Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Aplicación web para la creación de itinerarios turísticos personalizados

Web application for the creation of personalised tourist itineraries

ACOIDÁN MESA HERNÁNDEZ

La Laguna, 8 de julio de 2022

D. **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J profesor Ayudante Doctor de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor.

D. **José Andrés Moreno Pérez**, con N.I.F. 42.935.437-A Catedrático de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor.

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

Que la presente memoria titulada:

"Aplicación web para la creación de itinerarios turísticos personalizados"

ha sido realizada bajo su dirección por D. **Acoidán Mesa Hernández**,

con N.I.F. 46.260.862-L.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 8 de julio de 2022

Agradecimientos

En primer lugar, me gustaría agradecer a toda mi familia, por apoyarme en los momentos en los que he tenido presión realizando este trabajo. Pero en especial a mi madre, la cual es la que ha hecho posible que me encuentre en este punto, gracias a su esfuerzo sin límites y sin ella no sería la persona que soy.

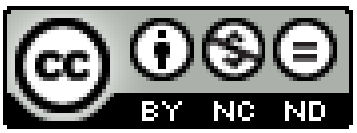
Por otro lado también quiero agradecer a mis amigos y a mi pareja, que han estado conmigo en todo momento en cada paso que he dado.

También agradezco a todos mis compañeros del grado, pero en especial a todos esos compañeros que se han vuelto amigos, ya que sin ellos hubiera sido muchísimo más difícil llegar hasta aquí. Entre ellos Borja Guanche Sicilia, Diego Rodríguez Pérez, Raúl Martín Rigor, Adrián Hernández Suárez, Alberto Ríos de la Rosa y Sergio Pitti de Armas.

En cuanto a lo académico consta, agradezco a cada uno de los profesores que me han enseñado y preparado para que llegue a realizar este trabajo de fin de grado. Pero me gustaría reconocer a Christopher Expósito Izquierdo, mi tutor del trabajo, y a Jose Andrés Moreno Pérez, mi cotutor. Han sido varios meses de trabajo en los cuales han habido altibajos que me han ayudado a solucionar.

Y por último pero no menos importante, quiero agradecer a Carmen Elvira Ramos Domínguez, coordinadora del TFG, la cual ha estado siempre atenta para resolverme las dudas que me han ido surgiendo.

Licencia



Resumen

El turismo es de los sectores más importantes de la sociedad actual, siendo millones el número de personas que se desplazan de un lugar a otro en los distintos medios de transporte, como aviones o barcos.

Los turistas quieren llevarse de su viajes experiencias insuperables, visitando los mejores lugares de la mejor manera posible, siempre ajustándose a sus preferencias y limitaciones. Es decir, quieren tener el mejor itinerario turístico posible. La elaboración de estos itinerarios puede causar problemas, debido al gran bombardeo de información, ofertas y promociones que hay en internet. Por lo que hay poca probabilidad de que se realice una valoración completa de esta. Para evadir este problema, los turistas suelen acudir a empresas físicas como agencias de viajes o utilizar aplicaciones informáticas que preparen sus itinerarios.

Con este trabajo de fin de grado se quiere resolver el Tourist Trip Design Problem (TTDP) a través de un sistema de recomendación que, valorando distintos criterios sea capaz de generar un itinerario turístico personalizado, con un conjunto de visitas ordenadas en el tiempo a distintos puntos de interés.

Este trabajo se basa en una aplicación *full stack* que dispone de un sistema generador de itinerarios turísticos que trata el problema de diseño de rutas turísticas, con el fin de generar itinerarios turísticos personalizados. En esta aplicación se dispone de una interfaz web donde se pueden registrar puntos de interés, turistas y sus preferencias de viajes. Para cada conjunto de preferencias acerca de un viaje, el sistema a través de un algoritmo de recomendación basado en una técnica heurística, es capaz de crear itinerarios personalizados teniendo en cuenta diversos criterios.

El aplicativo cuenta con distintas partes, entre ellas destacan el front-end, que cuenta con las vistas y componentes necesarios para que la visualización sea posible. Y por otro lado, destaca el back-end, el cual se basa en una API REST que cuenta con distintos *endpoints* con los que se gestionan las peticiones HTTP para que se pueda dar el flujo de datos.

Una vez terminada la aplicación, han sido realizadas numerosas experimentaciones en las que se han contemplado diferentes escenarios para, de esta manera, poder comprobar que el aplicativo funciona como se espera.

Palabras clave: Turismo, problema de diseño de itinerarios turísticos, sistema de recomendación, heurística, API

Abstract

Tourism is one of the most important sectors in today's society, with millions of people travelling from one place to another by various means of transport, such as planes or ships.

Tourists want to take away unsurpassed experiences from their trips, visiting the best places in the best possible way, always adjusting to their preferences and limitations. In other words, they want to have the best possible tourist itinerary. The elaboration of these itineraries can cause problems, due to the great bombardment of information, offers and promotions on the internet. As a result, there is little likelihood that a full assessment will be made. To circumvent this problem, tourists often turn to physical companies such as travel agencies or use software applications to prepare their itineraries.

The aim of this final degree project is to solve the Tourist Trip Design Problem (TTDP) by means of a recommendation system that, taking into account different criteria, is capable of generating a personalised tourist itinerary, with a set of visits ordered in time to different points of interest.

This work is based on a *full stack* application that has a tourist itinerary generator system that deals with the problem of designing tourist routes, in order to generate personalised tourist itineraries. This application has a web interface where points of interest, tourists and their travel preferences can be registered. For each set of preferences about a trip, the system, through a recommendation algorithm based on a heuristic technique, is able to create personalised itineraries taking into account various criteria.

The application has different parts, including the front-end, which has the necessary views and components to make visualisation possible. And on the other hand, the back-end, which is based on a REST API that has different *endpoint*s with which HTTP requests are managed so that the data flow can take place.

Once the application has been completed, numerous experiments have been carried out in which different scenarios have been considered. In this way, we were able to check that the application works as expected.

Keywords: Tourism, tourist trip design problem, recommender system, heuristics, API

Índice general

Índice de figuras

Índice de tablas

1. Introducción

1.1. Turismo

El turismo es uno de los sectores más importantes de la sociedad actual ya que proporciona trabajo a una gran cantidad de personas [1] y son mil millones los turistas que viajan alrededor de todo el planeta [2]. Hace décadas era mucho más complicado poder costearse un viaje y por lo tanto muy pocas personas eran las privilegiadas de poder acceder a ello. Pero, sin embargo, con el tiempo, la sociedad fue avanzando y cada vez son más los turistas que se trasladan de un lado a otro a través de diversos medios de transporte como pueden ser aviones, barcos y trenes, entre otros.

El turismo es fundamental para la economía y tiene varios niveles de participación en ésta [3]. Respecto a esta participación, los gobiernos invierten una gran cantidad de recursos económicos al sector turístico, para que los extranjeros que aterrizan en sus países puedan tener la mejor experiencia posible y, por lo tanto, quieran regresar en un futuro. Sin embargo, esta inversión se ha incrementado exponencialmente en los últimos años. Debido a un acontecimiento que marcaría un antes y un después en la historia, como ha sido la pandemia de COVID-19 o Coronavirus, una enfermedad producida por el virus SARS-CoV-2 que tuvo sus primeros casos en China, concretamente en la ciudad de Wuhan. Los cuales se fueron extendiendo hasta volverse una catástrofe mundial, produciendo más de 6 millones de muertos y más de 500 millones de contagios en todo el mundo en menos de 3 años [1]. Como se indica en [4], [5] y [6], este acontecimiento afectó a todos los sectores llegando a reducir muchísimo el precio de bienes como el petróleo y a subir considerablemente el precio de otros tantos. Las farmacias aumentaron su valor bursátil siendo miles el número de mascarillas vendidas por día y otros miles tests de antígenos, entre más recursos para combatir la enfermedad. Pero sin ninguna duda, uno de los sectores más golpeados por la pandemia ha sido el turismo, teniendo que cerrar sus puertas cientos de negocios, aeropuertos y muchas instalaciones más.

El confinamiento que sucedió en España debido a la COVID-19, con fecha de inicio de 15 de marzo de 2020, produjo que la población sólo pudiera salir de sus casas a realizar necesidades básicas cómo puede ser ir a los mercados a comprar, trabajar (las personas a las que no les pausaron su actividad laboral), y pocas actividades más [7]. Esto como ya se ha destacado y como indican [4], [5] y [6], produjo una gran caída en el turismo ya que los vuelos que muchas personas tenían comprados se cancelaron, sus sueños de viajar se frustraron, las reservas que tenían en hoteles y otros puntos de interés se cancelaron porque la mayoría de hoteles cerraron sus puertas ya que no alcanzaban ni un 50% de reservas de las permitidas (haciendo que muchas personas perdieran su puesto de trabajo), entre otras causas.

Actualmente, se está volviendo a retomar la actividad turística debido a que los ciudadanos de nuevo están recibiendo ingresos porque están trabajando otra vez, por lo que se pueden costear sus viajes. Por otro lado, los vuelos están permitidos a la mayoría de lugares, prácticamente todos los hoteles están abiertos, los puntos de interés están reabriendo sus puertas nuevamente y otros por primera vez aprovechando la situación de reapertura de los negocios, etc. También cabe destacar que los gobiernos están bastante implicados en que la actividad turística se retome de la manera más rápida y eficiente dentro de las posibilidades, proponiendo iniciativas, ofertas y distintas actividades para atraer a los turistas [8].

[1] "COVID-19 Data Explorer", *Our World in Data*. Disponible: https://ourworldindata.org/explorers/coronavirus-data-explorer.

1.2. Itinerarios turísticos

Para los turistas elegir un destino turístico, hay varios aspectos que tener en cuenta, entre ellos destaca el económico. Debido a esta misma razón dichos turistas cuando viajan quieren llevarse la mejor experiencia que puedan. Por lo que quieren poder visitar los mejores destinos, lugares, comer en los mejores restaurantes, etc. acorde a sus limitaciones y preferencias, como pueden ser:

- Presupuesto.
- Actividades que hacer.
- Fecha del viaje.
- Si quieren / pueden coger transporte.
- Entre otros.

Es decir, quieren tener el mejor *itinerario turístico* posible, un recorrido que visita distintas paradas y puntos de interés a través de medios de transporte o no, que tiene como fin satisfacer las preferencias de los turistas. Para ello, existen muchas empresas que se encargan de preparar dichos itinerarios a cambio de cobrar una comisión por ello, entre estas se encuentran agencias de viajes, oficinas de turismo, etc. Las funciones de estas empresas son asesorar a los clientes según sus necesidades y ofrecerles productos/servicios separados o juntos en un itinerario turístico. No obstante, muchas personas prefieren organizar ellos mismos sus viajes, buscando información en internet acerca de todos las paradas que hacer, opiniones de puntos de interés etc.

Tener un itinerario turístico aporta muchos beneficios y desventajas como se puede observar en la tabla 1.

Ventajas	Desventajas
Información de la mejor manera de vivir el viaje según el destino	Tener que buscar todo entre toda la información que se puede encontrar en internet
Ahorro de tiempo, ya que se decide con anterioridad	Puede causar estrés y ansiedad por no saber qué elegir
Ahorro económico	
Contratación actividades con antelación y así asegurar la distintas visitas	
Selección de hoteles según el gusto del turista	

Tabla 1. Ventajas y desventajas de tener un itinerario turístico

1.3. Aplicaciones informáticas para la recomendación de paquetes turísticos

Con la era digital y con internet en su máximo apogeo, muchas empresas físicas como agencias de viajes, están siendo sustituidas por aplicaciones informáticas que resuelven los mismos problemas, son más eficientes en muchas ocasiones y hacen que el usuario no se tenga que preocupar por el traslado o manera de comunicación con la empresa física.

Actualmente, hay diversas aplicaciones informáticas relacionadas con el turismo que cuentan con distintas funciones como pueden ser reservar vuelos y/o hoteles, buscar apartamentos y casas vacacionales, buscar transportes, etc. También hay aplicaciones que pueden servir para planificar los itinerarios turísticos según distintos criterios o preferencias del turista, ofreciéndole diferentes tipos de paquetes que puede elegir, algunas de ellas pueden ser:

- *City Trip Planner* [9]. Una aplicación web que teniendo en cuenta las preferencias de viaje de los turistas y los puntos de interés, a través de un sistema experto planifica un itinerario turístico para un turista concreto en 5 ciudades de Bélgica.
- *City Trip Traveller World* [2]. Una aplicación web que aunque no tenga una interfaz muy visual para el usuario, diseña un itinerario turístico personalizado adaptado a los distintos gustos de actividades y ubicaciones.
- *Visitacity* [3]**.** Una web que planifica guías de viajes personales a través de una interfaz de usuario bastante llamativa.

1.4. Objetivos

El objetivo general de este trabajo de fin de grado es la realización de una aplicación web capaz de generar itinerarios turísticos personalizados, teniendo en cuenta distintas preferencias de viajes y distintos puntos de interés.

En la aplicación también se debe poder gestionar información acerca de turistas y sus preferencias de viajes, y de los puntos de interés que se tienen en cuenta en la generación del itinerario turístico. Esta información se gestionará mediante una API REST.

El itinerario turístico generado será una visita ordenada en el tiempo a los distintos puntos de interés, basándose en horarios de apertura y cierre, presupuesto del turista, día y horas de llegada y de salida al destino, zona de alojamiento, etc.

También se presentan otros objetivos específicos con respecto al aplicativo, como por ejemplo que la interfaz web cuente con mapas interactivos para que los usuarios puedan usarlos o que en la visualización del itinerario turístico se presenten indicadores acerca del tiempo ocioso y de los tiempos que el turista dedicará al turismo y al transporte.

---

[2] W. Wörndl y A. Hefele, "Recommender System", *Citytrip.traveller-world.com*, 2015. Disponible: http://citytrip.traveller-world.com.

[3] "Visit A City: Create Your Personal Travel Guide", *Visitacity.com*. Disponible: https://www.visitacity.com.

1.5. Estructura de la memoria

Esta memoria comienza con una introducción para poner al lector en contexto, en esta se explica mínimamente el turismo, los itinerarios turísticos y los objetivos con los que cuenta este trabajo de fin de grado.

En un capítulo posterior se explica la problemática en la que se basa el trabajo, el problema de diseño de itinerarios turísticos.

En el capítulo 3, se presentan muchas de las derivaciones con las que cuenta el problema indicado en el capítulo 2 y varias aplicaciones que resuelven este problema generando itinerarios turísticos personalizados.

El capítulo 4 es el más extenso ya que es donde se presenta basándose en fundamentos de ingeniería del software la aplicación diseñada, se comenta su arquitectura y las tecnologías que se han utilizado. También se describen cada una de sus partes de manera detallada (entre ellas el front-end y la API REST, el back-end) y cómo se produce el flujo de datos.

En el capítulo 5 se presenta una experimentación del aplicativo en distintos escenarios, donde se puede observar que funciona correctamente.

Y para finalizar, se encuentran en el capítulo 6 el presupuesto que conlleva realizar este trabajo, en el capítulo 7 y 8 las conclusiones y líneas futuras (en inglés y en español). Y finalmente en el capítulo 9, la bibliografía que se ha seguido para realizar este documento.

