L'intégration de Kafka et Keycloak dans le projet de microservices de réseau social

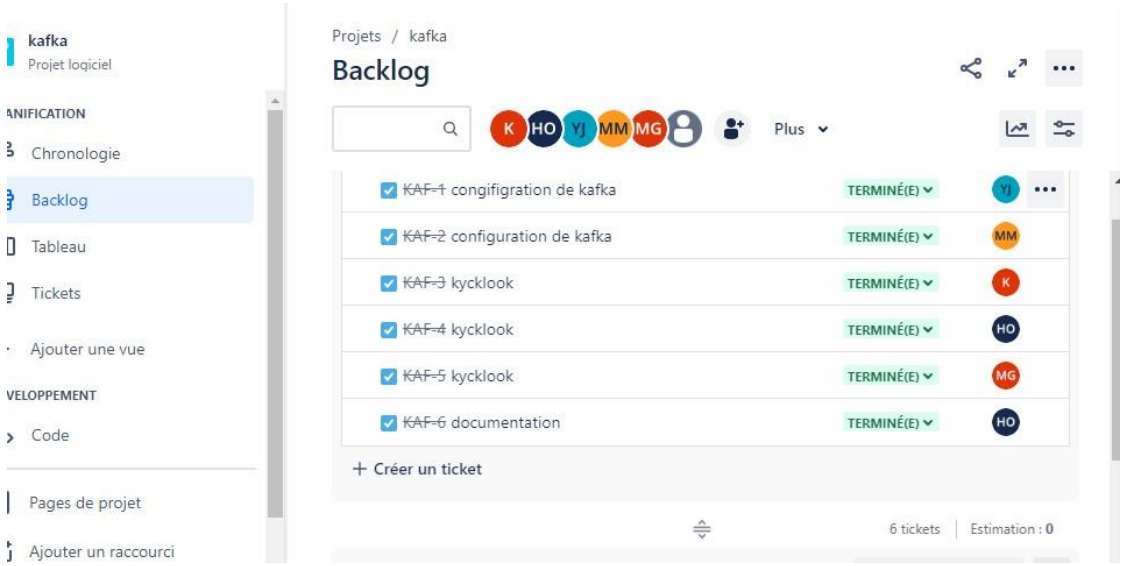
Introduction

Ce rapport présente les détails de l'intégration des technologies Apache Kafka et Keycloak dans notre projet de microservices de réseau social. L'objectif principal de cette intégration est d'améliorer la scalabilité, la sécurité et la réactivité de notre application, tout en offrant une expérience utilisateur fluide et efficace.

Outil de collaboration

Jira :

Pour réussir KANBAN, nous nous servons de 'JIRA' qui est un outil de gestion de projet en ligne. JIRA est une plateforme multifonction qui vise à faciliter la gestion de projet en aidant à suivre les tâches, identifier les blocages et partager des informations entre les membres d'une équipe. Il est basé sur une organisation des projets en tickets, chacune représentant des tâches. De plus, JIRA permet d'assurer le suivi des tickets et la d´définition d'un workflow adapté aux m´méthodes de travail. JIRA génère aussi des graphiques et visuels qui permettent de repérer en un coup d'œil l'état des différentes missions et d'identifier les problèmes à régler en priorité pour pouvoir avancer.



Intégration d'Apache Kafka

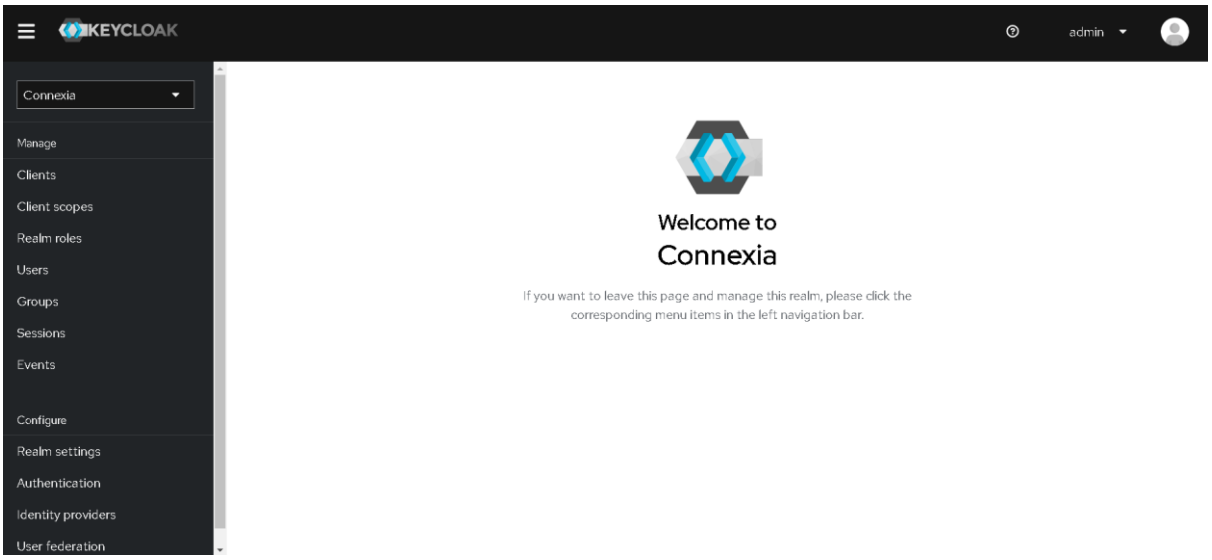
Apache Kafka est intégré comme courtier de messages pour faciliter la communication asynchrone entre les microservices. Chaque microservice produira des messages vers des sujets Kafka lorsqu'un événement se produira, par exemple, lors de l'inscription d'un utilisateur ou de la publication de contenu. D'autres microservices consommeront ces messages pour effectuer des actions ou des mises à jour pertinentes. Cette architecture assure une communication lâche et évolutive entre les composants.

