Diagramme de classes

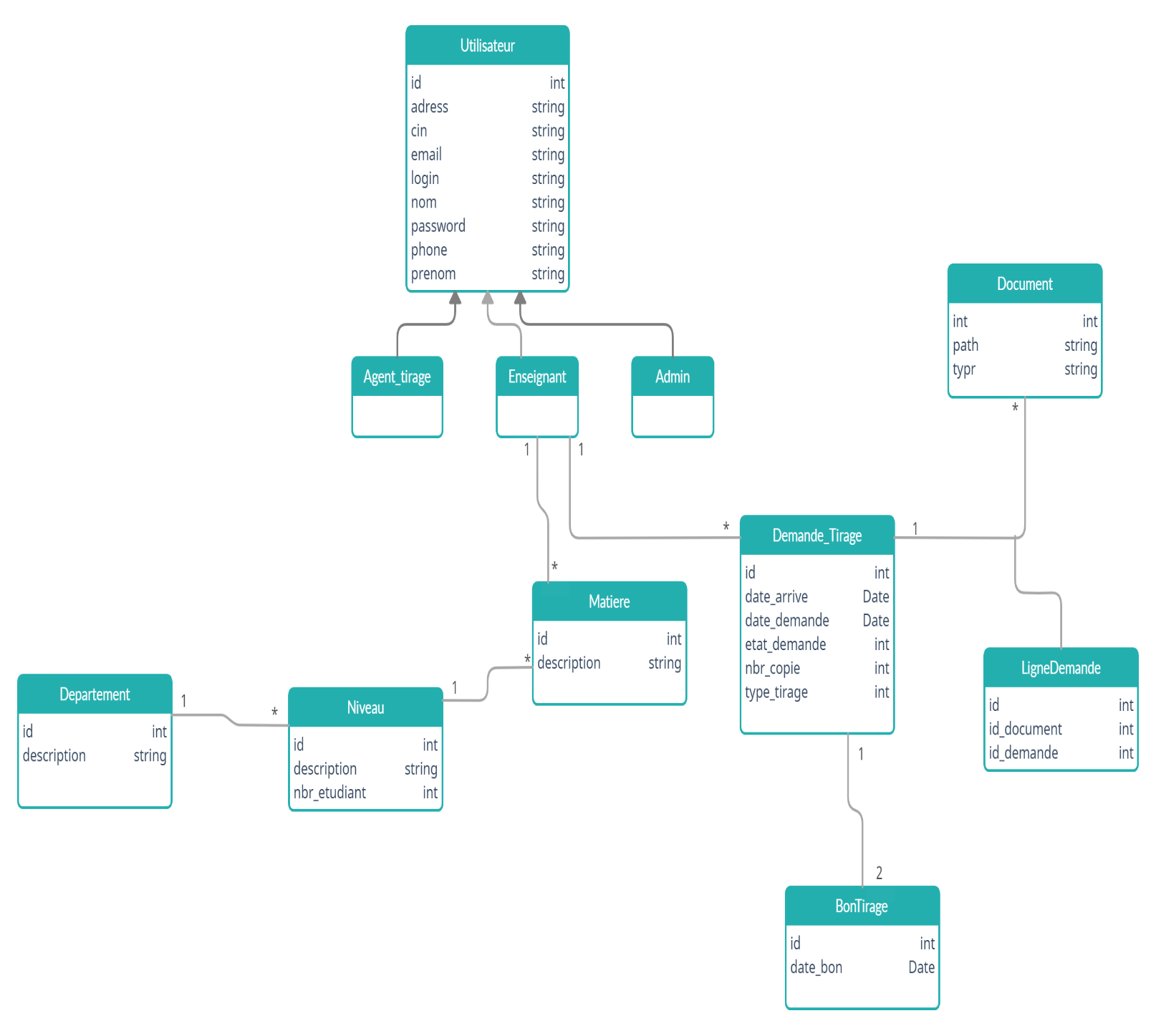
