Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

SISTEMAS DE RECOMENDACIÓN PARA LA HOSTELERÍA CON CONTEXTO DE LOCALIZACIÓN: APP ANDROID SOBRE SERVICIO WEB ESTABLECIDO

Alumno: Mario Quesada Moreno

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

Agosto, 2016

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don Luis Martínez López, tutor del Trabajo Fin de Grado titulado: Sistema de Recomendación para la hostelería con contexto de localización: App Android sobre servicio Web establecido, que presenta Mario Quesada Moreno, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, julio de 2016

El alumno:

QUESADA MORENO, MARIO (FIRMA)

Firmado digitalmente por QUESADA MORENO, MARIO (FIRMA)
Fecha: 2016.08.31 11:07:15 +02'00'

Mario Quesada Moreno

Los tutores:

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: 2016.08.31 11:22:57 +02'00'

Luis Martínez López

Contenido

Índice de Imágenes

Índice de Tablas.

1. INTRODUCCIÓN.

En este capítulo se va a realizar una introducción al Trabajo Fin de Grado (TFG), realizando una breve introducción a los temas relacionados con su desarrollo como son los Sistemas de Recomendación (SR) y Android. Posteriormente se presentarán el propósito de este TFG y sus objetivos. A continuación, le seguirá la estructura del TFG y el capítulo finalizará con una planificación temporal orientativa.

1.1. Introducción al Trabajo Fin de Grado

La gran cantidad de información accesible a través de la Web ha hecho que sea necesario en muchos servicios, sobre todo en los relacionados con el comercio electrónico, ayudar a los usuarios a encontrar de forma fácil los productos que necesitan. Esto es el fundamento de los sistemas de recomendación.

Dichos sistemas se basan en la similitud entre usuarios y permiten a los sitios webs, como Amazon, Youtube, Pandora, Netflix, etcétera, ofrecerle al usuario una navegación más personalizada y ágil.

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

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

Existen varios tipos de Sistemas de Recomendación, Handbook[28]: Basados en Contenido, Colaborativos, Basados en Conocimiento, Comunidad, Demográficos, así como hibridaciones de los anteriores tipos.

En sus inicios a los SR se accedía mediante navegadores en un PC. Sin embargo la movilidad y necesidad de ubicuidad de estos sistemas ha generado gran cantidad de software para dispositivos móviles. Algunos de estos dispositivos móviles, utilizan Android como sistema operativo, basado en Linux [4], diseñado principalmente para dispositivos con pantalla táctil como teléfonos inteligentes o tabletas. Inicialmente desarrollados por Android, Inc., que Google respaldó económicamente y más tarde compró en 2005. Android fue presentado en 2007

junto la fundación del Open Handset Alliance [5]: un consorcio de compañías de hardware, software y telecomunicaciones para avanzar en los estándares abiertos de los dispositivos móviles.

En el actual Trabajo Fin de Grado (TFG), se pretende, realizar una aplicación de Android para un sistema de recomendación de filtrado colaborativo, que se complementará con una serie de servicios REST para que una aplicación Android pueda ofertar recomendaciones que tengan en cuenta distintos contextos que pueden influir en la recomendación que reciba el usuario de la aplicación. La implementación y despliegue de SR a través de la web puede ser facilitado mediante el uso de servicios web.

Un servicio web es un programa software alojado en un servidor, que provee de una serie de funcionalidades mediante una interfaz. También se puede definir como [9]:

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

En capítulos posteriores se profundizará sobre los diferentes tipos de servicios que hay, así como sus ventajas y desventajas.

1.2. Propósito.

Desarrollo de un prototipo de aplicación móvil Android para la integración del contexto del usuario en un sistema de colaboración basado en filtrado colaborativo, para generar recomendaciones más personalizadas a los gustos del usuario.

1.3. Objetivos.

- Revisión bibliográfica.
- Análisis, diseño e implementación de una aplicación en Android.
- Conexión de la aplicación de Android con el sistema de recomendación a través de un servicio web.
- Implementación de la recomendación contextualizada por la ubicación del usuario en el sistema de recomendación y Google Maps.

1.4. Estructura del TFG.

A continuación, se hará una breve introducción de los capítulos que se van a desarrollar a lo largo del proyecto y los contenidos expuestos en él.

Como hemos visto el primer capítulo es una introducción general al proyecto, con una justificación de su realización, el propósito que persigue y los objetivos a conseguir.

El capítulo 2 se dedicará al estudio de sistema operativo Android, y la justificación de porqué se ha elegido determinada versión de la API de Android para el TFG, también se abordará una visión general de los servicios web, así como sus ventajas, inconvenientes, y diferentes tipos de servicios como pueden ser SOAP o REST.

El capítulo 3 se centra en los sistemas de recomendación dando una visión general de los mismos, para posteriormente profundizar en su funcionamiento, ventajas e inconvenientes, y terminar con una justificación de su utilización y ejemplos de sistemas de recomendación reales.

El capítulo 4 contiene el grueso principal del proyecto, ya que abarca todo el proceso de ingeniería del software para su desarrollo. Como proyecto software que es, se hará en primer lugar un breve repaso a las diferentes etapas de la Ingeniería de Software, para acto seguido aplicarlas en el desarrollo de nuestra aplicación. Así, se definirán los requerimientos funcionales y no funcionales para la aplicación, abordando la etapa de análisis, incluyendo el modelo de casos de uso. Acto seguido, se pasa a la etapa de diseño, haciendo especial hincapié en el diseño de la interfaz de la aplicación Android y las pruebas de usabilidad llevadas a cabo. Por último, repasaremos el proceso seguido para la etapa de implementación del servicio REST y el acceso a él desde Android.

La memoria concluye con el capítulo 5 en el que se expondrán las conclusiones obtenidas tras la realización de dicho proyecto.

1.5. Planificación Temporal.

A continuación se añade una planificación temporal del TFG.

Esta planificación temporal es orientativa ya que, se están definiendo las fechas de comienzo y de fin de las actividades, sin apriori conocer la complejidad de desarrollo del Sistema.

En la tabla 1 podemos ver la duración Estimada de las Actividades más relevantes del TFG. A continuación se enumera el significado de cada columna:

- *Id*: El identificador secuencial de la actividad.
- *Nombre*: Nombre proporcionado a la tarea.
- *Duración*: El número de días que comprenderá la realización de la tarea.
- *Comienzo*: Fecha de comienzo de la actividad.
- *Fin*: Fecha de fin de actividad
- *Predecesoras*: Actividades que es necesario haber realizado para desarrollar la actividad.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
0	**temporal estimada**	**179 días**	**jue 18/09/15**	**lun 25/05/16**	
1	Revisión Bibliográfica	40 días	**jue 18/09/15**	**mié 12/11/15**	
2	Análisis, diseño e implementación de la App	73 días	**mié 12/11/15**	**vie 19/02/16**	
3	Conexión App Android con Sistema Recomendación	39 días	**lun 23/02/16**	**jue 16/04/16**	2
4	Recomendación Contextualiza	27 días	**vie 17/04/16**	**lun 25/05/16**	3

Tabla 1.1 Duración Actividades

A continuación se puede ver, en la figura 1, el Diagrama de GANT de la planificación de las actividades.

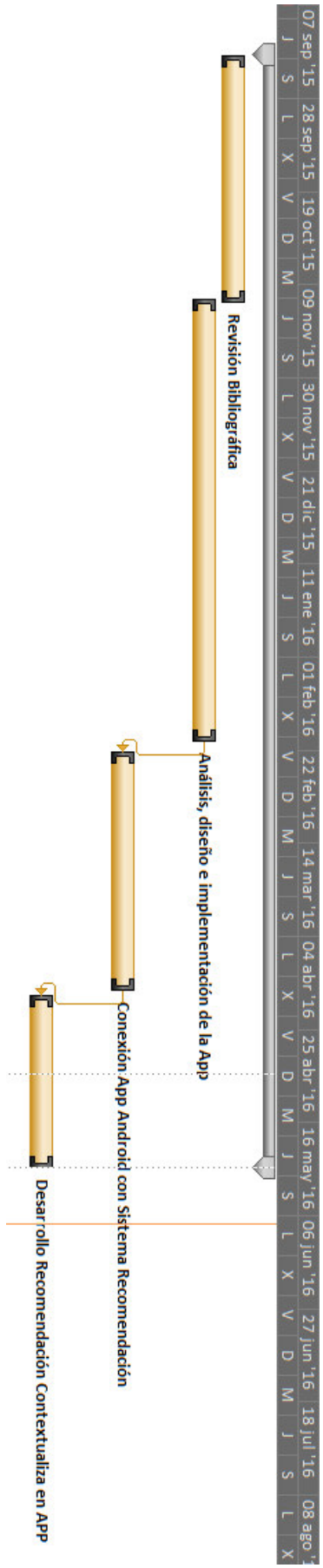


Figura 1.1 Diagrama de GANT

CAPÍTULO 2 Android

Este capítulo comienza con una introducción al capítulo, y a continuación se analizará en profundidad el sistema operativo Android, incluyendo una descripción de las características más destacadas que hemos de conocer acerca del mismo para entender el desarrollo de la aplicación que se realiza en este TFG.

Posteriormente, el capítulo finalizará con una revisión de los servicios web, analizando sus ventajas e inconvenientes.

2.1. Introducción.

Hoy en día vivimos en la sociedad de la información, tenemos la insaciable necesidad de estar informados de todo lo que sucede en el mundo en cualquier momento. Para acceder a esta información utilizamos dispositivos móviles.

En el mercado actualmente existen gran variedad de dispositivos móviles y sistemas operativos. Sin embargo, nosotros para la realización de este TFG hemos elegido Android; dado su gran cuota de mercado como se puede ver, en la figura 2 [6] o en la tabla 2 [6], a continuación.

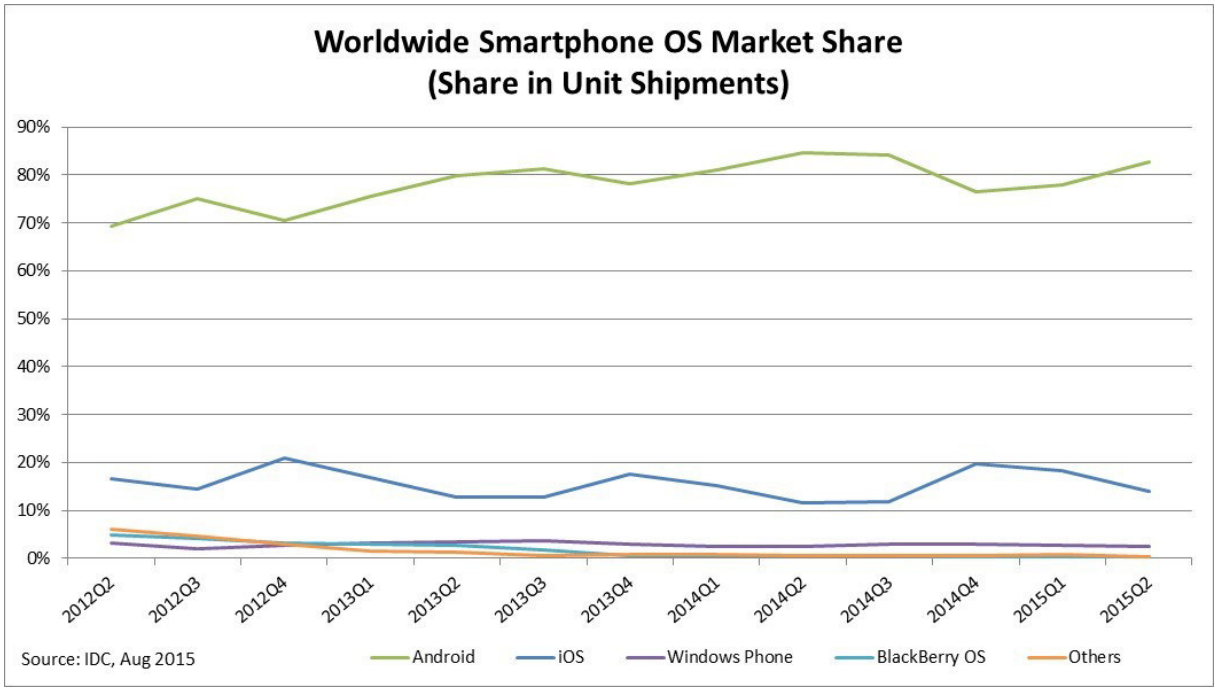


Figura 2.1 Comparativa cuota mercado por IDC

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Tabla 2.1 Comparativa cuota mercado, por IDC

Tras observar la figura y la tabla anteriores, vemos que Android a final del 2015 tiene un 82.6% de cuota de mercado a nivel mundial, esto ha sido un aliciente más para escoger este Sistema Operativo para dispositivos móviles como entorno para el desarrollo de este TFG.

2.2. Android.

Android es un sistema operativo desarrollado por Android inc, empresa respaldada por Google que finalmente terminó adquiriendo en el año 2005. En el año 2007 se creó el consorcio Handset Alliance1 con el objetivo de desarrollar estándares abiertos para móviles. Está formado por Google, Intel, Texas Instruments, Motorola, T-Mobile, Samsung, Ericsson, Toshiba, Vodafone, NTT DoCoMo, Sprint Nextel y otros [7].

Ese mismo año, se presenta el primer dispositivo móvil con Android, el T-Mobile G1. Pero es en el año 2008 cuando sale al mercado el primer móvil con Android, el HTC Dream. Desde ese primer lanzamiento, Android ha experimentado un gran crecimiento, siendo el sistema operativo móvil líder en cuanto a cuota de mercado se refiere, en la figura 2.2 podemos ver una comparativa con los datos del primer Cuatrimestre de 2015:

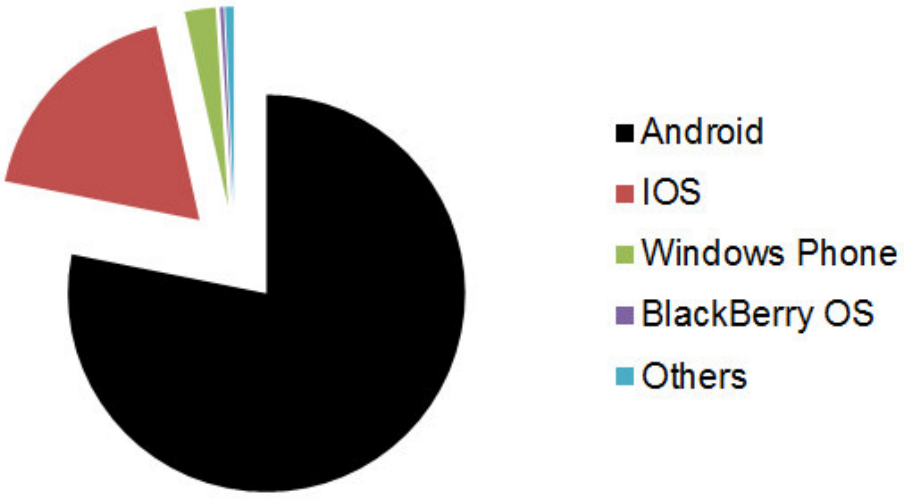


Figura 2.2 Cuota de mercado Sistemas Operativo 2015

La consolidación de Android como la plataforma líder para dispositivos móviles y la relativa facilidad a la hora desarrollar aplicaciones en cuanto a información disponible desde webs oficiales para desarrolladores han hecho que Android sea elegida como la plataforma idónea para la realización de este trabajo.

A continuación, se estudiará la evolución de Android, la arquitectura que tiene este sistema operativo y de qué componentes está estructurado, el ciclo de vida de los elementos principales en Android. Por último, veremos el nivel de seguridad que dispone el sistema operativo, qué tipos de almacenamiento utiliza y cuál es la estructura de un proyecto en Android.

2.2.1. Evolución de la API.

Desde su lanzamiento en 2007 Android ha lanzado un total de 22 versiones de la API del sistema operativo. Cada vez que lanza una nueva versión de la API, se añaden características nuevas y se mejoran muchas características de versiones anteriores. En la tabla 2.2 podemos observar la evolución, a lo largo del tiempo de las versiones, de la plataforma Android [7].

Versión	Nombre	Logo	API
Android 1.0	Apple Pie		1
Android 1.1	Banana Bread		2
Android 1.5	Cupcake		3
Android 1.6	Donut		4
Android 2.0	Éclair		5
Android 2.1	Éclair		7
Android 2.2	Froyo		8
Android 2.3	Gingerbread		9
Android 3.0	HoneyComb		11
Android 3.1	HoneyComb		12
Android 3.2	HoneyComb		13
Android 4.0	Ice Cream Sandwich		14
Android 4.0.3	Ice Cream Sandwich		15
Android 4.1	Jelly Bean		16
Android 4.2	Jelly Bean		17
Android 4.3	Jelly Bean		18
Android 4.4	Kitkat		19
Android 5.0	Lollipop		21

Android 5.1	Lollipop		22

Tabla 2.2 Versiones de Android.

En cuanto a la tasa de distribución, como podemos ver en la figura 2.3 y la tabla 2.3, Jelly Bean y Kitkat son las más utilizadas. Esto nos es de gran utilidad a la hora de desarrollar una aplicación, ya que a la hora de realizar una aplicación tenemos que definir la versión mínima que nuestra aplicación puede soportar: esto quiere decir que la aplicación no puede ser instalada en versiones inferiores a la versión mínima escogida.

En nuestro caso para la realización del TFG, se ha decido utilizar la API 16 como versión mínima, ya que existen ciertas características que fueron implementadas a partir de esta API que nos facilitan la tarea y abarca prácticamente el 89% del mercado en la actualidad con esta limitación.

Versión	Nombre	API	Distribución
2.2	Froyo	8	0.3%
2.3.3 - 2.3.7	Gingerbread	10	5.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.1%
4.1.x	Jelly Bean	16	14.7%
4.2.x		17	17.5%
4.3		18	5.2%
4.4	KitKat	19	39.2%
5.0	Lollipop	21	11.6%
5.1		22	0.8%

Tabla 2.3 Tasa de Distribución para las versiones de Android

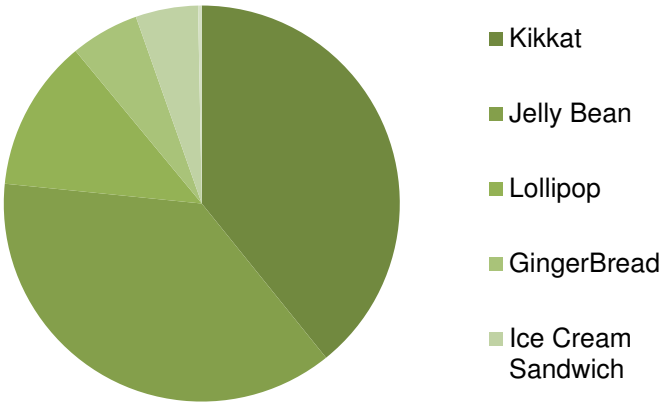


Figura 2.3 Tasa de Distribución para las versiones de Android

2.2.2. Arquitectura

Como se aprecia en la figura 2.4, la arquitectura de Android está dividida en cuatro capas [7]:

- *Núcleo*: Está formado por el sistema operativo Linux 2.6. Provee a la capa de aplicación de una interfaz para interactuar con el hardware, y ofrece servicios como la seguridad, el multiproceso, el manejo de memoria, la pila de protocolos, y el soporte de drivers para dispositivos. Es código abierto bajo licencia Apache 2.0 y GPL v2 [10].
- *Runtime*: Engloba la máquina virtual Dalvik basada en la máquina virtual de java pero optimizada para trabajar de manera eficiente con poca memoria y un procesador limitado. También incluye el "Core Libreries" que son la mayoría de librerías disponibles en el lenguaje Java.
- *Librerías nativas*: Son un conjunto de librerías en C/C++ compiladas en el código nativo del procesador. Estas librerías proporcionan acceso a las funcionalidades del sistema a través de la capa que explicaremos a continuación.
- *Entorno de aplicación*: Esta capa ha sido diseñada pensando en la reutilización de componentes de manera sencilla, por lo que facilita bastante el desarrollo de aplicación
- *Aplicaciones*: Engloba a todo el conjunto de aplicaciones instaladas en el dispositivo Android, ya sean aplicaciones instaladas en el dispositivo por defecto como un cliente de correo, o un calendario, como las

aplicaciones disponibles en el Play Store, desarrolladas por otros usuarios.

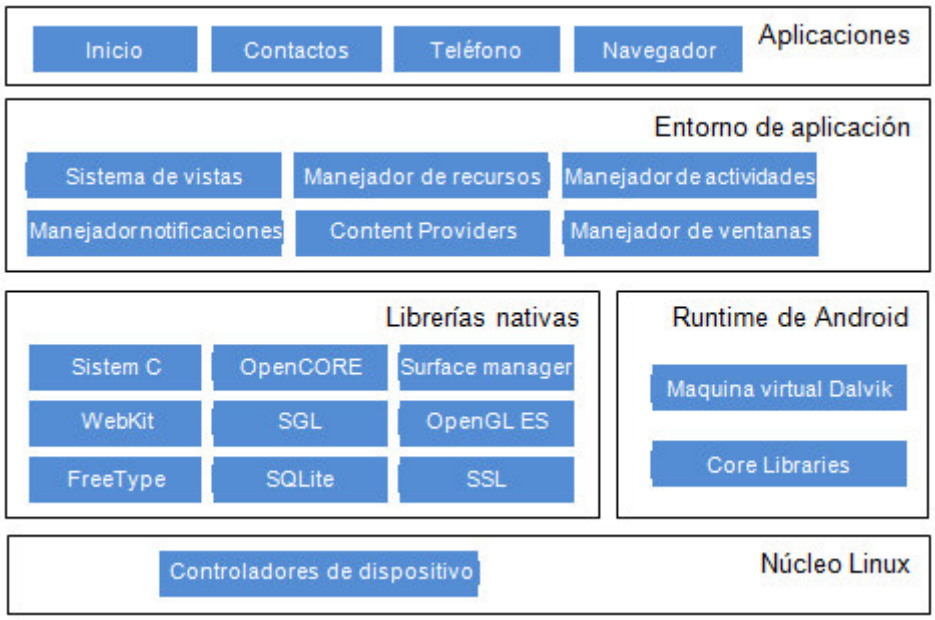


Figura 2.4 Capas de la arquitectura de Android

2.2.3. Componentes de una aplicación Android

Son los elementos clave en el desarrollo de aplicaciones para Android. Pueden representar

- *Views* (vistas): Cualquier elemento que forma la interfaz de usuario de la aplicación. Se pueden definir utilizando código Java, pero lo normal, es definirlas en un fichero XML.
- *Layout*: Es un conjunto de vistas agrupadas de una forma determinada, existe diferentes tipos de layout y al igual que las vistas se pueden definir en el código o en un fichero de XML.
- *Service* (Servicios): Se asemejan a los demonios de Unix o los servicios de Windows. En definitiva son procesos que se ejecutan en un segundo plano, y pueden ser locales (utilizados por aplicaciones del mismo terminal) o remotos (utilizados por aplicaciones desde otros terminales).
- *Intent* (Intención): Representan la voluntad de realizar alguna acción, como realizar una llamada o visitar una página web, es obligatorio utilizarlos en :

  - Lanzar una actividad
  - Lanzar un servicio
  - Lanzar un anuncio de tipo broadcast
  - Comunicarnos con un servicio.
- *Broadcast receiver* (Receptor de anuncios): Componente que permite registrar eventos del sistema, y son notificados una vez que suceden. Un ejemplo de este componente puede ser, el indicador de batería baja, una llamada entrante, etcétera.
- *Content Provider* (Proveedores de contenido): Es el mecanismo estándar usado en Android para compartir información con otros dispositivos sin poner en peligro la seguridad del sistema de ficheros.
- *Activity* (Actividad): Es el componente principal de una aplicación Android, ya que toda aplicación tendrá al menos una pantalla, esta pantalla será el activity, su principal función es la creación de la interfaz de usuario, y las activities que componen una aplicación son independientes entre sí.
- *Fragment* (Fragmentos): Representa una pequeña porción de la interfaz. Son una sección modular de una actividad.
- Por último hay que mencionar el AndroidManifest.xml, es un archivo XML único que tiene cada aplicación que permite al sistema operativo saber toda la información esencial de la aplicación, como los permisos con los que cuenta la aplicación, las librerías, versiones soportadas, componentes utilizados, etcétera.

2.2.4. Ciclo de vida de una actividad

Como ya se ha comentado anteriormente, una actividad es un elemento básico de visualización, una actividad o un conjunto de ellas forma una aplicación en Android, también puede contener servicios. En una aplicación Android el ciclo de vida lo controlan las actividades dado que el usuario no cambia de aplicación sino de actividad. Por esto el sistema mantendrá una pila de actividades con las actividades previamente visualizadas para que el usuario pueda volver a la anterior pulsando la tecla “atrás”.

La característica más importante de Android es que la destrucción de un proceso no es controlado por la aplicación si no por el sistema, es él quien decide cuando destruir el proceso. Si el sistema elimina el proceso de una aplicación y el usuario vuelve a ella, creará un nuevo proceso pero se habrá perdido el estado que tenía la aplicación. Esto hace que el desarrollador tenga la responsabilidad de almacenar el estado de las actividades, si se desea que conserve su estado cuando sea reiniciada. En la figura 2.5 [8] podemos ver todas las fases del ciclo de vida de una aplicación en Android.

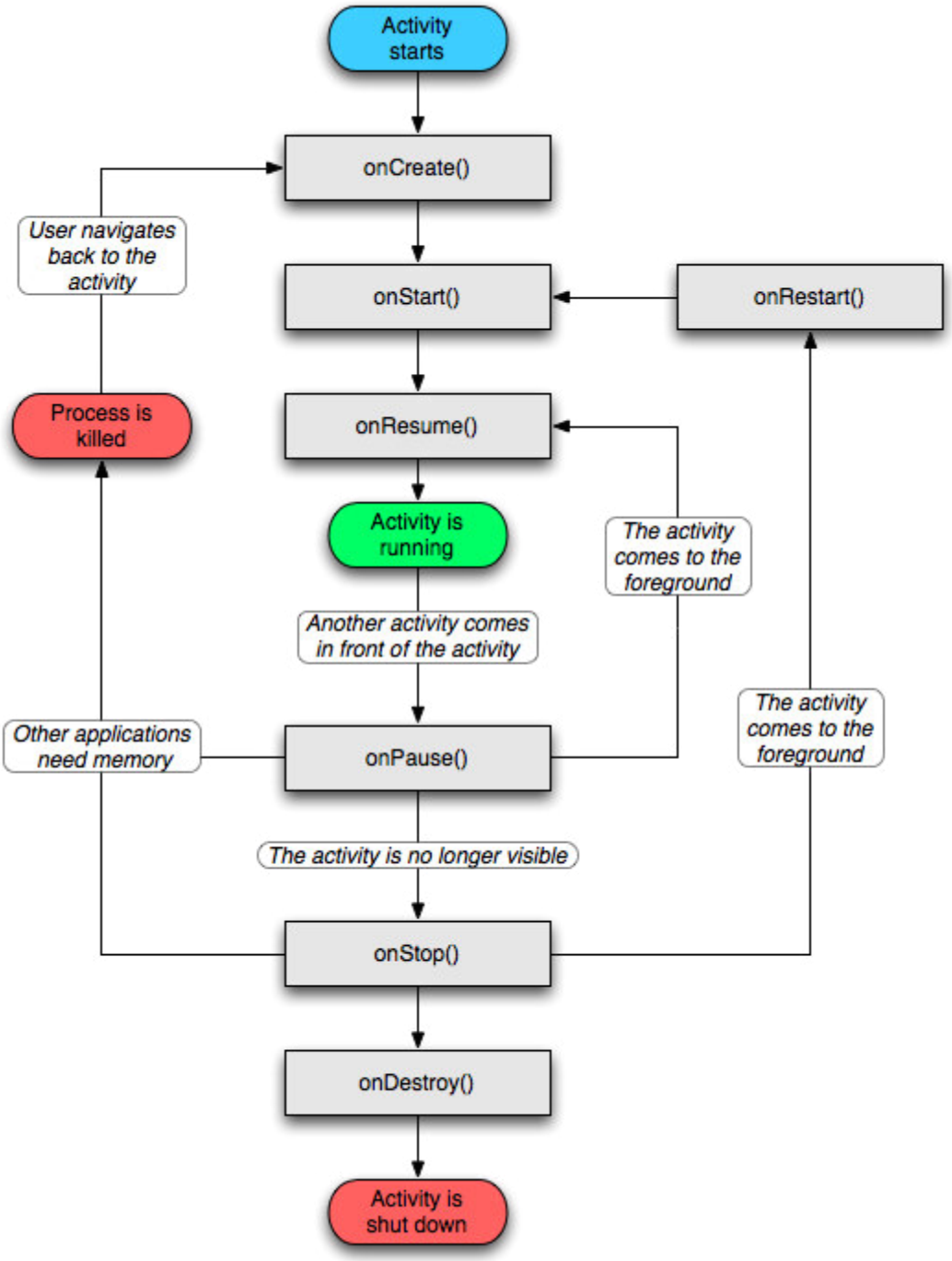


Figura 2.5 Ciclo de vida de una Actividad en Android

2.2.5. Ciclo de vida de un Fragmento.

Aparecen con las versión 3.0 (API 11) de Android por la necesidad de adaptar a las diferentes orientaciones del dispositivo la interface independientemente del dispositivo que se use.