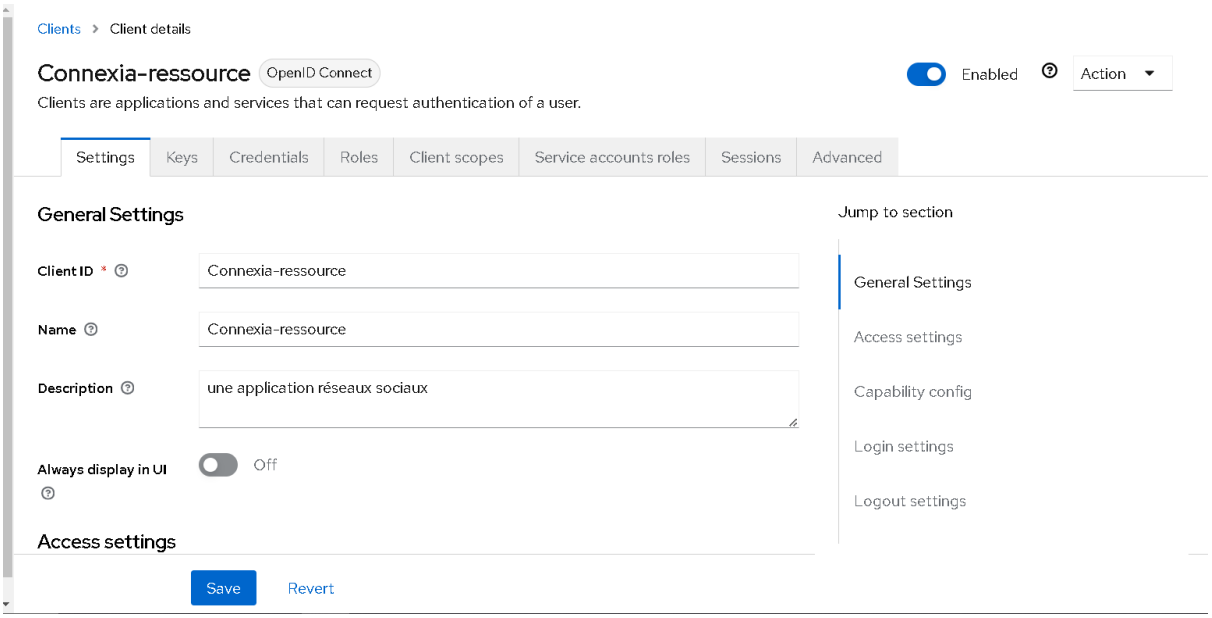
Intégration de Keycloak

Keycloak est intégré pour gérer l'authentification et l'autorisation des utilisateurs. Chaque microservice déléguera l'authentification des utilisateurs à Keycloak, garantissant ainsi un mécanisme d'authentification centralisé et sécurisé. Keycloak émettra des jetons JWT lors d'une authentification réussie, utilisés par les microservices pour autoriser les actions des utilisateurs en fonction des rôles et des permissions définis.