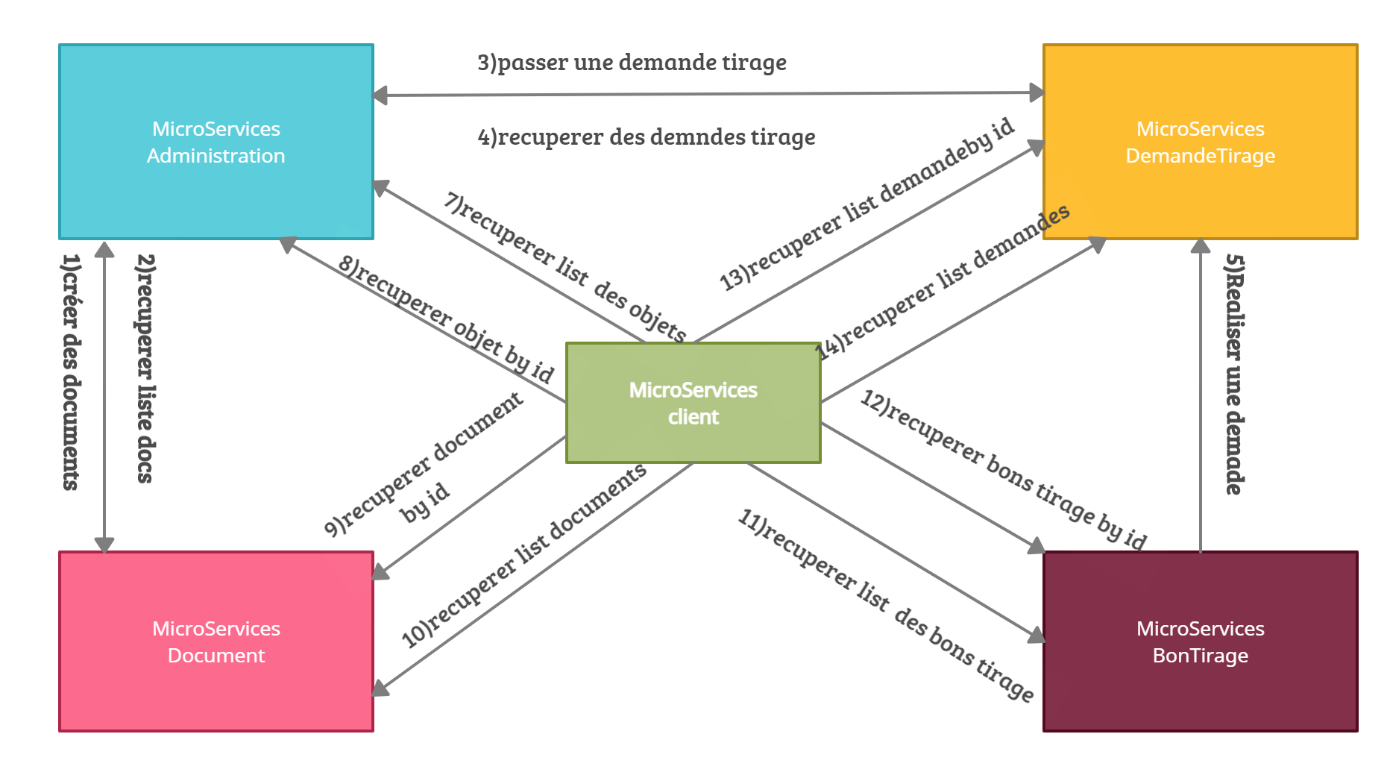
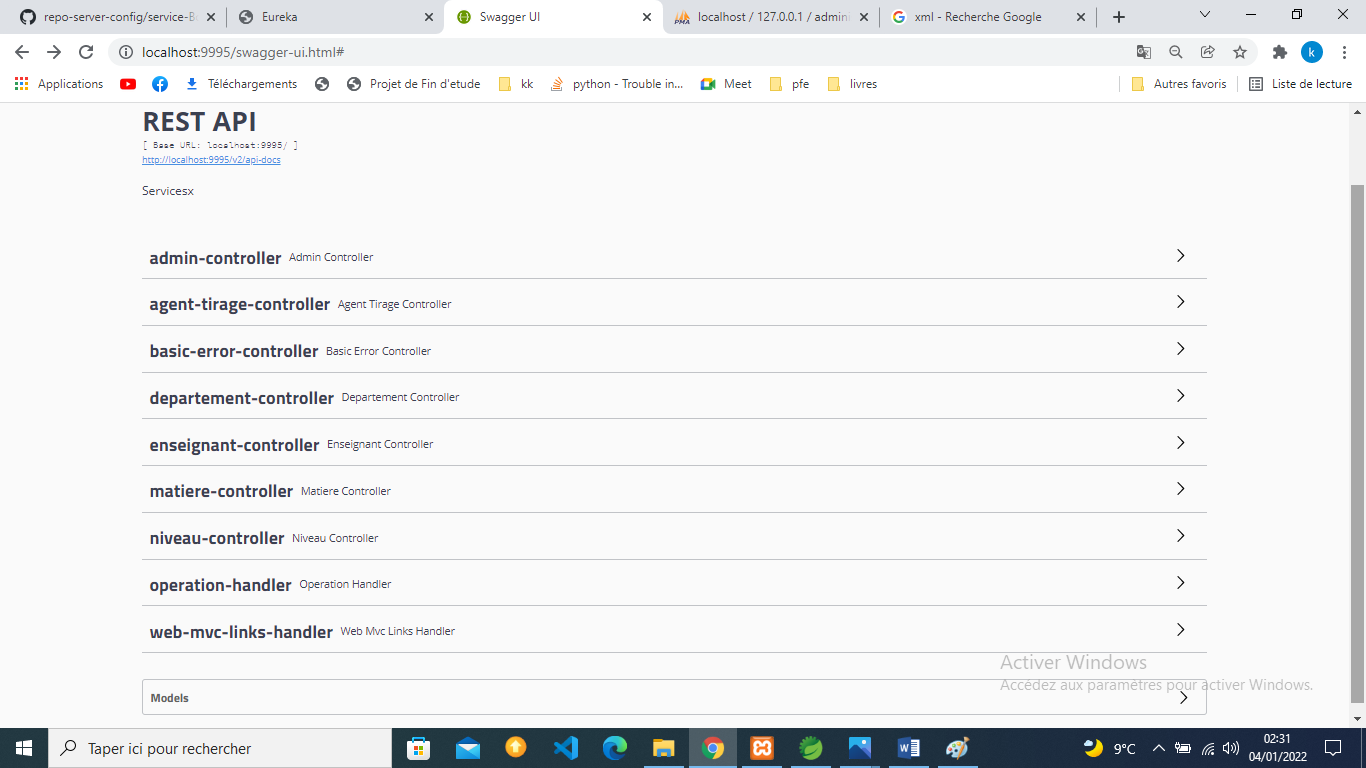
