Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN

Escuela Politécnica Superior (Jaén)

Trabajo Fin de Grado

SISTEMA WEB DE RECOMENDACIÓN MUSICAL BASADO EN SERVICIOS PARA APP ANDROID

Alumno: Juan Bailén Sevilla

Tutor: Prof. Dr. Luis Martínez López

D. Jorge Castro Gallardo

Dpto: Informática

Junio, 2015

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dr. D. Luis Martínez López y D. Jorge Castro Gallardo, tutores del Proyecto Trabajo Fin de Grado titulado: Sistema Web de Recomendación Musical basado en servicios para app Android, que presenta D. Juan Bailén Sevilla, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2015

El Alumno

D. Juan Bailén Sevilla

Los Tutores

Dr. D. Luis Martínez López

D. Jorge Castro Gallardo

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, Alfonso y Maty, porque gracias a su esfuerzo y sacrificio, he conseguido llegar hasta aquí.

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

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

Mi amigo y vecino, José Manuel, por haberme apoyado desde el primer hasta el último día de estos 4 años de carrera.

Todos los que me habéis apoyado durante la realización de este proyecto.

MUCHAS GRACIAS A TODOS.

Índice

Agradecimientos 5
Índice 7
Capítulo 1 Introducción 11
1.1 Introducción al proyecto 13
1.2 Propósito 15
1.3 Objetivos 15
1.4 Planificación temporal 16
1.5 Planificación de costos 18
1.6 Estructura del proyecto 21
Capítulo 2 Sistemas de recomendación 23
2.1. Introducción 25
2.2 Esquema Básico de Funcionamiento y Elementos de un SR 28
Realimentación en los SR 29
Datos Reales vs. Datos Sintetizados 29
Análisis Online vs. Análisis Offline 30
2.3. Clasificación de los Sistemas de Recomendación 30
2.3.1 Sistemas de Recomendación Basados en Contenido 30
2.3.2 Sistemas de Recomendación Colaborativos 31
2.3.3 Sistemas de Recomendación Basados en Conocimiento 32
2.3.4 Sistemas de Recomendación Basados en la Comunidad 32
2.3.5 Sistemas de Recomendación Demográficos 33
2.3.6 Sistemas de Recomendación Híbridos 33
2.4. Sistemas de Recomendación Colaborativos 35
Introducción y Orígenes 35
Algoritmos de Filtrado Colaborativo 36
Algoritmos basados en memoria o basados en usuario. 36
Algoritmos basados en modelos o basados en ítem 37
Problemas de los Sistemas de Recomendación Colaborativos 41
2.5 Aplicaciones de los Sistemas de Recomendación 42
Aplicaciones de Sistemas de Recomendación Colaborativos 44
Capítulo 3 Arquitectura y servicios 49
3.1 Introducción 51

3.2 Servicios web 51
Servicios REST 52
Métodos y códigos de estado HTTP 54
Estructura de las URLs 55
3.3 Patrones de diseño 55
Definición 55
Cualidades de un patrón de diseño 56
Patrón Controlador 56
Ventajas y desventajas del uso del patrón 58
3.4 Música en Android: Streaming 59
Aplicaciones de música Streaming en Android 61
Capítulo 4 Proceso de ingeniería software 63
4.1 Introducción 65
4.2 Análisis del sistema 66
Requisitos funcionales 66
Perfiles de Usuario 67
Casos de uso 68
Diagrama Frontera 68
Gestión usuario 69
Gestión reproducción 70
Gestión musical 71
Diagramas de secuencia 72
Radio personalizada 72
Escenarios 73
Gestión usuario 73
Radios 74
Gestión musical 76
4.3 Diseño 77
Diseño del Sistema 77
Diagrama de clases 78
Diagrama de Paquetes 79
Diseño de Datos 79
Esquema Entidad-Relación 80

Esquema conceptual modificado 81
Diseño de Interfaz 82
Diseño web 82
Metáforas 84
Prototipos 87
Story Board 90
4.4 Implementación 92
Arquitectura Cliente-Servidor 92
Componentes de la arquitectura Cliente-Servidor 93
Características de la arquitectura Cliente-Servidor 94
Ventajas de la arquitectura Cliente-Servidor 95
Desventajas de la arquitectura Cliente-Servidor 96
Framework 97
Definición 97
Spring 97
Ventajas de usar framework 98
Inconvenientes de usar framework 98
Lenguajes de desarrollo utilizados 99
Parte Cliente 99
Parte Servidor 99
4.5 Pruebas Software 102
Pruebas Unitarias 102
Características 102
Ventajas 103
Limitaciones 103
Pruebas de integración 104
Ventajas 104
Limitaciones 104
Casos de Test 105
Resultados 110
Capítulo 5 Conclusiones 115
Apéndice A: Manual de Instalación 119
Servidor 121

Cliente 126
Apéndice B: Manual de usuario 127
Manual de usuario 129
Identificación 129
Control de reproducción 131
Redes Sociales 133
Cuenta 134
Perfil 135
Gestión Musical 137
Radios 138
Bibliografía y Referencias 141

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", la que cual se explica ampliamente en los siguientes apartados, 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 y puntuado.

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 [4]: Basados en Contenido, Colaborativos, 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 el TFG en el desarrollo de una radio online colaborativa basada en servicios web y su interacción con una app 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.

Por tanto en este TFG, se pretende, crear un sitio web que albergue un servicio de Radio Colaborativa con música de licencia Creative Commoms y que ofrezca diversos servicios que puedan ser accedidos a través de una app Android [28].

A continuación se exponen tanto el propósito de esta aplicación como los objetivos necesarios para llevarla a su consecución con éxito.

1.2 Propósito

Diseño y desarrollo de una radio online colaborativa para web, basada en las preferencias musicales del usuario, y unos servicios aprovechables por una aplicación Android para su uso ubicuo.

1.3 Objetivos

1. Búsqueda y revisión bibliográfica.

2. Estudiar los métodos de reproducción de música streaming en sistemas web.

3. Realización de proceso de ingeniería software: Análisis, diseño implementación y pruebas.

4. Estudio, preparación e implementación de los métodos de acceso al servidor web de la aplicación.

5. Implementación de la aplicación web que permitirá escuchar la radio colaborativa en base al algoritmo de filtrado colaborativo especificado.

6. Redacción de la memoria.

A continuación se expone la planificación temporal realizada para el cumplimiento de los objetivos de este TFG.

1.4 Planificación temporal

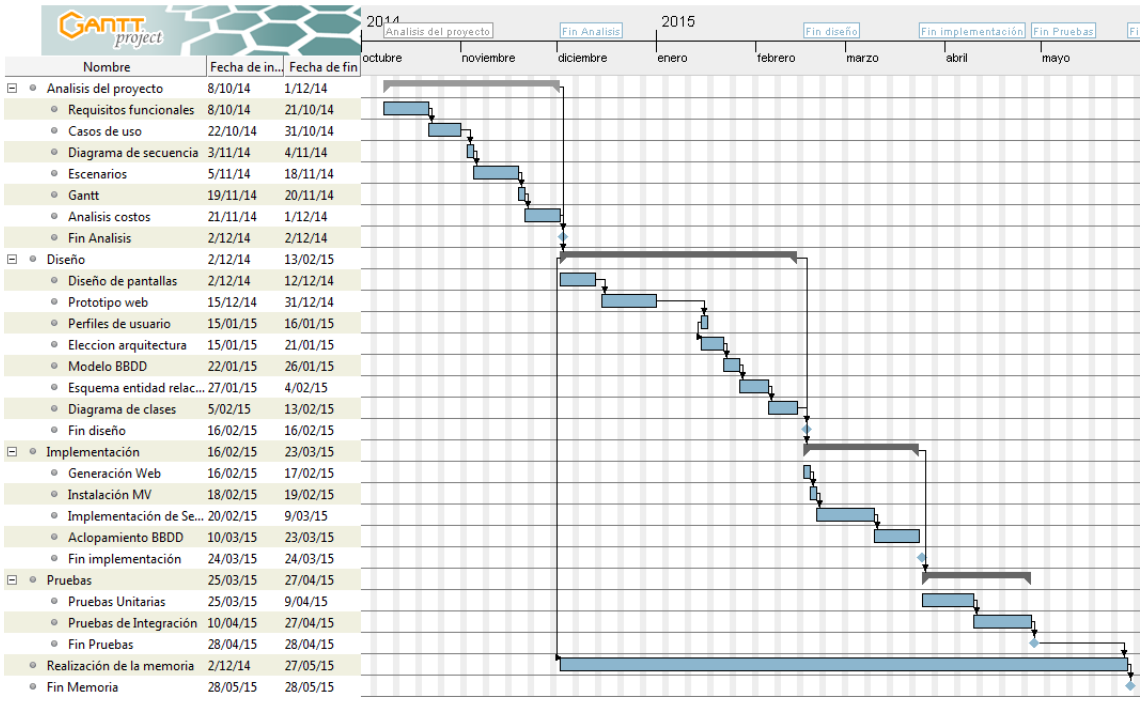

Figura 1.4.1: Diagrama de Gantt

En la figura 1.4.1 se observa la descomposición de tareas llevadas a cabo en este proyecto; así como su planificación temporal a lo largo del curso académico.

El proyecto se divide en cuatro fases; en la fase de análisis, la más importante del proyecto, se exponen los requisitos necesarios para el proyecto, su asignación, así como un análisis de tiempo y costo que se necesitarán en las siguientes fases.

El diseño nos da una imagen de cómo va a quedar el proyecto una vez implementado. Es una guía completa de los pasos a seguir para la consecución en el tiempo predicho con los recursos estimados.

La implementación y las pruebas están íntimamente ligadas, para conseguir cumplir los requisitos no funcionales que se le pueden exigir a este tipo de proyectos.

La memoria es el proceso que esta guiado por todas las demás tareas permitiendo así una documentación correcta y precisa.

En la figura 1.4.2 se muestra un esquema con la jerarquía y secuenciación de tareas mostrando el camino critico que lleva a la consecución del proyecto

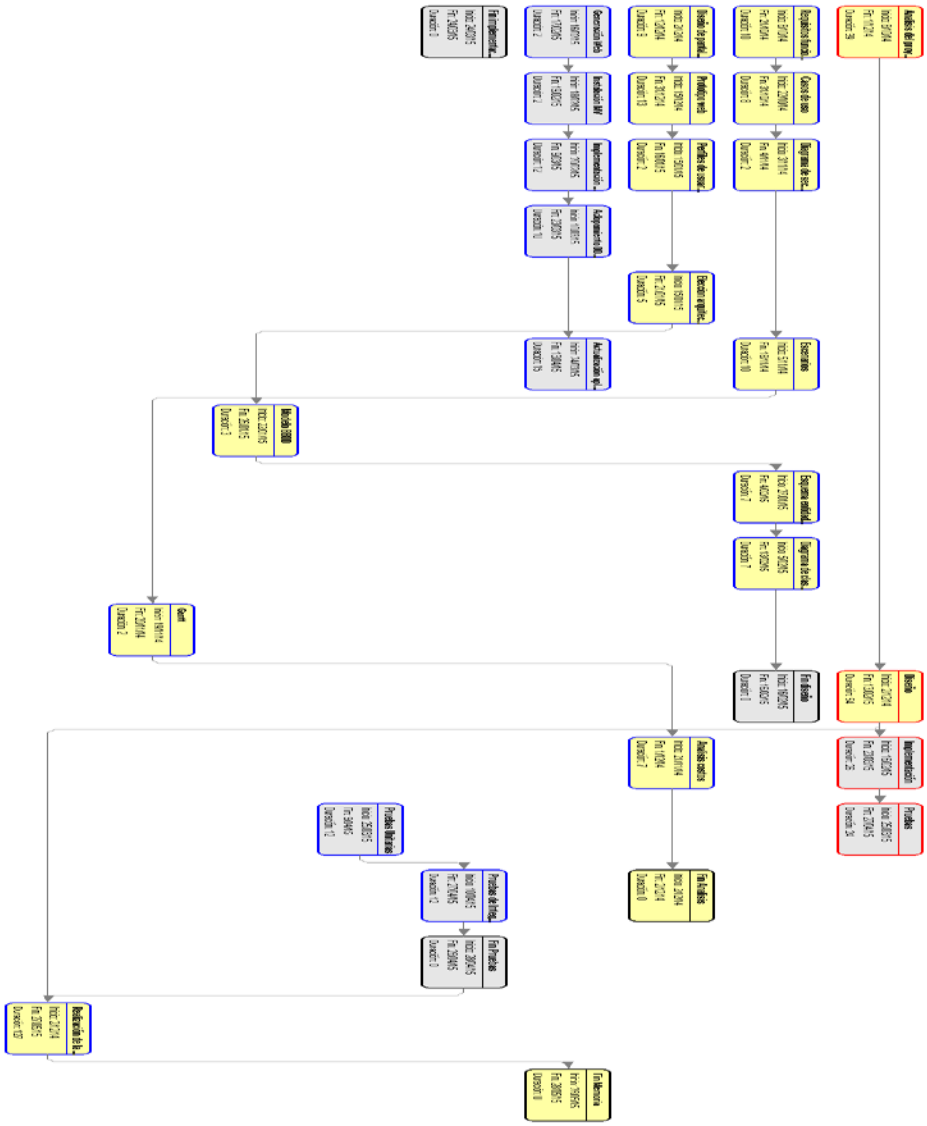

Figura 1.4.2: Diagrama de Actividades

A continuación se muestra la planificación de costos de proyecto, en la cual se realiza una predicción aproximada del tiempo, personal y producción de los mismos necesarios para la consecución del proyecto.

1.5 Planificación de costos

Para la estimación de costes asociados a este proyecto utilizaremos la metodología que nos ofrece el modelo COCOMO. Este modelo, basado en número de líneas de código, esfuerzo por persona y otras variables basadas en conocimientos y experiencia del personal encargado para el proyecto, nos dará una aproximación considerablemente buena tanto del esfuerzo de desarrollo como del tiempo necesario de desarrollo.

Al final de este proceso tendremos un esfuerzo promedio de cada mes así como la productividad necesaria para llevarlo a cabo.

COCOMO utiliza para su cometido unas fórmulas que son explicadas a continuación:

- $E=(a)(KLDC)^{b}(EAF)$
- $D=(c)(E)^{d}$
- PR=LCD/E
- P=E/D

Siendo "E" el esfuerzo aplicado en hombre mes, "D" el tiempo de desarrollo en meses, PR la productividad, P personal promedio necesario, "KLDC" número de miles de líneas de código estimado para el proyecto, LDC número de líneas de código y EAF el factor de ajuste del esfuerzo que se calcula valorando en la siguiente escala de 15 atributos.

ATRIBUTOS	Valor					
	Muy bajo	Bajo	Nominal	alto	Muy alto	Extra alto
Atributos del software						
Fiabilidad	0,75	0,88	1,00	1,15	1,40	
Tamaño de base de datos		0,94	1,00	1,08	1,16	
Complejidad	0,70	0,85	1,00	1,15	1,30	1,65
Atributos del hardware						
Restricciones del tiempo de ejecución			1,00	1,11	1,30	1,66
Restricciones de memoria virtual			1,00	1,06	1,21	1,56

Volatilidad de la máquina virtual		0,87	1,00	1,15	1,30	
Tiempo de respuesta		0,87	1,00	1,07	1,15	
Atributos de personal						
Capacidad de análisis	1,46	1,19	1,00	0,86	0,71	
Experiencia en la aplicación	1,29	1,13	1,00	0,91	0,82	
Calidad de programadores	1,42	1,17	1,00	0,86	0,70	
Experiencia en la máquina virtual	1,21	1,10	1,00	0,90		
Experiencia en el lenguaje	1,14	1,07	1,00	0,95		
Atributos del proyecto						
Técnicas actualizadas de programación	1,24	1,10	1,00	0,91	0,82	
Utilización de herramientas de software	1,24	1,10	1,00	0,91	0,83	
Restricciones de tiempo de desarrollo	1,22	1,08	1,00	1,04	1,10	

Figura 1.5.1: Tabla atributos COCOMO

Empezamos definiendo el tipo de proyecto que tenemos entre manos, según las especificaciones que nos impone el modelo COCOMO, definiéndolo así como semi-acoplado dado a los siguientes factores:

- Es realizado por una sola persona
- Desarrollo de software con ciertas restricciones
- No se basa en experiencias anteriores del desarrollador

Tipo de proyecto	a	b	c	d
Orgánico	2.4	1.05	2.5	0.38
Semiacoplado	3.0	1.12	2.5	0.35
Empotrado	3.6	1.20	2.5	0.32

Figura 1.5.2: Tipo de proyecto COCOMO

Al elegir el tipo de proyecto semi-acoplado, los valores para las variables "a, b, c, y d" quedan como se observa en la figura 1.5.2.

Para calcular el factor de ajuste del esfuerzo escogemos:

- Para los atributos del software un valor nominal
- Para los atributos del hardware valor alto
- Para los atributos del personal valor nominal
- Para los atributos del proyecto valor alto

Multiplicando los valores obtenidos mediantes estas asignaciones de la figura 1.5.1 obtenemos un valor de EAF= 1,24.

Aplicando estos resultados las formulas anteriores obtenemos estos resultados:

- $E=3*5^{1,12}*1,24=$ **22,56 pers/mes**
- $D=2,5*22,56^{0,35}=$ **7,44 meses**
- PR=5000/22,56= **221lineas de código/persona mes**
- P=22,56/7,44= **3,03 personas**

Conclusiones

Según estos resultados se necesitan un equipo de 3 personas trabajando alrededor de 7 meses, con un promedio de 221 líneas de código cada persona al mes, pero dado que nuestro plazo era de 9 meses y partíamos de código ya realizado, el proyecto puede ser llevado a cabo por una persona en el tiempo especificado.

Dado que la mitad del trabajo, sobre unos 3 meses y medio, recaería sobre un analista de software y que el salario medio entre los profesionales pertenecientes a este sector [29] es de 2481€/mes, su salario ascendería a la suma de **8683,5€**

La otra mitad del trabajo, sobre unos 3 meses y medio, recaería sobre un programador software y dado que el salario medio entre los profesionales pertenecientes a este sector es de 2100 €/mes, su salario ascendería a la suma de **7350€**

El coste total en euros para llevar a cabo este proyecto sería de **16033,5€**

1.6 Estructura 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 un análisis de la estructura y elementos básicos de los Sistemas de Recomendación, terminando por clasificación, atendiendo a su funcionamiento, ventajas e inconvenientes. Seguidamente nos centraremos en los sistemas colaborativos que irán acompañados de algunas aplicaciones reales de sistemas de colaboración existentes en Internet.

El capítulo 3 está dedicado a las tecnologías utilizadas en el desarrollo del proyecto; empezando por los servicios, concretamente a los servicios web. Se indicarán las principales ventajas y desventajas de utilizar este tipo de servicios. Por otro lado se explicarán los patrones de diseño utilizados, centrándonos en sus cualidades y profundizando en el patrón controlador, el cual es el más usado a lo largo de este proyecto. Por último se analizará 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.

El capítulo 4 supone el eje principal de la presente memoria, al ser el de mayor extensión y estar dedicado al proceso de ingeniería del software de este proyecto. Como proyecto software que es, se hará exhaustivo estudio a las diferentes etapas de la Ingeniería de Software, para acto seguido aplicarlas en el desarrollo de nuestro sistema. Así, se definirán el análisis del sistema, incluyendo el modelo de casos de uso y el análisis de perfiles de usuario como de escenarios. Acto seguido, se pasa a la etapa de diseño que comprende desde un diagrama de clases, pasando por el diseño de datos, hasta el diseño de la interfaz. Por último, repasaremos el proceso seguido para la implementación, incluyendo las herramientas de desarrollo utilizadas para tal fin, y pruebas para verificar el cumplimiento de los requisitos funcionales de la aplicación.

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 Web.

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 [14], 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; con este propósito hoy en día se utilizan 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.

A continuación se describe el esquema básico de funcionamiento de un Sistema de Recomendación

Figura 2.1.1: Esquema básico de Sistema de Recomendación [16]

Los SR están principalmente dirigidos a individuos que carecen de la suficiente experiencia personal o del tiempo necesarios para evaluar el potencial de un enorme número de ítems diferentes que, por ejemplo, un sitio Web puede ofrecer (Véase figura 2.1.1). 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. Ejemplos típicos de estas recomendaciones no personalizadas son las listas de los n mejores, por ejemplo, los diez mejores libros, las veinte 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 habitualmente 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 [4], tales 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 [4], entre los que destacan:

- 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 mucha 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 la estructura básica de funcionamiento de los Sistemas de Recomendación y los principales elementos que los componen.

2.2 Esquema Básico de Funcionamiento y Elementos de un SR

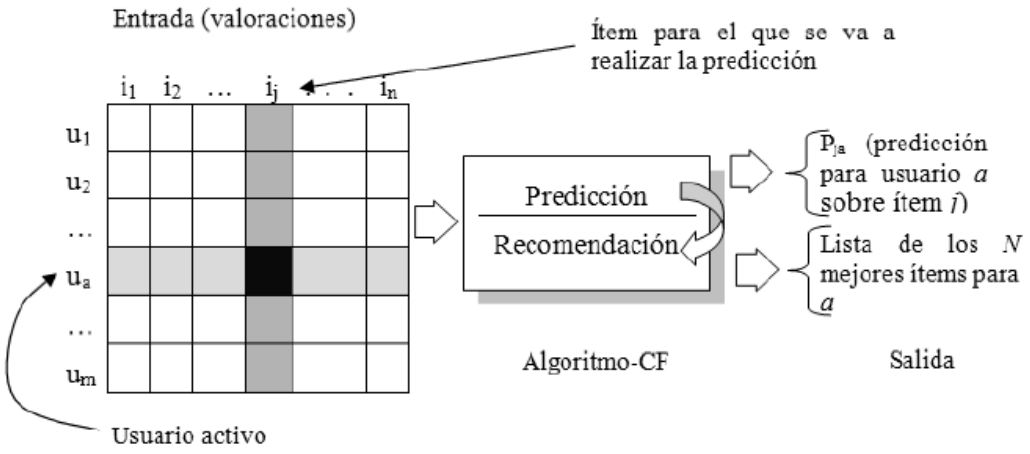

Figura 2.2.1: Esquema básico de funcionamiento de un SR

En la figura 2.2.1 se puede observar un esquema básico de un SR, el cual para funcionar necesita de los siguientes elementos:

- Base de datos: La calidad y cantidad de 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.

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.

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.

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.

Análisis Online vs. Análisis Offline

Es importante decidir si vamos a trabajar sobre los datos de manera online u offline ya que afecta a la cantidad y calidad de resultados. 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.

2.3. Clasificación de los Sistemas de Recomendación

Una vez dada una visión básica sobre 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.

2.3.1 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 recomendación basados en contenido comprende dos grandes etapas: analizar las descripciones de los productos valorados por los usuarios y predecir qué productos le pueden gustar.

2.3.2 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 la sección 2.4.

2.3.3 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].

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.

Las técnicas de razonamiento basadas 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).

2.3.4 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.

2.3.5 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 debido a la dificultad actual de obtener este tipo de información.

2.3.6 Sistemas de Recomendación Híbridos

Para solucionar las limitaciones que pueden tener las técnicas anteriormente estudiadas, se planteó como solución la hibridación de técnicas. En ella se combinan dos o más técnicas de recomendación para obtener un mejor rendimiento que utilizando una de ellas de forma independiente.

La hibridación puede solventar los problemas más comunes presentados por los sistemas como el colaborativo y otras técnicas de recomendación.

Los sistemas de recomendación híbridos que utilizan algoritmos de recomendación basados en contenido y colaborativos, independientemente de la técnica de hibridación que empleen, siempre presentarán el problema de la escalabilidad pues ambas técnicas necesitan una base de datos de valoraciones.

