Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Desarrollo de aplicaciones android: TodoMusica

Android apps development: TodoMusica

Sergio Manuel Rodríguez Vega

La Laguna, 2 de julio de 2020

D. **Alejandro Pérez Nava**, con N.I.F. 43821179-S profesor Asociado de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Fernando Pérez Nava**, con N.I.F. 42091420-V profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

”*Desarrollo de aplicaciones android: TodoMusica*”

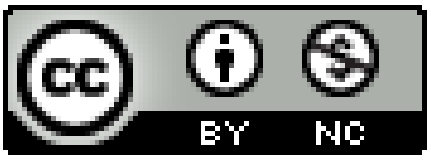
ha sido realizada bajo su dirección por D. **Sergio Manuel Rodríguez Vega**, con N.I.F. 79061112-S.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de julio de 2020

Agradecimientos

A D.Alejandro Pérez Nava y D.Fernando Pérez Nava por la coordinación y orientación del proyecto, así como a familia y amigos por la colaboración en el testeo o recomendaciones en posibles mejoras

Licencia



Resumen

El objetivo principal del presente proyecto es la obtención de una aplicación nativa para dispositivos android, en la cual se encuentre la información relacionada con las noticias de la actualidad del sector musical. Esta aplicación cuenta con los mecanismos necesarios para la obtención de la información por parte del usuario de la manera más sencilla posible. Entre dichos mecanismos se encuentra la herramienta de búsqueda por la cual los usuarios podrán encontrar a los artistas por su nombre y la posterior posibilidad de seguir dicho contenido, lo cual facilita la interacción del usuario.

Para el desarrollo ha sido necesario realizar un estudio de las aplicaciones presentes en el mercado las cuales cumplan un objetivo similar ya sean orientados al mismo u otro sector, como pueden ser las aplicaciones orientadas al sector de noticias del mundo del fútbol.

La obtención de los datos necesarios para completar el contenido de la aplicación se ha realizado mediante el uso de APIs (Interfaz de programación de aplicaciones), más en concreto de las de servicio web, las cuales permiten el intercambio de información entre un servicio web y una aplicación mediante una URL. La selección de las interfaces utilizadas se ha realizado tras previo estudio de las ofertas presentes en el mercado actual, seleccionando las que mejor se ajustan a las necesidades del proyecto.

Palabras clave: Nativo, API, Android

Abstract

The main objective of this project is to obtain a native application for android devices, in which you will find information related to current news of the music sector. This application has the necessary mechanisms to obtain the information by the user in the simplest possible way. Among these mechanisms is the search tool by which users can find artists by name and the subsequent possibility of following that content, which facilitates user interaction.

For the development it has been necessary to carry out a study of the applications present in the market which meet a similar objective, whether they are oriented to the same or another sector, such as applications aimed at the news sector of the world of football.

Obtaining the necessary data to complete the content of the application has been done through the use of APIs (Application Programming Interface), more specifically those of the web service, which allow the exchange of information between a web service and an application through an URL. The selection of the interfaces used has been made after prior study of the offers present in the current market, selecting the ones that best fit the needs of the project.

Keywords: Native, API, Android

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Antecedentes

A lo largo de la historia se han realizado gran diversidad de aplicaciones relacionadas con el mundo de la información y las noticias. No obstante dentro de este amplio catálogo no se encuentra una oferta referente al mundo de la música, o al menos no se encuentra una oferta líder y de gran relevancia.

1.1.1. Historia de las aplicaciones móviles

La primera aplicación móvil data de 1998, año en el que la compañía de telecomunicaciones sueca, Nokia, opta por añadir una pequeña aplicación en sus dispositivos móviles con la finalidad de amenizar la espera de sus usuarios en las situaciones más cotidianas, como pueden ser las esperas, tanto en las colas del supermercado, como en los trayectos en transporte público u otras situaciones. El nombre de dicha aplicación es "La serpiente", o, "Snake", el cual es una réplica de un videojuego que había causado furor en los años 70.

A día de hoy, el mundo relacionado con las tecnología móviles y el desarrollo de aplicaciones ha evolucionado, dando lugar a propuestas más variadas y complejas. Como consecuencia han pasado a formar parte de nuestra vida diaria y establecemos conexión con ellas de manera casi permanente.

Figura 1.1: Logotipo de la compañía Nokia

1.1.2. Historia de la industria musical

La industria musical es una asociación de compañías y particulares, cuyo objetivo es generar dinero mediante la creación de canciones o fragmentos de canciones, conciertos, composición de letras o partituras.

Este modelo surge entre 1930 y 1950, cuando la venta de grabaciones tomó mayor relevancia que la venta de partituras. En la primera década del año 2000, la industria musical sufrió cambios drásticos con la llegada de la distribución digital la generalizada de la música vía internet.

Un indicador llamativo de estos cambios es el total de ventas: desde el 2000, las ventas de música grabada han descendido sustancialmente mientras que las de música en vivo han incrementado con mucha importancia.

1.2. Alcance

El alcance de este proyecto, tiene como finalidad el desarrollo de una aplicación, la cual sea capaz de recopilar todos, o gran can cantidad de información acerca de la industria musical, formatee eso datos y ofrezca como resultado una opción cómoda para el usuario. Para la obtención de estos datos requeridos se utilizarán los contenidos alojados en la web, aprovechando así el mundo de las nuevas tecnologías.

El desarrollo de este proyecto será implementado en la plataforma Android Studio, haciendo uso de los lenguajes de programación, Java, PHP y xml. Por otra parte los servicios online, como lo son la base de datos, la ejecución de scripts de actualización y control o el almacenamiento de dichos scripts se realiza mediante la aplicación web de hosting 000webhost.

Esta plataforma presenta una opción económica y bastante competente para el desarrollo de todo tipo de aplicaciones web, ofreciendo gran cantidad de herramientas para la creación de sitios web, paneles de control personalizados y un alto rendimiento de servidores.

Debido al carácter actual del proyecto se opta por utilizar el paquete básico para el alojamiento de los servicios y los datos, el cual es absolutamente gratuito y la compañía promete que se mantendrá gratis por siempre. Ni siquiera tendrás que darles información de pago, pero si estás listo para mejorar a un plan de pago, puedes usar una variedad de opciones, desde Visa y MasterCard hasta Bitcoin y PayPal.

Si tenemos en cuenta que la oferta básica es gratuita, quizá no sea sorprendente conocer que muchos de los complementos también tienen precios razonables. El problema comienza cuando empiezas a añadir nuevos complementos y mejoras a tu paquete de hosting, dando lugar a que el coste final llegue a ser sustancialmente superior al de un paquete de pago que ofrezca una gama de características mejores.

	Location	Website	Connection	First Byte	Total
	Germany, Frankfurt	pesqa.info *(185.224.138.206)*	0.012 secs	0.025 secs	0.025 secs
	France, Paris	pesqa.info *(185.224.138.206)*	0.017 secs	0.034 secs	0.034 secs
	NL, Amsterdam	pesqa.info *(185.224.138.206)*	0.026 secs	0.032 secs	0.032 secs
	UK, London	pesqa.info *(185.224.138.206)*	0.026 secs	0.039 secs	0.039 secs
	Canada, Montreal	pesqa.info *(185.224.138.206)*	0.090 secs	0.181 secs	0.181 secs
	Canada, Toronto	pesqa.info *(185.224.138.206)*	0.125 secs	0.220 secs	0.221 secs
	USA, Dallas	pesqa.info *(185.224.138.206)*	0.127 secs	0.543 secs	0.549 secs
	USA, Los Angeles	pesqa.info *(185.224.138.206)*	0.159 secs	0.320 secs	0.320 secs
	USA, New York	pesqa.info *(185.224.138.206)*	0.206 secs	0.287 secs	0.287 secs
	USA, San Francisco	pesqa.info *(185.224.138.206)*	0.211 secs	0.364 secs	0.364 secs
	Brazil, Sao Paulo	pesqa.info *(185.224.138.206)*	0.221764 secs	0.440908 secs	0.440987 secs
	Singapore	pesqa.info *(185.224.138.206)*	0.238 secs	0.474 secs	0.474 secs
	India, Bangalore	pesqa.info *(185.224.138.206)*	0.660 secs	0.811 secs	0.811 secs
	USA, Atlanta	pesqa.info *(185.224.138.206)*	0.877 secs	0.990 secs	0.990 secs
	Australia, Sydney	pesqa.info *(185.224.138.206)*	1.794 secs	2.081 secs	2.081 secs
	JP, Tokyo	pesqa.info *(185.224.138.206)*	3.302 secs	3.549 secs	3.549 secs
	Average response time		0.341 secs	0.487 secs	0.487 secs
	Global Performance Grade *(Based on Total Time)*		PERFORMANCE GRADE: A+		

Figura 1.2: Rendimiento de los servidores del proveedor

1.3. Objetivos

El objetivo principal del proyecto es unificar los contenidos web relacionados con las noticias del sector musical, esto mediante una aplicación nativa android. Con la finalidad de cumplir este objetivo se presentan los siguientes puntos:

- Estudiar y conocer los tipos de información presentes en la web
- Estudiar y conocer los distintos modelos de aplicaciones android
- Conocer las técnicas del desarrollo android
- Obtención de datos
- Seguridad de los contenidos
- Desarrollo de un sistema de sesiones
- Desarrollo de sistema de noticias
- Desarrollo de sistema de recomendaciones
- Desarrollo de sistema de notificaciones

Capítulo 2

Obtención de datos

Uno de los aspectos principales y por consiguiente más importantes del proyecto es la información, puesto que se trata de una aplicación relacionada con el mundo de las noticias. Toda la información proporcionada es obtenida de la web por medio de diversos métodos.

2.1. API

2.1.1. ¿Que es una API?

Una interfaz de programación de aplicaciones, o API por sus siglas en inglés, es un conjunto de definiciones y protocolos utilizados para el desarrollo e integración de software, permitiendo la comunicación entre componentes a través de un conjunto de reglas.

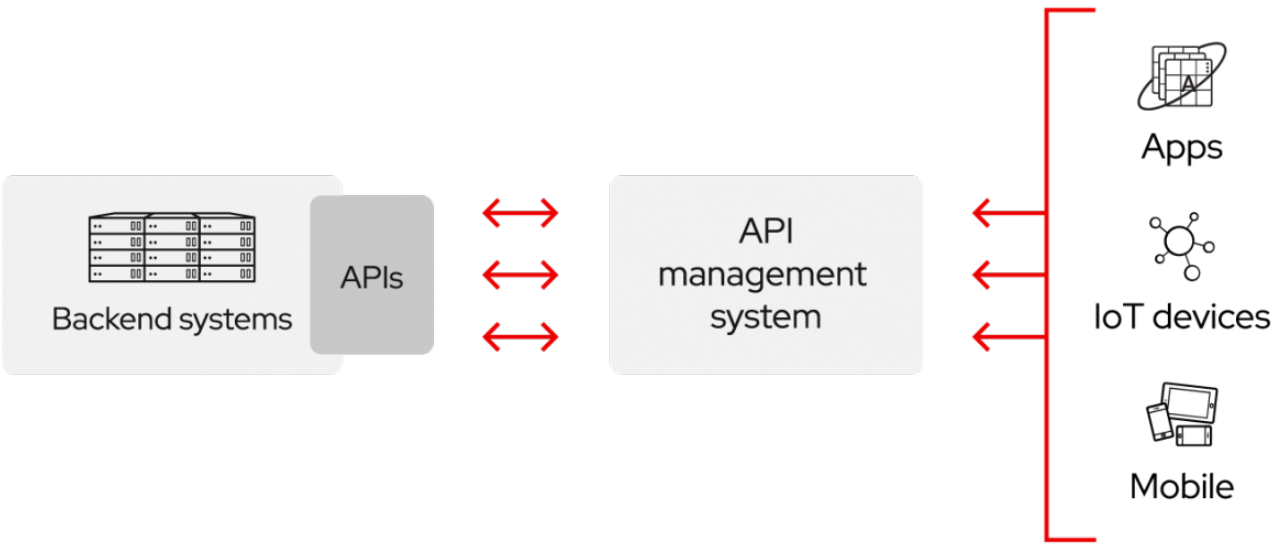


Figura 2.1: Funcionamiento de las API

Las API son medios simplificados de conexión entre la infraestructura por medio del desarrollo de aplicaciones nativas en la nube, pero también permitiendo compartir datos con los propios clientes u otros grupos de usuarios. Las API públicas representan un gran valor comercial debido que simplifican la forma de conectar y rentabilizar sus datos.

