Anexo IV: Manual del Programador

Trabajo de Fin de Grado
GRADO EN INGENIERÍA INFORMÁTICA

VNiVERSiDAD
D SALAMANCA

Septiembre de 2023

Autor
María González García

Tutores
Álvaro Lozano Murciego
María N. Moreno García

ÍNDICE

ILUSTRACIONES

1. Introducción

En el presente anexo se explica la estructura del directorio del código de la aplicación, así como el software necesario para instalar el entorno de desarrollo utilizado en la implementación del sistema y que permitirá llevar a cabo el mantenimiento de la plataforma.

En este anexo se hace referencia a conceptos y elementos del sistema y de la tecnología utilizada que están convenientemente documentados en el Anexo III.

2. Estructura del proyecto

Se describen a continuación las dos partes fundamentales del proyecto y sus repositorios de código para continuar su desarrollo.

2.1. Front-end

El código de la aplicación web se divide en varias carpetas principales que albergan los distintos archivos:

- **Carpeta /src**: en ella se encuentran los principales componentes que tendrán todas las vistas (barra de menú superior, pie de página, etc.), así como el archivo *App.vue,* que representa el menú de usuario y es la página principal a la que un usuario es redirigido cuando inicia sesión en el sistema.

- **Carpeta /src/pages**: en ella se encuentran el resto de archivos que conforman las otras vistas, tales como las que permiten modificar el apartado de exploración de itinerarios o el del historial de un usuario concreto.

- **Carpeta /src/pages/ruta**: contiene los archivos que representan cada uno de los pasos a completar cuando se realiza el proceso de nueva creación (la introducción de datos como fechas, la selección de puntos de interés, etc.).

- **Carpeta /src/service**: alberga los ficheros de JavaScript usados para la conexión con servicios externos, como para la comunicación con la API de Google Places.

- **Carpeta /src/stores**: en ella se encuentran las *stores* usadas para poder manejar la información del sistema, tales como la información del usuario, de los itinerarios, etc. Estos archivos son los requeridos cuando se utiliza Pinia.

Front-end

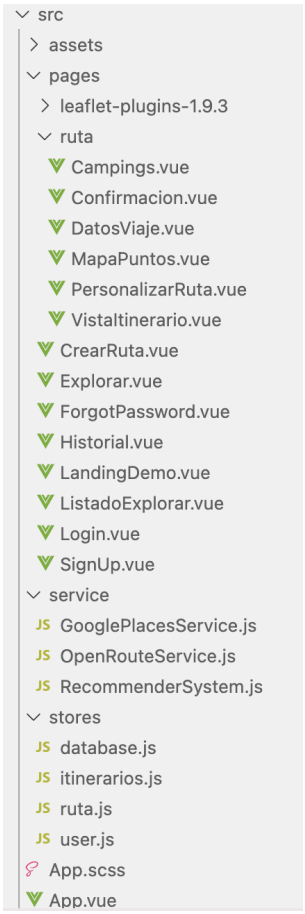


Ilustración 1 Distribución de los archivos y carpetas del proyecto de la aplicación web

El código se encuentra disponible en Github en el siguiente enlace:

https://github.com/MariaGG24/My-Itinerary

En el README se describen más detalles sobre la puesta en marcha del proyecto.

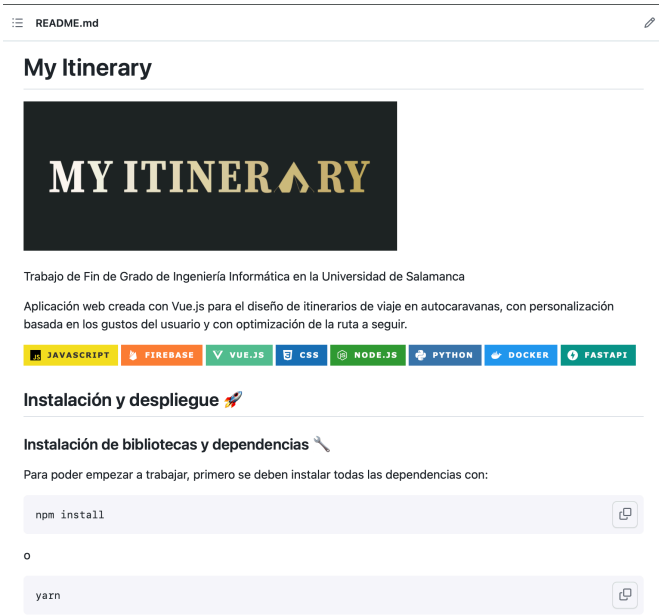


Ilustración 2 Extracto del README del repositorio Github del *front-end*

2.2. Back-end

Como parte correspondiente al *backend* de la aplicación, solo se dispone de un repositorio que alberga el sistema de recomendación desarrollado en forma de una API desarrollada en Python con el *framework* FAST API. El código de esta API se encuentra estructurado de la siguiente manera:

- **Carpeta app:** en ella se encuentran todos los archivos fuente que conforman la API desarrollada en Python. El archivo *main.py* es el fichero principal que hace uso del resto de los módulos Python desarrollados, cada uno en su fichero .py con responsabilidades bien definidas (interactuar con la API del LLM, de ORS, etc.).

- **Dockerfile:** es un archivo de texto que se utiliza para construir una imagen de Docker [1]. En este fichero se parte de una imagen base (como podría ser una distribución de Linux mínima), y se agregan capas de configuraciones y se instalan aplicaciones específicas requeridas por el sistema para que este pueda funcionar. Se hace esto para tener un entorno de despliegue replicable en cualquier sistema que soporte Docker. Este archivo es a su vez empleado para el despliegue, tanto en local para las pruebas como remoto en Fly.io [2] para el entorno de producción.

- **fly.toml**: archivo usado para la configuración y el despliegue de la API en la plataforma Fly.io. Recoge la configuración de la aplicación.

- **pyproject.toml:** archivo de configuración de proyecto Python que permite la definición y gestión de las dependencias, entornos virtuales y otros aspectos relaciones con la gestión de proyectos y sus herramientas. Lo emplea la biblioteca Poetry para la instalación de dependencias. Poetry [3] constituye una utilidad que simplifica la generación de un entorno virtual en Python, fundamentado en las dependencias del proyecto. Mediante esta herramienta, es posible especificar las bibliotecas de las que depende el proyecto, y Poetry procederá a instalarlas y mantenerlas actualizadas de manera automática.

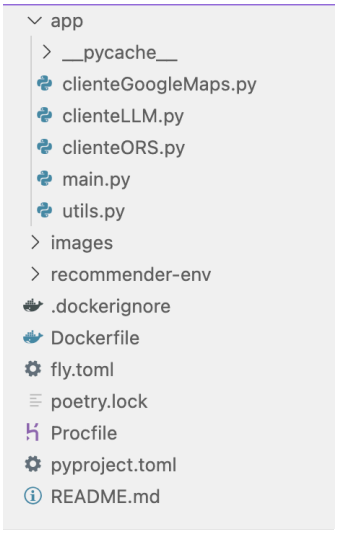


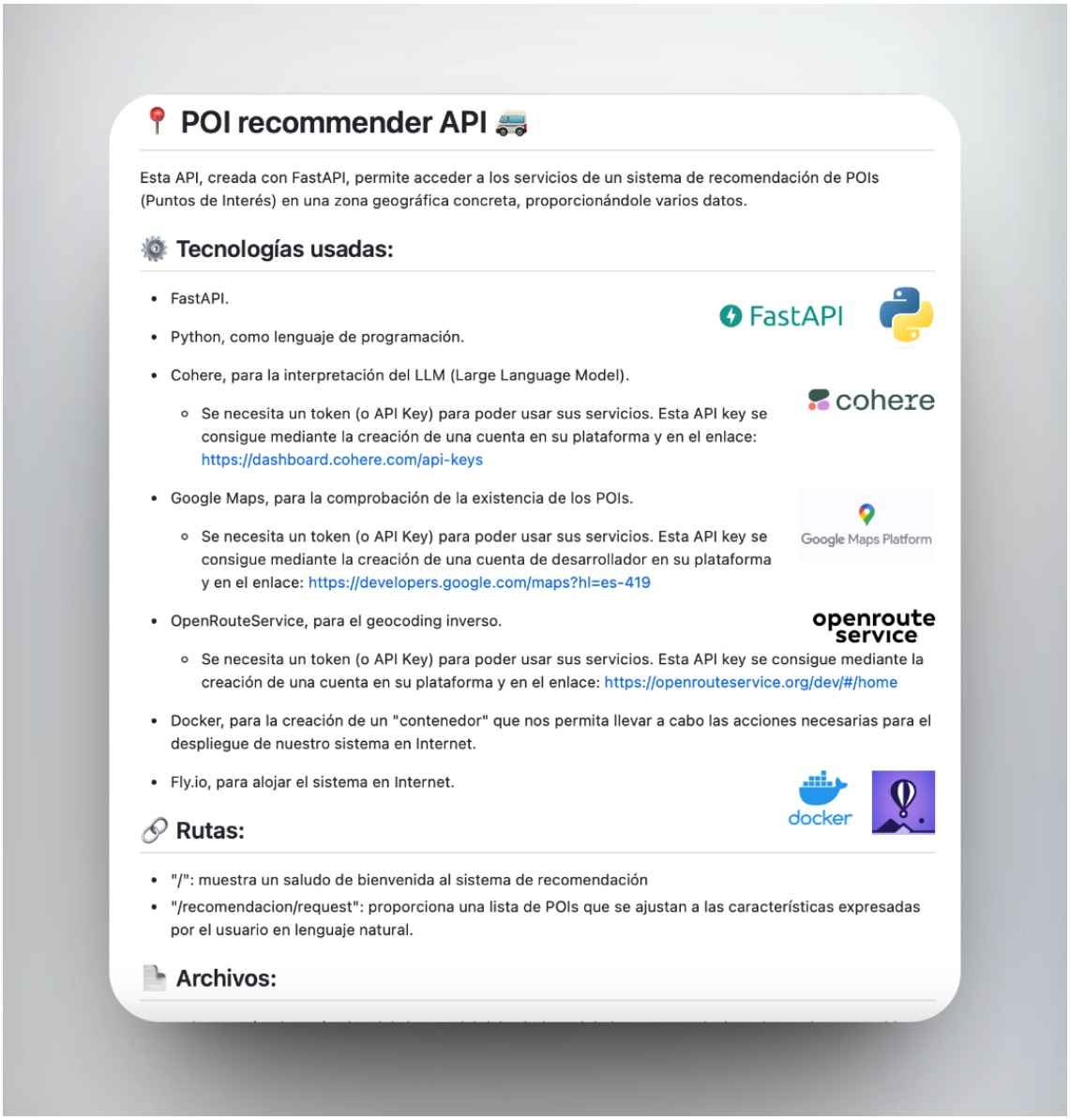
Ilustración 3 Distribución de los archivos y carpetas de la API del sistema de recomendación

Back-end

El código se encuentra disponible en Github en el siguiente enlace:

https://github.com/MariaGG24/recommender-api

En el README de este proyecto se describen más detalles sobre la puesta en marcha del proyecto.



📍 POI recommender API 🚐

Esta API, creada con FastAPI, permite acceder a los servicios de un sistema de recomendación de POIs (Puntos de Interés) en una zona geográfica concreta, proporcionándole varios datos.

⚙️ Tecnologías usadas:

- FastAPI.
- Python, como lenguaje de programación.
- Cohere, para la interpretación del LLM (Large Language Model).
  - Se necesita un token (o API Key) para poder usar sus servicios. Esta API key se consigue mediante la creación de una cuenta en su plataforma y en el enlace: https://dashboard.cohere.com/api-keys
- Google Maps, para la comprobación de la existencia de los POIs.
  - Se necesita un token (o API Key) para poder usar sus servicios. Esta API key se consigue mediante la creación de una cuenta de desarrollador en su plataforma y en el enlace: https://developers.google.com/maps?hl=es-419
- OpenRouteService, para el geocoding inverso.
  - Se necesita un token (o API Key) para poder usar sus servicios. Esta API key se consigue mediante la creación de una cuenta en su plataforma y en el enlace: https://openrouteservice.org/dev/#/home
- Docker, para la creación de un "contenedor" que nos permita llevar a cabo las acciones necesarias para el despliegue de nuestro sistema en Internet.
- Fly.io, para alojar el sistema en Internet.

🔗 Rutas:

- "/": muestra un saludo de bienvenida al sistema de recomendación
- "/recomendacion/request": proporciona una lista de POIs que se ajustan a las características expresadas por el usuario en lenguaje natural.

Archivos:

Ilustración 4 README del repositorio de la API

3. Documentación del código

Para realizar la documentación de cada uno de los ficheros del código, se ha utilizado la herramienta JSDoc [4]. Este estándar hace posible la documentación del código fuente mediante comentarios hechos en el mismo.

