# EXERCICE DE PROGRAMMATION PYTHON / JS

## DOCUMENTATION

07/11/2021

Réalisé par RAKOTOARISOA Van

# Informations

## 1) Technologie utilisée :

L'application web est codée en Python / JS côté serveur et côté front HTML5 / CSS3 native.
La base de données utilisé est SQLite3.
Aucun Framework côté JS n'a été utilisé mais comme l'exige l'énoncé, un des deux framework web pour Python a été utilisé (**Flask** en version **1.1**)

## 2) Contexte

Une « vulnérabilité » est représentée par 4 informations :

- son nom, un champ textuel ;
- son impact défini sur une échelle de 1 à 4 ;
- sa facilité d'exploitation définie sur une échelle de 1 à 4 ;
- son risque défini sur une échelle de 1 à 4 qui est calculé en fonction de l'impact et de la facilité d'exploitation.

On souhaite retranscrire ces informations dans un tableau et les enregistrer dans la base de données pour ensuite les afficher.

## 3) Fonctionnement

Pour mener à bien l'exercice, on utilise l'architecture MVC (model view controller) pour une meilleure répartition du travail avec des rôles différents.

3.1) Partie Controller

- Le fichier en python (app.py) est le controller qui agit comme un carrefour en fonction des différentes actions, requêtes qu'il recevra et les mènera chacun vers leur propre destination. Dans notre cas, par défaut, le controller pointe vers la page d'accueil ('/') si aucun paramètre n'est passé dans l'URL. Il affiche la vue HTML et stocke la liste des vulnérabilités dans une variable.

3.2) Partie HTML

- La partie interface (HTML) affiche son contenu et récupère également la liste des vulnérabilités que le controller lui a envoyé. Dans le corps du fichier HTML on effectue une itération sur cette liste pour l'afficher sous forme de tableau compréhensible par l'humain.

    - Cette partie contient le tableau et le formulaire permettant de créer une vulnérabilité. Il n'y a pas de balise form mais un simple button associé à une fonction javascript qui s'enclenche dès son utilisation.

        - Cette fonction récupère les valeurs saisis par l'utilisateur (nom, impact et exploitation) puis effectue une vérification si tout est renseigné. Ensuite vient le calcul du risque.
        - Si cette phase de vérification a été correcte, alors la fonction ajoute ce nouvel élément dans le tableau sans recharger la page (Ajax).

Si aucune erreur n'a été détectée, l'Ajax contenu dans la fonction va renvoyer les informations au controller POST /api/vulnerabilities qui lui se chargera d'exécuter une requête permettant d'enregistrer cette nouvelle saisie dans la base de données SQLite3.