En resumen, permiten habilitar el acceso a recursos y, al mismo tiempo, mantener la seguridad y el control de los mismos. Para conectarse a las API y crear aplicaciones que utilicen los datos o las funciones que estas ofrecen, se puede utilizar una plataforma de integración distribuida que conecte todos los elementos, incluidos los sistemas heredados.

2.2. Información general de artistas

A la hora de obtener la información básica referente a los artistas, como lo son, nombre, género, bibliografía... Se ha optado por el uso de las API de intercambio donde se puede consultar información. Para la selección de dichas API se ha realizado un estudio previo de las ofertas presentes en el mercado dando como finalistas las siguientes:

- Deezer: Es una plataforma de transmisión de música en línea que tiene una extensa colección de diferentes tipos de música. El servicio permite a los usuarios escuchar su música favorita en múltiples dispositivos, incluso sin conexión.
- Spotify: Es una plataforma para acceder a millones de canciones y reproducirlas en la web, computadora de escritorio o dispositivos móviles. Permite a los usuarios poder crear y compartir sus propias listas con canciones y recientemente compartir los propios videoclips.
- LAST.FM: Posee un sitio web en la cual encontrar una gran colección de música. Permite a los usuarios crear sus propios perfiles, rastrear la música que escuchan o acceder a otros servicios interesantes, los cuales van desde la creación de listas personalizadas hasta la interacción con otros usuarios.
- iTunes: Es una plataforma propiedad de Apple, Inc. que se utiliza principalmente para reproducir y administrar archivos de música de audio y video. El software se puede usar en los sistemas operativos Windows y Macintosh.

Con motivo de elegir qué API o APIs utilizar se ha realizado un estudio en el cual se comparan las características principales y fundamentales de cada una de las listadas anteriormente. Dicho estudio se basa en los siguientes campos:

- Características
- Información
- Precio
- Facilidad de uso

2.2.1. Estudio de las API

Deezer API

La API de Deezer ofrece acceso ilimitado, sin necesidad de identificación. Proporciona una gran cantidad de servicios para desarrollar aplicaciones web y permite descubrir el catálogo de música ofrecido por la compañía.

Características: La API proporcionada por deezer permite acceder a gran diversidad de contenidos ofrecidos por la empresa, como lo son, la información general de artistas, álbumes o canciones. Por otro lado ofrece métodos para la modificación y la eliminación de esos propios datos.

En nuestro caso solo aplicaremos el uso de las características de acceso a los datos, sin necesidad de modificarlos.

Información: Al ser necesaria únicamente la información personal del artista descartamos las otras posibilidades ofrecidas por la API. En este caso la información de la cual disponemos es la siguiente:

- **id:** Tipo de dato entero que proporciona el identificador del artista dentro de la API.
- **name:** Tipo de dato String donde se recoge el nombre del artista o grupos.
- **link:** Tipo de dato url donde se almacena al perfil del artista dentro de la aplicación de Deezer.
- **share:** Tipo de dato url donde se comparte el perfil del grupo o artistas.
- **picture[small, big, medium...]:** Tipo de dato url donde se aloja la imagen de perfil del grupo o artista.
- **album:** Tipo de dato entero donde se almacena el número de álbumes del grupo o artistas.
- **fan:** Tipo de dato entero donde se almacena el número de seguidores dentro de la aplicación de Deezer.
- **radio:** Tipo de dato booleano que retorna si el artista posee un smartradio.
- **tracklist:** Tipo de dato url donde se almacena una lista de los mejores éxitos del grupo o artista.

Precio: Gratuito

Facilidad de uso: La información se obtiene por medio de una solicitud típica HTTP GET. La url base indicada es la siguiente:

"https://api.deezer.com/version/service/id/method/?parameters"

Figura 2.2: URL de la petición de acceso Deezer

Los formatos de respuesta que ofrece son los siguientes:

Tipo	Cabecera	Extensión
JSON	application/json	.json
JSONP	application/json	.json
XML	application/xml, text/xml	.xml
PHP	application/xml, application/json	.php

Tabla 2.1: Formatos de respuesta Deezer

LAST.FM API

La API de Lastf.fm proporciona acceso a la información mediante consultas con respuesta REST (representational state transfer), con la que posees acceso ilimitado a los contenidos de la compañía. Al igual que la anterior, ofrece una gran cantidad de servicios para el desarrollo web.

Características: La API proporcionada por LAST.FM provee métodos para el acceso y la modificación de gran diversidad de datos relacionados con el mundo de la música, además de otros métodos para la creación y modificación de usuarios e incluso para acceder a la geolocalización.

En nuestro caso solo aplicaremos los métodos necesarios para acceder al perfil de un grupo o artista, puesto que el resto de información no es relevante para nuestra aplicación.

Información: En cuanto a la respuesta que obtenemos al consultar los datos del grupo o artista obtenemos los siguientes resultados:

- **name:** Tipo de dato String que contiene el nombre del grupo o artista.
- **mbid:** Tipo de dato String que contiene el identificador del usuario.
- **url:** Tipo de dato url que contiene el enlace al perfil del grupo o artista en la aplicación.
- **image:** Tipo de dato url que contiene el enlace a la imagen de perfil del grupo o artista.
- **star:** Tipo array que contiene la valoración del grupo o artista por medio de estrellas.
- **similar:** Tipo array que contiene la información de artistas similares al buscado.
- **tags:** Tipo array que contiene el género o géneros musicales asociados al grupo o artista.
- **bio:** Tipo de dato String que contiene la descripción bibliográfica del grupo o artista.

Precio: Gratuito

Facilidad de uso: La información se obtiene por medio de una solicitud típica HTTP GET. La url base indicada es la siguiente:

```
"http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=Cher&api_key=YOUR_API_KEY&format=json"
```

Figura 2.3: URL de la petición de acceso Last.fm

Los formatos de respuesta que ofrece son los siguientes:

Tipo	Cabecera	Extensión
JSON	application/json	.json
XML	application/xml, text/xml	.xml
REST	-	-

Tabla 2.2: Formatos de respuesta LAST.FM

Spotify API

La API de Spotify proporciona acceso a la información mediante consultas con respuesta REST (representational state transfer), con la que ofrece acceso a gran cantidad de información de la compañía. Al igual que las anteriores, ofrece una gran cantidad de servicios para el desarrollo web.

Características: Las API de Spotify permite leer las características de audio de las canciones y analizar sus diversos atributos. También puede integrar la música de Spotify en su aplicación, ofrecer recomendaciones a los usuarios y buscar y administrar artistas, álbumes, pistas o listas de reproducción.

Información: Al igual que se ha mencionado anteriormente únicamente nos interesa la información relacionada con el perfil del grupo o artista.

- **external urls:** Tipo de dato array que contiene los enlaces a todas las posibles plataformas donde podremos encontrar información del artista.
- **followers:** Tipo de dato array que contiene el número de seguidores del grupo o artista y un enlace de referencia.
- **genres:** Tipo de dato array que contiene el género o géneros asociados al grupo o artista.
- **href:** Tipo de dato url que contiene el enlace de la petición HTTP.
- **id:** Tipo de dato String que contiene el identificador del usuario.
- **images:** Tipo de dato array que contiene las imágenes de perfil en diversos tamaños.
- **name:** Tipo de dato String que contiene el nombre del grupo o artista.
- **popularity:** Tipo de dato entero que contiene la valoración del artista dentro de Spotify.
- **type:** Tipo de dato String que determina si se trata de un grupo o artista.

Precio: Gratuito

Facilidad de uso: La información se obtiene mediante una consulta de tipo HTTP GET, con la necesidad de obtener un token mediante la siguiente url:

https://api.spotify.com/v1/artists/id

Los formatos de respuesta que ofrece son los siguientes:

Tipo	Cabecera	Extensión
JSON	application/json	.json
REST	-	-

Tabla 2.3: Formatos de respuesta Spotify

iTunes API

La API permite colocar campos de búsqueda de contenido en iTunes Store y Apple Books Store. También puede llamar a una solicitud de búsqueda basada en ID para crear asignaciones entre su biblioteca de contenido y el catálogo digital.

Características: Las API de iTunes permite leer las características de las canciones y analizar sus diversos atributos. También puede integrar la música de iTunes en su aplicación, ofrecer recomendaciones a los usuarios y buscar y administrar álbumes, pistas o listas de reproducción.

Información: Esta API proporciona información referente a canciones o álbumes musicales únicamente.

Precio: Gratuito

Facilidad de uso: La información se obtiene mediante una consulta de tipo HTTP GET, con la necesidad de obtener un token mediante la siguiente url:

https://itunes.apple.com/search?parameterkeyvalue

Los formatos de respuesta que ofrece son los siguientes:

Tipo	Cabecera	Extensión
JSON	application/json	.json
REST	-	-

Tabla 2.4: Formatos de respuesta iTunes

2.2.2. Elección

Una vez analizadas las características de cada una de las principales APIs encontradas y recomendadas por otros desarrolladores, podemos concluir que las que mejor se integran a las necesidades previamente expuestas para el desarrollo de nuestra aplicación, son tanto la de Deezer, como la de LAST.fm, las cuales se han unido para aportar un mayor grado de completitud y una mayor riqueza a la aplicación.

2.3. Contenido de noticias

Para obtener la información referente a las noticias se ha producido un cambio, donde en un principio se realizaba el uso de la API ofrecida por microsoft, "Bing news seacrh", que en un primer análisis parecía ser una opción factible resultó que carecía del contenido necesario puesto que las noticias que ofrecía en ciertas ocasiones no coinciden con las solicitadas, por otro lado ha sido descartada debido a un cambio de políticas de microsoft solicitando una suscripción monetaria a cambio del servicio.

En la actualidad las noticias se obtienen mediante el servicio de google alerts, mediante el cual solicitamos que se nos envíe un mensaje al correo electrónico con los contenidos referentes a los artistas. Una vez recibido dicho mensaje es necesario formatear dicho contenido y almacenarlo en nuestra base de datos. Para este fin se ha utilizado la API oficial de Gmail ofrecida por google.

