

Spring Data Rest

Ruta API utilizando HAL Explorer: <http://localhost:8080/api>

PAGINACIÓN

Se hace uso de la paginación de forma nativa con Spring Data Rest. Permite obtener por defecto el JSON paginado con un cierto número de elementos ordenados. A su vez, se permite la navegación automática, ofreciendo en cada JSON los distintos links para viajar entre páginas.

- 1) Por defecto se muestra 20 elementos por página.
- 2) De forma manual se puede página del siguiente modo:
 - <http://localhost:8080/api/pilotos?page=0&size=2&sort=nombre.asc&projection=pilotoBasico>
 - Dónde:
 - page: número de página [comienza en 0]
 - size: número de elementos por página
 - sort: se debe indicar el campo y el orden ['campo',{'asc' o 'desc'}]
 - projection: proyección deseada, más abajo se especifican las proyecciones disponibles.
 - También se puede hacer de forma manual con un formulario desde el propio HAL Explorer. Por ejemplo, para formar el link anterior, rellenaríamos lo siguiente:

Template Request Input

URI Template

http://localhost:8080/api/pilotos{?page,size,sort,projection}

Parameters

page

0

size

2

sort

asc

projection

pilotoBasico


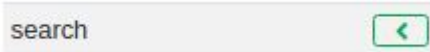
Expanded URI

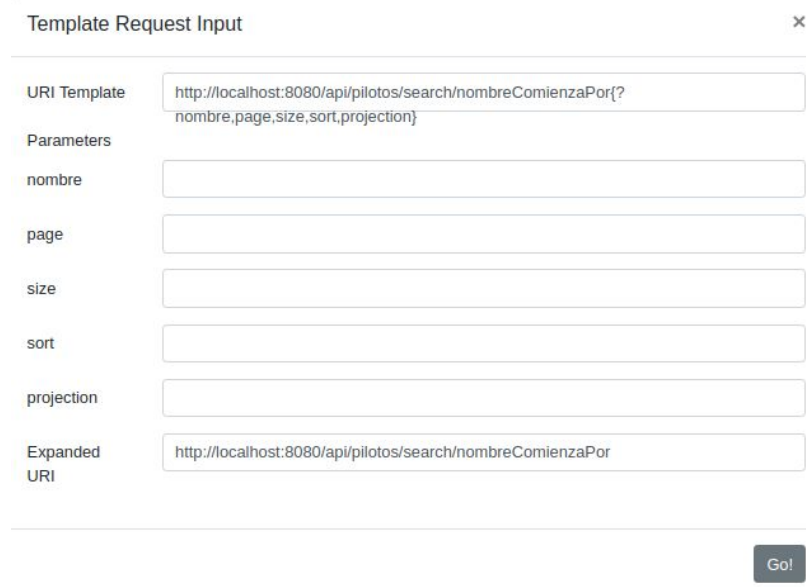
http://localhost:8080/api/pilotos?page=0&size=2&sort=asc&projection=pilotoBasico

Go!

BÚSQUEDA

Al igual que en la página la búsqueda se hace de forma nativa con Spring Data Rest. De forma gráfica se puede hacer desde HAL Explorer del siguiente modo, por ejemplo:

- Obtenemos los pilotos y una vez dentro: 
- Click en 'search': 
- Una vez hecho click nos aparecerán todas las posibles búsquedas, un ejemplo se click en 'nombreComienzaPor':



Template Request Input

URI Template:

Parameters:

- nombre:
- page:
- size:
- sort:
- projection:

Expanded URI:

Go!

De forma manual, como ejemplo se puede utilizar lo siguiente:

- <http://localhost:8080/api/pilotos/search/nombreComienzaPor?nombre=D>

A su vez estas búsquedas se puede paginar como en el anterior apartado, por ejemplo:

- <http://localhost:8080/api/pilotos/search/nombreComienzaPor?nombre=D&page=1&size=2>

RELACIONES

Los pilotos tienen:

- Un solo sponsor que puede ser compartido con otros pilotos.
- Cero o varios coches que no son compartidos con otros pilotos (Bidirecc.)
- Cero o varias residencias que no son compartidas con otros pilotos (Bidirecc.)

Un coche tiene un solo piloto (Bidireccional).

Una residencia tiene un solo piloto (Bidireccional).

Un sponsor puede ser compartido con varios pilotos (No bidireccional).

MÉTODOS EN LAS CLASES

En la clase Piloto se han utilizado varios métodos, para proporcionar datos de otras clases a las que se asocia. Estos métodos permiten obtener el número de coches, el número de residencias, el número del valor total de los coches y el valor medio de todos los vehículos de cada piloto.

USO

En su totalidad la aplicación se puede usar desde la siguiente ruta con HAL Explorer: <http://localhost:8080/api>

En dicho enlace, se documenta así misma las posibilidades: búsqueda, paginación, navegación entre asociaciones, etc..

Ya que no se reflejan las posibles proyecciones, las documentare manualmente:

Clases y sus proyecciones

- **Pilotos:**
 - <http://localhost:8080/api/pilotos> [Utiliza proyección 'PilotoBasico']
 - Otras proyecciones:
 - <http://localhost:8080/api/pilotos?projection=pilotoValor>
 - <http://localhost:8080/api/pilotos?projection=pilotoSponsor>
- **Residencias:**
 - <http://localhost:8080/api/residencias> [Utiliza proyección 'ResidenciaBasico']
- **Coches:**
 - <http://localhost:8080/api/coches> [Utiliza proyección 'CocheBásico']
- **Sponsors:**
 - <http://localhost:8080/api/sponsors> [No hace uso de ninguna proyección]

DESPLIEGUE EN DOCKER

Se ha simplificado la instalación para que solo con una línea de comando se pueda ejecutar la aplicación junto a la base de datos PostgreSQL. En primer lugar desde el 'application.properties' habrá que descomentar las 3 líneas indicadas para la ejecución desde Docker. Por último, hay ubicarse en la ruta del proyecto y desde la terminal ejecutar: **docker-compose up -d**

Una vez, realizado se podrá entrar a la api desde la anterior URL: <http://localhost:8080/api>

DESPLIEGUE EN HEROKU

En este último apartado se realizará el despliegue en Heroku, para ello, tras la instalación de Heroku CLI y la asignación de las SSH keys, se procederá a registrarnos/loguearnos en Heroku, donde tras subir el proyecto como repositorio se alojará en la nube:

<https://api-rest-sebas.herokuapp.com/api>

A través de este enlace se podrá realizar todo lo que anteriormente se hizo desde localhost, con la diferencia que en este caso se utilizará la base de datos PostgreSQL, y no H2.