esprit Se former autrement HONORIS UNITED UNIVERSITIES	Année Universitaire : 2024-2025 **Applications Web Distribuées**

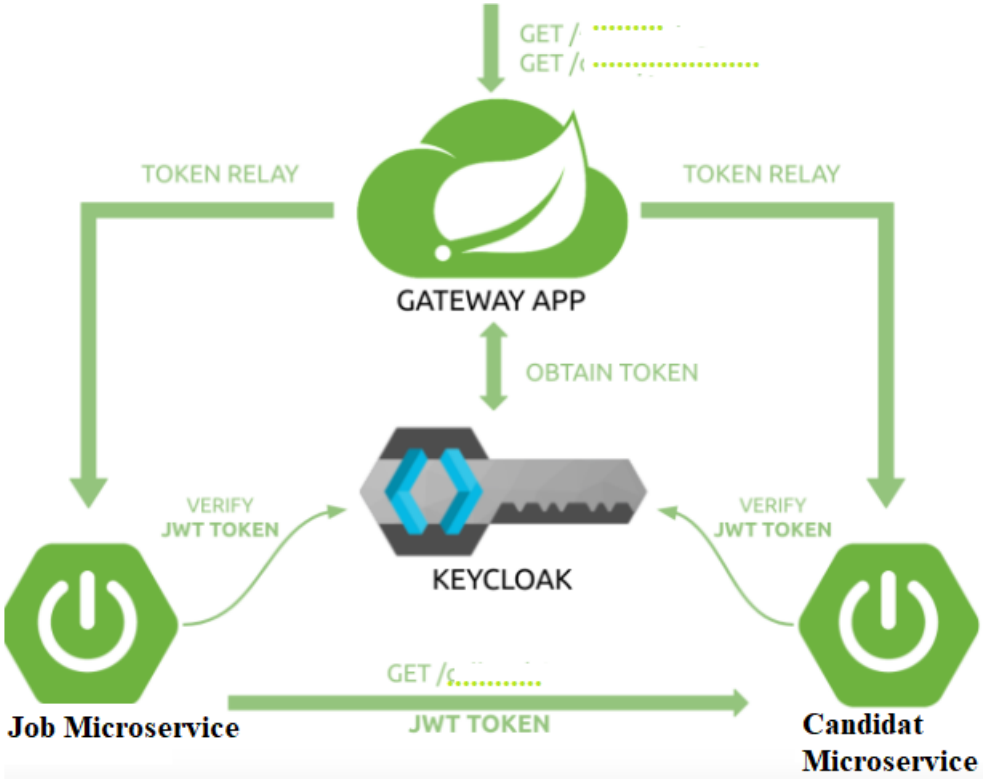
Sécurisation d'une API Gateway avec Keycloak

Objectif

- Assurer la sécurité d'une API Gateway en utilisant Auth2.0.
- Assurer la sécurité de l'accès vers l'ensemble des microservices.

