Architecture Logicielle

Rapport de Projet

Réalisé par

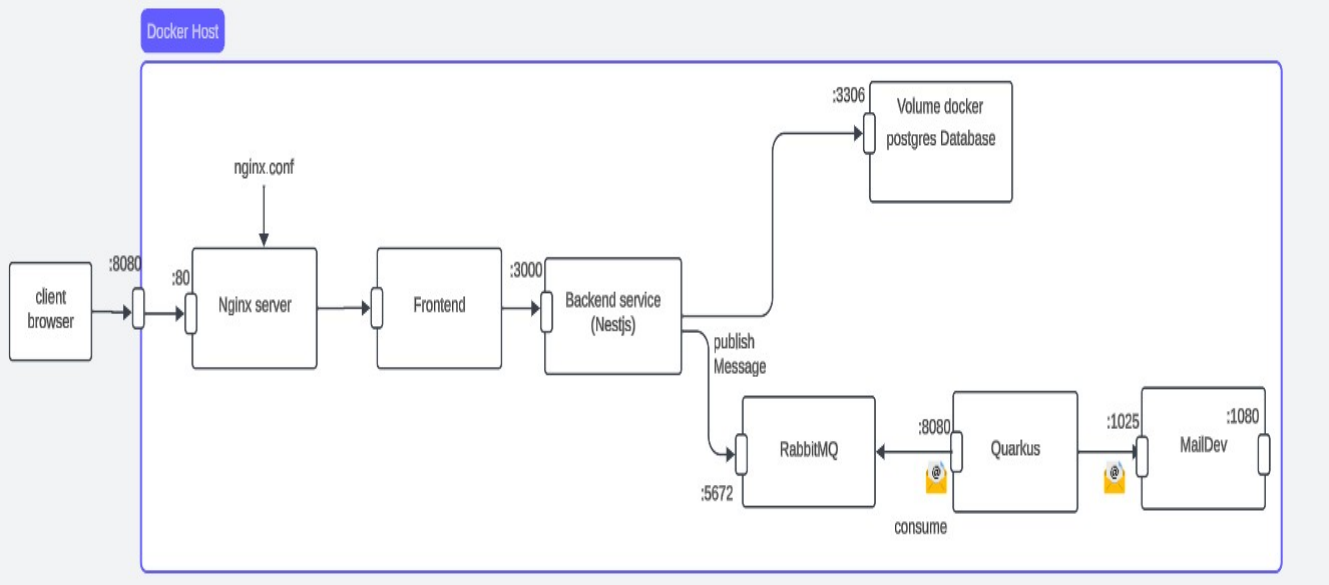
BERTHE Moussa – KONAN Yao Paul-David

Architecture Logicielle

Sommaire

Architecture Logicielle

1. Schéma de l’Architecture



2. Explication de l’Architecture et Alternatives

Architecture actuelle

L'architecture actuelle est une architecture distribuée composée de quatre composants principaux :

- **Un service frontend** qui s'interface avec l'application via un navigateur.
- **Un serveur Nginx** qui sert le contenu statique du frontend et proxy les requêtes vers le backend.
- **Un service backend** écrit en NestJS qui gère les logiques métier et publie des messages dans une queue RabbitMQ.
- **Un service Quarkus** qui consomme les messages de la queue RabbitMQ et les envoie aux utilisateurs via MailDev.
- **Un service Postgres** qui sert de basse de données pour stocker les données dans un volume docker en persistance.
- **Un service Maildev** qui sera le server mail qui recevra les mails destinées aux utilisateurs

Chaque composant est déployé dans un conteneur Docker.

Architecture Logicielle

Le Service frontend communique avec le backend via le port 3000. Le backend publie des messages dans la queue RabbitMQ via le port 5672. Les messages envoyés sur une queue de rabbitMQ sont reçus par le service Quarkus sous forme de bytes. Il faut donc les décoder avant leurs consommation via le port 8080. Après consommation, quarkus envoie ses messages sur le port 1025 de maildev qui les affiche sur sa page web sur le port 1080.

Architecture alternative

Une architecture alternative possible est une architecture monolithe. Dans cette architecture, tous les composants sont regroupés dans une seule application.

Cette architecture est plus simple à déployer et à gérer que l'architecture distribuée actuelle. Cependant, elle est également moins évolutive et moins résiliente.

Pour faire fonctionner une architecture monolithe dans ce cas, il faut combiner les fonctionnalités du backend et du service Quarkus dans une seule application. Cela peut être fait en utilisant une architecture API-first, où le backend expose une API REST qui est utilisée par le frontend.

Rôles des composants

Voici un résumé des rôles des différents composants de l'architecture actuelle :

- **Service frontend**: C’est une application angular qui s’occupe des interfaces utilisateurs et d’effectuer des requètes vers le backend pour le traitement des données.
- **Serveur Nginx** : sert le contenu statique du frontend et proxy les requêtes vers le backend.En plus dans son fichier de configuration, il s’occupe de la rédirection vers le frontend ou le backend en fonction des requettes.
- **Service backend** : Il gère non seulement les logiques métier mais aussi de la gestion de la base de données et de la publication des messages dans une queue RabbitMQ.
- **Service Quarkus** : Il consomme les messages venants de la queue RabbitMQ et les envoie sur le port 1025 du server SMTP MailDev.
- **Services rabbitMQ**: Il expose sa queue de reception de message pour recevoir les messges dépuis
- **Service Maildev**: Il reçoit les mails destinés aux utilisateurs sur le port 1025 et les affiche sur le port 1080
- **Service postgres**: Il sert de base de données pour les données persistantes

Compromis

Le choix entre l'architecture actuelle et l'architecture alternative dépend des besoins spécifiques de l'application. L'architecture actuelle est plus évolutive et plus résiliente, mais elle est également plus complexe à déployer et à gérer. L'architecture monolithe est plus simple à déployer et à gérer, mais elle est également moins évolutive et moins résiliente.

Dans le cas présent, l'architecture monolithe peut être une bonne option si l'application n'a pas besoin d'être très évolutive ou résiliente. Elle est également une bonne option si l'application est relativement petite et simple.

3. Liste des services Implémentés

Service frontend

- **Role dans l'architecture** : Fournit une interface utilisateur pour les utilisateurs.
- **Services offerts** : Affiche des pages web, permet aux utilisateurs d'interagir avec l'application.
- **Services consommés** : Service backend via le port 3000.

Serveur Nginx

- **Role dans l'architecture** : Sert le contenu statique du frontend et proxy les requêtes vers le backend.
- **Services offerts** : Serveur HTTP, proxy.
- **Services consommés** : Contenu statique du frontend, service backend via le port 3000.

Service backend

- **Role dans l'architecture** : Gère les logiques métier et publie des messages dans une queue RabbitMQ puis interagit avec la base de données.
- **Services offerts** : API REST, logique métier,typeorm et openAI
- **Services consommés** : Queue RabbitMQ via le port 5672.

Service Quarkus

- **Role dans l'architecture** : Consomme des messages de la queue RabbitMQ et les envoie aux utilisateurs via MailDev.
- **Services offerts** : API REST, envoi de messages sur la queue RabbitMQ via le port 8080.

Service Postgres:

- **Role dans l'architecture** : Servir de base se données
- **Services offerts** : permettre des actions sur la database via interface graphiques avec pgAdmin

Architecture Logicielle

MailDev

- **Role dans l'architecture** : Reçoit les messages destinés aux utilisateurs
- **Services offerts** : un service SMTP (port 1025) et un service web (port 1080)
- **Services consommés** : Aucun.

Prometheus:

- **Role dans l'architecture** : Prometheus collecte les métriques liées aux performances du front-end et du backend.
- **Services offerts** : vérification de l'utilisation des ressources (CPU, mémoire) et du nombre de messages en attente, le taux de consommation dans rabbitmq.
- **Service Consommé**: capvisor

Cadvisor:

- **Role dans l'architecture** : Il collecte les métriques des containers, les expose sous forme d'API HTTP puis les intègre dans prometheus.
- **Services offerts** : Inclure un système global de surveillance Prometheus, surveiller l'utilisation des ressources, les performances et la santé des conteneurs Docker.
- **Service Consommé**: backend

Statut de ces Services

Chacun de ces Services à été Implémenté

4. Options de Configuration de l'application

➔ Il faut se connecter à la base de données avec username, un password, database name, hostname et le port. Ces informations sont dans le fichier .env sous les noms de **POSTGRES_USER, POSTGRES_PASSWORD, POSTGRES_DB,POSTGRES_PORT,POSTGRES_CONNECTION,POSTGRES_HOST**

➔ Pour la connexion a rabbitMQ, il faut donner une URL, un username et un password. Les valeurs données ici sont respectivément **:**

RABBITMQ_URL,RABBITMQ_DEFAULT_USER, RABBITMQ_DEFAULT_PASS dans le .env et le docker-compose.yml

➔ Nginx est configurer dans le dossier nginx du projet notamment dans le fichier nginx.conf

Architecture Logicielle

➔ Prometheus est configuré dans le fichier prometheus.yml dans le dossier prometheus du projet

➔ Quarkus est configuré dans le fichiers application.properties dans le dossier ressources du projet quarkus

➔ Les commandes d'exécution des services:

- backend: npm start run:dev
- front-end: ng serve –open
- nginx: nginx, -g, daemon off
- prometheus: --config.file=/etc/prometheus/prometheus.yml

➔ Dépendances entre les services:

- le front-end dépend du back-end
- le back-end dépend de la base de données et de rabbtimq
- nginx dépend du back-end et du front-end
- quarkus dépend de rabbitmq
- maildev dépend de quarkus
- prometheus dépend du back-end et de cadvisor

➔ On a monté des volumes services

5. Comment utiliser l'application?

➔ Cloner le dépot git en ligne sur votre machine

➔ Lancer docker sur votre machine de préférence docker-desktop

➔ Lancer la commande docker compose up pour lancer tous les containers

➔ Ouvrir la page web de maildev à l'adresse localhost:1080

➔ Aller sur l'url localhost:81(car il y avait un process utilisant le port 80 sur ma machine) vous aurez la page d'accueil du server nginx.

➔ Cliquer sur le bouton frontend page pour avoir accès la page d'accueil du front-end

➔ Sur la barre de navigation cliquer sur l'item utilisateurs pour creer un/plusieurs utilisateurs.

➔ Créer une association avec certains utilisateurs en mettant leurs ID dans le champ MembersId

Architecture Logicielle

➔ Un mail sera envoyé à chacun des utilisateurs crées sur le server maildev

➔ Sur les le items association de la barre de navigation, cliquer sur le bouton Members de l’association sélectionnée.

➔ Modifier le role d’un membre en cliquant sur le bouton modify Role

➔ Un mail sera envoyé à l’utilisation dont le role a été changé