Trabajo de Fin de Grado.
MadridManía: Un sistema de recomendación social y de ocio para grupos

MEMORIA DEL PROYECTO

Carlos Guillermo López Pérez
Marta Oliver Valiente

———————

Dirigido por: Belén Díaz Agudo

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática
Universidad Complutense de Madrid
Junio 2015

Trabajo de Fin de Grado.
MadridManía: Un sistema de recomendación social y de ocio para grupos

Memoria de Trabajo de Fin de Grado

Ingeniería del Software e Inteligencia Artificial

06/2015

Departamento de Ingeniería del Software e Inteligencia Artificial

Facultad de Informática

Universidad Complutense de Madrid

Junio 2015

Autorización

Marta Oliver Valiente y Carlos Guillermo López Pérez, alumnos matriculados en el grado de ingeniería informática en la rama de las tecnologías de la información, autorizan, mediante el presente documento, a la Universidad Complutense de Madrid (UCM), a difundir y utilizar con fines académicos, no comerciales, y mencionando expresamente a sus autores, tanto en la propia memoria, como el código, la documentación y/o el prototipo desarrollado. Todo ello realizado durante el curso académico 2014 – 2015 bajo la dirección de Belén Díaz Agudo, profesora del Departamento de Ingeniería del Software e Inteligencia Artificial.

Marta Oliver Valiente Carlos Guillermo López Pérez

Agradecimientos

Si quisiéramos poder expresar los agradecimientos a todas las personas que nos han estado apoyando, sería muy complicado. Si queremos hacerlo en unas líneas, resulta mucho más complicado. Esperemos tener palabras para todos.

En primer lugar a nuestra directora de proyecto, Belén Díaz Agudo. Por confiar en nosotros, y permitirnos orientar el proyecto hacia nuestro aprendizaje, ofreciéndonos siempre un amplio abanico de posibilidades, que nos han permitido mejorar un proyecto ya construido, y aprender conceptos de asignaturas que nosotros no hemos cursado.

Gracias al resto de profesores, que tanto nos han enseñado. Ha sido una etapa muy intensa, y nos vamos con un buen recuerdo de todos. Por hacernos ingenieros, por compartir vuestros conocimientos con nosotros. Asimismo al resto del personal de nuestra facultad (gracias señoras de la limpieza por un entorno de trabajo tan limpio, personal de cafetería, conserjería…).

A nuestros compañeros, amigos en muchos casos. Por ayudarnos en épocas de exámenes. Cuando hemos coincidido en el mismo aula y cuando no. Por formar parte de esta gran etapa.

Cómo olvidarnos de nuestras familias, con el apoyo que muchas veces no sabemos ver, y que en muchos casos aparte de hacernos mejor estudiantes nos hacen algo que es todavía más importante: mejor personas.

Por último queremos expresar un agradecimiento mutuo entre nosotros, Marta y Guillermo. Por apoyarnos en los momentos más complicados y disfrutar en los buenos. Por tener siempre esperanza y creer en el otro hasta el final. De otra forma no hubiese sido posible.

A todos vosotros, que habéis mostrado vuestro apoyo durante estos años, os dedicamos este proyecto de fin de grado.

Resumen

Durante los últimos años han surgido multitud de plataformas cuyo objetivo es facilitar a sus usuarios la tarea de planificar sus salidas de ocio, generalmente incorporando algún tipo de sistema de recomendación que tenga en cuenta sus gustos o preferencias. El crecimiento en el uso de los smartphones, dispositivos que hacen que internet esté al alcance de la mano en cualquier momento y lugar, ha propiciado que recomendadores de ocio web clásicos como TripAdvisor y Groupon hayan pasado a formar parte también del mundo de las aplicaciones móviles, uniéndose a otros nuevos como Fever o Whatsred.

MadridManía surge como una continuación de Madrid Live[1] y tiene como objetivo el desarrollo de una nueva aplicación móvil que implemente un sistema de recomendación de planes de ocio grupales para la ciudad de Madrid. Nos hemos centrado en la generación de planes para grupos ya que este tipo de actividades raramente se lleva a cabo de manera individual.

Nuestro propósito es mejorar el funcionamiento del sistema anterior mediante la incorporación de un método de caracterización de grupos y la selección de un rango de edades que permita al usuario seleccionar manualmente el tipo de grupo hacia el que irá dirigida la recomendación y una nueva función de agregación que tenga en cuenta estos modelos de grupos, la adición de nuevas actividades de ocio (centros comerciales, parques de ocio, hoteles) que se unan a las ya existentes (museos, restaurantes, parques, paseos, cines) y la creación de planes de ocio de más de un día de duración. Además, el sistema sigue manteniendo el modo de funcionamiento con la red social Facebook que permite la creación de grupos incorporando a amigos que se encuentren en un radio de distancia cercano a nosotros.

Este prototipo incorpora los rangos de edades niños, jóvenes, jóvenes-adultos y adultos, así como los tipos de grupo familia, pareja, trabajo y amigos.

[1] Madrid Live: un sistema de recomendación de ocio social y contextual para la ciudad de Madrid (http://eprints.ucm.es/26502/)

Palabras clave: sistemas de recomendación, caracterización de grupos, modelos de grupo, redes sociales, CBR.

Abstract

Over the last few years, a considerable amount of platforms which aim to make the task of planning our free time activities easier for us has arisen, usually including some kind of recommender system that takes in mind our likes or preferences. The increasing use of smartphones and having access to the internet everywhere at any time has spurred that classic free time web recommender systems as TripAdvisor and Groupon develops into mobile applications, joining new ones like Fever or Whatsred.

MadridMania is a continuation of Madrid Live[2] and its main purpose is to develop a new mobile application that includes a recommender system of leisure activities for groups in the city of Madrid. We have concentrated on generating plans for groups as this type of activities are hardly ever doing alone.

We aim to improve the previous system by including a characterization group method and a range of ages that make it possible for users to manually select the type of group who are looking for a recommendation and a new aggregation function that takes this characterization in mind, adding new leisure activities (shopping malls, amusement parks, hotels) that join existing ones (museums, restaurants, parks, walks, cinemas) and creating plans for more than one day. Besides, the system keeps working with Facebook in order to generate groups by adding friends that are near us.

This prototype includes the range of ages children, teenagers, young adults and adults, as well as the the types of group family, couple, work and friends.

Keywords: recommender systems, group characterization, group models, social networks, CBR.

[2] Madrid Live: social and contextual leisure recommender for the city of Madrid (http://eprints.ucm.es/26502/)

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo explicamos la motivación para la realización del proyecto MadridManía, los objetivos que esperamos conseguir con él y una breve descripción acerca de cómo se estructura la memoria.

1.1. Motivación

Madrid es una ciudad cosmopolita que cuenta con multitud de actividades de ocio distintas para sus habitantes y visitantes. La cantidad de actividades distintas que ofrece es tal, que resulta lógico ayudarse de Internet a la hora de encontrar cuáles de éstas podrían adaptarse mejor a nuestras preferencias.

Para llevar a cabo esta búsqueda, un usuario puede navegar con su ordenador hasta una web de recomendación de planes de ocio como TripAdvisor o utilizar una aplicación móvil dentro de su *smartphone* como Fever, y obtener aquellas actividades que más se ajusten a sus a gustos en función de sus preferencias personales.

Basándonos en estos sistemas, MadridManía implementa un sistema de recomendación capaz de encontrar aquellos planes que más puedan satisfacer a un usuario a la hora de organizar una jornada de ocio por Madrid. Partiendo del sistema Madrid Live, tratamos de optimizar el diseño de este, para conseguir una aplicación que mejore la experiencia del usuario. En Madrid Live era obligatorio ingresar con nuestra cuenta de Facebook para poder utilizar la aplicación, lo cual limitaba su uso únicamente a usuarios de esta red social. Para paliar este problema, hemos desarrollado un nuevo modo de funcionamiento que se une al ya existente, al que hemos denominado ***modo invitado***. Este nuevo modo permite a cualquier usuario utilizar la aplicación para solicitar un plan de ocio.

Debido a que las actividades de ocio suelen llevarse a cabo en grupo, los sistemas de recomendación han evolucionado para cubrir esta demanda, surgiendo los denominados sistemas de recomendación grupales. Estos sistemas tienen en cuenta las preferencias de cada miembro del grupo y recomiendan actividades que satisfagan lo mejor posible a todos sus componentes. Para ello, MadridManía incorpora un sistema de recomendación grupal en el que se tienen en cuenta las relaciones sociales de las personas que componen el grupo, mejorando de esta forma la búsqueda de actividades.

En Madrid Live la interacción con grupos estaba limitaba, ya que los componentes de éste, debían ser usuarios de Facebook. MadridManía permite a cualquier usuario seleccionar el tipo de grupo hacia el que va dirigida la recomendación (familia, pareja, trabajo o amigos) y también, opcionalmente, un rango de edades para los componentes de dicho grupo (niños, jóvenes, jóvenes-adultos o adultos). La motivación para incluir estos tipos de grupos y rangos de edades y no otros, proviene del resultado de un análisis que determina hacia qué tipo de usuarios queremos que vaya dirigida nuestra aplicación (capítulo 6).

En base a las opciones seleccionadas por el usuario, se genera un ***modelo de grupo*** y se asignan distintos pesos a los tipos de actividades (sección…). De esta manera, en el caso de que el usuario seleccione el tipo de grupo familia, rango de edades niños y como actividades restaurante y cine, el modelo de grupo generado será el de una familia con niños y se le asignará un mayor peso a los restaurantes de comida rápida y a las películas infantiles.

Lo más complicado a la hora de llevar a cabo una recomendación es obtener las preferencias concretas de un usuario. En Madrid Live esto se realizaba por medio de un test de preferencias inicial. Al usuario se le presentaba un listado de tipos y subtipos de actividades que éste debía ordenar en función de su mayor o menor preferencia. Este test es efectivo a la hora de conocer los gustos de un usuario, pero resultaba tedioso de rellenar. ¿Por qué rellenar información acerca de museos si el usuario no desea en este momento que se le recomiende uno de ellos? Por esto, en MadridManía hemos decidido eliminar dichos tests iniciales y que sea la propia aplicación la que rellene estas preferencias en función de la edad y el tipo de grupo seleccionados. Así, evitamos el problema de que el uso de la aplicación resulte pesado y no perdemos con ello las preferencias del usuario para un tipo de plan determinado.

Por lo tanto, la motivación de este trabajo es permitir a cualquier usuario (modo invitado) planificar actividades de ocio por la ciudad de Madrid. En el caso de las recomendaciones para grupos, el objetivo, es proporcionar a los usuarios un sistema de recomendación grupal (modelos de grupo) mediante la selección del tipo de grupo y la edad a la que va dirigido el plan. Por último, es probable que en muchos casos los usuarios de nuestra aplicación prolonguen su estancia en Madrid por más de un día. Para tener en cuenta este caso, entre de las ***nuevas actividades*** incluídas (sección 5.3), hemos añadido la posibilidad de seleccionar un hotel al final del día para que el usuario pueda alojarse y seguir disfrutando al día siguiente de nuevas actividades de ocio.

1.2. Objetivos

El principal objetivo de MadridManía es ser una aplicación móvil útil para planificar actividades de ocio en grupo para la ciudad de Madrid. El sistema debe ser capaz de ofrecer recomendaciones que satisfagan a todos los componentes del grupo, ya sea obteniendo el contexto social de éste a través de Facebook o por medio de un modelo de grupo.

Para esto, en MadridManía se presentan las siguientes novedades respecto a Madrid Live:

1. Creación de un modo invitado que permita utilizar la aplicación sin necesidad de Facebook.
2. Creación de caracterización de grupos y modelo de grupos para llevar a cabo las recomendaciones.
3. Sustitución de test de preferencias iniciales por caracterización de grupos.
4. Incluir nuevas actividades de ocio al margen de las originales (centros comerciales, parques de ocio, hoteles).
5. Mejora de la interfaz de la aplicación.

1.3. Estructura de la memoria

Esta memoria se estructura de la siguiente manera:

1. En este primer capítulo se ha presentado la introducción, la motivación del sistema y los objetivos que se quieren lograr realizando este proyecto.

2. En el capítulo 2 se expone el estado del arte. Para realizar un proyecto de estas características es necesario realizar un estado del arte de los sistemas de recomendación individuales y grupales, haciendo especial hincapié en aquellos implementados en aplicaciones móviles.

3. El capítulo 3 describe los tipos de datos, las fuentes de información y detalla de manera funcional la aplicación.

4. A continuación, en el capítulo 4 se describe cuál es la arquitectura de MadridManía comparándola con la de su antecesora, Madrid Live, y especificando cuáles han sido los cambios que hemos realizado para adaptar los nuevos módulos o subsistemas.

5. En el capítulo 5 se presentan los módulos añadidos como subsistemas al proyecto anterior. Para cada uno, se incluye el objetivo del subsistema con el que queremos mejorar la aplicación, junto con una vista en forma esquemática y una explicación teórica.

6. En el capítulo 6 se describen las motivaciones que han fundamentado nuestra línea de desarrollo. En todo proyecto es clave una valoración experimental, pero en este caso la hemos realizado antes y después de la implementación, para obtener una lista de puntos a mejorar y contrastar los resultados de Madrid Live con MadridManía.

7. En el capítulo 7 se establecen conclusiones del proyecto a nivel técnico, de aprendizaje, motivación, implicación personal y creatividad. En definitiva, qué nos aporta el proyecto, y qué pretendemos con él.

8. Por último, en el capítulo 8 se explican las líneas de trabajo futuro sobre posibles nuevos subsistemas que se proponen para próximas implementaciones. Tanto para alumnos que realizan un proyecto similar, como para nosotros si continuamos con el desarrollo.

Chapter 1

Introduction

In this chapter we are going to explain the reason why we are doing this project and how MadridMania is developed. In addition to this,we will be giving a briefly description of the structure of this report and an explanation of how the target is going to be pursued.

1.1. Motivation

Madrid is a very international city with a wide range of activities not only for its citizens but also for its visitors. As a result of this huge cultural offer, it is logical not wasting time searching a wide variety of activities and reduce the spectrum by choosing the activities we are interested in. By this, it seems reasonable to take advantage of technologies such as the Internet to find which activity fits with our requirements. To achieve its objective, users usually navigate on their computer until they find a leisure website such as TripAdvisor or an app like Fever if they are using a smartphone. These services allow users to find activities affine to their priorities and are based on users' feedback helping them to find activities that suit with their personal preferences. MadridMania is based on the same principles. It offers leisure plans relevant for the users.

Leisure-activities are usually done in groups. As a matter of fact,Curate content systems have evolved to fulfill these features,and collaborative filtering has been developed. This method takes into account the preferences of each member of the group and returns activities that satisfied most of the member within the group. MadridMania embedded a collaborative filtering based on social relation between friends. Improving the relevance of the activity offers to the user.

In our first approach users had tosign in using their Facebook account to be able to use the app. We realized that these requirement constraint our potential user only to the ones that already are users of this social network. To solve this problem wehave developed a new pipeline added to the previous one. We have called it **guest mode.**

In guest mode is the user who decides which group is going to be typed and the age that is going to be searched (e.g. family, partner, work or friends) and, optionally, a range of ages for the members of that group (children, youth, young - adults or adults). The reason to include these features and avoid others is made by thinking in our target audience. Based on the selected options a model group is generated and different relevance is created for the different types of activities, giving a higher importance to one or another when carrying out a recommendation. For example, if we select the type **family** group,a range in **children age** and **restaurant or cinema activities**, The group model that will be generated is a family with kids and it will assigne a higher importance to the fast food restaurants and children's films. By this,the application has found the satisfaction of the group.

The hardest thing when performing a specific recommendation is to obtain a user's preferences. In our original project that was carried out through an initial preference test, in which the user is presented with a list of types and subtypes of activities that user had to order according to their degree of preference. This test is effective when it comes to know the tastes of a user, but is somewhat tedious to fill in. Why users need to provided information about museums if they are not interested at this moment?. For this reason, in MadridMania have decided to remove these tests and let the application fill in these preference itself automatically depending on the age and type of the selected group. With this, we avoid the problem that the use of the application would become annoying and at the same time not lose user’s preference.

Finally, it is likely that in many cases the users prolong their stay in the city of Madrid for over a day. To cover this case, we have included the possibility of generating plans for more than one day. We also have included some accommodations places like hotels, where users can spend the nights.

1.2. Objectives

The main objective of MadridManía is to become a useful mobile application for planning group leisure activities in the city of Madrid. The system must be able to provide recommendations to satisfy all the members of a group. Either by obtaining the social context of this through Facebook or by means of a model group.

For this, the following objectives MadridManía wants:

1. Creating a guest mode which allows using the application without Facebook.
2. Creating characterization of groups and model groups to carry out the recommendations.
3. Replacing characterization test group preferences.
4. Include new leisure activities outside the mainstream (centers Commercial, leisure parks, hotels).
5. Improved the user interface of the application.

1.3 Memory Structure

This report is structured as follows:

1. In this first chapter has presented the introduction, motivation system and the objectives to be achieved by performing this project.
2. In Chapter 2 the state of the art is exposed. For a project of this nature it is necessary to make a state of the art systems of individual and group recommendation, with particular emphasis on those implemented in mobile applications.
3. Chapter 3 works as a complement to the above. Described in more detail the architecture and each of the modules that define MadridManía.
4. Then, in Chapter 4 describes what architecture of MadridManía compared against that of its predecessor, Madrid Live, and specifying which were the changes we have made to adapt the new modules or subsystems.
5. In Chapter 5 add-on modules as subsystems to the previous project are presented. For each, the objective of the subsystem with which we want to

improve the application, along with a schematic view including a theoretical explanation.

6. In Chapter 6, one of the reasons that our line has based development suite. Every project must have an experimental evaluation, but in this case we have done before and after deployment to compare the results of Madrid Live
7. Chapter 7 conclusions of the project are set at a technical level, learning, motivation, personal commitment, creativity. In short, what we want to achieve with it.
8. Finally, in Chapter 8 lines of future work on possible new subsystems proposed for future implementations are explained. Both for students doing a similar project, and for us if we continue with development.

Capítulo 2

Estado del arte

2.1. Sistemas de recomendación

Durante nuestra vida cotidiana, es habitual que se nos presenten situaciones en las cuales tenemos que seleccionar una opción de entre varias disponibles. Tareas como elegir una película para ver o un restaurante para cenar se tornan complicadas cuando la cantidad de alternativas de las que disponemos es muy grande. Para paliar este problema surgen los denominados *sistemas de recomendación* [Bridge et al., 2006a], que tienen como tarea principal elegir ciertos objetos que cumplen con los requisitos de los usuarios, encontrándose estos objetos almacenados en un sistema informático [Wang, 1998].

2.1.1. Recomendadores individuales

Los sistemas recomendadores individuales tienen como objetivo sugerir elementos que sean del interés de los usuarios. Estas recomendaciones pueden llevarse a cabo de dos formas distintas: basándonos en su perfil y/o en sus preferencias actual o en su entorno social (por ejemplo, Amazon).

En función del lugar de procedencia de estas características, podemos diferenciar dos tipos de sistemas de recomendación:

1. **Basados en contenido**. Para llevar a cabo una recomendación se comparan las preferencias del usuario con las características del producto.
2. **Basados en filtrado colaborativo**. Para llevar a cabo una recomendación se buscan productos que otros usuarios con preferencias similares hayan valorado positivamente.

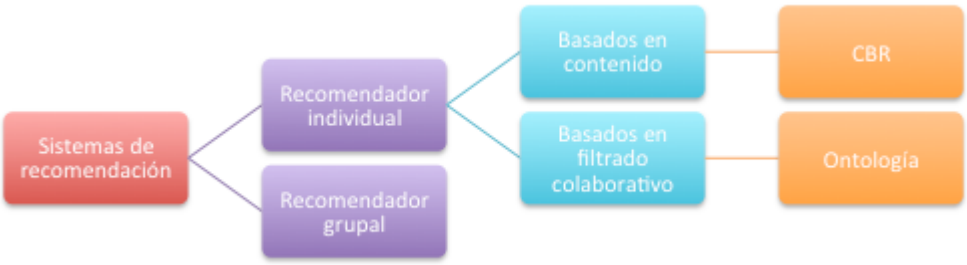


Figura 2.1: Sistemas recomendadores

En el caso de MadridManía se ha implementado un sistema de recomendación *basado en contenido*.

2.1.1.1. Sistemas de recomendación basados en contenido

Los sistemas de recomendación basados en contenido buscan productos cuyas características se ajusten lo mejor posible a las preferencias del usuario. La información obtenida se analiza para buscar propiedades, caracterísicas y cualidades de los productos. El razonamiento de estos sistemas se basa en que si a un usuario le gustan los productos de un determinado dominio, también le gustarán otros productos de características similares.

Los sistemas de recomendación basados en contenido poseen un gran paralelismo con el ***razonamiento basado en casos***[3] [Pazzani y Billsus, 2007], considerado uno de los paradigmas de resolución de problemas más exitosos y ampliamente utilizados el ámbito de la Inteligencia Artificial.

2.1.1.2. Razonamiento basado en casos: CBR

El Razonamiento basado en casos es el proceso de solucionar nuevos problemas basándose en las soluciones de problemas anteriores. Estos sistemas cuentan con una *base de casos* que se compone de problemas que ya fueron resueltos anteriormente, así como la solución a los mismos. De este modo, los nuevos problemas que surjan se resolverán mediante la adaptación de soluciones pasadas que fueron capaces de resolver problemas similares a estos. Si aparece un problema

[3] CBR, del inglés *Cased Based Reasoning*

resuelto con anterioridad, se recupera la solución de la base de casos, y posteriormente se aplica.

