

UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR



Grado en Ingeniería Informática

TRABAJO FIN DE GRADO

**DESARROLLO DE UN SISTEMA DE RECOGIDA DE
DATOS EN FACEBOOK APLICANDO GAMIFICACIÓN**

Diego Castaño Chillarón
Tutor: Alejandro Bellogín Kouki
Ponente: Pablo Castells Azpiliceta

JUNIO 2015

Resumen

El surgimiento del *Big Data* unido al auge de las redes sociales en los últimos años ha propiciado que éstas almacenen cada vez más datos de los usuarios, en un intento de rentabilizar su modelo de negocio, explotándolos principalmente a través de la publicidad dirigida. Facebook se encuentra a la cabeza en esta carrera por los datos, almacenando información de 1.44 billones de usuarios (datos personales, gustos, relaciones, etc.).

Con este creciente ecosistema surgen nuevas oportunidades para el área de la minería de datos, que podría servirse de este tipo de conjuntos para, por ejemplo, desarrollar sistemas de recomendación. Por otro lado, a la par que nuevas oportunidades, surgen nuevos desafíos, entre ellos el de cómo conseguir acceso a estos datos. La privacidad se ha convertido en una prioridad para los usuarios de las redes sociales, por lo que normalmente sólo podremos acceder a aquellos datos que los propios usuarios elijan compartir con nosotros.

El objetivo de este proyecto era estudiar la posibilidad de aplicar gamificación a una aplicación en Facebook para conseguir recoger los datos del máximo número posible de usuarios. Entendemos por gamificación la extrapolación de mecánicas propias de los videojuegos a otros contextos. En nuestro caso, se buscó especialmente potenciar la componente social para crear un efecto viral en la difusión de la aplicación.

En este documento se describe detalladamente cómo se analizó, diseñó, implementó, probó y monitorizó una aplicación que recoge datos de Facebook aplicando mecánicas de gamificación. Para reducir el dominio de la aplicación, se decidió que ésta analizaría los gustos musicales de los usuarios, mostrándole unos resultados personalizados y permitiendo comparar sus resultados con los de sus amigos.

La aplicación fue publicada y a lo largo de un mes logró recoger datos de 170 usuarios, incluyendo sus “Me gusta” y sus relaciones de amistad. Como resultado de este proyecto se publicó el código de la aplicación en un repositorio de GitHub y se cedió el conjunto de datos recogidos al grupo de investigación IRG (1) de la Universidad Autónoma de Madrid.

Palabras clave

Facebook, gamificación, datos, minería, recomendación.

Abstract

The rise of Big Data in addition to the growth of online social networks in recent years has made them store more and more user data in an attempt to monetize their business model, exploiting it mainly through targeted advertising. Facebook is ahead in this race for the data, storing information of 1.44 billion users (personal data, likes, relationships, etc.).

With this growing ecosystem new opportunities appear for the area of data mining, which could use this kind of data sets to, for instance, develop recommendation systems. On the other hand, new opportunities emerge together with new challenges, including how to access these data. Privacy has become a priority for users of social networks, so normally we will be able to access only the data that the users choose to share with us.

The main goal of this project was to study the possibility of applying gamification to a Facebook application in order to collect data of the maximum possible amount of users. Gamification is the act of extrapolating gaming mechanics to other contexts. In our case, it sought specifically to improve the social component to create a viral effect in the spreading of the application.

This document describes in detail how we analyzed, designed, implemented, tested and monitored an application that collects data from Facebook applying gamification mechanics. To reduce the application domain, it was decided that the application would analyze the musical likes of the users, showing some personalized results and allowing them to compare their results with their friend's ones.

The application was published and managed to collect data from 170 users over the course of a month, including their likes and their friendships. As a result of this project the application code was published in a GitHub repository and the yielded data set was transferred to the IRG (1) research group of the Autonomous University of Madrid.

Keywords

Facebook, gamification, data, mining, recommendation.

Índice de contenidos

Índice de contenidos	5
Ilustraciones	8
Glosario	11
1. Introducción	13
1.1 Motivación	13
1.2 Objetivos	14
1.3 Estructura	14
2. Tecnologías a utilizar	17
2.1 Tipos de aplicaciones en Facebook	17
2.2 Extracción de datos	18
2.2.1 APIs de Facebook	18
2.2.2 SDKs para la Graph API de Facebook	19
2.2.3 APIs de datos de música	20
2.2.4 SDKs para LastFM	21
2.3 Entorno de producción	21
2.4 Lenguajes	22
2.5 Herramientas de desarrollo	22
3. Análisis	25
3.1 Descripción de la aplicación	25
3.1.1 Datos a almacenar	25
3.1.2 Datos a analizar	26
3.1.3 Técnicas de gamificación a emplear	26
3.1.4 Resultados que mostrará la aplicación	27
3.2 Requisitos funcionales	27
3.3 Requisitos no funcionales	29
3.4 Casos de uso	29
3.4.1 Análisis de gustos	29

3.4.2	Compartir un resultado	30
3.4.3	Invitar a un amigo	30
4.	Diseño	31
4.1	Arquitectura.....	31
4.1.1	Arquitectura física (hardware)	31
4.1.2	Arquitectura monopágina (SPA)	32
4.2	Subsistemas en detalle.....	33
4.2.1	Servidor de base de datos	33
4.2.2	Servidor web	34
4.2.3	Cliente.....	36
4.2.4	Patrones de diseño empleados.....	39
5.	Implementación.....	41
5.1	Modelo de ciclo de vida aplicado	41
5.2	Servidor de base de datos	41
5.2.1	Especificaciones del servidor	41
5.2.2	Creación de tablas	42
5.2.3	Diagrama de tablas.....	42
5.2.4	Recopilación y carga inicial	43
5.3	Servidor web.....	44
5.3.1	Framework CodeIgniter.....	44
5.3.2	Hospedaje en Heroku	45
5.3.3	Integración con Facebook.....	48
5.4	Cliente	51
5.4.1	Pantallas	51
5.4.2	Bibliotecas JavaScript	55
6.	Pruebas	57
6.1	Pruebas unitarias	57
6.2	Pruebas de integración	58
6.3	Pruebas de casos de uso	59
6.3.1	Análisis de gustos	59
6.3.2	Compartir un resultado	59
6.3.3	Invitar a un amigo	59

7. Resultados	61
7.1 Seguimiento por Facebook Analytics	61
7.2 Conjunto de datos obtenido.....	62
8. Conclusión y trabajos futuros.....	63
9. Referencias.....	65
10. Anexos técnicos	69
10.1 Script de creación de tablas.....	69
10.2 Fichero de idioma: Castellano	71
10.3 Fichero de idioma: Inglés.....	72
10.4 Pruebas unitarias automatizadas con QUnit	73
10.5 Pruebas de integración automatizadas con Qunit.....	75
10.6 Prueba de caso de uso C1: Análisis de gustos	77
10.7 Prueba de caso de uso C2: Compartir un resultado.....	80
10.8 Prueba de caso de uso C3: Invitar a amigos	82
10.9 Seguimiento por medio de Facebook Analytics for Apps.....	84
10.10 Vistas creadas en la base de datos.....	86
10.11 Política de privacidad	87

Ilustraciones

Ilustración 1 – Arquitectura física.....	31
Ilustración 2 – Ciclo de vida en arquitectura tradicional	32
Ilustración 3 – Ciclo de vida en arquitectura monopágina	33
Ilustración 4 – Diagrama Entidad-Relación	34
Ilustración 5 – Diagrama UML del servidor.....	35
Ilustración 6 – Diagrama UML del cliente	37
Ilustración 7 – Diagrama UML del patrón Proxy	39
Ilustración 8 – Diagrama de secuencia de LastFMProxy.....	40
Ilustración 9 – Commits a lo largo del tiempo en el repositorio de GitHub	41
Ilustración 10 – Diagrama de tablas	42
Ilustración 11 – Fragmento del script de carga inicial	43
Ilustración 12 – Estructura de directorios de CodeIgniter	44
Ilustración 13 – Fichero composer.json	45
Ilustración 14 – Despliegue usando GIT.....	46
Ilustración 15 – Logs de Heroku	46
Ilustración 16 – Programación de cronjob	47
Ilustración 17 – Página principal del panel de control de Facebook Developers.....	48
Ilustración 18 – Apartado de ajustes de la aplicación en Facebook Developers	49
Ilustración 19 – Aprobación de acceso a likes.....	50
Ilustración 20 – Menú principal de Facebook Analytics for Apps	51
Ilustración 21 – Pantalla principal	52
Ilustración 22 – Pantalla de análisis.....	52
Ilustración 23 – Pantalla de resultados. Resultados de amigos	53
Ilustración 24 – Pantalla de resultados. Resultados del usuario.....	54
Ilustración 25 – Pruebas unitarias con PHPUnit.....	57
Ilustración 26 – Número de tuplas por tabla	62
Ilustración 27 – Grafo de relaciones de amistad entre usuarios.....	62
Ilustración 28 – Script SQL de creación de tablas.....	70
Ilustración 29 – Fichero de lenguaje castellano	71
Ilustración 30 – Fichero de lenguaje inglés.....	72
Ilustración 31 – Pruebas unitarias con PHPUnit.....	74
Ilustración 32 – Pruebas de integración con PHPUnit.....	76

Ilustración 33 – Pantalla principal	77
Ilustración 34 – Diálogo de autorización de permisos.....	77
Ilustración 35 – Pantalla de análisis.....	78
Ilustración 36 – Pantalla de resultados.....	78
Ilustración 37 – Mensaje informativo.....	79
Ilustración 38 – Notificación	79
Ilustración 39 – Compartir un resultado	80
Ilustración 40 – Diálogo de publicación	80
Ilustración 41 – Resultado publicado en el muro de un usuario.....	81
Ilustración 42 – Botón de invitar a amigos	82
Ilustración 43 – Envío de invitaciones a amigos.....	82
Ilustración 44 – Solicitud recibida	83
Ilustración 45 – Evolución del número de usuarios.....	84
Ilustración 46 – Notificaciones enviadas.....	84
Ilustración 47 – Accesos desde resultados publicados en el muro	85
Ilustración 48 – Número de eventos por tipo	85
Ilustración 49 – Creación de vistas en base de datos	86
Ilustración 50 – Política de privacidad	87
Ilustración 51 – Privacy Policy	88

Tablas

Tabla 1 – Caso de uso C1: Análisis de gustos	29
Tabla 2 – Caso de uso C2: Compartir un resultado	30
Tabla 3 – Caso de uso C3: Invitar a un amigo.....	30
Tabla 4 – Eventos capturados por Facebook Analytics.....	61

Glosario

AJAX

Acónimo de Asynchronous Javascript And XML. Tecnología utilizada para transmitir información entre un navegador y un servidor sin actualizar la página.

API

Acónimo de Application Program Interface. Interfaz de una aplicación que puede ser usada por otro programa.

Backend

Relacionado con el lado del servidor. Lo que el usuario no puede ver.

Biblioteca

También denominadas librerías. Conjunto de implementaciones funcionales que ofrece una interfaz bien definida para la funcionalidad que se invoca.

Big Data

Término amplio usado para referirse a grandes conjuntos de datos para los cuales no son adecuados los métodos tradicionales de procesamiento.

DOM

Acónimo de Document Object Model. Es una interfaz de programación para documentos HTML, XML y SVG.

Facebook

Servicio de red social online creado en 2004 por Mark Zuckerberg y sus compañeros de universidad.

Framework

Entorno de trabajo. Conjunto de herramientas diseñado para simplificar una tarea de desarrollo.

Frontend

Relacionado con el lado del cliente. Lo que el usuario puede ver.

Gamificación

Extrapolación de mecánicas propias de videojuegos a otros contextos.

Heroku

Plataforma en la nube que ofrece servicios de alojamiento y soporta múltiples lenguajes de programación.

HTTPS

Protocolo HTTP seguro. Utiliza certificados SSL o TLS para cifrar la comunicación.

JSON

Acrónimo de JavaScript Object Notation. Formato de intercambio de datos popular por ser más liviano que XML.

Like

También denominado “Me gusta”. Es una funcionalidad de Facebook mediante la cual un usuario puede dar a conocer que un determinado objeto (post, página, etc.) le gusta.

Minería de datos

Campo de las ciencias de la computación referido al proceso que intenta descubrir patrones en grandes volúmenes de datos

Red social

Entendido como servicio de red social. Medio de comunicación que se centra en establecer un contacto con otras personas por medio de Internet.

REST

Acrónimo de Representational State Transfer. Estilo de arquitectura de software consistente de normas y buenas prácticas para crear servicios web escalables.

SDK

Acrónimo de Software Development Kit. Conjunto de herramientas diseñado para desarrollar con una determinada tecnología.

Sistema de recomendación

Sistema de filtrado de información que busca predecir la puntuación o preferencia que un usuario daría a un ítem.

Twitter

Servicio de red social online que permite a sus usuarios enviar y leer mensajes de 140 caracteres denominados “tweets”.

1. Introducción

1.1 Motivación

A lo largo de los últimos 15 años se ha producido un auge de las redes sociales, liderado actualmente por Facebook, que cuenta con 1.44 billones de usuarios activos al mes (2), esto es, aproximadamente un 20% de la población total del planeta (3).

El crecimiento en la popularidad de las redes sociales ha brindado a los usuarios las herramientas necesarias para poder producir y compartir cada vez más contenido a través de Internet. Esto, unido al abaratamiento de los medios de almacenamiento, ha propiciado que las empresas recopilen ingentes cantidades de datos en un intento de rentabilizar sus modelos de negocio (4).

Así, las redes sociales se han convertido en una suerte de repositorios masivos de datos de lo más heterogéneo: Datos demográficos anónimos, información personal, relaciones entre usuarios, etc.

Con este creciente ecosistema surgen nuevas oportunidades para la minería de datos, y es que este tipo de conjuntos es especialmente atractivo para las áreas de análisis de redes sociales y sistemas de recomendación.

Por otro lado, también se plantean nuevos desafíos. ¿Cómo extraer información útil de cantidades tan grandes de datos? Para esta pregunta encontramos respuesta en los nuevos métodos de análisis, surgidos tras el boom del Big Data.

¿Cómo podemos acceder a esta información? Normalmente las redes sociales protegen la privacidad de sus usuarios de modo que sólo permiten acceder a aquella información que los usuarios eligen compartir. Así, por ejemplo, Twitter permite a sus usuarios crear cuentas públicas o protegidas y Facebook da la opción de limitar quién puede ver las publicaciones de un usuario (5).

Otro modo de acceder a estos datos, en el que se centrará este proyecto, es mediante aplicaciones embebidas en las propias redes sociales. En este caso el usuario autoriza a la aplicación a acceder a todos o alguno de sus datos.

Pero, ¿por qué querría alguien autorizar a una aplicación a acceder a sus datos en una red social? Aquí es donde entra en juego la gamificación.

Entendemos por gamificación la extrapolación de mecánicas propias de (video)juegos a otros contextos (6). En el caso que nos ocupa, la idea es recompensar al usuario (o darle algún tipo de retroalimentación) cuando autorice a nuestra aplicación a acceder a sus datos o interaccione con ella.

1.2 Objetivos

La finalidad de este trabajo es desarrollar un sistema de recogida de datos utilizando para ello una aplicación embebida en Facebook. Para incitar al usuario a que autorice a la aplicación para acceder a su información se analizarán sus gustos (“likes” en Facebook) y se mostrarán unos resultados personalizados.

Además, se aplicará gamificación para buscar la componente social y así conseguir un efecto viral, usando comparaciones entre los resultados del usuario y los de sus amigos.

La explotación del conjunto de datos que se obtenga queda fuera del alcance de este proyecto, aunque sí se analizará brevemente y se comentarán sus posibles utilidades.

1.3 Estructura

Este documento consta de las siguientes partes:

1. Introducción

Descripción de la motivación y el alcance del proyecto.

2. Tecnologías a emplear

Discusión sobre las tecnologías seleccionadas para implementar el proyecto.

3. Análisis

Descripción detallada del problema a resolver. Disección en requisitos (funcionales y no funcionales) y definición de los casos de uso.

4. Diseño

Explicación en profundidad de las decisiones tomadas para planificar la implementación.

5. Implementación

Descripción de las técnicas, métodos y herramientas utilizadas para desarrollar el diseño planteado.

6. Pruebas

Descripción de los procesos seguidos para asegurar el correcto funcionamiento y calidad de los módulos implementados.

7. Resultados

Descripción de los métodos que se usaron para monitorizar el uso de la aplicación, así como de los datos que se consiguieron recoger.

8. Conclusión y trabajos futuros

Disertación sobre el significado de los resultados obtenidos y futuras mejoras o extensiones.

9. Referencias

Listado de fuentes de información consultadas para el desarrollo de este proyecto.

10. Anexos técnicos

Listado de recursos técnicos complementarios que pudieran ser de interés.

2. Tecnologías a utilizar

A continuación se describen brevemente las tecnologías a emplear en la implementación de la aplicación.

2.1 Tipos de aplicaciones en Facebook

Dentro del ecosistema de Facebook podemos diferenciar distintos tipos de aplicaciones en base a la plataforma para la cual han sido desarrolladas. A continuación se describen las particularidades de cada tipo de aplicación.

Aplicación Android

Facebook ofrece un SDK para desarrollar aplicaciones Android que usen sus servicios desde Java. La aplicación requerirá que el dispositivo del usuario utilice una versión de Android igual o superior a la 2.3 (Gingerbread, i.e. API 9 o superior).

Permite acceso a las siguientes funcionalidades: Login, Share, App Invites, App Links, App Events, Ads, Audience Network, Graph API (7).

Aplicación iOS

Facebook ofrece otro SDK para desarrollar aplicaciones en dispositivos con sistema operativo iOS que utilicen sus servicios desde Objective C. La aplicación requerirá que el dispositivo del usuario utilice una versión de iOS igual o superior a la 7.

Permite acceso a las siguientes funcionalidades: Share, Login, Ads, App Events, Grap API, Audience Network, App Links (8).

Sitio web

Facebook permite que sitios web externos utilicen algunas de sus funcionalidades. Para que una aplicación no-embebida pueda hacer consultas a la API de Facebook el usuario deberá primero iniciar sesión en esa página (9).

