HTTP API with FastAPI

Activity data

Activity number	2
Professor	Ramon Amela Milian
Teaching language	English
Association	Individual
Delivery date	20-01-2025

Activity description

In this activity, HTTP methods will be worked on. The student will learn how to build an HTTP API using FastAPI. The starting point will be a minimal running FastAPI project, and some exercises will be done to achieve a simple but complete API. More precisely, we will develop the API for a storage file system.

On the one hand, an authentication API will be developed to register and authenticate users. At this point, an external identifier should be generated for each user. This external identifier should be an integer (and be unique for each one of the users). This external identifier should be stored next to files to identify the user to whom the file belongs to.

It is recommended to start developing from the result obtained in the first deliverable so that the result is better. Nevertheless, it is possible to start from the template, and all the missing pieces from the first deliverable will not be considered nor penalized. Even if continuing from the result of the first deliverable is preferred, it is recommended to look at the template to check if there is something that can be imported.

Competences

CE3 – Design and manage information systems for data treatment and storage

CT1 – Dominate the communication in several languages to express and understand messages in different contexts and personal, social, and professional situations.

COMPETENCES (CE / CT)	LEARNING RESULTS (RA)	EVALUATION INDICATORS (IE)
CE3	RA2 – Use the most standard techniques for the data processing and storage in cloud systems	IE3 – The student understands the principles of an HTTP API and can implement them
	RA5 – Justify the methods used to handle the data during its whole life cycle	IE6 – The student understands what a CRUD is and can implement it
CT1	RA7 – Communicate and debate a novel project in which coherent solutions are argued and critical thinking is demonstrated	IE11 – The student is able design an API and document its decisions

Activity contents

- HTTP methods
  - GET
  - POST
  - DELETE
- Use of headers
- Data hashing and encryption
- Use of HTTP frameworks
  - FastAPI

Task descriptions

- Create two apps called “authentication” and “files”
- Create routers inside both new applications
- Import the new routers from the main file

- Create the following endpoints in the newly created folders

In the following definitions, it is possible to identify the user with a string or with an email. Choose what you think is the best option.

It is important to note that always that the session token is passed as input, it should be sent through a header called "Auth".

- **Authentication**
  - register – POST

Input: at least, the minimum data that you consider necessary to code all the other endpoints.

Action: create a new user into the database

Output: code errors for the considered cases (for example, 200 for success, 409 when user already exists).

- login – POST

Input: user id and password.

Action: create a new session for this user

Output: connection token and code for success/errors

- logout – POST

Input: connection token

Action: close the session associated to the input token

Output: code for success/errors

- introspect – GET

Input: session token

Action: no action

Output: verify whether the token is correct or not and, in case it is, return the information associated to the user as well as the code for success/errors

- **Files**
  - files – GET

Input: session token

Action: no effect

Output: return the information for all the files owned by the user corresponding to the token passed in the call and the code for success/errors

  - files – POST

Input: file information and session token

Action: create a new file

Output: return the id for the created file and the code for success/errors

  - files/{id} – GET

Input: file id and session token

Action: no action

Output: return the information (and content if present) for the id given in the input and the code for success/errors

  - files/{id} – DELETE

Input: file id and session token

Action: delete the file with the id passed as parameter

Output: code for success/errors

  - files/{id} – POST

Input: file id, session token and file content

Action: fill in the file content

Output: code for success/errors

- files/merge – POST

Input: two file id, session token

Action: merge pdf's

Output: the id of the merged file and code for success/errors

It is important to note that there are two different endpoints to create files. One to upload the file information (like filename, description... and whatever information wants to be saved) and one to upload the file content.

In addition, it is also key to realize that all the common endpoints for authentication and files will be, respectively, in the same file. This makes it possible to store all the persistent information in local structures. This can be global dictionaries defined at the beginning of each one of the two files.

Finally, all endpoints must be correctly documented through swagger.

All the previous steps could be separated in the following summary points:

1. Implement and document an authentication API with all its basic components:
    a. Register
    b. Login
    c. Logout
    d. Introspect
2. Implement and document an API to handle files with basic components
    a. CRUD
    b. Complex functions

Formal aspects of tasks

All the project's code must be delivered in compressed format through the virtual campus. In addition, in the delivered text, the commit hash with the code in *GitHub* should be included. This hash should contain the same version as the delivered code.

Task	**Percentage**
1	60%
2	40%

Evaluation

Number	Task	Weight	IE
2	HTTP API with FastAPI	30%	IE3, IE6, IE11

Information resources

Capsules and videoconferences