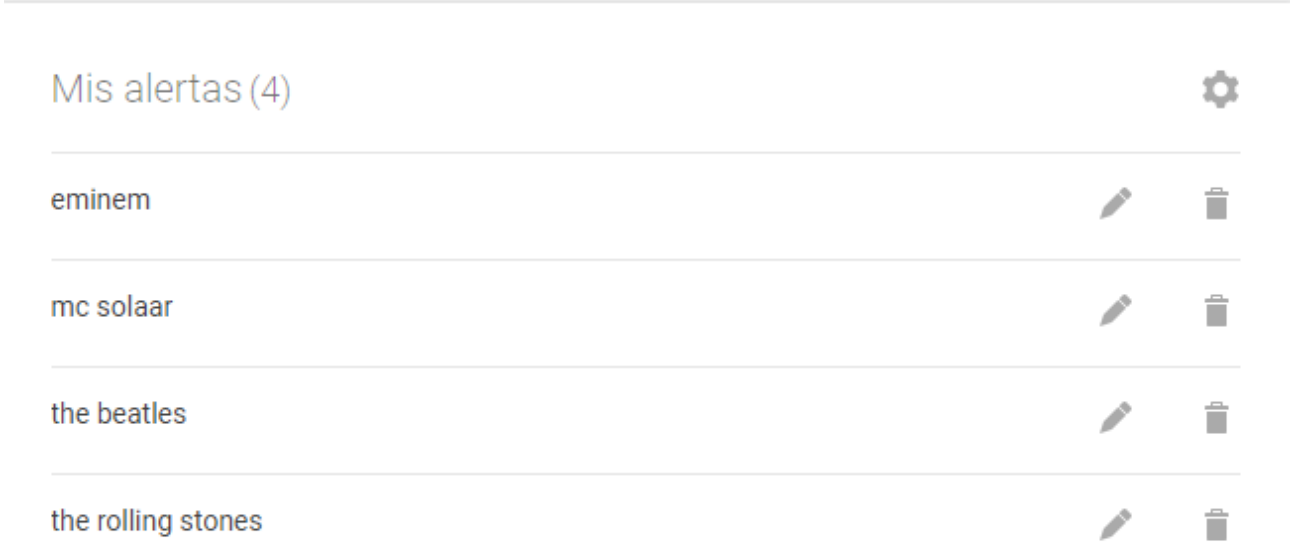


Figura 2.4: Contenido de google alert

2.3.1. Gmail API

Esta API permite acceder al contenido del correo electrónico dando la posibilidad de leer, redactar o incluso modificar mensajes, así como permite administrar los elementos básicos, como lo son los borradores, hilos o etiquetas.

Todo lo necesario para utilizar esta herramienta es la descarga de las librerías necesarias del cliente. Los usos más extendidos de esta API son:

- Extracción, indexación y copia de seguridad de correo
- Envío automatizado o programático de mensajes
- Migrar cuentas de correo electrónico de otros proveedores
- Establecer firmas de correo electrónico estandarizadas para usuarios en un dominio

2.4. Diagrama de interacción de la aplicación

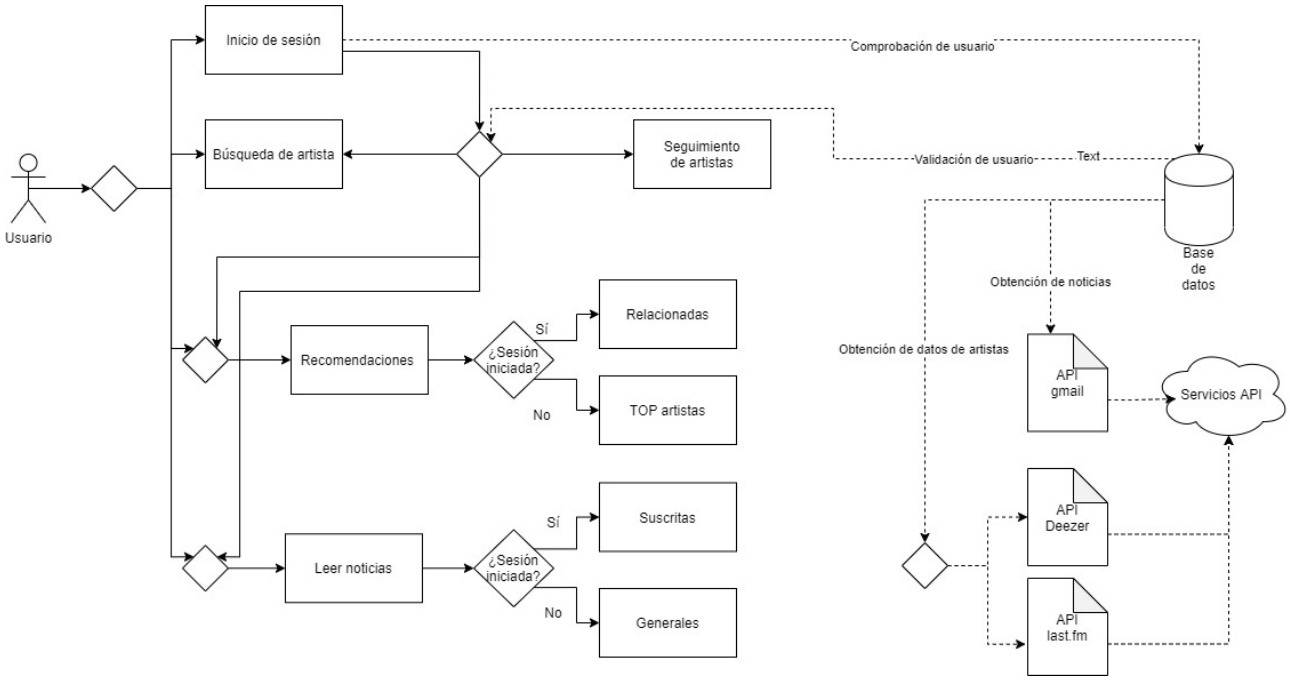


Figura 2.5: Flujo de interactividad de la aplicación

Capítulo 3

Desarrollo de la aplicación

En este apartado se expone el desarrollo completo, se tratarán temas, como el desarrollo de funcionalidades de la aplicación, la metodología aplicada o el desarrollo de la interfaz y el por que de estas elecciones.

3.1. Metodología

Para el desarrollo del presente proyecto y tras un análisis de las posibilidades se ha optado por la aplicación de una metodología ágil, en este caso se ha utilizado "Scrum", debido a las ventajas presentes frente al resto en cuanto a las necesidades presentes.

Entre las diferentes metodologías de un proyecto, Scrum es un método para trabajar en equipo a partir de iteraciones o Sprints. Así pues, Scrum es una metodología ágil, por lo que su objetivo será controlar y planificar proyectos con un gran volumen de cambios de última hora, en donde la incertidumbre sea elevada.

Se suele planificar por semanas. Al final de cada Sprint o iteración, se va revisando el trabajo validado de la anterior semana. En función de esto, se priorizan y planifican las actividades en las que invertimos nuestros recursos en el siguiente Sprint.

La metodología Scrum se centra en ajustar sus resultados y responder a las exigencias reales y exactas del cliente. De ahí, que se vaya revisando cada entregable, ya que los requerimientos van variando a corto plazo. El tiempo mínimo para un Sprint es de una semana y el máximo es de cuatro semanas.

3.2. Herramientas

3.2.1. Android Studio

Android Studio es un entorno de desarrollo, que provee las herramientas y servicios necesarios para la creación de aplicaciones Android por parte de los desarrolladores. Este tipo de entornos de desarrollo se encuentran presentes en gran variedad de los sistemas operativos de la actualidad.

En este entorno de desarrollo nos encontraremos con gran diversidad de herramientas para el desarrollo, esto incluye aquellas que implican desde el código, al diseño de la interfaz de usuario de la aplicación. Va a poder guiar a dicho creador en todo el proceso, para poder obtener de esta manera una aplicación, que se podrá publicar posteriormente en la Play Store, por ejemplo.

Los que quieran usar Android Studio lo podrán descargar en su ordenador, compatible con Windows, MacOS, ChromeOS y Linux, para poder trabajar en el mismo en la creación de dicha aplicación. Ningún desarrollador se encontrará por tanto con problemas para tener acceso a este entorno y poder trabajar con el mismo.

3.2.2. Visual studio code

Visual Studio Code es un editor de código desarrollado por Microsoft para Windows , Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. También es personalizable, por lo que los usuarios pueden cambiar el tema del editor, los atajos de teclado y las preferencias, además de poder descargar nuevos complementos y funcionalidades. Es gratuito y de código abierto.

Es compatible con varios lenguajes de programación y un conjunto de características que pueden o no estar disponibles para un idioma dado. Muchas de las características de Visual Studio Code no están expuestas a través de los menús o la interfaz de usuario. Más bien, se accede a través de la paleta de comandos o a través de archivos json.

En el rol de editor de código fuente, Visual Studio Code permite cambiar la página de códigos en la que se guarda el documento activo, el carácter que identifica el salto de línea y el lenguaje de programación del documento activo.

3.3. Modelo y diseño

Una vez realizado un estudio de los distintos modelos de programación Android se opta por realizar un desarrollo basado en MVC. En el patrón de arquitectura MVC (modelo – vista – controlador), basado en el principio de separación de intereses:

- El modelo contiene la información con la que el sistema trabaja, proporcionándole a la vista para que pueda mostrarla y permitiendo realizar cambios en ella desde el controlador.
- El controlador responde a acciones del usuario, modificando el modelo cuando sea necesario. Además, se comunica con la vista para que se actualice con los últimos cambios del modelo.
- La vista presenta al usuario la información del modelo.

Uno de los errores más comunes a la hora de utilizar MVC en Android ha sido y aún sigue siendo utilizar una Activity como controlador, haciéndola responsable de tareas de las que no debería ocuparse, como llamar directamente al modelo para realizar una modificación de algún dato, cuando esto debería ser tarea del controlador. Esto es claramente una violación del principio de responsabilidad única y al usar una Activity como controlador le estamos haciendo dependiente de la plataforma, por lo que su código podría verse afectado cuando la Activity sea destruida por el sistema.

Si queremos aplicar correctamente MVC en Android, las activities y fragments solo deberían ocuparse de mostrar los datos y notificar los eventos del usuario al controlador, siendo los únicos componentes ligados a la plataforma. Los controladores y modelos deberían ser clases separadas sin ninguna dependencia con Android, haciéndolos así más fáciles de testear.

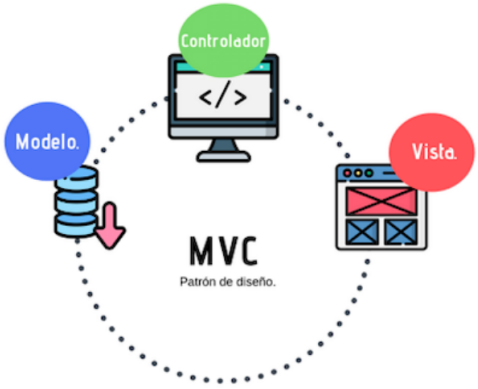


Figura 3.1: Modelo MVC

3.4. Funcionalidades

3.4.1. Autentificación

Debido al concepto del proyecto es necesario establecer un registro y control de los usuarios que van a utilizar la aplicación. Para cumplir este objetivo se ha implementado un sistema de autentificación, donde el usuario se registra mediante un nombre, un usuario, correo y contraseña. Este sistema almacena los datos en la base de datos dentro de una tabla específica.

Contenido: Este apartado consta de dos vistas, una para el inicio de sesión y otra para registrar a los usuarios. Cada vista posee un formulario donde se introducirán los datos, además de un texto de enlace para navegar entre ambas vistas.