Aplicación Canvas

Son aplicaciones embebidas que están integradas dentro de la propia red social mediante un marco (iframe) y no requieren ningún paso extra por parte del usuario. Son accesibles desde URLs internas de Facebook como: `apps.facebook.com/miaplicacion`

Aplicación multiplataforma

Facebook permite asociar a una aplicación de tipo “Canvas” otras de tipo Android y/o iOS, de modo que cada usuario reciba una experiencia ajustada al entorno nativo de sus dispositivos.

Elección del tipo de aplicación

Se escogió la opción embebida, ya que facilita el descubrimiento de la aplicación por parte de los usuarios y se integra con la red social de forma más natural, al no ser necesario abandonar el dominio de Facebook (10).

2.2 Extracción de datos

En este apartado se detallan qué fuentes de datos serán utilizadas en la aplicación, así como qué bibliotecas se utilizarán para acceder a las mismas.

2.2.1 APIs de Facebook

Actualmente Facebook mantiene tres APIs bien diferenciadas: Atlas API, Marketing API y Graph API. A continuación se describirá brevemente la utilidad de cada una.

Atlas API

Esta es la API de Facebook para anunciantes y partners. Permite crear y actualizar programáticamente la compra de anuncios y emplazamientos publicitarios, así como recuperar información sobre las entidades asociadas a una cuenta (11).

Marketing API

Mediante la API de marketing se puede acceder a la plataforma de publicidad de Facebook desde tus propias herramientas publicitarias. La API de marketing permite a todos los desarrolladores utilizar las mismas funciones que las herramientas de Facebook en una solución personalizada (12).

Graph API

La Graph API es el método primario para introducir y extraer datos de la plataforma de Facebook. Es una API de bajo nivel basada en HTTP que se puede usar para consultar datos, postear nuevas historias, gestionar anuncios, subir fotos y una variedad de otras tareas que una aplicación pudiera necesitar hacer (13).

Elección de API de Facebook

Dado que nuestra aplicación no requiere gestionar campañas ni emplazamientos publicitarios nos limitaremos a utilizar la Graph API, que es suficiente para que la aplicación pueda introducir y extraer datos.

En el momento en que se comenzó este proyecto (octubre de 2014) se decidió utilizar la versión más alta disponible, que entonces era la **2.2**, por lo que todos los métodos que se mencionen en el resto del documento harán referencia a dicha versión.

2.2.2 SDKs para la Graph API de Facebook

Estas bibliotecas encapsulan las llamadas a la API, y ofrecen una robusta implementación de la autenticación, comunicación, control de errores y parseo, ideal para simplificar el manejo estructurado de los datos.

Existen cinco SDKs principales desarrollados por Facebook para realizar consultas a la API desde diversas plataformas: Unity, iOS, Android, PHP y JavaScript. Por simplicidad a continuación sólo se describen los SDKs de PHP y JavaScript, ya que son los únicos compatibles con aplicaciones “Canvas”.

SDK para PHP

En nuestra aplicación será necesario recuperar grandes cantidades de información (todos los likes del usuario, sus amigos, etc.) en el lado del servidor (para que no sea visible al usuario) a gran velocidad. Para esto se utilizará el SDK para PHP en su versión 4.0.0 (14).

SDK para JavaScript

Es necesario que la aplicación pueda realizar ciertas consultas a la API también en el lado del cliente. Como mínimo, se utilizará para solicitar permisos de acceso a los datos del usuario. Esto solo puede realizarse desde JavaScript por lo que se usará el SDK específico para este lenguaje (9).

Elección de SDKs para la Graph API de Facebook

Como se verá más adelante, nuestra aplicación sigue un patrón MVC, tanto en el lado del servidor como en el lado del cliente, y se requiere de funcionalidades de Facebook en los modelos tanto en el backend como en el frontend, por lo que se usarán tanto el SDK de PHP como el de JavaScript.

2.2.3 APIs de datos de música

La aplicación deberá analizar los gustos musicales del usuario, por lo que fue necesario buscar una base de datos sobre música que fuera accesible programáticamente, es decir a través de una API.

API de Wikipedia

Wikipedia es una enciclopedia online gratuita de contenido abierto que se ha creado mediante el esfuerzo colaborativo de una comunidad. Cualquier usuario registrado en el sitio web puede crear un artículo para su publicación, mientras que no se requiere estar registrado para modificar artículos.

No dispone de API pública oficial. Existe, sin embargo una API no oficial en MediaWiki (15). Tiene datos muy limitados de artistas poco conocidos.

API de MusicBrainz

MusicBrainz (16) es una enciclopedia musical de contenido abierto que almacena metadatos de artistas y los hace disponibles al público.

Dispone de gran variedad de datos en una API pública oficial, bien estructurada, pero tampoco dispone de datos sobre artistas poco conocidos.

API de LastFM

Last.fm es un servicio para descubrir música que da recomendaciones personalizadas basadas en la música que escuchas. Dispone de una amplia base de datos creada a partir de los metadatos que sus usuarios publican al escuchar música (scrobbling (17)), por lo que su base de datos abarca mayor cantidad de artistas menos conocidos.

Last.fm dispone de una API REST pública y oficial extensamente documentada (18). Devuelve datos en formato XML o JSON.

Elección de API de datos de música

Se valoraron otras opciones (Wikipedia y MusicBrainz) pero finalmente fueron descartadas en favor de LastFM, ya que esta última contiene información de más artistas que las otras opciones, además de una API que se usa más ampliamente.

2.2.4 SDKs para LastFM

LastFM no ofrece ningún SDK oficial para acceder a su API. Sí ofrece, en cambio, varias alternativas no oficiales de código abierto. A continuación se describen los dos SDKs de código abierto que fueron valorados.

SDK para PHP

Este SDK para PHP fue desarrollado por Matt Oakes y consiste de una serie de clases que permiten acceder a información de usuarios, artistas, albums, tracks, grupos, eventos y tags. Esta librería dejó de ser mantenida en 2014. El código de esta biblioteca puede encontrarse en el siguiente repositorio: <https://github.com/matto1990/PHP-Last.fm-API>

SDK para JavaScript

Para consultar la API de LastFM desde JavaScript se utilizó una biblioteca no oficial que se ocupa de simplificar las llamadas asíncronas (AJAX) y cachear peticiones. El código de esta biblioteca puede encontrarse en el siguiente repositorio: <https://github.com/fxb/javascript-last.fm-api>

Elección de SDK para la API de LastFM

Nuestra aplicación deberá realizar múltiples consultas a la API de LastFM para analizar cada uno de los artistas que le gusten a un usuario. Este proceso puede ser largo y provocar que el servidor web se bloquee hasta que reciba respuesta por parte de la API. Por ello se decidió utilizar sólo el SDK para JavaScript, para liberar al servidor de la lenta tarea que supone realizar muchas consultas, tarea que es mucho más manejable mediante AJAX.

2.3 Entorno de producción

La aplicación se desarrollará sobre un entorno típico LAPP: Linux, Apache, PostgreSQL y PHP. Se escogió esta opción por ser un marco de trabajo muy extendido en desarrollo web y por ser compatible con las APIs requeridas.

Linux

Sistema operativo de código abierto utilizado ampliamente en servidores. En nuestro caso se utilizó la distribución Ubuntu en su versión 14.04 (19).

Apache 2

Servidor HTTP más utilizado del mundo. Es de código abierto y altamente configurable (20).

PostgreSQL

Software gestor de bases de datos relacionales de código abierto. Es multiplataforma y cumple con el estándar ACID (21).

PHP

Intérprete del lenguaje de scripting PHP utilizado para programar servidores web. Se utilizó la versión 5.4 (22) junto al gestor de dependencias Composer (23) por su compatibilidad con el SDK de Facebook.

2.4 Lenguajes

PHP

Lenguaje de scripting de propósito general especialmente pensado para el lado del servidor. Es rápido, flexible y ampliamente utilizado en sitios web de todo el mundo.

JavaScript

Lenguaje de scripting para el lado del cliente. Es funcional, orientado a objetos, con herencia prototípica y puramente asíncrono. Es soportado por defecto por la mayoría de navegadores web modernos.

2.5 Herramientas de desarrollo

NetBeans

Entorno de desarrollo de propósito general. Integra control de versiones, funciones de autocompletar, coloreado de sintaxis, depuración, generación de documentación y soporta la inclusión de múltiples plugins (24).

pgAdmin III

Software de administración para servidores de bases de datos PostgreSQL. Permite crear, destruir, consultar y editar tablas, ejecutar consultas, realizar backups, optimizar tablas e índices, etc (25).

GIT

Software gratuito para el control de versiones con énfasis en la rapidez, integridad de los datos y soporte para flujos de trabajo -lineales y/o distribuidos. En este caso se usará para enviar los cambios a un repositorio privado creado en GitHub (26).

Node.js

Plataforma construida sobre el potente motor V8 de JavaScript (creado por Google para Chrome). Permite crear aplicaciones escalables y es dirigido a eventos (27).

Además, dispone de un cómodo gestor de paquetes (Node Package Manager) que es ampliamente usado, independientemente del servidor web.

3. Análisis

A continuación se describirá la aplicación a desarrollar.

3.1 Descripción de la aplicación

En este apartado se describe brevemente el funcionamiento deseado de la aplicación final.

3.1.1 Datos a almacenar

Dado que el objetivo último de la aplicación es recoger datos de Facebook, el primer paso es definir qué datos se quieren almacenar. Para seleccionar qué datos nos interesan consultamos la guía de referencia para desarrolladores de Facebook (28) donde se describen los datos disponibles.

Datos del usuario

Nos bastará con almacenar el identificador único que Facebook atribuye a cada usuario. La información pública (como el nombre, foto de perfil, fecha de nacimiento, etc.) puede ser recuperada a posteriori consultando la API de Facebook con el identificador de usuario, por lo que no se consideró necesario almacenarla.

Gustos del usuario

Teniendo en cuenta que nos interesará utilizar los datos recolectados para el desarrollo de sistemas de recomendación, se optó por almacenar todos los gustos (*likes*) del usuario.

En Facebook cada like de un usuario viene representado como una relación con un objeto (29). Cada objeto viene representado por una página, por ejemplo: <https://www.facebook.com/cher>

Así, se decidió almacenar el identificador único que Facebook atribuye a cada objeto que el usuario tiene entre sus likes.

Además de esto, se dispuso almacenar también la categoría de cada objeto, para discernir si se trata de una película, un cantante, un equipo deportivo, etc. Esta información también podría ser recuperada a posteriori consultando a la API, pero se consideró que resultaría útil tenerlo almacenado de antemano para construir sistemas de recomendación especializados, como por ejemplo un recomendador de películas.

Amigos del usuario

Almacenaremos también las relaciones de amistad del usuario, esto es, los identificadores de usuario de sus amigos, teniendo en cuenta que en Facebook, a diferencia de Twitter, las relaciones de amistad son mutuas (bidireccionales, forman un grafo no dirigido).

Estos datos son útiles por dos motivos principales: Por un lado nos servirá para analizar la propagación de la aplicación y su efecto viral. Por otro lado servirá también para desarrollar sistemas de recomendación basados en relaciones sociales (30).

3.1.2 Datos a analizar

Independientemente de los datos que se almacenen, la aplicación deberá analizar algunos de ellos para dar mostrar unos resultados al usuario. Por simplicidad se decidió centrarse en analizar los gustos musicales del usuario. Esto es, aquellos objetos que el usuario tenga entre sus likes y que estén en la categoría de artistas (en la API de Facebook están etiquetados como “Musician/band”) (31).

Se valoró también la posibilidad de analizar la información proveniente de Spotify a través del método `/user/activities` (32) de la API de Facebook, ya que esta información no queda reflejada en los likes del usuario. Finalmente esta aproximación se descartó por dos razones principales: No todos los usuarios publican en Facebook su actividad de Spotify, lo que limitaría mucho la propagación viral de la aplicación. Además, este método ofrece demasiada información con mucho ruido (en qué momento se escuchó cierto segmento de una canción, cuándo añadió una canción a una cierta lista de reproducción, etc.) qué sería más costosa de analizar. No obstante, esta funcionalidad se podría integrar en el futuro de manera más o menos sencilla en el proyecto realizado.

3.1.3 Técnicas de gamificación a emplear

La finalidad de aplicar gamificación a la aplicación era potenciar su componente social, así como reforzar positivamente las interacciones del usuario con la misma. Los mecanismos que se eligieron fueron:

Barra de progreso

Las barras de progreso ayudan a reforzar la sensación de que se está avanzando. En este caso se utilizarán para mostrar el avance del análisis de los gustos musicales, ya que podría tratarse de un proceso relativamente largo y es necesario mantener al usuario en la aplicación.

Scoreboard / tabla de puntuaciones

Los scoreboards son un elemento básico para potenciar la competición entre jugadores. Un usuario más competitivo interactuará más con la aplicación y provocará un mayor nivel de retención. En nuestro caso se mostrarán scoreboards con los resultados del usuario junto a los resultados de sus amigos.

Personalización / logros

El hecho de atribuir logros a cada persona crea la sensación de estatus o perfil. Ese perfil representa a la persona en el contexto de la aplicación, haciéndola sentir participe de la experiencia e incrementando su compromiso.

En nuestro caso la aplicación mostrará resultados personalizados para cada usuario en función de sus gustos musicales.

3.1.4 Resultados que mostrará la aplicación

Edad musical

La edad musical de un usuario se calculará como la edad promedio de los fans de los artistas que le gustan al usuario. La información de los fans se obtiene del API de LastFM.

Para calcular los top 3 géneros favoritos del usuario se obtendrán los 3 géneros primarios de cada artista (a partir de LastFM) y se ordenarán por frecuencia.

Concierto ideal

Se buscará el álbum más antiguo de entre todos los álbumes de todos los artistas. Se dará por hecho que el artista dio un concierto tras la salida del álbum.

Recomendaciones

Se buscarán artistas similares a los que le gustan a usuario y se ordenarán por frecuencia. Los artistas similares se obtienen usando el API de LastFM.

Gráfica comparativa

Se mostrará una gráfica de barras a modo de scoreboard con la edad musical del usuario junto a la que obtuvo cada uno de sus amigos.

3.2 Requisitos funcionales

A continuación se sintetiza la funcionalidad anteriormente descrita en un listado de requisitos funcionales:

RF1

La aplicación mostrará un botón de “Empezar” que al ser pulsado por el usuario provocará que la aplicación pida permiso para acceder a sus datos: Información personal, likes y lista de amigos.

RF2

Una vez se haya autorizado a la aplicación, ésta mostrará una barra de progreso que se irá completando a medida que se analicen los gustos del usuario.

RF3

La aplicación mostrará los resultados descritos en el apartado 2.1.4 personalizados para el usuario.

RF4

Cada resultado tendrá un botón “¿Cómo funciona?” que al pulsarlo muestre un mensaje informativo sobre el algoritmo utilizado.

RF5

Cada resultado tendrá un botón “Compartir” que al pulsarlo publicará el resultado en el muro del usuario.

RF6

La aplicación mostrará junto a la gráfica comparativa un botón “Invitar a amigos” que al pulsarlo permita enviar solicitudes a los amigos del usuario que aún no hayan utilizado la aplicación.

RF7

La aplicación mostrará en todo momento un botón de “Me gusta” y otro de “Compartir” en el pie de página.

RF8

La aplicación notificará a los amigos de un usuario cuando éste vuelva a ejecutar la aplicación, enviándoles un mensaje que les incite a comparar sus resultados.

3.3 Requisitos no funcionales.

RNF1

La aplicación deberá estar hospedada en un servidor con https para cumplir con la política de Facebook para aplicaciones embebidas.

RNF2

La aplicación deberá tener una página con su política de privacidad y condiciones de uso https para cumplir con la política de permisos de Facebook.

RNF3

La aplicación deberá detectar automáticamente la localización del usuario y mostrar todas las cadenas de texto en castellano o inglés según corresponda.

RNF4

El análisis al completo no deberá durar más de medio minuto.

3.4 Casos de uso

A continuación se describen en formato de tabla los tres casos de uso principales.

3.4.1 Análisis de gustos

Tabla 1- Caso de uso C1: Análisis de gustos

Descripción	C1. El sistema analiza los likes del usuario.
Requisitos asociados	RF1-RF3, RF8, RNF1-RNF4
Precondiciones	<ul style="list-style-type: none">• El usuario tiene cuenta de Facebook y ha iniciado sesión.• El usuario ha navegado hasta la URL de la aplicación embebida.
Flujo normal	<ol style="list-style-type: none">1. El usuario hace click en el botón “Empezar”.2. La aplicación muestra una barra de progreso mientras se analizan sus gustos.3. Una vez finalizado el análisis la aplicación muestra los resultados.
Postcondiciones	<ul style="list-style-type: none">• La aplicación habrá almacenado el identificador único del usuario en Facebook.• La aplicación habrá almacenado los likes del usuario.• La aplicación habrá almacenado los amigos del usuario.
Flujos alternativos	<ol style="list-style-type: none">2. b. El usuario no tiene likes musicales3. b. La aplicación muestra un mensaje informativo.
Frecuencia esperada	1000 veces al día

3.4.2 Compartir un resultado

Tabla 2- Caso de uso C2: Compartir un resultado

Descripción	C2. El usuario comparte uno de sus resultados en su muro de Facebook.
Requisitos asociados	RF5, RNF1-RNF3
Precondiciones	<ul style="list-style-type: none">• Ha completado el caso de uso C1 y ha obtenido sus resultados personalizados
Flujo normal	<ol style="list-style-type: none">1. El usuario hace click en el botón “Compartir” del resultado que quiere compartir.2. El sistema muestra un diálogo para que el usuario escriba un comentario que publicar junto al enlace a la aplicación.3. El usuario hace click en el botón “Compartir” del diálogo.
Postcondiciones	<ul style="list-style-type: none">• Se publica en el muro del usuario un enlace a la aplicación, el texto que mostraba su resultado personalizado.
Flujos alternativos	-
Frecuencia esperada	1000 veces al día

3.4.3 Invitar a un amigo

Tabla 3 - Caso de uso C3: Invitar a un amigo

Descripción	C3. El usuario invita a un amigo a participar en la aplicación.
Requisitos asociados	RF6, RNF1-RNF3
Precondiciones	<ul style="list-style-type: none">• Ha completado el caso de uso C1 y ha obtenido sus resultados personalizados
Flujo normal	<ol style="list-style-type: none">1. El usuario hace click en el botón “Invitar a amigos”.2. La aplicación envía una invitación a los amigos que seleccione el usuario.
Postcondiciones	<ul style="list-style-type: none">• El usuario amigo recibirá una invitación con un enlace a la aplicación.
Flujos alternativos	-
Frecuencia esperada	1000 veces al día

4. Diseño

4.1 Arquitectura

En este apartado se describen las decisiones que se tomaron en cuanto al diseño arquitectónico de la aplicación, tanto a nivel de hardware como de software.

