2017

ESTUDIO DE TÉCNICAS DE PLANIFIACIÓN DINÁMICA DE PLANES DE OCIO

UNIVERSIDAD COMPLUTENSE
MADRID

MEMORIA DE TRABAJO FINAL DE GRADO

Realizadores:

Luis Federico Díaz Pérez

Eduardo Gil Ruiz

Carlos Alberto Martín Marcos

Jorge Elias Rashmawi Ruiz

Dirigido por:

Juan Antonio Recio García

Antonio Sánchez Ruiz-Granados

Autorización de difusión

Luis Federico Díaz Pérez, Eduardo Gil Ruiz, Carlos Alberto Martín Marcos y Jorge Elias Rashmawi Ruiz, y por otra parte, Juan Antonio Recio García y Antonio Sánchez Ruiz-Granados, como alumnos y tutores, del Trabajo de fin de Grado del Grado de Ingeniería Informática de la Facultad de Informática, autorizan mediante el presente documento a la Universidad Complutense de Madrid a difundir y utilizar solo con fines académicos (no comerciales) y mencionando expresamente a los autores del mismo, cuyos datos se verán reflejados a continuación. De la misma forma autoriza a la Universidad Complutense de Madrid a que sea depositado en acceso abierto en el repositorio institucional con el objeto de incrementar la difusión, uso e impacto del Trabajo fin de Grado en Internet y garantizar su preservación y acceso a largo plazo

Estudio de técnicas de planificación dinámica de planes de ocio

Curso Académico 2016-2017

Alumnos	**Tutores**
Luis Federico Díaz Pérez	Juan Antonio Recio García
Eduardo Gil Ruiz	Antonio Sánchez Ruiz-Granados
Carlos Alberto Martín Marcos	
Jorge Elias Rashmawi Ruiz	

Firma de los alumnos	**Firma de los tutores**

Tabla de contenido

Índice de figuras

Índice de tablas

AGRADECIMIENTOS

En primer lugar, el equipo desarrollador de este proyecto quiere agradecer a la Universidad Complutense de Madrid por habernos unido en este trabajo fin de carrera ya que todos nos conocimos en la universidad.

Por otro lado, agradecer a nuestras familias, amigos y conocidos el apoyo que nos han dado el día a día, ya no solo durante el desarrollo de este TFG sino durante toda la carrera. Gracias a ellos hemos conseguido superar todos los retos a los que nos hemos tenido que enfrentar.

También queremos dar las gracias a todos los profesores, técnicos, personal de la cafetería, personal de la secretaría, consejería y a todos los empleados de la Facultad de Informática.

Queremos agradecer también a todas las personas que nos han ayudado aportando ideas para la aplicación y han seguido de cerca este desarrollo y en especial a aquellas que participaron en la evaluación de usuarios.

Por último, un especial agradecimiento a nuestros tutores Juan Antonio Recio y Antonio Sánchez Ruiz-Granados por ayudarnos, apoyarnos y ante todo aguantarnos en este proceso largo de trabajo fin de grado, sin ellos el trabajo no sería lo mismo.

RESUMEN

Este proyecto, 'Olétrip', es la continuación de una versión anterior cuyos autores fueron Raquel Álvarez Hernández, Qiang Sun y YanYan Cheng.

Nuestro objetivo en el proyecto es mejorar las recomendaciones de un sistema de planes de ocio usando diferentes algoritmos para calcular las rutas más óptimas con la ayuda de un validador personalizado

Los algoritmos nuevos implementados en la aplicación son los siguientes:

- Algoritmo aleatorio: se implementó para ayudarnos a familiarizarnos con el código heredado del proyecto anterior, y posteriormente resultó útil para la implementación del algoritmo genético.
- Algoritmo de primero el mejor (BFS): se usa para la búsqueda de la ruta más corta después de que el usuario haya especificado todos los detalles del viaje.
- Algoritmo genético: este algoritmo se realizó con la idea de construir un plan óptimo, a pesar de saber que requiere de bastante tiempo y consumo de recursos... pero finalmente se optó por ese algoritmo para los viajeros que saben a qué ciudad quieren ir pero que no saben qué visitar.

Se explicará las ventajas y desventajas que nos supone optar por un algoritmo u otro, el cómo se ha implementado cada uno de ellos y los cambios que han sido necesarios abordar con respecto a la anterior versión del proyecto.

Incorporaremos a Olétrip un validador de planes que se encargará en la parte del *backend* de considerar la eficiencia que tiene un plan partiendo de unos requisitos que hemos definido.

Será de gran importancia esta nueva funcionalidad de validación ya que los algoritmos implementados manejan una cantidad importante de planes y de esta manera nos ayuda a evaluar qué planes son más eficientes.

La interfaz utilizada será la misma que en su anterior versión, incorporando una interfaz nueva que sirve para el recomendar de planes a los usuarios indecisos.

También hemos incorporado un simulador de planes para los usuarios Admin que ayuda a la hora de implementar alguna funcionalidad nueva en la aplicación.

De esta manera realizaremos una serie de evaluaciones con el usuario realizando un estudio de los distintos algoritmos, analizando así los resultados obtenidos y la satisfacción del usuario.

Palabras claves

algoritmo aleatorio, algoritmo BFS, algoritmo genético, validador, simulador, base de datos.

ABSTRACT

‘Oletrip’ is a touristic App that was created las year by the students Raquel Álvarez Hernández, Quiang Sun y YanYan Cheng as a final project.

This year our teams object is to improve this App by making better recommendations of a leisure plans system, using different type of algorithms to calculate the best road for a customer plan, with the help of a validator that was personalized by us.

The new algorithms that we implemented in our app are:

- Random algorithm: it was our first implementation in the app, it helped us as an introduction to the app developing because as we know we took a started app. Then it had it benefits because we needed a random algorithm for the implementation of the Genetic algorithm.
- Best First Search: we use it for the searching of the shortest route after the user specify all the details of the trip.
- Genetic algorithm: this algorithm was made with the idea of making the best plan, the inconvenient of using this algorithm is that it takes a lot of time building the best trip and consumes many resources at the same time. But we finally had the idea to use it to recommend people that don’t know what to visit in the cities that they chose.

We will explain the advantages and disadvantages of using every algorithm, how it was implemented and the changes that we needed to do to integrate our code to the last year’s version App.

Olétrip now has a validator that was implemented to help us to know what plan is better, for this validator we defined some requirements that we considered as a team to value the plans.

This functionality is so important for the new algorithm’s that we developed because of the many plans that these algorithms generates.

The app design will be the same except for some new incorporations that we developed that are:

A new interface that contains a planning recommendation for users that don’t know what to visit in the cities that they will travel to.

A developer option that helps to simulate a trip, for this option you need to have an admin role and it helps for the implementation of a code when you need an example to see if the code works.

Finally, we will contact some users to try our app and tell us what planes they like more depending of the algorithm that we use including the algorithms that were developed in the previous version, and we will analyze the results

Key words

Random algorithm, Best First Search algorithm, Genetic algorithm, Validator, Simulate, Database.

1. INTRODUCCIÓN Y MOTIVACIÓN

‘Olétrip’ es una aplicación web enfocada a aquellas personas que les gusta viajar. Es una herramienta que les permite organizar los sitios que desean visitar.

Organizar un viaje es una tarea realmente complicada si eres una persona a la que no le gusta perder tiempo y está interesada en visitar gran parte de los sitios turísticos y rincones de una ciudad.

Existen muchas webs que ofrecen rutas preestablecidas con los lugares típicos que puedes visitar o te ofrecen tours de lo que te quieran enseñar a un coste por lo general barato, pero con una ruta de la cual no te puedes desmarcar.

Con esta herramienta ofrecemos a los viajeros la oportunidad de obtener información sobre qué sitios ver en las diferentes ciudades establecidas en la web, ya que en la mayoría de casos un usuario no conoce todos los lugares que tiene una ciudad.

Con esta información el usuario podrá elegir sus puntos turísticos y la aplicación le recomendará su mejor ruta para optimizar su tiempo. Si el usuario no tiene claro qué lugares visitar o prefiere optar por un plan diseñado por la aplicación, ésta le ofrece un recomendador de actividades diseñadas en función de las ciudades a las que vaya a ir y los días de los que disponga.

Una vez el usuario tenga su plan podrá compartirlo con el resto de viajeros que le acompañen y puedan valorarlo para confirmar su interés y votar por las actividades que se van a realizar.

En estos momentos la aplicación está centrada en Andalucía por lo que las únicas ciudades y actividades que aparecen en la web son de esta Comunidad, pero está diseñada para que sea escalable y poder añadir en el futuro más Comunidades con sus respectivos lugares de interés.

1.1. OBJETIVO

El objetivo que pretendemos alcanzar con este proyecto es desarrollar un planificador de planes de ocio mediante la implementación de diferentes algoritmos.

Es importante destacar que este era un proyecto heredado de compañeros de la Facultad de Informática del año pasado, lo que ha llevado a invertir tiempo en analizar el código y adaptarlo a nuestras necesidades.

Para lograr el objetivo hemos realizado las siguientes tareas:

1. Configuración del entorno de trabajo en plataforma Windows y Mac.
2. Análisis y comprensión del código heredado.
3. Estudio de posible implementación de planificador[1].
4. Estudio e implementación del algoritmo BFS.
5. Estudio e implementación del algoritmo genético.
6. Evaluación de calidad de los planes para analizar lo óptimo que es un plan.
7. Simulador para agilizar las pruebas de desarrollo.
8. Recomendador de planes alternativo.
9. Mejoras con respecto a la anterior versión.

Con todos estos objetivos específicos se pretende conseguir:

1. Obtener los gustos y preferencias de las actividades que realizan los usuarios.
2. Generar planes más óptimos.
3. Mejorar la experiencia del usuario aportándole planes alternativos a los elegidos.
4. Ofrecer al usuario un recomendador de planes lo más acorde posible a sus preferencias.
5. Facilitar la labor de los futuros desarrolladores con la simulación de planes.

1.2. ESTRUCTURA DE LA MEMORIA

En la memoria se describen las siguientes partes:

En primer lugar, se aborda en profundidad el tema de los sistemas de recomendación y su implicación con el ocio y el turismo. En definitiva, trata sobre el estado del arte de los sistemas de recomendación.

En el cuarto capítulo (‘Tecnologías aplicadas’) se detalla las tecnologías que hemos utilizado tanto a nivel de desarrollo de *backend* y *frontend* como a nivel de gestión por parte del equipo.

Posteriormente se realiza una exhaustiva descripción del funcionamiento de la aplicación para los distintos algoritmos, incluyendo la nueva funcionalidad del

simulador y el recomendador de planes ilustrándolas con diferentes figuras para mejor entendimiento.

En el siguiente capítulo, ‘Arquitectura del sistema’ se explica básicamente qué es un plan y el importante uso que se le da a la base de datos. Dentro de este capítulo también se habla sobre la interfaz que se ha utilizado tanto en el uso manual de la elección de planes como en el simulador que proporciona planes automáticos.

En el capítulo siete, ‘Mecanismo de creación de rutas’ se explica con suficiente detalle el sistema de validación que ha sido empleado a la hora de ‘puntuar’ los planes que han sido recomendados al usuario, así como cada uno de los algoritmos que están implementados en la aplicación, así como sus principales ventajas y desventajas.

Se llevará a cabo una evaluación de usuarios donde se realiza un estudio con los diferentes planes que ofrecen los algoritmos, anotando los resultados obtenidos con la ayuda de gráficas y posibles mejoras que ayude a mejorar la aplicación.

En otro capítulo, se redactarán una serie de conclusiones y mejoras que puedan ser implementadas en una versión posterior en el futuro. Estas, son descritas tanto en castellano como en inglés.

En el capítulo diez cada miembro del grupo indica las aportaciones que ha ofrecido a lo largo del proyecto.

Finalmente se redacta una serie de apéndices donde se incluyen figuras, configuración del entorno de desarrollo, instalación del *backend*, *frontend* y base de datos, bibliografía utilizada, etc.

2. INTRODUCTION AND MOTIVATION

‘Oletrip’ is an App focused to people who likes to travel. This tool helps users to organize the places that they want to visit.

Planning a trip isn’t an easy task to do, if you are a person that doesn’t like to waste time and you’re interested in visiting the most places possible in your trip and enjoy it.

There are a lot of Webs that offers default routes with the typical places that the tourist can visits or offers tours that shows the travelers nice places with a low cost, but you have to follow the route from the beginning till the end.

With this App we offer the travelers the opportunity to get information about the places in the different selected cities in the web, because of most people doesn’t know all the places to visit in each city.

With this information the user could select the touristic points and the app will recommend the best route to enjoy the most time possible. If the user doesn’t know what to visit or he prefers to select a route that is designed by the App, this App offers a new functionality that allows the user to select the cities and the days that he wants to make the trip and the app will recommend the user the best route.

Once the user selects the trip to do he share it with his travel mates and they can value the trip to confirm their interest and they can vote to what activity’s they want to do.

At the moment the App only includes places in Andalucia, so that all the places and activities of the app are there, but the App is designed to includes a much more places in the future.

2.1. GOALS

Our goal in this project is to develop a leisure trip planner by the implementation of a different algorithms.

It is important to highlight that this project is the second part of a last year project, so we needed to invert time to familiarize with the code and to understand the functionality of the app so we could adapt it to our necessities.

To reach our goal we realized the next tasks:

1. Setting up the working environment on Windows and Mac platforms.
2. Analyze and understand the inherited code.
3. Studying the possibility of implementing a planner
4. The study and the implementation of a BFS Algorithm.
5. The study and the implementation of a Genetic Algorithm.
6. Evaluate the quality of a plan to analyze how much the plane is optimum.
7. Simulator to help the developers for testing.
8. An alternative recommender for trips.
9. Improving some functionalities from last version App.

With all these specified goals we pretend to achieve:

1. Get the users preferences of the activities to do.
2. Generate optimized plans.
3. Improve the users experience by providing them alternative plans to the chosen ones.
4. Offers the user a plan recommender based on their preferences.
5. Help future developers by using the simulator when they are testing new functionalities.

2.2. MEMORY STRUCTURE

In the memory we describe the next parts:

In the first place we talk about recommendation systems and its Implications with tourism and leisure plans. Basically it deals with the state of art recommendation systems.

In Chapter four (“Applied Technologies”) we detail the technologies that we used for the backend side and the frontend side, also we talk about the distribution management made by the team.

After that we made an exhaustive description about how the application works with the different algorithms, including the new functionality of the simulator and the plan recommender showing some image examples to be more clarified.

In the next chapter, 'System architecture' we explain basically what is a plan and what does it contains and the importance of the use of the data base. Inside this chapter we also explain the interface that we used to build a plan as for choosing the plans by the user or by the App simulator.

In chapter seven, 'Route creation mechanism' we explain the quality validator of a plan in detail that was used to rate the plans that the App recommended, also we explain all the algorithms that were implemented in the app and their advantages and disadvantages.

In the next chapter we explain the process of a user evaluation of the app that we developed for real users and the study of the algorithms that were implemented, contributing graphics and conclusions for possible improvements in the future for the App.

In this chapter, we talk about some conclusions that can contribute better functionality's and uses in the near future, these conclusions were explained in English and in Spanish.

In chapter 10 each member of the group explains his contributions in the Project during the development.

Last chapter contains the appendices that includes images, Configuration of the development environment, the installation of the code and the data base, the bibliography used, etc.

3. ESTADO DEL ARTE

3.1. SISTEMAS DE RECOMENDACIÓN

Estos sistemas han tenido gran aceptación ya que son útiles para filtrar la información y reducir el número de elementos que el usuario tiene que evaluar. Se deben tener en cuenta diferentes aspectos para diseñar sistemas de recomendación. A continuación, se describen algunos de estos aspectos, así como ejemplos de sistemas actuales.[2]

Una de las variables importantes es el volumen de la información, ya que de éste depende el detalle de las recomendaciones.

Existen también implicaciones sociales. Establecer un perfil de intereses de las recomendaciones puede ocasionar problemas en casos de imparcialidad. Además, la privacidad de los participantes debe tomarse en cuenta ya que no todos los usuarios gustan de exhibir sus preferencias o de no ser reconocidos por su aportación.

Existen multitud de paradigmas para la selección de elementos. Dos de ellos son, basados en contenido y filtrado colaborativo. Detallamos a continuación ambos tipos.

3.1.1 Recomendaciones basadas en contenido

Los sistemas de recomendación basados en contenido, emplean técnicas de recuperación de información.

Este método de recomendación presenta algunos problemas como la sobre-especialización; el sistema sólo muestra al usuario elementos similares a los que ya ha visto anteriormente. Algunas veces este problema es resuelto agregando a la búsqueda aleatoriedad como por ejemplo utilizando algoritmos genéticos.

En este tipo de sistemas es importante las extracciones de los atributos, por ejemplo, en nuestro caso, será fundamental obtener y comparar las distancias y tiempos de cada uno de los sitios escogidos para obtener la mejor ruta.

3.1.2 Filtrado colaborativo

Este tipo de sistemas de recomendación recopila información que ha sido exitosa para otros usuarios con el propósito de ahorrar tiempo y coste.

En el filtrado colaborativo, el sistema no analiza los elementos evaluados, sino que las recomendaciones se basan solamente en la similitud entre usuarios. Esto trae consigo algunos problemas, como se comenta a continuación.

El problema denominado *cold-start*[3] aparece si el usuario es nuevo, necesita completar información para que el sistema sea capaz de encontrar un grupo de usuarios similares.

Cuando un usuario llega al sistema, no es posible hacerle recomendaciones hasta que su perfil sea lo suficientemente completo para encontrarle un grupo de usuarios similares. Si a esto le añadimos que, si los gustos del usuario son poco comunes, encontrar estos usuarios será una tarea complicada. Podemos decir que las recomendaciones dependen directamente del número y variedad de usuarios en el sistema.

Se pueden definir tres tipos en este sistema de recomendación:

- Basado en memoria: calcula la similitud entre dos usuarios haciendo uso de mecanismos como la similitud de correlación de Pearson o similitud basada en el coseno entre vectores.
- Basado en modelo: Este método utiliza algoritmos de aprendizaje automático con el fin de encontrar patrones basados en los datos de entrenamiento.
- Híbridos[4]: Combina algoritmos basados en memoria y basado en modelo con el fin de obtener la mejor eficiencia de las recomendaciones. Podemos diferenciar dos técnicas dentro de los sistemas de recomendación híbridos:
    - Uso de pesos. Para cada resultado de cada técnica de recomendación se le da una puntuación y a través de un consenso se combinan los resultados. Esta combinación puedes ser aplicada utilizando una función que pondera la importancia de cada uno de los sistemas de recomendación.
    - Uso de técnicas de conmutación. El sistema conmuta entre las técnicas de recomendación en función de la situación actual según algún criterio de conmutación analizando los resultados obtenidos en evidencia pasada. Si aplicando una de las técnicas, esta no proporciona resultados, se da paso a aplicar la siguiente técnica. La desventaja de esta técnica es que, si al aplicar la primera técnica

si se encuentran resultados, las otras técnicas no se ejecutaran y es posible que se queden sin mostrar algunos resultados relevantes para el usuario.

3.2. SISTEMAS DE RECOMENDACIÓN RELACIONADOS CON EL OCIO Y TURISMO

Está previsto que el sector del turismo en España en 2017 sea más favorable con respecto al año pasado ya que se espera una mayor llegada de turistas según las encuestas.[5]

Es por ello que sea necesario, para mayor comodidad de estos turistas que no conocen de buena mano nuestros lugares más conocidos, la existencia de aplicaciones o sitios web que ayuden a recomendar hoteles, lugares donde comer, sitios a visitar, etc.

Se detallan a continuación algunas de ellas:

Mi nube

Esta aplicación hace recomendaciones sobre qué lugares visitar, comer u hospedarse. Utiliza las valoraciones de los usuarios que han visitado algún lugar.[6]

Una de las funcionalidades interesantes es que se puede tanto buscar vuelos como buscar lugares de interés turístico. En la aplicación el usuario puede compartir sus experiencias con los demás de manera que otro pueda basarse en estos comentarios para declinarse por un lugar u otro.

El usuario puede añadir fotos de sus viajes y se pueden observar los lugares que han visitado personas de nuestro entorno, así como comentarios y fotos añadidas por ellas.

El problema es que esta aplicación no te proporciona una ruta con los lugares escogidos, sino que simplemente muestra los lugares que se quiera visitar, como se muestra en la *figura 1*.[7]

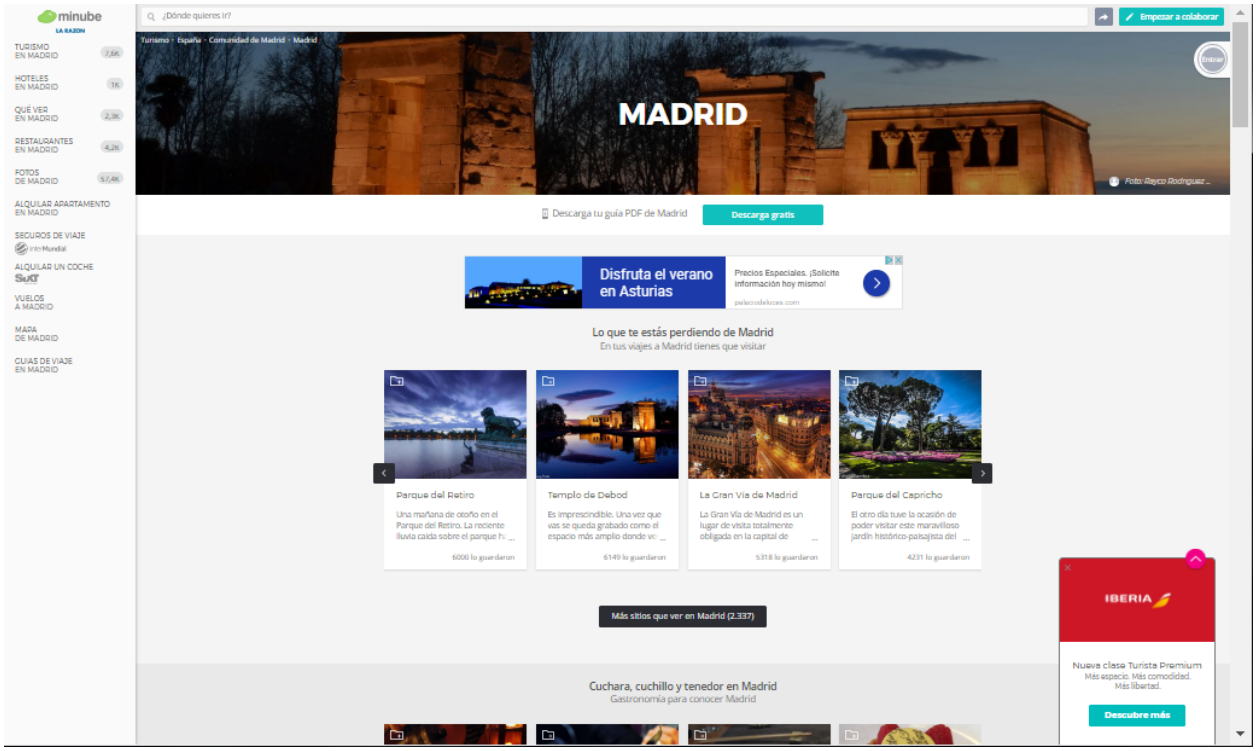

Figura 1 Mi nube

TripAdvisor

Es un sitio web que proporciona información relacionada con viajes incluyendo también un foro de viajeros. Utilizando técnicas de filtrado colaborativo, haciendo recomendaciones basados según las valoraciones numéricas y descriptivas sobre los sitios que los usuarios han registrado.

Esto es útil debido a que las actividades que se muestran en la página principal, están bien valoradas por los usuarios. Sin embargo, TripAdvisor[8] no es útil si el usuario desea realizar un viaje planificado (días, horas, etc.) con actividades que él ha elegido. Tenemos un ejemplo en la *figura 2*.

