

***Projet de fin semestre***

**Mise en place d’une une application web avec l’architecture micro services**

**Encadré par** : M. Khadija Ahaidous

**Réalisé par** : Yassine Archid

El Mehdi Fathi

1. *Introduction*

Dans ce projet, nous présentons notre projet de gestion d'école basé sur une architecture de microservices avec le framework Spring. Nous avons utilisé des technologies telles que Spring Cloud Gateway pour gérer les requêtes entrantes, OpenFeign pour faciliter les communications entre les microservices, Eureka pour la découverte de service et Kafka Streams.

Notre objectif était de développer une application moderne, évolutive et facile à maintenir pour la gestion des activités d'une école. Nous avons choisi de construire notre solution sur une architecture de microservices pour permettre une flexibilité accrue et une meilleure scalabilité pour répondre aux besoins de l'école.

Dans les sections suivantes, nous décrirons en détail la conception et la mise en œuvre de notre application, les défis rencontrés et comment nous les avons résolus, ainsi que les résultats obtenus.

1. *Cahier de charge*

Créer le micro-service student-service qui permet de gérer les étudiants.

2. Créer le micro-service course-service qui permet de gérer les cours.

3. Créer le micro-service professors-service qui permet de gérer les professeurs

4. Créer la Gateway Spring cloud Gateway avec une Configuration statique du système de

routage

5. Créer l'annuaire Eureka Discrovery Service.

6. Faire une configuration dynamique des routes de la gateway.

7. Créer le service d’absence Absence-Service en utilisant Open Feign.

8. Créer un client Web Angular (Etudiants, Professeurs, Cours, Absence)

9. Déployer le serveur keycloak :

- Créer un Realm

- Créer un client à sécuriser

- Créer des utilisateurs

- Créer des rôles (ETUD, PROFS,STUFF)

- Affecter les rôles aux utilisateurs

- Tester les différents modes d'authentification avec Postman en montrant les

contenus de Access-Token, Refresh Token

10. Sécuriser les micro-services et le frontend angular en déployant les adaptateurs

Keycloak :

a. Les étudiants peuvent seulement accéder à une synthèse de leurs absences.

b. Les professeurs peuvent accéder à la liste des étudiants et marquer l’absence.

c. Le stuff administratif peut gérer les étudiants, les professeurs, les cours et avoir

un accès sur les absences pour chaque étudiant.

11. Intégration du Bocker KAFKA.

12. Création d'un micro-service qui permet de produire aléatoirement des absences et de les

publier dans un Topic KAFKA.

13. Permettre au Micro-service déjà développé ABSENCE-SERVICE de consommer les

absences publier dans le Topic KAFKA et de les enregistrer dans sa base de données.

14. Créer un micro-service Data-Analytics-Service qui utilise l'API KAFKA Streams pour

effectuer du Real Time Stream Processing en consommant le streams d’absences

publiées dans le Topic KAFKA

15. Créer une Page Frontend qui permet de présenter en temps réel les courbes qui montrent

les résultats produits par le service du Data Analytics (Accès au STUFF seulement)

16. Déployer l'ensemble des services de l'application en utilisant des conteneurs Docker :

Créer les images Docker pour chaque service et e fichier Docker-compose.yml qui permet

de déployer toute l'application

17. Ajouter des Fonctionnalités supplémentaires de votre choix.

1. *Présentation du projet*

1-présentation

le projet sert à réaliser une application web de la gestion des école en utilisant plusieurs technologies qui permet l’implémentation du projet basé sur spring cloud.

2-choix des technologies



**Keycloak** est un système de gestion d'identité et d'accès open-source, qui vous permet de gérer les authentifications et autorisations pour vos applications web et mobiles. Il fournit une interface utilisateur pour gérer les utilisateurs et les rôles, ainsi que des fonctionnalités de sécurité telles que la gestion de sessions et la protection contre les attaques par rejeu. Keycloak peut être utilisé pour centraliser la gestion des identités pour une variété d'applications, ce qui facilite la gestion des accès pour les utilisateurs finaux.

**Apache Kafka** est un système de messagerie distribué conçu pour traiter des flux de données en temps réel dans un environnement hautement scalable et tolérable aux pannes. Il permet de publier et de souscrire à des messages organisés en "topics", ce qui en fait un outil populaire pour la mise en œuvre de architectures d'archivage de données en masse, de pipelines de traitement de données en temps réel et d'applications de messagerie. Kafka s'appuie sur un modèle de distribution de partitions pour permettre une évolutivité horizontale et une forte fiabilité.\*

**Kafka Streams** est une bibliothèque de traitement de flux de données en temps réel basée sur Apache Kafka. Il permet aux développeurs de construire des applications de traitement de données en temps réel qui peuvent lire des données depuis un ou plusieurs topics Kafka, les transformer et les produire à d'autres topics. Kafka Streams utilise une architecture de traitement basée sur des microservices, ce qui en fait une solution évolutive et scalable pour le traitement de données en temps réel. Il fournit également une API pour gérer la gestion des états et des événements, ce qui en fait un choix populaire pour la mise en œuvre de applications de traitement de données complexes.

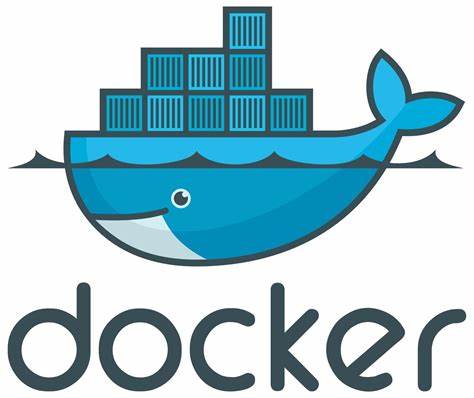
**Eureka** est un service de découverte de services open source développé par Netflix. Il fonctionne comme un annuaire pour les microservices dans un environnement distribué, permettant à chaque microservice de connaître l'emplacement des autres microservices nécessaires à son fonctionnement. Cela permet aux microservices de s'inscrire auprès de Eureka et de diffuser leur disponibilité aux autres microservices. Eureka aide à résoudre les dépendances entre les microservices et à améliorer la robustesse de l'application globale en permettant une redirection automatique des demandes vers des instances disponibles en cas de défaillance d'une instance.



**Postman** est une plateforme de développement de logiciels qui facilite les tests, les développements et la documentation des API (Application Programming Interfaces). Il fournit une interface utilisateur pour construire, tester et partager des requêtes HTTP et des requêtes API, ainsi qu'une variété d'outils pour travailler avec des API, tels que la construction de requêtes, la capture des réponses, l'analyse des codes de retour et la gestion des en-têtes. Postman est largement utilisé par les développeurs pour tester les API pendant le développement, la documentation et la démonstration des API à d'autres équipes. Il offre également des fonctionnalités pour la collaboration, telles que la partage et la documentation des collections d'API.

**Spring Cloud Gateway** est une implémentation de passerelle API open-source basée sur Spring Framework qui permet de gérer le trafic des API pour les applications distribuées. Il agit comme une couche d'abstraction pour les API, en fournissant des fonctionnalités telles que la sécurité, la répartition de charge, la transformation des données et la gestion des erreurs. En utilisant une approche de programmation basée sur les routes, Spring Cloud Gateway permet aux développeurs de définir comment le trafic API doit être dirigé et traité. Il intègre également bien avec d'autres composants du système Spring, tels que Eureka pour la découverte de services et Ribbon pour la répartition de charge.

