ECOLE NORMALE SUPÉRIEURE DE L'ENSEIGNEMENT
TECHNIQUE DE MOHAMMEDIA
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématique Informatique

Filière : Master-SDIA2

Modèle : Systèmes parallèle et Distribués

Rapport :

Examen Systèmes Distribués

Réalisé par :

- **Abdelkarim AGOUJIL**

Année Universitaire : 2022-2023

1. Introduction

On souhaite créer une application basée sur une architecture micro-service qui permet de gérer des réservations

concernant des ressources. Chaque réservation concerne une seule ressource. Une ressource est définie par son

id, son nom, son type (MATERIEL_INF0, MATERIEL_AUDIO_VUSUEL). Une réservation est définie par son id, son

nom, son contexte, , sa date, sa durée. Chaque réservation est effectuée par une personne. Une personne est

définie par son id, son nom, son email et sa fonction.

L'application doit permettre de gérer les ressources et les réservations. Pour faire plus simple, cette application

se composera de deux micro-services fonctionnels :

• Un Micro-service qui permet de gérer des « Resources-Service ».

• Un Micro-service qui permet de gérer les réservations effectuées par des personnes.

Les micro-services technique à mettre en place sont :

• Le service Gateway basé sur Spring cloud Gateway

• Le service Discovery base sur Eureka Server ou Consul Discovery (au choix)

• Le service de configuration basé sur Spring cloud config ou Consul Config (au choix)

Pour l'application, nous avons besoin de développer une frontend web, basé sur Angular Framework.

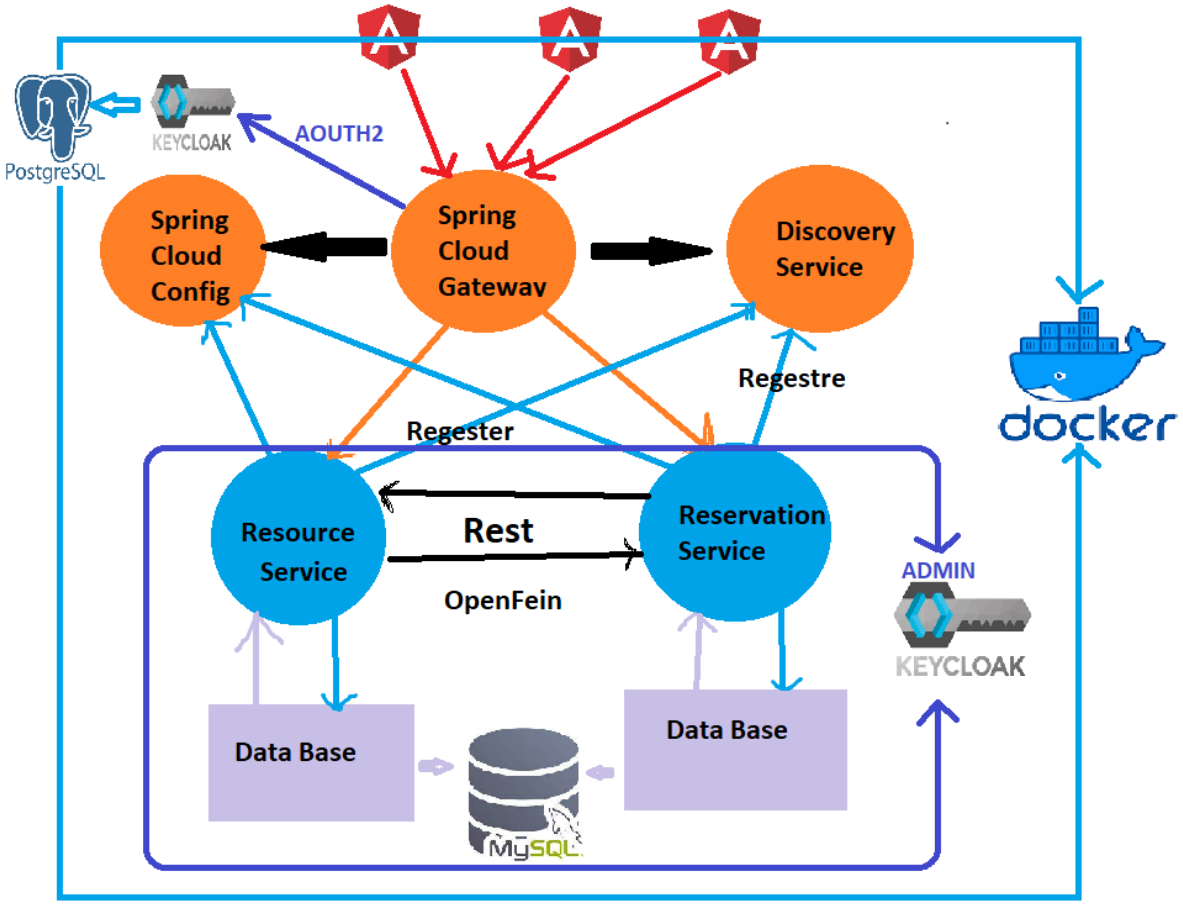
La sécurité de l'application est basée sur Oauth2 et OIDC avec Keycloak comme Provider

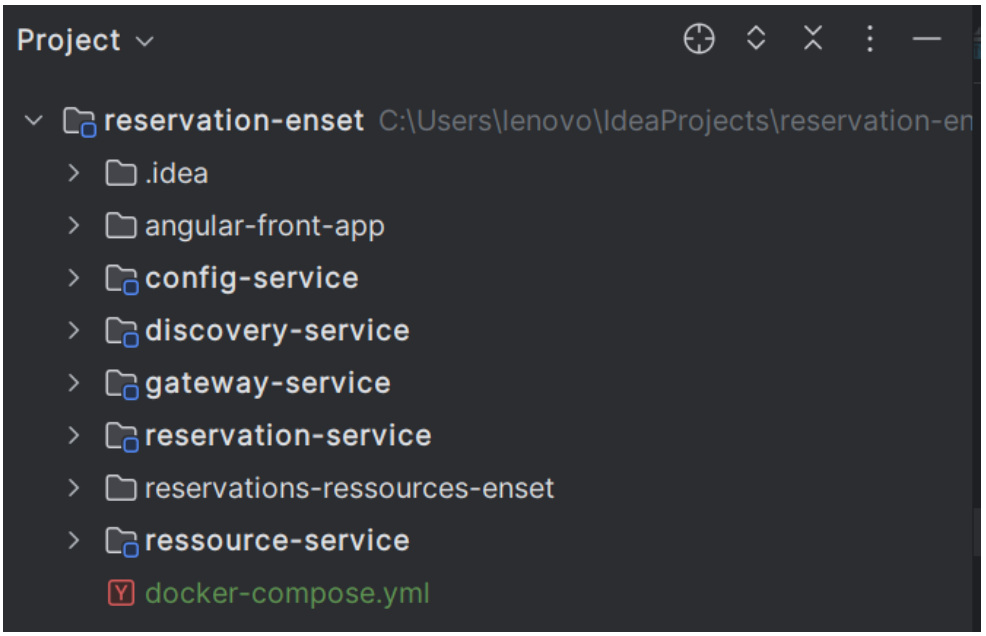
Pour les micro-services, il faut générer la documentation des web services Restfull en utilisant la spécification

OpenAPIDoc (Swagger). Prévoir aussi des circuit breakers basés sur Resilience4J comme solution de fault

tolerence

2. Architecture technique

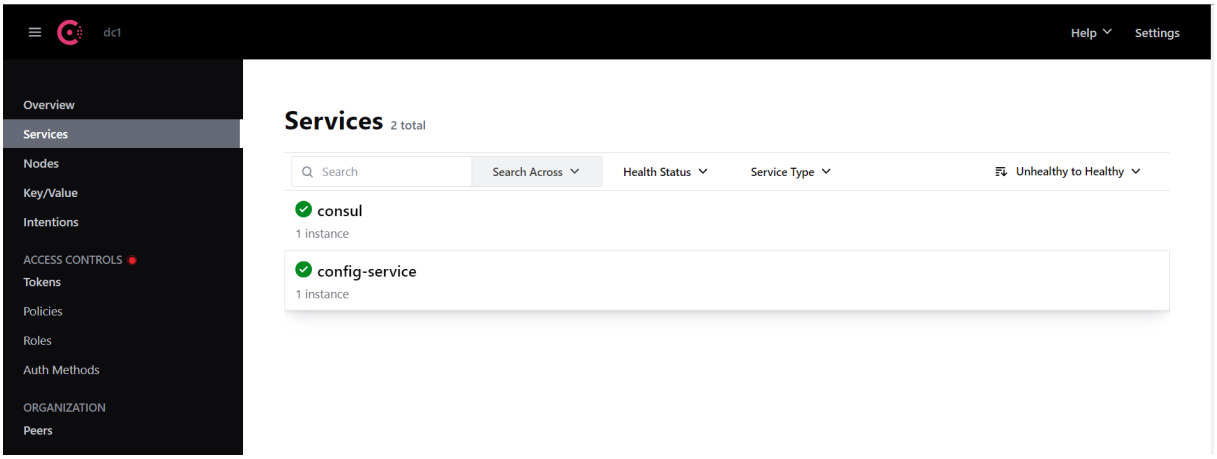


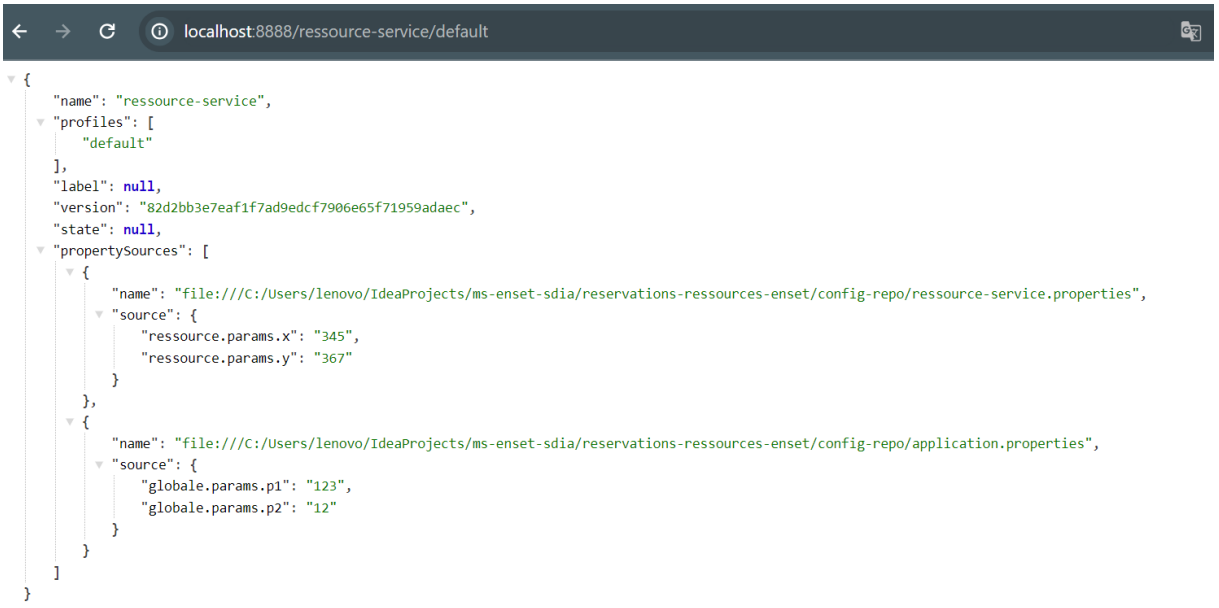


3. Développement du service config :

1) application.properties:

```
server.port=8888
spring.application.name=config-service
spring.cloud.config.server.git.uri=file:///C:/Users/lenovo/IdeaProjects/ms-enset-sdia/reservations-ressources-enset/config-repo
```





4. Développement du micro-service Ressource:

1. application.properties:

```
server.port=8081
spring.application.name=ressource-service
spring.config.import=optional:configserver:http://localhost:8888
```

2. entités:
   A. Resource:

```
package ma.sdia.ressourceservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;
import ma.sdia.ressourceservice.enums.ResourceType;
@Entity
@Getter @Setter @ToString @NoArgsConstructor
@AllArgsConstructor @Builder
public class Ressource {
    @Id
```

```
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;
    private String name;

    @Enumerated(EnumType.STRING)
    private ResourceType type;


}
```

2. enum:
    A. ResourceType:

```
package ma.sdia.ressourceservice.enums;


public enum ResourceType {
    MATERIEL_INFO, MATERIEL_AUDIO_VISUEL
}
```

3. repositories:
    B. ResourceType:

```
package ma.sdia.ressourceservice.repositories;


import ma.sdia.ressourceservice.entities.Ressource;
import
org.springframework.data.jpa.repository.JpaRepository
;
public interface ResourceRepository extends
JpaRepository<Ressource,Long> {
}
```