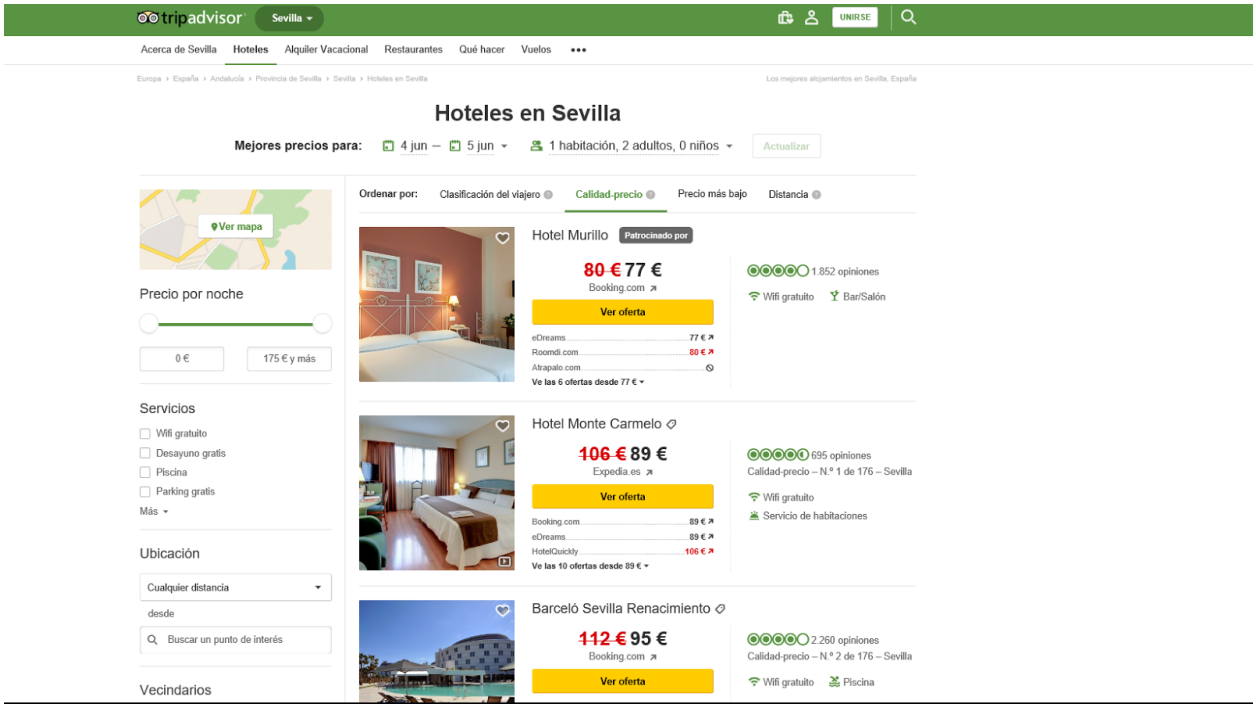

Figura 2 TripAdvisor

Guía Repsol

Es un sitio web que permite seleccionar lugares una vez escogida la ciudad dando al usuario la posibilidad de guardar el plan con los lugares deseados. Proporciona información sobre ellos, así como su ubicación.

Por contra, esta web no proporciona una ruta, es decir, día y hora en la que realizar la actividad, sino que el plan que proporciona simplemente consiste en la visualización de la información de cada lugar.

El sistema muestra recomendaciones, pero el usuario no puede observar valoraciones ni opiniones de otros usuarios que hayan visitado dichos lugares como podemos ver en la *figura 3*.

La aplicación te permite crear un plan y modificarlos realizando anotaciones.[9]

Figura 3 Guía Repsol

En la siguiente tabla se muestra una comparativa resumida de las tres aplicaciones:

Aplicación	Ofrece rutas	Basada en valoraciones del usuario	Sistema de recomendación utilizado
Mi nube	No	Sí	Filtrado colaborativo
TripAdvisor	No	Sí	Filtrado colaborativo
Guía Repsol	No	No	Filtrado colaborativo

Tabla 1 Comparativa de aplicaciones relacionadas con ocio y turismo

4. TECNOLOGÍAS APLICADAS

En este capítulo haremos una descripción general de las tecnologías aplicadas en el desarrollo de la aplicación. La mayoría de las decisiones de las tecnologías aplicadas han sido tomadas por los creadores de la aplicación, pero en nuestro caso hemos añadido alguna herramienta que hemos requerido para la evolución del proyecto.

La explicación de las tecnologías aplicadas se va a dividir en dos apartados, la parte *Backend* que es la capa de acceso a datos y la parte *Frontend* que representa la capa de presentación.

4.1. BACKEND

Java EE (Java Enterprise Edition)[10] es un conjunto de tecnologías y *APIs* (Application Program Interface) con el lenguaje *JAVA* que nos permite diseñar aplicaciones de gran escala y distribuidas de tipo multicapa sobre servicios de tipo Web.

La comunicación entre cliente y servidor se realiza mediante peticiones a través del protocolo *HTTP* implementado con un servicio *RESTful* y retorno de respuesta en formato *JSON*.[11]

Este servicio web está construido con *Jersey*, que es una implementación y mejora de la *API JAX-RS* para diseñar aplicaciones Web con métodos definidos mediante uso explícito de los métodos *HTTP*.

Esto nos permite hacer peticiones mediante los protocolos *POST* y *GET*.

La aplicación cuenta con el uso de *Hibernate* para el mapeo entre clases de entidades Java y las tablas de la base de datos y los correspondientes tipos de datos de Java y *SQL*.

Hibernate es un *framework* de código libre bajo licencia para entornos Java.

Este *framework* cuenta con unos métodos predefinidos para realizar operaciones frecuentes de datos, *save*, *delete*, *update*…

Para la realización de consultas más básicas se usa *HQL*.

Otra de las utilidades con las que cuenta la aplicación es el *framework Log4j* (Log for Java) que sirve para tener un fichero log para seguir las trazas de la aplicación.

El servicio está montado sobre un servidor local basado en *Apache tomcat versión 8.5.*

El sistema de gestor de base de datos fue *Mysql Workbench*, elegido por su facilidad de utilización y configuración.

Para el Desarrollo de la aplicación hemos utilizado el entorno de desarrollo *Eclipse NEON.1* junto las librerías de *maven* dependencias y el *JRE 1.8*.

Por último, se ha utilizado *MySQL Connector* que es un *Framework* que se encarga de conectar la aplicación con la base de datos.

4.2. FRONTEND

La Aplicación web ha sido construida con el lenguaje *HTML5* que es interpretado por los navegadores y diseñada para ser responsive y que se adapte a cualquier dispositivo móvil, tablet u ordenador. Para darle estilo a la aplicación se utilizó el lenguaje *CSS3*.

La aplicación también cuenta con *Bootstrap* que es un *framework CSS* y *Javascript* que permite dar forma a un sitio web que incluyen tipografías, botones, cuadros, menús y otros elementos que pueden ser utilizados en cualquier sitio web.[12]

Bootstrap es una excelente herramienta para crear interfaces de usuario limpias y totalmente adaptables a todo tipo de dispositivos y pantallas, sea cual sea su tamaño.

Además, *Bootstrap* ofrece las herramientas necesarias para crear cualquier tipo de sitio web utilizando los estilos y elementos de sus librerías.[13]

Desde la aparición de *Bootstrap* 3 el *framework* se ha vuelto bastante más compatible con desarrollo web *responsive*, entre otras características se han reforzado las siguientes:

- Soporte bastante bueno (casi completo) con *HTML5 y CSS3*, permitiendo ser usado de forma muy flexible para desarrollo web con unos resultados muy favorables.
- Se ha añadido un sistema *GRID* que permite diseñar usando un *GRID* de 12 columnas donde se debe plasmar el contenido, para desarrollar *responsive* de forma mucho más fácil e intuitiva.

A continuación, explicamos algunas de las librerías utilizadas:

- *Chart.js* es una librería utilizada para la creación de gráficas. Aprovecha la característica *Canvas* de *HTML5* y son adaptables entre dispositivos.
- *Slick.js* es una librería realizada con *JQuery* que se encarga de dar formato de visualización a los planes separados por días.
- *Moment.js* es una librería para dar formato a fechas a través del paso de un parámetro en milisegundos.

Otro de los lenguajes usados en la aplicación es *JavaScript* que es un lenguaje de programación que no requiere compilación y que es utilizado en el lado cliente del desarrollo, permite implementar funciones para la mejora de la interfaz web.

JavaScript también cuenta con una extensión que se llama *Jquery*[14] que se le ha dado mucho uso en la aplicación y es uno de los complementos más esenciales para el desarrollo web, ya que nos facilita mucho el desarrollo de aplicaciones y es compatible con todos los navegadores.

Por otra parte, nuestra aplicación contiene una biblioteca de componentes de *jQuery* que se llame *jQuery UI* que se encarga de añadir unas funcionalidades a nivel visual como por ejemplo el *Sortable* para ordenar los elementos o el *Draggable* que sirve para hacer que un elemento sea arrastrable y posicionarlo donde queramos.

Nuestra aplicación también utiliza *AJAX* que es una técnica que nos permite mediante unos programas escritos en *JavaScript* la comunicación asíncrona entre un servidor y un navegador en formato *XML*.

Por otra parte, para interpretar los datos entre el cliente y el servidor utilizamos *JSON* que nos permite representar datos de distintos tipos de forma sencilla a la hora del intercambio representadas con clave y valor.

4.3. HERRAMIENTAS DE GESTIÓN Y DESARROLLO

4.3.1. REPOSITORIO DE DATOS

Para organizar el código del proyecto es imprescindible el uso de un repositorio de código donde se pueda controlar tanto las versiones de un proyecto como las tareas que desarrolla cada uno de los miembros.

Hemos decidido usar como repositorio de código *Bitbucket* basado en *git* (*figura 4*). Esta herramienta ofrecida por *Atlassian* es un sistema de control de versiones distribuido. Hemos decidido usar este repositorio ya que ofrece la posibilidad de crear repositorios privados gratuitos de hasta 5 miembros por lo tanto se adapta perfectamente a nuestras necesidades.

Además, este repositorio ya se había usado y se usa en la actualidad por algunos miembros del grupo por lo que consideramos que era la mejor opción.

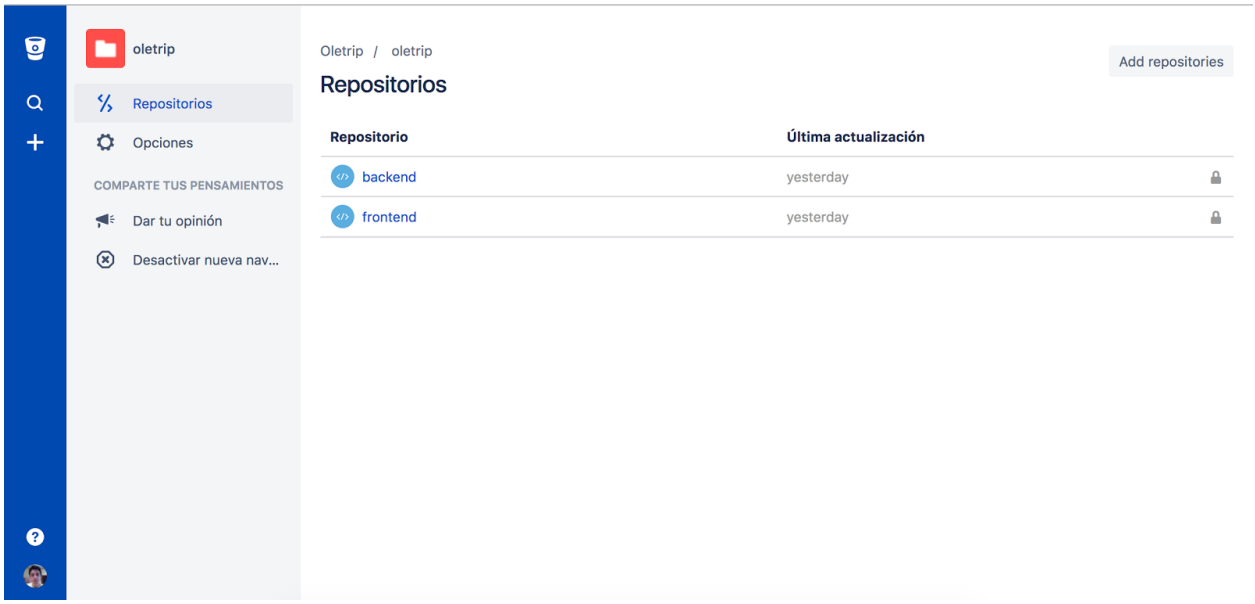

Figura 4 Repositorios

Este mecanismo nos permite a cada miembro del grupo tener una copia del proyecto para poder trabajar sobre ella e irla compartiendo con el resto de miembros del grupo y descargarse lo que el resto de compañeros comparte.

El primer paso fue crear un proyecto donde poder unificar los dos repositorios de código y después se creó un repositorio para la parte de *backend* y otro para la parte de *frontend*.

Para ambos repositorios se creó una rama master llamada producción (*figura 5*) donde estaría la versión más estable de la aplicación y otra de preproducción donde podríamos ir subiendo cada parte de desarrollo que fuésemos finalizando y de donde poder obtener el desarrollo del resto de compañeros.

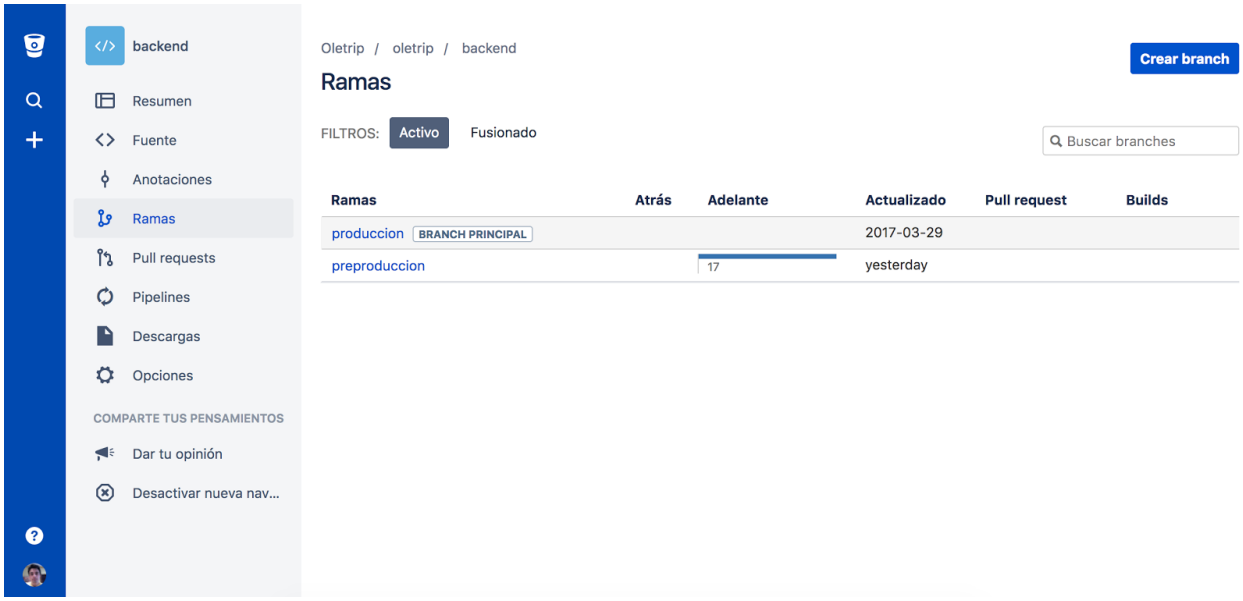

Figura 5 Ramas

El manejo del repositorio lo hemos realizado desde la herramienta *SourceTree* (*figura 6*). Esta herramienta permite que cada miembro del grupo pueda gestionar de manera visual las ramas compartidas de preproducción y de producción y poder crear nuevas ramas para los nuevos desarrollos que se hagan.

Se ha decidido que cada miembro del proyecto tenga una rama de desarrollo donde desarrollar cambios puntuales y generar nuevas ramas a partir de la rama master para cada desarrollo que se llevase a cabo en paralelo. De esta manera podemos separar perfectamente cada desarrollo e incluirlo a las ramas principales cuando se fuese finalizando.

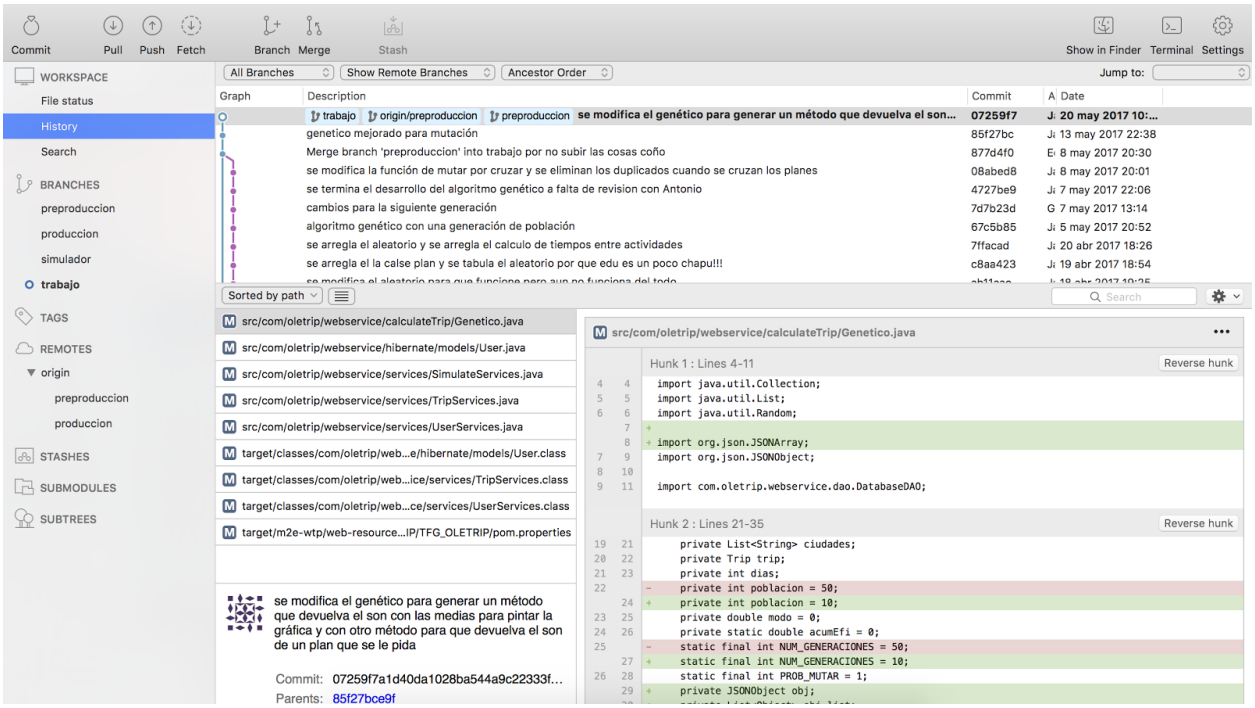

Figura 6 sourceTree

4.3.2 GESTIÓN DE TAREAS

Es necesario para cualquier equipo de desarrollo contar con alguna herramienta de gestión de proyectos, ayuda mucho a la hora de saber en qué punto se encuentra el proyecto, la aportación de cada miembro al proyecto, el reparto justo de las tareas y sobre todo en la organización del equipo.

Por esos motivos el equipo ha decidido optar por una herramienta de gestión de proyecto llamada *TRELLO*.

TRELLO es una herramienta que sirve para organizar las tareas del proyecto en un tablero, ese tablero se le asigna un nombre significativo que en nuestro caso hemos optado por poner los estados de las tareas. Por ello hemos creado 4 tableros con los siguientes nombres: Por hacer, en curso, en revisión y hecho.

Inicialmente todas las tareas(tickets) creadas se asignan al tablero de Por hacer, dentro de cada tarea se pueden adjuntar archivos, crear sub-tareas, ponerle fecha de caducidad, asignar etiqueta, hacer seguimiento, dejar comentarios etc.

Hemos creado 4 etiquetas con sus correspondientes colores para asignar a las tareas que son los siguientes:

- Código BACK
- Código FRONT
- Base de datos
- Memoria

Nuestro equipo ha decidido que un miembro se ocupe de la tarea de administrar el TRELLO, creando las tareas que se hablan durante la reunión semanal del grupo y asignarlas a los miembros del grupo de forma que todos los miembros trabajen de forma igualitaria y organizada.

En la siguiente *figura 7* se muestra la vista general de las tareas del TRELLO:

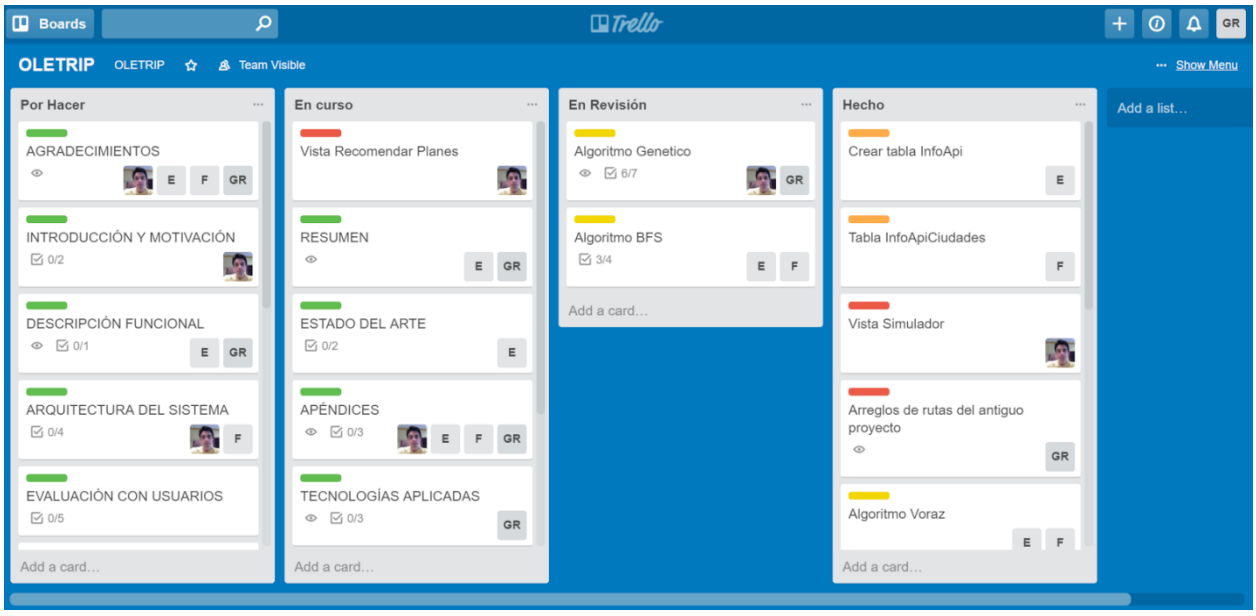

Figura 7 Trello

5. DESCRIPCIÓN FUNCIONAL

A continuación, detallaremos un ejemplo de cómo un usuario cualquiera puede usar la aplicación Olétrip.

5.1. EJEMPLO DE USO

Nada más arrancar la web se le mostrará la pantalla que se muestra en la *figura 8*.

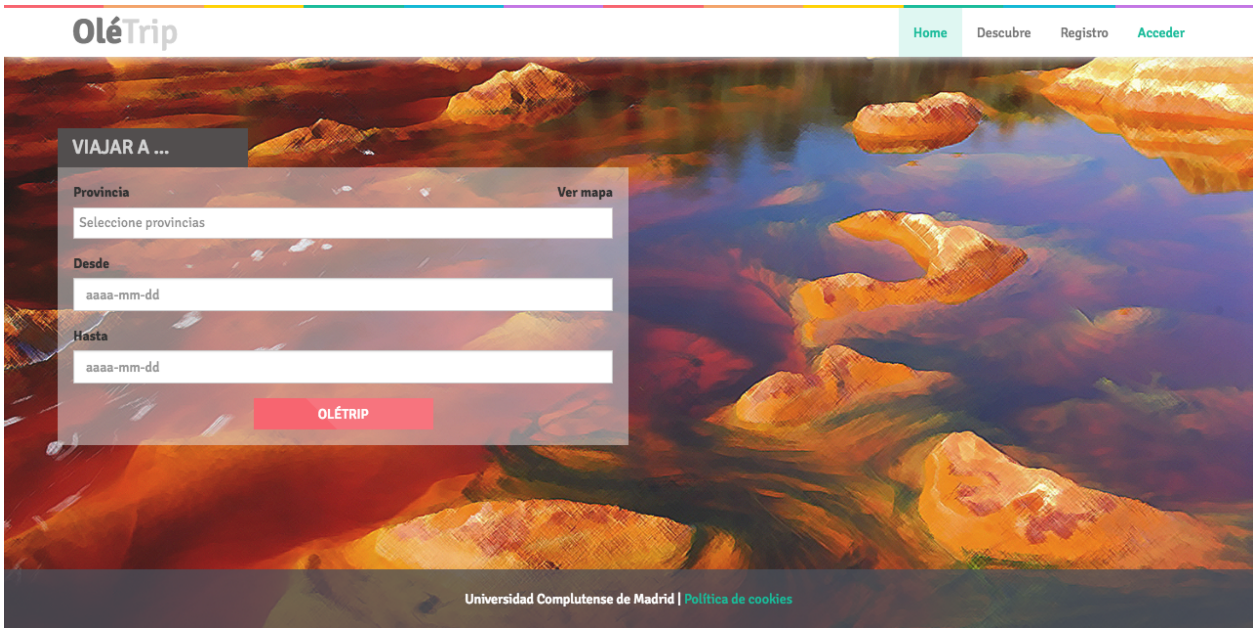

Figura 8 Home

La aplicación cuenta con un menú superior situado en la parte superior derecha desde donde el usuario encontrará:

- Home: acceso rápido al home de la web.
- Descubre: donde el usuario podrá consultar información sobre las ciudades y las actividades que hay dadas de alta actualmente en la aplicación, como podemos ver en la *figura 9*.

