Programming for IoT applications
Lab 3

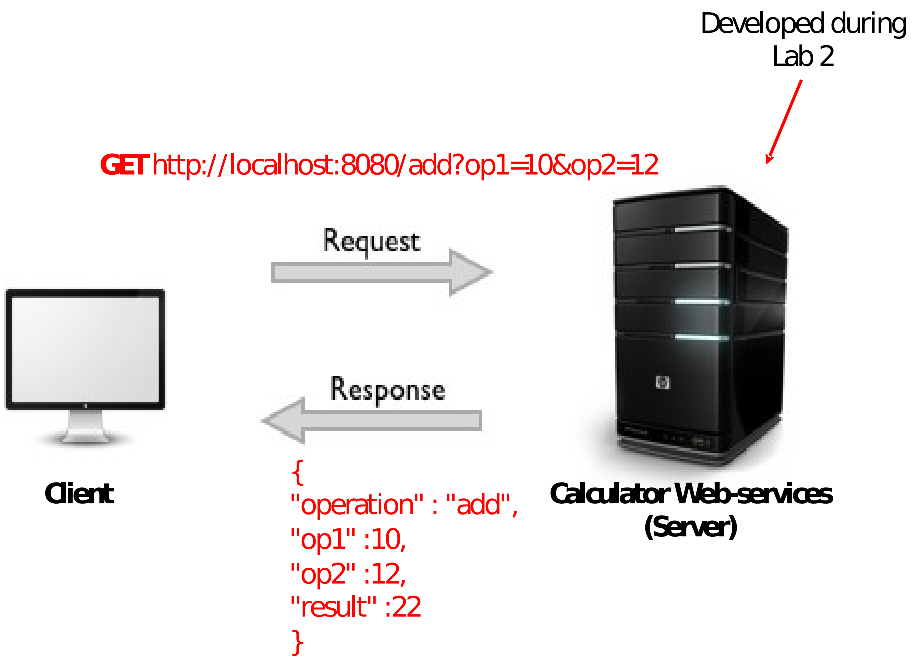
SUGGESTION: Use **Postman**, a chrome plugin, for testing REST web services by managing HTTP requests

---

Exercise 1. Develop a **client** python application for invoking the RESTful calculator developed in *Exercise_1* during the *Lab 2* that:

- asks to end-users the operation to be performed and two operands
- invokes the web services published by the RESTful
- shows the results to end-users in a friendly way (NOT the full JSON)

Architectural schema example



SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)

Exercise 2. Extend *Exercise_3* proposed during the *Lab 1*, for designing a RESTful-style discography manager.
Identify what are the proper HTTP methods (among GET, POST, PUT and DELETE) and develop the web services to:

1. **Search by name <deviceName>**: print all the information about the devices for the given <deviceName>
2. **Searchy by ID <id>**: print all the information about the devices for the given <id>
3. **Search by service <service>**: print all the information about the devices that provides the given <service>
4. **Search by measureType <type>**: print all the information about the device that provides such measure <type>
5. **Insert device:** insert a new device it that is not already present on the list (the ID is checked). Otherwise ask the end-user to update the information about the existing device with the new parameters. Every time that this operation is performed the "last_update" field needs to be updated with the current date and time in the format "yyyy-mm-dd hh:mm". The structure of the parameters of the file must follow the one of the ones that are already present
6. **Print all:** print the full catalog
7. **Exit:** save the discography (if changed) in the same JSON file provided as input.

Then, develop also the **client** python application for managing the discography by invoking the web services provided by the RESTful-style discography manager

Validate the JSON with http://jsonlint.com/

SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)

Exercise 3. Develop your REST web services for getting information from third-party web services for the real bike sharing in Barcelona (Spain). Choose the most suitable HTTP method among GET/POST/PUT/DELETE and the data-format.