Principe de fonctionnement

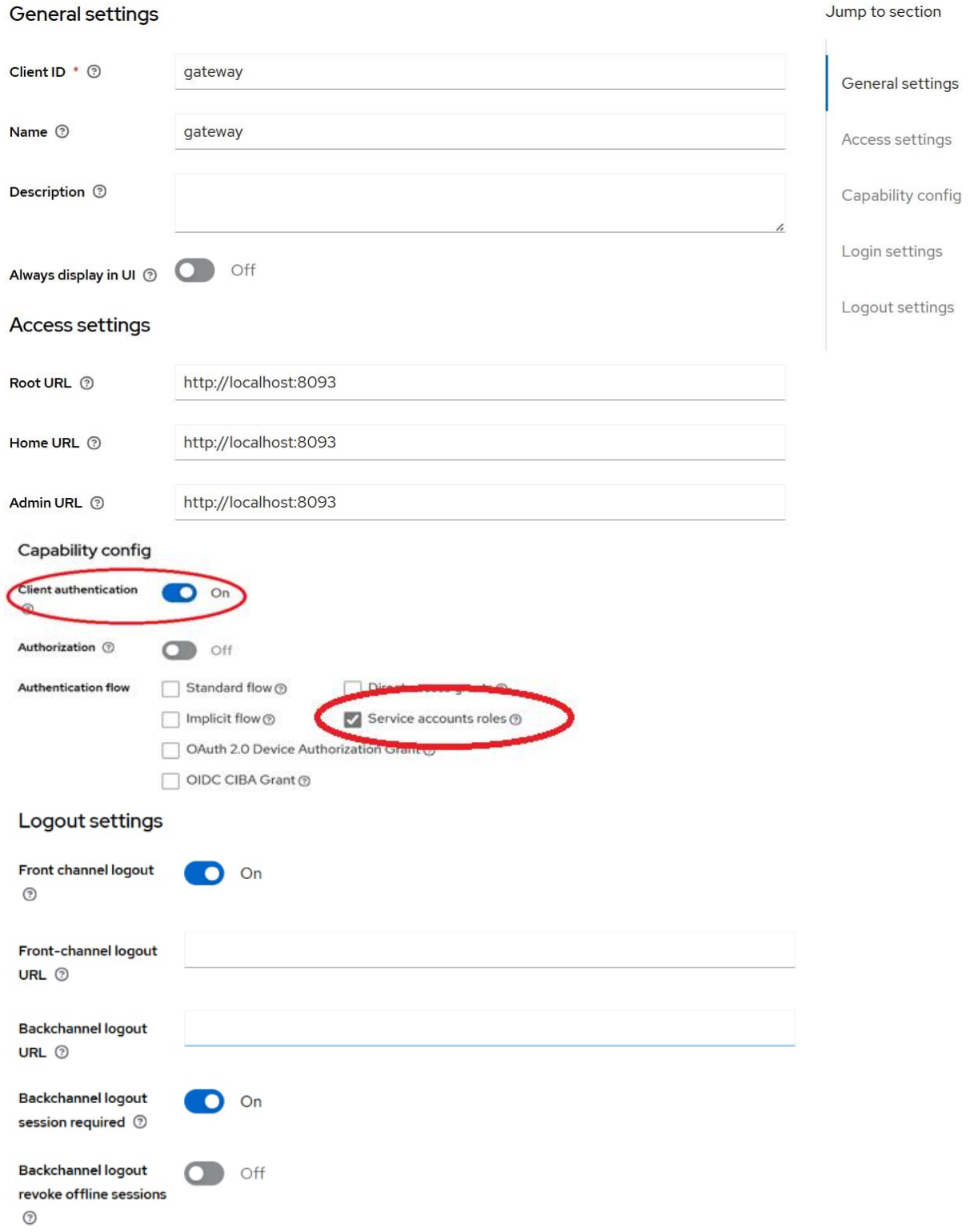
Lors de l'accès aux Microservices à travers le Gateway, celle-ci va renvoyer automatiquement l'utilisateur vers Keycloak pour récupérer un Token. Keycloak authentifiera l'utilisateur si besoin, puis renverra des informations sur l'utilisateur et le fameux Token à notre Viewer. Ce Token sera ensuite utilisé pour l'accès vers le microservice en question.



Les étapes à suivre

1. Créer un realm « **JobBoardKeycloack** »
2. Créer un client **gateway**
   - clientID : gateway (nom de l'application Microservice Gateway)
   - RootURL, HomeURL et AdminURL : http://localhost:8093 (Port de MS Gateway)

- Vérifier la configuration ajoutée selon les figures ci-dessous :



3. Dans le fichier pom.xml de votre projet Gateway, ajouter le code suivant :

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-webflux</artifactId>
    </dependency>
```

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-actuator</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-gateway</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-
client</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-oauth2-resource-
server</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-security</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
    </dependencies>
    <dependencyManagement>
        <dependencies>
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${spring-cloud.version}</version>
                <type>pom</type>
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <repositories>
```

```
            <repository>
               <id>netflix-candidates</id>
               <name>Netflix Candidates</name>
               <url>https://artifactory-
        oss.prod.netflix.net/artifactory/maven-oss-candidates</url>
               <snapshots>
                  <enabled>false</enabled>
               </snapshots>
            </repository>
         </repositories>
```

