

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ  
БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и  
автоматизированных систем

**КУРСОВОЙ ПРОЕКТ**

по дисциплине: Базы данных

тема: Агентство недвижимости

Автор работы \_\_\_\_\_ Гринева Марина Сергеевна ПВ-222

(подпись)

Руководитель проекта \_\_\_\_\_ Панченко Максим Владимирович

(подпись)

Оценка \_\_\_\_\_

Белгород, 2025

## Содержание

1 Введение.....	3
2 Основная часть.....	4
2.1 Постановка задачи.....	4
2.2 Проектирование базы данных .....	5
2.2.1 Анализ предметной области.....	5
2.2.2 Диаграмма «Сущность - Связь».....	6
2.2.3 Схема структуры базы данных .....	7
2.2.4 Нормализация базы данных .....	9
2.2.5 Описание процесса создания базы данных.....	12
2.4 Основной этап программы .....	12
2.3 Разработка автоматического резервного копирования базы данных.....	15
2.4 Результаты работы приложения.....	15
3 Заключение .....	23
4 Список литературы .....	24
5 Приложение .....	25

## Введение

В современном мире, где градостроительные проекты с большой скоростью, а границы между городами и регионами всё более прозрачны, рынок недвижимости превращается не только в одну из самых быстроразвивающихся отраслей экономики, но и в ключевой элемент социальной инфраструктуры. Агентства недвижимости, выступая посредником между спросом и предложением на покупку и аренду жилья, вынуждены оперативно управлять все более сложными бизнес-процессами: от подбора оптимальных вариантов до оформления договоров и выдачи документов.

Постоянно растущий спрос на разные форматы сделок — долгосрочная аренда, экспресс-аренда, первичный и вторичный рынок, инвестиционные проекты и элитная недвижимость — формирует высококонкурентную среду. Здесь решающую роль играют не только выгодные цены, но и скорость работы с данными, прозрачность взаимодействия с клиентом, эффективность внутреннего учета и своевременная генерация документов. Для успешного функционирования агентства недвижимости и обеспечения высокого уровня сервиса нужно применять современные инструменты автоматизации. В первую очередь — единую, хорошо спроектированную базу данных, где аккумулируются сведения об объектах (квартиры, дома, участки), клиентах и сотрудниках-агентах. Не менее важно иметь удобный интерфейс для создания, чтения, обновления и удаления записей с чётким разграничением прав, механизм миграций для отслеживания изменений структуры БД, а также встроенные функции экспорта и резервного копирования.

В связи с вышесказанным, данная курсовая работа посвящена разработке и внедрению приложения, основанного на системе управления базами данных (СУБД) MySQL с использованием языка Python и библиотеки `peewee` в качестве объектно-реляционного отображения (ORM). Программный комплекс обеспечивает автоматизированное управление ключевыми аспектами деятельности агента недвижимости: реализацией недвижимости, отслеживанием и учётом взаимоотношений с клиентами, управлением сделками, договорами, генерации документов, а так же созданием отчётности и анализа на основе накопленных данных

## **2. Основная часть**

### **2.1 Постановка задачи**

Целью данной курсовой работы является разработка программного обеспечения для агентства недвижимости, способного автоматизировать процессы управления данными, предоставить пользователю понятный интерфейс и повысить общую эффективность работы компании. Для достижения поставленной цели необходимо:

1. Разработать структуру базы данных, отражающую все ключевые сущности (клиенты, агенты, объекты недвижимости, сделки, договоры).
2. Реализовать графический интерфейс, обеспечивающий удобное взаимодействие с базой данных.
3. Обеспечить полноценную поддержку операций создания, чтения, обновления и удаления (CRUD) для всех сущностей в зависимости от ролей пользователей.
4. Внедрить систему ролей (администратор, агент) с разграничением прав доступа к функционалу приложения.
5. Реализовать механизм экспорта данных по отдельным таблицам и всего содержимого базы в форматы JSON и XLSX для дальнейшего анализа и отчётности.

## 2.2 Проектирование базы данных

### 2.2.1 Анализ предметной области

ФИО	Фамилия, имя, отчество клиента/агента
Паспортные данные	Серия, номер паспорта клиента/агента
Адрес	Город, улица, строение для указания адреса клиента/агента или объекта недвижимости
Персональные данные	Личные данные, адрес и данные для связи с клиентом/агентом
Клиент	Лицо, получающее услугу по приобретению недвижимости
Агент	Работник агентства недвижимости
Недвижимость	Объект недвижимости, включает описывающие его атрибуты и статус (в аренде/продан/свободен)
Участок	Земельный определенного типа участок
Дом	Строение с описанием этажности
Квартира	Квартира с атрибутами, ее описывающими
Сделка	Связывает агента, клиента и недвижимость договором, отражает статус процесса подписания договора (в процессе, завершен, отменен)
Договор	Фиксирует, что объект недвижимости участвует в сделке, описывает, на каких условиях
Продажа	Договор продажи включает дату приобретения недвижимости

Аренда	Договор аренды включает месячную стоимость аренды, дату начала и окончания действия договора.
--------	---

Связи между сущностями, которые отражает база данных:

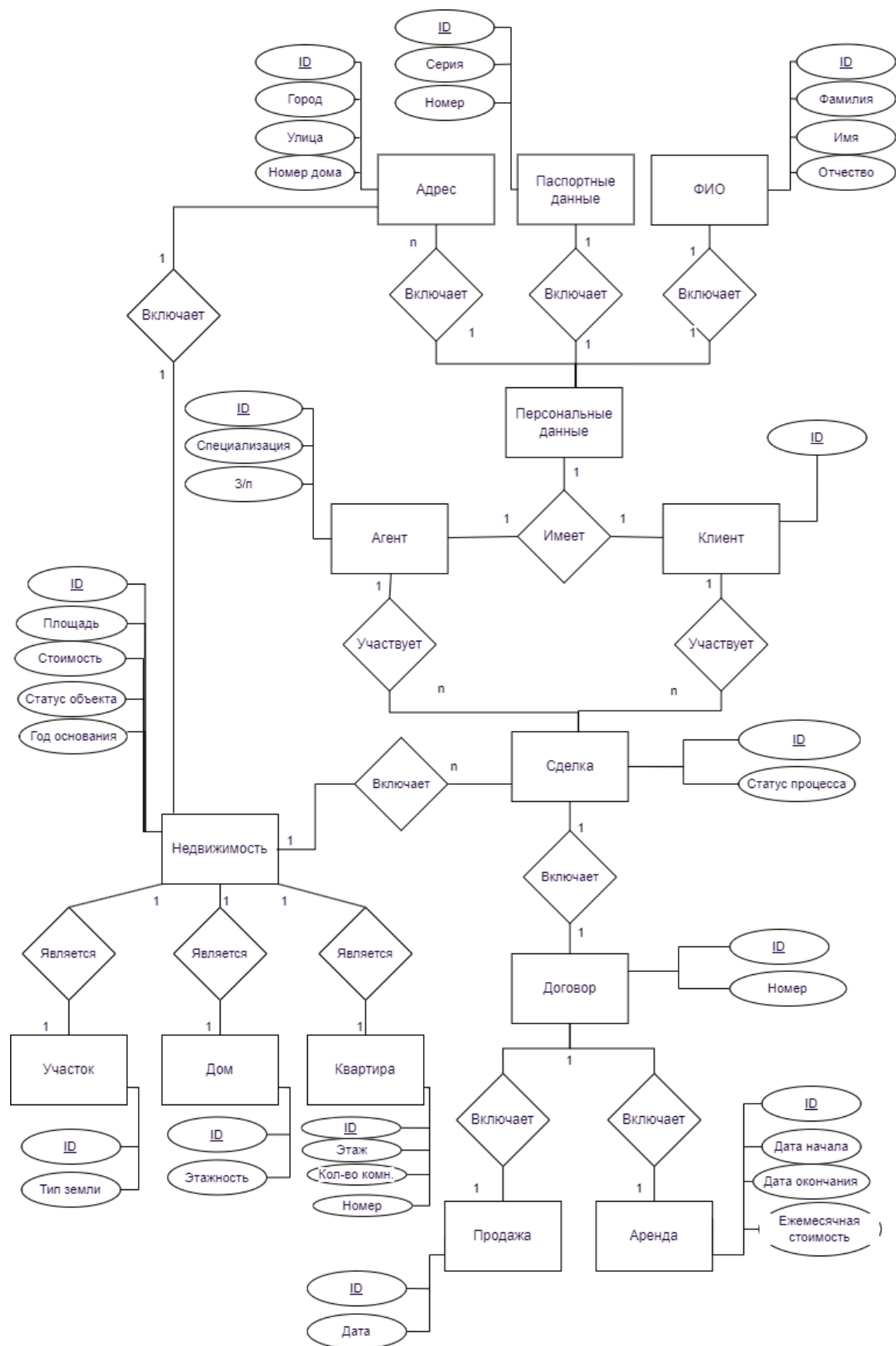
У клиента и агента есть **персональные данные**, которые включают в себя **ФИО**, **паспортные данные**, **адрес**.

Объект недвижимости представляет собой **квартиру**, **земельный участок** или **дом**, может быть связан с несколькими сделками (например, объект сначала арендовался, а затем был продан).

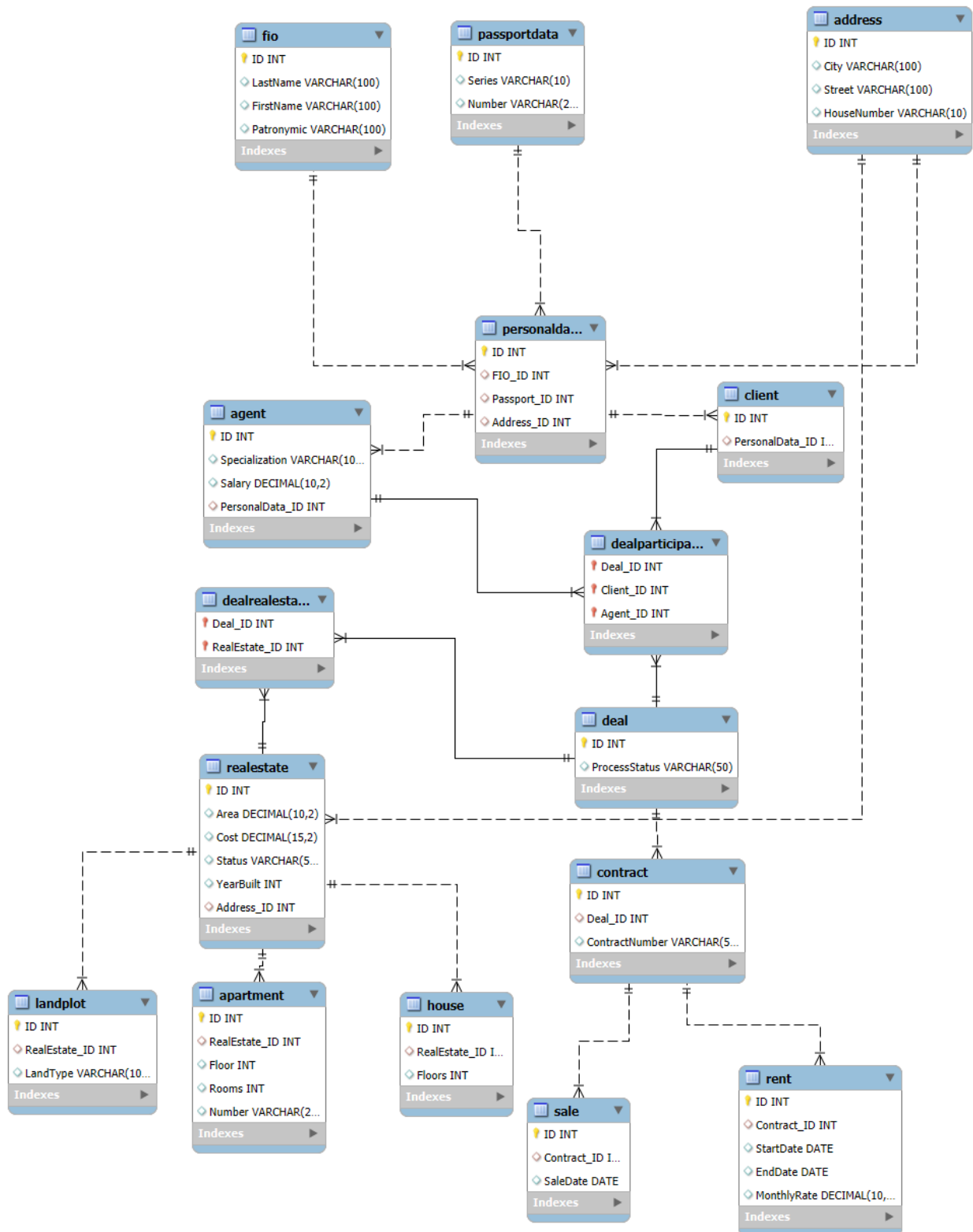
**Агент** от лица компании заключает с **клиентом сделку**, связанную с выбранной **недвижимостью** и формой ее приобретения. Клиент может участвовать в нескольких сделках, выступая покупателем или арендатором.

