Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

SISTEMA DE RECOMENDACIÓN MUSICAL CON CONTEXTO DE ESTADO DE ÁNIMO

Alumno: Iván Muñoz Troyano

Tutor: Prof. D. Luis Martínez López
Dpto: Departamento de informática

Septiembre, 2017

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Luis Martínez López , tutor del Proyecto Fin de Carrera titulado: Sistema de recomendación musical con contexto de estado de ánimo, que presenta Iván Muñoz Troyano, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Septiembre de 2017.

El alumno:

Iván Muñoz Troyano

Firmado digitalmente por Iván Muñoz Troyano
Nombre de reconocimiento (DN): cn=Iván Muñoz Troyano, o, ou, email=imt00011@red.ujaen.es, c=ES
Fecha: 2017.09.03 19:51:06 +02'00'

Iván Muñoz Troyano.

El tutor:

MARTINEZ LOPEZ LUIS - 26018033L

Firmado digitalmente por MARTINEZ LOPEZ LUIS - 26018033L
Nombre de reconocimiento (DN): c=ES, serialNumber=26018033L, sn=MARTINEZ LOPEZ, givenName=LUIS, cn=MARTINEZ LOPEZ LUIS - 26018033L
Fecha: 2017.09.04 11:13:53 +02'00'

Luis Martínez López.

Agradecimientos

En primer lugar, agradecer a mi familia los valores y educación que me han dado, por todos los esfuerzos que han realizado para que yo pudiera convertirme en la persona que soy hoy. Gracias por estar siempre a mi lado.

A mi pareja, que me ha acompañado durante todo este largo camino y ha sufrido sobre todo el proceso de realizar este TFG. Gracias a su inestimable ayuda, cariño y apoyo hasta el último momento, han hecho más fácil esta etapa.

A mis compañeros de Universidad, por todos los momentos vividos en estos años: clases, exámenes, prácticas, experiencias, risas… Me gustaría destacar a mi amigo Javier Martínez Jiménez, ya que nos conocimos al inicio de esta aventura y la vamos a terminar juntos.

A todos mis profesores de la carrera, ya que han sido partícipes de mi formación, fundamentalmente a mi tutor, Luis Martínez López, quien ha sido un ejemplo para mí por su profesionalidad y partícipe en la realización de este TFG por su dedicación, ayuda y consejo.

Por último y especialmente, a mi madre, que aunque no pueda verlo, espero que esté orgullosa de mi, ya que sin su apoyo no hubiera llegado hasta aquí.

Índice

Índice de ilustraciones

Índice de tablas

Capítulo 1:

INTRODUCCIÓN.

1. INTRODUCCIÓN.

El vertiginoso desarrollo de la tecnología ha irrumpido en la sociedad de tal modo que se ha convertido en uno de los principales productos de consumo en la actualidad. Con la llegada de Internet se ha favorecido la universalización de las relaciones y de la información que manejamos, con lo que podríamos afirmar que las tecnologías no están sujetas a fronteras.

Las tecnologías ya no son una simple herramienta de comunicación o trabajo, sino que a día de hoy son una de las causas fundamentales del cambio estructural de la sociedad. De hecho, los expertos dicen que estamos ante un nuevo tipo de sociedad, denominada *sociedad de la información* [37].

El acceso a Internet y el uso de los nuevos dispositivos móviles se ha extendido en los últimos años, llegando a una gran parte de la población. Como consecuencia de ello, han aparecido nuevas formas de entretenimiento como son por ejemplo los chats, los foros, los juegos y el modo en que consumimos la música. Debido a esto y gracias a las aplicaciones móviles, tenemos a nuestra disposición multitud de formas de disfrutar de la música.

Desde tiempos inmemoriales la música ha acompañado a la humanidad. El origen de la música podría remontarse a la prehistoria donde ya utilizaban la propia voz como instrumento para crear música. De hecho, la música ha estado presente a lo largo de toda la historia de la humanidad, en los acontecimientos más importantes, hechos históricos y celebraciones religiosas.

Hoy en día no tenemos el mismo acceso a la música como lo teníamos antes. Antiguamente las personas solamente conocían la música que se tocaba en directo en su entorno. Posteriormente, cuando la música comenzó a grabarse, la gente tenía acceso a una mayor oferta musical, aunque este medio era mucho más costoso y la música que se vendía era la que tenía una mayor demanda comercial. Actualmente, con la irrupción de Internet, tenemos acceso a una gran cantidad de música donde podemos encontrar desde la música más conocida hasta grupos que están comenzando su carrera artística, ofreciendo así a los consumidores una mayor diversidad de estilos musicales y canciones.

Una aplicación muy conocida en nuestros días como es Spotify, ofrece a los usuarios un extenso catálogo que está en torno a los treinta millones de canciones.

La sobrecarga de canciones disponibles en el momento actual hace que sea muy difícil encontrar canciones nuevas que se adapten a nuestros gustos, ya que la oferta es muy variada. Por eso, en los últimos años, se está produciendo un mayor auge de los sistemas de recomendación, ya que estos nos ayudan a explorar todo ese amplio catálogo de canciones de una manera más ajustada a nuestras necesidades.

Los sistemas de recomendación son muy útiles para realizar el filtrado de la gran cantidad de información que manejamos hoy en día. Un sistema de recomendación musical permite presentar a los usuarios las canciones que se adapten a su perfil particular, además de ofertar la gran cantidad de canciones disponibles en torno a diversas categorías.

No siempre toda la música que nos gusta cubre con nuestras necesidades del momento. Hay veces, por ejemplo, que necesitamos motivarnos y si nos ponen canciones con un ritmo más lento, aunque nos gusten, éstas no llegarán a animarnos. Con ello queremos hacer referencia a que la música puede llegar a cambiar nuestro estado de ánimo en un segundo y lograr que un mal día pase a ser mejor.

De hecho, la música se utiliza con fines psicoterapéuticos [38], como ocurre con la musicoterapia, la cual se orienta a la mejora de la calidad de vida de las personas a través de la misma. Otros ejemplos de ello son que en los eventos deportivos ofrecen música como incentivo, o que en las guerras se empleaban distintas canciones o himnos para desarrollar el coraje y confianza en los soldados.

Por todo lo anterior, en los sistemas de recomendación musicales es importante que se pueda introducir el estado de ánimo que se quiera mantener o conseguir en los usuarios.

1.1. Motivación.

Mi motivación para la realización de este TFG, es la de diseñar y desarrollar un servicio web en el cual se pueda recomendar música, no solo teniendo en cuenta los gustos o preferencias musicales, sino también atendiendo al estado de ánimo que queramos conseguir.

Para poder acceder a estas recomendaciones, también se va a diseñar y desarrollar una aplicación móvil para Android, la cual nos permitirá consultar tanto las distintas modalidades de recomendación, como la propia información de las canciones.

La comunicación entre la aplicación móvil y el servicio web se basará en el consumo de servicios API REST, ya que suele ser el patrón más elegido para el intercambio y la manipulación de datos en los servicios de Internet. De esta forma dotaremos a los usuarios de un acceso a estas recomendaciones en cualquier momento y en cualquier lugar.

1.2. Objetivos.

1. Búsqueda y revisión bibliográfica.
2. Estudio y análisis de los sistemas de recomendación.
3. Estudio e implementación de un servidor web con acceso a través de una API REST.
4. Estudio y desarrollo de aplicaciones Android.
5. Redacción de la memoria.
6. Redacción del manual de instalación.
7. Reacción del manual de usuario de la aplicación móvil.

1.3. Estructura de la memoria.

En esta fase final de la introducción de esta memoria, pasaremos a realizar un breve resumen de cada uno de los capítulos en los que se estructura este proyecto.

Como ya hemos visto, el primer capítulo se corresponde con la introducción de este proyecto TFG, comentamos la motivación que nos lleva a realizarlo y qué objetivos necesitamos cumplir para la realización del mismo.

El segundo capítulo se dedica a los sistemas de recomendación, en éste hemos comentando de manera general los mismos para, en un paso posterior, poder hablar sobre las distintas técnicas y su funcionamiento de una manera más específica, terminando con la justificación del sistema de recomendación escogido.

El siguiente capítulo, el número tres, se centrará en los servicios web, donde revisaremos las tecnologías que se usan y profundizaremos en los servicios REST, que son los elegidos para este proyecto.

En el capítulo 4, obtendremos una visión de Android, que es el sistema operativo elegido en este TFG. Se introducirán una serie de nociones teóricas sobre su historia, la arquitectura, los componentes y la seguridad, entre otros temas.

A partir de este punto dejamos a un lado la parte más teórica del proyecto para centrarnos en la realización del mismo. En los siguientes capítulos nos centraremos en el proceso de Ingeniería del Software. Así, el capítulo 5 se dedica a la etapa del análisis, donde sentaremos las bases de la estructura y objetivos del proyecto. A continuación, el capítulo 6 está centrado en el diseño de la aplicación, donde iremos aplicando las distintas técnicas para diseñar la base de datos, la interfaz con sus storyboards y los caminos de navegación para dar así una visión de cómo será la aplicación. Finalmente, para terminar con el proceso de Ingeniería del Software, en el capítulo 7 hablaremos sobre la implementación y pruebas del proyecto: cómo es su arquitectura, qué lenguajes han sido utilizados y la manera de comunicarse entre el servicio web y la aplicación Android. También se definen las pruebas que se utilizarán para validar la implementación.

En el octavo y último capítulo expondremos las conclusiones y analizaremos los objetivos cumplidos, añadiendo información sobre futuras mejoras que se pueden realizar.

Por último, podremos encontrar la bibliografía consultada para la realización de este proyecto, así como dos anexos, uno donde se explica la instalación del servidor y la aplicación móvil, y otro donde se detalla el manual de usuario para la utilización de la aplicación Android.

Capítulo 2:

SISTEMAS DE RECOMENDACIÓN.

2. SISTEMAS DE RECOMENDACIÓN.

En los tiempos actuales, la sobrecarga de información se hace cada vez más notable a medida que se tiene acceso a más y más fuentes de datos. Todos los días, los usuarios se enfrentan al reto de elegir una aplicación para su dispositivo móvil o escoger una buena película para ver el fin de semana entre miles y miles de opciones.

El auge de los smartphones en la última década, ha hecho que la sociedad se haya acostumbrado a proporcionar un gran volumen de datos acerca de ellos, sus opiniones, ideas, sus gustos, intereses, su geolocalización e infinidad de datos más, lo cual nos es muy útil para poder facilitarles una información personalizada sobre casi cualquier cuestión[8].

Estos son los principales motivos por los que los sistemas de recomendación se han hecho muy populares en los últimos años. Entre los ámbitos más extendidos se encuentran las recomendaciones de productos en tiendas online, películas, vídeos, música, libros, productos o recomendaciones de perfiles a los que seguir en redes sociales.

Los Sistemas de Recomendación (SR) podemos definirlos como un *conjunto de herramientas y técnicas que pueden proporcionarnos sugerencias sobre una serie de elementos o ítems que sean de interés para el usuario* [1, 2].

Los SR tienen como objetivo reducir el esfuerzo cognitivo del usuario al estudiar patrones de comportamiento que permitan predecir las posibles elecciones que realizaría una persona entre un conjunto de ítems con los que no se tiene una experiencia previa, ya que dicho usuario puede no querer evaluar o no tener conocimientos suficientes para evaluar la gran cantidad de ítems diferentes que tiene a su disposición.

Los ítems es el término general usado para hacer referencia a los elementos que los servicios de recomendación van a sugerir a los usuarios. Comúnmente estos elementos suelen ser de un tipo específico como puede ser una canción, un libro, un lugar…

Las recomendaciones se suelen mostrar como listas de ítems ordenados o rankings de objetos. Este modo de presentación suele ser la manera más simple, aunque existen otras.

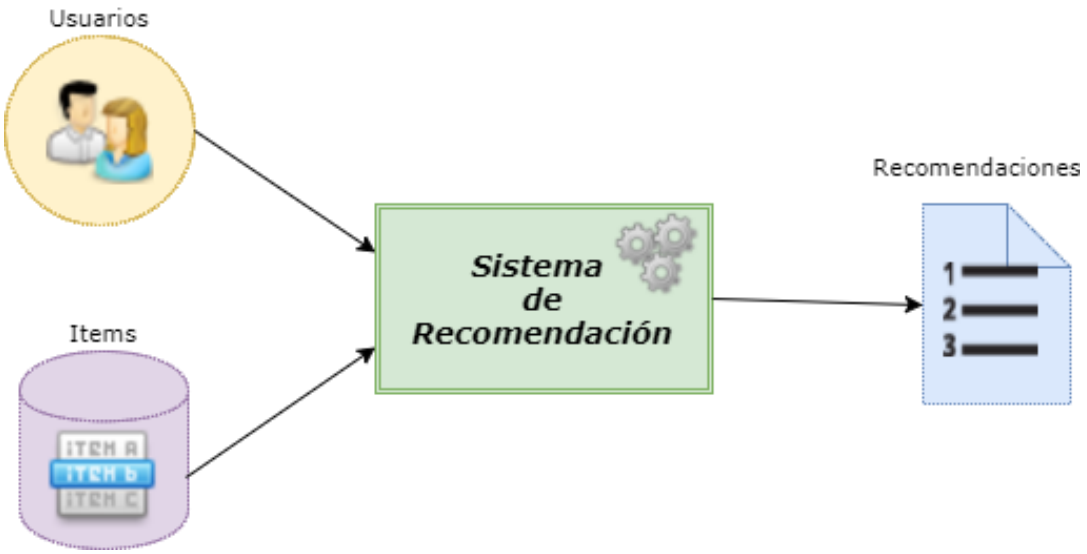

Ilustración 2.1 Esquema simple de un sistema de recomendación.

Para que los SR puedan realizar su tarea, la cual consiste en sugerir productos o servicios que sean útiles para el usuario, es necesario que se basen en los gustos o preferencias de los mismos. Estas pueden ser obtenidas de manera explícita, como puede ser el caso de las valoraciones, o de manera implícita, como puede ser inferidas del comportamiento o acciones de los propios usuarios.

El usuario puede comprobar las recomendaciones, puede aceptarlas o no, y puede conseguir, de manera inmediata o en una fase posterior, una retroalimentación. Estas acciones y retroalimentaciones se almacenan en la base de datos del SR y pueden utilizarse para generar nuevas recomendaciones en las próximas interacciones.

En los últimos años, los SR están demostrando ser una herramienta muy útil para ayudarnos a tratar la sobrecarga de información, sobre todo, ayudando a los usuarios a descubrir nuevos ítems que no conocen y que pueden ser importantes para sus propias necesidades.

Es importante destacar algunas empresas que desarrollan y utilizan estos SR para sus sitios web, como son:

- Amazon
- Netflix
- YouTube
- TripAdvisor
- Last.fm
- IMDB

El número de empresas que implementan SR como servicios que ofrecen a sus usuarios está en aumento. Las motivaciones que llevan al uso de estos sistemas son las siguientes [5]:

- *Incrementar el número de ítems vendidos:* A nivel comercial, ésta seguramente será la motivación más importante para los SR comerciales. Esto es posible porque los ítems recomendados suelen encajar con las necesidades y los deseos del usuario.

- *Vender ítems más diversos:* Otra función principal en un SR es sugerir al usuario ítems que posiblemente nunca se le hubieran ocurrido o hubieran sido difíciles de encontrar sin una precisa recomendación. Por ejemplo, en un SR de vídeos como YouTube, se le podría sugerir un vídeo no muy conocido pero que cumpla con las preferencias y restricciones del usuario.

- *Incrementar la satisfacción del usuario:* Un SR bien diseñado puede mejorar la experiencia del usuario en la aplicación o página web. Al tener un usuario contento seguramente ayudará al SR a mejorar sus recomendaciones, porque éste lo usará más.

- *Incrementar la fidelidad del usuario:* Un usuario será fiel a un sitio web si las recomendaciones obtenidas son de su interés. Esto hará que lo tenga como una página web de referencia, aumentando las iteraciones con el sistema y mejorando el modelo de recomendación.

- *Mejor entendimiento de lo que el usuario busca:* Otra función importante de un SR, es que toda la información recogida puede utilizarse para otras aplicaciones y puede reutilizar su conocimiento para ofrecer nuevos servicios a sus usuarios.

2.1. Elementos básicos de los sistemas de recomendación.

Los SR son sistemas de procesamiento de la información que están continuamente evaluando distintos tipos de datos con la finalidad de mejorar sus modelos de recomendación. Estos datos son frecuentemente sobre los usuarios a los que hay que generar sugerencias y sobre los ítems que tienen que recomendar.

Los datos necesarios por los SR hacen referencias a los siguientes tipos: ítems, usuarios y transacciones, siendo las transacciones información que relaciona a los usuarios con los ítems, como pueden ser las valoraciones.

- ***Ítems:***

  Son los elementos que van a ser sugeridos y se caracterizan por su utilidad, valor y/o complejidad. El valor asignado a estos ítems puede ser positivo o negativo, según si el elemento es apropiado o no para el usuario.

  Los ítems pueden presentarse utilizando distintos enfoques de información y representación: utilizando un sentido simple, como puede ser un identificador, o algo más complejo, como un conjunto de atributos, o como un concepto de una representación ontológica del dominio.

  Según su complejidad, podemos ver los siguientes ejemplos de ítems:

  - Complejidad baja: Sitios web, música, noticias, películas, libros.
  - Complejidad media: Smartphones, ordenadores, televisores, electrodomésticos.
  - Complejidad Alta: Decisiones médicas, trabajos, viviendas, inversiones financieras.

- ***Usuarios:***

  Como hemos visto antes, cada usuario posee características y objetivos distintos, por lo cual se pueden personalizar las recomendaciones. Gracias a los SR se puede usar cierta variedad de información.

  Cada usuario tendrá su propio modelo, el cual codificará sus preferencias, necesidades y restricciones. Estos modelos pueden ser generados directamente por sus puntuaciones a los ítems, o se pueden utilizar estas puntuaciones para inferir un vector de valores factor.

  Por tanto, un SR podría usar esta información para recomendar ítems a usuarios que fueron preferidos por usuarios similares o de confianza.

- ***Transacciones:***

  Las transacciones hacen referencia a la interacción registrada entre un usuario y el SR. Son útiles para el funcionamiento del algoritmo de generación de recomendaciones.

  Las valoraciones son la forma más común de transacción que utilizan los SR. Estas valoraciones se pueden obtener de manera explícita o implícita.

  - Datos explícitos: Son las puntuaciones solicitadas al usuario sobre un ítem específico dentro de una escala. Las valoraciones pueden obtenerse de diversas formas:
    - Valoraciones numéricas, por ejemplo en una escala del 1 al 10.
    - Valoraciones ordinales, como pueden ser “muy de acuerdo, de acuerdo, neutral, en desacuerdo”.
    - Valoraciones binarias, tales como elegir entre si te gusta o no algo.
    - Valoraciones unarias, como pueden ser el haber adquirido un ítem o no.

- Datos implícitos: El sistema tratará de inferir los gustos y preferencias de los usuarios basándose en las acciones y comportamiento de los mismos; por ejemplo, si un usuario escucha varias canciones de un género se puede inferir que ese género le gusta.

2.2. Técnicas de recomendación.

2.2.1. Recomendación basada en contenido.

Los SR basados en contenido son aquellos que utilizan las características de los ítems. Por lo tanto, son los sistemas que se basan en los modelos que se han realizado por el análisis que efectúa el usuario de los productos.

En definitiva, los SR basados en contenido analizan las características de los ítems y los comparan con los generados por el usuario para poder predecir que estos ítems cumplen con las necesidades del propio usuario. Básicamente es recomendar productos similares a los ya consumidos por los usuarios y que se saben que son del agrado del mismo.

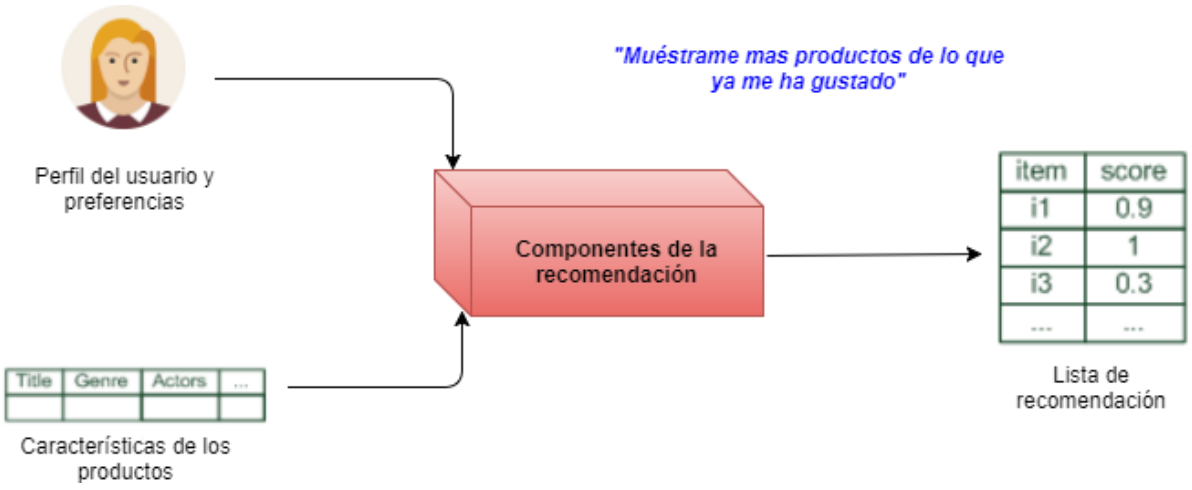

Ilustración 2.2 Esquema básico de recomendación basada en contenido.

Los SR basados en contenido se comprenden de dos fases: una para analizar las valoraciones que ha realizado el usuario, y otra para predecir qué productos se asemejan a sus gustos para ser recomendados.

Uno de los problemas que podemos encontrar en esta técnica es la sobre-especialización, ya que nos recomendará sobre nuestras propias preferencias, pero a veces estas preferencias pueden cambiar y el modelo generado ya no sería válido. También nos encontramos el problema de que cuando tenemos un usuario nuevo, al no tener transacciones de éste, el SR se ve incapaz de generar un modelo para dicho usuario. Por eso en muchos sitios web, en la fase de registro, nos preguntan sobre nuestros gustos, para poder solventar así dicho problema.

2.2.2. Recomendación basada en conocimiento.

Los SR basados en conocimiento utilizan una base de conocimiento donde están descritos los distintos productos y de qué manera satisfacen las necesidades de los usuarios. Por consiguiente, el sistema encuentra uno o varios productos que cumplen con las necesidades que ha solicitado.

Esta búsqueda de ítems a partir de las necesidades que ha especificado el usuario se realizará mediante un proceso de inferencia, permitiendo a estos sistemas obtener y almacenar la información de distintas formas.

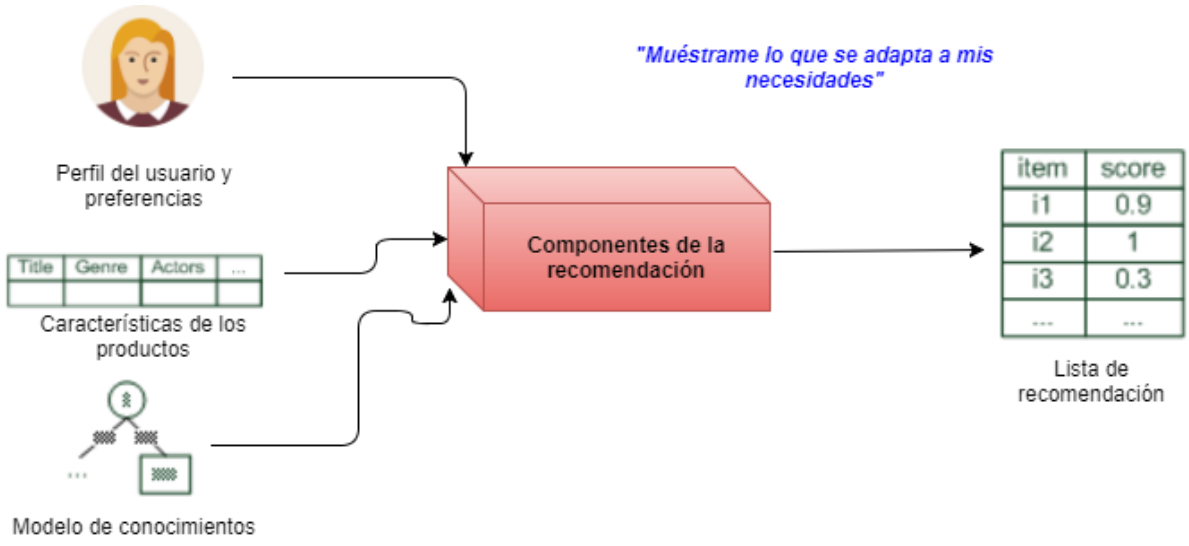

Ilustración 2.3 Esquema básico de recomendación basada en conocimiento.

Para que el funcionamiento de estos sistemas sea válido cuando se tiene poca información sobre los usuarios, es necesario tener un conocimiento adicional sobre el entorno en el que se está utilizando, sin olvidar también de tener la suficiente información sobre las características de los productos.

2.2.3. Recomendación con filtrado colaborativo.

Los SR con filtrado colaborativo emergieron a mediados de los años 90 y se basan en el concepto del “boca a boca”, ya que estos sistemas calculan las recomendaciones para el usuario utilizando toda la información obtenida de la interacción de los demás usuarios del sistema.

Si queremos tener un SR con filtrado colaborativo de garantía, es muy importante analizar bien las necesidades de nuestro problema para la elección correcta del algoritmo de filtrado colaborativo, ya que podemos encontrar dos categorías: los algoritmos basados en memoria o usuarios y los algoritmos basados en ítems.

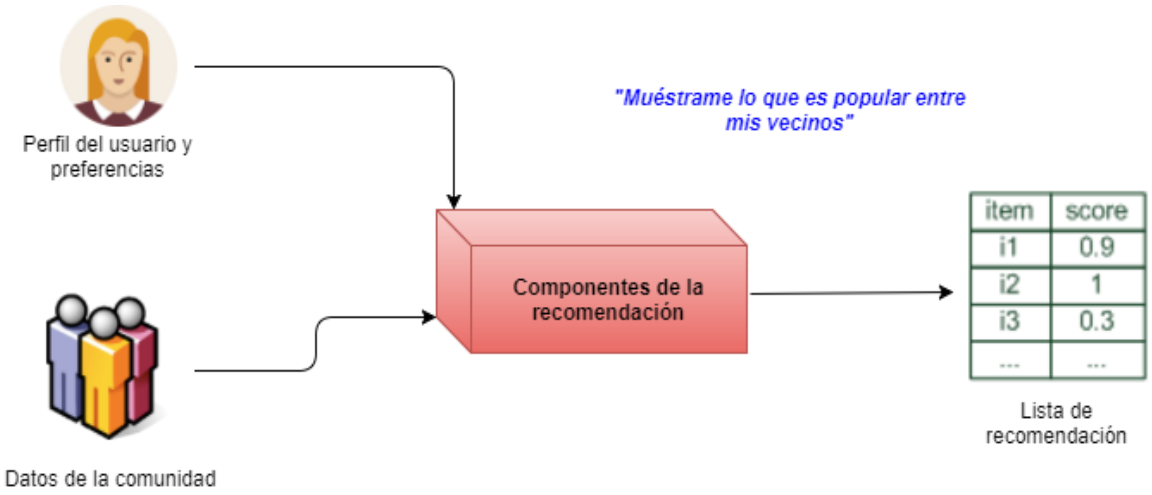

Ilustración 2.4 Esquema básico de recomendación con filtrado colaborativo.

El funcionamiento de los SR con filtrado colaborativo se puede dividir en tres fases, que son las siguientes:

1. Cálculo de vecinos cercanos: Consiste en la elección de los usuarios con preferencias y necesidades más parecidas al usuario que se le quiere realizar la recomendación.

Uno de los métodos más comunes para calcular la similitud de un usuario con sus vecinos es el coeficiente de correlación de Pearson, el cual calcula la dependencia entre dos variables:

$$sim(x,y) = \frac{\sum_{i \in P}(r_{x.i} - \overline{r}_x)(r_{y.i} - \overline{r}_y)}{\sqrt{\sum_{i \in P}(r_{x.i} - \overline{r}_x)^2 \ \sum_{i \in P}(r_{y.i} - \overline{r}_y)^2}}$$

2. Predicción de la valoración de preferencia: Se realiza una predicción para calcular el valor que el usuario otorgaría a cada uno de los productos que no ha valorado.

   Para calcular la predicción podemos elegir la técnica llamada Weighted Sum, la cual calcula la predicción de un producto por parte de un usuario como la suma de las valoraciones de dicho usuario sobre productos similares:

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

3. Se obtiene la recomendación: Realizadas las dos etapas anteriores, ya podemos ordenar la lista de ítems recomendados según su valor de preferencia, y se devuelve los N mejores resultados.

Los SR con filtrado colaborativo son de los sistemas más populares y utilizados hoy en día, ya que tienen muchas ventajas frente a otros sistemas. Aunque también tienen sus desventajas como son, por ejemplo, que necesitan una gran cantidad de datos para funcionar con calidad, sufren problemas de escalabilidad o que tienen problemas de arranque en frío. Éste último problema se puede presentar de dos formas más, ya que necesitamos un número suficiente de valoraciones de preferencias del usuario, y también, que se puede producir a nivel de los ítems, ya que si se introduce uno nuevo se necesitan valoraciones sobre éste para que sea recomendado.

2.2.4. Recomendación híbrida.

Los SR híbridos están basados en la combinación de dos o más técnicas de recomendación. Como hemos podido ver, todas las técnicas de recomendación tienen sus ventajas y desventajas, por eso la combinación puede ayudar a cubrir los inconvenientes de las otras técnicas, y viceversa.

Por ejemplo, como hemos visto anteriormente, cuando utilizamos los SR con filtrado sufrimos el problema del ítem nuevo. Este problema no está en los SR basados en contenido, por eso podemos cubrir ese inconveniente de la primera técnica con el uso de la segunda.

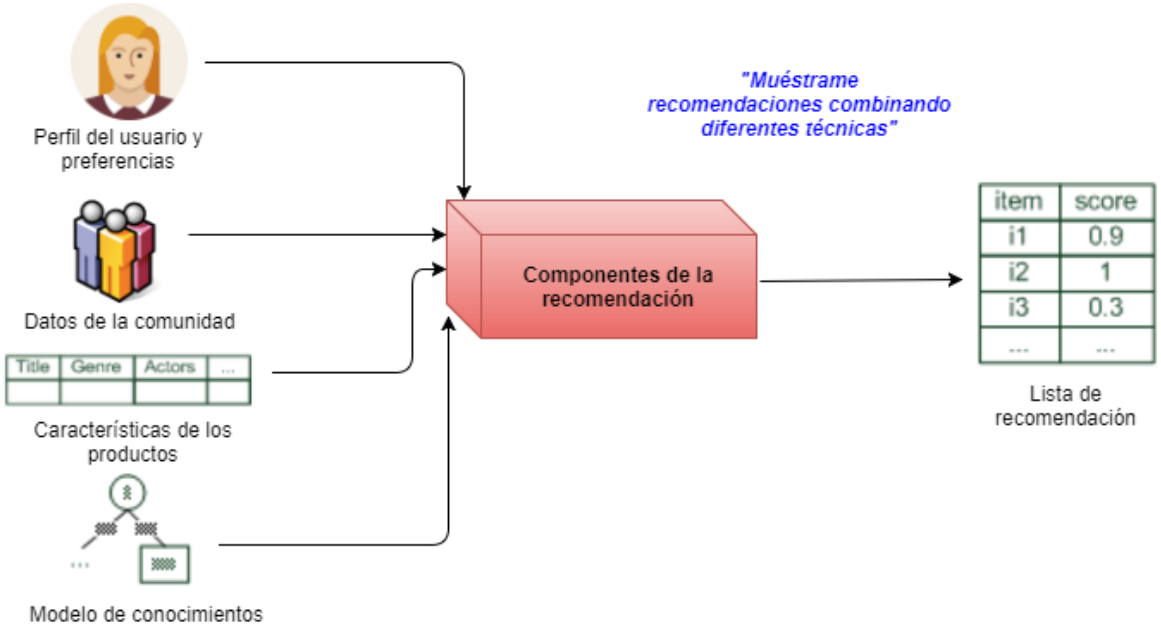

Ilustración 2.5 Esquema básico de recomendación híbrida.