Están alojados dentro de una actividad siempre y se ven afectados por el ciclo de vida de la actividad, si la actividad se pausa, todos los fragmentos contenidos en ella se pausaran. Como se muestra en la figura 2.6 [9], independientemente del ciclo de vida de la actividad que los contiene, los fragmentos tienen su propio ciclo de vida.

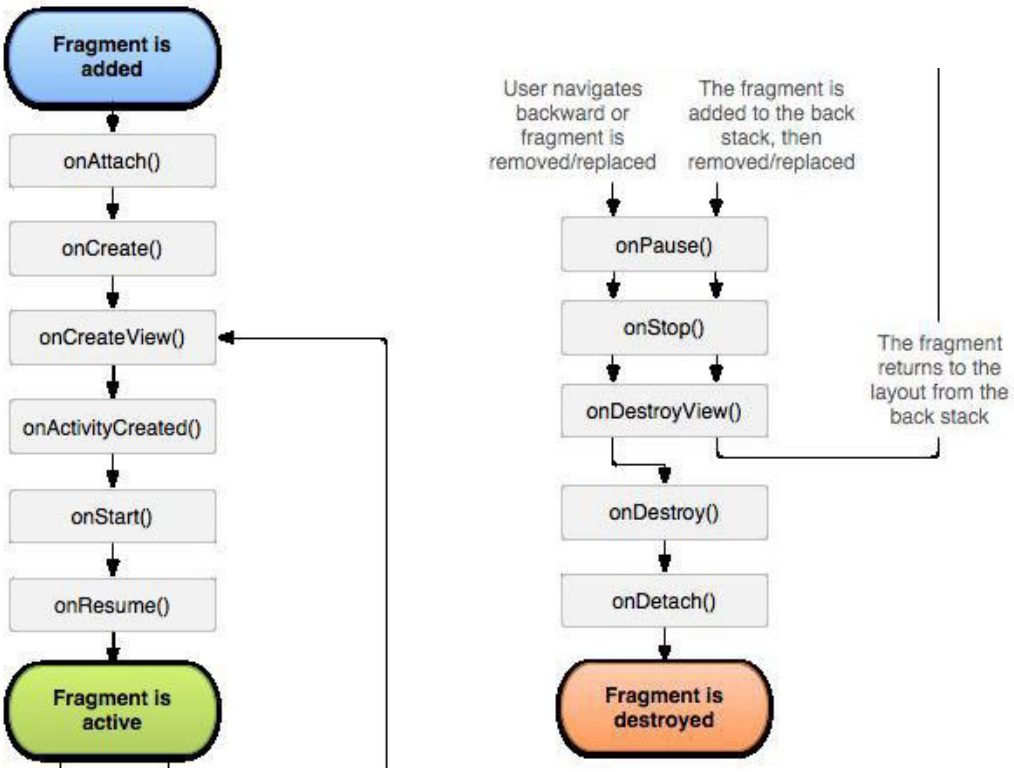


Figura 2.6 Ciclo de vida de un Fragmento

De todos los estados por los que puede pasar el ciclo de vida de un fragmento, siempre se recomienda implementar al menos lo siguientes:

- *OnCreate ()*: El sistema llama a esta función cuando la aplicación declara el fragmento, aquí se inicializan los elementos principales del fragmento.
- *OnCreateView ()*: Es llamada cuando el fragmento es cargado por primera vez en la interfaz.
- *OnPause ()*: Al menor indicio de que el usuario va a dejar el fragmento el sistema llama a esta función. Aquí es donde generalmente se guardan los cambios realizados que deberán usarse en futuras sesiones del usuario.

2.2.6. Seguridad.

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

En cuanto a las aplicaciones en concreto, Android define un nivel de seguridad más, que les permite el acceso a las distintas funcionalidades del sistema. Para que una aplicación pueda hacer uso de ciertas características, protegidas del dispositivo, es necesario declarar los permisos pertinentes dentro del archivo AndroidManifest.xml [10].

2.2.7. Almacenamiento

Android proporciona distintas opciones para el almacenamiento de información. En función de las necesidades de nuestra aplicación hemos de considerar una u otra opción. Vamos a describir algunas de ellas [11]:

- *Preferencias compartidas*: Es una clase que permite el almacenamiento de las preferencias del usuario en nuestra aplicación, permitiendo

almacenar los valores en pares de tipo clave/identificador y valor asociado a cada clave.

- *Almacenamiento interno*: Almacenamiento de datos en la memoria del dispositivo. Estos datos son de carácter privado, lo que significa que sólo nuestra aplicación puede acceder a ellos.
- *Almacenamiento externo*: Los dispositivos compatibles con el sistema operativo Android soportan almacenamiento externo. Los archivos guardados en este tipo de almacenamiento son accesibles desde fuera de nuestra aplicación y pueden ser modificados.
- *Bases de datos*: Android proporciona soporte para bases de datos SQLite. Todas las bases de datos creadas desde nuestra aplicación son accesibles desde cualquier clase de la misma, pero no por otras aplicaciones.

2.2.8. Estructura de un proyecto Android

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

Un proyecto de una aplicación para Android está formado por una jerarquía de directorios y archivos que conforman el esqueleto básico de la aplicación, como se aprecia en la figura 2.7.

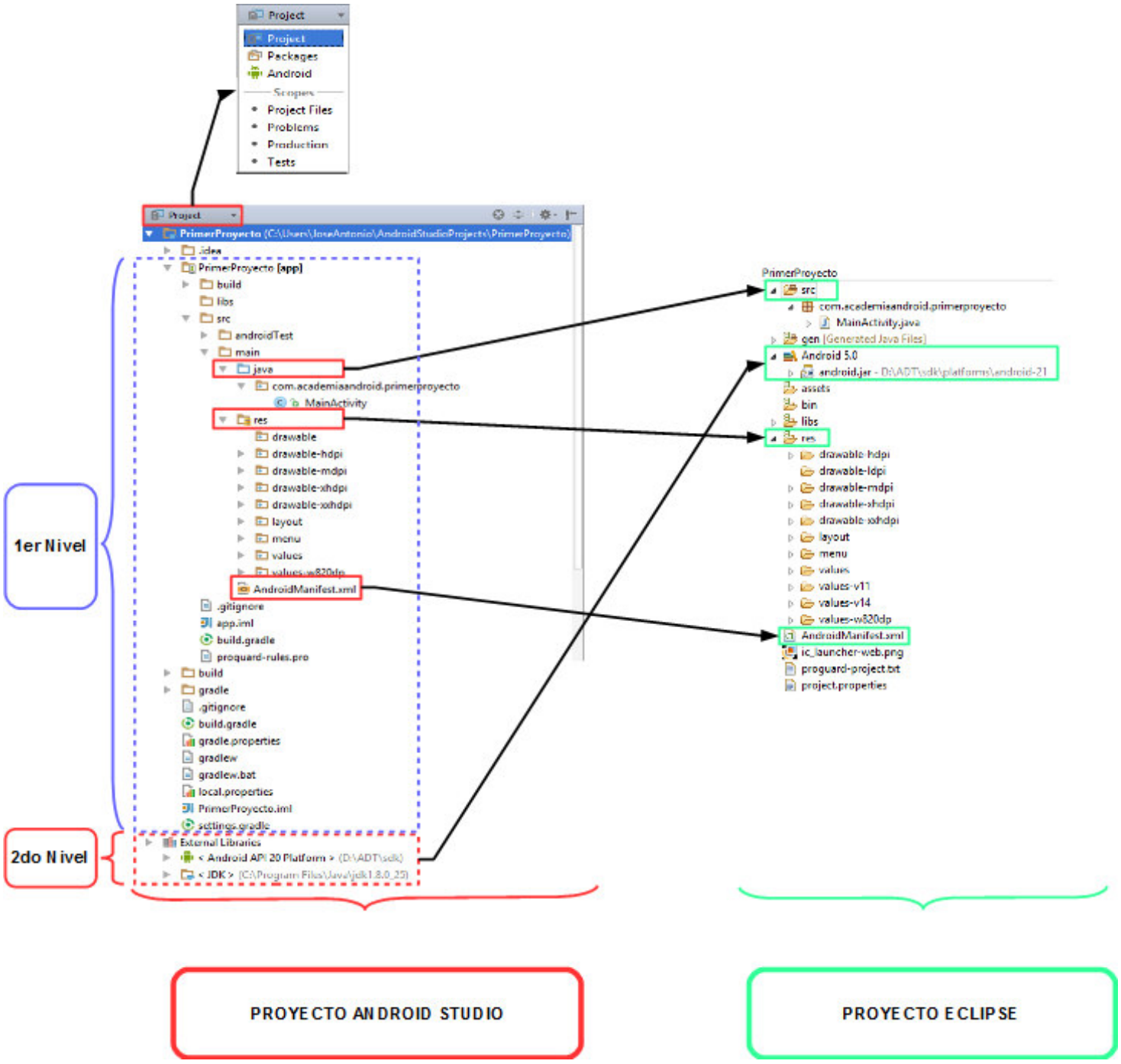


Figura 2.7 Comparativa de la estructura según el IDE (izquierda Eclipse, derecha Android Studio)

A continuación, vamos a introducir brevemente la utilidad y el contenido de los principales archivos y directorios, usados para realizar una aplicación Android:

- *AndroidManifest.xml*: Descrito en el apartado 2.2.3
- *Src/java*: Contiene los archivos de código fuente (.java) que controlan la lógica de la aplicación. Se organiza en paquetes de igual manera que las

aplicaciones Java, e inicialmente puedes encontrar en ella el archivo correspondiente al código fuente de la Activity que se ha creado al generar el proyecto.

- *Src/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.

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

- *External Libraries*: Android 4.4.2: Contendrá una librería android.jar que permitirá dar acceso a las APIs según las versiones elegidas y a otras librerías incluidas por el desarrollador necesarias para la aplicación.

2.3. Servicios Web

En el contexto de la arquitectura empresarial, la orientación a servicios y la arquitectura orientada a servicios, el servicio se puede definir como [12]:

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

Por otro lado OASIS (Organización para el Avance de Estándares de Información Estructurada) define un servicio como [12]:

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

Durante la realización de este trabajo, nos referiremos a los servicios como servicios web, ya que la web será el mecanismo de comunicación entre el cliente y el servidor. Un servicio web se puede definir como [13]:

"Un conjunto de aplicaciones o de tecnologías con capacidad para operar en la Web. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web."

Los servicios web pueden también ser considerados como una API web ya que suelen ser accedidos desde una red, la mayoría de las veces mediante internet y ejecutados en el sistema que los aloja (véase figura 2.8). Dicha figura ejemplifica como los usuarios acceden igualmente a un servidor web, que a un servidor de servicios mediante la capa HTTP (ejemplificada en la figura como internet) y es totalmente transparente para ellos, al tipo de servidor que acceden, la forma en la obtienen los datos, etcétera.

Las características de los servicios web son:

- Se basan en el protocolo HTTP
- Su flexibilidad
- Capacidad de interoperación

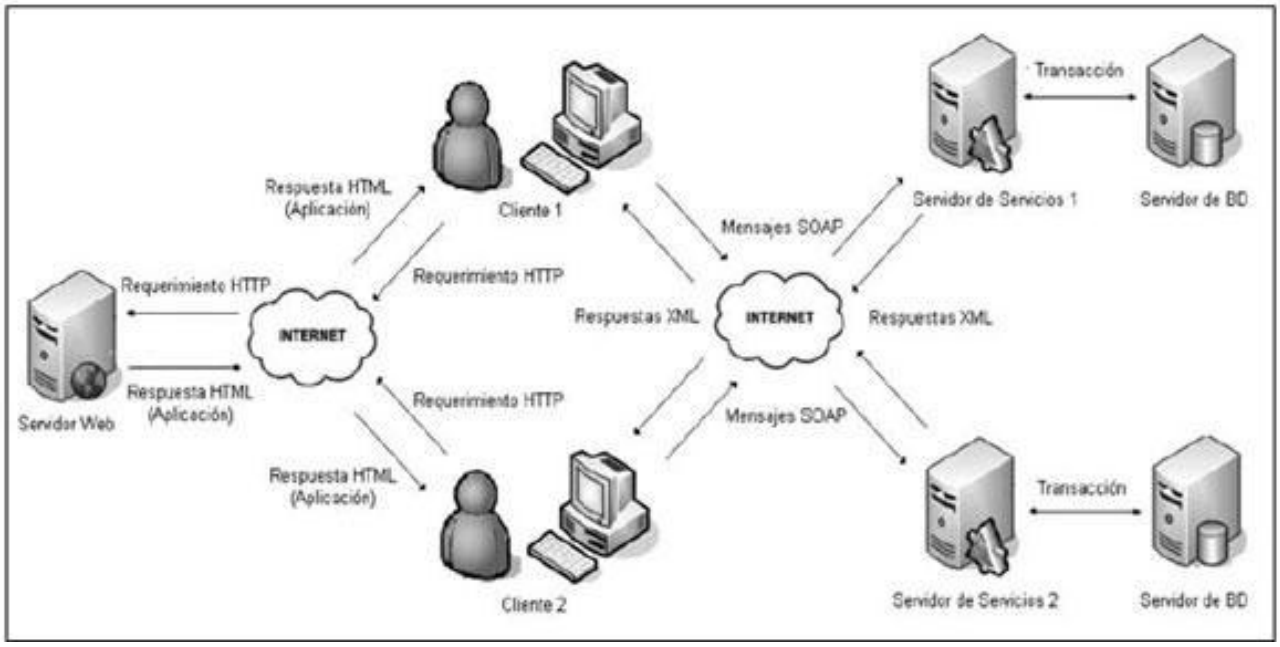


Figura 2.8 Esquema Servicio Web

A continuación, vamos a ver las ventajas y desventajas de los servicios web, una breve introducción a las tecnologías que utilizan los servicios web.

2.3.1. Ventajas de los servicios web

- Al ser accedidos de manera remota, ofrecen una gran interoperabilidad entre aplicaciones software basadas en diferentes plataformas independientemente de sus propiedades.
- Permiten integrar fácilmente servicios y software de otras compañías ubicadas en otros lugares geográficos diferentes.
- Fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

2.3.2. Desventajas de los servicios web

- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (Common Object Request Broker Architecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (Remote Method Invocation),

CORBA o DCOM (Distributed Component Object Model). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.

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

2.3.3. Tecnologías empleadas en los servicios web

Existen multitud de tecnologías para implementar un servicio web, en general las más utilizadas hoy en día son SOAP y REST. Vamos a ver en qué consisten [13,14, 16].

SOAP (Simple Object Access Protocol)

SOAP [16] es un protocolo para el intercambio de mensajes sobre redes de computadoras, generalmente usando HTTP. Está basado en XML, esto facilita la lectura, pero también los mensajes resultan más largos y, por lo tanto, considerablemente más lentos de transferir.

Existen múltiples tipos de modelos de mensajes en SOAP pero el más común es el “Remote Procedure Call” (RPC) o Llamada a Procedimiento Remoto, en donde un nodo de red (el cliente) envía un mensaje de solicitud a otro nodo (el servidor) y el servidor inmediatamente responde el mensaje al cliente.

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

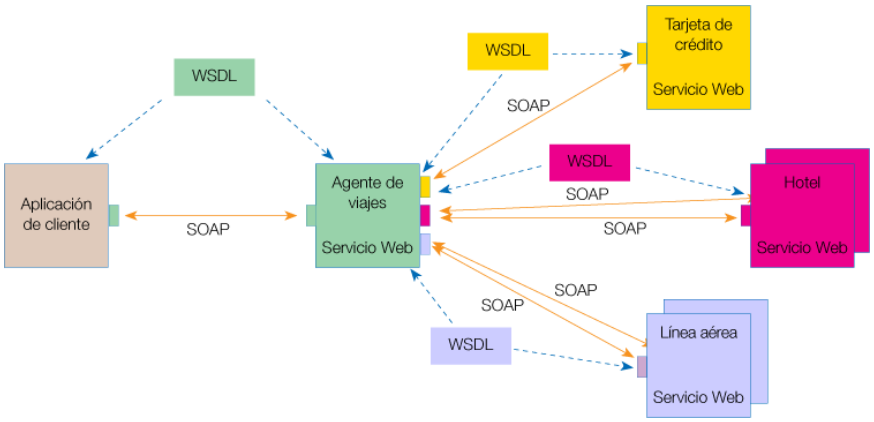


Figura 2.9 Arquitectura Servicio Web con SOAP

REST (REpresentational State Transfer)

REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. Y se refiere originalmente a un conjunto de principios de arquitectura para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos (Véase figura 2.10).

En la actualidad se usa en el sentido más amplio para describir cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos, en cualquier formato.

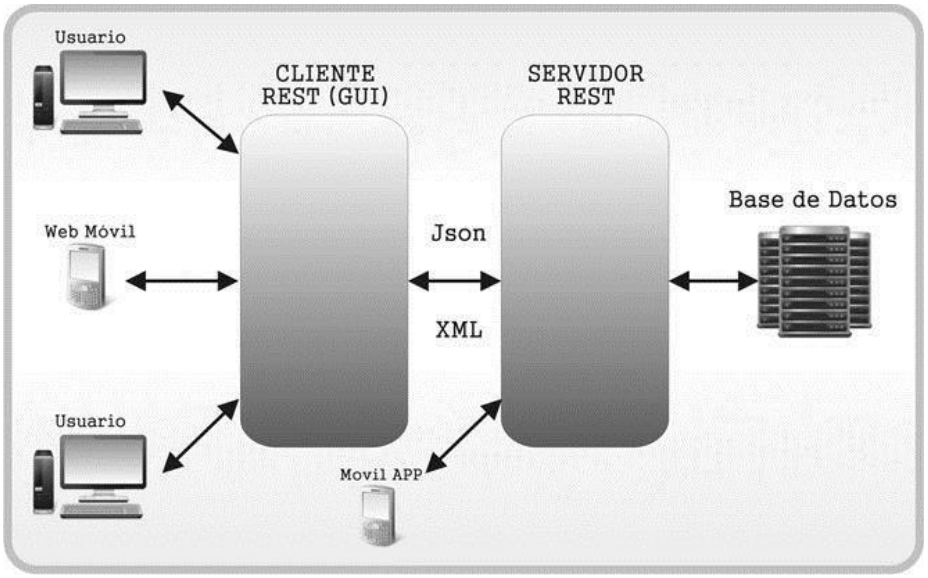


Figura 2.10 Arquitectura Servicio Web con REST

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

2.3.4. Servicios REST

Este término es utilizado en su mayoría para describir a cualquier interfaz que transmite datos específicos de un domino sobre HTTP, que usa un conjunto de operaciones bien definidas, HTTP define un pequeño conjunto de operaciones, las más importantes GET, POST, PUT y DELETE. Dicha interfaz también tiene una sintaxis universal para identificar los recursos.

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

• HTTP

• URL

• Representación de los recursos: XML/HTML/GIF/JPEG/…

• Tipos MIME: text/xml, text/html,

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

- *Cliente-servidor*: Define que deben de estar separados el cliente del servidor a través de interfaces uniformes, es decir el cliente no sabe nada de cómo se almacena la información, ni como se está obteniendo. Por otro lado los servidores no saben la manera en la que se está presentado la información en el cliente.
- *No manejan estado*: El servidor no debe de contener ningún contexto sobre el cliente que está haciendo la solicitud. La solicitud del cliente debe tener toda la información necesaria para poder procesar la solicitud en el servidor, esto permite crear aplicaciones más escalables sin que tener preocupación sobre cómo debe de responder el servidor a la perdida de la sesión del cliente por perdida de conectividad.
- *Capaces de almacenarse en caché*: En el WWW los clientes no tienen un mecanismo de almacenar las respuestas en caché. Las respuestas

deben de estar implícitas o en su defecto explícitamente deben definirse a sí mismas como almacenables en caché o no, para evitar que los clientes hagan uso inapropiado de información regresada por una solicitud.

- *Sistemas en capas*: El cliente no debe saber si está conectado directamente a un servidor final o a un intermediario. Un servidor intermediario puede ayudar a balancear las cargas y la escalabilidad de la aplicación.
- *Código bajo demanda*: Los servidores pueden ser capaces de extender la funcionalidad de un cliente transfiriéndole lógica que puedan ejecutar, por ejemplo Java Applets o JavaScript.
- *Interface Uniforme*: Son recursos individuales que deben de estar incluidos dentro de la solicitud.

Una API RESTful bien diseñada debería soportar los métodos HTTP más comunes (véase tabla 2.4). Existen otros métodos HTTP pero no suelen ser usados normalmente. Cada método debería usarse dependiendo del tipo de operación que se quiera realizar.

Método	Acción
Get	Obtener un recurso
Post	Crear un recurso
Put	Actualizar un recurso
Delete	Eliminar un recurso.

Tabla 2.4 Métodos HTTP más comunes.

Los códigos de estado HTTP (véase tabla 2.5) en el cuerpo de la respuesta indican a la aplicación cliente que acción debería realizar con la respuesta. Por ejemplo si el código de la respuesta es 200 significa que la petición al servidor se ha procesado correctamente. De la misma manera si el código de respuesta es el 401, la petición realizada no estaba autorizada.

Código de estado	Significado
200	Correcto
201	Creado
304	No modificado

400	Petición Incorrecta
401	No autorizado
403	Olvidado
404	No encontrado
422	Entidad no procesable
500	Error interno del servidor

Tabla 2.5 Códigos de estado HTTP

Para finalizar el capítulo hablaremos de las URLs en REST, dichas URLs deberían estas bien formadas, identificar fácilmente el recurso, ser independientes del formato (si un recurso es un archivo que se llame 234.pdf, el acceso al recurso debería ser /file/234 en lugar de /file/234.pdf) y por último mantener una jerarquía lógica. Si la API a la que se accede necesita una key o parámetros que no deban estar a la vista de cualquiera, se pueden añadir a los headers HTTP.

Ejemplos:

- GET http://servidor.es/API/facturas, Nos permite acceder a todo el listado de facturas.
- POST http://servidor.es/API/facturas, Nos permite crear una factura nueva.

CAPÍTULO 3 Sistemas de Recomendación

En este capítulo se centrará en los Sistemas de recomendación (SR), comenzando por una breve introducción, seguido de una clasificación de los SR. Finalmente concluiremos el capítulo haciendo una revisión de las tendencias actuales en los SR

3.1. Introducción

Como ya hemos visto en capítulos anteriores, existen gran variedad de dispositivos móviles smartphones, smartwatches, etcétera. Estos dispositivos proporcionan una gran cantidad de información acerca del usuario, este información puede ser aprovechada para conocer mejor a los usuarios, sus gustos, ideas, intereses, etcétera, y personalizar la información que se les facilita, o por otro lado ayudarles a decidir qué producto comprar.

A día de hoy estos sistemas forman parte de nuestro uso diario, debido a que en el año 2011 la cantidad de información generada por la humanidad era de 600 exabyte [23], 1 exabyte es 1000 petabyte y 1 petabyte es 1000 terabyte. Esto nos da una idea de la magnitud del problema a los que se enfrentan los usuarios a la hora de elegir un producto u otro, en internet. Aquí es donde los SR toman partido para ayudar y guiar al usuario entre tal cantidad de información.

Los SR se pueden definir [24, 25,26] como un conjunto de herramientas y software con capacidad de ofrecer sugerencias adaptándose a los intereses de los usuarios.

Los SR van dirigidos a usuarios que carecen de la suficiente experiencia o tiempo para evaluar el potencial de un enorme número de ítems diferentes. Demostrando ser un valioso medio para evitar la sobrecarga de información en distintos ámbitos como el comercio electrónico, turismo, educación, etcétera.

En su forma más simple una recomendación se presenta como una lista ordenada de ítems. Para realizar esta tarea el SR se basa en las preferencias, restricciones y conocimiento del usuario. Estos intereses y conocimiento del usuario deben quedar reflejado de forma explícita para que puedan ser procesados

computacionalmente y poder predecir los productos o servicios más adecuados para él.

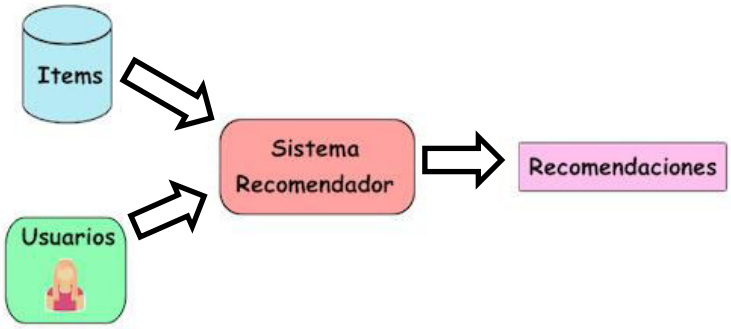


Figura 3.1 Esquema Simplificado SR

Los principales elementos que forman un SR son los usuarios, los ítems que son recomendados a los usuarios, y las valoraciones que los usuarios realizan sobre los ítems recomendados.

Los ítems son los objetos que van a ser recomendados, dependiendo del dominio de aplicación del SR, estos ítems variarán en su complejidad y valor de utilidad. El valor de un ítem puede ser positivo, negativo, este valor va asociado al coste cognitivo o monetario que le supone al usuario buscar ese ítem.

Los usuarios del SR pueden tener diferentes características e interés para usar el SR, por lo que la manera de modelar a los usuarios en nuestro SR depende del modelo de recomendación que adoptemos. Más adelante veremos las diferentes maneras de representar a los usuarios cuando valoremos los diferentes tipos de SR.

Las valoraciones son las transacciones almacenadas en el sistema de recomendación. Las transacciones son implícitas cuando el sistema trata de inferir las opiniones de los usuarios basándose en sus acciones y serán explícitas cuando se le pide al usuario una opinión en una escala de valoración. Las valoraciones explícitas se pueden obtener de la siguiente forma [27]:

- *Valoraciones numéricas*: De 1-5 estrellas.
- *Valoraciones ordinales*: Donde se le solicita al usuario que seleccione el término que mejor indica su opinión.

- *Valoraciones binarias*: En las que el usuario indica si el ítem es bueno o malo.
- *Valoraciones unarias*: Indican que el usuario ha contemplado el ítem o lo ha adquirido.

Actualmente existen varios tipos de SR, entre los que destacan los Sistemas de Recomendación Colaborativos, basados en contenido, basados en conocimiento e híbridos. Más adelante se verán las características de estos sistemas.

