МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| к.э.н., доцент |  |  |  |  | Т.Н.Елина |
| должность, уч. степень, звание |  | подпись, дата |  |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №5 |
| **Реализация приложения доступа к данным** по дисциплине: Базы данных |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. | 3843 |  |  |  | А.П. Конева |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2021

1. **Задача**

Реализовать приложение доступа к данным.

1. **Цель работы**

На одном из языков программирования спроектировать и реализовать приложение доступа к данным. Приложение должно выполнять следующие основные функции: ввод, редактирование, удаление данных, поиск и сортировка данных, отображение связанных таблиц на одной форме.

**Ход работы**

Используя ERwin Data Modeler, сгенерировали физическую схему базы данных из полученной в Лабораторной работе №4 физической модели данных через Forward Engineering**→** SchemaGeneration**.**

Для доступа к данным в MySQL необходимо десктопное приложение с графическим интерфейсом, в котором реализована возможность просмотра существующих в БД таблиц, добавления новых элементов в строки таблицы, редактирования элементов, удаления данных, сортировки данных, а также отображения связанных таблиц на одной форме.

Для взаимодействия с MySQL будем использовать язык программирования Python, а для реализации графической составляющей приложения – библиотеку Tkinter.

**Реализация**

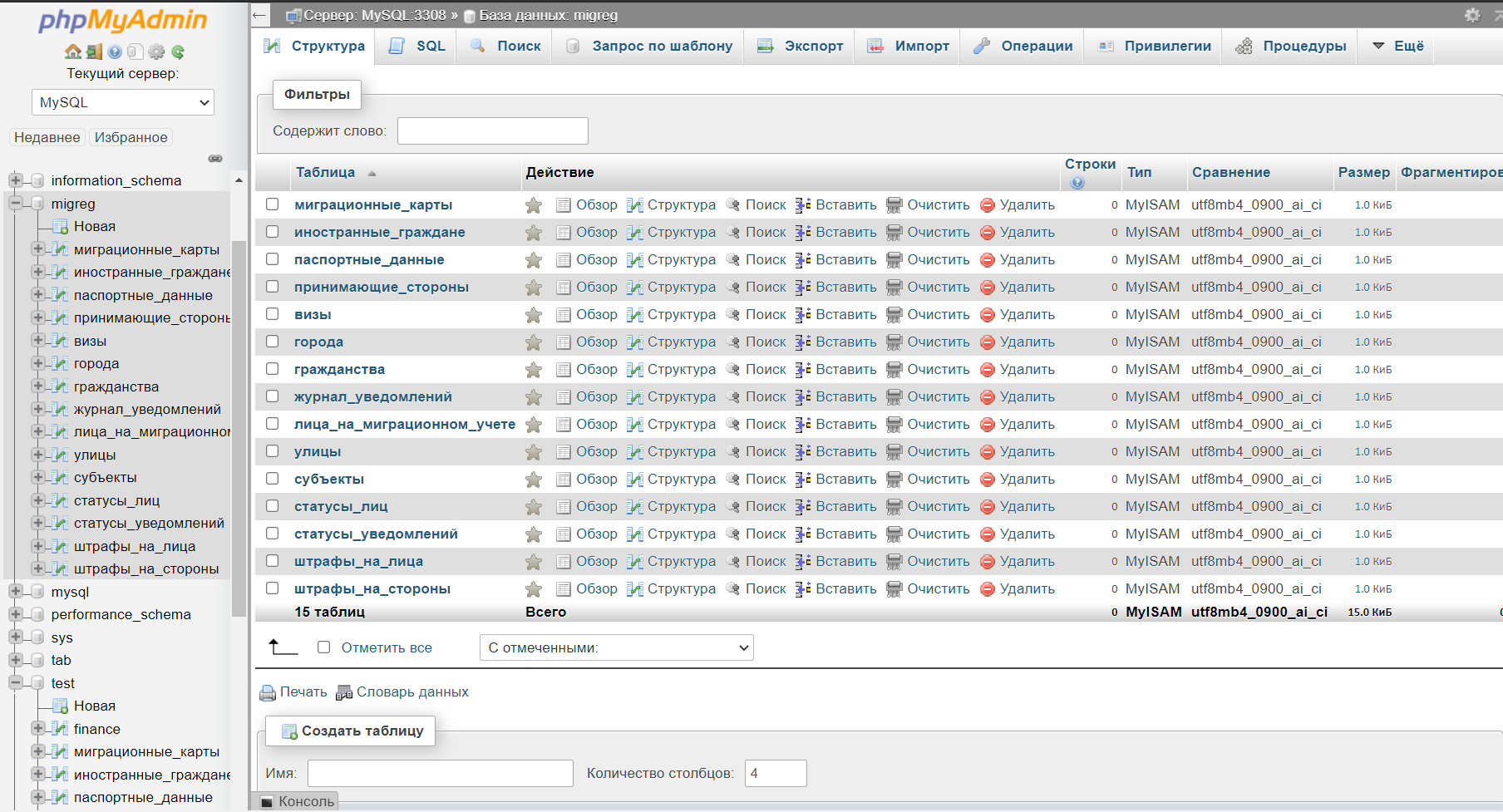
Для начала рассмотрим сформированную в MySQL базу данных migreg на рис.1.

Рис.1 База данных migreg в MySQL

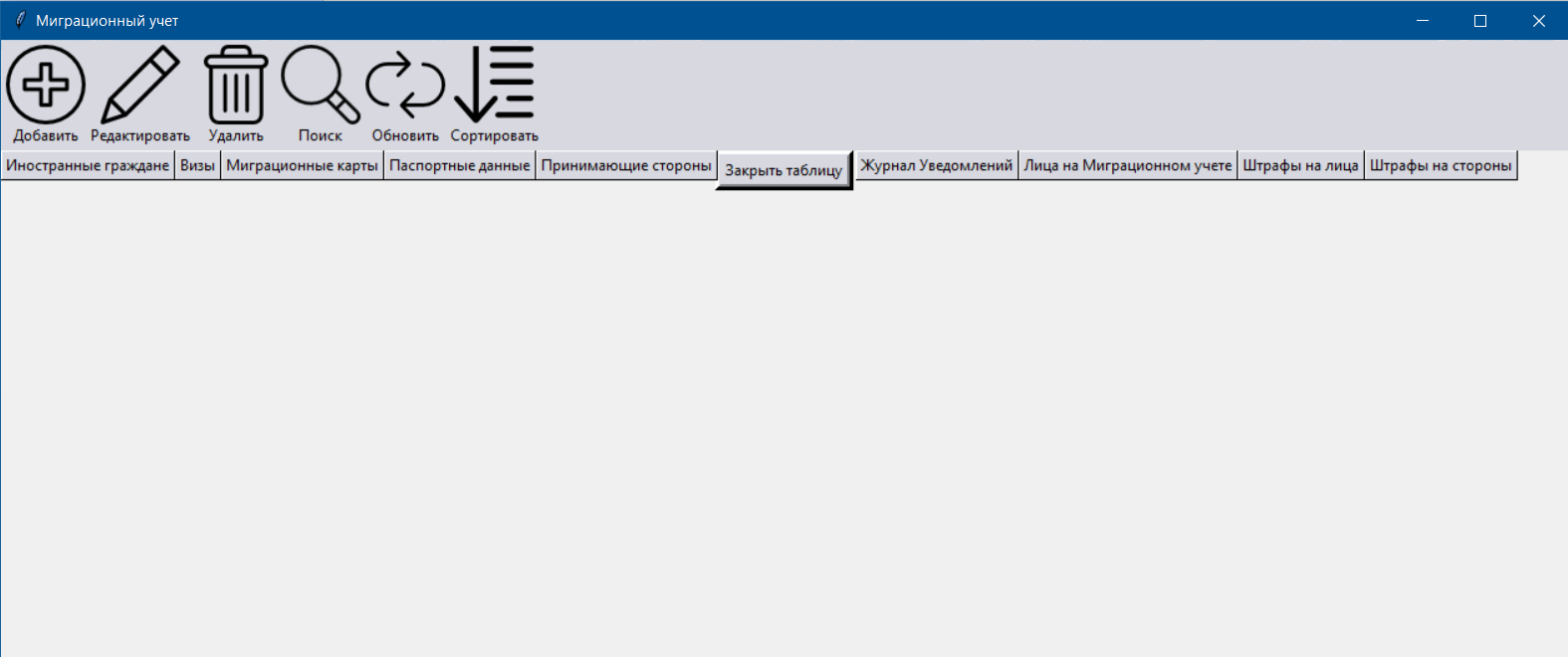
Перейдем к изучению главной страницы приложения. Главная страница изображена на рис.2 и представляет собой список существующих в БД таблиц. Также реализованы кнопки меню для работы с данными в таблице: «Добавить», «Редактировать», «Удалить», «Поиск», «Обновить», «Сортировать».