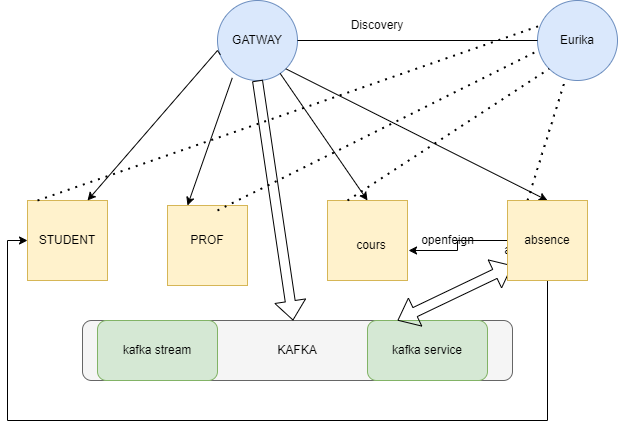
**JWT** signifie "JSON Web Token". C'est un standard ouvert (RFC 7519) qui définit un format de transmission de données sécurisé entre parties en utilisant un format de jeton compact et autonome. Les JWT sont généralement utilisés pour authentifier les utilisateurs en transmettant des informations d'identification telles que le nom d'utilisateur et les rôles d'un utilisateur à travers plusieurs applications distribuées. Les JWT sont cryptés et signés digitalement pour garantir l'intégrité et l'authenticité des données transmises. Les JWT sont souvent utilisés avec les API REST pour authentifier et autoriser les requêtes entrantes.



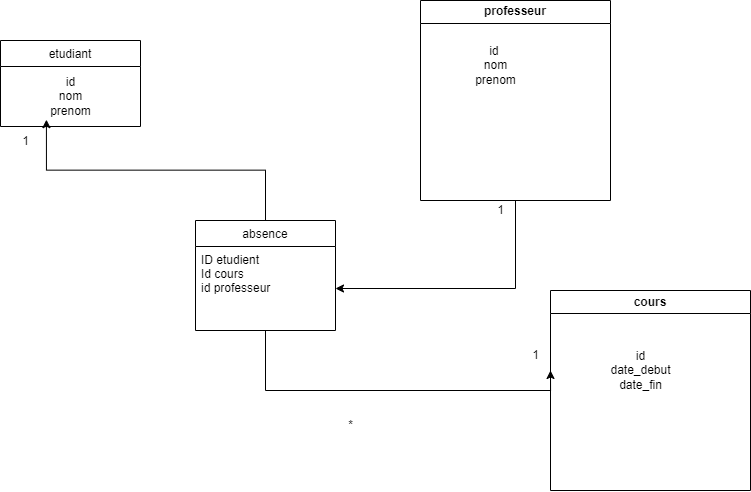
**Docker** est un logiciel d'empaquetage et de déploiement d'applications dans des conteneurs logiciels. Les conteneurs sont des environnements isolés qui regroupent les applications avec toutes leurs dépendances (telles que les bibliothèques et les configurations) pour les exécuter sur n'importe quelle infrastructure. Cela permet une portabilité accrue des applications et une plus grande efficacité en termes de ressources informatiques. Docker a également une forte communauté de développeurs qui créent et partagent des images de conteneurs préconfigurées pour une large gamme d'applications et de technologies.