Funciones: Añade el contenido referente a los usuarios de la aplicación, así como permite determinar qué usuarios son ya conocidos para la aplicación retornando sus datos y permitiendo establecer las relaciones pertinentes con los otros modelos.

Diseño: En cuanto al diseño se presenta un formulario con los campos necesarios para el registro y el acceso del usuario, con un botón que comienza la consulta con la base de datos.

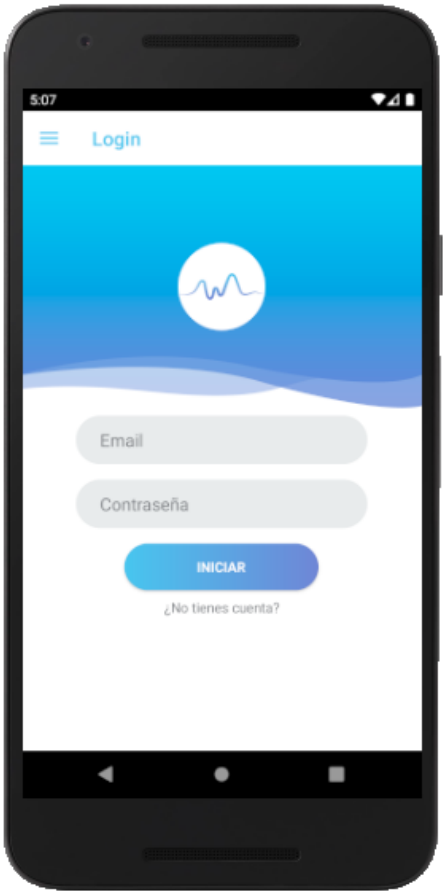


Figura 3.2: Modelo de login

3.4.2. Motor de búsqueda

Una de las preocupaciones principales del proyecto es la búsqueda de información y ¿Qué mejor forma de indexar esta información, que por el nombre de artista?. Es por esto que se ha decidido implementar un motor de búsqueda sencillo donde el usuario pueda consultar sin mayor dificultad la gama de artistas que posee la aplicación.

Contenido: El motor de búsqueda consta de una única vista, en la cual se encuentra una barra donde el usuario puede introducir el nombre de un artista de su interés, en caso de escribir el nombre completo y que este mismo se encuentre registrado en la app se mostrará dicho usuario. Por el contrario si no se escribe el nombre completo se mostrarán todos los artistas cuyos nombres posean coincidencia con el texto escrito.

Funciones: Se realiza una consulta a la base de datos con el contenido especificado en la barra de búsqueda, donde se retornan los datos de todos aquellos artistas los cuales coinciden con el parámetro introducido por medio del uso de LIKE en la consulta SQL.

Diseño: En cuanto al diseño se encuentra un cuadro de texto donde se introduce el nombre del grupo o artista y seguidamente aparece debajo una lista con los grupos o artistas que coincidan con los términos de la búsqueda.

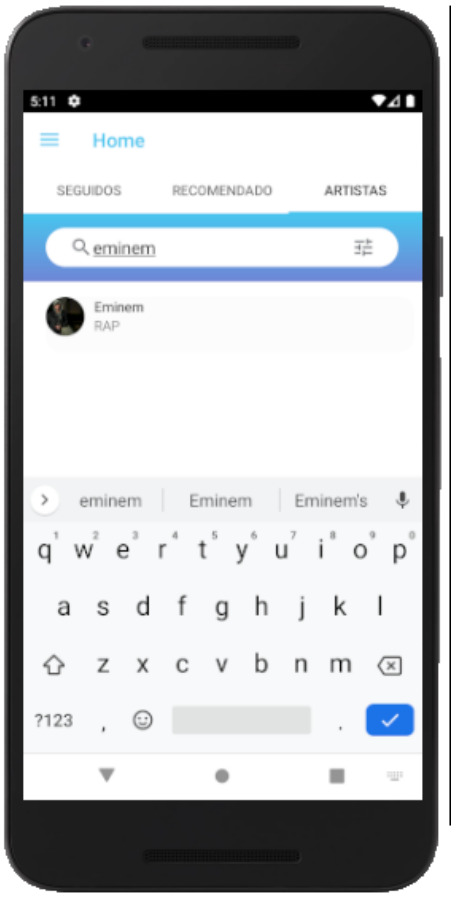


Figura 3.3: Modelo de búsqueda

3.4.3. Seguimiento de artista

Una vez realizada la búsqueda de nuestro artista de interés, lo ideal es mantener un control de sus noticias, ¿Como se consigue esto?, cómo hemos podido observar en la actualidad el mejor método para estar al día con la información lo más recurrente es un sistema de seguimiento. Una vez realizada esta conclusión se ha optado por añadirlo al proyecto.

Contenido: Para seguir a un artista se ha decidido además generar una vista de perfil de ese artistas, en la cual podemos encontrar, tanto su nombre, género o incluso el número de seguidores que posee en nuestra APP, así como el número de noticias que han sido añadidas referentes a el. Además de estos datos posee una feed con las últimas noticias referentes a dicho artista.

Funciones: Posee un botón que añade una entrada a la tabla de seguidores de nuestra base de datos, donde se establece una relación entre ambos mediante el uso de los id, uno de usuario y el otro de artista almacenados en las distintas tablas, permitiendo consultas rápidas y tablas sin exceso de columnas.

Diseño: En cuanto al diseño se muestra la imagen del artista o grupo acompañada del número de seguidores y el número de noticias asociadas, junto a esto se encuentra un botón que permite establecer la relación de seguimiento. Además en la parte inferior se muestra una lista con las últimas noticias asociadas al artista.

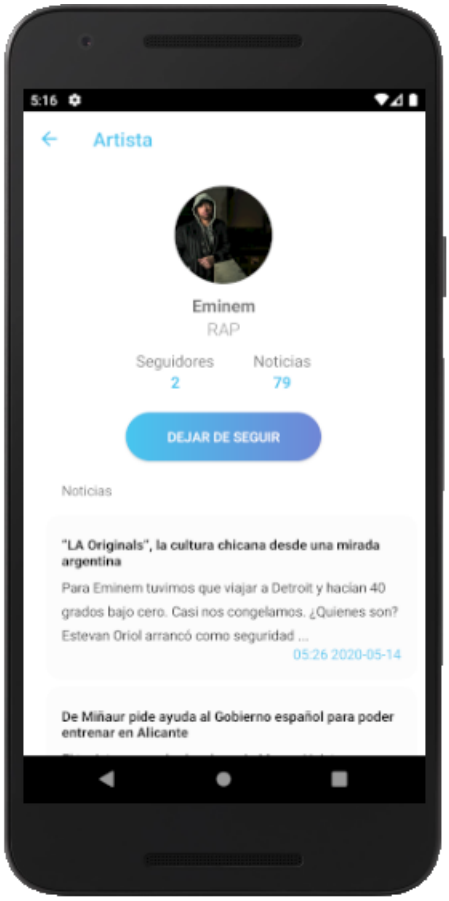


Figura 3.4: Modelo de seguimiento

3.4.4. Feed de noticias

En la pestaña de inicio se ha decidido implementar una feed con las diversas noticias alojadas en los servidores de la aplicación, para así dar una visión más accesible de las mismas.

Contenido: Dicha feed consta de una vista donde se muestran las noticias en una lista. Dichas noticias constan de un título, una breve descripción de la noticia, la fecha de publicación en nuestra app y un enlace externo a la noticia original. Estas noticias varían en función de si es un usuario registrado, donde se mostrarán las noticias de los artistas a los cuales sigue, o por otro lado si accede de forma anónima recibirá las últimas noticias añadidas a la aplicación independientemente del artista.

Funciones: La función es recoger los datos almacenados en la base de datos, con la condición de si es un usuario registrado o no. En el caso afirmativos se utilizará la tabla de relaciones definida en el punto anterior, donde se define si un usuario X sigue a un artista Y.

Diseño: En cuanto al diseño se presenta una lista con todas las noticias relacionadas de los artistas a los cuales se está suscrito. Los objetos de la lista muestran el titular, un breve resumen y la fecha de publicación.

Figura 3.5: Modelo de feed

3.4.5. Recomendaciones

Para mejorar la experiencia y personalización de la experiencia de los usuarios a la hora de utilizar la aplicación, se ha decidido añadir una vista donde aparezcan recomendaciones de artistas en base a los gustos personales de los usuarios.

Contenido: Consta de una vista donde se muestra una lista de artistas, donde se muestra una imagen del artista o grupo, su nombre y el género principal al que pertenecen. La lista mostrada varía en función de si se trata de un usuario registrado o no. En caso de que esté registrado se mostrarán todos aquellos artistas que pertenezcan al mismo género o uno similar a los artistas o grupos que el usuario siga. En caso contrario se mostrarán los artistas con mayor número de seguidores, puesto que podemos asumir que serán los mejores.

Funciones: Realiza una consulta a la base de datos retornando la información de los artistas. Dicha consulta varía en función de si es un usuario registrado o no, dando lugar a una consulta simple sobre una tabla o la implicación de varias para establecer la relación entre el usuario y los diversos artistas.

Diseño: En cuanto a diseño se presenta una lista con los artistas relacionados a los que sigue actualmente. Los objetos de dicha lista muestran la imagen, el nombre y el género musical del grupo o artista.

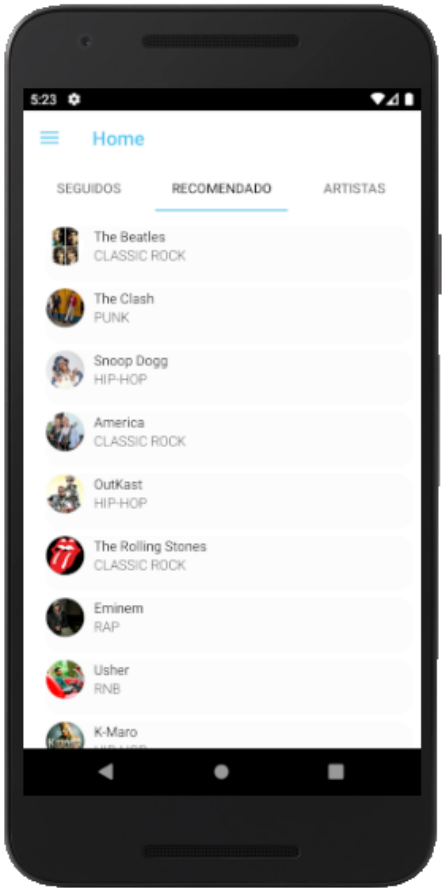


Figura 3.6: Modelo de recomendaciones

3.4.6. Notificaciones

A modo de mejorar aún más la experiencia de usuario se ha decidido implementar un sistema de notificaciones que avise al usuarios de cuando se han añadido nuevas noticias relacionadas con los artistas que sigue.

Contenido: Se implementa una notificación básica de Android, que consta de un mensaje base que indica que tiene nuevas noticias seguidas del nombre del artista implicado.

Funciones: Por medio de una función en segundo plano se realiza una monitorización del contenido de la base de datos, la cual al ser actualizada enviará un mensaje a la aplicación, la cual ejecutará la notificación.

3.5. Seguridad de contraseñas

3.5.1. El cifrado

Para comenzar a hablar sobre la seguridad de las contraseñas es importante conocer el término cifrado. En el mundo de la informática, el cifrado es la conversión de datos de un formato legible a un formato codificado, que solo se pueden leer o procesar después de haberlos descifrado.

