Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Trabajo de Fin de Grado

SISTEMA DE RECOMENDACIÓN DE ITINERARIOS ÓPTIMOS DE RECURSOS TURÍSTICOS COSTEROS Y MARÍTIMOS

SYSTEM OF RECOMMENDATION OF OPTIMAL ITINERARIES OF COASTAL AND MARITIME TOURIST RESOURCES

Autor: Jacobo Rodicio González

Tutorizado por:

Dagoberto Castellanos Nieves
Julio Antonio Brito Santana

La Laguna, 5 de septiembre de 2017

D. **Dagoberto Castellanos Nieves**, con N.I.F. 79234766-L profesor Contratado Doctor y D. **Julio Brito Santana**, con N.I.F. 42812193-Q profesor Titular de Universidad, adscritos al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna

CERTIFICAN

Que la presente memoria titulada:

"Sistemas de Recomendación de Itinerarios Óptimos de Recursos Turísticos Costeros y Marítimos."

ha sido realizada bajo su dirección por D. **Jacobo Rodicio González** con N.I.F 44471719T

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 septiembre de 2017

Agradecimientos

A mi familia por apoyarme en todo momento, creyendo en mí y sabiendo estar ahí para todo lo que he necesitado, gracias.

A mis amigos y compañeros quiénes han sabido entenderme, ayudarme y sin duda convertir los momentos de ocio en recuerdos inolvidables, gracias.

A mis tutores, Dagoberto Castellanos Nieves y Julio Brito Santana quiénes han conseguido transmitirme su ayuda, consejo y experiencia, posibilitando la finalización de este Trabajo de Final de Grado, gracias.

Resumen

El objetivo de este trabajo ha sido la documentación, diseño y creación de un ***SISTEMA DE RECOMENDACIÓN DE ITINERARIOS ÓPTIMOS DE RECURSOS TURÍSTICOS COSTEROS Y MARÍTIMOS*** para la isla de Tenerife.

Dicho sistema será capaz de ofrecer propuestas de rutas atractivas para el cliente y adaptar dichas propuestas tanto al tiempo estipulado como a las necesidades particulares del mismo.

Para alcanzar dicho fin, se ha creado una aplicación web a través de la cuál permite al usuario introducir sus preferencias personales. En base a las características personales el sistema genera un nuevo planing adaptado.

Para lograr esta tarea, se llevará a cabo una consulta a un servidor con los parámetros establecidos. Éste, por su parte, procesará los datos de la misma y apoyándose en una base de datos y un servidor externo, calculará un itinerario óptimo.

Por último, dicho itinerario será presentado al usuario sobre un mapa interactivo de la isla de Tenerife.

Keywords

Sistemas recomendadores, rutas turísticas, optimización de rutas, recomendadores de rutas

Abstract

The main objective of this TFG was the documentation, design and creation of a ***SYSTEM OF RECOMMENDATION OF OPTIMAL ITINERARIES OF COASTAL AND MARITIME TOURIST RESOURCES*** for the island of Tenerife.

This system will be able to offer suggestions of interesting routes to the customer and it will adapt those suggestions to the time and preferences.

To reach this, a web application was created through which the client introduces his personal specifications and, taking constance of those, the system generates a new planning.

Some queries to a server will be done. This server, helped by another external one and also a database, it will calculate an optimal itinerary.

Finally, the itinerary will be presented to the customer through an interactive map of the island of Tenerife.

Keywords

Recommmending systems, tourist routes, route optimization, route recommenders

Índice general

Capítulo 1

Introductorio

1.1. Idea motriz del proyecto

El sector servicios y en especial el turismo han sido y continúan siendo el principal sustento de Tenerife. Según la OMT[1] podemos entender el turismo como aquellas actividades que las personas realizan mientras están de viaje en entornos donde no es habitual que se encuentren, cuyos fines son el ocio, los negocios u otros y duran períodos inferiores a un año. Centrando nuestra atención en dichas líneas podemos apreciar ciertos puntos clave:

- En primer lugar, aparece la idea de ***actividad***. Esto quiere decir que las personas que nos visitan acuden con la idea preconcebida de llevar a cabo ciertas tareas, tareas que, independientemente de su índole (descubrimiento de la zona, relajación, deportivas...) nosotros debemos ofertar y promover.

[1] OMT: Organización Mundial del Turismo

- Siguiendo la misma definición, se identifica otra de las apreciaciones en las que radica la idea de turismo, y es el hecho de que el turista visita una zona, se envuelve de un ***entorno desconocido*** para él. Frente a esta concepción, es nuestro deber poner en conocimiento de los mismos toda información relevante acerca de las numerosas cualidades, recursos e innumerables puntos de interés con los que cuenta la isla de Tenerife.

- Por último y no por ello menos importante, cabe destacar la noción de ***temporalidad***. Debemos adecuar nuestra actividad para ser capaces de ofrecer al turista un plan completo, intentando maximizar el tiempo de su estancia.

En líneas generales y a modo de recapitulación, debemos ser capaces de proponer a nuestros visitantes ciertas tareas en función de sus intereses, siempre dando a conocer todo el potencial y belleza de la isla, adaptando y administrando de una manera eficiente el tiempo que dure su estadía, ayudando a que puedan obtener el máximo provecho de su visita y regresar a sus hogares con una experiencia totalmente positiva y enriquecedora.

Justamente esta es la idea motriz de este proyecto, crear una plataforma capaz de solventar y aunar todos los puntos tratados. De una forma más descriptiva, la plataforma ofrecerá a los visitantes una posible planificación en función de la duración de su estancia pudiendo refinar o adaptar dicho *planning* dependiendo de los gustos o preferencias del usuario actual de la misma e incluyendo entre otros, un factor realmente clave en el turismo costero como es el tiempo deseado en cada uno de los destinos que forman parte de la programación.

Teniendo una vista global del problema y de su posible planteamiento teórico, es necesario apreciar la magnitud del mismo y con ello tomar una decisión acerca del nivel de profundización así como el ámbito general del mismo. Cabe destacar que en Tenerife la ocupación turística en la zona sur y con un fin más dedicado al turismo de playa se eleva a un 77 % del total de turistas que acuden a la isla [1], por ello, se ha acotado el problema redu-

ciendo la oferta de actividades, puntos de interés... a un contexto primariamente costero y de litoral.

1.2. Antecedentes y estado del arte

A partir de la Segunda Guerra Mundial, la ciencia de la investigación operativa [9] comenzó a aplicarse a un amplio rango de sectores más allá del militar (al que siempre había estado ligada), sin embargo, el sector turístico permaneció muchos años sin ser objetivo de estudio alguno relativo a este campo.

Esta situación perduró durante años hasta que, en 2001, Godart [10] decide orientar la optimización matemática hacia la resolución de problemas dentro de dicho sector valiéndose del Problema del viajante de comercio (TSP[2]) [11] para la planificación de viajes. En su primera aproximación Godart tenía presente una selección de actividades y alojamientos de entre los que seleccionaban los de mayor interés (POIs[3]) minimizando los costes de transporte y estadía al mismo tiempo. En una segunda aproximación, el autor extendió la funcionalidad del problema planteado, consiguiendo que el modelo también maximizara "*el atractivo*"tanto de los alojamientos como de las actividades incluidas en la solución.

Esta segunda aproximación dió lugar a la aplicación YourTour[4]. Posteriormente, en el año 2003 en Italia, Ardissono et al. [16] propone el proyecto INTRIGUE (INteractive TouRist Information GUidE).

INTRINGUE se trata de un sistema recomendador para la ciudad de Torino, Italia, capaz de mostrar una recomendación y el motivo de la misma, además de permitir la programación de tours para grupos y la posibilidad de restringir el problema mediante las preferencias del grupo.

[2]TSP: por su traducción inglesa del Problema del Viajante de Comercio: Traveller Salesman Problem

[3]POI: Point Of Interest (Punto de Interés)

[4]www.yourtour.com: Personal Tour Planner

Años después de la publicación de INTRIGUE, Castillo et al. [17] presentan un sistema para la planificación de rutas turísticas basado en un modelo multi-agente. Dicho sistema tomaba en primera instancia los intereses de los usuarios y en base a los mismos intentaba predecir las posibles actividades de interés. El *"planning"* resultado tenía en cuenta diversos factores, como por ejemplo: horarios de apertura-cierre, precios de comidas y transporte, localización...

Un año más tarde, Lee et al. [18] presenta un sistema de recomendación que permite la planificación de viajes y rutas turísticas para la ciudad de Tainan, China. Este sistema empleaba dos agentes para la resolución del problema. El primero de ellos, conocido como agente de decisión por contexto buscaba conceptos en los recursos que encajaran con las preferencias expresadas por los usuarios. Posteriormente, el segundo de los agentes, conocido como agente de recomendación de rutas de viaje utilizaba lógica difusa para la selección y ordenamiento de de un "top 3"de los lugares históricos de la ciudad para visitar así como un "top 5"de los restaurantes mejor valorados o de mayor prestigio.

Casi de una forma paralela, con publicaciones entre los años 2008 y 2012, Vansteewegen y Oudheusden [12] [13] defienden la utilización de la Investigación Operativa [9] para la resolución de Tourist Trip Design Problems (TTDP). Resultado de estos estudios surgió el desarrollo de un sistema de apoyo basado en la web que contemplaba las preferencias de los usuarios, conocido como City Trip Planner[5]. Dicho sistema [14] planeaba visitas a la ciudad para múltiples días, contemplando el concepto de ventanas de tiempo para los POI que podían variar dependiendo del día. Además permitía programar descansos para comidas proponiendo para ello nuevos POIs.

[5] http://www.citytripplanner.com

1.3. Objetivos

Con la realización de la esta memoria así como del desarrollo de la aplicación web relativa a la misma se pretenden alcanzar una serie de objetivos planteados con anterioridad a la finalización de las mismas. Principalmente se pueden desglosar las expectativas del presente Trabajo de Final de Grado en los siguientes objetivos:

- **(OB1)**: Estudio del estado del arte de los sistemas de planificación de rutas con recomendaciones.
- **(OB2)**: Definición y diseño de modelos y métodos de planificación de rutas óptimas y recomendación de itinerarios de recursos turísticos costeros.
- **(OB3)**: Validación de modelos y métodos, empleando pruebas y datos de recursos costeros.
- **(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

Contextualización dentro del sector turístico

En este capítulo se introducen conceptos importantes para situar al lector en el contexto del enfoque aplicado y el porqué de la necesidad de este trabajo. Se comienza con la descripción de la evolución del turismo ya entrada la era tecnológica y de la información de una forma general. Posteriormente se acota el planteamiento a la situación concreta de Canarias. Por último en las siguientes secciones se definen términos importantes acerca del ámbito del proyecto así como una introducción a "la forma en la que el turismo se promociona", el "Todo Incluidoz la necesidad de ir más allá e innovar.

2.1. La constante evolución del turismo

Desde sus inicios el turismo ha experimentado una progresiva evolución en cuanto a numerosos factores a lo largo de diversas etapas, teniendo cada una de ellas un nivel de progresión diferente. Para el presente documento cabe enfocar nuestra atención en la etapa más reciente de dicha progresión, en la cuál podemos

contemplar cómo las nuevas tecnologías han cambiado en gran medida tanto el turismo en sí como la concepción y pensamiento del turista a la hora de planificar sus viajes.

La tecnología ha avanzado de forma desmesurada hasta el punto en que hoy en día un smartphone[1] es capaz de realizar operaciones que hace años eran imposibles aún mediante la utilización de los ordenadores más potentes de la época.

Este hecho, unido a que el número de móviles en el mundo ya en 2016 había ascendido hasta los desorbitados 4,61 mil millones de ejemplares [3], nos obliga a tener en cuenta que cada visitante que acude a nuestro destino cuenta con, al menos, un pequeño y potente ordenador de bolsillo (comúnmente denominados móviles, smartphones...).

Teniendo claro lo citado en líneas anteriores así como la inmensa cantidad de información presente en la nube[2] y la forma en la que desde un móvil se puede acceder a la misma mediante un simple "toque", es fácil comprender cómo el turista ha cambiado y cómo lo ha hecho también la manera de planificar sus viajes. Otro punto a destacar íntimamente relacionado con la tecnología y la era de la información recae en las valoraciones que los usuarios de un cierto destino, hotel, restaurante...

Dichas valoraciones, con sus comentarios y apreciaciones constituyen un conocimiento general que queda plasmado en Internet y quedará accesible a nuevos turistas, condicionando en mayor o menor medida las elecciones que estos hagan sobre sus viajes en base a dicha ïnformación extra".

"Hoy en día los turistas consumen y comparten sus experiencias, ya sea en redes sociales, blogs o webs especializadas (TripAdvisor, Booking...). Es mucha información al alcance de todos, pudiendo comparar y elegir aquello que más se adapta a los gustos y preferencias particulares, familiares o de grupo."[4].

[1] Smartphone: Un teléfono móvil que puede ser usado como un pequeño computador capaz de conectarse a Internet.

[2] Nube: De manera simplificada, la informática en la nube es la entrega de servicios informáticos (servidores, almacenamiento, bases de datos, redes, software, análisis, etc.) a través de Internet (“la nube”).

2.2. La evolución turística en España desde un punto de vista político

La política ha influido e influye sobre la manera en que se enfoca el turismo del país así como sobre los organismos encargados de la promoción, actuación y sostenimiento del mismo.

En esta sección veremos como ha evolucionado el turismo en España dependiendo de la situación política de cada época, presentando y describiendo a *"grosso modo"* los principales organismos de gobierno creados con el fin de administrar el turismo. También, a lo largo de las siguientes subsecciones se irán describiendo los principales cambios legislativos relativos a la regulación de la actividad turística del país.

Siguiendo las pautas citadas podremos dividir dicha evolución en tres grandes etapas:

2.2.1. Nacimiento del turismo moderno en España (1900-1936)

Esta etapa [19] parte de los tiempos de la monarquía de Alfonso XII hasta la finalización de la II República en el territorio español. Durante la **Monarquía Parlamentaria (1902-1923)** [26] se llevan a cabo grandes cambios con sus consecuentes avances en la materia turística española. Podemos decir que el surgimiento de turismo moderno en dichos años surge de la visión positivista por parte de las autoridades monárquicas españolas de cómo podría repercutir en nuestras arcas la presencia, el gasto de turistas extranjeros visitando nuestras tierras.

