Table of contents:

TP2 - Introduction Flask

But du TP

Le but du TP est de mettre en place un serveur BackEnd, avec différentes routes, qui seront sécurisées.
Les routes devrons vous permettre d'afficher du contenu.
Vous devrez également mettre en place un token afin de garantir une connexion sécurisée.
Dans ce TP vous ne prendrez pas en compte les failles XSS et injections SQL.
Pour les bouts de code je vous laisse aller voir le cours, tout est dedans !
Attention, si vous êtes sur Linux dans l'invité de commande ne vous connectez pas en tant que "root" sinon vous risquez d'avoir des soucis de droits.

Avant toute chose, sur le réseau de l'IUT vous risquez d'avoir quelques soucis liés au proxy, voici donc les 2 commandes à exécuter avant de commencer le TP :

```
export http_proxy=cache-etu.univ-artois.fr:3128
export https_proxy=cache-etu.univ-artois.fr:3128
```

Mise en place et configuration du serveur

Installation de Python et Flask

Avant toute chose, il faut installer Python ainsi que le package Flask.
Voici la commande Ubuntu à utiliser pour installer Python et pip :

```
sudo apt install python3 python3-pip
```

Ensuite il faut installer Flask, voici la commande à utiliser :

```
pip install Flask
```

> **INFO**
>
> Si vous avez un problème d'installation, il est possible que c'est le proxy qui bloque :
>
> ```
> pip install --proxy lien_du_proxy Flask
> ```

Création du serveur

Dans un premier temps vous allez créer un dossier sur votre PC, c'est ce dossier qui va contenir notre petit projet.

Ensuite nous allons créer notre fichier nommé "app.py", puis vous allez venir créer la base de votre serveur.

Je veux que le serveur tourne sur 127.0.0.1 et sur le port 5000.

N'oubliez pas de configurer vos variables d'environnement, sinon vous aurez une erreur et mettez en mode "développement" sinon vous travaillez en mode production.

```
# variables env
set FLASK_APP=votre_app.py
set FLASK_ENV=development

# sous linux
export FLASK_APP=votre_app.py
export FLASK_ENV=development
```

Configuration du serveur

La configuration pour ce TP est très simple, nous allons juste créer une simple route "/" afin de voir si tout fonctionne correctement.
Cette route sera la racine de notre serveur, à savoir : **http://localhost:5000/**

Créations des premières routes

Route about

Pour commencer vous allez créer votre route nommée “about”, je veux que lorsque j’arrive sur cette route un message s’affiche :

- “Bonjour je suis prénom”

Route hello

Ensuite vous allez devoir créer une route avec un paramètre, cette route devra s’appeler “hello” puis le paramètre sera le prénom qui sera affiché ainsi :

- “Bonjour à toi paramètre_prénom mon ami(e) !”

Une fois cela fait, merci d’appeler le professeur afin de valider le travail effectué.

API et Database

Installation d'un serveur mysql et configuration

Il faut tout d'abord installer un serveur mysql :

```
sudo apt install mysql-server
```

Ensuite il faut finir la configuration de l'installation :

```
sudo mysql_secure_installation
```

> ⚠ **ATTENTION**
>
> Durant la phrase de la **secure_installation**, s'il n'est pas demandé de choisir la sécurité du mot de passe, vous allez devoir réaliser la commande suivante :
>
> ```
> # Connexion en sudo mysql
> sudo mysql
>
> # On cherche les variables liées aux mots de passe
> ```

```
SHOW VARIABLES LIKE 'validate_password%';

# On modifie la variable pour ne pas avoir de contrainte sur le mot de passe
SET GLOBAL validate_password.policy=LOW;
```

Après avoir réalisé les actions ci-dessus, vous pouvez passer à la suite.

Après avoir réaliser l'installation il faut venir modifier le mot de passe utilisateur :

```
sudo mysql
```

Une fois dans le shell mysql, vous devez lancer la requête SQL suivante (en modifiant le mot de passe) :

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'password';
```

INFO

Il est possible que les changements ne soient pas pris en compte tout de suite, si c'est le cas alors nous allons lancer la commande pour "refresh" les privilèges :

```
FLUSH PRIVILEGES;
```

SUCCESS

Désormais pour vous connecter à votre serveur mysql il faudra utiliser la commande suivante :

```
mysql -u root -p
```

Installation package db, importation et configuration

Après avoir créer des routes, maintenant passons à l’étape suivante : MySQL
Vous disposez d’un accès à une DB, si ce n’est pas le cas installez un logiciel permettant

d’avoir accès à une db.
Pour installer le package il vous faudra sûrement rajouter et/ou modifier le proxy, pour cela vous devez utiliser une commande que vous avez utilisé dans un autre TP.

Installez le package Python pour MySQL :