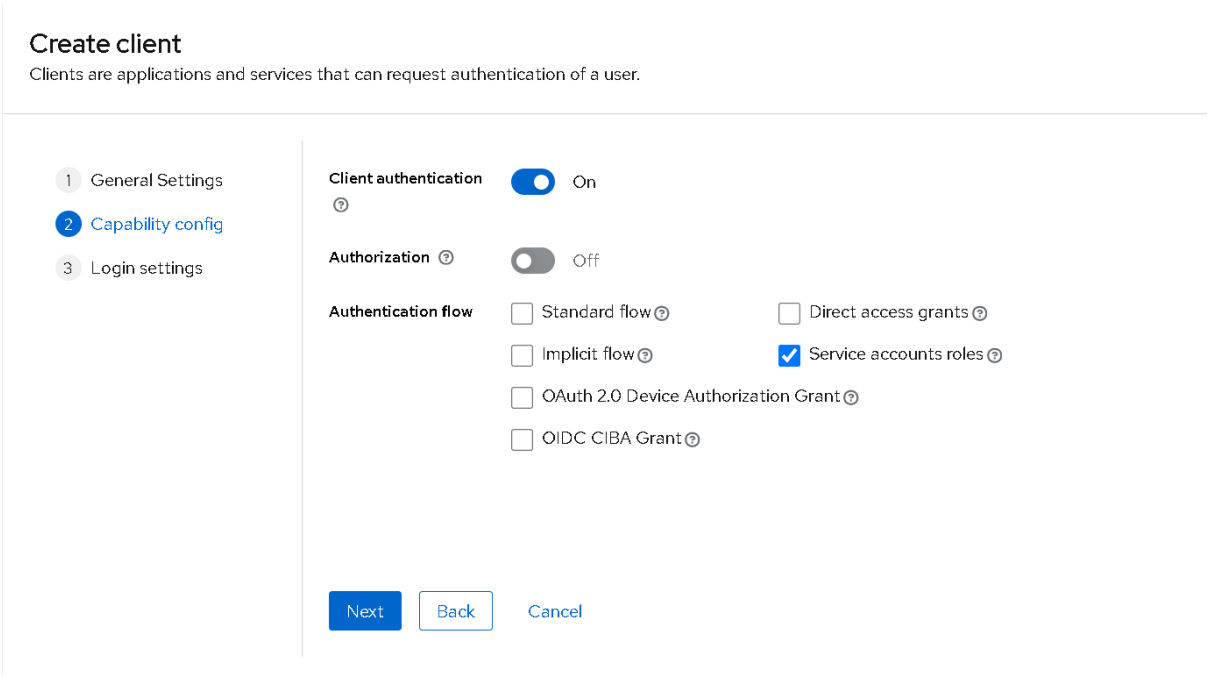
Les étapes :