El cifrado es el elemento fundamental de la seguridad de datos y es la forma más simple y la primera barrera para tratar impedir que alguien obtenga la información de un sistema informático con fines malintencionados o poco éticos.

Más allá de las ventajas lógicas que ofrece esta práctica sobre la información privada contra robos o amenazas, el cifrado también ofrece un medio para demostrar tanto la veracidad como el origen de la información. Se puede utilizar para verificar el origen de un mensaje y confirmar que no ha sufrido modificaciones durante la transmisión.

Figura 3.7: Modelo de cifrado

3.5.2. Tipos de cifrado

En la actualidad existe una gran cantidad de tipos de cifrado, no obstante hay dos que destacan sobre el resto. Estos tipos más extendidos y aceptados son los siguientes:

- **Criptografía Simétrica:** La criptografía de clave simétrica, también llamada criptografía de clave secreta o criptografía de una clave, es un método criptográfico en el cual se usa una misma clave para cifrar y descifrar mensajes en el emisor y el receptor. Las dos partes que se comunican han de ponerse de acuerdo de antemano sobre la clave a usar. Una vez que ambas partes tienen acceso a esta clave, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra con la misma clave.
- **Criptografía Asimétrica:** La criptografía asimétrica , también llamada criptografía de clave pública o criptografía de dos claves, es el método criptográfico que usa un par de claves para el envío de mensajes. Las dos claves pertenecen a la misma persona que recibirá el mensaje. Una clave es pública y se puede entregar a cualquier persona, la otra clave es privada y el propietario debe guardarla de modo que nadie tenga acceso a ella. Además, los métodos criptográficos garantizan que esa pareja de claves sólo se puede generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves.

Conociendo mejor estos tipos de cifrados, ahora veremos las ventajas y desventajas de ambos y las posibilidades que ofrecen a la hora de realizar un proyecto informático.

Velocidad: el cifrado simétrico es mucha más rápido y ágil, lo cual aporta ventajas en los tiempos de carga, por contraparte el modelo asimétrico consta de una carga más lenta, lo cual no es una desventaja absoluta, puesto que favorece evitar los ataques por fuerza bruta aumentando los tiempos de carga, mientras que para el usuario no supone tiempos de carga tan elevados.

Seguridad: el cifrado simétrico no es tan seguro, ya que el hecho de publicar la clave lo hace muy vulnerable. El cifrado asimétrico ofrece como ventaja, el hecho de poder establecer comunicaciones de forma segura, donde compartir claves públicas a terceros. Este cifrado establece clave pública manteniendo la clave privada de cada usuario.

Número de claves: la administración de claves también es un beneficio al usar el cifrado asimétrico, esto debido q que se necesita sólo un par de claves por usuario, para cada uno y, así poder cifrar los mensajes para los usuarios involucrados independientemente del número. En cambio, con el cifrado simétrico, a medida que aumenta el número de usuarios, aumenta el número de claves.

En cuanto a qué método utilizar para el desarrollo de un proyecto, se puede observar como no existe una respuesta definitiva para todos y, se trata de una elección en base a los requisitos técnicos del mismo, así como de las preferencias del desarrollador o la propia empresa para manejar sus datos sensibles en base a si priorizan la seguridad o la velocidad de respuesta, incluso optando por métodos híbridos.

3.5.3. MD5 salt

El la actualidad el método más simple aceptado para cifrar contraseñas es un doble MD5, donde se cifra una cadena por medio del cifrado y ese resultado se vuelve a cifrar. Esto se hace debido a que MD5 es un sistema muy vulnerable, que permite descifrar una gran cantidad de datos en un tiempo muy corto por medio de la fuerza bruta. Como alternativa se ha propuesto la utilización de salt, donde se amplían los tiempos de carga de las contraseñas. Este método tampoco aumenta en gran cantidad la seguridad, pero ofrece una poción rápida y de fácil implementación, aún así no se descarta un cambio de cifrado en un futuro para la aplicación.

En criptografía, la salt es una cadena aleatoria que se agrega a una palabra de entrada, para generar un hash diferente que solo con la palabra MD5 realmente no ofrece esta función en el algoritmo criptográfico, pero puede concatenar dos cadenas para obtener el mismo resultado

El principal motivo para utilizar salt en conjunto con MD5, es que favorece la seguridad ante ataques de fuerza bruta, los cuales son aquellos que tratan de introducir todas las posibles contraseñas hasta dar con la correcta, incrementando los tiempos de carga. En la próxima tabla podemos observar el número de intentos necesarios para comprobar todas las posibilidades de contraseña en base al número de caracteres, teniendo en cuenta mayusculas, minusculas y numeros.

Caracteres	Combinaciones
3	140.608
4	7.311.616
5	380.204.032
6	19.770.609.664

Tabla 3.1: Combinaciones de contraseñas

3.6. Interfaz gráfica

3.6.1. ¿Qué es el diseño UX/UI?

A pesar de lo que se suele pensar, UX y UI son conceptos diferentes. Sin embargo, están muy relacionados. Y la combinación de ambos es esencial para crear un diseño ideal para el consumidor.

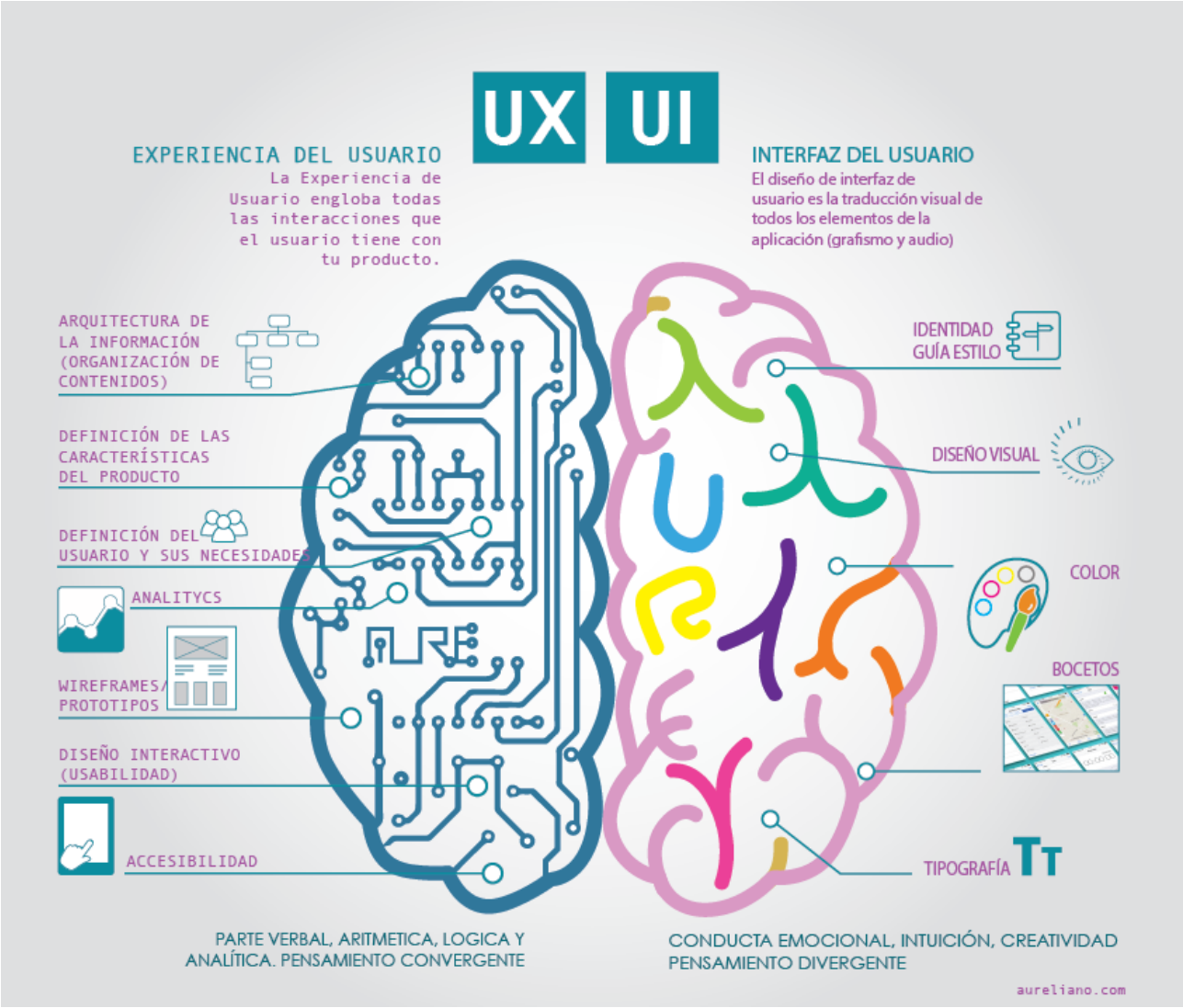


Figura 3.8: Modelo UI/UX

UX (User Experience): hace referencia a la forma en la que los usuarios interactúan con un producto o servicio. Es decir, cómo y para qué un usuario utiliza un objeto o interactúa con una web o app.

Sin duda, la esencia del diseño UX está en el conocimiento de los usuarios. En otras palabras, para crear un buen diseño UX hay que comprender las necesidades de los usuarios y, por supuesto, satisfacerlas de una forma simple y clara. Por lo tanto, un buen resultado es aquel que es útil para el usuario.

UI (User Interface): se centra en la parte visual. Es decir, si UX se encarga de que un producto sea útil para los usuarios, UI lo hace atractivo y visual.

Los colores, la tipografía, las imágenes son algunos de los elementos con los que trabaja el diseñador UI para hacer que un producto sea atractivo. Pero de nada sirve tener un producto bonito si no satisface las necesidades de los usuarios para los que está pensado. Por eso, UX y UI deben ir de la mano para lograr un producto 100 % pensado para los clientes.

3.6.2. Diseño implementado

Una vez entendidos los conceptos ux/ui, se ha optado por desarrollar una interfaz funcional y a la vez agradable visualmente. El apartado ux se consigue manteniendo una postura minimalista, con la mínima cantidad posible de botones en pantalla, añadiendo los estrictamente necesarios por parte del usuario.

Para el apartado ui, se ha realizado un estudio de varias paletas de colores que permitan un diseño tanto con motivos de fondo blanco, como negro dando lugar a una elección de dos colores azules, uno claro y uno oscuro para los efectos y detalles del logo y los botones, así como para los textos de apoyo. Por otro lado a la hora de presentar la información se elige implementar los cuadros de texto, botones, imágenes o listas con los bordes redondeados, lo cual da un aspecto menos agresivo para la vista y dan una mayor sensación de fluidez a la hora de navegar por la misma.