En base a tal reflexión se decide apostar por el fomento, por el crecimiento e impulso de España como destino turístico y es, con el Real Decreto de 6 de octubre de 1905 [27] cuando se crea una Comisión Nacional, que será el departamento encargado de llevar a cabo dicha tarea y cuyos fondos necesarios serán incluidos por el Ministerio de Fomento en el Presupuesto del Estado.

A pesar de que el desconocimiento prevalece sobre cuál fue su papel y que actividades llevó a cabo exactamente la Comisión Nacional, es cierto que desde su creación la situación dentro del sector mejoró en gran medida.

Mediante el Real Decreto de 19 junio de 1911 [28] se sustituye en España la susodicha Comisión Nacional por la Comisaría Regia del Turismo y Cultura Artística la cual no centrará su actuación en la promoción de España en el extranjero sino que también dedicará gran parte de sus esfuerzos y presupuesto del estado a incrementar y cuidar el patrimonio español.

Durante la llegada de la **Dictadura de Primo de Rivera (1923-1930** la situación permanece invariable a sus inicios, sosteniendo la misma línea de actuación que se venía siguiendo desde hace varios años, también con la misma escasez de recursos para la Comisión Regia [29].

Este hecho provocó que nuevas corrientes de pensamiento y nuevas influencias surgieran con el fervor deseo de crear una institución mayor y dotada de nuevos recursos, una institución capaz de hacer frente y de aprovechar las grandes posibilidades de crecimiento de turismo español.

A medida que la situación seguía inmune al cambio dichas corrientes se hicieron fuertes y audibles incluso en círculos más prestigiosos, llegando al fin, exitosamente, por el Real Decreto de 25 de abril de 1928 [30] a la creación del Patronato Nacional del Turismo. La índole de esta nueva institución era mayor, contaba con mayores recursos tanto a nivel personal como presupuestario.

La actuación e intervención de este nuevo Patronato Nacional del Turismo fue sin duda necesaria y provechosa para el estado español, llevando a buen término la satisfactoria creación del Servicio de Crédito Hotelero, el Título de Establecimiento Recomendado o la Cámara Oficial Hostelera entre otros.

Con la proclamación de la **República (1931-1936** se llevaron a cabo desde sus inicios numerosos cambios relativos al fomento del turismo español. La institución al cargo vigente hasta el momento, el Patronato Nacional fue el punto de mira de gran-

des modificaciones, sufriendo una casi absoluta disolución con recortes de plantilla, cambios en la administración y reasignación de su presupuesto, quedando casi reemplazado por la Dirección General de Turismo. Por suerte para el primero, se recuperó la actuación del mismo y se instauró un nuevo reglamento para el Patronato por el Decreto del 12 de enero de 1932 [24].

A pesar de los numerosos cambios y reformas, no son muchos los hitos en la evolución del turismo español en tiempos de República, destacando únicamente ciertas labores de promoción y propaganda así como el impulso e incremento de la actividad concerniente a la Red de Paradores y Albergues de Carretera.

2.2.2. La dictadura de Franco (1939-1975)

La primera etapa de la dictadura (1939-1950) ha sido una de las etapas más duras de España y de su consiguiente evolución en el ámbito turístico [21]. La guerra civil trajo consigo la decadencia en lo referente al número de visitantes extranjeros así como la definitiva desaparición del Patronato Nacional. Sin embargo, aún no acabada la guerra interna que estaba sufriendo el país, las autoridades franquistas crean el Servicio Nacional de Turismo en 1938 que se convertiría un año más tarde en la Dirección General de Turismo.

A lo largo del período franquista se han llevado a cabo múltiples medidas e hitos relacionados con el establecimiento del turismo como actividad y fuente de ingresos insustituible para las arcas del país, entre ellos cabe destacar:

- La reglamentación de la publicidad con fines de propaganda turística
- La creación definitiva del Crédito Hotelero
- Instauración con fines recaudatorios de la Póliza de Turismo
- Reconocimiento del Sindicato Nacional Hostelería y Similares

- Aprobación del Reglamento Nacional de Trabajo para la industria Hotelera y de Cafés, Bares y Similares
- Aprobación del Reglamento Ordenador de los Transportes por Carretera

En el período correspondiente a la segunda parte de la dictadura (1951-1962) podemos destacar la creación del Ministerio de Información y Turismo [22] integrado por las direcciones generales de: Prensa, Información, Radiodifusión, Cinematografía y Teatro, y Turismo. Las medidas llevadas a cabo en esta etapa se centran en reglar aspectos más concretos y en controlar ciertos aspectos que se habían obviado en tiempos anteriores.

Por último, durante el transcurso de los últimos años hasta la muerte del caudillo el país necesitaba recuperarse y se optó por el intento de maximizar el crecimiento tanto de oferta como de demanda. Sin embargo, debido a problemas presupuestarios dicho intento no alcanzó demasiado éxito.

2.2.3. La monarquía de Juan Carlos I (1975-2000)

Con la restauración de la monarquía se sucede un nuevo tiempo para el sector turístico español, pudiendo diferenciarse claramente tres etapas desde el punto de vista del poder político. En primer lugar [20] el presente hasta el momento Ministerio de Información y Turismo desaparece finalmente en julio de 1977 viéndose reemplazado por la Secretaría de Estado de Turismo. Este período nos deja medidas ciertamente relevantes destacando:

- Apoyo a la exportación de capitales españoles para la realización de inversiones turísticas en el extranjero.
- Financiación de circulante a empresas turísticas exportadoras.
- Declaración de Territorios de Preferente Uso Turístico.

- Liberalización de los precios de los establecimientos hoteleros.

La segunda etapa recae sobre el socialismo español (1982-1996). Esta etapa se divide en términos turísticos en dos períodos [31]. El primero de ellos marcado por la dejación de la Administración Central y por la culminación del proceso de transferencias turísticas a las Comunidades Autónomas, siendo estas últimas las que continuaron al margen de la Administración con una línea bastante continuista y sin grandes cambios.

La segunda etapa, enmarcada desde los inicios de la década de los noventa sumerge a España en un nuevo modelo, enfocado no a la estrategia de costos que regía la actuación turística en ese momento, sino a la consolidación de un producto de calidad, diversificado, sostenible y eficiente. Es en este período cuando surgen numerosos cambios dentro de la organización administrativa turística y se suceden también ciertos pasos importantes para la consolidación de ese nuevo objetivo común de alcanzar un producto de calidad.

En el año noventa se crea el Instituto de Promoción del Turismo que aunque bajo el concepto de centralizar los cambios orientados a la mejora turística, sufrirá ciertos cambios tan sólo unos meses después, convirtiéndose en el Instituto de Turismo en España (Turespaña) que se mantendría como organismo autónomo.

El Plan Futures [23], aprobado en 1992 se presenta como el primer gran paso en la definición de una estrategia turística nacional y del inicio de la colaboración real entre Estado y Comunidades Autónomas.

Por último, se estudia la etapa de la oposición, el tiempo del partido popular en el poder y cómo este ha afectado al sector turístico, con una actividad turística “gobernada” bajo el Ministerio de Economía y Hacienda al cargo de Rodrigo Rato.

En sus inicios, se sustituye la Secretaría General de Turismo por un nuevo órgano denominado Secretaría de Estado de Comercio, Turismo y Pymes. Cabe destacar que a pesar del cambio en el poder político del país, esto no supuso un cambio en las

políticas relacionadas con el sector, sino que se continuaron las medidas iniciadas en la etapa socialista continuando dicha línea con la promulgación del II Plan Marco de Competitividad y Planes de Excelencia y Dinamización Turística.

Además de lo citado, a partir de la clausura del III Congreso Nacional de Turismo [25] celebrado en Madrid en noviembre de 1997, se presentan un conjunto de medidas de actuación para los siguientes años con la implantación de un programa de turismo sostenible.

2.3. El caso de Canarias

En lo referente a las islas, el sector turístico evolucionó casi de forma independiente con respecto al resto del país. A grandes rasgos podemos diferenciar tres etapas clave del mismo [6].

2.3.1. Período anterior a 1957

En esta época se comenzaron a construir hoteles en el Valle de La Orotava cuyo fin era no tanto el de satisfacer las necesidades turísticas comunes, sino la de servir de alojamiento a personas enfermas que acudían a las islas por sus beneficiosas condiciones climáticas.

Viendo esta situación y el consecuente incremento de las personas extranjeras, principalmente de Inglaterra, surgió la actividad turística como tal.

2.3.2. Período comprendido entre 1957 y 1990

El hecho de que lugares donde antes no había nada se convirtieran en lugares turísticos generaba cierta nivel económico distribuido entorno a los turistas, la fuente económica.

Podemos decir que se sucede el primer boom turístico durante estos años, sin embargo, con la caída del dólar y el aumento del

precio del petróleo y materias primas llega una crisis energética a las islas.

Posteriormente, en gran parte gracias a las ventajas fiscales que se otorgaron a las islas [7] y gracias al segundo boom turístico, la actividad turística sigue creciendo hasta los años 90.

Además, en esta segunda etapa se puede apreciar uno de los cambios más significativos para las Islas Canarias y es el paso de un modelo de exportación de productos agrícolas a un modelo de servicios, el turismo.

2.3.3. A partir de 1990

Durante esta última etapa el tipo de turistas que acuden a las islas cambia radicalmente debido principalmente a que los ciudadanos de las grandes ciudades necesitaban no sólo ese turismo costero sino también un contacto con el medioambiente, con la naturaleza.

Es por ello que se procedió a la ampliación de las zonas protegidas medioambientales así como la conservación del patrimonio cultural. Esto conjunto al gran clima y a las playas de las islas amplió el rango de personas [33] para las que Canarias cumplía sus requisitos y/o expectativas.

Viendo cómo la cantidad de turistas aumentaba rápidamente y temiendo por la destrucción del entorno natural en pro a la construcción de centros hoteleros se promulga una ley conocida como moratoria turística que si bien pretendía mantener una situación estable a largo plazo no consiguió más que un desarrollo aún mayor, provocando el tercer boom turístico en Canarias que en realidad sigue teniendo su impacto a día de hoy destacando el de tipo territorial y de destrucción del suelo.

2.4. Turistas del siglo XXI

De manera intrínseca a la evolución del turismo se produce un incremento de la cantidad de personas que visitan nuevos lugares

y desconectan de su vida cotidiana aprovechando la amplia oferta turística actual.

En sus inicios y durante un largo período, el turista buscaba en su gran mayoría destinos de sol y playa, sin necesidad de una gran afluencia de actividades complementarias o extra más allá de buenos restaurantes y un buen servicio y trato por parte de las empresas hoteleras.

Esta percepción ha evolucionado paralelamente en gran medida, surgiendo nuevos grupos de turistas con nuevas aficiones e intereses además de un considerable cambio en la forma en que estos buscan y planean sus vacaciones.

En la actualidad [32], la tecnología forma parte activa de dicha búsqueda y planificación, además, el turista se ha convertido en el mejor crítico, aportando valoraciones sobre los destinos visitados, hoteles, actividades... que posteriormente servirán a nuevos visitantes como baremos para su posible elección.

Esta apreciación es ampliamente conocida a día de hoy tanto por turistas como hoteleros por lo que en este apartado nos centraremos en mayor medida en los nuevos sectores o grupos de turistas que surgen con la evolución actual tanto del turismo como de la sociedad así como las principales características del turista en el período en que nos encontramos.

2.4.1. Nuevos segmentos turísticos

Siguiendo la edición de la revista Hosteltur nos encontramos una interesante publicación que se centra en *"Los nuevos perfiles de turistas"*[8].

Según su estudio, se pueden diferenciar 10 nuevos sectores con los que la industria hotelera deberá contar y hacia los cuales deberá enfocar su actividad con el fin de poder adaptarse a la evolución del turista actual. A continuación se describen brevemente cada uno de los nuevos nichos fruto de dicho estudio:

- **Turistas con mascotas**: Son muchos los turistas que viajan con sus mascotas y en consecuencia, gran parte de la planificación del viaje se centra en las mismas.

  Esto ocurre así puesto que no todos los hoteles permiten su entrada o estancia conjuntas, igualmente ocurre con muchos museos, mercados y otros posibles lugares de interés para el turista, por lo que, parece lógico que un turista que desee viajar con su/s mascota/s planifique el viaje pensando en las mismas y en como su destino turístico se puede adaptar a las mismas y no al contrario.

- **Mujeres que viajan con mujeres**: Grupos de mujeres cuyo perfil responde a edades medias de 45 años con un nivel socio-económico medio-alto quienes buscan experiencias únicas buscando conocer la realidad femenina de los destinos que visitan, su cultura, su estado... Dichos grupos acostumbran no realizar viajes aislados sino que una vez comienzan, suelen realizar diversos viajes a raíz del primero.

- **Millennials**: Término que se refiere a las nuevas generaciones inmersas en un mundo mayormente tecnológico, demandando experiencias locales y la inmersión comunicativa a través de redes de calidad. Son continuos evaluadores de sus experiencias y vivencias dejando recogida dicha información y evaluación en las principales plataformas de internet.

- **Nuevos VIP**: Sector compuesto principalmente por viajeros relativamente jóvenes, cosmopolitas y nuevos ricos, buscan ya no el trato excesivamente atento como siempre se había entendido sino que se ven informales, no quieren una atención extremadamente personal, sin embargo cuando quieren un servicio, lo quieren impoluto.

- **Turismo halal**: Se refiere a los potenciales consumidores del turismo islámico o "halal"(admisible, en árabe) que cada vez despierta más interés en la industria turística a nivel global.

- **Viajes orientados a los niños**: Cada vez es más común ver viajes realizados por toda la familia pero enfocados casi de forma exclusiva en los más pequeños.

  La industria hotelera está reaccionando ante esta demanda y es común encontrar ejemplos de hoteles temáticos en los que las principales actividades de diversión se centran en satisfacer los requerimientos de los niños, con toboganes de agua en las piscinas, playas con parques de columpios... así como la proliferación de los menús infantiles, rutas familiares, etc.

- **PANK (Professional Aunt, No Kids)**: Se refiere a una tendencia en alza en los últimos años en mayor medida en Europa y EEUU en la que mujeres profesionales sin hijos llevan de viaje a sus sobrinos.

- **Viajeras de negocios**: Referente a viajes de mujeres por causa de negocios, actualmente en lugares como EEUU se encuentran a la par que en el caso del sexo masculino. En Europa por el momento, las viajeras de negocios superan el 25 % pero se espera un continuo crecimiento del sector por lo que las empresas hoteleras deberían prepararse al respecto.

- **Singles**: Nos encontramos en un período en que casi uno de cada cuatro hombre vive solo y con dicha proporción surge también un nuevo mercado enfocado a satisfacer las singulares preferencias y crear nuevos servicios específicos para este segmento. Juegan un papel fundamental en este sector la comunicación y las nuevas tecnologías.

- **Familias monoparentales**: Uno de los cambios sociológicos de los últimos años ha sido el incremento de las familias monoparentales y ello ha generado la necesidad de viajar con niños orientando los paquetes que se ofertan a la no exigencia de dos adultos como era normal en muchos ámbitos.

Como vemos, son muchos y muy variados los grupos al alza dentro del sector turístico. Por ello debemos ser capaces de atender a las necesidades específicas de cada uno de ellos, de interactuar de un modo u otro y poder proponer planes acorde a sus preferencias así como a sus tiempos de estancia.

2.5. Turismo costero y marítimo

Se ha visto a lo largo de las secciones anteriores la evolución que el turismo ha experimentado con el transcurso de los años así como también la evolución que el perfil de turista como tal ha sufrido con la llegada de la época moderna.

A pesar de la nueva tendencia por parte del turista de alejarse del turismo de sol y playa, de abrirse a nuevas experiencias más dinámicas e inspeccionar en mayor medida y de una forma más independiente los recursos que sus destinos elegidos ofrecen, dicha interacción con el entorno sigue incluyendo y centrándose en gran medida en la zona costera, ya siendo a través del enfoque más çonvencional.º mediante una aproximación enfocada a las numerosas actividades desarrolladas en torno a la zona costera de la región (surf, paddle surf, vela, windsurf, snorkeling...) [34].

El concepto de turismo de costa abarca el rango completo de turismo, ocio y actividades recreativas que tienen lugar en la zona costera o a lo largo de las aguas costeras. Estos incluyen el desarrollo del turismo costero (alojamiento, restaurantes, industria de alimentos y segundas residencias) y la infraestructura que apoya el desarrollo costero (p. ej. comercio minorista, puertos deportivos, y proveedores de actividades). También están incluidas actividades turísticas como barcos recreativos, ecoturismo de costa, cruceros, natación, pesca recreativa, snorkel y buceo. [5].

El marco de este trabajo se verá encuadrado en el mencionado turismo de costa, pudiendo ser confundido con el turismo marino, el cual podemos entender como una extensión del primero, incluyendo también aquellas actividades recreativas que es-

tán relacionadas con el turismo oceánico como la pesca en aguas profundas o los yates de crucero.

2.6. El Todo Incluido

Con la progresiva evolución del turismo así como de las preferencias del turista y la forma en que este planifica sus viajes han evolucionado paralelamente las estrategias seguidas por parte de los destinos para atraer más turistas.

En las últimas etapas del turismo se ha instaurado un sistema de oferta de paquetes turísticos, entendiendo los mismos como la combinación previa de al menos uno de los siguientes servicios de carácter turístico (alojamiento, transporte, otros servicios) siempre que dicha prestación sobrepase las 24 horas o incluya al menos una noche de estancia [2].

Como paquete turístico surge el concepto de Todo Incluido (TI), con el que el viajante adquiere un producto completo a un precio cerrado, pudiendo incluir el mismo diversos servicios más allá de únicamente el viaje o viaje + alojamiento.

En los últimos años se ha instaurado un sistema de oferta de paquetes turísticos. El éxito de dicho producto es incuestionable, en cambio, este cubre únicamente las necesidades para alcanzar el destino deseado por parte del turista así como aquellas relativas a hospedaje pero deja en el aire las actividades que el mismo llevará a cabo durante su estancia en la isla.

Capítulo 3

Metodologías y modelos algorítmicos para el diseño de itinerarios óptimos

---

En el presente capítulo se abordan tanto los modelos teóricos como algorítmicos empleados para la realización del Sistema de recomendación así como las aproximaciones y adaptaciones realizadas para alcanzar un resultado óptimo reflejando las diversas restricciones por parte del cliente de la aplicación.

3.1. Acercamiento al diseño de rutas turísticas

El problema del viajante de comercio (Travelling Salesman Problem) es un problema estudiado por un gran número de autores. Fruto de dicho estudio han surgido múltiples adaptaciones y modificaciones por parte de numerosos autores.

Una de dichas adecuaciones ha sido el enfoque selectivo del mismo, conocido como Selective Travelling Salesman Problem [35], cuya modificación trata una generalización del Team Orienteering Problem (TOP) [36]. En el TOP se presentan un conjunto

de consumidores potenciales y un beneficio potencial resultante de la visita a cada uno de los mismos. Una flota de vehículos está disponible para visitar a los clientes o consumidores dentro de un tiempo límite dado. El beneficio de un consumidor solamente puede ser recolectado"por uno de los vehículos como máximo.

Podemos considerar como objetivo de esta aproximación el de **identificar los clientes o consumidores que maximizarán el beneficio total, satisfaciendo en todo caso el límite de tiempo dado para cada vehículo.** Para el desarrollo e implementación del *"Sistema de recomendación de itinerarios óptimos de recursos turísticos costeros y marítimos"* se seguirá una aproximación al Team Orienteering Problem (TOP) [37] añadiendo la restricción de una ventana temporal y abordando de esta manera el Team Orienteering Problem with Time Windows (TOPWT).

La definición del modelo, aunque similar a los anteriores, tiene en cuenta en este caso una ventana de tiempo adicional motivada por diferentes situaciones prácticas en las que, en nuestro ejemplo anterior, los clientes no están disponibles a cualquier hora del día sino que tienen una hora de comienzo y una hora de fin para poder ser visitados por uno de los vehículos de la flota.

Otro caso práctico en el que deriva es en la situación en la que nuestros .ªnteriores clientesza no son sino lugares a visitar y nuestros .ªnteriores vehículos"turistas que desean visitar dichos lugares pero que se encuentran con que todos ellos no están disponibles para visita en los mismos espacios temporales, supongamos un museo por ejemplo, con su horario de apertura y de cierre.

En el contexto de nuestra aplicación, el objetivo entonces es el mismo que en el caso del Team Orienteering Problem con el añadido de que nuestras visitas, deberán respetar y ser compatibles con todas las rutas posibles dentro de una solución factible respetando las restricciones temporales correspondientes.

Siguiendo con lo citado en líneas anteriores se expone a continuación puntos clave del modelo [38]:

- Dirigido a diseñar un conjunto **m** de rutas factibles, una por cada día de estancia.

- En nuestro caso los puntos de interés (POI), **i**, son asociados con:

  - Una puntuación relativa al beneficio que aporta dicho POI, s_i
  - Un tiempo de visita, r_i
  - Un intervalo temporal [e_i, l_i] para el cuál el destino estará disponible para su visita

- Los puntos tanto de comienzo como de fin se corresponden con i=0.

- Los tiempos de recorrido entre los pares de puntos t_{ij}, i=0,1...n.

- Las posibles rutas factibles no podrán exceder el valor máximo de T^k_{max} de duración total del *k-ésimo* día teniendo en cuenta tanto los tiempos de viaje como los de duración de la visita y espera en los puntos de interés.

- Función objetivo

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

  En donde la última variable y_i^k es de tipo binario teniendo un valor de uno en caso de que el POI i sea visitado en la ruta k y un valor de cero en caso contrario. Recalcar llegados al final de la descripción del modelo algorítmico escogido que los puntos de interés (POI) llevados a nuestro problema serán en primera instancia las playas de la isla de Tenerife pudiendo ser incluidos en instancias posteriores los restaurantes, empresas de deportes acuáticos... como parte de los mismos.

3.2. Heurística utilizada

Si profundizamos en los diversos recursos presentes a día de hoy, desde las publicaciones escritas más primerizas en el tratamiento de problemas algorítmicos hasta los innumerables recursos online actuales encontraremos un gran número de posibles heurísticas y metaheurísticas adoptadas con la intención de optimizar el tiempo de resolución o en casos de asegurar una solución posiblemente no óptima en cierto tipo de problemas pero lo más refinada posible y en un tiempo aceptable.

En nuestro caso seguiremos una metaheurística constructiva GRASP (Greedy Randomized Adaptive Search Procedure) para la aproximación al problema y la obtención de soluciones aunque posiblemente no óptimas sí factibles y en un tiempo de cómputo aceptable. El algoritmo se corresponde con [38]:

```
function GRASP(maxIterations, sizeRCL)

 readInput();

 for k=1, ..., maxIterations do

     solution = GRASPConstructPhase(sizeRCL);

     solution = localSearch(solution);

     updateSolution(solution, bestSolution);

 end;

 return bestSolution;
```

```
end GRASP;
```

Figura 3.1: Pseudocódigo del algoritmo GRASP

Como podemos observar en la figura anterior, GRASP posee dos fases claramente diferenciadas. En primer lugar, se encuentra la fase de construcción y en segundo lugar la fase de búsqueda local.

Ambas se exponen a continuación, comenzando con la fase de construcción:

```
function GRASPConstructPhase(sizeRCL)

Initialization of m empty routes

While it's posible to visit POIs

  for each  poi_j POI

  find the best position k to insert poi_j in
  a partial route R according to greedy min. f(poi_j)

  Add the triplets (poi_j, i, R) to the Candidate List CL

  Create the Restricted Candidate List, RCL, with
  the top sizeRCL triplets (poi_j, i, R) from CL

  Select a random group of sizeRCL (poi_j, i, R) from RCL
```

```
  Update the route R by inserting the POI poi_j at position i

end GRASPConstructedPhase(sizeRCL);
```

Figura 3.2: Pseudocódigo de la fase de construcción

En esta primera fase se crea un conjunto de elementos candidatos con todos los elementos que pueden ser incorporados a la solución parcial, todas las m rutas vacías.

En las siguientes líneas se evalúan los costes de los respectivos candidatos y hasta el momento en que la solución esté completa o no se puedan seguir visitando rutas, se recorren todos los poi_j encontrando la mejor posición k en la que insertar cada punto para una ruta parcial R dependiendo de su coste evaluado.

El resultado de este proceso se añade a una lista de candidatos CL. A continuación se crea una Lista Restringida de Candidatos (RCL) con un número de elementos sizeRCL que serán los mejores de la original. Por último se escoge aleatoriamente uno de los puntos de dicha lista y se asigna a la solución.

La solución obtenida al finalizar el algoritmo, aunque factible, no tiene porque ser necesariamente óptima. Con esto presente, se aplica la segunda fase del algoritmo, la fase de búsqueda local (localResearch) con la intención de obtener la mejor solución posible. A continuación se expone el algoritmo correspondiente a esta segunda y última fase del algoritmo GRASP.

```
function localResearch(solution)

repeat
```

```
     find the best solution x neighbor of solution

     if f(x)<=f(solution) do solution = x

until f(x) > f(solution) for all neighborhood

return solution

end localResearch;
```

Figura 3.3: Pseudocódigo de la fase de búsqueda local

Como se ha citado en líneas anteriores, esta segunda fase pretende mejorar la solución resultado de la primera fase. Lo hace buscando entre los vecinos y comparando su valor y reemplazando la solución por el nuevo vecino en caso de encontrar uno mejor. El algoritmo itera hasta el momento en que no encuentre soluciones mejores a la actual dentro del conjunto de vecinos.

Capítulo 4

Tecnologías software empleadas para la realización del proyecto

---

La idea de este capítulo es la de brindar al lector una breve descripción de todas las tecnologías de las que se ha hecho uso para el correcto diseño, desarrollo, implementación y despliegue del ***"Sistema de recomendación de itinerarios óptimos de recursos turísticos costeros y marítimos"***.

4.1. Tecnologías generales

4.1.1. GitHub

Github [45] es una plataforma de desarrollo colaborativo de software para alojamiento de proyectos utilizando el sistema de control de versiones Git [46]. Además de lo citado, GitHub nos ofrece varias herramientas útiles para el desarrollo así como para el trabajo en equipo, entre ellas:

- Una **wiki** para el mantenimiento de las distintas versiones de las páginas.
- Un **sistema de seguimiento de problemas** permitiendo detallar problemas en el software o sugerencias.
- Una **herramienta de revisión de código**, donde se pueden además añadir diversos tipos de anotaciones.
- Un **visor de ramas** donde se pueden comparar los progresos en las distintas ramas de nuestros repositorios.

En el contexto de este proyecto, el código fuente del Sistema de recomendación así como cierta documentación relativa al mismo se encuentra alojado en el repositorio privado GitHub de la ULL:

https://github.com/etsiiull/RTCM.

4.1.2. WampServer

WampServer [47] es un entorno de desarrollo web para Windows que permite la creación de aplicaciones web con Apache2, PHP y una base de datos MySQL. Además permite la gestión de dicha base de datos de una manera sencilla a través de PhpMyAdmin.

Desde el comienzo del proyecto WampServer ha sido el entorno base sobre el que se ha desplegado y probado el código del Sistema de recomendación.

4.1.3. Eclipse Neon v3

Eclipse Neon 2 es una de las últimas ediciones del IDE[1] de código abierto Eclipse [63].

[1] IDE: Integrated Development Environment. Un entorno de desarrollo integrado es un entorno de programación que ha sido empaquetado como un programa de aplicación [65]

Eclipse cuenta con un gran número de herramientas para facilitar el trabajo del programador así como la integración con diversas tecnologías como Apache Maven [49] y Git [46]. Además de lo incluido en el paquete de instalación por defecto [64] pueden ser descargados e incluidos plugins externos con el fin de extender sus funcionalidades.

