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: Туристический оператор (реализация туров, договора, клиенты)

Автор работы _____ Хафизов Георгий Игоревич ПВ-222
(подпись)

Руководитель проекта _____ Панченко Максим Владимирович
(подпись)

Оценка _____

Белгород 2024

Содержание

Введение

В современном мире, где темп жизни постоянно ускоряется, а границы между странами стираются, туризм становится не только одной из наиболее динамично развивающихся отраслей экономики, но и важной составляющей социальной жизни человека. Туристические компании, как связующее звено между спросом и предложением в сфере отдыха, сталкиваются с необходимостью оперативного управления все более сложными бизнес-процессами. Постоянно растущий спрос на туристические услуги и постоянно расширяющееся разнообразие предложений в рамках пакетных туров, экскурсионных программ и индивидуальных маршрутов формируют конкурентную среду, в которой ключевую роль играют не только конкурентоспособная ценовая политика, но также эффективность управления всеми процессами и качественное, ориентированное на клиента обслуживание.

Для успешного функционирования туристического агентства, оптимизации его внутренних бизнес-процессов и обеспечения удовлетворенности клиентов необходимо использовать современные инструменты управления. Это касается таких аспектов как управления базой данных, где собраны все сведения о турах, клиентах, и договорах, для быстрого и оперативного доступа, формирования индивидуальных предложений и отчетов. Существенную роль также играет организация взаимоотношений с контрагентами и поставщиками, формирование графиков, автоматизация счетов и документооборота. Использование традиционных или устаревших методов учета данных и неэффективное управление процессами не позволяют достигнуть необходимой скорости реакции на потребности рынка и гарантировать высокую точность, увеличивают трудозатраты, повышают риск ошибок, а также задержек в обслуживании, и не гарантируют конкурентоспособность предприятия.

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 туристического оператора: реализацией туров, отслеживанием и учётом взаимоотношений с клиентами, управлением контрактами, расчётом стоимости, 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 туристического оператора, способного автоматизировать процессы управления данными, предоставить пользователю понятный интерфейс и повысить общую эффективность работы агентства. Для достижения поставленной цели необходимо:

1. Разработать структуру базы данных.
2. Реализовать пользовательский интерфейс, позволяющий взаимодействовать с базой данных.
3. Обеспечить возможность создания, чтения, обновления и удаления данных для всех сущностей в зависимости от прав пользователя.
4. Внедрить систему ролей, ограничивающую доступ пользователей к функционалу приложения.
5. Разработать функционал выгрузки данных по отдельным таблицам и всего объема данных базы данных в форматы JSON и xlsx для дальнейшего анализа.

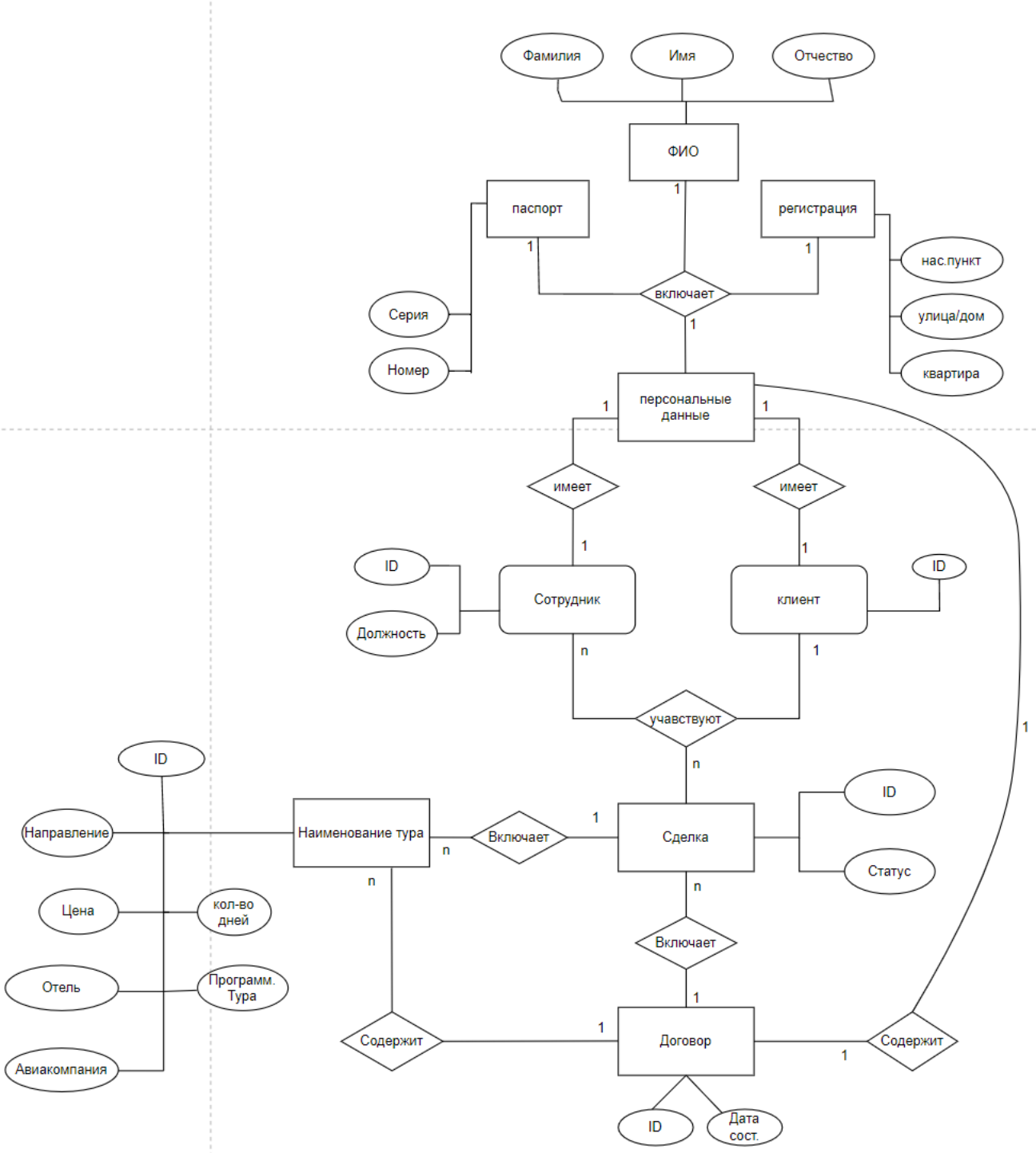
2.2 Проектирование базы данных

2.2.1 Анализ предметной области

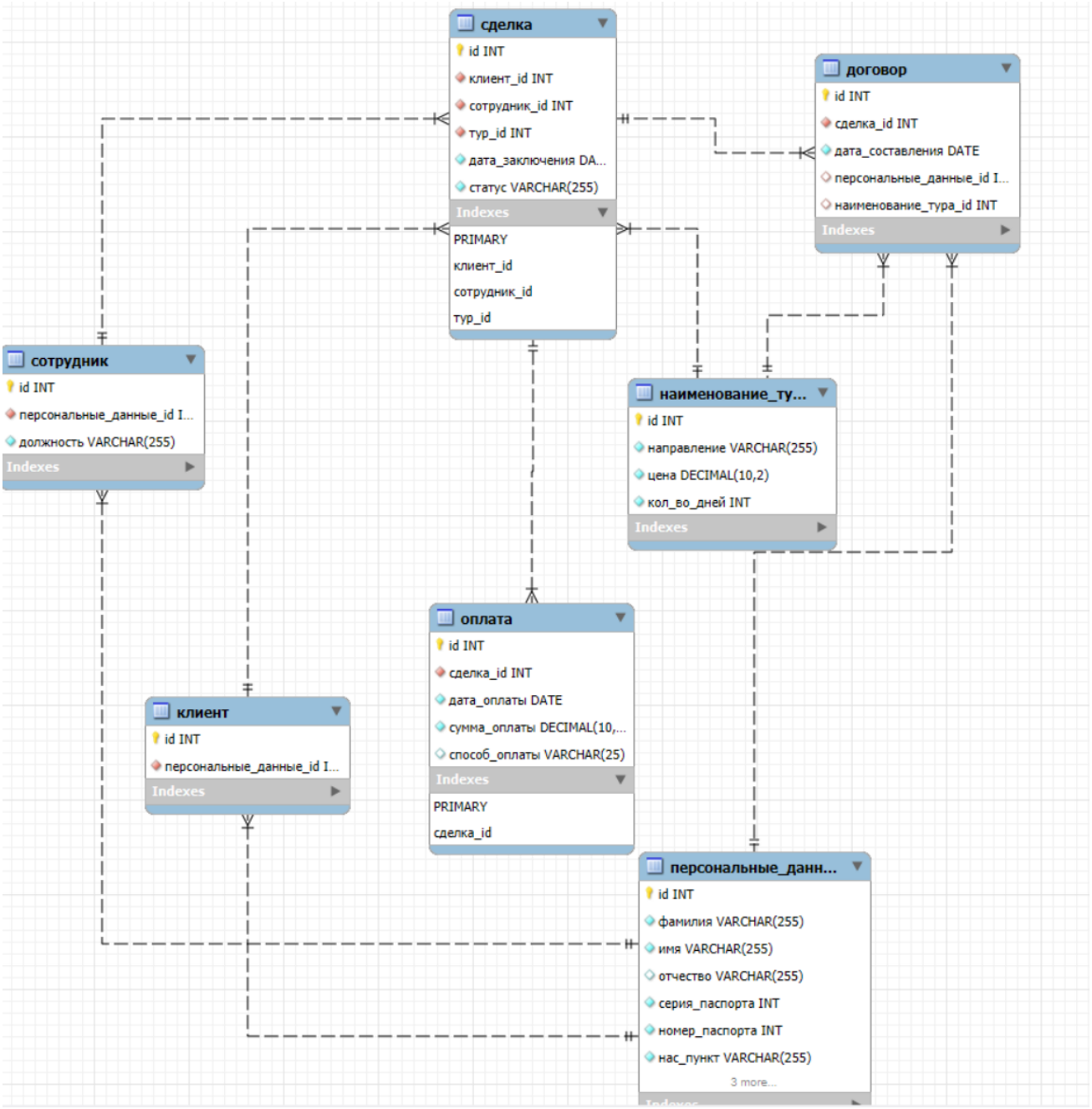
Сущность	Описание
Персональные данные	Личные данные, идентифицирующие клиента или сотрудника
Сотрудник	Работник туристического агентства
Клиент	Лицо, получающее туристические услуги
Сделка	Процесс оформления услуг, связывающий клиента, сотрудника и наименование тура
Договор	Условия сделки по туризму, связывающие сделку, наименование тура, а так же персональные данные
Наименование тура	Туристический продукт с параметрами, определяющими его стоимость и направление

У клиента и сотрудника есть **персональные данные**: ФИО, паспортные данные и адрес регистрации. **Тур** представляет собой туристическое направление с набором услуг, таких как отель, авиаперелет и программа тура. Тур может быть связан с несколькими договорами, поскольку он может продаваться разным клиентам. **Сотрудник** от лица туристического оператора заключает с клиентом **сделку**, связанную с выбранным **туром**. Клиент может участвовать в нескольких сделках, бронируя разные туры или повторно заказывая один и тот же тур. **Договор** фиксирует условия сделки. Договор связывает сделку с конкретным туром, описывая стоимость, продолжительность и направление тура, а так же **персональные данные** сотрудника и клиента. Сделка может включать одну или несколько оплат, которые фиксируют процесс оплаты услуг. Оплата может быть произведена разными способами.

2.2.2 Диаграмма «Сущность - Связь»



2.2.3 Схема структуры базы данных



2.2.4 Нормализация базы данных

База данных соответствует первой нормальной форме, так как мы преобразуем атрибут “Улица/Дом” в 2 разных “Улица” “Дом” атрибуты всех сущностей являются атомарными (неделимыми). Каждое значение хранится в отдельной ячейке таблицы, и повторяющиеся группы отсутствуют.

База данных находится во второй нормальной форме, так как она соответствует первой нормальной форме, и каждый неключевой атрибут функционально зависит от всего первичного ключа. В сущностях с составными ключами неключевые атрибуты зависят от всех частей составного ключа.

База данных находится в третьей нормальной форме, так как она соответствует второй нормальной форме, и каждый неключевой атрибут зависит от первичного ключа только напрямую, без транзитивных зависимостей.

2.2.5 Описание процесса создания базы данных

Создали новую БД на локальном сервере. Путем написания запроса

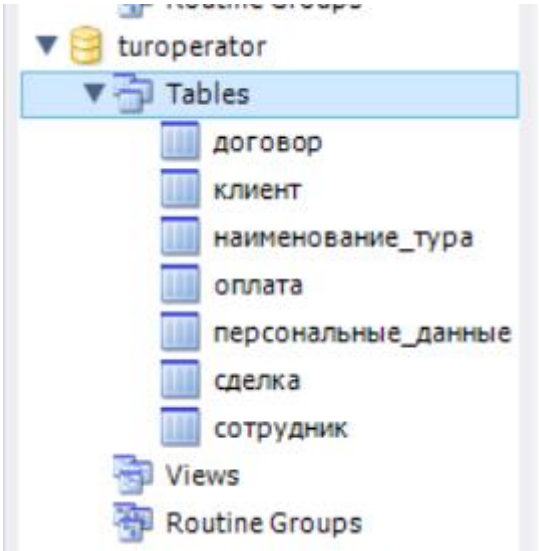
```
CREATE DATABASE IF NOT EXISTS turoperator;
```

Далее каждую сущность создавали так же – SQL запрос

```
CREATE TABLE IF NOT EXISTS `персональные_данные` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `фамилия` VARCHAR(255) NOT NULL,
    `имя` VARCHAR(255) NOT NULL,
    `отчество` VARCHAR(255),
    `серия_паспорта` INT NOT NULL,
    `номер_паспорта` INT NOT NULL,
    `нас_пункт` VARCHAR(255) NOT NULL,
    `улица_дом` VARCHAR(255) NOT NULL,
    `квартира` VARCHAR(255)
);

-- Создание таблицы "Сотрудник"
CREATE TABLE IF NOT EXISTS `сотрудник` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `персональные_данные_id` INT NOT NULL,
    `должность` VARCHAR(255) NOT NULL,
    FOREIGN KEY (`персональные_данные_id`) REFERENCES `персональные_данные`(`id`)
);

-- Создание таблицы "Клиент"
CREATE TABLE IF NOT EXISTS `клиент` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `персональные_данные_id` INT NOT NULL,
    FOREIGN KEY (`персональные_данные_id`) REFERENCES `персональные_данные`(`id`)
);
```

```
-- Создание таблицы "Наименование тура"
CREATE TABLE IF NOT EXISTS `наименование_тура` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `направление` VARCHAR(255) NOT NULL,
    `цена` DECIMAL(10, 2) NOT NULL,
    `кол_во_дней` INT NOT NULL
);

-- Создание таблицы "Сделка"
CREATE TABLE IF NOT EXISTS `сделка` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `клиент_id` INT NOT NULL,
    `сотрудник_id` INT NOT NULL,
    `тур_id` INT NOT NULL,
    `дата_заключения` DATE NOT NULL,
    `статус` VARCHAR(255) NOT NULL,
    FOREIGN KEY (`клиент_id`) REFERENCES `клиент`(`id`),
    FOREIGN KEY (`сотрудник_id`) REFERENCES `сотрудник`(`id`),
    FOREIGN KEY (`тур_id`) REFERENCES `наименование_тура`(`id`)
);

-- Создание таблицы "Договор"
CREATE TABLE IF NOT EXISTS `договор` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `сделка_id` INT NOT NULL,
    `дата_составления` DATE NOT NULL,
    FOREIGN KEY (`сделка_id`) REFERENCES `сделка`(`id`)
);

-- Создание таблицы "Оплата"
CREATE TABLE IF NOT EXISTS `оплата` (
    `id` INT PRIMARY KEY AUTO_INCREMENT,
    `сделка_id` INT NOT NULL,
    `дата_оплаты` DATE NOT NULL,
    `сумма_оплаты` DECIMAL(10, 2) NOT NULL,
    `способ_оплаты` VARCHAR(25),
     FOREIGN KEY (`сделка_id`) REFERENCES `сделка`(`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.

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

class Персональные_данные(BaseModel):
    фамилия = CharField()
    имя = CharField()
    отчество = CharField(null=True)
    серия_паспорта = IntegerField()
    номер_паспорта = IntegerField()
    нас_пункт = CharField()
    улица = CharField()
    дом = CharField()
    квартира = CharField(null=True)

class Сотрудник(BaseModel):
    персональные_данные = ForeignKeyField(Персональные_данные,
backref='сотрудники')
    должность = CharField()

class Клиент(BaseModel):
    персональные_данные = ForeignKeyField(Персональные_данные, backref='клиенты')
```

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

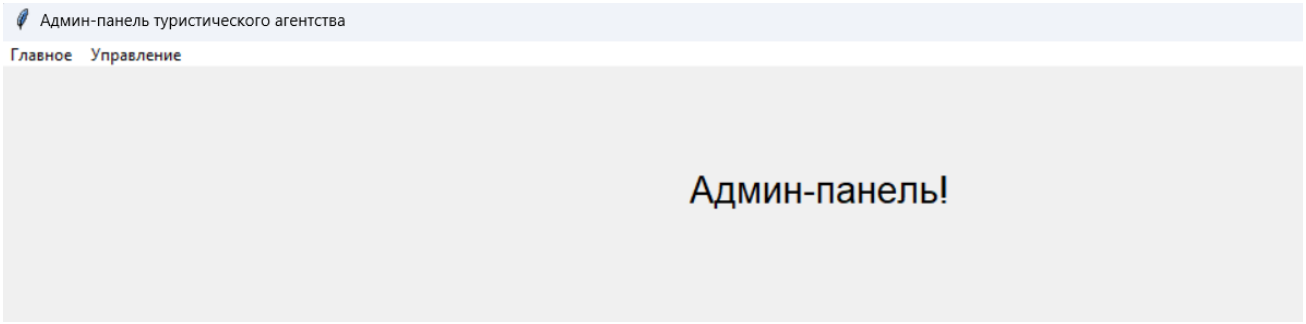
Пользовательский интерфейс построен с использованием библиотеки Tkinter и стилевого движка ttk, что позволяет создавать кросс-платформенный, удобный и интуитивно понятный интерфейс с возможностью тонкой настройки его внешнего вида.

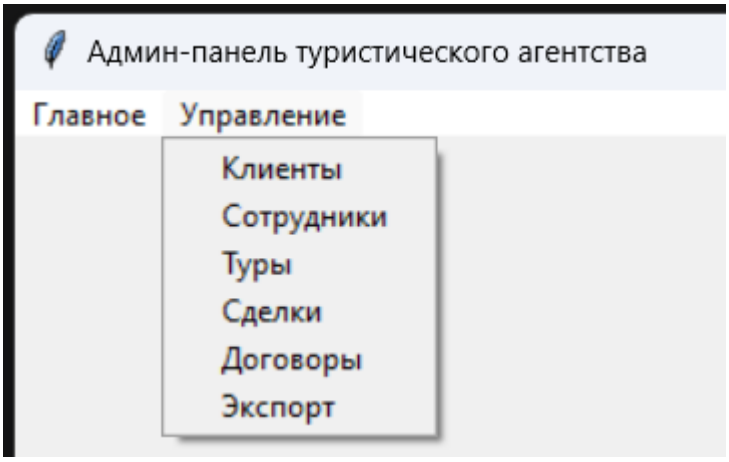
```
class AdminApp:
    def __init__(self, root, db, position):
        self.root = root
        self.root.title("Админ-панель туристического агентства")
        self.root.geometry("1200x900")

        self.style = ttk.Style(root)
        self.configure_styles()

        self.db = db
        self.position = position
        self.create_menu()
        self.create_welcome_frame()
```

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. Библиотека reportlab применяется для автоматической генерации PDF-документов на основе данных из БД.

```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from models import *
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import letter
from reportlab.lib.units import inch


class ContractsFrame(ttk.Frame):
    def __init__(self, parent, db, position):
        super().__init__(parent, padding=20)
        self.db = db
        self.position = position
        self.create_ui()

    def create_ui(self):
        ttk.Label(self, text="Управление договорами", font=("Arial",
16)).pack(pady=10)

        # Treeview для отображения договоров
        columns = ("id", "Клиент", "Тур", "Дата составления", "Сумма")
        self.tree = ttk.Treeview(self, columns=columns, show="headings")

        for col in columns:
            self.tree.heading(col, text=col)
        self.tree.pack(fill=tk.BOTH, expand=True)

        # Кнопка для скачивания PDF
        if self.position == "Директор" or self.position == "Менеджер":
            ttk.Button(self, text="Скачать PDF",
command=self.generate_pdf).pack(pady=10)

        self.update_contracts_list()

    def update_contracts_list(self):
        for row in self.tree.get_children():
            self.tree.delete(row)

        contracts = Договор.select()
        for contract in contracts:
            client = contract.персональные_данные
            tour = contract.наименование_тура
            client_name = f"{client.фамилия} {client.имя}"
            tour_name = f"{tour.направление}"

            deal = contract.сделка
            total_amount = 0
            if deal:
                payments = deal.оплаты
                for payment in payments:
                    total_amount += payment.сумма_оплаты

            self.tree.insert("", tk.END, values=(
```

```
            contract.id, client_name, tour_name, contract.дата_составления,
f"{total_amount:.2f}"))

    def generate_pdf(self):
        selected_item = self.tree.selection()
        if not selected_item:
            messagebox.showinfo("Ошибка", "Выберите договор для скачивания PDF")
            return
        contract_id = self.tree.item(selected_item, "values")[0]

        try:
            contract = Договор.get(Договор.id == contract_id)
            self.create_contract_pdf(contract)
        except Exception as e:
            messagebox.showinfo("Ошибка", f"Ошибка при генерации PDF: {e}")

    def create_contract_pdf(self, contract):

        filename = filedialog.asksaveasfilename(defaultextension=".pdf",
filetypes=[("PDF files", "*.pdf")])
        if not filename:
            return

        c = canvas.Canvas(filename, pagesize=letter)

        # Заголовок
        c.setFont("Helvetica-Bold", 16)
        c.drawCentredString(4.135 * inch, 10 * inch, "CONTRACT")
        c.line(0.5 * inch, 9.8 * inch, 7.7 * inch, 9.8 * inch)

        # Данные о договоре
        c.setFont("Helvetica", 12)
        c.drawString(0.75 * inch, 9.3 * inch, f"Number contract: {contract.id}")
        c.drawString(0.75 * inch, 9 * inch, f"Date: {contract.дата_составления}")

        c.drawString(0.75 * inch, 8.5 * inch, "Information about client:")
        client = contract.персональные_данные
        c.drawString(0.75 * inch, 8.2 * inch, f"FIO: {client.фамилия}
{client.имя} {client.отчество or ''}")
        c.drawString(0.75 * inch, 7.9 * inch, f"Pasport: {client.серия_паспорта}
{client.номер_паспорта}")
        c.drawString(0.75 * inch, 7.6 * inch,
                     f"Адрес: {client.нас_пункт}, {client.улица}, {client.дом}
{client.квартира or ''}")

        c.drawString(0.75 * inch, 7.1 * inch, "Information about toure:")
        tour = contract.наименование_тура
        c.drawString(0.75 * inch, 6.8 * inch, f"Where: {tour.направление}")
        c.drawString(0.75 * inch, 6.5 * inch, f"Price: {tour.цена}")
        c.drawString(0.75 * inch, 6.2 * inch, f"Days: {tour.кол_во_дней}")

        deal = contract.сделка
        total_amount = 0
        if deal:
            payments = deal.оплаты
            for payment in payments:
                total_amount += payment.сумма_оплаты
        c.drawString(0.75 * inch, 5.9 * inch, f"Total sum: {total_amount:.2f}")

        c.save()
        messagebox.showinfo("Успех", "PDF успешно сгенерирован!")
```





Приложении реализована работа с различными ролями пользователей: Агент, Менеджер, Директор, с разграничением доступа к функционалу, для более эффективного распределения ответственности.

```
AGENT = "Агент"
MANAGER = "Менеджер"
DIRECTOR = "Директор"
```

```
if self.position == MANAGER or self.position == DIRECTOR:
    manage_menu.add_command(label="Туры", command=self.show_tours_frame)

manage_menu.add_command(label="Сделки", command=self.show_deals_frame)

if self.position == DIRECTOR or self.position == MANAGER:
    manage_menu.add_command(label="Договоры", command=self.show_contracts_frame)

if self.position == DIRECTOR:
    manage_menu.add_command(label="Экспорт", command=self.show_export_frame)
```

```
ttk.Button(button_frame, text="Добавить",
command=self.add_client).pack(side=tk.LEFT, padx=5)

if self.position == "Директор" or self.position == "Менеджер" or self.position ==
"Агент":
    ttk.Button(button_frame, text="Обновить",
command=self.update_client).pack(side=tk.LEFT, padx=5)

if self.position == "Директор" or self.position == "Менеджер":
```

```
    ttk.Button(button_frame, text="Удалить",
command=self.delete_client).pack(side=tk.LEFT, padx=5)
```

2.5 Разработка автоматического резервного копирования базы данных

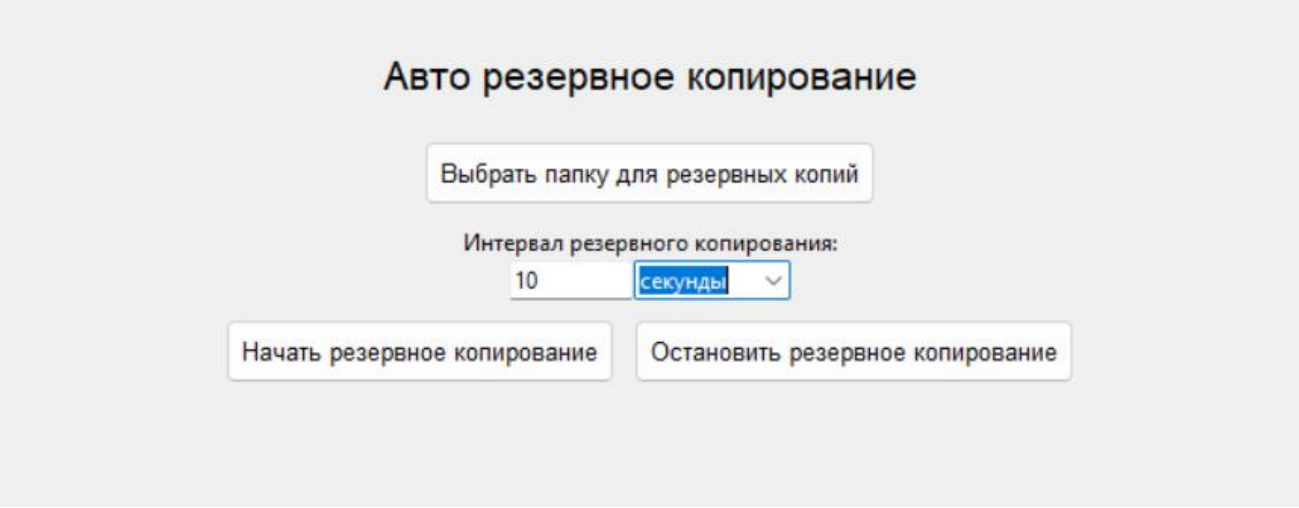
Для реализации используем

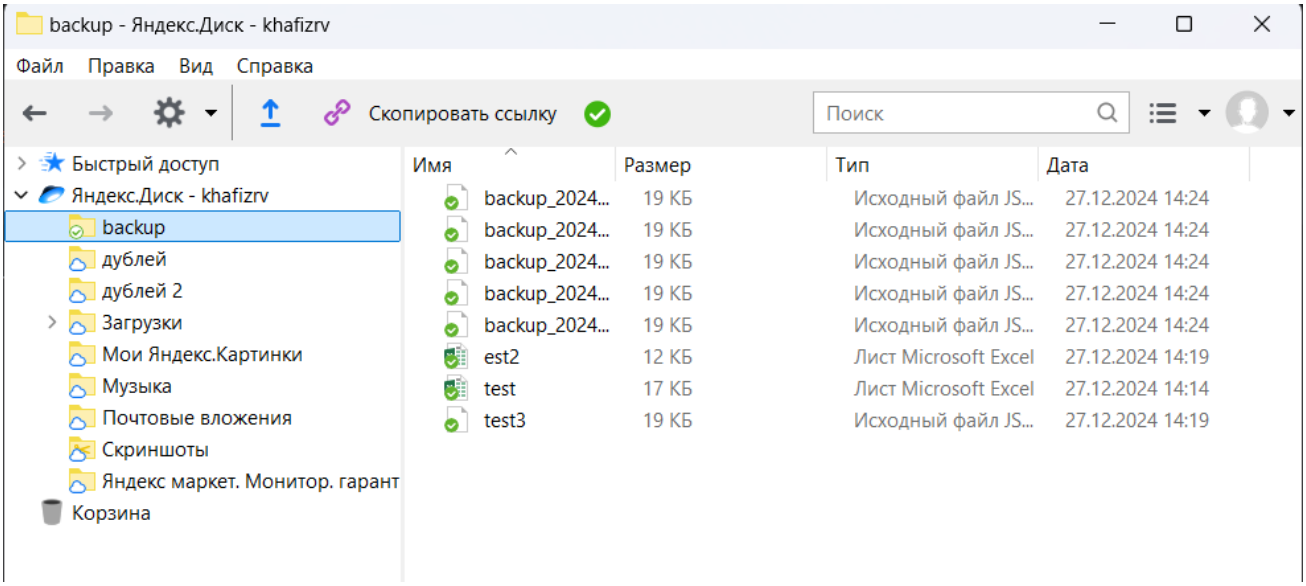
```
import tkinter as tk
from tkinter import ttk, messagebox, filedialog
from models import *
import json
from decimal import Decimal
import datetime
import threading
import time
import os


class BackupFrame(ttk.Frame):
    def __init__(self, parent, db, position):
        super().__init__(parent, padding=20)
        self.db = db
        self.position = position
        self.backup_folder = None
        self.interval = 3600
        self.interval_unit = "секунды"
        self.create_ui()
        self.backup_timer = None

    def create_ui(self):
        ttk.Label(self, text="Авто резервное копирование", font=("Arial",
16)).pack(pady=10)

        if self.position == "Директор":
            # Папка для резервной копии
            ttk.Button(self, text="Выбрать папку для резервных копий",
command=self.choose_backup_folder).pack(pady=10)

            # Выбор периода резервного копирования
            ttk.Label(self, text="Интервал резервного копирования:").pack()

            interval_frame = ttk.Frame(self)
            interval_frame.pack()

            self.interval_entry = ttk.Entry(interval_frame, width=10)
            self.interval_entry.insert(0, 1)
            self.interval_entry.pack(side=tk.LEFT)

            units = ["секунды", "минуты", "часы", "дни"]
            self.interval_unit_var = tk.StringVar(self)
            self.interval_unit_var.set(units[2])
            unit_dropdown = ttk.Combobox(interval_frame,
textvariable=self.interval_unit_var, values=units, width=10)
            unit_dropdown.pack(side=tk.LEFT)

            # Кнопки управления резервными копиями
            button_frame = ttk.Frame(self)
            button_frame.pack(pady=10)
```

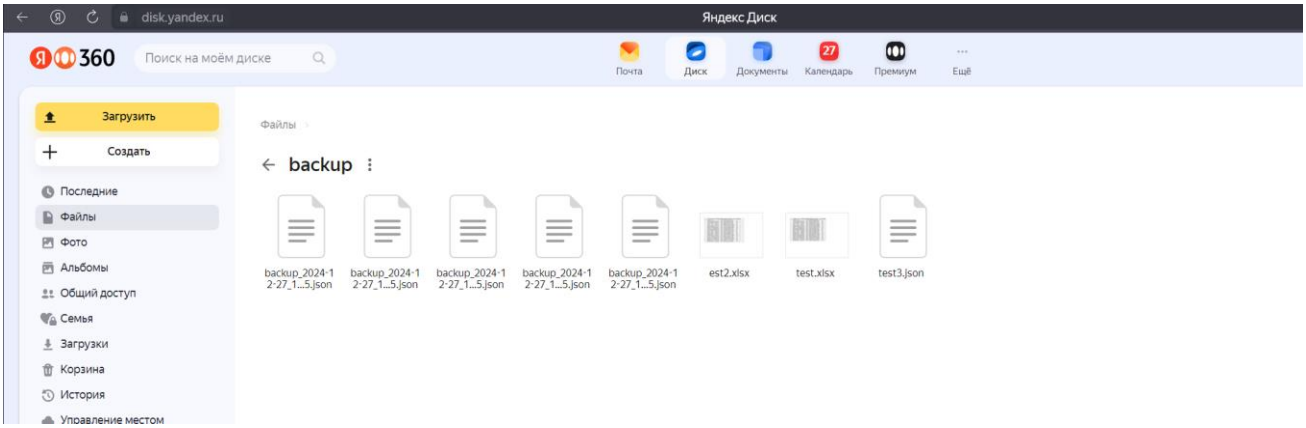
```
            ttk.Button(button_frame, text="Начать резервное копирование",
command=self.start_backup).pack(side=tk.LEFT,

padx=5)
            ttk.Button(button_frame, text="Остановить резервное копирование",
command=self.stop_backup).pack(
                side=tk.LEFT, padx=5)

    def choose_backup_folder(self):
        self.backup_folder = filedialog.askdirectory()
        if self.backup_folder:
            messagebox.showinfo("Успех", f"Папка для резервных копий выбрана:
{self.backup_folder}")

    def start_backup(self):
        if not self.backup_folder:
            messagebox.showinfo("Ошибка", "Выберите папку для резервных копий")
            return
        try:
            interval_value = int(self.interval_entry.get())
            unit = self.interval_unit_var.get()

            if unit == "секунды":
                self.interval = interval_value
            elif unit == "минуты":
                self.interval = interval_value * 60
            elif unit == "часы":
                self.interval = interval_value * 60 * 60
            elif unit == "дни":
                self.interval = interval_value * 60 * 60 * 24

        except Exception:
            messagebox.showinfo("Ошибка", "Введите корректный интервал")
            return

        self.perform_backup()
        if self.backup_timer:
            self.backup_timer.cancel()
        self.backup_timer = threading.Timer(self.interval, self.start_backup)
        self.backup_timer.start()
        messagebox.showinfo("Успех", f"Резервное копирование начато с интервалом
{interval_value} {unit}")

    def stop_backup(self):
        if self.backup_timer:
            self.backup_timer.cancel()
            messagebox.showinfo("Успех", "Резервное копирование остановлено")
        else:
            messagebox.showinfo("Ошибка", "Резервное копирование не запущено")

    def perform_backup(self):
        if not self.backup_folder:
            messagebox.showinfo("Ошибка", "Выберите папку для резервных копий")
            return

        try:
            all_data = self.get_all_data()
            now = datetime.datetime.now().strftime("%Y-%m-%d_%H-%M-%S")
            filename = os.path.join(self.backup_folder, f"backup_{now}.json")
            with open(filename, "w", encoding="utf-8") as outfile:
                json.dump(all_data, outfile, indent=4, ensure_ascii=False)
```

```
            print(f"Резервная копия успешно создана: {filename}")
        except Exception as e:
            messagebox.showinfo("Ошибка", f"Ошибка при резервном копировании
{e}")

    def get_all_data(self):
        tables = ['Персональные_данные', 'Сотрудник', 'Клиент',
'Наименование_тура', 'Сделка', 'Договор', 'Оплата']
        all_data = {}
        for table_name in tables:
            try:
                model = globals()[table_name]
                query = model.select()
                data = list(query.dicts())

                # Преобразуем все Decimal в float и date в строки
                for row in data:
                    for key, value in row.items():
                        if isinstance(value, Decimal):
                            row[key] = float(value)
                        if isinstance(value, datetime.date):
                            row[key] = value.strftime('%Y-%m-%d')

                all_data[table_name] = data
            except Exception as e:
                messagebox.showinfo("Ошибка", f"Ошибка при получении данных из
{table_name} таблицы {e}")
                return {}
        return all_data
```

Для примера выбрал интервал между бэкапами 10 сек

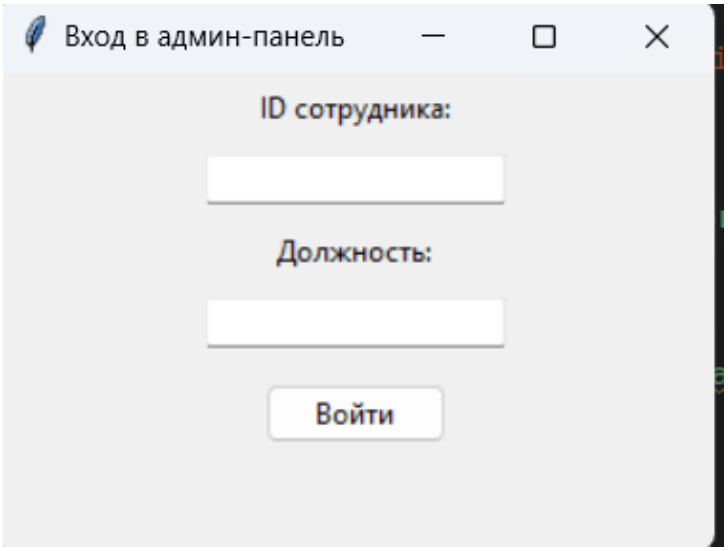


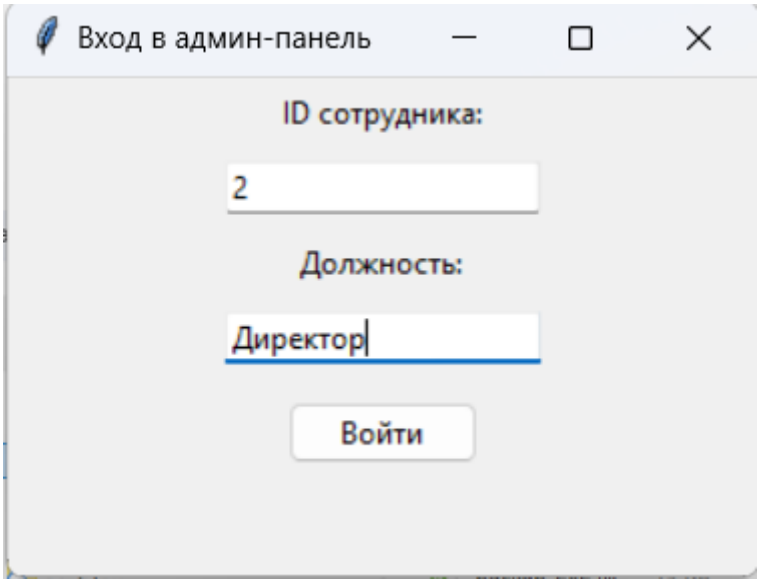


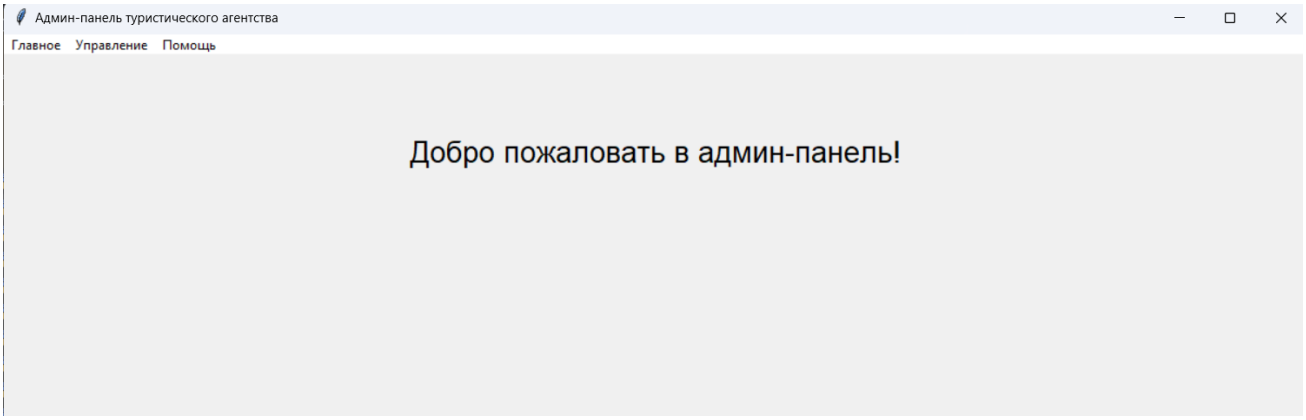


2.6 Результаты работы приложения

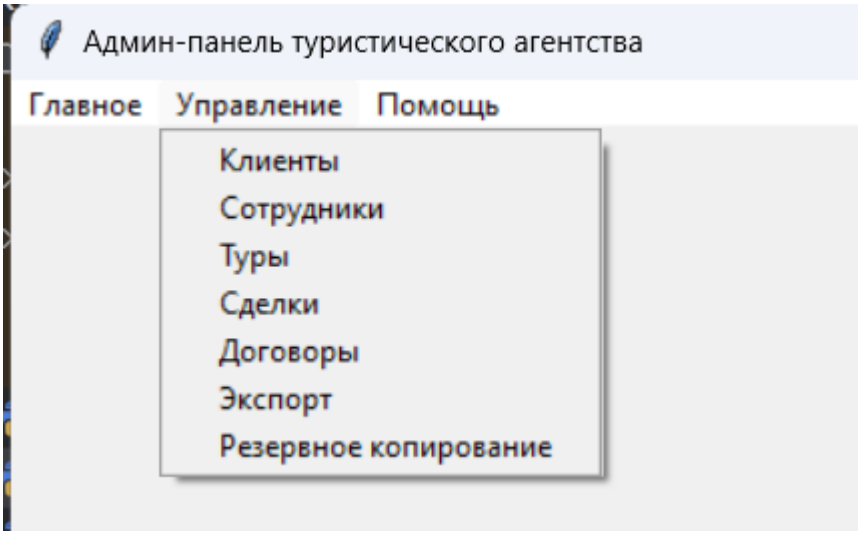
- При запуске надо пройти индентификацию. Каждая должность (Агент, Менеджер, Директор) обладает своими правами



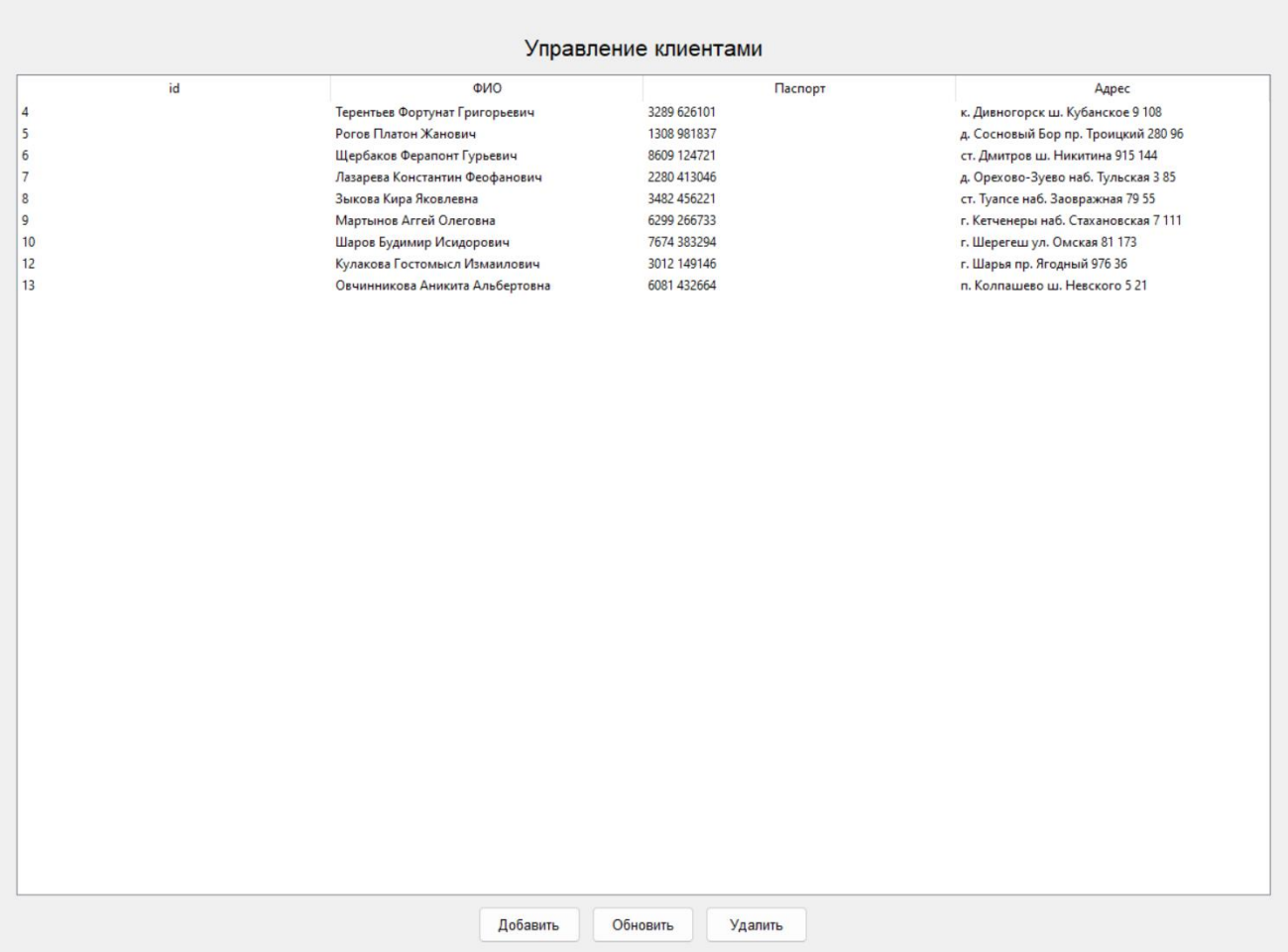


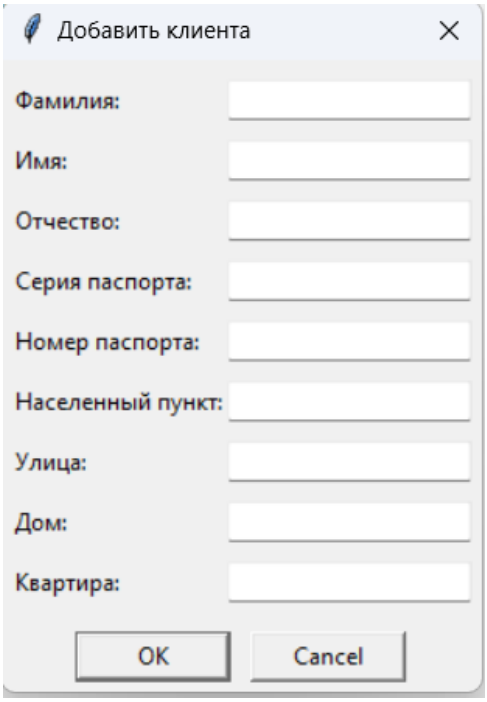


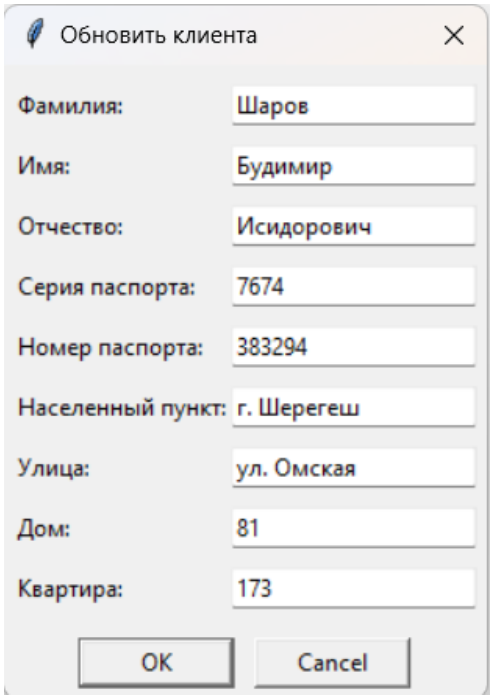
- Директор обладает всеми возможностями взаимодействия с БД

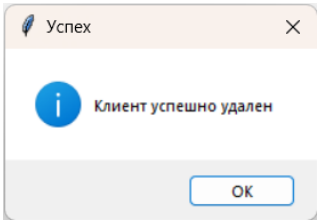


- Он может редактировать страницу с клиентами

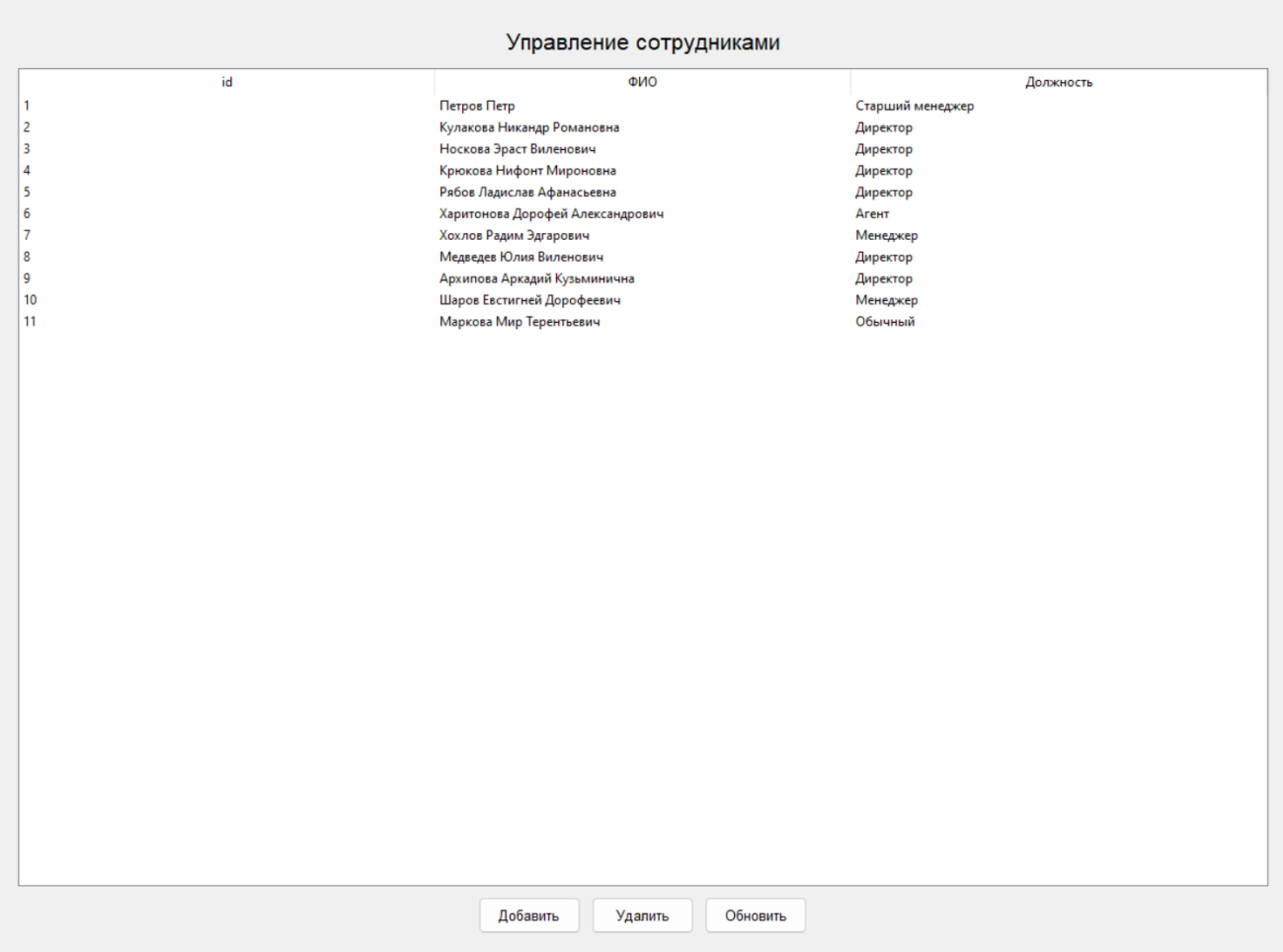


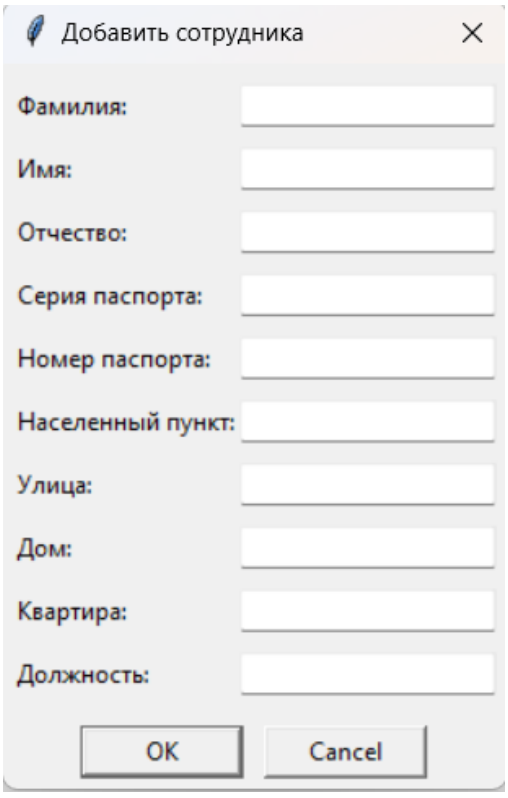


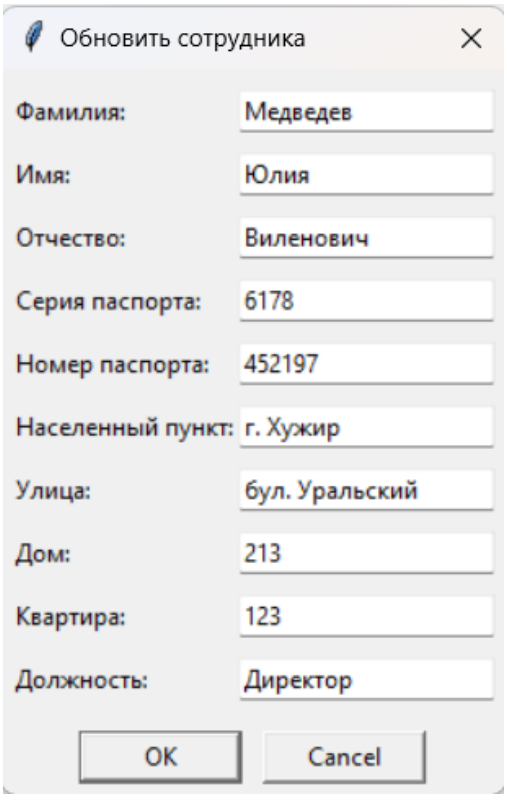


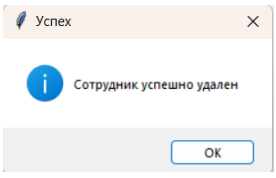


- Редактировать страницу с сотрудниками

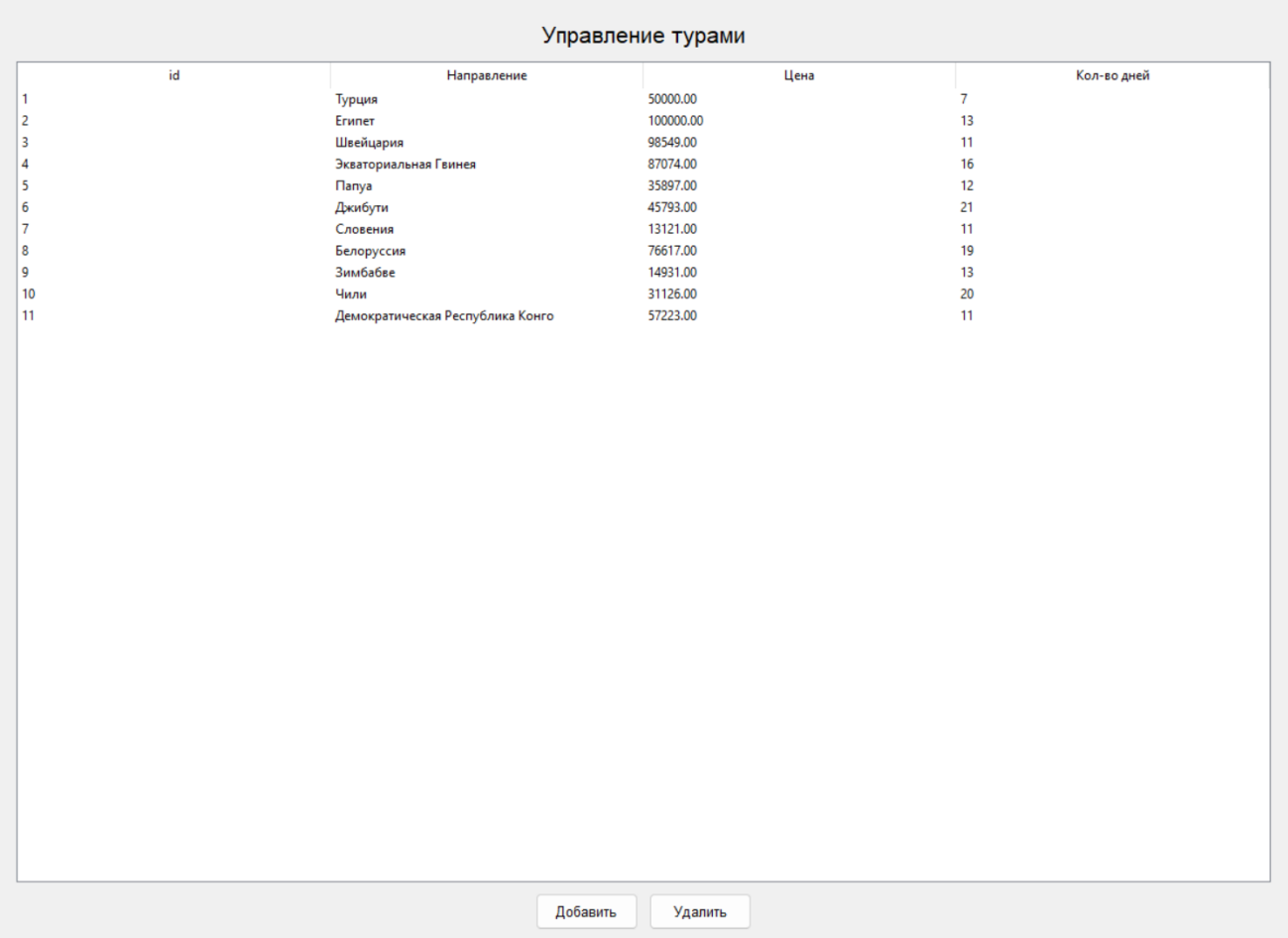


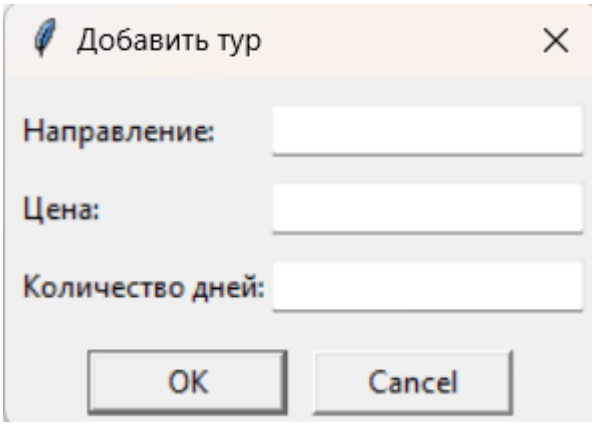


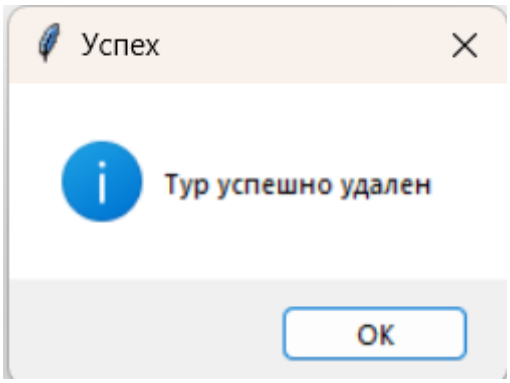




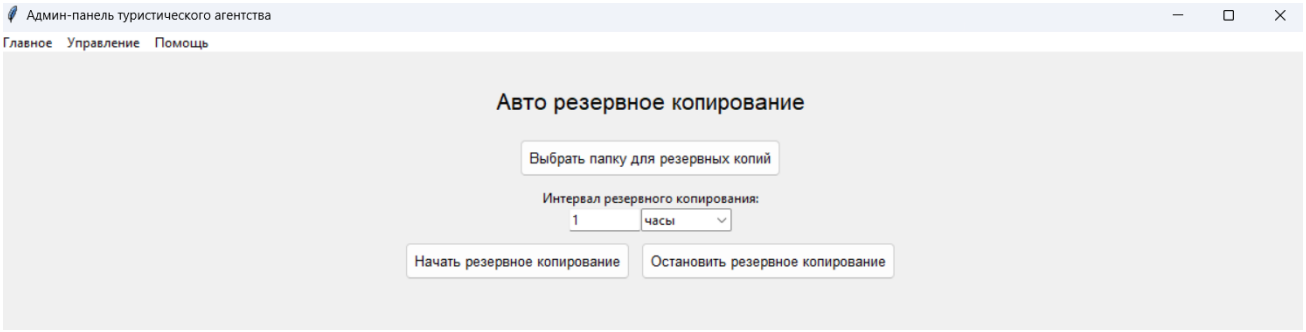
- Добовлять удолять туры



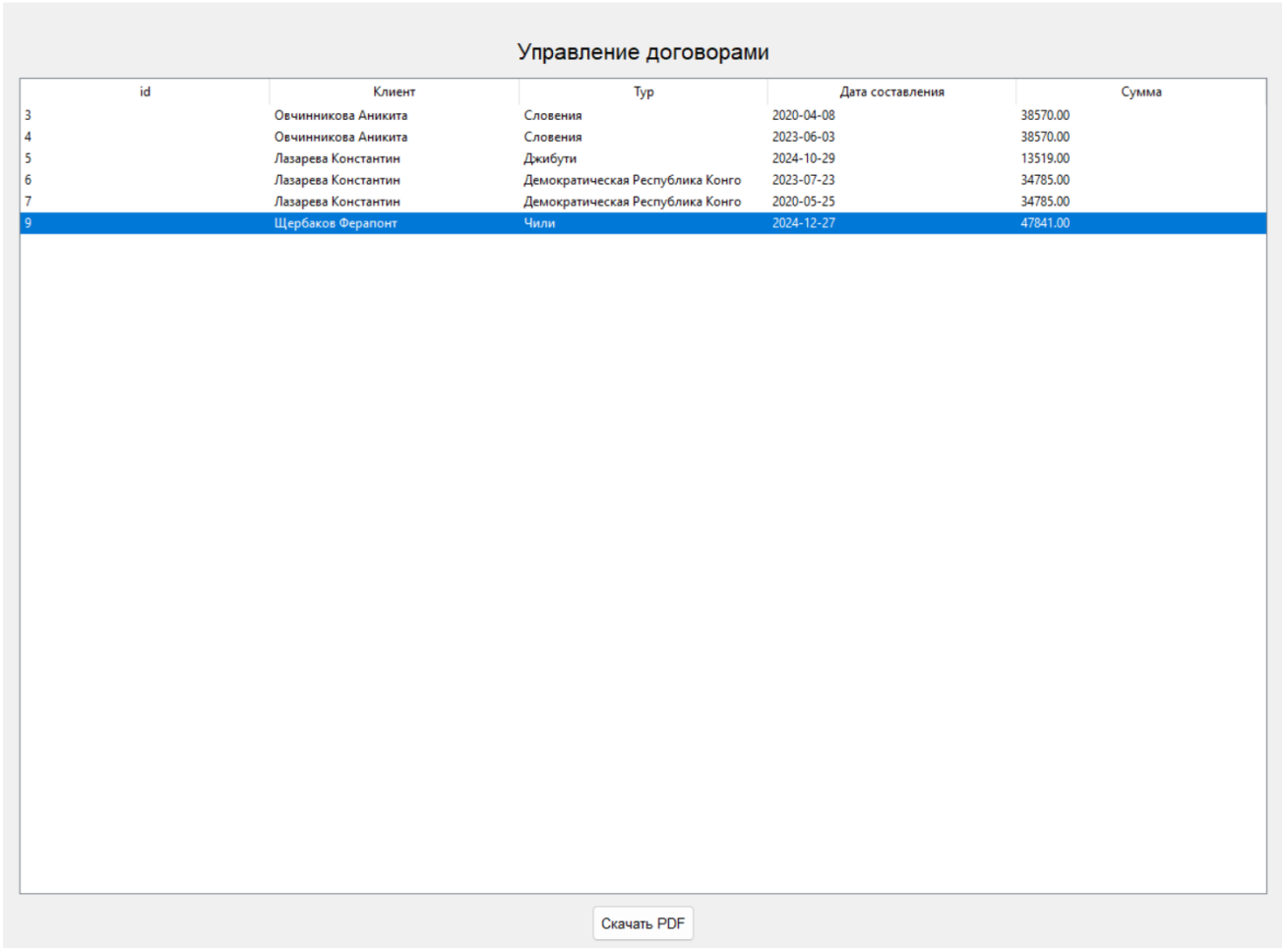


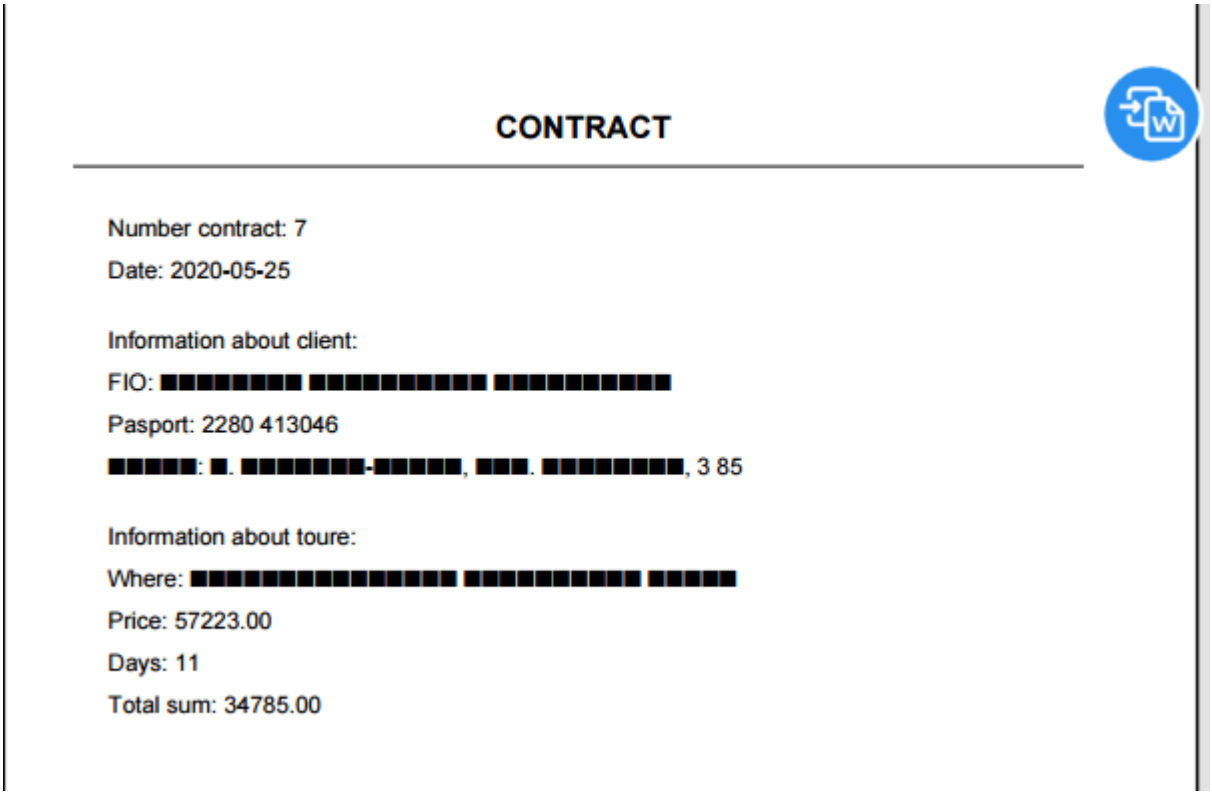


- Настраивать авто резервное копирование



- Формировать договоры для печати





CONTRACT

Number contract: 7

Date: 2020-05-25

Information about client:

FIO: [redacted]

Pasport: 2280 413046

[redacted]

Information about toure:

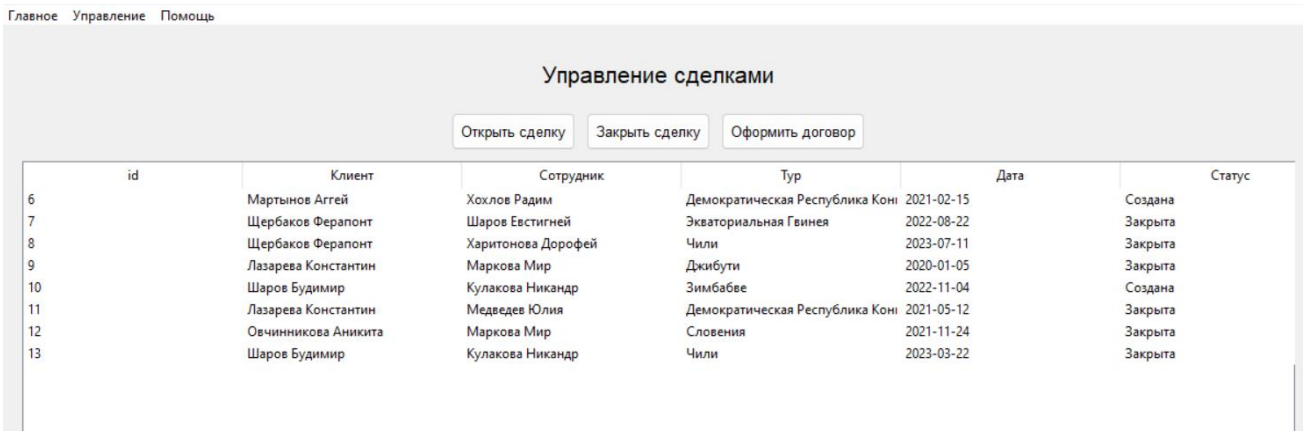
Where: [redacted]

Price: 57223.00

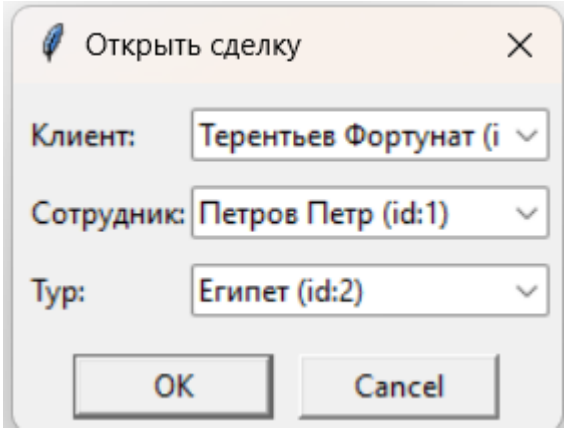
Days: 11

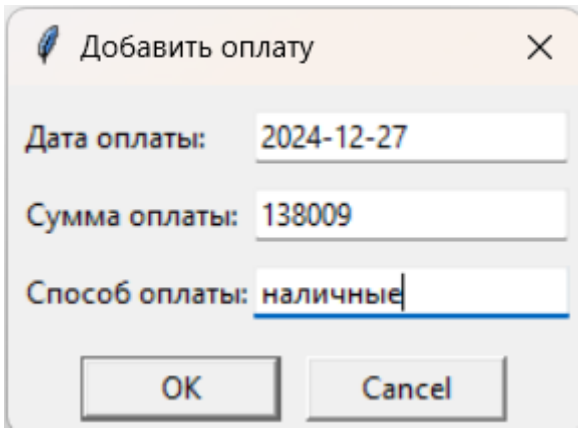
Total sum: 34785.00

- Управлять сделками

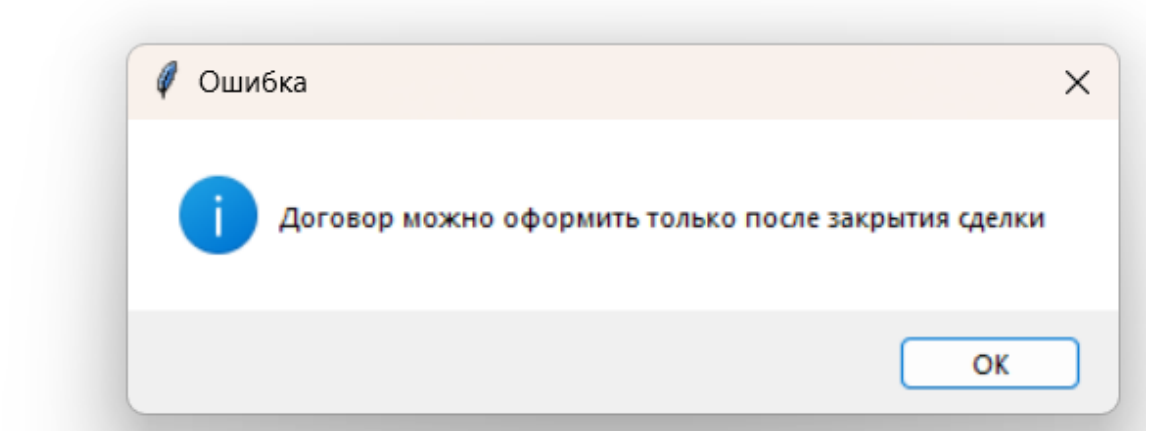


id	Клиент	Сотрудник	Тур	Дата	Статус
6	Мартынов Аггей	Хохлов Радим	Демократическая Республика Кони	2021-02-15	Создана
7	Щербаков Ферапонт	Шаров Евстигней	Экваториальная Гвинея	2022-08-22	Закрыта
8	Щербаков Ферапонт	Харитонова Дорофей	Чили	2023-07-11	Закрыта
9	Лазарева Константин	Маркова Мир	Джибути	2020-01-05	Закрыта
10	Шаров Будимир	Кулакова Никандр	Зимбабве	2022-11-04	Создана
11	Лазарева Константин	Медведев Юлия	Демократическая Республика Кони	2021-05-12	Закрыта
12	Овчинникова Аникита	Маркова Мир	Словения	2021-11-24	Закрыта
13	Шаров Будимир	Кулакова Никандр	Чили	2023-03-22	Закрыта

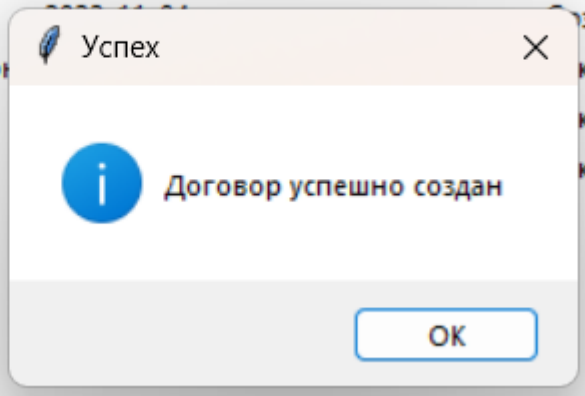




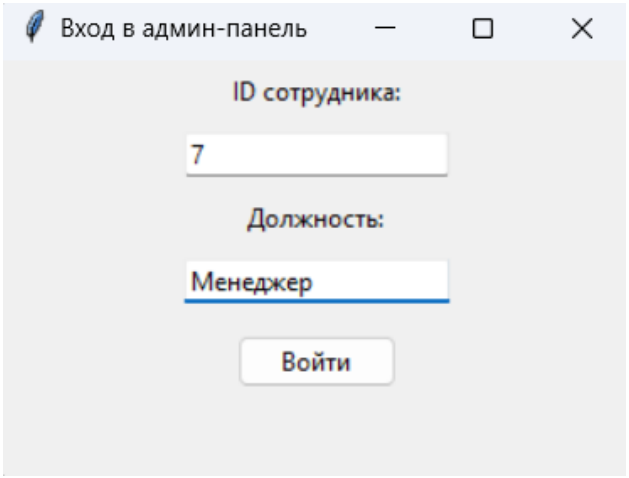
Джибути	2020-01-05	Закрыта
Зимбабве	2022-11-04	Создана
Демократическая Республика Кон	2021-05-12	Закрыта
Словения	2021-11-24	Закрыта
Чили	2023-03-22	Закрыта

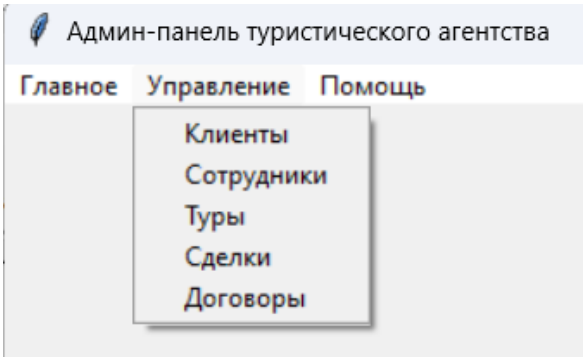


Чили	2023-07-11	Закрыта
Джибути	2020-01-05	Закрыта
Зимбабве		здана
Демократическая Республика Кон		крыта
Словения		крыта
Чили		крыта

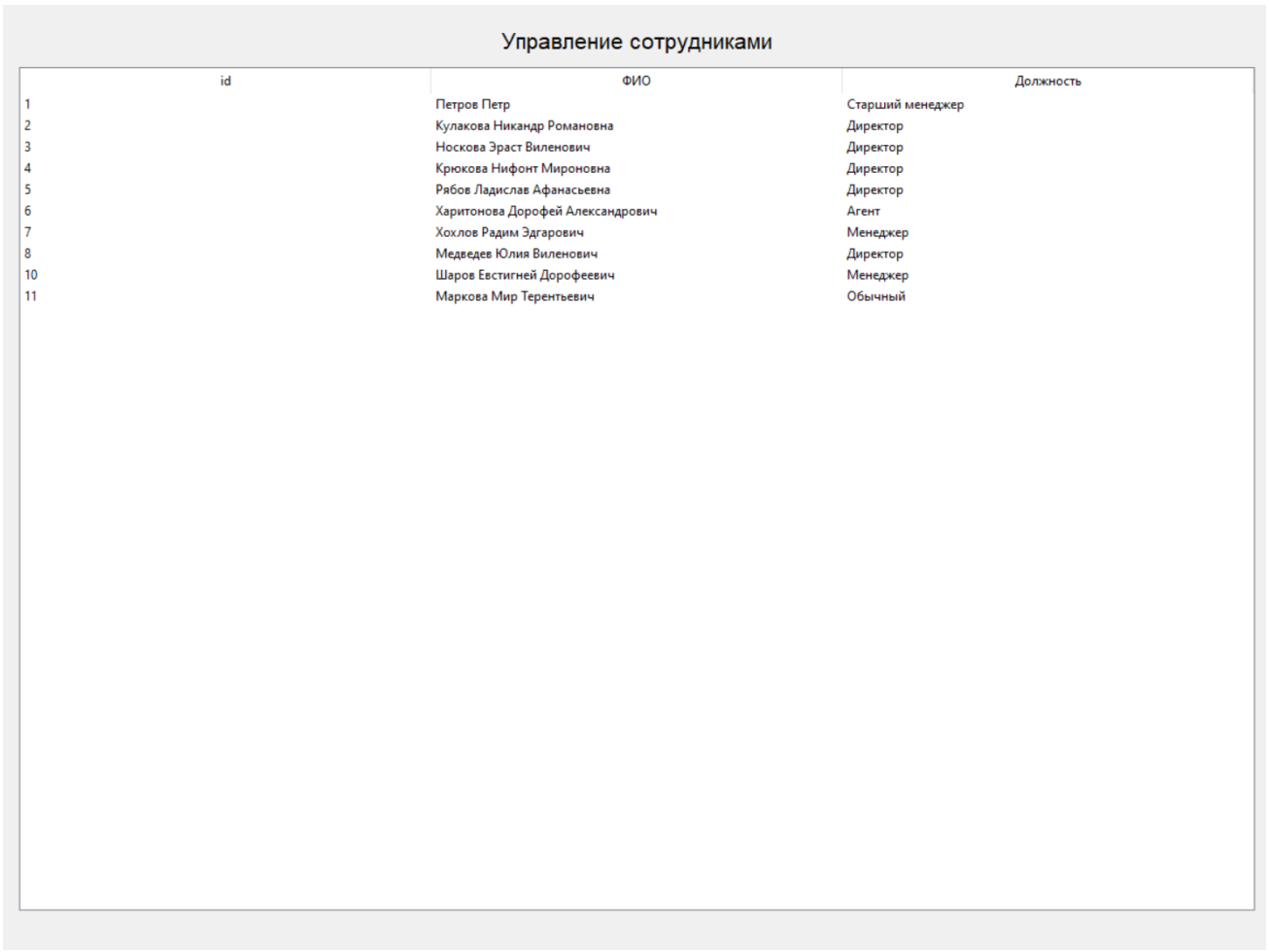


- Менеджер имеет меньше привелегий, не занимается backup-ами

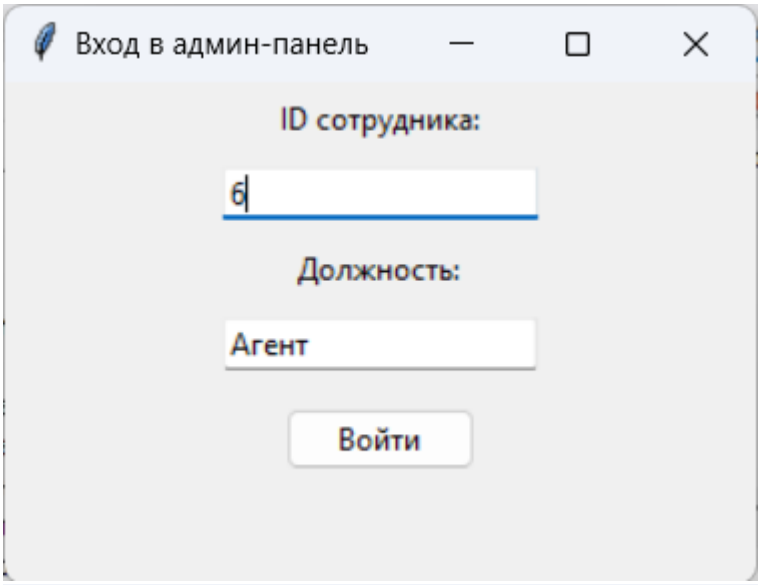


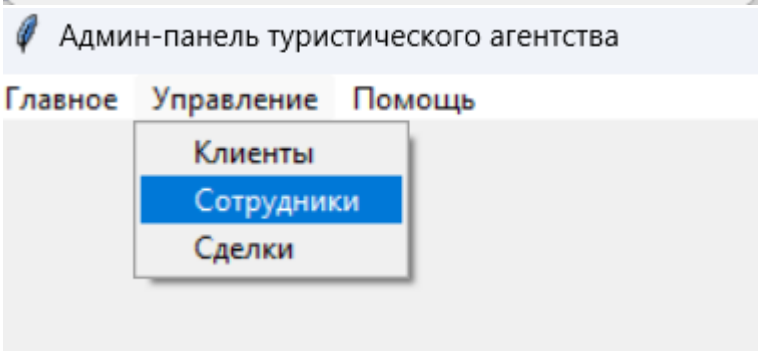


- Может просматривать но не может редактировть страницу с сотрудниками

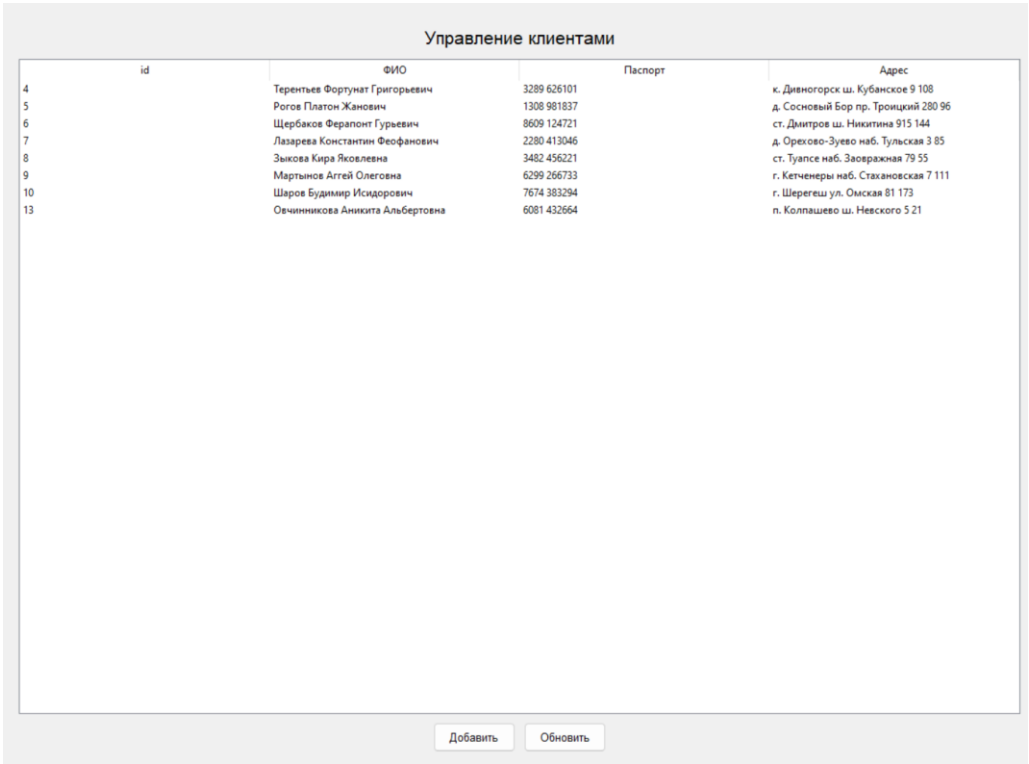


- Агент имеет меньше всего прав





- Может добавить но не может удалить клиентов



Заключение

В результате выполнения курсовой работы было создано работоспособное приложение для управления данными туристического агентства. Были применены полученные знания и навыки в области баз данных и объектно-ориентированного программирования на Python. Данный проект представляет собой хорошую 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, дата обращения: 22.11.2024)
2. Что такое Docker и как он работает
   (URL: https://skillbox.ru/media/code/kak-rabotaet-docker-podrobnyy-gayd-ot-tekhlida/, дата обращения: 22.11.2024)

3. Официальная документация Rabbitmq
   (URL: https://www.rabbitmq.com/docs, дата обращения: 23.11.2024)

4. REST API: что это такое и как работает
   (URL: https://skillbox.ru/media/code/rest-api-chto-eto-takoe-i-kak-rabotaet/, дата обращения: 23.11.2024)

5. Официальная документация Fast Api
   (URL: https://fastapi.tiangolo.com/tutorial/first-steps/, дата обращения: 25.11.2024)

6. Официальная документация SqlAlchemy
   (URL: https://www.sqlalchemy.org/, дата обращения: 25.11.2024)

7. Подробно про JWT/Хабр
   (URL: https://habr.com/ru/articles/842056/, дата обращения: 27.11.2024)

8. Redis - что это и для чего нужен? Пример использования
   (URL:https://skillbox.ru/media/code/znakomimsya_s_redis/, дата обращения: 27.11.2024)

9. Официальная документация SQLAdmin
   (URL:https://starlette-login.readthedocs.io/en/stable/tutorial/sqladmin/, дата обращения: 29.11.2024)

10. Официальная документация React TS
    (URL: https://react.dev/learn/typescript, дата обращения: 29.11.2024)
11. Официальная документация alembic
    (URL: https://alembic.sqlalchemy.org/en/latest/index.html, дата обращения: 02.12.2024)

5. Приложение

Полный код - https://github.com/Georgtor