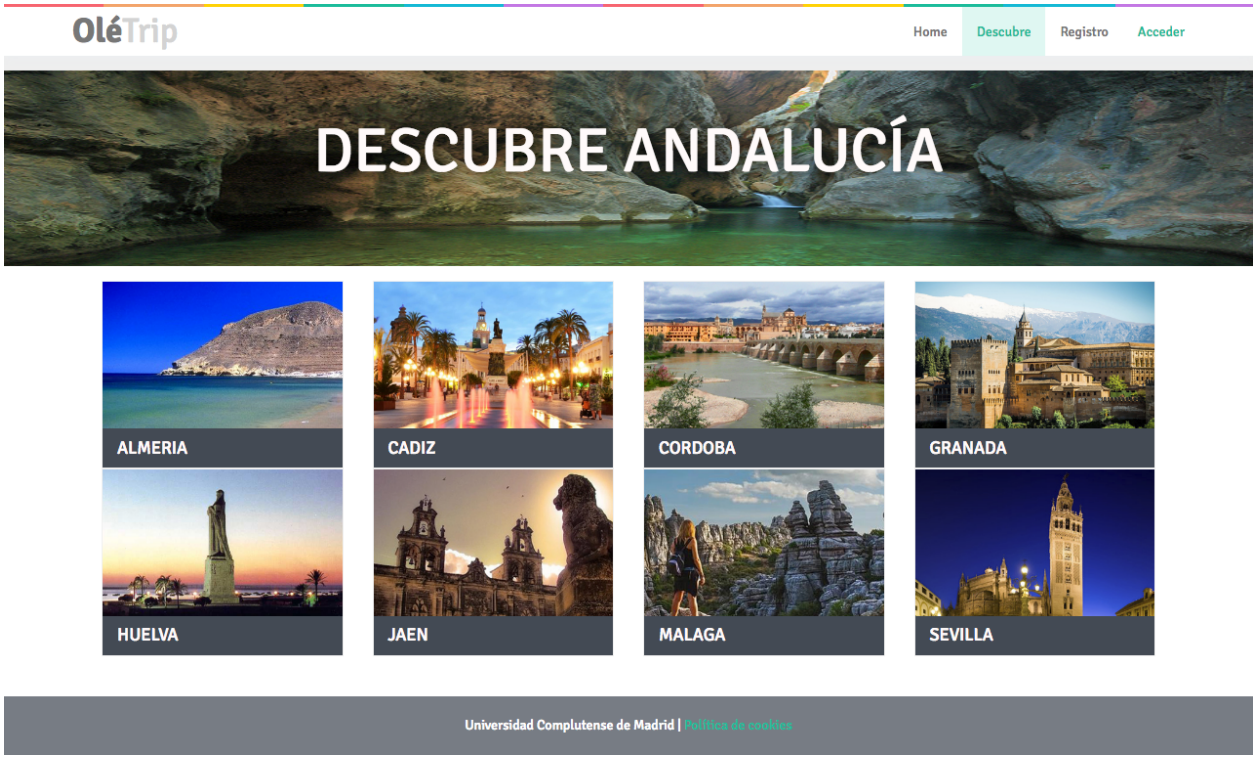

Figura 9 Descubre

- Registro: Si el usuario aún no está logado en la aplicación se mostrará esta opción para que se dé de alta en la aplicación (*figura 10*), y en caso de estar logado el botón pasa a ser 'Mis rutas' donde el usuario puede consultar las rutas que tenga guardadas.

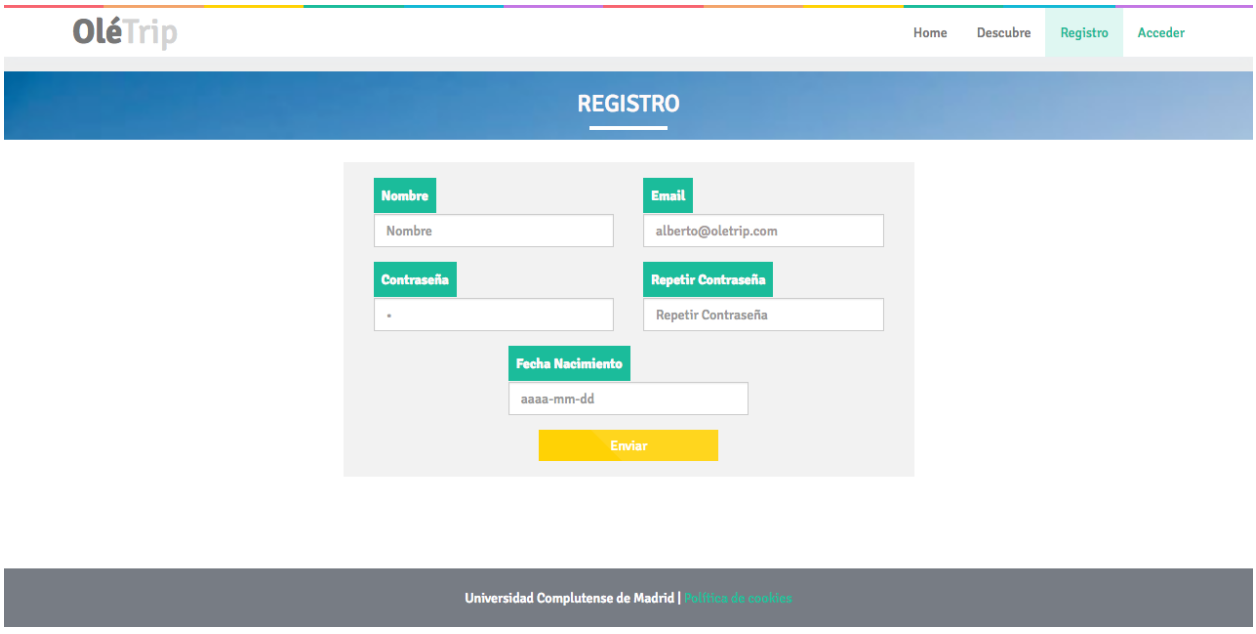

Figura 10 Formulario alta de usuario

- Acceder: Desde este enlace el usuario podrá logarse en la aplicación, (*figura 11*). En caso de estar ya logado le aparecerá un botón para desconectarse de la aplicación.

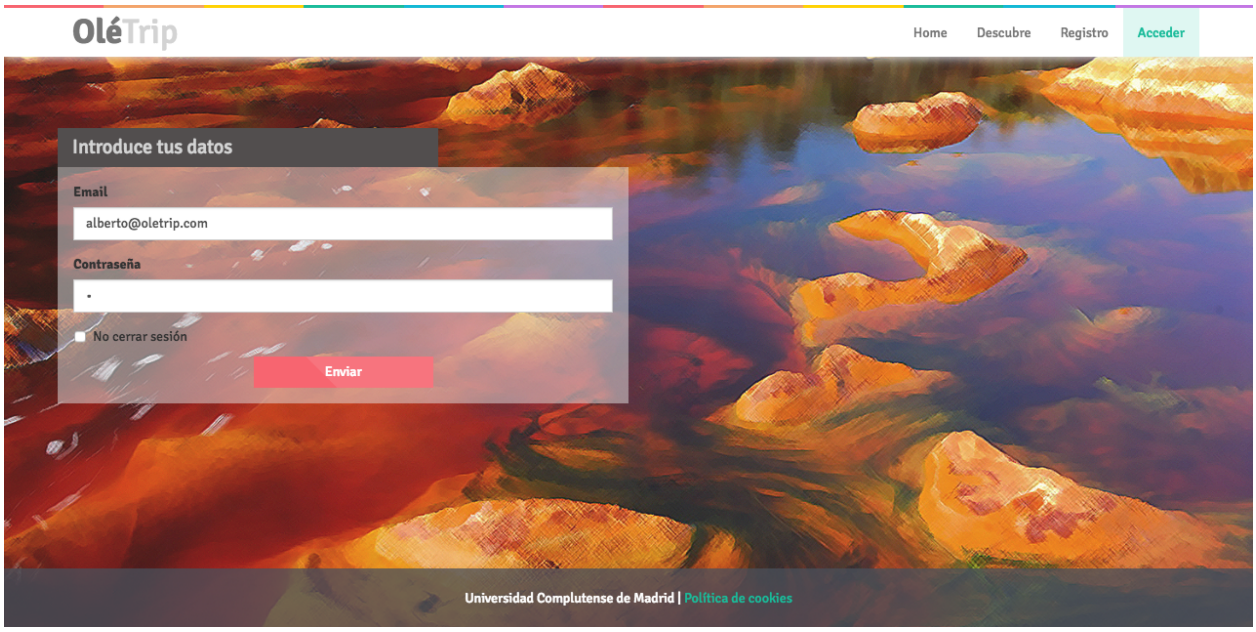

Figura 11 Login

Desde la Home el usuario puede comenzar a preparar su viaje. Para ello simplemente debe seleccionar todas las ciudades que quiere visitar y los días en los que lo quiera hacer. Una vez elegido debe pulsar el botón 'Enviar que le llevará a la siguiente página (*figura 12*).

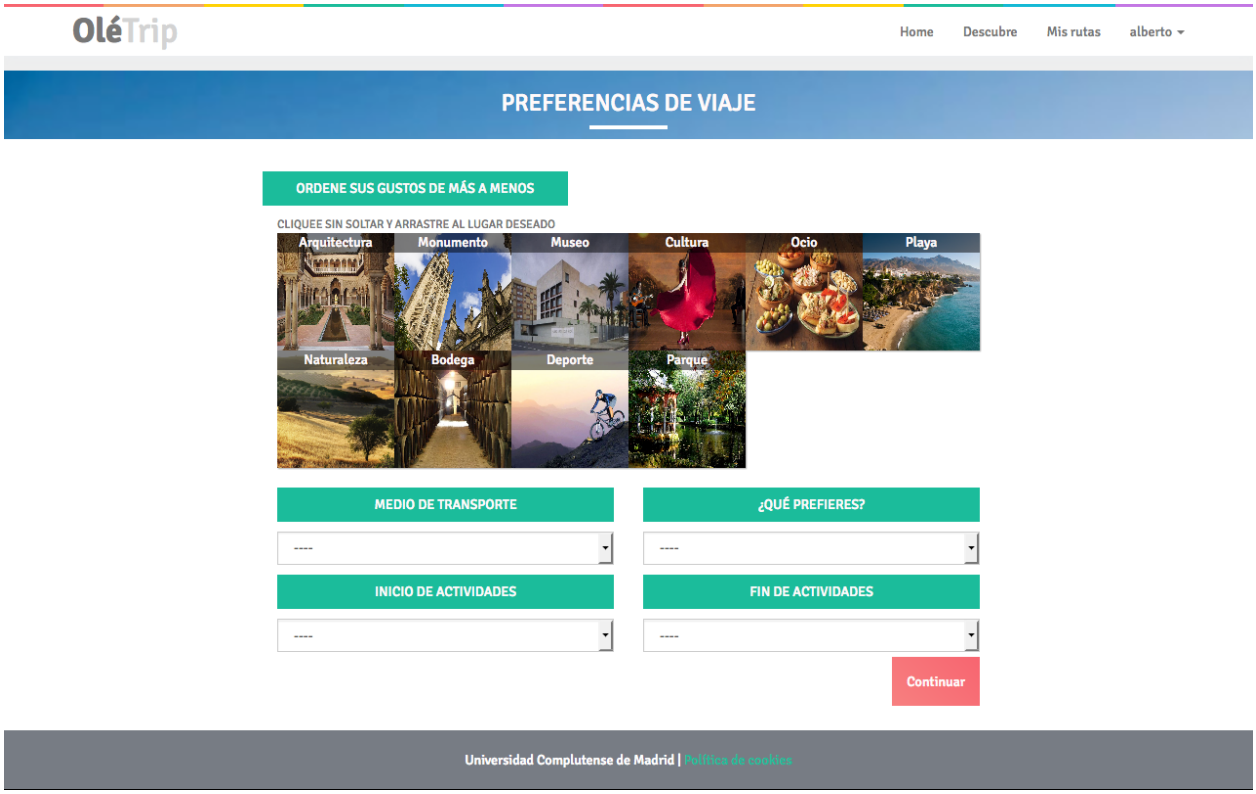

Figura 12 Preferencias Viaje

Desde aquí el usuario podrá elegir el medio de transporte a usar, el modo relax o tiempo completo, y el rango de horas en el que quiere empezar y finalizar las actividades de un día.

A continuación (*figura 13*), el usuario podrá seleccionar las actividades que quiere visitar o pedir a la aplicación que le recomiende un plan a su medida.

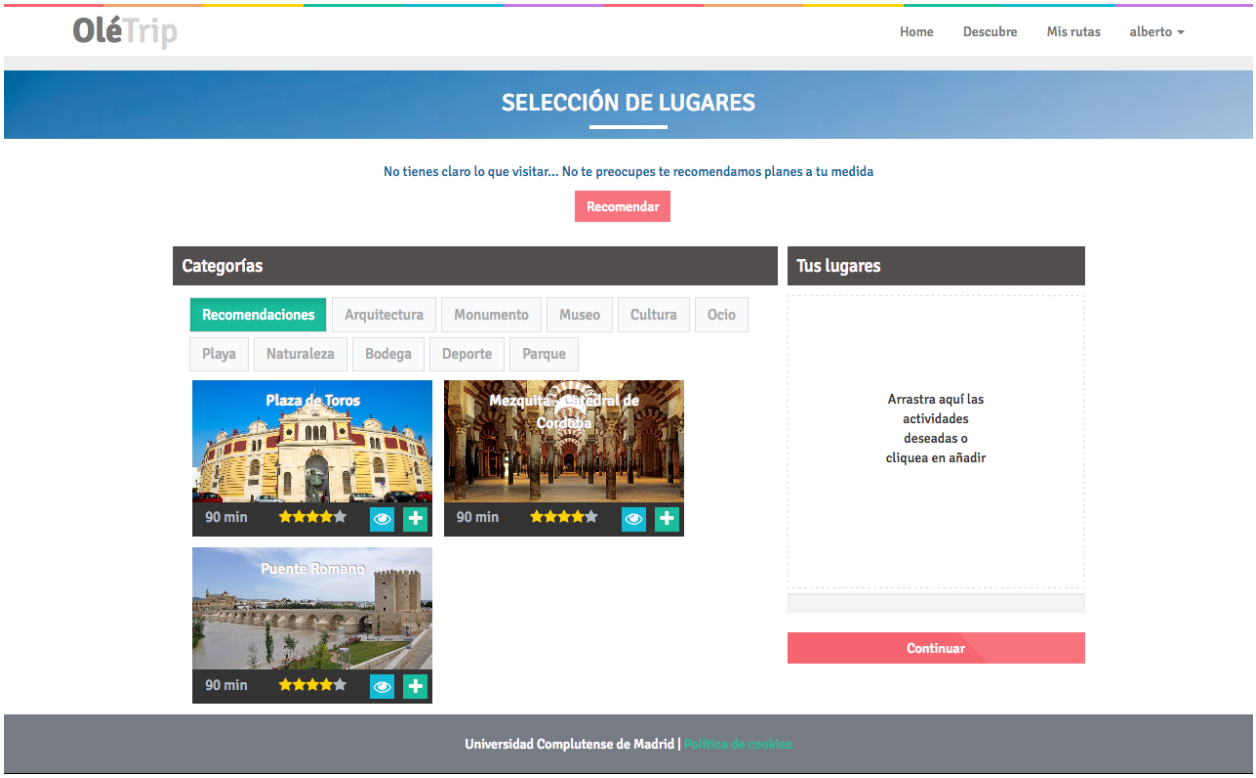

Figura 13 Selección de lugares / Recomendador

Si el usuario elige la opción de recomendar un plan se le mostrará la interfaz de la *figura 14*:

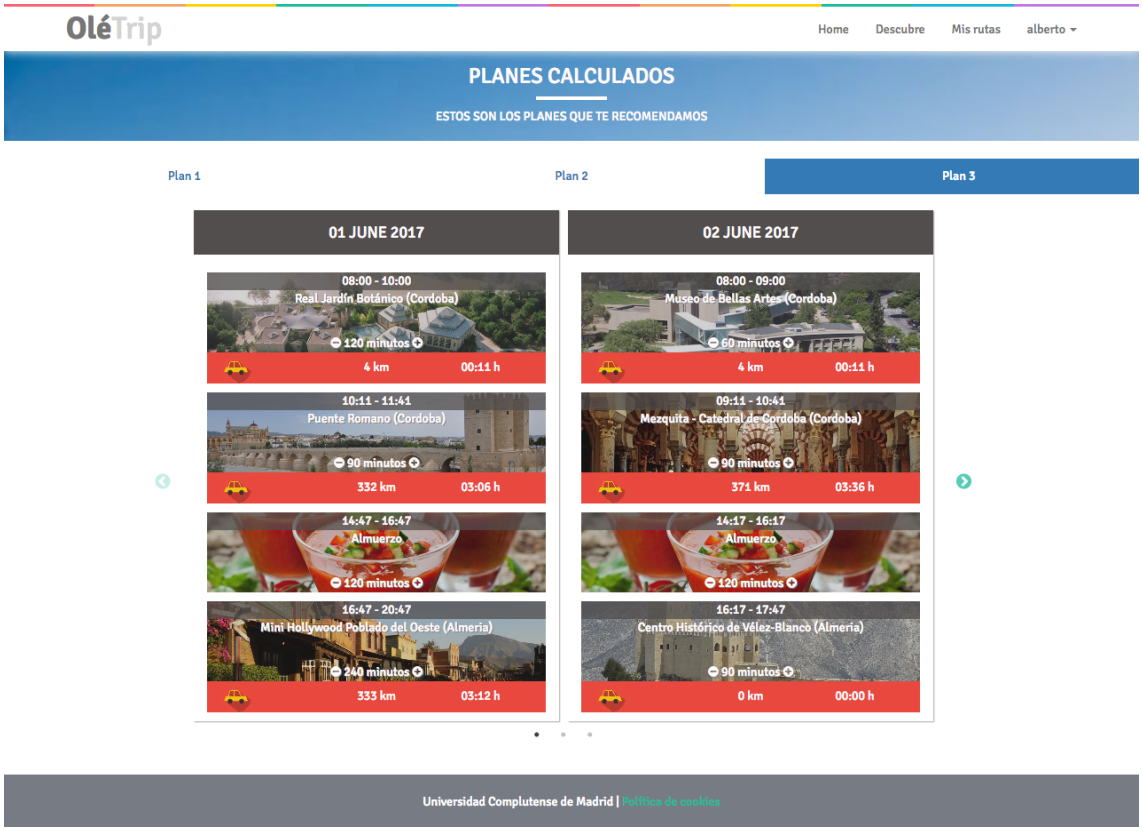

Figura 14 Recomendador

Donde podrá elegir uno de los tres planes que devuelve la aplicación.

Para el caso en el que el usuario elija las actividades que quiere visitar hemos realizado dos simulaciones, una con el algoritmo voraz que mostramos en la *figura 15* y otra con el algoritmo BFS en la *figura 16*.

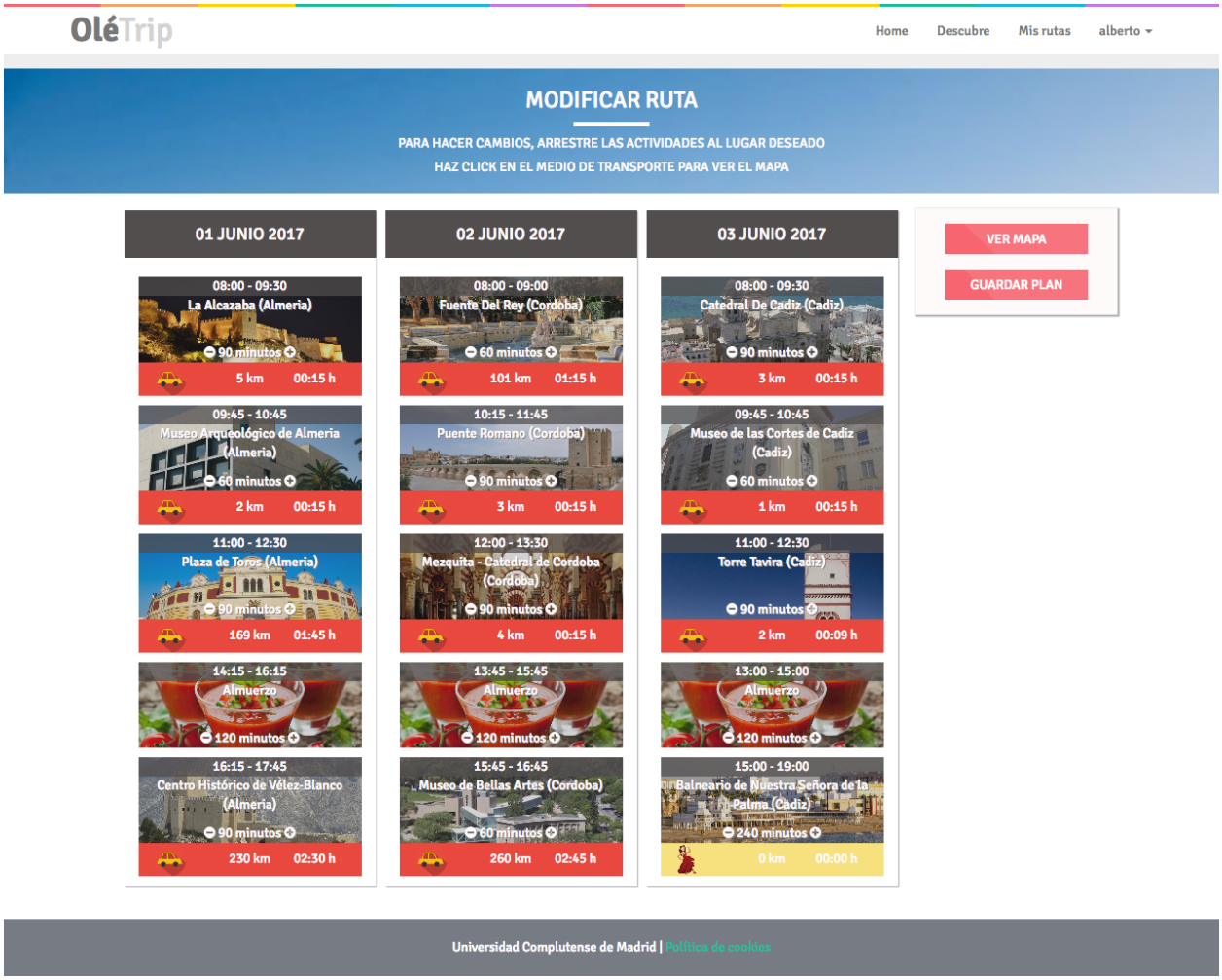

Figura 15 Plan voraz

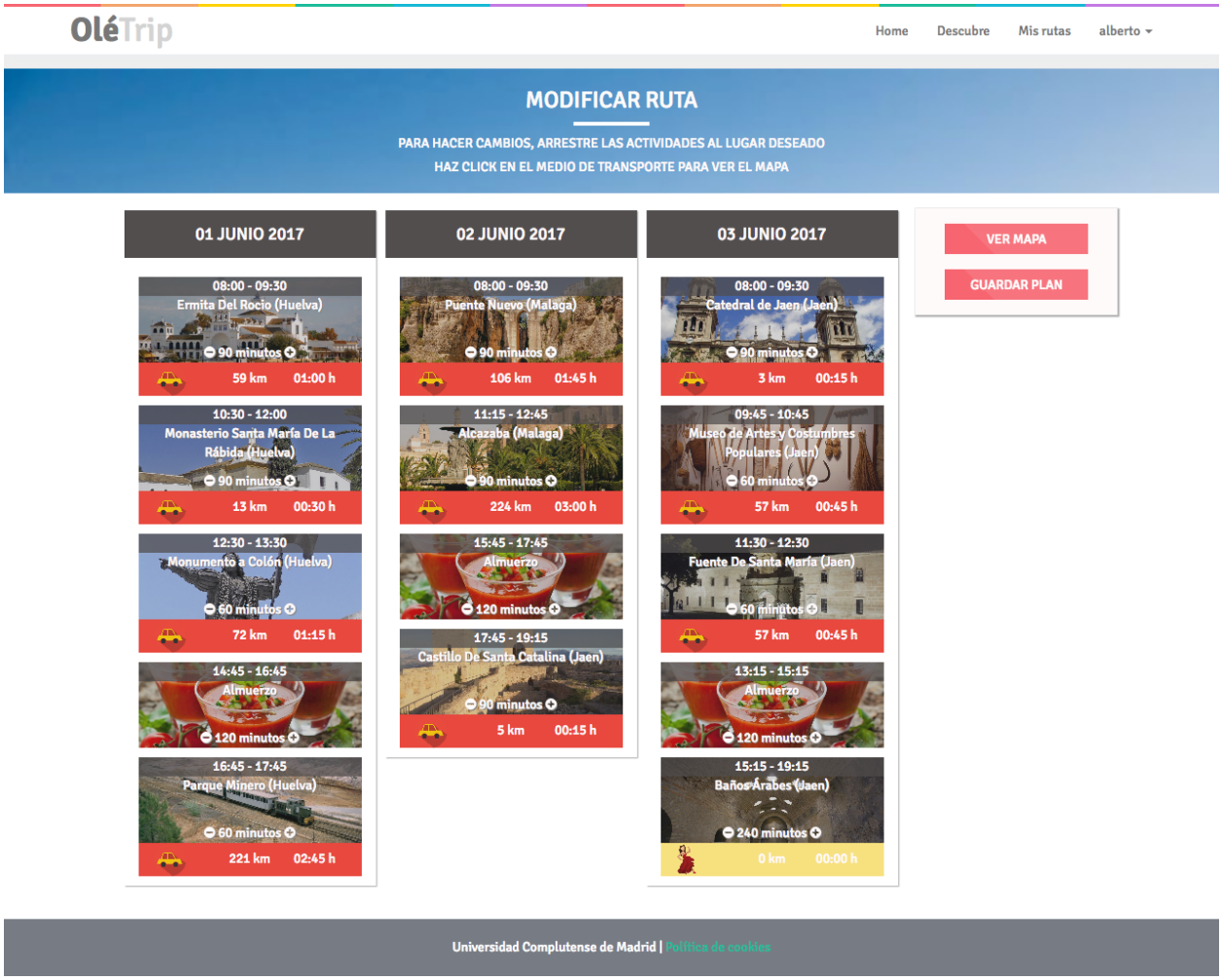

Figura 16 Plan BFS

Una vez generado el plan el usuario puede guardarlo independientemente del algoritmo escogido, así como su visualización en el mapa.

