Space X API Project

How to run the program

You need Java 11 and maven to run this project.

The source code can be found at https://github.com/demirelif/SpaceXProject

To run the back-end

```
mvn clean install
mvn spring-boot:run
```

The spring boot will be started at localhost:8080

The source code can be found at https://github.com/demirelif/SpaceXUI

To run the front-end

```
npm install
npm start
```

The back-end checks the port the request comes from; therefore, the front-end will only be able to fetch the response from back-end by localhost:3000

Use Cases

This section explains REST-API end points.

1 - General Information about Space X

GET /company-info

This methods returns the basic company info from the API.
Front-end fetches the data from localhost:8080 and stores “summary” to display.

2 - Number of successful and unsuccessful launches

GET /no-of-successful-launches
GET /no-of-unsuccessful-launches

The number of successful launches and the number of unsuccessful launches are obtained from two different methods to increase reusability. For the number of successful

launches, the launch data from the API are processed and the launches with `success` parameter `true` are counted.

3 - Total number of people in crew

GET /total-people-in-space

This method returns the total number of people in the crew.

4 - Total orbit length of crew-dragons in years

GET /total-time

The dragons list is obtained from the API and the dragons which has a crew capacity bigger than zero are considered. The total of all `orbit_duration_yr` data from such dragons are returned.

5 - Average mass of all rockets

GET /average-mass-of-rockets

The rockets list is obtained from the API and `mass` parameter from all the rockets are added and divided by the total number of rockets to obtained the average mass.

6 - Total number of launches from desired launchpad

6a - The launchpad list

GET /launchpads

Launchpad information from the API is obtained and stored in an array-list of Launchpad objects. The names of launchpads are presented to the user in a dropdown menu format.

6b - The total number of launches

GET /total-number-of-launches-from-launchpad @PARAM id

When user selects a launchpad name, the launchpad id is forwarded the total number of launches from that launchpad is fetched with the selected id.

7 - Latest launch

GET /latest-launch

The latest launch is fetched from API via back-end and displayed.

8 - Next launch

GET /next-launch

The next launch is fetched from API via back-end and displayed.

Design Decisions and Structure

For the back-end I decided to use Spring-boot since I had a brief experience on it. I designed the project as a Maven project to decrease the complexity of the dependencies and make it easier to build and run the project.
I decided to make a user interface for the project to display the use cases in a better way. It also provides an alternative to send requests via Postman-like applications.
In this project, I did not use Repository and Service since I did not use any database.

Model

`Dragon, Launch, Launchpad` and `Crew` classes are created to make the obtained data easier to use. Also, `DragonType` enum is used to assign dragon type as `CREW` or `CARGO`.

Controller

`ProjectController` class includes REST-API end points for the project.

`RestController` class sends requests and gets the data from SpaceX API and does required calculations.

Expected View of The Front-end