**FACULTE DES SCIENCES**

FACULTY OF SCIENCE

**Département de Mathématiques**

**et Informatique**

Department of Mathematics and

Computer Science

BP 67, Dschang (Cameroun)

E-mail : udsrectorat@univ-dschang.org

**RÉPUBLIQUE DU CAMEROUN**

REPUBLIC OF CAMEROON

*Peace – Work – Fatherland*

**UNIVERSITÉ DE DSCHANG**

UNIVERSITY OF DSCHANG

*Scholae Thesaurus Dschangensis Ibi Cordum*

P 96, Dschang (Cameroun) -Tél. /Fax (237) 233 45 13 811

Website: http://www.univ-dschang.org



**Licence III Informatique**

**INF 312 : Base de Données**

**Projet**

**Thème : Création d’un système de gestion des états**

Enseignant:

**Dr Benoit AZANGUEZET**

2023 /2024

|  |  |  |  |
| --- | --- | --- | --- |
| LISTE DES PARTICIPANTS | | | |
| **N°** | **Noms** | **Prénoms** | **Matricules** |
| 01 | KENFACK FONGANG | VICTOR CYNTICHE | CM-UDS-21SCI0555 |
| 02 | NANSEU NJAKOU | STEVE WILFRIED | CM-UDS-19SCI0892 |
| 03 | MFENTAM | MOHAMMED SALAM | CM-UDS-21SCI0941 |
| 04 | LOMOFOUET NDONGMO | HANDREY JAURESS | CM-UDS-21SCI0124 |
| 05 | FEUTSEU TCHUENKAM | IVAN LIONEL | CM-UDS-20SCI0027 |
| 06 | JIOTANG DONGMEZA | Rostov Perez | CM-UDS-21SCI1068 |
| 07 | SOFEUGANG | JUNIOR HERVE | CM-UDS-19SCI0100 |
| 09 | KEOU TCHEUMEDJIE | NINE-NICAISE | CM-UDS-21SCI0562 |
| 1O | YETTCHOM TCHIZE | ELODIE ROSITA | CM-UDS-21SCI0691 |
| 11 | KANE DJOUFACK | FLORIANE JOSEPHINE | CM-UDS-20SCI039 |
| 11 | DJOUFACK | ERICA | CM-UDS-19SCI0325 |

Chapitre1 : Front-end

## Objectif

Le front-end de notre projet repose sur une architecture basée sur les composants. En faite ces composants représentent les différentes parties de l’interface utilisateur interactive.

## **Technologies utilisées**

Notre plateforme repose sur l’utilisation du framework React.js. Ce framework est de nature JavaScript développe par Facebook et repose sur la notion de composant. Pour son utilisation elle nécessite un certains nombres d’installation de dépendance a l’instar de:

### **2.1 Node.js**

C’est un environnement d’exécution JavaScript cote serveur, ce qui signifie que les développeurs peuvent utiliser JavaScript a la fois cote client et cote serveur.

### **2.2 Npm**( Node Package Manager)

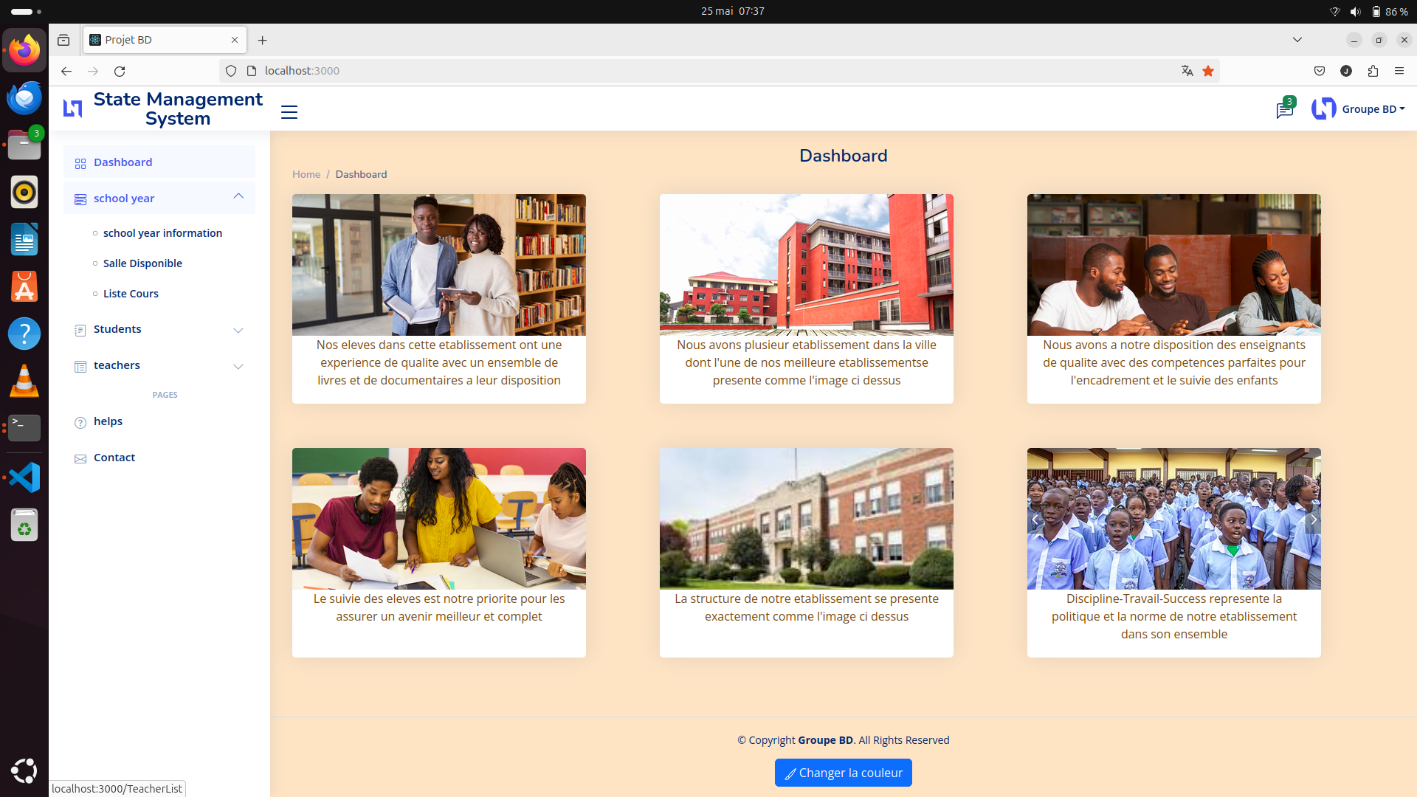
C’est un gestionnaire de paquets par défaut pour Node.js. Il permet d’installer, gérer et de partager des packages JavaScript et des dépendances pour des projets Node.js. Comme l’une des fonctionnalités nous avons:

Scripts personnalises: il permet aux développeurs de définir des scripts personnalises dans leur fichier package.json,ce qui facilite l’exécution de taches telles que la compilation,les tests et le déploiement.

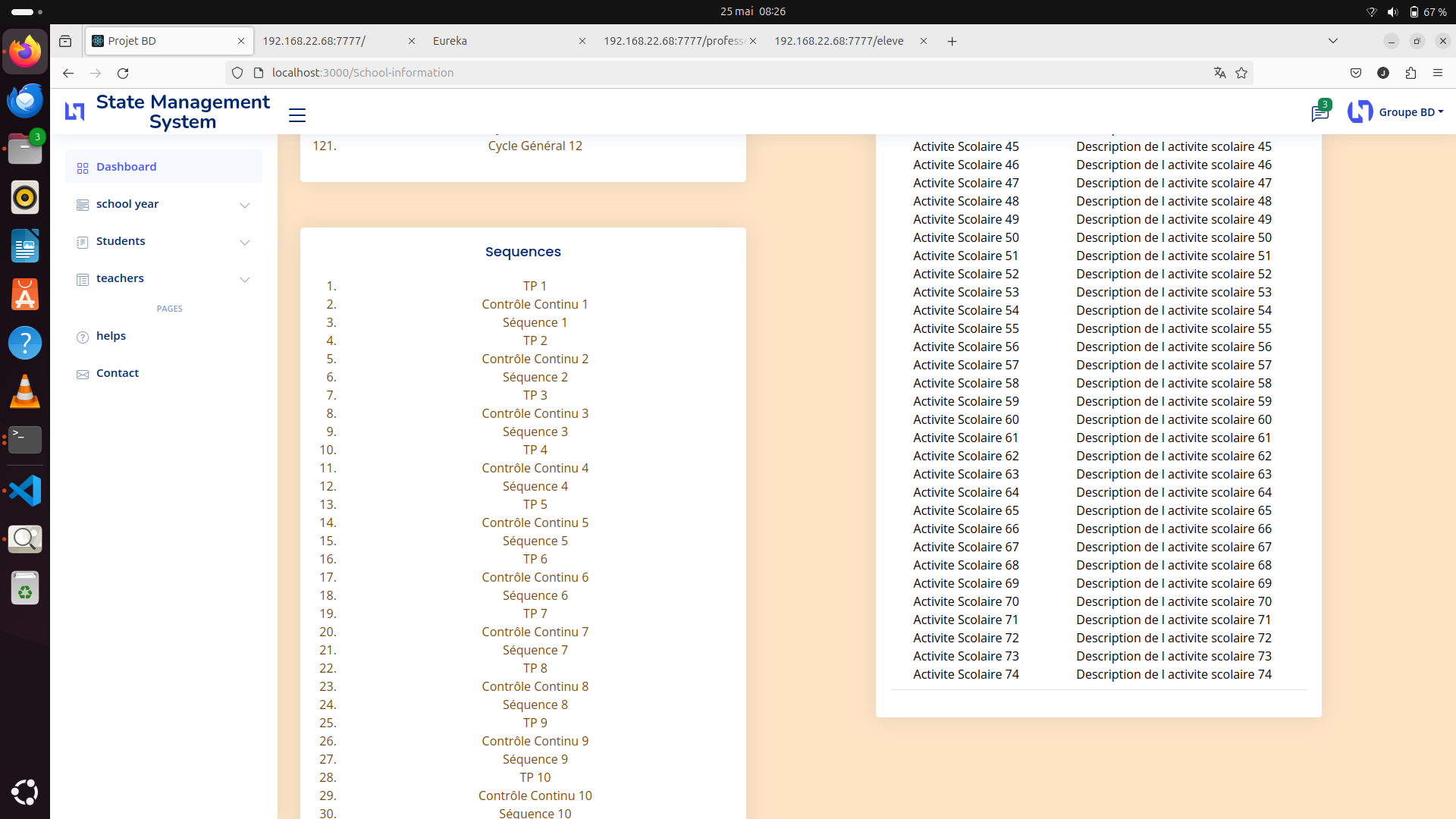
### **2.3 vite**

C’est un outil de développement rapide pour les applications react.js. Il est conçu pour améliorer le processus de développement en fournissant une configuration et des fonctionnalités prête a l’emploie, ce qui permet aux développeurs de se concentrer sur l’ecriture du code plutôt que sur la configuration.

## **Interface de nos différents composants**

****

## A screenshot of a computer Description automatically generated



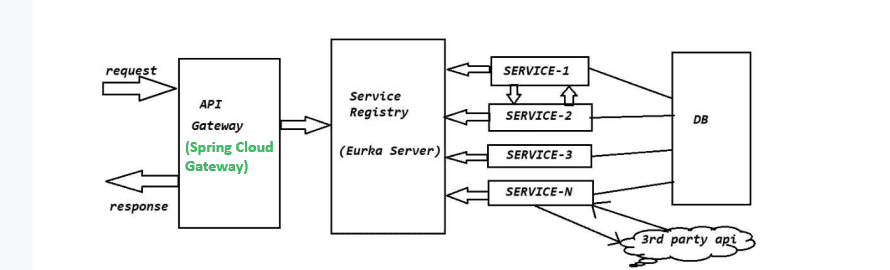
A screenshot of a computer

Description automatically generated

# Chapitre 2 : Back-end

## Objectif

Le back-end de notre projet repose sur une architecture basée sur Spring Boot, avec une configuration de Spring Cloud pour la production d'une API de micro-services. Ce back-end utilise à la fois MongoDB et PostgreSQL comme bases de données, associées à des serveurs de découverte de services Eureka, un serveur de traçage Zipkin, une Gateway et un système d'authentification pour accéder aux points de terminaison des micro-services. Ainsi, nous décrirons en détail le fonctionnement des micro-services, le rôle de chaque serveur et la communication entre les micro-services, les serveurs et les bases de données. Notre objectif est de fournir une plateforme flexible et performante pour la gestion et le traitement des données, en utilisant des technologies modernes et des pratiques recommandées qui offre une base solide pour le développement de micro-services en Java. En intégrant Spring Cloud, nous étendons les capacités de Spring Boot pour la création et la gestion de systèmes distribués complexes. Cette approche nous permet de développer et de déployer efficacement des micro-services indépendants, tout en assurant une communication transparente entre eux comme suit :



1. Technologies Utilisées

### Spring Boot

Spring Boot est utilisé comme Framework principal pour développer nos micro-services, offrant une configuration simple et une approche conventionnelle pour démarrer rapidement.

### Spring Cloud

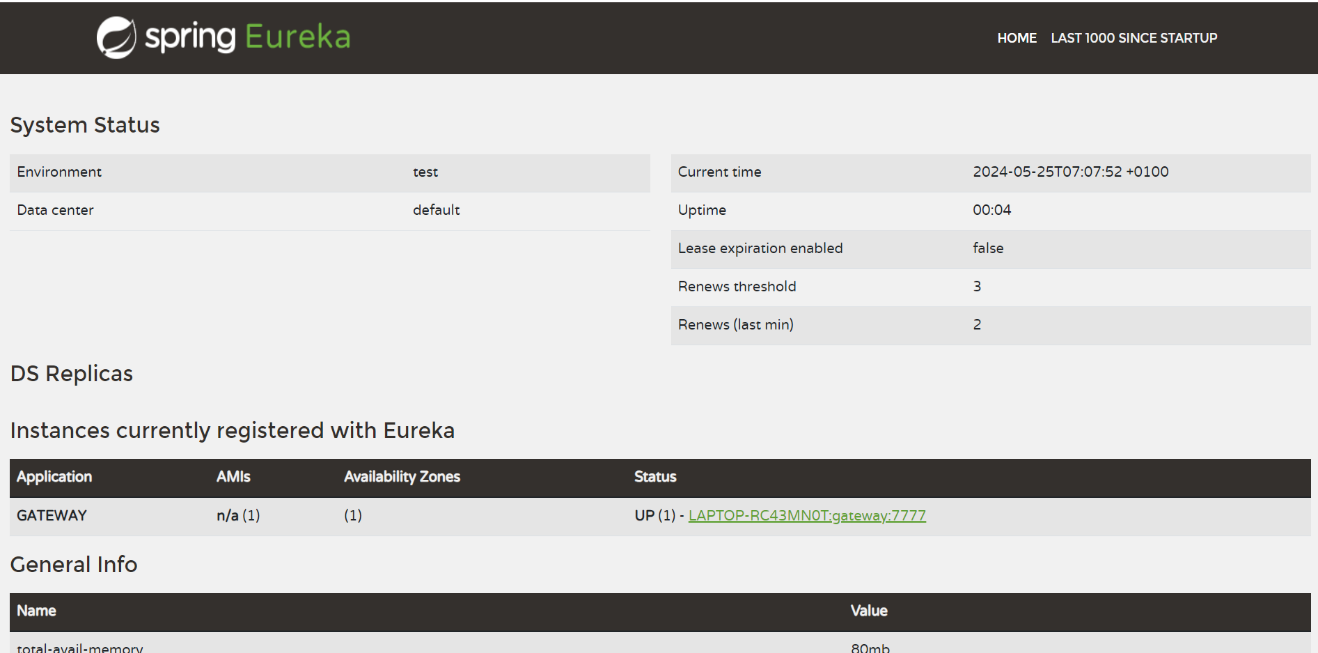
Spring Cloud facilite la création de systèmes distribués complexes en fournissant des outils pour le déploiement et la gestion de micro-services.

## MongoDB et PostgreSQL

Nous utilisons MongoDB et PostgreSQL comme bases de données pour stocker les données nécessaires à notre application, en choisissant MongoDB pour sa flexibilité dans le stockage de données non structurées et PostgreSQL pour sa robustesse dans la gestion des données relationnelles. Cette décision a été prise en fonction des besoins spécifiques de notre application, en utilisant MongoDB pour stocker des données non structurées nécessitant une flexibilité de schéma, tandis que PostgreSQL est utilisé pour des données plus structurées nécessitant une intégrité et une cohérence élevées.

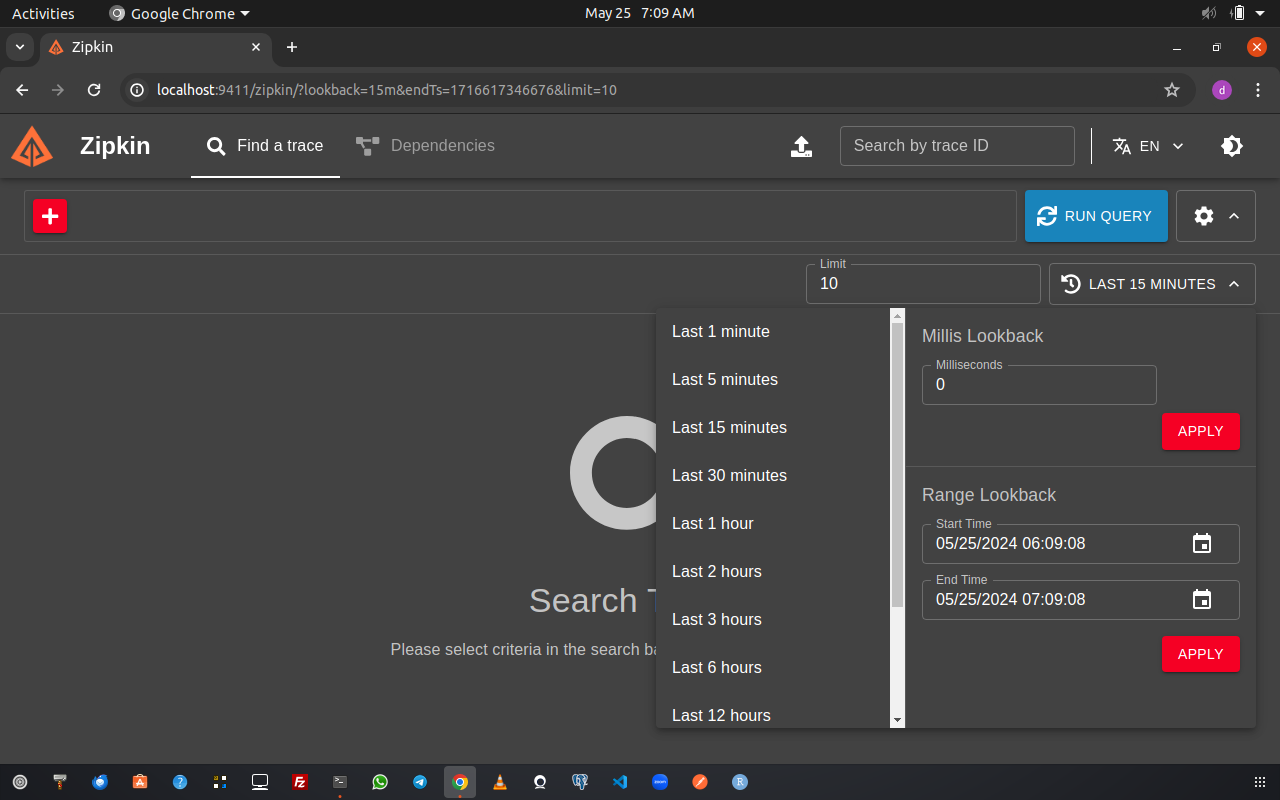
### Eureka

Eureka est utilisé comme serveur de découverte de services, permettant à chaque micro-service de s'enregistrer et de découvrir les autres services disponibles dans l'écosystème pour garantir la haute disponibilité et la résilience de notre système, nous avons mis en place un serveur de découverte de services Eureka. Ce serveur permet à chaque micro-service de s'enregistrer et de découvrir dynamiquement les autres services disponibles dans l'écosystème, facilitant ainsi la gestion des dépendances et des mises à jour.



### Zipkin

Zipkin est un serveur de traçage utilisé pour suivre et visualiser les requêtes à travers différents micro-services, facilitant le débogage et l'optimisation des performances Pour le suivi et la visualisation des requêtes à travers nos micro-services, nous avons intégré Zipkin, un serveur de traçage distribué. Zipkin nous permet d'identifier les goulets d'étranglement et d'optimiser les performances de notre système en analysant les temps de réponse et les dépendances entre les services.



### Gateway

La Gateway agit comme un point d'entrée unique pour toutes les requêtes client, dirigeant le trafic vers les micro-services appropriés en fonction des URL et des règles prédéfinies. En tant que point d'entrée unique pour toutes les requêtes client, notre Gateway agit comme un proxy inverse, dirigeant le trafic vers les micro-services appropriés en fonction des règles de routage prédéfinies. Cette approche simplifie la gestion du trafic et renforce la sécurité en centralisant l'authentification et l'autorisation des utilisateurs.

### Système d'Authentification

Un système d'authentification est mis en place pour sécuriser les points de terminaison des micro-services, garantissant que seuls les utilisateurs autorisés peuvent accéder aux données sensibles. Pour garantir la sécurité de notre système, nous avons mis en place un système d'authentification qui protège les points de terminaison des micro-services contre les accès non autorisés. Ce système permet de vérifier les identités des utilisateurs et de garantir que seuls les utilisateurs authentifiés et autorisés peuvent accéder aux données sensibles de notre application.

## Configuration

### Configuration des Micro-services

Chaque micro-service est configuré pour s'enregistrer auprès du serveur Eureka lors de son démarrage, permettant ainsi une découverte dynamique des services.

### Configuration des Bases de Données

Les configurations pour se connecter à MongoDB et PostgreSQL sont définies dans les fichiers de propriétés de chaque micro-service, assurant une gestion centralisée des paramètres de connexion.

### Configuration de la Gateway

La Gateway est configurée pour rediriger les requêtes HTTP entrantes vers les micro-services appropriés en fonction des règles de routage prédéfinies.

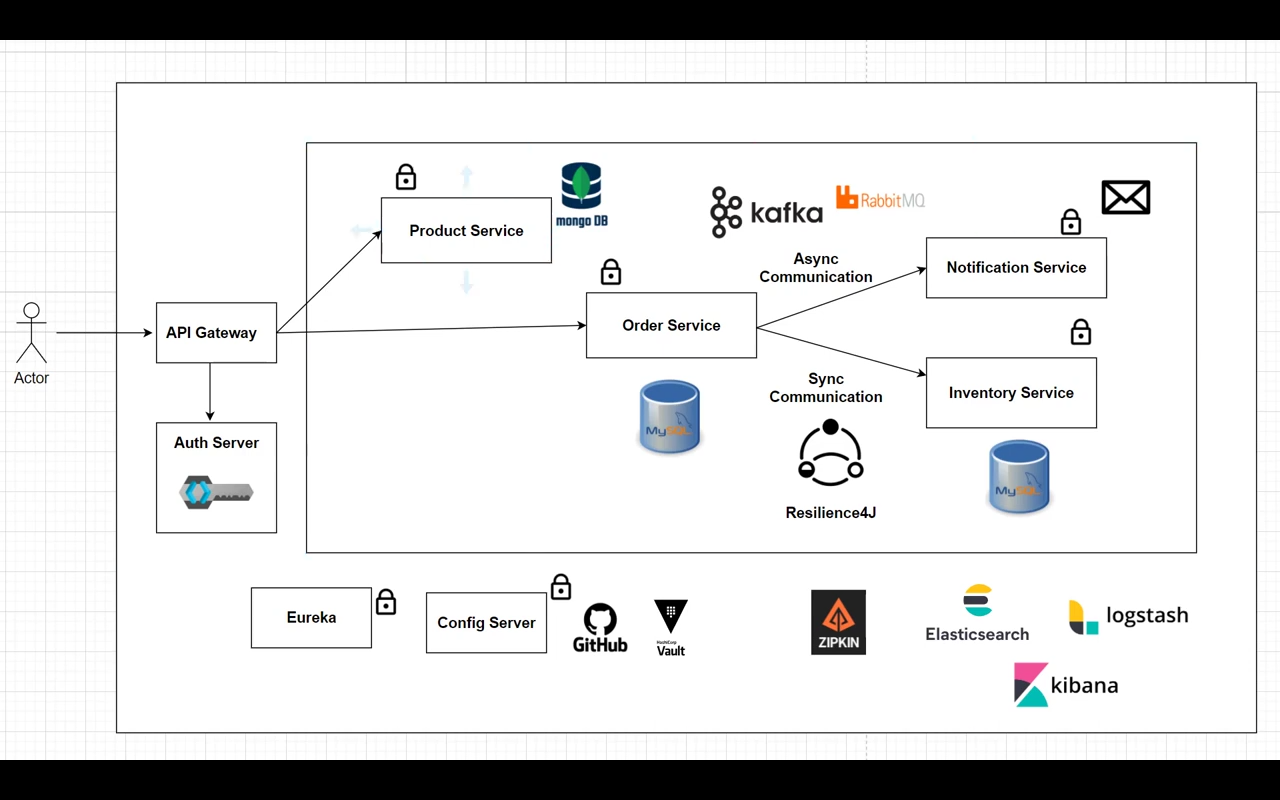
### Configuration de l'Authentification

Le système d'authentification est configuré pour intercepter les requêtes vers les points de terminaison des micro-services et vérifier les jetons d'authentification avant de permettre l'accès.

## Communication entre les Composants

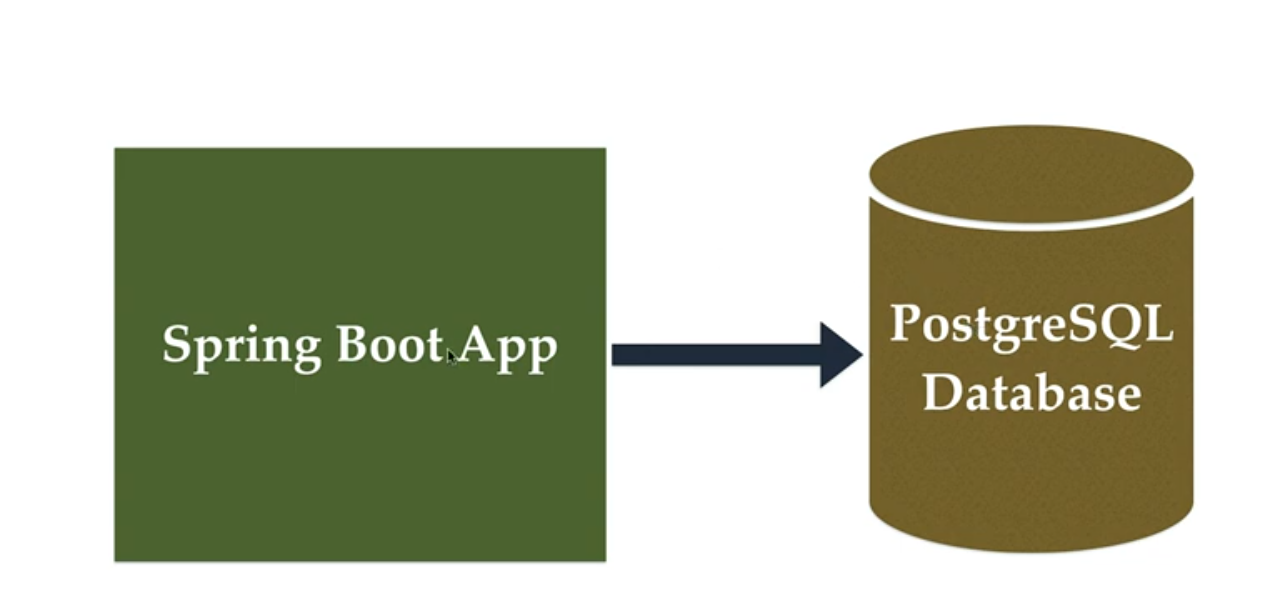
### Communication entre les Micro-services

Les micro-services communiquent entre eux via des requêtes HTTP RESTful, en utilisant les URL fournies par le serveur Eureka pour découvrir et atteindre les services nécessaires.



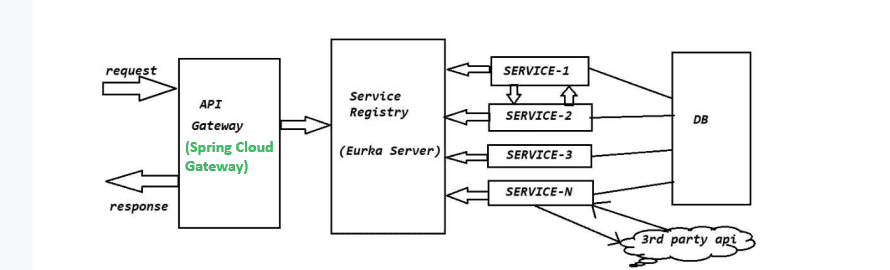
### Communication avec les Bases de Données

Chaque micro-service communique avec sa base de données respective en utilisant les bibliothèques de connexion appropriées, garantissant l'accès sécurisé et efficace aux données.



### Communication avec la Gateway

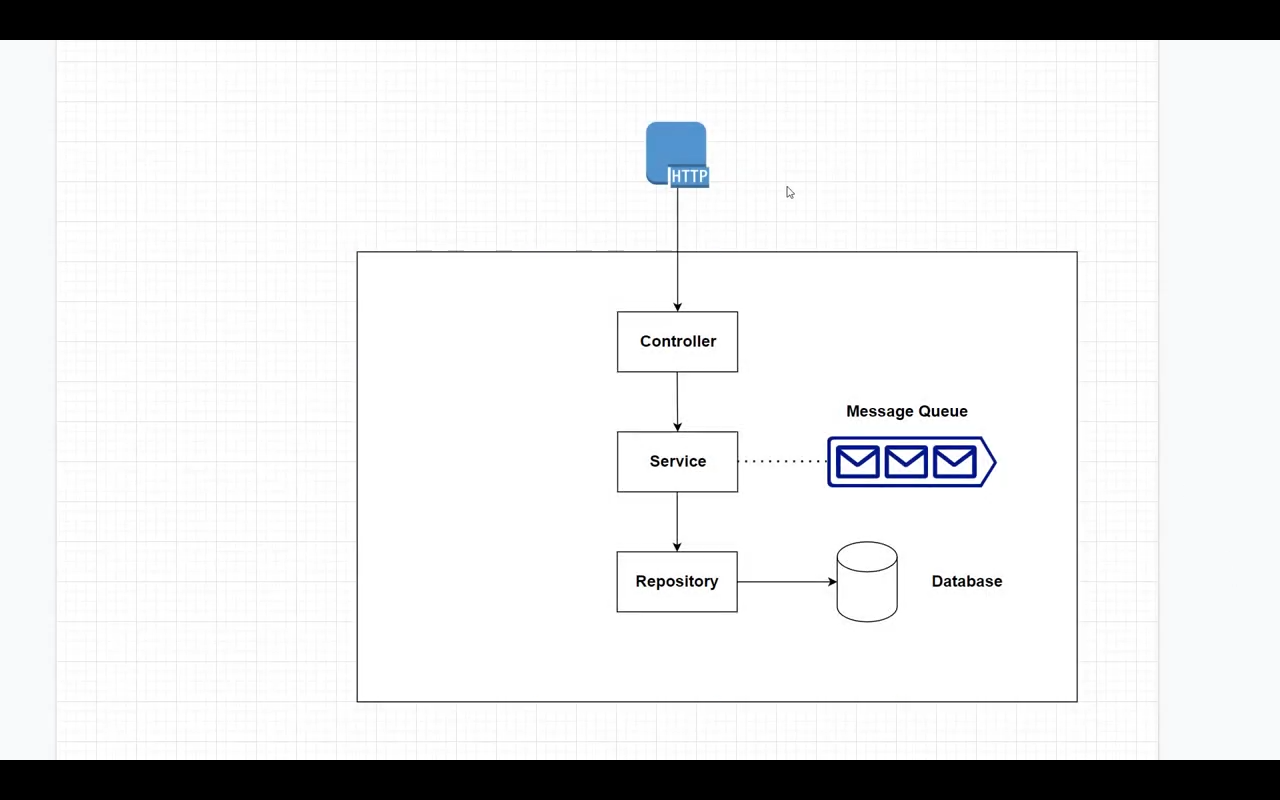
La Gateway communique avec les micro-services en agissant comme un proxy inverse, redirigeant le trafic client vers les micro-services appropriés en fonction des règles de routage définies.



## Exposition via la Gateway

### Exposition des Micro-services

Les micro-services sont exposés via la Gateway en définissant des points de terminaison RESTful accessibles aux clients externes via des requêtes http dont le déroulement est le suivant :



### Authentification via la Gateway

La Gateway intègre le système d'authentification pour sécuriser les points de terminaison des micro-services, nécessitant une authentification avant l'accès aux données.

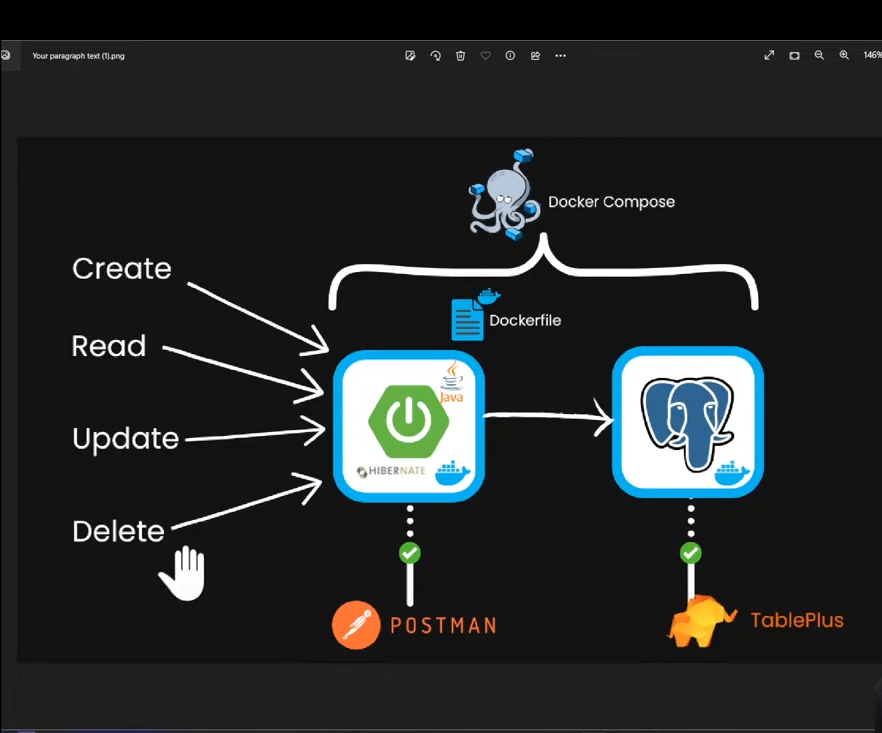
## Fonctionnement avec Front-end React

### Connexion avec la Gateway

Le front-end React se connecte à la Gateway en utilisant les URL exposées pour chaque micro-service, envoyant des requêtes HTTP pour récupérer et manipuler les données nécessaires.

### Intégration de l'Authentification

Le front-end React intègre le système d'authentification en envoyant des jetons d'authentification avec chaque requête vers la Gateway, assurant ainsi un accès sécurisé aux données.



# Annexe

Lien github: <https://github.com/regent-ndongmo/Projet_BD>