6. ARQUITECTURA DEL SISTEMA

En este capítulo se habla de la estructura plan, el diseño de la base de datos y las tablas añadidas y las nuevas funcionalidades de la interfaz web.

6.1. PLANIFICACIÓN DE UN VIAJE

En este apartado vamos a explicar la estructura de un plan.

El plan se compone en varios días, dentro de estos días tenemos una serie de actividades, que son los lugares a visitar y el almuerzo.

Los días son elegidos por el usuario cuando crea el plan, así como las actividades a realizar. Tienen una hora de inicio y una hora de fin, ambas introducidas por el usuario, que determinan el tiempo del que dispondremos cada día para la realización de las actividades.

6.2. BASE DE DATOS

La base de datos utilizada es la misma que se utilizaba en el proyecto del año pasado. Gran parte de la estructura ha permanecido intacta, de manera que solo hablaremos de los cambios que hemos introducido, no obstante, en la *figura 17* tenemos la estructura general.

La estructura de cada tabla en concreto se encuentra en el apéndice.

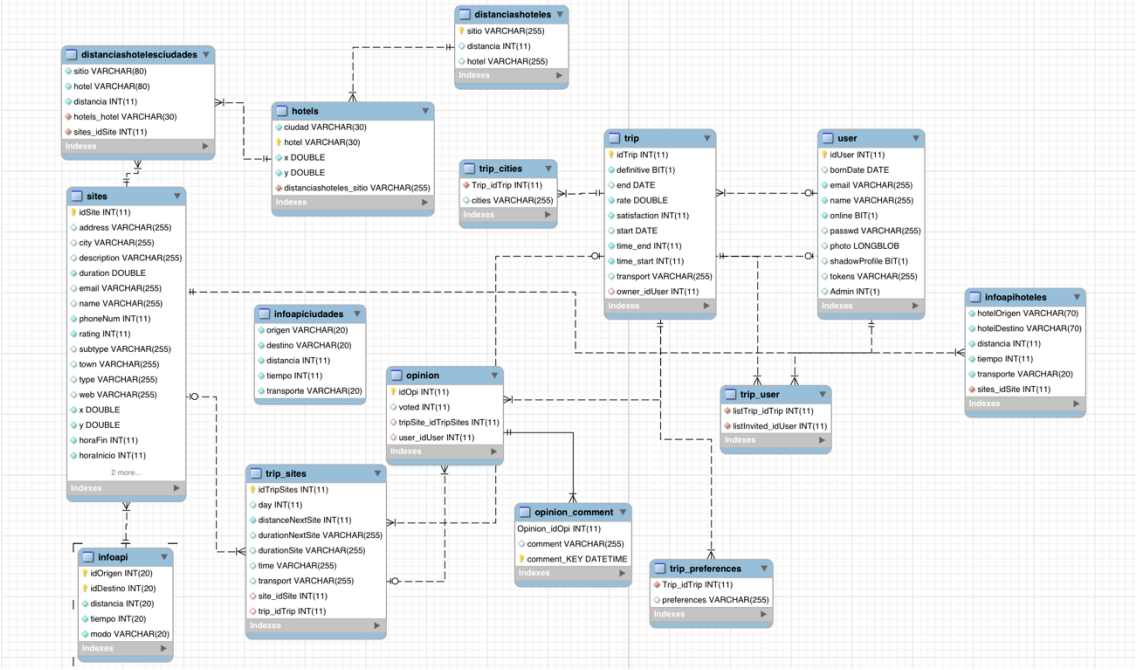

Figura 17 Estructura general de la Base de Datos

Modificaciones

- USER

Esta tabla ya existía. Su función es guardar información del perfil del usuario, pero le hemos introducido la columna '*admin*', cuya función es la de poder diferenciar los usuarios según sean *admin* o no.

Los usuarios de tipo *admin* tendrán acceso a vistas que los usuarios normales no perciben.

Solución de problemas y nuevas tablas

Para las siguientes tablas hay que mencionar que antes, todas las distancias y tiempos que se calculaban mediante consultas a la *API de Google*, estas consultas están limitadas a 2.500 por día, lo cual no suponía un problema, pero con la implementación de los nuevos algoritmos que se explican más adelante, este límite se alcanza fácilmente, de manera que fue necesario crear en principio dos nuevas tablas.

- INFOAPI

Esta tabla guarda las distancias y el tiempo entre un sitio y otro teniendo en cuenta el modo de transporte que utilice.

De cada sitio guardamos su idOrigen, idDestino, distancia, tiempo y transporte, además de tener en cuenta cual es el sitio de origen y cuál el sitio de destino, dado que la distancia y el tiempo del punto A al B y del punto B al A no tiene que ser la misma ya que pueden ser recorridos distintos.

- INFOAPICIUDADES

Esta tabla es parecida a la anterior, ya que también almacena las distancias y el tiempo, pero en este caso entre el centro de una ciudad y otra dependiendo del modo de transporte.

En este caso guardamos el nombre de cada ciudad, ya que no existe una tabla específica con ciudad que tenga una id para relacionarla (como ocurría con los sitios), y también guardaremos la distancia, tiempo y transporte.

Después se crearon tres nuevas tablas relacionadas con un hotel para cada ciudad y la información de distancias y tiempos, que son necesarias para el validador, el cual se explicará más adelante.

- HOTELS

Esta tabla representa la posición de un hotel dentro de una ciudad, del cual tenemos el nombre de la ciudad a la que pertenece, el nombre del hotel, y sus coordenadas ‘x’ que representa la longitud e ‘y’ que representa la latitud.

- INFOAPIHOTELES

Esta tabla guarda la distancia y el tiempo entre un hotel origen y un hotel destino dependiendo si es en coche o en transporte público, de la misma forma que lo hacen las tablas ‘infoapi’ e ‘infoapiciudades’.

- DISTANCIASHOTELESCIUDADES

Esta tabla guarda la distancia entre un sito y el hotel de la misma ciudad, también de la misma forma que la tabla anterior.

6.3. INTERFAZ

A continuación, mostramos la interfaz nueva que se ha añadido al proyecto.

6.3.1. RECOMENDADOR

Esta funcionalidad se desarrolló gracias al uso del algoritmo genético como se verá más adelante.

Decidimos crear esta interfaz para aquellos usuarios que no tengan preferencia en los lugares a visitar

Una vez que el usuario ha elegido las ciudades a visitar y el tiempo y modo disponible es el momento de elegir los sitios que guarda la aplicación referente a dichas ciudades damos la opción al usuario de elegir sus lugares a visitar o usar el recomendador de planes creado.

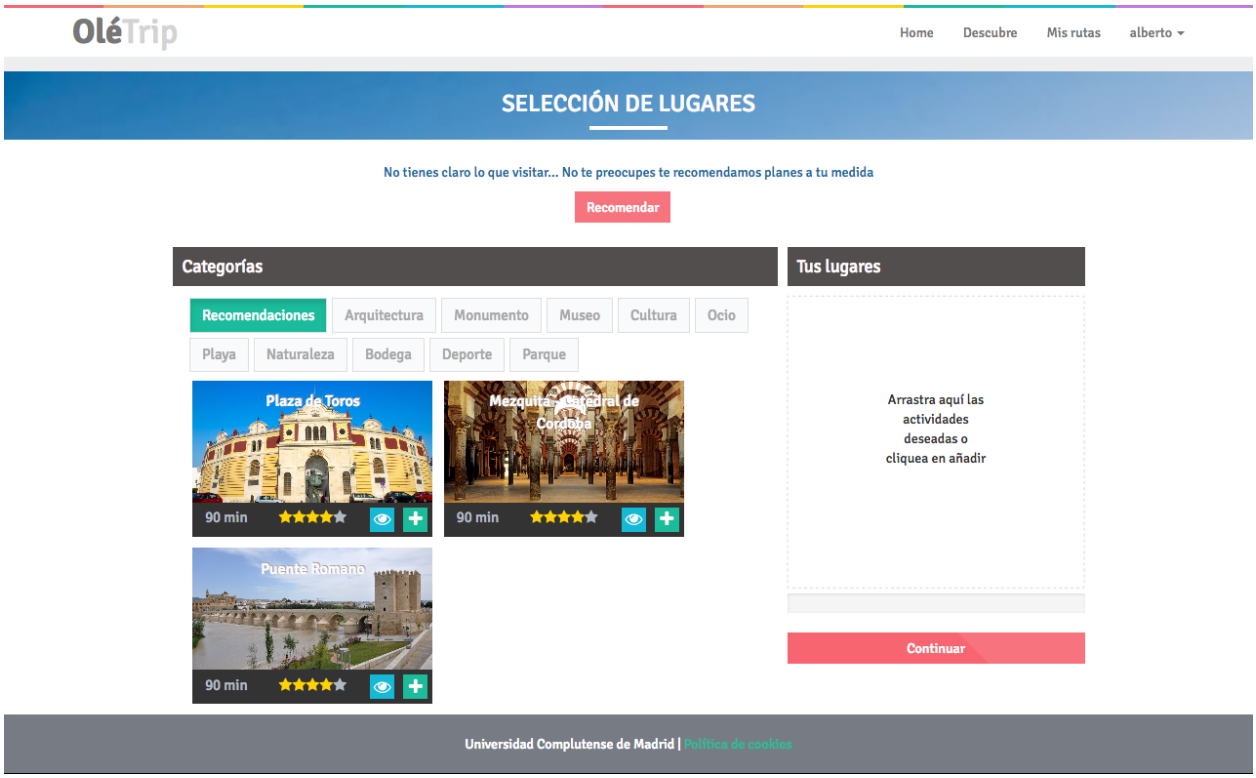

Figura 18 Selección de lugares / Recomendador

La aplicación muestra un botón al usuario que le permite obtener planes hechos a su medida tal y como se muestra en la *figura 18*.

Una vez seleccionada esta opción la aplicación lanza el algoritmo genético que obtiene la última población creada y devuelve los 3 mejores planes obtenidos de dicha generación y los pinta al usuario como se muestra en la *figura 19*.

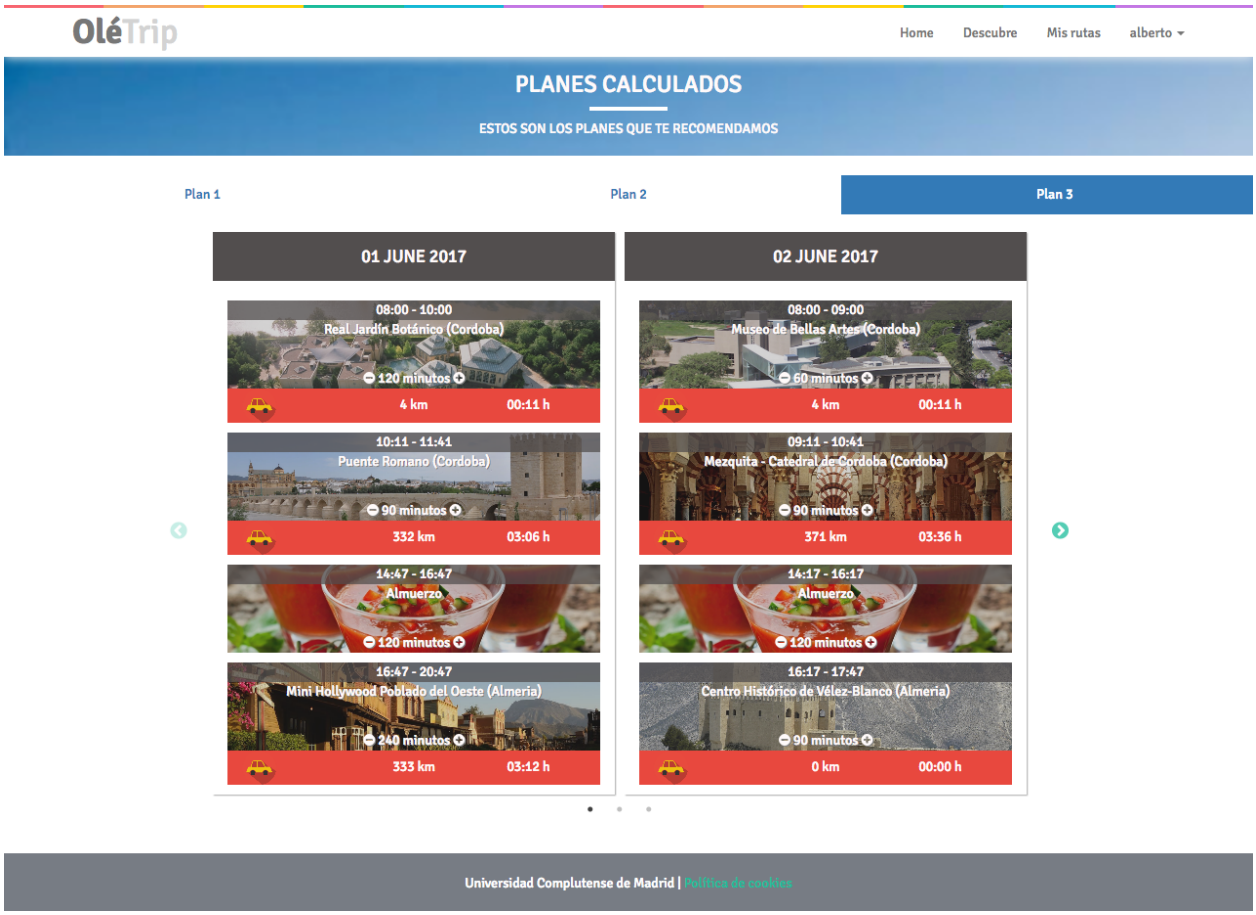

Figura 19 Recomendador

Para obtener las actividades hemos estimado que un día tiene una capacidad de tres actividades por lo que obtenemos una proporción de actividades en función de los días disponibles del usuario. A partir de ahí se obtienen las actividades aleatoriamente filtrando las ciudades elegidas por el usuario.

7. MECANISMO DE CREACIÓN DE RUTA

En este capítulo explicaremos detalladamente el método utilizado para evaluar la calidad de estos planes, así como cada uno de los algoritmos utilizados para generar un plan y por último la función de nuestro simulador.

7.1. EVALUACIÓN DE LA CALIDAD DE LOS PLANES

El validador es el encargado de evaluar la calidad de los planes calificándolos con un valor numérico y siendo este plan mejor cuanto mayor sea este valor.

Éste recibe el *JSONObject* generado con la información del plan a realizar, la cual contiene:

- El número de días que conforman el plan.
- Las actividades, pudiendo ser varias en un mismo día, con la información de la actividad, las distancias y tiempos hacia la siguiente actividad y el modo (“relax” o “quiero verlo todo”).

Hay que tener en cuenta que estas distancias no son en línea recta de una actividad a otra, ya que tienen que tener en cuenta las calles por las que puede acceder ya sea por coche o transporte público.

La ecuación para calcular el valor numérico es:

Valoración = ∑Actividades - ∑Pen. distancias - ∑Pen. pref. usuario

Cuantas más actividades, mayor será la valoración.

La penalización por distancias engloba tres tipos, las distancias entre actividades, las distancias entre hoteles, y las distancias entre actividad y hotel.

Según del tipo que sean se multiplican por una constante.

Para el caso de la distancia entre actividades, la constante multiplicativa hace que a partir de 300 km se penaliza el equivalente a una actividad (al ser números decimales si son por ejemplo 450 km restaría 1.5). Estos 300 km son un número elevado dado que las distancias de la *API de Google*, que son las utilizadas y se tiene en cuenta todo el camino a recorrer y no la distancia en línea recta, además hay que tener en cuenta que hay otro tipo de penalizaciones.

Igualmente, para la distancia entre hoteles y distancia entre actividad y hotel, la constante multiplicativa penaliza el equivalente a una actividad cuando el sumatoria de la distancia supere los 1000 km, igual que con la distancia entre actividades, este número es elevado, aunque no demasiado para que tengan preferencia los planes cuando se desarrollan de una ciudad a la siguiente más próxima y no a otra más lejana.

Por ejemplo, si tenemos tres actividades en Sevilla, tres en Huelva y tres en Córdoba, dado que Sevilla está entre Huelva y Córdoba, la distancia será menor cuando se haga el recorrido Huelva -> Sevilla -> Córdoba, o cuando se haga el recorrido Córdoba -> Sevilla ->Huelva.

Para las actividades dentro de una misma ciudad será parecido, solo que también hay que tener en cuenta la distancia de la actividad al hotel, de manera que también tomará importancia que la primera y última actividad del día estén lo más cerca posible del hotel.

La penalización por preferencia del usuario varía según el usuario haya elegido el modo de relax, o decidió que quería verlo todo.

Los planes por lo general se componen de cuatro actividades por día, pero según las distancias entre esas actividades o de la duración que tengan puede que entren menos planes en un día, la penalización será de 0.05 en modo “Relax” y de 0.1 en “Quiero verlo todo” por cada actividad de menos.

Obtención de datos.

Para el cálculo de distancias es preciso mencionar el esquema que se sigue para calcular las distancias según la actividad sea la primera o última del día o ciudad.

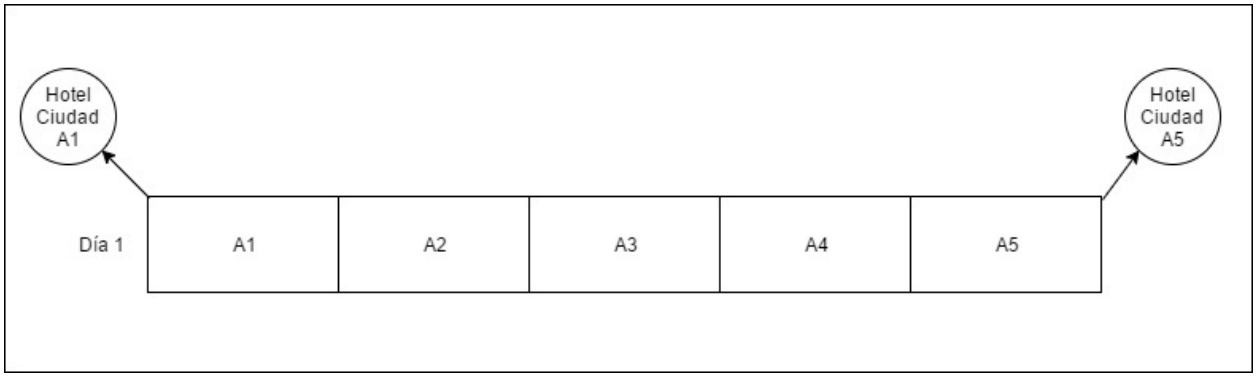

Figura 20 ejemplo plan 1 evaluación

En la *figura 20* podemos ver un caso de un día, en el que hay cinco actividades y empezamos el día en un hotel de la ciudad de la actividad A1, distancia la cual se contabilizaría en la distancia entre actividad y hotel.

Después tenemos las distancias de A1 a A2, de A2 a A3, de A3 a A4 y de A4 a A5, todas ellas forman parte de la distancia entre actividades.

Finalmente volveríamos a tener la distancia entre A5 y un hotel de la ciudad a la que pertenece la actividad A5.

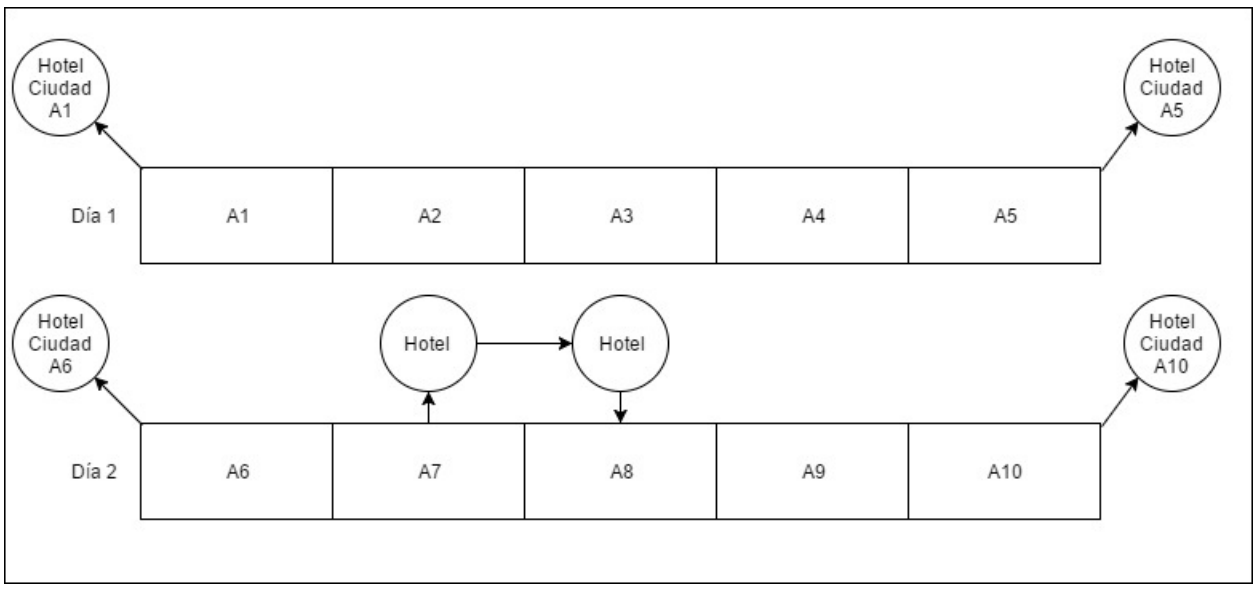

Figura 21 ejemplo plan 2 evaluación

En el día 2 de la *figura 21*, tenemos igual que en el día 1 distancia de hotel a actividad y de actividad a actividad, pero la A7 resulta ser la última actividad que se va a realizar de su ciudad, en ese caso, se iría al hotel para prepararse hacia la siguiente actividad, donde también iríamos al hotel de esa ciudad y posteriormente a la actividad, teniendo en cuenta en este caso distancia de A7 a hotel y de hotel a A8 para la distancia entre actividades y hoteles, y la distancia entre el hotel de la ciudad de la A7 y el hotel de la ciudad de la A8, en la distancia entre hoteles.

Siguiendo este esquema, la fórmula para calcular cada una de las distintas sumas de distancias es la siguiente.

La distancia entre actividades es simple. Sumaremos siempre la distancia entre una actividad y otra salvo en los casos en que sea la última actividad del día, porque se calculará la distancia al hotel y no a la próxima actividad, y tampoco se sumará si esta actividad es la última actividad que se realizará en esa ciudad, ya que volverán a tenerse en cuenta las distancias con hoteles y no entre actividades.

La distancia entre actividad y hotel, se hará en esos casos en los que no se sumaban la distancia entre actividades, es decir, cuando la actividad es la primera o última del día, o cuando es la primera o última actividad en una ciudad.

La distancia entre hoteles, siempre que cambiemos de hotel, debido a que cambiemos de ciudad, sumaremos esta distancia y esto puede suceder tanto a mitad del día como en el caso ejemplo entre la actividad A7 y A8, como a final del día, como sucedería si la actividad A5 y A6 fueran de distintas ciudades, también habría que tener en cuenta esa distancia.

La penalización según el modo varía en función de si el usuario eligió ir de relax o quería verlo todo, según se hayan realizado cuatro o menos actividades, de manera que en el modo de relax penalizará menos que en el modo quiero verlo todo.

Todos los valores de distancias están calculados en metros, y han sido obtenidos previamente mediante consultas a la base de datos, la cual se rellenó con consultas a la API de Google.

7.2 Aleatorio

7.2.1. Motivación

En primera instancia este algoritmo se realizó para familiarizarnos con el código, de forma que pudiésemos entender mejor su funcionamiento y estructura y fuésemos capaces de modificarlo para conseguir los resultados que requiriésemos en un futuro.

Posteriormente para la realización del algoritmo genético, partimos del aleatorio para generar poblaciones y combinar genes, por lo cual rediseñamos el algoritmo para poder dar soporte al genético.

