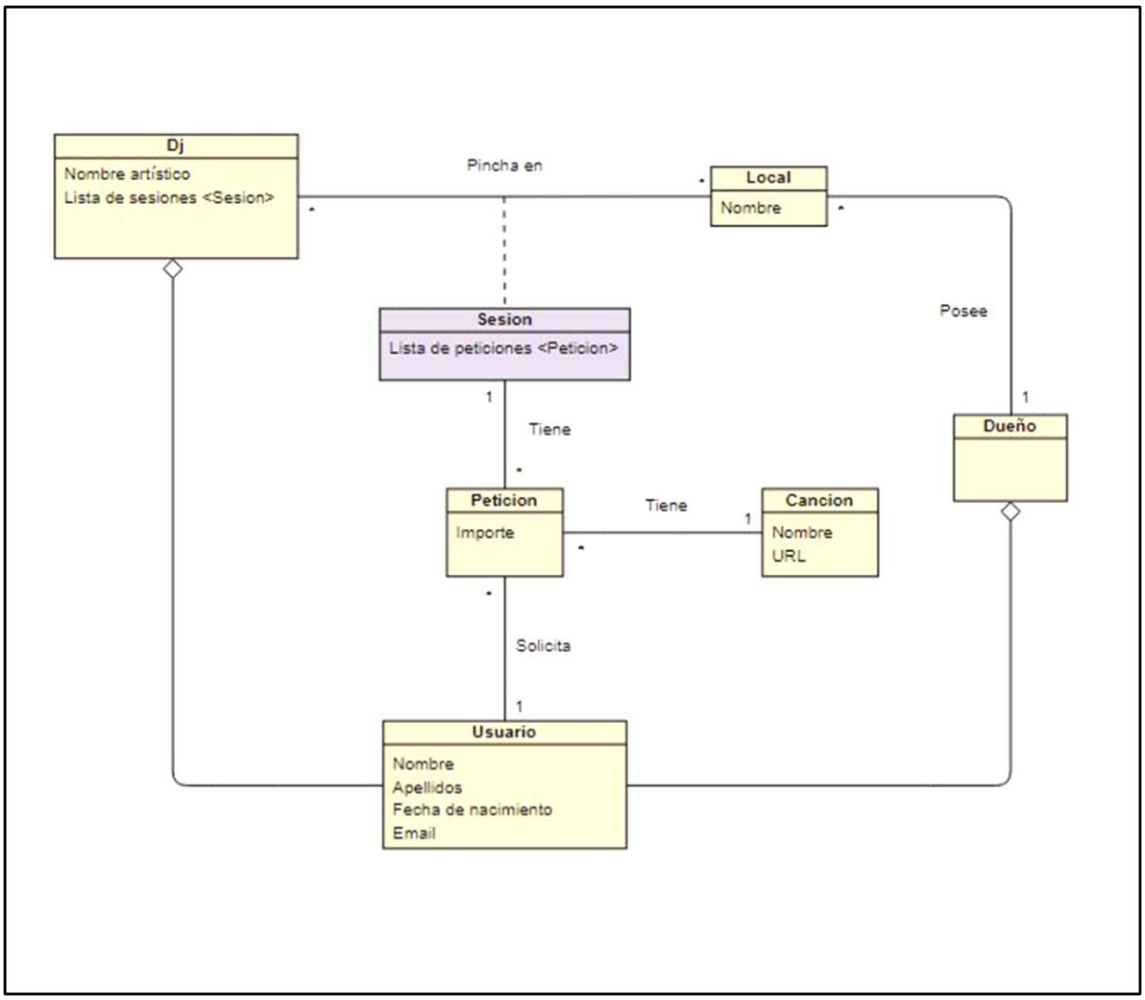


Ilustración 6. Diagrama de dominio

Para explicar las diferentes clases y atributos de estas, se adjunta una tabla para cada una de ellas:

Clase Usuario

Atributo	**Tipo**	**Descripción**
Nombre	String	Nombre del usuario.
Apellidos	String	Apellidos del usuario.
Fecha de nacimiento	Date	Fecha de nacimiento del usuario.
Email	String	Email único para cada usuario

Tabla 3. Clase Usuario

Clase DJ

Atributo	Tipo	Descripción
Nombre Artístico	String	Nombre público del Dj.
Lista de sesiones	List <Sesion>	Lista de sesiones disponibles para este Dj.

Tabla 4. Clase Dj

Clase Local

Atributo	Tipo	Descripción
Nombre	String	Nombre público del local de música.

Tabla 5. Clase Local

Clase Sesion

Atributo	Tipo	Descripción
Lista de peticiones	List <Peticion>	Lista de las peticiones creadas para esta sesión

Tabla 6. Clase Sesion

Clase Peticion

Atributo	Tipo	Descripción
Importe	Number	Cantidad de dinero vinculada a esta petición

Tabla 7. Clase Peticion

Clase Cancion

Atributo	Tipo	Descripción
Nombre	String	Título de la canción solicitada
Url	String	Enlace único a la canción determinada

Tabla 8. Clase Cancion

3.1.3. Límites del sistema

En este apartado se detallará el diagrama de contexto correspondiente que muestra una representación gráfica de alto nivel de los diferentes actores involucrados en el sistema.

Diagrama de contexto:

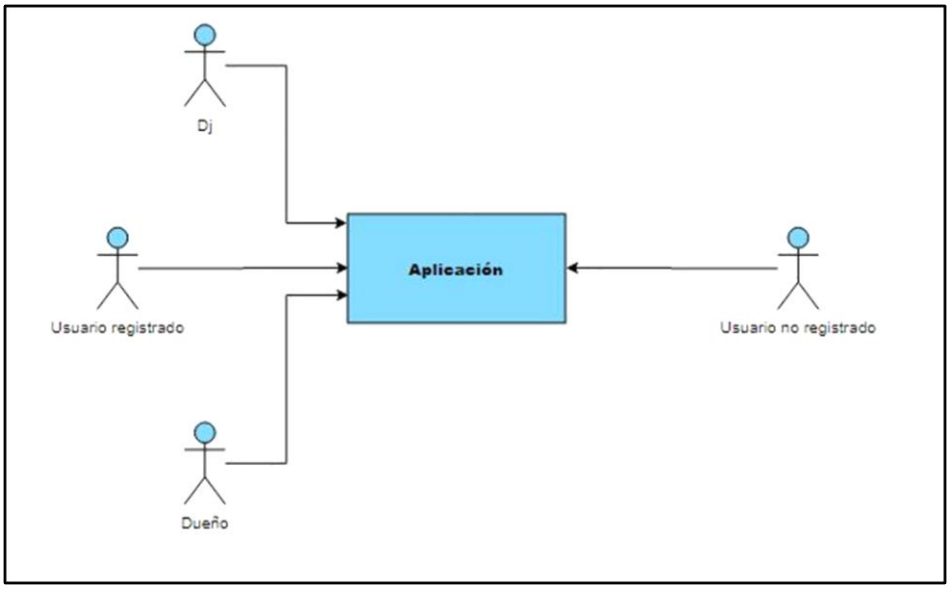


Ilustración 7. Diagrama de contexto

Descripción de los actores:

- **DJ**: usuario que utilizará la aplicación para crear sesiones en locales de música y aceptar peticiones. Será el encargado de aceptar o denegar dichas solicitudes.

- **Usuario registrado**: cualquier usuario registrado que no sea ni DJ ni dueño de un local. Podrá utilizar la aplicación para solicitar canciones determinadas en locales determinados.

- **Dueño**: responsable de un local de música. Utilizará la aplicación para gestionar sus locales dentro de esta.

- **Usuario no registrado**: cualquier persona que acceda a la aplicación sin haberse registrado previamente. Sólo podrá acceder a la pantalla de Login y Registro.

3.1.4. Características del sistema

Una vez introducido el contexto, a continuación estableceremos las características necesarias del sistema para cumplir los objetivos.

- **Registro de usuario:**

  Es necesario que se puedan registrar los usuarios para poder acceder a la funcionalidad de la aplicación. Se debe implementar también el registro de los dos tipos especializados de usuarios: DJ y dueño de un local.

- **Hacer peticiones:**

  Se debe permitir a los usuarios realizar diferentes peticiones de determinadas canciopnes para determinados DJ . Los usuarios podrán buscar una canción en específico a traves de un buscador que ofrecerá millones de títulos.

- **Crear sesiones:**

  Los DJ podrán crear diferentes sesiones para un determinado local. Estas sesiones tendrán una fecha y hora determinada.

- **Añadir local:**

  En cuanto a los dueños de locales, estos podrán añadirlos a la aplicación desde su perfil.

3.1.5. Casos de uso

Llegados a este punto, con toda la información descrita sobre la funcionalidad de la aplicación, se adjuntan los diferentes casos de uso que involucran a los actores definidos previamente

Usuario no registrado

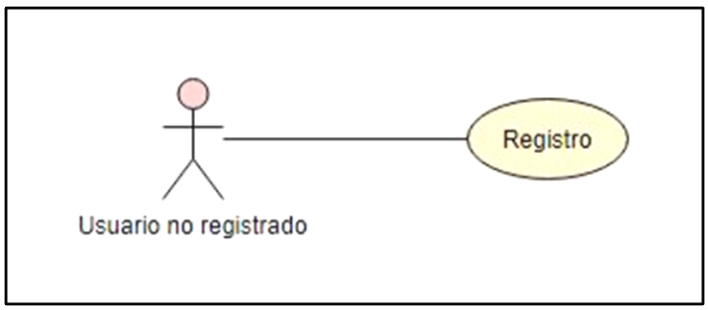


Ilustración 8. Diagrama caso de uso Usuario no registrado

Un usuario no registrado que acceda a la aplicación sólo podrá realizar la acción de registro.

Caso de uso 01	**Registro**
Descripción	Un usuario no registrado podrá registrarse a través de un formulario
Actores	Usuario
Precondición	No registrado
Secuencia Normal	1. Acceso al formulario de registro. 2. El usuario introduce sus datos y pulsa en "Registrar" 3. Se valida el registro.
Postcondiciones	El usuario deberá haber sido registrado en la base de datos
Excepciones	El usuario introduce datos incorrectos.
Comentarios	La contraseña debe tener letras, números y símbolos.

Tabla 9. Caso de uso Registro

Usuario registrado

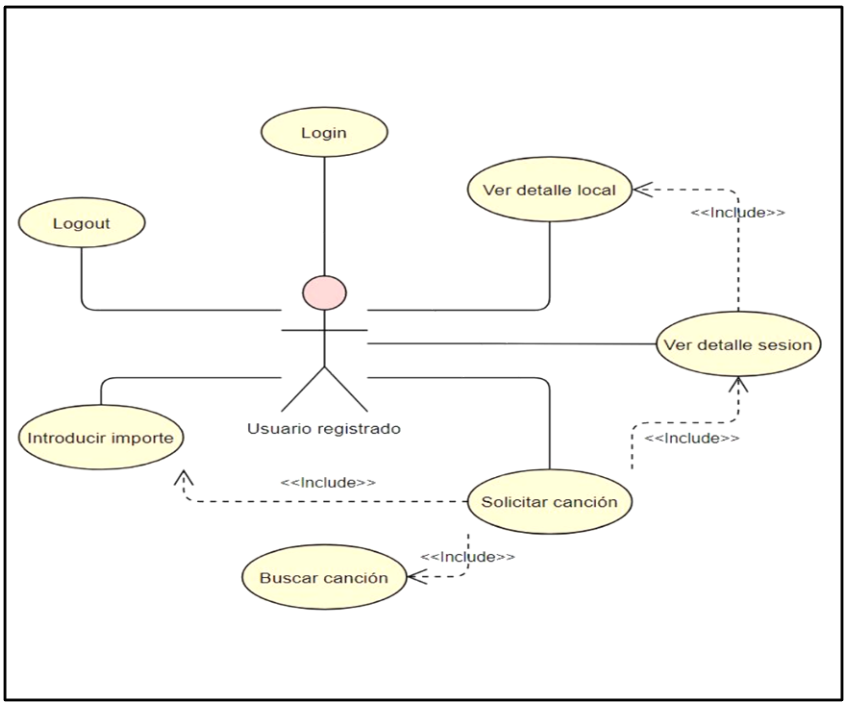


Ilustración 9. Diagrama caso de uso Usuario registrado

Caso de uso 02	**Login**
Descripción	Un usuario registrado introduce sus datos para acceder a la aplicación.
Actores	Usuario
Precondición	Registrado
Secuencia Normal	1. Introduce Email y Contraseña 2. Pulsa en "Login" 3. Se valida el inicio de sesión
Postcondiciones	El usuario deberá estar logeado y puede ver los artistas
Excepciones	El usuario introduce datos incorrectos.
Comentarios	

Tabla 10. Caso de uso Login

Caso de uso 03	**Logout**
Descripción	El usuario cierra su sesión actual.
Actores	Usuario
Precondición	Haber hecho login.
Secuencia Normal	1. El usuario pulsa en "Cerrar sesión" 2. La aplicación redirige al usuario a la pantalla de Login
Postcondiciones	El usuario es redirigdo a la pantalla de login
Excepciones	
Comentarios	

Tabla 11. Caso de uso Logout

Caso de uso 04	**Ver detalle local**
Descripción	Un usuario podrá seleccionar un local para ver su detalle
Actores	Usuario
Precondición	Usuario logeado
Secuencia Normal	1. El usuario pulsa sobre un local. 2. El usuario es redirigido a la página del local
Postcondiciones	El usuario es redirigido a la página del local
Excepciones	
Comentarios	En la página se mostrará el nombre artístico de los artistas que estén pinchando en dicho local

