Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo de APPS móviles "WikiMusic"

Mobile APPS development "WikiMusic"

Daniel Fumero Cruz

La Laguna, *01* de *julio* de 2019

D. **Alejandro Pérez Nava**, con N.I.F. 43821179S 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. 42091420V 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 APPS móviles "WikiMusic"

ha sido realizada bajo su dirección por D. **Daniel Fumero Cruz**,

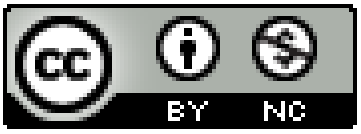
con N.I.F. 7864812B

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 21 de noviembre de 2018

Agradecimientos

A mis principales apoyos: mis padres, mi pareja y a mis amigos. Y como no a mis compañeros de estudios, a mis profesores y a toda mi familia. Quiero hacerles llegar mis más sinceras palabras de agradecimiento por haberme apoyado durante este viaje de formación y realización personal. ¡Muchas gracias!

Licencia



© Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial 4.0 Internacional.

Resumen

Se pretende realizar la implementación de un sistema basado en información utilizando la tecnología API REST para la extracción de datos de diferentes fuentes. En este caso, se decide integrar datos sobre artistas musicales, debiendo realizar múltiples implementaciones para facilitar su accesibilidad vía web y mediante una aplicación nativa en Android.

Palabras clave: API REST,WEB,Android...

Abstract

The intention is to implement an information-based system using API REST technology to extract data from different sources. In this case, it is decided to integrate data about musical artists, having to carry out multiple implementations to facilitate their accessibility via the web and through a native application on Android.

Keywords: API REST,WEB,Android...

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 Introducción

1.1 Justificación

A día de hoy, existen múltiples servicios API de distintas empresas dedicadas al ámbito musical. Pero la mayoría de ellas tienen ciertas carencias o están parcialmente incompletas al hacer referencia a ciertos artistas. Este proyecto busca unificar en una base de datos la mayor cantidad posible de información referente a los artistas musicales y su discografía a partir de estas API proveídas por diferentes medios aprovechando las ventajas de unas y otras supliendo así sus carencias y creando una rica base de datos del mundo de la música y sus diferentes artistas.

1.2 Alcance

Se pretende desarrollar mediante el estudio y uso de diferentes API una base de datos en MySQL que recopila diferentes datos sobre artistas musicales, para facilitar la gestión de esta base de datos se hará uso de la herramienta phpMyAdmin. Se desarrollará una página web haciendo uso de las últimas tecnologías como HTML 5,CSS 3, JQuery y el framework Bootstrap que nos facilitará el proceso de codificación. Esta web permitirá hacer consultas sobre los diferentes artistas musicales tanto así como la visualización de los datos, imágenes y algunos enlaces de interés como por ejemplo a sus redes sociales.

La implementa de la lógica de negocio de la aplicación se desarrollará haciendo uso de PHP que mediante las consultas a las diferentes API que devolverá en formato JSON toda la información, en caso de que esta no sea del todo completa se pasará a completar haciendo uso de otra de las API disponibles. De esta manera y tal como se ha especificado en el desarrollo de este proyecto, a partir de las consultas que vayan realizando los usuarios se irá rellenando nuestra base de datos para hacerse cada vez más amplia. Por lo que solo al principio, hará realmente uso de las API ya que una vez ese artista está insertado en la base de datos junto con toda su información pertinente, la aplicación pasará a mostrar los datos que ya tenemos almacenados ahorrando tiempo al no tener que realizar la consulta nuevamente.

El único inconveniente vendría a la hora de mantener actualizados los datos de la base de datos, para ello se ha solucionado mediante la creación de un script de refresco que permita cada cierto periodo de tiempo estipulado actualizar toda la información de los artistas. Así, de esta manera se consigue que la información más cambiante como

pueden ser los álbumes de los artistas estén siempre actualizados, sin tener que estar realizando consultas a la API continuamente, con lo problemas que ello conlleva.

Por último, se extrapola el desarrollo web a una implementación de una APP móvil en Android haciendo uso del lenguaje Java.

1.3 Objetivos

El estudio y desarrollo del trabajo se divide en los siguientes puntos:

- Aprendizaje y uso de múltiples API Rest para extraer información sobre los artistas y su carrera de diferentes proveedores tales como Spotify, Deezer, Itunes …
- Elaborar el diseño de una base de datos en la que almacenar la información, para evitar múltiples consultas en periodos cortos de tiempo, además de un sistema de refresco temporal para mantenerse actualizados.
- Diseño de una arquitectura software utilizando patrones de diseño adecuados para la gestión de las bases de datos, además de realizar una implementación cómoda para su posible extensión a otras plataformas.
- Desarrollo de una página web con la tecnología PHP, HTML 5, CSS 3, JavaScript y MySQL para realizar la interfaz entre la parte visual y la de gestión. Esta página permitirá la consulta y visualización de los datos biográficos.
- Tras el diseño software definido se pretende extrapolar el conocimiento utilizado en web para la implementación en móviles, se utilizará el entorno de desarrollo de Android Studio para su incorporación en el sistema Android.

1.4 Estado del arte

1.4.1 Antecedentes y estado actual

Actualmente existen algunas aplicaciones similares para la consulta de datos biográficos como pueden ser enciclopedias libres y otras fuentes tales como Wikipedia, Last FM, Itunes, etc. El objetivo de este proyecto es centralizar únicamente los datos de interés musical en una única aplicación, evitando la necesidad de realizar múltiples búsquedas por parte del usuario final, proveyendo esto una usabilidad añadida además del ahorro temporal de utilizar un conjunto de fuentes de información.

1.5 Metodología

1.5.1 Introducción

La metodología utilizada en este proyecto ha sido el modelo en cascada. Se trata de un modelo lineal con un proceso secuencial de diseño, a grandes rasgos podemos definirlo como una metodología de desarrollo que fluye desde un punto inicial hasta otro final, pasando por varias etapas claramente diferenciadas. Para este proyecto se definieron las siguientes etapas o fases.

Fases del modelo:

- **Fase 1** Análisis de requisitos del software
- **Fase 2** Diseño del sistema
- **Fase 3** Implementación
- **Fase 4** Verificación
- **Fase 5** Instalación y mantenimiento (Esta fase se obviara, puesto que este desarrollo es con meros fines académicos)

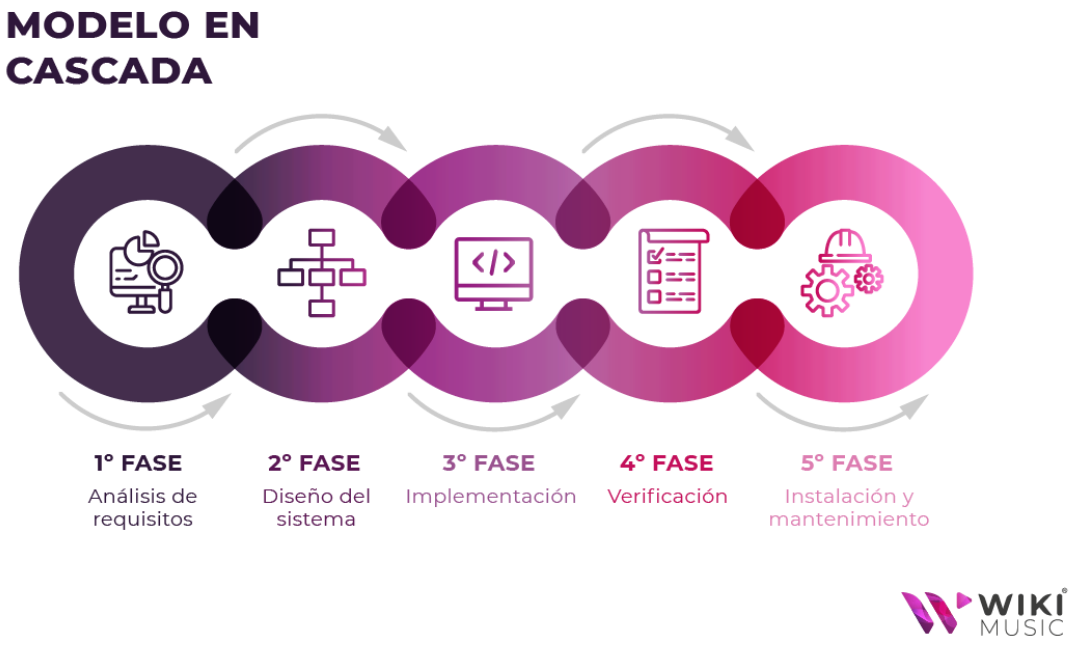


Figura 1: Modelo en cascada

Fase 1 Análisis de requisitos del software

En esta primera fase se analizan las necesidades del usuario o el cliente determinando así qué objetivos o características debe cubrir el software a desarrollar, especificando todo lo que debe hacer el sistema sin entrar en detalles técnicos. Esto permite estimar de manera rigurosa las necesidades del software antes de su diseño, costes, riesgos y plazos.