4.1.1 Arquitectura física (hardware)

La aplicación web es un sistema distribuido, del tipo cliente-servidor, que funciona sobre el protocolo https. A continuación se muestra un diagrama de las conexiones entre los distintos subsistemas hardware:

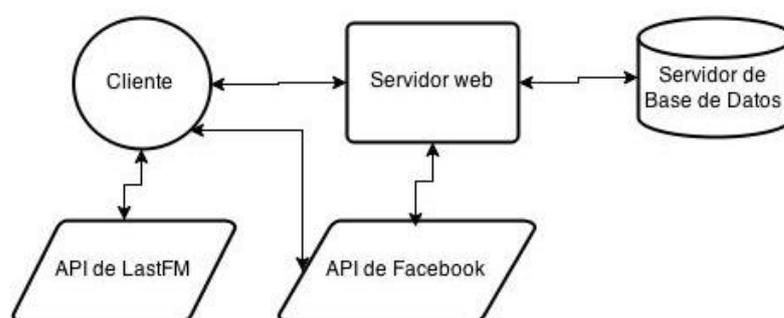


Ilustración 1 – Arquitectura física

En este diagrama se diferencian 5 subsistemas:

1. Cliente

Navegador web del usuario que ejecuta el código JavaScript de nuestra aplicación. Se comunica por el servidor web mediante peticiones GET y POST enviadas mediante AJAX. También se comunica con las APIs tanto de LastFM como de Facebook.

2. Servidor web

Recibe a las peticiones del cliente y las responde con documentos html o datos (mediante una API REST). A su vez, puede acceder a la base de datos o a la API de Facebook para recuperar información relevante para sus modelos.

3. Base de datos

Servidor externo que sólo recibirá conexiones entrantes desde el servidor web.

4. API de LastFM

API de LastFM será accedida únicamente desde el cliente, ya que se trata de una operación costosa.

Se ha decidido no efectuar llamadas a esta API desde el servidor, que de otra manera quedaría bloqueado esperando a que LastFM contestara a las peticiones, reduciendo la escalabilidad de la aplicación.

5. API de Facebook

La API de Facebook será accedida tanto desde el cliente (con JavaScript) como desde el servidor (con PHP). Desde el cliente se utilizará para realizar acciones como pedir autorización de permisos, compartir resultados o invitar a amigos.

4.1.2 Arquitectura monopágina (SPA)

Podemos diferenciar dos aproximaciones a la hora de estructurar una aplicación web: La tradicional (multipágina) y la monopágina (denominada single-page app o SPA).

En la siguiente figura se puede ver el ciclo de vida de una aplicación con arquitectura tradicional (33):

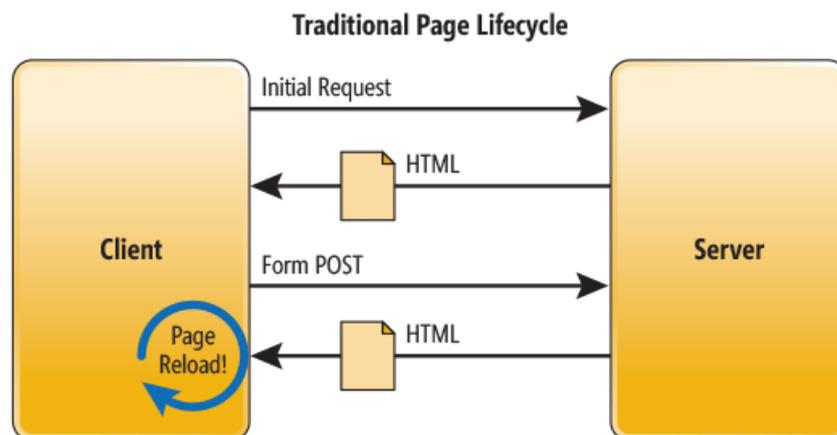


Ilustración 2 – Ciclo de vida en arquitectura tradicional

En este tipo de aplicaciones, cada vez que el cliente requiere de nuevos datos del servidor es necesario enviar una petición y esperar a que se cargue una nueva página al completo. En las aplicaciones monopágina, en cambio, no siempre es necesario cargar páginas al completo.

A continuación se muestra un diagrama que representa el ciclo de vida de una aplicación monopágina, con las peticiones HTTP que realiza el cliente y los documentos que envía como respuesta el servidor.

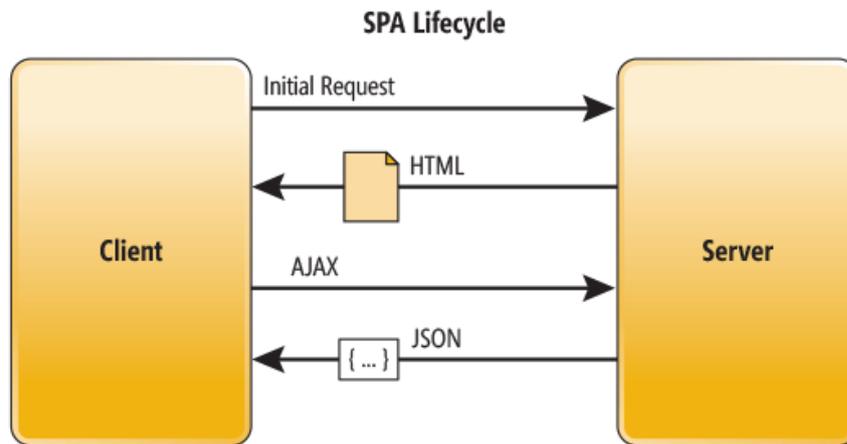


Ilustración 3 – Ciclo de vida en arquitectura monopágina

De esta manera, una vez que el cliente reciba el HTML básico de la página, si se requiere cargar nueva información desde el servidor para ser mostrada por pantalla, no será necesario recargar la página al completo, sino que se realizarán peticiones AJAX al servidor (y/o a las APIs correspondientes) y se actualizará solamente el contenido necesario de las vistas.

Se ha optado por emplear una arquitectura monopágina para liberar la carga de procesamiento del servidor todo lo posible.

4.2 Subsistemas en detalle

A continuación se describe en profundidad el diseño planteado para cada subsistema que compone la aplicación.

4.2.1 Servidor de base de datos

Aquí se especifica la estructura de los datos que deberá tener nuestra aplicación, teniendo en cuenta los requisitos funcionales anteriormente descritos.

4.2.1.1 Diagrama ER

En la siguiente figura podemos ver el diagrama entidad-relación diseñado para la base de datos de la aplicación:

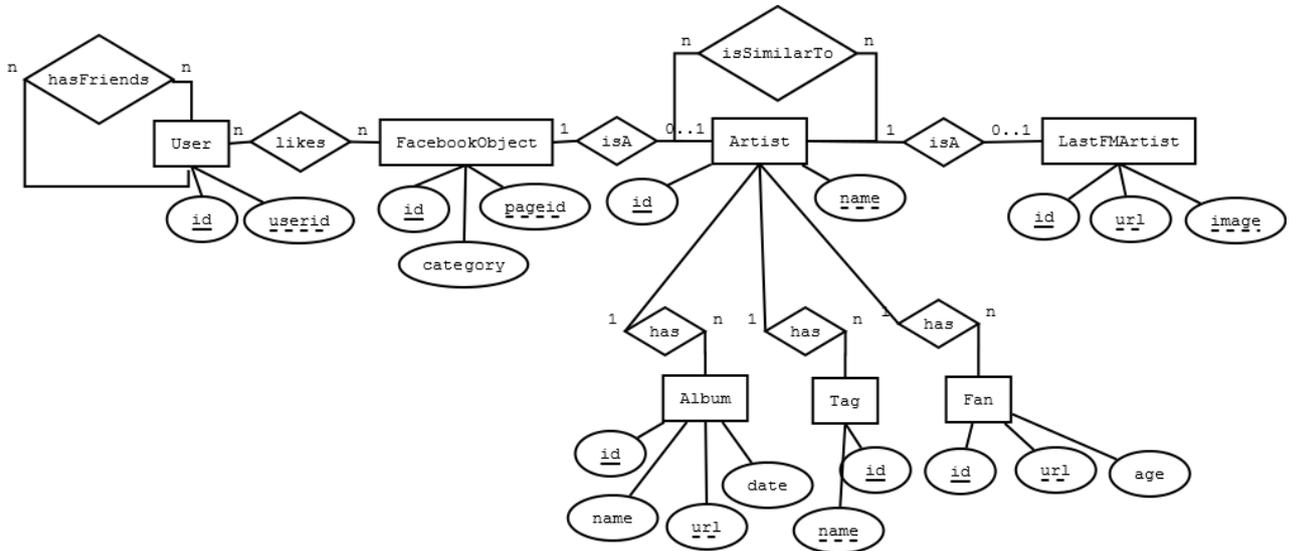


Ilustración 4 – Diagrama Entidad-Relación

Se pueden distinguir las entidades **User** y **FacebookObject** que representan usuarios y objetos de Facebook, respectivamente. Un usuario podrá ser amigo de otros usuarios y tendrá varios objetos entre sus likes.

Algunos de los objetos de Facebook serán artistas musicales, por ello la entidad **Artist** está relacionada con **FacebookObject**. Un artista podrá estar registrado o no en LastFM, por lo que la entidad **Artist** está relacionada con **LastFMArtist** en algunos casos.

Por otro lado están las entidades **Album**, **Tag** y **Fan**, que se relacionan con un artista y representan información proveniente de LastFM. Esta información será utilizada para calcular los resultados descritos en el apartado 3.1.4.

4.2.2 Servidor web

En los siguientes apartados se describe detalladamente el diseño orientado a objetos que se planteó para el lado del servidor de nuestra aplicación. Estos módulos serían implementados más adelante en PHP.

4.2.2.1 Diagrama UML

La siguiente figura muestra el diagrama UML del servidor, con los módulos que lo componen, los paquetes en que están englobados y sus dependencias entre sí:

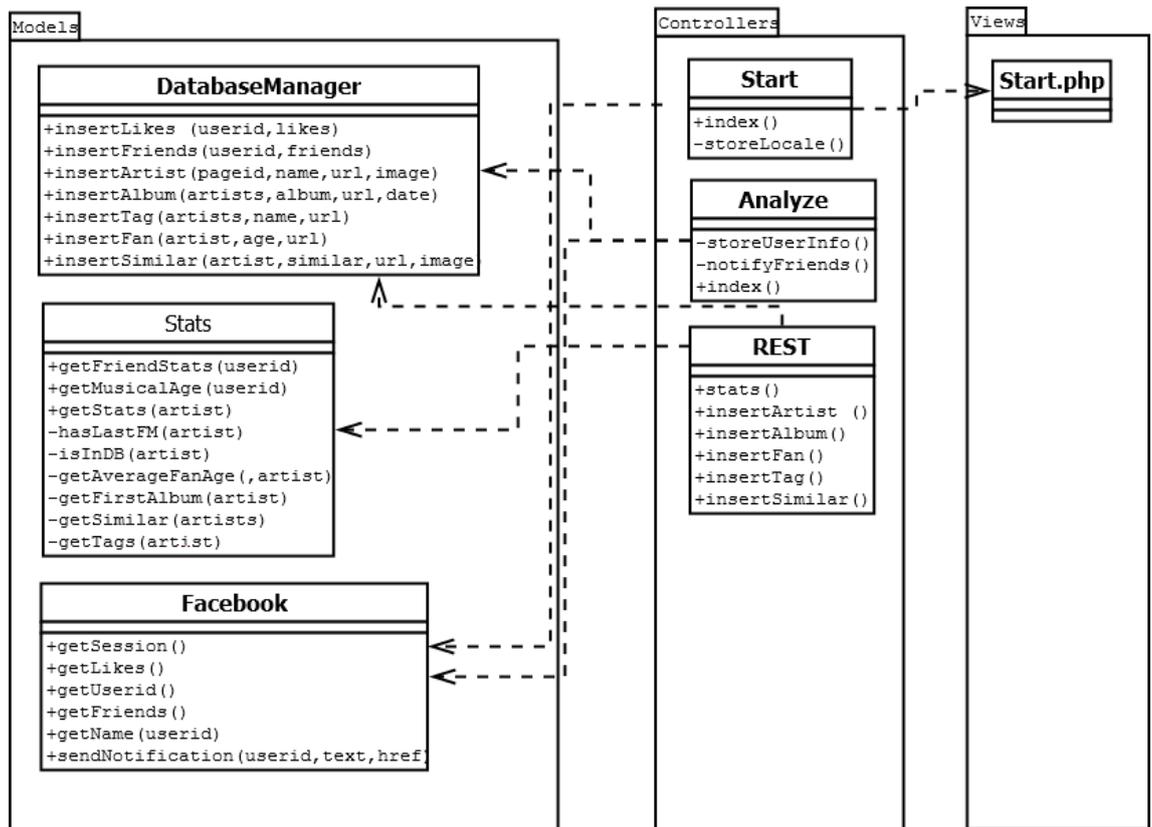


Ilustración 5 - Diagrama UML del servidor

4.2.2.2 Lógica de cada módulo

DatabaseManager

Modelo que se ocupa de ocultar la lógica que se utiliza para insertar objetos en las distintas tablas de la base de datos de manera consistente.

Stats

Este modelo reúne los métodos necesarios para extraer estadísticas de un artista (insertado de antemano) en la base de datos.

También contiene métodos para obtener las estadísticas de otros usuarios de la aplicación.

Facebook

Este modelo agrupa los métodos que requieren hacer peticiones a la API de Facebook. Hay métodos para recuperar la sesión de Facebook, para extraer información del usuario (nombre, likes y amigos) así como para enviar notificaciones a cualquier usuario en nombre de la aplicación.

Start

Controlador principal por defecto. Se encarga de almacenar la localización del usuario (para saber si usar cadenas en castellano o inglés) y mostrar la vista “start” con todo el HTML de la aplicación.

Analyze

Controlador que se ocupa de almacenar ordenadamente todos los datos del usuario (userid, amigos y likes) además de mandar notificaciones a los amigos del usuario y devolver en formato JSON los resultados que obtuvieron los amigos del usuario.

REST

Controlador que agrupa los métodos REST que se expondrán como API. Estos métodos serán consultados mediante peticiones GET o POST enviadas por AJAX e imprimirán los datos como un objeto en formato JSON.

Por un lado, el método “stats” devolverá estadísticas para un artista dado por método GET (almacenado previamente en base de datos).

Por otro lado, los métodos “insertArtist”, “insertAlbum”, “insertFan”, “insertTag” e “insertSimilar” encapsulan llamadas a los métodos de DatabaseManager desde una interfaz accesible por método POST.

4.2.3 Cliente

A continuación se describirán en detalle las clases diseñadas para el lado del cliente de nuestra aplicación. Más adelante, serán implementadas en JavaScript.

4.2.3.1 Diagrama UML

La siguiente figura muestra el diagrama UML del cliente, con los módulos que lo componen, los paquetes en que están englobados y sus dependencias entre sí.

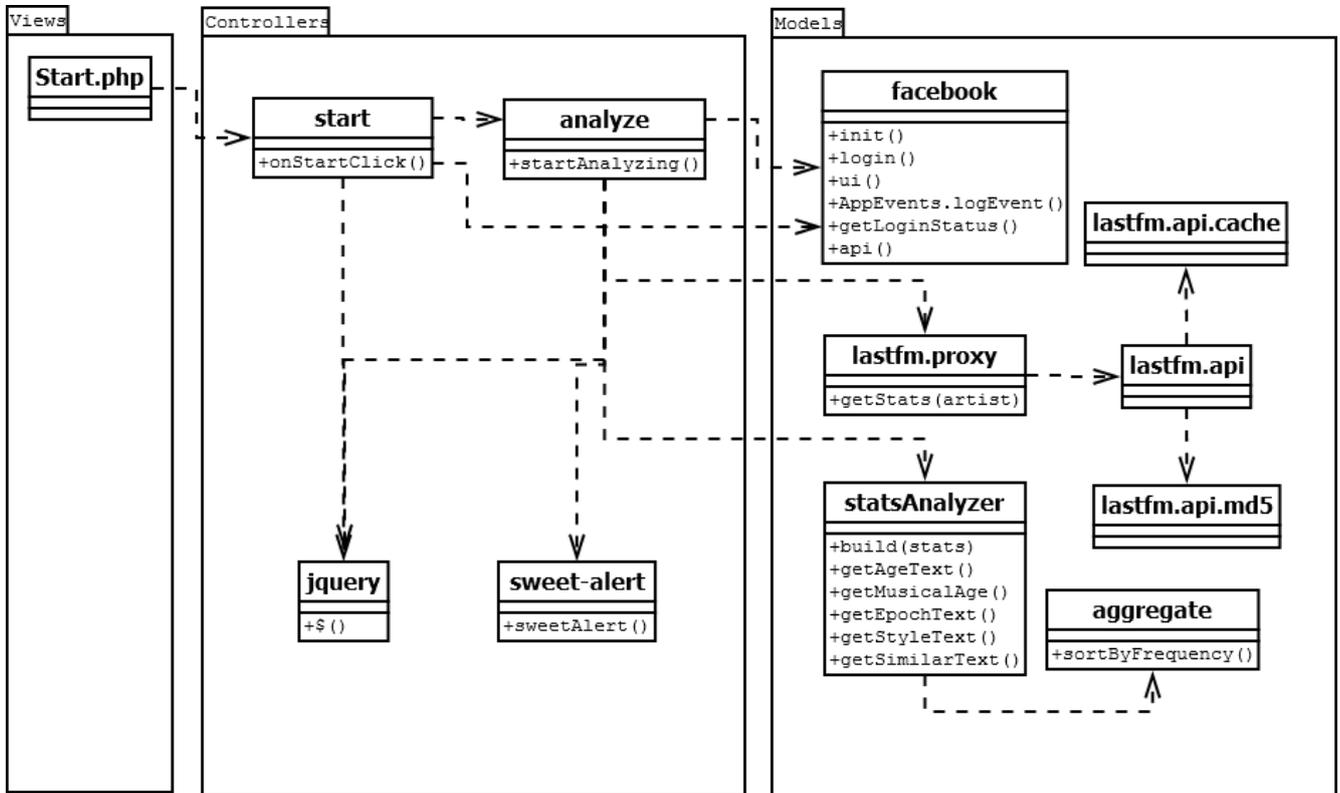


Ilustración 6 – Diagrama UML del cliente

No se describen los métodos del módulo `lastfm.api` por ser demasiado numerosos. Se pueden consultar en su repositorio (34).

4.2.3.2 Lógica de cada módulo

facebook

SDK oficial para acceder a la Graph API de Facebook.

lastfm.api

SDK no oficial para acceder a la API de LastFM.

lastfm.api.cache

Dependencia del SDK no oficial de LastFM. Caché interna que utiliza para almacenar temporalmente respuestas dadas por la API.

lastfm.api.md5

Dependencia del SDK no oficial de LastFM. Utilidad para calcular hashes MD5.

lastfm.proxy

Modelo que implementa un patrón proxy entre la base de datos y el SDK de LastFM.

statsanalyzer

Modelo que se ocupa de procesar las estadísticas de unos artistas datos para generar el texto personalizado que se le mostrará al usuario en los resultados.

aggregate

Modelo que se ocupa de aplicar operaciones de agregación. Dada una lista con elementos repetidos devuelve una lista de elementos no-repetidos ordenados por frecuencia de aparición.

sweet-alert

Biblioteca para mostrar ventanas modales.

jquery

Biblioteca para manipular el DOM y realizar llamadas AJAX.

start

Controlador principal por defecto. Recibe eventos de la vista Start.php y se ocupa de inicializar la API de Facebook y pedir permisos al usuario.

analyze

Controlador que se ocupa del proceso de análisis, desde la obtención de datos hasta mostrar los resultados personalizados.

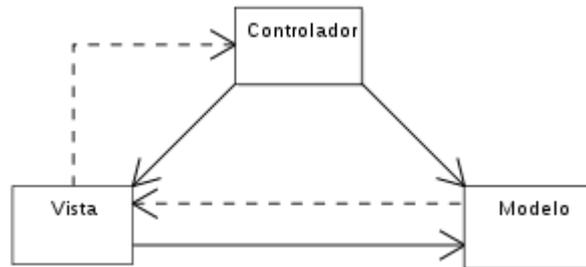
Start.php

Vista única que contiene todos los elementos HTML necesarios para la aplicación.

4.2.4 Patrones de diseño empleados

MVC

El siguiente diagrama UML representa el patrón de diseño MVC:



Como se puede apreciar en los diagramas UML, tanto del cliente como del servidor (ilustraciones 5 y 6), los módulos se han estructurado para establecer una separación entre los modelos, las vistas y los controladores. Con esto se consigue que los módulos estén menos acoplados unos con otros, de modo que puedan ser reutilizables.

Proxy

El siguiente diagrama UML representa el patrón de diseño Proxy:

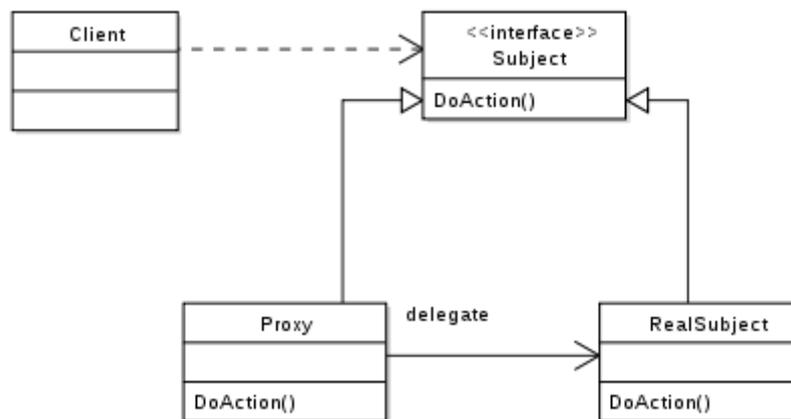


Ilustración 7- Diagrama UML del patrón Proxy

Como se puede observar la clase Proxy tiene la interfaz que realmente utiliza el cliente, pero delega parte de su funcionamiento a otra clase.

Se ha empleado este patrón para reducir el número de peticiones que se realizan a la API de LastFM. El siguiente diagrama de secuencia representa los mensajes intercambiados entre los módulos JavaScript Analyze y LastFMProxy, así como las llamadas AJAX que se realizan a la API REST de la aplicación y a la API de LastFM a través de su SDK.

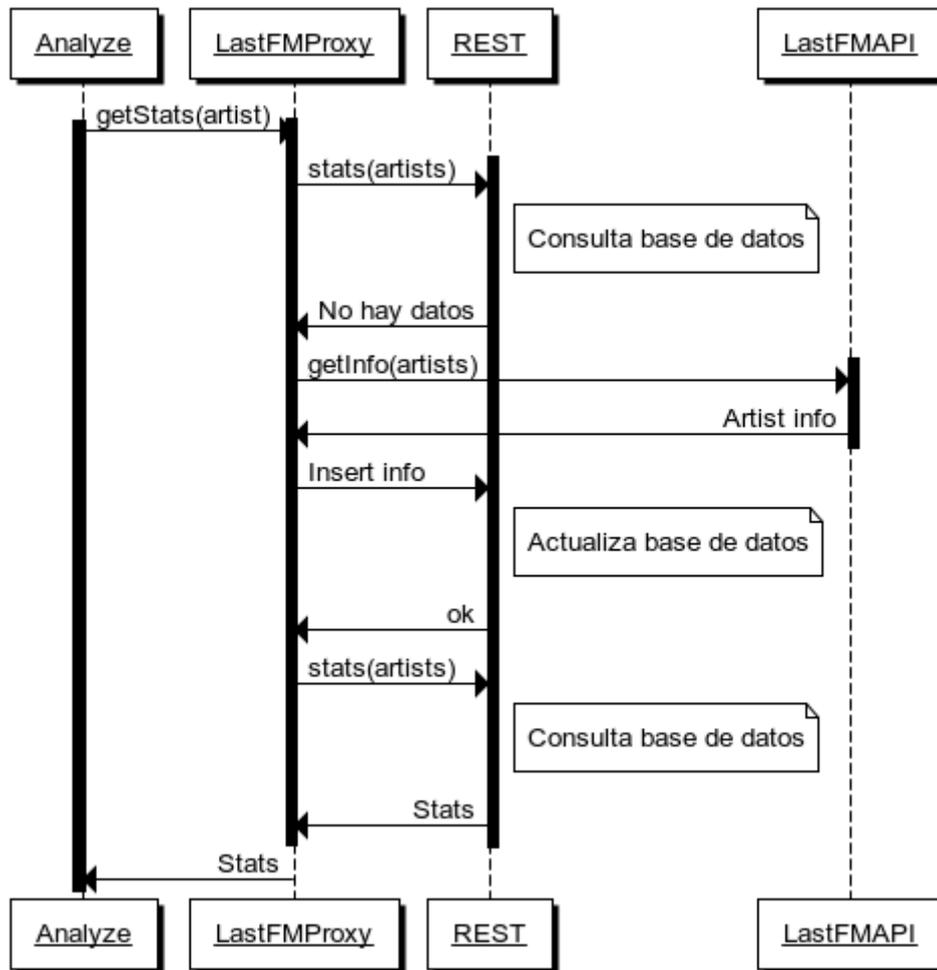


Ilustración 8 – Diagrama de secuencia de LastFMProxy

Se puede observar que el módulo Analyze pide estadísticas de un artista al módulo LastFMProxy. Éste, su vez, comprobará primero si existe información en la base de datos, consultando a la API REST.

Si no hay datos del artista en cuestión entonces se pedirá información a la API de LastFM y se introducirá en la base de datos también mediante la API REST.

Por último LastFMProxy vuelve a consultar a la API REST y esta vez sí devuelve las estadísticas del artista, que a su vez son devueltas a Analyze.

5. Implementación

5.1 Modelo de ciclo de vida aplicado

Dada la complejidad de la aplicación diseñada se optó por emplear un desarrollo iterativo (o incremental), que permitiera probar cada nueva funcionalidad antes de implementar la siguiente.

La siguiente ilustración muestra la evolución del número de cambios realizados en el repositorio de la aplicación a lo largo del tiempo:

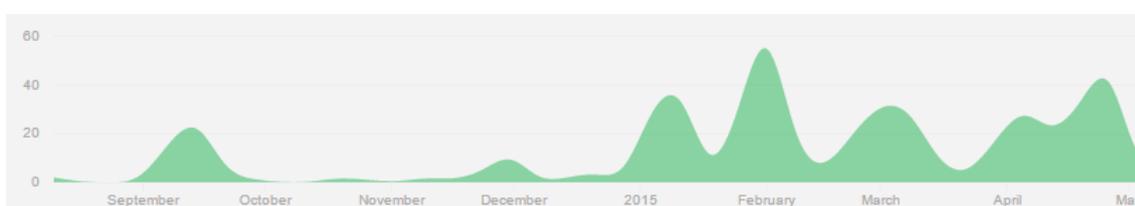


Ilustración 9 – Commits a lo largo del tiempo en el repositorio de GitHub

Se pueden observar a simple vista cómo los cambios se agrupan en periodos de tiempo que se corresponden con las iteraciones de desarrollo.

5.2 Servidor de base de datos

Aquí se define cómo se implementó el diseño de la base de datos anteriormente planeado.

5.2.1 Especificaciones del servidor

Dado que el presupuesto de este proyecto era nulo y que los servidores de base de datos gratuitos suelen tener limitaciones en cuanto a capacidad de almacenamiento y acceso, se optó por utilizar un servidor de base de datos externo (independiente del servidor web) ubicado en la Escuela Politécnica Superior de la Universidad Autónoma de Madrid.

El equipo utilizado como servidor tiene las siguientes especificaciones:

- Intel Core2 Quad a 2.66GHz
- 8GB de RAM
- Windows Vista, versión de 64bit.
- PostgreSQL, versión 9.3.5

5.2.2 Creación de tablas

Las tablas se crearon siguiendo el diseño planteado (ver Ilustración 4 – Diagrama Entidad-Relación). En el anexo 10.1 se puede consultar el script SQL que se utilizó para crear las tablas y aplicar las restricciones necesarias.

5.2.3 Diagrama de tablas

La siguiente figura muestra el diagrama de tablas generado a partir del script SQL anterior, utilizando el software MySQL Workbench.

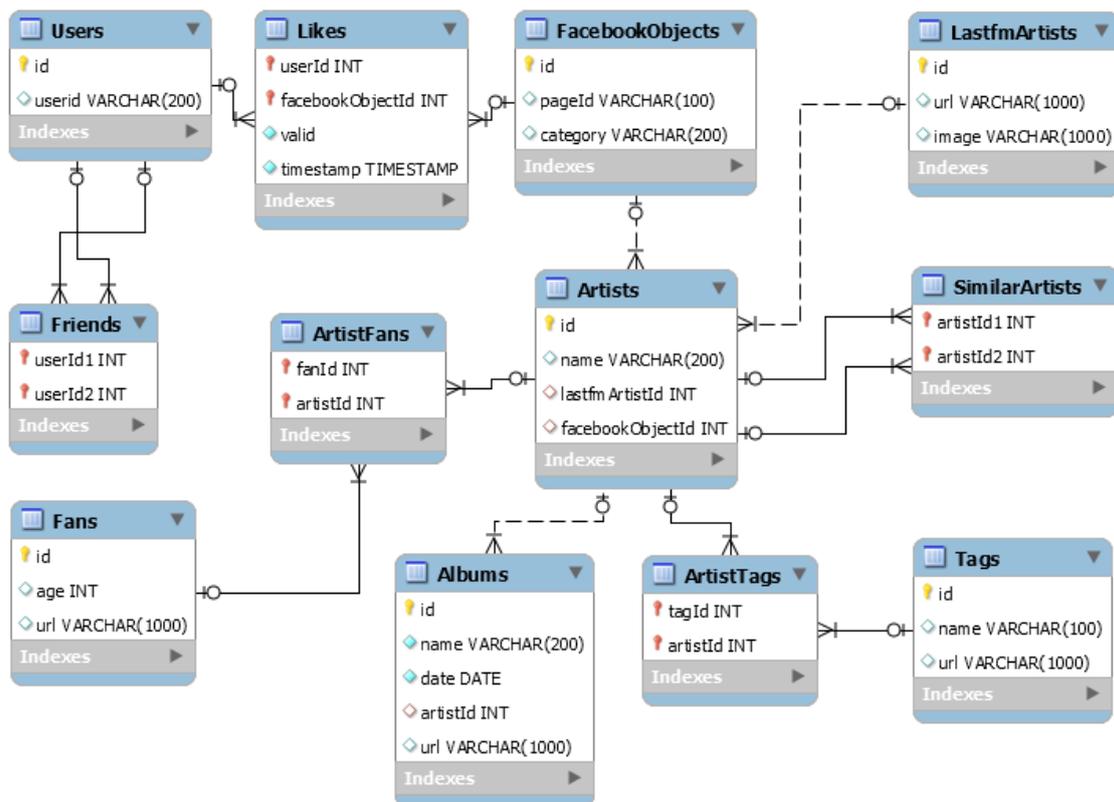


Ilustración 10 - Diagrama de tablas

Puede observarse un cambio con respecto al diseño planteado en el diagrama ER (ver Ilustración 4 – Diagrama Entidad-Relación); a la tabla Likes se le agregaron dos columnas más: “valid”, que es un booleano que indica si el like es válido en la actualidad o no, y “timestamp” que es el momento en el que el usuario dio “Me gusta” a ese objeto. Esto es interesante para poder usar la variable temporal en la posterior explotación de los datos.

5.3 Servidor web

En este apartado se describen las decisiones que se tomaron para implementar el subsistema del servidor web diseñado previamente.

5.3.1 Framework CodeIgniter

Para organizar los ficheros y facilitar la implementación del patrón MVC se utilizó el framework CodeIgniter (35).

Aquí se muestra la estructura de directorios que trae por defecto:

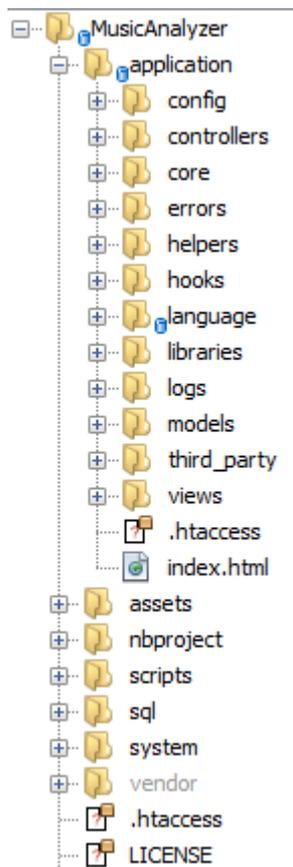


Ilustración 12 - Estructura de directorios de CodeIgniter

Como se puede observar, divide el código de la aplicación en los directorios “controllers”, “models” y “views” (entre otros).

Otra ventaja de CodeIgniter es que incluye utilidades, como la clase “Language” que se usó para internacionalizar la aplicación. En los anexos 10.2 y 10.3 se muestran los ficheros de lenguaje utilizados (castellano e inglés).

5.3.2 Hospedaje en Heroku

Para alojar la aplicación fue necesario encontrar un proveedor de hosting que ofreciera una plataforma compatible con los requerimientos no funcionales, en particular soportar PHP 5.6 y HTTPS.

Se escogió Heroku por encajar con estos requerimientos en su plan gratuito y por ser popular en la comunidad de desarrolladores de Facebook. En el propio sitio web de Heroku encontramos una guía de cómo utilizar sus servicios para implementar aplicaciones de Facebook (36). A continuación se listan las particularidades propias de este proveedor.

Soporte de HTTPS

Por defecto, toda aplicación creada en Heroku tiene disponible el acceso mediante HTTPS. Heroku utiliza para esto un certificado SSL compartido entre todas las aplicaciones gratuitas (37).

Plataforma Stack Cedar-14

Heroku permite seleccionar la plataforma sobre la cual queremos correr nuestra aplicación. En nuestro caso se escogió el stack “Cedar-14”, que se basa en Ubuntu 14.04 (38).

Uso de Composer

Toda aplicación en PHP que se hospede en Heroku deberá utilizar el gestor de dependencias Composer (23). Así, deberá haber un fichero “composer.json” en la raíz del proyecto que se usa para indicar al servidor qué bibliotecas deberá inyectar dinámicamente.

La siguiente ilustración muestra el contenido del fichero “composer.json” de nuestra aplicación:

```
{
  "require" : {
    "ext-mbstring": "*",
    "facebook/php-sdk-v4" : "4.0.*"
  }
}
```

Ilustración 13- Fichero composer.json

Como se puede observar, sólo se especifican dos dependencias: “ext-mbstring”, una biblioteca para manejar distintas codificaciones de cadenas, y “facebook/php-sdk-v4”, que es el SDK de Facebook en su versión 4.0.x.

