B2 - Introduction to Web Development

B-WEB-200

EPyTodo

A web ToDo app

1.3

{EPITECH.}

EPyTodo

group size: 2-3
repository name: epytodo_2017
repository rights: ramassage-tek
language: python 3

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).

Table of Contents

SUBJECT

EPyTodo is a project which you could rely on in the future.
Thanks to it, you'll handle all tasks which you need to do easily!
Into this project, you'll have to develop:

1. your MySQL database scheme
2. your web server using Flask
3. your HTML pages using Jinja2 (integrated with Flask)

The bootstrap will help you a lot !

+ MYSQL DATABASE

Into your database, you'll need to handle many users with their tasks.

All instructions have to be **strictly** followed

Create a file named epytodo.sql.
You will write into it all your database scheme.
The database name is **epytodo**.
Its tables are named :

1. user
2. task
3. user_has_task

Think about the last one : why do you need this ?
Maybe it has to do with relationship…

Here are all fields (with types or values) which you must have into your tables:

1. user table
    - user_id (mandatory not null)
    - username (mandatory not null)
    - password (mandatory not null)
    - etc.
2. task table
    - task_id (mandatory not null)
    - title (mandatory not null)
    - begin (optional value when creating a task, actual date by default)
    - end (optional value when creating a task, empty by default)
    - status (**not started** by default / **in progress** / **done**)
    - etc.
3. user_has_task table
    - fk_user_id
    - fk_task_id

Once your scheme is created, import your file into your MySQL server.

```
~/B-WEB-200> cat epytodo.sql | mysql -u root -p
```

Your sql file has to be placed at the root folder when turned in.
Do not insert any records into this file.

+ WEB SERVER

Files to turn in (refer to the bootstrap to know where placing them into your folders):

- run.py
- __init__.py
- models.py
- views.py
- controller.py

You server will implement a MVC architecture.

There's not only ONE way to implement it but it's **mandatory** to do so.

Look closely at schemes you can find.
Here's a TIP.
More explanations of what is attempted into each file:

- run.py : your entry program
- __init__.py : your app package file
- models.py : all objects / functions which will interact with your database
- views.py : all routes which are described into the API file
- controller.py : all interactions between your views and your models

We will add our config.py file.

All settings for your program (debug mode, database configuration, etc.) will be there!
Here are the required ones :

- DATABASE_NAME
- DATABASE_HOST
- DATABASE_SOCK
- DATABASE_USER
- DATABASE_PASS

Be sure that your config.py is similar and used effectively.

+ HTML PAGES

All data will transit into JSON format (see API file).
You are free to display all information as you want (plain text, lists, accordion, etc.).
This graphical part will be evaluated during your presentation.

BONUS

Do not integrate any extra features like allowing PUT or DELETE methods into your main delivery.
Do this into a **bonus** directory

Here is a minimal list of what you can do as a bonus:

- add a responsive design
- develop more functionnalities than expected:
  - Ajax requests
  - dynamic components (drag & drop cards, datepickers, etc.)
  - contacts system (adding friends, authorizations handling, etc.)
  - notifications
- implement a real login system (OAuth / SSO / LDAP / etc.)
- develop what's in your mind