Para concluir este epígrafe vamos a enumerar las motivaciones para implementar un Sistema de Recomendación [28]:

- *Incrementar el número de ítems vendidos*: Es la finalidad más importante de un SR comercial. Alcanzar este objetivo es posible porque los ítems recomendados encajan con las necesidades del usuario.
- *Vender ítems más diversos*: Otra función de los SR es permitir al usuario seleccionar ítems que posiblemente hubieran sido difíciles de encontrar sin su recomendación.
- *Incrementar la fidelidad del usuario*: Un usuario será fiel a un sitio web, cuando lo orienta correctamente en las recomendaciones.
- *Incrementar la satisfacción del usuario*: Un SR bien diseñado puede además mejorar la experiencia del usuario con el sitio o con la aplicación.
- *Mejor entendimiento de lo que el usuario busca*: Un SR puede aprovecharse para otras muchas aplicaciones, como es la descripción de las preferencias del usuario, tanto las recogidas explícitamente como las predichas por el sistema.

Estas características motivan a los proveedores de servicios y productos a implementar un SR. Pero no solo los proveedores pueden querer un SR, si aporta ayuda de forma efectiva a realizar sus tareas, también los usuarios pueden demandar un SR.

3.2. Clasificación de los Sistemas de Recomendación

Como ya vemos en la introducción actualmente hay diferentes tipos de SR según, se basen en contenido, conocimiento, colaborativos, etcétera, Pero primero vamos a distinguir entre las recomendaciones no personalizada y la recomendaciones personalizadas.

3.2.1. No Personalizados

La recomendación no personalizada se utiliza para solventar el problema que tienen los SR cuando accede un nuevo usuario, la ausencia de valoraciones para descubrir sus gustos y en consecuencia recomendar ítems en base a sus gustos. A este problema se le llama “Arranque en frío” (Cold Start).

Este problema se podría solucionar obteniendo valoraciones implícitas del usuario en las redes sociales, para tener un base sobre la que poder realizar una recomendación en base a sus gustos.

Las ventajas de este tipo de SR es la facilidad de implementar para que los ítems más populares o mejores valorados sean mostrados a los usuarios. Como contra punto la recomendación en estos sistemas es la misma para todos los usuarios.

3.2.2. Personalizados.

Al contrario de la recomendación no personalizada, este tipo de recomendación tiene en cuenta los gustos y opiniones de los usuarios, registrados de manera explícita en el SR, para realizar una recomendación única para cada usuario.

A continuación vamos a ver los diferentes tipos de SR de recomendación personalizada.

3.2.2.1. Basados en conocimiento.

Este tipo de sistemas usa bases de conocimiento sobre los usuarios y los productos para ver cuál satisface las necesidades de un usuario concreto. En todo este capítulo hemos hablado que los SR almacenan las valoraciones que hacen los usuarios a los ítems, en este tipo, el sistema almacena las necesidades de los usuarios.

Los SR basados en conocimiento tienden a trabajar mejor al principio que por ejemplo los basados en filtrado colaborativo, ya que necesita una gran cantidad de valoraciones de muchos usuarios, si no la precisión de la recomendación será bastante baja.

3.2.2.2. Basados en Contenido.

Este tipo de SR construye un modelo analizando las características y descripción de los ítems que el usuario ha valorado. Si el usuario ha valorado positivamente un video de natación, el sistema aprenderá a recomendar videos de este tipo.

Las ventajas de este tipo de sistemas son [31]:

- *Independencia del usuario*: Usa sólo los rating realizados por el usuario, para construir su recomendación, es decir, no necesita que otros usuarios hayan valorado más ítems.
- *Transparencia*: Se puede saber y conocer cuáles son las características de los ítems, que hacen que el ítem sea valorado.
- *Ítems nuevos*: Permite valorar y recomendar ítems introducidos posteriormente en el sistema y que no han sido valorados por nadie.

Desventajas de este tipo de sistemas son [31]:

- *Análisis limitado del contenido*: El número y tipo de características de los ítems que pueden analizar de manera automática es reducido,
- *Sobre especialización*: Este tipo de SR tiende recomendar ítems del mismo tipo que los más valorados por el usuario, así si el usuario ha valorado solo películas de Stanley Kubrick el SR solo le recomendará películas del mismo tipo.
- *Usuarios nuevos*: Las recomendaciones realizadas a usuarios nuevos tiene un bajo grado de confianza, es necesario que el sistema recoja una gran cantidad de valoraciones del usuario para que pueda entender los intereses del usuario y proporcionar una recomendación precisa.

3.2.2.3. Basados en filtrado Colaborativo.

Es la técnica de recomendación más popular y ampliamente implementada. Se basa en recomendar al usuario ítems que usuarios con gustos similares valoraron positivamente. El funcionamiento de esta técnica es el siguiente:

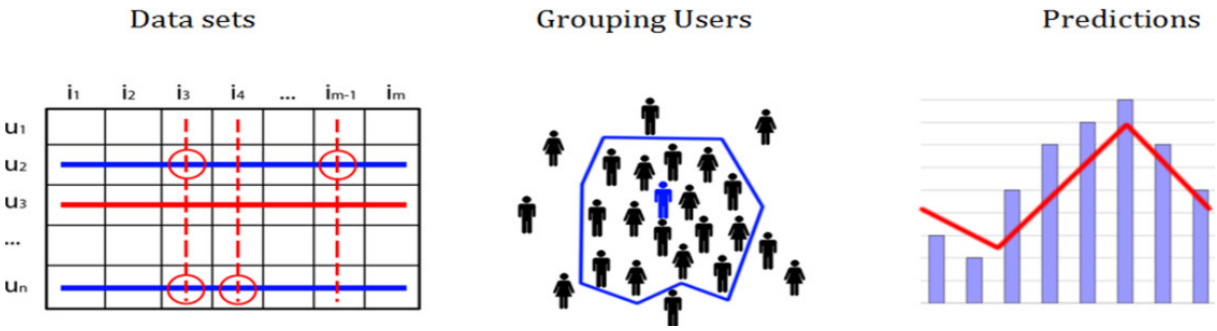


Figura 3.2 Esquema filtrado colaborativo [32]

1. Se calcula la similitud del usuario activo con los usuarios.
2. Se escogen los Knn vecinos más cercanos.
3. Se calcula la predicción en base a las valoraciones de los knn vecinos más cercanos.

Fórmula para calcular predicción:

$$p_{u,i} = \overline{r_u} + \frac{\sum_{u´ \epsilon N}(r_{u´,i} - \overline{r_{u´}}) \times s(u, u´)}{\sum_{u´ \epsilon N} |s(u, u´)|}$$

Dónde:

u = usuario activo

i = ítem a calcular

u´ = usuario perteneciente al Vecindario

N = subconjunto de usuario (N más cercanos)

s(u,u´) = similitud del usuario activo u con el usuario vecino

(Ecuación 3.1)

Para calcular la similitud los métodos más comunes son el coeficiente de correlación de Pearson, en el cual se mide la dependencia entre 2 variables. El coeficiente de correlación de Pearson no es indicado para usuarios con pocas valoraciones ya que tiende a dar un alto nivel de similitud.

Otro método de similitud es el coseno. Este método de similitud trata a los usuarios como un vector de dimensión m, siendo m el número total de ítems.

$$s(u,u\acute{}) = \frac{\overrightarrow{r_u} \cdot \overrightarrow{r_{u\acute{}}}}{||r_u||^2 \cdot ||r_{u\acute{}}||^2} = \frac{\sum_i r_{u,i} \cdot r_{u\acute{},i}}{\sqrt{\sum_i r_{u,i}{}^2} \cdot \sqrt{\sum_i r_{u\acute{},i}{}^2}}$$

Dónde:

$$r_{u,i} = Puntuación\ del\ usuario\ u\ al\ item\ i$$

$$r_{u\acute{},i} = Puntuación\ del\ usuario\ u\acute{}\ al\ item\ i$$

(Ecuación 3.2)

Como podemos ver el filtrado colaborativo sufre de problemas de escalabilidad cuando la base de datos de usuarios crece, esto es debido a que la búsqueda de los vecinos para un usuario tiene una complejidad algorítmica O(N), siendo N el conjunto de usuarios, o peor complejidad dependiendo de cómo se calcule la similitud de usuarios.

Para solucionar esto hay otra aproximación basada en similitud entre ítems, en lugar de entre usuarios, como hemos visto anteriormente.

En el filtrado colaborativo basado en ítem, la predicción se calcula teniendo en cuenta las valoraciones de los ítems. Esta técnica es similar a la recomendación basada en contenido, pero en lugar de calcular la similitud a partir de las características de los ítems, se calcula en base a los patrones de los usuarios.

A continuación podemos ver en la figura 3.3 una comparativa de cómo funciona el filtrado colaborativo basado en usuario y en ítem.

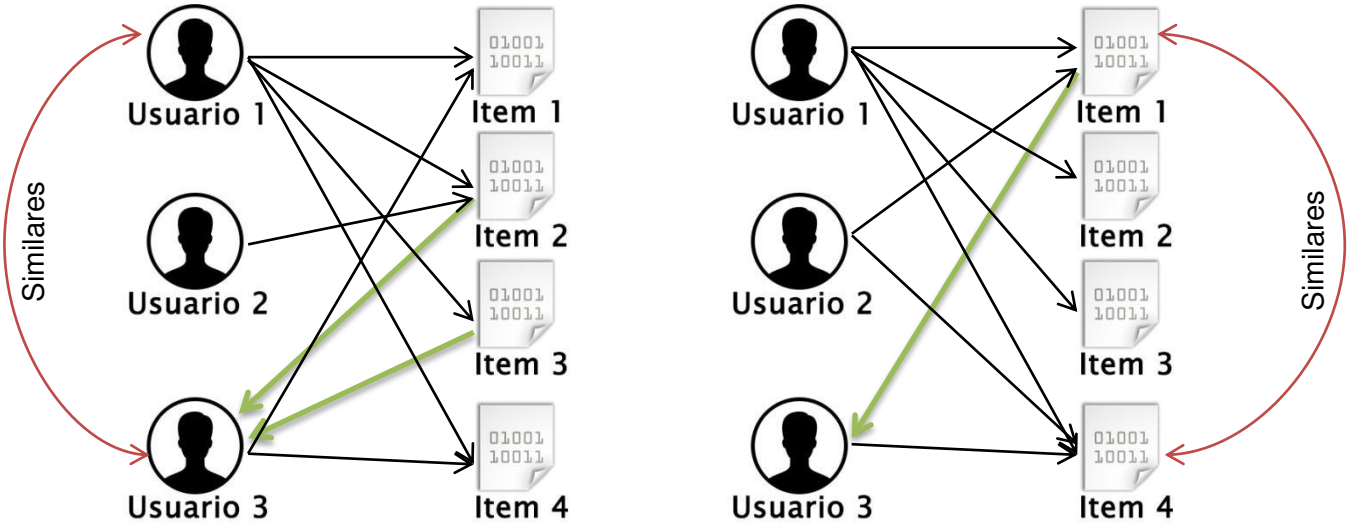


Figura 3.3 Comparativa Entre user-user e item-item

En la figura 3.3 podemos ver a la izquierda el funcionamiento del filtrado colaborativo basado en usuario, en el que el usuario 1 ha valorado todos los ítems, el usuario 2 ha valorado solo el ítem 2 y el usuario 3 ha valorado los ítems 1 y 4, por lo que al ser el usuario 1 y 3 similares, el sistema le recomendará los ítems 2 y 3. A la derecha tenemos el filtrado colaborativo basado en ítem, en el que el usuario 1 ha valorado todos los ítems, el usuario 2 ha valorado los ítems 1 y 4 y el usuario 3 ha valorado el ítem 4, cuando el SR calcule la similitud de los ítems, descubrirá que el ítem 1 y 4 son similares, realizando al usuario 3 la recomendación del ítem 1.

3.2.2.4. Basados en filtrado Hibrido.

Como hemos visto, todas las técnicas de recomendación tienes sus ventajas e inconvenientes, los SR basados en filtrado hibrido, combinan 2 o más técnicas de recomendación para paliar los inconvenientes de una técnica con las ventajas de otra técnica.

Por ejemplo los sistemas basados en filtrado colaborativo tienen el problema del ítem nuevo, que no podría ser recomendado al no haber sido valorado. Este problema se puede solucionar al implementar otra técnica de recomendación junto al filtrado colaborativo como puede ser la recomendación basada en contenido que no sufre del problema del ítem nuevo como ya vimos anteriormente.

3.3. Tendencias

En la actualidad la investigación en el campo de los SR se está centrando en la mejora de las recomendaciones mediante el uso de información del usuario. La manera de obtener esta información es de lo más variada, puede provenir de patrones de navegación, análisis de redes sociales, dispositivos móviles, etcétera.

En concreto de los dispositivos móviles podemos obtener con quien está, donde está o qué hora es, por ejemplo. Esto nos puede ayudar a mejorar las recomendación basándose en el lugar que este el usuario, para no mostrarle todas las posibilidades, si no las más cercanas, o sabiendo con quien está para tener en cuenta los gustos de sus acompañantes.

A continuación se verán dos técnicas para mejorar la recomendación basándose en el contexto del usuario y realizando una recomendación a grupos.

3.3.1. Recomendación Basada en contexto.

En los SR el conocimiento de este contexto tiene un gran interés, ya que la función de los SR es recomendar los ítems de adecuados, a los gustos e intereses del usuario, pero estos intereses pueden variar según donde se encuentre el usuario, o si se encuentra en compañía de otras personas.

Actualmente, esta información contextual está empezando a ser utilizada para mejorar las recomendaciones para por ejemplo paquetes vacacionales, web de comercio electrónico, etcétera,

Pero como tal, el contexto no tiene una definición, sino que depende del dominio en el que se use, así que vamos a enumerar que significado el contexto en algunos dominios de aplicación [33] y luego diremos que es para nosotros el domino:

- *Minería de Datos*: Se define el contexto como aquellos eventos los cuales caracterizan el ciclo de vida de un cliente y que puede determinar un cambio en sus preferencias para una compañía
- *Comercio Electrónico*: La intención con la cual un usuario realiza una compra en una aplicación de comercio electrónico.

- *Computación Ubicua*: Se entiende por contexto en este dominio como la localización del usuario, la identidad de las personas cercanas, los objetos alrededor y los cambios in estos elementos.

Para nosotros en este TFG el contexto del usuario va a hacer referencia a su localización, con la finalidad de mejorar la recomendación. La aplicación del contexto dentro del SR utilizado en este TFG se puede implementar de las siguientes maneras:

- *Pre-filtrado*: En esta técnica se utiliza el contexto del usuario para reducir el conjunto de ítems para calcular la recomendación (véase figura 3.4), esto permite usar técnicas de recomendación que no son eficientes con grandes conjuntos de ítems.

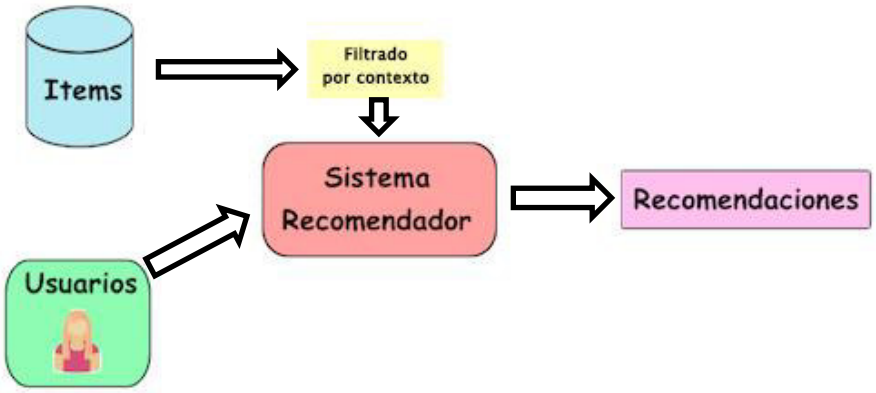


Figura 3.4 Recomendación pre-filtrada

- *Post-filtrado*: Una vez calculadas las recomendaciones del usuario según sus preferencias se aplica un filtrado de estas recomendaciones según su contexto (véase figura 3.5). Esta es la técnica implementada en este TFG.

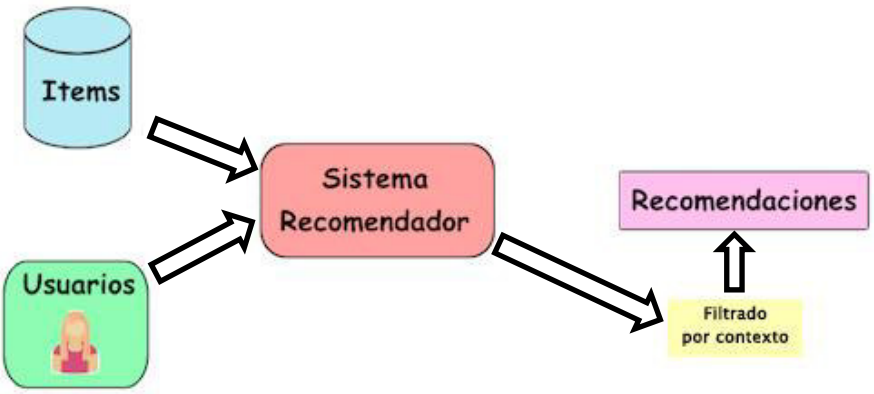


Figura 3.5 Recomendación post-filtrada

También remarcar la existencia de otra técnica de recomendación contextualizada:

- *Modelo contextualizado*: En lugar de almacenar los datos de la forma "tradicional" (<usuario, ítem, puntuación>), esta técnica, construye un modelo en el que también se tiene en cuenta el contexto de la valoración del ítem quedando de la siguiente forma <usuario, ítem, contexto, puntuación>. Añadiendo así el contexto al SR de recomendación en lugar de como simples funciones de filtrado previo al SR o posterior al SR. Dando lugar a un modelo de recomendaciones bastante fiable.

3.3.2. Recomendación a Grupos

Ya hemos visto las diferentes técnicas de recomendación, así como una introducción a las recomendaciones basadas en contexto, pero hay un tipo de recomendación que falta por introducir y que se ha implementado en la aplicación de este TFG a modo de prueba. Es la recomendación a grupos.

Hoy en día ya hemos comentado en la introducción de este capítulo, que tenemos disponible gran cantidad de información del usuario, y para el usuario. Pero siempre nos centramos en único usuario. Pero el ser humano es un ser social e intenta realizar el mayor número de tareas en grupo, está es la motivación de la recomendación a grupos, obtener una recomendación personalizada acorde a todos los gustos e intereses de los integrantes del grupo.

En este TFG para obtener una recomendación a gusto de todos los usuarios, se calcula la media para de valoraciones que han hecho los usuarios del grupo para

cada ítem, una vez obtenido esta lista (m-dimensional) con la valoración media de los usuarios, se aplica el filtrado colaborativo con está lista como si fuera un usuario más y la recomendación realizada por el sistema, en base a esta lista de valoraciones medias de los usuarios del grupo, se les envía a los integrantes del grupo.

CAPÍTULO 4 Ingeniería de Software

4.1. Introducción a la Ingeniería del Software

No hay una definición única para la Ingeniería del Software. A continuación se expondrán algunas definiciones:

"*La ingeniería del software es una disciplina para solucionar problemas comerciales mediante el diseño y desarrollo sistemas basados en software.*" [24]

"La creación y uso de los sólidos principios de ingeniería para obtener económicamente software que es fiable y funciona de manera eficiente en máquinas reales." [25]

El proceso de Ingeniería del Software se divide en las siguientes fases [24]:

- *Requerimientos y Especificación*: Se analiza el dominio en el cual el programa va a trabajar.
- *Diseño*: Asignación de los objetivos y la metodología a seguir para la consecución del proyecto.
- *Implementación*: Codificación del diseño en un lenguaje de programación determinado.
- *Pruebas*: Verificación y Validación del sistema.
- *Despliegue y Mantenimiento*: Ejecutar el sistema, corregir error y añadir nuevas características.

Una vez se ha hecho una breve introducción a la ingeniería del software, vamos a describir el sistema a desarrollar.

Este TFG consiste en el desarrollo de una App Android de recomendación para la hostelería basado en contexto. El sistema recomendará al usuario bares y restaurantes de su agrado mediante un algoritmo de filtrado colaborativo, esta recomendación también tendrá en cuenta geo localización del usuario, que vendrá determinada por su Smartphone.

4.1.1. Requisitos Funcionales

Es el primer paso de la Ingeniería del Software, se engloba dentro de la fase de análisis y un error a la hora de determinar los requisitos funcionales, puede suponer un incremento significativo en coste de desarrollo del sistema, y/o desarrollar un producto totalmente diferente a lo que el cliente solicita.

Los requisitos funcionales recogen el funcionamiento del sistema, “lo que tiene que hacer el sistema”. En el ámbito comercial, estos requisitos se extraen de entrevistas con el cliente o clientes, encuestas, etcétera,

En este caso, el ámbito es académico y dicho requisitos son conocidos desde el comienzo del TFG:

RF 1. Acceso:

- El usuario tendrá la posibilidad de acceder al sistema con su usuario de Google plus. En el caso de que no esté registrado, se registrará automáticamente con el usuario de Google+ .

RF 2. Obtener localización del usuario.

- Acceder a los datos GPS del usuario para obtener su posición o contexto.

RF 3. Recomendar al usuario una lista de restaurantes

- Consiste en ofrecer al usuario un lista de restaurante que puede ser de la siguiente forma:

    - Personalizada: predicción de restaurantes que podrían gustarle, según la información que REJA tiene almacenada del usuario.
    - No personalizada: lista de restaurantes que más le gusta a los usuarios en general, en caso de que este no tenga ninguna posible recomendación personalizada.
    - Basada en contexto: al usuario se le facilitará una lista de restaurantes personalizada, predichos por REJA, dentro de un radio

de búsqueda, en el cual el centro del radio de búsqueda es la posición del usuario.

- Permitir al usuario que modifique el radio de búsqueda de la recomendación. (en escala logarítmica).

- La recomendación al usuario siempre será personalizada excepto en el caso de que el usuario no tenga ninguna posible recomendación personalizada, esta será no personalizada.

RF 4. Visualización de la recomendación:

- Lista: los restaurantes facilitados por REJA se visualizaran en una lista.
- Geo localizada: En este caso los restaurantes se visualizaran en un mapa de Google Maps diferenciando el indicador de los mismos uno de otro por un número o una letra.

RF 5. Visualización de información de los restaurantes:

- La información de los restaurantes se mostrará de la siguiente manera cuando, estos se presenten al usuario en una lista.
  - Nombre.
  - Puntación.

- Para cuando el usuario desee información más detallada del restaurante, se le mostrará:
  - Nombre
  - Puntuación
  - Dirección
  - Teléfono

RF 6. Valorar restaurante.

- Consiste en que el usuario valore dicho restaurante y esta puntación quede registrada.

RF 7. Recomendación a grupos.

- Creación del Grupo: crear un grupo para recibir una recomendación acorde a los gustos de todos los miembros del mismo.

- Gestión del grupo: permite aceptar peticiones de alta en el grupo o eliminar miembros del grupo, así como modificar los datos del mismo, el identificador del grupo.

RF 8. Buscar Restaurantes.

- Facilitar una lista de restaurantes que concuerden con el conjunto de caracteres introducido.
- Simplemente buscara restaurantes por su nombre.

RF 9. Configuración.

- Facilitar al usuario la capacidad de personalizar su perfil, con características como, permitir que al abrir la aplicación siempre se muestre la pantalla de recomendación o la pantalla de búsqueda de restaurantes, la pantalla de recomendación a grupos o la pantalla de configuración.
- Modificar el radio de filtrado para la recomendación contextualizada.

RF 10. Gestionar Perfil.

- El usuario puede modificar las puntuaciones que ha realizado.

RF 11. Desvincular cuenta del dispositivo.

- El sistema ha de facilitar un mecanismo para que el usuario desvincule la cuenta de la aplicación del dispositivo.

4.2. Requisitos no Funcionales.

Habitualmente están relacionados con restricciones a tener en cuenta en relación con los requerimientos del sistema:

- Criterios de rendimiento.
- Previsión del volumen de datos.
- Consideraciones de seguridad.
- velocidad

Puesto que este proyecto consta de una arquitectura cliente servidor, habrá que especificar los requisitos mínimos de la maquina servidor, y del dispositivo cliente:

4.2.1. Servidor

Puesto que el servidor es actualmente la máquina virtual de REJA, la cual ya ha estado en funcionamiento el único requisito mínimo para un normal funcionamiento son:

- Hardware:
  - Memoria RAM: debe ser la suficiente para permitir un uso fluido del sistema.
- Software:
  - PHP 5.3

4.2.2. Cliente

La máquina cliente serán dispositivos móviles, basados en el Sistema Operativo Android

- Requisitos:
  - Tener instalado Google Play Services, en caso de que no esté instalado, la aplicación tiene que facilitar su descarga.
  - La versión del dispositivo debe estar comprendida entre la 2.2 y la 4.x
  - Tener el GPS.

4.3. Casos de uso

Los casos de uso se engloban dentro del análisis del sistema, y representan una funcionalidad concreta dada por el sistema

Como un flujo de eventos. También se puede definir como la representación de una situación o proceso de interacción de un usuario con la aplicación o sistema.

Está formado por:

- Nombre del caso de uso.
- Actores participantes.
- Condiciones de entrada.
- Flujo de eventos.
- Condiciones de salida

4.3.1. Actores.

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

Los Actores del sistema serán:

- Usuario: Corresponde con cada uno de los usuarios que entra en el sistema. Para pedir una recomendación o realizar una valoración.
- REJA: encargado de proporcionar la información necesaria
- Google Plus: encargado de ceder los datos a la aplicación.

4.3.2. Diagrama Frontera

Diagrama que describe completamente la funcionalidad de la aplicación de Android desarrollada en este TFG, con el fin de explicarle al cliente la funcionalidad del sistema completo a alto nivel.

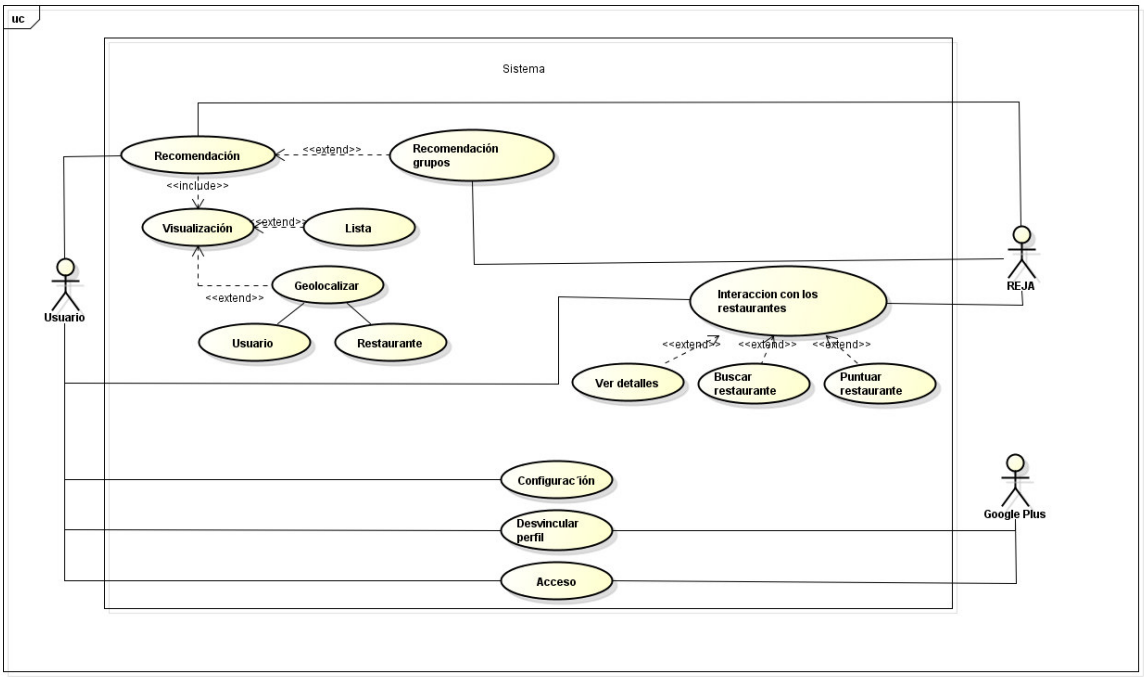


Figura 4.1 Diagrama Frontera

4.3.3. Caso de Uso 1: Acceso

Actores participantes: REJA y usuario.