Despliegue de aplicaciones

En Heroku, las aplicaciones se despliegan mediante el comando “push” de GIT, tal y como se muestra en la siguiente captura:

```
$ git push heroku master
Counting objects: 96, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (94/94), done.
Writing objects: 100% (96/96), 2.83 MiB | 28 KiB/s, done.
Total 96 (delta 6), reused 0 (delta 0)

----> PHP app detected
----> Bundling mcrypt version 2.5.8
----> Bundling Apache version 2.2.25
----> Bundling PHP version 5.3.27
----> Discovering process types
Procfile declares types -> (none)
Default types for PHP -> web

----> Compiled slug size: 25.1MB
----> Launching... done, v3
http://mytestapp02.herokuapp.com deployed to Heroku

To git@heroku.com:mytestapp02.git
 * [new branch]      master -> master
```

Ilustración 14- Despliegue usando GIT

Acceso a logs

Heroku permite consultar los logs de acceso y errores del servidor por consola, utilizando el comando “heroku logs”:

```
$ heroku logs ! tail
Your version of git is 1.9.2.. Which has serious security vulnerabilities.
More information here: https://blog.heroku.com/archives/2014/12/23/update_your_git_clients_
2015-05-30T23:15:11.209601+00:00 heroku[router]: at=info method=GET path="/" host=music-ana
2015-05-30T23:15:11.206576+00:00 applweb.1: 10.101.200.156 - - [30/May/2015:23:15:10 +0000]
2015-05-30T23:30:17.513455+00:00 heroku[router]: at=info method=GET path="/" host=music-ana
2015-05-30T23:30:17.511902+00:00 applweb.1: 10.216.128.228 - - [30/May/2015:23:30:17 +0000]
2015-05-30T23:45:11.715494+00:00 heroku[router]: at=info method=GET path="/" host=music-ana
2015-05-30T23:45:11.714280+00:00 applweb.1: 10.181.31.38 - - [30/May/2015:23:45:11 +0000]
2015-05-31T00:00:18.409466+00:00 heroku[router]: at=info method=GET path="/" host=music-ana
2015-05-31T00:00:18.406450+00:00 applweb.1: 10.180.41.39 - - [31/May/2015:00:00:17 +0000]
2015-05-31T00:15:11.788091+00:00 heroku[router]: at=info method=GET path="/" host=music-ana
2015-05-31T00:15:11.785250+00:00 applweb.1: 10.142.239.87 - - [31/May/2015:00:15:11 +0000]
```

Ilustración 15 - Logs de Heroku

Como se ve en la Ilustración 15 - Logs de Heroku, se muestra el momento en el que los usuarios (identificados por su IP y cabecera del navegador) realizaron cada petición, así como la respuesta que emitió el servidor.

Cronjobs

Heroku establece ciertos límites para su plan de hosting gratuito. Entre ellos, limita la ejecución de la aplicación a un único hilo contenedor o máquina virtual denominado “dyno”.

Un dyno entrará en estado de suspensión si pasados 30 minutos no se recibe ninguna petición. Si un usuario intenta acceder a la aplicación mientras el dyno está en suspensión, tardará alrededor de 20 segundos en cargar la página (tiempo que tarda en despertar el dyno más el tiempo de transmisión de la página en sí).

Para evitar este retraso en la carga de las páginas se utilizó el servicio gratuito de cron-job.org, que permite programar peticiones GET periódicas a sitios web para prevenir que entren en suspensión.

En la siguiente ilustración se muestra la configuración utilizada para mantener despierto el dyno de la aplicación:

Edit cronjob

Address

<http://music-analyzer.herokuapp.com>

Requires HTTP authentication

Username Password

Schedule

Every 30 minute(s)

Every day at 0 : 00

Every 1. of the month at 0 : 00

User-defined

Ilustración 16- Programación de cronjob

5.3.3 Integración con Facebook

En este apartado se describen los mecanismos utilizados para integrar la aplicación con el entorno y funcionalidades de Facebook.

5.3.3.1 Facebook Developers

Facebook pone a disposición de los desarrolladores el sitio web developers.facebook.com, desde el cual se pueden gestionar todas las aplicaciones.

Sección principal

La siguiente ilustración muestra la página principal de administración de la aplicación, desde la cual se puede acceder al resto de secciones.

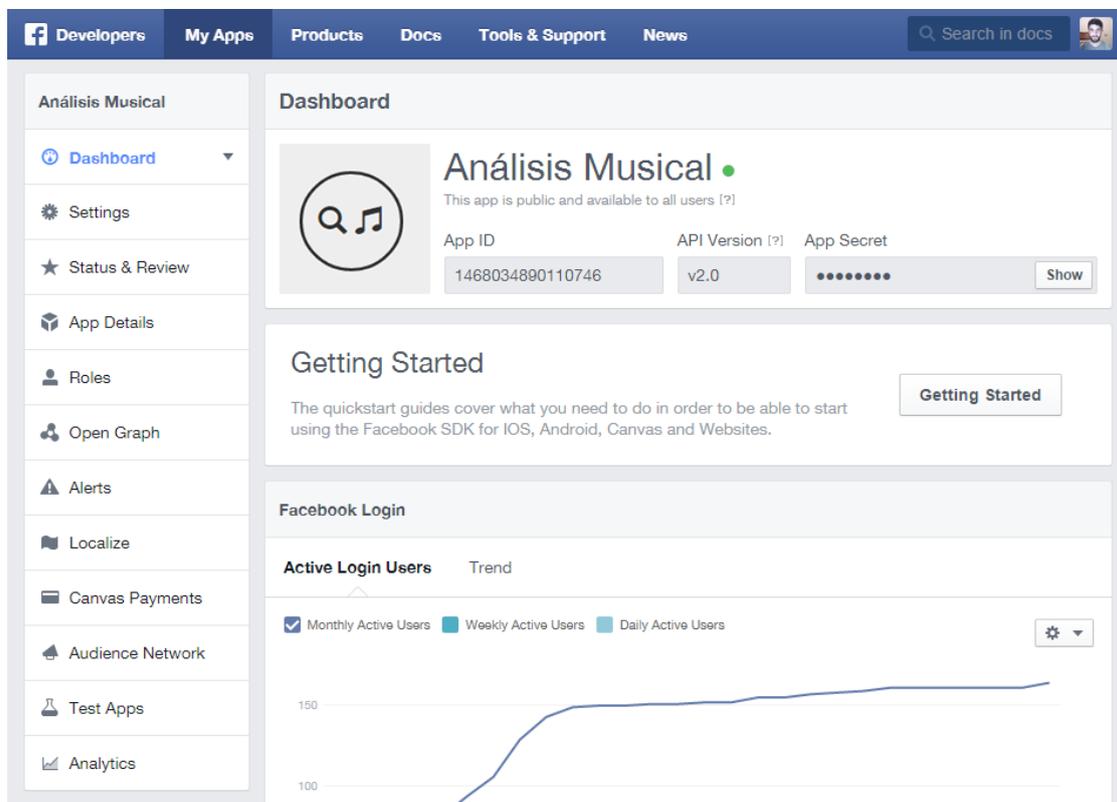
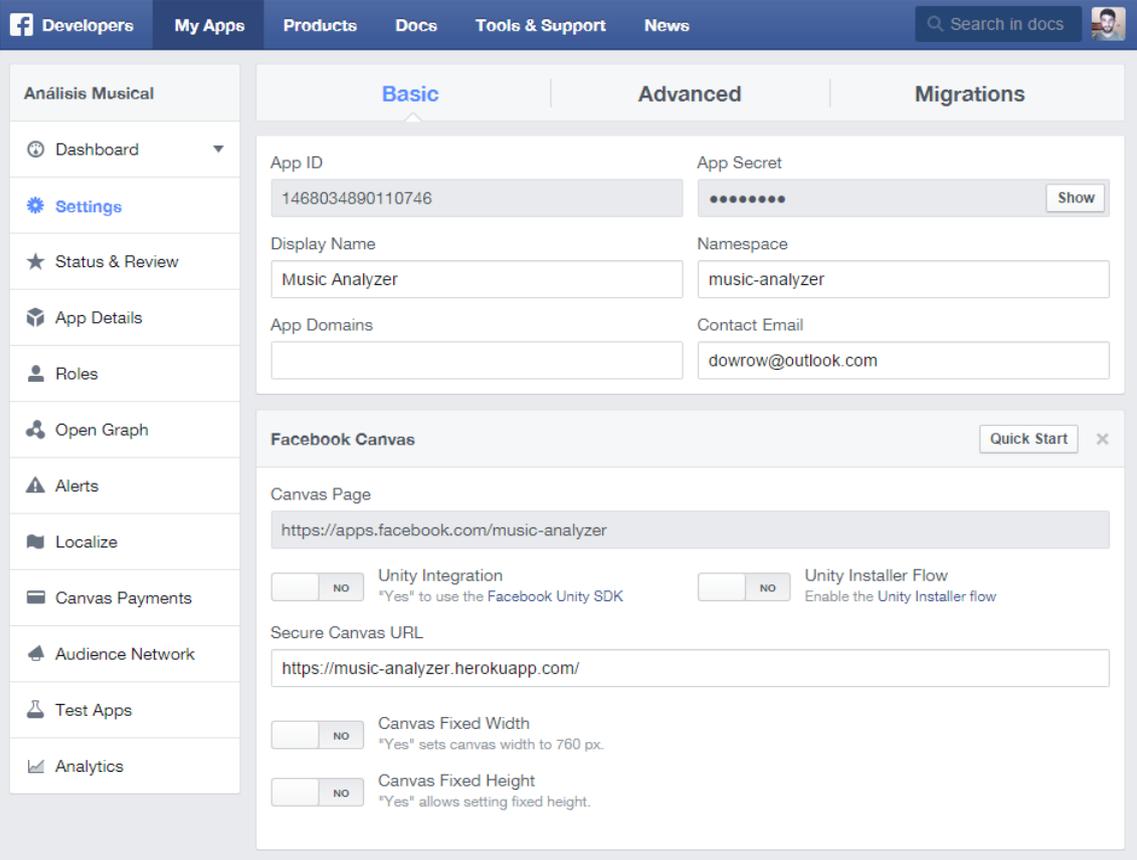


Ilustración 17 - Página principal del panel de control de Facebook Developers

Sección de ajustes

En la sección “Settings” podemos configurar los ajustes más importantes. En la Ilustración 18 - Apartado de ajustes de la aplicación en Facebook Developers, se muestra la configuración utilizada para nuestra aplicación.



The screenshot shows the Facebook Developers interface. At the top, there are navigation tabs: Developers, My Apps, Products, Docs, Tools & Support, and News. A search bar is on the right. On the left, a sidebar menu lists various settings categories: Dashboard, Settings (highlighted), Status & Review, App Details, Roles, Open Graph, Alerts, Localize, Canvas Payments, Audience Network, Test Apps, and Analytics. The main content area is titled 'Basic' and contains the following configuration fields:

- App ID:** 1468034890110746
- App Secret:** [Redacted] with a 'Show' button.
- Display Name:** Music Analyzer
- Namespace:** music-analyzer
- App Domains:** [Empty field]
- Contact Email:** dowrow@outlook.com

Below these fields is a section for 'Facebook Canvas' with a 'Quick Start' button. It includes:

- Canvas Page:** https://apps.facebook.com/music-analyzer
- Unity Integration:** [NO] *Yes* to use the Facebook Unity SDK
- Unity Installer Flow:** [NO] Enable the Unity Installer flow
- Secure Canvas URL:** https://music-analyzer.herokuapp.com/
- Canvas Fixed Width:** [NO] *Yes* sets canvas width to 760 px.
- Canvas Fixed Height:** [NO] *Yes* allows setting fixed height.

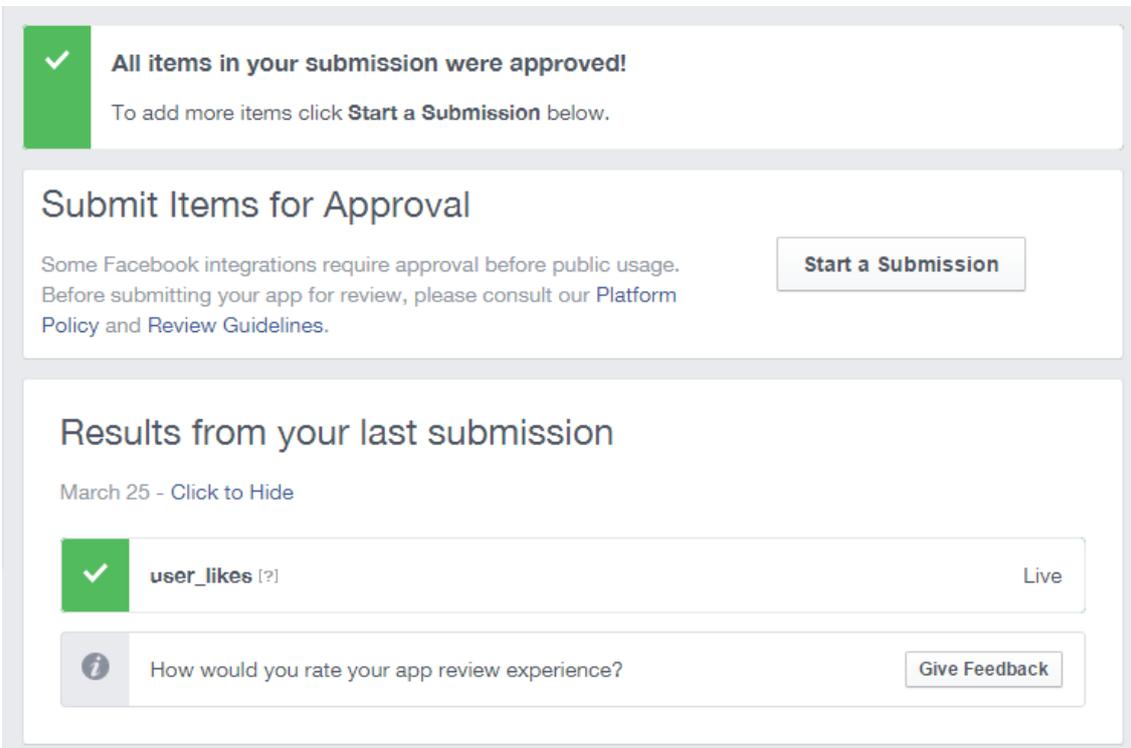
Ilustración 18 - Apartado de ajustes de la aplicación en Facebook Developers

Se especifica que se trata de una aplicación embebida (Facebook Canvas), la URL donde se aloja y el tamaño que deberá tener el marco contenedor. En nuestro caso no se especifican un alto ni un ancho fijos, ya que la aplicación se diseñó para adaptarse a cualquier tamaño de pantalla.

Solicitud de permisos

En el apartado “Status & Review” podemos solicitar el acceso a determinados permisos para nuestra aplicación. En nuestro caso fue necesario pedir “user_likes” para poder acceder a los likes del usuario.

El proceso de solicitud requirió explicar para qué se utilizarán los permisos, adjuntar una política de privacidad (que puede consultarse en el anexo 10.11) y enviar pantallazos de la aplicación. La aprobación de estos permisos se hace de forma manual por Facebook y tardó 5 días en completarse.



The screenshot displays a confirmation message at the top: "All items in your submission were approved!" with a green checkmark icon. Below this, it says "To add more items click **Start a Submission** below." The main section is titled "Submit Items for Approval" and includes a "Start a Submission" button. A note explains that some Facebook integrations require approval before public usage. The "Results from your last submission" section shows a submission from March 25, which is hidden. A table entry shows the "user_likes" permission is approved (green checkmark) and is currently "Live". At the bottom, there is a feedback prompt: "How would you rate your app review experience?" with a "Give Feedback" button.

✓ **All items in your submission were approved!**
To add more items click **Start a Submission** below.

Submit Items for Approval

Some Facebook integrations require approval before public usage. Before submitting your app for review, please consult our [Platform Policy and Review Guidelines](#).

Start a Submission

Results from your last submission

March 25 - [Click to Hide](#)

✓	user_likes [?]	Live
---	-----------------------	------

i How would you rate your app review experience? **Give Feedback**

Ilustración 19 - Aprobación de acceso a likes

5.3.3.2 Facebook Analytics for apps

Recientemente Facebook ha habilitado un servicio de analíticas para aplicaciones. Podemos acceder mediante la URL www.facebook.com/analytics/.

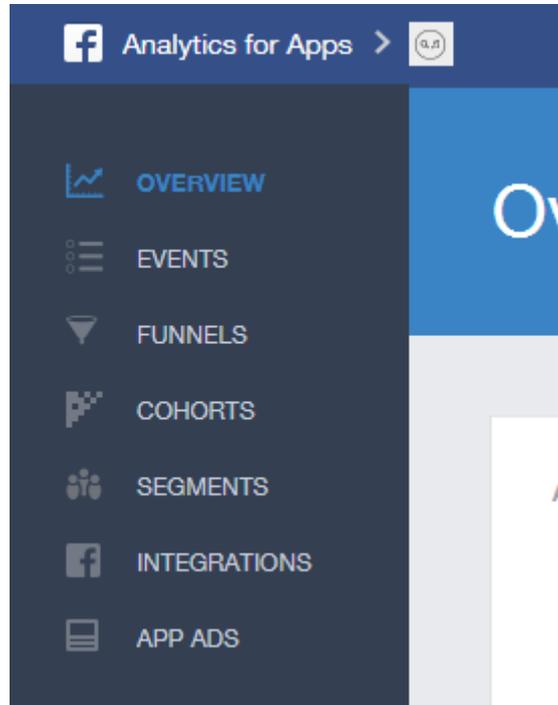


Ilustración 20 - Menú principal de Facebook Analytics for Apps

Este servicio nos da la posibilidad de almacenar información sobre eventos que suceden cuando los usuarios utilizan la aplicación, tales como clicks en un botón.

Permite también mostrar estadísticas sobre el número de inicios de sesión, notificaciones enviadas, fuentes de visitantes, etc. En definitiva, constituye una herramienta ideal para seguir el progreso del uso de nuestra aplicación.

5.4 Cliente

En los siguientes apartados se define cómo se implementó el diseño del subsistema cliente.

5.4.1 Pantallas

Las distintas pantallas fueron implementadas como una sola vista con componentes que cambian. Se implementó en HTML y CSS, haciendo uso del framework Bootstrap (39) para conseguir un diseño que se adaptara al mayor número posible de pantallas.

Pantalla principal

La siguiente captura muestra la pantalla principal:



Ilustración 21 - Pantalla principal

El logo está compuesto por iconos que vienen incluidos en Bootstrap rodeados de un borde circular. El resto de estilos son los aplicados por defecto por el framework.

