Práctica de servicios Web (I): servidores

Alvaro Tapia Muñoz, Jose Luis Bueno Pachón, Carlos Marín Corbera, Carmen González Ortega, Altair Bueno Calvente

5 nov 2022

Contents

Introducción

Para la realización de los servicios REST propuestos en el documento *Práctica de servicios Web (I): servidores*, se ha optado un desarrollo basado en microservicios, donde cada uno de los servicios planteados son servidores independientes.

Tecnologías Utilizadas

Todos los microservicios están desarrollados haciendo uso de Python. Más concretamente, se ha utilizado el framework FastAPI para el desarrollo de las APIs REST.

Para la persistencia de datos se ha optado por la base de datos de MongoDB, conectada a través del driver asíncrono Motor.

Instrucciones de Despliegue

Se facilita el fichero `docker-compose.yml` para poder hacer un despliegue de todos los microservicios utilizando la herramienta Docker Compose. Instrucciones más detalladas están disponibles en el fichero `README.md`

Para facilitar el despliegue de cada aplicación de forma individual, se proporciona un fichero `Dockerfile` para construir un contenedor de Docker. Las instrucciones detalladas sobre como utilizar la imagen y las opciones de configuración disponibles se pueden encontrar en el fichero `README.md`, dentro del directorio correspondiente a cada microservicio.

En caso de no disponer de las herramientas mencionadas, dentro del mismo fichero `README.md` en cada microservicio, se indica paso a paso cómo realizar un despliegue con uvicorn.

Esquema de las entidades

Los documentos almacenados en Mongo mantienen el siguiente esquema:

```
{
  // Clave primaria de Mongo
  "_id": "ObjectId",
  // Titulo de la vivienda
  "title": "string",
  // Descripción de la vivienda
```

```
  "description": "string",
  // Identificador del usuario dueño de la vivienda
  "user_id": "string",
  // Ubicación de la vivienda
  "location": "string",
  // Estado de la vivienda. Por defecto: available
  "state": "enum(available,deleted)",
  //Lista de fotos de la vivienda
  "url_photo": ["string"],
  //Coordenada geográfica longitud
  "longitude": "string",
  //Coordenada geográfica latitud
  "latitude": "string",
  // Lista de todas las reservas que ha recibido esta vivienda
  "bookings": [
    {
      "_id": "ObjectId",
      // Identificador del usuario que hace la reserva
      "user_id": "string",
      // Fecha de inicio de la reserva en formato YYYY-MM-DD
      "start_date": "string",
      // Fecha de fin de la reserva en formato YYYY-MM-DD
      "end_date": "string",
      // Estado de la reserva. Por defecto: reserved
      "state": "enum(reserved,canceled)"
    }
  ]
}
```

Datos de prueba para Mongo

En el interior de la carpeta `iweb`, se proporciona un fichero `houses.json` con el volcado de la colección de pruebas. Para restaurarla, basta con realizar una inserción de multiples elementos del contenido del fichero. Para automatizar el proceso, se proporciona un script de Python `iweb.py`. Requiere que la librería pymongo esté instalada

A2ReservasREST

Este microservicio se encarga de proporcionar los datos sobre las reservas.

Requisitos considerados

- **Al menos dos operaciones de consulta o búsqueda parametrizada**: `GET /booking`, `GET /booking/{booking_id}`, `DELETE /booking/{booking_id}`
- **Una operación de consulta sobre las relaciones entre las entidades**: `GET /booking` y sus parámetro de consulta `owner_id` y `house_id`

Endpoints REST disponibles

La siguiente lista es una especificación informal sobre los endpoints REST disponibles en el microservicio, a modo de resumen. La documentación completa se puede encontrar en el propio servidor, bajo las rutas `/docs` (SwaggerUI) y `/redoc` (Redoc). Además, se adjunta una copia local en el fichero `openapi.json`, dentro de la carpeta del proyecto.

- `GET /booking`: Devuelve una lista de reservas que cumplan con los filtros especificados. 10 reservas como máximo. Los parámetros de consulta son:
  - `skip`: Número de reservas a ignorar
  - `user_id`: Identificador de usuario (quien realiza la reserva)
  - `owner_id`: Identificador de usuario (propietario de la vivienda)
  - `before_date`: Fechas de fin anteriores
  - `after_date`: Fechas de inicio posteriores
  - `sort_by`: Campo utilizado para ordenar
  - `ascending`: Ordenar de forma ascendente. Por defecto: `false`
- `POST /booking`: Crea una nueva reserva. El cuerpo de la petición es un json con los siguientes campos:
  - `user_id`: Identificador del usuario que realiza la reserva
  - `house_id`: Identificador de la vivienda
  - `start_date`: Fecha de inicio de la reserva
  - `end_date`: Fecha de inicio de la reserva
- `GET /booking/{booking_id}`: Devuelve toda la información sobre la reserva con identificador `booking_id`
- `DELETE /booking/{booking_id}`: Cancela la reserva con identificador `booking_id`
- `GET /ping`: Ruta utilizada para validar que el servicio se encuentra disponible

Casos alternativos

En caso de error en una petición bien formada, se devolverá un mensaje de error siguiendo el siguiente formato.

```
{
  "detail": "error"
}
```

Donde el campo `detail` contiene uno de los siguientes códigos de error

- `ALREADY_BOOKED` (409 Conflict): No se puede reservar por un conflicto
- `NOT_FOUND` (404 Not Found): No se ha encontrado la vivienda

Datos Abiertos

El microservicio encargado de servir los datos abiertos es `A2datosabiertosREST`

Requisitos considerados

- **Al menos tres operaciones de consulta o búsqueda parametrizada sobre dichos datos abiertos**: `GET /gas-stations`, `GET /gas-stations/{latitude}/{longitude}`, `GET /average-stay`

Precio de carburantes en las gasolineras españolas

Hemos escogido un conjunto de datos abiertos con información sobre todas las gasolineras de España, incluyendo información sobre su posición geográfica, dirección y precio de los carburantes ofertados. Este será utilizado para mostrar en un mapa las gasolineras cercanas a una vivienda publicada en la aplicación, o las gasolineras en una determinada provincia. Si no se encuentran resultados en la búsqueda, el endpoint devuelve un mensaje indicándolo.