1. Installation du keycloak et la création d'un nouveau realm

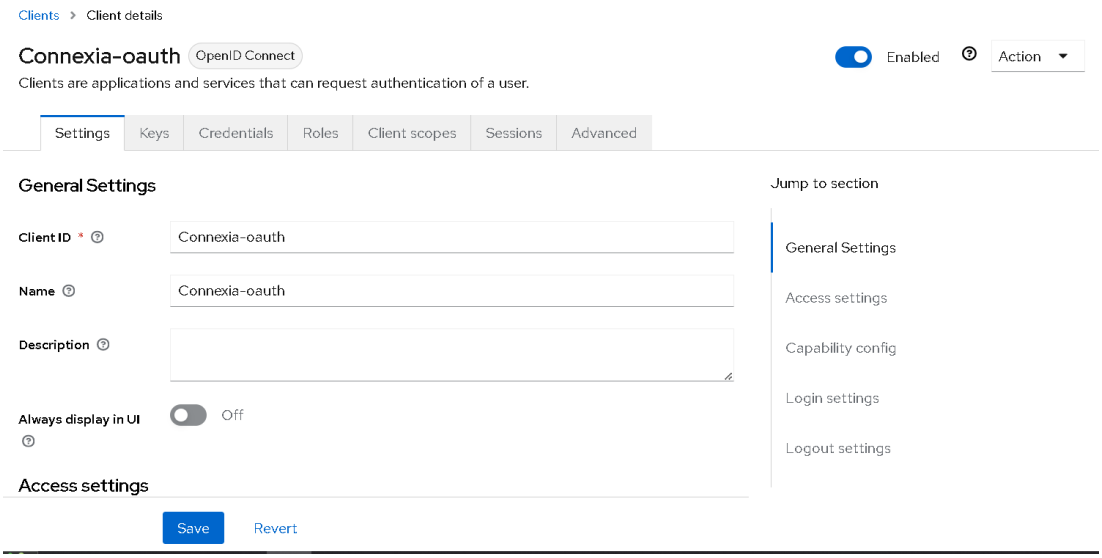


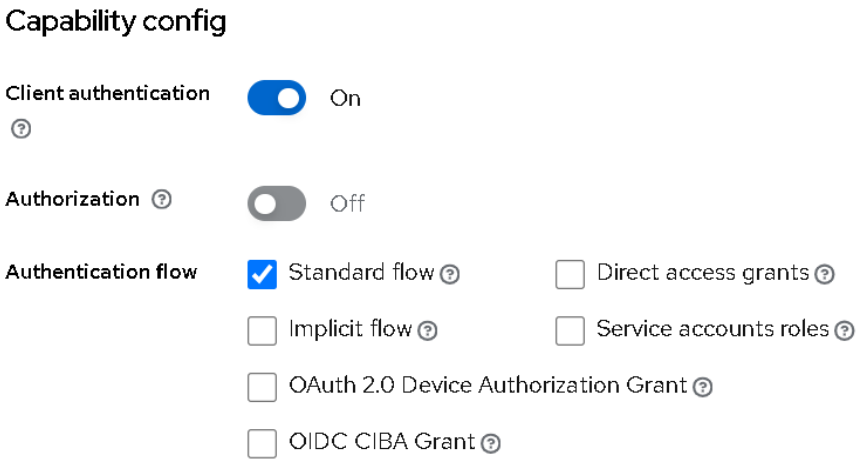


2. Creation d’un nouveau client connexia-ressource

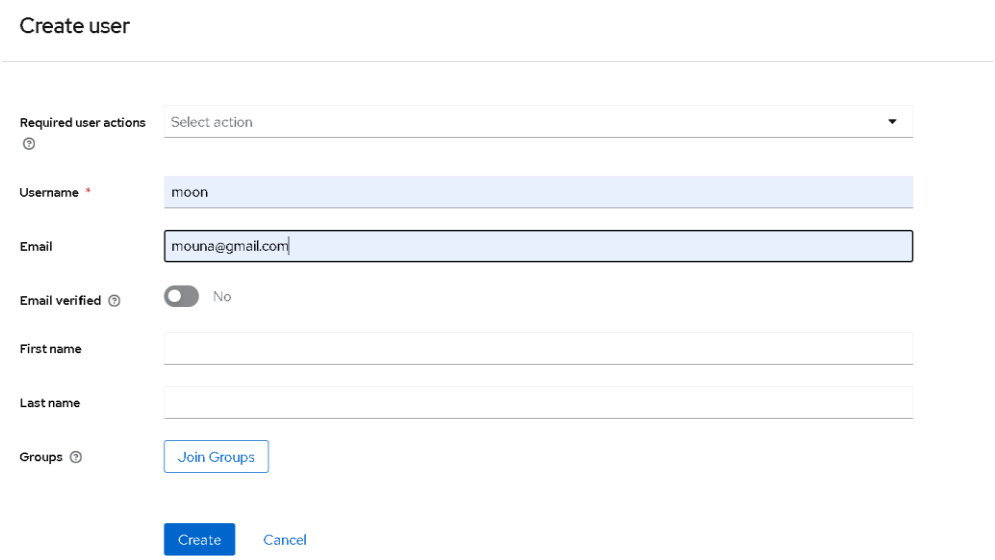


3. On crée un autre client pour oauth

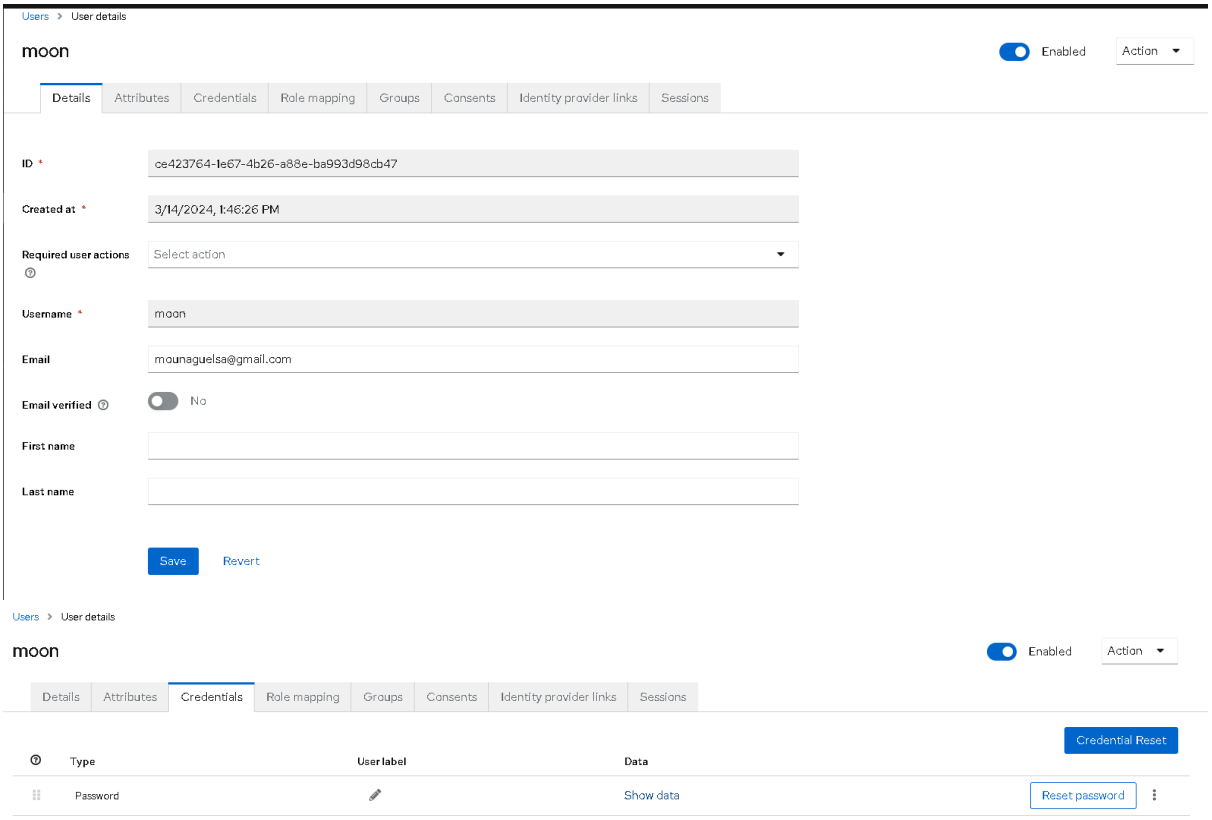




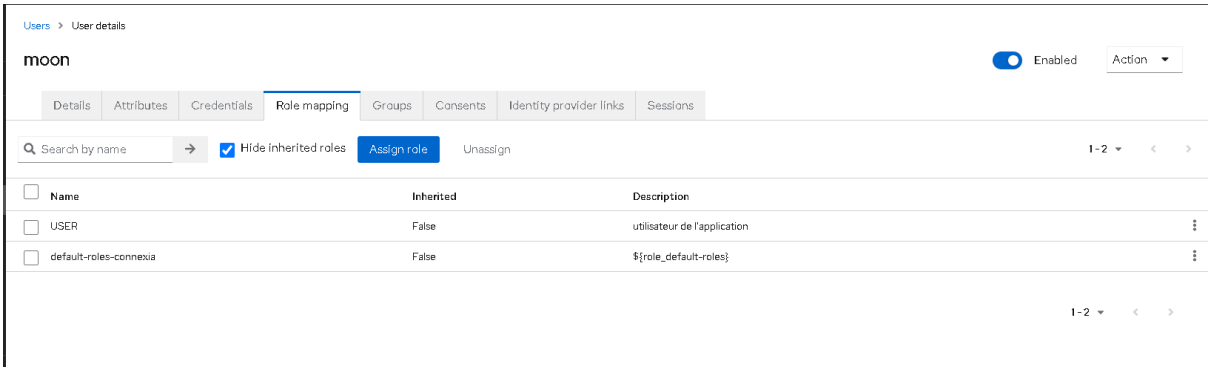
4. On crée un user



5. Après la création on assigne un mot de passe



6. On assigne un role au user



L'implémentation

Cette fonction `convert` est une implémentation de la méthode `convert` de l'interface `Converter` dans Java. Dans ce cas particulier, cette fonction est conçue pour convertir un objet Jwt (JSON Web Token) en une collection de rôles accordés (`GrantedAuthority`)

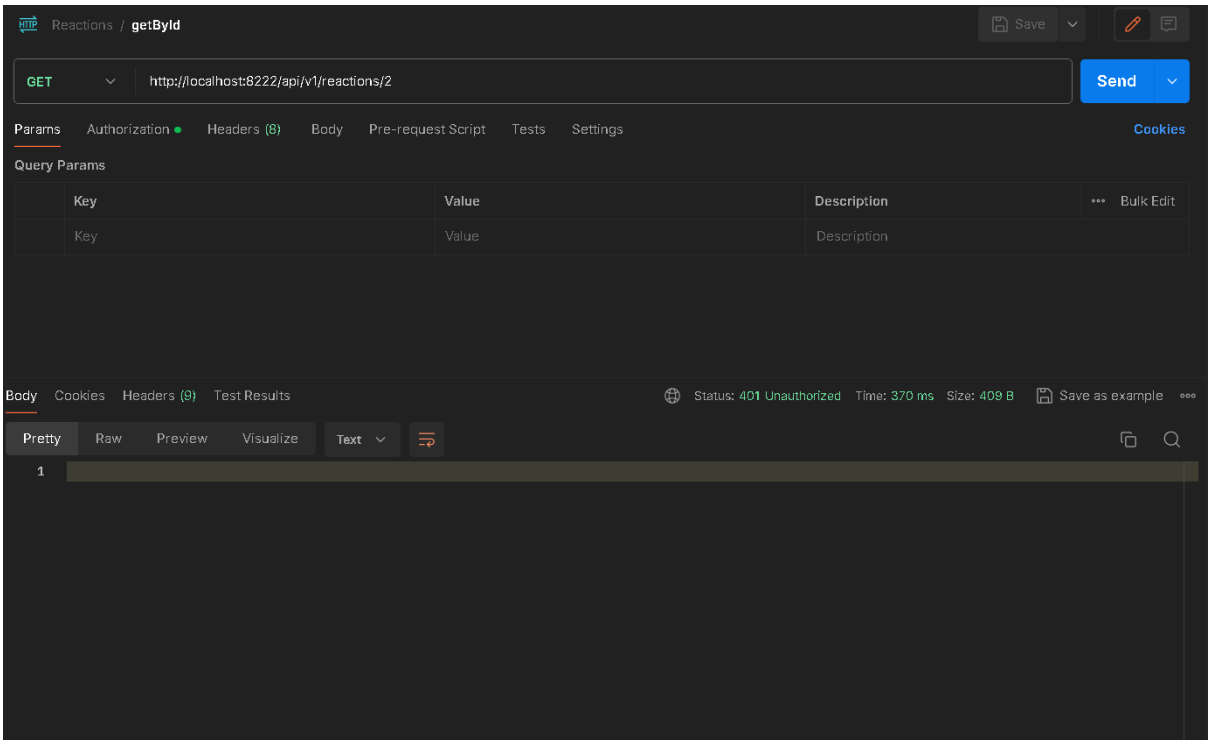
```java
package com.exemple.geteway.security;

import org.springframework.core.convert.converter.Converter;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.oauth2.jwt.Jwt;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

1 usage  Mouna GUELSA
public class KeycloakRoleConverter  implements Converter<Jwt, Collection<GrantedAuthority>> {

    Mouna GUELSA
    @Override
    public Collection<GrantedAuthority> convert(Jwt source) {
        Map<String, Object> realmAccess = (Map<String, Object>) source.getClaims().get("realm_access");
        if (realmAccess == null || realmAccess.isEmpty()) {
            return new ArrayList<>();
        }
        Collection<GrantedAuthority> returnValue = ((List<String>) realmAccess.get("roles"))
                .stream().map(roleName -> "ROLE_" + roleName) Stream<String>
                .map(SimpleGrantedAuthority::new) Stream<SimpleGrantedAuthority>
                .collect(Collectors.toList());
        return returnValue;
}
}
```