Рис.2 Главная страница

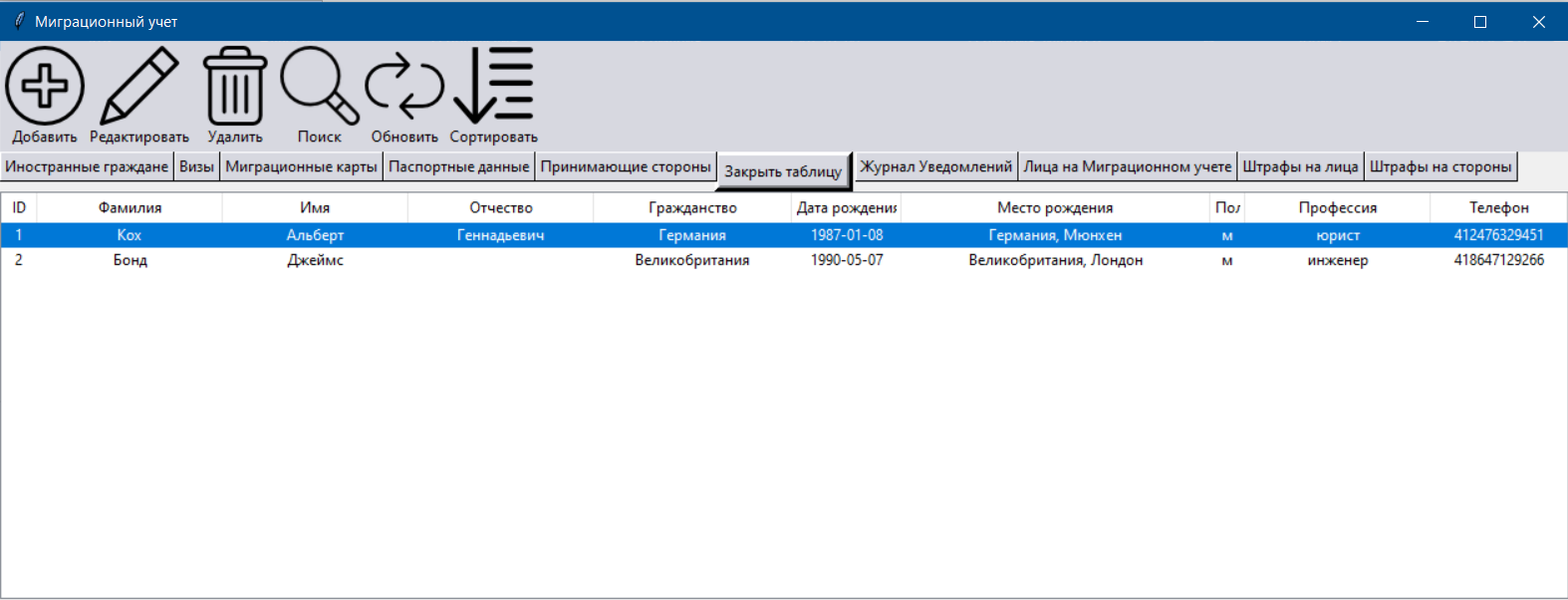
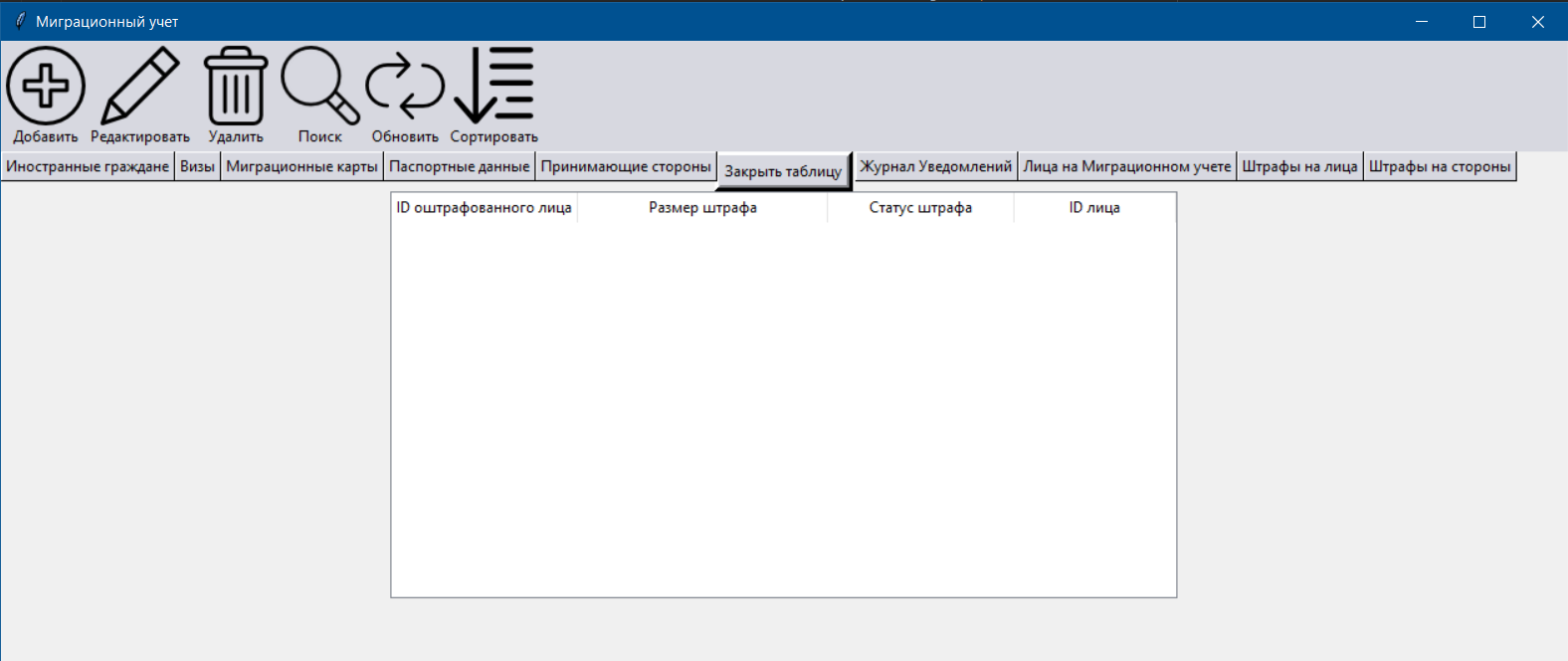
 Таблицы открываются по нажатию на кнопки с соответствующими названиями. Например, на рис. 3. показана таблица «Иностранные граждане», открытая после нажатия на кнопку «Иностранные граждане», а на рис.4. – таблица «Штрафы на лица».

Рис.3 Таблица «Иностранные граждане» на главной странице

 Рис.4 Таблица «Штрафы на лица» на главной странице

Для данной Лабораторной работы функции «Добавить», «Редактировать», «Удалить», «Поиск», «Обновить», «Сортировать» реализованы для таблицы «Иностранные граждане».

**Добавить**

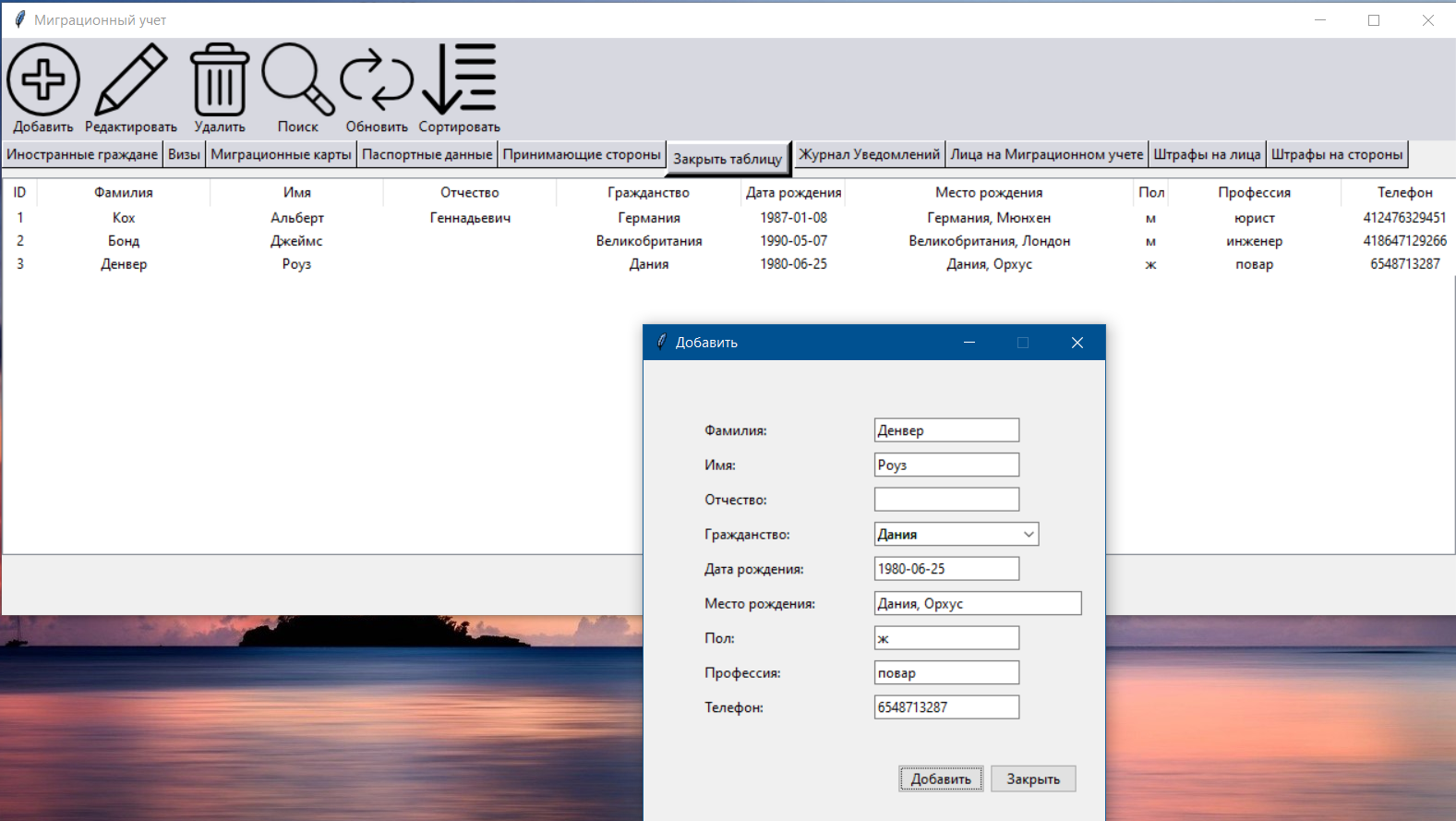
 Для добавления данных в таблицу необходимо нажать на кнопку “Добавить” и в открытом новом диалоговом окне внести данные в соответствующие поля ввода и нажать на кнопку «Добавить» в диалоговом окне, что изображено на рис.5

Рис.5 Демонстрация работы кнопки «Добавить»