1. *Architecture*

1-architecture du projet



2-diagramme de class



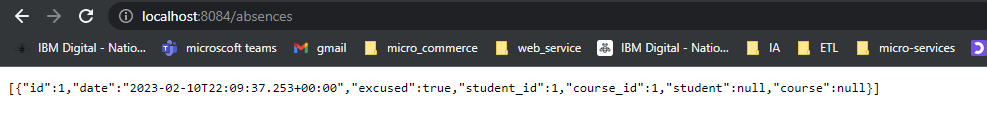
1. *Réalisation*

1-présentation des microservices

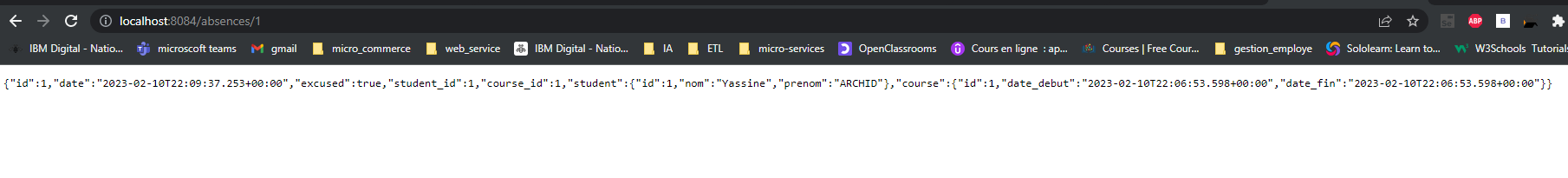








absencebyid

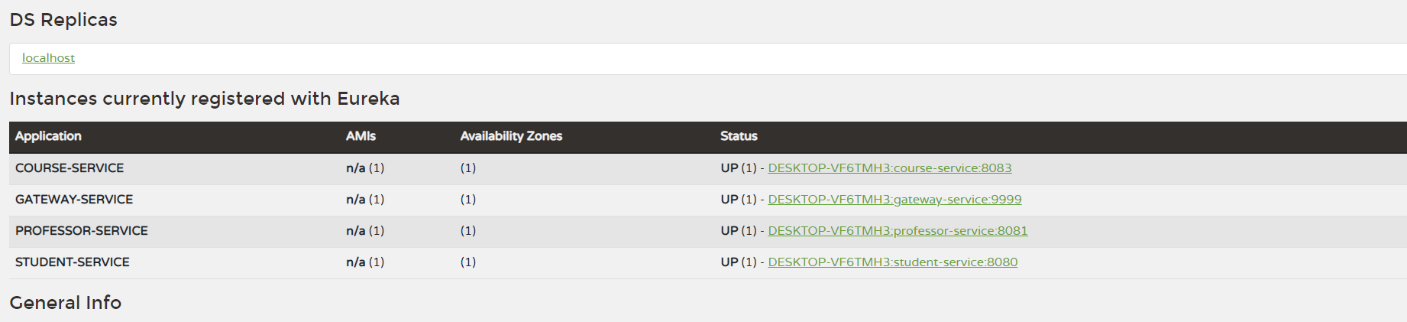




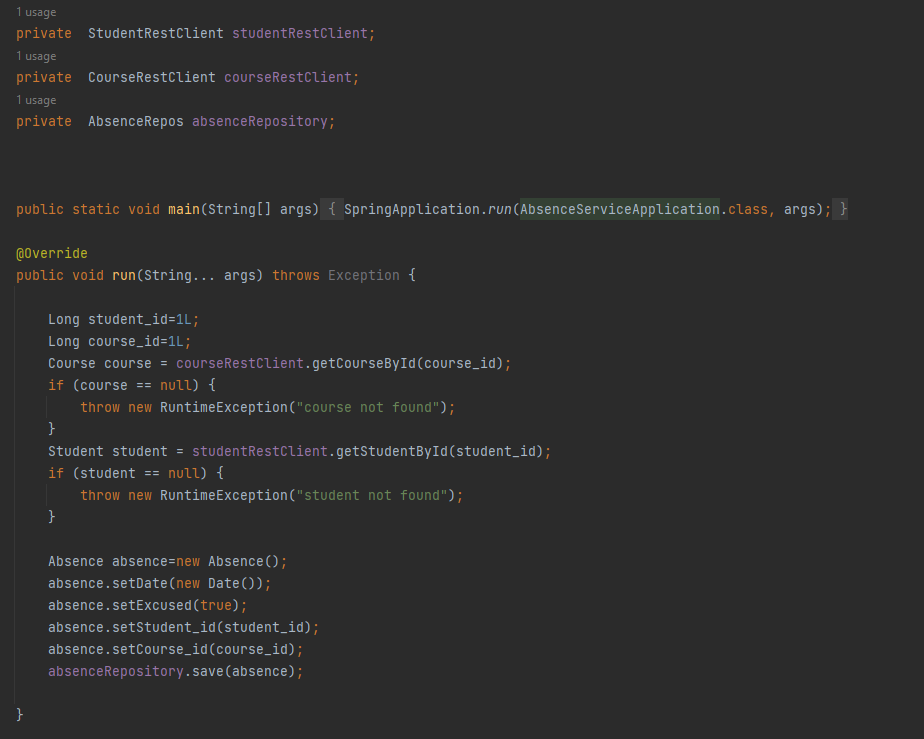
2-gateway statique/dynamique