Tabla 12. Caso de uso Ver detalle local

Caso de uso 05	**Solicitar una canción**
Descripción	El usuario solicita una nueva petición de canción
Actores	Usuario
Precondición	Usuario logeado Exista un DJ pinchando en almenos un local
Secuencia Normal	1. El usuario selecciona un local 2. El usuario pulsa sobre un artista 3. El usuario pulsa sobre “Nueva petición”
Postcondiciones	El usuario es redirigido a la pantalla de peticiones
Excepciones	
Comentarios	Al pulsar el botón atrás se deberá devolver al usuario a la pantalla del detalle de la sesión

Tabla 13. Caso de uso Solicitar una canción

Caso de uso 06	**Buscar una canción**
Descripción	El usuario busca una canción determinada para una petición
Actores	Usuario
Precondición	El usuario ha solicitado una nueva petición
Secuencia Normal	1. El usuario introduce un texto en el buscador 2. El usuario pulsa “Buscar” 3. Se muestran los resultados de la búsqueda
Postcondiciones	
Excepciones	La búsqueda no tiene resultados
Comentarios	Los resultados deberán mostrar una imagen del álbum

Tabla 14. Caso de uso Buscar una canción

Caso de uso 07	**Introducir importe**
Descripción	El usuario introduce el importe deseado para la canción solicitada
Actores	Usuario
Precondición	El usuario ha seleccionado una canción de la lista de resultados tras la búsqueda
Secuencia Normal	1. El usuario selecciona una canción de la lista de resultados. 2. El usuario introduce el importe deseado y pulsa en “Solicitar” 3. Se crea una nueva petición de canción para el artista seleccionado.
Postcondiciones	Se registra una nueva petición en la base de datos para el artista seleccionado.
Excepciones	Ya existe una petición para dicha canción con un importe mayor.
Comentarios	

Tabla 15. Caso de uso Introducir Importe

Caso de uso 08	**Ver detalle sesión**
Descripción	Un usuario podrá seleccionar una sesión para ver su detalle.
Actores	Usuario
Precondición	Usuario logeado Exista almenos un DJ con una sesión activa.
Secuencia Normal	1. El usuario pulsa sobre un local. 2. El usuario selecciona un artista
Postcondiciones	El usuario es redirigido a la página detalle de la sesión
Excepciones	No existe ningún Artista con una sesión activa.
Comentarios	En la página se mostrará el nombre artístico del DJ así como las diferentes peticiones para dicho DJ.

Tabla 16. Caso de uso Ver detalle sesion

Usuario Dj

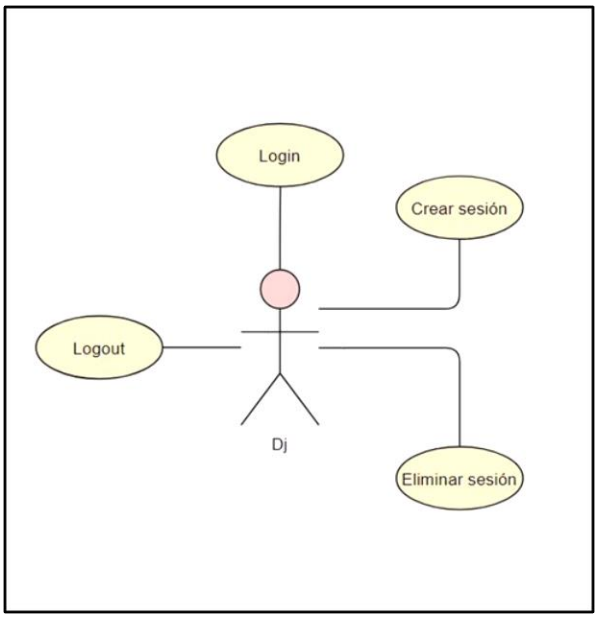


Ilustración 10. Diagrama casos de uso Usuario DJ

Dado que los actores DJ y Dueño de un local son a su vez un usuario registrado, se va a obviar la definición de los casos de uso Login y Logout correspondientes.

Caso de uso 09	**Crear sesión**
Descripción	Un usuario DJ registrado crea una nueva sesión para un local
Actores	DJ
Precondición	Haber hecho login en la app
Secuencia Normal	1. Pulsa sobre "Añadir nueva sesión" 2. Rellena los datos de la nueva sesión 3. Pulsa en Añadir
Postcondiciones	El usuario DJ al acceder a la app debería ver la nueva sesión añadida
Excepciones	El usuario DJ introduce un local de música no registrado en la app
Comentarios	

Tabla 17. Caso de uso Crear sesion

Caso de uso 10	**Eliminar sesión**
Descripción	Un usuario DJ registrado elimina una sesión disponible
Actores	DJ
Precondición	Haber hecho login en la app Disponer de al menos una sesión programada
Secuencia Normal	1. El usuario pulsa sobre el icono de eliminar para la sesión dada
Postcondiciones	La sesión se eliminará del listado de sesiones disponibles
Excepciones	
Comentarios	

Tabla 18. Caso de uso eliminar sesion

Usuario dueño de un local

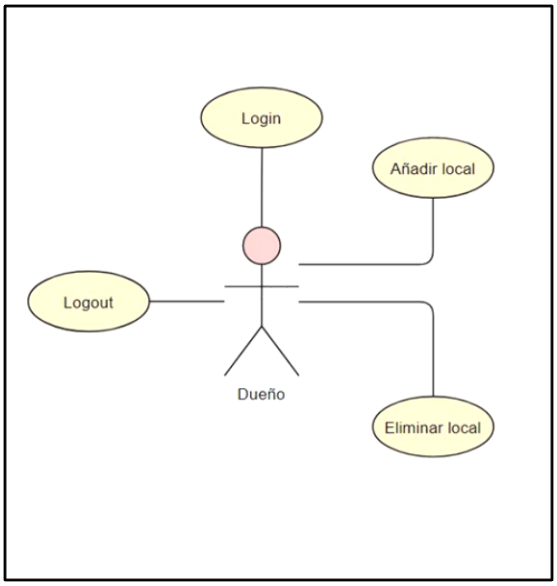


Ilustración 11. Diagrama casos de uso usuario dueño de un local

Caso de uso 11	**Añadir local**
Descripción	Un usuario dueño de un local añade un nuevo local a la aplicación
Actores	Dueño
Precondición	Registrado Haber hecho login en la app
Secuencia Normal	1. Pulsa sobre “Añadir nuevo local” 2. Rellena la información del nuevo local 3. Pulsa en “Añadir”
Postcondiciones	El nuevo local debe aparecer en su lista de locales
Excepciones	
Comentarios	

Tabla 19. Caso de uso Añadir local

Caso de uso 12	**Eliminar local**
Descripción	Un usuario dueño de un local elimina un local de la aplicación
Actores	Dueño
Precondición	Haber hecho login. Poseer al menos un local.
Secuencia Normal	1. El usuario pulsa sobre el icono de gestionar un local 2. El usuario pulsa sobre "Eliminar local" 3. El usuario pulsa en "Confirmar eliminar local"
Postcondiciones	El local es eliminado de la aplicación
Excepciones	
Comentarios	

Tabla 20. Caso de uso Eliminar local

3.1.6. Análisis de riesgos

En este apartado se detallarán los diferentes posibles riesgos que puedan afectar al usuario o a la infraestructura.