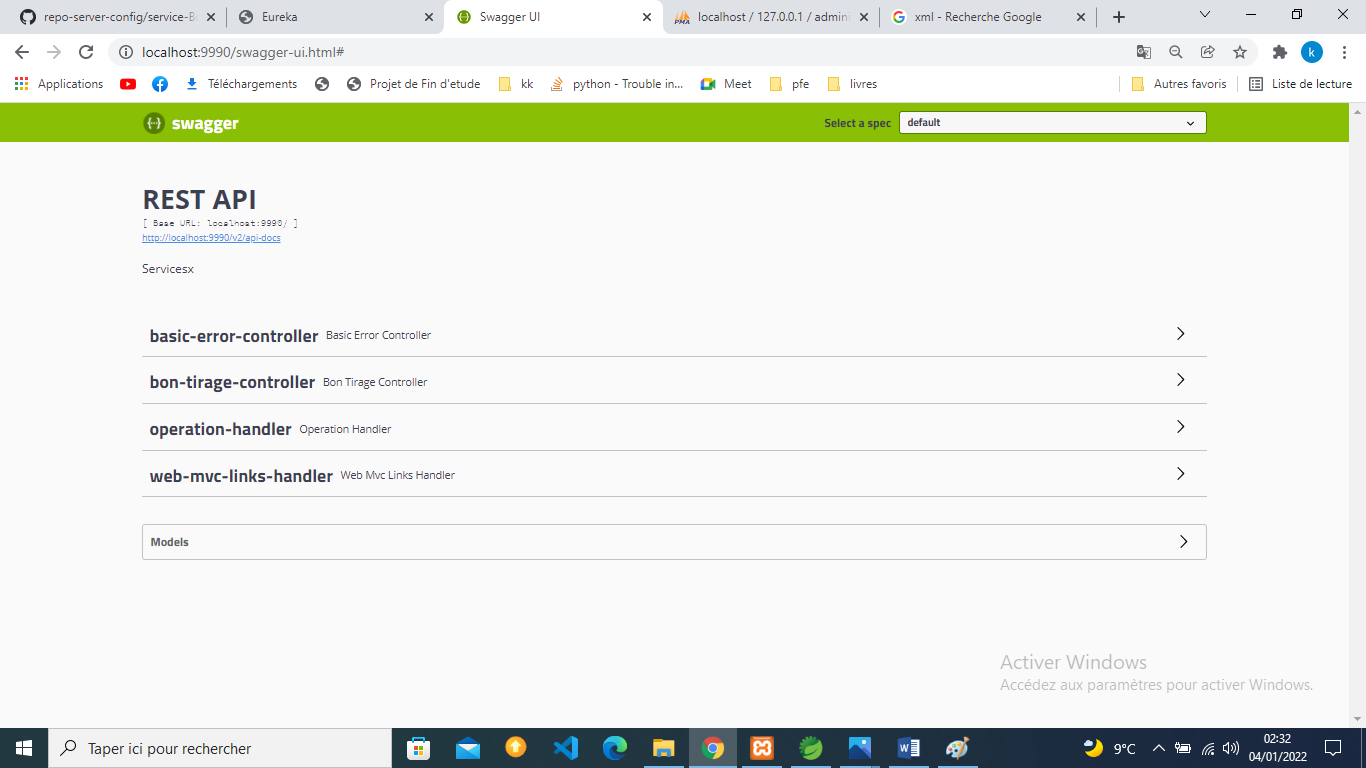
