**tblx-challenge-qa-engineer-template**

**Nuno Barreiro**

To create these tests, I’ve used Karate Framework and Junit 5.

Please note that this is the first time I am using Karate (I have experience using Postman, Robot Framework and RestAssured for API test automation) so the tests I present here are very basic and probably don’t follow the best practices for this tool.

**How to access/run the tests:**

The tests can be run in two ways:

* By right-clicking in the feature file > Run As > Cucumber feature.

Uma imagem com texto

Descrição gerada automaticamente

* By running the tests, by tags, in command line, using the following command:

project path> **mvn test -Dtest=SampleTest#testTags**.

You can change the tags to run specific test cases in the RunnerTest.java class.

Tokens are stored in Karate-config.js file. If tokens expire and you need new one to perform the tests, I’ll provide new ones.

**Code of the automated tests and comments that explain what they do:**

All the code is written as cucumber scenarios, written using Gherkin syntax, in feature files **(Path:/Strava/src/test/java/features).** There is a small explanation on what the Test Case does, written below each Scenario title.

There are two feature files, one for the Athlete endpoint and other for Activities endpoint. I didn’t use other support classes (apart from Karate-config.js ) or stepdefinitions file (a great advantage of karate, in my opinion).

JsonObjects with response payloads (used for expected response validation) can be found inside ResponseJson package)

**Documentation of tests cases that were automated and why they are relevant:**

**TC1 - Get Authenticated Athlete Profile Information**

Description: Return and validate the profile information of an authenticated Athlete.

Given the endpoint “<https://www.strava.com/api/v3/athlete>” endpoint.

When an authenticated user performs a GET request

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And response should match with the information of the authenticated athlete

**TC2 - Get Unauthenticated Athlete Profile Information**

Description: Validate if the request to view the profile information of an unauthorized athlete is not possible.

Given the endpoint “<https://www.strava.com/api/v3/athlete>” endpoint.

When an user without valid authentication performs a GET request to view profile information

Then the response status should be 401

And the response time should be less than 1000 milliseconds

And user should receive “Authorization Error” message

**TC3 – Update the weight of the currently authenticated athlete**

Description: Test PUT request to update the weight of the authenticated athlete and validate if it was successfully changed,

Given the endpoint “https://www.strava.com/api/v3/athlete”

When an authenticated user performs a PUT request to update the weight

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And athlete’s weight should be updated

**TC4 – Get Athlete Activity Stats**

Description: Test Get request to return and validate the return of the activity stats of an athlete, by ID.

Given the endpoint “https://www.strava.com/api/v3/athletes/13836528/stats”

When an authenticated user performs a GET request to get status by athlete id

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And athlete’s activity stats should be correctly returned

**TC 5 – Get Authenticated Athelete Activities**

Description: This Test Case allows us to return and validate the activities of an authenticated athlete.

Given the endpoint “https://www.strava.com/api/v3/athlete”

When an authenticated user performs a GET request

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And authenticated athlete’s activities should be correctly returned

**TC 6 – Get Activity by id**

Description: Return and validate an activity that is owned by the authenticated athlete, by activity id.

Given the endpoint “https://www.strava.com/api/v3/activities/{activity\_id}”

When an authenticated user performs a GET request

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And the activity that matches the id, should be correctly returned

**TC 7 – Get Activity by Invalid id**

Description: Validate that Get activity by id, using an invalid or non existing id, returns 404 Not Found, and that no activity is returned.

Given the endpoint “https://www.strava.com/api/v3/activities/{activity\_id}”

When an authenticated user performs a GET request

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And the activity that matches the id, should be correctly returned

**TC 8 – List Activity Comments**

Description: Returns and validates the comments on a given activity, by activity id.

Given the endpoint “https://www.strava.com/api/v3/activities/{activity\_id}/comments”

When an authenticated user performs a GET request

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And the activity comments, should be correctly listed

**TC 9 – Create An Activity**

Description: Tests the POST request to create a manual activity for an athlete and validates response body, for a successful activity creation.

Given the endpoint “https://www.strava.com/api/v3/activities

When an authenticated user performs a POST request with a request body

Then the response status should be 201

And the response time should be less than 1000 milliseconds

And the activity should be successfully created and match the request body

**TC 10 – Update An Activity**

Description: Tests the PUT request to update an activity that is owned by an authenticated athlete, by activity id, and validates if the update was successful.

Given the endpoint “https://www.strava.com/api/v3/activities/{activity\_id}”

When an authenticated user performs a PUT request with the information he wishes to update, in the request body

Then the response status should be 200

And the response time should be less than 1000 milliseconds

And the activity should be successfully updated