



MASTER DISTRIBUTED SYSTEMES AND ARTIFICIAL INTELLIGENCE

Distributed Systemes

Rapport Examen Final

Réalisé par :

Soufiane MOUHTARAM

Professor :

Pr. YOUSSEF Moahemd

1^{er} janvier 2024

TABLE DE CONTENTES

Table de Contentes	1
Introduction	2
Architecture	4
Consul Discovery	6
Keycloak	7
Swagger	8
Gateway Ressource-Service	10
Screen d'écrans de l'architecture de project Angular	11
Java diagram class generated by IntelliJ IDE	12
La structure de chaque micro-service	13
Les entities et DTOs de Ressource Service	13
Les entities et DTOs de Reservation Service	14
Open Feign et Circuit Breaker	15
Securité de reservation service	17

Ce rapport décrit la conception et la mise en œuvre d'une application de gestion de ressources et de réservations basée sur les microservices. L'application est sécurisée avec Keycloak et déployée avec Docker.

Les Technologies et les outils utilisess

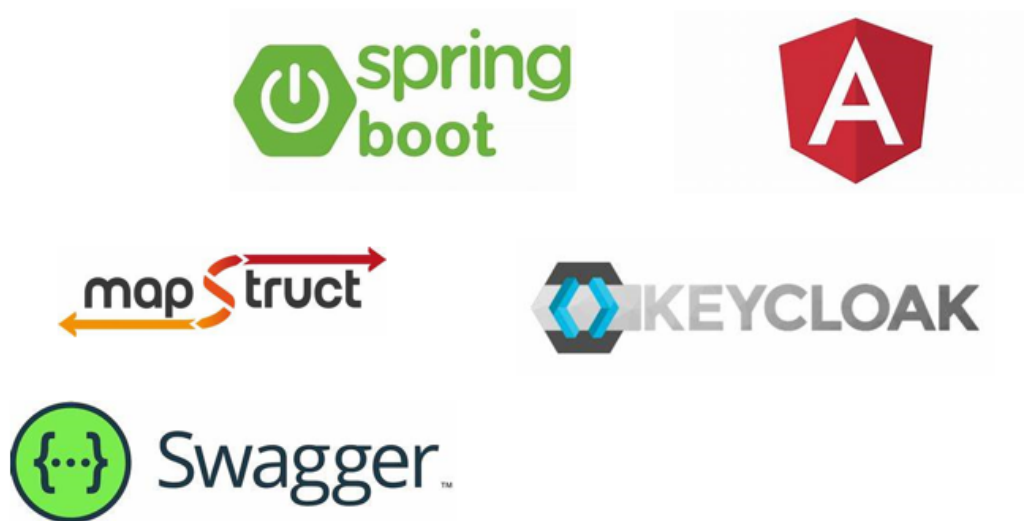


FIGURE 1 – Frameworks

Les outils de development



FIGURE 2 – Tools

Architecture

L'application est composée de deux microservices :

Le microservice ressource est responsable de la gestion des ressources de l'application. Il fournit une API qui permet aux autres microservices d'accéder aux ressources, telles que les données, les fichiers et les services externes.

Le microservice reservation est responsable de la gestion des réservations de l'application. Il fournit une API qui permet aux utilisateurs de réserver des ressources.

Sécurité

L'application est sécurisée avec Keycloak. Keycloak est une solution d'authentification et d'autorisation open source. Elle permet de sécuriser les applications microservices en garantissant que seuls les utilisateurs autorisés peuvent accéder aux ressources.

Les utilisateurs doivent s'authentifier auprès de Keycloak avant de pouvoir accéder à l'application. Keycloak fournit une variété de méthodes d'authentification, telles que l'authentification par mot de passe, l'authentification par certificat et l'authentification sociale.

Déploiement

L'application est déployée avec Docker. Docker est une plateforme de conteneurisation qui permet de déployer des applications microservices de manière simple et efficace. Les microservices sont empaquetés dans des conteneurs, qui sont des images légères et portables.

Les conteneurs sont déployés sur un cluster Kubernetes. Kubernetes est un système de gestion de conteneurs open source qui permet de gérer et de mettre à l'échelle des applications composées de microservices.

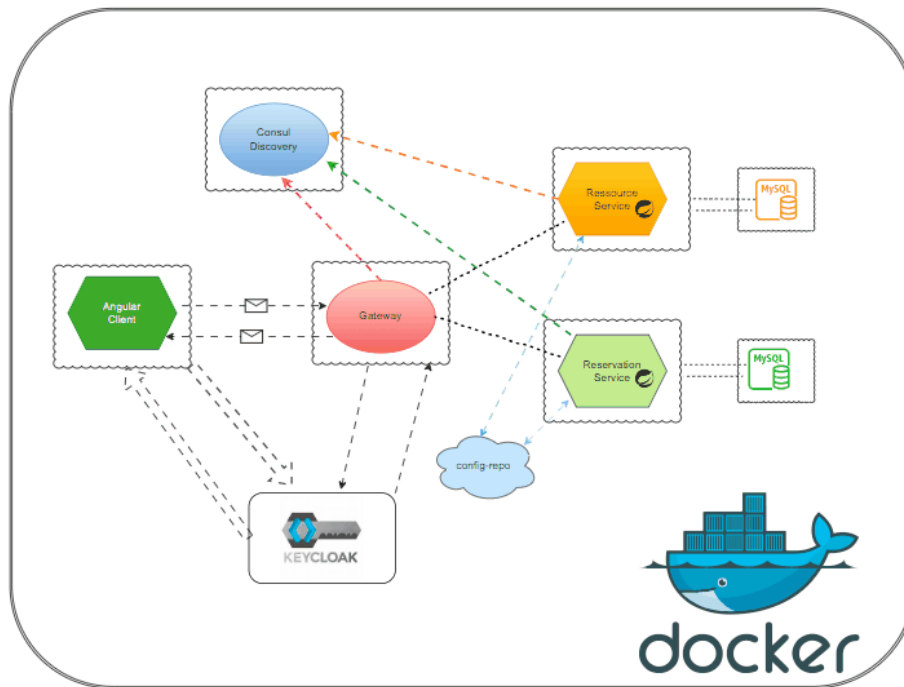


FIGURE 3 – Architecture de projet

ConsulDiscovery Consul Discovery :

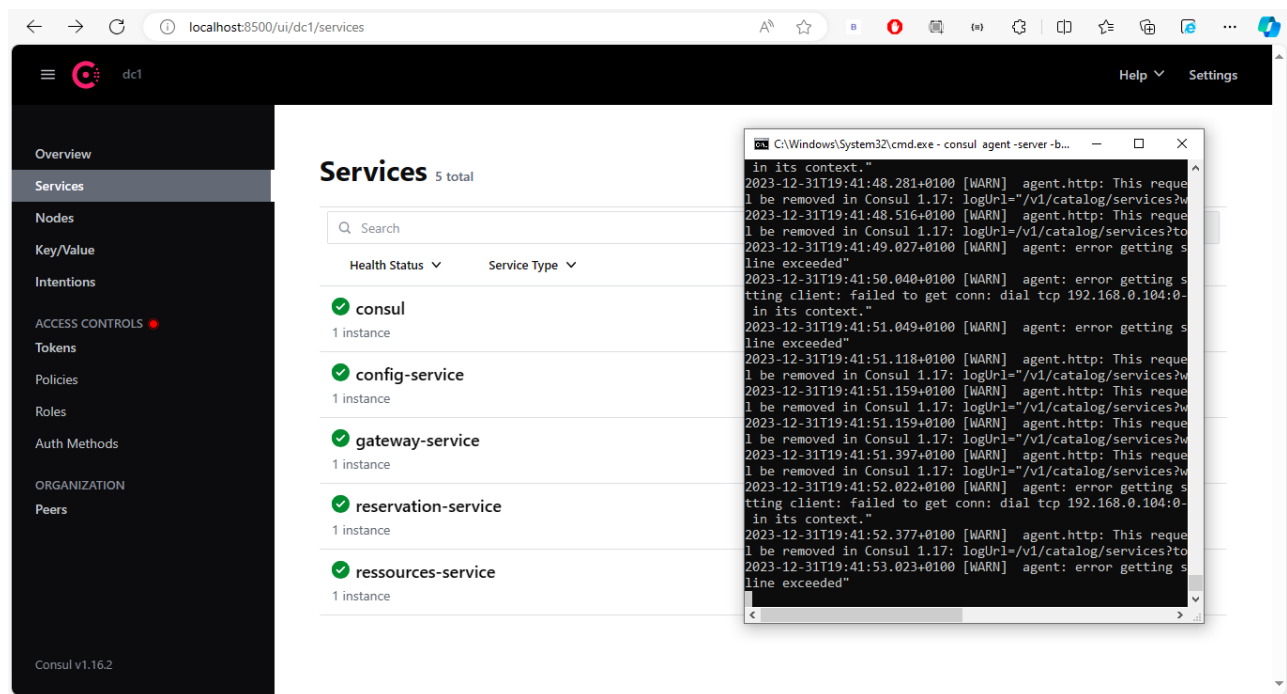


FIGURE 4 – Consul Discovery

Keycloak Keycloak interface :