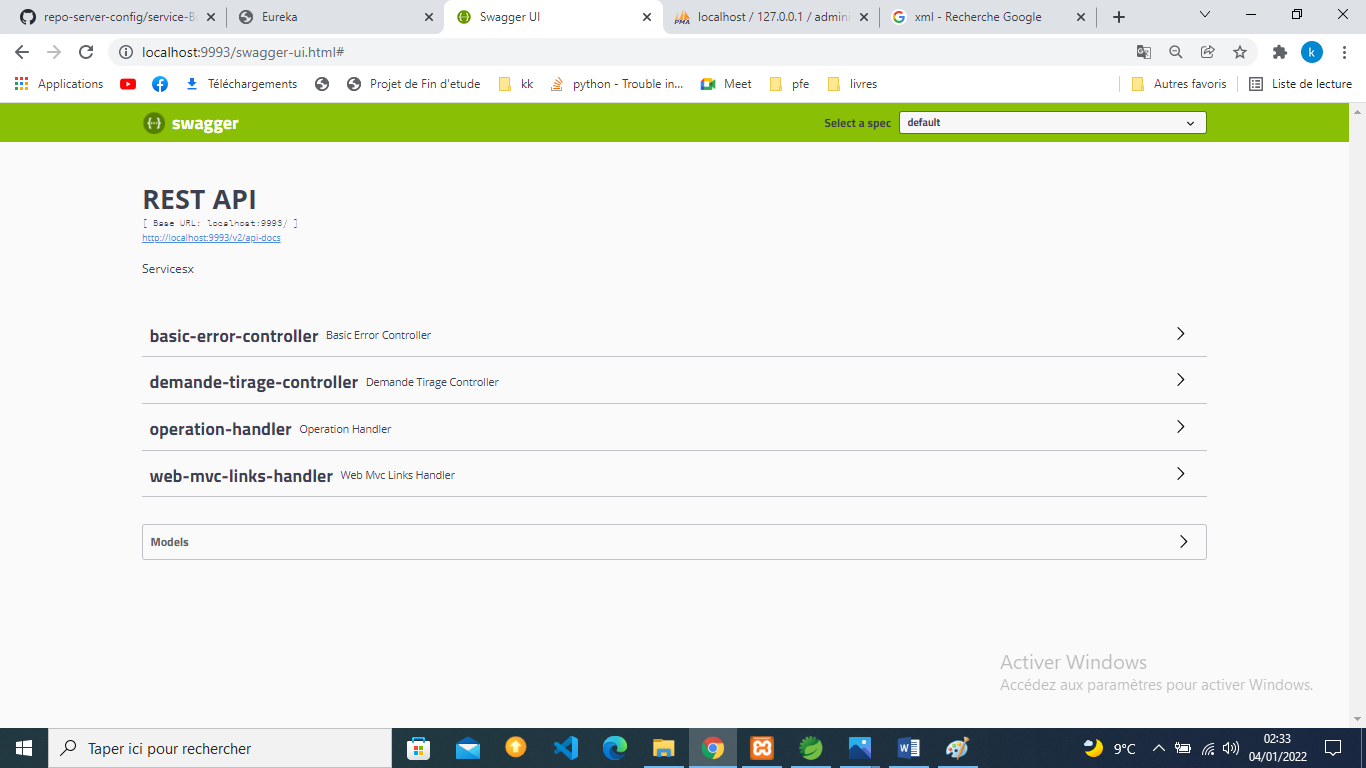
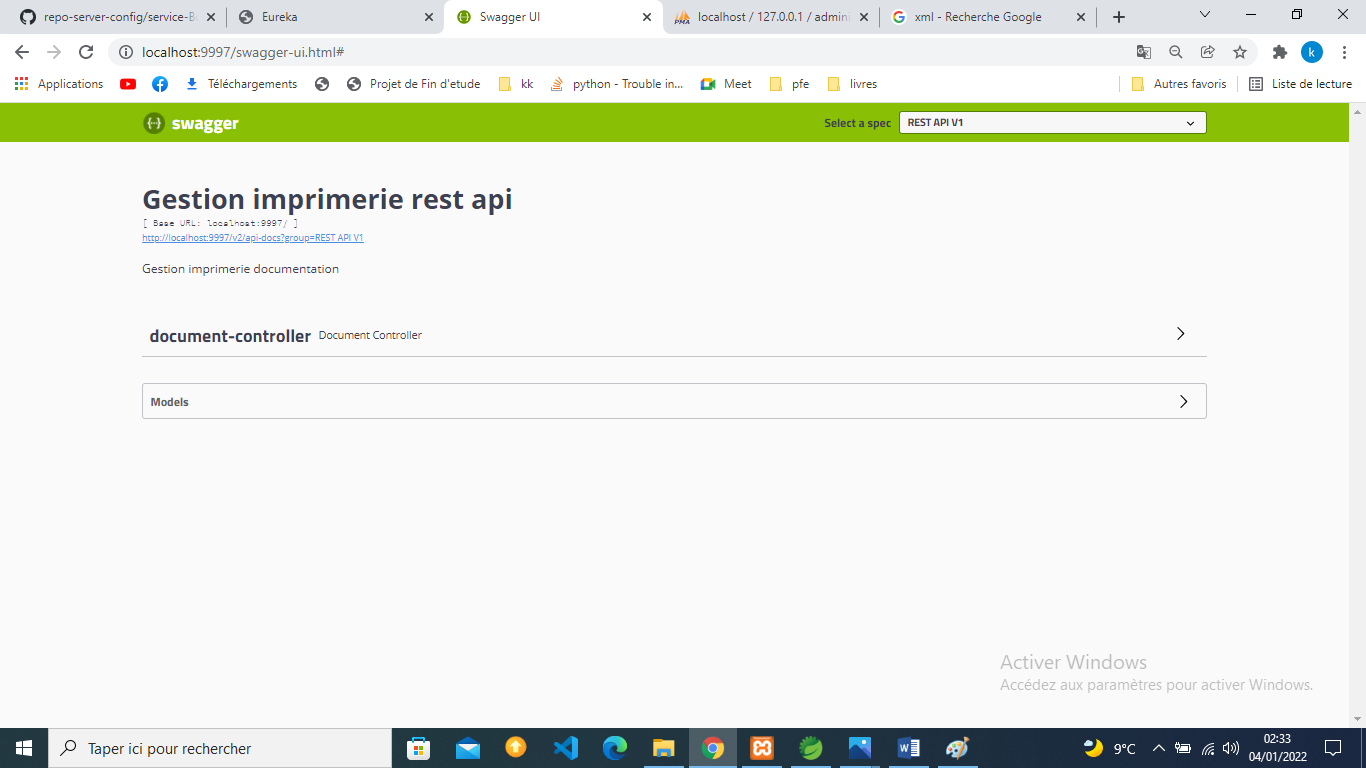


