## CREAR SALA

### RoomJava:

package entities;

import entities.User;

import java.util.ArrayList;

import java.text.SimpleDateFormat;

import java.sql.Date;

public class Room {

private int id;

private String name;

private String code;

private int amountParticipants;

private int maxAmountParticipants;

private Date initDate;

private Date endDate;

private boolean deleted = false;

private User admin;

private ArrayList<User> users;

private ArrayList<Quiz> quizzes;

SimpleDateFormat outputFormat = new SimpleDateFormat("dd/MM/yyyy");

public User getAdmin() {

return admin;

}

public void setAdmin(User admin) {

this.admin = admin;

}

public ArrayList<Quiz> getQuizzes() {

return quizzes;

}

public void setQuizzes(ArrayList<Quiz> quizzes) {

this.quizzes = quizzes;

}

public ArrayList<User> getUsers() {

return users;

}

public void setUsers(ArrayList<User> users) {

this.users = users;

}

public boolean isDeleted() {

return deleted;

}

public void setDeleted(boolean deleted) {

this.deleted = deleted;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public int getAmountParticipants() {

return amountParticipants;

}

public void setAmountParticipants(int amountParticipants) {

this.amountParticipants = amountParticipants;

}

public int getMaxAmountParticipants() {

return maxAmountParticipants;

}

public void setMaxAmountParticipants(int maxAmountParticipants) {

this.maxAmountParticipants = maxAmountParticipants;

}

public Date getInitDate() {

return initDate;

}

public String getInitDateAsString() {

if (initDate != null) {

return outputFormat.format(initDate);

}

return "";

}

public void setInitDate(Date initDate) {

this.initDate = initDate;

}

public Date getEndDate() {

return endDate;

}

public String getEndDateAsString() {

if (endDate != null) {

return outputFormat.format(endDate);

}

return "";

}

public void setEndDate(Date endDate) {

this.endDate = endDate;

}

public void addUser(User u) {

this.users.add(u);

this.amountParticipants=+1;

}

public Room() {}

public Room(int id, String name, String code, int amountParticipants, int maxAmountParticipants, Date initDate, Date endDate, boolean deleted) {

super();

this.name = name;

this.id = id;

this.code = code;

this.amountParticipants = amountParticipants;

this.maxAmountParticipants = maxAmountParticipants;

this.initDate = initDate;

this.endDate = endDate;

this.deleted = deleted;

}

public boolean isValid() {

return endDate.after( new Date(System.currentTimeMillis() ));

}

}

### IndexJSP (fragmento):

<div>

<article>

<strong>Crea una sala</strong>

<footer>

<form action="createroom" method="get">

<button type="submit">Crear</button>

</form>

</footer>

</article>

</div>

### CreateRoomJSP:

<%@ page import="entities.User" %>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min.css" />

<%

User usr = (User)request.getSession().getAttribute("user");

%>

<title>Crear una sala</title>

</head>

<body>

<header class="container">

<nav>

<ul>

<a href="index.jsp"><li><strong>TpJava WebQuizApp</strong></li></a>

</ul>

</nav>

</header>

<main class="container">

<h3>Crear una sala</h3>

<form action="createroom" method="post">

<article>

<fieldset>

<label>Nombre

<input name="name" placeholder="Nombre" autocomplete="given-name" required/>

</label>

<label>Cantidad máxima de personas

<input type="number" name="maxAmountParticipants" min=1 placeholder="Ingrese un número" autocomplete="given-name" required/>

</label>

<label>Código

<input name="code" maxlength="36" placeholder="Código" autocomplete="given-name" required/>

</label>

<label>Fecha de finalización

<input type="date" name="endDate" aria-label="Date" min="<%= java.time.LocalDate.now().plusDays(1).toString() %>" required/>

</label>

</fieldset>

<footer>

<button name="action" value="createRoomValidation" type="submit">Crear</button>

</footer>

</article>

</form>

</body>

### JoinExistingRoomJSP:

<%@ page import="entities.User" %>

<%@ page import="entities.Room" %>

<%@ page language="java" contentType="text/html; charset=UTF-8"

pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1" />

<link rel="stylesheet"

href="https://cdn.jsdelivr.net/npm/@picocss/pico@2/css/pico.min.css" />

<%

User usr = (User)request.getSession().getAttribute("user");