4. web:
    C. ResourceRestController:

```
package ma.sdia.ressourceservice.web;


import ma.sdia.ressourceservice.entities.Ressource;
import
ma.sdia.ressourceservice.repositories.ResourceReposit
```

```
ory;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;
import
org.springframework.web.bind.annotation.RestControlle
r;


import java.util.List;


@RestController
public class RessourceRestController {
   ResourceRepository resourceRepository;
   public RessourceRestController(ResourceRepository
resourceRepository) {
        this.resourceRepository = resourceRepository;
   }
   @GetMapping("/ressources")
   public List<Ressource> ressourceList(){
        return resourceRepository.findAll();
   }
   @GetMapping("/ressources/{id}")
   public Ressource resourceById(@PathVariable Long
id){
        return resourceRepository.findById(id).get();
   }


}
```

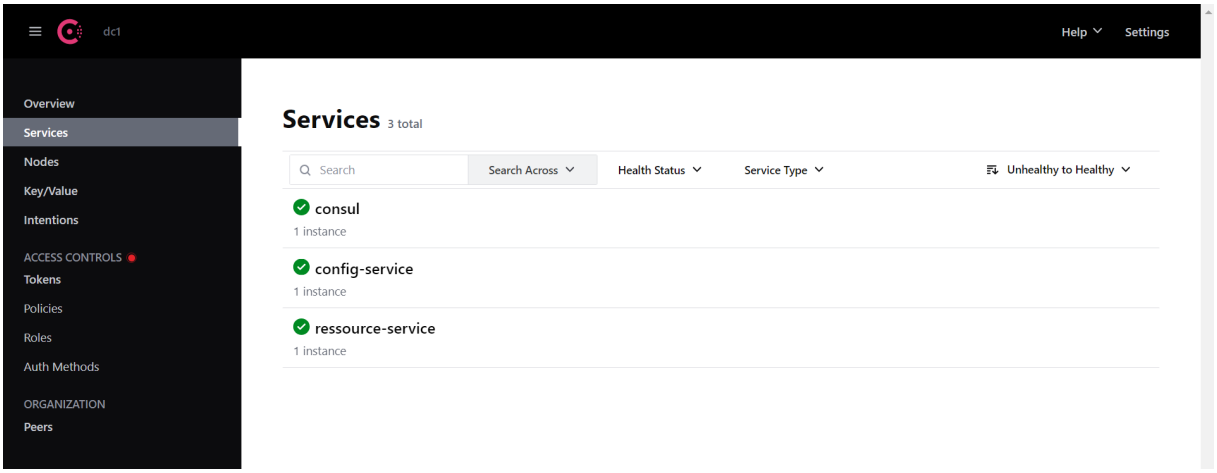
5. application:

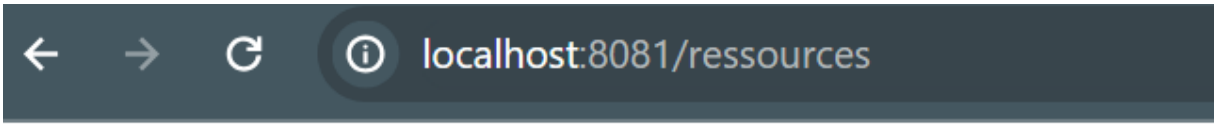
D. RessourceServiceApplication:

```
package ma.sdia.ressourceservice;


import ma.sdia.ressourceservice.entities.Ressource;
```

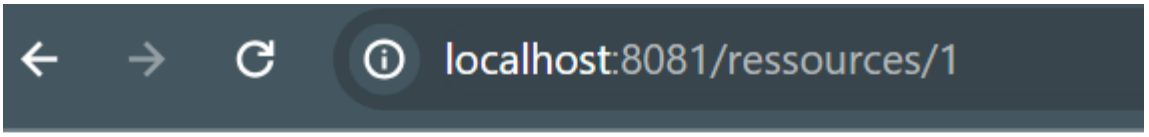
```java
import ma.sdia.ressourceservice.enums.ResourceType;
import
ma.sdia.ressourceservice.repositories.ResourceReposit
ory;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAppl
ication;
import org.springframework.context.annotation.Bean;


import java.util.List;


@SpringBootApplication
public class RessourceServiceApplication {


  public static void main(String[] args) {

SpringApplication.run(RessourceServiceApplication.cla
ss, args);
  }



  @Bean
  CommandLineRunner
commandLineRunner(ResourceRepository
resourceRepository){
     return args -> {


        List<Ressource> customerList=List.of(
              Ressource.builder()
                    .name("Resource1")
                    .type(ResourceType.MATERIEL_INFO)
                    .build(),
```

```
                Ressource.builder()
                        .name("Resource2")
.type(ResourceType.MATERIEL_AUDIO_VISUEL)
                        .build()
        );
        resourceRepository.saveAll(customerList);
    };
  }

}
```



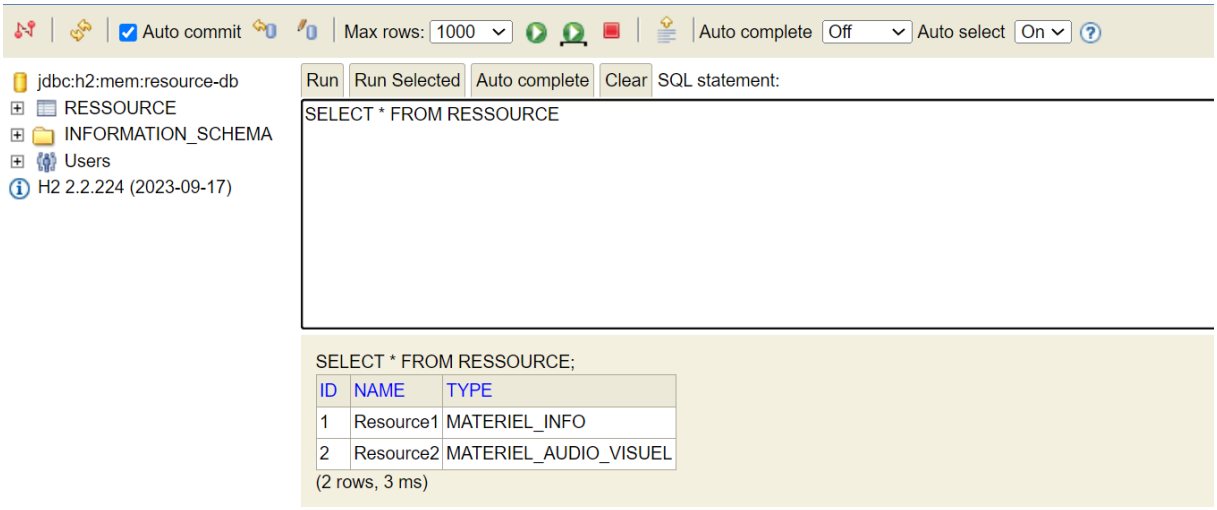



```
[
  {
    "id": 1,
    "name": "Resource1",
    "type": "MATERIEL_INFO"
  },
  {
    "id": 2,
    "name": "Resource2",
    "type": "MATERIEL_AUDIO_VISUEL"
  }
]
```


```
{
  "id": 1,
  "name": "Resource1",
  "type": "MATERIEL_INFO"
}
```


```
management.endPoints.web.exposure.include=*
spring.datasource.url=jdbc:h2:mem:resource-db
spring.h2.console.enabled=true
```



6. Développement du service gateway :

3. Application:

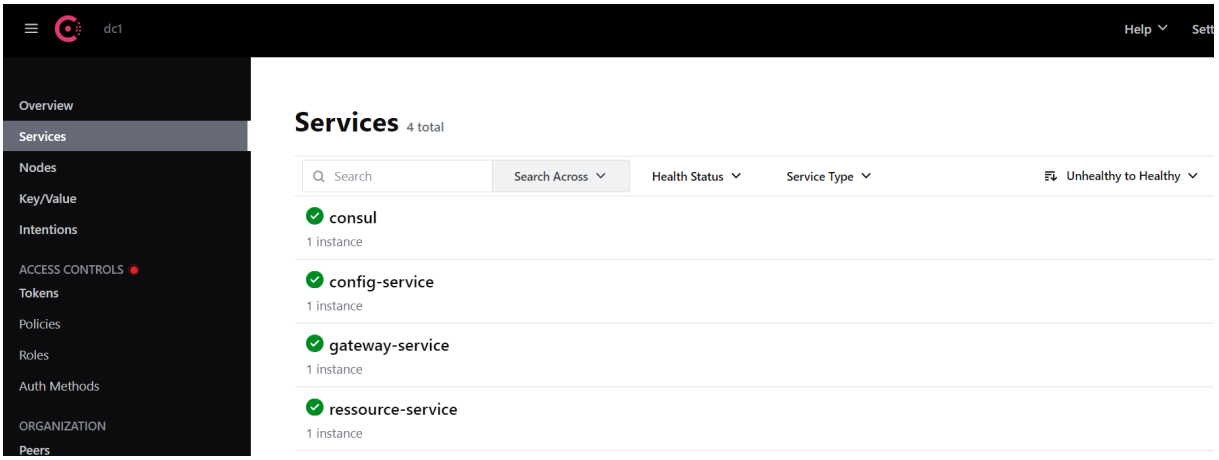
```
package ma.sdia.gatewayservice;


import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAppl
ication;
import
org.springframework.cloud.client.discovery.ReactiveDi
scoveryClient;
```

```
import
org.springframework.cloud.gateway.discovery.Discovery
ClientRouteDefinitionLocator;
import
org.springframework.cloud.gateway.discovery.Discovery
LocatorProperties;
import org.springframework.context.annotation.Bean;


@SpringBootApplication
public class GatewayServiceApplication {


  public static void main(String[] args) {

SpringApplication.run(GatewayServiceApplication.class
, args);
  }


  @Bean
  DiscoveryClientRouteDefinitionLocator
locator(ReactiveDiscoveryClient rdc,
DiscoveryLocatorProperties dlp){
     return new
DiscoveryClientRouteDefinitionLocator(rdc,dlp);
  }




}
```

2) application.properties:

```
server.port=9999
spring.application.name=gateway-service
spring.config.import=optional:configserver:http://loc
alhost:8888
```

4)Test:



7. Développement du micro-service Reservation :

4. Entités JPA et Interface JpaRepository basées sur Spring data

a) Entité Reservation

```
package ma.sdia.reservationservice.entities;

import jakarta.persistence.*;
import lombok.*;
import ma.sdia.reservationservice.model.Ressource;

import java.util.Date;

@Entity
@Getter @Setter @AllArgsConstructor @NoArgsConstructor
@Builder
public class Reservation {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;

    private String context;

    private Date date;
```

```
    private int duration;
    @Transient
    private Ressource ressource;
    private Long ressourceId;
    @Transient
    private Personne personne;
    private String personneId;



}
```

b) Entité Personne

```
package ma.sdia.reservationservice.entities;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;

@Entity
@Getter @Setter @AllArgsConstructor @NoArgsConstructor
@Builder
public class Personne {
    @Id
    private String id;
    private String name;
    private String email;
    private String fonction;
}
```

c) model:

```
package ma.sdia.reservationservice.model;

public enum ResourceType {
    MATERIEL_INFO, MATERIEL_AUDIO_VISUEL
}
```

```
package ma.sdia.reservationservice.model;

import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.*;

@Getter @Setter @ToString @NoArgsConstructor
@AllArgsConstructor @Builder
public class Ressource {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String type;

}
```

d) PersonneRepository:

```
package ma.sdia.reservationservice.repositories;

import ma.sdia.reservationservice.entities.Personne;
import
org.springframework.data.jpa.repository.JpaRepository;
```

```
public interface PersonneRepository extends
JpaRepository<Personne,String> {
}
```

e) ReservationRepositories:

```
package ma.sdia.reservationservice.repositories;

import
ma.sdia.reservationservice.entities.Reservation;
import
org.springframework.data.jpa.repository.JpaRepository;

public interface ReservationRepository extends
JpaRepository<Reservation,Long> {
}
```

f) RessourceRestClient

```
package ma.sdia.reservationservice.web;

import ma.sdia.reservationservice.model.Ressource;
import
org.springframework.cloud.openfeign.FeignClient;
import
org.springframework.web.bind.annotation.GetMapping;
import
org.springframework.web.bind.annotation.PathVariable;

import java.util.List;

@FeignClient("ressource-service")
public interface ResourceRestClient {
   @GetMapping("/ressources")
   public List<Ressource> allRessource();
   @GetMapping("/ressources/{id}")
```

```
    public Ressource findRessourceById(@PathVariable
Long id);


}
```

