01UDFOV/01TXYOV – WEB APPLICATIONS I

REST WITH EXPRESS

During this fifth lab, you will get started with the Express framework and build a series of REST APIs to support a web-based task manager.

EXERCISE 1 – DESIGN THE REST API

Design a set of REST API to support the main features of a task manager, which works like the one developed in the previous labs.

In particular, to recap, a **task** is composed by the following properties:

- *id*, a unique identifier for the task (mandatory)
- *description*, the textual task description (mandatory)
- *important*, whether the task is important or not (default to "not important", modelled as a boolean)
- *private* or *shared*, whether the task is private or shared with other users (default to "private"); consider this as a boolean value, as in the previous labs
- *project*, the project in which the task is inserted (optional, string)
- *deadline*, the due date (and/or hour) of the task (optional)
- *completed*, whether the task was completed or not (boolean, default to "false")

The API should allow an application to:

- Retrieve the list of all the available tasks
- Create a new task, by providing all relevant information – except the 'id'
- Retrieve a task, given its id
- Update an existing task, by providing all relevant information (all the properties except 'id' will overwrite the current properties of the existing task having that 'id')
- Delete an existing task, given its id

In addition, you may want an API to:

- Mark a task as "completed"
- Retrieve a list of all the tasks that fulfill a given property (e.g., all the important tasks, all the tasks with a given deadline, etc.); refer to the "filters" and "projects" functionality of the previous labs for a full list of options

Data passed to or received from the API should be in JSON format.

Open a text editor (VSCode is fine) and design there your API. Be sure to identify which are the resources you are representing. You might want to follow this structure for reporting each API:

```
[HTTP Method] [URL, with any parameter]
[One-liner about what this API is doing]
```

```
[Sample request, with body (if any)]
[Sample response, with body (if any)]
[Error responses, if any]
```

Hint: You are strongly encouraged to follow the design method presented in the lecture "REST API" (https://github.com/polito-WA1-2020/course-materials/blob/master/slide/3-03-REST.pdf) and inspired by the Google API Design Guide (https://cloud.google.com/apis/design/).

EXERCISE 2 – IMPLEMENT (AND TEST) THE API

Implement the designed REST API in Node.js with Express. The tasks are stored in an SQLite database, as shown in the example developed during the lectures (https://github.com/polito-WA1-2020/ex-express-fetch).

Test the realized API with a tool such as Postman (https://www.postman.com/downloads/) or RestClient for Firefox (https://addons.mozilla.org/en-US/firefox/addon/restclient/).

Hints:

1. To quick start this exercise, you can use the pre-filled SQLite database available on GitHub: https://github.com/polito-WA1-2020/lab5-rest-express
2. To visualize and edit the database, you can exploit the "DB Browser for SQLite" application, downloadable from https://sqlitebrowser.org/.