جـامعــة الحسـن الثـانـي بالدار البيـضـاء
ⵜⴰⵙⴷⴰⵡⵉⵜ ⵏ ⵃⴰⵙⵙⴰⵏ ⵜⵉⵙ ⵙⵉⵏ ⴳ ⴰⵏⴼⴰ
UNIVERSITÉ HASSAN II DE CASABLANCA

Département Mathématiques et Informatique

Filière :

« Génie Logiciel et Systèmes Informatiques Distribués »

Examen Architecture JEE et Middlewares

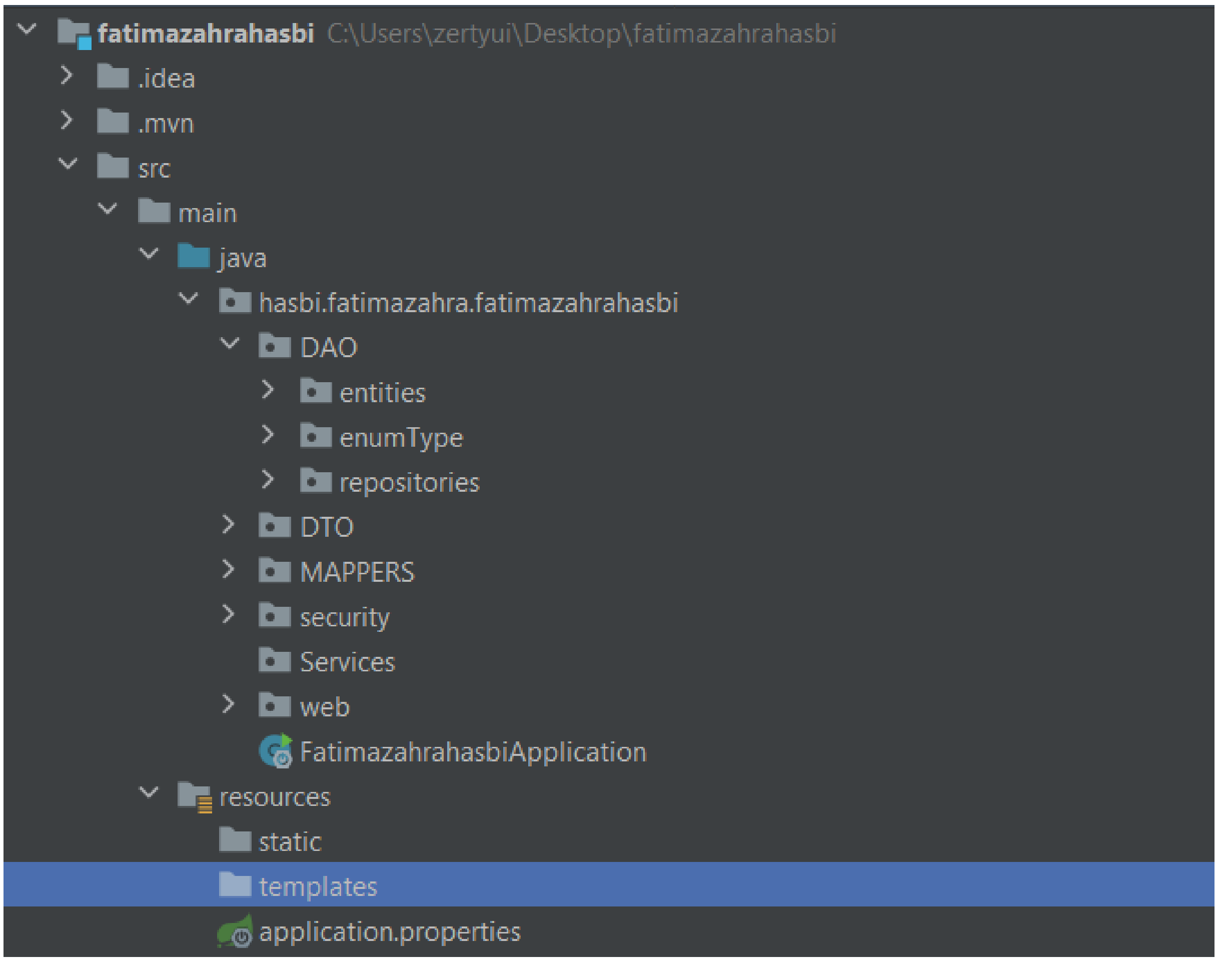
Fatima Zahra HASBI

Professeur : Mr Mohamed Youssfi

Année Universitaire : 2021-2022

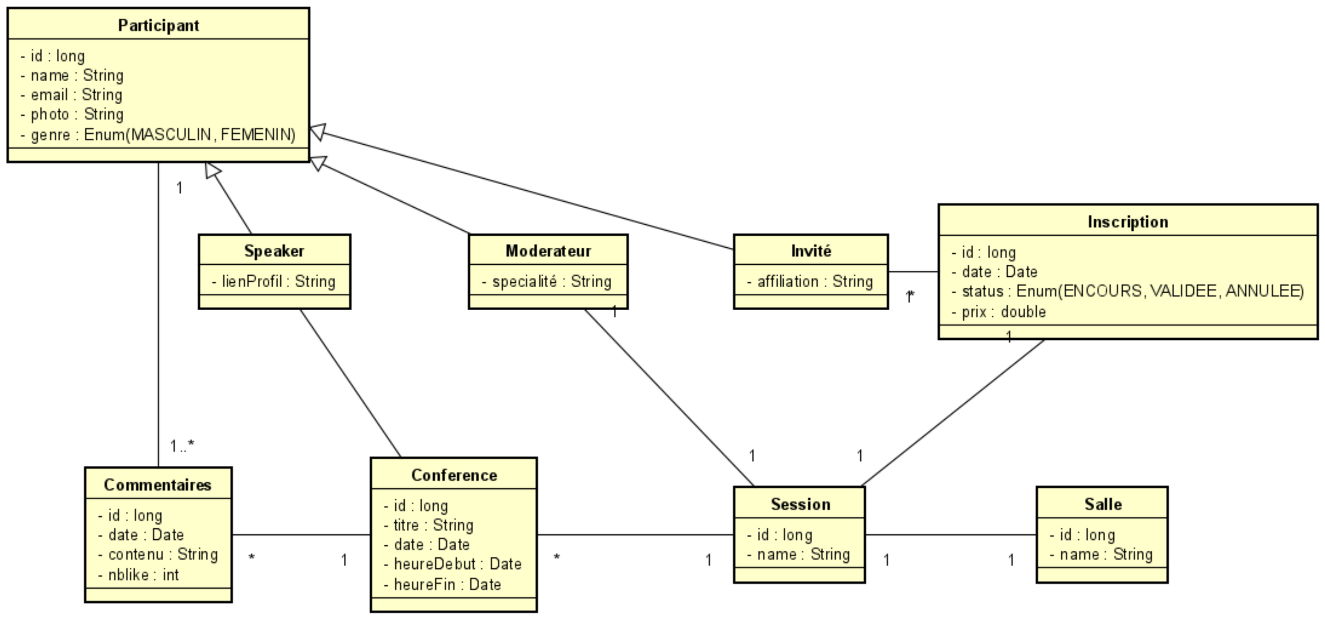
A. Conception :