2. Itinerarios turísticos personalizados

La problemática que se aborda en este trabajo de fin de grado se encuentra en la creación de los itinerarios personalizados dedicados a turistas. Como se ha destacado en el capítulo anterior, un itinerario turístico es la dirección y descripción de un camino con expresión de los lugares, accidentes, paradas, etc., que existen a lo largo de él [4]. Por lo tanto, organizar estos itinerarios puede ocasionar bastantes problemas, ya que hay que tener en cuenta diversos factores, entre los cuales se encuentran:

- Preferencias de los usuarios.
- Disponibilidad de los puntos de interés a visitar.
- Tiempo que se tarda en llegar a los puntos de interés.
- …

Un ejemplo de un itinerario turístico puede ser el siguiente: una persona que viaja a Santa Cruz de Tenerife y a la cual le gustan las actividades que tengan que ver con el sol y la playa, la cultura y la naturaleza. En su itinerario posiblemente introducirá puntos de interés como pueden ser playas, piscinas, parques de atracciones acuáticos, senderos, montañas, etc.

Respecto a este itinerario, a la hora de organizarlo puede contar con diversos problemas. Uno se puede encontrar poniéndose en el supuesto de que quiere ir a un restaurante determinado el lunes, en primer lugar debe comprobar la disponibilidad de este ya que es posible que esté cerrado, pero si estuviera abierto, por otro lado debe comprobarse si su presupuesto le permite comer en este restaurante teniendo en cuenta lo que se ha gastado y lo que se quiere gastar. Sin embargo, en el caso de que el restaurante estuviera cerrado, tendría que buscar otro restaurante/punto de interés con el que cubrir sus necesidades y esas horas que le quedan libres.

Otro problema y el principal con el que cuentan los turistas en la creación de sus itinerarios es con la búsqueda en Internet. Son muchísimas empresas y particulares los que lanzan ofertas todos los días para ofrecer servicios y productos de los cuales los usuarios pueden compartir reseñas y opiniones. El usuario que visualiza toda esta información es muy difícil que alcance a acceder a toda ya que hay un gran bombardeo de información. Por lo que puede dar lugar a que al usuario le cause cierto rechazo tener que organizar sus itinerarios.

Debido a esta razón entre otras, con el tiempo para solventar este problema han aparecido diversas herramientas informáticas que tienen como fin organizar itinerarios turísticos personalizados. En estas aplicaciones se diseñan algoritmos que se encargan de decidir qué puntos de interés quieren visitar los usuarios y de realizar una ruta entre estos según las preferencias de los turistas.

Como se indica en [10], el problema de la generación de rutas turísticas personalizadas se ha definido como el *problema de diseño de viajes turísticos* (TTDP, por sus siglas en inglés) y en el modelado de dicho problema se consideran los siguientes datos de entradas:

- Un conjunto de puntos de interés con una serie de atributos, como pueden ser:
    - Geolocalización

[4] "Definición de itinerario", RAE (Real Academia Española). Disponible: https://dle.rae.es/itinerario

    - Categorías
    - Horarios

- El tiempo de viaje entre todos los puntos de interés y, entre la posición inicial del turista y cada uno de los puntos de interés
- La puntuación que tendrá cada punto, para poder obtener una lista ordenada de los puntos de interés según su puntuación (la cual depende de su relación con el usuario)
- El límite de tiempo diario que el turista empleará en puntos de interés.
- Los datos del turista, preferencias, edad, actividades que le gusta realizar, presupuesto con el que cuenta, etc.
- El número de días que el usuario va a estar en el destino, ya que a cada día se le genera una ruta.

Una vez se tratan estas entradas y se aplica el algoritmo, la resolución del TTDP tienen como fin obtener un itinerario turístico compuesto por distintas visitas diarias ordenadas en el tiempo a los distintos puntos de interés (POI), respetando las restricciones y preferencias de los turistas relacionadas con el costo del viaje, el tiempo, el transporte y los atributos de los puntos de interés (como el horario). Es decir, las soluciones del TTDP se encargan de que las recomendaciones de los puntos de interés estén altamente relacionadas con las preferencias de los turistas.

Respecto a la perspectiva de enfoques algorítmicos para resolver el TTDP, existen varias variantes [10] que se explicarán en profundidad en el capítulo posterior, entre ellas se encuentran las siguientes dos:

- Una variante cuya resolución se dirige a encontrar un solo recorrido que maximice la suma de beneficios al visitar los POI respetando ciertas restricciones turísticas y atributos de puntos de interés.
- Una variante cuya resolución se dirige a encontrar múltiples recorridos en función del número de días que durará la visita (uno por día). En estos se suele tener en cuenta entre otros factores, las horas que dedica el turista al turismo.

3. Estado del arte

En la literatura se han realizado varios estudios acerca del problema de diseño de itinerarios turísticos personalizados, varios algoritmos han sido expuestos para ser valorados, así sea por su lógica o por su complejidad computacional. Muchísimas aplicaciones con distintos tipos de algoritmos de recomendación han aparecido a lo largo de los años, creando un ecosistema bastante competente en el mundo de los itinerarios turísticos personalizados.

El problema de selección y enrutamiento por los distintos puntos de interés en un itinerario turístico se le denomina *Tourist Trip Design Problem* (TTDP). Debido al aumento de viajeros en los últimos años, este problema ha sido objeto de investigación, creándose diversas aplicaciones con distintas técnicas para solventarlo.

La primera aplicación que se conoció para crear itinerarios turísticos fue *The Dynamic Tour Guide* (DTG) [11], un aplicativo móvil que generaba un itinerario turístico para unas horas en una ciudad indicada. Para ello el sistema consultaba los distintos *Tour Building Blocks* (TBB), que son las distintas estaciones del recorrido, como pueden ser los puntos de interés o los restaurantes para obtener información acerca de horarios, entre otra. Toda esta información que se obtenía consultando los TBB, se introducía junto con las preferencias de los usuarios en un algoritmo de coincidencia semántica y este retornaba en segundos un itinerario turístico casi óptimo que se desviaba mínimamente de la ruta óptima.

El TTDP se considera en la literatura como una variante a los problemas de orientación (*Orienteering Problem*, OP) [12]. En este tipo de problemas se tiene un conjunto de vértices a visitar que tienen asociado una puntuación que será sumada al coste de la ruta cuando se visite y lo que tratan de resolver estos problemas es la ruta óptima posible con la suma maximizada de costes/recompensas al visitar cada punto. El OP se puede tratar como una combinación [12][13] entre el problema de la mochila (*Knapsack Problem*, KP) [14], un problema de optimización combinatoria que tiene como objetivo proponer distintas soluciones para llenar una mochila que tiene un pesaje máximo, valorando qué objetos (los cuales tienen peso y valor), deben ser colocados en la mochila para maximizar el valor total del conjunto de objetos sin sobrepasar el peso máximo y entre el problema del vendedor viajero (*Traveling Salesman Problem*, TSP) [15], un problema que tiene muchas ramas, desde la informática hasta las matemáticas, que se basa principalmente en que un vendedor quiere visitar distintas ciudades una única vez, empezando y acabando en la misma, y teniendo como objetivo minimizar la distancia recorrida.

Para el modelado de rutas a través del OP existen diversas variantes, como el problema de orientación en equipo (*Team Orienteering Problem*, TOP) [16], el cual se basa en construir varios recorridos de manera simultánea maximizando la recompensa obtenida por visitar un conjunto de vértices, dónde la duración de cada recorrido está definida por un límite establecido anteriormente. También se puede encontrar el problema de orientación con ventanas de tiempo (*Orienteering Problem with Time Windows*, OPTW) [17]. Este problema tiene como entrada un conjunto de clientes (nodos), una duración de servicio (ventanas de tiempo) y un conjunto de arcos y tiene como objetivo crear una ruta que sin sobrepasar un tiempo límite máximo, empiece en un nodo y finalice en otro maximizando la ganancia total y observando en todo momento las restricciones de las ventanas de tiempo de cada cliente. Por otro lado, también se pueden encontrar como variante el problema de orientación dependiente del tiempo (*Time-Dependent Orienteering Problem*, TDOP) [18], el cual se basa en un problema de orientación que se encarga de generar rutas, pero en el cual el tiempo que se tarda de un vértice a otro puede cambiar acorde con el tiempo de salida de dichos

vértices.

La asociación del TTDP como un OP, se debe a que en este se quiere generar un itinerario turístico dónde se visiten los vértices (puntos de interés) con el objetivo de maximizar la suma de puntuaciones de los POI visitados. Para abordar el TTDP, matemáticos, ingenieros, científicos, etc. han abordado múltiples variantes. Aparte de las definidas con anterioridad, las cuales se han usado en algún momento para la creación de distintas aplicaciones generadoras de itinerarios [19], hay bastantes más variantes como por ejemplo el problema de planificación vacacional (*Vacation Planning Problem*, VPP), que se definió como un problema del campo de la optimización que tiene como fin encontrar un conjunto de visitas ordenadas a distintos destinos en una región concreta. Y, para cada uno de esos destinos, proporcionar unos recorridos diarios, una zona hotelera y un número de días de estancia.

En [20] se indica que, a la hora de generar un itinerario turístico personalizado, en lo que al transporte refiere, se debe tener en cuenta otros tipos de transporte. Y de ahí surge el siguiente problema, el problema de planificación de viajes en bicicleta (*Cycle Trip Planning Problem*, CTPP) que se aplicó en [21] y que como destacan sus autores se puede tratar como una variante del problema de orientación en arco (*Arc Orientation Problem*, AOP). Este último se basa en construir una ruta maximizando la suma de la puntuación y sin exceder un límite máximo de longitud definido.

En cuanto a la complejidad computacional del TTDP, en [19] se demuestra que es de tipo NP-duro y eso es un motivo por el que las heurísticas son los métodos apropiados para abordar estos problemas en casos reales.

Otras aplicaciones que existen que resuelven problemas de tipo TTDP o aproximaciones son por un lado *eCOMPASS* [22], una aplicación capaz de crear itinerarios personalizados multimodales de varios días, casi óptimos, teniendo en cuenta distintas medidas, como las pausas para el almuerzo en la planificación o el movimiento de los turistas entre puntos de interés a través de transporte público (siendo el primer planificador de viajes móviles que tiene en cuenta esto). Esta aplicación para construir los itinerarios ha basado el algoritmo que resuelve el TTDP en un nuevo enfoque heurístico que se basa en clústeres, denominado *SlackRoutes* [23], que tiene en cuenta la multimodalidad (utilizar más de un medio de transporte) cuando calcula los tiempos de un viaje a otro. Lo que consigue buenas rutas y la minimización de los retrasos de tiempo provenientes de paradas de tránsito.

Por otro lado, otra aplicación es *TourRec* [24]. Es una aplicación móvil, concretamente diseñada para dispositivos con sistema operativo Android, cuya funcionalidad se basa en crear itinerarios turísticos personalizados en un rango de horas determinado para individuos y grupos. Para ello, tiene en cuenta varios campos como el origen del usuario, el destino, la hora de inicio, la duración máxima del viaje y las categorías de los puntos de interés que el usuario quiera visitar (las cuales valora en una escala del 1 al 5, categorías como *Arts & Entertainment* o *Nightlife)*. Esta aplicación cuenta con diferentes algoritmos desarrollados para la elaboración de los itinerarios, uno de estos se basa en una variante del algoritmo de Dijkstra y otro en la agregación de modelos de usuario o recomendaciones individuales. Por lo que como indica [16] también puede utilizarse para comparar algoritmos de recomendación y estrategias de recomendación de grupos en estudios de campo.

Como se puede observar, todas las aplicaciones descritas, se basan en el mismo problema, el TTDP. Y cada una utiliza una aproximación distinta para resolverlo. Con el tiempo seguirán apareciendo nuevas técnicas y maneras de tratarlo para poder crear así itinerarios turísticos personalizados cada vez mejores.

4. Sistema generador de itinerarios turísticos personalizados

El aplicativo que constituye este trabajo de fin de grado es un sistema generador de itinerarios turísticos en una aplicación *full stack* que trata el TTDP, con el fin de generar itinerarios turísticos personalizados. En esta aplicación se dispone de una interfaz web donde se pueden registrar puntos de interés, con unos determinados atributos como lo son:

- Nombre
- Municipio
- Código postal
- Coste medio en euros que los turistas gastan en el punto.
- Número medio de horas que se está.
- Aplicaciones del punto
- Categorías: sol y playa, rural, naturaleza, gastronomía, cultura
- Días de apertura
- Hora de apertura y hora de cierre
- Descripción del punto
- Coordenadas del punto (latitud y longitud)

También se pueden registrar turistas (nombre, apellidos, correo electrónico y DNI) y para cada uno de ellos, sus preferencias para cada viaje con el fin de que se les genere un itinerario personalizado para su viaje. Para ello se ha desarrollado un sistema de recomendación cuya funcionalidad se basa en el uso de una metaheurística que tiene como entrada:

- Los puntos de interés que están registrados en la base de datos.
- Las preferencias del turista:
    - Coordenadas del hotel en el que se va a alojar (latitud y longitud).
    - Presupuesto con el que cuenta en euros.
    - Día y hora a la que llega al destino y día y hora a la que se va de este.
    - Hora de inicio y de finalización dedicada al turismo.
    - Actividades que quiera realizar (sol y playa, rural, naturaleza, gastronomía, cultura).
    - Si tiene transporte para desplazarse en largas distancias o no.

Este sistema de recomendación asigna unas puntuaciones a los puntos de interés con respecto a las distancias que tienen con la zona del hotel y con respecto a cómo acoplan con las actividades que se ha indicado en las preferencias de viaje que el turista quiere hacer. Después de dar las puntuaciones, el algoritmo recomendador genera el itinerario turístico personalizado y lo exporta como JSON para que se pueda visualizar en la interfaz gráfica web.

La aplicación se describe en los siguientes apartados de manera exhaustiva y utilizando herramientas de la ingeniería del software como diagramas UML.

Respecto a los usos de la aplicación, se pueden observar en la figura del diagrama de casos de usos UML Figura 1.

