Rapport du projet:

Système de Réservations

Module:

(Technologie Web & Web Sémantique)

MASTER : Systèmes Distribués et intelligence Artificielle (SDIA)

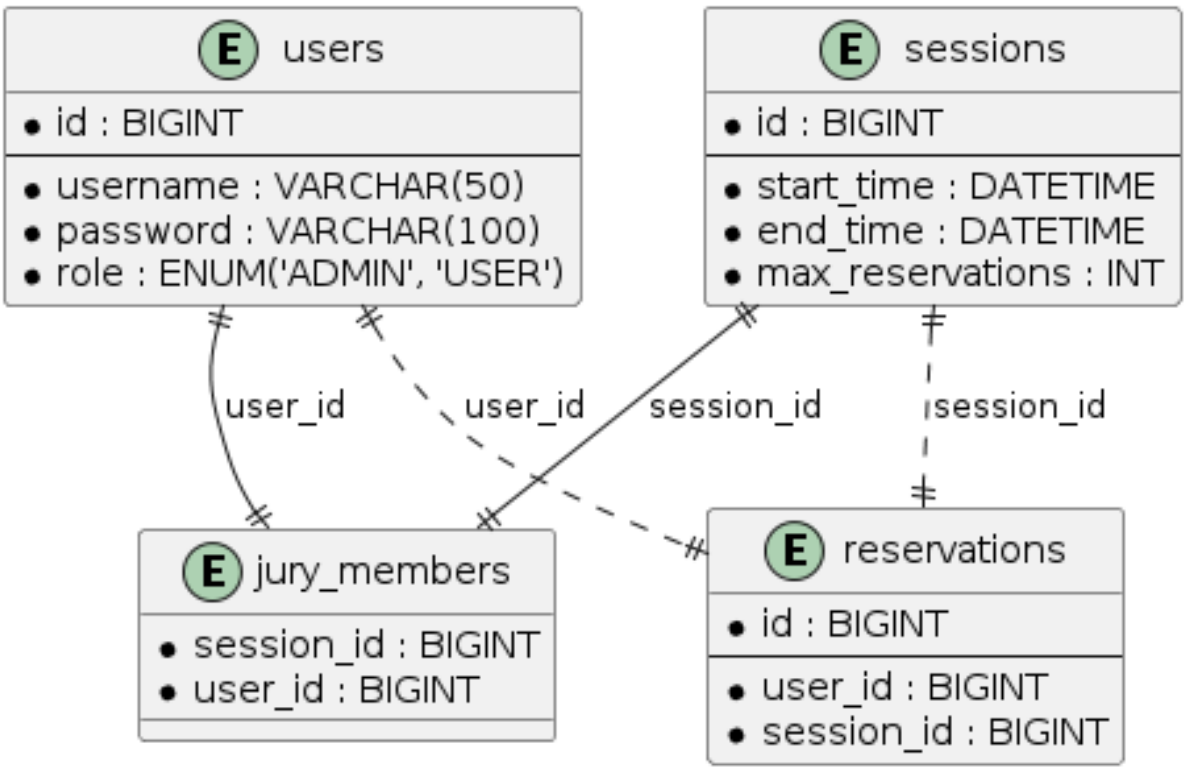
Elaboré par :

IGUIDER AMINE

Année Universitaire : 2023-2024

1. Conception :

Diagramme de classes :



Ce diagramme de classes définit une structure claire pour le système de réservation, incluant les rôles des utilisateurs, la gestion des réservations, et la planification des sessions. Les interfaces utilisateur sont conçues pour être ergonomiques et intuitives, permettant une gestion efficace des réservations et des annulations.

Classes et leurs attributs :

1- Classe User :

Elle représente les utilisateurs du système, qu'ils soient administrateurs ou utilisateurs normaux. Chaque utilisateur a un identifiant unique, un nom d'utilisateur et un mot de passe.

- Id : int : identifiant unique de l'utilisateur
- Username : string : Nom de l'utilisateur
- Password : string : Mot de passe de l'utilisateur

2- Classe Réservation :

Elle représente les réservations effectuées par les utilisateurs. Chaque réservation a un identifiant unique.

- Id : int : identifiant unique de la réservation

3- Sessions :

Elle représente les sessions de réservation créées par les administrateurs. Chaque session a une heure de début, une heure de fin, et un nombre maximum de réservations possibles.