De todas maneras, esta hibridación es bastante utilizada, porque en muchas situaciones tales valoraciones ya existen, o porque presentan más flexibilidad y mejores resultados que si emplearan estas técnicas de forma independiente.

A continuación se exponen en profundidad los Sistemas de Recomendación Colaborativos, ya mencionados de forma breve anteriormente, dado que son un pilar fundamental de la aplicación.

Tipos de hibridación en sistemas de recomendación

En la tabla 2.3.1 podemos observar los distintos tipos de hibridación que puede utilizar para los Sistemas de Recomendación con una breve descripción de cada uno

Método de hibridación	Descripción
Mediante Pesos	Las valoraciones (o votos) de varias técnicas de recomendación son combinadas para producir una única recomendación
Conmutación	El sistema utiliza una u otra técnica dependiendo de la situación actual
Mezcla	Las recomendaciones provenientes de varias técnicas de recomendación son mostradas al mismo tiempo
Combinación de características	Las características de las fuentes de datos de varias técnicas de recomendación son combinadas en un algoritmo de recomendación único
Cascada	La salida de una técnica es utilizada como una entrada de características de otro
Aumento de características	Uno de los sistemas de recomendación refina los resultados dados por otro
Meta-nivel	El modelo aprendido por un sistema de recomendación es utilizado como entrada de otro

Tabla 2.3.1 Métodos de hibridación

2.4. Sistemas de Recomendación Colaborativos

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

Introducción y Orígenes

Los Sistemas de Recomendación Colaborativos son aquellos que realizan recomendaciones basándose únicamente en términos de similitud entre factores, 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 podría aprenderse un modelo, al igual que ocurre en la minería de datos, y una fase de recomendación (online) en la que se aplicaría 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).

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 2.4.1). 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.

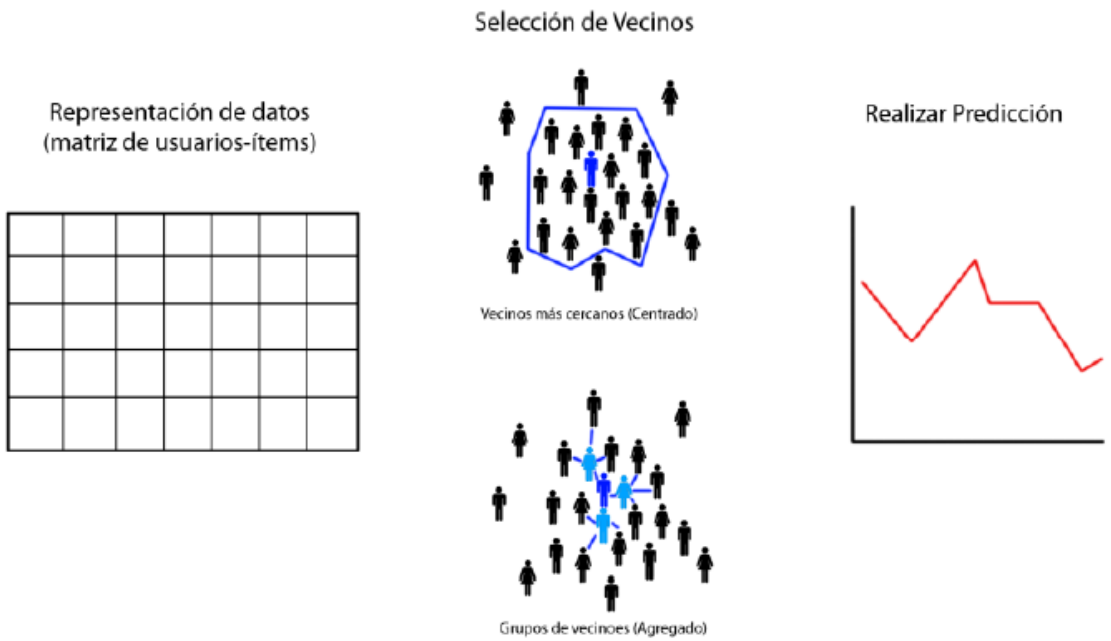

Figura 2.4.1: Las principales partes de un SR Colaborativo basado en memoria

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.

La mayoría no 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 actual con el fin de realizar una predicción para el usuario sobre 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.

- 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.

Los algoritmos basados en vecindarios examinan 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.
5. Estudio de calidad de la predicción. Se realizará mediante el uso de métricas, de nuevo existen varias métricas que permiten ver la calidad de las predicciones. En este caso proponemos el uso del Mean Absolute Error (MAE) para realizar dicho estudio.

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

Para realizar el cálculo de similitud de los ítems anteriores con respecto al ítem en cuestión necesitamos unas medidas para evaluar la similitud entre ítems; en nuestro caso hemos utilizado el Coeficiente de Correlación de Pearson.

Coeficiente de Correlación de Pearson

El Coeficiente de Correlación de Pearson (en adelante PCC – Pearson correlation coefficient) es la primera formulación estadística aparecida para el filtrado colaborativo. La correlación entre el usuario u y el usuario i se define como (véase figura 2.4.2):

$$r(u,i) = \frac{\sum_j (v_{u,j} - \overline{v_u})(v_{i,j} - \overline{v_i})}{\sqrt{\sum_j (v_{u,j} - \overline{v_u})^2 \sum_j (v_{i,j} - \overline{v_i})^2}}$$

PCC

siendo $v_{u,j}$ la valoración del usuario *u* sobre el item *j*, $v_{i,j}$ idem para usuario *i* e item *j*, y $\overline{v_u}$ y $\overline{v_i}$ los valores medios para los usuarios *u* e *i* respectivamente.

Figura 2.4.2: Correlación entre usuarios

Una vez calculada la similitud lo siguiente que necesitamos seleccionar de los k ítems más cercanos a este; para ello utilizamos el algoritmo KNN.

Algoritmo KNN

Este algoritmo trabaja directamente sobre los datos, sin construir ningún tipo de modelo, están basados en aprendizaje por analogía (véase figura 2.4.3). Dado un nuevo objeto a clasificar, el algoritmo se encarga de buscar los k objetos más cercanos al mismo, según cierta función de similitud. Una vez encontrados, la clasificación se realiza teniendo en cuenta las clases a las que dichos objetos pertenezcan.

Funcionamiento del algoritmo K-NN

Dado un conjunto de objetos $A=\{a_1,...,a_n\}$ cada uno de los cuales pertenece a cierta clase c_l del conjunto $C=\{c_1,...,c_m\}$, se tienen las siguientes funciones:

1. Función similitud $s: A \rightarrow R^+U\{0\}$
2. Función de pertenencia $f: A \rightarrow C$
3. Función δ definida como $\delta(a,b)=1$ si $a=b$ y $\delta(a,b)=0$ si $a\neq b$.

Dado un objeto x, el proceso de clasificación consiste en los siguientes pasos:

1. Se obtienen los k objetos de A más similares a x, es decir, $A_x=\{a_j^x,...,a_k^x\}$ tal que para todo objeto x' que no pertenece a A_x se cumple que $s(x,x')\leq min\ s(x,a_j^x)$, $j=1,...,k$.
2. Se clasifica el objeto de una de las dos maneras siguientes:
   a) Voto por la mayoría

$$\hat{f}(x) = \arg\max_{c\in C} \sum_{i=1}^{k} \delta\left(c, f\left(a_i^x\right)\right)$$

   b) Voto ponderado según la similitud

$$\hat{f}(x) = \arg\max_{c\in C} \sum_{i=1}^{k} s\left(x, a_i^x\right)\delta\left(c, f\left(a_i^x\right)\right)$$

Figura 2.4.3: Funcionamiento algoritmo KNN

Por ultimo necesitamos realizar el cálculo de la predicción del usuario activo sobre el ítem dado. Para ello utilizamos un algoritmo de predicción que se ajuste a nuestro problema como Weighted Sum.

Weighted Sum

La suma ponderada (en adelante WS – weighted sum) supone que todos los usuarios puntúan siguiendo aproximadamente la misma distribución. Se suele utilizar un multiplicador m como factor de normalización, y generalmente se utiliza $m=1/\Sigma_i\ |s_{u,i}|$, es decir, el inverso de la suma de similitudes. Además de la suma ponderada se puede utilizar también una suma media ajustada (en adelante WA – weighted adjusted sum) que pretende evitar un problema que presenta WS, y que consiste en no tomar en consideración el hecho de que usuarios diferentes puedan usar escalas diferentes de puntuación, es decir, diferentes usuarios pueden puntuar siempre o muy alto, o muy bajo, o siempre en los extremos, o sólo términos medios. Para ello en WA se utilizan desviaciones de la media para el usuario correspondiente (véase figura 2.4.4).

$v_{u,i} = \frac{\sum_j s_{u,j} v_{j,i}}{\sum_j s_{u,j}}$	$v_{u,i} = \frac{\sum_k s_{i,k} v_{u,k}}{\sum_k s_{i,k}}$	$v_{u,i} = \overline{v_u} + \frac{\sum_j s_{j,u} (v_{j,i} - \overline{v_j})}{\sum_j s_{j,u}}$	$v_{u,i} = \overline{v_u} + \frac{\sum_k s_{k,i} (v_{u,i} - \overline{v_k})}{\sum_k s_{k,i}}$
(WS user-user)	(WS item-item)	(WA- user-user)	(WA- item-item)

Figura 2.4.4: Desviaciones de la media WA

Así podremos obtener predicciones para los ítems deseados no puntuados por el usuario. El usuario activo proporciona al motor de predicción una serie de ítems para los que desea obtener predicciones y el motor de predicción le devuelve una lista de predicciones para esos ítems.

El estudio de calidad de la predicción se realizará mediante el uso de métricas; en este caso proponemos el uso del Mean Absolute Error (MAE) para realizar dicho estudio.

Métricas de precisión: MAE

El error absoluto medio (en adelante MAE - Mean Absolute Error) está considerado como una medida estadística para la estimación de la exactitud con la que el sistema realizará las predicciones. Este tipo de medidas tratan de verificar con qué grado de exactitud el sistema predice las valoraciones con respecto a las verdaderas valoraciones del usuario.

En nuestro dominio particular, esta medida nos va a decir cuán lejos están las estimaciones realizadas por el sistema de los datos reales. Visto de otro modo, el MAE nos va a decir la habilidad que nuestro algoritmo de filtrado colaborativo va a presentar a la hora de realizar predicciones.

No debemos perder de vista que esta medida, como su propio nombre indica, es una media del error que se produce, lo que quiere decir que aunque el valor de ese error absoluto medio sea bajo, pueden estimaciones individuales que sean totalmente erróneas.

$$MAE = |\overline{E}| = \frac{\sum_{i=1}^{n} |p_i - r_i|}{P}$$

(MAE)

Figura 2.4.5: Formula MAE

Siendo P el número total de predicciones realizadas, pi el valor de una predicción y r_i el valor real que corresponde a dicha predicción (véase figura 2.4.5). Cuanto menor sea el error absoluto medio, mayor será la exactitud con la que el sistema realizará las predicciones.

A pesar de las ventajas que presentan los Sistemas de Recomendación Colaborativos, también presentan una serie de problemas.

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.

2.5 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 2.5.1, 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 2.5.1: dominios para las aplicaciones SR más comunes

A continuación se exponen algunas aplicaciones reales de distintos sistemas de recomendación a modo de ilustración.

Aplicaciones 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

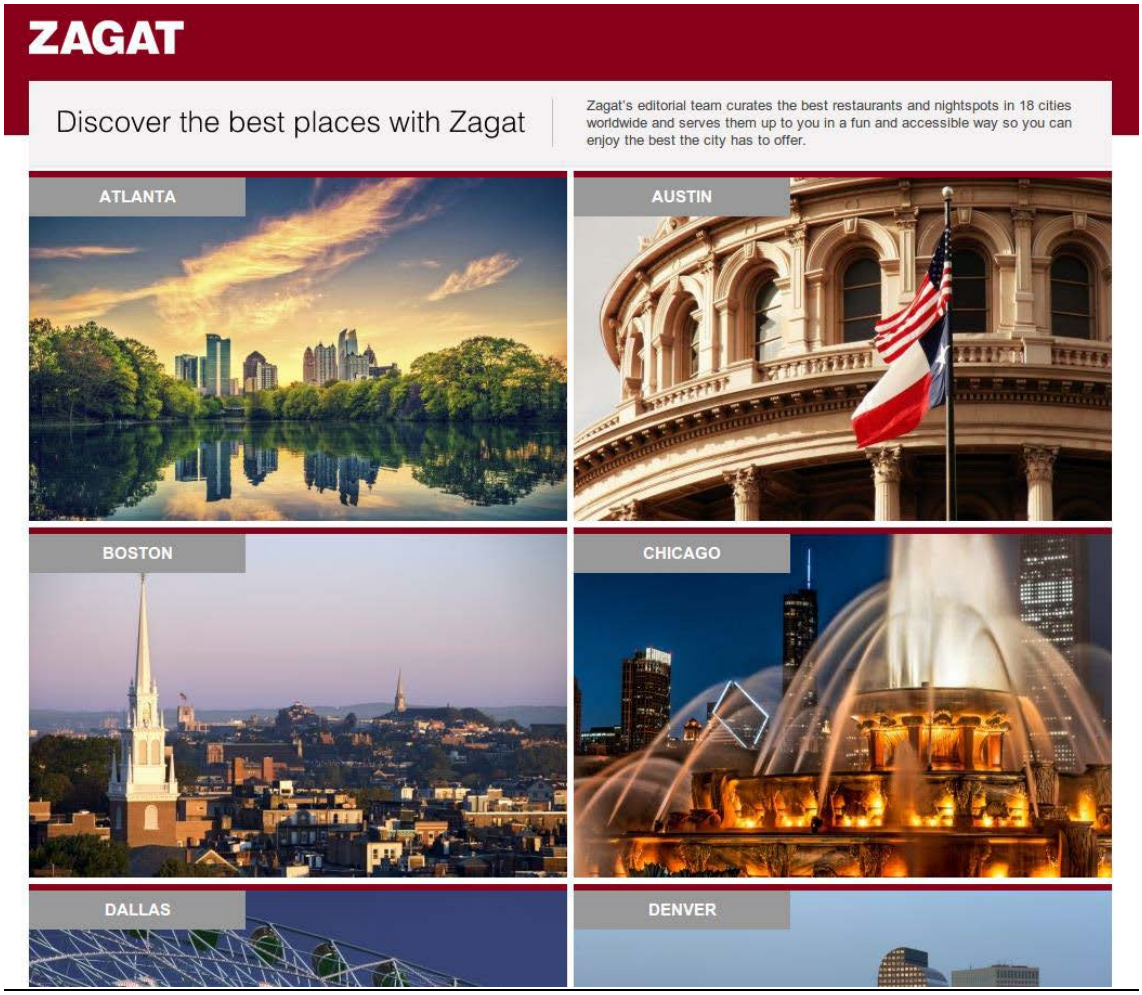

Figura 2.5.1: Portal de Zagat.com

Zagat Survey [23] 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 2.5.1) y de la aplicación Android (Véase figura 2.5.2).

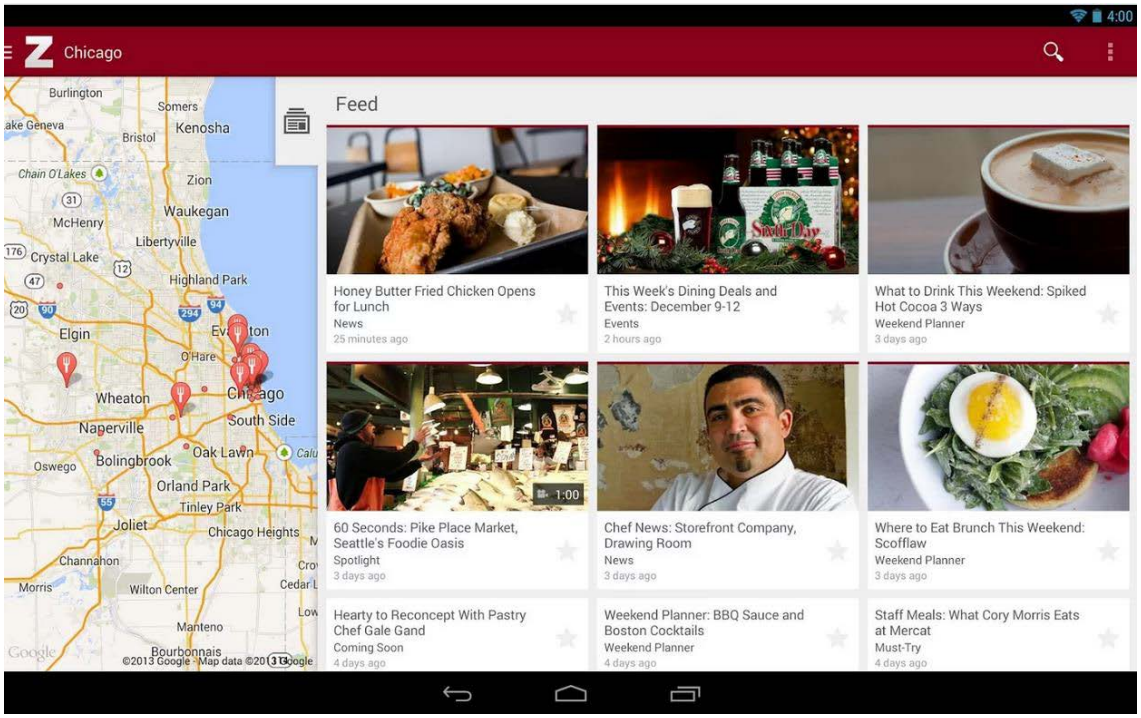

Figura 2.5.2: Aplicación Android de Zagat.com

Foursquare

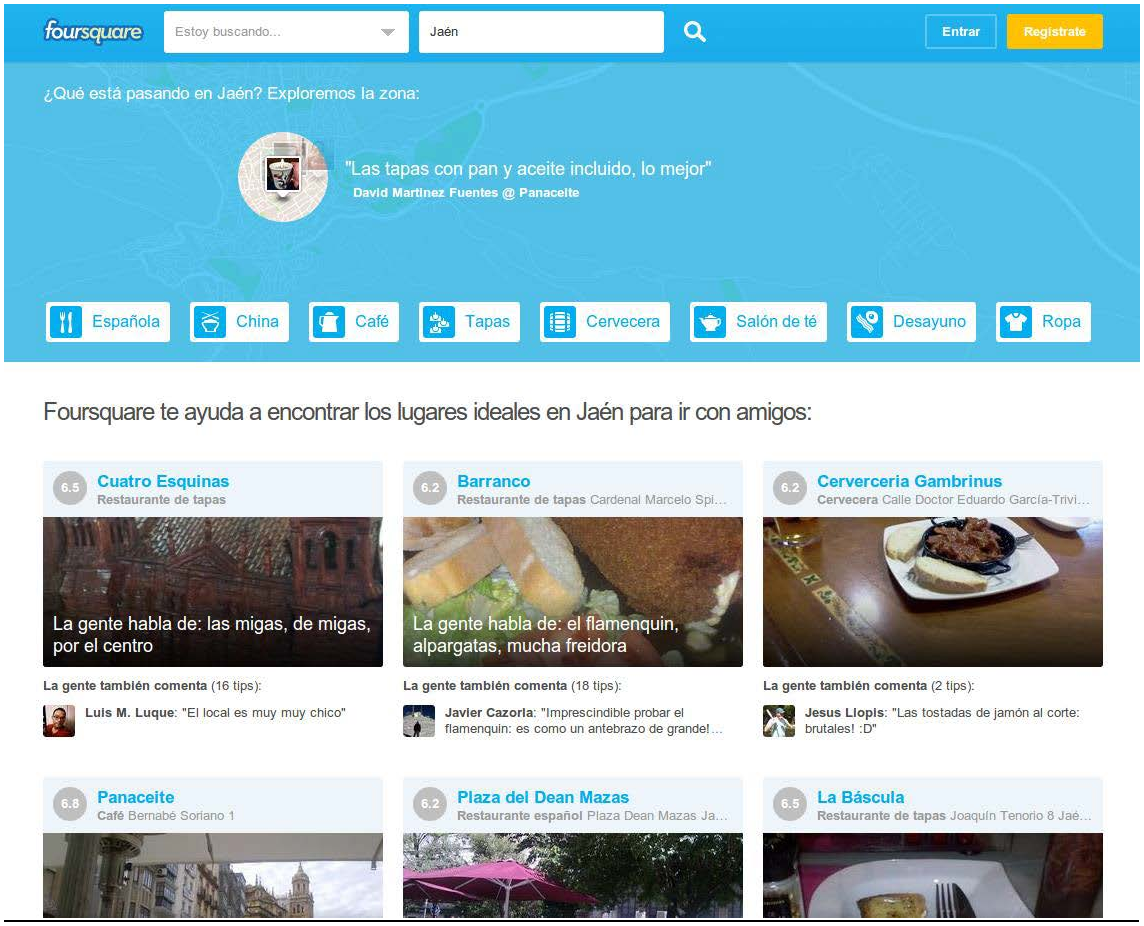

Figura 2.5.3: Portal de Foursquare.com

Los fundadores de Foursquare [24] lo definen como una red social basada en servicios de localización que incorpora elementos de juego (Véase figura 2.5.3).

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.

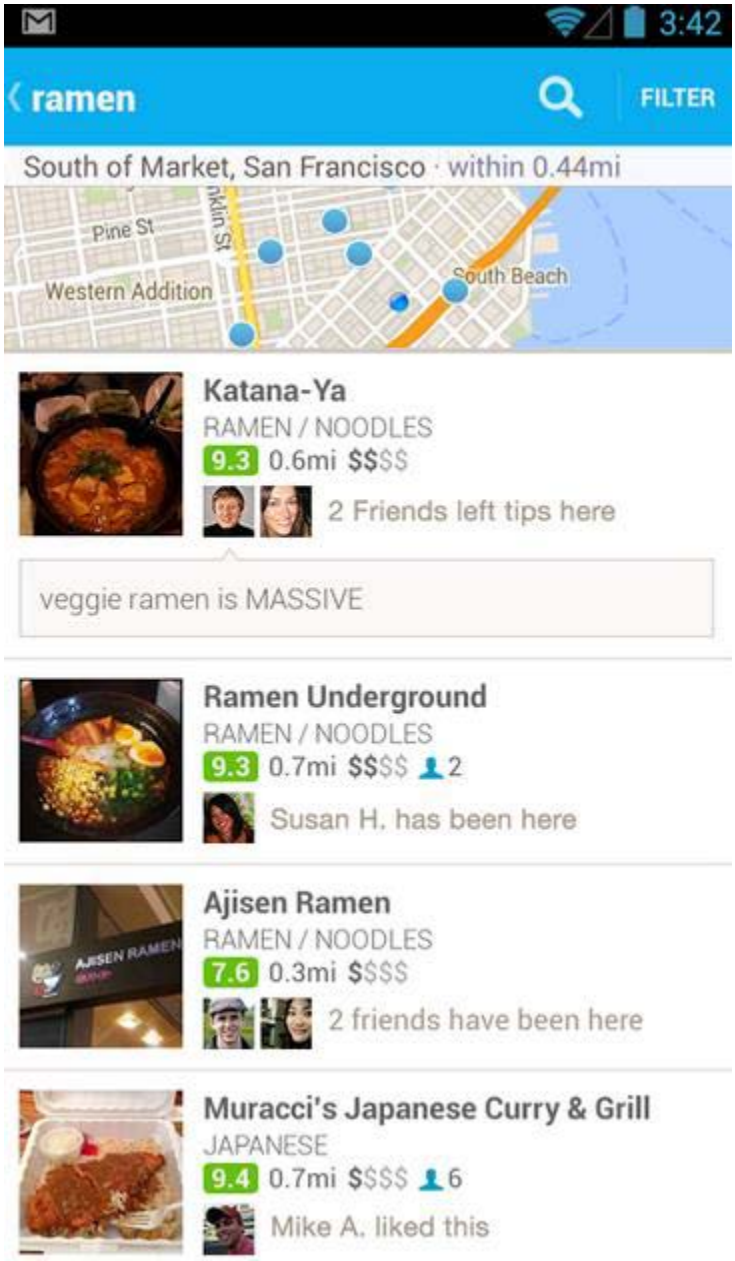

Figura 2.5.4: Aplicación Android de Foursquare.com

Desde la versión 3.0 de la aplicación Android (Véase figura 2.5.4) 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

Arquitectura y servicios

3.1 Introducción

A lo largo de este capítulo se hará una introducción a los servicios web, ya que de ellos se nutre esta aplicación para la gestión de usuarios y preferencias. También analizaremos la tecnología REST ya que será la utilizada en el proyecto, tanto para consumo como para ofrecimiento de recursos, y se indicarán las principales ventajas y desventajas de utilizar este tipo de servicios.

