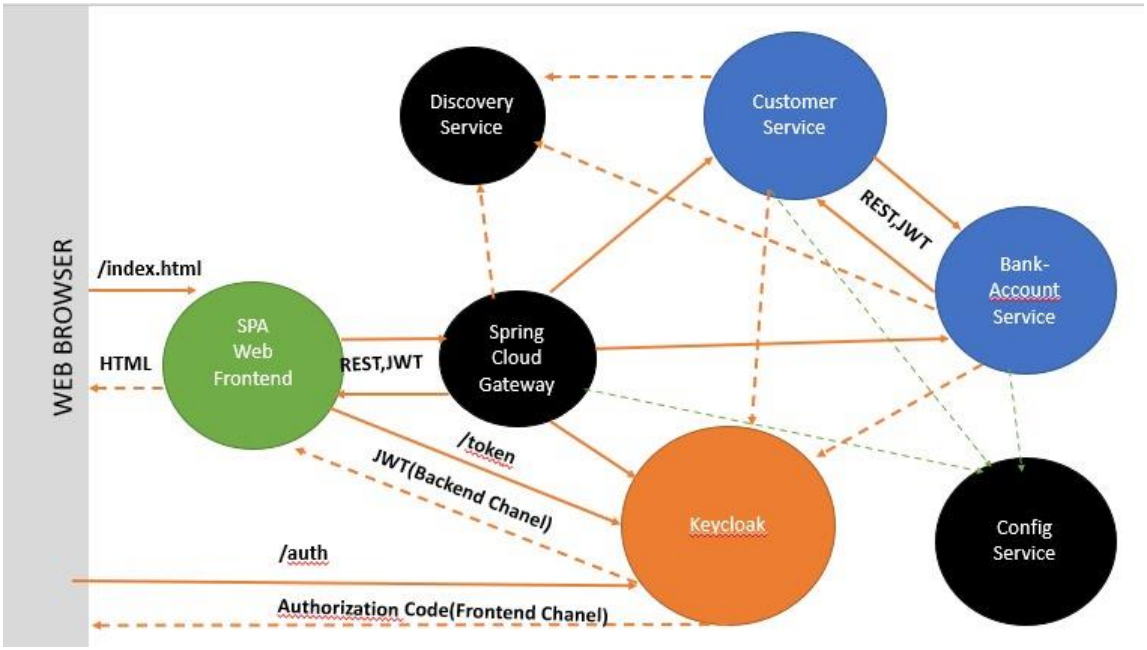
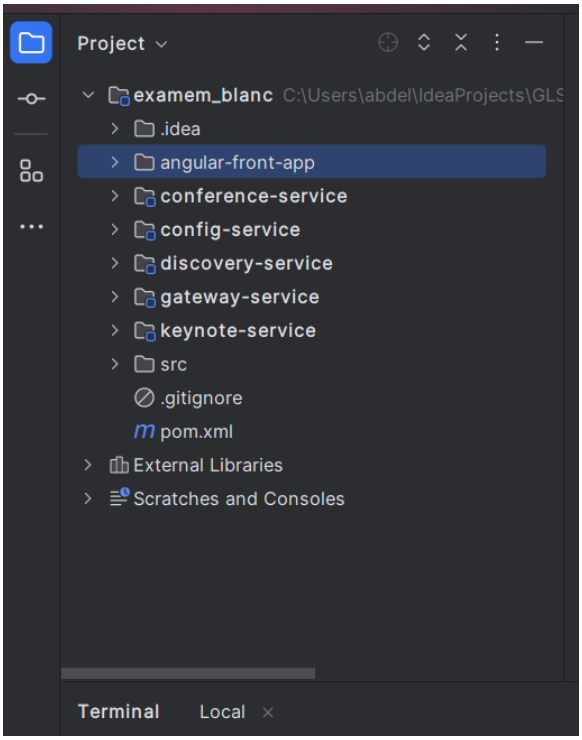
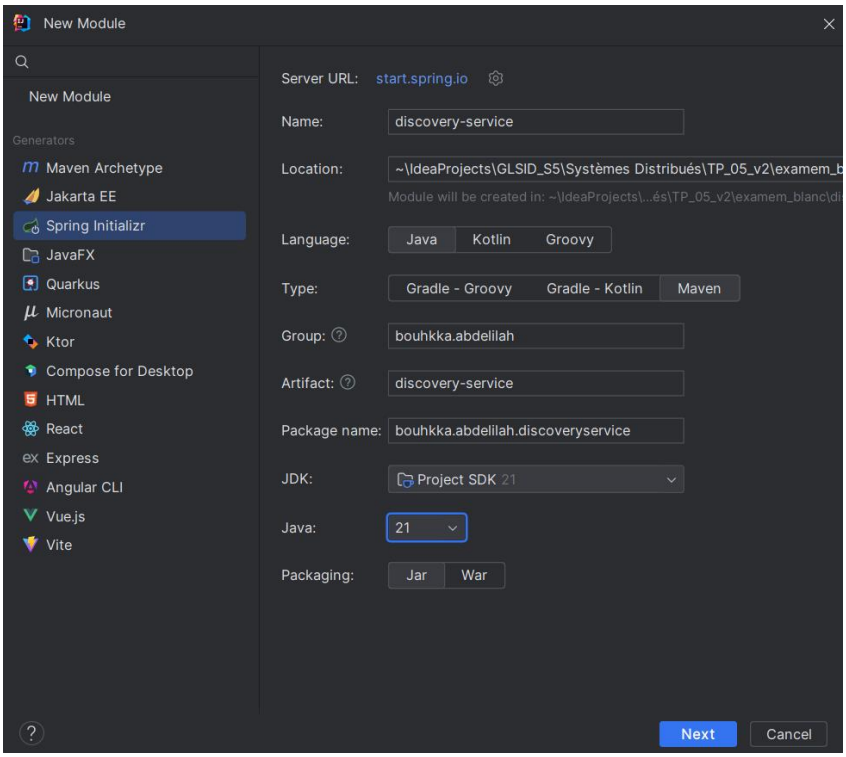
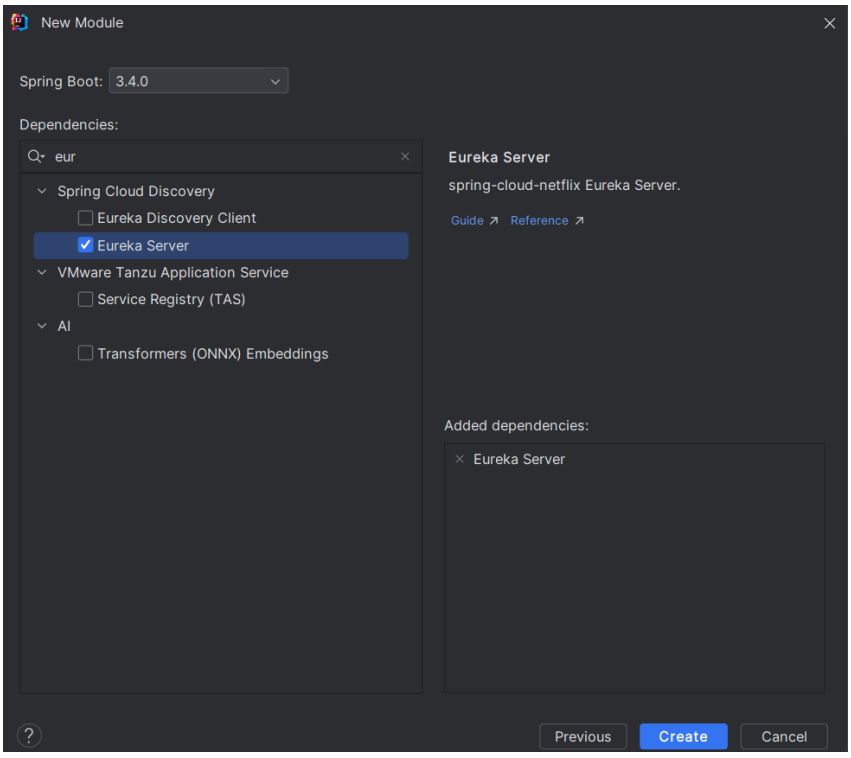
Établir une architecture technique du projet

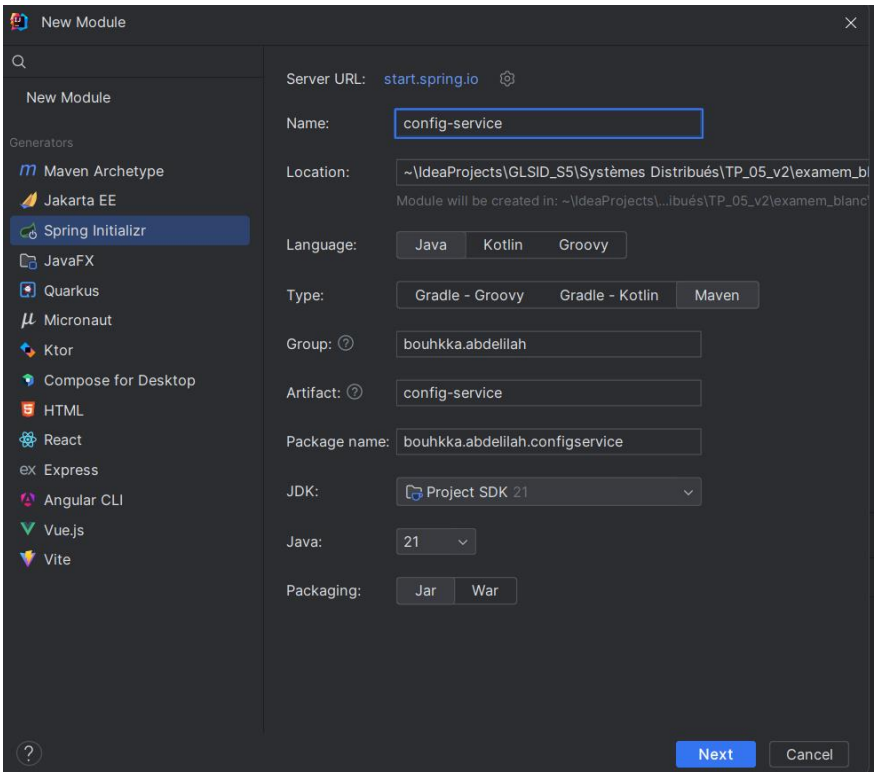


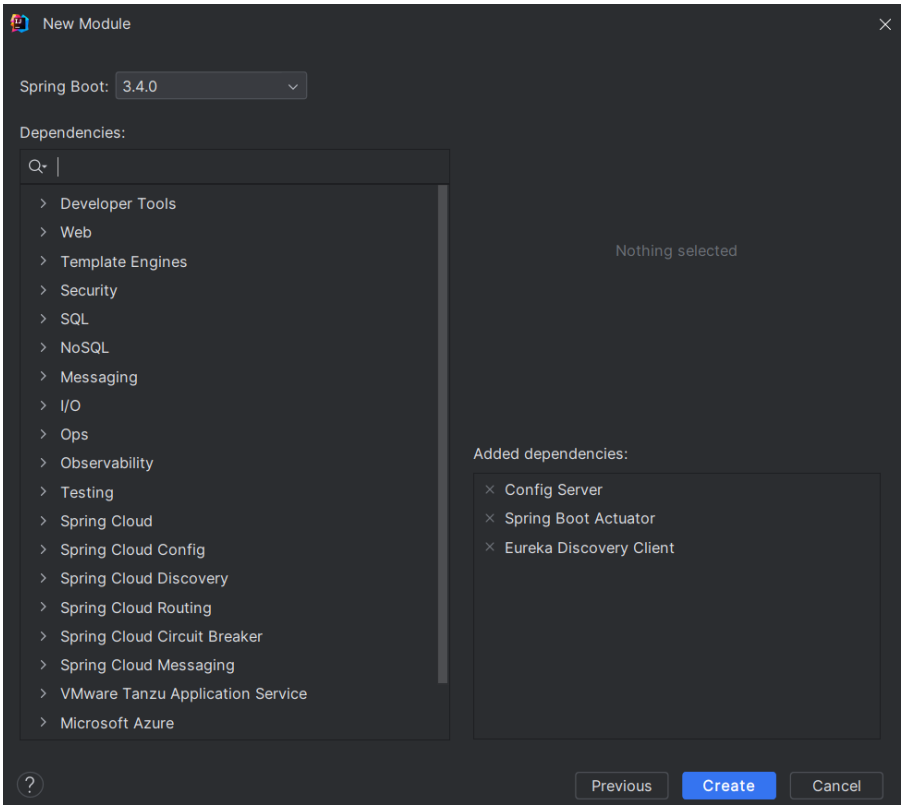
Créer un Projet Maven incluant les micro-services suivants : keynote-service, conference-service, gateway-service, discovery-service, config-service et angular-front-app

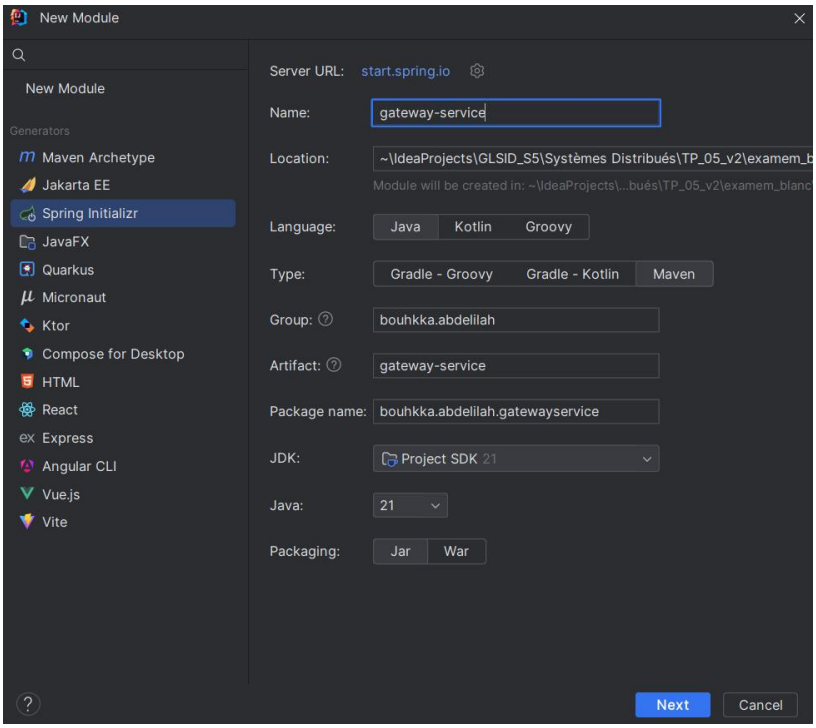


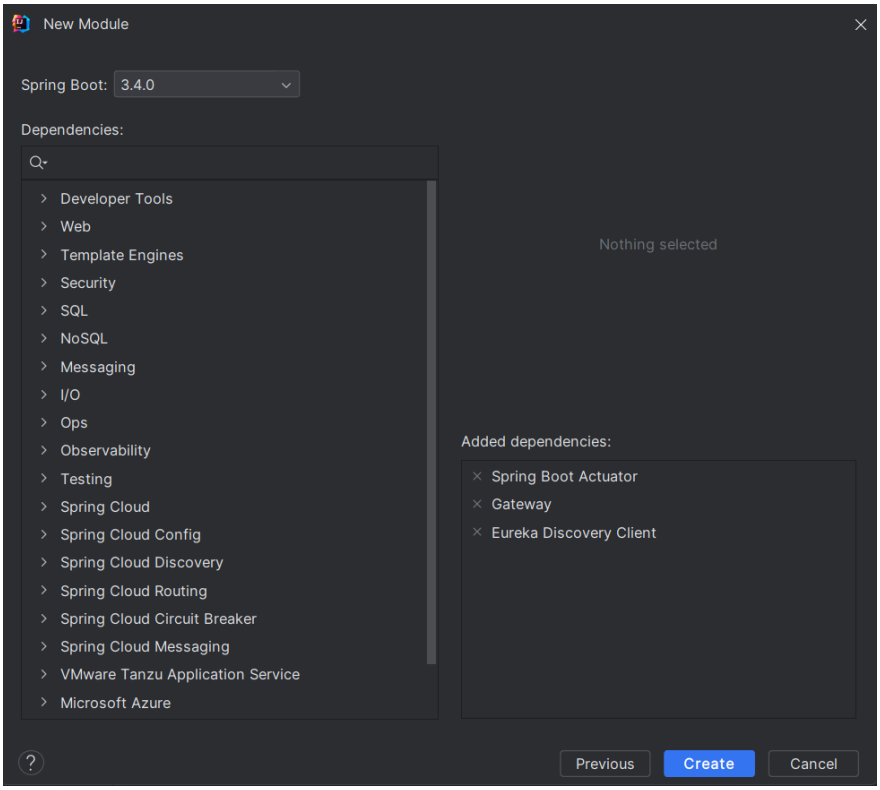

New Module
New Module
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Compose for Desktop
HTML
React
Express
Angular CLI
Vue.js
Vite
Server URL: start.spring.io
Name: discovery-service
Location: ~\IdeaProjects\GLSID_S5\Systèmes Distribués\TP_05_v2\examem_b
Module will be created in: ~\IdeaProjects\...és\TP_05_v2\examem_blanc\di
Language: Java Kotlin Groovy
Type: Gradle - Groovy Gradle - Kotlin Maven
Group: bouhkka.abdelilah
Artifact: discovery-service
Package name: bouhkka.abdelilah.discoveryservice
JDK: Project SDK 21
Java: 21
Packaging: Jar War
Next
Cancel

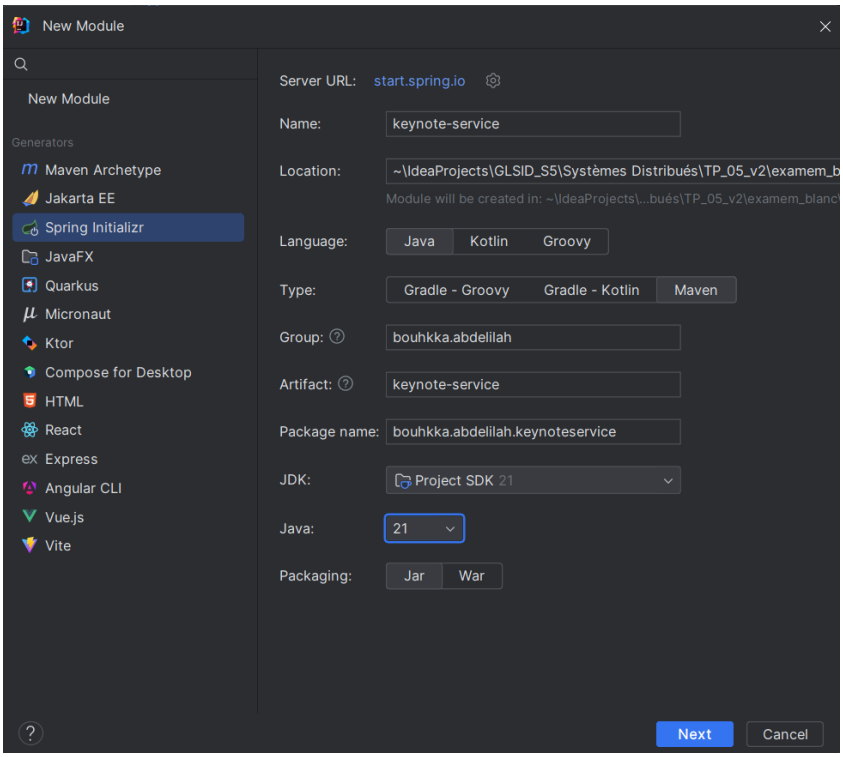

New Module
Spring Boot: 3.4.0
Dependencies:
eur
Spring Cloud Discovery
Eureka Discovery Client
Eureka Server
VMware Tanzu Application Service
Service Registry (TAS)
AI
Transformers (ONNX) Embeddings
Eureka Server
spring-cloud-netflix Eureka Server.
Guide
Reference
Added dependencies:
Eureka Server
Previous
Create
Cancel

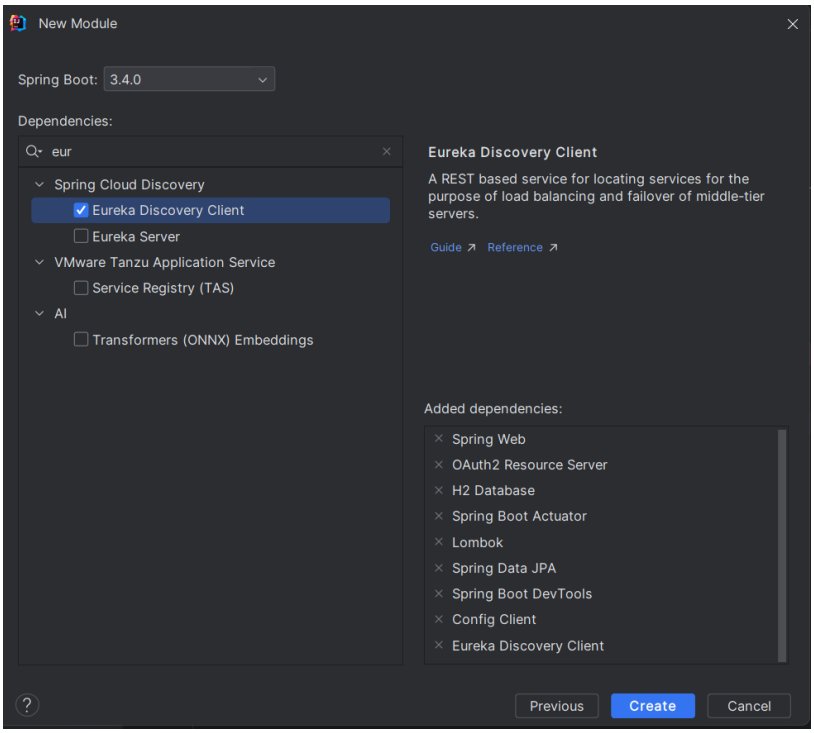

New Module
New Module
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Compose for Desktop
HTML
React
Express
Angular CLI
Vue.js
Vite
Server URL: start.spring.io
Name: config-service
Location: ~\IdeaProjects\GLSID_S5\Systèmes Distribués\TP_05_v2\examem_bl
Module will be created in: ~\IdeaProjects\...ibués\TP_05_v2\examem_blanc
Language: Java Kotlin Groovy
Type: Gradle - Groovy Gradle - Kotlin Maven
Group: bouhkka.abdelilah
Artifact: config-service
Package name: bouhkka.abdelilah.configservice
JDK: Project SDK 21
Java: 21
Packaging: Jar War
Next
Cancel

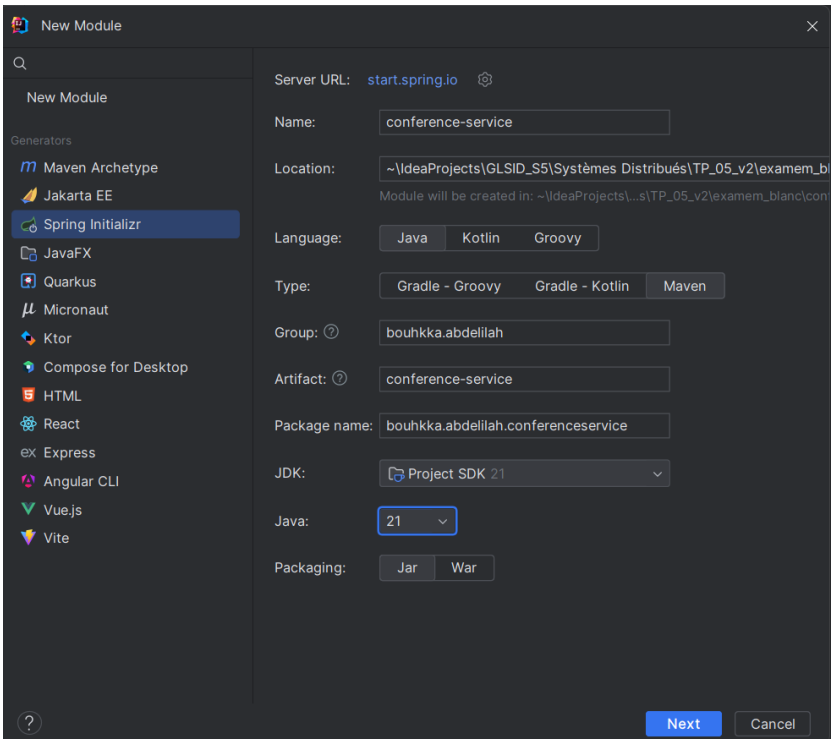

New Module
Spring Boot: 3.4.0
Dependencies:
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Messaging
VMware Tanzu Application Service
Microsoft Azure
Nothing selected
Added dependencies:
Config Server
Spring Boot Actuator
Eureka Discovery Client
Previous
Create
Cancel

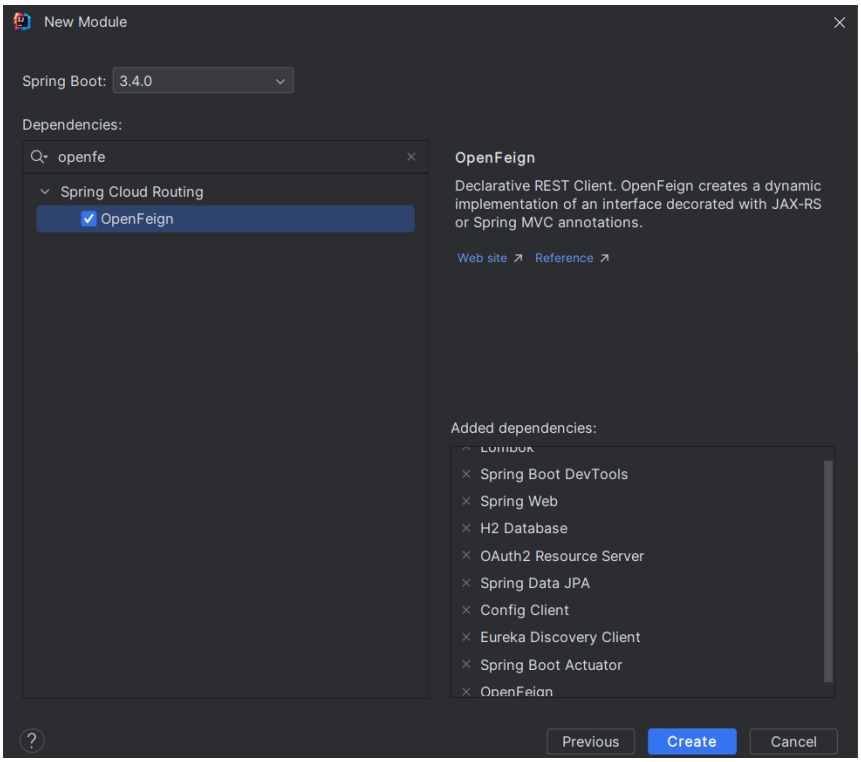

New Module
New Module
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Compose for Desktop
HTML
React
Express
Angular CLI
Vue.js
Vite
Server URL: start.spring.io
Name: gateway-service
Location: ~\IdeaProjects\GLSID_S5\Systèmes Distribués\TP_05_v2\examem_b
Module will be created in: ~\IdeaProjects\...bués\TP_05_v2\examem_blanc
Language: Java Kotlin Groovy
Type: Gradle - Groovy Gradle - Kotlin Maven
Group: bouhkka.abdelilah
Artifact: gateway-service
Package name: bouhkka.abdelilah.gatewayservice
JDK: Project SDK 21
Java: 21
Packaging: Jar War
Next
Cancel


New Module
Spring Boot: 3.4.0
Dependencies:
Developer Tools
Web
Template Engines
Security
SQL
NoSQL
Messaging
I/O
Ops
Observability
Testing
Spring Cloud
Spring Cloud Config
Spring Cloud Discovery
Spring Cloud Routing
Spring Cloud Circuit Breaker
Spring Cloud Messaging
VMware Tanzu Application Service
Microsoft Azure
Nothing selected
Added dependencies:
Spring Boot Actuator
Gateway
Eureka Discovery Client
Previous
Create
Cancel


New Module
New Module
Generators
Maven Archetype
Jakarta EE
Spring Initializr
JavaFX
Quarkus
Micronaut
Ktor
Compose for Desktop
HTML
React
Express
Angular CLI
Vue.js
Vite
Server URL: start.spring.io
Name: keynote-service
Location: ~\IdeaProjects\GLSID_S5\Systèmes Distribués\TP_05_v2\examem_b
Module will be created in: ~\IdeaProjects\...bués\TP_05_v2\examem_blanc
Language: Java Kotlin Groovy
Type: Gradle - Groovy Gradle - Kotlin Maven
Group: bouhkka.abdelilah
Artifact: keynote-service
Package name: bouhkka.abdelilah.keynoteservice
JDK: Project SDK 21
Java: 21
Packaging: Jar War
Next
Cancel


New Module
Spring Boot: 3.4.0
Dependencies:
eur
Spring Cloud Discovery
Eureka Discovery Client
Eureka Server
VMware Tanzu Application Service
Service Registry (TAS)
AI
Transformers (ONNX) Embeddings
Eureka Discovery Client
A REST based service for locating services for the purpose of load balancing and failover of middle-tier servers.
Guide ↗ Reference ↗
Added dependencies:
Spring Web
OAuth2 Resource Server
H2 Database
Spring Boot Actuator
Lombok
Spring Data JPA
Spring Boot DevTools
Config Client
Eureka Discovery Client
Previous
Create
Cancel





Développer et tester les micro-services discovery-service et gateway-service et config-service

- Gateway-service :

```java
@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);
    }
}
```

```properties
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
```

- discovery-service :

```java
@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServiceApplication {

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

}
```

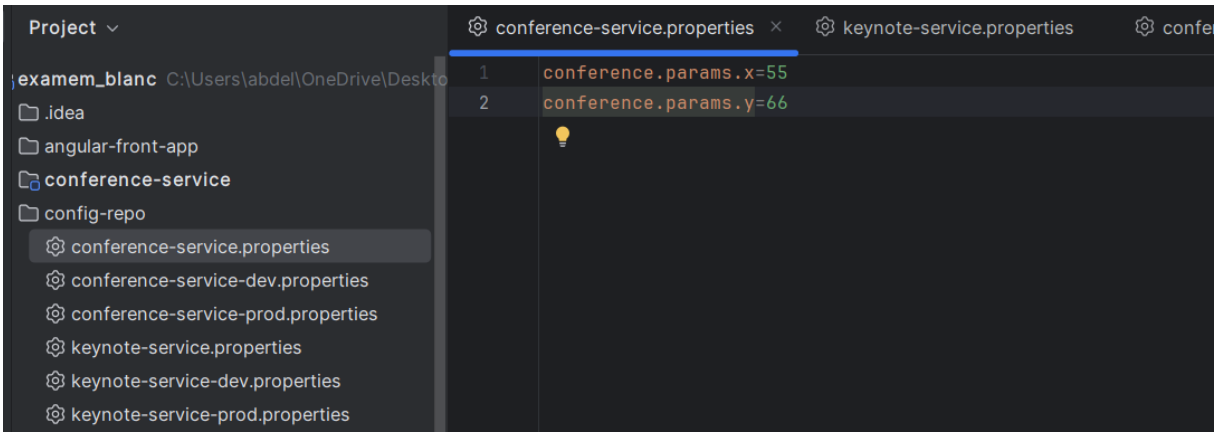
```properties
spring.application.name=discovery-service

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

- config-service :

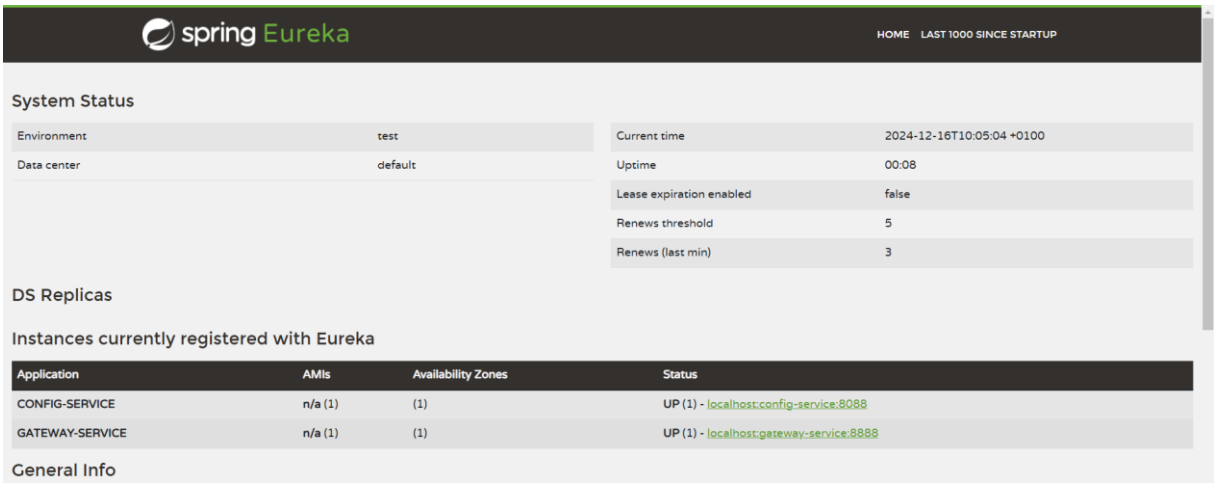
```java
@SpringBootApplication
@EnableConfigServer
@EnableDiscoveryClient
public class ConfigServiceApplication {

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

}
```

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

server.port=8088
spring.cloud.config.server.git.uri=file:///C:/Users/abdel/OneDrive/Desktop/GLSID_S5/examem_blanc/config-repo
```

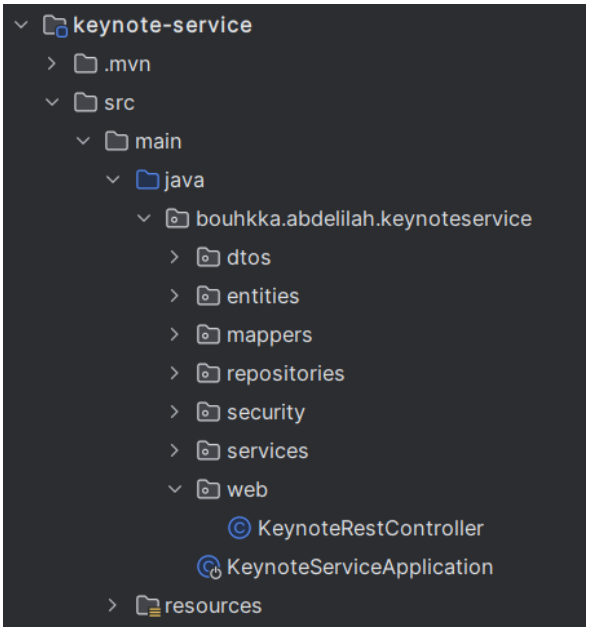
Créer un config-repo qui contient les fichiers de configurations :




```
PS C:\Users\abdel\OneDrive\Desktop\GLSID_S5\examem_blanc> cd .\config-repo\
PS C:\Users\abdel\OneDrive\Desktop\GLSID_S5\examem_blanc\config-repo> git init
Initialized empty Git repository in C:/Users/abdel/OneDrive/Desktop/GLSID_S5/examem_blanc/config-repo/.git/
PS C:\Users\abdel\OneDrive\Desktop\GLSID_S5\examem_blanc\config-repo> git add .
PS C:\Users\abdel\OneDrive\Desktop\GLSID_S5\examem_blanc\config-repo> git commit -m "first commit"
```



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



a. Entities :

```
9 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;
}
```

b. Repositoriy

```
import bouhkka.abdelilah.keynoteservice.entities.Keynote;
import org.springframework.data.jpa.repository.JpaRepository;

5 usages
public interface KeynoteRepository extends JpaRepository<Keynote, String> {
}
```

c. DTO :

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

Ajouter les dépendances de **mapstruct** et **springdoc openAPI**

```xml
        <dependency>
            <groupId>org.springdoc</groupId>
            <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
            <version>2.3.0</version>
        </dependency>
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct</artifactId>
            <version>1.5.0.Final</version>
        </dependency>
        <dependency>
            <groupId>org.mapstruct</groupId>
            <artifactId>mapstruct-processor</artifactId>
            <version>1.5.0.Final</version>
            <scope>provided</scope>
        </dependency>
```

d. Mapper :

```java
3 usages  1 implementation
@Mapper
public interface KeynoteMapper {
    no usages
    KeynoteMapper INSTANCE = Mappers.getMapper(KeynoteMapper.class);
    no usages  1 implementation
    KeynoteDTO keynoteToKeynoteDTO(Keynote keynote);
    no usages  1 implementation
    Keynote keynoteDTOToKeynote(KeynoteDTO keynoteDTO);
}
```

e. Keynote service :

```java
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);
}
```

f. Keynote service implémentation :

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

    @Autowired
    private KeynoteRepository keynoteRepository;
    no usages
    @Override
    public List<KeynoteDTO> getAllKeynotes() {
        List<Keynote> keynotes = keynoteRepository.findAll();
        return keynotes.stream() Stream<Keynote>
                .map(KeynoteMapper.INSTANCE::keynoteToKeynoteDTO) Stream<KeynoteDTO>
                .toList();
    }

    no usages
    @Override
    public Optional<KeynoteDTO> getKeynoteById(String id) {
        return keynoteRepository.findById(id)
                .map(KeynoteMapper.INSTANCE::keynoteToKeynoteDTO);
    }
```

```java
    @Override
    public KeynoteDTO createKeynote(KeynoteDTO keynoteDTO) {
        Keynote keynote = KeynoteMapper.INSTANCE.keynoteDTOToKeynote(keynoteDTO);
        Keynote savedKeynote = keynoteRepository.save(keynote);
        return KeynoteMapper.INSTANCE.keynoteToKeynoteDTO(savedKeynote);
    }

    no usages
    @Override
    public void deleteKeynote(String id) {
        keynoteRepository.deleteById(id);
    }
}
```

g. Controller

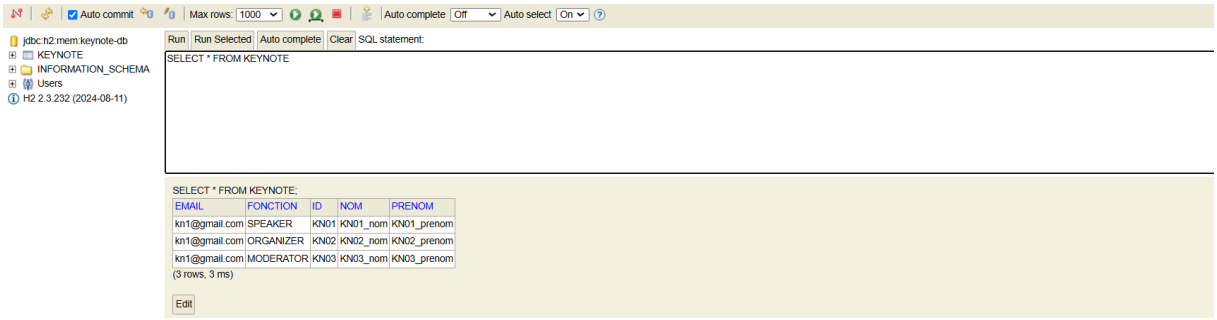
```java
@RestController
@RequestMapping("/api")
public class KeynoteRestController {

    7 usages
    private KeynoteService keynoteService;

    public KeynoteRestController(KeynoteService keynoteService) {
        this.keynoteService = keynoteService;
    }

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

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

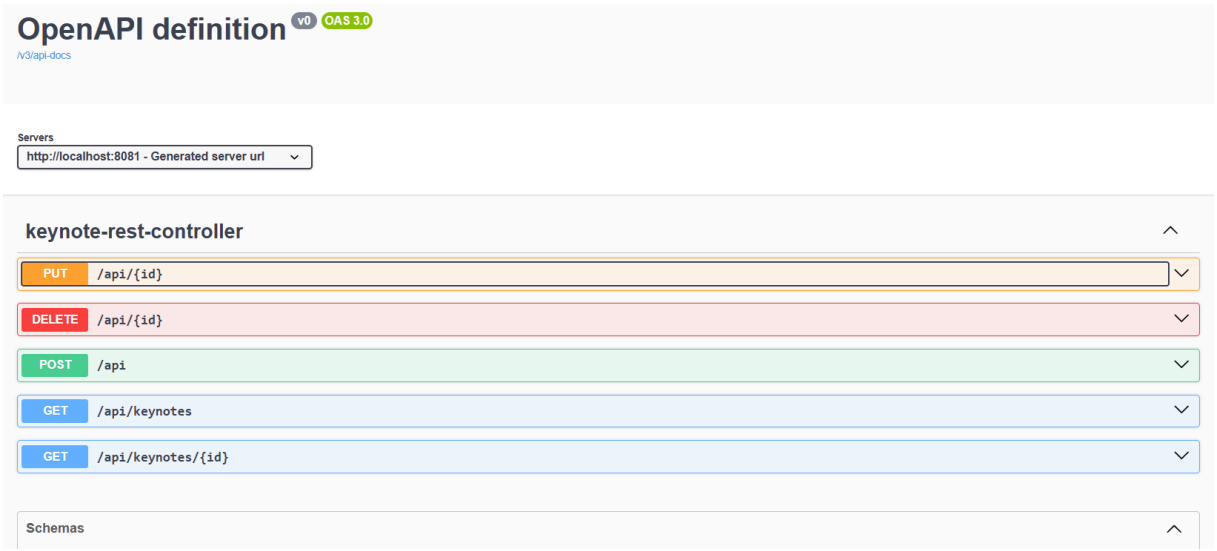
```java
@PostMapping
public KeynoteDTO createKeynote(@RequestBody KeynoteDTO keynoteDTO) {
    KeynoteDTO savedKeynote = keynoteService.createKeynote(keynoteDTO);
    return savedKeynote;
}

@PutMapping("/{id}")
public KeynoteDTO updateKeynote(@PathVariable String id, @RequestBody KeynoteDTO keynoteDTO) {
    if (!keynoteService.getKeynoteById(id).isPresent()) {
        return null;
    }
    keynoteDTO.setId(id);  // Make sure the ID is set
    KeynoteDTO updatedKeynote = keynoteService.updateKeynote(keynoteDTO);
    return updatedKeynote;
}

@DeleteMapping("/{id}")
public void deleteKeynote(@PathVariable String id) {
    keynoteService.deleteKeynote(id);
}
```

h. Application.properties :

```
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
```

Test :





GET /api/keynotes

Parameters Cancel

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8081/api/keynotes' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8081/api/keynotes
```

Server response

Code	Details
200	Response body


```
[
  {
    "id": "KN01",
    "nom": "KN01_nom",
    "prenom": "KN01_prenom",
    "email": "kn1@gmail.com",
    "fonction": "SPEAKER"
  },
  {
    "id": "KN02",
    "nom": "KN02_nom",
    "prenom": "KN02_prenom",
    "email": "kn1@gmail.com",
    "fonction": "ORGANIZER"
  },
```


GET /api/keynotes/{id}

Parameters Cancel

Name	Description
id * required string (path)	KN02

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8081/api/keynotes/KN02' \
  -H 'accept: */*'
```

Request URL

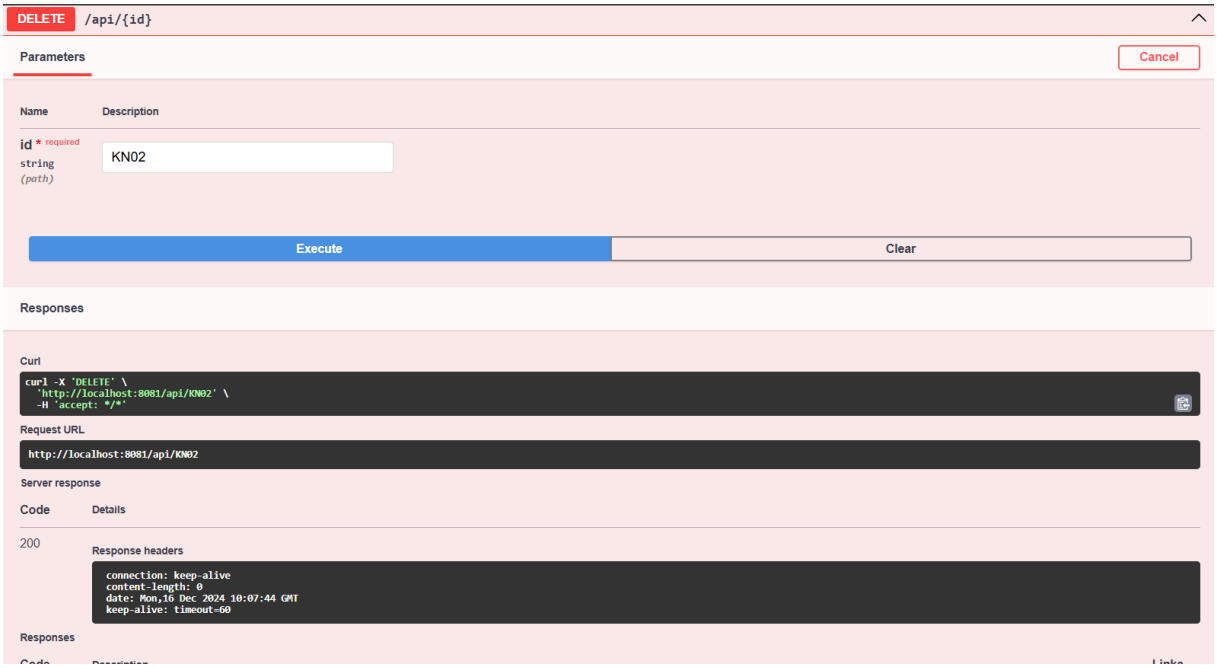
```
http://localhost:8081/api/keynotes/KN02
```

Server response

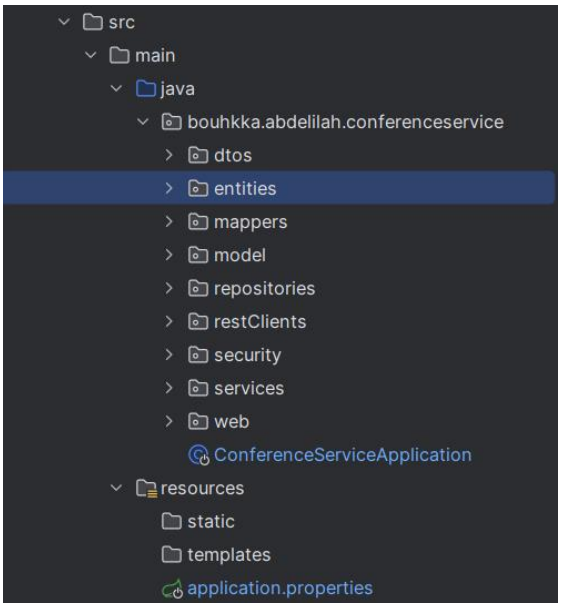
Code	Details
200	Response body


```
{
  "id": "KN02",
  "nom": "KN02_nom",
  "prenom": "KN02_prenom",
  "email": "kn1@gmail.com",
  "fonction": "ORGANIZER"
}
```


Download



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



a. 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;
}
```

b. DTO

```java
31 usages
@Data
public class ConferenceDTO {
    private Long id;
    private String titre;
    private String type;
    private LocalDate date;
    private int duree;
    private int nombreInscrits;
    private double score;
    private List<Keynote> keynotes;
}
```

c. Model

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

d. Mappers

```
8 usages  1 implementation
@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);
}
```

e. Repository

```
4 usages
public interface ConferenceRepository extends JpaRepository<Conference, String> {
}
```

f. RestClient

```
4 usages
@FeignClient(url = "http://localhost:8081", name = "keynote-service")
public interface KeynoteRestClient {
    @GetMapping("/api/keynotes")
    List<Keynote> getAllKeynotes();
    @GetMapping("/api/keynotes/{id}")
    Keynote findKeynoteById(@PathVariable String id);
}
```

g. ConferenceServices

```
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);
}
```

h. ConferenceServiceImpl :

```java
@Service
@Transactional
public class ConferenceServiceImpl implements ConferenceService {

    @Autowired
    private ConferenceRepository conferenceRepository;
    1 usage
    @Override
    public List<ConferenceDTO> getAllConferences() {
        List<Conference> conferences = conferenceRepository.findAll();
        return conferences.stream() Stream<Conference>
                .map(ConferenceMapper.INSTANCE::conferenceToConferenceDTO) Stream<ConferenceDTO>
                .toList();
    }

    2 usages
    @Override
    public Optional<ConferenceDTO> getConferenceById(Long id) {
        return conferenceRepository.findById(String.valueOf(id))
                .map(ConferenceMapper.INSTANCE::conferenceToConferenceDTO);
    }
```

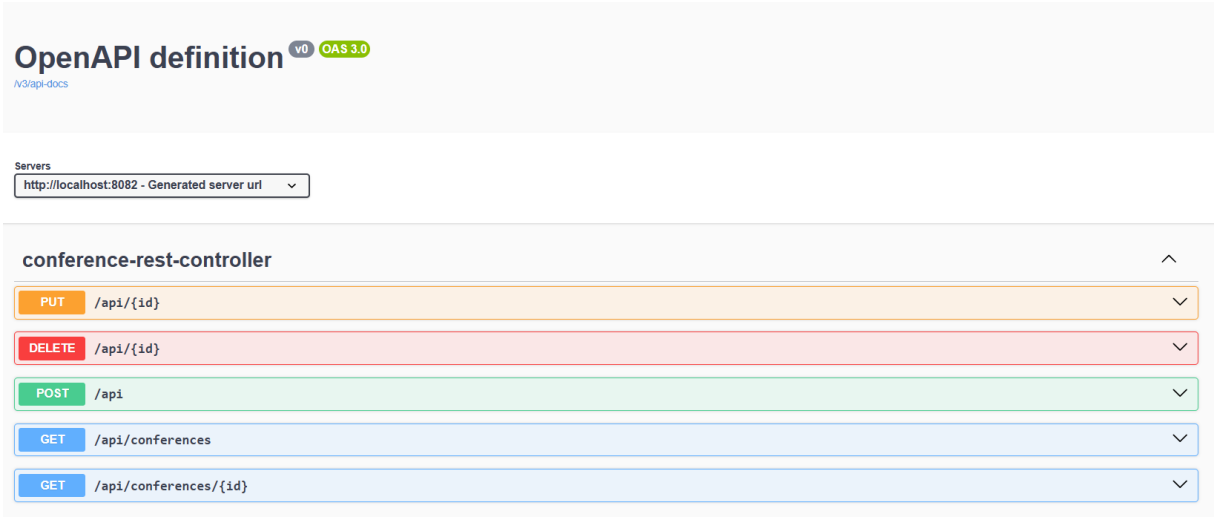
```java
    1 usage
    @Override
    public ConferenceDTO createConference(ConferenceDTO conferenceDTO) {
        Conference conference = ConferenceMapper.INSTANCE.conferenceDTOToConference(conferenceDTO);
        Conference savedConference = conferenceRepository.save(conference);
        return ConferenceMapper.INSTANCE.conferenceToConferenceDTO(savedConference);
    }

    1 usage
    @Override
    public void deleteConference(Long id) { conferenceRepository.deleteById(String.valueOf(id)); }

    1 usage
    @Override
    public ConferenceDTO updateConference(ConferenceDTO conferenceDTO) { return null; }
}
```

i. Controllers

```java
@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(); }
```

```java
    @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); }
```

j. Testing :

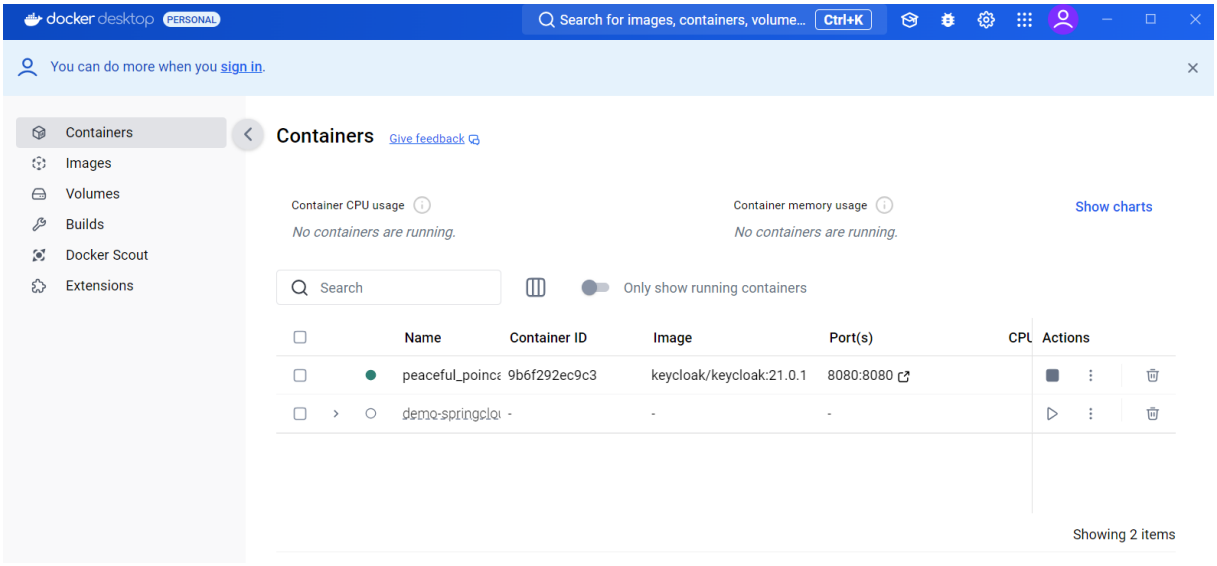


Développer un simple frontend web pour l'application

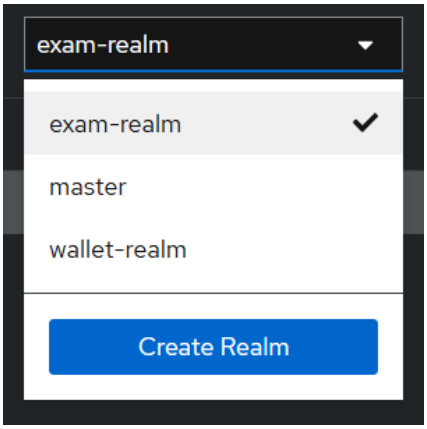
Sécuriser l'application avec une authentification Keycloak

Demarer Docker Desctop Et demarer Keycloak .

```
C:\Users\abdel>docker run -p 8080:8080 -e KEYCLOAK_ADMIN=admin -e KEYCLOAK_ADMIN_PASSWORD=admin quay.io/keycloak/keycloa
k:21.0.1 start-dev
Updating the configuration and installing your custom providers, if any. Please wait.
```



k. Creation d’un nouveau 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

Users

Users are the users in the current realm. Learn more

User list

Search user | Add user | Delete user | 1 - 3

Username	Email	Last name	First name	Status
user1	user1@gmail.com	User1LastName	User1FirstName	–
user2	user2@gmail.com	User2LastName	User2FirstName	–
user3	user3@gmail.com	User3LastName	User3FirstName	–

Creation des roles

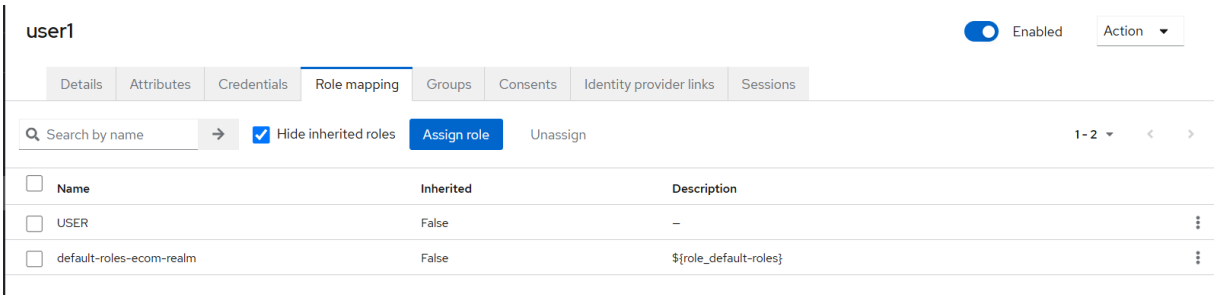
Realm roles

Realm roles are the roles that you define for use in the current realm. Learn more

Search role by name | Create role | 1 - 5

Role name	Composite	Description
ADMIN	False	–
USER	False	–

Affection des roles aux utilisateur



Déployer l’application avec Docker et Docker compose