Для запуска сначала нужно создать ещё одну таблицу в бд для сохранения пользователей:

use car\_rental\_system;

CREATE TABLE user\_accounts (

id INT IDENTITY(1,1) PRIMARY KEY,

username NVARCHAR(255) NOT NULL,

password NVARCHAR(255) NOT NULL,

role NVARCHAR(10) NOT NULL CHECK (role IN ('ADMIN', 'USER', 'GUEST'))

);

Проверить чтобы в persistence.xml были верные логин пароль урл.

**По структуре проекта:**

Появилась сущность User, связанная с таблицей user\_accounts. Появился также DAO класс UserDAO. Это нужно для хранения пользователей после регистрации.

Класс DispatcherServlet заменён на WelcomeFilter –в process() добавилось

var user = authenticateService.validateAndGetAuthUser(request, response);

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

Добавление пользователя в сессию. Главный прикол фильтра – функция doFilter. Она выполняет фильтрацию всех get/post запросов и(они через неё проходят, и можно их отсеивать по надобности) и потом уже передаёт в doGet doPost, это паттерн который называется chain of responsibility. В нашем случае при поступлении запроса проверяется роль пользователя, и если у него нет прав для доступа к странице, он отправляется домой.

if ((userRole == UserRole.GUEST && !isPublicPage))

{

httpResponse.sendRedirect(httpRequest.getContextPath() + "/home");

}

По webapp – index.html поменян на home.html, к нему приставлен контроллер HomeController. В нём проверяется какая роль стоит у пользователя в текущей сессии, кнопки появляются в зависимости от роли текущего пользователя, за это отвечают поля thymeleaf:

th:if="${role == 'ADMIN' or role == 'USER'}"

Три страницы добавлены в папке auth – для логина, для регистрации, и страница которая появляется после успешной регистрации для перенаправления на страницу логина. (при регистрации можно выставить тип аккаунта, админ или обычный пользователь)

За страницу логина отвечает AuthController, регистрации – RegisterController, который так же перенаправляет на register-info при успешной (/register-success) или неуспешной (/register-error)

Также появилась кнопка Logout когда пользователь уже авторизировался, за это отвечает LogoutController

**Папка services:**

За все операции с пользователями отвечает класс AuthenticateService – он работает с кукис, хранящие информацию и пользователе, производит проверку правильности введённого пароля и обрабатывает JWT токены с помощью JwtService. (JWT (JSON Web Token) токен – это штука для передачи и данных для аутентификации в клиент-серверных приложениях). JWT условно шифрует данные с помощью секретного ключа private static final String SECRET = "LcW4TaUC1H" (чтобы не передавать между клиентом и сервером пароли простыми строками, так делать нельзя). В базе данных хранится пароль захэшированный.

import org.mindrot.jbcrypt.BCrypt;

public class PasswordUtils {

public static String hashPassword(String plainTextPassword) {

return BCrypt.hashpw(plainTextPassword, BCrypt.gensalt());

}

public static boolean checkPassword(String plainTextPassword, String hashedPassword) {

return BCrypt.checkpw(plainTextPassword, hashedPassword);

}

}

Класс для хэширования и проверки совпадения хэша пароля (при проверке аунтефикации)

UserService – абстракция работы с таблицей юзеров, там есть добавление и поиск пользователя по имени.

GuestSessionService хранит список всех пользователей которые находятся на сайте, но не зарегистрированы. Тоже через JWT токен.