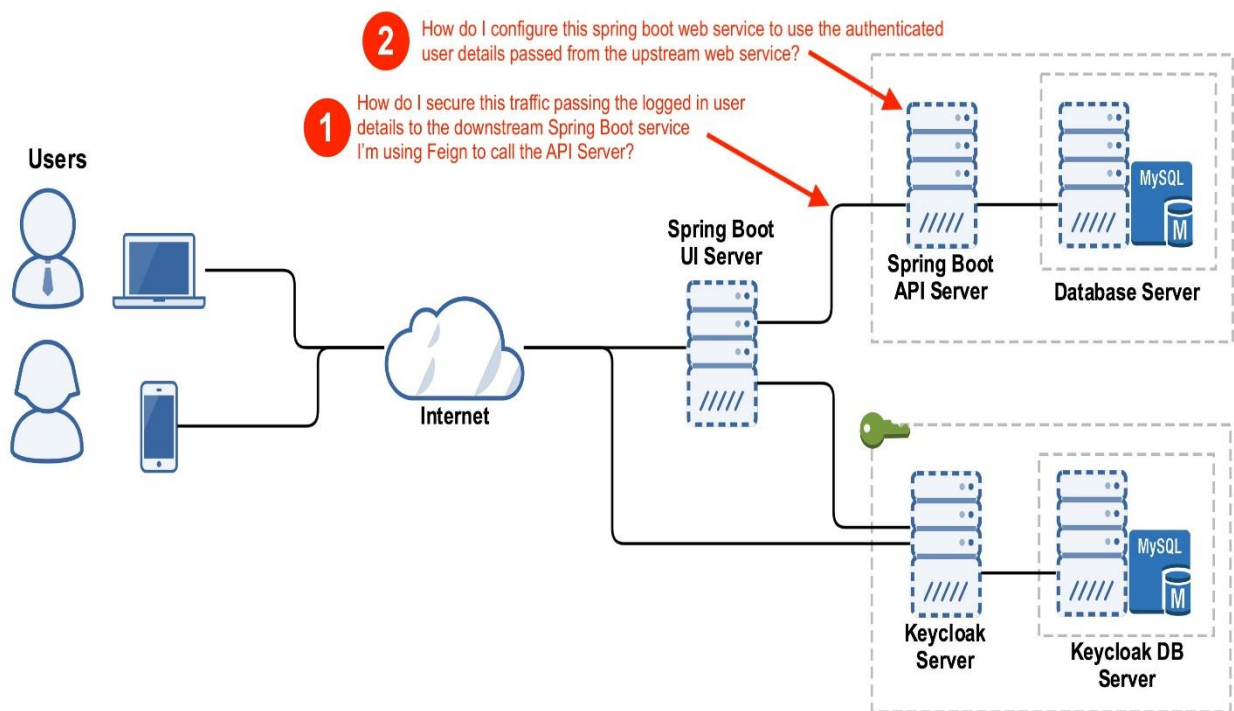


COMPTE-RENDU : ACTIVITE PRATIQUE N° 4 - SECURITE DES MICRO SERVICES AVEC KEYCLOAK

Filière : « Ingénierie Informatique : Big Data et Cloud
Computing » II-BDCC



Réalisé par :

Khadija BENJILALI

Encadré par :

Pr. Mohamed YOUSSEFI

Année Universitaire : 2022-2023

Sommaire

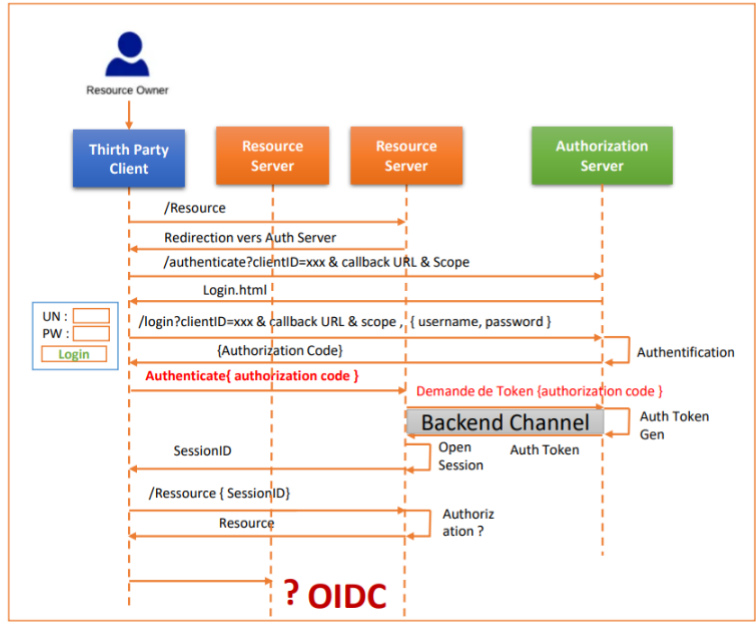
Travail à faire.....	3
PARTIE 1 : Keycloak.....	4
1. Télécharger Keycloak 19.....	4
2. Démarrer Keycloak et Création d'un compte Admin	5
3. Créer une Realm	6
4. Créer un client à sécuriser	6
5. Créer des utilisateurs.....	8
6. Créer des rôles	10
7. Affecter les rôles aux utilisateurs	11
8. Avec PostMan :.....	12
PARTIE 2 : Sécuriser L'architecture Micro services Du projet Customer-service, Inventory-service et Order-service	16
1. Dépendances de sécurité et de Keycloak	17
2. Ajouter les propriétés au fichier de configuration	17
3. Module Security	17
4. Test l'accès à la ressource	18
5. Récupérer tous les Customers	18

Travail à faire

OAuth2

- Protocole et Framework de délégation d'autorisations
- Architecture à trois parties
 - Client
 - Resource server
 - Authorization Server

