Escuela Politécnica Superior (Jaén)
Ingeniería en Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera

RADIO ONLINE COLABORATIVA PARA ANDROID

Alumno: Francisco Moya Pérez

Tutores: Prof. Dr. Luis Martínez López
Prof. Dr. Iván Palomares Carrascosa

Departamento: Informática

Junio, 2014

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dr. D. Luis Martínez López y Dr. D. Iván Palomares Carrascosa, tutores del Proyecto Fin de Carrera titulado: Radio Online Colaborativa para Android, que presenta D. Francisco Moya Pérez, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2014

El alumno

Los Tutores

D. Francisco Moya Pérez

Dr. D. Luis Martínez López
Dr. D. Iván Palomares Carrascosa

Agradecimientos

Quiero expresar mi más sincero agradecimiento a todas las personas que, de alguna forma, han contribuido a que este proyecto haya salido adelante, con especial mención a:

Mis padres, Nicolás y María, y mi tía, Paqui, porque gracias a su esfuerzo y sacrificio, he conseguido llegar hasta aquí.

Mi novia, Ana, porque me da la fuerza e ilusión para trabajar cada día.

Mis hermanos y familia en general, por ser como son, y por la confianza que siempre han depositado en mí.

Mis tutores, Luis e Iván, porque gracias a su confianza y dedicación en mí he conseguido realizar este magnífico proyecto.

Mi amigo y compañero, Diego, por haberme apoyado desde el primer hasta el último día de estos 5 años de carrera.

Todos los que me habéis apoyado durante la realización de este proyecto, especialmente a Jorge y Francisco.

MUCHAS GRACIAS A TODOS.

Índice general

Lista de figuras

Lista de tablas

CAPÍTULO 1
Introducción

1.1 Introducción al proyecto

El uso, cada vez mayor, de los múltiples servicios que ofrece Internet por parte de los usuarios es de lo más variado, y está propiciando la existencia de sitios Web que sirven de soporte para muchos de estos servicios, no sólo en el ámbito científico, académico o empresarial, sino también para el ocio y disfrute del usuario.

La radio por Internet es actualmente uno de los mayores atractivos de ocio y tiempo libre para los internautas en general, debido a su facilidad de uso y su alto grado de accesibilidad. Se fundamenta en el "Webcasting", es decir, la difusión a través de Internet de contenido multimedia, en este caso de audio. Para ello, se utiliza la tecnología conocida como "Streaming", que consiste en brindar al usuario la posibilidad de reproducir contenidos multimedia directamente y de forma paralela en el navegador Web, sin que este tenga que descargar dicho contenido en su ordenador.

En el ámbito de la radio por Internet cabe destacar la reciente irrupción de radios personalizadas colaborativas, que ayudan al usuario a encontrar nueva música de su agrado, basándose en sus preferencias, es decir, estudiando las características de la música que ya ha escuchado.

Algunos ejemplos de radios colaborativas los podemos encontrar en:

- **Pandora (www.pandora.com):** Mediante una interfaz muy conseguida, ayuda al usuario a generar una lista de artistas y canciones de su agrado desde el momento en que escucha su primera canción, basándose en las similitudes entre objetos (canciones).

- **Last.fm (www.lastfm.es):** Considerada como una red social a gran escala, construye a partir de estadísticas de otros usuarios registrados perfiles sobre los gustos musicales que se adecúen al usuario. Su servicio es de código abierto y se basa en un algoritmo de filtrado colaborativo.

El exitoso funcionamiento de las citadas radios online radica en el uso de lo que formalmente se conoce como Sistemas de Recomendación Colaborativos.

Hay varias formas de definir un Sistema de Recomendación, una de ellas es la siguiente:

Por Sistema de Recomendación se entiende al sistema que utiliza las opiniones de los usuarios que forman una comunidad, para ayudar a otros usuarios (que también pertenecen a esa misma comunidad) a encontrar contenidos de su agrado entre un gran número de contenidos existentes.

Existen varios tipos de Sistemas de Recomendación: Basados en Contenido, Colaborativos [6], Basados en Conocimiento, Comunidad, Demográficos, así como hibridaciones de los anteriores tipos.

Vista la importancia de las radios online hoy en día y en concreto de las radios personalizadas colaborativas, vemos la posibilidad de enmarcar la radio online colaborativa dentro del ecosistema Android.

Android es un sistema operativo basado en Linux [15], diseñado principalmente para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tabletas. Inicialmente desarrollados por Android, Inc., que Google respaldó económicamente y más tarde compró en 2005. Android fue presentado en 2007 junto la fundación del Open Handset Alliance [17]: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles.

Gracias a la importancia de Android y la cantidad de usuarios que lo utilizan nos permiten llevar la radio online colaborativa al bolsillo de millones de usuarios, que podrán escuchar cientos de canciones personalizadas con tan solo arrancar la aplicación.

En el caso del presente proyecto, se pretende, en primera instancia, escoger un algoritmo de filtrado colaborativo que se adecúe bien a un Sistema de Recomendación basado en una colección de archivos musicales de libre distribución, utilizando “Streaming” para su publicación a través de Internet, y ofreciendo así una aplicación Android de radio colaborativa por Internet que puedas llevar en el bolsillo.

Para ello, en primer lugar descargaremos una muestra significativa y variada de referencias de archivos musicales bajo licencia Creative Commons [1], que se encuentran en la base de datos de Internet Jamendo (http://www.jamendo.es).

Una vez creada dicha muestra, se desarrollará el servicio de acceso y se pondrá accesible en un servidor. Posteriormente se realizará una aplicación Android que accediendo al servicio creado recupere la lista de canciones a reproducir. Una vez recuperada la lista de canciones la radio empezará a retransmitir.

1.2 Propósito

Diseño y desarrollo de una radio online colaborativa para Android, basada en las preferencias musicales del usuario, y un servicio que ofrezca toda la información a la aplicación.

1.3 Objetivos

1. Búsqueda y revisión bibliográfica.
2. Estudio de las funcionalidades que podrá disponer la aplicación.
3. Estudiar los métodos de reproducción de música streaming en el sistema Android.
4. Estudio de algoritmos para creación de listas de recomendación.
5. Estudio, preparación e implementación de los métodos de acceso al servidor web de la aplicación.
6. Implementación de la aplicación Android que permitirá escuchar la radio colaborativa en base al algoritmo de filtrado colaborativo estudiado.
7. Redacción de la memoria.

1.4 Estructura y planificación del proyecto

A continuación, haremos una breve introducción a los diferentes capítulos en los que se estructura este proyecto y los contenidos expuestos en los mismos. Como hemos visto, este primer capítulo supone una introducción general al proyecto que se ha realizado, con una justificación de su realización, la definición del propósito y los objetivos que persigue.

El capítulo 2 aborda una visión general de los Sistemas de Recomendación. En primer lugar, se revisa la definición de Sistema de Recomendación, así como la justificación de su utilización. A continuación se analiza la estructura y elementos básicos de los Sistemas de Recomendación, terminando por repasar los distintos

tipos de Sistemas de Recomendación existentes, atendiendo a su funcionamiento, ventajas e inconvenientes. Esta visión de los Sistemas de Recomendación irá acompañada de algunos ejemplos reales de sistemas de colaboración existentes en Internet.

El capítulo 3 está dedicado al estudio del sistema operativo móvil Android y a la introducción a los servicios, concretamente a los servicios web. Por un lado se estudiará la historia, la arquitectura, los componentes y la seguridad en Android y por otro lado se revisará las tecnologías que más se usan en los servidores web hoy en día, profundizando en los servicios REST que serán la base de este proyecto.

El capítulo 4 supone el eje principal de la presente memoria, al ser el de mayor extensión y estar dedicado al proceso completo de desarrollo de este proyecto. Como proyecto software que es, se hará en primer lugar un fugaz repaso a las diferentes etapas de la Ingeniería de Software, para acto seguido aplicarlas en el desarrollo de nuestro sistema. Así, se definirán los requerimientos funcionales y no funcionales para el sistema, abordando la etapa de análisis, incluyendo el modelo de casos de uso y el análisis de la base de datos. Acto seguido, se pasa a la etapa de diseño, haciendo especial hincapié en el diseño de la interfaz de la aplicación Android y las pruebas de usabilidad llevadas a cabo. Por último, repasaremos el proceso seguido para la etapa de implementación del servicio REST y el acceso a él desde Android.

Una vez expuesto el desarrollo del proyecto, llegamos al quinto y último capítulo, dedicado a las conclusiones generales derivadas del desarrollo del mismo.

La sección final de esta memoria contiene los anexos dedicados a la instalación y al manual de usuario de la aplicación Android.

CAPÍTULO 2 Sistemas de recomendación

2.1. Introducción

El actual auge de las Tecnologías de la Información, germen de lo que conocemos como Sociedad de la Información y su evolución hacia la Sociedad del Conocimiento, han propiciado que las personas dispongamos cada vez de más información para realizar nuestros acometidos. Esto es en cierto modo una ventaja, pero también nos encontramos con frecuencia el problema de sobrecarga de información, lo cual puede llegar a dificultar seriamente la tarea de escoger la información más adecuada a nuestras necesidades.

Es en Internet donde esta situación se hace presente en gran medida, debido al cada vez mayor número de sitios dedicados a múltiples propósitos que ofrecen una considerable colección de información. El usuario necesita algún tipo de ‚ayuda' para elegir aquel contenido que sea de su interés. Así, en los últimos años los servicios de las citadas aplicaciones Web se han ido centrando en personalizar sus productos y/o servicios, de manera que consigan satisfacer las necesidades de cada usuario concreto.

Es aquí donde interviene una herramienta ampliamente utilizada, de forma satisfactoria, para resolver este inconveniente: se trata de los Sistemas de Recomendación.

Una posible definición de Sistema de Recomendación (que en adelante también nombraremos de forma abreviada como SR) [2] es:

Conjunto de herramientas y técnicas software capaces de proporcionar sugerencias de interés a un usuario con respecto a una serie de ítems. Estas sugerencias tratan de ajustarse a los gustos y necesidades del usuario, previamente recogidos por el sistema, y hacen referencia a diversos procesos de toma de decisiones, como qué productos comprar, qué música escuchar, o qué noticias leer.

Ítem es el término general usado para denotar qué es lo que un sistema recomienda a un usuario [3]. Un SR normalmente se centra en un tipo específico de ítem (por ejemplo películas, canciones o libros) y de acuerdo con su diseño, la interfaz gráfica mostrada al usuario y la técnica de recomendación utilizada, se personalizan las recomendaciones para proporcionar sugerencias que al usuario le sean útiles y efectivas para ese tipo específico de ítem.

Figura 1: Esquema básico de Sistema de Recomendación

Los SR están principalmente dirigidos a individuos que carecen de la suficiente experiencia personal o de las competencias necesarias para evaluar el potencial de un enorme número de ítems diferentes que, por ejemplo, un sitio Web puede ofrecer (Véase figura 1). Un ejemplo de ello es un sistema de

recomendación de libros que ayude a los usuarios a elegir qué libro leer. Dado que las recomendaciones suelen ser personalizadas, diferentes usuarios o grupos de usuarios recibirán diferentes sugerencias. Por otra parte, existen también recomendaciones no personalizadas [4], que son mucho más simples de generar y que son las que normalmente aparecen en revistas o periódicos. Ejemplos típicos de esta no personalización son las listas de los n mejores, por ejemplo, los diez mejores libros, las 20 mejores películas, etc. Aunque puedan resultar útiles y efectivas en ciertas situaciones, estos tipos de recomendaciones no personalizadas no son el objetivo final de un SR ni las que típicamente busca.

En su forma más simple, las recomendaciones personalizadas se ofrecen como listas clasificadas y ordenadas de ítems. Para realizar esta tarea, los SR tratan de predecir los servicios o productos que serán más adecuados, basándose en las preferencias, restricciones y conocimiento del usuario. Para que esta tarea computacional pueda ser completada, los SR deben adquirir de los usuarios sus preferencias, que pueden ser expresadas explícitamente, como por ejemplo valoraciones de productos, o inferidas interpretando las acciones llevadas a cabo por el usuario. Por ejemplo, un SR puede considerar la navegación sobre la página de un producto en particular como un signo implícito de preferencia o interés por los ítems mostrados en esa página [2].

Los SR han demostrado en los últimos años ser un valioso medio para hacer frente al problema de la sobrecarga de información en distintos ámbitos como el comercio electrónico, turismo, educación, etc. En definitiva los SR abordan este fenómeno guiando al usuario hacia ítems nuevos, aún no experimentados y que puedan ser relevantes para las necesidades del usuario. Frente a la petición de un usuario, que puede ser articulada, según el enfoque de recomendación, por el contexto del usuario y sus necesidades, los SR generan recomendaciones utilizando diversos tipos de conocimiento y datos sobre los usuarios, los elementos disponibles, y las transacciones anteriores almacenados

en bases de datos. El usuario puede entonces explorar las recomendaciones ofrecidas, aceptarlas o no y proporcionar, inmediatamente o en un paso posterior, una retroalimentación implícita o explícita. Todas estas acciones y reacciones del usuario pueden ser almacenadas en la base de datos de recomendación y utilizadas para la generación de nuevas recomendaciones en las próximas interacciones entre el usuario y el sistema.

Actualmente existen varios tipos de Sistemas de Recomendación [5], entre los que destacan los Sistemas de Recomendación Colaborativos, basados en contenido, basados en conocimiento, basados en la comunidad, demográficos e híbridos. Veremos con mayor profundidad las características de cada uno de ellos en un epígrafe posterior.

Tanto los Sistemas de Recomendación Colaborativos como los basados en contenido presentan el inconveniente de requerir una gran cantidad de información, tanto de usuarios del mismo como de los ítems que lo componen, para realizar recomendaciones de calidad y, por tanto, funcionar correctamente. Para resolver este inconveniente aparecieron otros tipos de Sistemas de Recomendación capaces de realizar recomendaciones de calidad sin necesidad de una gran cantidad de información: se trata de los Sistemas de Recomendación basados en conocimiento. Por último, de cara a mejorar aún más los resultados de los Sistemas de Recomendación, han aparecido en los últimos años los Sistemas de Recomendación híbridos, en los que se sintetizan las ventajas de dos o más de los anteriores tipos de Sistemas de Recomendación, para así lograr un resultado mejor al que cada uno de ellos lograría por separado.

A continuación, estudiaremos brevemente las motivaciones que nos llevan a utilizar un sistema de recomendación, la estructura básica de funcionamiento de los Sistemas de Recomendación y los principales elementos que los componen.

2.1.1 Motivación para usar los Sistemas de Recomendación

Vamos a distinguir entre el rol que juega quien ofrece el servicio de recomendación frente al usuario del mismo. Por ejemplo, un sistema de recomendación de viajes suele ser ofrecido al cliente bien por una empresa intermediaria con el fin de incrementar su volumen de negocio. Mientras tanto, las principales motivaciones de un usuario para acceder a cualquiera de estos sistemas es encontrar un hotel adecuado y eventos o lugares de interés para visitar en el sitio de destino.

De hecho, hay varias razones por las cuales los proveedores de servicios pueden querer aprovechar estas tecnologías:

- *Incrementar el número de ítems vendidos.* Esta es probablemente la función más importante para un SR comercial [3], por ejemplo, ser capaz de vender más ítems si se compara con los que usualmente se venden sin ningún tipo de recomendación. Este objetivo se alcanza porque los ítems recomendados suelen encajar con las necesidades y los deseos del usuario. El usuario reconocerá presumiblemente este hecho después de haber probado varias recomendaciones. Las aplicaciones no comerciales tienen fines similares, incluso si no supone un coste para el usuario que se asocia a la selección de un ítem.

- *Vender ítems más diversos.* Otra función principal en un SR es permitir al usuario seleccionar ítems que posiblemente hubieran sido difíciles de encontrar sin una precisa recomendación [3]. Por ejemplo, en un SR de películas como Netflix, el proveedor está interesado en alquilar todos los DVDs del catálogo, no sólo los más populares. Esto podría ser difícil sin un SR, puesto que el proveedor no puede permitirse el riesgo de

publicitar a un usuario películas que puede que no encajen con sus gustos personales.

- *Incrementar la satisfacción del usuario.* Un SR bien diseñado puede además mejorar la experiencia del usuario con el sitio o con la aplicación. Esto incrementará a cambio el uso del sistema y la probabilidad de que las recomendaciones sean aceptadas.

- *Incrementar la fidelidad del usuario.* Un usuario será fiel a un sitio web cuando lo orienta correctamente en las recomendaciones. En consecuencia, cuanto más interacciona el usuario con el sitio, más irá el sistema perfeccionando el modelo para dicho usuario.

- *Mejor entendimiento de lo que el usuario busca.* Otra función importante de un SR, que puede aprovecharse para muchas otras aplicaciones, es la descripción de las preferencias del usuario, tanto las recogidas explícitamente como las predichas por el sistema. El proveedor del servicio puede decidir reutilizar este conocimiento para otras metas tales como mejorar la gestión del stock o la producción de ítems. Por ejemplo, en el dominio de los viajes, las organizaciones de gestión de destinos pueden decidir informar sobre una región a nuevos sectores de clientes o publicitar un nuevo tipo de mensaje promocional derivado del análisis de los datos adquiridos por el SR.

Estas son las motivaciones más importantes que inducen a los proveedores de un servicio a introducir un SR. Pero los usuarios también pueden querer un SR si de forma efectiva aporta ayuda a la hora de llevar a cabo sus tareas o metas. Consecuentemente, un SR deberá equilibrar las necesidades de estos dos agentes y ofrecer un servicio valioso para ambos.

2.1.2. Esquema Básico de Funcionamiento y Elementos de un SR

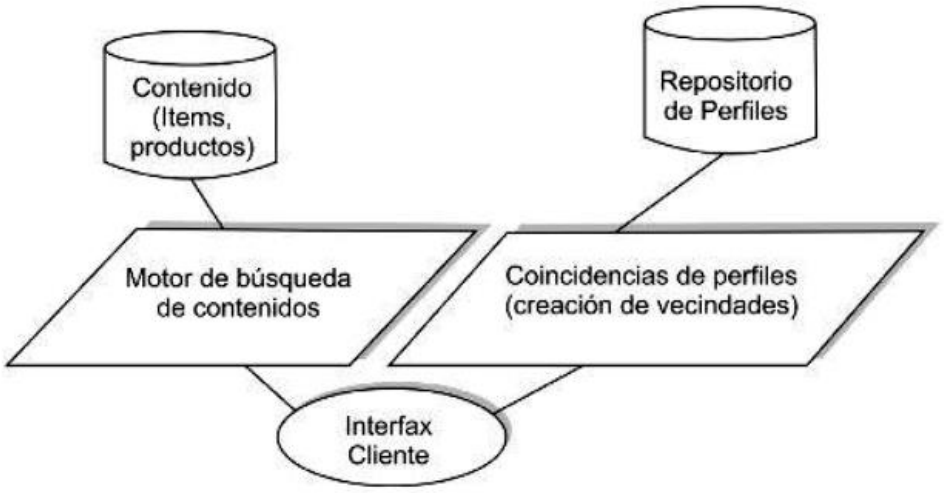

Figura 2: Esquema básico de funcionamiento de un SR

En la figura 2 se puede observar un esquema básico de un SR en el que se puede distinguir los siguientes elementos:

- Base de datos: La calidad de los datos almacenados en nuestra base de datos jugará un papel crucial a la hora de realizar recomendaciones con mayor o menor calidad.

- Perfiles de Usuario: Un usuario va “dándole forma” a su perfil personal a medida que utiliza el sistema. El perfil refleja los gustos/preferencias del usuario, fundamentales a la hora de discriminar objetos durante la recomendación.

- Predicciones: La predicción juega un papel fundamental dentro del esquema básico de todo SR. La predicción se basa en el perfil del usuario y en la información disponible en la base de datos con la que contamos.

2.1.3. Datos en los SR

Es importante tomar algunas decisiones a la hora de desarrollar un Sistema de Recomendación, tales como el tipo de realimentación que utilizará, el tipo de datos a utilizar o la forma en que dichos datos se analizarán.

2.1.3.1. Realimentación en los SR

Un Sistema de Recomendación no debe ser una entidad estática, sino que la calidad de sus recomendaciones ha de evolucionar con el tiempo, en base a la experiencia y nueva información adquiridas. Esto se consigue mediante los mecanismos de realimentación entre el sistema y las preferencias de los usuarios. Existen dos mecanismos de realimentación: la realimentación implícita y la realimentación explícita.

Realimentación Implícita

Un mecanismo de realimentación implícita es aquel que proporciona al SR información sobre las preferencias de los usuarios sin que estos sean conscientes de ello. Estas realimentaciones no se hacen de forma directa, sino mediante algunas medidas como pueden ser: el tiempo de visualización del objeto, el número de consultas del mismo, etc.

Presenta el problema de que depende demasiado del contexto y es bastante hipotética, ya que se hacen suposiciones (a partir de las mencionadas

medidas) sobre los gustos del usuario que no necesariamente tienen por qué ser ciertas.

Realimentación Explícita

La realimentación explícita se basa en la acción directa y deliberada del usuario para indicar aquellos objetos del sistema que le interesan. Esta acción se puede conseguir mediante votaciones numéricas o, simplemente, indicando si el objeto es o no del agrado del usuario. Este tipo de realimentación también presenta problemas, como son la voluntariedad del cliente o el tiempo invertido en ello.

2.1.3.2. Datos Reales vs. Datos Sintetizados

Otra cuestión interesante es la de elegir un conjunto de datos reales (recopilados de usuarios reales sobre objetos reales) o un conjunto de datos sintetizados (sin ninguna base real, específicamente creados para el Sistema de Recomendación). Estos últimos son más fáciles de obtener, ya que evitamos tener que realizar encuestas u otros métodos de recopilación de información real, aunque sólo se utilizan en las primeras fases del desarrollo del sistema, para posteriormente ser sustituidos por los datos reales una vez que se haya recopilado la información suficiente.

2.1.3.3. Análisis Online vs. Análisis Offline

Es importante decidir si vamos a trabajar sobre los datos de manera online u offline. En el análisis offline se emplea una técnica o algoritmo de filtrado para hacer predicciones sobre el conjunto de datos, evaluando los resultados de dichas predicciones mediante una o varias métricas de error. Este tipo de análisis

presenta la ventaja de ser rápido y económico, pero presenta a la vez dos inconvenientes importantes: el problema de la escasez de datos y el problema de obtener como único resultado la bondad de la predicción.

Por el contrario, el análisis online permite obtener más resultados, entre los que destacan la actuación de los usuarios participantes, la satisfacción de los mismos, etc. Sin embargo, resulta ser más lento y costoso que el análisis offline.

2.2. Clasificación de los Sistemas de Recomendación

Una vez sentadas las bases de los Sistemas de Recomendación, es el momento de revisar los diferentes tipos existentes de los mismos. Los Sistemas de Recomendación pueden ser implementados utilizando diversas técnicas [6], por lo que su clasificación dependerá de su funcionamiento para calcular recomendaciones.

2.2.1. Sistemas de Recomendación Basados en Contenido

Un Sistema de Recomendación basado en contenido es aquel que está basado en las características de los objetos, es decir, las recomendaciones se llevan a cabo basándose únicamente en un perfil creado a partir del análisis del contenido de los objetos que el usuario ha evaluado con anterioridad.

En otras palabras: estos sistemas extraen características de los objetos y las comparan con el perfil del usuario para predecir las preferencias de los usuarios sobre tales objetos. La idea es recomendar objetos similares en su contenido a objetos que ya sabemos que son del agrado del usuario en cuestión, es decir, los que forman parte de su perfil.

El filtrado basado en contenido era el tipo de Sistema de Recomendación más extendido antes de que se produjese la explosión del filtrado colaborativo, ya que los sistemas del primer tipo tienen un claro problema de sobre-especialización.

El funcionamiento de los sistemas de colaboración basados en contenido comprende dos grandes etapas: analizar las descripciones de los productos valorados por los usuarios y predecir qué productos le pueden gustar (Véase figura 3).

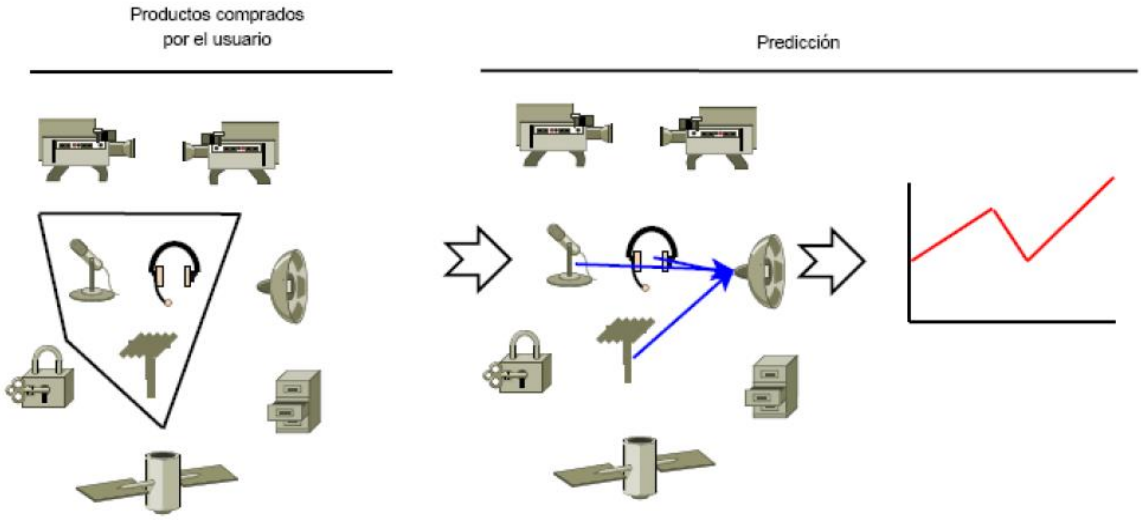

Figura 3: Esquema de funcionamiento de un Sistema de Recomendación basado en contenido

Este funcionamiento dependerá en gran medida del tipo de información descriptiva que se utilice en el perfil del usuario. Esta información se divide en:

- Conjunto de características: Asociado a cada producto tenemos un conjunto de características que lo describen [7]. Por ejemplo, kilometraje o consumo medio si estamos hablando de vehículos.

- Información textual sobre el producto: Se trata de un documento que describe al producto. Se diferencia del conjunto de características principalmente en que la información no está estructurada.

2.2.2. Sistemas de Recomendación Colaborativos

Se trata de los Sistemas de Recomendación más extendidos y consolidados en el mercado actual. Los Sistemas de Recomendación basados en un filtrado colaborativo son aquellos que realizan recomendaciones basándose en los términos de similitud entre usuarios, esto es, recomiendan objetos que son del agrado de otros usuarios con intereses similares.

Para la realización de un buen Sistema de Recomendación Colaborativo, que ofrezca recomendaciones de calidad, es fundamental emplear un buen algoritmo de filtrado colaborativo. Estos algoritmos se pueden clasificar en dos categorías: los algoritmos basados en memoria o usuario y los algoritmos basados en ítem. La mayor ventaja de las técnicas colaborativas es que son totalmente independientes de la representación interna de los objetos que se pueden recomendar.

Sin embargo, a medida que se ha extendido su utilización, se han detectado problemas como son: la escasez, la escalabilidad y el problema del ítem nuevo. Por ello, han llevado a cabo multitud de estudios y experimentos orientados a minimizar el efecto de estos problemas.

Este modelo de Sistemas de Recomendación constituye el principal eje en torno al cual versa el presente proyecto, por lo que veremos un estudio más detallado de los mismos en un epígrafe posterior.

2.2.3. Sistemas de Recomendación Basados en Conocimiento

Estos sistemas recomiendan ítems en función de cómo ciertas propiedades de un ítem satisfacen las necesidades y preferencias de un usuario y, en última instancia, cómo el ítem es útil para el usuario [8].

Ejemplos de este tipo de sistemas son los SR basados en casos. En estos sistemas una función de similitud estima en qué medida las recomendaciones coinciden con las necesidades del usuario. Aquí el valor de similitud puede interpretarse directamente como la utilidad de la recomendación para el usuario.

Los sistemas basados en restricciones son otro tipo de SR basados en conocimiento. En tanto en cuanto al conocimiento utilizado, ambos sistemas son parecidos: se recogen los requerimientos del usuario; se proponen automáticamente modificaciones para requerimientos inconsistentes cuando no se pueden encontrar soluciones; se explican los resultados de la recomendación. La principal diferencia subyace en la forma en la que las soluciones se calculan.

Los SR basados en casos determinan recomendaciones sobre la base de mediciones de similitud mientras que los basados en restricciones predominantemente explotan bases de conocimiento predefinidas que contienen reglas explícitas sobre cómo relacionar los requerimientos del cliente con las características de los ítems.

Los sistemas basados en conocimiento tienden a trabajar mejor que otros al principio de su implantación pero si no están equipados con componentes de aprendizaje pueden verse superados por otros métodos que pueden explotar registros de interacción hombre-máquina (como en filtrado colaborativo).

2.2.4 Sistemas de Recomendación Basados en la Comunidad

Este tipo de sistemas recomiendan ítems basándose en las preferencias de los amigos de los usuarios. Esta técnica sigue el lema “Dime quiénes son tus amigos, y te diré quién eres” [9]. Evidencias sugieren que la gente tiende a fiarse más de las recomendaciones de sus amigos que en las recomendaciones de individuos parecidos pero anónimos.

Esta observación, combinada con el crecimiento de la popularidad de las redes sociales, está generando un creciente interés en los sistemas basados en comunidades o, como se suele referirse a ellos, sistemas de recomendación sociales. Sin embargo, la investigación en esta área está todavía en una fase temprana y los resultados sobre el rendimiento de estos sistemas están todavía divididos a favor y en contra.

2.2.5 Sistemas de Recomendación Demográficos

Este tipo de SR [10] se basa en la recomendación de ítems según el perfil demográfico del usuario. Se asume que se deberían generar diferentes recomendaciones para nichos demográficos distintos. Muchos sitios web adoptan soluciones simples y efectivas de personalización basándose en este enfoque.

Por ejemplo, a los usuarios se les puede servir páginas web concretas de acuerdo con su lengua o país. O bien se pueden hacer sugerencias de personalización de acuerdo a la edad del usuario. Aunque este tipo de enfoques ha sido muy popular en la literatura comercial, ha habido relativamente poca investigación apropiada en el ámbito de los Sistemas de Recomendación Demográficos.

2.2.6 Sistemas de Recomendación Híbridos

Como hemos mencionado, cada uno de los modelos de recomendación anteriormente vistos presenta, además de sus ventajas, algún tipo de problema. Por ello, surgió la idea de solventar esta situación aunando sus puntos fuertes e intentando minimizar sus inconvenientes, mediante la hibridación de al menos dos de los diferentes tipos de Sistemas de Recomendación. De esta forma nacieron los Sistemas de Recomendación híbridos.

Los sistemas híbridos entre los basados en contenido y los colaborativos, por ejemplo, guardan las preferencias del usuario y las combinan con los objetos más relevantes para realizar las recomendaciones.

Existen también los sistemas híbridos entre los colaborativos y los basados en conocimiento, los basados en contenido y en conocimiento, e incluso entre los colaborativos y las redes sociales.

2.3. Sistemas de Recomendación Colaborativos

Debido a que este proyecto se sustenta sobre un algoritmo de filtrado colaborativo, a continuación vamos a hacer una revisión un poco más en detalle de este tipo de Sistemas de Recomendación [11].

2.3.1. Introducción y Orígenes

Los Sistemas de Recomendación Colaborativos son aquellos que realizan recomendaciones basándose únicamente en términos de similitud entre los usuarios, es decir, combinan las valoraciones de los objetos, identifican los gustos

comunes entre usuarios en base a dichas valoraciones y recomiendan así objetos que son del gusto de otros usuarios de gustos similares al usuario actual.