Ciclo CBR. El razonamiento basado en casos se explica mediante el ciclo CBR (Figura 2.2). El ciclo cuenta con 4 fases diferenciadas, que definimos a continuación:

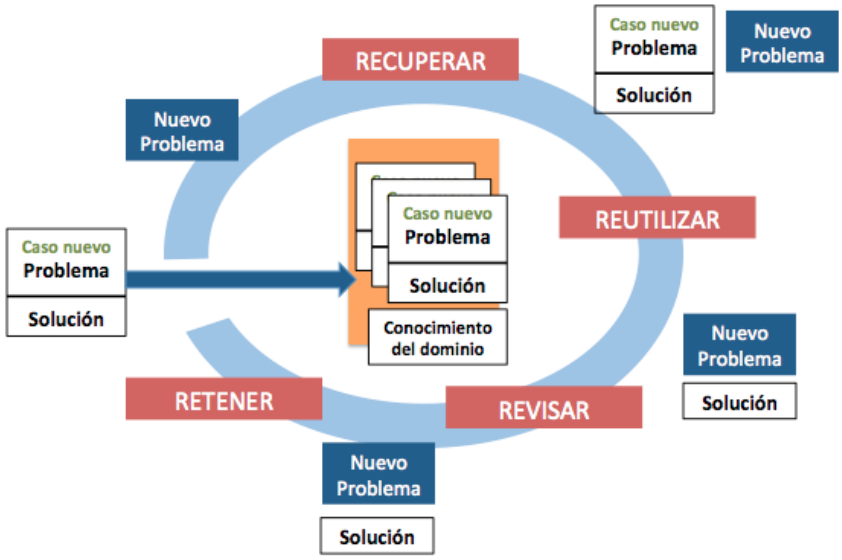


Figura 2.2: Ciclo CBR

1. **Recuperar.** En esta fase se recuperan los casos que presentan una mayor similitud para una consulta dada.

2. **Reutilizar.** Durante esta fase se extraen la solución del caso seleccionado para utilizarla.

3. **Revisar**. En esta fase se revisa esta solución y se modifica para que se ajuste mejor al caso que se quiere resolver.

4. **Recordar**. Finalmente, después de haberse aplicado la solución exitosamente, es importante recordar ésta y añadirla a la base de casos si aporta algo nuevo.

Puede observarse la similitud entre los sistemas de recomendación basados en contenido y los sistemas CBR, pues las características de los elementos y las preferencias de los usuarios que hay que emparejar se corresponden con los casos

que almacenan la descripción de un problema, siendo la recomendación solución a estos casos.
La aplicación del Ciclo CBR para las nuevas actividades se detalla a lo largo de la sección 5.3. El proceso, aplicado a los centros comerciales, es el siguiente:

Cuando ya se han recuperado correctamente todos los posibles centros comerciales, se han almacenado en una base de casos y se ha asignado un peso a cada uno de los atributos, se procede a ejecutar el ciclo CBR, que se encarga de evaluar cada uno de los elementos de la base de casos y devuelve un lista ordenada con un "valor de satisfacción", que indica el grado de ajuste de dicho centro comercial a las preferencias del usuario. Cuanto mayor sea el valor, mejor será el ajuste. Aunque sería posible recuperar solamente los N elementos de mayor "valor de satisfacción", se trabaja con toda la lista completa para aplicar filtrados posteriores.

2.1.1.3. Sistemas de recomendación basados en filtrado colaborativo

Al ser Internet una inmensa fuente de datos es necesario crear mecanismos capaces de filtrar toda esa información disponible. Una de las técnicas que han sido utilizadas por algunos sistemas de recomendación para enfrentarse a este problema es el filtrado colaborativo.

Un sistema de recomendación basado en filtrado colaborativo es aquel en el que las recomendaciones se llevan a cabo basándose en términos de similitud con otros usuarios. Estos sistemas, recomiendan elementos que han gustado a otros usuarios con intereses similares. Es decir, en vez sugerir elementos que le gustaban a un usuario en el pasado (como ocurría en los sistemas basados en contenido), escoge los que han preferido otros usuarios con iguales intereses o gustos.

El *workflow* de un sistema de filtrado colaborativo se describe en tres etapas:

1. Recoger las valoraciones que cada usuario da a diferentes elementos (en nuestro sistema, las distintas actividades: restaurantes, cines, parques de ocio, etc).
2. Comparar las valoraciones de un usuario con todas las demás y encontrar las personas que tienen gustos similares.

3. Recomendar elementos que usuarios similares han calificado positivamente, teniendo en cuenta que el usuario al que se le va a realizar la recomendación todavía no haya realizado la actividad.
   Se distinguen dos algoritmos de filtrado colaborativo:

- **Basados en memoria, o algoritmos de vecinos cercanos**[4] [Jannach et al. , 2011]: Utilizan la base de datos completa para generar una predicción. Emplean métodos estadísticos para determinar usuarios con un historial de valoraciones similar al usuario actual (vecinos) y a continuación, mediante un conjunto de algoritmos combinan las preferencias para realizar las recomendaciones.
- **Basados en Modelo** [Galán Niet, 2006] [Albin Rodriguez, 2009]: Primeramente desarrollan un modelo de puntuaciones de los usuarios sobre los ítems. En diferencia a los basados en memoria, no se se utilizan métodos estadísticos, sino una aproximación probabilística. Esta aproximación calcula el valor esperado de una predicción del usuario para cada ítem en función de ratings anteriores. Para este cálculo se utilizan diversos algoritmos de *aprendizaje Clustering* o *redes neuronales* [Nasraoi, 2004] (Por ejemplo, las Redes de Funcionamiento de Base Radial [Isasi Viñuela y Galván Leon, 2004]).

2.1.2. Recomendadores grupales

Los sistemas recomendadores individuales sugieren una serie de productos a un usuario. Sin embargo, no tienen en cuenta su uso dentro de una aplicación en la cual se va a recomendar un mismo producto a un grupo de usuarios, en lugar de a un usuario individual, como sería el caso de los recomendadores de planes de ocio. En estos últimos años es cuando se está empezando a trabajar en el desarrollo de técnicas que permitan proponer recomendaciones simultáneas a grupos de usuarios [Jameson & Smyth, 2007].

De esta necesidad nace un nuevo tipo de sistemas recomendadores, los denominados *recomendadores grupales*. Estos sistemas tratan de recomendar un producto que pueda satisfacer las necesidades de todos los miembros de un grupo [Masthoff, 2011].

[4] Del inglés, Nearest Neighbour

El sistema pueda dar diferentes resultados, dependiendo de la combinación de las preferencias que se utilicen. Por ejemplo:

1. **Mezclar las recomendaciones individuales,** obteniendo un conjunto de recomendaciones individuales de cada miembro del grupo en un único listado.

2. **Tratar de manera diferente a cada miembro del grupo**, prestando especial atención a aquellos usuarios que tienen características diferentes al resto (dar más peso a un usuario que otro).

3. **Crear un modelo de grupo** como si fuera un único usuario, de tal forma que se unen las preferencias de todos los usuarios, se tratan estos datos mediante una fórmula determinada y se obtiene un resultado similar al de un sistema de recomendación individual [Leiva, 2014].

4. **Crear una agregación en la que se valores recomendaciones individuales**, de forma que la recomendación escogida como mejor para el usuario, se valora de forma más alta. Para cada individuo el sistema escoge el conjunto de recomendaciones que tenga la mayor puntuación. Mediante una fórmula escogida se agregan todas las recomendaciones del conjunto final y se determina cuál es la mejor.

Realizando una aproximación de cuál de las distintas posibilidades se adapta mejor a nuestro proyecto, decidimos combinar la agregación de las puntuaciones individuales (método 4) junto con la creación de un modelo de grupo (método 3).

Para crear un modelo de grupo como si fuera un único usuario, se elabora una serie de prototipos de grupo para después escoger los más utilizados por los usuarios. En una primera fase, basamos el descarte de los grupos en base a encuestas realizadas (sección 6.2) a usuarios potenciales de la aplicación. Ofrecemos al usuario la posibilidad de seleccionar un modelo de grupo, una edad para la que va dirigido el plan seguido de una serie de actividades de ocio y el sistema se encarga de recomendar un plan. Esta sección es uno de los principales módulos introducidos y se detalla en la sección 5.2 de esta memoria. De esta manera enriquecemos la recomendación y ofrecemos al usuario dos maneras distintas y complementarias de satisfacer sus preferencias.

Para mezclar las puntuaciones individuales se crea una agregación de las puntuaciones de cada individuo para un conjunto de elementos. Posteriormente, se

realiza una predicción de la puntuación asignada para cada usuario miembro del grupo a cualquier actividad y recomendamos el conjunto de actividades que hayan obtenido el mayor valor en la valoración agregada.

Como hemos introducido anteriormente (sección 1.1), nuestra aplicación tiene dos modos de uso: identificando las credenciales con Facebook y modo invitado. En el modo de inicio de sesión con Facebook, para mezclar las puntuaciones individuales y crear una agregación, utilizamos la función *Trust Weighted Mean* [Madrid Live, 2014], incluyendo el valor de *trust* o confianza para que sea más eficiente a la hora de tener en cuenta factores sociales entre individuos.

El concepto de *trust* [Golbeck, 2006a] [Golbeck, 2006b] responde a la necesidad de incluir factores sociales en la función de agregación. Enriquece dicha función midiendo la confianza o intimidad que tienen dos personas entre sí. Para el cálculo se utiliza un modelo propuesto en una Tesis Doctoral [Quijano Sánchez, 2010] y cuyo valor viene en función de cuatro parámetros adicionales:

1. **Intimacy** (*intimidad*). Pretende medir la intimidad o cercanía de la relación.
2. **Intensity** (*intensidad*). Tiene como objetivo calcular la intensidad de la relación, es decir, si la amistad tiene un contacto diario o esporádico.
3. **Duration** (*duración*). Mide la solidez y estabilidad de la relación en el tiempo.
4. **Reciprocal services** (*entrega mutua*). Pretende estimar la entrega mutua de los componentes de la relación.

Para obtener estos valores, se extrae la información del usuario a través de Facebook (sección 3.2.1). Se almacena la información, se recopila la del resto de usuarios asistentes al plan y se aplica la función de agregación. De esta manera, tenemos perfil social de usuario más específico en función de la información recopilada para el usuario.

Para el modo invitado, debemos combinar la información obtenida en las restricciones impuestas por el usuario a través de la interfaz que proporciona el sistema (tipo de actividad, modelo de grupo, edad a la que va dirigido el plan) para establecer un conjunto de preferencias sobre las actividades. El método utilizado se basa en un filtrado de preferencias (sección 5.2.3.1) en función de los parámetros de

entrada que devuelve un resultado con las actividades que mejor se adaptan al usuario.

2.2. Fuentes de información en los sistemas de recomendación

Los sistemas de recomendación grupales tienen la dificultad de gestionar las técnicas de recomendación mencionadas en apartados anteriores. Acciones tales como adquirir las preferencias del grupo, ayudar al grupo en el proceso de toma de decisiones, o explicar al grupo las razones para elegir una recomendación pueden suponer una dificultad añadida.

La mayoría de estos sistemas desarrolla técnicas de adquisición de información que se aplican en los sistemas recomendadores individuales. Podemos hacer una distinción entre:

1. *Adquisición de preferencias sin especificación explícita*, en el que los usuarios no especifican sus preferencias con anterioridad. Se adquiere la información implícita de estos. Por ejemplo las visitas que realizan a un determinado sistemas y el tiempo que dedican a cada una de ellas.

2. *Especificación de preferencias explícita*, en la cual se requiere una especificación explícita de las preferencias de los usuarios. Por ejemplo, mediante un ratio de puntuación. Amazon utiliza un sistema de valoración en el que la puntuación se representa mediante un ratio de 1 a 5 estrellas (sección 2.3.1).

Tras adquirir la información sobre cada uno de los usuarios, se puede adaptar la especificación de las preferencias individuales a las preferencias de recomendación para el grupo. Esto puede ser más o menos difícil en función del tamaño y heterogeneidad del grupo.

Hay dos posibles opciones que se pueden utilizar para combinar las preferencias. La más habitual es que el sistema escoja la opción que satisfaga al mayor número de usuarios tomando como base sus preferencias individuales. También cabe la posibilidad de seleccionar la siguiente hipótesis: “la felicidad del grupo será igual a la del menos feliz de sus miembros” [Shafer et al., 2007].

Es necesario añadir métodos de agregación para combinar información sobre distintas preferencias individuales, para que el sistema pueda obtener la mejor recomendación para el grupo.

2.3. Ejemplos de sistemas de recomendación

Hoy en día existen numerosos sistemas de recomendación que son utilizados en diferentes ámbitos. A continuación se resumen los más conocidos y sus diferentes aplicaciones.

2.3.1. Sistema de recomendación de Amazon

La tienda virtual de Amazon cuenta con un algoritmo personalizado de recomendación denominado *filtrado colaborativo ítem a ítem*. Este algoritmo, en lugar de buscar similitudes entre usuarios, trata de relacionar los productos adquiridos y calificados por el usuario con otros existentes de características parecidas. Después, combina dichos productos similares en una lista de recomendaciones que se muestra al usuario [Linden et al. 2003].

Este proceso puede llegar a ser extremadamente costoso en términos de computación. Sin embargo, han conseguido implementarlo y ejecutarlo de manera notablemente eficiente, de modo que se comporta aceptablemente bien incluso trabajando con conjuntos de datos enormes.

Amazon cuenta con una base de más de 150 millones de usuarios, y recoge y almacena información de cada uno ellos. Este tipo de sistemas ofrecen resultados más fiables cuanto mayor es la cantidad de datos de los que se dispone, de modo que un producto es más o menos relevante en función de la cantidad de usuarios que lo hayan adquirido.

Este algoritmo falla para los casos de productos cuyo número de compras es pequeño, pues existe una cantidad mucho menor de datos acerca de ellos en relación a otros similares pero que cuentan con un mayor número de adquisiciones.

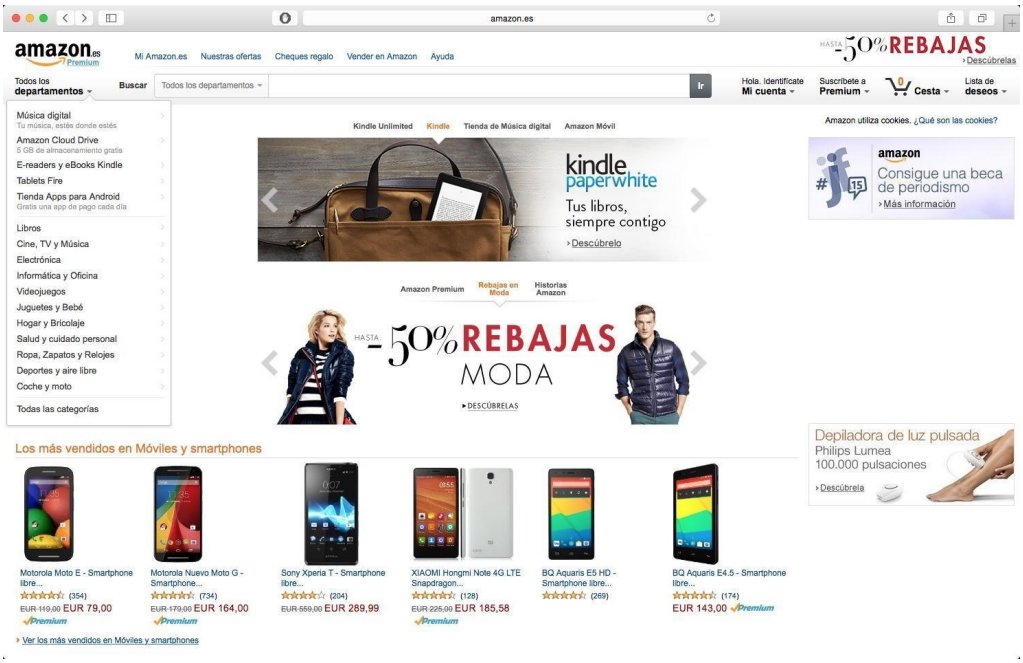


Figura 2.3: Sistema de recomendación Amazon

2.3.2. Sistema de recomendación de Filmaffinity

Filmaffinity es una web de votación y recomendación de películas. El algoritmo que implementa para llevar a cabo sus recomendaciones se denomina *recomendaciones de tus almas gemelas*, que funciona como un sistema de filtrado colaborativo [Filmaffinity, 2015].

A medida que un usuario puntúa películas en la web, el sistema va almacenando dichas votaciones y busca a otros usuarios, o *almas gemelas*, con puntuaciones similares a las tuyas. De este modo, se le mostrarán al usuario aquellas películas que sus *almas gemelas* evalúen positivamente, pues al tener gustos similares, es probable que estas otras películas también sean de su agrado.

Este sistema de recomendación tiene como desventaja que para llevar a cabo unas mejores recomendaciones, es necesario que el usuario evalúe un gran número de películas. Cuantas más películas haya evaluado, más puntos de comparación tendrá con otros usuarios y, por tanto, su *alma gemela* representará más fielmente sus gustos.

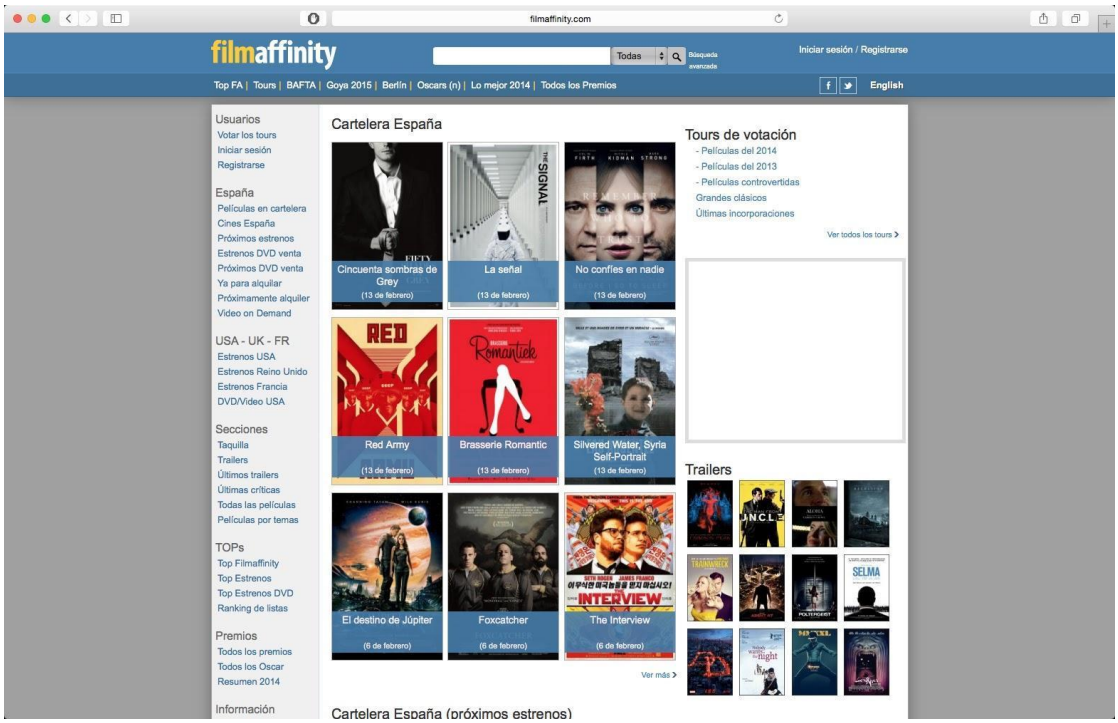


Figura 2.4: Sistema de recomendación FilmAffinity

2.3.3. Sistemas de recomendación de planes de ocio

A continuación queremos realizar un análisis de dos sistemas de recomendación social y de ocio: Whatsred y Fever. Hemos querido escoger para el análisis estos recomendadores ya que al igual que MadridManía son sistemas de recomendación de ocio para la ciudad de Madrid.

2.3.3.1 Sistema de recomendación Fever

Fever es una aplicación móvil de recomendación de planes de ocio para Madrid, Barcelona, Valencia y Nueva York. Se encuentra disponible para las plataformas Android e iOS y ofrece recomendaciones que se actualizan cada semana. Entre los planes que incluye se encuentran conciertos, fiestas, películas, restaurantes y lugares que visitar.

Como requisito indispensable para utilizar la aplicación es necesario que conectemos ésta con una cuenta de Facebook, aspecto que puede favorecer su carácter social pero que quizás no sea del gusto de todos los usuarios. No todo el

mundo que desee utilizar la aplicación tiene que tener necesariamente una cuenta en esta red social o, si la tiene, puede que no se "fíe" de cómo la aplicación va a tratar sus datos.

En cuanto a su funcionamiento, una vez le hemos dado permiso a la aplicación para conectarse con Facebook, se nos muestra un listado de *hashtags* (#cine, #geek, #shopping) de entre los cuales el usuario deberá elegir los tres que mejor lo definan. Es un aspecto a destacar la sencillez de este cuestionario y que no se bombardee al usuario con múltiples tests para realizar las recomendaciones.

Una vez completado este cuestionario, la aplicación genera un perfil con nuestros nombre e imagen de perfil de Facebook, y los tres *hashtags* seleccionados. Además, comenzamos a seguir automáticamente a nuestros amigos de Facebook que también hayan conectado su cuenta con la aplicación (al estilo de Twitter).

Dentro de cada perfil podemos ver la actividad reciente del usuario, incluyendo tanto los planes de ocio a los que acudió, como si comenzó a seguir a otros usuarios.

En cuanto a las recomendaciones, Fever nos ofrece planes patrocinados por empresas que podemos llevar a cabo hoy, mañana o después. Además, nos muestra el número de asistentes a cada uno de estos planes y el precio de la entrada para acudir a ellos. Una gran ventaja es que nos permite adquirir estas entradas desde la misma aplicación.

Los planes que se nos muestran se eligen en función de los *hashtags* que hubiésemos seleccionado al registrarnos con la aplicación. Cada plan tiene también su propio *hashtag*, de modo que si coincide con alguno de los nuestros, se nos muestra el plan.