**Договор** фиксирует условия **сделки**. Различается договор **аренды** и договор **продажи**.

## 2.2.2 Диаграмма «Сущность - Связь»



### 2.2.3 Схема структуры базы данных





## 2.2.4 Нормализация базы данных

### Первая нормальная форма (1НФ)

- Все атрибуты каждой таблицы являются атомарными: например, адрес разбит на отдельные поля «Город», «Улица», «Номер дома».
- В таблицах нет повторяющихся групп и списков: у каждого клиента, агента или объекта недвижимости хранятся ровно те атрибуты, которые описаны в схеме, и каждое значение занимает свою ячейку.

### Вторая нормальная форма (2НФ)

- База данных уже находится в 1НФ.
- Во всех таблицах отсутствуют частичные зависимости неключевых атрибутов от части составного ключа.

### Третья нормальная форма (3НФ)

- База данных уже находится во 2НФ.
- Каждый неключевой атрибут зависит напрямую от первичного ключа своей таблицы, без транзитивных зависимостей.

Таким образом, спроектированная структура удовлетворяет требованиям 1НФ, 2НФ и 3НФ, что гарантирует минимизацию избыточности и отсутствие аномалий при вставке, обновлении или удалении данных.

## 2.2.5 Описание процесса создания базы данных

Создали новую БД на локальном сервере. Путем написания запроса

```
CREATE DATABASE IF NOT EXISTS realestateagency;
```

Далее каждую сущность создавали так же – SQL запрос

```

CREATE DATABASE IF NOT EXISTS RealEstateAgency;
USE RealEstateAgency;

CREATE TABLE FIO (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    LastName VARCHAR(100),
    FirstName VARCHAR(100),
    Patronymic VARCHAR(100)
);

CREATE TABLE PassportData (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Series VARCHAR(10),
    Number VARCHAR(20)
);

CREATE TABLE Address (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    City VARCHAR(100),
    Street VARCHAR(100),
    HouseNumber VARCHAR(10)
);

CREATE TABLE PersonalData (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    FIO_ID INT,
    Passport_ID INT,
    Address_ID INT,
    FOREIGN KEY (FIO_ID) REFERENCES FIO(ID),
    FOREIGN KEY (Passport_ID) REFERENCES PassportData(ID),
    FOREIGN KEY (Address_ID) REFERENCES Address(ID)
);

CREATE TABLE Client (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    PersonalData_ID INT,
    FOREIGN KEY (PersonalData_ID) REFERENCES PersonalData(ID)
);

CREATE TABLE Agent (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Specialization VARCHAR(100),
    Salary DECIMAL(10,2),

```

```

        PersonalData_ID INT,
        FOREIGN KEY (PersonalData_ID) REFERENCES PersonalData(ID)
    );

CREATE TABLE RealEstate (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Area DECIMAL(10,2),
    Cost DECIMAL(15,2),
    Status VARCHAR(50),
    YearBuilt INT,
    Address_ID INT,
    FOREIGN KEY (Address_ID) REFERENCES Address(ID)
);

CREATE TABLE LandPlot (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    RealEstate_ID INT,
    LandType VARCHAR(100),
    FOREIGN KEY (RealEstate_ID) REFERENCES RealEstate(ID)
);

CREATE TABLE House (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    RealEstate_ID INT,
    Floors INT,
    FOREIGN KEY (RealEstate_ID) REFERENCES RealEstate(ID)
);

CREATE TABLE Apartment (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    RealEstate_ID INT,
    Floor INT,
    Rooms INT,
    Number VARCHAR(20),
    FOREIGN KEY (RealEstate_ID) REFERENCES RealEstate(ID)
);

CREATE TABLE Deal (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    ProcessStatus VARCHAR(50)
);

CREATE TABLE DealParticipant (

```

```

        Deal_ID INT,
        Client_ID INT,
        Agent_ID INT,
        PRIMARY KEY (Deal_ID, Client_ID, Agent_ID),
        FOREIGN KEY (Deal_ID) REFERENCES Deal(ID),
        FOREIGN KEY (Client_ID) REFERENCES Client(ID),
        FOREIGN KEY (Agent_ID) REFERENCES Agent(ID)
    );

CREATE TABLE DealRealEstate (
    Deal_ID INT,
    RealEstate_ID INT,
    PRIMARY KEY (Deal_ID, RealEstate_ID),
    FOREIGN KEY (Deal_ID) REFERENCES Deal(ID),
    FOREIGN KEY (RealEstate_ID) REFERENCES RealEstate(ID)
);

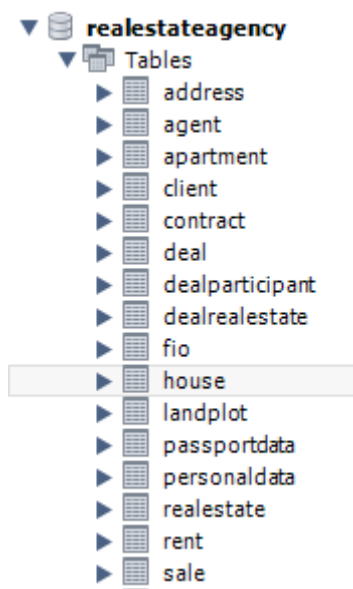
CREATE TABLE Contract (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Deal_ID INT,
    ContractNumber VARCHAR(50),
    FOREIGN KEY (Deal_ID) REFERENCES Deal(ID)
);

CREATE TABLE Sale (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Contract_ID INT,
    SaleDate DATE,
    FOREIGN KEY (Contract_ID) REFERENCES Contract(ID)
);

CREATE TABLE Rent (
    ID INT PRIMARY KEY AUTO_INCREMENT,
    Contract_ID INT,
    StartDate DATE,
    EndDate DATE,
    MonthlyRate DECIMAL(10,2),
    FOREIGN KEY (Contract_ID) REFERENCES Contract(ID)
);