Cabe señalar que en esta fase hay que ser especialmente rigurosos, ya que este modelo no permite añadir nuevos requisitos a mitad del proceso de desarrollo.

Fase 2 Diseño del Sistema

En esta etapa se describe la estructura del software, descomponiendo y organizando los elementos que pueden elaborarse por separado con las ventajas que esto conlleva a la hora tanto del propio desarrollo, como el mantenimiento o escalabilidad del software. Además es interesante hacer una distinción entre el diseño de alto nivel o arquitectónico y el diseño detallado. Mientras que el primero tiene como objetivo definir la estructura de la solución mediante la identificación de grandes módulos y sus relaciones, definiendo así la arquitectura de la solución elegida. En el segundo se definen los algoritmos a emplear y la organización del código.

En resumen, se buscan las metodologías, herramientas y en general un análisis del software necesario para el cumplimiento de los requerimientos que se han especificado en la fase 1 de análisis de requisitos del software.

Fase 3 Implementación

En esta fase se realizan los algoritmos y se programan los requisitos anteriormente especificados haciendo uso de las estructuras de datos diseñadas en la anterior fase.

Fase 4 Verificación

Una vez terminada la fase de implementación se verifican que todos los componentes del sistema funcionen correctamente y que se cumplen los requisitos.

Fase 5 Instalación y mantenimiento

Cuando se ha finalizado de desarrollar todas las funcionalidades y se ha comprobado su funcionamiento, se inicia esta fase instalando la aplicaciones pertinentes y se comprueba su funcionamiento en el entorno en el que se utilizara. Por último se aplicará el mantenimiento del software, ya sea corrigiendo posibles errores, mejoras de rendimiento

o la adición de características.

1.5.2 Ventajas e Inconvenientes

Ventajas de este modelo:

- Modelo fácil de implementar y entender.
- Metodología de trabajo efectiva, ya que se define antes de diseñar y se diseña antes de codificar.
- Buen funcionamiento en equipos de desarrollo reducidos y productos maduros.
- Ideal para proyectos estables, con requisitos claros y no cambian a lo largo del proyecto.

Inconvenientes del modelo:

- Cualquier error en las primeras fases conlleva un rediseño completo del proyecto.
- No se puede avanzar de etapa hasta que se ha finalizado la anterior.
- No se obtiene software funcional hasta el final del ciclo de desarrollo.
- No sirve para proyectos muy complejos, donde no se conoce claramente su alcance o que puedan surgir circunstancias que puedan alterar su desarrollo.
- No se pueden prever o identificar cuellos de botella con tiempo suficiente para poder abordarlos.

1.5.3 ¿Por qué se optó por este modelo?

Se ha optado por este modelo, debido a las estimaciones claras y bien definidas de los requisitos del proyecto desde un principio permitiendo mediante este modelo hacer un seguimiento exhaustivo de cada etapa, una planificación clara y un ágil desarrollo del proyecto.

Capítulo 2 API REST

En este capítulo se introducirá sobre los comienzos y avances de la tecnología llamada API REST «Application Programming Interface - Representational State Transfer» o «Interfaz de Programación de Aplicaciones - Transferencia de Estado Representacional». Esta tecnología ha sido la base fundamental para el desarrollo de este proyecto permitiendo la elaboración del trabajo de una manera efi

2.1 Introducción

2.1.1 Historia

Las API (del inglés: «Application Programming Interface») comenzaron aparecer aproximadamente en la década de los años 60 y su principal cometido no fue en el uso de ordenadores personales, sino más bien como bibliotecas en S.O (Sistemas Operativos).

Más tarde durante los años 70, se dio el salto con la aparición de los sistemas distribuidos permitiendo el acceso remoto a las API, con el objetivo de no sobrecargar al programador proporcionando un sistema de empaquetado y desempaquetado de datos. Sin embargo, los avances más relevantes fueron a finales de los 80 y 90 cuando surge la Programación Orientada a Objetos (POO) que permitía la instancia de «objetos» y la aparición del lenguaje HTML potenciando en gran medida la funcionalidad de las API.

Todos estos avances propiciaron el nacimiento de una nueva tecnología llamada «REST» o «Representational State Transfer» aparece a partir del año 2000 y cuyo lanzamiento revolucionó el desarrollo del software. Por lo que fue una gran alternativa a otras especificaciones ya existentes en esos momentos como podían ser CORBA o SOAP, que eran complejas y casi imposibles de depurar. Esto fue debido a que no existía un estándar de cómo deberían de usarse las API, estando diseñadas para ser flexibles pero no accesibles.

El pionero de este nuevo estilo de arquitectura fue Roy Fielding [19][Figura 2] que es conocido por crear la especificación del HTTP, además de ser uno de los fundadores del servidor web Apache y un referente internacional en la Arquitectura de Redes. Él fue el que definió las bases de esta arquitectura (REST) describiendolo como un enfoque para

el desarrollo de servicios web, y todo esto se le ocurrió durante la elaboración de su tesis doctoral.

Figura 2: Roy T. Fielding

Esta tecnología de intercambio y manipulación de datos está hoy en día presente en casi todas las empresas o aplicaciones del mercado. Cabe mencionar algunas de ellas tales como: Twitter, YouTube, Facebook, Spotify…

2.1.2 ¿Qué es una API REST?

Una API REST se podría definir como un estándar para la manipulación de datos de una manera más eficiente y sencilla en sistemas que utilicen el protocolo HTTP y permitiendo la obtención de datos en formatos como XML y JSON.

Figura 3: Representación Rest:API

Hablaremos del término RESTFULL si se cumple la arquitectura REST, en otras palabras, que sigue una serie de reglas o características establecidas:

- **Peticiones sin estado**, es decir, que cada petición que recibe el servidor es independiente.
- **Llamadas cacheables** evitando así solicitar varias veces el mismo dato, por lo tanto, mejorando así la escalabilidad y rendimiento de la aplicación.
- **Interfaz uniforme**, cada petición deben estar siempre identificadas con una única dirección URI «Uniform Resource Identifier».
- **Separación del cliente con el servidor**, es decir, el servidor no se implica en los

datos del cliente y el propio cliente no necesita conocer los detalles de la configuración del propio servidor.

- **Sistema de capas,** la arquitectura está separada por capas que permiten que cada una lleve a cabo una propia funcionalidad.

Los verbos o operaciones HTTP más relevantes dentro de un sistema REST son **GET, PUT, POST** y **DELETE** para leer o consultar, actualización, creación y eliminación.

2.1.3 Formatos XML vs JSON

Las API REST permiten una variedad diversa de formatos de respuesta ofreciendo así a sus usuarios una mayor flexibilidad, aunque los dos formatos más utilizados son XML y JSON. Ya que estos formatos permiten un intercambio de datos compatible, lo que satisface una gran variedad de casos de uso como pueden ser soluciones multiplataforma o aplicaciones cruzadas. Por ello a continuación veremos las ventajas y desventajas de cada uno de estos formatos y así elegir uno de ellos a razón de nuestros requisitos funcionales [16].

	XML	JSON
Ventajas	- Formato estructurado y fácil de comprender - Permite comentar el código - Extensible - Fácil validación - Admite más formatos de codificación - Compatible con namespace	- Alta velocidad de procesamiento - Menor tamaño de los datos - Nativo para JavaScript - Fácil acceso a los datos
Desventajas	- Formato estricto - Requiere mayor procesamiento - Mayor tamaño de los datos - Sintaxis redundante - Un error puede invalidar el documento	- No admite comentarios en el código - No es extensible - Notación confusa - Estructura engorrosa a simple vista - No es compatible con namespace - Solo admite codificación UTF-8

Tabla 1: XML vs JSON

Como conclusión se puede apreciar claramente que para sistemas con bajo nivel de procesamiento como dispositivos móviles y con la eficiencia como objetivo el JSON es claramente mejor. Y es por ello por lo que se ha utilizado en el desarrollo de este proyecto.

Un ejemplo de una devolución a una petición API REST podría ser la siguiente:

Ejemplo de salida JSON.

```
{
   "alumnos": [
          {
            "id": “1”,
            "nombre": "Marisa",
            "apellido": "Perez",
            "correo": "marperez@gmail.com",
            "movil": "679134598"
          },
          {
            "id": “2”,
            "nombre": "Pedro",
            "apellido": "Hernandez",
            "correo": "perhernan@hotmail.com",
            "movil": "657941357"
          }
   ]
}
```

Ejemplo de salida XML.

```
<colegio>
        <alumno>
                <id>1</id>
                <nombre>Marisa</nombre>
                <apellido>Perez</apellido>
                <correo>marperez@gmail.com</correo>
                <movil>679134598</movil>
        </alumno>
        <alumno>
```

```
            <id>2</id>
            <nombre>Pedro</nombre>
            <apellido>Hernandez</apellido>
            <correo>perhernan@hotmail.com</correo>
            <movil>657941357</movil>
        </alumno>
</colegio>
```

2.1.4 Ventajas del API REST

Uno de los beneficios fundamentales es que el cliente de una API REST puede ser cualquier tipo de aplicación, navegador o servicio. Algunos ejemplos pueden ser aplicaciones tan dispares como Android o iOS, navegadores como Firefox o Chrome, o incluso servicios como Alexa.