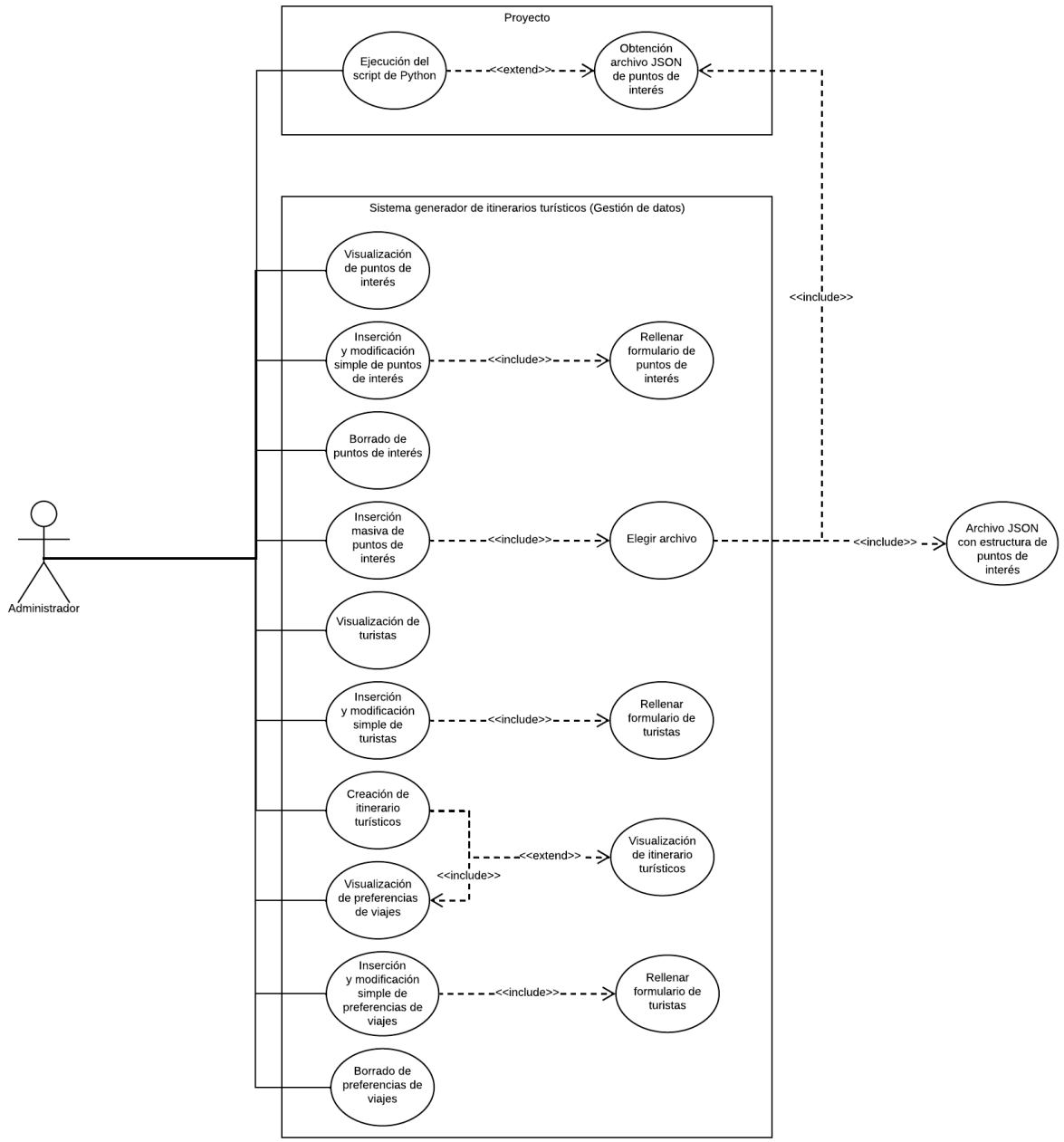


Figura 1. Diagrama de casos de usos UML del proyecto (https://bit.ly/3u4ZE00)

4.1. Tecnologías, arquitectura y partes del aplicativo

El aplicativo se ha realizado con diversas tecnologías cuya unión hacen posible su funcionamiento y se separa en distintas partes, una base de datos, un front-end, un back-end, un sistema recomendador, un script de Python con una función determinada, un script de Docker que sirve para el despliegue y un conjunto de archivos para realizar las GitHub Actions.

Se ha utilizado para desarrollar el aplicativo el patrón de arquitectura *Modelo - Vista - Controlador (MVC)*, este patrón separa en tres componentes diferentes los datos que se manejan en la aplicación (modelo), la interfaz gráfica con la que tratan los usuarios (vista), y el controlador, el cual se encarga de la lógica del aplicativo y de la transferencia de los datos entre el Modelo y la Vista (controlador). Respecto a estas tres partes, el modelo y el controlador se encuentran en el back-end y la vista en el front-end.

Para describir la arquitectura del aplicativo se ha utilizado un diagrama de componentes UML, estos diagramas representan la conexión entre los distintos componentes de un proyecto, así sea lógica o física. Se puede observar en la figura 2.

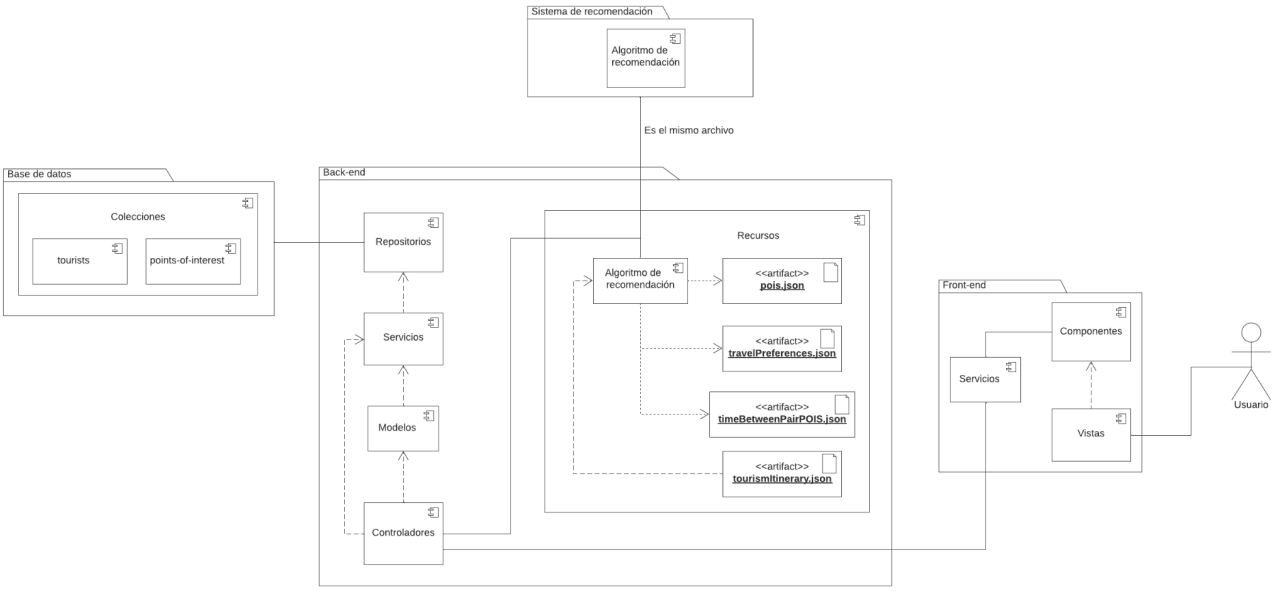


Figura 2. Diagrama de componentes del proyecto (https://bit.ly/3nie2hw)

En las subsecciones posteriores se explicarán cada uno de los paquetes (conjunto de componentes) que se pueden ver en el diagrama de componentes.

4.1.1. Base de datos

La base de datos que se usa en el aplicativo, es una base de datos *NoSQL*, ya que se basa en el sistema gestor de bases de datos NoSQL *MongoDB*. El cual es flexible, sencillo de usar, acopla con diferentes lenguajes de programación y aporta una gran facilidad de indexación.

Si el despliegue del aplicativo se inicia con un perfil de producción, el sistema usará la base de datos *MongoDB Atlas*, una base de datos de MongoDB en la nube completamente gestionada. Pero si se inicia con un perfil de desarrolló se levantará el servicio MongoDB y se usará una base de datos local, la cual al no encontrarse en Internet empieza totalmente vacía.

MongoDB gestiona un tipo de bases de datos NoSQL denominado *bases de datos de documentos*. En estas, se almacenan los datos en estructuras flexibles similares a JSON llamadas BSON, que en este tipo de bases de datos se conocen como documentos. Estos documentos como cualquier JSON, pueden contener muchos pares clave-valor distintos.

Los documentos son agrupados en colecciones, las cuales se definen para almacenar en estos documentos con una relación. Por ejemplo, en una colección denominada *purchases*, posiblemente se almacenarán documentos con una estructura que contenga pares de clave-valor relacionados con compras.

Cabe destacar que MongoDB permite que el modelo de los documentos se asignen a los objetos en el código de la aplicación. En el caso de este aplicativo, asignamos los modelos de MongoDB a objetos Java.

En la base de datos se tratan dos colecciones, `tourists` y `points-of-interest`, una sirve para almacenar documentos que representan turistas y la otra para almacenar documentos que representan puntos de interés, respectivamente.

4.1.2. Back-end

El back-end se ha realizado con el lenguaje de programación *Java* [5], un lenguaje comercializado por primera vez en 1995 por Sun Microsystems. En concreto se ha utilizado el framework *Spring* [6], debido a que es el framework más utilizado de *Java*. Sus principios son la rapidez, la simpleza y la productividad, y es utilizado para crear aplicaciones de alto rendimiento utilizando objetos *Java* sencillos o *Plain Old Java Objects (POJO)*.

Dentro de *Spring*, se ha utilizado *Maven*, una herramienta para la gestión y construcción de aplicaciones *Java*, que proporciona una estructura de directorios que es común en todos los proyectos que lo utilizan. Usa un archivo *"pom.xml"* (*Project Object Model*), a través del cual se realiza toda la construcción del proyecto, desde el orden de construcción hasta las dependencias. Para la geolocalización se ha usado *GeoJson* [7].

El back-end se basa en una API REST que sirve para la transferencia de datos del aplicativo, API REST, actúa como una capa intermedia entre la interfaz de usuario del cliente y la base de datos. Como se puede observar en el capítulo 4.2. Dicha API REST, cuenta con distintos "endpoints" que se explicarán posteriormente, a los que se puede acceder con distintas peticiones HTTP para el manejo los datos, entre ellas *"GET"* (para obtener datos), *"POST"* (para obtener datos), *"PUT"* (para modificar datos) y *"DELETE"* (para eliminar datos).

La estructura de directorios del back-end es bastante extensa, hay algunos directorios que es importante describirlos. Entre ellos, a destacar, se encuentra la carpeta *"src/main/resources"* dónde se almacenan los ficheros con los recursos que necesite el back-end.

Y por otro lado, la carpeta *"src/main/java/..."*, donde se guardan los archivos que contienen el código fuente de la aplicación. Dentro de este directorio se pueden encontrar más, como pueden ser *"domains"*, aquí se encuentran los modelos a través de los cuales se manejan los datos que la aplicación utiliza. Entre ellos se encuentran los que se muestran en el diagrama de clases UML Figura 3.

---

[5] *Java* | *Oracle*. Disponible: https://www.java.com.
[6] "Spring makes Java simple.", *Spring*. Disponible: https://spring.io.
[7] "GeoJSON", *Geojson.org*. Disponible: https://geojson.org.

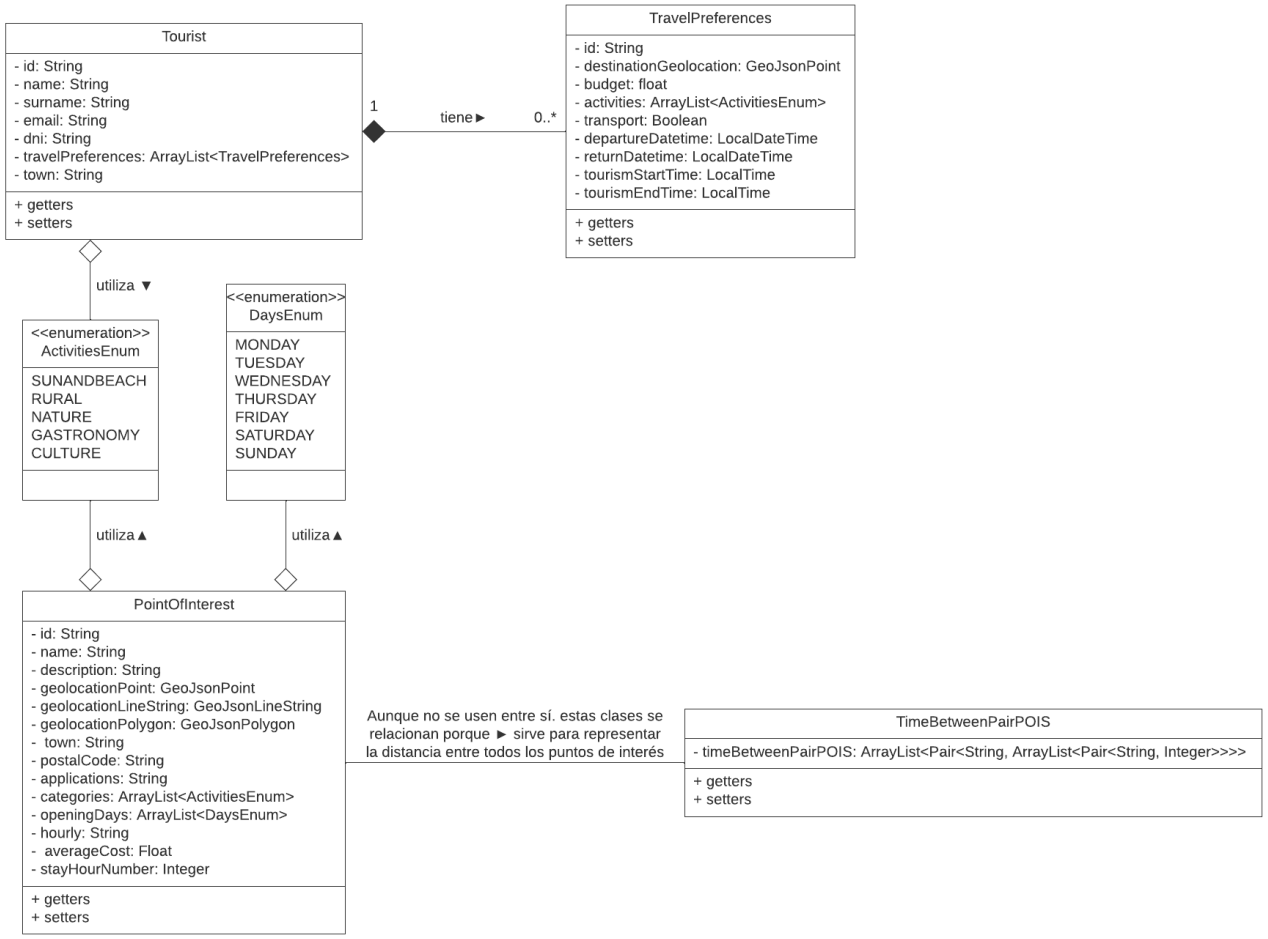


Figura 3. Diagrama de clases del back-end (https://bit.ly/3bzorD3)

Como se puede apreciar, hay 6 clases que se conectan entre sí para que sea posible la arquitectura lógica de datos, estas clases son las siguientes:

- `ActivitiesEnum`: Un enumerado con todos los tipos de actividades/categorías que los turistas pueden querer hacer y que los puntos tienen disponibles para pertenecer (*SUNANDBEACH*, *RURAL*, *NATURE*, *GASTRONOMY*, *CULTURE*)
- `DaysEnum`: Un enumerado con los días de la semana (*MONDAY*, *TUESDAY*, *WEDNESDAY*, *THURSDAY*, *FRIDAY*, *SATURDAY*, *SUNDAY*).
- `PointOfInterest`: Clase que representa un punto de interés, cabe destacar que esta clase referencia a la colección *"points-of-interest"* de la base de datos y cuenta con distintas funciones para acceder a los atributos privados y en estos se encuentran:
    - `private String name`
    - `private String description`
    - `private GeoJsonPoint geolocationPoint`
    - `private GeoJsonLineString geolocationLineString`

    - `private GeoJsonPolygon geolocationPolygon`
    - `private String town`
    - `private String postalCode`
    - `private String applications`
    - `private ArrayList<ActivitiesEnum> categories`
    - `private ArrayList<DaysEnum> openingDays`
    - `private String hourly`
    - `private Float averageCost`
    - `private Integer stayHourNumber`
- `TimeBetweenPairPOIS`: Clase a través de la cual se puede representar los tiempos entre cada par de puntos a partir de la distancia, ya que tiene un atributo que es un tipo "ArrayList", que sirve para almacenar el ID de los puntos de interés y los tiempos entre estos.
- `Tourist`: Clase que representa a un turista, cabe destacar que esta clase referencia a la colección *"tourists"* de la base de datos y cuenta con distintas funciones para acceder a los atributos privados y en estos se encuentran:
    - `private String id`
    - `private String name`
    - `private String surname`
    - `private String email`
    - `private String dni`
    - `private ArrayList<TravelPreferences> travelsPreferences`
- `TravelPreferences`: Clase unas preferencias de viaje de turista, a través de esta clase el sistema recomendador puede generar los itinerarios, comparando estas preferencias con los puntos de interés. Cuenta con distintas funciones para acceder a los atributos privados y en estos se encuentran:
    - `private String id`
    - `private GeoJsonPoint destinationGeolocation`
    - `private float budget`
    - `private ArrayList<ActivitiesEnum> activities`
    - `private Boolean transport`
    - `private LocalDateTime departureDatetime`

    - `private LocalDateTime returnDatetime`
    - `private LocalTime tourismStartTime`
    - `private LocalTime tourismEndTime`

En el directorio *"repositories"* hay dos archivos, estos son interfaces que extienden la interfaz del repositorio de MongoDB para poder utilizar las funciones para gestionar la base de datos (*"MongoRepository"*). A estas interfaces se le denominan repositorios, y hay un repositorio para las operaciones de gestión de la base relacionadas con puntos de interés, y otro para las relacionadas con turistas.