Diagramme de l’application



Faites communiquer vos Microservices grâce à Feign

**Feign** est un client HTTP qui facilite grandement l'appel des API exposées par les différents Microservices. Il est donc capable de créer et d'exécuter des requêtes HTTP basées sur les annotations et informations que l'on fournit. C'est un peu l'équivalent en code de Postman.

Activer feign dans le microservice appelé on ajoutant

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

rendez-vous à ClientApplicaion et on ajoute l'annotation @EnableFeignClients

creation interface proxy avec l’annoation feignClient(name = ‘nom-microservice-appelé’,url=’@ :numbrOfPort’) :pour regrouper les requéte

contient les signatures des méthodes appelé

Gestion et propagation des erreurs

Feign génére le fameux code 500 a tous les exceptions du web service

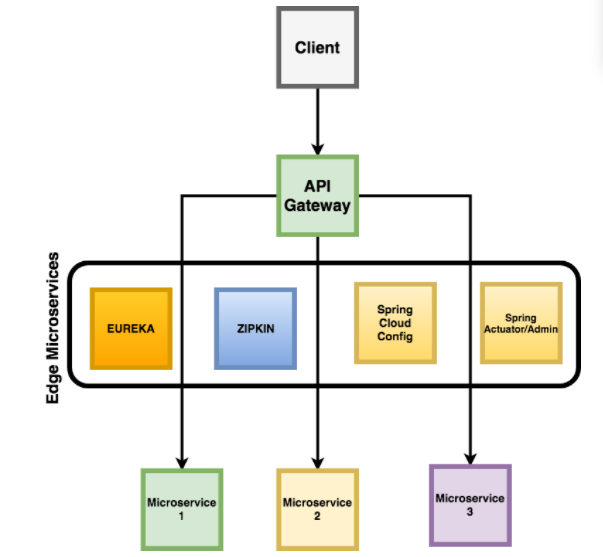
Il faut donc decrypter ou bien decoder l’exception génerée par feign