1. Etablir une architecture technique du projet



```
fatimazahrahasbi C:\Users\zertyui\Desktop\fatimazahrahasbi
  .idea
  .mvn
  src
    main
      java
        hasbi.fatimazahra.fatimazahrahasbi
          DAO
            entities
            enumType
            repositories
          DTO
          MAPPERS
          security
          Services
          web
          FatimazahrahasbiApplication
      resources
        static
        templates
        application.properties
```

2. Etablir un diagramme de classes qui montre les entités. On ne représentera que les attributs.



3. Etablir un Modèle Logique de données relationnel en adoptant la stratégie Single Table pour l'héritage.

- Participant (idParticipant, name, email, photo, genre, typeParticipant, lienProfil, specialite, affiliation, #idConf)

- Inscription (id, date, status, prix, #idParticipant, #idSession)

- Session (idSession, name, #idParticipant, #idSalle)

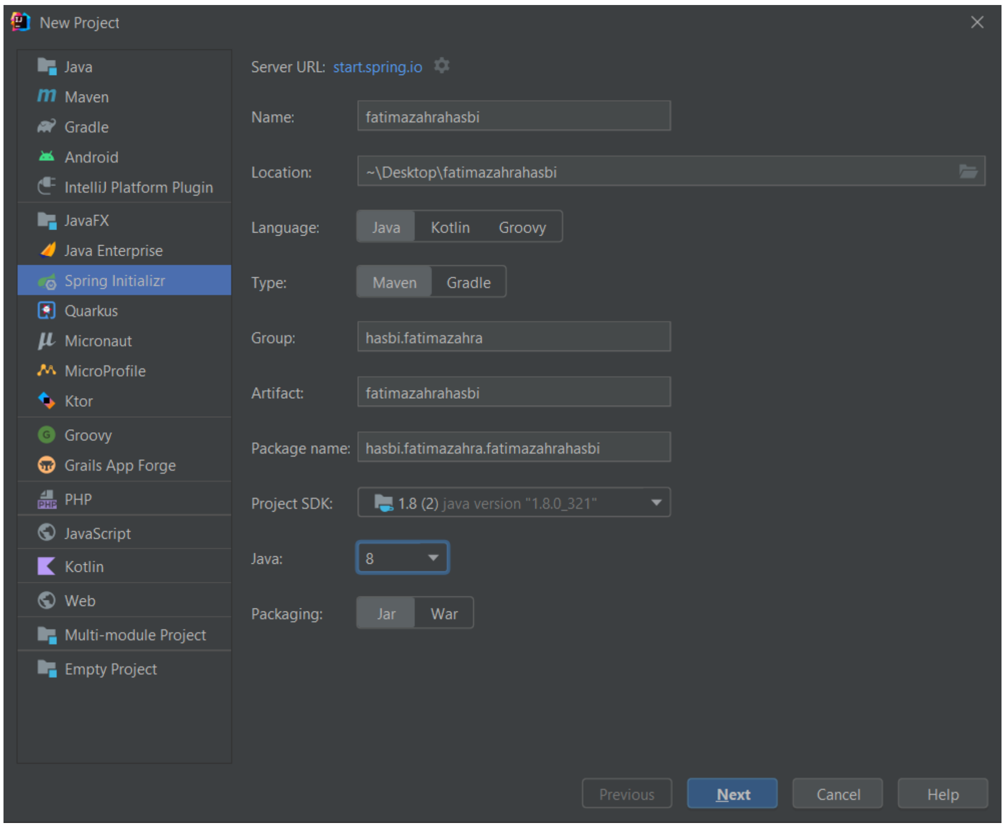
- Salle (idSalle, name, #idSession)

- Conference (idConf, titre, date, heureDebut, heureFin, #idParticipant, #idSession)

- Commentaire (id, date, contenu, nblike, #idParticipant, #idConference)

B. Implémentation :

1. Créer un projet Spring boot avec les dépendances requises. Les identifiants du projet : GroupId, ArtifactId et le package de base doivent contenir votre nom et prénom.