4. Sous le projet Gateway, ajouter la classe **SecurityConfig**

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.reactive.EnableWebFluxSecu
rity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.config.web.server.ServerHttpSecurity;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.security.web.server.SecurityWebFilterChain;

@Configuration
@EnableWebFluxSecurity
public class SecurityConfig {

  @Bean
    public SecurityWebFilterChain securityWebFilterChain(ServerHttpSecurity
serverHttpSecurity) {
        return serverHttpSecurity.csrf(ServerHttpSecurity.CsrfSpec::disable)
                .authorizeExchange(exchange -> exchange.pathMatchers("/eureka/**")
                        .permitAll()
                        .anyExchange().authenticated()
                ).oauth2ResourceServer((oauth) -> oauth
                        .jwt(Customizer.withDefaults()))
                .build();
    }


}
```

5. Dans le fichier **application.properties**, ajouter la propriété suivante :

```
spring.security.oauth2.resourceserver.jwt.issuer-
uri=http://localhost:8080/realms/JobBoardKeycloack
```

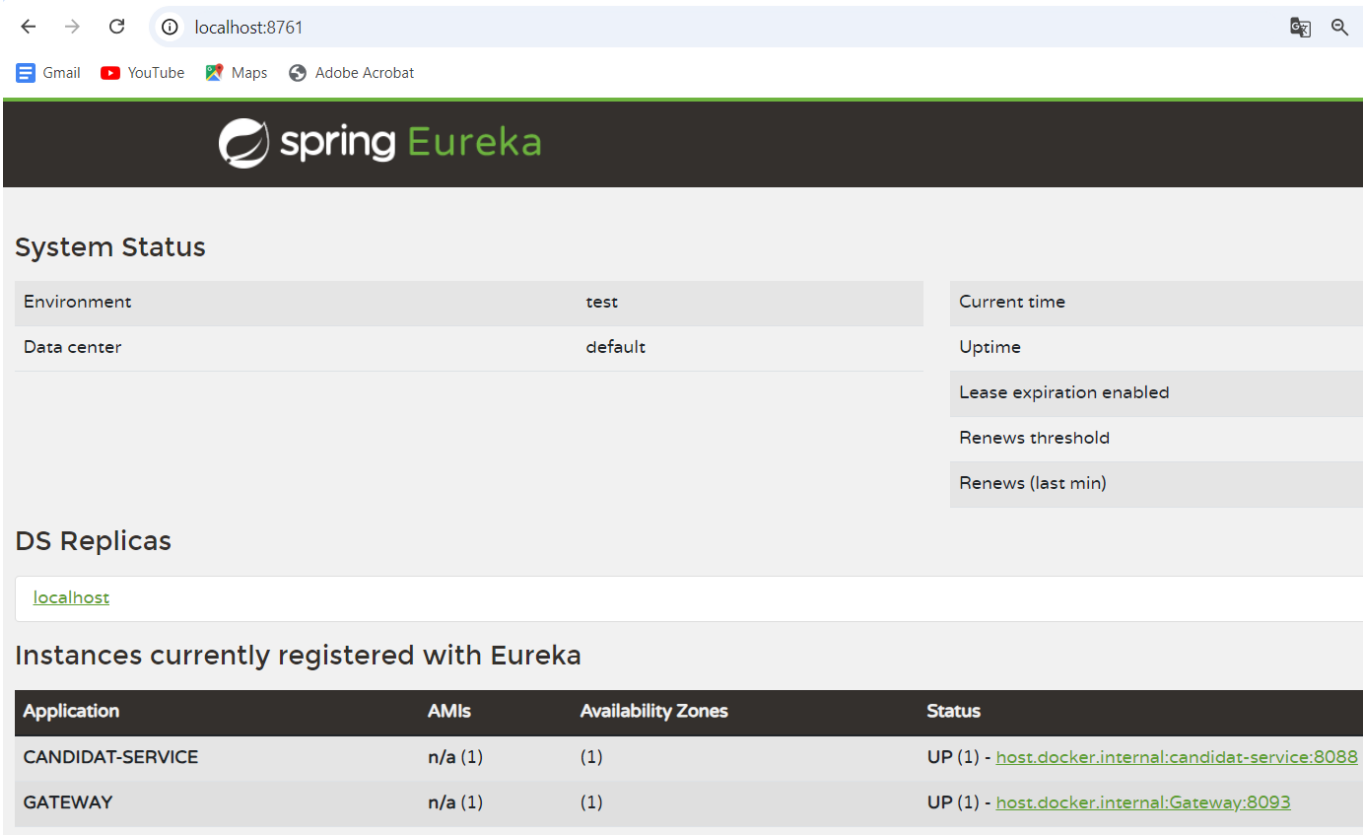
```
spring.application.name=gateway
server.port=8093

