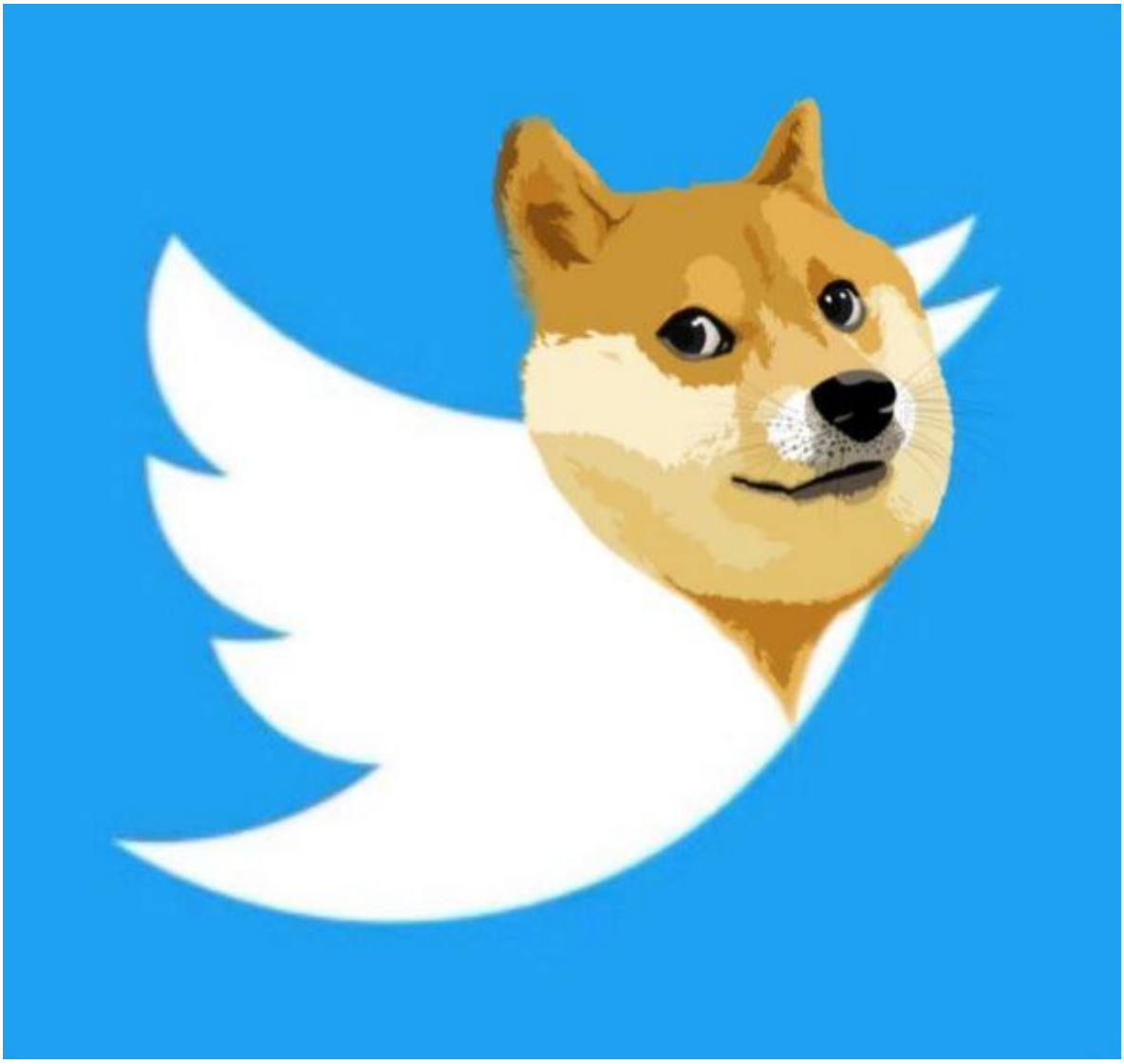
PROJET CLOUD COMPUTING

README PAETIE FRONT-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 constitué 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.

Remarque : je n'ai pas mis le code javascript ou CSS dans des fichier séparés car ça demande de renseigner des nouvelles routes dans l'API pour ouvrir ces fichier ce qui va compliquer le programme . Donc, j'ai préféré d'utiliser les balises 'style' et 'script' dans les fichiers HTML.

Partie interface :

L’interface de ce logiciel est constituer à travers six fichiers HTML.