Endpoints

- `GET /gas-stations`: Devuelve una lista de gasolineras (`List[EESSPrecio]`) filtrados por provincia y rótulo. Por defecto 10 gasolineras como máximo
  - `provincia`: `Optional[str]`. Nombre de la provincia

  - `rotulo`: `Optional[str]`. Nombre de la marca o rótulo de la gasolinera
  - `limit`: `int`. Número máximo de elementos a devolver. Por defecto, el valor es 10

- `GET /gas-stations/{latitude}/{longitude}`: Devuelve una lista de gasolineras (`List[EESSPrecio]`) que se encuentran como máximo en un área a partir de una geolocalización. Por defecto 10 gasolineras como máximo.

  - `latitude`: `float`. Latitud de la vivienda
  - `longitude`: `float`. Longitud de la vivienda
  - `area`: `int`. Límite del área en kilómetros. Por defecto, el valor es 5
  - `limit`: `int`. Número máximo de elementos a devolver. Por defecto, el valor es 10

El modelo del tipo de salida `EESSPrecio` se puede ver en `A2datosabiertosREST/src/models/gas_stati`

Estancia media de los viajeros por provincias y meses

El segundo conjunto de datos abiertos escogido contiene información de la estancia media en días de los viajeros por provincia y mes. Esto será utilizado para mostrar la media de días de estancia en función de la provincia donde se esté realizando la búsqueda, pudiendo filtrarse por mes específico o la media anual.

Endpoints

- `GET /average-stay`: Devuelve el valor de la estancia media (`Data`) de viajeros de una provincia en un mes o año.

  - `provincia`: `str`. Nombre de la provincia
  - `mes`: `Optional[str]`. Nombre del mes

El modelo del tipo de salida `Data` se puede ver en `A2datosabiertosREST/src/models/average_stay.py`

Casos Alternativos

En el caso de que no se introduzca una entrada correcta, se han definido dos excepciones que se elevarán cuando sea oportuno: `NoGasStations` y `NoDataFound`. Si no se encuentran resultados en la búsqueda, el endpoint devuelve un mensaje indicándolo.

NoGasStations Exception

En el caso de `NoGasStations`, se puede mostrar en dos ocasiones. Cuando no hay gasolineras dado una provincia y rótulo se mostraría un objeto

```
{
  "message": "No {rotulo} gas stations found in {provincia}"
}
```

O, en el caso de que no haya gasolineras dado un radio de búsqueda, una latitud y una longitud, se mostraría otro objeto de la misma forma

```
{
  "message": "No gas stations found within {kilometers}km from [{latitude}°,
   ↪  {longitude}°]"
}
```

NoDataFound Exception

Cuando buscamos la estancia media dada una provincia y un mes (o la media anual), puede darse el caso de que no haya datos sobre ello. No es que no se encuentre el recurso, simplemente que no hay información sobre ello. En este caso, se mostrará el objeto

```
{
  "message": "No data was found for [{provincia}, {mes/total}]"
}
```

A2ViviendasREST

Este microservicio se encarga de proporcionar los datos sobre las viviendas.

Requisitos considerados

- **Al menos dos operaciones de consulta o búsqueda parametrizada**: `GET /viviendas/{idCasa}`, `DELETE /viviendas/{idCasa}`
- **Una operación de consulta sobre las relaciones entre las entidades**: `GET /viviendas/{idCasa}/getBookingsAmount`

Endpoints REST disponibles

La siguiente lista es una especificación informal sobre los endpoints REST disponibles en el microservicio, a modo de resumen. La documentación completa se puede encontrar en el propio servidor, bajo las rutas `/docs` (SwaggerUI) y `/redoc` (Redoc). Además, se adjunta una copia local en el fichero `openapi.json`, dentro de la carpeta del proyecto.

- `POST /viviendas`: Crea una nueva vivienda. El cuerpo de la petición es un json con los siguientes campos:
  - `title`: Título de la vivienda
  - `description`: Descripción de la vivienda
  - `user_id`: Identificador del usuario que realiza la reserva
  - `location`: Lugar donde se encuentra la vivienda
  - `url_photo`: Fotos de la vivienda
  - `longitude`: Coordenada geográfica longitud de la vivienda
  - `latitude`: Coordenada geográfica latitud de la vivienda
- `GET /viviendas/{idCasa}`: Devuelve toda la información sobre la vivienda con identificador `idCasa`
- `PUT /viviendas/{idCasa}`: Modifica los campos de la vivienda con identificador `idCasa`
- `DELETE /viviendas/{idCasa}`: Borra la vivienda con identificador `idCasa`
- `GET /viviendas/{idCasa}/getBookingsAmount`: Devuelve la cantidad de reservas de una vivienda con identificador `idCasa`

Casos alternativos

En el caso de no encontrar una vivienda con el identificador proporcionado, se devolverá una excepción de tipo NotFoundError con su mensaje correspondiente dependiendo de la operación.

```
{
  "detail": "string"
}
```