Por otro lado se definirán los patrones de diseño, para a continuación poder centrarnos en sus cualidades y profundizar en el patrón controlador, el cual es el más usado a lo largo de este proyecto.

Por último se va a analizar el sistema operativo Android, incluyendo una descripción de las características más destacadas que hemos de conocer acerca del mismo ya que también se soporta sobre nuestros servicios.

3.2 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 [19]:

"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:

"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 [20]:

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

A continuación procedemos a explicar los servicios REST; ya que son los utilizados para la esta aplicación web en cuestión.

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 [31] 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 tener que preocuparse sobre cómo debe responder el servidor a la pérdida de la sesión del cliente por pérdida de conectividad.

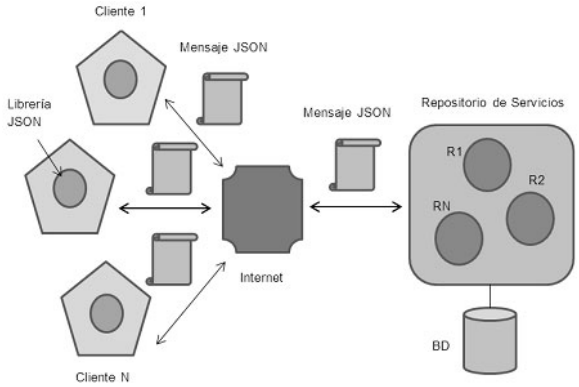

Figura 3.2.1: Esquema funcionamiento Servicios REST

- 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" genérica para las notificaciones.
- Generalmente fácil de construir y adoptar.

Y sus principales 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 ya que REST se apoya totalmente en el estándar HTTP. Posteriormente, se analizará cómo se estructuran las URLs en REST.

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 nuevo recurso
PUT	Actualizar un recurso existente
DELETE	Eliminar un recurso

Tabla 3.2.1: 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 3.2.2: Códigos de Respuesta HTTP

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).

A continuación se introducen los patrones de diseño centrándonos en el patrón controlador, el cual, es el más utilizado a lo largo de nuestra aplicación para el tratamiento de eventos.

3.3 Patrones de diseño

Los patrones de diseño [32] ayudan a los diseñadores a reutilizar con éxito diseños para obtener nuevos diseños dado que los diseñadores expertos no resuelven cada problema desde el principio.

El objetivo de los patrones es guardar la experiencia en diseños de programas orientados a objetos.

Los patrones de diseño hacen más fácil reutilizar con éxito los diseños y arquitecturas, ayudan a elegir entre diseños alternativos, hacen a un sistema reutilizable y evitan alternativas que comprometen la reutilización.

Los patrones de diseño hablan de cómo construir software, de cómo utilizar las clases y los objetos de forma conocida.

Definición

Un patrón de diseño es una solución a un problema general, utilizable para resolver un problema particular ajustando sus atributos. La solución no es determinista sino que ofrece un modelo de actuación que quien lo aplica debe adaptar a la situación concreta en la que se encuentra.

Un patrón se define a través de cuatro elementos:

- Nombre descriptivo que permita una referencia rápida del patrón.
- Problema que aborda y que indica cuando es interesante aplicar el patrón.
- Solución propuesta, que no describe una implementación concreta sino más bien una plantilla o guía abstracta de resolución en forma de disposición general de elementos de software.
- Consecuencias, ventajas e inconvenientes de su aplicación.

Cualidades de un patrón de diseño

- Encapsulamiento y abstracción: Cada patrón encapsula un problema bien definido y su solución en un dominio particular.
- Extensión y variabilidad: Cada patrón debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solucionar un gran problema.
- Generatividad y composición: Cada patrón una vez aplicado genera un contexto resultante, el que concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo.
- Equilibrio: Cada patrón debe realizar algún tipo de balance entre sus efectos y restricciones.

Patrón Controlador

Este patrón identifica por completo la aplicación, ocupándose de todos los eventos externos que ocupan a esta y representando un manejador artificial de eventos.

Esta estructura es capaz de soportar requisitos de usuario que se presentan mediante distintos estilos de interfaz y mejora el mantenimiento y la portabilidad.

Se basa en el principio de Separación Modelo-Vista. Los objetos del modelo (dominio) no deberían conocer directamente a los objetos de la vista (presentación), al menos como objetos de la vista; así por ejemplo, un objeto del modelo no debería enviar un mensaje directamente a un objeto ventana de la interfaz de usuario, pidiéndole que muestre algo, cambie de color, se cierre, etc.

Adicionalmente, este principio significa que las clases del dominio encapsulan la información y el comportamiento relacionado con la lógica de la aplicación.

Las clases de las ventanas son relativamente ligeras; es decir, son responsables de la entrada y salida y capturar los eventos de la interfaz, pero no mantienen datos ni proporcionan directamente ninguna funcionalidad de la aplicación.

La funcionalidad disponible para cualquier usuario debería adaptarse a sus necesidades, puesto que su visión del sistema puede ser distinta en función del tipo de usuario. Tiene sentido que cada usuario solamente tenga acceso a la parte pertinente de la funcionalidad del sistema. Esto se puede resolver utilizando una arquitectura que disponga de múltiples interfaces para la misma funcionalidad de núcleo:

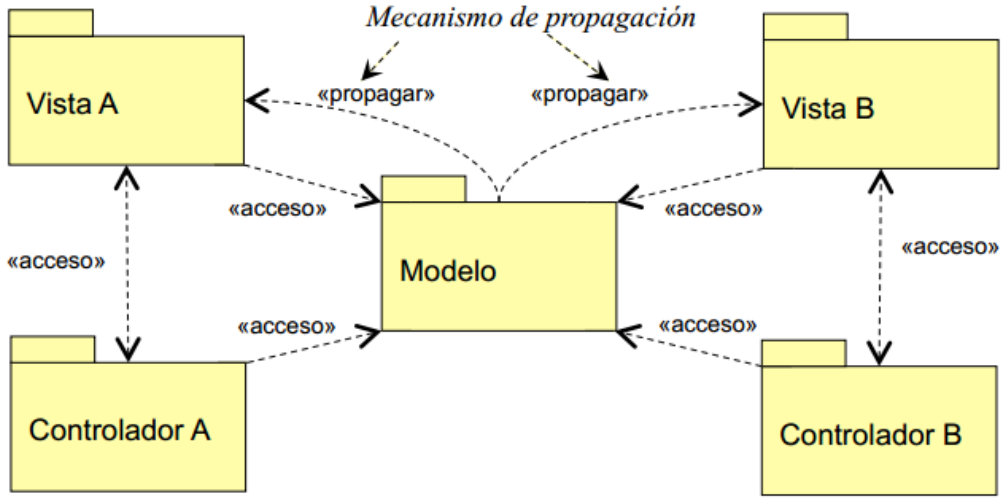

Figura 3.3.1: ejemplo patrón controlador en arquitectura MVC

Como se observa en la figura 3.3.1, las responsabilidades de los componentes de la arquitectura MVC [18] se distribuyen de la siguiente forma:

- Modelo: Aporta la funcionalidad central de la aplicación.
- Vista: Corresponde a un estilo y formato particular de presentación de información al usuario. La vista recupera datos del.
- Controlador: Acepta entradas del usuario en forma de eventos que disparan la ejecución de operaciones dentro del modelo. Esto puede causar cambios en la información y a cambio el disparador actualiza en todas las vistas asegurando que todos están actualizados.
- Mecanismo de propagación: Permite al modelo informar a cada vista de que los datos del modelo han cambiado y como resultado la vista debe actualizarse a sí misma. También suele denominarse mecanismo de dependencia.

El mecanismo de propagación, diferente para cada aplicación, también es gestionado por el controlador. Manejando la transferencia de datos entre la lógica de aplicación y las vistas de usuario.

Ventajas y desventajas del uso del patrón

Se tienen muchas ventajas como:

- La implementación se realiza de forma modular.
- Sus vistas muestran información actualizada siempre. El programador no debe preocuparse de solicitar que las vistas se actualicen, ya que este proceso es realizado automáticamente por el controlador de la aplicación.
- Cualquier modificación que afecte al dominio, como aumentar métodos o datos contenidos, implica una modificación sólo en el modelo y las interfaces del mismo con las vistas, no todo el mecanismo de comunicación y de actualización entre modelos.
- Las modificaciones a las vistas no afectan al modelo de dominio, simplemente se modifica la representación de la información, no su tratamiento.

Como desventajas tenemos:

- Para desarrollar una aplicación bajo el patrón de diseño controlador, es necesario una mayor dedicación en los tiempos iniciales del desarrollo. Normalmente el patrón exige al programador desarrollar un mayor número de clases que, en otros entornos de desarrollo, no son necesarias. Sin embargo, esta desventaja es muy relativa ya que posteriormente, en la etapa de mantenimiento de la aplicación, una aplicación con este patrón es mucho más simple, extensible y modificable que una aplicación que no lo implementa.
- Este patrón requiere la existencia de una arquitectura inicial sobre la que se deben construir clases e interfaces para modificar y comunicar los módulos de una aplicación. Esta arquitectura inicial debe incluir, por lo menos, un mecanismo de eventos para poder proporcionar las notificaciones que genera el modelo de aplicación; una clase Modelo, otra clase Vista y una clase Controlador genéricas que realicen todas las tareas de comunicación, notificación y actualización que serán luego transparentes para el desarrollo de la aplicación.
- Controlador es un patrón de diseño orientado a objetos por lo que su implementación es sumamente costosa y difícil en lenguajes que no siguen este paradigma.

A continuación se profundiza en la parte de Android que es necesario comprender para la correcta consecución de nuestra aplicación; que es la reproducción en Streaming.

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 [30]:

"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 3.4.1). 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.

Hablamos de 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.

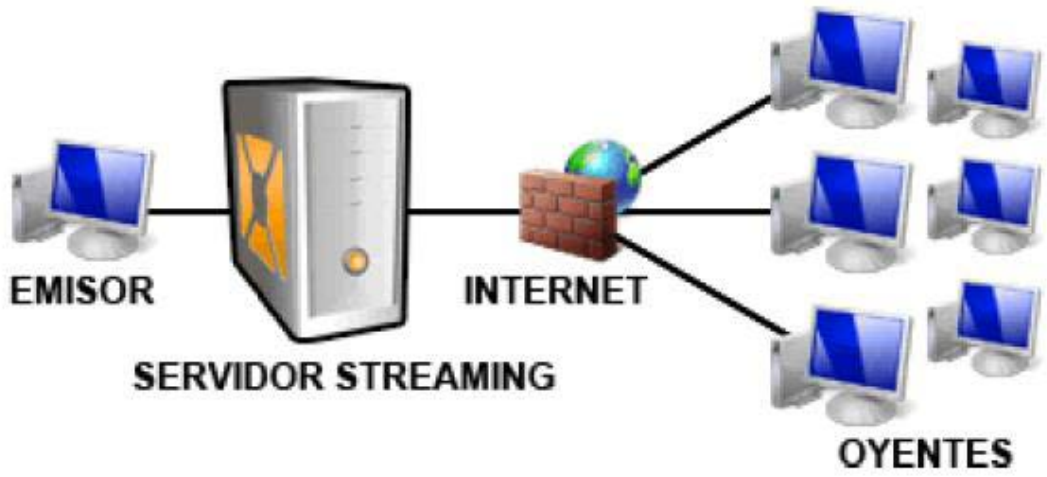

Figura 3.4.1: Ejemplo de estructura de audio-Streaming en directo

A continuación se muestran aplicaciones reales en las que se lleva a cabo el uso de Streaming para aplicaciones Android.

Aplicaciones de música Streaming en Android

- Last.fm

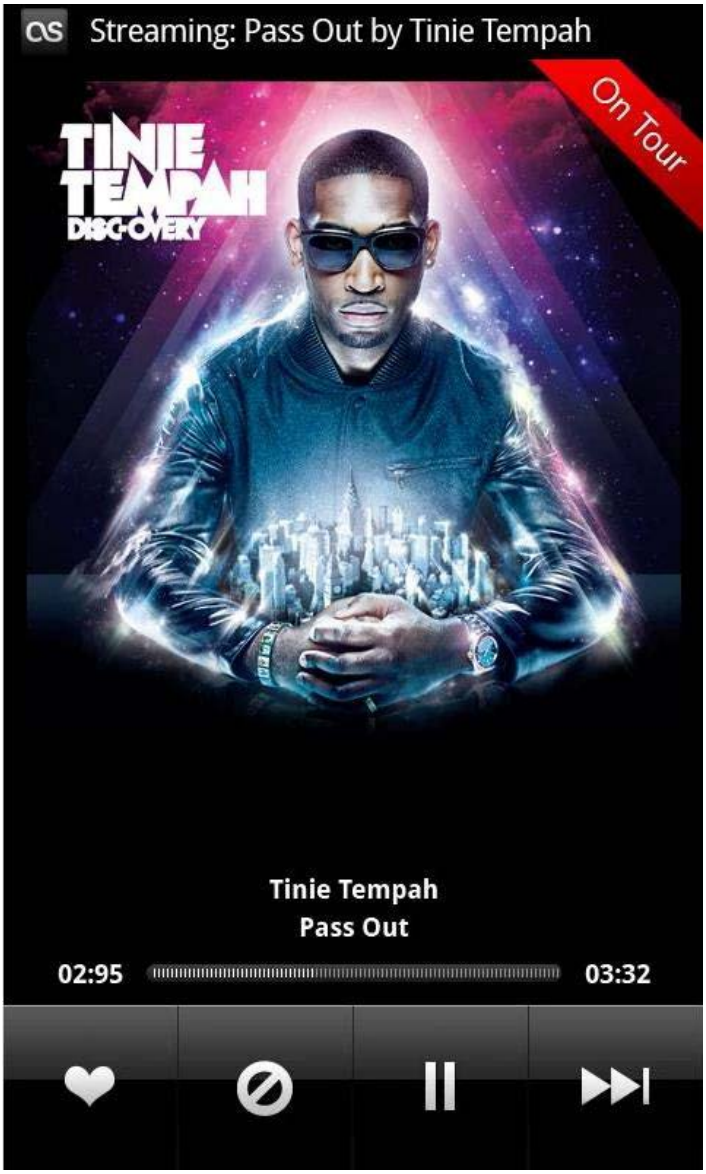

Figura 3.4.2: Aplicación Android de Last.fm

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 3.4.2) tiene entre 1.000.000 y 5.000.000 de descargas en Google Play.

- Pandora

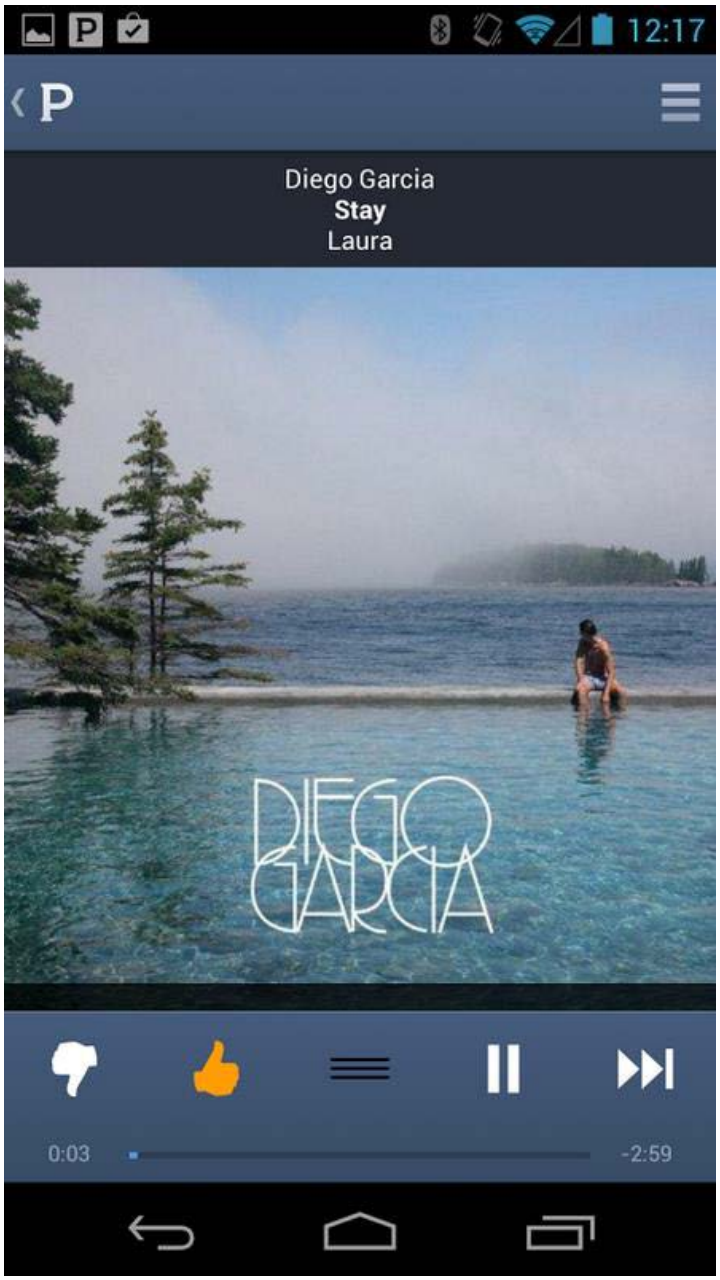

Figura 3.4.3: Aplicación Android de Pandora

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 3.4.3) tiene entre 100.000.000 y 500.000.000 de descargas en Google Play estando disponible solamente en Estados Unidos.

Capítulo 4

Proceso de ingeniería software

4.1 Introducción

En este capítulo se expone todo el proceso de ingeniería software para desarrollar este proyecto, partiendo del análisis del sistema, pasando por el diseño, tanto de sistema como de datos e interfaz, para finalmente terminar con la implementación de la misma y las pruebas necesarias para verificar la calidad del software obtenido a través del cumplimiento de sus requisitos obtenidos en la primera fase.

La aplicación web desarrollada en este TFG convive con una aplicación Android del prototipo presentado en [28] ampliando su funcionalidad a través de la implementación de unos servicios de gestión REST. Estos servicios así como una base de datos, comunes para ambas aplicaciones, están alojados en el servidor sinbad2 de la Universidad de Jaén.

En primer lugar se procede a exponer el análisis del sistema. En esta exposición se presentan los requisitos funcionales necesarios para que la aplicación posea toda la funcionalidad especificada, amén de los perfiles de usuario necesarios para la gestión de la aplicación y escenarios que muestran ejemplos de uso real de la aplicación.

En segundo lugar se procede a exponer el diseño. Para explicar fielmente el diseño del sistema se muestra un diagrama de clases de diseño que expresa la funcionalidad del sistema a través de sus componentes. Por último se muestra un diagrama de paquetes utilizado.

En última instancia se explican la arquitectura elegida (MVC) para la implementación así como el framework utilizado para su desarrollo; terminando por exponer las pruebas de caja utilizadas para verificar el correcto funcionamiento de la aplicación.

4.2 Análisis del sistema

Empezaremos por la enumeración de los requisitos funcionales que debe cumplir la aplicación según las especificaciones obtenidas por parte del cliente.

Requisitos funcionales

RF01 Gestionar usuario: El usuario podrá realizar todas las acciones referidas a su cuenta de usuario.

RF02 Darse de baja: El sistema debe ser capaz de ofrecer al usuario una forma de deshabilitar su cuenta de nuestro propio sistema.

RF03 Login: Un usuario que ya esté registrado en el sistema debe poder acceder a su perfil mediante un usuario y contraseña.

RF04 Cerrar sesión: El sistema debe ser capaz de ofrecer al usuario una forma de cerrar su sesión actual.

RF05 Radios: El sistema debe ser capaz de proporcionar un mecanismo para que el usuario elija la modalidad de radio a reproducir.

RF06 Reproductor: El usuario podrá detener y alterar el orden de reproducción.

RF07 Radio personalizada: El sistema debe ser capaz de ofrecer recomendaciones a los usuarios y permitir al usuario acceder a una lista de reproducción.

RF08 Radio normal: El sistema debe ser capaz de ofrecer una serie de canciones aleatorias y permitir al usuario acceder a una lista de reproducción.

RF09 Radio canciones favoritas: El sistema debe ser capaz de ofrecer una serie de canciones favoritas del usuario y permitir al usuario acceder a una lista de reproducción.

RF10 Radio artistas favoritos: El sistema debe ser capaz de ofrecer una serie de canciones de sus artistas favoritos y permitir al usuario acceder a una lista de reproducción.

RF11 Gestión musical: El sistema debe ser capaz de proporcionar un mecanismo para que el usuario seleccione sus preferencias musicales.

RF12 Puntuar canciones: El sistema ha de permitir al usuario evaluar canciones, ya sean canciones nuevas para él, o bien canciones ya evaluadas cuya puntuación desee modificar.

RF13 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.

A continuación se indican los perfiles de usuario necesarios para el uso y gestión de la aplicación.

Perfiles de Usuario

Para el manejo de la aplicación web que estamos analizando solo es necesario el uso de un perfil genérico; perfil usuario, dado que desde el cliente web no se puede modificar información sensible de nuestro sistema ya que accede a servicios virtualizados que están diseñados para una función concreta; todos para el aprovechamiento del usuario final.