```
pip install mysql-connector-python

# si soucis de version
pip3 install mysql-connector-python
```

Puis importez le dans votre application :

```
import mysql.connector as MC
```

Ensuite je vous laisse chercher dans le cours pour créer la connexion à votre db. Attention celle-ci devra être dans un **try except**.

Création de notre db

Vous allez devoir créer une db et une table nommée “users” avec cette structure :

- id : int auto_increment primary key
- pseudo : varchar(25)
- password : varchar(255)

Si vous souhaitez vous connectez à mysql via un invité de commande voici la commande :

```
mysql -u root -p
```

Donc soit vous le faîtes à la main soit vous téléchargez le fichier “users.sql” disponible dans moodle à la section du TP 2.

Création de la database et la table

Avant toute chose il faut créer la database (où sera stockée la table) :

```

```

```sql
CREATE DATABASE db_tp2;
```

Ensuite il faut exécuter le fichier SQL pour créer la table :

```
mysql -u root -p db_tp2 < users.sql
```

Et ensuite il faut vérifier si tout est bon :

```sql
USE db_tp2;
SHOW TABLES;
```

Vous pouvez même regarder si votre table 'users' contient bien les données :

```sql
SELECT * FROM users;
```

Afficher les utilisateurs

En utilisant le cours sur Flask, créez une route nommée “users”, cette route devra afficher la liste des utilisateurs (uniquement le pseudo), l’affichage devra être au format JSON.

Vous allez donc devoir utiliser un logiciel permettant de réaliser des requêtes API REST (Insomnia, Postman, etc...).

Afficher un utilisateur

Objectif suivant : Créer une route permettant d’afficher l’ID et le pseudo d’un utilisateur en utilisant son pseudo comme paramètre, si aucun résultat alors j’affiche un message d’erreur.
Exemple : http://localhost:5000/user/maxime → not found
Exemple : http://localhost:5000/user/user1 → id : 1, pseudo : user1
Résultat au format JSON et nom de la route “user”.

Insertion d'un utilisateur

En utilisant le cours sur Flask et sur le SQL, votre objectif est le suivant :

- Créer une route permettant de créer un utilisateur dans la table “users”, le pseudo et le mot de passe devrons être des paramètres de l’api.
- Les 2 paramètres devront être obligatoires, si ce n’est pas le cas alors retournez un message d’erreur.

Conseils :

- Le type de la requête sera POST et non du GET.
- SQL : utiliser INSERT INTO

Une fois cela fait, merci d’appeler le professeur afin de valider le travail effectué.

Sécurisation d'une API

Configuration du package

Maintenant vous devez mettre en place une route permettant de générer un token.
Installation du package JWT :

```
pip install PyJWT

# si soucis de version
pip3 install PyJWT
```

Importation du package :

```
import jwt
```

Pour cela créez une route nommée “login”.
Dans le cours vous avez un exemple afin de créer un token, je vous laisse donc allez jeter un oeil sur le cours, reprendre le code est l’adapter au TP.

Exigences :

- Le token devra contenir le pseudo de la personne connectée
- Durée du vie : 30min

Une fois cela fait, merci d’appeler le professeur afin de valider le travail effectué.

Sécuriser l'API

Reprenez la route “users”, je veux que désormais pour exécuter cette route, il faudra avoir un token valide.
Il existe un exemple dans le cours sur Flask à vous de reprendre le code et l’adapter.
Si je ne suis pas connecté avec un token valable alors je veux un message interdisant l’accès.

Data du token

Maintenant créez une route sécurisée (sans paramètre), qui affiche le message suivant :

- “Tu es connecté(e) sous le pseudo de : pseudo”

Conseil : Vous allez devoir décoder le token afin de récupérer une information

Une fois cela fait, merci d’appeler le professeur afin de valider le travail effectué.

Partie Bonus

Créez une route sécurisée par le token, qui permet de modifier le mot de passe d’un utilisateur, le mot de passe devra répondre à ces critères :

- longueur min : ≥ 6
- longueur max : ≤ 10
- doit contenir : 1 majuscule, 1 minuscule et 1 chiffre au minimum
- ne devra être le même que celui actuellement en cours de modification

Si un critère n’est pas respecter alors retournez un message d’erreur.

Une fois cela fait, merci d’appeler le professeur afin de valider le travail effectué.