2.2.5. Recomendaciones sensibles al contexto.

Los SR sensibles al contexto se usan en dominios donde no todos los productos son útiles para el usuario en cualquier momento y situación. Estos sistemas utilizan la información contextual de los usuarios para poder ajustar las recomendaciones a las necesidades de una forma más útil. En estos momentos, estos sistemas se están implantando para mejorar las recomendaciones [2, 6].

El contexto, como tal, no tiene una definición simple y concisa, sino que depende del dominio en el que se está utilizando el SR. A continuación mostraremos algunos ejemplos [15]:

- En el comercio electrónico, el contexto podría ser con la intención con la que un usuario realiza una compra.
- En minería de datos, el contexto podría ser los eventos que caracterizan un ciclo de vida de un cliente.
- En decisiones médicas, el contexto podría ser la edad, los antecedentes.
- En computación ubicua, el contexto se podría entender como la ubicación del usuario, la identidad de las personas, los objetos de alrededor.
- En paquetes vacacionales, podría ser el clima deseado, temporada…

Para introducir la información contextual en las recomendaciones, podemos utilizar estos dos enfoques [15]:

- **Pre-filtrado:** Con este enfoque se realiza una exploración de los datos de entrada eliminando los que no se consideran útiles por el contexto para que el SR no pueda recomendar esos ítems. Por ejemplo, en un SR de lugares de interés, podemos utilizar la localización de los usuarios para recomendar lugares de interés cercanos a él, con lo cual seguro que será más fácil que vaya a visitarlos.

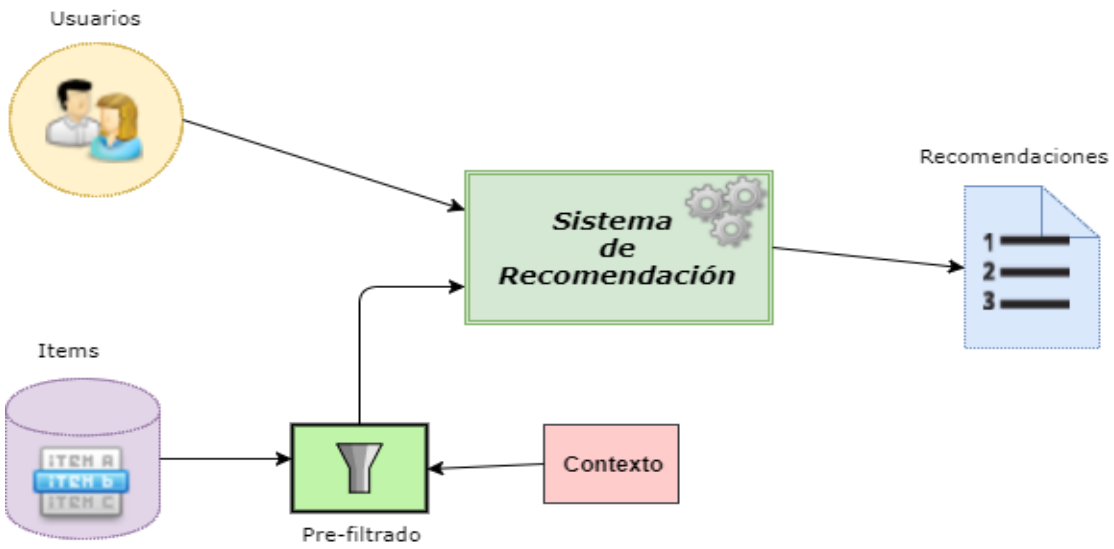

Ilustración 2.6 Esquema de recomendación con pre-filtrado.

- **Post-filtrado:** Al contrario que el pre-filtrado, la información contextual es aplicada después de calcular las recomendaciones. Una vez obtenida la lista de las recomendaciones, podemos reordenar los elementos por la información contextual y también podría eliminarse el ítem que por el contexto no satisfaga las necesidades del usuario.

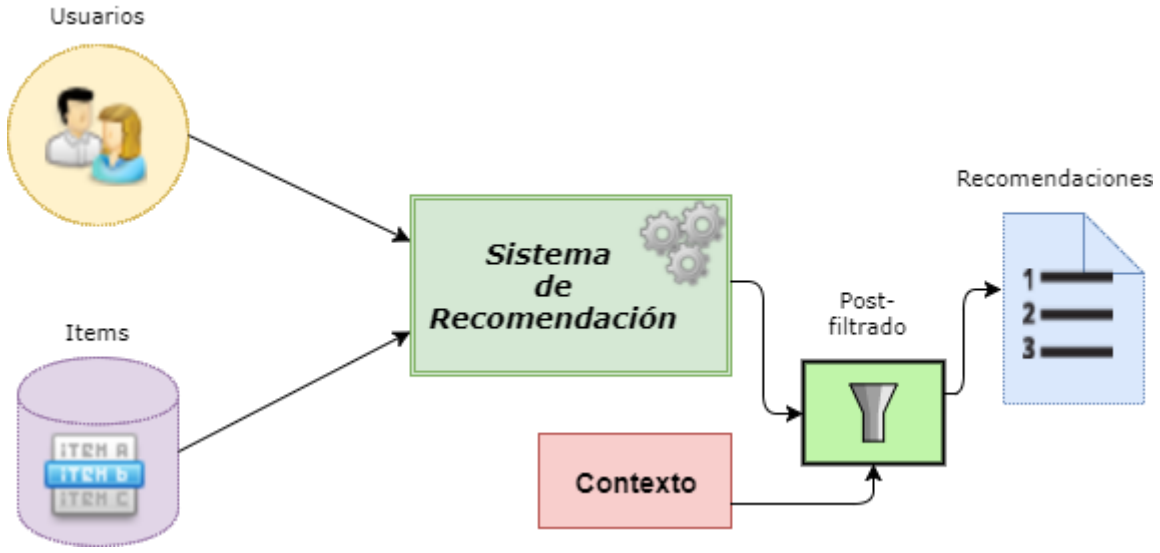

Ilustración 2.7 Esquema de recomendación con post-filtrado.

En este proyecto TFG se ha utilizado la técnica de filtrado colaborativo para la recomendación de canciones, ya que es una de las técnicas más implementadas y valoradas actualmente, con la técnica de post-filtrado para realizar recomendaciones sensibles al estado de ánimo; ya que, aunque una canción nos guste, puede que en el estado de ánimo actual no nos apetezca escucharla o incluso queramos que nos recomiende canciones de algún estado en concreto para intentar alcanzarlo, como podría ser escuchar canciones positivas cuando estamos tristes y queremos alegrarnos.

Capítulo 3:

SERVICIOS WEB.

3. SERVICIOS WEB.

El concepto de servicio web es complejo, pero a grandes rasgos podemos definir un servicio web como *un conjunto de aplicaciones o tecnologías interrelacionadas entre sí para ofrecer un servicio, utilizando para ello la web* [16]. Los usuarios o clientes demandan un servicio determinado y los proveedores permiten ofrecer dichos servicios a través de la web.

En el ámbito del proyecto que nos ocupa, hacemos referencia a los servicios web debido a que es necesaria de una comunicación entre el usuario y el servidor a través de Internet.

Los servicios web suelen ser considerados como APIs Web a los que se puede acceder dentro de una red, fundamentalmente Internet, y ejecutados en el sistema que los aloja.

Los servicios web se caracterizan por ser interoperables, superan las barreras geográficas, son flexibles por naturaleza y se basan en el protocolo HTTP.

En la ilustración 3.1 podemos ver de forma esquemática cuál es el funcionamiento de un servicio web:

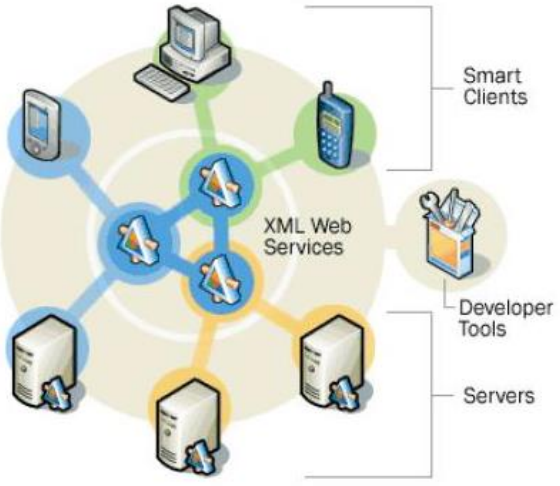

Ilustración 3.1 Esquema de funcionamiento de un servicio web.

3.1. Ventajas e inconvenientes de los servicios web.

En las siguientes líneas vamos a hacer una descripción de ventajas y desventajas más destacables que presentan los servicios web, las cuales es fundamental conocer a la hora de utilizar un servicio web [20].

3.1.1. Ventajas de un servicio web.

- Permiten operar entre distintas aplicaciones de software sin importar sus propiedades específicas o las propiedades de las plataformas sobre las que se instalen.

- Los servicios web fomentan los estándares y protocolos basados en texto, de tal manera que facilitan el acceso a su contenido y ayudan a entender su funcionamiento.

- Ofrecen la posibilidad de combinar fácilmente los servicios y el software de diferentes compañías ubicadas en diferentes lugares geográficos para proveer servicios integrados.

3.1.2. Inconvenientes de un servicio web.

- Los servicios web son programas conectados entre sí pero que no interactúan directamente con el usuario, es decir, no ofrece una interfaz de usuario para el usuario que demanda los servicios, tal y como podría proporcionarlo un servidor de páginas web. Por lo tanto, es una tarea fundamental del programador el realizar una interfaz de usuario para la conexión entre el usuario y estos programas.

- Los estándares abiertos de los servicios web tienen un nivel de desarrollo mucho más precario que los proporcionados por los estándares abiertos de computación distribuida, como por ejemplo

CORBA (Common Object Request Broker Architecture), a la hora de realizar comunicaciones entre el servicio web y el cliente que demanda el servicio.

- No tienen un rendimiento demasiado alto en comparación con otros modelos tales como RMI, CORBA o DCOM. Este inconveniente proviene fundamentalmente de la utilización de XML, ya que entre los objetivos de la utilización de este estándar no se encuentra el de mejorar la eficacia del procesamiento.

- Pueden esquivar medidas de seguridad basadas en cortafuegos debido a que están basados en HTTP, cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera.

3.2. Tecnologías utilizadas en los servicios web.

Existe una gran variedad de tecnologías a nuestra disposición a la hora de implementar un servicio web, entre todas ellas destacan fundamentalmente SOAP y REST, ya que actualmente son dos tecnologías muy utilizadas [10].

Para la realización de este proyecto hemos tomado como base la tecnología REST, es por ello por lo que a continuación pasamos a hacer una descripción más detallada de la misma.

3.2.1. REST (Representational State Transfer).

Originariamente el término REST hacía referencia a un conjunto de principios de arquitectura, aunque en la actualidad es utilizado, en un sentido más amplio, para describir cualquier interfaz web simple que utiliza XML y HTTP.

Dicho de otro modo, el servicio web REST es un estilo de arquitectura de software dirigido a sistemas hipermedias distribuidos y hace referencia a una colección de principios, los cuales resumen la forma en que los recursos son

definidos y diseccionados, para el diseño de arquitecturas en red. El funcionamiento de REST podemos verlo de modo sintetizado en la ilustración 3.2 [17]:

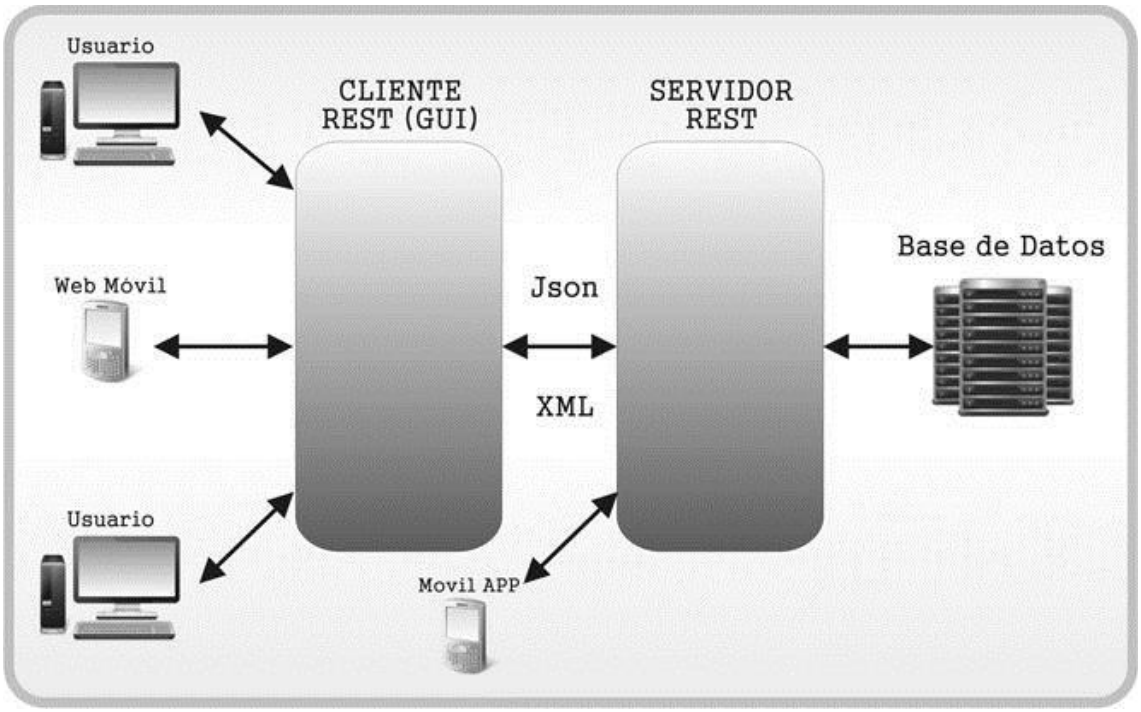

Ilustración 3.2 Funcionamiento de REST.

Un aspecto a subrayar es que la mayoría de los proveedores conocidos mundialmente tales como Yahoo, Google o Facebook, han migrado a esta tecnología. Todos ellos marcaron como obsoletos sus anteriores tecnologías empleadas y pasaron a usar el servicio REST, el cual es mucho más fácil de utilizar y está orientado a los recursos.

Es importante destacar que REST es un estilo de arquitectura basado en los siguientes estándares:

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

3.2.1.1. Principios elementales de REST.

REST define una serie de principios fundamentales que deben cumplir cualquier aplicación que pretenda llamarse REST, todas ellas son proporcionadas al utilizar el protocolo HTTP. Seguidamente, pasamos a detallar pormenorizadamente dichos principios [12]:

- *Protocolo cliente/servidor:* El cliente y el servidor deben estar separados mediante interfaces uniformes, de tal manera que el cliente no conoce cómo se almacena la información y el servidor desconoce el modo en que se presenta la información.
  Es decir, si se demanda un recurso a un servidor en el que se le pasa una serie de datos, el servidor no será capaz de recordarlos en futuras peticiones de un cliente, por lo tanto el estado lo tiene que mantener el cliente y esto lo realiza pasando el estado en cada una de las llamadas a un recurso del servidor.

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

- *Capaz de almacenarse en caché:* Los clientes no disponen de un mecanismo de almacenar las respuestas en caché. Las respuestas deben definirse como almacenables en caché de este modo se evita que los clientes hagan un uso inapropiado de información devuelta por una solicitud.

- *Sistema por capas:* El cliente no debe de saber si está conectado directamente a un servidor final o a un intermediario. De este modo el cliente conserva su independencia con respecto a las distintas capas que forman la arquitectura, por lo tanto, nuestro sistema no debe forzar

al cliente a saber por qué capas se tramita la información. 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 la lógica necesaria para que se puedan ejecutar, por ejemplo Java Applets o JavaScript.

- *Interface uniforme:* Son recursos individuales que deben de estar incluidos dentro de la solicitud, de tal forma que se aumenta la escalabilidad y rendimiento del sistema.

- *Sintaxis universal:* Se utiliza una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI, la cual es simplemente una cadena de caracteres que identifica los recursos de una red de forma unívoca.

- *Uso de hipermedios:* El empleo de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación, son básicamente HTML o XML, aunque también puede utilizarse JSON para obtener un recurso como respuesta de un servidor. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

3.2.1.2. Ventajas y desventajas de REST.

El servicio REST cuenta con una serie de ventajas y desventajas, las cuales pasamos a exponer:

Ventajas:

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

Desventajas:

- Escasas herramientas de desarrollo.
- Gran número de objetos.
- Resulta algo engorroso el manejo del espacio de nombres (URLs).
- Descripción sintáctica/semántica es muy informal (orientada al usuario).

3.2.1.3. Métodos y códigos de estado HTTP.

Un servicio REST debería soportar los métodos más comunes, los cuales están recogidos en la tabla 3.1, cada uno de ellos debería utilizarse en función del tipo de operación que se desee realizar.

En la arquitectura REST, por lo tanto, se establecen una serie de operaciones bien definidas que están orientadas a la consecución de una serie de recursos por parte del servidor y demandados por un cliente. Estas operaciones son las siguientes:

MÉTODO	ACCIÓN
GET	Obtener un recurso.
POST	Crear un nuevo recurso.
PUT	Actualizar un recurso existente.
PATCH	Actualizar un recurso existente de manera parcial.
DELETE	Eliminar un recurso.

Tabla 3.1 Códigos de estado HTTP.

Los códigos de estado HTTP en el cuerpo de la respuesta muestran a la aplicación cliente la acción que debería realizar con la respuesta. Como muestra de ello, podemos concretar por ejemplo que si el código de la respuesta es 201 significa que el objeto que contenía la petición se ha creado correctamente. De la misma manera si el código de respuesta es el 404, la petición a realizar no se encuentra.

En la tabla 3.2 podemos ver una breve ejemplificación de la lista de códigos de respuesta del protocolo HTTP, los cuales se agrupan en torno a cinco clases de respuesta:

CÓDIGO DE ESTADO	SIGNIFICADO
1xx: Respuestas informativas.	
100	Continuar.
102	Procesando la petición.
2xx: Peticiones correctas	
200	Correcto.
201	Creado.
3xx: Redirecciones	
301	Movido permanentemente.
304	No modificado.
4xx: Errores del cliente	
403	Prohibido.
404	No encontrado.
5xx: Errores de servidor.	
500	Error interno del servidor.
503	Servidor no disponible.

Tabla 3.2 Códigos de estado HTTP.

3.2.1.4. Estructura de las URLs.

A la hora de diseñar una URL es muy importante que éstas estén bien formadas y deberían ser visualmente entendibles. Para que esto sea posible, deben cumplir las siguientes reglas básicas:

- Deben mantener una jerarquía lógica.
- Deben de tener un formato independiente.
- Evitar utilizar verbos en los nombres de URL, ya que estos no deben implicar una acción.

- No debemos tener más de una URL para identificar un mismo recurso, es decir, deben ser únicas.
- En la URL no se hacen los filtrados de información de un recurso.

De acuerdo a los requisitos anteriores, una URL se estructura de la siguiente forma:

```
{protocolo}://{dominio o hostname}[:puerto (opcional)]/{ruta del recurso}?
{consulta de filtrado}
```

Ilustración 3.3 Estructura de una URL.

Como ejemplo de URL podemos tomar el siguiente:

GET localhost:8888:v1/recommendation?context=5

Capítulo 4:

ANDROID.

4. ANDROID.

A finales de la década de los 90, los teléfonos móviles empezaron a adquirir una mayor importancia en la sociedad, ya que permitían la comunicación inalámbrica entre personas. Esto supuso la eliminación de las “limitaciones físicas” y ofrecía una clara libertad para estar en contacto con los demás. La comunicación a través de estos primeros dispositivos móviles personales fue posible a través de las redes GSM.

En los años posteriores estos teléfonos móviles fueron adquiriendo nuevas funciones como la reproducción de música y vídeos o la toma de fotografías digitales.

A continuación, realizaremos una delimitación conceptual e histórica sobre el sistema operativo Android, analizaremos la forma en que ha ido evolucionando, su arquitectura y los principales componentes en los que se estructura. Además estudiaremos el ciclo de vida de la actividad de Android, uno de los elementos principales; así como el nivel de seguridad del que dispone, los tipos de almacenamiento que emplea y, finalmente, trataremos la forma en que se estructura un proyecto en Android.

4.1. Delimitación conceptual.

Android es un sistema operativo diseñado fundamentalmente para dispositivos móviles tales como teléfonos, tabletas, relojes, coches… y que se basa en el kernel de Linux.

Los inicios de este sistema operativo se remontan a 2003 con la creación de la empresa Android Inc. entre cuyos creadores destaca Andy Rubin. Posteriormente, en 2005, Google compró esta empresa que por aquella época estaba centrada en el software en general. Dos años después, el sistema operativo Android fue presentado oficialmente y se fundó el consorcio de compañías Open Handset Alliance, una alianza comercial formada por 84 compañías con la finalidad de desarrollar estándares abiertos para dispositivos móviles.

En octubre de 2008 se comercializó el primer teléfono con Android, el T-Mobile G1 (conocido como HTC Dream), lo cual supuso una revolución de la telefonía móvil inteligente. Aunque el comienzo de Android fue un poco lento, rápidamente se posicionó como uno de los sistemas operativos de referencia a nivel mundial.

Android cuenta con una gran comunidad de programadores que desarrollan aplicaciones para la tienda oficial de Android 'Google Play', la cual cuenta ya con más de un millón de aplicaciones a disposición de los usuarios.

En la actualidad existen numerosos sistemas operativos y la oferta de dispositivos móviles es muy amplia. En la ilustración 4.1 se muestra la cuota de mercado de los distintos sistemas operativos realizada por el servicio de estadística NetMarketShare a mediados del año 2017:

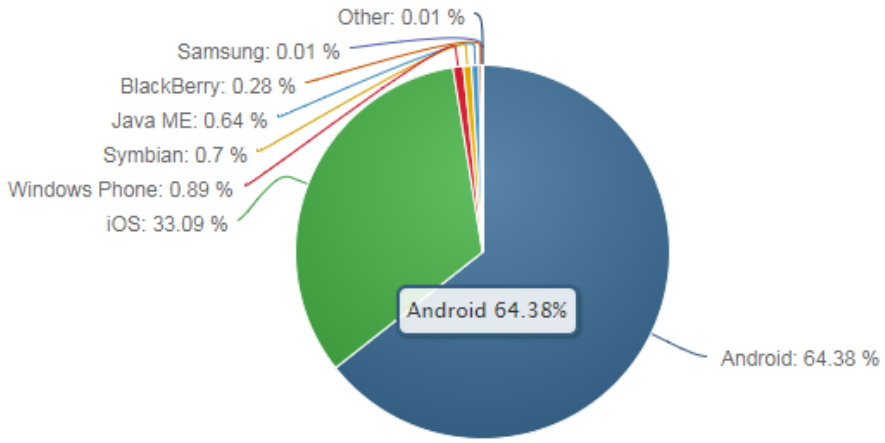

Ilustración 4.1 Cuota de mercado de SO.

En la ilustración 4.1 podemos apreciar cómo la cuota de mercado del sistema operativo Android es claramente superior frente al resto. Como dato relevante, en España las ventas de dispositivos móviles con el sistema operativo Android superan más del 90%.

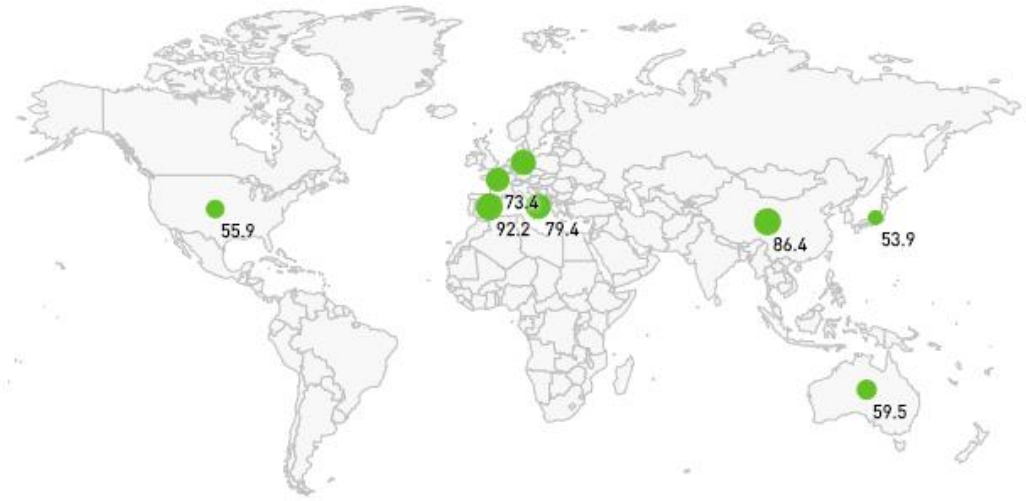

Ilustración 4.2 Cuota de mercado Android por países [Fuente: Kantar Worldpanel].

Android se ha ido consolidando de manera progresiva como una de las plataformas líderes para dispositivos móviles, quedando por delante de iOS y Windows Phone. Debido a la información disponible desde webs oficiales para desarrolladores y a la relativa facilidad para programar aplicaciones, este sistema operativo ha resultado ser la plataforma más idónea a la hora de realizar este proyecto.

4.2. Características.

Existe gran variedad de plataformas para dispositivos móviles: iOS, Windows Phone, Symbian, BlackBerry…, sin embargo, Android tiene una serie de características que lo diferencian de los demás. Este sistema operativo presenta las siguientes cualidades y especificaciones:

- *Adaptabilidad a cualquier tipo de hardware:* Android no ha sido diseñado para ser utilizado en teléfonos y tabletas exclusivamente, sino que además, podemos encontrar una gran variedad de dispositivos que emplean este sistema operativo tales como relojes, cámaras, electrodomésticos…

- *Plataforma realmente abierta:* Se trata de una plataforma de desarrollo libre que se basa en el kernel de Linux y utiliza código abierto.

- *Portabilidad asegurada:* El desarrollo de las aplicaciones en Java nos da la seguridad de que estas puedan ser ejecutadas en cualquier tipo de CPU, lo cual se consigue gracias al concepto de máquina virtual.

- *Arquitectura basada en componentes inspirados en Internet:* La interfaz de usuario se diseña en xml, lo que permite que la misma aplicación pueda ser ejecutada en los distintos tamaños de pantalla que nos podemos encontrar en los dispositivos.

- *Conectividad permanente del dispositivo a Internet.*

- *Gran cantidad de servicios incorporados*, como por ejemplo, bases de datos con SQL, navegador, reconocimiento y síntesis de voz, localización a través de GPS o redes…

- *Nivel de seguridad aceptable:* Cada aplicación cuenta con una serie de permisos que limitan su rango de actuación. Los permisos se integran dentro del mismo código de la aplicación y permiten el desempeño de las distintas funcionalidades.

- *Optimizado para baja potencia y poca memoria:* Se trata de una implementación de Google de la máquina virtual Java optimizada para dispositivos móviles.

4.3. Versiones de Android.

El sistema operativo Android ha sufrido una gran evolución desde sus inicios, las diversas actualizaciones por las que ha ido pasando han permitido la agregación de nuevas funcionalidades y el arreglo de errores detectados.

Los nombres que han recibido las distintas versiones de Android destacan por hacer referencia a postres o dulces, los cuales han sido denominados conforme a un orden alfabético [26]:

Nombre	Versión	APIs	Principales características
Apple Pie 	1.0	1	• Menú desplegable de notificaciones. • Widgets de escritorio. • Android Market.
Banana Bread 	1.1	2	• Actualizaciones automáticas. • Posibilidad de guardar archivos adjuntos en mensajes.
Cupcake 	1.5	3	• Grabación de vídeo. • Opción de auto-rotación. • Pantallas de transiciones animadas.
Donut 	1.6	4	• Cuadro de búsqueda rápida. • Diversidad de tamaños de pantalla. • Android Market.
Eclair 	2.0 2.1	5 7	• Google Maps Navigation. • Pantalla de inicio personalizada. • Síntesis de voz.
Froyo 	2.2	8	• Acciones de voz. • Zona Wi-Fi portátil. • Rendimiento superior.
Gingerbread 	2.3	9	• APIs de juegos. • NFC. • Gestión de la batería.

Honeycomb 	3.0 3.1 3.2	11 12 13	• Diseño optimizado para tablets. • Barra del sistema. • Función ‘Ajustes rápidos’.
Ice-Cream Sandwich 	4.0 4.0.3	14 15	• Pantalla de inicio personalizada. • Control del uso de datos. • Android Beam.
Jelly Bean 	4.1 4.2 4.3	16 17 18	• Google Now. • Notificaciones accionables. • Multiusuario.
KitKat 	4.4	19	• Voz: Ok Google. • Diseño envolvente. • Teléfono inteligente.
Lollipop 	5.0 5.1	21 22	• Material design. • Multipantalla. • Notificaciones en la pantalla de inicio.
Marshmallow 	6.0	23	• Google Now pulsando el botón de inicio. • Permisos de aplicaciones. • Batería más eficiente.
Nougat 	7.0 7.1	24 25	• Vista multiventana. • Modo de realidad virtual. • Nuevas opciones de personalización.
Oreo 	8.0	26	• Nuevo sistema de notifica-ciones. • Iconos adaptativos. • Android Go.

Tabla 4.1 Versiones de Android.

Las cifras en cuanto al uso de las distintas versiones de Android es bastante irregular, siendo Android Lollipop la versión más utilizada con un 32,5% de usuarios, seguida por Android Marshmallow (31,3%) y Android KitKat (20,8%). En la ilustración 4.3 se registran los datos comentados y en ella podemos ver que la mayoría de consumidores aún tienen un sistema operativo de hace más de 3 años:

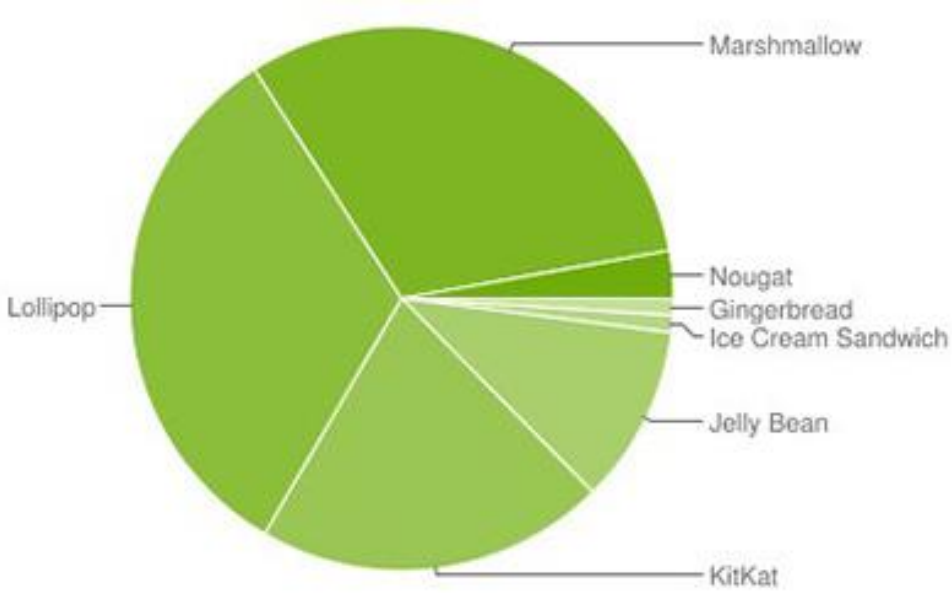

Ilustración 4.3 Tasa de distribución para las versiones de Android.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x	Jelly Bean	16	2.7%
4.2.x		17	3.8%
4.3		18	1.1%
4.4	KitKat	19	16.0%
5.0	Lollipop	21	7.4%
5.1		22	21.8%
6.0	Marshmallow	23	32.3%
7.0	Nougat	24	12.3%
7.1		25	1.2%

Ilustración 4.4 Tabla de distribución para las versiones de Android.