7.2.2 Algoritmo

El método de creación de ruta aleatorio se encarga de generar un plan aleatorio a partir de las distintas actividades que haya introducido el usuario.

Para ello elegimos una actividad aleatoriamente, las actividades que vamos obteniendo las añadimos a una lista junto con la distancia y tiempo hacia la próxima actividad y el modo de transporte (elegido por el usuario).

Esta distancia y tiempo se obtienen mediante consulta a la base de datos, con la id del sitio de origen y del sitio destino.

Finalmente obtenemos una lista de actividades con las distancias, tiempos y modo de transporte.

7.2.3 Ventajas y desventajas

La principal ventaja es su simplicidad, genera un plan sin apenas coste, útil en el caso de que queramos un plan cualquiera, que combinado mediante alguna otra lógica pueda dar planes con más sentido (como en el caso del genético haciendo uso del validador).

Por contra, el plan generado, no será ni mucho menos óptimo para el usuario ya que los lugares elegidos serán completamente al azar, lo que significa que, por ejemplo, un usuario puede tener una actividad en Sevilla y la siguiente sea en Granada, pudiendo volver a visitar un lugar de interés de Sevilla suponiendo un coste muy elevado al usuario.

7.3 Voraz

7.3.1 Motivación

Este algoritmo fue ya implementado en el proyecto anterior y ha sido muy útil para el desarrollo de los otros dos algoritmos que se explican más adelante, BFS y genético.

7.3.2 Algoritmo

Se basa en la cercanía entre los distintos lugares que conforman el plan elegido por el usuario. Se tomará en cuenta factores como el tiempo de la actividad y la distancia que hay entre cada una de ellas, así como la hora de la comida.

El funcionamiento del algoritmo es el siguiente:

Las actividades del plan son agrupadas por su ciudad correspondiente. El algoritmo empieza por la ciudad que más actividades posea calculando las distancias y tiempos entre cada uno de ellos. Cuando se alcanza el punto en el que haya que cambiar de ciudad, se elige el lugar más cercano al último de la ciudad anterior.

En la siguiente figura se ilustra el caso en el que, dentro de una misma ciudad, qué lugar escogería dado un lugar de partida como vemos en la *figura 22*:

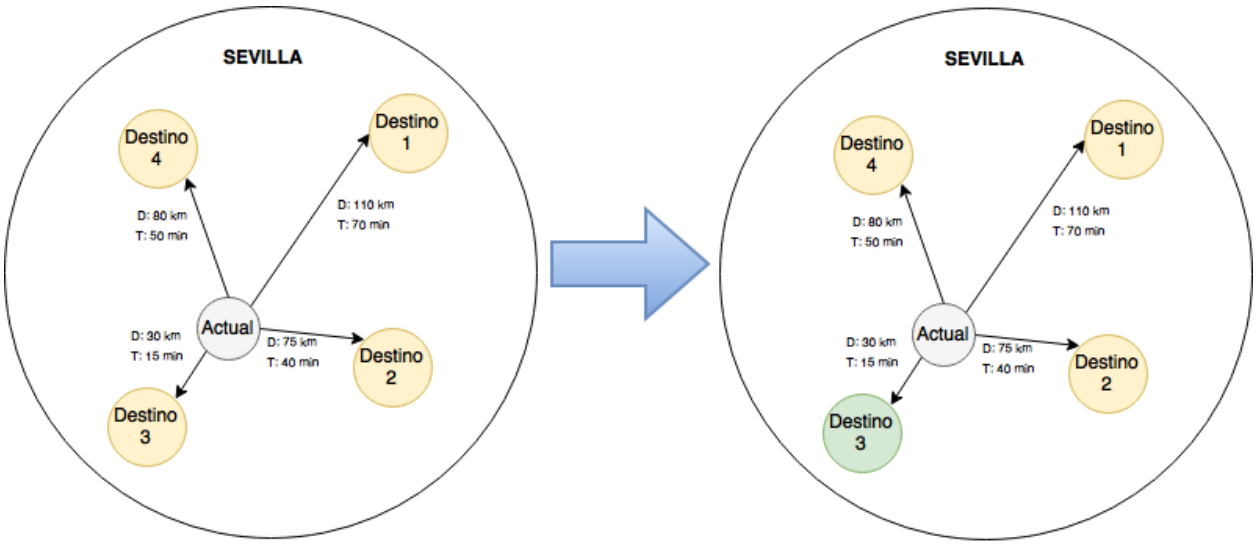

Figura 22 Ejemplo funcionamiento voraz 1

Así pues, el plan, en este punto, estaría compuesto por: {*Actual, Destino 3*}, siendo *Destino 3* en la siguiente iteración, el lugar de partida.

En la siguiente figura, suponemos que en la ciudad de Almería hemos visitado todos los lugares que queríamos visitar y la siguiente ciudad que tenemos que visitar, es Sevilla.

El algoritmo recorrerá todos los lugares de Sevilla que, y escogerá la más cercana al lugar de origen (*figura 23*).

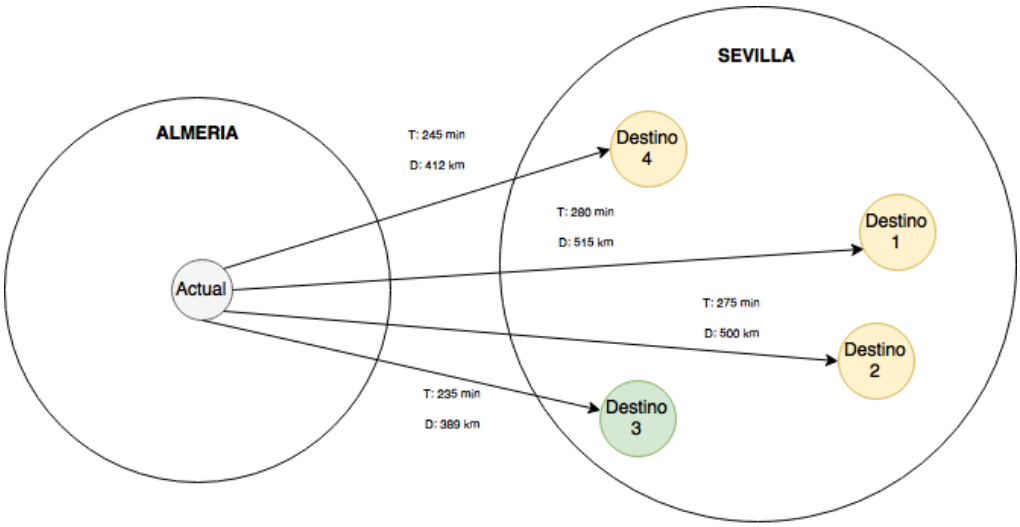

Figura 23 Ejemplo funcionamiento voraz 2

Así pues, el plan en este punto quedaría: {*Actual, Destino 3*}.

7.3.3 Ventajas y desventajas

La principal ventaja de este algoritmo consiste en que el plan que se le muestra al usuario es muy óptimo tomando siempre las distancias más cortas entre lugares.

Una desventaja es que el algoritmo no permite incluir información adicional como las preferencias del usuario, simplemente se basa en distancias.

7.4 BFS

En este algoritmo BFS (*Best-first Search*) nos centramos en ir recorriendo una serie de nodos tomando como punto de partida cada uno de los distintos lugares que el usuario haya elegido para visitar, completando a partir de esa primera actividad mediante una búsqueda heurística utilizando el algoritmo voraz del apartado anterior para desarrollar un plan completo a partir de cada uno de los sitios elegidos como punto de partida.[15]

Cada plan completo recibe una calificación según el validador teniendo en cuenta las distancias entre actividades y hoteles de la ciudad a la que pertenezcan estas actividades, estos planes se van almacenando en una cola de prioridad ordenados según la nota que hayan recibido del validador, con la lista de actividades que se componen por las contenidas en el nodo, y la valoración numérica que permite ordenar en la cola los nodos de mejor a peor.

Los nodos se van desarrollando cogiendo de la cola el nodo que mejor valoración haya tenido hasta ahora y se añaden como siguiente actividad las otras restantes, desplegando otra lista de nodos que volverán a desarrollarse de la misma forma que los anteriores.

El mejor plan hasta el momento (con todas sus actividades e información) se guarda para poder disponer de él directamente sin necesidad de tener que volver a hacer una llamada al voraz con las actividades del nodo que haya resultado mejor.

Este proceso tiene un límite de 10 segundos para encontrar la mejor opción posible de manera que proporcione una de las mejores planificaciones en un tiempo óptimo.

Ejemplo de una lista de actividades A1, A2, A3, A4, A5 en la *figura 24:*

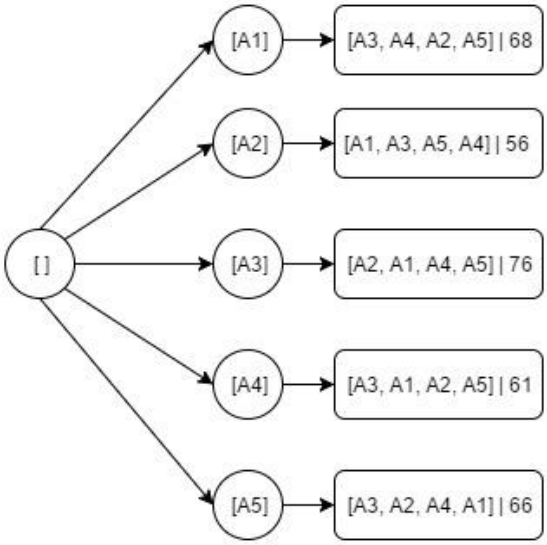

Figura 24 Despliegue nodos BFS

Se despliegan los primeros cinco nodos.

Como podemos ver en el dibujo, primero se realiza el método voraz tomando como punto de partida cada una de las actividades que el usuario quiere realizar, creando un plan del cual obtenemos una valoración de acuerdo al validador, este plan es completo desde el inicio, el cual guardaremos. Esto se guardaría en la cola de prioridad de la siguiente forma:

Lista Actividades	Valoracion
[A3]	76
[A1]	68
[A5]	66
[A4]	61
[A2]	56

Donde tenemos por un lado el nodo con la lista de actividades por las que empezar (en el principio solo hay una actividad) con su respectiva valoración.

Tabla 2 Cola prioridad 1

Esta cola de prioridad funciona de manera muy simple, guarda por un lado una lista de actividades que serían las pertenecientes al nodo (las actividades iniciales del plan a diseñar), y por el otro un valor numérico correspondiente a la valoración que ha recibido el plan generado a partir de ese nodo con el validador.

El orden que se guarda en la cola depende del número de la valoración de manera que si es el mayor ocupará el primer puesto de la cola.

El nodo más prometedor es el que comienza por la actividad A3, por lo tanto, seguiremos desarrollando más nodos a partir de ese.

A diferencia de la primera iteración en la que la cola de prioridad aún estaba vacía, en las siguientes se obtiene la actividad o lista de actividades que es más prometedora hasta el momento, y se descartan de la lista inicial de lugares que quiere visitar el usuario para que no se dupliquen, siendo la lista obtenida por la cola de prioridad las primeras actividades del plan que se vaya a generar, y las actividades restantes de nuevo volverán a ser ordenadas mediante el método voraz.

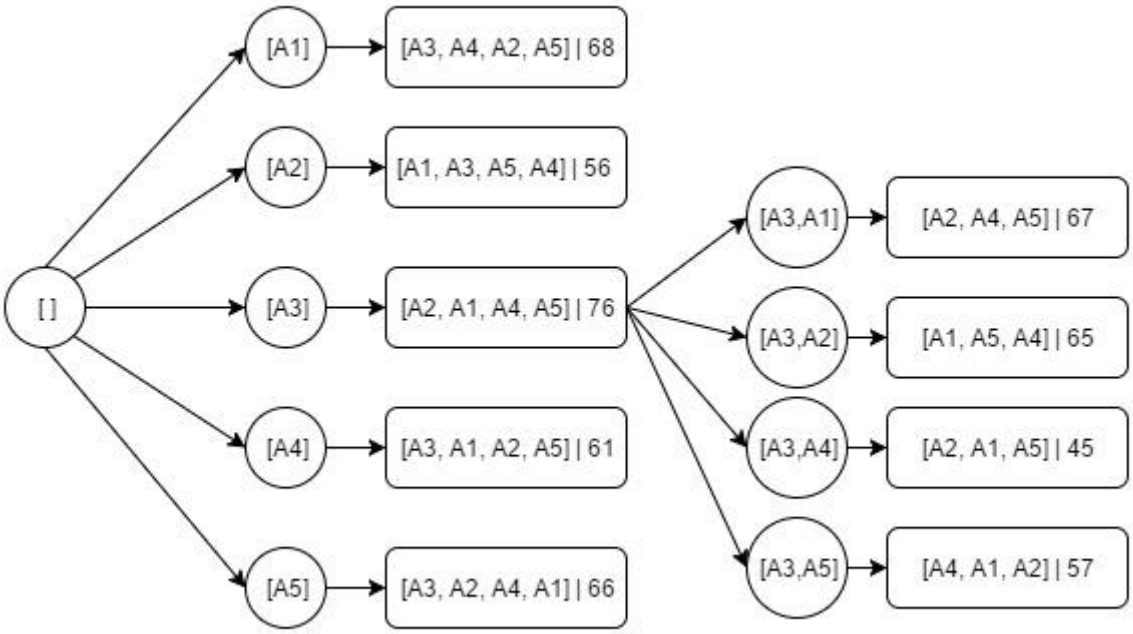

Figura 185 Despliegue nodos BFS 2

En este caso como muestra la *figura 25* podemos observar que, pese a que el nodo más prometedor era el de A3, al seguir desplegando los siguientes nodos tomando como inicio A3, el que mejor resultado ha obtenido ha sido [A3. A1] con un 67, no superando el nodo [A1] quedando este el primero en la cola de prioridad con una puntuación de 68, con lo que el próximo nodo a desarrollar sería este.

Lista Actividades	Valoracion
[A1]	68
[A3, A1]	67
[A5]	66
[A3, A2]	65
[A4]	61
[A3, A5]	57
[A2]	56
[A3, A4]	45

Como podemos ver el nodo [A1] es ahora el más prometedor y será el próximo en desarrollarse, en caso de que ninguno superase el de [A3, A1] este sería el que quedase primero en la cola y el próximo que se desarrollaría, en caso de que uno o varios de los nodos desarrollados a partir de [A1] lo superase, estos quedarían por encima y serían los siguientes.

Tabla 3 Cola prioridad 2

Este proceso se sigue realizando hasta un tiempo límite de 10 segundos o en su defecto hasta haberse realizado todas las opciones posibles, obteniendo como resultado el mejor plan generado hasta el momento que había sido previamente guardado, de manera que se encuentre la mejor opción posible obteniendo una de las mejores planificaciones con las actividades que el usuario quiere realizar en un tiempo óptimo.

7.4.1 Ventajas y desventajas

La mayor ventaja con respecto a los otros planes como el genético es que genera planes óptimos desde el principio, pero a diferencia del voraz estos pueden ir mejorando con el tiempo de forma que en un reducido lapso de tiempo podemos obtener un buen plan.

Por contra, cuanto mayor es el espacio de búsqueda mayor es el consumo de memoria, problema que se podría resolver limitando el tamaño de la cola de

prioridad, almacenando exclusivamente los mejores nodos, pero eso haría que el algoritmo no fuera completo.

Por otro lado, si queremos obtener el mejor plan posible, el cálculo de todos los nodos posible costaría muchísimo tiempo, así que en el hipotético caso de querer el mejor plan posible (según nuestro validador, basado en distancias), tardará demasiado en poder comprobar todos los casos posibles para garantizar el mejor plan posible.

7.5 GENÉTICO

7.5.1 Motivación

Debido a no encontrar un planificador que pudiese adaptarse a las necesidades que requería nuestro proyecto tuvimos que optar por desarrollar un algoritmo que nos permitiese analizar la evolución de planes entre distintas generaciones para comprobar que si invertimos algo de tiempo en el proceso de creación de un plan pudiésemos generar una serie de planes interesantes para el usuario.

7.5.2 Algoritmo

Un algoritmo genético (AG) es un método para solucionar problemas de optimización con o sin restricciones basándose en un proceso de selección natural que imita la evolución biológica.[16]

Para llevar a cabo este proceso se toma como muestra una población inicial de *n* individuos a los cuales se les analiza mediante una función de aptitud que delimita cuan buenos son para adaptarse al medio y poder valorar su probabilidad de sobrevivir a las siguientes generaciones.

Una vez valorados los individuos, se realizan funciones de mutación y se cruzan entre ellos para posteriormente descartar una serie de individuos que no pasarán a la siguiente generación. Este proceso de descarte se puede hacer de múltiples maneras, como por ejemplo volviendo a cruzarlos todos y descartando los peores, quedarte con individuos de forma aleatoria o haciendo una segmentación de los mismos. Dependiendo qué mecanismos se utilizan, alcanzaremos antes un máximo local.

La toma de decisión de la mutación se realiza mediante una probabilidad que tenga un gen para ser o no mutado.

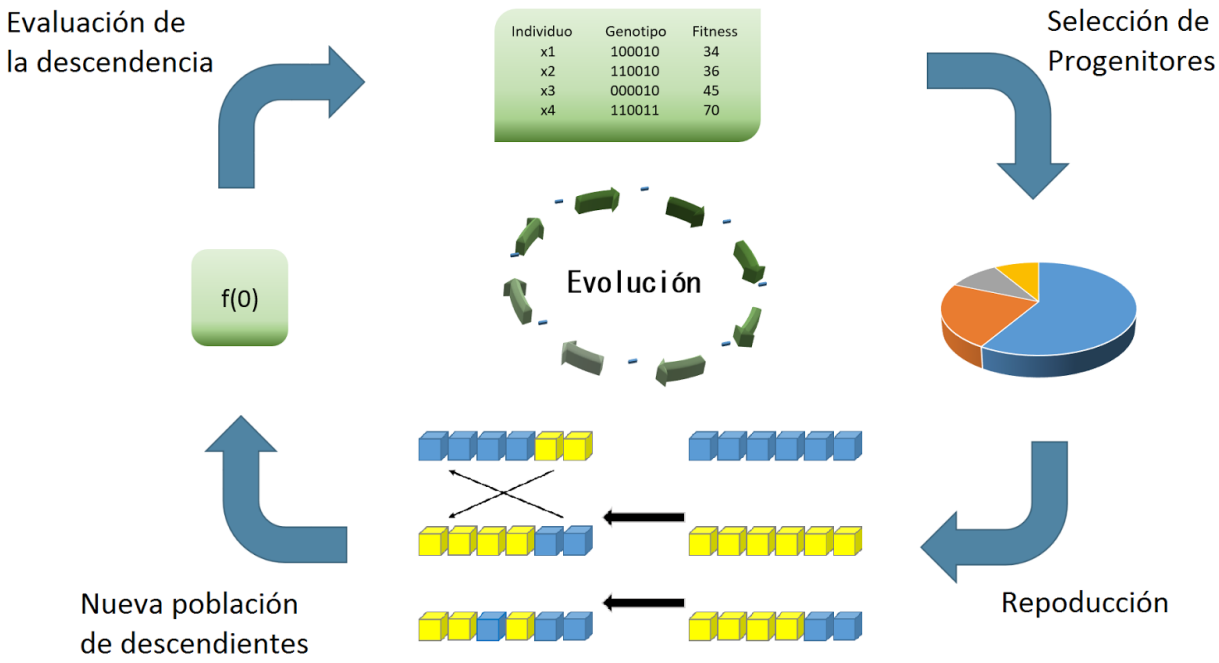

Figura 26 Estructura algoritmo Genético

En esta *figura 26* se muestra un breve esquema del funcionamiento de este algoritmo.

7.5.3 Implementación

Para el desarrollo de este algoritmo hemos requerido del uso de una lista con los sitios, una lista de ciudades, el idTrip generado y los días que durará el plan.

Definimos que tanto los planes como el número de poblaciones tendrán un tamaño mínimo de cincuenta. Estos parámetros se pasarán a la función.

También definimos que la probabilidad que tenga una actividad de un plan para mutar sea del 1%.

Es importante entender que en este contexto los planes que generamos serán definidos como cromosomas y las actividades como genes.

Una vez definidos todos los parámetros que tendrá nuestro algoritmo comenzamos a definir su funcionamiento.

Lo primero que hacemos es generar un plan con el algoritmo voraz, que en principio siempre estará en la élite de la población y pasará a la siguiente

generación y *n*-1 planes aleatorios siendo n el número de la población. De esta forma obtenemos la primera generación con la que trabajamos.

Una vez obtenida toda la población calculamos la eficiencia de cada plan y la eficiencia acumulada de la población.

Ahora que ya tenemos las eficiencias de los planes, lo que hacemos es ordenar los planes mediante un algoritmo de ruleta.[17] Este proceso genera un número aleatorio que dividido por la eficiencia acumulada nos permite obtener una probabilidad de la eficiencia acumulada con la que podremos ordenar los planes. Gracias a este mecanismo los planes ya quedan ordenados para poder pasarlos por parejas al método que los cruza.

El proceso de cruzar planes es algo complejo. Como hemos dicho antes recibe dos cromosomas de los cuales obtenemos sus actividades. Una vez tenemos las actividades procedemos a cruzarlas. Para este proceso hemos definido un punto de cruce obtenido de forma aleatoria. [18]

Cuando terminan de cruzarse pasamos a tener el doble de cromosomas, por un lado, tenemos los dos primeros cromosomas originales y los dos cromosomas cruzados.

Ahora que tenemos los cromosomas cruzados nos encontramos con el problema de tener genes duplicados en estos cromosomas.

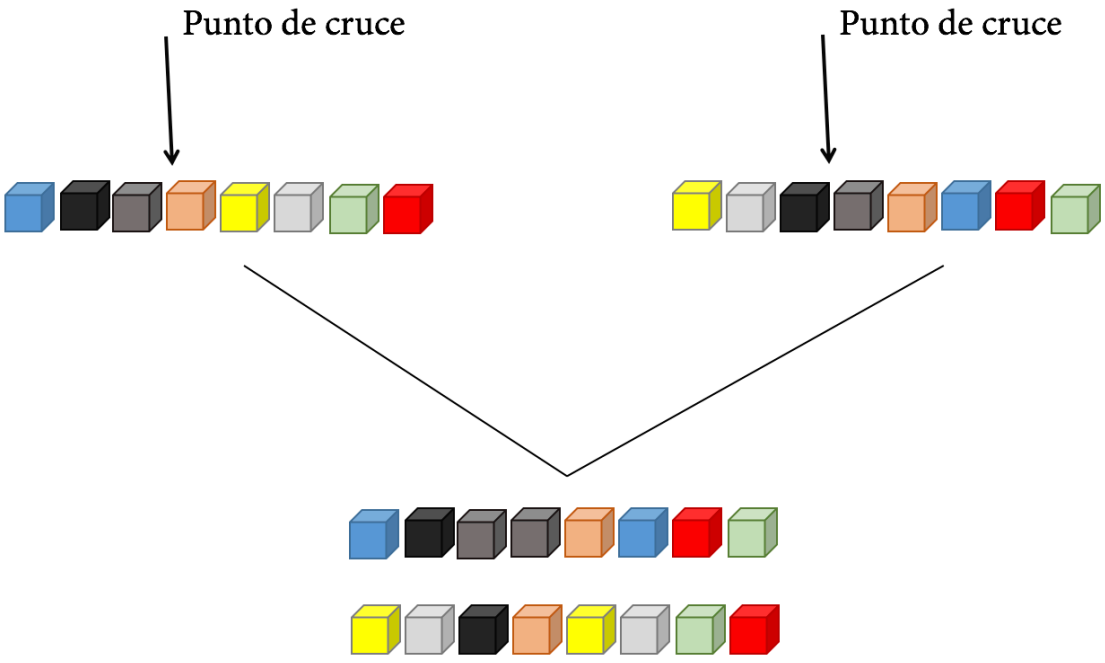

Figura 27 Cruce Genético

Para solucionar este problema que se muestra en la *figura 27*, se realiza un proceso de limpieza que consiste en quitar los genes duplicados de un plan. Para ello se obtienen todas las actividades de las ciudades elegidas y se descartan todas las actividades que ya se han elegido para no volver a duplicar actividades. Después obtenemos una actividad de forma aleatoria y la sustituimos por la repetida. Nuevamente encontramos un problema ya que al cambiar las actividades cambia sus distancias entre actividades por lo que lanzamos un proceso que recalcule esas distancias y los tiempos para mantener la coherencia del plan.

Ahora ya tenemos cuatro cromosomas diferentes de los cuales evaluamos la eficacia de los cromosomas cruzados que son los que cambian.

A continuación, realizamos un proceso de ordenación para asegurarnos tener la población ordenada por los planes con mejor eficiencia.

Una vez que hemos recorrido los *n* planes que tenemos que cruzar obtenemos *n**2 planes con sus eficiencias correspondientes ordenadas de mayor a menor por lo que nos queda una generación de *n**2 planes y pasamos a descartar la mitad de los planes para mantener la consistencia de *n* planes.

