OpenQCM API

Overview

Context

This project is carried out as part of *Datascientest's* Data Engineering training (Fev-Dec 2021). It takes part of the evaluation process for the module entitled *"FastAPI"* ,

Objective

For this evaluation, we consider a company creating quizzes via a smart-phone or web browser apps. The quizzes take the form of *Multiple Choice Question* (Questions à Choix Multiple in French, hence the name *OpenQCM*). The idea is to propose a service for individuals/companies to test themselves/their team's knowledge while having fun :D.

To optimize the architecture, the company wants to set up an API with the purpose of unifying the way to query a database and return a series of questions. The objective then is to create this API.

Data

The database is represented by a csv file that the api manipulate and stock at the current directory. The original version in available at :

https://dst-de.s3.eu-west-3.amazonaws.com/fastapi_fr/questions.csv

The database contains the following fields:

- ***question***: the title of the question
- ***subject***: the category of the question
- ***correct***: the list of correct answers
- ***use***: the type of *Multiple Choice Question (*MCQ) for which this question is used
- ***responseA***: answer A
- ***responseB***: answer B
- ***responseC***: response C
- ***responseD***: the answer D (if it exists)

The API

The API, referred to as ***OpenQCM*** from now on, is a simple HTTP REST API for technical quizzes including a wide variety of topics such as: *Automation, BDD, Classification, Data Science, Docker, Machine Learning, and lots more.*

Request parameters

A user can search for questions using the following pattern:

The user choose a type of test (use) and one or more categories (subject). The application produce MCQs of either 5, 10 or 20 questions (to be specified in user request, 5 being the default value). The API returns this number of questions in JSON format, c.f. Output format.

As the API must be able to generate many MCQs, these are returned in a random order: thus, a request with the same parameters may return different questions.

Authentication

Users are expected to have an account. Credentials verification is done using basic *username:password* authentication: the string containing Basic *username:password* is passed via the Authorization header (encoded/no)

EndPoints

OpenQCM has many endpoint, illustrated in the following:

Method	Path	Parameters	Details
GET	/status	None	Verify that the API is functional. (requires not identification)
GET	/users/me	None	Get the current user (requires identification)
GET	/subjects	None	List available subjects in the database (requires identification)
GET	/uses	None	List available uses in the database (requires identification)
GET	/qcm/	1- *Number* (type :integer) must be either 5, 10 or 20 *Default value* : 5 2- *Use* : (type string) Must be one of available uses from DB, e,g, - 'Test de positionnement', - 'Test de validation', - 'Total Bootcamp' *Default value* : 'Test de positionnement'	- Get a MCQ from database : - Containing 5,10 or 20 questions - Related to one or more subject (category) - Related to one use (e.g. *Test de positionnement*, *Test de validation*, etc) **c.f. Fig. 01**

		3- subjects: (type array[string]), could be one or more subjects availables from database.	c.f. Fig. 01
POST	/qcm/add/	- *question*, *subject*, *correct*, *use*, *responseA*, *responseB*, *responseC*, *responseD*,	- Create a new question (requires *admin* privileges). **c.f. Fig. 02**

