# Language and tool selection

When preparing for this project, I decided to use java due to the fact it has a lot of documentation around api testing. I preferred to use the cucumber framework as a BDD tool. Cucumber is the most used tool for BDD. Finally I used Resst-Assured for api testing. Resst-Assured is one of the most popular api testing tools. Also I used junit jupiter api for assertions.

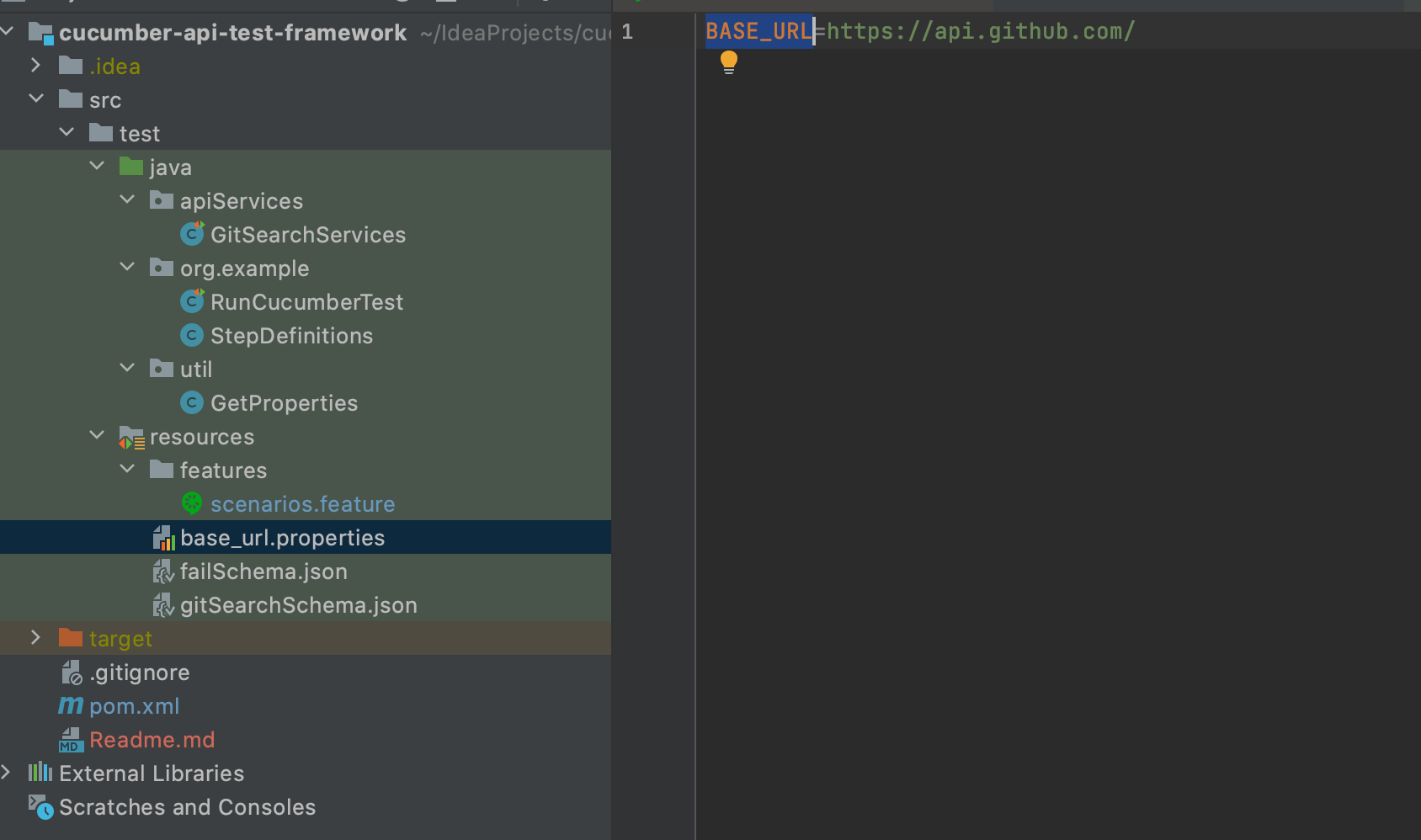
# Project Description

I prepared the project as parametric as possible this makes the project dynamic (more tests can be added in future) and very easy to manitan

## Base url, endpoint and parameters settings

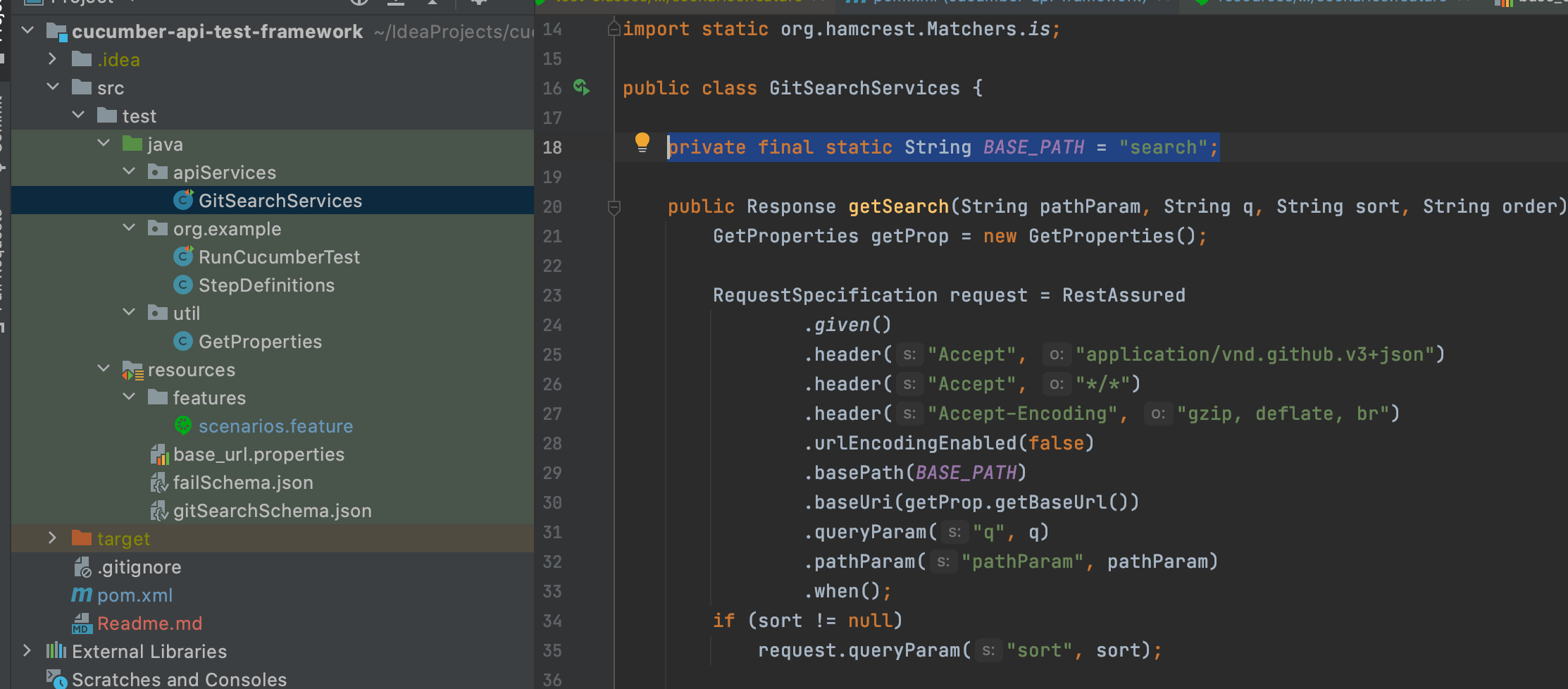
### Base Url

You can set the base url in the base\_url.properties file under test test resource.

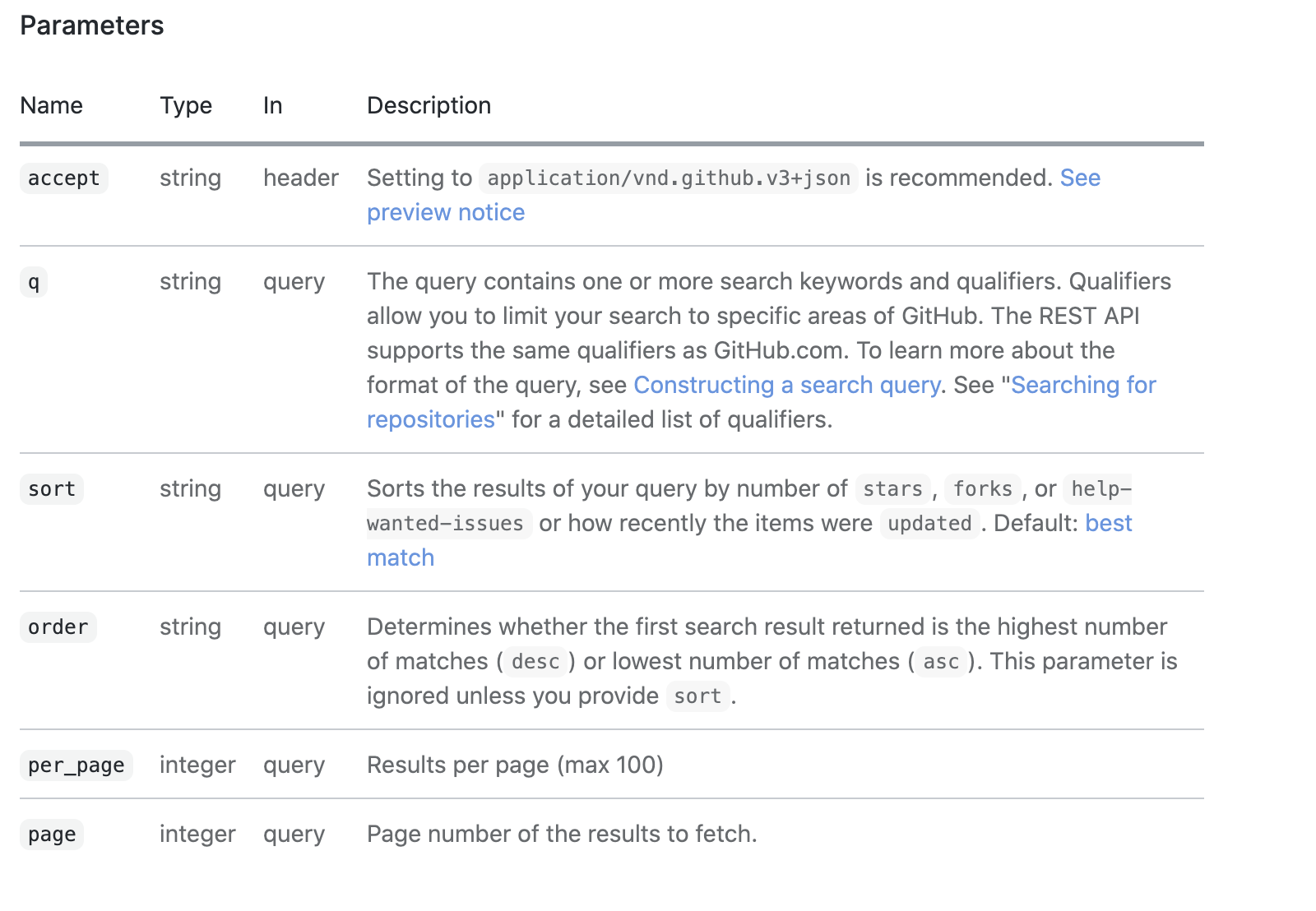


### End Point

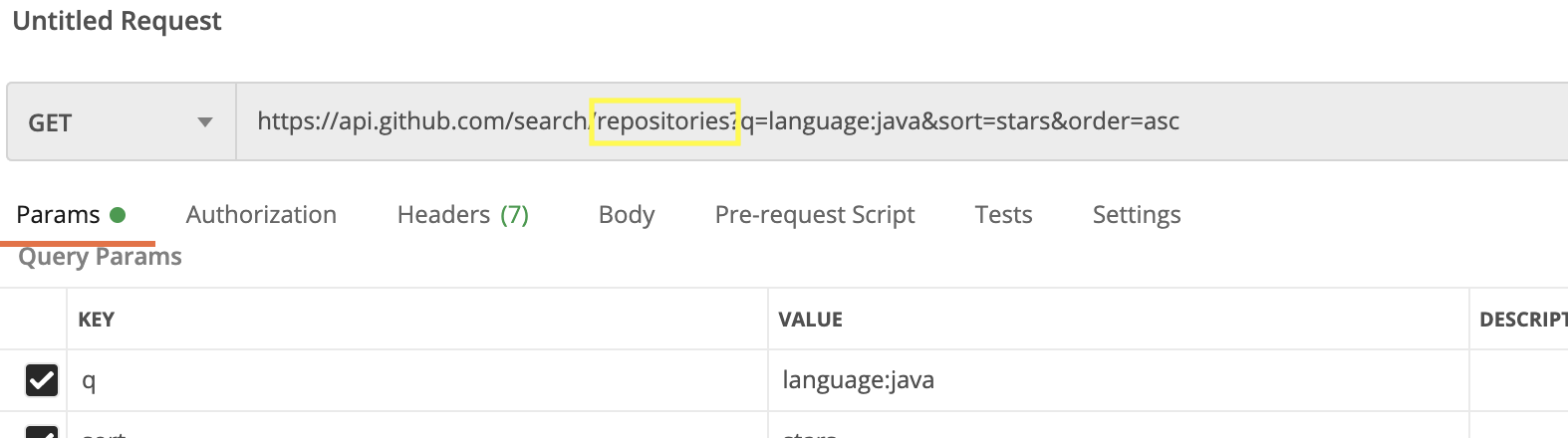
The endpoint is defined as a constant in the Search class. If you need to write tests for different api endpoints, you should define separate classes for them, just like the Page Object Model.