Su instalación se ha hecho mediante NodeJS con la orden:

```
npm install -g jsdoc
```

Una vez instalado JSDoc, es necesario tener una plantilla. En este caso, se ha empleado la plantilla *docdash* [5] debido a su claridad y agradable aspecto. Para instalarla, se introduce la orden:

```
npm i docdash
```

Después, se modifica el archivo *package.json* del proyecto, añadiendo una nueva línea de scripts para poder ejecutarlo más fácilmente:

Ilustración 5 Modificación del archivo *package.json*

Finalmente, para que este comando pueda funcionar, es necesario añadir un archivo de configuración al proyecto. Este archivo puede encontrarse en el siguiente repositorio de GitHub:

https://github.com/lpwj/five_minutes_tutorials/blob/jsdoc/jsDoc/src/jsdocConf.json

A continuación, se muestra un ejemplo de la documentación generada, que se aloja en la carpeta ***docs*** del repositorio:

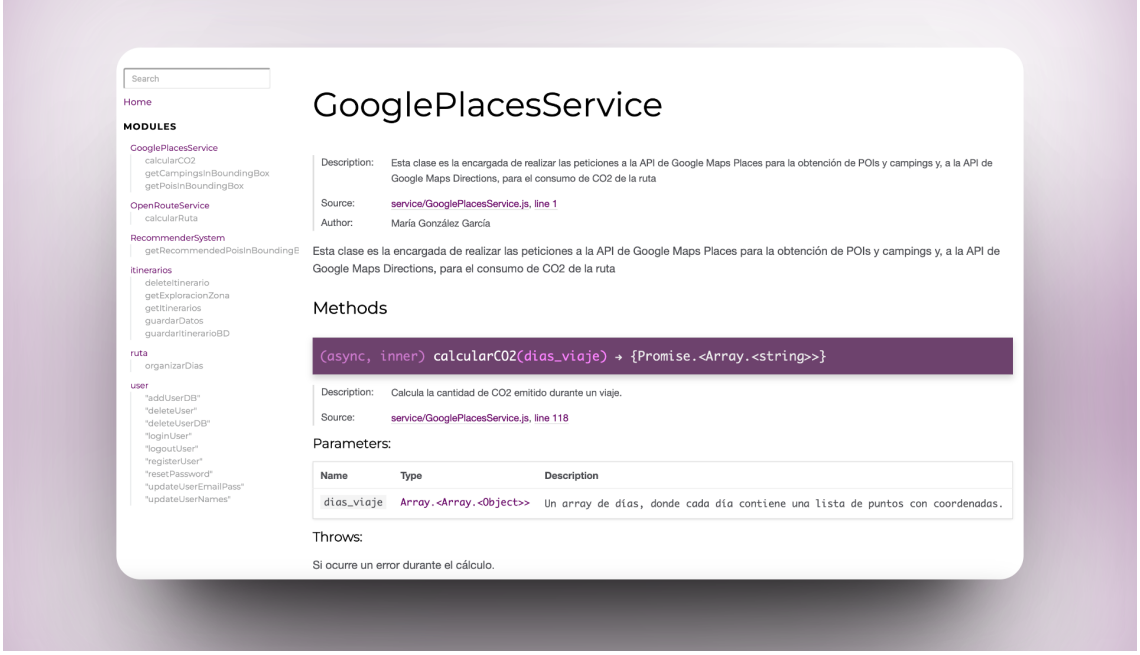


Ilustración 6 Ejemplo de la documentación generada

4. Entorno de desarrollo

Para el desarrollo de la aplicación se ha usado el *framework* Vue.js y como lenguaje de programación JavaScript.

Para el trabajo realizado con JavaScript, se ha requerido de la instalación de NodeJS [6], que es un gestor de paquetes conocido como ***npm*** para la instalación de las dependencias necesarias del proyecto.

4.1. IDE Recomendado para el desarrollo y configuración

Como IDE de desarrollo se ha utilizado VSCode o Visual Studio Code [7]. Este IDE es gratuito y de código abierto, lo que hace que sea una buena opción respecto a otros que presentar más restricciones en su uso. Además, permite la instalación de diversos *plugins* o extensiones que permiten un desarrollo más rápido, como es el caso de herramientas como *Codeium*[8], o que aportan funcionalidades que hacen tareas generalmente tediosas, como la documentación de los códigos, en algo simple y rápido que permite centrarse en otras actividades de más peso para el desarrollo de aplicaciones, como es el caso de JSDoc [4].

Además, VSCode permite la conexión con software de control de versiones, como Git [9], que hace posible un mejor mantenimiento de versiones de las aplicaciones, informando al desarrollador de los cambios realizados desde la última vez que se comprometió el código y permitiendo el compromiso por parte de varios autores

simultáneamente, estableciendo varias ramas que pueden llegar a converger, entre otras características.

