المدرسة العليا لأساتذة التعليم
التقني المحمدية
جامعة الحسن الثاني بالدار البيضاء

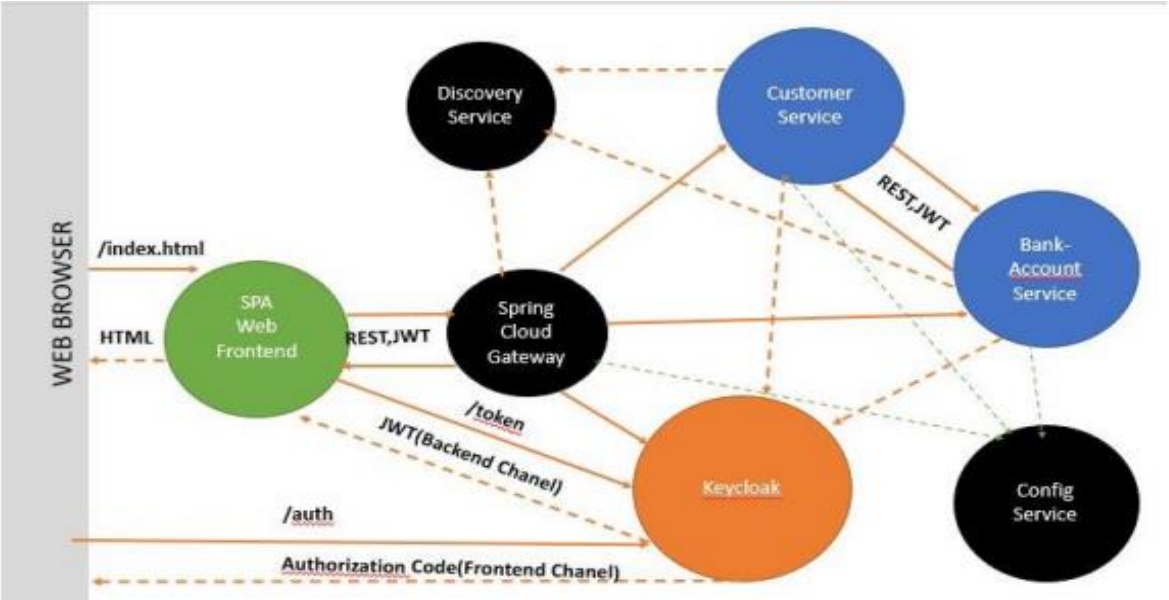
Comte Rendu

Examen Blanc Systèmes Distribués

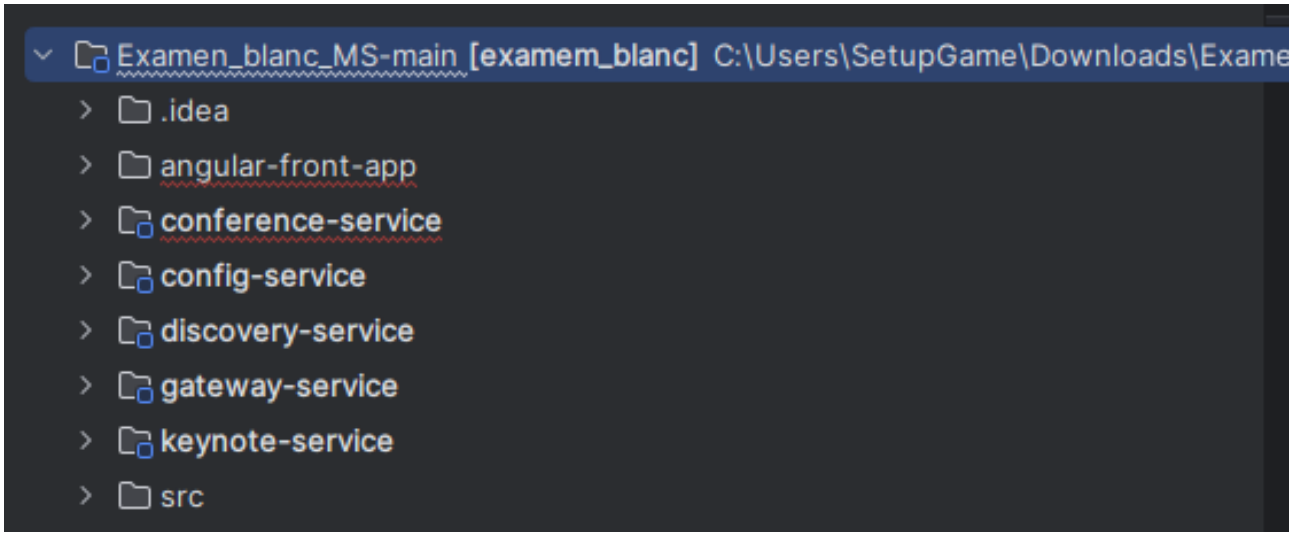
Sdia2

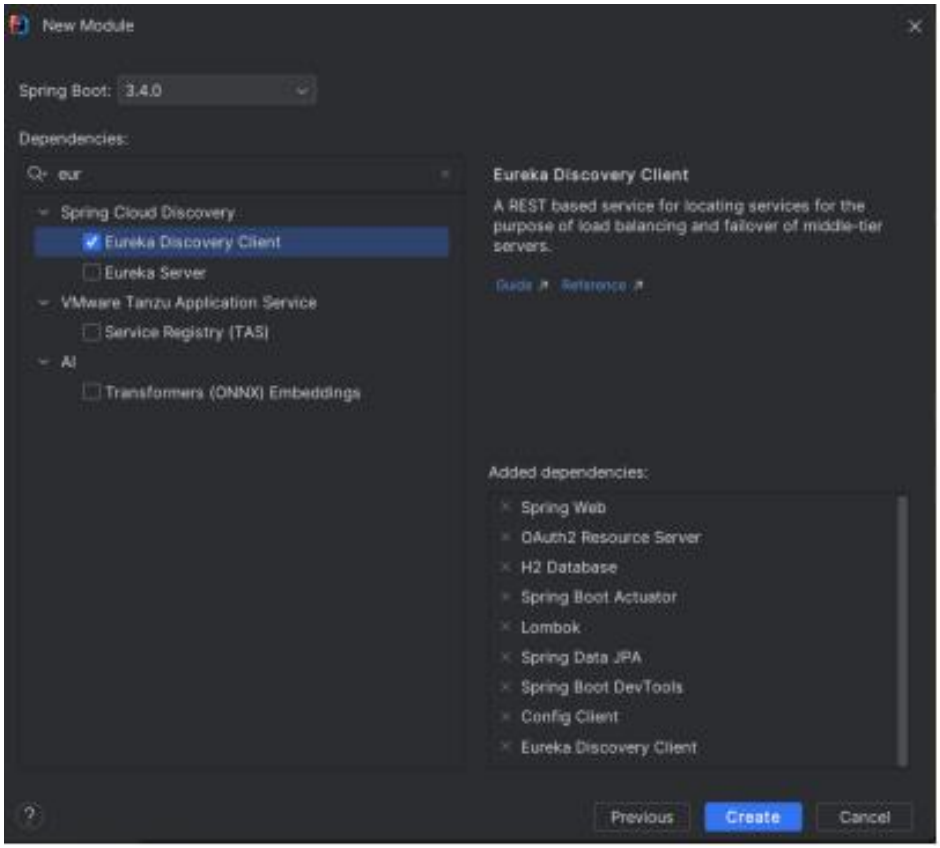
Nom : Elhakki Ossama

1. **Architecture technique du projet**



2. **Projet Maven incluant les micro-services suivants : keynote-service, conference-service,gateway-service, discovery-service, config-service et angular-front-app**





3. **Développer et tester les micro-services discovery-service et gateway-service et config-service**
    a. **Discovery-service**

```java
package elhakki.ossama.discoveryservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServiceApplication {

    public static void main(String[] args) { SpringApplication.run(DiscoveryServiceApplication.class, args); }

}
```

```yaml
spring.application.name=discovery-service

server.port=8761
eureka.client.fetch-registry=false
eureka.client.register-with-eureka=false
```

b. gateway-service

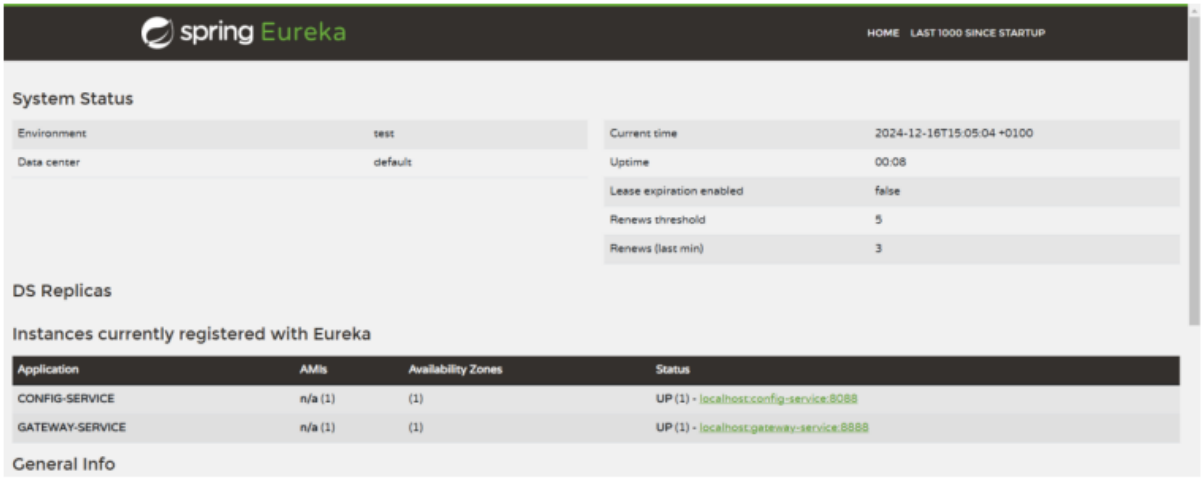
```java
package elhakki.ossama.gatewayservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.ReactiveDiscoveryClient;
import org.springframework.context.annotation.Bean;