Pantalla de análisis

La siguiente captura muestra la pantalla de análisis:

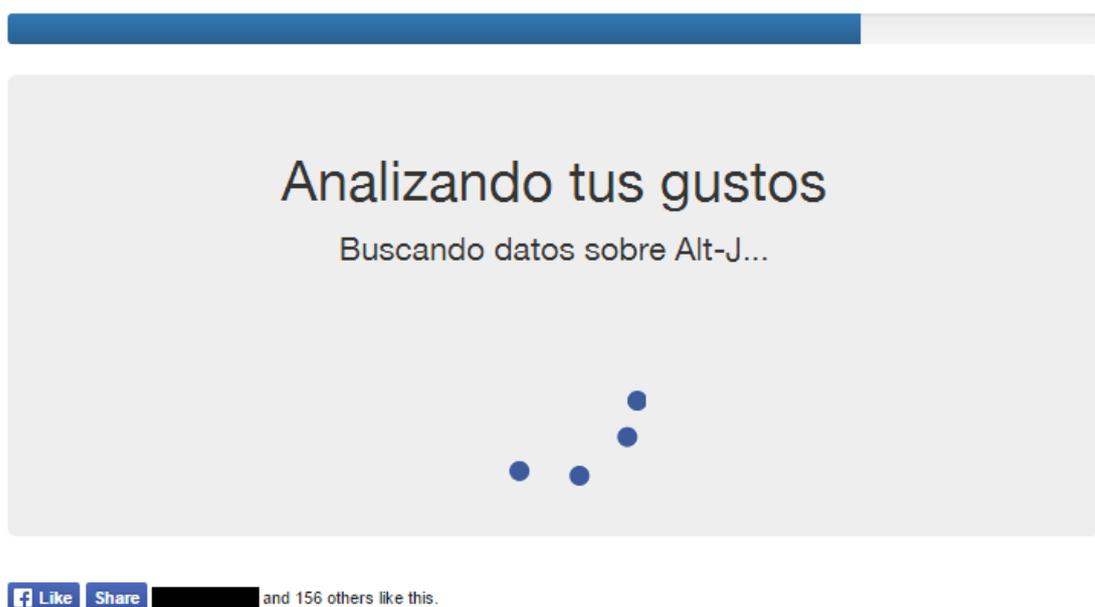


Ilustración 22 - Pantalla de análisis

Se agregó una animación GIF de carga y una barra de progreso que se implementó usando la clase “progress-bar” de Bootstrap.

Pantalla de resultados

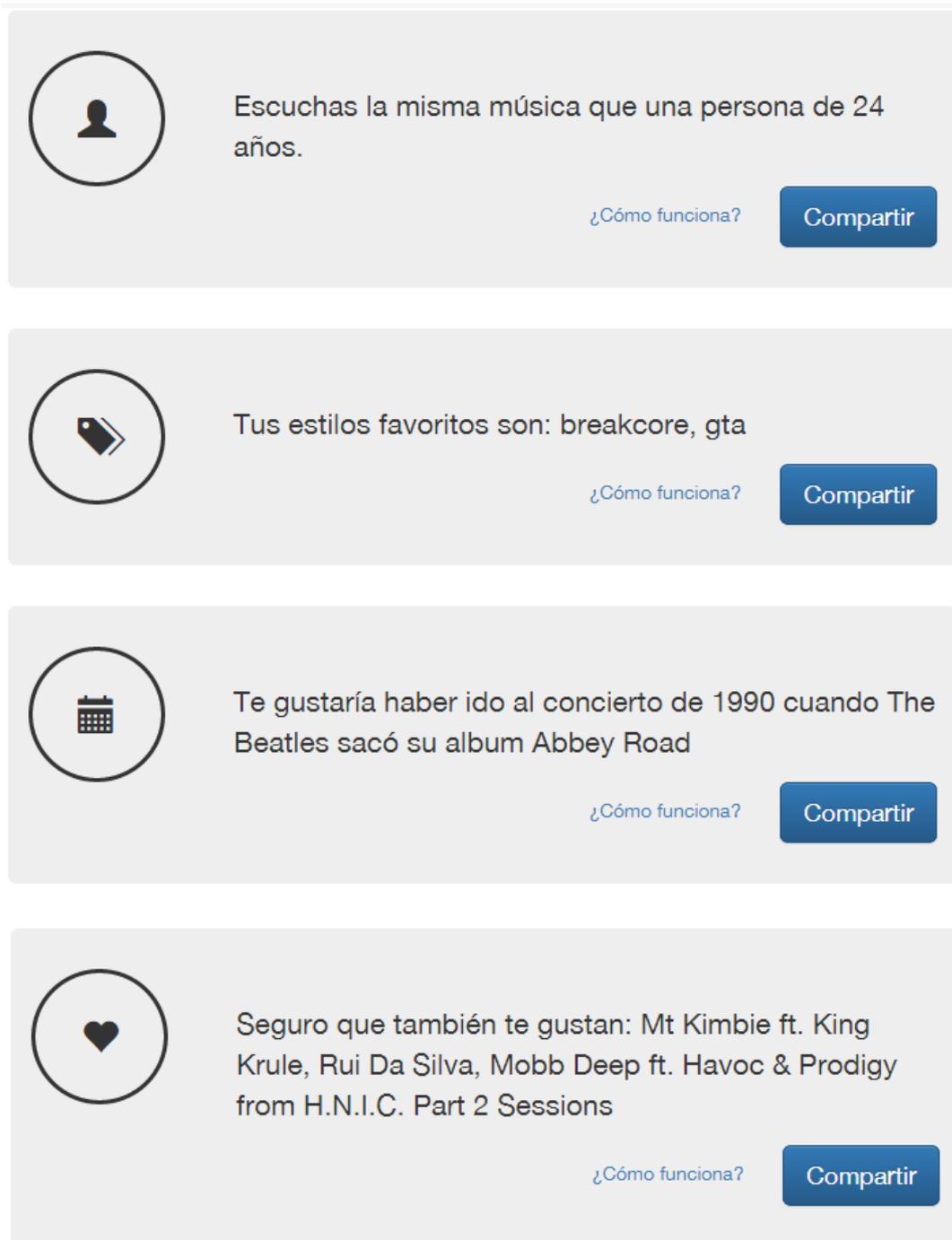
La siguiente captura muestra la parte superior de la pantalla de resultados:



Ilustración 23- Pantalla de resultados. Resultados de amigos

Esta gráfica comparativa fue implementada usando la biblioteca HighCharts (40).

La siguiente captura muestra parte inferior de la pantalla de resultados:



Like Share 157 people like this. Be the first of your friends.

Ilustración 24 - Pantalla de resultados. Resultados del usuario.

Los dibujos utilizados a modo de “placa” son también iconos de Bootstrap con un borde circular.

5.4.2 Bibliotecas JavaScript

RequireJS

Biblioteca para la definición de módulos e inyección de dependencias. Se utilizó para suplir la carencia de clases en JavaScript y así conseguir encapsulación (41).

HighCharts

Biblioteca gratuita (para uso no comercial) para dibujar gráficas de todo tipo (40).

6. Pruebas

A continuación se describen las pruebas efectuadas a lo largo del proceso de desarrollo. Nótese que la aplicación utiliza un gran número de bibliotecas y SDKs desarrollados por terceros que ya han sido probados, por lo cual se dio más importancia a cubrir la integración y los casos de uso de esos módulos que a sus pruebas unitarias.

6.1 Pruebas unitarias

Se implementaron pruebas de caja negra automatizadas para todos los métodos públicos de los modelos desarrollados en JavaScript. Para ello se utilizó la biblioteca QUnit (42).

En el anexo 10.4 se puede encontrar el código de las pruebas unitarias desarrolladas. A continuación se muestra el resultado de los tests a través de la interfaz HTML generada por QUnit.

Unit tests		
<input type="checkbox"/> Hide passed tests <input type="checkbox"/> Check for Globals <input type="checkbox"/> No try-catch Filter: <input type="text"/> <input type="button" value="Go"/>		
QUnit 1.18.0; Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36		
Tests completed in 1259 milliseconds. 14 assertions of 14 passed, 0 failed.		
1. Aggregate sort list (1)	Rerun	1 ms
2. Aggregate sort empty (1)	Rerun	0 ms
3. StatsAnalyzer build (1)	Rerun	1 ms
4. StatsAnalyzer get age text (1)	Rerun	1 ms
5. StatsAnalyzer get epoch text (1)	Rerun	1 ms
6. StatsAnalyzer get style text (1)	Rerun	0 ms
7. StatsAnalyzer get similar text (1)	Rerun	0 ms
8. StatsAnalyzer build empty (1)	Rerun	0 ms
9. StatsAnalyzer get age text empty (1)	Rerun	0 ms
10. StatsAnalyzer get epoch text empty (1)	Rerun	0 ms
11. StatsAnalyzer get style text empty (1)	Rerun	1 ms
12. StatsAnalyzer get similar text empty (1)	Rerun	1 ms
13. LastFMProxy get Cher stats (1)	Rerun	0 ms
14. LastFMProxy get unknown stats (1)	Rerun	0 ms

Ilustración 25 - Pruebas unitarias con QUnit

6.2 Pruebas de integración

Para probar los controladores desarrollados se realizaron pruebas de integración. En particular se efectuaron pruebas sobre el controlador REST, que integra los modelos DatabaseManager y Stats.

Estas pruebas se realizaron enviando peticiones POST y GET mediante AJAX a la API REST de la aplicación, y se automatizaron usando QUnit.

En el anexo 10.5 se puede encontrar el código de las pruebas de integración desarrolladas. A continuación se muestra el resultado de los tests a través de la interfaz HTML generada por QUnit.



The screenshot displays the QUnit test runner interface. At the top, it says "Integration tests". Below this, there are control options: "Hide passed tests", "Check for Globals", and "No try-catch", each with an unchecked checkbox. To the right is a "Filter:" input field and a "Go" button. The environment information is shown as "QUnit 1.18.0; Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.2357.81 Safari/537.36". The test results summary states: "Tests completed in 2383 milliseconds. 8 assertions of 8 passed, 0 failed." Below this, a list of 8 tests is shown, each with a "Rerun" link and a duration in milliseconds.

Test Name	Duration
1. REST API up and running (1) Rerun	1 ms
2. REST API insert album (1) Rerun	0 ms
3. REST API get unkown stats (1) Rerun	1 ms
4. REST API insert tag (1) Rerun	1 ms
5. REST API insert artist (1) Rerun	0 ms
6. REST API insert fan (1) Rerun	1 ms
7. REST API get Cher stats (1) Rerun	1 ms
8. REST API insert similar (1) Rerun	1 ms

6.3 Pruebas de casos de uso

Aquí se describen las pruebas realizadas para comprobar que se implementaron correctamente los casos de uso analizados.

6.3.1 Análisis de gustos

El proceso seguido para cubrir este caso de uso puede encontrarse en el anexo 10.6. El caso de uso fue probado satisfactoriamente, incluyendo el flujo alternativo.

6.3.2 Compartir un resultado

El proceso seguido para cubrir este caso de uso puede encontrarse en el anexo 10.7. El caso de uso fue probado satisfactoriamente.

6.3.3 Invitar a un amigo

El proceso seguido para cubrir este caso de uso puede encontrarse en el anexo 10.8. El caso de uso fue probado satisfactoriamente.

7. Resultados

A continuación se describen los métodos utilizados para monitorizar el uso de la aplicación, así como los datos recogidos a lo largo del mes de mayo.

7.1 Seguimiento por Facebook Analytics

Se utilizó la herramienta Facebook Analytics para monitorizar la actividad de los usuarios dentro de la aplicación. Esto se hizo mediante la captura de eventos. En nuestro caso se asignó un evento a cada enlace o botón de la aplicación:

Tabla 4 - Eventos capturados por Facebook Analytics

Evento	Significado
HowAge	El usuario hizo click sobre el enlace “¿Cómo funciona?” del resultado “Edad musical”.
HowEpoch	El usuario hizo click sobre el enlace “¿Cómo funciona?” del resultado “Concierto ideal”.
HowSimilar	El usuario hizo click sobre el enlace “¿Cómo funciona?” del resultado “Artistas similares”.
HowStyle	El usuario hizo click sobre el enlace “¿Cómo funciona?” del resultado “Géneros favoritos”.
InviteFriends	El usuario hizo click sobre el botón “Invitar a amigos”.
ShareAge	El usuario hizo click sobre el botón “Compartir” del resultado “Edad Musical”.
ShareEpoch	El usuario hizo click sobre el botón “Compartir” del resultado “Concierto ideal”.
ShareSimilar	El usuario hizo click sobre el botón “Compartir” del resultado “Artistas similares”.
ShareStyle	El usuario hizo click sobre el botón “Compartir” del resultado “Géneros favoritos”.
Start	El usuario hizo click sobre el botón “Empezar” de la aplicación.

En el anexo 10.9 se detallan las métricas y estadísticas que genera esta herramienta. De su uso se dedujo que el mecanismo que más viralidad produjo fue el relacionado con la componente social (en nuestro caso la gráfica comparativa).

7.2 Conjunto de datos obtenido

Pasado un mes desde la publicación de la aplicación se contaron las tuplas existentes en las tablas más representativas de la base de datos:

Tabla	Número de tuplas
Users	170
FacebookObjects	33445
Likes	32224
Artists	34005
LastfmArtists	33880

Ilustración 26 - Número de tuplas por tabla

Se utilizó la herramienta Gephi (43) para representar un grafo de las amistades entre los usuarios de Facebook, aplicando los algoritmos Expansion y ForceAtlas:

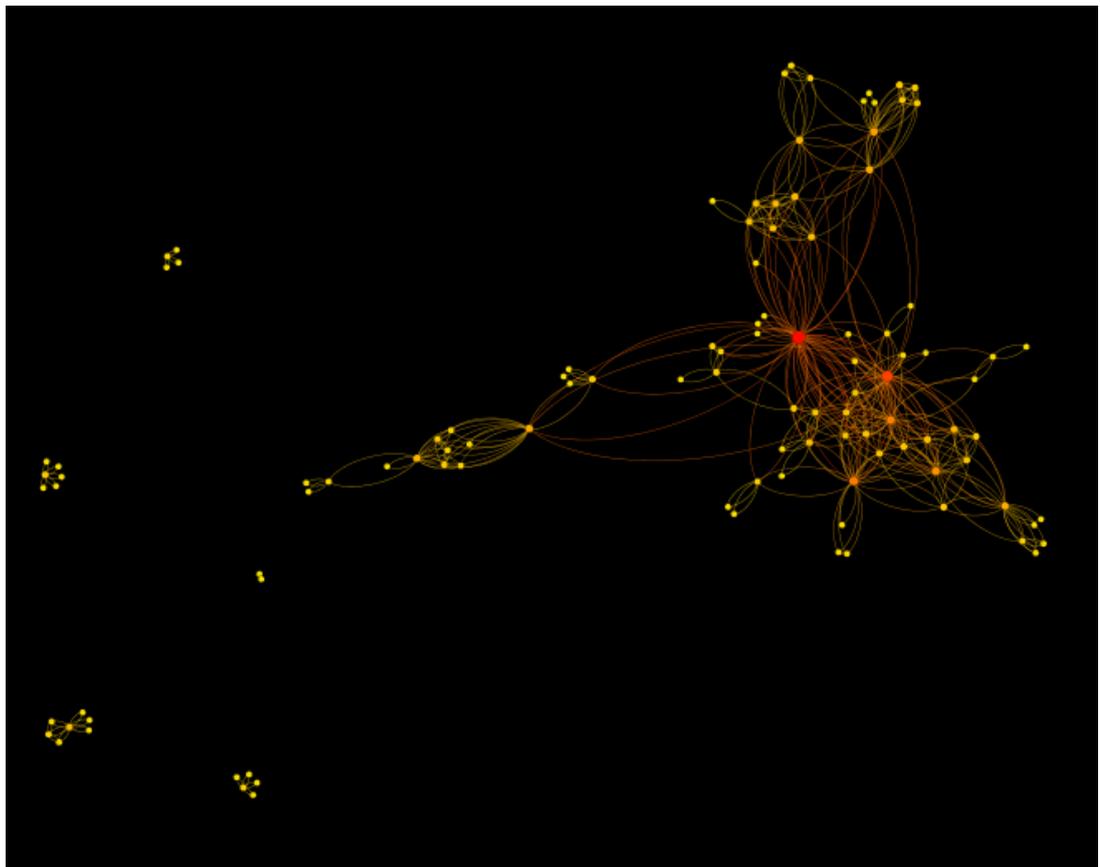


Ilustración 27 - Grafo de relaciones de amistad entre usuarios

Además se crearon varias vistas en la base de datos que son útiles para consultar estadísticas. Se pueden encontrar en el anexo 10.10.

8. Conclusión y trabajos futuros

En un mes se han logrado recoger datos de 170 usuarios gracias a la aplicación de mecanismos de gamificación en una aplicación de Facebook. Mediante el análisis con la herramienta Facebook Analytics se ha descubierto que el mecanismo que más viralidad produjo fue el relacionado con la componente social (en nuestro caso la gráfica comparativa), seguido de los resultados personalizados “edad musical” y “géneros favoritos”.

En cuanto a la parte formativa, durante el desarrollo de esta aplicación se han aplicado los conocimientos aprendidos durante la carrera, en particular han sido especialmente relevantes los relacionados con el área de Ingeniería de Software (metodologías, diseño orientado a objetos, uso de diagramas, etc.) a la hora de planificar el proyecto, y también los relacionados con el área de Sistemas Informáticos (PHP, SQL, HTML, CSS y JavaScript) a la hora de implementarlo.

También se han adquirido nuevos conocimientos útiles, en particular sobre: Gamificación, uso de repositorios GIT, manejo de APIs, integración de aplicaciones dentro del ecosistema de Facebook, utilización del framework CodeIgniter y desarrollo sobre la plataforma de Heroku.

Respecto al trabajo futuro sobre este proyecto, podrían hacerse diversas mejoras con el objetivo de incrementar la aceptación por parte de los usuarios. Algunas de las mejoras más relevantes serían:

- Incrementar el número de artistas que analiza la aplicación. Actualmente está limitado a 10 artistas porque la API de LastFM tarda demasiado en responder.
- Utilizar la nueva versión de la API de Facebook que permite acceder a la música que escuchan los usuarios en Spotify de manera ordenada (aún en versión beta) (31).
- Extender el dominio de la aplicación para analizar también películas y/o libros.