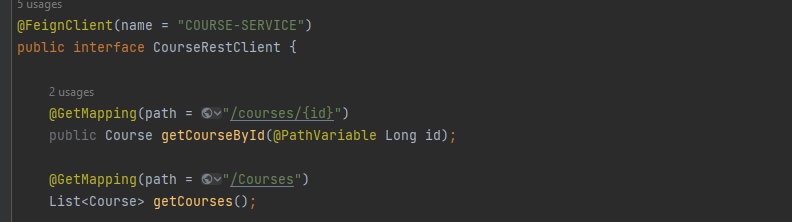


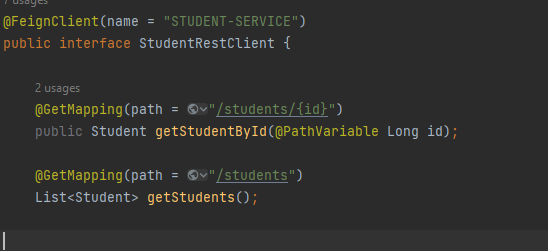
3-Discovery



4- relation entre les microservices en utilisant openfeign

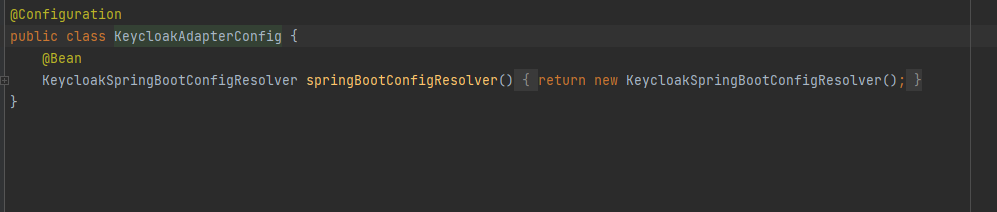


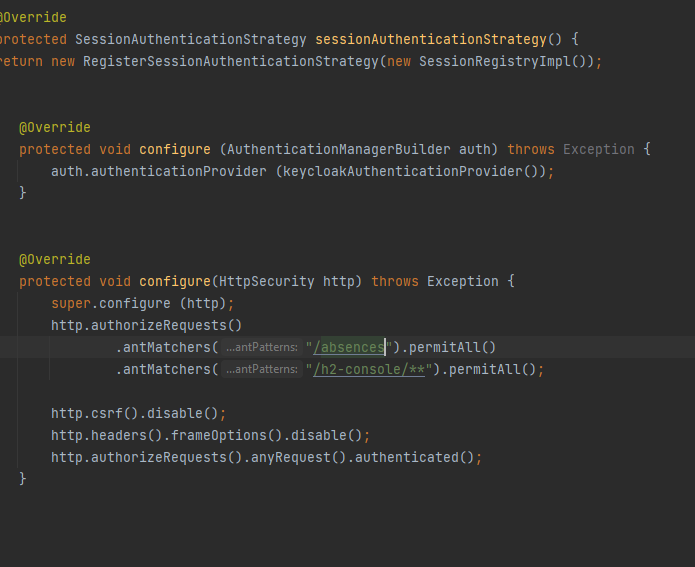




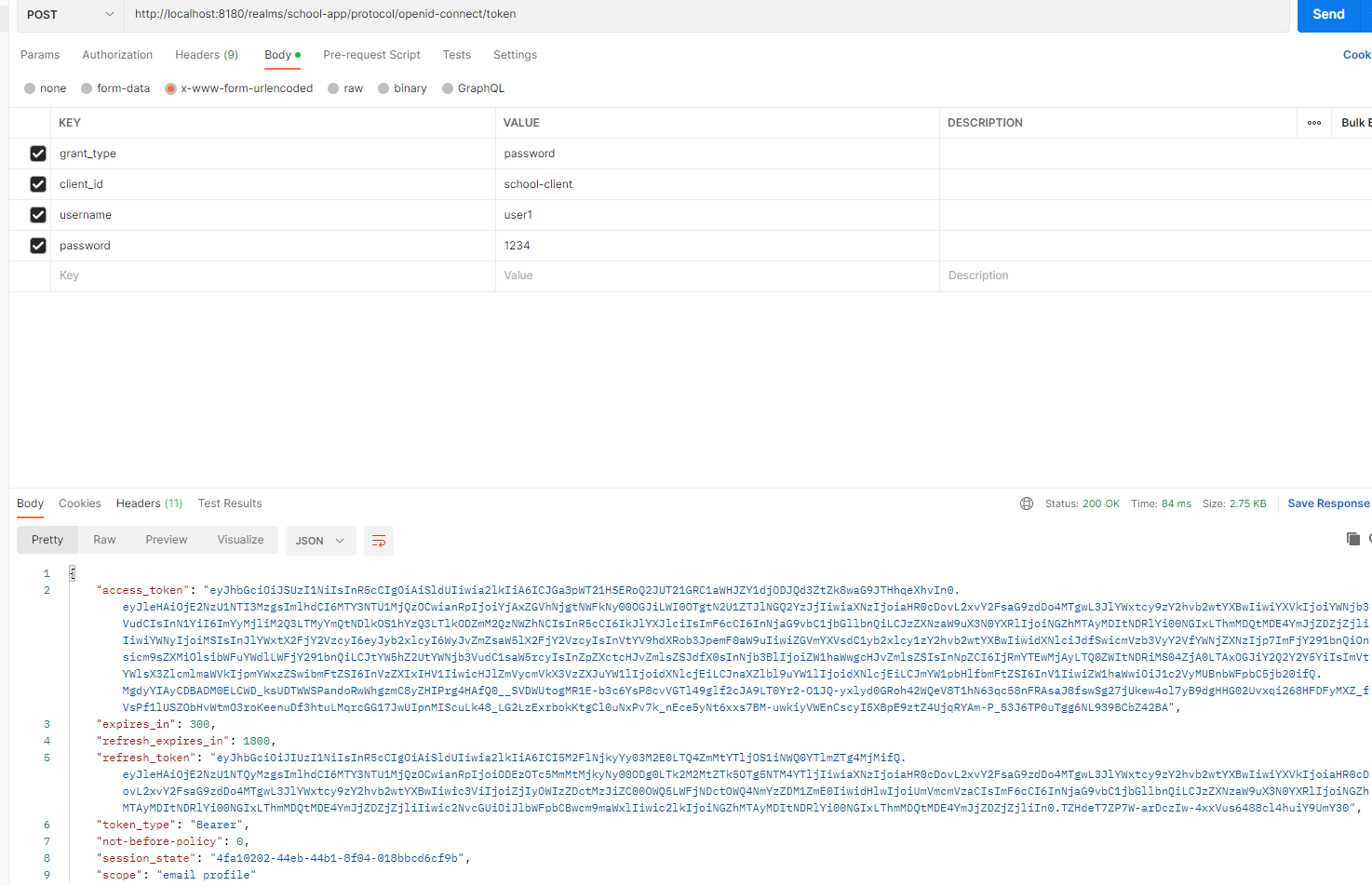
5-keyckloak