Condición de entrada: El usuario ha de tener una cuenta en Google+.

Flujos de eventos:

1. El usuario inicia la aplicación.
2. La aplicación muestra el botón de acceso con el método de autentificación estipulado, (Google+, Facebook, Twitter)
3. El método de autentificación despliega una pantalla para ceder los datos personales.
4. El usuario acepta ceder los datos.
5. El método de autentificación cede los datos personales.
6. La aplicación manda el identificador único y personal del usuario a REJA.
7. REJA comprueba el identificador del usuario.
    a. Si el identificador ya existe, accede al sistema.
    b. Si el identificador no existe, lo registra automáticamente y accede al sistema.
8. Tras acceder al sistema, se comprobara el estado del GPS y se le indicara al usuario que es necesario tenerlo activado para recomendaciones contextualizadas.

Condición de salida: el usuario queda identificado en el sistema.

4.3.4. Caso de uso 2: Recomendación.

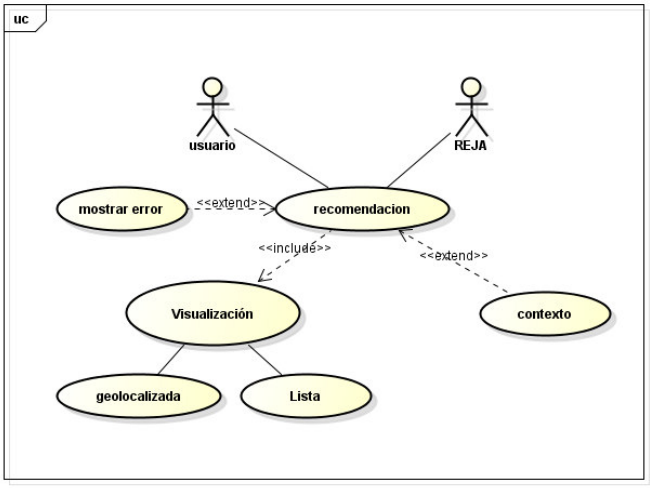


Figura 4.2 Caso de Uso de recomendación

Actores participantes:	REJA y usuario.
Condición de entrada:	El usuario ha accedido al sistema.
Flujos de eventos:	
1	El usuario solicita una recomendación
2	El sistema manda la petición a REJA
3	REJA devuelve la recomendación
4	La aplicación visualiza la recomendación.
5	
a	Si el usuario quiere visualizar los restaurantes de manera secuencial, elige visualizar la recomendación en modo lista y se realiza **S-1**
b	Si el usuario quiere visualizar los restaurantes en un mapa, elige recomendación geo localizada y se realiza **S-2**
c	Si el usuario quiere recibir una recomendación en base a su contexto, se realiza **S-3**
Excepciones	
E-1	En caso de un problema de conexión el caso de uso se reinicia por completo
E-2	En caso de no tener recomendaciones personalizadas, se mostrarán las recomendaciones no personalizadas.

4.3.4.1. Subflujo de eventos:

S-1: Lista de recomendaciones.

Actores participantes: REJA y usuario.

Condición de entrada: El usuario tiene una recomendación.

Flujos de eventos:

1		La aplicación muestra los restaurantes en una lista.
2		
	a	Si desea realizar una recomendación geo localizada, se realiza **S-2**
	b	Si desea realizar una recomendación basada en el contexto, se realiza **S-3**

Excepciones:

E.1	Se informara de cualquier posible error al realizar la conexión con REJA.
E.2	En caso de no tener recomendaciones personalizadas, se mostrarán las recomendaciones no personalizadas.

S-2: Geo localizada

Actores participantes: REJA y usuario.

Condición de entrada: El usuario tiene una recomendación.

Flujos de eventos:

1		El sistema muestra los restaurantes en el mapa.
2		
	a	Si desea visualizar la recomendación en una lista la aplicación realiza **S-1**
	b	Si desea realizar una recomendación basada en el contexto, se realiza **S-3**

Excepciones:

E.1	Se informara de cualquier posible error al realizar la conexión con REJA.
E.2	En caso de no tener recomendaciones personalizadas, se mostrarán las recomendaciones no personalizadas.

S-3: Contexto

Actores participantes:		Usuario y REJA.
Condición de entrada:		El usuario ha accedido al sistema, y tiene el GPS activo.
Flujos de eventos:		
1		El usuario solicita una recomendación basada en su contexto
2		El sistema
	a	Si la distancia es el máximo posible, la recomendación será geo localizada y se realizara **S-1**
3		La aplicación envía a REJA la petición
4		REJA devuelve la recomendación
5		La aplicación muestra los restaurantes.
6		
	a	Si desea visualizar la recomendación en una lista la aplicación realiza **S-1**
	b	Si desea realizar una recomendación geo localizada, se realiza **S-2**
Excepciones:		
	E.1	Se informara de cualquier posible error al realizar la conexión con REJA.
	E.2	Si el GPS no está activo se realizará **S-1**
	E.3	En caso de no tener recomendaciones personalizadas, se mostrarán las recomendaciones no personalizadas.

4.3.5. Caso de uso 3: Interacción con los restaurantes.

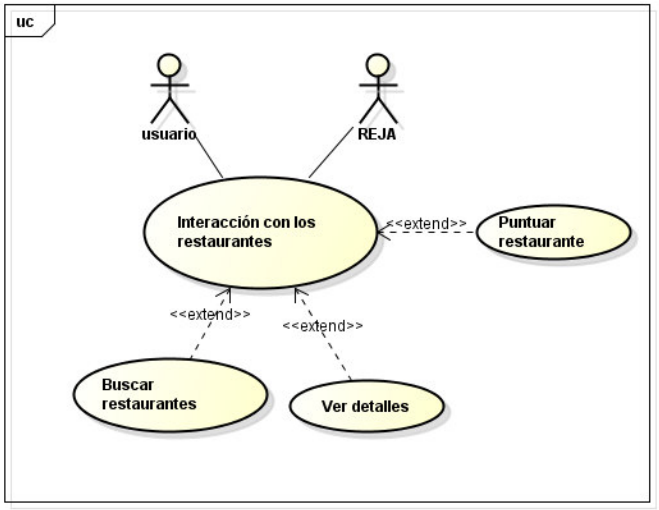


Figura 4.3 Caso de uso Interacción con los restaurantes.

Actores participantes:	Usuario y REJA.
Condición de entrada:	Tener acceso al sistema, y al menos un restaurante sobre el que realizar la consulta
Flujos de eventos:	
1	Si el usuario desea obtener más información de un restaurante. Se realiza **S-1.**
2	Si el usuario desea buscar un algún restaurante, introduciendo una cadena de texto se realiza **S-2**.
3	Si el usuario desea valorar un restaurante, se realiza **S-3**
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

4.3.5.1. Sublujo de eventos:

S-1: Ver detalles

Actores participantes:	Usuario, REJA.
Condición de entrada:	Tener un restaurante sobre que poder visualizar.
Flujos de eventos:	
1	La aplicación realiza la consulta de la información a la base de datos de REJA.
2	REJA manda la Información del restaurante
3	La aplicación muestra en una pantalla con la información del Restaurante (dirección, número, rating)
4	Si el usuario lo desea valorar el restaurante, se realiza **S-3**.
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

S-2: Buscar restaurantes.

Actores participantes:	Usuario, REJA.
Condición de entrada:	Tener acceso al sistema.
Flujos de eventos:	
1	El usuario introduce el nombre del restaurante, o una cadena de texto parecida.
2	La aplicación manda a REJA el texto.
2	REJA devuelve una lista de posibles candidatos.
3	La aplicación muestra la lista de candidatos en forma de lista.
4	Si el usuario desea ver los detalles, se realiza el caso de uso 4, **Ver detalles.**
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

S-3: Puntuar restaurante.

Actores participantes:	Usuario, REJA.
Condición de entrada:	Tener acceso al sistema y estar visualizando los detalles del restaurante.
Flujos de eventos:	
1	El usuario introduce la puntuación sobre el restaurante.
2	La aplicación manda a REJA la valoración.
2	REJA almacena la valoración.
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

4.3.7. Caso de uso 4: Recomendación a Grupos

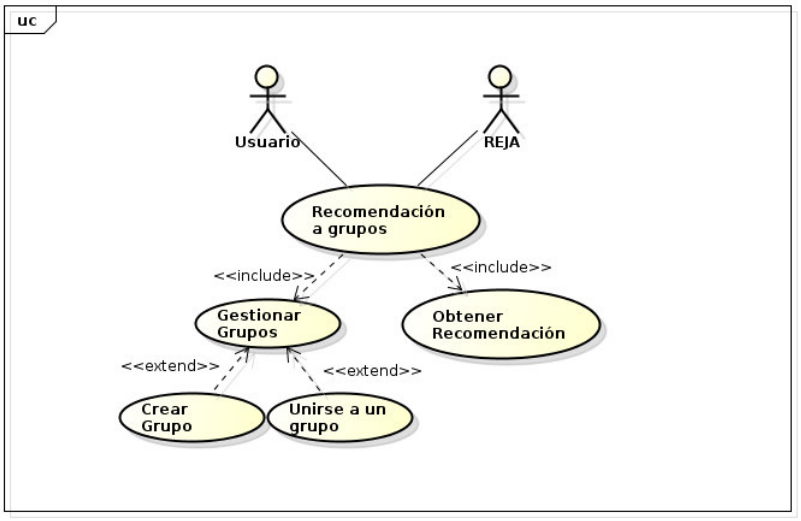


Figura 4.4 caso de uso Recomendación a Grupos

Actores participantes:	Usuario y REJA.	
Condición de entrada:	Tener acceso al sistema y pertenecer de un grupo	
Flujos de eventos:		
1		
	A	Si el usuario desea crear el grupo se realiza **S-1**
	B	Si el usuario desea unirse al grupo se realiza **S-2**
2		El usuario ya pertenece al grupo.
3		El administrador del grupo manda la petición a REJA, el contexto del grupo, será el del administrador del grupo.
4		Reja devuelve las recomendaciones en base a los gustos del grupo, y el contexto del administrador del grupo.

5		El administrador manda las recomendaciones al resto de usuarios del grupo.
6		Los usuarios visualizan las recomendaciones.
	a	Pueden visualizar la recomendación en formato lista
	b	Pueden ver la recomendación geo localizada.

Excepciones:

E.1	Se informará de cualquier posible error al realizar la conexión con REJA.
E.2	Se informará de cualquier posible error de comunicación entre los dispositivos
E.3	En caso de no tener recomendaciones personalizadas, se mostrarán las recomendaciones no personalizadas.

4.3.7.1. Subflujo de evento:

S-1: Crear grupo

Actores participantes: Usuario y REJA.

Condición de entrada: Tener acceso al sistema.

Flujos de eventos:

1		El usuario crea el grupo
2		Los usuarios se añaden al grupo.
3		El usuario gestiona las altas al grupo
	a.	Puede aceptarlos.
	b.	Rechazarlos
4		Una vez aceptados todos los usuarios, selecciona la distancia del radio de búsqueda.
5		Se vuelve al punto 3 del flujo de eventos principal

Excepciones:

E.1	Se informara de cualquier posible error al realizar la conexión con el servidor.

S-2: Unirse a un grupo

Actores participantes: Usuario y REJA.

Condición de entrada: Tener acceso al sistema y que haya grupos a los que unirse.

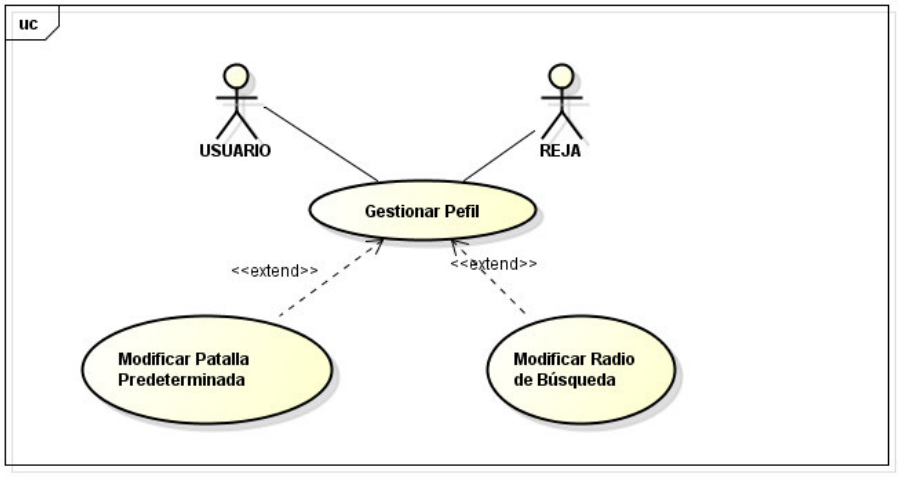
Flujos de eventos:

1	Solicita unirse al grupo
2	Manda al grupo su identificador personal.
3	Espera la confirmación de su solicitud
4	Una vez aceptado espera las recomendaciones al grupo.

Excepciones:

E.1	Se informara de cualquier posible error al unirse al grupo.

4.3.8. Caso de uso 5: Configuración



v

Figura 4.5 Caso de Uso configuración

Actores participantes: Usuario y REJA.

Condición de entrada: Tener acceso al sistema y estar identificado.

Flujos de eventos:

1	Si el usuario quiere, puede cambiar la pantalla de inicio predeterminada entre los diferentes tipos de recomendación.
2	Si el usuario quiere, puede modificar la distancia de búsqueda establecida por defecto para las recomendaciones basadas en contexto.

Excepciones:

E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

4.3.9. Caso de uso 6: Gestionar Perfil.

Actores participantes:	Usuario y REJA.
Condición de entrada:	Tener acceso al sistema y estar identificado.
Flujos de eventos:	
1	La aplicación solicita a REJA una lista con las valoraciones del usuario.
2	REJA devuelve una lista con las valoraciones del usuario.
3	La aplicación muestra al usuario la lista de valoraciones realizada
4	El usuario selecciona en la lista el restaurante que quiere volver a valorar.
5	El usuario introduce la nueva valoración
6	La aplicación manda a REJA la nueva valoración
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

4.3.10. Caso de uso 7: Desvincular Cuenta

Actores participantes:	Usuario y REJA.
Condición de entrada:	Tener acceso al sistema y estar identificado.
Flujos de eventos:	
1	El usuario solicita a la aplicación desvincular la cuenta de la aplicación
2	La aplicación tramita la petición y elimina toda la información de la cuenta de usuario en la aplicación.
Excepciones:	
E.1	Se informara de cualquier posible error al realizar la conexión con REJA.

4.4. Escenarios

Son representaciones ficticias que describen una posible interacción con una interfaz, con el fin de que varios de estos escenarios supongan la representación de un caso de uso.

Está formado por los siguientes elementos:

- Nombre único y unívoco.
- Una descripción
- los actores participantes
- el flujo de eventos.

Estas historias ficticias son de gran utilidad para los diseñadores, al describir una interacción con la interfaz, permiten anticiparse a los problemas. Dichas historias a pesar de ser ficticias deben desarrollarse con el mayor nivel de detalles posible, así los diseñadores les resulta más fácil debatir que discutir sobre una situación abstracta.

Escenario 1: Acceso de usuario al sistema.

Nombre:	Acceso del usuario Jorge
Descripción:	Jorge inicia sesión en el sistema identificándose para acceder a las funcionalidades
Actores participantes:	Jorge, REJA
Flujo de eventos:	
1.	Jorge inicia la aplicación
2.	La aplicación muestra el botón para iniciar sesión en Google Plus
3.	Google Plus despliega una pantalla para ceder los datos personales, ya que es la primera vez que accede.
4.	Jorge acepta ceder los datos
5.	Google plus cede los datos
6.	La aplicación manda el identificador único y personal del usuario a REJA.
7.	REJA comprueba el identificador del usuario.
8.	Como el identificador no existe, lo registra automáticamente.
9,	Antes acceder al sistema, este le pedirá al usuario que active el GPS, para las recomendaciones basadas en contexto y a grupos

10.	Jorge accede al sistema y se le despliega la pantalla de recomendación.

Escenario 2: Recibir recomendación.

Nombre:	Recibir recomendación geo localizada y contextualizada.
Descripción:	Mario ha iniciado sesión en el sistema y desea recibir una recomendación
Actores participantes:	Mario, REJA
Flujo de eventos:	
1.	Mario pulsa sobre el botón de recomendación contextualizada.
2.	Mario introduce un radio de búsqueda de 100M.
3.	La aplicación realiza la petición a REJA
4.	REJA devuelve la lista de recomendaciones (“La vestida”, “Casa Pepe”, “La Plaza”).
5.	La aplicación le muestra las recomendaciones en una lista.
6.	Mario quiere ver la recomendación geo localizada a Mario.
7,	Mario pulsa el botón de geo localizar.
8.	La aplicación muestra la recomendación geo localizada.
9.	Si Mario quiere volver a la lista se realiza el punto 3.

Escenario 3: Interacción con un Restaurante.

Nombre:	Ver detalles del restaurante “La Vestida”
Descripción:	Andrés ha iniciado sesión en el sistema y ha recibido una lista de recomendación, en la cual elige “La Vestida” para ver sus detalles
Actores participantes:	Andrés, REJA
Flujo de eventos:	
1.	Andrés accede a la aplicación
2.	La aplicación realiza la petición al servidor.
3.	REJA devuelve la lista de recomendaciones (“La vestida”, “Casa Pepe”, “La Plaza”).
4.	El sistema le muestra las recomendaciones a Andrés
5.	Andrés quiere ver los detalles de “La Vestida”
6.	En la lista de recomendaciones selecciona “La Vestida”
7.	La aplicación realiza la petición a REJA de la información del restaurante.
8.	REJA devuelve la información a la aplicación.
9.	La aplicación muestra a Andrés los detalles de “La Vestida” (dirección, número, rating)

10.	Si Andrés quiere realizar una reserva este podrá llamar al restaurante pulsando sobre el número teléfono.
11.	Si Andrés lo desea, puede valorar el restaurante.
A.	La aplicación manda la nota del Andrés al servidor.

Escenario 4: Recibir una recomendación Grupal.

Nombre:	Recomendación Grupal.
Descripción:	Juan, Andrés y Jorge han quedado para cenar y quieren recibir una recomendación en base a los gustos de los tres.
Actores participantes:	Juan, Andrés, Jorge, REJA
Flujo de eventos:	
1.	Juan, Andrés y Jorge acceden a la aplicación
2.	Los tres seleccionan recomendación a grupos, en sus aplicaciones
3.	Los tres acuerdan que Juan, administre el grupo.
4.	Juan Introduce el nombre del grupo y pulsa en crear grupo.
5.	Andrés y Jorge, seleccionan unirse a grupo,
6.	Buscan el grupo de Juan, y solicitan unirse a él.
7.	Juan los acepta la solicitud para unirse al grupo de Jorge y Andrés
8.	Juan solicita la recomendación.
9.	La aplicación realiza la petición a REJA
10.	REJA devuelve una lista de restaurantes (“Casa Nati” y “El machito”).
11.	La aplicación muestra a los miembros del grupo la lista de recomendaciones.

Escenario 5: Gestionar Perfil.

Nombre:	Modificar puntación del bar “Solera”.
Descripción:	Luis ha estado en otros bares después del bar “Solera” y cree que merece una puntuación peor, por lo que quiere modificar la anterior puntuación del bar.
Actores participantes:	Luis, REJA
Flujo de eventos:	
1.	Luis accede a la aplicación.
2.	Accede a los “Ajustes” de la aplicación.
3.	Selecciona Mis puntuaciones.
4.	Busca el bar “Solera” en lista de valoraciones realizadas .
5.	Pulsa sobre él.

6.	Modifica la puntuación, a un 2
7.	Pulsa en “Puntuar”.
8.	La aplicación envía a REJA la puntuación
9.	REJA modifica en la base de datos la puntuación.

Escenario 6: Configuración.

Nombre:	Configuración de ajustes.
Descripción:	Pedro quiere cambiar al configuración de la aplicación para que le muestre siempre la pantalla de “Búsqueda” al iniciar la aplicación y establecer un radio de búsqueda para la recomendación contextualizada de 2 Km.
Actores participantes:	Pedro, REJA
Flujo de eventos:	
1.	Pedro accede a la aplicación.
2.	Accede a los “Ajustes” de la aplicación.
3.	En el apartado de “Pantalla Principal”, selecciona Buscar restaurante.
4.	La aplicación guarda el cambio realizado en la configuración de la pantalla inicial.
5.	En la pantalla de ajustes modifica el radio de búsqueda a 2 Km
6.	La aplicación guarda el cambio realizado.

Escenario 7: Desvincular Cuenta.

Nombre:	Desvincular cuenta de la aplicación
Descripción:	Pedro desvincular su cuenta de la aplicación
Actores participantes:	Pedro, REJA
Flujo de eventos:	
1.	Pedro accede a la aplicación.
2.	Accede a los “Ajustes” de la aplicación.
3.	Pulsa en el botón “Desvincular Cuenta”.
4.	La aplicación elimina cierra la sesión de usuario.
5.	La aplicación muestra la pantalla de “Login”.

4.5. Diseño del Sistema.

Llegamos a una nueva fase de la ingeniería del software, igual de importante que las demás. Aunque todas son igual de importante, en esta etapa hay que tener mayor cuidado, ya que un error por insignificante que sea, puede suponer pasar de desarrollar un software de calidad, aun producto plagado de errores.

En el diseño del sistema podemos diferenciar, dos claros diseño. El diseño de datos por un lado, se encargar de establecer las estructuras de datos a utilizar, la manera de trabajar con esos datos, y la comunicación de los datos entre el cliente y el servidor.

Por otro lado tenemos el diseño de la interfaz define cual va a ser el aspecto visual de la aplicación. A la hora de diseñar la interfaz hay que tener en cuenta un gran número de factores, ya que la interfaz es el punto en el cual usuario interactúa con el usuario.

4.5.1. Diseño de Datos.

En esta fase vamos a determinar la estructura de los elementos de información del sistema. Estos elementos son [41,42]:

- De los restaurantes sabemos que poseen un identificador, su nombre, la dirección postal, un número de teléfono, el tipo de cocina que ofrecen y si disponen de terraza o no.
- Los usuarios conocemos su identificador, nombre, contraseña y dirección.
- De las puntuaciones sabemos el identificador del usuario que la ha hecho, el identificador del restaurante que las ha recibido, y cuando se hizo.
- De los Grupos sabemos que tienen un identificador del grupo y la fecha cuando han sido creados.

Como ya se indicó anteriormente, el sistema tiene una arquitectura cliente-servidor. Esto se traduce en que los datos se almacenan en el servidor. Para

almacenar esos datos se utiliza una base de datos, de esto modo para diseñar dicha base de datos se ha utilizado el modelo Entidad-Relación.

4.5.1.1. Modelo Entidad-Relación

El modelo entidad relación, también conocido como E-R (de ahora en adelante usaremos la denominación E-R) se componen de tres elementos:

- Entidades: Representan objetos, en nuestro caso, tendremos 3 entidades que representaran cada una a los usuarios a los restaurantes y las valoraciones. Cualquier elemento concreto de una entidad se denomina instancia. Se representan con un rectángulo y el nombre en el interior (véase figura 15).

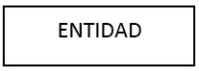


Figura 4.6 Entidad en modelo E-R

- Relaciones: Dependencias entre una o más entidades. Se representa con un rombo y en su interior el nombre de la relación. Si la relación de una entidad es consigo misma se denomina reflexiva (véase figura 16).

Figura 4.7 Relación en modelo E-R

- Atributos: Son características de una entidad o relación. Se representan con elipses etiquetadas con un nombre en su interior (véase figura 17).

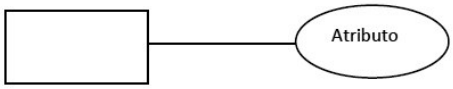


Figura 4.8 Atributo en modelo E-R

Estos son los 3 pilares fundamentales del modelo E-R. Ahora vamos a indicar otros aspectos a tener en cuenta:

- Claves: Cada entidad debe tener, puede estar formada por uno o más atributos, y se representa con el nombre del atributo subrayado, y representa unívocamente a una instancia.

- Relaciones uno a uno: una instancia de una entidad se relaciona solo con una instancia de otra entidad unívocamente. Se representa con dos rectángulos concéntricos y el nombre de la entidad en el interior del más pequeño.
- Se representa 1:1
- Relaciones uno a muchos: una instancia de una entidad se relaciona con varias instancias de otra entidad. Se representa 1:*
- Relaciones muchos a muchos: Cualquier instancia de una entidad se relaciona con varias instancias de otra entidad. Se representa **:**
- Entidades débiles: Estas entidades necesitan estar relacionadas con otras entidades para existir, por lo que estas entidades débiles no se pueden identificar

Ahora ya sabemos los elementos que forman parte de un diagrama E-R, ahora vamos a enumerar los pasos para desarrollar un diagrama E-R:

1. Convertir el enunciado del problema en un Esquema Conceptual (o EC) del mismo.
2. Convertir este EC en uno más refinado llamado Esquema Conceptual Modificado (ECM). En él se eliminan las relaciones muchos a muchos y las entidades débiles.
3. Obtener las tablas de la base de datos a partir del ECM.

4.5.1.2. Normalización en el modelo Entidad-Relación.

Con el fin de evitar la redundancia de datos, proteger la integridad de los datos o disminuir problemas de actualización de datos en las tablas, se aplican una serie de reglas o restricciones, llamadas Formas Normales [41,42].

Estas formas normales son:

- Primera Forma Normal (1FN): Una tabla está en 1FN si para un valor de la clave no hay grupos repetidos.

- Segunda Forma Normal (2FN): Una tabla está en 2FN si está en 1FN y además todos los atributos depende funcionalmente de ella, es decir, cada columna de la tabla debe depender de toda la clave.
- Tercera Forma Normal (3FN): Una tabla está en 3FN si está en 2FN y además no existe ninguna dependencia transitiva entre los atributos que no son de la clave.

Dado que las tablas y las relaciones que tenemos son bastantes sencillas, una vez que realicemos el Esquema Conceptual Modificado obtendremos las tablas de la base de datos de la aplicación.

Antes de pasar a ver el esquema conceptual de la bases de datos vamos a justificar el motivo por el cual, la entidad “Grupo” va a tener una marca temporal como atributo.

El fin de almacenar una marca temporal como atributo de un grupo, es para facilitar el mantenimiento de los grupos, ya que los grupos tienen un carácter temporal. Esta marca temporal facilita la eliminación de aquellos grupos que lleven tiempo inactivos o haya pasado un tiempo prudencial desde su creación. Así utilizando las características del gestor de base de datos que se utilice en la implementación de este TFG se puede realizar esta tarea de manera cíclica sin la intervención del administrador de la base de datos una vez este programado o implementado el evento.