- Id : int : identifiant de la session
- Start_time : Date : Heure de début de la session
- End_time : Date : Heure de fin de la session
-

4- Role :
Elle est de type 'Enumération' et elle représente les rôles possibles des utilisateurs dans le système : soit 'Admin' (administrateur) soit 'User' (utilisateur normal).
Le rôle peut avoir les valeurs :

- Admin
- User

Les relations entre les classes sont les suivantes : chaque utilisateur possède un rôle unique (administrateur ou utilisateur), chaque utilisateur peut effectuer plusieurs réservations liées à lui seul, et chaque réservation est associée à une session unique, tandis qu'une session peut avoir plusieurs réservations. Les administrateurs peuvent planifier des sessions, définir les membres du jury et fixer le nombre maximal de réservations par heure. Les utilisateurs peuvent réserver des sessions qui ne se chevauchent pas en termes d'horaires. L'interface utilisateur permet aux administrateurs de gérer les sessions et aux utilisateurs de réserver des créneaux, avec un affichage optimisé des disponibilités.

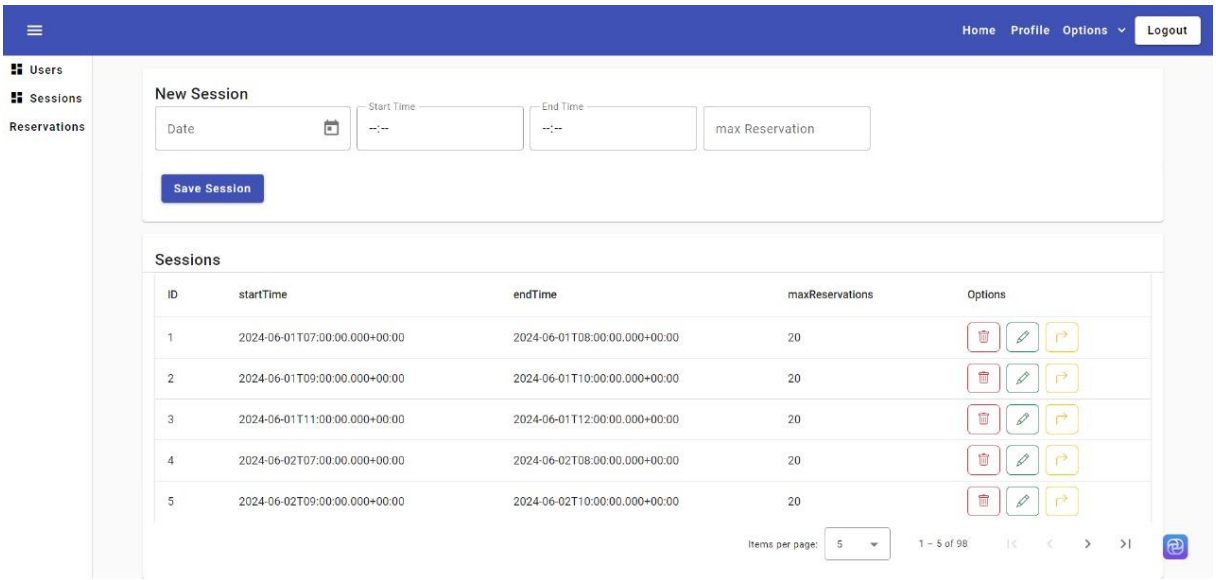
Parties importantes du code :

1. Partie session :

Code contrôleur :

```java
package sdia.reservationbackend.web;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;
import sdia.reservationbackend.dtos.SessionDTO;
import sdia.reservationbackend.exceptions.SessionNotFoundException;
import sdia.reservationbackend.services.ReservationService;

import java.util.List;

@RestController
@AllArgsConstructor
@Slf4j
@CrossOrigin("*")
public class SessionRestController {
    private ReservationService reservationService;
    @GetMapping("/sessions")
    public List<SessionDTO> SessionList(){
        return reservationService.SessionList();
    }

    @GetMapping("/sessions/{id}")
    public SessionDTO getSession(@PathVariable(name = "id") Long sessionId) throws SessionNotFoundException {
        return reservationService.getSession(sessionId);
    }

    @PostMapping("/session")
    public SessionDTO saveSession(@RequestBody SessionDTO sessionDTO){
        return reservationService.saveSession(sessionDTO);
    }

    @PutMapping("/sessions/{sessionId}")
    public SessionDTO updateSession(@PathVariable Long sessionId, @RequestBody SessionDTO sessionDTO){
        sessionDTO.setId(sessionId);
        return reservationService.updateSession(sessionDTO);
    }

    @DeleteMapping("/sessions/{id}")
    public void deleteSession(@PathVariable Long id){
        reservationService.deleteSession(id);
    }

}
```

Interface :