import org.springframework.cloud.gateway.discovery.DiscoveryClientRouteDefinitionLocator;
import org.springframework.cloud.gateway.discovery.DiscoveryLocatorProperties;


@SpringBootApplication
public class GatewayServiceApplication {

    public static void main(String[] args) { SpringApplication.run(GatewayServiceApplication.class, args); }

    @Bean
    DiscoveryClientRouteDefinitionLocator definitionLocator(
            ReactiveDiscoveryClient rdc,
            DiscoveryLocatorProperties properties){
        return new DiscoveryClientRouteDefinitionLocator(rdc, properties);
    }
}
```

```yaml
spring.application.name=gateway-service

server.port=8888
spring.cloud.discovery.enabled=true
```

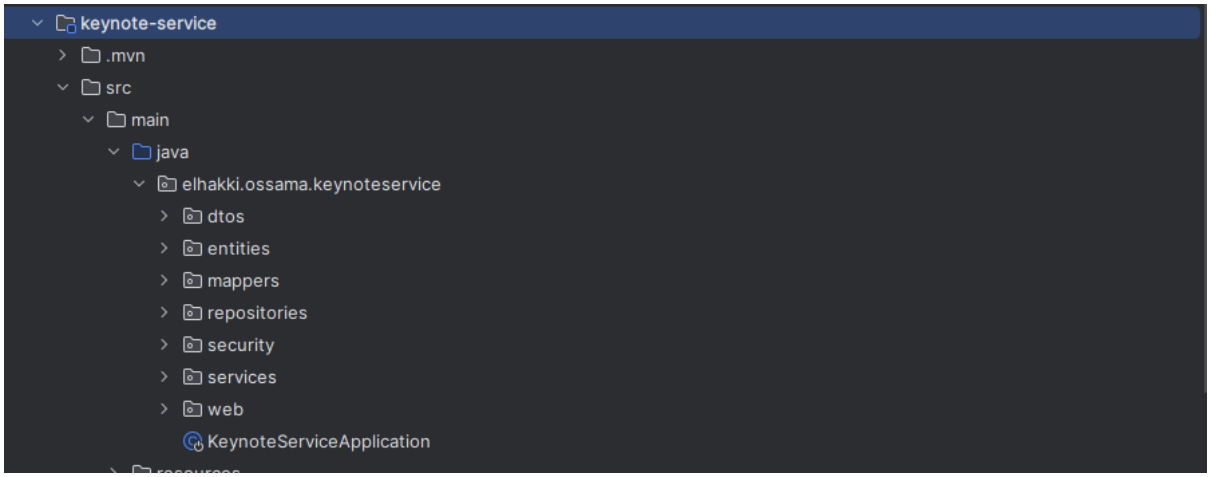
```yaml
spring:
  cloud:
    gateway:
      globalcors:
        corsConfigurations:
          '[/**]':
            allowedOrigins: "http://localhost:4200"
            allowedHeaders: "*"
            allowedMethods:
              - GET
              - POST
              - PUT
              - DELETE
```

c. config-service

```java
package elhakki.ossama.configservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
public class ConfigServiceApplication {

    public static void main(String[] args) { SpringApplication.run(ConfigServiceApplication.class, args); }

}
```