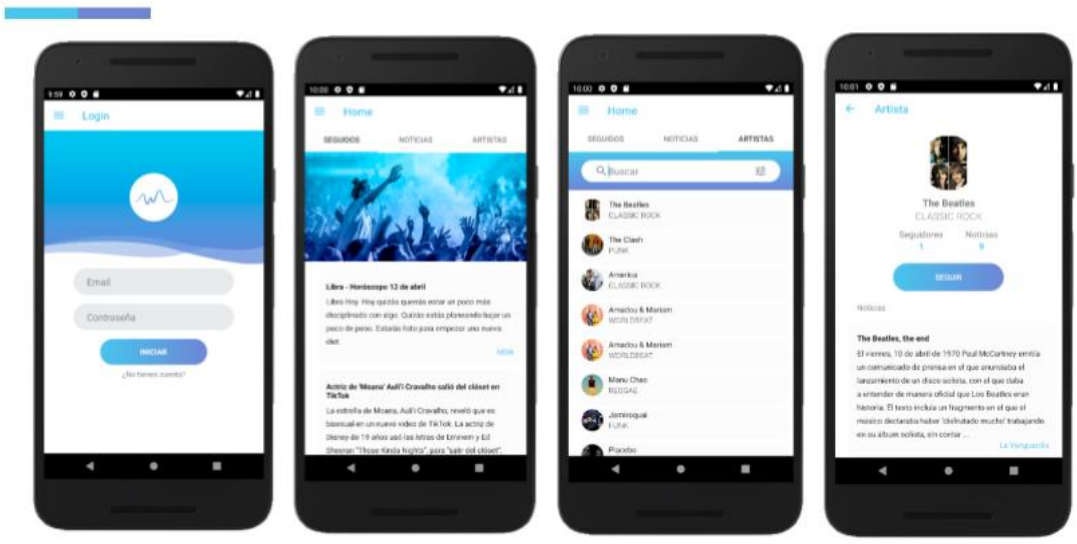


Figura 3.9: Diseño de interfaz

3.6.3. Diseño del isotipo

Siguiendo con la idea de colores y líneas poco agresivas se ha buscado un isotipo ideal para representar el concepto de la aplicación y tratándose de una aplicación musical, que mejor que una onda de sonido para representar la idea.

Figura 3.10: Isotipo TodoMusica

Capítulo 4

Desarrollo del servidor

A la hora de realizar un aplicación de este tipo es necesario que el usuario acceda a los datos, los cuales estarán almacenados en un servidor generalmente. Esta interacción se lleva a cabo mediante consultas, en las cuales accede desde datos relacionados a artistas, como a noticias o a su propia información personal.

El servidor, además de almacenar ser capaz de devolver los datos que ya posee, es capaz de actualizar su contenido por medio del uso de scripts, así como actualizar la base de usuarios, por medio del servicio de autentificación. Es por esto que es necesario establecer una correcta configuración del mismo para evitar conflictos de comunicación u otros posibles problemas, como los tiempos de carga.

Figura 4.1: Ilustración de servidor cloud

4.1. Herramientas

4.1.1. PHPMyadmin

phpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas web, utilizando un navegador web. Actualmente puede crear y eliminar Bases de Datos, crear, eliminar y alterar tablas, borrar, editar y añadir campos, ejecutar cualquier sentencia SQL, administrar claves en campos, administrar privilegios, exportar datos en varios formatos y está disponible en 72 idiomas. Se encuentra disponible bajo la licencia GPL Versión.

Este proyecto se encuentra vigente desde el año 1998, siendo el mejor evaluado en la comunidad de descargas de SourceForge.net como la descarga del mes de diciembre del 2002. Como esta herramienta corre en máquinas con Servidores Webs y Soporte de PHP y MySQL, la tecnología utilizada ha ido variando durante su desarrollo.

4.1.2. 000webhost administrador de ficheros

Un sistema de almacenamiento en la nube, también conocido por su término en inglés: cloud, no es más que un disco duro, pero en Internet.

Es decir, la nube es un un lugar donde puedes almacenar tus archivos y acceder a ellos fácilmente desde cualquier dispositivo, en cualquier lugar del mundo.

La plataforma 000webhost en la que se alojan los datos de la aplicación, provee una opción para almacenar contenidos por medio de un administrador de ficheros propios. Este administrador permite alojar ficheros de todo tipo, desde imágenes hasta ficheros de código, es por esto que se decide almacenar aquí todos aquellos scripts desarrollados en lenguaje .php necesarios para establecer las relaciones entre la aplicación y los contenidos de la base de datos.

4.1.3. 000webhost administrador de tareas CRON

En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano que ejecuta procesos o guiones a intervalos regulares. Los procesos que deben ejecutarse y la hora a la que deben hacerlo se especifican en el archivo crontab. El nombre cron proviene del griego chronos que significa "tiempo".

Para la ejecución de tareas en segundo plano, necesarias para la actualización de información, la plataforma ofrece una opción sencilla, donde se almacena un fichero de extensión .php en el administrador de ficheros y luego se referencia en el panel de administración indicando cada cuento tiempo se desea ejecutar.

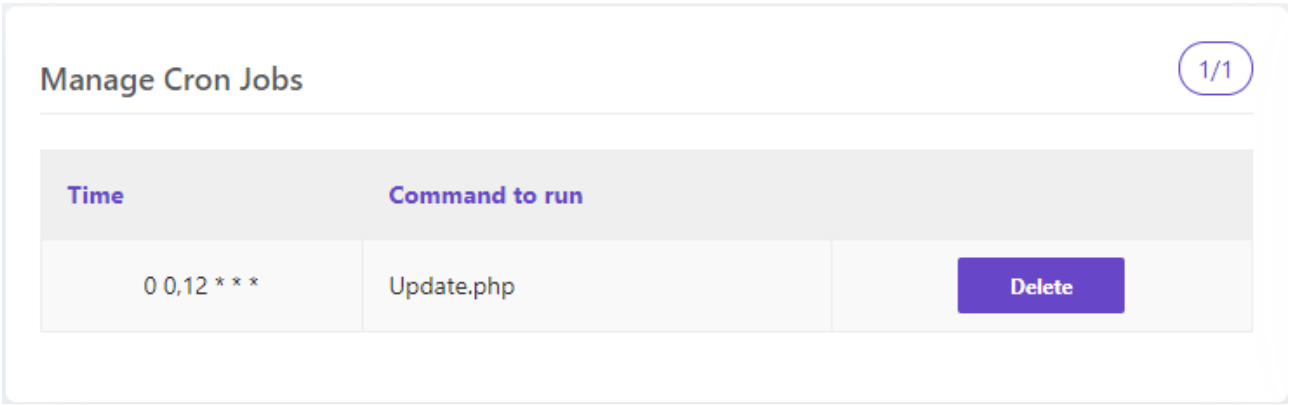


Figura 4.2: Interfaz de tareas CRON

4.2. Modelo y diseño

Debido a las necesidades de la aplicación, se ha optado por implementar un modelo cliente-servidor. Esta arquitectura se aplica en diferentes modelos informáticos alrededor del mundo donde su propósito es mantener una comunicaciones de información entre diferentes entidades de una red mediante el uso de protocolos establecidos y el apropiado almacenaje de la misma.

La principal importancia de este modelo es que permite conectar a varios clientes a los servicios que provee un servidor y como sabemos hoy en día, la mayoría de las aplicaciones y servicios tienen como gran necesidad que puedan ser consumidos por varios usuarios de forma simultánea.

El cliente es un computador pequeño con una estructura al igual a la que tenemos en nuestras oficinas u hogares la cual accede a un servidor o a los servicios del mismo a través de Internet o una red interna. Un claro ejemplo a este caso es la forma en que trabaja una empresa modelo con diferentes computadores donde cada uno de ellos se conectan a un servidor para poder obtener archivos de una base de datos o servicios ya sea correos electrónicos o aplicaciones.

El servidor al igual que el cliente, es una computadora pero con diferencia de que tiene una gran capacidad que le permite almacenar gran cantidad de diversos de archivos, o correr varias aplicaciones en simultáneo para así nosotros los clientes poder acceder los servicios.

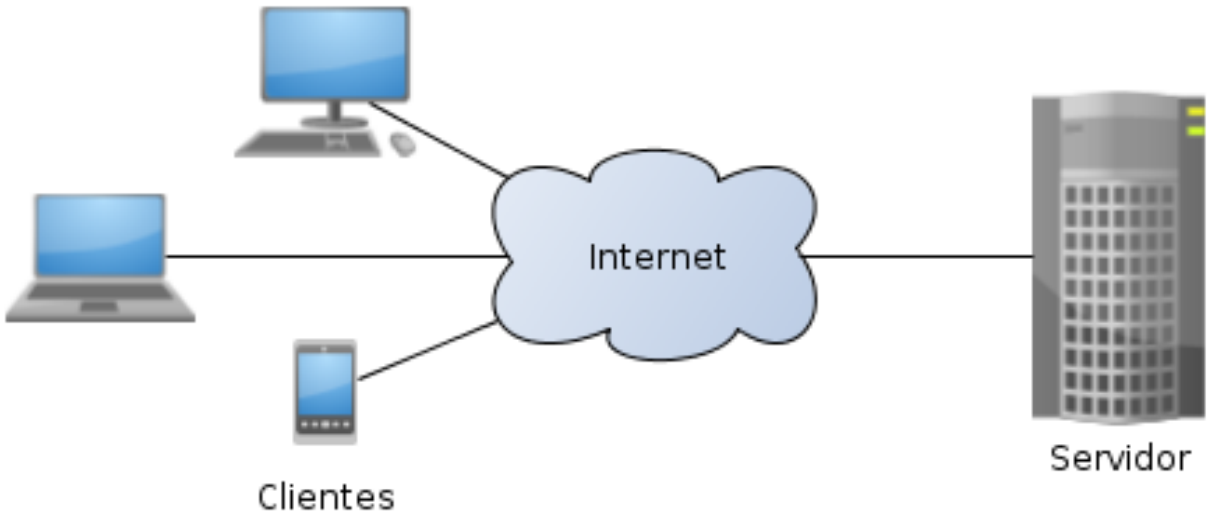


Figura 4.3: Modelo cliente-servidor

Este modelo cliente servidor tiene varias ventajas y desventajas las cuales son importantes mencionar y conocer a la hora de establecer si es lo que estamos necesitando o si se acomoda a lo que estamos buscando.

Ventajas:

- Facilita la integración entre diferentes sistemas y comparte información.
- Los sistemas creados bajo este esquema proveen mayor interacción con el usuario.
- La estructura modular facilita la integración de nuevas tecnologías y el crecimiento.
- Este modelo proporciona la capacidad de generar un orden de trabajo separando la interacción de las posibles áreas de trabajo sin generar conflictos.

Desventajas:

- Requiere habilidad para que un servidor sea reparado.
- La seguridad representa otro problema, el hecho que se comparte canales de información entre servidores y clientes favorece la aparición de problemas por malware.
- Se encuentra limitado por los costes económicos, debido a la necesidad de un hardware y un software especializado.

4.3. Funcionalidades

4.3.1. Base de datos

Se llama base de datos, o también banco de datos, a un conjunto de información perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, análisis y/o transmisión. Existen actualmente muchas formas de bases de datos, que van desde una biblioteca hasta los vastos conjuntos de datos de usuarios de una empresa de telecomunicaciones.

Las bases de datos son el producto de la necesidad humana de almacenar la información, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnéticas.

El manejo de las bases de datos se lleva mediante sistemas de gestión (llamados DBMS por sus siglas en inglés: Database Management Systems o Sistemas de Gestión de Bases de Datos).

Para el desarrollo de la aplicación se ha creado una base de datos relacional SQL, en la cual se han definido las siguientes tablas en base a las necesidades que se presentan para un correcto funcionamiento del proyecto.

Artist: En esta tabla se almacenan los contenidos relacionados con la información personal de los artistas o grupos que se encuentran disponibles en la aplicación. La estructura de datos utilizada es la siguiente.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	id 🔑	int(11)			No	*Ninguna*
☐	2	name	varchar(500)	utf8_unicode_ci		No	*Ninguna*
☐	3	picture	varchar(500)	utf8_unicode_ci		No	*Ninguna*
☐	4	followers	int(11)			No	*Ninguna*
☐	5	genre	varchar(800)	utf8_unicode_ci		No	*Ninguna*
☐	6	bio	varchar(20000)	utf8_unicode_ci		No	*Ninguna*

Figura 4.4: Tabla: Artists

- id: Campo donde se almacena el número identificador del artista o grupo
- name: Campo donde se almacena el nombre del artista o grupo
- picture: Campo donde se almacena la URL, donde se encuentra alojada la imagen de portada del artista o grupo
- followers: Campo numérico donde se almacena el número de seguidores del artista o grupo
- genre: Campo de texto en formato JSON, donde se almacena el conjunto de géneros musicales a los que pertenece el grupo o artista
- bio: Campo donde se almacena una pequeña biografía del artista o grupo

User: Esta tabla contiene la información personal asociada a cada usuario registrado en la aplicación. La estructura de datos es la siguiente.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios
☐	1	id 🔑	int(11)			No	*Ninguna*	
☐	2	name	varchar(50)	utf8_unicode_ci		No	*Ninguna*	
☐	3	surname	varchar(50)	utf8_unicode_ci		No	*Ninguna*	
☐	4	username	varchar(50)	utf8_unicode_ci		No	*Ninguna*	
☐	5	email	varchar(50)	utf8_unicode_ci		No	*Ninguna*	
☐	6	password	varchar(250)	utf8_unicode_ci		No	*Ninguna*	

Figura 4.5: Tabla: User

- id: Campo donde se almacena el número identificador del usuario
- name: Campo donde se almacena el nombre o nombres del usuario
- surname: Campo donde se almacena el apellido o apellidos del usuario
- username: Campo donde se almacena el nick o nombre de usuario
- email: Campo donde se almacena el email del usuario
- password: Campo donde se almacena la contraseña cifrada del usuario

Follow: Esta tabla establece la relación de seguimiento entre el usuario y un artista o grupo. La estructura de la misma es la siguiente.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**sender_id**	int(11)			No	*Ninguna*
☐	2	**receiver_id**	int(11)			No	*Ninguna*

Figura 4.6: Tabla: Follow

- sender_id: Campo donde se indica el id del usuario que sigue a un artista
- receiver_id: Campo donde se almacena el id del artista o grupo seguido por el usuario

News: Tabla donde se almacenan las noticias recopiladas por la aplicación y que son posteriormente mostradas en la feed. La estructura presentada en la tabla es la siguiente.

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
☐	1	**artist_id**	int(11)			No	*Ninguna*
☐	2	**tittle**	varchar(250)	utf8_unicode_ci		No	*Ninguna*
☐	3	**link**	varchar(250)	utf8_unicode_ci		No	*Ninguna*
☐	4	**date**	varchar(50)	utf8_unicode_ci		No	*Ninguna*
☐	5	**snippet**	varchar(500)	utf8_unicode_ci		No	*Ninguna*
☐	6	**thumbnail**	varchar(250)	utf8_unicode_ci		No	*Ninguna*
☐	7	**domain**	varchar(50)	utf8_unicode_ci		No	*Ninguna*

Figura 4.7: Tabla: NewsT

- artist_id: Campo donde se almacena el id del grupo o artista al que pertenece la noticia
- tittle: Campo del título o cabecera de la noticia
- link: Campo donde se almacena el enlace a la noticia original
- date: Campo donde se almacena la fecha de la publicación de la noticia en la base de datos
- snippet: Campo donde se almacena una pequeña introducción a la noticia
- thumbnail: Campo donde se almacena una imagen referencia a la noticia
- domain: Campo donde se almacena el nombre del periódico o revista que publica la noticia

Capítulo 5

Conclusiones y líneas futuras

Durante el desarrollo del proyecto se ha realizado un estudio, tanto del sector musical, como las nuevas tecnologías aplicadas al desarrollo de aplicaciones nativas para dispositivos android, desde el modelizado, hasta las nuevas técnicas de desarrollo UI/UX para cumplir con los propósitos y además ofrecer un producto de alta calidad. Durante el proceso se han producido gran diversidad de cambios en base a los problemas surgidos, siendo esto una fuente de inspiración para mejorar las funcionalidades propuestas en un origen. El desarrollo se ha llevado a cabo en dos secciones muy importantes:

En primer lugar, la configuración del servidor, donde se ha creado una base de datos con diversidad de tablas, cuyo objetivo es aportar mayor valor a la aplicación. La comunicación con dicha base de datos se realiza por medio de sentencias y funciones escritas en lenguaje PHP y la información no relacionada con los usuarios ha sido obtenida por medio de APIs. La selección de estas APIs se ha realizado tras un estudio riguroso de las ofertas del mercado, teniendo en cuenta que estas debían ser gratuitas, punto que ha obligado a modificar partes del código debido a los cambios de políticas de algunas utilizadas, las cuales han comenzado a ser de pago.

En cuanto al desarrollo de la aplicación en sí, se ha realizado en java, puesto que eran los requisitos iniciales propuestos, siendo utilizadas así funciones y sentencias específicas de android, por medio de las librerías andoidx. Por otro lado se han utilizado otras librerías necesarias para un correcto funcionamiento, destacando de entre otras, volley, picasso o retrofit. En cuanto al diseño visual se ha utilizado el lenguaje XML, típico en aplicaciones nativas de android.

Basándonos en los objetivos iniciales, podemos concluir que el proyecto ha sido un éxito, puesto que no solo se han obtenido los resultados esperados, sino que se han añadido nuevas funcionalidades no programadas en un inicio, siendo esta el sistema de recomendaciones o el propio diseño, para el cual ha sido necesario un estudio y una serie de test de usuario, para ofrecer una composición y una gama de colores amigables.

El proyecto, no se encuentra terminado en su totalidad, por lo cual es necesario proponer una nueva línea de trabajos futuros. Considerando el actual estado, las próximas funcionalidades a implementar serán las siguientes:

- **Administración del perfil de usuario:** En el cual cada usuario pueda modificar sus datos personales y gestionar los perfiles.
- **Pestaña de configuración:** En la cual se permita elegir el idioma en el cual se desea encontrar la información, el tema de la aplicación(Día/Noche) o gestionar la preferencia de las notificaciones.
- **Usabilidad y accesibilidad:** Añadir mayor accesibilidad y usabilidad a la aplicación, cumpliendo con los estándares europeos WCAG 2.1, para el desarrollo de aplicaciones.
- **Desarrollo de una plataforma web:** Permitir un uso similar o igual al de la aplicación desde un navegador web.
- **Desarrollo de roles:** Donde exista un rol administrador, el cual tenga acceso a las estadísticas y permisos para modificar el contenido de los apartados.

Capítulo 6

Summary and Conclusions

During the development of the project, a study has been carried out, both of the music sector, and of the new technologies applied to the development of native applications for android devices, from modeling, to new UI / UX development techniques to fulfill the purposes and also offer a high quality product. During the process has produced a great diversity of changes based on the problems that have arisen, this being a source of inspiration to improve the proposed functionalities in an origin. The development has been carried out in two very important sections:

Firstly, the server configuration, where a database with a variety of tables has been created, the objective of which is to add more value to the application. The communication with said database is carried out by means of sentences and functions written in PHP language and the information not related to users has been obtained through APIs. The selection of these APIs has been made after a rigorous study of the market offers, bearing in mind that these should be free, a point that has forced to modify parts of the code due to changes in the policies of some used, which have begun to be paid.

As for the development of the application itself, it has been done in java, since they were the initial requirements proposed, thus using functions and sentences specific to android, through the andoidx libraries. On the other hand, other libraries necessary for proper operation have been used, standing out among others, volley, picasso or retrofit. Regarding visual design, the XML language has been used, typical in native android applications.

Based on the initial objectives, we can conclude that the project has been a success, since not only have the expected results been obtained, but they have been added new functionalities not programmed at first, this being the system of recommendations or the design itself, for which a study and a series of tests have been necessary user, to offer a friendly composition and range of colors.

The project is not fully finished, so it is necessary to propose a new line of future work. Considering the current state, the next functionalities to implement will be the following:

- **User profile administration:** In which each user can modify their personal data and manage the profiles.
- **Configuration tab:** In which you can choose the language in which you want to find the information, the subject of the application (Day / Night) or manage the preference of notifications.
- **Usability and accessibility:** Add greater accessibility and usability to the application, complying with the European WCAG standards 2.1, for the development of applications.
- **Development of a web platform:** Allow similar or equal use to the application from a web browser.
- **Role development:** Where there is an administrator role, which has access to statistics and permissions to modify the content of the sections.

Capítulo 7

Presupuesto

Dadas las características del proyecto, se expone un presupuesto adaptado a la duración de dos meses, realizado por un único trabajador. En las siguientes tablas se detalla el presupuesto, tanto del trabajo realizado, como el de los elementos necesarios para su correcto desarrollo.

7.1. Mobiliario

En este apartado se desglosa el material de oficina necesario para el puesto de trabajo del desarrollador.

Tipo	Unidades	Precio/unidad	Total
Escritorio	1	129€	129€
Silla	1	103€	103€
Ordenador	1	800€	800€
Monitor	1	200€	200€
Periféricos	1	50€	50€

Tabla 7.1: Presupuesto mobiliario

7.2. Herramientas

En este apartado se desglosan los gastos asociados a las herramientas necesarias para el correcto desarrollo de la aplicación.

Tipo	Licencia	Total
Visual studio	Gratuita	0€
Android studio	Gratuita	0€
000webhost studio	Gratuita	0€

Tabla 7.2: Presupuesto herramientas

7.3. Suministro y similares

En este apartado se desglosan los gastos asociados al mantenimiento y suministro necesarios para el correcto desarrollo del proyecto.

Tipo	Meses	Precio/Mes	Total
Alquiler	3	500€	1500€
Electricidad	3	50€	150€
Internet	3	100€	300E
Hosting	3	0€	0€

Tabla 7.3: Presupuesto suministros

7.4. Salario

En este apartado se muestran los costes de contratación del desarrollador, teniendo en cuenta que este será un programador junior, con poca experiencia laboral.

Tipo	Horas	Precio/Hora	Total
Análisis	130	42€	5460€
Programación	220	32€	7040€

Tabla 7.4: Presupuesto salario

7.5. Presupuesto final

Presupuesto final, obtenido de la suma de los sub-apartados anteriormente expuestos, necesarios para el correcto desarrollo de la aplicación.

