1. Scope of the Test Plan

Frontend (ReactJS)

- User Interface (UI): Verify that all visual components (such as buttons, forms, lists, etc.) are functional and styled correctly.
- User Authentication: Validate the login, registration, and credential verification flow.
- Dashboard: Ensure that the display of product and order data works correctly.
- Product and Order Lists: Verify that products and orders are loaded correctly from the backend, displayed correctly in the UI, and that actions (edit, delete, add) work as expected.
- Form Validations: Validate required fields and restrictions (such as email format, password length, etc.).
- UI Performance: Verify load times and overall performance.

Backend (C# API) with postman

- User Authentication: Verify login functionality and error handling for incorrect or missing credentials.
- Product Management: Ensure that RESTful APIs for creating, reading, updating, and deleting products (CRUD) are implemented correctly.
- Order Management: Verify that APIs for handling creating, updating, deleting, and viewing orders are working correctly.
- Error Handling and Validations: Ensure that the backend correctly handles invalid input, errors, and exceptions (e.g. missing or incorrect product data).
- User Authorization: Check that protected routes are properly restricted and that only authenticated users can access certain APIs.
- Integration and Communication: Verify that communication between the frontend and backend (via APIs) is seamless and data is properly synchronized.

2. Testing Process Objectives

The testing process will encompass both frontend and backend functionalities. The primary types of tests to be conducted include functional, integration, accessibility, validation, and API tests.

Validation

The following pages will be analyzed: Home, Login, Dashboard, Products and Orders page.

For these pages, manual test cases will be implemented to validate the content and functionality of the website as well as the integration with the API.

The designed tests will be automated using Cypress and Postman and swagger for the connection to the API during backend tests.

Backend

The following entities will be analyzed: User, Product, Order.

For each part, tests were designed to validate the functionality, defining

User: Login user

- User with correct password
- Incorrect user and password
- Both fields empty
- Incorrect user and correct password

Product: Add product, update product data, delete a product, show all products.

Orders: Create a new order, update an order, delete an order, show all orders.

Resources:

The resources required to carry out the execution of the Test Plan will be the following:

- Environment: Windows 10, internet access, project running locally (API and web app).

- Tools: Dotnet 8, Node, Java 17, Maven. VS Code, Postman.

- Data set: access to test data to include in Postman Runners (“Data set for Postman - JSON files” folder).

- Browsers: Chrome, Edge, Electron.

Classification of severity

According to its importance and margin of attention:

- Critical: the failure found is in an essential functionality, or prevents some process in its entirety.

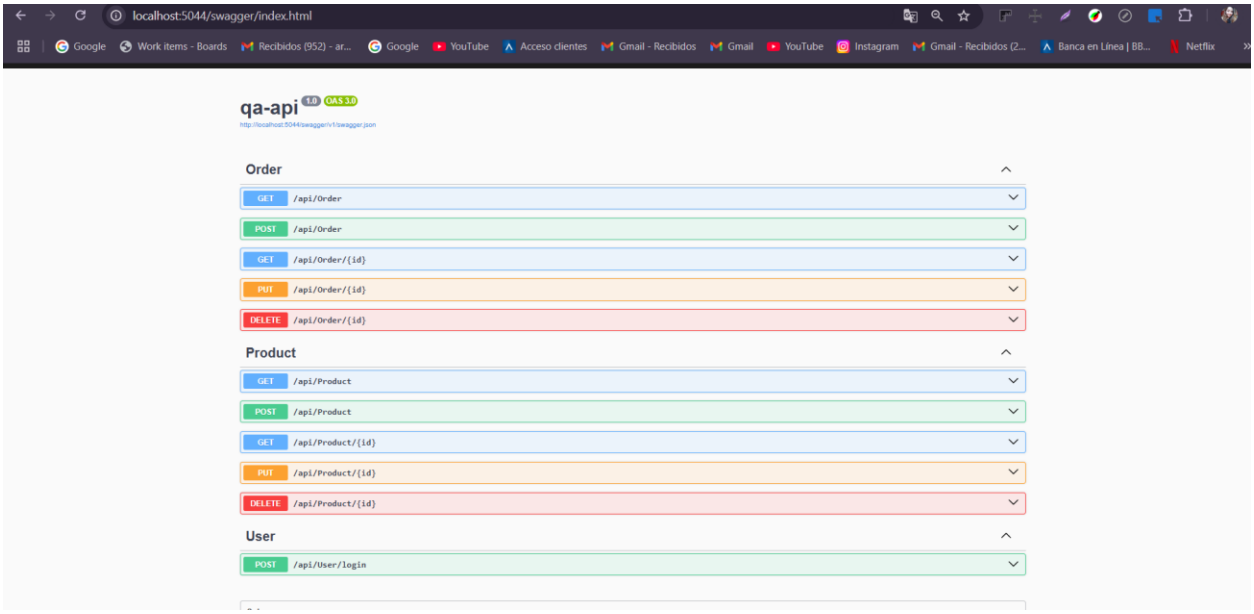
- High: the failure detected in a main process.

- Medium: the failure detected does not limit the operation of other processes, but is relevant.

- Low: the detected bug does not directly influence a functionality.

- Very low: the bug found does not influence a functionality, mainly visual errors.

Link repository



- Interoperability Verification: Validate that communication between the frontend (ReactJS) and the backend (C# API) is efficient and accurate.
- User Experience (UX) Validation: Verify that the user interface is intuitive, accessible, and free of visual glitches.
- Specific Objectives for the Frontend
- Verify that UI elements are interactive and functional.
- Verify that form validations (such as login and product creation) are implemented correctly.
- Ensure that navigation between different dashboard pages works correctly.
- Specific Objectives for the Backend
- Validate that the APIs correctly implement CRUD functionalities for products and orders.
- Ensure that authentication and authorization handling works correctly to prevent unauthorized access.

Manual testing test case

LOGIN

1. User and password correct

Response the app validates correctly and displays the token.

Reproduction steps:

1. Enter the user
2. Enter the password
3. Click on the login button

4. System responses

- Login
- Dashboard
- Products
- Orders

Login

testuser ••••••• Login

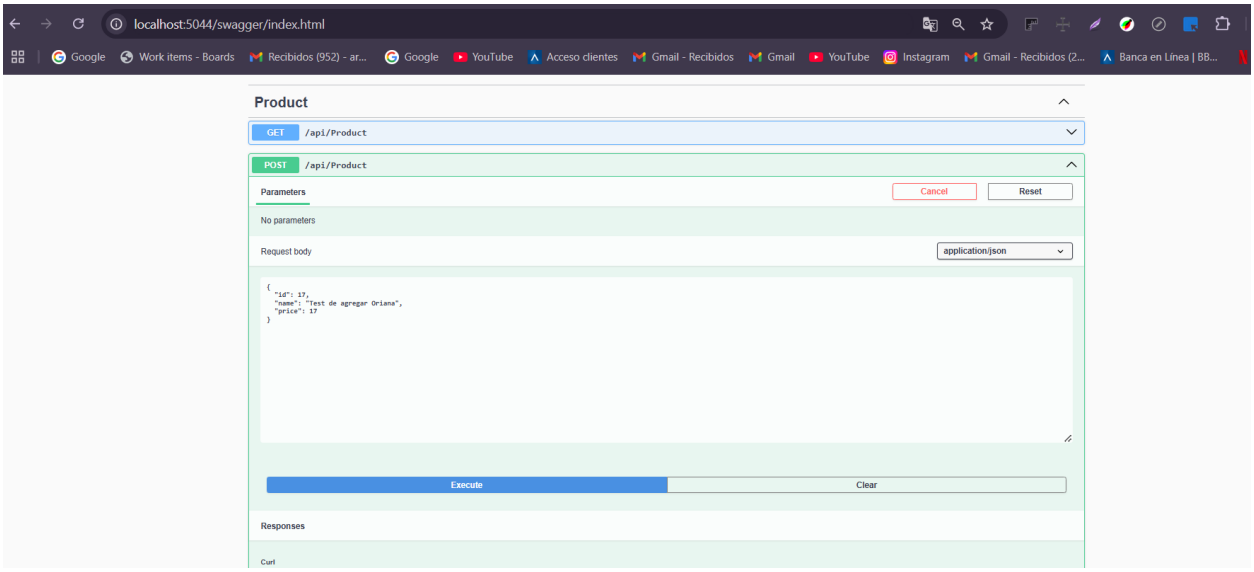
Logged in with token: sampletoken

When running the tests with POSTMAN Products an array appears with the data entered manually, but initially the application had empty values, it means that I have passed this data through the backend, data has been added, modified and deleted.