Los dispositivos que cuentan con la versión de Android Nougat son escasos debido a que están disponibles fundamentalmente para terminales de gama alta. En cuanto a la última versión, Android Oreo, no se cuenta con datos registrados ya que se acaba de realizar su lanzamiento oficial en agosto de 2017.

Toda esta información es fundamental a la hora de desarrollar la aplicación que ocupa este proyecto, debido a que es necesario indicar la versión mínima que nuestra aplicación puede soportar, es decir, la aplicación no podrá ser instalada en dispositivos con versiones inferiores a la versión mínima escogida.

Para la realización de la aplicación que ocupa este TFG, se ha utilizado como versión mínima compatible la API 15, debido a que ciertas características y funcionalidades fueron implementadas a partir de esta API, lo cual nos facilita la tarea. Aunque tengamos la restricción de que la aplicación es compatible con

dispositivos desde la API 15 en adelante, ésta es soportada por más del 90% de dispositivos Android en la actualidad, tal y como podemos ver en la ilustración 4.4 de fragmentación de Android.

4.4. Arquitectura.

Como ya se ha mencionado, Android es una plataforma orientada a dispositivos móviles que incluye un sistema operativo y gran cantidad de aplicaciones para el usuario.

A continuación, vamos a realizar una visión global de cada una de las capas que conforman su arquitectura. Cada una de las capas utiliza servicios ofrecidos por las anteriores y, a su vez, ofrece sus propios servicios a las capas de niveles superiores, tal y como se puede ver en la ilustración 4.5 [21]:

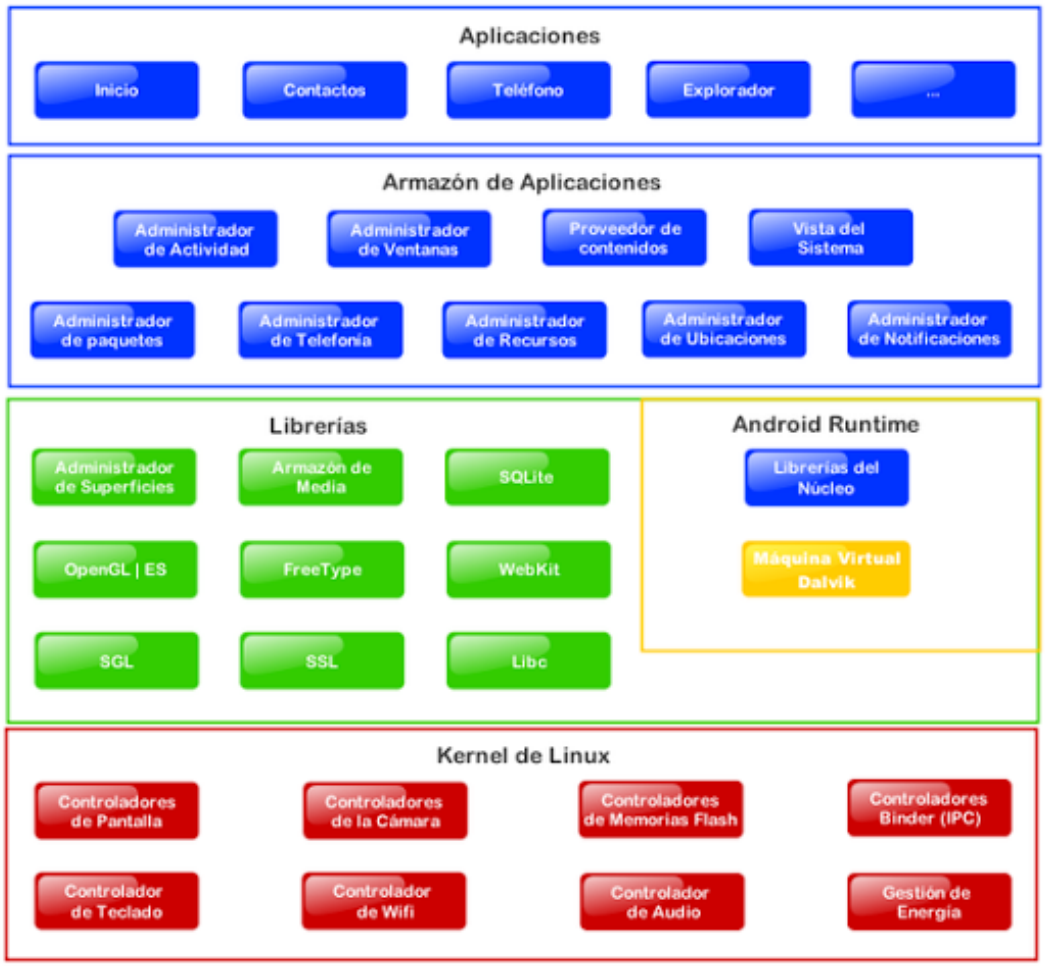

Ilustración 4.5 Arquitectura de Android.

- *Aplicaciones:* Contiene el conjunto de aplicaciones incluidas por defecto de Android (correo electrónico, calendario, agenda de contactos, mapas, navegador…), así como todas aquellas aplicaciones instaladas en el dispositivo que el usuario ha ido añadiendo posteriormente. Todas ellas están disponibles para su descarga a través de la plataforma Google Play. Las aplicaciones utilizan los servicios, las API y librerías de los niveles anteriores.

- *Framework de Aplicaciones:* Constituye el conjunto de herramientas de desarrollo de cualquier aplicación, por lo tanto, es una capa de acceso a las distintas APIs. Este nivel permite reutilizar los componentes por los que está formada fácilmente, lo cual simplifica el desarrollo de aplicaciones. Entre las APIs más importantes ubicadas aquí, podemos destacar: Activity Manager, Windows Manager, Telephone Manager, Content Provider, View System, Location Manager, entre otras.

- *Librerías:* Esta capa se corresponde con las librerías utilizadas por Android, las cuales han sido escritas utilizando C/C++ y están compiladas en el código nativo del procesador. Son una de las partes fundamentales de Android junto con el núcleo basado en Linux y proporcionan acceso a las funcionalidades del sistema. Entre las librerías más importantes ubicadas aquí, podemos encontrar: Librería libc, Librería Surface Manager, OpenGL/SL y SGL, FeeType, etc.

- *Runtime:* Se encuentra en el mismo nivel o capa que las librerías de Android y se basa en el concepto de máquina virtual de Java (lenguaje utilizado en el desarrollo de aplicaciones para Android). Está formado por las Core Libraries y la máquina virtual Dalvik, aunque desde la versión 5.0 utiliza el ART (Android Runtime).

- *Núcleo:* El sistema operativo Android utiliza el núcleo de Linux 2.6 y constituye una capa de abstracción para el hardware de los dispositivos móviles. El núcleo actúa como interfaz entre la aplicación y el

dispositivo, además de proporcionar servicios como el soporte de los drivers necesarios para que cualquier componente hardware pueda ser utilizado, el multiproceso, la seguridad, el manejo de la memoria y la pila de protocolos. Es de código abierto bajo licencia GPL v2.

4.5. Componentes.

A la hora de desarrollar una aplicación para Android, existen una serie de elementos clave que resultan imprescindibles para su buen desarrollo y funcionamiento. Cada uno de los elementos que componen una aplicación representa un punto a través del cual el sistema puede acceder a la aplicación, además algunos de ellos también implementan la lógica de la aplicación tras la interacción del usuario con la interfaz.

Entre los componentes de una aplicación en Android destacan los siguientes:

- *Vistas (View):* Son los elementos que componen la interfaz de usuario de una aplicación como un botón o una entrada de texto. Se pueden definir utilizando código Java, sin embargo, lo habitual es definirlas en un fichero XML.

- *Layout:* Es un conjunto de vistas que se agrupan de una determinada forma. Podemos disponer de distintos tipos de layouts para organizar las vistas de forma lineal, en cuadrícula o indicando la posición de cada vista. Al igual que las vistas, habitualmente se definen utilizando código XML.

- *Actividad (Activity):* Es el principal componente de una aplicación Android, ya que toda aplicación está formada por un conjunto de elementos básicos de visualización (pantallas de la aplicación). Cada una de esas pantallas se denomina actividad y su principal función consiste en crear la interfaz de usuario. Cada una de las actividades de la aplicación son independientes entre sí.

- *Servicio (Service):* Es un proceso que se ejecuta 'detrás', es decir, en segundo plano, sin la necesidad de una interacción con el usuario y permite realizar operaciones internas de la aplicación. Los servicios pueden ser locales (ejecutados en el mismo proceso) o remotos (ejecutados en procesos separados) [23].

- *Intención (Intent):* Hace referencia a la voluntad de realizar una acción, como por ejemplo visitar una página web. La intención se utiliza para:
  - Lanzar una actividad.
  - Lanzar un servicio.
  - Enviar un anuncio de tipo broadcast.
  - Comunicarnos con un servicio.

- *Fragment:* Está formado por varias vistas para crear un bloque funcional de la interfaz de usuario de modo que se pueda ajustar a la gran diversidad de tamaños de pantalla de los distintos dispositivos móviles que existen.

- *Receptor de anuncios (Broadcast receiver):* Permiten recibir y reaccionar frente a anuncios de tipo broadcast. Pueden ser generados por el sistema o por las aplicaciones, se ejecutan en segundo plano y suelen mostrar una notificación al usuario, como por ejemplo: una llamada entrante o batería baja.

- *Proveedores de contenido (Content Provider):* Es un mecanismo estándar que permite almacenar y compartir con otros dispositivos la información que manejan las aplicaciones sin poner en peligro la seguridad del sistema de ficheros.

Todos los componentes empleados para el desarrollo de una aplicación deben estar expresados en el archivo AndroidManifest.xml. Este archivo es único y diferente para cada aplicación por lo que va a permitir que el sistema operativo

conozca toda la información relevante acerca de una aplicación antes de que ésta sea ejecutada. Algunos ejemplos de archivos que se incluyen son los permisos propios de la aplicación, las librerías, las versiones soportadas, etc.

4.6. Ciclo de vida de una aplicación.

En el sistema operativo Android, el ciclo de vida de cualquier aplicación difiere bastante al ciclo de vida en cualquier otro sistema operativo. Esto se debe a que en Android el usuario no controla directamente el ciclo de vida de la aplicación, sino que es controlado principalmente por el propio sistema [22, 35].

Las aplicaciones en Android están formadas por un conjunto de actividades (elementos básicos de interacción con el usuario) y éstas controlan el ciclo de vida de cada aplicación. Las actividades tienen definidos una serie de estados por los que van pasando:

- Activa (Running).
- Visible (Paused).
- Parada (Stopped).
- Destruida (Destroyed).

Es muy importante conocer las distintas fases por las que pasa una aplicación, ya que hay ciertas actividades que solo tienen sentido en ciertos momentos del ciclo. En la ilustración 4.6 podemos ver todos los estados y métodos por los que puede pasar una aplicación en Android, lo cual nos va a ayudar para identificar en qué fase se encuentra la aplicación en cada momento:

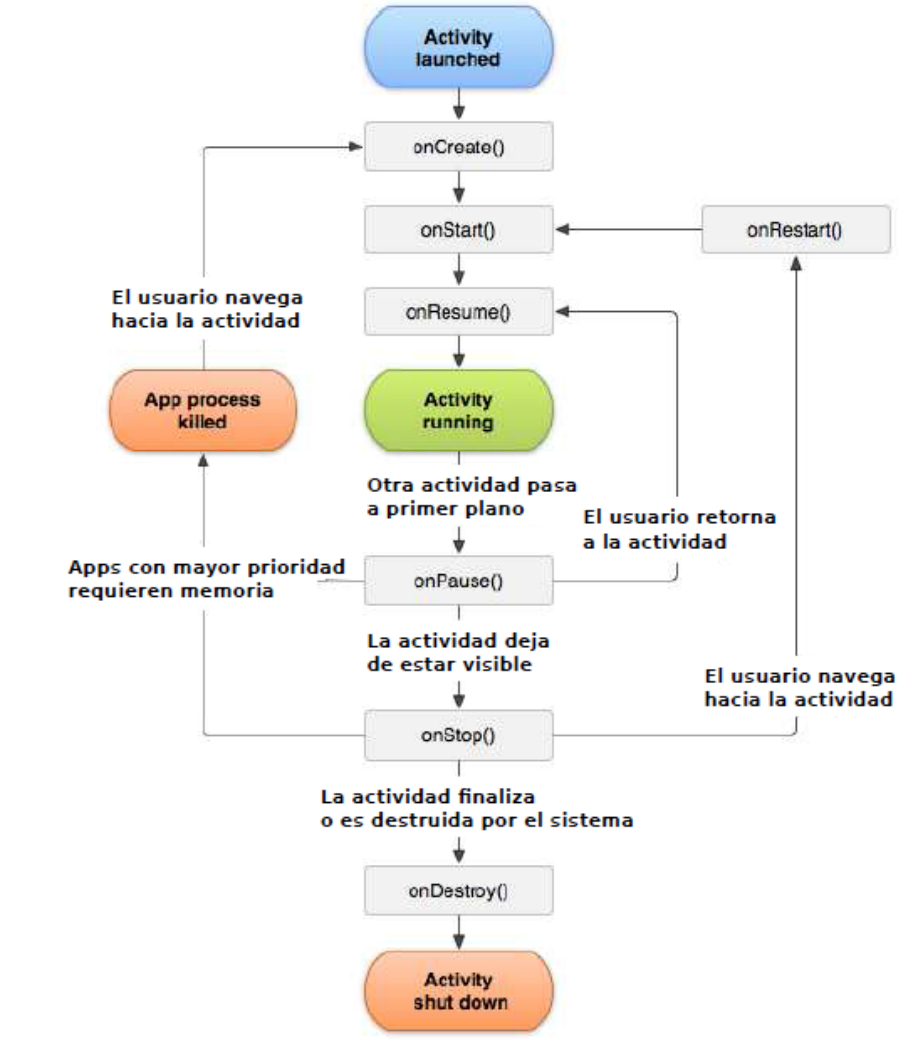

Ilustración 4.6 Ciclo de vida de una aplicación Android.

4.7. Seguridad.

Android cuenta con una serie de privilegios para implementar la seguridad de las aplicaciones. Cada aplicación ejecutada en este sistema operativo requiere de una identificación por parte del usuario y además cuenta con un identificador de grupo. Otro aspecto positivo es el hecho de que todas las aplicaciones han de estar firmadas mediante un certificado que permite identificar a su autor.

Más concretamente, haciendo referencia a las aplicaciones, Android añade un nivel más de seguridad el cual permite el acceso a las distintas funcionalidades del sistema. Por lo tanto, es necesario declarar los permisos pertinentes dentro del archivo AndroidManifest.xml, de este modo la aplicación podrá hacer uso de ciertas características que están protegidas en el dispositivo.

4.8. Almacenamiento de información.

Atendiendo a las necesidades que requiere en cada momento nuestra aplicación, podemos considerar las distintas opciones de almacenamiento de la información que nos proporciona el sistema operativo Android. A continuación, pasamos a describir algunas de ellas:

- *Almacenamiento interno:* Los datos quedan recogidos en la memoria del dispositivo. Toda esta información es de carácter privado y solamente nuestra aplicación puede acceder a ella.
- *Almacenamiento externo:* Los dispositivos compatibles con Android cuentan con un almacenamiento externo. Los archivos guardados aquí son accesibles de forma externa a la aplicación y se pueden modificar.
- *Bases de datos:* El almacenamiento de datos en Android tiene soporte para bases de datos SQLite. Todas las bases de datos creadas desde nuestra aplicación son accesibles desde cualquier clase de la misma, pero no permite que otras aplicaciones accedan.
- *Preferencias compartidas:* Permite el almacenamiento de las preferencias dadas por el usuario en la aplicación de tal forma que se puedan quedar almacenados datos como clave/identificador y el valor asociado a cada clave.

4.9. Estructura de un proyecto Android.

Al comenzar a crear un nuevo proyecto de Android aparecerán en la herramienta ‘Desarrollo’ una serie de archivos y carpetas ya creados de antemano que conforman nuestra aplicación.

Un aspecto fundamental para entender el desarrollo en Android es el conocimiento de las principales carpetas que componen el proyecto. Seguidamente vamos a describir brevemente la utilidad de los archivos y directorios más importantes que componen la estructura de un proyecto Android:

- *Src:* Este directorio incluye el código fuente que describe la lógica de la aplicación. Entre los archivos de esta carpeta destacan los siguientes:

  - *Java:* Se agrupa en distintos paquetes e inicialmente se pueden encontrar en ella el archivo correspondiente al código fuente de la Activity que se ha creado al generar el proyecto.
  - *Res:* Es la carpeta de recursos de la aplicación. En ella se mantendrán una serie de archivos en formato XML con los datos referentes a los recursos usados por la aplicación.
  - *AndroidManifest.xml:* Contiene información esencial necesaria sobre el sistema Android, la cual es imprescindible para poder ejecutar cualquier línea de código.

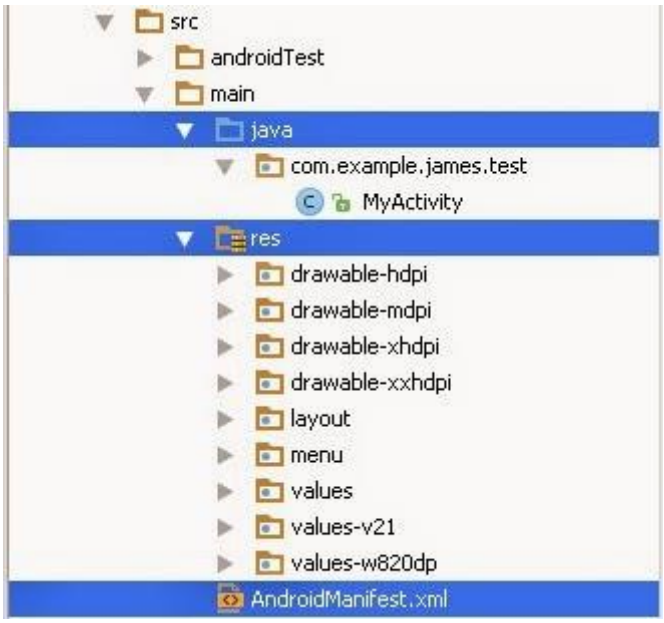

Ilustración 4.7 Estructura de paquetes de un proyecto Android.

- *Gradle Scripts:* En este apartado estará la configuración del Gradle, que es el encargado de empaquetar nuestra aplicación y todas las dependencias del proyecto, ya estén en local o en un repositorio en Internet: Maven, GitHub, etc.

Capítulo 5:

ANÁLISIS DEL PROYECTO.

5. ANÁLISIS DEL PROYECTO.

El proyecto que se va a diseñar es un sistema de recomendación online basado en un algoritmo de filtrado colaborativo, el cual recomendará canciones a los usuarios según el estado de ánimo deseado y, además, atendiendo a sus gustos y preferencias.

El proyecto se puede dividir en cuatro partes totalmente diferenciadas, las cuales son las siguientes:

- Un algoritmo de filtrado colaborativo, el cual calculará las recomendaciones a partir de las puntuaciones de los distintos usuarios. Estas recomendaciones también serán sensibles al contexto de estado de ánimo que se introduzca.
- Un servicio web, que se encargará de la comunicación entre el servidor y el prototipo de la aplicación móvil.
- Un prototipo de aplicación móvil, lo cual servirá al usuario para interactuar con el sistema para obtener las recomendaciones, e incluso podrá valorar las canciones y añadir el contexto del estado de ánimo.
- Por último, una base de datos en la que se almacenará toda la información referente a los usuarios, las canciones y las valoraciones realizadas por los mismos respecto a la música.

Tras exponer a grandes rasgos una breve presentación y las partes en las que se divide este proyecto y al tratarse de un proceso de desarrollo de software, a continuación vamos a aplicar las técnicas de Ingeniería del Software para llevarlo a cabo.

A lo largo del tiempo se han acuñado varias definiciones sobre la Ingeniería del Software, aunque ninguna de ellas se ha convertido en un estándar para la comunidad, la definición que se va a exponer a continuación resultaría perfectamente válida:

Ingeniería del Software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software (Bohem, 1976).

Mediante la aplicación correcta de diferentes normas y métodos podremos conseguir unos mejores resultados en lo que respecta tanto al desarrollo, como al uso del software, lo que nos permitirá satisfacer los objetivos fundamentales de la Ingeniería del Software.

Entre los objetivos de la Ingeniería del Software podemos destacar [30]:

- Mejorar el diseño de aplicaciones o software de tal modo que se adapten mejor.
- Mejorar la calidad al desarrollar aplicaciones con una complejidad elevada.
- Obtener una mayor exactitud tanto en los costes de los proyectos como en el tiempo de desarrollo de los mismos.
- Conseguir una mayor eficiencia de los sistemas con procesos que permitan medir, mediante normas específicas, la calidad del software, con la finalidad de obtener siempre la mejor calidad posible.
- Una organización más eficiente de los equipos de trabajo, tanto en las áreas de desarrollo, como en el área de mantenimiento de software.
- Detectar, a través de pruebas, mejoras en el funcionamiento del software desarrollado.

Para poder alcanzar los objetivos anteriormente descritos y realizar un correcto proceso de Ingeniería del Software debemos de realizar lo siguiente [29]:

- *Especificación de requerimientos:* Se obtiene el propósito del sistema, así como las propiedades y restricciones impuestas sobre el mismo.
- *Análisis del sistema:* Se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.
- *Diseño del sistema:* Se establecen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- *Implementación:* Consiste en la traducción del modelo lógico del sistema a código fuente.
- *Pruebas:* Verificación y validación del sistema.

5.1. Especificación de requerimientos.

Como hemos indicado anteriormente, la especificación de los requerimientos es la etapa inicial en los procesos de Ingeniería del Software. Este punto es muy complejo ya que se debe de conocer el propósito del proyecto software, de tal modo que cuando decidamos afrontar el desarrollo del mismo seamos capaces de cumplir el propósito perseguido. Para ello, es muy importante tener en cuenta que debemos satisfacer las restricciones que nos hayan impuesto, para no establecer diferentes limitaciones.

Al tratarse de un proyecto académico, el propósito se conoce desde el momento de la creación del mismo. En el caso de que el proyecto fuera de un ámbito comercial, la manera de determinar el propósito sería mediante una serie de herramientas como pueden ser: entrevistas con los clientes, encuestas con los posibles usuarios, estudios de la situación de la empresa y observaciones in situ de los procesos de desarrollo diarios.

Como hemos dicho anteriormente nuestro proyecto es de ámbito académico y el propósito es el siguiente:

- *Diseño y desarrollo de un sistema de recomendación musical con contexto de estado de ánimo y comunicación con un prototipo de aplicación móvil a través de un servicio web.*

Ahora es el momento de indicar cuales son los requerimientos del proyecto, pero antes debemos saber que los requerimientos de un proyecto software son el conjunto de propiedades o restricciones, definidas de manera precisa, que debe de satisfacer el proyecto software, para el cual se diferencian dos tipos de requerimientos:

- *Requerimientos funcionales:* Aquellos que se refieren expresamente al funcionamiento del sistema.
- *Requerimientos no funcionales:* Son todos aquellos requerimientos no referidos al estricto funcionamiento del sistema, sino a otros factores externos.

5.1.1. Requerimientos funcionales.

En este apartado vamos a detallar los requerimientos funcionales del proyecto. Básicamente va a consistir en indicar las funcionalidades que nuestro sistema debe de satisfacer para un correcto cumplimiento del propósito [30].

Al no tener unos clientes que nos indiquen qué funcionalidades debe de cumplir, ya que es un proyecto académico, vamos a basarnos en los distintos sistemas de recomendación basados en el contexto de ánimo que existen, y vamos a tratar de establecer unos requerimientos que cumplan con nuestros objetivos.

Los requerimientos funcionales serán:

RF1. Entrar o darse de alta en el sistema: Al tratarse de un servicio personalizado debemos de dotar al sistema de un mecanismo de login para identificar a cada usuario.

RF2. Mostrar recomendaciones personalizadas: El sistema tiene que ser capaz de ofrecer recomendaciones a los usuarios basadas en sus gustos y preferencias.

RF3. Obtener el estado de ánimo: El sistema deberá tener mecanismos para obtener el estado de ánimo del usuario.

RF4. Mostrar recomendaciones personalizadas en contexto de ánimo: El sistema debe ofrecer recomendaciones musicales basadas en el estado de ánimo que el usuario haya elegido.

RF5. Mostrar canciones aleatorias: El sistema debe de ofrecer un listado de canciones sin ningún criterio de recomendación.

RF6. Mostrar canciones favoritas: El sistema debe de ofrecer un listado de canciones que previamente fueron seleccionadas por el usuario como favoritas.

RF7. Añadir y eliminar favoritos: El sistema tiene que ser capaz de añadir o eliminar canciones de la lista de favoritos.

RF8. Valorar canciones: El sistema tiene que ser capaz de evaluar una canción según el criterio del usuario de cara a mejorar la recomendación personalizada.

RF9. Actualizar el modelo de recomendación: El sistema tiene que proporcionar mecanismos que permitan actualizar dicho modelo de recomendación.

RF10. Cerrar sesión: El sistema tiene que facilitar la desvinculación de la cuenta asociada en el dispositivo.

5.1.2. Requerimientos no funcionales.

Estos requerimientos no hacen referencia directa a las funciones específicas que tiene que entregar el sistema, sino que más bien son características y restricciones del mismo. Estos requerimientos no pueden modelarse y tampoco aparecen en los casos de usos. Algunos de estos pueden ser la seguridad, fiabilidad, escalabilidad, la respuesta en tiempo aceptable, la usabilidad, las restricciones de hardware y software, las restricciones legales, como puede ser en nuestro caso, las licencias para la reproducción de las canciones.

Como podemos ver, existen una gran variedad de requerimientos no funcionales, pero al tratarse de un proyecto de ámbito académico, no tenemos a ningún usuario que nos imponga unas restricciones muy fuertes, por lo cual seremos flexibles.

Para detallar mejor estos requerimientos vamos a especificarlos de manera dividida por un lado, para el caso del servidor web y, por otro lado, para la aplicación móvil, ya que son requerimientos distintos.

En el servidor web nos encontraremos los siguientes requerimientos:

- Requerimientos de hardware:
  - Es necesario tener un equipo informático. No vamos a entrar a concretar las características de distintos componentes, ya que todo dependerá de la carga que tenga que soportar, y actualmente no tenemos ninguna restricción pero es muy importante que tenga una interfaz de red para su conectividad.
- Requerimientos de software:
  - Sistema Operativo Ubuntu 14.04 64-bit
  - Apache 2.4
  - PHP 5.5
  - MySQL 5.5
  - PHPUnit 4.8
  - Java 8
- Requerimientos de escalabilidad:
  - Conexiones múltiples: El servidor web tiene que ser capaz de atender a distintas peticiones simultáneamente. El número de estas conexiones también dependerá de los requerimientos de hardware que tengamos.
- Requerimientos de seguridad:
  - Autenticación: Es la capacidad que tiene el servidor web de identificar a los usuarios que soliciten sus recursos.
  - Autorización: Son los derechos de los usuarios, los cuales podrán tener acceso a algunos recursos según el tipo de autorización.
- Requerimientos de modificabilidad:
  - El sistema se tiene que adaptar a nuevos cambios en la medida de lo posible, ya que en cualquier momento pueden cambiar los requerimientos tanto funcionales como no funcionales, y debería poder ser predecible estimar el esfuerzo necesario y costo de hacer el cambio.

En la aplicación móvil los requerimientos no funcionales serán los siguientes:

- Requerimientos de hardware:
  - En este caso sólo necesitamos un dispositivo móvil, con posibilidad de conexión a Internet mediante Wi-Fi o conexión de datos.
- Requerimientos de software:
  - Sistema operativo Android 4.0.3 o superior.
  - Google Play Services.
- Requerimientos de seguridad:
  - Autenticación: La aplicación tiene que ser capaz de identificar a los usuarios.
- Requerimientos de usabilidad:
  - Predecible: El simple vistazo a la interfaz tiene que ser suficiente para imaginar el resultado de la interacción con los distintos componentes.
  - Familiaridad: La interfaz debe de ser rápidamente reconocible para el usuario.
  - Consistencia: Los elementos se tienen que utilizar de la misma manera a lo largo de la aplicación con resultados semejantes, siguiendo una misma guía de estilos.
- Requerimientos de rendimiento:
  - Tiempo de respuesta: Es el tiempo de latencia que necesitamos para poder reflejar los cambios. Éste tiene que ser aceptable, tanto para la comunicación con el servidor web como por las interacciones internas de la aplicación.

5.2. Modelos de casos de uso.

En este punto del análisis, con los requisitos funcionales ya definidos, pasaremos a representar los distintos casos de usos que encontraremos en el sistema.

Los casos de usos tienen que explicar las distintas tareas que cualquier usuario debe realizar de manera cotidiana cuando haga uso de nuestra aplicación, ya que se crean con el objetivo de refinar el conjunto de requisitos de dicha función [31].

Primero vamos a pasar a definir los distintos diagramas necesarios para crear los casos de uso. Empezaremos definiendo el diagrama frontera en el que se muestra toda la funcionalidad:

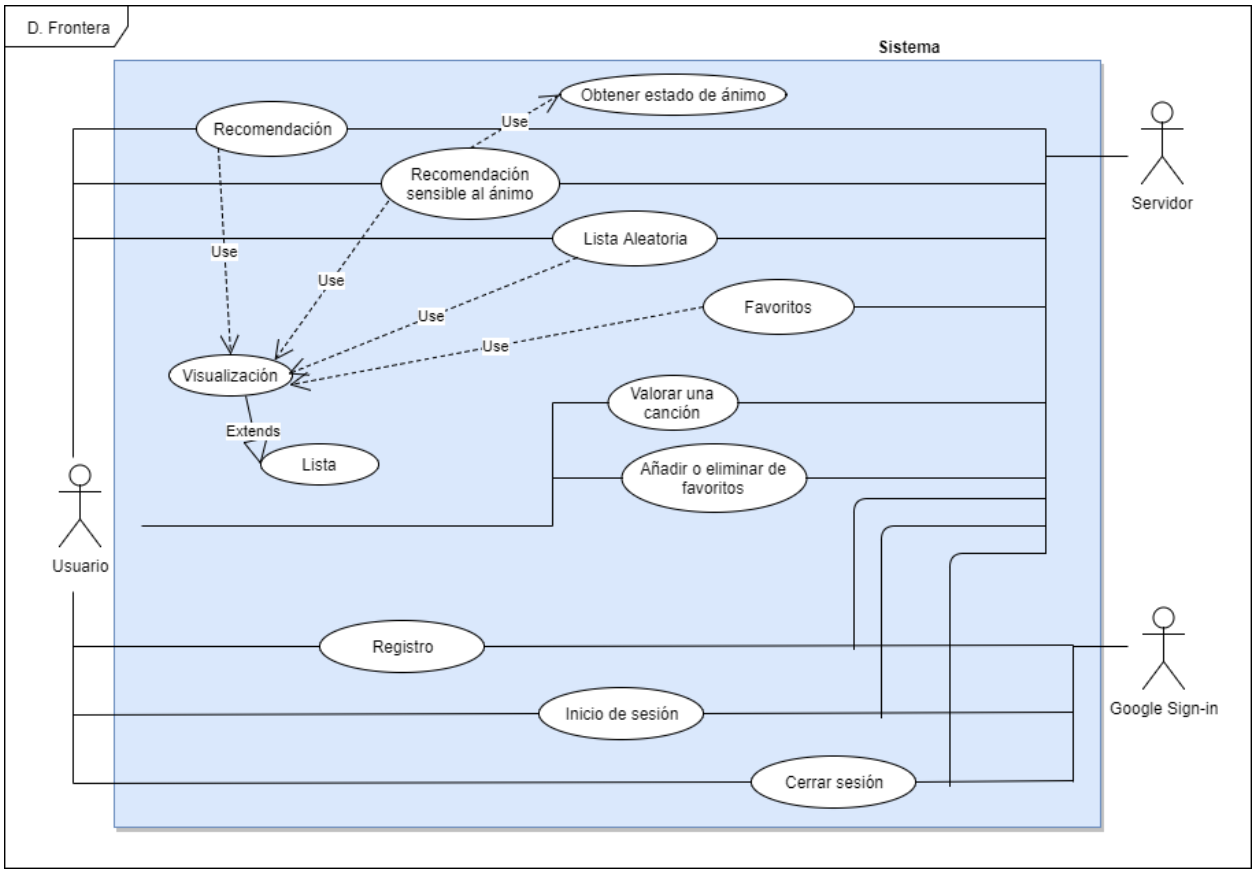

Ilustración 5.1 Diagrama frontera.

Caso de uso - Registro

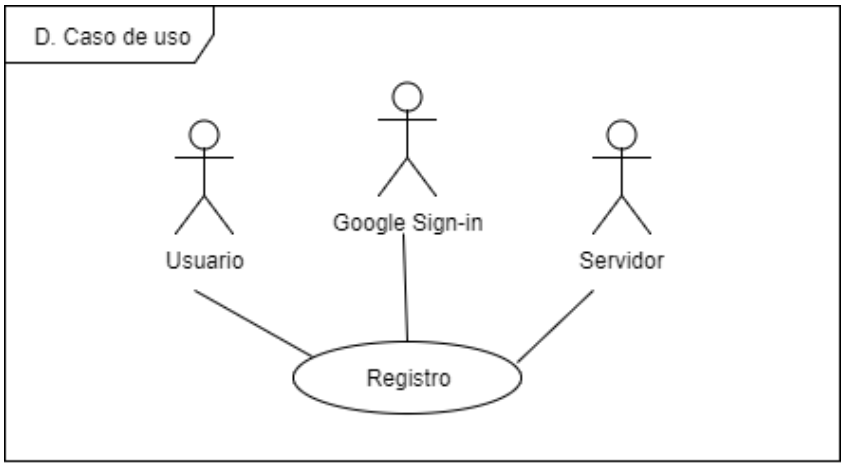

Ilustración 5.2 Diagrama de caso de uso - registro.

Caso de uso - Registro	
Actor primario	Usuario.
Actores participantes	Usuario, Google Sign-in, Servidor.
Condiciones previas	1. El usuario no está registrado en el servidor. 2. El usuario posee una o varias cuentas Google. 3. Tiene vinculada una o varias cuentas de Google en el móvil.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación mostrará un botón de inicio de sesión. 3. El actor Google Sign-in muestra la cuenta Google vinculada al dispositivo. 4. El usuario selecciona la cuenta que desea vincular. 5. El actor Google Sign-in autentifica al usuario cediendo sus datos al sistema. 6. El sistema envía el email al actor servidor. 7. El actor servidor comprueba si existe el usuario, al no existir lo da de alta en su sistema y le asigna un token de acceso. 8. El actor servidor cede a la aplicación el token de acceso personal del usuario.
Postcondiciones	El actor usuario ha sido registrado en el servidor y la aplicación tiene la sesión iniciada.

Tabla 5.1 - Caso de uso del registro.

Caso de uso – Inicio de sesión

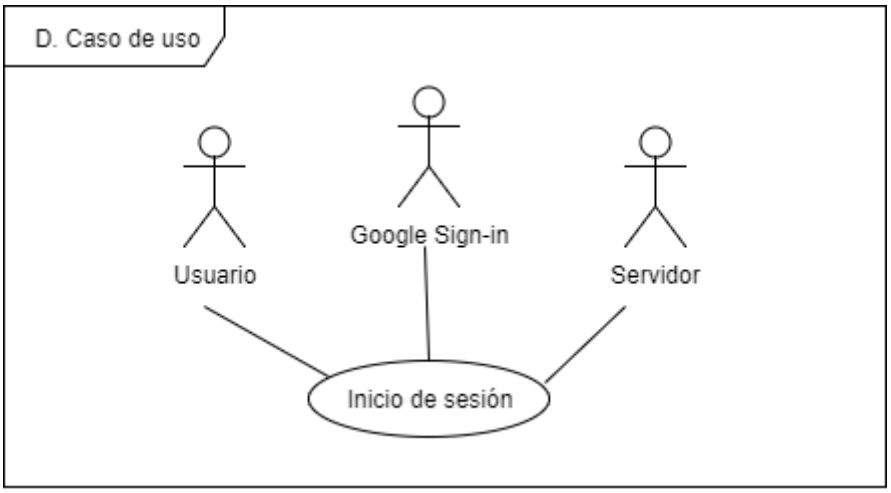

Ilustración 5.3 Diagrama de caso de uso – inicio de sesión.

Caso de uso – Inicio de sesión	
Actor primario	Usuario.
Actores participantes	Usuario, Google Sign-in, Servidor.
Condiciones previas	1. El usuario está registrado en el servidor. 2. El usuario no ha iniciado sesión.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación mostrará un botón de inicio de sesión. 3. El actor Google Sign-in muestra la cuenta Google vinculada al dispositivo. 4. El usuario selecciona la cuenta que desea vincular. 5. El actor Google Sign-in autentifica al usuario cediendo sus datos al sistema. 6. El sistema envía el email al actor servidor. 7. El actor servidor comprueba si existe el usuario, al existir obtiene su token de acceso. 8. El actor servidor cede a la aplicación el token de acceso personal del usuario.
Postcondiciones	El actor usuario ha sido autenticado y la aplicación tiene la sesión iniciada.

Tabla 5.2 - Caso de uso de inicio de sesión.

Caso de uso - Recomendación

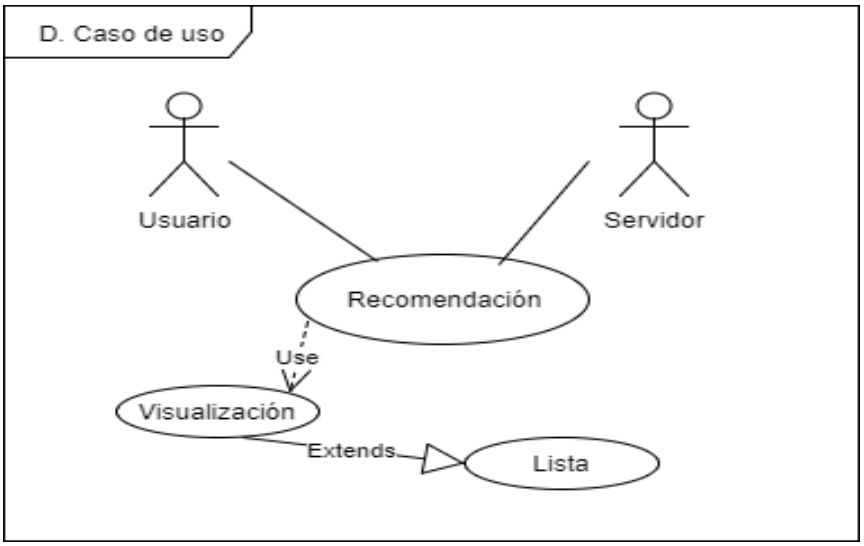

Ilustración 5.4 Diagrama de caso de uso – recomendación.

Caso de uso – Recomendación	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación. 2. El usuario tiene canciones valoradas.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación solicita de manera asíncrona al actor servidor que calcule el modelo de recomendación para el usuario. 3. La aplicación muestra el menú. 4. El usuario pulsa la opción de recomendación normal. 5. La aplicación solicita al servidor la lista de canciones recomendadas. 6. El servidor devuelve una lista de canciones previamente calculadas.
Postcondiciones	Se muestra la lista de canciones recomendadas con toda su información.

Tabla 5.3 - Caso de uso de ver lista de canciones recomendadas.

Caso de uso - Aleatorias

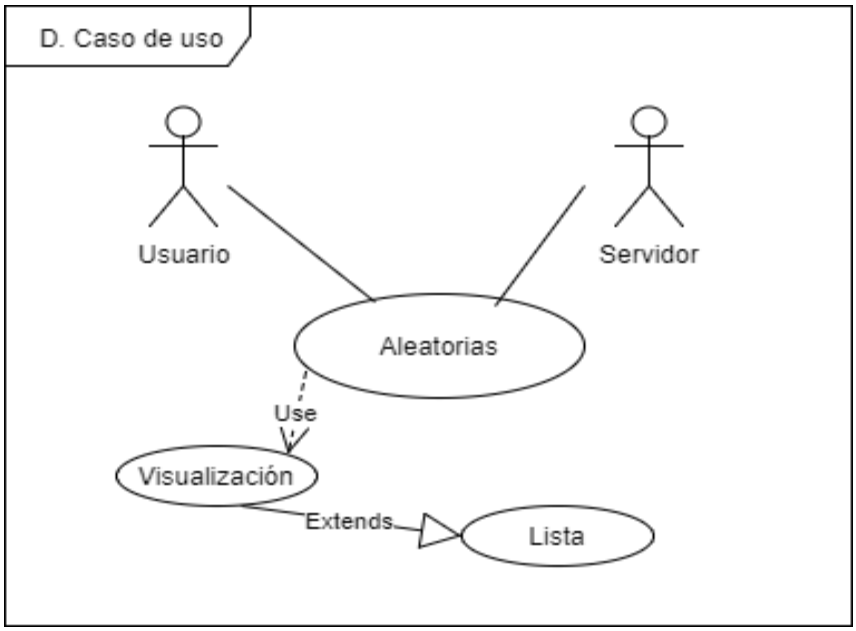

Ilustración 5.5 Diagrama de caso de uso – aleatorias.

Caso de uso – Aleatorias	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación muestra el menú. 3. El usuario pulsa la opción de lista aleatoria. 4. La aplicación solicita al servidor la lista de canciones aleatorias. 5. El servidor devuelve una lista de canciones.
Postcondiciones	Se muestra la lista de canciones aleatorias con toda su información.

Tabla 5.4 - Caso de uso de ver lista de canciones aleatorias.

Caso de uso – Recomendación sensible al ánimo

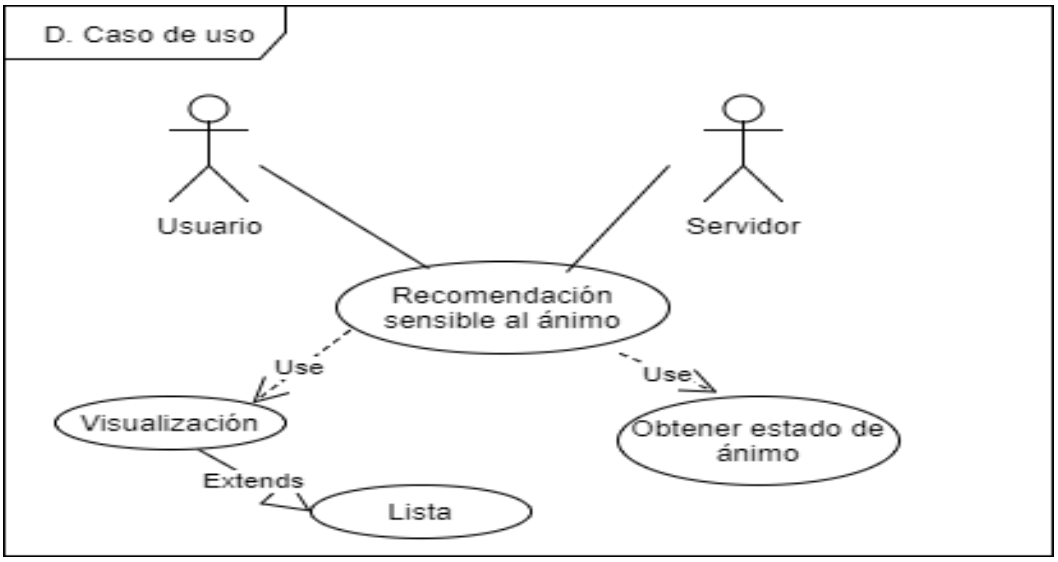

Ilustración 5.6 Diagrama de caso de uso - recomendación sensible al ánimo.

Caso de uso – Recomendaciones sensible al ánimo	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación. 2. El usuario tiene canciones valoradas.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación solicita de manera asíncrona al actor servidor que calcule el modelo de recomendación para el usuario. 3. La aplicación muestra el menú. 4. El usuario pulsa la opción de recomendación con contexto. 5. La aplicación muestra una pantalla con los diferentes estados de ánimo. 6. El usuario pulsa el estado de ánimo deseado. 7. La aplicación solicita al servidor la lista de canciones recomendadas basadas en el estado de ánimo elegido. 8. El servidor devuelve una lista de canciones.
Postcondiciones	Se muestra la lista de canciones recomendadas sensibles al contexto de ánimo elegido con toda su información.

Tabla 5.5 - Caso de uso de ver lista de canciones recomendadas según el contexto de ánimo.

Caso de uso – Valorar una canción

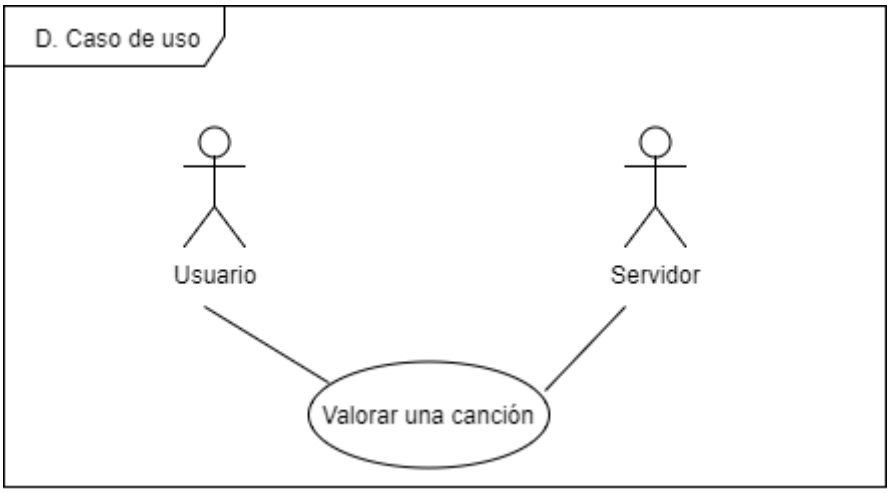

Ilustración 5.7 Diagrama de caso de uso - valorar una canción.

Caso de uso – Valorar una canción	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación. 2. El usuario entra en la pantalla donde se muestra la lista de canciones, ya sean recomendadas o no.
Flujo normal	1. La aplicación muestra la lista de canciones y en cada apartado se muestra un conjunto de estrellas. 2. El usuario pincha en las estrellas cambiando su valor entre 1 y 5 estrellas. 3. La aplicación manda al servidor la nueva valoración para esa canción y usuario. 4. El servidor actualiza en su registro de valoraciones la nueva puntuación obtenida.
Postcondiciones	Se muestra en información de la canción la nueva valoración y el servidor actualiza o añade esa nueva valoración.

Tabla 5.6 - Caso de uso de valorar una canción.

Caso de uso – Añadir o eliminar favorito

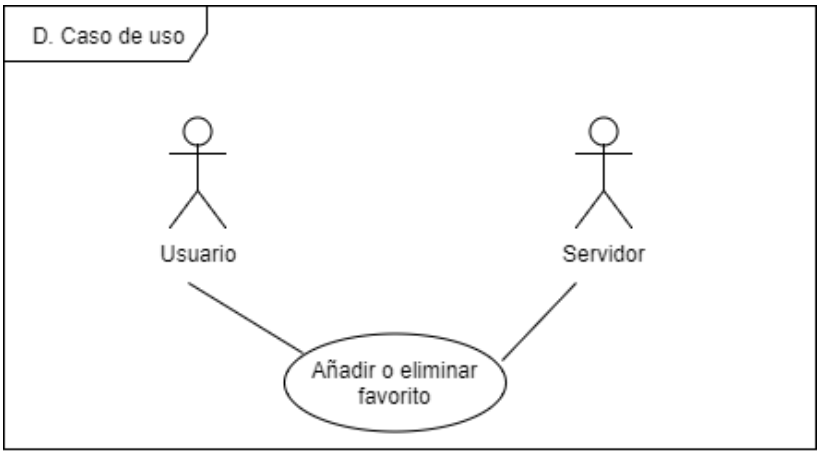

Ilustración 5.8 Diagrama de caso de uso - añadir o eliminar favorito.

Caso de uso – Añadir o eliminar favorito	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación. 2. El usuario entra en la pantalla donde se muestra la lista de canciones, sean recomendadas o no.
Flujo normal	1. La aplicación muestra la lista de canciones y en cada apartado se muestra un corazón. 2. El usuario pincha en un corazón gris. 3. La aplicación manda al actor servidor el estado de favorito para esa canción y usuario. 4. La aplicación actualiza de manera inmediata el cambio del corazón a rosa, para indicar que esa canción ha sido marcada como favorita.
Flujo alternativo	2. El usuario pincha en un corazón rosa. 3. La aplicación manda al actor servidor el estado de no favorito para esa canción y usuario. 4. La aplicación actualiza el corazón a gris, para indicar que esa canción no es considerada favorita.
Postcondiciones	La lista de canciones favoritas es actualizada tanto en el servidor como en la aplicación.

Tabla 5.7 - Caso de uso de añadir o eliminar una canción como favorita.

Caso de uso – Favoritos

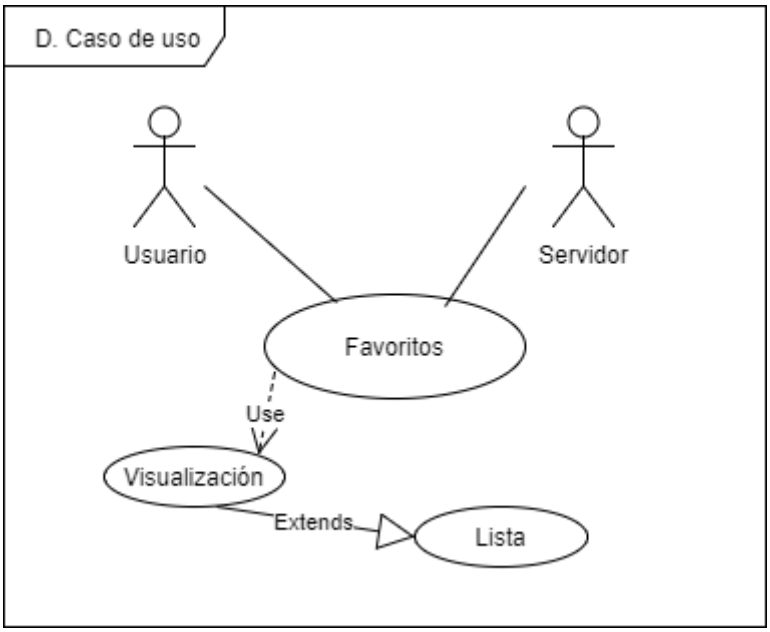

Ilustración 5.9 Diagrama de caso de uso - favoritos.

Caso de uso – Favoritos	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación muestra el menú. 3. El usuario pulsa la opción de lista de favoritos. 4. La aplicación solicita al servidor la lista de canciones marcadas como favoritas. 5. El servidor devuelve una lista de canciones.
Postcondiciones	Se muestra la lista de canciones favoritas con toda su información.

Tabla 5.8 - Caso de uso de ver lista de favoritos.

Caso de uso – Cerrar sesión

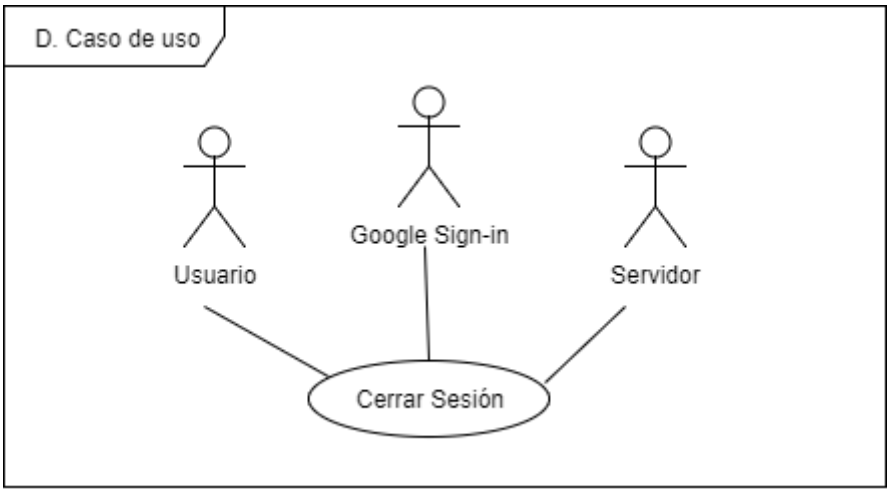

Ilustración 5.10 Diagrama de caso de uso - cerrar sesión.

Caso de uso – Cerrar sesión	
Actor primario	Usuario.
Actores participantes	Usuario, Servidor, Google Sign-in.
Condiciones previas	1. El usuario ha iniciado sesión en la aplicación.
Flujo normal	1. El usuario inicia la aplicación móvil. 2. La aplicación muestra el menú. 3. El usuario pulsa la opción de “mi cuenta”. 4. La aplicación muestra información del usuario y un botón para cerrar sesión. 5. La aplicación informa a Google Sign-in del cierre de sesión. 6. La aplicación muestra la pantalla de inicio de sesión.
Postcondiciones	Toda la información que se mantenía en la sesión proporcionada por Google Sign-in como por el servidor ha sido eliminada.

Tabla 5.9 - Caso de uso de cerrar sesión.

Capítulo 6:

DISEÑO DEL PROYECTO.

6. DISEÑO DEL PROYECTO.

En este punto vamos a tratar una de las partes más duras que nos podemos encontrar en el proceso de Ingeniería del Software, ya que es extremadamente delicada si queremos desarrollar un proyecto software de gran calidad, porque nos encontramos con multitud de técnicas que son muy diversas y complejas.

La elección incorrecta de algunas de estas técnicas pueden hacer que todos los esfuerzos en las tareas previas del proceso de Ingeniería del Software sea en vano, ya que repercutirá en el proceso de implementación de la solución, generándonos distintos problemas y mermando la calidad de dicha solución, por lo que si tomamos buenas decisiones podremos reducir la complejidad y la duración de la codificación.

Así que pasaremos a identificar los objetivos finales del sistema y elegiremos la estrategia que nos puede ofrecer una mejor visión de la aplicación, las cuales serán:

- Diseño de la base de datos:
  - Modelo Entidad/Relación.
  - Entidades finales.
- Diseño de la interfaz:
  - Metáforas:
  - Prototipos:
  - Caminos de navegación:

6.1. Diseño de la base de datos.

6.1.1. Datos.

Según la naturaleza de los datos con los que tenemos que interactuar, tenemos que analizar cuáles son los elementos que nos aportan información para poder definir las estructuras de datos más adecuados, ya que nuestro sistema tendrá que trabajar con ellas.

Podemos distinguir los siguientes elementos:

- *Los usuarios,* los cuales tienen un identificador que el sistema le asignará: datos personales, como son el nombre, email y, por último, una clave que es su token personal, necesario para acceder a algunas funcionalidades del servicio web.
- *Las canciones,* las cuales se componen de un identificador que el sistema le asigna, el nombre de la canción, el nombre del artista o grupo, el género, así como los valores de aurosal, valence y zone que se usan para calcular el estado de ánimo.
- *Las valoraciones:* estos datos representan la distintas puntuaciones que otorgan los usuarios a las canciones, se componen del id del usuario, el id de la canción, la valoración, la fecha de cuando se produjo la valoración y si la canción ha sido marcada como favorita o no.

6.1.2. Modelo entidad – relación.

Una vez analizados los distintos elementos de los que se compone nuestro sistema, debemos poder representarlos en una base de datos, por lo tanto, tenemos que diseñar las tablas que lo van a componer. Para ello, necesitamos realizar el diseño conceptual utilizando la técnica más extendida, que es, con diferencia, el modelo entidad-relación [34].

El modelo entidad-relación es una herramienta que permite representar cualquier abstracción, percepción y conocimiento en un sistema de información formado por objetos denominados entidades y relaciones. Este modelo de datos puede representarse visualmente con un diagrama entidad-relación.

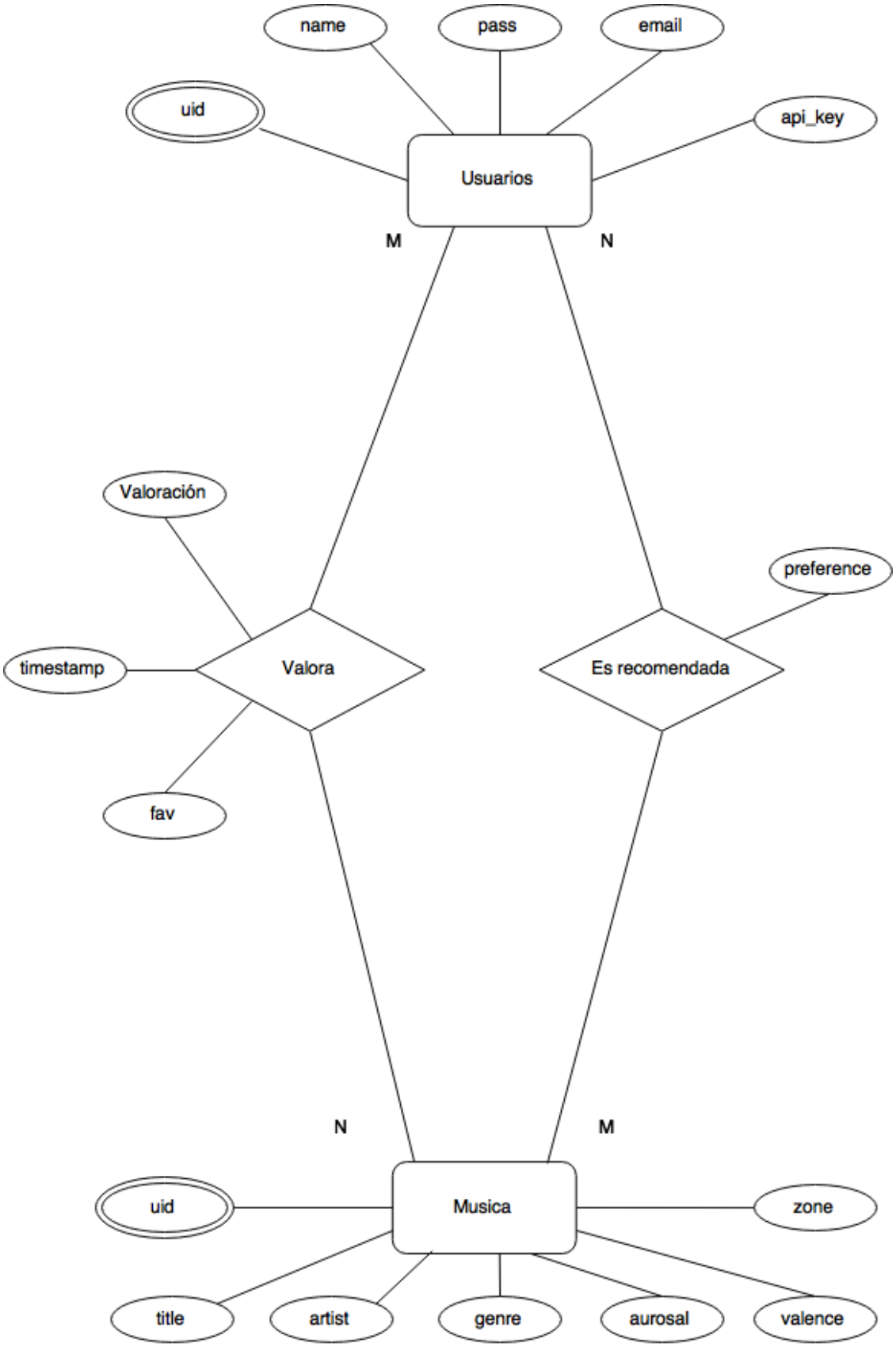

Ilustración 6.1 Esquema conceptual de la aplicación.

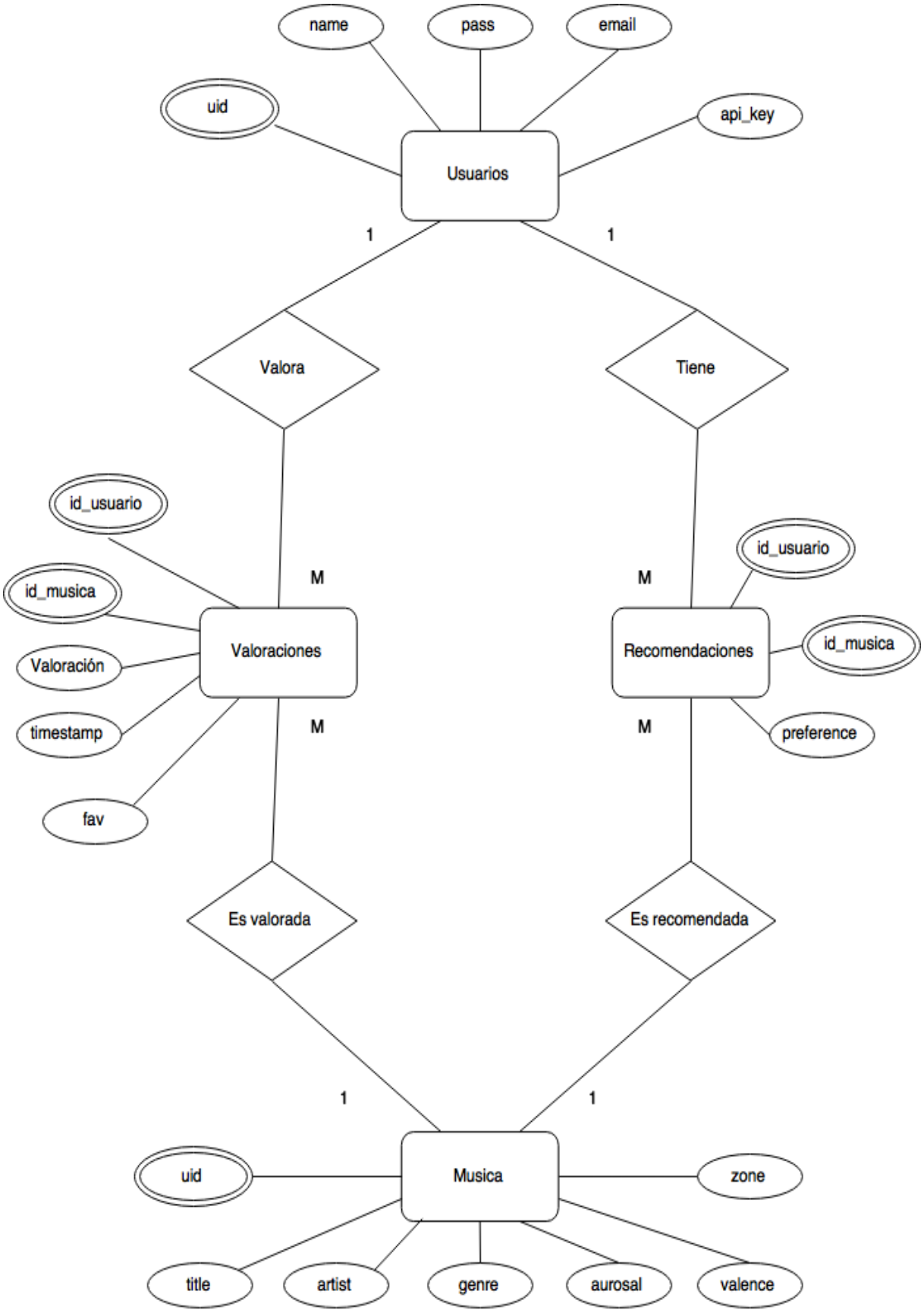

Ilustración 6.2 Esquema conceptual modificado de la aplicación.

6.1.3. Entidades finales

Utilizando como referencia el esquema conceptual modificado, anteriormente expuesto, podemos observar que necesitamos definir cuatro tablas para modelar nuestro sistema; son las siguientes:

- Usuarios:

Atributo	Tipo	Clave
uid	Entero (Único)	Primary key
name	Texto	
pass	Texto	
email	Texto	
api_key	Texto	

Tabla 6.1 Estructura de tabla de usuarios.

- Música:

Atributo	Tipo	Clave
uid	Entero (Único)	Primary key
title	Texto	
artist	Texto	
genre	Texto	
aurosal	Entero	
valence	Entero	
zone	Entero	

Tabla 6.2 Estructura de tabla de música.

- Valoraciones:

Atributo	Tipo	Clave
id_usuario	Entero	Primary key (Foreign key)
id_musica	Entero	Primary key (Foreign key)
valoracion	Entero	
timestamp	Fecha	
fav	Entero	

Tabla 6.3 Estructura de tabla de valoraciones.

- Recomendaciones:

Atributo	Tipo	Clave
id_usuario	Entero	Primary key (Foreign key)
id_musica	Entero	Primary key (Foreign key)
preference	Entero	

Tabla 6.4 Estructura de tabla de recomendaciones.

6.2. Diseño de la interfaz.

Esta parte del diseño es la más visible de la aplicación, ya que se trata de la parte con la que el usuario interacciona a fin de usar el software. Los usuarios pueden manipular y controlar el software y hardware por medio de las interfaces.

El hecho de que realicemos un buen diseño de las interfaces de usuario, puede hacer que nuestra aplicación pueda llegar a triunfar, ya que como hemos dicho antes, es el medio por el que los usuarios van a utilizar nuestro software.

Su objetivo es que las aplicaciones o los objetos sean más atractivos y, además, hacer que la interacción con el usuario sea lo más intuitiva posible, conocido esto como el diseño centrado en el usuario [36].

En este diseño debemos de favorecer la usabilidad y para eso tenemos que cumplir sus principios básicos, que son:

- Facilidad de aprendizaje.
- Flexibilidad.
- Robustez.
- Consistencia.
- Agradable.
- Atractiva.

6.2.1. Metáforas.

En el diseño de interfaces para los desarrollos de productos software es muy frecuente tener que intentar mostrar con imágenes conceptos abstractos para indicar la funcionalidad o tareas que podemos realizar con las aplicaciones, por eso está muy extendido el uso de metáforas.

Las buenas metáforas crean figuras mentales fáciles de recordar. Las interfaces pueden contener objetos asociados al modelo conceptual en forma visual, con sonido u otra característica perceptible por el usuario que ayude a simplificar el uso del sistema.

A continuación, vamos a explicar las distintas metáforas que hemos utilizado en nuestra aplicación:

Pantalla de estado de ánimo.

Esta metáfora se utiliza cuando queremos que el usuario elija un estado de ánimo para obtener la recomendación con contexto de ánimo. Para ello hemos utilizado sistemas de coordenadas, en los cuales se encuentran los distintos estados de ánimo. Para facilitar esta metáfora, hemos divido ese eje de coordenadas en nueve casillas uniformes y según esté ubicada una canción dentro de los ejes, corresponderá a un estado de ánimo u otro.

Las dos magnitudes que se intentan relacionar en esta metáfora son el eje de la “y”, el cual representa si el estado de ánimo es más enérgico o calmado, y el eje

de la “x”, que representa si el estado es más negativo o positivo, siendo el origen de coordenadas, que es la casilla central, un estado de ánimo neutral.

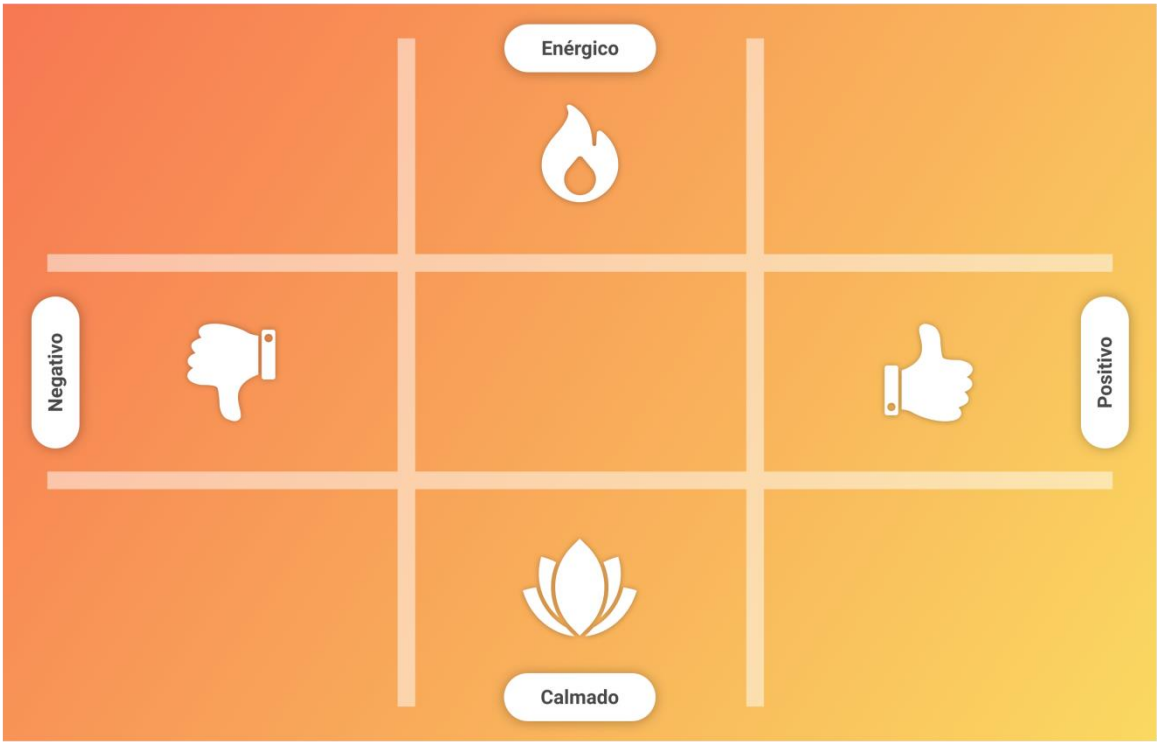

Ilustración 6.3 Metáfora para estados de ánimo.

Canción favorita.

Esta metáfora aparece en todas las pantallas en las que se muestran las listas de canciones, ya sean recomendadas o no. Se compone de dos imágenes, las cuales indican si la canción está marcada como favorita o no. El uso del corazón para esta metáfora está muy extendido en nuestros días, ya que la mayoría de aplicaciones que usamos a diario utilizan también esta metáfora.

Los dos estados en los que nos podemos encontrar esta metáfora son, por un lado, un corazón en el que su interior está vacío, el cual nos hace entender que no está marcado, y por otra parte, el siguiente estado es un corazón completamente rosa, el cual indica que se ha marcado como favorita una determinada canción.

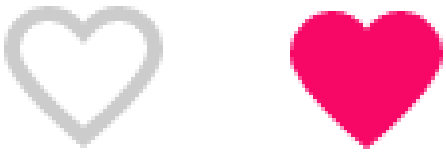

Ilustración 6.4 Iconos para la metáfora de favorito.

Valoración.

Para esta metáfora se ha utilizado la misma lógica que para la de favoritos, ya que se trata de una funcionalidad muy extendida en las aplicaciones para móviles, e incluso las propias librerías de interfaces de Android proporcionan herramientas para su uso. En nuestro caso, se han utilizado las estrellas y, como solo podemos valorar del 1 al 5, mostramos cinco estrellas.

Los diferentes estados que podemos tener de esta metáfora son cinco estrellas sin rellenar, lo cual indica que no se ha realizado ninguna valoración de la canción, y los siguientes estados corresponden a la valoración dada por el propio usuario, ya que según la puntuación se rellenarán de amarillo de 1 a 5 las estrellas.

Ilustración 6.5 Iconos para la metáfora de la valoración.

6.2.2. Prototipos.

En este punto vamos a diseñar un prototipo de nuestro sistema. Como bien sabemos, nuestro proyecto se compone de dos partes: un servicio web y una aplicación Android.

El servicio web no tiene ninguna interfaz, por lo tanto, no es necesario realizar un prototipo, en cambio, para la aplicación sí que es necesario. Para realizar este prototipo hemos decidido utilizar la técnica de definición de pantallas. Hay que recordar que el diseño de prototipos es una idea de lo que queremos que contenga nuestra interfaz, la cual no tiene por qué ser final y puede ser muy diferente según las necesidades que se necesiten en implementación.

- **Pantalla de inicio de sesión:**

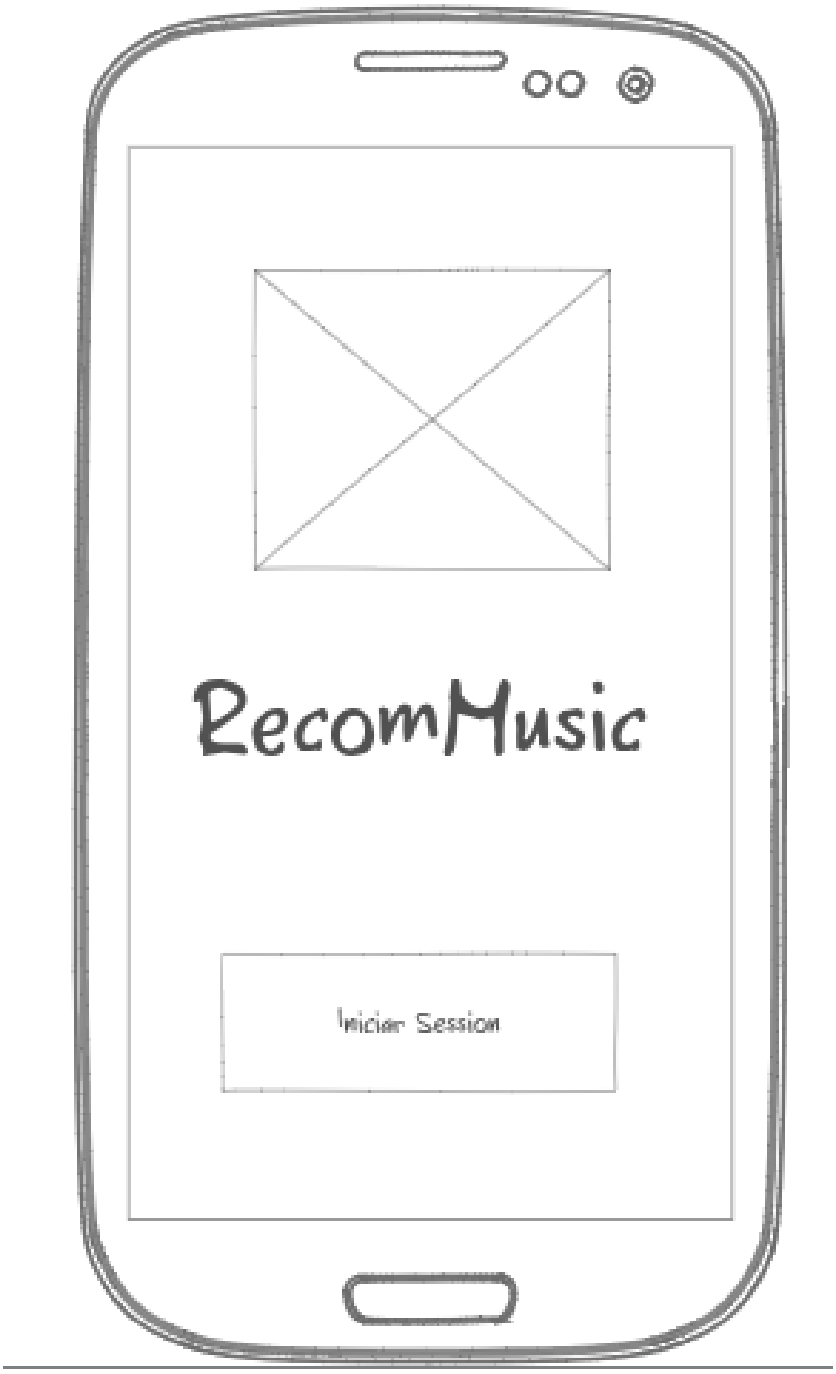

Ilustración 6.6 Prototipo de pantalla de inicio de sesión.

- **Pantalla de inicio o menú:**

Ilustración 6.7 Prototipo de pantalla de inicio.

- **Pantalla de lista de canciones:**

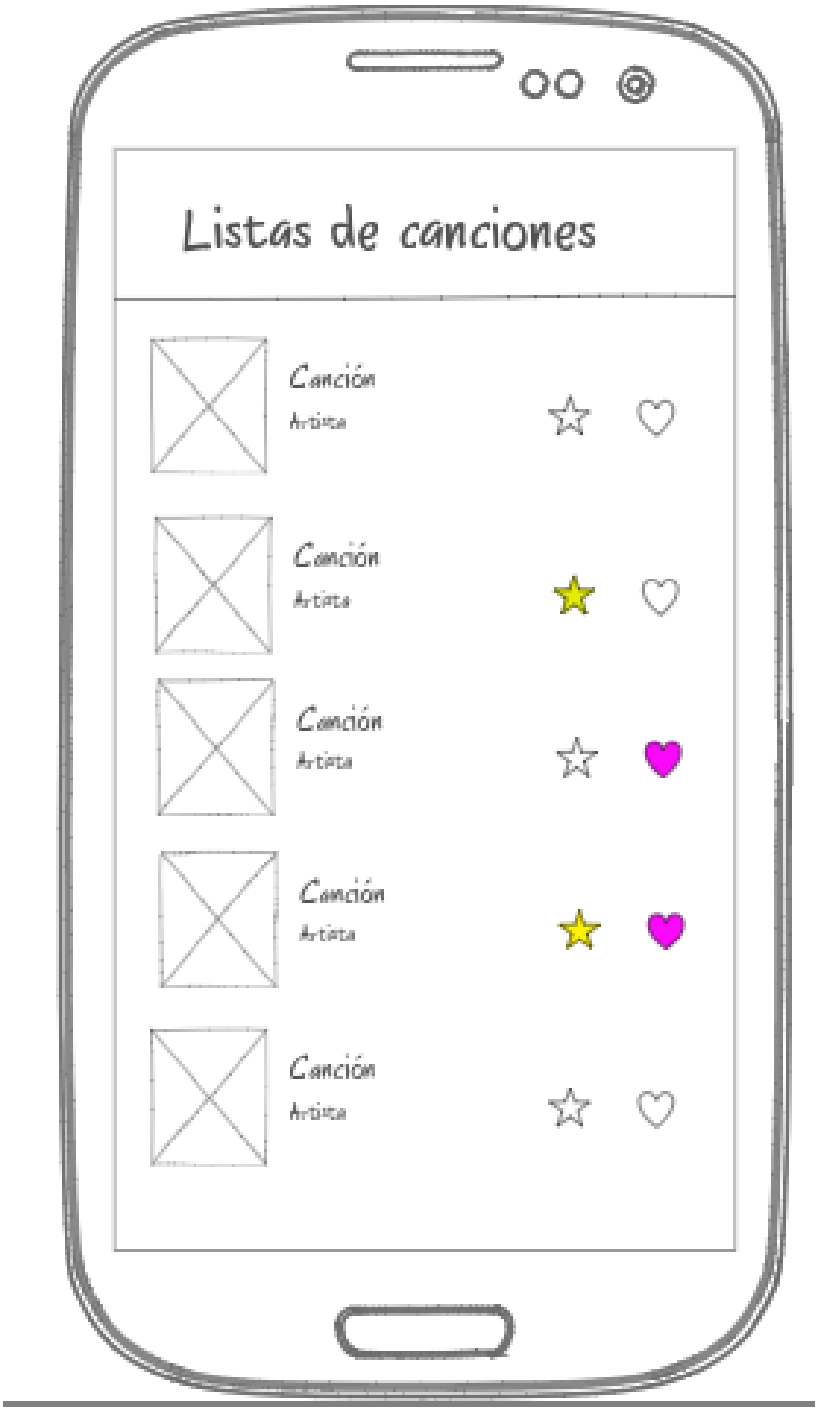

Ilustración 6.8 Prototipo de pantalla de lista de canciones.

- **Pantalla de lista de canciones vacía:**

Ilustración 6.9 Prototipo de pantalla de lista de canciones vacía.

- **Pantalla de error de conexión con servidor:**

Ilustración 6.10 Prototipo de pantalla de error de conexión con el servidor.

- **Pantalla de selección de estado de ánimo:**

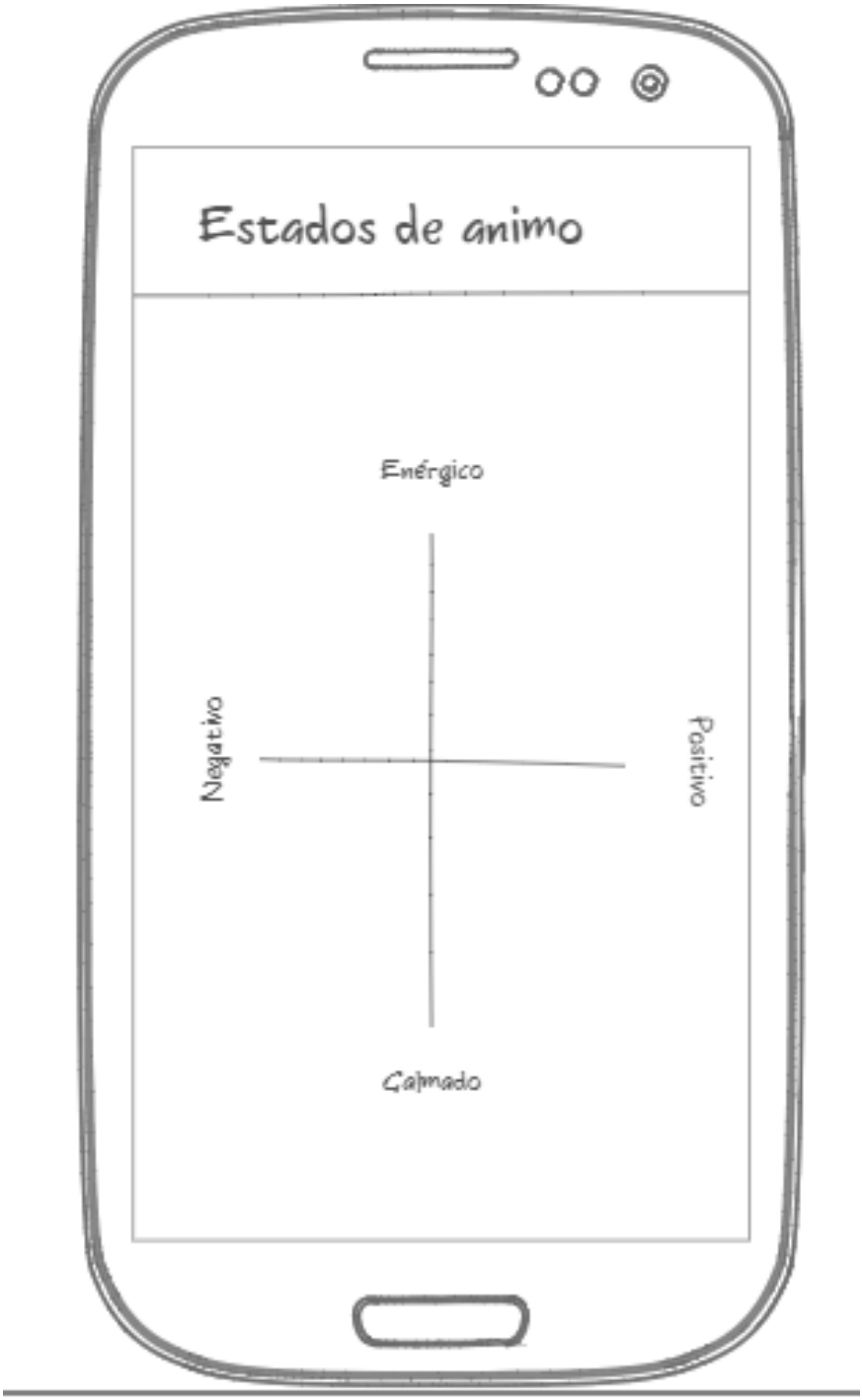

Ilustración 6.11 Prototipo de pantalla de selección de estado de ánimo.

- **Pantalla “Mi cuenta”:**

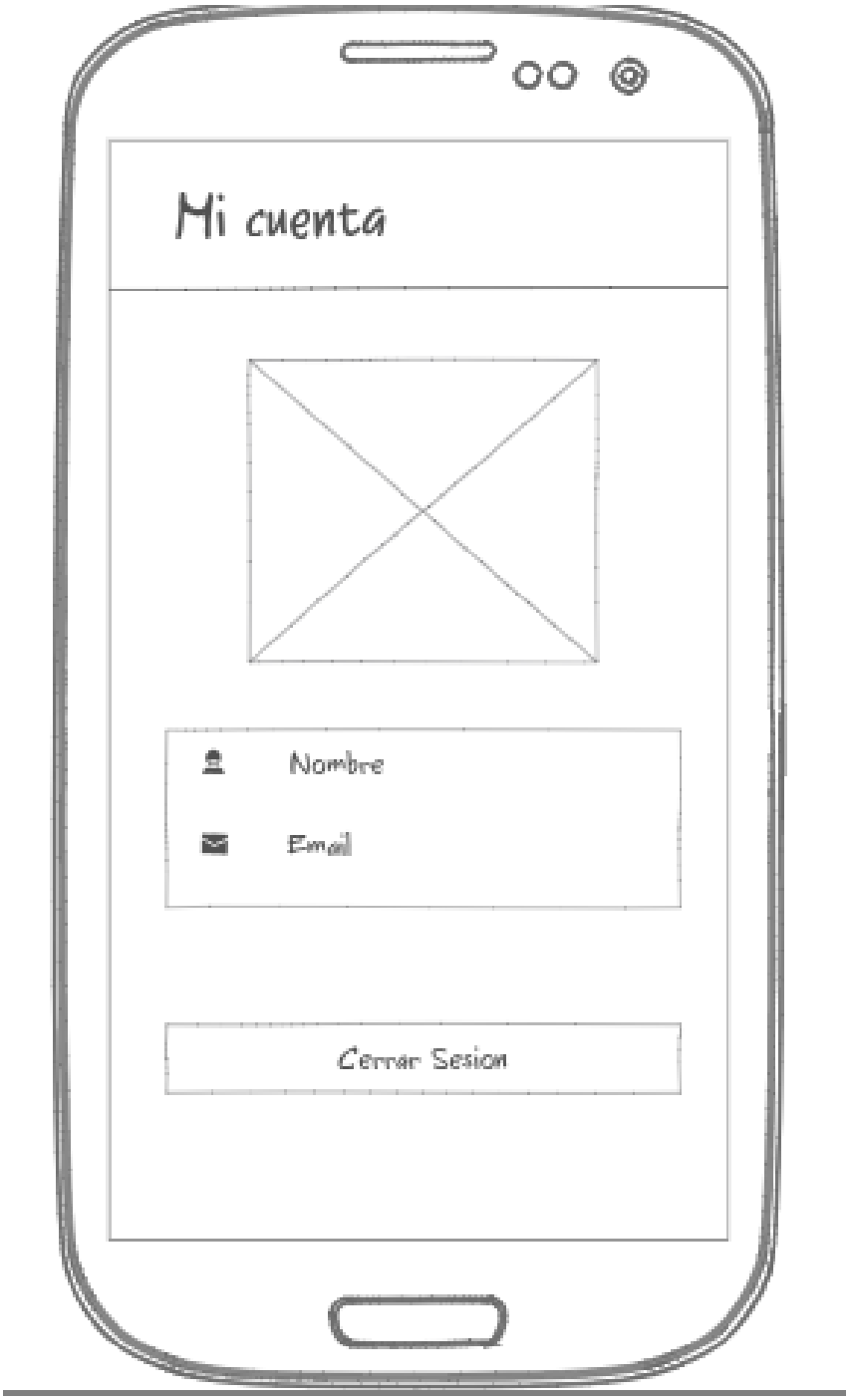

Ilustración 6.12 Prototipo de pantalla "Mi cuenta".

6.2.3. Caminos de navegación.

Los caminos de navegación se utilizan para poder dar una visión algo más dinámica de la interacción que los usuarios van a realizar en nuestra aplicación, con lo cual podremos estudiar si nuestra interfaz es usable.

La herramienta que vamos a utilizar para definir los caminos de navegación es el storyboard, que básicamente consiste en generar una secuencia de las pantallas generadas en la fase de prototipado, a través de las acciones que se realizan en cada una de ellas.

Los storyboards también sirven para realizar el test de interfaz con los usuarios para poder mejorar la usabilidad y detectar patrones comunes de los usuarios y algunos errores de la interfaz. La validación de los storyboards por parte de los usuarios podrá reducir costes de tiempo y económicos de la fase de implementación.

Los distintos flujos que vamos a mostrar en el storyboard son los siguientes:

- Inicio de sesión o registro.
- Ver lista de canciones recomendadas.
- Ver lista de canciones recomendadas según el estado de ánimo.
- Ver lista de canciones aleatorias.
- Ver lista de favoritos.
- Ver datos del perfil del usuario.
- Marcar como favorito.
- Puntuar una canción.

- **Storyboard de incio de sesión o registro.**

Para registrarse o iniciar sesión en aplicación son necesarias las siguientes acciones:

1. El usuario pincha en el botón de iniciar sesión.
2. Se mostrará una pantalla con las distintas cuentas asociadas.
3. El usuario selecciona una cuenta.
4. La aplicación mostrará el menú.

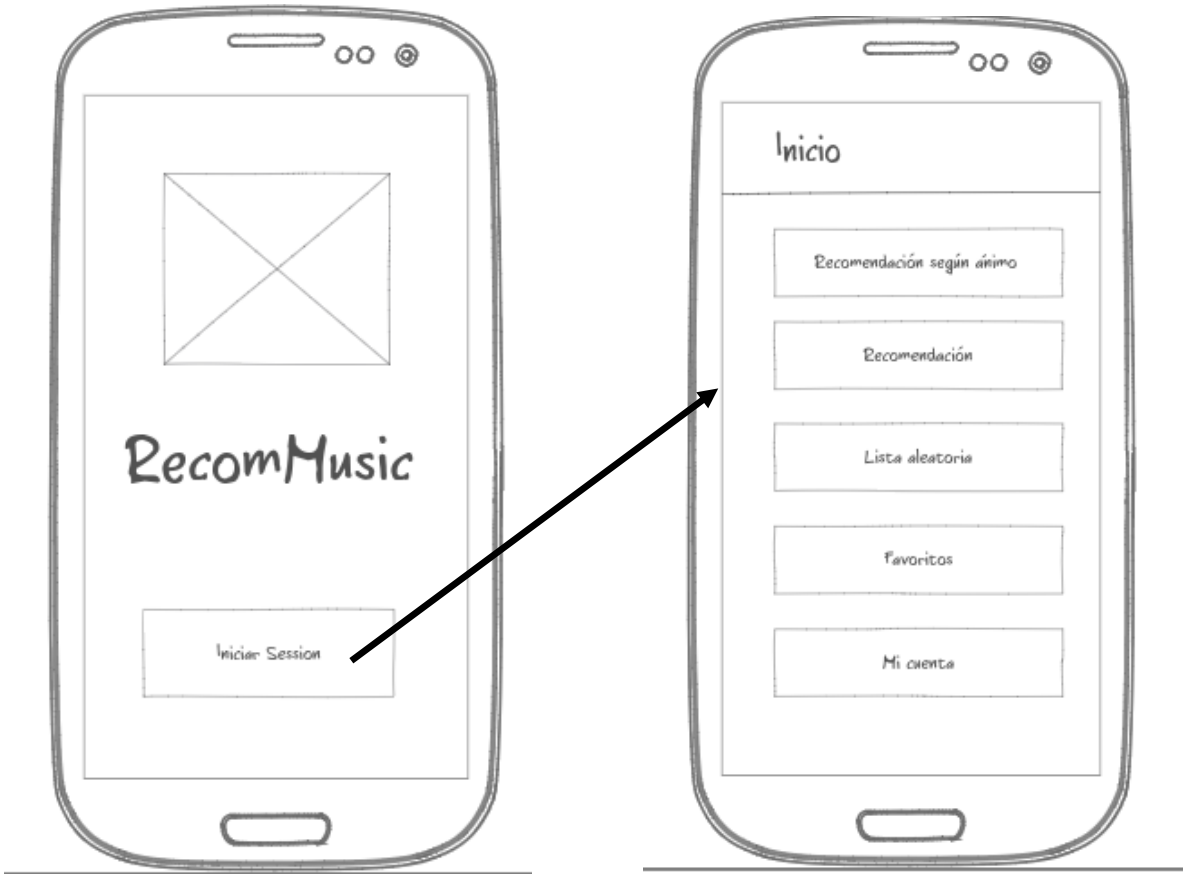

Ilustración 6.13 Storyboard de inicio de sesión o registro.

- **Storyboard de lista de canciones recomendadas.**

Para poder ver la lista de canciones recomendadas es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón de recomendación.
3. Se muestra la lista de canciones recomendadas.