Las técnicas para desarrollar los primeros Sistemas de Recomendación de filtrado colaborativo estaban basadas en métodos provenientes de la minería de datos [7]. Para ello, se distinguía entre una fase de aprendizaje (offline) en la que se aprende el modelo, al igual que ocurre en la minería de datos, y una fase de recomendación (online) en la que se aplica el modelo obtenido de la fase anterior a un problema de la vida real, produciéndose así las recomendaciones para los usuarios del sistema. No obstante, actualmente este tipo de técnica no se suele utilizar, ya que debido a la interacción de los usuarios con el sistema es más conveniente emplear un paradigma de aprendizaje relajado (el modelo se construye y actualiza durante el funcionamiento del sistema).

La base teórica de los Sistemas de Recomendación es simple: se forman grupos de usuarios más cercanos, que serán los que mantengan unos perfiles similares, y a un usuario de un grupo se le recomiendan objetos que aún no ha experimentado, pero sí han experimentado y valorado positivamente otros usuarios de su “grupo”.

2.3.2. Fases de los SR Colaborativos

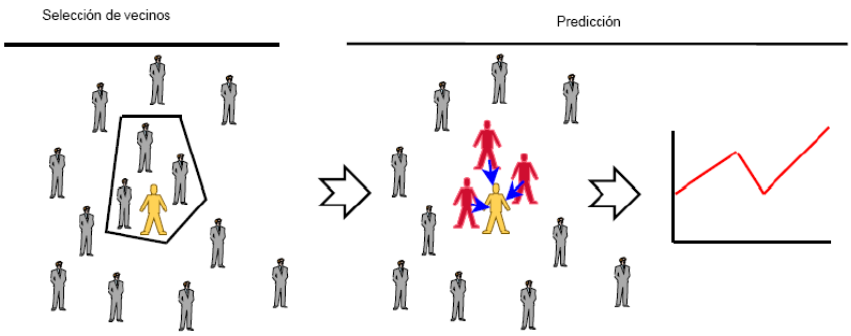

Figura 4: Funcionamiento de los Sistemas de Recomendación Colaborativos

Shardaham y Maes [36] distinguieron tres etapas fundamentales en el funcionamiento de todo Sistema de Recomendación Colaborativo (Véase figura 4):

1. El sistema guarda un perfil de cada usuario, que consta de las evaluaciones de objetos conocidos por él y que pertenecen a la base de datos sobre la que se trabajará.

2. En base a estos perfiles, se mide el grado de similitud entre los usuarios del sistema y se crean grupos de usuarios con características similares.

3. El sistema utiliza la información obtenida en los dos pasos anteriores para calcular las predicciones. A cada usuario se le recomendarán objetos que no haya evaluado previamente y que hayan obtenido los mayores valores para dicha predicción (Véase figura 5).

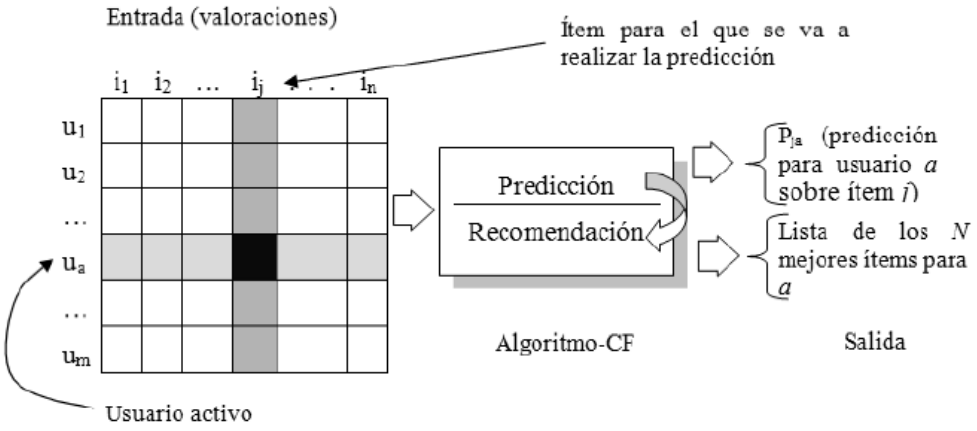

Figura 5: Proceso de cálculo de predicciones en SR Colaborativos

Por tanto, estos Sistemas de Recomendación no toman en consideración el contenido y las características de los productos que recomiendan, sino que sean del gusto de usuarios con un perfil semejante al del usuario que demanda el servicio.

2.3.3. Algoritmos de Filtrado Colaborativo

Para desarrollar un buen Sistema de Recomendación Colaborativo, es de vital importancia elegir un buen algoritmo de filtrado colaborativo. Existen distintas posibilidades a la hora implementar dicho algoritmo. Veamos con un poco de detalle los diferentes tipos de algoritmos de filtrado [12].

- Algoritmos basados en memoria o basados en usuario.

Estos algoritmos realizan las recomendaciones basándose en la base de datos completa, teniendo en cuentas todos los ítems previamente evaluados por el usuario (Véase figura 6). El funcionamiento de los algoritmos es el siguiente: se utilizan técnicas estadísticas para determinar un conjunto de vecinos al usuario activo y, posteriormente, se aplican algoritmos que combinan preferencias de esta vecindad para realizar las predicciones y recomendaciones.

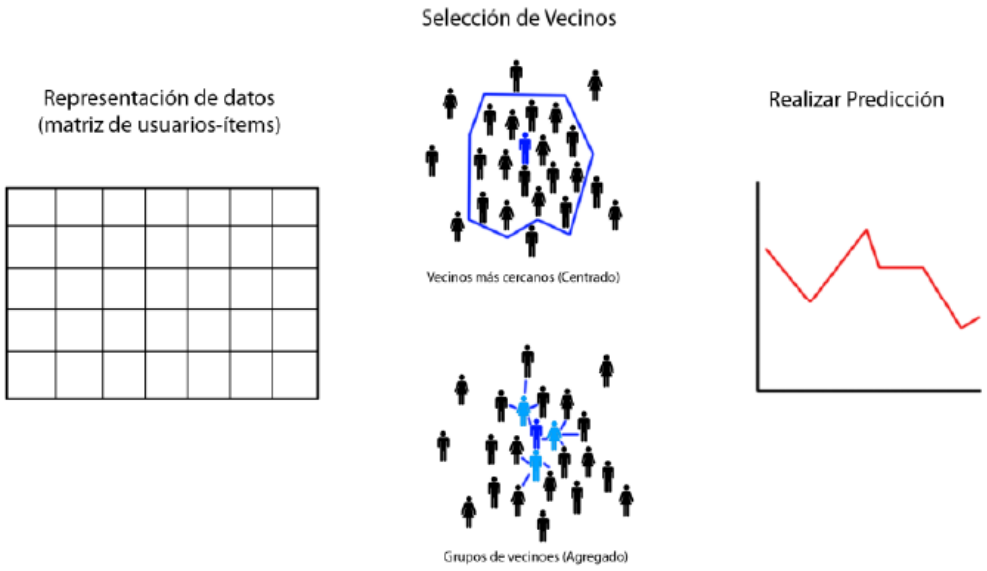

Figura 6: Las principales partes de un SR Colaborativo basado en memoria

Pese a ser bastante populares y exitosos en la práctica, suelen sufrir especialmente los problemas de escalabilidad y escasez. Se hizo necesario, pues, el desarrollo de otro tipo de algoritmos de filtrado colaborativo.

- Algoritmos basados en modelos o basados en ítem

Estos algoritmos proporcionan recomendaciones de ítems desarrollando primero un modelo (ya sea mediante redes bayesianas, modelos basados en reglas, clustering o modelos basados en vecindarios) de las puntuaciones de los usuarios sobre los ítems.

No se utilizan técnicas estadísticas sino una aproximación probabilística que calcula el valor esperado de una predicción del usuario dados sus puntuaciones sobre otros ítems. Es decir, estos algoritmos miran en el conjunto de ítems que el usuario activo ha puntuado o evaluado y calcula como de similar son estas puntuaciones con respecto al ítem activo con el fin de realizar una predicción para el mismo.

Dado que este es el tipo de algoritmo sobre el que se centrará el presente proyecto, vamos a ver a continuación las distintas técnicas más comunes para obtener el modelo de recomendación.

- *Redes bayesianas*: Esta técnica consiste en obtener el modelo a partir de un conjunto de entrenamiento con un árbol de decisión, donde los nodos y ramas representan información de los usuarios. Suele resultar útil cuando el conocimiento sobre el perfil de usuario cambia de forma lenta.

- *Modelos basados en reglas*: El procedimiento es similar al de la técnica anterior, con la diferencia de que el modelo obtenido tiene la forma de un conjunto de reglas del tipo “Antecedente => Consecuente”.

- *Técnicas de agrupamiento (clustering):* Hacen grupos de usuarios, denominados clusters, con gustos similares. Las predicciones para un individuo se realizan mediante la agregación de opiniones de otros usuarios del mismo grupo. La tarea de creación de los clusters conlleva un gran esfuerzo, pero una vez creados se obtiene un rendimiento bueno, ya que el grupo a considerar para realizar las recomendaciones queda considerablemente reducido.

- *Modelos basados en vecindarios*: Estos algoritmos proporcionan recomendaciones desarrollando en primer lugar un modelo, utilizando cualquiera de las tres técnicas anteriormente descritas, de las puntuaciones de los usuarios sobre los ítems.

Los algoritmos basados en vecindarios miran en el conjunto de ítems evaluados por el usuario activo para calcular cómo de parecidas son estas puntuaciones al ítem activo, con el fin de realizar una predicción para el mismo. Para realizar las recomendaciones se realizan las siguientes tareas:

1. Exploración del conjunto de ítems que el usuario ha valorado.

2. Cálculo de la similitud de los ítems anteriores con respecto al ítem o producto del cual queremos predecir la puntuación que el usuario le daría.

3. Selección de los k ítems más cercanos (Knn).

4. Cálculo de la predicción del usuario activo sobre el ítem dado como la media ponderada de las valoraciones del usuario hacia los k ítems más similares.

Dentro de los Sistemas de Recomendación Colaborativos basados en ítem, emplearemos este último modelo para el algoritmo de filtrado de este proyecto.

2.3.4. Medidas y Técnicas Empleadas en Algoritmos de Filtrado Colaborativo

Una vez estudiada la clasificación de algoritmos de filtrado colaborativo, pasamos a definir algunas medidas para realizar las recomendaciones en este tipo de sistemas.

El primer paso para la realización de dichas recomendaciones consiste en formar grupos con los usuarios o ítems más similares entre sí. Para ello se emplearán medidas de similitud y un algoritmo de clasificación K-nn [13]. Acto seguido, se empleará una técnica de predicción para estimar la valoración del usuario sobre ciertos ítems. En este proyecto se ha elegido el coeficiente coseno como medida de similitud para la implementación del algoritmo de filtrado colaborativo.

2.3.4.1. Notación

Antes de enumerar las fórmulas que veremos a continuación, es conveniente dejar clara la notación que se va a emplear para no producir confusión al lector:

- Un usuario es aquel elemento representado por $u_{\mathrm{i}} \in \mathrm{U} = \{u_1, u_2, \dots, u_{\mathrm{n}}\}$
- Un ítem será aquel elemento representado por $i_{\mathrm{i}} \in \mathrm{I} = \{i_1, i_2, \dots, i_{\mathrm{m}}\}$
- La similitud entre dos ítems i_{j} y i_{k} vendrá dada por $\mathrm{s}(i_{\mathrm{j}}, i_{\mathrm{k}})$
- Una evaluación de un usuario u_{i} sobre un ítem i_{j} vendrá dada por $r_{\mathrm{ui,ij}}$

- Una predicción de un usuario u_{i} sobre el ítem i_{j} se representará como $p_{\mathrm{ui,ij}}$

2.3.4.2. Algoritmo K-nn

Un paso crucial en la realización de un Sistema de Recomendación Colaborativo de calidad es la formación de grupos de usuarios (si es un SR Colaborativo basado en memoria) o de ítems (si es basado en modelos, como ocurre en este proyecto) de características similares. Esta actividad forma parte de lo que conocemos como problemas de clasificación, y existen diversas técnicas, llamadas clasificadores, para resolver dicho problema. Uno de los clasificadores más difundidos es el algoritmo K-nn, que se utilizará en este proyecto para formar los grupos de ítems más similares para cada uno de los ítems de la base de datos [13].

Veamos con mayor detalle en qué consiste el algoritmo K-nn:

Funcionamiento

Siendo i el objeto a clasificar, debemos seleccionar los k objetos con $\mathrm{K} = \{i_1, \ldots, i_{\mathrm{k}}\}$ tal que no existe ningún ejemplo i' fuera de K con $\mathrm{d}(\mathrm{i}, \mathrm{i}') < \mathrm{d}(\mathrm{i}, i_{\mathrm{j}}), \mathrm{j} = 1, \ldots, \mathrm{k}$

Una vez encontrado los k vecinos, se puede proceder a la clasificación de dos formas posibles:

- *Voto por la mayoría*: Se clasifica el nuevo objeto según la clase predominante en los objetos de K.

- *Voto compensado por la distancia*: Se clasifica el objeto en base a su distancia ponderada con el resto de objetos de K.

Descripción del algoritmo

1. Aparece un nuevo objeto i_a.
2. Se obtienen los k objetos del conjunto E más cercanos a i_a.
3. Se clasifica el objeto i_a de una de las dos formas antes mencionadas.[1]

Principales características

- Es un algoritmo robusto frente al ruido cuando k es moderado (k>1).
- Es muy eficaz cuando el número de clases posibles es alto y cuando los datos son heterogéneos o difusos.
- Tiene un orden de complejidad de $O(dn^2)$, siendo $O(d)$ la complejidad de la distancia de métrica empleada.
- El hecho de no utilizar modelos sino la base de datos al completo provoca que sea ineficiente en memoria.
- Sirve tanto para realizar clasificación como para predicción numérica.

En nuestro proyecto se empleará el algoritmo K-nn para seleccionar los k ítems más similares a cada uno de los ítems que componen nuestra base de datos.

[1] En este proyecto nos limitamos a utilizar el algoritmo K-nn para seleccionar los k vecinos más cercanos a cada uno de los objetos, por lo que no necesitamos implementar este último paso.

2.3.4.3. Medidas de Similitud

Para establecer la similitud entre objetos debemos definir una medida que nos permita evaluar el grado de parecido entre unos y otros.

Es importante resaltar que, en este proyecto, para calcular la similitud entre dos ítems x e y se tendrán en cuenta únicamente a aquellos usuarios que hayan evaluado a ambos ítems, no siendo tomados en consideración el resto (Véase figura 7).

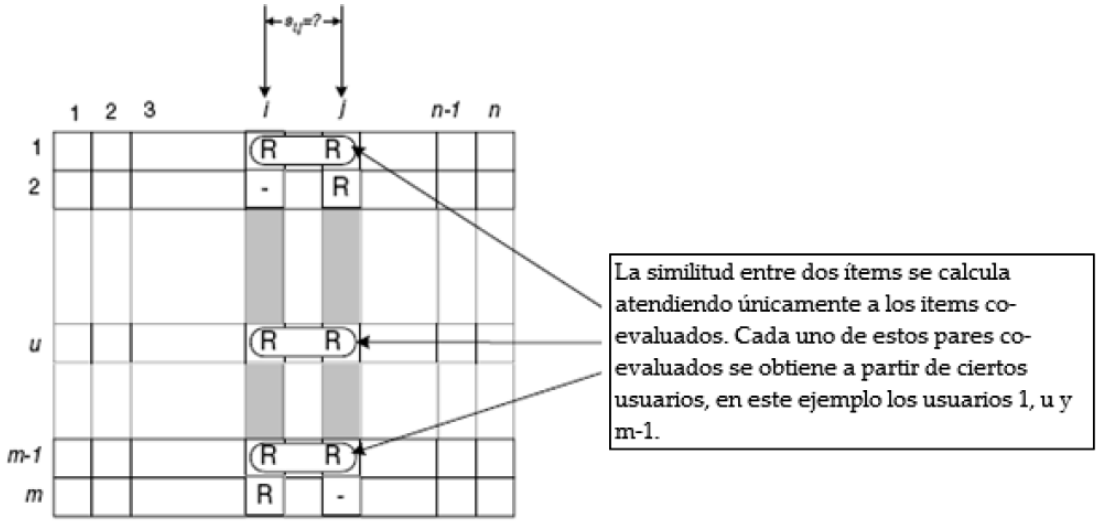

Figura 7: Cálculo de similitud basada solamente en ítems co-evaluados

Existen en la literatura multitud de medidas de similitud [14], de entre todas, nosotros vamos a revisar dos de las más utilizadas.

Coeficiente de Correlación de Pearson

Este coeficiente es un índice que mide la relación lineal entre dos variables cuantitativas, siendo independiente de la escala de medida de dichas variables y

dando un resultado dentro del intervalo [-1,1]. Se calcula mediante la siguiente expresión:

$$S(x,y) = \frac{\sum_{I=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

Siendo $\overline{x}$ e $\overline{y}$ la media de x e y, respectivamente.

Coeficiente Coseno

Este método supone que dos ítems x e y vienen representados por vectores en el espacio, por lo que la similitud entre ellos vendrá dada por el coseno del ángulo que forman. La expresión para su cálculo es la siguiente:

$$s(x,y) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n}(x_i)^2 \sum_{i=1}^{n}(y_i)^2}}$$

Siendo x_{i} el valor del objeto x para el usuario i , y_{i} el valor del objeto y para el usuario i y n el número de usuarios que han evaluado tanto x como y.

2.3.4.4 Algoritmos de Predicción

Tras haber calculado el conjunto de vecinos para cada ítem, hemos de combinar las valoraciones de dicho conjunto para realizar la predicción del usuario sobre dicho ítem. Elegir la técnica adecuada para realizar la predicción supone el paso más crucial del filtrado colaborativo.

El escoger un algoritmo de predicción u otro depende de la naturaleza del conjunto de datos, ya que cada algoritmo se ajusta mejor a un conjunto de datos específico. En nuestro caso, utilizaremos la técnica llamada weighted sum. Este

método calcula la predicción de un ítem i por parte de un usuario u_{a} como la suma de las evaluaciones del usuario ua sobre ítems similares a i. Cada una de estas evaluaciones viene ponderada por la correspondiente similitud $\mathrm{s(i,j)}$ entre los ítems i y j. Su expresión viene dada por:

$$p(u_a, i_a) = \frac{\sum_{h=1}^{k} s(i_a, i_h) * ru_a, i_h}{\sum_{h=1}^{k} |s(i_a, i_h)|}$$

Siendo k los k ítems más similares a i_{a}. Esta técnica intenta captar cómo el usuario activo evalúa a ítems similares al que quiere predecir. Para asegurarnos de que la predicción entra del rango previamente definido, es necesario ponderar estas evaluaciones con la similitud.

Ya hemos sentado las bases teóricas sobre las que nos hemos apoyado para implementar nuestro algoritmo de filtrado colaborativo. En el siguiente apartado citaremos los principales problemas que han surgido de la utilización de Sistemas de Recomendación Colaborativos.

2.3.5. Problemas de los Sistemas de Recomendación Colaborativos

Como hemos mencionado con anterioridad, los Sistemas de Recomendación Colaborativos presentan una serie de problemas que se han ido poniendo de relieve a medida que su utilización se ha extendido: escasez, escalabilidad y problema del ítem nuevo. Para tener una mejor concepción de estos, pasaremos a describirlos brevemente a continuación:

Escasez

Dada su naturaleza, los Sistemas de Recomendación Colaborativos requieren una gran cantidad de usuarios que realicen un igualmente grande número de puntuaciones sobre ítems similares, para poder así calcular los grupos de ítems cercanos y realizar recomendaciones de calidad. Si el número de usuarios registrados en el sistema es pequeño, o incluso si siendo elevado no han hecho suficientes puntuaciones, los cálculos de vecindad, predicción y recomendación serán deficientes y darán como consecuencia recomendaciones de poca calidad.

Escalabilidad

Para obtener la similitud entre usuarios, los Sistemas de Recomendación Colaborativos utilizan algoritmos de cálculo del vecino más cercano (Knn). El problema de estos algoritmos es su elevado coste computacional, que crecerá de forma lineal respecto al número de elementos existentes en la base de datos.

Problema del ítem nuevo

Este problema repercute sobre los dos elementos principales del Sistema de Recomendación, los usuarios y los ítems. En esencia, un ítem nuevo en el sistema apenas tendrá puntuaciones respecto a otros ítems ya existentes, por lo que no van a ser recomendados prácticamente nunca. En cuanto a los usuarios, un usuario nuevo en el sistema habrá realizado pocas puntuaciones sobre los ítems existentes. Esto provoca que encuadrarlos en un grupo de vecinos adecuado sea una labor compleja, por lo que las recomendaciones que reciba serán pobres.

En los últimos tiempos se ha realizado una gran cantidad de experimentos, estudios e investigaciones de cara a reducir el impacto de estos problemas.

Para reducir el problema de la escasez se han utilizado técnicas de puntuaciones implícitas, correlación entre ítems y filtrado híbrido. Para tratar el problema de la escalabilidad se ha optado por una técnica muy empleada en el análisis de datos [13]: la reducción de la dimensionalidad, además de aproximaciones basadas en modelos. Finalmente, para solventar el problema del ítem nuevo se han propuesto técnicas de minería de datos Web, como por ejemplo las relativas a árboles de decisión.

Tras haber estudiado los principales problemas de los Sistemas de Recomendación Colaborativos, y las estrategias para reducirlos, vamos a ver las principales aplicaciones de los sistemas de recomendación.

2.4 Aplicaciones de los Sistemas de Recomendación

La investigación en SR se ha llevado a cabo mayoritariamente basándose en aplicaciones de utilidad práctica, muchas de ellas pertenecientes al ámbito comercial ya que, independientemente de la contribución teórica, dicha investigación está generalmente orientada a mejorar los resultados de estas aplicaciones, tal que la investigación en SR abarca aspectos prácticos que se aplican a la implementación de estos sistemas. Estos aspectos son relevantes en las distintas fases del ciclo de vida de un SR, en concreto, el diseño del sistema, su implementación, mantenimiento y mejora durante la operación del sistema.

Los aspectos que se aplican a la fase de diseño incluyen factores que pueden afectar a la elección del algoritmo. Un aspecto muy importante a considerar es el dominio de la aplicación, puesto que tiene el principal efecto en el enfoque algorítmico que debería elegirse. A este respecto podemos examinar la

lista de la Tabla 1, en la que aparecen dominios para las aplicaciones SR más comunes.

Dominio	Aplicaciones y características
Entretenimiento	Recomendación de música en función de las preferencias (LastFM), de un perfil social (Spotify) e incluso de los estados de ánimo (Sonzga), de contenido audiovisual basándose en sistemas colaborativos (FilmAffinity, NetFlix) o de libros (QueLibroLeo), etc.
Personalización de contenido	Bibliotecas digitales personalizadas (MyLibrary), periódicos individualizados, filtros de correo electrónico, etc.
Recomendación de contenido	Motores de búsqueda, recomendación de noticias (Google News), recomendación de páginas web (Stumble Upon), etc.
Comercio electrónico	Recomendaciones para los consumidores sobre qué comprar, cuyo ejemplo más claro es Amazon.
E-Learning y Educación	Recomendación de objetos de aprendizaje, itinerarios de aprendizaje, personalización de contenidos en cursos, etc.
Servicios	Servicios de viajes, qué casas alquilar, qué restaurantes visitar, qué expertos consultar, con quién tener una cita, etc.

Tabla 1. Dominios y aplicaciones más comunes de los Sistemas de Recomendación

2.4.1. Ejemplos de Sistemas de Recomendación Colaborativos

Tras estudiar los fundamentos teóricos de los Sistemas de Recomendación Colaborativos, pasamos a ver algunos ejemplos reales de aplicaciones que utilizan sistemas de recomendación. Debido a que en este proyecto se desarrolla una aplicación Android y para ver la importancia hoy en día de este sistema operativo se mostrará la versión Android de los mismos ejemplos.

Zagat

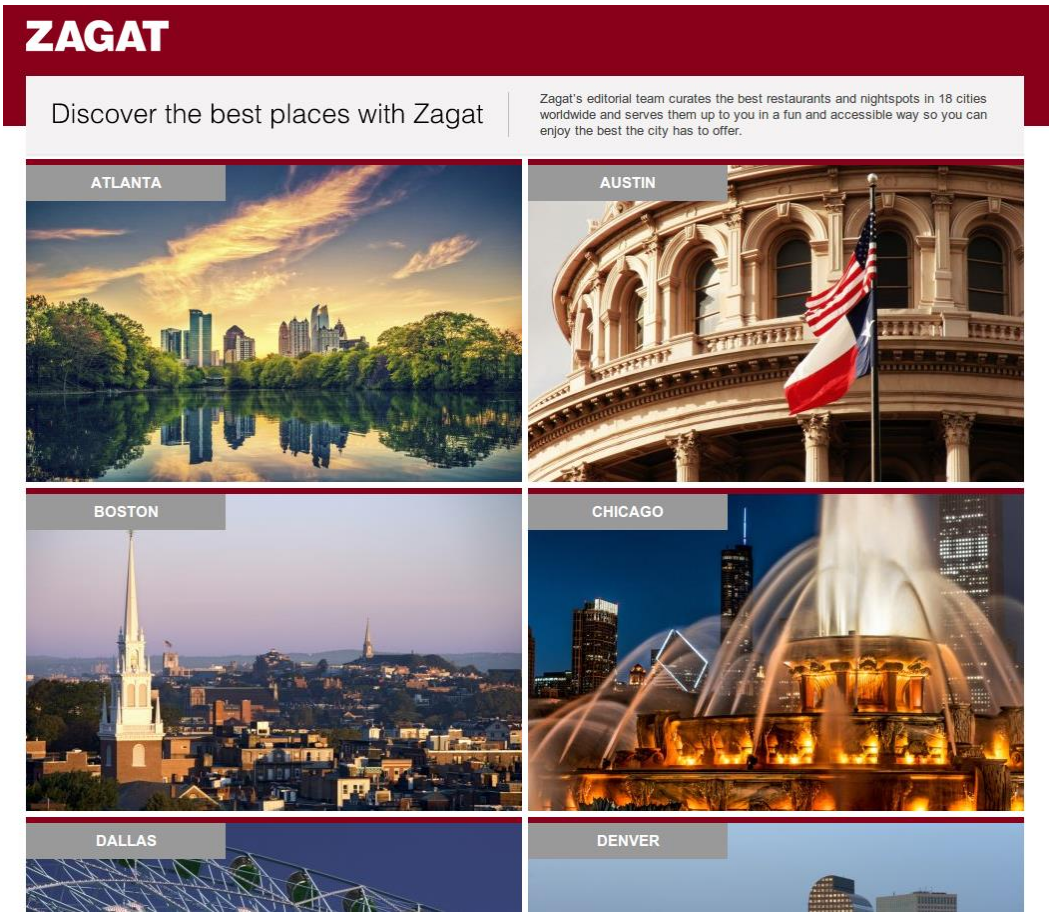

Figura 8: Portal de Zagat.com

Zagat [34] Survey es una empresa norteamericana fundada en 1979, dedicada a la edición de todo tipo de guías de restaurantes, tiendas, hoteles y clubes de diferentes ciudades de los Estados Unidos y Canadá. El usuario registrado puede valorar hasta 30 aspectos diferentes del local referido, además de introducir breves comentarios dando su opinión o experiencia en el mismo. A partir de estas valoraciones, los responsables de la empresa asignan sus puntuaciones en sus guías anuales, que servirán para realizar las recomendaciones a los usuarios a través de la Web (Véase figura 8) y de la aplicación Android (Véase figura 9).

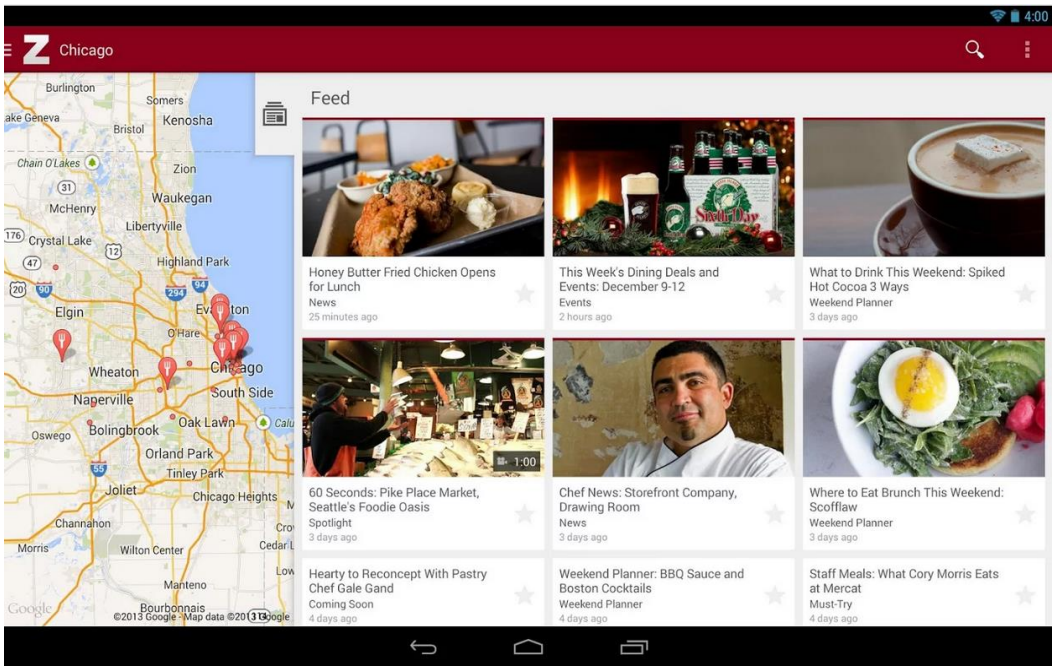

Figura 9: Aplicación Android de Zagat.com

Foursquare

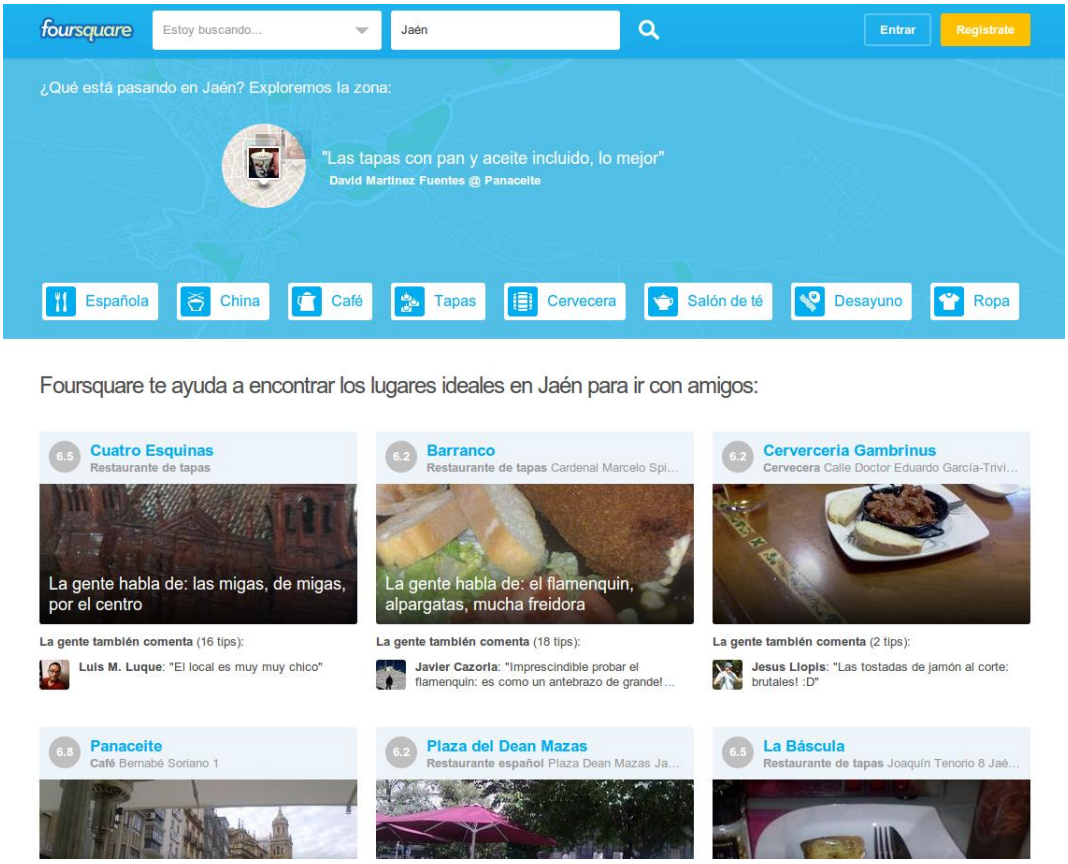

Figura 10: Portal de Foursquare.com

Los fundadores de Foursquare [35] lo definen como una red social basada en servicios de localización que incorpora elementos de juego (Véase figura 10).

Foursquare es la suma de tres conceptos:

1. Aplicación para teléfonos móviles con funcionalidad de ubicación.
2. Red social.
3. Permite registrar al usuario en cualquier tipo de lugar o espacio físico.

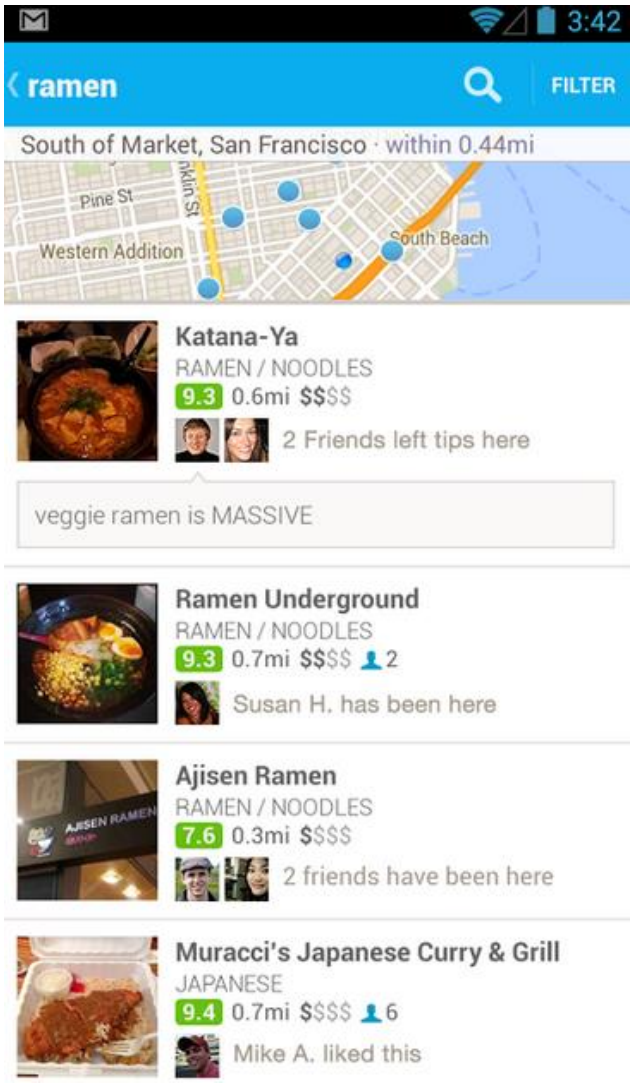

Figura 11: Aplicación Android de Foursquare.com

Desde la versión 3.0 de la aplicación (Véase figura 11) el motor está basado en filtrado colaborativo de tus propios check-ins y los check-ins de tus amigos. Para construir la matriz de similaridad de sitios emplean un clúster de 40 máquinas que consigue resolver 100 trillones de computaciones en apenas una hora. Su mayor reto fue el “arranque en frio”, es decir, que sitios recomendar al nuevo usuario que todavía no tiene muchos check-ins ni muchos amigos.

CAPÍTULO 3 Android y Servicios Web

3.1 Introducción

A lo largo de este capítulo se va a analizar el sistema operativo Android, incluyendo una descripción de las características más destacadas que hemos de conocer acerca del mismo para entender el desarrollo de la aplicación que se realiza en este proyecto.

Por otro lado se hará una introducción a los servicios, concretamente a los servicios web. Se indicarán las principales ventajas y desventajas de utilizar este tipo de servicios y las distintas tecnologías que se utilizan. Seguidamente se estudiarán más en profundidad los servicios REST debido a que va a ser la base tecnológica de nuestro sistema.

Por último se hará un breve análisis acerca de la música por internet, concretamente en Android, y veremos algunos ejemplos en Android.

3.2. Android

Android es un sistema operativo basado en Linux [15] para dispositivos móviles creado por Andy Rubin. En 2005 fue adquirido por Google Inc. [16] y Andy Rubin pasó a formar parte de la compañía como director de plataformas móviles para Google.

En el año 2007, cerca de 100 grandes compañías de varios sectores dentro de las telecomunicaciones, dispositivos móviles, semiconductores, desarrollo software y comercialización, formaron la Open Handset Alliance [17] con el objetivo de desarrollar estándares abiertos para móviles. El sistema operativo Android es una pieza fundamental de esta alianza, y parte de su código se encuentra liberado bajo la licencia Apache.

El primer lanzamiento de Android se llevó a cabo en noviembre de 2007, y en octubre del año siguiente se abrió Android Market para la descarga de aplicaciones. A partir de entonces, y debido a las características de código abierto, portabilidad y adaptación a cualquier tipo de hardware, optimización en el consumo de memoria o alta definición para gráficos y sonido [15], Android ha experimentado un gran crecimiento, hasta ser la plataforma dominante en cuanto a cuota de mercado. En la figura 12 podemos observar una comparativa de cuota de la cuota de mercado de los distintos sistemas operativos móviles en 2013:

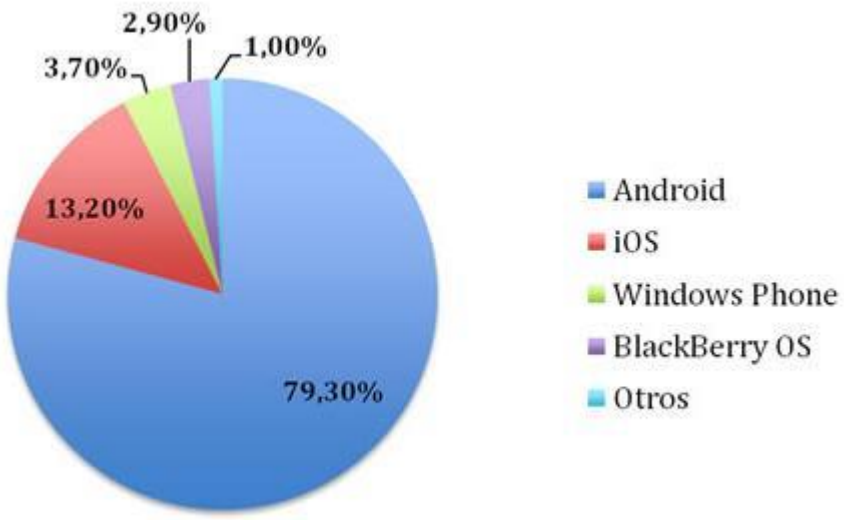

Figura 12: Cuota de mercado Sistemas Operativos 2013

Todas las características descritas anteriormente, así como la consolidación de Android como la plataforma líder para dispositivos móviles y la relativa facilidad a la hora desarrollar aplicaciones en cuanto a información disponible desde webs oficiales para desarrolladores han hecho que Android haya sido elegida como la plataforma idónea para la realización de este proyecto.

A continuación, estudiaremos brevemente la evolución de Android, que arquitectura sigue este sistema operativo y de que componentes está estructurado. Además analizaremos el ciclo de vida del elemento principal en Android, la actividad. Por último veremos el nivel de seguridad que dispone el sistema operativo, que tipos de almacenamiento utiliza y cuál es la estructura de un proyecto en Android.

3.2.1. Evolución de Android: versiones y tasa de distribución

El sistema operativo Android, desde su lanzamiento 2007, ha sacado a la luz un total de 19 versiones de la API del sistema operativo. Cada vez que lanza una nueva versión de la API añaden características nuevas y mejoran muchas características de versiones anteriores. En la tabla 2 podemos observar, la evolución en cuanto a versiones y nivel de API de la plataforma Android [15, 18]. Cada versión tiene un nombre comercial que hace referencia a un postre, siguiendo un orden alfabético.

Versión	Nombre	Logo	API
Android 1.0	Apple Pie		1
Android 1.1	Banana Bread (Petit Four)		2
Android 1.5	CupCake		3
Android 1.6	Donut		4

Android 2.0 / 2.1	Éclair		7
Android 2.2	Froyo		8
Android 2.3	Gingerbread		10
Android 3.0	Honeycomb		13
Android 4.0	Ice Cream Sandwich		15
Android 4.1	Jelly Bean		16
Android 4.2.2	Jelly Bean		17
Android 4.3	Jelly Bean		18
Android 4.4.3	Kitkat		19

Tabla 2. Versiones del sistema operativo Android

En cuanto a la tasa de distribución, como podemos observar en la figura 13, Jelly Bean, Gingerbread y KitKat son las más utilizadas. Esta información resulta de gran utilidad a la hora de desarrollar una aplicación, ya que es necesario indicar la versión mínima que nuestra aplicación puede soportar: esto significa que la aplicación no podrá ser instalada en dispositivos con versiones inferiores a la versión mínima escogida.

En nuestra aplicación Android se ha decidido establecer la API 15 como versión nueva, ya que existen ciertas características que fueron implementadas a partir de esta API y abarca prácticamente el 85% del mercado en la actualidad con esta limitación.

Version	Codename	API	Distribution
2.2	Froyo	8	0.8%
2.3.3 - 2.3.7	Gingerbread	10	14.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	12.3%
4.1.x	Jelly Bean	16	29.0%
4.2.x		17	19.1%
4.3		18	10.3%
4.4	KitKat	19	13.6%

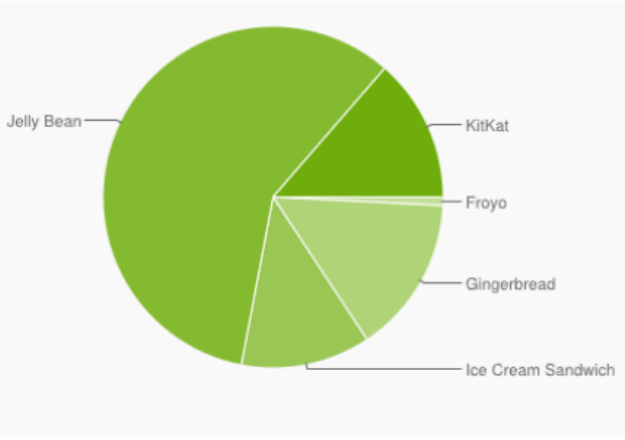

Figura 13: Tasa de distribución para las versiones de Android

3.2.2. Arquitectura

Como podemos ver en la figura 14, Android tiene una arquitectura dividida en cuatro capas [15]:

- *Núcleo*: Está formado por el sistema operativo Linux v2.6. Actúa, por tanto, como interfaz entre la aplicación y el dispositivo hardware, y proporciona servicios como la seguridad, el manejo de la memoria, el multiproceso, la pila de protocolos y el soporte de drivers para dispositivos. Es de código abierto bajo licencia GPL v2 [19].

- *Runtime*: Basado en el concepto del máquina virtual de Java (lenguaje utilizado para el desarrollo de aplicaciones para Android), Google desarrolló una nueva máquina virtual acorde a las limitaciones de los dispositivos móviles, llamada Dalvik.

- *Librerías nativas:* Son un conjunto de librerías en C/C++ compiladas en el código nativo del procesador. Estas librerías proporcionan acceso a las funcionalidades del sistema a través de la capa que explicaremos a continuación.

- *Entorno de la aplicación:* Pensada como una capa de acceso a las distintas APIs que permita reutilizar los componentes que la forman fácilmente, lo que simplifica el desarrollo de aplicaciones.

- *Aplicaciones:* Conjunto de aplicaciones instaladas en un dispositivo Android. Comprende tanto las instaladas en el dispositivo por defecto (tales como un cliente de correo electrónico, el calendario o la agenda de contactos) como las realizadas por otros desarrolladores disponibles que pueden ser descargadas a través de la plataforma Google Play.

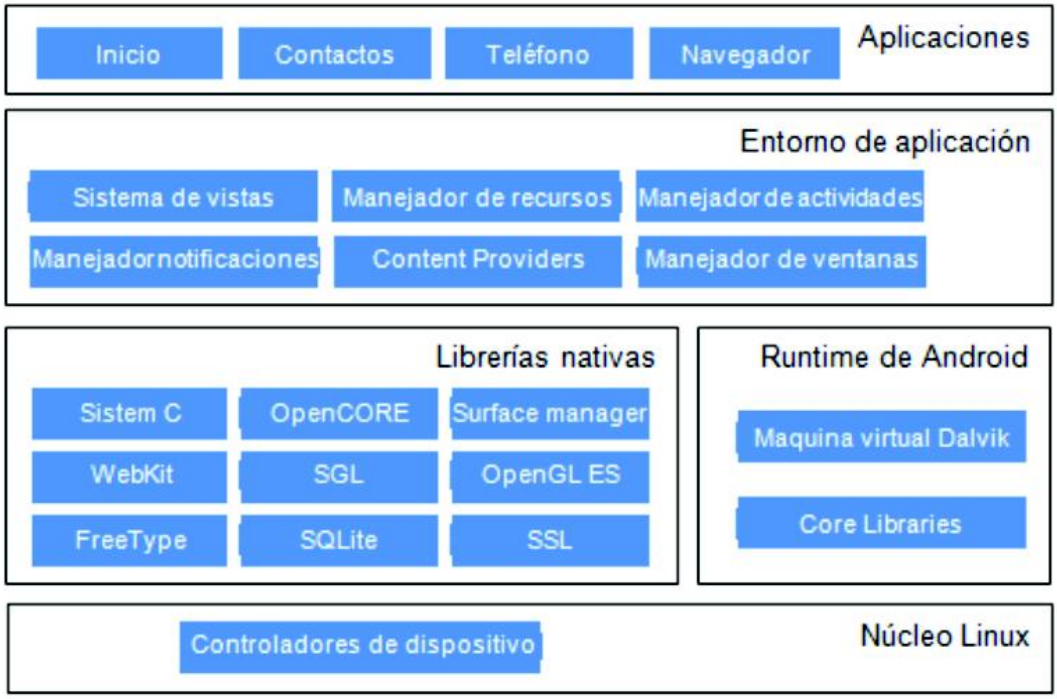

Figura 14: Arquitectura de Android

3.2.3. Componentes

Los componentes son elementos clave en el desarrollo de aplicaciones para Android. Cada componente representa un punto a través del cual el sistema puede acceder a la aplicación, y algunos de ellos también implementan la lógica de la aplicación tras la interacción del usuario con la interfaz.

Existen cuatro tipos de componentes en Android [15, 20]:

- *Actividades*: Una actividad implementa la lógica para una pantalla de la interfaz de usuario. Una aplicación, por tanto, estará formada por un conjunto de actividades independientes, aunque pueden interactuar entre ellas e intercambiar información.

- *Servicios*: Un servicio es un componente que se ejecuta en segundo plano y que permite realizar operaciones internas de la aplicación no asociadas a la interfaz de usuario.

- *Proveedores de contenido:* Permiten almacenar y compartir el conjunto de la información que manejan las aplicaciones.

- *Receptor de anuncios:* Permiten responder a anuncios de tipo broadcast. Estos anuncios pueden ser generados por el sistema o bien por una aplicación. No constan de interfaz, se ejecutan en segundo plano y suelen mostrar una notificación para anunciar la ocurrencia del evento.

Los componentes utilizados para el desarrollo de una aplicación han de ser declarados en el archivo AndroidManifest.xml. Cada aplicación tiene el suyo propio. El archivo AndroidManifest.xml permite al sistema operativo conocer toda la información esencial acerca de una aplicación antes de ejecutarla. Algunas de las que se incluye, además de los componentes, son los permisos con los que cuenta la aplicación, librerías, versiones soportadas, etc.

3.2.4. Ciclo de vida de una Actividad

El ciclo de vida de una aplicación Android es bastante diferente al ciclo de vida de una aplicación en otros sistemas operativos, como Windows. La mayor diferencia es que, en Android el ciclo de vida es controlado principalmente por el sistema, en lugar de ser controlado directamente por el usuario.

Es muy importante conocer bien las diferentes fases por las que pasa una aplicación, ya que hay ciertas actividades que sólo tienen sentido en determinadas fases del ciclo de vida de una actividad. En la figura 15 podemos ver todas las

fases por las que puede pasar una aplicación Android y en qué estado se encuentra la aplicación en cada momento.

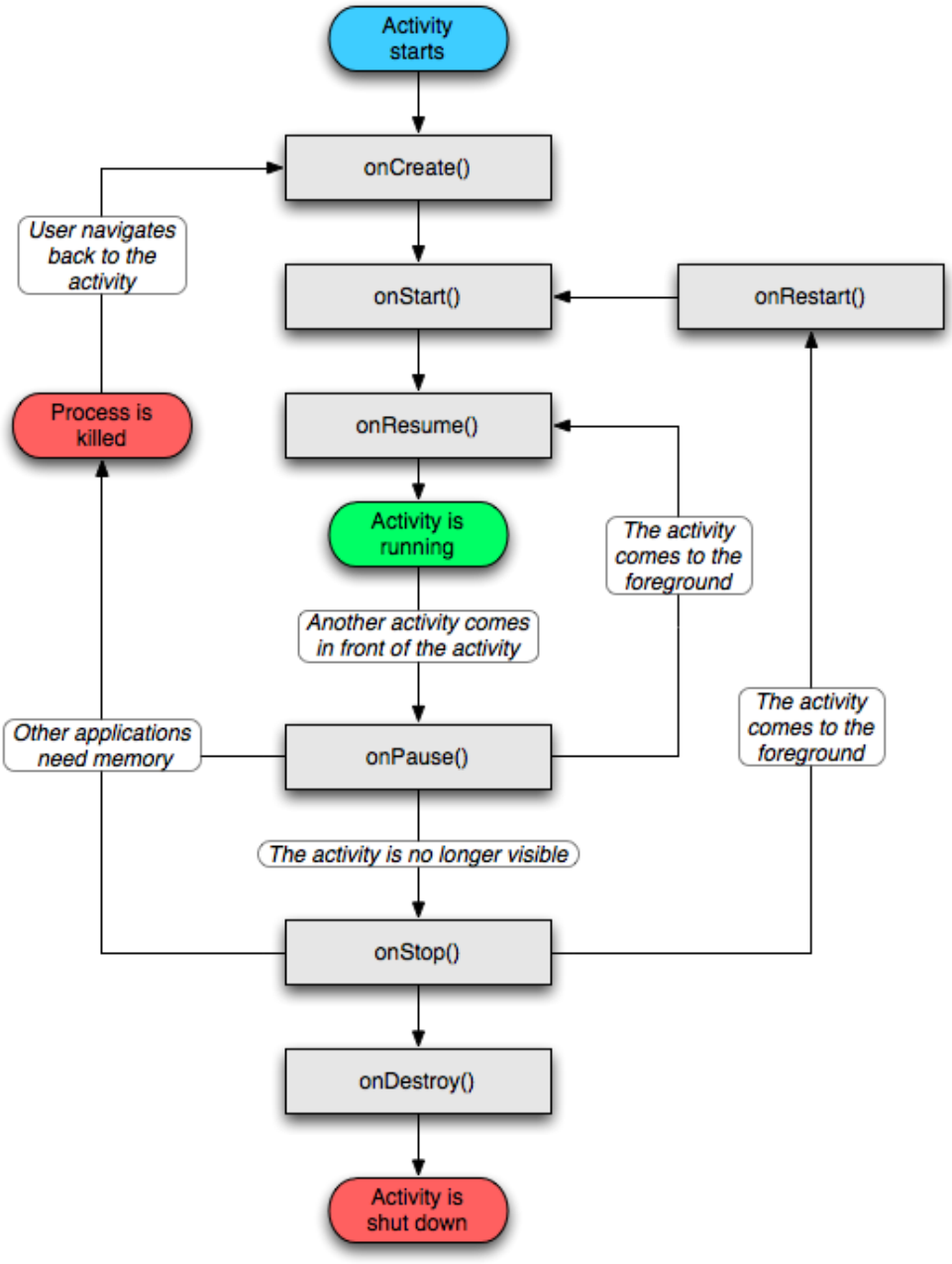

Figura 15: Ciclo de vida de una actividad Android

3.2.5. Seguridad

El sistema operativo Android en sí está basado en privilegios, y cada aplicación que se ejecuta en el sistema, está identificada bajo un identificador de usuario y un identificador de grupo. Además, todas las aplicaciones para Android han de estar firmadas a través de un certificado que permite identificar a su autor.

En cuanto a las aplicaciones en concreto, Android define un nivel de seguridad más, que les permite el acceso a las distintas funcionalidades del sistema. Para que una aplicación pueda hacer uso de ciertas características, protegidas del dispositivo, es necesario declarar los permisos pertinentes dentro del archivo AndroidManifest.xml [21].

3.2.6. Almacenamiento de información

Android proporciona distintas opciones para el almacenamiento de información. En función de las necesidades de nuestra aplicación hemos de considerar una u otra opción. Vamos a describir algunas de ellas [22]:

- *Preferencias compartidas*: Es una clase que permite el almacenamiento de las preferencias del usuario en nuestra aplicación, permitiendo almacenar los valores en pares de tipo clave/identificador y valor asociado a cada clave.

- *Almacenamiento interno*: Almacenamiento de datos en la memoria del dispositivo. Estos datos son de carácter privado, lo que significa que sólo nuestra aplicación puede acceder a ellos.

- *Almacenamiento externo***:** Los dispositivos compatibles con el sistema operativo Android soportan almacenamiento externo. Los archivos guardados en este tipo de almacenamiento son accesibles desde fuera de nuestra aplicación y pueden ser modificados.

- *Bases de datos*: Android proporciona soporte para bases de datos SQLite. Todas las bases de datos creadas desde nuestra aplicación son accesibles desde cualquier clase de la misma, pero no por otras aplicaciones.

3.2.7. Estructura de un proyecto Android

Al crear un nuevo proyecto de Android en la herramienta de desarrollo nos aparecerán ciertos archivos y carpetas ya creados que conforman nuestra aplicación. Es necesario realizar una introducción sobre las principales carpetas ya que es fundamental para entender el desarrollo en Android.

Un proyecto de una aplicación para Android está formado por una jerarquía de directorios y archivos que conforman el esqueleto básico de la aplicación (Véase figura 16).

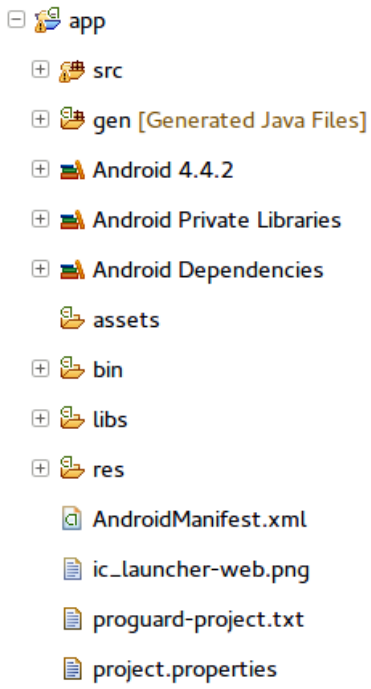

Figura 16: Jerarquía de directorios de una aplicación Android

A continuación, vamos a introducir brevemente la utilidad y el contenido de cada uno de los distintos archivos y directorios:

- **src**: Contiene el código fuente que describe la lógica de la aplicación. Todas las clases Java que la componen se almacenan en este directorio, agrupadas en distintos paquetes o dentro del paquete por defecto que contiene.

- **gen**: La información que contiene este directorio es generada automáticamente por el compilador, y no ha de ser modificada por el programador bajo ningún caso. Almacena ficheros con información relativa a las opciones de configuración del proyecto o de referencias que permiten conectar los elementos entre los distintos directorios.

- **Android 4.4.2**: Contendrá una librería android.jar que permitirá dar acceso a las APIs según las versiones elegidas.

- **Android Private Libraries**: Contiene varios archivos .jar que serán las librerías que hemos importado en nuestro proyecto.

- **Android Dependencies**: Es una carpeta virtual donde Eclipse muestra los archivos JAR de los que depende nuestro proyecto.

- **assets**: En este directorio es un repositorio de archivos en el que podemos incluir cualquier tipo de fichero externo que pueda ser utilizado por la aplicación en un momento determinado.

- **res**: Está compuesto por varios subdirectorios que almacenan distintos tipos de recursos como se describe en la figura 17:

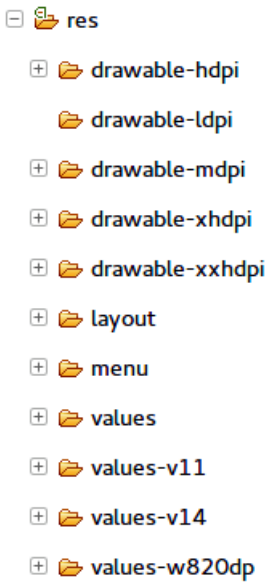

Figura 17: Directorio res y sus subdirectorios en un proyecto Android

- **drawable**: Contiene a los iconos utilizados en la aplicación. Cada una de los directorios drawable hace referencia a una densidad o tamaño de la pantalla del dispositivo que ejecuta la aplicación.
- **layout**: Contiene los archivos .xml en los que se han definido las distintas pantallas o interfaces de la aplicación.
- **menu**: Contiene los archivos .xml en los que se define la interfaz de las barras de menú utilizadas en la aplicación.
- **values**: Contiene los archivos dimens.xml, strings.xml y styles.xml que hacen referencia a dimensiones, cadenas de texto y estilos de diseño utilizados por la aplicación respectivamente.

- **AndroidManifest.xml:** Descrito en el apartado 3.1.3.
- **proguard-project.txt, project.properties**: Archivos de configuración generados automáticamente por el compilador que describen ciertas propiedades o características del proyecto. No deben ser modificados por el programador.

3.3. Servicios Web

En el contexto de la arquitectura empresarial, la orientación a servicios y la arquitectura orientada a servicios, el servicio se puede definir como [23]:

El Conjunto de funcionalidades de software relacionadas que pueden ser reutilizadas para fines diferentes, junto con las políticas que deben controlar su uso.

Por otro lado OASIS (Organización para el Avance de Estándares de Información Estructurada) define un servicio como [23]:

Mecanismo para permitir el acceso a una o más capacidades, en los que se presta el acceso mediante una interfaz y se ejerce de conformidad con las limitaciones y las políticas como se especifica en la descripción del servicio.

En el ámbito de este proyecto nos referimos concretamente a servicios web, ya que la comunicación entre cliente y servidor se realiza a través de internet. Un servicio web se puede definir como [24]:

Componente de software que utiliza un conjunto de protocolos y estándares para intercambiar datos entre aplicaciones sobre una red.

Los servicios web suelen ser considerados como APIs Web que pueden ser accedidos dentro de una red (principalmente internet) y ejecutados en el sistema que los aloja [25] (Véase figura 18).

Las características de los servicios web son:

- Son interoperables
- Superan las barreras geográficas
- Por naturaleza son flexibles
- Se basan en el protocolo HTTP

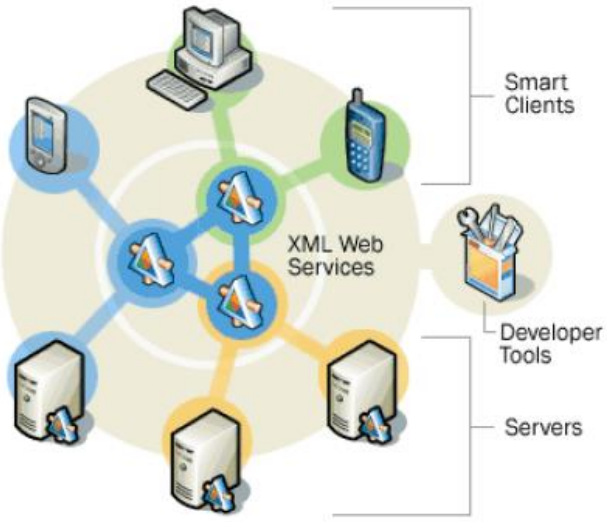

Figura 18: Esquema de un servicio web

A continuación, vamos a ver las ventajas y desventajas de los servicios web, una breve introducción a las tecnologías que utilizan los servicios web y por último haremos un análisis de los servicios REST ya que son la base tecnológica de este proyecto.

3.3.1. Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.

- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

3.3.2. Desventajas de los servicios web

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).

- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (Remote Method Invocation), CORBA o DCOM (Distributed Component Object Model). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.

- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

3.3.3. Tecnologías empleadas por los servicios web

Existen multitud de tecnologías para implementar un servicio web, en general las más utilizadas hoy en día son SOAP y REST. Vamos a ver en qué consisten [26].

SOAP (Simple Object Access Protocol)

SOAP es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP (Véase figura 19). Está basado en XML, esto facilita la lectura, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir.

Existen múltiples tipos de modelos de mensajes en SOAP pero, por lejos, el más común es el RPC, en donde un nodo de red (el cliente) envía un mensaje de solicitud a otro nodo (el servidor) y el servidor inmediatamente responde el mensaje al cliente.

Los mensajes SOAP, son independientes del sistema operativo, y pueden transportarse en varios protocolos de internet como SMTO, MIME y HTTP.

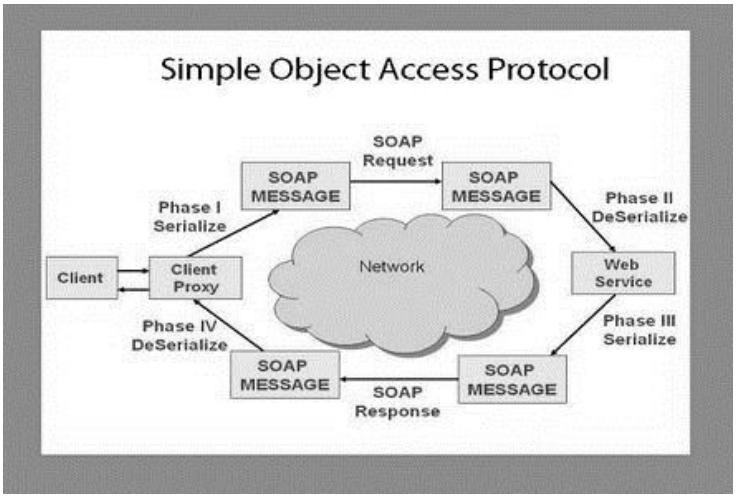

Figura 19: Funcionamiento de SOAP

REST (Representational State Transfer)

REST es un estilo de arquitectura de software dirigidos a sistemas hipermedias distribuidos como lo es la Web y se refiere específicamente a una colección de principios (los cuales resumen la forma en que los recursos son definidos y diseccionados) para el diseño de arquitecturas en red (Véase figura 20).

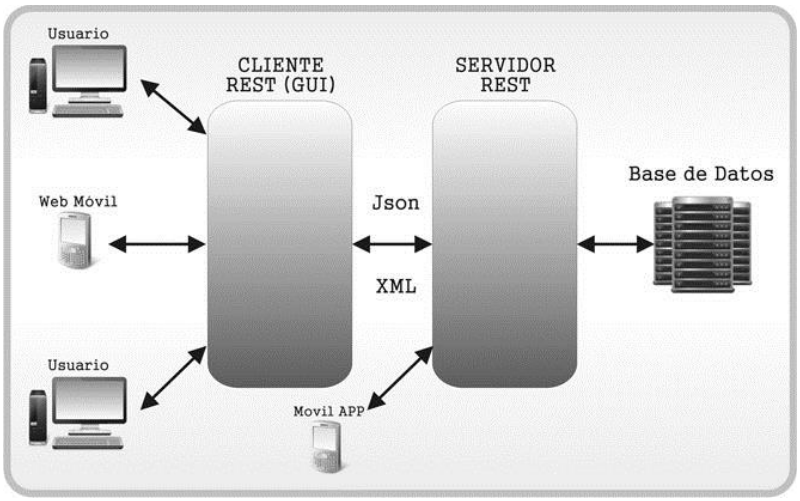

Figura 20: Funcionamiento de REST

En el siguiente apartado nos vamos a centrar en los servicios REST ya que es la tecnología en la que se basa este proyecto.

3.3.4. Servicios REST

Este término es utilizado en su mayoría para describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP sin una capa adicional, como lo hace SOAP, en tal sentido estos dos significados pueden chocar o incluso solaparse.

Ahora es importante entender que es posible diseñar un sistema software de gran tamaño de acuerdo con esta arquitectura propuesta sin utilizar HTTP o sin interactuar con la Web, así como también diseñar una simple interfaz XML+HTTP que no sigue los principios REST, y en cambio seguir un modelo RPC.

Es de esta forma que es importante remarcar el hecho de que REST no es un estándar, ya que es tan sólo un estilo de arquitectura, pero también está basado en los siguientes estándares:

- HTTP
- URL
- Representación de los recursos: XML/HTML/GIF/JPEG/…
- Tipos MIME: text/xml, text/html, ...

La arquitectura de REST tiene que cumplir con estos 6 principios.

- *Cliente-servidor*: Define que deben de estar separados el cliente del servidor a través de interfaces uniformes, es decir el cliente no sabe nada de cómo se almacena la información, ni como se está obteniendo. Por otro lado los servidores no saben la manera en la que se está presentado la información.

- *No maneja estado*: El servidor no debe de contener ningún contexto sobre el cliente que está haciendo la solicitud. La solicitud del cliente debe tener toda la información necesaria para poder procesar la solicitud en el servidor, esto permite crear aplicaciones más escalables sin que tener preocupación sobre cómo debe de responder el servidor a la pérdida de la sesión del cliente por pérdida de conectividad.

- *Capaz de almacenarse en caché*: Los clientes no tienen un mecanismo de almacenar las respuestas en caché. Las respuestas deben definirse como almacenables en caché, para evitar que los clientes hagan uso inapropiado de información devuelta por una solicitud.

- *Sistema en capas*: El cliente no debe de saber si está conectado directamente a un servidor final o a un intermediario. Un servidor intermediario puede ayudar a balancear las cargas y la escalabilidad de la aplicación.

- *Código bajo demanda:* Los servidores pueden ser capaces de extender la funcionalidad de un cliente transfiriéndole lógica que puedan ejecutar, por ejemplo Java Applets o JavaScript.

- *Interface Uniforme*: Son recursos individuales que deben de estar incluidos dentro de la solicitud.

Las *ventajas* de los servicios REST son las siguientes:

- Bajo consumo de recursos.
- Las instancias del proceso son creadas explícitamente.
- El cliente no necesita información de enrutamiento a partir de la URI inicial.
- Los clientes pueden tener una interfaz “listener” (escuchadora) genérica para las notificaciones.
- Generalmente fácil de construir y adoptar.

Y las *desventajas* que podemos encontrar son:

- Gran número de objetos.
- Manejar el espacio de nombres (URIs) puede ser engorroso.
- La descripción sintáctica/semántica muy informal (orientada al usuario).
- Pocas herramientas de desarrollo.

A continuación se introducirán los métodos HTTP y los códigos de estado HTTP más comunes. Posteriormente, se analizará cómo se estructuran las URLs en REST.

3.3.4.1. Métodos y códigos de estado HTTP

Una API RESTful bien diseñada debería soportar los métodos HTTP más comunes (GET, POST, PUT y DELETE). Existen otros métodos HTTP como OPTIONS o HEAD, pero no suelen ser usados normalmente. Cada método debería usarse dependiendo del tipo de operación que se quiera realizar.

Método	Acción
GET	Obtener un recurso
POST	Crear un recurso nuevo
PUT	Actualizar un recurso existente
DELETE	Eliminar un recurso

Tabla 3. Métodos HTTP

Los códigos de estado HTTP en el cuerpo de la respuesta indican a la aplicación cliente que acción debería realizar con la respuesta. Por ejemplo si el código de la respuesta es 200 significa que la petición al servidor se ha procesado correctamente. De la misma manera si el código de respuesta es el 401, la petición realizada no estaba autorizada.

Código de estado	Significado
200	Correcto
201	Creado
304	No modificado
400	Petición incorrecta
401	No autorizado
403	Olvidado
404	No encontrado
422	Entidad no procesable
500	Error interno del servidor

Tabla 4. Códigos de estado HTTP

3.3.4.2. Estructura de las URLs

En REST el diseño de las URLs debería estar bien formado y debería entenderse fácilmente. Cada URL para cada recurso debería ser identificado de forma única. Si la API necesita ser accedida con clave de API, esta debería estar guardada en los headers HTTP.

Ejemplo:

- GET http://pagina.com/v1/tasks/11 => Devuelve los detalles del proceso (task) con id 11.
- POST http://pagina.com/v1/tasks => Creará un nuevo proceso (task).

3.4. Música en Android: Streaming

Streaming es un término utilizado para denominar la tarea de ver u oír un archivo multimedia (vídeo, audio…) de forma directa en una aplicación móvil, sin necesidad de descargar dicho archivo en el dispositivo móvil. Se podría describir como “hacer un clic y obtener”. En términos más específicos, puede definirse como:

Una estrategia sobre demanda para la distribución de contenido multimedia a través de Internet.

Así, el Streaming de audio hace posible escuchar música a través de una aplicación móvil sin necesidad de descargar los archivos musicales a escuchar, ya que estos quedan almacenados en un buffer en el instante en que están siendo reproducidos.

Antes de que la tecnología Streaming apareciese en el año 1995, con el lanzamiento del sistema de transmisión de música online Real Audio 1.0, era necesario descargar el archivo de audio en el disco duro del usuario para la reproducción de música existente en Internet. Dado que estos archivos suelen tener un tamaño significativo, su descarga en el equipo podía suponer un proceso lento, especialmente en conexiones con límite de descarga no muy elevado.

Sin embargo, el Streaming permite descargar el archivo y reproducirlo al mismo tiempo, mientras aún está descargándose, consiguiendo así un tiempo de espera mínimo. Esta es la principal ventaja de esta tecnología, y lo que la ha hecho tan popular en los últimos años.

El funcionamiento del Streaming es el siguiente:

- En primer lugar, el dispositivo del usuario (cliente) se conecta con el servidor (proveedor del sitio Web) y éste inicia el envío del fichero multimedia en cuestión.

- El cliente comienza a recibir el fichero y en ese preciso instante construye un buffer de memoria donde empieza a guardar la información.

- Cuando se ha llenado el buffer con una pequeña parte del archivo, el dispositivo cliente empieza a reproducirlo, a la vez que continúa con la descarga. Normalmente el intervalo de tiempo necesario para ello es muy pequeño, casi nulo.

- El sistema está sincronizado para que el archivo se pueda reproducir mientras que el archivo se descarga, de modo que cuando el archivo acaba de descargarse el fichero ya lleva cierta cantidad de tiempo en reproducción.

- Si en algún momento la conexión sufre descensos de velocidad se utiliza la información que hay en el buffer, de modo que se pueda dar un margen temporal de tolerancia a dicho descenso en la conexión. Si la comunicación se cortase demasiado tiempo, el buffer se vaciaría y la ejecución el archivo se cortaría también hasta que se restaurase la señal.

Hoy en día podemos encontrar multitud de aplicaciones móviles y en concreto Android que saca partido a la tecnología Streaming (Véase figura 21). Un ejemplo de ello lo tenemos en las aplicaciones de radio o radios online, concepto en torno al cual gira el objetivo de este proyecto. Se trata aplicaciones Android en

las que el usuario tiene acceso a una colección de archivos musicales para escuchar sin necesidad de descargarlos en su dispositivo móvil, además de disponer de un perfil personal, etc. Muchas de estas aplicaciones de radio online trabajan incluso sobre la emisión de audio en vivo, como ocurre con las aplicaciones oficiales de las emisoras de radio.

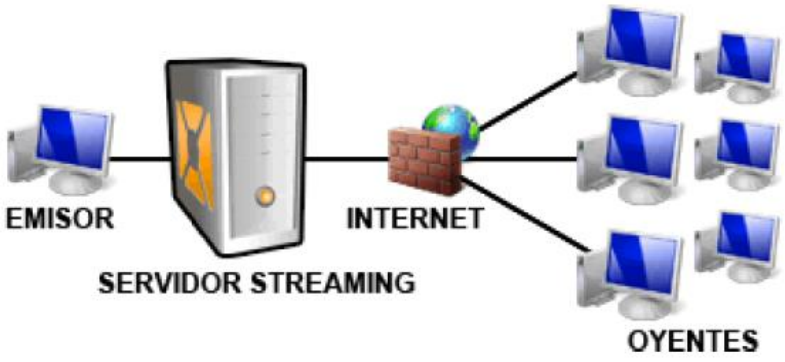

Figura 21: Ejemplo de estructura de audio-Streaming en directo

3.4.1. Ejemplos de música Streaming en Android

Last.fm

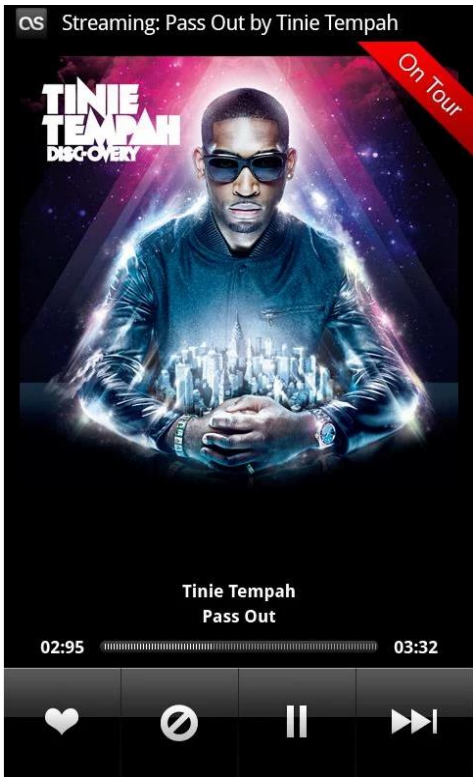

Figura 22: Aplicación Android de Last.fm

Last.fm es una famosísima red social, una radio online y además un Sistema de Recomendación Colaborativo de música, que se basa en los datos de los usuarios registrados para construir perfiles y estadísticas de gustos musicales, para realizar las recomendaciones. El servicio de que dispone es de código abierto. Se originó a partir de un proyecto de informática en la Universidad de Southampton (Reino Unido). La aplicación Android (Véase figura 22) tiene entre 1.000.000 y 5.000.000 de descargas en Google Play.

Pandora

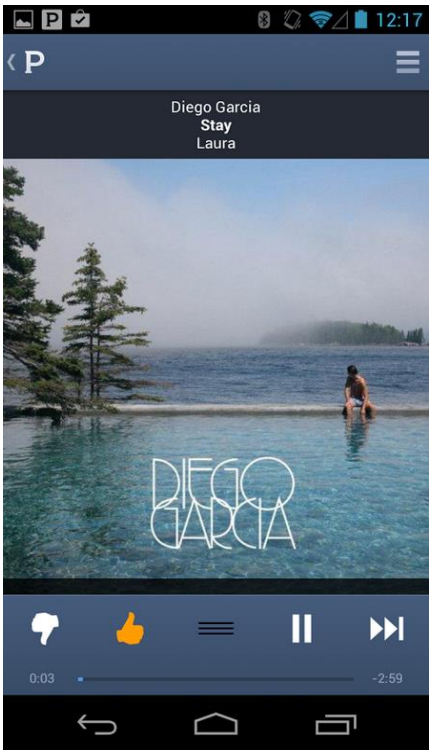

Figura 23: Aplicación Android de Pandora

Pandora es una radio online que también se basa en un algoritmo de filtrado colaborativo, recomendando a los usuarios canciones que son del agrado de otros usuarios con gustos similares. Su gran punto fuerte respecto a otras radios es la interfaz, muy intuitiva y fácil de utilizar. El usuario se crea su propia emisora de radio, con la ayuda opcional de un asistente en la que posteriormente puede añadir y editar canciones escogidas por él, además de aquellas canciones que le sean recomendadas. La aplicación Android (Véase figura 3.12) tiene entre 100.000.000 y 500.000.000 de descargas en Google Play estando disponible solamente en Estados Unidos.

CAPÍTULO 4 Ingeniería del software

4.1. Descripción del Proyecto

Una vez se han sentado las bases teóricas de los Sistemas de Recomendación, así como Android y los Servicios Web, es el momento de entrar con profundidad en el desarrollo de este proyecto.

A grandes rasgos, el proyecto que se ha realizado consiste en el desarrollo de una aplicación de una radio por Internet basada en un algoritmo de filtrado colaborativo, para recomendar canciones a los usuarios, siguiendo para todo ello una arquitectura cliente/servidor. Esta aplicación de radio colaborativa consta de cuatro componentes fundamentales, que suponen los cuatro pilares básicos del desarrollo del proyecto:

- Una base de datos, que incluye todos los datos relativos a usuarios del sistema, canciones, puntuaciones de los usuarios sobre las canciones, predicciones…

- Una interfaz para la aplicación, en forma de aplicación Android, desde la que los usuarios accederán al sistema, escucharán y evaluarán canciones, y recibirán recomendaciones de música a modo de listas de reproducción.

- Un algoritmo de filtrado colaborativo, que se encarga de calcular las predicciones a partir de los datos de puntuaciones, usuarios y canciones existentes en la base de datos.

- Un servicio Web para la comunicación entre el servidor y la aplicación Android.

Tras haber expuesto una presentación y una visión general del proyecto, con su propósito y objetivos [27], y una introducción teórica a los Sistemas de Recomendación, al ecosistema Android y a los servicios Web, es el momento de detallar, en base a técnicas de Ingeniería del Software, el proyecto realizado. La razón de ello es sencilla: estamos ante un proyecto de desarrollo software y, como tal, hemos de seguir la metodología de la Ingeniería del Software.

No existe una definición única y estandarizada para la Ingeniería del Software. Pese a ello, las dos que pasamos a exponer a continuación resultarán perfectamente válidas para nuestro acometido.

La Ingeniería del Software consiste en la construcción de software de calidad con un presupuesto limitado y un plazo de entrega en contextos de cambio continuo.

La Ingeniería del Software es el establecimiento y uso de principios y métodos firmes de ingeniería, para obtener software económico que sea fiable y funcione de manera eficiente en máquinas reales.

La Ingeniería del Software se compone de las siguientes actividades [28]:

- Especificación de requerimientos: Se obtiene el propósito del sistema, así como las propiedades y restricciones impuestas sobre el mismo.

- Análisis del sistema: Se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.

- Diseño del sistema: Se establecen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- Implementación: Consiste en la traducción del modelo lógico del sistema a código fuente.

- Pruebas: Verificar y validar el sistema.

En adelante profundizaremos en cómo se han llevado las citadas etapas durante el desarrollo de nuestro proyecto.

4.2. Especificación de Requerimientos

El primer paso en el proceso de Ingeniería del Software debe ser determinar el propósito último del proyecto, las propiedades que debe satisfacer y las restricciones a las que ha sido impuesto. Este es, sin duda, un paso de vital importancia dentro del desarrollo de un proyecto software, ya que sin conocer el propósito del mismo ni establecer las diferentes limitaciones que debe afrontar, será realmente difícil realizar una aplicación software capaz de cumplir dicho propósito.

En un proyecto de ámbito comercial para una empresa real, la forma de determinar el propósito es mediante una serie de estudios como pueden ser: entrevista con los clientes, encuestas con posibles usuarios, observaciones en vivo de la actividad diaria en la empresa, estudios de viabilidad o análisis de la situación actual de la empresa. En nuestro caso, estamos ante un proyecto de ámbito académico, por lo que el propósito es conocido desde la concepción del mismo:

Diseño y desarrollo de una radio online colaborativa para Android, basada en las preferencias musicales del usuario, y un servicio que ofrezca toda la información a la aplicación.

Tras haber establecido el propósito último del proyecto, el siguiente paso consiste en especificar los requerimientos del mismo. Los requerimientos de un proyecto software son el conjunto de propiedades o restricciones, definidas de forma totalmente precisa, que dicho proyecto ha de satisfacer. Existen dos tipos bien diferenciados de requerimientos:

- Requerimientos funcionales: Aquellos que se refieren expresamente al funcionamiento del sistema.

- Requerimientos funcionales: Son todos aquellos requerimientos no referidos al estricto funcionamiento del sistema, sino a otros factores externos.

En los dos epígrafes siguientes, definiremos estos requerimientos (tanto funcionales como no funcionales), para el proyecto sobre el que versa la presente memoria.

4.2.1. Requerimientos funcionales

Como hemos dicho, los requerimientos funcionales de un sistema software son aquellos que describen las funcionalidades que dicho sistema debe proporcionar a los usuarios del mismo, para cumplir su objetivo.

Al estar ante un proyecto de tipo académico, no disponemos de cliente alguno para obtener los requerimientos, por lo que nos basamos en otros Sistemas de Recomendación de música existentes en el mercado, de reconocido éxito, para establecer los requerimientos.

Los requerimientos funcionales serán:

1. *Darse de alta en el sistema:* El sistema debe proporcionar un mecanismo para que el usuario pueda registrarse en el sistema, de cara a ofrecerle un servicio personalizado.

2. *Escuchar la radio personalizada:* El sistema debe ser capaz de ofrecer nuevas recomendaciones a los usuarios y permitir al usuario acceder a una lista de reproducción fácilmente utilizable para poder escuchar las canciones que desee.

3. *Escuchar la radio normal:* El sistema debe ser capaz de ofrecer una serie de canciones aleatorias y permitir al usuario acceder a una lista de reproducción fácilmente utilizable para poder escuchar las canciones que desee.

4. *Escuchar la radio favoritos:* El sistema debe ser capaz de ofrecer una serie de canciones favoritas del usuario y permitir al usuario acceder a una lista de reproducción fácilmente utilizable para poder escuchar las canciones que desee.

5. *Puntuar canciones:* El sistema ha de permitir al usuario evaluar numéricamente canciones, ya sean canciones nuevas para él, o bien canciones ya evaluadas cuya puntuación desee modificar.

6. *Marcar canciones como favoritas:* El sistema ha de permitir al usuario indicar canciones como favoritas o como no favoritas en caso de que ya lo sea.

7. *Actualizar el modelo de recomendaciones:* El sistema debe actualizar su modelo de recomendación cada cierto tiempo, incorporando las nuevas

puntuaciones que los usuarios hayan hecho sobre las canciones.

8. *Desvincular cuenta de Google:* El sistema debe ser capaz de ofrecer al usuario una forma de desvincular su cuenta de Google del propio sistema.

4.2.2. Requerimientos No Funcionales

Los requerimientos no funcionales son las restricciones impuestas a los requerimientos funcionales del sistema. Pese a que en principio puedan no parecer demasiado relevantes, son tan importantes como los requerimientos funcionales y, en muchos casos, pueden incluso llegar a ser críticos para la aceptación del sistema. Normalmente, estos requerimientos especifican propiedades del sistema software en sí (velocidad, rendimiento…) y de la interfaz de usuario, además de todas las restricciones impuestas por la organización (plazo, legalidad vigente, política empresarial…).

Obviamente, al no estar ante un proyecto de tipo empresarial o comercial, no hay necesidad de someterse a restricciones organizacionales. Así, los requerimientos no funcionales deben obtenerse y analizarse a partir de las necesidades hardware y software de los equipos informáticos, para dar al usuario la funcionalidad requerida de forma eficiente, y de la interfaz gráfica entre el usuario y la aplicación.

Requerimientos del Equipo Informático

Dado que nuestro proyecto se sustenta sobre una arquitectura cliente/servidor, será necesario distinguir entre los requerimientos del equipo cliente y los requerimientos del servidor. A parte se necesitará un ordenador de desarrollo de similares características al equipo servidor.

Los **requerimientos del Equipo Informático del cliente** son bastante sencillos. Tan sólo es necesario un dispositivo móvil con el sistema operativo Android 4.0, con conexión a Internet (preferiblemente Wifi, aunque puede usarse con la conexión de datos) y que el sistema operativo tenga instalado Google Play Services para el inicio de sesión.

Los requerimientos del Equipo Informático del servidor, son bastante más ambiciosos y detallados, por lo que los dividiremos en dos grupos: requerimientos de hardware y requerimientos de software.

- *Hardware del Equipo Servidor*
  - Tipo de Procesador: Intel Xeon X3320 Quad Core a 2,50 GHz
  - Memoria caché interna: 6 MB de caché de nivel 2.
  - Bus del sistema: Bus frontal de 1333 MHz
  - Memoria de serie: 4 GB de memoria estándar.
  - Unidad de disco duro interna: 1 unidad SATA 250GB, 7.200 rpm
  - Almacenamiento masivo interno: 1 TB
  - Interfaz de red: Adaptador de servidor NC105i Gigabit incorporado.

- *Software del Equipo Servidor*
  - Sistema Operativo: Ubuntu, distribución libre de Linux.
  - Máquina virtual: Java v1.6.
  - PHP: Versión 5.0.
  - Navegador: Última versión del navegador Mozilla Firefox.
  - Sistema Gestor de Bases de datos: MySQL.
  - Servidor Web: Apache 2.

Parte del software necesario será proporcionado al administrador de la aplicación, el cual dispone de un manual para su instalación en el Anexo A.

4.2.3. Requerimientos de la Interfaz

Los requerimientos de la interfaz gráfica entre la aplicación y el usuario están estrechamente ligados a la usabilidad y sus principios [29]. La usabilidad se puede definir de varias formas:

Usabilidad se define como la medida en que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso específico.

Usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

La usabilidad puede definirse de forma coloquial como facilidad de uso, ya sea de una página Web, una aplicación informática o cualquier otro sistema que interactúe con el usuario.

Partiendo de estas tres definiciones, se pueden obtener los principios básicos de la usabilidad, los cuales se asociarán a los requerimientos no funcionales que deberá cumplir la interfaz gráfica:

- *Facilidad de aprendizaje:* Se refiere a aquellas características de la interfaz que permiten a los usuarios noveles comprender cómo usarla inicialmente y cómo obtener un máximo grado de productividad. Para que un sistema sea fácil de aprender debe reunir las siguientes características.
    - Predecible: El conocimiento de la historia de una interacción debe ser suficiente para determinar el resultado de una interacción futura.
    - Síntesis: El usuario debe poder captar fácilmente los cambios de estado del sistema.

- Familiaridad: Es importante diseñar el sistema de forma que los elementos de su interfaz resulten conocidos y familiares para el usuario.

- Generalización: Las tareas semejantes deben ser resueltas de modo parecido.

- Consistencia: Todos los mecanismos que se utilizan deben ser usados siempre de la misma manera, sea cual sea el momento en que se utilicen. Esto se consigue, por ejemplo, siguiendo guías de estilo y diseñando con un look & feel común.

- *Flexibilidad*: Este principio de usabilidad establece que debe haber varias formas en que el sistema y el usuario intercambian la información. Las características que hacen a una interfaz flexible son:

  - Control del diálogo por parte del usuario: Hay que proporcionar al usuario la capacidad, siempre que sea posible, para decidir cuándo empezar u acabar las operaciones.

  - Migración de tareas: Tanto el usuario como el sistema deben poder realizar una tarea en exclusiva, o pasarla al otro, o realizarla de forma conjunta.

  - Capacidad de sustitución: La capacidad de sustitución permite que valores equivalente puedan ser sustituidos los unos por los otros. Un ejemplo de ello es el del ancho de página en un procesador de textos, que puede expresarse en centímetros o en pulgadas.

  - Capacidad de adaptación: La interfaz debe poder adaptarse automáticamente a las necesidades del usuario actual.

- *Robustez:* Es el nivel de fiabilidad del sistema, o el grado en que el sistema es capaz de tolerar fallos durante la interacción. Las siguientes características hacen posible la robustez:
    - Navegabilidad: El usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.
    - Uso de valores por defecto: Ayudan al usuario mediante recuerdo pasivo. Mostrar un valor por defecto ayuda a que el usuario sepa qué tipo de valor debe introducir.
    - Persistencia: Un sistema persistente es aquel en el que las notificaciones al usuario permanecen como objetos manipulables después de su presentación.
    - Recuperación de información: La aplicación debe permitir el regreso a un estado anterior, tras algún error en una interacción previa.
    - Tiempo de respuesta: Es el tiempo que necesita el sistema para reflejar los cambios realizados por el usuario.

4.3. Análisis del Sistema

Una vez conocido, el propósito del proyecto software, las propiedades de este y las restricciones a las que debe someterse, es el momento de analizar el sistema y crear un modelo del mismo que sea correcto, completo, consistente, claro y verificable. Para ello, se definirán los casos de uso según los requerimientos previamente obtenidos y, acto seguido, se describirán los principales escenarios y flujos de eventos de dichos casos de uso [28].

4.3.1. Modelo de Casos de Uso

Un caso de uso representa una funcionalidad concreta dada por el sistema como un flujo de eventos. También se puede definir como la representación de una situación o proceso de interacción de un usuario con la aplicación o sistema.

Los casos de uso son tareas con significado, coherentes y con cierta independencia, que los actores realizan de forma cotidiana al utilizar el sistema. En un caso de uso pueden participar uno o varios actores.

En definitiva, los casos de uso explican cómo se realiza una tarea de forma precisa, y constan de los siguientes elementos:

- Nombre único e unívoco del caso de uso.
- Actores participantes.
- Condiciones de entrada.
- Flujo de eventos.
- Condiciones de salida.
- Requerimientos especiales.

Por lo tanto, la primera cuestión es la de determinar cuáles son los actores participantes en cada uno de los casos de uso.

Un actor modela una entidad externa que interacciona con el sistema, es decir, es un tipo de usuario del sistema. Al igual que ocurre con un caso de uso, un actor debe tener un nombre único y adecuado, que puede ir acompañado de una descripción del mismo.

En nuestro sistema contamos con los actores siguientes:

- *Usuario:* Se corresponde con cada uno de los usuarios que entra en el sistema para poder realizar tareas propias de ellos como escuchar una radio personalizada, evaluar canciones, etc.

- *BDMúsica:* Encargado de proporcionar las canciones para poder reproducirlas.

- *LoginGoogle:* Encargado de realizar el login con el sistema de Google.

- *Servidor:* Encargado de proporcionar la información que necesita la aplicación.

Una vez definidos cuáles van a ser los actores del sistema, llega el momento de crear los diferentes casos de uso. Para ello, es importante que cada uno de los requerimientos funcionales anteriormente definidos sean cubiertos por al menos uno de los casos de uso aunque, por otro lado, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que nos encontramos en una fase de refinamiento del sistema en la que pretendemos construir un modelo detallado del mismo.

El primer paso para la realizar el modelo de casos de uso de nuestro sistema es la obtención de los diversos diagramas de casos de uso de nuestro sistema. El primero de ellos es un diagrama frontera, es decir, un diagrama que describe completamente la funcionalidad de un sistema (Véase figura 24).

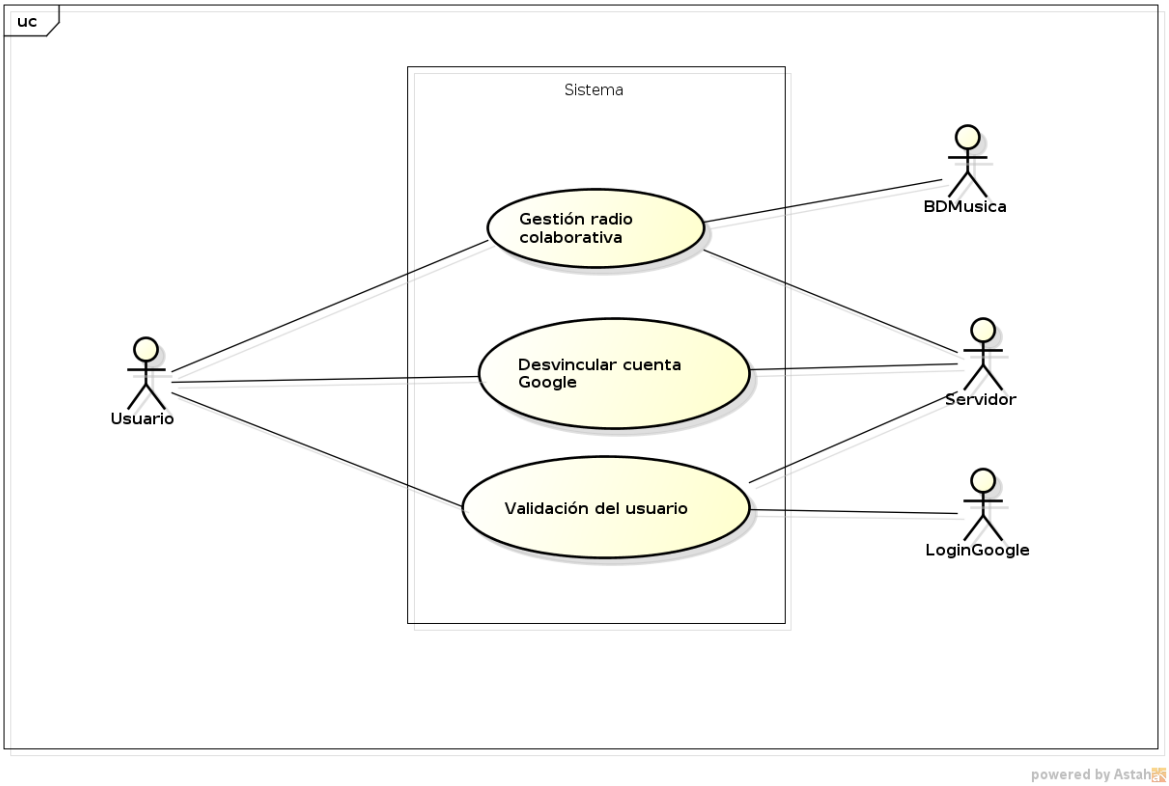

Figura 24: Diagrama frontera

Los casos de uso reflejados en un diagrama frontera pueden ser lo suficientemente precisos o, por el contrario, necesitar ser explicados en mayor detalle. A la hora de detallar un caso de uso se pueden emplear dos tipos de relaciones.

- <<extend>>: Se trata de una relación cuyo sentido es hacia el caso de uso a detallar, que representa comportamientos excepcionales del caso de uso.

- <<include>>: Es una relación cuyo sentido es contrario al de la relación <<extend>>, que representa un comportamiento común del caso de uso.

En el caso de nuestro sistema nos encontramos ante varios casos de uso que deben ser descritos con mayor profundidad.

A continuación, describimos detalladamente cada uno de los casos de uso mostrados en las figuras anteriores, y exponemos los diagramas que se identifican con aquellos casos de uso que necesitan un mayor nivel de detalle en su descripción.

Caso de Uso 1: Validación del usuario

Actores participantes: Usuario, LoginGoogle, Servidor.

Condiciones de entrada: El usuario no está validado en el sistema.

Flujo de eventos:

1. El usuario inicia la aplicación.
2. La aplicación muestra un botón para el inicio de sesión con Google Plus.
3. El usuario pulsa el botón de inicio de sesión con Google Plus.
4. El actor LoginGoogle muestra una pantalla para ceder a nuestra aplicación sus datos personales.
5. El usuario acepta ceder los datos a nuestra aplicación.
6. El actor LoginGoogle cede los datos personales del usuario a la aplicación.
7. La aplicación manda el email del usuario al actor Servidor.
8. El actor Servidor confirma a la aplicación la existencia del usuario y accede al sistema.

Condiciones de salida: El usuario queda identificado en el sistema.

Caso de Uso 2: Gestión radio colaborativa

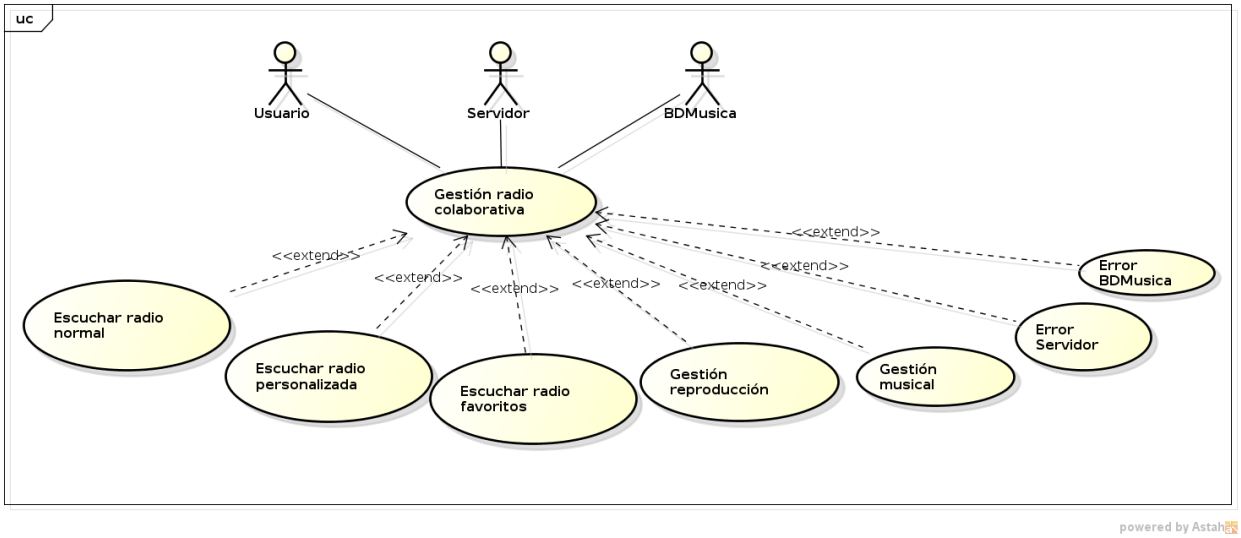

Figura 25: Caso de uso Gestión radio colaborativa

Actores participantes: Usuario, BDMúsica, Servidor.

Condiciones de entrada: El usuario ha accedido al sistema

Flujo de eventos:

1. El usuario accede a la aplicación.
2. La aplicación realiza la petición al actor Servidor.
3. El actor Servidor devuelve a la aplicación una lista con las canciones recomendadas para el usuario.
4. La aplicación realiza la petición al actor BDMúsica.
5. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
6. El sistema comienza la reproducción de la radio obtenida.

7.

a. Si el usuario desea cambiar a radio normal, elige Escuchar radio normal, y se realiza S-1.

b. Si el usuario desea cambiar a radio favoritos, elige Escuchar radio favoritos, y se realiza S-2.

Subflujo de eventos:

- S-1: Escuchar radio normal

Actores participantes: Usuario, BDMúsica, Servidor.

Condiciones de entrada: El usuario ha accedido al sistema

Flujo de eventos:

1. La aplicación realiza la petición al actor Servidor.
2. El actor Servidor devuelve a la aplicación una lista de canciones aleatorias.
3. La aplicación realiza la petición al actor BDMúsica.
4. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
5. El sistema comienza la reproducción de la radio obtenida.
6.
    a. Si el usuario desea cambiar a radio personalizada, elige Escuchar radio personalizada, y se vuelve al punto 2 del flujo principal.

    b. Si el usuario desea cambiar a radio favoritos, elige Escuchar radio favoritos, y se realiza S-2.

- S-2: Escuchar radio favoritos

Actores participantes: Usuario, BDMúsica, Servidor.

Condiciones de entrada: El usuario ha accedido al sistema

Flujo de eventos:

1. La aplicación realiza la petición al actor Servidor.
2. El actor Servidor devuelve a la aplicación una lista con las canciones favoritas del usuario.
3. La aplicación realiza la petición al actor BDMúsica.
4. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
5. El sistema comienza la reproducción de la radio obtenida.
6.
    a. Si el usuario desea cambiar a radio personalizada, elige Escuchar radio personalizada, y se vuelve al punto 2 del flujo principal.
    b. Si el usuario desea cambiar a radio normal, elige Escuchar radio normal, y se realiza S-1.

Condiciones de salida: El usuario ha cerrado la aplicación.

Excepciones:

E-1: Ha habido un error en la comunicación entre el sistema y el actor Servidor. El sistema informa al usuario de dicha situación. El caso de uso se reanuda desde su comienzo.

E-2: Ha habido un error en la comunicación entre el sistema y el actor BDMúsica. El sistema informa al usuario de dicha situación. El caso de uso se

reanuda desde su comienzo.

E-3: Error de conexión, durante la descarga de las canciones en la aplicación. El sistema se recupera una vez establecida la conexión de nuevo. El caso de uso se reanuda.

Caso de Uso 3: Gestión reproducción

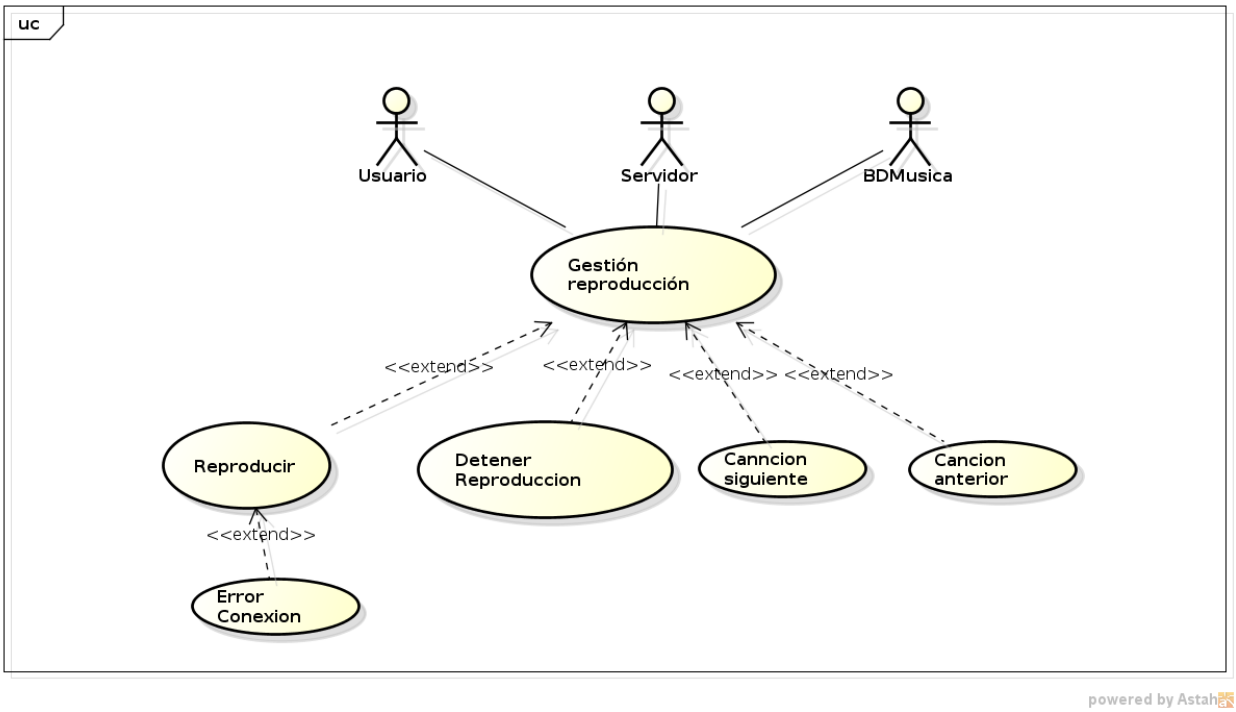

Figura 26: Caso de uso Gestión reproducción

Caso de Uso 4: Gestión musical

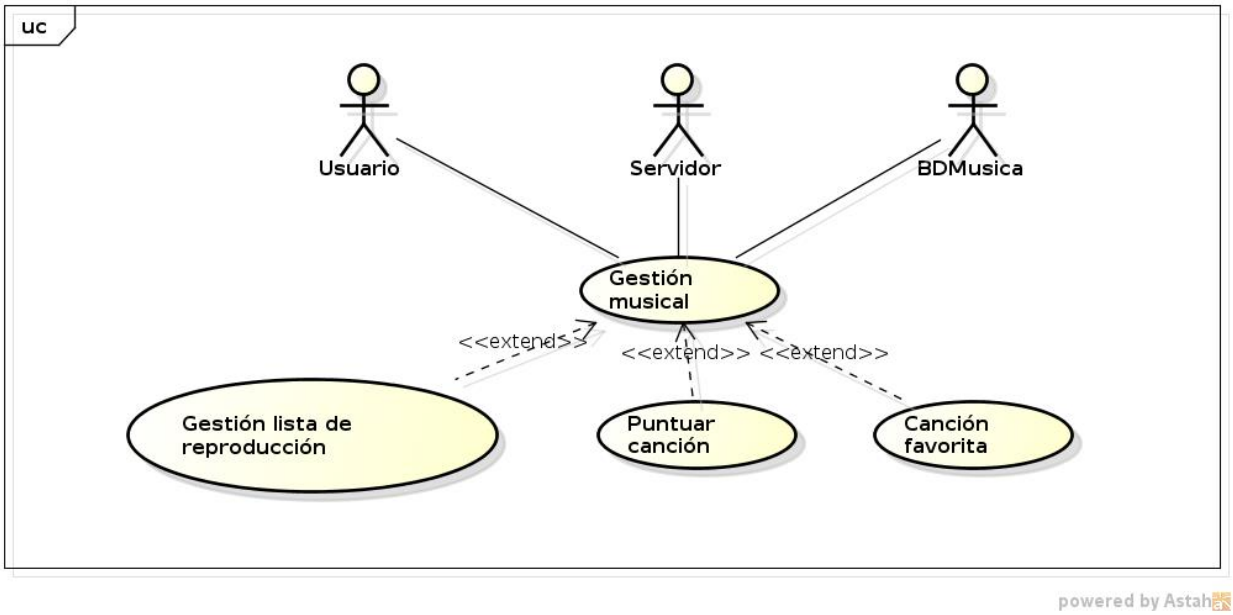

Figura 27: Caso de uso Gestión musical

4.3.2. Escenarios

Un caso de uso es una representación abstracta, una abstracción, de una funcionalidad del sistema a realizar. La representación concreta de un caso de uso se realiza mediante la creación de uno o más escenarios que muestren todas las posibles interacciones entre el sistema y los usuarios.

Los escenarios son historias ficticias que describen posibles interacciones con una interfaz. Son de gran utilidad porque permiten a los diseñadores anticiparse a los problemas. A pesar de ser historias ficticias, deben desarrollarse con el mayor nivel de detalle posible. Así, por ejemplo, los personajes deben tener nombre, motivaciones para usar la interfaz, deben ser situados en un contexto real con las restricciones que ello supone, etc. De esta manera es mucho más sencillo para los diseñadores discutir sobre la interfaz ya que a las personas les cuesta

más trabajo discutir las posteriores decisiones de diseño sobre una situación abstracta.

Esta forma de proceder, además, fuerza a los diseñadores a tener en cuenta el rango de usuarios que va a usar el sistema y el rango de actividades para las que lo van a usar. Los escenarios permiten hacer diferentes combinaciones de usuarios y actividades de forma que se tengan en cuenta todas las posibilidades.

Un escenario está formado por los siguientes elementos:

- Un nombre único y unívoco.
- Una descripción.
- Los actores participantes.
- El flujo de eventos.

Escenario 1: Validación de usuarios en el sistema

Nombre: Validación del usuario José.

Descripción: El usuario José inicia sesión en el sistema como usuario identificado para acceder a las funcionalidades.

Actores participantes: José, LoginGoogle y Servidor

Flujo de eventos:

1. El usuario inicia la aplicación.
2. La aplicación muestra un botón para el inicio de sesión con Google Plus.
3. El usuario José pulsa el botón de inicio de sesión con Google Plus.
4. El actor LoginGoogle muestra una pantalla para ceder a nuestra aplicación sus datos personales.
5. José acepta ceder los datos a nuestra aplicación.
6. El actor LoginGoogle cede los datos personales de José a la aplicación.
7. La aplicación manda el email de José al actor Servidor.
8. El actor Servidor confirma a la aplicación la existencia de José y entra en el sistema.

Escenario 2: Escuchar radio personalizada

Nombre: Escuchar la radio personalizada de Sergio.

Descripción: El usuario registrado en el sistema con el nombre Sergio desea escuchar su radio personalizada, compuesta con canciones de su agrado, además de canciones que el sistema le ha recomendado en base a sus gustos musicales.

Actores participantes: Sergio, BDMúsica y Servidor

Flujo de eventos:

1. Sergio accede a la aplicación.
2. La aplicación realiza la petición al actor Servidor.
3. El actor Servidor devuelve a la aplicación una lista con las canciones recomendadas para Sergio.
4. La aplicación realiza la petición al actor BDMúsica.
5. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
6. El sistema comienza la reproducción de la radio obtenida.

Escenario 3: Realizar puntuaciones sobre la radio

Nombre: Puntuar canción “The Mission”

Descripción: El usuario registrado en el sistema con el nombre Sergio quiere evaluar la canción “The Mission” de Josh Woodward como “Me gusta”.

Actores participantes: Sergio, BDMúsica y Servidor

Flujo de eventos:

1. Sergio accede a la aplicación.
2. La aplicación realiza la petición al actor Servidor.
3. El actor Servidor devuelve a la aplicación una lista con las canciones recomendadas para Sergio.
4. La aplicación realiza la petición al actor BDMúsica.
5. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
6. El sistema comienza la reproducción de la radio obtenida.
7. Sergio pulsa el botón de lista de reproducción y pulsa sobre la canción “The Mission”.
8. El sistema comienza la reproducción de la canción.
9. Sergio decide evaluar la aplicación y pulsa el botón “Me gusta”.
10. Cuando la canción finalice o el usuario decida escuchar otra canción, la aplicación realiza una petición al actor Servidor para añadir la puntuación.
11. El actor Servidor devuelve a la aplicación la confirmación de que se ha introducido la puntuación.

Escenario 4: Indicar favorita una canción

Nombre: Indicar favorita la canción “The Mission”

Descripción: El usuario registrado en el sistema con el nombre Sergio quiere indicar la canción “The Mission” de Josh Woodward como “Favorita”.

Actores participantes: Sergio, BDMúsica y Servidor

Flujo de eventos:

1. Sergio accede a la aplicación.
2. La aplicación realiza la petición al actor Servidor.
3. El actor Servidor devuelve a la aplicación una lista con las canciones recomendadas para Sergio.
4. La aplicación realiza la petición al actor BDMúsica.
5. El actor BDMúsica devuelve a la aplicación la información de las canciones a reproducir.
6. El sistema comienza la reproducción de la radio obtenida.
7. Sergio pulsa el botón de lista de reproducción y pulsa sobre la canción “The Mission”.
8. El sistema comienza la reproducción de la canción.
9. Sergio decide evaluar la aplicación y pulsa el botón “Favorita”.
10. Cuando la canción finalice o el usuario decida escuchar otra canción, la aplicación realiza una petición al actor Servidor para añadir que la canción a favoritas.
11. El actor Servidor devuelve a la aplicación la confirmación de que se ha añadida la canción como favorita.

4.4. Diseño del sistema

No cabe duda de que realizar de forma adecuada cada una de las actividades que conlleva la Ingeniería del Software es imprescindible para la realización de un proyecto software de calidad. Por ello, no se puede afirmar que ninguna de estas actividades sea más importante que el resto. Sin embargo, sí podemos decir que la actividad de diseño es la más delicada y la más laboriosa de realizar.

- Es delicada porque, de no llevarse a cabo correctamente, se hace imposible el codificar de manera correcta en la actividad de implementación el modelo obtenido anteriormente en la fase de análisis del sistema. Esto puede repercutir en hacer inútil todo el esfuerzo invertido en las primeras actividades de la Ingeniería del Software.

- Es laboriosa porque las estrategias a seguir para que la traducción entre modelo y código se lleve a cabo correctamente son muy diversas y bastante complejas.

Se puede decir, pues, que el diseño del sistema es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos en la actividad de implementación [30].

Sin embargo, el sistema no se suele diseñar de una sola pasada, sino que hemos de diferenciar entre el diseño y estructura de los datos que se van a manejar, y el diseño de la interfaz entre la aplicación y el usuario. Estas dos fases del diseño no se realizan de forma consecutiva una tras otra, sino que lo normal es realizarlas de forma paralela y finalizarlas simultáneamente.

4.4.1. Diseño de los datos

El objetivo de esta fase del diseño software es determinar la estructura que poseen cada uno de los elementos de información del sistema, es decir, la estructura de los datos sobre los que se va a trabajar. Estos elementos son:

- Las **canciones**, de las que conocemos el id de la canción en Jamendo, el nombre del artista o intérprete de la canción, el género musical al que pertenece la canción, las veces que ha sido valorada y si la canción está disponible.

- Los **usuarios**, de los que conocemos su identificador, nombre, contraseña, dirección de e-mail y clave de la API.

- Las **valoraciones**, de las que sabemos el usuario que realiza la valoración, la canción que ha sido puntuada, el valor de dicha valoración, cuando se realizó la valoración y si la canción es favorita es o no.

Una vez determinados cuales son los elementos de información del sistema, se deben obtener sus representaciones en forma de tablas de una base de datos [31]. Para ello, se debe realizar en primer lugar un diseño conceptual de la base de datos para, posteriormente, obtener las tablas requeridas. Para realizar este diseño conceptual se utilizará el modelo Entidad-Relación (E-R).

Modelo Entidad-Relación

El modelo Entidad-Relación (también conocido por sus iniciales, E-R) es una técnica de modelado de datos que emplea diagramas entidad-relación. No es la única técnica de modelado pero sí es, con diferencia, la más extendida y utilizada.

Un diagrama entidad-relación está compuesto por tres tipos de elementos principales:

- *Entidades*: Son objetos (cosas, conceptos o personas) sobre los que se tiene información (Véase figura 28). Se representan mediante rectángulos etiquetados con un nombre en su interior. Llamamos instancia a cualquier ejemplar concreto de una entidad.

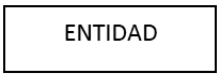

Figura 28: Entidad en modelo Entidad-Relación

- *Relaciones*: Interdependencias entre una o más entidades. Se representan mediante rombos etiquetados en su interior con un verbo. (Véase figura 29) Si la relación es entre una entidad consigo misma se denomina reflexiva, si es entre dos entidades se denomina binaria, si es entre tres ternaria y múltiple si es entre cuatro o más entidades (muy poco común).

Figura 29: Relación en modelo Entidad-Relación

- *Atributos*: Describen características propias de una entidad o relación (Véase figura 30). Se representan mediante elipses etiquetadas con un nombre en su interior.

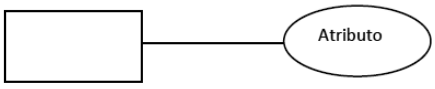

Figura 30: Atributo en modelo Entidad-Relación

En los diagramas entidad-relación también hay que tener en cuenta otros aspectos como los que indicamos a continuación:

- *Entidades débiles*: son aquellas entidades que no se pueden identificar unívocamente sólo con sus atributos, sino que necesitan estar relacionadas con otras entidades para existir. Se representan con dos rectángulos concéntricos de distinto tamaño con un nombre en el interior del más pequeño.

- *Cardinalidad de las relaciones*: Existen tres tipos de cardinalidades de una relación según el número de instancias de cada entidad que involucren.

- *Uno a uno*: Una instancia de la entidad A se relaciona solamente con una instancia de la entidad B. Se representan como 1:1.

- *Uno a muchos*: Cada instancia de la entidad A se relaciona con varias instancias de la entidad B. Se representan como 1:*.

- *Muchos a muchos*: Cualquier instancia de la entidad A se relaciona con cualquier instancia de la entidad B. Se representan como *:*.

- *Claves*: Cada entidad de un diagrama entidad-relación debe tener una clave, que debe estar formada por uno o varios de sus atributos. Dicha clave es la que distingue unívocamente una instancia de la citada entidad del resto.

Una vez conocidos los elementos que forman parte de un diagrama entidad-relación podemos empezar a desarrollar el modelo entidad-relación. Los pasos a seguir son los siguientes:

1. Convertir el enunciado del problema (o, como es nuestro caso, los elementos del sistema software) en un Esquema Conceptual del mismo.

2. Convertir este Esquema Conceptual (o EC) en uno más refinado, conocido como Esquema Conceptual Modificado (ECM).

3. Obtener las tablas de la base de datos a partir del Esquema Conceptual Modificado.

Normalización en el modelo Entidad-Relación

La normalización es un proceso que consiste en imponer a las tablas ciertas restricciones mediante una serie de transformaciones consecutivas. Con ello nos aseguramos de que las tablas contienen los atributos necesarios y suficientes para describir la realidad de la entidad que representan, separando aquellos que pueden contener información cuya relevancia permite la creación de otra nueva tabla.

Para asegurar la normalización, Codd estableció tres formas normales, las cuales hacen que toda base de datos que las cumple se considere normalizada.

Estas formas normales son:

- *Primera forma Normal (FN1)*: Una tabla está en FN1 si todos los atributos no clave dependen funcionalmente de la clave, o lo que es lo mismo, no existen grupos repetitivos para un valor de clave.

- *Segunda forma Normal (FN2)*: Una tabla está en FN2 si está en FN1 y además todos los atributos que no pertenecen a la clave dependen funcionalmente de ella de forma completa. De esta definición se puede concluir que una tabla en FN1 y cuya clave está compuesta por un único atributo ya está en FN2.

- *Tercera forma Normal (FN3)*: Una tabla está en FN3 si está en FN2 y además no existen atributos no clave que dependan transitivamente de la clave.

Dada la no muy elevada complejidad de las tablas que obtendremos para el modelo de datos de este proyecto, no realizaremos el proceso de normalización. Por tanto, las entidades obtenidas en el Esquema Conceptual Modificado constituirán las tablas de nuestra base de datos.

4.4.1.1. Esquema conceptual

Necesitamos convertir nuestros elementos de información en entidades o relaciones. En nuestro caso, Canciones y Usuarios pasarán a convertirse en entidades de nuestro esquema conceptual, y Valoraciones se convierte en relaciones que unen la entidad Canciones con Usuarios. Así, nuestro esquema conceptual se puede ver en la figura 31:

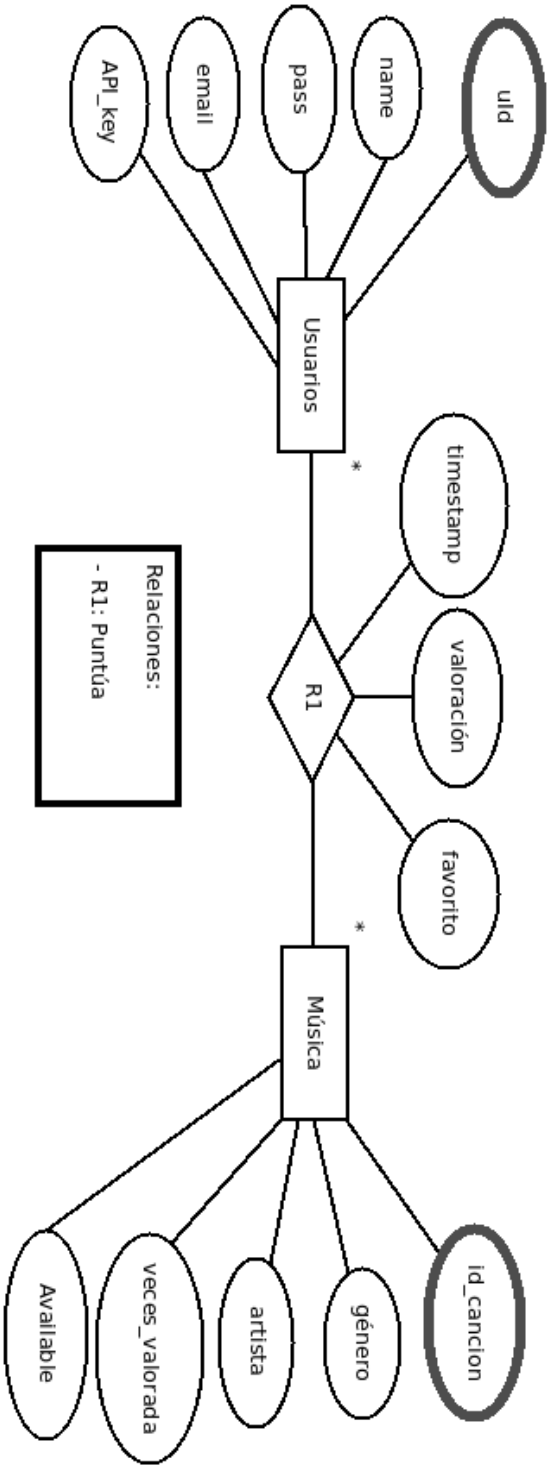

Figura 31: Esquema Conceptual de la aplicación

4.4.1.2. Esquema Conceptual Modificado

Para la obtención del Esquema Conceptual Modificado a partir del Esquema Conceptual se deben hacer los cambios que enunciamos a continuación:

- Eliminar todas las entidades débiles.
- Eliminar las relaciones de muchos a muchos.
- Eliminar las relaciones con atributos existentes en el Esquema Conceptual.

En nuestro caso, no es necesaria la eliminación de entidades débiles, ya que no las hay. Realizando las modificaciones oportunas, obtendremos una nueva entidad: Valoraciones derivada de la relación muchos a muchos existente en el EC. Nuestro Esquema Conceptual Modificado (ECM) quedará como muestra la figura 32.

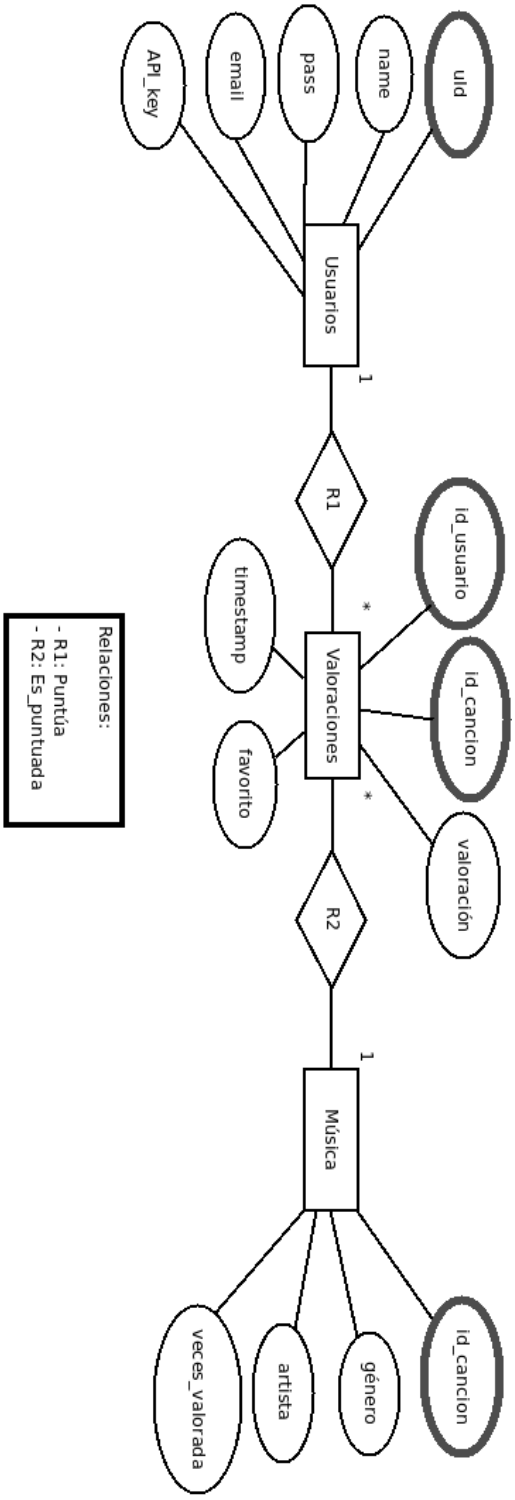

Figura 32: Esquema Conceptual Modificado de la aplicación

4.4.1.3. Tablas de la aplicación

Basándonos en el ECM obtenido previamente, podemos determinar un total de 3 tablas en la base de datos, teniendo en cuenta que:

- Cada entidad del ECM se transforma en una tabla.
- Los atributos de una entidad se convierten en los campos de las tablas respectivas.

Por tanto, según el ECM, obtendremos las siguientes tablas:

- La entidad Música da lugar a la tabla MÚSICA.
- La entidad Usuarios da lugar a la tabla USERS.
- La entidad Valoraciones da lugar a la tabla VALORACIONES.

MÚSICA

Tabla de 12111 filas con los siguientes atributos cada una:

- _ID: Entero. Llave primaria. Identificador numérico y univoco de la canción.
- ARTIST: Cadena de texto. Autor de la canción.
- VECES_VALORADA: Entero. Veces que se ha valorado la canción.
- AVAILABLE: Entero. Indica si en este momento está la canción disponible.
- GENERO: Cadena de texto. Género al que la canción pertenece. Puede tomar los siguientes valores:
    - Metal, disco, world, poprock, jazz, indie, classical, reggae, techno, industrial, pop, hiphop, lounge, country, latin, rock, electronic, dance.

USERS

Se trata de una tabla que contiene tantas filas como usuarios hay registrados en el servicio y tienen los siguientes atributos:

- UID: Entero. Llave primaria. Identificador numérico y univoco del usuario.
- NAME: Cadena de 60 caracteres. Nombre del usuario.
- PASS: Cadena de 32 caracteres. Contraseña del usuario codificada mediante un algoritmo de cifrado.
- MAIL: Cadena de 64 caracteres. Dirección de correo-e del usuario.
- API_KEY: Cadena de 100 caracteres. Clave de acceso a la API.

VALORACIONES

Tabla que contiene los siguientes campos:

- ID_USUARIO: Entero. Llave primaria. Llave externa. Identificador del usuario.
- ID_CANCION: Entero. Llave primaria. Llave externa. Identificador de la canción.
- VALORACION: Entero de longitud 1. Puntuación (1, 2, 3, 4 ó 5) del usuario sobre la canción.
- TIMESTAMP: Fecha. Fecha en la que se realizó la valoración.
- FAV: Entero. Indica si la canción es favorita para el usuario.

Sin embargo existe una excepción. Dado que vamos a utilizar la librería de Recomendación [32] es necesario añadir varias tablas más para obtener las recomendaciones. En concreto son:

- Tabla itemFeatures (características de los ítems).
- Tabla item_profiles (predicciones de ítems).
- Tabla userFeatures (características de usuarios).
- Tabla recommendations_table (recomendaciones para usuarios).

4.4.2. Diseño de la Interfaz

En esta etapa del diseño del sistema software se define cual va a ser la apariencia visual de la aplicación, es decir, se define la interfaz visual entre el usuario y la aplicación. Sin duda, llevar a cabo un buen diseño de la interfaz resulta decisivo debido a que esta debe ser atractiva para el usuario de la aplicación pero al mismo tiempo debe resultar fácil de entender y de trabajar sobre ella [29].

En este apartado definiremos describiremos y analizaremos las metáforas y revisaremos el diseño de pantallas y los caminos de navegación (storyboards) empleados para el desarrollo de la interfaz de nuestro prototipo de aplicación software.

4.4.2.1. Metáforas

Una metáfora es el empleo de un objeto con un significado o dentro de un contexto diferente al habitual. En el diseño de una interfaz gráfica, la utilización de metáforas resulta muy útil ya que permiten al usuario, mediante la comparación con otro objeto o concepto, comprender de forma más intuitiva las diversas tareas que la interfaz permite desarrollar. Las metáforas juegan un papel muy importante en el éxito o fracaso de una interfaz, por tanto deben diseñarse con mucho cuidado.

Al igual que pasa en el ámbito de la literatura, para que una metáfora cumpla con su cometido, el desarrollador de la aplicación y el usuario final de la misma deben una tener una base social y cultural similar. Es muy posible que el uso de un icono de manera metafórica sea entendido de una manera por el usuario occidental y de otra bien distinta por un usuario oriental. Pensemos, por ejemplo, en el icono de una mano con el pulgar levantado: para un usuario occidental es muy probable que signifique que la tarea se ha realizado con éxito. Sin embargo, un usuario natural de Turquía, por ejemplo, consideraría este gesto como un acto ofensivo.

Es necesario, pues, tener en cuenta la base cultural de los usuarios para diseñar una buena metáfora. En definitiva, hay que intentar que las metáforas empleadas sean lo más universales posibles, a fin de que sean comprendidas a la perfección por la mayor parte del público potencial.

Pero las metáforas no solo dependen del tipo de aplicación (escritorio, Web, app) sino también del ámbito de la misma. Por ejemplo, el carrito de la compra es una metáfora conocida por todos, pero si nuestra aplicación no va a vender producto alguno al usuario no conviene utilizarla ya que puede dar lugar a confusión.

En nuestra aplicación hemos utilizado las siguientes metáforas:

Me gusta

Esta metáfora aparece en la parte inferior izquierda de la pantalla y tiene dos estados, marcado o sin marcar. Si no está marcado significa que todavía no me gusta y si está marcado significa que la canción me gusta (Véase figura 33).

Figura 33: Icono para la metáfora "Me gusta"

No me gusta

Esta metáfora aparece en la parte inferior derecha de la pantalla y tiene dos estados, marcado o sin marcar. Si no está marcado significa que todavía me gusta y si está marcado significa que la canción no me gusta (Véase figura 34).

Figura 34: Icono para la metáfora "No me gusta"

Canción favorita

Esta metáfora aparece en la parte superior izquierda de la pantalla y tiene dos estados, marcado o sin marcar. Si no está marcado significa que todavía es una canción favorita y si está marcado significa que la canción es una canción favorita (Véase figura 35).

Figura 35: Icono para la metáfora "Canción favorita"

Comenzar música

Esta metáfora aparece en la parte inferior centrado en la pantalla y significa comenzar a reproducir la música (Véase figura 36).

Figura 36: Icono para la metáfora "Comenzar música"

Parar música

Esta metáfora aparece en la parte inferior centrado en la pantalla y significa parar la reproducción de música (Véase figura 37).

Figura 37: Icono para la metáfora "Parar música"

Siguiente canción

Esta metáfora aparece en la parte inferior derecha de la pantalla y significa reproducir la canción siguiente de la lista de reproducción (Véase figura 38).

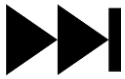

Figura 38: Icono para la metáfora "Siguiente canción"

Anterior canción

Esta metáfora aparece en la parte inferior izquierda de la pantalla y significa reproducir la canción anterior de la lista de reproducción (Véase figura 39).

Figura 39: Icono para la metáfora "Anterior canción"

Lista de reproducción

Esta metáfora aparece en la parte superior derecha de la pantalla y significa ver una lista donde podamos ver todas las canciones, y de ahí la elección de un símbolo de lista (Véase figura 40).

Figura 40: Icono para la metáfora "Lista de reproducción"

Flujo de reproducción

Esta metáfora aparece en la parte inferior centrado en la pantalla y significa el progreso de la canción. A parte a izquierda y a derecha se indica el progreso en segundos (Véase figura 41).

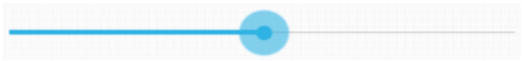

Figura 41: Icono para la metáfora "Flujo de reproducción"

Modo radio normal

Esta metáfora aparece en el menú lateral y en la barra de acciones superior y significa que estás reproduciendo la radio normal, que implica canciones aleatorias de todo el repertorio, y de ahí viene la elección de un símbolo de aleatoriedad (Véase figura 42).

Figura 42: Icono para la metáfora "Modo radio normal"

Modo radio personalizada

Esta metáfora aparece en el menú lateral y en la barra de acciones superior y significa que estás reproduciendo la radio personalizada, que implica canciones personalizadas para el usuario, y de ahí viene la elección de un símbolo de flujo de sonido (Véase figura 43).

Figura 43: Icono para la metáfora "Modo radio personalizada"

Modo radio favoritos

Esta metáfora aparece en el menú lateral y en la barra de acciones superior y significa que estás reproduciendo la radio favoritos, que implica canciones favoritas del usuario, y de ahí viene la elección de un símbolo de valor (Véase figura 44).

Figura 44: Icono para la metáfora "Modo radio favoritos"

Desvincular cuenta Google

Esta metáfora aparece en el menú lateral y significa que no queremos seguir dando los datos personales de la cuenta de Google Plus a la aplicación, y de ahí viene la elección de un símbolo de salida (Véase figura 45).

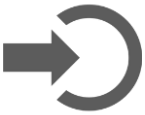

Figura 45: Icono para la metáfora "Desvincular cuenta Google"

4.4.2.2. Prototipos de la aplicación

En este apartado, vamos a definir la estructura de nuestra interfaz con el usuario, ya que la parte del servidor no tendrá interfaz ninguna pues se ejecutará automáticamente sin intervención manual.

Mediante el diseño de prototipos se pretende tener un esbozo de lo que será la interfaz de usuario de nuestra aplicación. Dichos prototipos no expresan el diseño final, simplemente dan una idea de lo que será nuestro sistema para el usuario final. Estos prototipos serán, por tanto, susceptibles de cambio durante el proceso de implementación.

En nuestro caso, optamos por el diseño de pantallas como técnica de definición de prototipos de la interfaz. La mejor herramienta para ello es, simplemente, usar dibujos hechos a mano con lápiz y papel. La razón es que estos diseños de pantalla son un vehículo para poder analizar la calidad de nuestro diseño, por ello no es adecuado perder tiempo ni dinero realizando estos diseño previos con alguna herramienta especializada.

Pantalla Login

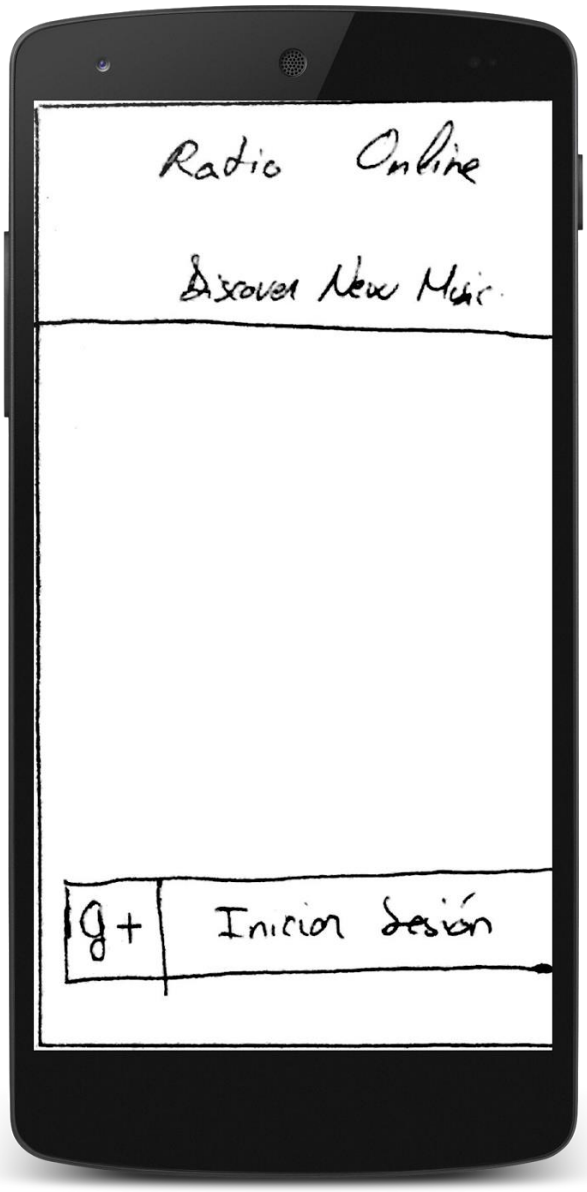

Figura 46: Pantalla "Login"

Pantalla Reproductor

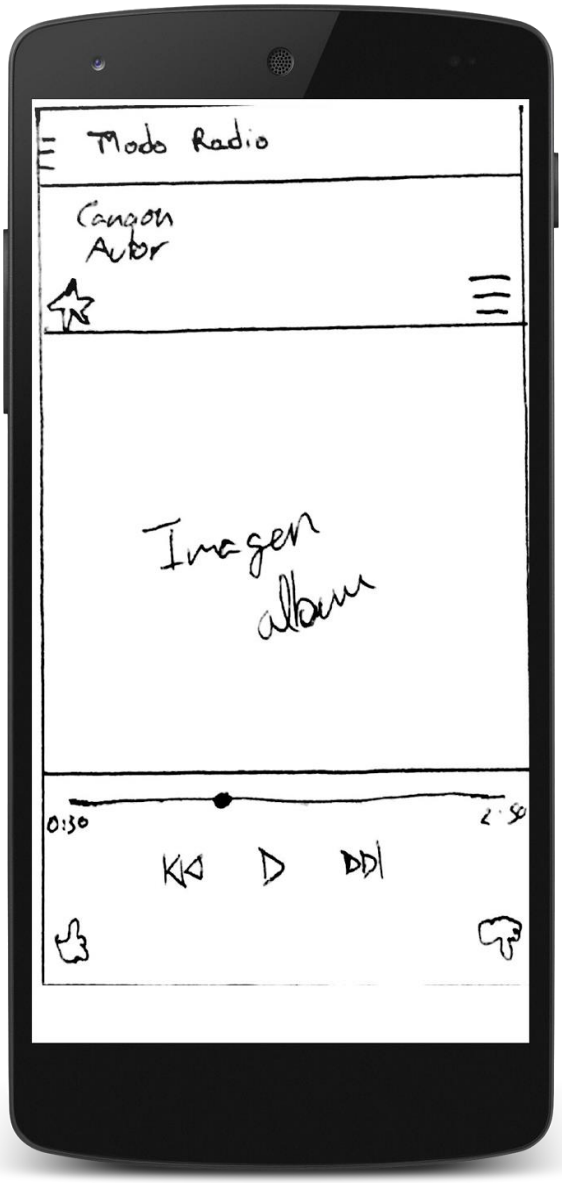

Figura 47: Pantalla "Reproductor"

Drawer Pantalla Reproductor

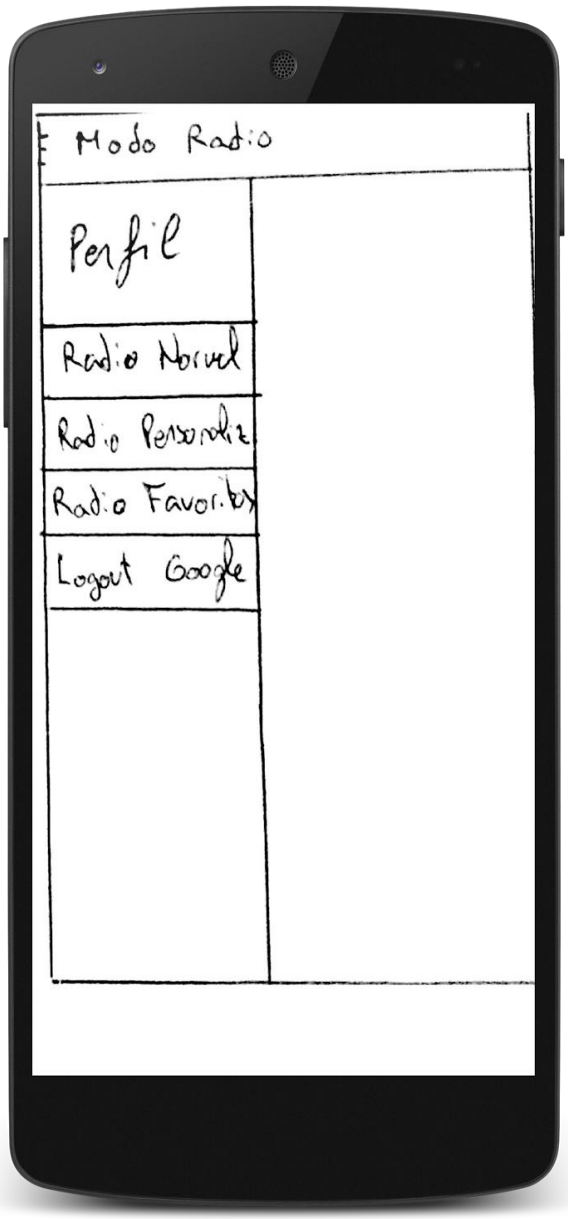

Figura 48: Pantalla "Drawer Pantalla Reproductor"

4.4.2.3. Caminos de navegación

Hasta este momento tenemos un diseño visual de la interfaz estática, es decir, cada pantalla diseñada individualmente, pero no tenemos una idea de si en el conjunto de la interacción, la acción va a transcurrir de forma fluida y comprensible para el usuario. Para ello vamos a diseñar la interfaz en movimiento y comprobar que es usable.

Para estudiar los caminos de navegación se empleará una herramienta llamada storyboard, que consiste en mostrar, a modo de secuencia, las diferentes pantallas por las que se va pasando al realizar el usuario una determinada acción sobre la aplicación.

El procedimiento es el siguiente: se sitúan capturas de las pantallas de la interfaz unidas mediante flechas para indicar el camino que sigue la interacción. La posición de origen de las flechas debe ayudar a entender cuál es el elemento que ha desencadenado el paso de una pantalla a otra. Los storyboards también están muy ligados a los escenarios anteriormente vistos.

El storyboard sirve de prototipo para ser evaluado por el usuario y poder introducir correcciones en fases tempranas, ya que cuanto más tiempo se tarde en validar una interfaz, más coste de tiempo y trabajo acarreará.

A continuación, mostramos los storyboards para las acciones más importantes que se pueden llevar a cabo en nuestro sistema.

- Storyboard Validación de usuario / cierre de sesión.
- Storyboard Puntuar canción.
- Storyboard Marcar canción favorita.
- Storyboard Escuchar radio normal.
- Storyboard Escuchar radio favoritos.

Storyboard Validación de usuario / cierre de sesión

Para la validación del usuario (Véase figura 49):

1. El usuario pulsa el botón de “Iniciar sesión”.
2. Google muestra un mensaje para iniciar sesión.
3. Inicia la pantalla de reproductor de música.

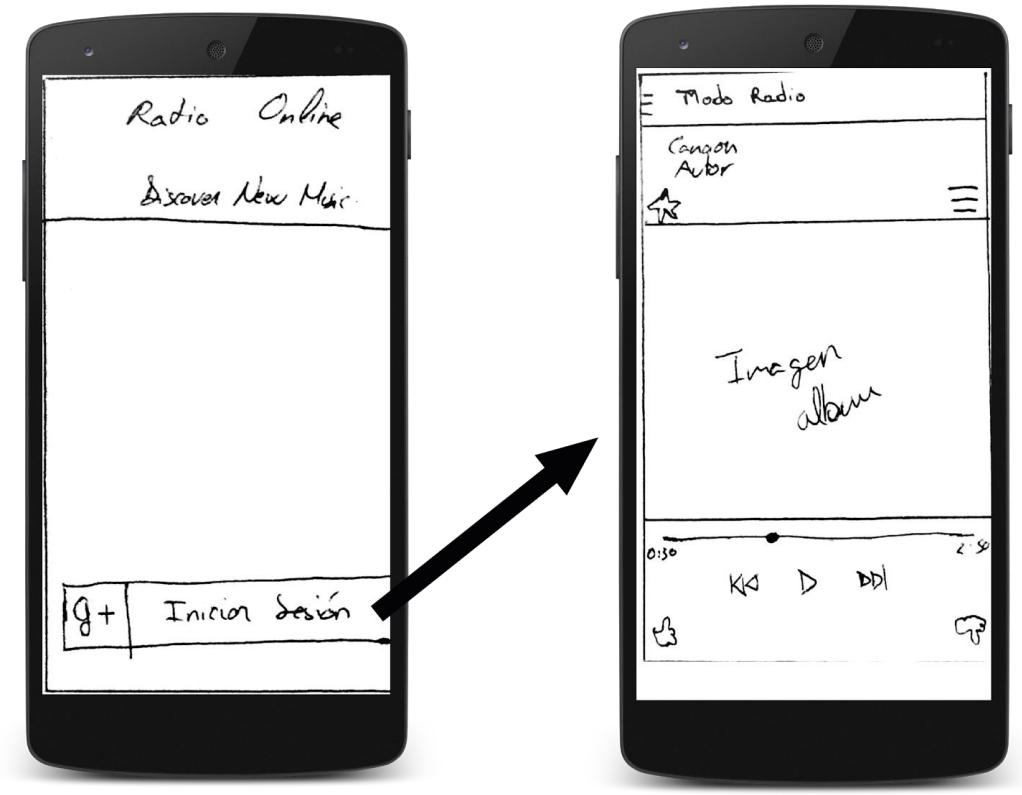

Figura 49: StoryBoard para “Validación de usuario”

Para el cierre de sesión (Véase figura 50):

1. El usuario abre el menú lateral.
2. El usuario pulsa “Logout de Google”
3. La aplicación vuelve a la pantalla de inicio de sesión.

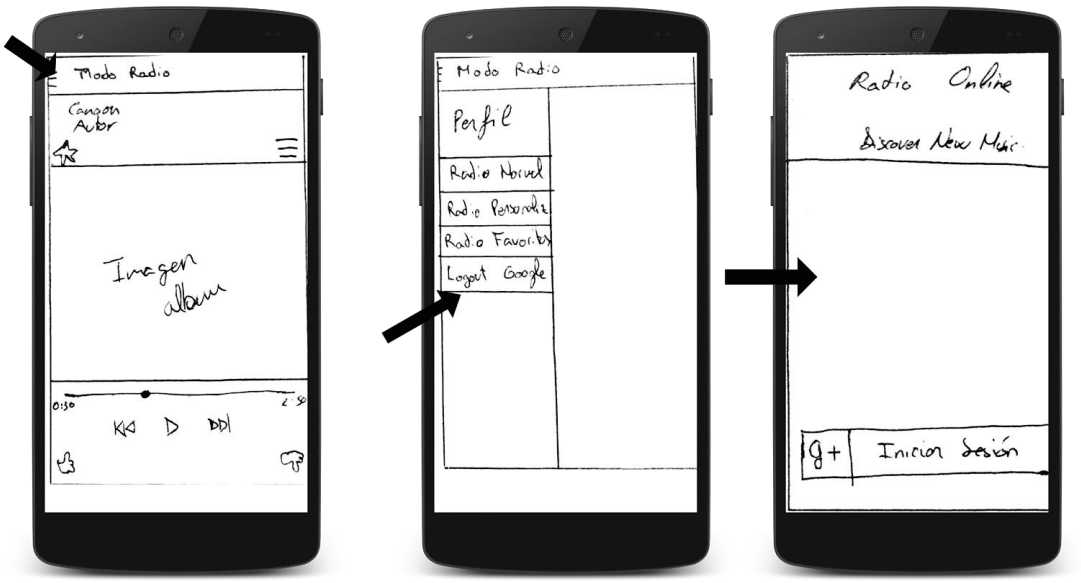

Figura 50: StoryBoard para “Cierre de sesión”

Storyboard Puntuar canción

1. El usuario pulsara el icono “me gusta” (abajo a la izquierda) o el icono “no me gusta” (abajo a la derecha) en la figura 51.

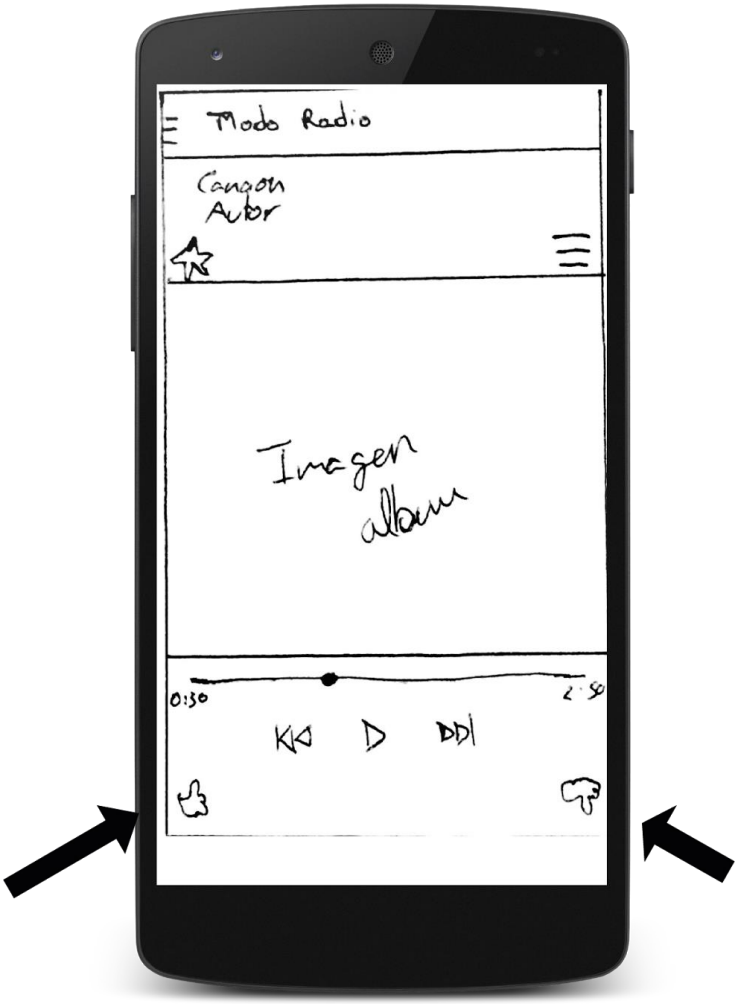

Figura 51: StoryBoard para “Puntuar canción”

Storyboard Marcar canción favorita

1. El usuario pulsa el icono de favorito (arriba a la izquierda) en la figura 52.

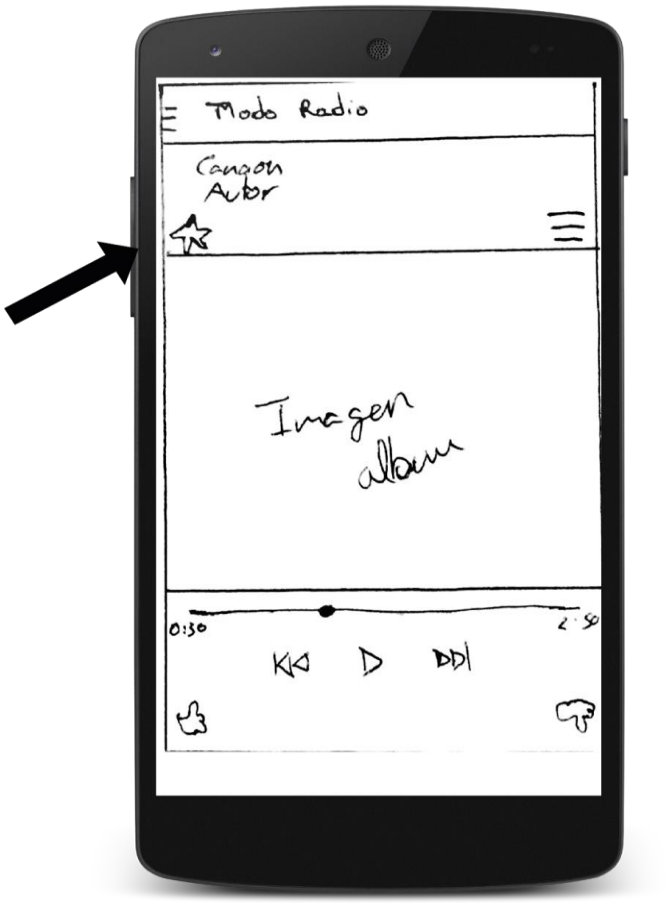

Figura 52: StoryBoard para "Marcar canción favorita"

Storyboard Escuchar radio normal (Véase figura 53)

1. El usuario abre el menú lateral.
2. El usuario pulsa “Radio normal”.
3. La aplicación vuelve a cargar la pantalla “Reproductor”.

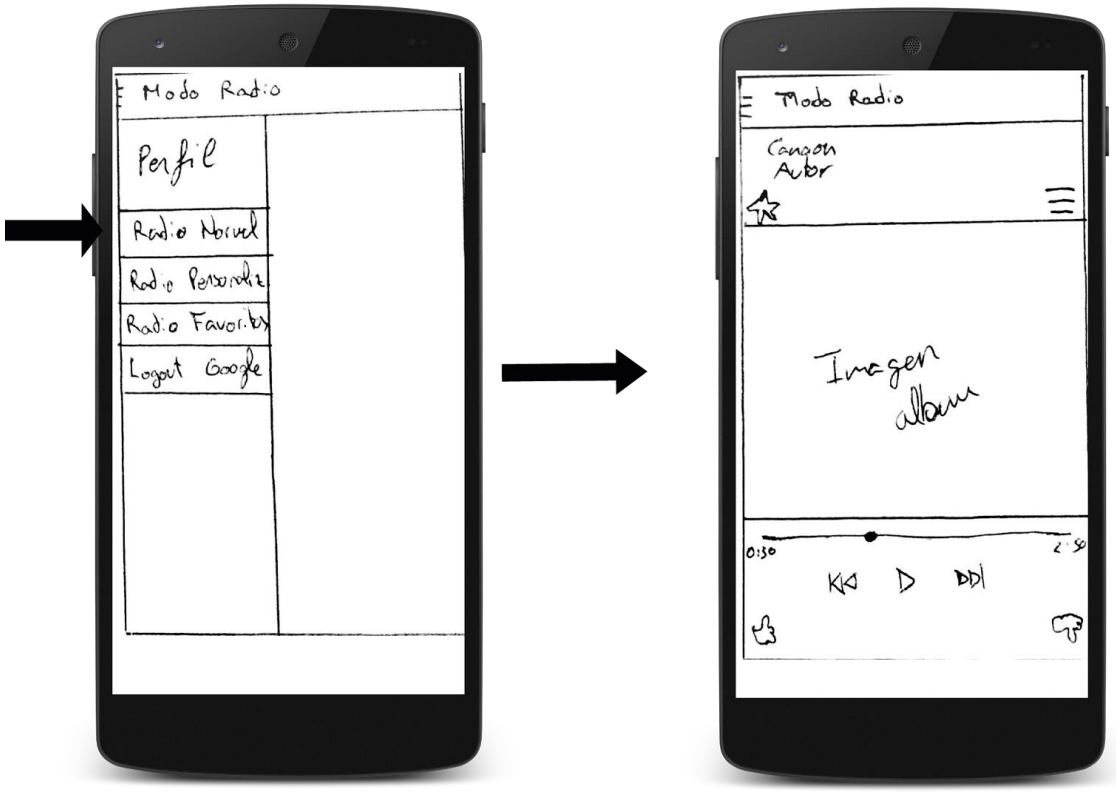

Figura 53: StoryBoard para “Escuchar radio normal”

Storyboard Escuchar radio favoritos (Véase figura 54)

1. El usuario abre el menú lateral.
2. El usuario pulsa “Radio favoritos”.
3. La aplicación vuelve a cargar la pantalla “Reproductor”.

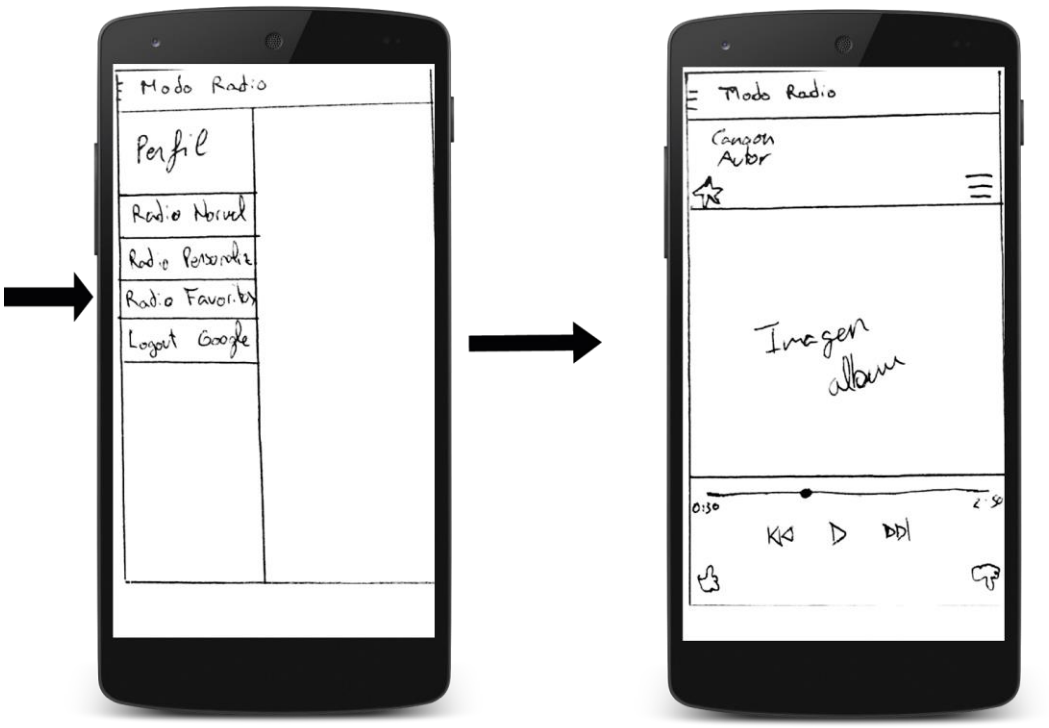

Figura 54: StoryBoard para “Escuchar radio favoritos”

4.5. Implementación

La implementación es la actividad final de la Ingeniería del Software, aquella en la que el modelo obtenido en las actividades anteriores se debe transformar en código fuente. Para ello, se debe ser cuidadoso en la elección del lenguaje de programación empleado para la codificación y de la herramienta utilizada para generarla.

En el caso de nuestra aplicación, desarrollaremos un Sistema de Recomendación con una arquitectura cliente/servidor (Véase figura 55). El funcionamiento de este tipo de arquitectura es sencillo: la aplicación se encuentra en un servidor central, al que los usuarios acceden por medio un software cliente, en nuestro caso una aplicación Android, a través de un servicio REST. Una vez que ha accedido a la aplicación, el usuario realiza peticiones al servicio REST y este emite la respuesta concreta, adecuada y comprensible para el cliente.

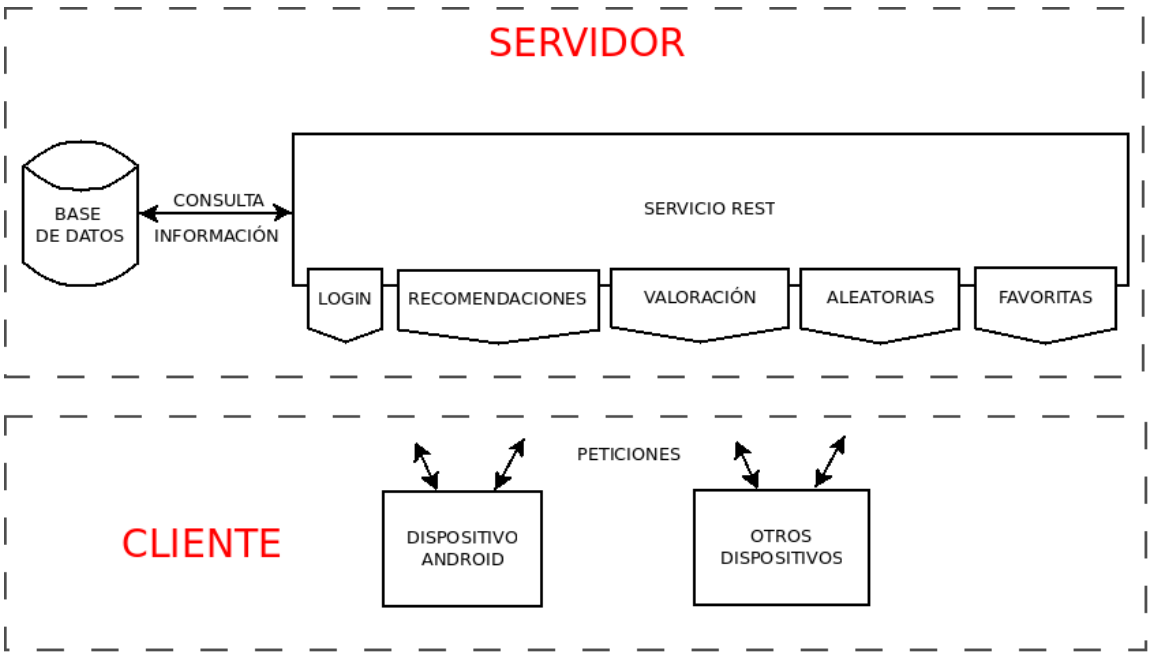

Figura 55: Arquitectura cliente-servidor del proyecto

Una arquitectura cliente/servidor elimina del usuario final de la aplicación, por tanto, la tarea de tener que instalar dicha aplicación en su máquina, consiguiendo además que cada usuario acceda únicamente a la información que le corresponde. Además, dado su diseño modular, esta arquitectura es fácilmente escalable y ampliable tanto en nuevos clientes como en servidores añadidos.