4.1.4. phpMyAdmin

phpMyAdmin [61] es una herramienta software gratuita escrita en PHP para la cómoda e intuitiva administración de MySQL a través de la web. Soporta un amplio rango de operaciones tanto de MySQL como de MariaDB. Proporciona una interfaz de usuario a través de la que se pueden realizar diversas operaciones (manejo de base de datos, tablas, columnas, índices...).

Ofrece también una "sección de sentencias.ª través de la que podremos realizar las diferentes operaciones deseadas de igual manera que si estuviéramos trabajando a través de una consola SQL.

Fue la herramienta escogida para las operaciones relacionadas con la base de datos del presente proyecto también por su integración y/o inclusión en diversas herramientas como en el caso de WampServer [47].

4.1.5. Doxygen

Doxygen [54] es la herramienta estándar para la documentación de código escrito en C++ y cuenta con soporte para otros lenguajes como por ejemplo C, Objective-C, C#, PHP, Java, Python, IDL...

Esta herramienta cuenta con ciertas facilidades y ventajas a la hora de crear una buena documentación del código fuente de nuestros proyectos:

- Doxygen puede generar documentación on-line (HTML) así como manuales de referencia off-line (en Latex).

- Se puede configurar para extraer el código de archivos fuente indocumentados.

- Se puede utilizar incluso para la creación de manuales web como el de su propia página oficial, generado con Doxygen.

Ha sido la herramienta utilizada para la documentación del código del servidor de la aplicación web.

4.1.6. Wunderlist

Wunderlist [48] trata de una aplicación multiplataforma para la gestión de tareas y *"task-list"* que nos permite tener nuestras listas y notas sincronizadas entre todos nuestros dispositivos. Es totalmente gratuita y nos permite la creación de listas personalizadas además de las que vienen por defecto.

Aunque no se trata de una herramienta estrictamente necesaria para el desarrollo de este proyecto, se incluye en esta sección debido a su gran utilidad en lo que a organización de tareas se refiere sincronizándose automáticamente permitiendo tomar notas de ideas en cualquier momento y lugar.

4.2. Tecnologías específicas del lado del servidor

4.2.1. Java SE Development Kit 8

Java SE Development Kit 8 [76] se trata de una versión de desarrollo para la construcción de aplicaciones, applets y componentes escritos en el lenguaje de programación Java [78].

Ha sido necesario para el desarrollo de la parte servidora de la aplicación web.

4.2.2. Apache Maven

Apache Maven [49] es un software de gestión de proyectos así como una herramienta de comprensión. Se basa en el concepto de *"proyecto de modelo de objeto"* (POM del inglés: Project Object Model) y es capaz de proporcionar informes y documentación así como gestionar la construcción y dependencias de proyectos a partir de un código específico.

Ha sido utilizado en este proyecto como herramienta de administración, compilación y construcción del código Java residente en el servidor.

4.2.3. Jersey

Jersey RESTful Web Services in Java [50] se trata de un *framework* de código abierto y de gran calidad para el desarrollo de servicios web en Java siguiendo una arquitectura REST que provee soporte para JAX-RS [52] APIs y sirve como una referencia de implementación JAX-RS (JSR 311 & JSR 339).

Además de servir de estándar para Oracle, Jersey cuenta con su propio kit de herramientas y además de los aspectos más técnicos, cuenta con una amplia biblioteca de recursos y documentación para facilitar el desarrollo a través de dicho framework.

4.2.4. Apache Tomcat v9.0

El software Apache Tomacat es una implementación de código abierto de las tecnologías Java Servlet, JavaServer Pages, Java Expression Language y Java WebSocket desarrolladas bajo el *"Java Community Process"* [53].

Dicha implementación conjunta nos permite construir y arrancar entornos de servidores HTTP en los cuáles poder ejecutar código Java.

4.2.5. MySQL

MySQL [55] es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation y considerada como la base de datos de código abierto más popular del mundo.

MySQL está desarrollado bajo licencia dual GPL/Licencia comercial. Este esquema de doble licenciamiento es posible puesto que es un proyecto patrocinado por una empresa privada que posee el copyright de la mayor parte del código. Desarrollado en su mayor parte en ANSI C y C++ ha llegado a ser considerado como uno de los cuatro componentes de la pila de desarrollo LAMP y WAMP.

En líneas generales, MySQL nos permite, a través de una serie de comandos, poseer nuestra información en una base de datos y recuperarla en el momento en que sea necesaria de una manera eficiente y rápida [56].

Concretando al caso que se presenta, la base de datos gestionada por MySQL poseerá los puntos de interés (POI) como pueden ser las playas con sus puntos geográficos y demás características, los alojamientos, restaurantes...

4.2.6. PHP

PHP [57] es un lenguaje de script del lado del servidor así como una herramienta poderosa a la hora de crear páginas web dinámicas e interactivas.

Podemos ver a PHP como una alternativa eficiente y gratuita a ASP de Microsoft.

PHP aúna una serie de ventajas frente a sus competidores. Entre las más relevantes destacaremos las siguientes [60]:

- PHP se ejecuta en varias plataformas (Windows, Linux, Unix, Mac OS X, etc).
- PHP es compatible con casi todos, por no decir todos los servidores utilizados en la actualidad (Apache, IIS, etc.)

- PHP soporta una amplia gama de bases de datos.
- PHP es gratuito.
- PHP es fácil de aprender y funciona de una manera eficiente en el lado del servidor

4.2.7. Project OSRM

Open Source Routing Machine (OSRM) [67] es un motor de alto rendimiento para el cálculo de caminos mínimos en redes viales de mapas mundiales. Está escrito en C++ y combina sus propios algoritmos con datos de carreteras de libre acceso provenientes del proyecto OpenStreetMap.

En el entorno de la aplicación web, un servidor OSRM ha sido creado de forma local al servidor primario con la función de calcular y retonar matrices de distancia reales permitiendo a la aplicación, como función destacada, el cálculo de rutas óptimas entre los diversos puntos de la solución de un itinierario.

4.3. Tecnologías específicas del lado del cliente

4.3.1. Servidor HTTP Apache

El servidor HTTP Apache [66] es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. El servidor Apache es desarrollado y mantenido por una comunidad de usuarios bajo la supervisión de la Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Apache presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido. Jugó un papel fundamental en el desarrollo de la World Wide Web.

En lo referente al presente proyecto, el Servidor HTTP Apache en su versión 2.0 viene integrado en una de las herramientas comentadas en líneas superiores, WampServer [47].

4.3.2. HTML5

HTML5 [70] es la quinta y mayor versión del lenguaje *çore"* de la World Wide Web: HTML (Hypertext Markup Language).

HTML [71] es el lenguaje de marcas estándar para la creación de páginas y aplicaciones web. HTML describe la estructura de una página web de una manera semántica y puede albergar embedidos programas escritos en un lenguaje de script como puede ser JavaScript permitiendo modificar el contenido y comportamiento de las páginas web.

4.3.3. CSS3

CSS3 [72] es el nombre de la versión más reciente de CSS (Cascade Style Sheets) [73] [77]. Se trata de un lenguaje de hojas de estilo utilizado para describir la presentación de un documento escrito en HTML [71] o XML (incluyendo *"dialectos"* del mismo como SVG o XHTML).

A grandes rasgos podemos decir que CSS describe cómo los elementos deben ser renderizados. De forma diferente a versiones anteriores, CSS3 se encuentra dividido en numerosos elementos denominados módulos. Cada uno de estos módulos añade nuevas capacidades inexistentes en CSS2 o extiende alguna de las características ya existentes.

4.3.4. JavaScript

JavaScript [79] (comúnmente abreviado como JS) es un lenguaje ligero e interpretado, orientado a objetos, con funciones de

primera clase está normalmente orientado a páginas web aunque también tiene amplia presencia en entornos sin navegador, por ejemplo en node.js o en Apache CouchDB.

Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa. Contrariamente a la falsa idea popular, JavaScript no es "Java interpretativo".

Las capacidades dinámicas de JavaScript incluyen construcción de objetos en tiempo de ejecución, listas variables de parámetros, variables que pueden contener funciones, creación de scripts dinámicos (mediante eval), introspección de objetos (mediante for ... in), y recuperación de código fuente (los programas de JavaScript pueden decompilar el cuerpo de funciones a su código fuente original).

En lo relativo a la interacción con páginas web JavaScript, gran parte de la potencia se debe a la *ïnteracción"* con el DOM (Document Object Model). A través de JavaScript podremos acceder a nodos del mismo con instrucciones del tipo:

document.getElementById("menu"); (acceder a un elemento por su id), pero también podremos crear nodos (por ejemplo document.createElement("h1");
document.body.appendChild(heading);). Todo esto posible gracias a la existencia del DOM.

4.3.5. Librerías JavaScript adicionales

Para el desarrollo de la parte cliente escrita en JavaScript se han utilizado determinadas librerías adicionales, en algunos casos debido a las facilidades que estas mismas ofrecen a la hora de programar y en otros debido a que eran dependencia explícita de otras.

Entre todas ellas cabe destacar Underscore.js [74] y JQuery [75]. La primera de ellas nos ofrece numerosos auxiliares para la programación sin tener que extender objetos incorporados dentro del lenguaje. En segundo lugar, JQuery simplifica para nosotros ciertas tareas que serían más complejas, como por ejemplo:

la manera de interactuar con los documentos HTML, manipulación del DOM, manejo de eventos...

4.3.6. Leaflet

Leaflet [58] es una librería JavaScript de código abierto para la representación de mapas geográficos interactivos. Ocupando únicamente 38 KB posee todas las características cartográficas necesarias para el correcto desarrollo e inclusión de mapas interactivos.

Leaflet funciona correctamente en la mayor parte de plataformas tanto de escritorio como móviles y puede ser extendida su funcionalidad a través de plugins. Además de lo anterior, cabe destacar que cuenta con una buena comunidad y una API intuitiva y muy bien documentada.

En lo tocante a este TFG ha sido la herramienta fundamental en el desarrollo tanto del mapa interactivo como de la inclusión de los marcadores sobre los POIs en este Sistema de recomendación de itinerarios óptimos de recursos turísticos costeros y marítimos.

4.3.7. Leaflet Routing Machine

Leaflet Routing Machine [59] es un plugin para extender la funcionalidad de leaflet permitiéndonos la representación de rutas por carretera entre dos o más puntos de un mapa.

Cuenta con una buena documentación y es compatible con numerosos motores de cálculo de rutas georreferenciadas, como el caso del empleado en este proyecto, Open Source Routing Machine (OSRM), el cual es su motor por defecto.

Permite además todo tipo de configuraciones en la interfaz de visualización de las rutas así como otros aspectos configurables.

Esta herramienta ha tenido especial importancia puesto que es la encargada de representar las rutas sobre el mapa interactivo de la aplicación.

4.3.8. Backbone.js

Backbone.js [51] es un framework para código javascript que otorga una estructura diferenciada a las aplicaciones web suministrando modelos con enlaces clave-valor y eventos personalizados, colecciones con una API con innumerables funciones, vistas con manejo declarativo de eventos, conectando todo ello sobre una interfaz RESTful a través de envío y procesamiento de documentos en formato JSON.

Backbone.js ha sido la tecnología utilizada para la implementación de la parte cliente de la aplicación siguiendo un patrón de diseño Modelo-Vista-Controlador (MVC).

4.3.9. Atom

Atom [62] se trata de uno de los editores de texto más populares en la actualidad, con una interfaz actual y totalmente personalizable con la posibilidad de añadir plugins para innumerables funciones y soporte para detectores de errores en la sintaxis de nuestro código en la mayoría de los lenguajes de programación existentes.

Ha sido el editor de texto utilizado para el desarrollo de la parte cliente de la aplicación.

Capítulo 5

Análisis de requisitos

En el presente capítulo se analizarán y expondrán los requisitos que se exige cumpla el sistema de recomendación desglosados dependiendo del lado de aplicación (servidor o cliente) así como de su funcionalidad.

Además se ofrecen los diagrama de casos de uso para ambos y como último punto del capítulo, las fuentes o recursos empleados para la obtención de los datos reales empleados para el desarrollo de la aplicación.

5.1. Requisitos del lado del servidor

Por parte del servidor, este deberá cumplir con los siguientes puntos:

5.1.1. Requisitos funcionales

- Gestión de peticiones: El servidor podrá manejar las peticiones que le serán realizadas a modo de consulta con unos parámetros especificados a través de un fichero tipo JSon.

- Cálculo de rutas óptimas: El servidor deberá ser capaz de resolver las rutas óptimas a partir de los parámetros especificados por las peticiones realizadas por el cliente.
- Volcado de resultado: El servidor será capaz de volcar la solución hallada en el *cálculo de itinerarios* en un formato tipo JSon a través a una dirección IP o URL específica.
- Base de información: El servidor deberá servir de base para el alojamiento y puesta a disposición del cliente cierta información de interés a través de la publicación de la misma en un formato tipo JSon en una IP o URL específica.

5.1.2. Requisitos no funcionales

- Consultas a la base de datos: El servidor deberá ser capaz de comunicarse con la base de datos que aloja la información relativa al proyecto (POIs y alojamientos).
- Obtención de la matriz de distancia: El servidor deberá realizar las peticiones pertinentes a un servidor auxiliar para tanto la obtención de las matrices de distancia como su procesado y salvado para cálculos posteriores.

5.2. Requisitos del lado del cliente

A continuación se presentan los requisitos que deberá cumplir la parte cliente de la aplicación

5.2.1. Requisitos funcionales

- Obtención de datos del servidor: La aplicación cliente deberá ser capaz de acceder a las IPs o URLs en que el servidor proporciona los datos en formato JSon para su obtención o consulta.

- Mapa interactivo: La aplicación deberá representar un mapa interactivo de la isla de Tenerife en que se representarán los itinerarios obtenidos.

- Preferencias del usuario: La aplicación debe contar con algún método de entrada de datos para atender las diferentes especificaciones del usuario final en base a diversas categorías relevantes en el planteamiento del problema como los días de visita, el lugar de partida...

- Restricción del problema: Con las preferencias introducidas en *Preferencias del usuario* se realizará una petición parametrizada al servidor.

- Trazado de itinerarios óptimos: Una vez la parte cliente obtenga los datos ofrecidos por el servidor como resultado deberá además de extraerlos, interpretarlos y dibujarlos sobre el *mapa interactivo*.