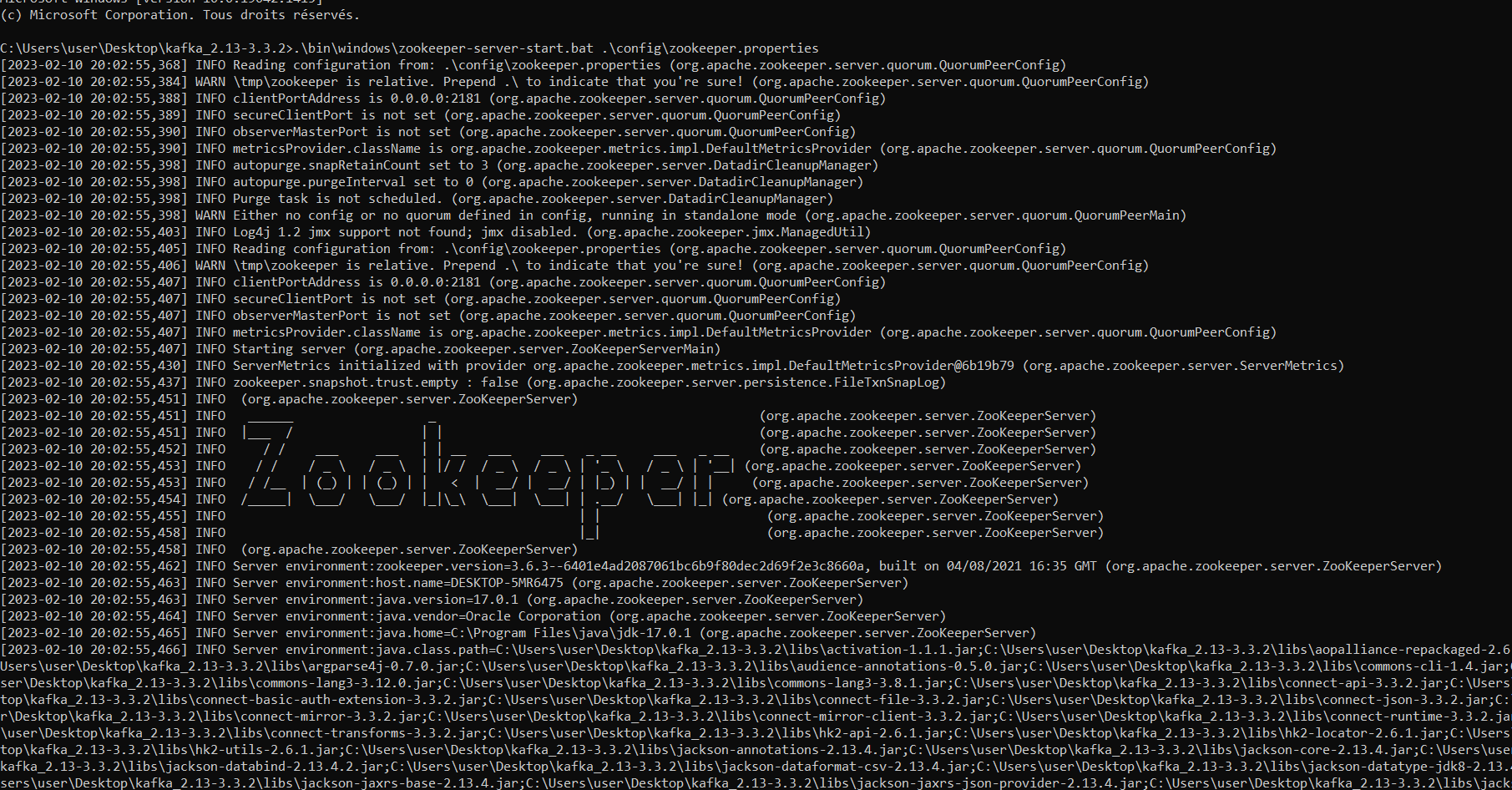


Pour générer le token on a fait recours à postman

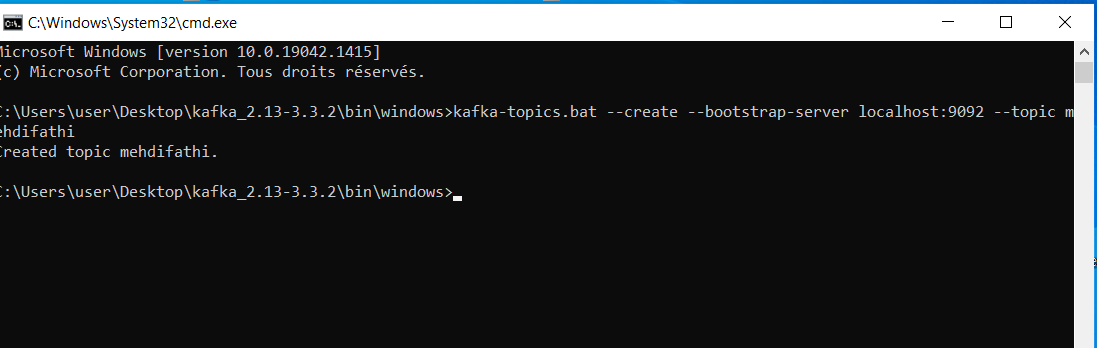


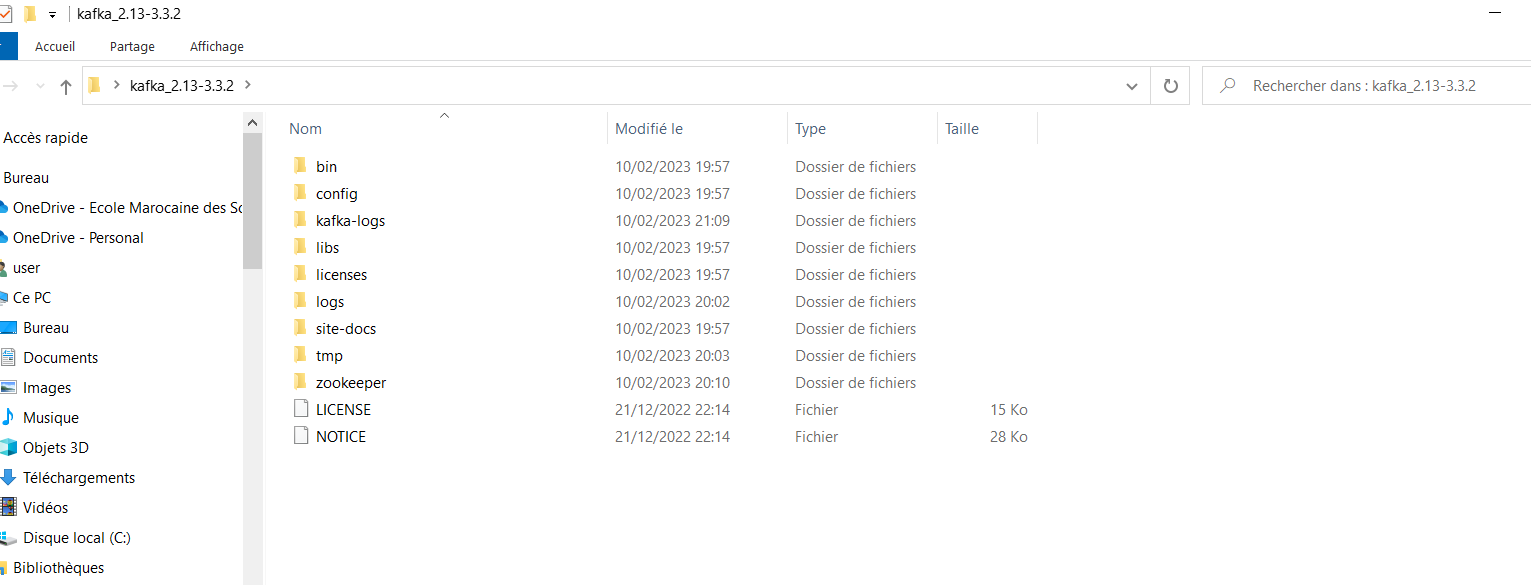
6-Kafka

6.1 KAFKA INTEGRATION

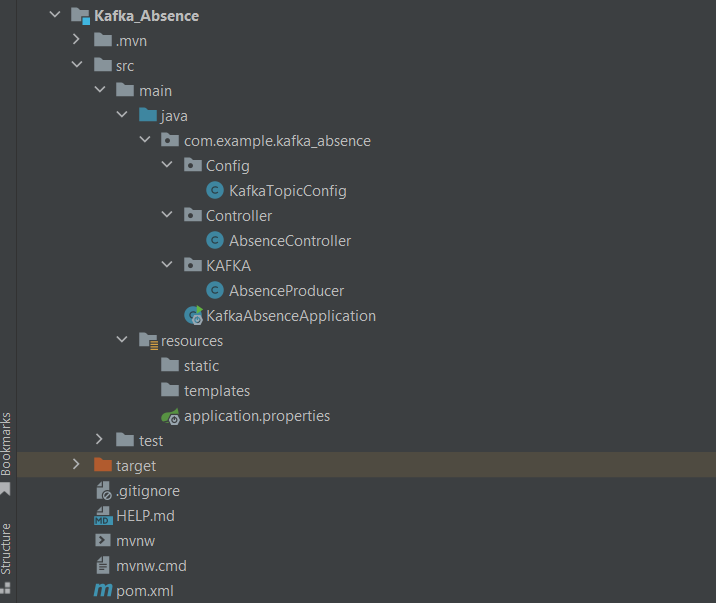


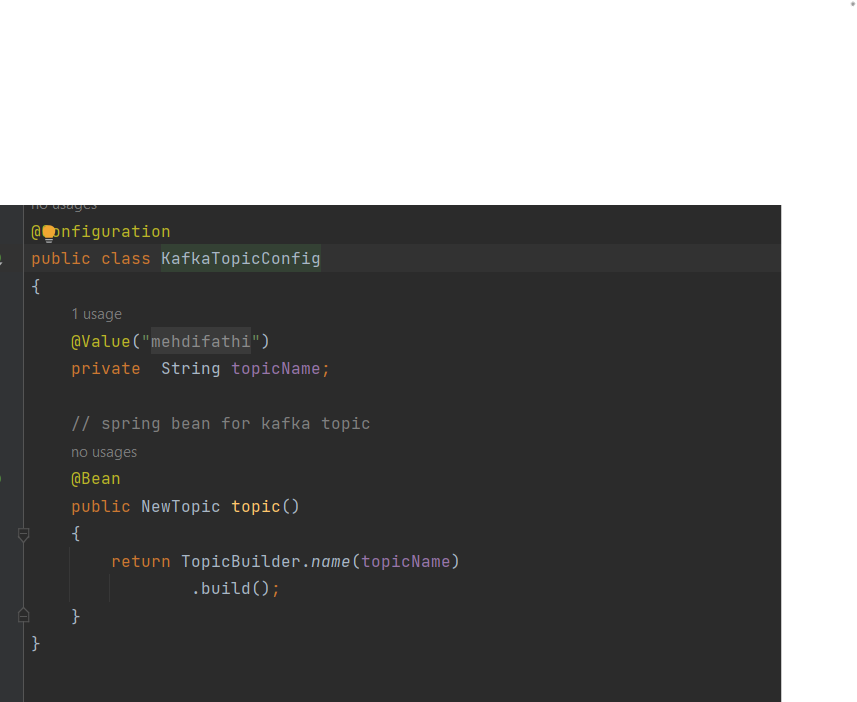
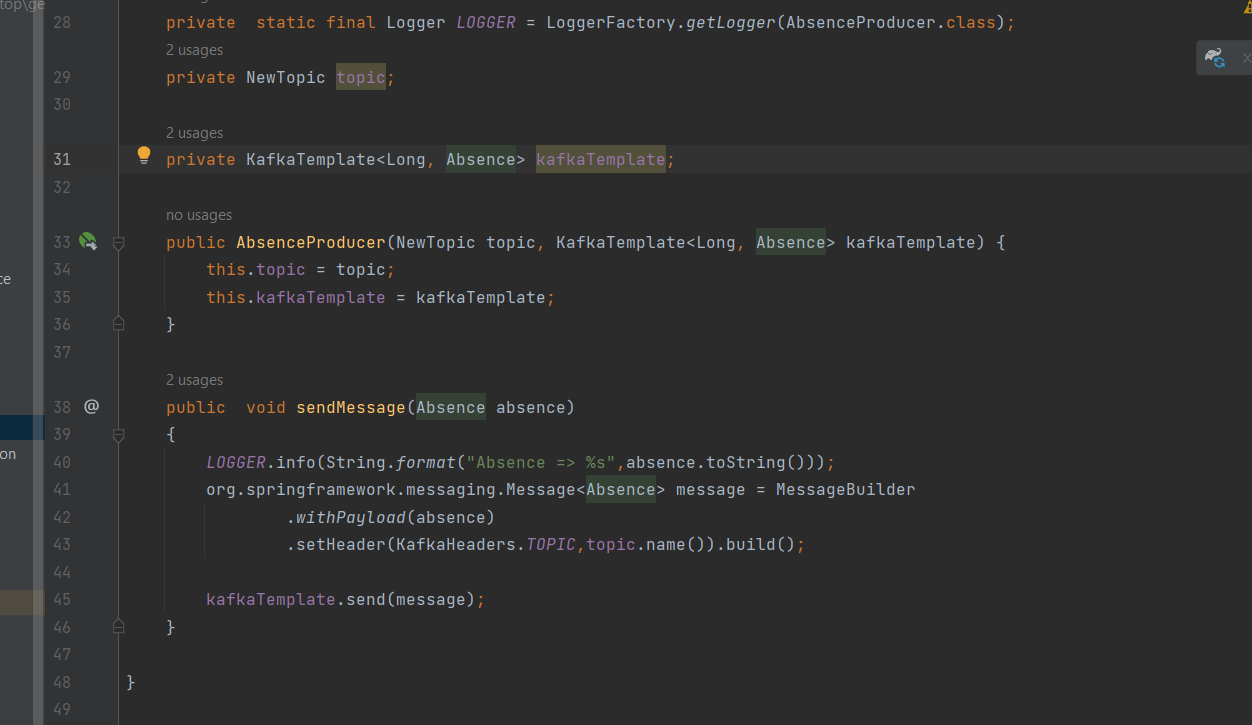