4.5.1. Lenguajes de programación utilizados

Parte Cliente

- *Java*: Lenguaje orientado a objetos cuyo potencial reside en la compilación a código intermedio o bytecode de las aplicaciones desarrolladas en este lenguaje. Esto permite que una aplicación pueda ejecutarse en múltiples plataformas y de manera independiente al hardware a través de una máquina virtual java.

  Java es el lenguaje utilizado para desarrollar la lógica de las aplicaciones para Android. Como hemos explicado en el apartado 3.1.3, utiliza la máquina virtual Dalvik para la ejecución en sistemas operativos Android.

- *XML*: Lenguaje de marcas para documentos que permite almacenar información de forma estructurada. Android lo utiliza como estándar para el desarrollo de las interfaces de usuario.

Parte Servidor

La parte servidor consiste en el desarrollo de un Servicio REST. El concepto de Servicio REST es estudiado en el apartado 3.2. La tabla de recursos que se ha diseñado en este proyecto la podemos ver en la tabla 5:

URL	Método	Parámetros	Descripción
/login	POST	email	Identifica el email del usuario dentro del sistema. Si no existe crea una cuenta de usuario.
/recommendations	GET	API Key	Devuelve todas las recomendaciones para el usuario al que pertenece el API Key.
/ratings	POST	API Key, rating, idCancion	Inserta la valoración al usuario que pertenezca el API Key.
/random	GET	API Key	Devuelve una lista de canciones aleatorias para el usuario al que pertenezca el API Key.
/favourites	GET	API Key	Devuelve una lista de canciones favoritas del usuario al que pertenezca el API Key.

Tabla 5. Recursos API REST del proyecto

La implementación de esta parte se ha realizado con el lenguaje PHP y utilizando un framework PHP llamado Slim Framework ya que facilita la implementación de una API REST.

PHP, acrónimo recursivo de Hypertext Preprocessor, es un lenguaje de programación interpretado, que se ejecuta del lado del servidor y genera contenido dinámico a petición del cliente.

Se trata de un lenguaje con una serie de importantes ventajas frente a otros lenguajes que realizan funciones parecidas, como son las siguientes:

- Es un lenguaje multiplataforma.
- Capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, destaca su conectividad con MySQL (de gran utilidad en la implementación de este proyecto).
- Lectura y manipulación de datos desde diversas fuentes, incluyendo datos que pueden introducir los usuarios desde formularios HTML.
- Capacidad de expandir su potenciales gracias al enorme abanico de módulos disponibles (llamados extensiones o ext's).
- Posee una amplia documentación en su página oficial, entre la cual cabe destacar la explicación de todas las funciones del sistema, mediante ejemplos y archivos de ayuda.
- No precisa de la declaración de variables.
- Es libre, por lo que se presenta como una alternativa fácilmente accesible para todos.
- Permite las técnicas de programación Orientada a Objetos.
- Permite la creación de formularios para la Web.

- Amplia biblioteca nativa de funciones incluida. No requiere definición de tipos de variables ni un manejo avanzado a bajo nivel.

Por otro lado tenemos Slim Framework. Slim Framework es un micro framework basado en PHP que ayuda a crear muy rápido y de una manera muy simple potentes aplicaciones web y las APIs.

Las características de Slim son:

- Potente enrutador.
- Métodos HTTP comunes y personalizados.
- Parámetros en la ruta, con comodines y condiciones.
- Redireccionamiento de rutas
- Prestación para plantillas con vistas personalizadas
- Mensajes flash
- Cookies seguras con cifrado AES-256
- Cacheo HTTP
- Escritura de logs con escritores de log personalizados
- Captura de errores y depuración
- Fácil configuración

4.5.2. Herramientas de desarrollo

Una vez descritos los lenguajes de programación empleados para este proyecto, pasamos a especificar las herramientas empleadas para el desarrollo de la aplicación a través de dichos lenguajes.

En este proyecto se han implementado dos partes, una cliente y otra servidora, y cada una de ellas, por necesidades del lenguaje, ha necesitado de distintas herramientas de desarrollo.

La parte cliente, que consiste en una aplicación Android y que su lenguaje de programación es Java, ha sido implementada en el entorno de desarrollo Eclipse.

Eclipse es un Entorno Integrado de Desarrollo, del inglés Integrated Development Environment (IDE), para todo tipo de aplicaciones libres, inicialmente desarrollado por IBM, y actualmente gestionado por la Fundación Eclipse (Véase figura 56).

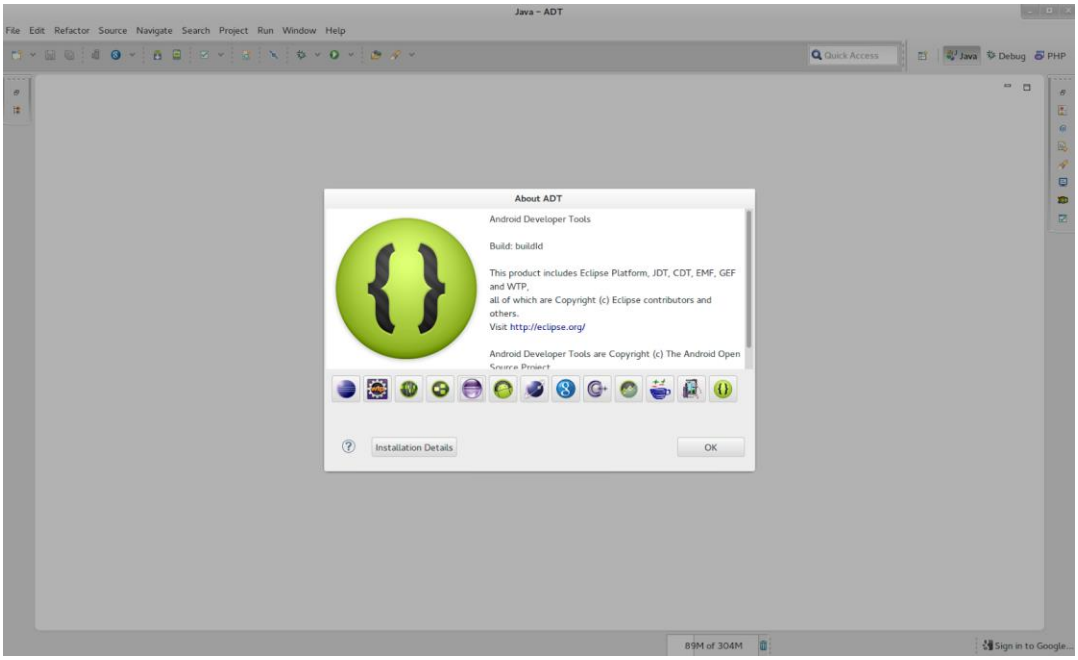

Figura 56: Herramienta de desarrollo “Eclipse”

Herramienta para el programador desarrollada principalmente para el desarrollo de aplicaciones Java, es posible añadirle nuevas funcionalidades al editor, a través de nuevos módulos (plugins), para programar en otros lenguajes de programacion ademas de Java como C/C++, Ruby, Cobol, etc.

Es además el entorno más utilizado por los desarrolladores de aplicaciones Android, por esto ha sido elegido para la realización de este proyecto.

Por otro lado, la parte servidora, consiste en una API REST, que ha sido implementada en PHP y la base de datos es MySQL. Para el desarrollo del código PHP se ha utilizado un editor de texto genérico como Notepad++ y para el manejo de la base de datos MySQL se ha usado la herramienta phpMyAdmin.

Notepad++ es un editor de código fuente gratuito, que soporta varios lenguajes de programación y se ejecuta bajo sistemas operativos Windows (Véase figura 57). Algunas de las características más destacables de Notepad++ son:

- Sintaxis coloreada y envoltura de sintaxis.
- Autocompletado.
- Soporte para buscar/reemplazar expresiones regulares.
- Resaltado de paréntesis y sangrías.
- Grabación y reproducción de macros.
- Multi-Vista.
- Multi-Documento.

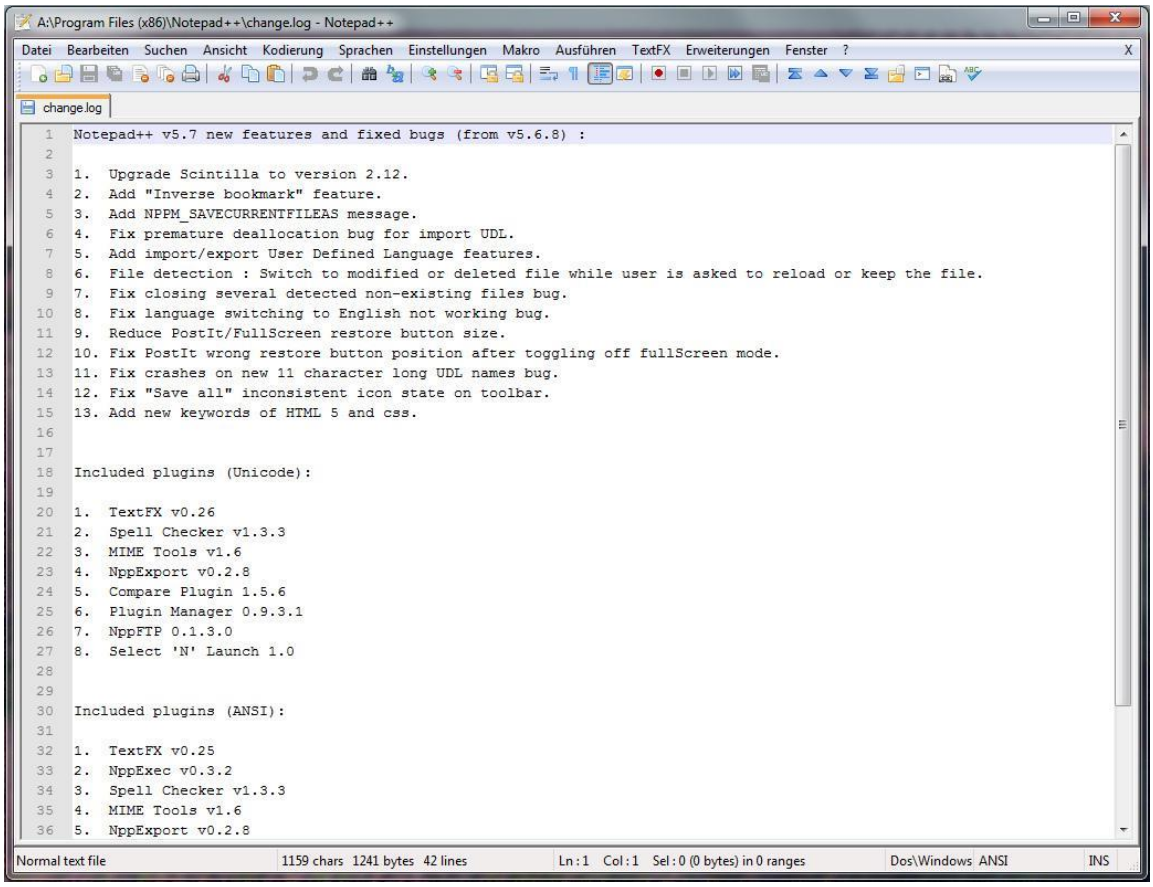

Figura 57: Herramienta de desarrollo “Notepad++”

PhpMyAdmin es una herramienta escrita en PHP con la intención de manejar la administración de MySQL a través de páginas Web, utilizando Internet. Actualmente permite la creación y eliminación de bases de datos, creación, eliminación y modificación de tablas, borrar, editar y añadir campos, ejecutar sentencias SQL, administrar claves en campos, administrar privilegios, y exportar e importar datos en varios formatos. Está disponible en 50 idiomas, y bajo licencia GPL (Véase figura 58).

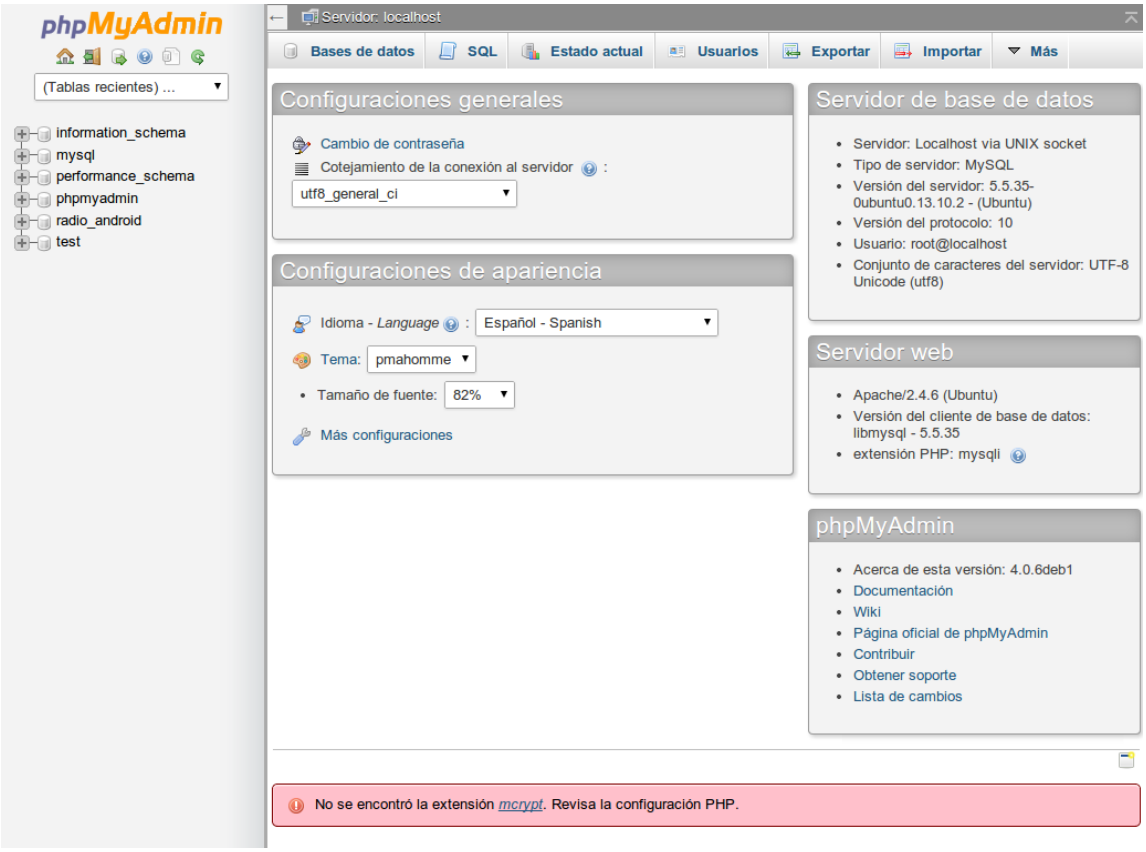

Figura 58: Herramienta de desarrollo “phpMyAdmin”

CAPÍTULO 5
Conclusiones

Actualmente vivimos en una sociedad en la que la cantidad de información que los seres humanos recibimos para realizar prácticamente cualquier tarea es cada vez mayor, llegando a ser casi imposible de procesar de forma natural. El fuerte desarrollo de la Sociedad de la Información se debe, en buena parte, a la mejora de las comunicaciones en los últimos años, el aumento de los medios de almacenamiento de información y, sobretodo, a la rápida y reciente explosión de Internet.

En muchas de las operaciones que los usuarios realizan con frecuencia en Internet, se hace presente el problema de la sobrecarga de información, por culpa del cual en muchas ocasiones el usuario que realiza una búsqueda en Internet pueda llegar a sentirse agobiado y confundido ante una gran cantidad de información difícil de asimilar, e incluso abandonar su intento de búsqueda.

Para solucionar este problema, aparecieron los Sistemas de Recomendación, que ayuda al usuario a encontrar aquella información de su interés mediante recomendaciones basadas en información proveniente del propio usuario, de otros usuarios o de expertos en la materia. Así, esta herramienta abre las puertas a los usuarios a un mundo donde únicamente reciben la información que realmente quieren recibir. Es por esta razón que, en un corto intervalo de tiempo, los sistemas de recomendación han alcanzado una gran popularidad, que se estima que aumente en los próximos años.

En este proyecto se ha desarrollado una aplicación Android de una radio online capaz de proporcionar a sus usuarios canciones de su agrado, además de otras canciones que, sin haber sido evaluadas por el usuario, se estima que sean de su gusto. Para ello, la aplicación se ha desarrollado en torno a un Sistema de Recomendación Colaborativo, siguiendo todo el sistema en su conjunto la arquitectura cliente/servidor.

Desde los primeros momentos de la concepción del proyecto, la intención era crear un servicio que permitiera a cualquier tipo de aplicación acceder al sistema de recomendación musical. De esta manera diferentes usuarios desde distintos sistemas se registren en él, escuchen diferentes canciones de distintos artistas y géneros musicales y realicen puntuaciones sobre las canciones. Basándose en estas puntuaciones, el Sistema de Recomendación crea un perfil de usuario y ofrece al usuario una serie de canciones recomendadas, de acuerdo con los gustos del propio usuario y de otros usuarios de gustos similares a él.

Para la realización del proyecto hemos recopilado un conjunto de datos musicales. Para ello, hemos optado por la obtención de una base de datos de álbumes musicales reales, todos ellos bajo algunas licencia de libre distribución del tipo Creative Commons, de manera que la versión prototipo de nuestro sistema cuenta con 12111 canciones de 18 géneros musicales diferentes. El servidor no aloja en ningún momento ningún archivo musical, sino que se solicitan directamente a la base de datos de música.

Además, hemos realizado un estudio de los diferentes tipos de Sistemas de Recomendación, el funcionamiento de los distintos algoritmos de filtrado y, basándonos en el estudio previo realizado en el proyecto “Radio online basada en un motor de filtrado colaborativo” [33], hemos reutilizado el algoritmo de filtrado colaborativo basado en ítem y, más concretamente, en el vecino más cercano (K-nn) implementado en el proyecto de Iván Palomares. De forma paralela, hemos desarrollado la parte cliente del proyecto, con el análisis, diseño e implementación de una aplicación Android usable para la comunicación con el usuario, donde el usuario puede escuchar y evaluar canciones y obtener recomendaciones.

Para el desarrollo de este proyecto hemos seguido todas las etapas de la Ingeniería del Software. En primer lugar, se determinaron las propiedades que el sistema debía satisfacer, así como las restricciones a las que se encuentra sometido. Seguidamente se ha creado un modelo del sistema correcto, completo, consistente, claro y verificable. Finalmente se ha codificado este modelo en una versión prototipo y se ha instalado en el servidor.

Por último, añadir que para el autor del proyecto ha supuesto una enorme satisfacción el haber conseguido darle forma a una idea que, desde bastante tiempo atrás, habían plasmado tanto él como los tutores del proyecto; y el ver cómo poco a poco, y a pesar de las pequeñas adversidades que inevitablemente siempre van surgiendo durante el desarrollo de cualquier aplicación, mi aplicación ha conseguido llegar a buen puerto. Además, este proyecto me ha aportado conocimientos, a veces totalmente nuevos y otras veces para profundizar, en campos como el de los Sistemas de Recomendación, el desarrollo Android, el diseño y administración de bases de datos y la propia Ingeniería del Software. Esto ha supuesto para mí un verdadero reto desde los primeros momentos de su realización, pero a la vez y sobretodo, ha sido una experiencia muy positiva y enriquecedora.

Anexo A Manual de instalación

Antes de comenzar con el manual de instalación del sistema vamos a describir el contenido del DVD que acompaña al proyecto ya que algunos contenidos serán necesarios para la instalación del sistema. El contenido del DVD es el siguiente:

- PDF con la documentación del proyecto.
- Software VirtualBox 4.3.12 para Windows.
- Backup de la máquina virtual utilizada para el servidor.
- Código de la parte Servidor (RadioOnlineService).
- Código del Cliente Android (RadioOnlineApp).
- Video demostración de la aplicación Android.

Dado que nuestro sistema está basado en una arquitectura cliente servidor y se han implementado ambas partes, se van a describir los pasos para la instalación de ambas partes.

Servidor

Para la instalación de la parte servidor habría que configurar los puertos de acceso, aparte de instalar el software siguiente:

- Gestor de bases de datos MySQL.
- Java 1.7
- PHP 5
- phpMyAdmin

Por simplicidad de instalación se ha decidido incluir una copia de la máquina virtual utilizada en el sistema desplegado. A continuación se va a proceder a instalar la máquina virtual con el software VirtualBox incluido en el DVD. Por último habría que modificar la dirección de acceso en el proyecto Android.

Pasos a seguir:

1. Ejecutamos el instalador y pulsamos “Next” para continuar con la instalación (Véase figura 59).

Figura 59: Instalador de VirtualBox

2. Aceptamos la instalación por defecto y pulsamos “Next” para que comience la instalación (Véase figura 60).

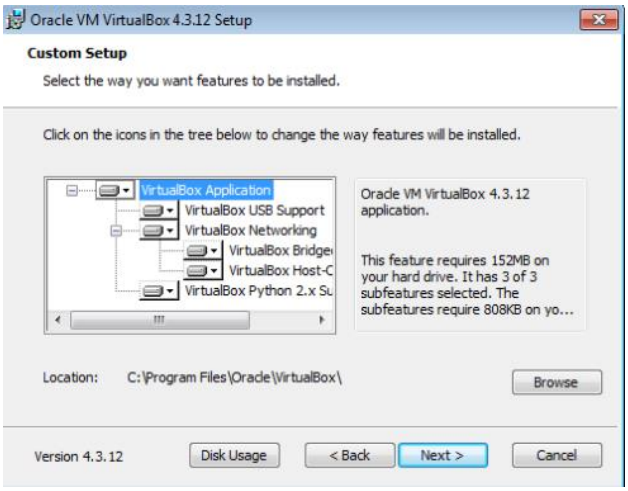

Figura 60: Instalación por defecto VirtualBox

3. La instalación ha finalizado y pulsamos “Finish” para abrir la aplicación (Véase figura 61).

Figura 61: Instalación finalizada de VirtualBox

4. Dentro de VirtualBox pulsamos Archivo > Importar servicio virtualizado... (Véase figura 62).

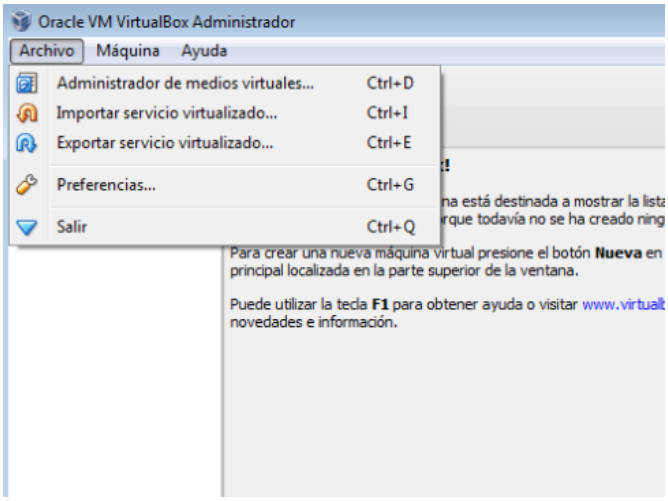

Figura 62: Abrir menú importar en VirtualBox

5. Seleccionamos el archivo “PFC-Radio.ova” dentro del DVD y pulsamos “Next” para continuar con la importación (Véase figura 63).

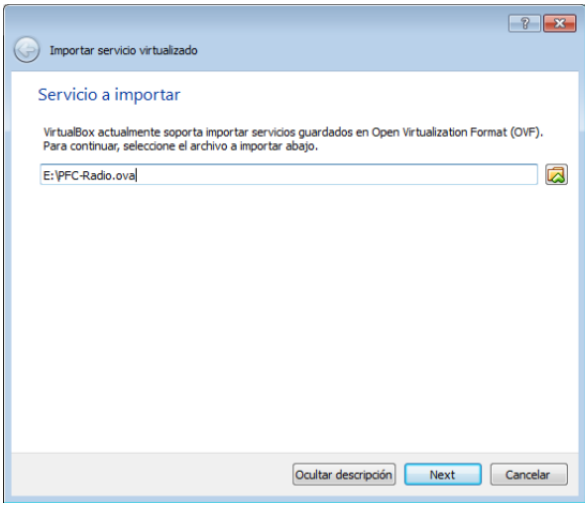

Figura 63: Seleccionar backup máquina virtual

6. Por último pulsamos “Importar” y comenzará la importación (Véase figura 64).

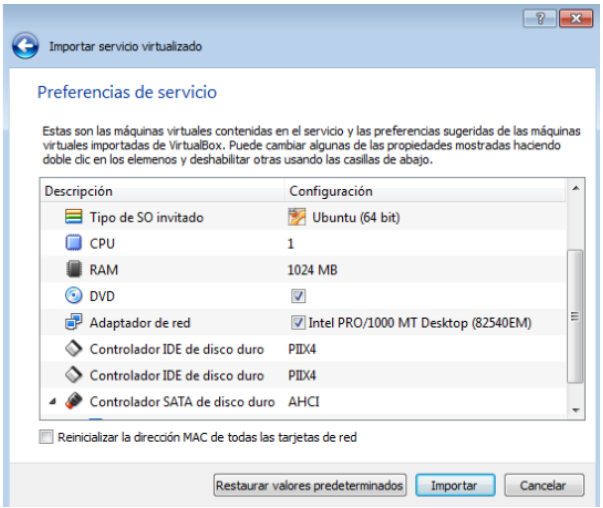

Figura 64: Información de la máquina virtual

7. Ahora nos aparecerá la nueva máquina virtual, ahora pulsando “Iniciar” la máquina arrancará (Véase figura 65).

Figura 65: Iniciamos la máquina virtual

8. Por último, como podemos ver en la figura 66, simplemente tenemos que introducir el usuario y la contraseña (están en la descripción de la máquina virtual).

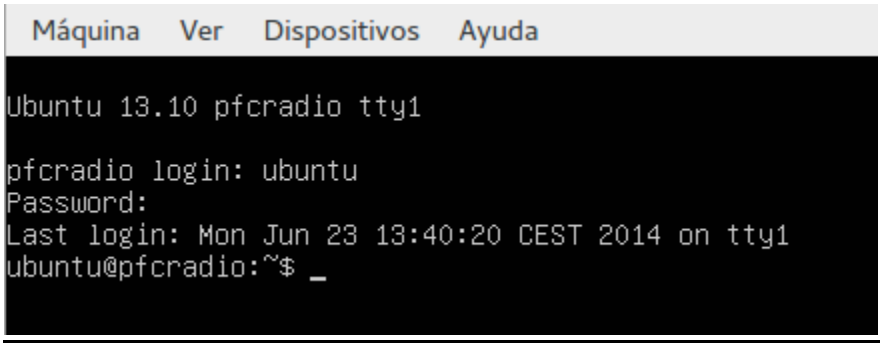

Figura 66: Inicio de sesión en la máquina virtual

9. Para poder ejecutar la aplicación es necesario cambiar la URL de despliegue por la dirección IP de la máquina virtual en la aplicación Android, concretamente en el paquete utilidades, dentro de la clase Utilidades en la línea 26 (Véase figura 67).

```
public static String baseUrl = "http://ceatic.ujaen.es:
```

Figura 67: URL base del servidor

Cliente

A continuación se van a describir cada uno de los pasos que se han de seguir para instalar la aplicación cliente en un dispositivo Android. Para ello, en primer lugar, se ha de comprobar que la versión del sistema operativo Android del dispositivo es igual o superior a 4.0 y que Google Play Services está instalado en el dispositivo. En el DVD que se adjunta existe una carpeta llamada RadioOnlineApp que contiene el código fuente de la aplicación y todos los archivos necesarios para su instalación.

Pasos a seguir:

1. Activar la opción Fuentes desconocidas dentro de Ajustes > Seguridad de nuestro dispositivo. Tras ello se pedirá la confirmación a través de una ventana y el usuario debe aceptar (Véase figura 68).

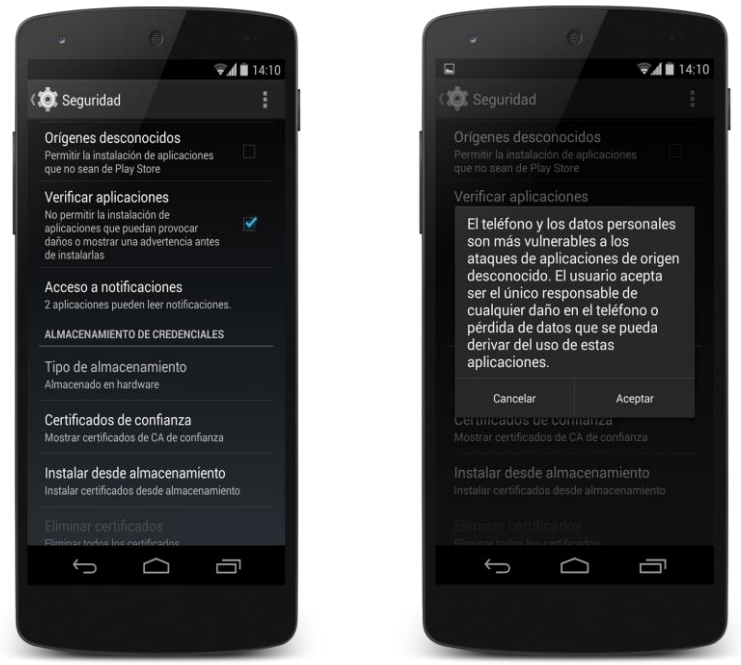

Figura 68: Activación de opciones de instalación en el dispositivo

2. Acceder al directorio RadioOnlineApp/bin y copiar el archivo Radio.apk. En Android, los archivos con esta extensión son archivos comprimidos que contienen toda la información necesaria para poder ejecutar una aplicación en un dispositivo.
3. Conectar el dispositivo mediante cable USB al ordenador.
4. Acceder a la memoria del dispositivo y pegar el archivo Radio.apk (Véase figura 69).

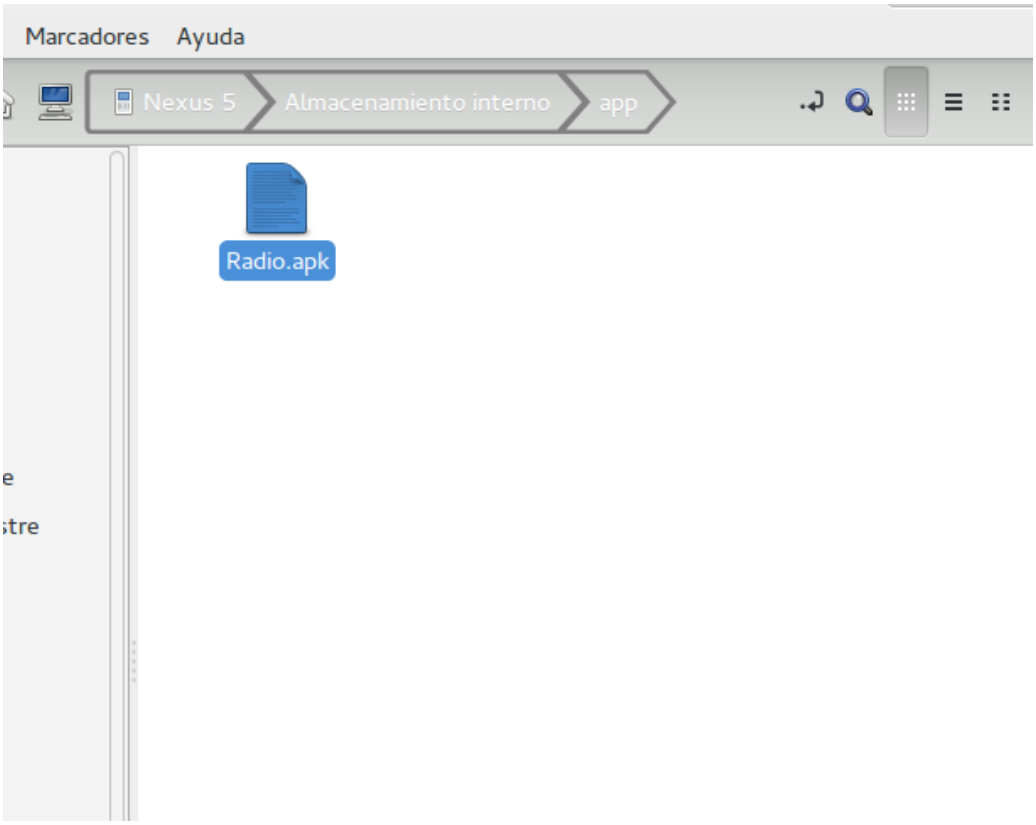

Figura 69: Copiar el archivo Radio.apk en la memoria del dispositivo

5. Una vez que el archivo se ha transferido al dispositivo bastará con seleccionarlo desde un administrador de archivos en Android para poder instalar la aplicación (Véase figura 70).