Para este proceso hemos decidido salvar siempre a el 5% de los mejores para que pasan automáticamente a la siguiente generación y para los *n*-(5%) restantes realizamos de nuevo un proceso de ruleta que los agrupa por su probabilidad como hemos explicado anteriormente y a esos *n*-(5%) restantes les aplicamos antes un proceso de mutación antes de pasarlos a la siguiente generación.[19]

El proceso de mutación consiste en modificar un gen por otro de forma aleatoria. Como hemos explicado antes nuestra probabilidad de mutación es de un 1%.

Para este proceso recorremos todos los genes que tiene un cromosoma y lanzamos un *random* de 1 a 100, si sale 1 mutamos por lo que obtenemos de nuevo un plan aleatorio de las ciudades indicadas y descartando los que ya tenemos para evitar duplicados y lo sustituimos en el cromosoma. Si sale otro número no mutamos.

Ahora ya si tenemos la siguiente población lista para un nuevo proceso de evolutivo, pero antes de calcular la siguiente generación calculamos la eficiencia media de esa generación de esta manera podremos observar la evolución que se produce entre las distintas generaciones. Una vez obtenida dicha generación repetimos el proceso hasta que termine el número de poblaciones que tenemos previamente definido.

7.5.4 Finalidad y uso

Este algoritmo fue analizado y diseñado para poder encontrar una alternativa a un planificador.

La idea principal era usarlo como una herramienta que nos permitiese valorar la evolución de un plan de tal forma que pudiésemos decidir si merecía la pena invertir tiempo y recursos en generarlo de esta forma ya que es un algoritmo bastante costoso.

Después de implementarlo hemos podido realizar un estudio de la evolución de las siguientes generaciones que nos ha permitido demostrar cómo el proceso a pesar de ser costoso consigue mejorar la población inicial pero que a medida que transcurre el tiempo se estabiliza hasta llegar a un punto casi muerto de no evolución.

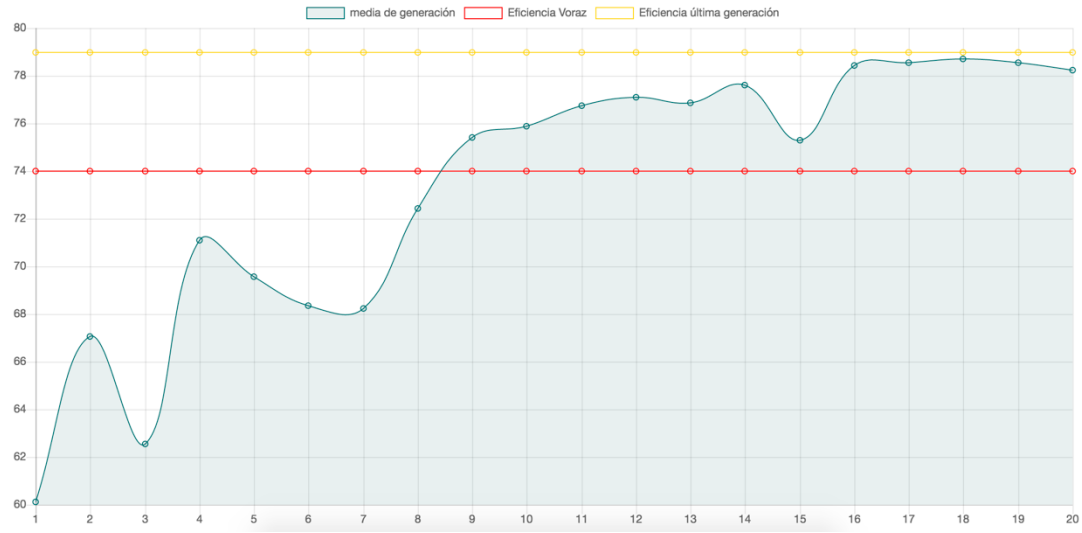

Figura 28 Gráfica evolución generaciones

Como podemos observar en esta *figura 28* la línea verde nos muestra la evolución transcurrida entra las diferentes generaciones, la línea roja muestra la eficiencia que tenía el voraz con el que parte como élite el algoritmo y la línea amarilla muestra la eficiencia del mejor plan de la última generación.

Ya que tenemos este algoritmo decidimos utilizarlo como hemos visto antes para desarrollar un recomendador de planes. Como con el algoritmo obtenemos la mejor generación de todas y dentro de esa generación están todos los planes

ordenados por eficiencia de mayor a menor lo que hacemos es calcular cuál son esos planes y devolverlos a la aplicación para mostrárselos a los usuarios.

7.5.5 Ventajas y desventajas

- No se requiere conocimientos del problema a resolver muy específicos para poder implementar este algoritmo.
- Trabaja a la vez con diferentes eficiencias calculadas para cada plan de manera simultánea.
- Usan operadores probabilísticos.
- Necesita mucho tiempo y consumo de recursos para realizar la creación de planes que finalmente acaban descartados.
- Puede converger rápidamente la función dependiendo de cómo elijamos los planes que pasarán a la siguiente generación y de la probabilidad de mutación que usemos.

7.6. SIMULADOR

Esta funcionalidad se implementó como herramienta para los desarrolladores con el fin de optimizar el tiempo invertido en las pruebas ya que tener que hacer todo el proceso de selección de sitios es una tarea bastante tediosa, además, cada vez que se quiere utilizar un algoritmo implica tocar el código para cambiar la llamada a nuestra *API* y que ejecute el algoritmo deseado.

Como herramienta diseñada para desarrolladores sólo es accesible para usuarios con el rol de administrador como vemos en la *figura 29*.

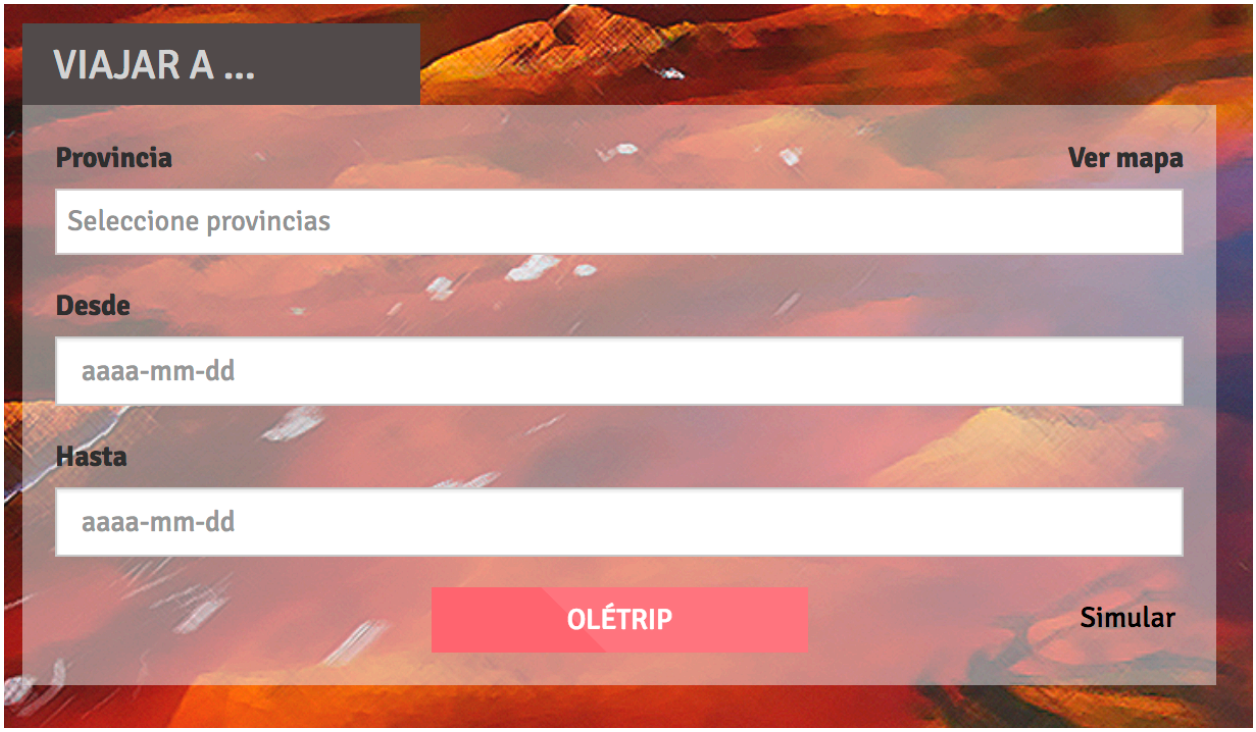

Figura 199 inicio admin con opción simular

El funcionamiento es muy sencillo. El desarrollador simplemente debe introducir las ciudades, el número de días, las actividades, el modo y el algoritmo sobre el cual se desea hacer las pruebas.

Internamente se obtienen de forma aleatoria tantos sitios como actividades se hayan elegido filtrando por las ciudades elegidas. En caso de no seleccionar ninguna ciudad, realiza la misma tarea, pero sin usar ese filtro.

Cuando ya tiene los sitios puede crear un plan de forma normal usando el algoritmo seleccionado.

A parte de usarlo como herramienta para ahorrar tiempo en las pruebas nos permite tener una valoración visual del algoritmo genético. Cuando se usa este algoritmo al finalizar su ejecución nos muestra una gráfica que nos permite ver la evolución de las generaciones como se muestra en la *figura 30*.

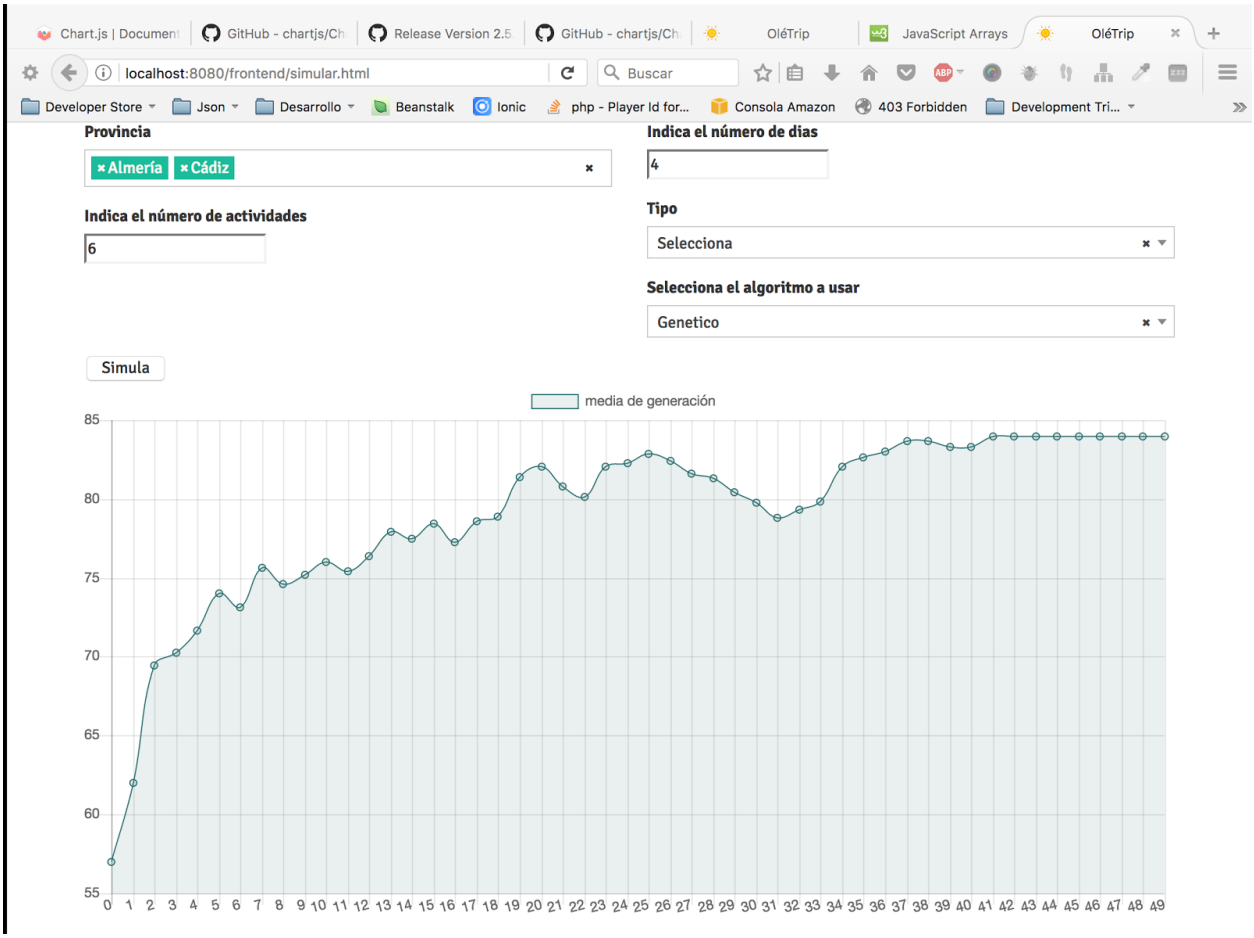

Figura 30 Gráfica evolución generaciones

8. EVALUACIÓN CON USUARIOS

8.1. OBJETIVOS DE LA EVALUACIÓN

Después del desarrollo de los algoritmos de búsqueda descritos en el documento y enfocarlos de diferentes maneras hemos realizado un pequeño estudio con usuarios para recibir *feedback* que nos ayude a mejorar la aplicación.

Esta evaluación no está dirigida al uso de toda la aplicación ya que ese trabajo fue realizado por los anteriores miembros del proyecto.

8.2. PARTICIPANTES

Los participantes con los que hemos contado para este estudio tienen edades comprendidas entre los 25 y los 60 años, tanto hombres como mujeres, el número final de encuestados ha sido de 12 personas de entre ellos hay diferentes perfiles profesionales y gustos turísticos.

8.3. DESARROLLO DE LA EVALUACIÓN

El escenario de la evaluación de los encuestados es el siguiente:

Cada integrante del grupo ha estado presente en tres evaluaciones con el fin de resumir la funcionalidad de la aplicación y sus objetivos.

Después de explicar al usuario el motivo de la encuesta empezamos preguntando algunos datos de cada usuario como la edad y el sexo, también preguntamos al usuario sobre sus preferencias a la hora de organizar un viaje.

El escenario continuó solicitando al usuario que cree un plan personalizado para que luego se le muestre el plan construido usando los tres algoritmos implementados.

Una vez contestadas las valoraciones sobre los planes personalizados someteremos al usuario a probar nuestro recomendador que utiliza el algoritmo genético para que lo valoren según sus experiencias.

Finalmente hicimos una serie de preguntas respecto a la valoración general de la aplicación y la experiencia del usuario.

8.4. RESULTADOS OBTENIDOS

A continuación, vamos a detallar las preguntas que se han realizado a los usuarios encuestados, los resultados obtenidos y la valoración que hemos concluido después de realizar este estudio.

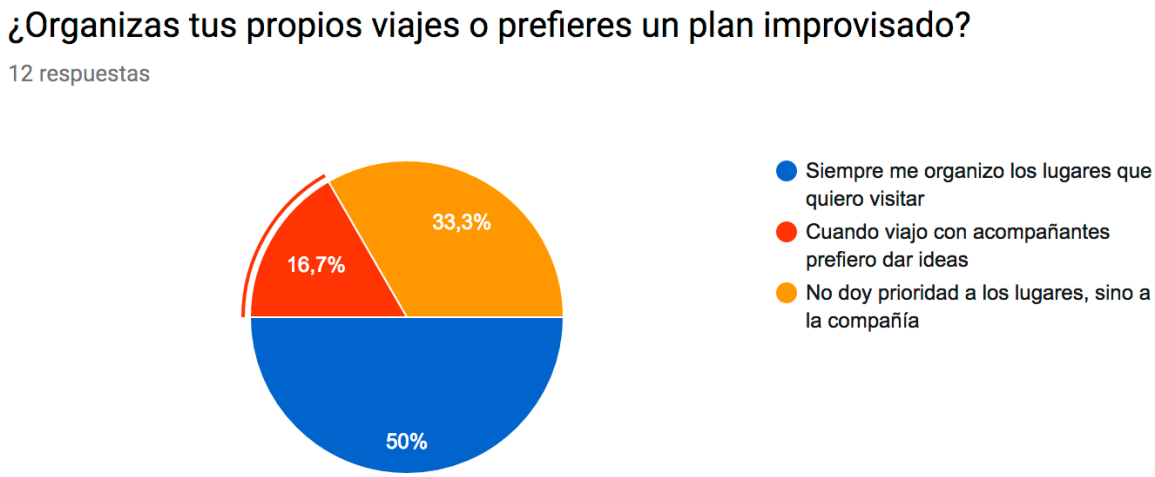

Figura 31 Encuesta 01

Con esta pregunta (*figura 31*), pretendemos averiguar si los usuarios encuestados se ocupan de la organización de un viaje. De esta manera podemos concluir que, de ser así, la aplicación les puede ayudar con la información de la que disponemos y podemos ofrecerle al usuario y en caso contrario dar valor al recomendador de planes desarrollado. Por lo general suelen organizar sus viajes, pero en la mitad de los casos usar un recomendador de viajes es útil para los usuarios.

Las tres preguntas siguientes hacen referencia a una prueba realizada con cada uno de los algoritmos desarrollados. Con esta pregunta pretendemos obtener *feedback* del usuario a cerca de la experiencia que han tenido usando los distintos algoritmos, aunque para ellos sea transparente.

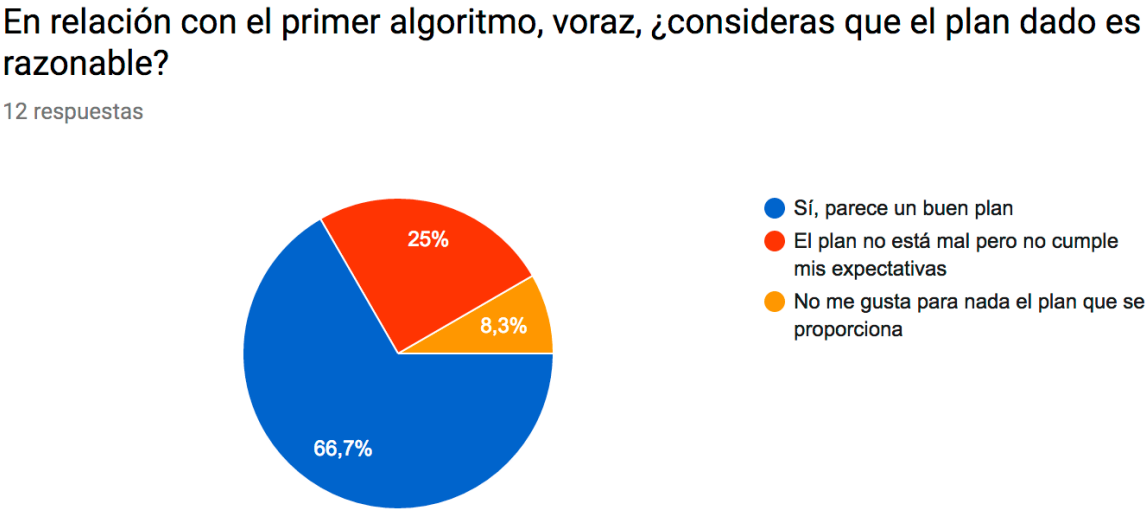

Figura 32 Encuesta 02

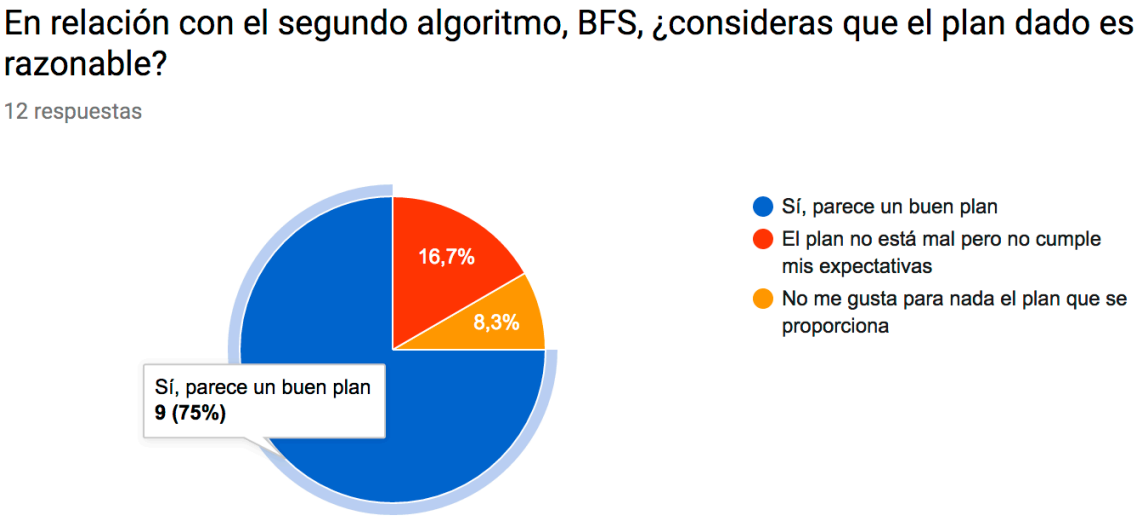

Figura 33 Encuesta 03

Haciendo una comparativa entre el plan generado al usuario con el algoritmo voraz (*figura 32*), y el algoritmo BFS (*figura 33*), observamos resultados muy similares. Se puede observar que ha habido preferencia por el BFS. Esto se debe, teniendo en cuenta que el usuario no entiende del funcionamiento del algoritmo, a que simplemente el plan se ajustaba mejor a sus gustos. Cabe destacar que el BFS puede generar un plan de características igual o mejor que el voraz.

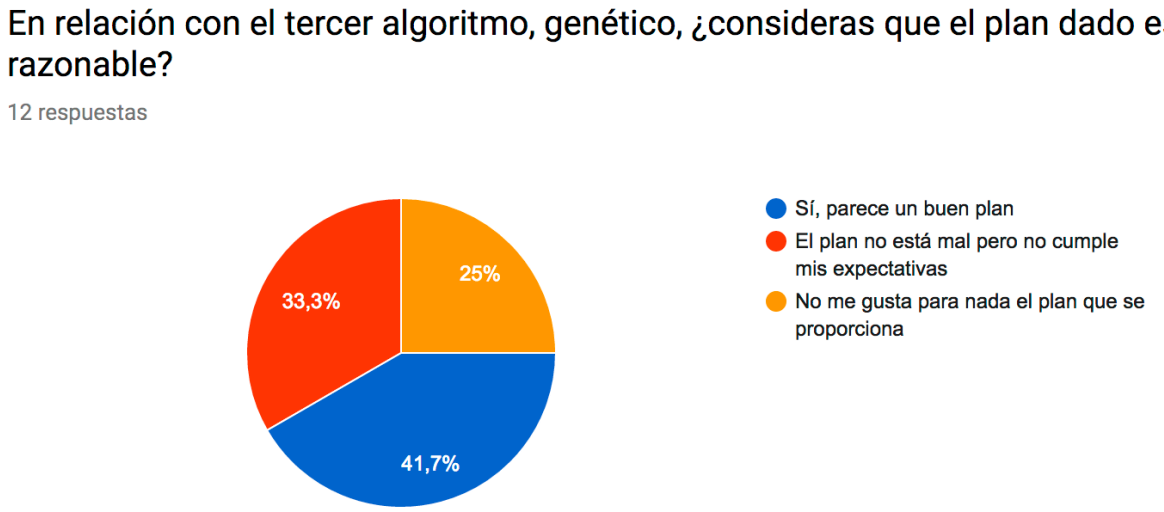

Figura 34 Encuesta 04

Para el caso del uso del algoritmo genético (*figura 34*) vemos que no están conformes, cosa normal ya que modifica los planes por la repetición de planes como explicamos en el funcionamiento del algoritmo, por eso es buena opción usarlo como recomendador y no como generador de un plan establecido por el usuario.

Con la siguiente pregunta pretendemos comprobar si el usuario valora positivamente que la aplicación añada actividades al plan en el caso de que quede tiempo libre una vez visitado los lugares que le interese.

Por lo que podemos deducir que es una buena idea tener este desarrollo en la aplicación ya que a la mayoría de personas les parece buena idea. Entendemos que los usuarios que solo quieren visitar los lugares escogidos son los más rigurosos puesto que preparan sus viajes totalmente al detalle.

