



T7 - MSc Pool

T-POO-700

API

Project





Respect carefully the subject nomenclature for schemas and endpoints.

First of all, Gotham City Hall wants to deploy the application on several platforms (mainly computers and different types of terminals). It is also considering a mobile version.

The goal of this project is to create a platform-independent API for exchanging and storing data.



Before you start, make sure you have **finished** and **assimilated all the concepts** covered in the Bootstrap.



The goal of this project is to get an API, and only an API. We do not want a front-end (HTML, JS).

Use the right flags when creating!

Here is the list of mandatory schemas for your application :

```
users = {
    "username": "string - required - can't be null",
    "email":    "string - required - can't be null - X@X.X",
}

clocks = {
    "time":      "datetime - required - can't be null",
    "status":    "boolean - required - true ( when clock'in) - can't be null",
    "user":      "employee - required - can't be null"
}

workingtimes = {
    "start":     "datetime - required - can't be null - YYYY-MM-DD HH:mm:ss",
    "end":       "datetime - required - can't be null - YYYY-MM-DD HH:mm:ss",
    "user":      "employee lie - required - can't be null"
}
```



Nothing fancy here, only obvious points. Obviously, you can add some more schemas and check the accuracy of your database after your migration.

Concerning the CRUD, here is a list of demanded endpoints

USER

- a GET method : `http://localhost:4000/api/users?email=XXX&username=YYY`
- a GET method : `http://localhost:4000/api/users/:userID`
- a POST method : `http://localhost:4000/api/users`
- a PUT method : `http://localhost:4000/api/users/:userID`
- a DELETE method : `http://localhost:4000/api/users/:userID`



WORKING TIME

- a GET (ALL) method : `http://localhost:4000/api/workingtimes/:userID?start=XXX?end=YYY`
- a GET (ONE) method : `http://localhost:4000/api/workingtimes/:userID/:workingtimeID`
- a POST method : `http://localhost:4000/api/workingtimes/:userID`
- a PUT method : `http://localhost:4000/api/workingtimes/:id`
- a DELETE method : `http://localhost:4000/api/workingtimes/:id`

CLOCKING

- a GET method : `http://localhost:4000/api/clocks/:userID`
- a POST method : `http://localhost:4000/api/clocks/:userID`



The `POST` endpoint of this part handles both the departure and the arrival of the users; think hard about your route!



Be rigorous! Check carefully the accuracy of your database after your migration, and test extensively your different endpoints with `Postman`