****

**Prueba Técnica**

**Documentación de API**

**Daniel de la Torre Miguélez**

**13/12/2021**

1. **Decisiones sobre el desarrollo del API.**

En primer lugar, tras investigar el API que utilizaremos como fuente de datos (https://swapi.dev/) se observa que existen desarrollos para la conexión con esta API en diferentes lenguajes.

Para nuestro caso, ya que vamos a desarrollarlo en Spring Boot (Java), tomamos el desarrollo en Java de Maarten Dekker para realizar las llamadas que recuperen los datos de Swapi.

(Maarten Dekker API: <https://github.com/maartendekker1998>)

Se ha tomado como método de persistencia para las misiones la base de datos MySQL. Instalada y gestionada a través de la herramienta XAMPP.

Se adjunta, dentro del directorio “Doc” en GitHub, un fichero de exportación de la base de datos generado desde phpMyAdmin con la información necesaria para poder replicar el esquema y la tabla, así como los datos existentes al momento de realizar esta documentación.

(Fichero: pruebamisiones.sql)

Partiendo del enunciado propuesto, las operaciones solicitadas se han desarrollado asumiendo los siguientes puntos:

* Se incluye un identificador único para las misiones. Esto se representa mediante un valor auto numérico en la base de datos. Este identificador se recupera como parte del objeto de salida en las operaciones de listado.
  + Ej identificador: id=3 🡪 <https://swapi.dev/api/starships/3/>
  + Ej id multiple: <http://127.0.0.1:8080/api/ListaMisiones/7,5>
* En las operaciones que incluyen los parámetros de capitanes, planetas o nave, estos vendrán informados con los id que les identifican en swapi.
* En el caso de parámetros que puedan ser múltiples como los capitanes o los planetas estos vendrán informados como id separados por comas.
* Para la fecha de inicio pasada por parámetro se ha tomado el formato siguiente: yyyy-mm-dd hh:mi:ss (Ya que la precisión es de horas los valores de minutos y segundos se pasarían a 0)

Dado esto, las operaciones han quedado implementadas de la siguiente forma:

* Lista de Misiones - <http://localhost:8080/api/ListaMisiones>
* Lista de Misiones

[http://localhost:8080/api/ListaMisiones/{capitanes}](http://localhost:8080/api/ListaMisiones/%7bcapitanes%7d)

* Nueva Misión

[http://localhost:8080/api/NuevaMision/{fechaInicio}/{idNave}/{capitanes}/{numTripulacion}/{planetas}](http://localhost:8080/api/NuevaMision/%7bfechaInicio%7d/%7bidNave%7d/%7bcapitanes%7d/%7bnumTripulacion%7d/%7bplanetas%7d)

1. **Problemas encontrados.**

Para la elaboración de la documentación se ha intentado realizar mediante la herramienta open-api y swagger-ui. Se ha dejado implementado y es posible acceder a ella a través de la URL indicada a continuación, poniendo como ejemplo la utilizada en el servidor local.

<http://localhost:8080/swagger-ui.html>

La herramienta no ha ofrecido los resultados esperados y no se ha podido finalizar la configuración, ni ofrece toda la información. Por ello se ha optado por realizar este documento con la información necesaria sobre la API.

1. **Test**

Para las pruebas realizadas se ha utilizado la herramienta Postman. Se adjunta en el proyecto GitHub, en el directorio “Doc” un fichero JSON para poder importar la colección de request con los test lanzados. También se adjunta un fichero comprimido con las capturas de pantalla de las pruebas en la aplicación Postman realizadas en el servidor local.

Fichero Postman: ApiREST\_Test.postman\_collection.json

Capturas: PruebasPostman.zip

1. **Conclusiones.**

Aunque la implementación realizada cumple con las especificaciones solicitadas me hubiera gustado disponer de más tiempo para poder mejorarlo utilizando otros métodos o herramientas que hicieran más óptimo el código.