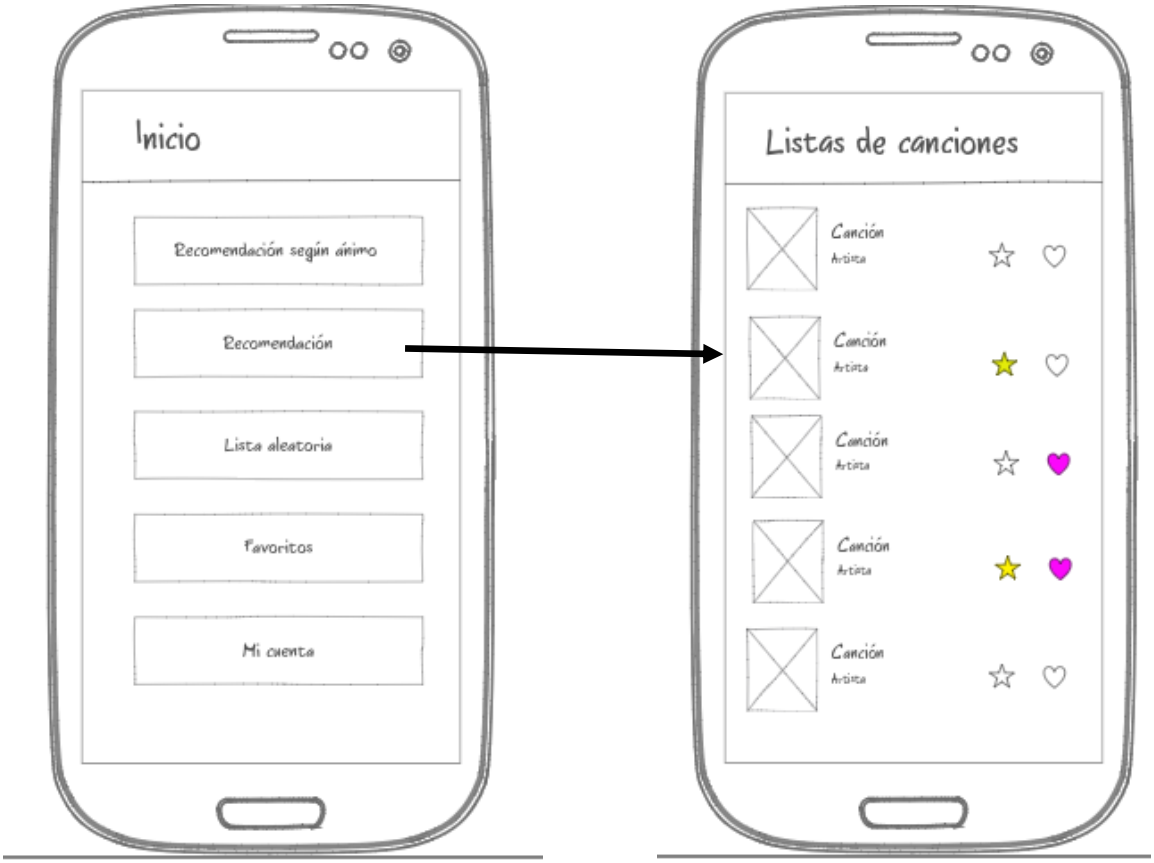

Ilustración 6.14 Storyboard de lista de canciones recomendadas.

- **Storyboard de recomendación según el estado de ánimo.**

Para poder ver las listas de canciones recomendadas según el contexto de ánimo es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón de recomendación según ánimo.
3. La aplicación muestra los distintos estados de ánimo.
4. El usuario pulsa el estado de ánimo deseado.
5. Se muestra la lista de canciones recomendadas según el contexto.

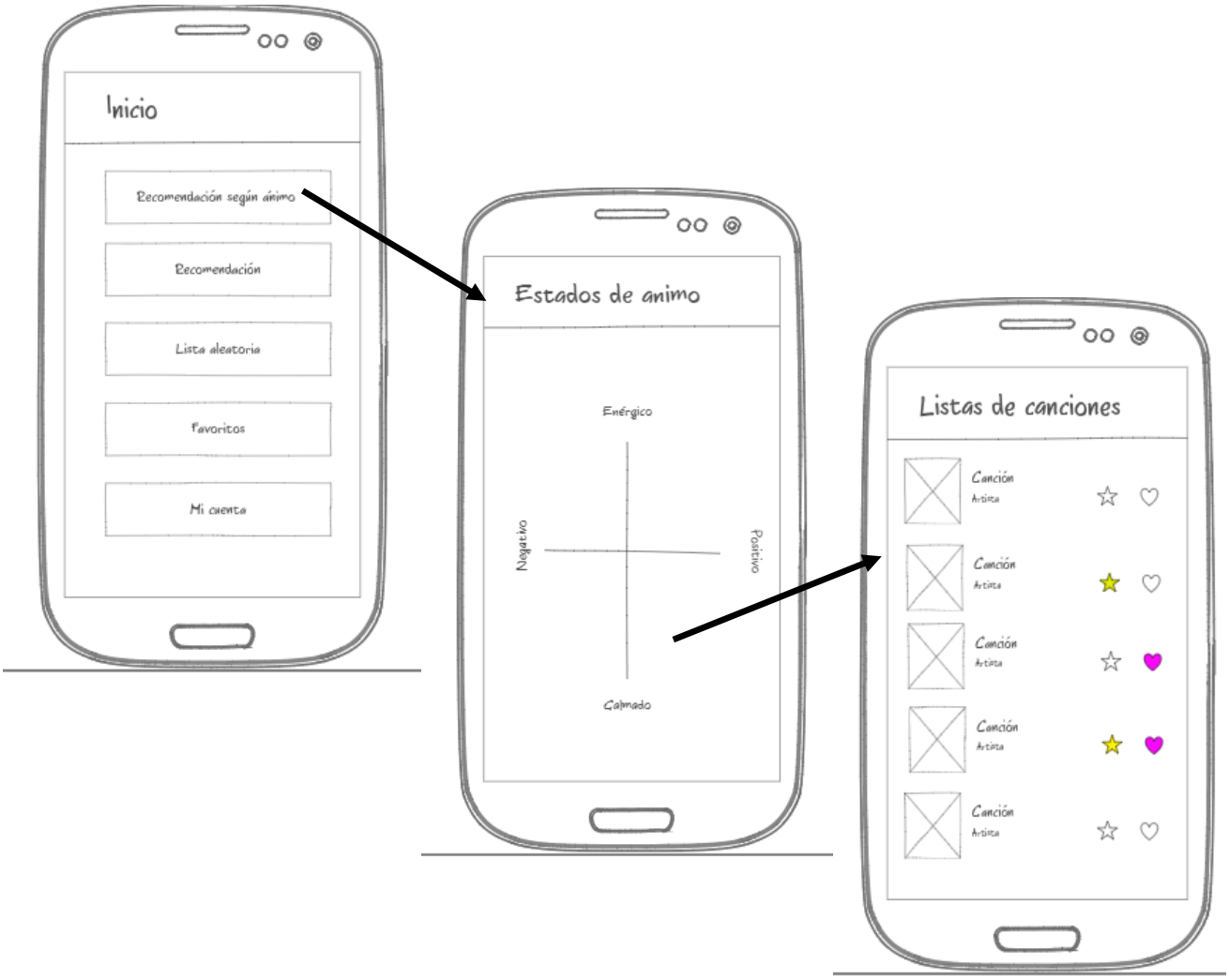

Ilustración 6.15 Storyboard de recomendación según el contexto de ánimo.

- **Storyboard de lista de canciones aleatorias.**

Para poder ver la lista de canciones aleatorias es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón “lista aleatoria”.
3. Se muestra la lista de canciones aleatorias.

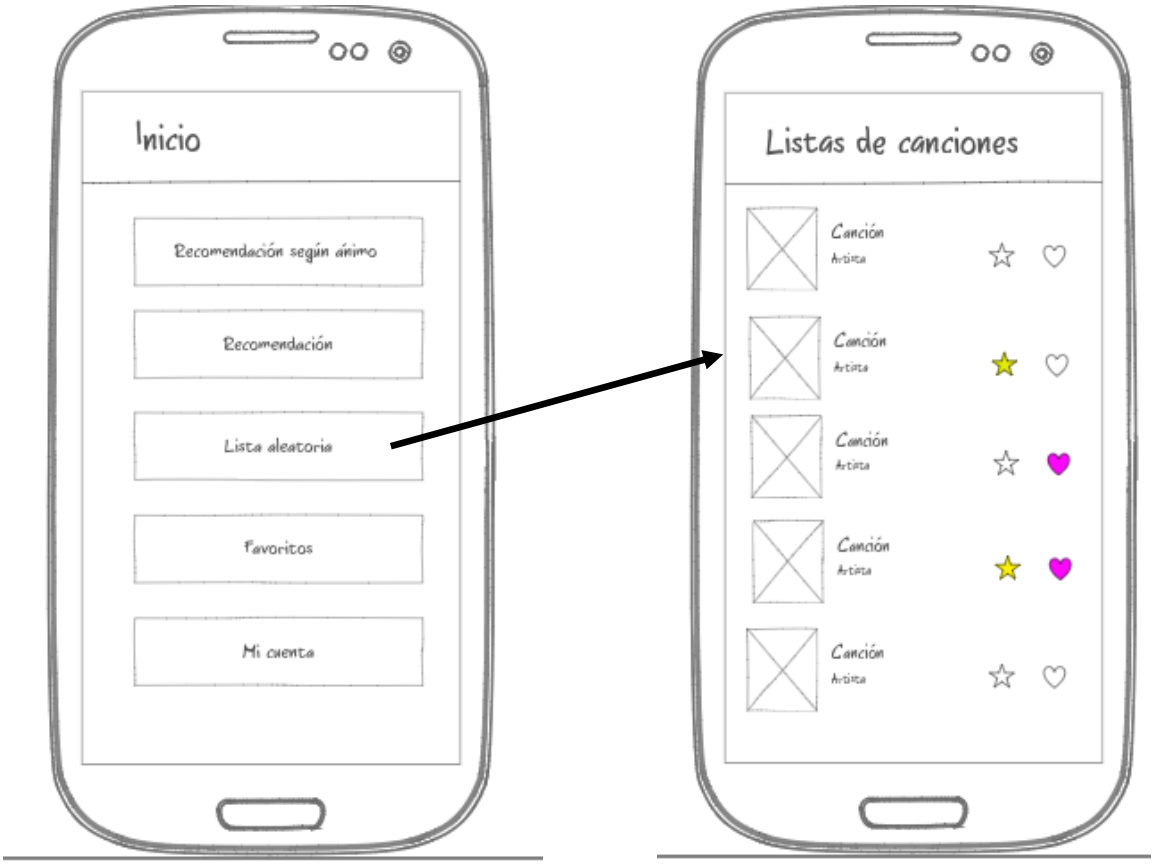

Ilustración 6.16 Storyboard de lista de canciones aleatorias.

- **Storyboard de lista de canciones favoritas.**

Para poder ver la lista de canciones favoritas es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón de favoritos.
3. Se muestra la lista de canciones favoritas.

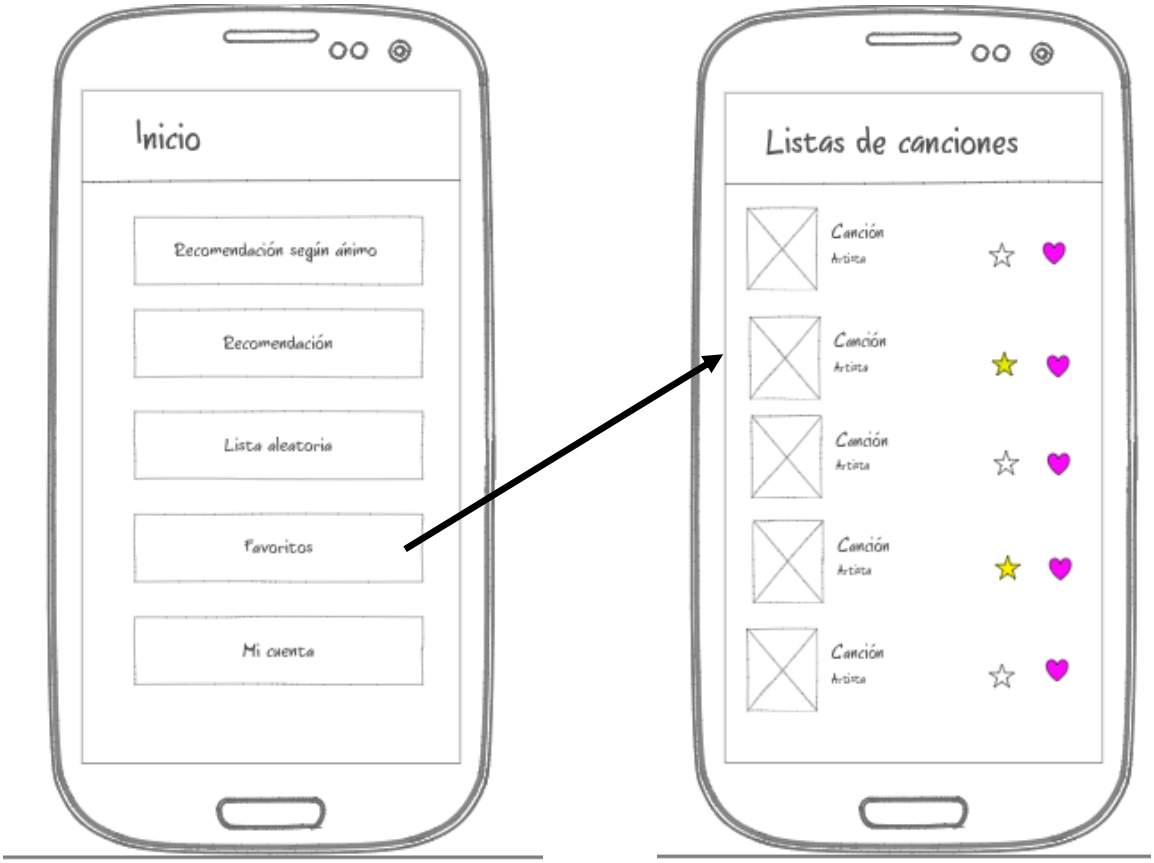

Ilustración 6.17 Storyboard de favoritos.

- **Storyboard del perfil de usuario.**

Para poder ver el perfil del propio usuario es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón “Mi cuenta”.
3. Se muestran los datos del perfil.

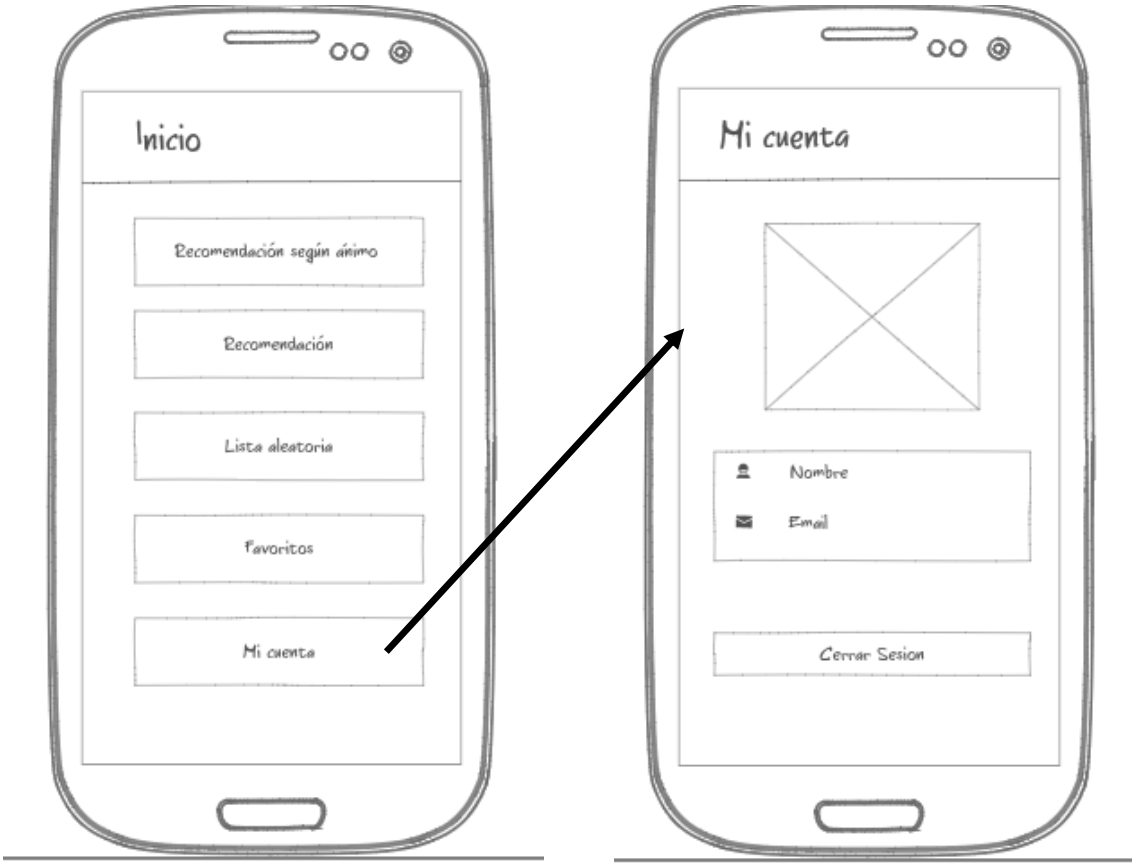

Ilustración 6.18 Storyboard de “Mi cuenta”.

- **Storyboard de marcar una canción como favorita.**

Para poder marcar o desmarcar una canción como favorita es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario puede pulsar sobre cualquier botón que genere una lista de canciones.
3. Se muestra la lista de canciones.
4. El usuario pulsa sobre el icono del corazón de una de las canciones.
5. Si el corazón está desmarcado, la aplicación lo pinta de rosa, y si no lo está, realiza la función al contrario.

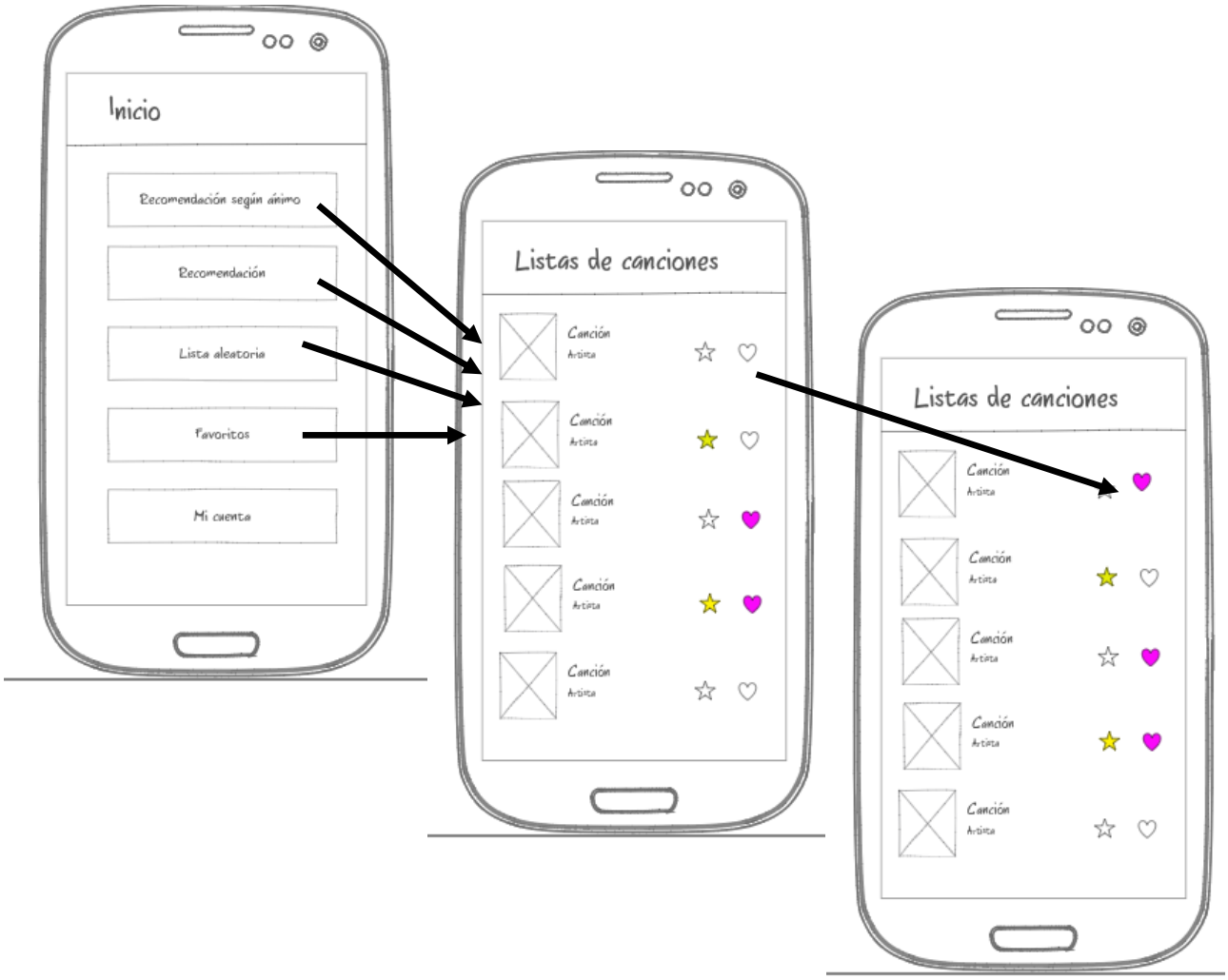

Ilustración 6.19 Storyboard de marcar/desmarcar favorito.

- **Storyboard de valorar una canción.**

Para poder valorar una canción es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario puede pulsar sobre cualquier botón que genere una lista de canciones.
3. Se muestra la lista de canciones.
4. El usuario pulsa sobre el icono de la estrella y asigna una puntuación.
5. La estrella pasará a pintarse de amarillo.

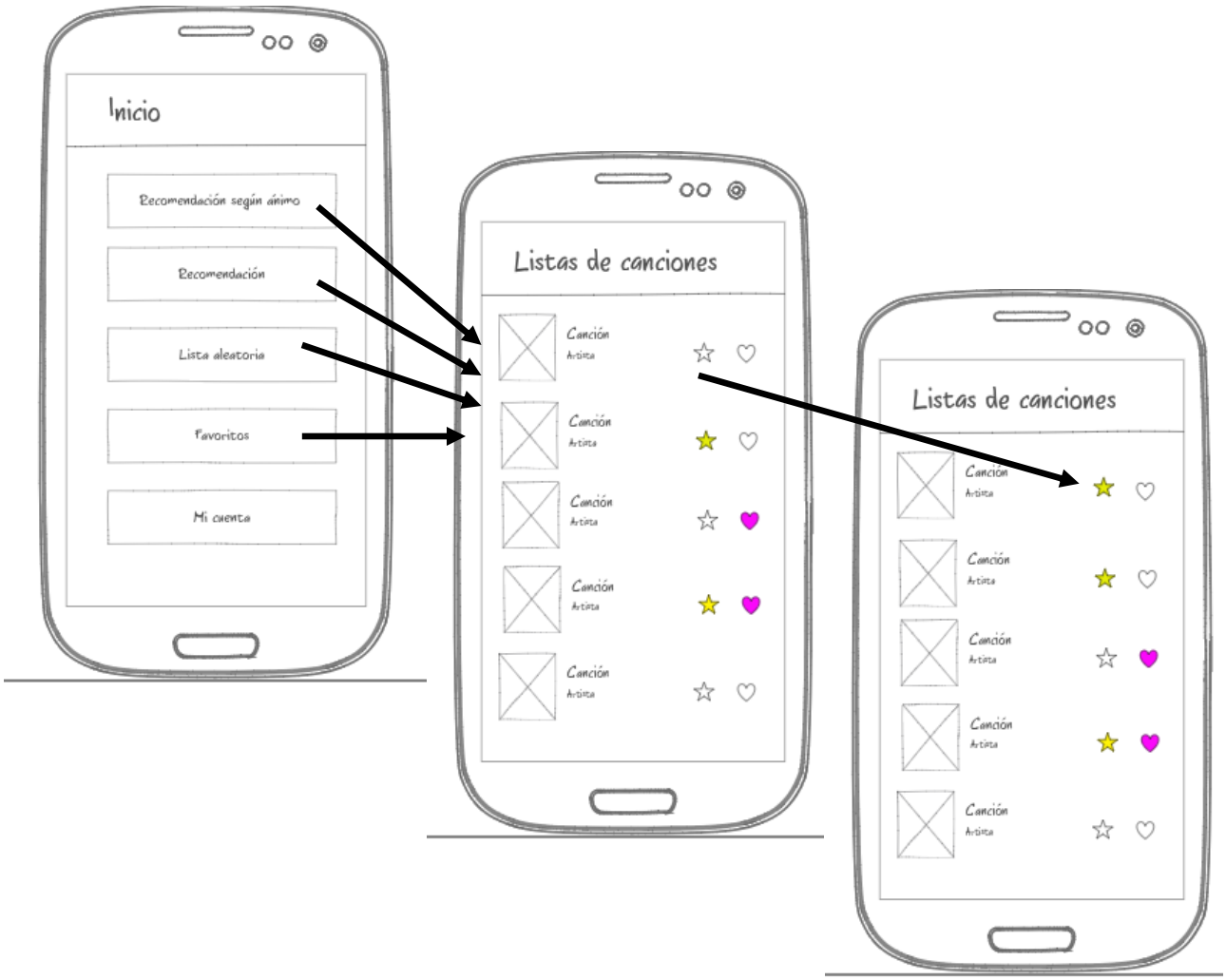

Ilustración 6.20 Storyboard de valorar una canción.

- **Storyboard de cerrar sesión.**

Para poder cerrar sesión es necesario realizar las siguientes acciones:

1. Inicia la aplicación.
2. El usuario pulsa sobre el botón “Mi cuenta”.
3. Se muestran los datos del perfil.
4. El usuario pulsa en el botón de cerrar sesión.
5. La aplicación muestra la pantalla de inicio de sesión.

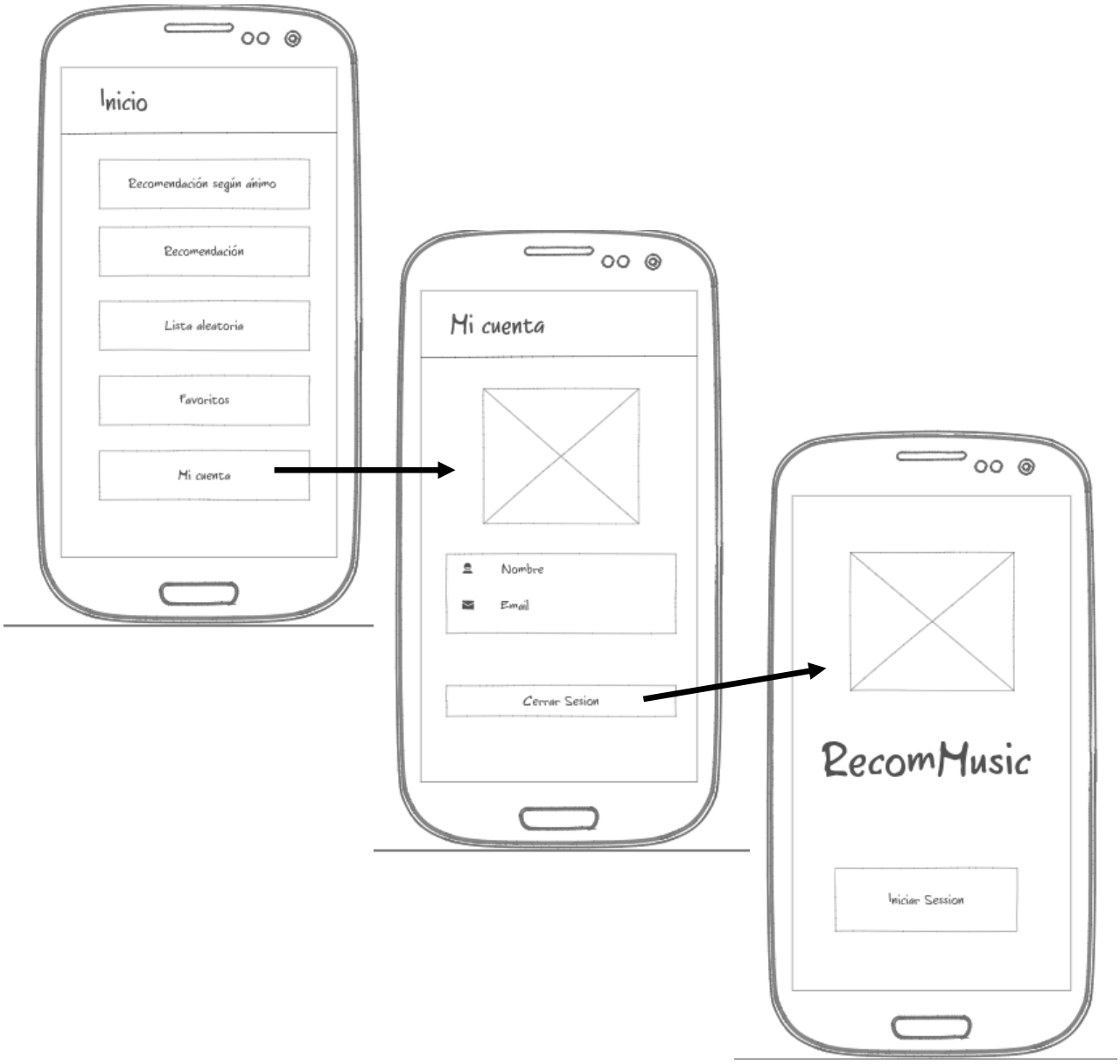

Ilustración 6.21 Storyboard de cerrar sesión.

Capítulo 7:

IMPLEMENTACIÓN Y PRUEBAS.

7. IMPLEMENTACIÓN Y PRUEBAS.

Este punto pone fin al proceso de la Ingeniería del Software. Ahora es cuando todo el proceso de análisis y desarrollo anteriormente realizado se va transformar en código fuente. También realizaremos pruebas para validar que la implementación realizada es de calidad y cumple con los objetivos y requerimientos funcionales propuestos en la fase de análisis.

Antes de empezar con la implementación, tendremos que tomar decisiones sobre qué lenguajes de programación utilizaremos, qué arquitectura es la más idónea para nuestro sistema, así como qué frameworks y librerías nos ayudarán en esta etapa. Todas estas decisiones influirán en la calidad del software que realicemos, además de los costes económicos y de tiempo.

Para ponernos en contexto, vamos a describir los elementos que contiene nuestro TFG:

- *Servicio web:* se encargará de proporcionar mecanismos de comunicación entre el servidor y el cliente.
- *Aplicación Android:* será el medio de comunicación entre los usuarios y el servidor, donde los usuarios podrán ver las recomendaciones.
- *Base de datos:* es la capa de persistencia de los datos con los que funciona nuestro sistema.
- *Algoritmo de filtrado colaborativo:* realizará el cálculo de las recomendaciones musicales para los distintos usuarios.

Para poder llevar a cabo nuestro TFG, vamos a basarlo en una arquitectura cliente/servidor. Como su propio nombre indica, esta arquitectura se divide la responsabilidad en dos grandes bloques, uno es la parte servidora y otro la cliente. El algoritmo de filtrado colaborativo y la base de datos se encuentran en la parte del servidor, que en nuestro caso será un servidor central para todos los clientes, y la parte cliente es nuestra aplicación Android. Estas dos partes se comunicarán a través de un servicio REST que se encuentra en el servidor y la cual expone una API para su comunicación con los clientes.