### Parameters

Github search api has the following parameters. The Github search api has the following parameters. I have only defined what is needed for the test. 

I also used the second part of the end point as a path parameter. In fact, in my opinion, it is already a parameter because you can change what you are searching for from that part, for example, if you want, you can set as issue or user to search for issue or user.



# 

# Assertions

In order to make the test scenarios more parametric, I preferred to verify with junit by pulling the parameter I want to verify from json instead of verifying with json body.While doing this, I used the "as" method of "Resst-Assured" to first convert the json body to the HashMap object and then use it as I wanted.

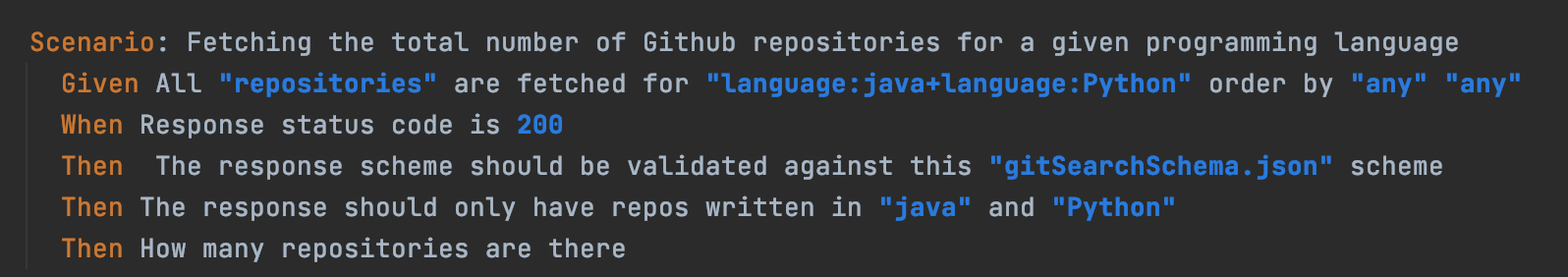


# Scenarios

Every request made in the scenarios has a validation. For example; When we request repositories in Java language, I confirm that there are only repositories written in java. The script structure is quite flexible. If we want, we can test the unsuccessful request scenario without writing a new code. Even for this I added an extra test scenario in the feature file.

#### The working logic of the scenarios is as follows;

First I send a request, then check if it has the requested status code. If it has the requested code, I check the json body in the response to see if the requested schema is also returned. I used the Rest-Assured JsonSchemaValidator class for schema check. Thanks to this control, we also have some kind of contraction test. If there is something that does not fit the schema in the request response, a change or a missing parameter, we will be able to find out immediately with this test. Afterwards, I check the other assertions



As a result, if you want to test other git search apis on [this link](https://docs.github.com/en/free-pro-team@latest/rest/reference/search) in the future, it can be done by changing the parameters in this scenario. If schema checking is desired, a json scheme for the response must be created. Normally this scheme should be requested from the developer, but if there is no such chance, support is available at https://jsonschema.net/ .

# 

# How tests are run and reporting

A computer with Maven and java will be sufficient to run the tests. All you have to do is go to the root directory of the project (the directory where the pom file is) through the terminal and run the following command.

To run the tests and **generate a report** type:

mvn clean test

**sample-rest-api-cucumber\target\cucumber-reports** to see the generated cucumber report

Look for **overview-features.html** and click on it