Estas interfaces se usan en los servicios, que son las clases en las que se define la lógica de las funciones relacionadas con la base de datos y se determina que va a pasar cuando se llame a los métodos para gestionarla. Los servicios se encuentran en la carpeta *"services"* y esta, a su vez, tiene dos subdirectorios: *"services"*, donde se definen la interfaces con los métodos que van a usarse y *"servicesImpl"*, donde se extienden las interfaces de la carpeta *"services"* y se implementan los métodos que se definen en estas interfaces con un atributo que representan los repositorios.

Estos servicios son usados en los controladores, los cuales se pueden encontrar en la carpeta *"controllers"*. Los controladores con los que cuenta el back-end son *"PointOfInterestController"*, *"TouristController"* y *"RecommendationSystemController"*.

Estos controladores son los que definen los *"endpoints"* (la URL a la que la interfaz de usuario del cliente manda las peticiones HTTP esperando una gestión de datos y una respuesta con un código de éxito, 2xx) de la API REST y las operaciones que se realiza cuando se manda una petición a alguno de estos. Los *"endpoints"* que gestionan son los siguientes:

- `/pois`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:
    - `HTTP GET`*:* Cuando se mande esta petición, se recogerán de la batos todos los puntos de interés y se devolverán al cliente que mandó la petición, junto con el código de estado HTTP 200.
    - `HTTP POST`: Cuando el cliente mande esta petición deberá pasar en el cuerpo de la petición un objeto JSON con la estructura de un objeto punto de interés. Si no lo hace la API REST le responderá con una excepción con el código de estado HTTP 405, que representa un error del cliente y con el mensaje de error *"Method Not Allowed"*.

      Dicho objeto pasado se transformará en un objeto de la clase correspondiente, después se comprobarán los datos de este y si son válidos, se almacenará en la base de datos y si no lo son, se lanzará una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y con un mensaje determinado según el fallo en la estructura del JSON pasado. Pero si todo se ha ejecutado correctamente, se devolverá en la respuesta la estructura del objeto punto de interés añadido y un código de estado HTTP 201, con el mensaje *"Created"*.
    - `HTTP DELETE`: Cuando se mande este tipo de petición, se borrarán de la base de datos todos los puntos de interés.

- `/pois/{id}`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:
    - `HTTP GET`: Cuando se mande esta petición, se buscará en la base de datos el documento del punto de interés correspondiente al ID pasado en la URL como {id} y se devolverá junto con un código de estado HTTP 200. Sin embargo, si ese ID no se encuentra en la base de datos, la API REST lanzará una excepción con el código de estado HTTP 404 y con el mensaje de error *"Not Found"*.
    - `HTTP DELETE`: Si se manda esta petición, se borrará de la base de datos el punto de interés correspondiente al ID pasado en la URL como {id}. Pero si este no se encuentra, la API REST lanzará una excepción con el código de estado HTTP 404 y con el mensaje de error *"Not Found"*.
    - `HTTP PUT`: Cuando se mande esta petición, se modificará el punto de interés correspondiente al ID pasado en la URL como {id} y se devolverá en la respuesta el JSON del punto de interés modificado y un código de estado HTTP 201 con el mensaje *"Created"*. Pero, sería necesario pasar en el cuerpo de la petición un objeto JSON con la estructura de un objeto punto de interés con las modificaciones que se intenten hacer. Si este objeto se manda con algún dato erróneo en lo que refiere a estructura (por ejemplo, con el nombre vacío) se lanzará una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y con un mensaje determinado según el fallo en la estructura del JSON pasado.

      Si no hay modificaciones entre el que está registrado en la base de datos y el que se está pasando en la petición, será lanzada una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y el mensaje *"No changes to modify the item"*.
- `/tourists`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:
    - `HTTP GET`: Cuando se mande esta petición, se le solicitarán a la base de datos todos los turistas y se devolverán al cliente que mandó la petición, junto con el código de estado HTTP 200.
    - `HTTP POST`: Cuando el cliente mande esta petición deberá pasar en el cuerpo de la petición un objeto JSON con la estructura de un objeto turista. Si no lo hace la API REST le responderá con una excepción con el código de estado HTTP 405 y con el mensaje de error *"Method Not Allowed"*.

      Dicho objeto pasado se transformará en un objeto de la clase correspondiente, después se comprobarán los datos de este y si son válidos, se almacenará en la base de datos y si no lo son, se lanzará una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y con un mensaje determinado según el fallo en la estructura del JSON pasado. Pero si todo se ha ejecutado correctamente, se devolverá en la respuesta la estructura del objeto turista añadido y un código de estado HTTP 201, con el mensaje *"Created"*.
- `/tourists/{id}`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:
    - `HTTP GET`: Cuando se mande esta petición, se buscará en la base de datos el

documento del turista correspondiente al ID pasado en la URL como {id} y se devolverá junto con un código de estado HTTP 200. Sin embargo, si ese ID no se encuentra en la base de datos, la API REST lanzará una excepción con el código de estado HTTP 404 y con el mensaje de error *"Not Found"*.

- `HTTP DELETE`: Si se manda esta petición, se borrará de la base de datos el turista correspondiente al ID pasado en la URL como {id}. Pero si este no se encuentra, la API REST lanzará una excepción con el código de estado HTTP 404 y con el mensaje de error *"Not Found"*.

- `HTTP PUT`: Cuando se mande esta petición, se modificará el turista correspondiente al ID pasado en la URL como {id} y se devolverá en la respuesta el JSON del punto de interés modificado y un código de estado HTTP 201 con el mensaje *"Created"*. Pero, sería necesario pasar en el cuerpo de la petición un objeto JSON con la estructura de un objeto turista con las modificaciones que se intenten hacer. Si este objeto se manda con algún dato erróneo en lo que refiere a estructura (por ejemplo, con el nombre vacío) se lanzará una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y con un mensaje determinado según el fallo en la estructura del JSON pasado.

  Si no hay modificaciones entre el que está registrado en la base de datos y el que se está pasando en la petición, será lanzada una excepción con el código de estado HTTP 400, con el mensaje de error *"Bad Request"* y el mensaje *"No changes to modify the item"*.

- `/tourists/{id}/travels-preferences`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:

  - `HTTP POST`: Cuando se manda esta petición se deberá pasar en el cuerpo un objeto JSON con la estructura de un objeto preferencias de viajes. Para que se pueda almacenar un nuevo registro acorde a las preferencias de viajes en el documento del turista correspondiente al ID pasado en la url como {id}. Si no pasa el objeto JSON, la API REST le responderá con una excepción con el código de estado HTTP 405 y con el mensaje de error *"Method Not Allowed"*.

    Si todo se ha ejecutado correctamente y se han añadido las preferencias de viajes al documento del turista, se devolverá en la respuesta la estructura de las preferencias de viaje añadidas y un código de estado HTTP 201, con el mensaje *"Created"*

- `/{id}/travels-preferences/{idT}`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:

  - `HTTP PUT`: Si se manda esta petición, también se deberá pasar en el cuerpo un objeto JSON con la estructura de un objeto preferencias de viajes, ya que este tipo de petición sirve para modificar las preferencias de viaje correspondientes al ID {idT}, del documento del turista correspondiente al ID pasado en la url como {id}. Si se modifica correctamente se devolverá en la respuesta la estructura de las preferencias de viaje añadidas y un código de estado HTTP 201, con el mensaje *"Created"*. Pero sin embargo, si por ejemplo, no se pasa el objeto JSON en el cuerpo de la petición, se lanzará una excepción con el código de estado HTTP 405 y con el mensaje de error *"Method Not Allowed"*.

    - `HTTP DELETE`: Cuando se mande esta petición, se borrará el registro de las preferencias de viaje correspondientes al ID {idT}, del documento del turista correspondiente al ID pasado en la url como {id}

- `/recommendation-system/{idT}/{idTP}`. Este *endpoint* acepta el siguiente tipo de peticiones HTTP:

    - `HTTP GET`: A este "endpoint" se accede a través de una petición HTTP de tipo GET y pasándole el ID del turista al que se le quiere generar el itinerario como {idT} y el ID de las preferencias que se quieren tener en cuenta para generarlo como {idTP}, se encarga de devolver un itinerario turístico personalizado ejecutando el algoritmo de recomendación. Para ello, lo primero que hace la función asociada a este "endpoint" es preparar los datos para el sistema, como puede ser las preferencias de viaje indicadas del turista indicado, los puntos de interés y las tiempos entre todos los pares de estos. Estos datos se escriben en ficheros que se almacenarán en la carpeta *"src/main/resources"* y posteriormente se ejecuta el script con el algoritmo de recomendación que se encuentra en esa carpeta. Este trata esos datos, genera el itinerario turístico personalizado y lo devuelve para que se pueda mandar como respuesta al cliente que mandó la petición, junto con un código de estado HTTP 200.

      Si hubiera cualquier fallo, así sea en la preparación de los datos o en la generación del itinerario, se lanzaría una excepción con un código de estado HTTP que represente error (4xx – error del cliente o 5xx – error del servidor).

Cuando los controladores intentan introducir documentos en la base de datos, antes se comprueba que estos datos son correctos y válidos para su ingreso. De eso se encargan los validadores que se encuentran en el directorio *"validators"*. Existen en el proyecto dos validadores, PointOfInterestValidator y TouristValidator.

En estos validadores se comprueban los datos de los atributos del objeto a introducir. Como por ejemplo, se comprueba que los nombres no estén vacíos, que la estructura de los correos electrónicos de los turistas cumpla con unos criterios acorde a unas expresiones regulares (como que sea *"xxxxxx@xxxx.xxx")*, etc.

4.1.3. Front-end

El front-end se ha realizado con el lenguaje de programación *Javascript* [25], un lenguaje interpretado que se usa en la mayoría de proyectos en el mundo de la programación y que se caracteriza entre otras cosas, por servir de scripting en programación web. En concreto se ha utilizado el framework *VueJS* [8] (configurándolo con el instalador de paquetes *yarn*), cuyas funciones principales son el renderizado y el sistema de componentes. Además, se le pueden añadir muchas funcionalidades añadiendo bibliotecas externas. Se ha utilizado porque la curva de aprendizaje de este es poco pronunciada y se puede llevar a cabo una buena estructura para la visualización debido al desglose que permite en componentes.

Cabe destacar que se ha usado la librería *Vuetify*, la cual permite usar componentes basados en *"Material Design"* para de esta manera, facilitar la creación de la interfaz gráfica. Y se ha utilizado para crear mapas interactivos, *Leaflet* [9], una biblioteca de código abierto que es utilizada para crear

[8] "Vue.js - The Progressive JavaScript Framework | Vue.js", *Vuejs.org*. Disponible: https://vuejs.org.
[9] "Leaflet — an open-source JavaScript library for interactive maps", *Leafletjs.com*. Disponible:

mapas web con posibilidad de interactividad con estos.

Por otro lado, se ha usado la librería *Vue I18n* para la realización de la internacionalización del aplicativo, haciendo que este disponible en los idiomas español e inglés. Y para hacer posible el enrutamiento de la aplicación se ha usado la librería *Vue Router*.

La estructura de directorios del front-end es una estructura bastante ordenada por partes, creada en su totalidad por el framework *VueJS*, y que permite al desarrollador tener todos sus archivos organizados.

En cuanto al análisis de la estructura, en lo que refiere a los archivos que se encuentran en el directorio raíz, hay numerosos archivos de configuración, como por ejemplo el *"vue.config.js"*, que sirve para la configuración de *VueJS*. O "nginx.conf" y *"jsconfig.json"*, que sirven para el despliegue y para la configuración de *Node.js*, respectivamente, entre otros. Es importante tener en cuenta que *"App.vue"*, es el archivo donde empieza a crearse la aplicación Vue.

Respecto a los directorios, se puede encontrar la carpeta *"node modules"*, en la cual se almacenan todos los paquetes o dependencias necesarios para la ejecución del front-end. También está la carpeta *"public"*, donde se almacenan los ficheros estáticos que no serán procesados por el framework, como el *"index.html"* y el *"favicon.ico"*.

Y por otro lado, está el directorio más importante y más complejo internamente, ya que en este se almacena el código fuente del proyecto Vue. Este es el *"src"* y dentro se pueden encontrar directorios como *"assets"***,** donde se encuentran archivos estáticos como imágenes, vídeos, etc. O como "plugins", en el cual, como indica su nombre, se almacenarán los ficheros relacionados con plugins, en el caso del aplicativo sólo se encuentra el archivo *"vuetify.js"*. Además, se encuentra el directorio *"router"*, que está relacionado con la librería *Vue Router* y el directorio *"locales"* que está relacionado con la librería *Vue I18n*.

Dentro de *"src"*, también está el directorio *"services"*, el cual contiene los servicios que hacen posible la conexión con el back-end, debido a que son clases que cuentan con funciones que se encargan de realizar las peticiones al back-end que este es capaz de contemplar. Hay 3 servicios, *"PointOfInterestService"*, *"TouristsService"* y *"TourismItineraryService"*, los cuales realizan peticiones al back-end que tienen que ver con puntos de interés, turistas y preferencias de viajes, y con la generación de itinerarios turísticos, respectivamente.