- **Implicit Flow :**
 - Le serveur d'autorisation envoie directement le Token d'authentification pendant la redirection vers le callback
- **Authorization-code Flow :**
 - Le serveur d'autorisation envoie directement un code d'autorization éphémère pendant la redirection vers le callback (Resource server)
 - Resource server envoie le code dans une requête en back channel au serveur d'autorisation pour demander le token
- **Authorization-code + PKCE Flow :**



Partie 1 :

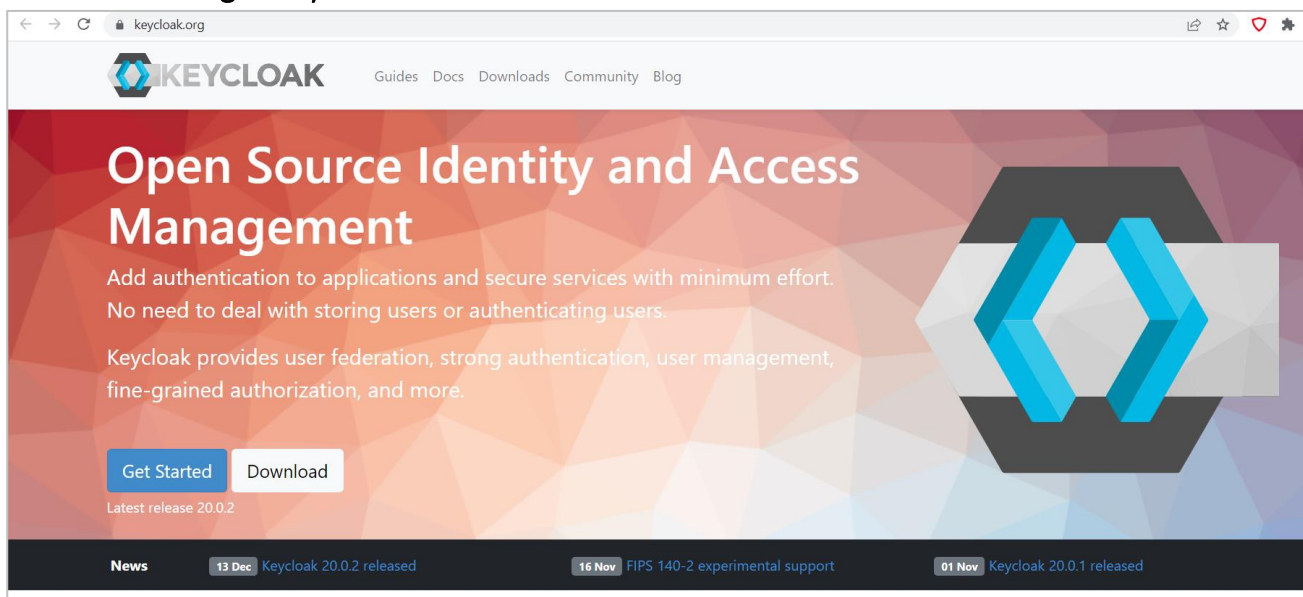
- **Télécharger Keycloak 19**
- **Démarrer Keycloak**
- **Créer un compte Admin**
- **Créer une Realm**
- **Créer un client à sécuriser**
- **Créer des utilisateurs**
- **Créer des rôles**
- **Affecter les rôles aux utilisateurs**
- **Avec PostMan :**
 - **Tester l'authentification avec le mot de passe**
 - **Analyser les contenus des deux JWT Access Token et Refresh Token**
 - **Tester l'authentification avec le Refresh Token**
 - **Tester l'authentification avec Client ID et Client Secret**
 - **Changer les paramètres des Tokens Access Token et Refresh Token**

Partie 2 :

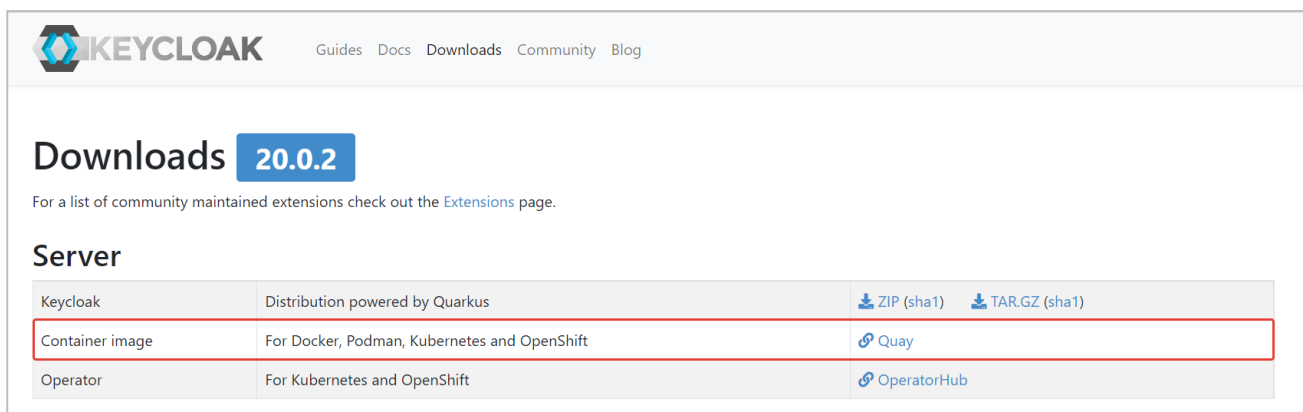
- **Sécuriser L'architecture Micro services Du projet Customer-service, Inventory-service et Order-service**

PARTIE 1 : Keycloak

1. Télécharger Keycloak 19

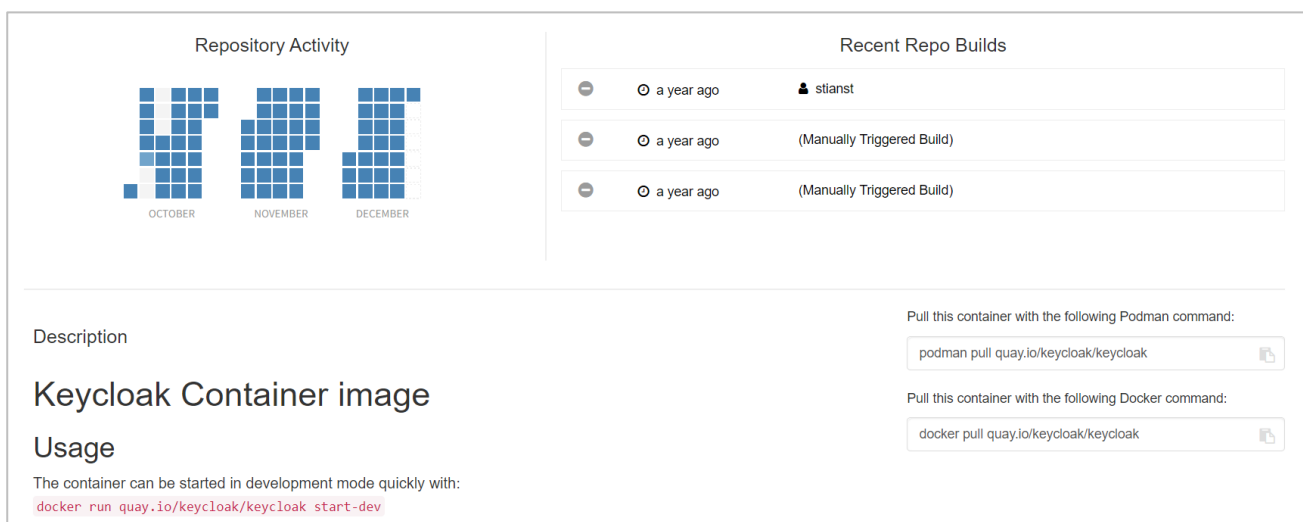


The screenshot shows the Keycloak website homepage. The header includes the Keycloak logo and navigation links: Guides, Docs, Downloads, Community, and Blog. The main content area features the title "Open Source Identity and Access Management" and a description: "Add authentication to applications and secure services with minimum effort. No need to deal with storing users or authenticating users. Keycloak provides user federation, strong authentication, user management, fine-grained authorization, and more." Below this are "Get Started" and "Download" buttons, and a note "Latest release 20.0.2". A footer section contains a "News" tab and three items: "13 Dec Keycloak 20.0.2 released", "16 Nov FIPS 140-2 experimental support", and "01 Nov Keycloak 20.0.1 released".



The screenshot shows the Keycloak Downloads page for version 20.0.2. The header includes the Keycloak logo and navigation links: Guides, Docs, Downloads, Community, and Blog. The main content area features the title "Downloads 20.0.2" and a note: "For a list of community maintained extensions check out the [Extensions](#) page." Below this is a "Server" section with a table:

Keycloak	Distribution powered by Quarkus	ZIP (sha1)	TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	Quay	
Operator	For Kubernetes and OpenShift	OperatorHub	



The screenshot shows the Keycloak Container image page. The header includes the Keycloak logo and navigation links: Guides, Docs, Downloads, Community, and Blog. The main content area features the title "Downloads 20.0.2" and a note: "For a list of community maintained extensions check out the [Extensions](#) page." Below this is a "Server" section with a table:

Keycloak	Distribution powered by Quarkus	ZIP (sha1)	TAR.GZ (sha1)
Container image	For Docker, Podman, Kubernetes and OpenShift	Quay	
Operator	For Kubernetes and OpenShift	OperatorHub	

Repository Activity

Recent Repo Builds

Description

Keycloak Container image

Usage

The container can be started in development mode quickly with:

```
docker run quay.io/keycloak/keycloak start-dev
```

Pull this container with the following Podman command:

```
podman pull quay.io/keycloak/keycloak
```

Pull this container with the following Docker command:

```
docker pull quay.io/keycloak/keycloak
```

➤ Exécuter la commande suivante dans le terminal

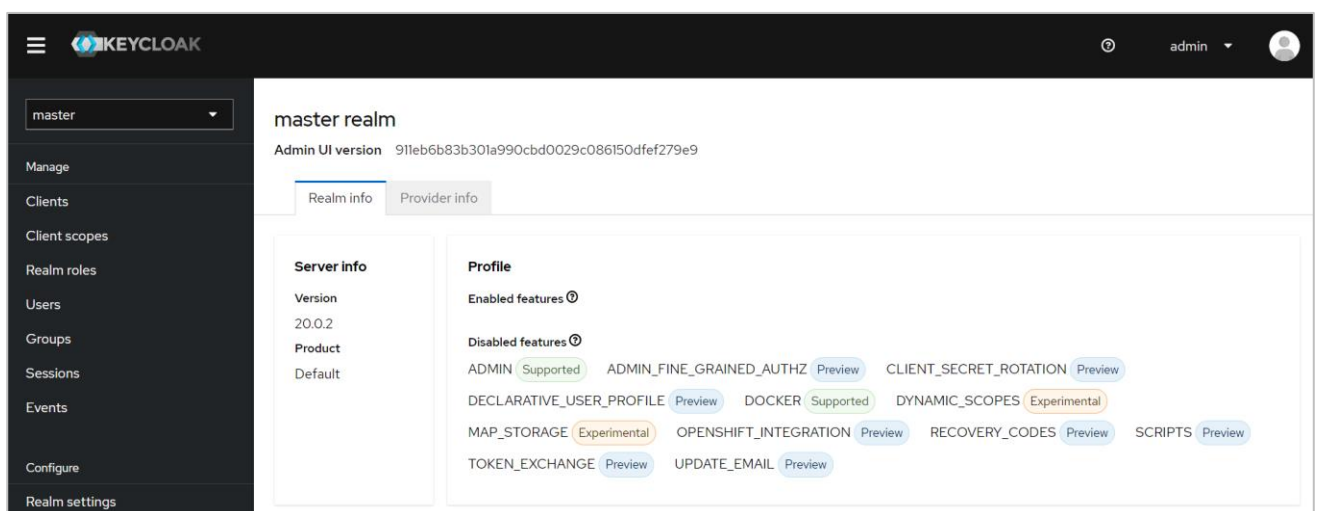
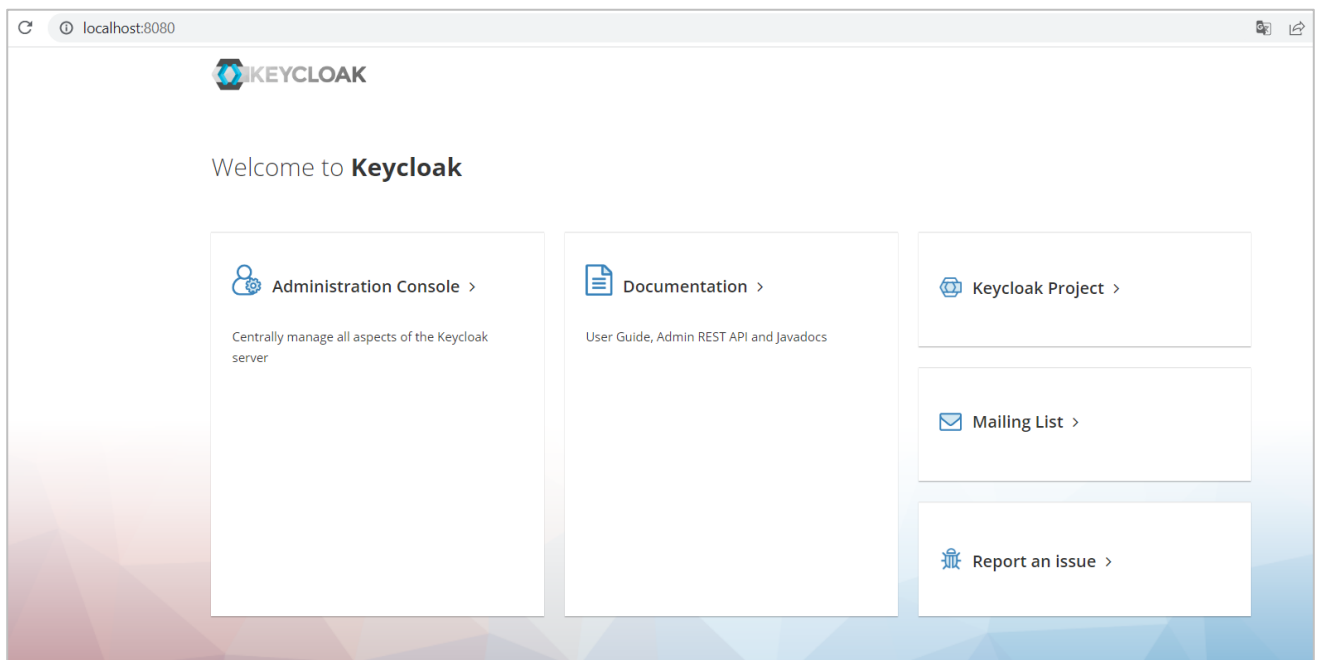
```
docker pull quay.io/keycloak/keycloak
```

```
C:\Users\dell>docker run quay.io/keycloak/keycloak start-dev
Unable to find image 'quay.io/keycloak/keycloak:latest' locally
latest: Pulling from keycloak/keycloak
3e2a8131eeab: Pull complete
d001b67d2428: Pull complete
5dc4e2096006: Pull complete
Digest: sha256:2e6b1012417b725ffe031eb7cc3d89365c1f4d9e9877b222195c7067a1e8810e
Status: Downloaded newer image for quay.io/keycloak/keycloak:latest
Updating the configuration and installing your custom providers, if any. Please wait.
```

2. Démarrer Keycloak et Création d'un compte Admin

➤ Démarrer Keycloak avec l'une des commandes suivantes :

```
docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e
KEYCLOAK_ADMIN_PASSWORD=benjilali2000 quay.io/keycloak/keycloak start-dev
```



3. Créer une Realm

KEYCLOAK

admin

master

Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control.

Resource file

Drag a file here or browse to upload

Browse...

Clear

1

Upload a JSON file

Realm name *

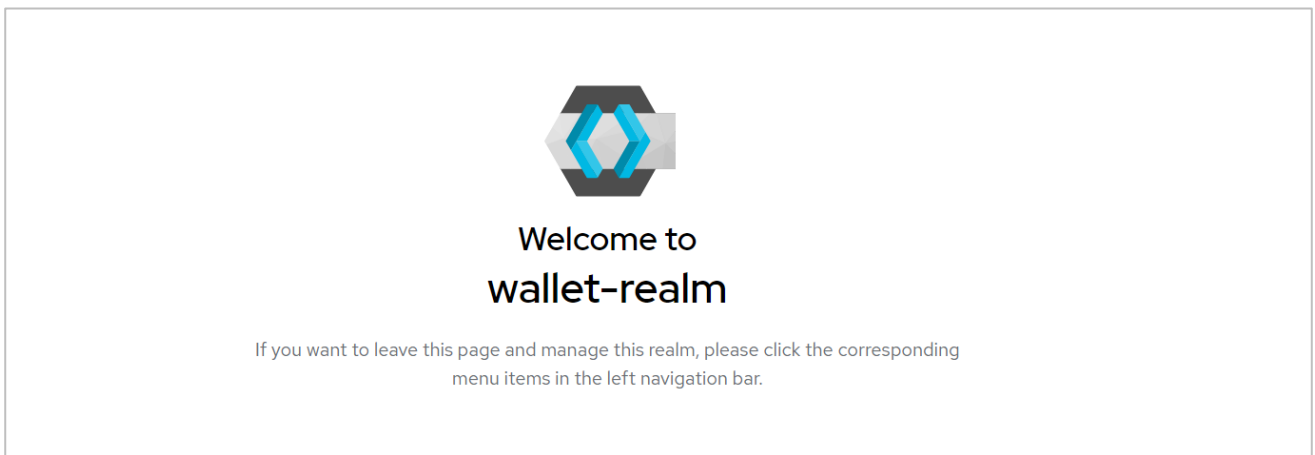
wallet-realm

Enabled

☒ On

Create

Cancel



4. Créer un client à sécuriser

[Clients](#) > Create client

Create client

Clients are applications and services that can request authentication of a user.

1 General Settings

Client type ?

OpenID Connect

Client ID * ?

wallet-client

Name ?

client1

Description ?

Always display in console ?

☐ Off

Next

Back


Cancel

Create client

Clients are applications and services that can request authentication of a user.


1 General Settings


2 **Capability config**


Client authentication  ☐ Off

Authorization  ☐ Off

Authentication flow ☒ Standard flow 

☒ Direct access grants 

☐ Implicit flow 

☐ Service accounts roles 

☐ OAuth 2.0 Device Authorization Grant 


☐ OIDC CIBA Grant 


Save


Back

Cancel

Access settings


Root URL 

Home URL 

Valid redirect URIs 

http://localhost:4200/

[+ Add valid redirect URIs](#)

Valid post logout
redirect URIs 


http://localhost:4200/*

[+ Add valid post logout redirect URIs](#)

Web origins 

http://localhost:4200/

[+ Add web origins](#)

Admin URL 

*

Jump to section

[General Settings](#)

[Access settings](#)

[Capability config](#)

[Login settings](#)


[Logout settings](#)

Clients

Clients are applications and services that can request authentication of a user. [Learn more](#)

Clients list

Initial access token

 Search for client



Create client

Import client

1 - 7

Client ID	Type	Description	Home URL
account	OpenID Connect	—	http://localhost:8080/realms/wallet-realm/account/
account-console	OpenID Connect	—	http://localhost:8080/realms/wallet-realm/account/
admin-cli	OpenID Connect	—	—
broker	OpenID Connect	—	—
realm-management	OpenID Connect	—	—
security-admin-console	OpenID Connect	—	http://localhost:8080/admin/wallet-realm/console/
wallet-client	OpenID Connect	—	http://localhost:4200/

5. Créer des utilisateurs

➤ User1 : Mohamed

[Users](#) > [Create user](#)

Create user

Enabled

Action ▾

Username *

Mohamed

🔒

Email

mohamed@gmail.com

🔒

Email verified ?

Off

First name

mohamed

Last name

BENJILALI

Required user actions ?

Select action

🔒 ▾

Groups ?

Join Groups

Create

Cancel

mohamed

Enabled

Details

Attributes

Credentials

Role mapping

Groups

Consents

Identity provider links

Sessions

ID *

Od69f9db-ce27-4ced-8a5c-f3a57a829c5d

Created at *

12/26/2022, 2:50:24 PM

Username *

mohamed

Email

mohamed@gmail.com

🔒

Email verified ?

Off

First name

mohamed

Last name

BENJILALI

Required user actions ?

Select action

🔒 ▾

mohamed
Details
Attributes
Credentials
Role mapping
Groups
Consents
Identity provider links
Sessions

Set password for mohamed
Password *
Password confirmation *
Temporary ?
Save
Cancel

➤ User2 : Khadija

Users
>
Create user

Create user
Enabled

Username *
Email
Email verified ?
First name
Last name
Required user actions ?
Groups ?

Users

Users are the users in the current realm.
[Learn more](#)

User list

Permissions

Q Search user

→

Add user

Delete user

<input type="checkbox"/>	Username	Email	Last name	First name
<input type="checkbox"/>	khadija	khadija@gmail.com	BENJILALI	khadija
<input type="checkbox"/>	mohamed	mohamed@gmail.com	BENJILALI	Mohamed

6. Créer des rôles

➤ Role1 : USER

[Realm roles](#) > Create role

Create role

Role name *

USER

Description

Save

Cancel

➤ Role2 : ADMIN

[Realm roles](#) > Create role

Create role

Role name *

ADMIN

Description

Save

Cancel

Realm roles

Realm roles are the roles that you define for use in the current realm. [Learn more](#)

Search role by name



Create role

1-7

Role name	Composite	Description
ADMIN	False	-
USER	False	-

7. Affecter les rôles aux utilisateurs

➤ Affecter le User1(Mohamed) au Role1(USER) :

Assign roles to mohamed account

Filter by realm roles

Search by role name

→

1 - 4

<

>

<input type="checkbox"/>	Name	Description
<input type="checkbox"/>	ADMIN	
<input type="checkbox"/>	offline_access	\${role_offline-access}
<input type="checkbox"/>	uma_authorization	\${role_uma_authorization}
<input checked="" type="checkbox"/>	USER	

1 - 4<>

Assign

Cancel

mohamed

DetailsAttributesCredentialsRole mappingGroupsConsentsIdentity provider linksSessions

Search by name

→

☒ Hide inherited roles

Assign role

Unassign

<input type="checkbox"/>	Name	Inherited	Description
<input type="checkbox"/>	USER	False	-
<input type="checkbox"/>	default-roles-wallet-realm	False	\${role_default-roles}

➤ Affecter le User2(Khadija) aux Role1(USER) et Role2(ADMIN) :

Assign roles to mohamed account

Filter by realm roles

Search by role name

→

1 - 4

<

>

<input type="checkbox"/>	Name	Description
<input checked="" type="checkbox"/>	ADMIN	
<input type="checkbox"/>	offline_access	\${role_offline-access}
<input type="checkbox"/>	uma_authorization	\${role_uma_authorization}
<input checked="" type="checkbox"/>	USER	

1 - 4<>

Assign

Cancel

khadija

Details

Attributes

Credentials

Role mapping

Groups

Consents

Identity provider links

Sessions

Q

Search by name

→

☒ Hide inherited roles

Assign role

Unassign

<input type="checkbox"/>	Name	Inherited	Description
<input type="checkbox"/>	USER	False	–
<input type="checkbox"/>	default-roles-wallet-realm	False	\${role_default-roles}
<input type="checkbox"/>	ADMIN	False	–

8. Avec PostMan :

➤ Tester l'authentification avec le mot de passe

http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token

POST

http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Headers

8 hidden

	KEY	VALUE
<input checked="" type="checkbox"/>	Content-Type	application/x-www-form-urlencoded
	Key	Value

http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token

POST

http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	grant_type	password			
<input checked="" type="checkbox"/>	client_id	wallet-client			
<input checked="" type="checkbox"/>	username	mohamed			
<input checked="" type="checkbox"/>	password				
	Key	Value	Description		

➤ Analyser les contenus des deux JWT Access Token et Refresh Token

<pre> 50Ijp7InJvbGVzIjpbIm1ibmFnZS1hY2NvdW50I iwbWFuYWdlLWFfYjY2IbnQtbGlua3MiLCJ2aWV3 LXByb2ZpbGU1XX19LCJzY29wZSI6InBjb2ZpbGU gZW1haWwiLCJzaWQ0OiIzMtGzZjRkZi05M2U2LT Q0ZmUtOWI4MC01MGZlYWY5Y2U2MTgiLCJlbWFPb F92ZXJpZm1lZCI6ZmFsc2UsIm5hbWUiOiJNb2hh bWVkJEFJTkpJTEFMSSIsInByZWZlcnJlZD91c2V ybmFtZSI6Im1vaGFTZWQlLCJnaXZlbn19YW1lIj oiTW9oYW1lZCI6ImZhbnVseV9uYW1lIjo1QkVOS k1MQXxJiIiwZW1haWwiOiJtb2hhbWVhZmVkbWVl LmNvbS9J. SdTeN3oeAk919bhSxNFuZ4ky7eKU92 H6g3LnB2mkZcpJnfPKcMS7oBX7im22xmCgBoziJ hMpXevAK4hqljY8Nm7nrEzuvF8ZD_dm28YAAp3 rKInEF2Sm3bCVcaq8PFmuaIbhl1NJf_oEFJSvLV gpP- 6_aWwvHE1B8tdoplC9zHsvsT10yW1ms370c- LxoxGye21Mh8ssuivyByft_Zhl4dwQIijznyBe0 fz3QgfUqygj2b0vImFOC5cuAZr8S9mb5jqBPmc0 XK64h32Vzy3vPmW2jmFXIRMZnziebKL0Uycr_yY iuzEF7EcrQbENOR4xNzwwkakfsBugyRj87XF8g </pre>	<pre> "realm_access": { "roles": ["offline_access", "default-roles-wallet-realm", "uma_authorization", "USER"] }, "resource_access": { "account": { "roles": ["manage-account", "manage-account-links", "view-profile"] }, "scope": "profile email", "sid": "3183f4df-93e6-44fe-9b80-50feaf9ce618", "email_verified": false, "name": "Mohamed BENJILALI", "preferred_username": "mohamed", "given_name": "Mohamed", "family_name": "BENJILALI", "email": "mohamed@gmail.com" } </pre>
	<div>VERIFY SIGNATURE</div>

```

"realm_access": {
  "roles": [
    "offline_access",
    "default-roles-wallet-realm",
    "uma_authorization",
    "USER"
  ]
},
"resource_access": {
  "account": {
    "roles": [
      "manage-account",
      "manage-account-links",
      "view-profile"
    ]
  }
},
"scope": "profile email",
"sid": "3183f4df-93e6-44fe-9b80-50feaf9ce618",
"email_verified": false,
"name": "Mohamed BENJILALI",
"preferred_username": "mohamed",
"given_name": "Mohamed",
"family_name": "BENJILALI",
"email": "mohamed@gmail.com"
}

```

VERIFY SIGNATURE

XK64h32Vzy3vPmW2jmFXIRMZnziebKL0Uycr_yY
iuzEF7EcrQbENOR4xNzwkkakfsBuggyRj87XF8g

VERIFY SIGNATURE

RSASHA256(
base64UrlEncode(header) + " ." +
base64UrlEncode(payload),

{
"e": "AQAB",
"kty": "RSA",
"n": "zmfQUWBvU6u35uR1tV_1"
},
)

Private Key in PKCS #8, PKCS #
1, or JWK string format. The k
ey never leaves your browser.

SHARE JWT

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  {
    "e": "AQAB",
    "kty": "RSA",
    "n": "zmfQUWBvU6u35uR1tV_i
  },
  Private Key in PKCS #8, PKCS #
  1, or JWK string format. The k
  ey never leaves your browser.
)
```

Signature Verified

SHARE JWT

➤ Tester l'authentification avec le Refresh Token

BodyCookiesHeaders (11)Test Results

PrettyRawPreviewVisualizeJSON

Status: 200 OKTime: 122 msSize: 2.83 KBSave Response

```
1i0i1w9Y0YtW1LzSiCmNDmW5e9V9tW1l1j0i1qKUSKlMQUXJl1w1ZwInawW1Uf3t2dnNDWqKQGTyW1SLnVDSJ9.
2SdTeN3oeAk91bShSn1FuZ4k7eKU92H6g3LnB2mK2cpJnEPKMS7oBx7im22xmCgBoziJhMpexvAK4qJy8Nm7nrcZusvF8ZD_dm28YAap3rKiEfZ5m3bCvCaq8PFmuaIbhL1NJf_eEFJ3vJVGpP_6_awwHE1B8tdnLC9zHsvsT18yWt1
3ms379c-Lxx0gye21Mh8ssuivByft_Zh14dw0I1jznBy8eBf30gUuqy3j2b0vImFOC5cuZa2tS9m56jqBpMcOXk6432Vzy3pVmZjmfXIRM2inebLk0Uyccr_yY1uzEF7EcirQBENOR4kNzwwkakIsBugsRj87XF8g".
4
5"expires_in": 300,
6"refresh_expires_in": 1800,
7"refresh_token": "py3hbgci0i1JuZt1Ni1IsInR5cIc0iA1s1dU1w1a21k1fIA6iCnZ3cNGhZC1kM20ELTR1NjgtYtQXMS82MGfJmZjNDcMjMf0i.
8eYleHAio1E2zn1Wljz20Dm5Im1hdC6MTy3jZjA2N0MylwanRpiIgo1MTBjVju1ZGvYtMGlz2oS89qJfJw1L1ND0Atn0y00k0w0TQ1Zj03t1wXz1j0iaAH8cDovL2xvY2Fsa69zD04MdgW331Ywxtcy93YwxsZQxtcmvhhG0L1CJhdh0L
90i0JdHw0r18vb9g7Yxob3N08jg0w0AvcvnhbG1z3dhb6x1dC1zyWfSbSiSn1N1Y16i1gWtC1NMW4LMWYzEtnDA3y11MTY4LWJ1Y3T3MGzJ20BkNyiIsInR5cI6i1L3n31c2g1LCJhenAi01J3YwxsZQxtY2xpZw50i1w1c2Vzc2Y1
10b19z6F8ZSi6j1JxND0NmNGRMLTkzZTYtNDmRZS95YjgwLTuWzVhZj1jZTYXOCiIsInRj3b1l1j0i1cVnZs81bFpbCIsInR2PCi61jMx00NmNGRMLTkzZTYtNDmRZS95YjgwLTuWzVhZj1jZTYXOCj9.
11P2azMk7Kw2W6ombY1X_h212BMGuSuT82AecnSDA".
12
13"token_type": "Bearer",
14"not-before-policy": 0,
15"session_state": "3183f4df-93e6-44fe-9b80-50feaf9ce618",
16"scope": "profile email"
```

POST

http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> grant_type	refresh_token	
<input checked="" type="checkbox"/> client_id	wallet-client	
<input type="checkbox"/> username	mohamed	
<input type="checkbox"/> password	benjilal2000	
<input checked="" type="checkbox"/> refresh_token	eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldiIiwia2kiOiAiClxNzc3NGNhZC1kM2E	
Key	Value	Description

Body

Cookies

Headers (9)

Test Results

Pretty

Raw

Preview

Visualize

JSON

Status: 200 OK

Time: 17 ms

Size: 2.55 KB

Save Response

```

1  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldiIiwia2kiOiAiClxNzc3NGNhZC1kM2E"
2
3  "expires_in": 360,
4  "refresh_expires_in": 1800,
5  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldiIiwia2kiOiAiClxNzc3NGNhZC1kM2E",
6  "token_type": "bearer",
7  "username": "mohamed"

```

Capability config

Client authentication ⓘ

☒

On

Authorization ⓘ

☐

Off

Authentication flow

☒ Standard flow ⓘ

☒ Direct access grants ⓘ

☐ Implicit flow ⓘ

☒ Service accounts roles ⓘ

☐ OAuth 2.0 Device Authorization Grant ⓘ

☐ OIDC CIBA Grant ⓘ

Jump to section

General Settings

Access settings

Capability config

Login settings

Logout settings

Clients are applications and services that can request authentication of a user.

➤ Changer les paramètres des Tokens Access Token et Refresh Token

PARTIE 2 : Sécuriser L'architecture Micro services Du projet Customer-service, Inventory-service et Order-service

1. Dépendances de sécurité et de Keycloak

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>org.keycloak</groupId>
  <artifactId>keycloak-spring-boot-starter</artifactId>
  <version>19.0.2</version>
</dependency>
```

2. Ajouter les propriétés au fichier de configuration

```
keycloak.realm=wallet-realm
keycloak.resource=wallet-client
keycloak.bearer-only=true
keycloak.auth-server-url=http://localhost:8080
keycloak.ssl-required=none
```

3. Module Security

```
KeycloakAdapterConf.java x
1  package org.sid.customerservice.Security;
2  import ...
5
6  @Configuration
7  public class KeycloakAdapterConf {
8      @Bean
9      KeycloakSpringBootConfigResolver keycloakConfigResolver() {
10         return new KeycloakSpringBootConfigResolver();
11     }
12 }
```

```

KeycloakConfig.java x
1 package org.sid.customerservice.Security;|
2 import ...
13
14 @KeycloakConfiguration
15 @EnableGlobalMethodSecurity(prePostEnabled = true)
16 public class KeycloakConfig extends KeycloakWebSecurityConfigurerAdapter {
17     @Override
18     protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {
19         return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());
20     }
21
22     @Override
23     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
24         auth.authenticationProvider(keycloakAuthenticationProvider());
25     }
26
27     @Override
28     protected void configure(HttpSecurity http) throws Exception {
29         super.configure(http);
30         http.csrf().disable();
31         http.authorizeRequests().anyRequest().permitAll();
32     }
33 }

```

4. Test l'accès à la ressource

```

+ [Icons] Run with: No Environment v
1 POST http://localhost:8080/realms/wallet-realm/protocol/openid-connect/token
2 Accept: application/json
3
4 grant_type=password&username=khadija&password=benjilali2000&client_id=wallet-client
5

```

```

{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50L3N1bWU6IiwiaXNjaW50eXkiOiA6ICJVanRDRlPR2JMQWVVTjA5ODlr",
  "expires_in": 300,
  "refresh_expires_in": 1800,
  "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50L3N1bWU6IiwiaXNjaW50eXkiOiA6ICJVanRDRlPR2JMQWVVTjA5ODlr",
  "token_type": "Bearer",
  "not-before-policy": 0,
  "session_state": "9366bf20-58e4-4369-b879-12d9a145b65a",
  "scope": "profile email"
}
Response file saved.

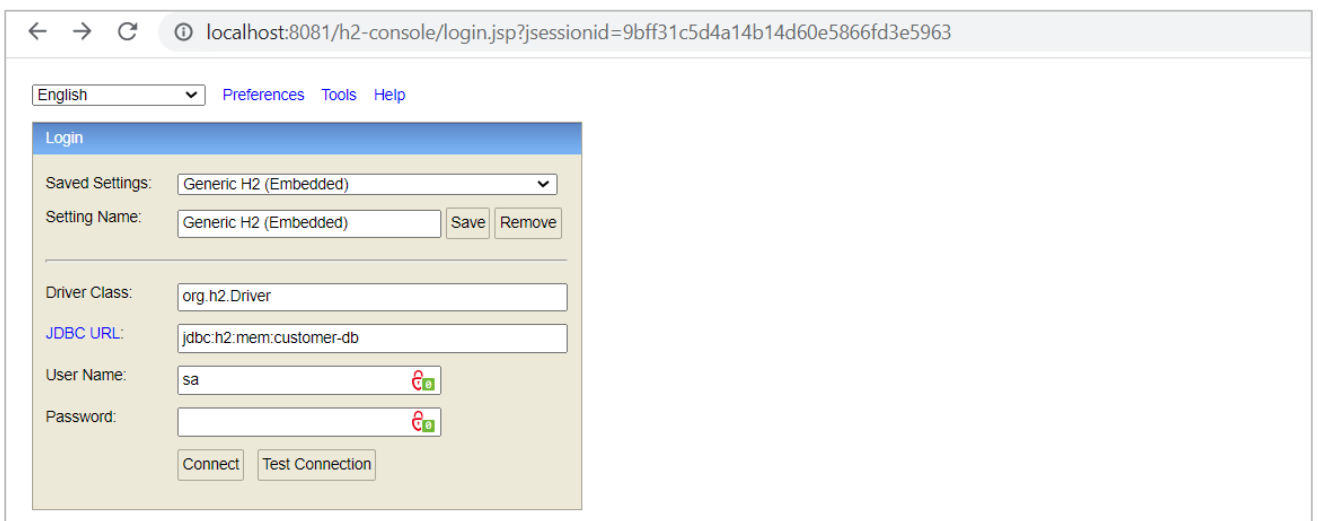
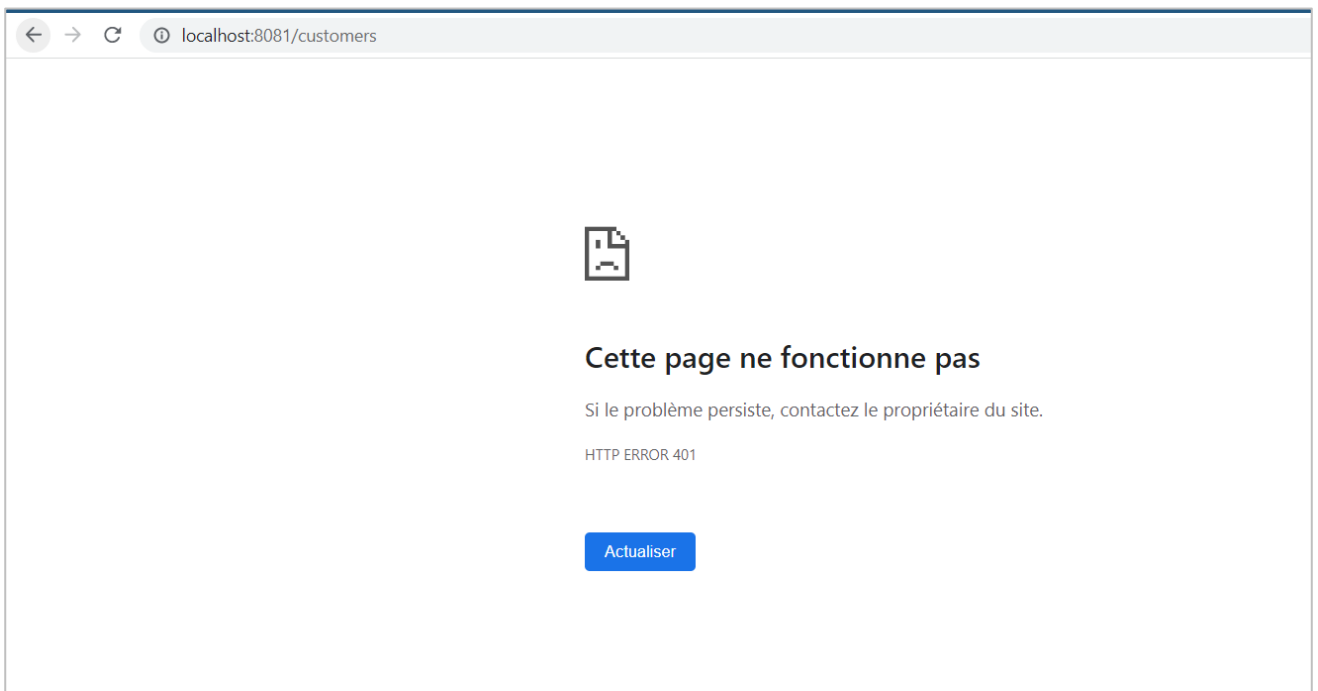
```

5. Récupérer tous les Customers

```

@Override
protected void configure(HttpSecurity http) throws Exception {
    super.configure(http);
    http.csrf().disable();
    //http.authorizeRequests().anyRequest().permitAll();
    http.authorizeRequests().antMatchers(...antPatterns: "/h2-console/**").permitAll();
    http.headers().frameOptions().disable();
    http.authorizeRequests().anyRequest().authenticated();
}

```



SELECT * FROM CUSTOMER;

ID	EMAIL	NAME
1	med@gmail.com	Mohamed
2	hassan@gmail.com	Hassan
3	imane@gmail.com	Imane

(3 rows, 21 ms)