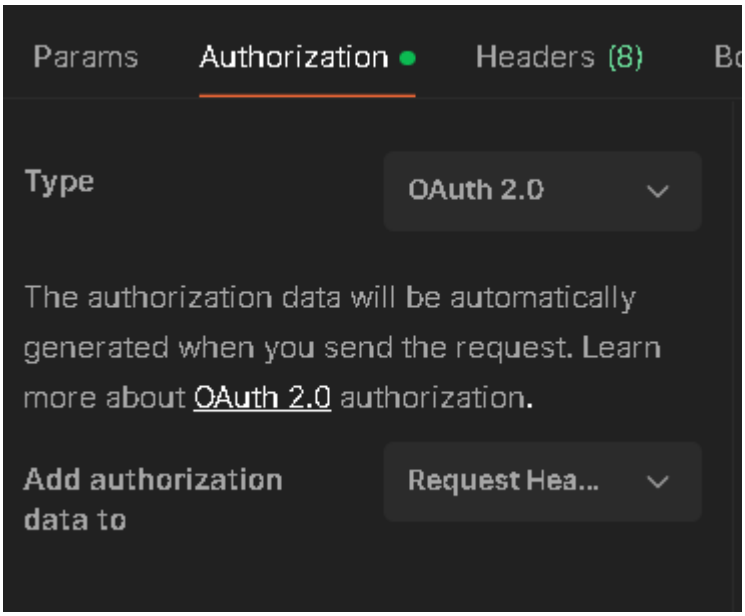
Configuration de sécurité

```java
@Configuration
@EnableWebFluxSecurity
public class SecurityConfig {
    Mouna GUELSA
    @Bean
    public SecurityWebFilterChain springSecurityFilterChain(ServerHttpSecurity serverHttpSecurity) {
        serverHttpSecurity.authorizeExchange(exchanges -> exchanges.anyExchange().hasRole("USER"))
                .oauth2ResourceServer(oAuth2ResourceServerSpec -> oAuth2ResourceServerSpec
                        .jwt(jwtSpec -> jwtSpec.jwtAuthenticationConverter(grantedAuthoritiesExtractor())));
        serverHttpSecurity.csrf(csrfSpec -> csrfSpec.disable());
        return serverHttpSecurity.build();
    }
    1 usage  Mouna GUELSA
    private Converter<Jwt, Mono<AbstractAuthenticationToken>> grantedAuthoritiesExtractor() {
        JwtAuthenticationConverter jwtAuthenticationConverter =
                new JwtAuthenticationConverter();
        jwtAuthenticationConverter.setJwtGrantedAuthoritiesConverter
                (new KeycloakRoleConverter());
        return new ReactiveJwtAuthenticationConverterAdapter(jwtAuthenticationConverter);
}
}
```