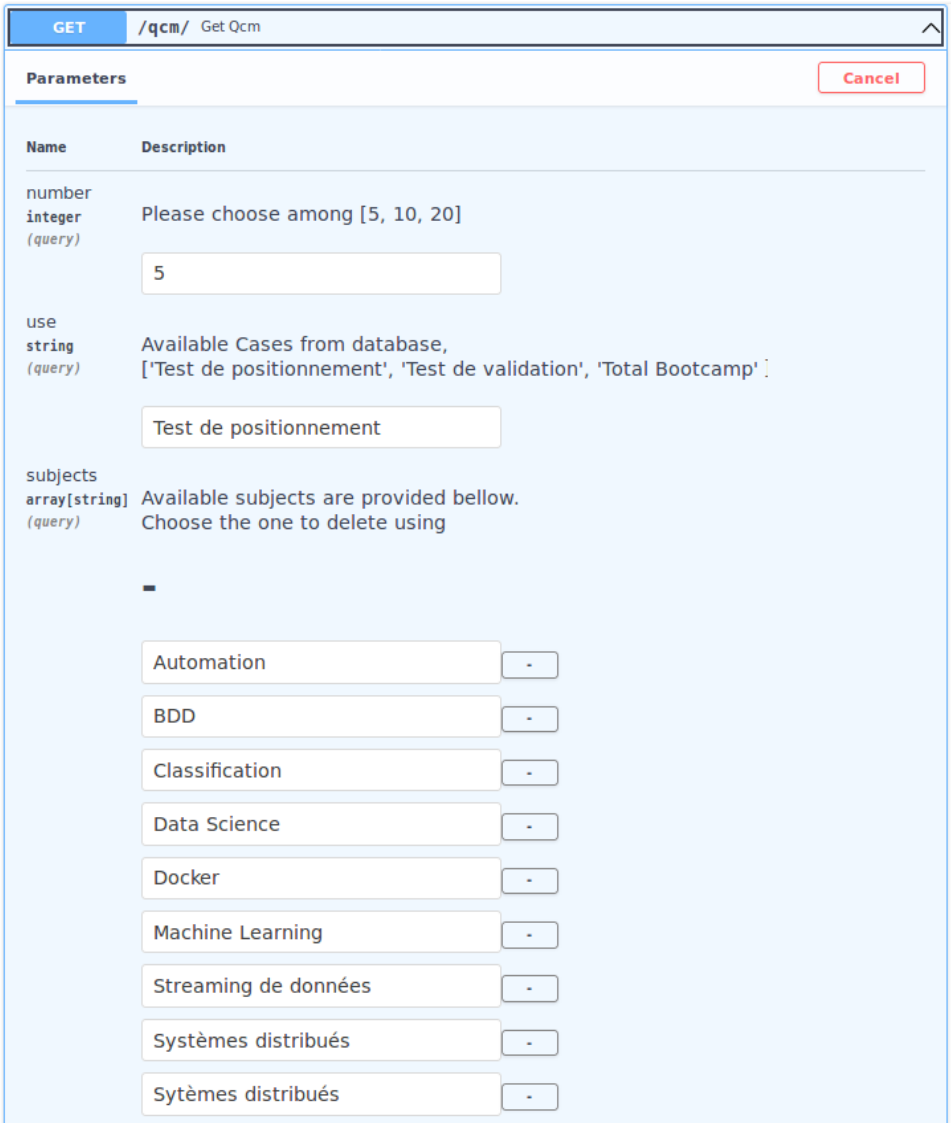

Fig. 01 : "Get QCM" endpoint, as illustrated in openAPI

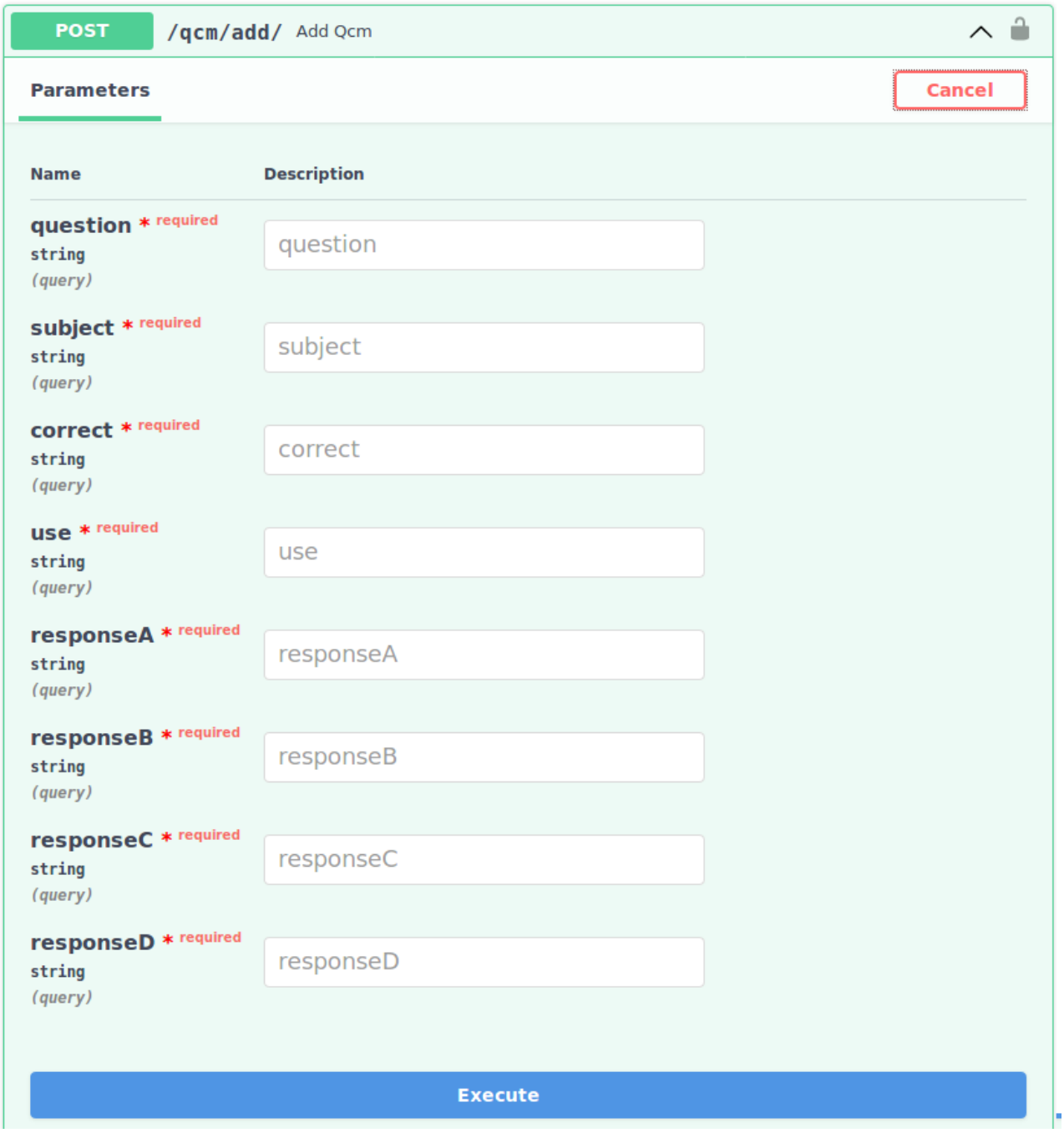

Fig. 2 : “Add QCM” endpoint as illustrated in openAPI

Examples

The following example will output 10 random questions.

Curl

```
curl -X 'GET' \
  'http://127.0.0.1:8000/qcm/?number=10&use=Test%20de
   %20positionnement&subjects=Data%20Science&subjects=Docker&subjects=Machine
   %20Learning&subjects=Syst%C3%A8mes%20distribu%C3%A9s' \
   -H 'accept: application/json'
```

Request URL

```
http://127.0.0.1:8000/qcm/?number=10&use=Test%20de
%20positionnement&subjects=Data%20Science&subjects=Docker&subjects=Machine
%20Learning&subjects=Syst%C3%A8mes%20distribu%C3%A9s
```

This returns a JSON object with the results in an array you can iterate over.


```
{
  "response_code": 0,
  "results": {
    "use": "Test de positionnement",
    "subject": [
      "Docker",
      "Machine Learning",
      "Systèmes distribués",
      "Data Science"
    ],
    "number": 10,
    "results": [
      "Le théorème CAP oppose",
      "Dans Hadoop, les partitioners permettent",
      "Hive permet",
      "Spark se différencie de Hadoop par",
      "Quels sont les trois éléments constitutifs de Hadoop ?",
      "Lors de l'étape de Map d'un wordcount appliqué à la phrase \"cette phrase
est une phrase\", les valeurs émises sont:",
      "Docker permet de persister des changements",
      "DockerHub est",
      "Des containers Docker peuvent communiquer entre eux grâce à",
      "Docker est utilisé"
    ]
  }
}
```


Response Headers

```
    content-length: 616
    content-type: application/json
    date: Thu,29 Jul 2021 15:28:08 GMT
    server: uvicorn
```