Trabajo de Fin de Grado

Recomendador de rutas para actividades de turismo rural

Routes recommender for rural tourism activities

Mauricio José Orta Rodríguez

La Laguna, 4 de junio de 2017

D. **Dagoberto Castellanos Nieves**, con N.I.F. 79234766-L Profesor Contratado Doctor adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **José Andrés Moreno Pérez**, con N.I.F. 42935437-A Catedrático de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"*Recomendador de rutas para actividades de turismo rural*"

ha sido realizada bajo su dirección por D. **Mauricio José Orta Rodríguez**, con N.I.F. 42290682-F.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de junio de 2017

Agradecimientos

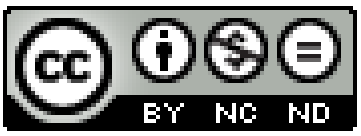
A mis padres, por apoyarme y siempre creer en mi.

A mis tutores: Dagoberto Castellanos Nieves y José Andrés Moreno Pérez, y al investigador de la universidad Airam Expósito Márquez. Gracias a su activa colaboración y aporte de propuestas oportunas durante el período en que desarrollé este Trabajo de Fin de Grado, el mismo pudo salir adelante.

A mis amigos y compañeros de la Universidad, por los buenos momentos compartidos y por darme una mano más de una vez para avanzar en el grado hasta el final.

A los programadores de la empresa StartDevs: José Isaac Hernández e Iradiel García, quienes generosamente contribuyeron con sus conocimientos, herramientas y experiencia con aplicaciones web para ayudarme con la implementación de la resultante de este Trabajo de Fin Grado.

Licencia



Resumen

El objetivo de este trabajo ha sido documentar los aspectos más significativos del diseño, desarrollo y despliegue de un sistema recomendador de turismo para la generación de itinerarios de rutas de actividades de turismo rural.

Para ello, fueron esenciales los conocimientos y la experiencia en programación adquirida a lo largo de la carrera, en especial la aportada por las asignaturas de "Modelado de sistemas de software", "Análisis de sistemas de Software", "Diseño arquitectónico y patrones" y "Laboratorio de desarrollo y herramientas", las cuales el autor cursó durante el tercer y cuarto curso del itinerario de Ingeniería del Software del Grado en Ingeniería Informática de la Universidad de La Laguna.

El sistema resultante es un sistema recomendador de turismo que consta de una aplicación web y, a través de la interacción con un servidor que calcula y optimiza rutas georreferenciadas, obtiene y dibuja en un mapa interactivo itinerarios de rutas de senderismo en la isla de Tenerife por días de estadía, en las cuales se destacan las rutas a visitar en cada uno y su información relevante. El usuario puede filtrar las características de las rutas a mostrar a través de un formulario. La aplicación web fue desplegada y se encuentra disponible en línea en un servidor de la plataforma IaaS de la Universidad de La Laguna, para que de esta forma sirva como referencia para trabajos futuros.

Palabras clave: Sistema recomendador, Vehicle Routing Problem, Tourist Trip Design Problem, Turismo, Tenerife

Abstract

The objective of this work was to compile the most significant aspects of the design, development and deployment of a tourism recommender system for the creation of route itineraries of rural tourism activities.

The programming knowledge and experience acquired during the university course were essential to achieve this goal, especially those from the subjects of "Modelado de Sistemas de Software", "Análisis de Sistemas de Software", "Diseño Arquitéctonico y Patrones" and "Laboratorio de Desarrollo y Herramientas", which the author attended during the third and fourth year of the "Ingeniería del Software" itinerary of the "Grado en Ingeniería Informática" of the University of La Laguna.

The resulting system is a recommender system with a web application that, through interaction with a server which calculates and optimizes georeferenced routes, gets and draws in an interactive map itineraries of hiking trails on the island of Tenerife for days of stay, in which the the hiking trails to be visited in each one of them and their information are highlighted. The user can filter the characteristics of the hiking trails to be displayed through a form. The web application was deployed and is available online on a server of the IaaS platform of the University of La Laguna, in order to serve as a reference for future works.

Keywords: *Recommender system, Vehicle Routing Problem, Tourist Trip Design Problem, Tourism, Tenerife*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Iniciativa

Las naciones están más conectadas que nunca en nuestra historia; esta es una de las mayores consecuencias del mundo globalizado. El desarrollo de maravillas tecnológicas como los ordenadores personales, la Internet, la World Wide Web, los smartphones y otros dispositivos portátiles avanzados en en las últimas décadas han mejorado la comunicación e interacciones humanas en formas jamás imaginadas anteriormente; permitiendo a personas de todo el mundo intercambiar información de manera casi instantánea sin importar qué tan grandes son las distancias entre ellas. En la actualidad, la web es una gran piscina de datos libremente disponibles de clientes potenciales (especialmente en redes sociales tales como Facebook, Twitter, etc.). Mediante el procesamiento y correcta transformación de esos datos en información útil, las organizaciones y empresas de diferentes áreas económicas pueden optimizar sus productos y servicios para satisfacer mejor las necesidades de sus clientes o expandirse a mercados previamente ignorados. En esta era, la información realmente es poder, y esto es aún más evidente en la industria del turismo.

Con un crecimiento continuo y profunda diversificación, según los últimos datos publicados por UNWTO, la organización de las Naciones Unidas para el turismo mundial [1], el turismo representa más del 10 % de la actividad económica mundial superando el billón de euros. Según esta misma fuente, el crecimiento del turismo internacional se sitúa ya por encima del 5 % representando el 7 % de las exportaciones mundiales; el 30 % del sector servicios. También según esta fuente, en el año 2015 se superaron los 6.000 millones de turistas nacionales y los 1.186 millones de turistas internacionales (se pronostican unos 1.800 millones anuales para 2030). El turismo aporta 1 de cada 11 puestos de trabajo en todo el mundo, alcanzando el 12 % de los puestos de trabajo en España y hasta el 40 % en Canarias. En España se reciben anualmente más de 70 millones de turistas anuales que aportan unos ingresos de 60.000 millones de euros (alre-

dedor del 10 % de la economía nacional) [2]. En Canarias se han superado en el pasado 2016 los 13 millones de turistas y la actividad económica relacionada con el turismo se sitúa por encima del 70 % de la de toda Canarias [3].

Es bien sabido que el turismo se puede dividir en distintas tendencias, y una de las que ha cobrado más fuerza en los últimos años es la del "senderismo" y turismo de montaña. Uno de los atractivos turísticos más importantes de las Islas Canarias, es la diversidad y belleza de los espacios rurales y de montaña para la práctica de actividades como el senderismo, las rutas en bicicleta o de otros tipos de vehículos adaptados a estos entornos. Los turistas, cuando visitan alguna o varias de las islas, frecuentemente buscan parajes de interés y actividades relacionadas con las que pasar un tiempo de vacaciones en el que puedan disfrutar de los parajes naturales, a la vez que obtienen beneficios saludables ajustados a sus características personales y preferencias. Para aprovechar este potencial es necesario poner a disposición del turista las distintas alternativas de forma organizada y proactiva. Aparte de facilitar de forma accesible toda la información relacionada, proponer o recomendar las actividades que mejor se ajusten a sus características particulares y su planificación temporal y espacial en los días disponibles de su visita.

La competitividad de las pequeñas empresas asociadas al turismo rural y de montaña en un mercado global mejoraría aprovechando los recursos tecnológicos disponibles para la gestión y aprovechamiento de la información. Existen diversas alternativas en este contexto que apuntan hacia esta línea de compartir experiencias e información: los sistemas de recomendación (SR).

Los SR en el ámbito del turismo ofrecen al turista que realiza un viaje a un destino encontrar los recursos adecuados a sus características y preferencias ofreciéndole una relación de puntos de interés filtrados y ordenados que se le proponen.

La propuesta de sistema recomendador a desarrollar en este proyecto pretende ofrecer itinerarios personalizados que incluyan rutas para actividades asociadas al mundo rural y de montaña, y que optimicen los itinerarios con criterios de eficiencia y máxima satisfacción de las preferencias del turista. El producto será una aplicación cliente-servidor, cuyo front-end o parte cliente será una aplicación web desarrollada en HTML, CSS y distintos frameworks basados en Javascript en los que se profundizará posteriormente, y un back-end o parte servidor que se apoya en un servidor Tomcat que ejecuta un código Java relativo a un algoritmo basado en el Tourist Trip Design Problem o TTDP [4] (leer Antecedentes y estado del arte). Dicho algoritmo usa en esta implementación una heurística "greedy randomized adaptive search procedure" (GRASP) [5] para realizar sus cálculos, y retorna la mejor solución encontrada como un fichero JSON en una dirección web que la aplicación cliente o front-end puede descargar para dibujar las rutas del itinerario a partir de sus datos.

1.2. Antecedentes y estado del arte

El modelo de problema que resuelven estos sistemas ha recibido el nombre de Tourist Trip Design Problem (TTDP). Las primeras investigaciones sobre el TTDP aparecen a principios del siglo actual por los trabajos del profesor Godart en Bélgica [6–11] que dieron lugar a la aplicación web YourTour (`http://www.yourtour.com/`) que aún continua en funcionamiento. Posteriormente comenzaron los trabajos liderados por P. Vansteenwegen a partir de su tesis doctoral en 2008 con publicaciones entre 2008 y 2013 [12–19] que dieron lugar a la aplicación "CityTripPlanner" (`http://www.citytripplanner.com`). Del 2011 al 2014 se desarrolló el proyecto europeo eCOMPASS (FP7-ITC) dentro del programa "Smart Cities & Sustainability" para el desarrollo e implementación de un planificador de rutas multimodales urbanas para usuarios de móviles que dio lugar a la app DAILYtrip y a un buen número de publicaciones en esta área lideradas por D. Gavalas de 2009 hasta 2014 [20–23].

Más recientemente se han publicados nuevos enfoques contemplando los grupos de turistas [24] y el descubrimiento de sitios (serendipity) [25–27]. En Internet se pueden encontrar varios planificadores de rutas turísticas urbanas, como VisitaCity [28] y City Trip Traveller [29] [este está desarrollado como resultado de un TFG]. Asimismo, existen aplicaciones móviles para la construcción por GPS y sugerencia de rutas turísticas generales como OruxMaps [30], disponible en Android, y WikiLoc [31], disponible tanto para dispositivos Android como iPhone.

Por otro lado, se han desarrollado varios servicios más especializados en la sugerencia de rutas de senderismo y turismo de montaña, entre las que se pueden destacar mapmyhike [32], AllTrails [33] y Ramblr [34], las tres con aplicaciones para Android y iPhone. Estas herramientas permiten a los usuarios elegir entre un amplio rango de rutas recomendadas por la web en distintos destinos, mantener un seguimiento de su actividad en ellas, valorar su experiencia final, e incluso crear y compartir sus propias rutas al resto de los usuarios de la comunidad.

1.3. Objetivos

Los objetivos concretos que persigue este proyecto son los siguientes:

- (OB1). Estudio del estado del arte de los sistemas de planificación de rutas con recomendaciones, y de los sistemas de información sobre actividades de montaña y del mundo rural.
- (OB2). Definición y diseño de modelos y herramientas de planificación óptima de actividades rurales y de montaña y de recomendación de itine-

rarios ajustados a las necesidades y preferencias de los turistas.

- (OB3). Validación de modelos y métodos, empleando pruebas y datos de recursos rurales y de montaña.
- (OB4). Obtención de un prototipo software como servicio web integrado en una plataforma georreferenciada de gestión.
- (OB5). Creación de documentación técnica, entregables y memoria final del proyecto.

Capítulo 2

Metodologías y modelos algorítmicos para el diseño de rutas turísticas

En este capítulo se explican los modelos teóricos y algorítmicos empleados para el diseño de las rutas y el desarrollo del sistema recomendador.

2.1. Diseño de rutas turísticas

Los problemas de diseño de rutas turísticas (Tourist Trip Design Problems, TTDP) consisten seleccionar los puntos de interés (PDI) a visitar por un turista atendiendo a sus restricciones y al beneficio o grado de satisfacción que produce su visita. El turista dispone en su estancia de un cierto número de días para organizar las visitas a los puntos de interés mediante rutas de duración limitada. Se parte de un conjunto de puntos posibles a visitar de los que se conoce, el beneficio o grado de satisfacción, la duración de la visita y el intervalo de tiempo en el que puede realizarse. El beneficio total que intenta maximizar el turista es la suma de los beneficios obtenidos en cada visita.

Los elementos que forman parte del modelo son:

- Un conjunto de Puntos de Interés (PDI, Points of Interests o POI en inglés), asociado a un índice i, $i = 1, 2, \ldots, n$ y con los siguientes atributos:
  - Una puntuación o beneficio s_i.
  - Un tiempo de duración de la visita r_i.
  - Un intervalo de tiempo $[e_i,\ l_i]$ dentro del que se puede realizar la visita.
- Un punto de partida y de llegada de cada una de las rutas denotado por

$i = 0$.

- Los tiempos de recorrido entre los pares de puntos t_{ij}, $i = 0, 1, \ldots, n$.
- Un tiempo máximo T^k_{max} de duración total de la ruta del k-ésimo día considerando los tiempos de viaje, visita y espera en los PDI; $k = 1, \ldots, m$.
- Una función objetivo

$$max \sum_{k=1}^{m} \sum_{i=1}^{n} s_i y_i^k$$

En la que y_i^k corresponde a una variable de visita para cada uno de los Puntos de Interés i, $i = 1, \ldots, n$, y cada ruta k, $k = 1, \ldots, m$. Dicha variable es binaria, teniendo como valor un 1 si el Punto de Interés i es visitado en la ruta k y un 0 de no ser así.

El problema de optimización resultante coincide con el Team Orienteering Problem with Time Windows (TOPTW) [35] [36] que se ha estudiado en la literatura científica con algunas modificaciones. Se consideran como puntos de interés los senderos de la isla de Tenerife que están señalizados por las autoridades correspondientes; en este caso el Cabildo de Tenerife.

2.2. Heurística de diseño

Para dar soluciones de alta calidad al problema de optimización planteado se selecciona la metaheurística constructiva GRASP (Greedy Randomized Adaptive Search Procedure) que comprende dos fases: fase constructiva y búsqueda local (Local Search). En la fase constructiva se genera una solución partiendo de m rutas vacías añadiendo sucesivamente un PDI desde una Lista Restringida de Candidatos (Restricted Candidate List, RCL) a alguna de las rutas hasta que no sea posible añadir nuevos puntos. En la fase de búsqueda local se reemplaza la solución por la mejor de sus soluciones vecinas mientras exista mejora. Estas fases se ejecutan sucesivamente durante un cierto número de iteraciones.

En la siguiente figura (2.1), se puede apreciar un pseudocódigo (en Inglés) del algoritmo de la metaheurística:

Function GRASP(maxIterations,SizeRCL)
1. readInput();
2. for $k = 1$, ..., maxIterations do:
   a) solution = GRASPConstructPhase(sizeRCL);
   b) solution = localSearch(solution);
   c) update(solution,bestSolution);
3. end;
4. return bestSolution

End GRASP

Figura 2.1: Pseudocódigo de la metaheurística GRASP

En esta implementación para el TOPTW, la función de construcción (ver figura 2.2) primero crea un conjunto de Solución con m rutas vacías, las cuales corresponden a las que se desean generar, y se evalúa el coste incremental de todos los puntos de interés a partir de sus tiempos con una función greedy. Entonces, y mientras la solución no sea completa y se puedan seguir visitando puntos, se recorren todos los j puntos de interés, encontrando la mejor posición k en la cual añadir cada punto para una ruta parcial R de acuerdo a su coste incremental evaluado. Dicha información se añade a una lista de candidatos CL. Al terminar de recorrer los puntos, se crea una lista restringida de candidatos RCL con los sizeRCL mejores de la original. Finalmente, se elige de manera aleatoria alguno de esos mejores puntos y el conjunto de Solución es actualizado con el mismo. Dicho proceso se repite hasta que se obtiene una solución completa y factible para el problema.

function GRASPConstructPhase(sizeRCL)
1. Initialization of m empty routes (they leave and arrive to the depot)
2. While it is possible to visit POIs
   a) For each $poi_j \in POI$
      1) Find the best position k to insert poi_j in a partial route R according to greedy min. time function $f(poi_j)$
      2) Add the group of triplet (poi_j,i,R) to the Candidate List CL
   b) Create the Restricted Candidate List, RCL, with the top $sizeRCL$ triplets (poi_j,i,R) from CL
   c) Select a random group of $sizeRCL$ (poi_j,i,R) from RCL
   d) Update the route R by inserting the POI poi_j at position i

end GRASPConstructPhase;

Figura 2.2: Pseudocódigo de fase de construcción de algoritmo GRASP para problema TOPTW

La solución factible obtenida en la fase de construcción no siempre es la óptima, por lo que se recurre a la fase de búsqueda local (ver figura 2.3) para

intentar mejorarla. Dicha búsqueda trabaja en forma iterativa reemplazando sucesivamente la solución actual *solution* por una mejor solución x dentro de su conjunto de "vecinos". Finaliza cuando ya no se encuentran otras soluciones mejores dentro del conjunto de vecinos

---

function localSearch(solution)

1. Repeat
   - *a*) Find the best solution x neighbor of *solution*
   - *b*) If $f(x) \leq f(solution)$ do $solution = x$
2. until $f(x) > f(solution)$ for all neighborhood
3. return *solution*;

end localSearch;

Figura 2.3: Pseudocódigo de fase de búsqueda local para algoritmo GRASP

Capítulo 3

Análisis de requisitos de Recomendador de rutas para actividades de turismo rural

En este capítulo se muestran los requisitos funcionales y no funcionales estipulados para el sistema recomendador desarrollado en este Trabajo de Fin de Grado. También se muestran sus diagramas de casos de uso y las fuentes usadas para los puntos de interés y alojamientos de las rutas a recomendar.

