C1- Python

C-Python

Flask-D02

Going deeper

CODING ACADEMY
BY EPITECH

Foreword

Time To Play Again !?

Before anything this subject is the follow up of yesterday's.
You will have three days to finish up those two subjects.
So if you haven't completed step 6 yesterday, it's quite normal.
Before anything you must finish up yesterday's subject.

Good? Ok, now:

Did you enjoy the MVC Rush !? Well !!
Let's start again using Flask this time !

You'll have to implement article view and controller as well as comments mechanism.

Exercise 1

MVC MVC: 7 pts

File to hand in: Flask_D02/

So, when you remember the MVC RUSH, you are suppose to cry ...
But, this time, it must be easier, Flask is in da place.
You must implement this architecture:

Models/
    Model.py
Controllers/
    articlesController.py
    usersController.py
Config/
    db.py
app.py

There is only one model file in the model folder you need only 4 functions in that file, but it must be able to handle any kind of request. (POST, PUT, GET and DELETE)
We are still working API style so no templates should be rendered in this app, only json code should be returned, you may use your other app (client.py) to test it or use Postman.
The db.py file must contain every config for the database and all the functions linked to it (get_db, init_db, connect_db, ...).
app.py is still the entry point of your app. The return of the JSON data must be done by the app.py file.
usersController.py will contain every controllers linked to the User (trim input, check mail type, crypt password, etc ...).
articlesController.py will do the same for the Articles.

The Article table must be as such: id, title, body, creationDate, userId.
I'll let you guess the types.

It should be possible to delete, post, get(one and all) and delete an article.

Exercise 2

5 pts

File to hand in: Flask_D02/

You must be logged in to make an Article.
Only the owner of an article can delete or update it.

Exercise 3

4 pts

File to hand in: Flask_D02/

Let's take back your client app:
You need to make it look something like:

```
Controllers/
        articlesController.py
        usersController.py
Templates/
                register.html
                login.html
                ...
        Layouts/
                navbar.html
        js/
                ...
        Css/
                ...
        Img/
                ...
```

Yes there is controllers on the client side as well, why?
Because you should never trust a user.
This is what we call client side security.
And this the only way to deal with users.

Because if they can, they WILL fuck it up.
So WATCHOUT.
But also to be sure that you don't make API calls just for typos.
You still have to keep those on the server side because your users do not have to go throught your front app. Some of them might even now how to download Postman. Damn users, getting smarter and all.

So you need to have both.
And yes the things you are going to check are quite similar on both side.
Your controllers must handle all the HTTP Request done to your API.
The routing on the Client side doesn't have to be the same that on the server side.

Exercise 4

4 pts

File to hand in: Flask_D02/

Let's add some comments to this.

You'll need a commentsController fot both app.
They should be displayed below the article they are linked to.

Comments: id, message, creationDate, articleId, userId.

You must be logged in to make a comment.
Only the owner can update or delete it.

Bonus

10 pts

File to hand in: Flask_D02/

Replace the client side with REACT =D
Enjoy.