Esto es debido gracias a la ya comentada separación entre el cliente y el servidor mejorando la portabilidad a otras plataformas, aumenta la escalabilidad de los proyectos. Además la API REST es independiente de cualquier plataforma o lenguaje por lo que se adapta perfectamente a cualquier desarrollo y requiere pocos recursos del servidor. Lo único que sí es indispensable es que las respuestas a las peticiones que se hagan en los formatos válidos como son normalmente XML o JSON.

2.1.5 Que diferencia una API de una API REST

La principal diferencia la encontramos en que REST es un tipo de API y se trata de un estilo arquitectónico que se encarga de gobernar el comportamiento de clientes y servidores en aplicaciones en red. Mientras que API en sí es un conjunto general de protocolos que se implementa sobre el software de aplicación ayudándolo a interactuar con otros componentes de software proporcionando una interfaz. Básicamente REST está orientado para a las aplicaciones web ocupándose de solicitudes y respuestas HTTP lo que lo hace utilizable por cualquier lenguaje de programación. En resumen no todas las API son REST, pero todos los servicios REST son API.

2.2 Estudio y elección de la API

Para el desarrollo de este proyecto, un criterio fundamental era la correcta elección de la API. Por lo que antes de comenzar con el proyecto se tuvo que hacer un estudio preliminar de cuáles son las alternativas que existen actualmente en el mercado y de ellas

cuales tenían las características más interesantes y necesarias para llevar a cabo el proyecto. Tras una exhaustiva búsqueda y recopilación de información se consideraron las siguientes como las más interesantes, ya que son gratuitas, fáciles de usar y están bien documentadas.

Figura 4: Deezer logo

DEEZER

Se trata de un sitio web de transmisión de música que permite descargar y escuchar canciones de forma gratuita. La API de Deezer [4] nos permite buscar canciones por álbum, género, artista…

Figura 5: Last.FM logo

LAST.FM

LastFM [2] tiene una de las bases de datos más grandes de datos de música que existen. Y poseen una potente API pudiendo obtener pistas de tendencias y artistas, listas de música, información de los artistas y mucho más de manera gratuita, aunque hay que registrarse en su web para poder hacer uso de ella y generar un API Key.

De esta API obtenemos muchos datos para la aplicación mediante diferentes consultas:

- **Devolver detalles del artista mediante el nombre del artista usando el método artist.getInfo.**
  method=artist.getinfo&artist= {Nombre del artista}
  Ejemplo:

http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=coldplay&lang=es&api_key=b198a6e7909cc0f874dd7a5209e0ce88&format=json

Last FM de manera peculiar devuelve los datos de forma jerárquica repitiendo campos dependiendo de la etiqueta por lo que para su facilidad de entendimiento se hará referencia de la siguiente manera.

artist->name: Contiene el nombre del artista.

artist->mbid: Se trata del número de identificación del usuario.

artist->url: Enlace de LastFM del perfil del artista.

artist->image: Enlace a la imagen del artista (inutilizado por LastFM por cuestiones de limitación de uso).

artist->similar: Contiene la información de los artistas relacionados.

artist->listeners: Número de oyentes del artista dentro de LastFM.

artist->playcount: Número de reproducciones del artista dentro de LastFM.

tags->name: Nombre de los géneros relacionados con el artista.

tags->url: Enlace dentro de LastFM a los géneros relacionados.

bio->published: Fecha de publicación de la biografía del artista.

bio->summary: Resumen de la biografía del artista.

bio->content: Contiene la biografía del artista.

- **Devolver todos los álbumes de un artista usando el nombre del artista usando el método artist.getTopAlbums.**

  method=artist.gettopalbums&artist= {Nombre del artista}

  Ejemplo:
  http://ws.audioscrobbler.com/2.0/?method=artist.gettopalbums&artist=coldplay&api_key=b198a6e7909cc0f874dd7a5209e0ce88&format=json

album->name: Nombre del álbum.

album->playcount: Número de reproducciones de ese álbum.

album->mbid: Se trata del número de identificación del usuario.

album->url: Enlace dentro de LastFM al álbum.

album->image: Enlace a la imagen del álbum (inutilizado por LastFM por cuestiones de limitación de uso).

Figura 6: Spotify logo

SPOTIFY

Es una de las bases de datos de las aplicaciones de transmisión de música más grandes del mundo. Además de las funcionalidades habituales y al igual que las otras API de datos de música mencionadas, Spotify [3] tiene puntos fuertes que le permiten buscar álbumes, artistas y pistas. Los datos devueltos incluyen metadatos y carátulas del álbum.

Como se puede apreciar más o menos las tres API son de grandes compañías relacionadas con el mundo de la música y ofrecen servicios similares. Sin embargo, una vez testadas cada una de ellas, se han observado los siguientes inconvenientes:

El principal problema de las API de Deezer y Spotify es que no permiten la búsqueda directa de un artista o álbum a través de su nombre u otra referencia, si no a partir de un campo ID (Identificador único) para cada uno de ellos que además cambia dependiendo de la API utilizada y en muchos casos incluso hay ID´s con campos vacíos (sin artistas asociados).

Por lo que para poder trabajar con ellas tendríamos que copiar todo el contenido de la API en nuestra base de datos para así poder trabajar directamente con los nombres de los artistas u otra información relevante, además de crear nuestra propia estructura de datos con la información que consideremos necesaria para este proyecto.

Ya que unos de los requisitos principales del proyecto es que los artistas se vayan almacenando en la base de datos según los usuarios vayan realizando las consultas y creando así nuestra propia estructura de datos, pero de esta manera no se podría llevar a cabo con estas API.

Otro inconveniente es que las API son de reproducción de música y se pueden obtener información sobre álbumes y demás, pero no así una biografía completa del artista que es otro requisito esencial en el proyecto. Por lo que queda la API de LAST.FM que sí que nos facilita la información necesaria tanto álbumes como biografía necesarias para el proyecto. Pero con un inconveniente y es que las imágenes han pasado a desaparecer devolviendo siempre la misma imagen estándar, esto se debe a que limitaron su uso en

los últimos cambios que le realizaron a la API. Pero aún así es una de las más interesantes y completas para este proyecto y el tema de las imágenes puede no llegar a ser un criterio tan esencial ya que, en el proyecto se busca intercalar la información de distintas API complementando las carencias de unas con otras. Y pasamos a mencionar la última de las API de código abierto llamada theaudiodb.

Figura 7: The Audio DB logo

THEAUDIODB

The Audio DB [1] es código abierto y se trata de una base de datos comunitaria de ilustraciones y datos de audio con una API que devuelve los datos en un formato JSON y nos permite todo lo mencionado anteriormente y sin necesidad de registrarnos. Y el único inconveniente encontrado es que al ser comunitaria no es de las más completas y pueden no existir ciertos datos de artistas.

De esta API obtenemos muchos datos para la aplicación mediante diferentes consultas:

- **Devolver detalles del artista mediante el nombre del artista.**
  search.php? S = {Nombre del artista}
  Ejemplo: theaudiodb.com/api/v1/json/1/search.php?s=coldplay

  idArtist: Id del artista dentro de la base de datos de Audio DB.
  strArtist: Contiene el nombre del artista.
  intBornYear: Año de nacimiento de un artista.
  intDiedYear: Año de fallecimiento de un artista.
  strStyle: Estilo musical del artista.
  strGenre: Género musical del artista.
  strWebsite: Dirección oficial de la página web del artista.
  strFacebook: Dirección oficial del Facebook del artista.
  strTwitter: Dirección oficial del Twitter del artista.
  strBiographyEN: Biografía del artista en inglés.
  strBiographyES: Biografía del artista en español.

strGender: Género del artista, es decir, si es hombre o mujer.
intMembers: Número de miembros del grupo.
strCountry: País de nacimiento del artista.
strArtistThumb: Enlace a una imagen del artista.

- **Devolver todos los álbumes de un artista usando el nombre del artista.**
  searchalbum.php? S = {Nombre del artista}
  Ejemplo: theaudiodb.com/api/v1/json/1/searchalbum.php?s=coldplay

  idAlbum: Id del álbum dentro de la base de datos de Audio DB.
  idArtist: Id del artista dentro de la base de datos de Audio DB.
  strAlbum: Nombre del álbum.
  strArtist: Nombre del artista del álbum.
  intYearReleased: Año de lanzamiento del álbum.
  strStyle: Stylo musical del álbum.
  strGenre: Género musical del álbum.
  strAlbumThumb: Enlace que contiene la imagen de la carátula del álbum.
  strDescriptionEN: Descripción del álbum en inglés.
  strDescriptionES: Descripción del álbum en español.