3.1. Requisitos

3.1.1. Aplicación back-end/servidor

- Requisitos funcionales
  - **Recepción de peticiones de cliente:** el servidor será capaz de recibir peticiones que el cliente le envíe a través de consultas con parámetros específicos en un fichero .json a sus direcciones web.
  - **Cálculo y retorno de rutas óptimas:** a partir de los datos de las peticiones de la aplicación cliente, el servidor calculará y retornará a través de una dirección web la información de las rutas óptimas generadas en un formato .json que la aplicación cliente pueda descargar e interpretar para dibujarlas.
  - **Publicación de lista de alojamientos:** el servidor tendrá acceso a una lista de alojamientos que el usuario puede elegir como punto

de partida desde la aplicación cliente, y la publicará en una dirección web bajo formato .json para su acceso.

- Requisitos no funcionales
  - **Acceso a base de datos de puntos de interés y alojamientos:** el servidor se ha de conectar a una base de datos con puntos de interés (destinos y rutas de senderismo) y alojamientos de Tenerife, obteniendo su información para usarla en el cumplimiento de sus requisitos funcionales.
  - **Envío de peticiones a servidor de cálculo de matrices de distancia:** con los puntos de interés obtenidos de la base de datos, el servidor enviará peticiones con su lista a otro servidor secundario que calculará la matriz de distancia existente entre ellos.
  - **Importación de matriz de distancia:** tras calcular la matriz de distancia entre los puntos de interés, el servidor secundario respectivo la retornará como un fichero .json que el servidor importará y guardará para su uso en el cálculo de rutas óptimas.

3.1.2. Aplicación front-end/cliente

- Requisitos funcionales
  - **Importación de conjuntos de datos del servidor:** la aplicación cliente deberá ser capaz de acceder a los datos de las listas de alojamientos y a las rutas óptimas que el servidor proporciona bajo ficheros .json en direcciones web.
  - **Mapa de Tenerife:** la aplicación deberá mostrar un mapa geográfico y de carreteras de Tenerife, mediante un mapa de Open Street Maps.
  - **Formulario para generación de rutas:** deberá existir un formulario que el usuario deberá rellenar con los siguientes campos relativos a :
    - Datos de alojamiento
      - Alojamiento.
      - Días de estadía.
    - Datos de rutas/destinos de senderismo
      - Distancia máxima (en metros).
      - Altura máxima (en metros).
      - Altura mínima (en metros).

- ◇ Desnivel acumulado ascendente (en metros).
- ◇ Desnivel acumulado descendente (en metros).
- ◇ Dificultad de rutas (fácil, moderada o difícil).
- ◇ Tipo de rutas (lineal o circular).
- ◇ Tipo de transporte usado para moverse al destino (coche propio o transporte público).

La información de dichos campos se parametrizará en una petición al servidor que permitirá a este último calcular un itinerario filtrado de destinos, el cuál será usado por la aplicación cliente para el dibujado de las rutas.

- **Dibujado de rutas:** tras mandar la petición de sugerencia de rutas al servidor e importar su respuesta, la aplicación cliente extraerá los datos de la misma para dibujar las rutas, señalando sus puntos de interés con marcadores y los caminos para llegar a ellos con colores.
- **Información de marcadores:** cada uno de los marcadores de los puntos de interés de las rutas deberán mostrar información alusiva a los mismos al darles click, y así mismo datos relativos a su situación dentro de su ruta sugerida (tiempo de llegada, tiempo de salida, ruta/día del itinerario a la que pertenece).

- Requisitos no funcionales
  - **Plataformas:** será una aplicación web, compatible con todos los navegadores más populares del mercado.
  - **Interfaz:** la interfaz deberá ser atractiva, siendo lo más ligera (en términos de carga) e intuitiva posible.

3.2. Diagramas de casos de uso

- Servidor

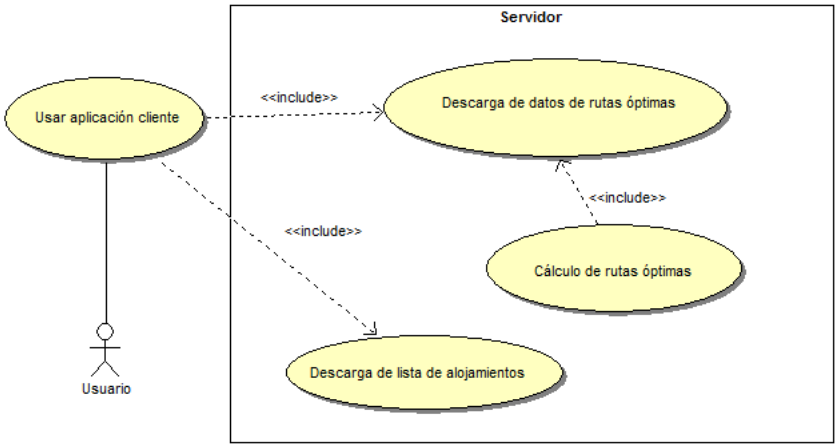


Figura 3.1: Casos de uso del servidor

- Aplicación cliente

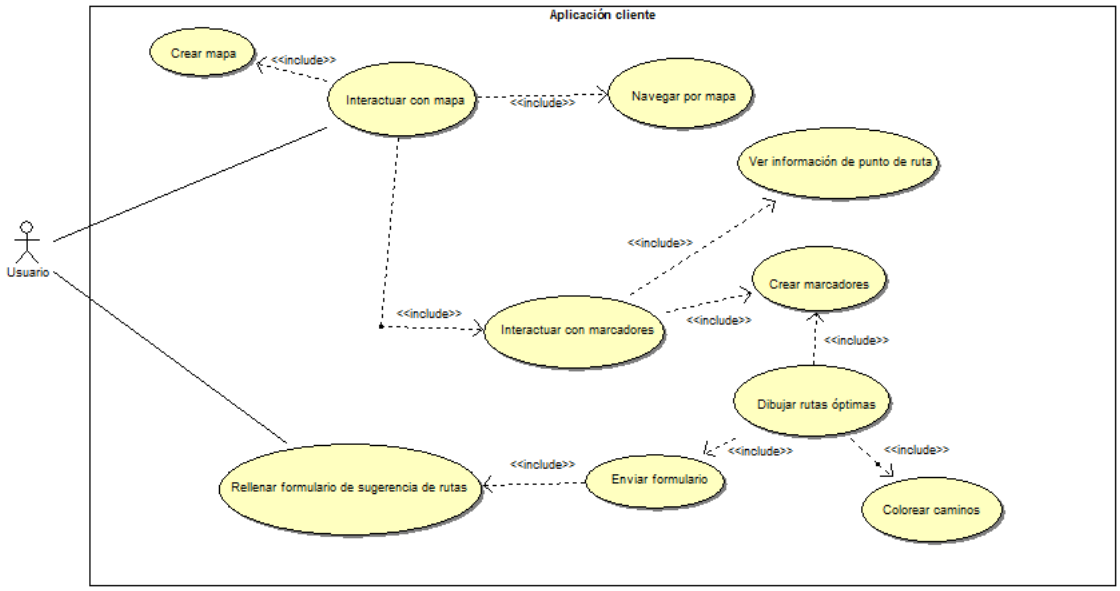


Figura 3.2: Casos de uso de aplicación cliente

3.3. Fuente de los datos

3.3.1. Rutas de senderismo

Debido a que el objetivo de la aplicación en cuestión a desarrollar fue el de "ofrecer itinerarios personalizados que incluyan rutas para actividades asocia-

das al mundo rural y de montaña", se requirió obtener una lista de puntos de interés que cumplan esos criterios y que además dispongan de información lo suficientemente completa para identificarlos con claridad, y filtrarlos en función de distintos criterios o preferencias que puedan tener los usuarios.

En el curso de esta investigación, la fuente elegida de estos puntos de interés resultó ser Webtenerife [37], la web de promoción turística oficial de la provincia de Santa Cruz de Tenerife gestionada por la empresa Turismo en Tenerife.

Figura 3.3: Logo de Webtenerife

Su sitio web cuenta con una lista bastante extensa y bien documentada de rutas de senderismo a lo largo de toda la isla. Dicha documentación incluye parámetros mencionados en los requisitos funcionales como la distancia de recorrido, las alturas, la dificultad, el tipo, entre otros.

Al no estar disponible en ficheros de formato de datos abiertos, se extrajo y procesó manualmente la información de una muestra de 30 rutas de distintos tipos y dificultades presentes en la página web.

3.3.2. Alojamientos

Como también la aplicación permitiría al usuario elegir un sitio de alojamiento como punto de partida y destino para sus itinerarios, se buscó una lista de datos de hoteles y otros tipos de alojamientos existentes en la isla.

A diferencia del caso anterior, si se consiguieron publicados en formato de datos abiertos en el sitio web de Open Data Canarias.

Figura 3.4: Logo de Open Data Canarias

El formato de datos abiertos en el que estuvieron guardados dichos datos es JSON [38]. En la siguiente imagen se muestra la estructura de sus objetos.

```
{
	"help": "Listado de alojamientos turísticos de Tenerife",
	"listado": [
		{
			"ows_LinkTitle": "Aguere",
			"ows_Zona": "Metropolitana",
```

```
"ows_Georeferencia": "(28.48967677234651,
   -16.318953036825405)",
"ows_CodigoPostal": "38201",
"ows_Direccion": "C/ Obispo Rey Redondo, 55",
"ows_Telefono": "922 31 40 36",
"ows_Fax": "922 63 16 33",
"ows_Web": "http://www.hotelaguere.es",
"ows_Mascotas": "",
"ows_Accesibilidad": "",
"ows_FechaActualizacion": "2015-04-11 16:04:42",
"ows_Horarios": "",
"ows_TipoActividad": "Hotel",
"ows_Actividad": "Alojamientos",
"ows_Activo": "1",
"ows_Email": "reservas@hotelaguere.es",
"ows_Municipio": "San Cristóbal de la Laguna",
"ows_Categorias": "Bar,Cafetería,Internet,Internet
   habitación,Jardín-Terraza,Peluquería,Recepción,Sala
   conferencias,Salón de convenciones,Teléfono en
   habitación,Televisión,Vigilante,",
"ows_DistanciaPlaya": "",
"ows_DistanciaCentroComercial": "",
"ows_DistanciaPuerto": "",
"ows_DistanciaParadaBus": "",
"ows_DistanciaTFS": "",
"ows_DistanciaTFN": "",
"ows_Suites": "1",
"ows_Piscinas": "",
"ows_Bares": "",
"ows_Restaurantes": "",
"ows_Habitaciones": "23",
"ows_SubTipo": "1 estrella",
"ows_EmpresaNombre": "",
"ows_DescripcionIngles": "",
"ows_DescripcionAleman": "",
"ows_DescripcionRuso": "",
"ows_DescripcionItaliano": "",
"ows_ComoLlegarEspanol": "A 10 minutos del Aeropuerto
   Norte TFN a menos de 15 km de Santa Cruz ",
"ows_ComoLlegarIngles": "",
"ows_ComoLlegarAleman": "",
```

```
                "ows_ComoLlegarFrances": "",
                "ows_ComoLlegarItaliano": "",
                "ows_HorariosIngles": "",
                "ows_HorariosAleman": "",
                "ows_HorariosFrances": "",
                "ows_HorariosItaliano": ""
        },
        ...
        ]
}
```


Figura 3.5: Fichero JSON de alojamientos de Open Data Canarias

Debido a que la lista del fichero era muy extensa (con información de más 400 alojamientos en la isla), se tomó como muestra un número de 15 alojamientos para su futuro uso en las pruebas de la aplicación.

Capítulo 4

Diseño de aplicación de Recomendador de rutas para actividades de turismo rural

En este capítulo se detalla la arquitectura del sistema recomendador.

4.1. Visión general

El modelo de software elegido para el sistema recomendador es el de "cliente-servidor", el cual lo divide en una aplicación cliente y un servidor que interactúan entre si a través de una conexión de red [39].

La aplicación cliente es la que utiliza el usuario final y le brinda a este una interfaz con la que ingresar datos y enviar solicitudes a un servidor, que realiza tareas a partir de la información de sus solicitudes y le devuelve resultados al cliente para su aprovechamiento. Asimismo, este servidor interactúa con otros servidores secundarios a los que les pide información procesada.

En el siguiente diagrama de despliegue se muestra cómo sería tal estructura para este proyecto en particular:

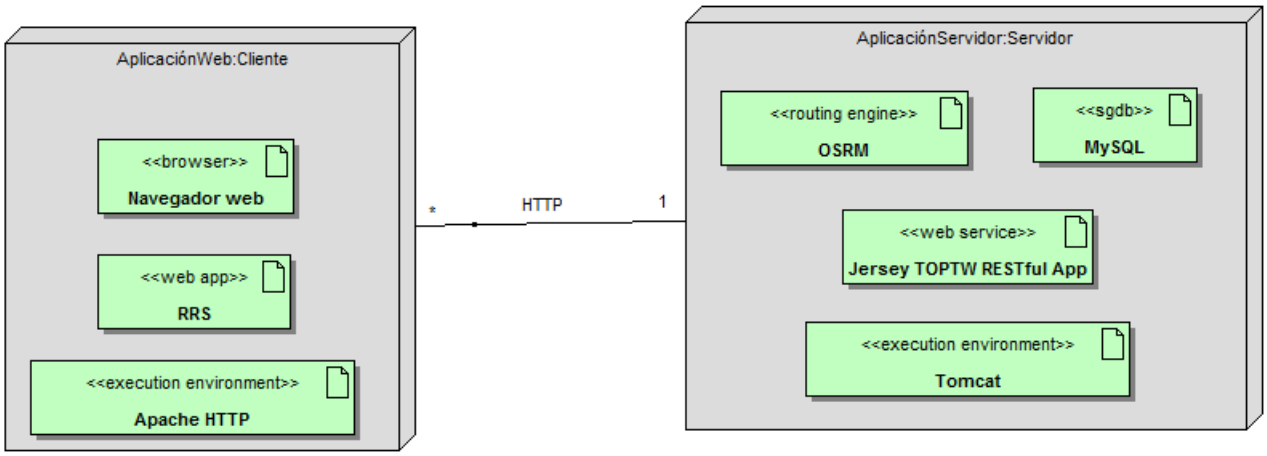


Figura 4.1: Diagrama de despliegue del sistema recomendador

La aplicación cliente y sus librerías, bajo el nombre de Recomendador de Rutas de Senderismo (RRS), es arrancada por un servidor Apache y se abre por medio de un navegador web.

Esta aplicación web se conecta a un servidor externo a la hora de generar los itinerarios de rutas de senderismo y le envía peticiones para que este las calcule y las resuelva como un problema de Team Orienteering with Time Windows y le retorne la información necesaria para representarlas en su interfaz visual. Este servidor es un servicio web RESTful escrito en Java con la ayuda del framework Jersey, el cual es arrancado por un servidor Tomcat. Dicho web service se conecta y pide información a una base de datos MySQL con los datos de los alojamientos y destinos de senderismo, y a un motor de cálculo de rutas georreferenciadas OSRM que determina y proporciona la matriz de distancias (en este caso, representadas como tiempos de viaje) reales existentes entre tales puntos de interés.

A continuación, se profundiza en las arquitecturas de tanto el código del servidor como de la de la aplicación cliente.

4.2. Servidor

Como se mencionó previamente, el servidor es un servicio web basado en un modelo RESTful (en español, Transferencia de Estado Representacional) [40], consistente en la interacción con el cliente con transferencias de información con métodos HTTP (como GET o POST) en direcciones claras e intuitivas, bajo formatos de datos web estandarizados y versátiles como XML y JSON.

Dicho modelo fue escrito en Java. Un lenguaje de programación orientado a clases y objetos que, a pesar de no ser propiamente web como HTML, CSS o

JavaScript, puede ser traducido a un servicio web gracias a frameworks como Jersey, el cual fue usado para lograr ese objetivo.