New Project

Server URL: start.spring.io

Name: fatimazahrahasbi

Location: ~\Desktop\fatimazahrahasbi

Language: Java Kotlin Groovy

Type: Maven Gradle

Group: hasbi.fatimazahra

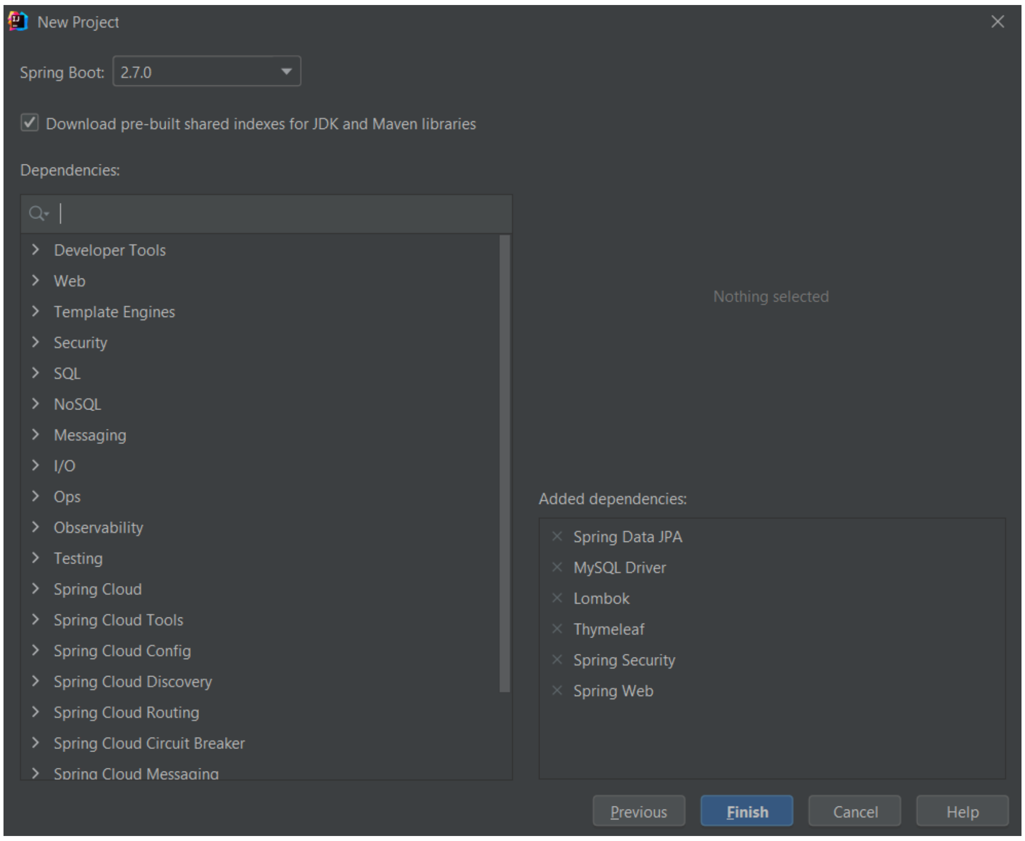
Artifact: fatimazahrahasbi

Package name: hasbi.fatimazahra.fatimazahrahasbi

Project SDK: 1.8 (2) java version "1.8.0_321"

Java: 8

Packaging: Jar War


New Project

Spring Boot: 2.7.0

Download pre-built shared indexes for JDK and Maven libraries

Dependencies:

Nothing selected

Added dependencies:

- Spring Data JPA
- MySQL Driver
- Lombok
- Thymeleaf
- Spring Security
- Spring Web

2. Couche DAO

a) Créer les entités JPA

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;
@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Commentaire {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    private String contenu;
    private int nblike;
    @ManyToOne
    private Participant participant;
    @ManyToOne
    private Conference conference;
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Conference {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    private String heureDebut;
    private String heureFin;
    @ManyToOne
    private Participant speaker;
    @ManyToOne
    private Session session;
    @OneToMany(mappedBy = "conference",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Commentaire> commentaires = new ArrayList<>();
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.enumType.Status;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.Date;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Inscription {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Date date;
    @Enumerated(EnumType.STRING)
    private Status status;
    private double prix;
    @ManyToOne
    private Participant invite;
    @ManyToOne
    private Session session;
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import hasbi.fatimazahra.fatimazahrahasbi.DAO.enumType.Genre;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "TYPE",length = 15)
@Data @NoArgsConstructor @AllArgsConstructor
public abstract class Participant {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
    private String photo;
    @Enumerated(EnumType.STRING)
    private Genre genre;
    private String lienProfil;
    private String specialite;
    private String affiliation;
    @OneToMany(mappedBy = "speaker",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Conference> conferences = new ArrayList<>();
    @OneToMany(mappedBy = "participant",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Commentaire> commentaires = new ArrayList<>();
    @OneToMany(mappedBy = "invite",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Inscription> inscriptions = new ArrayList<>();
    @OneToMany(mappedBy = "moderateur",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Session> sessions = new ArrayList<>();
}
```

```
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("Invite")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Invite extends Participant{
    private String affiliation;
}
```

```
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("Moderateur")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Moderateur extends Participant{
    private String specialite;
}
```

```
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.DiscriminatorValue;
import javax.persistence.Entity;

@Entity
@DiscriminatorValue("Speaker")
@Data
@NoArgsConstructor
```

```java
@AllArgsConstructor
public class Speaker extends Participant{
    private String lienProfil;
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Salle {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @OneToMany(mappedBy = "salle",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Session> sessions = new ArrayList<>();
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.entities;

import com.fasterxml.jackson.annotation.JsonProperty;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Data @NoArgsConstructor @AllArgsConstructor
public class Session {
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    @ManyToOne
    private Participant moderateur;
    @ManyToOne
    private Salle salle;
    @OneToMany(mappedBy = "session",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Inscription> inscriptions = new ArrayList<>();
    @OneToMany(mappedBy = "session",fetch = FetchType.LAZY)
    @JsonProperty(access = JsonProperty.Access.WRITE_ONLY)
    private List<Conference> conferences = new ArrayList<>();
}
```

b) Créer les interfaces JPA Repository basées sur Spring Data

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Commentaire;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CommentRepository extends JpaRepository<Commentaire, Long> {
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Conference;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ConferenceRepository extends JpaRepository<Conference, Long> {
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Inscription;
import org.springframework.data.jpa.repository.JpaRepository;

public interface InscriptionRepository extends
JpaRepository<Inscription, Long> {
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Participant;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ParticipantRepository extends
JpaRepository<Participant, Long> {
}
```

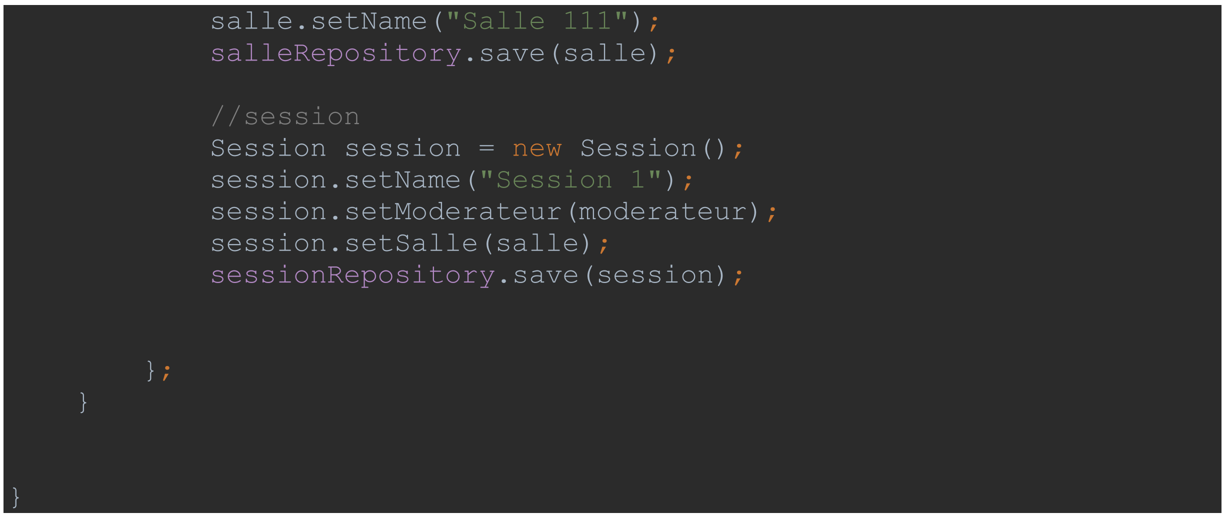
```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Salle;
import org.springframework.data.jpa.repository.JpaRepository;

public interface SalleRepository extends JpaRepository<Salle, Long> {
}
```

```java
package hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Session;
import org.springframework.data.jpa.repository.JpaRepository;
```

