**Exercise REST**

**Exercise 1: Refuelling**

Create a new project in your corresponding GitHub-folder (according to the instructions in the document on Canvas “How to create a project in IntelliJ) and add the following classes (in a package called model) according to the UML-diagram below:

Afbeelding met tafel

Automatisch gegenereerde beschrijving

Don’t forget to commit and push regularly!

The method getFuelConsumption() returns the consumption per 100 km. (For example: if you have driven 500 km since the last refuelling and you are now refuelling 30 liters, your consumption is 6 liters/100 km.)

Make unit tests for the Refuelling class with more than 70% coverage.

(Tests for the RefuellingService class have to be made in the next part)

Create a RestController with a RefuellingService-object as attribute.

Make sure that the refuellingList of this RefuellingService-object is filled with 3 Refuelling-objects at project start-up.

The assignment is that you will build the REST services for the backend while another team builds the frontend in React. You have agreed that next

REST services would be provided:

* Via **/api/** **refuellings** to retrieve all refuelling operations
* Via **/api/** **refuelling/{id}** to retrieve 1 refuelling on the basis of its id
* Via **/api/** **refuellings/add** to add a new refuelling-object to the ArrayList, since we are not yet working with a database you will have to make sure that this refuelling service is assigned a unique id and added to the stored ArrayList of refuelling services. This service will return the newly added refuelling-object. Make sure that the http-status is set to “created”.
* Via **/api/refuelling/update/{id}** to modify a refuelling with a given id in the ArrayList. This service returns the modified refuelling.
* Via **/api/refuelling/delete/{id}** to remove a refuelling with a certain id from the ArrayList. This service returns the number of refuelling-objects in the ArrayList.

Test alle these functionalities using the http-requests in the *refuellingREST.postman\_collection.json* (see Canvas). Create a Postman-testscript by adding tests to every request.

Notice: the JSON-output of a Refuelling-object includes the fuelConsumption as if it is a “normal attribute”:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

So, also the RestController reads the method “getFuelConsumption” as a derived attribute…

What we did so far was very similar to the example in the presentation. Can you now also provide the possibility of using **/api/refuellings/totalConsumption to** get the total consumption of all saved fuel sessions? For example, for these refuelling sessions:

Afbeelding met tafel

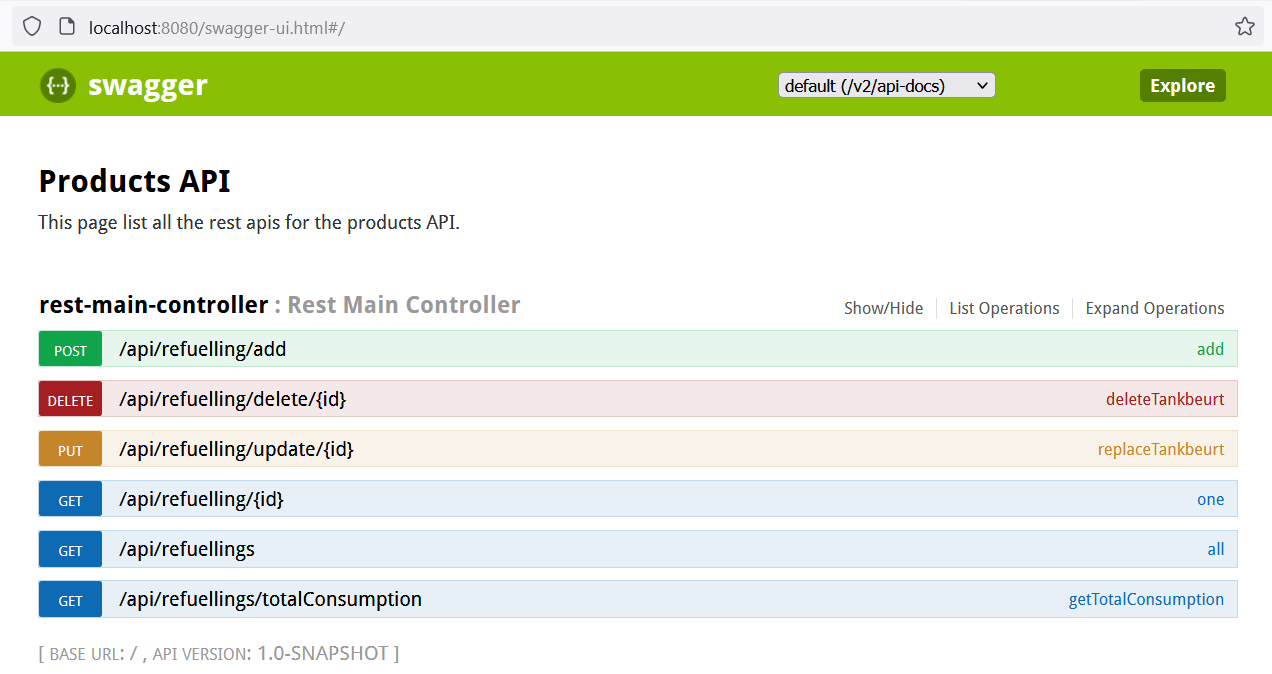
Automatisch gegenereerde beschrijving

Would that give the following output:

Afbeelding met tekst

Automatisch gegenereerde beschrijving

Now create documentation for your project using Swagger.



**Exercise 2: contact list**

<http://jsonplaceholder.typicode.com/> is a free fake API that you can use if you need data for testing & prototyping.

The task in this exercise is to build a Rest service that returns the result in just the same form as: [http://jsonplaceholder.typicode. com/users](http://jsonplaceholder.typicode.com/users)

Below you will find a copy of the first 3 users:

[

{

"id": 1,

"name": "Leanne Graham",

"username": "Bret",

"email": "Sincere@april.biz",

"address": {

"Street": "Kulas Light",

"Suite": "Apt. 556",

"city": "Gwenborough",

"zip code": "92998-3874",

"geo": {

"lat": "-37.3159",

"lng": "81.1496".

}

},

"phone": "1-770-736-8031 x56442",

"website": "hildegard.org",

"company": {

"name": "Romaguera-Crona",

"catchPhrase": "Multi-layered client-server neural-net",

"bs": "harness real-time e-markets".

}

},

{

"id": 2,

"name": "Ervin Howell",

"username": "Antonette",

"email": "Shanna@melissa.tv",

"address": {

"Street": "Victor Plains",

"Suite": "Suite 879",

"city": "Wisokyburgh",

"zip code": "90566-7771",

"geo": {

"lat": "-43.9509",

"lng": "-34.4618".

}

},

"phone": "010-692-6593 x09125",

"Website": "anastasia.net",

"company": {

"name": "Deckow-Crist",

"catchPhrase": "Proactive didactic contingency",

"bs": "synergize scalable supply-chains".

}

},

{

"id": 3,

"name": "Clementine Bauch",

"username": "Samantha",

"email": "Nathan@yesenia.net",

"address": {

"Street": "Douglas Extension",

"suite" means "Suite 847",

"city": "McKenziehaven",

"zip code": "59590-4157",

"geo": {

"lat": "-68.6102",

"lng": "-47.0653".

}

},

"phone": "1-463-123-4447",

"website': 'ramiro.info

"company": {

"name": "Romaguera-Jacobson",

"catchPhrase": "Face to face bifurcated interface",

"bs": "e-enable strategic applications".

}

}

]

Create 2 services in the rest application:

1. One service to add multiple users (simultaneously) to an ArrayList of users (which you store in your restcontroller)
2. One service to retrieve all saved users (in the same format)

Then use Postman to load the above list of users into the ArrayList of users and also test via Postman whether you get the saved users back when you retrieve them all.