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)

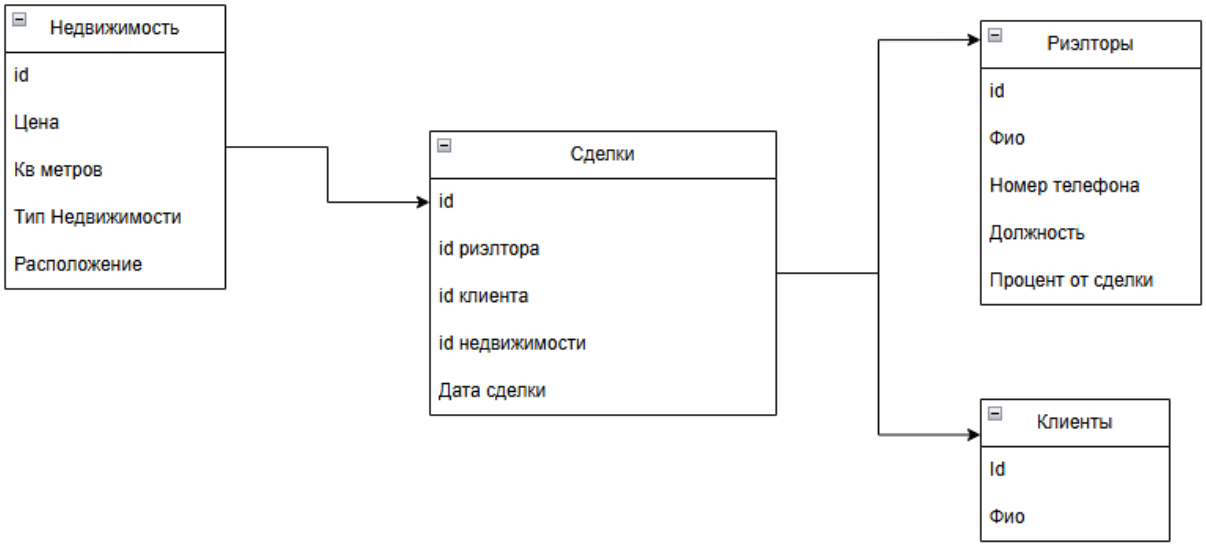
- Клиенты (users)

  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..

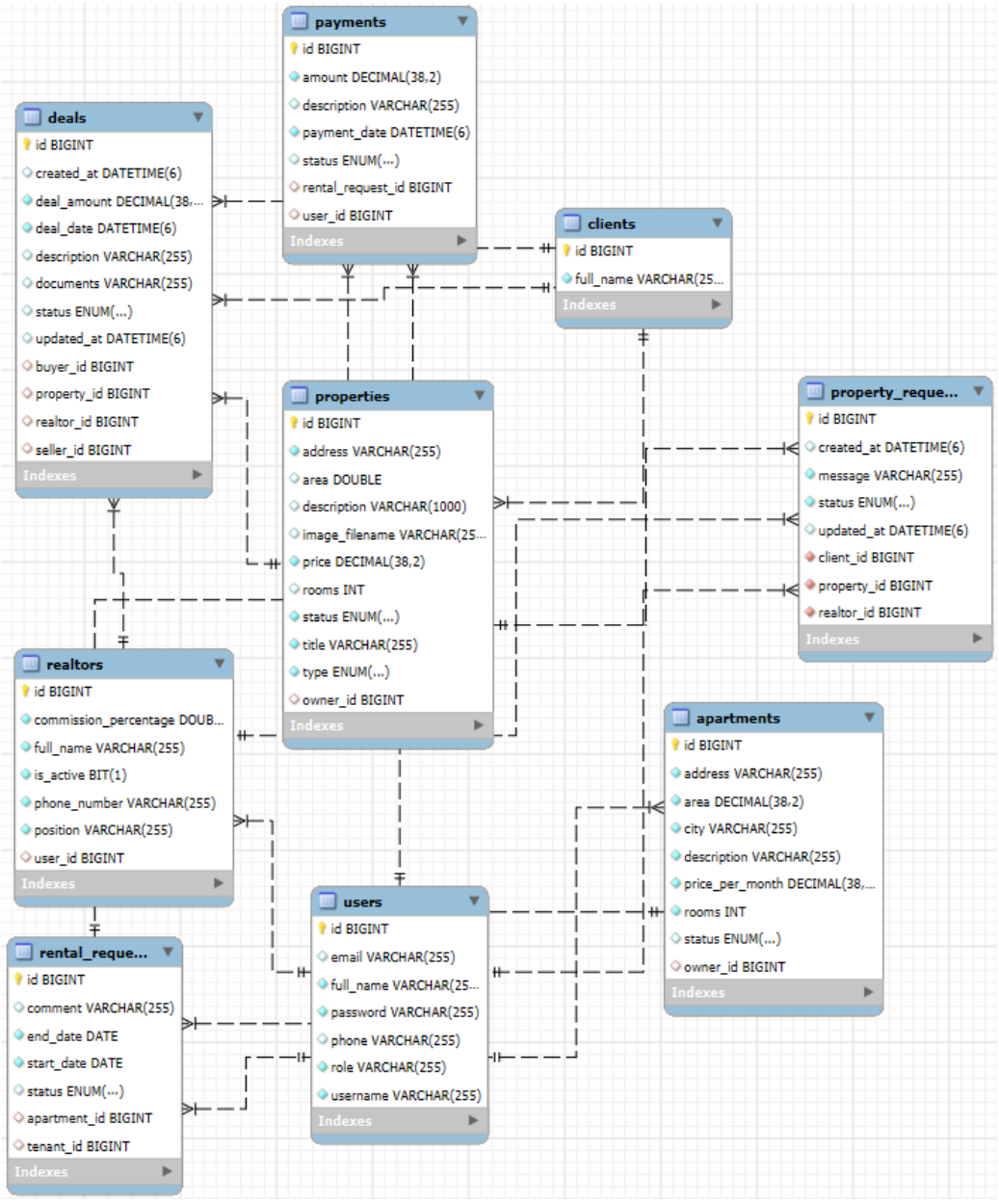
2. Сделки → Риэлторы

   Сделку сопровождает один риэлтор.

3. Сделки → Клиенты

   Сделку совершает один клиент.

Физическая модель



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
}
```

2. 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);
    }
}
```

3. 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";
    }
}
```

4. 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.em
pty());
   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.

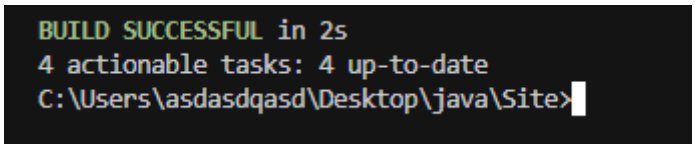
2. PropertyRequestServiceTest:

- 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).

3. 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.