Para el proyecto se han seleccionado los siguientes datos de las distintas API utilizadas:

- **Last.Fm** de esta API en concreto se han utilizado los siguientes campos:
  - artist->name
  - bio->content
  - artist->image
  - album->name
  - album->image

- **TheAudioDB** y de esta API se han utilizado los siguientes campos:
  - strArtist
  - intBornYear
  - intDiedYear
  - strStyle
  - strGenre
  - strWebsite
  - strFacebook

- strTwitter
- strBiographyES
- strCountry
- strArtistThumb
- strAlbum
- intYearReleased
- strAlbumThumb

Estos campos se han mezclado para elaborar una base de datos propia y adecuada para este proyecto supliendo las carencias encontradas en cada una de ellas.

Capítulo 3 Desarrollo Web

En este capítulo se detallan todos detalles tanto de implementación como de desarrollo realizados en la aplicación web, así como las fases, herramientas y tecnologías utilizadas en esta parte del proyecto, terminando con una revisión de las distintas vistas del aplicativo web.

3.1 Herramientas

Para el desarrollo de este proyecto se ha utilizado como apoyo las siguientes herramientas:

Figura 8: Sublime Text logo

- **Sublime Text:** Es un editor de texto y de código fuente multiplataforma, que fue desarrollado inicialmente como una extensión de Vim. Soporta una gran variedad de lenguajes y destaca principalmente por ser ligero.

Figura 9: XAMPP logo

- **XAMPP**: Se trata de un software libre que te permite instalar de manera sencilla un servidor apache en tu ordenador de manera local. Incluye además un sistema de base de datos MySQL, un gestor de base de datos phpMyAdmin, intérprete de

PHP, servidor FTP Filezilla… En general trae todo lo necesario para levantar un servidor completamente operativo. Esta herramienta nos permite probar y desarrollar nuestro trabajo de manera local y nos abstrae de las engorrosas configuraciones de un servidor permitiéndonos centrarnos en el desarrollo del proyecto, además de no depender de acceso a internet.

Figura 10: Github logo

- **Github:** Se trata de un servicio de control de versiones Git, nos permite tener en la nube nuestro proyecto y controlar los cambios realizados en el proyecto. Así tendremos la posibilidad de tener un buen control sobre el código, de las versiones y poder volver atrás en caso de error o poder tener una versión general de nuestro proyecto.

Figura 11: Bootstrap logo

- **Bootstrap**: Es una herramienta multiplataforma y de código abierto para el diseño rápido de sitios y aplicaciones web desarrollado por Twitter [8]. Esta contiene plantillas de diseño con diferentes elementos basados en HTML, CSS y extensiones JavaScript adicionales. Este framework solo se ocupa del desarrollo del frontend y entre sus grandes ventajas, tenemos la adaptabilidad que permite ajustarse a cualquier resolución y dispositivo. Evitando así las complejidades de adaptar nuestro proyecto con un diseño responsive o la compatibilidad entre diferentes navegadores evitando entrar en las características de diseño propias de cada uno de ellos.

Figura 12: JQuery logo

- **JQuery:** Se trata de una librería de JavaScript de código abierto [10]. Esta herramienta simplifica la tarea de programar en JavaScript y nos permite simplificar la codificación en javascript comprimiendo varias líneas de código mediante el uso de sus diferentes funciones, de esta forma no es necesario escribir bloques enteros de código, por lo que necesitamos menos tiempo, menos código y por lo tanto los errores serán menos frecuentes en nuestro proyecto. Esta librería nos permite manipular elementos del DOM, cambios del diseño CSS y realizar peticiones AJAX entre otras cosas.

Figura 13: PhpMyAdmin logo

- **phpMyAdmin**: Herramienta gratuita escrita en PHP, que permite el manejo y la administración de bases de datos MySQL a través de una interfaz sencilla mediante un portal web, que admite gran variedad de operaciones en MySQL y MariaDB.

Figura 14: Twitter typeahead

- **Typeahead:** Se trata de una biblioteca desarrollada en javascript, que permite implementar en nuestros sistemas el autocompletado o sugerencias rápidas en una barra de búsqueda [11]. Mejorando la experiencia de usuario, además de permitir una escritura anticipada el usuario ahorra tiempo y reducimos la cantidad de errores posibles, ya que el usuario tiene menos probabilidades de cometer un error

ortográfico.

Figura 15: Adobe Illustrator logo

- **Adobe Illustrator:** Es un software de pago y de edición gráfica desarrollado por Adobe Systems. Éste está dedicado al dibujo vectorial y al diseño de elementos gráficos, utilizándose tanto para la maquetación web, gráficos para móviles, interfaces web, diseño editorial, diseño de marca, realización de infografías...

  Es una herramienta versátil que permite producir rápidamente gráficos flexibles, es decir editables y sin perder calidad. Para este proyecto se ha hecho uso de la versión de prueba gratuita de 7 días.

Figura 16: Adobe Photoshop logo

- **Adobe Photoshop:** es un software de pago desarrollado por la misma empresa que el anterior software mencionado, pero con la diferencia de que es un editor dedicado exclusivamente al retoque de fotografías e incluso de gráficos. Dicho programa sirve para crear, modificar, editar o retocar cualquier imagen sin perder la calidad de la misma. Es muy utilizado por cualquier empresa dedicada al sector de publicidad, diseño o relacionadas. Para este proyecto se ha hecho uso de la versión de prueba gratuita de 7 días.

3.2 Tecnologías

3.2.1 Tecnologías de cliente

Son aquellas que permiten establecer comunicaciones con el servidor y crear interfaces de usuario, para este proyecto se han utilizado las siguientes tecnologías:

Figura 17: HTML 5 logo

- **HTML 5:** Se trata de la quinta revisión del lenguaje de marca HTML, un estándar diseñado para la elaboración de páginas web que permite definir una estructura básica y su contenido mediante etiquetas [13]. Este lenguaje permite gran adaptabilidad, una estructura lógica y facilidad de interpretación tanto por máquinas como por humanos.

Figura 18: CSS 3 logo

- **CSS 3:** Lenguaje de diseño gráfico que permite definir y crear la presentación de un documento estructurado mediante un lenguaje de marca [12]. La separación de lenguaje HTML del CSS busca mejorar la accesibilidad al documento, proveer mayor flexibilidad y un control en la especificación de características presentacionales. Además de reducir complejidad y repetición de código en su estructura.

Figura 19: JavaScript logo

- **JavaScript:** Lenguaje de programación interpretado, que permite la creación de páginas webs dinámicas. Este lenguaje está principalmente enfocado en el frontend,agregando mayor interactividad y experiencia de usuario a la página web.

Aunque también puede usarse en el backend por ejemplo mediante el framework Node.js. Entre sus ventajas y características cabe destacar su ligereza, su modelo multiplataforma, es imperativo, estructurado, prototipado y orientado a objetos.

3.2.2 Tecnologías de servidor

Son aquellas que nos permiten implementar comportamientos de la aplicación web en el servidor, para este proyecto se han utilizado las siguientes tecnologías:

Figura 20: MySQL logo

- **MySQL:** Es un sistema de bases de datos relacional de código abierto aunque cuenta también con una versión comercial y que sigue el modelo cliente-servidor. Las ventajas de este sistema de gestión se encuentran algunas como que se ejecuta en cualquier plataforma, es fiable, fácil de usar, está estandarizado dentro de la industria del software, es seguro y posee un gran rendimiento.

Figura 21: PHP logo

- **PHP:** Lenguaje de código abierto, con las siguientes características es multiparadigma, imperativo, funcional, procedural y orientado a objetos [9]. Permite ser incrustado directamente en código HTML y es procesado en un servidor web por un intérprete. El cliente recibe los resultados después de que el servidor interprete el código PHP, entre sus múltiples posibilidades están algunas tales como la de enviar y recibir cookies,crear imágenes a partir de datos, o presentar resultados en otros estándares de datos o lenguajes como XML. También permite generar archivos y almacenarlos en el sistema de archivos en vez de presentarlos, generando así contenido dinámico pudiendo surgir de otros sitios, además de la propia base de datos de nuestro sistema. Por último, ese lenguaje puede enlazarse con otros como puede ser Java o interactuar con otros servidores usando cualquier protocolo.

3.3 Desarrollo

Una vez estudiadas y encontras las API a utilizar se procedió al desarrollo de la aplicación web, que permitirá la correcta búsqueda y visualización de los datos así como la lógica de negocio de la aplicación y las API. Esta implementación se ha realizado haciendo uso de un servidor local con Xampp que contiene tanto el código de la página web como la base de datos que se ha desarrollado, además se ha utilizado el sistema Github para el control de versiones y copia de seguridad del proyecto.

El proyecto hace uso del patrón Modelo Vista Controlador (MVC). Este patrón de arquitectura de software separa los datos de la aplicación, la interfaz de usuario y la lógica de negocio en tres componentes distintos que se desarrollará a continuación.

- **Modelo:** contiene una representación de los datos que maneja la aplicación y su lógica de negocio.
- **Vista**: es la interfaz de usuario y los mecanismos de interacción de este.
- **Controlador**: es el intermediario entre el Modelo y la Vista.