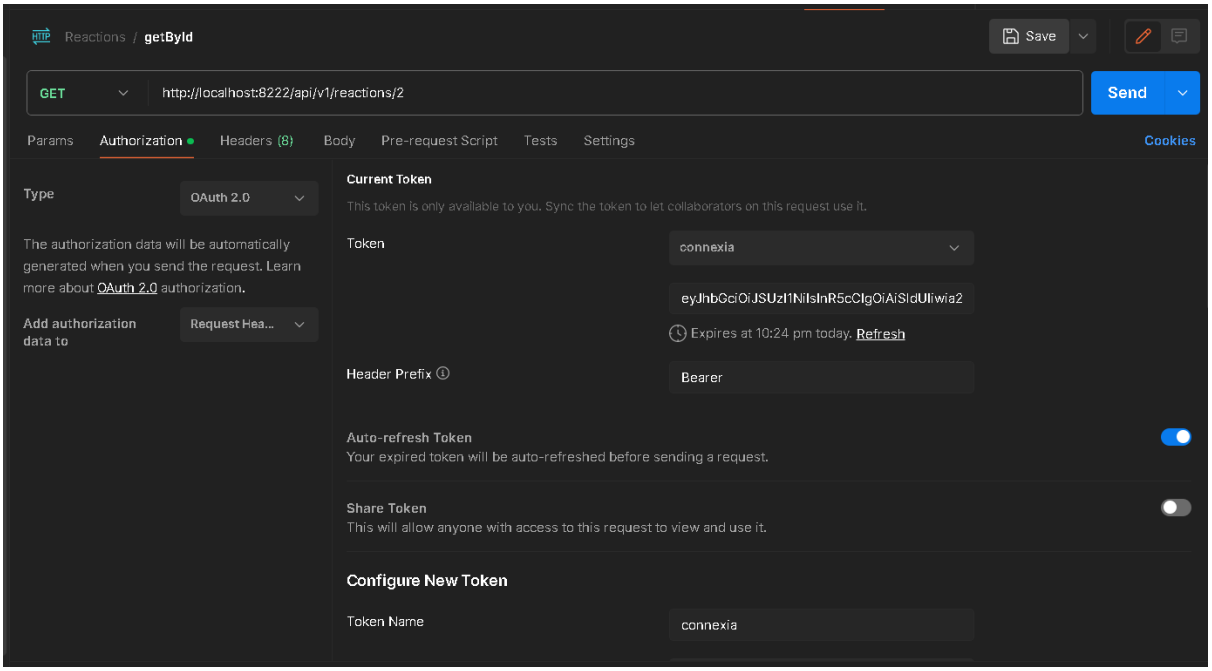
Postman

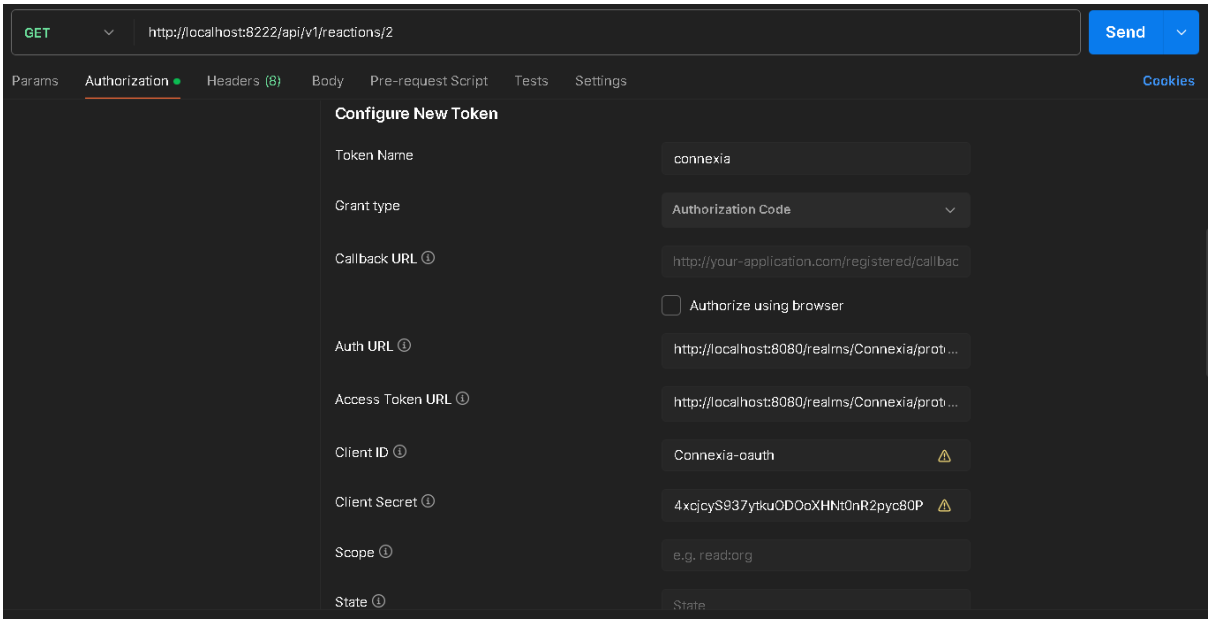
1. Dans la partie Autorisation on choisit : oauth2.0