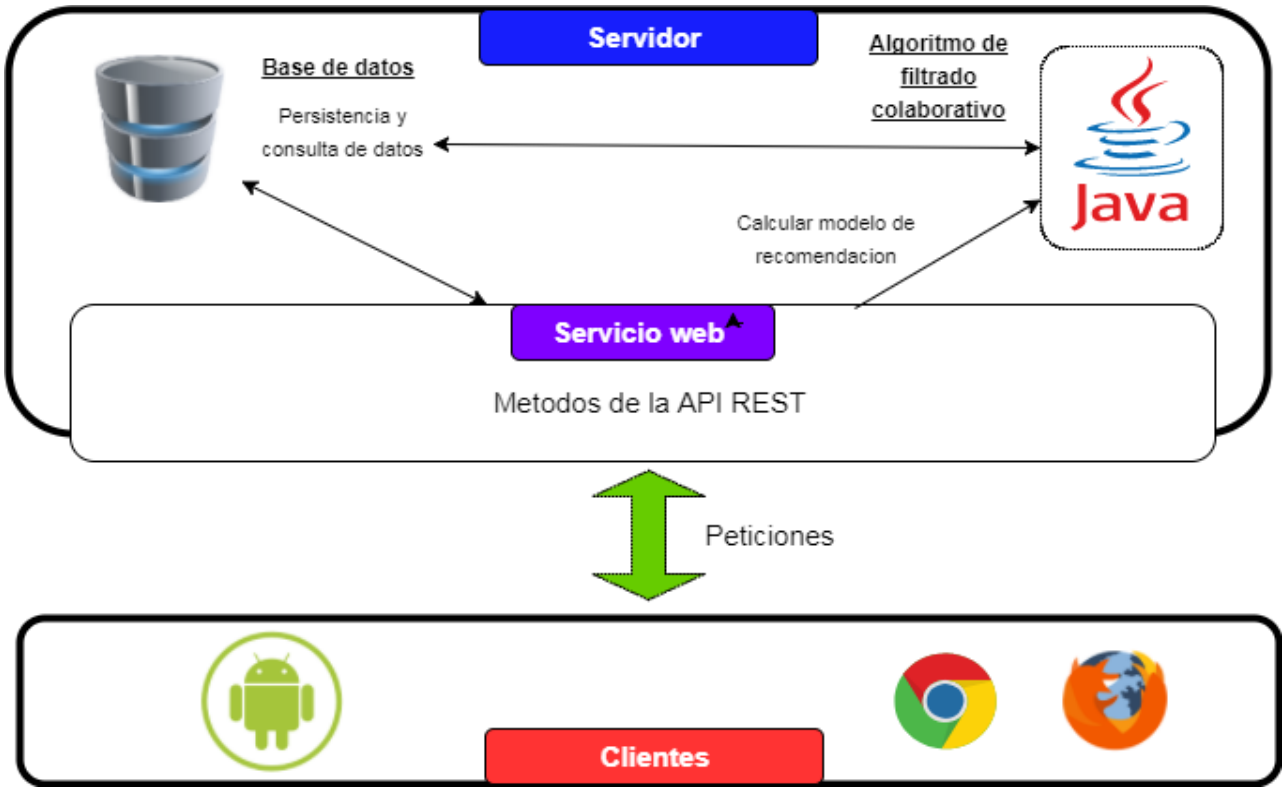

Ilustración 7.1 Arquitectura cliente/servidor del proyecto.

7.1. Lenguajes de programación utilizados.

Al seguir una arquitectura cliente/servidor tenemos que separar las decisiones en base al cliente y al servidor, por eso los lenguajes utilizados pueden ser diferentes.

7.1.1. Aplicación cliente.

Al ser una aplicación en Android, los lenguajes que hemos utilizado son Java y XML [24].

- *Java:* es un lenguaje orientado a objetos de propósito general. Su fuerte radica en su código, que es compilado a bytecode, el cual es el lenguaje que es ejecutable en la Java Virtual Machine (JVM), lo cual hace que cualquier programa Java sea multiplataforma e independiente del hardware.

En la aplicación Android se utiliza el lenguaje Java para modelar toda la lógica de la aplicación.

- *XML:* es un meta-lenguaje que permite definir lenguajes de marcas y se utiliza para almacenar datos en forma legible. En Android es utilizado para la definición de las interfaces de las pantallas.

7.1.2. Aplicación servidor.

En la aplicación servidor desarrollaremos un servicio web que tendrá una API REST. Para ello vamos a utilizar el lenguaje PHP (Hypertext Preprocessor) que es un lenguaje de programación de código abierto. Además, vamos a utilizar un framework basado en este lenguaje el cual no facilitará la implementación del servicio web.

Para realizar esta aplicación podríamos haber utilizado otros lenguajes y frameworks, pero ahora pasaré a detallar qué ventajas nos han hecho decidirnos por estos.

Las ventajas de PHP:

- Es un lenguaje multiplataforma.
- Es un lenguaje de Open Source.
- Existe una gran comunidad.
- Existe gran soporte, ya que podemos encontrar numerosas guías, ejemplos y librerías para el desarrollo.
- Contiene diversos conectores de bases de datos.
- Fácil modularización del código.

Las ventajas de Slim:

- Soporte para métodos HTTP.
- Fácil instalación y configuración.
- Renderización de plantillas y vistas personalizadas.
- Mensajes Flash.
- Uso de caché HTTP.
- Logs personalizados.
- Gestión de errores.

7.2. Definición de la API.

En este punto vamos a definir las rutas que expondrá el servidor para que los clientes puedan acceder a los recursos que ofrece nuestro servidor. Esta capa proporciona independencia para la parte del cliente, ya que cualquier aplicación cliente en cualquier plataforma puede comunicarse con nuestro servidor a través de los métodos proporcionados por la API (Interfaz de Programación de Aplicaciones).

Existen varios tipos de APIs pero nosotros nos centraremos en las de servicios web, que son las que nos afectan.

Las APIs de servicios web son las interfaces de desarrollo de aplicaciones que permiten el intercambio de información entre un servicio web y una aplicación cliente. Usualmente este intercambio se realiza a través de peticiones HTTP o HTTPS. Tanto en la petición como en la respuesta se proporciona información y metadatos normalmente en formato XML o JSON [17,18].

A continuación, mostraremos una tabla con las distintas rutas que ofrece nuestra API REST:

URI	Verbo	Parámetros	Acción
/login	POST	Email	Dado el email obtenido en los parámetros, devuelve su api key. En el caso de no existir el email lo añade a la BBDD.
/recomendations	GET	Api key Opcional: context	Devuelve una lista de recomendaciones según el usuario de la api key. En caso de añadir como parámetro el campo context, devuelve la lista de recomendación según la zona del contexto.
/recomendations/calculate	GET	Api key	Calcula un nuevo modelo de recomendación para el usuario propietario de la api key.
/random	GET	Api key	Devuelve una lista de canciones aleatoria según el usuario de la api key.
/favourites	GET	Api key	Devuelve la lista de canciones favoritas marcadas por el usuario de la api key.
/ratings	POST	Api key, rating, IdCancion	Genera una valoración para la canción y el usuario al que pertenezca la api key.

Tabla 7.1 Rutas de la API REST.

7.3. Obtención de información de la BBDD.

Al tener que desarrollar un sistema de recomendación, tal y como hemos podido ver en el capítulo 2, uno de los requisitos indispensables es obtener información sobre los ítems, ya que son los elementos que vamos a recomendar a los usuarios. Para que el sistema de recomendación sea de calidad, debemos de tener un gran número de canciones junto con sus características.

La idea inicial era conseguir que las canciones estuvieran bajo la licencia Creative Commons para poder hacer uso de todos sus datos. Para ello, estuvimos analizando la plataforma Jamendo, ya que ofrece canciones con licencia de distribución libre, pero al obtener dichas canciones vimos que no nos ofrecían ningún tipo de información para poder obtener datos sobre el estado de ánimo que refleja la canción.

Todo esto nos hizo pensar en la posibilidad de que cada usuario fuera valorando el estado que según a él mismo le sugiriera cada canción para poder así recomendar las canciones de manera sensible al contexto de ánimo. Con esta opción vimos que iba a ser un proceso muy laborioso para el usuario y que no íbamos a poder ofrecer desde el inicio unas buenas recomendaciones. Todo esto hizo que tuviéramos que analizar otras plataformas como Musicovery.

La plataforma Musicovery tenía una gran base de datos de canciones que contenían información para poder definir un estado de ánimo de cada canción, pero nos encontramos con el problema de que no eran canciones de distribución libre, por lo que el audio de la canción o cierta información sobre la misma, no nos era proporcionada.

Además de ser una herramienta comercial, no nos permitía el acceso total a sus datos y teníamos un número muy limitado de peticiones que podíamos realizar. Como este proyecto es un proyecto académico, nos pusimos en contacto con ellos y, aunque no nos dieron un acceso total a sus canciones, nos permitieron realizar peticiones a su servicio para obtener información sobre unas 5.000 canciones.

Con la información obtenida podríamos obtener el estado de ánimo pero con el inconveniente de no poder obtener el audio. Para obtener esa información se realizó una pequeña aplicación que consistía en un cliente que realizaba peticiones a la API de Musicovery y volcaba la información obtenida a la base de datos de nuestro servidor.

7.4. Pruebas y validación.

Después de implementar nuestro sistema, ya solo nos queda realizar el testeo de nuestro software para comprobar la calidad del mismo.

En este punto realizaremos unas pruebas de caja negra, ya que ésta técnica nos permite abstraernos de la estructura interna del código, de los detalles de implementación y solo se centra en verificar la funcionalidad.

Usaremos los requerimientos funcionales del apartado 5.1.1 para establecer los casos a la hora de realizar las pruebas de caja negra.

7.4.1. Casos de prueba.

- *Prueba 1:* Registro de usuario [RF1].

Pre-condiciones	El usuario debe de tener una cuenta de Google+.
Acción	El usuario pulsa el botón de “Inicio de sesión” y elige una cuenta.
Resultado esperado	El sistema añade sus datos y muestra la pantalla de inicio.

- *Prueba 2:* Inicio de sesión correctamente [RF1].

Pre-condiciones	El usuario debe tener una cuenta de Google+ y estar registrado.
Acción	El usuario pulsa el botón de “Inicio de sesión” y elige una cuenta.
Resultado esperado	El sistema valida sus datos y muestra la pantalla de inicio.

- *Prueba 3:* Obtener recomendaciones musicales personalizadas [RF2].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en el menú.
Acción	El usuario pulsa en el botón de “Recomendación”.
Resultado esperado	Se muestran la lista de canciones recomendadas.

- *Prueba 4:* Obtener el estado de ánimo del usuario [RF3].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en el menú.
Acción	El usuario pulsa en el botón de “Recomendación con contexto de ánimo”.
Resultado esperado	La aplicación muestra una pantalla donde se ven reflejados nueve estados de ánimo para que sean seleccionados.

- *Prueba 5:* Recomendar lista de música personalizada sensible al contexto de ánimo [RF4].

Pre-condiciones	El usuario debe haber iniciado sesión y encontrarse en la pantalla de selección de estado de ánimo.
Acción	El usuario selecciona un estado de ánimo.
Resultado esperado	La aplicación muestra la lista de canciones recomendada sensible al estado de ánimo seleccionado.

- *Prueba 6:* Obtener lista de canciones aleatorias [RF5].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en el menú.
Acción	El usuario pulsa en el botón de “Lista aleatoria”.
Resultado esperado	La aplicación muestra una lista de canciones de modo aleatorio.

- *Prueba 7:* Mostrar lista de canciones favoritas [RF6].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en el menú.
Acción	El usuario pulsa en el botón de “Favoritos”.
Resultado esperado	La aplicación muestra la lista de canciones favoritas.

- *Prueba 8:* Marcar una canción como favorita [RF7].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en cualquiera de las pantallas con canciones.
Acción	El usuario pulsa sobre el corazón gris de una canción.
Resultado esperado	El corazón cambia a un color rosa y la canción se añade en el sistema como favorita para el usuario.

- *Prueba 9:* Desmarcar una canción como favorita [RF7].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en cualquiera de las pantallas con canciones.
Acción	El usuario pulsa sobre el corazón rosa de una canción.
Resultado esperado	El corazón cambia a un color gris y la canción se elimina en el sistema como favorita para el usuario.

- *Prueba 10:* Añadir una valoración de un usuario a una canción [RF8].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en cualquiera de las pantallas con canciones.
Acción	El usuario pulsa sobre las estrellas de una canción.
Resultado esperado	El número de estrellas seleccionadas cambian de color y el sistema añade esa valoración.

- *Prueba 11:* Modificar una valoración de un usuario a una canción [RF8].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en cualquiera de las pantallas con canciones.
Acción	El usuario pulsa sobre las estrellas de una canción ya valorada.
Resultado esperado	El número de estrellas seleccionadas cambian de color y el sistema modifica esa valoración.

- *Prueba 12:* Eliminar una valoración de una canción para un usuario [RF8].

Pre-condiciones	El usuario debe haber iniciado sesión y estar en cualquiera de las pantallas con canciones.
Acción	El usuario desliza el dedo hacia la izquierda sobre las estrellas de una canción ya valorada.
Resultado esperado	Las estrellas se vuelven grises y el sistema elimina la valoración.

- *Prueba 13:* Actualizar el modelo de recomendación al iniciar la aplicación [RF9].

Pre-condiciones	El usuario debe de tener una cuenta de Google+ y estar registrado.
Acción	El usuario pulsa el botón de “Inicio de sesión” y elige una cuenta.
Resultado esperado	El sistema actualiza el modelo de recomendación y cuando termina muestra un mensaje.

- *Prueba 14:* Cerrar sesión [RF10].

Pre-condiciones	El usuario debe haber iniciado sesión.
Acción	El usuario se dirige hacia la pantalla de “Mi Cuenta” y pulsa el botón de cerrar sesión.
Resultado esperado	La aplicación muestra la pantalla de inicio de sesión.

7.4.2. Resultados obtenidos.

Prueba	Resultado
Prueba 1	Correcto
Prueba 2	Correcto
Prueba 3	Correcto
Prueba 4	Correcto
Prueba 5	Correcto
Prueba 6	Correcto
Prueba 7	Correcto
Prueba 8	Correcto
Prueba 9	Correcto
Prueba 10	Correcto
Prueba 11	Correcto
Prueba 12	Correcto
Prueba 13	Correcto
Prueba 14	Correcto

Capítulo 8:

CONCLUSIONES.

8. Conclusiones.

Este Trabajo Fin de Grado ha consistido en la implementación de un servicio web capaz de realizar recomendaciones musicales, en las que además hemos añadido como contexto el estado de ánimo. También hemos desarrollado una aplicación móvil en Android que nos permitirá consultar esas recomendaciones en cualquier lugar ya que hemos seguido en su desarrollo una arquitectura cliente/servidor.

El servicio web contiene funciones básicas, pero necesarias para lograr los objetivos que nos propusimos para la realización de este TFG. Estas funciones son las de registrar nuevos usuarios, visualizar recomendaciones a través de un algoritmo de filtrado colaborativo; además, podemos añadirle información contextual para que dichas recomendaciones estén filtradas por el estado de ánimo que indiquemos; nos permite valorar canciones para mejorar nuestros modelos de recomendación y también podemos seleccionar las canciones como favoritas para tener un acceso directo a ellas.

Para la realización del sistema de recomendación sensible al contexto de ánimo necesitábamos una amplia base de datos musical que tuvieran información sobre el estado de ánimo al que pertenecían para no tener el problema del arranque en frío y que el sistema generara unas valoraciones válidas. Esta ha sido uno de las mayores dificultades que hemos encontrado en la realización de este TFG, ya que es muy fácil encontrar información sobre canciones, pero es muy difícil que esa información tenga datos sobre el estado de ánimo al que pertenece cada canción.

Se ha conseguido poder obtener un conjunto de casi 5.000 canciones de distintos géneros musicales valoradas por el estado de ánimo, esto tiene el inconveniente de no estar bajo la licencia de distribución libre Creative Commons.

El servicio web se ha implementado cumpliendo con todas las metodologías del proceso de Ingeniería del Software descritas anteriormente y siguiendo los estándares de las tecnologías que hemos utilizado en la realización de este proyecto.

No se ha implementado ninguna interfaz compleja a nivel del servicio web ya que hemos desarrollado una aplicación móvil en Android para poder interactuar de una forma más amena con el servicio web.

En cuanto al desarrollo de la aplicación Android se ha implementado de manera que dé soporte a todas las operaciones que nos permite realizar el servicio web, generando un código modular con orientación a objetos y clases que realizan una sola función para conseguir que el código sea mantenible, fácil de entender y que podamos añadir de manera simple nueva funcionalidad.

Al ser esta aplicación la puerta de entrada del usuario al sistema de recomendación, hemos intentado generar una interfaz que sea intuitiva, atractiva, funcional y que tenga un aspecto familiar a otras aplicaciones utilizadas por los usuarios. Aunque me gustaría remarcar que estamos ante un prototipo y no una aplicación final. En la fase de pruebas se han realizado distintos test de caja negra, obteniendo resultados satisfactorios y validando los requisitos.

Personalmente, esta experiencia ha sido dura pero a la vez muy satisfactoria ya que me ha permitido plasmar en este proyecto muchas de las capacidades y conocimientos que he ido adquiriendo a lo largo de la carrera.

8.1. Áreas de mejora.

Aunque hayamos cumplido con todos los objetivos marcados, esto no significa que no se podrían realizar mejoras y añadir nuevas funcionalidades en las distintas partes de este proyecto. A continuación se detallan algunas propuestas de mejora:

- Incluir nuevos métodos de login a la aplicación móvil, como por ejemplo a través de Facebook, Twitter, Instagram…
- Añadir funciones de reproducción de música a la aplicación Android.
- Implementar la aplicación en el sistema operativo iOS.
- Añadir una funcionalidad para inferir el estado de ánimo del usuario.

Bibliografía.

[1] Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

[2] Kantor, P. B. (2015). *Recommender systems handbook*. F. Ricci, L. Rokach, & B. Shapira (Eds.). Berlin, Germany: Springer.

[3] Yoo, K. H., Gretzel, U., & Zanker, M. (2012). *Persuasive recommender systems: conceptual background and implications*. Springer Science & Business Media.]

[4] Balabanović, M., & Shohom, Y. (1997). Content-based, collaborative recommendation. *Communications of the ACM*, *40*(3).

[5] Schafer, J. B., Konstan, J., & Riedl, J. (1999, November). Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce* (pp. 158-166). ACM.

[6] Adomavicius, G., & Tuzhilin, A. (2005). 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*, *17*(6), 734-749.

[7] Sauter, V. L. (2014). *Decision support systems for business intelligence*. John Wiley & Sons.

[8] Woerndl, W., Schueller, C., & Wojtech, R. (2007, April). A hybrid recommender system for context-aware recommendations of mobile applications. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on* (pp. 871-878). IEEE.

[9] Chen, Y., & Pu, P. (2013). Cofeel: Using emotions to enhance social interaction in group recommender systems. In *Alpine Rendez-Vous (ARV) 2013 Workshop on Tools and Technology for Emotion-Awareness in Computer Mediated Collaboration and Learning*.

[10] Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). Web services. In *Web Services* (pp. 123-149). Springer Berlin Heidelberg.

[12] Richardson, L., & Ruby, S. (2008). *RESTful web services*. "O'Reilly Media, Inc.".

[14] Christensen, J. H. (2009, October). Using RESTful web-services and cloud computing to create next generation mobile applications. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications* (pp. 627-634). ACM.

[15] Adomavicius, G., & Tuzhilin, A. (2011). Context-aware recommender systems. In *Recommender systems handbook* (pp. 217-253). Springer US.

[16] “W3C” - http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb - Fecha de última consulta: 23/05/2017.

[17] Amodeo, E. (2014). Principios de Diseño de API's REST. *España: LeanPub*.

[18] Mulloy, B. (2012). Web API Design-Crafting Interfaces that Developers Love.

[19] Servicio Web - http://es.wikipedia.org/wiki/Servicio_web - Fecha de última consulta: 20/05/2017.

[20] Morales, C. A. (2010). Estado del Arte: Servicios Web.

[21] Arquitectura de la plataforma Android https:/developer.android.com/guide/platform/index.html - Fecha de última consulta: 03/06/2017

[22] Activity en Android - https://developer.android.com/guide/components/activities.html - Fecha de última consulta: 03/06/2017

[23] Servicios en Android - https://developer.android.com/guide/components/services.html- Fecha de última consulta: 03/06/2017

[24] Curso de programación en Android - http://www.sgoliver.net/blog/curso-de-programacion-android/ - Fecha de última consulta: 10/06/2017.

[25] Curso Android http:/www.hermosaprogramacion.com/android/ - Fecha de última consulta: 25/06/2017.

[26] "La historia de Android" - https://www.android.com/history- Fecha de última consulta: 28/08/2017.

[27] Gorton, I. (2006). *Essential software architecture*. Springer Science & Business Media.

[28] Martin, R. C. (2009). *Clean code: a handbook of agile software craftsmanship*. Pearson Education.

[29] Freeman, E., Robson, E., Bates, B., & Sierra, K. (2004). *Head First Design Patterns: A Brain-Friendly Guide*. "O'Reilly Media, Inc.".

[30] Rogers, P. (2005). Ingeniería de Software un Enfoque Práctico. *Editorial McGraw-Hill, Madrid*.

[31] Larman, C. (1999). *UML y Patrones*. Pearson.

[32] Joshua Kerievsky (2004). Refactoring to Patterns. Editorial Addison-Wesley Signature.

[33] Desarrollo iterativo y creciente - https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente - Fecha de última consulta: 14/05/2017.

[34] Korth, H., & Silberschatz, A. (1993). *Fundamentos de bases de datos*. Madrid.

[35] Gironés, J. T. (2012). *El gran libro de Android*. Marcombo.

[36] Shneiderman, B. (2010). *Designing the user interface: strategies for effective human-computer interaction*. Pearson Education India.

[37] "Sociedad de la información" - https://es.wikipedia.org/wiki/Sociedad_de_la_informacion -- Fecha de última consulta: 14/07/2017.

[38] "Musicoterapia" - http://www.lamusicoterapia.com/definiciones-de-musicoterapia/ -- Fecha de última consulta: 14/07/2017

ANEXO I:

MANUAL DE INSTALACIÓN.

ANEXO I – Manual de instalación.

Antes de empezar con la instalación, es necesario revisar el contenido del DVD que acompaña a este TFG, ya que en él se encuentra el código generado para la realización de este proyecto. En éste podemos encontrar:

- PDF de la documentación del proyecto.
- Código de la aplicación Android (Recommusic).
- Código del servicio web.
- Código para la creación de máquinas virtuales con Vagrant.

En este anexo explicaremos la instalación de la parte del servidor y de la aplicación cliente en Android.

Servidor

Para facilitar la instalación de la parte del servidor, hemos decidido utilizar Vagrant que es una herramienta gratuita de línea de comandos que permite generar entornos de desarrollo reproducibles y compartibles de forma muy sencilla. Para ello, Vagrant crea y configura máquinas virtuales a partir de simples ficheros de configuración.

En los ficheros de configuración de Vagrant se le indica desde qué sistema operativo se instalará, los requisitos hardware que tendrá, la configuración de red que necesitará y, además, podremos indicar la instalación de software adicional.

Para poder utilizar Vagrant solamente necesitamos instalar dos aplicaciones software. Uno es el software de virtualización que queremos: por defecto Vagrant utiliza VirtualBox (aunque se podría instalar otro); y el otro software necesario es el propio de Vagrant, que a continuación se pasará a explicar.

Este manual está basado en la instalación en el sistema operativo Ubuntu, aunque cualquiera de los programas se puede encontrar para las plataformas de Mac y Windows.

1. Accedemos a la siguiente web https://www.virtualbox.org/wiki/Downloads que corresponde con la página de VirtualBox para instalar la última versión de la aplicación.
2. Seleccionamos el enlace de descarga correspondiente a nuestra versión, en nuestro caso es Ubuntu 16.04 – AMD de 64 bits, y el archivo se descargará en nuestro equipo.

Ilustración I.1 Página de descarga de VirtualBox.

3. Abrimos una terminal de consola y accedemos al directorio donde se descargó la aplicación.
4. Ahora toca ejecutar el archivo descargado con la instrucción sudo dpkg –i más el nombre del fichero, en nuestro caso ha sido el siguiente:

```
sudo dpkg -i virtualbox-5.1_5.1.6-110634~Ubuntu~xenial_amd64.deb
```

5. En unos pocos minutos la instalación habrá terminado y podremos abrir la aplicación.

Ilustración I.2 Pantalla de inicio de VirtualBox.

Ahora pasaremos a instalar Vagrant en nuestro equipo, para ello realizaremos el mismo proceso que para VirtualBox.

6. Accedemos a la página de descarga de Vagrant: https://www.vagrantup.com/downloads.html.
7. Seleccionamos la versión más reciente para nuestro sistema operativo, en nuestro caso Debian – 64 bits y descargamos el archivo.

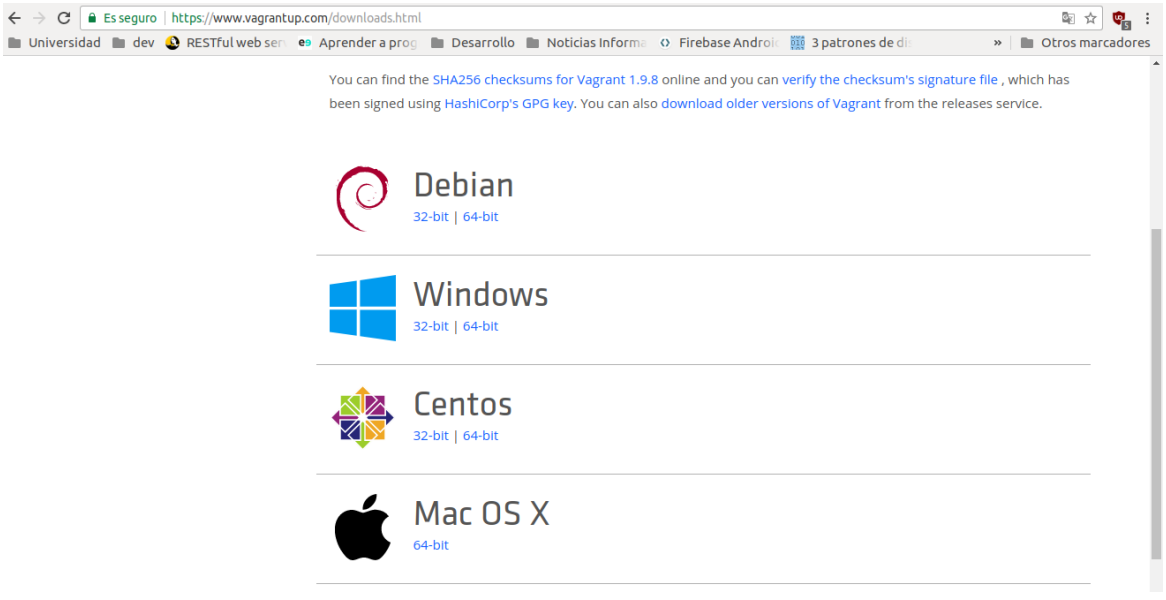

Ilustración I.3 Página de descarga de Vagrant.

8. Desde la terminal volvemos acceder al directorio de descarga.
9. Ejecutamos la instrucción de instalación sudo dpkg –i más el nombre del fichero:

```
sudo dpkg -i vagrant_1.6.5_x86_64.deb
```

10. En unos instantes ya tendremos Vagrant instalado y, si queremos verificarlo, podemos ejecutar el siguiente comando:

```
vagrant –v
```

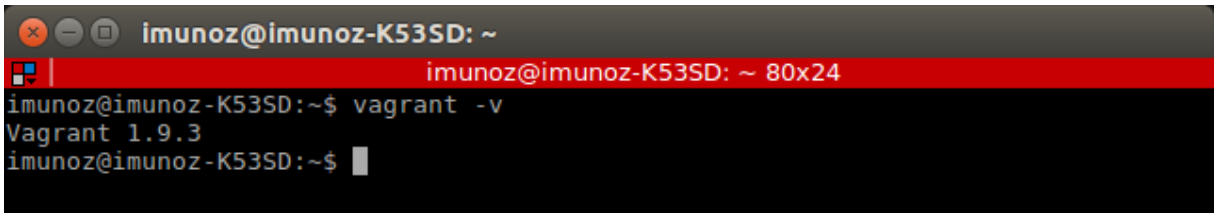

Ilustración I.4 Comprobación de instalación.

Ya tenemos instalado todo el software externo necesario.

11. A continuación, debemos copiar en nuestro equipo la carpeta del DVD que contiene los archivos de configuración para la virtualización con Vagrant.
12. Desde una terminar accedemos a la carpeta que hemos copiado anteriormente y ejecutamos el comando para levantar la máquina virtual:

```
vagrant up
```

13. Al ser la primera vez que lo ejecutamos, leerá los archivos de configuración (Vagrantfile y provision.sh) para preparar todo el entorno:

Instalará el siguiente software:

- Ubuntu 14.04 64-bit
- Apache 2.4
- PHP 5.5
- MySQL 5.5
- XDebug
- PHPUnit 4.8
- Composer
- Java 8

Además realizará la importación de la BBDD, añadirá el servicio web a los directorios de Apache y dará permisos de ejecución al servicio de recomendación.

14. Por último, ya tendremos nuestra máquina virtual disponible y lista para su funcionamiento.

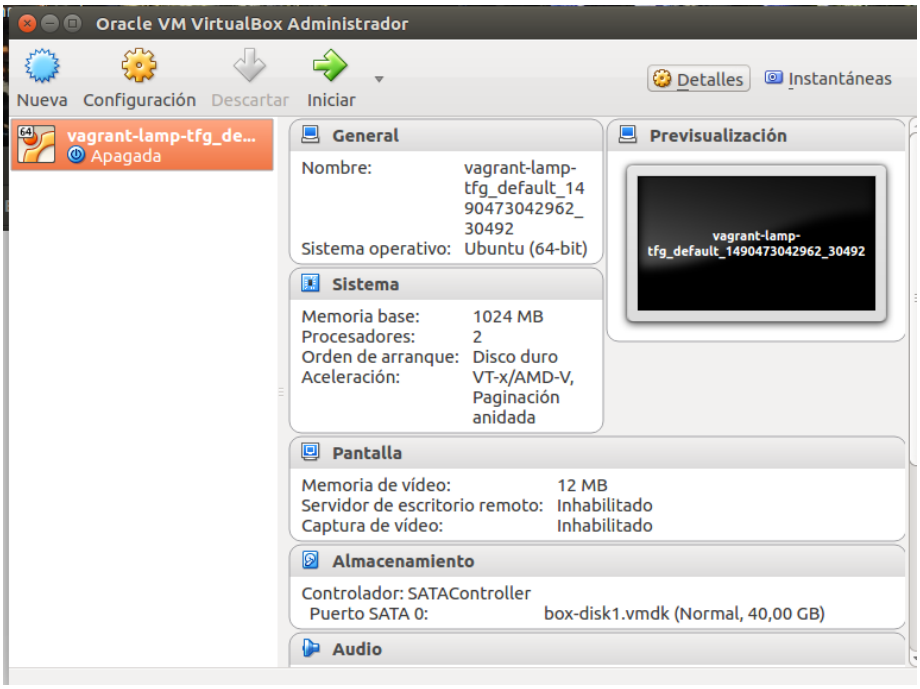

Ilustración I.5 VirtualBox con entorno de Vagrant

Cliente

Ahora pasamos a describir el proceso para la instalación de la aplicación en un móvil Android. Detallaremos cómo hacer la instalación de manera manual ya que nuestra aplicación no se encuentra subida a los servidores de Google Play Store.

1. Debemos configurar nuestro móvil para que permita instalar una aplicación que no se encuentre en Google Play. Para ello debemos de ir a “Ajustes -> Seguridad en nuestro móvil” y activar la opción “Orígenes desconocidos”.