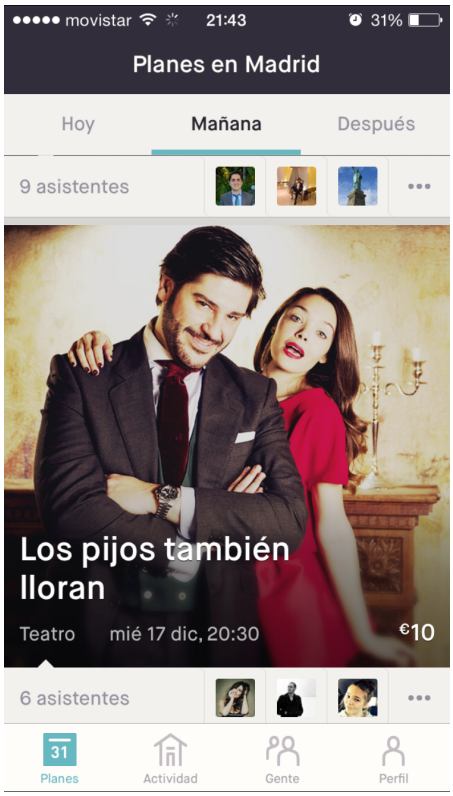


Figura 2.5: Sistema de Recomendación Fever

Puntos positivos:

1. Planes actualizados semanalmente.
2. Introducción mínima de datos de forma directa por parte del usuario.
3. Interacción social.
4. Compra de entradas desde la propia aplicación.
5. Interfaz de usuario atractiva.

Puntos negativos:

1. Obligación de registro con Facebook.
2. Planes de ocio que solo ocupan una parte del día.
3. Falta de categorización de los planes mostrados.
4. Número de planes limitado en base a empresas patrocinadoras de la aplicación.
5. Los planes se centran en la ciudad donde nos encontremos, pero no tienen en cuenta la cercanía frente a la localización actual del usuario.
6. Ideas seleccionadas para aplicar en nuestro sistema

7. Enlazar con alguna web de venta de entradas para los planes ofrecidos, lo que facilita mucho al usuario la tarea de acudir a un plan.
8. Diseño de la interfaz de usuario. Un diseño agradable a la vista hace que los usuarios tengan una mejor experiencia con la aplicación.

2.3.3.2 Sistema de recomendación Whatsred

Whatsred es una aplicación móvil que tiene como fin recomendar planes de ocio, centrándose principalmente en el sector de la hostelería. No es necesario iniciar sesión con Facebook ni estar registrado para utilizar la aplicación. La mayoría de sus planes son actividades tal como desayunos, comidas, cenas, aperitivos, etc. Las recomendaciones se dividen en cinco categorías: *Planes cerca de ti*, *comamos juntos*, *a salir*, *hacer deporte*, y *otros planes*.

Una vez que seleccionamos el plan deseado, se puede consultar por ejemplo datos sobre el establecimiento, el período de validez, los detalles del sitio, la ubicación, y finalmente una opción “vale! lo quiero!”.

En el caso de registrarnos, encontraremos accesibles una serie de opciones que hacen que Whatsred sea una aplicación mucho más personalizada. Podemos personalizar nuestro perfil, indicando nuestras preferencias así como las categorías en las que queremos que nos envíen notificaciones, en qué lugar y en qué fecha. Además ofrece una plataforma de comunicación por medio de mensajes, en la que se pueden hacer valoraciones de los planes categorizando con estrellas y comentarios. Permite igualmente tener una lista de planes que “me gustan”, así como de sitios que “me gustan”.

Figura 2.6: Sistema de Recomendación Whatsred

Puntos positivos

1. Herramienta gratuita.
2. Acceso en modo invitado.
3. Muchos establecimientos hoteleros lo utilizan.
4. Resulta sencillo elegir los planes de ocio.
5. Ubicación manual o automática (se puede ver en mapa).

Puntos negativos

1. Planes limitados (la mayoría son de hostelería).
2. Número de planes restringido en base a empresas patrocinadoras de la aplicación.
3. Muchas opciones no se pueden realizar si no estás registrado.

2.4.4 Conclusión de los sistemas de recomendación

En la actualidad podemos encontrar una gran variedad de sistemas de recomendación. Hemos repasado algunos de los más utilizados, centrándonos especialmente en Fever y Whatsred, ya que están disponibles para Android y proponen planes para grupos.

En el siguiente cuadro resumen (Tabla 2.1) aparece una comparativa de las principales características de estos recomendadores. Para elaborar la tabla se han consultado páginas web de análisis de aplicaciones[5] (como en el caso de Amazon), la lista de opiniones de Google Play para Fever[6] y Whatsred[7] y foros de usuarios que han utilizado el sistema en el caso de FilmAffinity[8].

Recomendador	Función social	Inicio de sesión anónimo	Interfaz de usuario atractiva	Funciones para grupos	Experiencia de usuario
Amazon	★	★★★	★★★	★★	★★
FilmAffinity	★	★★	★	★	★
Fever	★★★	–	★★★	★★	★★
Whatsred	★★	★★	★★	★★	★★

Tabla 2.1: Comparativa Sistemas de Recomendación

[5] Amazon app: Análisis valoración y opiniones del sistema (http://marketing4ecommerce.net/amazon-app-analisis-valoracion-y-opiniones/)
[6] Fever: Play Store (https://play.google.com/store/apps/details?id=com.feverup.fever&hl=es)
[7] Whatsred: (https://play.google.com/store/apps/details?id=com.whatsred.whatsred&hl=es)
[8] FilmAffinity: foro Meristation (http://zonaforo.meristation.com/topic/1371816/)

2.5. Definiciones y tecnologías usadas

Para llevar a cabo el desarrollo de MadridManía se han empleado diversas tecnologías. En este apartado se resumen todas ellas, haciendo especial hincapié en aquellas que consideramos de mayor relevancia.

Figura 2.7: Tecnologías utilizadas

2.5.1. Desarrollo Android

Android es un sistema operativo móvil, propiedad de Google, el cual se encuentra presente en multitud de *smartphones* y *tablets*. Es la interfaz y funciones que poseen muchos de los teléfonos actuales, propiciado por ser de código abierto, y se encuentra respaldado por una enorme comunidad de usuarios desarrolladores que trabajan día a día para mejorarlo.

Con Android el usuario puede trabajar de forma sencilla, y a al ser propiedad de Google, permite agregar servicios de esta empresa de forma muy sencilla, tales como Gmail o Google+. Además de estos servicios, existen SDKs que permiten agregar funcionalidades adicionales como la conexión con redes sociales.

2.5.2. Desarrollo del servidor

A continuación se explican las tecnologías utilizadas para el desarrollo en el servidor:

2.5.2.1. J2EE

Es una plataforma de programación que permite desarrollar y ejecutar software de aplicaciones utilizando el lenguaje Java. Permite utilizar diversas arquitecturas de varias capas distribuidas, apoyándose en componentes software modulares ejecutándose sobre un servidor de aplicaciones.

J2EE cuenta con especificaciones API (JDBC, RMI, JMS) además de Servicios Web y XML y permite interactuar con dichas especificaciones. Además configura algunas especificaciones que permiten trabajar con Enterprise JavaBeans, servlets, portlets (siguiendo la especificación de Portlets Java), Java ServerPage y varias tecnologías de servicios web.

Todas estas características permiten crear una aplicación multiplataforma y escalable, a la vez que integrable con las tecnologías citadas anteriormente. Además el servidor de aplicaciones puede manejar transacciones, la seguridad, escalabilidad, concurrencia y gestión de los componentes desplegados. Esto contribuye a que cualquier desarrollador pueda concentrarse más en la lógica de negocio de los componentes en lugar de en tareas de mantenimiento de bajo nivel.

2.5.2.2. SQL

SQL (Structured Query Language) es una lenguaje de programación declarativo que permite el acceso a base de datos relacionales especificando además diversos tipos de operaciones sobre ellas. Para la gestión de las bases de datos, en MadridManía nos hemos decantado por MySQL, debido que era conocido de otras asignaturas, como Aplicaciones Web.

Este intérprete de lenguaje SQL permite crear bases de datos, tablas, además de operaciones básicas como son la inserción de datos, modificación, eliminación, ordenación y consultas entre otras operaciones, convirtiéndose así en una buena herramienta de administración de nuestra de base de datos.

En la actualidad es un estándar muy utilizado y la mayoría de los sistemas que utilizan bases de datos lo soportan, de modo que el lenguaje que utilizamos para almacenar la información acerca de usuarios y actividades en la base de datos de MadridManía.

2.5.2.3. JSON

JSON (JavaScript Object Notation) es un sistema de intercambio de datos ligero que facilita a las personas la comprensión JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.

La simplicidad de JSON ha dado lugar a la generalización de su uso, especialmente como alternativa a XML en AJAX. Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos en este contexto, es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON.

2.5.2.4. jCOLIBRI

Es una plataforma desarrollada por el grupo GAIA de la FDI[9] en 2005 [Recio García, 2008] con el fin de desarrollar proyectos basados en el razonamiento basado en casos. Diseñado para ser reutilizado por diferentes familias y dominios CBR, ofrece muchos de los elementos necesarios para el desarrollo de estos sistemas. Proporciona al desarrollador de ejemplos necesarios para comprobar el correcto funcionamiento del framework, además de ejemplos en forma de documentación. Además cuenta con 5 estrategias diferentes basadas en métodos de recuperación, con 7 métodos de selección y 30 tipos diferentes de métricas de similitud.

2.6. Conclusiones

En este capítulo se hace un repaso de los diferentes tipos de recomendadores que existen. Se introduce el concepto de ciclo CBR, y se extiende la explicación de los recomendadores grupales, en la que queremos focalizar nuestro proyecto. Posteriormente se procede a describir algunos de los recomendadores más utilizados en la actualidad, para centrarnos en Fever y Whatsred, que son dos de las

[9] Facultad de Informática

aplicaciones de recomendación de planes y ocio más utilizadas en plataforma móvil (Android).

Para tener una mejor aproximación de los puntos que se pretenden incorporar a MadridManía, hemos presentado una lista de puntos a favor y en contra de cada aplicación que converge finalmente en una tabla comparativa de las principales características.

Por último se han descrito las principales tecnologías utilizadas para la realización del proyecto. Destacan jCOLIBRI, Android y JSON.

Capítulo 3

MadridManía

MadridManía es un sistema para recomendar actividades y planes de ocio situados en la ciudad de Madrid, especialmente para grupos. Consta de dos modos de inicio de sesión: mediante la red social Facebook o un modo de acceso libre denominado modo invitado, que permite a cualquier usuario no registrado acceder para solicitar recomendaciones. Dado que es un sistema orientado a grupos, es posible tanto obtener recomendaciones creando un grupo personalizado, o bien a partir de un modelo de grupo. En ambos casos el objetivo es optimizar las recomendaciones y la experiencia del usuario.

La interfaz se ha desarrollado con tecnología Android mediante un cliente nativo (*frontend*) que se conecta a un servidor (*backend*) que atiende las peticiones. El uso de Android permite integrar funcionalidades en el sistema que son útiles a la hora de realizar una recomendación (como puede ser el GPS).

En este capítulo se da una visión completa de la funcionalidad de MadridManía. Además se especifican las fuentes de información con las que se realizan las recomendaciones.

3.1 Fuentes de información

Para poder obtener información que posteriormente es utilizada en el sistema se hace uso de diversas fuentes tanto externas como internas a la aplicación. La información obtenida queda almacenada y se procesan sus datos.
A continuación se describen los tipos de datos y las fuentes de información utilizadas para adquirirlos.

3.1.1 Tipos de datos

- **Datos estáticos:** Estos datos una vez obtenidos, no se modifican con frecuencia. No son cambios que se realicen continuamente. Por ejemplo el caso de restaurantes o centros comerciales, es poco probable que se

actualicen con frecuencia, por tanto esta información no se modifica. Esto no impide que periódicamente se proceda a actualizar la base de datos del sistema para incluir nuevos centros comerciales. Entre los tipos de datos estáticos podemos encontrar los relacionados con museos, restaurantes, parques, paseos, centros comerciales, parques de ocio y hoteles.

- **Datos dinámicos:** En esta categoría encontramos datos susceptibles de ser modificados frecuentemente en el tiempo, encontrando en esta sección los hoteles. Disponibilidad, relación calidad precio, o valoraciones cambian constantemente, y no podemos almacenar la información de manera estática (la disponibilidad de un hotel puede variar de un día a otro).

3.1.2 Catálogos de fuentes de información

A continuación, se describen brevemente las fuentes de información utilizadas actualmente por nuestro sistema:

- **11870[10]:** es una web que permite a los usuarios guardar y compartir sitios y servicios de cualquier parte del mundo mediante opiniones, fotos y vídeos. Cuenta con servicios tanto para usuarios individuales como para empresas, y además dispone de una aplicación móvil en distintas versiones para los principales sistemas operativos de *smartphone*.
  La web presenta un buscador en el que el usuario puede introducir el sitio o ciudad para el cual se desea obtener actividades, y a continuación, filtrar los resultados que se muestran por tipo de actividad (restaurantes, hoteles, tiendas, etc.). Dispone de una gran variedad de opciones disponibles, lo que lo convierte en una muy buena alternativa a la hora de planificar nuestras actividades de ocio para cualquier lugar que vayamos a visitar.

  De cara al desarrollo, su aspecto más útil es que proporciona una sencilla API web a través de la cual pueden realizarse consultas y obtener la información necesaria para MadridManía fácilmente.

[10] 11870: Los mejores sitios de tu ciudad (https://11870.com/)

A través de 11870 obtenemos la información necesaria acerca de los parques, restaurantes, centros comerciales y paseos de Madrid. El caso de estos últimos el proceso difiere un poco del resto de actividades, pues para su elaboración se recupera una lista de monumentos o puntos de interés a partir de la cual se genera un paseo de no más de dos horas de duración uniendo varios de los puntos encontrados.

- **Google Places**[11]**:** es un servicio que Google ofrece a empresas y propietarios permitiéndoles generar fichas de sus negocios (nombre, descripción, imágenes, etc.). Una gran ventaja que presenta este servicio es que los establecimientos inscritos en él figuran también a su vez en Google Maps.

  Google Places ofrece información detallada de segmentación sobre los usuarios que visitan cada una de estas fichas, tales como estadísticas de visualizaciones. Esto puede ser de gran utilidad para los propietarios, ya que les facilitar la tarea de publicitarse a través de Internet.

  Al igual que en el caso de 11870, Google Places también cuenta con una API web de consulta que facilita la recuperación de datos y hace que su inclusión en Madridmanía sea relativamente sencilla. Utilizamos este servicio para la inclusión de los museos en nuestro sistema.

- **Guía del ocio**[12]**:** es una web de consulta acerca de actividades de ocio para toda España. Ofrece un conjunto variado de actividades (restaurantes, teatros, conciertos, etc.), pero al contrario que en los dos servicios anteriores, no cuenta con una API web de consulta. Por esta razón, es necesario realizar un análisis total del código HTML de la página web y recuperar de ésta la información.

  En este caso, solo utilizamos este servicio para la obtención de las carteleras de cine una vez al día, almacenando los datos como una lista de cines disponibles con las sesiones y películas de cada uno, actores, géneros de películas, etc.

[11] Google Places: API (https://developers.google.com/places/documentation/?hl=es)
[12] Guía del ocio: (http://www.guiadelocio.com/)

- **Sistema de recomendación de hoteles HSR[13]:** Para consultar una lista de hoteles cercanos a nuestra posición accedemos a este sistema de recomendación pasándole las coordenadas cartográficas. Principalmente se trata de un recomendador de hoteles para las ciudades más visitadas en España, que permite encontrar hoteles cercanos a la ubicación enviada. Además podemos realizar un análisis estadístico con el fin de ver la reputación de dicho hotel en función de las valoraciones de otros usuarios. Analiza expresiones regulares que permiten detectar entre otros aspectos los servicios que posee o no el hotel buscado. De esta forma es posible detectar si un hotel se adapta a las necesidades del usuario de la forma más objetiva posible.

En la siguiente trama se muestran, a modo de resumen, las fuentes de información utilizadas y qué actividad se obtiene de cada una de ellas.

Actividad	Fuentes de datos	Categoría
Museos	Google Places	Estática
Restaurantes	11870	Estática
Parques	11870	Estática
Paseos	11870	Estática
Cines	Guía del ocio	*Estática*
Centros comerciales	11870	Estática
Parques de ocio	11870	Estática
Hoteles	HSR	Dinámica

Tabla 3.1: Fuentes de información para las actividades

[13] HSR: Sistema de recomendación de Hoteles (Roberto Yánez, Eddie Cerpa) (http://mlh.fdi.ucm.es:8080/HSR/index.html)

3.2 Características de MadridManía

MadridManía es un sistema de recomendación social y contextual para grupos de actividades de ocio para la ciudad de Madrid que cuenta con una interfaz cliente *Android* (*frontend*) que realiza peticiones a un servidor (*backend*) a través de una API (más detalles de la arquitectura en el capítulo 4).

En la aplicación, los usuarios pueden solicitar planes de ocio a través del cliente *Android* y el sistema devolverá los que determine más adecuados para ese usuario en función de las restricciones de recomendación impuestas.

El sistema pretende la combinación de diversas actividades entre las que se incluyen museos, cines, restaurantes, paseos, parques, centros comerciales, parques de ocio y hoteles cuya búsqueda se realiza a partir de una serie de catálogos de actividades obtenidos mediante servicios web.

Para precisar la recomendación, MadridManía genera perfiles de preferencias para cada usuario y modelo de grupo. De esta manera el sistema siempre trata que las recomendaciones generadas encajen lo mejor posible para cada usuario y grupo.

Para la actualización de preferencias, el modo de inicio de sesión con Facebook cuenta con una base de datos que almacena las preferencias de cada usuario y que es posible actualizar tantas veces como se quiera a través de un test. Por lo tanto, sólo es necesario realizar este test si se pretende que el sistema nos recomiende algún tipo de actividad específica, y en ningún caso es necesario efectuar el test cada vez que un nuevo usuario utilice la aplicación. Esta funcionalidad se completa según el modelo de grupo y la edad de los miembros del plan mediante unas reglas de filtrado, que permiten establecer un conjunto de preferencias para cada grupo o usuario, sin necesidad de rellenarlas manualmente. Además, el sistema es capaz de analizar el comportamiento de los usuarios, siendo posible determinar si le ha gustado o no un determinado plan propuesto, mediante una valoración de actividades realizadas. Dado el caso que el sistema haya recomendado a un usuario un museo, centro comercial o restaurante y se hayan valorado de forma negativa, se detecta para posteriormente proceder a precisar estas recomendaciones, de forma que se optimicen.

Con el objetivo de mejorar resultados se hace uso de servicios externos como son la *geolocalización* o información de contexto (parámetros externos que rodean la recomendación). La hora y fecha del sistema, la localización y la detección de amigos cercanos para añadirlos al plan son elementos que permiten aceptar o descartar actividades al recomendar un plan. Para ello se hace uso de la tecnología móvil *Android*, que permite extraer gran cantidad de información de contexto a través del GPS.

En lo que a la solicitud de la recomendación se refiere, con el objetivo de enriquecer lo máximo posible las recomendaciones, una de las principales líneas de desarrollo e investigación ha sido realizar la búsqueda de planes orientados a un grupo. Como se ha comentado anteriormente, se basa en un filtrado de preferencias. A partir de un tipo de plan específico como puede ser familiar, de pareja, trabajo o un grupo de amigos permite obtener las preferencias para las diversas actividades de ocio que seleccione el usuario. Por ejemplo: si solicitamos la búsqueda de actividades para un plan familiar que incluya niños, es poco probable que se integre el contenido de una película de terror o un restaurante de pescado, por las bajas probabilidades de éxito. Sin embargo, si recomendamos una película infantil o de animación y un restaurante familiar como el Tommy Mel's, es casi seguro que la recomendación será satisfactoria para el usuario. Evitamos así tener que reordenar las preferencias cada vez que solicitamos una recomendación para un plan de ocio en grupo, y se puede enriquecer mucho más la recomendación, ya que las funciones de agregación pueden no resultar del todo precisas. Por ejemplo: si agregamos un conjunto de amigos al plan de actividad museo, varios de ellos tienen entre sus preferencias un museo de artes escritas, a la hora de realizar una recomendación, el sistema seleccionará un museo de este tipo, aunque el plan que busquemos sea orientado para niños.

3.3 Nuevas actividades

Teniendo en cuenta la variedad de posibilidades de ocio que ofrece la ciudad de Madrid, MadridManía incluyen nuevas actividades para que el abanico de planes de ocio sea más amplio. Estas actividades las puede seleccionar el usuario a la hora de solicitar una recomendación en la pantalla de recomendación

(sección 3.6.4) Las nuevas actividades son: centros comerciales, parques de ocio y hoteles. La integración de ellas se detalla en la sección 5.3.

3.4 Modelos de grupo

Parte del éxito en las recomendaciones ofrecidas por cualquier sistema orientado a actividades en grupo se basa en la capacidad proponer planes alternativos que cubran las distintas necesidades surgidas en el contexto social de sus componentes. Para ello MadridMania sugiere una lista de modelos grupales en los que se encuentran cubiertas todas las necesidades que pudieran surgir a esos usuarios. Hemos tratado de incidir en la búsqueda del más amplio espectro posible, y por tanto, reflejar el máximo número de individuos insistiendo prioritariamente en sus preferencias o necesidades. Todo ello unido a simplificar lo más posible la interfaz de usuario.

Intentando optimizar al máximo este sistema, y en base al análisis realizado en la fase inicial del proyecto (sección 6.2), concluimos que los modelos de grupo que debían integrar la aplicación son: *Familia, Parejas, Trabajo y Amigos*. Básicamente en estos cuatro grupos están incluidos la mayoría de planes que abordan los usuarios cuando realizan actividades en grupo. Además de estos modelos de grupo, era necesario incluir un perfil en el que estén reflejadas el mayor número posible de etapas sociales, mediante rangos concretos de edades que distingan con claridad las necesidades de cada una de dichas etapas. Llegando pues al resultado de que *niños*, *jóvenes (18 a 25)*, *jóvenes (25 a 30)* y *adultos* reúnen los requisitos y necesidades detectadas por los usuarios de la aplicación. Por lo tanto, contamos con dos nuevos parámetros para solicitar una recomendación: modelo de grupo y edad. Estos parámetros son seleccionados por el usuario y vienen propuestos por el sistema (sección 5.2).

Por lo tanto, mediante los modelos de grupo y la caracterización por edades en rangos bien diferenciados, se enriquece el sistema de recomendación y se dota a la aplicación de la autonomía necesaria para que recomiende un plan de ocio útil, sencillo y satisfactorio.