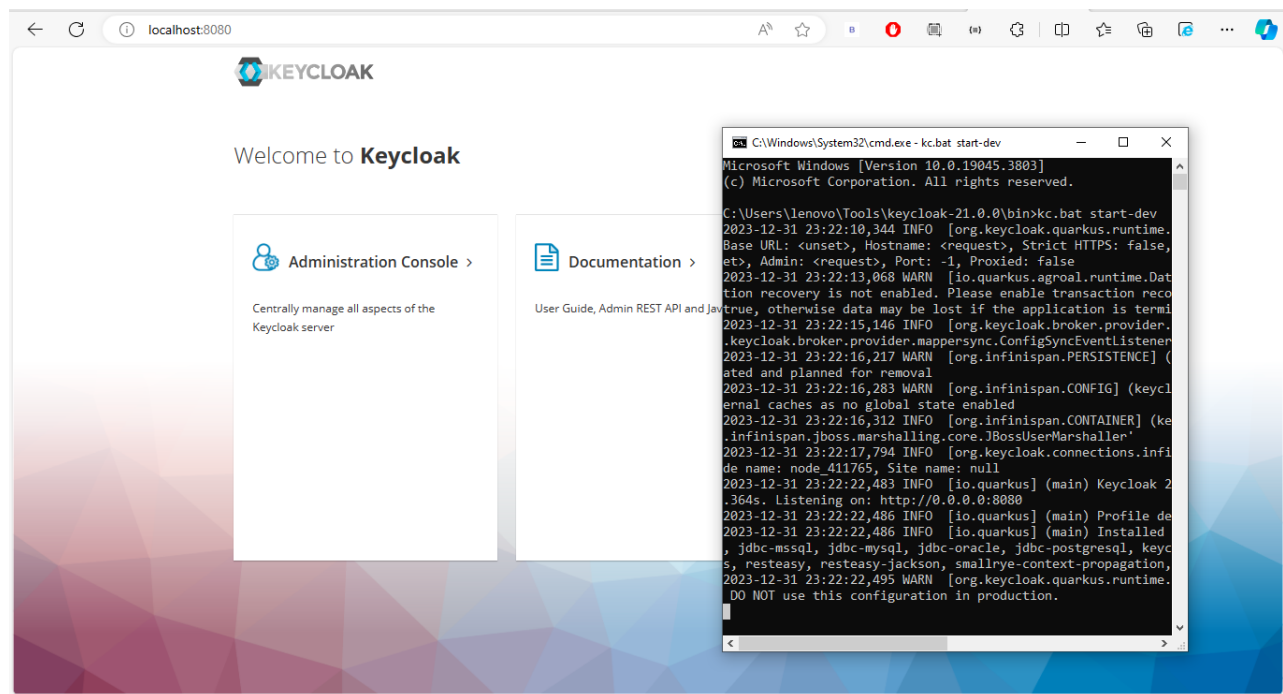


FIGURE 5 – Keycloak

Swagger Documentation : Microservice reservation-service :

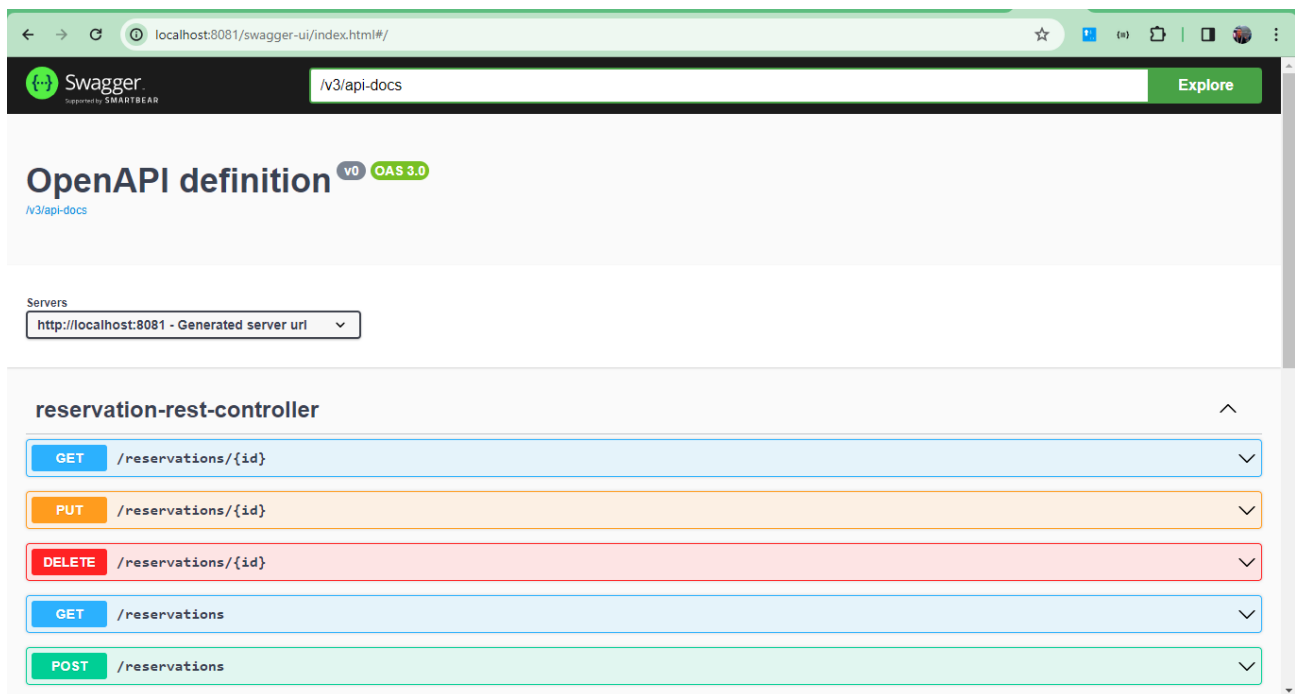


FIGURE 6 – Swagger microservice reservation-service

Swagger Documentation : Microservice ressource-service :

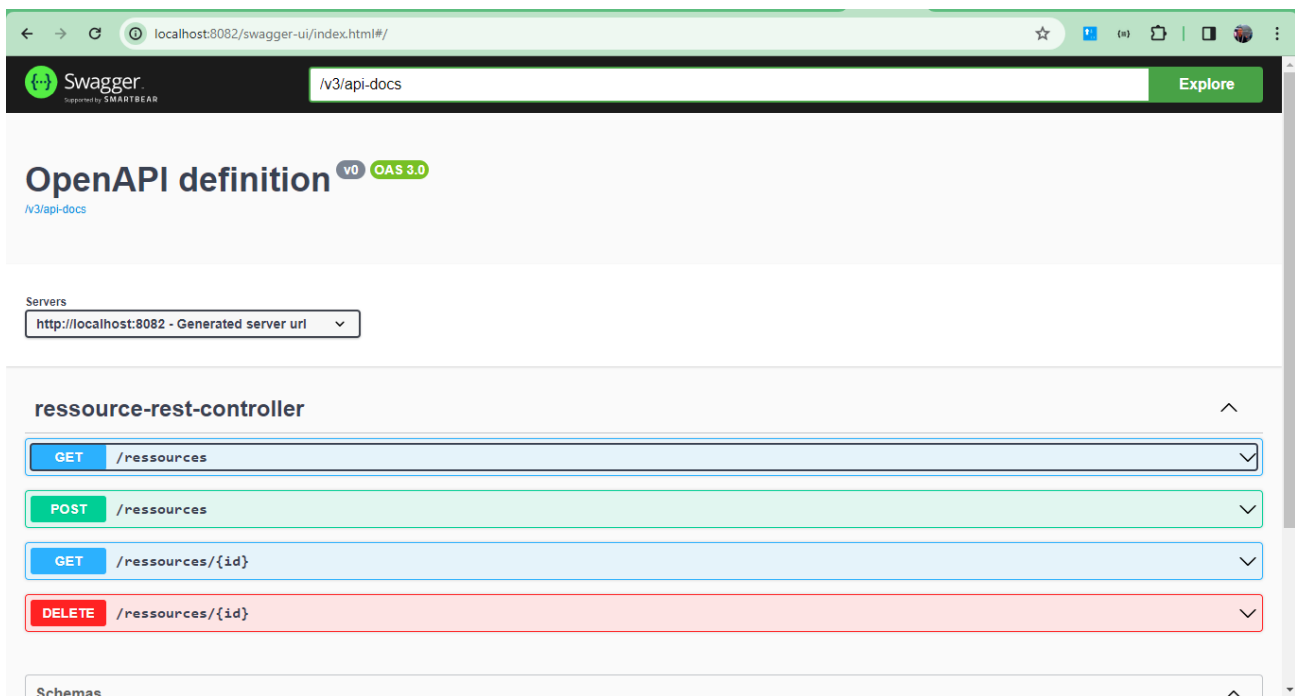


FIGURE 7 – Swagger microservice ressource-service

LES TESTS A TRAVERS POSTMAN

Envoyer un requette GET a 'http://localhost:9999/reservation-service/reservations' avec le acces-token pour avoir la listes des reservations.