Finalmente, como resultado de este Trabajo de Fin de Grado, se ha determinado que el código implementado estará disponible en el siguiente repositorio de GitHub:

<https://github.com/dowrow/AnalisisMusical>

Usuario: dowrow

Repositorio: AnalisisMusical

9. Referencias

1. IRG. [En línea] Mayo de 2015. <http://ir.ii.uam.es/>.
2. Number of monthly active Facebook users. [En línea] Mayo de 2015. <http://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/>.
3. World population. [En línea] Mayo de 2015. <http://www.worldometers.info/world-population/>.
4. What a big data business model looks like. [En línea] Mayo de 2015. <https://www.facebook.com/notes/european-innovation-academy/what-a-big-data-business-model-looks-like/307219186060417>.
5. Facebook's privacy settings. [En línea] <https://www.facebook.com/help/325807937506242/>.
6. Gabe Zichermann, Christopher Cunningham. *Gamification by Design - Implementing Game Mechanics in Web and Mobile Apps*. s.l. : O'Reilly Media, 2011.
7. Facebook developers for Android. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/android>.
8. Facebook Developers for iOS. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/ios>.
9. Facebook Javascript SDK - Quickstart. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/javascript/quickstart/v2.2>.
10. Comparativa entre tipos de aplicaciones en Facebook. [En línea] Mayo de 2015. <http://www.adweek.com/socialtimes/canvas-connect-websites-best/261427>.
11. Atlas API Docs. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/reference/atlas-api/campaign-management>.
12. Marketing API Docs. [En línea] Mayo de 2015. https://developers.facebook.com/docs/marketing-apis?locale=es_ES.

13. Graph API overview. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/graph-api/overview/>.
14. Facebook's PHP SDK Docs. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/reference/php/4.0.0>.
15. MediaWiki. [En línea] Mayo de 2015. <http://en.wikipedia.org/w/api.php>.
16. MusicBrainz homepage. [En línea] Mayo de 2015. <https://musicbrainz.org>.
17. Scrobble. *Urban Dictionary*. [En línea] Mayo de 2015. <http://www.urbandictionary.com/define.php?term=Scrobble>.
18. LastFM API. [En línea] Mayo de 2015. <http://www.last.fm/api>.
19. Ubuntu 14.04. [En línea] Mayo de 2015. <http://releases.ubuntu.com/14.04/>.
20. Apache. [En línea] Mayo de 2015. <http://httpd.apache.org/>.
21. PostgreSQL homepage. [En línea] Mayo de 2015. <http://www.postgresql.org/>.
22. PHP 5.4. [En línea] Mayo de 2015. http://php.net/releases/5_4_0.php.
23. Composer homepage. [En línea] Mayo de 2015. <https://getcomposer.org/>.
24. *Netbeans*. [En línea] Mayo de 2015. <https://netbeans.org/>.
25. pgAdminIII. [En línea] Mayo de 2015. <http://www.pgadmin.org/>.
26. *Github*. [En línea] Mayo de 2015. <https://github.com/>.
27. *Node.js*. [En línea] Mayo de 2015. <https://nodejs.org/>.
28. User endpoint. *Graph API reference*. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/graph-api/reference/user>.
29. User/likes endpoint. *Graph API reference*. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/graph-api/reference/user/likes>.
30. Recomendación basada en relaciones sociales. [En línea] Mayo de 2015. <http://oferarazy.com/PDF/Social%20Relationships%20in%20Recommender%20Systems.pdf>.
31. Music endpoint beta. *Open Graph reference*. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/opengraph/music>.

32. User/activities endpoint. *Graph API reference*. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/graph-api/reference/user/activities/>.
33. Arquitectura SPA vs tradicional. *MSDN*. [En línea] Mayo de 2015. <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>.
34. SDK de LastFM para Javascript. *Github*. [En línea] Mayo de 2015. <https://github.com/fxb/javascript-last.fm-api>.
35. *CodeIgniter*. [En línea] Mayo de 2015. <http://www.codeigniter.com/>.
36. Guía para desarrolladores de Facebook. *Heroku*. [En línea] Mayo de 2015. <https://devcenter.heroku.com/articles/facebook>.
37. SSL endpoint. *Heroku*. [En línea] Abril de 2015. <https://devcenter.heroku.com/articles/ssl-endpoint>.
38. Cedar14 Stack. *Heroku*. [En línea] Abril de 2015. <https://devcenter.heroku.com/articles/cedar>.
39. *Bootstrap homepage*. [En línea] Febrero de 2015. <http://getbootstrap.com/>.
40. *Highcharts homepage*. [En línea] Abril de 2015. <http://www.highcharts.com/>.
41. *RequireJS homepage*. [En línea] Mayo de 2015. <http://requirejs.org/>.
42. *QUnit homepage*. [En línea] Mayo de 2015. <https://qunitjs.com/>.
43. *Gephi*. [En línea] Mayo de 2015. <http://gephi.github.io/>.
44. Facebook's Javascript SDK Docs. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/javascript>.
45. User/music endpoint. *Graph API reference*. [En línea] Mayo de 2015. <https://developers.facebook.com/docs/graph-api/reference/user/music/>.
46. Getting started with PHP. *Heroku*. [En línea] Mayo de 2015. <https://devcenter.heroku.com/articles/getting-started-with-php#introduction>.
47. SSL endpoint. *Heroku*. [En línea] Mayo de 2015. <https://devcenter.heroku.com/articles/ssl-endpoint>.

10. Anexos técnicos

10.1 Script de creación de tablas

```
--
-- DB creation script
--
CREATE TABLE LastfmArtists (
    id serial PRIMARY KEY,
    url varchar(1000) UNIQUE,
    image varchar(1000)
);

CREATE TABLE FacebookObjects (
    id serial PRIMARY KEY,
    pageId varchar(100) UNIQUE,
    category VARCHAR(200)
);

CREATE TABLE Artists (
    id serial PRIMARY KEY,
    name varchar (200) UNIQUE,
    lastfmArtistId int REFERENCES LastfmArtists(id),
    facebookObjectId int REFERENCES FacebookObjects(id)
);

CREATE TABLE Albums (
    id serial PRIMARY KEY,
    name varchar (200) NOT NULL,
    date date NOT NULL,
    artistId int REFERENCES Artists(id),
    url varchar(1000) UNIQUE
);

CREATE TABLE Tags (
    id serial PRIMARY KEY,
    name varchar (100) UNIQUE,
    url varchar(1000) UNIQUE
);

CREATE TABLE ArtistTags (
    tagId int REFERENCES Tags(id),
    artistId int REFERENCES Artists(id),
    PRIMARY KEY (tagId, artistId)
);
```

```

CREATE TABLE Fans (
    id serial PRIMARY KEY,
    age int default 0,
    url varchar(1000) UNIQUE
);

CREATE TABLE ArtistFans (
    fanId int REFERENCES Fans(id),
    artistId int REFERENCES Artists(id),
    PRIMARY KEY (fanId, artistId)
);

CREATE TABLE SimilarArtists (
    artistId1 int REFERENCES Artists(id),
    artistId2 int REFERENCES Artists(id),
    PRIMARY KEY (artistId1, artistId2)
);

CREATE TABLE Users (
    id serial PRIMARY KEY,
    userid varchar(200) UNIQUE
);

CREATE TABLE Likes (
    userId int REFERENCES Users(id),
    facebookObjectId int REFERENCES FacebookObjects(id),
    valid BOOLEAN NOT NULL DEFAULT FALSE,
    timestamp TIMESTAMP NOT NULL DEFAULT NOW(),
    PRIMARY KEY (userId, facebookObjectId)
);

CREATE TABLE Friends (
    userId1 int REFERENCES Users(id),
    userId2 int REFERENCES Users(id),
    PRIMARY KEY (userId1, userId2)
);

```

Ilustración 28 - Script SQL de creación de tablas

10.2 Fichero de idioma: Castellano

```
<?php

$lang['start_title'] = "Análisis Musical";
$lang['start_description'] = "Descubre lo que tu música dice de ti.";
$lang['start_button'] = "Empezar";
$lang['share_image'] = "https://music-analyzer.herokuapp.com/assets/img/share_image_es.jpg";

// ES
$lang['analyze_title'] = "Analizando tus gustos";
$lang['analyze_description'] = "Descubre lo que tu música dice de ti.";
$lang['analyze_share_image'] = "https://music-analyzer.herokuapp.com/assets/img/share_image_es.jpg";

$lang['analyze_status'] = "Buscando datos sobre";
$lang['analyze_status_album'] = "Buscando álbumes de";
$lang['analyze_status_fan'] = "Buscando fans de";
$lang['analyze_status_tag'] = "Determinando el estilo de";
$lang['analyze_status_similar'] = "Buscando artistas similares a";

$lang['analyze_error_title'] = "Parece que hubo un problema";
$lang['analyze_error_status'] = "No encontramos música entre tus likes.";

$lang['analyze_results_title'] = "Tus resultados";
$lang['analyze_friend_results_title'] = "Resultados de amigos";

$lang['analyze_result_age_1'] = "Escuchas la misma música que una persona de ";
$lang['analyze_result_age_2'] = " años.";
$lang['analyze_result_no_age'] = "No pudimos calcular tu edad musical.";

$lang['analyze_result_style'] = "Tus estilos favoritos son: ";
$lang['analyze_result_no_style'] = "Eres tan hipster que no pudimos determinar qué géneros te gustan.";

$lang['analyze_result_epoch_1'] = "Te gustaría haber ido al concierto de ";
$lang['analyze_result_epoch_2'] = " cuando ";
$lang['analyze_result_epoch_3'] = " sacó su album ";
$lang['analyze_result_no_epoch'] = "La música que te gusta es atemporal.";

$lang['analyze_result_similar'] = "Seguro que también te gustan: ";
$lang['analyze_result_no_similar'] = "No encontramos ningún artista similar a los que te gustan.";

$lang['share_button'] = "Compartir";
$lang['how_button'] = "¿Cómo funciona?";

$lang['how_age'] = "Buscamos los fans de cada artista que te gusta y promediamos sus edades.";
$lang['how_style'] = "Buscamos los géneros de cada artista que te gusta y los agregamos.";
$lang['how_epoch'] = "Buscamos el album más antiguo de los artistas que te gustan.";
$lang['how_similar'] = "Utilizamos las recomendaciones de Last.fm.";

$lang['your_musical_age'] = "Tu edad musical";
$lang['musical_age'] = "Edad musical";
$lang['you_are_the_first'] = "Eres el primero de tus amigos en usar esta app. ¡Invita a tus amigos y compara";
$lang['invite_friends'] = "Invitar a amigos";
$lang['you'] = "Tú";
$lang['try_music_analyzer'] = "Descubre lo que tu música dice de ti. ¡Prueba la app Análisis Musical!";

$lang['notification_message'] = "ha actualizado su edad musical. ¡Compara vuestros resultados!";
```

Ilustración 29 - Fichero de lenguaje castellano

10.3 Fichero de idioma: Inglés

```
<?php

// Strings used in start.php view
$lang['start_title'] = "Music Analyzer";
$lang['start_description'] = "Discover what your music says about you.";
$lang['start_button'] = "Start";
$lang['share_image'] = "https://music-analyzer.herokuapp.com/assets/img/share_image_en.jpg";

// EN
$lang['analyze_title'] = "Analyzing your likes";
$lang['analyze_description'] = "Discover what your music says about you.";
$lang['analyze_share_image'] = "https://music-analyzer.herokuapp.com/assets/img/share_image_en.jpg";

$lang['analyze_status'] = "Collecting data about";
$lang['analyze_status_album'] = "Searching albums by";
$lang['analyze_status_fan'] = "Searching fans of";
$lang['analyze_status_tag'] = "Determining the style of";
$lang['analyze_status_similar'] = "Searching artists similar to";

$lang['analyze_error_title'] = "Looks like there's a problem";
$lang['analyze_error_status'] = "We were unable to find music among your likes.";

$lang['analyze_results_title'] = "Your results";
$lang['analyze_friend_results_title'] = "Your friends results";

$lang['analyze_result_age_1'] = "You listen to the same music as a ";
$lang['analyze_result_age_2'] = "-year-old person.";
$lang['analyze_result_no_age'] = "We weren't able to calculate your musical age.";

$lang['analyze_result_style'] = "Your favorite genres are: ";
$lang['analyze_result_no_style'] = "You are such a hipster we weren't able to determine what your favourite genres are.";

$lang['analyze_result_epoch_1'] = "You wish you were to the concert of ";
$lang['analyze_result_epoch_2'] = " when ";
$lang['analyze_result_epoch_3'] = " released the album ";
$lang['analyze_result_no_epoch'] = "The music you like is atemporal.";

$lang['analyze_result_similar'] = "You will surely like this too: ";
$lang['analyze_result_no_similar'] = "We weren't able to find any artist similar to the ones you like.";

$lang['share_button'] = "Share";
$lang['how_button'] = "How does it work?";

$lang['how_age'] = "We find the fans of every artist you like and calculate their average age.";
$lang['how_style'] = "We find the genres of every artist you like and aggregate them.";
$lang['how_epoch'] = "We find the oldest album released by the artists you like.";
$lang['how_similar'] = "We use Last.fm recommendations.";

$lang['your_musical_age'] = "Your musical age";
$lang['musical_age'] = "Musical age";
$lang['you_are_the_first'] = "You are the first of your friends to try this app. Invite friends and compare your results!";
$lang['invite_friends'] = "Invite friends";
$lang['you'] = "You";
$lang['try_music_analyzer'] = "Discover what your music says about you. Try Music Analyzer!";

$lang['notification_message'] = "has updated his musical age. Compare your results!";
```

Ilustración 30 - Fichero de lenguaje inglés

10.4 Pruebas unitarias automatizadas con QUnit

```
define(['statsanalyzer', 'aggregate', 'LastFMProxy'], function(StatsAnalyzer,
    Aggregate, LastFMProxy) {

    /**
     * LastFM Proxy tests
     */
    var dummyCallback = function() {
    };
    LastFMProxy.getStats('112915798740840', 'Cher', dummyCallback,
        dummyCallback, dummyCallback, dummyCallback,
        dummyCallback, function(stats) {
        QUnit.test("LastFMProxy get Cher stats", function(assert) {
            assert.ok(stats.length > 1, "Passed!");
        });
    });
    LastFMProxy.getStats('0', 'asdasdasdsdf', dummyCallback, dummyCallback,
        dummyCallback, dummyCallback, dummyCallback,
        function(stats) {
            QUnit.test("LastFMProxy get unknown stats", function(assert) {
                assert.ok(stats === '1', "Passed!");
            });
        });
    });

    /**
     * Aggregate tests
     */
    var list1 = ['d', 'a', 'c', 'c', 'b', 'd', 'c', 'b', 'd', 'd'];
    var sorted1 = [['d', 4], ['c', 3], ['b', 2], ['a', 1]];
    QUnit.test("Aggregate sort list", function(assert) {
        assert.equal(Aggregate.sortByFrequency(list1).join(),
            sorted1.join(), "Passed!");
    });
    QUnit.test("Aggregate sort empty", function(assert) {
        assert.equal(Aggregate.sortByFrequency([]).join(), '', "Passed!");
    });

    /**
     * StatsAnalyzer tests
     */
    var mockupStats = {};
    mockupStats['Melendi'] = {
        averageFanAge: 1,
        firstAlbum: [{name: 'album1', date: '2013-01-25T00:11:02+0000'}],
        similar: [{name: 'Bob Marley'}, {name: 'Justin Bieber'},
            {name: 'King Africa'}],
        tags: [{name: 'Gitaneo'}, {name: 'Cani'}, {name: 'Reggae'}]
    };
});
```

```

mockupStats['Melendi2'] = {
    averageFanAge: 2,
    firstAlbum: [{name: 'album2', date: '2013-01-25T00:11:02+0000'}],
    similar: [{name: 'Bob Marley'}, {name: 'Justin Bieber'},
              {name: 'King Africa'}],
    tags: [{name: 'Gitaneo'}, {name: 'Cani'}, {name: 'Reggae'}]
};
mockupStats['Melendi3'] = {
    averageFanAge: 3,
    firstAlbum: [{name: 'album3', date: '2013-01-25T00:11:02+0000'}],
    similar: [{name: 'Bob Marley'}, {name: 'Justin Bieber'},
              {name: 'King Africa'}],
    tags: [{name: 'Gitaneo'}, {name: 'Cani'}, {name: 'Reggae'}]
};

QUnit.test("StatsAnalyzer build", function(assert) {
    assert.ok(StatsAnalyzer.build(mockupStats), "Passed!");
});
QUnit.test("StatsAnalyzer get age text", function(assert) {
    assert.equal(StatsAnalyzer.getAgeText(), 'part_12part_2', "Passed!");
});
QUnit.test("StatsAnalyzer get epoch text", function(assert) {
    assert.equal(StatsAnalyzer.getEpochText(),
                 'part12013part2Melendipart3album1', "Passed!");
});
QUnit.test("StatsAnalyzer get style text", function(assert) {
    assert.equal(StatsAnalyzer.getStyleText(),
                 'part1Gitaneo, Cani, Reggae', "Passed!");
});
QUnit.test("StatsAnalyzer get similar text", function(assert) {
    assert.equal(StatsAnalyzer.getSimilarText(),
                 'part1Bob Marley, Justin Bieber, King Africa', "Passed!");
});
QUnit.test("StatsAnalyzer build empty", function(assert) {
    assert.ok(StatsAnalyzer.build({}), "Passed!");
});
QUnit.test("StatsAnalyzer get age text empty", function(assert) {
    assert.equal(StatsAnalyzer.getAgeText(), 'No age', "Passed!");
});
QUnit.test("StatsAnalyzer get epoch text empty", function(assert) {
    assert.equal(StatsAnalyzer.getEpochText(), 'No epoch', "Passed!");
});
QUnit.test("StatsAnalyzer get style text empty", function(assert) {
    assert.equal(StatsAnalyzer.getStyleText(), 'no tags', "Passed!");
});
QUnit.test("StatsAnalyzer get similar text empty", function(assert) {
    assert.equal(StatsAnalyzer.getSimilarText(), 'no similar', "Passed!");
});
});