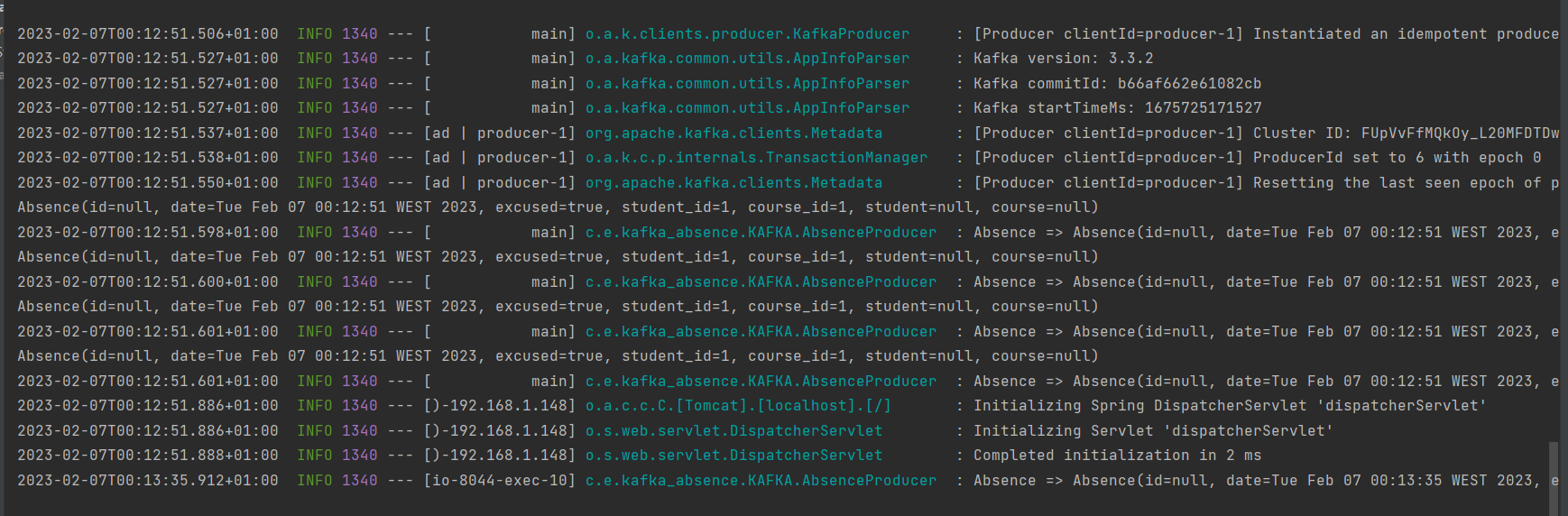
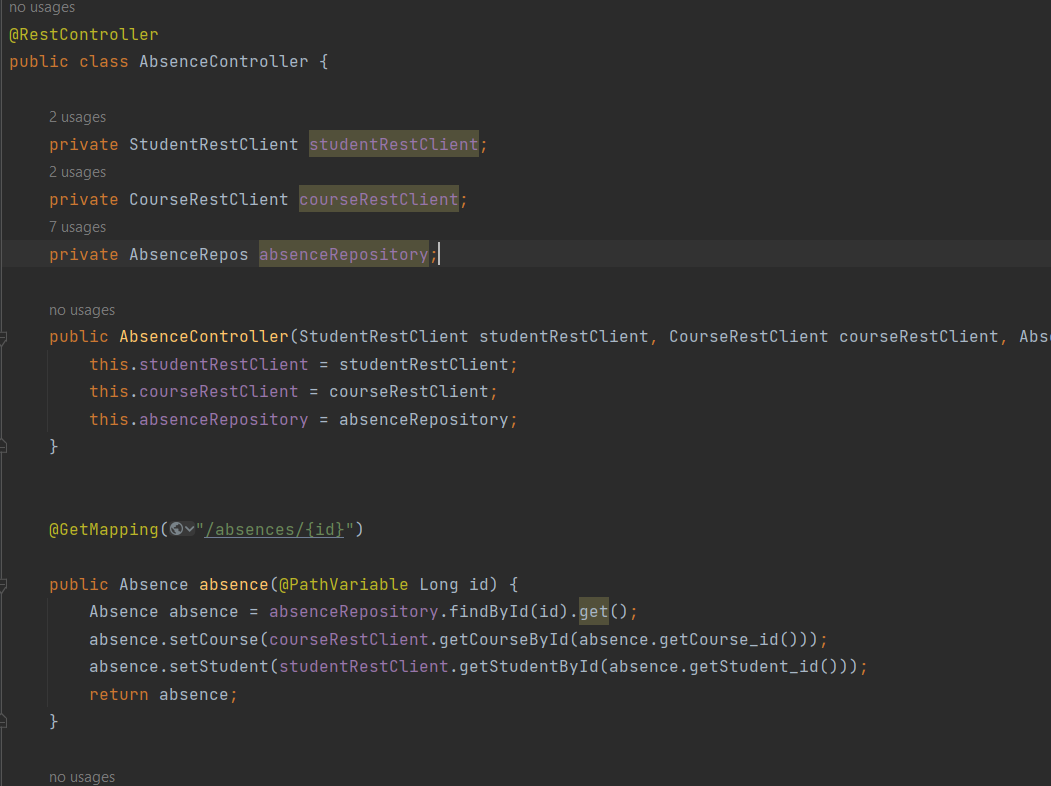