- Marcadores interactivos: Los puntos parte del resultado en el mapa (marcadores) deberán ofrecer información adicional acerca del punto que geolocalizan en el momento en el que el usuario lo demande a través de la interacción con los mismos.

5.2.2. Requisitos no funcionales

- Interfaz de usuario: La interfaz debe ser clara e intuitiva además de atractiva visualmente.

- Accesibilidad: La aplicación debe seguir ciertas pautas o guías para romper posibles barreras y de esta manera poder llegar al máximo número de personas manteniendo su operabilidad y funcionalidad.

5.3. Diagramas de casos de uso

En esta sección se presentan los diagramas de casos de uso correspondientes con la parte cliente en primera instancia y a continuación con la parte servidora de la aplicación web.

5.3.1. Diagrama de casos de uso: Cliente

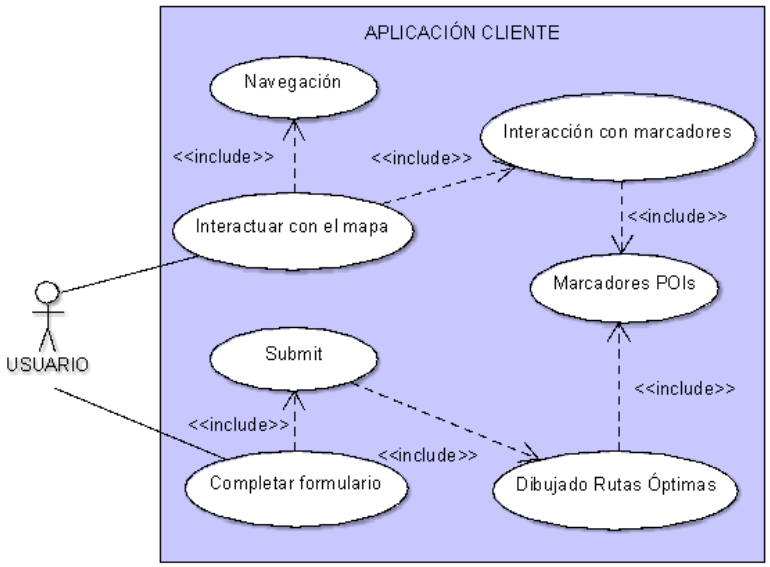

Diagrama de casos de uso: Cliente

5.3.2. Diagrama de casos de uso: Servidor

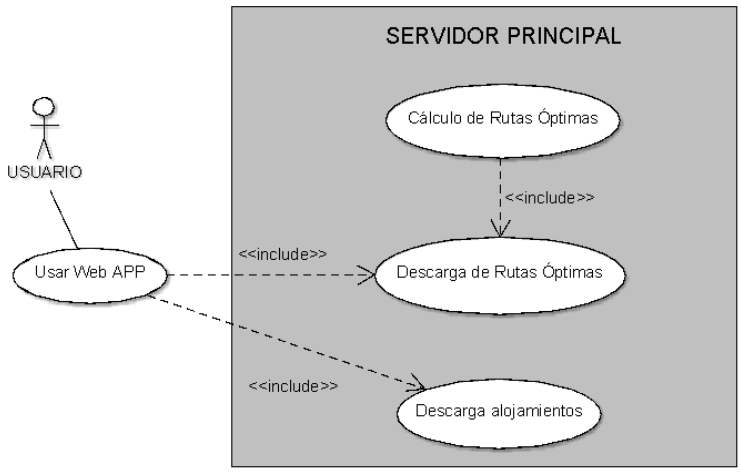

Diagrama de casos de uso: Servidor

5.4. Datos de la aplicación

Bajo el objetivo de ofrecer un sistema capaz de recomendar itinerarios óptimos de recursos turísticos costeros y marítimos ha sido necesaria la obtención de datos relativos a alojamientos y playas de la isla de Tenerife.

5.4.1. Alojamientos

La aplicación web debe ofrecer al usuario la posibilidad de seleccionar su lugar de estancia desde el que partirán y al que llegarán los itinerarios propuestos por la aplicación.

Teniendo esto en cuenta y para la obtención de estos datos se ha recurrido a la web del ayuntamiento de Santa Cruz de Tenerife cargando los datos desde un fichero CSV [39] en el que se

encuentra una selección de 16 hoteles del ayuntamiento de Santa Cruz de Tenerife y determinadas características de los mismos como muestra para las pruebas de la aplicación .

5.4.2. Playas

Siendo el principal recurso de atracción turística dentro de un ámbito costero y marítimo se ha realizado una recopilación de 20 de las playas más conocidas de Tenerife combinando los datos proporcionados por Open Data Canarias en formato CSV [40] e información consultada en webtenerife en su sección de interés: "*Qué visitar*"[41].

De esta forma se ha conseguido recopilar posibles características relevantes para la elección de una playa frente a otra escogiendo para la muestra playas de diferentes características para así poder probar todos los posibles casos a la hora de atender las preferencias personales de los usuarios.

Capítulo 6

Diseño de la aplicación

A través del presente capítulo se expone y explica la arquitectura y diseño elegidos para la realización del Sistema de Recomendación.

Comenzando desde un punto de partida global, se ha escogido el modelo cliente-servidor [42], de modo que la implementación de la misma queda dividida en dos partes perfectamente diferenciadas: la aplicación cliente y la aplicación servidora. La interacción entre las mismas se lleva a cabo a través de una conexión de red.

De forma muy general, la aplicación cliente es arrancada por un servidor Apache y accesible al usuario a través del navegador web. Como se ha comentado en líneas superiores, esta parte ciente se encuentra conectada a un servidor al cual envía la información necesaria para la construcción del problema TOPTW y se encargará también de recibir la resolución del mismo y en base a esta representar la solución sobre un mapa interactivo de la isla de Tenerife.

La parte servidora la ocupa un servicio web escrito en Java a través del framework Jersey siguiendo los principios REST (REpresentational State Transfer) [43] siendo arrancado este por un servidor Tomcat. Dicha parte servidora realiza a su vez conexio-

nes con dos servicios: por una parte se conecta y realiza consultas a la base de datos MySQL responsable de albergar la información tanto de los alojamientos disponibles para el cliente como de las playas (POIs) que formarán parte del problema y por ende de la posible solución.

Por otra parte, se encarga de realizar una serie de peticiones a un servidor de enrutamiento externo OSRM encargado del cálculo de rutas georreferenciadas y capaz de ofrecer una matriz de distancias reales entre los diferentes POIs y/o alojamientos que se soliciten.

En los siguientes apartados se profundiza en la arquitectura del código de cada una de las partes de la aplicación así como en la estructura de la base de datos citada anteriormente.

6.1. Parte Servidora

A continuación se muestra un diagrama y se explica la función principal de cada una de las clases creadas para el correcto funcionamiento de este servicio web RESTful.

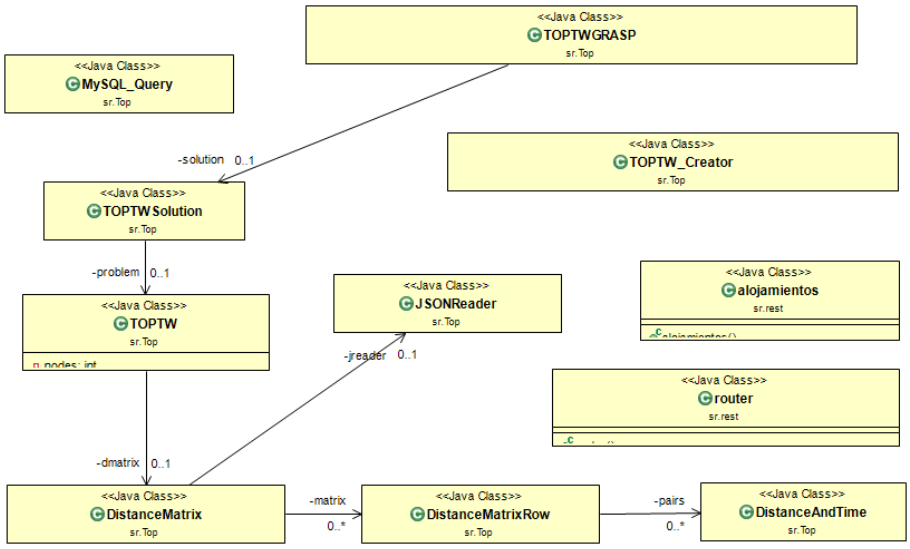

Figura 6.1: Diagrama de clases: Servidor

En la figura anterior se pueden observar las diversas clases que interfieren y trabajan de forma conjunta para el correcto funcionamiento del servidor de la aplicación del Sistema de recomendación.

A continuación se expone de forma resumida el papel que cada una de ellas juega en el funcionamiento final de la aplicación, en el siguiente capítulo se profundizará en cada una de ellas de una forma más extensa y detallada.

- Clase TOPTW: Representa un problema Team Orienteering Problem with Time Windows.

  Alberga toda la información correspondiente a los Puntos de Interés del Problema (POIs) incluyendo todas las características de cada uno de ellos además de los parámetros deseados para la obtención de la solución.

- Clase TOPTWSolution: se corresponde con una solución de un problema TOPTW.

- Clase TOPTWGrasp: se encarga de aplicar la metaheurística GRASP para una solución dada obtenida de un problema TOPTW.

- Clase MySQL_Query: podemos dividir la funcionalidad de esta clase en dos partes claramente diferenciadas.

  En primera instancia se encarga de la realización de una consulta a la base de datos cumpliendo con las preferencias introducidas por el cliente.

  En segundo lugar, en base a los resultados obtenidos de la consulta (Query) realizada en su primera etapa, construye un problema TOPTW en base a los mismos.

- Clase JSONReader: se encarga de ofrecer un mecanismo para la lectura y procesamiento de objetos de tipo JSon a partir de la url en la que se encuentran.

- Clase alojamientos: se encarga de consultar la base de datos para obtener los hoteles disponibles en la misma. A continuación retorna la misma de forma que la aplicación cliente pueda utilizar dichos datos para ofrecerlos al usuario en forma de selector.
- Clase router: es el servicio que solicita la parte cliente una vez el usuario ha completado sus preferencias. Por una parte, se procesan las preferencias y se construye y procesa un problema en base de las mismas.

  La segunda función de la misma es la de devolver al cliente la solución con los parámentros necesarios para que éste la procese y represente.
- Clase DistanceMatrix: representa la matriz de distancias entre todos los puntos que implementa a partir de una consulta al servidor OSRM parte de la aplicación. Cada una de las filas la ocupa un objeto de la clase DistanceMatrixRow la cual contiene a su vez instancia de la clase DistanceAndTime.
- Clase DistanceAndTime: alberga un tiempo y distancia determinados. Se puede entender como un auxiliar para la clase DistanceMatrix en conjunción con la clase DistanceMatrixRow explicada a continuación.
- Clase DistanceMatrixRow: representa cada una de las filas de la matriz de distancia, siguiendo una estructura definida por un hash en el que la clave la ocupa el nombre del punto mientras que el valor del mismo una instancia de la clase anterior.

6.2. Parte cliente

Como se ha citado en páginas anteriores, la parte cliente de la aplicación ha sido implementada siguiendo el patrón modelo-vista-controlador (MCV) [69].

El fundamento de las aplicaciones de que siguen dicho patrón es el de dividir las mismas en dichos componentes. De una forma muy resumida, el modelo alberga los datos utilizados por la aplicación, las vistas todo aquello relativo a la interfaz visual y por último, sobre los controladores recae la transformación y gestión de los datos residentes en los modelos y su puesta en manos de las vistas para su posterior interpretación y representación.

Para la implementación del mismo en la parte cliente se ha utilizado el software Backbone.js explicado en detalle previamente en la sección de tecnologías.

Backbone se vale del patrón MVC con un acercamiento que extiende su concepto base incluyendo además las colecciones, las cuales podemos entender siguiendo la definición formal de la misma [44] como agrupaciones de elementos que reúnen unas características comunes.

A continuación describiremos la forma en la que se ha abordado esta implementación en la parte cliente de la aplicación final.

6.2.1. Modelos

Como se ha comentado anteriormente, los modelos son los encargados de mantener los datos necesarios que, en el contexto que nos ocupa son los relativos a, por una parte, los alojamientos (hoteles) en los que el usuario pudiera residir durante su estancia en la isla y por otra parte los datos necesarios para la representación del mapa de la isla así como para la representación de los itinerarios óptimos en función de las preferencias del usuario proporcionados por la parte servidora.

Para el primer caso se ha implementado un archivo denominado alojamientos_model.js y para el segundo mapa_model.js los cuales se detallarán en profundidad en el capítulo de desarrollo.

6.2.2. Colecciones

Las colecciones se ocupan de mantener elementos con unas características en común, podemos entenderlas como agrupaciones de elementos similares. Dado que solamente se presenta un mapa, únicamente ha sido necesaria una colección para albergar alojamientos, la cuál se implementa en el archivo alojamientos_collection.js.

6.2.3. Vistas

Las vistas son las encargadas de proporcionar la interfaz visual de los diferentes apartados de la aplicación permitiendo separarla en función del ámbito de las mismas, es decir que por ejemplo si solamente fuera necesario cargar una tabla residente en una página, podríamos hacerlo atendiendo a los eventos "despertados"por la vista o modelo relativos a dicha tabla sin la necesidad de recargar la página completa.

Para la implementación visual de nuestra aplicación se ha escogido la implementación de cuatro vistas.

La más general de las vistas de la aplicación sería la implementada en el archivo main_view.js la cuál es padre de las demás y sobre ella se renderizan las mismas con sus consiguientes componentes visuales.

La segunda de las vistas la compone el formulario ofrecido al usuario para la especificación de sus preferencias, dicha vista, implementada en un archivo bajo el nombre de menu_view.js alberga a su vez un campo en el que se renderizará la tercera de las vistas, alojamientos_view.js. Para aclarar esto, la vista del menú tiene entre sus elementos un selector para el alojamiento a escoger por el usuario, dichos alojamientos son insertados en el selector gracias a la segunda vista citada.

La última de las vistas sería la relativa a la representación del mapa en la página web. Además de la propia visualización del mapa se encarga también de poner en vista del usuario las

rutas y marcadores que delimitan las soluciones obtenidas desde la parte servidora.