Feign propose une interface nommée ErrorDecoder pour decoder la reponse (exception) http

Il suffit d’implémenter l’interface errorDecoder dans le class custom error encoder pour decoder l’exception et lancer le vrai exception spécifique (runtim exception spécifique)

Les Edge Microservices sont des Microservices spécialisés dans l'orchestration des Microservices centraux responsables de la logique de l'application.

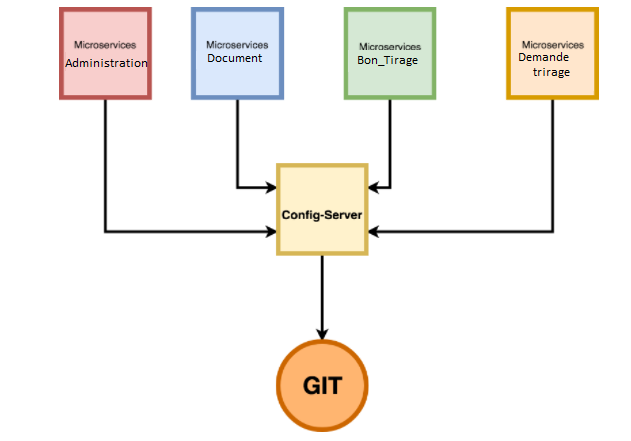
Les Edge Microservices les plus populaires sont ceux publiés et utilisés par Netflix pour sa plateforme. Ils sont en grande partie regroupés sous Spring Cloud



1. Configuration

Le edge microservice « Spring cloud config » permette de centraliser tous les fichier de configuration dans un depot git et se positionne comme serveur de fichier de configuration

On n’est pas besoin d’arrêter nos instances pour mettre a jour application.propreties



1-Ajouter dépendance spring cloud config puis ajouter dans application.proreties :

Server.port/spring.application.name/spring.cloud.config.git.uri =@repogit

Et ajouter @enableConfiServer + @refreshScope dans application.propreties

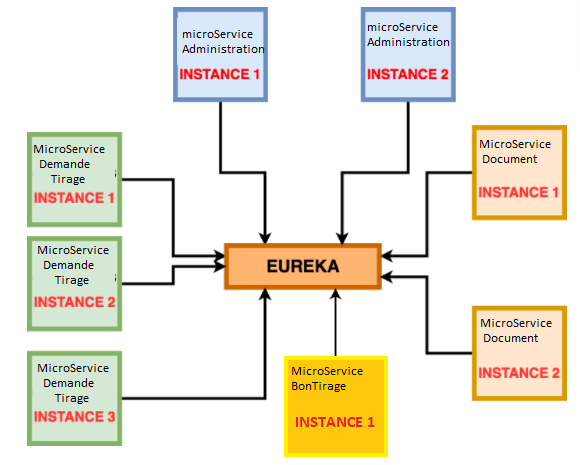
2-Renommer application.propreties avec nomSprig.propreties + push git

3-Ajouter spring-cloud-starter-config dans tous les microsService et Ajouter bootstrap.proreties qui contient :

\*spring.application.name / spring.cloud.config.uri

1. La découvrabilité

Le edge microservice « eureka » permmette de grader les traces des url des intances disponible et vérifier régulièrement sont état leurs états.



-Ajouter les 2 dépendances Eureka server et config client et Ajouter @enableEurekaServer

-definir le serverPort/springApplicationName/eurekaClientServerUrlDefaultZone

-Ajouter coté microServiceMetier dépendance springCloudStarterNetflixEurekaServer

Puis spécifier [eureka.client.ServiceUrl.defaultZone=@serverEureka](mailto:eureka.client.ServiceUrl.defaultZone=@serverEureka) + Ajouter @enableDiscoveryClient

1. L’quilibrage de charge (Load balancing)

Le edge microservice ribbon est un load balancer coté client qui permette a chaque microservice de faire l’appelle a un autre qui est disponible afin d’équilibrer le charge

(choisir a tour de role)

