Webforce3

Formation POO en Python

Projet de fin de séquence No. 2

Cahier des charges express

(version 2 du projet *alumni*)

Objectif

Mister M. veut toujours une application stockant des informations sur ses étudiants.
Cette application doit toujours permettre aux étudiants (ou à Mister M.) de gérer des fiches étudiants.

[option] Dans l'idéal, il aimerait toujours pouvoir saisir les notes obtenues par ses étudiants et obtenir quelques statistiques à leur sujet.

Planning prévisionnel

Lundi 4 : Description des fonctionnalités avec le format BDD (cf. plus loin)
Mardi 5 : Codage de l'appli en utilisant `sqlite3`
Mercredi 6 : Suite du codage de l'appli en utilisant `sqlite3`

(Mister M. entend parler de SQLAlchemy et il trouve trop cool d'utiliser un ORM...)

Jeudi 7 : Recodage de l'appli sans `sqlite3` mais avec SQLAlchemy *(le client est ch.. mais il est têtu et c'est lui qui paie...)*
Vendredi 8 à midi : Livraison de l'appli

Spécification fonctionnelles

Au lancement, l'application demande de ***s'authentifier ou de créer un compte***.
Un seul utilisateur est pré-enregistré, pour Mister M. (identifiant '`admin`' et mot de passe '`Admin42`').
Tous les comptes créés par la suite sont des "comptes étudiant" caractérisés par un identifiant et un mot de passe.

Remarque : Pour être accepté, un mot de passe doit comporter ***au moins 8 caractères, une minuscule, une majuscule et un chiffre***.
Dans un premier temps, il ne sera pas possible de réinitialiser un mot de passe.
Les mots de passe ne doivent pas être stockés en clair dans la base de données.

Un étudiant doit pouvoir ***créer sa fiche, la modifier et la supprimer***. Il doit également pouvoir consulter n'importe quelle autre fiche.
Mister M. doit pouvoir créer, consulter, modifier ou supprimer n'importe quelle fiche.

Contenu de la base de données

Les informations à enregistrer sont :

- les nom et **prénom**
- l'adresse (numéro, voie, code postal et **ville**)
- le numéro de téléphone
- le **mail**

- une photo (ou plutôt ***un chemin vers celle-ci***)

- le niveau estimé dans les différents sujets étudiés : Linux, Algorithmique, UML, HTML, CSS, JavaScript, Bootstrap, Python, Java, SQL (ce niveau sera évalué de 1 à 5 étoiles)

Les informations soulignées devront être renseignées pour permettre l'enregistrement (ou la modification) de la fiche.

Les données ne seront acceptées que si elles sont au format attendu. Par exemple XX-XX-XX-XX-XX pour un numéro de téléphone ou xxxxxxxxx@xxxxx.xxx pour un mail.

Spécification techniques

L'appli doit être codée en Python et posséder une interface graphique construite avec `tkinter`.
Les données doivent être stockées dans une base SQLite.

L'appli sera développée en* BDD (Behavior Driven Development) *en commençant par écrire les fonctionnalités selon le format* Given - When - Then (- And) *suivant :

> **Given** a user leaving a comment
> **When** the comment is over 1 000 characters
> **Then** the comment should not be saved
> **And** the user should see an error message

Le code sera structuré selon les principes de la Programmation Orientée Objet pour le rendre modulaire et pouvoir le faire évoluer plus aisément (ajout de fonctionnalités futures).

Classes, méthodes et fonctions seront chacune documentées par une* docstring*.

Critères d'évaluation

Aspects fonctionnels (9 pts)	Evaluation
L'appli fonctionne lorsqu'on la lance	
Les mots de passe sont stockés sous forme de hash dans la base de données	
C - On peut créer une fiche	
R - On peut consulter une fiche	
U - On peut modifier une fiche	
D - On peut supprimer une fiche	
Les fonctionnalités accessibles à l'administrateur / un étudiant correspondent à celles prévues par le cahier des charges	
Les entrées non valides de l'utilisateur sont gérées	
L'appli ne bogue à aucun moment	
L'appli a une fonctionnalité supplémentaire (2 points bonus)	
Design UI / UX (3 pts)	
L'appli a une interface graphique	
L'interface utilisateur est agréable à regarder	
L'interface utilisateur est agréable à utiliser	
Qualité du code (6 pts)	
Le code est en anglais (mais l'interface utilisateur est en français)	
Le nommage des variables, fonctions, classes, attributs et méthodes est pertinent	
Les classes sont pertinentes	
Les fonctions et méthodes font toutes moins de 20 lignes *(2 points bonus si elles font toutes moins de 10 lignes)*	
La PEP 8 est respectée	
La lisibilité globale du code est bonne	
Documentation du code (2 pts)	
Chaque fonction, méthode et classe possède une docstring	
Les docstrings sont éclairantes lorsqu'on les appelle avec `help()`	