```

Ilustración 31 - Pruebas unitarias con QUnit

10.5 Pruebas de integración automatizadas con Qunit

```
define(['jquery'], function($) {

    /**
     * Integration tests for REST API
     */
    var url = 'https://music-analyzer.herokuapp.com/rest';

    $.get(url, function(res) {
        QUnit.test("REST API up and running", function(assert) {
            assert.ok(res === "REST API OK", "Passed!");
        });
    });

    $.get(url + '/stats?artist=asdasdasdasd&pageid=0', function(res) {
        QUnit.test("REST API get unkown stats", function(assert) {
            var obj = JSON.parse(res);
            assert.ok(obj === 0, "Passed!");
        });
    });

    $.get(url + '/stats?artist=Cher&pageid=112915798740840', function(res) {
        QUnit.test("REST API get Cher stats", function(assert) {
            var obj = JSON.parse(res);
            assert.ok(obj.averageFanAge && obj.firstAlbum && obj.similar &&
                obj.tags, "Passed!");
        });
    });

    $.post(url + '/insertartist', {name: 'TestArtist', url: 'TestURL',
        pageid: '000000', image: 'TestImage'}, function(res) {
        QUnit.test("REST API insert artist", function(assert) {
            assert.equal(res, 'ok', "Passed!");
        });
    });

    $.post(url + '/insertalbum', {artist: 'TestArtist', album: 'TestAlbum',
        url: 'TestURL', date: '2013-01-25T00:11:02+0000'},
        function(res) {
            QUnit.test("REST API insert album", function(assert) {
                assert.equal(res, 'ok', "Passed!");
            });
        });
});
```

```

$.post(url + '/inserttag', {artist: 'TestArtist', name: 'TestTag',
  url: 'TestURL'}, function(res) {
  QUnit.test("REST API insert tag", function(assert) {
    assert.equal(res, 'ok', "Passed!");
  });
});

$.post(url + '/insertfan', {artist: 'TestArtist', url: 'TestURL',
  age: '0'}, function(res) {
  QUnit.test("REST API insert fan", function(assert) {
    assert.equal(res, 'ok', "Passed!");
  });
});

$.post(url + '/insertsimilar', {artist: 'TestArtist',
  similar: 'TestSimilar', url: 'TestURL', image: 'TestImage'},
function(res) {
  QUnit.test("REST API insert similar", function(assert) {
    assert.equal(res, 'ok', "Passed!");
  });
});
});

```

Ilustración 32 - Pruebas de integración con QUnit

10.6 Prueba de caso de uso C1: Análisis de gustos

1. El usuario hace click en el botón “Empezar”.



Ilustración 33 – Pantalla principal

2. El usuario autoriza a la aplicación a acceder a sus datos.



Ilustración 34 – Diálogo de autorización de permisos

3. La aplicación muestra una barra de progreso mientras se analizan sus gusto

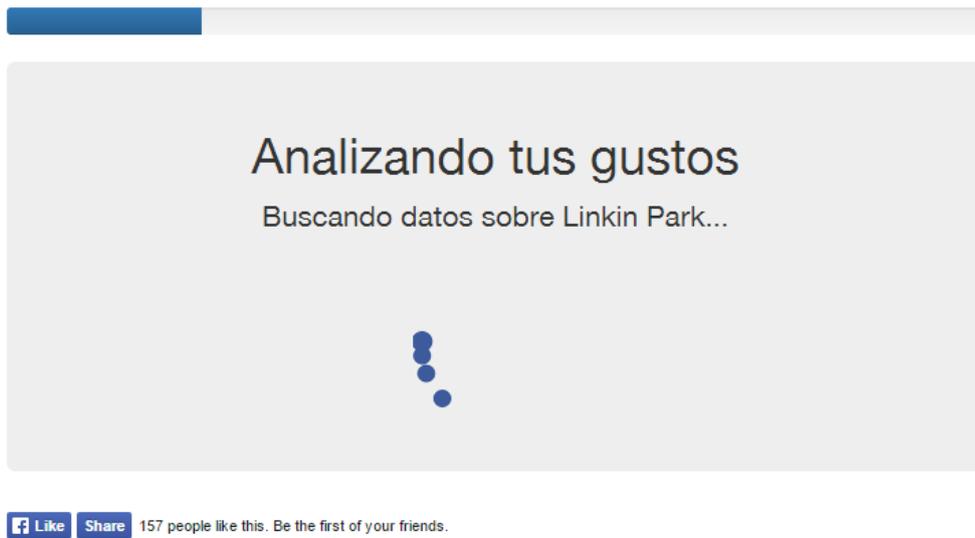


Ilustración 35 – Pantalla de análisis

4. Una vez finalizado el análisis la aplicación muestra los resultados.



Ilustración 36- Pantalla de resultados

Flujo alternativo 2.b: Si el usuario no tiene likes musicales la aplicación muestra un mensaje informativo

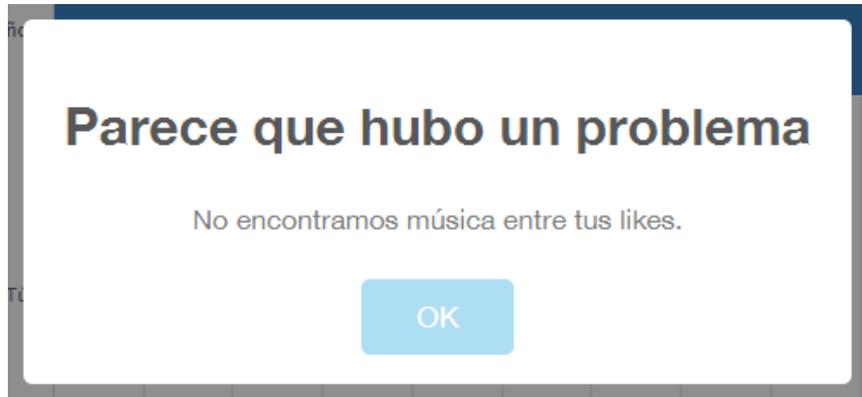


Ilustración 37 – Mensaje informativo

Postcondición: Los amigos del usuario recibirán una notificación incitándoles a comparar sus resultados:

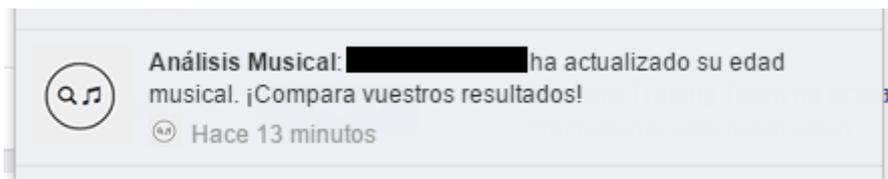


Ilustración 38 – Notificación

10.7 Prueba de caso de uso C2: Compartir un resultado

1. El usuario hace click en el botón “Compartir” del resultado que quiere compartir.

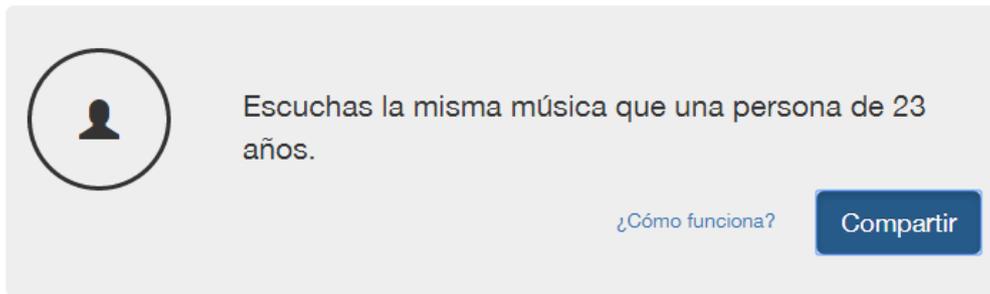


Ilustración 39 – Compartir un resultado

2. El sistema muestra un diálogo para que el usuario escriba un comentario que publicar junto al enlace a la aplicación.



Ilustración 40 – Diálogo de publicación

3. El usuario hace click en el botón “Compartir” del diálogo.

Postcondición: Se publica en el muro del usuario un enlace a la aplicación, el texto que mostraba su resultado personalizado.



Ilustración 41- Resultado publicado en el muro de un usuario

10.8 Prueba de caso de uso C3: Invitar a amigos

1. usuario hace click en el botón “Invitar a amigos”.

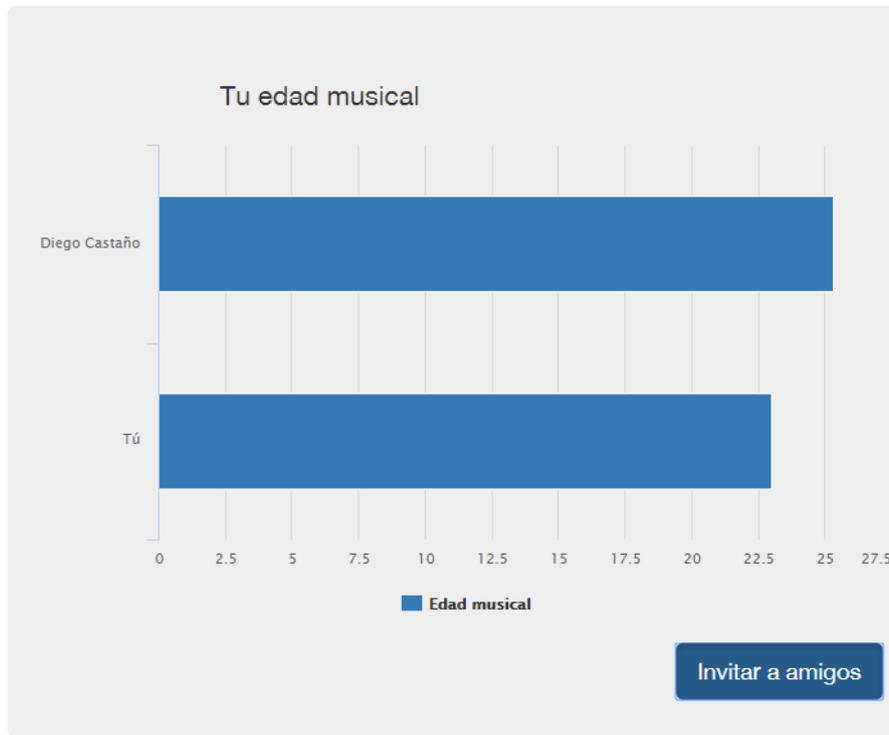


Ilustración 42 – Botón de invitar a amigos

2. La aplicación envía una invitación a los amigos que seleccione el usuario.



Ilustración 43 – Envío de invitaciones a amigos

Postcondición: El usuario amigo recibirá una invitación con un enlace a la aplicación.



Ilustración 44 - Solicitud recibida

10.9 Seguimiento por medio de Facebook Analytics for Apps

A continuación se muestran métricas de interés obtenidas mediante la herramienta Facebook Analytics for Apps.

La siguiente gráfica muestra la evolución del número de usuarios a lo largo de mayo:

Facebook Login Activity > Total Login Users ▾

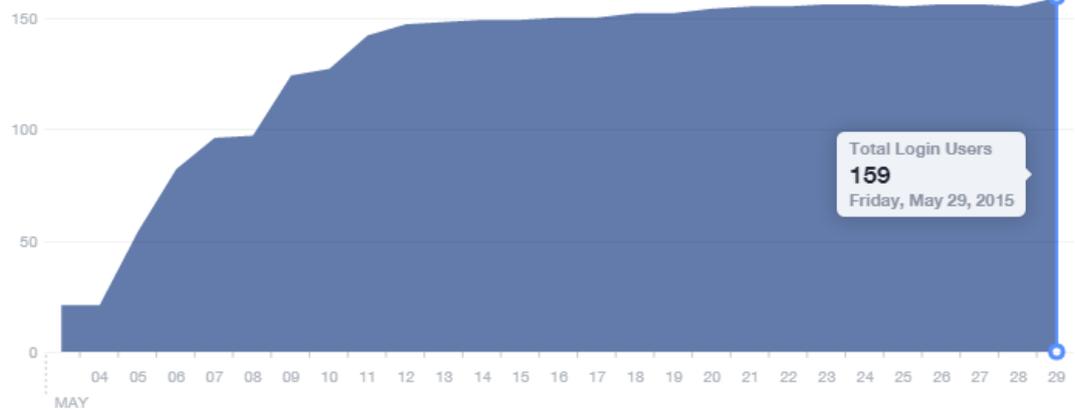


Ilustración 45 - Evolución del número de usuarios

Aquí se representan el número de notificaciones enviadas a lo largo del mes de mayo:

App Notifications > App Notifications Sent ▾



Ilustración 46 - Notificaciones enviadas

Podemos ver también el número de accesos que se realizaron a nuestra aplicación proviniendo de resultados compartidos en muros de los usuarios:

Positive Engagement with Stories > Clicks ▾

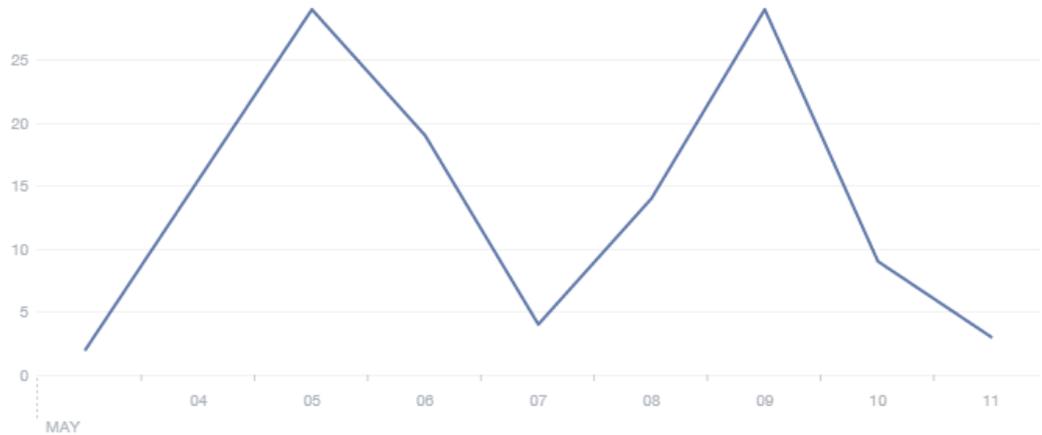


Ilustración 47 - Accesos desde resultados publicados en el muro

La herramienta también permite listar el número de eventos de cada tipo que se produjeron:

Name	Count	Period/Period	Today's Count	Unique Users ⓘ
HowAge	28	+ 100%	0	24
HowEpoch	20	+ 100%	0	16
HowSimilar	17	+ 100%	0	13
HowStyle	28	+ 100%	0	24
InviteFriends	31	+ 100%	2	12
ShareAge	20	+ 100%	2	11
ShareEpoch	6	+ 100%	0	6
ShareSimilar	4	+ 100%	0	3
ShareStyle	13	+ 100%	0	11
Start	553	+ 100%	10	202

Ilustración 48 - Número de eventos por tipo

10.10 Vistas creadas en la base de datos

A continuación se muestra el script SQL utilizado para crear las vistas:

```
create view user__n_likes as
-- user, number of likes
select id, users.userid as f_id, count(facebookobjectid) as n_likes
from users JOIN likes
on users.id = likes.userid
group by id
order by n_likes desc;

create view like__n_users as
-- page, category, n_users
select facebookobjects.id, facebookobjects.pageid, count(likes.userid) as n_users
from likes join facebookobjects on facebookobjects.id=likes.facebookobjectid
group by facebookobjects.id
order by n_users desc;

create view user__category__n_category as
-- user, category, n_category
select users.id, users.userid as f_id, category, count(*) as n_category
from users join likes on users.id = likes.userid
join facebookobjects on facebookobjects.id=likes.facebookobjectid
group by users.id, category
order by n_category desc;

create view user__n_diff_categories as
-- user, n_different_categories
select users.id, users.userid as f_id, count(distinct category) as n_diff_category
from users join likes on users.id = likes.userid
join facebookobjects on facebookobjects.id=likes.facebookobjectid
group by users.id
order by n_diff_category desc;
```

Ilustración 49 - Creación de vistas en base de datos

La vista `user__n_likes` muestra el número de likes almacenados para cada usuario, la vista `like__n_users` muestra el número de usuarios a los cuales les gusta un cada objeto, la vista `user__category__n_category` muestra cuántos objetos de cada categoría le gustan a cada usuario y la vista `user__n_diff_categories` muestra el número de categorías distintas que le gustan a un usuario.

10.11 Política de privacidad

A continuación se muestra el documento de política de privacidad asociado a la aplicación.

Versión en castellano:

Política de privacidad

1. *Qué información se recopila*

Esta aplicación almacena el identificador único de usuario asignado por Facebook, así como datos sobre sus gustos (“likes”) y, opcionalmente, los identificadores únicos de usuario de los amigos a los que invite a la aplicación.

2. *Cuándo se recopila la información*

La información se recopila en el momento en el que el usuario confirma haber leído y aceptado esta política de privacidad y tras conceder los permisos necesarios en la ventana modal de Facebook.

3. *Cómo se usa la información recopilada*

La información recopilada se utiliza para obtener estadísticas acerca de los gustos musicales del usuario y elaborar un perfil en base a las mismas. Opcionalmente podrá ser analizada comparativamente contra las estadísticas de otros usuarios (amigos) que utilicen la aplicación.

4. *Acceso para actualización, corrección y borrado*

Los datos almacenados se actualizarán cada vez que el usuario ejecute la aplicación. En caso de querer que unos datos determinados sean eliminados se podrá solicitar dicho borrado poniéndose en contacto con nosotros (ver punto 7).

5. *Terceras partes*

La información anónima almacenada podrá ser utilizada con fines de investigación. En ningún caso la información será cedida ni vendida a terceros.

6. *Política de cambios*

Se reserva el derecho a modificar esta política de privacidad en cualquier momento y sin previo aviso.

7. *Detalles de contacto*

Para resolver cualquier duda referente a esta política o solicitar la eliminación de datos, puede escribir a la dirección: diego.castanno@estudiante.uam.es

Al haber internacionalizado la aplicación también fue necesario adjuntar una versión en inglés:

Privacy policy

1. What information is collected

This application stores the user's unique identifier assigned by Facebook, as well as data on their interests ("likes") and, optionally, the user unique identifiers of the friends you invite to the application.

2. When information is collected

Information is collected at the time the user confirms having read and accepted this privacy policy and grant the necessary permissions in the Facebook modal window.

3. How the collected information is used

The collected information is used to obtain statistics about the musical tastes of the user and create a profile based on the same. Optionally can be analyzed comparatively against other users statistics (friends) who use the application.

4. Update, correction and deletion access

The stored data will be updated each time the user runs the application. If you want certain data to be removed you may request such deletion by contacting us (see item 7).

5. Third parties

The anonymous stored information may be used for research purposes. The stored information will not be transferred nor sold to third parties under any circumstance.

6. Policy changes

We deserve the right to modify this privacy policy at any time without notice.

7. Contact Details

To resolve this policy regarding any questions or request deletion of data, please write to this address: diego.castanno@estudiante.uam.es

Ilustración 51 - Privacy Policy