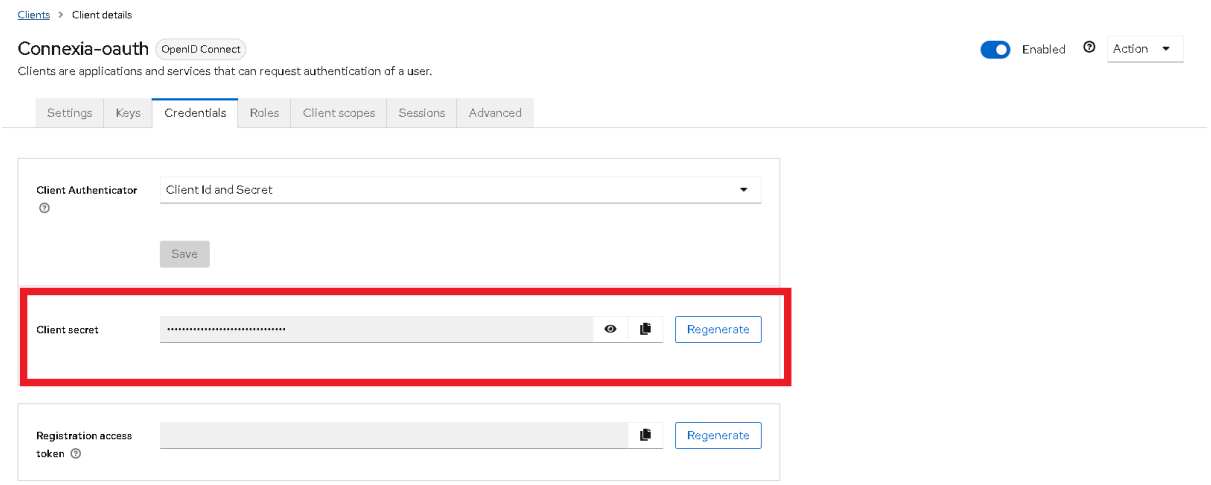


2. On remplit les champs demandes

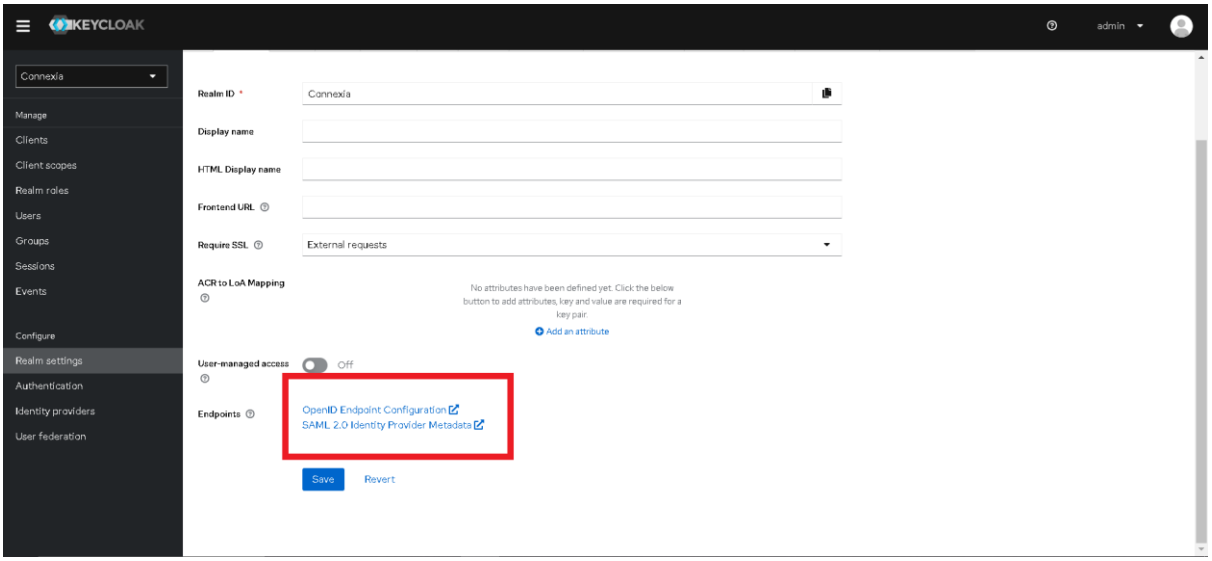




3. Pour l client secret on extraire from keykloak



4. Pour authURL on l'extrait from



Impact sur le projet

L'intégration de ces nouvelles technologies aura un impact significatif sur notre projet. Elle améliorera la scalabilité en permettant à nos microservices de gérer efficacement les charges de travail élevées. La sécurité sera renforcée grâce à l'authentification centralisée et aux mécanismes de contrôle d'accès fournis par Keycloak

Conclusion

En intégrant Apache Kafka et Keycloak dans notre projet de microservices de réseau social, nous visons à améliorer la scalabilité, la sécurité et la réactivité de notre application. Cette intégration nous permettra d'offrir une expérience utilisateur fluide et efficace. Nous sommes confiants que ces nouvelles technologies renforceront notre architecture et amélioreront nos fonctionnalités, conduisant ainsi à un produit final de haute qualité.