**My Insurance app acceptance tests**

**What do we want to test?**

* We want to test the endpoint "/api/v1/users/<id>/products" that is the new functionality added to the API.
* This endpoint returns the list of products for a concrete user.
* This is defined in file: [project/controllers/api/user\_controller.py](https://github.com/ricardoahumada/myinsuranceapp/blob/main/project/controllers/api/user_controller.py). The code is

....

@app.route('/api/v1/users/<id>/products', methods=['GET'])

@jwt\_required()

def api\_get\_user\_products(id):

# print(get\_jwt\_identity)

products = get\_user\_products(id)

return jsonify(products)

....

* As can be seen, the endpoint:
  + defines the path "/api/v1/users/<id>/products". <id> must be replaced by a concrete user id, like 1 or 2 or 3.
  + requires a token with "@jwt\_required()"
  + returns a list of products in json format

**How do we test it?**

* For testing this, we need to send requests to the endpoint.
* We are going to need to send a token, because is a "restricted" area. So we need to get a token first (in a concrete function).
* Then, we can send a request with the token and evaluate the response, verifying that we receive the response code 200 and a list of products (another test function).
* We can test too invalida situations, like sending invalid/fake tokens or requesting products for non-existing users (e.g. user 34567, doesn't exist in database)

**Tests steps**

1. First, we need a function to get the token: since the endpoint is restricted area.
2. Then, a function for sending VALID requests to the endpoint, using the valid token. - If you want you can test various valid scenarios in various functions.
3. Finally, is a good idea to create a function for sending INVALID requests to the endpoint.
   * For example sending invalid or fake tokens
   * Requesting non-existing users
   * etc.
   * you can use various functions for this

**Two types of tests**

We can use two types of tokens for testing the endpoint:

* Using the **"flask test\_client"**: to simulate the requests.
  + For this, we need to **import the app** in the test case.
  + We DON'T need the app running for real.
  + You can see these tests in: [tests/acceptance-flask/test\_app\_flask\_tester.py](https://github.com/ricardoahumada/myinsuranceapp/blob/main/tests/acceptance-flask/test_app_flask_tester.py)
* Using the python **"request"** library: for sending real requests to the endpoint.
  + For this we DO need the **app be running** (executing: python3 runserver.py)
  + You can see these tests in: [tests/acceptance-request/test\_app\_request.py](https://github.com/ricardoahumada/myinsuranceapp/blob/main/tests/acceptance-request/test_app_request.py)