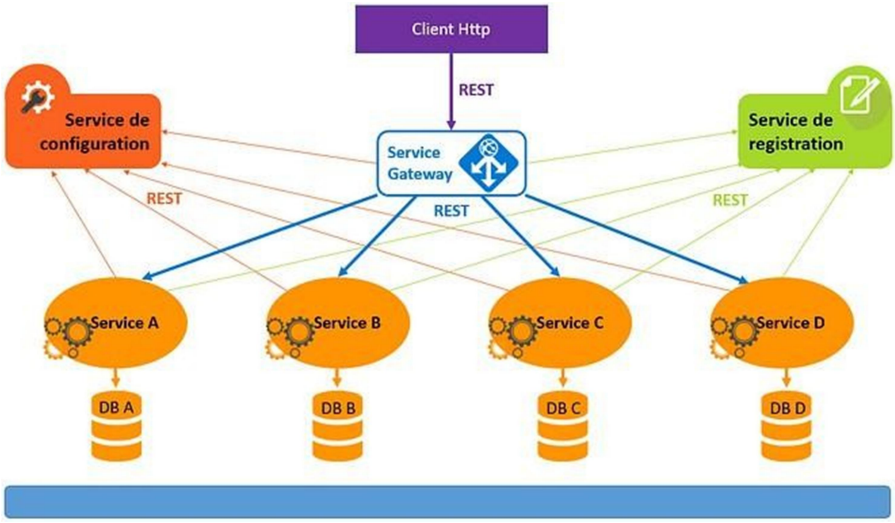
Mini-Projet :
"Application Web Distribuée - Microservices"

Le présent document vise à expliquer le travail requis pour le mini-projet "Application Web Distribuée - Microservices". L'objectif principal de ce projet est de développer une application basée sur une architecture de microservices, où deux services distincts communiquent entre eux.

Chaque microservice possède sa propre base de données, ce qui permet l'utilisation de différentes technologies de stockage, telles que MySQL, MongoDB, etc. En outre, plusieurs autres composants doivent être mis en place, notamment un serveur de registre, un proxy server et un serveur de configuration. L'ensemble de l'application doit être sera exécuté en utilisant Docker Compose.



Description du projet :

- **Création des microservices** :

Chaque étudiant doit développer deux microservices qui interagissent entre eux. Par exemple, un service pourrait être axé sur la gestion des articles, tandis que l'autre s'occupe du stock. Les microservices doivent être indépendants et communiquer entre eux via des mécanismes de communication appropriés.

- **Bases de données distinctes** :

Chaque microservice doit avoir sa propre base de données. Vous pouvez choisir différentes technologies de stockage pour chaque microservice, comme MySQL, MongoDB, ou d'autres, en fonction des besoins spécifiques de chaque service.

- **Communication entre les microservices :**

Les microservices doivent communiquer entre eux de manière **sécurisée** et efficace. Les étudiants doivent implémenter un mécanisme de communication inter-services (REST API et message brokers). **La communication synchrone** doit gérer les erreurs potentielles (circuit breaker)

- **Serveur de registration** :

Il est nécessaire de mettre en place un serveur de registre pour permettre aux microservices de s'enregistrer et de découvrir les autres services disponibles dans le système. Cela facilite la communication entre les microservices et permet de gérer facilement les mises à l'échelle et les remplacements.

- **Proxy server (service Gateway):**

Le proxy server est responsable du routage des requêtes vers les microservices appropriés. Il doit être configuré de manière dynamique afin de pouvoir gérer les modifications de configuration sans nécessiter de redéploiement de l'application.

- **Serveur de configuration :**

Un serveur de configuration doit être mis en place pour centraliser la gestion des configurations de l'application. Il doit être capable de récupérer les fichiers de configuration à partir d'un référentiel Git, ce qui facilite la gestion des modifications de configuration et permet de suivre l'historique des changements.

- **Utilisation de Docker Compose :**

Pour garantir une exécution homogène de l'application, toutes les parties du système doivent être lancées en utilisant un seul fichier de configuration Docker Compose. Cela facilite le déploiement de l'application sur différentes machines et assure une cohérence dans l'environnement d'exécution.

- **Sécurisation des microservices avec SSO (Keycloak) :**

Les microservices doivent être sécurisés en utilisant une solution SSO (Single Sign-On) telle que Keycloak. Chaque microservice doit exiger une authentification via Keycloak pour accéder à ses endpoints(APIs).

Bon travail !