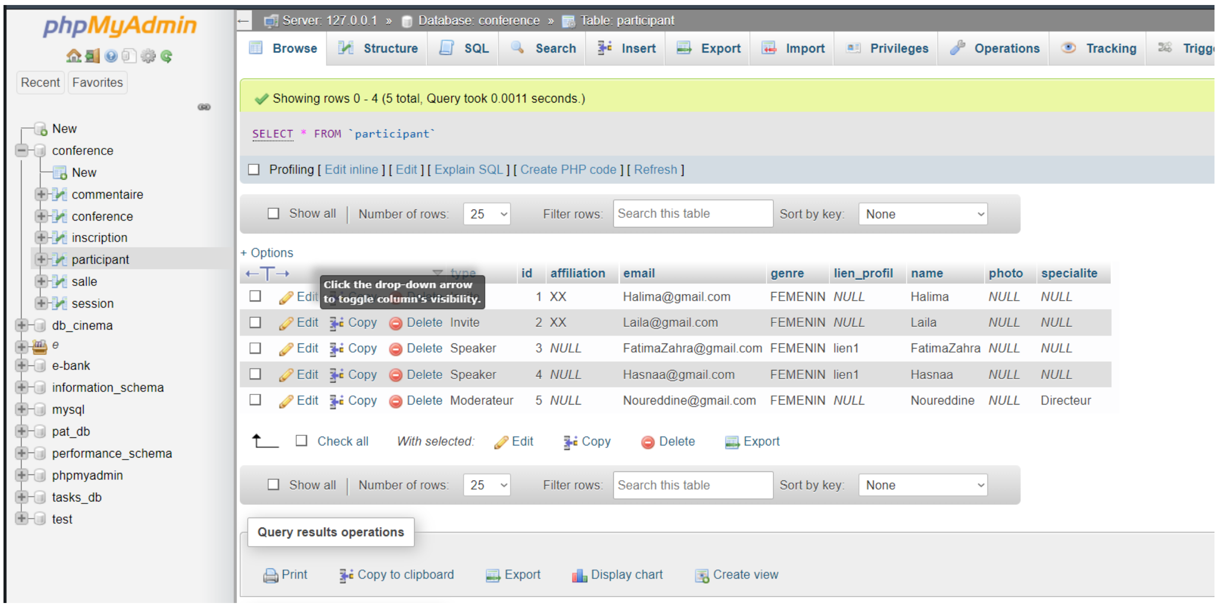
```java
public interface SessionRepository extends JpaRepository<Session, Long>
{
}
```

c) Tester la couche DAO avec une application qui alimente la base de données avec quelques enregistrements de test.

```java
@SpringBootApplication
public class FatimazahrahasbiApplication {

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

    @Bean
    CommandLineRunner start(
            ParticipantRepository participantRepository,
            SalleRepository salleRepository,
            InscriptionRepository inscriptionRepository,
            SessionRepository sessionRepository){
        return args -> {

//              //inscription
//              Inscription inscription = new Inscription();
//              inscription.setDate(new Date());
//              inscription.setPrix(300);
//              inscription.setStatus(Status.VALIDEE);
//              inscriptionRepository.save(inscription);
//              inscriptionRepository.save(inscription);
            //creation des participants
            //invite
            Stream.of("Halima","Laila").forEach(name->{
                Invite invite=new Invite();
                invite.setName(name);
                invite.setEmail(name+"@gmail.com");
                invite.setGenre(Genre.FEMENIN);
                invite.setAffiliation(Math.random()<1?"XX":"YY");
                participantRepository.save(invite);
            });
            //speaker
                Stream.of("FatimaZahra","Hasnaa").forEach(name->{
                Speaker speaker=new Speaker();
                speaker.setName(name);
                speaker.setEmail(name+"@gmail.com");
                speaker.setGenre(Genre.FEMENIN);
                speaker.setLienProfil(Math.random()<1?"lien1":"lien2");
                participantRepository.save(speaker);
            });
            //Moderateur
                Moderateur moderateur=new Moderateur();
                moderateur.setName("Noureddine");
                moderateur.setEmail("Noureddine@gmail.com");
                moderateur.setGenre(Genre.FEMENIN);
                moderateur.setSpecialite("Directeur");
                participantRepository.save(moderateur);
            //Salle
            Salle salle = new Salle();
```



```
            salle.setName("Salle 111");
            salleRepository.save(salle);

            //session
            Session session = new Session();
            session.setName("Session 1");
            session.setModerateur(moderateur);
            session.setSalle(salle);
            sessionRepository.save(session);


        };
    }

}
```



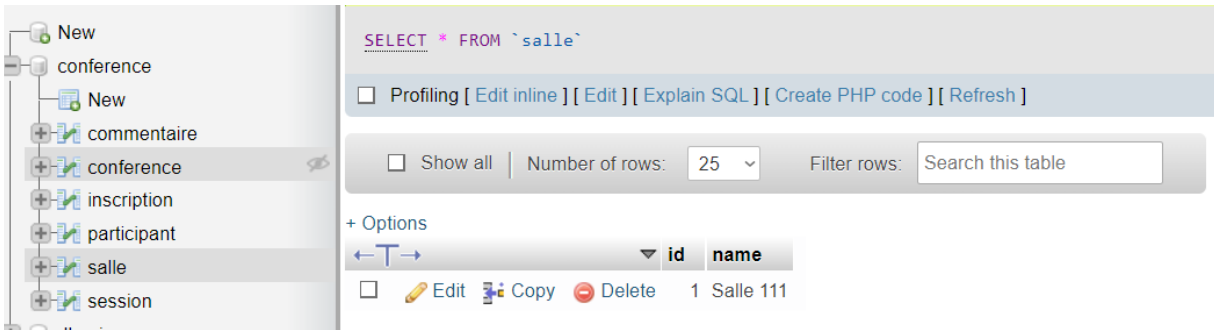
Server: 127.0.0.1 » Database: conference » Table: participant

Showing rows 0 - 4 (5 total, Query took 0.0011 seconds.)

```
SELECT * FROM `participant`
```

Click the drop-down arrow to toggle column's visibility.

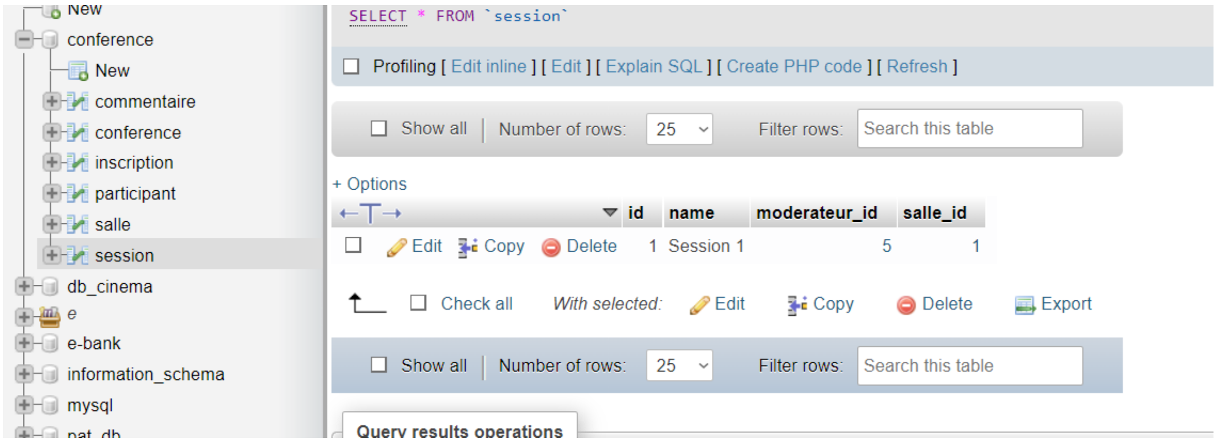
				type	id	affiliation	email	genre	lien_profil	name	photo	specialite
☐	Edit				1	XX	Halima@gmail.com	FEMENIN	NULL	Halima	NULL	NULL
☐	Edit	Copy	Delete	Invite	2	XX	Laila@gmail.com	FEMENIN	NULL	Laila	NULL	NULL
☐	Edit	Copy	Delete	Speaker	3	NULL	FatimaZahra@gmail.com	FEMENIN	lien1	FatimaZahra	NULL	NULL
☐	Edit	Copy	Delete	Speaker	4	NULL	Hasnaa@gmail.com	FEMENIN	lien1	Hasnaa	NULL	NULL
☐	Edit	Copy	Delete	Moderateur	5	NULL	Noureddine@gmail.com	FEMENIN	NULL	Noureddine	NULL	Directeur



```
SELECT * FROM `salle`
```

				id	name
☐	Edit	Copy	Delete	1	Salle 111

New
conference
New
commentaire
conference
inscription
participant
salle
session
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options


New
conference
New
commentaire
conference
inscription
participant
salle
session
db_cinema
e
e-bank
information_schema
mysql
pat_db

```
SELECT * FROM `session`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	id	name	moderateur_id	salle_id
Edit Copy Delete	1	Session 1	5	1

Check all With selected: Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table

Query results operations

3. Couche Web REST API :

En créant les DTO et les mappeurs requis,

LES DTOS

```
@Data
public class CommentDTO {
    private Long id;
    private Date date;
    private String contenu;
    private int nblike;
}
```

```
@Data
public class ConferenceDTO {
    private Long id;
    private Date date;
    private String heureDebut;
    private String heureFin;
}
```

```
@Data
public class InscriptionDTO {
    private Long id;
    private Date date;
    private Status status;
    private double prix;
}
```

```
@Data
public class ParticipantDTO {
    private String type;
}
```

```
@Data
public class InviteDTO {
    private Long id;
```

```java
    private String name;
    private String email;
    private String photo;
    private Genre genre;
    private String affiliation;
}
```

```java
@Data
public class ModerateurDTO {
    private Long id;
    private String name;
    private String email;
    private String photo;
    private Genre genre;
    private String specialite;
}
```

```java
@Data
public class SpeakerDTO {
    private Long id;
    private String name;
    private String email;
    private String photo;
    private Genre genre;
    private String lienProfil;
}
```

```java
@Data
public class SalleDTO {
    private Long id;
    private String name;
}
```

```java
@Data
public class SessionDTO {
    private Long id;
    private String name;
}
```

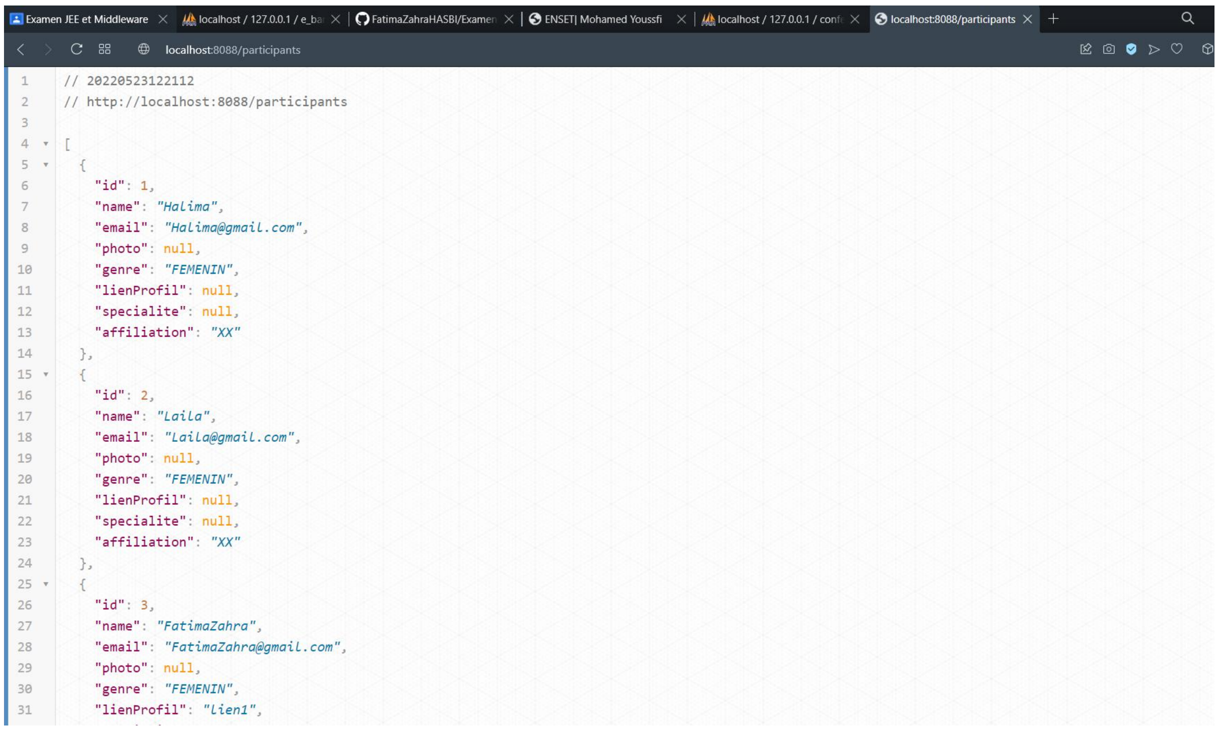
Mappers

```java
package hasbi.fatimazahra.fatimazahrahasbi.MAPPERS;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.Conference;
import hasbi.fatimazahra.fatimazahrahasbi.DTO.ConferenceDTO;
import org.springframework.beans.BeanUtils;
import org.springframework.stereotype.Service;
```

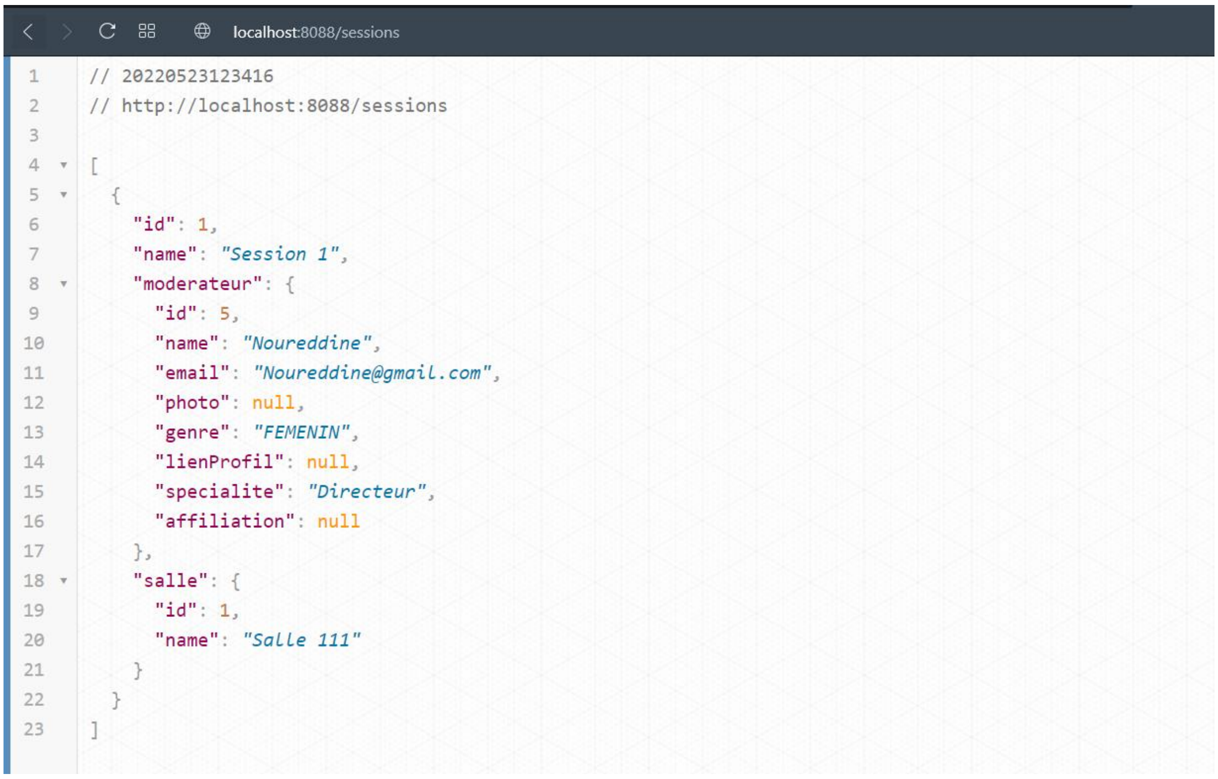
```java
@Service
public class classMappers {
    public ConferenceDTO fromConference(Conference conference){
        ConferenceDTO conferenceDTO=new ConferenceDTO();
        BeanUtils.copyProperties(conference,conferenceDTO);
        return  conferenceDTO;
    }
    public Conference fromConferenceDTO(ConferenceDTO conferenceDTO){
        Conference conference=new Conference();
        BeanUtils.copyProperties(conferenceDTO, conference);
        return  conference;
    }
}
```

- Créer le Web service RESTful qui permet de gérer les sessions et les conférences
- Créer le Web service RESTful qui permet de gérer les inscriptions

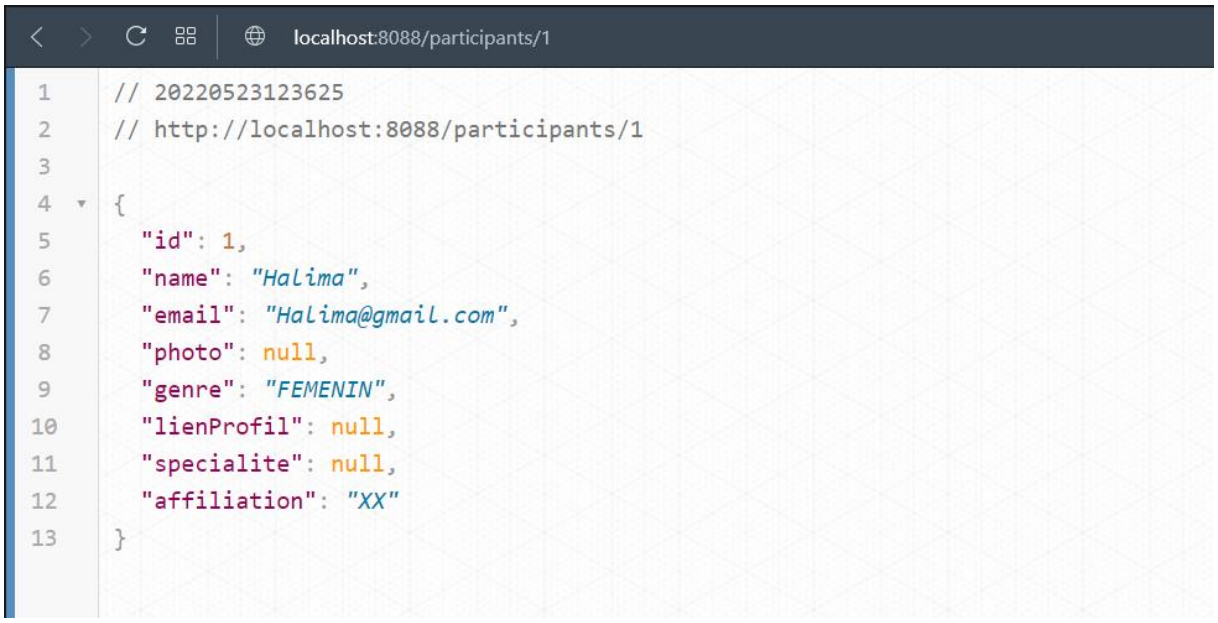
```java
package hasbi.fatimazahra.fatimazahrahasbi.web;

