université
PARIS-SACLAY
Langages Dynamiques
Master 1 ISD

Projet

1 Cahier des charges

On souhaite réaliser une interface graphique de *reporting* permettant de faire un rendu graphique synthétique de statistiques sur un ensemble de *tweets* (i.e. messages publiés sur la plateforme Twitter™).

L'application demandée doit être une application client/serveur, les deux parties interagissant via le protocole HTTP. Elle doit respecter les contraintes suivantes :

1.1 Serveur

1.1.1 Serveur Web

Le serveur Web doit être réalisé entièrement en Python. Il doit être capable de :
- servir des fichiers
- gérer des sessions HTTP
- reconnaître certaines URL spéciales et déclancher l'exécution de code Python arbitraire lorsqu'on navigue vers ses URLs
- mettre à disposition du code Python exécuté un objet représentant la session HTTP ainsi que les paramètres de la requête HTTP (la partie de l'URL de la forme : `?param1=v1&param2=v2&...`).

La bibliothèque standard Python fournit une classe SimpleHTTPServer qui permet de démarer un serveur HTTP sur un port donné et de servir les fichiers du répertoire courant. On pourra se servir de cette classe comme d'une base à étendre avec les nouvelles fonctionnalités demandées.

1.1.2 Requêtes sur les Tweets

Le moteur de requêtes doit être réalisé entièrement en Python en utilisant Panda. Votre code doit être capable de :
- importer un fichier CSV contenant les tweets
- effectuer des requêtes sur les tweets

1.2 Client

Le client consiste en un ensemble de pages HTML et fichiers Javascript. L'interface graphique doit proposer une barre de recherche, permettant **au minimum** de rechercher tous les tweets dont le corps du message contient une certaine chaîne de caractères. L'interface graphique affiche ensuite un *résumé* graphique de la recherche en indiquant :
- Le nombre de tweets trouvé
- Leur répartition par pays (sous forme d'un graphique « camembert » ou histogramme)
- Les *hashtags* associés à ces tweets (sous forme d'un graphique)
- ...

Le client doit être implémenté entièrement en pur CSS/HTML/Javascript et faire des requêtes *asynchrones* au serveur (XmlHTTPRequest).

2 Modalités

Le projet est à faire en binôme. Les monômes ne sont acceptés que s'ils ne laissent personne seul contre son gré. Les trinômes ne sont pas acceptés.

Le rendu consistera en un rapport et un rendu du code. Le rapport (entre 5 et 10 pages) doit détailler des choix de conceptions et d'implémentation (organisation du code, exemples judicieux de code complexe, structures de données utilisées, bugs subtils rencontrés) et **la repartition du travail au sein du binôme**. Il est *obligatoire* que chacun des membres réalise une partie du code Javascript (client) et une partie du code Python (serveur).

La date de rendu est le dimanche 3 mai, 20h00.

Il est interdit d'utiliser un *framework* que ce soit pour la partie client ou la partie serveur (i.e. pas de Django, Jquery, Angular, . . .). Il est suggéré de se répartir le code de la manière suivante :

Membre 1 du binôme Serveur Web en Python, affichage en Javascript

Membre 2 du binôme Partie requête en Python, et connexion asynchrones en Javascript

vous pourrez bien sûr ré-équilibrer un peu en fonction de l'évolution du projet (i.e. si la partie graphique s'avère plus conséquente, le membre 2 peut aussi y participer).

Barème indicatif

- rapport : 4 points
- organisation et lisibilité du code, tests et scripts : 6 points
- Serveur Web simple ne gérant pas de requêtes simultanées de plusieurs clients, affichage basique et *timeout* si le serveur ne répond pas assez vite : 2 points
- Gestion des sessions et d'utilisateurs multiples côté serveur (donc utlisation du multithreading dans la partie données) : 2 points
- Recherche avancée dans le client sur d'autres critères que les critères minimaux demandés (2 points)
- Absence de *timeout* : le serveur enregistre une requête du client, calcule le résultat et publie à une URL particulière les résultats quand ils sont disponibltes. Le client charge périodiquement cette URL tant que les résultats ne sont pas disponibles : 2 points
- Déploiement aisé : 1 point (installation simple, fichier README, . . .).