4. **Développer et tester le micro-service Keynote-service (Entities, DAO, service, DTO, Mapper,RestController)**



Entities :

```java
package elhakki.ossama.keynoteservice.entities;


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

13 usages
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Getter @Setter @Builder @ToString
public class Keynote {
    @Id
    private String id;
    private String nom;
    private String prenom;
    private String email;
    private String fonction;
}
```

Services:

```java
package elhakki.ossama.keynoteservice.services;

import ...

4 usages  1 implementation
public interface KeynoteService {
    1 usage  1 implementation
    List<KeynoteDTO> getAllKeynotes();
    2 usages  1 implementation
    Optional<KeynoteDTO> getKeynoteById(String id);
    1 usage  1 implementation
    KeynoteDTO createKeynote(KeynoteDTO keynoteDTO);
    1 usage  1 implementation
    void deleteKeynote(String id);
    1 usage  1 implementation
    KeynoteDTO updateKeynote(KeynoteDTO keynoteDTO);
}
```

```java
@Service
@Transactional
public class KeynoteServiceImpl implements KeynoteService{

    @Autowired
    private KeynoteRepository keynoteRepository;
    1 usage
    @Override
    public List<KeynoteDTO> getAllKeynotes() {
        List<Keynote> keynotes = keynoteRepository.findAll();
        return keynotes.stream() Stream<Keynote>
                .map(KeynoteMapper.INSTANCE::keynoteToKeynoteDTO) Stream<KeynoteDTO>
                .toList();
    }

    2 usages
    @Override
    public Optional<KeynoteDTO> getKeynoteById(String id) {
        return keynoteRepository.findById(id)
                .map(KeynoteMapper.INSTANCE::keynoteToKeynoteDTO);
    }

    1 usage
    @Override
    public KeynoteDTO createKeynote(KeynoteDTO keynoteDTO) {
        Keynote keynote = KeynoteMapper.INSTANCE.keynoteDTOToKeynote(keynoteDTO);
        Keynote savedKeynote = keynoteRepository.save(keynote);
```

Dto:

```java
package elhakki.ossama.keynoteservice.dtos;

import lombok.Data;

26 usages
@Data
public class KeynoteDTO {
    private String id;
    private String nom;
    private String prenom;
    private String email;
    private String fonction;
}
```

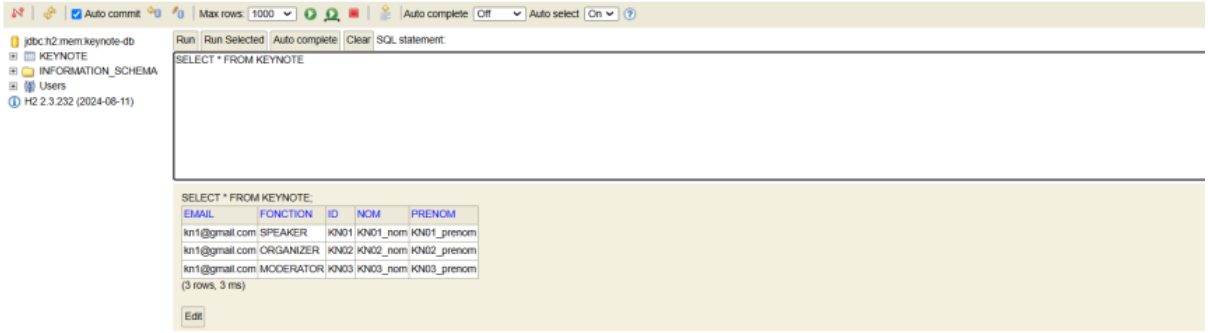
RestController:

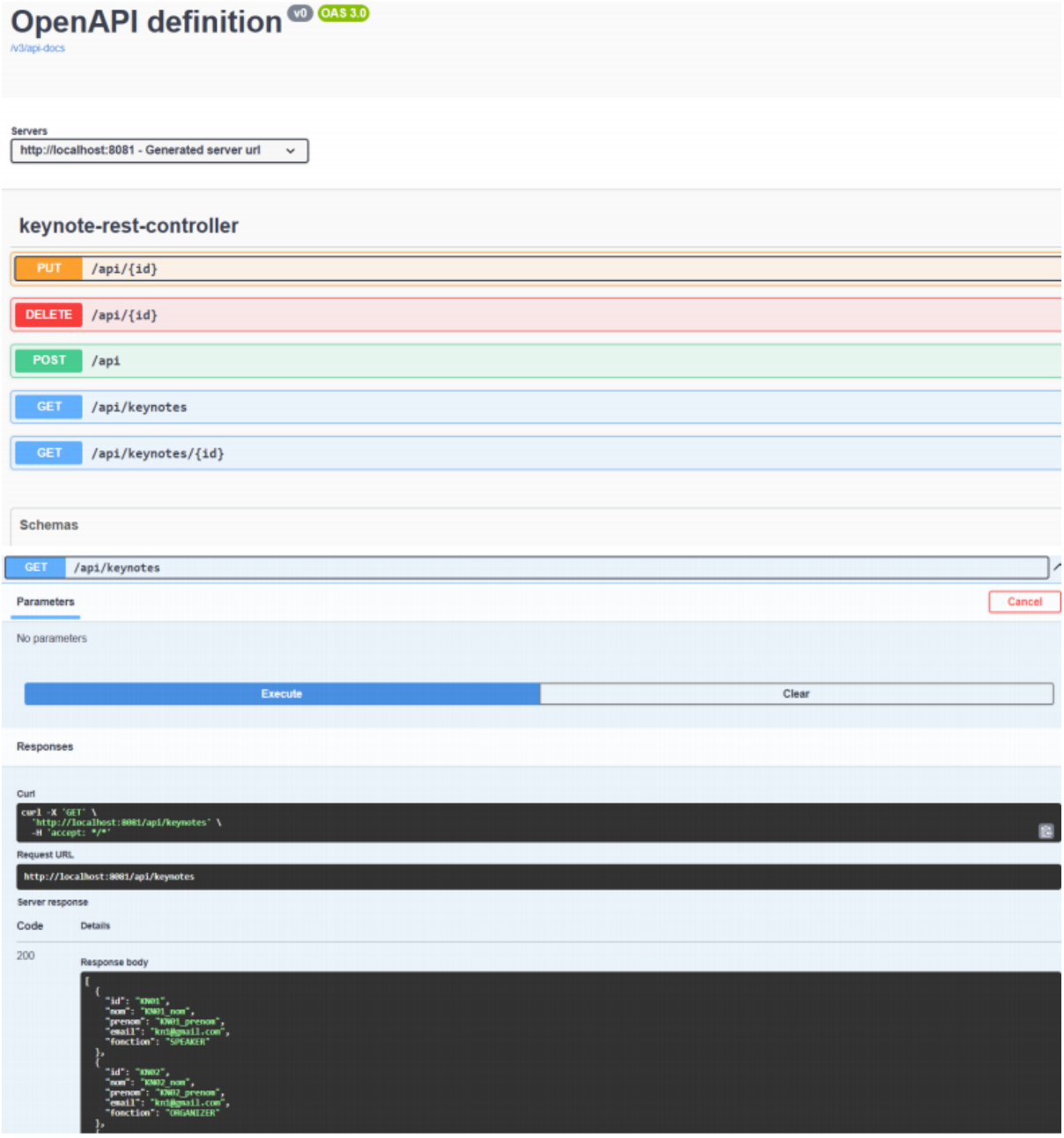
```java
package elhakki.ossama.keynoteservice.web;

import ...

@RestController
@RequestMapping("/api")
public class KeynoteRestController {

    7 usages
    private KeynoteService keynoteService;

    public KeynoteRestController(KeynoteService keynoteService) { this.keynoteService = keynoteSer

    @GetMapping("/keynotes")
    public List<KeynoteDTO> keynoteList() { return keynoteService.getAllKeynotes(); }

    @GetMapping("/keynotes/{id}")
    public KeynoteDTO keynoteById(@PathVariable String id) { return keynoteService.getKeynoteById(

    @PostMapping
    public KeynoteDTO createKeynote(@RequestBody KeynoteDTO keynoteDTO) {
        KeynoteDTO savedKeynote = keynoteService.createKeynote(keynoteDTO);
        return savedKeynote;
    }

    @PutMapping("/{id}")
    public KeynoteDTO updateKeynote(@PathVariable String id, @RequestBody KeynoteDTO keynoteDTO) {
        if (!keynoteService.getKeynoteById(id).isPresent()) {
```

App propreties :

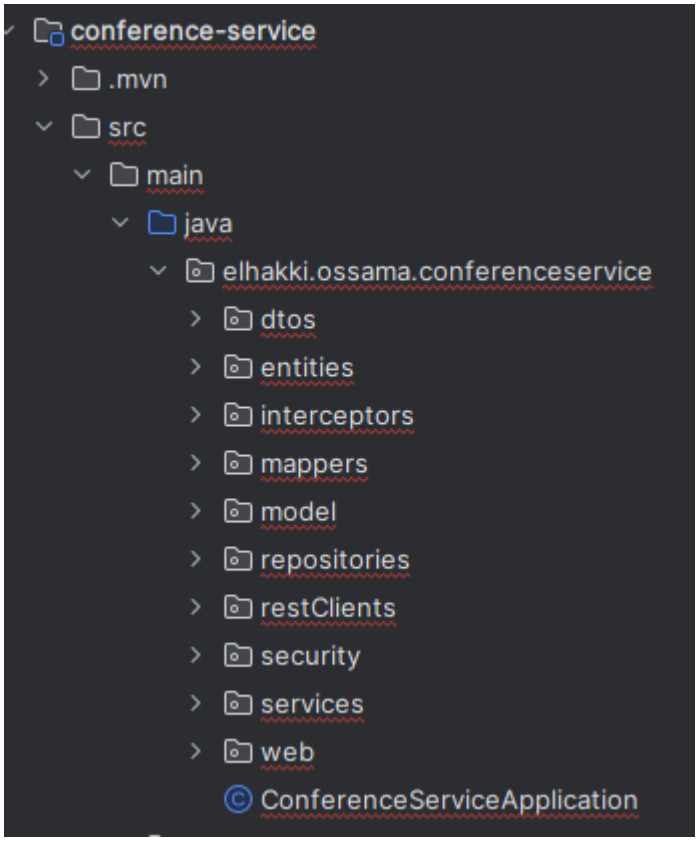
```
spring.application.name=keynote-service

server.port=8081
spring.datasource.url=jdbc:h2:mem:keynote-db
spring.h2.console.enabled=true

spring.cloud.config.enabled=false
spring.cloud.discovery.enabled=false

spring.security.oauth2.resourceserver.jwt.issuer-uri=http://localhost:8080/realms/exam-realm
spring.security.oauth2.resourceserver.jwt.jwk-set-uri=http://localhost:8080/realms/exam-realm/prot
```

Test :





5. **Développer et tester le micro-service conférence-service (Entities, DAO, service, DTO, Mapper,RestController, Client Rest Open Feign)**



Entities:

```java
@Entity
@NoArgsConstructor
@AllArgsConstructor
@Getter @Setter @Builder @ToString
public class Conference {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String titre;
    private String type;
    private LocalDate date;
    private int duree;
    private int nombreInscrits;
    private double score;

    @Transient
    private List<Keynote> keynotes;
}
```

Mappers:

```java
@Mapper
public interface ConferenceMapper {
    4 usages
    ConferenceMapper INSTANCE = Mappers.getMapper(ConferenceMapper.class);
    3 usages  1 implementation
    ConferenceDTO conferenceToConferenceDTO(Conference conference);
    1 usage  1 implementation
    Conference conferenceDTOToConference(ConferenceDTO conferenceDTO);
}
```

DTO:

```
public class ConferenceDTO {
    no usages
    private Long id;
    no usages
    private String titre;
    no usages
    private String type;
    no usages
    private LocalDate date;
    no usages
    private int duree;
    no usages
    private int nombreInscrits;
    no usages
    private double score;
    no usages
    private List<Keynote> keynotes;
}
```

Model:

```
@NoArgsConstructor
@AllArgsConstructor
@Getter
@Setter
@Builder
@ToString
public class Keynote {
    private String id;
    private String nom;
    private String prenom;
    private String email;
    private String fonction;
}
```

ConferencesServices:

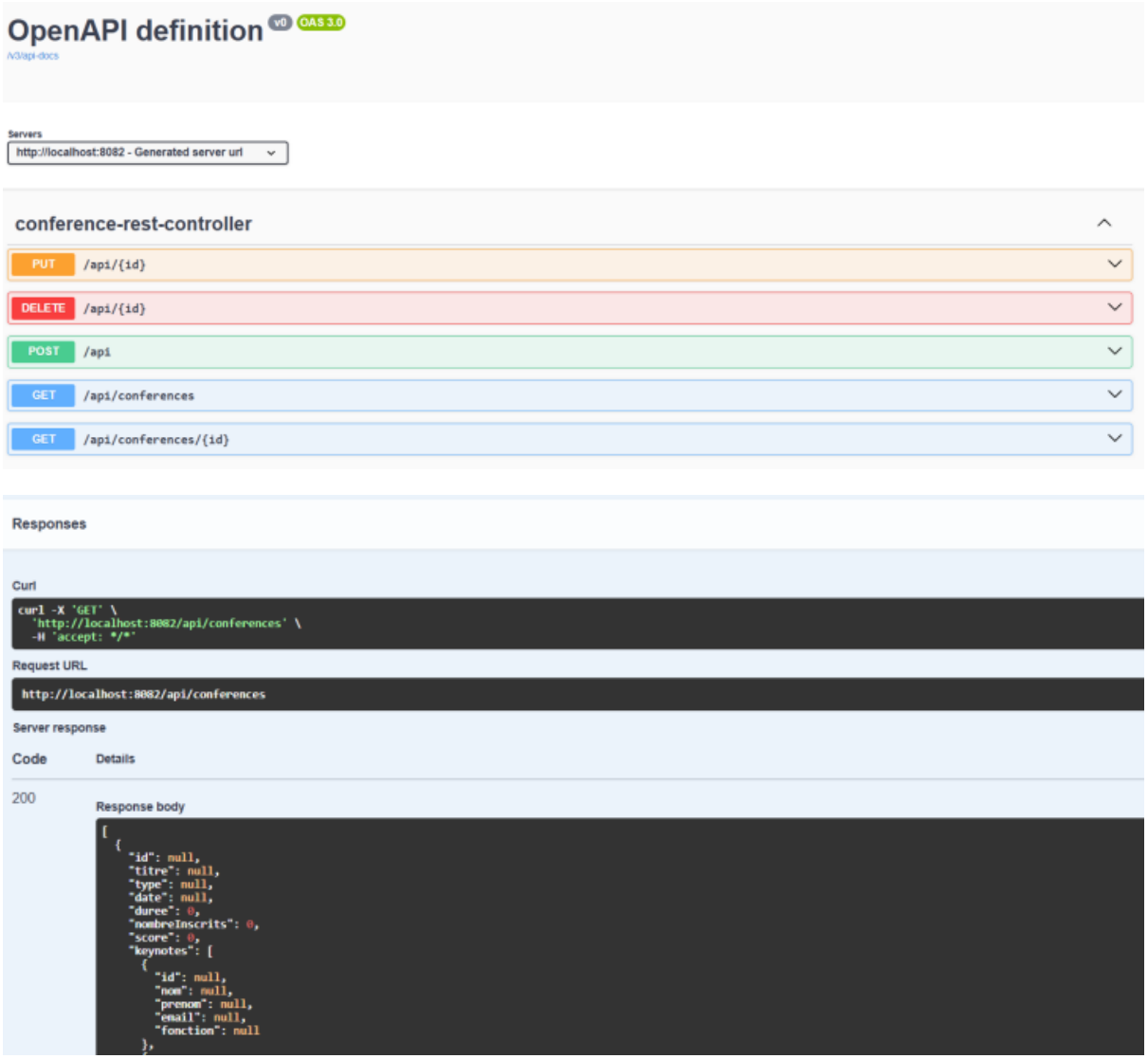
```
4 usages  1 implementation
public interface ConferenceService {
    1 usage  1 implementation
    List<ConferenceDTO> getAllConferences();
    2 usages  1 implementation
    Optional<ConferenceDTO> getConferenceById(Long id);
    1 usage  1 implementation
    ConferenceDTO createConference(ConferenceDTO conferenceDTO);
    1 usage  1 implementation
    void deleteConference(Long id);
    1 usage  1 implementation
    ConferenceDTO updateConference(ConferenceDTO conferenceDTO);
}
```

Controller:

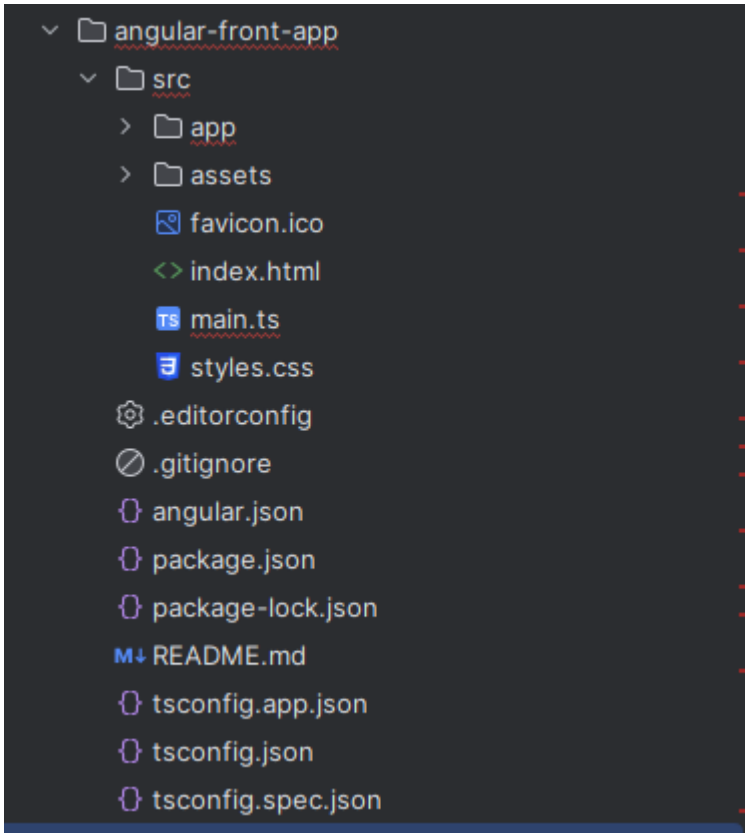
```
@RestController
@RequestMapping("/api")
public class ConferenceRestController {

    7 usages
    private ConferenceService conferenceService;
    2 usages
    private KeynoteRestClient keynoteRestClient;

    public ConferenceRestController(ConferenceService conferenceService, KeynoteRestClient keynoteRestClient) {
        this.conferenceService = conferenceService;
        this.keynoteRestClient = keynoteRestClient;
    }

    @GetMapping("/conferences")
    public List<ConferenceDTO> conferenceList(){
        List<ConferenceDTO> conferences = conferenceService.getAllConferences();
        conferences.forEach(c->{
                c.setKeynotes(keynoteRestClient.getAllKeynotes());
        });
        return conferences;
    }

    @GetMapping("/conferences/{id}")
    public ConferenceDTO conferenceById(@PathVariable Long id) { return conferenceService.getConferenceById(id).get(); }
```

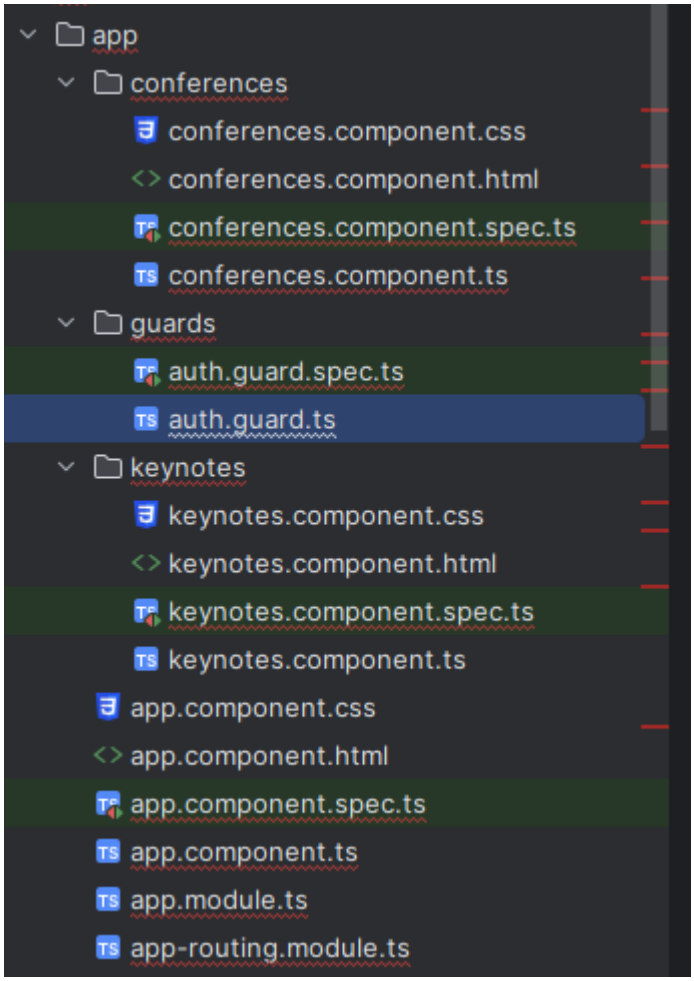
```
@PostMapping
public ConferenceDTO createConference(@RequestBody ConferenceDTO conferenceDTO) {
    ConferenceDTO savedConference = conferenceService.createConference(conferenceDTO);
    return savedConference;
}

@PutMapping("/{id}")
public ConferenceDTO updateConference(@PathVariable Long id, @RequestBody ConferenceDTO conferenceDTO) {
    if (!conferenceService.getConferenceById(id).isPresent()) {
        return null;
    }
    conferenceDTO.setId(id);  // Make sure the ID is set
    ConferenceDTO updatedConference = conferenceService.updateConference(conferenceDTO);
    return updatedConference;
}

@DeleteMapping("/{id}")
public void deleteConference(@PathVariable Long id) { conferenceService.deleteConference(id); }
```

Testing :



6. Développer un simple frontend web pour l'application


angular-front-app
src
app
assets
favicon.ico
index.html
main.ts
styles.css
.editorconfig
.gitignore
angular.json
package.json
package-lock.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json


app
conferences
conferences.component.css
conferences.component.html
conferences.component.spec.ts
conferences.component.ts
guards
auth.guard.spec.ts
auth.guard.ts
keynotes
keynotes.component.css
keynotes.component.html
keynotes.component.spec.ts
keynotes.component.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
app-routing.module.ts

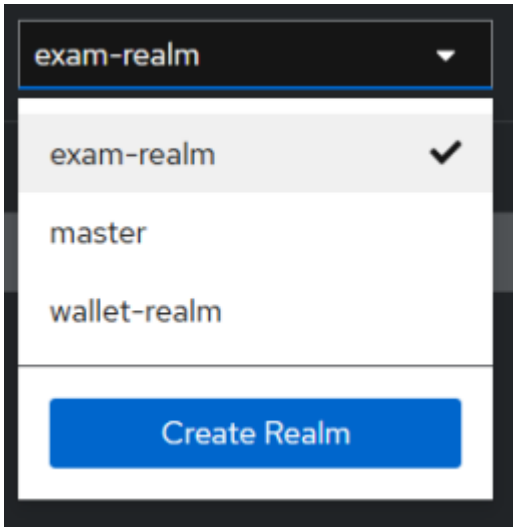
```
const routes: Routes = [
  {path : "keynotes", component : KeynotesComponent, canActivate : [AuthGuard], data : {roles :['A
  {path : "conferences", component : ConferencesComponent, canActivate : [AuthGuard], data : {role
];
```

7. Sécuriser l'application avec une authentification Keycloak

Demarrer docker et keycloak:

```
Invite de commandes
Microsoft Windows [version 10.0.19045.5247]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\SetupGame>docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/key
cloak:21.0.1 start-dev
```

Creation d'un realm



Creation de client :

Clients

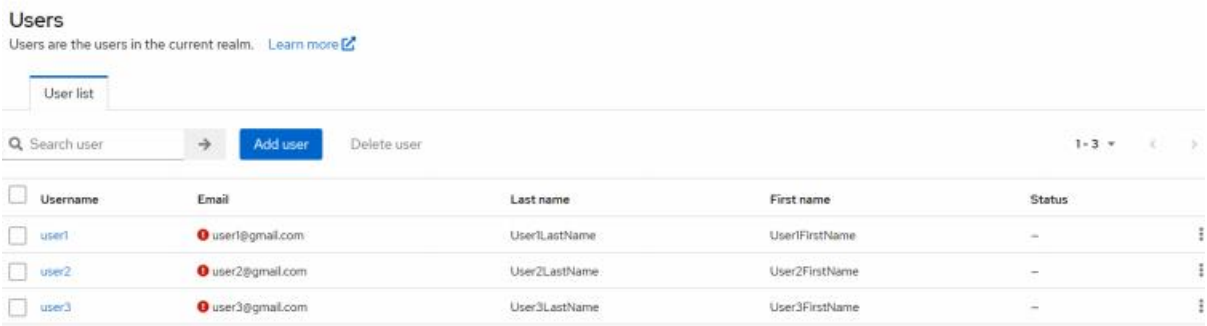
Clients are applications and services that can request authentication of a user. Learn more

Clients list | Initial access token | Client registration

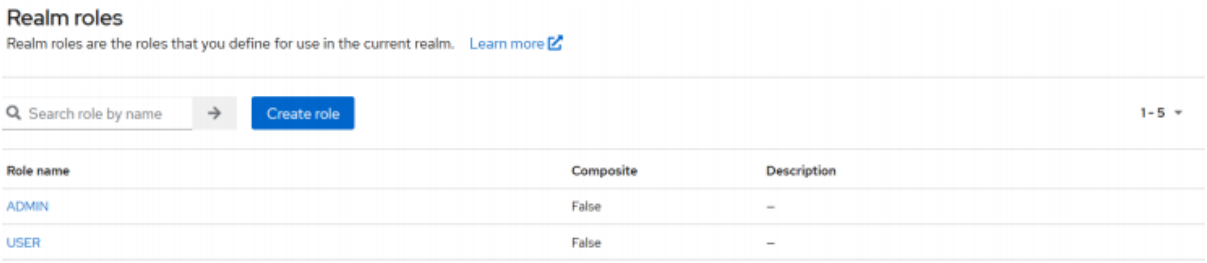
Search for client | Create client | Import client | 1-7

Client ID	Name	Type	Description	Home URL
account	${client_account}	OpenID Connect	–	http://localhost:8080/realms/exam-realm/account/
account-console	${client_account-console}	OpenID Connect	–	http://localhost:8080/realms/ecom-realm/account/
admin-cli	${client_admin-cli}	OpenID Connect	–	–
broker	${client_broker}	OpenID Connect	–	–
exam-client-ang	–	OpenID Connect	–	–
realm-management	${client_realm-management}	OpenID Connect	–	–
security-admin-console	${client_security-admin-console}	OpenID Connect	–	http://localhost:8080/admin/exam-realm/console/

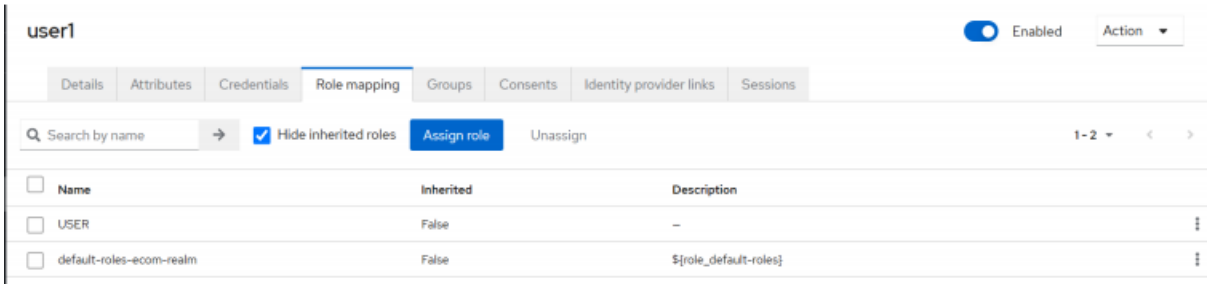
Creation des utilisateurs:



Creation des roles :



Donner des roles aux utilisateurs :



Backend modification :

JWTAuthConverter:

```java
@Component
public class JwtAuthConverter implements Converter<Jwt, AbstractAuthenticationToken> {
    1 usage
    private final JwtGrantedAuthoritiesConverter jwtGrantedAuthoritiesConverter=new JwtGrantedAuthoritiesConverte
    @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"));
    }
    1 usage
    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());
    }

}
```

SecurityConfig :

```java
@EnableMethodSecurity(prePostEnabled = true)
public class SecurityConfig {
    2 usages
    private JwtAuthConverter jwtAuthConverter;
    public SecurityConfig(JwtAuthConverter jwtAuthConverter) { this.jwtAuthConverter = jwtAuthConverter; }
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity httpSecurity) throws Exception {
        return httpSecurity
                .cors(Customizer.withDefaults())
                .sessionManagement(sm->sm.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .csrf(csrf->csrf.disable())
                .headers(h->h.frameOptions(fo->fo.disable()))
                .authorizeHttpRequests(ar->ar.requestMatchers("/h2-console/**").permitAll())
                .authorizeHttpRequests(ar->ar.requestMatchers("/api/keynotes/**").hasAuthority("ADMIN"))
                .authorizeHttpRequests(ar->ar.anyRequest().authenticated())
                .oauth2ResourceServer(o2->o2.jwt(jwt->jwt.jwtAuthenticationConverter(jwtAuthConverter)))
                .build();
    }
    @Bean
    CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(Arrays.asList("*"));
        configuration.setAllowedMethods(Arrays.asList("*"));
        configuration.setAllowedHeaders(Arrays.asList("*"));
        configuration.setExposedHeaders(Arrays.asList("*"));
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration( pattern: "/**", configuration);
        return source;
```

Intercepteur au service conférence :

```java
@Component
public class FeignInterceptor implements RequestInterceptor {
    @Override
    public void apply(RequestTemplate requestTemplate) {
        SecurityContext context = SecurityContextHolder.getContext();
        Authentication authentication = context.getAuthentication();
        JwtAuthenticationToken jwtAuthenticationToken= (JwtAuthenticationToken) authentication;
        String jwtAccessToken = jwtAuthenticationToken.getToken().getTokenValue();
        requestTemplate.header( name: "Authorization", ...values: "Bearer "+jwtAccessToken);
    }
}
```

Test PostMan:

POST http://localhost:8080/realms/exam-realm/protocol/openid-connect/token Send

Params Authorization Headers (9) Body Scripts Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description	Bulk Edit
username	user2		
password	1234		
grant_type	password		
client_id	exam-client-ang		

Body Cookies Headers (11) Test Results 200 OK 72 ms 2.88 KB

Pretty Raw Preview Visualize JSON


```
1 {
2     "access_token": "eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICJaX1VFV0ZPcjBNWV01RDhSZy1UTmhiblhXQmFpV1ljaFB5N0JEwnd6dXNrIn0.
        eyJleHAiOjE3MzQzNTE0NzgsImlhdCI6MTczNDM1MTE3OCw1anRpIjoiNTYyMmIwNTYtMWNjNS00ZWUwLWE5NGMtY2QzNTdhOGI0ZDBkIiw1aXNzIjo1aHR0cDovL2xvY2Fs
        aG9zdDo4MDgwL3JlYWxtcy9leGFtLXJlYWxtIiwiYXVkIjoiYWNjb3VudCIsInN1YiI6IjEwMDM1MjIyLTJkM2UtNDRjNy1iZmE5LTliZWRmOWQwMTFjNiIsInR5cCI6IkJl
        YXJlciIsImF6cCI6ImV4YW0tY2xpZW50LWFuZyIsInNlc3Npb25fc3RhdGUiOiJlYjNlZmZhNS0zMTZlLTRkMmMtYTEyYS04M2IzNTI2MzBhNDciLCJhY3I1OiIxIiwiYWxs
        b3dlZC1vcmlnaW5zIjpbImh0dHA6Ly9sb2NhbGhvc3Q6NDIwMCJdLCJyZWFsbV9hY2Nlc3MiOnsicm9sZXMiOlsiZGVmYXVsdC1yb2xlcy1lY29tLXJlYWxtIiwib2ZmbGlu
        ZV9hY2Nlc3MiLCJBRE1JTiIsInVtYV9hdXRob3JpemF0aW9uIiwiVVNFUiJdfSwicmVzb3VyY2VfYWNjZXNzIjp7ImFjY291bnQiOnsicm9sZXMiOlsibWFuYWdlLWFjY291
        bnQiLCJtYW5hZ2UtYWNjb3VudC1saW5rcyIsInZpZXctcHJvZmlsZSJdfX0sInNjb3BlIjoicHJvZmlsZSBlbWFpbCIsInNpZCI6ImV1M2VmZmE1LTMxNmUtNGQyYy1hMTJh
        LTgzYjM1MjYzMGE0NyIsImVtYWlsX3ZlcmlmaWVkIjpmYWxzZSwibmFtZSI6IlVzZXIyRmlyc3ROYW1lIFVzZXIyTGFzdE5hbWUiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJ1
        c2VyM1IsImdpdmVuX2NhbWUiOiJVc2VyMkZpcnN0TmFtZSIsInZhbWlseV9uYW1lIjoiVXNlcjMYXN0TmFtZSIsImVtYWlsIjoidXNlcjJAZ21haWwuY29tIn0.
        bxfmyKXxP6wcsMrK4oWuH97sMVJ3Ot9UuYrRIVNsOgBPB_VfxfGky1zRpwz4a1wbFcVdOG4Lg8UcjmkUN51X1Fqj-lpnAIrPMDHrsjm03z5LVF4D1zfH5DMf3Na9hl34YTcB
        mbHvdhq_NscRexnj-9WNtCTkBzOgmtgFU9HGKuQdDN32GQeMfAW401dKDAGw1wsrHVXerWY3qBCY7G7NYFWLsCOVM8VUqKWT7PJ6ItXtOiHbeD33sM0N3LgmmX6LUhLEe_d
        JZdHvo4elaPbIBF8Q1Z9iFaHYkh_C3O2G7kj1I7Owwka21ygR26TOfSoLsy9YvM1WWWIq5q0SNV44w",
3     "expires_in": 300,
4     "refresh_expires_in": 1800,
5     "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCIgOiAiSldUIiwia2lkIiA6ICIxOWVmZDg4MC05MDU0LTRhOTUtOWFkYS05MjFlOTZhMjlMTEifQ.
```


http://localhost:8082/api/conferences Save Share

GET http://localhost:8082/api/conferences Send

Params Authorization Headers (8) Body Scripts Settings Cookies

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwi...

Body Cookies Headers (13) Test Results 200 OK 46 ms 1.27 KB

Pretty Raw Preview Visualize JSON


```
1 [
2     {
3         "id": null,
4         "titre": null,
5         "type": null,
6         "date": null,
7         "duree": 0,
8         "nombreInscrits": 0,
9         "score": 0.0,
10        "keynotes": [
11            {
12                "id": null,
13                "nom": null,
```

http://localhost:8081/api/keynotes

Save Share

GET http://localhost:8081/api/keynotes Send

Params Authorization Headers (8) Body Scripts Settings Cookies

uth Type

Bearer Token

he authorization header will be automatically enerated when you send the request. Learn ore about Bearer Token authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.

Token eyJhbGciOiJSUzI1NiIsInR5cCIgOiAiSldUIiwi...

Body Cookies Headers (13) Test Results 200 OK 8 ms 599 B

Pretty Raw Preview Visualize JSON


```
[
    {
        "id": null,
        "nom": null,
        "prenom": null,
        "email": null,
        "fonction": null
    },
```