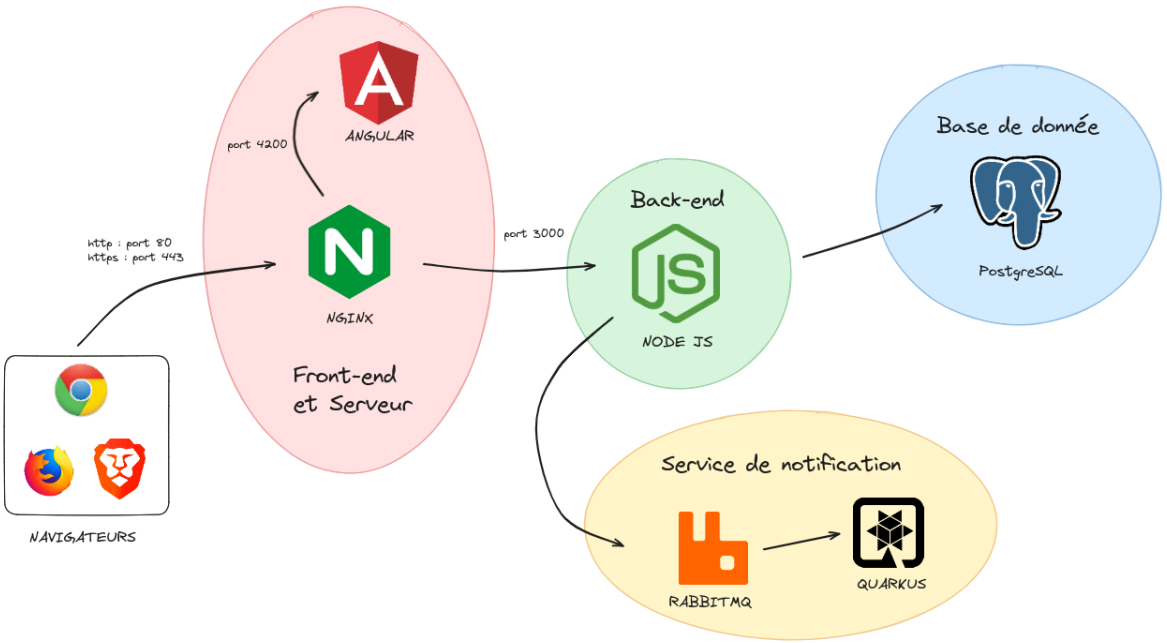
Rault Antoine Laminie Théo

Rapport TP Architecture logiciel

1. Schéma d'architecture des services



2. Explication de l'architecture

L'architecture est basée sur plusieurs conteneurs docker. Il y en a un pour le front-end et nginx, pour le back-end, pour la base de données et enfin pour le service de notification.Tout ces différents conteneurs seront regroupés dans un docker-compose pour pouvoir les lancer tous simultanément avec une unique commande.

Nous avons utilisé un seul conteneur pour nginx et le front-end car cela permet à nginx de récupérer le dossier dist du front-end avec les fichier html,css et typescript compilé en JavaScript. Cela permet entre autres d'améliorer les performances.

Nous n'avons pas réussi à implémenter une application web fonctionnelle avec une base de données tel que postgreSQL et nous n'avons pas fait de service de notification. Nous avons donc seulement deux conteneurs.

Le conteneur du front-end expose son port 80 au port 8000 tandis que le conteneur de back-end son port 3000 au port 3000. La communication entre le front-end et le back-end est faite par des requêtes sur le port 3000.

3. Liste des services implémentés

Nous avons implémenté les services suivant :

- Un conteneur pour le front-end et nginx
- Un conteneur pour le back end

4. État de chaque service

liste des services :

- Front-end et Nginx : implémenté
- Back end : implémenté
- Base de donné : proposé
- Service de notification : proposé

5. Instructions d'utilisation

Pour commencer il faut build les différents conteneur, pour cela il faut aller dans les répertoire Back-end et Front-end puis lancer les commandes suivante:

```
docker build -t back-end .
docker build -t front-end .
```

Ensuite pour lancer l'application web en mode développement, il faut lancer la commande suivante en se plaçant dans le répertoire du projet :

```
docker compose -f docker-compose.dev.yaml up -d
```

Cela lancera les conteneurs pour le front-end et le back-end, le front-end sera accessible sur le port 8000 et le back-end sur le port 3000.
Pour lancer l'application web en mode production, c'est la même chose mais cette fois-ci avec la commande :

```
docker compose -f docker-compose.prod.yaml up -d
```

On aura ensuite accès au front-end sur le port 8000. (Le back-end reste accessible sur le port 3000, nous n'avons pas réussi à faire fonctionner notre application sans le rendre accessible en mode production.)
Pour ensuite fermer les conteneurs lancés précédemment, on peut utiliser les commandes suivantes qui vont stopper et supprimer les conteneurs et les volumes :

```
docker compose -f docker-compose.prod.yaml down
docker compose -f docker-compose.dev.yaml down
```

6. Images Docker de production

Nous avons tout d'abord deux dockerfiles dans la partie back-end et front-end de notre projet pour pouvoir obtenir nos conteneurs. Dans le Dockerfile du front-end, on commence par spécifier que l'on souhaite utiliser la dernière version de node, on définit ensuite /app comme répertoire de travail dans le conteneur et on copie tout le front-end dedans. On donne ensuite les commandes pour lancer le projet, soit une commande pour installer les dépendances de node.js et une seconde pour construire l'application angular en mode

production dans le conteneur. On spécifie ensuite que l'on utilise nginx puis on copie le dist du projet app dans le répertoire html de nginx et on copie également les fichier de configuration de nginx. Et pour finir on donne le port à exposer, ici le 80. Pour le dockerfile du back-end, on définit également le répertoire de travail, on copie les fichiers dedans et on installe les dépendances node.js. Et ensuite on donne la commande pour lancer le back-end et le port a exposé, le port 3000. On utilise ensuite le docker-compose du mode production pour lancer les deux conteneur simultanément, le docker-compose est seulement composé d'une liste des deux services avec le nom de l'image et les ports de l'ordinateur et du conteneur qui communiqueront.