Cabe destacar que aunque no en todo caso, las vistas implementan plantillas HTML para su correcta visualización en la página siendo elementos con cierta identificación los puntos de anclaje para las mismas. En el caso del sistema de recomendación se han declarado dos plantillas, main_template.js y menu_template.js asociadas respectivamente a las vistas main_view.js y menu_view.js.

6.3. Base de Datos

Toda la información necesaria para el correcto funcionamiento de la aplicación se encuentra albergada en una base de datos MySQL.

Dicha base de datos implementa dos tablas, cada una de ellas correspondiente al tipo de datos o información requeridos por el sistema.

A continuación se muestra una representación de las mismas especificando los campos de cada una de ellas así como su tipo de datos e información adicional acerca de las restricciones de las mismas:

Column	Type	Null	Default
Id	int(11)	No	
geo_long	varchar(45)	No	
geo_lat	varchar(45)	No	
nombre	varchar(45)	Yes	NULL
bandera_azul	varchar(10)	No	
grado_limpieza	varchar(20)	Yes	NULL
tipo_arena	varchar(20)	Yes	NULL
dificultad_acceso	varchar(20)	Yes	NULL

Column	Type	Null	Default
hamacas	varchar(10)	Yes	NULL
Chiringuitos	varchar(10)	Yes	NULL
Socorrista	varchar(10)	Yes	NULL
score	int(11)	No	
tiempo_apertura	float	Yes	NULL
tiempo_cierre	float	Yes	NULL
tiempo_visita	float	Yes	NULL

Tabla 6.1: Tabla playas

Column	Type	Null	Default
Id_al	int(11)	No	
geo_long	varchar(45)	No	
geo_lat	varchar(45)	No	
nombre_al	varchar(45)	No	
categoria_al	int(11)	Yes	NULL
direccion	varchar(45)	Yes	NULL

Tabla 6.2: Tabla alojamientos

En la tabla playas podemos ver una representación de la estructura de la misma en la que se definen los datos relativos a las playas de la isla.

El primero de los campos se refiere al número de identificación que se asigna a cada uno de ellos, siendo este único para toda la base de datos.

A continuación geo_long y geo_lat guardan las coordenadas que definen el posicionamiento geográfico de las diversas playas.

El campo score indica el beneficio que la playa aportaría a la solución del problema. Dicho score ha sido establecido siguiendo

un patrón en función de las diferentes características que dicha playa posee, concretamente se ha aplicado la siguiente fórmula para la obtención de los mismos:

$$SCORE = (30 * \text{grado de limpieza} + 10 * \text{hamacas} + 20 * \text{chiringuito} + 30 * \text{socorrista} + 20 * \text{dificultad de acceso} + \text{bandera}) * 10$$

Siendo los valores relativos a cada campo los siguientes:

Para el grado de limpieza se asignan los valores:

- Valor: -10 en caso de un grado de limpieza bajo
- Valor: 0 en caso de un grado de limpieza medio
- Valor: 1 en caso de un grado de limpieza alto

Para la dificultad de acceso a la playa se asignan los siguientes valores:

- Valor: 1 en caso de una dificultad de acceso baja
- Valor: 0 en caso de una dificultad de acceso media
- Valor: -10 en caso de una dificultad de acceso alta

Para la distinción de bandera azul se asigna:

- Valor: 50 en caso de poseer la distinción de bandera azul
- Valor: 0 en caso de no poseer tal distinción

Para el resto de valores de la ecuación, i.e hamacas, chiringuito y socorrista se asignarán los valores que se muestran a continuación:

- Valor: 1 en caso de poseer el recurso
- Valor: 0 en caso de no poseer tal recurso

Por su parte, la tabla alojamientos muestra la estructura relativa a los registros de hoteles utilizados.

Entre sus campos podemos distinguir Id_al que guarda un identificador único en toda la tabla, geo_long y geo_lang encargados de albergar las coordenadas de cada entrada, su nombre, categoría del establecimiento expresada en estrellas hoteleras y por último la dirección en la que el hotel se encuentra.

Capítulo 7

Implementación de la aplicación

En el presente capítulo se detallan los componentes implementados para el desarrollo de servidor y cliente.

Como se ha detallado en el capítulo de tecnologías para el correcto funcionamiento de la aplicación se ha llevado a cabo la instalación de un servidor Open Source Router Machine (OSRM) pero debido a que la actividad se ha ceñido a su instalación y puesta a punto no se incluye en líneas posteriores.

7.1. Servidor

El servidor principal es el encargado de realizar las peticiones pertinentes a la base de datos para la obtención de datos, de desplegar los servicios web y de la construcción, resolución y comunicación de un problema TOPTW aplicando la heurística GRASP.

Otra de las actividades imprescindibles para el correcto funcionamiento de la aplicación es la creación de la matriz de distancias cuyo proceso comienza con una petición a un servidor

auxiliar (OSRM).

En líneas posteriores se detalla y expone el código utilizado para las funciones más importantes del servidor principal.

7.1.1. Interacción con la parte cliente

El servidor principal ofrece determinados servicios a la parte cliente, comenzando en primera instancia con la consulta a la base de datos para la obtención de los hoteles disponibles, su transformación a un archivo de tipo JSon y su puesta a disposición del cliente a través de la ruta:

http://localhost:8085/SR_WS/webS/alojamientos

La función que se encarga de la implementación de esta funcionalidad se denomina **getAlojamientos** de la clase *alojamientos.java* del paquete *sr.rest* de la aplicación.

Se puede ver a continuación un extracto de la misma:

```
@Path("/alojamientos")
public class alojamientos
{
  @GET
  @Produces("aplication/json")
  public Response getAlojamientos
        (@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/sr_tfg","root", "");
                  } catch (SQLException e) {
```

```
                    e.printStackTrace();
                }
        Statement miStatement = connection
    .createStatement();
        MySQL_Query miQuery = new MySQL_Query();
        return Response.status(200)
     .header("Access-Control-Allow-Origin", "*")
         .header("Access-Control-Allow-Methods",
     "GET, POST, DELETE, PUT")
         .entity(miQuery.getAlojamientos(miStatement)
     .toString(1)).build();
        }
}
```

Figura 7.1: Función getAlojamientos

Esta función crea una conexión con la base de datos para obtener los hoteles a través de una instancia de la clase MySQL_Query. A través de una función de dicha instancia, getAlojamientos, se pone a disposición en la ruta señalada un objeto JSon con los datos obtenidos de la consulta.

Otra de las funciones principales del servidor en cuanto a la interacción con la parte cliente es **getRoutesFromInput** de la clase router.java del mismo paquete que la anterior.

```
@Path("{formJ}")
@GET
@Produces("application/json")
public Response getRoutesfromInput
  (@Context ServletContext context ,
   @PathParam("formJ") JSONObject formJ) throws
```

```
InstantiationException, IllegalAccessException,
ClassNotFoundException, SQLException,
JSONException, IOException
{
      int Id_al = formJ.getInt("alojamiento_base");
      int dias_estancia = formJ.getInt("dias_estancia");

      boolean[] bandera = new boolean[2];
      bandera[0] = formJ.getBoolean("bandera-si");
      bandera[1] = formJ.getBoolean("bandera-no");

      boolean[] g_limpieza = new boolean[3];
      g_limpieza[0] = formJ.getBoolean("g_limpieza_alto");
      g_limpieza[1] = formJ.getBoolean("g_limpieza_medio");
      g_limpieza[2] = formJ.getBoolean("g_limpieza_bajo");

      boolean[] tipo_arena = new boolean[2];
      tipo_arena[0] = formJ.getBoolean("arena-blanca");
      tipo_arena[1] = formJ.getBoolean("arena-negra");

      boolean[] dif_acceso = new boolean[3];
      dif_acceso[0] = formJ.getBoolean("dif_acceso_bajo");
      dif_acceso[1] = formJ.getBoolean("dif_acceso_medio");
      dif_acceso[2] = formJ.getBoolean("dif_acceso_alto");

      boolean[] hamacas= new boolean[2];
      hamacas[0] = formJ.getBoolean("hamacas-si");
      hamacas[1] = formJ.getBoolean("hamacas-no");

      boolean[] chiringuitos = new boolean[2];
      chiringuitos[0] = formJ.getBoolean("chiringuitos-si");
      chiringuitos[1] = formJ.getBoolean("chiringuitos-no");

      TOPTW_Creator miConexion = new TOPTW_Creator(Id_al,
      bandera, g_limpieza, tipo_arena, dif_acceso,
      hamacas, chiringuitos);
```

```
TOPTW miProblema = miConexion
.createProblem(dias_estancia);
TOPTWSolution miSolution = new TOPTWSolution(miProblema);
TOPTWGRASP grasp = new TOPTWGRASP(miSolution);
grasp.GRASP(50, 5);
String mejorSol= grasp.getBest_solution().toString(1);
return Response.status(200)
.header("Access-Control-Allow-Origin", "*")
        .header("Access-Control-Allow-Methods",
"GET, POST, DELETE, PUT")
        .entity(mejorSol).build();
}
```

Figura 7.2: Función getRoutesFromInput

Podemos dividir el objetivo de la función anterior en:

- Obtención de las preferencias del usuario desde un objeto JSon, parámetro principal de la misma.
- Construcción, solución y optimización de la solución de un problema TOPTW de características personalizadas.
- Retorno de la solución optimizada en formato JSon.

7.1.1.1. Interacción con la base de datos

Como hemos visto en la sección anterior, la creación del problema TOPTW es llevada a cabo por la función **createProblem** de la clase MSQL_Creator cuyo código se expone a continuación:

```
public TOPTW createProblem(int days) throws
        SQLException, JSONException, IOException
{
  MySQL_Query miQuery = null;
  TOPTW problema = null; Statement miStatement;

  miStatement = connection.createStatement();
  miQuery = queryMode();
  int numberRow = miQuery
        .getNumberofRows(miStatement);

  problema = new TOPTW(numberRow, days, "car");

  ResultSet rs = miQuery
.getStartPoint(miStatement, this.Id_al);
  problema = setOrigin(problema, rs);
  rs = miQuery.getPOIs(miStatement);
  int i = 1;
  while(rs.next()) {

   problema.setLatitude(i,
        rs.getFloat("geo_lat"));
   problema.setLongitude(i,
        rs.getFloat("geo_long"));
   problema.setName(i, rs.getString("nombre"));
   problema.setBandera(i,
        rs.getString("bandera_azul"));
   problema.setGradoLimpieza(i,
        rs.getString("grado_limpieza"));
   problema.setTipoArena(i,
        rs.getString("tipo_arena"));
  problema.setDificultadAcceso(i,
                rs.getString("dificultad_acceso"));
  problema.setHamacas(i, rs.getString("hamacas"));
  problema.setChiringuitos(i,
                rs.getString("Chiringuitos"));
```

```
    problema.setSocorrista(i,
                    rs.getString("Socorrista"));
    problema.setReadyTime(i,
                    rs.getDouble("tiempo_apertura"));
     problema.setDueTime(i,
           rs.getDouble("tiempo_cierre"));
     problema.setServiceTime(i,
           rs.getDouble("tiempo_visita"));
     problema.setScore(i, rs.getInt("score"));

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

Figura 7.3: Función createProblem

La actividad de la función comienza estableciendo una conexión con la base de datos y realizando una consulta apoyándose de la clase MySQL_Reader tanto para la obtención de los datos como para la obtención del número de filas devueltas por dicha consulta.

En base a dicho número de filas (puntos de interés que coinciden con las especificaciones del usuario) además de los días de estancia escogidos (número máximo de rutas a generar) crea un problema TOPTW. A continuación mediante la función setOrigin se establece el punto de partida para el problema dado, i.e el hotel que el usuario ha seleccionado desde la aplicación cliente y a través de un bucle se establecen los valores de los atributos del problema.

Por último, se llama a la construcción de la matriz de distancias necesaria para completar la construcción del problema.

7.1.2. Interacción con el servidor OSRM

Se ha citado en capítulos anteriores la importancia del servidor auxiliar OSRM que implementa un motor de cálculo de rutas georreferenciadas.

La interacción con el mismo es necesaria para poder construir la matriz de distancias que influirá en la solución del problema. La función que lleva a cabo la comunicación con el mismo es calculate_distanceMatrix la cuál con ayuda de sus clases hermanas construye la matriz de distancia necesaria para la solución del problema.

A continuación se expone y explica el código fuente de dicha función:

```
public JSONObject calculate_distanceMatrix(TOPTW problema)
                        throws JSONException, IOException
        {
        String matrix_points = "http://192.168.1.43:5000/" +
                        "table/v1/driving/";
                for (int i = 0; i<problema.getPOIs()+1; i++)
                {
                  String point = "";
                  point = "" + problema.getLongitude(i) +","+
                        problema.getLatitude(i);
                        if (i<problema.getPOIs())
                                matrix_points += point+";";
                        else
                                matrix_points += point;
                }
                JSONObject distancematrix = jreader
                        .readJsonFromUrl(matrix_points);
                return distancematrix;
        }
```

Figura 7.4: Función calculate_distanceMatrix

La clase comienza guardando en una variable de tipo String el formato de la llamada al servidor OSRM y a continuación se va concatenando la misma con los puntos que formarán parte de la matriz de distancia respetando el formato de la petición aceptado por el servidor OSRM.

A continuación se envía la petición y se retornan los resultados retornados en formato JSon por el servidor gracias al método del readJsonFromURL de la clase JSONReader del paquete sr.Top.

A continuación veremos el código de la función createDistanceMatrix que asigna diferentes tiempos a los pares de la matriz de distancia definitiva, como se puede ver a continuación:

```
public void createDistanceMatrix(TOPTW problema) throws
                JSONException, IOException
  {
JSONObject cd = calculate_distanceMatrix(problema);
  int i =0;
  int auxi=0;
  while (i<cd.getJSONArray("durations").length())
   {
    int j = 0;
    int auxj=0;
    DistanceMatrixRow row = new DistanceMatrixRow();
    while (j<cd.getJSONArray("durations").length())
    {
     if (i != j) {
                double distance = 0;
                double time = 0;
           distance = cd.getJSONArray("durations")
                        .getJSONArray(i).getDouble(j);
           time = cd.getJSONArray("durations")
```