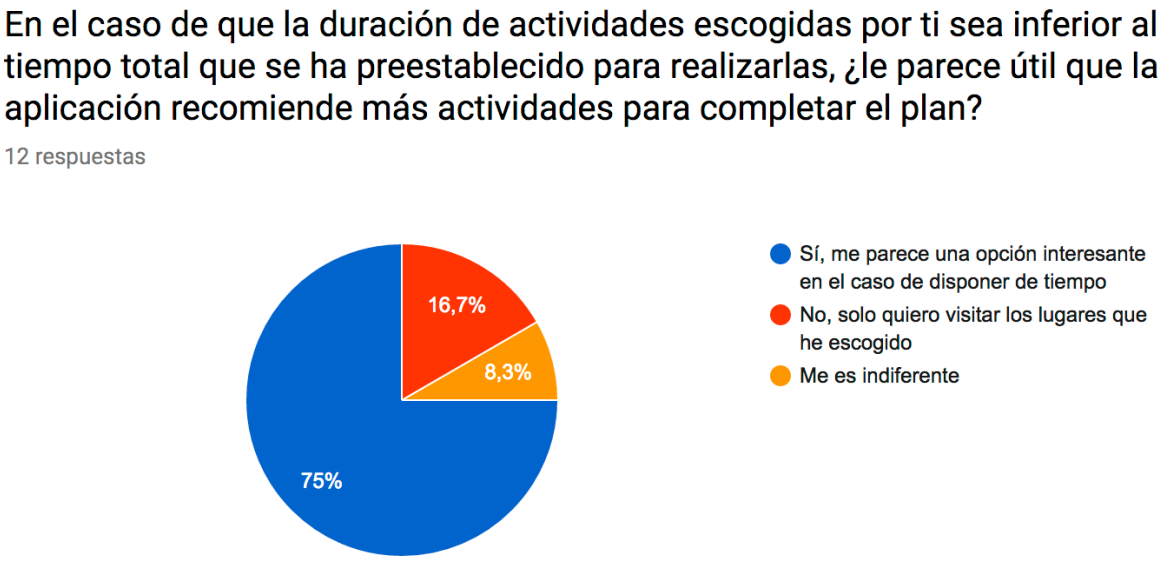

Figura 35 Encuesta 05

La aplicación cuenta con gran carga de información la cual consideramos bastante útil para ayudar al usuario, pero nos interesa saber si a ellos les resulta tan importante para futuras versiones de la aplicación atender a la demanda de los usuarios.

Podemos concluir en la *figura 36* que la información proporcionada es buena, pero a los usuarios les gustaría tener más detalle de las jornadas de las actividades e incluso la necesidad de incluir hoteles o restaurantes.

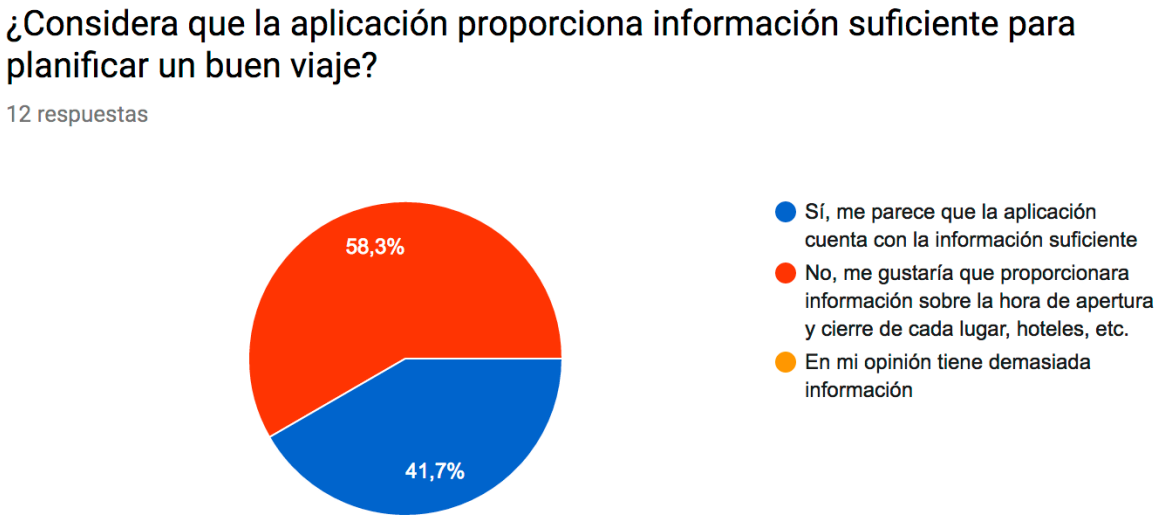

Figura 36 Encuesta 06

Esta pregunta (*figura 37*), se ha realizado a propósito para añadir valor a la nueva interfaz y funcionalidad que hemos creado para el recomendador de planes mediante el uso del algoritmo genético.

Las respuestas posibles serán las siguientes:

- Es una buena solución para los más indecisos.
- Lo usaría en ciudades que no hubiese demasiados lugares a visitar.
- Me gusta que ofrezca varias opciones.
- Debería ser más rápido.
- Ningún plan de los ofrecidos me ha convencido.

Podemos concluir que tiene potencial para un gran número personas que no tienen prioridad visitar unos sitios u otros o incluso como un aporte de ideas para poder empezar a valorar un viaje. Debería mejorarse el tiempo que necesita para calcular los planes ya que en algunos casos requiere de demasiado tiempo.

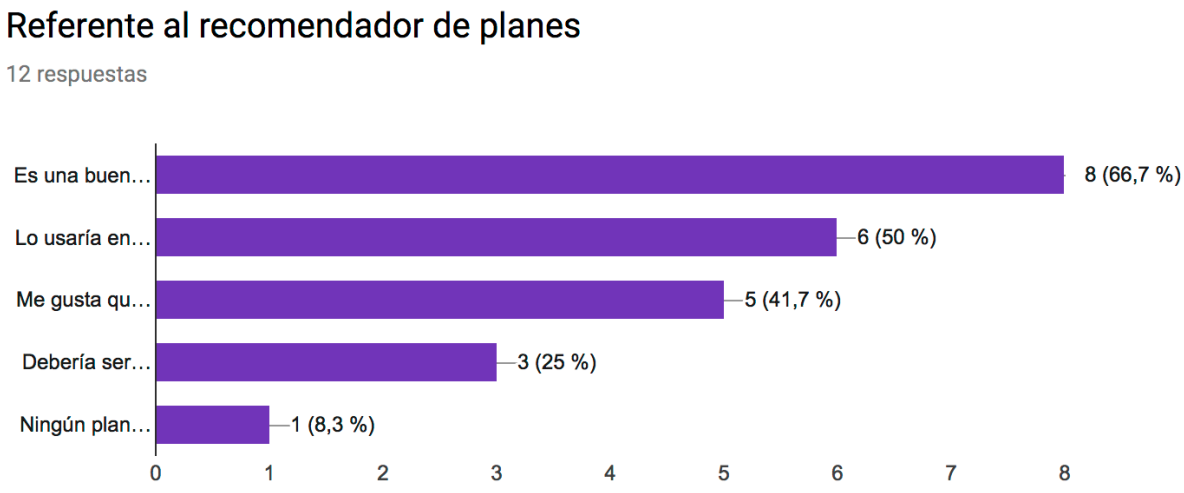

Figura 37 Gráfica evaluación 1

Por lo general a los usuarios encuestados les ha resultado intuitiva y amigable la aplicación y han quedado satisfechos con los planes proporcionados por lo que han valorado positivamente la aplicación en su conjunto como vemos en la *figura 38*.

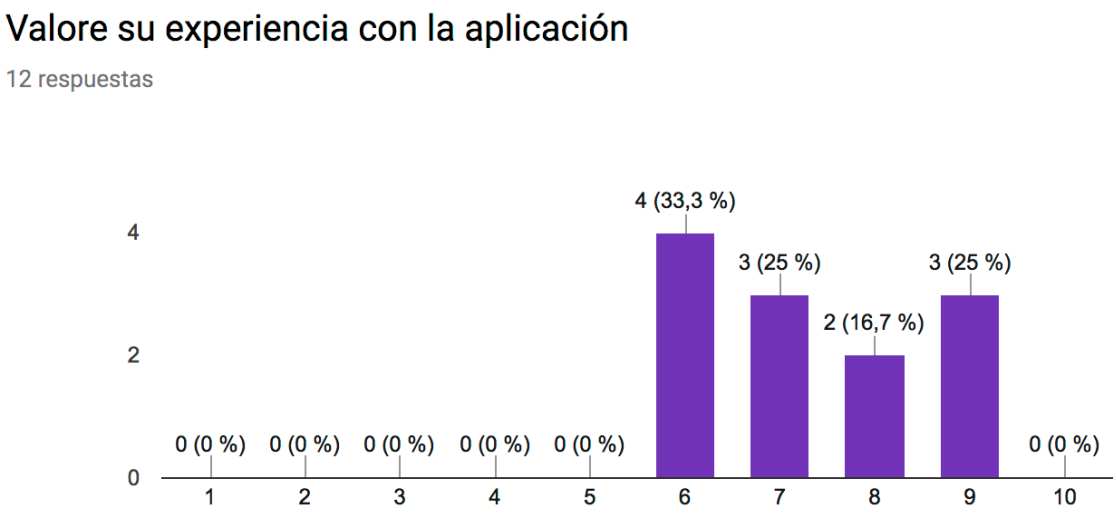

Figura 38 Gráfica evaluación 2

8.5. MEJORAS CONCRETAS A REALIZAR

Como consecuencia de los resultados obtenidos, se ha llegado a la conclusión de que el sistema debería de proporcionar más información acerca de hoteles disponibles en las ciudades, además de los horarios de apertura y cierre de los lugares. Parte de los usuarios encuestados prefieren que hubiera la posibilidad de elegir el hotel de cada ciudad que quiera visitar.

En referencia a autocompletar planes en el caso de que sobren días, sería interesante que el usuario pudiera elegir si desea que el plan se rellene con sitios recomendados para realizar en el tiempo restante o no incluir más sitios además de los que el usuario haya elegido.

También sería interesante que el usuario pudiera elegir por qué sitio empezar en vez de que el algoritmo que proceda lo genere solo.

9. CONCLUSIONES Y VISIÓN FUTURA

Como consecuencia de no encontrar un planificador que se adapte a las necesidades del proyecto aparte de la complejidad del lenguaje *PDDL* decidimos entre el equipo desarrollador y los tutores implementar los algoritmos descritos a lo largo de la memoria.[20]

Todos ellos nos proporcionan un plan usando sus mecanismos. Está claro que si el usuario busca un plan óptimo y busca que todos sus lugares escogidos estén sin ninguna duda en el plan recomendado, se debe optar por el algoritmo voraz o por el BFS, para aquellos usuarios que no tengan claro qué visitar el recomendador de planes generado por el Genético es la mejor opción. A pesar de esto, de cara al usuario el tipo de algoritmo que se utilice le va a ser indiferente.

En nuestra opinión la aplicación cubre de manera amplia los requisitos que un usuario podría pedir a la hora de organizar un viaje por Andalucía ya que al usuario se le proporciona la información principal (duración del viaje, distancia entre lugares, día en que se desarrolla la actividad, etc.). A pesar de eso la aplicación podría contar con más información como se nos ha insinuado en la evaluación de usuarios como es el caso de incluir más hoteles (solo tiene uno por ciudad), e incluso restaurantes, que la aplicación se extrapole a toda España y no solo a Andalucía, etc., que básicamente consistiría en aumentar la información en la base de datos.

En relación al aumento de hoteles también podría ser buena opción añadir reserva de hoteles.

Otra mejora, sería añadir horarios real y precios de los lugares turísticos para que en caso de que un sitio esté cerrado un día no lo contemple a la hora de generar el plan. Como visión mucho más futura que desde la aplicación se pudiese realizar dicha reserva.

Sería interesante realizar modificaciones de un plan en tiempo real, por ejemplo, si un usuario está ya disfrutando de sus vacaciones y surge un imprevisto que pueda amoldar su plan a sus necesidades de ese momento. Esto se podría hacer utilizando el recomendador de planes, pero adaptándolo solo al día y ciudad en concreto.

La aplicación también podría traducirse en otros idiomas para los turistas que no hablen el español, sin ir muy lejos al inglés ya que es un idioma muy hablado a nivel mundial.

Si este proyecto se amplía se podrían implementar fácilmente estos cambios que, lamentablemente, a nosotros nos ha sido imposible desarrollar por falta de tiempo.

En definitiva, es un proyecto en el que se puede trabajar todavía en bastantes aspectos, partiendo de la base que ya es capaz de generar un buen plan, con la posibilidad de añadir detalles que para los usuarios sea útil.

10. CONCLUSIONS AND FUTURE WORK

As a result of not finding an automated planner that complete our necessities for the application and the difficulty of the PDDL language to be dominated in short time we decided as a team with our managers to implement the algorithms that are described in this memory.

Our App provides for the users several types of algorithms to calculate the best route using its own mechanisms, for users that wants to choose their whole trip without any changes and using the best route for it the Voraz and BFS algorithms are the recommended to use. In the other hand for users who are open to any plan and doesn't know what to visit the Genetic algorithms is their best choice. Despite this the type of algorithm that is used it will be indifferent for the users.

In our opinion the app has enough information for the users to organize a plan in Andalucía since the user is provided with the main information (journey duration, distance between places, the day of the activity, etc).

In the future the application could have more information, like adding more hotels (every city have only one hotel) and for example restaurants. It could be extended to all the regions of Spain and not only Andalucía, basically by adding more places to the database, we came with these conclusions from the user evaluation that we applied during the process.

Another option for the hotels could be adding an option to reserve a room hotel from the App.

Another improvement, would be to add real timetables and prices of tourist sites so that in case a site is closed one day the plan replaces this site with another or just remove this site. Reserving tickets to visit touristic places from the app good be a good idea too.

It could be interesting to make modification to a trip in real time vacation, for example if a user is already on the trip and for some reason he needs to change a plan he can do it in this moment with his device. It could be used in the recommender of plans adapting it only for the indicated day and city.

The App could also be translated in other languages for tourists who don't speak Spanish, for example to English because of the high amount of people who talks English in the world.

All these modifications could be added in the nearly future, unfortunately we couldn't do more due to the short time.

Finally, as a final conclusion this app could be improved more from some different aspects, starting from generating good plans with the possibility of adding some details that are useful for users.

11. APORTACIONES

A continuación, se expone las aportaciones de cada miembro del grupo al proyecto. Hay que señalar que el trabajo se ha repartido de la forma más equitativa posible.

11.1. APORTACIONES DE EDUARDO

Lo primero que quiero señalar es que el desarrollo del proyecto lo he realizado más conjuntamente con Luis, sobre todo a la hora de desarrollar el código.

Primeramente, hablaré sobre mis aportaciones en la implementación:

En primera instancia los profesores nos sugirieron familiarizarnos con el código heredado del anterior proyecto. Así pues, comenzamos a realizar un primer algoritmo aleatorio, el cual serviría luego útil para otro algoritmo, el genético. Mi compañero Luis y yo desarrollamos este algoritmo aleatorio de manera que respetase la estructura diseñada previamente para que el plan pudiera ser mostrado al usuario correctamente.

Posteriormente, para realizar otro algoritmo, el BFS se requería que la aplicación solicitara multitud de llamadas a la *API de Google* para calcular distancias y tiempos entre lugares, etc.

Esto supuso que cambiásemos todas las llamadas a la API por llamadas a la base de datos. Mi compañero Luis y yo creamos las nuevas tablas que necesitamos: *hotels, infoapi, infoapihoteles, distanciashotelesciudades, infoapihoteles*. Gracias a estas tablas el algoritmo BFS no daría problemas por el excesivo uso de las llamadas a la *API de Google*. Como consecuencia cambiamos el código necesario para que se llamara a la base de datos. También, en el código, mapee todas las tablas en *Hibernate* para que las consultas a la base de datos tuvieran efecto.

El BFS, a pesar de invocar a un algoritmo voraz, requería de unos parámetros distintos por lo que opté por realizar otra clase java para que realizase este algoritmo, pero recibiendo por argumento diferentes parámetros.

Hay que decir que, debido a que nos surgían dudas sobre cómo implementar el algoritmo, mi compañero Luis y yo solicitamos una tutoría a Antonio para que nos las solventara.

Ayudé a mi compañero Luis con la implementación del validador de planes.

Cada cambio importante que hacía en el código lo subía al repositorio para que mis compañeros pudieran tener la última versión y trabajar con ella.

Por otro lado, me encargué de implementar el código necesario para que se recomendara lugares si el número de lugares que deseaba visitar el usuario era escaso en relación con el número de días seleccionado. Lo realicé tanto para el BFS como para el voraz.

Una vez terminado con la implementación, hablaré ahora sobre el desarrollo de la memoria.

En primer lugar, nos reunimos todo el equipo para realizar la estructura y ver cuál era la que mejor se adecuaba a nuestras necesidades. Cuando la estructura estaba terminada, hablamos con nuestros profesores para que dieran su visto bueno y realizar algún cambio si fuera necesario.

Una vez definida y aprobada por los profesores, me centré individualmente en el capítulo 1, concretamente en el punto 1.2, 'Estructura de la memoria'; posteriormente en el capítulo 3 (completo), 'Estado del arte'; finalmente en el capítulo 7.2, 'Voraz'.

He ayudado también a mi compañero Jorge en el capítulo 4.3.2 a completar la información sobre las diferentes librerías que hemos utilizado para el desarrollo del código en lo que se refiere al *Frontend*.

Ayudé a mi compañero Alberto a realizar la descripción funcional (capítulo 5) para hacer que estuviera lo más detallado posible tanto a nivel de descripción como de figuras.

Posteriormente realizamos la evaluación de usuario la cual conté con la colaboración de familiares y amigos para hacer posteriormente una valoración de los resultados con todo el equipo.

Todo el equipo colaboró en el capítulo 9, 'Conclusiones y Visión Futura' en el que se hace una valoración final del proyecto.

Todos realizamos una revisión completa de la memoria para que el texto tenga sentido y no queden frases que no se comprendan o no se sepa de qué se habla.

11.2. APORTACIONES DE LUIS

En primer lugar, al ser la continuación de un proyecto, el primer paso era familiarizarse con el código.

Aunque gran parte del proyecto lo hemos hecho en conjunto reuniéndonos los cuatro miembros, a la hora de dividirnos tareas lo hemos hecho en grupos de dos, por lo que la mayoría de mis aportaciones son en conjunto con Eduardo.

Para ello, el primer paso fue realizar un algoritmo aleatorio, que nos fue de ayuda para entender mejor todo el proceso desde que el usuario introducía los parámetros, con los días, horas, actividades etc. hasta que recibe el plan ya generado.

Una vez fuimos capaces de entender todos los pasos que se iban siguiendo y podíamos modificar el código para obtener los resultados que quisiéramos, lo siguiente fue desarrollar algoritmos que generasen mejores planes.

Junto a mi compañero Eduardo nos pusimos con el algoritmo BFS, para el cual también implementamos una cola de prioridad, que nos permitiese ordenar los planes en función de unos parámetros.

Para esto, también implementamos un método de validación de planes, donde pudiésemos tener en cuenta esos parámetros para valorar los planes.

Para los planes generados por el BFS nos apoyamos en el algoritmo voraz, pero tuvimos que introducirle algunos cambios.

Además, con el BFS al ejecutar multitud de veces el método voraz, se llegaba rápidamente al límite de consultas de la *API de Google*, de manera que hubo que cambiar todas estas llamadas a la *API* por consultas a las base de datos, de forma que tuvimos que generar varias claves para poder exceder el límite de llamadas a la *API* mientras nos encargábamos de rellenar la base de datos con toda la información que pudiésemos necesitar y así en un futuro no necesitar más que consultas en la base de datos y poder prescindir del resto.

Para ello rellené nuevas tablas *infoapi*, en un principio, con toda la información que se obtenía de estas llamadas a la API, de manera que solo tuviéramos que depender de consultas a la base de datos.

Posteriormente y para poder utilizarlo para la validación de los planes introducimos las tablas *hotels, infoapihoteles, distanciahotelesciudades y infoapihoteles* para poder obtener información de distancias con respecto a los hoteles de cada ciudad.

Añadimos además todas todo lo necesario en *Hibernate* para que la base de datos estuviese bien sincronizada.

Realizamos también algunas modificaciones en el algoritmo aleatorio y en el método de validación de planes para que estos pudieran ser usados para el genético

También con mi compañero Eduardo, implementamos un recomendador de planes para los casos en los que ya se han rellenado el plan de las actividades que el usuario deseaba visitar y todavía queda tiempo para realizar más actividades.

Con respecto a la memoria, primero nos reunimos en conjunto para organizar la estructura de la memoria, una vez organizado, primero me encargue de explicar en la sección de mecanismos de creación de ruta el algoritmo BFS y ayudar en el del voraz.

Posteriormente explicar la motivación que nos llevó a hacer el algoritmo aleatorio, así como su funcionamiento.

Una vez finalizada la sección de mecanismos de creación de rutas continué con la arquitectura del sistema, la parte de planificación de un viaje, en la que se explica la estructura de un plan para que sea más fácil entender en qué consiste a la hora de mencionarlo a lo largo de la memoria.

También la estructura de la base de datos, la explicación de por qué se insertó nuevas tablas y la utilidad que le dimos.

La evaluación de calidad de los planes y en qué consiste y los parámetros que utilizamos para valorar un plan.

Nos organizamos para realizar cada uno la evaluación de usuarios con un cuestionario en conjunto y nos reunimos para evaluar las respuestas y obtener conclusiones de los resultados obtenidos.

Finalmente organicé el índice de imágenes y tablas, así como el enlace entre ellas.

11.3. APORTACIONES DE ALBERTO

Antes de comenzar a detallar mis aportaciones realizadas cabe destacar que por norma general todo el trabajo realizado a lo largo de estos meses se ha repartido por parejas ya que somos un grupo de cuatro personas y consideramos que era la forma más eficiente de trabajar.

La persona con la que he realizado más trabajo en conjunto ha sido Jorge.

A continuación, detallaré una lista con las tareas más relevantes en las que he participado.

- Instalación y configuración del entorno

Para poder trabajar con este proyecto heredado requerimos de una configuración del entorno *Java* con *Apache Tomcat*. Era la primera vez que nos veníamos en la necesidad de contar con un entorno de trabajo así y debimos configurarlo por lo que me tocó investigar cómo montar este ecosistema que nos permitiese trabajar de la forma más cómoda posible. Después de varios días y más de un intento fallido pudimos configurar el entorno tanto en *IOS* como en *Windows*. Esta tarea la realice junto a Jorge y posteriormente ayudamos a nuestros compañeros a configurar sus entornos.

Una vez configurado el entorno era el momento de familiarizarnos con el código. Para ello nuestros tutores nos propusieron crear un sencillo planificador basado en creaciones de planes de forma aleatoria. Esta tarea la realizamos entre todos los miembros ya que todos necesitábamos conocer cómo y dónde se generaban los planes que posteriormente íbamos a tener que modificar. El análisis de esta tarea fue realizado por todos los miembros del grupo, pero la mayor parte del desarrollo fue llevada a cabo por Eduardo y Luis.

- Repositorio de código

Como hemos explicado anteriormente elegimos como repositorio de código *Bitbucket* y la herramienta *SourceTree*. Esta decisión la tomamos entre Jorge y yo ya que en la actualidad la usamos para trabajar y consideramos que es sencilla.

Por tanto, entre mi compañero y yo nos encargamos de crear las cuentas para los otros miembros del grupo e instalar en sus ordenadores las herramientas necesarias y darles las pautas básicas para trabajar.

Una vez creado el grupo de trabajo decidimos generar 2 repositorios de trabajo que nos permitieran mantener separados la parte *backend* de la parte *frontend*, yo me encargué de la subida inicial del back y *Jorge* de la subida inicial del *front*.

Posteriormente acordamos cuales serían nuestras ramas principales y cómo generaríamos las ramas secundarias para la generación de los nuevos desarrollos.

- Simulador

El simulador de código fue una iniciativa que surgió de la necesidad de ahorrar tiempo en las pruebas y la depuración del código ya que, aunque generar un plan con nuestra aplicación no requiere de mucho tiempo, cuando tienes que generar planes a cada instante se convierte en una gran suma de tiempo invertido.

Por este motivo Jorge y yo comenzamos este desarrollo que posteriormente fue cobrando más importancia ya que decidimos incluir las gráficas para el algoritmo genético y la visualización del plan generado.

- Algoritmo genético

Al no encontrar un planificador que satisficiera nuestras necesidades, nuestros tutores nos dieron la alternativa de buscar unos algoritmos que pudiéramos reemplazarlo.

Fue en este momento donde surgieron en algoritmo BFS y Genético, ambos fueron analizados por todos los miembros del grupo, pero yo me encargué junto con Jorge del desarrollo completo del genético

Para el desarrollo de este algoritmo tuvimos buscar mucha información y ejemplos en internet para saber que era y cómo desarrollarlo ya que era la primera vez que oíamos hablar de este algoritmo. Fue fundamental la ayuda de nuestros tutores ya que nos aportaron documentación y nos dedicaron algunas tutorías para consultar dudas y resolver problemas que nos íbamos encontrando.

- Recomendador

El último desarrollo que realicé fue la creación de una funcionalidad que nos permitiese dar valor al algoritmo genético. Por esta razón se nos ocurrió a mi compañero Jorge y a mí desarrollar esta parte y presentársela a los profesores.

- Memoria

Mi aportación para este documento ha sido la explicación del algoritmo genético, buscar información sobre otras aplicaciones similares, realizar evaluación de usuarios, explicar las nuevas interfaces, explicar un ejemplo de uso de la aplicación, aplicar los estilos para el formato definitivo y revisión.

11.4. APORTACIONES DE JORGE

Nada más tener todos los datos y material necesario para seguir con el proyecto del año pasado la primera tarea fue montar el entorno de desarrollo en nuestros ordenadores locales, de esta tarea nos hemos encargado yo y mi compañero Alberto, tuvimos que investigar un poco sobre todos los requisitos que necesita la aplicación para funcionar localmente. Después de instalar la aplicación y comprobar que todo funciona perfectamente tanto para el uso como para desarrollo decidimos crear un manual de instrucciones para configurar el entorno que se encuentra en el segundo apartado del capítulo de Apéndices.

