2. Descripción API

Desarrollamos una RESTful API. A continuación describimos las restricciones del estilo arquitectónico REST utilizado.

1. Requests stateless

   Si se puede observar que los endpoint tienen como entrada únicamente parámetros en la query o body, sin necesidad de estado anterior o exponiendo estado. Probando que son stateless

2. Sustantivos, nunca verbos

   Para manipular nuestros enpoints se utilzan sustantivos y no verbos. Por ejemplo para maniuplar las sesiones se utliza el nombre sesions.

3. Plural ante singular

   Para facilitar la interpretación de nuestras API se nombraron los recursos en plural.

4. Intuitiva y simple

   Para cumplir con este punto se definieron los recursos lo más intuitivos posibles. De todas formas se necesita documentación extra para saber como instanciar los headers cuando los mismos son requeridos.

5. Utilizar el ? para ocultar la complejidad

   Para esto, la única request que necesita parámetros son colocados luego del ? del endpoint como queryparams. Únicamente fue necesario para hacer el filtrado de Houses en base a un id de TouristPoint y con fechas determinadas

6. Manejo de errores

   Para cumplir con esto, se creo ExceptionFilter donde en base a la excepciones que tienen las request, imprime cierto status (200, 400, 500)

Descripción del mecanismo de autenticación

Para la autenticacion, se crean usuarios que al realizar el login se les asigna un token. Luego al tener este token, se puede acceder a las request que lo necesitan. Esta clase es AuthorizationFilter, siendo llamada antes de realizar la request. Esto impide

que cuando una autorizacion es necesaria, rechaze el request antes de ejecutarlo si es que no tiene permisos para el mismo.

En nuestro caso, las unicas que no tienen permisos son las request que muestran informacion con la excepcion del logueo quien tampoco pide un token .

Descripción general de códigos de error

Los errores que decidimos mostrar son : 400, 401, 500.

1. 400 : Este error es para cuando en el sistema tuvo algún problema realizando la request
2. 401: Este error es mostrado cuando la request necesita token o el token que se tiene es invalido
3. 500: Este error es devuelto cuando hay problemas en el servidor

Descripción general de códigos de respuesta aceptada

1. 200 : Este código se lanza cuando la request tuvo éxito

Descripción de los resources de la API

URL base

1. Clase Booking tiene URL base api/bookings
2. Clase Category tiene URL base api/categories
3. Clase House tiene URL base api/houses
4. Clase Person tiene URL base api/persons
5. Clase Region tiene URL base api/regions
6. Clase TouristPoint tiene URL base api/touristpoints
7. Clase Session tiene URL base api/sessions

Descripción de cada endpoint

1. api/bookings
    - Resource : Booking
    - Description : Se busca, agrega o elimina la entidad Booking de mi sistema en la base de datos

- Endpoints (Verbo + URI) : Aparte de la base, puede ser que al buscar, editar o elminar cierto id , queda la siguiente URI : api/bookings/id
- Parameters : Para los parametros de la request se puede tener el id que va en la URI y/o el modelo de entrada BookingModel.
- Responses : Para la respuesta tenemos dos modelos, el básico o el detalle. La idea es plasmar únicamente el detalle cuando quiero buscar un Booking específico. Para el resto de las request siempre se devuelve el básico . Se pueden ver los parámetros de salida en BookingBasicModel y BookingDetailModel. En caso que haya algún problema realizado alguna acción de la lógica de la clase, se retorna error 400 con una breve descripción de la razón.
- Headers : Para los headers, el unico que es necesario es el de autorizacion que tiene que tener un token valido. Este parametro es solicitado para las request Post, Put y Delete

2. api/categories

   - Resource : Category
   - Description : Se busca, agrega o elimina la entidad Category de mi sistema en la base de datos
   - Endpoints:
     - GET /api/bookings
     - GET /api/bookings/id
     - PUT /api/bookings
     - POST /api/bookings
     - DELETE /api/bookings
     - DELETE /api/bookings/id