Example of bike sharing JSON:

```
{
"url_icon": "\/icons\/ubicacio.png",
"url_icon2": "\/icons\/ubicacio2.png",
"url_icon3": "\/icons\/ubicacio3.png",
"url_icon4": "\/icons\/ubicacio4.png",
"url_icon5": "\/icons\/ubicacio5.png",
"estacions_icon": "\/icons\/estacions.png",
"parametros_filtro": [],
"stations": [{
        "id": "496",
        "type": "BIKE",
        "latitude": 41.404839,
        "longitude": 2.17482,
        "streetName": "C\/ PROVEN\u00c7A, 445",
        "streetNumber": "",
        "slots": 4,
        "bikes": 9,
        "type_bicing": 2,
        "electrical_bikes": 0,
        "mechanical_bikes": 4,
        "status": 1,
        "disponibilidad": 0,
        "icon":"\/icons\/ubicacio-0.png",
        "transition_start": "",
        "transition_end": ""
},    {
        "id": "424",
        "type": "BIKE",
        "latitude": 41.379632,
        "longitude": 2.192662,
        "streetName": "PG. MAR\u00cdTIM DE LA BARCELONETA",
        "streetNumber": "",
        "slots": 7,
        "bikes": 20,
        "type_bicing": 2,
        "electrical_bikes": 1,
        "mechanical_bikes": 19,
        "status": 1,
        "disponibilidad": 75,
        "icon":"\/icons\/ubicacio-0.png",
        "transition_start": "",
        "transition_end": ""
}
]
}
```

In detail, the application needs to get in real-time the information from https://www.bicing.barcelona/en/get-stations and provide web services to:

1. Order the bike-stations by available *“slots”* and display first N stations. The user can also choose to have the results in ascending or descending order (by default descending).
   The parameters for this web service are:
   - **N:** number of stations to display. It is optional and by default N=10.
   - **order:** It is optional and the default value is descending

2. Order the bike-stations by available *"bikes"* and display first N stations. The user can also choose to have the results in ascending or descending order (by default descending).
   The parameters for this web service are:
   - **N:** number of stations to display. It is optional and by default N=10.
   - **order:** It is optional and the default value is descending

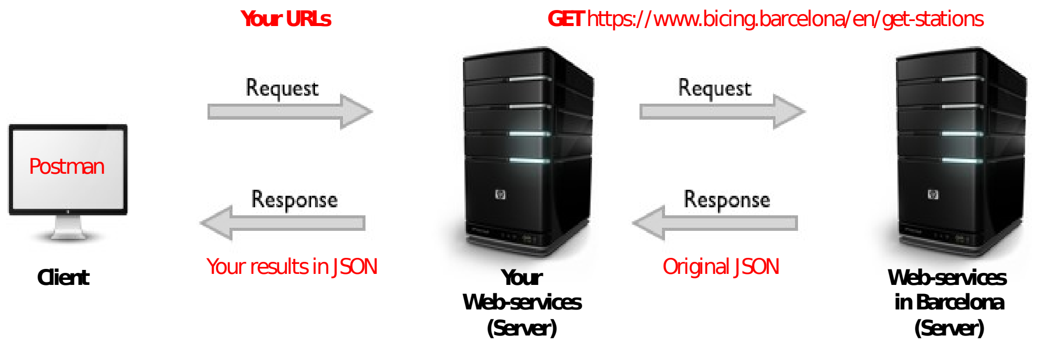
3. Get all the bike-stations with more than N available *"electrical_bikes"* and more than M free *"slots"*
   The parameter for this web service is:
   - **N:** number of available bikes. It is optional and by default N=10.
   - **M:** number of free slots. It is optional and by default M=5.

4. Count all the available *"bikes"* and all the free *"slots"* in the city.

Each web service must verify if the parameters are correct otherwise rise an HTTP error with the right HTTP status code.

All the Web Services responses should be in JSON. Validate it with http://jsonlint.com/

Architectural schema example


Your URLs
GET https://www.bicing.barcelona/en/get-stations
Request
Request
Postman
Response
Response
Your results in JSON
Original JSON
Client
Your Web-services (Server)
Web-services in Barcelona (Server)

SUGGESTION: use the library *requests* to open ULRs and read their contents (take care in handling exceptions)