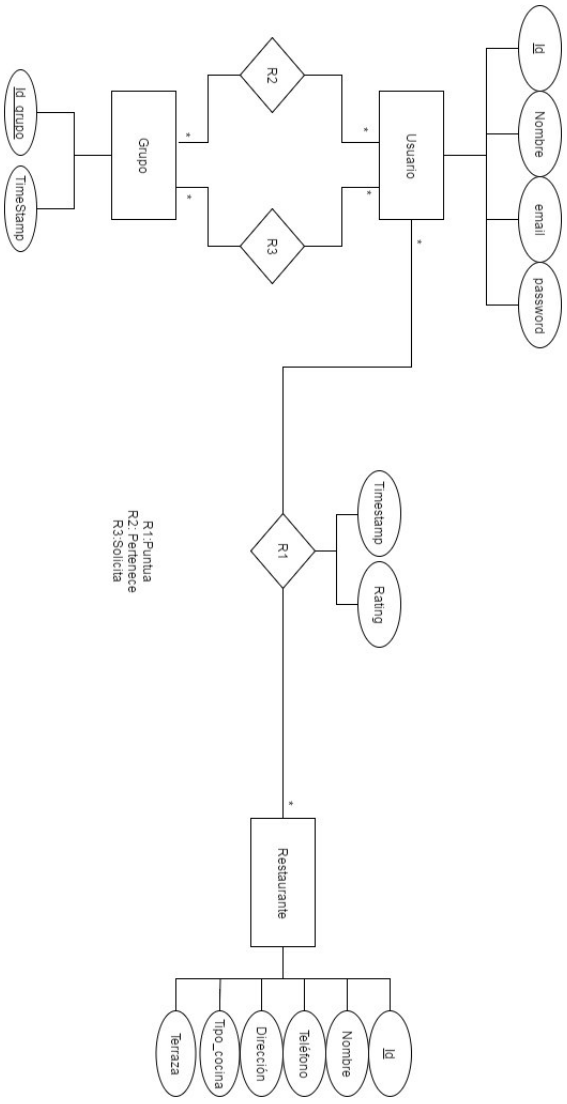


Figura 4.9 Esquema Conceptual de la aplicación

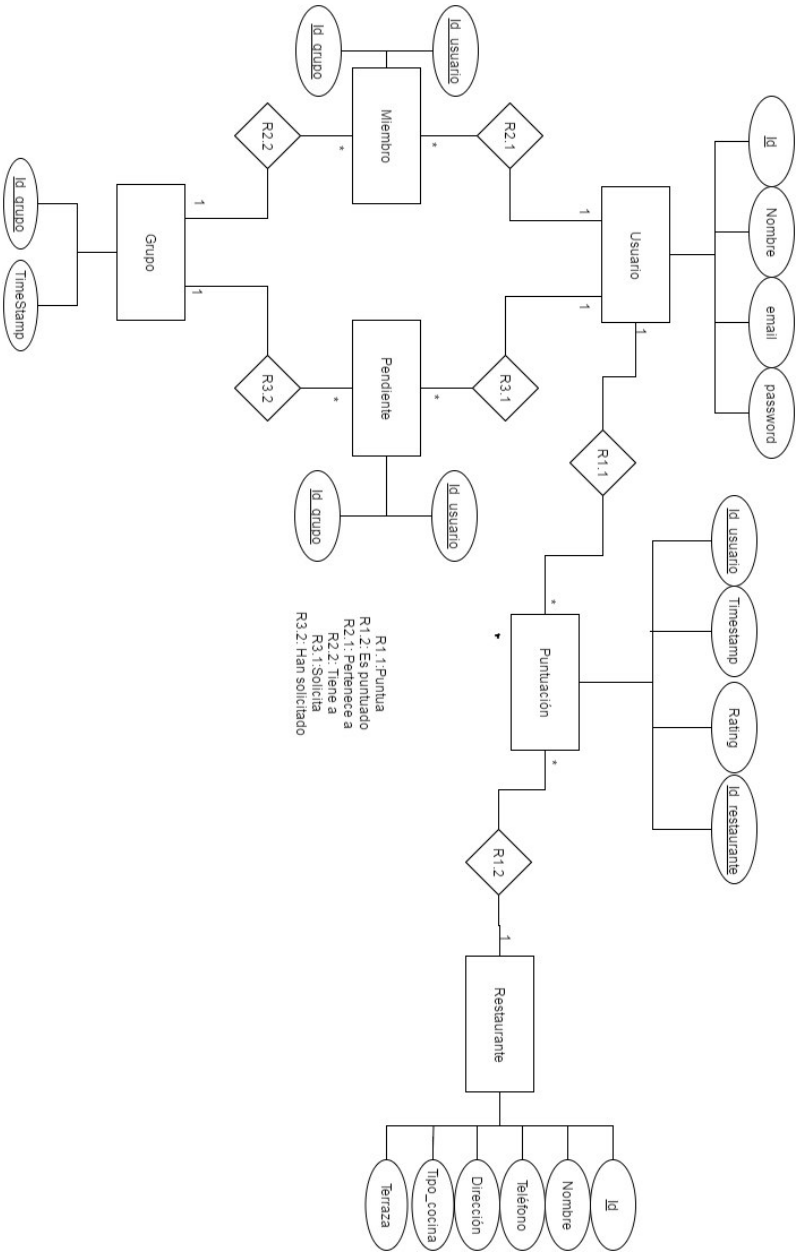


Figura 4.10 Esquema Conceptual Modificado de la aplicación

Las tablas de la base de datos son las siguientes:

Usuario

- Id: Entero, clave primaria de la tabla.
- Username: cadena de texto. Único, nombre de usuario para cada cuenta, será único para cada usuario.
- Email: Cadena de texto. Almacena el correo electrónico del usuario.
- Pass: Cadena de texto, será almacenada mediante la función hash MD5.

Restaurantes.

- Id: Entero, clave primaria de la tabla
- Nombre: Cadena de texto
- Teléfono: Entero, ¿Debería ser único?
- Dirección: Cadena de texto
- Tipo_cocina: (Cada tipo de cocina es una columna con valor 1 o 0).
- Terraza: Entero, tomar los valores 1 y 0. 0 indica que no tiene terraza, 1 que si

Puntuación

- Id_usuario: Entero. Clave primaria. Clave externa, identificador del usuario
- Id_item: Entero. Clave primaria. Clave externa, identificador del restaurante.
- Rating: flotante. Comprendido entre 0.0 y 5.0. Puntuación que le da un usuario concreto a un restaurante determinado.
- Timestamp: Fecha. Fecha en la que se realizó la valoración.

(Las tablas en joomla que tienen este cometido son jos1524_users, jos1524_item. Los ratings en guardan en la base de datos sad_reja en la tabla ratings)

Grupo

- Id_grupo: Entero. Clave primaria. Clave externa, identificar del grupo.

- Timestamp: Fecha, Fecha en la que se creó el grupo.

Miembro y Pendiente

- Id_grupo: Entero. Clave primaria. Clave externa, identificar del grupo.
- Id_usuario: Entero. Clave primaria. Clave externa, identificador del usuario

4.5.2. Diseño de Interfaz

Una vez realizado el diseño de los datos y definido la forma de comunicación entre el cliente y el servidor llega el momento de diseñar la interfaz. Está interfaz será la apariencia visual de la aplicación. Pero a la hora de diseñar una interfaz, hay que tener en cuenta la usabilidad.

El termino usabilidad (proviene del inglés usability) y se refiere a la facilidad con la que una persona puede usar o aprender a usar una herramienta fabricada por un ser humano. En interacción persona-ordenador se refiere al grado en el cual un usuario específico puede lograr un objetivo específico en un tiempo determinado.

Para poder obtener un bueno grado de usabilidad en nuestra interfaz, vamos a utilizar metáforas, prototipos de las pantallas y caminos de navegación para lograr un bueno diseño de nuestra interfaz.

4.5.2.1. Metáforas

Al igual que tradicionalmente usamos las metáforas para referirnos a conceptos abstractos en el uso del lenguaje, en informática usamos las metáforas para referirnos a diversas tareas que la interfaz permite desarrollar.

Actualmente tienen un papel dominante en el diseño de interfaces debido a que permiten a los desarrolladores construir programas que pueden ser usados por comunidades de usuarios más diversos, al poder comunicar conceptos abstractos de una forma familiar y accesible[43,44].

En nuestra aplicación vamos a utilizar las siguientes metáforas:

Restaurante.

Esta metáfora se visualizará cuando la recomendación se esté mostrando de manera geo localizada. Indica la situación de un restaurante, en el mapa.

Figura 4.11 Icono para metáfora "Restaurante"

Usuario.

Esta metáfora se visualizará cuando la recomendación contextualizada se esté mostrando de manera geo localizada. Indica la situación del usuario, en el mapa.

Figura 4.12 Icono para metáfora "Usuario"

Distancia de búsqueda

Debido a que tenemos una recomendación basada en el contexto del usuario, se hace indispensable que el usuario pueda indicar a la aplicación, el radio de búsqueda que desea para la recomendación, de esa necesidad, surge el uso de esta metáfora, la cual indica en el extremo izquierdo unos pocos de metros, y en el derecho los kilómetros máximos de búsqueda.

Figura 4.13 Icono para metáfora "Distancia de Búsqueda"

Llamar

Esta metáfora aparece en la pantalla de ver detalles, y nos facilita la posibilidad de llamar al restaurante, desde la misma aplicación.

Figura 4.15 Icono para metáfora "Llamar"

Puntuación

Puesto que los restaurantes tienen una puntuación y se pueden puntuar, dicha puntuación se ejemplificara con un grupo de 5 estrellas, si dicha puntuación es un 2.5 solo habrá dos estrellas y media de color naranja, y el resto de color gris.

Figura 4.16 Icono para metáfora "Puntuación"

Unirse al grupo.

Tal como representa esta metáfora, la acción sobre ella, supone la adición sobre otra cosa, en concreto a un grupo de recomendación

Figura 4.17 Icono para metáfora "Unirse al grupo"

Eliminar del grupo.

Esta metáfora le aparece al administrador del grupo de recomendación, cuando un usuario se une al grupo, al pulsar sobre esta metáfora, quedará automática eliminada del grupo el usuario correspondiente.

Figura 4.18 Icono para metáfora "Eliminar del grupo"

4.5.2.2. Prototipos de interfaz

Ya hemos definido las metáfora que se van a utilizar y su significado dentro de la aplicación. En este apartado vamos a presentar los prototipos de las pantallas de navegación que tendrá nuestra aplicación.

Dichos prototipos son un esbozo de lo que será el sistema para el usuario final, por lo que no representan un diseño final.

Pantalla de Login

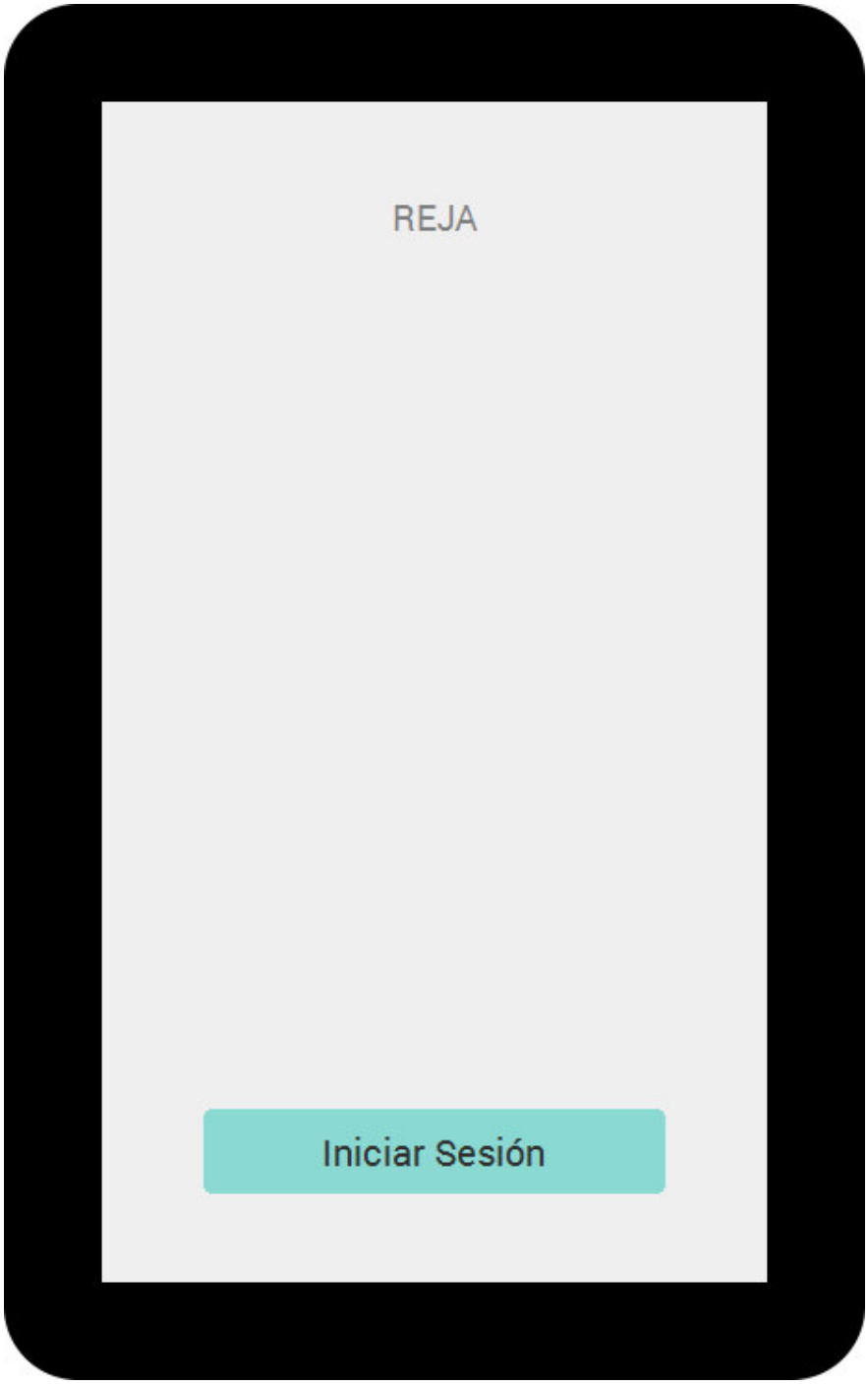


Figura 4.19 Pantalla “Login”

Pantalla recomendación: modo lista

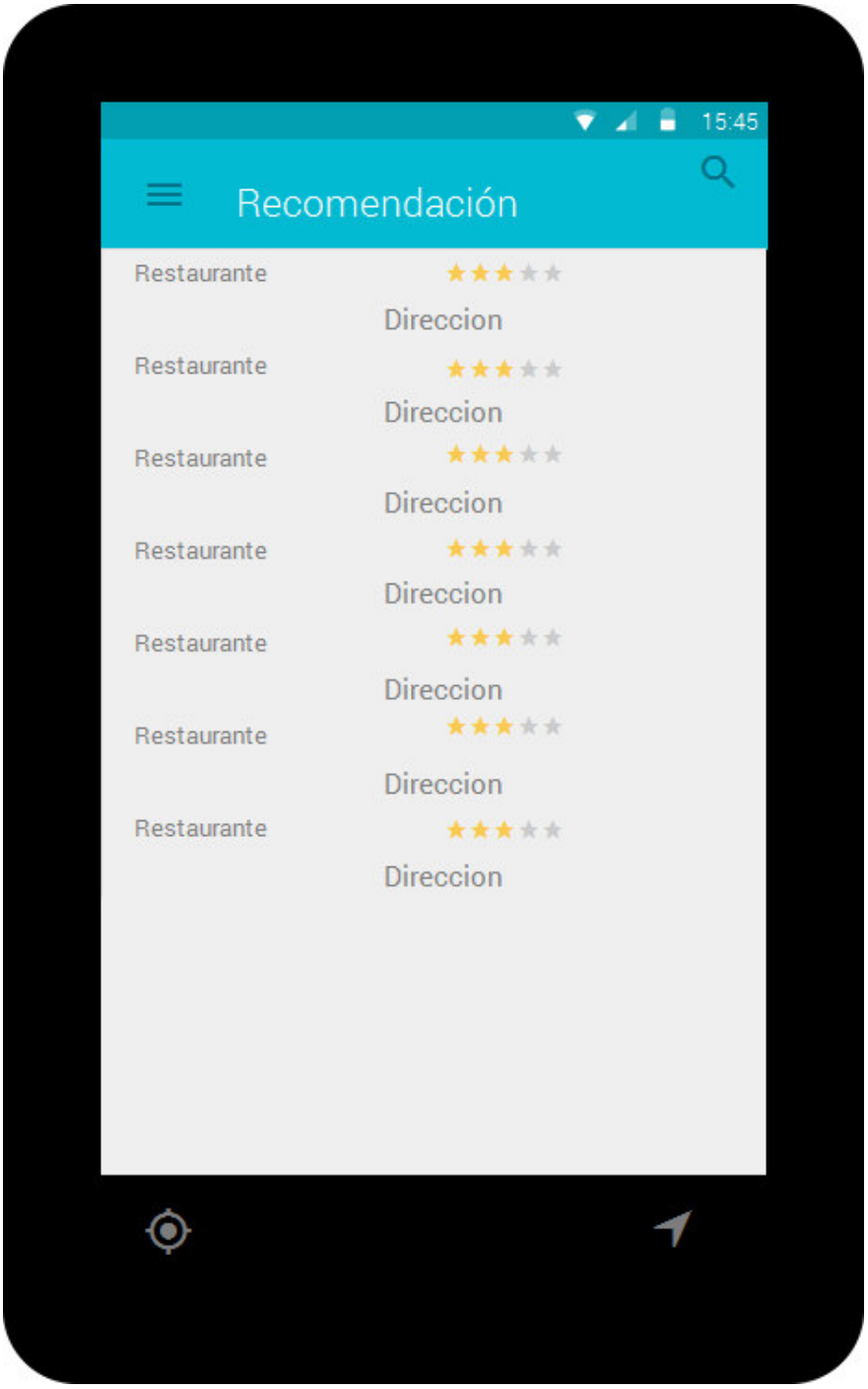


Figura 4.20 Pantalla "Recomendación modo lista"

Pantalla recomendación: modo geo localizado

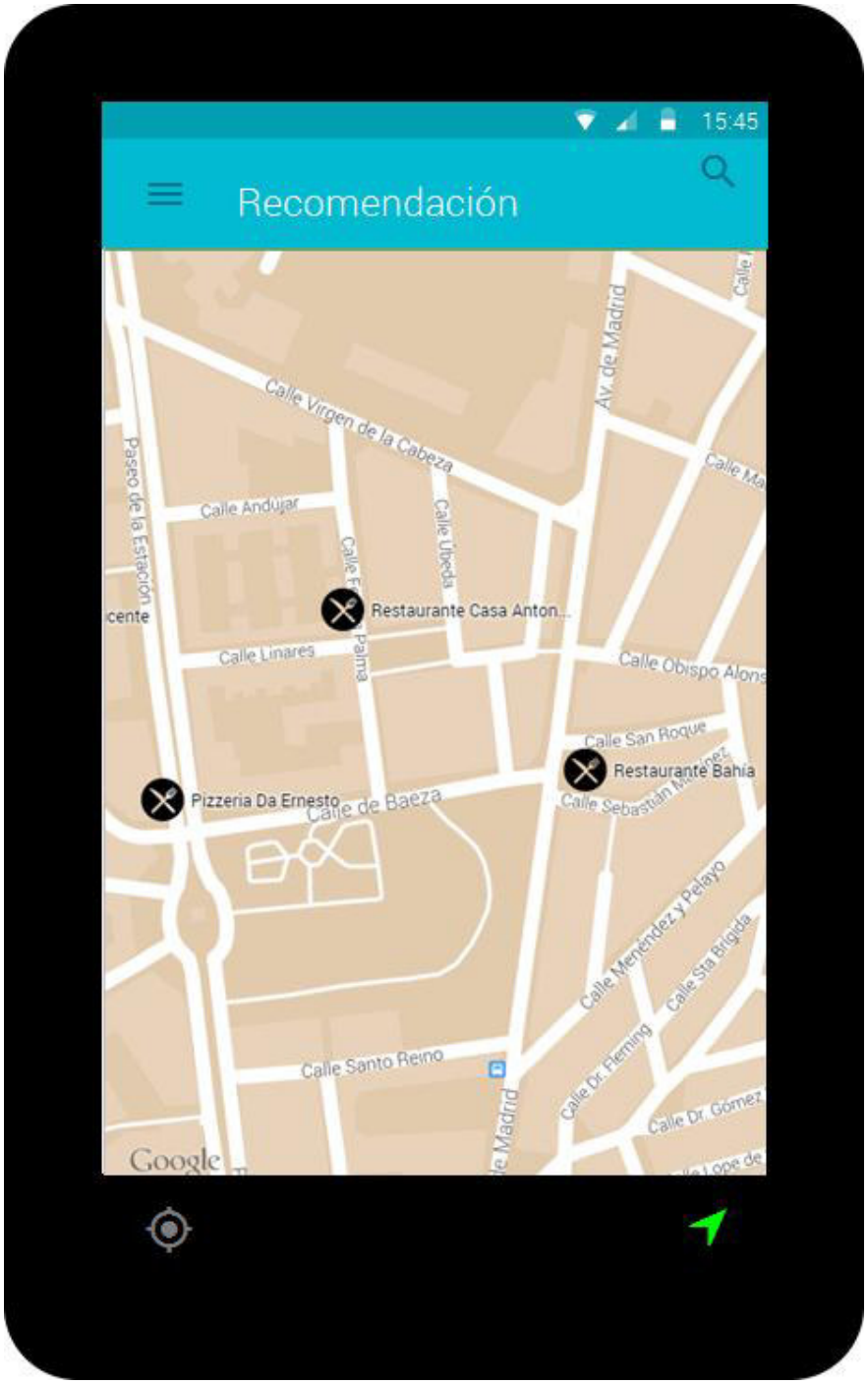


Figura 4.21 Pantalla "Recomendación geo localizada"

Pantalla recomendación contextualizada: modo lista

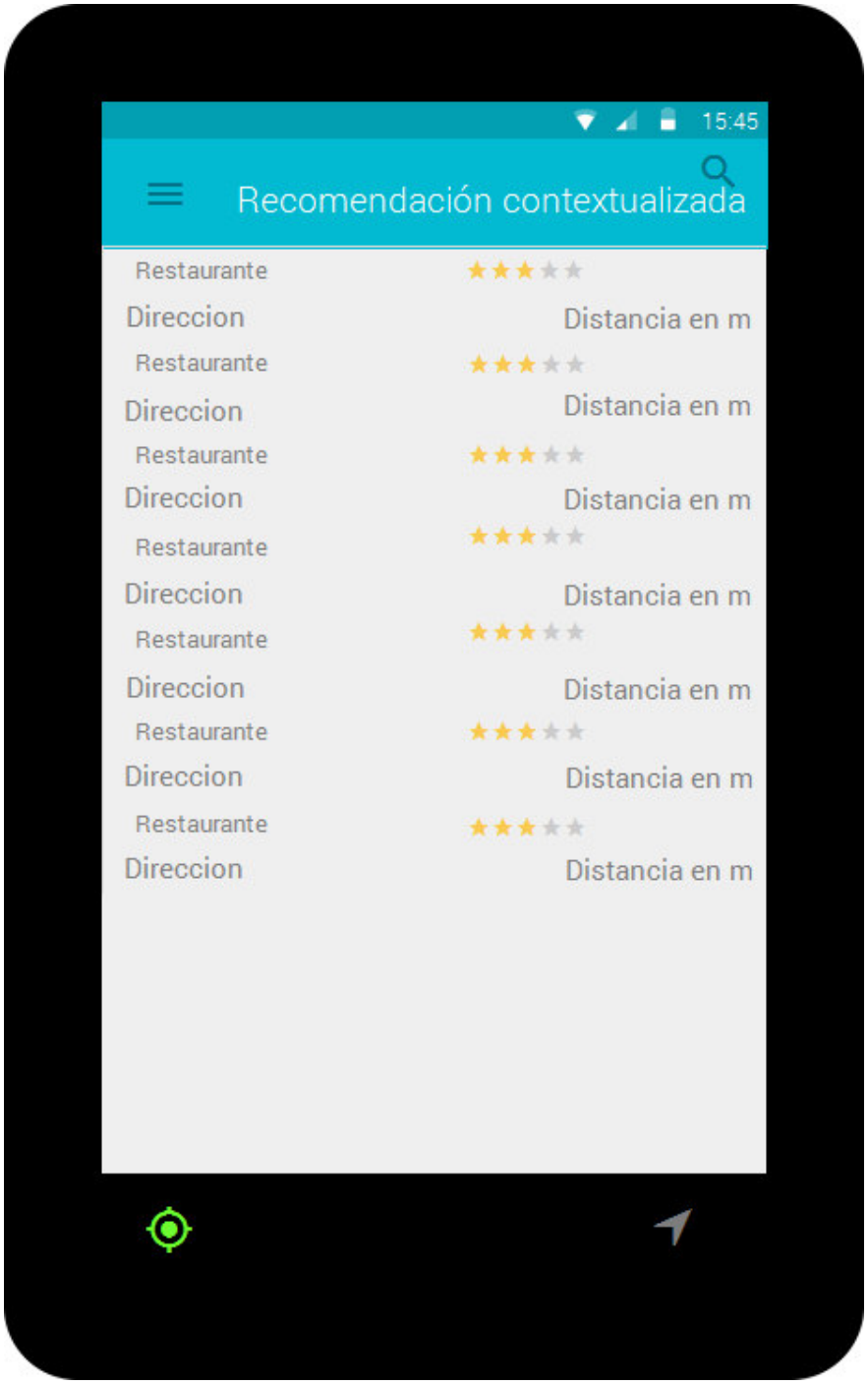


Figura 4.22 Pantalla "Recomendación Contextualizada modo lista"

Pantalla recomendación contextualizada: modo geo localizado

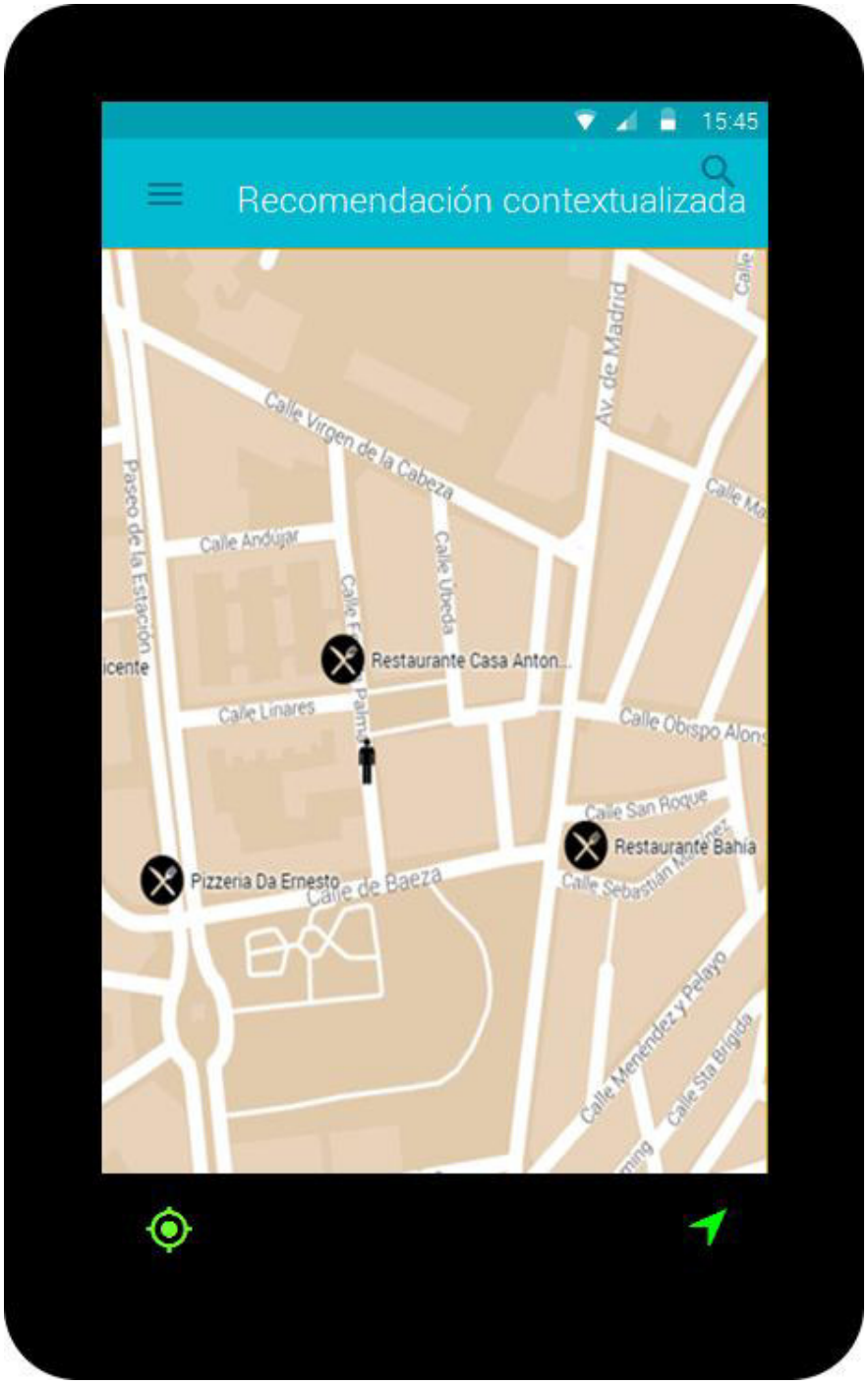


Figura 4.23 Pantalla "Recomendación Contextualizada geo localizada"

Pantalla recomendación a grupos: Crear grupo

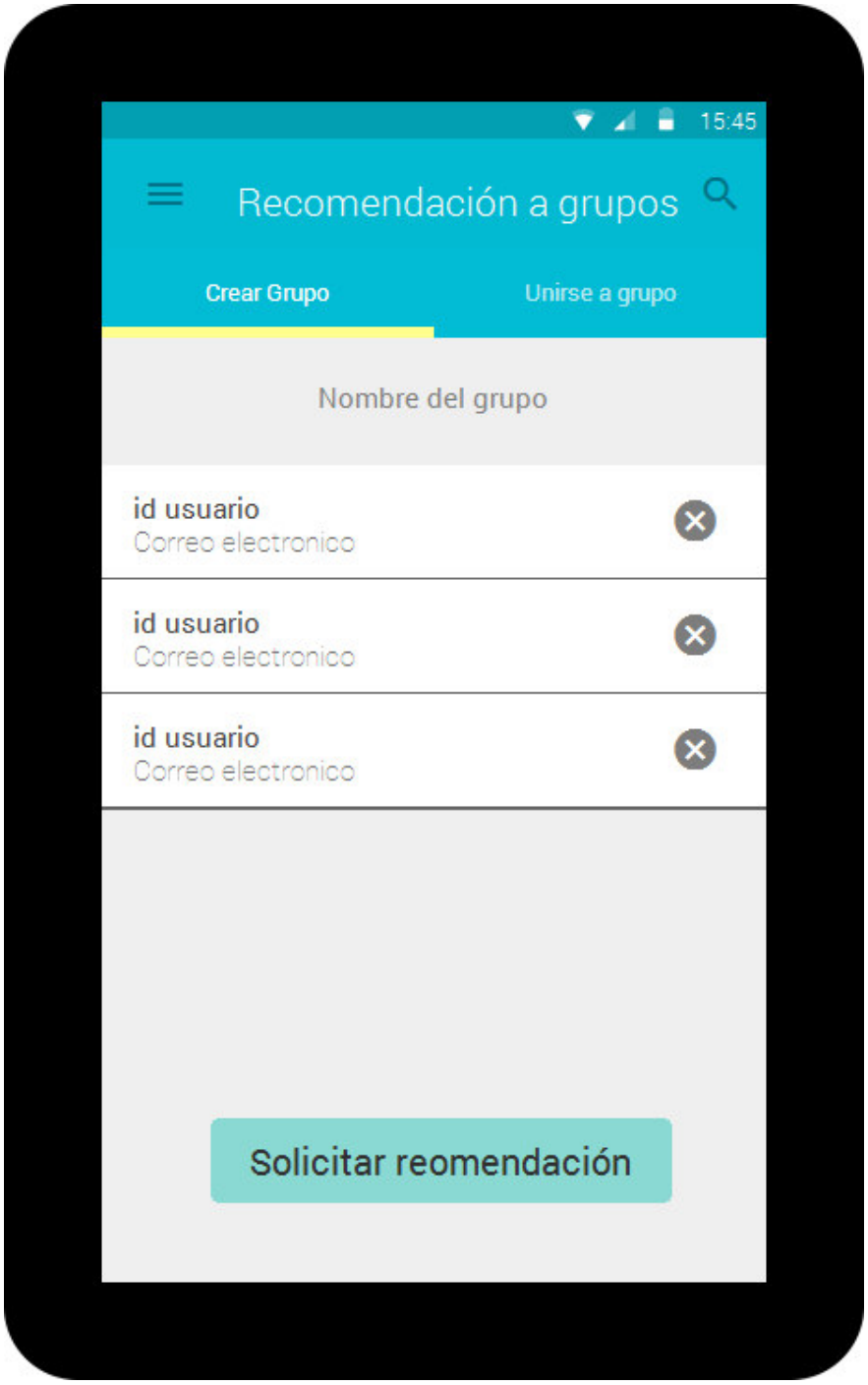


Figura 4.24 Pantalla "Crear grupo"

Pantalla recomendación a grupos: Unirse a grupo

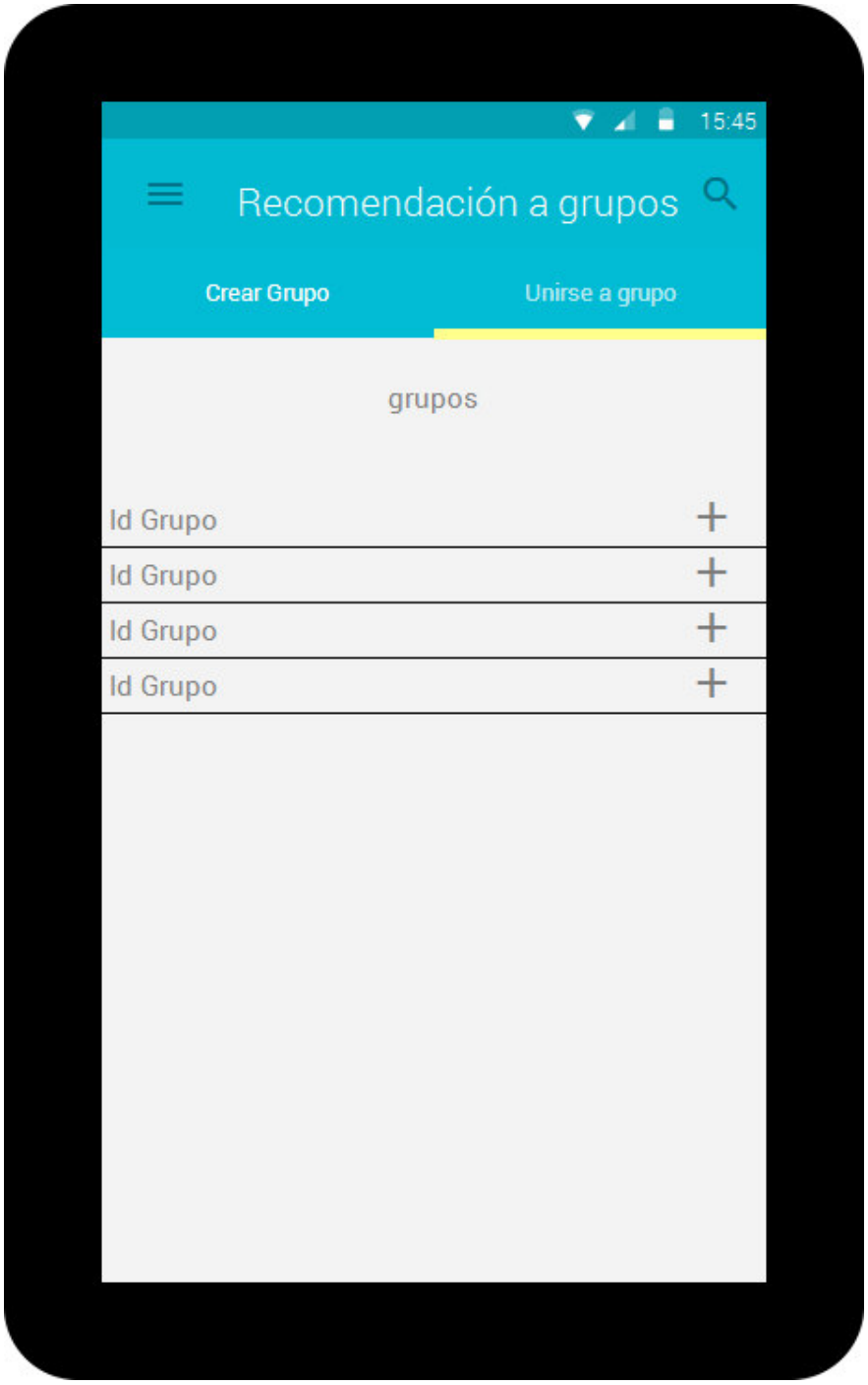


Figura 4.25 Pantalla "Unirse a un grupo"

Pantalla detalles de un restaurante.

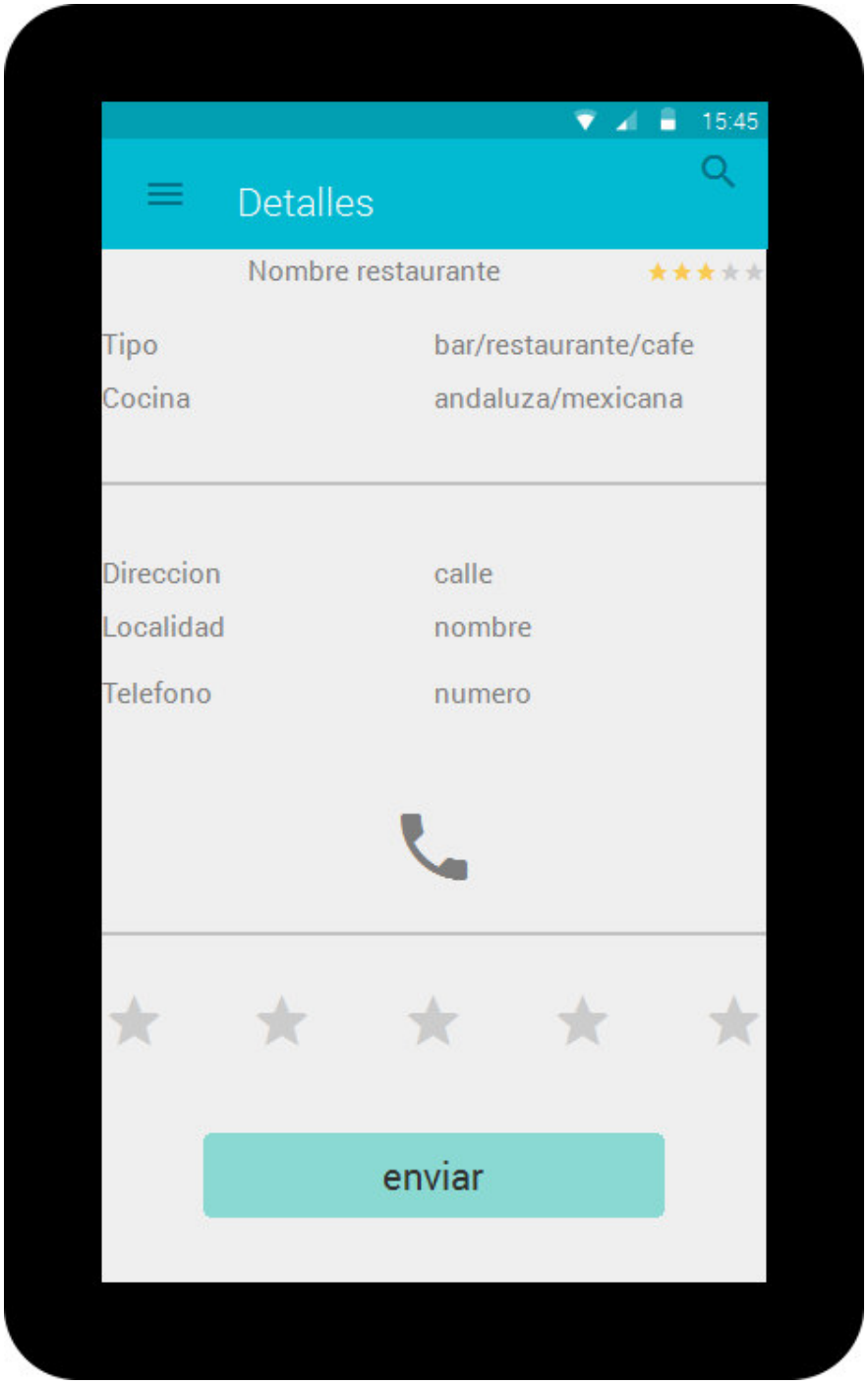


Figura 4.26 Pantalla "Detalles de un restaurante"

Pantalla gestionar Perfil

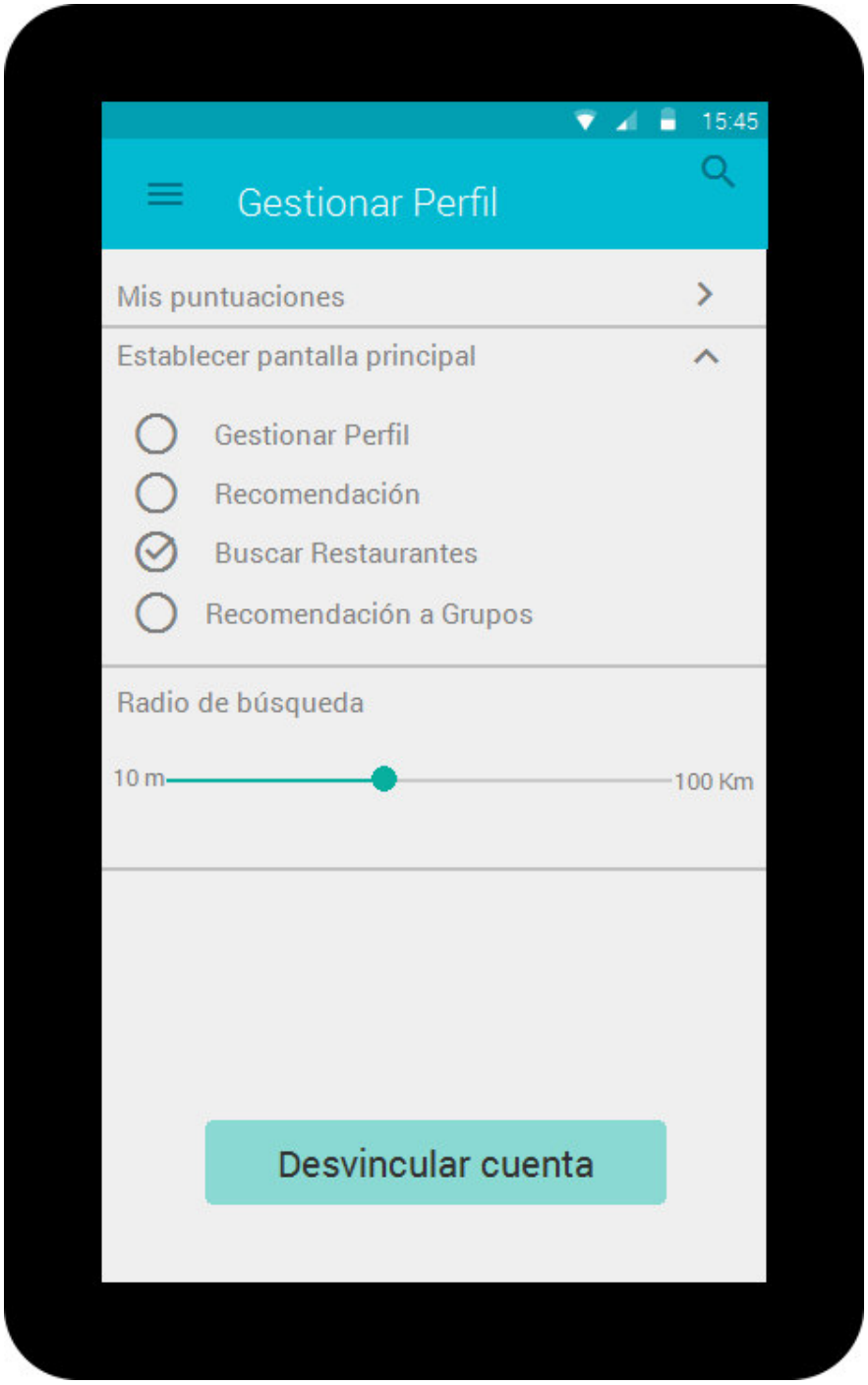


Figura 4.27 Pantalla "Gestionar perfil"

4.5.2.3. Caminos de Navegación

Hasta ahora todo el diseño de la interfaz ha representado una visión estática de la misma, al diseñar individualmente cada pantalla. Pero la interacción del usuario con la interfaz va a transcurrir de manera más fluida, y tendrá varios caminos de navegación.

Para estudiar la navegación del usuario, se va a utilizar una herramienta llamada Storyboards. Al igual que los Escenarios son una realización ficticia de un caso de uso, los Storyboards narran de manera secuencial, las pantallas que el usuario va visualizando hasta llegar a una acción concreta.

Este prototipo en papel permite que el usuario evalúe y valide la interfaz en fases tempranas con el fin de que las correcciones, en caso de haberlas, no sean costosas. Ya que el coste de estas correcciones aumenta cuanto más tarde se hagan.

La manera de proceder, para el desarrollo será la siguiente: las pantallas de la interfaz se situaran de manera secuencial, según el orden en el que le aparezcan al usuario.

A continuación se muestran los storyboards para algunas de las acciones que se pueden realizar en la aplicación:

- Storyboard inicio de sesión
- Storyboard geo localización de recomendación
- Storyboard búsqueda de restaurante.
- Storyboard unirse a un grupo

Storyboard para inicio de sesión.

Para iniciar sesión del usuario (véase figura 4.28):

1. El usuario pulsa el botón “iniciar sesión”
2. Google muestra un mensaje para seleccionar la cuenta.
3. El usuario selecciona su cuenta.
4. Inicia la pantalla de recomendación.

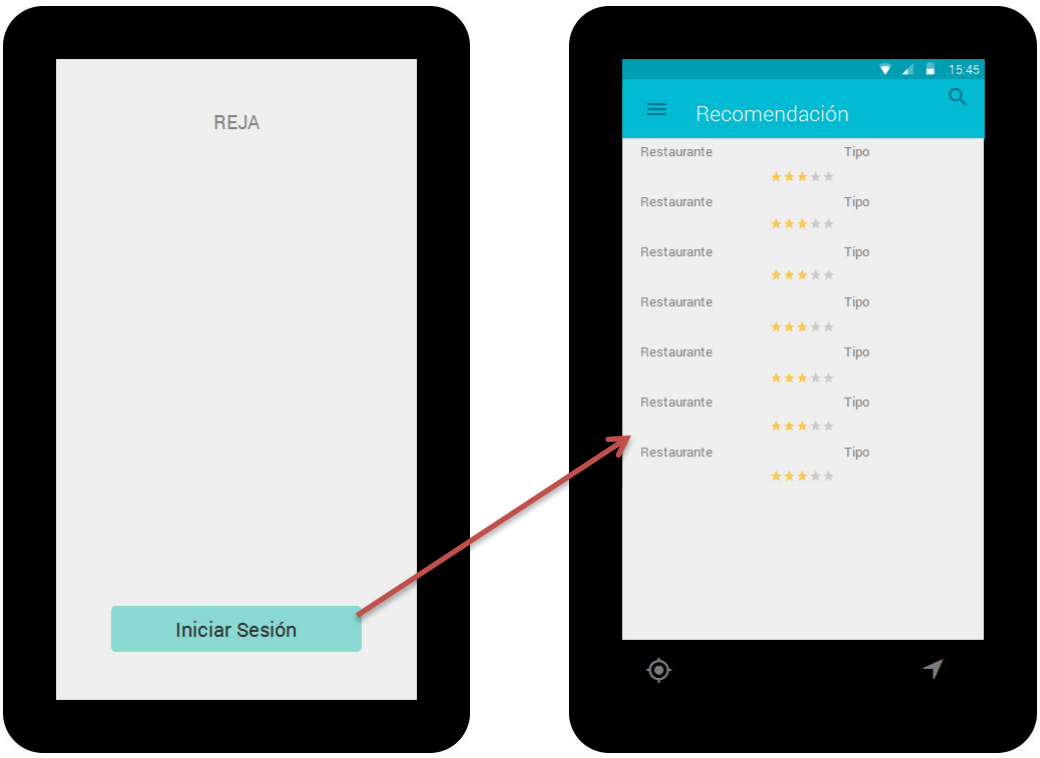


Figura 4.28 Storyboard inicio de sesión

Storyboard para geo localizar una recomendación

Para geo localizar la recomendación (véase figura 4.29):

1. El usuario pulsa el botón “geo localizar”.
2. Se inicia la pantalla de Google Maps para visualizar los restaurantes.

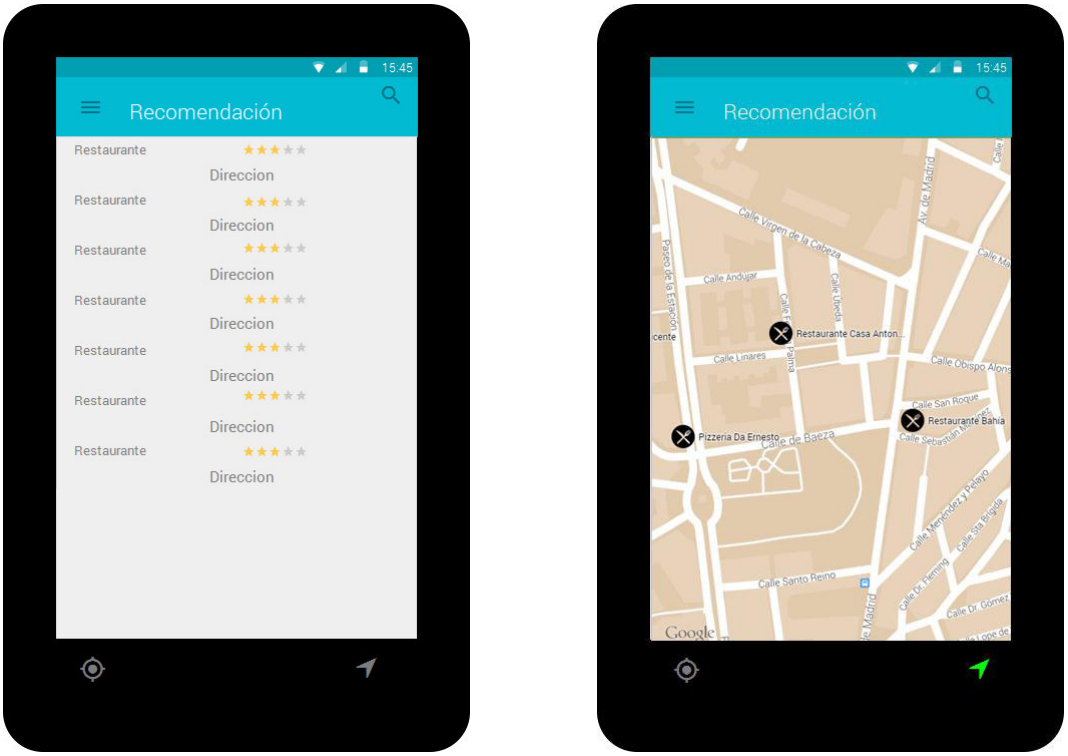


Figura 4.29 Storyboard geo localizar recomendación

Storyboard para buscar restaurante.

Para la búsqueda de un restaurante (véase figura 4.30):

1. El usuario despliega el menú lateral.
2. Pulsa el botón del menú “Buscar Restaurante”.
3. Introduce el nombre del restaurante en la caja de texto.
4. Pulsa el botón aceptar
5. Se envía el nombre
6. El sistema devuelve el nombre de los restaurantes similares

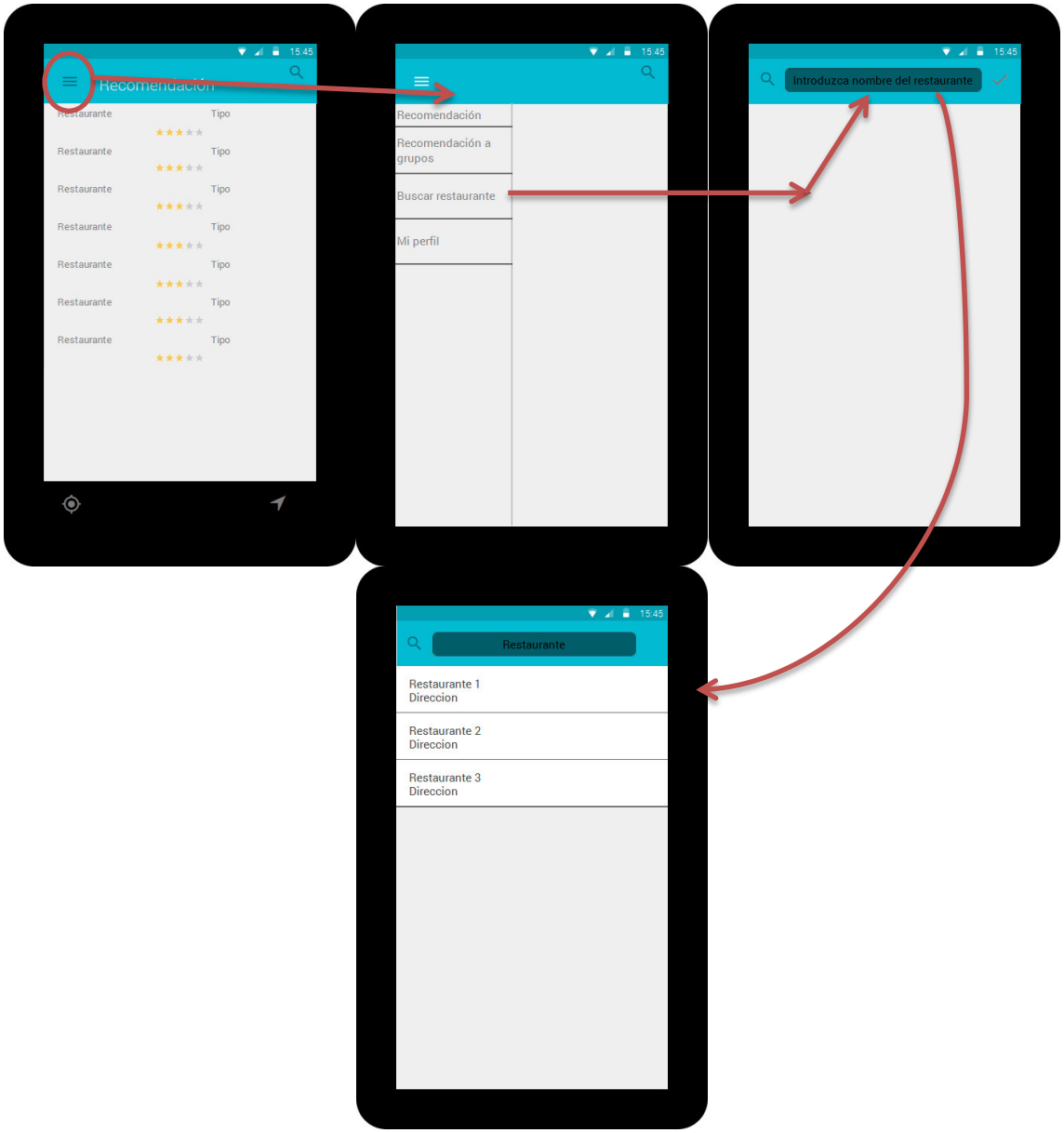


Figura 4.30 Storyboard buscar restaurante

StoryBoards unirse a un grupo.

Para unirse a un grupo (véase figura 4.31):

1. El usuario despliega el menú lateral.
2. Selecciona la opción del menú “Recomendación a grupos”.
3. Se inicia la pantalla recomendación a grupos.
4. Pulsa en la pestaña Unirse a grupo.
5. Inserta en el campo de texto el nombre del grupo.
6. Acepta el nombre del grupo a buscar.
7. Aparecen las opciones de grupos para unirse.
8. Pulsa en el icono “+” para unirse al grupo que desee.

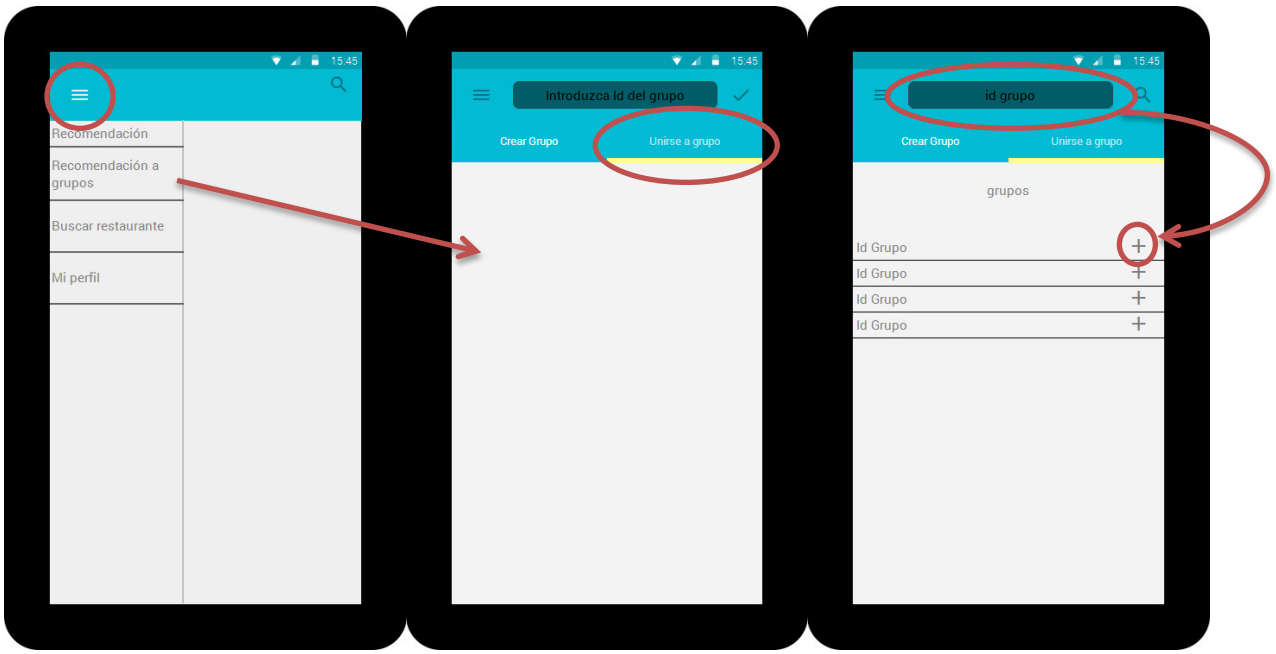


Figura 4.31 Storyboard unirse a un grupo

4.6. Implementación

Hemos llegado a la fase final de la ingeniería del software. Este es el momento en el que todo el proceso anterior realizado de ingeniería del software se transforma en código.

Durante este apartado, elegiremos el lenguaje de programación correspondiente a cada parte de nuestro sistema. El sistema tendrá una arquitectura cliente/servidor.

El TFG se sustenta en los siguientes elementos:

- *Servicio Web*: estable la comunicación entre el cliente (aplicación en Android) y el servidor.
- *Aplicación Android*: Será la interfaz entre el servidor y el usuario. Permitirá al usuario recibir recomendaciones, evaluar bares y restaurantes, buscar restaurantes y llamar a estos restaurantes.
- *Algoritmo de filtrado colaborativo*: Se encarga de calcular las recomendaciones a partir de las puntuaciones de otros usuarios similares.
- *Base de datos*: Contiene todos los datos relativos a los usuarios, los restaurantes, puntuaciones, predicciones, etcétera,

En los apartados anteriores hemos visto que estructura que tendrán los datos para ser almacenados en el servidor, pero nos quedan aún por resolver de qué manera accederá el cliente a esos datos. Tal como se describió en capítulos anteriores, será a través de una Servicio Web, esto es así para dotar una mayor independencia a los datos de la interfaz de usuario.

Esta independencia entre datos se puede ver como una Interfaz Programación de Aplicaciones (Application Programming Interfaz, API).

Una API es un mecanismo comunicación entre componentes software, proporcionando un conjunto de funciones para acceder a los servicios obviando la complejidad de un sistema o aplicación. Esto permite por ejemplo, que el equipo de

desarrollo de interfaz, pueda implementar y probar la interfaz en fases tempranas, independientemente de los datos, solo respetando esas “entradas” y “salidas” proporcionadas por la API.

La justificación del uso de esta tecnología reside en varios aspectos, el primero de ellos su abstracción ya que permite realizar llamadas e interactuar con el sistema de recomendación teniendo con conocimiento mínimo sobre estos sistemas. Otro aspecto fundamental es la reutilización de código que está probado y funciona correctamente en otro software a través de las funciones que ofrece.

A continuación podemos ver la tabla 4.1 en la que se definen las llamadas al servicio, de la siguiente manera:

- *URI*: Identificador Recursos Uniforme, el nombre de la función.
- *Método*: Método HTTP utilizado para la llamada al servicio.
- *Paramentos*: Nombre del parámetro a necesario para la llamada.
- *Definición*: Breve descripción del funcionamiento de la llamada.

URI	Método	Parámetros	definición
/login	Post	email	Identifica al usuario dentro del sistema, si no existe crea una cuenta de usuario.
/recommendations/:id	Get	Id:	Solicita la recomendación del usuario al que pertenece el id_user
/details	Post	IdItem,idUser	Solicita toda la información referente al restaurante al que pertenece el identificador del restaurante con la puntuación dada por el usuario.
/context	Post	IdUser,my_lat,my_long,maxDist	Solicita al sistema la recomendación del usuario con dicho id_user, pero solo devolverá los restaurantes dentro del radio de búsqueda desde la posición del usuario.
/rating	Post	IdUuser,idIitem, rating	Inserta o modifica el rating del restaurante con dicho id_item dado por el usuario con dicho id_user
/search	post	restaurant:String	Solicita los restaurantes cuyos nombres son similares a la cadena introducida .
/addGroup	Post	adminId, groupId	Crea un grupo
/accept	Post	groupId, userId	Acepta un usuario en el grupo

/deny	Post	groupId, userId	Elimina un usuario del grupo
/:groupId/members	Post	userId,groupId	Solicita los miembros en el grupo.
/searchGroup	Post	textGroup	Busca los grupos a partir de un texto
Join	Post	groupId,userId	Solicita unirse a un grupo
/groupRecommendation	Post	groupId,adminId	Solicita la recomendación a gusto de todos los usuarios del grupo .
/contextGroup	Post	IdUser,my_lat,my_long,maxDist, groupId,adminId	Solicita la recomendación contextualizada para un grupo de usuarios.

Tabla 4.1 URIs del Servicio Web

Cada recurso devolverá el código de estado que corresponda, según como haya transcurrido la ejecución de la solicitud realizada.

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

Tabla 4.2 Estados del Servicio Web

4.6.1. Lenguaje de programación en el servidor.

Para llevar a cabo esta API, vamos a implementar un servicio REST. Esta implementación se hará con el lenguaje de programación PHP (Hypertext Preprocessor) es un lenguaje de programación de código abierto.

PHP se interpreta en el lado del servidor, y permite incrustar código en páginas HTML con el objetivo de dotarlas de contenido dinámico.

A continuación se enumeran las ventajas de PHP para ser escogido como lenguaje de programación en este TFG:

- Destaca la capacidad de conexión con la mayoría de manejadores de bases de datos, especialmente MySQL que ha sido el utilizado en este TFG.
- Gran volumen de documentación.
- No es necesario declarar variables.
- Permite el uso de Programación Orientada a Objetos.
- Gran modularidad y capacidad de añadir funcionalidades a su gran número de módulos.

También destacar el uso de Slim Framework para la creación de la API. Este micro framework de PHP facilita enormemente crear aplicaciones webs. Se ha escogido slim microframework:

- Instalación
- No es necesario guardar contexto de la sesión del usuario en el servidor.
- Documentación.

Algunas de las características de Slim son:

- Compatible con todos los métodos HTTP.
- Permite depuración de error y tratamiento de excepciones.
- Escritura de logs.
- Parámetros en ruta, con comodines y condiciones.

4.6.2. Lenguaje de programación en el cliente.

Debido a que el cliente es una aplicación en Android, los lenguajes de programación son Java y XML.

- Java: Es un lenguaje orientado a objetos, su potencial radica en que es un lenguaje interpretado y no compilado, esto quiere decir que, que una aplicación Java puede ejecutarse en distintas plataformas indistintamente del hardware a través de una máquina virtual de Java.

  Java es el lenguaje utilizado en Android para definir la lógica de la aplicación.

- XML: Es un lenguaje de marcas para documentos, que permite almacenar documentos de manera estructurada. Android lo utiliza para definir la interfaz de usuario.

4.7. Pruebas y Validación.

En esta fase se van a realizar las pruebas de caja negra para garantizar la calidad del software que hemos desarrollado.

El propósito de las pruebas de caja negra es encontrar un conjunto de entradas que satisfagan completamente los requisitos funcionales. En este tipo de test se ignora la estructura de control.

Para la realización de este test, vamos a enumerar unos casos de test que se corresponderán con los requisitos funcionales definidos en el apartado 4.2.

4.7.1. Casos de Prueba

Los casos de prueba diseñados serán los siguientes:

Test 1: Login correcto del usuario. [Requisito Funcional 1]

Pre-condiciones:	El usuario debe tener Cuenta en Google+
Acción :	El usuario selecciona su cuenta de Google+
Checkpoint 1:	La aplicación valida los datos y muestra la pantalla de inicio

Test 2: Obtener localización del usuario [Requisito Funcional 2]

Pre-condiciones:	El usuario debe tener Cuenta en Google+
Acción	El activa el GPS del Smartphone
Checkpoint 1:	La aplicación obtiene la latitud y longitud del dispositivo.

Test 3: Recomendar al usuario una lista de restaurantes [Requisito Funcional 3]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de recomendación
Checkpoint 1:	La aplicación muestra la recomendación

Test 4: Recomendar al usuario una lista de restaurantes basada en la posición del usuario. [Requisito Funcional 3]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario activa la contextualización de la recomendación
Checkpoint 1:	La aplicación filtra los restaurantes de la recomendación para mostrar los que están cercanos al usuario.

Test 5: Visualización de la recomendación en una lista [Requisito Funcional 4]

Precondiciones :	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de recomendación y elige visualizar la recomendación en una lista
Checkpoint 1:	La aplicación muestra la recomendación en una lista.

Test 6: Visualización de la recomendación en el mapa [Requisito Funcional 4]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de recomendación y elige visualizar la lista de recomendación Geo localizada.
Checkpoint 1:	La aplicación muestra la recomendación Geo localizada en el mapa

Test 7: Visualización de información de los restaurantes [Requisito Funcional 5]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de recomendación y elige visualizar la recomendación Geo localizada.
Checkpoint 1:	La aplicación muestra la recomendación Geo localizada en el mapa

Test 8: Valorar restaurante [Requisito Funcional 6]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación
Acción :	El usuario selecciona el número de estrellas para valorar el restaurante.
Checkpoint 1:	La aplicación manda al servidor la valoración y responde con un mensaje de confirmación.

Test 9: Recomendación a grupos. Crear un grupo. [Requisito Funcional 7]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación
Acción :	El usuario accede a la pantalla “Crear Grupo”, e introduce el nombre del grupo
Checkpoint 1:	La aplicación muestra el mensaje recibo del servidor.

Test 10: Recomendación a grupos. Unirse a un grupo. [Requisito Funcional 7]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla “Unirse a Grupo”, e introduce el nombre del grupo y solicita unirse al grupo deseado
Checkpoint 1:	La aplicación muestra en una lista los grupos candidatos.
Checkpoint 2:	La aplicación muestra al usuario que su solicitud está pendiente de aprobación por el administrador del grupo.

Test 11: Buscar Restaurante [Requisito Funcional 8]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de “Búsqueda” e introduce una cadena de texto similar al nombre del restaurante que busca.
Checkpoint 1:	La aplicación muestra todos los restaurantes que contienen la cadena de texto introducida por el usuario.

Test 12: Configuración. Pantalla inicial. [Requisito Funcional 9]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de “Ajustes” y selecciona la pantalla deseada como pantalla inicial.
Checkpoint 1:	La aplicación guarda las preferencias de pantalla de inicio

Test 13: Configuración. Radio recomendación contextualizada. [Requisito Funcional 9]

Pre-condiciones:	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de “Ajustes” modifica la distancia filtrado para recomendación contextualizada deslizando el slider
Checkpoint 1:	La aplicación guarda las preferencias de radio de búsqueda.

Test 14: Modificar Puntuaciones realizadas [Requisito Funcional 10]

Precondiciones :	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de “Ajustes” y selecciona la opción “Mis puntuaciones” y selecciona el restaurante que quiere volver a valorar.
Checkpoint 1:	La aplicación manda al servidor la nueva valoración del usuario

Test 15: Desvincular cuenta del dispositivo [Requisito Funcional 11]

Precondiciones :	El usuario debe tiene que estar logueado en la aplicación con su cuenta de Google +
Acción :	El usuario accede a la pantalla de “Ajustes” y selecciona la opción desvincular cuenta.
Checkpoint 1:	La aplicación elimina las preferencias de usuario y cierra la sesión del usuario en la aplicación.

4.7.2. Resultados Obtenidos.

Test 1	
Checkpoint 1	Ok
Test 2	
Checkpoint 1	Ok
Test 3	
Checkpoint 1	Ok
Test 4	
Checkpoint 1	Ok
Test 5	
Checkpoint 1	Ok
Test 6	
Checkpoint 1	Ok
Test 7	
Checkpoint 1	Ok
Test 8	
Checkpoint 1	Ok
Test 9	
Checkpoint 1	Ok
Test 10	
Checkpoint 1	Ok
Checkpoint 2	Ok
Test 11	
Checkpoint 1	Ok
Test 12	
Checkpoint 1	Ok
Test 13	
Checkpoint 1	Ok
Test 14	
Checkpoint 1	Ok
Test 15	
Checkpoint 1	Ok

4.8. Despliegue y Mantenimiento.

Dado que la aplicación desarrollada en este TFG es un prototipo, no se desplegará de manera comercial, así el manteamiento consistirá en sucesivas iteraciones de mejora y desarrollo del prototipo con fines de investigación

CAPÍTULO 5 Conclusiones

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

En muchas de las operaciones que los usuarios realizan con frecuencia en Internet, se hace presente el problema de la sobrecarga de información, esto ocasiona que el usuario que hace uso de los servicios que se ofrecen en Internet pueda llegar a sentirse agobiado y confundido ante una gran cantidad de información difícil de asimilar, e incluso dejar usar dichos servicios.

Para solucionar este problema, aparecieron los Sistemas de Recomendación, que ayudan al usuario a encontrar aquella información de su interés mediante recomendaciones basadas en información proveniente del propio usuario, de otros usuarios o de expertos en la materia. Así, los usuarios reciben la información que realmente quieren recibir. Esto ha propiciado que los sistemas de recomendación hayan experimentado un gran auge y popularidad, que seguirá aumentado en los próximos años.

En este proyecto se ha desarrollado una aplicación Android. Para ello, la aplicación se ha desarrollado en torno a un Sistema de Recomendación Colaborativo, utilizando una arquitectura cliente/servidor.

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

Por último, añadir que para el autor del proyecto ha supuesto una enorme satisfacción el haber conseguido darle forma a una idea que, desde bastante tiempo atrás, habían plasmado tanto él como el tutor del trabajo fin de grado; y ver cómo

poco a poco, y a pesar de las dificultadas que inevitablemente surgen durante el desarrollo de cualquier aplicación. Además, este proyecto me ha aportado conocimientos, totalmente nuevos en el sistema operativo Android y profundizar en campos como el de los Sistemas de Recomendación, el diseño y administración de bases de datos y la propia Ingeniería del Software. Suponiendo un verdadero reto desde los primeros momentos de su realización, pero a la vez y sobretodo, ha sido una experiencia muy positiva y enriquecedora.

Anexo A Manual de Instalación

Antes de comenzar con el manual de instalación del sistema, vamos a enumerar el contenido del DVD de este TFG.

- PDF con la documentación del Trabajo Fin de Grado.
- Backup máquina virtual utilizada para el servidor.
- Backup máquina virtual utilizada para el cliente.
- Código de la parte del servidor (RejaAPI).
- Código de la parte del clienta (RejaApp)

Dado que el sistema está basado en una arquitectura cliente servidor, y para la realización de este proyecto se han implementado ambas partes, vamos a describir el proceso de instalación en ambas partes.

Servidor

Por simplicidad a la hora de usar el sistema completo de este TFG se facilita una máquina virtual en la cual está instalado el servidor con el siguiente software:

- PHP 5.4.4
- Apache 2.2.2
- Java 1.7
- phpMyAdmin

Para instalar la máquina virtual, vamos a “preferencias” y seleccionamos “importar servicio virtualizado” como podemos ver en la figura A.1

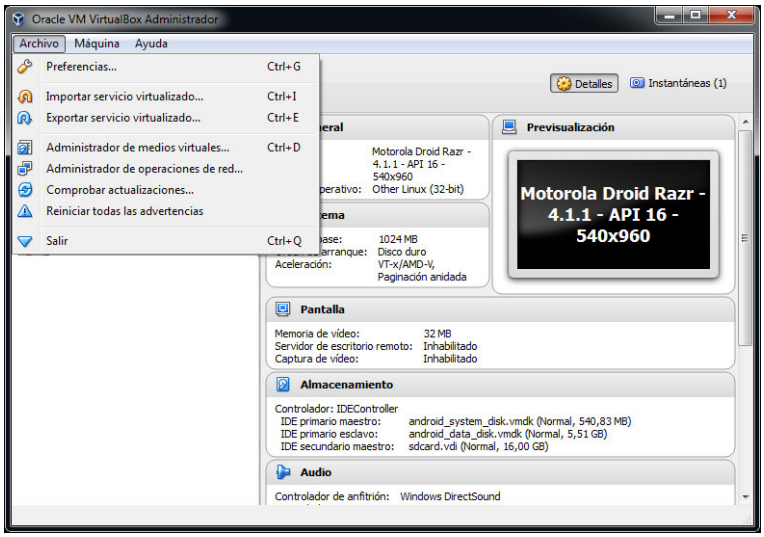


Figura A.1 Menú VirtualBox

A continuación se nos desplegará el asistente de importación, como vemos en la figura A.2

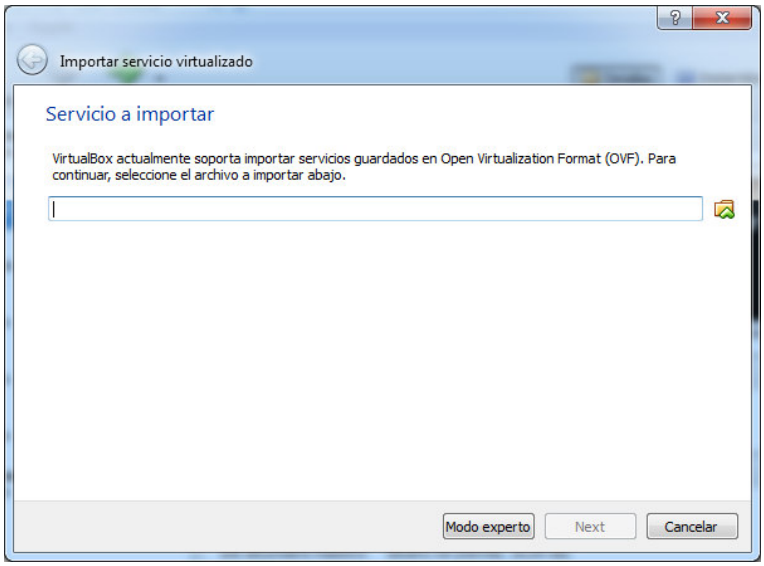


Figura A.2 Asistente de Importación

Seleccionamos el archivo “reja.ova” del DVD y pulsamos en “next”, a continuación se nos desplegará una ventana más del asistente en el que reconoce la máquina virtual y nos muestra su configuración por si deseamos modificar alguno antes de comenzar la importación (véase figura A.3)

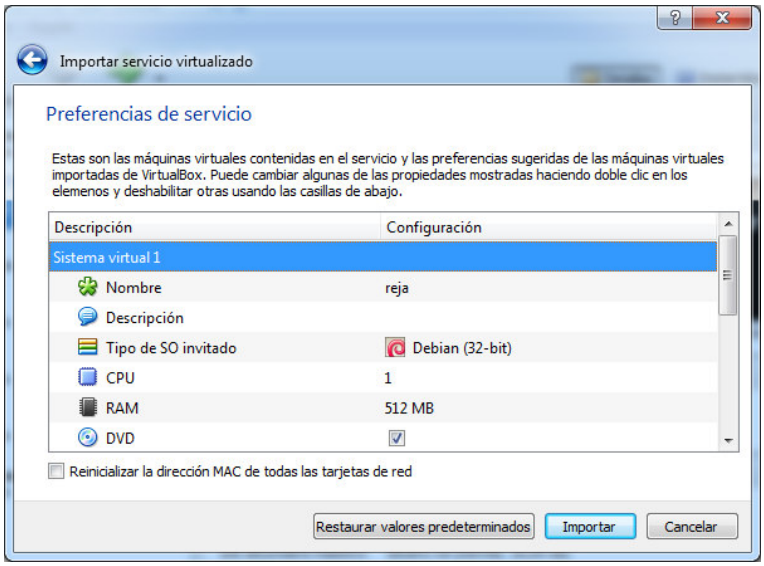


Figura A.3 Ventana de configuración

Una vez modificada la configuración, si fuera necesario, pulsamos en el botón "Importar" y esperamos a que VirtualBox realice la importación. Este proceso puede llevar unos minutos.

Cuando finalice el proceso veremos que tenemos un nuevo dispositivo de virtualización en nuestra lista como se puede observar en la figura A.4

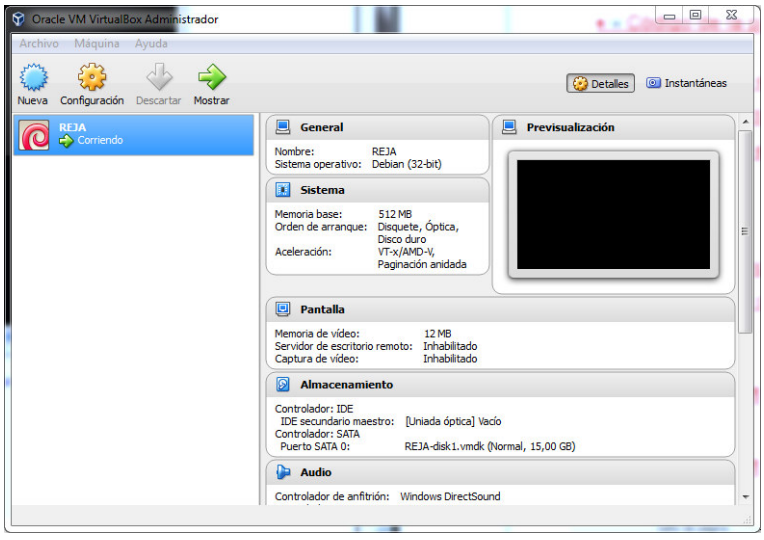


Figura A.4 Lista de Máquinas

Para finalizar, si hemos realizado el proceso correctamente, podremos iniciar la máquina virtual y acceder al sistema con el usuario y contraseña, estos están adjuntos en el apartado de descripción de la configuración de máquina virtual (véase figura A.5.

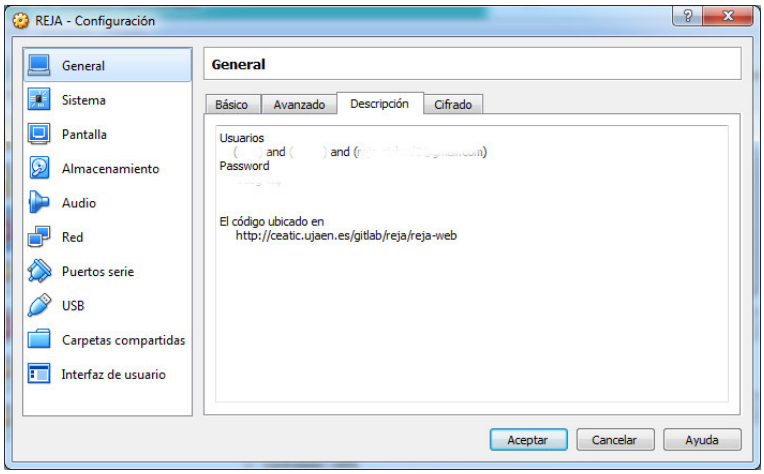


Figura A.5 Descripción Máquina Virtual

Cliente

Con el fin de poder ejecutar la app desarrollada sin depender de un dispositivo físico, se ha utilizado el software de virtualización Genymotion. Este software nos permite crear máquinas virtuales de Android, sobre VirtualBox.

Al estar basado en VirtualBox, el proceso para importar la máquina virtual de Android es el mismo, pero al finalizar la importación en lugar de iniciar la aplicación desde VirtualBox, tenemos que iniciarla desde Genymotion.

En este aparto vamos a explicar cómo configurar correctamente la tarjeta Ethernet virtual que VirtualBox crea para dar conexión a internet a la máquina virtual de Genymotion.

Esto se debe a la manera en que Genymotion configurar VirtualBox para que pueda tener acceso en red. Y es que utiliza una opción de red de VirtualBox llamado “Solo-Anfitrión”, que crea una red virtual sin necesidad de que el equipo anfitrión tenga una tarjeta de red. Y es que tras el proceso de importación nuestra máquina virtual de Android puede tener una dirección IP distinta a la que se usó para desarrollar la App por lo que una opción sería cambiar la dirección IP en la clase utils de Android (véase figura A.6) y volver a compilar todo la aplicación.

Figura A.6 URL base del servidor.

La otra opción es modificar la configuración de red de la conexión virtual creada por VirtualBox. Para ello tenemos que irnos a “Preferencias” (véase figura A.7)

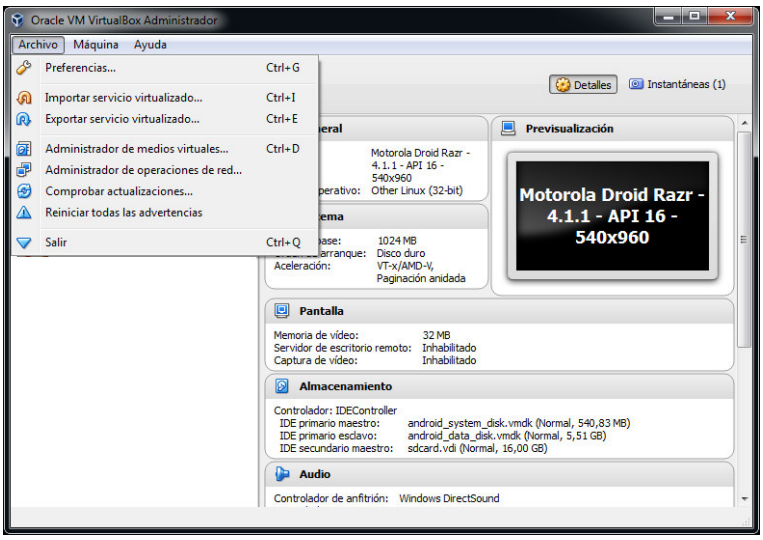


Figura A.7 Menú de preferencias.

Y Entrar en la opción de “Red” y seleccionar “Redes solo-anfitrión” (véase figura A.7).

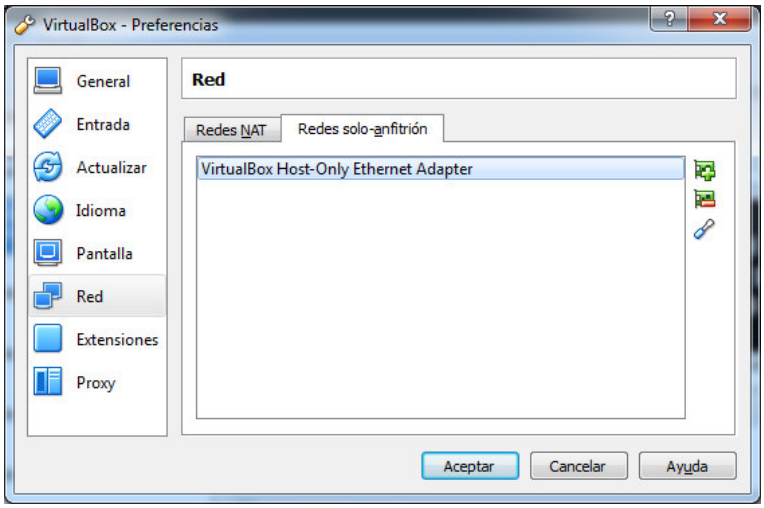


Figura A.8 Lista de redes.

Seleccionamos la red que use nuestra máquina virtual. Y hacemos doble click sobre ella, se nos desplegará la figurar A.9. En ella podemos ver 2 pestañas que hacen referencia a la dirección que tendrá el adaptador y al juego de dirección que tendrá el DCHP.

Como en el código de la App se especificó, la IP 192.168.56.1, daremos al adaptador de red esa IP, a continuación asignamos el rango de direcciones que se les dará a las máquinas virtuales que usen este adaptador, estas IP tendrán que ser

mayores que 192.168.56.1, en este caso ya que es la que le hemos dado al adaptador, . Es importante remarcar que al cambiar la dirección IP del adaptador cambiemos también la del DHCP ya que de no coincidir las IP VirtualBox creará otro adaptador nuevo con otra IP distinta.

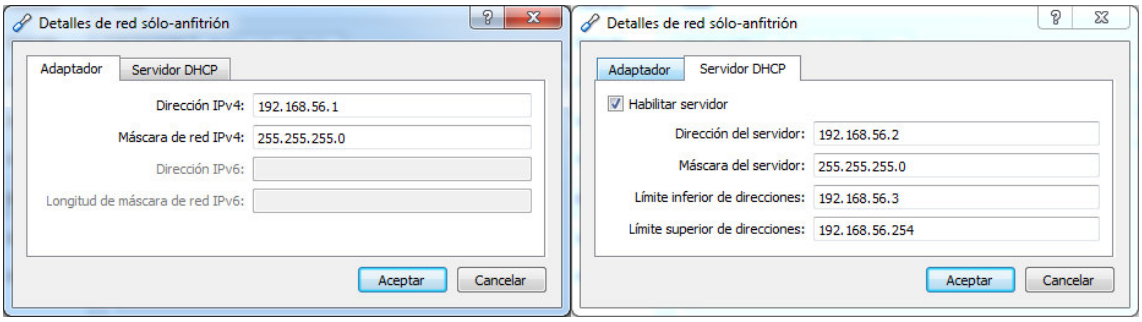


Figura A.9 Configuración servidor

Anexo B Manual de Usuario

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

B.1. Identificación.

Para acceder a la aplicación seleccionamos “iniciar Sesión” y nos aparecerá una ventana emergente con una listas de las cuentas de Google que tenemos en el dispositivo seleccionamos la que quieres utilizar (véase figura B.1) y automáticamente nos identifica en el sistema.

Tras acceder por primera vez a la aplicación y realizar la identificación de manera correcta, la aplicación nos muestra el menú principal.

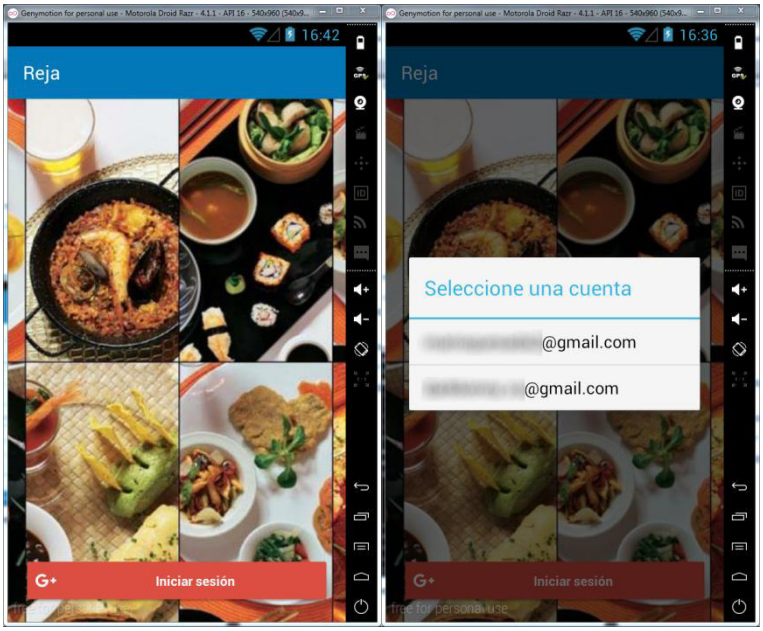


Figura B.1 Proceso identificación

B.2. Menú Principal.

El menú principal consta de 4 opciones. La primera corresponde a la pantalla "Recomendación" donde se podrá acceder a las recomendaciones generadas por el sistema de recomendación. A continuación, la segunda opción "Búsqueda", en la que se podrá realizar la búsqueda de restaurantes en base a una cadena de texto. La tercera opción es "Grupos" donde se podrá crear un grupo para que otros usuarios se unan al grupo, o buscar un grupo de otro usuario para unirse al grupo, con el fin de recibir una recomendación acorde a los gustos de todos los miembros del grupo. Por último, cuarta opción, es "Ajustes", nos permite modificar algunas opciones de la aplicación que se verán más adelante.

B.3. Recomendación

Para recibir la recomendación únicamente hay seleccionar "Recomendación" en el menú principal, y recibiremos automáticamente la recomendación calculada.

En esta pantalla encontramos un menú inferior con 3 opciones: "Lista", "Contexto" y "Geolocalizar".

La opción "Lista" nos indica que estamos visualizando la recomendación en una lista.

La opción "contexto" calcula la recomendación teniendo en cuenta la ubicación GPS del usuario (véase figura B.2).

Figura B.2 Recomendación Contextualizada

La opción “Geolocalizar” muestra sobre el mapa donde se ubican cada uno de los restaurantes ofrecidos en la recomendación y recomendación contextualizada (véase figura B.3).

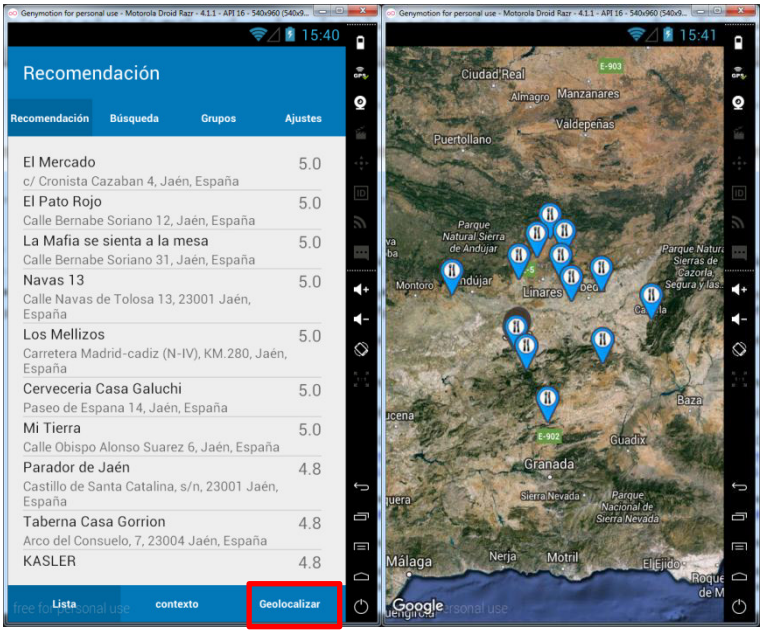


Figura B.3 Proceso Geo localizar Recomendación

B.4. Búsqueda de un restaurante.

Al seleccionar la segunda opción del menú principal, “Búsqueda”, accedemos a la pantalla donde podemos realizar una búsqueda de los restaurantes a partir de una cadena de texto. Por ejemplo, si introducimos la palabra “casa”, la aplicación nos mostrará todos los restaurantes que contengan en su nombre la palabra “casa” (Véase figura B.4).

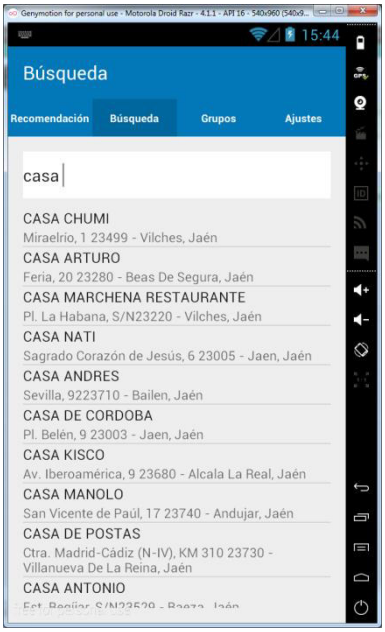


Figura B.4 Pantalla Búsqueda

B.5. Ver Detalles.

La aplicación nos facilita restaurantes de diversas maneras, por recomendación, buscándolos por su nombre, o viendo nuestras valoraciones (como veremos más adelante). Para poder ver la información de cada restaurante de manera más extensa, solo tenemos que seleccionar el restaurante deseado en la lista y se mostrará una pantalla con la información del restaurante (véase figura B.5). En esta pantalla podemos llamar al restaurante seleccionando el icono de llamada y valorar el restaurante seleccionando la nota en la barra de estrellas (de 1 a 5 la valoración) y mandar la puntación seleccionando “Puntuar”.

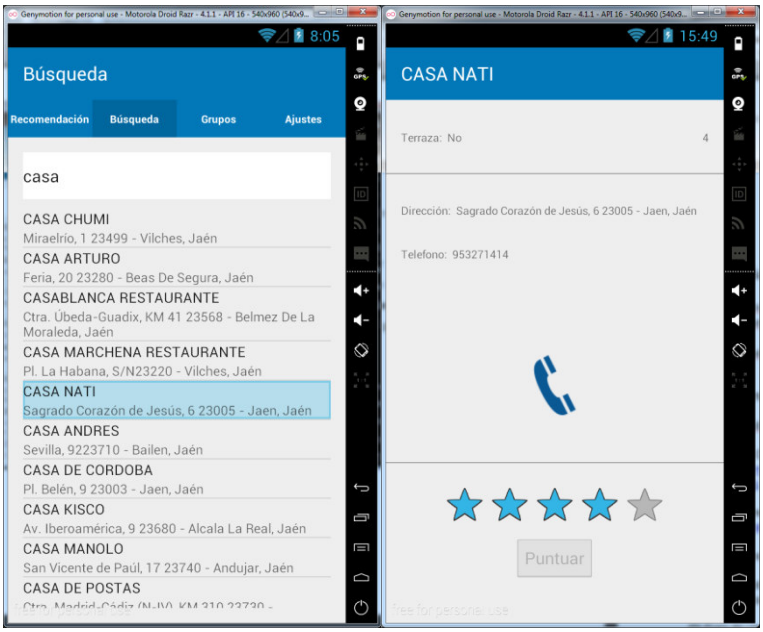


Figura B.5 Ver detalles.

B.6. Grupos.

En esta pantalla, lo primero que observamos es un menú con 2 opciones: “Crear” y “Unirse” (véase figura B.6).

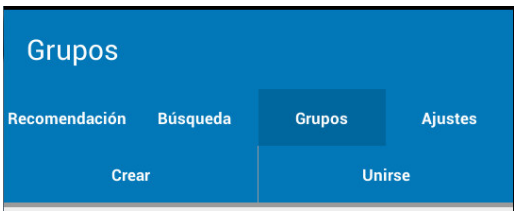


Figura B.6 Menú Grupos.

Al seleccionar “Crear” nos mostrará la pantalla donde podemos crear nuestro grupo para que otros usuario, se unan al grupo.

Al seleccionar “Unirse” nos mostrara la pantalla donde podemos buscar un grupo en concreto para unirnos al grupo.

B.6.1. Crear

En la pantalla introducimos el nombre del grupos y seleccionamos el botón “Crear”, en el caso de que el grupo ya haya sido creado por otro usuario, la aplicación nos mostrará un mensaje de error: “Este grupo ya existe” (véase figura B.7).

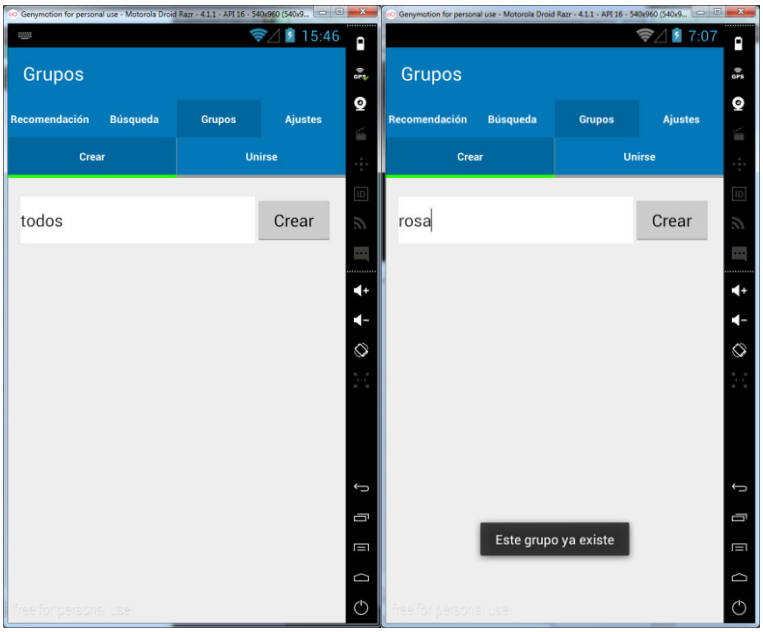


Figura B.7 Pantalla Crear Grupo

Una vez creado solo hay que esperar que lleguen los usuarios, los usuarios aparecerán en una lista llamada “Peticiones Pendiente”.

Para rechazarlos a un usuario hay que seleccionar el icono con una “×”. Los usuarios que deseemos aceptar los haremos miembros del grupo seleccionando el icono con él “✓”, estos usuario pasaran a verse en la lista de “Miembros del grupo”. Una vez estén los usuarios deseados en el grupo pedimos la recomendación seleccionando “Solicitar Recomendación” (véase figura B.8).

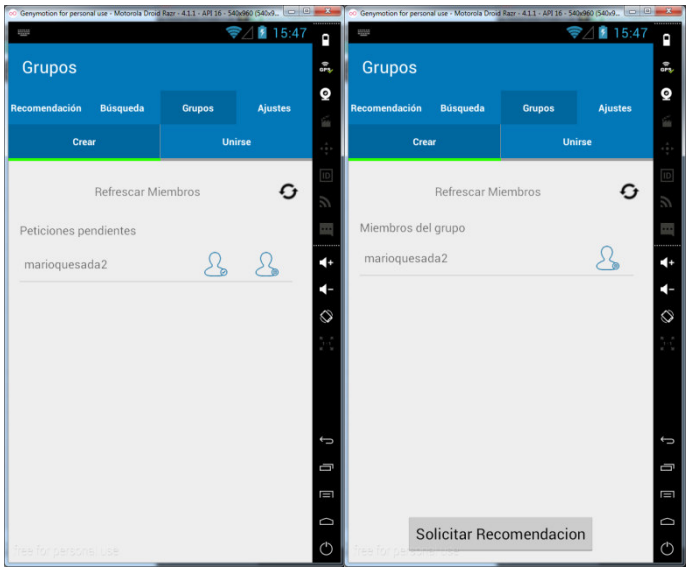


Figura B.8 Proceso gestión usuarios grupo

La recomendación recibida se mostrará en la aplicación del administrador, de la misma manera que una recomendación individual.

B.6.2. Unirse a un Grupo.

Para unirse a un grupo, hay que seleccionar la opción “Unirse”. Una vez en esta pantalla introducimos el nombre del grupo y la aplicación nos mostrará los grupos que coinciden con el texto introducido.

Si no introducimos texto alguno, la aplicación nos mostrará todos los grupos registrados en el sistema (véase figura B.9).

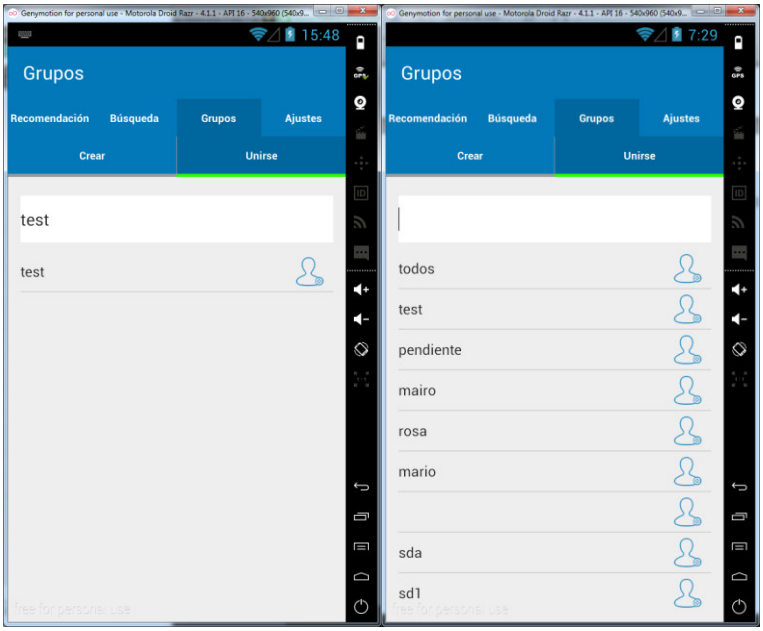


Figura B.9 Unirse a un grupo

Para solicitar la membresía en un grupo, únicamente hay que seleccionar el grupo en la lista, y esperar que el administrador del grupo, nos acepte.

B.7. Ajustes

En la última opción del menú principal, tenemos los “Ajustes”, en los que tenemos (véase figura B.10):

- *Mis puntuaciones*: Las puntuaciones realizadas a los restaurantes para poder modificarlas.
- *Pantalla Principal*: Cambiar la pantalla de predefinida cuando se inicie la aplicación.

- *Distancia*: Cambiar el radio de filtrado para la recomendación contextualizada.
- Desvincular la cuenta.

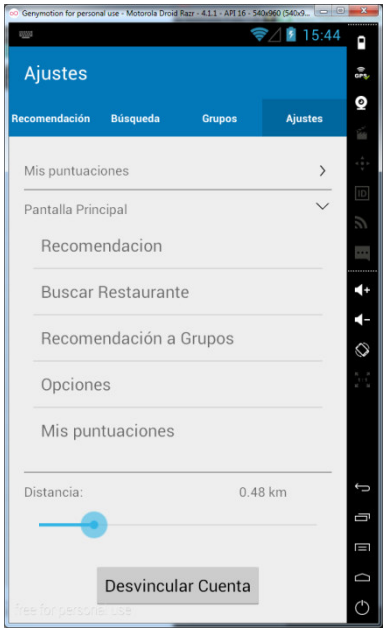


Figura B.10 Pantalla de Ajustes

B.7.1. Mis puntuaciones

En esta pantalla podemos ver una lista de todas las valoraciones que hemos realizado, asi como borrar valoraciones, o modificarlas.

Para borrar una valoracion, hay que mantener seleccionado la valoraciones deseada durante un breve periodo de tiempo, tras esto, la aplicación nos preguntará si deseamos eliminar dicha valoración (vease figura B.11).

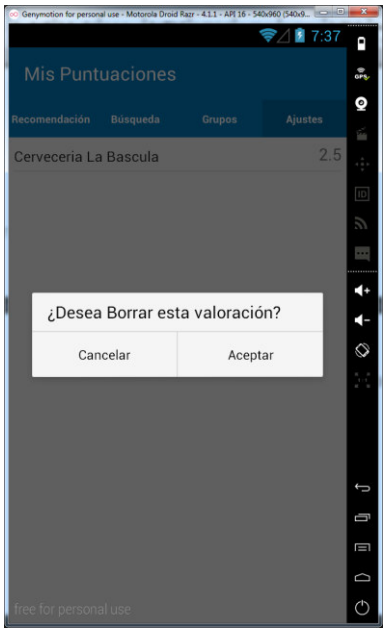


Figura B.11 Eliminar Valoración

B.7.2. Pantalla Principal

Para modificar la pantalla que nos muestra la aplicación cuando esta se inicia, solo hay que seleccionar la opción que deseamos.

B.7.3. Distancia

Para modificar el rango de busqueda para la recomendación contextualizada, solo hay que seleccionar en la barra la distancia que queremos, en la parte izquierda, tenemos un valor de 0 metros y en la derecha un valor de 71 Kilómetros.

B.7.4. Desvincular Perfil

Para eliminar nuestra cuenta de la aplicación, solo tenemos que seleccionar “Desvincular Cuenta”.

Bibliografía

[1] Balabanovic, M. y Shoham, Y. (1997), "**Content-based, callaborative recommendation**". Communications of the ACM, Vol.40

[2] Martin Fowler con Kendall Scott, "**UML Gota a Gota**", Pearson Educación, cop.1999

[3] Sandy Carter, "**The New Language Of Business Soa & Web 2.0**", IBM Press/Pearson 2007

[4] Bill N. Schilit, Norman Adams, and Roy Want, "**Context-Aware Computing Applications**", , IEEE Workshop on Mobile Computing Systems and Applications, December 8-9 1994

[5] **Pervasive Computing (Ubiquitous Computing)**, http://internetofthingsagenda.techtarget.com/definition/pervasive-computing-ubiquitous-computing. Último acceso: marzo 2016

[6] **Smartphone OS Market Share, 2015 Q2**, http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Último acceso: marzo 2016

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

[8] **Ciclo de vida Activiy**, http://developer.android.com/reference/android/app/Activity.html, Último acceso: marzo 2016

[9] **Ciclo de vida de un Fragment**, http://developer.android.com/intl/es/guide/components/fragments.html. Último acceso: marzo 2016

[10] **Permisos Android**, http://developer.android.com/guide/topics/security/permissions.html, Último Acceso: mayo 2016

[11] **Almacenamiento Android**, http://developer.android.com/guide/topics/data/data-storage.html. Ultimo acceso: mayo 2016

[12] **Service (systems architecture),** http://en.wikipedia.org/wiki/Service_(systems_architecture). Último acceso: mayo 2016

[13] **Guía Breve de Servicios Web**, http://www.w3c.es/Divulgacion/GuiasBreves/ServiciosWeb. Ultimo acceso: mayo 2016

[14] **Tasa distribución de las versiones e la API**, https://developer.android.com/about/dashboards/index.html. Ultimo acceso: mayo 2016

[15] **SOAP vs REST**, http://carlosmayta.blogspot.com.es/. Último acceso: mayo 2016

[16] **REST vs SOAP al servicio de la web**, http://inusual.com/articulos/rest-vs-soap-al-servicio-de-la-web/. Último acceso: mayo 2016

[17] **Reprensentational State Transfer**, https://es.wikipedia.org/wiki/Representational_State_Transfer. Último acceso: mayo 2016

[18] **Service-oriented architecture,** https://en.wikipedia.org/wiki/Service-oriented_architecture. Último acceso: mayo 2016

[19] **Servicio Web**, https://es.wikipedia.org/wiki/Servicio_web. Último acceso: mayo 2016

[20] Thomas Erl, Benjamin Carlyle, Cesare Pautasso, Raj Balasubramanian, "**SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST**", Prentice Hall 2012

[21] Douglas K. Barry, "**Web Services, Service-Oriented Architectures, and Cloud Computing**", Morgan Kaufmann 2ª edición

[22] Leonard Richardson; Sam Ruby, "**RESTful Web Services**", O'Reilly Media, Inc. 2007

[23] **¿Cuánta Información se Genera y Almacena en el Mundo?**, https://documania20.wordpress.com/2013/09/16/cuanta-informacion-se-genera-y-almacena-en-el-mundo/. Último acceso abril 2016

[24] **¿Que son los Sistemas de Recomendación?**, http://jarroba.com/que-son-los-sistemas-de-recomendacion/. Último acceso: abril 2016

[25] Michael D. Ekstrand, John T. Riedl, Joseph A. Konstan, "**Collaborative Filtering Recommender Systems**",

[26] Guy Shani and Asela Gunawardana, "**Evaluating Recommendation Systems**"

[27] Pazzani M.J., "**A Framework for Collaborative, Content-Based and Demographic Filtering**"

[28] Francesco Ricci, Lior Rokach and Bracha Shapira, "**Recommender Systems Handbook**"

[29] Joseph A. konstan, John Riedl, "**Deconstructing Recommender Systems**"

[30] Evan Miller, "**How Not To Sort By Average Rating**"

[31] Pasquale Lops, Marco de Gemmis and Giovanni Semeraro, "**Content-based Recommender Systems: State of the Art and Trends**"

[32] Emilio J. Castellano, Manuel J. Barranco, Luis Martínez, "**Academic Orientation Supported by Hybrid Intelligent Decision Support System**".

[33] Gediminas Adomavicius, Alexander Tuzhilin, "**Context-Aware Recommender Systems**"

[34] Sauter Vicki Lynn, "**Decision support systems for business intelligence**", Hoboken, N.J. : Wiley, c2010 2ª edición.

[35] Balabanovic, M. y Shoham, Y. (1997), **"Content-based, callaborative recommendation**". Communications of the ACM, VOL 40

[36] **Open Handset Alliance**, http://www.openhandsetalliance.com/. Último acceso: junio 2016

[37] **Estadísticas Android**, https://developer.android.com/about/dashboards/index.html?utm_source=suzunone, Ultimo acceso: junio 2016

[38] Pressman Roger S**," Software Engineering: A Practitioner's Approach**", McGraw-Hill 8ª edición

[39] Ivan Marsic, "**Software Engineering**", Rutgers 2012.

[40] Roger S. Pressman, "**Software Engineering, A Practition Approach**", McGraw-Hill 5ª Edición.

[41] Ramez Elmasri, Shamkant B. Navathe, "**Fundamentos de Sistemas de Bases de Datos**", Pearson 5ª edición.

[42] Thomas M. Connolly, Carolyn E. Begg, "**Database Systems**", 3ªedición.

[43] Alan Dix, Janet Finlay, Gregory D. Abowd, Russell Beale, "**Human-Computer Interaction**", Pearson 3ª edición.

[44] Ben Shneiderman, Catherine Plaisant, "**Designing the User Interface**", Pearson 5ª edición.

[45] Pressman Roger S.," **Software Engineering: A Practitioner's Approach**.", McGraw-Hill 8ª Edición.

[46] **"Guía de Validación y Verificación",** www.inteco.es/file/XaXZyrAaEYfaXKiMJlkT_g. Último acceso: junio 2016

[47] Robert C. Martin, **"Clean Code**", Anaya 2012