CHARTER

-----------------------------------------------

Identify capabilities and areas of potential instability of the “rest api todo list manager” for todos

Build

-----------------------------------------------

Java -jar runTOdoManagerRestAPI-1.5.5.jar

Area

-----------------------------------------------

Main functions and capabilities

Environment

-----------------------------------------------

MacOS Ventura 13.1

Screen resolution: 3024 × 1964

START

-----------------------------------------------

5:10 PM 09/23/2024

Other Data

-----------------------------------------------

TESTER

-----------------------------------------------

Andrei Sandor

[andrei.sandor@mail.mcgill.ca](mailto:andrei.sandor@mail.mcgill.ca)

260977451

#DURATION

45 minutes

\*\*\* The number inside parenthesis after the time is the reference of the script found in POSTMAN\_TODOS for that observation. The format of the name of the script in POSTMAN\_TODOS folder is

number inside parenthesis-MethodType Method

After this number is the API call that I tried to do\*\*\*

The format of the observation is like this.

Time (Reference POSTMAN\_TODO): **Method TYPE** Method. Behavior

5:10 PM (1): **GET** /todos. It contains all the initial/current todos. It works as expected.

5:12 PM (2): **HEAD** /todos. It does seem to work since the 201 value is get, but there is nothing shown.

5:14 PM (3): **POST** /todos. With a good format (containing the title, description, doneStatus), it works as expected, it creates a new task with a different id.

5:16 PM (4): **POST** /todos. With a missing title in the JSON, the app crashes.

5:18 PM (5): **POST** /todos. Without only doneStatus in the JSON file, it creates a new task with doneStatus set to false. The id somehow incremented by two with respect to the first successful post.

5:20 PM (6): **POST** /todos. Without only description in JSON, the description is set to “” and the todos is created nicely.

5:22 PM (7): **GET** /todos/1. It returns a specific instance of todo using id 1 . It works as expected.

5:24 PM (8): **GET** /todos/10. The id 10 doesn’t exist and the app crashes which makes sense.

5:26 PM (9): **HEAD** /todos/1. It returns nothing, but it seems to work.

5:28 PM (10): **HEAD** /todos/10. With a wrong id the app seems to crash which is expected.

5:30 PM (11): **POST** /todos/1. With an existent id (i.e 1) and a complete JSON with title, description, doneStatus, the task is amended which makes sense.

5:32 PM (12): **POST** /todos/10. With a non-existent id (i.e 10). The app crashes which is expected.

5:34 PM (13): **POST** /todos/1. With an existent id (i.e.1) and a JSON missing a title. It works as expected, it amends with the other fields (description, doneStatus).

5:36 PM (14): **POST** /todos/1 With an existent id (i.e.1) and a JSON missing a doneStatus. It works as expected, it amends with the other fields (title, description).

5:38 PM (15): **POST** /todos/1 With an existent id (i.e.1) and a JSON missing a description. It works as expected, it amends with the other field (title, doneStatus).

5:40 PM (16): **PUT** /todos/1 With an existent id (i.e 1) and a complete JSON. The task is amended which makes sense.

5: 42 PM (17): **PUT** /todos/10 With a non-existent id (i.e 10). The app crashes which makes sense.

5:43 PM (18): **PUT** /todos/1 With an existent id (i.e 1) and a missing title, it crashes. The task crashes which doesn’t make sense, POST doesn’t do that. This is a BUG.

5:44 PM (19): **PUT** /todos/1 With an existent id (i.e 1) and a missing doneStatus/description, it amends. It sets the field of doneStatus to False and the description to “”. **This is a bug/area of potential risk.**

5:45 PM (20): **DELETE** /todos/1 With an existent id (i.e) the todos is removed as expected.

5:46 PM (21) **DELETE** /todos/10 With a non existent id (i.e 10). There is an error as expected.

5:47 PM (22) **POST** /todos with a field that doesn’t exist, it crashes.

5:47 PM (23) **POST** /todos/1 with only a title amends only the title. It works as expected.

5:48 PM (24) **POST** /todos/1 with only a description amends only the title. It works as expected.

5:49 PM (25) **POST** /todos/1 with only a done status amends only the doneStatus.

5:50 PM: (26) **GET** /todos/1/tasksof. It works as expected. It returns the complete taskof of todo 1. The id is correct

5:50 PM: (27) **GET** /todos/10/tasksof. It doesn’t work as expected. With an invalid id (10), it should return an error since the id doesn’t exist, but it returns the id 1 todo. **This is a bug/area of potential risk.**

5:51 PM (28): **HEAD** /todos/1/tasksof. It passes with 201 by using a todo1 which is valid.

5:51 PM: (29) **HEAD** /todos/10/tasksof. It doesn’t work as expected. With an invalid id (10), it should return an error since the id doesn’t exist, but it returns the id 1 todo. **This is a bug/area of potential risk.**

5:52 PM (30) **POST** /todos/1/tasksof. It returns nicely the new taskof created for todos 1 which exists

5:53 PM (31) **POST** /todos/10/tasksof. It crashes since id 10 doesn’t exist. It works as expected.

5:54 PM (32) **DELETE** /todos/2/tasksof/1. It deletes nicely the taskof with id 1 of todos with id2 where both exist.

5:55 PM (33) **DELETE** /todos/2/tasksof/1. It crashes since id 1 doesn’t exist, works as expected

END TIME:

--------------------------------------------------

5:55 PM 09/23/2024

NEW TESTING IDEAS:

----------------------------------------------------

Try to work with a negative ID to see if it works.

Try to work with some input values that do not make sense.

Analyse more deeply headers.

SUMMARY OF SESSION FINDINGS

-----------------------------------------------

For GET, with correct id where needed, it returns correctly the todos; otherwise, with an invalid ID, it returns an error. For the /tasksof scenarios, it returns well the todos with a correct id, but an incorrect id, it still returns something even if it shouldn’t.

For HEAD, with correct id where needed, it doesn’t return some headers, but it seems to return a good HTTP value. A wrong id crashes the app except for /tasksof scenarios which returns a todo on a wrong id.

For POST, a title is needed for the todos/ while for /todos/id with a correct JSON, it is fine. If the id is correct, it won’t crash and if the id is incorrect it will crash, but we need a title. Also, the todos of newly created id can increment by one more.

For PUT, a title is needed for a todos/id. Sometimes it updates description/done Status as expected, while some cases, it relies on the default value.. A wrong id crashes the app.

For DELETE, with an existent id, it doesn’t crash (it deletes), with an nonexistent id, it does crash since there is no the respective todo.

LIST OF CONCERNS

---------------------------------------------------

ID increases by 1 for a POST request with an error. This concern can happen for all of the POST requests.

Doing a POST without title gives no error while the PUT without title gives an error.

There is no way to save the data. Closing the application deletes the memory.

This app runs on the terminal which is not an ideal mode of using it. It can close easier than other scenarios.

FILES REFERENCED OR CREATED DURING THE SESSION

-----------------------------------------------

runTodoManagerRestAPI-1.5.5.jar (API build file)

POSTMAN TODO (Screenshots of the script)