

Documentación

# API Star Wars

Juan Mera Menéndez

## Índice

Documentación de uso .....	1
Introducción.....	1
End-points.....	1
Documentación de implementación y revisión .....	3
Integración con API existente de star wars .....	3
Persistencia .....	4
Diagrama E-R .....	4
Ejecución.....	4

# Documentación de uso

## Introducción

Esta API facilita la gestión de misiones permitiendo realizar operaciones de creación de nuevas misiones, el listado de misiones o la recomendación de estas. No requiere autenticación de ningún tipo. Se comunica con otro sistema mediante otra API para obtener los datos de naves, personas y planetas. Como formato de respuesta emplea json.

## End-points

El sistema proporciona los siguientes end-points:

- <http://localhost:8090/missions?page=1&size=3> - GET  
Para obtener un listado de misiones. También se puede añadir el parámetro “search” para buscar por nombre de los capitanes participantes en la misión:  
<http://localhost:8090/missions?page=1&size=3&search=vader>

La respuesta de una página de misiones de tamaño 3 es la siguiente:

```
{
  "current": "http://localhost:8090/missions?null",
  "results": [
    {
      "id": "1",
      "initialDate": "2022-12-20T14:04:26.068529",
      "endDate": "2022-12-21T20:04:26.068529",
      "starship": {
        "id": "3",
        "name": "Sentinel-class landing craft",
        "crew": "5",
        "passengers": "75"
      },
      "captains": [
        {
          "id": "3",
          "name": "R2-D2"
        }
      ],
      "planets": [
        {
          "id": "3",
          "name": "Yavin IV",
          "diameterKM": "10200"
        }
      ],
      "crew": "25",
      "durationHours": "30",
      "reward": "10000.0"
    },
    ...
  ]
}
```

- <http://localhost:8090/missions> - POST  
Enviando como cuerpo de la petición un json con fecha en formato ISO y los siguientes campos:

```
{
  "initialDate": "2022-12-20T12:12:00",
  "starship_id": 1,
  "people_ids": [1],
  "planet_ids": [1],
  "crew": 170,
  "reward": 1000
}
```

Y obteniendo el siguiente como respuesta:

```
{
  "id": "7",
  "initialDate": "2022-12-20T12:12",
  "endDate": "2022-12-20T18:12",
  "starship": {
    "id": "1",
    "name": "CR90 corvette",
    "crew": "30-165",
    "passengers": "600"
  },
  "captains": [
    {
      "id": "1",
      "name": "Luke Skywalker"
    }
  ],
  "planets": [
    {
      "id": "1",
      "name": "Tatooine",
      "diameterKM": "10465"
    }
  ],
  "crew": "170",
  "durationHours": "6",
  "reward": "1000"
}
```

- <http://localhost:8090/missions/{id}>  
Para obtener de vuelta la misión que tenga ese identificador.
- <http://localhost:8090/missions/next?criteria=reward>  
El sistema nos recomienda la siguiente mejor misión pendiente en función de dos criterios configurables con el parámetro “criteria”:
  - “reward”: la recompensa obtenida al completar la misión
  - “rewardPerHour”: la recompensa en relación con la duración de la misión

En sucesivas llamadas, el sistema nos va recomendado la siguiente misión según el criterio. Una vez recomendada una, la saca del subsistema recomendador y no la tendrá en cuenta para la siguiente llamada. Una salida para criterio "reward":

```
{
  "id": "3",
  "initialDate": "2022-12-20T14:04:26.104528",
  "endDate": "2022-12-20T15:04:26.104528",
  "starship": {
    "id": "4",
    "name": "Death Star",
    "crew": "342,953",
    "passengers": "843,342"
  },
  "captains": [
    {
      "id": "4",
      "name": "Darth Vader"
    }
  ],
  "planets": [],
  "crew": "342960",
  "durationHours": "1",
  "reward": "20000.0"
}
```

## Documentación de implementación y revisión

### Integración con API existente de star wars

Los datos de personas, naves y planetas se cargan y se almacenan en la base de datos al comienzo de la ejecución. Por este motivo, la aplicación se demora un poco en poder comenzar a recibir peticiones.