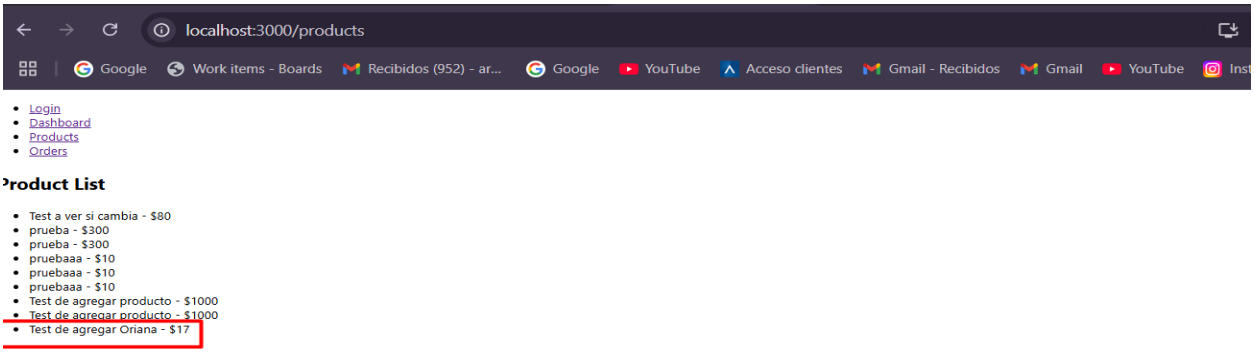
Product

Add PRODUCT FOR API WITH METHOD POST

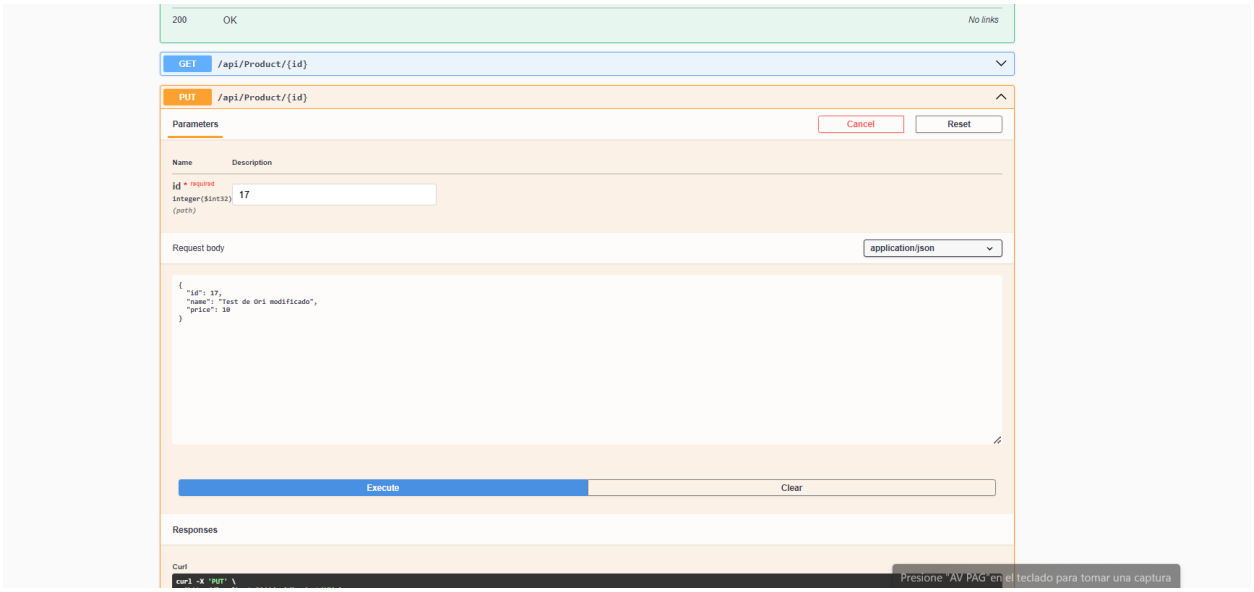
CASE #1



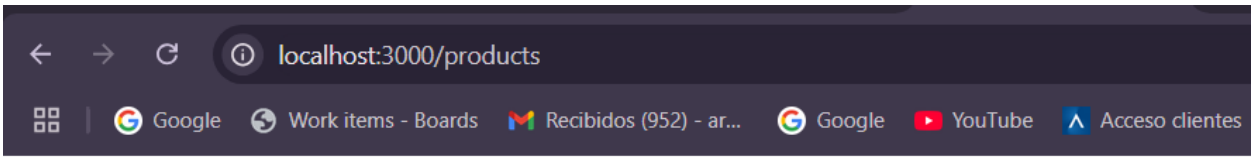
Result



CASE 2: PUT
MODIFY PRODUCT FOR API WITH METHOD PUT



Result

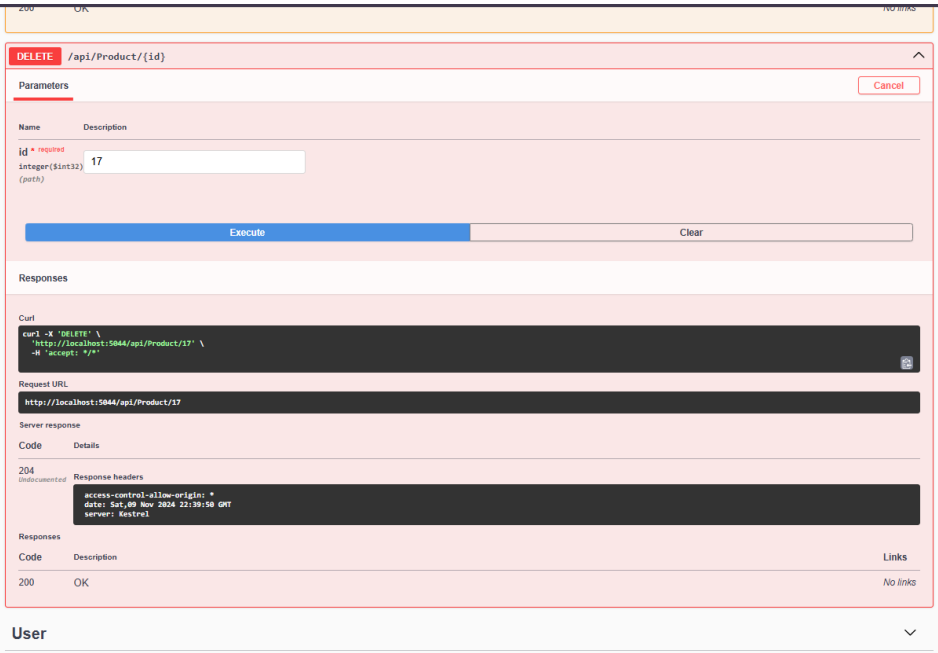


- Login
- Dashboard
- Products
- Orders

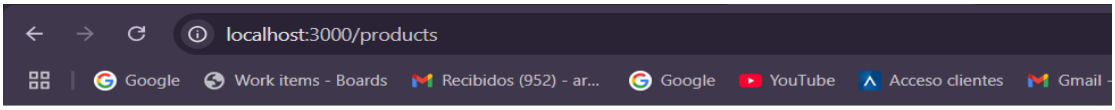
Product List

- Test a ver si cambia - $80
- prueba - $300
- prueba - $300
- pruebaaa - $10
- pruebaaa - $10
- pruebaaa - $10
- Test de agregar producto - $1000
- Test de agregar producto - $1000
- Test de Ori modificado - $10

PRODUCT FOR API WITH METHOD DELETE
CASE 3 DELETE:



Result:

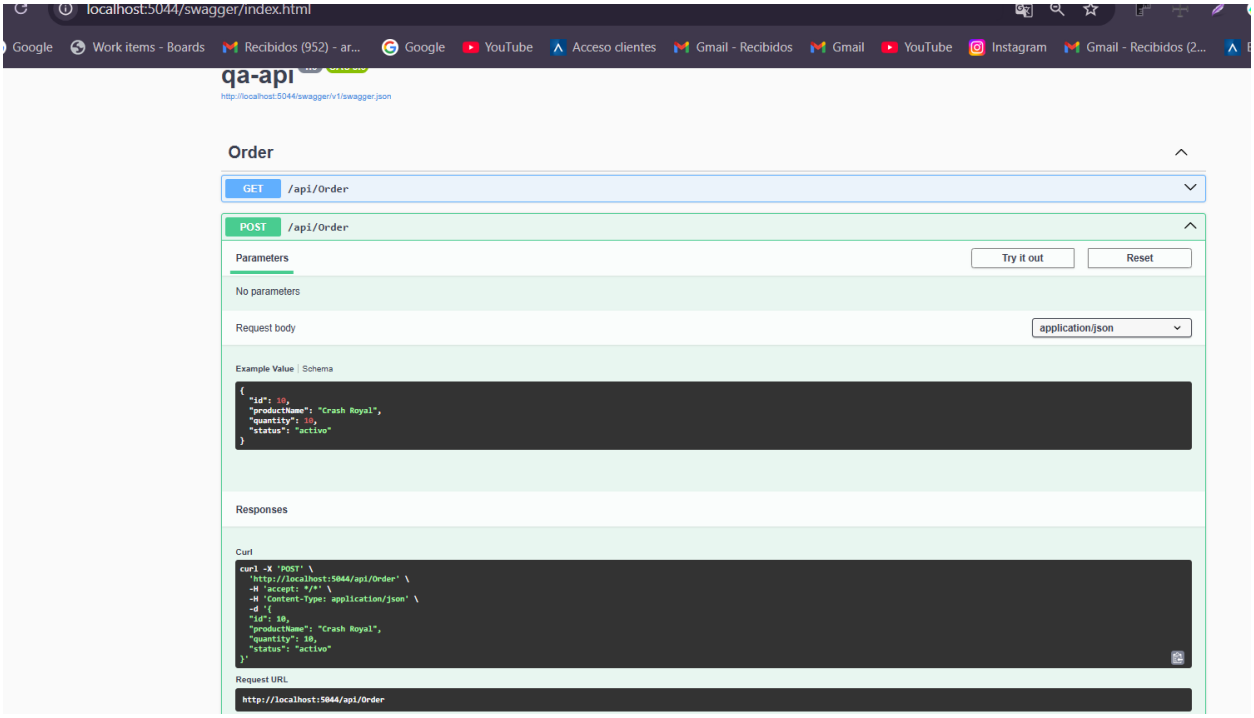


- Login
- Dashboard
- Products
- Orders

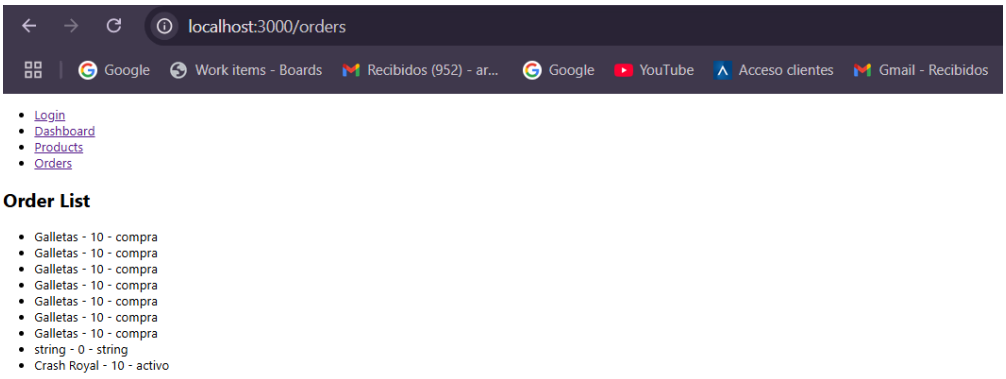
Product List

- Test a ver si cambia - $80
- prueba - $300
- prueba - $300
- pruebaaa - $10
- pruebaaa - $10
- pruebaaa - $10
- Test de agregar producto - $1000
- Test de agregar producto - $1000

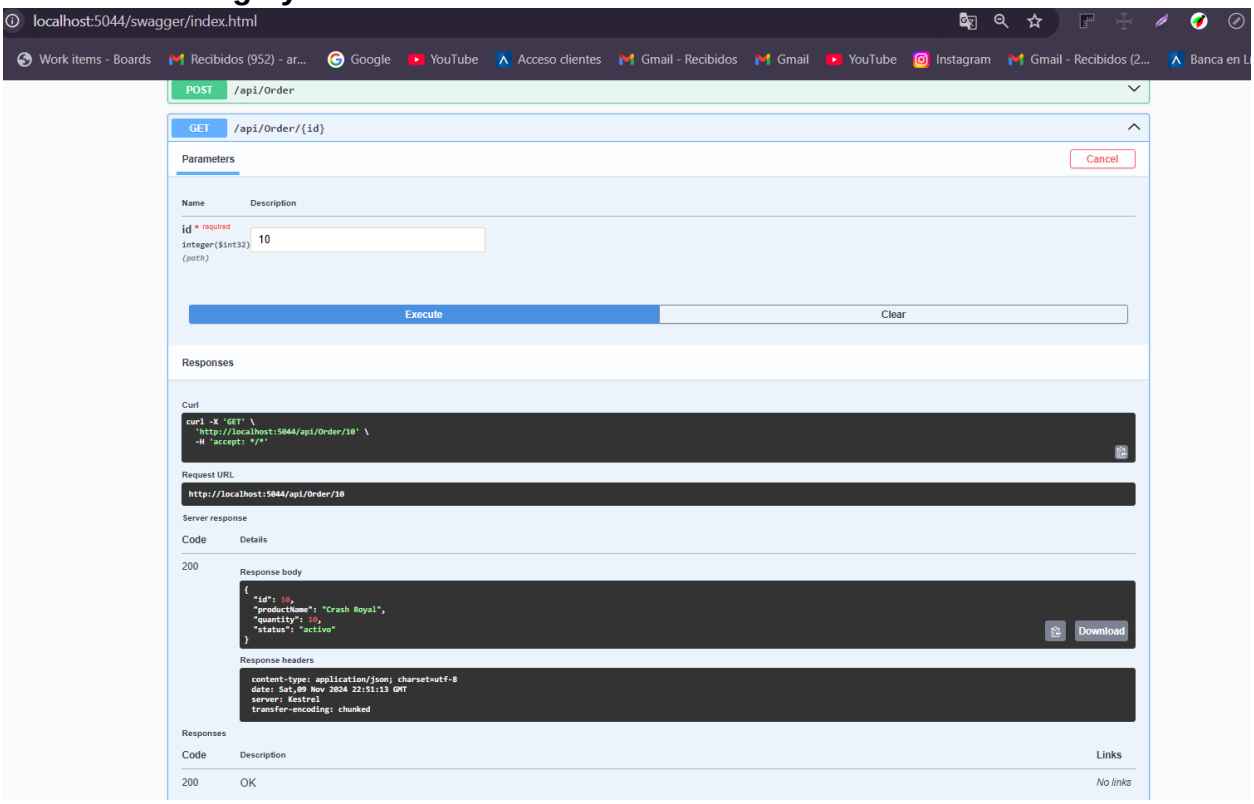
CASE 1
CREATE ORDER FOR API WITH METHOD POST



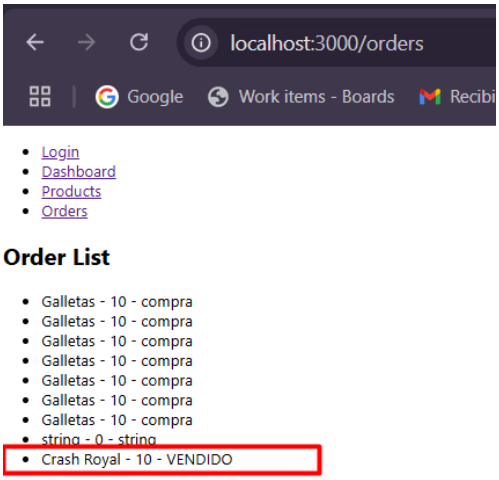
RESULT:



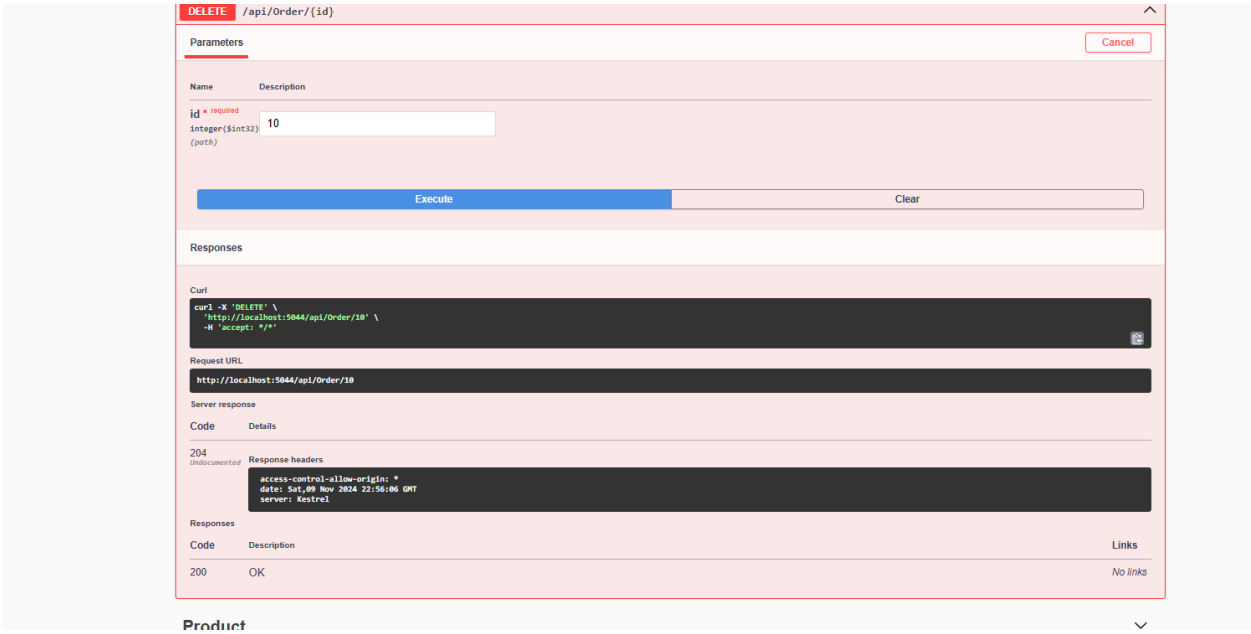
CASE 2 Looking by ID



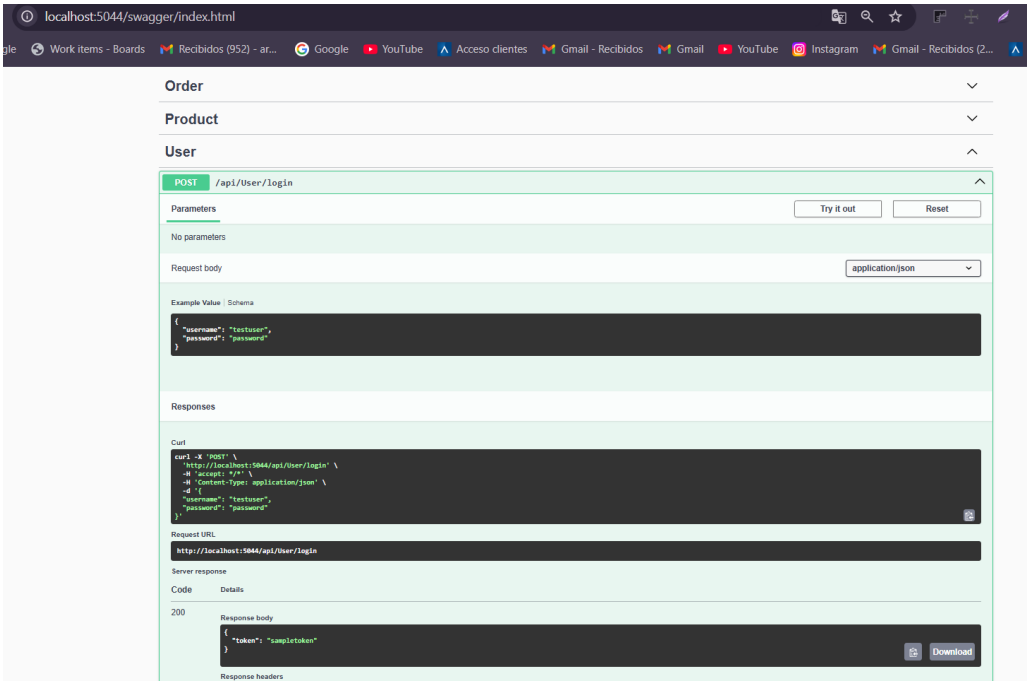
CASE 3 : PUT



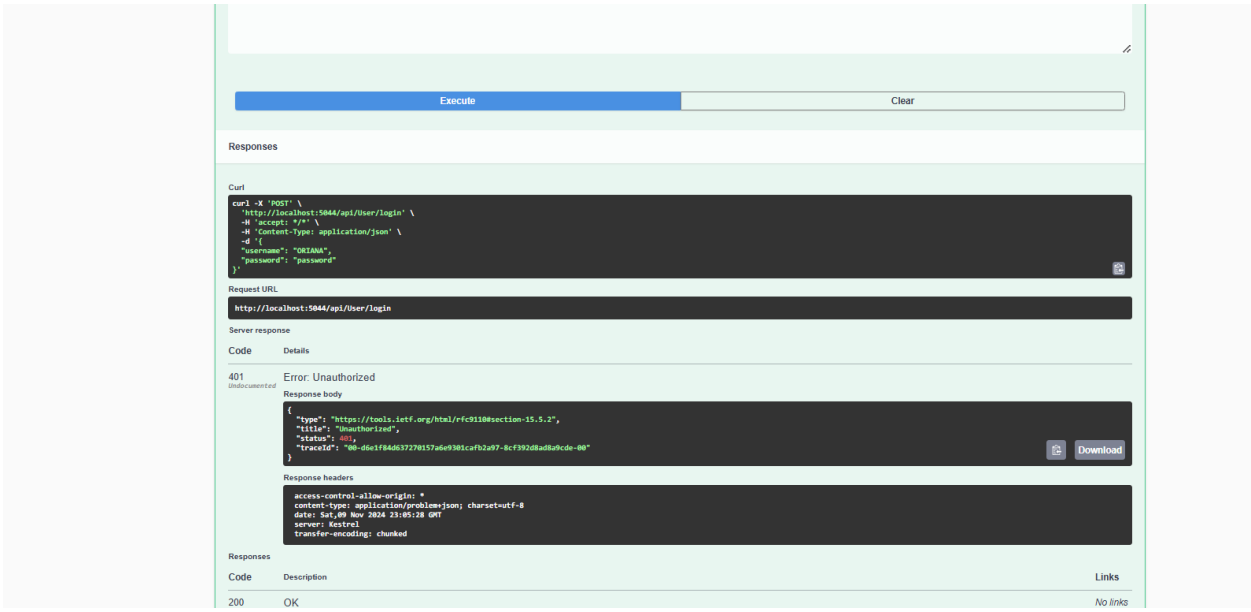
CASE 4: DELETE



USERS CASE 1: POST
ADD

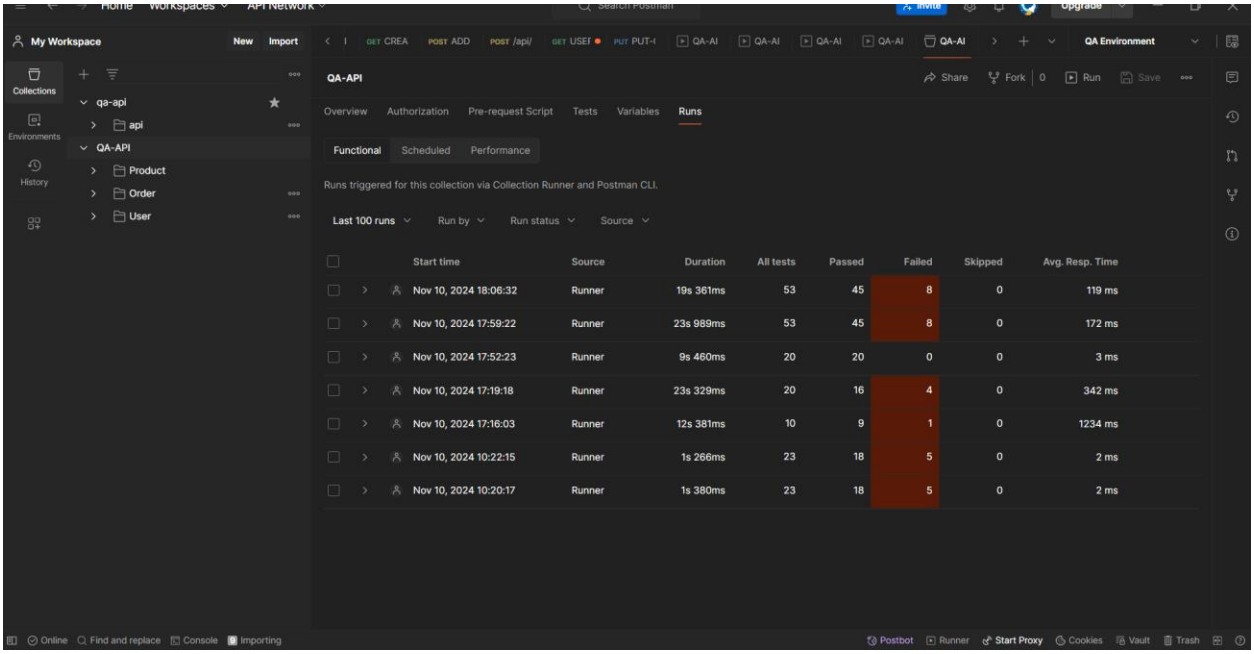


If any data is included that is not in the database it will look like this



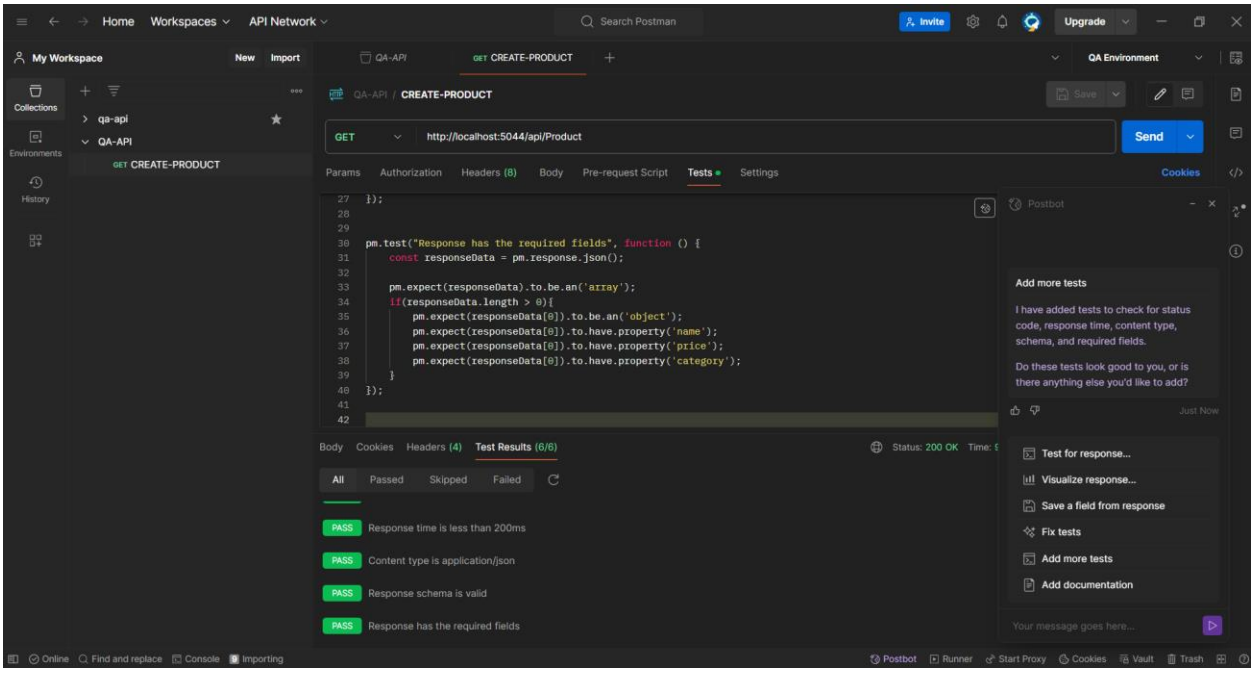
Postman testing

Execute the file

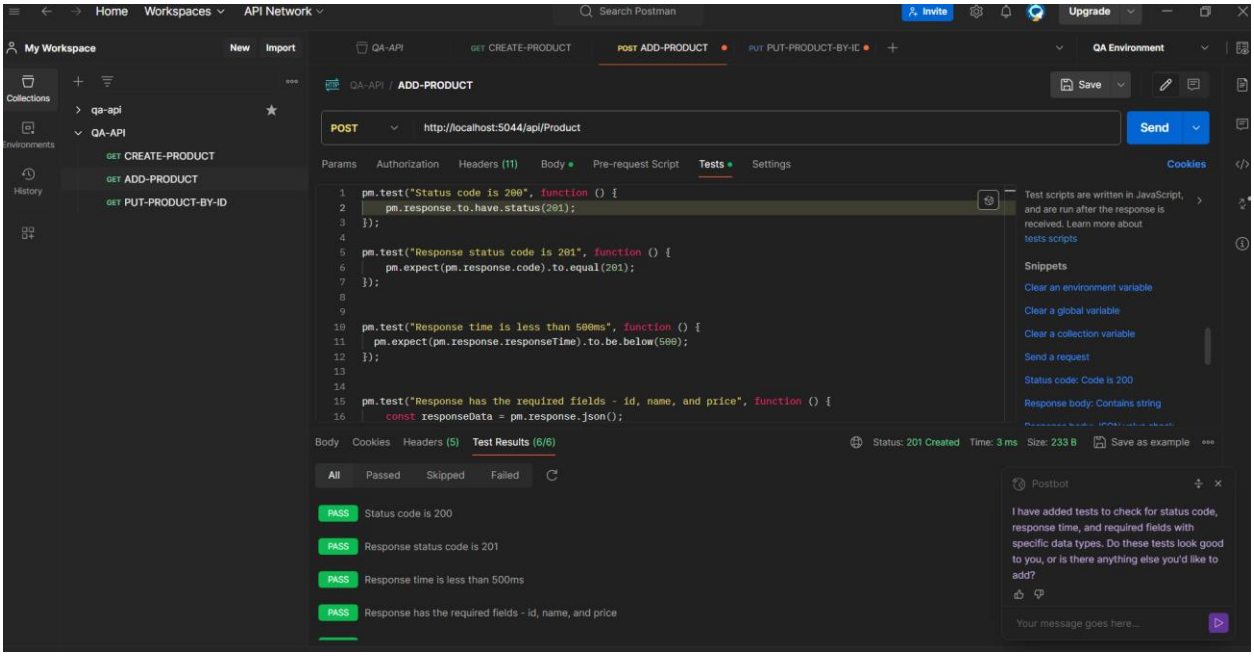


Evidence

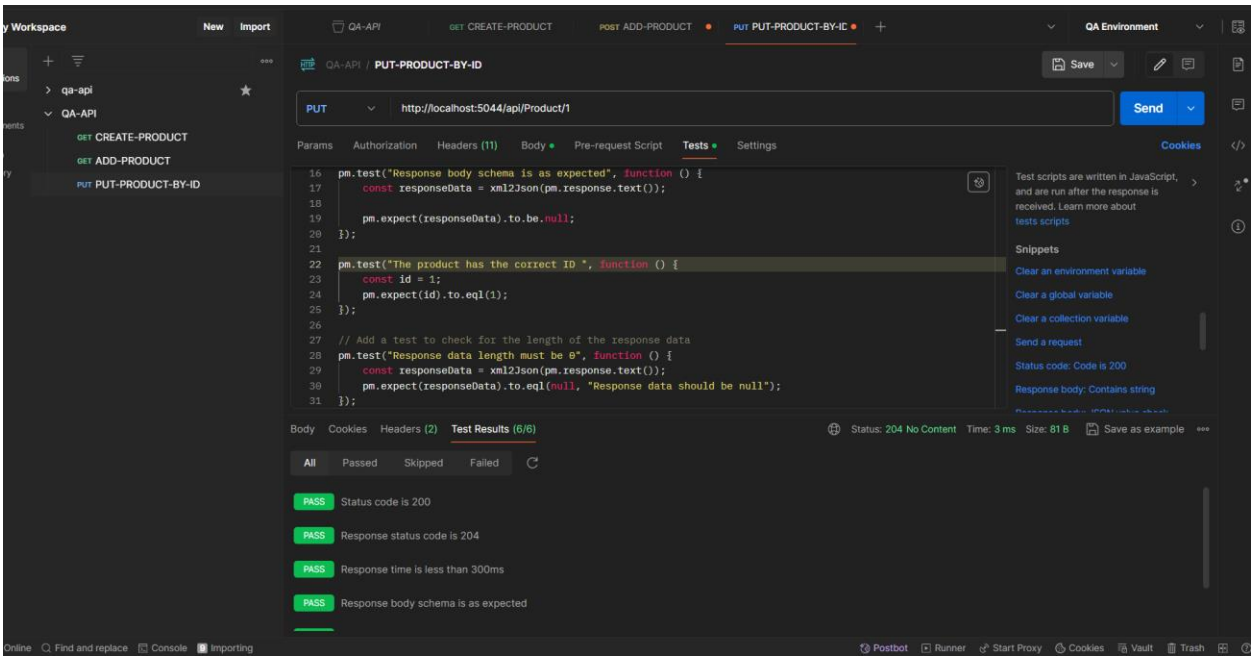
Create Product



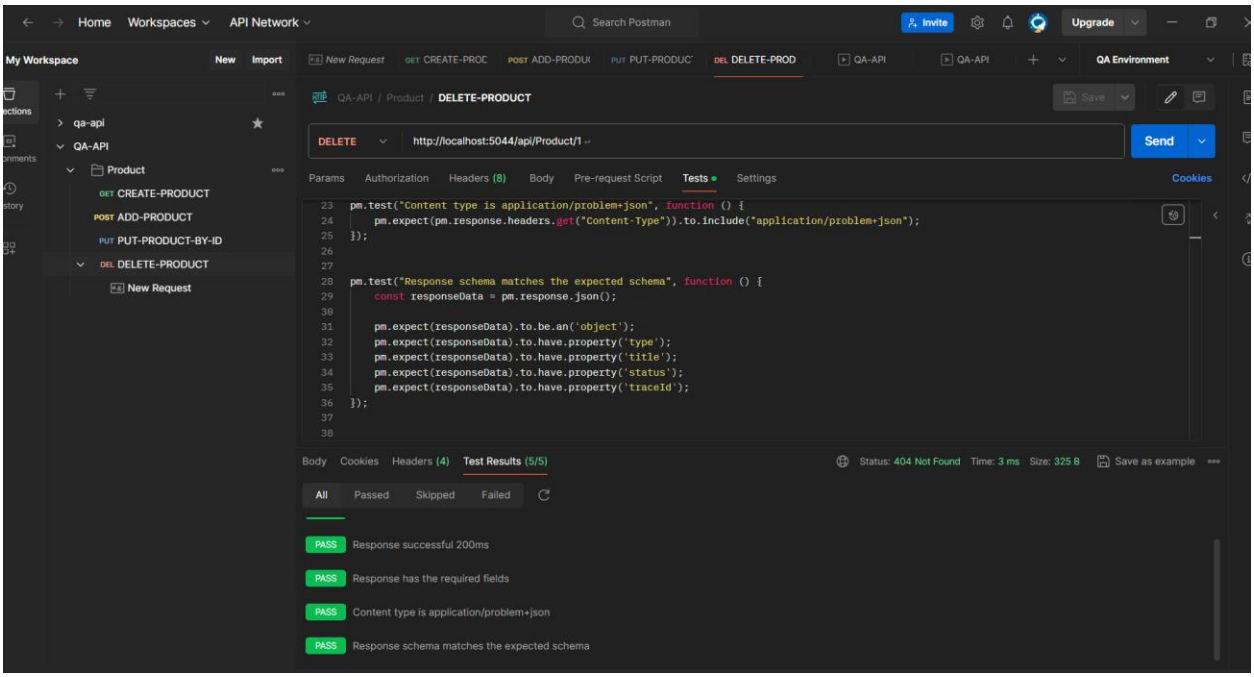
Add Product



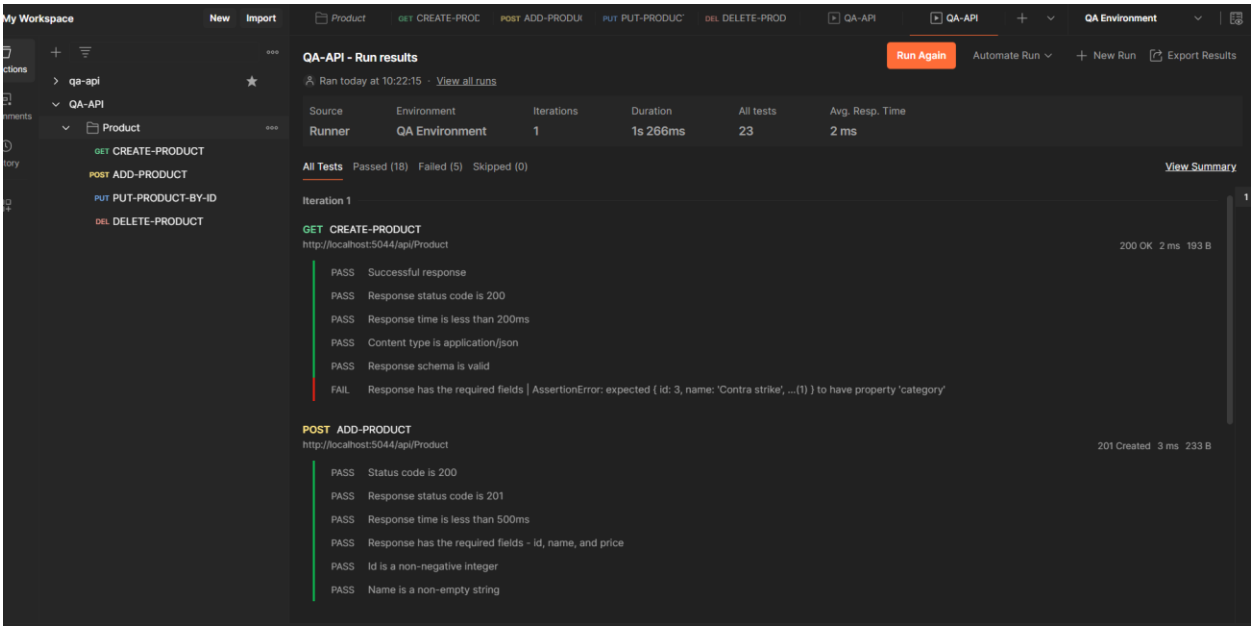
PUT-BY ID Product



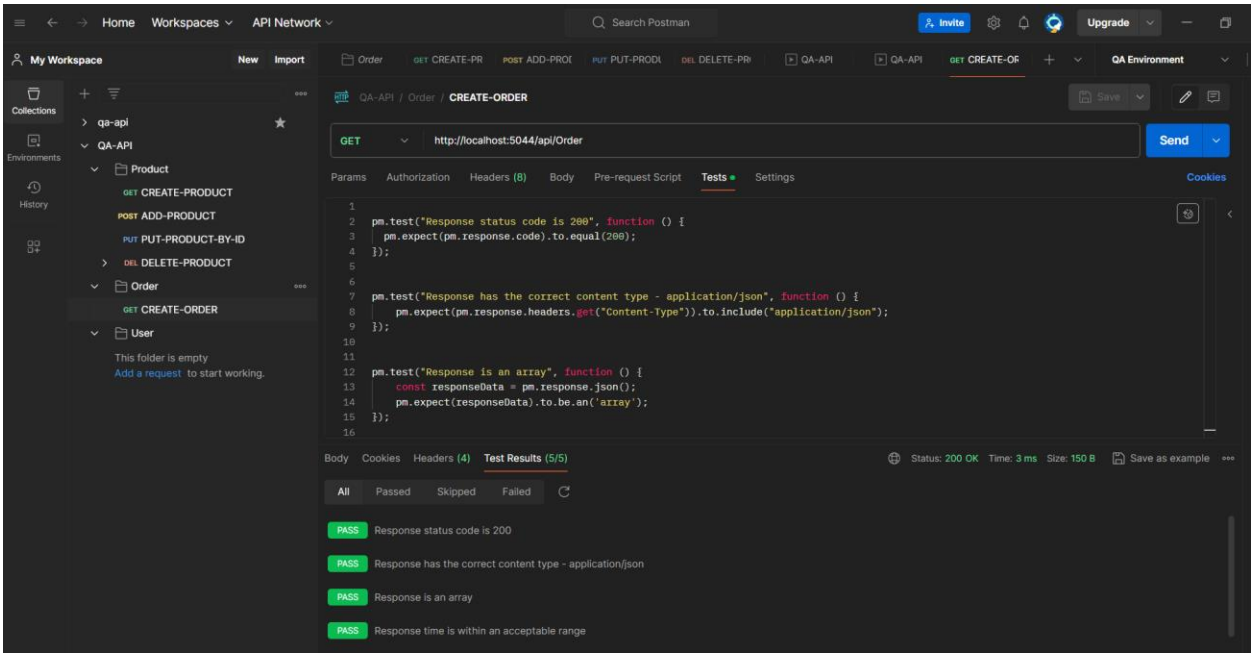
DELETE Product



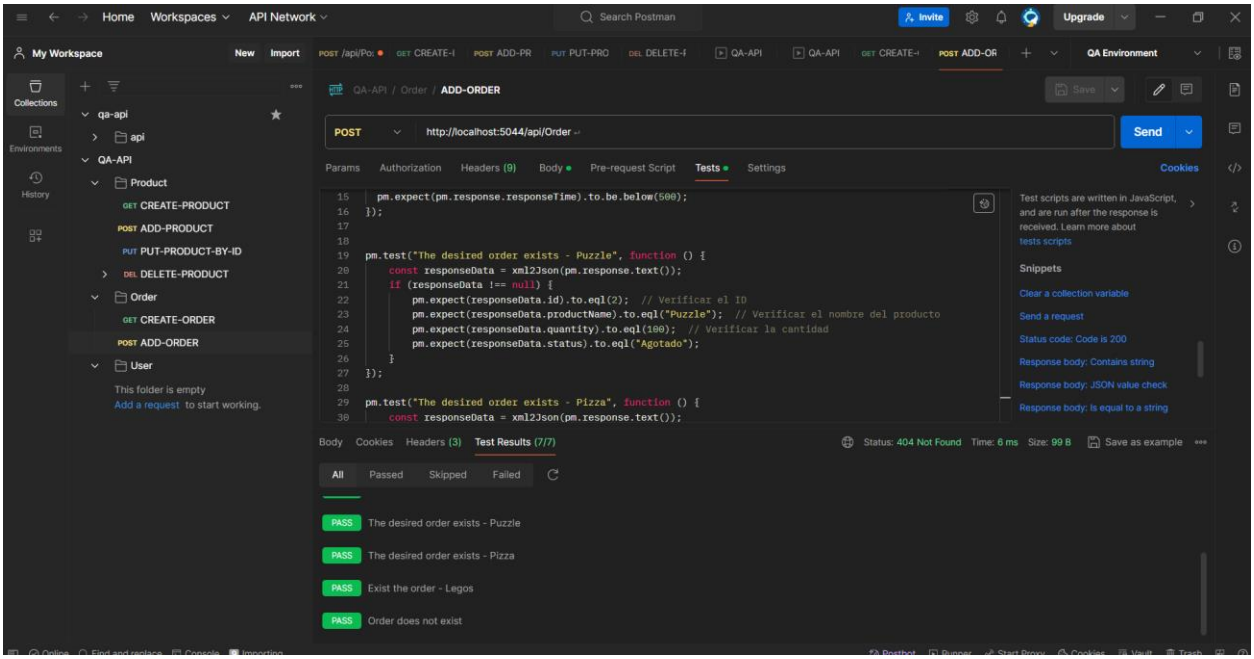
RUN PRODUCT TEST CASE



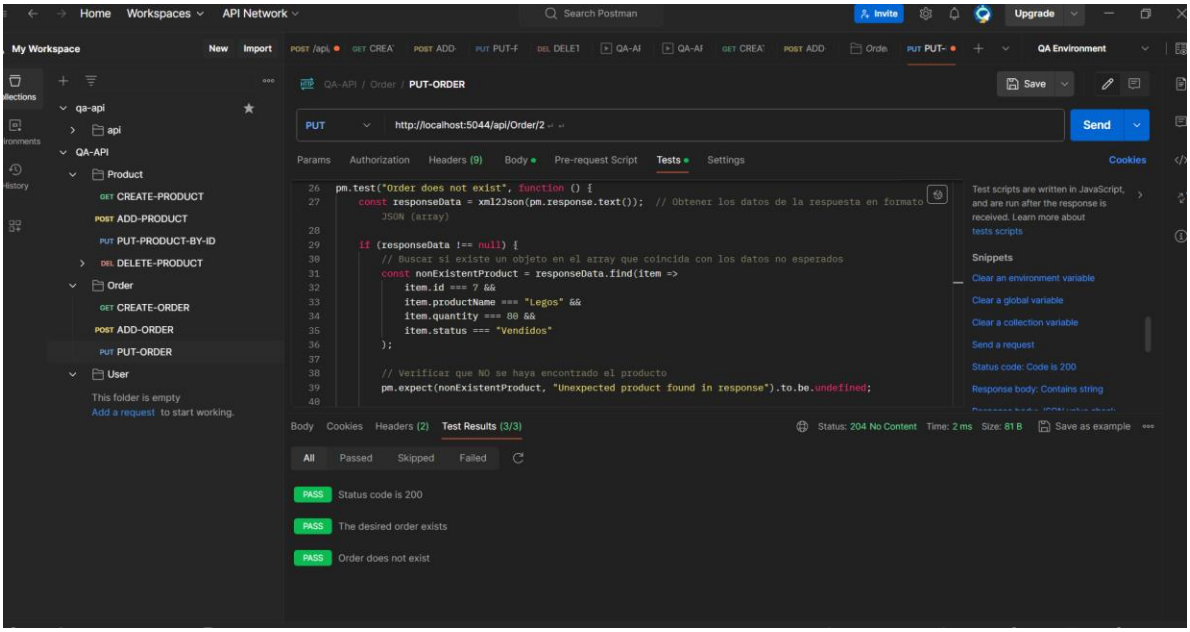
GET ORDER



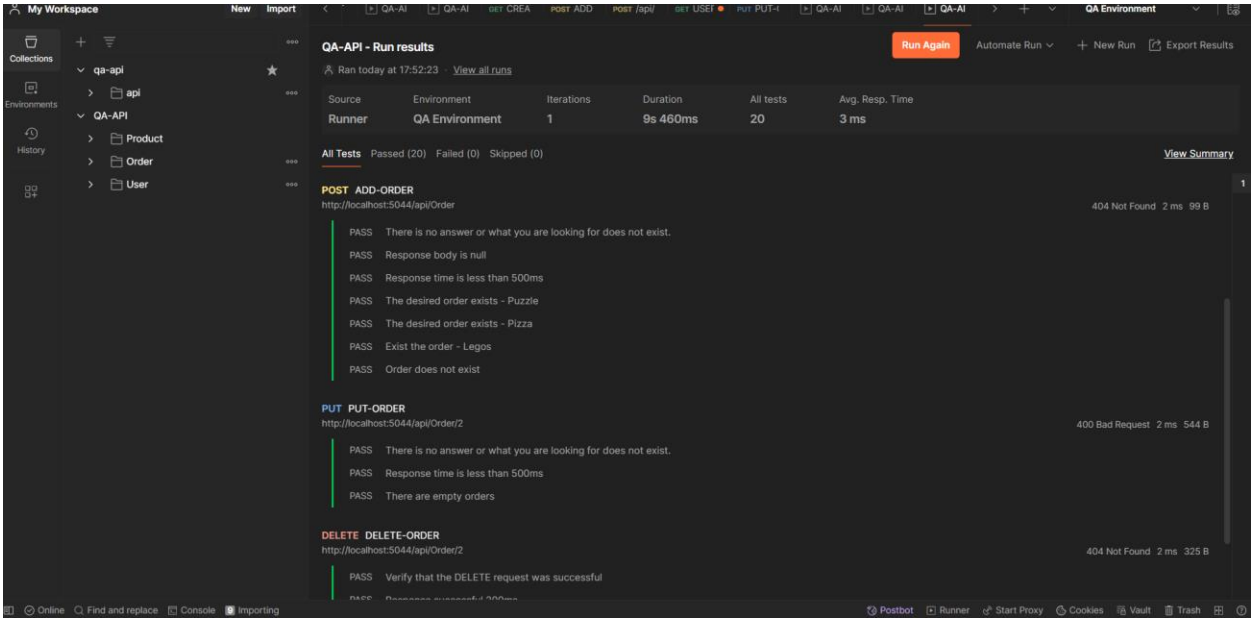
POST ADD - ORDER



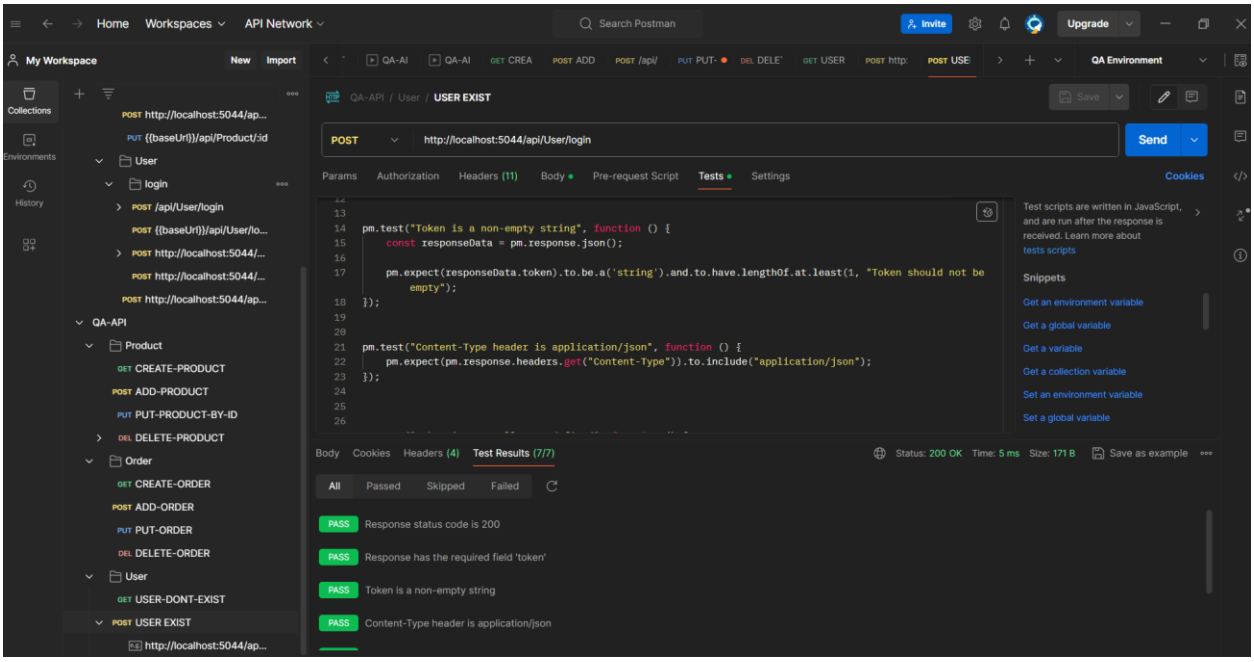
PUT ORDER



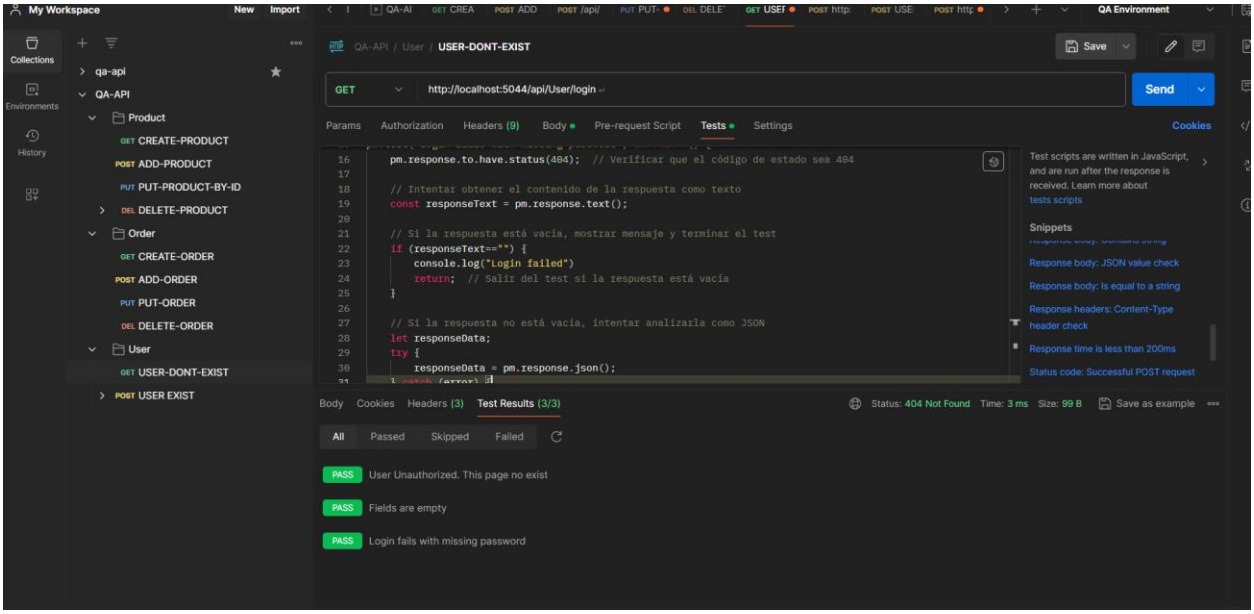
RUN ORDER



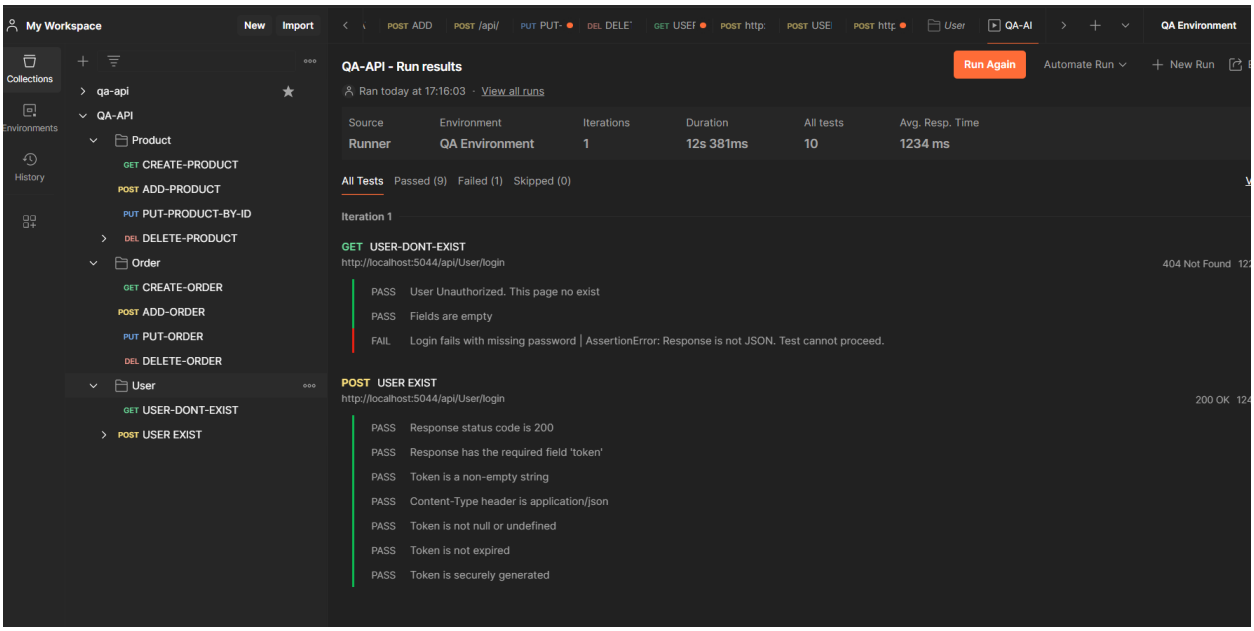
USER EXIST



USER DON’T EXIST



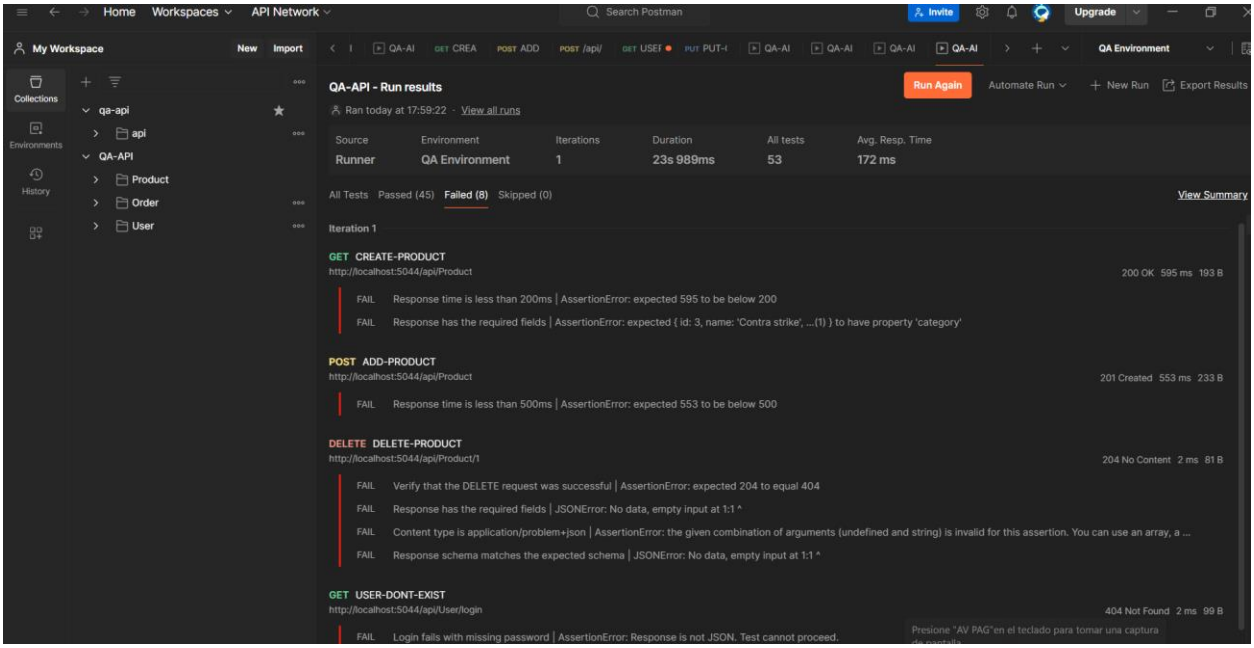
RUN TEST



ANALISYS

When executing the complete program, some elements have failed, either due to test rendering issues that have taken longer, or in some cases the page or the requested information has not been obtained. 8 test cases have failed

EVIDENCE:



So 45 test cases have passed