**Курсовой проект на тему : «Система аренды недвижимости»»**

Веб-приложение на Java с использованием Spring Boot, предназначенное для управления недвижимостью, пользователями, риэлторами и сделками.

В нем реализованы функции регистрации, аутентификации, работы с объектами недвижимости, запросами клиентов, профилями риэлторов и пользователей, а также управление сделками и платежами.

**Цели и задачи**

Разработка реляционной базы данных, обеспечивающей хранение и управление данными о недвижимости, пользователях, риэлторов. База данных должна минимизировать избыточность данных, обеспечивать целостность информации и поддерживать возможность масштабирования.

Для осуществления данной работы следует выполнить следующие задачи:

1. Спроектировать структуру базы данных.
2. Обеспечить взаимосвязь таблиц для эффективного управления данными.
3. Реализовать функции управления данными: создание, обновление и удаление записей.

Реализованная база данных станет основой серверной части информационной системы автошколы, обеспечивая быстрый доступ к данным, согласованное взаимодействие всех компонентов.

**Структура базы данных**

Структура базы данных, включающая следующие сущности:

* + Недвижимость (apartaments)

Хранит информацию об объектах недвижимости, доступных для покупки или продажи. Это может быть квартира, дом, коммерческое помещение и т.д. (id, цена, кв метров, тип недвижимости, расположение)

* + Сделки (payments)

Фиксирует факт сделки между клиентом и риэлтором по определённому объекту недвижимости.. (id, id риэлтора, id клиента, id недвижимости, Дата сделки)

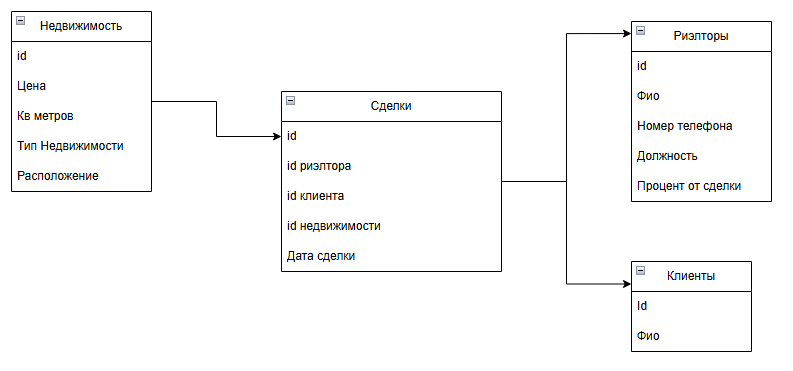
* + Клиенты (users)

Содержит данные о клиентах, которые участвуют в сделках. (id, Фио)

* + Риелторы (rental\_requests)

Хранит информацию о сотрудниках агентства недвижимости, которые занимаются сопровождением сделок. (id, Фио, номер телефонп, должность, процент от сделки)

**ER диаграмма**



**Анализ связей между таблицами**

Блок-схема показывает следующие отношения:

1. Сделки **→** Недвижимость

Каждая сделка привязана к одному объекту недвижимости..

1. Сделки **→** Риэлторы

Сделку сопровождает один риэлтор.

1. Сделки **→** Клиенты

Сделку совершает один клиент.

**Физическая модель**A computer screen shot of a computer program

AI-generated content may be incorrect.

Таблицы и их связи:

1. users

Основной справочник пользователей.

Поля: idusers, email, full\_name, password, phone\_number, role, username.

Связи:

* apartments.owner\_id → users.idusers — владелец квартиры.
* rental\_requests.tenant\_id → users.idusers — арендатор (запрос аренды).
* payments.user\_id → users.idusers — пользователь, совершивший платёж.

2. apartments

Хранит данные о квартирах.

Поля: id, address, city, description, monthly\_rent, status, number\_of\_rooms, total\_area, owner\_id.

Связи:

* owner\_id → users.idusers — кто владеет квартирой.
* rental\_requests.apartment\_id → apartments.id — на какую квартиру подан запрос.
* payments.apartment\_id → apartments.id — за какую квартиру сделан платёж.

3. rental\_requests

Хранит информацию о запросах на аренду.

Поля: id, admin\_comment, comment, end\_date, request\_date, start\_date, status, apartment\_id, tenant\_id.

Связи:

* apartment\_id → apartments.id — арендуемая квартира.
* tenant\_id → users.idusers — пользователь, сделавший запрос.

4. payments

Хранит информацию об оплатах.

Поля: id, amount, description, payment\_date, payment\_method, status, apartment\_id, user\_id.

Связи:

* apartment\_id → apartments.id — платёж за квартиру.
* user\_id → users.idusers — пользователь, оплативший.

5. realtors

информация о риелторах (связаны с пользователями, есть комиссия, статус активности и т.д.).

Поля: id, commission\_percentage, full\_name, is\_active, phone\_number, position, user\_id

Связи:

* user\_id → users.id (каждый риелтор — это пользователь)
* deals.realtor\_id → realtors.id (риелтор участвует в сделке)
* property\_requests.realtor\_id → realtors.id (риелтор обрабатывает заявку)

6. clients

Данные клиентов, которые могут подавать заявки на объекты.

Поля: id, full\_name

Связи:

* property\_requests.client\_id → clients.id (клиент подаёт заявку)

7. properties

объекты недвижимости, выставленные на продажу или аренду.

Поля: id, address, area, description, image\_filename, price , rooms, status, title, type, owner\_id

* Связи:   
  owner\_id → users.id (владелец недвижимости)
* deals.property\_id → properties.id (объект участвует в сделке)
* property\_requests.property\_id → properties.id (по объекту подают заявки)

8. deals

сделки по продаже недвижимости между продавцом и покупателем с участием риелтора.

Поля: id, created\_at, deal\_amount, deal\_date, description, documents, status, updated\_at, buyer\_id, property\_id, realtor\_id, seller\_id

* buyer\_id → users.id (покупатель)
* seller\_id → users.id (продавец)
* property\_id → properties.id (объект сделки)
* realtor\_id → realtors.id (риелтор сделки)

9. property\_requests

заявки клиентов на интересующие их объекты недвижимости.

Поля: id, created\_at, message, status, updated\_at, client\_id, property\_id, realtor\_id

* client\_id → clients.id (клиент-заявитель)
* property\_id → properties.id (объект заявки)
* realtor\_id → realtors.id (риелтор, обрабатывающий заявку)

Таблицы и их поля

**1. users**

* id — уникальный идентификатор пользователя
* email — email пользователя
* full\_name — полное имя
* password — пароль
* phone — телефон
* role — роль (например, admin, user, realtor)
* username — имя пользователя

**2. realtors**

* id — уникальный идентификатор риелтора
* commission\_percentage — процент комиссии
* full\_name — имя риелтора
* is\_active — активен ли риелтор
* phone\_number — телефон
* position — должность
* user\_id — внешний ключ на users.id (связь с пользователем)

**3. clients**

* id — уникальный идентификатор клиента
* full\_name — имя клиента

**4. properties**

* id — уникальный идентификатор недвижимости
* address — адрес
* area — площадь
* description — описание
* image\_filename — имя файла изображения
* price — цена
* rooms — количество комнат
* status — статус (ENUM)
* title — заголовок
* type — тип недвижимости (ENUM)
* owner\_id — внешний ключ на users.id (владелец)

**5. apartments**

* id — уникальный идентификатор квартиры
* address — адрес
* area — площадь
* city — город
* description — описание
* price\_per\_month — цена за месяц
* rooms — количество комнат
* status — статус (ENUM)
* owner\_id — внешний ключ на users.id (владелец)

**6. deals**

* id — уникальный идентификатор сделки
* created\_at — дата создания
* deal\_amount — сумма сделки
* deal\_date — дата сделки
* description — описание
* documents — документы
* status — статус (ENUM)
* updated\_at — дата обновления
* buyer\_id — внешний ключ на users.id (покупатель)
* property\_id — внешний ключ на properties.id
* realtor\_id — внешний ключ на realtors.id
* seller\_id — внешний ключ на users.id (продавец)

**7. payments**

* id — уникальный идентификатор платежа
* amount — сумма
* description — описание
* payment\_date — дата платежа
* status — статус (ENUM)
* rental\_request\_id — внешний ключ на rental\_requests.id
* user\_id — внешний ключ на users.id

**8. property\_requests**

* id — уникальный идентификатор запроса на недвижимость
* created\_at — дата создания
* message — сообщение
* status — статус (ENUM)
* updated\_at — дата обновления
* client\_id — внешний ключ на clients.id
* property\_id — внешний ключ на properties.id
* realtor\_id — внешний ключ на realtors.id

**9. rental\_requests**

* id — уникальный идентификатор запроса на аренду
* comment — комментарий
* end\_date — дата окончания аренды
* start\_date — дата начала аренды
* status — статус (ENUM)
* apartment\_id — внешний ключ на apartments.id
* tenant\_id — внешний ключ на users.id

Примеры кода

1.Модель пользователя (User.java)

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String username;

private String password;

@Enumerated(EnumType.STRING)

private UserRole role;

// геттеры и сеттеры

}

1. Сервис регистрации пользователя (UserService.java)

@Service

public class UserService {

@Autowired

private UserRepository userRepository;

@Autowired

private PasswordEncoder passwordEncoder;

public void registerNewUser(User user) {

if (userRepository.findByUsername(user.getUsername()).isPresent()) {

throw new RuntimeException("User already exists");

}

user.setPassword(passwordEncoder.encode(user.getPassword()));

userRepository.save(user);

}

}

1. Контроллер аутентификации (AuthController.java)

@Controller

@RequestMapping("/auth")

public class AuthController {

@Autowired

private UserService userService;

@PostMapping("/register")

public String register(@ModelAttribute User user) {

userService.registerNewUser(user);

return "redirect:/auth/login";

}

}

1. Репозиторий недвижимости (PropertyRepository.java)

public interface PropertyRepository extends JpaRepository<Property, Long> {

List<Property> findByStatus(PropertyStatus status);

}

5. Тест на регистрацию пользователя (UserServiceTest.java)

@Test

void registerNewUser\_success() {

User user = new User();

user.setUsername("testuser");

user.setPassword("password");

when(userRepository.findByUsername("testuser")).thenReturn(Optional.empty());

when(passwordEncoder.encode("password")).thenReturn("hashed");

when(userRepository.save(any(User.class))).thenReturn(user);

userService.registerNewUser(user);

assertEquals("hashed", user.getPassword());

verify(userRepository, times(1)).save(user);

}

Юнит тесты

1. UserServiceTest:

* Регистрация нового пользователя (успешная и если пользователь уже существует).
* Поиск пользователя по имени.
* Проверка, что пароли хешируются.
* Проверка, что при попытке зарегистрировать уже существующего пользователя выбрасывается исключение.

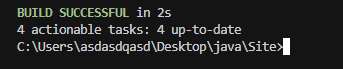
1. PropertyRequestServiceTest:

* Создание нового запроса на недвижимость.
* Поиск запросов по риэлтору.
* Поиск запросов по клиенту.
* Обновление статуса запроса (успешно и случай, когда запрос не найден — выбрасывается исключение).

1. DemoApplicationTests:

* Проверка, что Spring Boot приложение успешно поднимается (contextLoads).

Таким образом, тесты покрывают основные сервисы, связанные с пользователями и запросами на недвижимость, а также базовую работоспособность приложения. Все тесты прошли успешно.



Кратко о возможностях проекта

* Регистрация и вход пользователей (клиенты, риэлторы, администраторы)
* Просмотр, создание, редактирование и удаление объектов недвижимости
* Оформление и обработка запросов на аренду/покупку недвижимости
* Управление сделками и платежами
* Профили пользователей и риэлторов
* Защита данных с помощью Spring Security

**Итог**

Разработанная база данных успешно решает задачи учета и управления процессами аренды недвижимости.