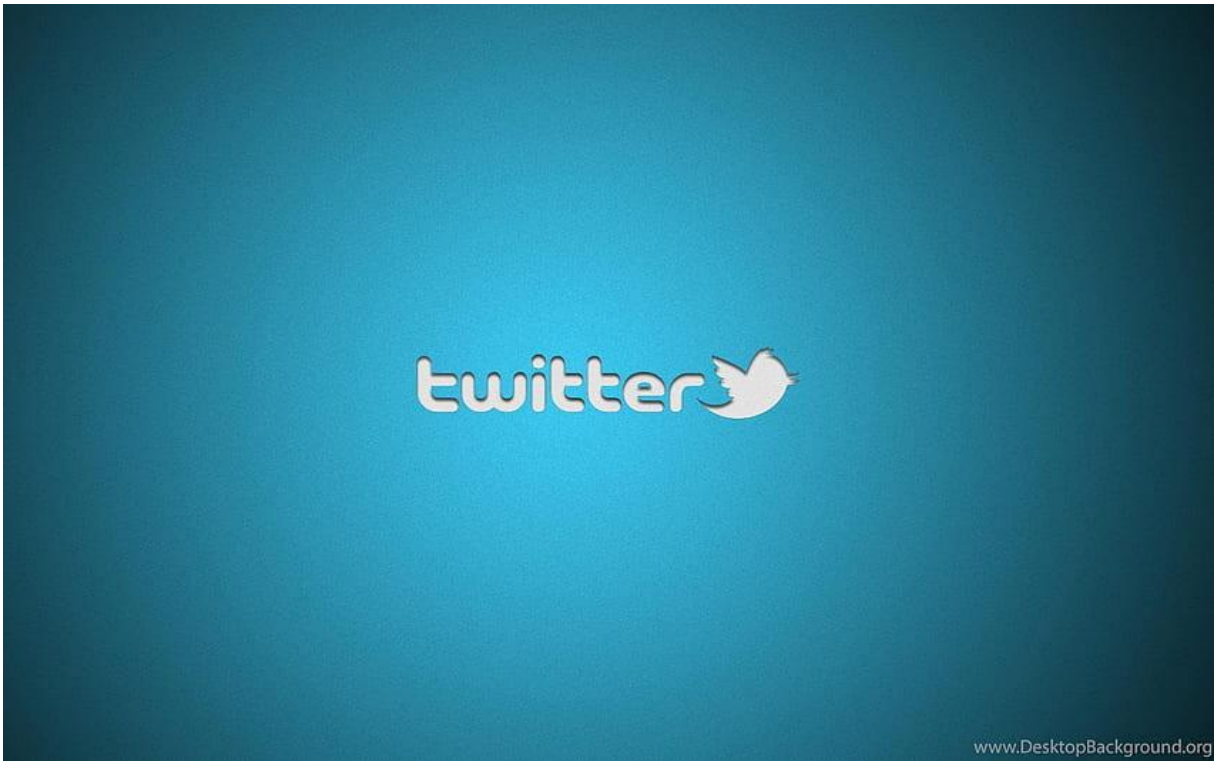
PROJET CLOUD COMPUTING

README PAETIE BACK-END



Réaliser par : Sami EZZAHID 4A-IE-ILC

Introduction :

Le projet consiste à recréer Twitter sous forme de microservices.

Le backend sera réalisé sous forme d'API qui permettra les fonctionnalités suivantes :

- *afficher tous les tweets*
- *enregistrer les tweets dans Redis*
- *afficher les tweets liés à une personne*
- *afficher les sujets*
- *afficher les tweets liés à un sujet.*

Les données seront stockées dans Redis sous forme de dictionnaire. La base de données contiendra deux ensembles de clés : l'un pour les tweets et l'autre pour les utilisateurs.

Enfin, le frontend sera réalisé avec HTML/CSS/JS et communiquera avec l'API pour appeler les différentes routes codée avec **PYTHON** et le framework **FLASK** . en plus de deux Dockerfile permettront le lancement du backend et du frontend .

Le répertoire de ce projet est constituer principalement de deux parties :

Partie Api : programme des différents routes de l'API.