El perfil de usuario base se muestra a continuación (véase la figura 4.2.1)

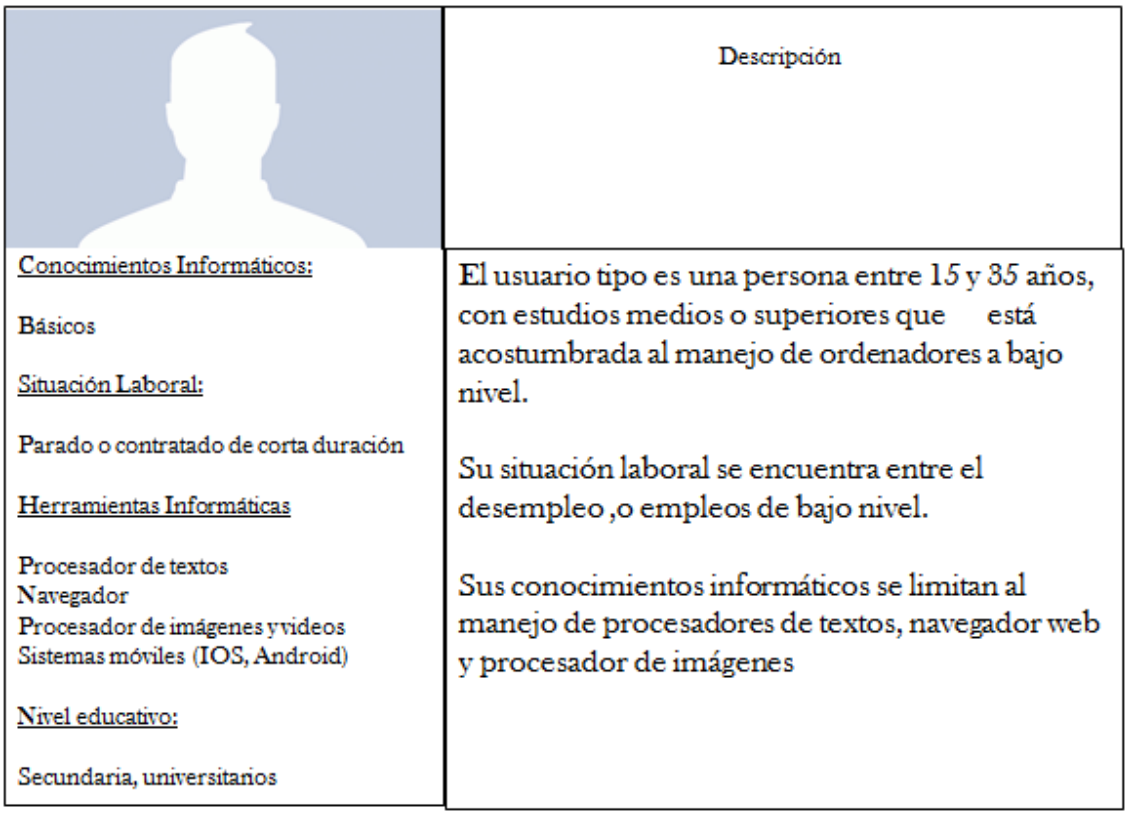

Figura 4.2.1: Perfil de usuario

A continuación se exponen los casos de uso. Un caso de uso representa una clase de funcionalidad dada por el sistema como un flujo de eventos.

Casos de uso

En ingeniería del software, un caso de uso es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada caso de uso proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico.

Diagrama Frontera

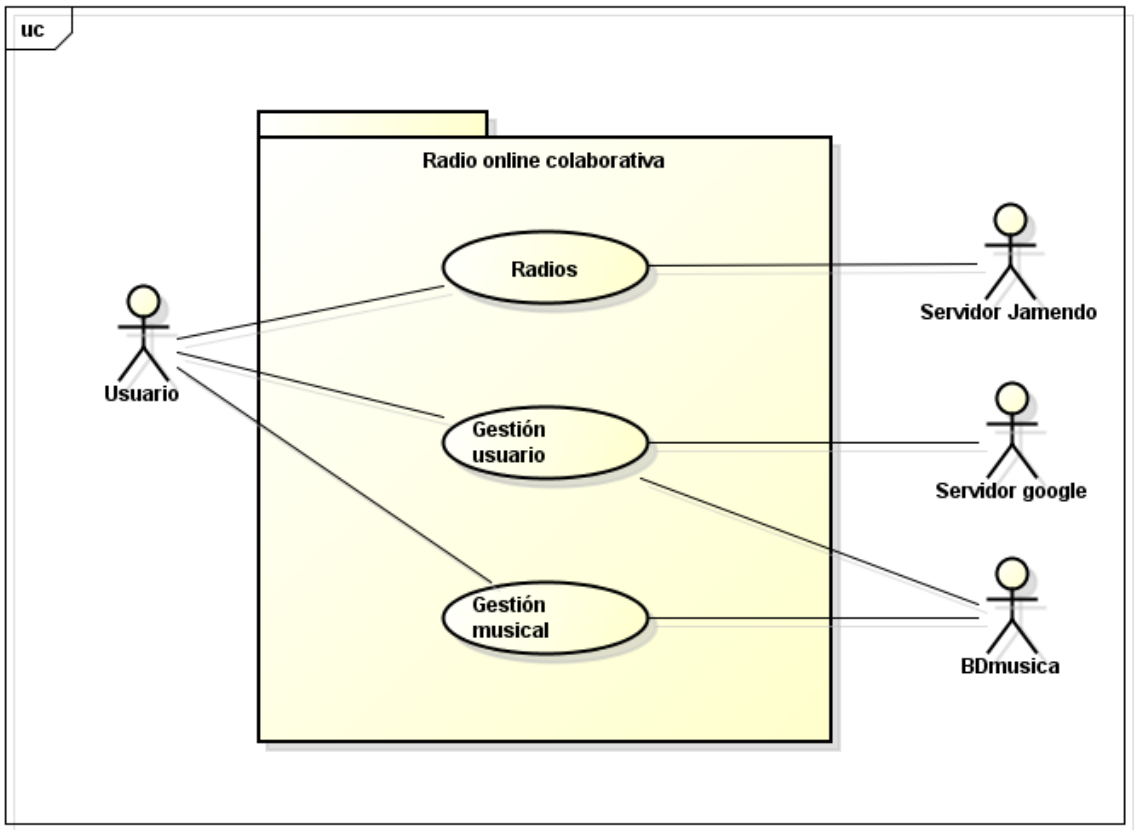

Figura 4.2.2: Diagrama frontera

Un diagrama frontera es aquel en que se muestran los límites del sistema ofreciendo una visión general de este; y de como este se relaciona con los actores y otros sistemas.

Se puede observar en este diagrama concreto (véase figura 4.2.2) en que partes se divide el sistema y cuáles de ellas se relacionan a su vez con usuarios y otros sistemas.

Gestión usuario

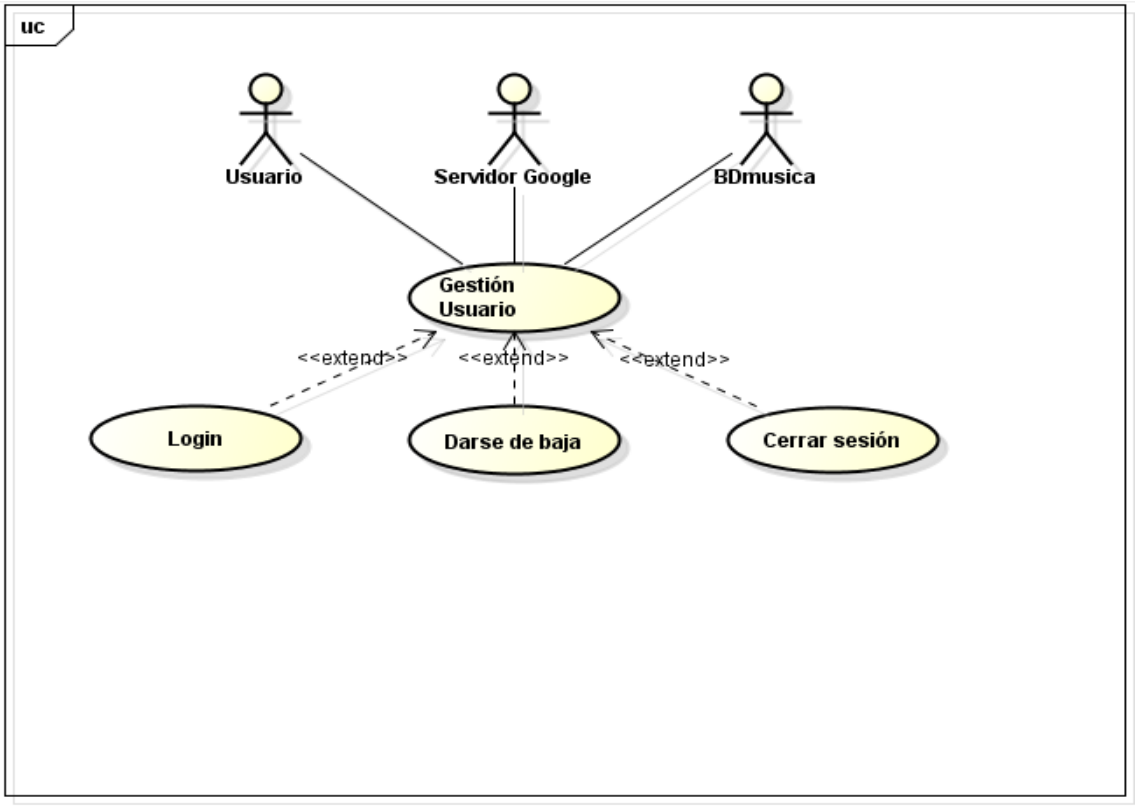

Figura 4.2.3: Caso de uso Gestión de usuario

CU-05 Gestión Usuario		
Dependencias	• CU-01 Entrada al sistema	
Precondición	-El usuario ha accedido al sistema -Conexión con el servidor pre establecida	
Descripción	El sistema debe proporcionar todos los servicios descritos en el caso de uso	
Secuencia normal	Paso	Acción
	1	El usuario se logea
	2	El servidor devuelve sus credenciales
	3	Si sus credenciales no existen se guardan en BBDD
Postcondición	El usuario es guardado en base de datos	
Excepciones	Paso	Acción
	3	No se puede establecer conexión con la BBDD
Comentarios	Puede crearse un usuario y acceder a los servicios con él o bien eliminarlo o cerrar sesión	

Gestión reproducción

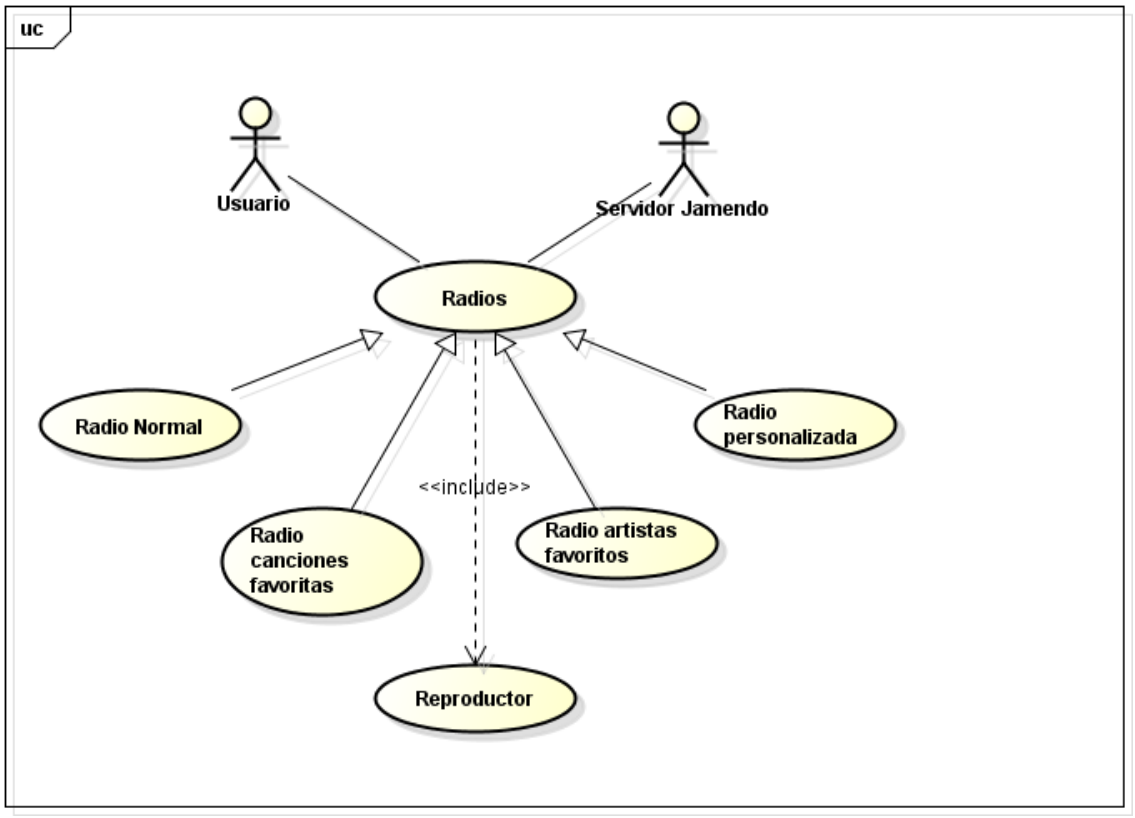

Figura 4.2.4: Caso de uso Gestión Reproducción

CU-02 Gestión Reproducción		
Dependencias	• CU-01 Entrada al sistema	
Precondición	-Conexión con el servidor pre establecida -El usuario ha accedido al sistema	
Descripción	El sistema debe proporcionar todos los servicios descritos en el caso de uso	
Secuencia normal	Paso	Acción
	1	El servicio radio de reproducción llama reproductor
	2	Reproductor hace una consulta a la BBDD
Postcondición	La reproducción se está ejecutando	
Excepciones	Paso	Acción
	2	No se puede establecer conexión con la BBDD
Comentarios	El usuario decide cambiar al tipo de radio que considere oportuna en cada momento; así como su flujo de reproducción	

Gestión musical

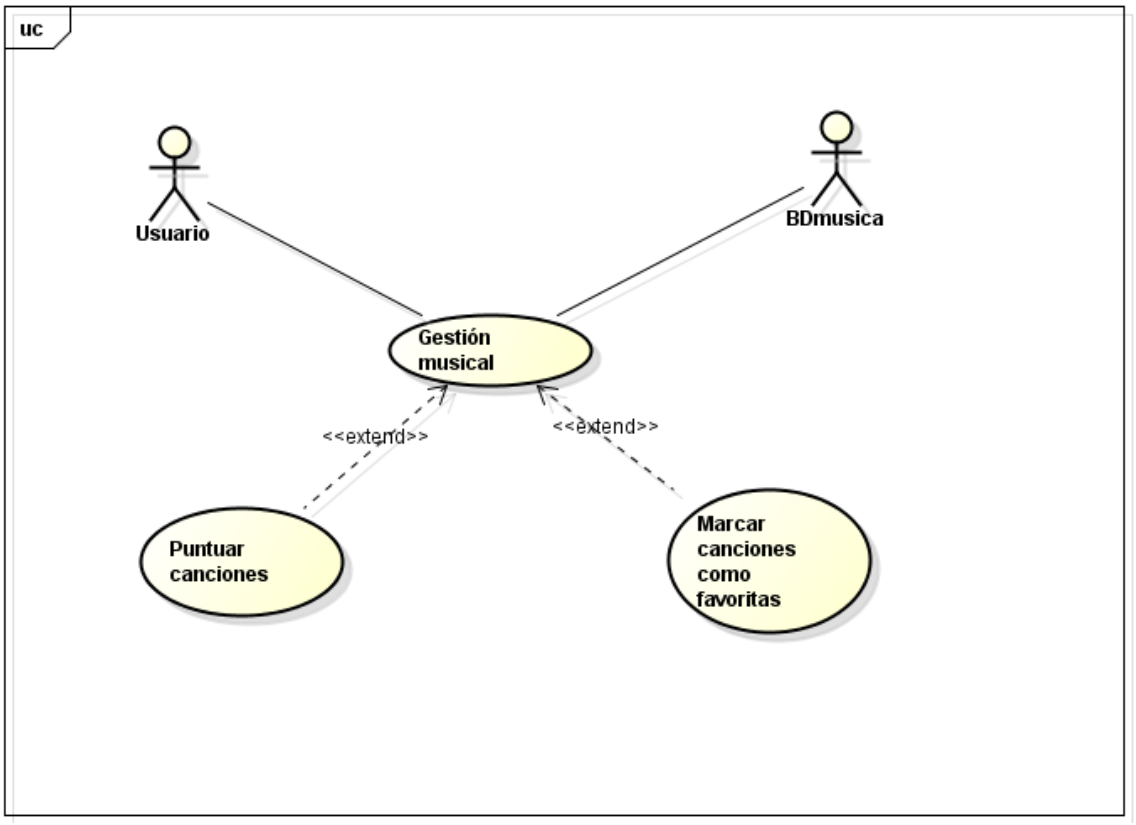

Figura 4.2.5: Caso de uso Gestión Musical

CU-04 Gestión musical		
Dependencias	• CU-01 Entrada al sistema • CU-03 Login en el sistema	
Precondición	-El usuario ha entrado en la pantalla principal -Conexión con el servidor pre establecida	
Descripción	El sistema debe proporcionar todos los servicios descritos en el caso de uso	
Secuencia normal	Paso	Acción
	1	El usuario hace su acción
	2	Las acciones son guardadas en la base de datos
Postcondición	Las selecciones son almacenadas	
Excepciones	Paso	Acción
	2	No se puede establecer conexión con el servidor
Comentarios	El usuario selecciona las recomendaciones que le gustan y las añade a su lista	

En estos casos de uso analizamos la relación existente entre los componentes software de la aplicación. Con las tablas podemos deducirla secuencia de interacciones que se desarrollarán entre el sistema y el actor en respuesta a un evento que inicia el actor principal sobre el propio sistema

Diagramas de secuencia

Un diagrama de secuencia muestra la interacción de un conjunto de objetos en una aplicación a través del tiempo y se pueden modelar para cada caso de uso. Mientras que el diagrama de casos de uso permite el modelado de una vista del sistema, el diagrama de secuencia contiene detalles de implementación del sistema, incluyendo los objetos y clases que se usan para implementar el escenario y mensajes intercambiados entre los objetos (véase figura 4.2.6).

Radio personalizada

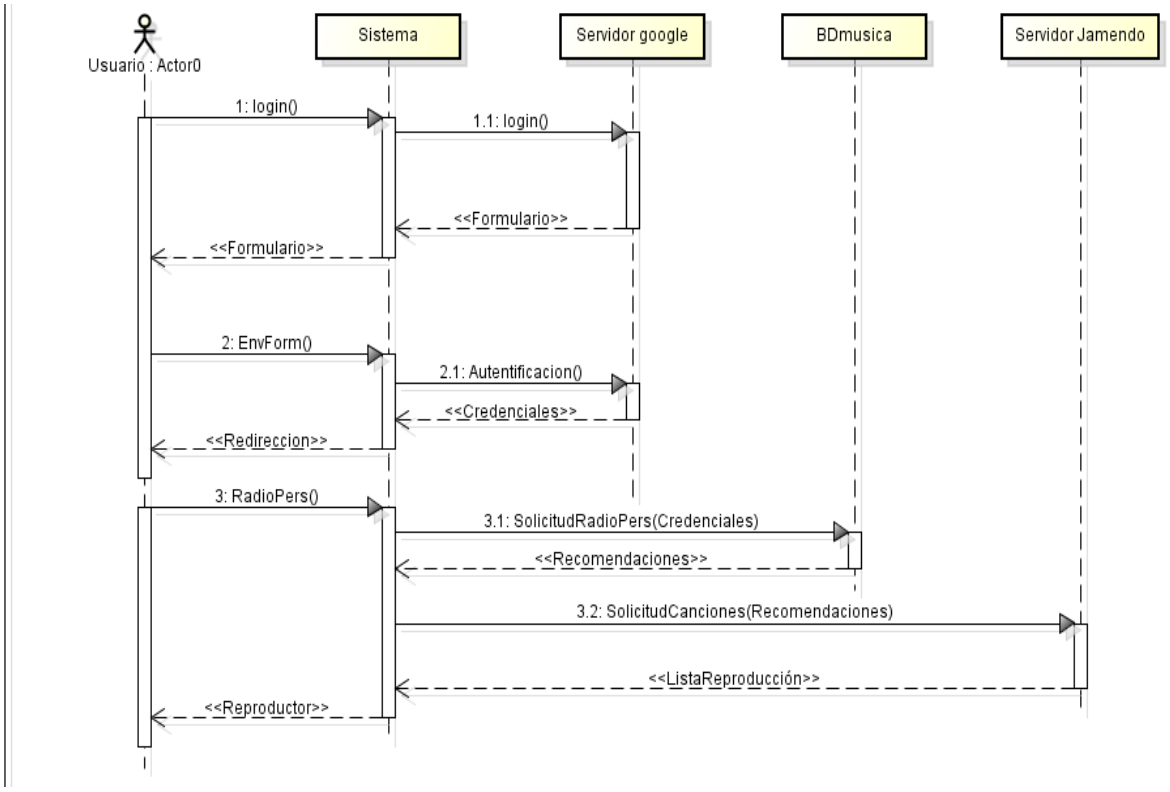

Figura 4.2.6: Diagrama de secuencia de Login

En este diagrama se muestra la interacción que tiene el usuario con el sistema, y este a su vez con otros sistemas, para realizar el login de un usuario. Para ello se muestran detalles como los mensajes intercambiados y el flujo temporal de interacción.

Escenarios

Gestión usuario

Login

Escenario principal

Juan accede a la página y pincha en identificarse. El sistema muestra un formulario a rellenar con los campos email y contraseña y un enlace de ayuda por si el usuario ha olvidado la contraseña. El usuario en ese momento introducirá sus credenciales, usuario: juan@gmail.com y contraseña: juan1234 en el sistema y accede con su cuenta.

Escenarios alternativos

1. El usuario "juan" no existe en el sistema. El sistema creara un nuevo perfil para Juan basándose en las credenciales devueltas por google
2. El usuario "juan" existe pero la contraseña "juan1234" es incorrecta. El sistema advertirá a Juan de que la contraseña es incorrecta y le pedirá que lo intente de nuevo
3. Juan no recuerda su contraseña. El sistema emitirá un mensaje que reflejara este error y le instará a volver a intentarlo.
4. La base de datos no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
5. El servicio de identificación no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Darse de baja

Escenario principal

Juan, ya identificado en el sistema, pincha en eliminar cuenta. El sistema desvincula la cuenta de juan de la aplicación e invalida todos los registros de Juan de la base de datos y lo devuelve al estado de no identificado.

Escenarios alternativos

1. La base de datos no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de identificación no está disponible. El sistema muestra un mensaje advirtiendo de esto a Juan y le sugiere que lo intente dentro de unos minutos.

Cerrar sesión

Escenario principal

Juan, ya identificado en el sistema, pincha en cerrar sesión. El sistema elimina las credenciales de Juan de la sesión actual y lo devuelve al estado de no identificado.

Escenarios alternativos

1. El servicio de identificación no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Radios

Reproductor

Escenario principal

Juan accede a la página y en la sección del reproductor selecciona el botón de reproducir. A continuación empieza a sonar la lista de reproducción.

Escenarios alternativos

1. El servidor de música no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de reproducción no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Radio artistas favoritos

Escenario principal

Juan, una vez identificado en el sistema, selecciona la opción "Radio artistas favoritos". El sistema por su parte llama al servicio de recomendación y genera una lista de reproducción con canciones aleatorias de los artistas cuyas canciones han sido marcadas como favoritas.

Escenarios alternativos

1. El servidor de música no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de reproducción no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Radio canciones favoritas

Escenario principal

Juan, una vez identificado en el sistema, selecciona la opción "Radio canciones favoritas". El sistema por su parte llama al servicio de canciones favoritas y genera una lista de reproducción con canciones marcadas como favoritas.

Escenarios alternativos

1. El servidor de música no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de reproducción no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Radio personalizada

Escenario principal

Juan, una vez identificado en el sistema, selecciona la opción "Radio personalizada". El sistema por su parte llama al servicio de recomendaciones y genera una lista de reproducción basada en el modelo de recomendación

Escenarios alternativos

1. El servidor de música no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de reproducción no está disponible. El sistema muestra un mensaje advirtiendo de error a Juan y le sugiere que lo intente dentro de unos minutos.

Radio normal

Escenario principal

Juan selecciona la opción "Radio normal". El sistema por su parte llama al servicio de música aleatoria y genera una lista de reproducción aleatoria de la música existente en el servidor de música.