1) Page de connexion (page_1)

Le corps de la page contient la barre de navigation et le formulaire de connexion, qui comprend un champ de texte pour le nom d'utilisateur et un bouton de soumission.

La partie JavaScript fournie ici est exécutée lorsqu'un utilisateur clique sur le bouton de soumission. La fonction "submitForm()" récupère la valeur entrée dans le champ de texte et stocke cette valeur dans le stockage local de l'utilisateur. Ensuite, la fonction redirige l'utilisateur vers une autre page ("home") en utilisant la propriété "window.location.href".

2) Page d’accueil (page de tweet)

Le script JavaScript utilisé récupère le nom d'utilisateur stocké dans le stockage local, enregistre un nouveau tweet et navigue vers différentes pages du site en utilisant des événements déclenchés par des clics sur les boutons de la barre de navigation. La fonction "saveTweet()" est appelée lorsque l'utilisateur soumet un nouveau tweet et utilise l'API Fetch pour envoyer les données du tweet au serveur via une requête POST.

3) Show all tweets

Le code JavaScript est utilisé pour ajouter des fonctionnalités interactives à la page, notamment pour récupérer la liste des hashtags à partir de l'API Flask en utilisant l'objet fetch() et afficher les résultats dans la page HTML en créant des éléments HTML avec createElement() et en les ajoutant à la page avec appendChild().

Dans cette page html les données sont afficher dans une table qui est défini dans le code HTML avec l'ID "tweets-table". Il est configuré pour avoir une bordure avec une épaisseur de 1 pixel et une couleur grise claire (#ddd). Les cellules du tableau sont alignées à gauche et ont un remplissage de 8 pixels. Les

cellules de la ligne impaire ont un arrière-plan gris très clair (#f2f2f2) pour faciliter la lecture des lignes. La première ligne du tableau est configurée avec un arrière-plan de couleur bleue foncé (#315B7D) et une police blanche pour permettre de la distinguer du reste des lignes.

Le style de la table est défini dans le bloc de style CSS dans l'en-tête HTML du document. Le bloc de style CSS contient également deux autres styles : un pour le conteneur de formulaire "tweet-container" et un pour les boutons de navigation ".nav-btn". Le style "tweet-container" définit les propriétés flexbox qui alignent le contenu du formulaire au centre de la page et ajustent sa hauteur à 100 % de la hauteur de la vue. Le style ".nav-btn" définit les propriétés de style pour les boutons de navigation, notamment la couleur de fond, la couleur de la police, le remplissage, la marge et le rayon de bordure.

4) Afficher les tweets par personne

Ce fichier HTML contient une table pour afficher les tweets. La partie CSS définit les styles pour le centrage du formulaire, les boutons de la barre de navigation et la table des tweets.

La partie JavaScript gère la recherche par nom d'utilisateur et l'affichage des tweets correspondants. Elle utilise la méthode fetch pour récupérer les données des tweets depuis le serveur et utilise la méthode createElement pour créer des éléments HTML pour chaque tweet récupéré et les ajouter à la table des tweets. Il y a également une fonction pour désactiver le bouton de recherche pendant le chargement des tweets et le réactiver après un délai de 2 secondes afin d'éviter l'exécution redondante de la fonction 'chercher_P' qui vide la table avent de les remplir avec les résultat de la nouvelle qui peut être vide.

5) Afficher les tweet selon un #ashtag

La partie principale de la page permet à l'utilisateur de rechercher des tweets par hashtag en saisissant le hashtag dans un formulaire de recherche. La recherche déclenche une requête AJAX vers le serveur qui récupère les tweets correspondants à partir de la base de données. Les tweets sont ensuite affichés dans un tableau sous forme de lignes, où chaque ligne représente un tweet et les colonnes représentent les informations associées au tweet telles que le timestamp, le nom d'utilisateur, le message et le hashtag. Le code JavaScript gère la soumission du formulaire de recherche et la manipulation du DOM pour afficher les résultats de la recherche.

6) Afficher la liste des #ashtag

La partie JavaScript utilise l'API fetch pour récupérer les données des hashtags depuis l'URL '/hashtags' de l'API Flask définie dans le backend. Elle convertit les données JSON en un tableau d'objets JavaScript, puis parcourt le tableau pour créer des éléments de liste HTML contenant les hashtags et les ajouter à la page. Cette partie permet donc d'effectuer des requêtes AJAX pour récupérer des données dynamiques depuis le backend et les afficher dans la page web.