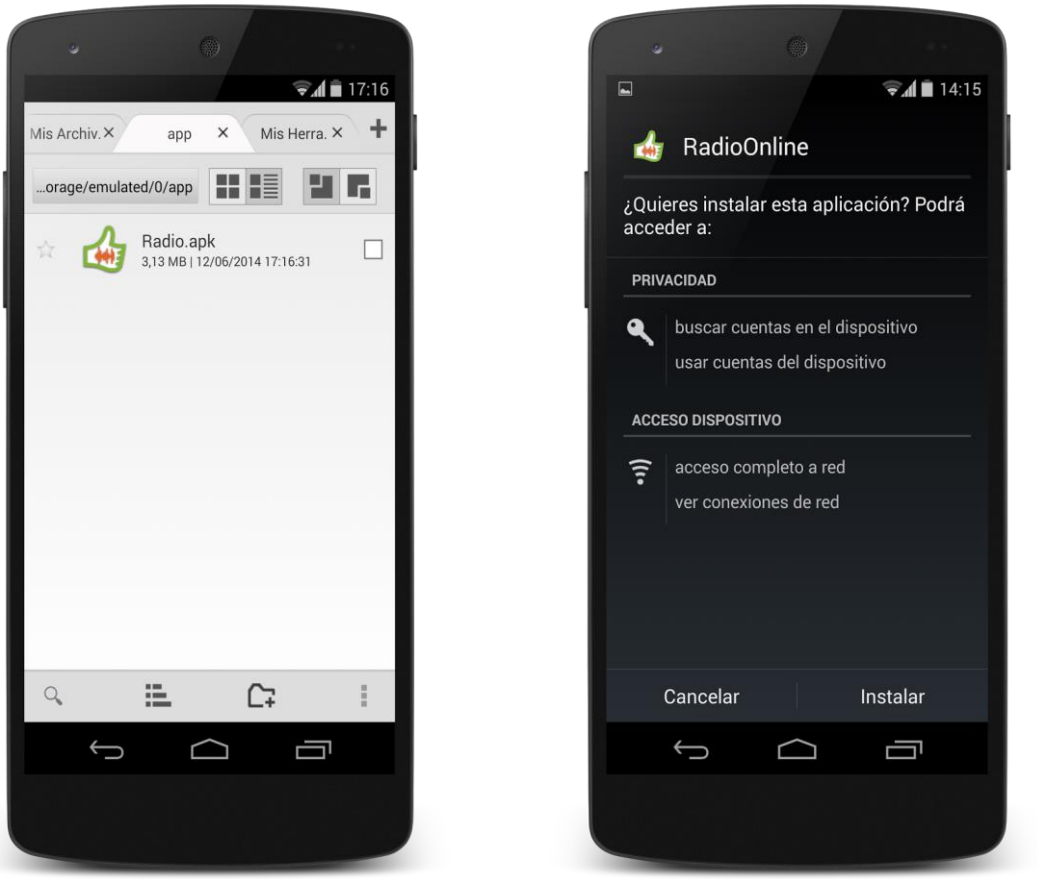

Figura 70: Buscando la aplicación en el dispositivo e instalando la aplicación

6. Una vez se pulsa “Instalar” se procederá a la instalación y posteriormente se podrá acceder a la aplicación (Véase la figura 71).

Figura 71: Aplicación RadioOnlineApp instalada en el dispositivo

Anexo B Manual de usuario

A lo largo de este manual vamos a describir cómo acceder a toda la funcionalidad de la aplicación para permitir realizar un uso correcto de la misma. Para ello, vamos a seguir un orden de ejecución de acuerdo al orden en que se han colocado las distintas funciones dentro de la pantalla principal, de más usadas a menos usadas.

Identificación

Para acceder a todo lo que RadioOnlineApp ofrece hemos de identificarnos en el sistema con nuestra cuenta de Google. Como podemos ver en la figura 72 bastará con pulsar sobre iniciar sesión y Google nos mostrará una pantalla para aceptar que cedemos nuestros datos. Una vez cedamos nuestros datos de Google nos identificará en el sistema.

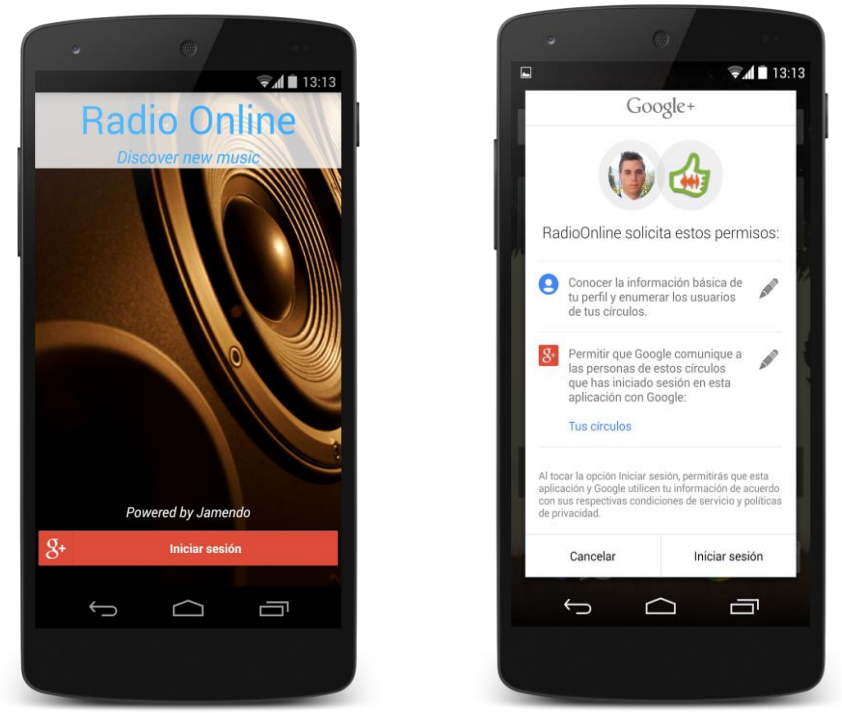

Figura 72: Proceso iniciar sesión en la aplicación

A continuación la aplicación nos mostrará la pantalla del reproductor de música en modo recomendación (Véase figura 73). El sistema devolverá las canciones recomendadas para el usuario, se actualizarán los datos de la canción (nombre, autor, imagen, duración,...) en la interfaz y comenzará a reproducir la canción.

Figura 73: Pantalla reproductor musical al iniciar sesión

Reproducir Radio Normal

Podemos acceder a la radio normal desde el drawer lateral izquierdo de la aplicación en la opción “Radio Normal”, como podemos ver en la figura 74. A continuación se cerrará el drawer y el sistema solicitará la lista de reproducción. Una vez la aplicación tenga la lista, de igual manera que antes se actualizarán los datos de la canción en la interfaz y comenzará la reproducción.

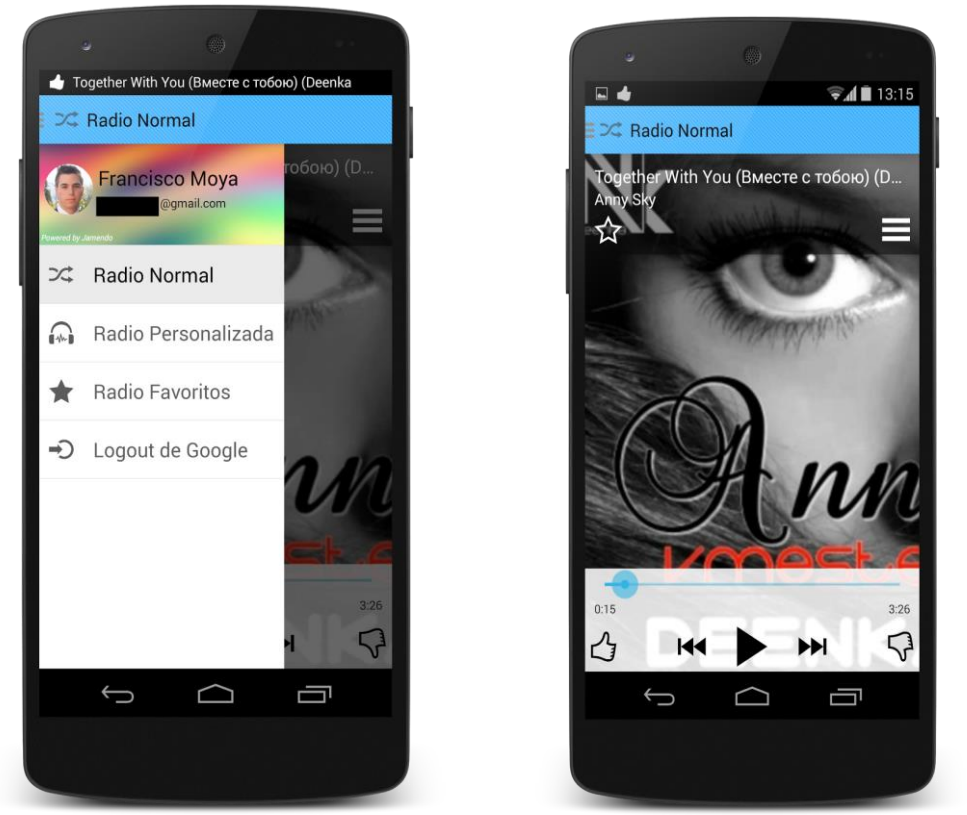

Figura 74: Proceso radio normal en la aplicación

Reproducir Radio Favoritos

Podemos acceder a la radio favoritos desde el drawer lateral izquierdo de la aplicación en la opción “Radio Favoritos”, como podemos ver en la figura 75. A continuación se cerrará el drawer y el sistema solicitará la lista de reproducción. Una vez la aplicación tenga la lista, de igual manera que antes se actualizarán los datos de la canción en la interfaz y comenzará la reproducción.

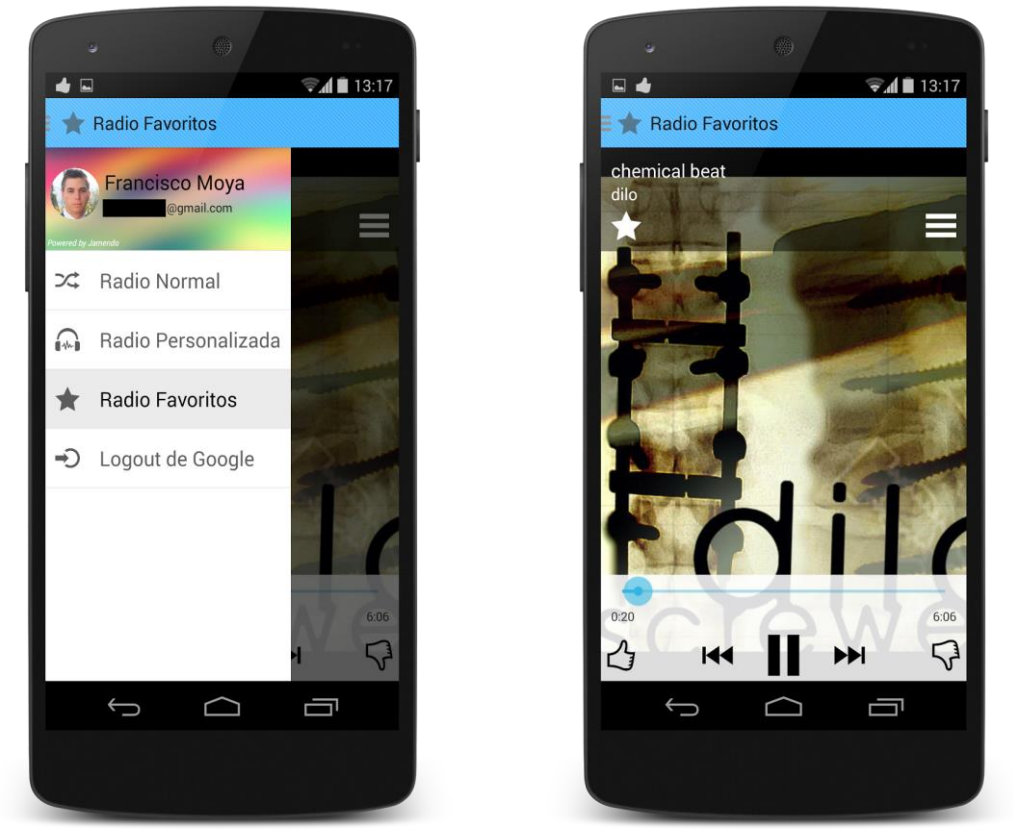

Figura 75: Proceso radio favoritos en la aplicación

Valorar una canción

Desde la pantalla del reproductor musical disponemos de varias formas de valorar una canción. Si la canción te gusta simplemente pulsando el icono de “Me gusta”, como podemos ver en la figura 76, la canción se valorará positivamente en el sistema.

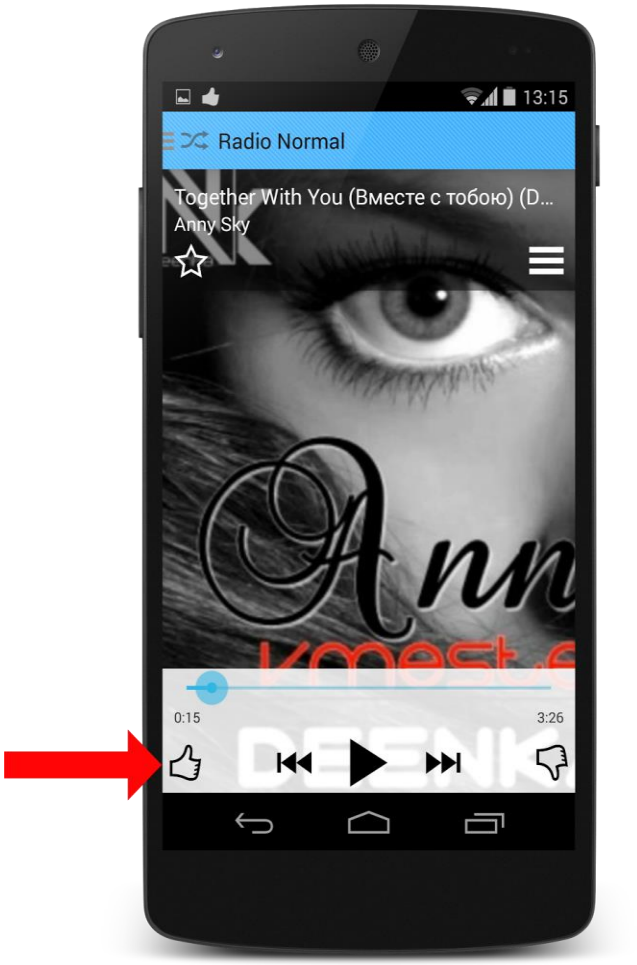

Figura 76: Valorar con “Me gusta” en la aplicación

En el caso de que no te guste la canción simplemente pulsando el icono de “No me gusta”, como podemos ver en la figura 77.

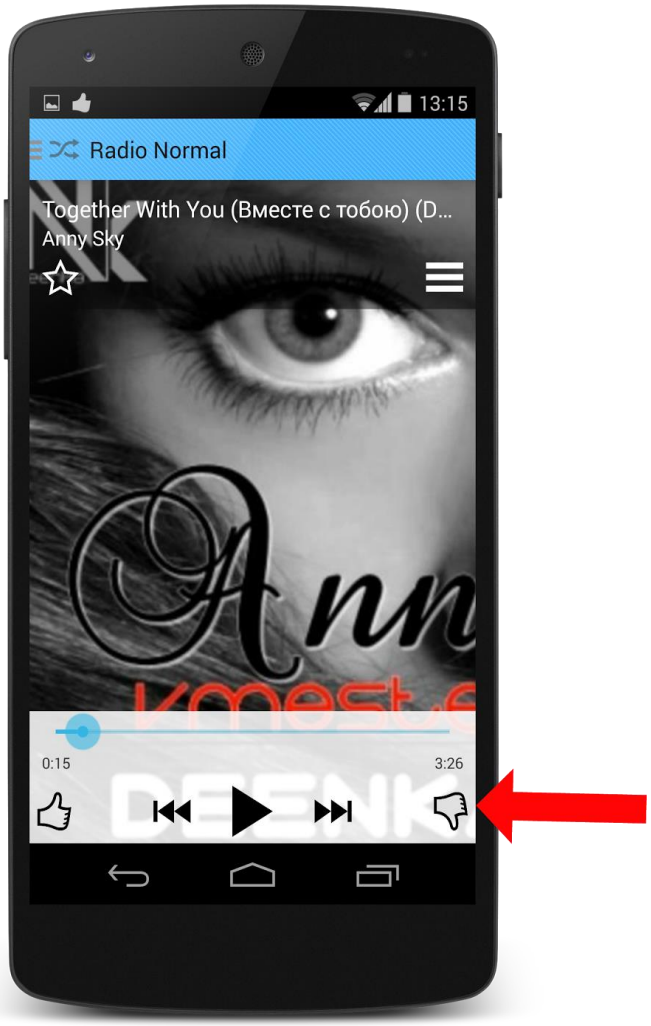

Figura 77: Valorar con “No me gusta” en la aplicación

Por último, aparte de decir si te gusta o no una canción, puedes marcar una canción como favorita pulsando el icono de “Favorito”, como podemos ver en la figura 78.

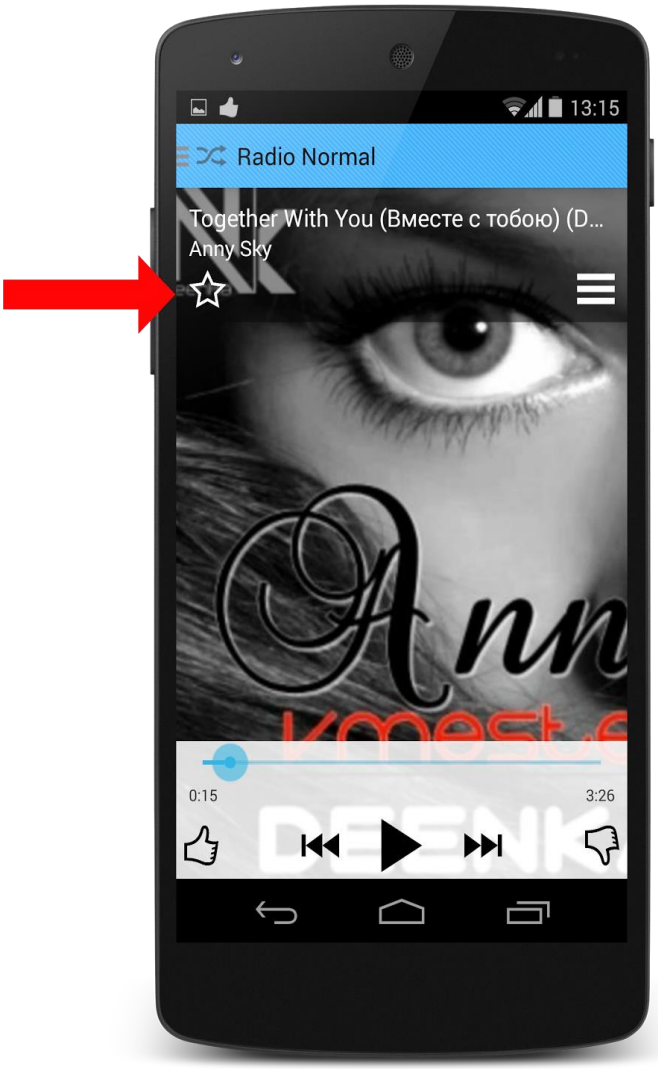

Figura 78: Valorar con “Favorita” en la aplicación

Desvincular la cuenta de Google

Podemos desvincular la cuenta de Google desde el drawer lateral izquierdo de la aplicación en la opción “Logout de Google”, como podemos ver en la figura 79. Ahora se vuelve a la pantalla de inicio de sesión por si el usuario quisiera volver a iniciar sesión.

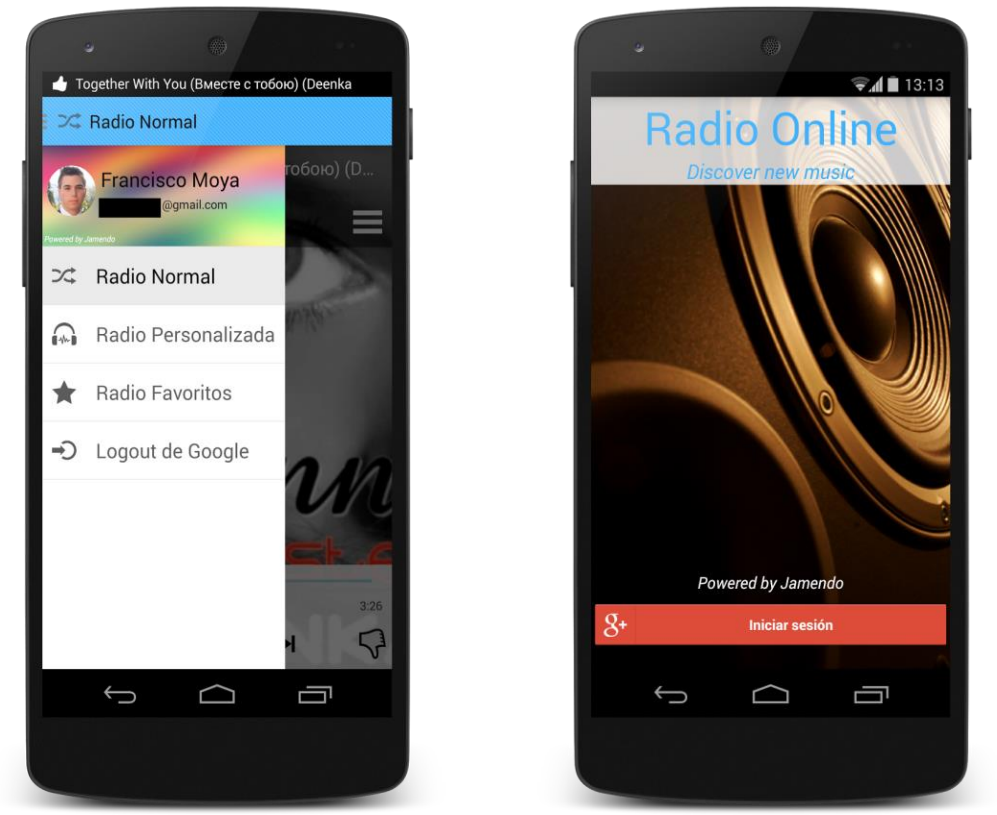

Figura 79: Proceso de desvinculación de la cuenta de Google en la aplicación

Ver lista de reproducción

Mientras que el usuario se encuentre en la pantalla Reproductor el usuario puede ver la lista de canciones que se están reproduciendo en esa radio. Simplemente tiene que pulsar sobre el icono de “Lista de reproducción” y mostrará una subpantalla emergente con una lista con las canciones que contiene la radio, como podemos ver en la figura 80. Si el usuario pulsa sobre cualquiera de las canciones la canción comenzará a reproducirse instantáneamente.

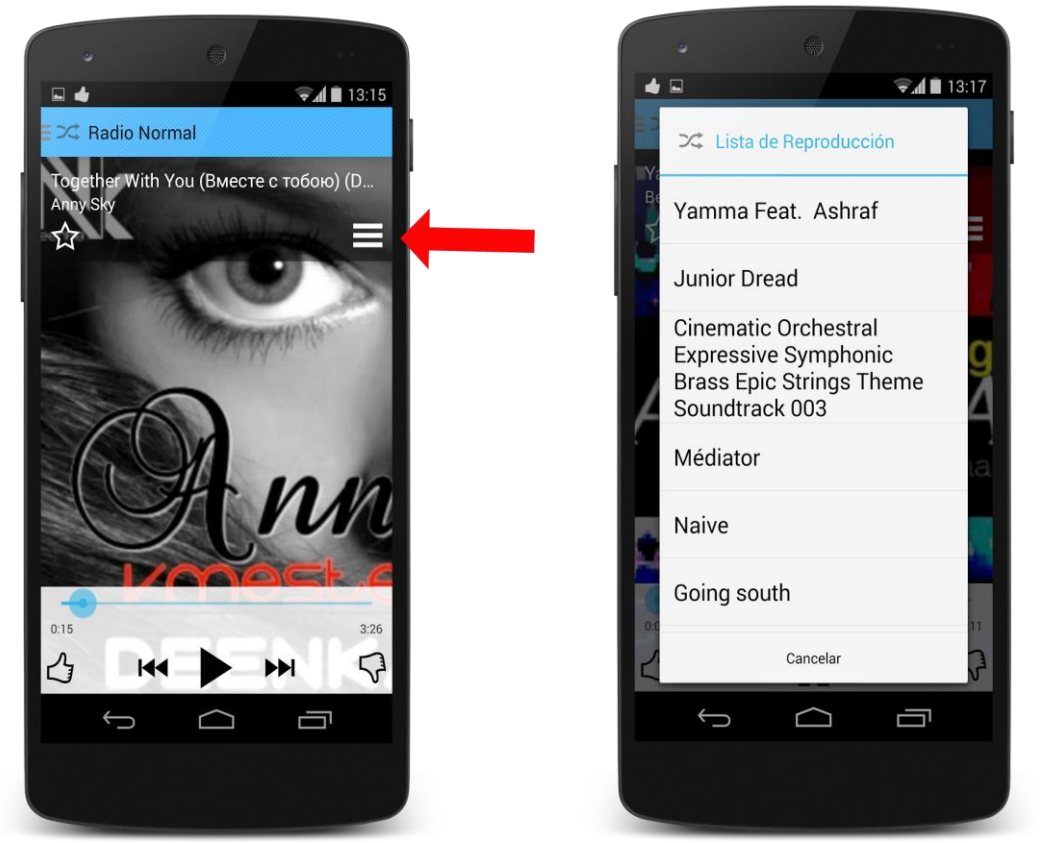

Figura 80: Ver lista de reproducción en Android

Bibliografía y Referencias

[1] Creative Commons España, http://es.creativecommons.org. Último acceso: Junio 2014.

[2] Adomavicius, G. and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering, 2005. 17(6): p. 734-749.

[3] Herlocker, J.L., et al., Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems, 2004. 22(1): p. 5-53.

[4] Ricci, F., L. Rokach, and B. Shapira, Introduction to Recommender Systems Handbook, in Recommender Systems Handbook, F. Ricci, et al., Editors. 2011, Springer US. P. 1-35

[5] Adomavicius, G., et al., Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems, 2005. 23(1): p. 103-145.

[6] Balabanovic, M. y Shoham, Y. (1997), “Content-based, callaborative recommendation”. Communications of the ACM, Vol.40

[7] Cho, Y.H., J. Kyeong Kim, “A personalized recommender system based on Web usage mining and decision tree induction” Expert Systems with Applications, Volume 23, Issue 3, 1 October 2002, Pages 329-342.

[8] Burke, R., Knowledge-based Recommender Systems. Encyclopedia of Library and Information Systems, 2000. 69(32).
[9] Arazy, O., N. Kumar, and B. Shapira, Improving Social Recommender Systems. IT Professional, 2009. 11(4): p. 38-44.

[10] Pazzani, M.J., A Framework for Collaborative, Content-Based and Demographic Filtering. Artificial Intelligence Review, 1999. 13(5-6): p. 393-408.

[11] Jung, K.Y., Choi, J.H., Rim, K:W:, Lee, J.H., "Development of Design Recommender System Using Collaborative Filtering", Digital libraries: technology and management of indigenous knowledge for global access, Vol. 2911, 2003.

[12] Papagelis, M., Plexousakis, D., Rousidis, I., Theoharopoulos, E., "Qualitative Analysis of User-based and Item-based Algorithms for Recommendation Systems", Institute of Computer Science, Foundation for Research and Technology- Hellas.

[13] Yu, K., Xu, X., Tao, J., Ester, M., Kriegel H.P., "Instance Selection Techniques for Memory-Based Collaborative Filtering", 12th international workshop on database and expert systems applications, proceedings, 2001.

[14] Breese, J., Heckerman, D., Kadie, C., "Empirical Analysis of Predictive Algorithms for Collaborative Filtering"

[15] Jesús Tomás Gironés: "El gran libro de Android", Marcombo 2013.

[16] Wallace Jackson: "Android Apps for absolute beginners", Apress 2012.

[17] Open Handset Alliance, http://www.openhandsetalliance.com/. Último acceso: Junio 2014.

[18] http://developer.android.com/about/dashboards/index.html. Último acceso: Junio 2014.
[19] http://source.android.com/source/licenses.html. Último acceso: Junio 2014.

[20] http://developer.android.com/guide/components/fundamentals.html. Último acceso: Junio 2014.
[21] http://developer.android.com/guide/topics/security/permissions.html. Último acceso: Junio 2014.

[22] http://developer.android.com/guide/topics/data/data-storage.html. Último acceso: Junio 2014.

[23] Service (systems architecture), http://en.wikipedia.org/wiki/Service_(systems_architecture) . Último acceso: Junio 2014.

[24] SOA, http://www.slideshare.net/Mache007/arquitectura-orientada-a-servicios-soa-12818946. Último acceso: Junio 2014.

[25] Servicio Web, http://es.wikipedia.org/wiki/Servicio_web . Último acceso: Junio 2014.

[26] SOAP y REST, http://carlosmayta.blogspot.com.es/ . Último acceso: Junio 2014.

[27] Dawson, C.W. y Martin, G. (2002), “El proyecto fin de carrera en Ingeniería Informática: Una guía para el estudiante” Ed. Prentice Hall.

[28] Breet D. McLaughli, Pollice, G., West, D. “Head First Object-Oriented Analysis and Design”. Ed. O’Reilly.

[29] Conde, F., “Interacción Persona-Ordenador: ¿Pero qué narices es una buena interfaz de usuario?”. Apuntes de la asignatura.

[30] Freeman, E., Sierra, K., Bert Bates, “Head First Design Patterns”. Ed.O’Reilly.

[31] Feito Higueruela, F., Ruiz de Miras, J., Molina Aguilar, A. (1996), “Análisis y Gestión de Datos”, Editorial Universidad de Jaén.

[32] Software RecommenderLib, http://ceatic.ujaen.es/redmine/projects/recommenderlib/ . Último acceso: Junio 2014.

[33] PFC Radio Online basada en un motor de Filtrado Colaborativo: http://sinbad2.ujaen.es/cod/archivosPublicos/pfc/pfc_ivan_palomares.pdf

[34] Zagat, http://en.wikipedia.org/wiki/Zagat. Último acceso: Junio 2014.

[35] Foursquare, http://es.wikipedia.org/wiki/Foursquare. Último acceso: Junio 2014.

[36] U. Shardanand and P. Maes, “Social information filtering: Algorithms for automating “word of mouth”,” pp. 210–217, 1995.