**Редактировать**

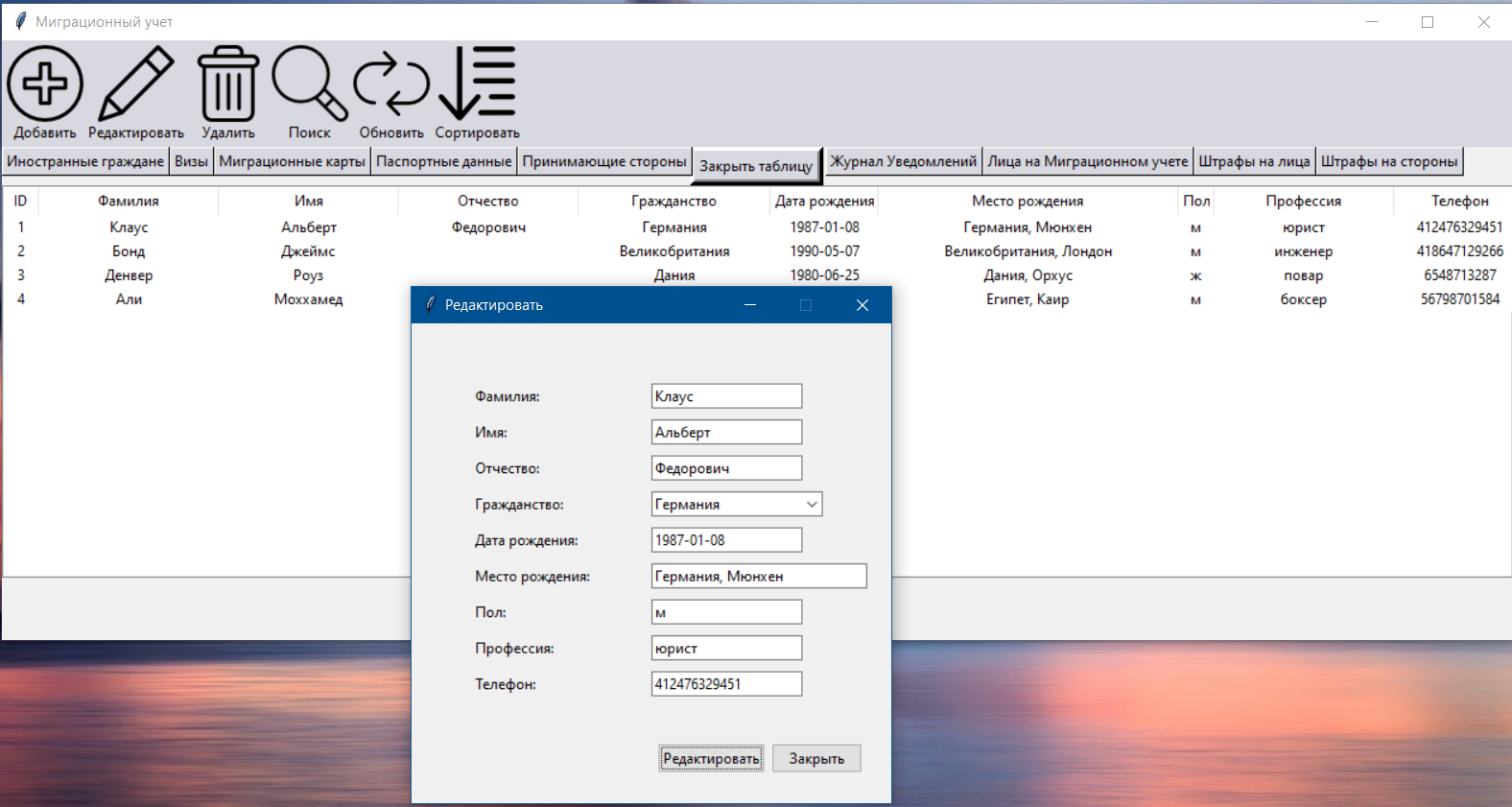
****Для редактирования данных в таблице необходимо нажать на строку, в которую нужно внести изменения, а после нажать на кнопку “Редактировать”. В открытом новом диалоговом окне в полях ввода демонстрируется текущая информация из строки. Требуется внести измененные данные в соответствующие поля ввода и нажать на кнопку «Редактировать» в диалоговом окне. Результат редактирования строки с ID=1 после нажатия изображен на рис.6.

Рис.6 Демонстрация работы кнопки «Редактировать»

**Удалить**

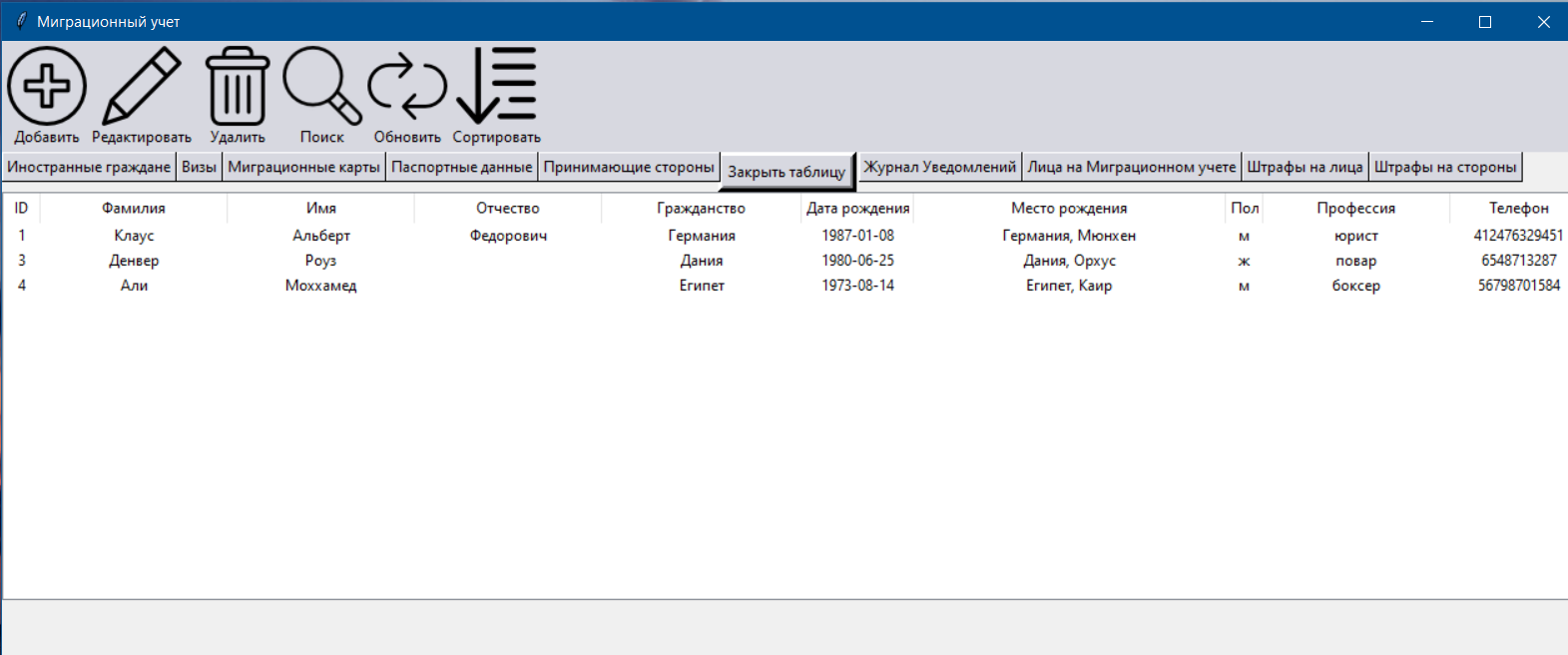
Для удаления данных из таблицы необходимо нажать на строку, которую нужно удалить, а после нажать на кнопку “Удалить” из меню. Например, выберем строку с ID=2 и удалим ее. Результат изображен на рис.7.

Рис.7 Демонстрация работы кнопки «Удалить»

**Поиск**

Для поиска данных в таблице необходимо нажать на кнопку «Поиск» в меню и в открывшемся диалоговом окне в поле ввода ввести ключевое слово, которое необходимо найти в таблице. Ключевое слово может быть как одиночным символом, так и набором символов, включая цифры, так и целым словом. После надо нажать на кнопку “Поиск” в диалоговом окне. Результаты поиска сразу демонстрируются в таблице в основном окне, что изображено на рис.8.

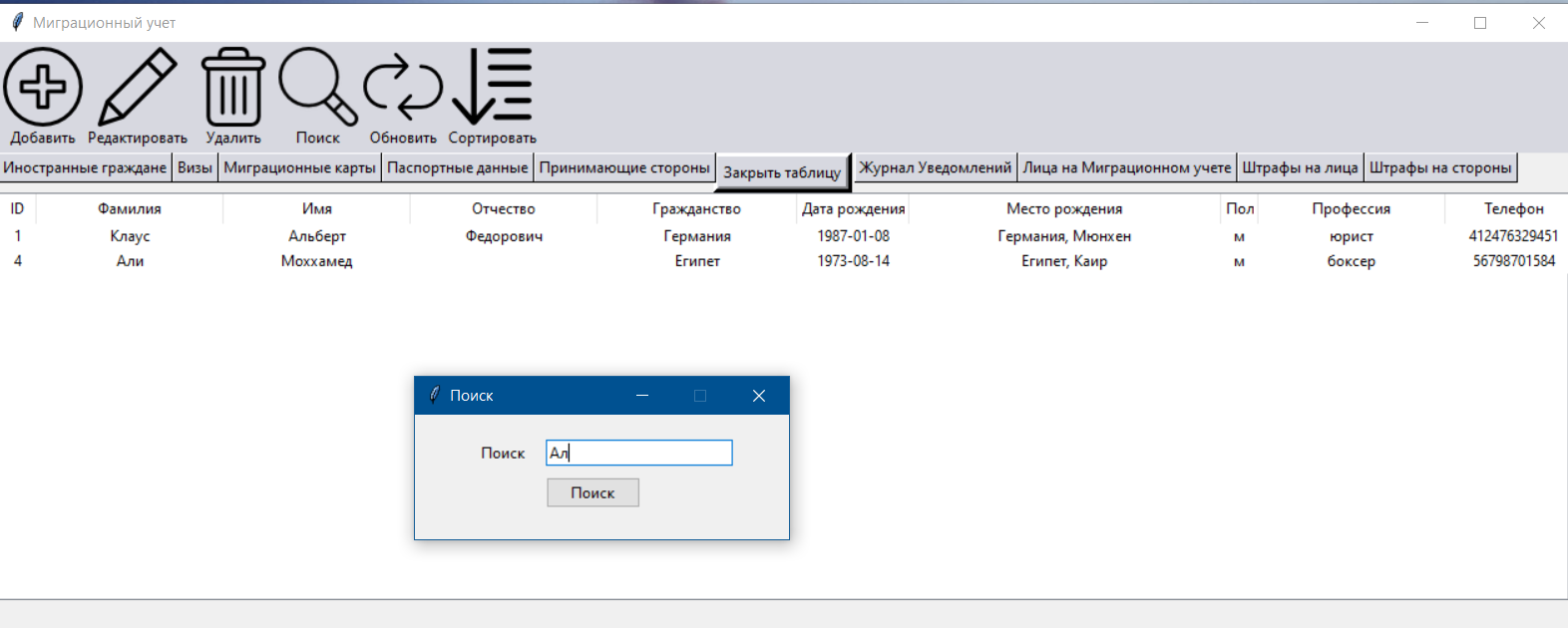


Рис.8 Демонстрация работы кнопки «Поиск»