g) ReservatinRestController

```
package ma.sdia.reservationservice.web;


import ma.sdia.reservationservice.entities.Personne;
import
ma.sdia.reservationservice.entities.Reservation;
import ma.sdia.reservationservice.model.Ressource;
import
ma.sdia.reservationservice.repositories.PersonneReposi
tory;
import
ma.sdia.reservationservice.repositories.ReservationRep
ository;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;


import java.util.Date;
import java.util.List;
import java.util.Random;


@RestController
public class ReservationRestController {
    private ReservationRepository
reservationRepository;
    private ResourceRestClient resourceRestClient;
    private PersonneRepository personneRepository;
    public
ReservationRestController(ReservationRepository
```

```
reservationRepository, ResourceRestClient
resourceRestClient, PersonneRepository
personneRepository) {
        this.reservationRepository =
reservationRepository;
        this.resourceRestClient = resourceRestClient;
        this.personneRepository = personneRepository;
    }

    @GetMapping("/reservations")
    public List<Reservation> reservationList(){
        List<Reservation> reservationList =
reservationRepository.findAll();
        reservationList.forEach(rserv->{
            rserv.setPersonne(
personneRepository.findById(rserv.getPersonneId()).get
());

rserv.setRessource(resourceRestClient.findRessourceByI
d(rserv.getRessourceId()));
        });
        return reservationList;
    }
    @GetMapping("/reservation/{id}")
    public Reservation reservationById(@PathVariable
Long id){
        Reservation reservation=
reservationRepository.findById(id).get();
        Ressource
ressource=resourceRestClient.findRessourceById(reserva
tion.getRessourceId());
        reservation.setRessource(ressource);
        return reservation;
    }
    @GetMapping("/users")
```

```
    public List<Personne> personneList(){
        List<Personne>
personneList=personneRepository.findAll();
        return personneList;
    }
    @GetMapping("/users/{id}")
    public Personne personneById(@PathVariable String
id) {
        Personne p =
personneRepository.findById(id).get();
        return p;
    }

    @PostMapping("/reserve")
    public ResponseEntity<Reservation>
createReservation(@RequestBody Reservation
reservation) {

System.out.println("--------------------------------")
;

System.out.println(reservation.getPersonneId());

System.out.println("--------------------------------")
;
        Personne personne =
personneRepository.findById(reservation.getPersonneId(
)).orElse(null);
        Ressource ressource =
resourceRestClient.findRessourceById(reservation.getRe
ssourceId());

        if (personne != null && ressource != null) {
            reservation.setPersonne(personne);
            reservation.setRessource(ressource);
            reservation.setName("reservation"+new
```

```
Random().nextInt(100));
            reservation.setDate(new Date());
            reservation.setContext("Informatique");
            reservation.setDuration(new
Random().nextInt(2, 30));
            Reservation createdReservation =
reservationRepository.save(reservation);

            return new
ResponseEntity<>(createdReservation,
HttpStatus.CREATED);
        } else {
            return new
ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
    }
    @PostMapping("/addUser")
    public ResponseEntity<Personne>
createPersonne(@RequestBody Personne p) {
        Personne personne =
personneRepository.findById(p.getId()).orElse(null);
        if (personne == null ) {
            Personne personne1= Personne.builder()
                    .id(p.getId())
                    .name(p.getName())
                    .email(p.getEmail())
                    .fonction("Unknow")
                    .build();
            Personne createdPersonne =
personneRepository.save(personne1);

            return new
ResponseEntity<>(createdPersonne, HttpStatus.CREATED);
        } else {
```

```
            return new
ResponseEntity<>(HttpStatus.BAD_REQUEST);
        }
    }
}
```

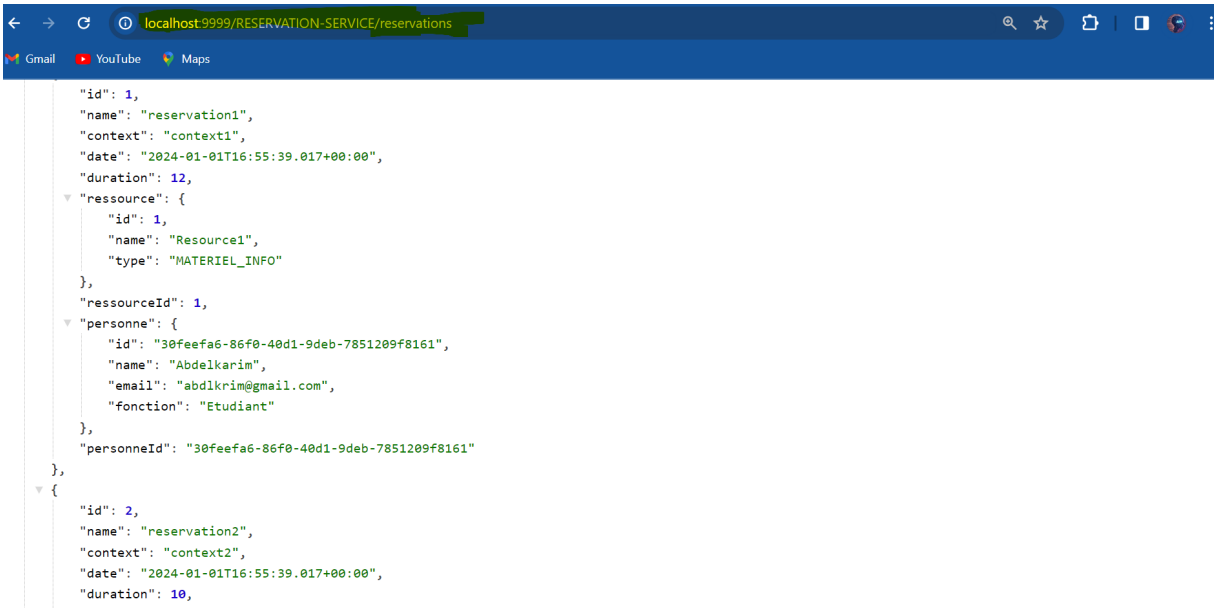
h) ReservationServiceApplication

```
package ma.sdia.reservationservice;


import ma.sdia.reservationservice.entities.Personne;
import
ma.sdia.reservationservice.entities.Reservation;
import ma.sdia.reservationservice.model.Ressource;
import
ma.sdia.reservationservice.repositories.PersonneReposi
tory;
import
ma.sdia.reservationservice.repositories.ReservationRep
ository;
import
ma.sdia.reservationservice.web.ResourceRestClient;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootAppli
cation;
import
org.springframework.cloud.openfeign.EnableFeignClients
;
import org.springframework.context.annotation.Bean;
import java.util.Date;
import java.util.List;
import java.util.Random;
import java.util.UUID;
```

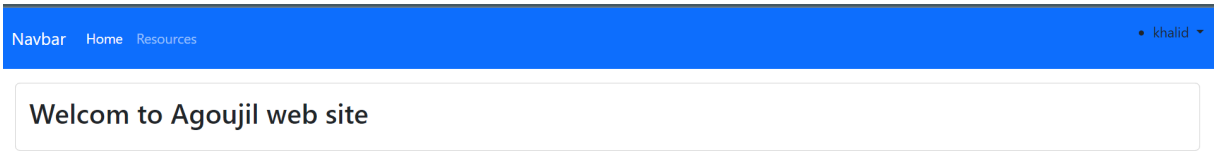
```
@SpringBootApplication
@EnableFeignClients
public class ReservationServiceApplication {

  public static void main(String[] args) {

SpringApplication.run(ReservationServiceApplication.class, args);
  }

  @Bean
  CommandLineRunner commandLineRunner(ReservationRepository reservationRepository, ResourceRestClient resourceRestClient, PersonneRepository personneRepository) {
     return args -> {
        Personne personne1=new Personne(UUID.randomUUID().toString(),"Abdelkarim","abdlkrim@gmail.com","Etudiant");
        Personne personne2=new Personne(UUID.randomUUID().toString(),"Agoujil","agoujil@gmail.com","Etudiant");
        personneRepository.save(personne1);
        personneRepository.save(personne2);


List<Ressource>ressourceList=resourceRestClient.allRessource();


        personneRepository.findAll().forEach(p -> {
           Reservation reservation1 = Reservation.builder()
                   .name("reservation1")
                   .context("context1")
                   .date(new Date())
```

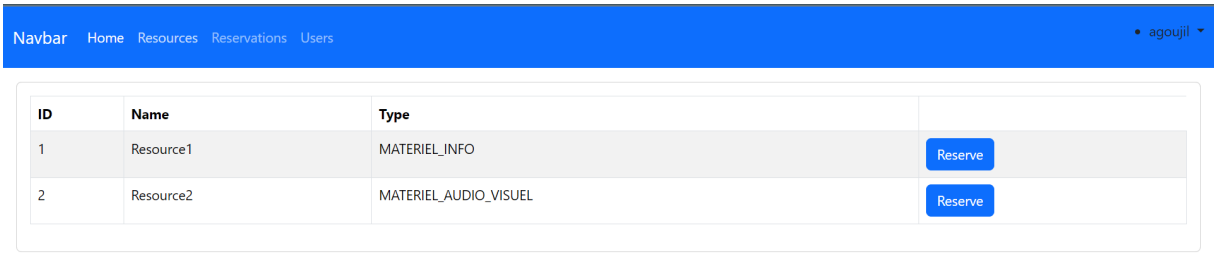
```
                .duration(new Random().nextInt(2,
30))

.ressourceId(ressourceList.get(0).getId())
                .personne(p)
                .personneId(p.getId())
                .build();
            Reservation reservation2 =
Reservation.builder()
                .name("reservation2")
                .context("context2")
                .date(new Date())
                .duration(new Random().nextInt(2,
30))

.ressourceId(ressourceList.get(1).getId())
                .personne(p)
                .personneId(p.getId())
                .build();
            reservationRepository.save(reservation1);
            reservationRepository.save(reservation2);
        });


    };


  }
}
```

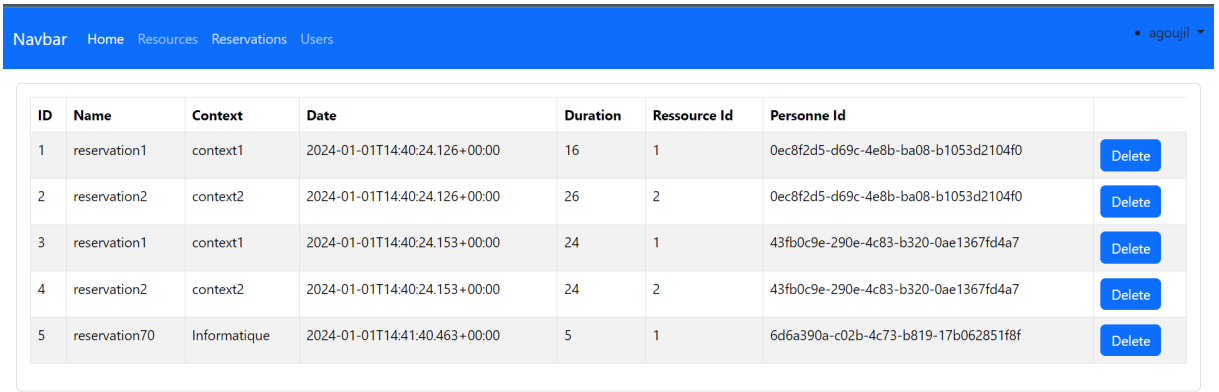
i) Teste:

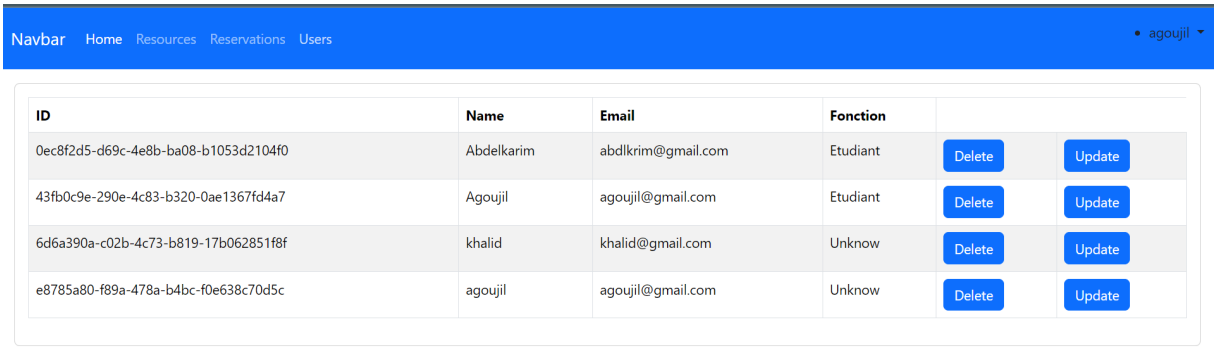


8. Développer un simple frontend web pour l'application:

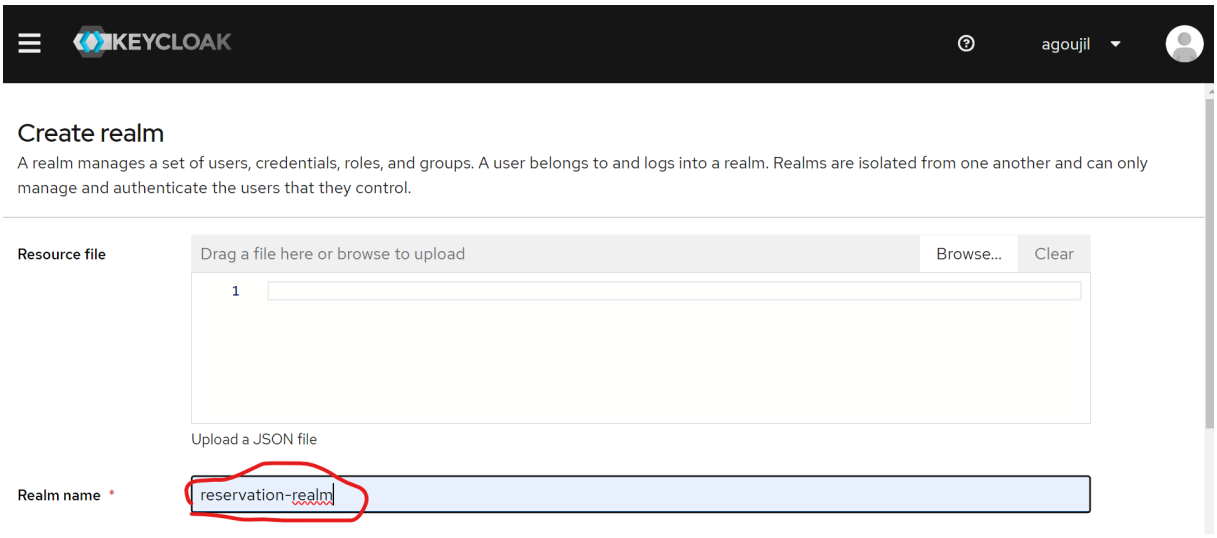


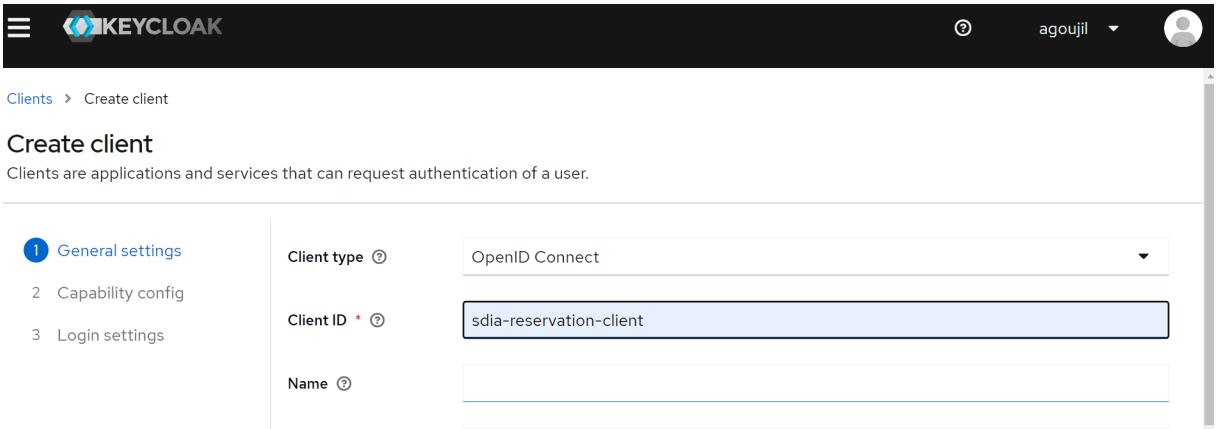


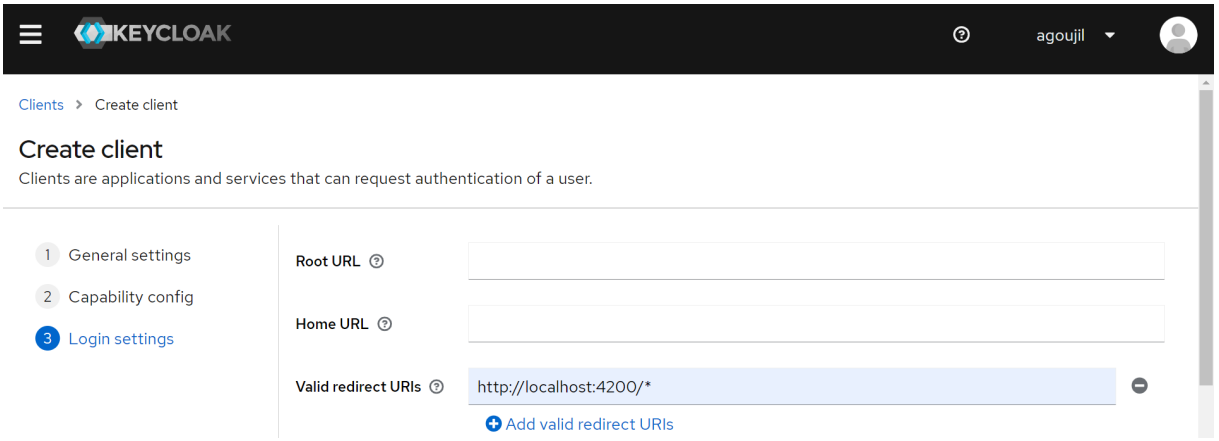




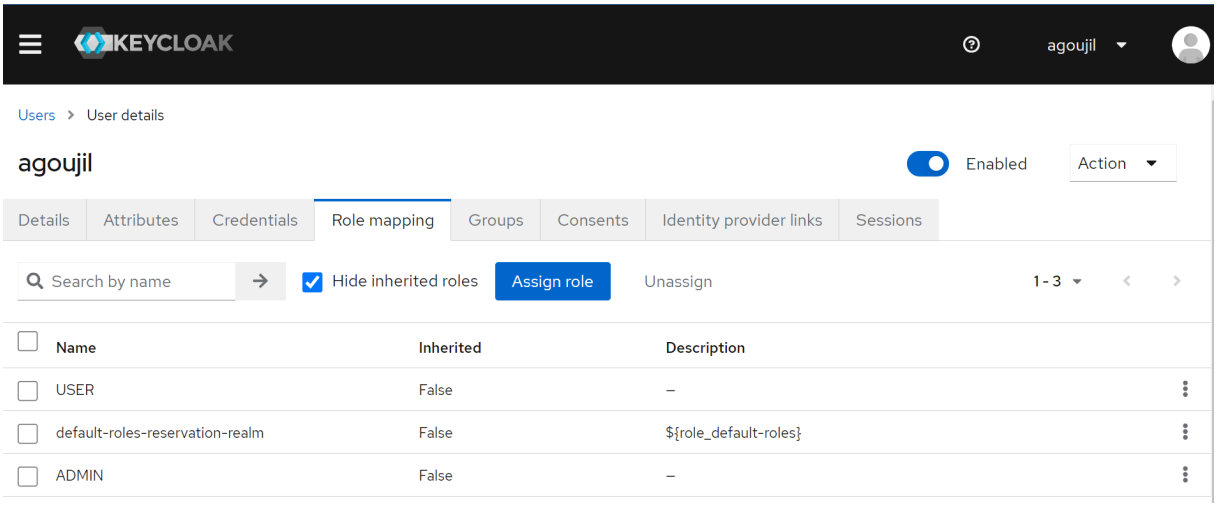
9. Sécuriser l'application avec une authentification Keycloak:

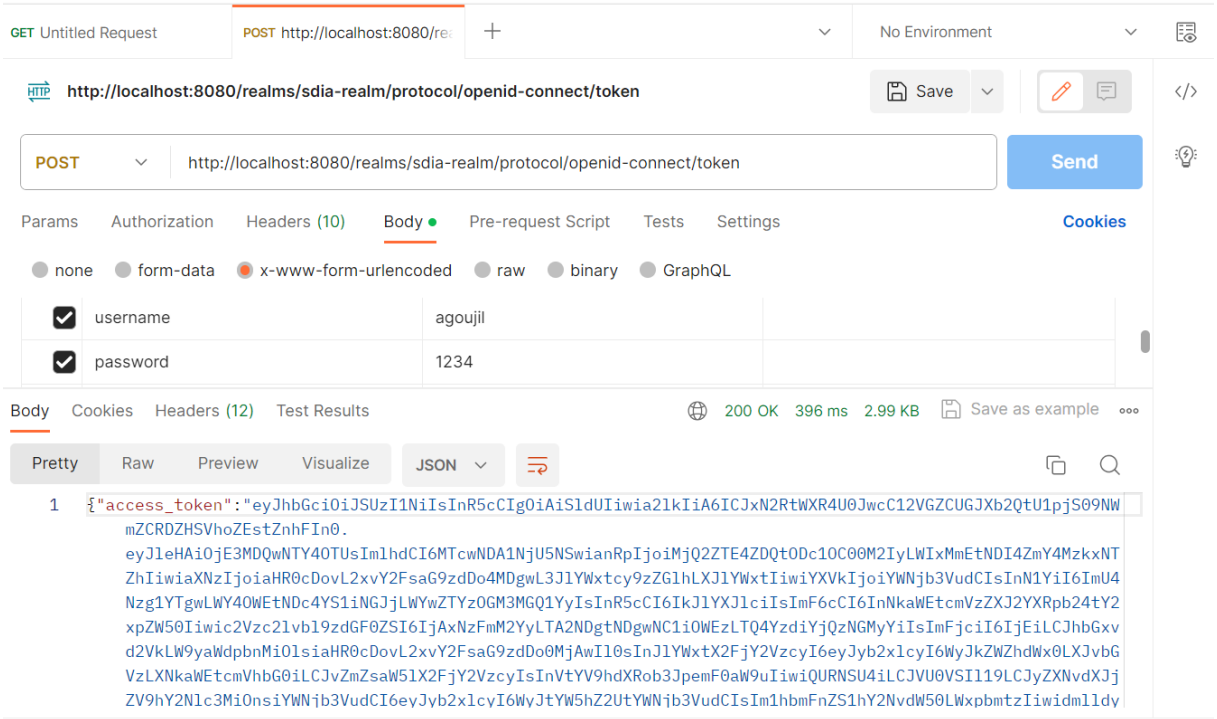


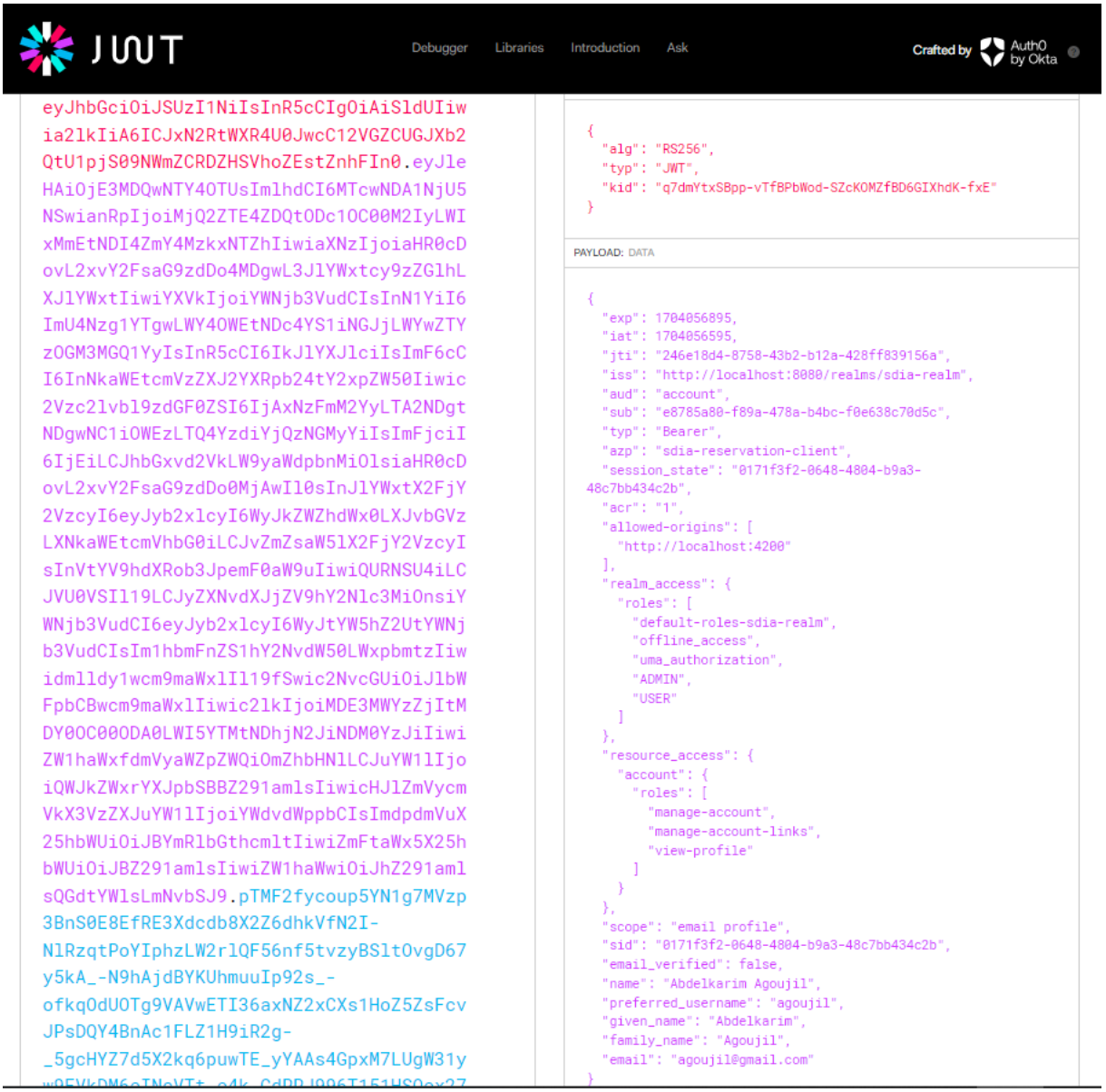


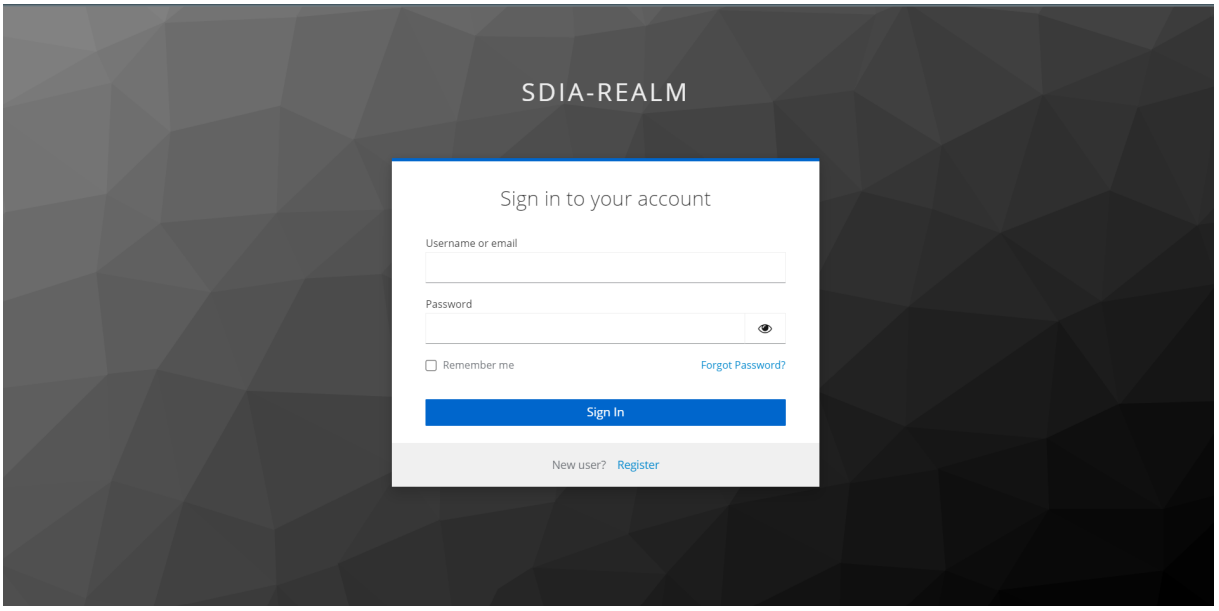

KEYCLOAK
agoujil
Clients
Create client
Create client
Clients are applications and services that can request authentication of a user.
1 General settings
2 Capability config
3 Login settings
Root URL
Home URL
Valid redirect URIs
http://localhost:4200/*
Add valid redirect URIs

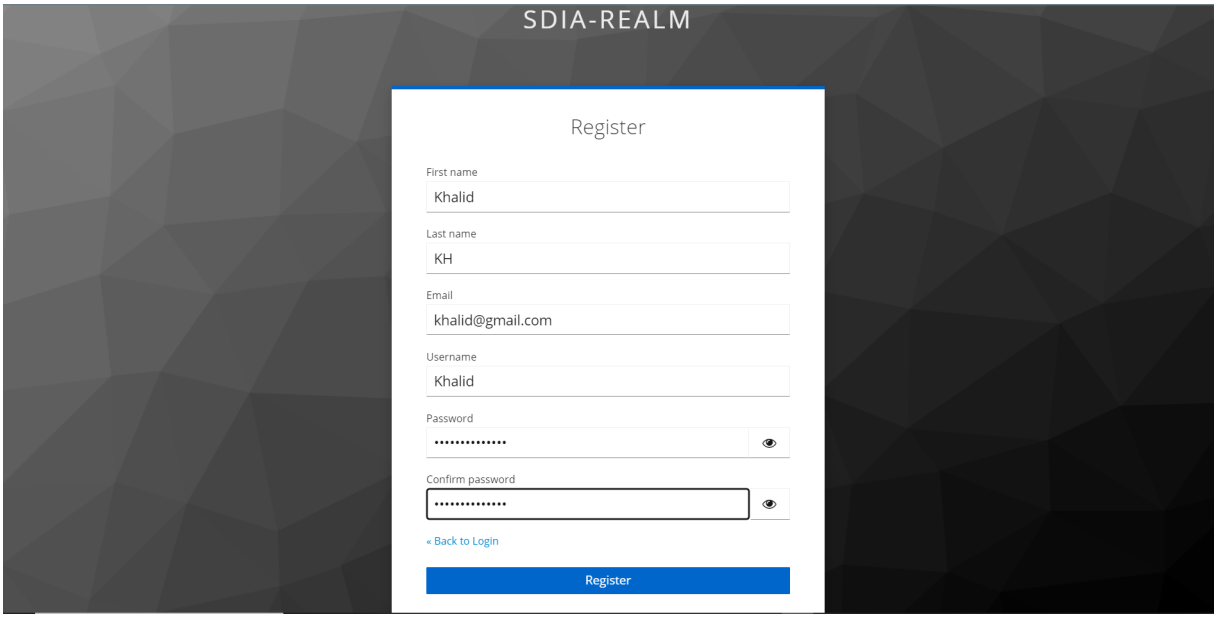

KEYCLOAK
agoujil
Create user
Required user actions
Select action
Username *
agoujil
Email
agoujil@gmail.com
Email verified
No
First name
ABDELKARIM
Last name
AGOUJIL


KEYCLOAK
agoujil
Users
User details
agoujil
Enabled
Action
Details
Attributes
Credentials
Role mapping
Groups
Consents
Identity provider links
Sessions
Search by name
Hide inherited roles
Assign role
Unassign
1 - 3
Name
Inherited
Description
USER
False
–
default-roles-reservation-realm
False
${role_default-roles}
ADMIN
False
–

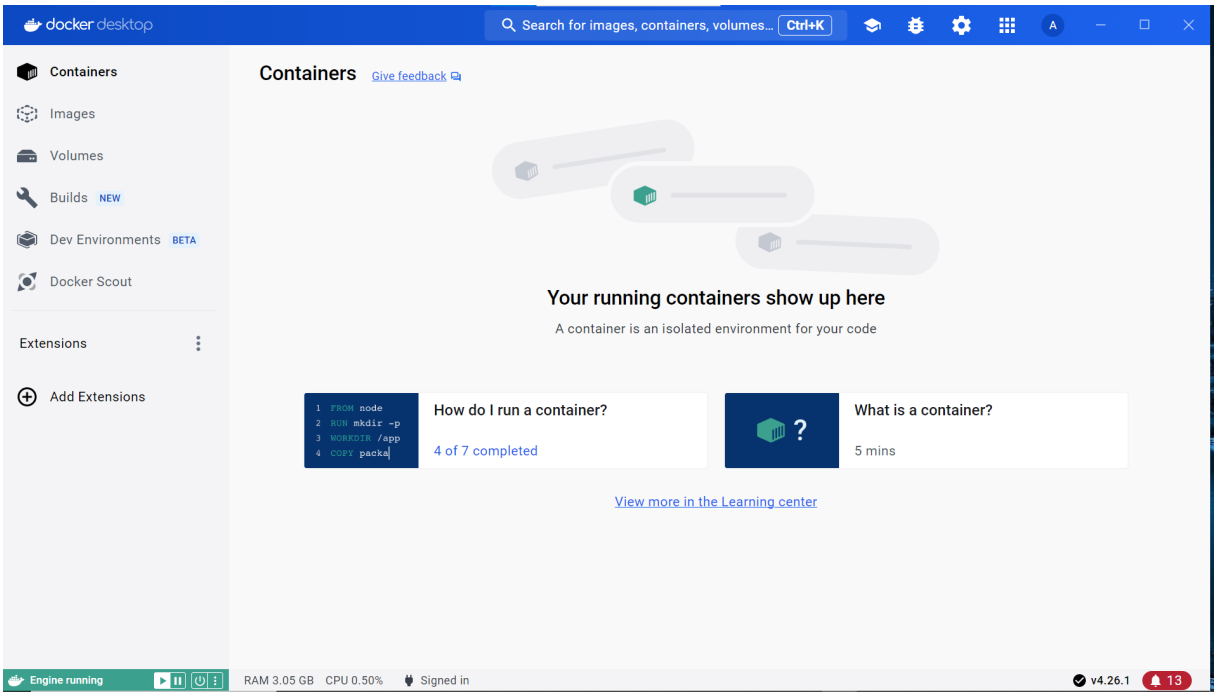

GET Untitled Request
POST http://localhost:8080/re
No Environment
http://localhost:8080/realms/sdia-realm/protocol/openid-connect/token
Save
POST
http://localhost:8080/realms/sdia-realm/protocol/openid-connect/token
Send
Params
Authorization
Headers (10)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
username
agoujil
password
1234
Body
Cookies
Headers (12)
Test Results
200 OK
396 ms
2.99 KB
Save as example
Pretty
Raw
Preview
Visualize
JSON
1
{"access_token":"eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJxN2RtWXR4U0JwcC12VGZCUGJXb2QtU1pjS09NW
mZCRDZHSVhoZEstZnhFIn0.
eyJleHAiOjE3MDQwNTY4OTUsImlhdCI6MTcwNDA1NjU5NSwianRpIjoiMjQ2ZTE4ZDQtODc1OC00M2IyLWIxMmEtNDI4ZmY4MzkxNT
ZhIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdDo4MDgwL3JlYWxtcy9zZGlhLXJlYWxtIiwiYXVkIjoiYWNjb3VudCIsInN1YiI6ImU4
Nzg1YTgwLWY4OWEtNDc4YS1iNGJjLWYwZTYzOGM3MGQ1YyIsInR5cCI6IkJlYXJlciIsImF6cCI6InNkaWEtcmVzZXJ2YXRpb24tY2
xpZW50Iiwic2Vzc2lvbl9zdGF0ZSI6IjAxNzFmM2YyLTA2NDgtNDgwNC1iOWEzLTQ4YzdiYjQzNGMyYiIsImFjciI6IjEiLCJhbGxv
d2VkLW9yaWdpbnMiOlsiaHR0cDovL2xvY2FsaG9zdDo0MjAwIl0sInJlYWxtX2FjY2VzcyI6eyJyb2xlcyI6WyJkZWZhdWx0LXJvbG
VzLXNkaWEtcmVhbG0iLCJvZmZsaW51X2FjY2VzcyIsInVtYV9hdXRob3JpemF0aW9uIiwiQURNSU4iLCJVU0VSI119LCJyZXNvdXJj
ZV9hY2Nlc3MiOnsiYWNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNjb3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiwidmlldy

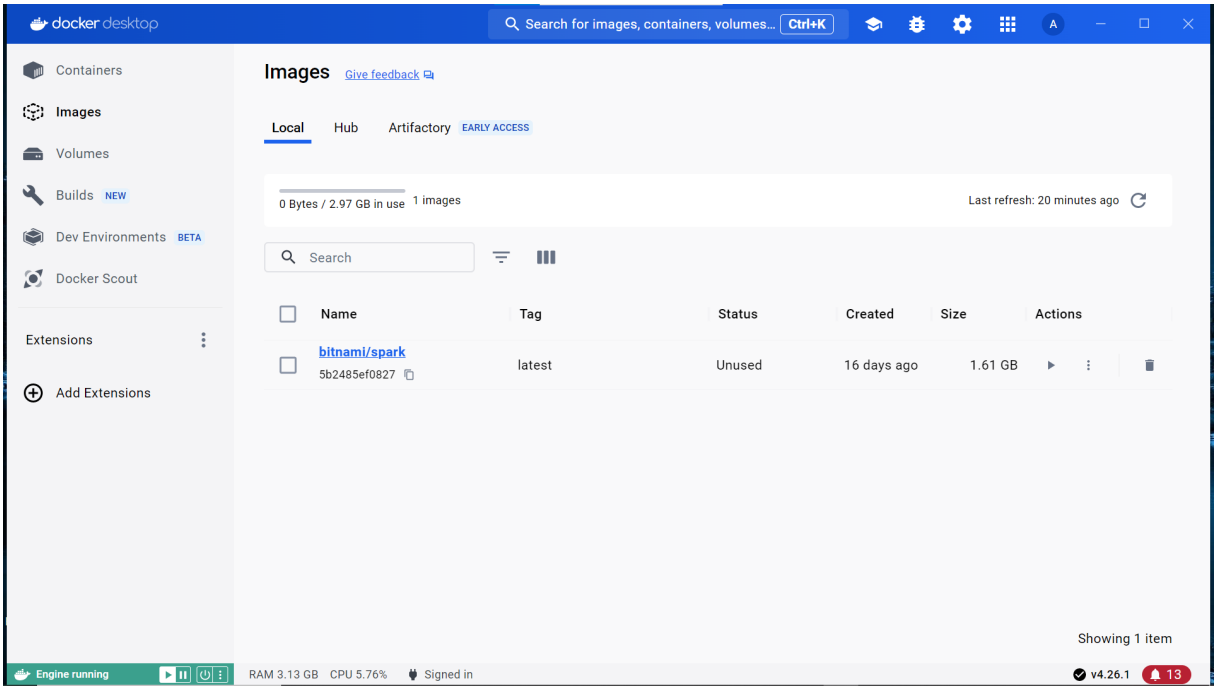

JWT
Debugger
Libraries
Introduction
Ask
Crafted by Auth0 by Okta
eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiw
ia2lkIiA6ICJxN2RtWXR4U0JwcC12VGZCUGJXb2
QtU1pjS09NWmZCRDZHSVhoZEstZnhFIn0.eyJle
HAiOjE3MDQwNTY4OTUsImlhdCI6MTcwNDA1NjU5
NSwianRpIjoiMjQ2ZTE4ZDQtODc1OC00M2IyLWI
xMmEtNDI4ZmY4MzkxNTZhIiwiaXNzIjoiaHR0cD
ovL2xvY2FsaG9zdDo4MDgwL3JlYWxtcy9zZGlhL
XJlYWxtIiwiYXVkIjoiYWNjb3VudCIsInN1YiI6
ImU4Nzg1YTgwLWY4OWEtNDc4YS1iNGJjLWYwZTY
zOGM3MGQ1YyIsInR5cCI6IkJlYXJlciIsImF6cC
I6InNkaWEtcmVzZXJ2YXRpb24tY2xpZW50Iiwic
2Vzc2lvbl9zdGF0ZSI6IjAxNzFmM2YyLTA2NDgt
NDgwNC1iOWEzLTQ4YzdiYjQzNGMyYiIsImFjciI
6IjEiLCJhbGxvd2VkLW9yaWdpbnMiOlsiaHR0cD
ovL2xvY2FsaG9zdDo0MjAwIl0sInJlYWxtX2FjY
2VzcyI6eyJyb2xlcyI6WyJkZWZhdWx0LXJvbGVz
LXNkaWEtcmVhbG0iLCJvZmZsaW5lX2FjY2VzcyI
sInVtYV9hdXRob3JpemF0aW9uIiwiQURNSU4iLC
JVU0VSIl19LCJyZXNvdXJjZV9hY2Nlc3MiOnsiY
WNjb3VudCI6eyJyb2xlcyI6WyJtYW5hZ2UtYWNj
b3VudCIsIm1hbmFnZS1hY2NvdW50LWxpbmtzIiw
idmlldy1wcm9maWxlIl19fSwic2NvcGUiOiJlbW
FpbCBwcm9maWxlIiwic2lkIjoiMDE3MWYzZjItM
DY0OC00ODA0LWI5YTMtNDhjN2JiNDM0YzJiIiwi
ZW1haWxfdmVyaWZpZWQiOmZhbHNlLCJuYW1lIjo
iQWJkZWxrYXJpbSBBZ291amlsIiwicHJlZmVycm
VkX3VzZXJuYW1lIjoiYWdvdWppbCIsImdpdmVuX
25hbWUiOiJBYmRlbGthcmltIiwiZmFtaWx5X25h
bWUiOiJBZ291amlsIiwiZW1haWwiOiJhZ291aml
sQGdtYWlsLmNvbSJ9.pTMF2fycoup5YN1g7MVzp
3BnS0E8EfRE3Xdcdb8X2Z6dhkVfN2I-
NlRzqtPoYIphzLW2rlQF56nf5tvzyBS1tOvgD67
y5kA_-N9hAjdBYKUhmuuIp92s_-
ofkqOdUOTg9VAVwETI36axNZ2xCXs1HoZ5ZsFcv
JPsDQY4BnAc1FLZ1H9iR2g-
_5gcHYZ7d5X2kq6puwTE_yYAAs4GpxM7LUgW31y
{
"alg": "RS256",
"typ": "JWT",
"kid": "q7dmYtxSBpp-vTfBPbWod-SZcKOMZfBD6GIXhdK-fxE"
}
PAYLOAD: DATA
{
"exp": 1704056895,
"iat": 1704056595,
"jti": "246e18d4-8758-43b2-b12a-428ff839156a",
"iss": "http://localhost:8080/realms/sdia-realm",
"aud": "account",
"sub": "e8785a80-f89a-478a-b4bc-f0e638c70d5c",
"typ": "Bearer",
"azp": "sdia-reservation-client",
"session_state": "0171f3f2-0648-4804-b9a3-
48c7bb434c2b",
"acr": "1",
"allowed-origins": [
"http://localhost:4200"
],
"realm_access": {
"roles": [
"default-roles-sdia-realm",
"offline_access",
"uma_authorization",
"ADMIN",
"USER"
]
},
"resource_access": {
"account": {
"roles": [
"manage-account",
"manage-account-links",
"view-profile"
]
}
},
"scope": "email profile",
"sid": "0171f3f2-0648-4804-b9a3-48c7bb434c2b",
"email_verified": false,
"name": "Abdelkarim Agoujil",
"preferred_username": "agoujil",
"given_name": "Abdelkarim",
"family_name": "Agoujil",
"email": "agoujil@gmail.com"
}


SDIA-REALM
Sign in to your account
Username or email
Password
Remember me
Forgot Password?
Sign In
New user? Register

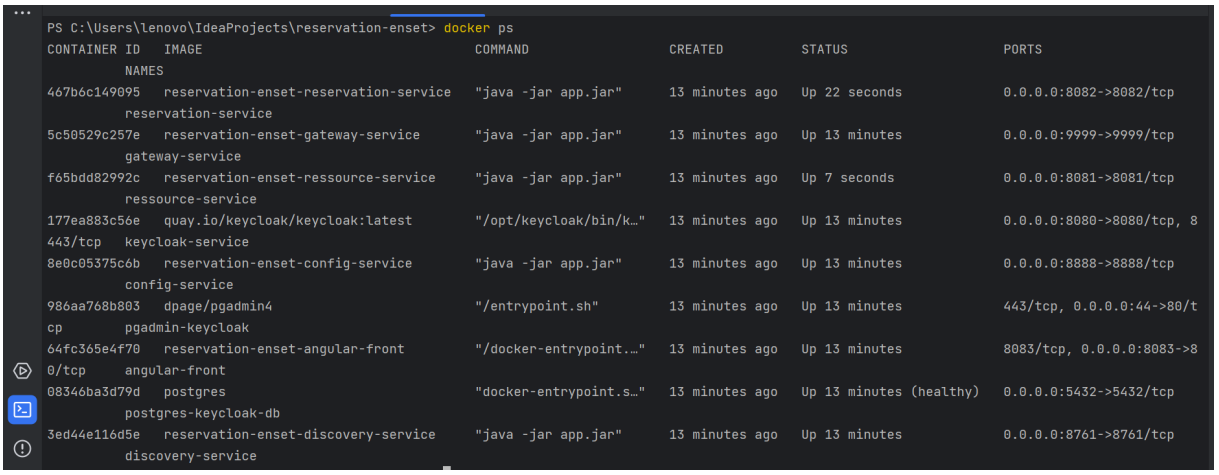


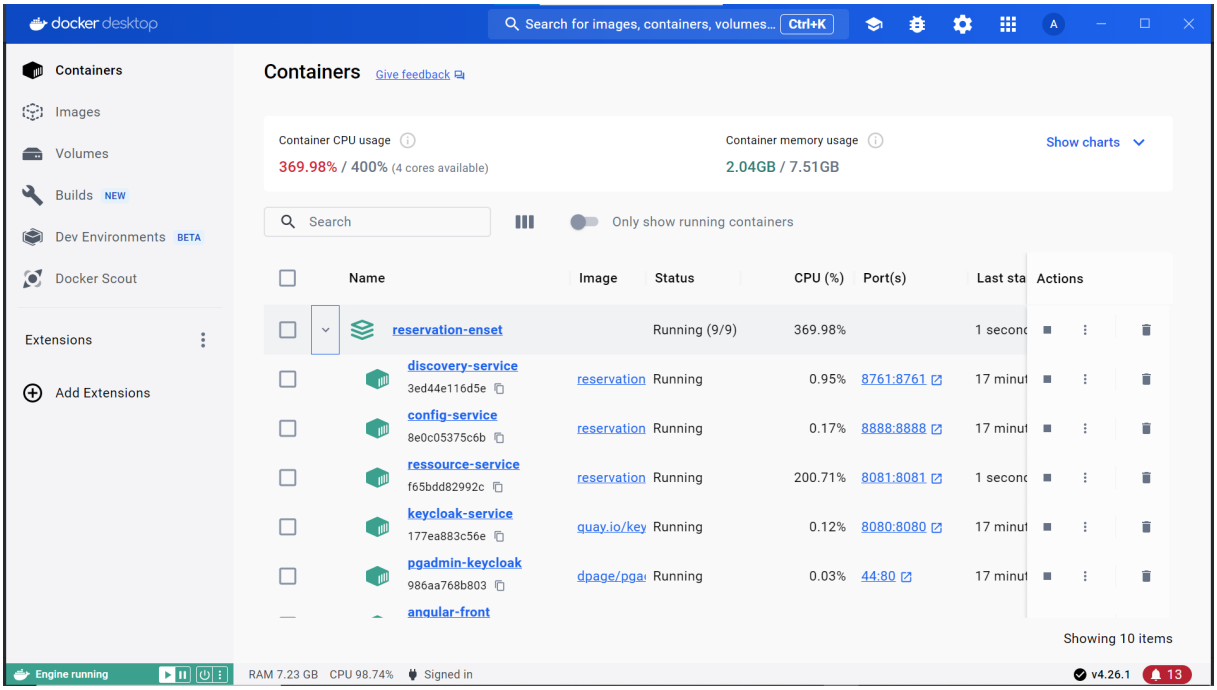
10. Déployer l'application avec Docker et Docker compose :




docker desktop
Search for images, containers, volumes... Ctrl+K
Containers
Images
Volumes
Builds NEW
Dev Environments BETA
Docker Scout
Extensions
Add Extensions
Images
Give feedback
Local
Hub
Artifactory EARLY ACCESS
0 Bytes / 2.97 GB in use
1 images
Last refresh: 20 minutes ago
Search
Name
Tag
Status
Created
Size
Actions
bitnami/spark
5b2485ef0827
latest
Unused
16 days ago
1.61 GB
Showing 1 item
Engine running
RAM 3.13 GB
CPU 5.76%
Signed in
v4.26.1
13

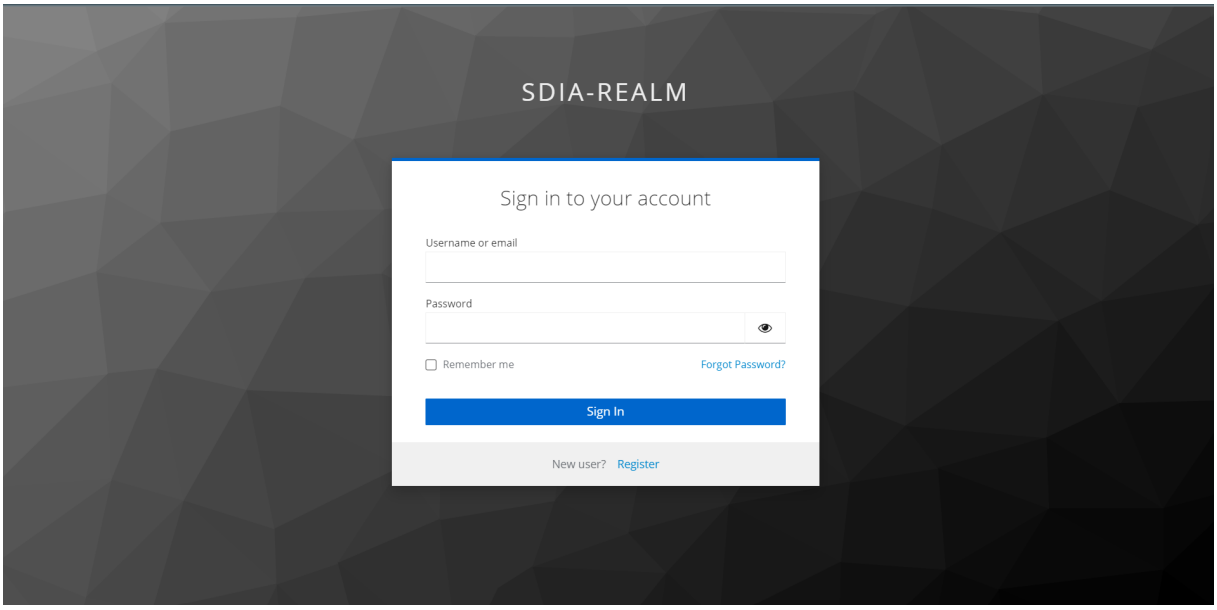

[+] Running 9/10covery-service Started
- Network reservation-enset_default Created
- Container postgres-keycloak-db Started
- Container angular-front Started
- Container discovery-service Started
- Container pgadmin-keycloak Started
- Container config-service Started
- Container keycloak-service Started
- Container gateway-service Started
- Container ressource-service Started
- Container reservation-service Started


PS C:\Users\lenovo\IdeaProjects\reservation-enset> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
467b6c149095 reservation-enset-reservation-service "java -jar app.jar" 13 minutes ago Up 22 seconds 0.0.0.0:8082->8082/tcp reservation-service
5c50529c257e reservation-enset-gateway-service "java -jar app.jar" 13 minutes ago Up 13 minutes 0.0.0.0:9999->9999/tcp gateway-service
f65bdd82992c reservation-enset-ressource-service "java -jar app.jar" 13 minutes ago Up 7 seconds 0.0.0.0:8081->8081/tcp ressource-service
177ea883c56e quay.io/keycloak/keycloak:latest "/opt/keycloak/bin/k…" 13 minutes ago Up 13 minutes 0.0.0.0:8080->8080/tcp, 8443/tcp keycloak-service
8e0c05375c6b reservation-enset-config-service "java -jar app.jar" 13 minutes ago Up 13 minutes 0.0.0.0:8888->8888/tcp config-service
986aa768b803 dpage/pgadmin4 "/entrypoint.sh" 13 minutes ago Up 13 minutes 443/tcp, 0.0.0.0:44->80/tcp pgadmin-keycloak
64fc365e4f70 reservation-enset-angular-front "/docker-entrypoint.…" 13 minutes ago Up 13 minutes 8083/tcp, 0.0.0.0:8083->80/tcp angular-front
08346ba3d79d postgres "docker-entrypoint.s…" 13 minutes ago Up 13 minutes (healthy) 0.0.0.0:5432->5432/tcp postgres-keycloak-db
3ed44e116d5e reservation-enset-discovery-service "java -jar app.jar" 13 minutes ago Up 13 minutes 0.0.0.0:8761->8761/tcp discovery-service

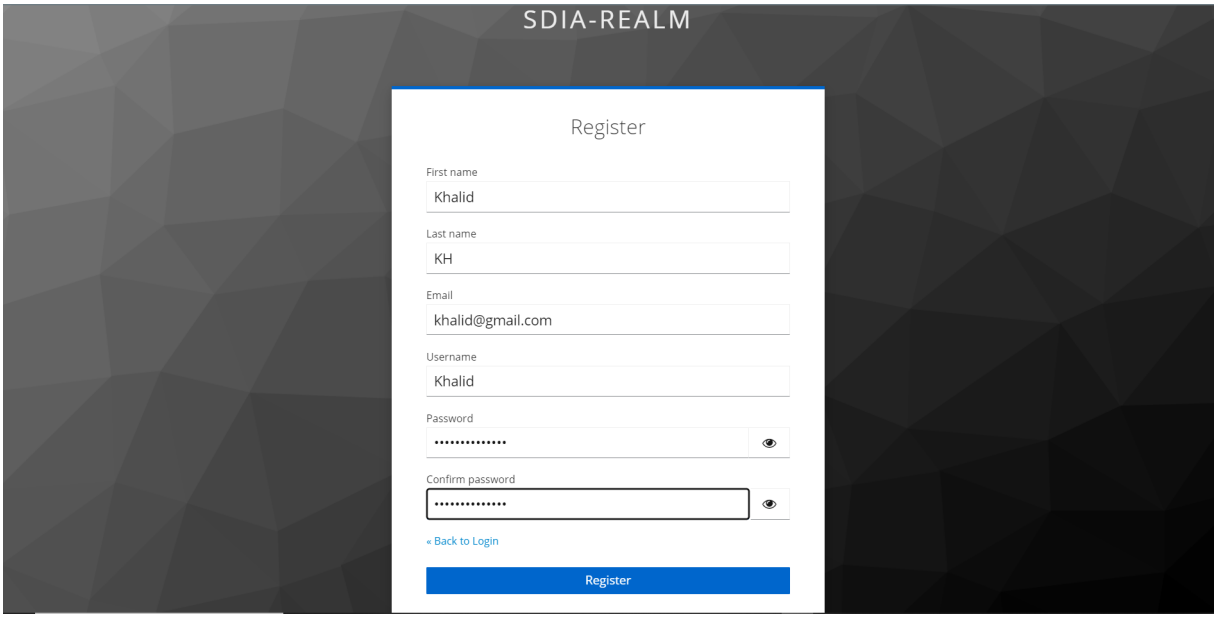


11. Tester l'Application:

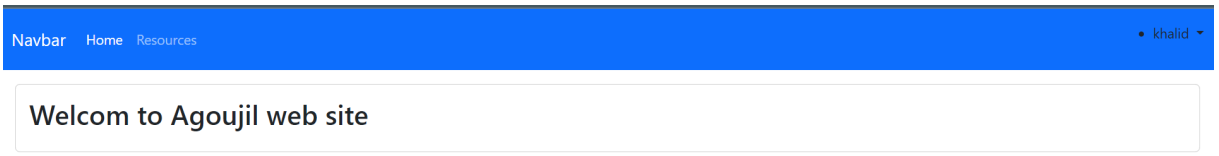
1. Authentification:



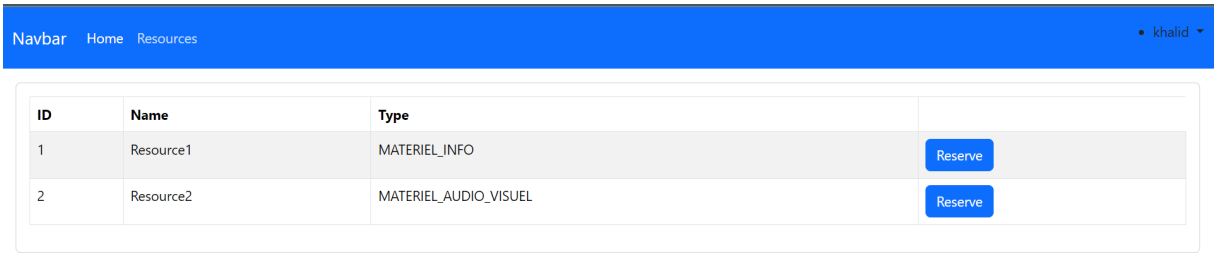
2. Register



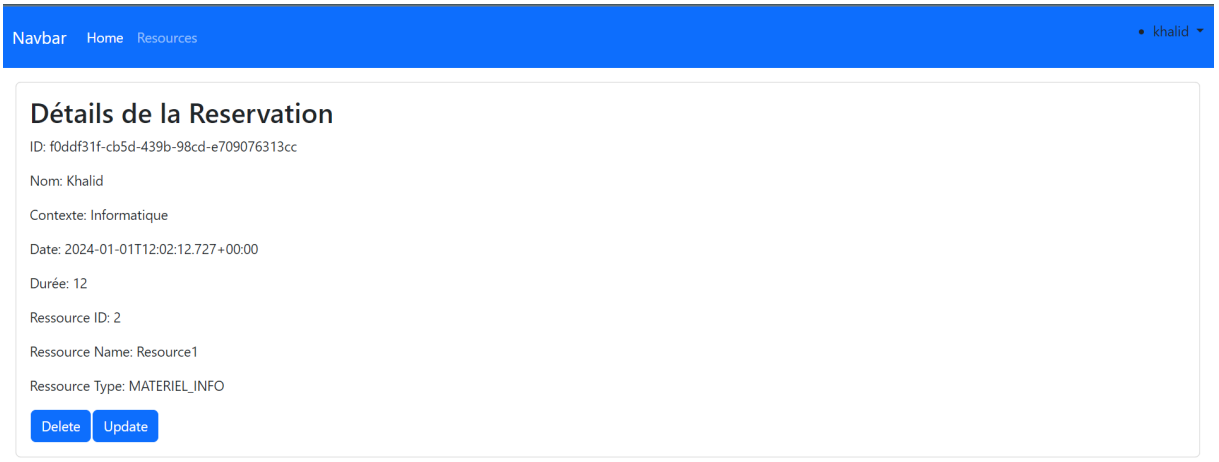
3. l'interface pour les clients

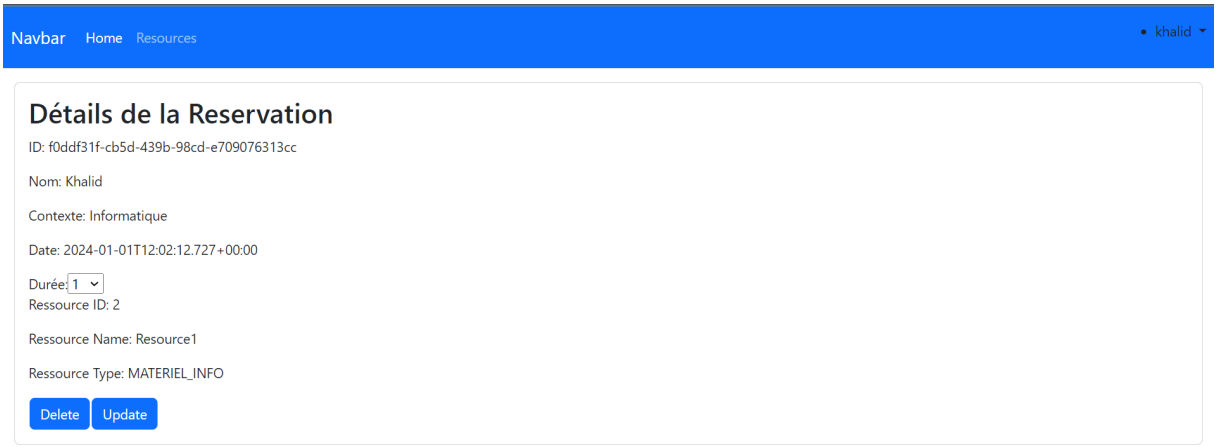


4. Réserver une ressource

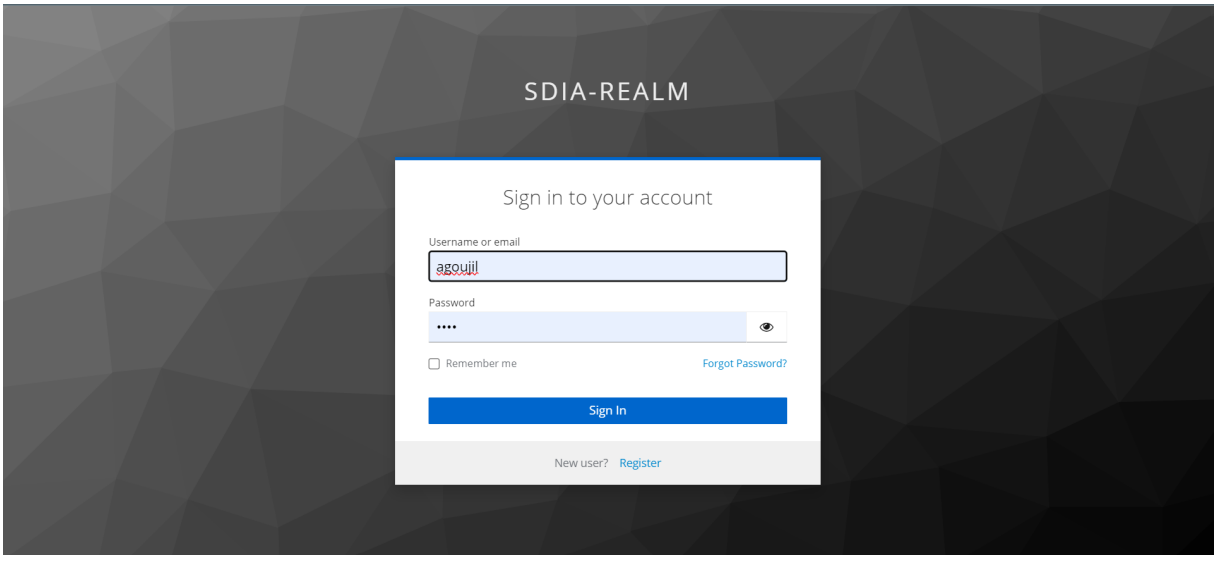


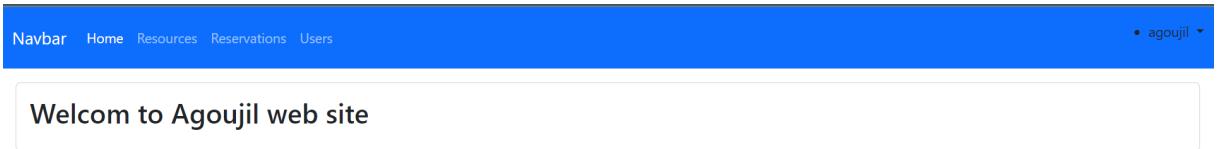
5. Consulter et modifier la réservation





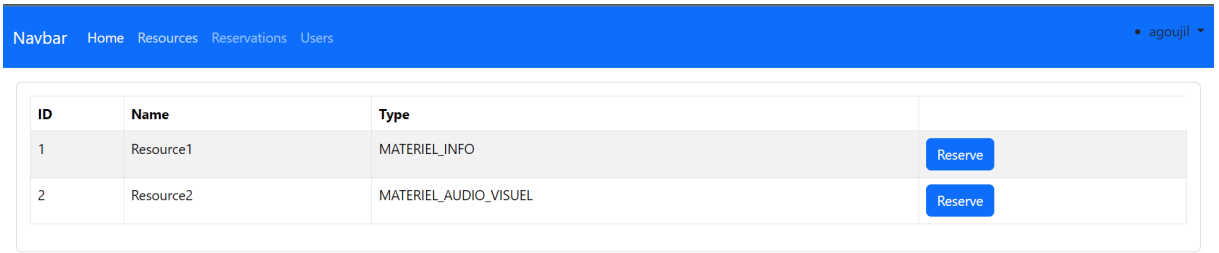
6. l'interface pour l'Admin



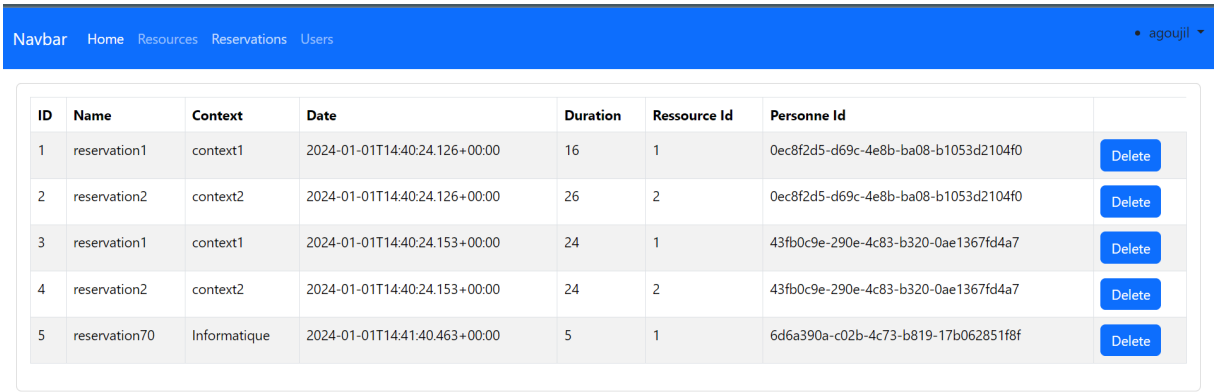


Welcom to Agoujil web site

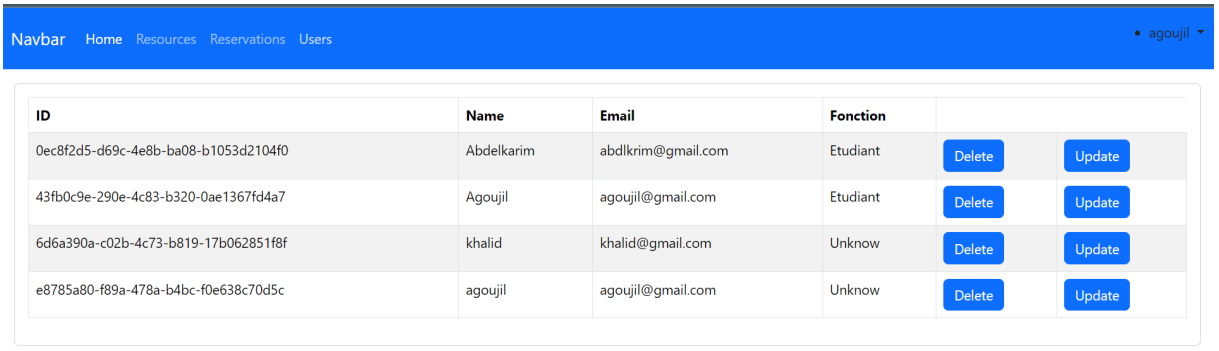
7. Consulter les ressources



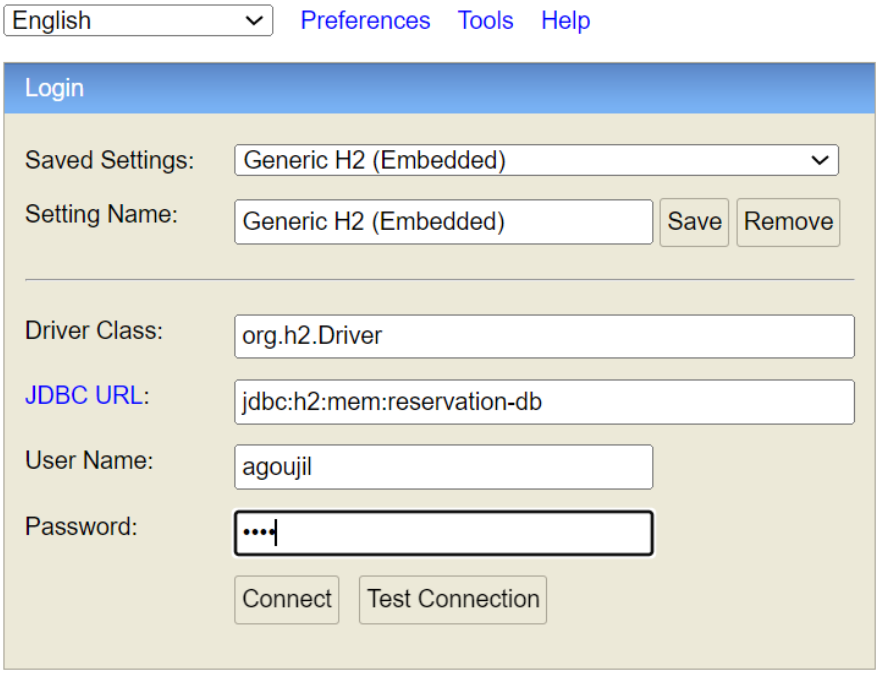
8. Consulter les réservations

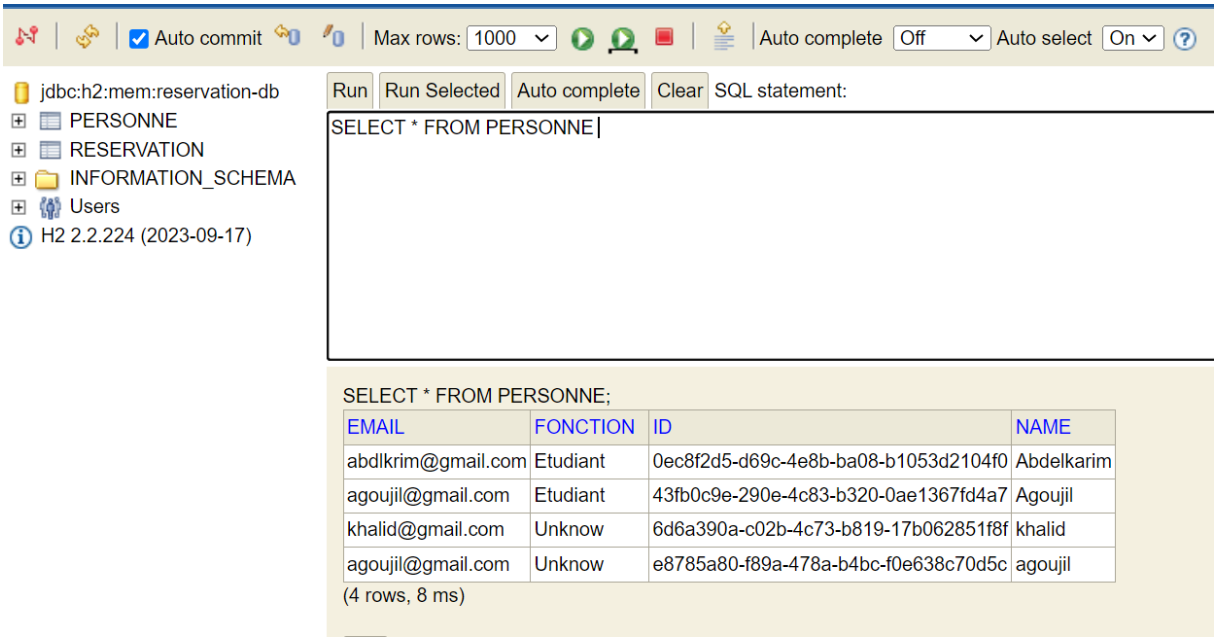


9. Consulter les utilisateurs



10. Consulter H2 data base

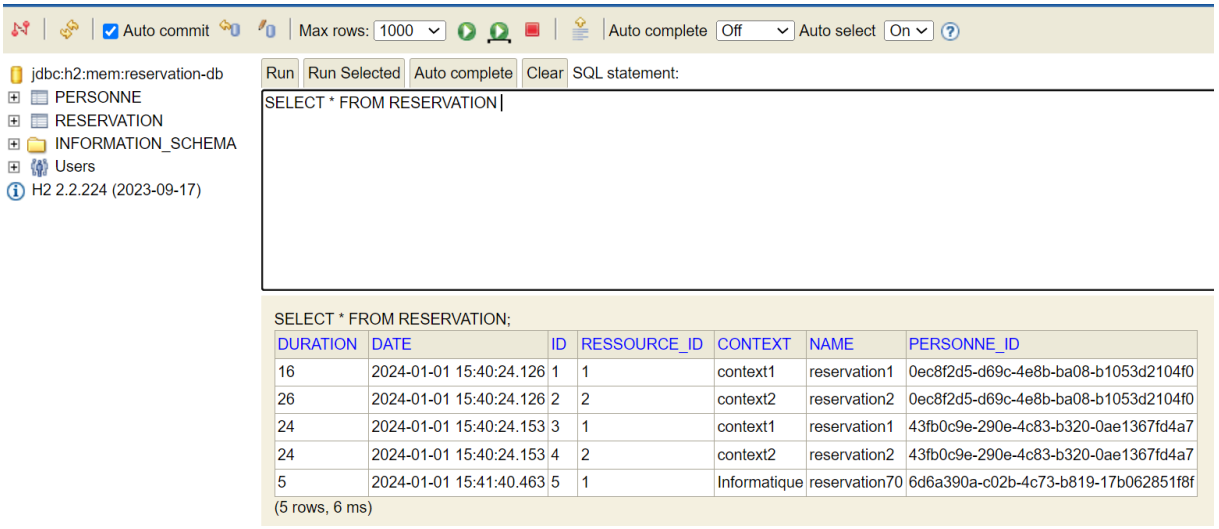




SELECT * FROM PERSONNE;

EMAIL	FONCTION	ID	NAME
abdlkrim@gmail.com	Etudiant	0ec8f2d5-d69c-4e8b-ba08-b1053d2104f0	Abdelkarim
agoujil@gmail.com	Etudiant	43fb0c9e-290e-4c83-b320-0ae1367fd4a7	Agoujil
khalid@gmail.com	Unknow	6d6a390a-c02b-4c73-b819-17b062851f8f	khalid
agoujil@gmail.com	Unknow	e8785a80-f89a-478a-b4bc-f0e638c70d5c	agoujil

(4 rows, 8 ms)


Auto commit
Max rows: 1000
Auto complete Off
Auto select On
jdbc:h2:mem:reservation-db
PERSONNE
RESERVATION
INFORMATION_SCHEMA
Users
H2 2.2.224 (2023-09-17)
Run
Run Selected
Auto complete
Clear
SQL statement:
SELECT * FROM RESERVATION
SELECT * FROM RESERVATION;
DURATION DATE ID RESSOURCE_ID CONTEXT NAME PERSONNE_ID
16 2024-01-01 15:40:24.126 1 1 context1 reservation1 0ec8f2d5-d69c-4e8b-ba08-b1053d2104f0
26 2024-01-01 15:40:24.126 2 2 context2 reservation2 0ec8f2d5-d69c-4e8b-ba08-b1053d2104f0
24 2024-01-01 15:40:24.153 3 1 context1 reservation1 43fb0c9e-290e-4c83-b320-0ae1367fd4a7
24 2024-01-01 15:40:24.153 4 2 context2 reservation2 43fb0c9e-290e-4c83-b320-0ae1367fd4a7
5 2024-01-01 15:41:40.463 5 1 Informatique reservation70 6d6a390a-c02b-4c73-b819-17b062851f8f
(5 rows, 6 ms)