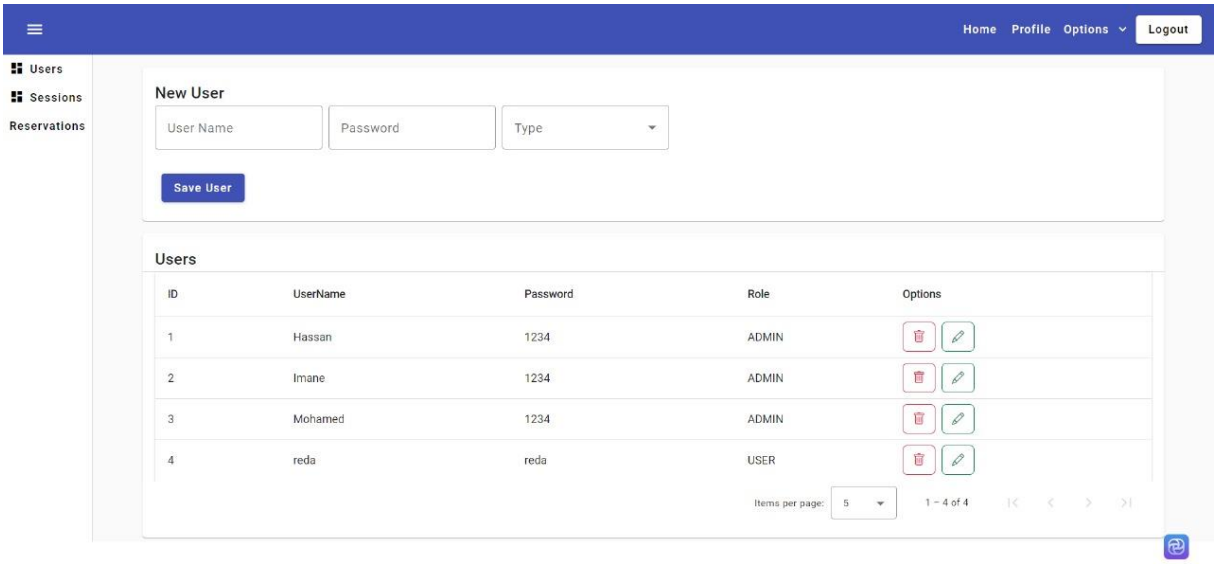


2. Partie user :

Code contrôleur :

```java
package sdia.reservationbackend.web;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;
import sdia.reservationbackend.dtos.UserDTO;
import sdia.reservationbackend.exceptions.UserNotFoundException;
import sdia.reservationbackend.services.ReservationService;

import java.util.List;

@RestController
@AllArgsConstructor
@Slf4j
@CrossOrigin("*")
public class UserRestController {
    private ReservationService reservationService;

    @GetMapping("/users")
    public List<UserDTO> listUsers(){
        return reservationService.listUsers();
    }

    @GetMapping("/users/{id}")
    public UserDTO getUser(@PathVariable(name = "id") Long userId) throws UserNotFoundException {
        return reservationService.getUser(userId);
    }

    @PostMapping("/user")
    public UserDTO saveUser(@RequestBody UserDTO userDTO){
        return reservationService.saveUser(userDTO);
    }

    @PutMapping("/users/{userId}")
    public UserDTO updateUser(@PathVariable Long userId, @RequestBody UserDTO userDTO){
        userDTO.setId(userId);
        return reservationService.updateUser(userDTO);
    }

    @DeleteMapping("/users/{id}")
    public void deleteUser(@PathVariable Long id){
        reservationService.deleteUser(id);
    }

}
```

Interface :


Home
Profile
Options
Logout
Users
Sessions
Reservations
New User
User Name
Password
Type
Save User
Users
ID
UserName
Password
Role
Options
1
Hassan
1234
ADMIN
2
Imane
1234
ADMIN
3
Mohamed
1234
ADMIN
4
reda
reda
USER
Items per page:
5
1 – 4 of 4

3- reservation :

Code contrôleur :

```java
package sdia.reservationbackend.web;

import lombok.AllArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;
import sdia.reservationbackend.dtos.ReservationDTO;
import sdia.reservationbackend.exceptions.ReservationNotFoundException;
import sdia.reservationbackend.services.ReservationService;

import java.util.List;

@RestController
@AllArgsConstructor
@Slf4j
@CrossOrigin("*")
public class ReservationRestController {
    private ReservationService reservationService;

    @GetMapping("/reservations")
    public List<ReservationDTO> ReservationList(){
        return reservationService.ReservationList();
    }
    @GetMapping("/reservations/{id}")
    public ReservationDTO getReservation(@PathVariable(name = "id") Long reservationId) throws ReservationNotFoundException {
        return reservationService.getReservation(reservationId);
    }

    @PostMapping("/reservation")
    public ReservationDTO saveReservation(@RequestBody ReservationDTO reservationDTO){
        return reservationService.saveReservation(reservationDTO);
    }

    @PutMapping("/reservations/{reservationId}")
    public ReservationDTO updateReservation(@PathVariable Long reservationId, @RequestBody ReservationDTO reservationDTO){
        reservationDTO.setId(reservationId);
        return reservationService.updateReservation(reservationDTO);
    }

    @DeleteMapping("/reservations/{id}")
    public void deleteReservation(@PathVariable Long id){
        reservationService.deleteReservation(id);
    }
}
```