Las clases en Java creadas para el servidor se muestran en este diagrama:

4.2.1. Diagrama de clases

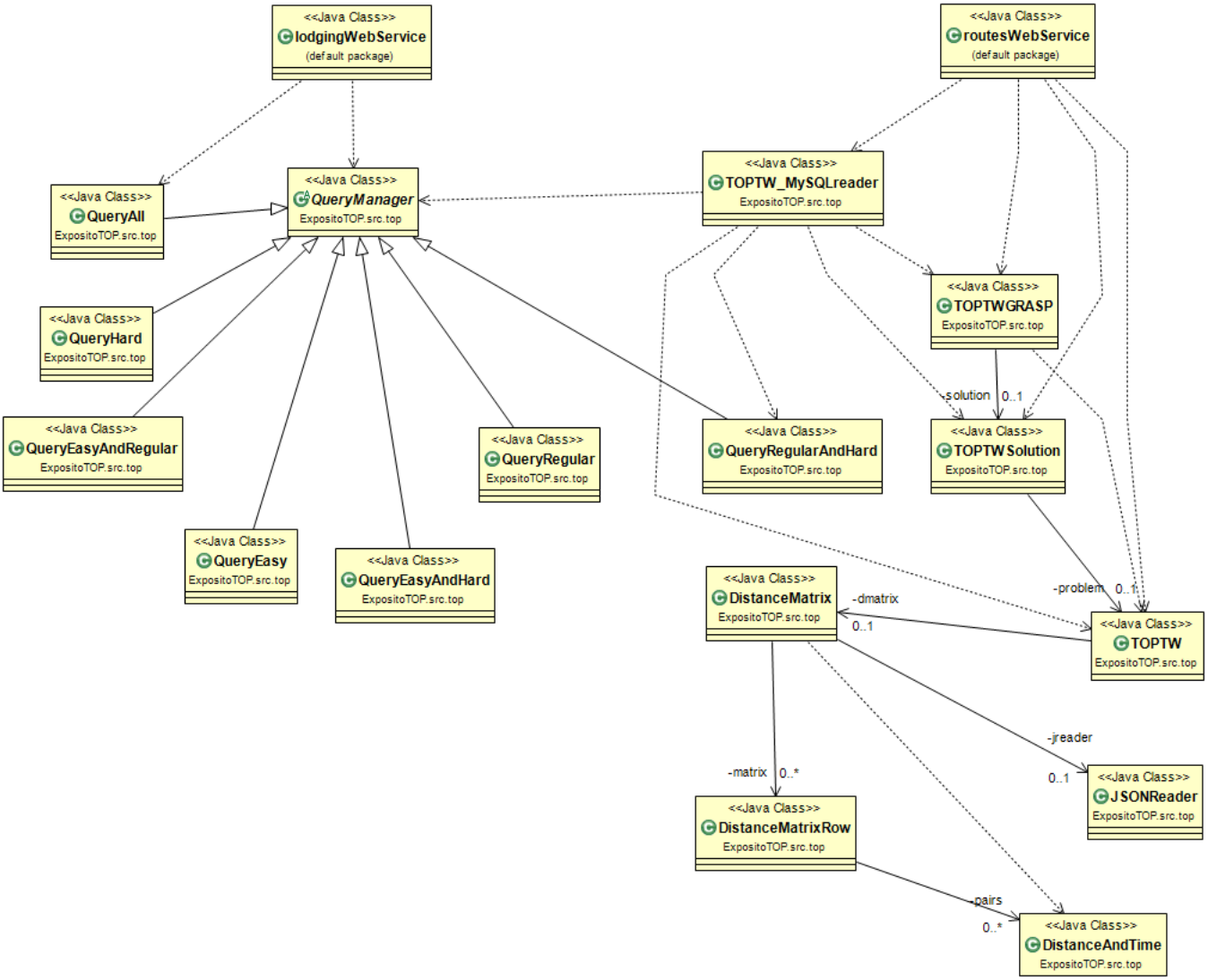


Figura 4.2: Diagrama de clases del servidor (dar click para ver con métodos y atributos)

En el diagrama se pueden apreciar dos grupos de clases principales: las clases correspondientes al problema TOPTW y su resolución y aquellas correspondientes al servicio web que se encarga de retornar los resultados de sus métodos en direcciones web del servidor.

- La clase TOPTW representa a un problema de Team Orienteering with Time Windows con vectores que almacenan la información de cada una de las rutas de senderismo que comprende; tales como el nombre de las

rutas, la su distancia, su altura máxima, su altura mínima, entre otros, así como los parámetros deseados para el cálculo de su solución, como el número máximo de rutas a generar a partir de sus puntos de interés, el tiempo límite para cada una de ellas y la "puntuación" de cada uno de los destinos. Entre los elementos más importantes de esta clase, se encuentra la matriz de distancia de los puntos del problema, que se representa por la clase DistanceMatrix. Clase que a su vez se compone de objetos DistanceMatrixRow, correspondientes a sus filas, que albergan a su vez un número determinado de pares DistanceAndTime alusivos a la distancia y el tiempo de viaje existente entre el punto de interés de la fila y el de una columna.

- JSONReader [41] no es más que una clase con métodos que permiten leer y extraer objetos JSON guardados en URLs.
- TOPTWSolution corresponde a una solución de un problema TOPTW dado y sus datos de interés: el número de rutas generadas, las listas de puntos de interés predecesores y sucesores, sus posiciones, los tiempos de espera de cada uno, su puntuación total.
- TOPTWGRASP se encarga de calcular la solución óptima de un problema TOPTW por medio de una implementación del algoritmo GRASP.
- La clase TOPTW_MySQLReader se encarga de acceder a la base de datos MySQL con la información de los puntos de interés de acuerdo a los parámetros especificados por el usuario y construye un problema TOTPW a partir de ellos.
- lodgingsWebService y routesWebService son los servicios web que se encargan de obtener y retornar en una dirección la información y los resultados de los métodos clave de las clases previas. routesWebService retorna una lista de todos los alojamientos disponibles en la base de datos, necesaria por la aplicación cliente para crear una lista de selección de los mismos en su formulario, y routesWebService es aquel al que el cliente envía peticiones para el cálculo de itinerarios de rutas de senderismo con parámetros específicos, las cuales procesa, retornando una solución con su información necesaria para proyectarla en la aplicación cliente.

El código de los métodos principales de cada clase y sus estructuras de datos se detallan a fondo en el capítulo de Desarrollo.

4.2.2. Base de datos

Como se ha comentado anteriormente, el lado servidor de la aplicación cuenta con una base de datos relacional MySQL, la cual tiene la siguiente estructura (ver figura 4.3):

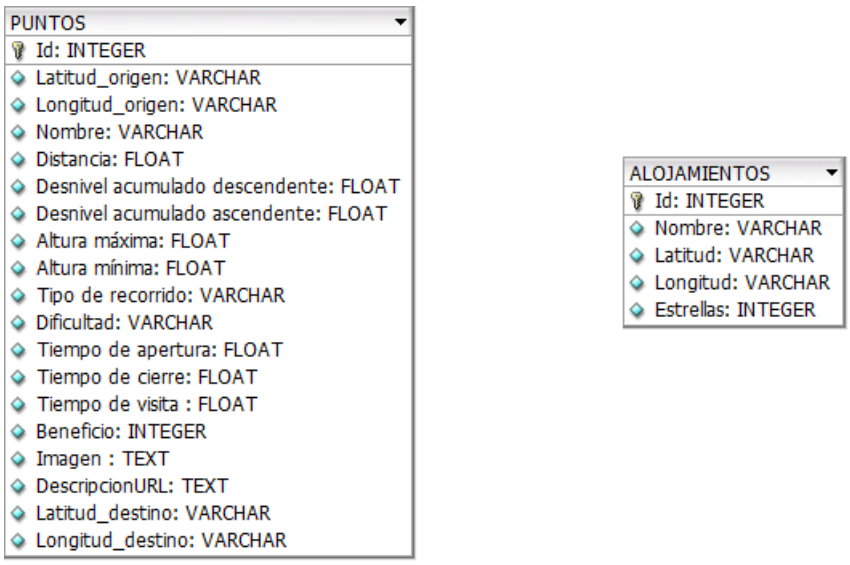


Figura 4.3: Tablas de base de datos

La tabla PUNTOS representa a los distintos puntos de interés a visitar y su información, es decir, las rutas de senderismo. ALOJAMIENTOS almacena la información básica de los hoteles, posadas y demás establecimientos de hospedaje que pueden elegirse como punto de partida y destino final para el itinerario.

Es importante señalar que en esta implementación, el "beneficio" de la tabla de puntos se refiere a la puntuación de cada punto de interés (en este caso, las rutas de senderismo) tomada en cuenta en el modelo TOPTW. Dichas puntuaciones se asignaron en función de la dificultad de las rutas de senderismo: 50 para las fáciles, 70 para las moderadas y 100 para las difíciles.

4.3. Aplicación cliente

Para la aplicación cliente, al ser una aplicación web, se escogió una arquitectura inspirada en el patrón de Modelo Vista Controlador o MVC [42], uno de los más usados y más prácticos para dicho tipo de aplicaciones.

El MVC divide a la aplicación en tres partes principales tal como dice su nombre: modelos, vistas y controladores. Los modelos corresponden a los datos usados por la aplicación y sus estructuras, las vistas a la interfaz visual de la aplicación y, por último, los controladores se encargan de transformar y gestionar los datos de esos modelos para ponerlos a la orden de las vistas.

4.3.1. Diagrama de componentes

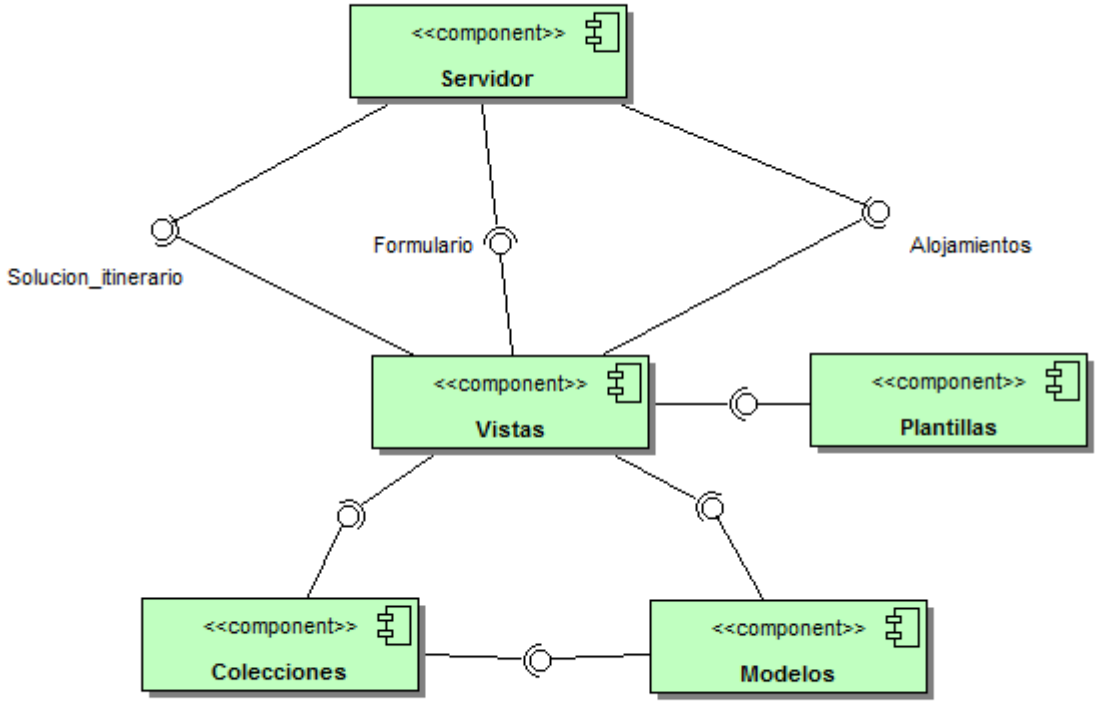


Figura 4.4: Diagrama de componentes de aplicación cliente

El diagrama muestra los cuatro grupos principales de componentes de la aplicación cliente y sus relaciones. En esta implementación del MVC se ha considerado una estructura adicional junto a los modelos llamada "colecciones", las cuales no son más que listas de objetos pertenecientes a un mismo modelo. Asimismo, los controladores son indistinguibles de las vistas, formando parte de las mismas, las cuales utilizan "plantillas" iniciales de interfaces visuales sobre las que hacen las representaciones de los datos.

Si bien no forma parte de la aplicación cliente, se ha destacado el servidor en el diagrama para mostrar su interacción con las vistas de la primera.

En las subsecciones a continuación, se detalla el contenido de cada uno de los cuatro grupos de la arquitectura.

4.3.2. Modelos

Figura 4.5: Modelos de datos de aplicación cliente

Para la aplicación solo se contemplaron dos modelos, uno para el mapa interactivo en el que se muestran las rutas y otro para los alojamientos seleccionables desde el formulario.

4.3.3. Colecciones

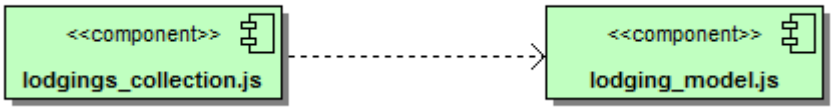


Figura 4.6: Colecciones de aplicación cliente

Solo existe una colección en esta implementación, que corresponde a una de alojamientos al ser el único modelo de datos con más de una instancia perteneciente al mismo.

4.3.4. Plantillas

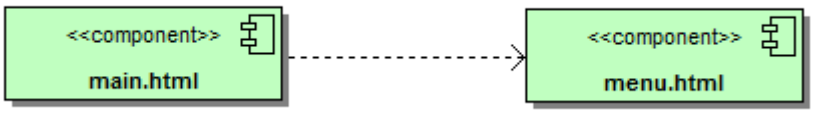


Figura 4.7: Plantillas de aplicación cliente

Se cuentan con dos plantillas HTML para las vistas: main y menu. Main corresponde a la interfaz principal de la aplicación sobre la que se proyectan la mayoría de sus componentes, y menu a la interfaz del formulario usado para la introducción de los datos necesarios en la generación de los itinerarios. menu se representa sobre la interfaz de main por medio de las vistas.

4.3.5. Vistas

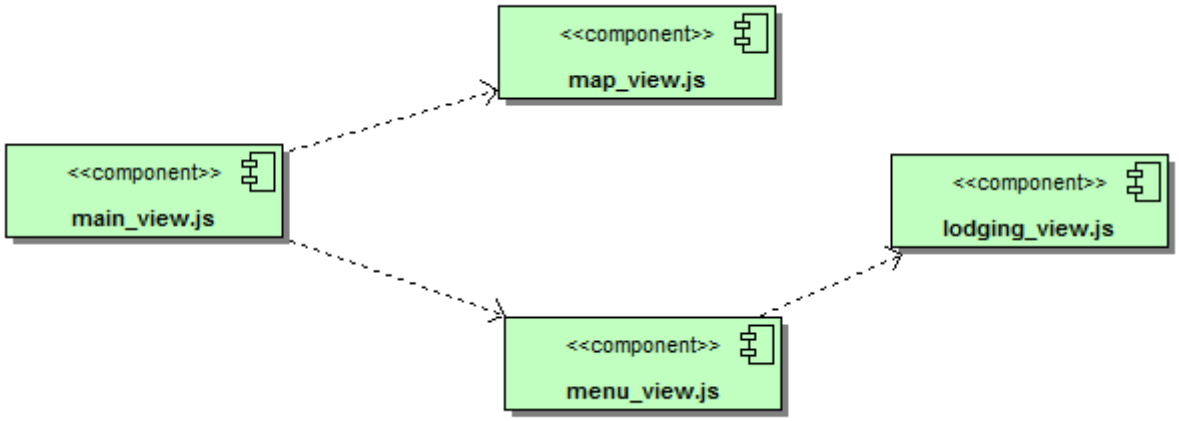


Figura 4.8: Vistas de aplicación cliente

Cuatro vistas controlan la representación visual de los datos manejados por la aplicación web: main_view, map_view, menu_view y lodging_view.

main_view se encarga de montar la plantilla del mismo nombre y proyectan (o renderizan) sobre ella los componentes visuales soportados por sus demás vistas hermanas.

Como sugieren sus nombres, map_view crea el mapa interactivo de la aplicación y todos sus elementos (incluyendo los marcadores y las rutas a la hora de solicitar los itinerarios). menu_view crea y visualiza el formulario para la generación de rutas contenido en su plantilla respectiva y, por último, lodging_view lista y asigna a un campo correspondiente del formulario de menu los alojamientos disponibles almacenados en la base de datos del servidor.