Escenarios alternativos

1. El servidor de música no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de reproducción no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Gestión musical

Puntuar canción

Escenario principal

Juan, una vez identificado en el sistema, selecciona la canción "Dangerous" y la valora con 3 corazones. El sistema por su parte llama al servicio de puntaciones y guarda dicha selección en la base de datos.

Escenarios alternativos

1. La base de datos no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de puntuaciones no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

Marcar canción como favorita

Escenario principal

Juan, una vez identificado en el sistema, selecciona la canción "Dangerous" y la marca como favorita. El sistema por su parte llama al servicio de puntaciones y guarda dicha selección en la base de datos.

Escenarios alternativos

1. La base de datos no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.
2. El servicio de puntuaciones no está disponible. El sistema muestra un mensaje advirtiendo de un error a Juan y le sugiere que lo intente dentro de unos minutos.

A continuación se procede al diseño del sistema partiendo de los datos y conclusiones obtenidos del apartado de análisis.

4.3 Diseño

Como se especifica en la introducción de este capítulo el diseño consta de tres partes:

- Diseño del Sistema
- Diseño de Datos
- Diseño de Interfaz

Procedemos a la especificación del diseño de sistema en el apartado siguiente de esta sección.

Diseño del Sistema

El diseño del sistema empieza en este punto con un diagrama de clases donde se muestran los elementos necesarios y las relaciones entre ellos para un correcto funcionamiento de la aplicación

Diagrama de clases

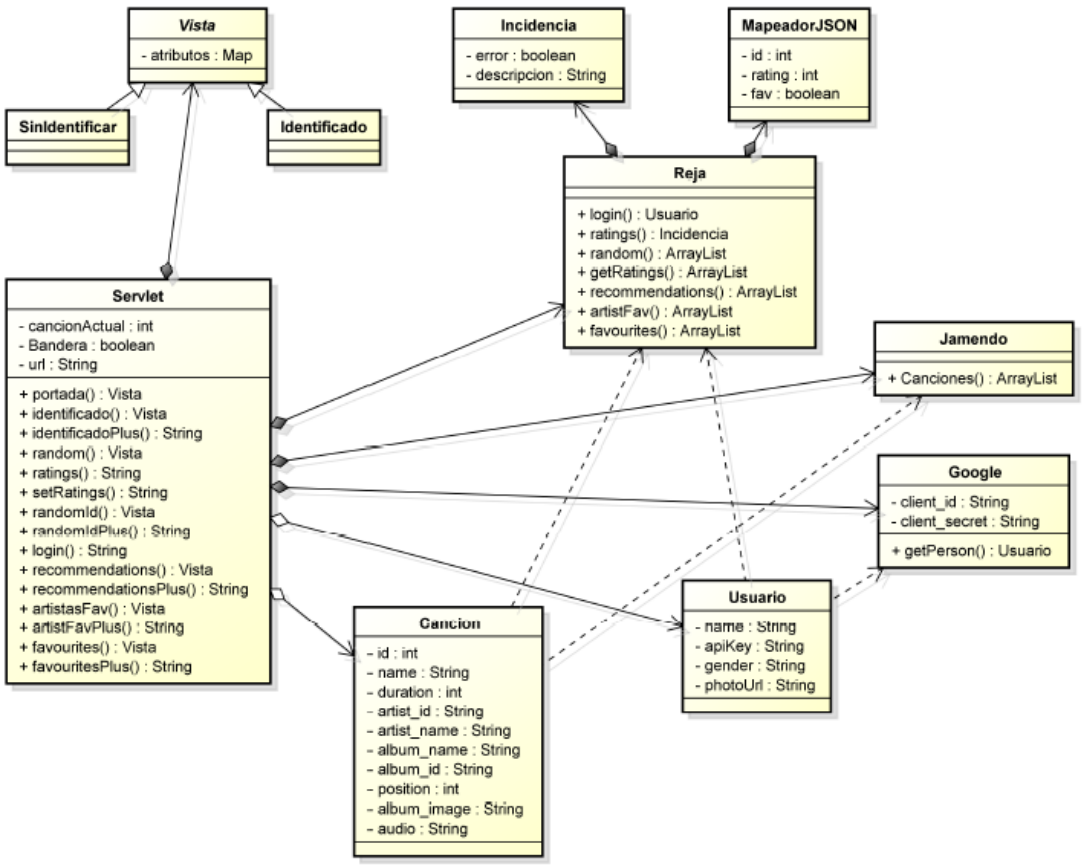

Figura 4.3.1: Diagrama de clases de diseño

En este diagrama se distinguen tres grupos de clases (véase figura 4.3.1):

Las clases pertenecientes a la lógica de aplicación o modelo, formado por canción, usuario, incidencia y mapeador Json. Estas dos últimas solo se utilizan en momentos puntuales para convertir en un formato afable las operaciones del servicio Reja; el cual se explicara a continuación

Las clases pertenecientes a servicios, formadas por Google, Jamendo y Reja. Estas clases son utilizadas para el aprovechamiento de servicios externos, cada una de ellas especializada en el servidor que les da nombre, accediendo a los servicios que ofrecen para nutrir de información necesaria nuestra aplicación como son las canciones de Jamendo, el usuario de Google, y la gestión de la base de datos de Reja.

Por último se encuentra la clase servlet. La cual se encarga de ser el controlador que gestione las vistas con información perteneciente al modelo y gestionar los eventos provenientes de las mismas, ofreciendo así un sistema interactivo.

A continuación se muestra un diagrama de paquetes con esta división definida.

Diagrama de Paquetes

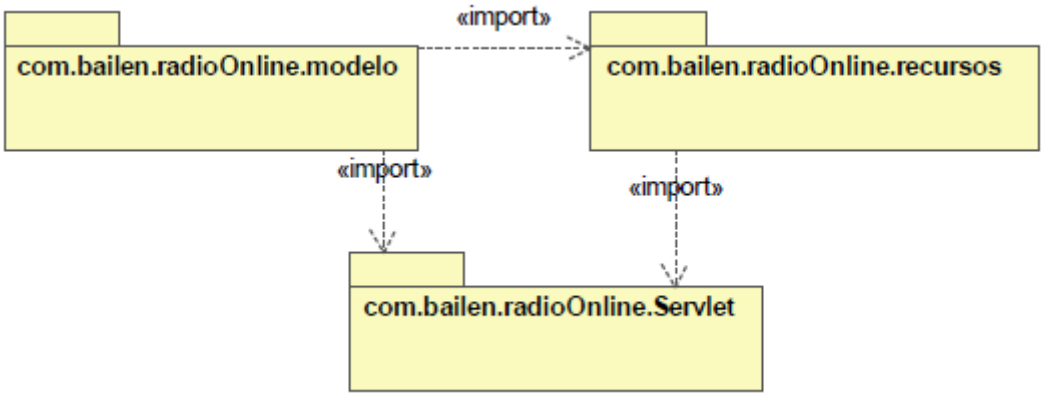

Figura 4.3.2: Diagrama de paquetes

En este diagrama se observa (véase figura 4.3.2) como se ha hecho una división de clases en paquetes según su funcionalidad en la aplicación teniendo en cuenta que clases necesitan más cohesión unas con otras.

A continuación se expone el diseño de datos.

Diseño de Datos

En este apartado estudiaremos la base de datos sobre la que las aplicaciones, tanto esta que nos atañe como la app Android, están montadas. En esta base de datos se encuentran los datos de usuarios necesarios para la identificación así como identificadores de canciones y puntuaciones para poder hacer gestiones sobre ellos; como por ejemplo las recomendaciones.

Hemos optado por la obtención de una base de datos de álbumes musicales reales, todos ellos bajo algunas licencias 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.

Empezaremos con el esquema entidad relación de dicha base de datos y acabaremos con el esquema conceptual modificado.

Esquema Entidad-Relación

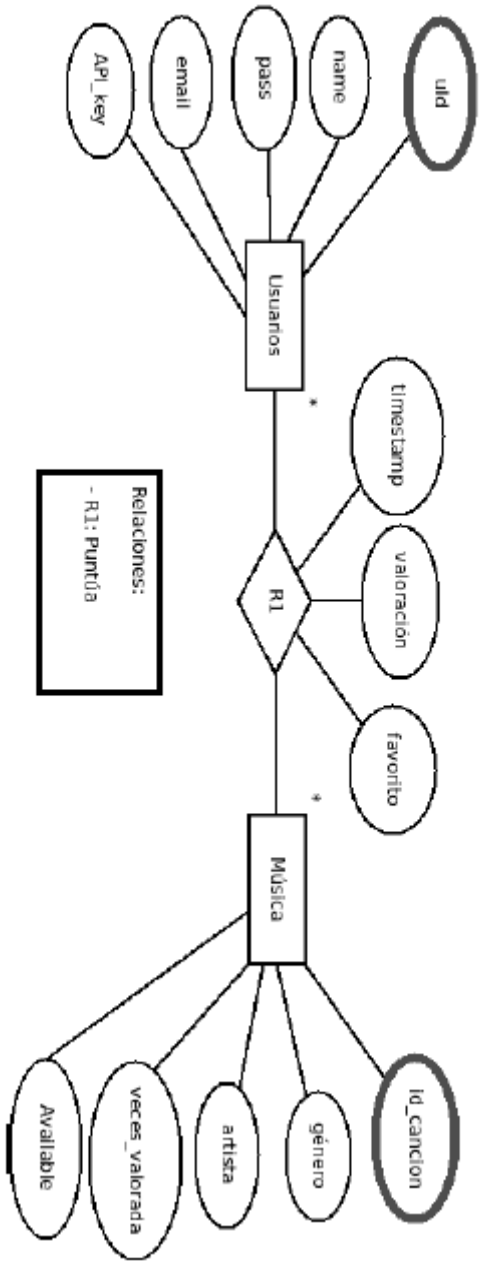

Figura 4.3.3: Esquema Entidad-Relación

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 4.3.3.

A continuación procedemos con esquema conceptual modificado.

Esquema conceptual modificado

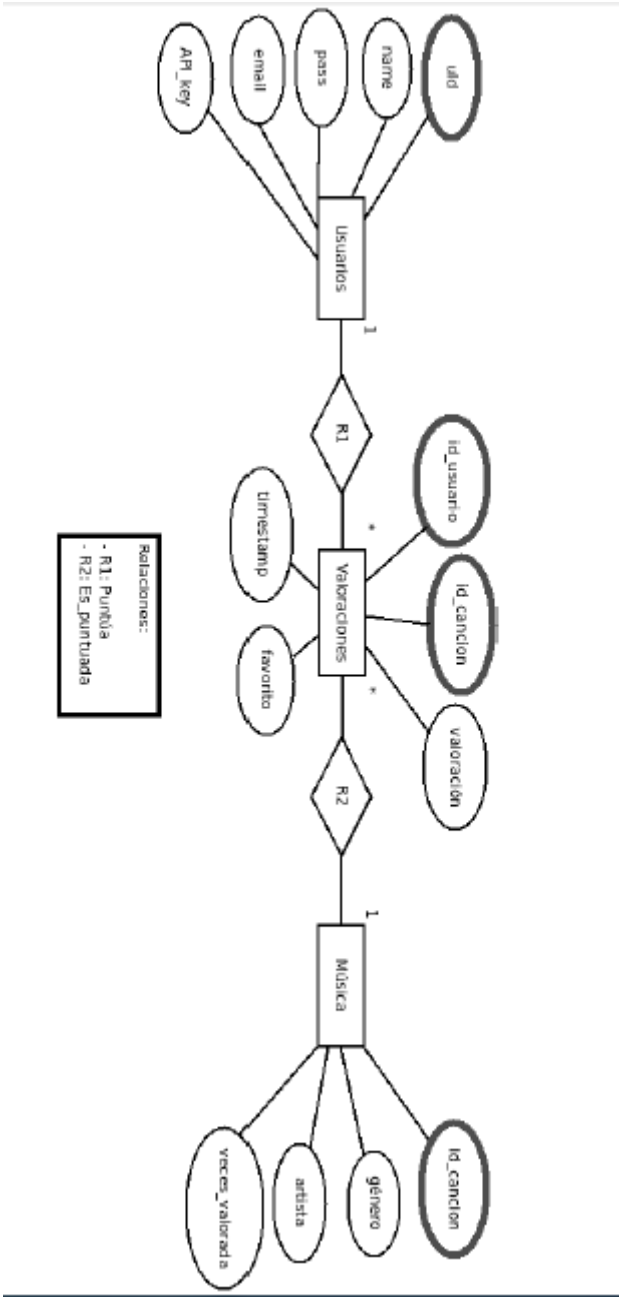

Figura 4.3.4: Esquema Conceptual Modificado

Para la obtención del Esquema Conceptual Modificado (véase figura 4.3.4) a partir del Esquema Entidad-Relación (véase figura 4.3.3) 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 4.3.4.

Procedemos ahora con el diseño de interfaz.

Diseño de 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 [21].

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

Diseño web

El diseño web es una actividad que consiste en la planificación, diseño e implementación de sitios web. No es simplemente una aplicación de diseño convencional, ya que requiere tener en cuenta la navegabilidad, interactividad, usabilidad, arquitectura de la información y la interacción de medios como el audio, texto, imagen, enlaces y video.

La unión de un buen diseño con una jerarquía bien elaborada de contenidos, aumenta la eficiencia de la web como canal de comunicación e intercambio de datos, que brinda posibilidades como el contacto directo entre el productor y el consumidor de contenidos.

El diseño web ha tenido una amplia aplicación en los sectores comerciales de internet; especialmente en la World Wide Web. A menudo la web se utiliza como expresión plástica en sí.

Los elementos de estilo que han sido utilizadas en la página son:

Fuente pantalla "sin identificar" menús

- Tipo de Letra: "Neon Lights"
- Tamaño: 23pt
- Color de la letra: #b549b4
- Color de fondo de la Página: #000000
- Alineación del texto: Center

Fuente pantalla "identificado" menús

- Tipo de Letra: "Neon Lights"
- Tamaño: 23pt
- Color de la letra: #3e2eb7
- Color de fondo de la Página: #000000
- Alineación del texto: Center

Fuente lista reproducción e interior menús

- Tipo de Letra: "Arial"
- Tamaño: 15pt
- Color de la letra: #00bf72
- Color de fondo de la Página: #000000
- Alineación del texto: Left

A continuación se describe que es una metáfora de diseño y las que hemos aplicado en esta aplicación para aumentar su grado de intuición.

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 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:

Tocadiscos

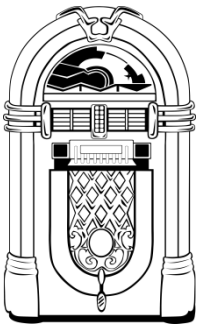

Figura 4.3.5: Metáfora tocadiscos

Como el escritorio en los sistemas operativos, la metáfora del tocadiscos como ecosistema para la aplicación favorece la predisposición del usuario a introducirse en una aplicación musical.

Me gusta

Figura 4.3.6: Metáfora me gusta

Simboliza el grado de agrado que una canción tiene sobre el usuario que la está escuchando. Tiene tres estados que corresponden a cuantos corazones están marcados, a más corazones marcados más agrado.

Canción favorita

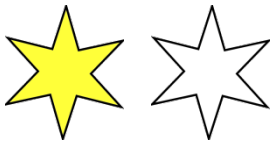

Figura 4.3.7: Metáfora canción favorita

Simboliza si una canción es favorita para el usuario que la está escuchando. Tiene dos estados que corresponden a si la estrella está coloreada o no, coloreada es favorita y no coloreada no favorita

Comenzar música

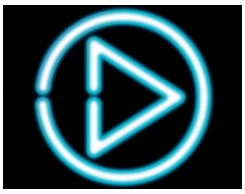

Figura 4.3.8: Metáfora comenzar música

Esta metáfora aparece en la parte perteneciente al reproductor y significa comenzar a reproducir la música.

Parar música

Figura 4.3.9: Metáfora detener música

Esta metáfora aparece en la parte perteneciente al reproductor y significa parar la reproducción de la música.

Siguiente canción

Figura 4.3.10: Metáfora siguiente canción

Esta metáfora aparece en la parte perteneciente al reproductor y significa pasar a la siguiente canción de la lista de reproducción.

Anterior canción

Figura 4.3.11: Metáfora anterior canción

Esta metáfora aparece en la parte perteneciente al reproductor y significa pasar a la siguiente anterior de la lista de reproducción.

A continuación se muestran los prototipos utilizados en el diseño de la página.

Prototipos

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.

La herramienta utilizada para ello es "Axure" [33], una herramienta de prototipado que se basa en el encuadre de elementos y superposición. Con esta herramienta solo se consigue la parte visual de la que disfrutara el usuario introduciendo la funcionalidad posteriormente.

Pantalla principal

Figura 4.3.12: Prototipo pantalla principal

Pantalla usuario autentificado

Figura 4.3.13: Prototipo pantalla usuario identificado

Como observamos ambas pantallas (figuras 4.3.12 y 4.3.13) poseen los mismos elementos:

- Parte superior: Se encuentran los enlaces con las redes sociales y el menú dotado de cuatro opciones acompañadas del logotipo de la radio
- Parte media: Se encuentra el reproductor con los botones necesarios para la gestión de la reproducción, así como la lista de reproducción con las canciones que van a sonar a continuación.
- Parte inferior: Corresponde al pie de página, donde se encuentran el logotipo de la universidad así como la información de contacto necesaria.

La principal diferencia se encuentra en el cambio del patrón de color; el cual ayuda al usuario a situarse mejor en la pantalla que se encuentra. Al no cambiar la estructura de las páginas la navegación es más consistente entre ambas.

A continuación se exponen los caminos de navegación entre una página y otra; así como los modos de acceso la implementación de la aplicación.

Story Board

Un storyboard, en el contexto de ingeniería software, es un conjunto de ilustraciones mostradas en secuencia con el objetivo de servir de guía para entender la transición entre las distintas pantallas de una aplicación software.

Identificación

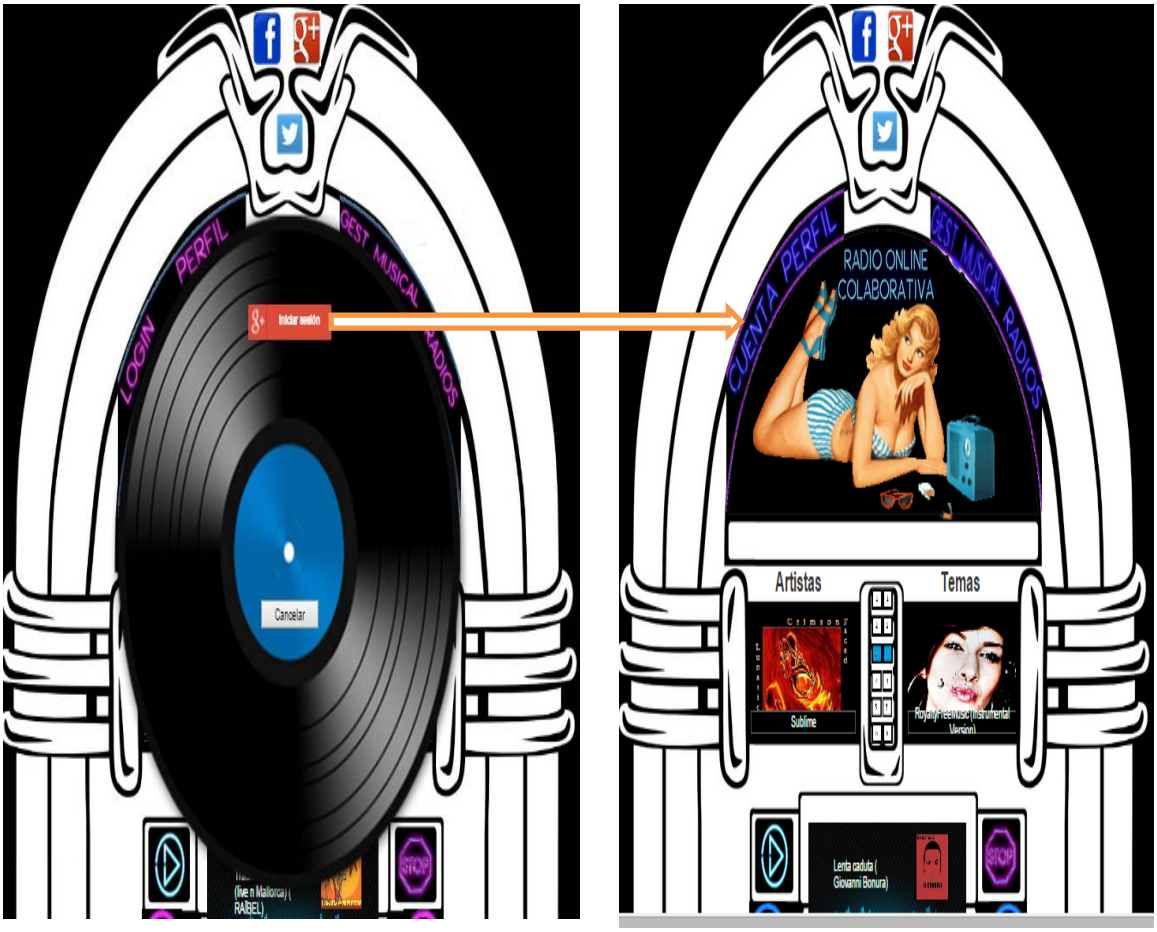

Figura 4.3.14: Story Board Identificación

Al pulsar el botón de "iniciar sesión" en la página principal, el sistema nos redirige hacia la página de usuarios identificados con nuestro perfil; lo que quiere decir que las acciones que se encuentran en esta pantalla están personalizadas hacia el usuario que acaba de acceder (véase figura 4.3.14).

Cerrar sesión/Eliminar Cuenta

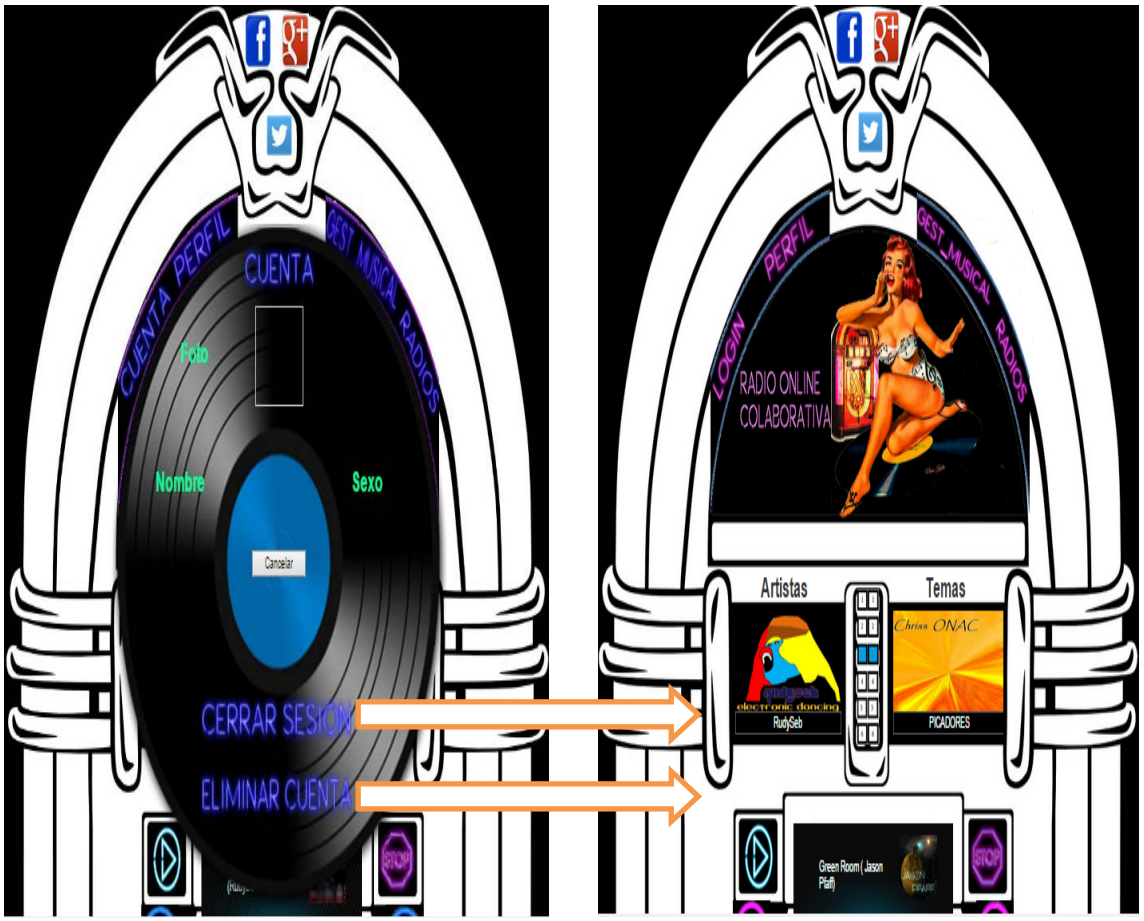

Figura 4.3.15: Story Board Cerrar Sesión/Eliminar Cuenta

Tanto al cerrar sesión como al eliminar la cuenta, el sistema nos redirige a la página principal como usuario no autentificado (véase figura 4.3.15).