**Обновить**

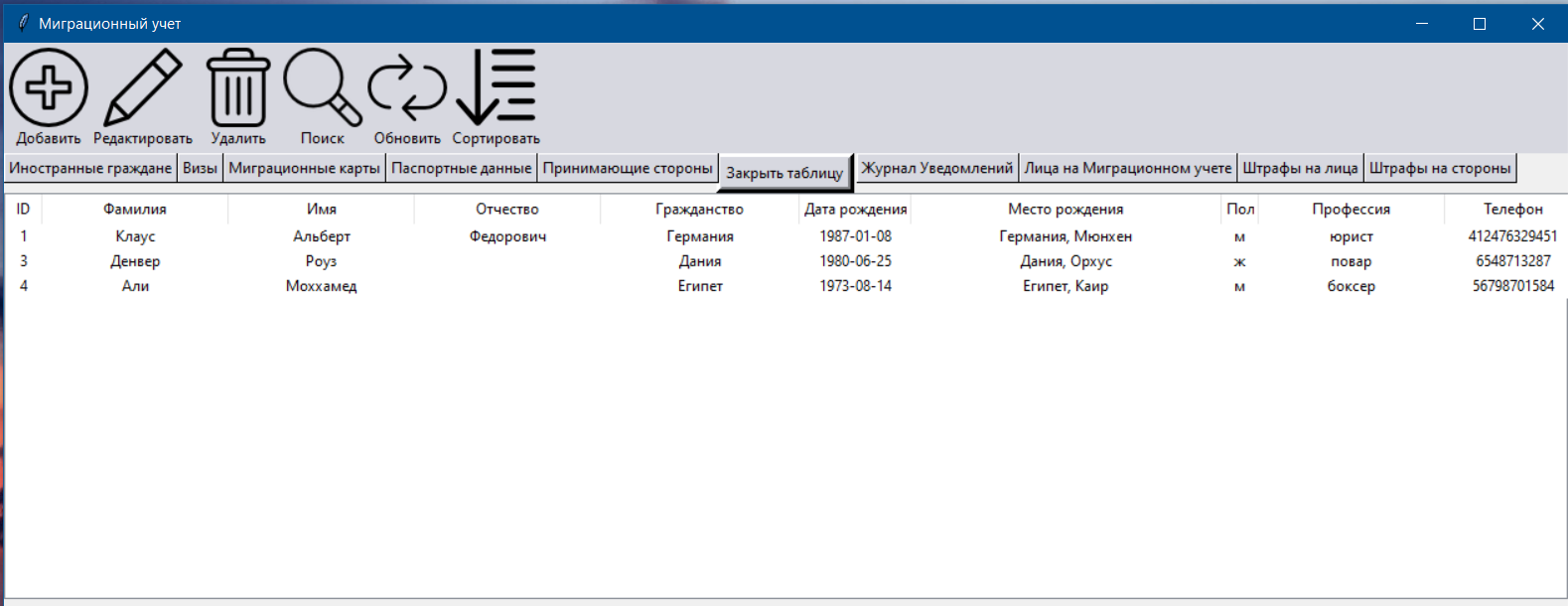
Кнопка «Обновить» позволяет восстановить все текущие записи из таблицы базы данных в интерфейсе программы после совершения таких действий как «Поиск», «Сортировка». На рис. 9 представлена демонстрация работы кнопки «Обновить» после выполнения поиска на рис. 8.

Рис.9 Демонстрация работы кнопки «Обновить»

**Сортировать**

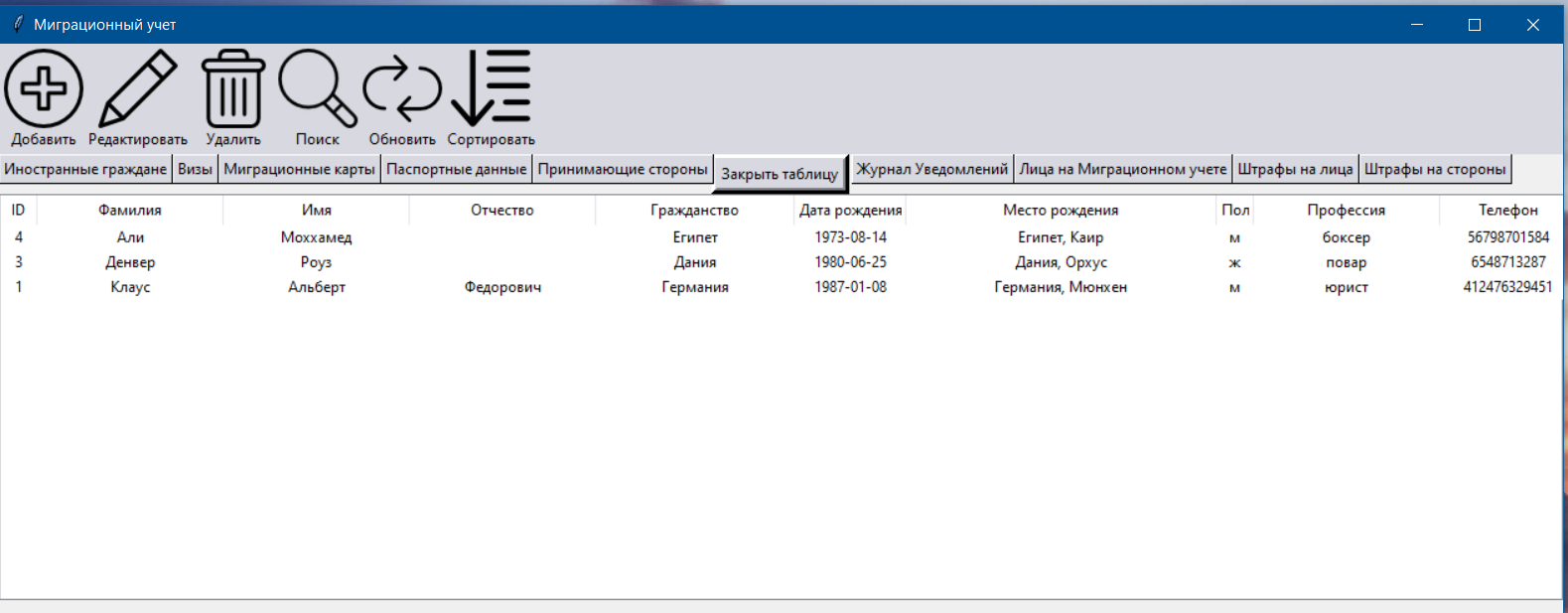
Для сортировки данных в столбце Фамилия в алфавитном порядке используется кнопка «Сортировать», результаты нажатия на которую представлены на рис.10.

Рис.10 Демонстрация работы кнопки «Сортировать» по столбцу Фамилия

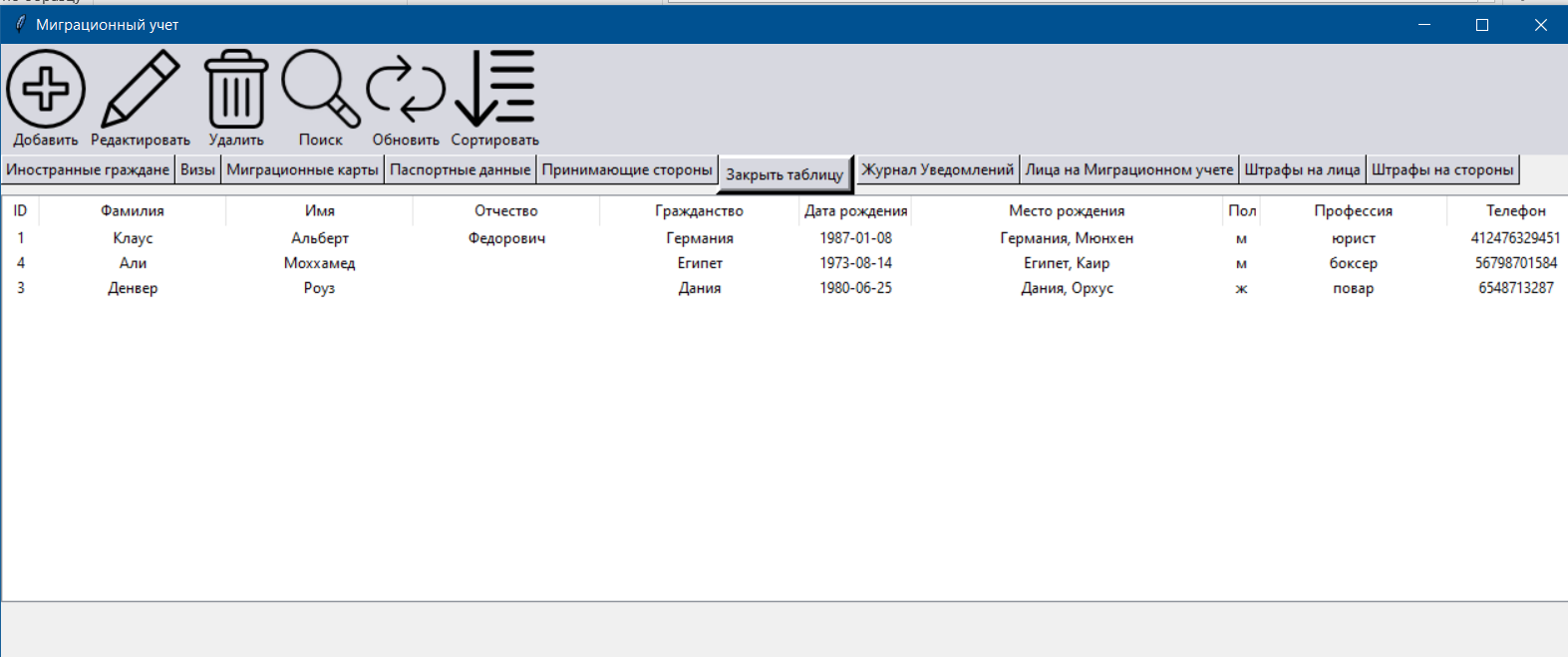
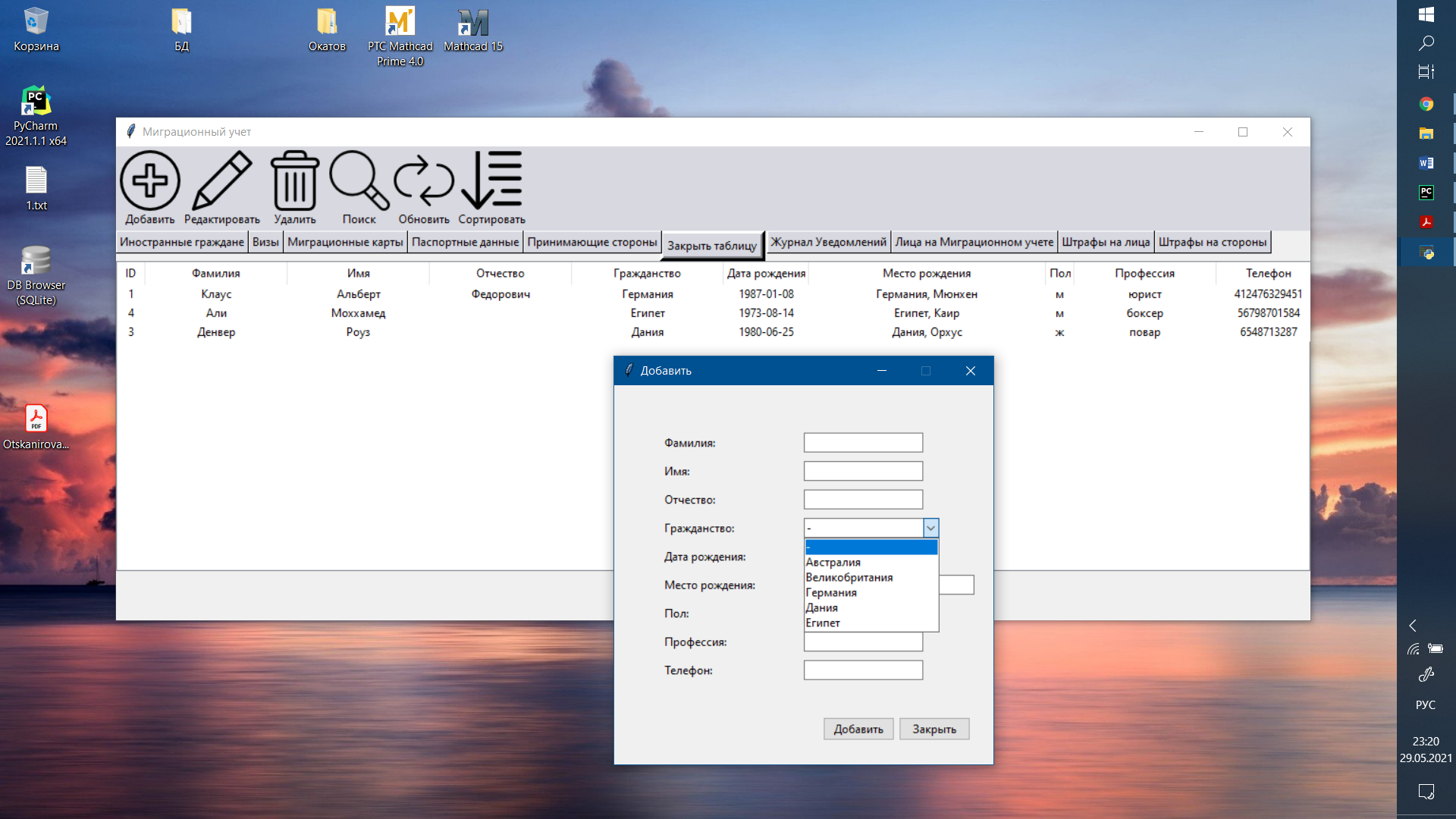
Для реализации сортировки по другим столбцам необходимо дважды кликнуть на название столбца. Например, на рис. 11 изображен результат сортировки столбца Имя.

Рис.11 Демонстрация работы кнопки «Сортировать» по столбцу Имя

**Отображение связанных таблиц на одной форме**

Можем видеть отображение выпадающего списка «Гражданства» при добавлении новых данных в таблицу «Иностранные граждане» в новом диалоговом окне «Добавить», что изображено на рис.12.

Рис.12 Выпадающий список значений из таблицы «Гражданства»

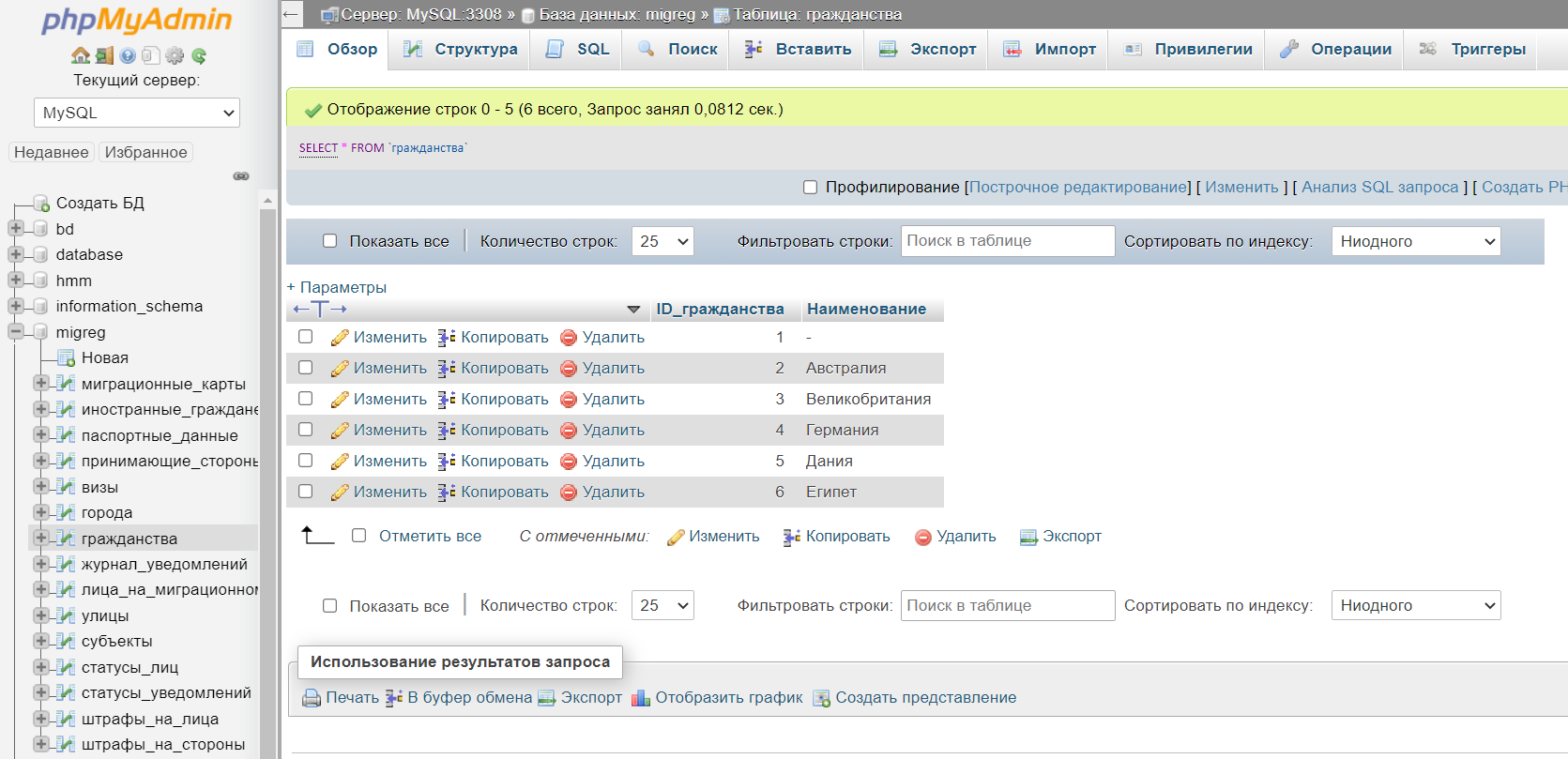
 На рис. 12 в таблице «Иностранные граждане» в столбце «Гражданство» видим данные из столбца «Наименование» таблицы «гражданства», содержимое которой отображено на рис.13.

Рис.13 Таблица «гражданства» в БД

**Вывод**

Для доступа к данным в MySQL спроектировали и реализовали десктопное приложение, код которого находится в Приложении, с удобным графическим интерфейсом, в котором реализована возможность просмотра существующих в БД таблиц, добавления новых элементов в строки таблицы, редактирования элементов, удаления данных, сортировки данных, а также отображения связанных таблиц на одной форме.

Для взаимодействия с MySQL был использован язык программирования Python, а для реализации графической составляющей приложения – библиотека Tkinter. Примеры работы приложения представлены на рис.2-12.

**Список литературы:**

1. Python MySQL Tutorial – Режим доступа: <https://pythobyte.com/python-mysql-tutorial-f329519b/>
2. Корректировка ячейки в таблице и сортировка данных в Python– Режим доступа: <https://www.piknad.ru/pytab3.php>
3. О прибытии иностранного гражданина – Режим доступа: <https://support.kontur.ru/pages/viewpage.action?pageId=4589781>
4. TreeView – Режим доступа: <https://tk-tutorial.readthedocs.io/en/latest/tree/tree.html>
5. Forward Engineering/Schema Generation– Режим доступа: <https://erwin.com/bookshelf/public_html/2020R2/Content/User%20Guides/erwin%20Help/foward_engineering_schema_generation.html>

**Приложение. Листинг программы**

**Erwin**

CREATE TABLE Визы

(

К\_номер\_визы INTEGER NOT NULL,

Дата\_выдачи DATE NULL,

Срок\_действия DATE NULL,

Категория\_визы VARCHAR(20) NULL,

Цель\_визита VARCHAR(20) NULL,

Идентификатор\_визы INTEGER NULL,

ID\_иностранного\_гражданина INTEGER NULL,

К\_Номер\_паспорта INTEGER NULL,

PRIMARY KEY (К\_номер\_визы)

);

CREATE TABLE Города

(

ID\_города INTEGER NOT NULL AUTO\_INCREMENT,

Наименование VARCHAR(20) NULL,

ID\_субъекта INTEGER NOT NULL,

PRIMARY KEY (ID\_города,ID\_субъекта)

);

CREATE TABLE Гражданства

(

ID\_гражданства INTEGER NOT NULL AUTO\_INCREMENT,

Наименование VARCHAR(20) NULL,

PRIMARY KEY (ID\_гражданства)

);

CREATE TABLE Журнал\_Уведомлений

(

К\_номер\_Уведомления INTEGER NOT NULL AUTO\_INCREMENT,

К\_дата\_подачи DATE NOT NULL,

ID\_статуса INTEGER NULL,

ID\_принимающей\_стороны INTEGER NULL,

ID\_лица INTEGER NULL,

PRIMARY KEY (К\_номер\_Уведомления,К\_дата\_подачи)

);

CREATE TABLE Иностранные\_граждане

(

ID\_иностранного\_гражданина INTEGER NOT NULL AUTO\_INCREMENT,

Фамилия VARCHAR(20) NULL,

Имя VARCHAR(20) NULL,

Отчество VARCHAR(20) NULL,

Дата\_рождения DATE NULL,

Место\_рождения VARCHAR(20) NULL,

Пол VARCHAR(20) NULL,

Профессия VARCHAR(20) NULL,

Телефон INTEGER NULL,

ID\_гражданства INTEGER NULL,

PRIMARY KEY (ID\_иностранного\_гражданина)

);

CREATE TABLE Лица\_на\_Миграционном\_учете

(

ID\_лица INTEGER NOT NULL AUTO\_INCREMENT,

ID\_иностранного\_гражданина INTEGER NULL,

Дом INTEGER NULL,

Корпус INTEGER NULL,

Квартира INTEGER NULL,

Срок\_до DATE NULL,

ID\_статуса\_лица INTEGER NULL,

ID\_улицы INTEGER NULL,

ID\_города INTEGER NULL,

ID\_субъекта INTEGER NULL,

К\_Номер\_паспорта INTEGER NULL,

PRIMARY KEY (ID\_лица)

);

CREATE TABLE Миграционные\_карты

(

К\_Номер\_Мигр\_карты INTEGER NOT NULL,

КПП\_въезда VARCHAR(20) NULL,

Способ\_пересечения\_границы VARCHAR(20) NULL,

Дата\_въезда DATE NULL,

Срок\_пребывания DATE NULL,

ID\_иностранного\_гражданина INTEGER NULL,

К\_Номер\_паспорта INTEGER NULL,

PRIMARY KEY (К\_Номер\_Мигр\_карты)

);

CREATE TABLE Паспортные\_данные

(

К\_Номер\_паспорта INTEGER NOT NULL,

Кем\_выдан VARCHAR(20) NULL,

Дата\_выдачи DATE NULL,

ID\_иностранного\_гражданина INTEGER NULL,

PRIMARY KEY (К\_Номер\_паспорта)

);

CREATE TABLE Принимающие\_стороны

(

ID\_принимающей\_стороны INTEGER NOT NULL AUTO\_INCREMENT,

ФИО\_Наименование VARCHAR(20) NULL,

Дата\_рождения\_Регистрации DATE NULL,

Серия\_паспорта INTEGER NULL,

Номер\_паспорта INTEGER NULL,

PRIMARY KEY (ID\_принимающей\_стороны)

);

CREATE TABLE Статусы\_лиц

(

ID\_статуса\_лица INTEGER NOT NULL AUTO\_INCREMENT,

Статус\_лица VARCHAR(20) NULL,

PRIMARY KEY (ID\_статуса\_лица)

);

CREATE TABLE Статусы\_Уведомлений

(

ID\_статуса INTEGER NOT NULL AUTO\_INCREMENT,

Наименование VARCHAR(20) NULL,

PRIMARY KEY (ID\_статуса)

);

CREATE TABLE Субъекты

(

ID\_субъекта INTEGER NOT NULL AUTO\_INCREMENT,

Наименование VARCHAR(20) NULL,

PRIMARY KEY (ID\_субъекта)

);

CREATE TABLE Улицы

(

ID\_улицы INTEGER NOT NULL AUTO\_INCREMENT,

Наименование VARCHAR(20) NULL,

ID\_города INTEGER NOT NULL,

ID\_субъекта INTEGER NOT NULL,

PRIMARY KEY (ID\_улицы,ID\_города,ID\_субъекта)

);

CREATE TABLE Штрафы\_на\_лица

(

ID\_оштрафованного\_лица INTEGER NOT NULL AUTO\_INCREMENT,

Размер\_штрафа INTEGER NULL,

Статус\_штрафа VARCHAR(20) NULL,

ID\_лица INTEGER NULL,

PRIMARY KEY (ID\_оштрафованного\_лица)

);

CREATE TABLE Штрафы\_на\_стороны

(

ID\_оштрафованной\_стороны INTEGER NOT NULL AUTO\_INCREMENT,

Размер\_штрафа INTEGER NULL,

Статус\_штрафа VARCHAR(20) NULL,

ID\_принимающей\_стороны INTEGER NULL,

PRIMARY KEY (ID\_оштрафованной\_стороны)

);

ALTER TABLE Визы

ADD FOREIGN KEY R\_67 (ID\_иностранного\_гражданина) REFERENCES Иностранные\_граждане (ID\_иностранного\_гражданина) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Визы

ADD FOREIGN KEY R\_77 (К\_Номер\_паспорта) REFERENCES Паспортные\_данные (К\_Номер\_паспорта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Города

ADD FOREIGN KEY R\_62 (ID\_субъекта) REFERENCES Субъекты (ID\_субъекта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Журнал\_Уведомлений

ADD FOREIGN KEY R\_8 (ID\_статуса) REFERENCES Статусы\_Уведомлений (ID\_статуса) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Журнал\_Уведомлений

ADD FOREIGN KEY R\_9 (ID\_принимающей\_стороны) REFERENCES Принимающие\_стороны (ID\_принимающей\_стороны) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Журнал\_Уведомлений

ADD FOREIGN KEY R\_74 (ID\_лица) REFERENCES Лица\_на\_Миграционном\_учете (ID\_лица) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Иностранные\_граждане

ADD FOREIGN KEY R\_65 (ID\_гражданства) REFERENCES Гражданства (ID\_гражданства) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Лица\_на\_Миграционном\_учете

ADD FOREIGN KEY R\_40 (ID\_иностранного\_гражданина) REFERENCES Иностранные\_граждане (ID\_иностранного\_гражданина) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Лица\_на\_Миграционном\_учете

ADD FOREIGN KEY R\_56 (ID\_статуса\_лица) REFERENCES Статусы\_лиц (ID\_статуса\_лица) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Лица\_на\_Миграционном\_учете

ADD FOREIGN KEY R\_63 (ID\_улицы, ID\_города, ID\_субъекта) REFERENCES Улицы (ID\_улицы, ID\_города, ID\_субъекта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Лица\_на\_Миграционном\_учете

ADD FOREIGN KEY R\_76 (К\_Номер\_паспорта) REFERENCES Паспортные\_данные (К\_Номер\_паспорта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Миграционные\_карты

ADD FOREIGN KEY R\_72 (ID\_иностранного\_гражданина) REFERENCES Иностранные\_граждане (ID\_иностранного\_гражданина) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Миграционные\_карты

ADD FOREIGN KEY R\_81 (К\_Номер\_паспорта) REFERENCES Паспортные\_данные (К\_Номер\_паспорта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Паспортные\_данные

ADD FOREIGN KEY R\_75 (ID\_иностранного\_гражданина) REFERENCES Иностранные\_граждане (ID\_иностранного\_гражданина) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Улицы

ADD FOREIGN KEY R\_61 (ID\_города, ID\_субъекта) REFERENCES Города (ID\_города, ID\_субъекта) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Штрафы\_на\_лица

ADD FOREIGN KEY R\_53 (ID\_лица) REFERENCES Лица\_на\_Миграционном\_учете (ID\_лица) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE Штрафы\_на\_стороны

ADD FOREIGN KEY R\_82 (ID\_принимающей\_стороны) REFERENCES Принимающие\_стороны (ID\_принимающей\_стороны) ON DELETE CASCADE ON UPDATE CASCADE;

**main.py**

import tkinter as tk  
from tkinter import ttk  
import pymysql  
  
class Main(tk.Frame):  
 def \_\_init\_\_(self, root):  
 super().\_\_init\_\_(root)  
 self.init\_main()  
 self.db = db  
 self.view\_records()  
  
 btn\_table = tk.Button(text="Закрыть таблицу", bg='#d7d8e0', bd=5, command=self.close\_table)  
 btn\_table1 = tk.Button(text="Иностранные граждане", bg='#d7d8e0', bd=1, command=self.open\_tableInGr)  
 btn\_table2 = tk.Button(text="Визы", bg='#d7d8e0', bd=1, command=self.open\_tableViz)  
 btn\_table3 = tk.Button(text="Миграционные карты", bg='#d7d8e0', bd=1, command=self.open\_tableMC)  
 btn\_table4 = tk.Button(text="Паспортные данные", bg='#d7d8e0', bd=1, command=self.open\_tablePD)  
 btn\_table5 = tk.Button(text="Принимающие стороны", bg='#d7d8e0', bd=1, command=self.open\_tablePS)  
 btn\_table6 = tk.Button(text="Журнал Уведомлений", bg='#d7d8e0', bd=1, command=self.open\_tableJUv)  
 btn\_table7 = tk.Button(text="Лица на Миграционном учете", bg='#d7d8e0', bd=1, command=self.open\_tablePersonsOnMigReg)  
 btn\_table8 = tk.Button(text="Штрафы на лица", bg='#d7d8e0', bd=1, command=self.open\_tableFinesOnFaces)  
 btn\_table9 = tk.Button(text="Штрафы на стороны", bg='#d7d8e0', bd=1, command=self.open\_tableFinesOnSides)  
 btn\_table.pack(side=tk.TOP)  
 btn\_table1.place(x=0, y=89)  
 btn\_table2.place(x=140, y=89)  
 btn\_table3.place(x=177, y=89)  
 btn\_table4.place(x=308, y=89)  
 btn\_table5.place(x=430, y=89)  
 btn\_table6.place(x=687, y=89)  
 btn\_table7.place(x=818, y=89)  
 btn\_table8.place(x=994, y=89)  
 btn\_table9.place(x=1096, y=89)  
  
  
  
  
 def init\_main(self):  
  
 toolbar = tk.Frame(bg='#d7d8e0', bd=2)  
 toolbar.pack(side=tk.TOP, fill=tk.X)  
  
 self.add\_img = tk.PhotoImage(file='add.png')  
 btn\_open\_add\_dialog = tk.Button(toolbar, text='Добавить', command=self.open\_add\_dialog, bg='#d7d8e0', bd=0,  
 compound=tk.TOP, image=self.add\_img)  
 btn\_open\_add\_dialog.pack(side=tk.LEFT)  
  
 self.edit\_img = tk.PhotoImage(file='edit.png')  
 btn\_edit\_dialog = tk.Button(toolbar, text='Редактировать', bg='#d7d8e0', bd=0, image=self.edit\_img,  
 compound=tk.TOP, command=self.open\_edit\_dialog)  
 btn\_edit\_dialog.pack(side=tk.LEFT)  
  
 self.delete\_img = tk.PhotoImage(file='delete.png')  
 btn\_delete = tk.Button(toolbar, text='Удалить', bg='#d7d8e0', bd=0, image=self.delete\_img,  
 compound=tk.TOP, command=self.delete\_records)  
 btn\_delete.pack(side=tk.LEFT)  
  
 self.search\_img = tk.PhotoImage(file='search.png')  
 btn\_search = tk.Button(toolbar, text='Поиск', bg='#d7d8e0', bd=0, image=self.search\_img,  
 compound=tk.TOP, command=self.open\_search\_dialog)  
 btn\_search.pack(side=tk.LEFT)  
  
 self.update\_img = tk.PhotoImage(file='update.png')  
 btn\_update = tk.Button(toolbar, text='Обновить', bg='#d7d8e0', bd=0, image=self.update\_img,  
 compound=tk.TOP, command=self.view\_records)  
 btn\_update.pack(side=tk.LEFT)  
  
 self.sort\_img = tk.PhotoImage(file='sort.png')  
 btn\_sort = tk.Button(toolbar, text='Сортировать', bg='#d7d8e0', bd=0, image=self.sort\_img,  
 compound=tk.TOP, command=self.sort\_record1)  
 btn\_sort.pack(side=tk.LEFT)  
  
 self.tree = ttk.Treeview(self, columns=( 'ID\_иностранного\_гражданина', 'Фамилия', 'Имя', 'Отчество',  
 'ID\_гражданство', 'Дата\_рождения', 'Место\_рождения', 'Пол', 'Профессия', 'Телефон'), height=15, show='headings')  
  
 self.tree.column('ID\_иностранного\_гражданина', width=30, anchor=tk.CENTER)  
 self.tree.column('Фамилия', width=150, anchor=tk.CENTER)  
 self.tree.column('Имя', width=150, anchor=tk.CENTER)  
 self.tree.column('Отчество', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_гражданство', width=160, anchor=tk.CENTER)  
 self.tree.column('Дата\_рождения', width=90, anchor=tk.CENTER)  
 self.tree.column('Место\_рождения', width=250, anchor=tk.CENTER)  
 self.tree.column('Пол', width=30, anchor=tk.CENTER)  
 self.tree.column('Профессия', width=150, anchor=tk.CENTER)  
 self.tree.column('Телефон', width=110, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID', command=self.sort\_record )  
 self.tree.heading('Фамилия', text='Фамилия', command=self.sort\_record1)  
 self.tree.heading('Имя', text='Имя', command=self.sort\_record2)  
 self.tree.heading('Отчество', text='Отчество', command=self.sort\_record3)  
 self.tree.heading('ID\_гражданство', text='Гражданство', command=self.sort\_record4)  
 self.tree.heading('Дата\_рождения', text='Дата рождения', command=self.sort\_record5)  
 self.tree.heading('Место\_рождения', text='Место рождения', command=self.sort\_record6)  
 self.tree.heading('Пол', text='Пол', command=self.sort\_record7)  
 self.tree.heading('Профессия', text='Профессия', command=self.sort\_record8)  
 self.tree.heading('Телефон', text='Телефон', command=self.sort\_record9)  
  
 self.tree.pack()  
  
  
 def open\_tableInGr(self):  
 self.tree = ttk.Treeview(self, columns=( 'ID\_иностранного\_гражданина', 'Фамилия', 'Имя', 'Отчество',  
 'ID\_гражданство', 'Дата\_рождения', 'Место\_рождения', 'Пол', 'Профессия', 'Телефон'), height=15, show='headings')  
  
 self.tree.column('ID\_иностранного\_гражданина', width=40, anchor=tk.CENTER)  
 self.tree.column('Фамилия', width=150, anchor=tk.CENTER)  
 self.tree.column('Имя', width=150, anchor=tk.CENTER)  
 self.tree.column('Отчество', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_гражданство', width=160, anchor=tk.CENTER)  
 self.tree.column('Дата\_рождения', width=90, anchor=tk.CENTER)  
 self.tree.column('Место\_рождения', width=250, anchor=tk.CENTER)  
 self.tree.column('Пол', width=35, anchor=tk.CENTER)  
 self.tree.column('Профессия', width=150, anchor=tk.CENTER)  
 self.tree.column('Телефон', width=120, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID')  
 self.tree.heading('Фамилия', text='Фамилия')  
 self.tree.heading('Имя', text='Имя')  
 self.tree.heading('Отчество', text='Отчество')  
 self.tree.heading('ID\_гражданство', text='Гражданство')  
 self.tree.heading('Дата\_рождения', text='Дата рождения')  
 self.tree.heading('Место\_рождения', text='Место рождения')  
 self.tree.heading('Пол', text='Пол')  
 self.tree.heading('Профессия', text='Профессия')  
 self.tree.heading('Телефон', text='Телефон')  
  
 self.tree.pack()  
  
 def open\_tableViz(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'К\_номер\_визы', 'Дата\_выдачи', 'Срок\_действия', 'Категория\_визы', 'Цель\_визита', 'Идентификатор\_визы',  
 'ID\_иностранного\_гражданина', 'К\_Номер\_паспорта'), height=15, show='headings')  
  
 self.tree.column('К\_номер\_визы', width=160, anchor=tk.CENTER)  
 self.tree.column('Дата\_выдачи', width=120, anchor=tk.CENTER)  
 self.tree.column('Срок\_действия', width=120, anchor=tk.CENTER)  
 self.tree.column('Категория\_визы', width=180, anchor=tk.CENTER)  
 self.tree.column('Цель\_визита', width=160, anchor=tk.CENTER)  
 self.tree.column('Идентификатор\_визы', width=160, anchor=tk.CENTER)  
 self.tree.column('ID\_иностранного\_гражданина', width=180, anchor=tk.CENTER)  
 self.tree.column('К\_Номер\_паспорта', width=160, anchor=tk.CENTER)  
  
 self.tree.heading('К\_номер\_визы', text='Номер визы')  
 self.tree.heading('Дата\_выдачи', text='Дата выдачи')  
 self.tree.heading('Срок\_действия', text='Срок действия')  
 self.tree.heading('Категория\_визы', text='Категория визы')  
 self.tree.heading('Цель\_визита', text='Цель визита')  
 self.tree.heading('Идентификатор\_визы', text='Идентификатор визы')  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID иностранного гражданина')  
 self.tree.heading('К\_Номер\_паспорта', text='Номер паспорта')  
  
 self.tree.pack()  
  
 def open\_tableMC(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'К\_Номер\_Мигр\_карты', 'КПП\_въезда', 'Способ\_пересечения\_границы', 'Дата\_въезда', 'Срок\_пребывания',  
 'ID\_иностранного\_гражданина', 'К\_Номер\_паспорта'), height=15, show='headings')  
  
 self.tree.column('К\_Номер\_Мигр\_карты', width=180, anchor=tk.CENTER)  
 self.tree.column('КПП\_въезда', width=180, anchor=tk.CENTER)  
 self.tree.column('Способ\_пересечения\_границы', width=180, anchor=tk.CENTER)  
 self.tree.column('Дата\_въезда', width=150, anchor=tk.CENTER)  
 self.tree.column('Срок\_пребывания', width=160, anchor=tk.CENTER)  
 self.tree.column('ID\_иностранного\_гражданина', width=200, anchor=tk.CENTER)  
 self.tree.column('К\_Номер\_паспорта', width=160, anchor=tk.CENTER)  
  
 self.tree.heading('К\_Номер\_Мигр\_карты', text='Номер Миграционной карты')  
 self.tree.heading('КПП\_въезда', text='КПП въезда')  
 self.tree.heading('Способ\_пересечения\_границы', text='Способ пересечения границы')  
 self.tree.heading('Дата\_въезда', text='Дата въезда')  
 self.tree.heading('Срок\_пребывания', text='Срок пребывания')  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID иностранного гражданина')  
 self.tree.heading('К\_Номер\_паспорта', text='Номер паспорта')  
  
 self.tree.pack()  
  
 def open\_tablePD(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'К\_Номер\_паспорта', 'Кем\_выдан', 'Дата\_выдачи', 'ID\_иностранного\_гражданина'), height=15, show='headings')  
 self.tree.column('К\_Номер\_паспорта', width=160, anchor=tk.CENTER)  
 self.tree.column('Кем\_выдан', width=300, anchor=tk.CENTER)  
 self.tree.column('Дата\_выдачи', width=130, anchor=tk.CENTER)  
 self.tree.column('ID\_иностранного\_гражданина', width=200, anchor=tk.CENTER)  
  
 self.tree.heading('К\_Номер\_паспорта', text='Номер паспорта')  
 self.tree.heading('Кем\_выдан', text='Кем выдан')  
 self.tree.heading('Дата\_выдачи', text='Дата выдачи')  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID иностранного гражданина')  
  
 self.tree.pack()  
  
 def open\_tablePS(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'ID\_принимающей\_стороны', 'ФИО\_Наименование', 'Дата\_рождения\_Регистрации',  
 'Серия\_паспорта', 'Номер\_паспорта'), height=15, show='headings')  
 self.tree.column('ID\_принимающей\_стороны', width=160, anchor=tk.CENTER)  
 self.tree.column('ФИО\_Наименование', width=350, anchor=tk.CENTER)  
 self.tree.column('Дата\_рождения\_Регистрации', width=200, anchor=tk.CENTER)  
 self.tree.column('Серия\_паспорта', width=140, anchor=tk.CENTER)  
 self.tree.column('Номер\_паспорта', width=160, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_принимающей\_стороны', text='ID принимающей стороны')  
 self.tree.heading('ФИО\_Наименование', text='ФИО/Наименование')  
 self.tree.heading('Дата\_рождения\_Регистрации', text='Дата рождения/Регистрации')  
 self.tree.heading('Серия\_паспорта', text='Серия паспорта')  
 self.tree.heading('Номер\_паспорта', text='Номер паспорта')  
  
 self.tree.pack()  
  
 def open\_tableJUv(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'К\_номер\_Уведомления', 'К\_дата\_подачи', 'ID\_статуса', 'ID\_принимающей\_стороны', 'ID\_лица'), height=15, show='headings')  
  
 self.tree.column('К\_номер\_Уведомления', width=150, anchor=tk.CENTER)  
 self.tree.column('К\_дата\_подачи', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_статуса', width=210, anchor=tk.CENTER)  
 self.tree.column('ID\_принимающей\_стороны', width=170, anchor=tk.CENTER)  
 self.tree.column('ID\_лица', width=140, anchor=tk.CENTER)  
  
 self.tree.heading('К\_номер\_Уведомления', text='Номер Уведомления')  
 self.tree.heading('К\_дата\_подачи', text='Дата подачи')  
 self.tree.heading('ID\_статуса', text='Статус Уведомления')  
 self.tree.heading('ID\_принимающей\_стороны', text='ID принимающей стороны')  
 self.tree.heading('ID\_лица', text='ID лица')  
  
 self.tree.pack()  
  
 def open\_tablePersonsOnMigReg(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'ID\_лица', 'ID\_иностранного\_гражданина', 'Дом', 'Корпус', 'Квартира', 'Срок\_до', 'ID\_статуса\_лица',  
 'ID\_улицы', 'ID\_города', 'ID\_субъекта', 'К\_Номер\_паспорта'), height=15, show='headings')  
  
 self.tree.column('ID\_лица', width=50, anchor=tk.CENTER)  
 self.tree.column('ID\_иностранного\_гражданина', width=170, anchor=tk.CENTER)  
 self.tree.column('Дом', width=60, anchor=tk.CENTER)  
 self.tree.column('Корпус', width=50, anchor=tk.CENTER)  
 self.tree.column('Квартира', width=70, anchor=tk.CENTER)  
 self.tree.column('Срок\_до', width=90, anchor=tk.CENTER)  
 self.tree.column('ID\_статуса\_лица', width=160, anchor=tk.CENTER)  
 self.tree.column('ID\_улицы', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_города', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_субъекта', width=150, anchor=tk.CENTER)  
 self.tree.column('К\_Номер\_паспорта', width=150, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_лица', text='ID лица')  
 self.tree.heading('ID\_иностранного\_гражданина', text='ID иностранного гражданина')  
 self.tree.heading('Дом', text='Дом')  
 self.tree.heading('Корпус', text='Корпус')  
 self.tree.heading('Квартира', text='Квартира')  
 self.tree.heading('Срок\_до', text='Срок до')  
 self.tree.heading('ID\_статуса\_лица', text='Статус лица')  
 self.tree.heading('ID\_улицы', text='Улица')  
 self.tree.heading('ID\_города', text='Город')  
 self.tree.heading('ID\_субъекта', text='Субъект')  
 self.tree.heading('К\_Номер\_паспорта', text='Номер паспорта')  
  
 self.tree.pack()  
  
 def open\_tableFinesOnFaces(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'ID\_оштрафованного\_лица', 'Размер\_штрафа', 'Статус\_штрафа', 'ID\_лица'), height=15, show='headings')  
  
 self.tree.column('ID\_оштрафованного\_лица', width=150, anchor=tk.CENTER)  
 self.tree.column('Размер\_штрафа', width=200, anchor=tk.CENTER)  
 self.tree.column('Статус\_штрафа', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_лица', width=130, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_оштрафованного\_лица', text='ID оштрафованного лица')  
 self.tree.heading('Размер\_штрафа', text='Размер штрафа')  
 self.tree.heading('Статус\_штрафа', text='Статус штрафа')  
 self.tree.heading('ID\_лица', text='ID лица')  
  
 self.tree.pack()  
  
 def open\_tableFinesOnSides(self):  
 self.tree = ttk.Treeview(self, columns=(  
 'ID\_оштрафованной\_стороны', 'Размер\_штрафа', 'Статус\_штрафа', 'ID\_принимающей\_стороны'), height=15, show='headings')  
  
 self.tree.column('ID\_оштрафованной\_стороны', width=170, anchor=tk.CENTER)  
 self.tree.column('Размер\_штрафа', width=200, anchor=tk.CENTER)  
 self.tree.column('Статус\_штрафа', width=150, anchor=tk.CENTER)  
 self.tree.column('ID\_принимающей\_стороны', width=170, anchor=tk.CENTER)  
  
 self.tree.heading('ID\_оштрафованной\_стороны', text='ID оштрафованной стороны')  
 self.tree.heading('Размер\_штрафа', text='Размер штрафа')  
 self.tree.heading('Статус\_штрафа', text='Статус штрафа')  
 self.tree.heading('ID\_принимающей\_стороны', text='ID принимающей стороны')  
  
 self.tree.pack()  
  
 def records(self, Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон):  
 self.db.insert\_data(Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон)  
 self.view\_records()  
  
 def edit\_record(self, Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон):  
 self.db.c.execute('''UPDATE иностранные\_граждане SET Фамилия=%s, Имя=%s, Отчество=%s,   
 ID\_гражданство=%s, Дата\_рождения=%s, Место\_рождения=%s, Пол=%s, Профессия=%s, Телефон=%s WHERE ID\_иностранного\_гражданина=%s''',  
 (Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол,  
 Профессия, Телефон, self.tree.set(self.tree.selection()[0], '#1')))  
 self.db.conn.commit()  
 self.view\_records()  
  
 def view\_records(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def delete\_records(self):  
 for selection\_item in self.tree.selection():  
 self.db.c.execute('''DELETE FROM иностранные\_граждане WHERE ID\_иностранного\_гражданина=%s''',  
 (self.tree.set(selection\_item, '#1'),))  
 self.db.conn.commit()  
 self.view\_records()  
  
 def search\_records(self, Искомое1, Искомое2, Искомое3, Искомое4, Искомое5, Искомое6, Искомое7,  
 Искомое8, Искомое9 ):  
 Искомое1 = ('%' + Искомое1 + '%',)  
 Искомое2 = ('%' + Искомое2 + '%',)  
 Искомое3 = ('%' + Искомое3 + '%',)  
 Искомое4 = ('%' + Искомое4 + '%',)  
 Искомое5 = ('%' + Искомое5 + '%',)  
 Искомое6 = ('%' + Искомое6 + '%',)  
 Искомое7 = ('%' + Искомое7 + '%',)  
 Искомое8 = ('%' + Искомое8 + '%',)  
 Искомое9 = ('%' + Искомое9 + '%',)  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане WHERE Фамилия LIKE %s OR Имя LIKE %s   
 OR Отчество LIKE %s OR ID\_гражданство LIKE %s OR Дата\_рождения LIKE %s OR Место\_рождения LIKE %s   
 OR Пол LIKE %s OR Профессия LIKE %s OR Телефон LIKE %s''',  
 (Искомое1, Искомое2, Искомое3, Искомое4, Искомое5, Искомое6, Искомое7, Искомое8, Искомое9))  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY ID\_иностранного\_гражданина ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record1(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Фамилия ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record2(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Имя ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record3(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Отчество ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record4(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY ID\_гражданство ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record5(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Дата\_рождения ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record6(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Место\_рождения ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record7(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Пол ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record8(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Профессия ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def sort\_record9(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане ORDER BY Телефон ASC''')  
 [self.tree.delete(i) for i in self.tree.get\_children()]  
 [self.tree.insert('', 'end', values=row) for row in self.db.c.fetchall()]  
  
 def open\_add\_dialog(self):  
 Add()  
  
 def open\_edit\_dialog(self):  
 Edit()  
  
 def open\_search\_dialog(self):  
 Search()  
  
 def close\_table(self):  
 self.tree.pack\_forget()  
  
  
  
class Add(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_(root)  
 self.init\_add()  
 self.view = app  
  
 def init\_add(self):  
 self.db = db  
 self.title('Добавить')  
 self.geometry('400x400+400+300')  
 self.resizable(False, False)  
  
 label\_Фамилия = tk.Label(self, text='Фамилия:')  
 label\_Фамилия.place(x=50, y=50)  
 label\_Имя = tk.Label(self, text='Имя:')  
 label\_Имя.place(x=50, y=80)  
 label\_patr = tk.Label(self, text='Отчество:')  
 label\_patr.place(x=50, y=110)  
 label\_select = tk.Label(self, text='Гражданство:')  
 label\_select.place(x=50, y=140)  
 label\_dofb = tk.Label(self, text='Дата рождения:')  
 label\_dofb.place(x=50, y=170)  
 label\_pofb = tk.Label(self, text='Место рождения:')  
 label\_pofb.place(x=50, y=200)  
 label\_Пол = tk.Label(self, text='Пол:')  
 label\_Пол.place(x=50, y = 230)  
 label\_prof = tk.Label(self, text='Профессия:')  
 label\_prof.place(x=50, y=260)  
 label\_Телефон = tk.Label(self, text='Телефон:')  
 label\_Телефон.place(x=50, y=290)  
  
  
 self.entry\_Фамилия = ttk.Entry(self)  
 self.entry\_Фамилия.place(x=200, y=50)  
  
 self.entry\_Имя = ttk.Entry(self)  
 self.entry\_Имя.place(x=200, y=80)  
  
 self.entry\_patr = ttk.Entry(self)  
 self.entry\_patr.place(x=200, y=110)  
  
 self.combobox = ttk.Combobox(self)  
 self.combobox.place(x=200, y=140)  
  
 self.db.c.execute('''SELECT `Наименование` FROM гражданства''')  
 self.db.conn.commit()  
 self.combobox['values'] = self.db.c.fetchall()  
 self.combobox.current(0)  
  
  
 self.entry\_dofb = ttk.Entry(self)  
 self.entry\_dofb.place(x=200, y=170)  
  
 self.entry\_pofb = ttk.Entry(self)  
 self.entry\_pofb.place(x=200, y=200, width=180)  
  
 self.entry\_Пол = ttk.Entry(self)  
 self.entry\_Пол.place(x=200, y=230)  
  
 self.entry\_prof = ttk.Entry(self)  
 self.entry\_prof.place(x=200, y=260)  
  
 self.entry\_Телефон = ttk.Entry(self)  
 self.entry\_Телефон.place(x=200, y =290)  
  
  
 btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)  
 btn\_cancel.place(x=300, y=350)  
  
 self.btn\_ok = ttk.Button(self, text='Добавить')  
 self.btn\_ok.place(x=220, y=350)  
 self.btn\_ok.bind('<Button-1>', lambda event: self.view.records(self.entry\_Фамилия.get(),  
 self.entry\_Имя.get(),  
 self.entry\_patr.get(),  
 self.combobox.get(),  
 self.entry\_dofb.get(),  
 self.entry\_pofb.get(),  
 self.entry\_Пол.get(),  
 self.entry\_prof.get(),  
 self.entry\_Телефон.get()))  
  
 self.grab\_set()  
 self.focus\_set()  
  
class Edit(Add):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_edit()  
 self.view = app  
 self.db = db  
 self.default\_data()  
  
 def init\_edit(self):  
 self.title('Редактировать')  
 btn\_edit = ttk.Button(self, text='Редактировать')  
 btn\_edit.place(x=205, y=350)  
 btn\_edit.bind('<Button-1>', lambda event: self.view.edit\_record(self.entry\_Фамилия.get(),  
 self.entry\_Имя.get(),  
 self.entry\_patr.get(),  
 self.combobox.get(),  
 self.entry\_dofb.get(),  
 self.entry\_pofb.get(),  
 self.entry\_Пол.get(),  
 self.entry\_prof.get(),  
 self.entry\_Телефон.get()))  
  
 self.btn\_ok.destroy()  
  
 def default\_data(self):  
 self.db.c.execute('''SELECT \* FROM иностранные\_граждане WHERE ID\_иностранного\_гражданина=%s''',  
 (self.view.tree.set(self.view.tree.selection()[0], '#1'),))  
 row = self.db.c.fetchone()  
 self.entry\_Фамилия.insert(0, row[1])  
 self.entry\_Имя.insert(0, row[2])  
 self.entry\_patr.insert(0, row[3])  
 self.combobox.current(0)  
 self.entry\_dofb.insert(0, row[5])  
 self.entry\_pofb.insert(0, row[6])  
 self.entry\_Пол.insert(0, row[7])  
 self.entry\_prof.insert(0, row[8])  
 self.entry\_Телефон.insert(0, row[9])  
  
class Search(tk.Toplevel):  
 def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.init\_search()  
 self.view = app  
  
 def init\_search(self):  
 self.title('Поиск')  
 self.geometry('300x100+400+300')  
 self.resizable(False, False)  
  
 label\_search = tk.Label(self, text='Поиск')  
 label\_search.place(x=50, y=20)  
  
 self.entry\_search = ttk.Entry(self)  
 self.entry\_search.place(x=105, y=20, width=150)  
  
  
 btn\_search = ttk.Button(self, text='Поиск')  
 btn\_search.place(x=105, y=50)  
 btn\_search.bind('<Button-1>', lambda event: self.view.search\_records(self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get(),  
 self.entry\_search.get()))  
 btn\_search.bind('<Button-1>', lambda event: self.destroy(), add='+')  
  
  
  
class DB:  
 def \_\_init\_\_(self):  
 self.conn = pymysql.connect(  
 host="localhost",  
 port=3308,  
 user="root",  
 database="migreg",  
 passwd=""  
 )  
  
 self.c = self.conn.cursor()  
 def insert\_data(self, Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон):  
 self.c.execute('''INSERT INTO иностранные\_граждане(Фамилия, Имя, Отчество, ID\_гражданство,   
 Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)''',  
 (Фамилия, Имя, Отчество, ID\_гражданство, Дата\_рождения, Место\_рождения, Пол, Профессия, Телефон))  
 self.conn.commit()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 db = DB()  
 app = Main(root)  
 app.pack()  
 root.title("Миграционный учет")  
 root.geometry("1260x500+100+200")  
  
  
  
 root.resizable(True, True)  
 root.mainloop()