Projet : site web avec BDD

Gros projet de ce trimestre : un site web dynamique avec une base de données derrière.

Le site sera codé en Python avec le module flask (pour la partie site web) et sqlite3 (pour la partie base de données).

Le projet est à faire par équipe de 2 ou 3, et sera évalué par une soutenance *individuelle* orale de 15 minutes (5 minutes de présentation, 10 minutes de questions). Il constituera une part importante de la note du 3e trimestre.

Date de rendu du projet : 9 mai. C'est la date de retour des vacances de Pâques, qui est la semaine de vos épreuves. Il est **fortement recommandé** de finir le projet avant les vacances !

Semaine des soutenances : Semaine du 16 au 20 mai.

La **Base de données** pourra contenir une ou plusieurs tables. Le site pourra proposer diverses choses, par exemple :

- Rechercher des données dans une (ou plusieurs) table, et afficher le ou les résultats,
- Ajouter une nouvelle entrée dans une table,
- Modifier et/ou supprimer une entrée d'une table

Le site proposera naturellement plusieurs pages web qui permettront ces différentes actions. Si vous avez plusieurs tables, et que vous n'avez codé que des actions sur une seule des deux tables, ce n'est pas gênant.

Le rendu HTML devra être valide (utiliser le validateur `http://validator.w3.org/`), et un minimum de CSS pour la mise en forme est fortement recommandé, mais c'est la programmation qui est au cœur de l'évaluation.

Le dossier que vous allez rendre (à vos noms) contiendra donc :

- Le ou les fichier(s) python qui génèrent les pages web
- Le fichier de base de données
- Le dossier `static` avec dedans vos fichiers statiques (CSS, images, scripts JS, . . .). Vous pouvez bien sûr organiser cet espace en plusieurs sous-dossiers.
- Si vous l'avez utilisé, le dossier `templates`
- Éventuellement d'autres dossiers si c'est pertinent dans votre cas.

Un espace *par équipe* (accessible en lecture-écriture) sera créé sur la plateforme monnuage pour y sauvegarder votre projet, et c'est là que vous le laisserez pour la deadline.

Conseils pour démarrer : commencez par réfléchir au schéma de votre base de données. Quelles sont vos données ? Quels attributs ? Quels sont les liens entre les différentes données ? Y a-t-il une table d'association ? Etc. Écrire soigneusement le schéma relationnel.

Vous pouvez faire des premiers essais avec sqlitebrowser afin de tester si la structure de votre BDD correspond à ce dont vous avez besoin.

Une fois que ce schéma est posé, réfléchissez à l'architecture de votre site. Quelles seront les différentes pages ? Pensez qu'une page avec un formulaire est en fait deux pages : une pour le formulaire, une pour l'« envoi » du formulaire (et la page "formulaire envoyé" par exemple).

Vous pouvez alors répartir les tâches entre les différents membres du groupe (différentes pages à écrire/coder ? Aspect HTML/CSS du site ? etc).

Outils :

- Toute la documentation SQL (kit de survie, TPs effectués)
- Documentation HTML fournie (1ere),
- Documentation CSS fournie (1ere),
- Fiche sur les formulaires et flask (1ere),
- Pour celles et ceux qui le souhaitent, le TP d'intro aux « templates ».

Remarques/aides diverses : FLASK : dans flask, tous les fichiers "normaux" (images, fichiers CSS, ...) doivent être mis dans un dossier « static ». Il faudra les inclure avec leur chemin, par exemple `<link rel="stylesheet" href="static/style.css" type="text/css">` dans votre code HTML pour include la feuille de style `style.css`.

Il semble que, parfois, sous Windows, le `app.run(debug=True)` ne fonctionne pas. Dans ce cas enlever le `debug=True` (mais vous aurez moins d'infos en cas de bug).

Si on met un `print` dans le code, le texte affiché s'affiche dans la console Python (et non sur la page web), ce qui peut être utile pour débugguer.

SQL : Pour avoir un incrément automatique d'un id dans une table, le mot-clé est `AUTOINCREMENT` (à la création de la table). Par exemple

```
CREATE TABLE usager (id INTEGER PRIMARY KEY AUTOINCREMENT, nom STRING)
```

Et en insérant une ou plusieurs valeurs, il suffirait d'insérer le nom, et l'id serait mis automatiquement.

Remarque : si flask est tout à fait adapté pour créer un « vrai » site web dynamique, sqlite n'est pas idéal pour ce genre d'application : c'est plutôt pour une base de données mono-utilisateur (vous avez remarqué qu'on ne pouvait ouvrir qu'une fois la base de données en écriture).

Il aurait fallu utiliser un SGBD de type client-serveur adapté à du multi-utilisateur, comme PostgreSQL, MySQL/MariaDB, ...

Mais dans ce cas la base de données est plus complexe à « déplacer » car ce n'est pas un simple fichier, donc on se contentera de sqlite pour ce projet. Si vous souhaitez un jour adapter votre projet « pour de vrai » vous pouvez regarder (par exemple) la documentation de mariadb pour python, la syntaxe est similaire (il n'y aurait pas beaucoup de choses à changer dans votre code).