Cuando cerramos sesión simplemente las credenciales del usuario y su perfil son eliminados de la sesión manteniéndose en base de datos; mientras que cuando eliminamos la cuenta, los registros del usuario, como preferencias musicales, son desvinculados de él. De manera que la próxima vez que acceda al sistema sus registros estarán vacíos.

A continuación se expone la implementación; la cual comprende la descripción del framework que se ha utilizado así como la arquitectura utilizada.

4.4 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 primer lugar pasamos a explicar los fundamentos de la arquitectura utilizada para este proyecto, arquitectura cliente servidor.

Arquitectura Cliente-Servidor

La arquitectura software es el diseño de más alto nivel de la estructura de un sistema. Es lo que proporciona el punto de inicio para el diseño software; y la arquitectura cliente servidor para una aplicación web, como es este caso es la más adecuada. A continuación se aclaran sus funciones.

El servidor debe negociar con su Sistema Operativo un puerto (casi siempre conocido) donde esperar las solicitudes (Véase figura 4.4.1). El servidor espera pasivamente las peticiones en un puerto conocido que ha sido reservado para el servicio que ofrece. El cliente también solicita, a su sistema operativo, un puerto no usado desde el cual enviar su solicitud y esperar respuesta. Un cliente ubica un puerto arbitrario, no utilizado y no reservado, para su comunicación.

En una interacción se necesita reservar solo uno de los dos puertos, asignando un identificador único de puerto para cada servicio, facilitando así la construcción de clientes y servidores.

Los servidores por lo general son más difíciles de construir que los clientes pues aunque se implantan como programas de aplicación deben manejar peticiones concurrentes, así como reforzar todos los procedimientos de acceso y protección del sistema computacional en el que corren, y protegerse contra todos los errores posibles. El cliente y el servidor pueden interactuar en la misma máquina.

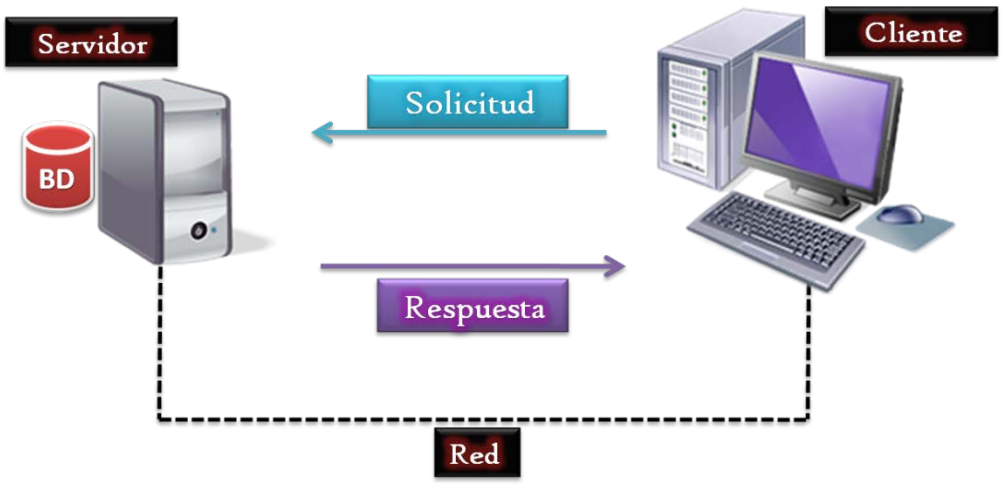

Figura 4.4.1: Arquitectura Cliente-Servidor

Componentes de la arquitectura Cliente-Servidor

Cliente: Programa ejecutable que participa activamente en el establecimiento de las conexiones. Envía una petición al servidor y se queda esperando por una respuesta. Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.

Servidor: Es un programa que ofrece un servicio que se puede obtener en una red. Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante. Al ser posible implantarlo como aplicaciones de programas, puede ejecutarse en cualquier sistema donde exista TCP/IP y junto con otros programas de aplicación. El servidor comienza su ejecución antes de comenzar la interacción con el cliente. Su tiempo de vida o de interacción es indefinido.

Los servidores pueden ejecutar tareas sencillas o complejas (caso del servidor ftp en el cual se deben realizar operaciones antes de devolver una respuesta). Los servidores sencillos procesan una petición a la vez (son secuenciales o interactivos), por lo que no revisan si ha llegado otra petición antes de enviar la respuesta de la anterior.

Los más complejos trabajan con peticiones concurrentes aun cuando una sola petición lleve mucho tiempo para ser servida (caso del servidor ftp que debe copiar un archivo en otra máquina). Son complejos pues tienen altos requerimientos de protección y autorización. Pueden leer archivos del sistema, mantenerse en línea y acceder a datos protegidos y a archivos de usuarios. No puede cumplir a ciegas las peticiones de los clientes, deben reforzar el acceso al sistema y las políticas de protección. Los servidores por lo general tienen dos partes:

1. Programa o proceso que es responsable de aceptar nuevas peticiones: Maestro o Padre.

2. Programas o procesos que deben manejar las peticiones individuales: Esclavos o Hijos.

Características de la arquitectura Cliente-Servidor

- Combinación de un cliente que interactúa con el usuario, y un servidor que interactúa con los recursos a compartir. El proceso del cliente proporciona la interfaz entre el usuario y el resto del sistema. El proceso del servidor actúa como un motor de software que maneja recursos compartidos tales como bases de datos, impresoras, Módem, etc.
- Las tareas del cliente y del servidor tienen diferentes requerimientos en cuanto a recursos de cómputo como velocidad del procesador, memoria, velocidad y capacidades del disco e input-output devices.
- Se establece una relación entre procesos distintos, los cuales pueden ser ejecutados en la misma máquina o en máquinas diferentes distribuidas a lo largo de la red.
- Existe una clara distinción de funciones basadas en el concepto de "servicio", que se establece entre clientes y servidores.
- La relación establecida puede ser de muchos a uno, en la que un servidor puede dar servicio a muchos clientes, regulando su acceso a los recursos compartidos.
- Los clientes corresponden a procesos activos en cuanto a que son estos los que hacen peticiones de servicios. Estos últimos tienen un carácter pasivo, ya que esperan peticiones de los clientes.
- No existe otra relación entre clientes y servidores que no sea la que se establece a través del intercambio de mensajes entre ambos. El mensaje es el mecanismo para la petición y entrega de solicitudes de servicios.
- El ambiente es heterogéneo. La plataforma de hardware y el sistema operativo del cliente y del servidor no son siempre los mismos. Precisamente una de las principales ventajas de esta arquitectura es la posibilidad de conectar clientes y servidores independientemente de sus plataformas.
- El concepto de escalabilidad tanto horizontal como vertical es aplicable a cualquier sistema Cliente-Servidor. La escalabilidad horizontal permite agregar más estaciones de trabajo activas sin afectar significativamente el rendimiento. La escalabilidad vertical permite mejorar las características del servidor o agregar múltiples servidores.

Ventajas de la arquitectura Cliente-Servidor

- Existencia de plataformas de hardware cada vez más baratas constituye una de las más palpables ventajas de este esquema, la posibilidad de utilizar máquinas mucho más baratas que las requeridas por una solución centralizada, basada en sistemas grandes (mainframes). Además, se pueden utilizar componentes, tanto de hardware como de software, de varios fabricantes, lo cual contribuye considerablemente a la reducción de costos y favorece la flexibilidad en la implantación y actualización de soluciones.
- Facilita la integración entre sistemas diferentes y comparte información, permitiendo por ejemplo que las máquinas ya existentes puedan ser utilizadas pero utilizando interfaces más usables para el usuario. De esta manera, se pueden integrar PCs con sistemas medianos y grandes, sin necesidad de que todos tengan que utilizar el mismo sistema operativo.
- Al favorecer el uso de interfaces gráficas interactivas, los sistemas construidos bajo este esquema tienen una mayor y más intuitiva interactividad con el usuario. En el uso de interfaces gráficas para el usuario, presenta la ventaja, con respecto a uno centralizado, de que no siempre es necesario transmitir información gráfica por la red pues esta puede residir en el cliente, lo cual permite aprovechar mejor el ancho de banda de la red.
- La estructura inherentemente modular facilita además la integración de nuevas tecnologías y el crecimiento de la infraestructura computacional, favoreciendo así la escalabilidad de las soluciones.
- Contribuye además a proporcionar a los diferentes departamentos de una organización, soluciones locales, pero permitiendo la integración de la información.

Desventajas de la arquitectura Cliente-Servidor

- El mantenimiento de los sistemas es más difícil pues implica la interacción de diferentes partes de hardware y de software, distribuidas por distintos proveedores, lo cual dificulta el diagnóstico de fallos.
- Cuenta con muy escasas herramientas para la administración y ajuste del desempeño de los sistemas.
- Es importante que los clientes y los servidores utilicen el mismo mecanismo (por ejemplo sockets o RPC), lo cual implica que se deben tener mecanismos generales que existan en diferentes plataformas.
- Hay que tener estrategias para el manejo de errores y para mantener la consistencia de los datos.
- El desempeño (performance), problemas de este estilo pueden presentarse por congestión en la red, dificultad de tráfico de datos, etc.

En concreto en este TFG se han implementado unos servicios REST en el servidor sinbad2 de la universidad de Jaén, así como un servidor, con una serie de servicios REST, para una aplicación web que se aprovecha de estos.

De esta manera la app Android se aprovecha de los servicios ofrecidos desde sinbad2 al igual que el servidor para la aplicación web (véase figura 4.4.2):

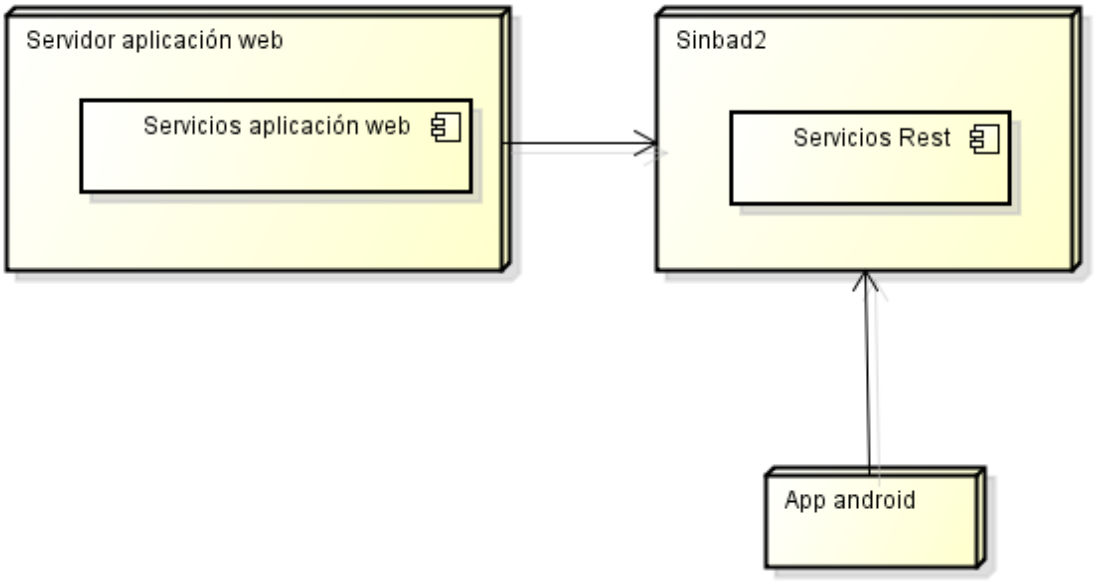

Figura 4.4.2: Esquema de funcionamiento Cliente-Servidor

A continuación pasamos a describir el framework utilizado para la implementación de esta aplicación.

Framework

Definición

La palabra inglesa "framework" (marco de trabajo) define, en términos generales, un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

En el desarrollo de software, un framework o infraestructura digital, es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. Típicamente, puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado, entre otras herramientas, para así ayudar a desarrollar y unir los diferentes componentes de un proyecto.

En definitiva es una herramienta de ayuda en el desarrollo de una aplicación.

A continuación explicamos el framework "Spring" centrándonos en sus principales componentes.

Spring

"Spring" es un framework para el desarrollo de aplicaciones y contenedor de inversión de control, de código abierto para la plataforma Java. Su gran potencial se debe a sus módulos:

- Contenedor de inversión de control: permite la configuración de los componentes de aplicación y la administración del ciclo de vida de los objetos Java, se lleva a cabo principalmente a través de la inyección de dependencias.
- Programación orientada a aspectos: habilita la implementación de rutinas transversales.
- Acceso a datos: se trabaja con RDBMS en la plataforma java, usando Java Database Connectivity y herramientas de Mapeo objeto relacional con bases de datos NoSQL.
- Gestión de transacciones: unifica distintas APIs de gestión y coordina las transacciones para los objetos Java.
- Modelo vista controlador: Un framework basado en HTTP y servlets, que provee herramientas para la extensión y personalización de aplicaciones web y servicios web REST.

Aunque existen muchos más módulos de este framework; estos pueden considerarse los más utilizados. A continuación exponemos las ventajas y los inconvenientes de usar un framework.

Ventajas de usar framework

- Rapidez. Como una gran parte del trabajo está ya hecho, el usar un framework te permitirá ir mucho más rápido en la creación de sitios web una vez sepas cómo funciona el framework.
- Comodidad. Al haber sido ya desarrolladas muchas de las funcionalidades que utilizas en tu día a día, no vas a tener que pararte a pensar cómo hacer las cosas, lo que puede ser francamente cómodo.
- Componentes. Los frameworks robustos traen muchos componentes adicionales en forma de plugins de JavaScript, lo que te permite implementar soluciones un poco más avanzadas sin necesidad de desarrollarlas tú mismo.
- Soporte. Los frameworks conocidos tienen una amplia comunidad online detrás, por lo que te será fácil encontrar respuesta a tus dudas en caso de que estas surjan.

Si el presupuesto es bajo o tienes un tiempo limitado, entonces usar un framework te puede facilitar enormemente la vida, ya que una buena parte del trabajo está ya hecho.

Inconvenientes de usar framework

- Aprendizaje. Cada framework funciona de un modo distinto, con lo que hay que aprender a utilizarlo.
- Exceso de código. Un buen framework viene pensado para cubrir las necesidades de la mayoría de desarrolladores, pero esto no significa que en cada proyecto se vayan a desarrollar todas ellas. Hay frameworks como Bootstrap que te permiten personalizar la descarga, pero si no es así, puedes estar utilizando mucho código que no necesitas.
- Semántica. Los frameworks utilizan un sistema de grid con una serie de clases extra que permiten identificar el número de columnas a ocupar. Los nombres de estas clases no son semánticos y, por tanto, el código del framework está más pensado para funcionalidad y rapidez que para semántica.
- Aspecto final. Al traer muchas cosas ya desarrolladas "de fábrica", el aspecto puede ser bastante similar de un sitio a otro si no se personaliza el CSS del framework. De aquí que se piensa que los frameworks es una herramienta más bien pensada para desarrolladores que no para diseñadores.

El uso o no de un framework depende del caso. Más rápido y barato, pero con menos control y necesidad de aprenderlo antes.

Para terminar con la implementación, acabamos con la definición de los lenguajes de desarrollo utilizados.

Lenguajes de desarrollo utilizados

Parte Cliente

- Javascript: Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.
- HTML: Hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código para la definición de contenido de una página web, como texto, imágenes, videos, entre otros.

Parte Servidor

- 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.
- XML: Lenguaje de marcas para documentos que permite almacenar información de forma estructurada. Spring lo utiliza para las configuraciones.

La parte implementada en el servidor sinbad2 consta de una serie de servicios REST. La tabla de recursos que se ha diseñado en este proyecto la podemos ver en la tabla 4.4.1:

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, fav	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.
/ratings	GET	API Key	Devuelve una lista de canciones votadas por el usuario al que pertenezca la API Key
/favouriteArtists	GET	API Key	Devuelve una lista de canciones pertenecientes a los artistas cuyas, algunas, canciones haya valorado como favoritas

Tabla 4.4.1: Servicios REST

La implementación de esta parte se ha realizado con el lenguaje 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 último, en este capítulo de ingeniería software, describiremos las pruebas realizadas para la verificación de los requisitos funcionales necesarios para que la aplicación se ajuste a las pretensiones expuestas por el cliente.

4.5 Pruebas Software

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada.

Las pruebas son básicamente un conjunto de actividades dentro del proceso de ingeniería software. Dependiendo del tipo de pruebas, estas actividades podrán ser implementadas en cualquier momento de dicho proceso de desarrollo.

Para la verificación de esta aplicación vamos a utilizar tres tipos de pruebas:

- Casos de Test: Se testan los requisitos funcionales de la aplicación.
- Unitarias: Comprueban la funcionalidad de cada módulo independientemente
- Integración: Comprueban el correcto funcionamiento de la integración de unos módulos con otros.

Sin más dilación comenzamos con las pruebas unitarias.

Pruebas Unitarias

Una prueba unitaria [26] es una forma de comprobar el correcto funcionamiento de un módulo de código. Esto sirve para asegurar que cada uno de los módulos funcione correctamente por separado. Luego, con las Pruebas de Integración, se podrá asegurar el correcto funcionamiento del sistema o subsistema en cuestión.

La idea es escribir casos de prueba para cada función no trivial o método en el módulo, de forma que cada caso sea independiente del resto.

Seguidamente se muestran las características principales de este tipo de pruebas, tanto como sus ventajas y limitaciones.

Características

Para que una prueba unitaria tenga la calidad suficiente se deben cumplir los siguientes requisitos:

- Automatizable: No debería requerirse una intervención manual. Esto es especialmente útil para integración continua.
- Completas: Deben cubrir la mayor cantidad de código.
- Repetibles o Reutilizables: No se deben crear pruebas que sólo puedan ser ejecutadas una sola vez. También es útil para integración continua.
- Independientes: La ejecución de una prueba no debe afectar a la ejecución de otra.

- Profesionales: Las pruebas deben ser consideradas igual que el código, con la misma profesionalidad, documentación, etc.

Ventajas

El objetivo de las pruebas unitarias es aislar cada parte del programa y mostrar que las partes individuales son correctas. Proporcionan un contrato escrito que el trozo de código debe satisfacer. Estas pruebas aisladas proporcionan cinco ventajas básicas:

- Fomentan el cambio: Las pruebas unitarias facilitan que el programador cambie el código para mejorar su estructura (lo que se ha dado en llamar refactorización), puesto que permiten hacer pruebas sobre los cambios y así asegurarse de que los nuevos cambios no han introducido errores.
- Simplifica la integración: Puesto que permiten llegar a la fase de integración con un grado alto de seguridad de que el código está funcionando correctamente. De esta manera se facilitan las pruebas de integración.
- Documenta el código: Las propias pruebas son documentación del código puesto que ahí se puede ver cómo utilizarlo.
- Separación de la interfaz y la implementación: Dado que la única interacción entre los casos de prueba y las unidades bajo prueba son las interfaces de estas últimas, se puede cambiar cualquiera de los dos sin afectar al otro, a veces usando objetos mock (mock object) para simular el comportamiento de objetos complejos.
- Los errores están más acotados y son más fáciles de localizar: Dado que tenemos pruebas unitarias que pueden desenmascararlos.

Limitaciones

Es importante darse cuenta de que las pruebas unitarias no descubrirán todos los errores del código. Algunos enfoques se basan en la generación aleatoria de objetos para amplificar el alcance de las pruebas de unidad. Esta técnica se conoce como testing aleatorio (RT, por random testing). Por definición, sólo prueban las unidades por sí solas.

Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto. Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir en realidad la unidad de programa bajo estudio. Las pruebas unitarias sólo son efectivas si se usan en conjunto con otras pruebas de software.

A continuación se exponen las pruebas de integración

Pruebas de integración

Pruebas integrales o pruebas de integración [27] son aquellas que se realizan en el ámbito del desarrollo de software una vez que se han aprobado las pruebas unitarias. Únicamente se refieren a la prueba o pruebas de todos los elementos unitarios que componen un proceso, hecha en conjunto, de una sola vez.

Consiste en realizar pruebas para verificar que un gran conjunto de partes de software funcionan juntos.

Las pruebas de integración (algunas veces llamadas integración y testeo I&T) es la fase de la prueba de software en la cual módulos individuales de software son combinados y probados como un grupo. Son las pruebas posteriores a las pruebas unitarias

Veamos sus ventajas y limitaciones

Ventajas

- Cobertura de código: Si definimos cobertura como las líneas de código recorridas durante la ejecución de un Test, podemos decir que los Test de Integración recorren mucho más código que los Unitarios ya que pasan por más regiones de código. Sin embargo y dependiendo del diseño del código, habrá regiones de código que requieran de complejas condiciones iniciales para que el código sea alcanzado por un Test de Integración lo cual es un punto "en contra" de los Test de Integración
- Mantenibilidad de los Test: Este es un punto muy controvertido. Mientras que algunos mantienen que los test unitarios son más fáciles de mantener otros, que los test de integración son más fáciles de mantener pero la verdad que siempre dependerá de los tipos de cambios a los que se vean sometidos el código.

Limitaciones

- Cobertura falseada por los test unitarios: Como comentaba en el apartado de cobertura, se podrían dar regiones de código que pueden ser difícilmente alcanzable por las pruebas de integración. Si estas regiones estuvieran recorridas mediante pruebas unitarias se estaría ocultando este dato, es decir, no conoceríamos regiones de código que nunca se recorren y que pueden ser re implementadas para que sea un código realmente útil.
- Veracidad de las pruebas: De todos es sabido que imposible implementar, en un tiempo razonable, un conjunto de pruebas que garanticen que un software de cierto tamaño no falle. Y esta imposibilidad aumenta con el tamaño del proyecto.

Casos de Test

Test 1: Autenticación de usuario

REQUISITOS TESTEADOS	RF-03
Acción	Un usuario pulsa el botón de iniciar sesión y aparece la pantalla de google para su autenticación; a continuación el usuario introduce su email y contraseña.
Checkpoint 1	El sistema debe mostrar la pantalla para usuarios autentificados.