4.2. Firebase

Al trabajar con *Firebase*, es necesaria una manera de comunicarse con sus servicios (*Hosting, Firestore*). Para ello, se instala su herramienta globalmente conocida como ***Command Line Interface*** (CLI) de *Firebase* mediante las órdenes siguientes:

```
$ npm install-g firebase-tools
```

Para iniciar sesión en esta plataforma y poder llevar a cabo las acciones necesarias como desarrollador, se usa la orden:

```
$ firebase login
```

Para desplegar la aplicación, se utiliza:

```
$ firebase deploy
```

El comportamiento de esta última orden viene determinada por el archivo *firebase.json*.

4.3. Vue.js

Para trabajar en el *frontend*, se debe instalar el CLI de Vue [10], que permite compilar el proyecto y levantar un servidor web local para el desarrollo del mismo. Para instalar el CLI de Vue se usa la orden:

```
$ npm install-g @vue/cli
```

Para instalar las dependencias y desplegar un servidor local:

```
$ npm install
$ npm run serve
```

Para compilar el proyecto con el fin de ser desplegado:

```
$ npm run build
```

4.4. Sistema de recomendación

4.4.1. Entorno de desarrollo

El sistema de recomendación empleado ha sido realizado en Python con el *framework* Fast API, como se ha mencionado anteriormente. Para empezar con el desarrollo, lo primero que se debe hacer es crear un directorio en la carpeta que se desee:

```
$ cd carpetaContenedora
$ mkdir nuevoDirectorio
  $ cd nuevoDirectorio
```

Una vez ya en el nuevo directorio, se crea un entorno virtual de Python empleando el módulo *venv*. Esto se consigue con la orden:

```
$ python-3 -m venv nombreEntorno-env
```

El siguiente paso es activar el entorno virtual. Esta acción permite que cambie temporalmente el entorno de Python de la sesión actual de la terminal, pudiendo así utilizar el entorno virtual especificado. En este caso, la orden para sistemas UNIX es como sigue:

```
$ source nombreEntorno-env/bin/activate
```

Una vez introducido el comando, es posible instalar las dependencias y herramientas necesarias para poder empezar a trabajar. Se deben instalar el *framework* Fast API y la herramienta *uvicorn*, que es un servidor web ASGI (*Asynchronous Server Gateway Interface*) de alto rendimiento que se utiliza comúnmente con marcos de trabajo modernos como el empleado. Para ello, se utiliza la orden:

```
$ pip install fastapi
$ pip install uvicorn
```

Cuando ya se hayan instalado todas las herramientas requeridas, se creará un fichero llamado *main.py*, que será el que permita realizar las acciones necesarias para que la API pueda ofrecer las funcionalidades correspondientes. Los métodos implementados se podrán ir probando de manera local. Esto se consigue con la orden:

```
$ uvicorn app.main:app --reload
```

Al introducir el comando anterior, se mostrará una dirección a la que se podrá acceder desde cualquier navegador y, de esta forma, ir comprobando el correcto funcionamiento de lo implementado. Además, con la opción *--reload* se indica que, según se vayan realizando cambios en los archivos del directorio, se vaya recargando la dirección para estar siempre actualizados.

4.4.2. Documentación de la API

De entre las muchas características y funcionalidades que aporta Fast API, una de ellas es la de generación automática de documentos. Es decir, cuando se crea una API con esta herramienta, automáticamente se generará un documento con toda la información de los métodos accesibles, detallando el tipo de método, los parámetros requeridos y las posibles respuestas por parte del sistema, entre otros datos. Así como también permite realizar pruebas en las que se pueda apreciar su funcionamiento.

La documentación del sistema desarrollado se encuentra en el siguiente enlace:

https://myitineraryrecommender.fly.dev/docs

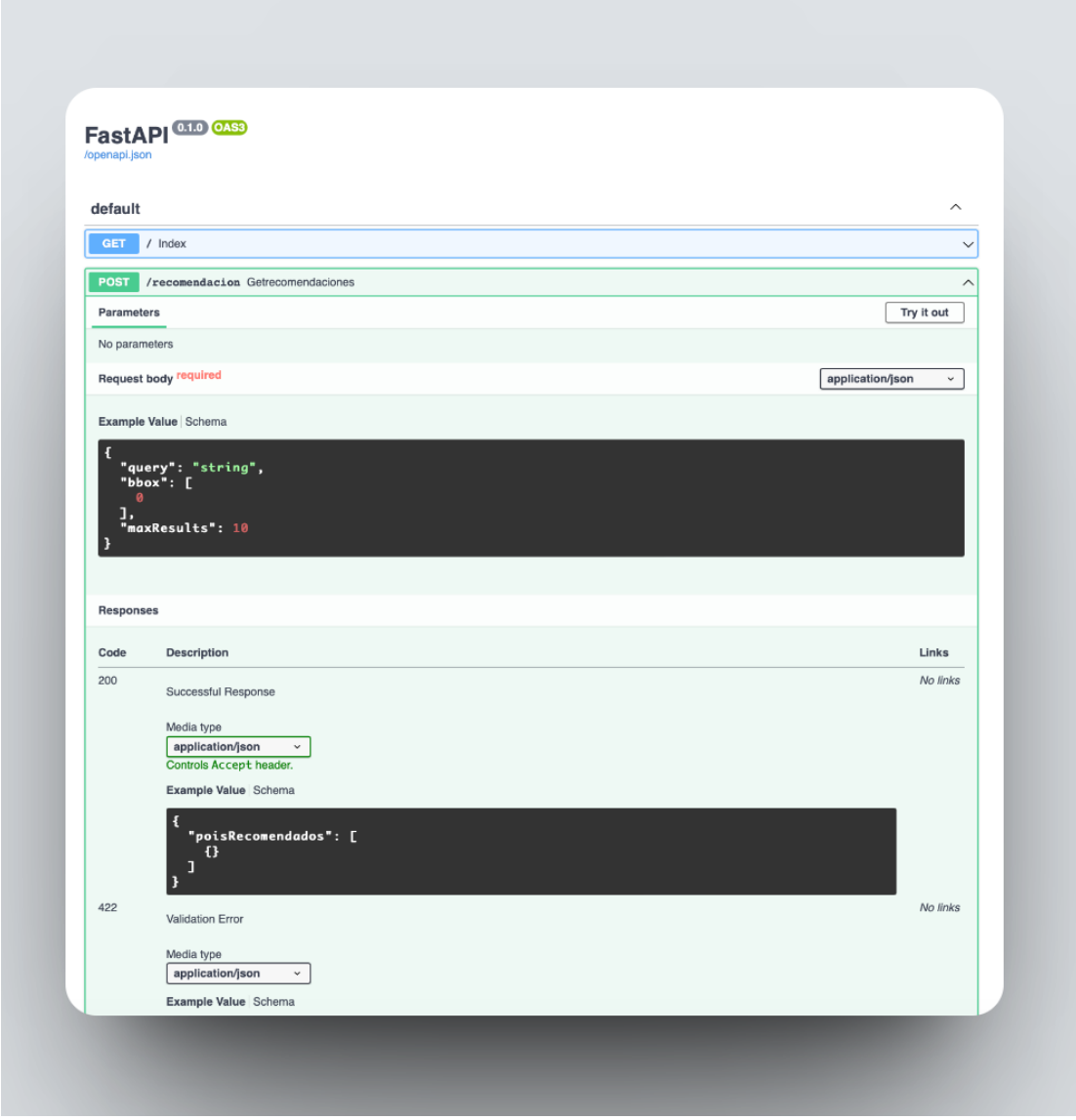


Ilustración 7 Documentación automática de la API desarrollada

4.4.3. Despliegue de la API en Fly.io

Una vez que la API esté lista y funcione correctamente de manera local, es necesario realizar su despliegue para que pueda ser accedida remotamente. Para ello, se usa Fly.io, como se ha comentado anteriormente. Lo primero que se debe hacer es instalar la herramienta (para otro sistema operativo que no sea MacOS, se debe consultar la documentación de la plataforma [11]):

```
$ curl -L https://fly.io/install.sh | sh
```

Tras la instalación, se deben copiar las siguientes líneas en el fichero de configuración del dispositivo en el que se trabaje:

```
export FLYCTL_INSTALL="/Users/nombreUsuario/.fly"
     export PATH="$FLYCTL_INSTALL/bin:$PATH"
```

Después, se inicia sesión con las credenciales correspondientes mediante la orden:

```
$ flyctl auth login
```

Para poder crear la aplicación que será desplegada, se emplea la orden:

```
$ flyctl launch
```

Finalmente, para desplegarla se hace uso del comando:

```
$ flyctl deploy
```

Cuando todo el proceso haya sido realizado, se podrá abrir la nueva aplicación mediante:

```
$ flyctl open
```

También será necesario configurar los secretos necesarios (API KEYS) que necesita la aplicación en producción. Esto es posible realizarlo desde la propia consola de configuración de Fly.io

4.5. Configuración de secretos de servicios externos

Tanto en la parte de cliente como en la del sistema de recomendación se hace uso de servicios externos que deberán ser configurados con sus debidas API KEYS. Se presentan a continuación los servicios utilizados y dónde conseguir estas credenciales para su uso.

4.5.1. Cohere

Cohere [12] es una plataforma que proporciona modelos de procesamiento de lenguaje natural, ayudando a las interacciones persona–ordenador.

En este contexto, se usa para el sistema de recomendación. Su función se basa en recolectar la descripción dada por el usuario, donde se especifican sus gustos o su propósito para este viaje concreto, y crear un *prompt* que permita ser procesado por un LLM (*Large Language Model*).

Para su utilización, es necesario conectarse a su API gracias a un *token* o una *API Key*, la cual se puede conseguir mediante el registro en su plataforma y, una vez realizado el registro, en el siguiente enlace: https://dashboard.cohere.com/api-keys

4.5.2. OpenRouteService

OpenRouteService [13] es una plataforma que ofrece una gran variedad de geo–servicios, como *geocoding*, cálculo de rutas optimizadas entre varios puntos geográficos, etc. Además de la amplia gama de servicios que ofrece, también hay que destacar que es de uso gratuito y código abierto, así como que el acceso a todos estos servicios se hace mediante la conexión a una API única.

Para su utilización, es necesario conectarse a su API gracias a un *token* o una *API Key*, la cual se puede conseguir mediante el registro en su plataforma y, una vez realizado el registro, en el siguiente enlace: https://openrouteservice.org/dev/ - /home

4.5.3. Google Maps

Google, además de ser uno de los buscadores más utilizados a nivel global, también cuenta con una gran cantidad de servicios que se ofrecen de manera gratuita (al menos hasta que no se supere una cantidad determinada de solicitudes). Además de la base de datos NoSQL que es *Firebase* y que se ha mencionado anteriormente, Google cuenta con una API de mapas que permite obtener distinta información.

En este contexto, se han utilizado dos de estos servicios:

- **Google Maps Places** [14]: permite realizar acciones como la búsqueda de lugares en un mapa dado el nombre, o el filtrado de lugares que se ajustan a unos determinados parámetros proporcionados (como el tipo de establecimiento o sitio

a buscar). Este servicio se ha utilizado tanto en la parte de la aplicación web, para poder localizar aquellos lugares destacables en la zona de visita del usuario, así como también en la parte del sistema de recomendación, donde su utilización se ha enfocado más en la comprobación de la existencia de aquellos lugares que devolvía la inteligencia artificial.

- **Google Maps Directions** [15]: permite calcular rutas entre varios puntos geográficos, entre otras características, pero, en este caso, se ha empleado en la parte de la aplicación web para el cálculo del consumo de CO2 producido por los trayectos a realizar por el usuario.

Ambos servicios son accesibles gracias a la misma *API Key*, la cual se consigue mediante la creación de una cuenta de desarrollador en su plataforma y, una vez realizado el registro, en el siguiente enlace: https://developers.google.com/maps?hl=es-419

En este caso, una vez que se haya creado la cuenta, habrá que dirigirse al perfil de desarrollador que se haya creado y, en el apartado de "APIs y servicios habilitados", indicar todos aquellos a los que se desee acceder con nuestra *API Key*.

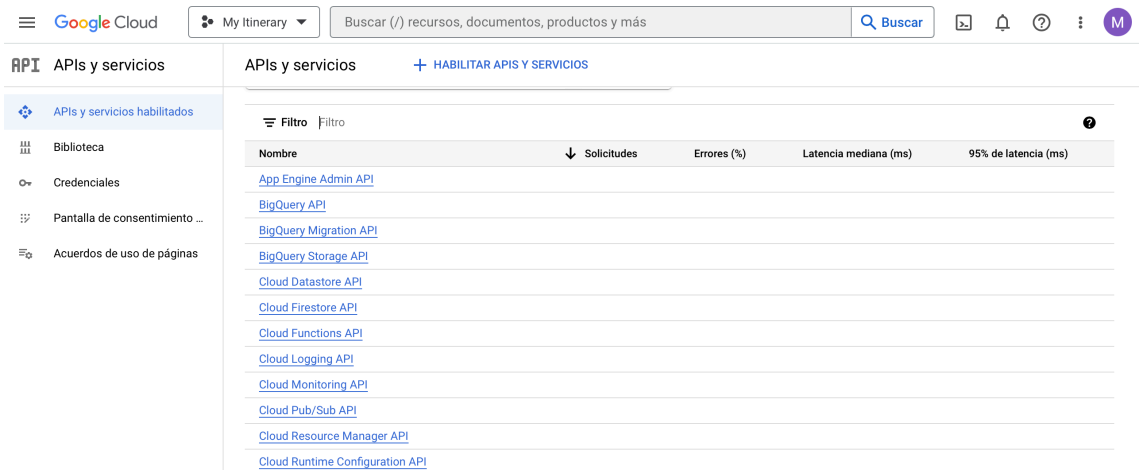


Ilustración 8 Catálogo de servicios ofrecidos por Google

4.6. PWA

Para convertir el proyecto en una PWA, es necesario instalar el *plugin* correspondiente. Esta acción se hace mediante Node.js:

```
npm i @vue/cli-plugin-pwa
```

Para poder adecuar las características de la PWA a los datos del proyecto, se añade en *package.json* un campo "*cli-plugin-pwa"* que contenga toda la información que se desee:

```
"cli-plugin-pwa": {
  "name": "My Itinerary",
  "themeColor": "#C5B06A",
  "msTileColor": "#C5B06A",
  "appleMobileWebAppCapable": "yes",
  "appleMobileWebAppStatusBarStyle": "black",
  "iconPaths": {
    "faviconSVG": "./public/img/icons/tentIcon.png",
    "favicon32": "./public/img/icons/tentIcon.png",
    "favicon16": "./public/img/icons/tentIcon.png",
    "appleTouchIcon": "./public/img/icons/tentIcon.png",
    "maskIcon": "./public/img/icons/tentIcon.png",
    "msTileImage": "./public/img/icons/tentIcon.png"
  }
},
```

Ilustración 9 Modificación de *package.json* para la PWA

Una vez que esté instalado, para comprobar si se ha tenido éxito en la tarea, se debería observar el siguiente icono (mostrado en el recuadro rojo en la ilustración 10) cuando se acceda a la aplicación web:

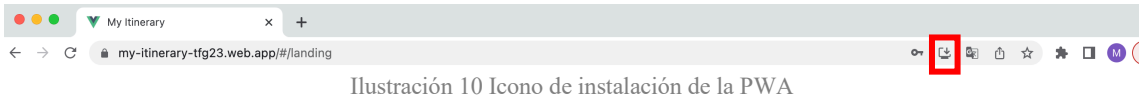


Ilustración 10 Icono de instalación de la PWA

5. Referencias

[1] “Docker: Accelerated Container Application Development.” https://www.docker.com/ (accessed Aug. 27, 2023).

[2] “Fly.io.” https://fly.io/ (accessed Aug. 27, 2023).

[3] “Poetry - Python dependency management and packaging made easy.” https://python-poetry.org/ (accessed Sep. 02, 2023).

[4] “JSDoc.” https://jsdoc.app/ (accessed Aug. 27, 2023).

[5] “clenemt/docdash: :zap: Lodash inspired JSDoc 3 template/theme.” https://github.com/clenemt/docdash (accessed Sep. 04, 2023).

[6] “About | Node.js.” https://nodejs.org/en/about (accessed Jul. 04, 2023).

[7] “Visual Studio Code - Code Editing. Redefined.” https://code.visualstudio.com/ (accessed Aug. 27, 2023).

[8] “Codeium · Free AI Code Completion & Chat.” https://codeium.com/ (accessed Aug. 27, 2023).

[9] “Git.” https://git-scm.com/ (accessed Aug. 27, 2023).

[10] “Vue CLI.” https://cli.vuejs.org/ (accessed Jul. 04, 2023).

[11] “Install flyctl · Fly Docs.” https://fly.io/docs/hands-on/install-flyctl/ (accessed Aug. 27, 2023).

[12] “About | Cohere.” https://cohere.com/about (accessed Aug. 27, 2023).

[13] “Openrouteservice.” https://openrouteservice.org/ (accessed Aug. 27, 2023).

[14] “Biblioteca de Places | API de Maps JavaScript | Google for Developers.” https://developers.google.com/maps/documentation/javascript/places?hl=es-419 (accessed Aug. 27, 2023).

[15] “Servicio Directions | API de Maps JavaScript | Google for Developers.” https://developers.google.com/maps/documentation/javascript/directions?hl=es-419 (accessed Aug. 27, 2023).