4.3.6. Interfaz gráfica de aplicación

La interfaz gráfica de la aplicación es lo más sencilla e intuitiva posible, contando con la siguiente estructura señalada a continuación:

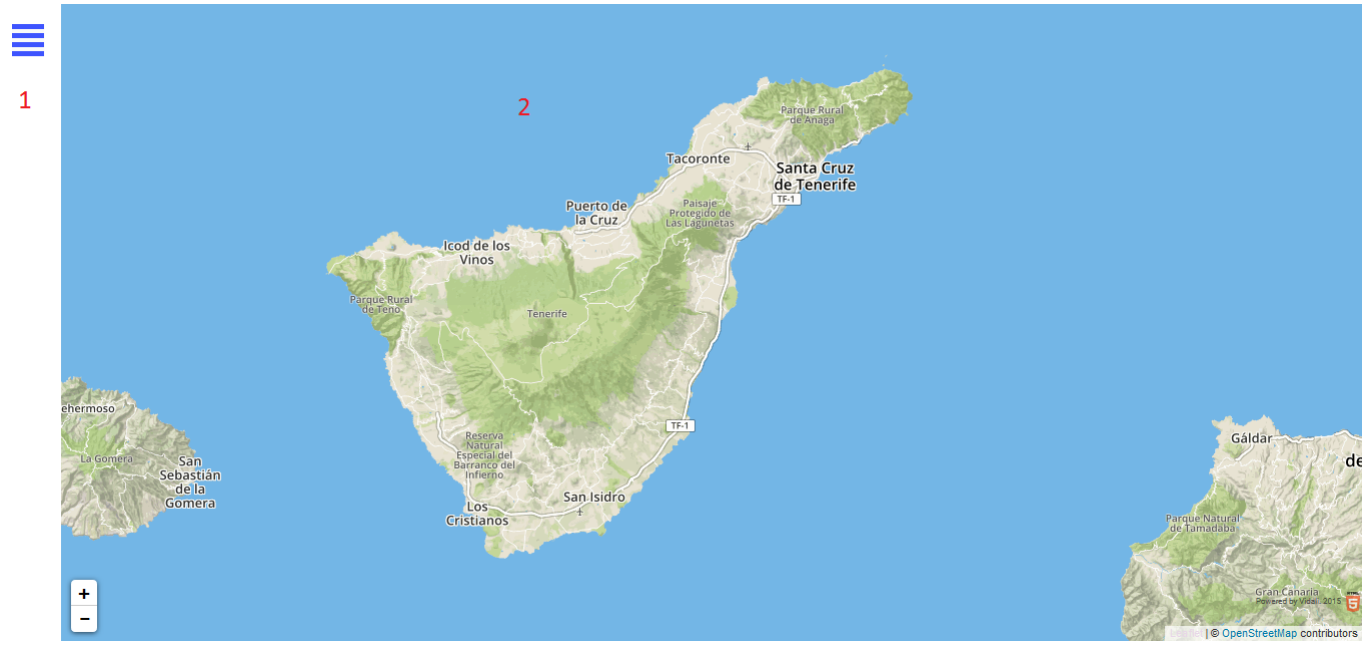


Figura 4.9: Interfaz principal de aplicación web

1. Barra desplegable de menu de formulario.
2. Mapa interactivo de la aplicación.

Al darle click al ícono de menu de la barra, se despliega el formulario para la generación de itinerarios de rutas.

Figura 4.10: Interfaz de formulario para generación de itinerarios

Dicho formulario puede cerrarse dando click al ícono de "X" donde antes se encontraba el alusivo al menu.

Si completamos el formulario y lo enviamos, después de algunos segundos se ocultará el formulario y aparecerán las rutas por día y sus destinos de senderismo del itinerario calculados por el servidor.

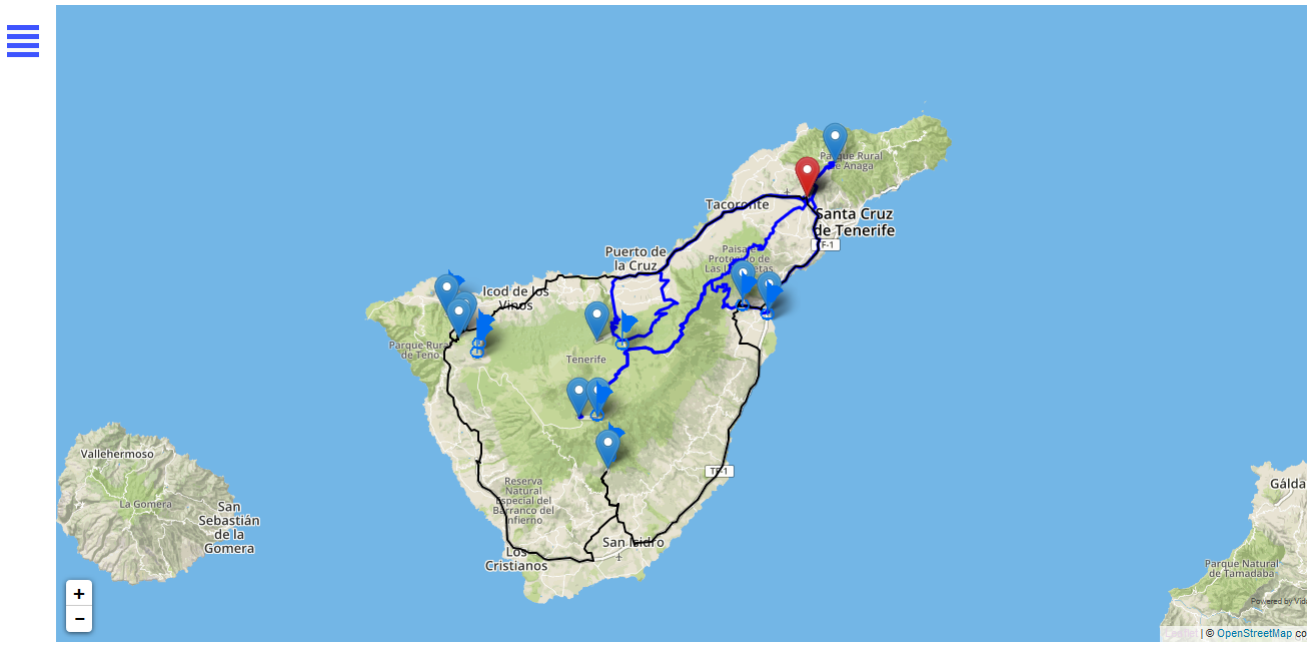


Figura 4.11: Rutas generadas de itinerario con los marcadores de sus destinos

Cada ruta tiene un color distinto para ayudar a diferenciarlas entre si, lo cual es necesario puesto a que suelen solaparse con frecuencia. Los marcadores azules representan las rutas de senderismo a visitar y el rojo al alojamiento

desde donde se partirá e irá al final de cada día. Cabe destacar que las rutas de senderismo de tipo lineal tienen un marcador adicional en forma de bandera que sirve para señalar el punto final de su ruta, lo cual puede ser de interés ya que muchas veces no es deseable o factible regresar al punto de inicio.

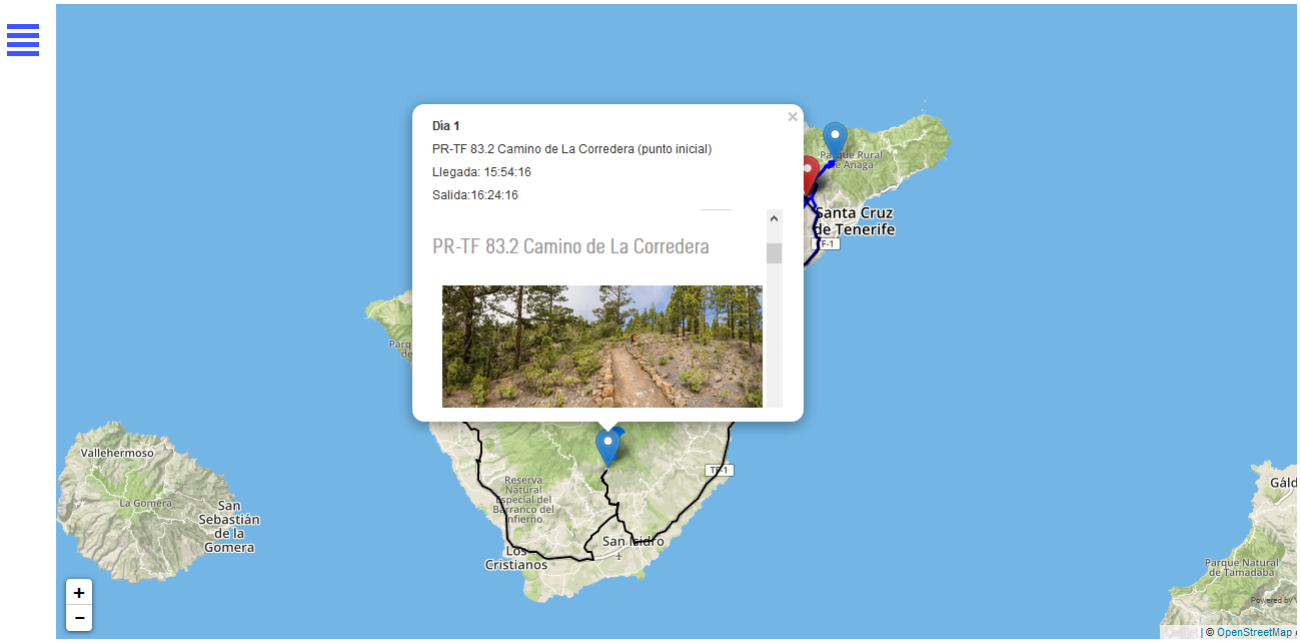


Figura 4.12: Ventanas de marcadores de rutas y destinos

Al dar click en el marcador de una ruta, se nos abrirá una ventana con su nombre, tiempo de llegada y partida, y una subventana con una vista de la información detallada de la ruta, proporcionada por su artículo en Webtenerife.

Capítulo 5

Desarrollo de aplicación de Recomendador de rutas para actividades de turismo rural

En este capítulo se reseñan los aspectos más relevantes y significativos del desarrollo del código fuente tanto del servidor como de la aplicación cliente del sistema recomendador, destacando sus métodos y secciones correspondientes al cumplimiento de los requisitos funcionales establecidos en el análisis.

5.1. Servidor

El servidor fue escrito en Java, y despliega servicios web que reciben las peticiones del cliente, los cuales le retornan los resultados procesados por las clases relativas al TOPTW y su algoritmo de resolución basado en la heurística GRASP.

Este apartado se comenzará describiendo las funciones de los servicios web, y luego las de las clases de la familia TOPTW más relevantes en cuanto al cumplimiento de los requisitos funcionales.

5.1.1. Servicios web

El servidor cuenta con dos ficheros de clases alusivas a sus servicios web de cálculo de itinerarios de rutas óptimas y lista de alojamientos disponibles: routesWebService.java y lodgingWebService.java, respectivamente.

La función de la clase routesWebService utilizada para recibir las peticiones de la aplicación cliente y usar su información para calcular y retornarle una solución es getRoutesfromInput.

```
@Path("{j}")
@GET
@Produces("application/json")
public Response getRoutesfromInput(@Context ServletContext context,
@PathParam("j") JSONObject j)
throws InstantiationException, IllegalAccessException,
ClassNotFoundException, SQLException, JSONException, IOException
{
        //you can specify in your method argument
        int hotel = j.getInt("lodging");
        int days_stay = j.getInt("days_of_stay");
        float maxHeight = (float)j.getDouble("max_height");
        float minHeight = (float)j.getDouble("min_height");
        float max_elevation_gain = (float)j.getDouble("max_elevation_gain");
        float max_elevation_loss = (float)j.getDouble("max_elevation_loss");
        float maxDistance = (float)j.getDouble("max_distance");

        boolean[] difficulties = new boolean[3];
        difficulties[0] = j.getBoolean("easy");
        difficulties[1] = j.getBoolean("moderate");
        difficulties[2] = j.getBoolean("hard");
        boolean[] route_types = new boolean[3];
        route_types[0] = j.getBoolean("lineal");
        route_types[1] = j.getBoolean("circular");
        String transport = j.getString("transport");
        TOPTW_MySQLreader connection = new TOPTW_MySQLreader(hotel,
        maxDistance, maxHeight, minHeight, max_elevation_gain,
        max_elevation_loss, difficulties, route_types, transport);
        TOPTW problem = connection.createProblem(days_stay);
        TOPTWSolution solution = new TOPTWSolution(problem);
        TOPTWGRASP grasp = new TOPTWGRASP(solution);
        grasp.GRASP(50, 5);
        System.out.println(grasp.getBest_solution()
        .getDouble("score"));
        String result = grasp.getBest_solution().toString(1);
        System.out.println(result);
        return Response.status(200).header("Access-Control-Allow-Origin",
        "*").header("Access-Control-Allow-Methods", "GET, POST,
        DELETE, PUT").entity(result).build();
}
```

Figura 5.1: Función getRoutesfromInput

La función primero recoge una serie de parámetros pasados como atributos de un objeto JSON enviado en la petición del cliente. Estos parámetros son usados para crear la instancia TOPTW con las restricciones de rutas y destinos

deseadas con la clase TOPTW_MySQLreader y su función createProblem. Al obtener el problema, se genera una solución inicial a partir del mismo con un objeto de la clase TOPTWSolution y se le aplica el algoritmo GRASP, para optimizarla con un objeto de la clase TOPTWGrasp y su método análogo.

Al obtener la solución óptima, se retorna como un objeto JSON gracias al método getBest_Solution que la transforma a dicho formato de datos, siguiendo la siguiente estructura:

```
{
 "score": 1030,
 "solution": {
  "route_3": [
   {
    "acc. ascent": "0.0",
    "image": "null",
    "min height": "0.0",
    "ready time": "0.0",
    "acc. descent": "0.0",
    "max height": "0.0",
    "due date": "75600.0",
    "distance": "0.0",
    "service time": "0.0",
    "long_end": "0.0",
    "link": "null",
    "leave time": "0",
    "type": "null",
    "lat_end": "0.0",
    "long": "-16.318953036825405",
    "difficulty": "null",
    "arrive time": "0",
    "name": "Hotel Aguere",
    "cust no.": "33",
    "lat": "28.48967677234651"
   },
   {
    "acc. ascent": "1000.38",
    "image": "http://www.webtenerife.com/es/que-hacer/naturaleza/senderismo/
       senderos/publishingimages/pr-tf-83-altos-granadilla-fc.jpg",
    "min height": "894.0",
    "ready time": "36000.0",
    "acc. descent": "37.38",
    "max height": "1860.0",
    "distance": "7800.0",
    "service time": "12600.0",
    "long_end": "-16.604251861572266",
    "link": "http://www.webtenerife.com/que-hacer/naturaleza/senderismo/
       senderos/prtf83-altos-granadilla.htm",
    "leave time": "48600.0",
    "type": "Lineal",
    "lat_end": "28.189035415649414",
```

```
 "long": "-16.59272575378418",
 "difficulty": "Dificil",
 "arrive time": "36000.0",
 "name": "PR-TF 83 Altos de Granadilla",
 "due time": "64800.0",
 "cust no.": "23",
 "lat": "28.13254737854004"
},
{
 "acc. ascent": "211.0",
 "image": "http://www.webtenerife.com/es/que-hacer/naturaleza/senderismo/
    senderos/publishingimages/3-roques-garcia-parque-nacional-teide-fc.jpg
    ",
 "min height": "2025.0",
 "ready time": "36000.0",
 "acc. descent": "215.0",
 "max height": "2187.0",
 "distance": "3500.0",
 "service time": "7200.0",
 "long_end": "-16.63128662109375",
 "link": "http://www.webtenerife.com/que-hacer/naturaleza/senderismo/
    senderos/roques-garcia.htm",
 "leave time": "57756.4",
 "type": "Circular",
 "lat_end": "28.223140716552734",
 "long": "-16.63128662109375",
 "difficulty": "Facil",
 "arrive time": "50556.4",
 "name": "P.N. El Teide - Roques de García",
 "due time": "64800.0",
 "cust no.": "3",
 "lat": "28.223140716552734"
},
{
 "acc. ascent": "369.0",
 "image": "http://www.webtenerife.com/es/que-hacer/naturaleza/senderismo/
    senderos/publishingimages/2-arenas-negras-parque-nacional-teide-fc.jpg
    ",
 "min height": "2038.0",
 "ready time": "36000.0",
 "acc. descent": "356.0",
 "max height": "2304.0",
 "distance": "7600.0",
 "service time": "10800.0",
 "long_end": "-16.56631851196289",
 "link": "http://www.webtenerife.com/que-hacer/naturaleza/senderismo/
    senderos/arenas-negras.htm",
 "leave time": "69390.6",
 "type": "Circular",
 "lat_end": "28.303800582885742",
 "long": "-16.56631851196289",
 "difficulty": "Moderada",
```

```
    "arrive time": "58590.6",
    "name": "P.N. El Teide - Arenas negras",
    "due time": "64800.0",
    "cust no.": "2",
    "lat": "28.303800582885742"
   },


   {
    "acc. ascent": "0.0",
    "image": "null",
    "min height": "0.0",
    "ready time": "0.0",
    "acc. descent": "0.0",
    "max height": "0.0",
    "distance": "0.0",
    "service time": "0.0",
    "long_end": "0.0",
    "link": "null",
    "leave time": "71917.20000000001",
    "type": "null",
    "lat_end": "0.0",
    "long": "-16.318953036825405",
    "difficulty": "null",
    "arrive time": "71917.20000000001",
    "name": "Hotel Aguere",
    "due time": "75600.0",
    "cust no.": "33",
    "lat": "28.48967677234651"
   }
  ],
  "route_2": [
  ...
  ],
  "route_1": [
        ...
  ],
  "route_0": [
  ...
  ]
 },
 "number of nodes": 30,
 "time_cost": 283761.5,
 "max time per route": 75600,
 "max number of routes": 4
}
```


Figura 5.2: Objeto JSON retornado por getRoutesfromInput

La cual comprende a las rutas para cada día del itinerario de la solución (en este ejemplo 4), los detalles de sus puntos de interés (incluyendo el alojamiento

de partida) y demás datos relevantes de la solución (como el coste de tiempo máximo permitido por ruta, el número de rutas generadas y el número de nodos involucrados).

```
@GET
@Produces("application/json")
public Response getLodgings(@Context ServletContext context)
throws SQLException, InstantiationException
IllegalAccessException, ClassNotFoundException
{
       Connection connection = null;
       try {
              Class.forName("com.mysql.jdbc.Driver").newInstance();
              connection = DriverManager.getConnection
              ("jdbc:mysql://localhost/rtvrptw","root", "");
              }
              catch (SQLException e)
              {
              e.printStackTrace();
              }
       Statement s = connection.createStatement();
       QueryManager qm = new QueryAll();
       return Response.status(200).header("Access-Control-Allow-Origin", "*")
       .header("Access-Control-Allow-Methods",
       "GET, POST, DELETE,PUT") .entity(qm.getLodgings(s)
       .toString(1)).build();
}
```

Figura 5.3: Función getLodgings

Por otro lado, la función getLodgings del servicio web lodgingWebService hace algo similar, creando una conexión a la base de datos MySQL y usando un gestor de consultas (QueryManager), el cual retorna con el método análogo getLodgings un objeto JSON que sigue la estructura a continuación:

```
{"alojamientos": [
        {
         "id": 1,
         "nombre": "Hotel Aguere"
        },
        {
         "id": 2,
         "nombre": "Hotel rural Finca Salamanca"
        },
        {
         "id": 4,
         "nombre": "Hotel rural Hotel Rural San Miguel"
        },
        {
         "id": 5,
         "nombre": "Casa rural Casa Anton Piche"
        },
...
]}
```

Figura 5.4: Objeto JSON retornado por getLodgings

5.1.2. Lectura de base de datos y creación de instancia TOPTW

Tal como se puede apreciar en la función getRoutesfromInput de routesWebService, la clase encargada de acceder a la base de datos MySQL y crear una instancia de un modelo TOPTW con el alojamiento y la rutas de senderismo según los parámetros del usuario es TOPTW_MySQLreader.

El método con el que obtiene los datos de la base de datos y crea el problema es createProblem.

```
public TOPTW createProblem(int days) throws SQLException,
JSONException, IOException
{
        QueryManager qm = null;
        TOPTW problem = null;
        Statement s;
        s = connection.createStatement();
        qm = queryMode();
        int numberRow = qm.getNumberofRows(s);
        System.out.println(numberRow);
        problem = new TOPTW(numberRow, days, this.transport);
        ResultSet rs = qm.getStartPoint(s, this.hotel);
        problem = setOrigin(problem, rs);
        rs = qm.getPOIs(s);
        int i = 1;
        while(rs.next()) {
               if (rs.getFloat(5)<=this.max_distance && rs.getFloat(14)<=this.
                  max_time && rs.getFloat(6) <= this.max_elevation_loss && rs.
                  getFloat(7)<=this.max_elevation_gain && rs.getFloat(8)<=this
                  .max_height && rs.getFloat(9) >= this.minimum_height)
               {
                      problem.setLatitudeOrigin(i, rs.getFloat(2));
                      problem.setLongitudeOrigin(i, rs.getFloat(3));
                      problem.setName(i, rs.getString(4));
                      problem.setDistance(i, rs.getFloat(5));
                      problem.setAccumulated_descent_meters(i, rs.getFloat(6))
                          ;
                      problem.setAccumulated_ascent_meters(i, rs.getFloat(7));
                      problem.setMax_height(i, rs.getFloat(8));
                      problem.setMinimum_height(i, rs.getFloat(9));
                      problem.setRoute_type(i, rs.getString(10));
                      problem.setDifficulty(i, rs.getString(11));
                      problem.setReadyTime(i, rs.getDouble(12));
                      problem.setDueTime(i, rs.getDouble(13));
                      problem.setServiceTime(i, rs.getDouble(14));
                      problem.setScore(i, rs.getInt(15));
                      problem.setImage(i, rs.getString(16));
                      problem.setLink(i, rs.getString(17));
                      problem.setMaxTimePerRoute(problem.getDueTime(0));
                      problem.setLatitude_end(i, rs.getFloat(18));
                      problem.setLongitude_end(i, rs.getFloat(19));
               }
```

```
        i++;
}
problem.calculateRealDistanceMatrix();
return problem;
}
```

Figura 5.5: Función createProblem

Esta función, mediante la ayuda de la clase QueryManager que gestiona las principales consultas a la base de datos, se conecta a la misma y determina el número de puntos de interés obtenidos de acuerdo a los parámetros más estrictos (dificultades y tipos de rutas). Entonces, crea una instancia del TOPTW a partir de esa información y los días de estadía (es decir, el número máximo de rutas a generar), a la cual se configura a continuación su punto de partida (esto es, el alojamiento escogido por el usuario) y los datos de cada uno de los puntos de interés que cumplen con las restricciones de la petición del usuario.

De dichos puntos, se calcula una matriz de distancia real con el método calculateRealDistanceMatrix (el cual invoca al método createDistanceMatrix, detallado en el siguiente apartado) antes de retornar la instancia TOPTW completamente construida.

5.1.3. Cálculo de matriz de distancia

El cálculo de la matriz de distancia real se realiza mediante una petición a un servidor local del motor de cálculo de rutas georreferenciadas Open Source Routing Machine o OSRM (más información en la sección de Tecnologías y herramientas), la cual se realiza con el método calculate_distanceMatrix de la clase DistanceMatrix.

```
        public JSONObject calculate_distanceMatrix(TOPTW problem)
        throws JSONException, IOException
{
        String matrix_points = "http://127.0.0.1:5000/table/v1/driving/";
        System.out.println(problem.getPOIs());
                for (int i = 0; i<problem.getPOIs()+1; i++)
                {
                        String point = "";
                        if (problem.getRoute_type(i)!=null && problem.
                           getRoute_type(i).matches("Lineal") && problem.
                           getTransport().matches("PublicTransport"))
                        {
                                point = "" + problem.getLongitudeOrigin(i)
                                +","+problem.getLatitudeOrigin(i)+";"
                                +problem.getLongitude_end(i)+","
                                +problem.getLatitude_end(i);
                        }
                        else
                                point = "" + problem.getLongitudeOrigin(i) +","+
                                   problem.getLatitudeOrigin(i);
                        if (i<problem.getPOIs())
                                matrix_points += point+";";
                        else
                                matrix_points += point;
                }
        System.out.println(matrix_points);
        JSONObject distancematrix = jreader.readJsonFromUrl(matrix_points);
        //System.out.println(distancematrix.toString(1));
        return distancematrix;

}
```

Figura 5.6: Función calculate_distanceMatrix

Si el tipo de transporte para recorrer las rutas se trata de transporte público, se toman en cuenta los puntos de destino de las rutas de senderismo lineales, ya que en vez de regresar al punto de partida el turista cogería alguna "guagua" o bus desde dicho punto final. Una vez listados todos los puntos que formarán parte de la matriz de distancia, se envía la petición, y su resultado retornado por una URL es leído y procesado como un objeto JSON por el método readJsonFromUrl de la clase JSONReader.

Para configurar la matriz de distancia del problema de acuerdo a esa información se emplea el método createDistanceMatrix.

Dicho método, dependiendo de si el usuario elige un coche propio o transporte público como el tipo de transporte a usar, asigna diferentes tiempos a los pares de la matriz de distancia definitiva. Si la ruta inicial del par es una ruta de senderismo lineal, el punto de origen del par será el punto de llegada de dicha

ruta de senderismo inicial.

```
public void createDistanceMatrix(TOPTW problem) throws
JSONException, IOException
{
        JSONObject calculated_dandt = calculate_distanceMatrix(problem);
        int i =0;
        int auxi=0;
        while (i<calculated_dandt.getJSONArray("durations").length())
        {
                if (problem.getRoute_type(auxi)!= null &&
                        problem.getRoute_type(auxi).matches("Lineal") &&
                        problem.getTransport().matches("PublicTransport"))
                        i++;
                int j = 0;
                int auxj=0;
                DistanceMatrixRow row = new DistanceMatrixRow();
                while (j<calculated_dandt.getJSONArray("durations").length())
                {
                        if (i != j)
                 {
                                double distance = 0;
                                double time = 0;
                                if (calculated_dandt.getJSONArray("durations").
                                   getJSONArray(i).getDouble(j)>300)
                                {
                                        if (problem.getTransport().matches("Car"))
                                        {
                                                distance = calculated_dandt.
                                                   getJSONArray("durations").
                                                   getJSONArray(i)
                                                .getDouble(j);
                                                time =
                                                calculated_dandt.getJSONArray("
                                                   durations").getJSONArray(i)
                                                .getDouble(j);
                                        }
                                        else
                                        {
                                                distance = (calculated_dandt.
                                                   getJSONArray("durations").
                                                   getJSONArray(i)
                                                .getDouble(j)+900)*1.2;
                                                time =
                                                (calculated_dandt.getJSONArray("
                                                   durations").getJSONArray(i)
                                                .getDouble(j)+900)*1.2;
                                        }
                                }
```

```
                    else
                    {
                            double auxdistance =
                            calculated_dandt.getJSONArray("durations").
                               getJSONArray(i).getDouble(j)*50000/3600;
                            distance = auxdistance*3600/5000;
                            time = auxdistance*3600/5000;

                    }
                    DistanceAndTime dandt = new DistanceAndTime(distance,
                       time);
                    row.add_pair(problem.getName(auxj), dandt);
                    if(problem.getRoute_type(auxj)!=null &&
                    problem.getRoute_type(auxj).matches("Lineal") &&
                    problem.getTransport().matches("PublicTransport"))
                            j++;
                    }
                    else
                    {
                            DistanceAndTime dandt = new DistanceAndTime(0, 0)
                               ;
                            row.add_pair(problem.getName(auxj), dandt);
                            if(problem.getRoute_type(auxj)!=null &&
                            problem.getRoute_type(auxj).matches("Lineal") &&
                            problem.getTransport().matches("PublicTransport")
                               )
                                j++;
                    }
                    matrix.put(problem.getName(auxi), row);
                    j++;
                    auxj++;
                    }
            i++;
            auxi++;
            }
}
```

Figura 5.7: Función createDistanceMatrix

Hay que destacar que en esta implementación tanto la distancia como el tiempo se miden en función del tiempo de viaje entre ambos puntos de los pares. Cuando la distancia entre los dos es menor a dos kilómetros (lo cual se estima a un tiempo menor o igual a 300 segundos en base a una velocidad de referencia de 50 km/h), se asume que el turista se mueve a pie y el tiempo a asignar es aquel en que tardaría en recorrer dicha distancia aproximada (calculada con una regla de tres) con una velocidad de caminata promedio de 5 km/h.

5.1.4. Optimización de solución de TOPTW con algoritmo GRASP

Una vez generada una solución inicial del TOPTW, se procede a mejorarla con el método GRASP de la clase TOPTWGrasp [43].

```
 public void GRASP(int maxIterations, int maxSizeRCL)
{
	double averageFitness = 0.0;
	double bestSolution = 0.0;
	for(int i = 0; i < maxIterations; i++) {

		this.computeGreedySolution(maxSizeRCL);

		// IMPRIMIR SOLUCION
		double fitness = this.solution.evaluateFitness();
		//System.out.println("Press Any Key To Continue...");
		//new java.util.Scanner(System.in).nextLine();
		averageFitness += fitness;
		if(bestSolution < fitness) {
			bestSolution = fitness;
			//System.out.println(this.solution.getInfoSolution());
			this.solution.setSolution_data();
			this.best_solution = this.solution.getSolution_data();
			System.out.println(this.best_solution);
			//this.best_solution=this.solution;
		}
		//double fitness = this.solution.printSolution();

		/******
		*
		* Local search
		*
		*/
	}
	averageFitness = averageFitness/maxIterations;
	System.out.println(" --> MEDIA: "+averageFitness);
	System.out.println(" --> MEJOR SOLUCION: "+bestSolution);
}
```

Figura 5.8: Función GRASP

Esta función implementa las distintas fases del algoritmo (explicado en la sección de Metodologías y modelos algorítmicos). Recibe como parámetros el número máximo de iteraciones para aplicar la heurística y el el tamaño de la la lista restringida de candidatos o RCL.

Primero, inicializa las variables que guardarán la puntuación promedio y la mejor de todas las soluciones. Entonces, para cada una de las i iteraciones,

genera una solución factible para el problema con la función de construcción computeGreedySolution, que utiliza el tamaño de la RCL establecido. Tras obtener dicha solución, se procede a la fase de búsqueda local, verificando si su puntuación total es superior a la última mejor guardada. Si lo es, la solución de la iteración en cuestión se vuelve la nueva mejor del problema y se guarda como un objeto JSON (best_solution). Dicho proceso se repite mejorando continuamente las soluciones obtenidas hasta que se llega al número máximo de iteraciones establecido, tras lo cual se muestran en pantalla las puntuaciones media y mejor resultantes.

5.2. Aplicación cliente

Al ser una aplicación web, el código de la aplicación cliente se desarrolló combinando los lenguajes HTML, CSS y JavaScript, siguiendo una arquitectura basada en el patrón Modelo Vista Controlador. En los siguientes apartados se destacan los ficheros y funciones más importantes de la misma.

5.2.1. Vista y plantilla principal

Tal como se muestra en el diseño, la vista sobre la que se articula la aplicación y el resto de sus vistas es main_view.js.

```
define([
  'jquery', 'underscore', 'backbone', 'bootstrap', 'config', 'globalFunctions
     ',
  'views/menu_view',
  'views/map_view',
  'text!templates/main.html'],
  function ($, _, Backbone, Bootstrap, Config, Global, MenuView, MapView, Main
     ) {


    var MainView = Backbone.View.extend({
      /**
      * Main view of RRS
      * @augments Backbone.View
      * @constructor
      * @name MainView
      */
      /**
      * @memberof MainView
      * @desc Saves the div where will be placed the template.
      */
      el: $('#container'),
      list: null,
      page: null,
      map: null,

          ...
          });

          return MainView;

          });
```

Figura 5.9: Extracto de código de main_view.js

Esta vista contiene métodos que cargan y anexan las demás (map_view.js y menu_view.js) a su plantilla HTML análoga main.html y controla eventos que modifican el comportamiento de sus componentes visuales.

```
<div id="wrapper">
       <div class="overlay"></div>
       <div id="mainmenu"></div>
       <div id="entityPanel"></div>
       <div id="page-content-wrapper">
              <div id="searchedBox"></div>
              <div id="searchAllPanel"></div>
              <button id='hamburger' type="button" class="hamburger is-closed
                 " data-toggle="offcanvas" data-placement="right" title="Ver
                 formulario">
```

```
                        <span class="hamb-top"></span>
                        <span class="hamb-middle"></span>
                        <span class="hamb-bottom"></span>
                </button>
                <section id="action"></section>
                <section id="list">
                        <div class="row fullwhite">
                                <div id="listcontent"></div>
                        </div>
                </section>
                <aside id="side">
                        <div id="pointsListWrapper" class="sideWrapper">
                                <div id="pointsList" class="is-closed">
                                </div>
                        </div>
                        <div id="showPointWrapper" class="sideWrapper2">
                                <div id="showPoint" class="is-closed">
                                </div>
                        </div>
                        <div id="routesBoxWrapper" class="sideWrapper">
                                <div id="routesBox" class="is-closed">
                                </div>
                        </div>
                        <div id="showRouteWrapper" class="sideWrapper2">
                                <div id="showRoute" class="is-closed">
                                </div>
                        </div>
                        <div id="showStopWrapper" class="sideWrapper2">
                                <div id="showStop" class="is-closed">
                                </div>
                        </div>
                </aside>
                <div id="map"></div>


        </div>
</div>

<footer class="footer">
        <p class="pull-right">Powered by RRS. 2017 <img src="images/html5.png"
           width="20"/></p>
</footer>

<!-- Aqui elementos visuales que deben aparecer en algun momento como modales,
    botones, cajas, etc -->
<!-- Update Box-->
```

```
<form id="updater">
       <div class="widget-area no-padding blank">
              <div class="row">
                     <div class="col-md-12">
                            <div class="status-upload">
                                   <textarea id="update-box" name="update"
                                      class="updater post-text" placeholder="
                                      Que ocurre en este lugar?" ></textarea>
                                   <!-- Status Upload <ul>
                                          <li><a title="" data-toggle="
                                             tooltip" data-placement="bottom
                                             " data-original-title="Agregar
                                             Foto"><i class="fa fa-picture-o
                                             "></i></a></li>
                                   </ul>-->
                                   <a class="update-status btn btn-success
                                      green margShare"><i class="fa fa-share
                                      "></i> Compartir</a>
                            </div><!-- Status Upload -->
                     </div><!-- Widget Area -->
              </div>
       </div>
</form>

<!-- Create Group Modal-->
<div class="modal" id="createGroupModal" tabindex="-1" role="dialog" aria-
   labelledby="modalLabel" aria-hidden="true"></div>
```

Figura 5.10: Código en HTML de plantilla main.html

5.2.2. Vista y plantilla menu

menu_view.js es la vista que carga y visualiza el formulario de datos para la generación de itinerarios de rutas de senderismo.

```
define([
  'underscore',
  'backbone',
  'config',
  'text!templates/menu.html',
  'globalFunctions',
  'collections/lodgings_collection',
  'views/lodging_view'
], function(_, Backbone,Config,Template, Global, lodgingCollection,
    lodgingView){
        var MenuView = Backbone.View.extend({
       /**
      * Menu view implementation for RRS
      * @augments Backbone.Router
      * @constructor
      * @name MenuView
      */
    el: '#mainmenu',
    /**
     * @memberof MenuView
     * @desc Saves the value of the active menu
     */

    activeMenu: null,

        data_collection: null,

        lodgingView: null,

        ...
        });

        return MenuView;

        });
```

Figura 5.11: Extracto de código de menu_view.js

Dicho formulario se encuentra definido por la plantilla menu.html.

```
<nav class="navbar navbar-inverse navbar-fixed-top" id="sidebar-wrapper" role
   ="navigation">
  <ul class="nav sidebar-nav">
  <h4>Datos para la generación de itinerarios de rutas</h4>
   <form name="miformulario" action="" id = "formulario">
                <div class="form-group">
                 <label>Alojamiento</label>
                <select class ="form-control" class="selectpicker" id = '
                   lodging-template' >
```

```
 </select>
 </div>
  <div class="form-group">
 <label>Días de estadía</label>
<input type="text" class="form-control" id="tiempomaxInput"
   name="tiempomax" value="2" maxlength="50">
</div>
  <div class="form-group">
 <label>Distancia máxima (metros)</label>
<input type="text" class="form-control" id="distanciamaxInput"
   name="distanciamax" value="20000" maxlength="50">
 </div>
</div>
<div class="form-group">
 <label>Desnivel acumulado ascendente</label><br><label>máximo (
    metros)</label>
<input type="text" class="form-control" id="desnivelAscenInput"
    name="desnivelascen" value="1000" maxlength="50">
</div>
<div class="form-group">
 <label>Desnivel acumulado descendente</label><br><label> máximo
     (metros)</label>
<input type="text" class="form-control" id="desnivelDescenInput
   " name="desniveldescen" value="2000" maxlength="50">
</div>
<div class="form-group">
 <label>Altura máxima (metros)</label>
<input type="text" class="form-control" id="alturamaxInput"
   name="alturamax" value="3000" maxlength="50">
</div>
<div class="form-group">
 <label>Altura mínima (metros)</label>
<input type="text" class="form-control" id="alturaminInput"
   name="alturamin" value="0" maxlength="50">
</div>
<div class="form-group">
<label class="checkbox">Dificultad de rutas:</label>
<input type="checkbox" id="Op1-1" name="Opcion" value="Facil"
   checked>Fácil.
<br>
<input type="checkbox" id="Op1-2" name="Opcion" value="Moderada
   " >Moderada.
<br>
<input type="checkbox" id="Op1-3" name="Opcion" value="Dificil"
    >Difícil.
</div>
<div class="form-group">
<label class="checkbox">Tipos de rutas:</label>
<input type="checkbox" id="Op2-1" name="Opcion" value="Facil"
   checked>Lineal.
<br>
```

```
                <input type="checkbox" id="Op2-2" name="Opcion" value="Moderada
                    " >Circular.
                <br>
                </div>
                <div class="form-group">
                <label>Tipo de transporte:</label><br>
                <input type="radio" name="transporte" value="Car" checked>
                    Coche propio<br>
                  <input type="radio" name="transporte" value="PublicTransport">
                      Transporte público<br>
                </div>
                <div class="text-center">
                <input type="button" value="Enviar" class="btn btn-primary" id
                    ="SubmitBtn"> <input type="reset" value="Reset" class="btn
                    btn-primary">
        </ul>
</nav>
```

Figura 5.12: Código en HTML de plantilla menu.html

menu_view.js tiene como vista anexa a lodging_view.js.

5.2.3. Vista lodging

lodging_view.js es la vista que obtiene la lista de alojamientos del servidor y la carga en el campo "select" respectivo del formulario de su vista padre menu_view.js.

```
define([
  'underscore',
  'backbone',
  'config',
  'globalFunctions'
], function(_, Backbone,Config,Global){

 var lodgingView = Backbone.View.extend({
     /**
      * Lodging view implementation for RRS
      * @augments Backbone.Router
      * @constructor
      * @name LodgingView
      */
           /**
     * @memberof LodgingView
     * @desc Displays the lodgings data into the select field of the menu bar
        form.
     */
    render: function () {
                template= _.template("<option value = <%= id %>> <%= nombre
                    %> </option>");
        $("#lodging-template").append(template(_.extend(this.model.toJSON()
       )));

    }
});
return lodgingView
});
```

Figura 5.13: Código de lodging_view.js

5.2.4. Vista de mapa

La vista encargada de proyectar el mapa interactivo sobre la vista principal es map_view.js.

```
define([
  'jquery', 'underscore', 'backbone', 'moment', 'bootstrap',
  'leaflet', 'leaflet-awesome-markers', 'leaflet-routing-machine', 'leaflet-
     control-geocoder',
  'models/map_model', 'config'
], function($, _, Backbone, Moment, Bootstrap,
  Leaflet, AwesomeMarkers, Geocoder, RoutingMachine,
  MapModel, Config){
    var mapView = Backbone.View.extend({
      /**
      * Map view implementation for RRS
      * @augments Backbone.Router
      * @constructor
      * @name MapView
      */
      /**
      * Saves the map model.
      * @memberof MapView
      * @instance
      * @type {object}
      */
      model: new MapModel(),
      OSMap: null,
      basicMap: null,
      MapBox: null,
      transportMap: null,
      StartDevsMap: null,
      baseLayers: null,
      markerList: null,
          routeColors: null,

          ...
          });
            return mapView;
          });
```

Figura 5.14: Extracto de código de map_view.js

Esta vista cuenta con la función encargada de dibujar las rutas de los itinerarios y sus marcadores cuando el usuario así lo solicita, su nombre es renderRoutes y consta de varias partes.

```
renderRoutes: function()
        {
               String.prototype.toHHMMSS = function () {
                     var sec_num = parseInt(this, 10); // don't forget the
                        second param
                     var hours  = Math.floor(sec_num / 3600);
                     var minutes = Math.floor((sec_num - (hours * 3600)) /
                        60);
                     var seconds = sec_num - (hours * 3600) - (minutes * 60);

                     if (hours  < 10) {hours  = "0"+hours;}
                     if (minutes < 10) {minutes = "0"+minutes;}
                     if (seconds < 10) {seconds = "0"+seconds;}
                     return hours+':'+minutes+':'+seconds;
                     };
               this.markers.clearLayers();
               this.mapa.removeLayer(this.markers);
                           var map = this.mapa;
                                       var routeControls = this.controls;
                                        if (routeControls.length >= 1)
               {
                      for (var i = 0; i<routeControls.length; i++)
                      {
                             routeControls[i].removeFrom(map);
                      }
                      routeControls.length = 0;
               }
               var itineraryInfo = this. formDatatoJSON();
                                   var stay = itineraryInfo["days_of_stay"];
                                   var transport = itineraryInfo["transport
                                      "];
              var JSONrequest = JSON.stringify(itineraryInfo);
               var hamburger_cross = this.hamburger_cross;
              var markers = this.markers;
              var colors = this.routeColors;
              console.log(JSONrequest);
       this.drawRoutes(JSONrequest, stay, transport, routeControls, colors,
          markers, map, hamburger_cross);

        }
```

Figura 5.15: Funcion renderRoutes

Primero, y de haber existido una ejecución previa, dicha función elimina todos los marcadores y rutas dibujados en el mapa, para entonces cargar varias funciones y variables [44], entre las que se encuentra itineraryInfo. La misma guarda un objeto JSON retornado por la función formDatatoJSON, la cual recoge y reúne como atributos de un objeto de ese tipo los valores de los campos del formulario introducidos o seleccionados por el usuario.

Entonces, dicho objeto JSON es formateado como una cadena de una sola línea con la función stringify, cuyo resultado es guardado por la variable JSONRequest. Esta, junto a otras variables necesarias como las referentes a los marcadores y control de rutas, se asignan como parámetros a la función drawRoutes, que es la que envía la petición de cálculo de itinerario al servidor. Igualmente, obtiene y extrae la información de la solución retornada por el mismo, dibujando sus rutas como sugiere su nombre, acompañándolas con marcadores relativos a los destinos de cada una.

5.3. Despliegue del sistema recomendador de rutas en el IaaS

Tras culminar al desarrollo del sistema recomendador, se realizó un despliegue del mismo en uno de los servidores externos de la plataforma IaaS de la Universidad de La Laguna [45].

En dicho servidor se instalaron y configuraron una serie de tecnologías empleadas con frecuencia en el despliegue de aplicaciones cliente-servidor conocidas bajo el acrónimo de LAMP [46], el cual representa a sus herramientas:

- Sistema operativo Linux.
- Apache.
- MySQL/MariaDB.
- Perl, PHP o Python.

En este caso, el sistema operativo fue la versión 16.04 del sistema operativo Ubuntu (el cual está basado en Linux). Sobre él se montó un Apache HTTP Server 2.0 y, como se ha mencionado con anterioridad, se empleó MySQL como base datos. El lenguaje de programación de servidor que articula las interacciones entre esos componentes fue PHP.

Igualmente, se instaló y desplegó un servidor Tomcat para lanzar aplicaciones web escritas en Java (en concreto, el lado servidor del sistema recomendador), y una instancia del motor de cálculo de rutas georreferenciadas OSRM, usado por las aplicaciones web y el servidor del sistema para buena parte de la determinación de las rutas. Se puede ver más información sobre esta y otras herramientas en el capítulo 6.

El sistema recomendador se encuentra disponible a través del siguiente enlace `http://10.6.128.207/staging/`.

Para poder acceder a él, es necesario conectarse y autenticarse en la VPN de la Universidad de La Laguna [47].

Capítulo 6

Tecnologías de software empleadas en el desarrollo y despliegue

En este capítulo se listan las tecnologías y herramientas de software que se emplearon para la construcción y despliegue del sistema recomendador, tanto en su lado de servidor como en su lado de cliente.

6.1. Generales

6.1.1. GitHub

GitHub [48] [49] es una plataforma de desarrollo colaborativo de software basada en el sistema de control de versiones Git. Permite el alojamiento y revisión en línea del código fuente de proyectos de software de código abierto o propietario, así como su gestión y construcción a lo largo de su evolución.

Figura 6.1: GitHub

El código fuente del sistema recomendador, tanto de su aplicación web cliente como de su servidor, se encuentra alojado en el repositorio GitHub privado de la Universidad: `https://github.com/etsiiull/RTVRPTW`, el cual posteriormente se hará público tras la publicación de esta memoria.

6.1.2. WampServer

WampServer [50] es un entorno gratuito de desarrollo web para Windows que incopora un servidor Apache2, PHP, y una base de datos MySQL integrada con PhpMyAdmin, permitiendo una gestión simplificada de las misma.

Figura 6.2: WampServer

Tanto el lado cliente del sistema recomendador como la base de datos MySQL y el motor OSRM del lado servidor fueron desplegados y probados con esta plataforma en el ordenador propio que se usó durante el desarrollo.

6.2. Lado del servidor

6.2.1. Java SE Development Kit 8

Java SE Development Kit 8 [51] [52] [53], es la última versión del entorno de desarrollo para construir aplicaciones, applets y componentes en Java. Java es un lenguaje de programación basado en clases y objetos altamente tipeado, concurrente y multi propósito desarrollado y mantenido por Oracle.

Figura 6.3: Java

Tiene la particularidad de ser compilado e interpretado, ya que el código fuente es compilado en instrucciones bytecode y binarios que solo pueden ser leídos por una Máquina Virtual de Java. Esta característica lo vuelve multiplataforma, ya que sin importar el sistema operativo, cualquier sistema con una Máquina Virtual de Java podrá ejecutar cualquier aplicación escrita en dicho lenguaje.

6.2.2. Apache Maven

Apache Maven [54] es un una herramienta para la gestión y construcción de proyectos de software de la Apache Software Fundation. Basada en el concepto

de Project Object Model, es capaz de gestionar la construcción, las dependencias, los reportes y la documentación de un proyecto de software a través de diversas tareas y comandos predefinidos.

Figura 6.4: Apache Maven

El código fuente Java del servidor del sistema recomendador fue construido, administrado y compilado a través de un proyecto Maven.

6.2.3. Jersey RESTful Web Services framework

Jersey [55] es un framework de código abierto para el desarrollo de servicios web RESTful en Java. Provee soporte para APIs basadas en JAX-RS, la interfaz de programación de aplicaciones de Java oficial de Oracle para la creación de servicios web basados en el modelo REST, y así mismo sirve como su implementación estándar por parte de Oracle.

Figura 6.5: Jersey

Además de implementar el JAX-RS, Jersey dispone de su propio toolkit, características adicionales y utilidades para simplificar aún más el desarrollo de servidores y clientes bajo el modelo REST.

6.2.4. Apache Tomcat 8.0

Apache Tomcat 8.0 [56] [57] es la última versión de la implementación de código abierto de las tecnologias Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket, las cuales permiten construir y arrancar entornos de servidores web HTTP en los que se puede ejecutar código Java y retornar resultados con el mismo. Es desarrollada y mantenida por la Apache Software Foundation.

Figura 6.6: Apache Tomcat

El servidor del sistema recomendador corre el servicio web escrito en Java con la ayuda de Jersey gracias a esta herramienta.

6.2.5. MySQL

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation. Se considera como el gestor de base datos open source más popular del mundo y uno de los líderes en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

Figura 6.7: MySQL

MySQL [58] [59], como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiendo su interacción con los lenguajes de programación más utilizados como PHP, Perl, Java, JavaScript y su integración en distintos sistemas operativos.

El lado servidor del sistema recomendador cuenta con una base de datos bajo este sistema gestor, la cual alberga información (de puntos geográficos de destinos de senderismo y de hoteles y otros alojamientos) a la que el código del servicio web escrito en Java accede para poder realizar sus cálculos y retornar los resultados deseados.

6.2.6. PHP

PHP [60] es un lenguaje de programación abierto empleado para el desarrollo web.

Figura 6.8: PHP

Está hecho para ser ejecutado desde servidores y generar código HTML que pueda ser enviado al lado cliente. La herramienta reseñada a continuación: phpMyAdmin, está escrita en código de dicho lenguaje.

6.2.7. phpMyAdmin

phpMyAdmin [61] es un una herramienta de software libre desarrollada en PHP pensada para facilitar a administradores una interfaz visual por medio de navegadores web con la que gestionar sus bases de datos MySQL.

Figura 6.9: phpMyAdmin

Fue usada con frecuencia para la configuración y supervisión de la base de datos MySQL de la aplicación.

6.2.8. Open Source Routing Machine 5

Open Source Routing Machine [62] [63], abreviado como OSRM, es una implementación de código abierto en C++ de un motor de alto rendimiento para el cálculo de cáminos mínimos en redes viales de mapas mundiales. Combina algoritmos de cálculo de rutas sofisticados con datos de carreteras de libre acceso provenientes del proyecto OpenStreetMap.

Figura 6.10: OSRM

El lado servidor del sistema recomendador tiene un servidor local de esta herramienta, la cual permite calcular y retornar en formato JSON matrices de distancia reales (en términos de tiempo de viaje) entre uno o más puntos geográficos. La aplicación web también utiliza este servidor local para calcular el camino óptimo entre los puntos de la solución de un itinerario.

6.2.9. Eclipse Mars 4.5

Eclipse Mars [64] [65] es una de las ediciones del entorno de desarrollo integrado (del inglés Integrated Development Environment o IDE) de código abierto Eclipse. Con soporte para Java y otros lenguajes de programación, y una gran cantidad de herramientas y facilidades para el programador a la hora de escribir el código fuente de sus proyectos de software.

Figura 6.11: Eclipse IDE

Entre algunas de sus características se pueden destacar:

- Integración con Maven.
- Integración con Git.
- Herramientas de desarrollo exclusivas para Java.
- Recomendadores de código para desarrolladores de Java.
- Editores de y herramientas para código XML.
- Posibilidad de instalar otros plugins externos para extender sus funcionalidades.

Todas ellas fueran aprovechadas para el desarrollo y mantenimiento del lado servidor del sistema recomendador.

6.2.10. Doxygen

Doxygen [66] es la herramienta standard de facto para la generación de documentación de diversos lenguajes de programación como C, C++, PHp, Java, entre otros.

Figura 6.12: Doxygen

Con esta herramienta se creó documentación HTML y Latex automatizada del código Java escrito en el lado del servidor.

6.3. Lado del cliente

6.3.1. Apache HTTP Server

El Apache HTTP Server [67] es un servidor HTTP de código abierto compatible con diversos sistemas operativos.

Figura 6.13: Apache HTTP Server

La versión 2.0 de esta plataforma fue la elegida como el servidor en el que se montó la aplicación web tanto en el equipo de desarrollo (como parte de Wamp) como en el de despliegue. Es fruto del proyecto del mismo nombre de la Apache Software Foundation, al igual que muchas de las herramientas listadas en este capítulo.

6.3.2. Navegadores web

Los navegadores web [68] son programas que permiten abrir páginas web, leyendo e interpretando su código HTML, CSS y JavaScript. Los más usados en la actualidad son Chrome, Safari, Firefox e Internet Explorer [69].

Figura 6.14: Principales navegadores web (Safari, Chrome, Firefox e IE)

Las aplicaciones web se abren a través de estas herramientas indispensables. La aplicación cliente del sistema recomendador fue probada con frecuencia con los navegadores Google Chrome, Mozilla Firefox y Microsoft Edge. Aparte de algunas diferencias leves en la presentación de algunos elementos de la misma, la aplicación fue usable y funcionó sin problemas en los tres.

6.3.3. HTML5

HTML5 (HyperText Markup Language, versión 5) [70] [71] es la quinta y más reciente edición del lenguaje básico de la World Wide Web: HTML.

Figura 6.15: HTML5

El HTML es un lenguaje de marcado para la elaboración de páginas web. Es un estándar que sirve de referencia del software que conecta con la elaboración de páginas web en sus diferentes versiones. Define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, videos, juegos, entre otros. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web.

Al ser el lado cliente del sistema recomendador una aplicación web, este fue junto a CSS uno de los lenguajes básicos a emplear.

6.3.4. CSS3

El CSS (Cascading Style Sheets) [72] [73] , es el lenguaje de diseño para definir y crear la presentación visual de documentos HTML, XHTML y XML. CSS 3 es el nombre de la versión más reciente existente hasta el momento.

Figura 6.16: CSS 3 [74]

Al igual que el HTML, ha sido creado y estandarizado por el World Wide Web Consortium (W3C).

6.3.5. Javascript

JavaScript [75] [76], un lenguaje de programación interpretado, implementado como parte de los navegadores web del lado del cliente o el servidor, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo, Java y JavaScript tienen semánticas y propósitos diferentes.

Figura 6.17: JavaScript [77]

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). Todas las librerías usadas para el desarrollo de la aplicación cliente del sistema recomendador están escritas en este lenguaje.

6.3.6. jQuery

jQuery [78] [79] es una librería de JavaScript que simplifica notablemente la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web gracias a una API fácil de usar compatible con una gran cantidad de navegadores.

Figura 6.18: jQuery

Como otras similares del mismo lenguaje, es una librería libre y de código abierto que facilita a los desarrolladores varias funcionalidades que de otra forma serían más complejas y complicadas de implementar.

6.3.7. Bootstrap

Bootstrap [80] es un framework de código abierto para el desarrollo de aplicaciones en HTML, CSS y JavaScript, cuyo principal atractivo son sus series de plantillas y estilos visuales que se adaptan automáticamente a las dimensiones de la pantalla del dispositivo en que sea abierta

Figura 6.19: Bootstrap

Ha sido adoptado por una gran cantidad de sitios y aplicaciones web importantes en el mundo (tales como Twitter).

6.3.8. Backbone.js

Backbone.js [81] es un framework que permite dar a una aplicación en JavaScript una estructura basada en el patrón de diseño de software Modelo Vista Controlador (MVC) mediante la representación de datos como modelos (que a la vez se pueden agrupar en colecciones), y vistas gestionadas por controladores de eventos que los representan.

Figura 6.20: Backbone.js

Con este framework se logró otorgar una arquitectura robusta y bien definida a la aplicación cliente del sistema recomendador.

6.3.9. Underscore.js

Underscore.js [82] es una librería de JavaScript que brinda un gran abanico de auxiliares de programación sin tener que extender ningún objeto incorproado dentro del lenguaje. Está pensado para complementar a jQuery y Backbone.js.

Figura 6.21: Underscore.js

Es desarrollada y mantenida por los mismos creadores de Backbone.js

6.3.10. Leaflet

Leaflet [83] es una librería JavaScript de código abierto para la representación de mapas geográficos interactivos.

Figura 6.22: Leaflet

Es compatible tanto para navegadores de escritorio como para dispositivos móviles y cuenta con una gran cantidad de plugins para extender sus funcionalidades. Fue fundamental para la representación del mapa y los marcadores de destinos de senderismo y alojamientos en la aplicación cliente del sistema recomendador.

6.3.11. Leaflet Routing Machine

Leaflet Routing Machine [84] es un plugin de código abierto para la librería Leaflet que permite representar rutas por carreteras entre dos o más puntos en un mapa. Es compatible con una gran cantidad de motores de cálculo de rutas georreferenciadas (entre los que se encuentra OSRM, su motor por defecto) y tiene gran cantidad de opciones y funcionalidades que lo vuelven muy versátil y personalizable.

Figura 6.23: Leaflet Routing Machine

Junto a Leaflet, es una de las herramientas claves usadas en la aplicación cliente, ya que con ella se lograron describir en pantalla las rutas a tomar para moverse entre los puntos de interés de los itinerarios.

6.3.12. Notepad++

Notepad++ [85] es un editor de código abierto con soporte para diversos lenguajes de programación, entre los que se encuentra HTML, CSS y JavaScript.

Figura 6.24: Notepad++

Escrito en C++ y usando una API de Win32 puro, está pensando como una alternativa libre y más potente para el Notepad tradicional incorporado por defecto en los sistemas operativos Windows. Con este editor se escribió y gestionó el código fuente de todo el lado cliente del sistema recomendador.

6.3.13. JSDoc

JSDoc [86] es un generador de documentación para código fuente JavaScript.

Figura 6.25: JSDoc

Cumpliendo el mismo papel que Doxygen en el lado del cliente, esta herramienta se encargó de crear la documentación HTML del código JavaScript más relevante de la aplicación web.

6.4. Observaciones

En conjunto, todas estas tecnologías y herramientas de software permitieron desarrollar el proyecto con éxito. Se consultaron diversas fuentes y expertos para su elección, entre los que se puede destacar el equipo de desarrolladores de StartDevs [87], una StartUp tinerfeña conformada por egresados de la Universidad de La Laguna, con experiencia en la creación de aplicaciones y soluciones de software con funcionalidades de georreferenciación.

Capítulo 7

Conclusiones

En este capítulo se describen las conclusiones posteriores a la realización del Trabajo Fin de Grado, a fin de revisar los objetivos que fueron establecidos en los distintos apartados durante su realización y para comprobar el grado de satisfacción de cada uno de ellos. Tras este análisis, se introducen algunas líneas futuras para otros posibles Trabajos de Fin de Grado, en los que se podría mejorar o extender las funcionalidades del sistema recomendador resultante del trabajo presente.

Este sistema ha consistido en una aplicación cliente-servidor con una interfaz de usuario en el lado cliente intuitiva y fácil de usar. Esta aplicación web cliente, bautizada con el nombre de Recomendador de Rutas de Senderismo, permite al usuario obtener recomendaciones optimizadas de itinerarios de rutas de senderismo para distintos días de estadía, con representación visual sobre un mapa interactivo mediante el rellenado y envío de un formulario con parámetros adaptados a sus necesidades o criterios particulares.

7.1. Conclusiones

En el capítulo 1 de este Trabajo de Fin de Grado se expuso una serie de objetivos, cuyos resultados y conclusiones sobre su cumplimiento son los siguientes:

- Se ha llevado a cabo una recopilación lo suficientemente completa y concisa de la historia y evolución de los sistemas recomendadores de turismo, destacando además el estado del arte en cuanto a herramientas de ese tipo especializadas en el ámbito del turismo rural y de montaña.
- Se han elegido y definido un modelo matemático (TOPTW) y uno heurístico (GRASP) funcionales para la generación de itinerarios de rutas de actividades de turismo rural y de montaña, ajustados a las necesidades del turista (el usuario).

- Se ha empleado una muestra de datos reales y fiables de rutas de senderismo (Webtenerife) y alojamientos en Tenerife (Open Data Canarias), con las cuales se han obtenido resultados satisfactorios al probarlos en el sistema recomendador desarrollado.
- Se ha conseguido crear una aplicación web funcional y de interfaz agradable e intuitiva disponible en la VPN de la Universidad de La Laguna, la cual se conecta a un servidor local que maneja datos georreferenciados, siendo capaz de retornar información procesada para la representación visual de itinerarios de rutas de senderismo en un mapa interactivo.
- Se ha documentado el código desarrollado tanto en su lado servidor como cliente, y a partir del mismo y del resto de las labores realizadas durante el Trabajo de Fin de Grado, se ha podido crear esta memoria final.

Por tanto, como se han logrado satisfacer todos los objetivos propuestos, se puede concluir de que este Trabajo de Fin Grado consiguió terminarse con éxito.

En su realización fue necesario estudiar y aprender a usar y configurar herramientas tales como Jersey, Tomcat, OSRM y varias librerías JavaScript como Backbone.js, Leaflet y Leaflet Routing Machine, con la que no se contaba con experiencia previa. Igualmente, se profundizó en otras ya conocidas como el lenguaje de programación Java, HTML, CSS, JavaScript básico y MySQL.

Como causa de la falta de experiencia mencionada con algunas herramientas, se presentaron algunas dificultades: tales como la creación y puesta en marcha de servicios web basados en Java con Jersey y Tomcat, la conexión con el motor de cálculo de rutas georreferenciadas OSRM, la representación visual de las rutas en la aplicación web y la personalización de la estructura Modelo Vista Controlador de esta establecida con la librería Backbone.js .

En este Trabajo de Fin de Grado se pudo apreciar la potencia y el abanico de posibilidades que la tecnología de los sistemas recomendadores tienen dentro del sector del turismo, especialmente en una región donde el mismo es el motor económico principal como Canarias, por lo que sería ideal continuar investigaciones sobre la misma en el futuro para perfeccionarla y extender sus funcionalidades. Construir un sistema de tales características fue una labor bastante enriquecedora e interesante; la experiencia y conocimientos aportados por la misma han sido invaluables para la formación profesional del autor.

7.2. Líneas de trabajo futuro

Como colofón a este capítulo, se enseña una lista de líneas de trabajo futuro para el sistema recomendador resultante de este Trabajo de Fin de Grado. Estas consisten en nuevas funcionalidades y características que podrían estudiarse e

implementarse para mejorar así el trabajo realizado durante este proyecto:

- **Integración con TITSA:** durante el desarrollo del trabajo se estudió la posibilidad de indicar dentro de la representación visual del itinerario guaguas de Transportes interburbanos de Tenerife (TITSA, la principal entidad de transporte público de la provincia de Santa Cruz de Tenerife) que el usuario puede tomar desde cierto punto para llegar al siguiente de su ruta sugerida. Debido a su complejidad y al poco tiempo restante en el momento de su sugerencia, se decidió no implementarla en este proyecto.
- **Más recomendaciones a sugerir:** se podrían recomendar al turista destinos relativos a actividades como ciclismo y restaurantes además de las de senderismo. Habría que modificar y perfeccionar la heurística asociada a la sugerencia de las rutas o jugar con las puntuaciones de algunos de los puntos de interés para establecer tiempos razonables entre estas actividades (sobretodo las horas para ir a los restaurantes).
- **Desarrollo de una interfaz gráfica acorde a los requisitos funcionales actuales optimizada para dispositivos móviles**: si bien la aplicación se ve bien en ordenadores y tablets, su interfaz en dispositivos móviles es mejorable. Se podría estudiar la inclusión de clases y estilos Responsive a la aplicación web que se adapten mejor a esas dimensiones de pantalla reducidas.

  De la aplicación web creada podría generarse una versión para dispositivos móviles Android y iOS a través de herramientas tales como Apache Cordova.
- **Añadir más idiomas:** al ser aplicación dirigida al turismo, y como gran parte de los turistas de la isla son extranjeros, sería oportuno añadir la opción de traducir la interfaz y los resultados de la aplicación a otros idiomas como el Inglés y el Alemán.

Capítulo 8

Summary and Conclusions

This chapter describes the conclusions following the completion of the Final Degree Project, in order to review the objectives that were established in the different sections during their implementation and to check the degree of satisfaction of each one of them. Following this analysis, some future lines for other possible or potential Final Degree Projects are introduced, in which the functionalities of the recommender system resulting from this project could be extended or improved.

This recommender system is based on a client - server application with an intuitive and easy to use user interface on the client side. This one, named "Recomendador de Rutas de Senderismo", allows the user to obtain optimized hiking trails itineraries recommendations with visual representations in an interactive map for different days of stay, by filling and sending a form with parameters adapted to their needs or particular criteria.

8.1. Conclusions

The first chapter of this Final Degree Project set out a series of objectives, these are their results and conclusions:

- A sufficiently complete and concise compilation of the history and evolution of the tourism recommender systems has been carried out, also emphasizing the state of the art in such tools specialized in the field of rural and mountain tourism.
- A functional mathematical model (TOPTW) and heuristic (GRASP) have been chosen and defined for the generation of itineraries of rural and mountain tourism activities routes, adjusted to the needs of the tourist (the user).
- A sample of real and reliable data of hiking trails (Webtenerife) and lod-

gings in Tenerife (Open Data Canary Islands) have been used, with which satisfactory results have been obtained when tested in the developed tourism recommender system.

- It has been possible to create a functional and intuitive web interface application available in the VPN of the "Universidad de La Laguna", which connects to a local server that handles georeferenced data, being able to return processed information for the visual representation of hiking trails itineraries on an interactive map.
- The source code developed on both the server and client side has been documented, and from this and the rest of the work done during the Final Degree Project, it was possible to write this final paper.

Therefore, as all the proposed objectives have been met, it can be concluded that this Final Degree Project was successfully accomplished.

In its realization it was necessary to study and learn how to use and configure tools such as Jersey, Tomcat, OSRM and several JavaScript libraries like Backbone.js, Leaflet and Leaflet Routing Machine, with which there was no previous experience. Also, it was deepened in others already known like the Java programming language, HTML, CSS, basic JavaScript and MySQL.

As a result of the lack of experience mentioned with some tools, there were some difficulties: such as the creation and start-up of Java-based web services with Jersey and Tomcat, the connection to the OSRM route engine, the visual represensation of the routes in the web application and the customization of the Model View Controller structure established with the Backbone.js library.

In this Final Degree Project the power and the range of possibilities that the technology of the recommender systems have within the tourism sector could be appreciated, especially in a region where it is the main economic activity such as the Canary Islands, for which it would be ideal to continue research on the same to perfect it and extend its functionalities. Building a system of such characteristics was a rather enriching and interesting labor; The experience and knowledge provided by it have been invaluable for the professional training of the author.

8.2. Future lines of work

To conclude this chapter, a list of future lines is presented for the recommender system resulting from this Final Degree Project. These are new features and characteristics that could be studied and implemented to enhance the work done during this project:

- **TITSA integration**: during the development of the project, it was con-

sidered the possibility of indicating buses or "guaguas" of "Transportes interurbanos de Tenerife S.A." (TITSA, the main public transport entity of the province of Santa Cruz de Tenerife) within the visual representation of the itineraries, that the user can take from a certain point to get to the next of his suggested route. Due to its complexity and the short time remaining at the moment of its suggestion, it was decided to not implement it in this project.

- **More recommendations to suggest:** recommendations with tourism destinations relative to activities such as cycling and restaurants could be suggested to the user along those of hiking. It would be necessary to modify and refine the heuristic associated with the suggestion of the routes or to play with the point of interest scores to establish reasonable times between these activities (above all, the hours to visit and spend time in the restaurants).
- **Development of a graphic interface along the current functional requirements optimized for mobile devices:** while the web application looks good on computers and tablets, its graphic interface on mobile devices could be improved. The inclusion of more proper Responsive classes and styles to the web application to fit those reduced screen dimensions could be considered.

  From the developed web application, a version for mobile Android and iOS devices could be generated through tools such as Apache Cordova.
- **More languages to choose:** as it is an application aimed at tourism, and since a large part of the tourists on the island of Tenerife are foreigners, it would be appropiate to add the option of translating the interface and the results of the application into other languages such as English and German.

Capítulo 9

Presupuesto

En este capítulo se realiza una estimación del presupuesto del proyecto, en términos de las horas dedicadas y de los recursos utilizados para su elaboración.

9.1. Presupuesto del trabajo realizado

9.1.1. Tiempo invertido

Actividad	**Duración (horas)**	**Precio por hora (€)**	**Coste (€)**
Análisis de requisitos	60	20	1200
Diseño	40	20	800
Desarrollo	140	15	2100
Evaluación y pruebas	20	15	300
Despliegue	20	15	300
TOTAL	300	-	4700

Tabla 9.1: Presupuesto del tiempo invertido

9.1.2. Hardware utilizado

Dispositivo	**Descripción**	**Coste (€)**
TOSHIBA SATELLITE C50D	Ordenador portátil	365

Tabla 9.2: Presupuesto del hardware utilizado

Como ya se disponía del ordenador antes de la realización del Trabajo de Fin de Grado, su coste estimado no se suma al presupuesto del tiempo invertido.

9.1.3. Software utilizado

Software	**Coste (€)**
GitHub	0
WampServer	0
Java SE Development Kit 8	0
Apache Maven	0
Jersey RESTful Web Services framework	0
Apache Tomcat 8.0	0
MySQL	0
phpMyAdmin	0
Open Source Routing Machine 5	0
Eclipse Mars 4.5	0
Doxygen	0
Apache HTTP Server	0
Mozilla Firefox, Google Chrome, Microsoft Edge	0
Bootstrap	0
Backbone.js	0
Underscore.js	0
Leaflet	0
Leaflet Routing Machine	0
Notepad++	0
JSDoc	0
Windows 10	135

Tabla 9.3: Presupuesto del software utilizado

El sistema operativo Windows 10 ya venía incluido dentro del ordenador utilizado, por lo que al igual que el del mismo, su coste no afecta al presupuesto real del tiempo invertido.

Referencias

[1] UNWTO, *UNWTO Tourism Highlights, 2016 Edition*, 2016.

[2] Instituto de Estudios Turicos - página web oficial [online]. 2017 Disponible en `http://www.turismodecanarias.com/promoturturismocanarias/` [Consulta: 29 de Mayo del 2017].

[3] Promotur Turismo de Canarias - página web oficial [online]. 2017 Disponible en `http://www.turismodecanarias.com/promoturturismocanarias/` [Consulta: 29 de Mayo del 2017].

[4] D. Gavalas, C. Konstantopoulos, K. Mastakas and G. Pantziou, "A Survey on Algorithmic Approaches for Solving Tourist Trip Design Problems". Journal of Heuristics 20(3), 2014, 291–328.

[5] M. Resende and C. Ribeiro, *GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURES*. 1ra ed. Kluwer Academic Publishers, 2002.

[6] J.M. Godart, "Combinatorial optimisation based decision support system for trip planning" in *International conference on information and communication technologies in tourism*, Austria, 1999, pp. 318–327.

[7] J.M. Godart, "Sightseeing tour planning based on combinatorial optimization: a tool for hotel marketing", *Int. J. of Hospitality Information Technology*, 1(2), 21–33, 2000.

[8] J.M. Godart, "Using the trip planning problem for computer-assisted customization of sightseeing tours" in *International conference on information technology and tourism*, Austria, 2001, pp. 377–386.

[9] J.M. Godart, "Combinatorial optimisation for trip planning" *JORBEL*, 41, 59–68, 2001.

[10] J.M. Godart, "Beyond the trip planning problem for effective computer-assisted customization", *Inform. & comm. technologies in tourism*, Austria, pp. 163–172, 2003.

[11] J.M. Godart, "Challenges in real world sightseeing tour optimization using meta-heuristics" in *EC'05 Proceedings of the 6th WSEAS international conference on Evolutionary computing*, 2005, pp. 233–238.

[12] P. Vansteenwegen, "Planning in Tourism and Public Transportation - At-

traction Selection by Means of a Personalised Electronic Tourist Guide and Train Transfer Scheduling", PhD thesis, Katholieke Universiteit Leuven, 2008.

[13] P. Vansteenwegen, W. Souffriau, G. V. Berghe and D. Van Oudheusden, "Iterated local search for the team orienteering problem with time windows", *Computers & Operations Research*, 36:3281–3290, 2009.

[14] A. Garcia, M. T. Linaza, O. Arbelaitz and P. Vansteenwegen, "Intelligent routing system for a personalised electronic tourist guide", *Information and Communication Technologies in Tourism 2009*, 185–197, 2009.

[15] W. Souffriau and P. Vansteenwegen, "Tourist Trip Planning Functionalities: State-of-the-Art and Future" in *International Conference on Web Engineering, ICWE10*, 2010.

[16] P. Vansteenwegen, W. Souffriau, G. V. Berghe and D. Van Oudheusden, "The City Trip Planner: An expert system for tourists", *Expert Systems with Applications*, 38 (2011) 6540–6546 [`www.citytripplanner.com`], 2011.

[17] W. Souffriau and P. Vansteenwegen, "Trip Planning Functionalities: State-of-the-Art and Future". *Information Technology & Tourism*, 12(4), 305–315, 2011.

[18] P. Vansteenwegen, W. Souffriau and D. Van Oudheusden, "The orienteering problem: A survey", *EJOR*, 209(1), 1–10, 2011.

[19] A. Garcia, P. Vansteenwegen, O. Arbelaitz, W. Souffriau and M. T. Linaza, *Integrating public transportation in personalised electronic tourist guides. C&O.R.*, 40(3), 758–774, 2013.

[20] M. Kenteris, D. Gavalas and D. Economou, "An innovative mobile electronic tourist guide application", *Personal and Ubiquitous Computing*, 13:103–118, 2009.

[21] M. Kenteris, D. Gavalas and D. Economou, "Electronic mobile guides: a survey", *Personal and Ubiquitous Computing*, 15:97–111, 2011.

[22] D. Gavalas, M. Kenteris, C. Konstantopoulos and G. Pantziou, "Web application for recommending personalised mobile tourist routes", *IET Software*, 6(4), 313–322, 2012.

[23] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou and N. Vathis, "Efficient heuristics for the time dependent team orienteering problem with time windows", *Comp. & O.R.*, 62, 152–163, 2014.

[24] K.H. Lim, "Recommending and Planning Trip Itineraries for Individual Travellers and Groups of Tourists" in *ICAPS Conference 2016*, 2016.

[25] S. Ribeiro, "Design and Evaluation of a Heuristic Algorithm for Recommending Travel Regions", Master s Thesis, Department of Computer Scien-

ce, Technical University of Munich, Germany, 2016.

[26] W. Wörndl and A. Hefele, "Generating Paths Through Discovered Places-of-Interests for City Trip Planning" in *ENTER 2016 eTourism Conference*, Bilbao, Spain, 2016.

[27] W. Wörndl, *Solving Tourist Trip Design Problems from a Users Perspective*, 2016.

[28] Visit A City: Create Your Personal Travel Guide - página web oficial, [online]. 2017 Disponible en `http://www.visitacity.com/` [Consulta: 30 de Mayo del 2017].

[29] Recommender System - página web oficial, [online]. 2017 Disponible en `Citytrip.traveller-world.com` [Consulta: 19 de Febrero del 2017].

[30] Oruxmaps - página web oficial [online]. 2017 Disponible en `Oruxmaps.com` [Consulta: 19 de Febrero del 2017].

[31] Wikiloc - Rutas Y Puntos De Interés GPS Del Mundo, *Página web oficial* [online]. 2017 Disponible en `es.wikiloc.com` [Consulta: 19 de Febrero del 2017].

[32] Resource For Hiking Trails And Hiking Maps — Mapmyhike - página web oficial [online]. 2017 Disponible en `Mapmyhike.com` [Consulta: 19 de Febrero del 2017].

[33] Outdoor Guides — Hiking, Camping, Trail Running, Dog Friendly Trails — Alltrails.Com - página web oficial [online]. 2017 Disponible en `AllTrails.com` [Consulta: 19 de Febrero del 2017].

[34] Ramblr - página web oficial [online]. 2017 Disponible en `Ramblr.com` [Consulta: 19 de Febrero del 2017].

[35] Cleveralgorithms.com, *Greedy Randomized Adaptive Search - Clever Algorithms: Nature-Inspired Programming Recipes* [online]. 2015 Disponible en `http://www.cleveralgorithms.com/nature-inspired/stochastic/grasp.html` [Consulta: 19 Febrero del 2017].

[36] A. Expósito, *Team Orienteering Problem with Time Windows*. Universidad de La Laguna, 2016.

[37] Webtenerife - página web oficial [online]. 2017 Disponible en `http://www.webtenerife.com/` [Consulta: 18 de Mayo del 2017].

[38] Open Data Canarias, *Alojamientos - JSON* [online]. 2015 Disponible en `http://opendatacanarias.es/datos/dataset/tdt-alojamientos/resource/84c1f174-92a8-4c4d-a631-9e322d895ca6` [Consulta: 18 de Mayo del 2017].

[39] Technopedia, *What is the Client-Server Model? - Definition from Technopedia* [online]. 2017 Disponible en `https://www.techopedia.com/`

definition/18321/client-server-model [Consulta: 07 de Mayo del 2017].

[40] Alex Rodríguez, "RESTful Web services: The basics", *IBM* [online]. 2015 Disponible en https://www.ibm.com/developerworks/library/ws-restful/ [Consulta: 07 de Mayo del 2017].

[41] StackOverflow, *simplest way to read JSON from a URL in Java* [online]. 2010 Disponible en: http://stackoverflow.com/questions/4308554/simplest-way-to-read-json-from-a-url-in-java [Consulta: 13 de Mayo del 2017].

[42] M. A. Alvz, "Qu MVC" [online]. 2014 Disponible en https://desarrolloweb.com/articulos/que-es-mvc.html [Consulta: 30 de Mayo del 2017].

[43] J. Brito, A. Expósito and J. A. Moreno, "Solving the Team Orienteering Problem with fuzzy scores and constraints,"2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), Vancouver, BC, 2016, pp. 1614-1620.

[44] StackOverflow, *JavaScript seconds to time string with format hh:mm:ss* [online]. 2011 Disponible en: http://stackoverflow.com/questions/6312993/javascript-seconds-to-time-string-with-format-hhmmss [Consulta: 13 de Mayo del 2017].

[45] Universidad de La Laguna, *Plataformas de soporte para docencia, investigación y al puesto de trabajo* [online]. 2016 Disponible en https://www.ull.es/servicios/stic/category/iaas/ [Consulta: 18 de Mayo del 2017].

[46] Es.wikipedia.org, *LAMP* [online]. 2017 Disponible en https://es.wikipedia.org/wiki/LAMP [Consulta: 18 de Mayo del 2017].

[47] Universidad de La Laguna. (2016). *Servicio de VPN de la ULL.* [online] Disponible en https://www.ull.es/servicios/stic/tag/vpn/ [Consulta: 18 de Mayo del 2017].

[48] GitHub - página web oficial [online]. 2017 Disponible en: https://github.com/ [Consulta: 05 De Mayo del 2017].

[49] Es.wikipedia.org, *GitHub* [online]. 2017 Disponible en: https://es.wikipedia.org/wiki/GitHub [Consulta: 05 De Mayo del 2017].

[50] WampServer - página web oficial [online]. 2017 Disponible en: http://www.wampserver.com/en/ [Consulta: 06 De Mayo del 2017].

[51] Oracle, *Java Programming Language* [online]. 2017 Disponible en http://docs.oracle.com/javase/8/docs/technotes/guides/language/index.html [Consulta: 03 de Mayo del 2017].

[52] Oracle, *Java SE Development Kit 8 - Downloads* [online]. 2017 Disponi-

ble en `http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html` [Consulta: 03 de Mayo del 2017].

[53] En.wikipedia.org, *Java (programming language)* [online]. 2017 Disponible en `https://en.wikipedia.org/wiki/Java_(programming_language)` [Consulta: 03 de Mayo del 2017]

[54] Apache Maven Project - página web oficial [online]. 2017 Disponible en `https://maven.apache.org/` [Consulta: 04 de Mayo del 2017].

[55] Jersey - página web oficial [online]. 2017 Disponible en `https://jersey.java.net/` [Consulta: 04 de Mayo del 2017].

[56] Apache Tomcat - página web oficial [online]. 2017 Disponible en `http://tomcat.apache.org/` [Consulta: 04 de Mayo del 2017].

[57] En.wikipedia.org, *Apache Tomcat* [online]. 2017 Disponible en `https://en.wikipedia.org/wiki/Apache_Tomcat` [Consulta: 04 de Mayo del 2017].

[58] MySQL - página web oficial [online]. 2017 Disponible en `https://www.mysql.com/` [Consulta: 04 de Mayo del 2017].

[59] Es.wikipedia.org, *MySQL* [online]. 2016 Disponible en: `https://es.wikipedia.org/wiki/MySQL` [Consulta: 04 De Mayo del 2017].

[60] PHP. (2017). *Qué es PHP?*. [online] Disponible en `http://php.net/manual/es/intro-whatis.php` [Consulta: 19 de Mayo del 2017].

[61] phpMyAdmin - página web oficial [online]. 2017 Disponible en `https://www.phpmyadmin.net/` [Consulta: 19 de Mayo del 2017].

[62] Project OSRM. *Página web oficial*. [online] Disponible en `http://project-osrm.org/` [Consulta: 04 de Mayo del 2017].

[63] En.wikipedia.org, *Open Source Routing Machine* [online]. 2017 Disponible en `https://en.wikipedia.org/wiki/Open_Source_Routing_Machine` [Consulta: 04 de Mayo del 2017].

[64] Eclipse Fundation. (2017). *Eclipse IDE for Java Developers*. [online] Disponible en `https://eclipse.org/downloads/packages/eclipse-ide-java-developers/neon3` [Consulta: 04 de Mayo del 2017].

[65] Es.wikipedia.org, *Eclipse (software)* [online]. 2017 Disponible en `https://es.wikipedia.org/wiki/Eclipse_(software)` [Consulta: 04 de Mayo del 2017].

[66] Doxygen - página web oficial [online]. 2017 Disponible en `http://www.stack.nl/~dimitri/doxygen/` [Consulta: 23 de Mayo del 2017].

[67] Apache HTTP Server Project - página web oficial [online]. 2017 Disponible en `https://httpd.apache.org/` [Consulta: 19 de Mayo del 2017].

[68] En.wikipedia.org, *Web browser* [online]. 2017 Disponible en: `https://en.wikipedia.org/wiki/Web_browser` [Consulta: 06 de Mayo del 2017].

[69] En.wikipedia.org, *Usage share of web browsers* [online]. 2017 Disponible en: `https://en.wikipedia.org/wiki/Usage_share_of_web_browsers` [Consulta: 06 de Mayo del 2017].

[70] W3C, *HTML5* [online]. 2017 Disponible en Enlace [Consulta: 02 de Mayo del 2017].

[71] W3C - página web oficial [online]. 2017 Disponible en `http://www.w3c.es/` [Consulta: 02 de Mayo del 2017].

[72] Es.wikipedia.org, *Hojas de estilos en cascada* [online]. 2017 Disponible en `https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada` [Consulta: 03 de Mayo del 2017].

[73] W3C, *CSS* [online]. 2017 Disponible en `https://www.w3.org/TR/CSS/` [Consulta: 02 de Mayo del 2017].

[74] Brands of the world, *CSS3* [online]. 2017 Disponible en `https://www.brandsoftheworld.com/logo/css3` [Consulta: 02 de Mayo del 2017].

[75] Es.wikipedia.org, *JavaScript* [online]. 2017 Disponible en: `https://es.wikipedia.org/wiki/JavaScript` [Consulta: 03 De Mayo del 2017].

[76] JavaScript - página web oficial [online]. 2017 Disponible en `https://www.javascript.com/` [Consulta: 03 de Mayo del 2017].

[77] Brands of the world, *JavaScript* [online]. 2017 Disponible en `https://www.brandsoftheworld.com/logo/javascript` [Consulta: 03 de Mayo del 2017].

[78] jQuery - página web oficial [online]. 2017 Disponible en: `https://jquery.com/` [Consulta: 05 De Mayo del 2017].

[79] jQuery Foundation - página web oficial [online]. 2017 Disponible en: `https://jquery.org/` [Consulta: 05 De Mayo del 2017].

[80] Bootstrap - página web oficial [online]. 2017 Disponible en: `http://getbootstrap.com/` [Consulta: 05 De Mayo del 2017].

[81] Backbone.js - página web oficial [online]. 2017 Disponible en: `http://backbonejs.org/` [Consulta: 05 De Mayo del 2017].

[82] Underscore.js - página web oficial [online]. 2017 Disponible en: `http://underscorejs.org/` [Consulta: 06 De Mayo del 2017].

[83] Leaflet - página web oficial [online]. 2017 Disponible en: `http://leafletjs.com/` [Consulta: 05 De Mayo del 2017].

[84] Leaflet Routing Machine - página web oficial [online]. 2017 Disponible en: `http://www.liedman.net/leaflet-routing-machine/` [Consulta: 05 De

Mayo del 2017].

[85] Notepad++ - página web oficial [online]. 2017 Disponible en: `https://notepad-plus-plus.org/` [Consulta: 05 De Mayo del 2017].

[86] Use JSDoc - página web oficial [online]. 2017 Disponible en `http://usejsdoc.org/index.html` [Consulta: 23 de Mayo del 2017].

[87] StartDevs - página web oficial [online]. 2017 Disponible en: `https://startdevs.com/` [Consulta: 14 de Mayo del 2017].

[88] PC Componentes, *Toshiba Satellite C50D AMD E1-2100/4GB/500GB/15.6* [online]. 2017 Disponible en `https://www.pccomponentes.com/toshiba-satellite-c50d-amd-e1-2100-4gb-500gb-15-6-` [Consulta: 21 de Mayo del 2017].

[89] Microsoft Store, *Windows 10 Home* [online]. 2017 Disponible en `https://www.microsoftstore.com/store/mseea/es_ES/pdp/productID.320437800?VID=320438100&s_kwcid=AL!4249!3!139726077995!!!g!150255702304!&WT.mc_id=es_datafeed_pla_google_pointitsem_windows&ef_id=V6zfAQAAACTTtQZN:20170520234726:s` [Consulta: 21 de Mayo del 2017].