

Programming for IoT applications

Lab 3

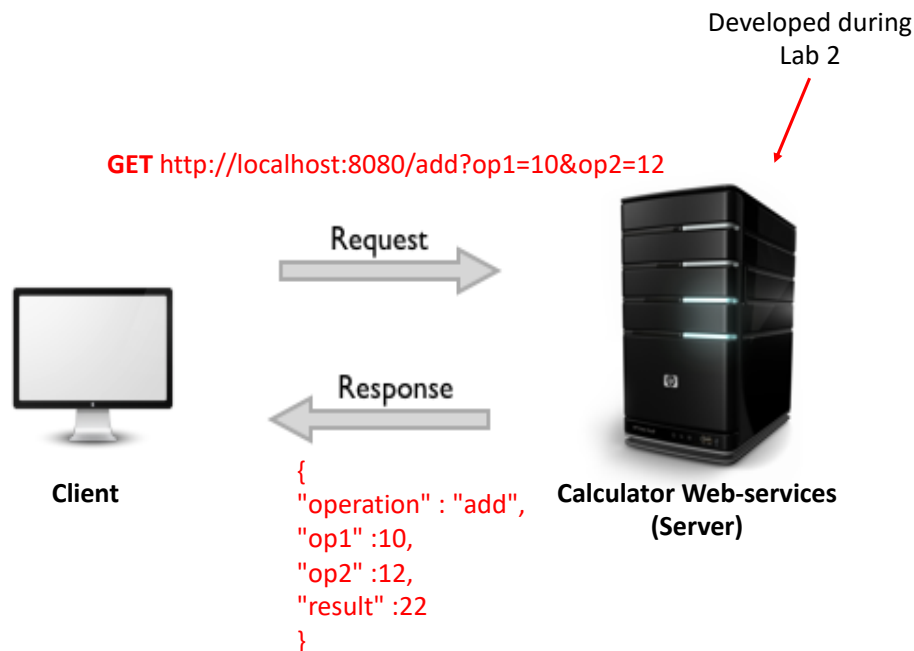
SUGGESTION: Use **Postman**, a chrome plugin, for testing REST web services by managing HTTP requests. In case you use **VSCode**, you can use **ThunderClient** for testing REST web services.

Exercise 1

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

- asks 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 URLs and read their contents (take care in handling exceptions)

Exercise 2

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

- **searchByName:** print all the information about the devices for the given <DeviceName>
- **searchByID:** print all the information about the devices for the given <DeviceID>
- **searchByService:** print all the information of the devices that provides the given <ServiceType>
- **searchByMeasureType:** print all the information about the device that provides such <measureType>
- **insertDevice:** insert a new device if it 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 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 file must follow the one of the ones that are already present.
- **printAll:** print the full catalog

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

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 are:

- **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, it must 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



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