Título	Un DJ acepta las canciones pero no las reproduce
Descripción	La persona responsable de la música de un local podría aceptar las peticiones de canciones y cobrar dicha petición pero finalmente no reproducirla en el local.
Impacto	Mala experiencia con la aplicación y caída de usuarios.
Medidas	Incorporar un sistema de puntuaciones a los DJ con el fin de identificar a aquellos que frecuentan esta acción

Tabla 21. Tabla 1 análisis de riesgos

Título	Sistema de pagos complejo
Descripción	Los usuarios podrían cancelar una petición si el sitema de pago de la app no es adecuado, rápido e intuitivo.
Impacto	Bajo número de transacciones
Medidas	Realizar un estudio de UX para diseñar un pago de solicitudes intuitivo y rápido.

Tabla 22. Tabla 2 análisis de riesgos

Título	Rechazo de pagos por móvil
Descripción	Algunos usuarios podrían rechazar el realizar pagos por el móvil y preferirían pagar en efectivo.
Impacto	Bajo número de transacciones
Medidas	Crear tarjetas de pago llamadas “Mefis Points” que los usuarios puedan comprar físicamente en el local y canjear en la app.

Tabla 23. Tabla 3 análisis de riesgos

Título	Gestión de la base de datos
Descripción	La persona responsable de gestionar la base de datos podría tener problemas debido al volumen de datos generado
Impacto	Ralentización de los servicios de la aplicación e incluso inutilidad de esta
Medidas	Servicios backup para preservar la disponibilidad continua de la app.

Tabla 24. Tabla 4 análisis de riesgos

3.2. Identificación y análisis de soluciones posibles

Una vez definidas todas las necesidades para nuestra aplicación, en este apartado se detallarán las diferentes opciones planteadas para la implementación de este proyecto.

En primer lugar se tratará el tema de la gestión de usuarios. Desde un inicio se tenía una idea clara sobre el Login y Registro de nuevos usuarios. Para ello, se haría uso de un token de autenticación mediante el extendido protocolo Oauth**[5].**

Una vez decidido la forma de gestionar los usuarios, tocaba buscar la tecnología adecuada para ello. Tras indagar en la web, en una primera instancia encontré Amazon Cognito**[6].** Esta herramienta de Amazon permite gestionar de forma fácil el control de los usuarios de una aplicación. Dado que en un proyecto anterior en el curso utilicé MongoDB**[7]** como base de datos, estas dos tecnologías serían mi propuesta inicial.

Sin embargo, poco después di con la suite de Google Firebase**[8]**. En un principio solo me interesé en esta por la autenticación de usuarios. Pero tras explorar su plataforma, observé que además incluía un módulo que permitía gestionar una base de datos NoSQL**[9].**

De esta forma no tendría que incorporar una segunda tecnología para la gestión de la base de datos, pues Google Firebase me permitía implementar ambas desde la misma plataforma. Podría utilizar esta plataforma como el backend casi completo de la aplicación. Por un lado me permitía gestionar los usuarios y por el otro servía al mismo tiempo como base de datos para la aplicación.

Para continuar con el backend y sus servicios, sólo faltaba buscar una forma de obtener una librería de canciones que estuviese actualizada, que fuera internacional y que fuera rápida. Tras una búsqueda rápida en Google el primer resultado fue Shazam API**[10]**.

Todas las opciones se basaban en peticiones API REST**[11]**. Sin embargo, quería asegurarme que la API que escogiese tuviera la lista de canciones, así como sus remixes, más actualizada posible. Es por eso que la API de Spotify**[12]** era la clara candidata.

Finalmente, tuve que decidir que framework de desarrollo utilizar para la implementación de la aplicación. En este apartado no tuve muchas dudas, pues durante el curso realicé, junto a mis compañeros, una aplicación móvil empleando como framework de desarrollo Angular**[13]**. Además, este framework permite la creación de una aplicación móvil híbrida

3.3. Solución propuesta

Una vez analizadas las diversas opciones disponibles para el desarrollo del proyecto, en este apartado se especificará la solucion final propuesta.

Como framework de desarrollo se utilizará Angular y, más concretamente, su framework Ionic**[14]** el cual permite crear una aplicación móvil. El lenguaje de programación será, por tanto, TypeScript**[15]**. Este último es un lenguaje basado en el famoso JavaScript**[16]**, aunque con tipado fuerte.

Para gestionar el registro y acceso por parte de los usuarios, se utilizará el módulo Firebase Authentication de Firebase. Este módulo de firebase se acompañará del módulo Firebase Firestore el cual permitirá gestionar una base NoSQL.

Como se ha comentado en el apartado anterior, la librería utilizada para realizar la búsqueda de canciones será Spotify API.

Finalmente, el editor de código será Visual Studio Code**[17]** y se empleará Github**[18]** como repositorio Git.

4. Diseño de la solución

Una vez definidos los diferentes requisitos del sistema, se va a detallar a continuación el diseño de la solución propuesta.

4.1 Arquitectura del sistema

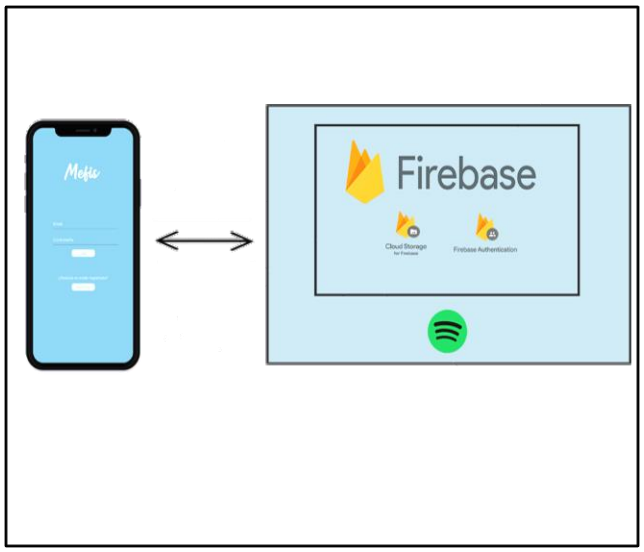


Ilustración 12. Arquitectura software del sistema

La arquitectura software del proyecto está reflejada en la ilustración anterior. Por un lado, el dispositivo móvil en el que se encuentra instalada la aplicación y será el objeto de interacción del usuario con el sistema.

Como se ha mencionado en el capítulo 3.3, el framework de desarrollo será Angular. Este se caracteriza por utilizar un patrón Modelo – Vista – Controlador aunque no es realmente así.

Una de las características de Angular es que permite sincronizar los datos de la vista y el modelo (two-way data binding). Es decir, proporciona la habilidad de modificar un dato en la vista y que este dato cambie simultáneamente en el modelo, por lo que la independencia que se produce en un modelo-vista-controlador clásico no se reproduce en este caso.

Es por esto que no se puede afirmar que el patrón utilizado es MVC (Modelo Vista Controlador) si no que se podría identificar como MVVM (Modelo Vista Vista Modelo).

Por el otro lado, los servicios del sistema que interactúan con la aplicación. Los módulos de Firebase encargados de gestionar el registro y acceso de los usuarios así como de proveer a la aplicación toda la información de la base de datos y la librería de Spotify que permite realizar la búsqueda de canciones.

4.2 Diseño detallado

En este apartado se detallará el diseño interno de la aplicación. Se ilustrarán los diferentes módulos o componentes internos de la misma.

Otra de las principales características de Angular son los componentes. En una aplicación Angular cada ventana o vista de la aplicación corresponde a un componente el cual dispone de su propia lógica (y vista) y es además independiente del resto de componentes.

Esto se traduce en que cada módulo de la aplicación es independiente del resto, lo que garantiza un gran desacoplamiento.

A continuación se van a explicar los diferentes módulos o componentes que conforman la aplicación:

- **Login**: la pantalla de Login de la aplicación. Su lógica permite acceder a un usuario ya registrado mediante unas credenciales determinadas.

- **Register**: módulo de registro. Permite a un usuario registrarse en el sistema al introducir una serie de datos requeridos. Este componente además tiene otro sub-componente:

    - **Register-verify**: corresponde con la pantalla de confirmación de registro. Cuando un usuario se registra en la aplicación debe confirmar dicho registro mediante un correo electrónico.

- **Home**: la pantalla principal de la aplicación. Cualquier usuario registrado accederá a ella tras hacer login. En esta pantalla se mostrarán los diferentes locales disponibles en los que se podrán realizar peticiones de canciones. Si se accede con un usuario Dj, se mostrarán las diferentes sesiones asociadas a ese Dj o, en su defecto, podrá crear una. Finalmente, si se accede como dueño de un local, se mostrarán los diferentes locales asociados a ese usuario, así como la posibilidad de gestionar cada uno de ellos.

- **Local**: esta pantalla corresponde al detalle de un local. En ella se muestra información de un local así como las diferentes sesiones que se están llevando o llevarán a cabo en él.

- **Session**: este módulo corresponde a una sesión determinada en un local determinado con un Dj determinado. Mostrará información de este Dj así como el local asociado y mostrará además las diferentes peticiones llevadas a cabo por los usuarios.

- **Request:** la pantalla de peticiones. En enviar una nueva petición con un importe determinado. Al igual que con el módulo de Register, este módulo cuenta con otro componente extra:

    - **Request-result:** corresponde con la pantalla que muestra los resultados de la búsqueda de una canción.

Otra característica de los componentes de Angular es que pueden integrase dentro de otro componente, ampliando la funcionalidad de este. Podemos crear así componentes que carezcan de interfaz o vista pero su lógica pueda aprovecharse en otros módulos de la aplicación. Estos componentes se les conoce como servicios.

Los servicios en angular son componentes con patrón Singleton que permiten añadir contexto global a la aplicación. Estos son los que se encargan de hacer peticiones a los servicios de BackEnd y ofrecerlos al resto de componentes.

Así, estos son los diferentes componentes servicios que conforman la aplicación:

- **Auth service:** este es el servicio que gestiona el login y registro de usuarios. Este servicio hace uso de los servicios de Firebase Authentication y es utilizado en los módulos Login y Register.

- **Artists service:** este servicio provee a la aplicación de toda la información necesaria relativa a los Djs, locales, peticiones y sesiones. Extrae la información de la base de datos del módulo Firebase Firestore y lo ofrece a los módulos Home, Local, Session y Request de la aplicación.

- **Spotify service:** servicio que permite a la aplicación hacer uso de la API de Spotify y realizar búsquedas de canciones determinadas. Es utilizado por el módulo o componente Request y más concretamente por Request-result.

Finalmente se adjunta una ilustración con los diferentes componentes que conforman la aplicación a fin de quede más claro el diseño interno de la aplicación:

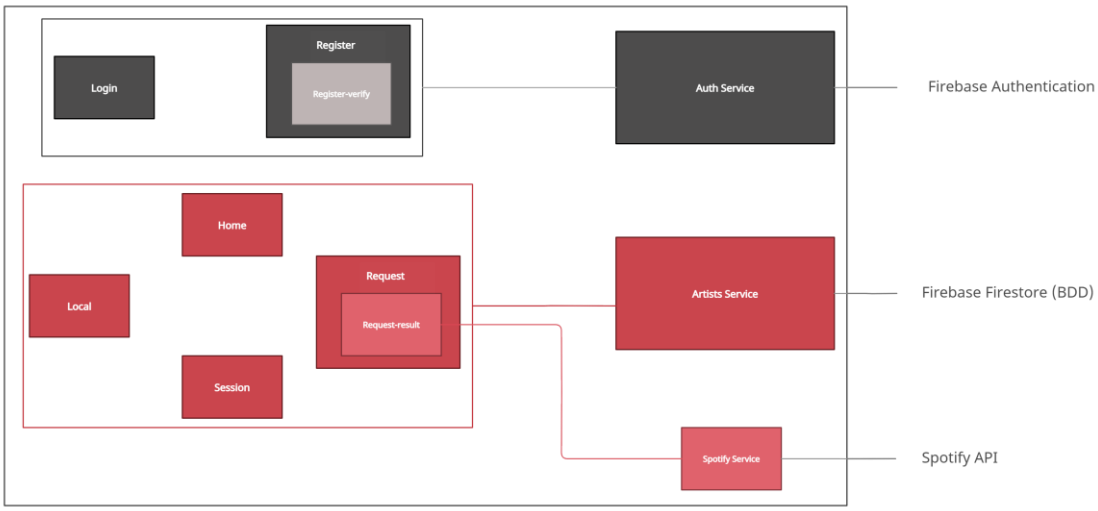


Ilustración 13. Arquitectura interna de la aplicación

4.3 Tecnología utilizada

En esta sección se presentarán las diferentes tecnologías utilizadas para el desarrollo del proyecto.

4.3.1 Angular

Angular es un framework de código abierto desarrollado por Google para facilitar la creación de aplicaciones wb de una sola página, también llamadas SPA**[19]** (Single Page Application). Como se mencionaba anteriormente, una de las características más importantes de Angular es que sus aplicaciones se basan en componentes que controlan la vista y la lógica de forma independiente los unos de los otros.

Esto permite además la creación de componentes que carecen de interfaz pero hacen de mediadores entre el FrontEnd y el BackEnd de la aplicación y pueden integrarse en cualquier otro componente.

Para la creación de aplicaciones móviles, Angular cuenta con un framework especializado para ello, llamado Ionic. Ionic es un SDK de frontend de código abierto que permite desarrollar aplicaciones híbridas, es decir, aplicaciones para iOS, Android y Web desde el mismo código.

Para ello utiliza un compilador llamado Capacitor**[20]** que traduce nuestro código base en el correspondiente a cada plataforma, de forma que podemos desarrollar funcionalidades nativas para cada una (uso de cámara, almacenamiento, gestos táctiles…etc)

4.3.2 Firebase

Firebase es una plataforma en la nube para el desarrollo de aplicaciones web y móvil, creada en 2011 y adquirida por Google en 2014. En un principio básicamente consistía en una base de datos en tiempo real, pero con el tiempo fueron añadiendo más funcionalidades hasta agrupar los SDK de varios productos de Google en una misma plataforma. De esta forma, encontramos en la misma plataforma las siguientes funcionalidades: desarrollo, crecimiento, monetización y análisis. A continuación, se describirán algunas de estas funcionalidades para cada categoria:

- **Desarrollo**: Firebase permite gestionar la autenticación de usuarios. Ofrece un sistema de autenticación que permite el acceso y el registro de estos. Por otro lado, Firebase ofrece también un servicio de base de datos en tiempo real No SQL, así como almacenamiento en la nube.

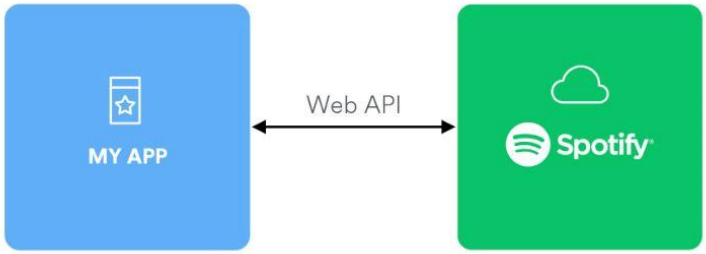
- **Crecimiento**: Firebase permite integrar la aplicación en los resultados de búsqueda de Google, lo que ayuda en cuanto al crecimiento de esta. Incluye, además, links inteligentes que redirigen al usuario a contenidos concretos de la aplicación, ideal para campañas publicitarias.

- **Monetización**: AdMob permite integrar publicidad en nuestras aplicaciones de forma sencilla. Todo ello gestionado desde la misma plataforma.

- **Análisis**: Firebase incluye diversas analíticas a incorporar en nuestra aplicación con el fin de realizar un seguimiento de las actividades de los usuarios

Para nuestro proyecto solo hemos hecho uso de las herramientas de la categoría de desarrollo, ya que es la fase en la que nos encontramos. Sin embargo no se descarta utilizar el resto de herramientas que ofrece esta plataforma en las diferentes fases que experimentará la aplicación.

4.3.3 Spotify API



Una API(application programming interface) es una librería o herramienta que los desarrolladores de un producto ofrecen para que puedan utilizar sus servicios. En este caso, Spotify ofrece una API para integrar sus servicios en nuestra aplicación.

Esta API de Spotify nos permite realizar búsquedas de casi cualquier contenido músical. Artistas, álbumes, listas de reproducción y, sobre todo, canciones. Ofrece, además, la posibilidad de integrar su reproductor de música en nuestra aplicación.

4.3.4 Typescript

TypeScript es un lenguaje de progrmación lanzado en Octubre de 2012 basado en el famoso lenguaje JavaScript. A diferencia de este, TypeScript agrega funcionalidad y sintaxis orientada a objetos y clases como en Java o C. Es el lenguaje que se utiliza en el framework de desarrollo Angular.

4.3.5 Visual Studio Code

Para el desarrollo del proyecto, se ha utilizado la herramienta Visual Studio Code como editor de código fuente. Cabe diferenciar esta herramienta del IDE de desarrollo Visual Studio**[23]**. Se ha elegido esta ya que no es necesario una Suite completa como Visual Studio para desarrollar una aplicación mediante el framework Angular.

Visual Studio Code nace en 2015 de la mano de Microsoft. Es gratuito y de código abierto. La principal característica de este editor es que compatible varios lenguajes de progrmaación. Mediante plug-ins, es capaz de extender su funcionalidad en función de las necesidades del desarrollador. Así, incluye soporte para la depuración, control de versiones de Git, resaltado de sintáxis, autocompletado inteligente de código, fragmentos y refactorización de código.

5. Desarrollo de la solución propuesta

El desarrollo de la solución propuesta anteriormente se ha llevado a cabo empleando una metodología incremental.

Tras decidir las tecnologías a emplear, el primer paso fue crear un proyecto de Angular vacío. Este proyecto vacío se subió a un repositorio de Github con el fin de poder llevar a cabo un control de versiones de este. Aunque sea el único desarrollador de la aplicación, es una buena práctica.

El siguiente paso fue diseñar la pantalla de Login y Registro de la aplicación, incluyendo sus respectivos componentes en el proyecto:

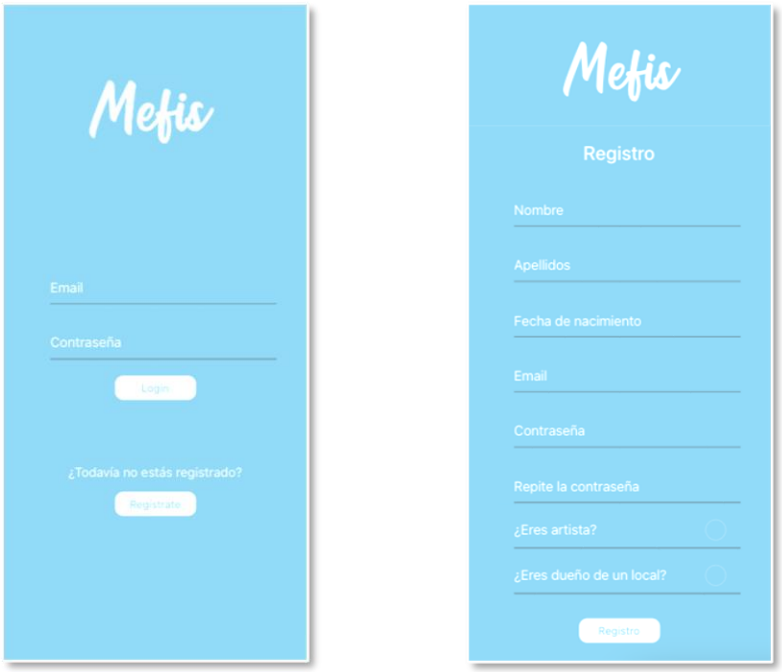


Ilustración 14. Pantallas Login y Registro

Una vez implementadas ambas pantallas, se añadió el componente servicio AuthService que hace uso el módulo Auth de firebase para la gestión de usuarios.