     Aparte de la base, puede ser que al buscar, editar o elminar cierto id , queda la siguiente URI : api/categories/id
   - Parameters : Para los parametros de la request se puede tener el id que va en la URI y/o el modelo de entrada CategoryModel.
   - Responses : Para la respuesta tenemos dos modelos, el básico o el detalle. La idea es plasmar únicamente el detalle cuando quiero buscar un Category específico. Para el resto de las request siempre se devuelve el básico . Se pueden ver los parámetros de salida en CategoryBasicModel y CategoryDetailModel. En caso que haya algún problema realizado alguna acción de la lógica de la clase, se retorna error 400 con una breve descripción de la razón.
   - Headers : Para los headers, el unico que es necesario es el de autorización que tiene que tener un token valido. Este parametro es solicitado para las request Post, Put y Delete

3. api/houses

- Resource : House
- Description : Se busca, agrega o elimina la entidad House de mi sistema en la base de datos
- Endpoints :
  - Obtener todas las casas GET /api/houses
  - Obtener una region con cierto id GET /api/houses/id
  - Actualizar una region PUT /api/houses
  - Agregar una region POST /api/houses
  - Eliminar todas las casas DELETE /api/houses
  - Eliminar region con cierto id DELETE /api/houses/id
- Parameters : Para los parametros de la request se puede tener el id que va en la URI y/o el modelo de entrada HouseModel. En el caso del filtrado por TouristPoint y fechas, en la URI se pide HouseSearchModel con la información necesaria para realizar la request
- Responses : Para la respuesta tenemos dos modelos, el básico o el detalle. La idea es plasmar únicamente el detalle cuando quiero buscar un House específico. Para el resto de las request siempre se devuelve el básico . Se pueden ver los parámetros de salida en HouseBasicModel y HouseDetail-Model. En caso que haya algún problema realizado alguna acción de la lógica de la clase, se retorna error 400 con una breve descripción de la razón.
- Headers : Para los headers, el unico que es necesario es el de autorizacion que tiene que tener un token valido. Este parametro es solicitado para las request Post, Put y Delete

4. api/persons

- Resource : Person
- Description : Se busca, agrega o elimina la entidad Person de mi sistema en la base de datos
- Endpoints :
  - Obtener todas las personas GET /api/persons
  - Obtener una region con cierto id GET /api/persons/id
  - Actualizar una region PUT /api/persons
  - Agregar una region POST /api/persons
  - Eliminar todas las personas DELETE /api/persons
  - Eliminar region con cierto id DELETE /api/persons/id
- Parameters : Para los parametros de la request se puede tener el id que va en la URI y/o el modelo de entrada PersonModel.
- Responses : Para la respuesta tenemos dos modelos unicamente básico. Se pueden ver los parámetros de salida en PersonBasicModel. En caso que haya algún problema realizado alguna acción de la lógica de la clase, se retorna error 400 con una breve descripción de la razón.

- Headers : Para los headers, el unico que es necesario es el de autorizacion que tiene que tener un token valido. Este parametro es solicitado para las request Post, Put y Delete

5. api/regions

   - Resource : Region
   - Description : Se busca, agrega o elimina la entidad Region de mi sistema en la base de datos
   - Endpoints :
     - Obtener todas las regiones GET /api/regions
     - Obtener una region con cierto id GET /api/regions/id
     - Actualizar una region PUT /api/regions
     - Agregar una region POST /api/regions
     - Eliminar todas las regiones DELETE /api/regions
     - Eliminar region con cierto id DELETE /api/regions/id
   - Parameters : Para los parametros de la request se puede tener el id que va en la URI y/o el modelo de entrada RegionModel.
   - Responses : Para la respuesta tenemos dos modelos, el básico o el detalle. La idea es plasmar únicamente el detalle cuando quiero buscar un Region específico. Para el resto de las request siempre se devuelve el básico . Se pueden ver los parámetros de salida en RegionBasicModel y RegionDetailModel. En caso que haya algún problema realizado alguna acción de la lógica de la clase, se retorna error 400 con una breve descripción de la razón.
   - Headers : Para los headers, el unico que es necesario es el de autorizacion que tiene que tener un token valido. Este parametro es solicitado para las request Post, Put y Delete