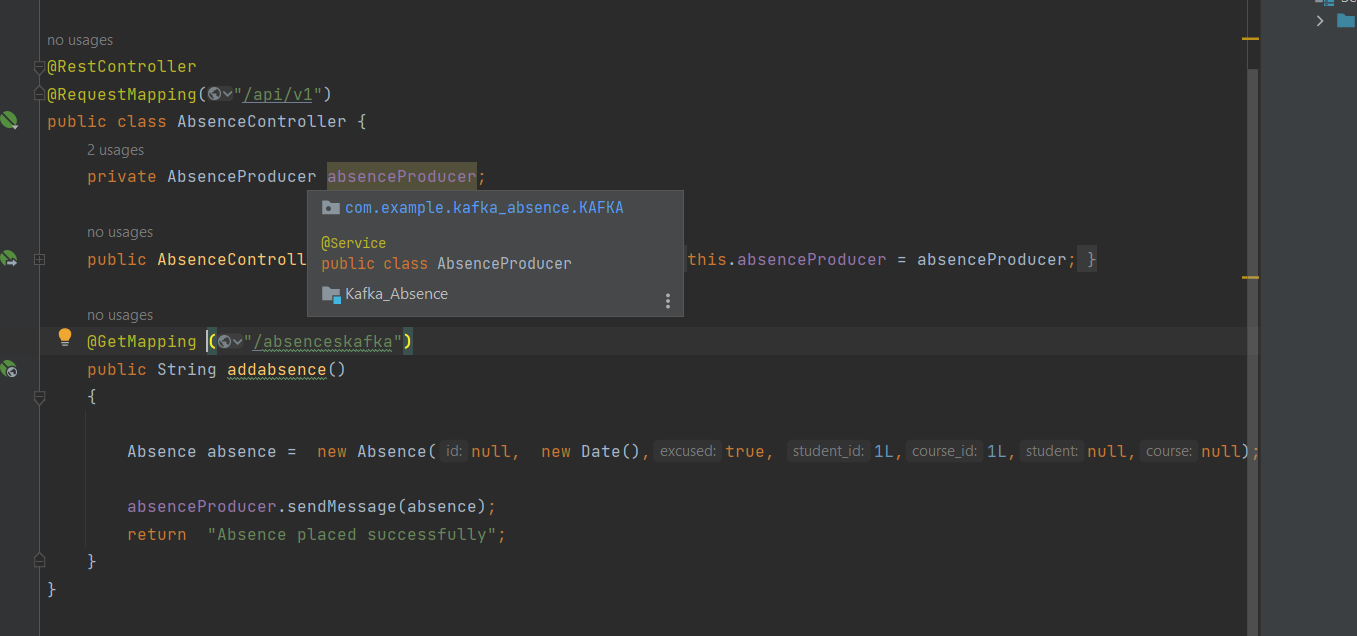
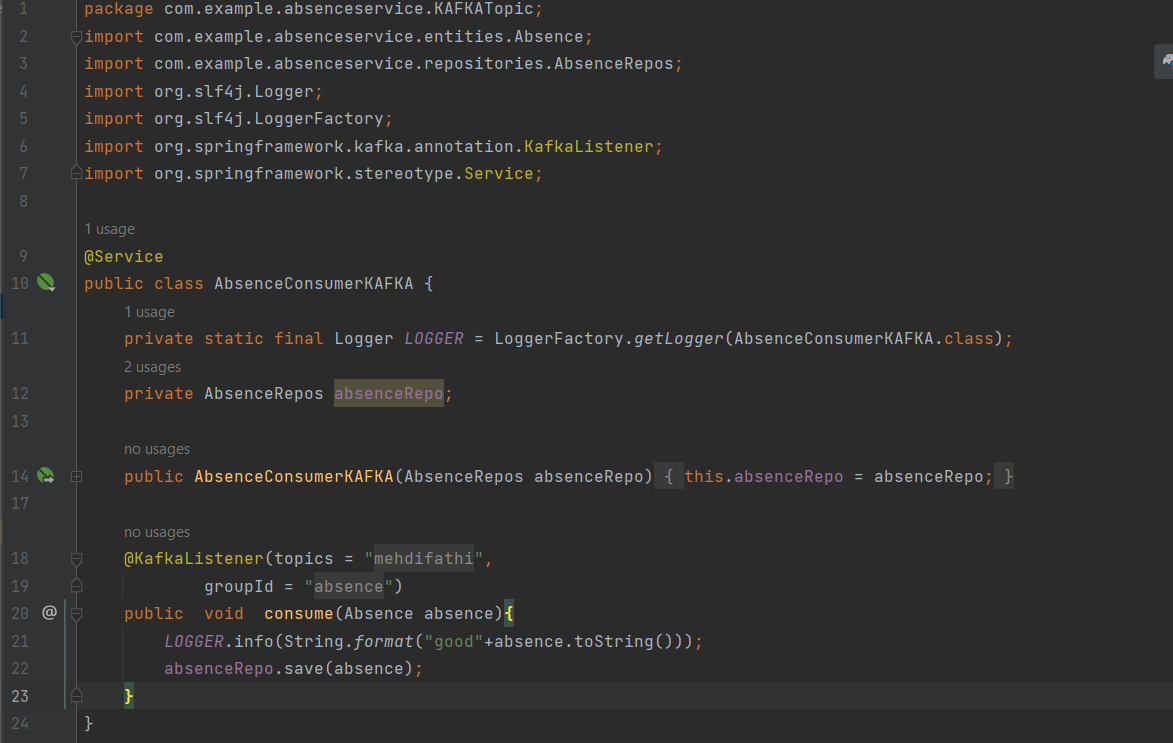