La siguiente tarea fue subir el código tanto el *backend* como el *frontend* a un repositorio de datos entonces yo me encargue de subir la parte *front*.

A partir de ahí empezamos a familiarizarnos con el código, esa tarea se empezó después de una reunión con los tutores que nos propusieron desarrollar un algoritmo sencillo que es el aleatorio, entonces cada miembro del grupo empezó a probarlo por su cuenta y apoyarnos cuando un miembro del grupo tenía alguna duda en la aplicación.

Tras trastear con el código implementando un algoritmo aleatorio, mi compañero Alberto y yo vimos la necesidad de implementar un simulador de planes debido a que a la hora de hacer pruebas con el algoritmo aleatoria el proceso de crear un viaje personalizado nos llevaba y nos hacía perder mucho tiempo. Entonces implementamos el simulador entre los dos y creamos el nuevo campo de *Admin* dentro de la tabla *User* en la base de datos para que en función de los permisos se pueda acceder a esa pantalla o no.

Después de familiarizarnos con el código nos pusimos a explorar junto a los tutores como queremos implementar un buen planificador de rutas para la aplicación, y es cuando nuestros tutores nos propusieron investigar un poco sobre la planificación automática. Entonces nos dividimos en dos equipos unos para buscar un planificador que se adapte con nuestras necesidades y otros mirar tutoriales sobre el lenguaje *PDDL*. Yo junto a mi compañero Alberto nos dedicamos a mirar la parte de buscar un planificador que se adecuara.

Una vez que hemos visto que la idea de la planificación automática era inviable para nuestro proyecto por varios motivos, nos propusimos dos algoritmos como alternativa que son el BFS y el algoritmo genético, a mi junto a mi compañero Alberto nos correspondió la parte de investigación y el desarrollo de este algoritmo.

Tuvimos que ir varias veces a tutorías con el profesor Antonio Sánchez para que nos ayudará a entender el funcionamiento de este algoritmo para poder implementarlo.

Cuando se terminó la implementación del algoritmo genético, y tras ver los resultados que mostraba ese algoritmo, vimos que cambiaba los planes elegidos por el usuario de forma que el plan que crea el usuario sale muy distinto al plan inicial debido a como se había desarrollado el algoritmo. Entonces pensamos en aprovechar ese algoritmo y crear un recomendador de planes de forma que el usuario no tenga que elegir los sitios que quiere visitar y dejar que nuestra aplicación le recomiende los sitios con sus mejores rutas posibles.

Otra de las tareas que me tocó hacer es la implementación inicial del evaluador de calidad de los planes y definir sus requisitos, claro que la segunda parte fue junto a todos mis compañeros.

Otra aportación mía fue la idea de crear un proyecto en un gestor de tareas que en nuestro caso usamos el *Trello* como hemos explicado anteriormente, también me encargue en llevarlo para organizarnos, reflejar las tareas y comprobar si los miembros del grupo han cumplido las tareas dentro el tiempo para avisarles.

El arreglo de alguna funcionalidad o diseño respecto al proyecto del año pasado como por ejemplo la funcionalidad del *Logout* que a la hora de instalar el proyecto en local no funcionaba (no se reseteaban las cookies) entonces mirando posibles soluciones en foros como el foro de *Stackoverflow*[21] conseguí corregirlo.

Finalmente Respecto a la memoria me ocupe de la parte de distribuir las tareas por el *Trello*, crear el documento inicial en el drive y juntar todas la notas y textos que el grupo tenía por separado, a partir de ahí empezamos a escribir en un documento conjunto, de ahí me encargue de escribir la parte del agradecimiento, las parte del resumen y conclusiones finales en inglés, también me tocó aportar en la parte de tecnologías aplicadas junto a mi compañero Alberto, yo me centré más en la parte *Frontend* y en el capítulo de ‘Herramientas de gestión de desarrollo’. En la parte de arquitectura del sistema contribuí junto a mi compañero Alberto en la parte de interfaz, respecto a la evaluación de usuarios participamos todo el equipo tanto para pensar las preguntas y escribirlas como

para la prueba con el usuario. Y por último respecto a la memoria meter información al apartado de los Apéndices.

APÉNDICES

A. BASE DE DATOS

En este apartado mostraremos las tablas de base de datos que se han modificado y las tablas nuevas que se han creado respecto al proyecto heredado.

Tabla user

Esta Tabla fue modificada y representa los perfiles de los usuarios

Atributos	Tipo	Obligatorio	Comentario
idUser	Int(11)	SI	Es la clave primaria de la tabla que se incrementa automáticamente
bornDate	Date	NO	Representa la fecha de nacimiento
Email	Varchar(255)	SI	Es el campo del correo electrónico además tiene que ser único
Online	Bit(1)	SI	Para saber si el usuario está conectado o no
Passwd	Varchar(255)	SI	Para la contraseña del usuario
Photo	longblob	NO	Por si el usuario quiere poner foto de perfil

shadowProfile	Bit(1)	NO	Para saber si el usuario ha sido invitado o se ha registrado por su cuenta
Tokens	Varchar(255)	SI	Para la identificación del usuario por temas de seguridad
Admin	Bit(1)	SI	Para diferenciar los tipos de usuario si es admin o usuario normal

Tabla 4 User

*los campos con el siguiente color son los que han sido añadidos a las tablas modificadas

Tabla distanciasHotelesCiudades

Esta tabla Representa la distancia entre un sito y el hotel de la misma ciudad

Atributos	Tipo	Obligatorio	Comentario
Sitio	varchar(80)	SI	El nombre del sitio
hotel	varchar(80)	SI	El nombre del hotel
distancia	int(11)	SI	La distancia entre el sitio y el hotel

Tabla 5 DistanciasHotelesCiudades

Tabla hotels

Esta tabla Representa la posición de un hotel dentro de una ciudad

Atributos	Tipo	Obligatorio	Comentario
ciudad	varchar(30)	SI	El nombre de la ciudad
hotel	varchar(30)	SI	El nombre del hotel
x	Double	SI	Coordenada de longitud
y	Double	SI	Coordenada de latitud

Tabla 6 Hotels

Tabla infoApi

Esta Tabla Representa las distancias y el tiempo entre un sitio y otro dependiendo si es en coche o en transporte público.

Atributos	Tipo	Obligatorio	Comentario
idOrigen	int(20)	SI	Representa el id del sitio origen
idDestino	int(20)	SI	Representa el id del sitio destino
distancia	int(20)	SI	La distancia entre el origen y el destino

tiempo	int(20)	SI	El tiempo que se tarda entre el origen y el destino
modo	varchar(20)	SI	Modo coche o transporte público

Tabla 7 InfoApi

Tabla infoApiCiudades

Esta tabla representa la distancia y el tiempo entre el centro de una ciudad y otra dependiendo del modo de transporte.

Atributos	Tipo	Obligatorio	Comentario
origen	varchar(20)	SI	Representa el nombre del sitio origen
destino	varchar(20)	SI	Representa el nombre del sitio destino
distancia	int(11)	SI	La distancia entre el origen y el destino
tiempo	int(11)	SI	El tiempo que se tarda entre el origen y el destino
transporte	varchar(20)	SI	Modo coche o transporte público

Tabla 8 InfoApiCiudades

Tabla infoApiHoteles

Esta tabla representa las distancias y el tiempo entre un hotel origen y un hotel destino dependiendo si es en coche o en transporte público.

Atributos	Tipo	Obligatorio	Comentario
hotelOrigen	varchar(70)	SI	Representa el nombre del hotel origen
hotelDestino	varchar(70)	SI	Representa el nombre del hotel destino
Distancia	int(11)	SI	La distancia entre el origen y el destino
Tiempo	int(11)	SI	El tiempo que se tarda entre el origen y el destino
Transporte	varchar(20)	SI	Modo coche o transporte público

Tabla 9 InfoApiHoteles

B. CONFIGURACIÓN DEL ENTORNO

Para Windows

Instalación de Eclipse y Java:

Descargar e instalar Eclipse Neon:
http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/neon/3/eclipse-jee-neon-3-win32-x86_64.zip&mirror_id=1253

Descargar el JRE necesarios para tu máquina:

http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html

Una vez instalado el entorno de trabajo hay que crear un workspace en la ruta que se desee trabajar.

Instalación BACKEND:

Primero abrimos Eclipse para posteriormente importar el proyecto como se muestra en la *figura 39* y *figura 40*.

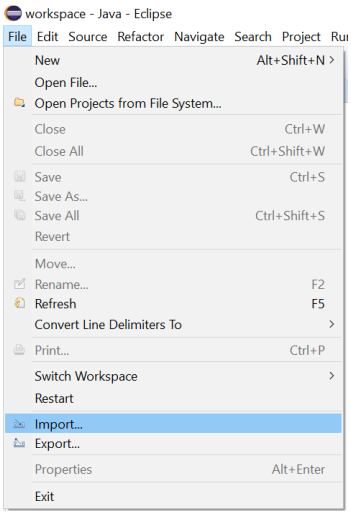

Figura 39 instalación 01

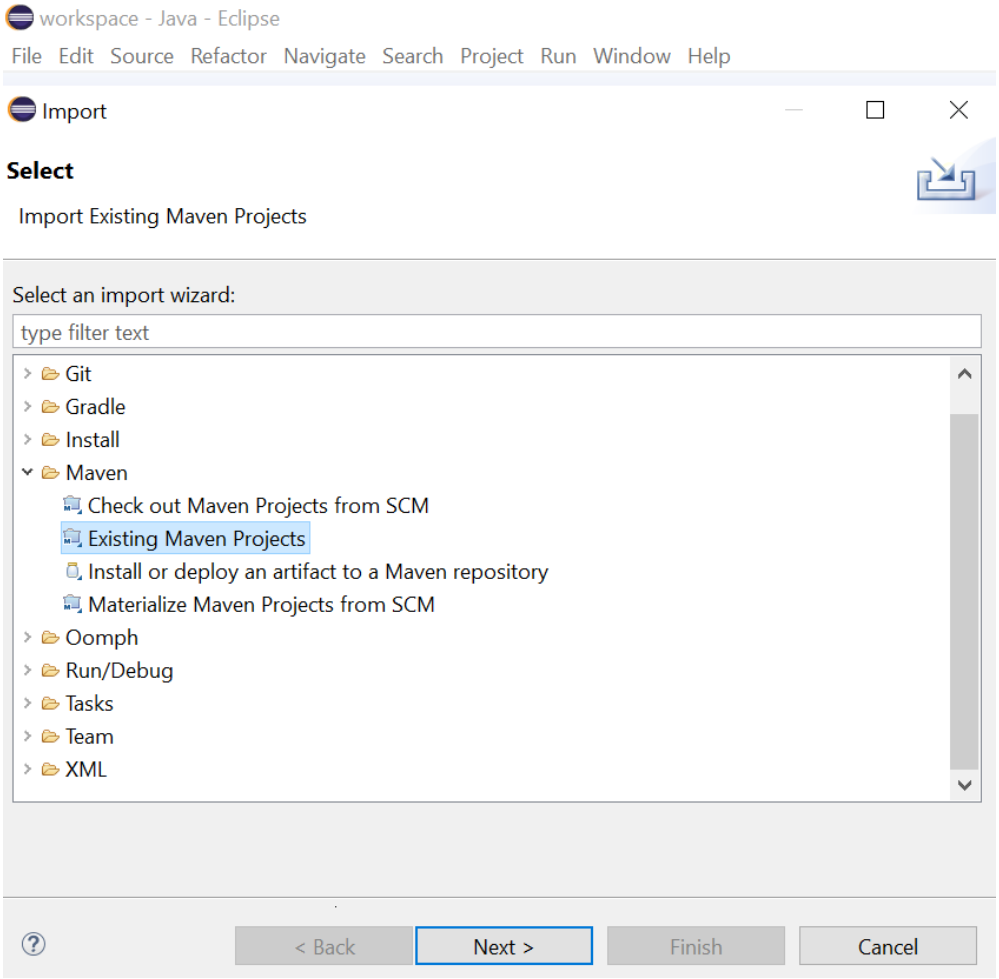

Figura 40 instalación 02

Finalmente elegimos la ruta del fichero del back en nuestro equipo y le damos a importar

Instalación FRONTEND:

El primer paso es crear un nuevo proyecto tipo web: Dynamic Web Project, Posteriormente darle un nombre al proyecto y elegir la ruta del código del Frontend como se muestra en la *figura 41*.

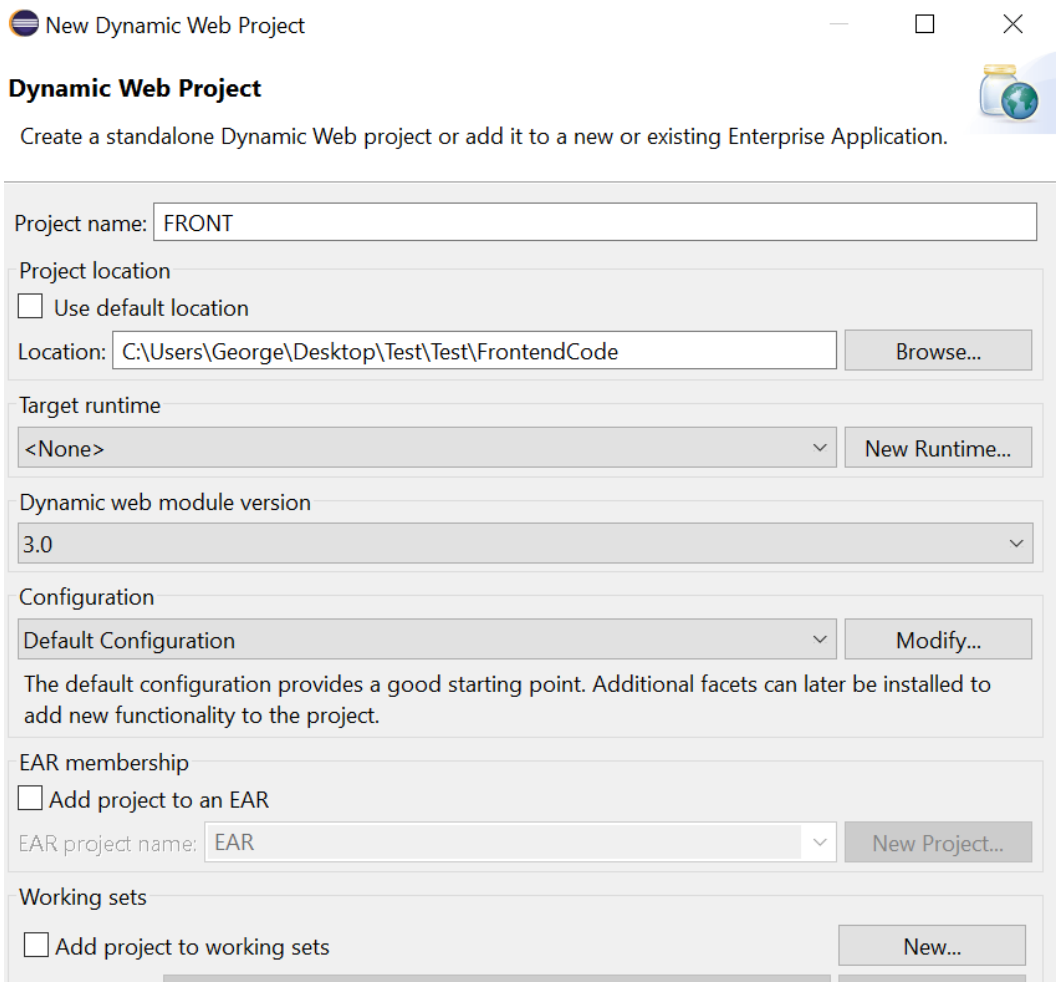

Figura 41 instalación 03

Instalación Servers:

Una vez configuradas las dos partes del back y front toca configurar la parte del servidor Tomcat.

Antes se debe descargar el Apache Tomcat.

https://tomcat.apache.org/download-70.cgi

Después vamos a la parte de servers (*figura 42*).

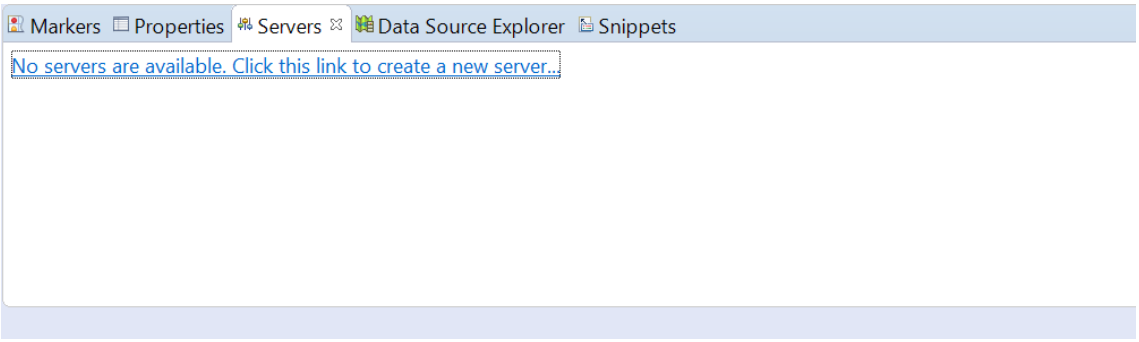

Figura 42 instalación 04

Y elegimos la versión del tomcat que nos hemos descargado como muestra la *figura 43*.

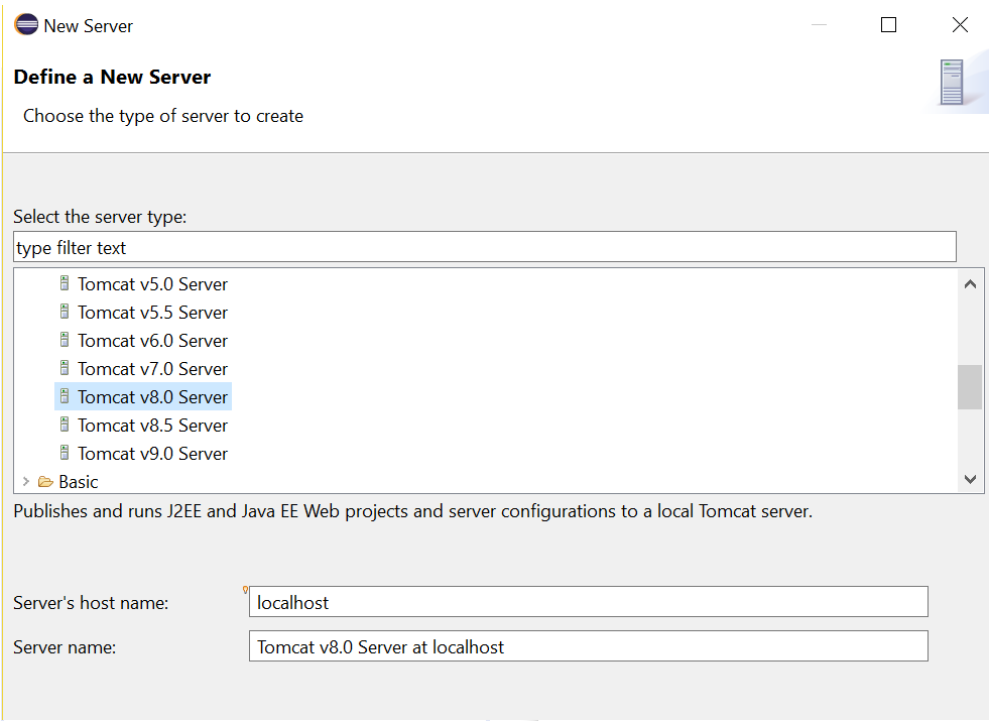

Figura 43 instalación 05

En esta configuración hay que elegir la ruta del tomcat que nos hemos descargado y posteriormente añadir los JARS del front y el back.

Una vez hecha toda la configuración anterior ya podemos ejecutar el código como muestra la *figura 44*:

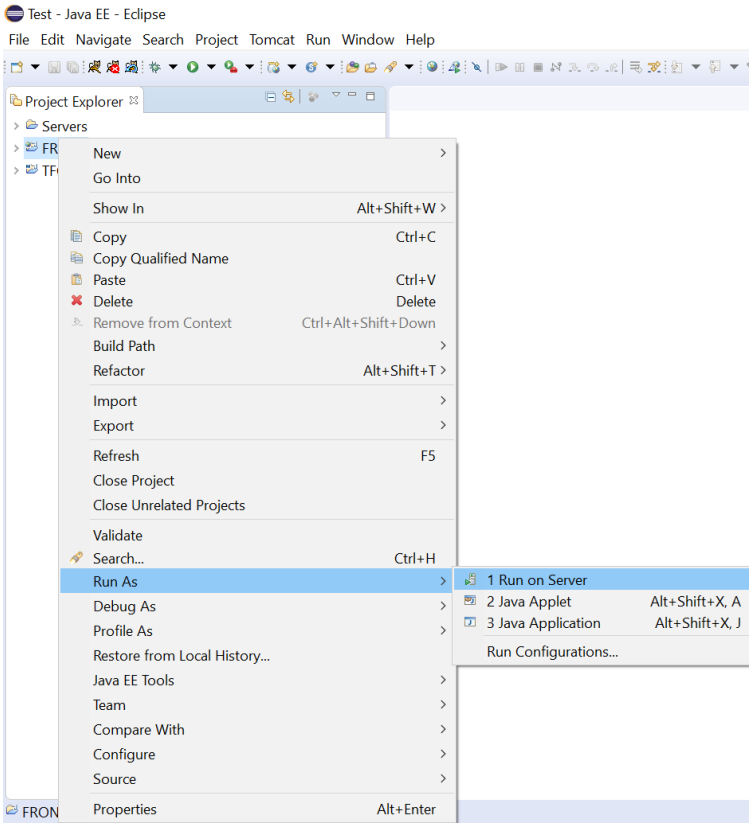

Figura 44 instalación 06

Este proceso arrancará un simulador de navegador web en Eclipse, pero se puede modificar para ejecutarlo sobre el navegador que prefiramos (Mozilla, Chrome...).

Para poder hacerlo simplemente tenemos que ir a Preferencias-> General -> Web Browser.

Instalación Base de Datos:

Para la instalación de la base de datos se debe descargar e instalar el XAMP.

https://www.apachefriends.org/es/index.html

Una vez instalado hay que lanzarlo para iniciar el apache y el MYSQL, después hay que abrir un navegador e ir al siguiente enlace localhost/phpmyadmin.

Una vez dentro importamos la base de datos proporcionada.

BIBLIOGRAFÍA

1. Planificador: *http://planning.domains*
2. Dietmar Jannach, Markus Zanker, Alexander Felfernigm Gerhard Friedrich. (2011). Recommender Systems
3. Arranque en frío: https://es.wikipedia.org/wiki/Arranque_en_fr%C3%ADo
4. Método Híbrido de Recomendación: *http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-50062016000400010*
5. Hugo Gutiérrez. (2017). Así será el turismo en España en 2017: más turistas, precios más altos y hoteles renovados. 2017/04/26, de El País Sitio web: *http://economia.elpais.com/economia/2017/04/26/actualidad/1493211122_897645.html*
6. Adrián Latorres. (2011). Descubre Minube, la mejor red social de viajeros, ahora ya para Android. 02/09/2011, de El androide libre. Sitio web: *https://elandroidelibre.elespanol.com/2011/09/descubre-minube-la-mejor-red-social-de-viajeros-ahora-ya-para-android.html*
7. Mi nube: *http://www.minube.com*
8. Wikipedia. TripAdvisor, de Wikipedia Sitio web: *https://es.wikipedia.org/wiki/TripAdvisor*
9. Guía Repsol: *https://www.guiarepsol.com/*
10. Oracle. (2015). Java EE Documentation, de Oracle Sitio web: *http://docs.oracle.com/javaee/7/index.html*
11. JSON: *https://jsonlint.com/*
12. Bootstrap: *https://raiolanetworks.es/blog/que-es-bootstrap/*
13. Bootstrap: *http://getbootstrap.com/*
14. jQuery API: *http://api.jquery.com/*
15. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. (2001). Introduction to Algorithms, Second Edition. MIT Press and McGraw-Hill. ISBN 0-262-03293-7. Section 22.3: Depth-first search, pp.531 - 539.

16. Lourdes Araujo, Carlos Cervigón. (2009). Algoritmos evolutivos: un enfoque práctico. RA-MA EDITORIAL
17. Lipowski, Roulette-wheel selection via stochastic acceptance (arXiv:1109.3627)
18. Bäck, Thomas. (1996). Evolutionary Algorithms in Theory and Practice, p. 120, Oxford Univ. Press
19. Algoritmo la Ruleta: *http://www.edc.ncl.ac.uk/highlight/rhjanuary2007g02.php*
20. Foro Stackoverflow: *https://stackoverflow.com/*
21. Malik Ghallab, Dana Nau, Paolo Traverso. (2016). Automated Planning and Acting. Cambridge: Cambridge University Press