3.5 El contexto y perfil de usuario

Una de las características ofrecidas por cualquier sistema de recomendación es la capacidad de extraer información útil acerca del usuario y sus preferencias con el fin de que la recomendación sea lo más acertada posible. Para ello, se elaboran perfiles de preferencias de usuario y grupo, que a partir de la información introducida en el dispositivo móvil del usuario, puede hacer uso de métodos que completan la información necesaria demandada en cada momento y que puede ser utilizada a la hora de realizar la recomendación. La recogida de toda esta información se puede dividir en tres categorías:

3.5.1 Datos estáticos

Esta información que cambia en el tiempo con poca frecuencia suscita datos que no suelen ser modificados. Por tanto en esta categoría se incluyen por ejemplo las preferencias sobre museos, restaurantes, centros comerciales, etc.

3.5.2 Contexto social

Una de las grandes ventajas que supone que el usuario forme parte de red una social, es obtener gran información que posteriormente es utilizada para mejorar la recomendación. De la red social (Facebook) se extrae la interacción de los componentes del grupo y los amigos del usuario. Además como el sistema cuenta con los modelos de grupo, optimizamos el rendimiento del recomendador combinando las ventajas que supone utilizar la información obtenida en la red social, con las posibilidades que ofrecen dichos modelos de grupo, facilitando, además, que al realizar la recomendación para un grupo no sea necesario interactuar con ninguna red social.

3.5.3 Datos dinámicos

En esta categoría se incluye la información que es susceptible de cambiar rápidamente. Por tanto, dicha información tiene que ser actualizada frecuentemente. Por ejemplo: la ubicación del usuario o la hora del día son elementos importantes que tienen que ser actualizados con asiduidad.

3.6 Descripción de la aplicación

A continuación, tras haber revisado la funcionalidad básica de la aplicación, características, fuentes de información, modelos de grupo e información de contexto, se detalla la interfaz y funcionalidad de aplicación . Así se podrá tener una visión general de cuál es el proceso desde que se inicia la aplicación hasta que se recibe un plan tras solicitar la recomendación.

3.6.1 Pantalla inicial

Cuando abrimos la aplicación se nos muestra una pantalla inicial que nos da acceso a los dos diferentes métodos utilizados en MadridManía. Para ello, el usuario puede pulsar el botón “Iniciar sesión”, que dará acceso al recomendador utilizando la red social Facebook; existiendo otro botón “Modo invitado” con el que se permitirá al usuario acceder sin tener que comenzar sesión.

Figura 3.1: Pantalla inicial de MadridManía

3.6.1.1 Inicio de sesión desde la red social

La API de Facebook proporciona una pantalla de inicio de sesión que solicita las credenciales de acceso. Una vez han sido verificadas, se crea una nueva sesión para el usuario en el servidor. Además almacena los datos de la sesión en la aplicación *Android*.

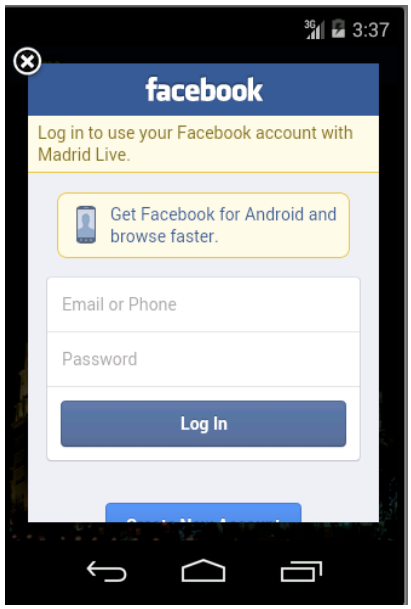


Figura 3.2: Pantalla credenciales Facebook

3.6.1.2 Acceso desde el modo invitado

Sin necesidad de realizar "Login", se crea una nueva sesión para el usuario en el servidor. Además permite usar el resto de funcionalidades de la aplicación como son el sistema de recomendación para grupos o la geolocalización.

3.6.2 Menú principal

El menú principal permite acceder a las diferentes pantallas de la aplicación. Dependiendo de si se trata del modo invitado o el usuario inicia sesión con Facebook, la aplicación permite completar diferentes acciones (como es el caso de cerrar sesión que tiene efecto una vez tengamos una sesión iniciada activa).

Figura 3.3: Menú de la aplicación

3.6.3 Sistema de geolocalización

El sistema de geolocalización requiere de un dispositivo con conexión GPS, además de que el usuario esté conectado a Internet o alguna red. Se activa al iniciar una sesión (tanto en modo invitado como al realizar "login"), recoge la posición del usuario y se encarga de sincronizarla con el servidor. A la hora de realizar una recomendación, permite seleccionar si queremos utilizar nuestra posición, o la queremos seleccionar manualmente, determinando la posición exacta del usuario y sus amigos (Facebook). Por supuesto, esta funcionalidad es totalmente configurable, y permite al usuario activarla o desactivarla en cualquier momento mediante la barra de menú lateral (Figura).

3.6.4 Pantalla de recomendación

En esta pantalla podemos encontrar tres puntos principales que constituyen los principales parámetros a la hora de solicitar una petición al servidor.

1. **Actividades:** Podemos seleccionar un conjunto de actividades al realizar un plan.
2. **Opciones de selección de contexto:** En este punto podemos especificar al sistema si queremos un plan para ahora o más adelante (horario) y seleccionar opciones de geolocalización.

3. **Opciones de selección de grupos:** Permite al usuario seleccionar el tipo de plan que desea (*Familiar*, *Pareja*, *Trabajo* o *Amigos*) y la edad para la cual van dirigidas las actividades del plan (*niños*, *jóvenes de 18 a 25*, *jóvenes de 25 a 30* y *adultos*).

Figura 3.4: Selección de actividades

3.6.4.1 Sistema de recomendación

Una vez que el usuario introduzca los parámetros de la recomendación (actividades, información de contexto, grupo social y edad), se envía al servidor esta información y haciendo uso del sistema de recomendación basado en plantillas devuelve al usuario un plan de ocio. Este proceso aplicado a las nuevas actividades que hemos añadido se describe en el capítulo 5 (sección 5.3.3).

3.6.4.2 Recomendación para grupos

En lo que se refiere al proceso de selección de grupos, se da acceso al usuario a seleccionar las opciones de guardar, cargar grupos personalizados o elegir un modelo de grupo según el plan deseado. La descripción de las tres opciones es la que sigue a continuación:

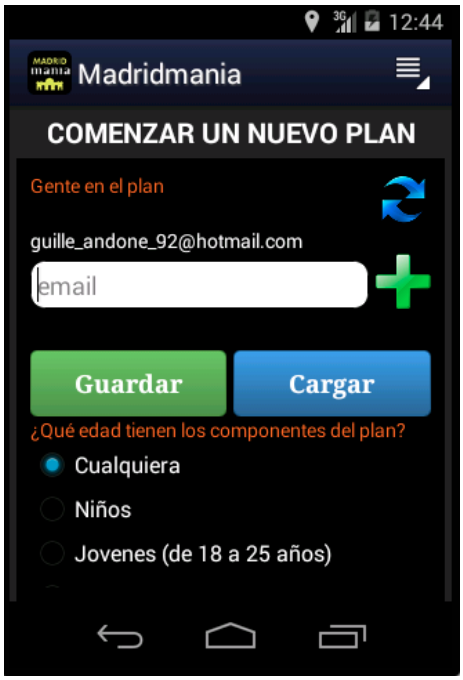


Figura 3.5: Creación de grupos personalizados

1. **Amigos en el plan:** Se detectan todos los amigos cercanos y se agregan al plan, pudiendo el usuarios agregar o eliminar.
2. **Grupos personalizados:** Permite al usuario gestionar y cargar grupos de amigos ya creados para llevar a cabo el plan (Figura 4.5).
3. **Modelos de grupo:** Permite seleccionar un modelo de grupo entre una lista que ofrece el sistema (Figura 4.6).
4. **Edades de grupo:** Da lugar a escoger la edad a la que va dirigida el plan de ocio (Figura 4.7).

Hay que tener en cuenta que los dos primeros apartados no están disponibles en el acceso a modo invitado.

Figura 3.6: Selección de modelo de grupo

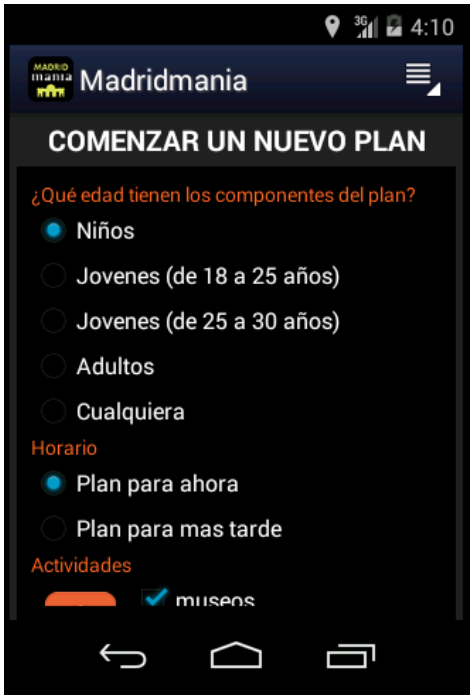


Figura 3.7: Selección de edad

3.6.5 Resultado de la recomendación

Toda vez que el sistema devuelve un plan de ocio con una recomendación, se muestra en una pantalla la lista de actividades con información sobre cada una de ellas (*nombre, dirección, descripción y horario*). La recomendación ofrecida por MadridManía puede ser aceptada o denegada por el usuario.

Figura 3.8: Resultado de la recomendación

3.6.6 Valoración de actividades

Después de que el usuario haya aceptado o rechazado el plan aparecerá una pantalla para valorar dichas actividades según el criterio de dicho usuario. Esta valoración (de 1 a 5 puntos) sirve de orientación para futuras recomendaciones para este (en el caso de Facebook) y otros usuarios. Una de las grandes utilidades del método de valoración, es que para el modo invitado, aunque no tengamos datos del usuario (por tanto sin capacidad para ver si le hemos recomendado esa actividad previamente) permite retroalimentar el sistema de una forma provechosa para otros usuarios, utilizando la puntuación de esas actividades con posterioridad.

Conclusiones

Durante este capítulo hemos procurado describir con detalle cómo funciona MadridMania haciendo un recorrido a lo largo de los catálogos y fuentes de información escogidas, así como el sistema de recomendación y la parte *frontend* (implementada como aplicación Android).

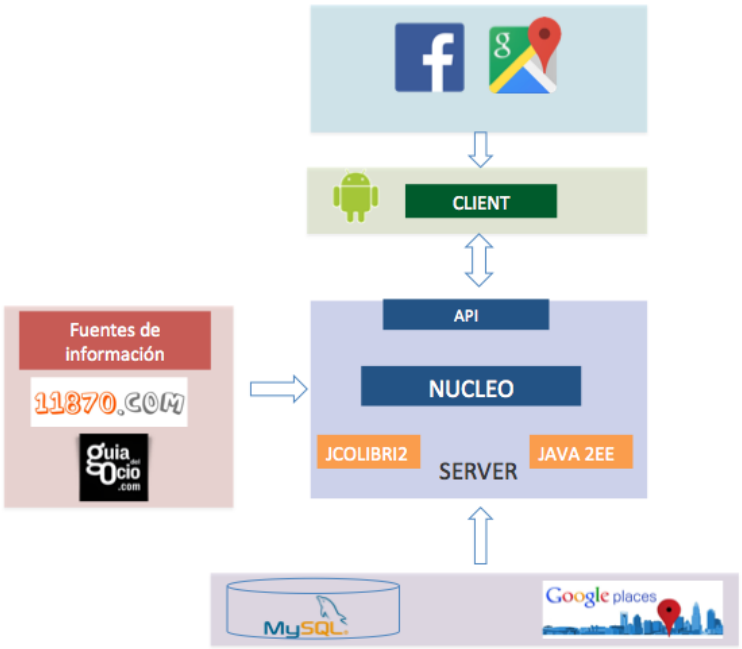
También se ha determinado la funcionalidad de los modelos de grupo, la gestión del contexto físico (geolocalización) y social así como su utilización dentro del sistema.

Capítulo 4

Arquitectura del sistema

Para la arquitectura de MadridManía nos hemos decantado por una estructura que responde al modelo *frontend*/*backend*, en el que los clientes Android realizan peticiones al servidor a través de una API. El *backend* se constituye de dos módulos: API y núcleo. Los componentes básicos se presentan a continuación a modo de resumen gráfico, acompañado de una breve descripción:

1. **Núcleo:** Es el módulo principal de MadridManía, que contiene los componentes necesarios para el correcto funcionamiento de la aplicación y el sistema. Principalmente permite el acceso y la gestión de la información que proporciona la base de datos o servicios externos (17870.com, Google Places, Guía del Ocio, Sistema de Recomendación de hoteles HSR).



4.1 Estructura de la aplicación

2. **API:** Esta capa conecta los clientes (aplicaciones Android) y el núcleo del servidor. Está implementada en J2EE (por sencillez y multitud de

funcionalidades que ofrece) y permite realizar multitud de llamadas a los servidores.Para explicar su funcionamiento con el sistema, podemos decir que la API actúa de intermediario entre los clientes y el núcleo.

A continuación se muestra un gráfico a modo resumen de la interacción de la API con el sistema:

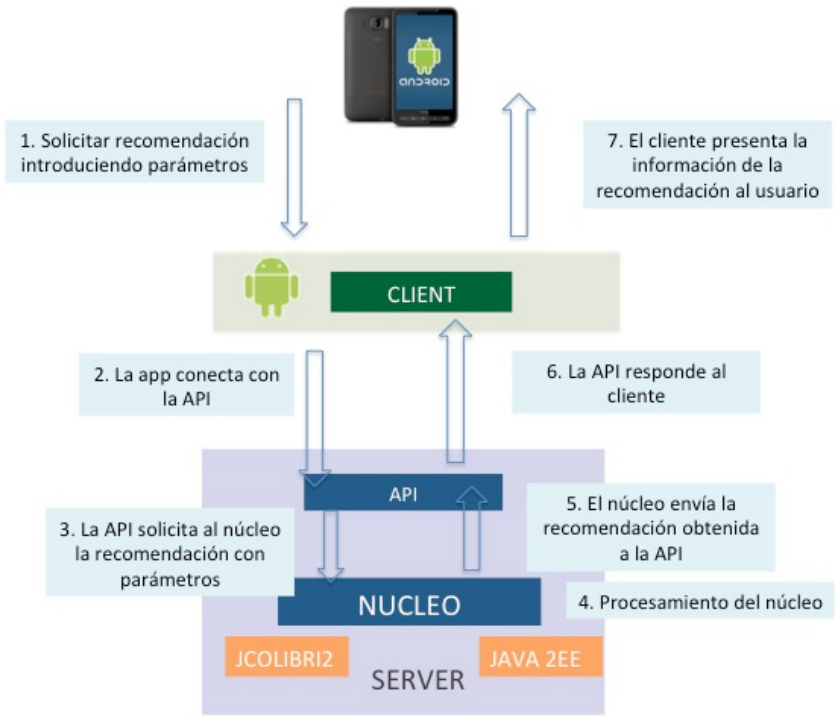


Figura 4.2: Interacción de la API con el sistema

Cuando el usuario solicita una recomendación introduce una serie de preferencias que se traducen en parámetros de búsqueda. Es en este punto donde el cliente conecta con la API y envía una petición a dicha API. Ésta solicita al núcleo que le devuelva la recomendación con los parámetros que el usuario ha introducido. Tras el procesamiento por parte del núcleo de la petición, se envía la recomendación obtenida a la API, que responde al cliente, presentado finalmente los datos obtenidos en el proceso al usuario de la aplicación.

3. **Cliente:** Es el frontend de MadridManía. Se trata de una aplicación desarrollada en Android y permite al usuario interaccionar con el sistema mediante una interfaz gráfica. Recopila las preferencias, acciones y valoraciones del usuario para mostrarle los resultados de las diferentes operaciones.

Para tener una mejor interacción, el cliente Android recibe información de servicios adicionales como la red social Facebook (a través de la cual el usuario accede a la aplicación mediante el método de usuarios registrados) o Google Maps (empleado para ofrecer funcionalidad relacionada con la geolocalización).

4.1. Núcleo del sistema

En esta sección abordaremos todos los detalles concretos de implementación de los componentes internos del backend, como pueden ser la gestión de la base de datos o el sistema de recomendación. Este núcleo, que está desarrollado completamente en Java y forma parte de un proyecto completamente independiente, se encarga de todo el procesamiento y gestión de datos y contiene todas las funcionalidades que necesitará la capa de servicios (API).

A continuación vamos a profundizar un poco más en los aspectos más relevantes de esta parte del sistema:

4.1.1. Gestión de la base de datos

Para que MadridManía sea capaz de realizar las recomendaciones correctamente necesita almacenar un conjunto de datos que reutilizará en futuras recomendaciones. Para ello se establece una conexión entre el núcleo y base de datos MySQL en la que están implementadas las distintas tablas de almacenamiento de datos (veáse .

Para el diseño de esta parte del núcleo se ha diseñado una clase por cada tabla de la base de datos exceptuando las tablas para plantilla, plantilla-actividad, plantilla-usuario, controladas desde la clase *PlantillasSQL.java*. Esta clase contiene las funciones necesarias para realizar las operaciones básicas sobre tablas (inserción, modificación o recuperación de registros). Estos métodos son llamados desde otros puntos de la aplicación, por lo que ofrecen una capa entre la base de datos y el núcleo para hacer más sencillas las consultas.

La clase *MySQLConnector.java* contiene los datos de la conexión a la base de datos (usuario, servidor y puerto) y proporciona la conexión entre cada tabla y su

entrada en la base de datos. De esta manera, todas las clases tienen como atributos al menos un elemento de la conexión y el objeto *Conection*, para realizar las respectivas consultas SQL. Dicho objeto se obtiene a partir de la instancia *MySQLConnector*.

Como en la mayoría de casos que se utilizan estas instancias, se utiliza el patrón Singleton para la clase *MySQLConnector*, que gestiona la base de datos. Como siempre que utilizamos este patrón, queremos evitar la existencia de múltiples conexiones a la base de datos, pudiéndose quedar ésta bloqueada. Con el uso del Singleton conseguimos diseñar y utilizar la clase de manera que sólo exista un único objeto de esa clase durante toda la ejecución. Es decir, un objeto *MySQLConnector*, y éste se llama en la primera referencia a la clase. Al crear más referencias, se devuelve la misma instancia que se creó en la primera llamada.

Este planteamiento lógicamente conlleva a que desarrollar un método estático llamado initConnection, que devuelve la conexión a la base de datos que se ha especificado cuando se instancia el objecto *MySQLConnector*.

4.1.2. Recomendador

MadridManía es un recomendador basado en plantillas de actividades con el que se pretende obtener una recomendación que se adapte a las necesidades de cada usuario. Para su composición se utiliza el concepto de plantillas abstractas y concretas (ejemplo en sección 5.3.3).

El proceso a grandes rasgos es el siguiente:

Se toma una plantilla abstracta creada por el sistema o usuario indicando en ella los tipos de actividades que desea incluir en el plan, el horario y localización en el que va a transcurrir dicha plantilla. Por ejemplo una plantilla abstracta se compone de museos, restaurantes, paseos y centros comerciales

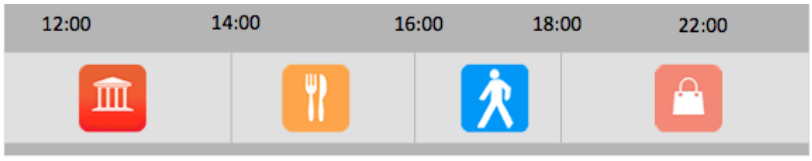


Figura 4.3: Ejemplo de plantilla abstracta

Una vez que tenemos la plantilla, a la que llamamos *Plantilla Abstracta* (Figura) la aplicación genera distintas actividades concretas asignadas a cada tipo de actividad abstracta. Dado el caso de la plantilla anterior, podría proponer "El Museo del Prado" como Museo, "KFC" como Restaurante, "Madrid RIO" como Paseo e "Islazul" como centro comercial.

En este momento, cuando una Plantilla Abstracta tiene actividades concretas pasa a llamarse Plantilla Concreta (Figura)

Figura 4.4: Ejemplo de plantilla concreta

4.1.2. Base de datos local

En general, las aplicaciones Android cuentan con una base de datos local SQLite que mantiene en todo momento información necesaria para el uso de la aplicación. El principal motivo de mantener esta base de datos y no realizar conexiones a servidor (conexión con la capa de servicios) es acceder a información que se consulta frecuentemente. De esta manera se optimiza el rendimiento de las búsquedas, obteniendo datos con mucha rapidez.

4.2. Conexión con la red Social

En esta sección se explica la conexión que realiza el sistema con la red social, ya que a lo largo de la memoria (por ejemplo la sección 5.1.2) detallamos la interacción del cliente con el servidor, incluyendo los métodos necesarios para realizar la conexión.

Para la interacción social, MadridManía se integra con un servicio externo que provee de datos acerca de los usuarios y su entorno social. Lo más apropiado es utilizar una red social para realizar la integración. Aunque se ha optado por Facebook, no habría ningún problema en utilizar cualquier otro servicio, ya que la arquitectura de MadridManía está pensada para trabajar con cualquier servicio externo o red social.

A continuación se describe brevemente la estructura del servicio para gestionar la conexión con la red social Facebook:

- **ConectorSocial.java:** Interfaz que contiene los métodos genéricos que puede usar el cliente para trabajar con la red social.
  Algunos de los métodos definidos en esta interfaz son *login()*, *logout()*, *comprobarSesion()* que comprueba si las sesiones están activas o *sincronizarDatosRedSocial()* que envía información en segundo plano sobre la red social y la envía al servidor.

- **ConectorFacebook.java:** Implementa los métodos del interfaz anterior. Por sencillez se integra el SDK de Facebook para realizar llamadas a métodos del propio SDK. Por ejemplo para el inicio de sesión (método *login()*) se utiliza el propio método que proporciona el SDK, que muestra el formulario, solicita, almacena y procesa las credenciales de inicio de sesión.
  Dado el caso que se ha mencionado anteriormente, si decidiésemos cambiar el servicio o red social, deberíamos crearnos una nueva clase ConectorNuevo.java que implemente el interfaz ConectorRedSocial.java. De igual forma, completaríamos la clase ConectorNuevo.java con la integración del SDK que proveyera de los métodos que necesitáramos implementar.