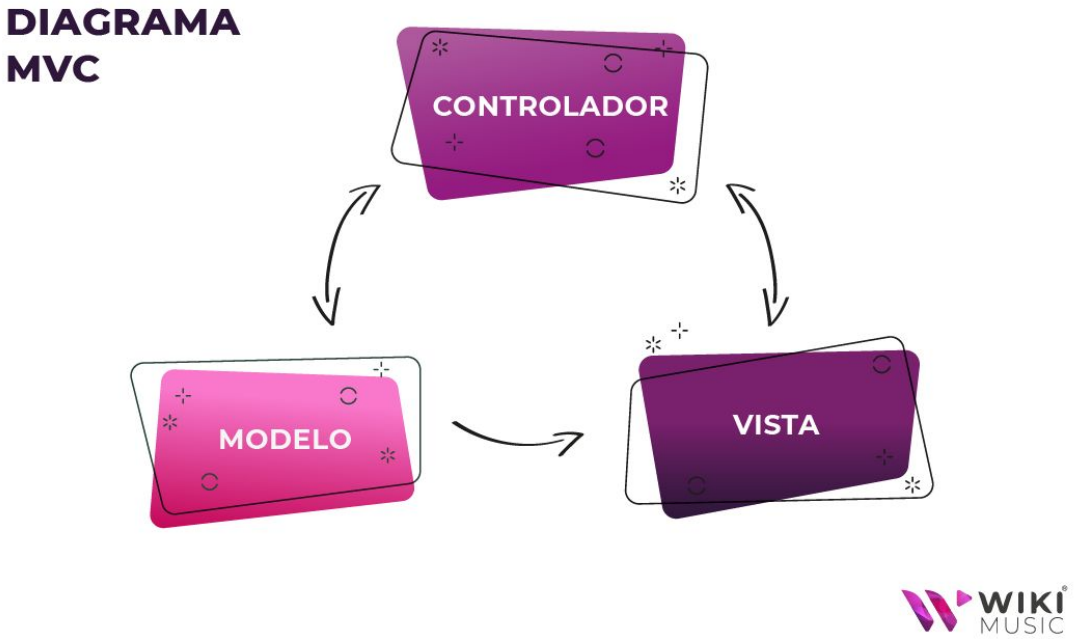


Figura 22: Patrón MVC

Para la Vista se hizo uso del framework Bootstrap que permite una rápida codificación inicial haciendo uso de los lenguajes HTML5, CSS3 y Javascript utilizando el framework JQuery, todo esto codificado mediante el uso del software gratuito Sublime Text. Como diseño para la web se optó por un diseño minimalista utilizando los colores de referencia del logo que se diseñó para esta aplicación.

Figura 23: Logo de aplicación desarrollada “Wikimusic”

Estos colores se eligieron haciendo referencia al logo de la ULL con unas tonalidades similares. La web consta de dos vistas, el Index o página principal que permite realizar búsquedas de los artistas y un TOP de artistas con el número de visitas realizadas y otra vista con el resultado de la búsqueda que muestra la información del artista como es su biografía, discografía y enlaces de interés a sus redes sociales en caso de que las hubiera.

El Modelo contiene la lógica de negocio de la aplicación, para ello se ha codificado haciendo uso del lenguaje PHP. Se ha considerado separar en diferentes ficheros por temas de escalabilidad y flexibilidad del proyecto en líneas futuras, de manera que por un lado están los accesos y modificaciones de la base de datos, y por otro las conexiones a las API y el tratamientos de los datos devueltos en formato JSON para su posterior inserción dentro de la base de datos.

Para la base de datos se ha usado MySQL y su gestión se ha realizado mediante la herramienta phpMyAdmin que facilita la administración, realización de la estructura y las pruebas iniciales de inserción y manipulación de datos obtenidos de la API.

El controlador se encargará de gestionar el cambio de vistas y el tráfico de datos entre las dos capas Modelo y Vista .

Por último, se ha realizado el script de actualización de los datos también haciendo uso del lenguaje PHP. Básicamente cada cierto periodo de tiempo estipulado y configurable desde el servidor que usemos para el alojamiento se ejecutara y se encargará de conectar con nuestra base de datos y traerse todos los artistas contenidos para realizar

nuevamente las peticiones a las API refrescando los datos a su versión más actualizada tanto de biografía, como discografía siendo esta última la más susceptible a cambios.

Este método presenta varios problemas debido a una serie de inconvenientes. La más evidente sería que si no se ejecutase ese script en cierto periodo de tiempo los datos no estarían actualizados por lo que no serían fiables y la aplicación acabaría en desuso. Otra es que la única forma de actualizarlo es traer todos los datos que tenemos y sobreescribirlos, debido a que es la forma más sencilla de hacerlo puesto que tendríamos que contrastarlos con los de las API en busca de cambios y esto es un proceso lento y complejo además el uso de distintas API lo hace aún más complicado, por lo que se a optado por sobreescribirlos. En definitiva es un proceso lento y exponencial ya que a más completa la base de datos más se tardará en actualizar pero es el mejor modo encontrado hasta el momento.