Descripción	Total
Mobiliario	1282€
Herramientas	0€
Suministros	1950€
Salario	12500€
Total	15732€

Tabla 7.5: Presupuesto final

Apéndice A

Algoritmos del servidor

A.1. Algoritmo de obtención de artistas

```
/*********************************************************************************
*
* AddArtist.php
*
*********************************************************************************
*
* AUTOR: Sergio Manuel Rodríguez Vega
*
*
* FECHA: 12/04/2020
*
*
* DESCRIPCION: Algoritmo que actualiza la base de datos de los artistas de la aplicación
*
*
*********************************************************************************/

<?php
    $con = mysqli_connect("localhost", "id12880103_root", "root123", "id12880103_todomusica");
    $urlName = 'https://api.deezer.com/artist/';

    if ($fh = fopen('ID.txt', 'r')) {
        while (!feof($fh)) {
            $i = fgets($fh);
        }
        fclose($fh);
    }
    $error = 0;

    while ($i <= 100){
        $auxurl = $urlName . $i;
        $json = file_get_contents($auxurl);
        $data = json_decode($json,true);

        if ( isset($data['name']) ){
            $error = 0;
            $name = $data['name'];
            $picture = $data['picture_xl'];
            $artistName = strtr($name, " ", "+");
```

```
            echo $name . "\n";

            $urlContent = "http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=" . $artis
            $jsonContent = file_get_contents($urlContent);
            $dataAux = json_decode($jsonContent, true);

            if (isset($dataAux['artist'])){
                $dataContent = $dataAux['artist'];
                $dataTags = $dataContent['tags'];

                $auxArray = array();
                foreach ($dataTags['tag'] as $j){
                    array_push($auxArray, $j['name']);
                }

                $genre = json_encode($auxArray);
                $bio = substr($dataContent['bio']['content'],0,15000);
                $followers = 0;

                $statement = mysqli_prepare($con, "INSERT INTO artists (name, picture, followers, ge
                mysqli_stmt_bind_param($statement, "ssiss", $name, $picture, $followers, $genre, $bi
                mysqli_stmt_execute($statement);
            }
        }

        $i++;
        file_put_contents('ID.txt', $i);
    }

?>
```

A.2. Algoritmo de seguimiento de artistas

```
/*****************************************************************************
 *
 * Follow.php
 *
 *****************************************************************************
 *
 * AUTOR: Sergio Manuel Rodríguez Vega
 *
 * FECHA: 26/04/2020
 *
 * DESCRIPCION: Algoritmo que establece el seguimiento de un artista
 *
 *
 *****************************************************************************/

 <?php
    $con = mysqli_connect("localhost", "id12880103_root", "root123", "id12880103_todomusica");

    $userId = $_POST["userId"];
```

```
    $artistId = $_POST["artistId"];
    $statement = mysqli_prepare($con, "INSERT INTO follow (sender_id, receiver_id) VALUES (?, ?)");
    mysqli_stmt_bind_param($statement, "ii", $userId, $artistId);
    mysqli_stmt_execute($statement);

    $response = array();
    $response["success"] = true;

    echo json_encode($response);
?>
```

A.3. Algoritmo de obtención de noticias

```
/*********************************************************************************
 *
 * Update.php
 *
 ********************************************************************************
 *
 * AUTOR: Sergio Manuel Rodríguez Vega
 *
 * FECHA: 31/04/2020
 *
 * DESCRIPCION: Fichero de actualización de noticias
 *
 *
 ********************************************************************************/

    <?php

        // Adding libs
        require_once __DIR__.'/gmailAPI/vendor/autoload.php';

        // Get client function
        function getClient() {
            $client = new Google_Client();
            $client->setAccessType('offline');
            $client->setApprovalPrompt('force');
            $client->setApplicationName('Todo Musica');
            $client->addScope('https://mail.google.com/');
            $client->setAuthConfigFile('credentials.json');
            $client->setScopes(Google_Service_Gmail::MAIL_GOOGLE_COM, Google_Service_Gmail::GMAIL_CO
            $client->setRedirectUri('https://todomusicatest.000webhostapp.com/test.php');

            // Load previously authorized token from a file, if it exists.
            // The file token.json stores the user's access and refresh tokens, and is
            // created automatically when the authorization flow completes for the first
            // time.
            $tokenPath = 'token.json';
            if (file_exists($tokenPath)) {
                $accessToken = json_decode(file_get_contents($tokenPath), true);
                $client->setAccessToken($accessToken);
            }
```

```
    // If there is no previous token or it's expired.
    if ($client->isAccessTokenExpired()) {
        // Refresh the token if possible, else fetch a new one.
        if ($client->getRefreshToken()) {
            $client->fetchAccessTokenWithRefreshToken($client->getRefreshToken());
        } else {
            // Request authorization from the user.
            if (!isset($_GET['code'])) {
                $auth_url = $client->createAuthUrl();
                header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));
            }

            else {
                $client->authenticate($_GET['code']);
                $merengue = $client->getAccessToken();

                if (!file_exists(dirname($tokenPath))) {
                    mkdir(dirname($tokenPath), 0700, true);
                }

                file_put_contents($tokenPath, json_encode($merengue));

                $authCode=$merengue['refresh_token'];
                echo $authCode;
            }
        }
        // Save the token to a file.
        /**/
    }
    return $client;
}


// Read mail messages function
function listMessages($service, $userId) {
    $pageToken = NULL;
    $messages = array();
    $opt_param = array();

    $opt_param["maxResults"] = 500;
    $opt_param["q"] = "is:unread";
    //do {
    try {
        if ($pageToken) {
        $opt_param['pageToken'] = $pageToken;
        }
        $messagesResponse = $service->users_messages->listUsersMessages($userId, $opt_param)
        if ($messagesResponse->getMessages()) {
            $messages = array_merge($messages, $messagesResponse->getMessages());
            $pageToken = $messagesResponse->getNextPageToken();
        }
    } catch (Exception $e) {
        print 'An error occurred: ' . $e->getMessage();
    }

    foreach ($messages as $message) {
    }
```

```
    return $messages;
}

// Get header function
function getHeaderArr($dataArr) {
    $outArr = [];
    foreach ($dataArr as $key => $val) {
        $outArr[$val->name] = $val->value;
    }
    return $outArr;
}

// Get body function
function getBody($dataArr) {
    $outArr = [];
    foreach ($dataArr as $key => $val) {
        $outArr[] = base64url_decode($val->getBody()->getData());
        //break;
    }
    return $outArr;
}

// URL decode function
function base64url_decode($data) {
    $content = base64_decode( strtr($data, '-_,', '+/='));

    $content = str_replace('<', '', $content);
    $content = str_replace('>', '', $content);

    return $content;
}

// Body parser function
function bodyParse($body, $artist) {
    $str = empty($body[1]) ? '' : $body[1];

    $aux = explode ("/script", $str);
    $aux1 = $aux[0];
    $aux2 = substr($aux1, 94);


    $json = json_decode($aux2, true);

    foreach ( $json["cards"][0]["widgets"] as $key => $value) {
        $tittle = $value["title"];
        $snippet = $value["description"];
        $url = $value["url"];
        $domain = "unknown";

        insertDatabase ($artist, $tittle, $domain, $snippet, $url);
    }
}

// Start with upper function
function starts_with_upper($str) {
    $chr = substr($str, 0, 1);
```

```
    return IntlChar::isupper($chr);
}

// IsUrl function
function isUrl($str){
    if (substr($str, 0, 4) === 'https') {
        return TRUE;
    }else{
        return FALSE;
    }
}

// Get message function
function getMessage($service, $userId, $messageId) {
    try {
        $message = $service->users_messages->get($userId, $messageId);

        return $message;
    } catch (Exception $e) {
        print 'An error occurred: ' . $e->getMessage();
    }
}

// List label function
function listLabels($service, $userId, $optArr = []) {
    $results = $service->users_labels->listUsersLabels($userId);

    if (count($results->getLabels()) == 0) {
    print "No labels found.\n";
    } else {
    print "Labels:\n";
    foreach ($results->getLabels() as $label) {
        printf("- %s\n", $label->getName());
    }
    }
}

// Insert database function
function insertDatabase ($artist, $tittle, $domain, $snippet, $url) {

    $con = mysqli_connect("localhost", "id12880103_root", "root123", "id12880103_todomusica"

    echo $artist . "<br>";
    $finalArtist = "%" . $artist . "%";

    $statement = mysqli_prepare($con, "SELECT id FROM artists WHERE name LIKE ?");
    mysqli_stmt_bind_param($statement, "s", $finalArtist);
    mysqli_stmt_execute($statement);

    mysqli_stmt_store_result($statement);
    mysqli_stmt_bind_result($statement, $id);

    while(mysqli_stmt_fetch($statement)){
        $finalId = $id;
        echo "ola" . "<br>";
    }

    $thumbnail = "no image";
```

```
        $date = date("h:i Y-m-d");

        $statement = mysqli_prepare($con, "INSERT INTO news (artist_id, tittle, link, date, snip
        mysqli_stmt_bind_param($statement, "isssss", $finalId, $tittle, $url, $date, $snippet,
        mysqli_stmt_execute($statement);
    }

    // Main function
    $client = getClient();
    $service = new Google_Service_Gmail($client);
    $user = 'me';

    //listLabels($service, $user);

    $messages = listMessages($service, $user, [
        'maxResults' => 1,
        'labelIds' => 'INBOX',
    ]);

    foreach ($messages as $message) {
        //print 'Message with ID: ' . $message->getId() . "\n";

        $msgObj = getMessage($service, $user, $message->getId());

        $headerArr = getHeaderArr($msgObj->getPayload()->getHeaders());

        $subject = $headerArr['Subject'];
        //echo $alert;
        //echo "<br>";

        if (substr($subject, 0, 18) === 'Alerta de Google -') {
            $artist = substr($subject, 19);
            $bodyArr = getBody($msgObj->getPayload()->getParts());
            bodyParse($bodyArr, $artist);
        }

        $mods = new Google_Service_Gmail_ModifyMessageRequest();
        $mods->setRemoveLabelIds(array('UNREAD'));
        $service->users_messages->modify('me', $message->getId(), $mods);
    }

?>
```

Bibliografía

[1] Android Developers. Cómo depurar tu app. Acceso: 10/04/2020
https://developer.android.com/studio/debug/
[2] Android Developers. Intents y filtros. Acceso: 10/04/2020
https://developer.android.com/guide/components/intents-filters?hl=es
[3] Android Developers. Tareas en segundo plano. Acceso 02/06/2020
https://developer.android.com/training/best-background?hl=es
[4] Android Developers. Animaciones y transiciones. Acceso 08/05/2020
https://developer.android.com/training/animation?hl=es
[5] Raywenderlich. Tutrorial de encriptación android. Acceso 15/04/2020
https://www.raywenderlich.com/778533-encryption-tutorial-for-android-getting-started
[6] Microsoft. Cognitive services. Acceso: 10/05/2020
https://azure.microsoft.com/es-es/services/cognitive-services/
[7] Material design. Desarrollo de interfaces. Acceso 08/05/2020
https://material.io/
[8] Android Developers. Criptografía. Acceso: 25/04/2020
https://developer.android.com/guide/topics/security/cryptography?hl=es-419
[9] MD5online. ¿Que es MD5 salt y por qué usarlo?. Acceso 25/04/2020
https://www.md5online.org/blog/md5-salt-hash/
[10] JQuery FOundation. JQuery. Acceso: 20/05/2020
https://jquery.com/
[11] Android Developers. Android Studio. Acceso: 10/04/2020
https://developer.android.com/studio
[12] CodigoFacilito. MVC. Acceso 15/04/2020
https://codigofacilito.com/articulos/mvc-model-view-controller-explicado
[13] Square. Picasso android. Acceso: 15/05/2020
https://square.github.io/picasso/
[14] 000webHost. Página oficial del hosting de la aplicación. Acceso: 09/04/2020
https://www.000webhost.com/
[15] Apple. iTunes API. Acceso 10/04/2020
https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/
[16] Deezer. Deezer API. Acceso: 10/04/2020
https://developers.deezer.com/api
[17] Spotify. Spotify API. Acceso: 10/04/2020 https://developer.spotify.com/documentation/web-api/
[18] LAST.fm. LAST.fm API. Acceso: 10/04/2020
https://www.last.fm/api/?lang=es