Tabla 4.5.1: Test 1

Test 2: Autenticación de usuario incorrecta

REQUISITOS TESTEADOS	RF-03
Acción	Un usuario pulsa el botón de iniciar sesión y aparece la pantalla de google para su autenticación; a continuación el usuario introduce su email y contraseña.
Checkpoint 1	La pantalla de google debe mostrar que las credenciales son incorrectas.

Tabla 4.5.2: Test 2

Test 3: Cerrar sesión

REQUISITOS TESTEADOS	RF-04
Acción	Un usuario pulsa el botón de cerrar sesión
Checkpoint 1	El sistema debe mostrar la pantalla principal para usuarios no autentificados

Tabla 4.5.3: Test 3

Test 4: Error al cerrar sesión

REQUISITOS TESTEADOS	RF-04
Acción	Un usuario pulsa el botón de cerrar sesión
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.4: Test 4

Test 5: Darse de baja

REQUISITOS TESTEADOS	RF-02
Acción	Un usuario pulsa el botón de eliminar cuenta
Checkpoint 1	El sistema debe mostrar la pantalla principal para usuarios no autentificados

Tabla 4.5.5: Test 5

Test 6: Error al eliminar cuenta

REQUISITOS TESTEADOS	RF-02
Acción	Un usuario pulsa el botón de eliminar cuenta
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.6: Test 6

Test 7: Puntuar canciones

REQUISITOS TESTEADOS	RF-12
Acción	Un usuario selecciona la puntuación que considere más adecuada para una canción y pulsa enviar
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados

Tabla 4.5.7: Test 7

Test 8: Error al puntuar canciones

REQUISITOS TESTEADOS	RF-12
Acción	Un usuario selecciona la puntuación que considere más adecuada para una canción y pulsa enviar
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.8: Test 8

Test 9: Marcar canción como favorita

REQUISITOS TESTEADOS	RF-13
Acción	Un usuario selecciona una canción como favorita y pulsa enviar
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados

Tabla 4.5.9: Test 9

Test 10: Error al marcar canción como favorita

REQUISITOS TESTEADOS	RF-13
Acción	Un usuario selecciona una canción como favorita y pulsa enviar
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.10: Test 10

Test 11: Radio personalizada

REQUISITOS TESTEADOS	RF-07
Acción	Un usuario selecciona la opción de radio personalizada
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados y cambiar la lista de reproducción

Tabla 4.5.11: Test 11

Test 12: Error radio personalizada

REQUISITOS TESTEADOS	RF-07
Acción	Un usuario selecciona la opción de radio personalizada
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.12: Test 12

Test 13: Radio normal

REQUISITOS TESTEADOS	RF-08
Acción	Un usuario selecciona la opción de radio normal
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados y cambiar la lista de reproducción

Tabla 4.5.13: Test 13

Test 14: Error radio normal

REQUISITOS TESTEADOS	RF-08
Acción	Un usuario selecciona la opción de radio normal
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.14: Test 14

Test 15: Radio canciones favoritas

REQUISITOS TESTEADOS	RF-09
Acción	Un usuario selecciona la opción de radio favoritos
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados y cambiar la lista de reproducción

Tabla 4.5.15: Test 15

Test 16: Error radio canciones favoritas

REQUISITOS TESTEADOS	RF-09
Acción	Un usuario selecciona la opción de radio favoritos
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.16: Test 16

Test 17: Radio artistas favoritos

REQUISITOS TESTEADOS	RF-10
Acción	Un usuario selecciona la opción de radio artistas favoritos
Checkpoint 1	El sistema debe devolverlo a la página para usuarios identificados y cambiar la lista de reproducción

Tabla 4.5.17: Test 17

Test 18: Error radio artistas favoritos

REQUISITOS TESTEADOS	RF-10
Acción	Un usuario selecciona la opción de radio artistas favoritos
Checkpoint 1	El sistema debe mostrar una pantalla de error indicando que ha habido un problema

Tabla 4.5.18: Test 18

Resultados

Comenzamos con los casos de test:

Test	Resultado
Test 1	OK
Test 2	OK
Test 3	OK
Test 4	OK
Test 5	OK
Test 6	OK
Test 7	OK
Test 8	OK
Test 9	OK
Test 10	OK
Test 11	OK
Test 12	OK
Test 13	OK
Test 14	OK
Test 15	OK
Test 16	OK
Test 17	OK
Test 18	OK

Tabla 4.5.19: Resultados Casos Test

Los resultados de los casos de test han sido positivos en todos los casos. Ademas se han realizado a través de "Junit" las pruebas unitarias pertenecientes a los grupos de servicios y lógica de negocio, dado que son elementos que trabajan independientemente de otras clases y son una muestra representativa de los resultados globales.

Clase canción

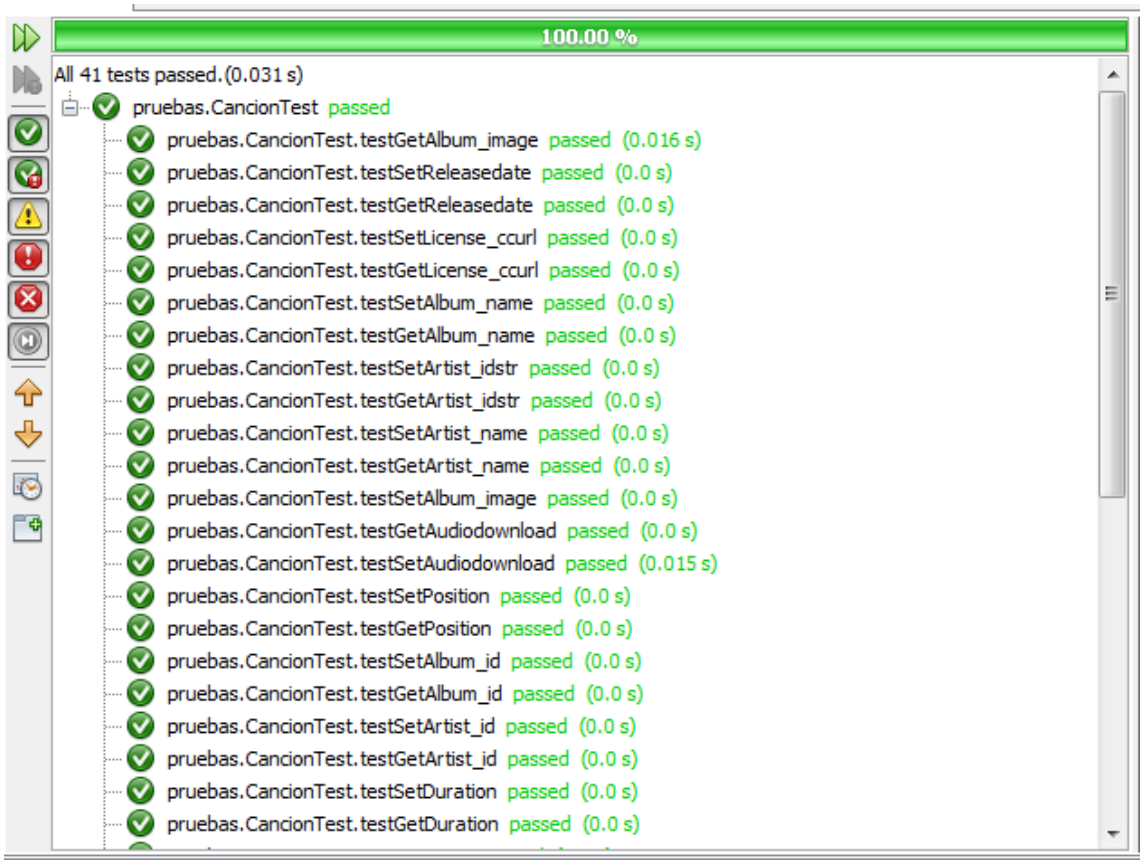

Figura 4.5.1: Resultado prueba unitaria clase canción

Clase Reja

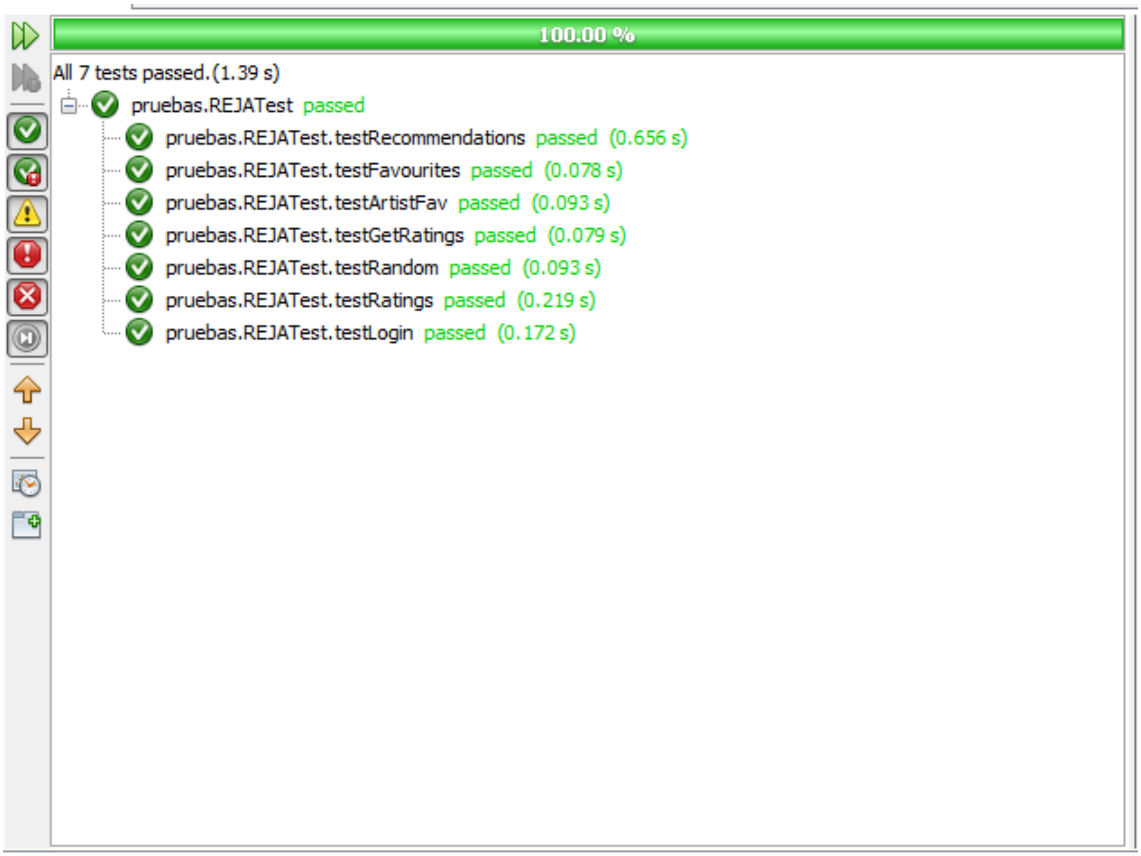

Figura 4.5.2: Resultado prueba unitaria clase Reja

Una vez verificadas las pruebas unitarias sobre las clases que poseen dependencias con otras clases procedemos a realizar la prueba de integración a la clase controladora; ya que esta constituye la integración de todos los elementos restantes de la aplicación.

Clase Servlet

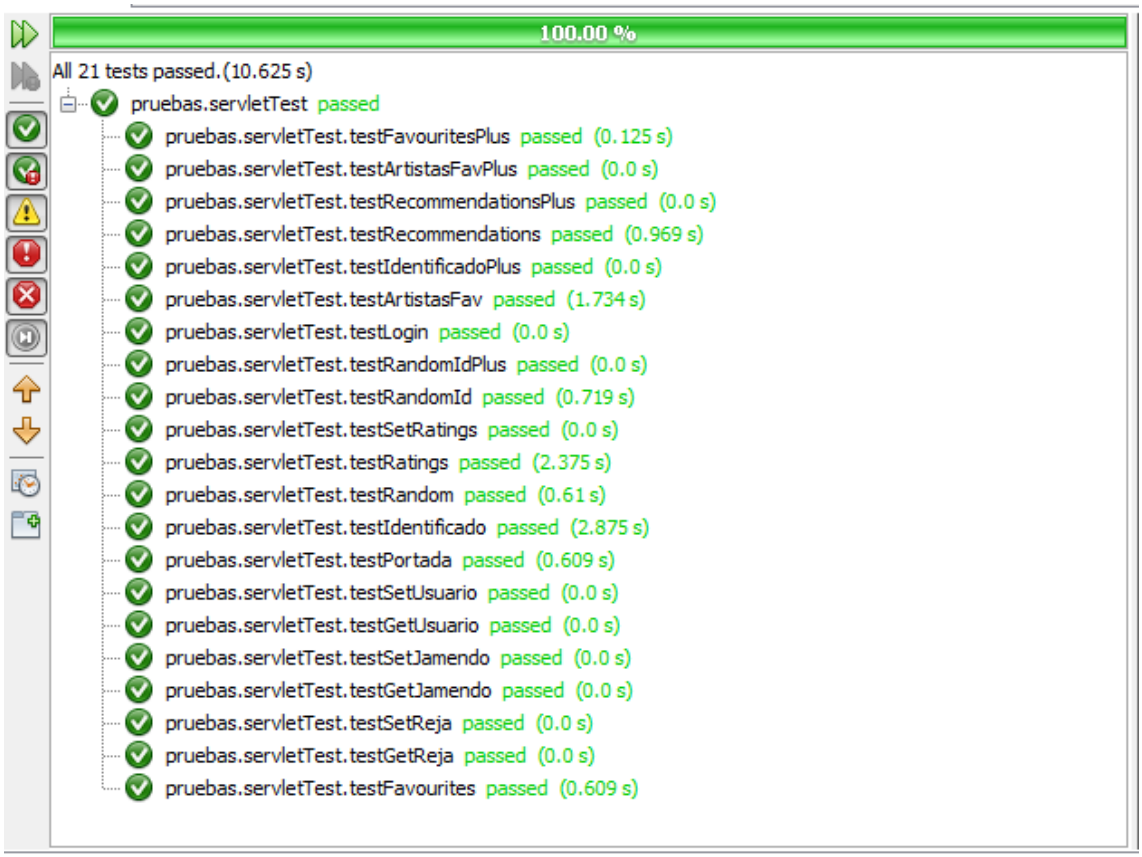

Figura 4.5.3: Resultado prueba integración clase servlet

Verificadas las pruebas tanto unitarias (figuras 4.5.1 y 4.5.2) como de integración (figura 4.5.3) damos por finalizado el capítulo de ingeniería software y procedemos a exponer las conclusiones obtenidas de este proyecto.

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 Web 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 de este sistema, la intención era crear un servicio que permitiera a cualquier tipo de aplicación acceder al sistema de recomendación musical; ahora con esta aplicación web sumada a la aplicación Android prexistente, nos acercamos a ese objetivo. 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 licencias 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" [22], hemos reutilizado el algoritmo de filtrado colaborativo basado en ítem y, más concretamente, en el vecino más cercano (K-nn). De forma paralela, hemos desarrollado este proyecto, con el análisis, diseño e implementación y pruebas de una aplicación web 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 y desplegado en un servidor.

Por último, añadir que para el autor del proyecto ha supuesto una enorme satisfacción el haber conseguido darle forma a esta idea 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. 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.

Apéndice 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.
- Backup de la máquina virtual utilizada para el servidor.
- Código de la Aplicación.

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. Señalar que la aplicación puede ser accedida si necesidad de instalación en http://150.214.174.25: 8064/RadioWebPrueba

Servidor

Para la instalación de la parte servidor habría de instalar el software siguiente:

- GlassFish Server 4.1
- Java EE 7 Web

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.

Pasos a seguir:

1. Ejecutamos el instalador y pulsamos “Next” para continuar con la instalación

Figura A.1: Instalación VirtualBox 1

2. Aceptamos la instalación por defecto y pulsamos "Next" para que comience la instalación

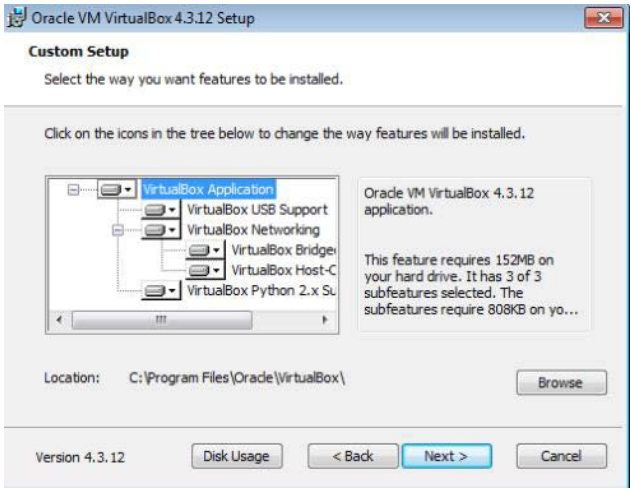

Figura A.2: Instalación VirtualBox 2

3. La instalación ha finalizado y pulsamos "Finish" para abrir la aplicación.

Figura A.3: Instalación VirtualBox 3

4. Dentro de VirtualBox pulsamos Archivo > Importar servicio virtualizado...

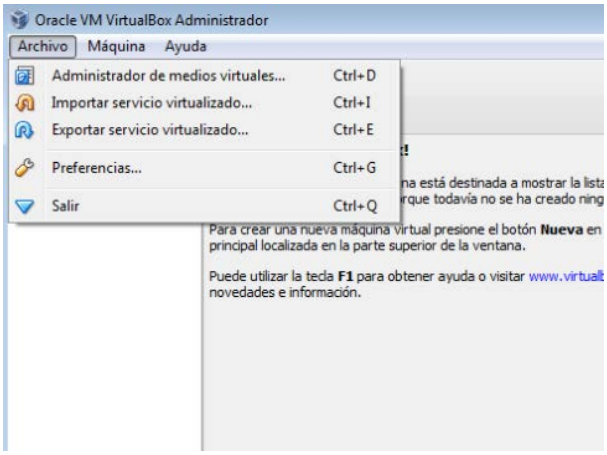

Figura A.4: Importar servicio virtualizado 1

5. Seleccionamos el archivo "TFG-Radio.ova" dentro del DVD y pulsamos "Next" para continuar con la importación.

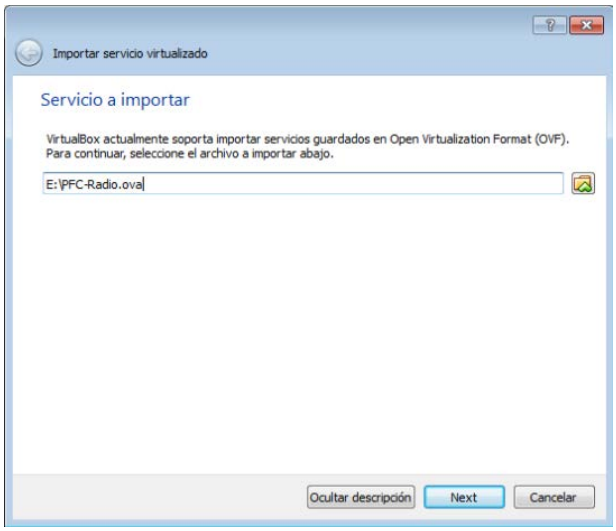

Figura A.5: Importar servicio virtualizado 2

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

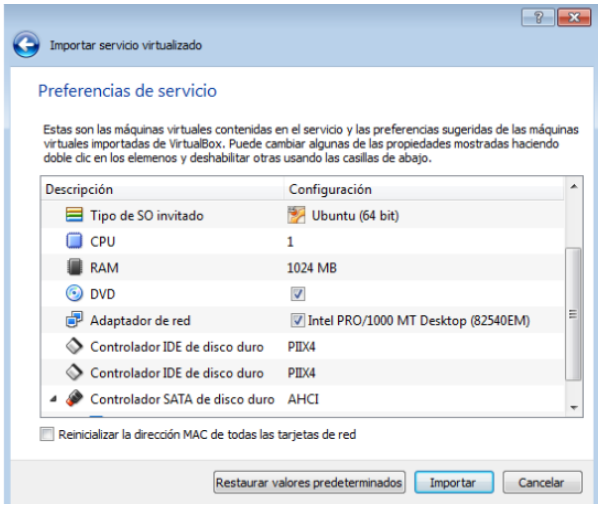

Figura A.6: Importar servicio virtualizado 3

7. Seguidamente nos aparecerá la nueva máquina virtual, ahora pulsando “Iniciar” la máquina arrancará.

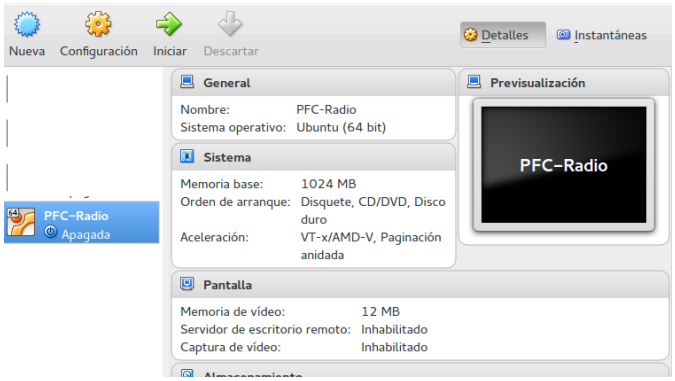

Figura A.7: Arrancar máquina virtual

8. Una vez dentro de Windows para desplegar nuestra aplicación en GlassFish, abrimos el navegador y tecleamos la siguiente URL:

http://localhost:4848/

9. En la siguiente pantalla que nos aparece pulsamos a la izquierda a aplicaciones y dentro de esta deploy:

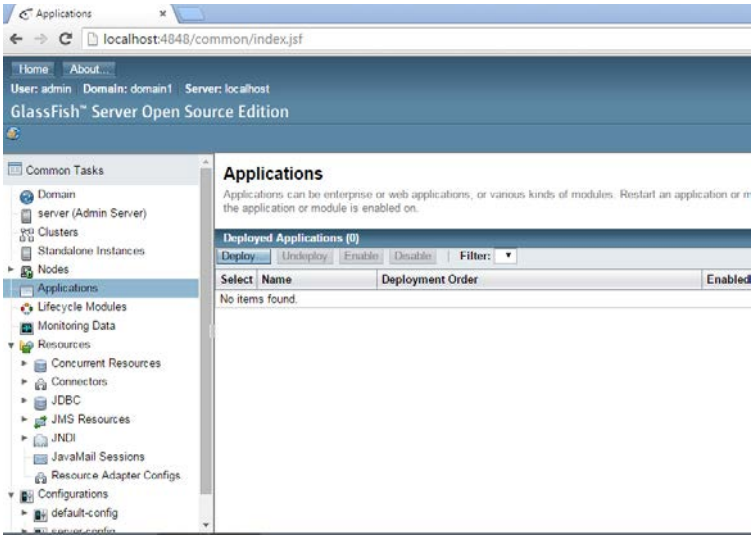

Figura A.8: Desplegando aplicación en Glassfish 1

10. A continuación pulsamos en "Choose file" y elegimos el archivo "radioWebPrueba.war" que se encuentra en la siguiente ruta "C:\Users\tfg\Documents\NetBeansProjects\RadioOnline\dist".