- **ConectorFacebook.java:** Clase Factoría que mantiene una única instancia activa en el sistema del conector de la red social (de tipo *ConectorRedSocial*) y proporciona un método estático para acceder a la misma desde todo el cliente.

4.3. Cliente Android

En esta sección se trata el Cliente de Android desde el punto de vista del funcionamiento, la gestión de la base de datos, y la conexión con el servidor. Se detallarán los módulos necesarios para la comprensión, sin entrar en detalles completos de la implementación de cada actividad.

4.3.1 Actividades

Una aplicación Android se compone de una serie de Actividades. Cada una de ellas ofrecen al usuario una serie de funcionalidades (ventana de inicio de sesión mediante login o modo invitado, solicitar la recomendación de un plan...). Podemos encontrarlas en el paquete *com.ssii.madridlive* del proyecto MadridManía (frontend). En dicho paquete están incluidas las nuevas actividades implementadas (*RecomendacionInvitadoActivity.java*,*PrincipalActivityInvitado.java*, etc). Todas ellas heredan de la clase *GeneralActivity.java*, ya que presentaban un conjunto de características comunes (como un botón para navegar entre actividades, o una estructura semejante en los layouts).

Se han añadido opciones de control de sesiones para diferenciar la sesión en modo invitado (sección 5.1), y se ha optimizado la interfaz de la aplicación para una mejor experiencia de usuario (sección 3.6).

Capítulo 5

Diseño de los subsistemas.

A lo largo del proyecto se han introducido distintas funcionalidades para completar el sistema precedente (mejora de interfaz, optimizaciones de código, etc). Además se han incluido módulos o subsistemas que aumentan significativamente la versatilidad y funcionalidad de la aplicación.

Este capítulo versará sobre los diversos susbsistemas que constituyen las más importantes novedades respecto a Madrid Live: *modo invitado*, *modelos de grupo* y *el módulo de recomendación para nuevas actividades*.

5.1 Subsistema: modo invitado

El modo invitado permite utilizar el sistema de un usuario que no esté registrado, pudiendo disfrutar de las ventajas del sistema de recomendación (obtención de un plan de ocio que satisfaga las necesidades personales y sociales del usuario).

5.1.1. Objetivo del subsistema

El objetivo de incluir un modo invitado en el sistema, es poder hacer uso de la aplicación sin tener que utilizar Facebook u otra red social obligatoriamente. Queremos dotar al sistema de la capacidad de realizar una recomendación rápida y sencilla para actividades, en un espacio y tiempo determinados.

Además, de la misma manera que se ha podido adaptar el sistema que anteriormente conectaba con Facebook, se pretende acoplar de forma independiente el funcionamiento en modo invitado con cualquier otra red social en un futuro para obtener más información (Twitter, Linkedin…) sin necesidad de realizar un cambio específico para cada subsistema.

5.1.2 Vista de datos y diseño del subsistema

En esta sección englobamos de forma conceptual y concreta la interacción del subsistema con el resto de componentes, desde que el usuario inicia la aplicación hasta que recibe una recomendación (la descripción desde el punto de vista de la arquitectura se ha tratado previamente en el capítulo 4).

En primer lugar, después de que el usuario seleccione la opción "modo invitado" en la pantalla inicial, en la clase *LoginActivity.java* se procesa la solicitud, se comprueba si existe alguna sesión activa, y se lanza la actividad *PrincipalActivityInvitado.java*. En el caso de que quisiéramos aumentar la funcionalidad mediante nuevas actividades o la inclusión de otras redes sociales, hemos desarrollado un conjunto de clases específicas, para que los parámetros de conexión sean independientes de la implementación (*user* y *password* del usuario, identificador del móvil). Otra ventaja es que a la hora de diseñar nuevas pantallas de la aplicación, los *layout* generados sean de fácil adaptación. Una vez en la pantalla principal, se realiza una petición al servidor mediante el método *ConectarListaRecomendaciones()* que devuelve una lista de los planes mejor valorados y más visitados por los usuarios. Pulsando el botón "Recomiéndame un plan" finaliza la actividad, y obtenemos la nueva pantalla para realizar la recomendación. Dicha pantalla contiene el formulario para completar nuestras preferencias de una forma rápida y sencilla. A partir de las restricciones impuestas por el usuario sobre el plan que desea realizar, la clase *RecomendacionInvitadoActivity.java* inicia una conexión con el servidor mediante el método *ConectarSolicitudRecomendación.java* enviando los parámetros de recomendación (actividades, modelos de grupo, edades e información de contexto) y recibiendo una lista de resultados. Estos resultados se devuelven en una nueva pantalla, mostrando una vista del plan recomendado (nombre de la actividad, descripción, dirección, horario). Para finalizar el proceso, la clase *ResultadoRecomendadorInvitadoActivity.java* permite la valoración del plan (aceptar o rechazar) mediante dos métodos que retroalimentan el sistema actualizando la base de datos con información útil para otros usuarios.

A continuación se presenta la vista del subsistema de forma esquematizada, que describe el comportamiento de MadridManía para el modo invitado.

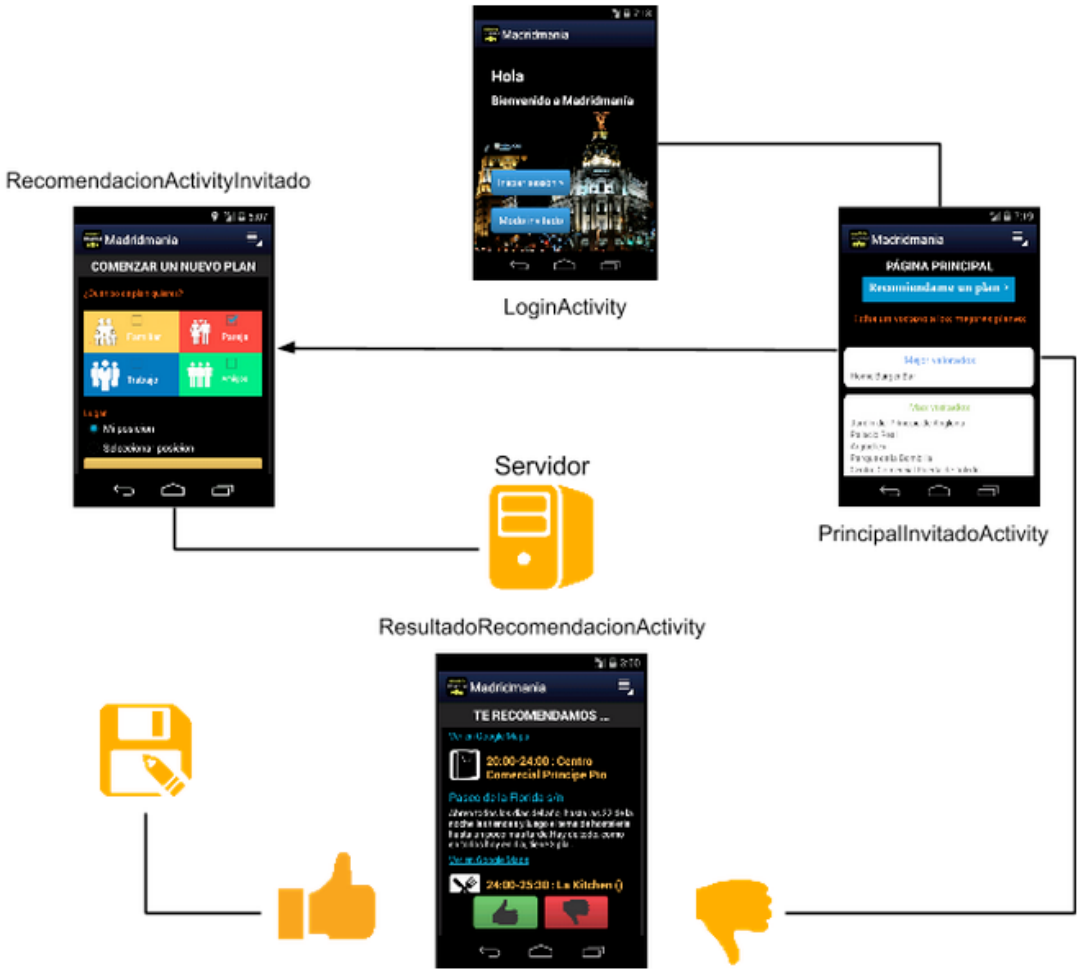


Figura 5.1: Esquema de diseño y vista de datos del subsistema del modo invitado

5.2. Subsistema: Modelos de grupo

Los modelos de grupo permiten obtener un conjunto de preferencias necesarias para la recomendación, y posteriormente, devuelven al usuario un plan de ocio sin tener que realizar un test inicial.

5.2.1. Objetivo de uso del subsistema

Además del modo invitado, uno de los puntos en los que queremos incidir es en que MadridManía mejore la experiencia del usuario en lo referente a actividades realizadas en grupo. Por ello, para satisfacer las necesidades demandadas por los usuarios, hemos elaborado una categorización por grupos y

edades, que pretende realizar una recomendación ajustada a un modelo de grupo concreto y de una edad determinada.

5.2.2. Modelo de grupo

La interacción de los usuarios en entornos de ocio es diferente en función del entorno donde se encuentren, ya que deben adaptar sus interacciones a las características y capacidades de las herramientas disponibles. Además, como se ha dicho anteriormente, en los recomendadores sociales y de ocio que utilizan Internet es importante asegurarse que tanto las actividades que se proponen a los usuarios como las herramientas colaborativas que se les facilitan se hayan sido diseñado de acuerdo a las necesidades de éstos y que se sientan cómodos mientras se encuentran interactuando dichos entornos [Soller, 07].

Para poder adaptar las actividades colaborativas a los usuarios de un grupo social, es necesario almacenar información sobre los propios grupos. Las características relacionadas con los grupos sociales constituirán el modelo de grupo. La información relevante que se puede almacenar dentro de los modelos de grupo, comprende información sobre los propios miembros de los grupos y los roles que tendrán asignados (en caso de estar especificados), así como las actividades repartidas a cada grupo de ocio, resultados obtenidos en actividades colaborativas ya realizadas, opiniones de los usuarios sobre experiencias previas, número de contribuciones o restricciones temporales entre actividades, entre otros.

También se puede incluir información dinámica relacionada con el conocimiento del grupo y con el de cada uno de los individuos dentro de los grupos de ocio. Para crear un modelo o grupo equilibrado, según Barros [Barros, 00] hay que tener en cuenta la evolución del número de contribuciones de un usuario o del grupo, el tamaño de las contribuciones, el grado de interactividad frente al resto de los miembros de un grupo, la evolución de la discusión, etc.

Un grupo social es un sistema formado por un conjunto de individuos que desempeñan roles recíprocos dentro de la sociedad [Turner, J. C. ,1982]. Las personas dentro de él actúan de acuerdo con unas mismas normas, valores y objetivos acordados y necesarios para el bien común del grupo y la consecución de sus fines. Se puede definir a partir de una serie de variables mensurables en el nivel económico, laboral, educativo, social, de ocio, etc.

5.2.2.1 Descripción de modelos de grupo

En MadridManía se han especificado diferentes perfiles sociales bien diferenciados, y que llamaremos modelos de grupo sociales.

Existen 4 modelos de grupo sociales, en función de los cuales, para cada actividad, se selecciona un conjunto de preferencias que satisfacen las restricciones del grupo y que hemos podido extraer del experimento inicial (sección 6.2):

> **Familiar:** Se incluyen en este modelo las posibles relaciones entre familiares (paternofiliales, primos…).
>
> **Pareja:** En un alto porcentaje de casos, cuando buscamos plan es para disfrutarlo con nuestra pareja (una tarde en el cine acompañado de una buena cena es uno de los planes más solicitados).
>
> **Trabajo:** Otra de las recomendaciones más demandadas es la relación laboral entre compañeros(cenas de empresa, comidas de trabajo…).
>
> **Amigos:** Es el grupo en el que está incluido el más amplio segmento poblacional (amigos de la universidad, de la infancia...).

Otro de los rasgos más importante a la hora de realizar una recomendación en grupo es diferenciar para qué edad va dirigido el plan. Por eso establecemos un rango por edades. Así podemos diferenciar dentro de un mismo modelo de grupo si la actividad va dirigida a un niño, un joven o un adulto. Para lo cual hemos consignado la siguiente clasificación:

> **Niños:** Rango de edad que incluye actividades muy concretas y específicas. (películas infantiles, hamburgueserías, parques de atracciones...)
>
> **Jóvenes:** Aquí se establecen dos caracteres bien diferenciados
>
> > **18 a 25 años:** Usuarios que salen y “se mueven” más. Con unos gustos más inclinados al divertimento.
> >
> > **25 a 30 años:** Segmento en el que se incluye un tipo de joven con unas necesidades que responden a una diferente interpretación de las actividades lúdicas.

Adultos: Población muy activa en el posible uso de la aplicación, debido fundamentalmente a sus necesidades sociales o familiares (visitas museo, restaurantes, centros comerciales...).

Cualquiera: Como requisito fundamental no queremos limitar el uso de la aplicación. Al no seleccionar ninguna edad determinada, teniendo en cuenta otros factores para la recomendación.

5.2.3 Vista de datos y diseño del subsistema

Seguidamente en esta sección queremos definir desde un punto de vista técnico el funcionamiento de este módulo. Inicialmente se recogen los datos introducidos por el usuario en la aplicación cliente. Estos parámetros constituidos por un *conjunto de actividades*, *información de contexto* (geolocalización y horario), un *modelo de grupo* y *edad* son enviados (método *`solicitarRecomendacion()`*) al servidor. Posteriormente se procesa la información obtenida en la clase *`RecomendadorServlet.java`* y en el método `doPost` (que procesa las peticiones del cliente en el servidor) se aplica el algoritmo de filtrado siguiente:

Algoritmo de filtrado para modelos de grupo

1. *Filtrado por actividad: Selecciona el conjunto de actividades que forman parte del plan.*
2. *Filtrado por modelo de grupo: Determina cuál es el modelo de grupo escogido por el usuario.*
3. *Filtrado por edad: Selecciona la edad para la que va destinada la actividad de ocio.*

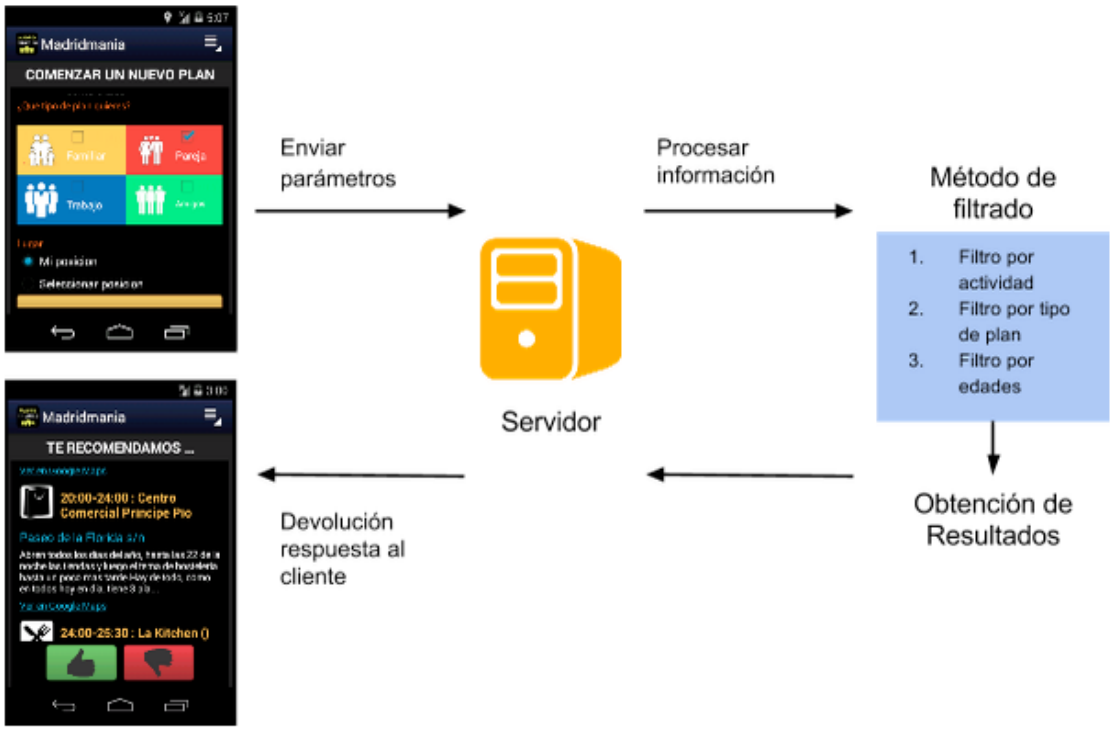


Figura 5.2: Esquema de diseño y vista de datos del subsistema -modelos de grupo

Una finalizado el proceso de filtrado, se obtiene una lista de resultados que se devuelve al cliente y se presenta al usuario.

5.2.3.1 Filtrado de preferencias

Este algoritmo de filtrado de preferencias tiene como objetivo seleccionar, dado una actividad en la que se almacena un conjunto de datos (un caso podría ser el tipo de película para un cine), la que mejor podría cumplir los requisitos para adaptarse a un usuario. Continuando con el ejemplo anterior, en el caso de la actividad “cines”, para un tipo de grupo “familiar”, el sistema seleccionará películas con categoría “familiar”. Si además añadimos información sobre la edad, el sistema determinará con mayor precisión las preferencias para esa solicitud. Siguiendo en la línea anterior, si añadimos el parámetro “edad”, además de películas familiares, nos recomendará de tipo infantil, animación o ciencia ficción.

Ejemplo anterior con aplicación del algoritmo:

1. *Filtrado por actividad: Dado un conjunto de actividades (museos, cines y centros comerciales) se selecciona cada actividad para poder ser filtrada. En nuestro caso y siguiendo el ejemplo, seleccionamos "cines".*
2. *Filtrado por modelo de grupo: Se selecciona el modelo de grupo escogido por el usuario. En este caso "familiar".*
3. *Filtrado por edad: Se determina la edad del usuario. En este caso "niños"*

Con los filtros establecidos, se obtiene una lista de resultados. El proceso es el siguiente:

Para el recomendador de cada actividad (cines, restaurantes, museos, etc), la clase RecomendadorActividad.java realiza una llamada a un método que asigna valores a las distintas categorías almacenadas en las tablas SQL de cada actividad.

Continuando la descripción del ejemplo visto anteriormente, la clase *RecomendadorCines.java* llamaría al método *getPesosUsuario()* pasándole como parámetro el modelo de grupo y la edad del usuario, y este devolvería la información en el formato de una lista de preferencias requerida para cada actividad.

Finalmente, tras ser procesadas todas las peticiones, se devuelve al cliente una lista de resultados.

5.3. Subsistema: módulos de recomendación

El sistema de recomendación de MadridManía se compone de varios submódulos recomendadores que, una vez unidos, dan forma a la funcionalidad de recomendación de actividades de la aplicación.

El proyecto original ya contaba con recomendadores para las actividades de museos, parques, paseos, restaurantes y cines, de modo que nosotros en este apartado sólo incluiremos los que han sido desarrollados para las nuevas actividades que se han añadido añadido. Explicaremos brevemente cada uno de estos submódulos así como su forma de acoplarse con el resto.

Cabe destacar que la implementación de todos estos módulos recomendadores, tanto los del proyecto original como los nuevos que se han

desarrollado, se encuentra basada en un proceso CBR y se ha llevado a cabo utilizando el framework jCOLIBRI.

5.3.1. Recomendador de centros comerciales

El primer módulo del que hablaremos en este apartado es el recomendador de centros comerciales. A continuación se describe su objetivo, vista de datos y diseño.

5.3.1.1. Objetivo del subsistema

El primer módulo del que hablaremos es el de centros comerciales, que localiza en la base de datos un posible centro comercial que recomendar el usuario. Para llevar a cabo una recomendación de esta actividad el recomendador utiliza las valoraciones de otros usuarios y la localización del centro comercial. La implementación concreta de este módulo se encuentra en la clase *RecomendadorCentrosComerciales.java*.

5.3.1.2. Vista de datos y diseño del subsistema

El recomendador lleva a cabo el siguiente proceso a la hora de llevar a cabo una recomendación:

- **Obtención de los datos**
  El primer paso realizado por el recomendador es la búsqueda en la base de datos de todos los posibles centros comerciales a recomendar y el mapeo de cada uno de los elementos recuperados para crear un objeto de la clase *CentroComercialDescription.java*, que engloba todos los atributos de los que se hará uso en la recomendación.
  Una vez que ha sido generada esta base de datos, se asignan los pesos a los diferentes atributos a evaluar para cada centro comercial (localización, valoraciones de otros usuarios, etc.), con lo que se consigue dar una mayor o menor importancia a una u otra propiedad de los centros comerciales.

- **Ciclo CBR**
  Cuando ya se han recuperado correctamente todos los posibles centros comerciales, se han almacenado en una base de casos y se ha asignado un

peso a cada uno de los atributos, se procede a ejecutar el ciclo CBR, que se encarga de evaluar cada uno de los elementos de la base de casos y devuelve un lista ordenada con un "valor de satisfacción", que indica el grado de ajuste de dicho centro comercial a las preferencias del usuario. Cuanto mayor sea el valor, mejor será el ajuste.
Aunque sería posible recuperar solamente los N elementos de mayor "valor de satisfacción", se trabaja con toda la lista completa para aplicar filtrados posteriores.

- **Filtrado y recuperación de las soluciones**
  Una vez obtenida la lista de centros comerciales evaluada y ordenada de mayor a menor en función del "valor de satisfacción", se procede a aplicar un filtrado sobre el horario del centro comercial para confirmar si resulta apropiado para las restricciones que se han impuesto. En caso de no serlo, esta actividad se descarta y se busca otro posible centro comercial de la lista. Cuando este filtrado finaliza, se devuelve como resultado aquel centro comercial que tenga un mayor "valor de satisfacción" y que cumpla el filtro anterior, mapeando dicho resultado en un objeto de la clase *CentroComercialSolution.java*.