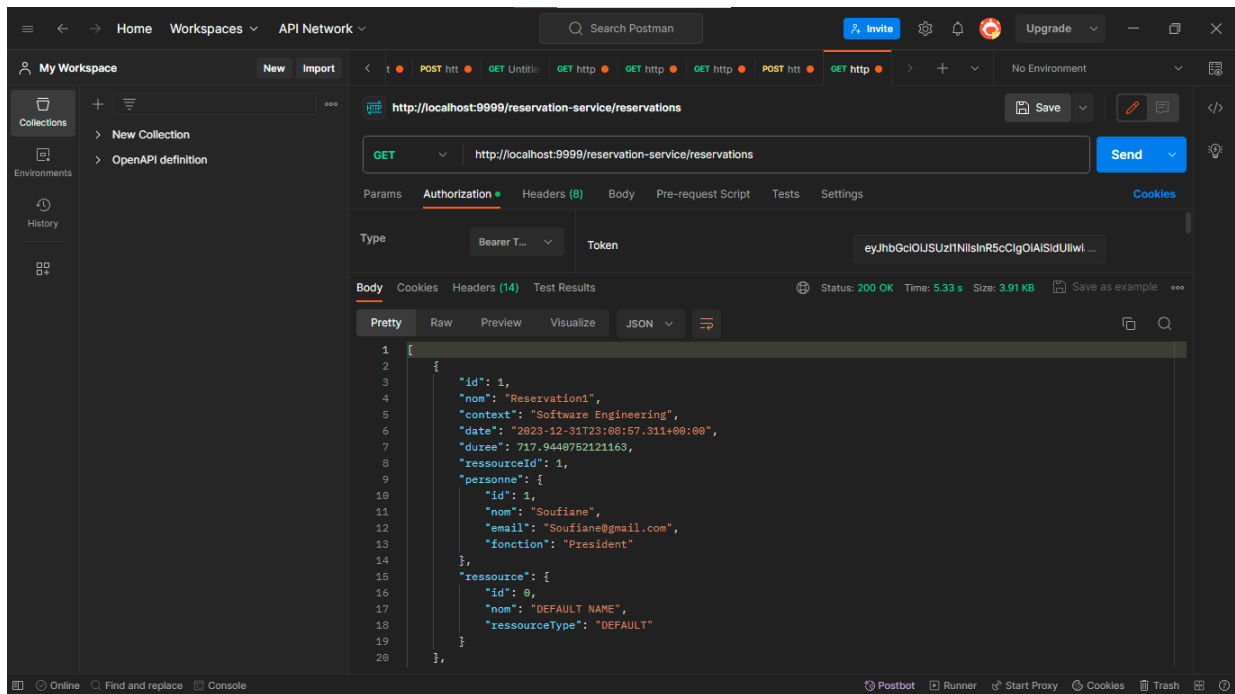


FIGURE 8 – ressource-service

SCREEN D'ÉCRANS DE VS CODE

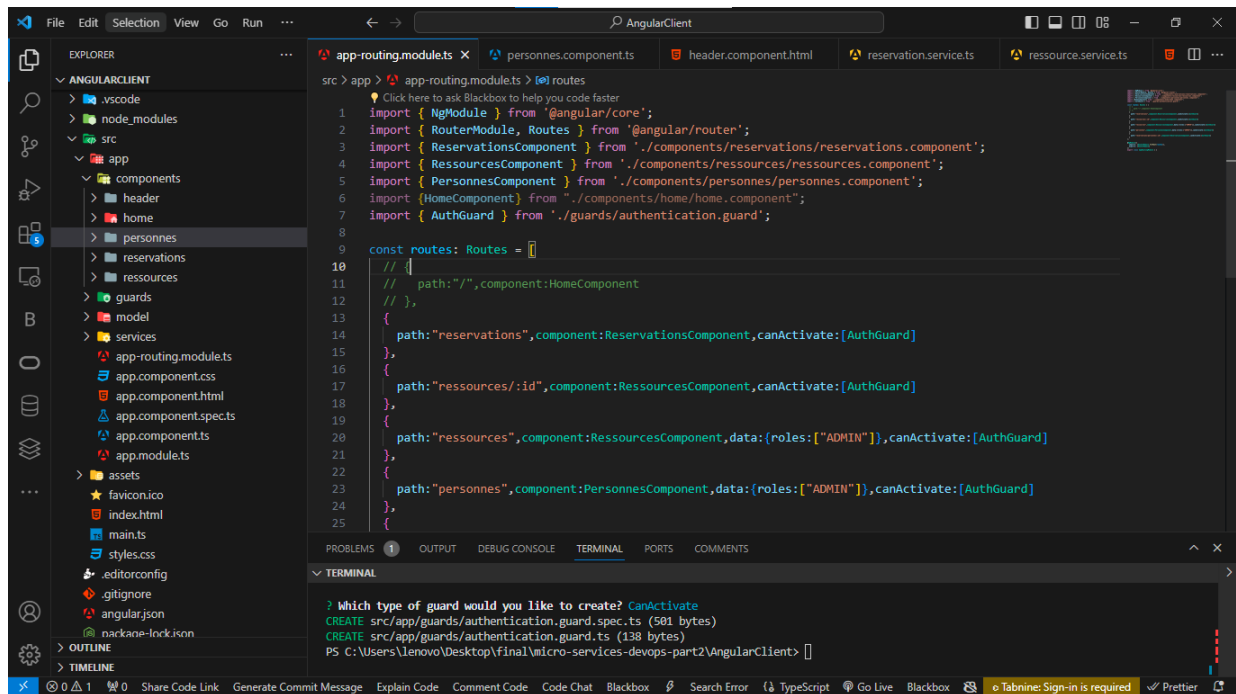


FIGURE 9 – l'architecture de project Angular

DIGRAMME DE CLASSE DE PROJET

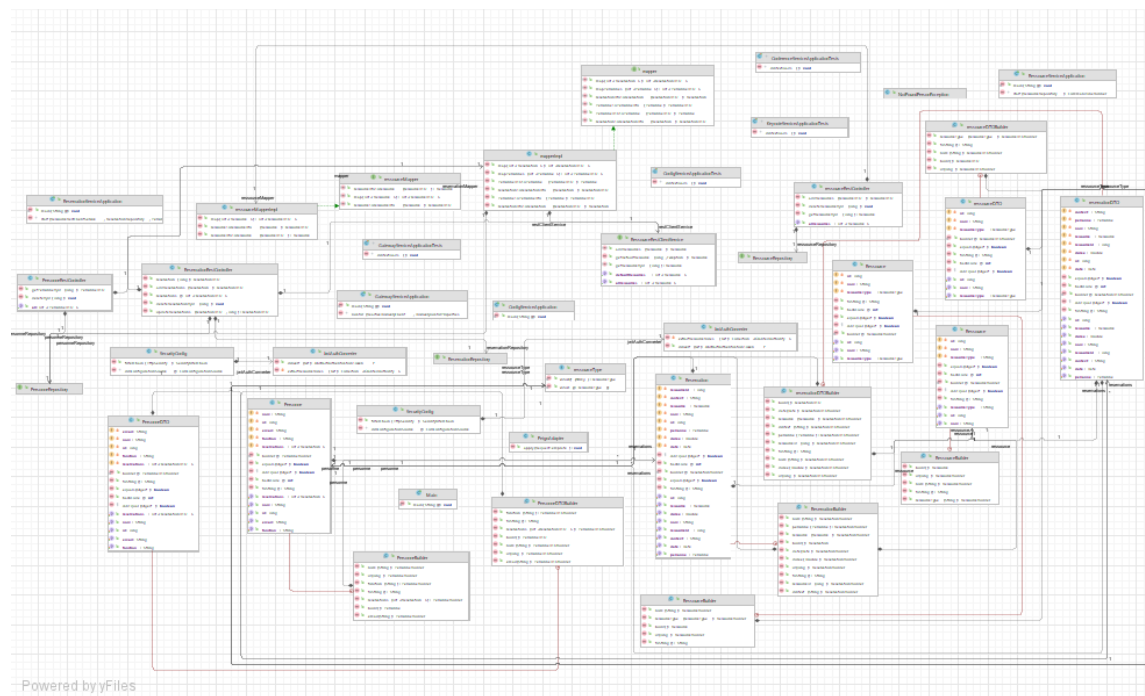


FIGURE 10 – Diagramme de classe

STRUCTURE DE MICRO-SERVICES

La structure de chaque micro-service

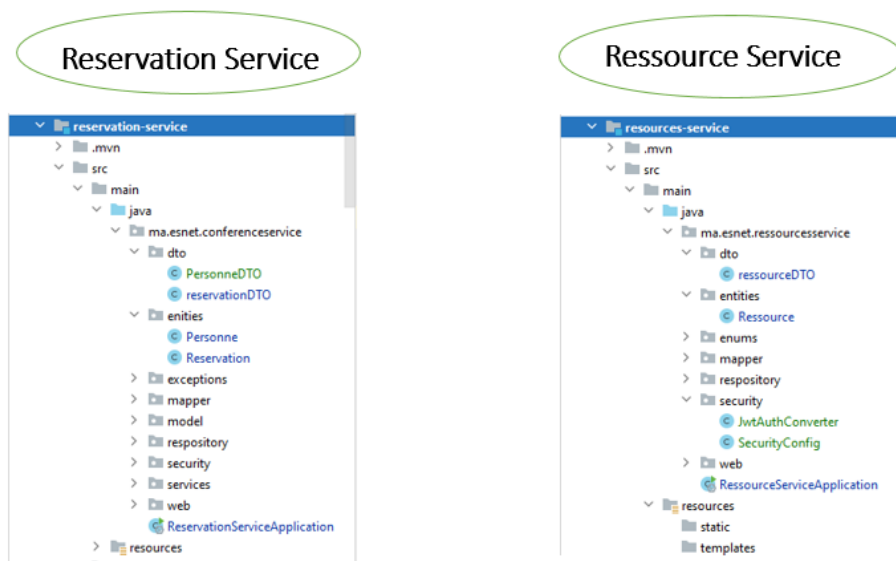


FIGURE 11 – Structure complete

Les entites et DTOs de Ressource Service

Ressource Service

Entities

```

A Soufiane
@Entity
@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class Ressource {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    @Enumerated(EnumType.STRING)
    private RessourceType ressourceType;
}

```

DTO

```

17 usages A Soufiane *
@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class ressourceDTO {
    private Long id;
    private String nom;
    private RessourceType ressourceType;
}

```

FIGURE 12 – Entities et DTOs

Les entities et DTOs de Reservation Service

Reservation Service

Entities

```

A Soufiane *
@Entity
@Data @AllArgsConstructor @NoArgsConstructor @Builder
public class Reservation {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    private String context;
    private Date date;
    private Long ressourceId;
    private Double duree;
    @ManyToOne
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private Personne personne;
    @Transient
    private Ressource ressource;
}

A Soufiane *
@Entity
@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class Personne {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String nom;
    @Email
    private String email;
    private String fonction;
    @OneToOne(mappedBy = "personne")
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Reservation> reservations;
}

```

DTO

```

@Data
@AllArgsConstructor @NoArgsConstructor @Builder
public class reservationDTO {
    private Long id;
    private String nom;
    private String context;
    private Date date;
    private Double duree;
    private Long ressourceId;
    private Personne personne;
    private Ressource ressource;
}

@Data @NoArgsConstructor @AllArgsConstructor @Builder
public class PersonneDTO {
    private Long id;
    private String nom;
    private String email;
    private String fonction;
    private List<reservationDTO> reservations;
}

```

FIGURE 13 – Entities et DTOs

OPEN FEIGN ET CIRCUIT BREAKER

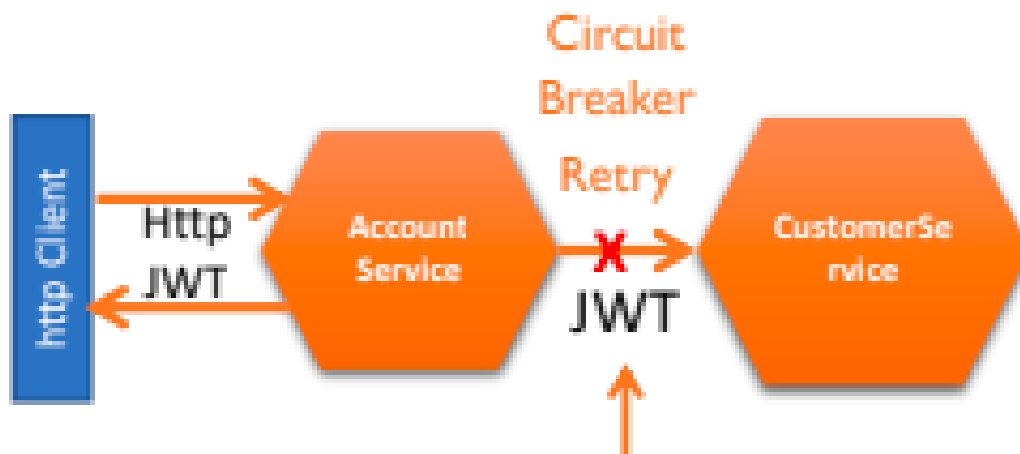


FIGURE 14 – Circuit breaker

Pour communiquer la service reservation avec ressource reservation quand elle a besoin de données de ressource d'un reservation et la gestion de tolérance en panne en cas où l'autre service est n'a pas disponible.


```

7 usages  ⚡ Soufiane *
@FeignClient(name = "ressources-service", configuration = FeignAdapter.class)

public interface RessourceRestClientService {
    2 usages  ⚡ Soufiane *
    @GetMapping("/ressources/{id}")
    @CircuitBreaker(name = "ressourcesService", fallbackMethod = "getDefaultResource")
    Ressource getResourceById(@PathVariable Long id);
    new *
    @GetMapping("/ressources")
    @Retry(name = "retryAllResources", fallbackMethod = "getDefaultResources")
    List<Ressource> getAllressources();
    1 usage new *
    @PostMapping("/ressources")
    Ressource addRessources(@RequestBody Ressource ressource);
    new *
    default Ressource getDefaultResource(Long id, Exception e){
        return Ressource.builder()
            .ressourceType("DEFAULT")
            .nom("DEFAULT NAME")
            .id(0L)
            .build();
    }
    new *
    default List<Ressource> getDefaultResources() { return List.of(); }
}

```

FIGURE 15 – Open Feign et Circuit Breaker

Configuration de la Sécurité

```
@Configuration
@EnableMethodSecurity(prePostEnabled = true)
@EnableWebSecurity
public class SecurityConfig {
    private JwtAuthConverter jwtAuthConverter;

    public SecurityConfig(JwtAuthConverter jwtAuthConverter) {
        this.jwtAuthConverter = jwtAuthConverter;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        return http
            .csrf(csrf->csrf.disable())
            .authorizeHttpRequests(ar -> ar.requestMatchers("/actuator/**",
                "/v3/api-docs/**",
                "/swagger-ui/**",
                "/swagger-ui.html",
                "health/**",
                "/h2-console/**").permitAll())
            .authorizeHttpRequests(ar->ar.anyRequest().authenticated())
            .oauth2ResourceServer(o2->o2.jwt(jwt->jwt.
                jwtAuthenticationConverter(jwtAuthConverter)))
            .headers(h->h.frameOptions(fo->fo.disable()))
            .sessionManagement(sm->sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .build();
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(Arrays.asList("*"));
        configuration.setAllowedHeaders(Arrays.asList("*"));
        configuration.setAllowedMethods(Arrays.asList("*"));
    }
}
```

```
    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", configuration);
    return source;
}
}
```

@Component

```
public class JwtAuthConverter implements Converter<Jwt, AbstractAuthenticationToken> {  
    private final JwtGrantedAuthoritiesConverter jwtGrantedAuthoritiesConverter=new  
        JwtGrantedAuthoritiesConverter();
```

@Override

```
public AbstractAuthenticationToken convert(Jwt jwt) {  
    Collection<GrantedAuthority> authorities = Stream.concat(  
        jwtGrantedAuthoritiesConverter.convert(jwt).stream(),  
        extractResourceRoles(jwt).stream()  
    ).collect(Collectors.toSet());  
    return new JwtAuthenticationToken(jwt,  
        authorities,jwt.getClaim("preferred_username"));  
}
```

```
private Collection<GrantedAuthority> extractResourceRoles(Jwt jwt) {  
    Map<String , Object> realmAccess;  
    Collection<String> roles;  
    if(jwt.getClaim("realm_access")==null){  
        return Set.of();  
    }  
    realmAccess = jwt.getClaim("realm_access");  
    roles = (Collection<String>) realmAccess.get("roles");  
    return roles.stream().map(role->new  
        SimpleGrantedAuthority(role)).collect(Collectors.toSet());  
}
```

```
}
```

Open Feign Adapter quand la reservation envoyer une resquette [GET, POST, PUT, DELETE] a ressource service et a besoin de acces token.

```
@Component
public class FeignAdapter implements RequestInterceptor {

    @Override
    public void apply(RequestTemplate requestTemplate) {
        SecurityContext context = SecurityContextHolder.getContext();
        JwtAuthenticationToken jwtAuthenticationToken = (JwtAuthenticationToken)
            context.getAuthentication();
        String tokenValue = jwtAuthenticationToken.getToken().getTokenValue();
        requestTemplate.header("Authorization","Bearer "+tokenValue);
    }
}
```

UTILISATION DE KEYCLOAK

Configuration de Keycloak

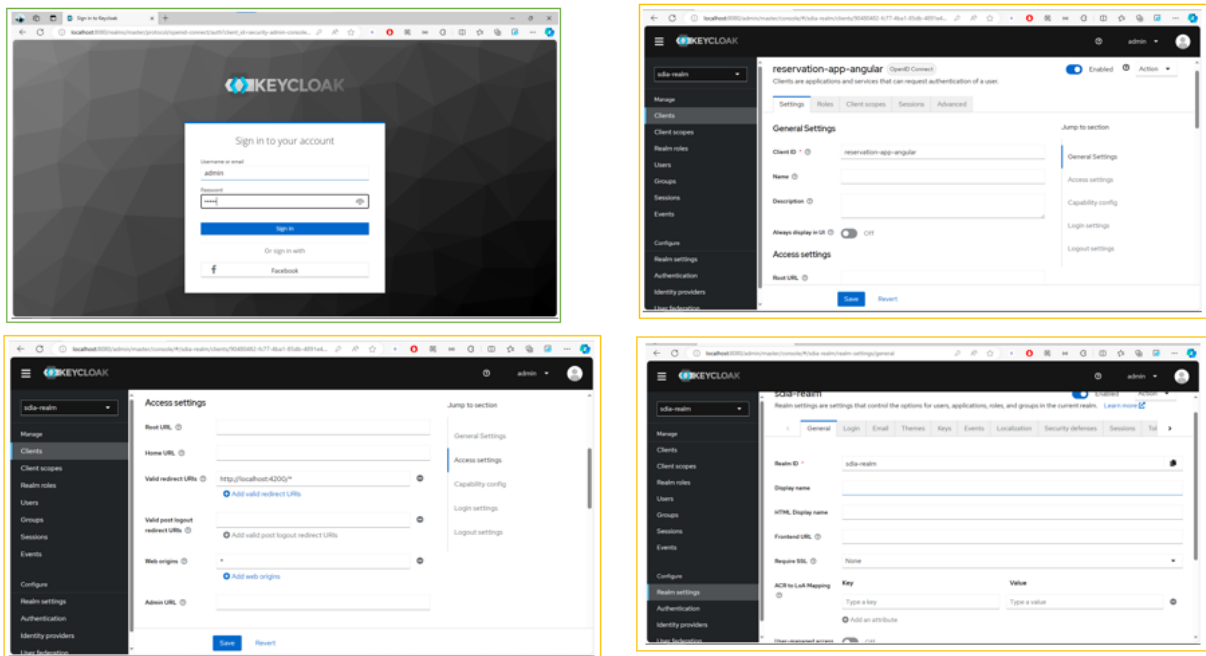


FIGURE 16 – Keycloak

Gestion des utilisateurs dans Keycloak

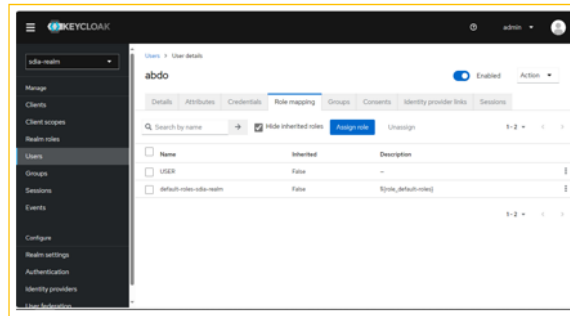
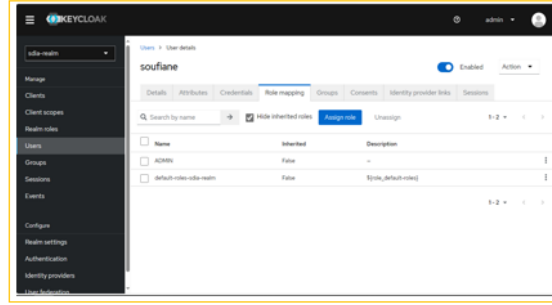
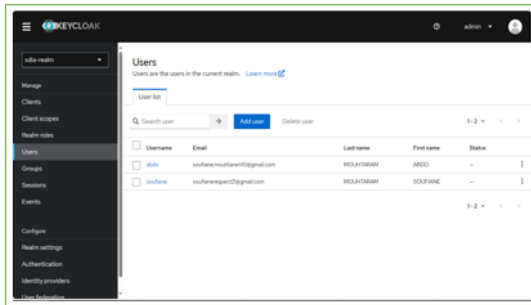


FIGURE 17 – Gestion des utilisateurs

LapageAccueildenotreFrontendAngular

Faire Authentification pour permettre a acces a les services seures.

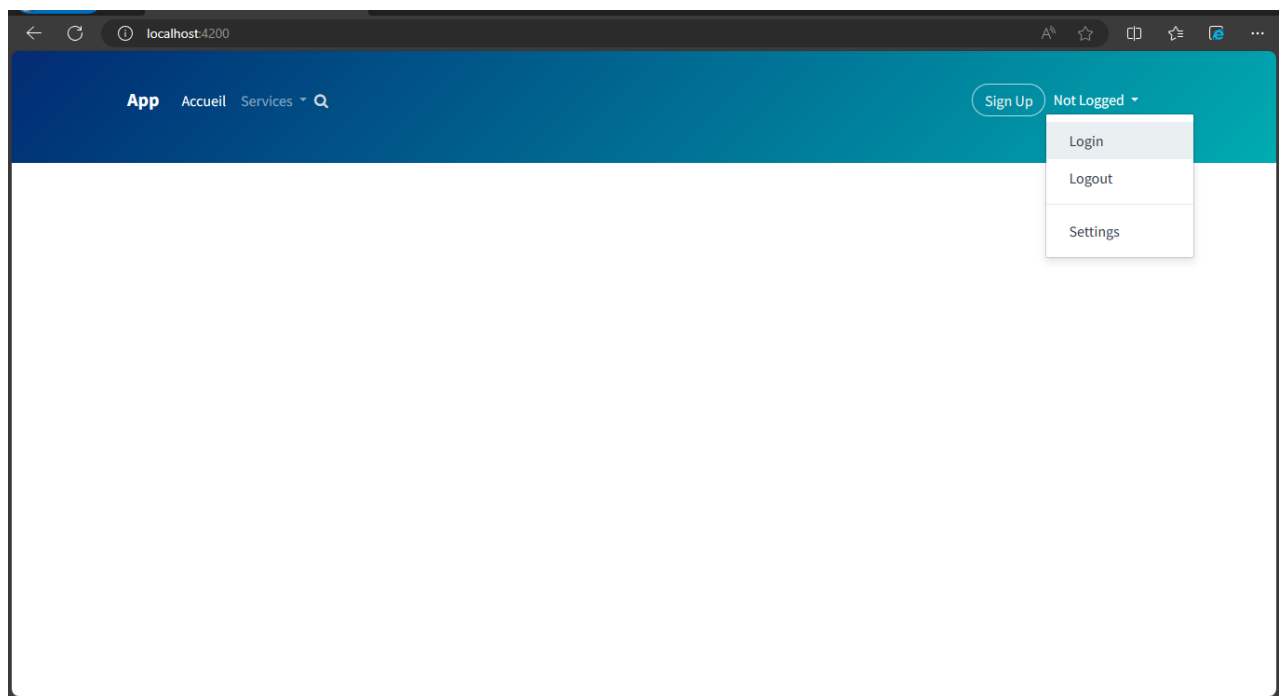


FIGURE 18 – Login

Après la clique sur Login il va vous redireger vers Keycloak pour faire l'authentification :

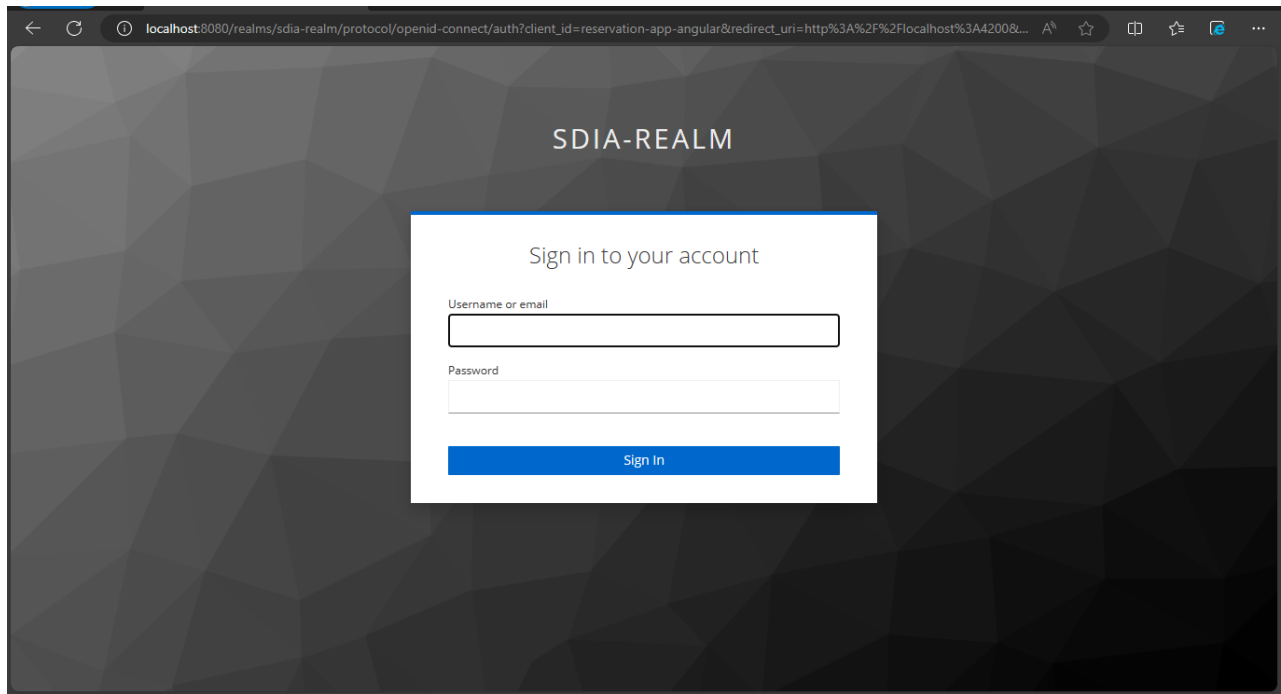


FIGURE 19 – Login User in keycloak

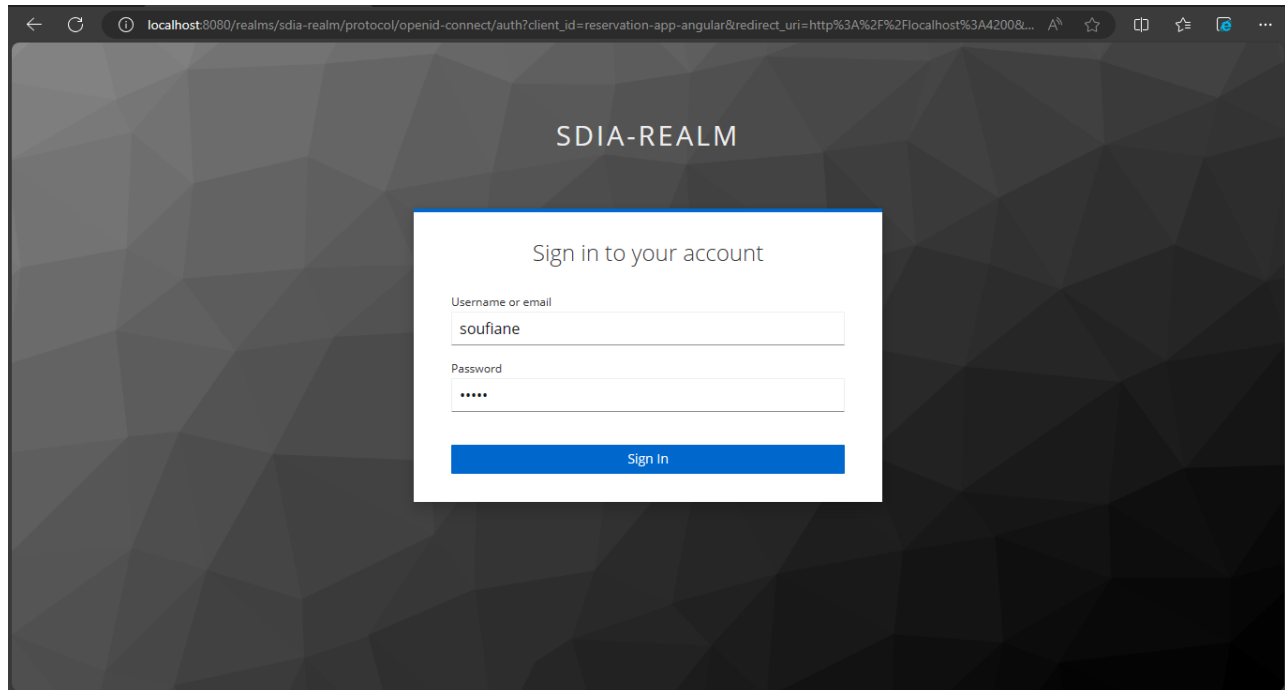


FIGURE 20 – Utilisateur a Role ADMIN

CRUD Angular La page de Utilisateur a role ADMIN il peut faire :

- La suppression.
- La modification.
- L'ajout.

#	Nom	Context	Date	Duree	Personne	Ressource
1	Reservation1	Software Engineering	2024-01-01T03:07:07.568+00:00	641.84	Soufiane	Voir
2	Reservation2	AI	2024-01-01T03:07:07.618+00:00	181.75	Soufiane	Voir
3	Reservation3	Software Engineering	2024-01-01T03:07:07.621+00:00	730.93	Soufiane	Voir
4	Reservation1	Software Engineering	2024-01-01T03:07:07.626+00:00	792.75	Mohamed	Voir
5	Reservation2	Software Engineering	2024-01-01T03:07:07.628+00:00	296.58	Mohamed	Voir

FIGURE 21 – User a Role ADMIN

Modification d' une reservation

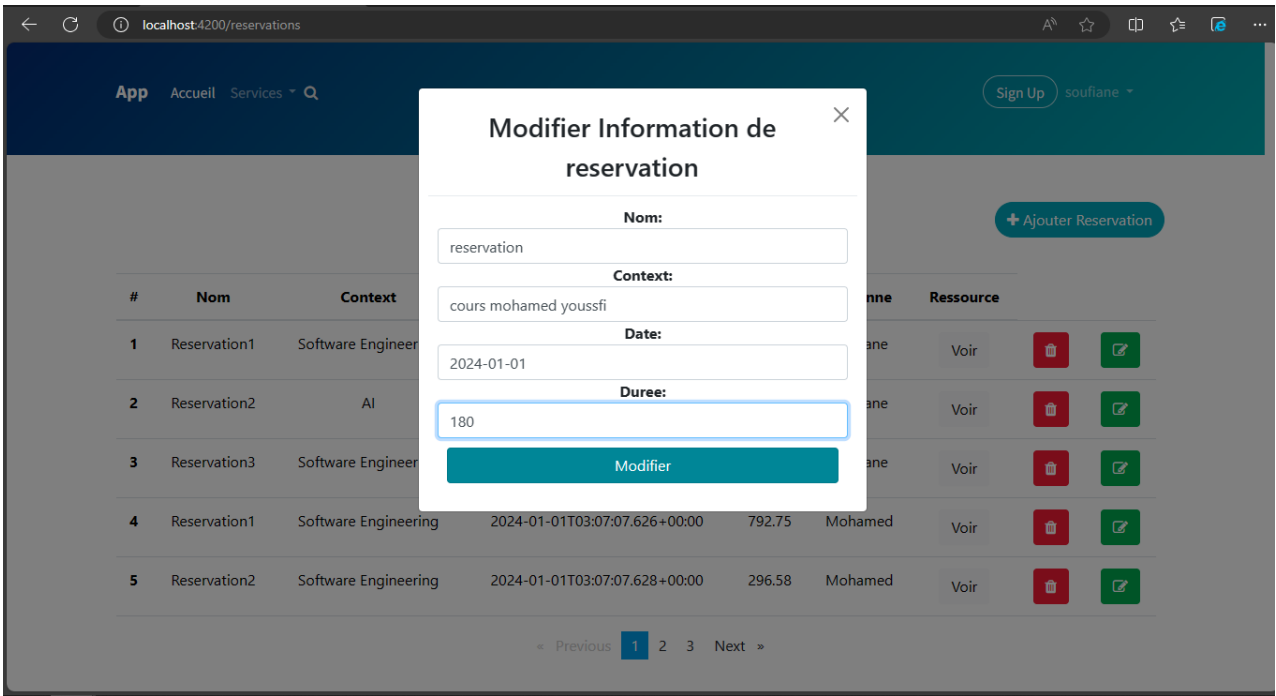


FIGURE 22 – Modification de reservation

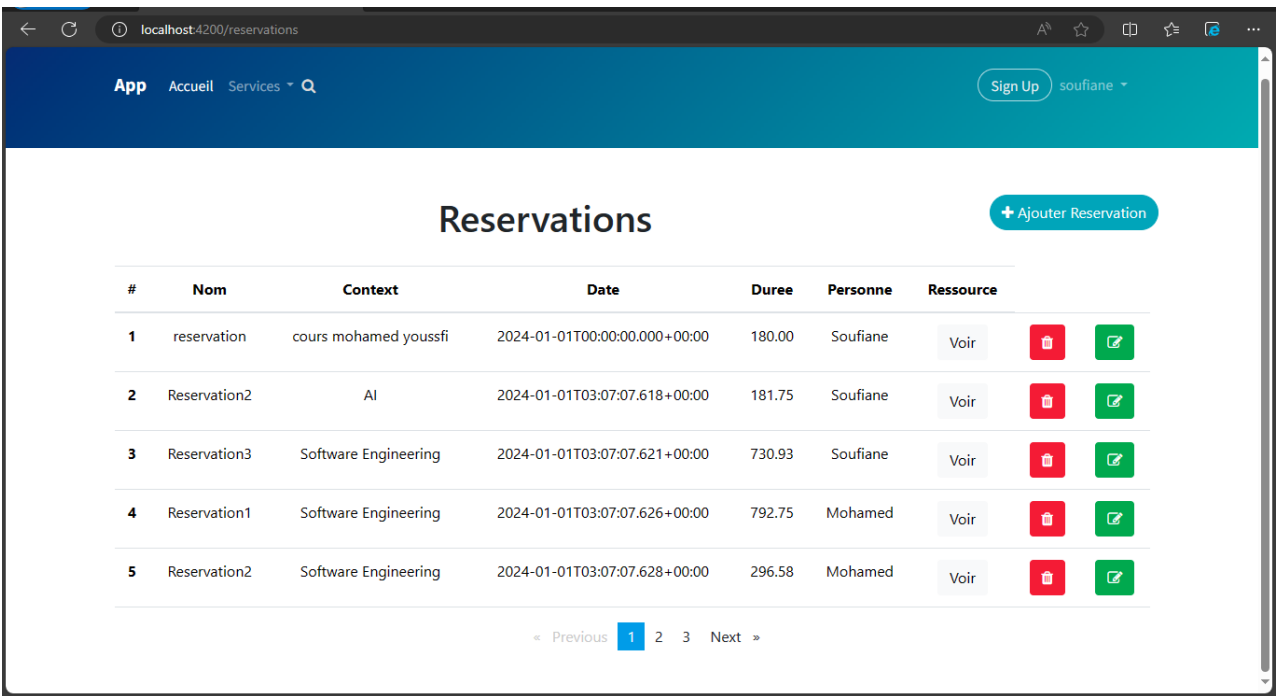


FIGURE 23 – Modification de reservation

Après la modification voir la reservation 1

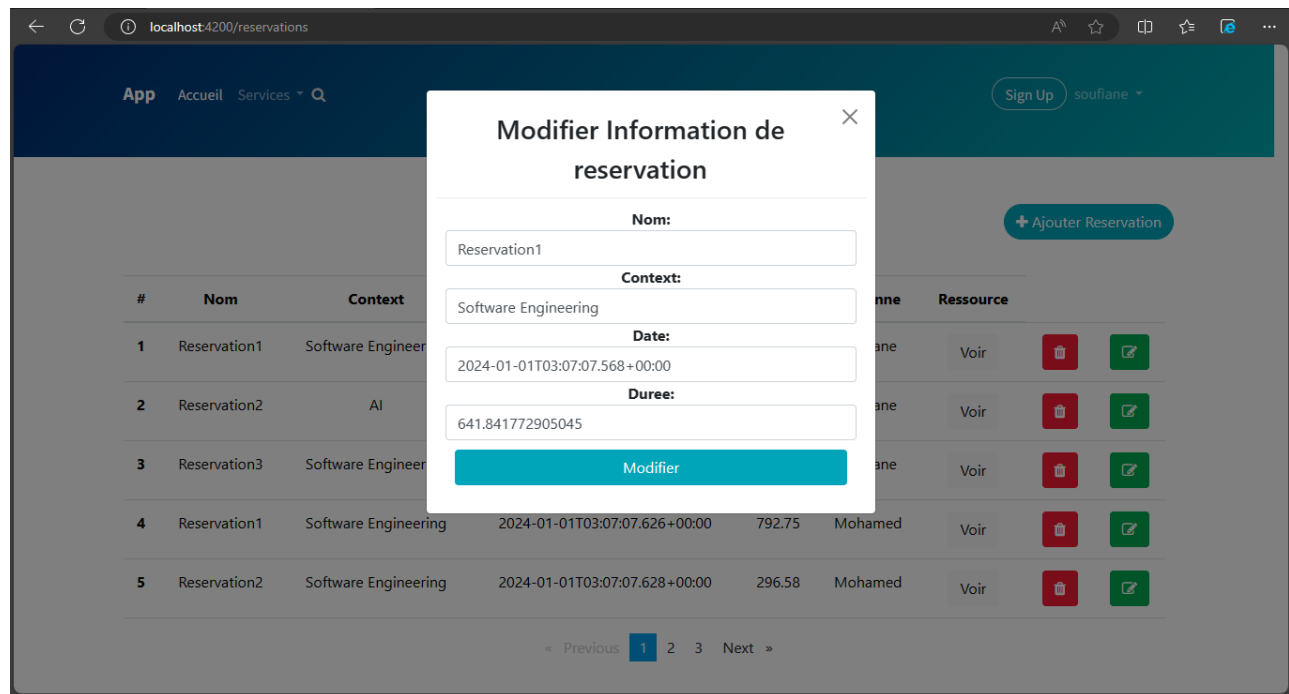


FIGURE 24 – Modification de reservation

Ajout d'une reservation

Ajout d'une reservation avec les donnees d'une ressource et de personne tous ils vont ajouté

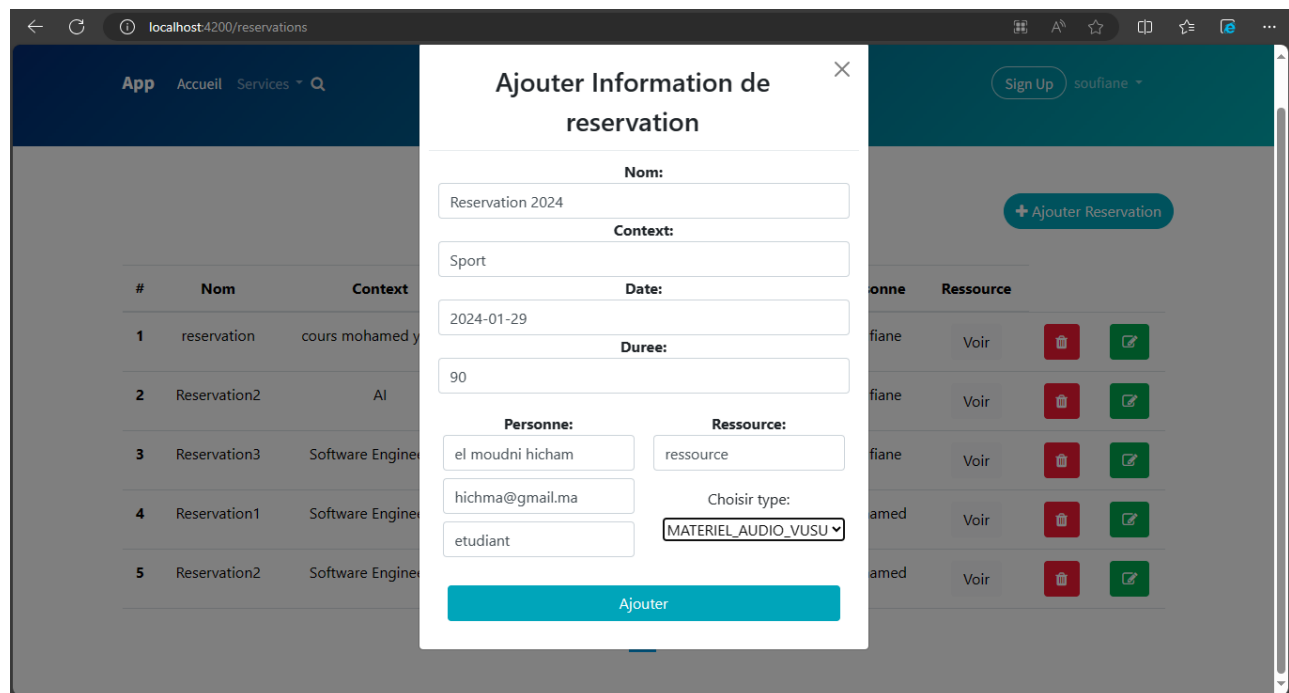


FIGURE 25 – Ajout d'une reservation

Après l'ajout d'une reservation

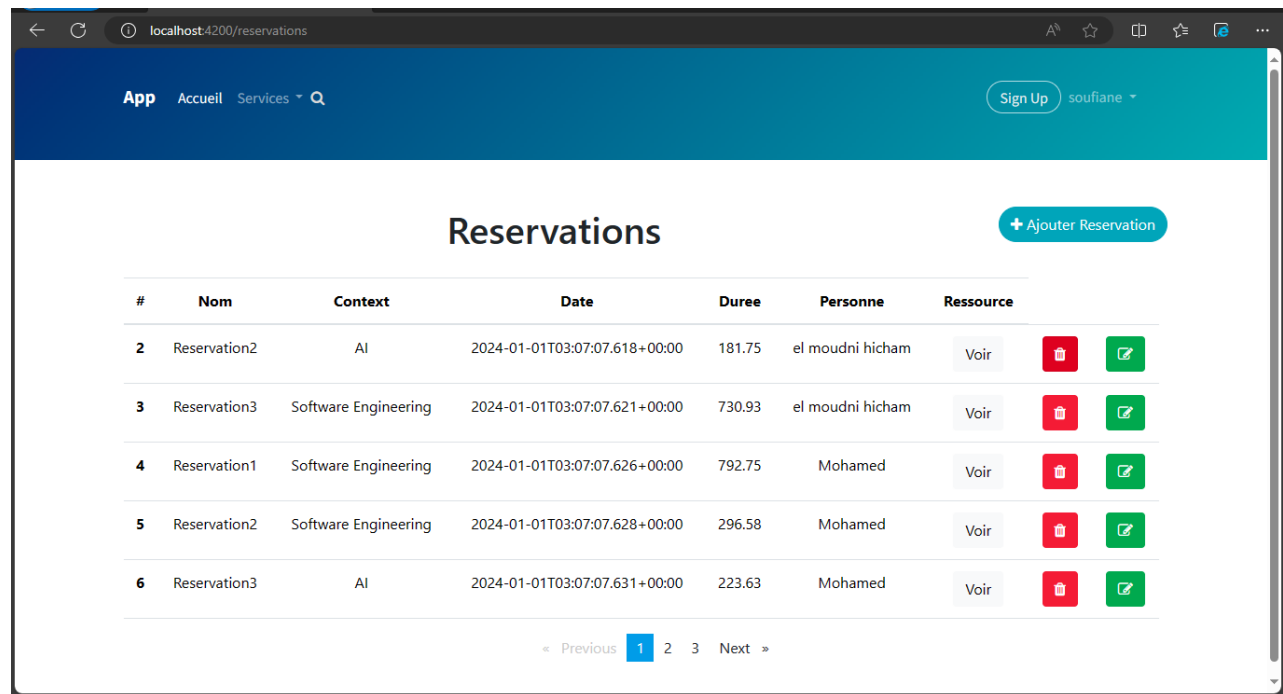


FIGURE 26 – Ajout d'une reservation

La suppression d'une reservation

la suppression de la reservation qu'a id == 1

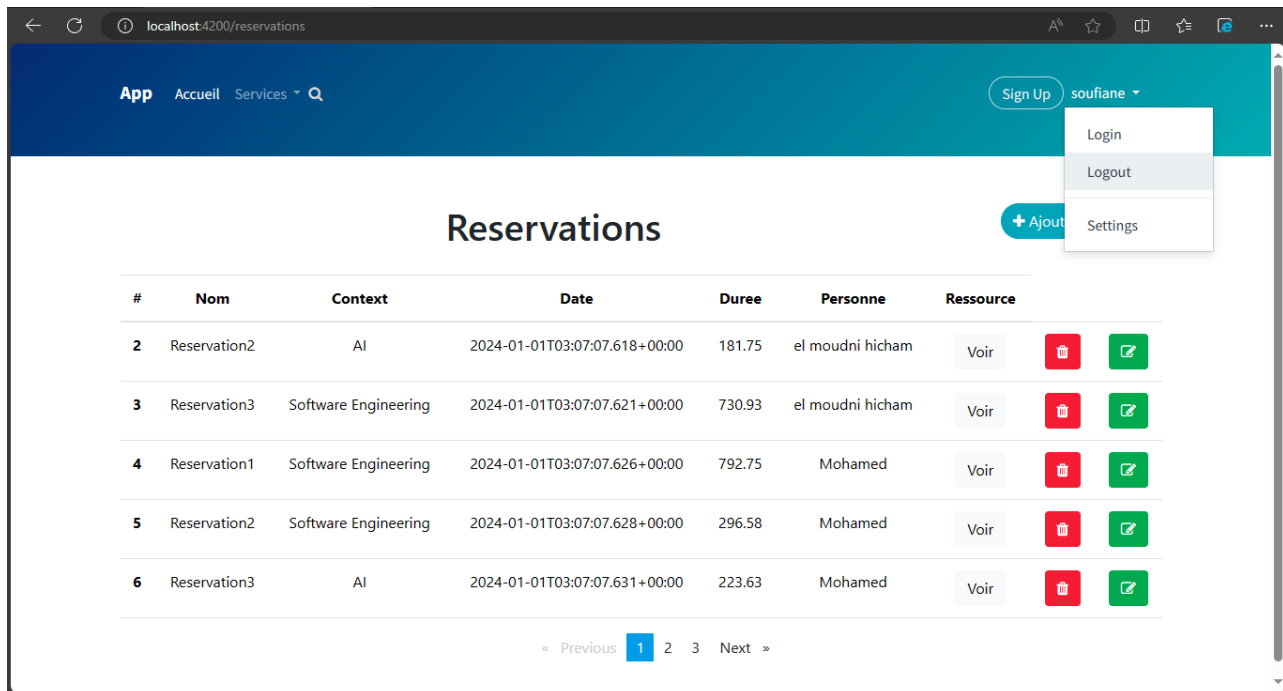


FIGURE 27 – suppression d'une reservation

Authentication avec utilisateur a role USER

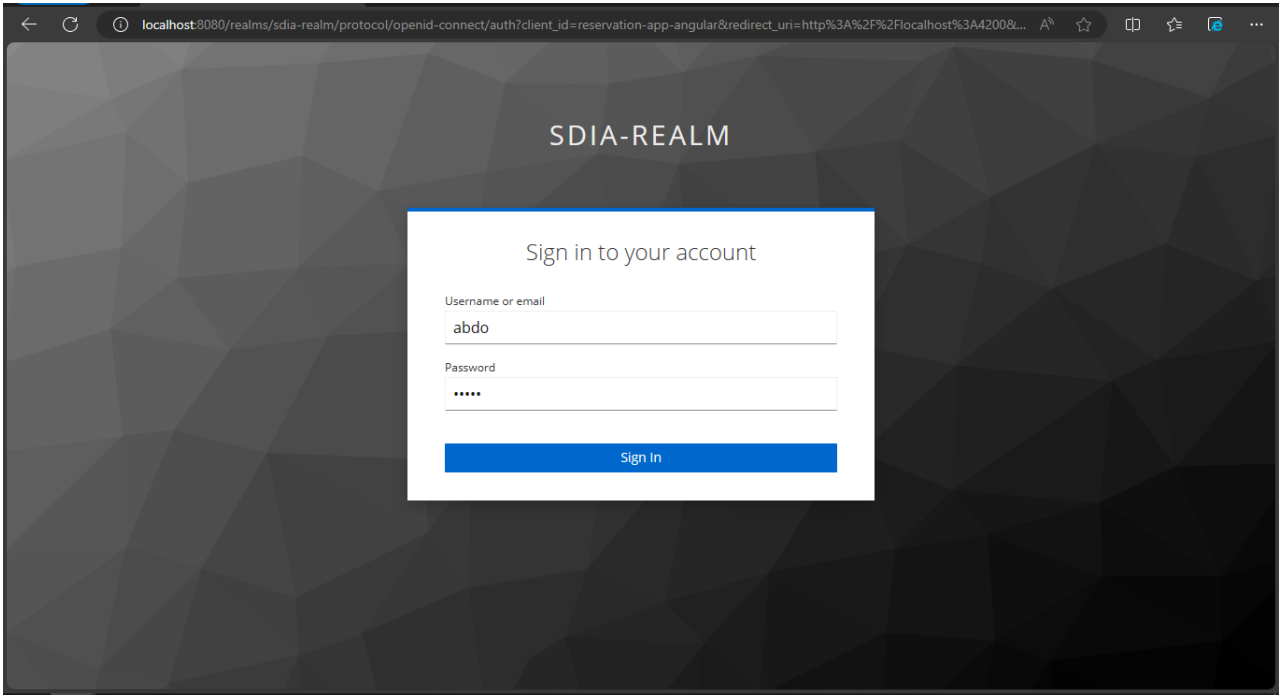


FIGURE 28 – Utilisateur a Role USER

Reservation Page

App Accueil Services

Sign Up abdo

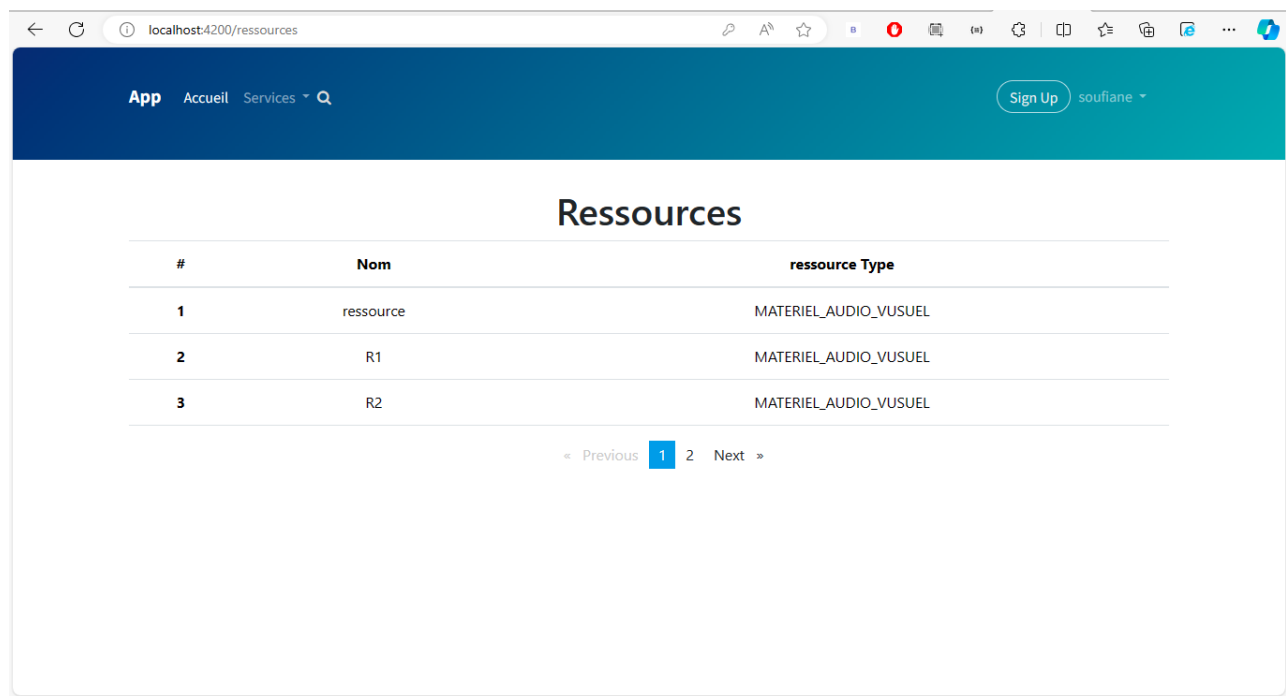
Reservations

#	Nom	Context	Date	Duree	Personne	Ressource
2	Reservation2	AI	2024-01-01T03:07:07.618+00:00	181.75	el moudni hicham	Voir
3	Reservation3	Software Engineering	2024-01-01T03:07:07.621+00:00	730.93	el moudni hicham	Voir
4	Reservation1	Software Engineering	2024-01-01T03:07:07.626+00:00	792.75	Mohamed	Voir
5	Reservation2	Software Engineering	2024-01-01T03:07:07.628+00:00	296.58	Mohamed	Voir
6	Reservation3	AI	2024-01-01T03:07:07.631+00:00	223.63	Mohamed	Voir

< Previous 1 2 3 Next >

FIGURE 29 – Utilisateur a Role USER

Ressource Page



#	Nom	ressource Type
1	ressource	MATERIEL_AUDIO_VUSUEL
2	R1	MATERIEL_AUDIO_VUSUEL
3	R2	MATERIEL_AUDIO_VUSUEL

« Previous 1 2 Next »

FIGURE 30 – Utilisateur a Role USER