import hasbi.fatimazahra.fatimazahrahasbi.DAO.entities.*;
import hasbi.fatimazahra.fatimazahrahasbi.DAO.repositories.*;
import lombok.AllArgsConstructor;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@AllArgsConstructor
public class webController {
    ParticipantRepository participantRepository;
    ConferenceRepository conferenceRepository;
    InscriptionRepository inscriptionRepository;
    SessionRepository sessionRepository;
    SalleRepository salleRepository;

    @GetMapping(path = "/participants")
    public List<Participant> participants(){
        return participantRepository.findAll();
    }

    @GetMapping(path = "participants/{id}")
    public Participant getParticipant(@PathVariable Long id){
        return participantRepository.findById(id).orElse(null);
    }

    @PostMapping("/participants")
    public Participant saveParticipant(@RequestBody Participant
participant){
        return participantRepository.save(participant);
    }

    @DeleteMapping("/participants/{id}")
    public void deleteParticipant(@PathVariable Long id){
        participantRepository.deleteById(id);
    }
    @GetMapping(path = "/conferences")
    public List<Conference> conferences(){
        return conferenceRepository.findAll();
    }

    @GetMapping(path = "conferences/{id}")
    public Conference getConference(@PathVariable Long id){
        return conferenceRepository.findById(id).orElse(null);
    }

    @PostMapping("/conferences")
    public Conference saveConference(@RequestBody Conference
conference){
        return conferenceRepository.save(conference);
    }

    @DeleteMapping("/conferences/{id}")
    public void deleteConference(@PathVariable Long id){
        conferenceRepository.deleteById(id);
    }

    @GetMapping(path = "/inscriptions")
    public List<Inscription> inscriptions(){
        return inscriptionRepository.findAll();
    }

    @GetMapping(path = "inscriptions/{id}")
    public Inscription getInscription(@PathVariable Long id){
        return inscriptionRepository.findById(id).orElse(null);
    }

    @PostMapping("/inscriptions")
    public Inscription saveInscription(@RequestBody Inscription
inscription){
        return inscriptionRepository.save(inscription);
    }

    @DeleteMapping("/inscriptions/{id}")
    public void deleteInscription(@PathVariable Long id){
        inscriptionRepository.deleteById(id);
    }

    @GetMapping(path = "/sessions")
    public List<Session> sessions(){
        return sessionRepository.findAll();
    }

    @GetMapping(path = "/sessions/{id}")
    public Session getSession(@PathVariable Long id){
        return sessionRepository.findById(id).orElse(null);
    }

    @PostMapping("/sessions")
    public Session saveSession(@RequestBody Session session){
        return sessionRepository.save(session);
    }
    @DeleteMapping("/sessions/{id}")
    public void deleteSession(@PathVariable Long id){
        sessionRepository.deleteById(id);
    }


}
```


Examen JEE et Middleware
localhost / 127.0.0.1 / e_ba
FatimaZahraHASBI/Examen
ENSET| Mohamed Youssfi
localhost / 127.0.0.1 / conf
localhost:8088/participants
localhost:8088/participants

```
// 20220523122112
// http://localhost:8088/participants

[
  {
    "id": 1,
    "name": "Halima",
    "email": "Halima@gmail.com",
    "photo": null,
    "genre": "FEMENIN",
    "lienProfil": null,
    "specialite": null,
    "affiliation": "XX"
  },
  {
    "id": 2,
    "name": "Laila",
    "email": "Laila@gmail.com",
    "photo": null,
    "genre": "FEMENIN",
    "lienProfil": null,
    "specialite": null,
    "affiliation": "XX"
  },
  {
    "id": 3,
    "name": "FatimaZahra",
    "email": "FatimaZahra@gmail.com",
    "photo": null,
    "genre": "FEMENIN",
    "lienProfil": "lien1",
```

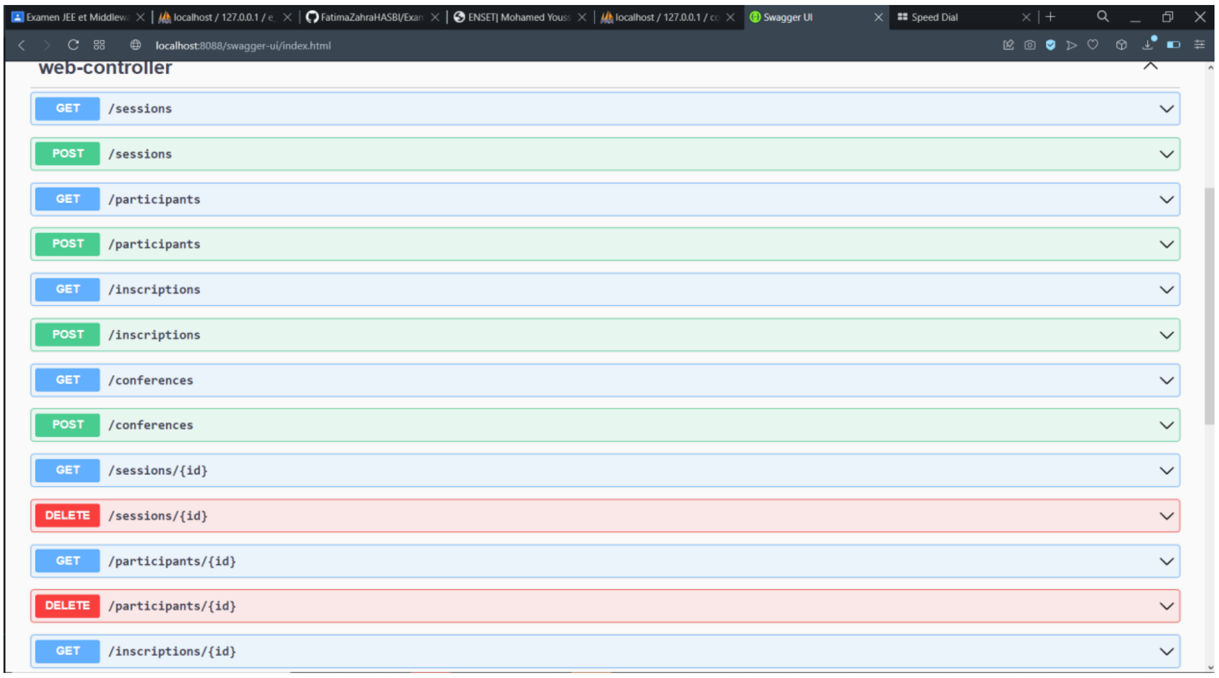

localhost:8088/sessions

