**Département Mathématiques et Informatique**

**Compte rendu**

**Filière :**

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

**II-BDCC**

**Examen JEE**

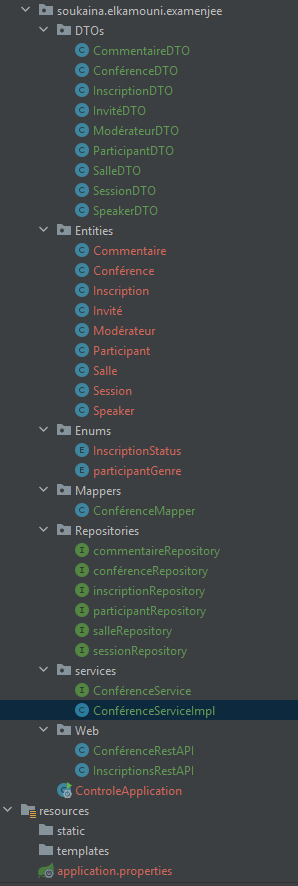
**le 23/05/2022**

Réalisé par :

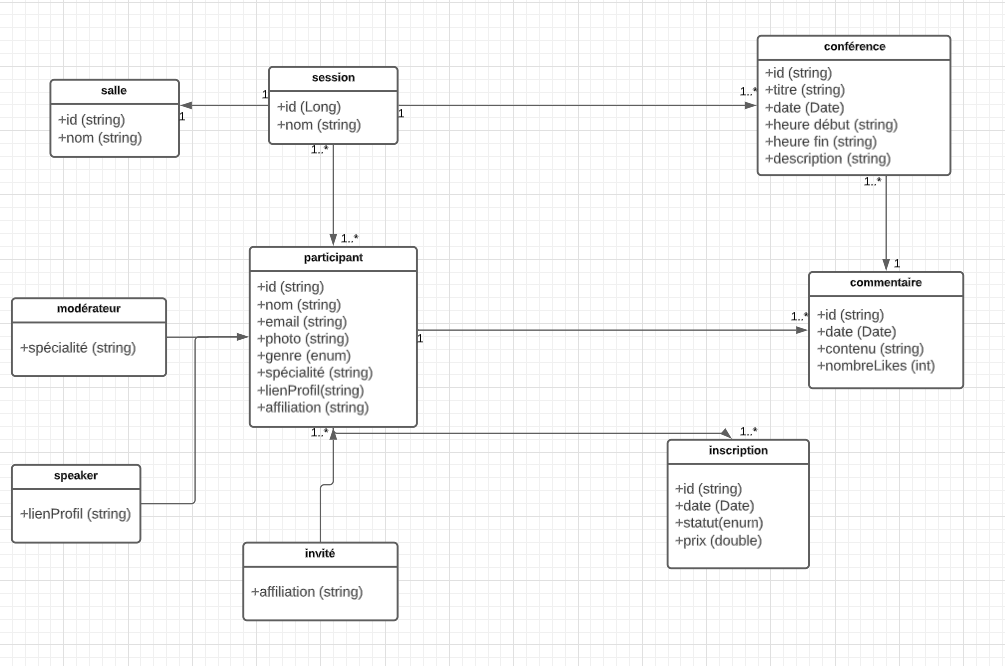
Soukaina EL KAMOUNI

**Année Universitaire : 2021-2022**

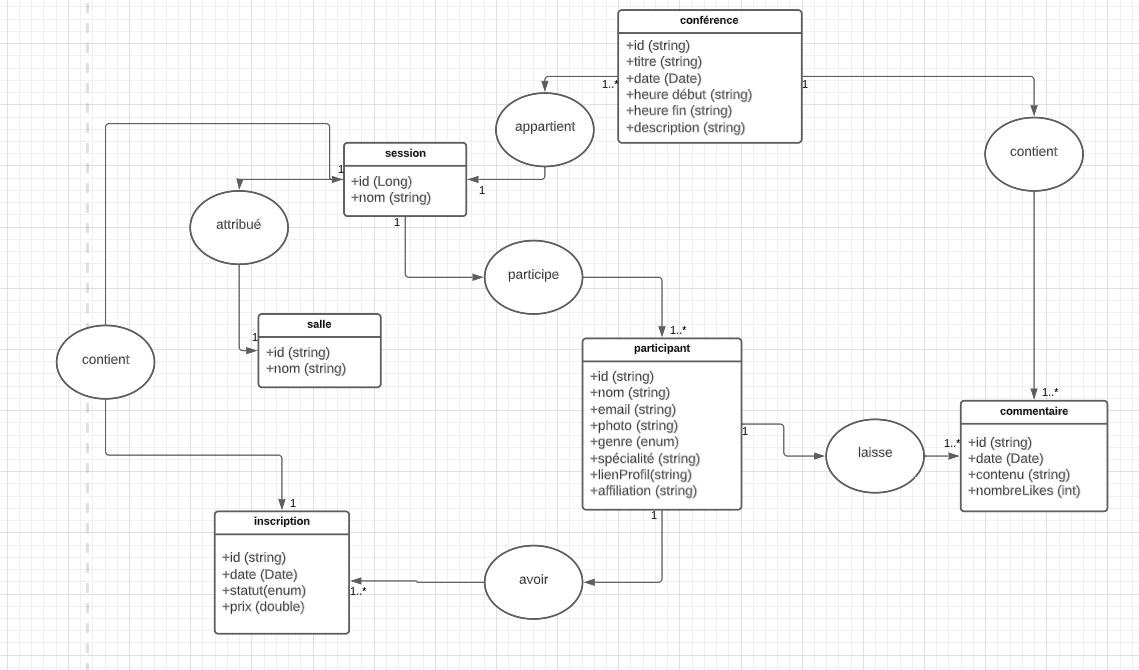
1. Conception :
   1. Architecture Technique du projet :



* 1. Diagramme de Classe :



* 1. Modèle logique de données :



1. Implémentation :
   1. Projet Settings :

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>2.7.0</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>soukaina.elkamouni</groupId>  
 <artifactId>examenJEEsoukainaElkamouni</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>ExamenJEE</name>  
 <description>ExamenJEE</description>  
 <properties>  
 <java.version>1.8</java.version>  
 </properties>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springdoc/springdoc-openapi-ui -->  
 <dependency>  
 <groupId>org.springdoc</groupId>  
 <artifactId>springdoc-openapi-ui</artifactId>  
 <version>1.6.8</version>  
 </dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

<dependency>

<groupId>org.webjars</groupId>

<artifactId>bootstrap</artifactId>

<version>5.1.3</version>

</dependency>  
  
 <!--<dependency>  
 <groupId>com.h2database</groupId>  
 <artifactId>h2</artifactId>  
 <scope>runtime</scope>  
 </dependency>-->  
 <dependency>  
 <groupId>org.thymeleaf.extras</groupId>  
 <artifactId>thymeleaf-extras-springsecurity5</artifactId>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/nz.net.ultraq.thymeleaf/thymeleaf-layout-dialect -->  
 <dependency>  
 <groupId>nz.net.ultraq.thymeleaf</groupId>  
 <artifactId>thymeleaf-layout-dialect</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 <configuration>  
 <excludes>  
 <exclude>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 </exclude>  
 </excludes>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

* 1. Couche DAO:
     + - Entité Commentaire:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.springframework.format.annotation.DateTimeFormat;  
  
import javax.persistence.\*;  
import java.util.Date;  
  
@Data  
@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
public class Commentaire {  
 @Id  
 private Long id;  
 @Temporal(TemporalType.*DATE*)  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 private Date date;  
 private String contenu;  
 private int nombreLikes;  
 @ManyToOne  
 private Conférence conférence;  
}

* + - * Entité Conférence:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.springframework.format.annotation.DateTimeFormat;  
  
import javax.persistence.\*;  
import java.util.Date;  
import java.util.List;  
  
@Data  
@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
public class Conférence {  
 @Id  
 private String id;  
 private String titre;  
 @Temporal(TemporalType.*DATE*)  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 private Date date;  
 @DateTimeFormat(pattern = "hh:mm:ss")  
 private Date heureDébut;  
 @DateTimeFormat(pattern = "hh:mm:ss")  
 private Date heureFin;  
 private String description;  
 @ManyToOne  
 private Speaker speaker;  
 @ManyToOne  
 private Session session;  
 @OneToMany(mappedBy = "conférence",fetch = FetchType.*LAZY*)  
 private List<Commentaire> commentaireList;  
 @ManyToMany(mappedBy = "conférenceList",fetch = FetchType.*LAZY*)  
 private List<Participant> participantList;  
}

* + - * Entité Inscription:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import org.springframework.format.annotation.DateTimeFormat;  
import soukaina.elkamouni.examenjee.Enums.InscriptionStatus;  
  
import javax.persistence.\*;  
import java.util.Date;  
import java.util.List;  
  
@Data  
@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
public class Inscription {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 @Temporal(TemporalType.*DATE*)  
 @DateTimeFormat(pattern = "yyyy-MM-dd")  
 private Date date;  
 @Enumerated(EnumType.*STRING*)  
 private InscriptionStatus statut;  
 private Double prix;  
 @ManyToOne  
 private Invité invité;  
 @ManyToOne  
 private Session session;  
  
}

* + - * Entité Modérateur:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.DiscriminatorValue;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.OneToMany;  
import java.util.List;  
  
@Entity  
@Data  
@DiscriminatorValue("MODERATEUR")  
@AllArgsConstructor  
@NoArgsConstructor  
public class Modérateur extends Participant {  
 private String specialite;  
 @OneToMany(mappedBy = "modérateur",fetch = FetchType.*LAZY*)  
 private List<Session> sessions;  
}

* + - * Entité Participant:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
import soukaina.elkamouni.examenjee.Enums.participantGenre;  
  
import javax.persistence.\*;  
import java.util.List;  
  
@Inheritance(strategy = InheritanceType.*SINGLE\_TABLE*)  
@DiscriminatorColumn(name = "TYPE", length = 10)  
@Data  
@Entity  
@AllArgsConstructor  
@NoArgsConstructor  
public class Participant {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
 private String nom;  
 private String email;  
 private String photo;  
 @Enumerated(EnumType.*STRING*)  
 private participantGenre genre;  
 @ManyToMany  
 private List<Conférence> conférenceList;  
}

* + - * Entité Salle:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.ManyToOne;  
import java.util.List;  
  
@Entity  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class Salle {  
 @Id  
 private Long id ;  
 private String nom;  
  
}

* + - * Entité Session:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.\*;  
import java.util.List;  
  
@Entity  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
public class Session {  
 @Id  
 private String id;  
 private String nom;  
 @OneToMany(mappedBy = "session",fetch = FetchType.*LAZY*)  
 private List<Conférence> conférences;  
 @ManyToOne  
 private Modérateur modérateur;  
 @OneToMany(mappedBy = "session",fetch = FetchType.*LAZY*)  
 private List<Inscription> inscriptionList;  
}

* + - * Entité Speaker:

package soukaina.elkamouni.examenjee.Entities;  
  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
import javax.persistence.DiscriminatorValue;  
import javax.persistence.Entity;  
import javax.persistence.FetchType;  
import javax.persistence.OneToMany;  
import java.util.List;  
  
@Entity  
@Data  
@DiscriminatorValue("SPEAKER")  
@AllArgsConstructor  
@NoArgsConstructor  
public class Speaker extends Participant {  
 private String profilPersonnel;  
 @OneToMany(mappedBy = "speaker",fetch = FetchType.*LAZY*)  
 private List<Conférence> conférences;  
}

* + - * Les enums:

package soukaina.elkamouni.examenjee.Enums;  
public enum InscriptionStatus {  
 *EnCours*, *Validée*,*Annulée*}

package soukaina.elkamouni.examenjee.Enums;  
  
public enum participantGenre {  
  
 *MASCULIN*,*FEMININ*}

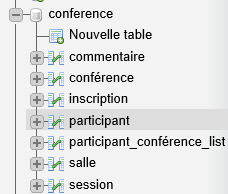
* + - * Répository simulaire à celui-là :

package soukaina.elkamouni.examenjee.Repositories;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import soukaina.elkamouni.examenjee.Entities.Conférence;  
  
@Repository  
public interface conférenceRepository extends JpaRepository<Conférence,String> {  
}

* + - * Application Main :

package soukaina.elkamouni.examenjee;  
  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.annotation.Bean;  
import soukaina.elkamouni.examenjee.Entities.\*;  
import soukaina.elkamouni.examenjee.Enums.InscriptionStatus;  
import soukaina.elkamouni.examenjee.Enums.participantGenre;  
import soukaina.elkamouni.examenjee.Repositories.\*;  
  
import java.util.Date;  
import java.util.UUID;  
import java.util.stream.Stream;  
  
@SpringBootApplication  
public class ControleApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(ControleApplication.class, args);  
 }  
 @Bean  
 CommandLineRunner commandLineRunner(participantRepository participantRepository,  
 salleRepository salleRepository,  
 sessionRepository sessionRepository,  
 inscriptionRepository inscriptionRepository,  
 conférenceRepository conférenceRepository){  
 return args -> {  
  
  
 Stream.*of*("soukaina", "salma").forEach(name -> {  
 Speaker speaker=new Speaker();  
 speaker.setNom(name);  
 speaker.setEmail(name + "@gmail.com");  
 speaker.setGenre(Math.*random*()>0.5? participantGenre.*MASCULIN*:participantGenre.*FEMININ*);  
 speaker.setProfilPersonnel("speaker");  
 participantRepository.save(speaker);  
 });  
  
 Stream.*of*("salle1", "salle2").forEach(name -> {  
 Salle salle=new Salle();  
 salle.setNom(name);  
 salleRepository.save(salle);  
 });  
  
 salleRepository.findAll().forEach(acc->{  
 for (int i = 0; i < 5; i++) {  
 Session session=new Session();  
 session.setId(UUID.*randomUUID*().toString());  
 session.setNom("session"+i);  
 sessionRepository.save(session);  
 }  
 });  
  
  
 sessionRepository.findAll().forEach(session->{  
 for (int i = 0; i < 5; i++) {  
 Inscription inscription= new Inscription();  
 inscription.setDate(new Date());  
 inscription.setPrix(Math.*random*()\*1000);  
 inscription.setStatut(InscriptionStatus.*EnCours*);  
 inscriptionRepository.save(inscription);  
 }  
  
 for (int i = 0; i < 5; i++) {  
 Conférence conférence=new Conférence();  
 conférence.setId(UUID.*randomUUID*().toString());  
 conférence.setDate(new Date());  
 conférence.setHeureDébut(new Date());  
 conférence.setHeureFin(new Date());  
 conférence.setDescription("conference"+i);  
 conférenceRepository.save(conférence);  
 }  
  
 conférenceRepository.findAll().forEach(acc->{  
 for (int i = 0; i < 5; i++) {  
 Stream.*of*("mohammed", "hassan").forEach(name -> {  
 Modérateur modérateur=new Modérateur();  
 modérateur.setNom(name);  
 modérateur.setEmail(name + "@gmail.com");  
 modérateur.setGenre(participantGenre.*MASCULIN*);  
 modérateur.setSpecialite("moderateur");  
 participantRepository.save(modérateur);  
 });  
 }  
 });  
 });  
  
  
 };  
 }  
  
  
}

* + - * La base de données:



* 1. Couche Web REST API:
     + - Mappeur:

package soukaina.elkamouni.examenjee.Mappers;  
  
import org.springframework.beans.BeanUtils;  
import org.springframework.stereotype.Service;  
import soukaina.elkamouni.examenjee.DTOs.\*;  
import soukaina.elkamouni.examenjee.Entities.\*;  
  
@Service  
public class ConférenceMapper {  
  
 public InvitéDTO fromInvité(Invité invité){  
 InvitéDTO invitéDTO = new InvitéDTO();  
 BeanUtils.*copyProperties*(invité, invitéDTO);  
 invitéDTO.setType(invité.getClass().getSimpleName());  
 return invitéDTO;  
 }  
  
 public Invité fromInvitéDTO(InvitéDTO invitéDTO){  
 Invité invité = new Invité();  
 BeanUtils.*copyProperties*(invitéDTO, invité);  
 return invité;  
 }  
  
 public ModérateurDTO fromModérateur(Modérateur modérateur){  
 ModérateurDTO modérateurDTO = new ModérateurDTO();  
 BeanUtils.*copyProperties*(modérateur, modérateurDTO);  
 modérateurDTO.setType(modérateur.getClass().getSimpleName());  
 return modérateurDTO;  
 }  
  
 public Modérateur fromModérateurDTO(ModérateurDTO modérateurDTO){  
 Modérateur modérateur = new Modérateur();  
 BeanUtils.*copyProperties*(modérateurDTO, modérateur);  
 return modérateur;  
 }  
  
 public SpeakerDTO fromSpeaker(Speaker speaker){  
 SpeakerDTO speakerDTO = new SpeakerDTO();  
 BeanUtils.*copyProperties*(speaker, speakerDTO);  
 speakerDTO.setType(speaker.getClass().getSimpleName());  
 return speakerDTO;  
 }  
  
 public Speaker fromSpeakerDTO(SpeakerDTO speakerDTO){  
 Speaker speaker= new Speaker();  
 BeanUtils.*copyProperties*(speakerDTO, speaker);  
 return speaker;  
 }  
  
 public SessionDTO fromSession(Session session){  
 SessionDTO sessionDTO = new SessionDTO();  
 BeanUtils.*copyProperties*(session, sessionDTO);  
 return sessionDTO;  
 }  
  
 public Session fromSessionDTO(SessionDTO sessionDTO){  
 Session session= new Session();  
 BeanUtils.*copyProperties*(sessionDTO, session);  
 return session;  
 }  
  
 public ConférenceDTO fromConférence(Conférence conférence){  
 ConférenceDTO conférenceDTO = new ConférenceDTO();  
 BeanUtils.*copyProperties*(conférence,conférenceDTO);  
 conférenceDTO.setSpeaker(fromSpeaker(conférence.getSpeaker()));  
 return conférenceDTO;  
 }  
  
 public Conférence fromConférenceDTO(ConférenceDTO conférenceDTO){  
 Conférence conférence = new Conférence();  
 BeanUtils.*copyProperties*(conférenceDTO, conférence);  
 conférence.setSpeaker(fromSpeakerDTO(conférenceDTO.getSpeaker()));  
 return conférence;  
 }  
}

* + - * Conférence REST API:

package soukaina.elkamouni.examenjee.Web;  
  
import lombok.AllArgsConstructor;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RestController;  
import soukaina.elkamouni.examenjee.DTOs.CommentaireDTO;  
import soukaina.elkamouni.examenjee.DTOs.ConférenceDTO;  
import soukaina.elkamouni.examenjee.DTOs.ParticipantDTO;  
import soukaina.elkamouni.examenjee.DTOs.SessionDTO;  
import soukaina.elkamouni.examenjee.Entities.Session;  
import soukaina.elkamouni.examenjee.services.ConférenceServiceImpl;  
  
import java.util.List;  
  
@RestController  
@AllArgsConstructor  
public class ConférenceRestAPI {  
 private ConférenceServiceImpl conférenceService;  
  
  
 @GetMapping("/conferences/{conferenceId}")  
 public ConférenceDTO getConférence(@PathVariable String conferenceId){  
 return conférenceService.getConféranceDTO(conferenceId);  
 }  
  
 @GetMapping("/conferences")  
 public List<ConférenceDTO> listConférences(){  
 return conférenceService.listConférence();  
 }  
 @GetMapping("/participants")  
 public List<ParticipantDTO> listParticipant(){  
 return conférenceService.listParticipant();  
 }  
  
 @GetMapping("/participants/{participantId}")  
 public ParticipantDTO getParticipant(@PathVariable Long participantId){  
 return conférenceService.getParticipant(participantId);  
 }  
  
 @GetMapping("/conferences/{conferenceId}/commentaires")  
 public List<CommentaireDTO> listCommentaires(){  
 return conférenceService.listCommentaires();  
 }  
  
 @GetMapping("/sessions")  
 public List<SessionDTO> listSessions(){  
 return conférenceService.listSessions();  
 }  
  
 @GetMapping("/sessions/{sessionId}")  
 public SessionDTO getSession(@PathVariable String sessionId){  
 return conférenceService.getSession(sessionId);  
 }  
}

* + - * Inscription REST API:

1. package soukaina.elkamouni.examenjee.Web;  
     
     
   import lombok.AllArgsConstructor;  
   import org.springframework.web.bind.annotation.GetMapping;  
   import org.springframework.web.bind.annotation.PathVariable;  
   import org.springframework.web.bind.annotation.RestController;  
   import soukaina.elkamouni.examenjee.DTOs.InscriptionDTO;  
   import soukaina.elkamouni.examenjee.services.ConférenceServiceImpl;  
     
   import java.util.List;  
     
   @RestController  
   @AllArgsConstructor  
   public class InscriptionsRestAPI {  
    private ConférenceServiceImpl conférenceService;  
     
    @GetMapping("/inscriptions/{inscriptionId}")  
    public InscriptionDTO getInscription(@PathVariable String inscriptionId){  
    return conférenceService.getInscriptionDTO(inscriptionId);  
    }  
     
    @GetMapping("/inscriptions")  
    public List<InscriptionDTO> listInscriptions(){  
    return conférenceService.listInscriptions();  
    }  
   }
   * + - Conférence Service:

package soukaina.elkamouni.examenjee.services;  
  
import soukaina.elkamouni.examenjee.DTOs.\*;  
import soukaina.elkamouni.examenjee.Entities.\*;  
  
import java.util.List;  
  
public interface ConférenceService {  
 ConférenceDTO getConféranceDTO(String id);  
 List<ConférenceDTO> listConférence();  
 ConférenceDTO updateConférence(String id);  
 ConférenceDTO deleteConférence(String id);  
  
 List<ParticipantDTO> listParticipant();  
 InvitéDTO saveInvité(Invité invité);  
 ModérateurDTO saveModérateur(Modérateur modérateur);  
 SpeakerDTO saveSpeaker(Speaker speaker);  
 ParticipantDTO getParticipant(Long id);  
 SpeakerDTO getSpeakerDTO();  
 ModérateurDTO getModérateur();  
 InvitéDTO getInvité();  
 ParticipantDTO updateParticipant(Long id);  
 ParticipantDTO deleteParticipant(Long id);  
 List<CommentaireDTO> listCommentaires();  
  
 List<SessionDTO> listSessions();  
 SessionDTO getSession(String id);  
  
 InscriptionDTO getInscriptionDTO(String inscriptionId);  
  
 List<InscriptionDTO> listInscriptions();  
}

1. Sécurity: