WEB ANALYTICS COURSE 4 - SEMESTER 2
BACHELOR IN DATA SCIENCE AND ENGINEERING

LAB 2 APIs - CREATING MY FIRST WEB API

1. LAB PREPARATION:

- **Study and have clear the concepts explained in the theory class.**
- **It is assumed students have experience in using Python notebooks. This lab requires to use a local installation (e.g., local python installation + Jupyter). This lab does not work with Google Colab**
- **Study the code included in the baseline files we provide to complete the lab: server.ipynb and client.ipynb. We recommend you to complete Milestone 1 to test the correct connectivity between the server and the client before coming to the first lab session.**

2. INTRODUCTION:

The goal of this lab is to create a simple but operative Web API. To this end the lab is divided into 7 different milestones that need to be completed within the 2 sessions assigned to this lab. Students have to request the presence of the instructor everytime they want the instructor to verify whether the solution for one (or more) milestones is correct.

In this lab, students will implement two players: the ***client*** and the ***server***. The client is in charge of generating requests in order to retrieve data from the server they query. The server role is to process the requests, retrieve the data from an associated database and send back the requested information. In this lab, students will have to fill two files referred to as ***server.ipynb*** and ***client.ipynb*** using python in order to provide and validate the correct functionality of the tasks defined for each milestone.

The instructors are providing a very simple version of both files that allow the client to send queries and obtain a dummy answer for each of the milestones. In addition, we are providing a JSON file that will be used as the server database. The name of the file is **json_rand_users_5000.json.** Finally, the instructors are also providing a small csv file that will be exclusively used in the last milestone (Milestone 7), f**ile_acces_token_M7.csv.**

You can download the four files **server.ipynb**, **client.ipynb**, **json_rand_users_5000.json** and **file_acces_token_M7.csv** from Aula Global.

Following, we provide practical information about the lab logistics:
- The lab will be done in groups of 2 students.
- The lab uses two complete consecutive sessions (4 hours). The students are expected to complete the 7 milestones proposed in the lab.
- The final mark will be computed as a function of the number of milestones successfully completed

- Each group should also share their lab notebook with the professor upon the finalization of the lab.
- Upon completing each milestone, students should call the professor, who will check the correctness of the solution (*If the professor is busy, do not wait for them, move to the next milestone. Similarly, some students may prefer to complete several milestones before asking the instructor to check them.Similarly, some students may prefer to complete several milestones before asking the instructor to check them.*).

3. FILES

In this section, we describe the content of the files and explain what part of them has to be modified by the students.

server.ipynb:

- Main method
  - It starts the execution of the server.
  - The server will run until you stop it or an error/exception happens.
  - You DO NOT have to modify this part of the code.
- A class MyServer with a function do_GET:
  - This class handles the received requests.
  - This function also distributes the queries to the appropriate function based on the invoked endpoint (there is 1 endpoint per milestone as explained later in this document).
  - This function also sends back the HTTP output code and the JSON-formatted answer.
  - You DO NOT have to modify this part of the code
- 7 different functions one per endpoint to be implemented.
  - Each function corresponds to one milestone of the lab.
  - Each function currently includes a dummy code that has to be substituted by the actual code covering the functionality requested in each particular endpoint (and milestone).
  - You HAVE to modify this part of the code.
  - We list the endpoints next. Later on this manuscript, we will describe what is the functionality each point has to implement
    - test_connectivity_M1
    - country_count_M2
    - country_count_M3
    - age_groups_M4
    - users_M5
    - users_control_error_M6
    - access_token_M7

LAB 2 APIs - CREATING MY FIRST WEB API

- The code is commented and above each function it includes an explanation of what is expected from that function (which corresponds to a specific endpoint and milestone).

client.ipynb:

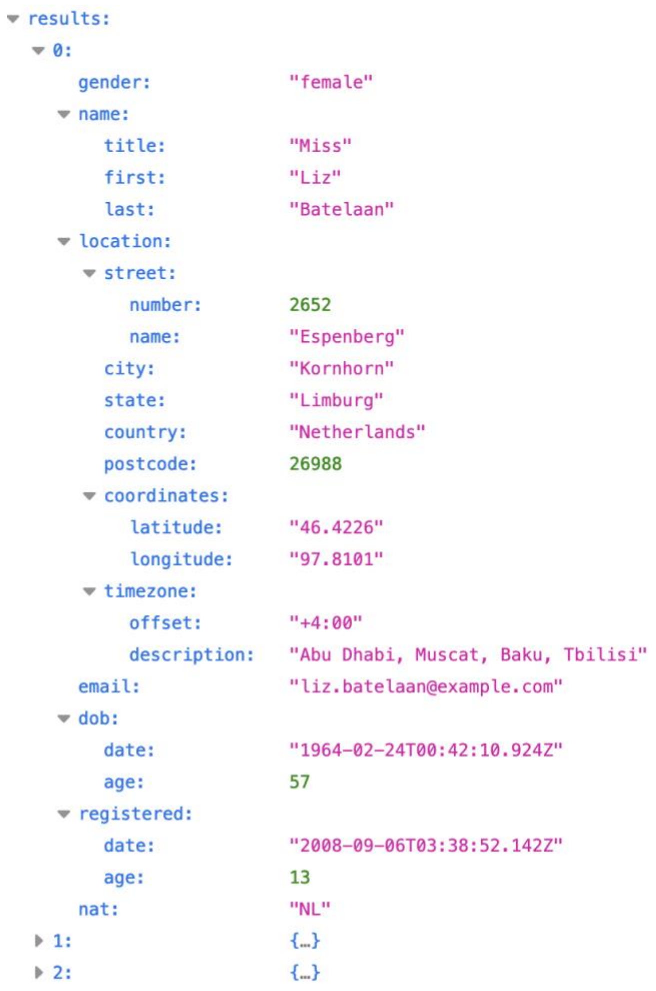
- Function connect_to_server(url)
    - This function sends a request to the server using the url it receives as parameter
    - This function returns the HTTP status code of the response and the contentn of the response itself
    - It SHOULD NOT be necessary to modify this function.
- List of endpoints that can be queried
- Preliminary code that prints the received output on the screen.
    - You HAVE to modify this code to validate that the implementation of the different endpoints is working properly.
    - This will require to manipulate the JSON response to print in the screen whatever is asked in this manuscript for each particular milestone

json_rand_users_5000.json:

- This file has been created using the API https://randomuser.me/api
- It is formatted as a JSON file.
- It will play the role of the database of the server.
- It includes 5000 random generated users that include the following information
    - gender
    - name
        - title
        - first
        - last
    - location
        - street
            - number
            - name
        - city
        - state
        - country
        - postcode
        - coordinates
            - latitude
            - longitude
        - timezone

LAB 2 APIs - CREATING MY FIRST WEB API

        - offset
        - description
  - email
  - don (which corresponds to date of birth)
    - date
    - age
  - registered (which corresponds to the time the account was created)
    - date
    - age
  - nat (which correspond to the nationality



```
▼ results:
  ▼ 0:
      gender:           "female"
    ▼ name:
        title:          "Miss"
        first:          "Liz"
        last:           "Batelaan"
    ▼ location:
      ▼ street:
          number:       2652
          name:         "Espenberg"
        city:           "Kornhorn"
        state:          "Limburg"
        country:        "Netherlands"
        postcode:       26988
      ▼ coordinates:
          latitude:     "46.4226"
          longitude:    "97.8101"
      ▼ timezone:
          offset:       "+4:00"
          description:  "Abu Dhabi, Muscat, Baku, Tbilisi"
      email:            "liz.batelaan@example.com"
    ▼ dob:
        date:           "1964-02-24T00:42:10.924Z"
        age:            57
    ▼ registered:
        date:           "2008-09-06T03:38:52.142Z"
        age:            13
      nat:              "NL"
  ▶ 1:                  {…}
  ▶ 2:                  {…}
```

Fig 1. Example of a user in JSON users file

file_acces_token_M7.csv

WEB ANALYTICS COURSE 4 - SEMESTER 2
BACHELOR IN DATA SCIENCE AND ENGINEERING

LAB 2 APIs - CREATING MY FIRST WEB API

This file will be only used in the last milestone (M7). It includes the user ID and associated access token for 20 users. This file will be used to validate whether the user ID and access token included in the request required in M7 is valid (see Milestone 7 in Section 3 for more details). You can find below a table with the actual content of the file.

User_ID	Access_Token
1231	eeJiqB3RnUCuAI9061ntzg==
3213	MJgloJUbPPHyXie2HO6CQw==
1459	EeXfxoQi5pdWOkJTujYGFQ==
2231	2VmO5wUgDzzk0cYVyvhOxA==
9187	$P$B2qcD/.FkLKT0WP286HjRiL.aUyxtJ/
8121	$P$BZzGZ/Rk9h4LtOZ2no3B.WSlrkssKX0
9123	$P$BZlw7PF1j.XDezr9sUj6moCjBlSx/e0
8631	eeb15117bd15cf4c9474acda0b2f6598
7642	da6b4d1dd5977f304a64c4e50a9da3f12ae6cb72
5438	$P$BGDF.QgHPRCnEiLOZhW1nESvY0lX2y1
7423	ZHNmc2RhZmFzZGY=
6721	$P$BmqF8977bjATprjcsd0A0a60Sd52bK/
5555	BkL5pUZd5o7mQTSNFXEEmRtCwBgjQC1
6589	e947f489c7fb37313533bfb96903f3bc659e9c11
4231	3568f731f0d7d1d9504007a2859835dab365b77c
7879	764f1c0faf361de32f831a927bfb4a8f
8963	$P$BBrzwYk6WFj6rZPZ04al6AaQ7HHh551
7684	8a39d05673d8bdd9402e0671f64a1102

LAB 2 APIs - CREATING MY FIRST WEB API

4092	54a7b18f26374fc200ddedde0844f8ec
4576	$P$B3MoN2EzAJlGKzKak0dFsd2FGyVXiB.

4. MILESTONES EXPLANATION

In this section, we explain in detail the 7 milestones (one per endpoint) to be accomplished. We explain the functionality the server has to provide as well as what is the expected output the client has to generate on the screen to show the correct operation of the endpoint.

4.1. MILESTONE 1

- **GOAL:** Test the correct connectivity between the client and the server.
- **FUNCTION**: test_connectivity_M1(text)
- **endpoint example:** http://localhost:8080/test_connectivity_M1
- **DISPLAY ON CLIENT:** The content of the JSON response
  Example:
  {"results": [{"Connectivity": "OK"}]}

4.2. MILESTONE 2

- **GOAL:** Return a JSON response including the number of users per country in the database based on the field “nat”.
- **FUNCTION:** count_countries_M2(text)
- **endpoint example:** http://localhost:8080/count_countries_M2
- **DISPLAY ON CLIENT:** Process the JSON in the response and print on the screen the sorted list of countries according to the number of users as follows.
  1: TR(317)
  2: DE(317)
  3: NO(315)
  4: BR(312)
  5: IR(311)
  6: CA(297)
  7: FR(297)
  8: CH(296)
  9: IE(293)
  10: DK(290)
  11: NZ(288)
  12: AU(285)
  13: ES(281)

14: GB(278)
15: NL(276)
16: FI(276)
17: US(271)

4.3. MILESTONE 3

- **GOAL:** It is the same as M2 but the client can specify the list of countries for which it wants to obtain the response as a parameter in the query with the field "countries". This means if the query only includes AU, BR and CA, the server will only include the number of users from Australia, Brazil and Canada in the JSON response.
- **FUNCTION:** count_countries_M3(text)
- **endpoint example**: http://localhost:8080/count_countries_M3?countries=AU,BR,CA
- **list available countries**: AU, BR, CA, CH, DE, DK, ES, FI, FR, GB, IE, IR, NO, NL, NZ, TR, US
- **DISPLAY ON CLIENT:** Process the JSON in the response and print on the screen the sorted list of countries according to the number of users as follows.

1: BR(312)
2: CA(297)
3: AU(285)

4.4. MILESTONE 4

- **GOAL:** The server has to generate a JSON response that contains the information for the number of users in the database within the following age groups: <18, 18-29, 30-49, 50-64, >64.
- **FUNCTION:** age_groups_M4
- **endpoint example**: http://localhost:8080/age_groups_M4
- **DISPLAY ON CLIENT:** Process the JSON in the response and print on the screen the number of users per age group as follows.

Age <18: 0
Age 18-29: 634
Age 30-49: 1762
Age 50-64: 1452
Age >64: 1152

LAB 2 APIs - CREATING MY FIRST WEB API

4.5. MILESTONE 5

- **GOAL:** Find the users in the database meeting the parameters values included in the request. The parameters and the values they can adopt are:
    - **countries**=AU,BR,CA,CH,DE,DK,ES,FI,FR,GB,IE,IR,NO,NL,NZ,TR,US [use the "nat" field for this]
    - **gender**=male,female
    - **month**= 01,02,03,04,05,06,07,08,09,10,11,12
    - **age**: A list of values separated by comma. Any value between 18 and 99 is valid.

  If a parameter is not included in the request you need to consider all the possible values for that parameter when generating the response. For instance, if the gender parameter is not included in the request then the response should include both male and female users. In addition, parameters can be provided in any order.

  For each user meeting the parameters value you need to store in the response JSON the following parameters:
    - Name (including the <first> and <last> fields)
    - Gender (using the field <gender>)
    - Age (using the field <age> included in the filed <dob>)
    - Country (using the field <country> included in the field <location>)
    - Postcode (using the field <postcode> included in the field <location>)
    - Email (using the field <email>)

- **FUNCTION:** users_M5
- **endpoint example:** http://localhost:8080/users_M5?countries=FR,GB,IE,IR,NO,NL,NZ,TR,US&gender=female&month=01,02,03,04,05&age=30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45
- **ERROR MANAGEMENT:** It is not required to implement error handling in this milestone.
- **DISPLAY ON CLIENT:** You need to process the JSON in the response and provide the total number of users and display the content associated with the 100 first users in the answer. Note that sometimes the parameters selection will lead to answers including less than 100 users. In that case, print on the screen all the users. The output on the screen should follow the format below:

LAB 2 APIs - CREATING MY FIRST WEB API

```
NUM. USERS:165
User 1:
        -Name: Linne Hoogkamer
        -Gender: female
        -Age: 40
        -Country: Netherlands
        -Postcode: 42173
        -Email: linne.hoogkamer@example.com
User 2:
        -Name: Enola Martin
        -Gender: female
        -Age: 43
        -Country: France
        -Postcode: 93894
        -Email: enola.martin@example.com
User 3:
        -Name: Mathilda Skjevik
        -Gender: female
        -Age: 42
        -Country: Norway
        -Postcode: 6630
        -Email: mathilda.skjevik@example.com
User 4:
        -Name: Valentine Thomas
        -Gender: female
        -Age: 35
        -Country: France
        -Postcode: 84452
        -Email: valentine.thomas@example.com
 .....
 .....
 .....
User 100:
        -Name: Cecilia Asphaug
        -Gender: female
        -Age: 42
        -Country: Norway
        -Postcode: 5052
        -Email: cecilia.asphaug@example.com
```

LAB 2 APIs - CREATING MY FIRST WEB API

4.6. MILESTONE 6

- **GOAL:** Implement the error handling for the previous milestone. The handling has to verify the following potential errors:

  - Countries: If a country does not belong to the group AU,BR,CA,CH,DE,DK,ES,FI,FR,GB,IE,IR,NO,NL,NZ,TR,US.
  - Gender: If a gender is neither female nor male.
  - Month: If the mode is an invalid value out of the range 01-12.
  - Age: If it is out of the range 18-99.
  - Field: If after ? in the query there is a field different than “countries”, “gender”, “age” and “month”.

  If at least one of the errors above is identified the request should be considered wrong and the JSON response should report all the discovered errors. In additiion, you should take into account:

  - If the query is correct and the process is successful the response should include a status code 200.
  - If the query includes some error and the process fails the response should include a status code 400 (Bad request in HTTP protocol).
  - You need to include all the errors so you should not stop the verification of the query format when you find the first error.

- **FUNCTION:** users_control_error_M6(text)
- **endpoint example** (same parameters as M5):

http://localhost:8080/users_control_error_M6?countries=FR,GB,IE,IR,NO,NL,NZ,TR,US,IT,SW,MX&gender=women,men&month=01,05,06,12,15,23&age=30,31,32,33,34,35,36,37,38,39,40,12,15,123&city=Madrid

- **DISPLAY ON CLIENT:** If the query is correct display the same aspect as on M5. If the query is wrong the server has to create a JSON response including all the detected errors. In this case, the client has to print on the screen the JSON response received from the server. Next, we show an example of the JSON response for the query above that includes errors in all the fields and includes a not supported field.

{"results": [{"Country Error": "The following country IDs are not supported: IT,SW,MX",
"Gender Error": "The following gender options are not supported: women,men",
"Month Error": "The following month options are not supported: 15,23",
"Age Error": "The following month options are not supported: 12,15,123",
"Field Error": "The following fields are not valid: city"}]}

LAB 2 APIs - CREATING MY FIRST WEB API

4.7. MILESTONE 7

- **GOAL:** This last milestone aims to incorporate some security principles such as a user ID and an associated access token. That means the request will not be processed and the server will not retrieve the information from the database if: (i) the user ID is not included in the file “file_acces_token_M7.csv”, (ii) the user ID is included in the file but the access_token for that user is incorrect. We will use as baseline the code of M5. This means, if the check of the user ID and access token is correct the JSON answers will be the same provided in M5.
- **FUNCTION:** access_token_M7(text)
- **User ID**: It is included in the field <user_ID> in the parameters
- **access token**: It is included in the field <access_token> in the parameters
- **endpoint example:** http://localhost:8080/access_token_M7?access_token=$P$BZlw7PF1j.XDezr9sUj6moCjBlSx/e0&countries=FR,GB,IE,IR,NO,NL,NZ,TR,US&user_ID=9123&gender=male,female&month=01,05,06,12&age=30,31,32,33,34,35
- **DISPLAY ON CLIENT:** In case of error, you need to display the content of the JSON response on the screen. There are three possible errors. Next, we show the expected output for each of them.

  1- No fields included (could be one of them or both of them)
  {"results": [{"user_ID Error": "user_ID field not included in the parameters.", "Access_Token Error": "Access_Token field not included in the parameters."}]}

  2- UserID does not exist
  {"results": [{"user_ID Error": "user_ID 2131 not found in the database."}]}

  3- User ID exists but access_token wrong
  {"results": [{"access_token Error": "access_token 7423hjmn23nsd.121 wrong for the userID 9123"}]}