```
// 20220523123416
// http://localhost:8088/sessions

[
  {
    "id": 1,
    "name": "Session 1",
    "moderateur": {
      "id": 5,
      "name": "Noureddine",
      "email": "Noureddine@gmail.com",
      "photo": null,
      "genre": "FEMENIN",
      "lienProfil": null,
      "specialite": "Directeur",
      "affiliation": null
    },
    "salle": {
      "id": 1,
      "name": "Salle 111"
    }
  }
]
```


localhost:8088/participants/1

```
// 20220523123625
// http://localhost:8088/participants/1

{
  "id": 1,
  "name": "Halima",
  "email": "Halima@gmail.com",
  "photo": null,
  "genre": "FEMENIN",
  "lienProfil": null,
  "specialite": null,
  "affiliation": "XX"
}
```

4. Générer la documentation SWAGGER des API RESTful



web-controller

```
GET /sessions
POST /sessions
GET /participants
POST /participants
GET /inscriptions
POST /inscriptions
GET /conferences
POST /conferences
GET /sessions/{id}
DELETE /sessions/{id}
GET /participants/{id}
DELETE /participants/{id}
GET /inscriptions/{id}
```

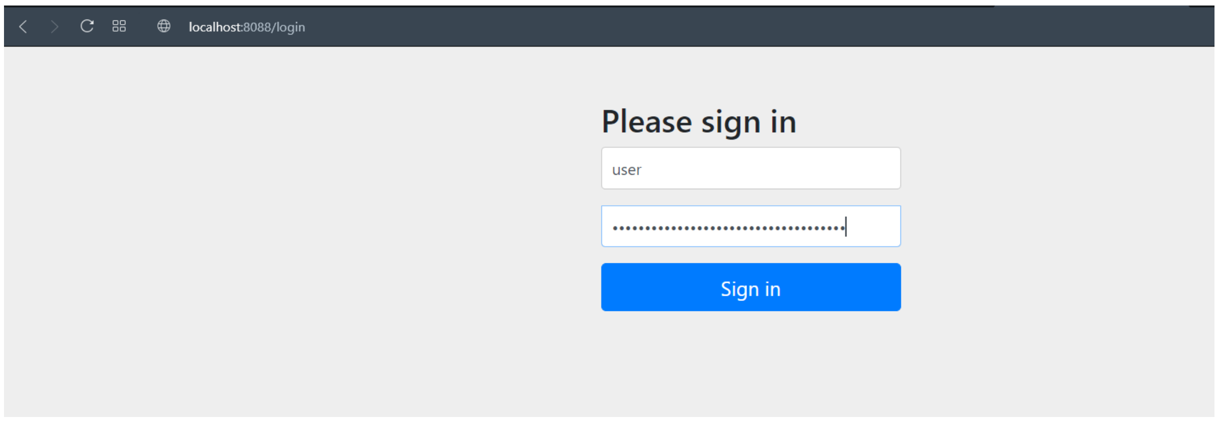
5. Tester les Web service avec un client REST comme Postman

6. Proposer une application frontend en utilisant Angular Framework (de préférence) ou Thymeleaf.

7. Sécuriser d'accès à cette application en se basant sur Spring security avec un système d'authentification des utilisateurs avec 3 types de rôles « ROLE_INVITE », « ROLE_MODERATEUR », « ROLE_CONFERENCIER » et « ADMIN » en choisissant des autorisations appropriées à ses rôles

```
package hasbi.fatimazahra.fatimazahrahasbi.security;

import lombok.AllArgsConstructor;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.config.annotation.authentication.builders.
AuthenticationManagerBuilder;
import
org.springframework.security.config.annotation.web.builders.HttpSecurit
y;
import
org.springframework.security.config.annotation.web.configuration.Enable
WebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSec
urityConfigurerAdapter;
import org.springframework.security.crypto.password.PasswordEncoder;

import javax.sql.DataSource;

@Configuration
```

```
@EnableWebSecurity
@AllArgsConstructor
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private DataSource dataSource;
    PasswordEncoder passwordEncoder;
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws
Exception {
        auth.jdbcAuthentication()
                .dataSource(dataSource)
                .usersByUsernameQuery("select username as principal,
password as credentials, active from users where username=?")
                .authoritiesByUsernameQuery("select username as
principal, role from users_roles where username=?")
                .rolePrefix("ROLE_")
                .passwordEncoder(passwordEncoder);

    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin();
        http.authorizeRequests().antMatchers("/").permitAll();

http.authorizeRequests().antMatchers("/admin/**").hasAuthority("ADMIN")
;

http.authorizeRequests().antMatchers("/roleinvite/**").hasAuthority("RO
LE_INVITE");

http.authorizeRequests().antMatchers("/rolemoderateur/**").hasAuthority
("ROLE_MODERATEUR");

http.authorizeRequests().antMatchers("/roleconferencier/**").hasAuthori
ty("ROLE_CONFERENCIER");

http.authorizeRequests().antMatchers("/webjars/**").permitAll();
        http.authorizeRequests().anyRequest().authenticated();
        http.exceptionHandling().accessDeniedPage("/403");
    }

}
```


localhost:8088/login

Please sign in

user

Sign in