6. api/sessions

   - Resource: Person
   - Description : Se busca, postear el ususario que realiza sesion para obtener un token
   - Endpoints :
     - Agregar usuario en Session para obtener token POST /api/bookings
   - Parameters : No tiene parametros
   - Responses : La respuesta puede ser 200 si fue agregado o modificado el token, si hay algun problema en el proceso dira 400
   - Headers : No tiene

Swagger

Además de esta documentación, se encuentra la de swagger. Al compilar el paquete WebApi, se puede ver en el https://localhost:5001/index.html tiene el swagger de nuestra API

Figura 2.1:

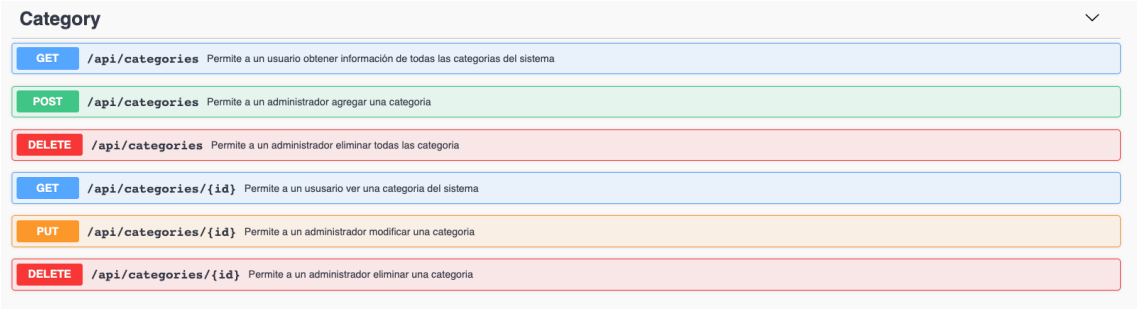

Figura 2.2:

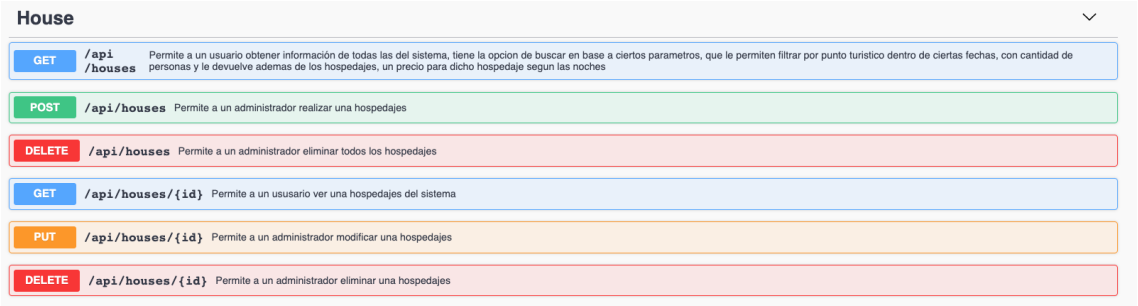

Figura 2.3:

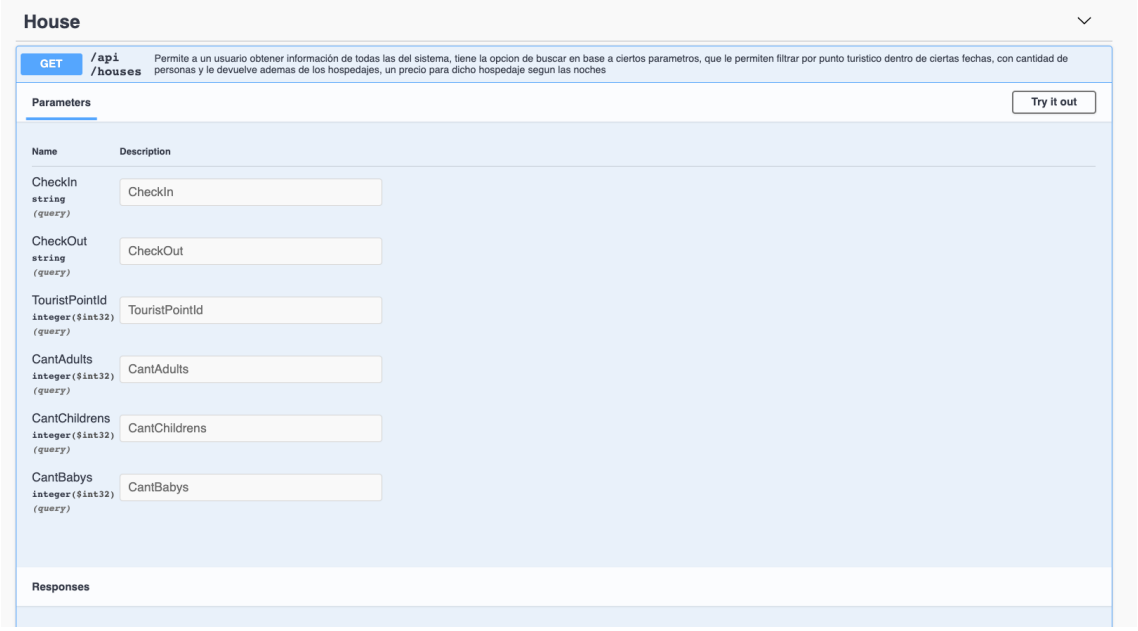

Figura 2.4:

Figura 2.5:

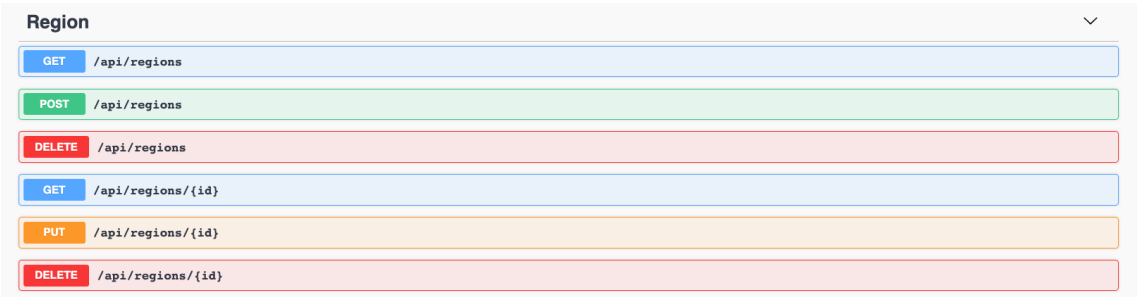

Figura 2.6:

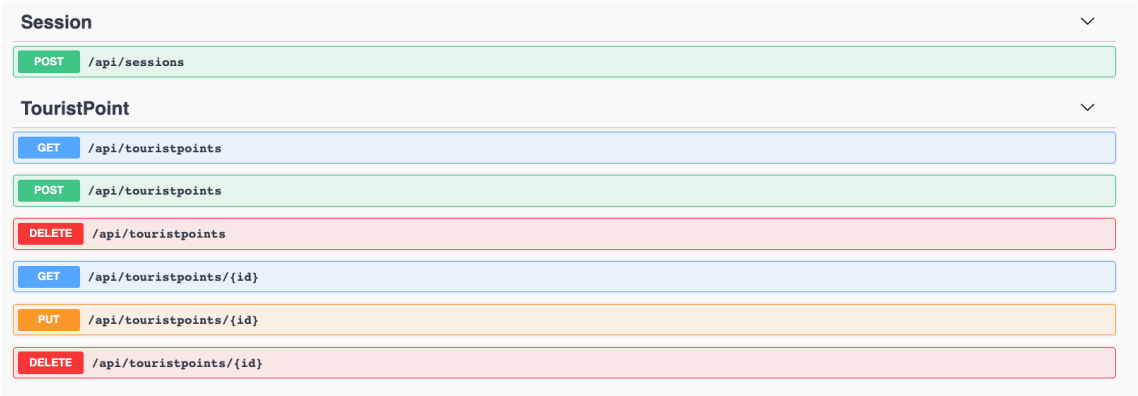
Session
POST /api/sessions
TouristPoint
GET /api/touristpoints
POST /api/touristpoints
DELETE /api/touristpoints
GET /api/touristpoints/{id}
PUT /api/touristpoints/{id}
DELETE /api/touristpoints/{id}

Figura 2.7: