ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

Rapport du Examen

Filière :
« Ingénierie Informatique : Big Data et Cloud Computing »

II-BDCC

Examen Systèmes Distribués

Réalisé par :

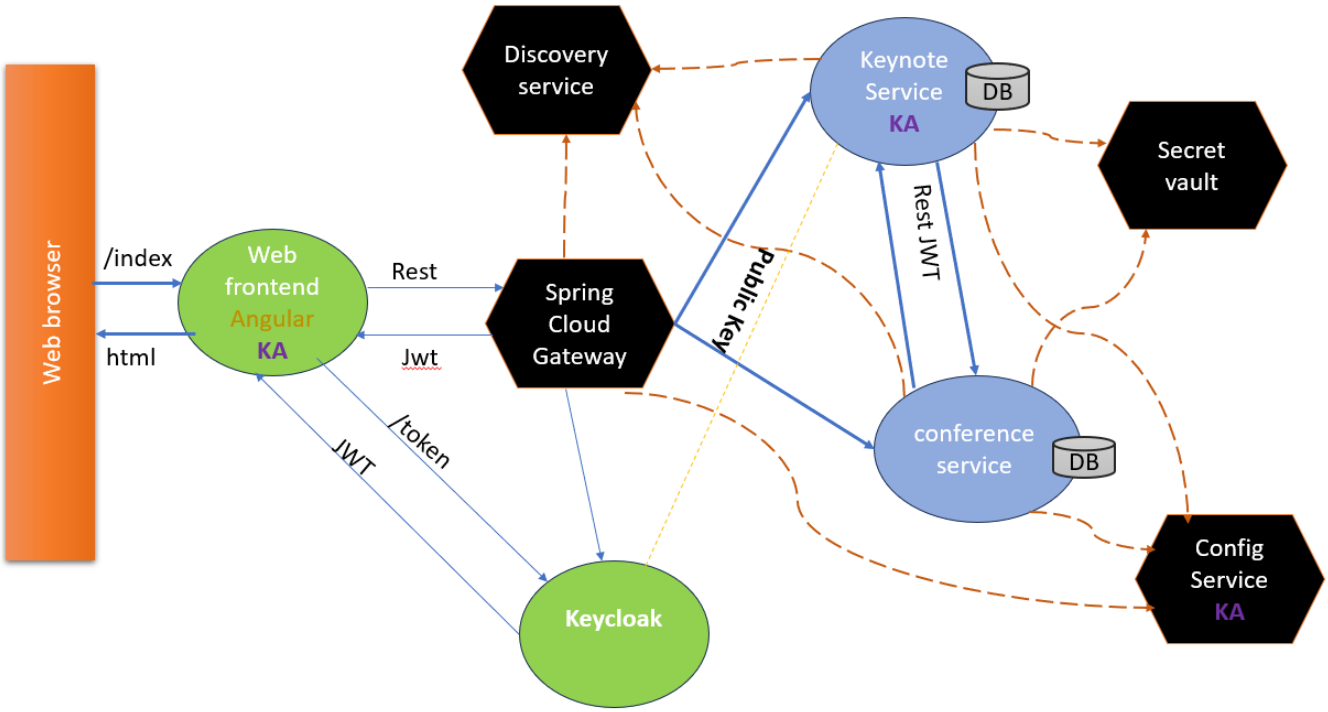
OUADOUCH Hassan

réalisé par :

Pr. Mohamed YOUSSFI

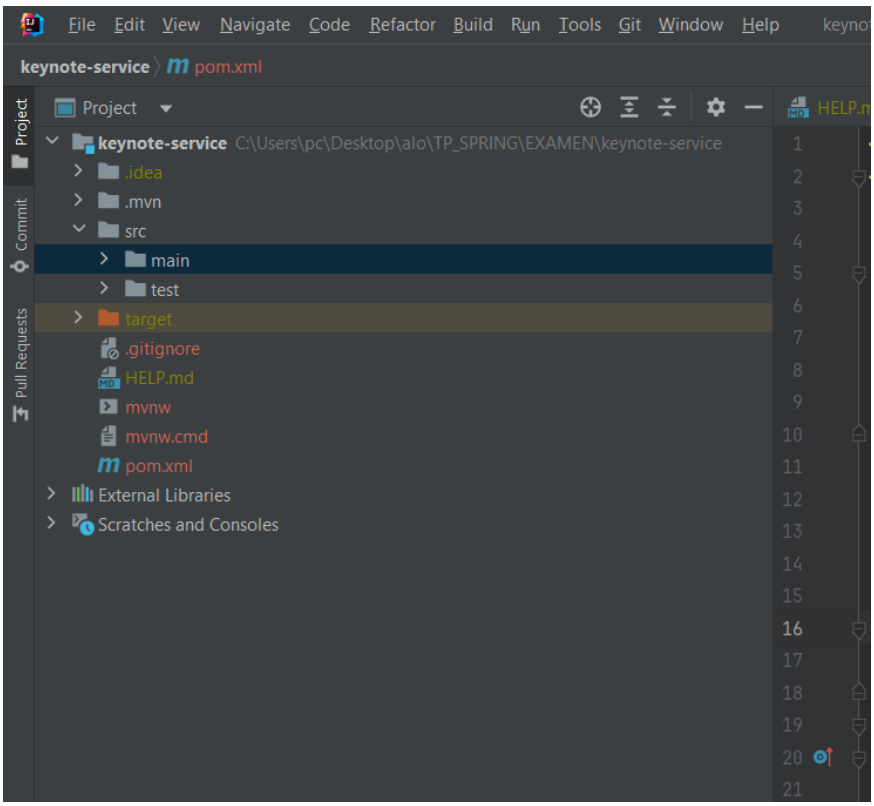
Année Universitaire : 2023-2024

1. Établir une architecture technique du projet

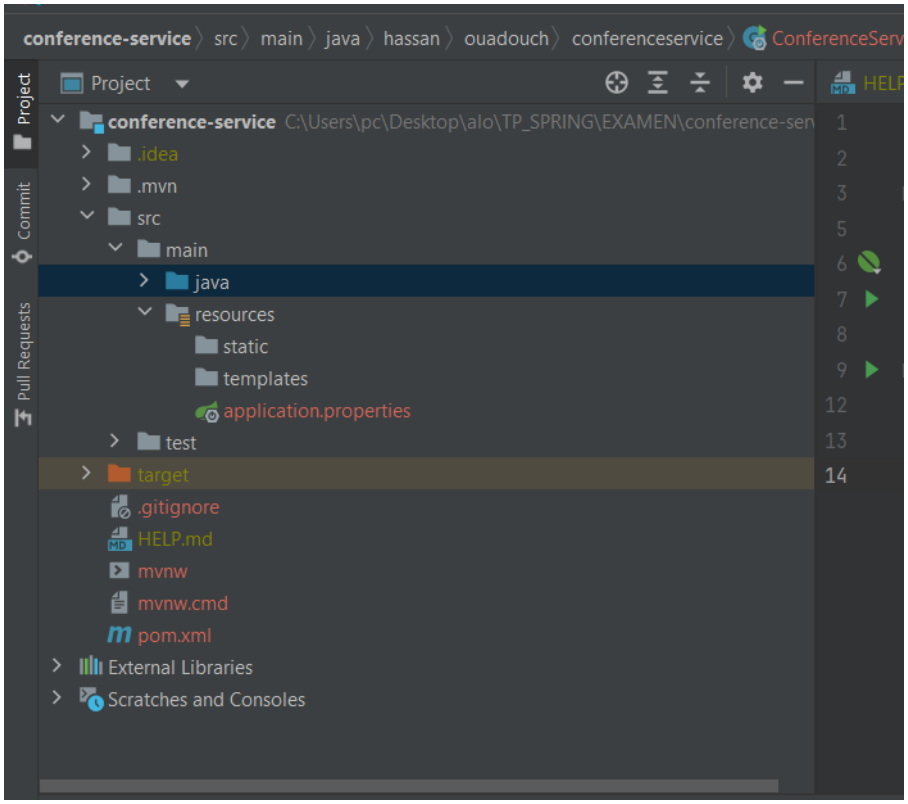


2. Créer un Projet Maven incluant les micro-services suivants

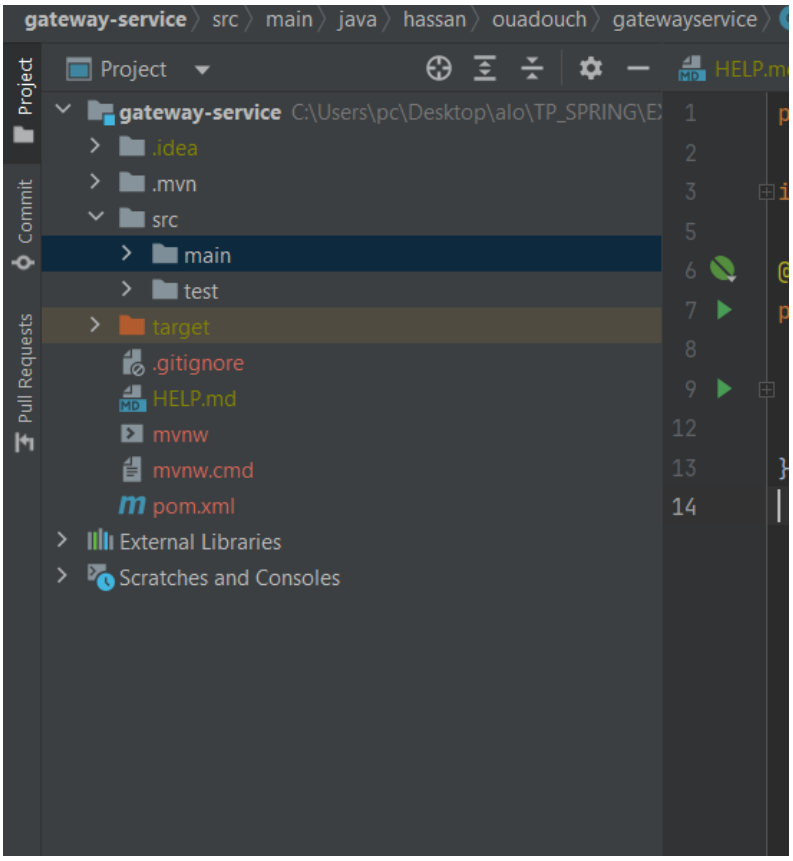
- keynote-service



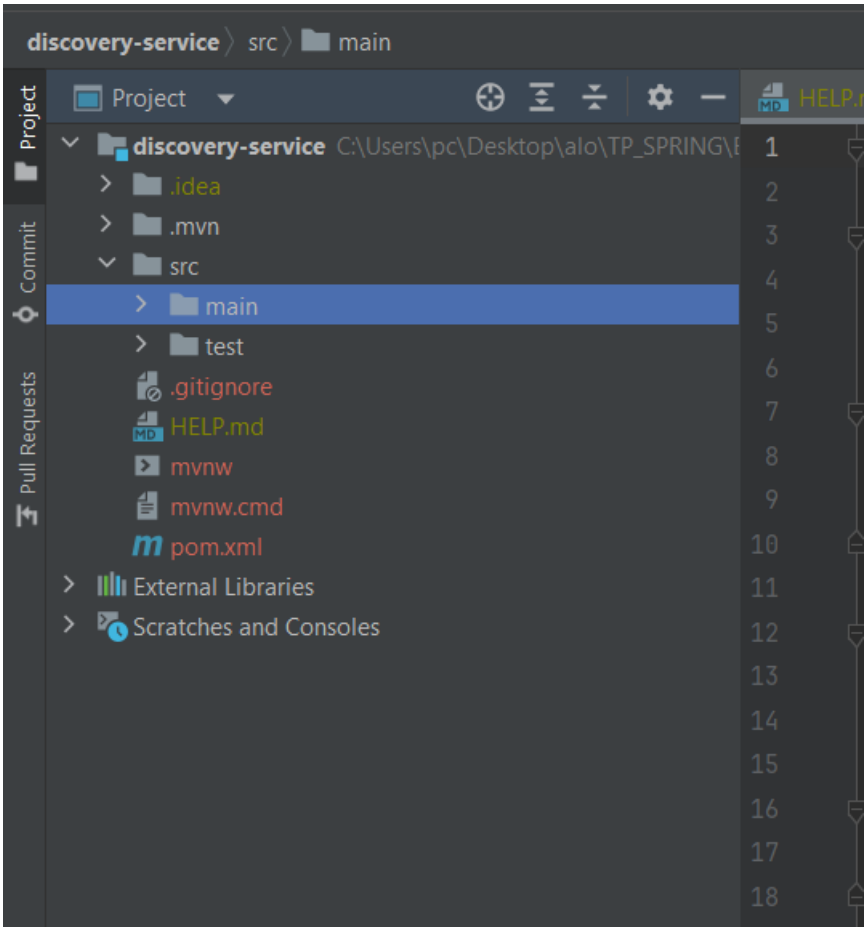
➢ conference-service



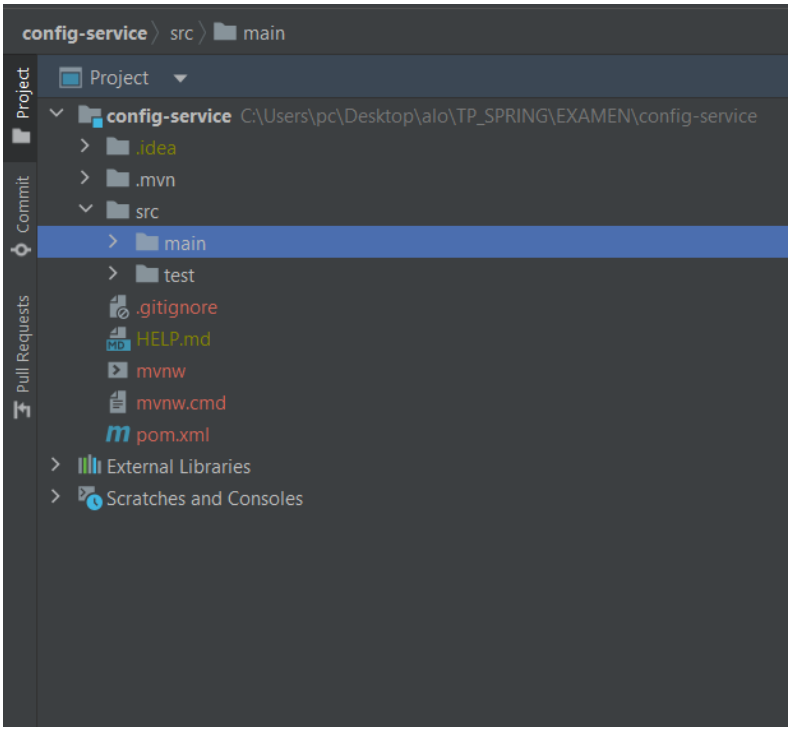
➢ gateway-service



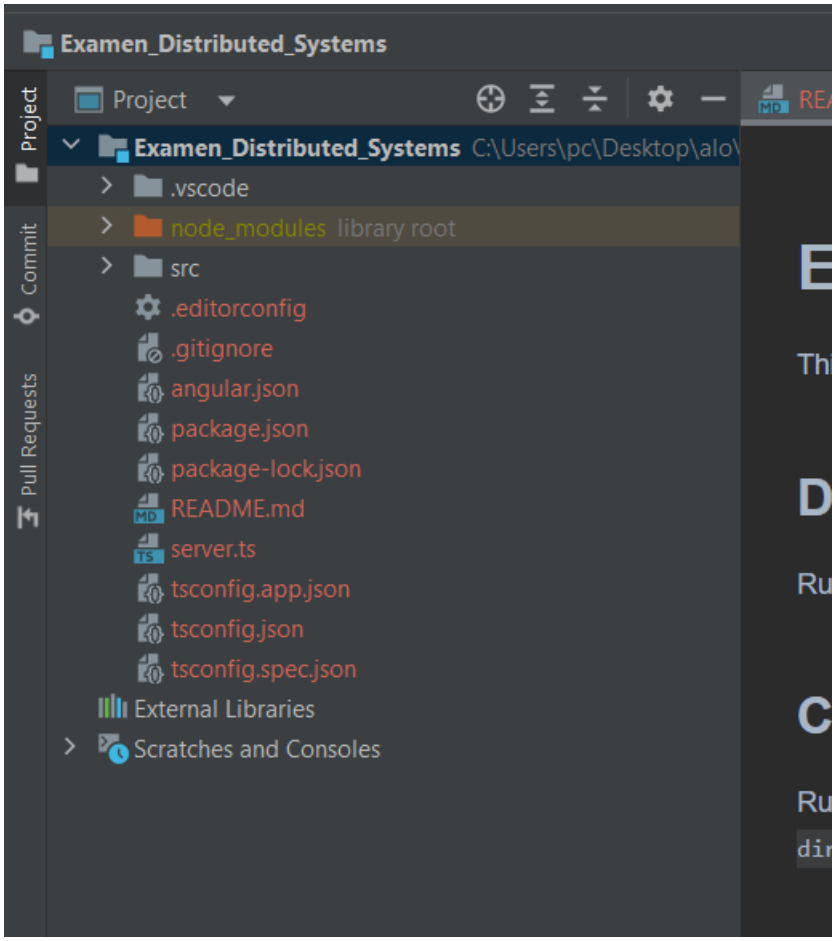
➢ discovery-service



➢ config-service



- angular-front-app



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

- Discovery-service

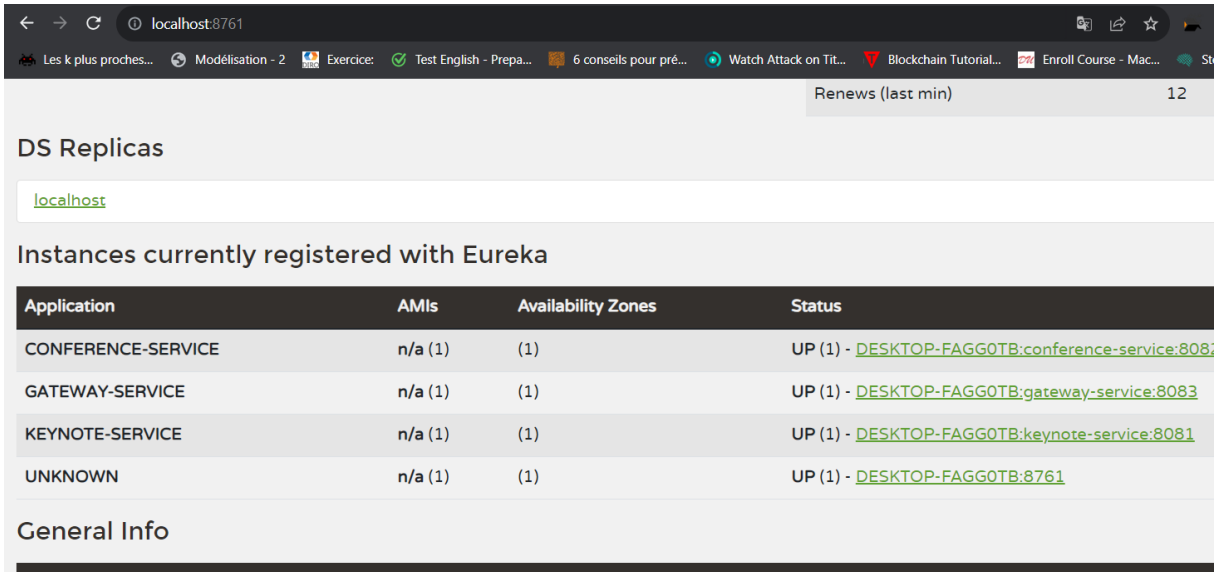
```
package Hassan.Ouadouch.discoveryservice;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServiceApplication {

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


}
```

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

- Gateway-service

```
package hassan.ouadouch.gateway;

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

@SpringBootApplication
public class GatewayApplication {

    public static void main(String[] args) {
        SpringApplication.run(GatewayApplication.class, args);
    }
    @Bean
    DiscoveryClientRouteDefinitionLocator dynamicRoutes(
            ReactiveDiscoveryClient rdc,
            DiscoveryLocatorProperties dlp){

        return new DiscoveryClientRouteDefinitionLocator(rdc,dlp);

    }


}
```

```
server.port=8083
spring.application.name=gateway-service
spring.cloud.discovery.enabled=true
```



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

- Entities

```
package Hassan.Ouadouch.keynoteservice.entities;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.ToString;
@Entity
@Data  @AllArgsConstructor  @NoArgsConstructor @ToString
    public class Keynote {
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;
        private String name;
        private String email;
    }
```

- DAO

```
package Hassan.Ouadouch.keynoteservice.DAO;

import Hassan.Ouadouch.keynoteservice.entities.Keynote;
import org.springframework.data.jpa.repository.JpaRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface KeynoteRepository  extends JpaRepository<Keynote,Long> {

}
```

- Service

```
package Hassan.Ouadouch.keynoteservice.service;

import Hassan.Ouadouch.keynoteservice.DAO.KeynoteRepository;
import Hassan.Ouadouch.keynoteservice.entities.Keynote;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class KeynoteService {
    @Autowired
    private KeynoteRepository keynoteRepository;

    public List<Keynote> getAllKeynotes() {
        return keynoteRepository.findAll();
    }

}
```

- DTO

```java
package Hassan.Ouadouch.keynoteservice.DTO;

import jakarta.persistence.Entity;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data @NoArgsConstructor  @AllArgsConstructor @Builder
public class KeynoteDTO {
    private Long id;
    private String name;
    private String email;
}
```

- Mapper

```java
package Hassan.Ouadouch.keynoteservice.Mapper;

import Hassan.Ouadouch.keynoteservice.DTO.KeynoteDTO;
import Hassan.Ouadouch.keynoteservice.entities.Keynote;
import org.springframework.stereotype.Component;

@Component
public class KeynoteMapper {
    public KeynoteDTO keynoteToKeynoteDTO(Keynote keynote) {
        KeynoteDTO keynoteDTO = new KeynoteDTO();
        keynoteDTO.setId(keynote.getId());
        keynoteDTO.setName(keynote.getName());
        keynoteDTO.setEmail(keynote.getEmail());
        return keynoteDTO;
    }

    public Keynote keynoteDTOToKeynote(KeynoteDTO keynoteDTO) {
        Keynote keynote = new Keynote();
        keynote.setId(keynoteDTO.getId());
        keynote.setName(keynoteDTO.getName());
        keynote.setEmail(keynoteDTO.getEmail());
        return keynote;
    }
}
```

- Web

```java
package Hassan.Ouadouch.keynoteservice.web;

import Hassan.Ouadouch.keynoteservice.DTO.KeynoteDTO;
import Hassan.Ouadouch.keynoteservice.Mapper.KeynoteMapper;
import Hassan.Ouadouch.keynoteservice.entities.Keynote;
import Hassan.Ouadouch.keynoteservice.service.KeynoteService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;
import java.util.stream.Collectors;

@RestController
@RequestMapping("/keynotes")
public class KeynoteController {
    @Autowired
    private KeynoteService keynoteService;

    @Autowired
    private KeynoteMapper keynoteMapper;

    @GetMapping("/")
    public List<KeynoteDTO> getAllKeynotes() {
        List<Keynote> keynotes = keynoteService.getAllKeynotes();
        return keynotes.stream()
                .map(keynoteMapper::keynoteToKeynoteDTO)
                .collect(Collectors.toList());
    }

}
```

```java
package Hassan.Ouadouch.keynoteservice;

import Hassan.Ouadouch.keynoteservice.DAO.KeynoteRepository;
import Hassan.Ouadouch.keynoteservice.entities.Keynote;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import
org.springframework.data.rest.core.config.RepositoryRestConfiguration;

import java.util.stream.Stream;

@SpringBootApplication
public class KeynoteServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(KeynoteServiceApplication.class, args);
    }
    @Bean
    CommandLineRunner start(KeynoteRepository keynoteRepository,
                            RepositoryRestConfiguration
repositoryRestConfiguration
    ){
        repositoryRestConfiguration.exposeIdsFor(Keynote.class);
        return args->{
```

```
            Stream.of("hassan", "abdo", "karima").forEach(name -> {

                keynoteRepository.save(new
Keynote(1L,name,name+"@gmal.com"));

            });

            keynoteRepository.findAll().forEach(System.out::println);
        };
    }

}
```

```
2023-12-25T12:06:01.551+01:00  INFO 14220 --- [keynote-service] [  restartedMain] com.netflix.discovery.DiscoveryClient    : Getting all instance registry info from the eureka se
2023-12-25T12:06:01.593+01:00  INFO 14220 --- [keynote-service] [  restartedMain] com.netflix.discovery.DiscoveryClient    : The response status is 200
2023-12-25T12:06:01.594+01:00  INFO 14220 --- [keynote-service] [  restartedMain] com.netflix.discovery.DiscoveryClient    : Starting heartbeat executor: renew interval is: 30
2023-12-25T12:06:01.595+01:00  INFO 14220 --- [keynote-service] [  restartedMain] c.n.discovery.InstanceInfoReplicator     : InstanceInfoReplicator onDemand update allowed rate p
2023-12-25T12:06:01.597+01:00  INFO 14220 --- [keynote-service] [  restartedMain] com.netflix.discovery.DiscoveryClient    : Discovery Client initialized at timestamp 17035023615
2023-12-25T12:06:01.615+01:00  INFO 14220 --- [keynote-service] [  restartedMain] o.s.c.n.e.s.EurekaServiceRegistry         : Registering application KEYNOTE-SERVICE with eureka w
2023-12-25T12:06:01.616+01:00  INFO 14220 --- [keynote-service] [  restartedMain] com.netflix.discovery.DiscoveryClient    : Saw local status change event StatusChangeEvent [time
2023-12-25T12:06:01.616+01:00  INFO 14220 --- [keynote-service] [nfoReplicator-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_KEYNOTE-SERVICE/DESKTOP-FAGGOTB:keyno
2023-12-25T12:06:01.624+01:00  INFO 14220 --- [keynote-service] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8081 (http) with context path
2023-12-25T12:06:01.625+01:00  INFO 14220 --- [keynote-service] [  restartedMain] .s.c.n.e.s.EurekaAutoServiceRegistration : Updating port to 8081
2023-12-25T12:06:01.635+01:00  INFO 14220 --- [keynote-service] [nfoReplicator-0] com.netflix.discovery.DiscoveryClient    : DiscoveryClient_KEYNOTE-SERVICE/DESKTOP-FAGGOTB:keyno
2023-12-25T12:06:01.652+01:00  INFO 14220 --- [keynote-service] [  restartedMain] H.O.k.KeynoteServiceApplication          : Started KeynoteServiceApplication in 4.844 seconds (p
Keynote(id=1, name=karima, email=karima@gmal.com)
2023-12-25T12:06:01.680+01:00  INFO 14220 --- [keynote-service] [  restartedMain] .ConditionEvaluationDeltaLoggingListener : Condition evaluation unchanged
```