Los componentes son almacenados en el directorio "components" dentro de "src", hay un gran número de componentes, entre ellos se encuentran:

- `AddPointOfInterestComponent`: Componente que construye un formulario para añadir puntos de interés. Este proporciona un mapa interactivo en el que el usuario que los está introduciendo puede pulsar para que se carguen automáticamente las coordenadas de la geolocalización del punto (latitud y longitud). También proporciona entradas para que se introduzcan los distintos datos con los que un punto de interés puede contar, como el nombre, la descripción, las categorías, etc. Cuando se pulsa el botón *"ADD"* o *"AÑADIR"*, el servicio *"PointOfInterestService"*, se encarga de mandar una petición POST al back-end con el objeto construido con los datos del formulario, para que se añada a la base de datos.
- `AddTouristComponent`: Componente que construye el formulario para añadir turistas. Este proporciona entradas para que se introduzcan los distintos datos con los que un turista

https://leafletjs.com.

puede contar, como el nombre, apellidos, DNI y correo electrónico. Cuando se pulsa el botón *"ADD"* o *"AÑADIR"*, el servicio *"TouristService"*, se encarga de mandar una petición POST al back-end con el objeto construido con los datos del formulario, para que se añada a la base de datos.

- `AddTravelPreferencesComponent`: Componente que construye el formulario para añadir preferencias de viajes. Este formulario cuenta con un selector para elegir el turista al que se le quiere asignar las preferencias de viaje, con un mapa interactivo para que cuando el turista pulse se carguen automáticamente las coordenadas de la geolocalización del hotel en el que se va a alojar, y con distintas entradas para introducir datos con respecto a estas preferencias, como las actividades que desea hacer el turista, si tiene transporte o no, el presupuesto con el que cuenta, etc. Cuando se pulsa el botón *"ADD"* o *"AÑADIR"*, el servicio *"TouristService"*, se encarga de mandar una petición POST al back-end con el objeto construido con los datos del formulario, para que se añadan las preferencias de viaje del turista indicado a la base de datos.

- `HomeComponent`: Componente que construye la página de inicio, que cuenta con un título y el logo de la página web.

- `LanguageSwitcherComponent`: Componente que se utiliza dentro del componente *"ToolbarComponent"*, que construye un selector que se encarga de cambiar el idioma del aplicativo según la opción seleccionada, el cual se puede cambiar también en la URL. Por defecto la opción seleccionada de este selector es *"en"*, por lo que el idioma con el que empieza la aplicación es el inglés.

- `PointsOfInterestComponent`: Componente a través del cual se visualizan los puntos de interés registrados en la base de datos. Para ello, primeramente, el servicio "PointOfInterestService" realiza una petición GET al back-end para obtener todos los puntos de interés que se encuentran en la base de datos. Posteriormente a través de un mapa interactivo se muestran con marcadores y a través de de una tabla se muestran listados. En la tabla, a la derecha de cada registro en *"Actions"* o *"Acciones"*, se puede encontrar un botón con el icono de un lápiz para modificar el registro y otro con el icono de una papelera para eliminarlo.

  Cuando se vaya a modificar un punto de interés porque se pulsó el marcador o el botón del lápiz, se abrirá en la parte superior un formulario igual al de *"AddPointOfInteretsComponent"*, pero en este, cuando se pulsa el botón *"MODIFY"*, o *"MODIFICAR"*, el servicio *"AddPointOfInteretsComponent*" mandará al back-end una petición PUT para que este modifique el documento relacionado a dicho punto de interés.

  Cabe destacar que en la parte superior de este componente hay tres botones:

  - *"AÑADIR PUNTO DE INTERÉS"* o *"ADD POINT OF INTEREST"*: Redirige al usuario al formulario para añadir un punto de interés.

  - *"BORRAR TODOS LOS PUNTOS DE INTERÉS"* o *"DELETE ALL POINTS OF INTEREST"*: Cuando se pulsa se abre una ventana emergente de seguridad para asegurar que el botón no se pulsó sin querer, y cuando se confirma el borrado, el servicio nombrado anteriormente manda una petición DELETE al back-end para borrar todos los puntos de interés

  - *"INSERCIÓN MASIVA DE PUNTOS DE INTERÉS"* o *"MASIVE INSERTION OF*

POINTS OF INTEREST": Cuando se pulsa se despliega una ventana emergente con un formulario, que tiene una entrada para seleccionar un fichero que contenga JSONs con la estructura de puntos de interés, los cuales cuando se pulse el botón *"ADD"* o *"AÑADIR",* se añadirán a la base de datos. Para que se añadan los JSON deben tener la estructura que muestra la figura 4.


```
{
  "town": "Adeje",
  "postalCode": 38670,
  "categories": [
    "SUNANDBEACH"
  ],
  "name": "Playa El Puertito de Armeñime",
  "description": "Playa de arena gris en ensenada natural",
  "applications": "Zona de bañlo local. Zona de baño turística. Zona de buceo",
  "stayHourNumber": 5,
  "averageCost": 15,
  "openingDays": [
    "MONDAY",
    "TUESDAY",
    "WEDNESDAY",
    "THURSDAY",
    "FRIDAY",
    "SATURDAY",
    "SUNDAY"
  ],
  "geolocationPoint": {
    "type": "Point",
    "coordinates": [
      "-16.768215",
      "28.113725"
    ]
  }
},
```


Figura 4. JSON de ejemplo de un punto de interés

- `ToolbarComponent`: Componente que construye la barra superior de la aplicación web. La cual cuenta con el logo del aplicativo a la izquierda y a la derecha con el componente *"LanguageSwitcherComponent"* y con 3 botones:
    - Botón con icono de avión: Redirige a la página de preferencias de viajes de los turistas
    - Botón con icono de persona: Redirige a la página de gestión de turistas
    - Botón con icono de ruta: Redirige a la página de gestión de los puntos de interés turísticos.
- `TourismItineraryComponent`: Componente que construye la visualización de un itinerario turístico personalizado a través de un mapa interactivo dónde se ven los puntos que el usuario visitará y unas tarjetas (una por día), con la descripción detallada de cada visita. Para ello al componente se le pasan unas propiedades, las cuales son el ID del turista y el ID de las preferencias de viaje con las que se quiere generar el itinerario. Posteriormente con estos datos, el servicio *"TourismItineraryService"* realiza una petición GET al back-end

para obtener el itinerario turístico personalizado, generado de la ejecución del algoritmo de recomendación.

- `TouristsComponent`: Componente a través del cual se visualizan los turistas registrados en la base de datos. Para ello, primeramente, el servicio *"TouristsService"* realiza una petición GET al back-end para obtener todos los turistas que se encuentran en la base de datos. Posteriormente a través de una tabla se muestran listados. También cabe destacar que en la tabla, a la derecha de cada registro en *"Actions"* o *"Acciones"*, se puede encontrar un botón con el icono de un lápiz para modificar el registro y otro con el icono de una papelera para eliminarlo. También se encuentra en la parte superior del componente el botón *"AÑADIR TURISTA"* o *"ADD TOURIST"*, el cual redirige al usuario al formulario del *"AddTouristComponent"*.

- `TravelPreferencesComponent`: Componente a través del cual se visualizan las preferencias de viajes registrados en la base de datos. Para ello, primeramente, el servicio *"TouristsService"* realiza una petición GET al back-end para obtener todos los turistas que se encuentran en la base de datos y de estos se extraen las preferencias de viajes. Posteriormente a través de una tabla se muestran listadas. También cabe destacar que en la tabla, a la derecha de cada registro en *"Actions"* o *"Acciones"*, se puede encontrar un botón con el icono de un lápiz para modificar el registro, otro con el icono de una papelera para eliminarlo y otro con el icono de un avión y un marcador para generar un itinerario turístico acorde a las preferencias del registro en el que se encuentra el botón (esta es la manera de generar los itinerario turísticos personalizados). Este último botón redirige al usuario a la visualización del itinerario turístico mediante el componente *"TourismItineraryComponent"*. También se encuentra en la parte superior del componente el botón *"AÑADIR PREFERENCIAS DE VIAJE"* o *"ADD TRAVEL PREFERENCES"*, el cual redirige al usuario al formulario del componente *"AddTravelPreferencesComponent"*.

Todos estos componentes descritos son usados en las vistas, que se encuentran en el directorio *"views"*. Estas cuelgan de *"App.vue"*, donde se utiliza el componente *"ToolbarComponent"*, para que se vea en todas las vistas. La vistas creadas son las siguientes:

- `AddPointOfInterestView`: Utiliza el componente *"AddPointOfInterestComponent"*.
- `AddTouristView`: Utiliza el componente *"AddTouristComponent"*.
- `AddTravelPreferencesView`: Utiliza el componente *"AddTravelPreferencesComponent"*.
- `HomeView`: Utiliza el componente *"HomeComponent"*.
- `PointsOfInterestView`: Utiliza el componente *"PointOfInterestComponent"*.
- `TourismItineraryView`: Utiliza el componente *"TourismItineraryComponent"*.
- `TouristsView`: Utiliza el componente *"TouristsComponent"*.
- `TravelsPreferencesView`: Utiliza el componente *"TravelsPreferencesComponent"*.

4.1.4. Despliegue

Para el despliegue de la aplicación se ha utilizado *Docker* [10] y se guardan las imágenes creadas a través de las *GitHub Actions* en *Docker Hub*, un servicio de repositorios de *Docker* para alojar imágenes Docker en la nube.

Se tiene un archivo *"Dockerfile"* en el back-end que indica el despliegue de este y otro en el front-end. Y por último, un archivo denominado *"docker-compose.yml"*, que se encuentra en el directorio raíz.

El *"docker-compose.yml"*, se puede ejecutar con dos perfiles:

- *dev*: Perfil de desarrollo, inicia la imagen subida a Docker Hub del back-end, la imagen subida a Docker Hub del front-end y un servicio de MongoDB para la base de datos, por lo que la base de datos empieza vacía cuando se despliega con este perfil.
- *prod*: Perfil de producción, inicia la imagen subida a Docker Hub del back-end, la imagen subida a Docker Hub del front-end y se conecta a una base de datos de MongoDB Atlas, por lo que la aplicación empieza con datos cargados.

4.1.5. Script de python

Se ha creado un script con el lenguaje *Python*, ya que es un lenguaje que contiene muchas librerías las cuales ayudan a solventar los problemas de una manera más rápida y eficiente. Este script se encarga de pasar la información de más de 200 puntos de interés turísticos desde un archivo Excel a un archivo y formato JSON.

Este script se utiliza para la inserción masiva de los puntos de interés turísticos.

[10] *Docker*. Disponible: https://www.docker.com/.

4.2. Flujo de datos

El flujo de datos del aplicativo es el que se puede observar en la figura 5.

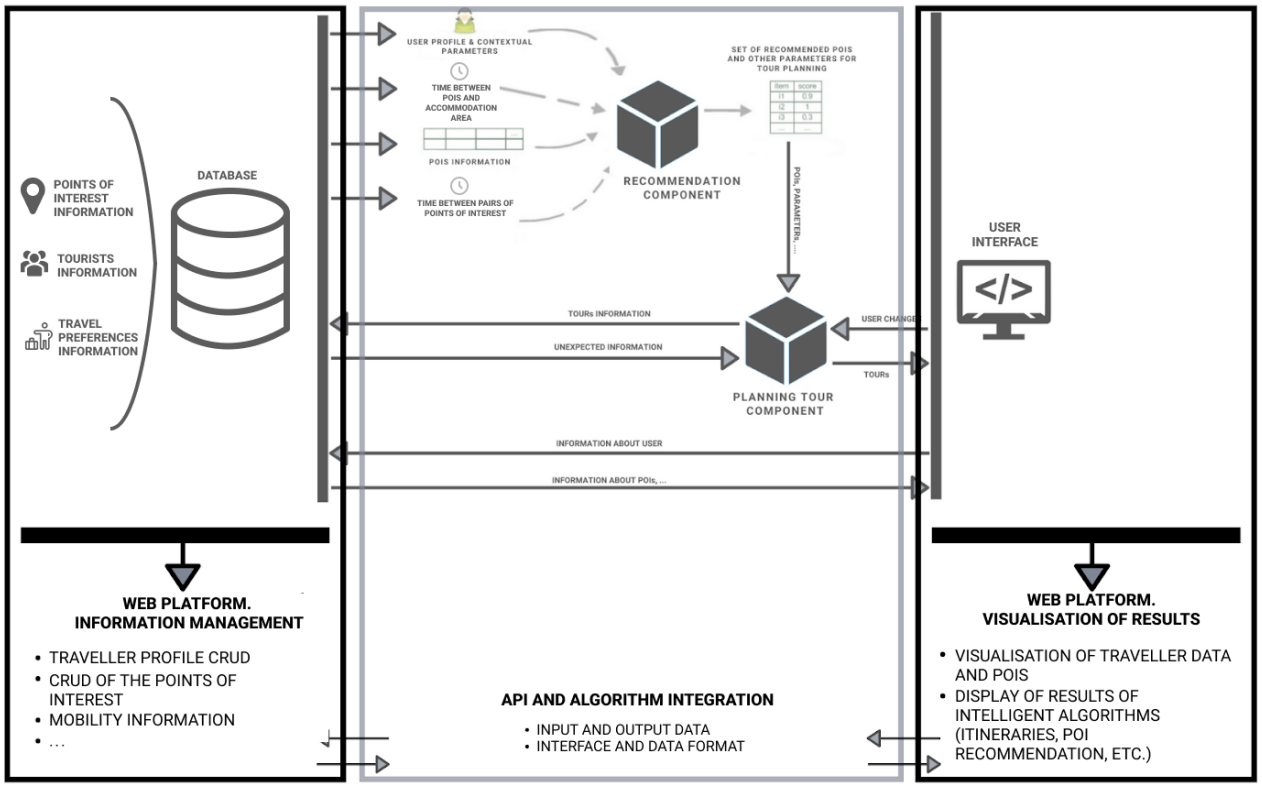


Figura 5. Flujo de datos (https://bit.ly/3nKfnOA)

Los usuarios se conectan al front-end, donde ven la interfaz con los mapas interactivos, donde visualizan los itinerarios turísticos personalizados generados y visualizan, modifican o añaden datos de los puntos de interés, turistas y preferencias de viajes. Estos datos los pueden manejar debido a que el front-end se conecta con el back-end y se transfieren datos entre ellos. A su vez, el back-end se conecta con la base de datos para traer o almacenar dicha información transferida o a transferir. Esta base de datos cuenta con la información de los puntos de interés, de los turistas y de las preferencias de viajes.

Y por otro lado, el back-end también se conecta con el sistema de recomendación ejecutándolo para que acorde a unas preferencias de viaje genere un itinerario turístico.

En conclusión, respecto a los datos:

- La base de datos los almacena.
- El back-end los transfiere entre la base de datos, el front-end y el sistema recomendador.
- El front-end los visualiza y contiene los formularios con los que se introducen los datos para la adición de los recursos a la base de datos.
- El sistema recomendador recoge los datos necesarios y genera el itinerario turístico.

5. Sistema de recomendación turístico

Los sistemas de recomendación son últimamente muy utilizados en la sociedad para todo tipo de ámbitos, como para recomendar ítems a los distintos usuarios en tiendas o para recomendar puntos de interés a turistas. Hay distintos tipos de sistemas de recomendación, entre estos se encuentran los *"sistemas de recomendación basados en contenido"* o los *"sistemas de recomendación colaborativos"*.

El sistema de recomendación que se ha realizado para el aplicativo, está basado en una técnica heurística la cual valorando distintos tipos de criterios añade al itinerario a generar unos puntos de interés u otros. Se ha realizado con el lenguaje de programación *Javascript* [25].

Al algoritmo de recomendación, le entran tres ficheros JSON, uno con datos acerca de los puntos de interés, uno con los tiempos entre todos los pares de estos y uno con las preferencias del viaje con las que se quiere generar el itinerario. Y finalmente, el algoritmo acaba proporcionando un itinerario turístico personalizado generado, el cual almacena en un fichero JSON de salida. El JSON de salida del itinerario turístico, es un itinerario completo con distintas visitas organizadas por días a puntos de interés, con horas de llegada y de salida, y con lo que cuesta la visita. También proporciona distintos indicadores que ayudan a valorar el itinerario (Serán mencionados más adelante).

En el siguiente enlace se puede observar una carpeta de Google Drive con dos subcarpetas, *"Entradas"* y *"Salida"*, donde se muestran ejemplos de los ficheros JSON de entrada y del fichero JSON de salida, respectivamente:

https://drive.google.com/drive/folders/1uYD5I1oYSH37vN3CBNB3Fe75M52m2Pot?usp=sharing

En el algoritmo de recomendación creado, lo primero que hace es recoger los datos de los archivos indicados como entrada, que contienen los datos necesarios para la generación del itinerario.

A cada punto de interés se le da una puntuación de 0 a 1 con respecto al tiempo al que están del hotel en el que el turista se va a alojar. Los tiempos en el algoritmo se calculan suponiendo que 1 kilómetro se recorre en 1 minuto (Esta suposición se puede cambiar ajustando la fórmula, ya que fue la que se tomó inicialmente y la que se mantuvo). Por otro lado, las distancias se calculan en kilómetros a partir de una fórmula que utiliza la latitud, la longitud y el radio del planeta, a esta fórmula se le denomina *"Fórmula de Haversine"*.

Esta puntuación calculada, se suma con otra puntuación de 0 a 1 que corresponde a cómo se relacionan las categorías del punto con las actividades que el turista quiere realizar. Por ejemplo:

- Si un punto de interés tiene como categoría *[SUNANDBEACH]*, el turista tiene como actividades a realizar *[SUNANDBEACH, RURAL]*, y, a su vez, es el punto más cercano, este tendrá una puntuación de 2 (1 por acoplar perfectamente con las actividades del turista y 1 por ser el punto más cercano).

- Si un punto de interés tiene como categorías *[SUNANDBEACH, RURAL]* y el turista tiene de actividades que quiere hacer *[SUNANDBEACH]*, la puntuación correspondiente a esta relación será de 0.5, porque el punto de interés acopla a la mitad de sus actividades. Ya que el punto no es solo de *SUNANDBEACH*, sino que también tiene *RURAL*, actividad que el turista no quiere hacer. Posteriormente a esta puntuación se le sumará la puntuación con respecto a la distancia y si fuera el punto más cercano al alojamiento del turista, su puntuación sería de 1.5 (1 de la distancia y 0.5 del acoplamiento con las actividades del turista).

Una vez hayan sido calculadas las puntuaciones, se ordenan los puntos de interés acorde a su puntuación y se van preparando todas las variables necesarias para que se ejecute el bucle que genera el itinerario turístico.

Entre estas variables que se preparan se tienen en cuenta varios factores:

- Si el turista llega al destino a una hora la cual no tiene un margen de 7 horas con la hora de finalización de turismo que indicó en las preferencias, no se le prepara itinerario para el día de su llegada. Pero sin embargo, si tiene ese margen de horas, sí se le prepara itinerario para ese día, pero se le prepara 3 horas después de su llegada ya que se considera que este tiempo es el que el turista necesita para adaptarse al lugar, dejar sus maletas en el hotel, etc.

- Si el turista se va del destino antes de las 13:00 horas, no se le prepara itinerario para ese día, debido a que se le prepararía un itinerario demasiado corto.

Posteriormente empieza el bucle que genera el itinerario turístico múltiple. Este bucle no para hasta que no se ha completado un itinerario para cada día que el turista está de viaje. Dentro de este, hay otro bucle que sirve para generar el itinerario turístico simple de cada día. Y no termina hasta que se hayan completado las horas de turismo (hora de finalización de turismo - hora de inicio de turismo) o hasta que no se hayan comprobado todos los puntos de interés registrados.

En cada iteración del bucle que genera los itinerarios de cada día, se va mirando un punto de interés aleatorio entre los 30 primeros (límite puesto) del vector de puntos de interés ordenados por puntuaciones . Si estos 30 están valorados y no se pueden introducir porque están cerrados, están a una distancia mayor de la que el turista se puede permitir moverse debido al transporte, etc. o se han usado (es decir, no se puede introducir ninguno), se va incrementando el límite de 10 en 10 (valorándose posteriormente los 40 primeros, 50..). Si el límite llega al número de puntos de interés registrados, significa que se han valorado todos los puntos y, están usados o no pueden introducirse en el itinerario. Por lo que se deja el tiempo restante hasta la hora de finalización de turismo como tiempo ocioso y se pasa a generar el itinerario turístico simple del siguiente día.

Cuando se escoge un número correspondiente al índice de un punto de interés, se valoran para estos los criterios en el siguiente orden:

1. Si el turista no tiene transporte no se le recomiendan puntos que le queden a más de 30 minutos de su alojamiento.

2. Si el día que se está generando el itinerario el punto de interés no está abierto se descarta.

3. Si la hora a la que va a encontrar el turista en el punto está cerrado se descarta.

4. Si las horas medias a estar en el punto no superan la hora límite máxima del viajero para hacer turismo, si al turista le queda presupuesto como para visitar dicho punto y si el punto no se ha visitado, se introduce en el itinerario.

Cuando se introducen los puntos de interés en los itinerarios simples se introducen como visitas (la primera y la última visita de cada día siempre es el alojamiento), las cuales cuentan con:

- Hora de llegada al punto de interés
- Hora de salida del punto de interés
- Punto de interés a visitar

Una vez se generan los itinerarios simples, se introducen en el itinerario múltiple, junto con el día en el que se hará el itinerario y, con los indicadores acerca del tiempo ocioso, del tiempo dedicado al turismo y del tiempo dedicado al transporte, con respecto a las horas disponibles del turista para hacer turismo.

Cuando se hayan completado todos los días, el algoritmo devuelve el itinerario turístico múltiple, junto con indicadores del itinerario completo que se van recogiendo a lo largo de la generación del itinerario, los cuales son:

- Tiempo y porcentaje dedicado al turismo
- Tiempo y porcentaje dedicado al transporte
- Tiempo y porcentaje ocioso

6. Experimentación del aplicativo

Para comprobar el correcto funcionamiento de la aplicación web, se realizará una experimentación donde se plantean dos escenarios, los cuales se basan en dos turistas diferentes con preferencias de viajes y destinos distintos.

Para comenzar, lo primero que se deberá realiza es ejecutar el siguiente comando:

```
sudo COMPOSE_PROFILES=dev docker-compose up
```

Este comando iniciará la aplicación con el perfil de desarrollador, por lo que levantará el contenedor de *MongoDB* para la base de datos, con una base de datos vacía. Y por otro lado el contenedor del back-end (que se conectará a la base de datos mencionada) y del front-end.

Una vez iniciada la aplicación se irá a la dirección http://localhost:8081, se verá la pantalla inicial, que se muestra en la figura 6.

Figura 6. Pantalla Home del aplicativo en inglés

Como se puede observar el selector que se encuentra a la derecha de la *"toolbar"*, sirve para cambiar el idioma. Para esta experimentación, será cambiado a Español (*"es"* en el selector). Por lo que la pantalla pasará a ser como se ve en la Figura 7.

Figura 7. Pantalla Home del aplicativo en español

En cuanto a los botones superiores, se encuentra un botón con un icono de un avión, otro con un ícono de dos personas y otro con un icono de una ruta y un marcador. Estos botones sirven para dirigir al usuario a la gestión de preferencias de viajes, de turistas o de puntos de interés, respectivamente.

Inicialmente, se deberán insertar los puntos de interés. La manera más cómoda de realizar dicha función es insertando los puntos de interés masivamente a través de un fichero. Por lo que lo primero que se hará para esto es dirigirse a la carpeta del proyecto *"excel_to_json"*, la cual cuenta con el script de *Python* que transforma la información acerca de 274 puntos de interés de Tenerife, disponible en el archivo *Excel* que se encuentra en la carpeta *"data"*, a un formato determinado en un archivo JSON con la estructura necesaria para su introducción masiva. Para la creación de este fichero de salida se ejecutará el siguiente comando:

```
python3 excel_to_json_pois.py -i ../data/Recursos\ Turismo\ Azul\
                Tenerife.xlsx -s Recursos\ TMCTF
```

La ejecución de este comando, creará en la carpeta *"data"* el archivo "*data-pois.json"*, con la información que se puede observar en la figura 8.

```
[
  {
    "town": "Buenavista del Norte",
    "postalCode": 38480,
    "categories": [
      "CULTURE"
    ],
    "name": "Faro de Punta de Teno",
    "description": "Faro de envergadura media rojo y blanco. Se encuentra ubicado en un cabo saliente en el punto m\u00e1s meridional de la isla, y separa los mares
    "applications": "Referencia y aviso costero para navegantes. Punto de inter\u00e9s cultural",
    "stayHourNumber": 3,
    "averageCost": 10,
    "openingDays": [
      "MONDAY",
      "TUESDAY",
      "WEDNESDAY",
      "THURSDAY",
      "FRIDAY",
      "SATURDAY",
      "SUNDAY"
    ],
    "geolocationPoint": {
      "type": "Point",
      "coordinates": [
        "-16.923003",
        "28.341981"
      ]
    }
  },
  {
    "town": "Buenavista del Norte",
    "postalCode": 38480,
    "categories": [
      "SUNANDBEACH"
    ],
    "name": "Charcos de Punta de Teno",
    "description": "Peque\u00f1as pozas de agua marina de roca volc\u00e1nica situadas anexas a la Playa de Punta de Teno. El acceso es dificultoso, puesto que se l
    "applications": "Zona de ba\u00f1o local. Punto pesquero. Zona de buceo",
    "stayHourNumber": 5,
    "averageCost": 15,
    "openingDays": [
      "MONDAY",
      "TUESDAY",
      "WEDNESDAY",
      "THURSDAY",
      "FRIDAY",
      "SATURDAY",
      "SUNDAY"
    ],
    "geolocationPoint": {
      "type": "Point",
      "coordinates": [
        "-16.918513",
        "28.342735"
```


Figura 8. Fichero JSON obtenido a través de la ejecución del script de Python

Una vez creado el fichero, habrá que pulsar en el aplicativo web el icono de la ruta con el marcador, y se accederá a la pantalla de gestión de puntos de interés, que se observa en la figura 9.

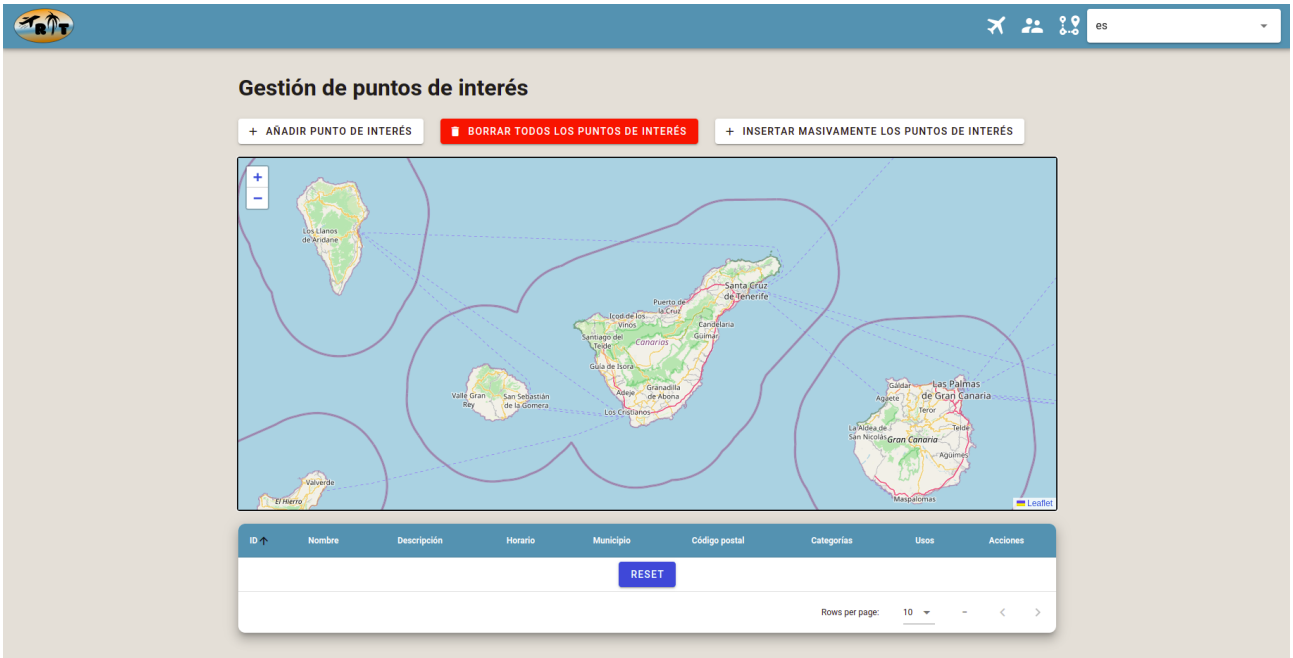


Figura 9. Pantalla de gestión de puntos de interés sin puntos registrados

Como se puede observar en la figura 9, no hay ningún punto de interés registrado. Para insertar masivamente los puntos se pulsará el botón que indica dicha acción, lo que desplegará una ventana emergente que se muestra en la Figura 10, donde se seleccionará el fichero obtenido con la ejecución del script de *Python* y se pulsará *"CONTINUAR"*.

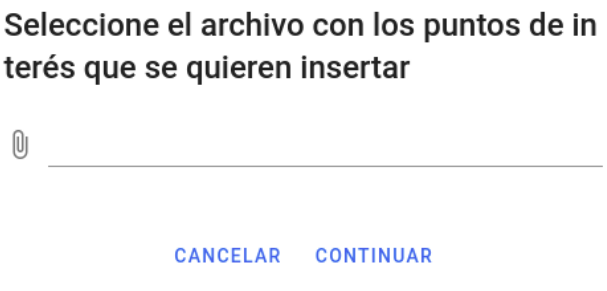


Figura 10. Ventana emergente de inserción masiva de puntos de interés

Una vez insertados, la pantalla de gestión se verá de la manera que se observa en la figura 11.

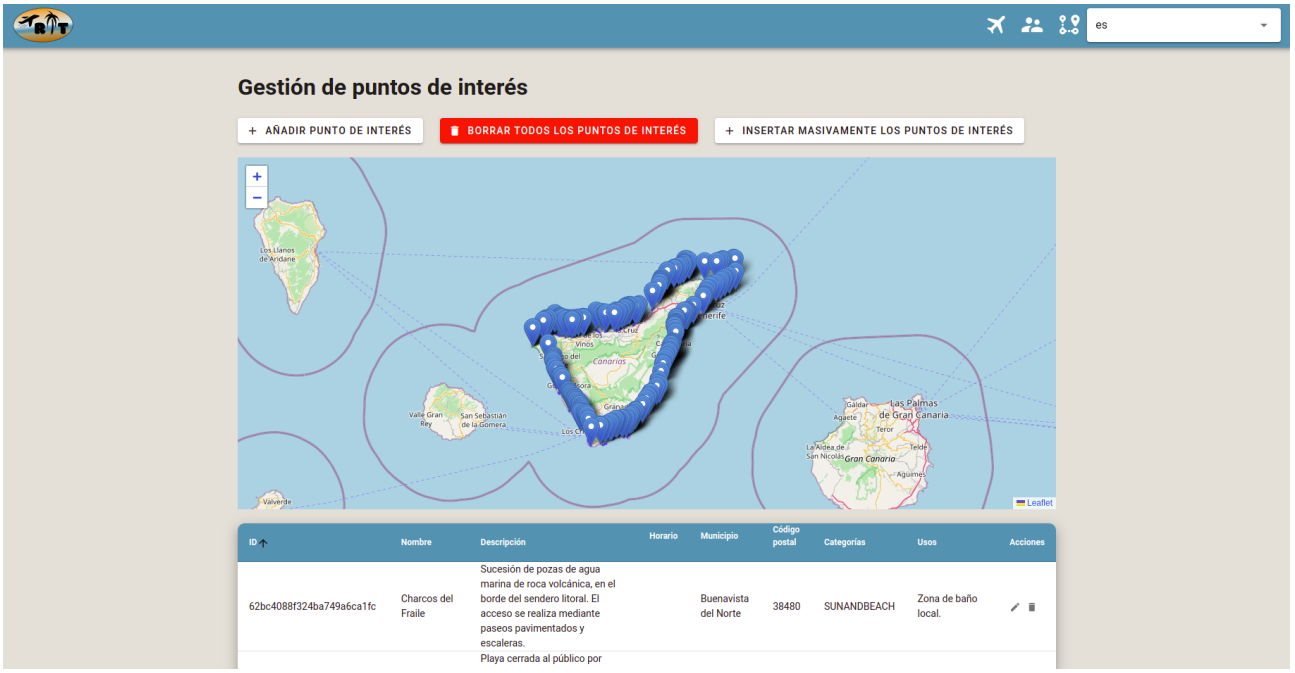


Figura 11. Ventana emergente de inserción masiva de puntos de interés

A continuación se muestran los dos escenarios para visualizar la generación de los itinerarios turísticos personalizados. La generación de estos se ve limitada debido a que el número de puntos insertados es bastante bajo para hacer una recomendación óptima (274) y la mayoría de los puntos son de la categoría *"SUNANDBEACH"*, ya que como se puede observar, al concentrarse en las zonas costeras de la isla, son bastantes playas y puertos.

Si se introdujera en vez de 274 puntos, 3000, y de categorías bastante variadas, se podría experimentar una recomendación mucho más adecuada a las preferencias del turista.

6.1. Escenario 1

Para este escenario, en primer lugar, se creará un turista. Para ello, se pulsará el botón con el icono de las dos personas, lo que redirigirá a la pantalla de gestión de turistas, mostrada en la figura 12.

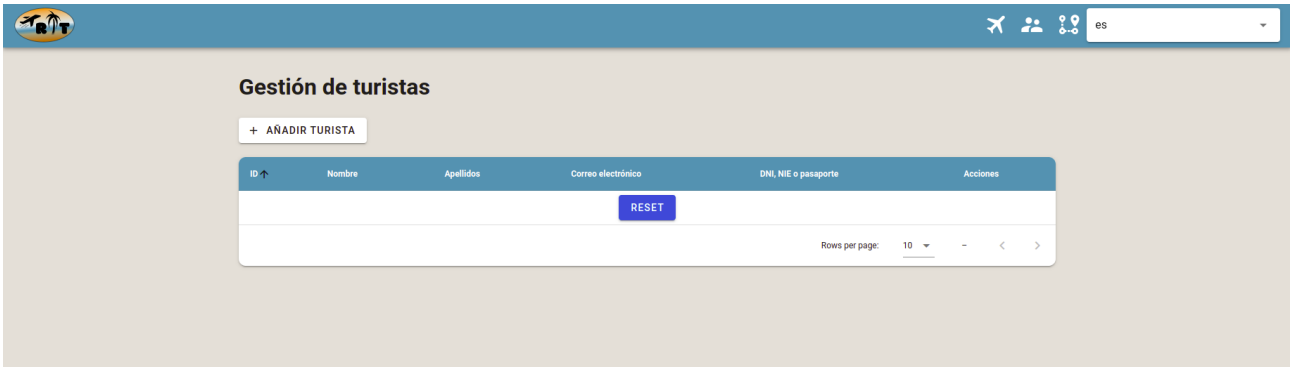


Figura 12. Pantalla de gestión de turistas sin turistas añadidos

Para registrar el turista se pulsará en el botón *"+ AÑADIR TURISTA"* y se completará el formulario con los datos mostrados en la figura 13, posteriormente se pulsará el botón *"AÑADIR"*.

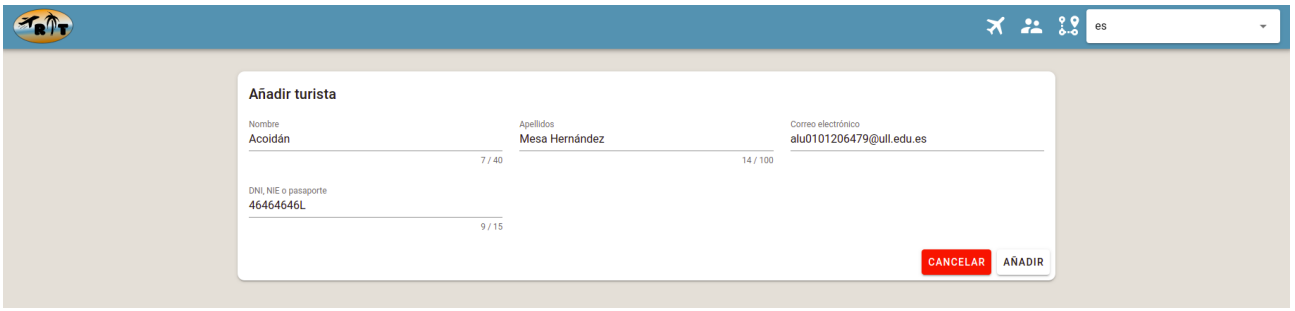


Figura 13. Formulario de creación de turistas

Cabe destacar que en la inserción de datos, tanto de turistas, como de preferencias de viajes o puntos de interés, si el sistema detecta algún fallo, como por ejemplo que la sintaxis del correo electrónico sea incorrecta, se indicará al usuario que lo está registrando.

Posteriormente, se deberá pulsar el botón del icono del avión para ir a la página de gestión de preferencias de viajes, que es prácticamente igual a la de turistas. En esta, pulsará el botón *"Añadir preferencias de viajes"*. Esto redirigirá al formulario de creación de preferencias de viajes, dónde se pondrán las que se muestran en la Figura 14 y se pulsará el botón *"AÑADIR"*

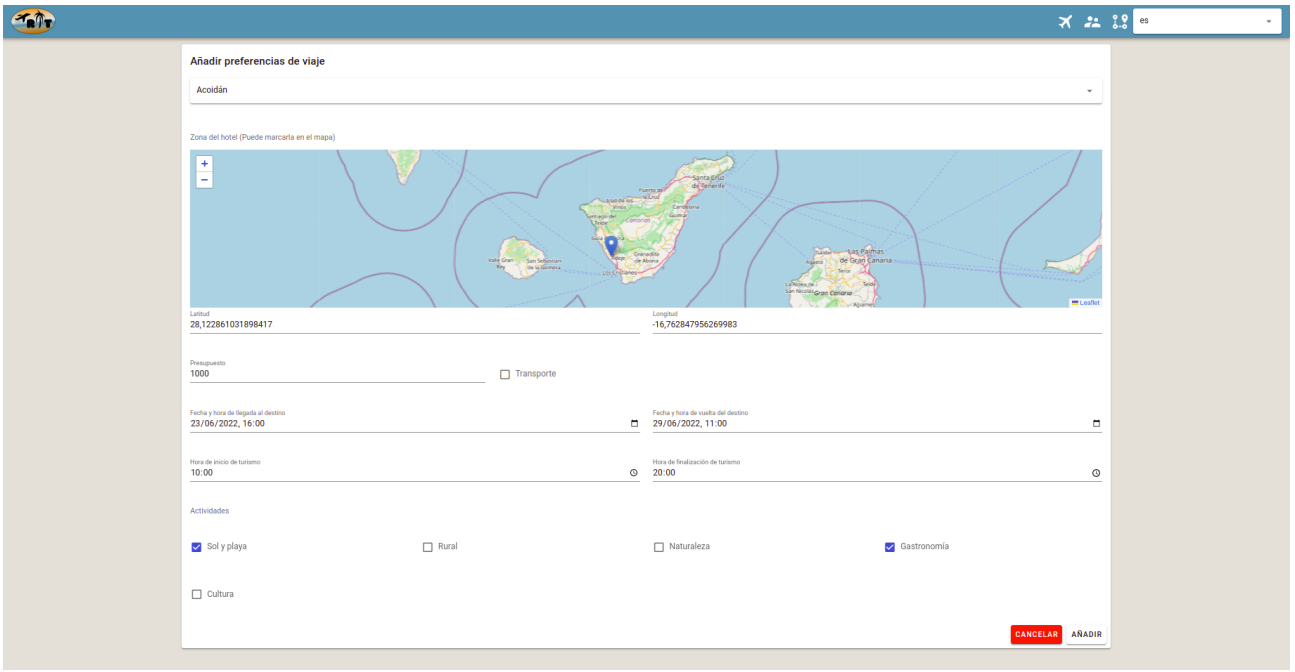


Figura 14. Formulario de creación de preferencias de viaje para el escenario 1 (https://bit.ly/3npP5kl)

Con las preferencias de viaje añadidas, se verá la entrada respectiva a estas, en la tabla con la que cuenta la pantalla de gestión de preferencias de viajes. Como se puede observar en la figura 15.

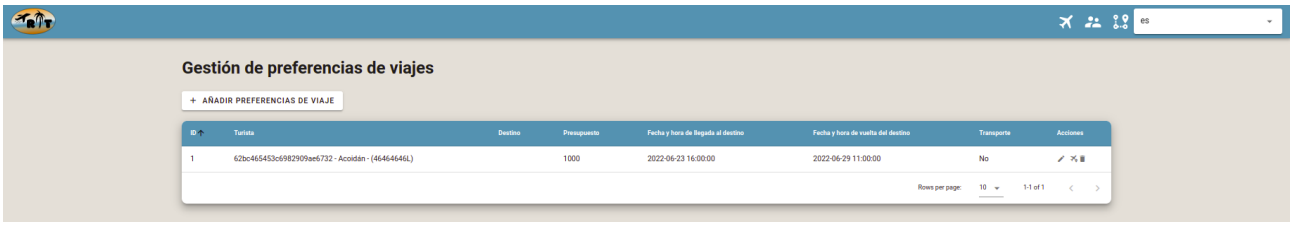


Figura 15. Pantalla de gestión de turistas con preferencias de viajes

En dicha figura, se puede observar a la derecha del registro un botón de un avión y un marcador, este es el que se pulsará para generar el itinerario turístico para estas preferencias.

El itinerario turístico personalizado generado se puede visualizar en el siguiente enlace:

https://drive.google.com/file/d/1hd4DGXFy9XDFMoOpzs06vAmQcfo0sbZl/view?usp=sharing

Si se analiza este itinerario, en primer lugar, se puede observar que para el día que llega el turista y para el día que se va, no se le organiza itinerario. Esto se debe a que el día que llega no tiene 7 horas de margen entre la hora a la que llega y la hora a la que finaliza de hacer turismo, ya que llega a las 16:00 y termina de hacer turismo a las 20:00 (4 horas de margen, que se consideran para que el usuario se hospede en el hotel, deposite sus cosas y conozca el lugar). Por otro lado, el día que se va no se le organiza itinerario porque se va antes de las 13:00 horas.

Además, cabe destacar que el itinerario se ha generado de la manera que se indica en el capítulo 5. Debido a este motivo, como se puede observar, se le ha generado un itinerario múltiple con un itinerario simple para cada día de su viaje, no excediendo nunca el rango de horas definidas para hacer turismo. Empezando y acabando siempre en el alojamiento.

Los itinerarios simples muestran toda la información acerca de los puntos de interés a visitar, como descripción y coste de estos. Estos puntos son recomendados también teniendo en cuenta el presupuesto del usuario. Este presupuesto se va descontando y se va valorando que el coste de los puntos no supere el presupuesto disponible del usuario antes de cada visita.

Otro aspecto importante a destacar, que ya se ha indicado anteriormente, es que, como el turista tiene como preferencias las actividades *"SUNANDBEACH"* y *"GASTRONOMY"*, y cómo la mayoría de puntos introducidos tienen de categoría *"SUNANDBEACH",* siendo menos de 10 los que tienen la categoría *"GASTRONOMY"*. Se le ha recomendado al turista casi en la totalidad del itinerario puntos de interés de categoría *"SUNANDBEACH"*. Pero sin embargo, como puso también *"GASTRONOMY"*, se le ha recomendado la "Lonja de los abrigos".

También se puede observar que se le han recomendado puntos cercanos a su zona de alojamiento, esto se debe a que se ha indicado que el turista no tiene transporte, por lo que no se le han recomendado puntos de interés que le queden a más de 30 minutos de su zona de hospedaje.

Los indicadores y porcentajes acerca de las horas dedicadas al turismo, al transporte y al tiempo ocioso, teniendo en cuenta el tiempo total del turista para hacer turismo (tanto de cada día, como global), indican el número de horas que el turista dedica a hacer turismo, al tiempo ocioso y a transportarse de un punto al otro.

6.2. Escenario 2

Para este escenario, se creará un turista (De la misma manera que se hizo en el escenario anterior), con los siguientes datos:

- Nombre: Martín
- Apellidos: Mesa Hernández
- Correo electrónico: alu0101141414@ull.edu.es
- DNI: 14141414M

Para este turista, se le crearán preferencias de viaje que se observan en la figura 16.

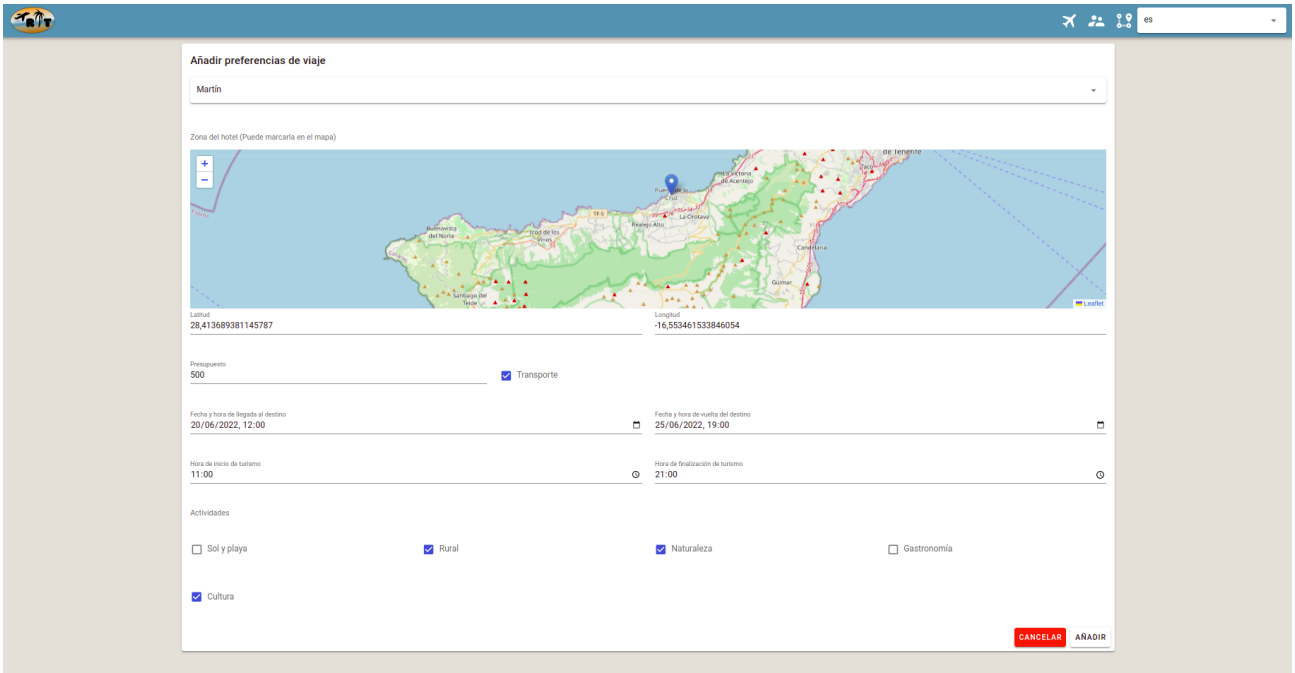


Figura 16. Formulario de creación de preferencias de viaje para el escenario 2 (https://bit.ly/3yvEJG5)

El itinerario que se ha generado a partir de estas preferencias es el que se encuentra en el siguiente enlace:

https://drive.google.com/file/d/1gRuF_h7c1dKS17WKqQKH6MB4uLZdPV74/view?usp=sharing

Hay varias cosas a tener en cuenta de este itinerario turístico generado:

- El día que llega el turista al destino se le asignó un itinerario, ya que llega a las 12:00 y termina de hacer turismo a las 21:00 (9 horas de margen). Es importante darse cuenta de que ese día el itinerario comienza 3 horas después de su llegada, ya que estas 3 horas están consideradas para conocer el lugar y poder hospedarse.

- El día que se marcha, también se le genera itinerario, ya que se va a las 19:00 de la tarde. Es importante darse cuenta que ese día el itinerario termina varias horas antes de la hora a la que se va. Esto se debe a que estas horas se consideran necesarias para el turista poder organizar su salida.

- En este escenario se han recomendado puntos más lejanos a la zona de hospedaje del turista, ya que al indicar que tiene transporte, se supone que tiene una capacidad de desplazamiento mayor.

- Respecto a las preferencias, se han recomendado en su mayoría, puntos de interés que se acoplan con las preferencias del turista. Hay algún punto de interés que no acopla exactamente con sus preferencias pero como puntúa bastante por la distancia, se ha recomendado. Como por ejemplo, la *"Lonja del Puerto de la Cruz"*, la cual no puntúa por las actividades que el turista quiere realizar, ya que tiene como categoría *"GASTRONOMY"* (la cual no se ha seleccionado entre las actividades que el turista quiere realizar), pero si puntúa por distancia, ya que se encuentra muy cerca de la zona de hospedaje.

7. Conclusiones y líneas futuras

Como conclusión a este trabajo y como se ha observado en el apartado de experimentación, entre los demás, se han cumplido los objetivos indicados en el capítulo 1, ya que se ha creado una aplicación que es capaz de generar itinerarios turísticos personalizados basándose en un sistema de recomendación que tiene en cuenta las preferencias de los turistas.

Como se ha indicado y como marcaban los objetivos, la información se gestiona mediante una API REST (que se encuentra en el back-end), la interfaz web cuenta con mapas interactivos, etc.

Se han presentado la mayoría de dificultades en el aprendizaje de las tecnologías, ya que han habido un gran número de errores, a los cuales se les ha dedicado muchas horas para hacer posible su arreglo. Por ejemplo, en los mapas interactivos, han habido muchísimas dificultades para el acoplamiento de GeoJson con Leaflet.

Todo es mejorable y debido a la limitación temporal, no se han podido realizar muchas mejoras en el aplicativo, pero se han anotado para proponerlas como líneas de futuro. Algunas de estas son las siguientes:

- Implantar un inicio de sesión para que los usuarios puedan tener sus cuentas y gestionar su propia información.
- Crear una interfaz de usuario que sea más orientativa al sector comercial, teniendo así dos perfiles de usuarios, administradores y no administradores.
- Integrar datos de transporte y condiciones meteorológicas.
- Incrementar el número de preferencias a valorar.
- Poder realizar reservas en los puntos de interés que la requieran desde la propia interfaz web.

La aplicación proporcionada es capaz de tener un gran impacto en el sector del turismo, ya que no se encuentran en abundancia y no suelen ser conocidas aplicaciones web que se dediquen a la generación de itinerarios turísticos personalizados.

Si al aplicativo se le aplican las líneas de futuro indicadas y se mantiene las funcionalidades con las que ya cuenta, proporcionará una gran satisfacción a los turistas que la usen. Y por lo tanto, es capaz de llegar a ser bastante conocido, teniendo un gran impacto en el sector turístico.

8. Summary and conclusions

As a conclusion to this article and as has been observed in the experimentation section, among the others, the objectives indicated in chapter 1 have been met, since an application has been created that is capable of generating personalised tourist itineraries based on a recommendation system that takes into account the preferences of tourists.

As indicated in the objectives, the information is managed by means of an API REST (which is located in the back-end), the web interface has interactive maps, etc.

There have been many difficulties in learning the technologies, as there have been a large number of bugs, which have taken many hours to fix. For example, in the interactive maps, there have been many difficulties in coupling GeoJson with Leaflet.

Everything can be improved and due to time constraints, it has not been possible to make many improvements to the application, but these have been noted and proposed as lines for the future. Some of these are as follows:

- Implement a login so that users can have their own accounts and manage their own information.
- Create a user interface that is more oriented to the commercial sector, thus having two user profiles, administrators and non-administrators.
- Integrate transport data and weather conditions.
- Increase the number of preferences to be assessed.
- To be able to make reservations at the points of interest that require it from the web interface itself.

The application provided is capable of having a great impact on the tourism sector, as web applications dedicated to the generation of customised tourism itineraries are not widely available and are not usually known.

If the application is applied to the indicated lines of the future and the functionalities it already has are maintained, it will provide great satisfaction to the tourists who use it. And therefore, it is capable of becoming well known, having a great impact on the tourism sector.

9. Presupuesto

El presupuesto se ha realizado teniendo en cuenta los distintos tipos de actividades realizadas, el número de horas empleadas y el coste por horas, cobrando 20 euros la hora de desarrollo y a 10 euros la hora de documentación. Esto se puede ver en la tabla 2.

Concepto	Horas	Coste total
Licitación de requisitos	70 h	1400 €
Desarrollo del aplicativo	180 h	3600 €
Documentación	60 h	600 €
Total	**310 h**	**5600 €**

Tabla 2. Presupuesto

10. Bibliografía

[1] E. Julca Meza, "Un enfoque a la importancia del turismo", *Turismo y Patrimonio*, no. 10, pp. 133-136, 2016. Disponible: https://doi.org/10.24265/turpatrim.2016.n10.09.

[2] R. Sharpley, *Tourism, Tourists and Society*, 5ª ed. Londres, 2018. Disponible: https://doi.org/10.4324/9781315210407

[3] F. Carner, "Encadenamientos generados por el sector turismo", *CEPAL*, 2001. Disponible: http://hdl.handle.net/11362/24140. [Acceso: 23 Junio 2022].

[4] F. Madrid y J. Díaz Rebolledo, "Del dato al relato en turismo", *Coronavirus y turismo*, Centro de Investigación y Competitividad Turística Anáhuac (CICOTUR), no. 6, 2020. Disponible: https://www.anahuac.mx/mexico/cicotur/sites/default/files/2020-03/Doc06_Coronavirus_Turismo_CICOTUR.pdf.

[5] M. Fernández Alles, "El impacto de la crisis sanitaria del covid-19 en el sector turístico español", *Desarrollo, Economía y Sociedad*, vol. 9, no. 1, pp. 36-40, 2020. Disponible: https://doi.org/10.38017/23228040.655.

[6] R. Llorente Heras, "Impacto del COVID-19 en el mercado de trabajo: un análisis de los colectivos vulnerables", Instituto Universitario de Análisis Económico y Social (IAES), 2020. Disponible: http://www.iaes.es/uploads/2/0/8/6/20860996/dt_02_20.pdf.

[7] B. Sandín, R. Valiente, J. García-Escalera and P. Chorot, "Impacto psicológico de la pandemia de COVID-19: Efectos negativos y positivos en población española asociados al periodo de confinamiento nacional", *Revista de Psicopatología y Psicología Clínica*, vol. 25, no. 1, p. 1, 2020. Disponible: https://doi.org/10.5944/rppc.27569.

[8] F. Bauzá Martorell y F. Melgosa Arcos, *Turismo post Covid-19: El turismo después de la pandemia global. Análisis, perspectivas y vías de recuperación*, 1ª ed. Salamanca: Ediciones Universidad de Salamanca, 2020. Disponible: https://doi.org/10.5944/rppc.27569.

[9] P. Vansteenwegen, W. Souffriau, G. Berghe y D. Oudheusden, "The City Trip Planner: An expert system for tourists", *Expert Systems with Applications*, vol. 38, no. 6, pp. 6540-6546, 2011. Disponible: https://doi.org/10.1016/j.eswa.2010.11.085.

[10] D. Gavalas, C. Konstantopoulos, K. Mastakas y G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems", *Journal of Heuristics*, vol. 20, no. 3, pp. 291-328, 2014. Disponible: https://doi.org/10.1007/s10732-014-9242-5.

[11] K. ten Hagen, R. Kramer, M. Hermkes, B. Schumann y P. Mueller, "Semantic Matching and Heuristic Search for a Dynamic Tour Guide", *Information and Communication Technologies in Tourism* 2005, pp. 149-159, 2005. Disponible: https://doi.org/10.1007/3-211-27283-6_14.

[12] I. Chao, B. Golden y E. Wasil, "A fast and effective heuristic for the orienteering problem", *European Journal of Operational Research*, vol. 88, no. 3, pp. 475-489, 1996. Disponible: https://doi.org/10.1016/0377-2217(95)00035-6.

[13] A. Gunawan, H. Lau y P. Vansteenwegen, "Orienteering Problem: A survey of recent

variants, solution approaches and applications", *European Journal of Operational Research*, vol. 255, no. 2, pp. 315-332, 2016. Disponible: https://doi.org/10.1016/j.ejor.2016.04.059.

[14] H. Salkin y C. De Kluyver, "The knapsack problem: A survey", *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127-144, 1975. Disponible: https://doi.org/10.1002/nav.3800220110.

[15] M. Jünger, G. Reinelt y G. Rinaldi, "The traveling salesman problem", *Handbooks in Operations Research and Management Science*, pp. 225-330, 1995. Disponible: https://doi.org/10.1016/S0927-0507(05)80121-5.

[16] H. Tang y E. Miller-Hooks, "A tabu search heuristic for the team orienteering problem", *Computers & Operations Research*, vol. 32, no. 6, pp. 1379-1407, 2005. Disponible: https://doi.org/10.1016/j.cor.2003.11.008.

[17] M. Kantor y M. Rosenwein, "The Orienteering Problem with Time Windows", *The Journal of the Operational Research Society*, vol. 43, no. 6, p. 629, 1992. Disponible: https://doi.org/10.2307/2583018.

[18] A. Gunawan, Z. Yuan y H. Lau, "A mathematical model and metaheuristics for Time Dependent Orienteering Problem", *Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling*, pp. 202-217, 2014. Disponible: https://ink.library.smu.edu.sg/sis_research/2669/.

[19] Z. Liao y W. Zheng, "Using a heuristic algorithm to design a personalised day tour route in a time-dependent stochastic environment", *Tourism Management*, vol. 68, pp. 284-300, 2018. Disponible: https://doi.org/10.1016/j.tourman.2018.03.012.

[20] J. Ruiz-Meza y J. Montoya-Torres, "A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines", *Operations Research Perspectives*, vol. 9, 2022. Disponible: https://doi.org/10.1016/j.orp.2022.100228.

[21] C. Verbeeck, P. Vansteenwegen y E. Aghezzaf, "An extension of the arc orienteering problem and its application to cycle trip planning", *Transportation Research Part E: Logistics and Transportation Review*, vol. 68, pp. 64-78, 2014. Disponible: https://doi.org/10.1016/j.tre.2014.05.006.

[22] D. Gavalas, V. Kasapakis, C. Konstantopoulos, G. Pantziou, N. Vathis y C. Zaroliagis, "The eCOMPASS multimodal tourist tour planner", *Expert Systems with Applications*, vol. 42, no. 21, pp. 7303-7316, 2015. Disponible: https://doi.org/10.1016/j.eswa.2015.05.046.

[23] D. Gavalas, C. Konstantopoulos, K. Mastakas, G. Pantziou y N. Vathis, "Efficient Heuristics for the Time Dependent Team Orienteering Problem with Time Windows", *Applied Algorithms*, pp. 152-163, 2014. Disponible: https://doi.org/10.1007/978-3-319-04126-1_13.

[24] D. Herzog, C. Laß y W. Wörndl, "TourRec — A Tourist Trip Recommender System for Individuals and Groups", *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 496-497, 2018. Disponible: https://doi.org/10.1145/3240323.3241612.

[25] A. Guha, C. Saftoiu and S. Krishnamurthi, "The Essence of JavaScript", *ECOOP 2010 – Object-Oriented Programming*, pp. 126-150, 2010. Disponible: https://doi.org/10.1007/978-3-642-14107-2_7.