**Лаба 2**



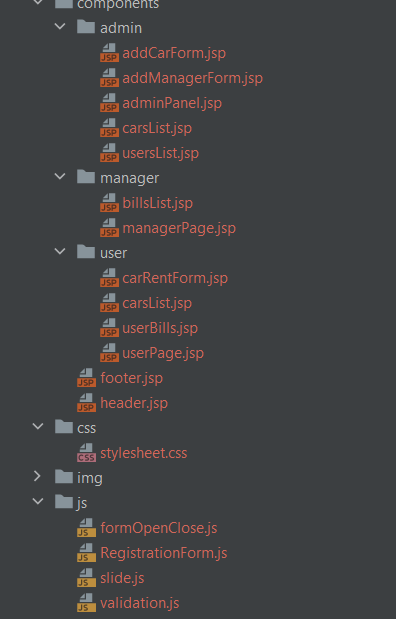
Інтерфейси(Код імплементації на гіт)

public interface AdminDAO {  
 List<Car> getAllCars();  
 boolean addCar(Car car);  
 boolean deleteCar(int idCar);  
 boolean editCar(Car car);  
 List<User> getAllUsers();  
 boolean addManager(User user);  
 boolean banUser(int idUser);  
 boolean unbanUser(int idUser);  
}

public interface BillDAO {  
 List<Bill> allBillsWithStatusInRent();  
 boolean editBillStatus(int idBill, String status);  
 boolean editReturnStatus(int idBill, String status);  
 List<Bill> allUserBills(User user);  
 boolean createBill(Bill bill);  
 boolean changeBillPaymentStatus(int idbill);  
}

public interface CarDAO {  
 Car getCarById(int id);  
 List<Car> allCars();  
 List<Car> allCarsSortedByName();  
 List<Car> allCarsSortedByNameDESC();  
 List<Car> allCarsSortedByPrice();  
 List<Car> allCarsSortedByPriceDESC();  
 List<Car> carsByBrand(String brand);  
 List<Car> carsByClass(String carClass);  
 List<Car> carsByPrice(double start, double end);  
}

public interface UserDAO {  
 User signIn(String email, String password);  
 User register(User user);  
}



Реалізація форм, валідація та інще(Реалізація на гіт)

**Лаба 3**

Транзакція

statement.executeUpdate(SQL);

connection.commit();

} catch (SQLException e) {

System.out.println("SQLException. Executing rollback to savepoint...");

connection.rollback(savepointOne);

}

Фільтри на доступ до адмін контенту

@WebFilter(urlPatterns = {"/AddCar", "/AddManager", "/BanUser", "/DeleteCar", "/EditCar", "/GetAllCars", "/GetAllUsers", "/UnbanUser"})  
public class AdminFilter implements Filter {  
 private static final Logger *LOG* = LoggerFactory.*getLogger*(AdminFilter.class);  
 HttpSession session;  
 User user;  
  
 public void init(HttpServletRequest req) {  
 session = req.getSession();  
 user = (User) session.getAttribute("user");  
 }  
  
 @Override  
 public void doFilter(ServletRequest req,  
 ServletResponse resp,  
 FilterChain filterChain)  
 throws IOException, ServletException {  
 HttpServletRequest request = (HttpServletRequest) req;  
 init(request);  
 if (user == null && !user.getRole().equals("admin") && user.getBanStatus()) {  
 ((HttpServletResponse) resp).sendRedirect("index.jsp");  
 }  
 filterChain.doFilter(req, resp);  
 }  
}

Валідація даних (js)

function phoneMask() {  
 var mask = "+38(0\_\_)-\_\_\_-\_\_-\_\_";  
 var phoneNumber = document.getElementById("phone").value;  
 var numbersInPhoneNumber = phoneNumber.toString().replace(/\D+/g, "");  
 numbersInPhoneNumber = numbersInPhoneNumber.slice(0, 12);  
 if (numbersInPhoneNumber.length == 1) {  
 mask = mask.replace("\_", numbersInPhoneNumber.charAt(0));  
 }  
 for (let i = 3; i < numbersInPhoneNumber.length; i++) {  
 mask = mask.replace("\_", numbersInPhoneNumber.charAt(i));  
 }  
 var cursorPosition = mask.search("\_");  
 document.getElementById("phone").value = mask;  
 document.getElementById("phone").setSelectionRange(cursorPosition, cursorPosition);  
}  
  
function validateDate(element, DATE\_REQUIRED) {  
 const currentDate = new Date();  
 const d = new Date(element.value);  
 if (d instanceof Date && !isNaN(d) && d.getFullYear() > 1900 && d<currentDate) {  
 return true;  
 }  
 return showError(element, DATE\_REQUIRED);  
}  
  
function submitRegisterForm() {  
 const form = document.getElementById("register");  
 const EMAIL\_INVALID = "Please enter a correct email address format";  
 const EMAIL\_REQUIRED = "Please enter your email";  
 let emailValid = validateEmail(form.elements["email"], EMAIL\_REQUIRED, EMAIL\_INVALID);  
 const PASS\_REQUIRED = "Please enter your pass";  
 let passValid = hasValue(form.elements["password"], PASS\_REQUIRED);  
 const SURNAME\_REQUIRED = "Please enter your surname";  
 let surnameValid = hasValue(form.elements["surname"], SURNAME\_REQUIRED);  
 const NAME\_REQUIRED = "Please enter your name";  
 let nameValid = hasValue(form.elements["name"], NAME\_REQUIRED);  
 const DATE\_REQUIRED = "Please enter your Date";  
 let dateValid = validateDate(form.elements["date"], DATE\_REQUIRED);  
 const NUMBER\_REQUIRED = "Please enter your Number";  
 let numberValid = validateNumber(form.elements["phoneNumber"], NUMBER\_REQUIRED);  
 if (emailValid && passValid && surnameValid && nameValid && dateValid && numberValid) {  
 form.submit();  
 }  
}  
  