Partie Interface : les différents fichiers HTML qui définissent l'interface homme machine.

Partie API :

J'ai programmé les différents routes qui définissent l'api avec Python et le framework Flask. Ces routes qu'on peut classer dans deux catégories :

- Routes pour assurer la navigation entre les différents fichiers/pages HTML grâce à la commande

>> window.location.href ="nom_fichier_ciblé"

Commande qui envoie une requête GET au fichier de l'API. Le fichier de l'API ; que j'ai appelé 'app.py' ; à son rôle, ouvre la page demander grâce à la commande

>> render_template()

- Route contenant des fonctions qui traitent les requtes envoyer par l'interface, selon la tache demandée.

Principe de fonctionnement :

L'architecture que j'ai adopter pour ce logiciel c'est que tout d'abord l'utilisateur est demandé de renseigner un nom d'utilisateur qui sera enregistre localement et qui vas servir comme clé pour plusieurs données créer lors de l'utilisation. Ensuite, l'utilisateur est rediriger automatique ment vers la page d'accueil ou la page des tweet où il peut saisir et enregistrer ses tweets. Après chaque tweet, le programme affiche une petite fenêtre pour confirmer l'enregistrement du tweet.

En plus, l'utilisateur peut choisir de voir la totalité des tweet enregistrées parmi les différents utilisateurs, afficher les tweet par personne ou afficher les tweet contenant un hashtag spécifique, ou bien afficher la liste des hashtages utilisées

Les fonctions de l'API :

1) save_tweet()

Il s'agit d'une route Flask nommée "/tweet" qui permet de créer un tweet en enregistrant les données fournies dans une base de données. La méthode de requête HTTP utilisée est POST. Les données requises pour créer un tweet sont le nom d'utilisateur de l'auteur et le tweet lui-même, tandis que le sujet est facultatif. La date et l'heure actuelles sont également enregistrées. Enfin, un message JSON est renvoyé pour confirmer que le tweet a été créé avec succès.

2) get_all_tweets()

Ce code définit une route sur une application Flask qui renvoie tous les tweets stockés dans une base de données Redis au format JSON. Il utilise la méthode HTTP GET pour récupérer les tweets. Il parcourt tous les éléments clés de la base de données Redis en utilisant la méthode "keys()" et pour chaque clé, il récupère la valeur associée (un tweet) à l'aide de la méthode "get()". Ensuite, il vérifie si le tweet est présent et le convertit en dictionnaire Python à l'aide de la bibliothèque "json" si c'est le cas. Enfin, il ajoute tous les tweets sous forme de dictionnaire à une liste et renvoie cette liste sous forme de réponse JSON.

3) get_tweets_by_username(username)

Cette route permet d'obtenir une liste de tweets pour un nom d'utilisateur donné en tant que paramètre d'URL. Les tweets sont stockés sous forme de dictionnaires dans une liste appelée "tweets". Les timestamps des tweets pour l'utilisateur sont obtenus à partir d'une base de données nommée "db_user" en utilisant la méthode "lrange". Ensuite, chaque timestamp est utilisé pour obtenir le dictionnaire de tweet à partir d'une autre base de données appelée "db_tweet". Enfin, la liste de dictionnaires de tweets est retournée sous forme de réponse JSON.

4) get_tweets_by_hashtag(hashtag)

Cette route est une endpoint pour une API Flask qui prend en paramètre une chaîne de caractères appelée hashtag et qui retourne une liste de tweets associés à ce hashtag. Elle utilise une boucle pour parcourir une base de données (db_tweet), extraire les tweets qui contiennent le hashtag dans leur champ "sujet", puis les ajouter à une liste de tweets.

5) Get_hashtag()

Ce code est une fonction Python qui utilise le framework Flask pour créer une route HTTP qui répond aux requêtes HTTP GET.

La fonction "get_hashtags()" utilise une base de données "db_tweet" pour scanner tous les éléments et extraire les données des tweets (stockées en format JSON). Si les tweets contiennent un champ "sujet", les hashtags correspondants (qui sont des mots précédés d'un symbole "#") sont extraits et stockés dans un ensemble (une collection d'éléments uniques).