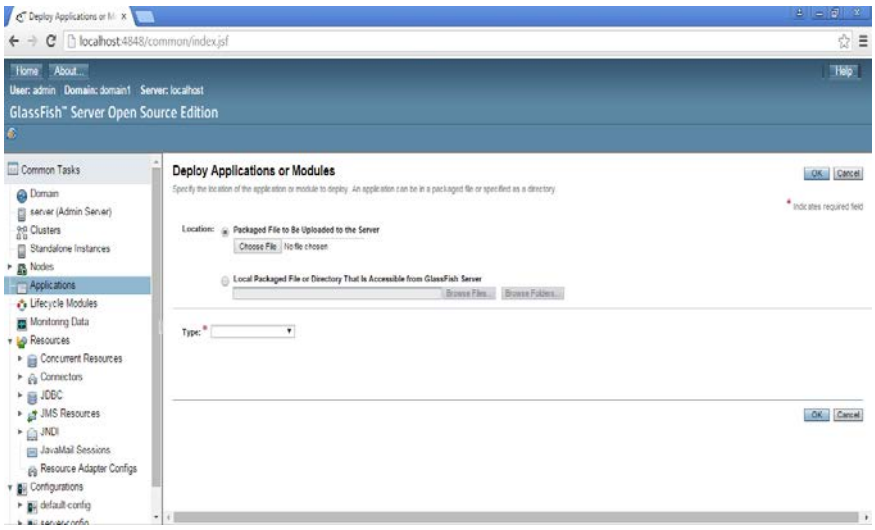

Figura A.9: Desplegando aplicación en Glassfish 2

11. Por ultimo seleccionamos “launch” y ya estará la aplicación lista para usarse

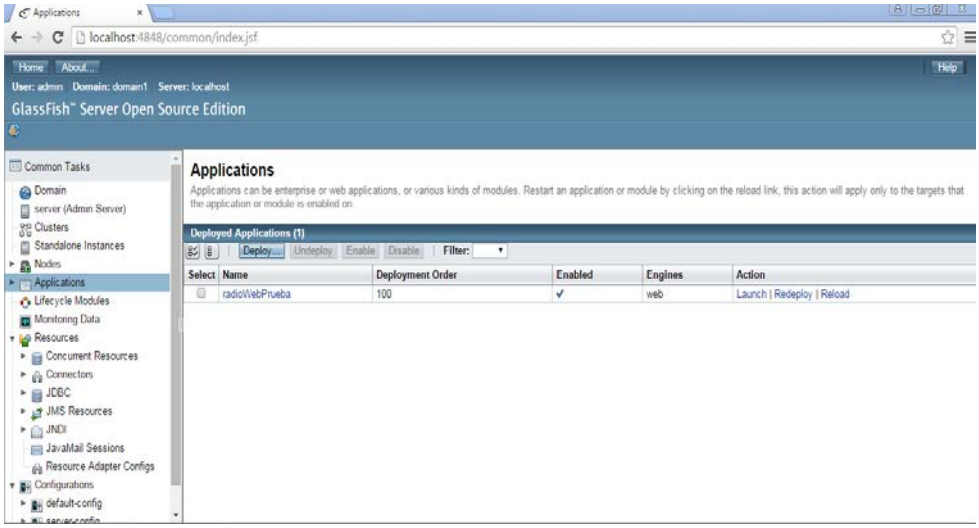

Figura A.10: Desplegando aplicación en Glassfish 3

Cliente

Al realizar las acciones descritas anteriormente la radio se montara sobre un host local al que se accede por el puerto predeterminado. Por lo que para la parte del cliente, al tratarse de una aplicación web, solo necesitamos un navegador e incluir en la barra de direcciones la siguiente URL:

http://localhost:8080/radioWebPrueba.

En esa URL se puede acceder a la aplicación y seguir todo el proceso explicado a continuación en el anexo de manual de usuario para su uso.

Para una mejor comprensión, se expone que la parte instalada en este anexo pertenece a la sección enmarcada de rojo en la figura A.11.

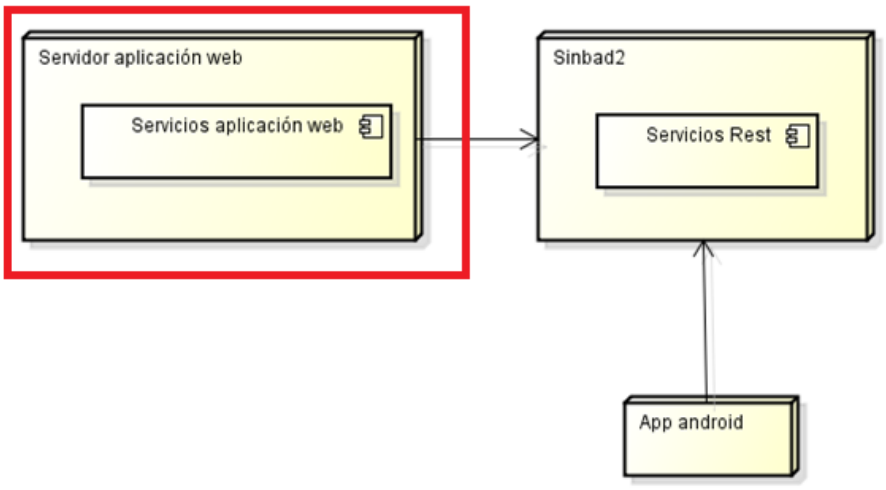

Figura A.11: Servidor implementado

Apéndice B:

Manual de usuario

Manual de usuario

En este manual de usuario se describen todas las acciones posibles a llevar a cabo en la aplicación; así como ilustraciones instruyendo paso a paso la interacción de un usuario con la misma.

Identificación

En la pantalla inicial, cuando aún no se encuentra el usuario identificado encontramos estas opciones:

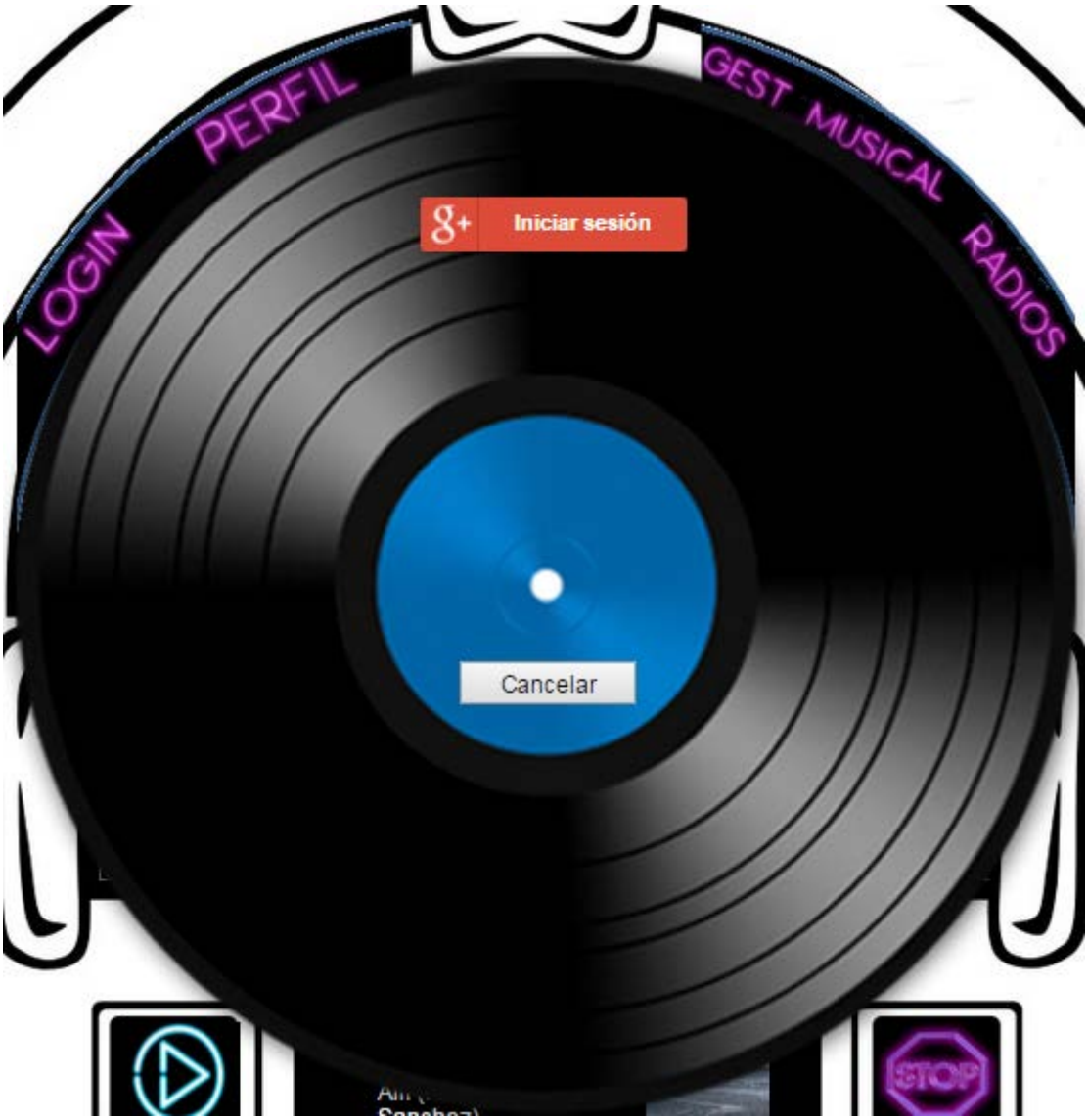

Figura B.1. Identificación

En esta sección se muestran cuatro opciones para el usuario; pero en realidad solo son un reflejo de las opciones que puede llevar a cabo una vez identificado. Al intentar acceder a cualquiera de estas opciones el sistema instará al usuario a identificarse.

El procedimiento de identificación se basa en perfiles de google; ya que existe una aplicación para Android que comparte perfiles con esta aplicación para que el usuario pueda tener la misma experiencia utilizando cualquiera de las dos alternativas.

Al hacer click en el botón rojo que se titula "iniciar sesión" nos enviará a la siguiente página para que introduzcamos nuestras credenciales, con el método usual de google, y así poder acceder a la sección para usuarios autentificados.

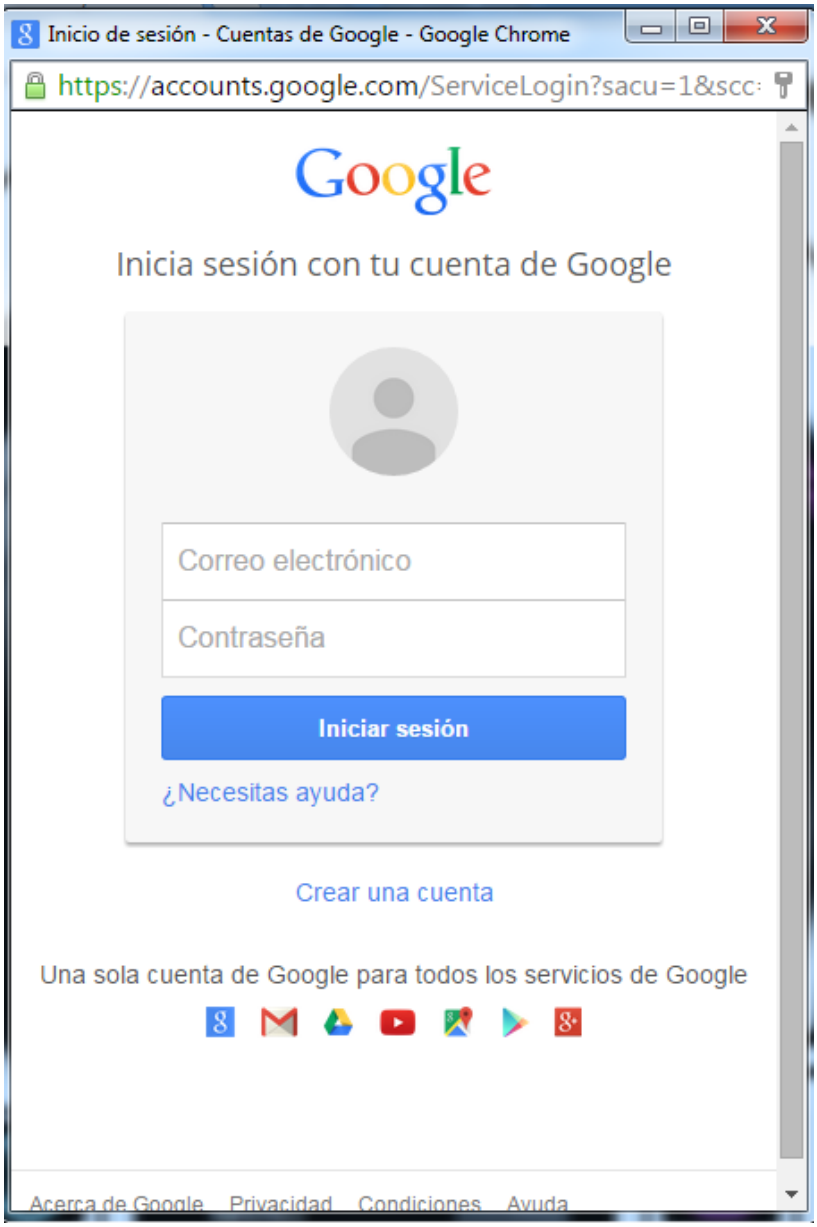

Figura B.2. Identificación google

Control de reproducción

Al entrar en la página de la aplicación, tanto si estas autentificado como si no, el reproductor de audio entra en funcionamiento. En ambas páginas se encuentra a la misma altura y posee la misma funcionalidad.

Figura B.3. Reproductor musical

Como se observa en la figura B.3, el reproductor consta de tres secciones:

- Control de flujo de reproducción.
- Sección de reproducción actual.
- Lista de reproducción.

Control de flujo de reproducción

Con control de flujo de reproducción nos referimos a los botones, fácilmente reconocibles de infinidad de aparatos electrónicos y reproductores digitales, cuyas funciones son iniciar la reproducción, detenerla, avanzar una canción y retroceder a la canción inmediatamente anterior.

Estos botones son la única parte interactiva con el usuario de esta sección.

Sección de reproducción actual

En la sección encuadrada entre los botones que controlan el flujo de reproducción se encuentra esta sección. En ella encontramos una animación a modo de ecualizador de fondo; esta animación está en movimiento cuando la reproducción está activa y se detiene cuando el usuario decide parar la reproducción, indicando al usuario si la reproducción esta activa o no.

Superpuesta a esta animación encontramos el nombre de la canción que se encuentra actualmente sonando y una imagen que ilustra dicho texto.

Lista de reproducción

Contenida en un cuadro de neón, rosa cuando el usuario no está identificado y azul cuando lo está, encontramos la lista de reproducción con el mismo formato que en la sección anterior, nombre de la canción e imagen, donde se muestran las canciones que se van a escuchar a continuación; mostrando dinámicamente cual es la prioridad de arriba abajo. Es decir, la primera canción de la lista será la siguiente en escucharse y así sucesivamente.

Redes Sociales

En ambas páginas, en la parte superior, se encuentran los botones de las redes sociales; cada uno identificado con su logotipo correspondiente.

Figura B.4: Enlaces redes sociales

Al hacer click en cada uno de los botones, la web, nos redirige hacia su correspondiente página en las redes sociales asociadas.

Cuenta

Una vez el usuario se ha identificado en la página y ha accedido a la sección para usuarios autentificados; podemos acceder a sus cuatro menús.

El primero de ellos es cuenta, donde se encuentran los elementos que se muestran a continuación.

Figura B.5: Menú Cuenta

En esta sección se muestra información sobre el usuario; como son su fotografía, nombre completo y sexo; siempre y cuando estén estos datos debidamente implementados en su cuenta Gmail.

En la parte inferior encontramos dos botones:

- Cerrar Sesión: Este botón sustituye los atributos de su perfil de usuario en esta sesión por otros por defecto, y se le devuelve a la primera pantalla como usuario no autentificado que es.

- Eliminar Cuenta: Como el botón anterior nos devuelve a la pantalla inicial, como usuario no identificado. Pero en esta ocasión el sistema desvincula sus puntuaciones de su perfil de usuario; de manera que la próxima vez que se autentifique no tendrá puntuaciones asignadas.

Perfil

Figura B.6: Menú Perfil

En esta sección el usuario puede re puntuar sus canciones puntuadas anteriormente; cambiando su valor asociado en su perfil.

Siempre se está cambiando la puntuación que se muestra en el menú.

Encontramos tres tipos de botones:

Flechas

Los botones con forma de flechas que encontramos, una en dirección hacia la derecha y otra hacia la izquierda, sirven para cambiar la canción a re puntuar.

Como resulta evidente la flecha hacia la derecha sirve para pasar a la canción siguiente, mientras que su homóloga para regresar a la canción anterior.

Corazones y Estrella

Como ya se explicó en la sección de las metáforas, los corazones sirven para mostrar el agrado de la canción hacia el usuario; mientras más corazones más agrado. La estrella en cambio sirve para marcar la canción como favorita o desmarcarla, en caso de que estuviera marcada de una interacción anterior.

Ambas metáforas cambian de estado al hacer click sobre ellas.

Enviar y Cancelar

Estos botones entran en interacción al hacer click sobre ellos. El botón Enviar, cuando el usuario ya ha cambiado las puntuaciones que considera oportunas, es pulsado; cambiando así las puntuaciones asociadas al perfil de usuario.

El botón Cancelar sin embargo, deshace todas las acciones realizadas por el usuario en esta sección anteriormente a ser pulsado. A continuación cierra el menú y deja al usuario en la página que se encontraba.

Gestión Musical

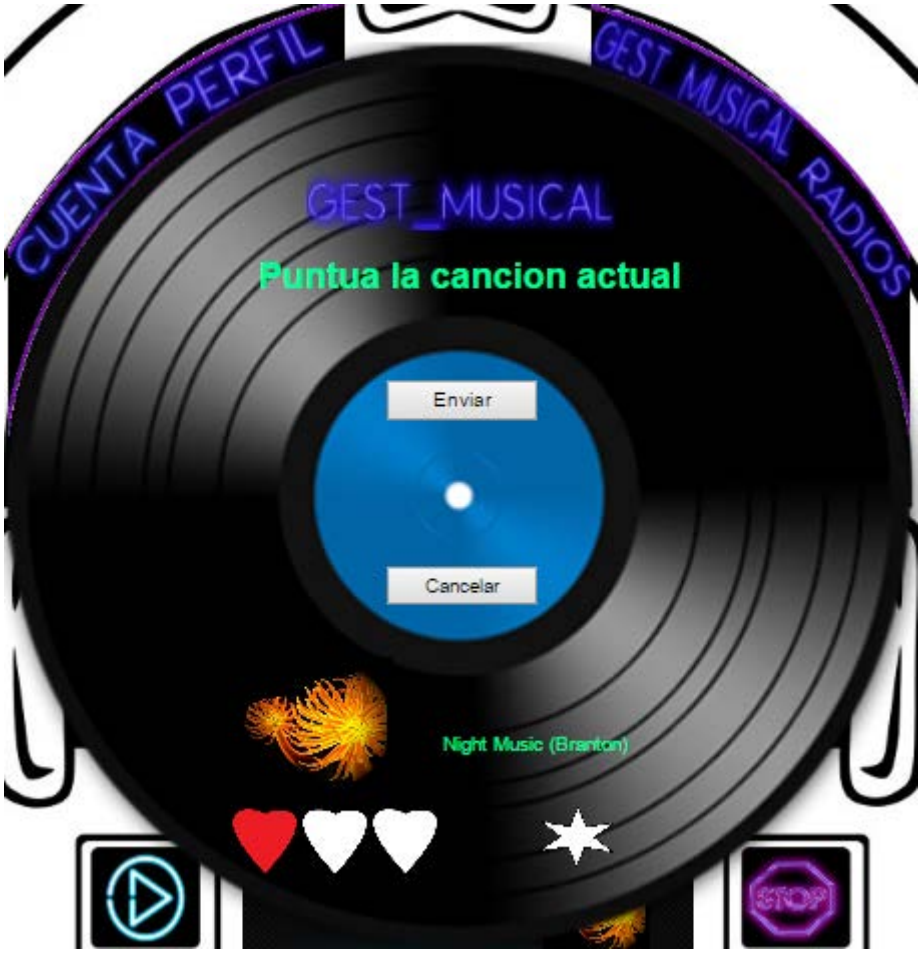

Figura B.7: Menú Gestión Musical

La funcionalidad de este menú es muy similar a la del menú anterior. La única diferencia es que en este caso se puntúa la canción que se está escuchando actualmente, que se muestra con su imagen y título dentro del menú

El comportamiento de las metáforas de corazones y estrella es idéntico al menú explicado anteriormente, sirviendo de igual manera para expresar el agrado del usuario hacia la canción que se encuentra sonando actualmente.

Al interactuar con el botón Enviar, la puntuación hace cambiar su valor asociado con el perfil usuario, mientras que al pulsar Cancelar, se cancela la puntuación y se cierra el menú.

Radios

Figura B.8: Menú Radios

En esta sección encontramos cinco botones:

- Radio Normal: Al pulsar en este botón la lista de reproducción se sustituye por otra constituida por canciones aleatoriamente seleccionadas de las existentes en el servidor de música

- Radio Personalizada: Al pulsar en este botón la lista de reproducción se sustituye por otra constituida por canciones recomendadas por el sistema de recomendación, del que se encuentra la radio dotada, basadas en puntuaciones sobre canciones realizadas con anterioridad.

- Radio Artistas favoritas: Al pulsar en este botón la lista de reproducción se sustituye por otra constituida por canciones pertenecientes a los artistas cuyas canciones, algunas de ellas, han sido seleccionadas como favoritas en una puntuación anterior en el tiempo.

- Radio Canciones Favoritas: Al pulsar en este botón la lista de reproducción se sustituye por otra constituida por canciones que han sido seleccionadas como favoritas en una puntuación anterior en el tiempo.

- Cancelar: Al pulsar en este botón se cierra el menú actual y se devuelve al usuario a la página anterior sin cambios en el estado de la misma

Para finalizar la documentación de este proyecto se procede a exponer la bibliografía y las referencias utilizadas durante el proyecto

Bibliografía y Referencias

[1] Creative Commons España, http://es.creativecommons.org. Último acceso: Junio 2015.

[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] José Luis Mateo: "Sociedad del conocimiento", ARBOR 2006

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

[16] R.R. Yager: "Fuzzy Logic Methods in Recommender Systems", Fuzzy Sets and Systems, Volume 136, Issue 2 (2003).

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

[18] Camarena Sagredo: "Ciencia Ergo Sum", vol. 19 (2012)

[19] Service (systems architecture), http://en.wikipedia.org/wiki/Service_(systems_architecture) . Último acceso: Junio 2015.

[20] SOA, http://www.slideshare.net/Mache007/arquitectura-orientada-a-servicios-soa-12818946. Último acceso: Junio 2015.

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

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

[23] Zagat, http://en.wikipedia.org/wiki/Zagat. Último acceso: Junio 2015.

[24] Foursquare, http://es.wikipedia.org/wiki/Foursquare. Ultimo acceso: Junio 2015.

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

[26] Pablo Andrés Barrientos: "Enfoque para pruebas de unidad basado en la generación aleatoria de objetos", 2014

[27] R. Pressman: "Ingeniería del Software. Un Enfoque Práctico." 7a Ed. McGraw-Hill, 2010

[28] TFG Radio Online Colaborativa para Android: http://sinbad2.ujaen.es/cod/archivosPublicos/pfc/pfc_francisco_moya.pdf

[29] BOE: Núm. 67 Sección III pag.24636. Resolución de 19 de Marzo 2015.

[30] Shawn Van Every: "Pro Android Media" Apress 2013

[31] Antonio Luís Cardador Cabello: "Desarrollo de aplicaciones web distribuidas" ic editorial 2015

[32] Stephen Stelting, Olav Maassen: "Patrones de diseño aplicados a Java" Pearson Educación, 2003

[33] Axure, http://www.axure.com/learn. Ultimo acceso: Junio 2015