```
                        .getJSONArray(i).getDouble(j);
            DistanceAndTime dt = new
                        DistanceAndTime(distance, time);
            row.add_pair(problema.getName(auxj), dt);
          } else {
            DistanceAndTime dt = new DistanceAndTime(0, 0);
            row.add_pair(problema.getName(auxj), dt);
          }
          matrix.put(problema.getName(auxi), row);
          j++;
          auxj++;
        }
        i++;
        auxi++;
 }
}
```

Figura 7.5: Función createDistanceMatrix

En primer lugar se llama a la función citada líneas arriba para obtener un objeto JSon con todas las distancias entre los POIs.

A continuación se recorre la sección de duraciones del JSon devuelto se asignan sus valores en el orden adecuado a una matriz local.

Dicha matriz está formada por dos elementos, el primero de ellos que guarda el nombre de la playa y el segundo que se compone por una instancia de un objeto que guarda la distancia y tiempo para ese punto. Cabe detallar en este punto que ambas, distancia y tiempo son tratadas de igual manera en función del tiempo.

7.1.3. Aplicación del algoritmo GRASP

La aplicación del algoritmo se realiza sobre una solución previa de un problema TOPTW a través del método GRASP de la clase TOPTWGrasp:

```
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);
            double fitness = this.solution
                                .evaluateFitness();

            averageFitness += fitness;
            if(bestSolution < fitness) {
            bestSolution = fitness;
            System.out.println(this.solution
                .getInfoSolution());
            this.solution.setSolution_data();
            this.best_solution = this
                .solution.getSolution_data();

         }
    }
    averageFitness = averageFitness/maxIterations;
    System.out.println("MEDIA:"+averageFitness);
    System.out.println("MEJOR_SOLUCION:"+bestSolution);
}
```

Figura 7.6: Función GRASP

Esta función implementa cada una de las fases del algoritmo GRASP.

Comienza con una inicialización de las variables necesarias y a continuación comienza la obtención de una solución factible a través de computeGreedySolution.

La siguiente operación que realiza se corresponde con la segunda fase de la heurística, en la que se realiza una búsqueda local. Si en esta búsqueda encuentra soluciones mejores, actualiza la solución final y por último se guarda dicha solución en formato JSon.

A continuación se puede ver un extracto de la solución en formato JSon:


```
{
 "score": 290,
 "solution": {"route_0": [
  {
   "grado_limpieza": "null",
   "ready␣time": "0.0",
   "due␣date": "83880.0",
   "tipo_arena": "null",
   "service␣time": "0.0",
   "leave␣time": "0",
   "socorrista": "null",
   "long": "-16.2531253897",
   "chiringuitos": "null",
   "hamacas": "null",
   "arrive␣time": "0",
   "dificultad_acceso": "null",
   "name": "HOTEL␣MENCEY",
   "bandera": "null",
   "cust␣no.": "0",
```

```
 "lat": "28.4749255299"
},
{
 "grado_limpieza": "ALTA",
 "ready_time": "25200.0",
 "tipo_arena": "NEGRA",
 "service_time": "18000.0",
 "leave_time": "IMAGE",
 "socorrista": "SI",
 "long": "-16.602529525756836",
 "chiringuitos": "SI",
 "hamacas": "NO",
 "arrive_time": "SERVICE_TIME",
 "dificultad_acceso": "BAJA",
 "name": "EL_SOCORRO",
 "due_time": "82800.0",
 "bandera": "SI",
 "cust_no.": "4",
 "lat": "28.393951416015625"
},
{
 "grado_limpieza": "ALTA",
 "ready_time": "25200.0",
 "tipo_arena": "NEGRA",
 "service_time": "18000.0",
 "leave_time": "IMAGE",
 "socorrista": "SI",
 "long": "-16.369667053222656",
 "chiringuitos": "SI",
 "hamacas": "SI",
 "arrive_time": "SERVICE_TIME",
 "dificultad_acceso": "BAJA",
 "name": "LA_ARENA",
 "due_time": "82800.0",
 "bandera": "SI",
 "cust_no.": "1",
```

```
    "lat": "28.35394287109375"
   },
   {
    "grado_limpieza": "null",
    "ready␣time": "0.0",
    "tipo_arena": "null",
    "service␣time": "0.0",
    "leave␣time": "IMAGE",
    "socorrista": "null",
    "long": "−16.2531253897",
    "chiringuitos": "null",
    "hamacas": "null",
    "arrive␣time": "SERVICE␣TIME",
    "dificultad_acceso": "null",
    "name": "HOTEL␣MENCEY",
    "due␣time": "83880.0",
    "bandera": "null",
    "cust␣no.": "0",
    "lat": "28.4749255299"
   }
  ]},
  "number␣of␣nodes": 5,
  "time_cost": 67695.59999999999,
  "max␣time␣per␣route": 75000,
  "max␣number␣of␣routes": 1
}
```


Figura 7.7: Muestra solución en JSon

7.2. Cliente

En este capítulo se expone el código principal de la parte cliente del sistema.

Como se ha citado en capítulos anteriores, en su desarrollo se ha seguido un patrón MVC en el que el la parte fundamental reside en las vistas y las plantillas (templates) asignadas a las mismas.

La plantilla principal, **main_view** es la encargada de manejar al resto, inicializando y llamando a las funciones pertinentes de las mismas.

A continuación podemos ver un extracto de main_view:

```
...
  events: {
        'click#Submit': 'renderRoutes'
          },

initialize: function () {
              },

render: function (page, id) {
          var template = _.template(MainTemplate);
          this.$el.html(template);
          this.page = page;
          this.load(page, id);
        },
...
```

Figura 7.8: Extracto de main_view

En primer lugar se puede apreciar la creación de un evento sobre el elemento con id Submit correspondiente al botón calcular

rutas del formulario. Dicha acción activa un evento que comienza el flujo de la aplicación llamando a la función renderRoutes de la vista map_view. Podemos ver también en la función render como se asigna la plantilla main al elemento correspondiente del DOM. Dicho elemento, líneas más arriba definido, es el que marca la estructura o punto de anclaje de la vista, en este caso ***el: $('main')***.

La plantilla asignada se encarga únicamente de crear los elementos en el DOM sobre los que se anclarán las vistas del mapa y del formulario.

El formulario es el primer elemento de la interfaz visual. A continuación podemos ver un extracto del código de la vista formulario_view:

```
define([
  'underscore',
  'backbone',
  'config',
  'text!templates/formulario_template.html',
  'collections/alojamientos_collection',
  'views/alojamientos_view'
], function(_, Backbone, Config, FormularioTemplate,
                alojamientosColec, alojamientosView){
   var FormularioView = Backbone.View.extend({

     el: '#miformulario',

     alojamientos_datos: null,
     alojamientos_view: null,

    initialize: function (data) {
        this.alojamientos_datos = new alojamientosColec();
        this.alojamientos_datos.bind("reset",
                _.bind(this.renderLodgingList, this));
                    },
```

```
    renderPrep: function()
    {
        this.alojamientos_datos.fetch({reset: true});
    },

 ...

}
```

Figura 7.9: Extracto de formulario_view

En primer lugar se define el punto de anclaje (#miformulario) y se instancian variables relativas a datos y vista de los alojamientos. En la inicialización se crea una nueva instancia de la colección alojamientos_collection y se enlaza el botón del formulario reset con una nueva renderización de los datos del modelo.

Por último se hace un fetch para traer los datos de los alojamientos al modelo y se habilita lo citado anteriormente para el reset.

Esta vista tiene asociado un template que sirve para la representación visual de la misma y en este caso, tiene a su vez un punto de anclaje para la lista de alojamientos disponibles a través de un selector.

A continuación podemos ver un extracto del código:

```
<form name="miformulario" action="" id="miformulario">

  <div class="selectores-1">
  <label for="selector-alojamientos">Seleccione su hotel<label>
  <select name="Id_al" id="selector-alojamientos"></select>
```

```
<label for="num_days">Números de estadía: </label>
<input type="text" name="num_days" value="1" id="num_days">
</div>

<div class="selectores-2">

<label class="checkbox">Bandera azul:</label>
<input type="checkbox" id="bandera-si" value="SI">SI
<input type="checkbox" id="bandera-no" value="NO">NO

...
```

Figura 7.10: Extracto de formulario_template

Podemos observar como se comienza con la creación de un formulario cuyo id *miformulario*, es utilizado por main_template vista anteriormente para su anclaje. A continuación se aprecia un formulario común que, en su primer elemento, implementa un selector cuyos valores son cargados desde la vista de alojamientos_view.

La última parte de la aplicación es la del mapa interactivo, compuesto por su modelo y vista. El modelo es una extensión simple de Backbone.Models por lo que no entraremos en ello.

La vista se encarga de obtener la solución de la URL del servidor, su procesamiento y con la ayuda de una petición al servidor auxiliar, la representación de las diferentes rutas en el mapa. Podemos ver un extracto de la misma a continuación:

```
...

this.routes = $.getJSON('http://localhost:8085/SR_WS/' +
                                        'webS/rutas/'+JSONrequest,

function(data) {
        var solution = data["solution"];
    for (var i = 0; i<stay; i++)
           {
            POIs = [];
            var route_i = solution["route_"+i];
            for (var j = 0; j<route_i.length; j++)
            {
            var lat = route_i[j]["lat"];
            var lon = route_i[j]["long"];
            POIs.push(L.latLng(Number(lat), Number(lon)));

            ...