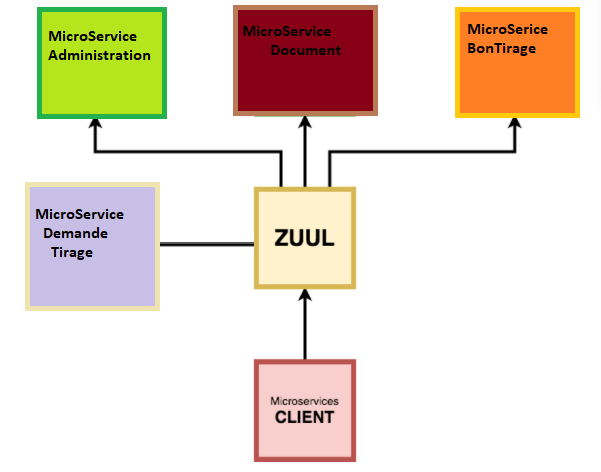
-ajouter dépendance spring.cloud.strater.netflix.ribbon

Puis ajouter a l’interface proxy @rebbonClient(‘name=applicationName’)

+Ajouer au propreties nomMicroService.ribbon.listOfServers=localhost :9001,…,…

1. API Gateway

Le edge microservice zuul est un api gateway qui permette d’installer un point d’entrer unique au microservice : parmis ses avantage la sécurité .



Ajouter Dépendance zull + Ajouter @enableZuulProxy + Ajouter les configuration necessaire dans applicationPropreties

Zuul fontionne nativement avec eureka

Par exemple Pour récupérer la liste des demande il suffit localhost :PortZuul/nomMicroService/DemadeTirage

Dans notre cas j’ai travaillé directement avec cloud Gateway

Puis

@Bean

RouteLocator routes(RouteLocatorBuilder builder) {

//http://localhost:8989/GeAllEnseignant

return builder.routes()

.route(r -> r.path("/administration/\*\*")

.filters(

f->f

//pour transformer le path

.rewritePath("/administration/(?<segment>.\*)","/${segment}")

//cercuit breaker

.hystrix(h->h.setName("administration")

.setFallbackUri("forward:/defaultEnseignant")

)

)

.uri("lb://service-administration")

.id("r1"))

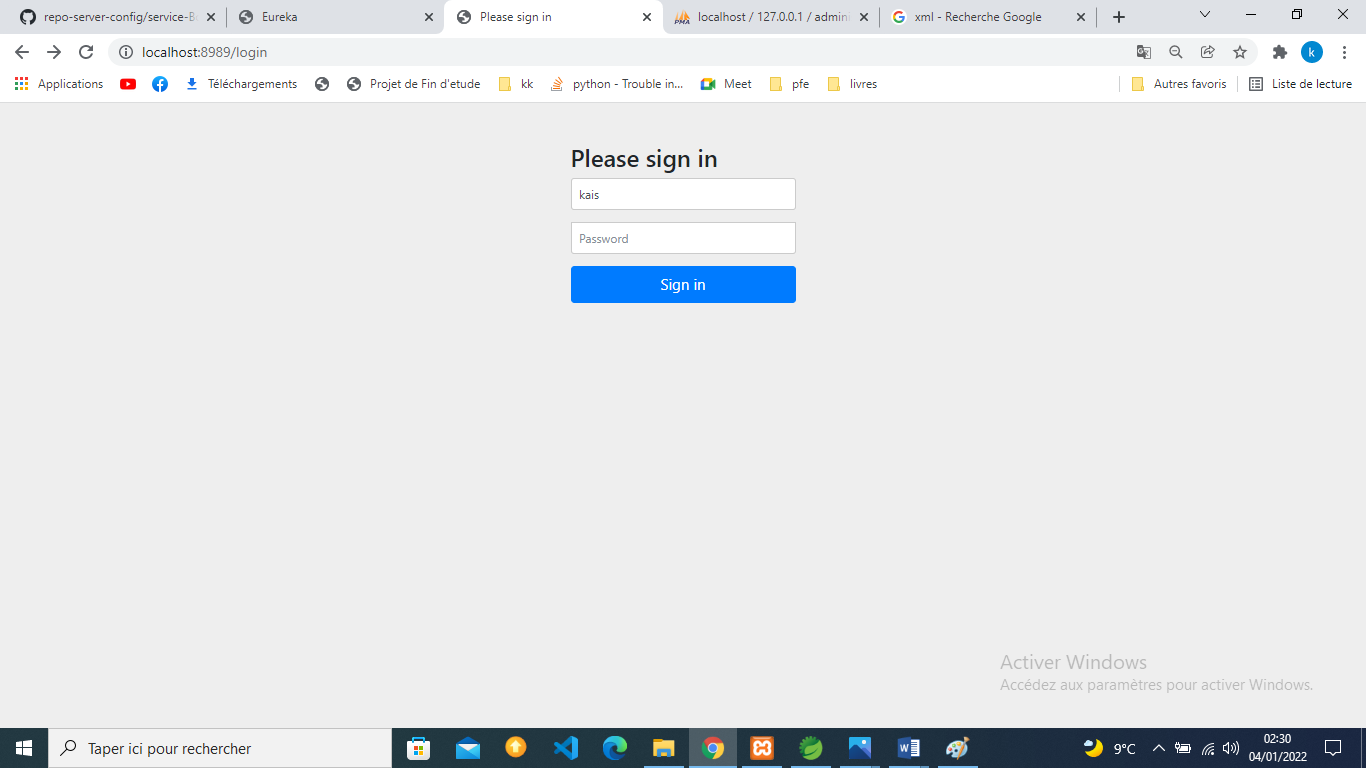
.build();