- **Función de agregación**
  Para el caso de los centros comerciales no existe ningún tipo de preferencia específica para un "tipo" de centro comercial en concreto. Por esto, cuando el recomendador solicita un centro comercial, en lugar de aplicar una función de agregación, el recomendador trata de ofrecer el centro comercial más cercano y con mejores puntuaciones de otros usuarios.

5.3.2. Recomendador de parques de ocio

A continuación se presenta el módulo de recomendación de planes de ocio, muy similar al recomendador anterior de centros comerciales.

5.3.2.1. Objetivo del subsistema

El recomendador de parques de ocio se encarga de localizar en la base de datos un posible parque de ocio que recomendar al usuario. Como en el caso de los centros comerciales, este recomendador trabaja con las valoraciones de los usuarios y la localización del parque de ocio.

5.3.2.2. Vista de datos y diseño del subsistema

Para obtener la recomendación de un parque de ocio, el módulo sigue los siguientes pasos:

- **Obtención de los datos**
  En primer lugar, el recomendador de parques de ocio accede a la base de datos local del servidor para generar un base de casos que contenga todos los posibles parques de ocio a recomendar. Estos parques de ocio se mapean en un objeto de la clase *ParqueDeOcioDescription.java*, el cual cuenta con todos los atributos necesarios a la hora de llevar a cabo la recomendación.
  Tras haber generado esta base de casos, se le asignan diferentes pesos a los atributos a evaluar para cada parque de ocio.

- **Ciclo CBR**
  El siguiente paso realizado después de la carga de la base de casos y la asignación de pesos a los atributos es la ejecución del ciclo CBR para la evaluación de cada uno de los parques de ocio. Al igual que en los demás recomendadores, la evaluación se lleva a cabo mediante la asignación de un "valor de satisfacción" a cada uno de los posibles parques de ocio.
  Cuando se ha conseguido la lista ordenada de parques de ocio, se trabaja con ella en su totalidad para aplicarle un posterior filtrado.

- **Filtrado y recuperación de las soluciones**
  Para cada parque de ocio de la lista anterior se aplica un filtro consistente en una comprobación sobre el horario y la duración de la actividad. En el caso de que esta no se ajuste al filtrado, se descarta la actividad y se comprueba el siguiente elemento de la lista.
  Como recomendación se devuelve aquel parque de ocio que cumpla el filtro

anterior y posea el mayor “valor de satisfacción” posible mapeado dentro de un objeto de la clase *ParqueDeOcioSolution.java*.

- **Función de agregación**
  Al igual que en el caso anterior, no aplicamos ninguna función de agregación para este tipo de recomendación. El motivo es que actualmente no existe ningún elemento diferenciador para diferentes parques de ocio almacenados en las preferencias del usuario. Por ejemplo: se podría haber recogido información acerca del tipo de parque ocio, como parque de atracciones, parque temático o parque acuático, que seguramente tendrían características que encajarían mejor o peor con los usuarios.
  Actualmente la recomendación de parques de ocio juega con atributos básicos como la localización o las valoraciones de otros usuarios.

5.3.3. Recomendador de hoteles

Este recomendador tiene la particularidad de recuperar un resultado a través de la API de HSR, otro proyecto de grado realizado este año y que trata sobre un sistema de valoración de hoteles.

5.3.3.1. Objetivo del subsistema

El objetivo de este sistema es obtener un hotel que recomendar a un usuario si selecciona este tipo de actividad. Este recomendador tiene la particularidad de obtener el hotel a recomendar de forma dinámica en lugar de seleccionar un resultado de entre los que se encuentran almacenados en una base de datos en el servidor.

5.3.3.2. Vista de datos y diseño del subsistema

El recomendador de hoteles realiza los siguiente pasos para la elección de un hotel:

- **Obtención de los datos**
  La API se HSR devuelve el mejor hotel en base a un análisis sintáctico de los

resultados que devuelve dicho hotel en Google. Los compañeros que han llevado a cabo este proyecto han adaptado su funcionalidad para que nosotros podamos obtener el mejor hotel en una ubicación dada en base al análisis que ellos llevan a cabo. Este resultado se mapea en un objeto de la clase *HotelDescription.java*, que incluye todos los atributos relevantes para la recomendación.

- **Ciclo CBR**
  En el caso de este recomendador no se emplea el ciclo CBR para seleccionar el hotel más adecuado para la recomendación de una lista de opciones pues HSR ya nos devuelve el mejor hotel para la ubicación indicada.

- **Filtrado y recuperación de las soluciones**
  A continuación, se comprueba que el hotel devuelvo cumple con las restricciones horarios impuestas y se mapea éste es un objeto de la clase *HotelSolution.java*.

5.3.3. Recomendador de plantillas abstractas

Por último, en este apartado se describe el funcionamiento del recomendador de plantillas abstractas.

5.3.3.1. Objetivo del subsistema

El recomendador de plantillas abstractas se encarga de generar una plantilla de actividades completamente nueva cuyo contenido será rellenado posteriormente con los recomendadores de actividades concretas (cines, restaurantes, parques, paseos, museos, centros comerciales, parques de ocio y hoteles). Al igual en todos los casos anterior, nos encontramos ante un sistema de recomendación CBR.

5.3.3.2. Vista de datos y diseño del subsistema

Para este recomendador, la secuencia de operaciones a llevar a cabo sería la siguiente:

- **Obtención de los datos**

  El primer paso que realiza este recomendador es recuperar todas las posible configuraciones de actividades disponibles, mapeando éstas en objetos de la clase *PlantillaDescription.java*.

  Se recuperan combinaciones de la forma: 09:00 - 11:00 paseo, 09:45-15:45 paseo + parque + restaurante, etc. y se guardan los datos necesarios de cada una de ellas en los que se basará la recomendación (lista de actividades, horario del plan, etc.).

  Todas estas combinaciones se almacenan en una lista que será la bases de casos empleada después en el ciclo CBR.

  A continuación se procede a asignar una serie de pesos a cada uno de los atributos que se tendrán en cuenta a la hora de evaluar cada plan abstracto, es decir, aquellos parámetros que permitirán asignar un valor de "importancia" a unas propiedades frente al resto.

  Los atributos más importantes a evaluar son el horario del plan concreto y la lista de actividades a fin de que encajen lo mejor posible con las restricciones impuestas por el usuario y el perfil de preferencias.

- **Ciclo CBR**

  Ya partiendo de una serie de restricciones, una base de casos y unos pesos fijados para cada atributo, se procede a ejecutar el ciclo CBR para obtener una lista ordenada de planes abstractos en función de su "valor de satisfacción", que indica cómo se ajusta dicho plan a las preferencias del usuario. A mayor valor, más adecuada será la recomendación de ese plan para el usuario.

- **Filtrado y recuperación de la solución**

  A diferencia de los recomendadores de actividades, en este caso no es necesario realizar un filtrado posterior ya que el resultado devuelvo siempre es el de mayor "valor de satisfacción" posible. Esto se debe a que solo se accede a este recomendador si no hay otra forma de obtener un plan mejor (es decir, es el último recurso) y debe devolver siempre un plan válido.

- **Rellenado de la plantilla**

  Con el plan abstracto ya definido, el recomendador de plantillas abstractas procede a rellenar cado uno de los huecos disponibles con actividades concretas (centros comerciales, cines, etc.) según vayan siendo necesarios.

El sistema de recomendación de plantillas abstractas no utiliza ninguna función de agregación ya que, si fuese necesario, éste ya se aplicaría dentro de cada recomendador individual.

Conclusiones

Dentro de este capítulo, hemos tratado de explicar con el mayor rigor y claridad posible el funcionamiento de los tres principales subsistemas introducidos en MadridManía: *modo invitado*, *modelos de grupo* y el *módulo recomendador para nuevas actividades*. En cada uno de ellos, se ha definido los objetivos, la vista de datos y el diseño. Además, para entender mejor su funcionamiento, se han descrito con la mayor claridad posible conceptos de utilidad, para así conseguir su mejor comprensión.

Capítulo 6

Valoración experimental

En este capítulo hemos pretendido explicar que para realizar una modificación o mejora del sistema se ha de prestar especial atención a las posibles necesidades de los potenciales usuarios de la aplicación. Para ello se han realizado una serie de encuestas con el objeto de analizar los datos obtenidos, y posteriormente mejorar la aplicación.

A continuación, procuramos mostrar cómo ha sido realizado el diseño del experimento buscando un amplio abanico de opiniones para poder extraer los resultados deseados.

6.1. Diseño del experimento inicial

Para ver cuáles eran las necesidades de los usuarios, la búsqueda se realizó entre segmentos de población de las distintas edades, grupos sociales [Turner, J.C., 1982], grupos culturales tratando con ello de obtener la mayor cantidad posible de datos, que con posterioridad hemos podido utilizar para confeccionar los nuevos componentes del sistema. Sirve como ejemplo el grupo laboral (compañeros de trabajo). En este grupo hay individuos de distintas edades, distintas aficiones y con características familiares diferenciadas. El grupo de amigos y conocidos pertenecientes al ámbito universitario es el más explorado y evaluado, ya que es muy amplio, diverso y nuestro contacto es directo y permanente.

Contrastando todas estas especificaciones, llegamos a concluir cuales son las cuestiones más interesantes que deseamos que nos responda cada usuario. A esto podemos añadir la información (que nos resulta más relevante) de otros sistemas de recomendación analizados en el estado del arte (Fever, Whatsred, etc).

Hemos añadido una serie de cuestiones tanto a nivel individual como grupal para un rango un de 50 individuos entre las que se incluyen:

- I*nteracción del usuario con el sistema.*
- *Modo de inicio de sesión.*
- *Inclusión de nuevas actividades.*
- *Gestión de los grupos.*
- *Modelos grupo para actividades de ocio.*

6.2. Resultados del experimento inicial

Una vez obtenidos los resultados, se recogieron todos los datos recogidos en las evaluaciones. Se pretende explicar que para realizar una modificación o mejora del sistema se ha de prestar especial atención a las posibles necesidades de los potenciales usuarios de la aplicación. Por ello, se han establecido una serie de encuestas con el objetivo de mejorar el sistema, teniendo en cuenta las opiniones de dichos usuarios.

Desde la perspectiva de la interacción del usuario con la aplicación, dado que se comprobó que suponía un inconveniente por su gran amplitud, hemos considerado gestionar de manera diferente el test de preferencias iniciales (sección 5.2.3.1).

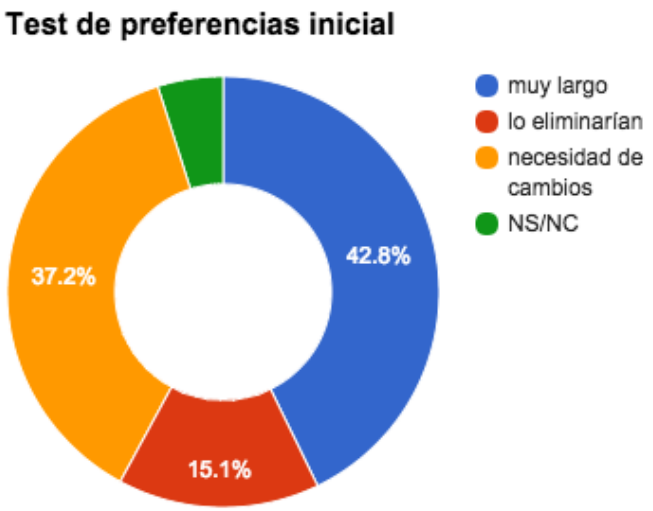


Figura 6.1: Resultados del Test de preferencias inicial

Otro punto considerado fundamental es que los usuarios, en términos prácticos, demandaban poder utilizar la aplicación sin tener que recurrir necesariamente a iniciar sesión en Facebook u otra red social (Como en el caso de Whatsred).

Figura 6.2: Resultados de la encuesta sobre la creación de un modo invitado

Según se aprecia en el gráfico superior, la creación de un modo de acceso invitado ha sido acogida muy positivamente entre los encuestados, por lo que sin duda es uno de los módulo añadidos prioritariamente al sistema. Provee al usuario de la capacidad de poder utilizar de forma rápida la aplicación y el sistema de recomendación (figura 6.2).

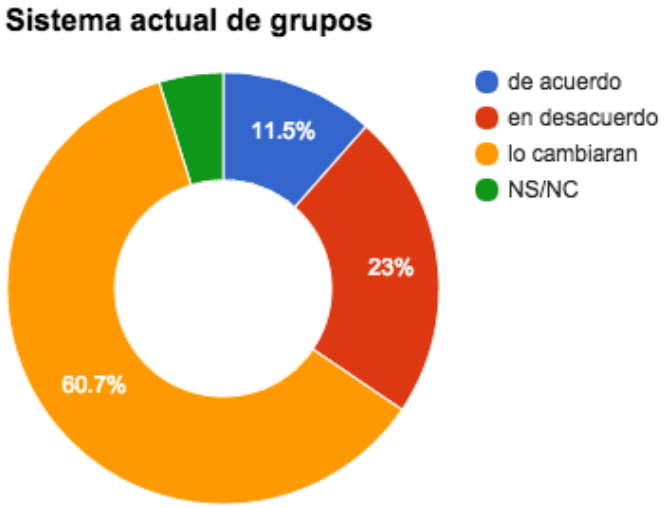


Figura 6.3: Resultados de la encuesta sobre el sistema actual de grupos

Conforme se procedió al estudio de la gestión de grupos, vimos la necesidad de mejorar la funcionalidad para actividades y planes de ocio grupales. Por tanto, se incluyó una nueva técnica basada en los modelos de grupo (sección 5.2) que dotase al usuario de la capacidad de seleccionar dichos planes y actividades. Para la elaboración de los modelos, se ofertaba la posibilidad a los usuarios de seleccionar un modelo grupal ante un listado de oportunidades. Posteriormente se vio la necesidad de que los grupos que debían formar para del sistema son: *familia*, *parejas*, *trabajo* y *amigos*.

Figura 6.4: Resultados de la encuesta: modelos de grupo escogidos

Finalmente, teniendo presentes las demandas de los usuarios, se procedió a la incorporación de nuevas actividades de ocio. Tratando siempre de enfocar que estas actividades fueran realizadas grupalmente, al ser un sistema de recomendación orientado para grupos.

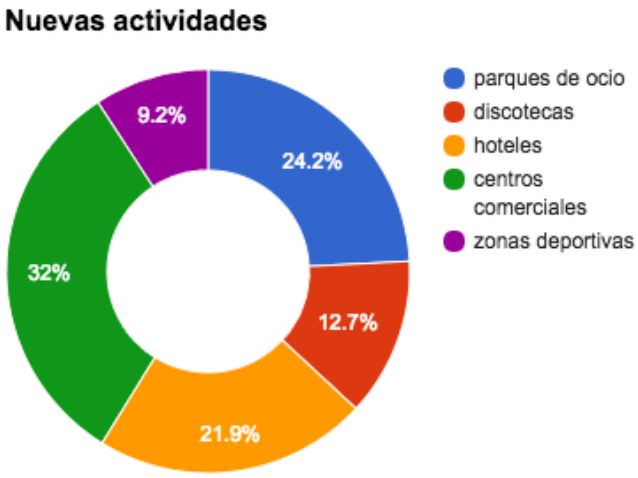


Figura 6.5: Resultados de la encuesta: nuevas actividades

Fueron seleccionadas por tanto: *parques de ocio*, *centros comerciales* y *hoteles*.

6.3 Experimento final: MadridManía

Una vez cosechados los resultados del experimento inicial, analizado cuáles eran las necesidades del usuario, y modificado el sistema, el siguiente paso es realizar una evaluación a posteriori. De este modo podrá verse si los cambios realizados han solucionado las necesidades detectadas.

6.3.1 Diseño del experimento final: MadridManía

De igual forma que en la evaluación anterior, hemos añadido una serie de cuestiones tanto a nivel individual como grupal para un rango un de 50 individuos:

- I*nteracción del usuario con el sistema.*
- *Modo de inicio de sesión.*
- *Inclusión de nuevas actividades.*
- *Gestión de los grupos.*
- *Modelos grupo para actividades de ocio.*

6.3.1 Resultados del experimento final: MadridManía

Se pretende demostrar en esta sección, que los cambios realizados en MadridManía han tenido valoraciones positivas.

Al modificar la recogida de preferencias y el diseño de la aplicación, la interacción con el usuario ha mejorado notablemente. Se puede apreciar (Figura 6.6) que el porcentaje de usuarios satisfechos ronda el 90%.

Figura 6.6: Resultados de la encuesta: satisfacción con la interacción

En el capítulo de las líneas de trabajo futuro se especifica cómo mejorar el sistema de recomendación.

El modo invitado proporciona un acceso a cualquier usuario de la aplicación para ofrecer un conjunto de actividades que forman un plan de ocio. Esta funcionalidad ha sido acogida con gran interés por los usuarios, ya que permite obtener una recomendación rápida y sencilla, y casi el 95% de las evaluaciones dan resultados positivos (Figura 6.7).

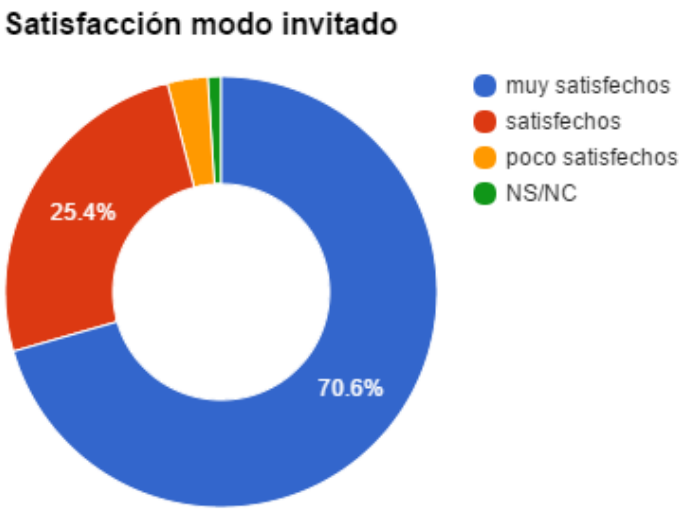


Figura 6.7: Resultados de la encuesta: satisfacción con el modo invitado

Para analizar las evaluaciones relativas a la inclusión de nuevas actividades se procedió de la siguiente forma:

A cada individuo que formaba parte del experimento, se le especificó que indicase el conjunto actividades que demandaban su interés. Por lo tanto, pudimos comprobar (Figura 6.8) si ha sido un acierto incluir estas nuevas actividades (centros comerciales, parques de ocio y hoteles).

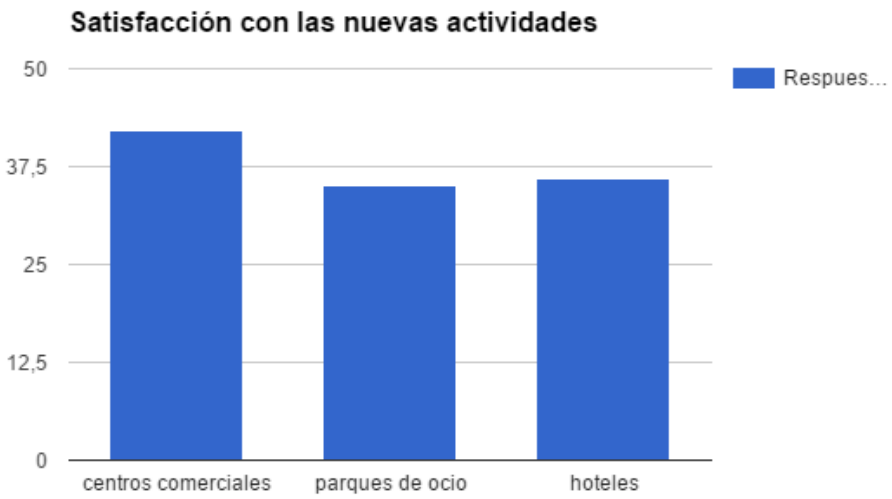


Figura 6.8: Resultados de la encuesta: nuevas actividades

Por último, al tratar de mejorar la recomendación para grupos, realizamos una evaluación de la gestión actual realizada, para poder compararla con la que se hacía anteriormente. El 82,7% está de acuerdo con el nuevo sistema (Figura 6.9), que permite crear un grupo personalizado, o obtener recomendaciones de planes de ocio grupales simultáneamente. Por tanto, la satisfacción con la nueva funcionalidad que ofrece el sistema es bastante notable.

Figura 6.9: Resultados de la encuesta: nuevas actividades

6.4 Conclusiones del experimento

En este capítulo hemos realizado un recorrido por las evaluaciones de los usuarios, a los que en términos finales va dirigida la aplicación. Para proceder a establecer una línea más clara de trabajo, se ha realizado una recogida de opiniones inicial, identificando las necesidades de los usuarios. Posteriormente, tras realizar los cambios pertinentes y añadir módulos, se ha querido tomar constancia de si estas modificaciones han sido correctas. Tras el análisis de datos extraído en el experimento final, podemos concluir que MadridManía ha mejorado la funcionalidad del sistema, a partir de las restricciones impuestas por los usuarios.

Puntos positivos

1. Dos métodos de acceso: Facebook e invitado.
2. Diferentes alternativas para realizar actividades en grupo.
3. Resulta sencillo elegir los planes de ocio.
4. La interfaz mejora la usabilidad.

Puntos negativos

1. El numero de modelos de grupo que se puede escoger es limitado.
2. El usuario no puede crear los modelos de grupo, vienen establecidos por el sistema.

Capítulo 7

Conclusiones del proyecto

En el capítulo 6 se ha descrito el experimento de validación realizado junto con las conclusiones obtenidas de éste, ahora en este capítulo se comentan las conclusiones generales de todo el proyecto y las aportaciones a éste de cada uno de nosotros.

7.1. Objetivos del proyecto

En primer lugar, en lo referido al cumplimiento de objetivos en el proyecto, se han cerrado todos los puntos propuestos:

- **Creación de un modo invitado.** El primer gran cambio que hemos llevado respecto al proyecto original ha sido la creación de un modo invitado que complementa el modo de funcionamiento con Facebook que existía antes. De este modo es posible utilizar la aplicación sin necesidad de acceder a ella utilizando una red social. Sin embargo, al tratarse MadridManía de un recomendador de planes de ocio grupales, ha sido necesario realizar cambios como la caracterización de grupos para conseguir que las recomendaciones sigan siendo lo más precisas posibles.

- **Caracterización de grupos.** Este objetivo venía impuesto en cierta manera a coalición del punto anterior, pues la creación de un modo invitado implicaba perder la creación de grupos de Facebook del modo original de funcionamiento de la aplicación. En este caso optamos por crear modelos de grupo en base a un rango de edades y un tipo de relación entre sus miembros para caracterizar a grupos de usuarios que creemos que serían los que podrían estar interesados en utilizar un tipo de aplicación como MadridManía. Gracias a estos grupos, es posible asignar un mayor o menor peso a distintos tipos de

actividades y ajustar las recomendaciones a ellos aunque en este modo no contemos con la información proporcionada por Facebook.

- **Inclusión de nuevas actividades de ocio.** Para poder ofrecer unas recomendaciones más amplias a lo usuarios, era necesario añadir nuevas actividades de ocio al catálogo existente. Las actividades con las que contaba el proyecto original más las tres que han sido añadidas han conseguido que MadridManía ofrezca recomendaciones bastante variadas y aproveche aún más las múltiples posibilidades de ocio que oferta la ciudad de Madrid.

- **Mejora de la interfaz de la aplicación.** Por último, uno de los puntos que queríamos mejorar del proyecto original era la interfaz de la aplicación. Con MadridManía hemos intentado aportar un aspecto más moderno a las pantallas, así como añadir nuevos botones e iconos que muestran de manera más clara su función.

Dejando de los cambios que hemos realizado, debemos comentar también aquellos conocimientos que hemos adquirido durante la realización de este proyecto.

Como ya se comentó al inicio de esta memoria, los sistemas de recomendación son un tema que ha cobrado especial importancia durante los últimos años. Multitud de servicios web dedicados a la oferta de productos o servicios (Amazon, Tripadvisor, Fever, Whatsred, etc.) aprovechan las posibilidades que ofrecen los sistemas de recomendación para garantizar a sus usuarios una mejor experiencia al visitarlos y mejorar sus posibilidades de venta. Nos ha resultado muy interesante comprender desde mi primera mano cómo funciona un sistema recomendador y creemos que estos conocimientos que hemos adquirido acerca de ellos pueden sernos de utilidad de cara al futuro.

Otro de los puntos clave que nos ha proporcionado la realización de este proyecto ha sido el estudio de los modelos de grupo y cómo poder dividir a un público objetivo para nuestra aplicación en subgrupos con características distintas. El estudio que hemos llevado a cabo para asignar un mayor o menor peso a una serie de parámetros para ajustar las recomendaciones a un grupo de usuarios nos ha hecho comprender la dificultad que entraña ofrecer una recomendación que se ajuste a los gustos o preferencias de éstos, y la multitud de información que se ha de tener en cuenta si quiere hacerse de la manera más correcta posible.

El uso de APIs para recuperar información de otros servicios y utilizarla en nuestra aplicación ha sido otro punto que nos ha resultado de especial interés durante el desarrollo de MadridManía. La API de 11870, con la que nosotros hemos trabajado principalmente para nuestro proyecto, ofrece multitud de tipos distintos de actividades de ocio para distintas ciudades de España, aunque nosotros solo nos hayamos centrado en la ciudad de Madrid. Una vez hecho el desarrollo para mapear los datos recibidos de una API en objetos Java manejables por nuestros recomendadores, añadir nuevas actividades ya sea de ésta o de otras APIs es relativamente sencillo.

Cabe destacar el uso que hemos tenido que realizar tanto de la tecnología Android para la implementación de la parte del cliente de MadridManía como la parte de servidor a partir de J2EE. Ambos habíamos desarrollado anteriormente para Android, pero desarrollar un proyecto entero como este nos ha hecho ver la multitud de posibilidades que ofrece este sistema operativo y los dispositivos móviles en general, y como rebuscando por Internet se pueden encontrar montones de información acerca de cómo añadir multitud de funcionalidades a una aplicación. Cada vez que nos ha surgido algún problema durante el desarrollo, hemos sido capaces de encontrar cómo solucionarlo de forma más o menos rápida, lo que es muy de agradecer para los desarrolladores. En la parte del servidor sí que nos encontrábamos algo más verdes y hemos tenido que partir casi desde cero hasta llegar a entender su funcionamiento y su manera de intercambiar información con el cliente de Android. Esto último, junto con los sistemas de recomendación, es probablemente lo que pensamos que más nos puede aportar de cara a nuestra vida laboral.

7.2. Aportaciones de cada miembro del grupo

En este apartado se resumen las aportaciones de cada miembro del grupo en el desarrollo de este proyecto. Cabe destacar que hemos intentado realizar un reparto de tareas equitativo, para que la carga de trabajo sea similar.

7.2.1. Aportaciones de Guillermo

En esta sección describo en detalle cuáles han sido mis aportaciones a lo largo del desarrollo del proyecto de este trabajo de fin de grado y su memoria. En primer lugar describiré las tareas que he llevado a cabo en lo referido al diseño de la aplicación, posteriormente mencionaré mis aportaciones en lo que se refiere al desarrollo (a nivel de programación) y continuaré con aquellas partes de la memoria que han sido redactadas por mí.

Primeramente, en lo que al diseño de la aplicación se refiere, ha sido necesario estudiar cómo funcionan los sistemas basados en conocimiento (transparencias de ISBC[14]), los sistemas de recomendación que podemos encontrar en la actualidad (Whatsred o FilmAffinity) y la aplicación predecesora (Madrid Live). Dado que es un sistema orientado a grupos, se puede enriquecer el sistema mediante técnicas utilizadas en los recomendadores grupales. En nuestro caso, nos decantamos por **crear un modelo de grupo**. Para obtener las características de dicho modelo, realicé un conjunto de encuestas (sección 6.1) que posteriormente distribuimos a personas de nuestro entorno (al ser el tamaño de la muestra 50 usuarios, acordamos cumplimentar 25 cada uno).

En cuanto al propio desarrollo de la aplicación, me he centrado sobre todo en la parte *frontend*. Me he encargado de añadir en la pantalla de solicitar la recomendación los campos para seleccionar grupos y actividades, así como la gestión interna de los mismos. Se incluye en este punto la conexión cliente-servidor utilizando Servlets para enviar y recibir parámetros.

El desarrollo *backend* lo he centrado más en la gestión de grupos, mediante métodos para seleccionar las preferencias a la hora de realizar una recomendación. Para el modo invitado, hemos trabajado colaborativamente creando métodos específicos que controlan el acceso a usuarios de Facebook o no registrados. Cabe destacar que han sido necesarios cambiar muchos puntos de la aplicación (tanto en cliente como el servidor), ya que esta estaba pensaba para la recomendación a

[14] ISBC: Ingeniería de Sistemas basados en Conocimiento, GAIA, Departamento del Software e Inteligencia Artificial. Universidad Complutense de Madrid.

usuarios registrados. Estas aportaciones se especifican a lo largo del capítulo 5, ya que constituyen las principales novedades que incorpora MadridManía.

Para realizar la memoria contribuimos a elaborar un índice común y fuimos rellenado contenidos. En función de las aportaciones que cada miembro había hecho en el proyecto, se centró más en la redacción de sus contenidos. El anexo se elaboró de forma conjunta. Para la parte de inglés yo hice la introducción y mi compañero las conclusiones.

Por último, a modo de resumen, se establece una comparativa en porcentaje de los trabajos realizados, intentando que la carga fuera equitativamente al 50% tanto en dificultad, como en contenido.

Porcentaje de trabajo de Guillermo:

- Diseño: 60%
- Bases de datos: 20%
- Funcionalidades: 50%
- Subsistemas 50 %
- Memoria 40 %

7.2.2. Aportaciones de Marta

En esta sección describo en detalle cuáles han sido mis aportaciones a lo largo del desarrollo del proyecto de este trabajo de fin de grado y su memoria. En primer lugar describiré las tareas que he llevado a cabo en lo referido al diseño de la aplicación, posteriormente mencionaré mis aportaciones en lo que se refiere al desarrollo (a nivel de programación) y continuaré con aquellas partes de la memoria que han sido redactadas por mí.

Primeramente, en lo que al diseño de la aplicación se refiere, ha sido necesario estudiar cómo funcionan los sistemas basados en conocimiento (transparencias de ISBC[15]), los sistemas de recomendación que podemos encontrar en la actualidad (Fever o Amazon) y la aplicación predecesora (Madrid Live). Dado que es un sistema orientado a grupos, se puede enriquecer el sistema mediante técnicas utilizadas en los recomendadores grupales. En nuestro caso, nos decantamos por **crear un modelo de grupo**. A raíz de las encuestas diseñadas por mi compañero, distribuimos las mismas a personas de nuestro entorno (al ser el tamaño de la muestra 50 usuarios, acordamos cumplimentar 25 cada uno).

En cuanto al propio desarrollo de la aplicación, me he centrado sobre todo en la parte *backend*. Me he encargado de añadir las **nuevas actividades:** centros comerciales, parques de ocio y hoteles (este último en colaboración con Guillermo). En este punto, cabe destacar la implementación de todos estos módulos recomendadores basados en un proceso CBR y que utilizan el framework jCOLIBRI. Estas aportaciones se especifican a lo largo de la sección 5.3.

Para el modo invitado, hemos trabajado colaborativamente creando métodos específicos que controlan el acceso a usuarios de Facebook o no registrados. Cabe destacar que han sido necesarios cambiar muchos puntos de la aplicación (tanto en cliente como el servidor), ya que esta estaba pensaba para la recomendación a usuarios registrados. Estas aportaciones se especifican a lo largo del capítulo 5, ya que constituyen las principales novedades que incorpora MadridManía.

[15] ISBC: Ingeniería de Sistemas basados en Conocimiento, GAIA, Departamento del Software e Inteligencia Artificial. Universidad Complutense de Madrid.

Para realizar la memoria contribuimos a elaborar un índice común y fuimos rellenado contenidos. En función de las aportaciones que cada miembro había hecho en el proyecto, se centró más en la redacción de sus contenidos. El anexo se elaboró de forma conjunta. Para la parte de inglés yo hice las conclusiones y mi compañero la introducción.

Por último, a modo de resumen, se establece una comparativa en porcentaje de los trabajos realizados, intentando que la carga fuera equitativamente al 50% tanto en dificultad, como en contenido:

Porcentaje de trabajo de Marta:

- Diseño: 40%
- Bases de datos: 80%
- Funcionalidades: 50%
- Subsistemas 50 %
- Memoria 40 %

Chapter 7

Conclusion

Chapter 6 described the validation experiment conducted in conjunction with the conclusions drawn from it. In this chapter the general conclusions of the whole project and contributions to it of each of us are discussed.

7.1. Project

First, with regard to targets in the project we have done all the items proposed:

- **Creating a guest mode**. The first big change with respect to the original project has been the creation of a guest mode so that complements the operation mode links that existed before. This makes it possible to use the application without having to access it using a social network. However, MadridManía as being a leisure plan recommender, it was necessary to make changes as the characterization of groups to make recommendations remain as accurate as possible.

- **Characterization of groups**. This goal was imposed in some way coalition of the previous point. For the creation of a guest mode it meant losing creating Facebook groups operating mode of the original application. In this case we chose to create models of group based on a range of ages and a relationship among its members to characterize groups of users that we believe would be those who might be interested in using a type of application as MadridManía. Through these groups, you can assign more or less weight to different types of activities and adjust them to the recommendations although in this mode does not have the information provided by Facebook.

- **Inclusion of new leisure activities**. To offer broader recommendations to users, it was necessary to add new leisure activities to the existing catalog. The activities that had the original project plus the three that have been

added have made MadridManía to offer wide range of recommendations and take further the many leisure opportunities offered by the city of Madrid.

- **Improved application interface**. Finally, one of the points that we wanted to improve from the original project was user interface of the application. With MadridManía we have attempted to provide a more modern look to the screens as well as add new buttons and icons that show more clearly its function.

In addition to the changes we have made, we should remark those skills that we have acquired during this project.

As already mentioned at the beginning of this specification, recommendation systems are an issue that has become particularly important in recent years. Many web services dedicated to fulfilled users requirements (Amazon, Tripadvisor, Fever, Whatsred, etc.) exploit the potential of recommendation systems to ensure its users more personal and relevant experience and improve their chances of selling. We have learnt a lot from understanding how a recommender system works and we believe that this knowledge we have gained can be useful to us in the future.
Another interesting point that has given us the realization of this project has been the study of group models and how to divide a target audience for our application into subgroups with different characteristics. The study we conducted to assign different weights to a number of parameters to adjust the recommendations to a group of users has made us understand the difficulty of providing a recommendation that satisfy the tastes and preferences of users, and the multitude of information that has to be taken into account if you want to be in the most accurate as possible.
Using APIs to retrieve information from other services and use it in our application has been another point that we have found of particular interest for the development of MadridManía. 11870 API, with which we have worked primarily for our project offers many different types of leisure activities to various cities in Spain, but we just have focused on the city of Madrid. Once done the development for mapping data received from an API in Java objects manageable by our recommenders, add new activities either this or other APIs is relatively simple.

Worth to mention the use we had to perform both Android technology for implementing the client side of MadridManía and the server part within J2EE. Both had previously done some Android development. However, develop a whole project like this has made us see the multitude of possibilities offered by this operating system and mobile devices in general, and how as searching through Internet you can find lots of information about adding several features to an application. Whenever we have faced a problem during development, we have been able to find how to fix it in a reasonable amount of time, which is much appreciated from the developers' viewpoint. On the server side we found something more inexpert and we had to start almost from scratch until we have understood its operation and how to exchange information with the Android client. This along with recommendation systems, is what we think can be more useful in our future career.

Capítulo 8

Líneas de trabajo futuro

El capítulo final expone una idea de trabajo para establecer una visión de cara a posibles modificaciones o renovaciones con vistas al futuro. Hay muchas líneas abiertas en el proyecto actual de cara a optimizar la funcionalidad. A continuación se procederá a trazar una visión general de ellas con el objetivo de abrir el abanico lo más posible.

- **Caracterización de grupos.** Una vez mejorado el sistema de recomendación para grupos (modelos de grupo), una posible línea de desarrollo es que el sistema sea capaz de detectar el tipo de grupo que mejor se adapte a él en base a solicitudes de recomendación realizadas anteriormente.
- **Inclusión de nuevas actividades de ocio.** En nuestra aplicación hemos añadido nuevas actividades que ofrecen una recomendación más completa. Retomando la idea del ejemplo anterior, una funcionalidad interesante sería que MadridMania no sólo mostrara las actividades mejor valoradas o más visitadas, si no una lista de posibles actividades que pudiesen ser del agrado del usuario, sin que este tuviera que solicitar la recomendación al sistema.
- **Integración con nuevas redes sociales:** MadridMania es una aplicación social. Al estar integrada con Facebook, provee al sistema de información obtenida sobre el usuario y sus amigos. Cabe la posibilidad de integrar cualquier otra red social recogiendo, por ejemplo, información sobre actividades. Un posible caso es que a partir de la red social Twitter, analizando cuales son las tendencias actuales, y si existe en el sistema alguna actividad referente, te la recomendara.
- **Nuevas fuentes de información:** Aunque la API de 11870, Google Place y la Guía del Ocio proporcionan al sistema un amplio catálogo de actividades, hemos incluido un servicio de información externo para el

tipo de actividad “hoteles”. Como se comenta en la sección de las fuentes de información, hace uso de un analizador de expresiones que devuelve el hotel mejor valorado a partir de la longitud y la latitud. La inclusión de estos sistemas de valoración, permite que las actividades recomendadas por el usuario sean de mejor calidad, ya que optimiza los resultados de búsqueda en base a un conjunto de parámetros. Por ejemplo, se podría seleccionar un restaurante o un museo eligiendo el más barato o caro posible (parámetros enviados), o que en función del tipo de grupo, el sistema recomiende uno que mejor se adapte a las necesidades del usuario.

Para finalizar, aunque Android es una tecnología actual, tiene gran cuota de mercado y proporciona un entorno de desarrollo para realizar aplicaciones, un posible cambio sería migrar MadridManía a cualquier otra plataforma (iPhone, Windows Phone…).

Apéndice A

Bases de datos

En este apartado se muestra la estructura de la base de datos MySQL de forma esquemática mediante una serie de figuras, en las que se pueden apreciar las principales tablas utilizadas con la representación de atributos y relaciones entre ellas.

A.1. Almacenamiento de las plantillas realizadas por los usuarios

Por cada plantilla, se incluyen las actividades de las que está compuesta y la fecha de realización de la misma. Estos datos se almacenan en las tablas de la Figura A.1, que recogen información sobre las plantillas de planes de ocio realizados por cada usuario.

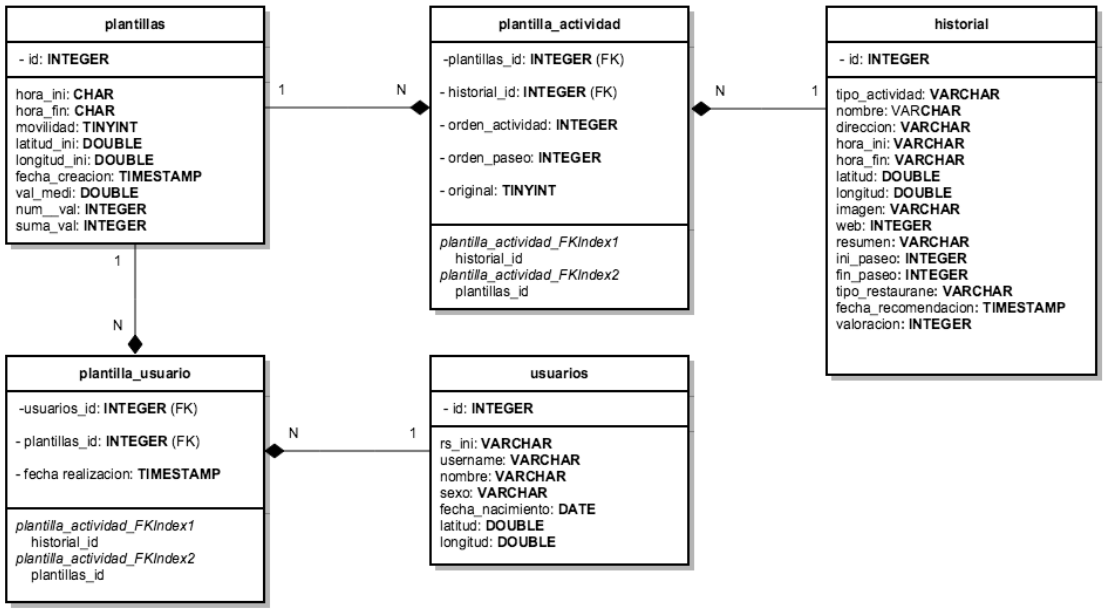


Figura A.1

A.2. Almacenamiento de datos para la recomendación

A continuación se muestran las tablas que almacenan los datos y son necesarias para realizar las recomendaciones, ya que contienen información acerca de las diferentes actividades de ocio (Figura A.2 y Figura A.3).

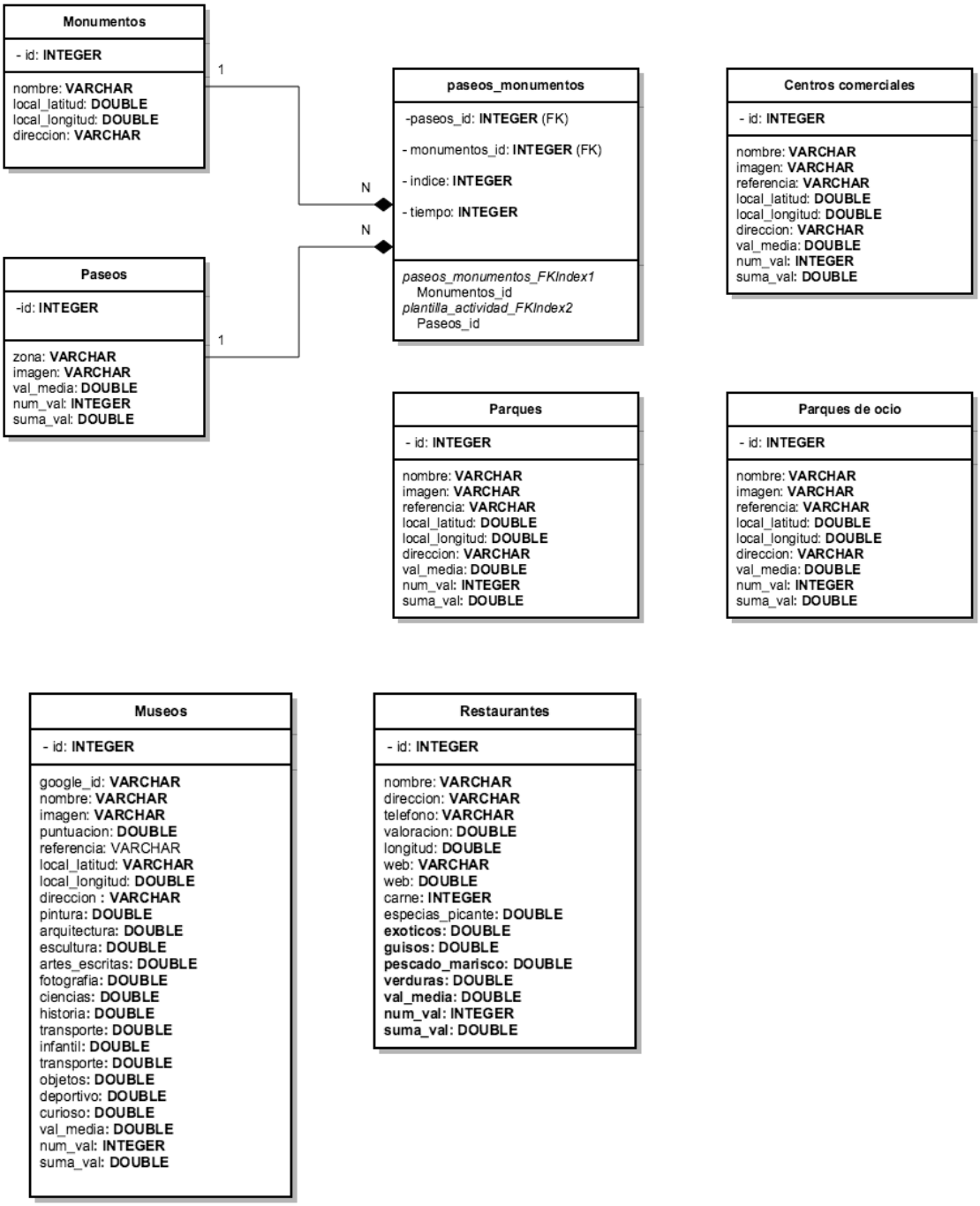

Monumentos
- id: INTEGER
nombre: VARCHAR
local_latitud: DOUBLE
local_longitud: DOUBLE
direccion: VARCHAR
1
N
paseos_monumentos
-paseos_id: INTEGER (FK)
- monumentos_id: INTEGER (FK)
- indice: INTEGER
- tiempo: INTEGER
paseos_monumentos_FKIndex1
Monumentos_id
plantilla_actividad_FKIndex2
Paseos_id
Centros comerciales
- id: INTEGER
nombre: VARCHAR
imagen: VARCHAR
referencia: VARCHAR
local_latitud: DOUBLE
local_longitud: DOUBLE
direccion: VARCHAR
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE
Paseos
-id: INTEGER
zona: VARCHAR
imagen: VARCHAR
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE
1
N
Parques
- id: INTEGER
nombre: VARCHAR
imagen: VARCHAR
referencia: VARCHAR
local_latitud: DOUBLE
local_longitud: DOUBLE
direccion: VARCHAR
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE
Parques de ocio
- id: INTEGER
nombre: VARCHAR
imagen: VARCHAR
referencia: VARCHAR
local_latitud: DOUBLE
local_longitud: DOUBLE
direccion: VARCHAR
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE
Museos
- id: INTEGER
google_id: VARCHAR
nombre: VARCHAR
imagen: VARCHAR
puntuacion: DOUBLE
referencia: VARCHAR
local_latitud: VARCHAR
local_longitud: DOUBLE
direccion : VARCHAR
pintura: DOUBLE
arquitectura: DOUBLE
escultura: DOUBLE
artes_escritas: DOUBLE
fotografia: DOUBLE
ciencias: DOUBLE
historia: DOUBLE
transporte: DOUBLE
infantil: DOUBLE
transporte: DOUBLE
objetos: DOUBLE
deportivo: DOUBLE
curioso: DOUBLE
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE
Restaurantes
- id: INTEGER
nombre: VARCHAR
direccion: VARCHAR
telefono: VARCHAR
valoracion: DOUBLE
longitud: DOUBLE
web: VARCHAR
web: DOUBLE
carne: INTEGER
especias_picante: DOUBLE
exoticos: DOUBLE
guisos: DOUBLE
pescado_marisco: DOUBLE
verduras: DOUBLE
val_media: DOUBLE
num_val: INTEGER
suma_val: DOUBLE

Figura A.2

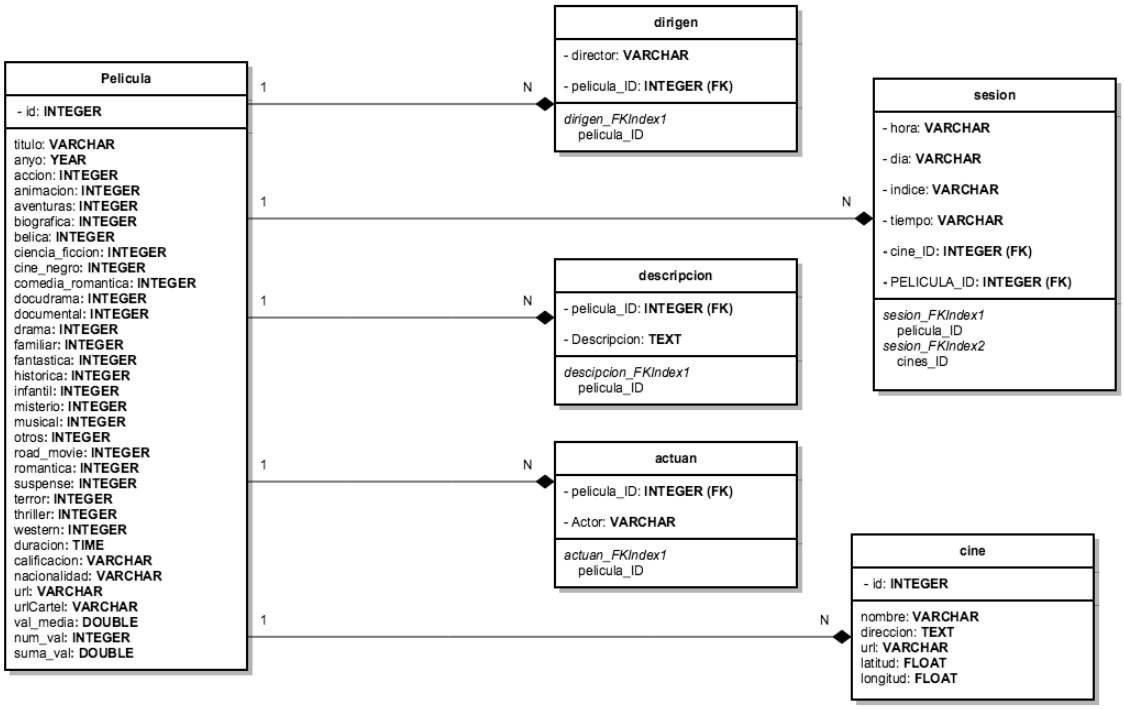


Figura A.3

A.3. Almacenamiento del perfil del usuario y datos para los grupo

La Figura A.4 recoge las tablas que intervienen en el almacenamiento de los datos referentes al perfil del usuario. Es decir, la colección de tablas que contiene los gustos o preferencias de cada usuario (Figura A.4) Se incluye un ejemplo de las tablas utilizadas para la gestión de preferencias en el modelo de grupo familiar.

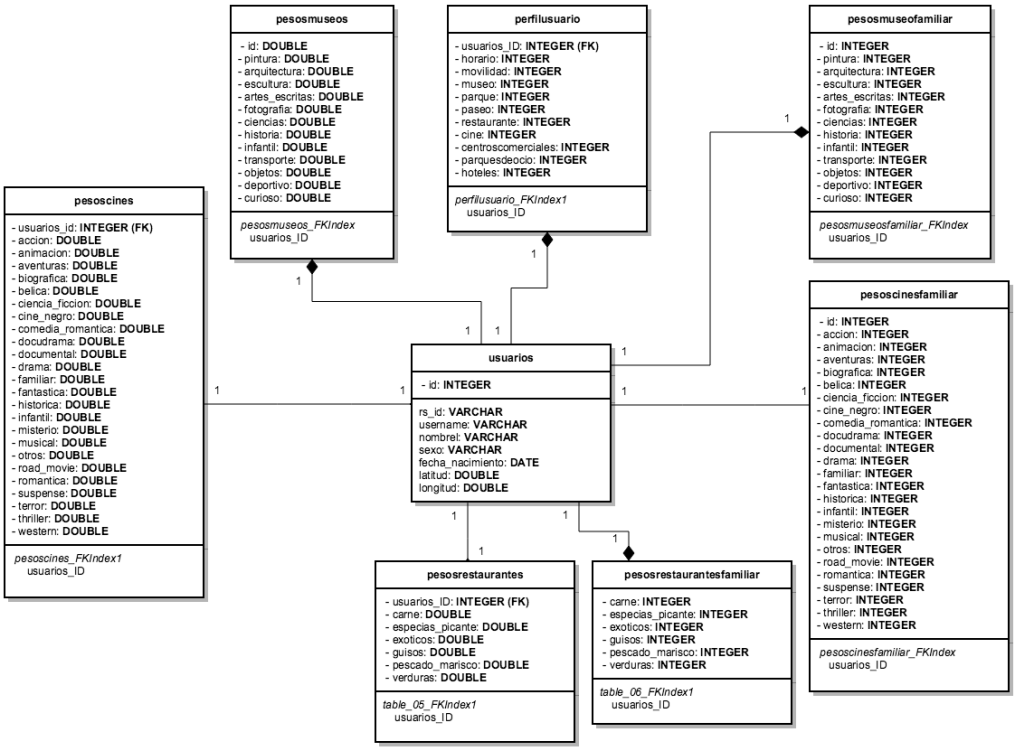

pesosmuseos
- id: DOUBLE
- pintura: DOUBLE
- arquitectura: DOUBLE
- escultura: DOUBLE
- artes_escritas: DOUBLE
- fotografia: DOUBLE
- ciencias: DOUBLE
- historia: DOUBLE
- infantil: DOUBLE
- transporte: DOUBLE
- objetos: DOUBLE
- deportivo: DOUBLE
- curioso: DOUBLE
pesosmuseos_FKIndex
usuarios_ID
perfilusuario
- usuarios_ID: INTEGER (FK)
- horario: INTEGER
- movilidad: INTEGER
- museo: INTEGER
- parque: INTEGER
- paseo: INTEGER
- restaurante: INTEGER
- cine: INTEGER
- centroscomerciales: INTEGER
- parquesdeocio: INTEGER
- hoteles: INTEGER
perfilusuario_FKIndex1
usuarios_ID
pesosmuseofamiliar
- id: INTEGER
- pintura: INTEGER
- arquitectura: INTEGER
- escultura: INTEGER
- artes_escritas: INTEGER
- fotografia: INTEGER
- ciencias: INTEGER
- historia: INTEGER
- infantil: INTEGER
- transporte: INTEGER
- objetos: INTEGER
- deportivo: INTEGER
- curioso: INTEGER
pesosmuseosfamiliar_FKIndex
usuarios_ID
pesoscines
- usuarios_id: INTEGER (FK)
- accion: DOUBLE
- animacion: DOUBLE
- aventuras: DOUBLE
- biografica: DOUBLE
- belica: DOUBLE
- ciencia_ficcion: DOUBLE
- cine_negro: DOUBLE
- comedia_romantica: DOUBLE
- docudrama: DOUBLE
- documental: DOUBLE
- drama: DOUBLE
- familiar: DOUBLE
- fantastica: DOUBLE
- historica: DOUBLE
- infantil: DOUBLE
- misterio: DOUBLE
- musical: DOUBLE
- otros: DOUBLE
- road_movie: DOUBLE
- romantica: DOUBLE
- suspense: DOUBLE
- terror: DOUBLE
- thriller: DOUBLE
- western: DOUBLE
pesoscines_FKIndex1
usuarios_ID
usuarios
- id: INTEGER
rs_id: VARCHAR
username: VARCHAR
nombrel: VARCHAR
sexo: VARCHAR
fecha_nacimiento: DATE
latitud: DOUBLE
longitud: DOUBLE
pesoscinesfamiliar
- id: INTEGER
- accion: INTEGER
- animacion: INTEGER
- aventuras: INTEGER
- biografica: INTEGER
- belica: INTEGER
- ciencia_ficcion: INTEGER
- cine_negro: INTEGER
- comedia_romantica: INTEGER
- docudrama: INTEGER
- documental: INTEGER
- drama: INTEGER
- familiar: INTEGER
- fantastica: INTEGER
- historica: INTEGER
- infantil: INTEGER
- misterio: INTEGER
- musical: INTEGER
- otros: INTEGER
- road_movie: INTEGER
- romantica: INTEGER
- suspense: INTEGER
- terror: INTEGER
- thriller: INTEGER
- western: INTEGER
pesoscinesfamiliar_FKIndex
usuarios_ID
pesosrestaurantes
- usuarios_ID: INTEGER (FK)
- carne: DOUBLE
- especias_picante: DOUBLE
- exoticos: DOUBLE
- guisos: DOUBLE
- pescado_marisco: DOUBLE
- verduras: DOUBLE
table_05_FKIndex1
usuarios_ID
pesosrestaurantesfamiliar
- carne: INTEGER
- especias_picante: INTEGER
- exoticos: INTEGER
- guisos: INTEGER
- pescado_marisco: INTEGER
- verduras: INTEGER
table_06_FKIndex1
usuarios_ID

Figura A.4

Apéndice B

API

En esta sección se muestra esquemáticamente las llamadas más importantes que ofrece la capa de servicios y el *Servlet* asociado.

B.1. Gestión de recomendaciones y actividades

B.1.1 Recomendadores generales

Recupera un listado del servidor con las actividades mejor valoradas y los lugares más visitados.

- **Servlet:**
  LateralServlet.java
- **URL:**
  <servidor:puerto>/TurismoAPI/lateral.do
- **Método:**
  GET
- **Parámetros:**
  Sin parámetros
- **Resultado:**
  - ***mensaje:*** mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - ***status:*** estado de la solicitud. "OK" si la petición se ha completado correctamente o "ERROR" si ha ocurrido algún error.
- **Ejemplo:**
  - **Conexión a:**

<servidor:puerto>/TurismoAPI/lateral.do

  - **Resultado**

{

‘’valorado’’ : ‘’actividad mas valorada.’’,

‘’visitado’’ : [‘’lugar1’’, ‘’lugar2’’]

‘’tamanyo’’ : 2,

‘’status’’ : ‘’OK’’

}

B.1.2 Valorar una actividad

Para una actividad concreta, recibe una valoración y actualiza la base de datos del servidor.

- **Servlet:**
  ValorarServlet.java
- **URL:**
  <servidor:puerto>/TurismoAPI/valorar.do
- **Método:**
  GET
- **Parámetros:**
  - ***tipoActividad:*** tipo de actividad valorada (cines, centros comerciales, parques de ocio, etc.).
  - ***recomendación:*** nombre de la actividad que se ha valorado.
  - ***valoración*:** valoración realizada por el usuario para cada actividad. Los valores están comprendidos entre 0 y 5.

- **Resultado:**
  - ***mensaje:*** mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - ***status:*** estado de la solicitud. “OK” si la petición se ha completado correctamente o “ERROR” si ha ocurrido algún error.
- **Ejemplo:**

- **Conexión a:**

<servidor:puerto>/TurismoAPI/valorar.do?tipo=CENTROS_COMERCIALES&recomendacion=Centro Comercial Príncipe Pío & valoracion = 4

- **Resultado:**

{

‘’mensaje’’ : ‘’Se ha guardado la valoración correctamente..’’,

‘’status’’ : ‘’OK’’

}

B.1.3 Solicitar una recomendación

Para una actividad concreta, recibe una valoración y actualiza la base de datos del servidor.

- **Servlet:**
  RecomendadorServlet.java
- **URL:**
  <servidor:puerto>/TurismoAPI/recomendador.do
- **Método:**
  POST
- **Parámetros:**
  - ***tipo:*** *debe tomar siempre el valor “PlantillaConcreta”.*
  - ***horario:*** horario en el que se espera realizar el plan.
  - ***posición*****:** posición inicial del plan. Se indicarán la *latitud* y *longitud* de la posición separadas por coma.
  - **actividades:** Tipos de actividades que el usuario quiere en el plan (museo, parques de ocio, centros comerciales, etc). Deberán ir separados por comas.
  - ***tipo de grupo:*** Modelo de grupo escogido por el usuario para una recomendación en grupo. Puede tomar entre [0,1]

elementos de este conjunto de valores: “familiar”, “pareja”, “trabajo” y “amigos”.

- ***edad:*** Edad seleccionada por el usuario para la que va destinada los planes de ocio. Puede tomar entre uno de los siguientes valores: “ninos”, “jovenes”, “jovenesplus”, “adultos” y “cualquiera”.
- ***usuarios:***
  - ***Modo Facebook:*** identificadores de los usuarios que van a participar en el plan separados por comas. El usuario con la sesión activa nunca viene en el listado.
  - ***Modo invitado:*** vacío.
- ***tipoRecomendación:*** Indica el tipo de función de agregación empleada en recomendaciones grupales. Puede tomar los valores “maxima”, “media”, “minima” o “trustmean”.

- **Resultado:**
  - ***sol:*** sólo en caso de éxito. Se trata de un vector JSON con la lista de actividades para el plan resultante y la información obtenida de cada una de ellas.
  - ***tamanyo:*** sólo en caso de éxito. Indica el número de actividades del plan devuelto.
  - ***mensaje:*** mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - ***status:*** estado de la solicitud. “OK” si la petición se ha completado correctamente, y “ERROR” si ha ocurrido algún error.

- **Ejemplo:**
  - **Conexión a:**

<servidor:puerto>/TurismoAPI/recomendador.do

  - **Resultado**

```
{
‘‘id’’:‘‘-1’’,
‘‘sol’’:[
{
‘‘tipo’’ :

‘‘MUSEO’’,
‘‘actividad’’:{
‘‘nombre’’:‘‘Museo de
‘‘direccion’’ : ‘‘Avenida de los Re...’’,
‘‘resumen’’ : ‘‘El museo se creó en abril de 1941 t...’’,
‘‘imagen’’ ‘‘web’’ : ‘‘ ’’,
‘‘horario’’ : ‘‘12:00-14:00’’,
‘‘posicion’’ : {‘‘latitud’’ : ‘‘41.43’’,‘‘longitud’’ : ‘‘-
5.66’’},
,{
‘‘tipo’’ : ‘‘CENTROCOMERCIAL’’,
‘‘actividad’’ : {
‘‘nombre’’ : ‘‘Centro Comercial Príncipe Pío’’,
‘‘direccion’’ : ‘‘Paseo de la Florida, s/n (ho...)’’,
‘‘resumen’’ : ‘‘Abre todos los días del a...’’,
‘‘imagen’’ :
‘‘horario’’ : ‘‘14:00-15:30’’,
‘‘posicion’’ : {‘‘latitud’’ : ‘‘42.42’’,‘‘longitud’’ : ‘‘-
2.68’’},
],
‘‘tamanyo’’ : ‘‘2’’,
‘‘status’’ : ‘‘OK’’}

}
```


B.1.4 Aceptar una recomendación

Para una actividad concreta, recibe una valoración y actualiza la base de datos del servidor.

- **Servlet:**
  HistorialServlet.java
- **URL:**
  <servidor:puerto>/TurismoAPI/historial.do
- **Método:**
  POST
- **Parámetros:**
  - ***plantilla:*** Se compone de un JSON que contiene el plan completado que ha realizado un usuario.
- **Resultado:**
  - ***mensaje:*** mensaje informativo ofrecido como respuesta. Especialmente útil en caso de error.
  - ***status:*** estado de la solicitud. "OK" si la petición se ha completado correctamente o "ERROR" si ha ocurrido algún error.
- **Ejemplo:**
  - **Conexión a:**

<servidor:puerto>/TurismoAPI/historial.do?

Resultado:

{

''mensaje'' : ''El historial se ha actualizado correctamente..'',

''status'' : ''OK''

}

Bibliografía

1. Wang, P. (1998). Why recommendation is special? Workshop on Recommender Systems, part of the 15th National Conference on Artificial Intelligence, vol. 15, páginas 111-113.
2. Balabanovic, M., Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Commun. ACM, vol. 40, páginas 66-72.
3. Pazzani, M.J., Billsus, D. (2007). *Content-Based Recommendation Systems,* The Adaptive Web.
4. Jameson, A. y Smith, B. Recommendation to Groups. The adaptative Web, 2007.
5. López Gómez, Javier y Bezares Álvarez, Marina y Marín Fernández, Roberto (2014) *Madrid Live: un sistema de recomendación de ocio social y contextual para la ciudad de Madrid* http://eprints.ucm.es/26502/.
6. Adomavicius, G., Sankaranarayanan R., Sen, S. y Tuzhilin A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. ACM Transactions on Information Systems (TOIS), vol. 23, páginas 103-145.
7. Leiva, J. L., Guevara, A., Rossi, C., & Aguayo, A. (2014). Realidad aumentada y sistemas de recomendación grupales.
8. Huecas, G., Salvachúa, J. (2010). Filtros colaborativos y sistemas de recomendación. http://es.slideshare.net/ghuecas/filtros-colaborativos-y-sistemas-de-recomendacin. Online, consultado en febrero del 2015, apartados 3.6.1, 3.6.2 y 3.9.4.
9. Rojas Castellanos, Y., (2014). Sistema de recomendación por filtrado colaborativo para el sistema de publicación de contenido multimedia - VideoWeb 1.0. International Journal of Innovation and Applied Studies, vol. 6, páginas 326-334.
10. Burke, R. (2002). Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, vol. 12, páginas 331-370.
11. Linden, G., Smith B., Jeremy Y. (2003). Item-to-Item Collaborative Filtering. Revista IEEE Internet Computing.
12. Filmaffinity, 2015. ¿Qué son las almas gemelas? en Preguntas más frecuentes. http://www.filmaffinity.com/es/faq.php

13. Masthoff, J. (2011). Group recommender systems: Combining individual models.
14. Quijano Sánchez, L. (2010). Impacto de los factores y organizaciones sociales en los procesos de recomendación para grupos. Proyecto Fin de Máster en Sistemas inteligentes http://eprints.ucm.es/11321/1/Memoria_Proyecto_Fin_de_Master.pdf
15. Recío García, J.A. JColobrí. Una plantaforma móvil multi-nivel para la construcción y generación de sistemas de razonamiento basado en casos. Tesis Doctoral, Universidad Complutense de Madrid, 2008.
16. Golbeck, J. Combining provenance with trust in social networks for semantic web content, 2006b.
17. Golbeck, J. Generating predictive movie recommendations from trus in social networks, 2006b.
18. Barros, A. J. da S; Lehfeld, N. A. de S. (2000). "Fundamentos de metodologia".
19. [Turner, J. C. (1982). "Towards a cognitive redefinition of the social group"]
20. Huecas G., Salvachúa J. (2010), Filtros colaborativos y sistemas de recomendación, http://es.slideshare.net/ghuecas/filtros-colaborativos-y-sistemas-de-recomendacin