3.4 Diseño e Interfaz

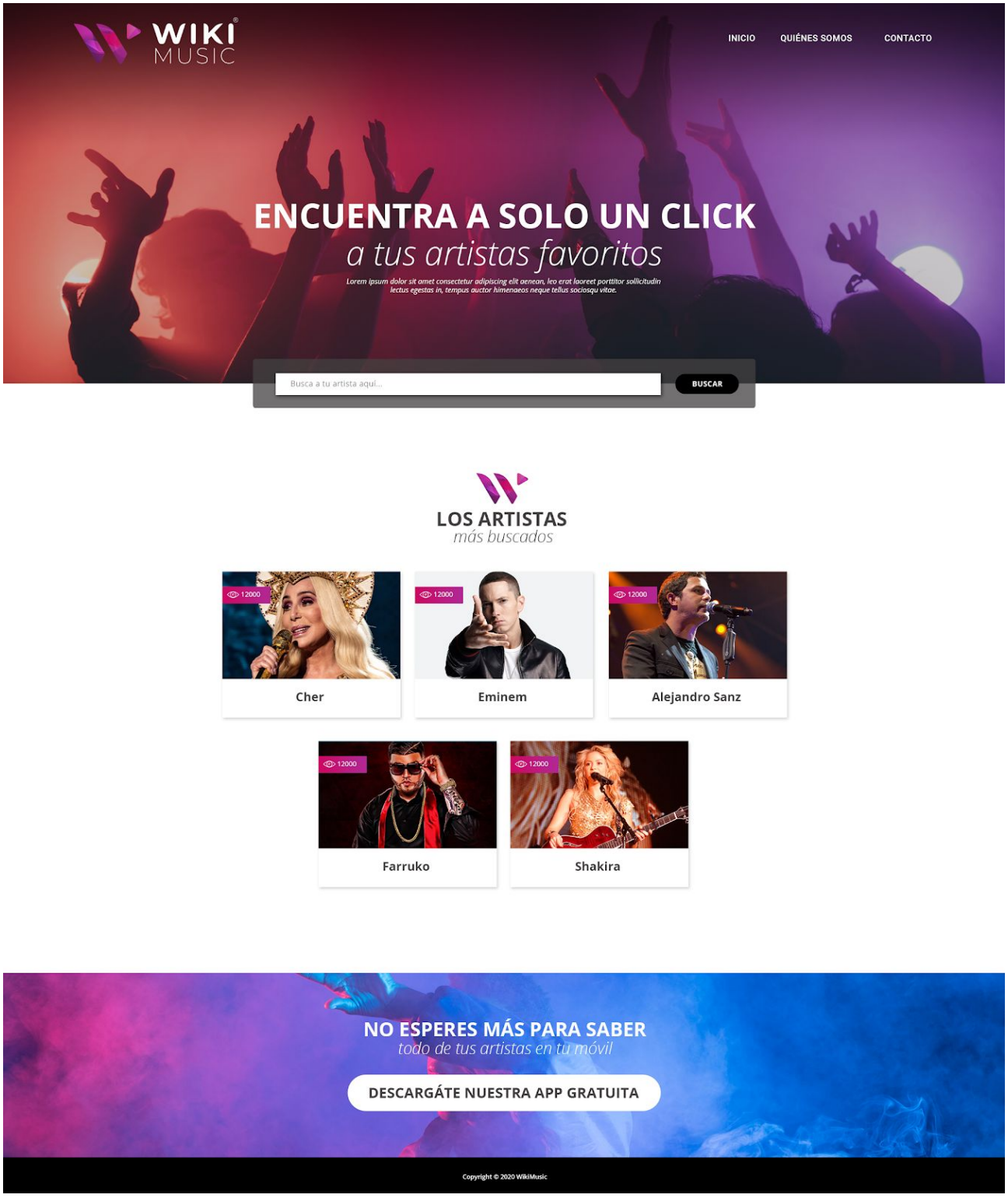


Figura 24: Vista Index

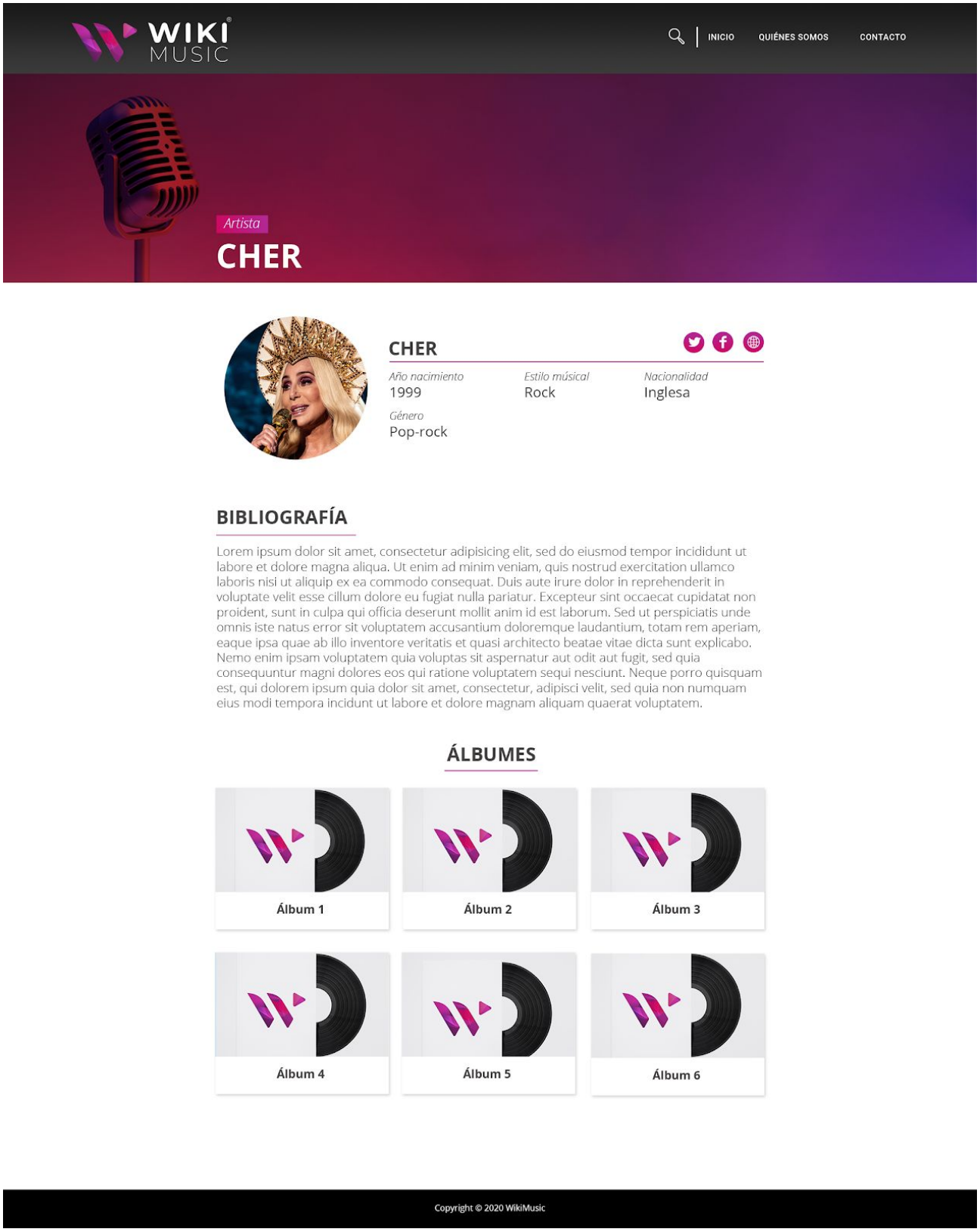


Figura 25: Vista resultado de búsqueda

Capítulo 4 APP Móvil

En este capítulo se detallan todos detalles tanto de implementación como de desarrollo realizados en la aplicación móvil, así como las fases, herramientas y tecnologías utilizadas en esta parte del proyecto, terminando con una revisión de las distintas vistas del modelo móvil en Android.

4.1 Herramientas

Figura 26: Android Studio logo

- **Android Studio:** Entorno de desarrollo integrado para la plataforma Android, está basado en IntelliJ IDEA de JetBrains y está publicado de manera gratuita [6]. Esta herramienta nos permite entre otras cosas, algunas tales como la previsualización de la aplicación desarrollada en diferentes smartphones y tablets viendo su funcionamiento y su diseño permitiéndonos hacer un seguimiento de nuestro trabajo de una manera más cercana al desarrollo final y sus posibles implementaciones. Otras como un sistema de compilación basado en Gradle, integración con GitHub, un entorno unificado entre otras funciones.

4.2 Tecnologías

Figura 27: Java logo

- **Java:** Lenguaje de programación desarrollado inicialmente por Sun Microsystems aunque actualmente se encuentra en propiedad de Oracle tras ser adquirido por este. Es un lenguaje orientado a objetos con un paradigma imperativo con independencia de plataforma permitiendo que los desarrolladores escriban el programa una sola vez y ejecutarlo en cualquier dispositivo, además dicho lenguaje es nativo para el desarrollo de aplicaciones Android.

  Algunas de sus características principales son:
  - Ofrece la funcionalidad de un lenguaje potente, derivado de C y C++ pero con un enfoque más sencillo.
  - Tiene una gran biblioteca estándar
  - Independencia de plataforma o arquitectura, gracias a la máquina virtual de Java que crea un puente entre la aplicación y el hardware del dispositivo.
  - Es multihilo.
  - Es seguro y sólido, ya que proporciona mecanismos para la administración automática de la memoria como el recolector de basura y provee canales de comunicación seguros protegiendo la privacidad de los datos.

4.3 Desarrollo

Tras finalizar el desarrollo de la mayor parte de este proyecto, se pasa a extrapolar la implementación web a una aplicación Android haciendo uso del lenguaje Java. Lo primero que hay que conseguir, es conectar la aplicación a nuestro servidor local Xampp. Permitiendo poder hacer uso de los mismos ficheros PHP anteriormente desarrollados para la aplicación web que permiten la conexión y obtención de los datos entre la API y la base de datos. Para ello necesitamos importar cierta librerías para el trato de los datos en formato JSON y poder trabajar con ellos de forma cómoda ya sea como Objetos o Arrays de datos además de permitir manejar las excepciones que pudieran producirse.

Una curiosidad de Android en el apartado de conexión a nuestro servidor Xampp en local, es que no viene a ser la dirección tipica de acceso de localhost como es 127.0.0.1 en el puerto 8080 sino que en Android tenemos que ir la la direccion local 10.0.2.2 en el mismo puerto 8080. De esta manera ya podremos acceder al servidor Xampp y trabajar con él y con el contenido de nuestro proyecto en esta caso la conexión a los ficheros PHP con las diferentes implementaciones y consultas con las API y a nuestra base de datos.

Una vez obtenido los datos y consultas necesarias en la aplicación se procede a cargar mediante el sistema de vistas de Android. Para su correcta visualización se desarrollan listas dinámicas que permitan la visualización de los artistas y álbumes por lo que es necesario en Android crear los Adapters y Layout oportunos para añadirlos a nuestras Activities, ya que por defecto actualmente no existen nativamente. Otro de los problemas

encontrados en Android es la imposibilidad de cargar desde una URL una imagen y no tener que descargarlas y almacenarlas en el sistema con los problemas de almacenamiento de memoria que ello acarrearía, para subsanar este problema se ha hecho uso de un plugin llamado Picasso una herramienta open source que facilita mucho el trabajo de manipulación de imágenes en pocas líneas de código, permitiéndonos la carga directa desde URL con la ventaja de almacenar las imágenes en caché asi como manipulacion y transformacion de imágenes complejas con un uso mínimo de memoria.

Finalmente se realizan las pruebas pertinentes y se finaliza el proyecto, a continuación se muestran las diferentes vistas de la aplicación Android.