Room room = (Room)request.getAttribute("room");

%>

<title>Unirse a una sala</title>

</head>

<body>

<header class="container">

<nav>

<ul>

<a href="index.jsp"><li><strong>TpJava WebQuizApp</strong></li></a>

</ul>

</nav>

</header>

<main class="container">

<article>

<strong>La sala con código <%=room.getCode()%> ya existe</strong>

<p>El nombre es <%=room.getName()%></p>

<p>¿Desea ingresar?</p>

<footer>

<form action="room" method="post" style="display:contents;">

<input type="hidden" value="<%=room.getCode()%>" name="code">

<button style="width: 45%; margin:30px;">Aceptar</button>

</form>

<a href=createRoom >

<button style="width: 45%; margin:30px; background-color:grey">Volver</button>

</a>

</footer>

</article>

</main>

</body>

### 

### CreateRoomServlet:

package servlets;

import java.io.IOException;

import java.time.LocalDate;

import java.time.format.DateTimeFormatter;

import java.sql.Date;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import entities.Room;

import entities.User;

import logic.RoomLogic;

/\*\*

\* Servlet implementation class CreateRoomServlet

\*/

@WebServlet({ "/CreateRoomServlet", "/createroom", "/CreateRoom", "/CREATEROOM", "/createRoom" })

public class CreateRoomServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public CreateRoomServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

User usr = (User) request.getSession().getAttribute("user");

if (usr == null) {

request.setAttribute("headTitle", "Acceso denegado");

request.setAttribute("bodyTitle", "Debe iniciar sesión para crear a una sala");

request.setAttribute("buttonAction", "login.html");

request.setAttribute("buttonMessage", "Iniciar sesión");

request.getRequestDispatcher("WEB-INF/message.jsp").forward(request, response);

return;

}

request.getRequestDispatcher("WEB-INF/createroom.jsp").forward(request, response);

}

/\*\*

\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)

\*/

protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

User usr = (User) request.getSession().getAttribute("user");

if (usr == null) {

request.setAttribute("headTitle", "Acceso denegado");

request.setAttribute("bodyTitle", "La sesión expiró");

request.setAttribute("buttonAction", "login.html");

request.setAttribute("buttonMessage", "Iniciar sesión");

request.getRequestDispatcher("WEB-INF/message.jsp").forward(request, response);

return;

}

RoomLogic roomLogic = new RoomLogic();

Room room = new Room();

Integer maxAmountParticipants;

String roomcode = request.getParameter("code");

String roomname = request.getParameter("name");

Date endDate = Date.valueOf(request.getParameter("endDate"));

room.setEndDate(endDate);

room.setName(roomname);

room.setCode(roomcode);

room.setInitDate(new Date(System.currentTimeMillis()));

try {

maxAmountParticipants = Integer.parseInt(request.getParameter("maxAmountParticipants"));

room.setMaxAmountParticipants(maxAmountParticipants);

} catch (NumberFormatException e) {

maxAmountParticipants = -1;

}

if(maxAmountParticipants == -1 || maxAmountParticipants < 1) {

request.setAttribute("headTitle", "Caracteres invalidos");

request.setAttribute("bodyTitle", "Debe ingresar caracteres válidos para crear la sala");

request.setAttribute("buttonAction", "createRoom");

request.setAttribute("buttonMessage", "Volver a intentar");

request.getRequestDispatcher("WEB-INF/message.jsp").forward(request, response);

return;

}

Room existingRoom = roomLogic.checkExistingRooms(room);

if (existingRoom != null) {

request.setAttribute("room", existingRoom);

request.getRequestDispatcher("WEB-INF/joinexistingroom.jsp").forward(request, response);

return;

}

roomLogic.save(room);

Room createdRoom = roomLogic.getRoomByCode(room);

roomLogic.setRoomAdmin(usr, createdRoom);

roomLogic.addUserToRoom(usr, createdRoom);

request.setAttribute("headTitle", "Sala creada");

request.setAttribute("bodyTitle", "La sala ha sido creada con éxito");

request.setAttribute("buttonAction", "index.jsp");

request.setAttribute("buttonMessage", "Aceptar");

request.getRequestDispatcher("WEB-INF/message.jsp").forward(request, response);

}

}

### 

### RoomLogic (Métodos utilizados):

public Room checkExistingRooms(Room r) {

ArrayList<Room> roomsFromDb = roomDB.getAll();

for (Room aRoom: roomsFromDb) {

if(aRoom.getCode().equals(r.getCode())) {

return aRoom;

}

}

return null;

}

public void save(Room r) {

roomDB.save(r);

}

public Room getRoomByCode(Room r) {

Room room = roomDB.getByCode(r);

if (room != null) {

return room;

} else {

return null;

}

}

public void setRoomAdmin(User u, Room r) {

userRoomDB.setRoomAdmin(u, r);

}

public void addUserToRoom(User u, Room r) {

userRoomDB.addUserToRoom(u, r);;

}

### RoomDB (Métodos utilizados):

public ArrayList<Room> getAll(){

String sqlSelect = "select r.\*, (SELECT COUNT(\*) FROM users\_rooms ur WHERE ur.room\_id = r.id) AS amount\_participants from rooms r";

Connect connect = new Connect();

Connection con = connect.getConnection();

ArrayList<Room> rooms = new ArrayList<Room>();

if (con != null) {

try {

PreparedStatement stm = con.prepareStatement(sqlSelect);

ResultSet rs = stm.executeQuery();

while (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String code = rs.getString("code");

int amountParticipants = rs.getInt("amount\_participants");

int maxAmountParticipants = rs.getInt("max\_amount\_participants");

Date initDate = rs.getDate("init\_date");

Date endDate = rs.getDate("end\_date");

boolean deleted = rs.getBoolean("deleted");

rooms.add(new Room(id, name, code, amountParticipants, maxAmountParticipants, initDate, endDate, deleted));

}

con.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return rooms;

}

public void save(Room r) {

String sqlSelect = "insert into rooms(name, code, max\_amount\_participants, init\_date, end\_date) values(?, ?, ?, ?, ?)";

Connect connect = new Connect();

Connection con = connect.getConnection();

if (con != null) {

try {

PreparedStatement stm = con.prepareStatement(sqlSelect);

stm.setString(1, r.getName());

stm.setString(2, r.getCode());

stm.setInt(3, r.getMaxAmountParticipants());

stm.setDate(4, r.getInitDate());

stm.setDate(5, r.getEndDate());

stm.executeUpdate();

con.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

public Room getByCode(Room r) {

String sqlSelect = "select r.\*, (SELECT COUNT(\*) FROM users\_rooms ur WHERE ur.room\_id = r.id) AS amount\_participants from rooms r where r.code = ?";

Connect connect = new Connect();

Connection con = connect.getConnection();

Room room = null;

if (con != null) {

try {

PreparedStatement stm = con.prepareStatement(sqlSelect);

stm.setString(1, r.getCode());

ResultSet rs = stm.executeQuery();

if (rs.next()) {

int id = rs.getInt("id");

String name = rs.getString("name");

String code = rs.getString("code");

int amountParticipants = rs.getInt("amount\_participants");

int maxAmountParticipants = rs.getInt("max\_amount\_participants");

Date initDate = rs.getDate("init\_date");

Date endDate = rs.getDate("end\_date");

boolean deleted = rs.getBoolean("deleted");

room = new Room(id, name, code, amountParticipants, maxAmountParticipants, initDate, endDate, deleted);

}

con.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return room;

}

### UserRoomDB (Métodos utilizados):

public void addUserToRoom(User u, Room r){

String sqlSelect = "INSERT INTO users\_rooms (user\_id, room\_id) VALUES (?, ?)";

Connect connect = new Connect();

Connection con = connect.getConnection();

if (con != null) {

try {

PreparedStatement stm = con.prepareStatement(sqlSelect);

stm.setInt(1, u.getId());

stm.setInt(2, r.getId());

stm.executeUpdate();

con.close();

} catch(SQLException e){

e.printStackTrace();

}

}

}

public void setRoomAdmin(User u, Room r) {

String sqlSelect = "UPDATE rooms SET admin = ? WHERE rooms.id = ?";

Connect connect = new Connect();

Connection con = connect.getConnection();

if (con != null) {

try {

PreparedStatement stm = con.prepareStatement(sqlSelect);

stm.setInt(1, u.getId());

stm.setInt(2, r.getId());

stm.executeUpdate();

con.close();

} catch (SQLException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}