Guide de développeur

Cloner le projet

Le projet se trouve dans le lien [3]. Ainsi pour le cloner, veuillez utiliser la commande suivante:

```
git clone https://github.com/wssuite/webapp.git
```

Déploiement du projet

Le déploiement à partir des images pré-construites est expliqué dans le document [1].

Pour des fins de développement, le développeur peut utiliser le fichier `docker-compose-dev.yml` afin d'utiliser `docker compose`

Afin d'utiliser le fichier docker-compose-dev.yml en exécutant la commande suivante:

```
docker compose -f docker-compose-dev.yml up
```

À l'opposition du fichier docker-compose.yml, le fichier docker-compose-dev.yml reconstruit l'image de nouveau en utilisant les derniers changements apportés au code source. Ainsi, nous pouvons forcer la récréation de certains services en utilisant la commande suivante:

```
docker compose -f docker-compose-dev.yml up -d --force-recreate
--no-deps --build <NOM_DE_SERVICE>
```

Enfin, pour arrêter les serveur, nous pouvons utiliser la commande suivante:

```
docker compose -f docker-compose-dev.yml down -rmi local -v
```

```
L'option -rmi local efface les images locaux crées par docker compose
```

```
L'option -v efface les volumes utilisé par docker compose
```

Description des composantes du projet

Le serveur frontfacingserver

Installation de dépendance

Ce serveur est développé avec Python de préférence version 3.10. Afin d'installer les dépendances, il est conseillé d'utiliser un environnement virtuel. Veuillez consulter le document [2] afin d'installer l'environnement virtuel et afin de l'activer.

Une fois l'environnement installé, le développeur aura besoin d'installer les dépendances. Pour faire cela, dans la directoire du frontfacingserver, exécutez la commande suivante:

```
pip install -r requirements.txt
```

Démarrer le serveur sans docker compose

Le serveur communique avec une base de données MongoDB. Par conséquent, afin de rouler le code du serveur sans docker compose, le développeur aura besoin d'une base de données MongoDB. Par conséquent, avant de démarrer le serveur, le développeur peut utiliser, dans la racine du projet, la commande suivante afin de démarrer le service de base de données:

```
docker compose -f docker-compose-dev.yml up mongodb
```

Ensuite, dans la directoire frontfacingserver, le développeur peut démarrer le serveur en utilisant la commande suivante:

```
python app.py
```

Cette méthode présente une limite car elle ne permet pas la génération d'horaires. En effet, afin de générer un horaire, nous avons besoin de démarrer le serveur HAProxy et les services d'invocation du code C++.

Tests automatiques du serveur

Des tests ont été développés pour s'assurer du fonctionnement du serveur sans avoir la nécessité de le démarrer. Pour les tests, des modules externes comme `mongomock` et `pyfakefs` afin de simuler une base de données MongoDB et un système de fichier respectivement. Les tests automatiques se limitent à simuler la connexion avec le module HAProxy.

D'ailleurs, les tests automatiques représentaient une bonne stratégie de vérification des fonctionnalités.

Afin d'exécuter les tests automatiques, le développeur peut utiliser la commandes suivantes:

```
pytest .
```

Note: Les tests ont un statut de succès en utilisant un ordinateur Unix. Certains tests peuvent échouer sur Windows à cause de la notation utilisée par le système de fichier.

Le développeur peut également s'assurer de la couverture du code en utilisant les commandes suivantes:

```
coverage run -m pytest .
coverage report -m
```

Le serveur cpp-invoker

Ce serveur est développé avec Python version 3.7 afin d'assurer la compatibilité de la version de Python avec l'image Docker. Ce serveur n'a pas de tests automatiques comme il est de petite taille et son but est de rouler le code C++ en utilisant la ligne de commande.

Il est suggéré d'utiliser ce serveur directement en utilisant docker compose.

Les dépendances de ce serveur se trouvent dans le fichier requirements.txt du dossier cpp-invoker. Ainsi, il est possible d'exécuter la commande suivante afin d'installer les dépendances:

```
pip install -r requirements.txt
```

L’application frontend

L’application frontend est une application web dévéloppé avec le cadriciel Angular afin d’installer les différents paquets npm vous avez besoin de quelques prérequis

Prérequis

1. Node.js version >= 16.17.0
2. Npm version >= 8.15.0

Afin d’installer Node

- Linux/MacOS
    - Nvm[4] est recommandée afin de pouvoir changer entre les versions de Node.js plus facilement
- Windows
    - Scoop[5] est recommandée
    - Une fois scoop est installé vous pouvez télécharger nvm en utilisant la commande `scoop install nvm`
    - Vous pouvez téléchargé Node.js en utilisant la commande `nvm:nvm install 16.17.0`

Une fois Node.Js et NPM sont installés vous pouvez installés les paquets npm nécessaires pour compiler l’application en éxecutant la commande `npm` ci à partir de la racine du répertoire front-end-application. Pour démarrer l’application, exécutez la commande `npm start`.
Si une erreur est remcontré en exécutant la commande `npm start` après l’exécution de la commande `npm ci`, assurez vous d’effacer le dossier “*.angular*” qui se trouve dans le répertoire front-end-application et réexécutez de nouveau la commande `npm start`.
Pour plus d’informations veuillez vous référer à la documentation d’Angular [6]

Références:

[1] https://docs.google.com/document/d/1ZJS3wKXP1VqBnR2pxIax8sy4orN9K8QPzXxSkw45wdA/edit#heading=h.qhd07je2rkat

[2] https://docs.python.org/3/library/venv.html

[3] https://github.com/wssuite/webapp

[4] https://github.com/nvm-sh/nvm

[5] https://scoop.sh/

[6] https://angular.io/docs