PROGRES

Rapport du Mini Projet 2

26/11/2021

Réalisé par :

- **Aouam Yasmina 28718479**
- **Adrien Koumgang T. 21109170**

Rapport Mini Projet 2

Le but de ce Mini-Projet 2, en premier lieu est de travailler sur la bibliothèque Python Bottle pour proposer une API pour le site http :dbpl.uni-trier.de/db/ht/ sur lequel nous trouvons les publications scientifiques en informatique

Par la suite, le but est de réaliser un serveur web qui se sert de cette API Web développée auparavant, sous forme de formulaire pour le client qui souhaite effectuer des recherches.

Par soucis de lecture du fichier (3Go) à cause de sa taille importante. Nous avons essayé une manière d'extraire tout le contenu des 5 dernières années et les écrire sur un nouveau fichier, mais rien n'y fait, nous avons toujours eu ce même soucis. Par manque de temps nous n'avons pas pu résoudre le soucis en question. Nous nous somme donc appuyé sur le fichier dbpl_2020_2021.xml.

Nous avons également eu quelques soucis sur certaines test lors de la partie deux que nous n'avions pas pû résoudre par manque de temps.

Exercice 1 : Mise à disposition d'une API Web

Nous avons préalablement définie des fonctions à intégrer dans les @routes

def get_authors_info(name: str) -> dict: Cette fonction prend en argument le nom d'un auteur sous forme de chaines de caractères, et retourne le résultat sous forme d'un dictionnaire.

Nous avons parcouru toutes les balises child de root, pour identifier toutes les publications liées au nom de l'auteur passé à l'argument <name>. Même chose pour identifier tous les co-auteurs liés à l'auteur recherché. Nous regroupons les résultats, chacune des publications dans une liste et chacun des co-auteurs dans une liste.

Avec set appliqué au résultats mis dans les listes, nous évitons la répétition de chaque élément de la liste. Le résultat final est un dictionnaire qui indique la longueur, en d'autres termes le nombre de publications et le nombre de co-auteurs.

def get_publications_author(name: str, start: int = 0, count: int = 100) -> list:

Cette fonctionne fonction de la même manière que pour la fonction qui permet d'obtenir le nombre de publication et de co-auteurs.

A chaque fois que nous ajoutons les publications d'un auteur recherché dans une liste, nous faisons en sorte qu'elles rentrent dans l'intervalle délimité par les paramètres start et count, ensuite nous arrêtons de stocker à un seuil délimité selon le count (nombre de publications à afficher).

def get_coauthors_author(name: str, start: int = 0, count: int = 100) -> list:

Cette fonction permet d'obtenir la liste des co-auteurs associé à l'auteur recherché, de la même manière que précédemment, nous parcourons les child pour trouver les co-auteurs ayant travaillé sur les mêmes publications qu'avec l'auteur passé en paramètre <name>.

En effet les paramètres start et count ont la même fonction que précédemment.

Nous avons utilisé une liste pour ranger le résultat des co-auteurs.

def get_search_substring_author(search_string: str, start: int = 0, count: int = 100) -> list:

La fonction prend en paramètre une chaîne de caractères et renvoie par défaut les 100 premiers auteurs dont le nom contient "searchString" sous la forme d'une chaîne de caractères.

Ici le raisonnement est le même que les précédents.

Le résultat est stocké dans une liste.

def get_distance(name_origin: str, name_destination: str) -> int:

Fonction qui permet de calculer la distance de collaboration entre les deux auteurs.

Il faut parcourir un bon nombre de fois le fichier xml afin de trouver une distance minimale. Nous avons rien pu faire pour optimiser la rapidité. Lorsqu'il y a un nombre important de co-auteurs d'un auteur est plutôt important, le programme mets un certain temps à s'exécuter.

Exercice 2 : Test unitaire d'une API Web

Dans cette partie, nous avons testé les valeurs possibles de start et count, on a testé les noms d'auteurs qui ne sont pas valide, ainsi que ceux qui sont valides.

Exercice 3 : Site Web d'utilisation d'une API Web

Le but de cet exercice est d'associer à l'API Web développé, un serveur Web, pour proposer une interface Web graphique qui lui permet de saisir les recherches sous forme d'un formulaire.

Nous avons connecté ce serveur sur le port 8080, quant à l'Api Web, nous l'avons connecté au port 8081.

Dans cet exercice pour rendre le code plus performant nous avons crée un fichier où nous avons réunies des Templates pour chacune des routes.

@bottle.get(''/'') : nous avons définis une route primaire, pour voir un message de bienvenue en lançant juste le site http://localhost:8084/

- **@bottle.get("/authors/infos")**

  def info_authors(): cette fonction permet à l'utilisateur de chercher la liste complète des publications d'un auteur et de ses co-auteurs en saisissant une chaine de caractère et renvoie donc à @bottle.post('/authors/infos ').

- **@bottle.post("/authors/infos")**

  def get_info_authors(): La function requests.get permet de demander une requête web qui permet d'exectuer la fonction associé à '/authors/<name>/' définie dans l'Api web, la réponse est recupéré et est affiché avec le template « author_infos.tpl ».

- **@bottle.post("/authors/publications")**

  def get_publications_author(): Cette fonction permet d'afficher la liste des publications de l'auteur demandé.

- **@bottle.post("/authors/coauthors")**

def get_coauthors_author(): Cette fonction permet d'afficher la liste des co auteurs de l'auteur demandé.

- **@bottle.post("/authors/search")**

def get_search_author(): Cette fonction permet d'afficher les auteurs contenant la ou les chaînes de caractère demandé

- **@bottle.post("/authors/distance")**

def get_authors_distance(): Cette function affiche la réponse à la requête de chercher de distance entre deux auteurs.