# eureka registration
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
eureka.client.register-with-eureka=true

spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/JobBoardKeycloack
```

6. Maintenant, nous allons tester l'accès sécurisé vers le MS de G. Candidats.

Exécuter les projets :

- Gateway
- Candidat
- Eureka

7. Accéder au serveur eureka

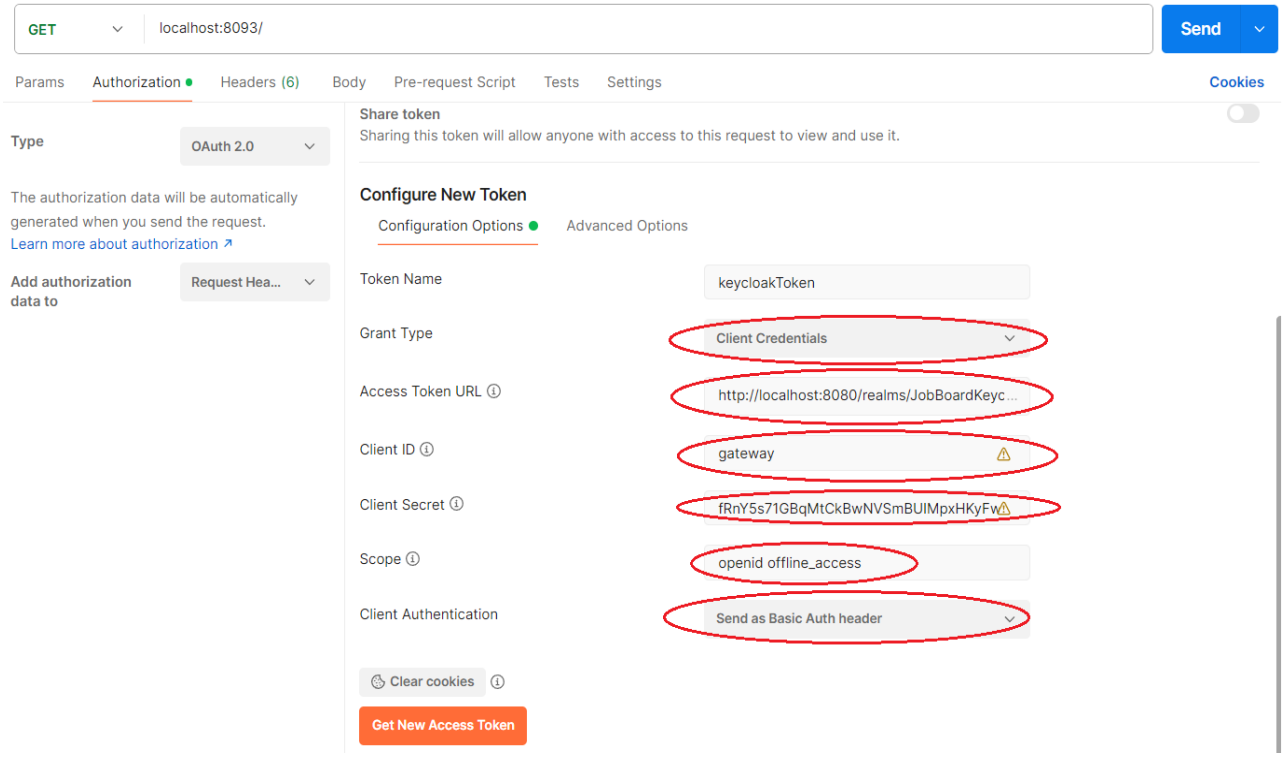


NB :

- 8093 : est le port de l'API Gateway
- Le path d'accès au MS de Candidat est candidats

```
r->r.path("/candidats/**")
```

8. Sur le postman, nous allons générer un token à travers le Keycloak
   - Veuillez choisir Authorization-> OAuth 2.0
   - Veuillez configurer un nouveau token



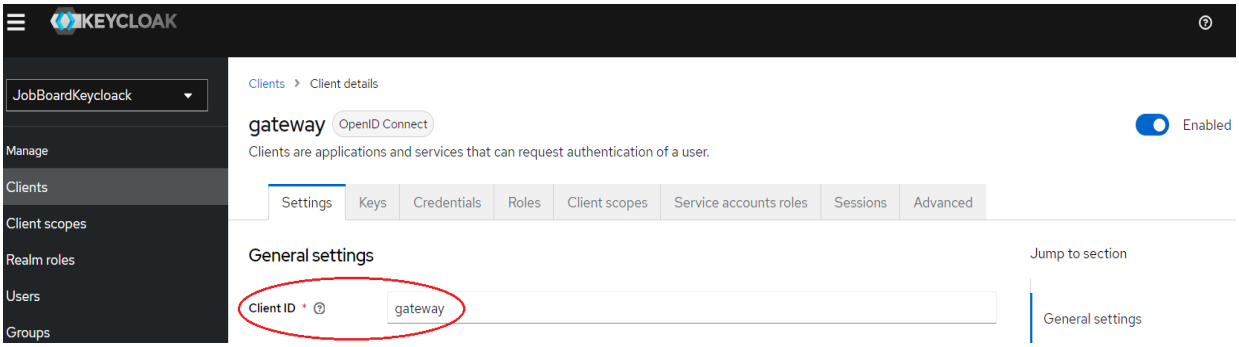
Comme le montre la figure ci-dessus :

- Token Name : veuillez choisir KeycloakToken
- Grant type : Client credentials
- Access Token URL : Pour récupérer cette valeur, au niveau du Realm Settings du keycloak, veuillez cliquer sur le lien OpenID EndPoint Configuration. Puis, récupérer la valeur de token-endpoint

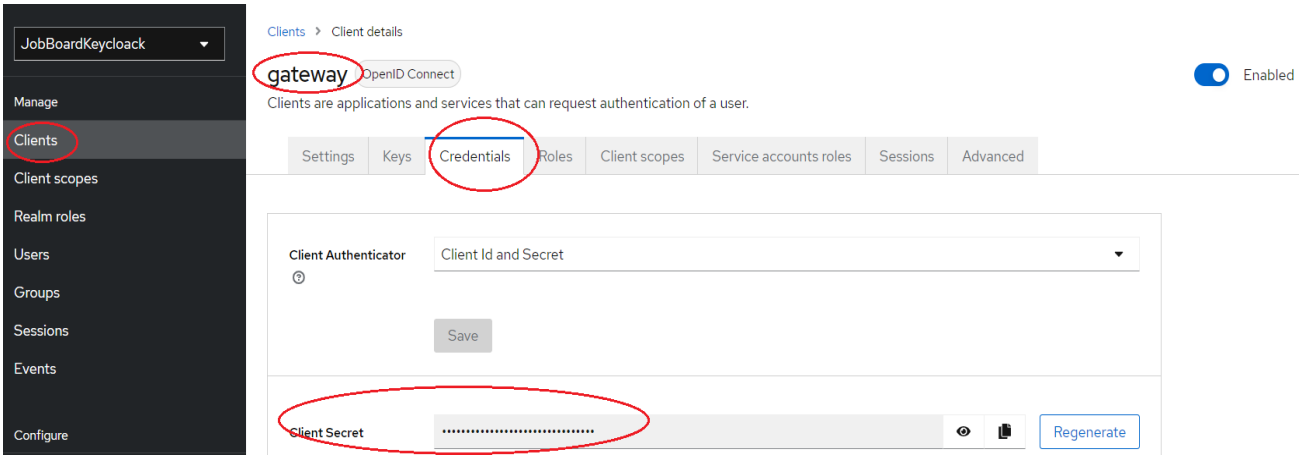
---

{"issuer":"http://localhost:8080/realms/JobBoardKeycloack","authorization_endpoint":"http://localhost:8080/realms/JobBoardKeycloack/protocol/openid-connect/auth","token_endpoint":"http://localhost:8080/realms/JobBoardKeycloack/protocol/openid-connect/token","introspection_endpoint":"http://localhost:8080/realms/JobBoardKeycloack/protocol/openid-

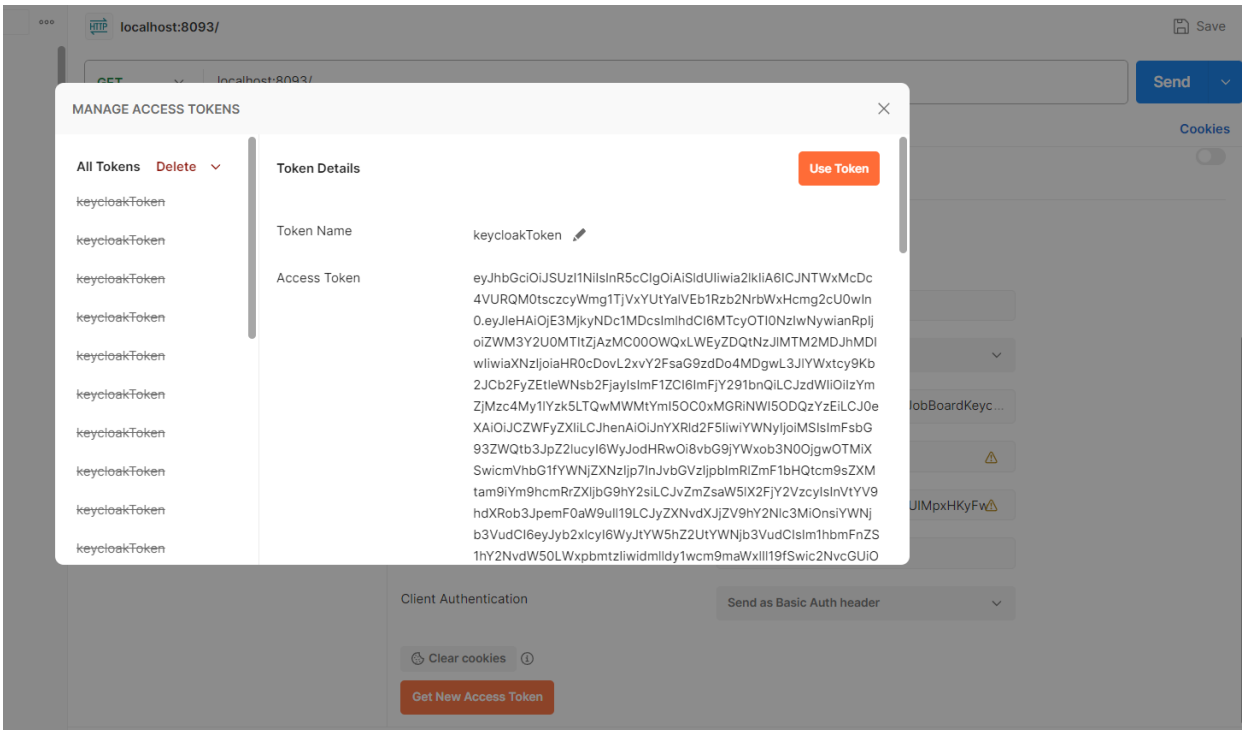
- Client ID : la valeur Client ID spécifié dans Keycloak