}

Comme le gateway est le seul point d’entrée donc il suffit uniquement de sécuriser sont accés

Donc Ajouter le dépendance spring security qui offre des mécanisme d’authentification

Plus que modifier ApplicationPropreties springSecurityUserName/SpringSecurityUserPassword



1. Tracage des requétes

Le edge microService zipkin permette de suivre le tracage des requétes puisque une requête traverse plusieur microService et en cas d’erreur c’est difficile de savoir a quel niveau se produite cette erreur

1-Ajouter spring sleuth aux microServices springCloudStraterSleuth

2-il faut Ajouter un class de configuration sleuthConfig avec une methode DefaultSampler qui retourne sampler.Always\_sample 🡺identifier tous les requétes avec des id .

Donc a chaque fois le log d’un service a un autre contient [nomApplication,IDTracage,SpanId,Export :trueORfalse]

3-telecharger zipkin et le lancer http://localhost:9411

4-Ajouter spring-cloud-sleuth-zipkin a tous les microServices

1. Debugger l’application grace a spring Actuator

Ajouter dépendances spring boot actuaor au microServices

\*Actuator fonctionne en expose des endpoint : {health/info/env/metric/beans…}

Ajouter dans ApplicationProperties

Management.endPoints.web.exposure.include=\*

Spring Admin est un edge microService qui regroupe tous les retours de spring actuator et expose uninterface graphique

Il possède 2 version serveur/client

Ajouter dépendance spring Admin puis @EnableAdminServer puis liée ce microService avec eureka (pour récupérer la liste des microServices)

Ajouter dans ApplicationProperties

Eureka.instance.health-check-url-path= /acuator/health

Résumé

Les Edge Microservices sont des Microservices qui s'occupent principalement de l'orchestration des Microservices métier contenant la logique de l'application.

Feign est un outil permettant de faire communiquer les Microservices très simplement entre eux, en générant automatiquement les requêtes HTTP adéquates à partir de classes appelées proxies.

Les fichiers de configuration des Microservices peuvent être externalisés vers un dépôt GIT, par exemple grâce à Spring Boot Config. Cet Edge Microservice va alors récupérer la dernière version de chaque fichier de configuration et la servir au Microservice correspondant.

Pour forcer l'actualisation des données d'un fichier de configuration dans  un Microservice, il suffit d'envoyer une requête POST vers l'endpoint/refresh exposé par Actuator.

Eureka est un autre Edge Microservice qui permet de garder un registre de toutes les instances disponibles des Microservices métier. Chaque Microservice qui intègre Eureka ira s'enregistrer à chaque démarrage ou lancement d'une nouvelle instance. Eureka vérifie ensuite régulièrement que les Microservices sont toujours disponibles afin de mettre à jour son registre.

Ribbon est un équilibreur de charge côté client. Une fois intégré dans un Microservice, il est capable de communiquer automatiquement avec Eureka afin de choisir l'instance à appeler d'un Microservice donné. Un algorithme s'occupe ainsi de répartir les requêtes sur toutes les instances disponibles.

Lorsque vous avez un nombre important de Microservices, ZUUL se positionne comme un Microservice servant de point d'entrée unique aux autres Microservices. C'est ce que l'on appelle une API Gateway.

ZULL va permettre ainsi de réaliser des opérations communes à tous les Microservices, comme la sécurisation de l'accès, la mise en place d'un nombre d'appels limite à l'API, la transformation des requêtes afin de les enrichir avant qu'elles n'atteignent les Microservices cibles, etc.