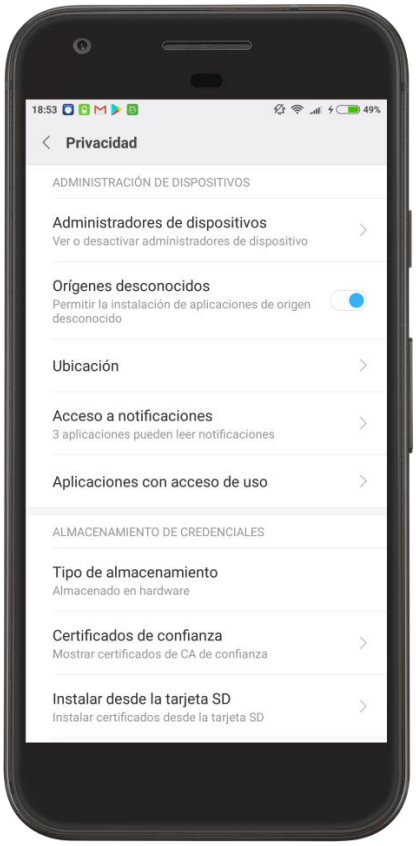

Ilustración I.6 Pantalla de seguridad.

2. Transferimos el archivo .APK que se encuentra en el DVD al dispositivo móvil. Esto podremos realizarlo por cualquier medio.
3. Desde el dispositivo seleccionamos el archivo .APK para su instalación.

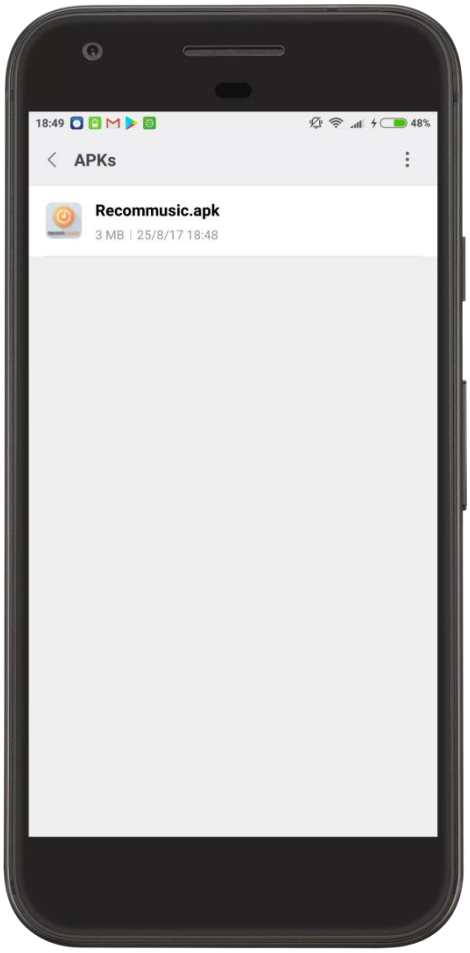

Ilustración I.7 Archivo de instalación de la aplicación.

4. En la pantalla nos indicará si estamos de acuerdo con la instalación y, si es necesario, nos indicará qué permisos tenemos que aceptar.

Ilustración I.8 Pantalla de confirmación de instalación.

5. Al pulsar el botón de “Instalar”, se procederá a su instalación y en unos segundos podremos acceder a la aplicación.

Ilustración I.9 Pantalla de finalización de la instalación.

ANEXO II:

MANUAL DE USUARIO.

ANEXO II - Manual de usuario de la aplicación móvil.

Registro o inicio de sesión.

Una vez instalada la aplicación, debemos de localizar en el menú de nuestro dispositivo Android el icono de la aplicación “Recommusic” y pulsar sobre él para abrirla y comenzar a utilizarla.

En primer lugar, la aplicación nos mostrará la pantalla de bienvenida para iniciar sesión, tal y como se muestra en la ilustración II.1. Al pulsar el botón “Iniciar sesión”, aparecerán las cuentas de Gmail vinculadas con el dispositivo (ilustración II.2). Indicaremos la cuenta con la que vamos a acceder a la aplicación y, seguidamente, nos pedirá la confirmación de si queremos ceder nuestros datos. Tras la correspondiente confirmación, habremos iniciado sesión o, si es nuestra primera vez, nos habremos registrado correctamente.

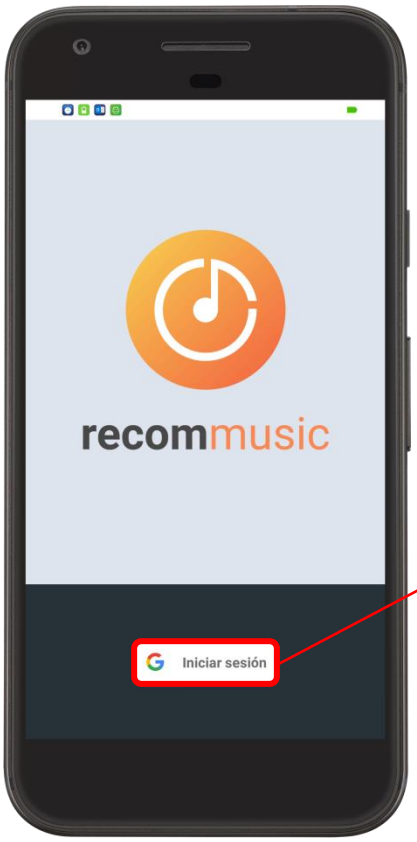

Ilustración II.1 Pantalla de inicio de la aplicación.

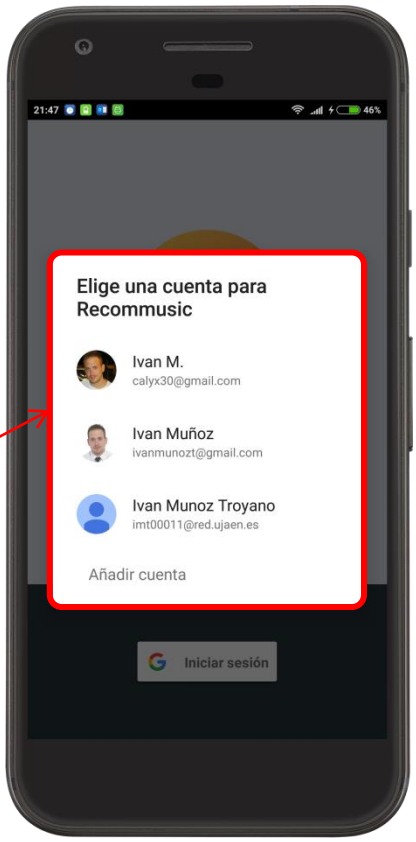

Ilustración II.2 Cuentas vinculadas.

Solicitar lista de canciones recomendadas según el contexto de ánimo.

Para acceder a la lista de recomendaciones, una vez iniciada la sesión en la aplicación, aparecerá el menú con las distintas opciones (ilustración II.3). También podremos acceder a este menú desde un menú lateral.

En el menú de inicio veremos las distintas opciones que aparecen, en este caso, pulsaremos sobre el botón de “Recomendación con contexto de ánimo”, el cual nos mostrará una pantalla con los diferentes grados de estados de ánimo (ilustración II.4). En esta nueva pantalla aparece una cuadrícula dividida en nueve partes que se corresponden con los nueve estados de ánimo implementados: *negativo y enérgico*, *enérgico*, *positivo y enérgico*, *negativo*, *neutral*, *positivo*, *negativo y calmado*, *calmado*, y finalmente, *positivo y calmado*.

A continuación, seleccionaremos el estado de ánimo deseado y se mostrará un listado de canciones con las recomendaciones referidas a dicho estado.

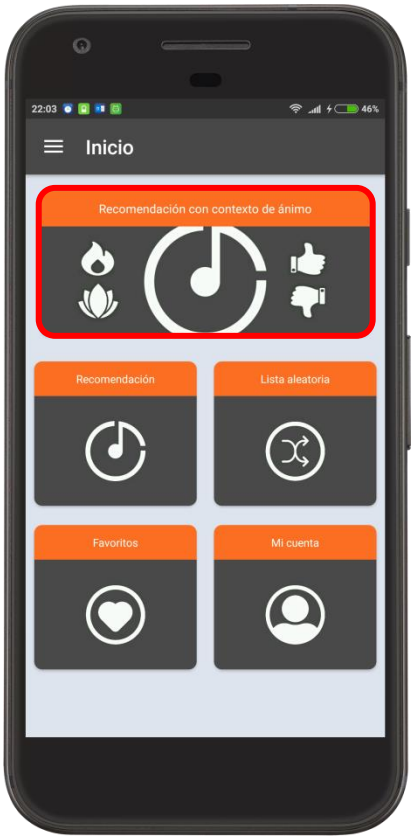

Ilustración II.3 Menú de inicio.

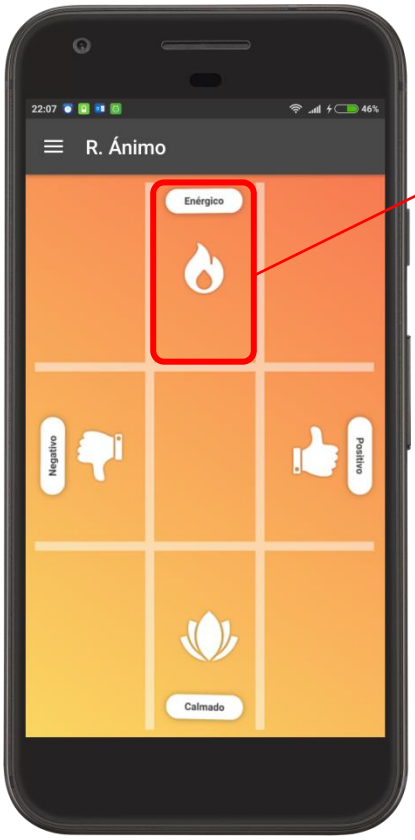

Ilustración II.4 Recomendación con contexto de ánimo.

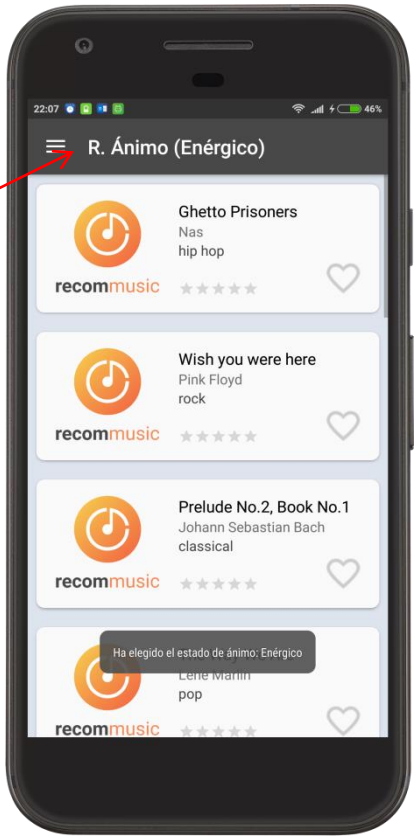

Ilustración II.5 Recomendación según estado de ánimo.

Solicitar lista de canciones recomendadas.

Esta acción se realiza de forma muy parecida a la anterior, lo único en lo que difiere es que esta recomendación no es sensible al estado de ánimo.

Para acceder a la lista de canciones recomendadas necesitamos pulsar el botón “Recomendación” desde el menú principal (ilustración II.3). A continuación, se mostrará la lista de recomendaciones ajustadas al perfil del usuario.

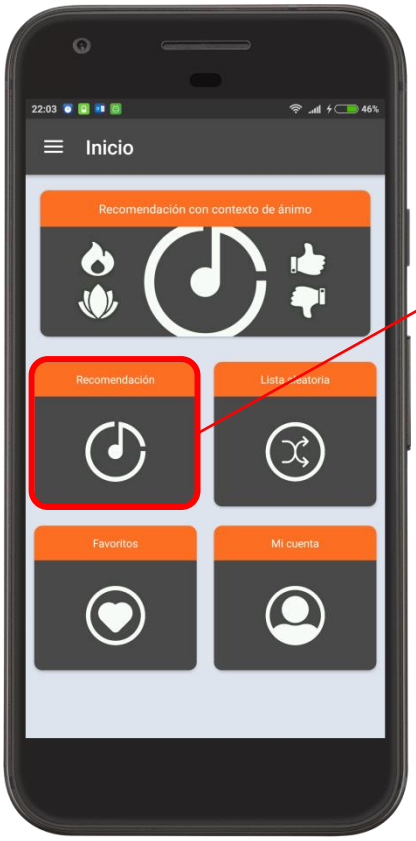

Ilustración II.3 Menú de inicio.

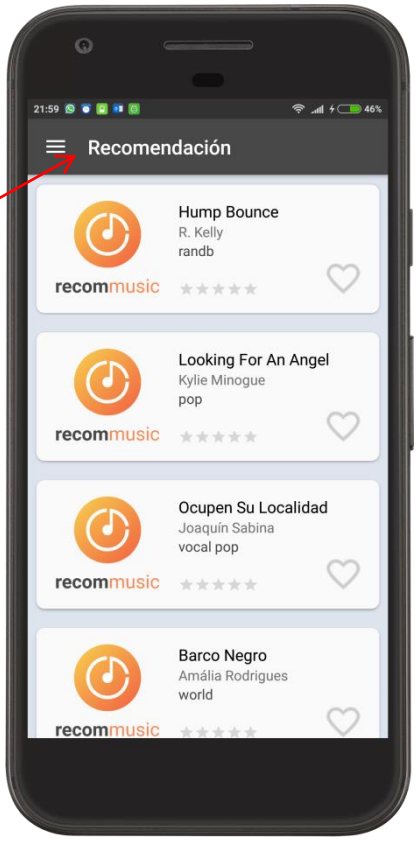

Ilustración II.6 Listado de canciones recomendadas.

Solicitar lista de canciones aleatorias.

Desde la pantalla del menú, pulsaremos ahora la opción de “Lista aleatoria” (ilustración II.3). En unos segundos se mostrará un listado aleatorio de canciones sin ningún criterio de recomendación.

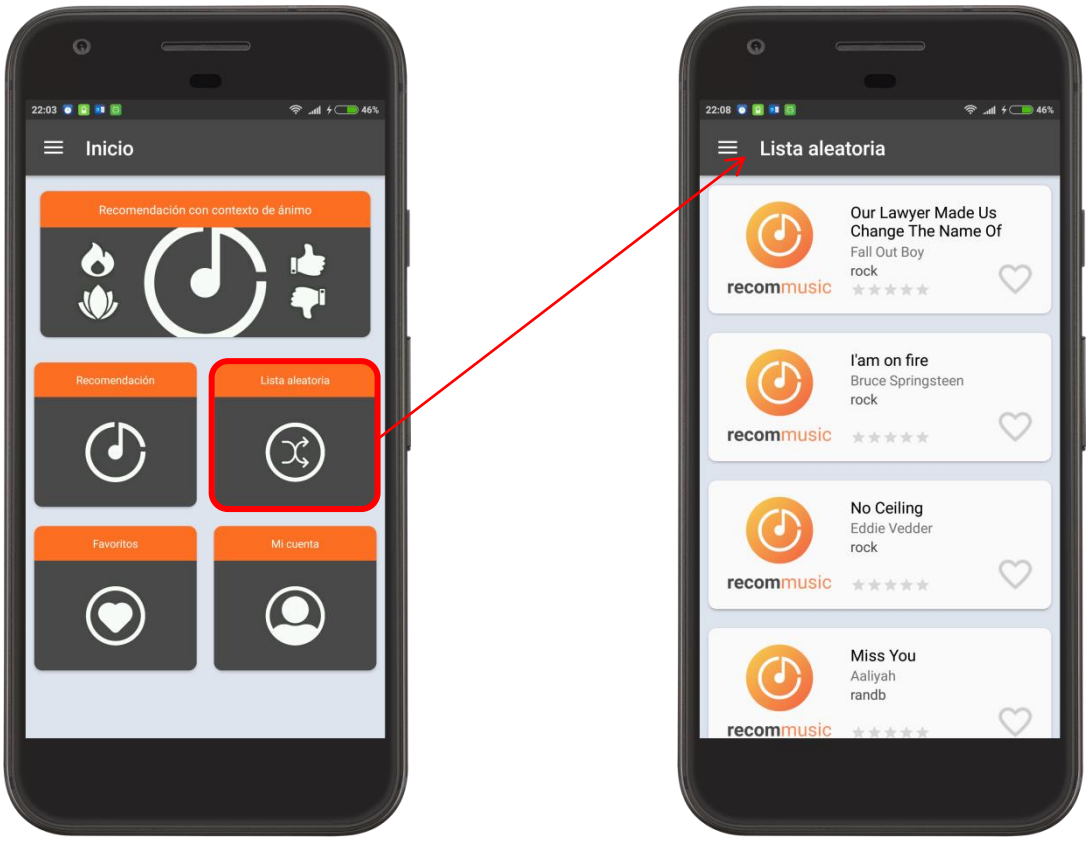

Ilustración II.3 Menú de inicio.

Ilustración II.7 Listado aleatorio de canciones.

Ver lista de favoritos.

Desde el menú de inicio, elegiremos pulsar sobre el botón de “Favoritos” (ilustración II.3). En una nueva pantalla se mostrará el listado de canciones que previamente se hayan marcado como favoritas. En el caso de que no se haya señalado ninguna canción como favorita, este listado aparecerá vacío.

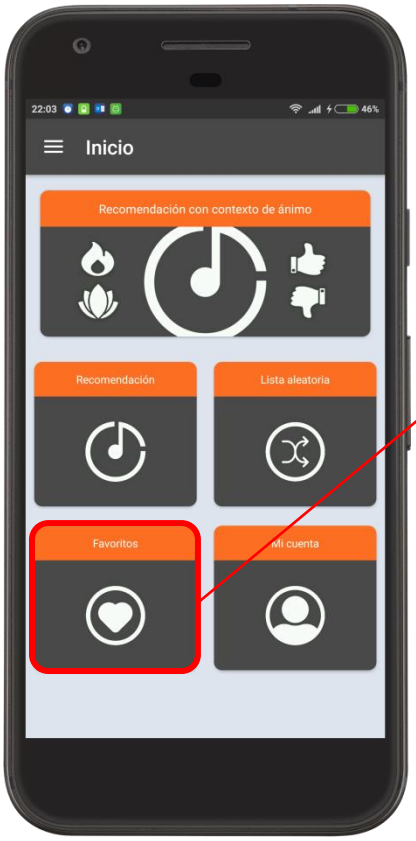

Ilustración II.3 Menú de inicio.

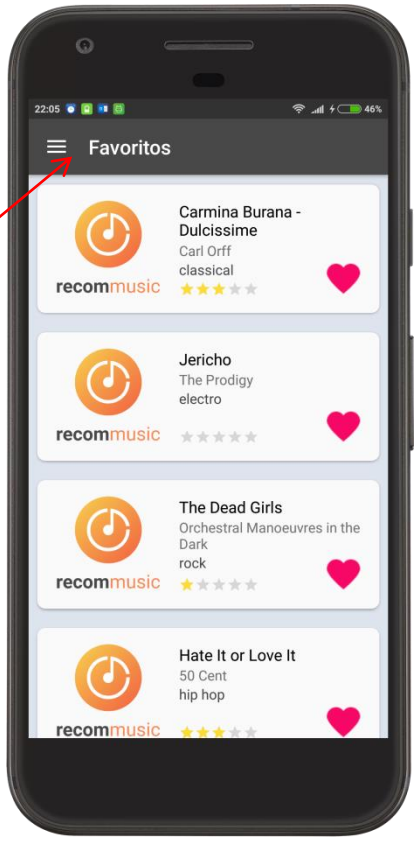

Ilustración II.8 Listado de canciones favoritas.

Valorar una canción.

En cualquiera de las pantallas donde se muestran las canciones, podremos realizar una valoración sobre las mismas. Cada canción puede ser valorada con un máximo de cinco estrellas que hace referencia a una escala del 1 al 5. Si se marcan dos estrellas se le estará danto 2 puntos a la canción, si se marcan cuatro estrellas se le dará 4 puntos y así sucesivamente.

Para puntuar una canción, pulsaremos directamente sobre la estrella deseada, por ejemplo, si pulsamos sobre la tercera estrella se pintarán de amarillo tres estrellas, con lo que daremos 3 puntos a la canción.

También podremos modificar o eliminar una valoración dada. Si pulsamos sobre una estrella de una canción que ya está valorada, esta puntuación será modificada.

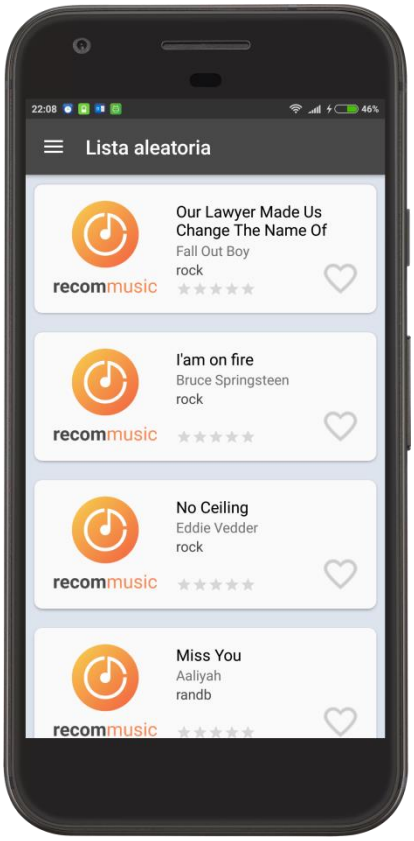

Ilustración II.9 Lista de canciones.

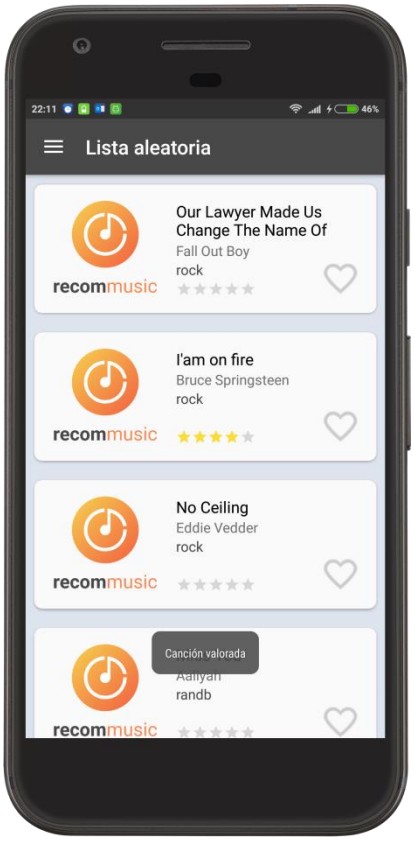

Ilustración II.10 Añadir una valoración.

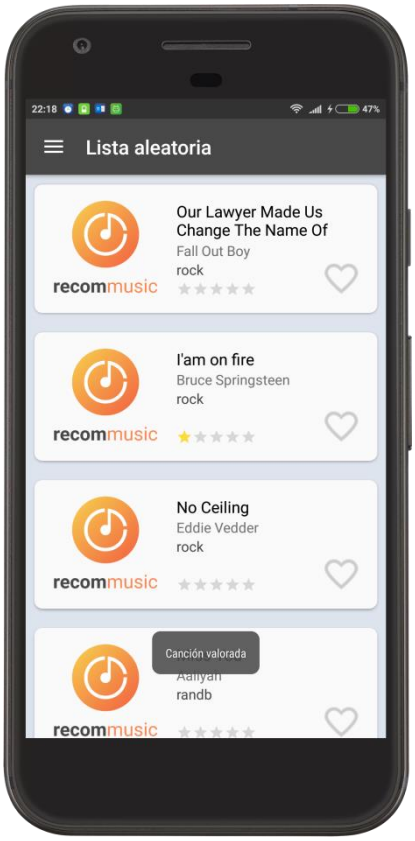

Ilustración II.11 Modificación de una valoración.

Añadir a favoritos.

Una vez que nos encontremos en alguna de las pantallas que nos muestran las listas de canciones, podremos marcar cualquier canción como favorita. Cada canción se muestra junto con un icono en forma de corazón.

Si por primera vez pulsamos sobre el corazón de una determinada canción, éste cambiará de color y se añadirá a la lista de favoritos (ver ilustración II.8). En caso de que volvamos a marcar el corazón, éste se desmarcará y la canción abandonará la lista de favoritos, a la cual se accede como ya hemos comentado anteriormente.

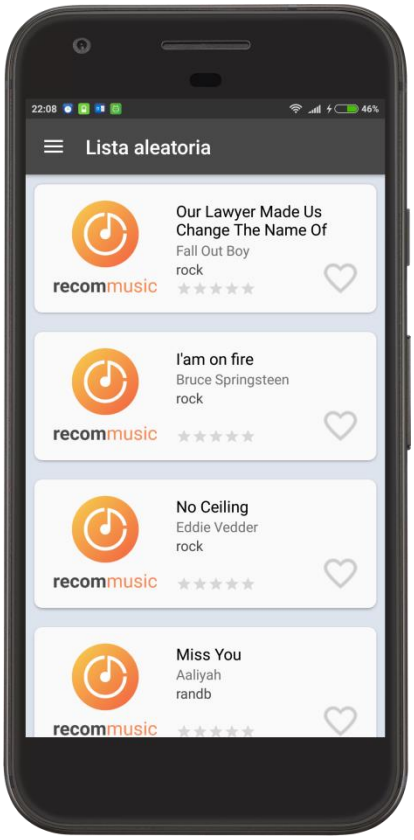

Ilustración II.12 Lista de canciones.

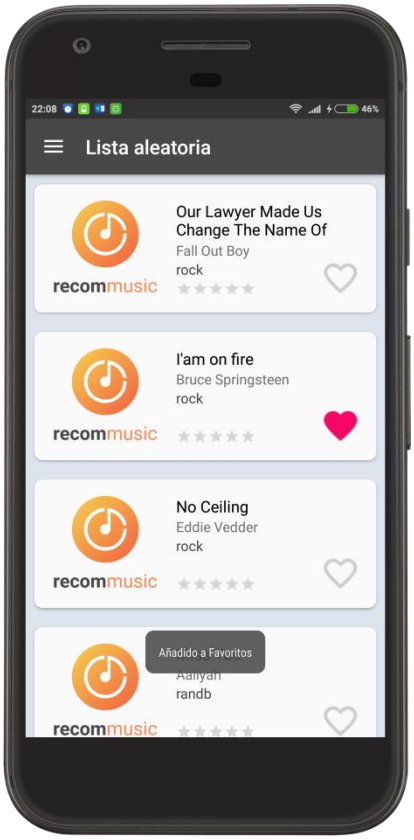

Ilustración II.13 Añadir una canción como favorita.

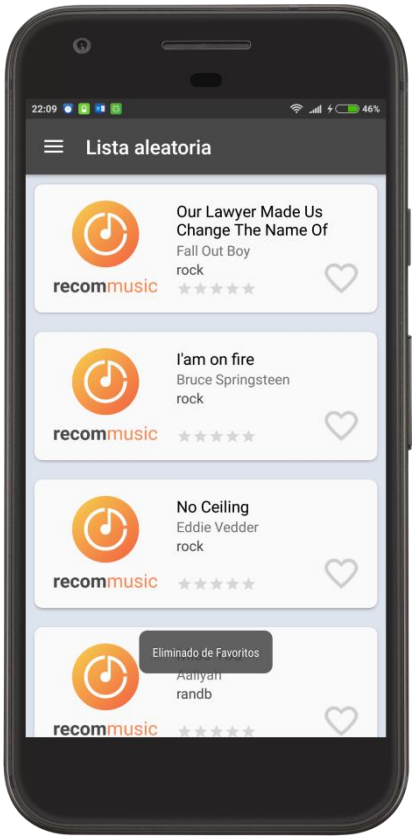

Ilustración II.14 Eliminar una canción de "Favoritos".

Cerrar sesión.

Si queremos cerrar sesión en nuestra aplicación, deberemos ir a la pantalla de "Mi cuenta" (ilustración II.16). A ésta se podrá acceder desde el menú lateral o desde el menú de inicio.

Una vez en la pantalla de "Mi cuenta", al final de la información del usuario pulsaremos sobre el botón "Cerrar sesión", a partir del cual nos redirigirá a la pantalla de inicio de sesión (ver ilustración II.1).

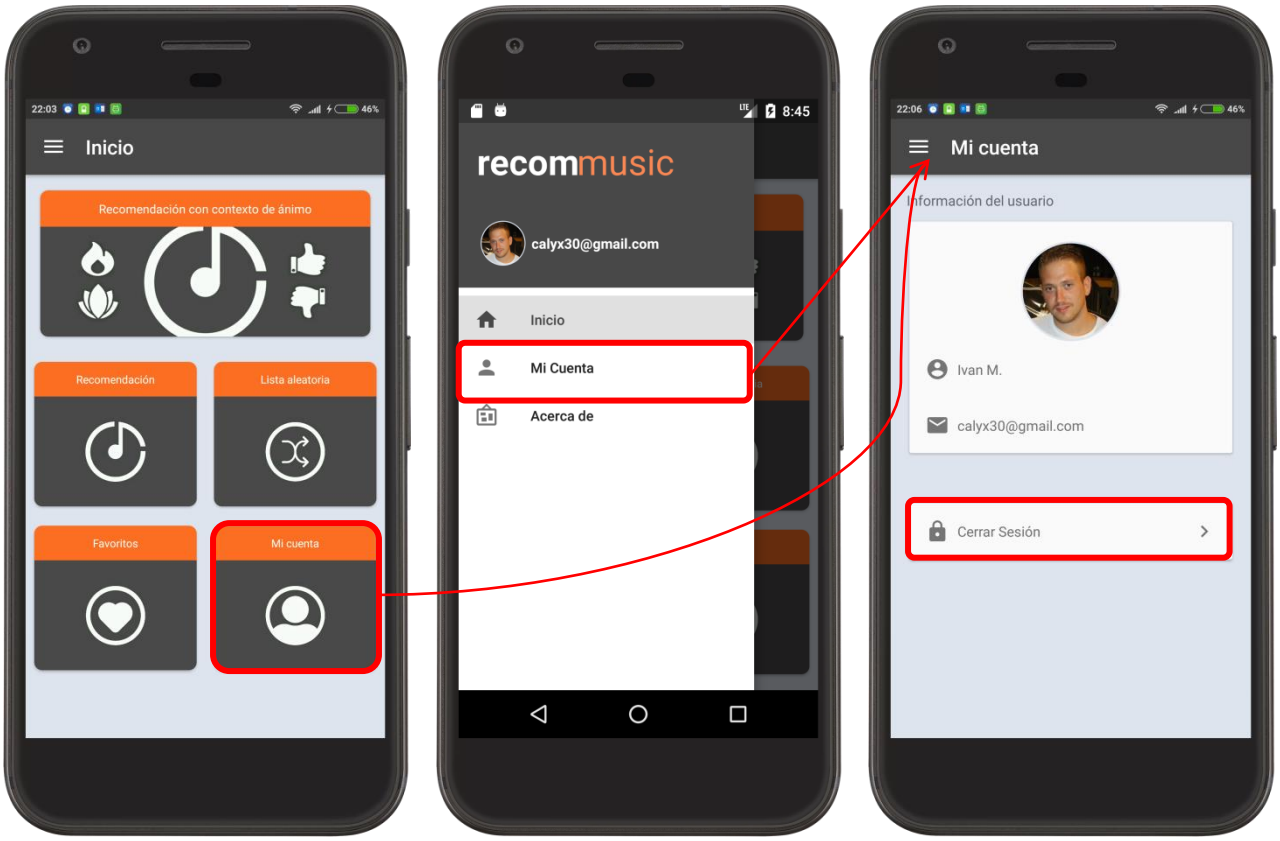

Ilustración II.3 Menú de inicio. **Ilustración II.15 Menú lateral.** **Ilustración II.16 Pantalla "Mi cuenta".**