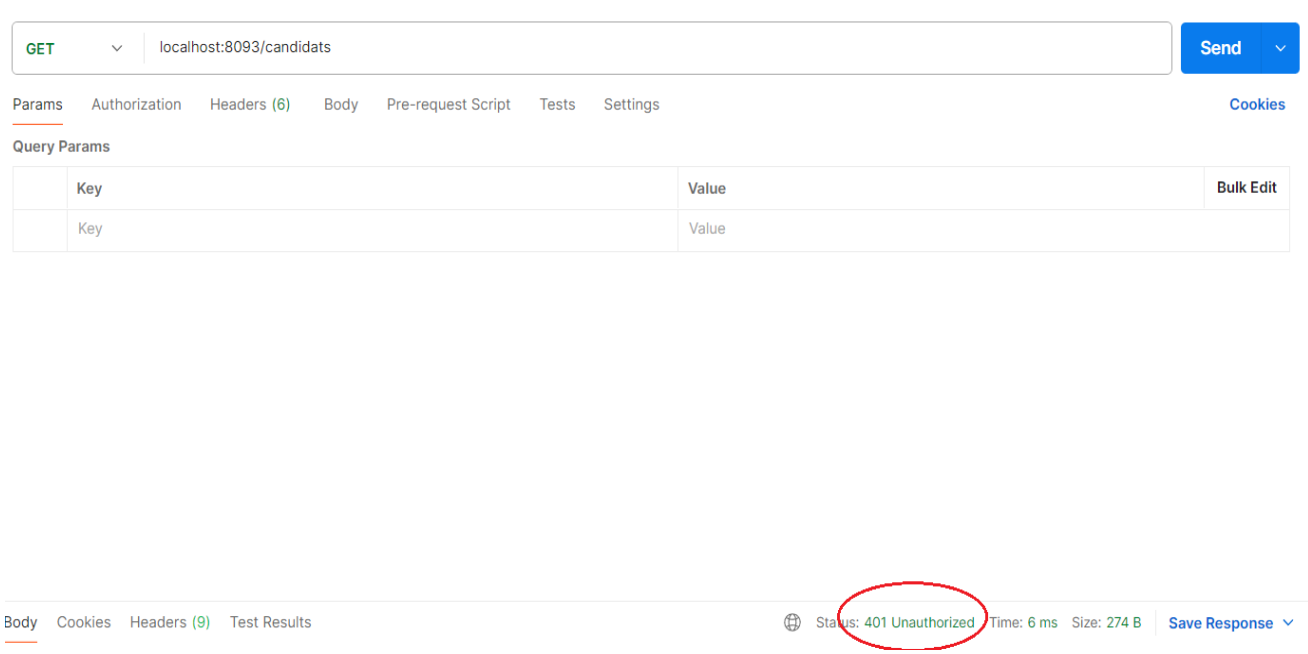
- Client secret :
  Au niveau du Gateway Client, veuillez choisir l'option Credentials. Puis, récupérer la valeur client secret



  - Scope : openid offline_access
  - Client Authentication : Send a Basic Auth header
  - Cliquer sur Get New Access Token. Vous pouvez voir la Valeur de token générée.



  - Veuillez choisir use token pour utiliser cette valeur comme clé de sécurité.
  - Tester l'accès vers le Microservice Candidat comme suit :
    - Sans clé de sécurité



- Avec la clé de sécurité