6.1 KAFKA Absence Service

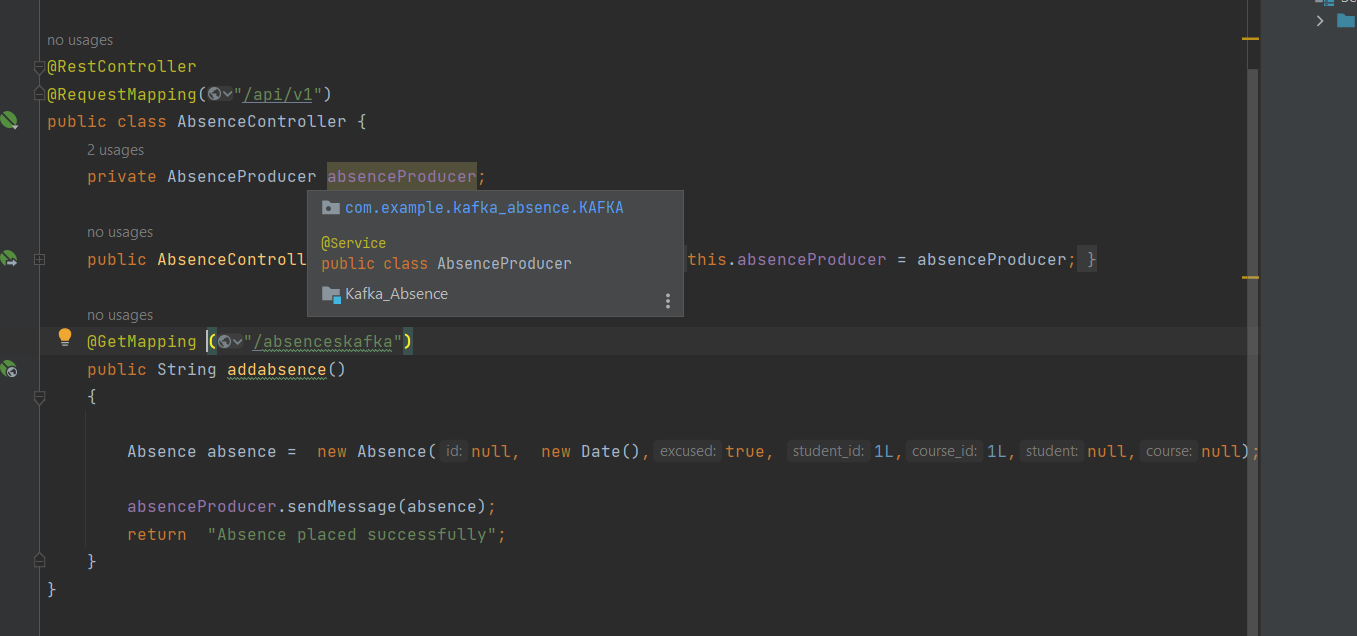


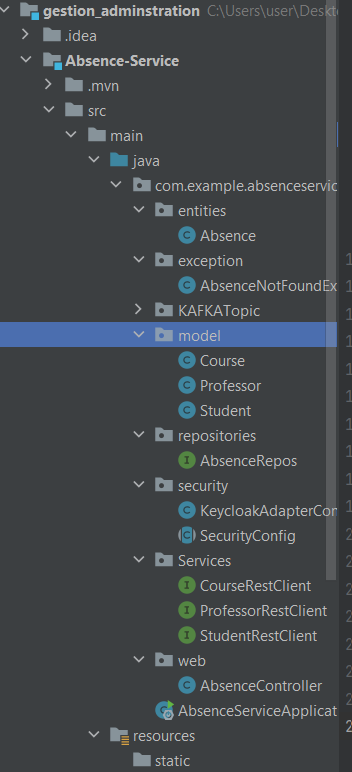




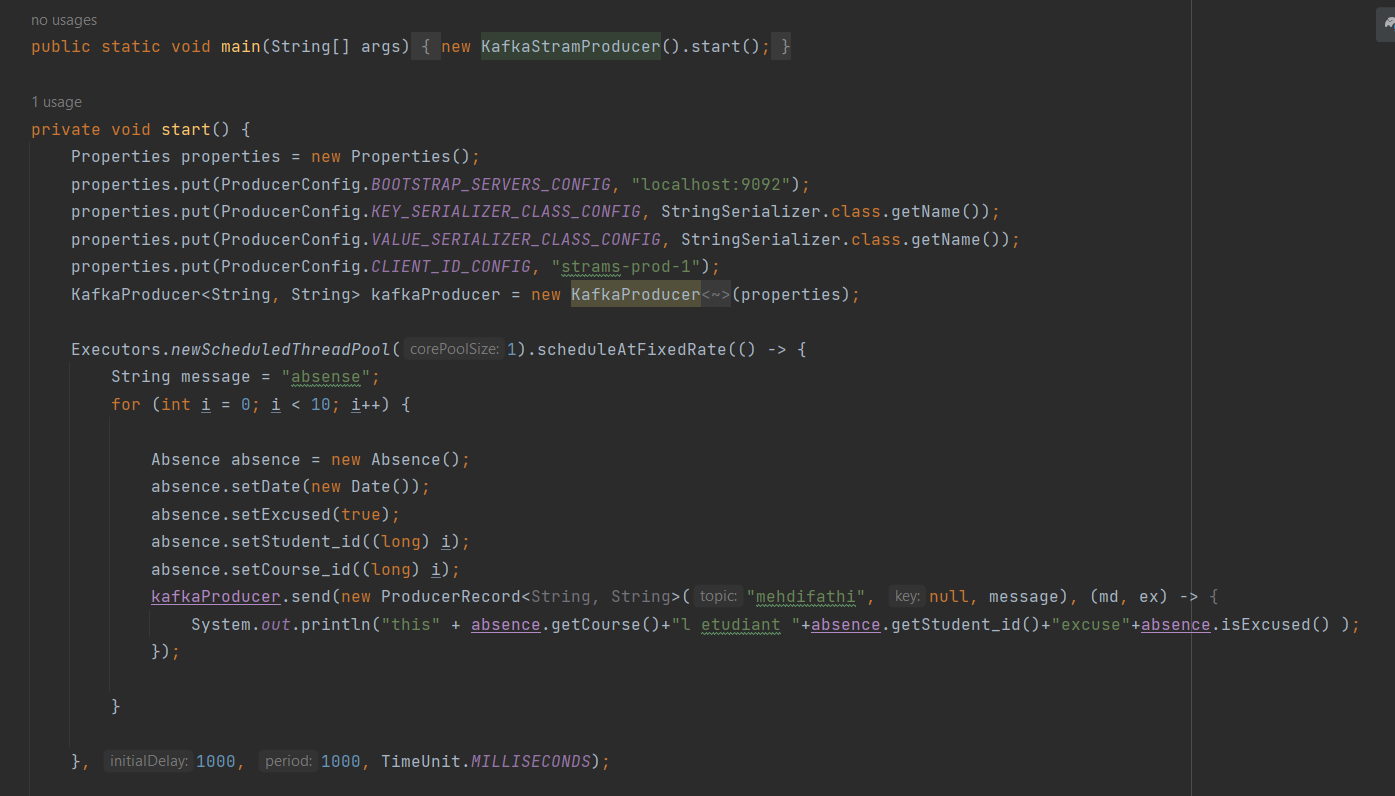
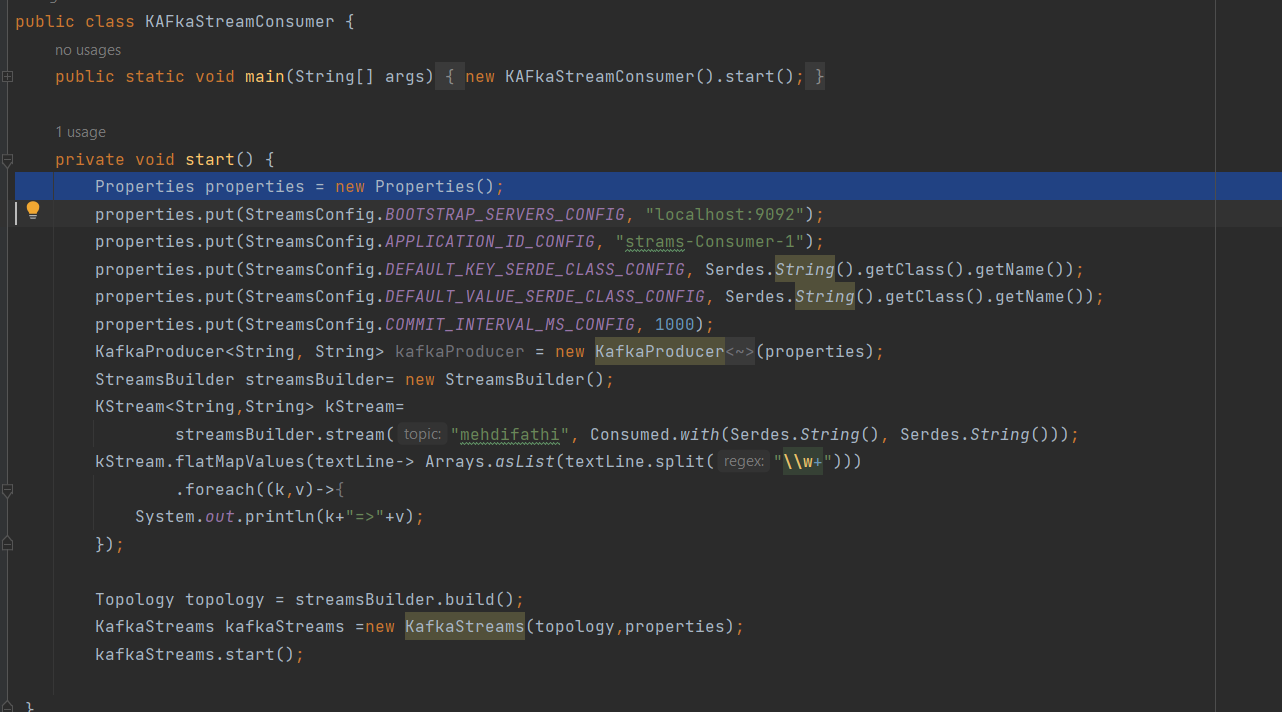
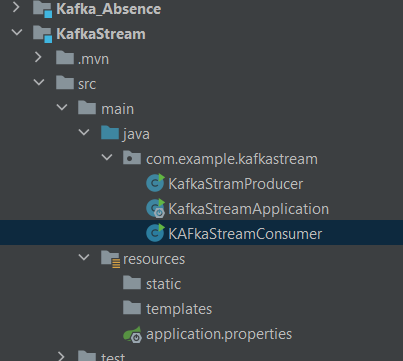


6.1 KAFKA Absence-Service





6.1 KAFKA Steam



1. *Conclusion*

En conclusion, notre projet de gestion d'école basé sur une architecture de microservices avec le framework Spring a été une réussite. Nous avons réussi à développer une application moderne, évolutive et facile à maintenir pour la gestion des activités de l'école.

L'utilisation de technologies telles que Spring Cloud Gateway, OpenFeign, Eureka et Kafka Streams nous a permis de construire une solution flexible, scalable et robuste pour répondre aux besoins actuels et futurs de l'école.