```

**Результат:**



## 2.4 Основной этап программы

Данное программное обеспечение представляет собой десктопное приложение, разработанное с использованием объектно-ориентированного подхода на языке Python 3.11. В качестве ядра приложения используется ORM Peewee, который обеспечивает эффективное, простое и безопасное взаимодействие с реляционной базой данных MySQL.

```
from peewee import Model, CharField, IntegerField, ForeignKeyField,
MySQLDatabase

db = MySQLDatabase(
    'realestate_agency',
    user='root',
    password='your_password',
    host='localhost',
    port=3306
)

class BaseModel(Model):
    class Meta:
        database = db

class PersonalData(BaseModel):
    last_name      = CharField()          # Фамилия
    first_name     = CharField()          # Имя
    patronymic     = CharField(null=True) # Отчество
    passport_series = IntegerField()       # Серия паспорта
    passport_number = IntegerField()       # Номер паспорта
    city           = CharField()          # Город
    street         = CharField()          # Улица
    house_number   = CharField()          # Дом
    apartment      = CharField(null=True) # Квартира

class Agent(BaseModel):
    personal_data = ForeignKeyField(PersonalData, backref='agents')
    specialization = CharField()          # Специализация агента
    salary         = IntegerField()       # Зарплата

class Client(BaseModel):
```

```
personal_data = ForeignKeyField(PersonalData, backref='clients')
```

При этом достигается независимость логики приложения от конкретных особенностей используемой СУБД.

Пользовательский интерфейс построен с использованием библиотеки Tkinter и стилового движка ttk, что позволяет создавать кросс-платформенный, удобный и интуитивно понятный интерфейс с возможностью тонкой настройки его внешнего вида.

```
class MainApp:
    def __init__(self, master, user):
        self.master = master
        self.master.title("Агентство недвижимости — Главное меню")
        self.master.geometry("400x500")
        self.user = user

    ttk.Label(master, text=f"Добро пожаловать, {user.username}!", font=("Arial",
        14)).pack(pady=10)
        ttk.Label(master, text=f"Роль: {user.role}", font=("Arial",
10)).pack(pady=5)

        if user.role == 'admin':
            self.add_button("Клиенты", self.open_clients, "primary")
            self.add_button("Агенты", self.open_agents, "primary")

        self.add_button("Недвижимость", self.open_realestate, "secondary")
        self.add_button("Сделки", self.open_deals, "secondary")
```

Для реализации функциональности импорта данных и представления их в наиболее востребованных форматах используется библиотека pandas, для записи файлов в формате Excel и встроенный модуль json для JSON.

```
import json
import pandas as pd
from models import *

EXPORT_FOLDER = "exports"

def export_clients():
    data = []
    for c in Client.select():
        fio = c.personal_data.fio
        addr = c.personal_data.address
        passport = c.personal_data.passport
        data.append({
            "id": c.id,
            "ФИО": f"{fio.last name} {fio.first name} {fio.patronymic or ''}",
            "Паспорт": f"{passport.series} {passport.number}",
            "Адрес": f"{addr.city}, {addr.street}, {addr.house number}"
        })
    df = pd.DataFrame(data)
    df.to_excel(f"{EXPORT_FOLDER}/clients.xlsx", index=False)
    df.to_json(f"{EXPORT_FOLDER}/clients.json", force_ascii=False,
indent=4)
    return True

def export_all_to_json():
    all_data = {}

    def get_all_rows(model):
        return [dict(row. data ) for row in model.select()]

    all_data["clients"] = get_all_rows(Client)
    all_data["agents"] = get_all_rows(Agent)
    all_data["real estate"] = get_all_rows(RealEstate)
    all_data["deals"] = get_all_rows(Deal)
    all_data["contracts"] = get_all_rows(Contract)
    all_data["sales"] = get_all_rows(Sale)
    all_data["rents"] = get_all_rows(Rent)

    with open(f"{EXPORT_FOLDER}/all data.json", "w", encoding="utf-8") as
f:
        json.dump(all_data, f, ensure_ascii=False, indent=4)
    return True
```



В приложении реализована работа ролями пользователей:

Администратор, пользователь, с разграничением доступа к функционалу, для более эффективного распределения ответственности.

```
from getpass import getpass
from models import mysql_db, User

def register_user():
    print("Регистрация пользователя")
    username = input("Имя пользователя: ")
    password = getpass("Пароль: ")
    role = input("Роль (admin / agent): ").lower()

    if role not in ('admin', 'agent'):
        print("Неверная роль. Пользователь не создан.")
        return

    try:
        User.create(username=username, password=password, role=role)
        print("Пользователь успешно зарегистрирован.\n")
    except:
        print("Ошибка: пользователь с таким именем уже существует.\n")

def login():
    print("Вход")
    username = input("Имя пользователя: ")
    password = getpass("Пароль: ")

    try:
        user = User.get(User.username == username)
        if user.password == password:
            print(f"Добро пожаловать, {user.username}! Ваша роль: {user.role}")
            return user
        else:
            print("Неверный пароль.")
    except User.DoesNotExist:
        print("Пользователь не найден.")
    return None

def list_users(current_user):
    if current_user.role != 'admin':
        print("Только администраторы могут просматривать список пользователей.\n")
    return

    print("Список пользователей:")
    for user in User.select():
        print(f"{user.id}. {user.username} ({user.role})")
    print()

def menu():
    mysql_db.connect()
    current_user = None
    while True:
        print("""\n=== Меню пользователя ===
1. Регистрация
```

```

2. Вход
3. Список пользователей
0. Выход"""")
    choice = input("Выберите действие: ")

    if choice == '1':
        register_user()
    elif choice == '2':
        current_user = login()
    elif choice == '3':
        if current_user:
            list_users(current_user)
        else:
            print("Сначала выполните вход.")
    elif choice == '0':
        break
    else:
        print("Неверный выбор.\n")

if __name__ == '__main__':
    menu()