El siguiente componente fue la pantalla verificación de registro. En un primer momento la verificación no era necesaria, pero dado su facilidad de implementación se decidió incluirla.

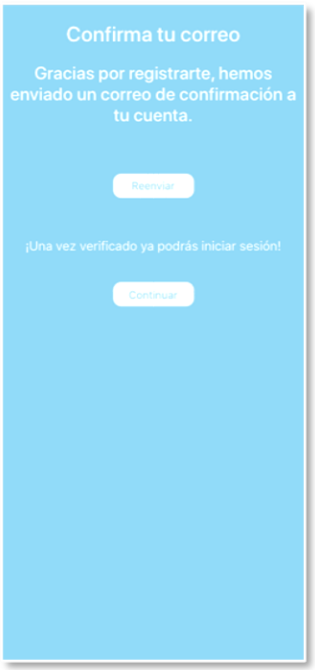


Ilustración 15. Pantalla verificación registro

Dado que ya podía registrar usuarios, se implementó el componente Home. En este punto se creó, además, el servicio ArtistService. En este momento se pobló la base de datos NoSQL con datos de prueba con el fin de validar el ultimo desarrollo.

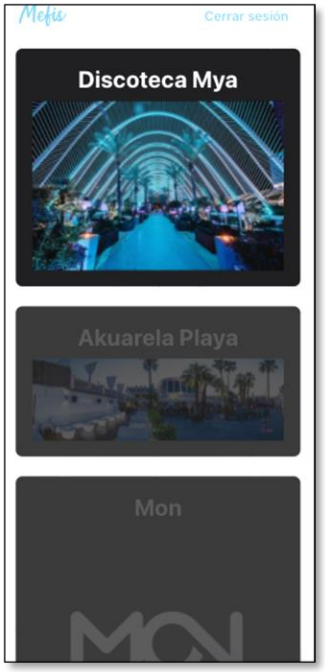


Ilustración 16. Pantalla Home usuario

El siguiente paso consistió en la creación de los componentes Local y Artist. Dado que ya se había incluído la implementación del módulo Firestore de Firebase, se pudo reflejar la información de los locales y artistas de forma inmediata:

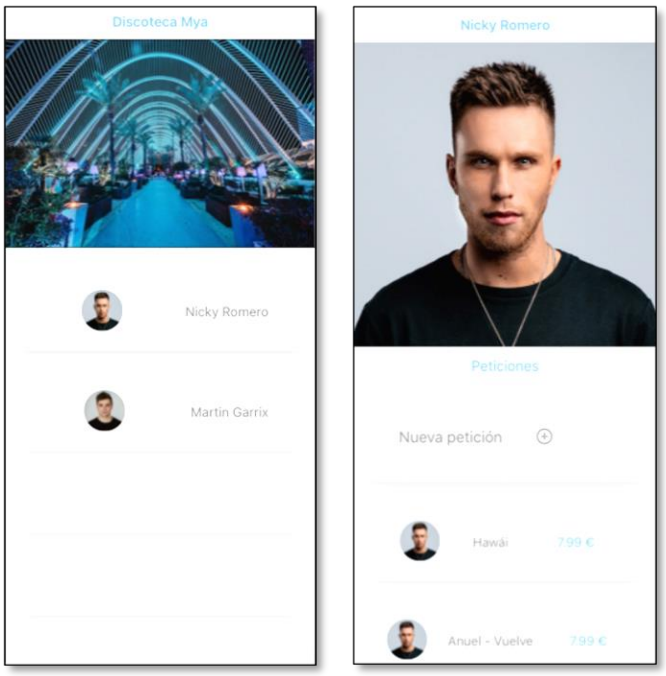


Ilustración 17. Pantallas Local y Artista

Lo siguiente fué la implementación de los componentes Request y RequestResults. En este ciclo de desarrollo fue necesario implementar, además, la integración de la API de Spotify en nuestra aplicación, por lo que se creó un tercer componente servicio llamado SpotifyService:

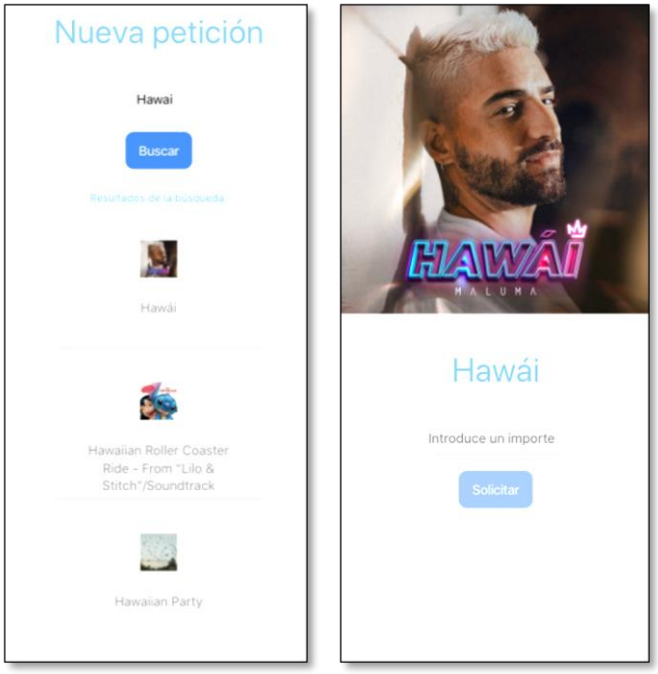


Ilustración 18: Pantallas Request y RequestResult

Llegados a este punto la aplicación ya contaba con la funcionalidad completa para un usuario corriente. Quedaba por implementar pues, en el último ciclo de desarrollo, la funcionalidad de los DJ y dueños de un local. Para ello se modificó el componente Home para que mostrara una información u otra en función del tipo de usuario que accedía a ella:

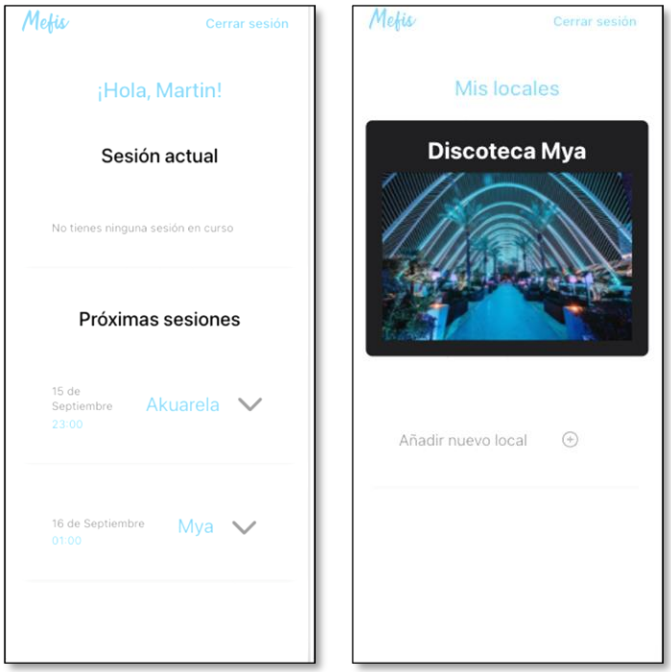


Ilustración 19. Pantallas Home para usuario DJ y usuario dueño de un local

6. Implantación

Si atendemos al modelo de negocio y funcionalidad de la aplicación, no es necesario realizar ninguna instalación de hardware.

Desde la vista del usuario, basta con acceder a la tienda de aplicaciones de su dispositivo e instalar la aplicación en cuestión. Una vez registrado podrá utilizarla completamente. En un primer momento no habrán muchos locales registrados y esto puede provocar una baja aceptación.

Con respecto a los Djs, de igual forma que los usuarios, únicamente tendrán que descargar la aplicación y registrarse en ella. A partir de entonces podrán crear sesiones. Sin embargo de inicio tendrán el mismo problema que los usuarios normales y es que dependerán del registro de locales para hacer uso de la aplicación.

Finalmente, los dueños de locales son la pieza más fundamental en el inicio de la aplicación. Este tipo de usuario puede descargarla de igual forma que los actores antreriores y registrarse. Una vez registrados, podrán incluir sus locales de música en la aplicación de forma que el resto de actores puedan realizar sus funcionalidades previamente definidas.

7. Conclusiones

Gracias a este TFG he aprendido como desarrollar una aplicación híbrida que sea válida tanto para iOS como Android. Inicialmente el TFG iba a ser orientado únicamente a la planificación del proyecto, pero la curiosidad ha hecho que decididera aprender por mi cuenta estas nuevas tecnologías y a desarrollar una primera versión (MVP) de la aplicación.

A lo largo del desarrollo me he encontrado con varios problemas. Dado que no conocía completamente la tecnología, la mayor parte del tiempo ha sido dedicada a la lectura de la documentación disponible en la web, así como foros de programación (StackOverFlow).

Me he dado cuenta que existen miles de este tipo de APIs y que existen para prácticamente cualquier cosa. Lo que más me ha fascinado ha sido el hecho de que las grandes empresas como Twitter, Spotify o Youtube disponen de información pública de su API para que las integres en tus aplicaciones. Esto me ha hecho reflexionar sobre lo importante que es la comunidad y el open code para el desarrollo de nuevas tecnologías.

7.1. Relación del trabajo desarrollado con los estudios cursados

El haber estudiado la rama de ingenieria de software me ha ayudado mucho en el desarrollo de este proyecto. Esta rama enseña las diferentes fases por las que pasa el desarrollo de un producto software por lo que estoy familiarizado con las diferentes partes de este proyecto.

En primer lugar, gracias a la asignatura de AER, he aprendido a realizar un análisis y especificación de requisitos en función a los estándares actuales, en este caso al estándar IEEE 29148:2018.

El framework de Angular ha sido algo nuevo para mi durante el último curso de la carrera. Aunque este no formara parte de la planificación docente, gracias a la asignatura de PIN he tenido que desarrollar una aplicación web junto a mis compañeros y hemos elegido Angular como tecnología tras comprobar que es una de las más utilizadas actualmente.

El haber aprendido los conocimientos básicos de Javascript en la asignatura de TSR me ha ayudado bastante a la hora de aprender Typescript, ya que este está basado en Javascript aunque es fuertemente tipado.

Las asignaturas de ISW y DDS han sido de gran utilidad en cuanto a la parte de arquitectura y estructura del proyecto. Del mismo modo, la asignatura de IPC me ha ayudado a la hora de diseñar las diferentes interfaces de la aplicación.

Como conclusión final, pienso que la carrera de ingeniería informática te enseña los diferentes caminos que se pueden escoger a la hora de desarrollar tu vida profesional en este sector. Se imparten asignaturas de todos los ámbitos y te permite especializarte en una de las áreas de la informática que más te llame la atención y hacia la que quieras orientar tu vida profesional.

8. Trabajos futuros

Mi objetivo es tener una aplicación completamente funcional y escalable para el invierno de 2022. Me gustaría publicarla en las tiendas de aplicaciones de ambas plataformas (App Store y Google Play) y, aunque no tenga apenas descargas, experimentar el proceso íntegro de desarrollo de una app hasta que esta se publica en los markets.

Así mismo me gustaría participar en concursos de emprendimiento organizados por la UPV presentando ya sea esta aplicación u otra que se me ocurra en el futuro. Lo que tengo claro es que quiero orientar mi vida profesional al desarrollo de software.

9. Referencias

[1] Proyecto de desarrollo de una aplicación móvil para enviar recomendacionesal DJ:

https://riunet.upv.es/handle/10251/69027

[2] Google Play Store:

https://play.google.com/store?hl=es_419&gl=ES

[3] App Store:

https://www.apple.com/es/app-store/

[4] IEEE29148:2018:

https://webstore.iec.ch/preview/info_isoiecieee29148%7Bed2.0%7Den.pdf

[5] OAuth:

https://www.nts-solutions.com/blog/oauth-que-es.html

[6] Amazon Cognito:

https://docs.aws.amazon.com/es_es/cognito/latest/developerguide/what-is-amazon-cognito.html

[7] MongoDB:

https://www.mongodb.com/es/what-is-mongodb

[8] Google Firebase:

https://firebase.google.com/docs

[9] NoSQL:

https://www.grapheverywhere.com/bases-de-datos-nosql-marcas-tipos-ventajas/

[10] Shazam API:

https://rapidapi.com/apidojo/api/shazam

[11] API REST:

https://www.redhat.com/es/topics/api/what-is-a-rest-api

[12] Spotify API:

https://developer.spotify.com/documentation/web-api/

[13] Angular:

https://angular.io/guide/what-is-angular

[14] Ionic:

https://ionicframework.com/docs

[15] TypeScript:

https://www.typescriptlang.org/

[16] JavaScript:

https://developer.mozilla.org/es/docs/Web/JavaScript

[17] Visual Studio Code:

Https://code.visualstudio.com/

[18] GitHub:

https://github.com/

[19] SPA:

https://www.kikopalomares.com/blog/que-es-una-web-spa-single-page-application

[20] Capacitor:

https://capacitorjs.com/docs