            }
     ...
}
```

Figura 7.11: extracto map_view

En el extracto podemos ver como se hace una petición al servidor principal de la aplicación para obtener los puntos de la solución. Luego se extraen de la misma la latitud y longitud de los puntos de la solución y se guardan en POIs que se utilizará con la interacción con OSRM y Leaflet para la representación de las rutas.

Capítulo 8

Conclusiones

En el presente capítulo se realiza un análisis del trabajo llevado a cabo en la realización del "Sistema de recomendación de itinerarios óptimos de recursos turísticos costeros y marítimos".

Dicho análisis se apoya principalmente en la comparación entre el trabajo realizado y los objetivos definidos.

Para cerrar este capítulo se presentan unas líneas base de mejora y enriquecimiento de la aplicación a tener en cuenta de darse el caso de una continuación en su desarrollo.

8.1. Conclusiones

A continuación se presentan las conclusiones obtenidas en cuanto al cumplimiento de los objetivos planteados:

- Se ha concretado y definido el ámbito costero y sus actividades más relevantes en lo tocante al ámbito definido. Se ha llevado a cabo un exhaustivo estudio del arte y de la actuación de los Sistemas de Recomendación así como de aquellas implementaciones cuya similitud con el objetivo de

este documento se haya considerado razonable, o bien por los resultados de las mismas o bien por los algoritmos y técnicas utilizadas para su desarrollo.

Se ha realizado un estudio de la historia del turismo en España así como, de una forma más concreta, del caso de Canarias. Además, se ha analizado la evolución del perfil turístico con el paso de los años y la instauración de esta nuestra "época de la información y las tecnologías".

- Se ha definido y elegido como modelo matemático el Tourist Orienteering Problem with Time Windows (TOPTW) y se ha aplicado una heurística GRASP para la obtención de una solución factible.

- Se ha desarrollado la aplicación utilizando datos reales tanto de puntos de interés costeros como alojamientos de la isla de Tenerife.

- Se ha creado una aplicación web que se conecta a un servidor local de gestión de datos georreferenciados para la obtención de la información necesaria para la construcción, resolución y representación de itinerarios óptimos de recursos turísticos costeros y marítimos.

- Se ha llevado a cabo la documentación de los diferentes ámbitos que el desarrollo de la aplicación recoge y, a partir de los mismos, se ha creado el presente documento como memoria final del proyecto.

Con lo citado en líneas anteriores se puede concluir que el Trabajo de Fin de Grado se ha llevado a cabo con éxito consiguiendo el cumplimiento de los objetivos marcados.

Cabe destacar tanto las dificultades como los conocimientos adquiridos durante el desarrollo de este trabajo, fundamentados principalmente por la inexperiencia o desconocimiento de los temas y tecnologías más importantes para su correcto desarrollo.

De una manera más concreta, la mayor parte del estudio se ha centrado en la comprensión general del flujo y funcionamiento

de una aplicación cliente-servidor, el patrón de diseño MVC y por supuesto de las herramientas y librerías involucradas, con las que no se contaba con experiencia alguna.

Como conclusión más personal, sin duda la realización de este proyecto ha sido una actividad enormemente enriquecedora, llegando a conocer la evolución y situación actual tanto del turismo como del perfil turístico y desde un punto de vista más tecnológico, la invaluable formación en herramientas potentes y actuales cuyos aplicativos y soluciones son innumerables.

8.2. Líneas futuras

En las siguientes líneas se presentan varias de las posibles líneas de mejora a seguir en una posible continuación con el desarrollo de la aplicación:

- **Inclusión de actividades deportivas y de ocio**: Canarias cuenta con una gran oferta de empresas dedicadas a la realización de actividades acuáticas como pueden ser los deportes acuáticos, salidas en barco, fiestas en la playa... Un gran punto a tratar sería la inclusión de dichas actividades como partes de los itinerarios recomendados por la aplicación.

- **Inclusión del tiempo atmosférico**: La idea consistiría en la interacción con un servicio de pronóstico a través de las APIs que éstos proporcionan a desarrolladores para la obtención de una predicción del tiempo atmosférico a incluir en cada uno de nuestros puntos de interés.

- **Inclusión de valoraciones de los usuarios**: Se trataría de llevar a cabo una interacción con TripAdvisor y servicios similares. A través de dicha interacción se determinarían las valoraciones relativas a los posibles puntos de interés, pudiendo restringir la búsqueda en base a una cierta valoración media e incluso ver comentarios sobre los puntos de interés.

- **Ampliación de los conjuntos de datos a ofrecer**: Se trataría principalmente de la ampliación de los alojamientos así como de los destinos posibles pudiendo incluir charcas naturales y similares.
- **Desarrollo de la interfaz gráfica**: Se trataría de la mejora del estilo y usabilidad de la aplicación así como una correcta adaptación para dispositivos móviles y tabletas.

Capítulo 9

Presupuesto

En este capítulo se ofrece un presupuesto para el desarrollo íntegro del proyecto desglosado en función de las horas invertidas así como en el coste del software utilizado y por último el hardware empleado durante el período que la realización de este TFG ha abarcado.

Valoracióon del tiempo invertido			
Tarea	Tiempo (horas)	€/hora	Coste (€)
Análisis de requisitos	75	18	1350
Diseño	30	18	540
Desarrollo	125	16	2000
Pruebas	30	16	480
TOTAL	260	-	4370

Tabla 9.1: Presupuesto en función del tiempo invertido

Valoración del hardware utilizado	
Hardware	Coste (€)
Ordenador portátil: ASUS K55V	500

Tabla 9.2: Presupuesto en función del hardware utilizado

Al tratarse de un ordenador personal ya en posesión antes de la realización de este Trabajo de Fin de Grado su coste es orientativo y no afecta al presupuesto general del proyecto.

Valoración del software utilizado	
Software	Coste (€)
Github	0
WampServer	0
Eclipse Neon	0
phpMyAdmin	0
Doxygen	0
Wunderlist	0
Java SE Development Kit	0
Apache Maven	0
Jersey	0
Apache Tomcat	0
MySQL	0
OSRM	0
Leaflet	0
Leaflet RM	0
Librerías adicionales	0
Apache HTTP Server	0
Backbone.js	0

Tabla 9.3: Presupuesto en función del software utilizado

Bibliografía

[1] Turismo de Tenerife y Desarrollo Económico Cabildo de Tenerife. Documents sección indicadores-turísticos (2016)

[2] Elena Fonseca, Turismo, Hotelería y Restaurantes, marzo de 2007

[3] Informe Ditrendia, Mobile en España y el mundo [online], 2016 Disponible en: **https://ditrendia.es/informe-ditrendia-mobile-en-espana-y-en-el-mundo-2016/**

[4] J. A. Fraiz Brea, La constante evolución del turismo: evolución, tecnología, nuevos productos y experiencia, PASOS Vol. 13 N.o 4. Special Issue Págs. 739-740. 2015

[5] C.M. Hall Ocean & Coastal Management 44 (2001) 601-618

[6] Prepublicación año 2001 - Naturaleza de las Islas Canarias. Fernández-Palacios, J.M. y Martín Esquivel, J.L. (Eds) Editorial Turquesa. S/C Tenerife. España Cap. 34 LA MIRADA TURÍSTICA DE CANARIAS. Agustín Santana Talavera

[7] Régimen Económico Fiscal. Gobierno de Canarias - Consejería de Economía, Industria, Comercio y Conocimiento [online]. 2017 Disponible en **http://www.gobiernodecanarias.org/principal/**

[8] Revista Hosteltur, Los nuevos perfiles de turistas, 2014-07-01, nº 14, ref HOSTELTUR 240.

[9] Investigación de operaciones - Entrada Wikipedia [online]. Ult. ed. 12 de junio de 2017 Disponible en **https://es.wikipedia.org/wiki/Investigación_de_operaciones**

[10] Godart, J.M.: Combinatorial optimisation for trip planning. Belgian Journal of Operations Research, Statistics and Computer Science 41(1-2), 59–68 (2001)

[11] Travelling Salesman Problem - Discrete Mathematics sección: Unsolved Problems [online]. 2017 Disponible en **http://mathworld.wolfram.com/TravelingSalesmanProblem.html**

[12] Vansteenwegen, P., Van Oudheusden, D.: The Mobile Tourist Guide: an OR Opportunity. OR Insight 20(3), 21–27 (2007)

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

[14] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The city trip planner: an expert system for tourists. Expert Systems with Applications (2010)

[15] Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: a survey. European Journal of Operational Research (2011)

[16] Ardissono, L., Goy, A., Petrone, G., Signan, M., Torasso, P.: Intrigue: personalized recommendation of tourism attractions for desktop and handset devices. Applied Artificial Intelligence 17(8-9), 687–714 (2003)

[17] Castillo, L., Armengol, E., Onaindía, E., Sebastiá, L., González-Boticario, J., Rodríguez, A., Fernández, S., Arias, J.D., Borrajo, D.: Samap: An user-oriented adaptive system

for planning tourist visits. Expert Systems with Applications 34, 1318–1332 (2008)

[18] Lee, C.-S., Chang, Y.-C., Wang, M.-H.: Ontological recommendation multi–agent for tainan city travel. Expert Systems with Applications 36, 6740–6753 (2009)

[19] Research Gate - La evolución del turismo de masas en España, 1900 - 1936 [descarga online]. Enero 2016 Disponible en **https://www.researchgate.net/publication/310506205_LOS_ANTECEDENTES_DEL_TURISMO_DE_MASAS_EN_ESPANA_1900-1936**

[20] Daniel Muñiz Aguilar, Evolución histórica de la política de turismo social española. Estudios turísticos, nº 147 (2001) pp 141-156

[21] Joaquin Aurioles Martin - Mediterráneo Económico. Las nuevas formas del turismo. mº 5 de la Colección Mediterráneo Económico

[22] Miguel Ángel Campos Seoane - Ministerio de turismo, Un poco de historia [online]. 26 septiembre 2011 Disponible en: **https://www.hosteltur.com/135010_ministerio-turismo-poco-historia.html**

[23] Agencia Estatal Boletín Oficial del Estado. Plan FUTURES BOE núm. 105, de 1 de mayo de 1996, páginas 15452 a 15457

[24] Agencia Estatal Boletín Oficial del Estado. Ley 48/1963, de 8 de julio, sobre competencia en materia turística. BOE núm. 164, de 10 de julio de 1963, páginas 10730 a 10730

[25] Francisco López Palomeque. Política turística y territorio en el escenario de cambio turístico. Boletín de la A.G.E, Nº28 - 199 págs. 23-38

[26] Congreso de los diputados. La restauración - Reinado de Alfonso XII [online]. Disponible en:

http://www.congreso.es/portal/page/portal/Congreso/Congreso/Hist_Normas/PapHist/Restaur/ReyAlfonsoXIII

[27] Agencia Estatal Boletín Oficial del Estado. Real Decreto 720/2005, de 20 de junio. BOE núm. 160, de 6 de julio de 2005, páginas 23894 a 23895

[28] Agencia Estatal Boletín Oficial del Estado. Ley 5/1990. BOE núm. 156, de 30 de junio de 1990, páginas 18710 a 18724

[29] Carlos Javier Pardo Abad. Análisis geográfico del turismo en España. Editorial Universitaria Ramon Areces, 4.2.2: p122-124

[30] Agencia Estatal Boletín Oficial del Estado. Real Decreto-ley de 25 de abril de 1928. BOE núm. 117, de 26 de abril de 1928, páginas 474 a 483

[31] Fernando Bayón Mariné & Mª Cruz Alonso Sutil. 50 años del turismo español: Un análisis histórico y estructural. Editorial Centro de Estudios Ramón Aceres S.A, p385-p391

[32] AprendedeTurismo.org - Pincipales tendencias de turismo en 2016 [online]. Disponible en: **https://www.aprendedeturismo.org/principales-tendencias-del-turismo-en-2016/**

[33] WebTenerife.com - Balance turístico [online]. 2016 Disponible en: **http://www.webtenerife.com/es/investigacion/situacion-turistica/informes-situacion-turistica/documents/informe%20situacion%20turistica%202016.pdf**

[34] alimientatubienestar.es - Los deportes náuticos. 2017 Disponible en: **https://www.alimentatubienestar.es/mejores-deportes-para-el-verano-deportes-nauticos/**

[35] Gilbert Laporte, Silvano Martello, The selective travelling salesman problem. Discrete Applied Mathematics, Volume 26, Issue 2, 1990, Pages 193-207

[36] I.-M. Chao, B. Golden, and E. A. Wasil. The team orienteering problem. European Journal of Operational Research, 88: p464-474, 1996.

[37] Pieter Vansteenwegen, Wouter Souffriau, Greet Vanden Berghe, Dirk Van Oudheusden, Iterated local search for the team orienteering problem with time windows. Computers & Operations Research, Volume 36, Issue 12, 2009, Pages 3281-3290

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

[39] Santa Cruz de Tenerife. Hoteles CSV [online]. Ult. ed. 16 de febrero de 2015 Disponible en: **http://opendatacanarias.es/datos/dataset/tdt-alojamientos/ resource/84c1f174-92a8-4c4d-a631-9e322d895ca6**

[40] Open Data Canarias. Playas Tenerife [online]. 2015 Disponible en: **http://www.opendatacanarias.es/datos/dataset/tdt-playas-de-tenerife**

[41] Webtenerife. Sección Turismo [online]. 2017 Disponible en: **http://www.webtenerife.com/que-visitar/playas**

[42] Universidad de Colima - Docente BDD - Cliente Servidor [online]. 2017 Disponible en: **http://docente.ucol.mx/sadanary/public_html/bd/cs.htm**

[43] En.wikipedia.org, Representational state transfer [online]. Ult. ed. 25 Agosto de 2017 Disponible en: **https://en.wikipedia.org/wiki/Representational_state_transfer**

[44] Definiciones.de - Colección [online]. Disponible en: **https://definicion.de/coleccion/**

[45] GitHub - Página web oficial [online]. 2017 Disponible en: **https://www.github.com/**

[46] Git - Página web oficial [online]. 2017 Disponible en: **https://www.github.com/**

[47] WampServer - Página web oficial [online]. 2017 Disponible en: **https://git-scm.com/**

[48] Wunderlist - Página web oficial [online]. 2017 Disponible en: **https://www.wunderlist.com/**

[49] Apache Maven - Página web oficial [online]. 2017 Disponible en **https://maven.apache.org/**

[50] Jersey RESTful Web Services in Java - Página web oficial [online]. 2017 Disponible en **https://jersey.github.io/**

[51] Backbone.js - Página oficial [online]. 2017 Disponible en: **http://backbonejs.org/**

[52] Java API for RESTful Web Services - Entrada Wikipedia.org [online]. 2017 Disponible en **https://en.wikipedia.org/wiki/Java_API_for_RESTful_Web_Services**

[53] Java Community Process - Community Development of Java Technology Specifications [online]. 2017 Disponible en **https://jcp.org/en/introduction/overview**

[54] Doxygen - Página oficial [online]. 2017 Disponible en: **http://www.stack.nl/ dimitri/doxygen/**

[55] MySQL - Definición en Wikipedia [online]. 2017 Disponible en **https://es.wikipedia.org/wiki/MySQL**

[56] MySQL ¿Qué es y para que sirve?- artículo de isocialweb [online]. 2017 Disponible en **https://isocialweb.agency/mysql-que-es-y-para-que-sirve/**

[57] PHP - PHP5 Tutorial [online]. 2017 Disponible en **https://www.w3schools.com/pHp/default.asp**

[58] Leaflet - Página oficial [online]. 2017 Disponible en: **http://leafletjs.com/**

[59] Leaflet Routing Machine - Página oficial [online]. 2017 Disponible en: **http://www.liedman.net/leaflet-routing-machine/**

[60] PHP - PHP5 Tutorial. Sección: Introducción [online]. 2017 Disponible en **https://www.w3schools.com/pHp/php_intro.asp**

[61] phpMyAdmin - Página oficial [online]. 2017 Disponible en **https://www.phpmyadmin.net/**

[62] Atom - Página oficial [online]. 2017 Disponible en **https://atom.io/**

[63] Eclipse Neon verssion 2.0 - Página oficial [online]. 2017 Disponible en **http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon2**

[64] Eclipse Neon verssion 2.0 Packages - Página oficial [online]. 2017 Disponible en **http://www.eclipse.org/downloads/packages/release/Neon/2**

[65] Entrada - Entorno de Desarrollo Integrado [online]. 25 de enero de 2013 Disponible en **https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/**

[66] Servidor HTTP Apache - Entrada Wikipedia [online]. Ult. ed. 15 de julio de 2017 Disponible en **https://es.wikipedia.org/wiki/Servidor_HTTP_Apache**

[67] Open Source Routing Machine - Página oficial [online]. 2017 Disponible en **http://project-osrm.org/**

[68] Navegadores web en Internet. SPN 3.14 vesión 1.1.2.3.5.8. Blog oficial SPN314 [online]. 2013 Disponible en **http://spn314.blogspot.com/2013/11/cuantos-navegadores-de-internet-hay.html**

[69] Desarrollo.web - Qué es MCV [online]. 02 de enero de 2014 Disponible en: **https://desarrolloweb.com/articulos/que-es-mvc.html**

[70] A vocabulary and associated APIS for HTML and XHTML [online]. W3C recommendation 28 de octubre de 2014 Disponible en **https://www.w3.org/TR/2014/REC-html5-20141028/**

[71] HTML - Entrada wikipedia [online]. Ult. ed. 9 de julio de 2017 Disponible en **https://en.wikipedia.org/wiki/HTML**

[72] CSS3. Entrada Wikipedia [online]. Ult. ed. 14 de julio de 2017 Disponible en **https://en.wikipedia.org/wiki/Cascading_Style_Sheets#CSS_3**

[73] MDN - Mozzilla Developer Network. Sección Web technology for developers CSS [online]. 2017 Disponible en **https://developer.mozilla.org/en-US/docs/Web/CSS**

[74] Underscore.js - Página oficial [online]. 2017 Disponible en: **http://underscorejs.org/**

[75] JQuery - Página oficial [online]. 2017 Disponible en: **http://jquery.com/**

[76] Oracle - Java SE Development Kit 8 [online]. 2017 Disponible en: **http://www.oracle.com/technetwork/java/javase/downloads/ jdk8-downloads-2133151.html**

[77] Cascade Style Sheets. Entrada Wikipedia [online]. Ult. ed. 14 de julio de 2017 Disponible en **https://en.wikipedia.org/wiki/Cascading_Style_Sheets**

[78] Wikipedia - Java (programming language) [online]. Disponible en: **https://en.wikipedia.org/wiki/Java_(programming_language)**

[79] MDN - Mozzilla Developer Network. Sección Web technology for developers JavaScript [online]. 2017 Disponible en **https://developer.mozilla.org/es/docs/Web/JavaScript**