function submitLoginForm() {  
 const form = document.getElementById("signIn");  
 const EMAIL\_INVALID = "Please enter a correct email address format";  
 const EMAIL\_REQUIRED = "Please enter your email";  
 let emailValid = validateEmail(form.elements["email"], EMAIL\_REQUIRED, EMAIL\_INVALID);  
 const PASS\_REQUIRED = "Please enter your pass";  
 let passValid = hasValue(form.elements["password"], PASS\_REQUIRED);  
 if (emailValid && passValid) {  
 form.submit();  
 }  
}  
  
function showMessage(input, message, type) {  
 const msg = input.parentNode.querySelector("small");  
 msg.innerText = message;  
 input.className = type ? "success" : "error";  
 return type;  
}  
  
function showError(input, message) {  
 return showMessage(input, message, false);  
}  
  
function showSuccess(input) {  
 return showMessage(input, "", true);  
}  
  
function hasValue(input, message) {  
 if (input.value.trim() === "") {  
 return showError(input, message);  
 }  
 return showSuccess(input);  
}  
  
function validateEmail(input, requiredMsg, invalidMsg) {  
 if (!hasValue(input, requiredMsg)) {  
 return false;  
 }  
 const emailRegex =  
 /^(([^<>()\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)\*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;  
 const email = input.value.trim();  
 if (!emailRegex.test(email)) {  
 return showError(input, invalidMsg);  
 }  
 return true;  
}  
  
function validateNumber(input, requiredMsg) {  
 if (!hasValue(input, requiredMsg)) {  
 return false;  
 }  
 var phoneNumber = input.value;  
 var numbersInPhoneNumber = phoneNumber.toString().replace(/\D+/g, "");  
 if (numbersInPhoneNumber.length != 12) {  
 return showError(input, requiredMsg);  
 }  
 return true;  
}

Помилки, також в фронтенд реалізації при неправильно введених даних

request.setAttribute("errorMassage", "You can't add this car, qualityClass has edited");  
response.sendRedirect("error.jsp");

Логування

import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;

private static final Logger *LOG* = LoggerFactory.*getLogger*(AdminFilter.class);

*LOG*.info("Admin logged in ");

<?xml version="1.0" encoding="UTF-8"?>  
<Configuration status="WARN">  
 <Appenders>  
 <Console name="LogToConsole" target="SYSTEM\_OUT">  
 <PatternLayout pattern="%highlight{%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n}{TRACE=magenta}" disableAnsi="false"/>  
 </Console>  
 </Appenders>  
 <Loggers>  
  
 <Logger name="com.example.autoRent" level="info" additivity="false">  
 <AppenderRef ref="LogToConsole"/>  
 </Logger>  
  
 <Root level="error">  
 <AppenderRef ref="LogToConsole"/>  
 </Root>  
 </Loggers>  
</Configuration>

Сесії

request.getSession().setAttribute("user", user);

**ЛАБА 4**

Логування

<?xml version="1.0" encoding="UTF-8"?>  
<Configuration status="WARN">  
 <Appenders>  
 <Console name="LogToConsole" target="SYSTEM\_OUT">  
 <PatternLayout pattern="%highlight{%d{HH:mm:ss.SSS} [%t] %-5level %logger{36} - %msg%n}{TRACE=magenta}" disableAnsi="false"/>  
 </Console>  
 </Appenders>  
 <Loggers>  
  
 <Logger name="com.example.autoRent" level="info" additivity="false">  
 <AppenderRef ref="LogToConsole"/>  
 </Logger>  
  
 <Root level="error">  
 <AppenderRef ref="LogToConsole"/>  
 </Root>  
 </Loggers>  
</Configuration>

*LOG*.error("Connection pool .getDataSource error," +  
 " Context lookup environment doesn't exist" +  
 " java:comp/env/jdbc/ConnectionPool", e);

*LOG*.error("Fail to add car, ROLLBACK", e);

*LOG*.error("Fail to add user, ROLLBACK", e);

*LOG*.error("Fail to login user, ROLLBACK", e);

Юніт тести

class DatasourceTest {  
  
 private DataSource datasource;  
  
 @org.junit.jupiter.api.BeforeEach  
 void setUp() throws SQLException {  
 datasource = ConnectionPool.*getDataSource*();  
  
 if (!datasource.getConnection().isValid(100)) {  
 System.*exit*(-1);  
 }  
 }  
  
 @org.junit.jupiter.api.Test  
 void insertTable() throws SQLException {  
  
 *assertTrue*(datasource.getConnection().isValid(200));  
  
 }  
  
}

public class TestMyServlet extends Mockito {  
  
 @Test  
 public void testServlet() throws Exception {  
 HttpServletRequest request = *mock*(HttpServletRequest.class);  
 HttpServletResponse response = *mock*(HttpServletResponse.class);  
  
 *when*(request.getParameter("idCar")).thenReturn("0");  
  
 StringWriter stringWriter = new StringWriter();  
 PrintWriter writer = new PrintWriter(stringWriter);  
 *when*(response.getWriter()).thenReturn(writer);  
 new AddCar().doPost(request, response);  
 *verify*(request, *atLeast*(1)).getParameter("idCar");   
 writer.flush();   
 *assertTrue*(stringWriter.toString().contains("boolean"));  
 }  
}