Para facilitar la revisión, una vez realizado el procedimiento de descargar y almacenar los datos una vez, realizando los siguientes cambios obviamos esos tediosos pasos en siguientes ejecuciones:

1. Cambiar en el fichero application.properties el valor de la línea seleccionada de "create" a "validate"

```
server.port=8090
spring.datasource.url=jdbc:hsqldb:hsqldb://localhost:9001
spring.datasource.username=SA
spring.datasource.password=
spring.datasource.driver-class-name=org.hsqldb.jdbcDriver
spring.jpa.hibernate.ddl-auto=create
```

2. Comentando en la clase “StarwarsAPIApplication” el contenido del método “postconstruct”:

```
@PostConstruct
public void postconstruct() {
    try {
        /* starshipService.loadData();
        peopleService.loadData();
        planetService.loadData();
        missionService.loadTestData(); */
    } catch (Exception e) {
        LoggerFactory.getLogger(Log.class).error("ERROR: loading the data - " + e.getMessage());
    }
    LoggerFactory.getLogger(Log.class).info("ALL DATA LOADED");
}
```

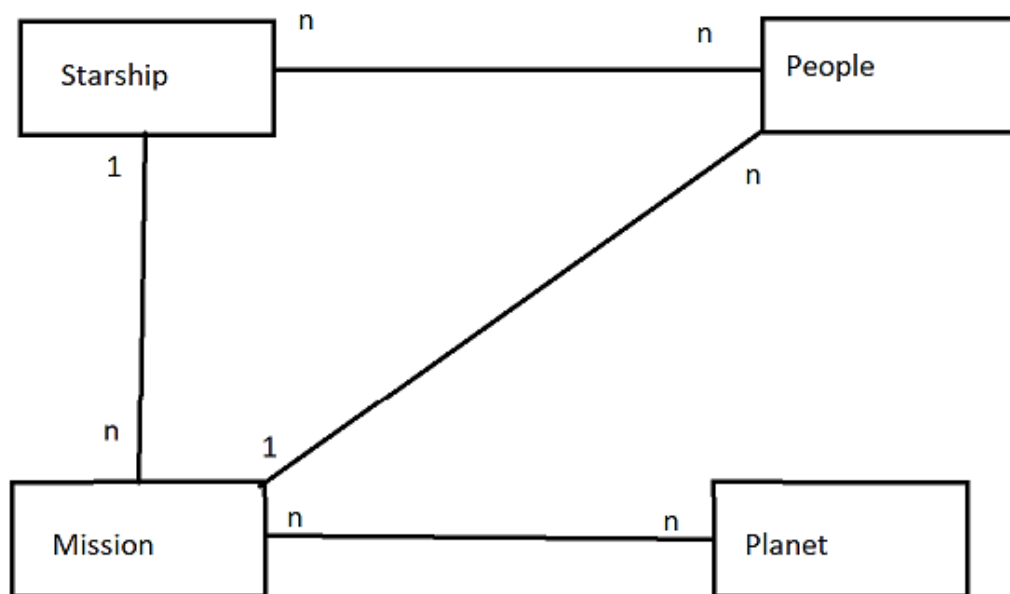
### Persistencia

El mecanismo de persistencia escogido es una base de datos local HSQLDB. Por la simplicidad y agilidad que aporta al desarrollo. También por la posibilidad de enviar dicha base de datos vacía y no tener que disponer de ningún proveedor de almacenamiento en la nube con su correspondiente coste.

También se opta por JPA como api de persistencia para el acceso a los datos.





La base de datos subida junto con el código al repositorio está vacía. Junto con la carga de los datos del api de Star Wars se crean algunas misiones de prueba.

### Diagrama E-R



### Ejecución

Para la ejecución de la aplicación, bastara con ejecutar la base de datos como servidor local. Para ello ejecutamos el fichero seleccionado de la carpeta bin.

Nombre	Fecha de modificación	Tipo	Tamaño
 runManager.bat	20/02/2022 17:51	Archivo por lotes ...	1 KB
 runManagerSwing.bat	20/02/2022 17:51	Archivo por lotes ...	1 KB
 runServer.bat	20/02/2022 17:51	Archivo por lotes ...	1 KB
 runWebServer.bat	20/02/2022 17:51	Archivo por lotes ...	1 KB

Con la base de datos corriendo, podemos ver su contenido con el fichero runManagerSwing de la misma carpeta que el anterior. Este nos proporciona una interfaz grafica sencilla con la que ver las tablas y poder ejecutar consultas SQL.

Después de ejecutar la base de datos, podemos ejecutar la aplicación con normalidad. La base de datos subida junto con el código al repositorio está vacía. Junto con la carga de los datos del api de Star Wars se crean algunas misiones de prueba.

El enlace al repositorio es: <https://github.com/juanmera01/starwars-API/tree/master> y el contenido se encuentra en la rama master.