4.4 Diseño e Interfaz

Figura 28: Vista splash APP

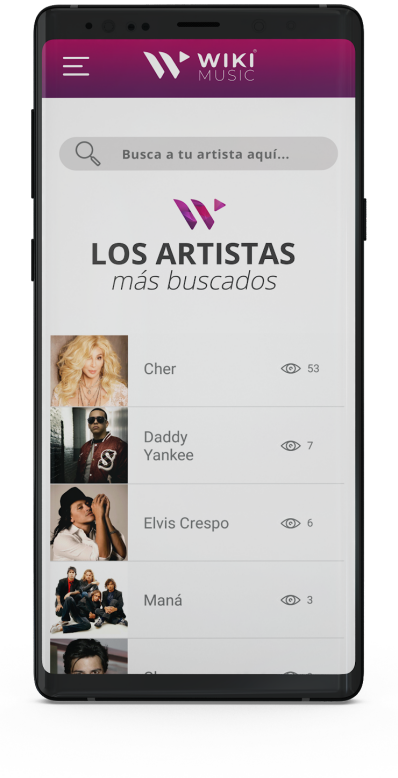


Figura 29: Vista principal

Figura 30: Vista menú lateral

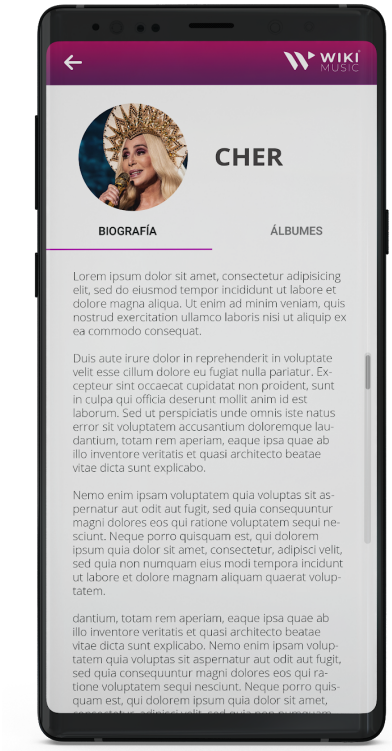


Figura 31: Vista información del artista

Capítulo 5 Conclusiones y líneas futuras

En definitiva, estamos ante un proyecto muy interesante que busca aunar y centralizar en un solo lugar la mayor cantidad de datos posibles acerca del mundo musical y sus artistas creando una gran base de datos propia. Pudiendo convertirse en el principal centro de información especializado y ser un referente de información en el mundo de la música.

Principalmente como requisito indispensable en el avance de este proyecto, sería el desarrollo de la APP para el sistema iOS incrementar así el número de usuarios y el alcance del proyecto.

Entre posibles mejoras del proyecto podrían ser, crear una API propia que devuelva nuestros datos almacenados de manera que se pueda extrapolar a otros proyectos y propiciar así su crecimiento. Añadir y enriquecer el contenido obtenido de las API como pueden ser mediante videos, lista de principales singles, permitir reproducir las canciones o previsualizaciones, etc. Crear diferentes perfiles de búsqueda y no limitarlo a sólo por su nombre, relacionarlos entre sí por género musical o similares y proporcionar sugerencias. Desarrollar un sistema modular implementando por ejemplo un patrón estrategia que permita cambiar de API de manera más sencilla, manipularlas o añadir nuevas funcionalidades.

Y por último, otro factor a optimizar sería el sistema de obtención de los datos mediante las diferentes API mejorando la velocidad y la forma en la que se intercalan las API para suplir sus carencias por falta de información.

Capítulo 6 Summary and Conclusions

In short, we are dealing with a very interesting project that seeks to unite and centralize in one place as much data as possible about the musical world and its artists, creating a large database of its own. Being able to become the main specialized information center and be a benchmark for information in the world of music.

Mainly as an indispensable requirement in the progress of this project, it would be the development of the APP for the iOS system, thus increasing the number of users and the scope of the project.

Possible improvements to the project could include creating an API of its own that returns our stored data so that it can be extrapolated to other projects and thus promote their growth. Add and enrich the content obtained from the APIs such as through videos, list of main singles, allowing songs to be played or previews, etc. Create different search profiles and do not limit it to just by name, relate them to each other by musical genre or the like and provide suggestions. Develop a modular system implementing, for example, a strategy pattern that makes it easier to change APIs, manipulate them or add new functionalities.

And finally, another factor to optimize would be the system of obtaining the data through the different APIs, improving the speed and the way in which the APIs are intercalated to fill their deficiencies due to lack of information.

Capítulo 7 Presupuesto

En este capítulo se abordarán de la forma más detallada posible y de manera aproximada los gastos del desarrollo e implementación del proyecto.

7.1 Gastos fijos de empresa

Gastos fijos del mantenimiento de la empresa y los servicios básicos aproximados durante el periodo de duración del proyecto.

Concepto	Meses	Cuota	Total
Alquiler del local	4	550€	2.200€
Luz	4	45€	180€
Agua	4	30€	120€
Internet	4	59€	236€

Tabla 2: Resumen de gastos fijos

7.2 Gastos en servicios y software

Gastos de contratación de servicios y software aproximados durante el periodo de duración del proyecto.

Concepto	Meses	Cuota	Total
Hosting	4	14€	56€
Dominio	1 año	15€	15€
GitHub enterprise	4	21€	84€

Tabla 3: Resumen de servicios y software

7.3 Gastos salariales

Gastos derivados de la contratación de personal cualificado aproximado durante el periodo de duración del proyecto.

Categoría	Horas	Euros/Hora	Total
Jefe de desarrollo	320	52€	16.640€
Ingeniero informático	640	42€	26.880€
Tecnico superior informatico	640	25€	16.000€
Tecnico superior informatico	640	25€	16.000€
Diseñador gráfico	320	32€	10.240€

Tabla 4: Resumen de salarios

7.4 Gastos totales

Gastos totales de la realización del proyecto con una duración aproximada de 4 meses.

Concepto	Total
Gastos fijos de empresa	2.736€
Gastos en servicios y software	155€
Gastos salariales	85.760€
Total	88.651€

Tabla 5: Resumen gastos

Capítulo 8 Anexo

8.1 Diagrama de casos de uso

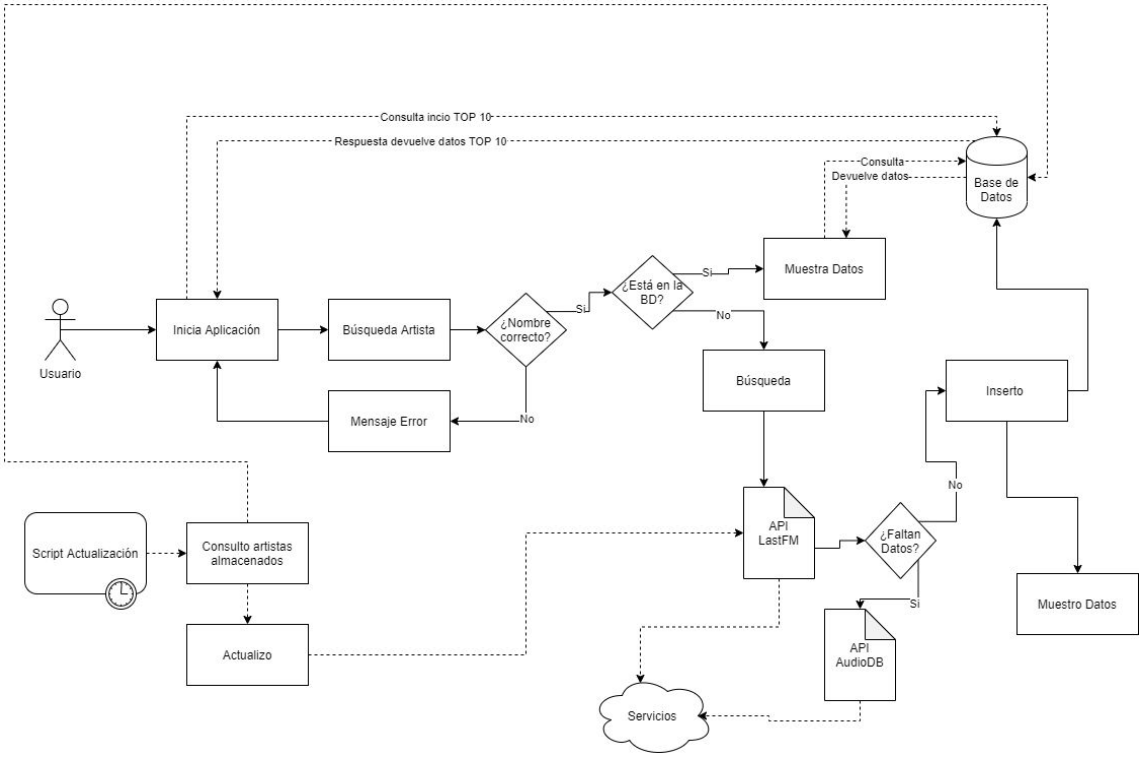


Figura 32: Diagrama caso de uso

8.2 Fragmento de código de búsqueda de un artista

```
/*****************************************************************************
*
* Fichero: bdApi.php
*
*****************************************************************************
*
* AUTOR: Daniel Fumero Cruz
*
* FECHA: 20/05/2020
*
```

```
* DESCRIPCIÓN: Busqueda de informacion de un artista
*
*********************************************************************************/

//Si el nombre del artista no está vacío comienzo con su búsqueda
 if ( isset($_POST['inpt_search']) && trim($_POST['inpt_search']) != '' ){

    $artist = mysql_real_escape_string($_POST['inpt_search']);
    unset($_POST['inpt_search']);

    //Realizamos una consulta para saber si ya está en nuestra base de datos
    $consulta = "SELECT id FROM artist WHERE name LIKE '".$artist."%'";
    $res = mysql_query($consulta,$conexion);

            //Si no está en la base de datos
    if (mysql_num_rows($res) <= 0) {

     //Realizamos la búsqueda en la primera API
      $json =
file_get_contents("https://www.theaudiodb.com/api/v1/json/1/search.php?s=".urlencode($a
rtist));
      $data = json_decode($json,true);

      //Si no es correcto el nombre del artista no lo inserto en la base de datos
      if ($data["artists"] != '') {

        // Si el Nombre del artista es correcto continuó obteniendo sus datos
        $name = mysql_real_escape_string($data["artists"][0]["strArtist"]);
        $style = mysql_real_escape_string($data["artists"][0]["strStyle"]);
        $genre = mysql_real_escape_string($data["artists"][0]["strGenre"]);
        $country = mysql_real_escape_string($data["artists"][0]["strCountry"]);
        $biography = mysql_real_escape_string($data["artists"][0]["strBiographyES"]);

        //Si no hay biografía del artista en la primera API buscamos en la otra API
        if ($biography == '') {

           $json2 =
file_get_contents("http://ws.audioscrobbler.com/2.0/?method=artist.getinfo&artist=".urlenc
ode($artist)."&lang=es&api_key=b198a6e7909cc0f874dd7a5209e0ce88&format=json");
           $data2 = json_decode($json2,true);
           $aux = $data2["artist"]["bio"]["content"];
           $biography = mysql_real_escape_string(substr_replace($aux," ",strpos(
$aux,"<a"),strlen("$aux")));
```

```
        }

        $url_web = $data["artists"][0]["strWebsite"];
        $url_facebook = $data["artists"][0]["strFacebook"];
        $url_twitter = $data["artists"][0]["strTwitter"];
        $image = $data["artists"][0]["strArtistThumb"];

        //Obtenidos todos los datos los meto en la base de datos
        $sql = "INSERT INTO artist
(name,style,genre,country,biography,url_web,url_facebook,url_twitter,image,visits)
VALUES
('$name','$style','$genre','$country','$biography','$url_web','$url_facebook','$url_twitter','$i
mage',0)";

        mysql_query($sql,$conexion) or die ("<center>ERROR al añadir artista</center>");

        //Se realiza la búsqueda de los álbumes de ese artista en la API
        $json =
file_get_contents("https://theaudiodb.com/api/v1/json/1/searchalbum.php?s=".urlencode($
artist));
        $data = json_decode($json,true);

        //Se obtienen los datos
        foreach($data["album"] as $values)
        {

         $title =  mysql_real_escape_string($values["strAlbum"]);
         $year = $values["intYearReleased"];
         $img =  $values["strAlbumThumb"];
                //echo $title. " ".$year." ".$img." ";

          //Se insertan los datos en la base de datos
         $sql = "INSERT INTO album (id_artist,title,year,image) VALUES
('$id','$title','$year','$img')";
         mysql_query($sql,$conexion) or die ("<center>ERROR al añadir
álbumes</center>");

       }

      //Si el nombre del  artista es incorrecto
     }else{
```

```
    //Capturó el flag  para mostrar un error
    $error = true;

   }

 //Si el nombre del artista ya está en la base de datos
 }else{

   //Se obtiene su ID para mostrarlo posteriormente
   $row = mysql_fetch_assoc($res);
   $id = $row['id'];

 }

//-----------------------------------------------------------------------------------------------------------------
```

8.3 Fragmento del script de actualización

```
/*****************************************************************************
*
* Fichero: update.php
*
*****************************************************************************
*
* AUTOR: Daniel Fumero Cruz
*
* FECHA: 24/05/2020
*
* DESCRIPCIÓN: Script actualización de datos de artistas BD
*
*****************************************************************************/

$consulta = "SELECT id,name FROM artist ";
$resultado = mysql_query($consulta,$conexion);

//Obtengo todos los artistas de la base de datos
 while ($fila = mysql_fetch_array($resultado, MYSQL_NUM)) {

//Para cada uno de ellos se consulta en la API
                    $json =
file_get_contents("https://www.theaudiodb.com/api/v1/json/1/search.php?s=".urlencode($fi
la[1]));
```

```
        $data = json_decode($json,true);

      //Se obtienen los datos
      $style = mysql_real_escape_string($data["artists"][0]["strStyle"]);
      $genre = mysql_real_escape_string($data["artists"][0]["strGenre"]);
      $country = mysql_real_escape_string($data["artists"][0]["strCountry"]);
      $biography = mysql_real_escape_string($data["artists"][0]["strBiographyES"]);
      $url_web = $data["artists"][0]["strWebsite"];
      $url_facebook = $data["artists"][0]["strFacebook"];
      $url_twitter = $data["artists"][0]["strTwitter"];
      $image = $data["artists"][0]["strArtistThumb"];

//Se realiza la actualización de los datos de la base de datos por los obtenidos en la API
      $sql = "UPDATE artist SET biography='$biography', style='$style', genre='$genre',
country='$country', url_web='$url_web', url_facebook='$url_facebook',
url_twitter='$url_twitter', image='$image'  WHERE id='".$fila[0]."'";
      mysql_query($sql,$conexion) or die ("<center>ERROR al Actualizar los
Artistas</center>");
//Los álbumes no se pueden actualizar directamente, porque posiblemente hayan
cambiado agregando la nueva discografía. Por lo que además de actualizar habría que
insertar creando complejidad y lentitud en el script. Para simplificar el script y aumentar su
velocidad se borrarán los álbumes y se re insertaran actualizados.

//No sabemos si han insertado albumes nuevos no podemos actualizar borramos y
re-insertamos
      $sql = "DELETE FROM album WHERE id_artist='".$fila[0]."'";
mysql_query($sql,$conexion) or die ("<center>ERROR al Actualizar los
Álbumes</center>");

//Consultamos los álbumes de cada artista a la API
      $json =
file_get_contents("https://theaudiodb.com/api/v1/json/1/searchalbum.php?s=".urlencode($f
ila[1]));
      $data = json_decode($json,true);

      //Obtenemos los datos
      foreach($data["album"] as $values)
      {
       $title =  mysql_real_escape_string($values["strAlbum"]);
       $year = $values["intYearReleased"];
       $img =  $values["strAlbumThumb"];

     //Insertamos los datos nuevos en la base de datos
```

```
      $sql = "INSERT INTO album (id_artist,title,year,image) VALUES
('".$fila[0]."','$title','$year','$img')";
       mysql_query($sql,$conexion) or die ("<center>ERROR al añadir álbumes</center>");
      }

}
//-----------------------------------------------------------------------------------------------------------------------
```

Bibliografía

[1] Documentación API the audiodb. Visitada: 6/5/2020

https://www.theaudiodb.com/api_guide.php

[2] Documentación API LastFM. Visitada: 6/5/2020

https://www.last.fm/api/

[3] Documentación API Spotify. Visitada: 6/5/2020

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

[4] Documentación API Deezer. Visitada: 6/5/2020

https://developers.deezer.com/api

[5] Información API música. Visitada: 6/5/2020

https://rapidapi.com/blog/top-free-music-data-apis/

[6] Documentación Android Studio. Visitada: 20/6/2020

https://developer.android.com/studio/intro?hl=es-419

[7] Documentación Android. Visitada: 22/6/2020

https://developer.android.com/docs

[8] Documentación Bootstrap. Visitada: 17/5/2020

https://getbootstrap.com/docs/4.5/getting-started/introduction/

[9] Documentación PHP. Visitada: 18/5/2020

https://www.php.net/docs.php

[10] Documentación JQuery. Visitada: 23/5/2020

https://api.jquery.com/

[11] Documentación Typeahead. Visitada: 24/5/2020

https://github.com/twitter/typeahead.js/blob/master/doc/jquery_typeahead.md

[12] Documentación CSS. Visitada: 14/5/2020

http://www.w3schools.com/css/

[13] Documentación HTML. Visitada: 14/5/2020

http://www.w3schools.com/html/

[14] Información API REST. Visitada: 28/6/2020
https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos

[15] Información API RESTFULL. Visitada: 28/6/2020

http://www.weblantropia.com/2016/05/24/restful-api-que-es/

[16] Información XML y JSON. Visitada: 29/6/2020

https://insights.dice.com/2019/01/25/xml-vs-json-difference-developers/

[17] Información metodologías. Visitada: 29/6/2020
https://www.digital55.com/desarrollo-tecnologia/mejores-metodologias-agiles-creacion-software/

[18] Información metodología en cascada. Visitada: 1/7/2020
https://obsbusiness.school/es/blog-project-management/metodologia-agile/pros-y-contras-de-la-metodologia-en-cascada

[19] Información Roy T. Fielding. Visitada: 1/7/2020

https://www.ics.uci.edu/~fielding/

[20] . Información API. Visitada: 28/6/2020

https://juanda.gitbooks.io/webapps/content/api/arquitectura-api-rest.html

[21] Información modelo cascada. Visitada: 30/6/2020
https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada

[22] Documentación Picasso Android. Visitada: 22/5/2020

https://square.github.io/picasso/