Test cases selection

I have selected 10 test cases to automate taking into account the following reasons:

- Having at least one important test case for each feature
- Test not only the happy path but also some negative scenarios

Analysis of the api test cases execution

Report can be found in the current folder as behave_report.html.

8 of 10 of the test cases are passing, the 2 that are failing are because of the following issues:

- Issue 1: Test case "Get an order by an invalid id" is failing.
  - Expected result: According to the Swagger documentation, the response code should be 400 with the message "Invalid ID supplied".
  - Actual result: The code response is correct, but the message is: "Input error: couldn't convert `%` to type `class java.lang.Long`.
- Issue 2: Test case "Login with an invalid user and password" is failing.
  - Expected result: According to the Swagger documentation, for invalid user or password the response code should be 400 with the message "Invalid username/password supplied"
  - Actual result: The code response is 200 and there is retrieved a user session as it was a valid user.

In conclusion, for the 10 test cases executed, 2 issues were found. There were some other behaviors that caught my attention, but as they were not specified in the documentation, I haven't create the test cases or report the issues, but I think they are issues that should be managed from the BE. Eg:

- It is possible to create 2 pets with the same id.
- It is possible to create pets with just the id. They are created empty with just that data. I think that the name and the category should be also mandatory.

Analysis of the performance execution with Locust

Different executions were executed to test different traffic.

1. To start, the apis were tested with 50 users, with a spawn-rate of 5 users per second. This retrieves a successful result related to the apis performance (between 4.5 ms and 51ms average) but some endpoints retrieves exceptions:

    - PUT to /api/v3/pet/ - 500 (Internal Server Error) —> 7 times
    - POST to /api/v3/store/order - 500 (Internal Server Error) —> 1 time

    Report can be found in current folder as *report_50us_5.html*

2. In second, place, the traffic was increased gradually to 70 users, with a spawn-rate of 7 users per second. The results were very similar. The average time was good, but there were some failures:

    - PUT to /api/v3/pet/ - 500 (Internal Server Error) —> 6 times
    - GET to /api/v3/pet/5 - 500 (Internal Server Error) —> 1 time
    - GET /api/v3/store/order/70 - 500 (Internal Server Error) —> 1 time

    Report can be found in current folder as *report_70us_7.html*

3. Finally the apis were tested with 100 users, to see if the values were the same, or if the max capacity could be reached. The results changed significantly. The average time was good, but after some minutes a lot of apis started retrieving 500 constantly.

    - PUT to /api/v3/pet/ - 500 (Internal Server Error) —> 657 times
    - POST to /api/v3/pet/ - 500 (Internal Server Error) —> 658 times
    - GET to /api/v3/pet/150 - 500 (Internal Server Error) —> 628 times
    - GET to /api/v3/pet/5 - 500 (Internal Server Error) —> 663 times
    - GET /api/v3/store/order/70 - 500 (Internal Server Error) —> 271 times
    - POST /api/v3/store/order - 500 (Internal Server Error) —> 270 times
    - GET /api/v3/store/order/6 - 500 (Internal Server Error) —> 267 times

Conclusion: In a fast analysis, it can be seen that the response times are acceptable, however some apis are retrieving exception. Some considerations:

- At least the PUT to /api/v3/pet/ request should be reviewed since it is failing even with a lower traffic.
- With 100 users the system crash, so should be analyzed which is the max users that the app must accept, to see if there is needed an improvement on that side.