```

## 2.5 Разработка автоматического резервного копирования базы данных

Приложение позволяет выбрать папку и задать интервал (в секундах, минутах или часах). Кнопка «Остановить» прерывает таймер и останавливает дальнейшие резервные копирования:

```

import ttkbootstrap as ttk
from ttkbootstrap.constants import *
from tkinter import messagebox, filedialog
import threading
import datetime
import os
import json
from models import *

class BackupGUI:
    def __init__(self, master):
        self.master = master
        self.master.title("Автоматическое резервное копирование")
        self.master.geometry("400x350")

```

```

self.backup_folder = None
self.backup_timer = None
self.interval = 3600 # по умолчанию 1 час

ttk.Label(master, text=" Резервное копирование", font=("Arial",
16)).pack(pady=10)

# Выбор папки
ttk.Button(master, text=" Выбрать папку для резервных копий",
bootstyle="secondary", command=self.choose_backup_folder).pack(pady=10)

# Интервал
ttk.Label(master, text="🕒 Интервал резервного
копирования:").pack()
interval_frame = ttk.Frame(master)
interval_frame.pack(pady=5)

self.interval_entry = ttk.Entry(interval_frame, width=10)
self.interval_entry.insert(0, "10") # по умолчанию 10
self.interval_entry.pack(side=ttk.LEFT)

self.interval_unit_var = ttk.StringVar(value="секунды")
ttk.Combobox(interval_frame,
               textvariable=self.interval_unit_var,
               values=["секунды", "минуты", "часы"],
               state="readonly",
               width=10).pack(side=ttk.LEFT, padx=5)

# Кнопки управления
btn_frame = ttk.Frame(master)
btn_frame.pack(pady=20)

ttk.Button(btn_frame, text="Начать", bootstyle="success",
command=self.start_backup).pack(side=ttk.LEFT, padx=5)
ttk.Button(btn_frame, text="Остановить", bootstyle="danger",
command=self.stop_backup).pack(side=ttk.LEFT, padx=5)

def choose_backup_folder(self):
    self.backup_folder = filedialog.askdirectory()
    if self.backup_folder:
        messagebox.showinfo("Готово", f"Выбрана
папка:\n{self.backup_folder}")

```

```

def start_backup(self):
    if not self.backup_folder:
        messagebox.showerror("Ошибка", "Сначала выберите папку для
сохранения")
        return

    try:
        val = int(self.interval_entry.get())
        unit = self.interval_unit_var.get()

        if unit == "секунды":
            self.interval = val
        elif unit == "минуты":
            self.interval = val * 60
        elif unit == "часы":
            self.interval = val * 60 * 60
    except:
        messagebox.showerror("Ошибка", "Введите корректный
интервал")
        return

    self.schedule_backup()
    messagebox.showinfo("Успех", f"Бэкап будет происходить каждые
{val} {unit}")

def schedule_backup(self):
    self.perform_backup()
    self.backup_timer = threading.Timer(self.interval,
self.schedule_backup)
    self.backup_timer.start()

def stop_backup(self):
    if self.backup_timer:
        self.backup_timer.cancel()
        self.backup_timer = None
        messagebox.showinfo("Остановлено", "Резервное копирование
остановлено")
    else:
        messagebox.showinfo("Неактивно", "Резервное копирование ещё
не запущено")

def perform_backup(self):

```

```

        if not self.backup_folder:
            return

        data = self.get_all_data()
        now = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
        path = os.path.join(self.backup_folder, f"backup_{now}.json")

        with open(path, "w", encoding="utf-8") as f:
            json.dump(data, f, ensure_ascii=False, indent=4)

    def get_all_data(self):
        def model_to_dict_list(model):
            return [dict(obj.__data__) for obj in model.select()]

        return {
            "clients": model_to_dict_list(Client),
            "agents": model_to_dict_list(Agent),
            "real_estate": model_to_dict_list(RealEstate),
            "deals": model_to_dict_list(Deal),
            "contracts": model_to_dict_list(Contract),
            "sales": model_to_dict_list(Sale),
            "rents": model_to_dict_list(Rent),
        }

    def run():
        win = ttk.Toplevel()
        BackupGUI(win)

if __name__ == '__main__':
    mysql_db.connect()
    app = ttk.Window(themename="flatly")
    BackupGUI(app)
    app.mainloop()

```

Для примера интервал между бэкапами 10 сек:

**Авто резервное копирование**

Выбрать папку для резервных копий

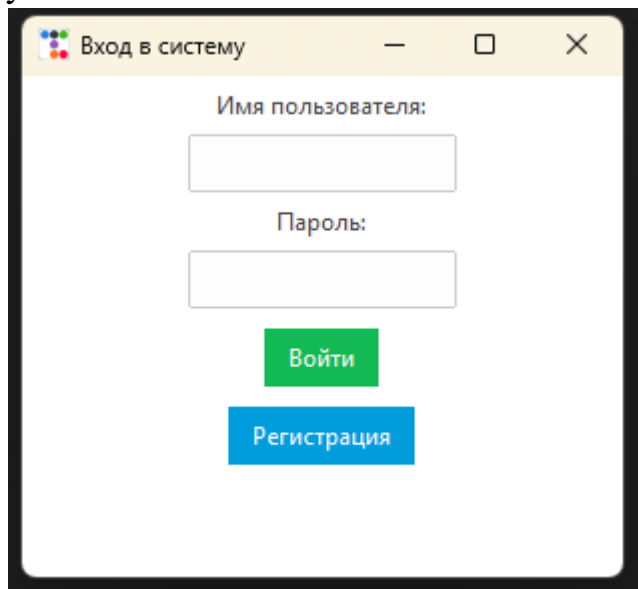
Интервал резервного копирования:

10 секунды

Начать резервное копирование    Остановить резервное копирование

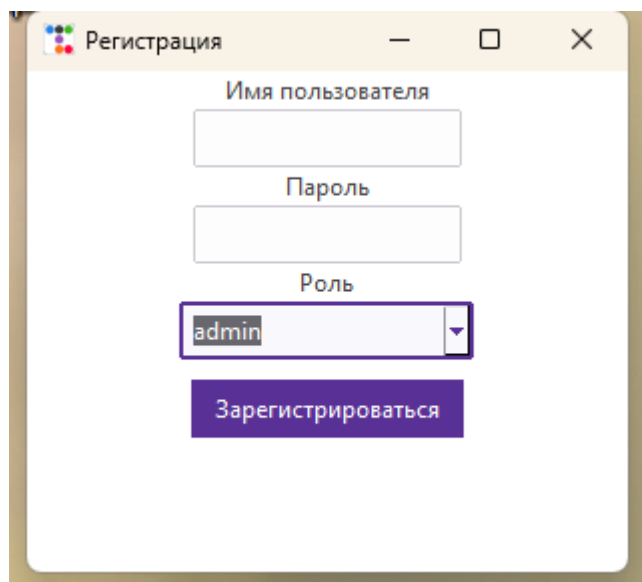
## 2.6 Результаты работы приложения

Вход в программу:



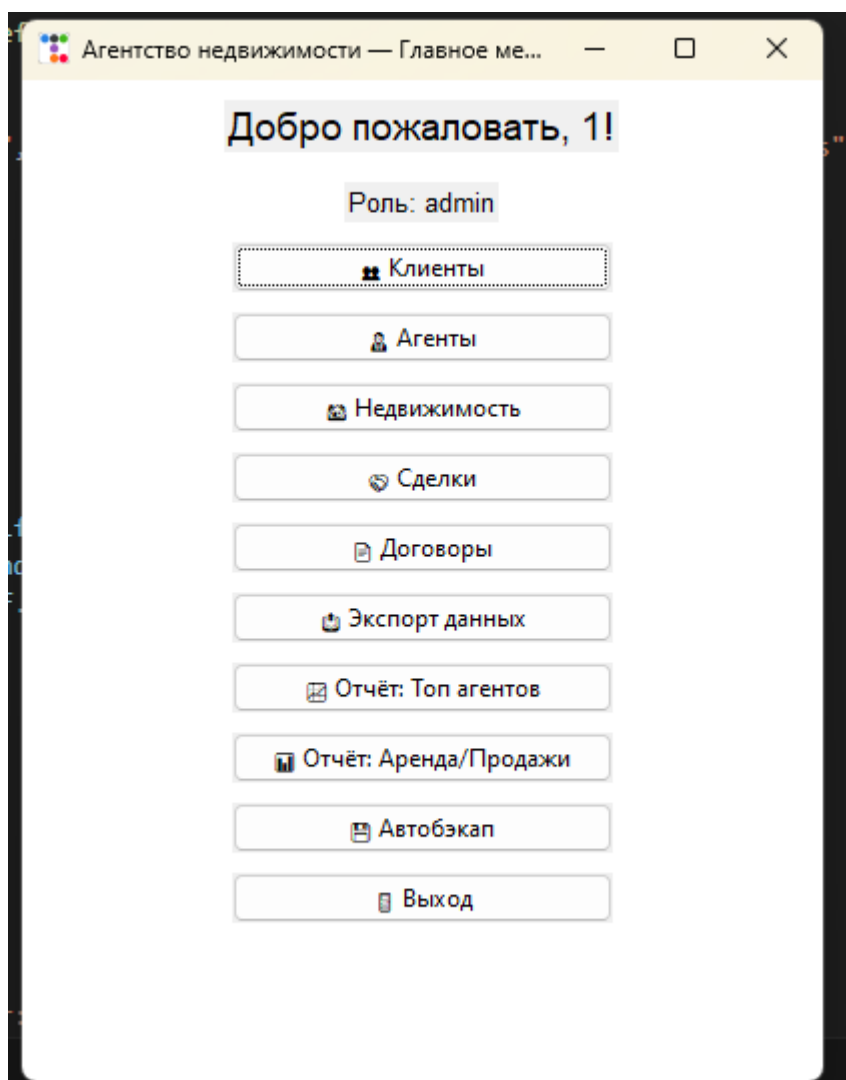
The screenshot shows a window titled "Вход в систему" (Login). It contains two text input fields: "Имя пользователя:" (Username) and "Пароль:" (Password). Below the password field are two buttons: a green "Войти" (Login) button and a blue "Регистрация" (Registration) button.

Регистрация нового пользователя:

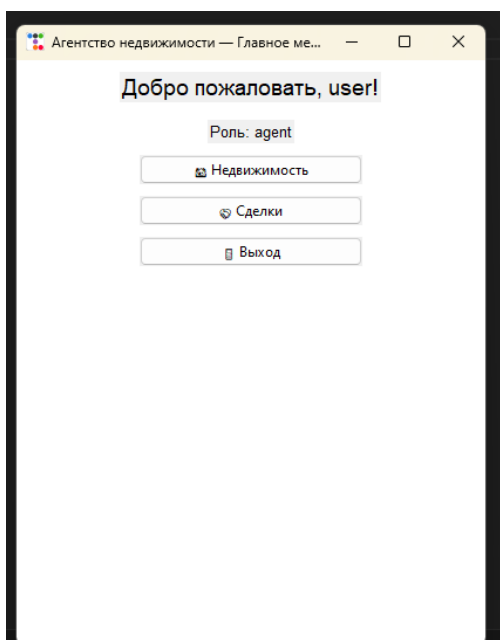


The screenshot shows a window titled "Регистрация" (Registration). It contains three input fields: "Имя пользователя" (Username), "Пароль" (Password), and "Роль" (Role). The "Роль" field is a dropdown menu with "admin" selected. Below these fields is a purple button labeled "Зарегистрироваться" (Register).

## Панель сотрудника (администратор)

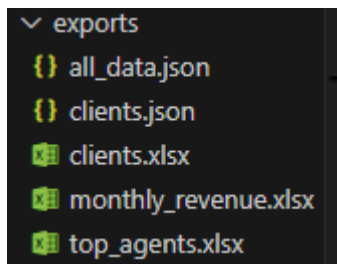


## Панель клиента:

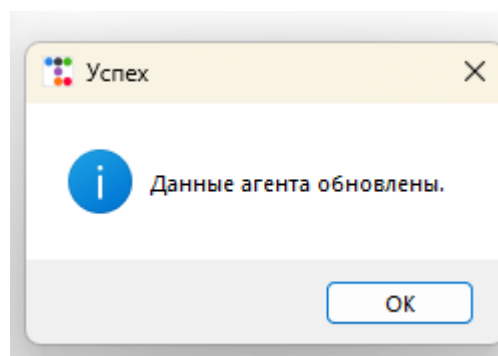
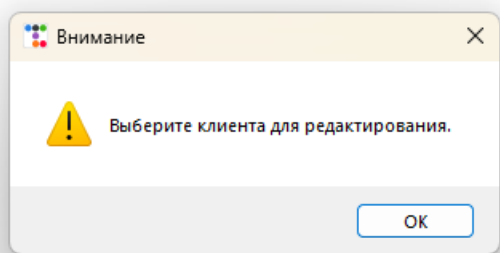
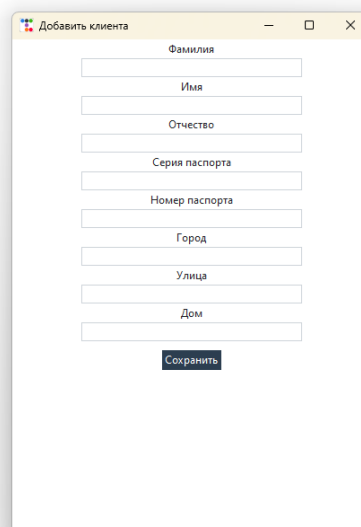
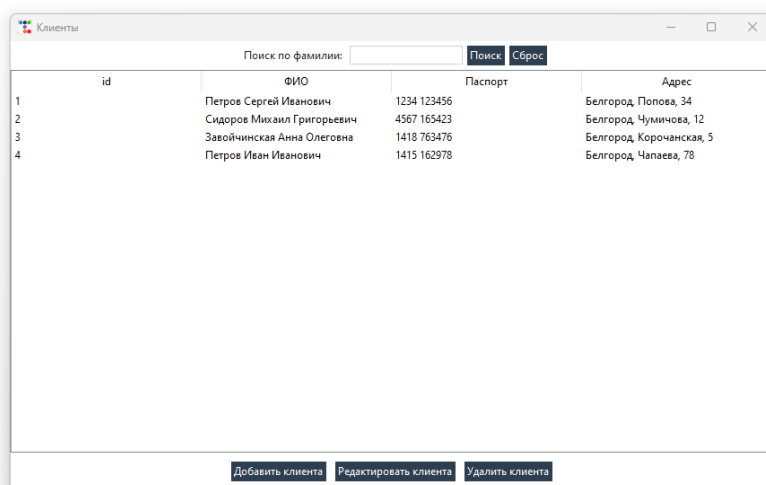




Экспорт данных:



Администратор может редактировать данные всех таблиц, также есть возможность поиска по таблице:



Агенты

Поиск по фамилии:

Поиск

Сброс

id	ФИО	Адрес	Паспорт	
1	Иванов Александр Александрович	Белгород, Есенина, 3	1418 999999	Прод:
2	Курочкина Ирина Витальевна	Белгород, Попова, 76	1416 457954	аренд

Добавить агента

Редактировать агента

Удалить агента

Редактировать агента

Фамилия

Иванов

Имя

Александр

Отчество

Александрович

Серия паспорта

1418

Номер паспорта

999999

Город

Белгород

Улица

Есенина

Дом

3

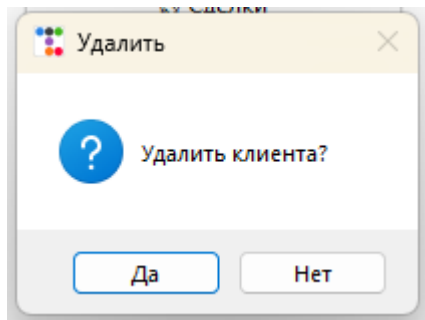
Специализация

Продажа

Зарплата

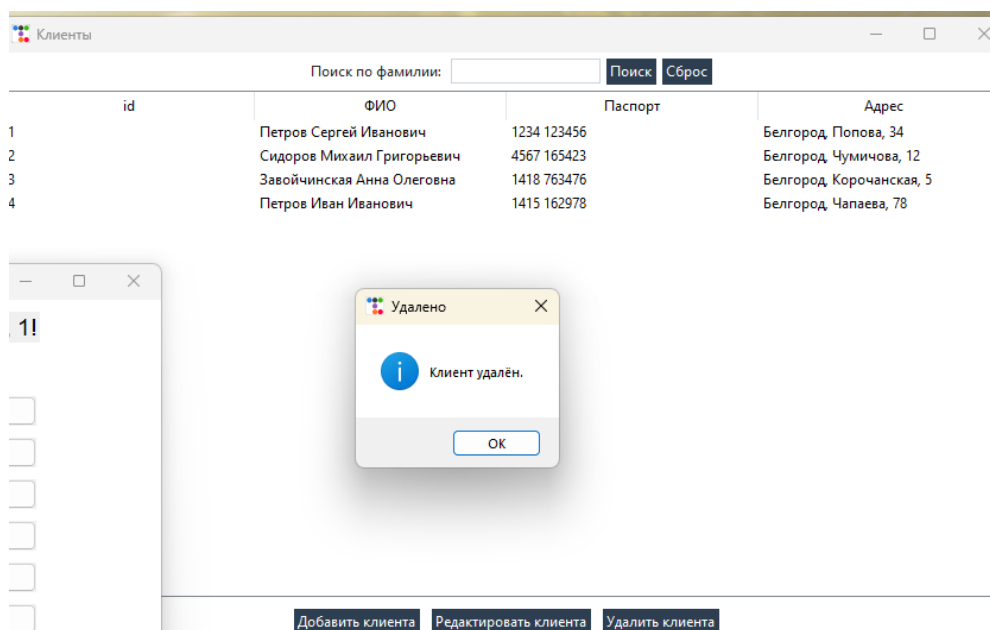
56008.00

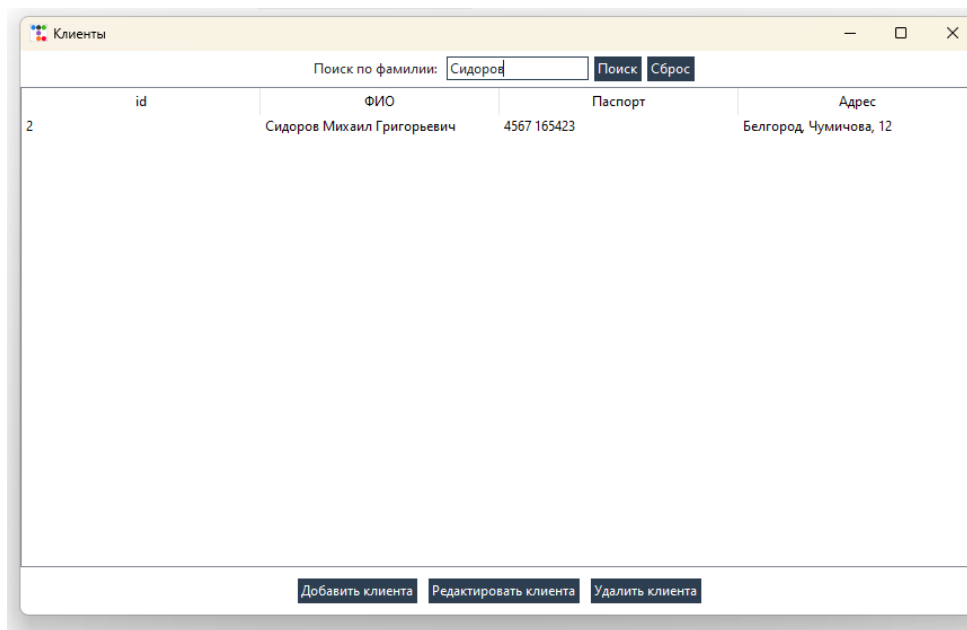
Сохранить изменения



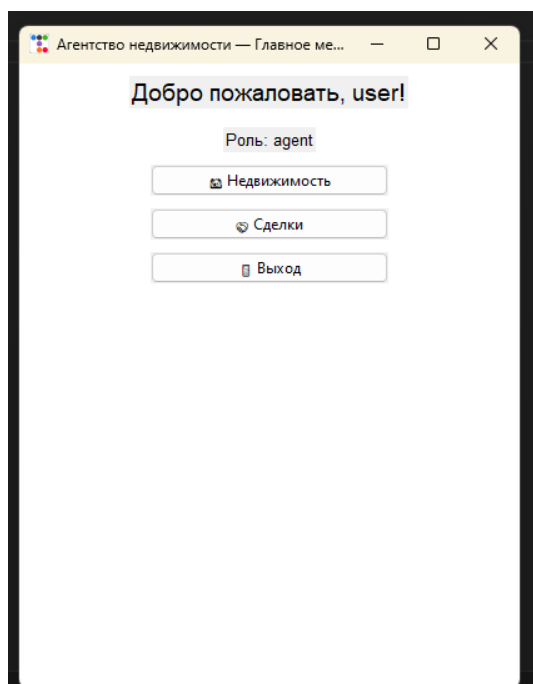
Клиенты				
Поиск по фамилии: <input type="text"/> <span>Поиск Сброс</span>				
id	ФИО	Паспорт	Адрес	
1	Петров Сергей Иванович	1234 123456	Белгород, Попова, 34	
2	Сидоров Михаил Григорьевич	4567 165423	Белгород, Чумичова, 12	
3	Завойчинская Анна Олеговна	1418 763476	Белгород, Корочанская, 5	
4	Петров Иван Иванович	1415 162978	Белгород, Чапаева, 78	
5	Власова Виктория Александровна	1418 164678	Белгород, Попова, 72	

Добавить клиента Редактировать клиента Удалить клиента





У клиента есть доступ к просмотру разрешенных таблиц, редактировать таблицы такой пользователь не может:



## Настройка резервного копирования (администратор):

Админ-панель туристического агентства

— □ ×

[Главное](#) [Управление](#) [Помощь](#)

### Авто резервное копирование

Выбрать папку для резервных копий

Интервал резервного копирования:  
1 часы

Начать резервное копирование

Остановить резервное копирование

## **Заключение**

В результате выполнения курсовой работы было создано работоспособное приложение для управления данными агентства недвижимости. Были применены полученные знания и навыки в области баз данных и объектно-ориентированного программирования на Python. Данный проект представляет собой основу для дальнейшего развития и совершенствования. Работа над проектом позволила получить ценный опыт в разработке программного обеспечения для управления базами данных.

## 4 Список литературы

1. Микросервисная архитектура  
(URL: [https://ru.wikipedia.org/wiki/Микросервисная\\_архитектура](https://ru.wikipedia.org/wiki/Микросервисная_архитектура), дата обращения: 29.03.2025)
2. Что такое Docker и как он работает  
(URL: [https://skillbox.ru/media/code/kak-rabotaet-docker-podrobnyy-gayd- ot-tekhlica/](https://skillbox.ru/media/code/kak-rabotaet-docker-podrobnyy-gayd-ot-tekhlica/), дата обращения: 29.03.2025)
3. Официальная документация Rabbitmq  
(URL: <https://www.rabbitmq.com/docs>, дата обращения: 29.03.2025)
4. REST API: что это такое и как работает  
(URL: <https://skillbox.ru/media/code/rest-api-cto-eto-takoe-i-kak-rabotaet/>, дата обращения: 29.03.2025)
5. Официальная документация Fast Api  
(URL: <https://fastapi.tiangolo.com/tutorial/first-steps/>, дата обращения: 29.03.2025)
6. Официальная документация SQLAlchemy  
(URL: <https://www.sqlalchemy.org/>, дата обращения: 29.03.2025)
7. Подробно про JWT/Хабр  
(URL: <https://habr.com/ru/articles/842056/>, дата обращения: 29.03.2025)
8. Redis - что это и для чего нужен? Пример использования  
(URL: [https://skillbox.ru/media/code/znakomimsya\\_s\\_redis/](https://skillbox.ru/media/code/znakomimsya_s_redis/), дата обращения: 29.03.2025)
9. Официальная документация SQLAlchemy  
(URL: <https://starlette-login.readthedocs.io/en/stable/tutorial/sqladmin/>, дата обращения: 29.03.2025)
10. Официальная документация React TS

(URL: <https://react.dev/learn/typescript>, дата обращения: 29.03.2025)



## **5. Приложение**

Полный код - <https://github.com/gremaree/realestateagency>