TravelFit

Memoria de Proyecto Final de Grado

Multimedia

Ingeniería Web

Autor: Jorge Truchuelo de la Fuente

Consultor: Jordi Ustrell Garrigos

Profesor: Joan Soler Adillon

01/10/2019

Dedicatoria

A mi mujer Ana y a mi hijo Marcos, por su amor, por su apoyo, por todo el ánimo en cada momento, por su compresión y por el tiempo que me han dado, no solo en los últimos días si no desde el comienzo del grado. Sin vosotros este todo carecería de sentido y no habría sido posible.

Mi más profundo agradecimiento.

Abstract

El proyecto que se quiere llevar a cabo con el presente Trabajo Final de Grado (TFG) es el desarrollo de una aplicación web que sirva para crear rutas e itinerarios turísticos basados en las preferencias y gustos del usuario. La idea es que se trate de una herramienta útil y llamativa para que un viajero pueda organizar los diferentes hitos que quiera recorrer a lo largo de sus días de estancia.

Para el desarrollo de esta aplicación se emplearán principalmente las siguientes tecnologías: Para el backend se empleará tecnología PHP con el framework Laravel, mientras que para el desarrollo del frontend se utilizará tecnología HTML, CSS y Javascript conjugado con el framework Angular y Bootstrap para una visualización moderna.

Palabras clave: ingeniería web, aplicación web, viajar, turismo, ocio, SPA, PHP, Laravel, Angular, Bootstrap.

Abstract (English version)

The project to be carried out with this Final Degree Project (TFG) is the development of a web application that serves to create tourist routes and itineraries based on user preferences and tastes. The idea is that it is a useful and striking tool for a traveller to organize the different landmarks they want to travel throughout their days of stay.

The following technologies will be used mainly for the development of this application: PHP framework with the Laravel framework will be used for the backend, while HTML, CSS and Javascript technology combined with the Angular framework and Bootstrap for a modern visualization will be used for frontend development.

Keywords: web engineering, web application, travel, tourism, leisure, SPA, PHP, Laravel, Angular, Bootstrap.

Índice

Figuras y tablas

Índice de figuras

Índice de tablas

1. Introducción

1.1 Contexto y justificación del trabajo

Hoy en día se pueden encontrar multitud de sitios web para la reserva de vacaciones, hoteles, vuelos, etc. es fácil que hayamos visto anuncios por televisión u otros medios de empresas como Kayak, Trivago, Booking o Lastminute por ejemplo, para realizar estas tareas de manera autónoma a la hora de realizar nuestros viajes de forma alternativa a un agencia de viajes tradicional. También comenzamos a ver otros sitios web que ofrecen una gran cantidad de actividades, tours o experiencias para realizar durante la estancia, hablamos de web tan conocidas como TripAdvisor u otras más recientes como Civitatis.

También existen algunas aplicaciones web como Utrip, TripHobo o Inspirock con el objetivo de ayudar al viajero a buscar y planear las actividades y visitas durante su estancia, basadas en las preferencias del usuario. Es dentro de este marco donde TravelFit quiere entrar a competir, ofreciendo una alternativa llamativa y sencilla para la planificación de las actividades a realizar durante la visita a una ciudad.

En mi experiencia personal, he probado múltiples aplicaciones web de la misma índole y he encontrado múltiples fallos de funcionamientos o experiencias de uso no gratas, obligaciones de crear un usuario para poder hacer uso de la aplicación u otras trabas que me han hecho pensar en la ventana de oportunidad para crear una aplicación web para este nicho.

La idea que se quiere llegar a materializar en este TFG es la creación de una aplicación web SPA (*single page application*) mediante la cual se ofrezca un experiencia diferente y alternativa para conocer un destino vacacional, una aplicación desde la que un usuario pueda decidir si solo quiere ver los museos más importantes o los restaurantes de la ciudad que visita, una aplicación que, basándose en la oferta de ocio de una ciudad, los días disponibles y las preferencias del usuario, cree una ruta recomendada y personalizada. Todo ello sin apartar la mirada del uso simple, rápido y sólido que puede resultar clave para que un usuario la utilice.

2. Descripción

El objeto de este Trabajo Final de Grado (TFG) es el desarrollo de una aplicación web SPA que ayude al usuario a crear rutas turísticas a medida sobre puntos de interés basados en sus preferencias. De este modo, la aplicación podrá ser utilizada por cualquier persona como herramienta para organizar las actividades a realizar durante su estancia en un destino.

Así, la idea principal de este TFG es crear una herramienta para la organización de actividades y puntos de interés sobre un destino que vaya a visitar el usuario, donde éste podrá componer, gestionar y organizar su itinerario en base a sus preferencias y sobre la oferta de actividades turística existente en el destino.

A nivel técnico, la aplicación está dividida en un servicio de backend basado en API REST, creado sobre un el framework PHP Laravel y una BBDD MySQL De esta forma, se busca crear un servicio que pueda ser consumido por diferentes clientes como puede ser una aplicación web, una app móvil u otros.

La otra parte de la aplicación está constituida por el frontend, una aplicación web SPA basada en el framework de desarrollo Angular que actúa como cliente del API y que sirve una interfaz amigable, intuitiva y sencilla para la interacción por parte de un usuario. Para el diseño de la interfaz se emplea Bootstrap, framework CSS que otorga capacidades de adaptación a todo tipo de dispositivos y pantallas con un diseño limpio y actual.

Además, cabe mencionar que es necesario consumir servicios de terceros para proveer nuestra aplicación de datos de ubicaciones, destinos, puntos de interés, etc. En este caso se recurre al uso del Sygic Travel API para la obtención de la información relacionada con los destinos y los puntos de interés, del API Maptiler para obtener datos de autocompletado en campos de búsqueda de destino, y del API OpenStreetMaps para obtención de mapas de distintas ubicaciones.

De este modo, se espera crear una aplicación que pueda ser escalable y que pueda evolucionar a través de la ampliación de sus funciones básicas.

Esta memoria se presta a recoger aquellos puntos necesarios para el desarrollo de la aplicación, desde su concepción hasta su puesta en producción, para ello se describe a lo largo de los apartados todos y cada uno de los pasos que han sido necesarios para la elaboración. A modo de resumen, encontramos:

- Apartado 1 Introducción: Donde contexto en el que se sitúa la aplicación, así como la justificación que lleva al desarrollo de ésta.
- Apartado 2 Descripción: Objetivo e idea principal del TFG, así como una breve descripción de cómo se obtendrá el resultado final.
- Apartado 3 Objetivos: Objetivos principales y secundarios que se desean alcanzar al desarrollar la

aplicación.

- Apartado 4 Escenario: Análisis de aplicaciones similares en el mercado, así como conclusiones para tener en cuenta en nuestra aplicación.
- Apartado 5 Contenidos: Explicación de los distintos apartados/secciones que componen la aplicación
- Apartado 6 Metodología: Definición de la metodología de trabajo seguida para la elaboración de la aplicación.
- Apartado 7 Arquitectura de la aplicación: Descripción de la arquitectura sobre la que se constituye la aplicación y de las tecnologías empleadas.
- Apartado 8 Plataforma de desarrollo: Listado del equipo técnico empleado en el desarrollo de la aplicación:
- Apartado 9 Planificación: Calendario de trabajo y diagrama de Gantt
- Apartado 10 API's utilizadas: Detalle del uso de API de terceros para el funcionamiento de la aplicación.
- Apartado 11 Diagramas UML: Diagramas UML para la definición de los casos de uso y estructura lógica de las distintas entidades que componen la aplicación y la base de datos.
- Apartado 12 Prototipos: Prototipado de baja calidad para el diseño y arquitectura de la información de la interfaz de usuario.
- Apartado 13 Usabilidad/UX: Análisis de uso para un correcto desarrollo de la interfaz de usuario.
- Apartado 14 Seguridad: Medidas adoptadas para crear una aplicación segura frente a los distintos riesgos informáticos.
- Apartado 15 Requisitos de instalación / Implantación / Uso: Descripción de los pasos a seguir para implantar nuestra aplicación en un entorno de producción
- Apartado 16 Bugs: Listado de bugs detectados durante el desarrollo de la aplicación y las pruebas y test de uso en las distintas funcionalidades de la aplicación.
- Apartado 17 Proyección a futuro: Información, predicciones y sugerencias acerca de ampliaciones a futuro del trabajo, y/o lista de mejoras a realizar en hipotéticas futuras versiones del servicio/aplicación.
- Apartado 18 Conclusiones: Conclusiones extraídas a lo largo de desarrollo del TFG.

3. Objetivos

3.1 Principales

Los principales objetivos por alcanzar con el TFG son:

- Ser capaz de crear con éxito una aplicación web basada en tecnologías actuales que permita crear rutas personalizadas de ocio sobre un destino a gusto y criterio del usuario.
- Documentar y desarrollar todas las fases necesarias para el desarrollo de la aplicación web, cumpliendo con la ejecución total de un proyecto.
- Demostrar los conocimientos necesarios para ser capaz de ejecutar proyectos complejos.

3.2 Secundarios

Otros objetivos que se desean cumplir con el desarrollo de este TFG son:

- Adquirir experiencia en el desarrollo de aplicaciones web en todas las capas que lo componen (*Full Stack Developer*).
- Profundizar y afianzar el conocimiento adquirido a lo largo del grado Multimedia.
- Ser capaz de tener un aplicación completa y funcional en el tiempo estimado, cumpliendo los objetivos principales.

4. Escenario

Debido al expansión de Internet, el sector de las aplicaciones enfocadas a viajes y al ocio y el turismo asociados a estos, con diversidad de funcionalidades como la reserva y búsquedas de hoteles, transportes, tours, planificación, etc. han incrementado en número y se encuentra ampliamente extendido tanto a través de aplicaciones y páginas web, como a través de aplicaciones para dispositivos móviles.

En la actualidad existen diversas soluciones web que comparten objetivos y funcionalidad con TravelFit, es decir, la creación de itinerarios de actividades en un destino vacacional según las preferencias del usuario. Por ello, es de vital importancia analizar las distintas aplicaciones web existentes en el mercado actual con mismo o similar propósito.

Se escogen algunas de estas aplicaciones para poder crear un escenario de estudio y posterior planteamiento y desarrollo de la aplicación web. De las más de veinte opciones que se han encontrado, omitiendo las aplicaciones exclusivas para dispositivos smartphones o tabletas, nos basamos en las tres opciones que se han considerado las mejores opciones por diversos motivos como el funcionamiento, opciones, filtros, experiencia de uso, aspecto visual, etc.

TripHobo[1]: Con más de un millón usuarios y uno de los mayores repositorios de itinerarios de viaje planeados sobre esta aplicación por los mismos. Se trata de una de las mejores opciones que se han encontrado por sus amplias posibilidades.

Tabla 1. Ventajas / Desventajas TripHobo

Ventajas	Desventajas
Aplicación tipo SPA.	El diseño es simple y sobrio, no impacta visualmente, además existen inconsistencias en el diseño de la interfaz*.
Gran base de datos y repositorio de itinerarios creados por sus usuarios.	El manejo y modificación del itinerario creado, así como la ausencia de otras formas de ver la información de los distintos elementos que forman el itinerario.
Otras utilidades de navegación y descubrimiento de los contenidos por la aplicación web, a través de secciones como “Things To Do”, “Itinerarios”, “Hotels”, “Tours” o “Blog”.	
Posibilidad de reserva de alojamientos, espectáculos, tours o compra de tickets desde la misma aplicación.	Dificultad para reemplazar o agregar elementos que componen el itinerario actual y también en otras operaciones como cambiar las fechas de inicio/fin de este.

[1] *https://www.triphobo.com*

Herramienta de presupuesto de viaje.	El output final del itinerario es únicamente un calendario con las distintas actividades dispuestas a lo largo de las fechas de viaje, sin mostrar la información del itinerario en otro formato.
Gestión de usuarios.	
Funciones Sociales / compartir.	
Filtros por preferencia de categoría para la planificación del viaje.	
Puedes planear múltiples destinos por viaje	
Múltiples opciones y agregados durante el asistente de itinerario como poder escoger un vuelo, trasportes o alojamiento	
Sencillez de uso mediante un asistente de creación de itinerario.	
Listado de actividades por ciudad para agregar de manera sencilla como preferidas a la ruta.	

* inconsistencias en el diseño de la aplicación, donde se puede apreciar que en un caso los botones se encuentran pegados al campo de introducción y en otro caso no es así, variado según la sección de la aplicación. Incluso se puede apreciar en algún caso cierta curvatura en las esquinas y en el resto no.

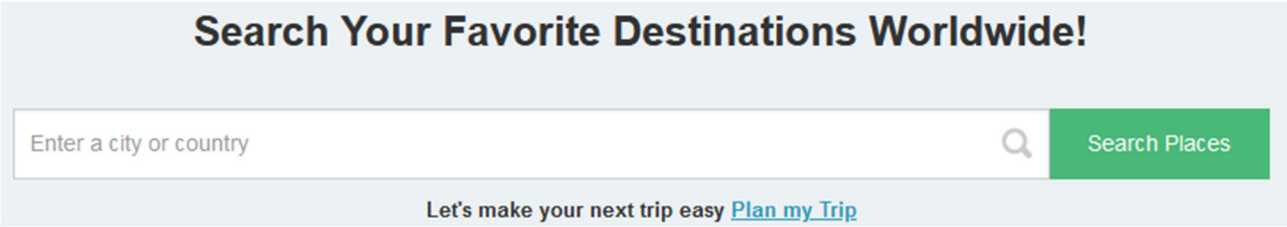


Figura 1. Campo de introducción y botones sección Things To Do

Figura 2. Campo de introducción y botones sección Itineraries

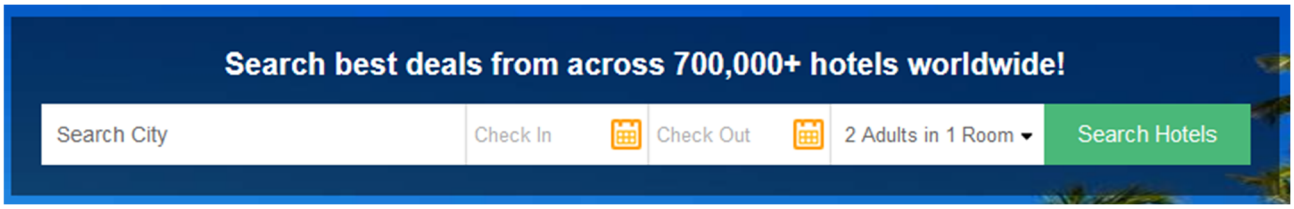


Figura 3. Campo de introducción y botones sección Hotels

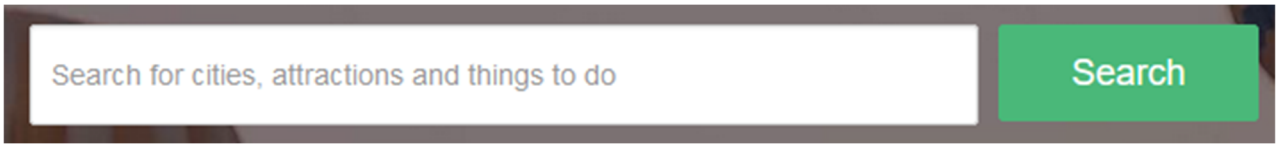


Figura 4. Campo de introducción y botones sección Tours

Inspirock[2]: Otra de las mejores opciones para la elaboración de itinerarios basados en las preferencias del usuario. Se caracteriza por tener unas amplias posibilidades de uso y buena experiencia en su manejo, se puede decir que es la mejor herramienta en este ámbito que hemos encontrado.

Tabla 2. Ventajas / Desventajas Inspirock

Ventajas	Desventajas
Aplicación SPA	La aplicación no te deja seleccionar ningún punto de interés o actividad preferido antes de la creación del itinerario, dando una opción inicial creada únicamente sobre unas pocas opciones de preferencias.
Herramienta Wizzard / asistente sencillo, con opciones simples y preferencias	
Potentes y sencillas opciones para agregar, eliminar o modificar el itinerario y sus parámetros	
Posibilidad de ver la información del itinerario desde un mapa, calendario o timeline	Aunque posee la posibilidad de buscar itinerarios de otros usuarios, esta se encuentra muy oculta y de una manera poco intuitiva*.
Puedes realizar reservas de alojamiento, restaurantes y compra de tickets de actividades	
Puedes incluir otras ciudades	
Facilidad de uso e interfaz sencillo e intuitivo.	
Muestra un resumen de todos los tickets necesarios a reservar/comprar del conjunto de actividades que forma el itinerario.	
Puedes ver información de las distintas actividades que componen el itinerario de manera sencilla.	
Conexión con TripAdvisor, una de las mayores plataformas de internet de recomendación de lugares, puntos de interés, vuelos, actividades o restaurantes entre otros.	
Chat online de ayuda	
Posibilidad de crear usuario para la gestión de los itinerarios	
Posibilidad de agregar a otro usuario como viajero en tu itinerario	
Posibilidad de compartir itinerario	
Puedes agregar eventos personalizados al itinerario	
Amplias capacidades de gestión de los itinerarios que un usuario tiene, aún sin identificarse en la plataforma. Soporte de la aplicación para operar en distintos idiomas y monedas.	

[2] *https://www.inspirock.com/*

* para poder buscar los itinerarios creados por otros usuarios hay que bajar al pie de la página y pulsar en *Plans by other users*, este es el único punto de acceso a dicha funcionalidad.

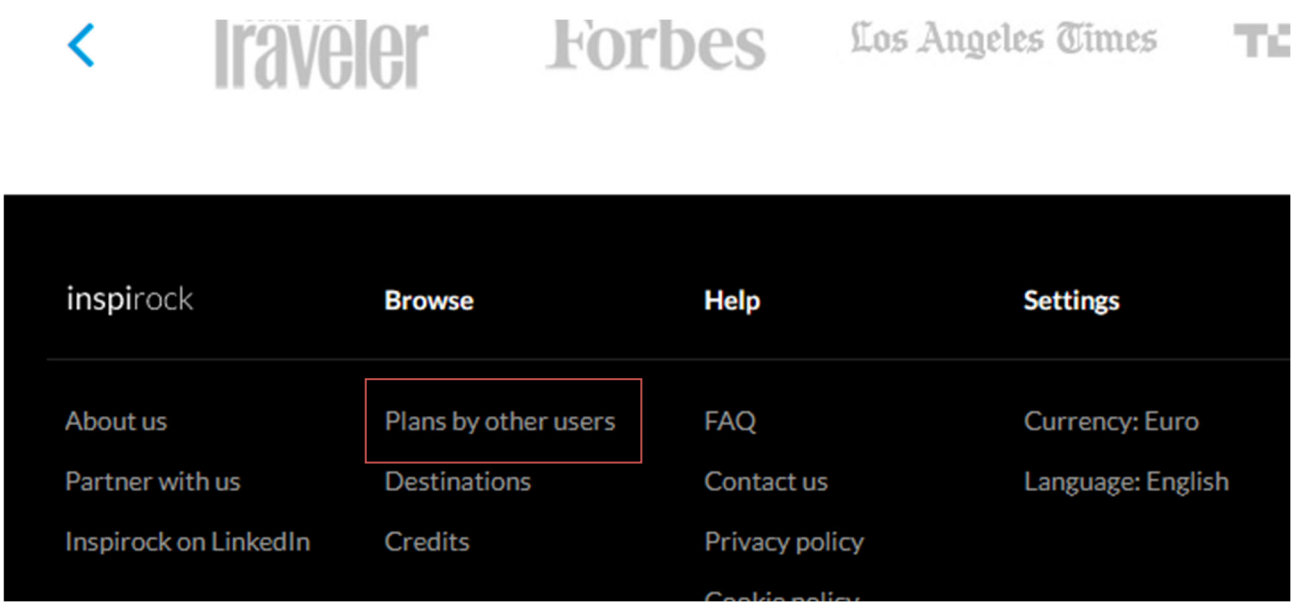


Figura 5. Enlace pie de página a sección Plans by other users Inspirock

Sygic Travel Maps[3]: Se trata de una herramienta para planificar los viajes diferente a las vistas hasta ahora, pues su metodología de uso cambia y no tiene ningún tipo de asistente que crea un itinerario inicia a medida según la elección de unas preferencias por parte del usuario. En esta herramienta es el usuario el que va navegando por los distintos puntos de interés que se divisan en un mapa o a través de una guía de la localidad, y es el usuario quien va agregando cada punto de interés a su viaje, el cual debe configurar según sus fechas y otros parámetros.

Tabla 3. Ventajas / Desventajas Sygic Travel Maps

Ventajas	Desventajas
Aplicación SPA	La gestión de los puntos de interés no es acertada, pues tienes que ir agregando una a uno a la planificación, indicando de manera manual el día y la hora a la que quieres ir, por lo tanto, es el usuario quien debe planificar de manera manual todo el itinerario*.
Es una herramienta fácil de usar e intuitiva.	
Posibilidad de usar plantillas predefinidas sobre la ciudad que se quiere visitar.	
Posibilidad de reservar hoteles y tours, así como alquiler de vehículos.	A la hora de gestionar varios viajes, no queda claro con el panel de viaje.
Gestión de usuarios y posibilidad de guardar viajes.	El uso de los puntos de interés favoritos no es intuitivo, ni son sugeridos en otros puntos de la aplicación, ni tampoco resaltan como una preferencia a incluir en el itinerario.
Funciones de compartir y para agregar más de un invitado al viaje.	
Se puede escoger múltiples idiomas, así como diferente moneda.	

* La gestión del itinerario se realiza a través de un panel en el cual debes ir pasando día a día, sin tener una imagen global del itinerario y sus actividades. Tampoco nada que invite a agregar un nuevo plan al día seleccionado, siendo la única manera escogiendo un punto del mapa y configurando que día y horas sucede.

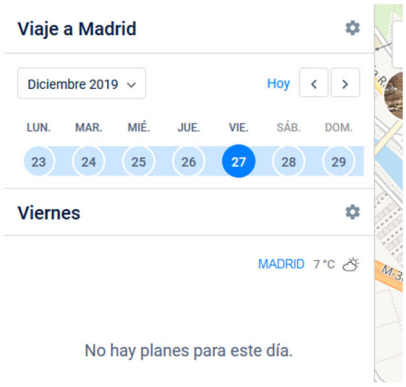


Figura 6. Panel itinerario Sygic Travel Maps

[3] *https://maps.sygic.com/*

Tras analizar las distintas herramientas presentes se ha llegado a las siguientes conclusiones:
Actualmente existe gran diversidad de herramientas en la red que comparten la misma función principal de uso, pero muchas de ellas son herramientas anticuadas, algunas de ellas dan errores de funcionamiento o se encuentran prácticamente abandonadas y/o desactualizadas. Por otro lado, un pequeño número de ellas, cómo las que se han analizado, son herramientas con un buen desarrollo y gran potencia de uso y amplias opciones de configuración.

De las herramientas que se han analizado se concluye que nuestra aplicación debe aportar las siguientes características:

- La aplicación debe tener un diseño moderno, sencillo y actual, con una usabilidad e interacción simple, además, es obligatorio que sea *responsive* para la adaptación a todo tipo de dispositivo y pantalla.
- Debe estar desarrollada con tecnologías actuales.
- Es buena idea el uso de un asistente de creación de itinerario o interacción mediante pasos.
- El uso de aplicaciones de tipo SPA hace una navegación y uso adecuado para este tipo de herramienta.
- La herramienta debe tener capacidad para gestionar usuarios y los itinerarios relacionados, pero es buena idea que el usuario pueda dar amplio uso aún sin estar registrado.
- La herramienta debe tener capacidad para gestionar los itinerarios y las actividades que los componen.
- Es vital que la gestión de los puntos de interés sea muy fácil y práctica.
- La información del itinerario y su composición debe ser de fácil interpretación, siendo clave en nuestra aplicación el uso de diferentes vistas. Además, su manipulación debe ser también fácil para el usuario.
- Las capacidades de compartir y reutilizar itinerarios ya creados por ti u otros usuarios es clave para destacar frente a la competencia. Debe ser una funcionalidad prioritaria.
- Es clave el uso de una adecuada base datos de puntos de interés, por lo que se debe contemplar el uso de BBDD de terceros para la obtención de datos.

5. Contenidos

Los contenidos que componen la aplicación en sus diferentes posibilidades de uso se describen a continuación:

- **Portada/Inicio**: Se trata de la página inicial que contiene la presentación de la aplicación. Contiene una sección para comenzar a preparar un itinerario y otra sección destinada a la presentación de las últimas rutas registradas en el sistema. Desde esta página el usuario de la aplicación puede iniciar un nuevo itinerario, buscar en el repertorio de itinerarios de otros usuarios o acceder a otras secciones a través del menú de navegación superior.

- **Contacto**: Esta página contiene la información de contacto con la empresa para reportar problemas o realizar cualquier tipo de consulta. Está compuesta por un formulario de contacto. El usuario puede rellenar el formulario de contacto y enviarlo.

- **Itinerario**: Muestra la información de itinerario en sus diferentes vistas. Desde esta misma sección y en cada una de sus vistas (resumen, timeline, mapa), el usuario puede ver distintos aspectos y detalles del itinerario, y realizar ajustes sobre el itinerario.
  Esta misma sección sirve para mostrar el detalle de los itinerarios creados por otros usuarios, pero no se podrá realizar ninguna modificación.
  Es la página núcleo de la aplicación y donde reside la mayor complejidad y experiencia de uso.

- **Itinerarios**: Muestra un listado de los itinerarios creados por otros usuarios. En esta página, el usuario puede navegar entre los distintos itinerarios que están compartidos por la comunidad de usuarios de la aplicación. El usuario puede acceder a ver el detalle de un itinerario de la lista.

- **Mis Itinerarios**: Muestra la lista de itinerarios creados por el usuario. Con opciones de gestión como la eliminación. Esta sección existe únicamente para usuarios registrados dentro de la aplicación.

- **Editar perfil**: Esta sección contiene la información del usuario y ciertas opciones de gestión como cambiar la contraseña o el nombre de usuario.

- **Registro de Usuario**: Esta página es para que el usuario se pueda registrar en la página web. Desde esta sección, el usuario puede rellenar el formulario de registro para darse de alta como usuario de la aplicación.

En todo momento, el usuario tiene la capacidad iniciar sesión y acceder a las secciones de usuario registrado desde cualquier punto de la aplicación, desde la barra de navegación superior, elemento presente de manera constante en la aplicación.

6. Metodología

Para la elaboración y desarrollo de este TFG se ha decidido seguir una metodología tradicional debido a que se realiza dentro del marco de un proyecto académico definido por cuatro hitos / fases y una defensa final, es por ello por lo que las distintas etapas que seguirá este TFG se basarán en la metodología tradicional en cascada.

Figura 7: Esquema metodología en cascada.

En el trascurso del TFG seguiremos las siguientes etapas:

- Análisis: En esta primera fase, se crea la definición del proyecto a realizar. Se detalla el alcance y los requisitos necesarios para el correcto desarrollo de la aplicación web propuesta.
- Diseño: Diseño y prototipado del producto, así como de las funciones y componentes de la aplicación. Además, se definirá la arquitectura y la plataforma de desarrollo y se crearan los diagramas UML y mockups de la aplicación.
- Implementación: Se efectúa la implementación de la aplicación web según el análisis y diseño previos.
- Pruebas: Se ejecutarán las pruebas necesarias para comprobar el correcto funcionamiento de la aplicación.

Se darán múltiples iteraciones sobre cada una de las etapas con la finalidad de mejorar cada punto que presenta el desarrollo de este TFG, así como para realizar diversos ajustes en el alcance y funcionalidad de la aplicación, de este modo se busca conseguir un grado de desarrollo alto y ajustado a las capacidades y tiempo necesario para la implementación total.

De manera paralela al desarrollo de las etapas anteriormente descritas, se realizará estudios para adquirir el conocimiento necesario para aplicar las tecnologías citadas. Se creará una metodología de trabajo personal enfocada en ir resolviendo los diferentes retos que plantea desarrollar la aplicación planteada, y se preparará un entorno de desarrollo personal para obtener un workflow optimizado.

7. Arquitectura de la aplicación

El diseño de la aplicación web está basado en la arquitectura multinivel, es decir, en la programación por capas. Las cuales se dividen en capa de presentación, de negocio y de datos.

- **Capa de presentación**: representada por una SPA que se encargará de UX/UI y representa el cliente, responsable de realizar las peticiones HTTP a la capa de negocio y recibir la respuesta en formato JSON. Esta capa es el frontend de la aplicación, su implementación esta realizada en Angular.

- **Capa de negocio**: compuesta por una aplicación web que implementa una interfaz de programación de aplicaciones (API) con arquitectura REST. Esta capa tiene la responsabilidad de atender las peticiones del cliente a través de protocolo HTTP, aplicar la lógica principal del programa y gestionar los accesos necesarios a la capa de datos, para dar la respuesta a la petición del cliente en formato JSON. El desarrollo de la capa de negocio se basa en el framework Laravel.

- **Capa de datos**: formada por una SGBD, alberga los datos de la aplicación y gestiona el acceso y la modificación de los estos a petición de la capa de negocio, comunicación que se realiza a través de consultas SQL. En este caso, esta capa está implementada por el SGBB MySQL.
  La capa de negocio junto con la capa de datos forma el backend de la aplicación web.

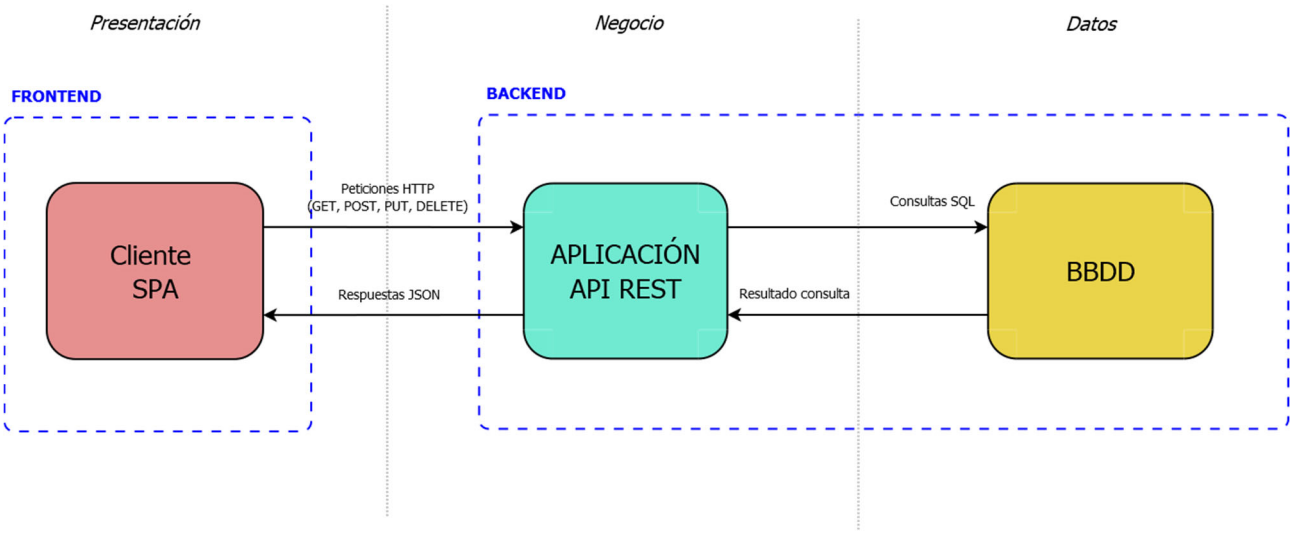


Figura 8. Esquema de aplicación en 3 capas.

La finalidad que se busca es aplicar estrategias de comunicación entre el cliente y el servidor en la que el documento HTML se cargue una única vez y los datos sean gestionados mediante tecnología AJAX. Estrategia ampliamente extendida que se emplea para mejorar la experiencia de usuario con la aplicación, además de evitar la carga de renderizado en el lado del servidor. Esta en la base del desarrollo de la web moderna y de las *Progressive Web Application* o PWA.

7.1 Backend

Laravel: Se trata de un framework de código abierto basado en PHP que sirve para la creación de aplicaciones y servicios web. Su estructura de aplicación se basa en el patrón de arquitectura MVC, que establece una separación de responsabilidades para los datos, la lógica de negocio y su representación en diferentes componentes. En este caso, el componente de la vista no será usado, debido a que emplearemos una arquitectura REST para la comunicación de la aplicación frontend con la aplicación backend.

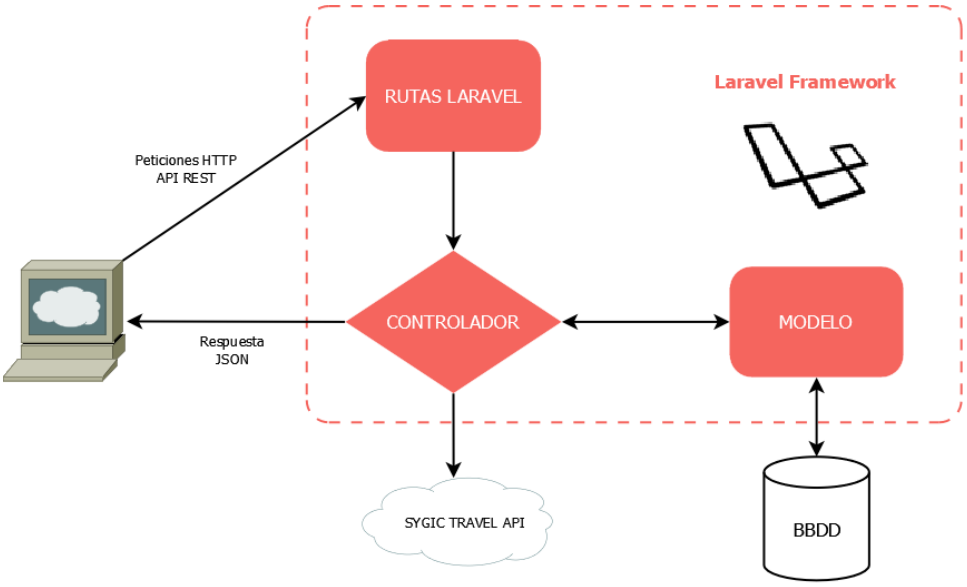


Figura 9. Esquema aplicación Laravel

API REST: La arquitectura de REST describe una interfaz estándar de comunicación entre sistemas que emplea el protocolo HTTP para realizar peticiones sin estado, dentro de un marco definido de acciones muy concretas para obtener y operar con datos pasados habitualmente en formato XML o JSON.
Se puede decir que una aplicación es ***RESTful*** cuando cumple todas las reglas de la arquitectura REST.

Estas reglas son:

- Interfaz uniforme
- Peticiones sin estado
- Cacheable
- Separación de cliente y servidor
- Sistema de Capas
- Código bajo demanda (opcional)

Esta arquitectura será implementada a través de las rutas que ofrece Laravel y constará de las siguientes rutas:

Tabla 4. Rutas URI API Backend

Método	URI	Descripción
GET	api/itineraries	Obtener todos los itinerarios públicos
POST	api/itineraries	Agregar un itinerario nuevo
GET	api/itineraries/{itinerary}	Obtener un itinerario público
PUT	api/itineraries/{itinerary}	Actualizar los datos de un itinerario
DELETE	api/itineraries/{itinerary}	Eliminar un itinerario
POST	api/itineraries/{user}	Obtener todos los itinerarios particulares del usuario
POST	api/itineraries/{user}/{itinerary}	Obtener un itinerario privado
POST	api/login	Iniciar sesión de usuario
POST	api/logout	Cerrar sesión de usuario
GET	api/users/{user}	Obtener los datos de un usuario registrado
PUT	api/users/{user}	Actualizar los datos de un usuario registrado
POST	api/register	Registrar un nuevo usuario en la aplicación
POST	api/contact	Envía formulario de contacto
POST	api/new_itinerary	Crear un nuevo itinerario nuevo

Como se puede observar en el diagrama del backend, existe una conexión con la plataforma Sygic Travel API para la obtención de datos de destinos y puntos de interés. El uso de este API es imprescindible para el funcionamiento de la aplicación pues otorga una completa BBDD de datos sobre puntos de interés, lugares, etc. Por ello, se ha implementado un controlador que se encarga gestionar la información necesaria y devolverla de una manera adaptada a las necesidades requeridas para la aplicación TravelFit.

MySQL: Es un sistema gestor de bases de datos, representa la capa de datos de la aplicación. Se trata de una base de datos relacional que responde al modelo E-R y al uso de tablas para la estructuración de datos. Al tratarse de la persistencia del sistema general, sus principales funciones se rigen por CRUD (*Create, Read, Update and Delete*), acciones que se emparejan con las peticiones REST.

En la aplicación, la BBDD está compuesta por las tablas:

- **Itineraries**: en la cual se guardan los datos de un itinerario de un usuario. Se relaciona con la tabla usuario para conocer quién es el propietario de cada itinerario
- **Destinations**: para almacenar la información de un destino de un itinerario. Esta tabla se relaciona con la tabla *itineraries* dado que un Itinerario debe estar compuesto por al menos un destino.
- **Pois**: para guardar los datos de los puntos de interés referentes a un destino. Esta tabla se relaciona con *destinations* en cuanto que un destino ninguno, uno o múltiples puntos de interés.
- **Users**: Registra la información de un usuario en la plataforma.

- **Migrations**: tabla necesaria para el desarrollo del sistema, pues alberga la información de las modificaciones realizadas en diferentes etapas del desarrollo del sistema.

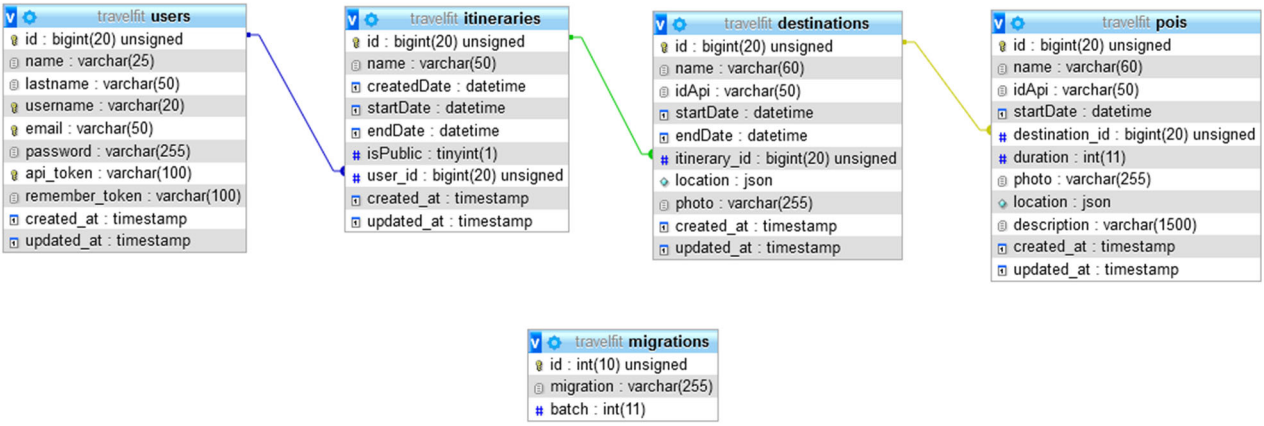


Figura 10. Diagrama de tablas de la base de datos

7.2 Frontend

Angular: Se trata de un framework de código abierto para desarrollar aplicaciones web SPA, enfocado al desarrollo de aplicaciones web situadas en el lado cliente, frontend, sigue los principios de la arquitectura MVVM, pudiendo operar también bajo una arquitectura MVC y está basado en el uso de componentes para la implementación de las aplicaciones.

La aplicación SPA está formada por múltiples componentes, representado cada uno de ellos una parte de la aplicación. La estructura principal de los componentes que forman la aplicación es la siguiente:

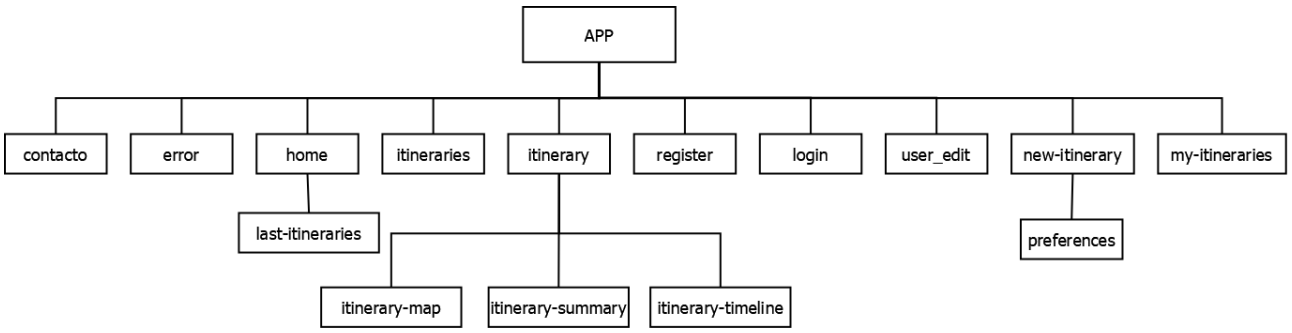


Figura 11. Diagrama de componente Angular frontend.

8. Plataforma de desarrollo

El entorno de desarrollo de la aplicación está dispuesto por la siguiente configuración:

Software:

- Sistemas operativos Windows 10
- Visual Studio Code v1.39
- Sublime Text 3.2.2
- Postman 7.9.0
- Git 2.23
- WAMP64 (Apache 2.4.39 + PHP 7.2.18 + MySQL 5.7.26)
- Firefox 70
- Chrome 78
- Composer 1.9.0
- Node.js 12.13.1
- Framework Laravel 6.0
- Framework Angular 8.0 / Angular CLI 8.3.0

Herramientas de respaldo online:

- GitHub
- OneDrive

Software adicional:

- Adobe Photoshop
- GanntProject
- Microsoft Office
- Balsamiq Mockups
- Dia

Hardware:

Equipo principal

- CPU Intel Core i7 3930k
- Memoria Kingston 32GB RAM 1333 Mhz
- Placa lógica Asus P9X79 Deluxe
- Disco SSD Samgsung 960 EVO
- Pantalla Dell U2713H

Equipo de apoyo

- MacBook Pro 15” Mid. 2009

9. Planificación

Para la correcta planificación del TFG se empleará la planificación orientada a objetivos, en la que se prima que debe conseguirse antes que cómo debe conseguirse. De este modo, se muestra a continuación un plan de hitos basado en las fechas dispuestas en el plan académico que recoge la ejecución de este TFG.

Tabla 5. Planificación TFG

Hitos	Fecha inicio	Fecha fin
PEC 1 – Definición y planificación del TFG	18/09/2019	01/10/2019
• Definición del proyecto	18/09/2019	20/09/2019
• Descripción y Objetivos	21/09/2019	22/09/2019
• Definición de metodología y contenido	23/09/2019	25/09/2019
• Planificación de proyecto	26/09/2019	27/09/2019
• Comienzo elaboración de memoria	21/09/2019	30/09/2019
• Entrega PEC 1	01/10/2019	01/10/2019
PEC 2 – Revisión pasos previos. Análisis y diseño de la aplicación. Comienzo del desarrollo.	02/10/2019	30/10/2019
• Estudio / Investigación tecnologías	05/10/2019	20/10/2019
• Preparación plataforma de desarrollo	21/10/2019	23/10/2019
• Comienzo implementación de la aplicación	24/10/2019	29/10/2019
• Definición de la arquitectura del sistema	01/10/2019	05/10/2019
• Diagramas UML Sistema	06/10/2019	13/10/2019
• Sketches y Mockups Web	14/10/2019	18/10/2019
• Estudio usabilidad / UX	19/10/2019	21/10/2019
• Entrega PEC 2	30/10/2019	30/10/2019

Hitos	Fecha inicio	Fecha fin
PEC 3 – Revisión pasos previos. Avance en el desarrollo de la aplicación.	31/10/2019	09/12/2019
• Revisión y modificaciones memoria PEC 2	04/11/2019	05/12/2019
• Desarrollo Backend	30/10/2019	15/11/2019
• Desarrollo Frontend	16/11/2019	01/12/2019
• Pruebas / Test y correcciones	02/12/2019	07/12/2019
• Entrega PEC 3	09/12/2019	09/12/2019
PEC 4 Entrega Final – Finalización del TFG (Memoria y proyecto). Elaboración presentación y defensa del proyecto.	10/12/2019	06/01/2020
• Revisión y modificaciones memoria PEC 3	14/12/2019	05/01/2020
• Finalización desarrollo Backend y Frontend	09/12/2019	23/12/2019
• Pruebas / Test y correcciones	24/12/2019	29/12/2019
• Elaboración presentación proyecto	29/12/2019	03/01/2020
• Entrega Final – FIN PROYECTO	07/01/2020	07/01/2020

A continuación, se muestra un diagrama de Gantt con la planificación detalla sobre la ejecución de las diferentes actividades a realizar en cada una de las distintas etapas. Solo cabe mencionar que se tomará como unidad de medida laboral diaria media jornada laboral.

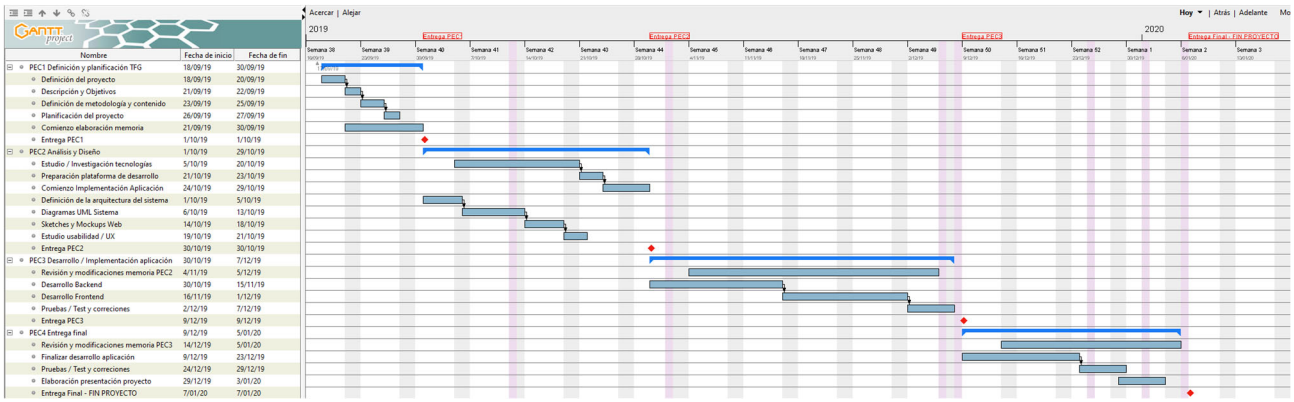


Figura 12. Diagrama de Gantt planificación desarrollo

10. APIs utilizadas

Sygic Travel API

La aplicación TravelFit requiere de una amplia base de datos con información de ubicación de ciudades/destinos, así como de puntos de interés de carácter turístico en cada uno de estos destinos. Es un punto clave, pues la calidad de esta información, así como su nivel de detalle, definen la calidad de los elementos que componen un itinerario en nuestra aplicación, por ello se recurre a una conocida y robusta base de datos del servicio Sygic Travel API[4]. La elección de este servicio ha sido debida a que ofrecía una relación adecuada en cuando al volumen de peticiones aceptadas y la cantidad y tipo de información que proporciona bajo un plan gratuito. Se han tenido que descartar otras opciones debido a la imposibilidad de obtener acceso para un desarrollo con carácter de aprendizaje o por ser de pago.

Para emplear esta API se ha integrado un controlador en el backend que se encarga de recibir las peticiones de nuestra aplicación, solicitar la información a Sygic Travel API y devolver los datos necesarios.

Sygic Travel API dispone de múltiples terminaciones de uso, la aplicación hace un uso principal de *Places*, una terminación destinada a la obtención de los datos de destino y puntos de interés desde la base de datos Sygic Places.
Las llamadas que se realizan contra el servicio son principalmente tres, y se realizan de manera consecutiva, según el controlador se encarga de obtener los datos y crear un itinerario según las peticiones del usuario. El procedimiento es el siguiente:

1) Llamadas para encontrar un destino y obtener su información básica:
   /places/list + parámetros: 'query' => nombre_destino, 'levels' => 'city|town|village', 'tags' => 'city|town|village', 'limit' => '1'
   A través de esta llamada, podemos localizar el destino introducido por el usuario, *nombre_destino*, dentro del sistema, para buscar a continuación sus puntos de interés.
2) Llamadas para la obtención de la imagen de un destino:
   /places/{id}/media
   Con esta llamada, obtenemos un listado de imágenes disponible sobre un destino, el sistema obtiene la primera imagen de los resultados.
3) Llamadas para la obtención de un listado de puntos de interés y su información correspondiente:
   /places/list + parámetros: 'parents' => destino, 'levels' => 'poi', 'limit' => número_max, 'categories' => lista_categorias, 'categories_not' => 'traveling'
   Por medio de esta llamada, obtenemos únicamente los puntos de interés de un destino, bajo las categorías proporcionadas por el usuario en la creación del itinerario.

---

[4] Página del producto https://travel.sygic.com/en/b2b/solutions y documentación del API http://docs.sygictravelapi.com/1.2/

Una vez obtenida toda la información, es el propio sistema de TravelFIt el encargado de realizar los algoritmos adecuados para la creación del itinerario y devolverlo al cliente.

Maptiler Geocoding API

Al comenzar un itinerario, es necesario comenzar con un destino, para ello, hemos empleado el servicio Geocoding de Maptiler[5] que proporciona un API con funcionalidad de auto relleno en el campo de introducción de destino. Ha sido necesario el uso de esta API puesto que Sygic Travel API no otorgaba dicha funcionalidad de manera nativa. Se ha escogido esta API frente otras opciones como Google Places API por ser una alternativa completamente funcional a los servicios de gran buscador.

El uso de esta API esta implementado en el frontend de la aplicación, a través de un componente y un servicio encargados de proporcionar resultados según el usuario escribe el nombre de un destino. Por lo que tan solo ha sido necesario implementar el componente dentro de la aplicación para su correcto funcionamiento.

OpenStreetMaps API

El origen de los mapas mostrados en TravelFit proviene de OpenStreetMaps[6] y el consumo de su API se realiza mediante el uso de la librería Javascript Leaflet[7] dentro del frontend de TravelFit.
Se ha escogido este API por resultar una alternativa potente y fácil de implementar dentro de la aplicación, sin necesidad de registros, y ofreciendo una calidad de servicio excelente bajo licencias de software open-source.

[5] *https://cloud.maptiler.com/geocoding/*
[6] *https://wiki.openstreetmap.org/wiki/API*
[7] *https://leafletjs.com/*

11. Diagramas UML

11.1 Casos de uso

A continuación, se muestran los diagramas de casos de uso presentes en la aplicación:

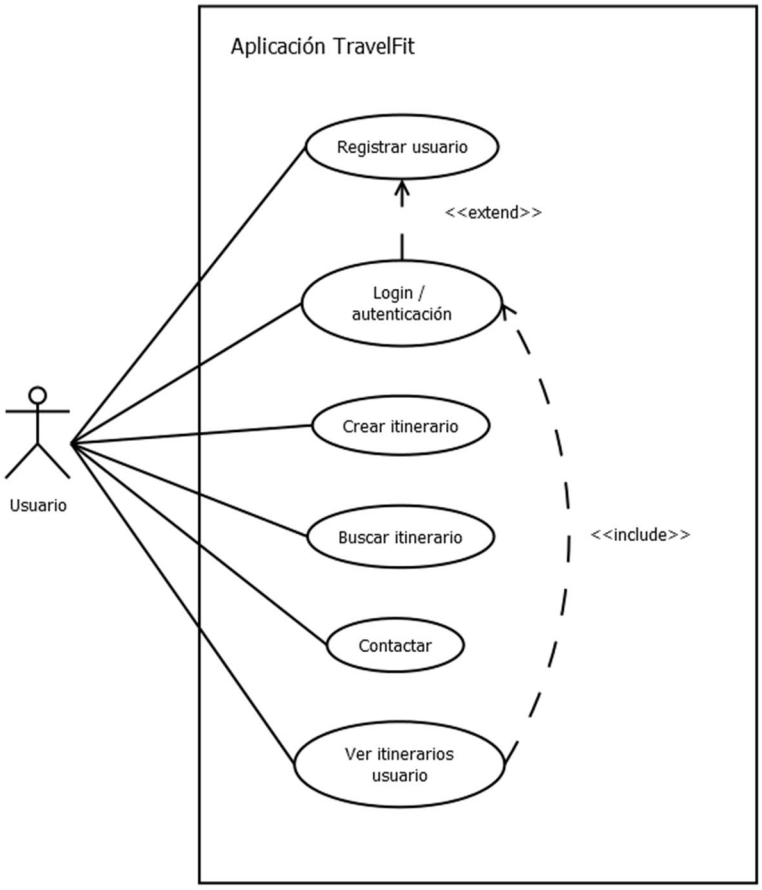


Figura 13. Caso de Uso general TravelFit

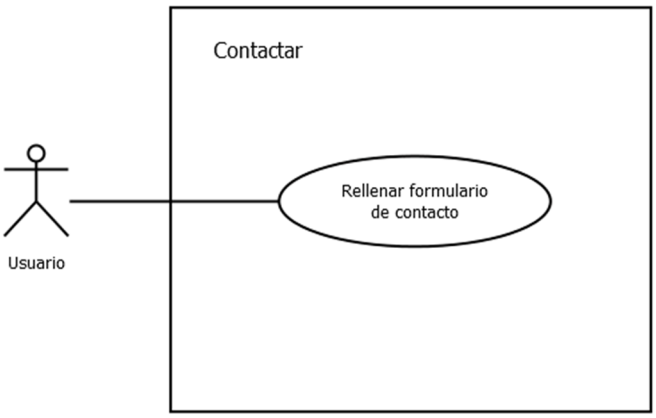
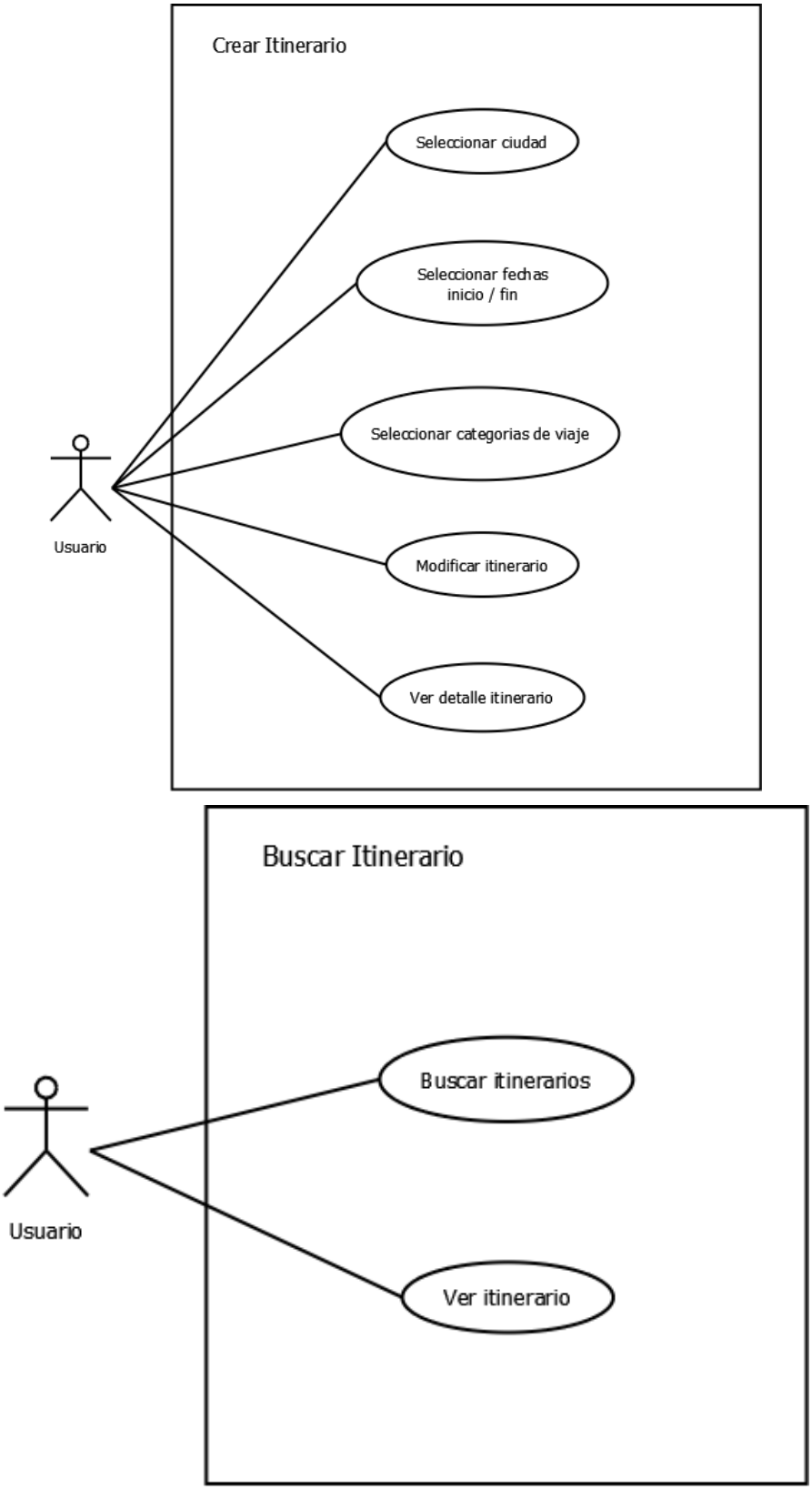
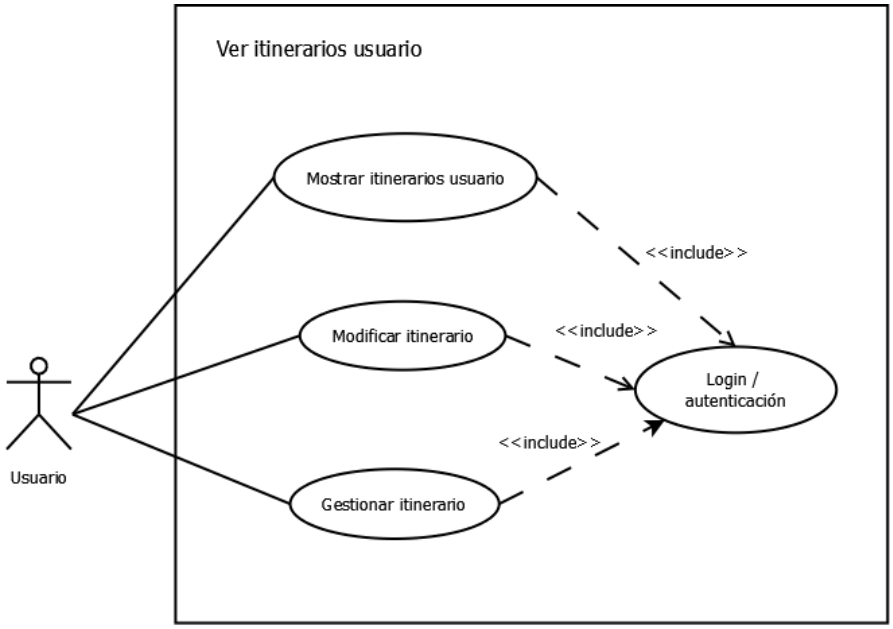
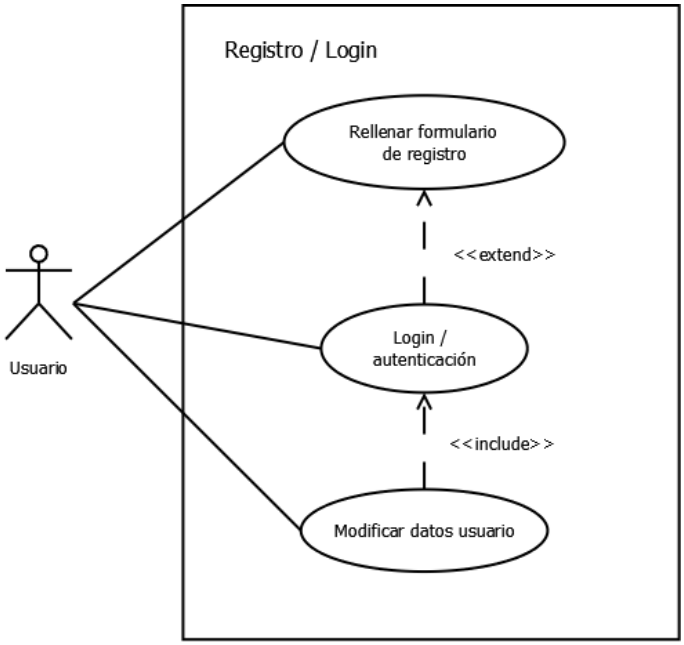


Figura 14. Caso de Uso Contactar


Crear Itinerario
Seleccionar ciudad
Seleccionar fechas
inicio / fin
Seleccionar categorias de viaje
Modificar itinerario
Ver detalle itinerario
Usuario
Buscar Itinerario
Buscar itinerarios
Ver itinerario
Usuario


Ver itinerarios usuario
Mostrar itinerarios usuario
<<include>>
Modificar itinerario
<<include>>
Login / autenticación
<<include>>
Gestionar itinerario
Usuario


Registro / Login
Rellenar formulario de registro
<<extend>>
Login / autenticación
<<include>>
Modificar datos usuario
Usuario

11.2 Especificación de los casos de uso

Tabla 6. Especificación CU001

CU001: Registro de usuario en la plataforma.
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Garantías en caso de éxito: el sistema guardará la información de registro del nuevo usuario y mostrará por pantalla la confirmación de éxito en la operación.
Escenario principal de éxito: 1. El usuario accede a la sección “Registrarse” a través de la barra de navegación. 2. El usuario rellena el formulario de registro. 3. El sistema valida la información introducida por el usuario. 4. El sistema habilita el botón de registro si la información es válida. 5. El usuario envía el formulario. 6. El sistema confirma la creación del nuevo usuario.
Extensiones: 3a. El sistema muestra campos del formulario con datos que no son válidos. 3a1. El usuario vuelve a introducir los datos según las indicaciones dadas por el sistema. 6a. El sistema informa que ya existe un usuario con el mismo usuario o con el mismo correo electrónico. 6a1. El usuario cambia el correo electrónico o el nombre de usuario empleado.

Tabla 7. Especificación CU002

CU002: Contactar con TravelFit.
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Garantías en caso de éxito: el sistema registrará la petición de contacto realizada por el usuario y mostrará por pantalla la confirmación de éxito en la operación.
Escenario principal de éxito: 1. El usuario accede a la sección “Contacto”. 2. El usuario rellena el formulario de contacto. 3. El sistema valida la información introducida por el usuario. 4. El sistema habilita el botón de envío si la información es válida. 5. El usuario envía la petición de contacto. 6. El sistema confirma la solicitud de contacto.
Extensiones: 3a. El sistema muestra campos del formulario con datos que no son válidos. 3a1. El usuario vuelve a introducir los datos según las indicaciones dadas por el sistema.

Tabla 8. Especificación CU003

CU003: Iniciar sesión en la aplicación
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Garantías en caso de éxito: el sistema inicia sesión en la aplicación.
Escenario principal de éxito: 1. El usuario pulsa en “Iniciar sesión” sobre la barra de navegación. 2. El usuario introduce sus credenciales. 3. El sistema valida la información introducida por el usuario. 4. El sistema habilita el botón para iniciar sesión si los datos son válidos. 5. El usuario inicia sesión. 6. El sistema adapta la interfaz con las opciones del usuario tras ingresar al sistema.
Extensiones: 3a. El sistema muestra los campos con datos que mal formateados. 3a1. El usuario vuelve a introducir los datos según las indicaciones dadas por el sistema. 6a. El sistema informa que las credenciales son incorrectas. 6a1. El usuario debe introducir unas credenciales correctas. 6a2. El usuario debe registrarse en el sistema para validar sus credenciales.

Tabla 9. Especificación CU004

CU004: Navegar por los itinerarios creados en el sistema
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Garantías en caso de éxito: el sistema muestra un listado de itinerarios creados por otros usuarios en el sistema, y el detalle de un itinerario al seleccionarlo.
Escenario principal de éxito: 1. El usuario accede a la sección de itinerarios a través del menú de navegación. 2. El sistema muestra el listado de todos los itinerarios registrados en el sistema 3. El usuario navega a través de los resultados mostrados. 4. El usuario hace clic sobre un ítem del listado. 5. El sistema muestra el detalle del itinerario.

Tabla 10. Especificación CU005

CU005: Crear un itinerario
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Garantías en caso de éxito: El sistema presenta un itinerario según los parámetros del usuario.
Escenario principal de éxito: 1. El usuario rellena el campo de destino y selecciona las fechas de inicio y fin del itinerario. 2. El sistema muestra un listado de categorías de puntos turísticos. 3. El usuario selecciona las categorías según sus preferencias y gustos. 4. El sistema crea un itinerario según los datos recopilados y lo muestra al usuario. 5. El usuario revisa el itinerario generado en la aplicación.
Extensiones: 5a. El usuario modifica el itinerario mostrado. 5a1. El usuario introduce cambios en el itinerario. 5a2. El sistema actualiza los datos del itinerario. 6a. El sistema registra el itinerario creado por el usuario. **Precondición**: el usuario debe estar registrado en la plataforma.

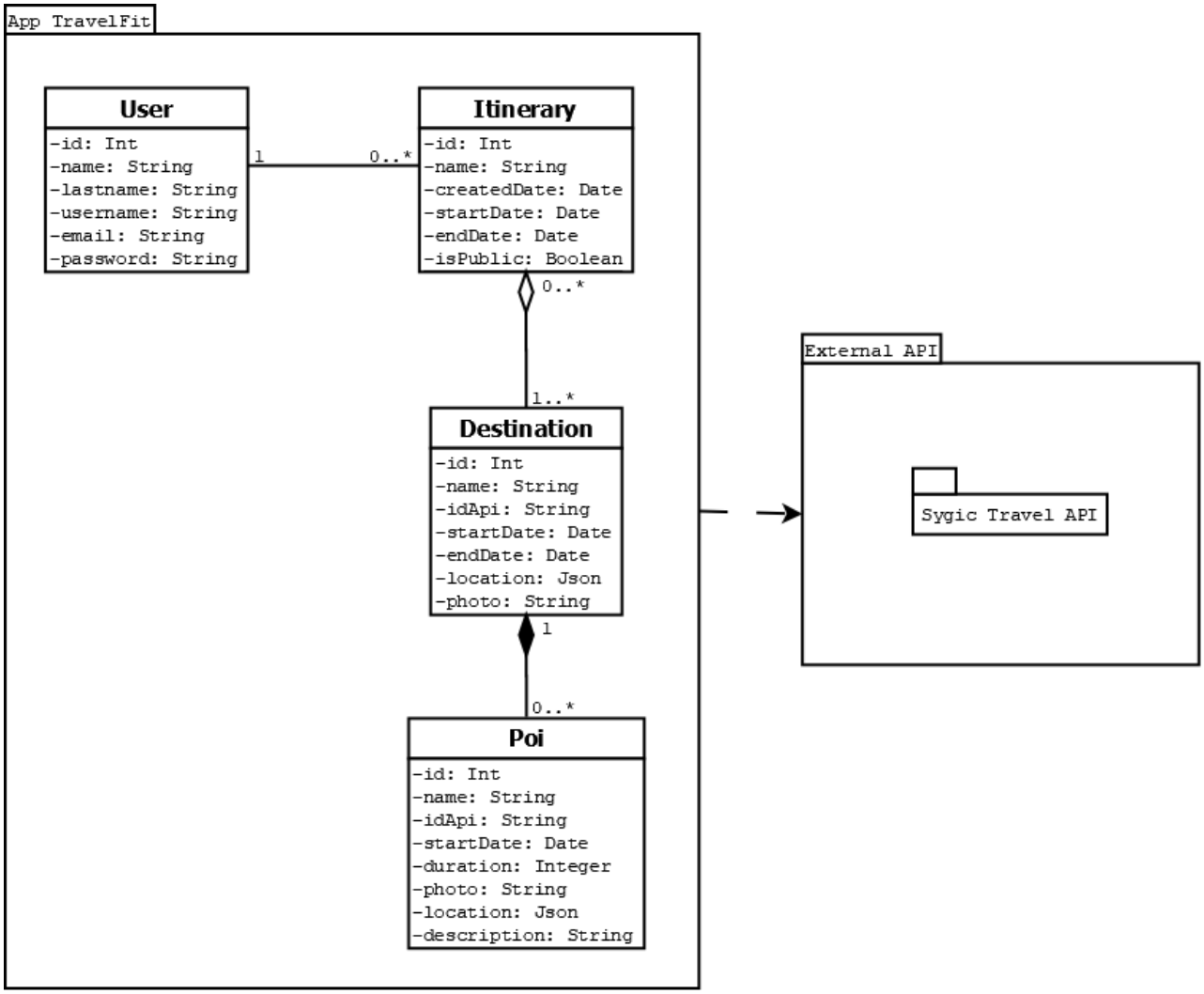
Tabla 11. Especificación CU006

CU006: Modificar los datos de usuario
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Precondición: El usuario debe esta registrado en el sistema y haber iniciado sesión.
Garantías en caso de éxito: el sistema registra los cambios efectuados.
Escenario principal de éxito: 1. El usuario accede al panel de usuario. 2. El usuario indica que quiere modificar los datos. 3. El sistema muestra los campos disponibles para modificar. 4. El usuario realizar las modificaciones oportunas. 5. El sistema valida los datos introducidos y graba los cambios realizados.
Extensiones: 5a. El sistema muestra al usuario que los datos introducidos nos válidos y no registra los cambios. 5a1. El usuario modifica los datos introducidos. 5a2. El sistema valida los datos introducidos y graba los cambios realizados.

Tabla 12. Especificación CU007

CU007: Mostrar los itinerarios propios del usuario actual.
Actor principal: Usuario
Ámbito: Aplicación TravelFit
Nivel de Objetivo: Usuario
Precondición: El usuario debe esta registrado en el sistema y haber iniciado sesión.
Garantías en caso de éxito: el sistema muestra los itinerarios disponibles.
Escenario principal de éxito: 1. El usuario accede a sus itinerarios a través del menú de navegación. 2. El usuario selecciona un itinerario del listado mostrado. 3. El sistema muestra los detalles del itinerario seleccionado. 4. El usuario realizar las modificaciones oportunas. 5. El sistema valida los datos introducidos y graba los cambios realizados.
Extensiones: 4a. El usuario elimina el itinerario. 4a1. El sistema muestra el listado de itinerarios de usuario. 4b1. El sistema informa que no ha sido posible eliminar el itinerario. 5a. El sistema muestra al usuario que los datos introducidos nos válidos y no registra los cambios. 5a1. El usuario modifica los datos introducidos. 5a2. El sistema valida los datos introducidos y graba los cambios realizados.

11.3 Diagrama de clases



12. Prototipos

12.1 Lo-Fi

Página inicio de la aplicación.

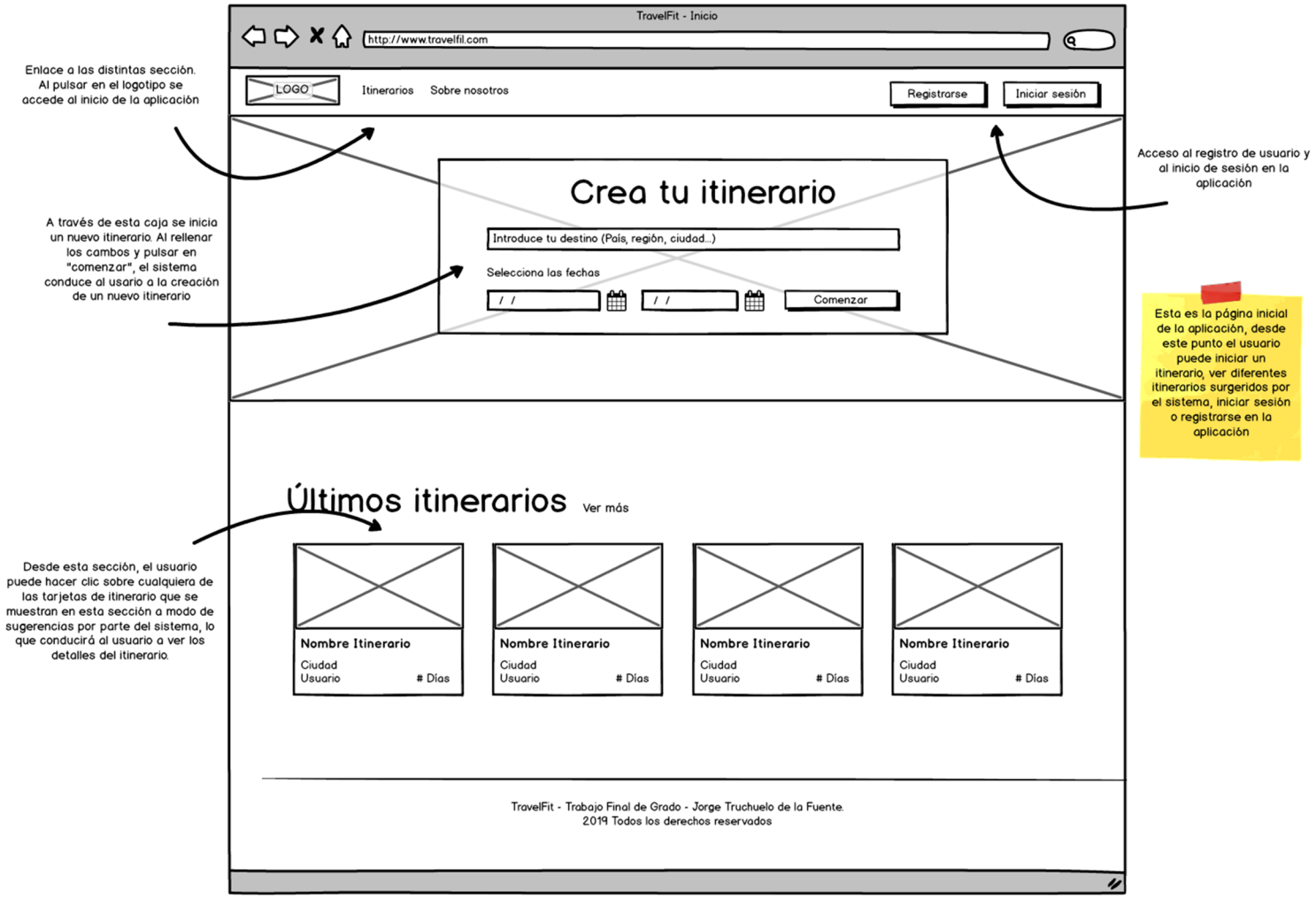
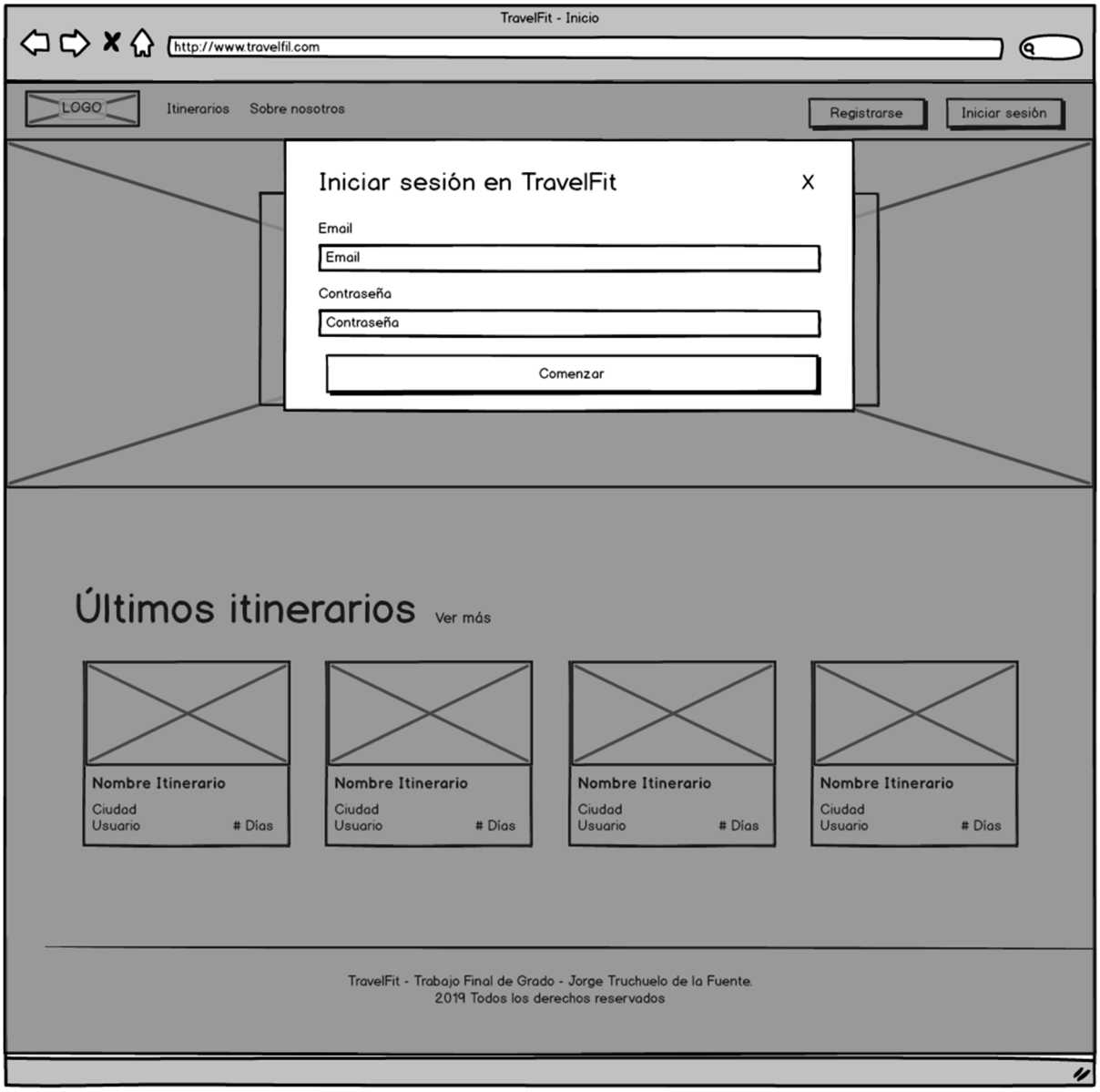


Figura 15. Página inicio de la aplicación

Página de iniciar sesión.



Muestra de la ventana de acceso de usuario a la aplicación

Figura 16. Página inicio de iniciar sesión

Página inicial con usuario registrado

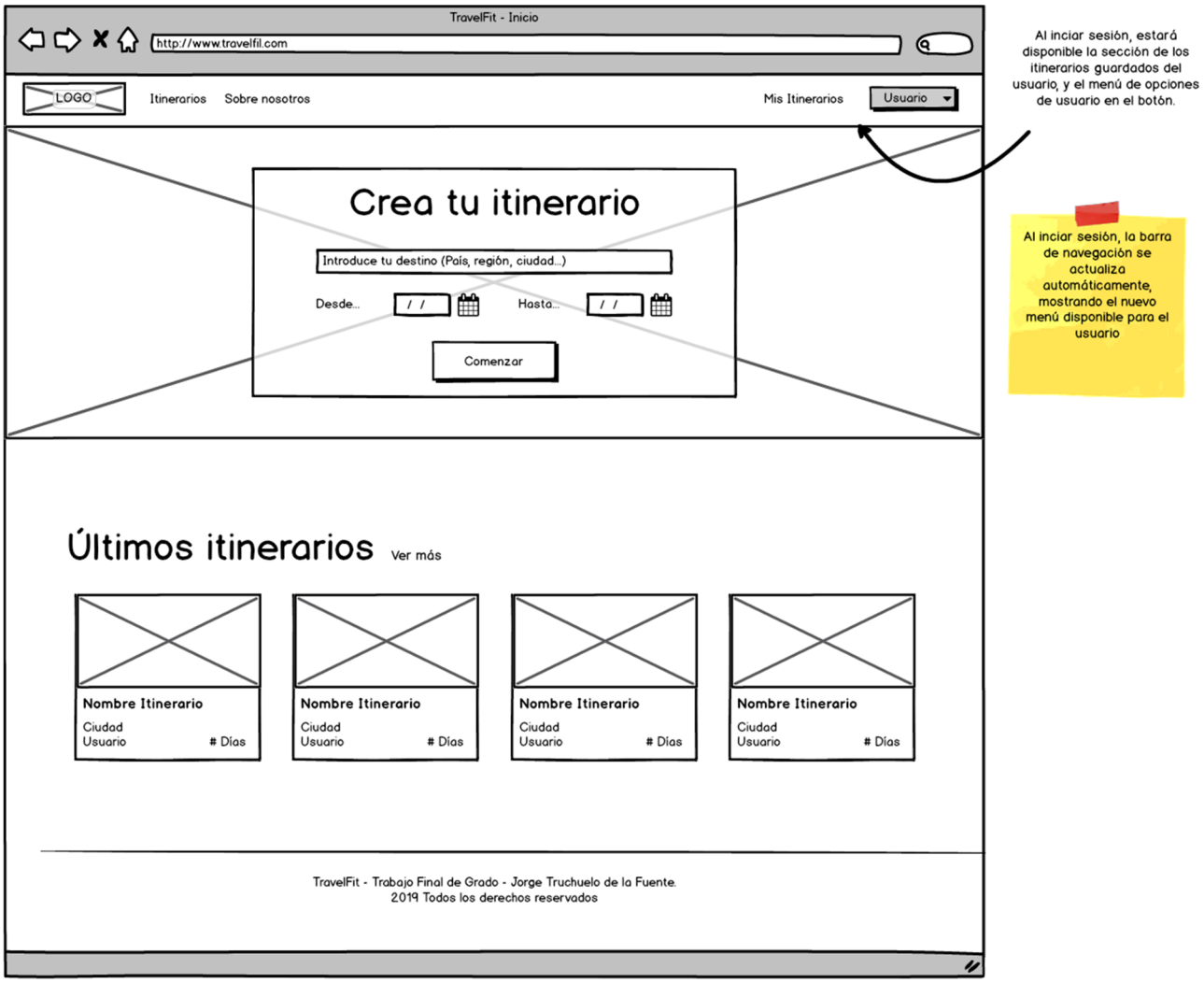


Figura 17. Página inicial con usuario registrado

Página de contacto

Figura 18. Página de contacto

Página de registro de usuario.

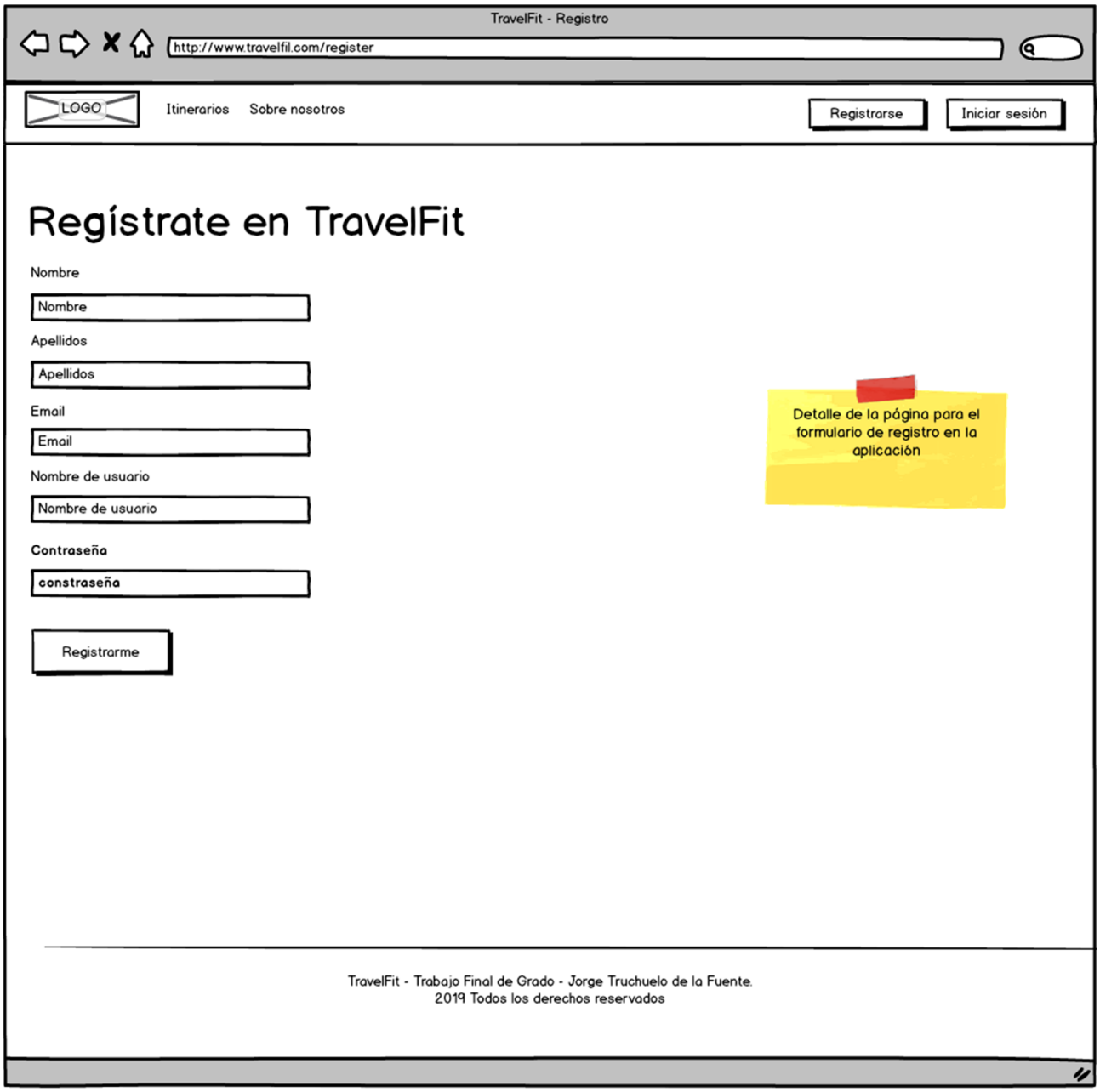


Figura 19. Página de registro de usuario.

Página Mis Itinerarios. Para mostrar los itinerarios almacenados de un usuario registrado en la aplicación

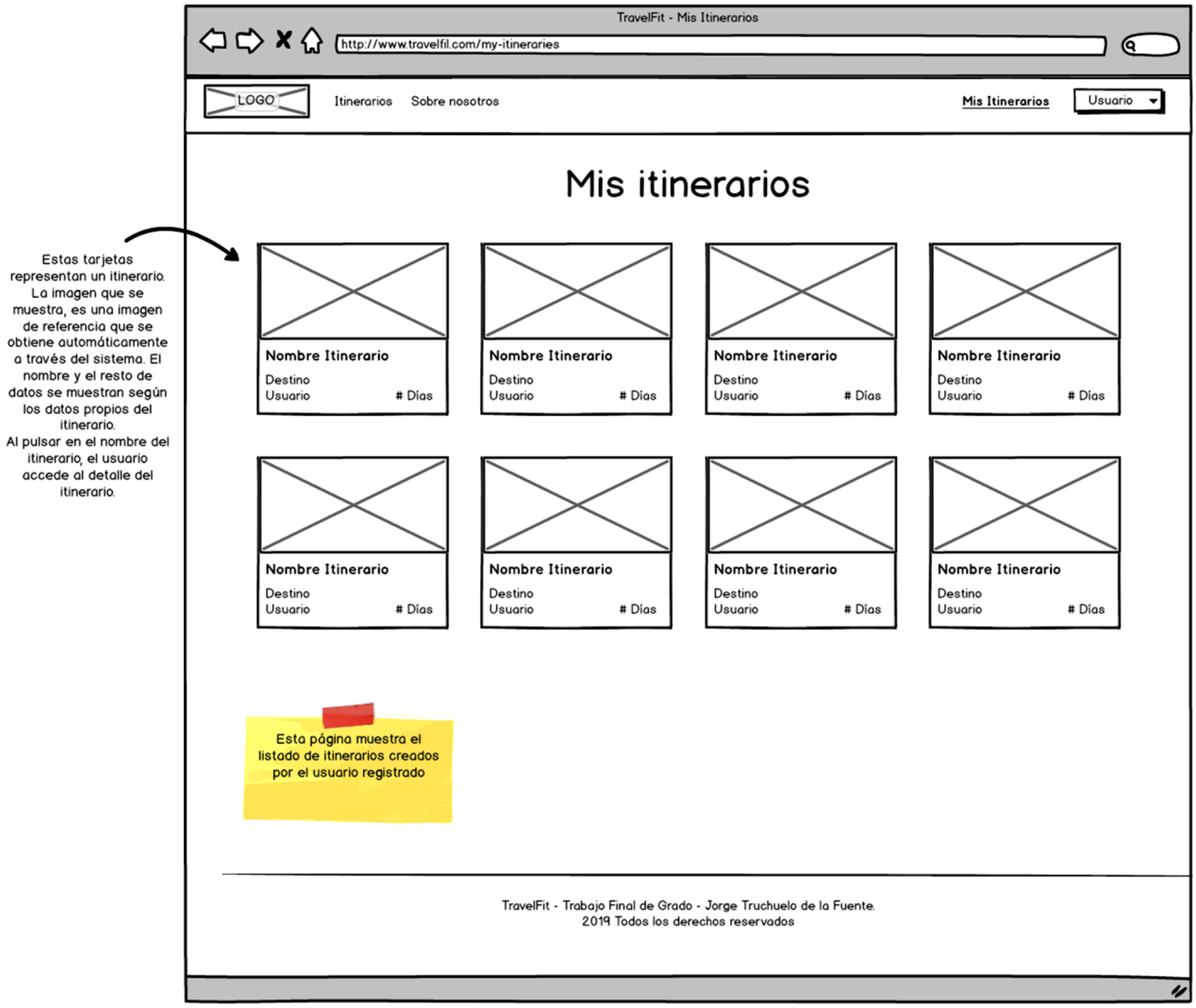


Figura 20. Página Mis Itinerarios

Página de itinerarios públicos: Para mostrar todos los itinerarios públicos registrados en el sistema.

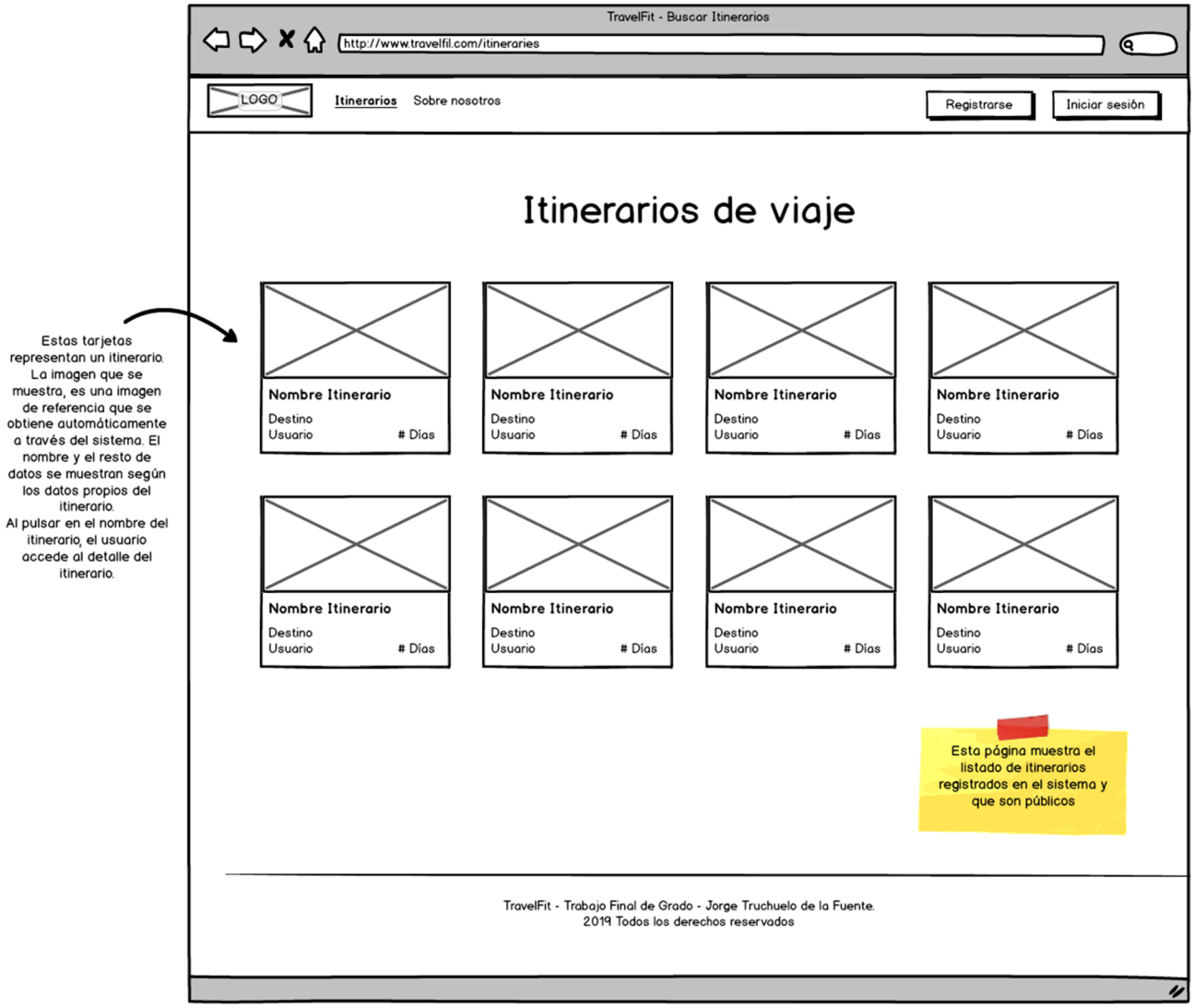


Figura 21. Página de itinerarios públicos

Página Itinerario – Resumen: Página que muestra el resumen de un itinerario.

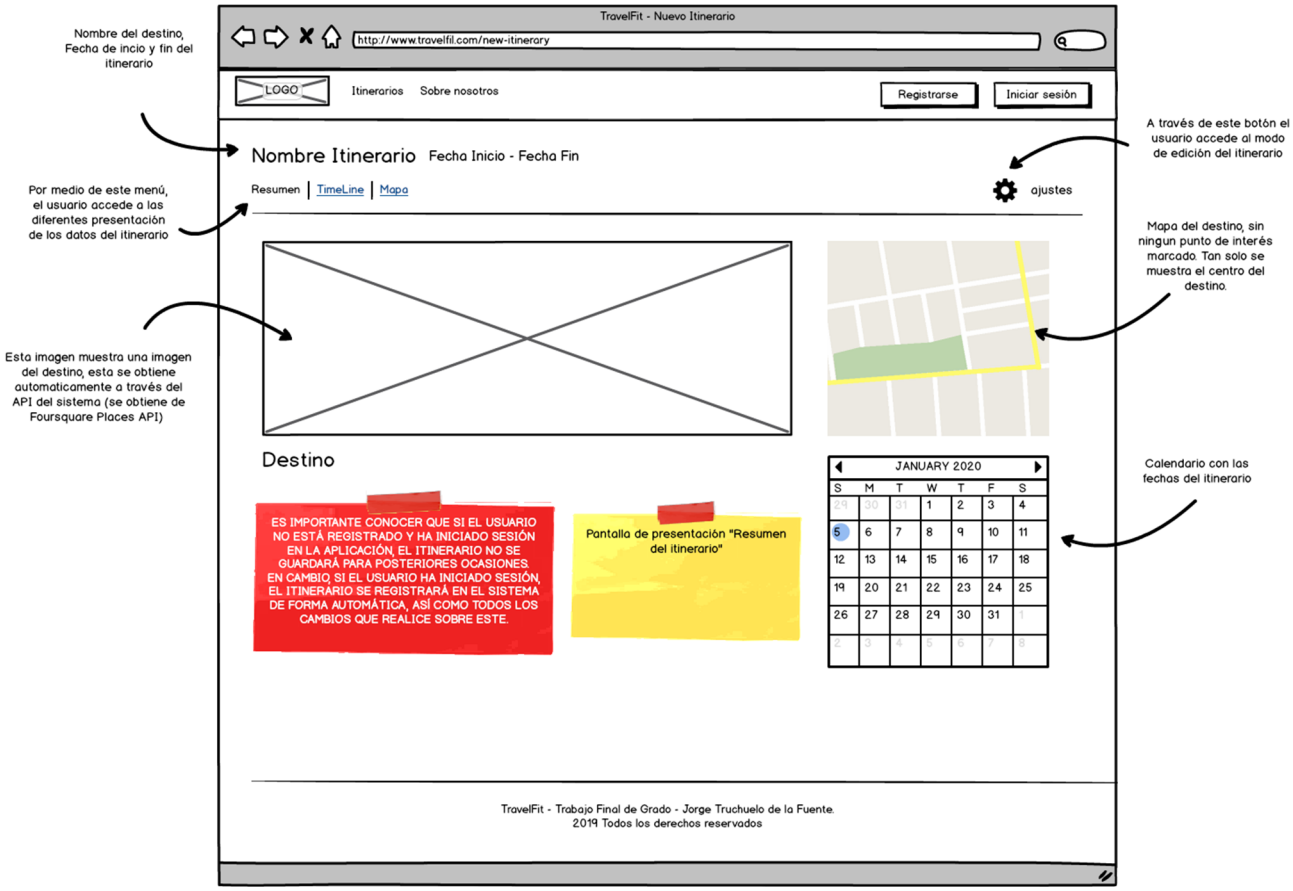


Figura 22. Página Itinerario - Resumen

Página Itinerario – Ajustes.

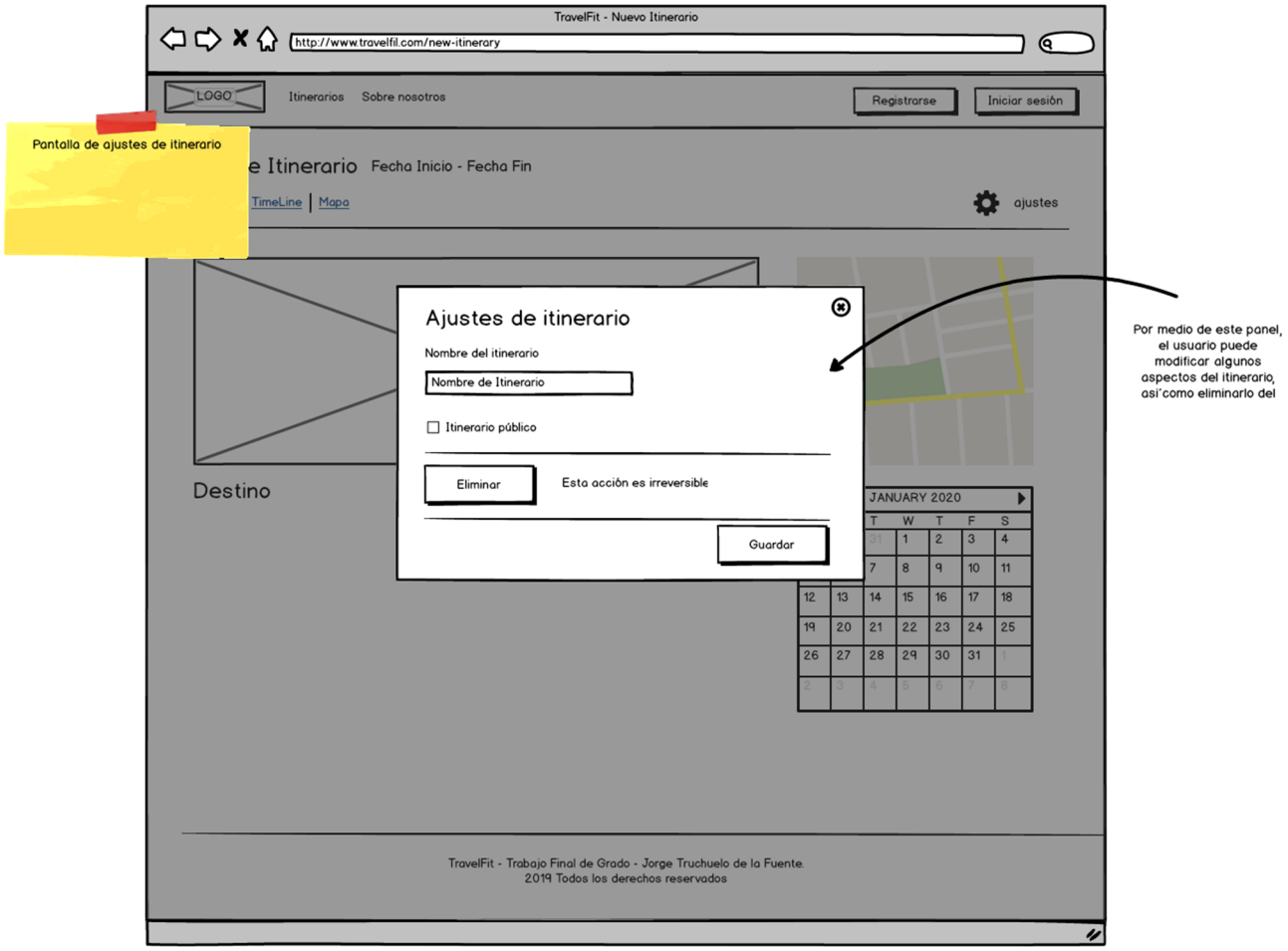


Figura 23. Página Itinerario - Ajustes

Página Itinerario – Mapa: Página que muestra el mapa de un itinerario.

Figura 24. Página Itinerario - Mapa

Página Itinerario – Timeline: Página que muestra el listado de puntos de interés de un itinerario.

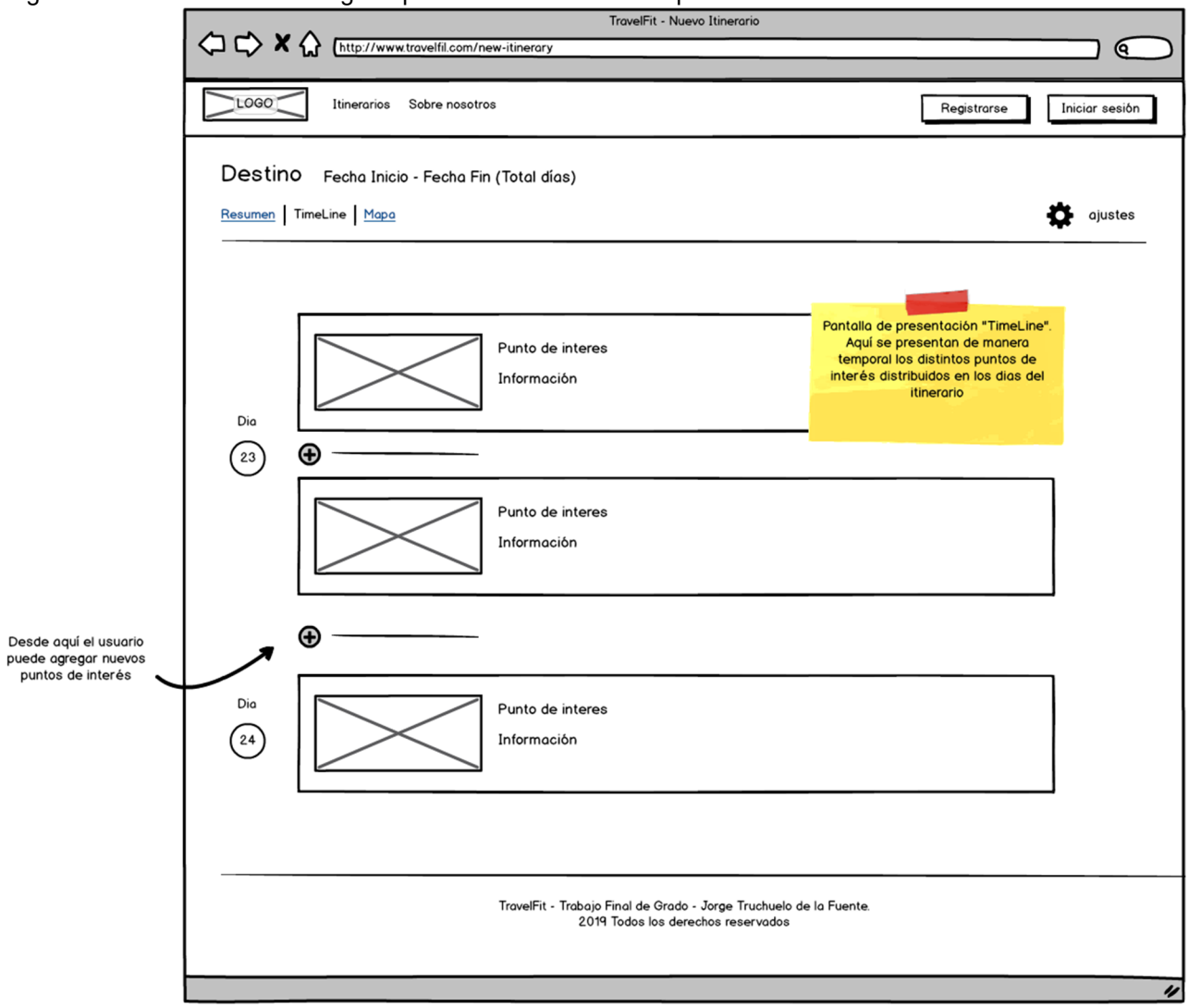


Figura 25. Página Itinerario - Timeline

Estados barra de navegación

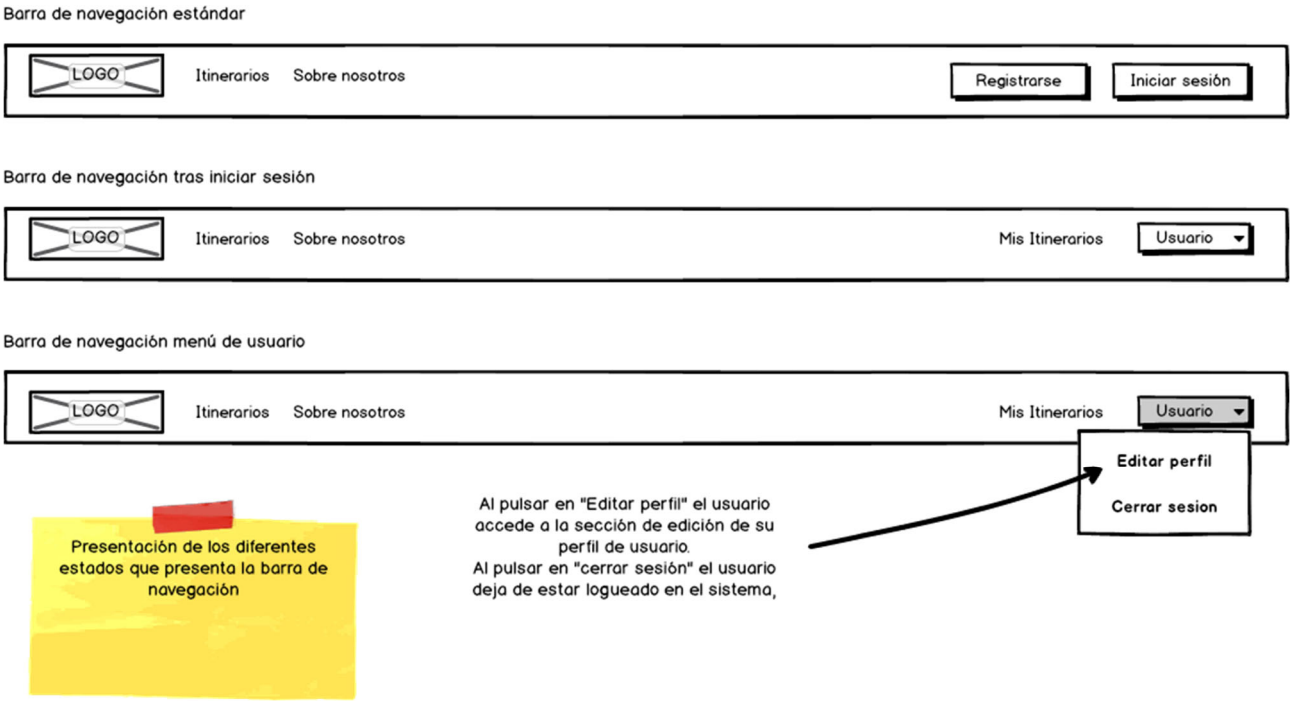


Figura 26. Estados barra de navegación

13. Usabilidad/UX

Para el adecuado desarrollo de la aplicación y de su interfaz gráfica, se ha aplicado una metodología de diseño centrado en el usuario, por sus siglas DCU, en la que se han definido los perfiles de usuario, se ha definido la intención comunicativa del producto, las necesidades generales del proyecto, así como una temática general para la aplicación. De este modo, se ha podido perfilar de una manera general a los distintos usuarios de la aplicación, en que contexto van a utilizarla y los contenidos que tendrá.

Así, hemos podido canalizar los conocimientos extraídos hacia el uso adecuado de distintos principios y técnicas de usabilidad y de experiencia de usuario a la interfaz gráfica e interacciones de la aplicación para obtener un resultado positivo, categorizando en cuatro principios: Comodidad, Ser intuitiva, Consistencia, Fiabilidad[8]:

Comodidad: Con la intención de crear una sensación de confort para el usuario se han tenido en cuenta estos factores:

- La aplicación se diseña para que el tanto el nuevo usuario como el que conoce la aplicación sea completamente autónomo.
- El usuario puede retornar a cualquier punto de manera sencilla. Es decir, podrá acceder a cualquier sección de la aplicación desde cualquier punto en el que se encuentre.
- La estructura de la aplicación es sencilla y simple, gracias a las pocas secciones que la componen, así se ha podido evitar el uso de mapas o índices de la aplicación.
- Interfaz explorable y con rutas rápidas a las distintas secciones.
- Se ha tenido presente la ley de Fitts, dando mayor protagonismo (tamaño y/o visibilidad) a los elementos de las funciones principales, generación de un itinerario o a la navegación por la colección itinerarios, realzando la utilidad de la aplicación.
- En lenguaje usado es claro y conciso.
- Se emplea técnicas de auto relleno en los campos de búsqueda de ciudad para facilitar el uso.
- La aplicación usa URLs amigables
- Enlaces generales y directos.

Ser intuitiva: Para conseguir que la interfaz gráfica y el uso de la aplicación sea intuitivo, se ha tenido presente los siguientes principios:

- Arquitectura de la información. Los contenidos han sido estructurados de una manera coherente y convenientemente estructurada, con especial énfasis en los elementos necesarios para crear un

[8] [1]***Goto, Kelly***.*Brand Value and the User Experience.Disponible en línea en: <http://www.digital-web.com/articles/brand_value_and_the_user_experience/>*

itinerario, empleando sistemas de clasificación ambiguos y exactos, la navegación es jerárquica de 2 niveles y lineal, el etiquetado de las opciones es coherente y existen una jerarquía visual de los elementos. También se ha empleado un sistema de búsqueda para que el usuario pueda encontrar un itinerario.

- Las opciones de uso son fáciles, interacciones sencillas y claramente indicadas, uso de *rollover* en todas aquellas opciones con las que pueda interactuar el usuario.
- La iconografía empleada, así como los elementos de interacción o los términos empleados son de convención, estándares y coherentes respecto a otras aplicaciones o páginas web, para que el usuario tenga una curva de aprendizaje mínima.
- El punto anterior se traduce en existen pocos elementos a memorizar gracias al uso de convenciones y formas de navegación que el usuario ya ha experimentado en otras aplicaciones.

Consistencia: Para que la navegación por las diferentes secciones de la aplicación resulte coherente:

- La identidad de la aplicación es constante, al igual que el comportamiento y los elementos de interacción, existiendo continuidad entre las diferentes secciones de esta. Esto se consigue aplicando una consistencia en la estructura de las distintas páginas, así como en los elementos que la forman (retícula, gama de colores, tipografía e iconografía) y en las acciones de interacción o en la terminología aplicada.

Fiabilidad: Para realzar la sensación de seguridad y fiabilidad para el usuario:

- Se pone a disposición del usuario una sección de contacto con la empresa para poder transmitir sus opiniones.
- Se emplean imágenes con calidad, contenido bien formateado.
- El usuario está informado en todo momento del estado del sistema, para recibir una respuesta adecuada al utilizar la herramienta. Con la visibilidad del estado, el usuario podrá recibir información sobre los procesos que se estén llevando a cabo, y así tener *feedback* sobre sus acciones.
- Se tienen en cuenta los tiempos de respuesta de la aplicación, para que no sean excesivamente largos.

Además de los principios y factores mencionados también se han tenido en consideración los siguientes puntos:

- La aplicación es de tipo SPA para que la experiencia de uso sea fluida.
- Control y libertad del usuario: Para que el usuario sienta que controla la aplicación se otorga la capacidad de modificar cualquier parámetro de un itinerario y capacidad para modificar sus datos personales.

- Diseño estético y minimalista: Con el principio básico de usabilidad KISS por bandera, se ha diseñado una aplicación limpia, sin recarga, sin información ni datos que no sean de interés del usuario o no estén relacionados con la aplicación.
- Se emplean técnicas de adaptabilidad para que la aplicación se vea correctamente en diferentes tamaños de pantalla.
- La tipografía escogida es clara, tipografía lineal *sans-serif*, y va de la mano con el contenido de la web, el color y el tamaño están pensados para facilitar la lectura y mantener la armonía con la aplicación
- La selección de la paleta de color se adecua a las emociones que se quiere transmitir con la aplicación.

Por último, hay que mencionar que para cumplir con la metodología de diseño centrado en el usuario hay que cumplir con la fase de evaluación/testeo y poder medir el uso de la aplicación. En el trascurso de este TFG, esta fase se experimenta mientras se desarrollan las distintas etapas de diseño y desarrollo de la interfaz de usuario, de esta manera, se pueden resolver aquellos problemas que puedan surgir del prototipado inicial. Por ello responder a las preguntas cómo ¿es fácil navegar por la aplicación?, ¿el contenido está organizado coherentemente?, ¿el menú de navegación es sencillo e intuitivo? o ¿estoy cómodo al usar la aplicación?, durante todo el desarrollo de la aplicación es de vital importancia para lograr una usabilidad y experiencia correcta para la aplicación.

14. Seguridad

La seguridad del sistema se basa en las siguientes pautas:

Autenticación de usuarios por JWT (*JSON Web Token*): La autenticación del usuario se realiza mediante el uso del estándar abierto JWT, donde un usuario hace una petición de login, con su email y contraseña, al sistema y éste, tras comprobar que las credenciales sean correctas, devuelve un token que será empleado en el resto de las comunicaciones cliente->servidor a través de las cabeceras protegidas de las peticiones realizadas hacia el API.

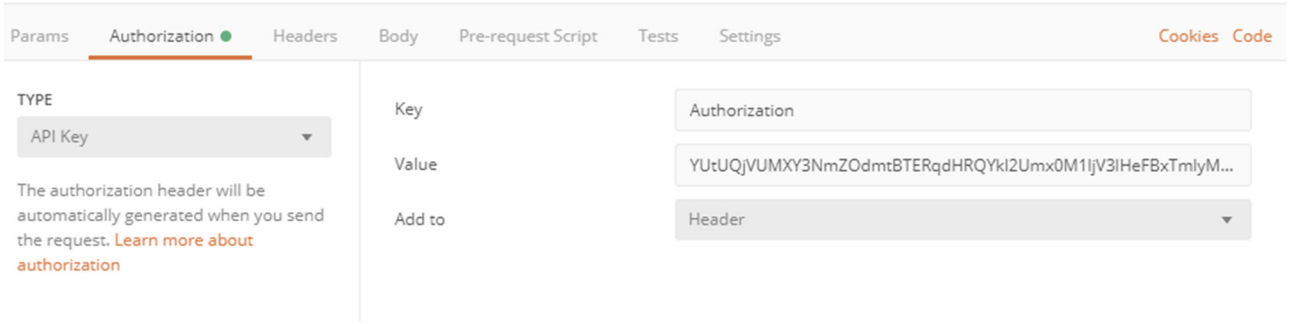


Figura 27. Ejemplo de configuración de cabecera API Key en la aplicación Postman.

El token se genera cada vez que el usuario accede a la plataforma y se elimina cuando este hace logout. Para generarlo se hace uso de una función hash Bcrypt.

Algunas de las llamadas al API del backend están protegidas y requieren que el usuario se haya autenticado en la plataforma, y de este modo poder acceder a ciertas funcionalidades como puede ser la modificación de un itinerario creado, en cuyo caso solo podría editar el usuario propietario. Para ello se han empleado middleware en las rutas que ofrece el API para comprobar la existencia de un token válido en las cabeceras de las peticiones.

Las contraseñas de los usuarios no se guardan en el sistema, en su lugar se guarda el hash resultante de aplicar una función Bcrypt.

Además, en el frontend se aplican medidas para proteger las rutas que deben accedidas por usuario registrados, para ello se emplea el uso de los Guards que proporciona el framework. Además, se emplean técnicas para ocultar o mostrar aquellas secciones de una página que requieran iniciar sesión en la aplicación.

En cuanto a la validación de los datos introducidos, principalmente las validaciones se realizan en dos puntos. En primer lugar, los campos de introducción de datos que aparecen en diversos lugares de la

aplicación cuentan con las correspondientes validaciones por parte del navegador y de las utilidades que proporciona HTML5.

En segundo lugar, encontramos las validaciones que se producen dentro del backend. Por un lado, la aplicación valida todos los campos que se reciben según una serie de reglas definidas, esto se realiza gracias a que Laravel implementa en todos sus controladores métodos para adecuada validación de las solicitudes HTTP entrantes con una poderosa variedad de reglas. Por otro lado, Laravel también emplea un ORM (*Object-Relational mapping*) llamado Eloquent, para la relación entre los datos del sistema y la base de datos. Este ORM incorpora en su núcleo medidas de protección frente ataques de tipo *SQL injection*, de modo que de manera nativa, Laravel se asegura que ningún usuario pueda realizar este tipo de ataques.

15. Requisitos implantación

15.1 Requisitos

Los siguientes requisitos son necesarios para el correcto funcionamiento de la aplicación, por ello es necesario que estén previamente instalados en el sistema y configurados correctamente:

- Servidor Apache versión >= 2.4.39. Debe tener activado el módulo rewrite_module, así extensiones de PHP requeridas por Laravel[9].
- Servidor MySQL 5.7.26 o como alternativa el SGBD MariaDB.
- PHP 7.2
- Node.js v12.*.*
- Angular/CLI (instalación global.

15.2 Preparación

Paso 1. Instalación de la aplicación backend. Debemos situar el contenido de la carpeta travelfit-backend en la ruta donde se vaya a situar la aplicación, por ejemplo: `/htdocs/www/travelfit-backend`, Esta ubicación depende de la instalación de nuestro servidor, e incluso del sistema operativa que lo gobierna. Si la finalidad es subirla a un servidor remoto, primero deberemos configurar la aplicación, como se detalla más adelante en el paso 3, para asegurar su correcto funcionamiento.

Paso 2. Instalación de la base de datos. A continuación, debemos crear una base de datos que empleará la aplicación. Esto lo podemos realizar, por ejemplo, a través de phpMyAdmin, si es que se dispone de ello en el servidor. Es importante que la base de datos que se cree tenga un cotejamiento de tipo **utf8mb4_unicode_ci** para que no exista ningún problema con la aplicación y la codificación que emplea.

Paso 3. Configuración variable entorno backend. Debemos configurar ciertos parámetros para el correcto funcionamiento del backend, estos parámetros se configuran a través del fichero `.env` que se ubica la raíz de la aplicación. Las directivas que debemos modificar son aquellas relacionas con la conexión a la BBDD:

- DB_HOST=127.0.0.1 – Dirección IP (si es que la BBDD se encuentra en otra ubicación)
- DB_PORT=3306 – Puerto (si es que varía del valor por defecto)
- DB_DATABASE=travelfit – Nombre de la BBDD creada en el servidor.
- DB_USERNAME=root – Nombre de usuario de la base de datos.
- DB_PASSWORD= - Constraseña del usuario de la BBDD.

[9] https://laravel.com/docs/6.x/installation

También podría ser necesario modificar otros aspectos como:

- APP_NAME=Travelfit – Nombre de la áplicación
- APP_ENV=local – Entorno de la aplicación (“production” si el entorno es de producción)
- APP_DEBUG=true – Indicaremos “false” si no queremos que muestre los errores que se producen, útil en un entorno de producción.
- APP_URL=http://travelfit.com.devel – La URL a la que atiende el servidor donde se aloja la aplicación.

Cabe mencionar que además debemos modificar las variables de entorno relacionadas con Sygic Travel API, como es el TOKEN o la URL del API.

- SYGIC_API_URL=https://api.sygictravelapi.com/1.2/es/
- SYGIC_API_KEY=(Por seguridad no se ha incorporado el token 2en este documento)

Tras realizar estas modificaciones es aconsejable ejecutar los siguientes comandos dentro de la ruta de la aplicación:

```
php artisan config:clear
php artisan cache:clear
```

Por último y por seguridad, se recomienda generar una nueva llave para la aplicación:

```
php artisan key:generate
```

Paso 4. Creación de tablas de la base de datos. Para la creación de las tablas podemos emplear dos métodos según sea el escenario.

1) Por un lado, podemos ejecutar las migraciones creadas en Laravel para que, de manera automática, cree las tablas necesarias. Esto lo podemos realizar lanzando el comando que se muestra a continuación desde la ruta raíz de la aplicación.

   ```
   php artisan migrate
   ```

2) El siguiente método se trata de emplear un fichero denominado **travelfit_utf8mb4_unicode_ci.sql** adjunto en la carpeta del proyecto. Se trata de un volcado completo de la base de datos con cierto contenido ya almacenado para no comenzar desde cero. Este fichero se puede importar a la BBDD desde, por ejemplo, phpMyAdmin y se crearán todas las tablas necesarias además de incorporar datos a las mismas.

Paso 5. Configuración CORS. Si el frontend y el backend se situan en servidores con diferente URL, como es el caso, es conveniente habilitar ciertas directivas para evitar la protección CORS. Para ello, hay varios métodos en función de si tenemos un VirtualHost habilitado por ejemplo. Una buena solución puede ser crear un fichero .htaccess en la carpeta raíz de la aplicación, con el siguiente contenido.

```
Header Set Access-Control-Allow-Origin "*"
Header set Access-Control-Allow-Methods "POST, GET, OPTIONS, DELETE, PUT"
Header set Access-Control-Allow-Headers "x-requested-with, Content-Type,
origin, authorization, accept, client-security-token"
```

Otra posibilidad puede ser habilitar en el ficher httpd.conf de Apache el módulo headers_module, y agregar al mismo la siguiente configuración:

```
<IfModule mod_headers.c>
  Header set Access-Control-Allow-Origin "*"
</IfModule>
```

Paso 6. Configuración URL backend en la aplicación frontend. Debemos configurar la URL del backend dentro de la configuración de la aplicación Angular, para ello modificáremos el fichero global.ts situado en `/travelfit-front/src/app/services/` y cambiaremos la ruta. Hay que tener en cuenta que por defecto Laravel apunta a su carpeta `public/`, y además deberemos agregar `api/` a la ruta, la ruta final podría ser algo similar a esto: `http://localhost/travelfit/public/api/`.

15.3 Ejecución

Al levanter el servidor Apache y el servidor MySQL, ya estará disponible la parte del backend, por lo que no haría falta ningún paso adicional.

Para ejecutar la aplicación frontend, dentro de un entorno local, podemos ejecutar el servidor de Angular a través del commando `ng serve` en la carpeta raíz de la aplicación, travelfit-frontend.
Si queremos poner la aplicación en producción, bastará con crear una compilación de la aplicación a través del commando `ng build --base-href /myUrl/`, donde `/myUrl/` es la ruta de directorio donde se vaya a situar el contenido de la carpeta `/dist`. Por último, copiar el contenido de la carpeta `/dist` a la ubicación que hayamos elegido.

Al accede mediante el navegador, tendremos nuestra aplicación corriendo de manera correcta.

16. Bugs

A lo largo del desarrollo de la aplicación se ha ido realizando múltiples pruebas y verificando que no se produjera ningún error.

1) Uno de los bugs que se ha detectado y no se ha podido solucionar, es el uso de teclas sobre el listado de posibles ciudades a la hora de buscar un destino al comenzar un nuevo itinerario. Al ir introduciendo el nombre de una ciudad destino, aparece un listado de ítems sobre los que se puede seleccionar nuestro destino. Si atacamos con el ratón, no hay problema, pero al desplazarnos entre los ítems por medio de los cursores, en algunos momentos, no se selecciona el ítem correspondiente.
   Esto es debido a que el API Maptiler, no solo lista ciudades, también lista cualquier ubicación que pueda coincidir con el nombre, y al aplicar un filtro para dejar solo aquellos que corresponden con ciudades, este no hace que se pueda saltar correctamente de un ítem a otro mediante teclas.

2) Los mapas de OpenStreetMaps en ocasiones tardan mucho en cargar. No se trata especialmente de un bug, pero en ocasiones se ha dado el caso que no han llegado a cargar las capas del mapa correspondiente, debido al tiempo excesivo de carga. Se entiende que, al ser un servicio gratuito, no siempre atiende las peticiones con la velocidad adecuada.

3) La aplicación tenía implementada funcionalidades para guardar una caché de todas las llamadas lanzadas contra el API externa, pero su funcionamiento ha estado dando problemas al cachear de manera errónea las llamadas y dando resultados erróneos, lo que ha conducido a la decisión de anular dicha funcionalidad en el sistema.

Además de los fallos reportados anteriormente, hasta ahora no se han dado más, pues siempre se ha tenido cuidado de ir depurando aquellos errores que han ido surgiendo durante el desarrollo y el testeo de la aplicación.

17. Proyección a futuro

El resultado de crear TravelFit es una aplicación muy sencilla y básica en comparación al resto de propuestas como las citadas en el escenario de este TFG, lo que me hace manifestar de forma segura que esta aplicación tiene muchas áreas de mejora, como pueden ser las siguiente:

De primera mano, muchas funcionalidades relacionadas con la modificación y gestión de los itinerarios han quedado fuera del alcance del proyecto pues, a pesar de haber sido planteadas en un principio, se han tenido que descartar por diversos motivos. Estas mejoras son:

- Implementar funcionalidades para la gestión de los puntos de interés dentro de un itinerario tales como: modificación de los parámetros o eliminación de un punto de interés, así como capacidad para agregar nuevos elementos al itinerario.
- Implementar funcionalidades para compartir los itinerarios entre los usuarios registrados en la aplicación.

Como posibles mejoras que se pueden incluir en la aplicación en futuras fases mencionamos las siguientes:

- Implementar funcionalidades para que un itinerario pueda tener múltiples destinos y no solo uno.
- Incluir la oferta de tour provenientes suministrados por Sygic Travel API.
- Agregar funcionalidades de reserva de ticket sobre aquellos puntos de interés que lo requieran.
- Capacidad para crear rutas entre los distintos puntos de interés de un mismo día.
- Agregar funcionalidades de reserva de alojamiento en aquellos destinos que se visiten.
- Sistema de autenticación OAuth para poder iniciar sesión a través de nuestras cuentas de Google o Facebook.
- Desarrollo de una aplicación nativa para dispositivos Android e iOS.
- Mejora en el diseño de interfaz de usuario.

Como parte positiva, hay que indicar que el hecho de haber empleado tecnologías actuales como Angular, Bootstrap o Laravel, así como emplear un sistema desacoplado entre cliente y servidor, otorgan a la aplicación facilidad para la ampliación, mejora y modificación que requieran las mejoras propuestas.

Varias de las propuestas futuras serían muy sencillas de integrar pues el sistema fue diseñado en un principio para cumplir con esas cualidades, lo que ayuda en gran parte de su implementación sin la necesidad de grandes modificaciones, e incluso sin necesidad de ellas en la base del backend.

Por todo ello, TravelFit puede ser una aplicación con muchas posibilidades de proyección, capaces de medirse cara a cara frente a otras soluciones actuales del mercado.

18. Conclusiones

La realización de este proyecto ha sido un camino duro, propiciado por los siguientes problemas:

- Desconocimiento del alcance real de la idea principal de la aplicación y de sus funcionalidades adheridas, lo que ha traído múltiples problemas en la planificación tanto temporal como del alcance del proyecto.
- Emplear tecnologías desconocidas que son la base del desarrollo de la aplicación, los framework Angular y Laravel, requiriendo mucho tiempo y dedicación para su dominio, más del previsto en un inicio.
- Búsqueda de API's externas para la obtención de datos de lugares.
- Inexperiencia en el desarrollo de una aplicación web real.

A nivel personal, en el trascurso de este TFG, he podido obtener una idea real de la complejidad que tiene otras aplicaciones similares a TravelFit como *Inspirock* o *TripHobo*, lo que me hace valorar el trabajo que puede haber detrás de cada una de ellas, aunque también creo que la carencia de experiencia y el haberme propuesto el empleo de tecnologías desconocidas ha jugado un papel crucial en la dificultad, puesto que ahora, en conocimiento de estas tecnologías, se haría cosas de manera diferente y las haría de una manera más rápida y eficaz.

Sin duda, la idea original de la aplicación era mucho más ambiciosa y completa, una idea que habría dado lugar a una aplicación con capacidades de gestión mejoradas sobre los itinerarios y los puntos de interés, una aplicación más llamativa que ofrecer y que pudiera competir con las distintas alternativas del mercado.

A nivel técnico, llego a la conclusión de haber acertado con las tecnologías escogidas. Actualmente la oferta es muy amplia, tanto en tecnologías para el desarrollo de backend como de fronted, lo cual fue motivo de duda al comienzo de este TFG, pero creo que la elección de Laravel ha sido muy acertada, pues he aprendido un potente framework con muchas posibilidades, en continuo desarrollo y una gran comunidad detrás.

De igual modo el framework de desarrollo Angular, aunque cabe decir que su curva de aprendizaje me ha resultado más compleja y creo que, si hubiera utilizado el sistema de vistas de Laravel o Vue.js, podría haber ahorrado muchos problemas en el desarrollo, pero de igual modo, se trata de una tecnología de gran potencia con mucho futuro por delante.

En cuanto a las tareas relacionadas con la gestión y planificación del proyecto, he vivido en persona lo difícil que es acertar con los tiempos de planificación que se plantean al comienzo de un proyecto y, de igual modo, darme cuenta de la realidad a la hora de plantear el alcance a obtener en el proyecto, la planificación y las diversas desviación y complicaciones que van surgiendo durante el recorrido, lo que me hace tomar

conciencia de las complejas labores y la coordinación que debe haber en un equipo de desarrollo para obtener los objetivos marcados en los plazos fijados.

Por último, el viaje realizado a través de presente TFG, así como los conocimientos adquiridos, han resultado una experiencia de gran valor, un aprendizaje que de otro modo no podría haber adquirido y no lo podría haber valorado igual, aunque no deja de tener un sabor agridulce al no haber alcanzado con efectividad todos los objetivos planteados con este proyecto.

Anexo 1. Entregables del proyecto

Lista de archivos entregados y su descripción.
El proyecto se ha entregado mediante un único fichero ZIP cuyo contenido está organizado en tres carpetas. La distribución de las carpetas, así como de los archivos que contienen es la siguiente:

Carpeta Proyecto:
Archivo: **PEC_FINAL_prj_Truchuelo_de_la_Fuente_Jorge.zip**

Este archivo contiene dos carpetas:
travelfit-backend - Contiene la totalidad del proyecto destinado al backend.
travelfit-frontend - Contiene todos los ficheros de desarrollo del frontend.

Además, incluye un archivo con el volcado completo de la base de datos, con itinerarios ya creados, así como usuarios, para no tener que comenzar de cero en su uso:
travelfit_utf8mb4_unicode_ci.sql – Archivo volcado BBDD.

Carpeta Memoria:
Archivo: **PEC_FINAL_mem_Truchuelo_de_la_Fuente_Jorge.zip**

Este archivo contiene la memoria del proyecto.
PEC_FINAL_mem_Truchuelo_de_la_Fuente_Jorge.pdf – Documento de la memoria.

Carpeta Presentación:
Archivo: **PEC_FINAL_vid_Truchuelo_de_la_Fuente_Jorge.mp4** – Video de la presentación-defensa del proyecto.

PEC_FINAL_prs_Apellidos_Nombre.pdf – Documento de presentación comercial del proyecto.

Anexo 2. Código fuente (extractos)

A continuación, se muestra un ejemplo de un controlador de ruta creado en Laravel, en concreto se trata del controlador encargado de recibir una petición de nuevo itinerario, y devolver en formato JSON los datos del itinerario creado.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Helpers\SygicAPI;
use DateTime;
use DateInterval;
use stdClass;

class SygicAPIController extends Controller
{
    protected $sygicApi;

    public function __construct(SygicAPI $sygicApi)
    {
        $this->sygicApi = $sygicApi;
    }

    public function new(Request $request)
    {
        // Obtengo datos recibidos front
        $itinerary = $request->new_itinerary;
        $destination = $itinerary['destinations'][0];
        $categories = $request->categories;

        // Obtengo datos del destino a traves del API Sygic
        $dataDestination = $this->sygicApi->getDestinationData($destination['name']);

        // Actualizo los datos del destino e itinerario
        $destination['id'] = 1;
        $destination['name'] = $dataDestination->data->places[0]->name_translated != ""
? $dataDestination->data->places[0]->name_translated : $dataDestination->data-
>places[0]->name;
        $destination['location'] = $dataDestination->data->places[0]->location;
        $destination['idApi'] = $dataDestination->data->places[0]->id;
        $destination['startDate'] = date('Y-m-d h:i:s',
strtotime($destination['startDate']));
```

```
        $destination['endDate'] = date('Y-m-d h:i:s',
strtotime($destination['endDate']));
        $destination['photo'] = $this->sygicApi->getDetails($destination['idApi']);
        $destination['itinerary_id'] = $itinerary['id'];
        $itinerary['createdDate'] = date('Y-m-d h:i:s',
strtotime($itinerary['createdDate']));
        $itinerary['startDate'] = $destination['startDate'];
        $itinerary['endDate'] = $destination['endDate'];

        // Comienzo a buscar los POI del destino
        // Cargo las categorías que hayan sido seleccionadas
        $searchCategories = [];
        if ($categories['discovering'] == true) $searchCategories[] = 'discovering';
        // if ($categories['eating'] == true) $searchCategories[] = 'eating';
        // if ($categories['going_out'] == true) $searchCategories[] = 'going_out';
        if ($categories['hiking'] == true) $searchCategories[] = 'hiking';
        if ($categories['playing'] == true) $searchCategories[] = 'playing';
        if ($categories['shopping'] == true) $searchCategories[] = 'shopping';
        if ($categories['sightseeing'] == true) $searchCategories[] = 'sightseeing';
        if ($categories['doing_sports'] == true) $searchCategories[] = 'doing_sports';
        // var_dump(implode('|', $searchCategories));die();

        // Calculo el número de días para saber cuantos días debe rellenar.
        $startDate = new DateTime($destination['startDate']);
        $endDate = new DateTime($destination['endDate']);
        $numDays = $endDate->diff($startDate)->format("%a");

        $factor = 3; // factor de activades / num. dias itinerario

        // Obtengo los POIS del destino
        $dataPois = $this->sygicApi->getDestinationPois($destination['idApi'],
$searchCategories, $factor*$numDays)->data->places;

        // Relleno los días del destino
        $destination['pois'] = $this->addDestinationPois($dataPois, $startDate,
$numDays, $destination['id']);
        $itinerary['destinations'] = $destination;

        // Devuelvo el itinerario
        return response()->json($itinerary, 200);
    }

    private function addDestinationPois (Array $poisList, DateTime $startDate, int
$numDays, int $destination_id) {
```

```
        $hora = 10;
        $actualDate = $startDate;
        foreach ($poisList as $poi) {
            if ($numDays >= 0) {

                $object = new stdClass();
                $object->id = '';
                $object->destination_id = $destination_id;
                $object->name = $poi->name;
                $object->idApi = $poi->id;
                $object->location = $poi->location;
                $object->duration = $poi->duration_estimate;
                $object->photo = $poi->thumbnail_url;
                $object->description = $poi->perex;

                if($object->duration > 3600) {
                    if ($hora == 10) {
                        $actualDate->setTime($hora, 00);
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 16;
                    } else if ($hora == 12) {
                        $hora = 16;
                        $actualDate->setTime($hora, 00);
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 10;
                        $actualDate->add(new DateInterval('P1D'));
                        --$numDays;
                    } else if ($hora == 16) {
                        $actualDate->setTime($hora, 00);
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 10;
                        $actualDate->add(new DateInterval('P1D'));
                        --$numDays;
                    }
                } else if ($object->duration <= 3600) {
                    if ($hora == 10) {
                        $actualDate->setTime($hora, 00);
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 12;
                    } else if ($hora == 12) {
                        $actualDate->setTime($hora, 00);
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 16;
                    } else if ($hora == 16) {
                        $actualDate->setTime($hora, 00);
```

```
                        $object->startDate = $actualDate->format('Y-m-d H:i:s');
                        $hora = 10;
                        $actualDate->add(new DateInterval('P1D'));
                        --$numDays;
                    }
                }

                $pois[] = $object;
            }
        }

        // Devolver Array
        return $pois;
    }
}
```

En las siguientes imágenes se muestra la lógica del componente creado en Angular para mostrar los itinerarios públicos.

```
import { Component, OnInit } from '@angular/core';
import { Itinerary } from '../../models/itinerary';
import { ItineraryService } from '../../services/itinerary.service';
import { UserService } from '../../services/user.service';

@Component({
  selector: 'itineraries',
  templateUrl: './itineraries.component.html',
  styleUrls: ['./itineraries.component.css'],
  providers: [UserService],
})

export class ItinerariesComponent implements OnInit {

  public pageTitle: string;
  public itineraries: Array<Itinerary>;
  public userToken;
  public identity;

  constructor(
    private _itineraryService: ItineraryService,
    private _userService: UserService,
  ) {
    this.pageTitle = 'Itinerarios de viaje';
  }
```

```
  ngOnInit() {
    this.identity = this._userService.getIdentity();
    this.getItineraries();
  }

  calculateDays(itinerary:Itinerary): number {
    let a = new Date (itinerary.startDate);
    let b = new Date (itinerary.endDate);
    let difference = b.getTime() - a.getTime();
    return difference / (1000 * 3600 * 24);
  }

  getItineraries(){
    this._itineraryService.getItineraries().subscribe(
      response => {
        if (response.data.status == 'success') {
          this.itineraries = response.data.itineraries;
          // console.log(this.itineraries);
        }
      },
      error => {
        console.log(<any>error);
      }
    );
  }

}
```

Anexo 3. Librerías/Código externo utilizado

Para la elaboración de la aplicación web TravelFit se han empleado los siguientes frameworks y librerías de código:

- Laravel[10]

  Framework PHP sobre el que está elaborado el backend.

- Angular[11]

  Framework para aplicaciones JavaScript empleado para el desarrollo del frontend de la aplicación.

- Bootstrap[12]

  Librería CSS para la elaboración del diseño y del grid de la aplicación, así como para la adecuación y comportamiento frente a pantallas y dispositivos de distinto tamaño.

- JQuery[13]

  Librería JavaScript para la manipulación de documentos HTML y los elementos que lo conforman. Requerida por Angular.

- ng-bootstrap[14]

  Librería de componentes para Angular basados en Bootstrap. Es una librería que aporta una librería de componentes predefinidos y de fácil uso para integrar dentro de Angular, bajo el diseño y funcionalidades de Bootstrap.

- Maptiler Geocoder[15]

  Componente JavaScript para realizar búsquedas geográficas sobre el servicio MapTiler. Se ha integrado dentro de Angular para realizar las funciones de búsqueda de destinos de un itinerario al comienzo del proceso.

- Leaflet[16]

  Librería de componentes para Angular basados en Bootstrap. Es una librería que aporta una librería de componentes predefinidos y de fácil uso para integrar dentro de Angular, bajo el diseño y

[10] *https://laravel.com/*
[11] *https://angular.io/*
[12] *https://getbootstrap.com/*
[13] *https://jquery.com/*
[14] *https://ng-bootstrap.github.io/#/home*
[15] *https://www.npmjs.com/package/@maptiler/geocoder#geocodersetlanguagelanguage*
[16] *https://leafletjs.com*

Anexo 4. Capturas de pantalla

Aquí se muestra una colección de capturas de pantalla de la aplicación final en funcionamiento.

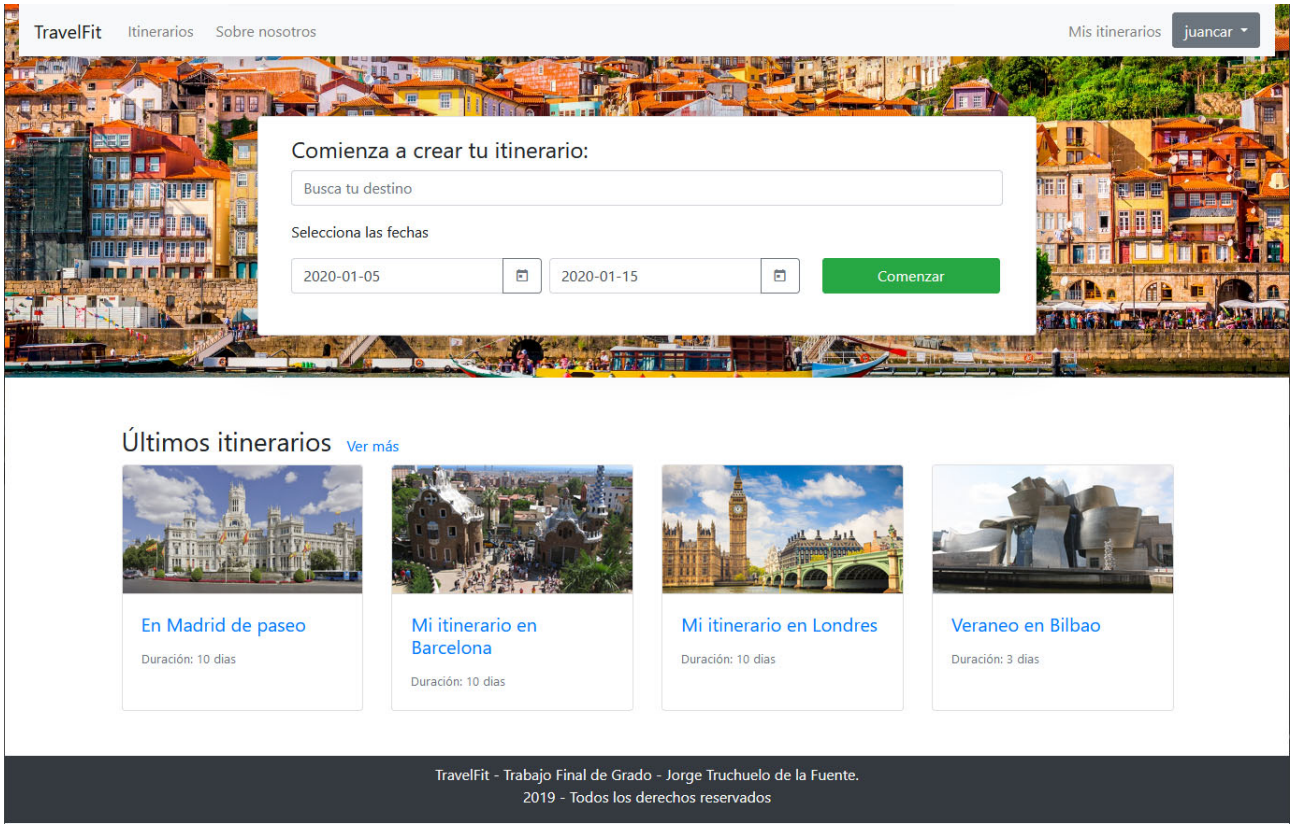


Figura 28. Pantalla Inicial TravelFit

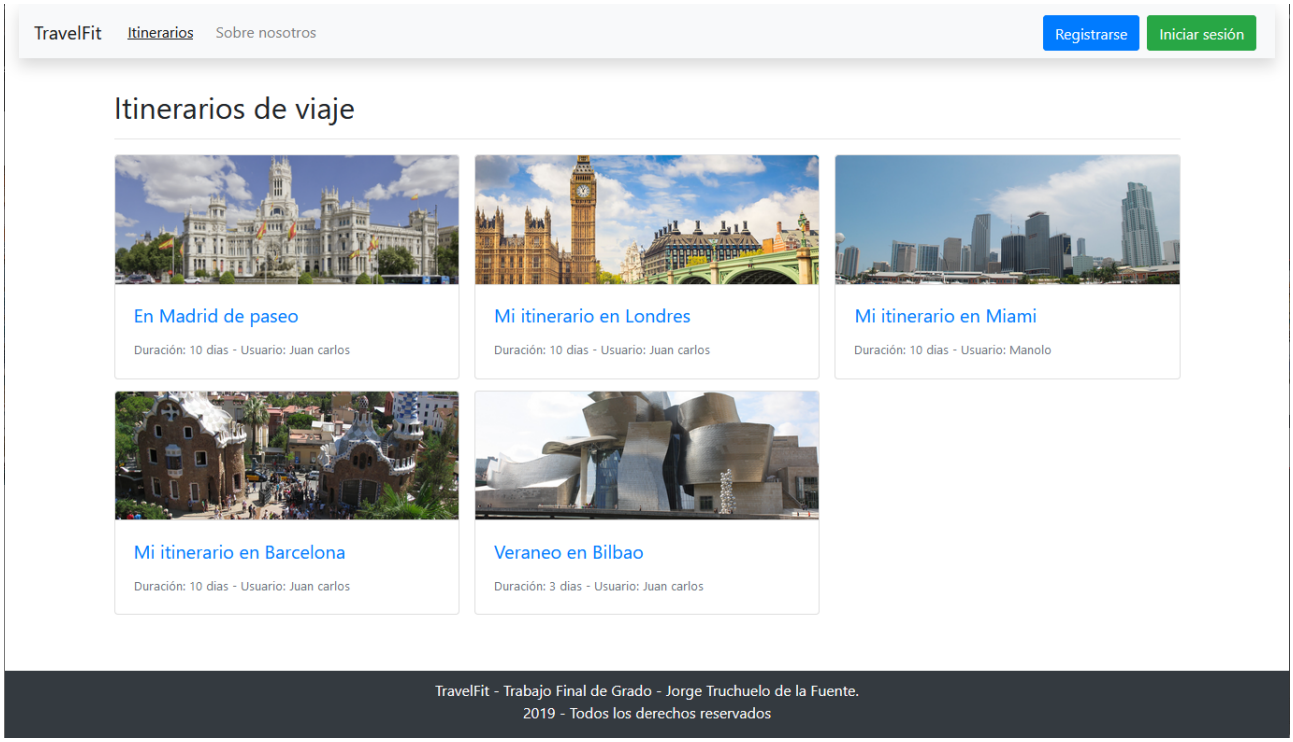


Figura 29. Pantalla de Itinerarios

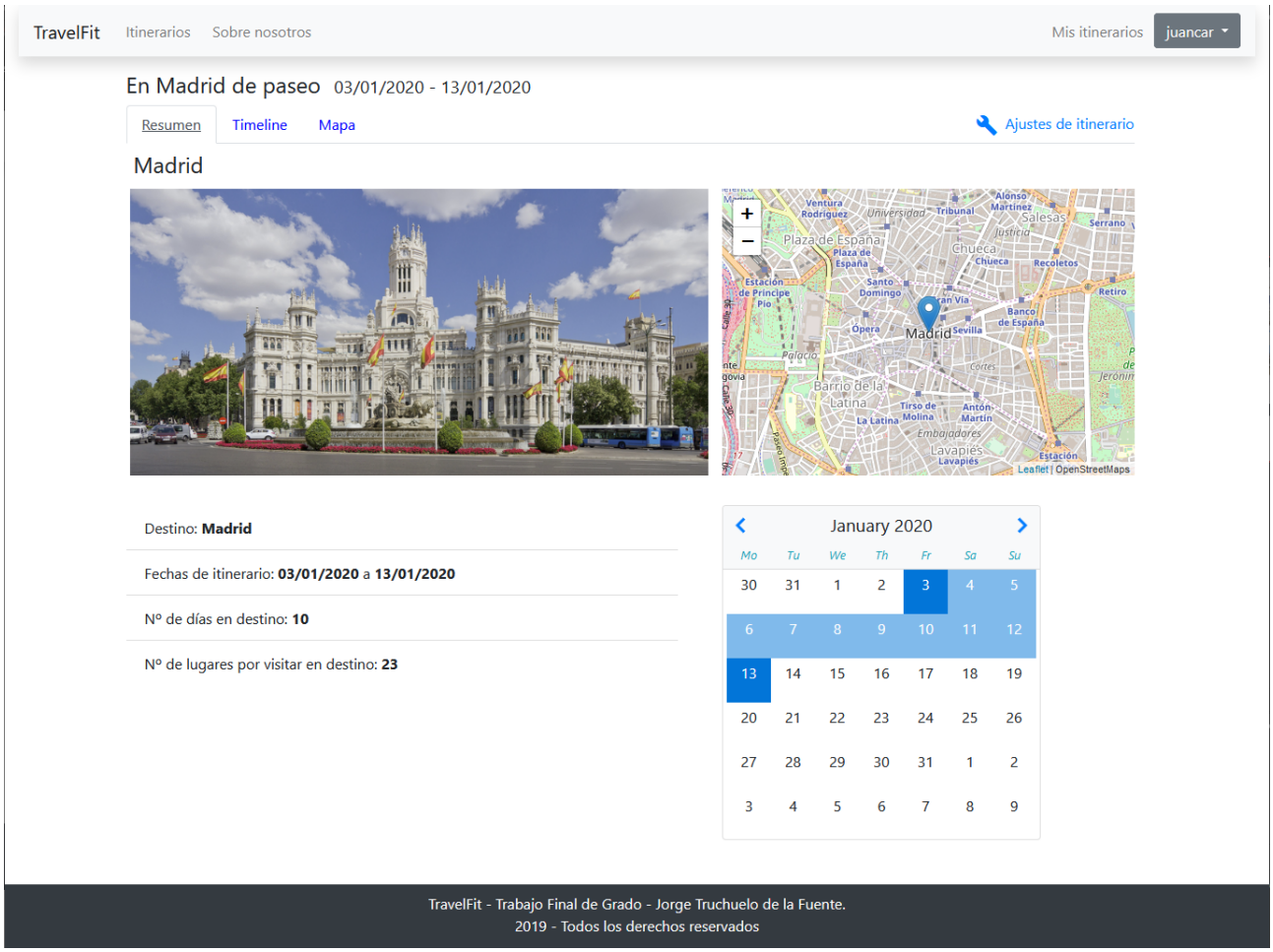


Figura 30. Pantalla Itinerario - Resumen

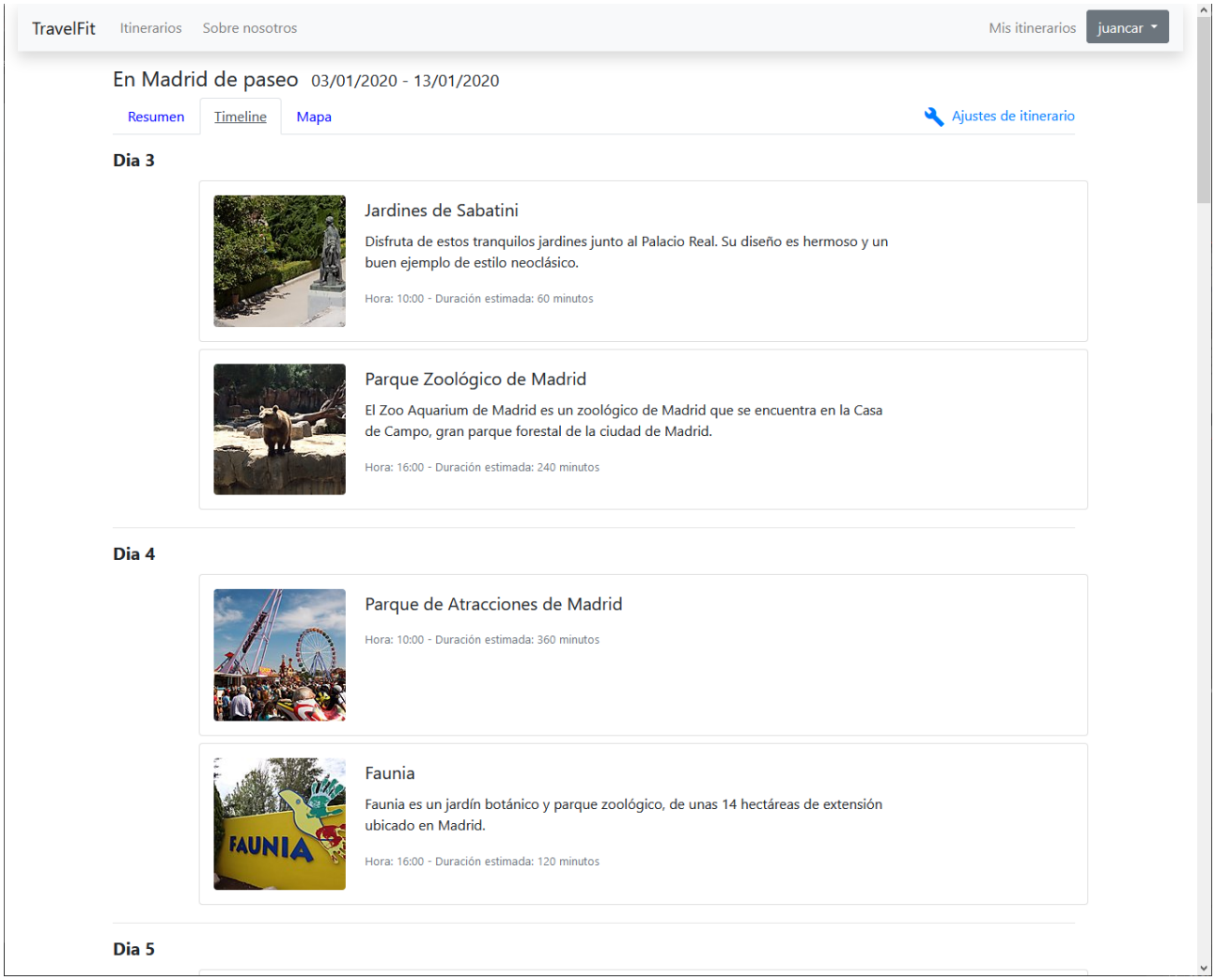


Figura 31. Pantalla Itinerario - Timeline

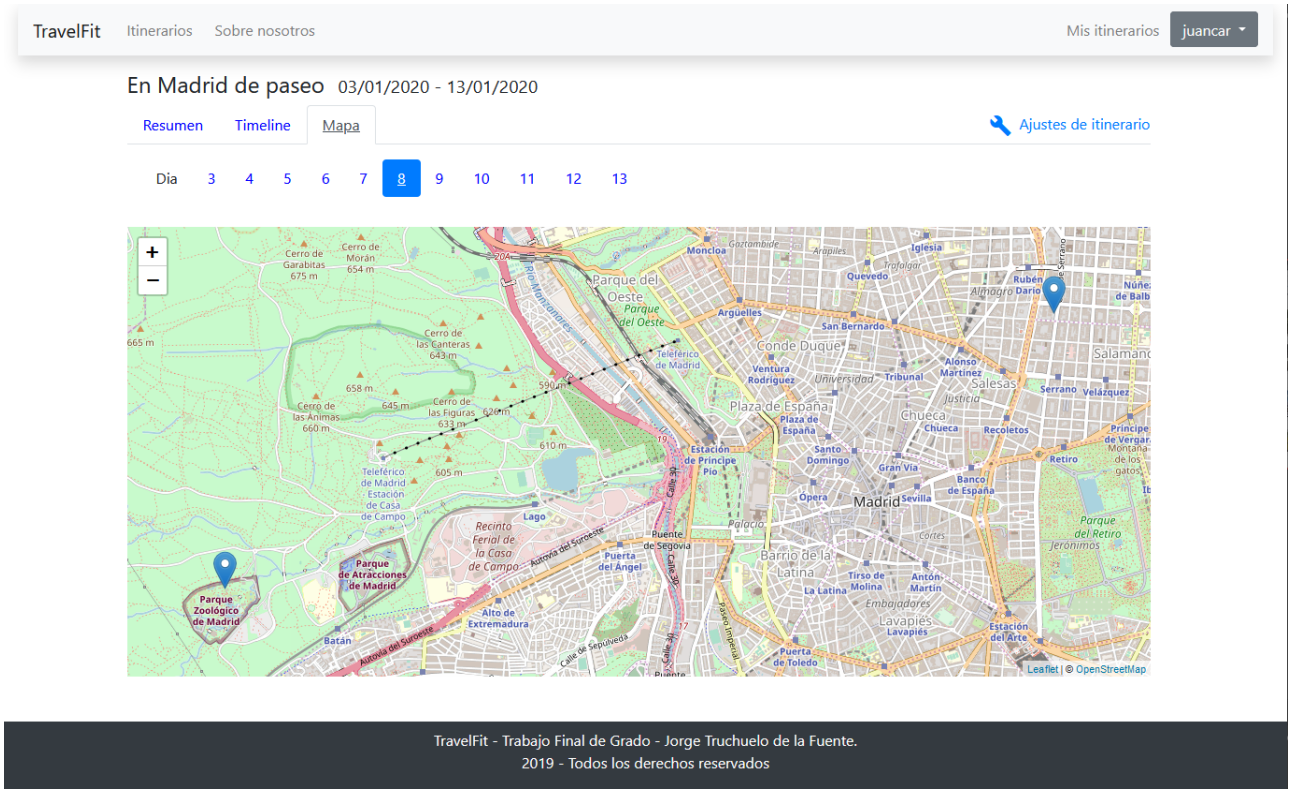


Figura 32. Pantalla Itinerario - Mapa

Anexo 5. Bibliografía

- *Manual de Angular* – [En línea] [Consulta: noviembre 2019] - https://desarrolloweb.com/manuales/manual-angular-2.html
- *Documentación oficial de Angular* – [En línea] [Consulta: 2019] - https://angular.io/
- *Documentación oficial Bootstrap Widgets* – [En línea] [Consulta: 2019] - https://ng-bootstrap.github.io/#/getting-started
- *Documentación official de Bootstrap* - [En línea] [Consulta: 2019] - https://getbootstrap.com/docs/4.4/getting-started/introduction/
- *A Complete Guide to Flexbox* - [En línea] [Consulta: 2019] - https://css-tricks.com/snippets/css/a-guide-to-flexbox/
- *Sygic Travel API* - [En línea] [Consulta: 2019] - http://docs.sygictravelapi.com/1.2/
- *Documentación oficial de Laravel* - [En línea] [Consulta: 2019] - https://laravel.com/docs/6.x/
- *Documentación de Laravel en español* - [En línea] [Consulta: 2019] - https://documentacion-laravel.com/
- *Build a REST API with Laravel API resources – Chimezie Enyinnaya* - [En línea] [Consulta: 2019] - https://blog.pusher.com/build-rest-api-laravel-api-resources/
- *Consuming third party APIs with Laravel Resources – Jeff* - [En línea] [Consulta: 2019] - https://medium.com/@jeffochoa/consuming-third-pary-apis-with-laravel-resources-c13a0c7dc945
- *Documentación oficial Leaflet* - [En línea] [Consulta: 2019] - https://leafletjs.com/reference-1.6.0.html

Anexo 6. Vita

Jorge Truchuelo de la Fuente, nacido en Madrid en 1983. Es titulado en Producción de Audiovisuales y Espectáculos por el IORTVE en el año 2009. Desde entonces, se ha dedicado a la producción audiovisual de *Motion Graphics* en diferentes empresas relacionadas con e-learning. Con gran pasión por la informática desde su infancia, ha buscado el nexo entre esta pasión y la producción de audiovisual.

Cinco años atrás, decide emprender un proyecto personal de cambio profesional en su vida, así, emprende el camino matriculándose en el Grado Multimedia en la UOC, combinando de este modo su trabajo con su pasión y con la esperanza de poder ampliar sus bases, nociones y conocimientos para continuar mezclando sus aficiones y aplicarlas en su desarrollo profesional.