Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра 307 «Цифровые технологии и информационные системы»

ОТЧЁТ

О выполнении лабораторной работы по предмету

«Базы данных»

«Создание системы, реализующей телефонный справочник»

|  |  |
| --- | --- |
|  | Выполнил:  Студент группы М3О-212Б-22  Кудряшев Д. А.  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  | Проверил:  Доцент, к.т.н.  Склеймин Ю.Б.  Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |

Москва 2023 г.

# Содержание

[Содержание 2](#_Toc153298872)

[Цель работы 3](#_Toc153298873)

[Постановка задачи и выбор средств реализации 4](#_Toc153298874)

[Схема реализации СУБД 5](#_Toc153298875)

[Руководство пользователя 7](#_Toc153298876)

[Тестирование 14](#_Toc153298877)

[Вывод 28](#_Toc153298878)

[Приложение А 29](#_Toc153298879)

# Цель работы

Создать систему, реализующую телефонную книгу, с использованием архитектуры клиент-сервер.

# Постановка задачи и выбор средств реализации

Необходимо реализовать серверную часть на PostgreSQL, для работы с базой данных использовалась программа pgAdmin. В качестве языка программирования, для создания архитектуры клиент-сервер, был выбран язык Python, в котором легко реализовать связь с базой данных с помощью библиотеки psycopg2 и создать приложение с помощью PyQt5. Также для упрощения работы по проектированию интерфейса воспользуемся QT Designer.

# Схема реализации СУБД

Структура базы данных телефонного справочника представлена на рисунке 1.

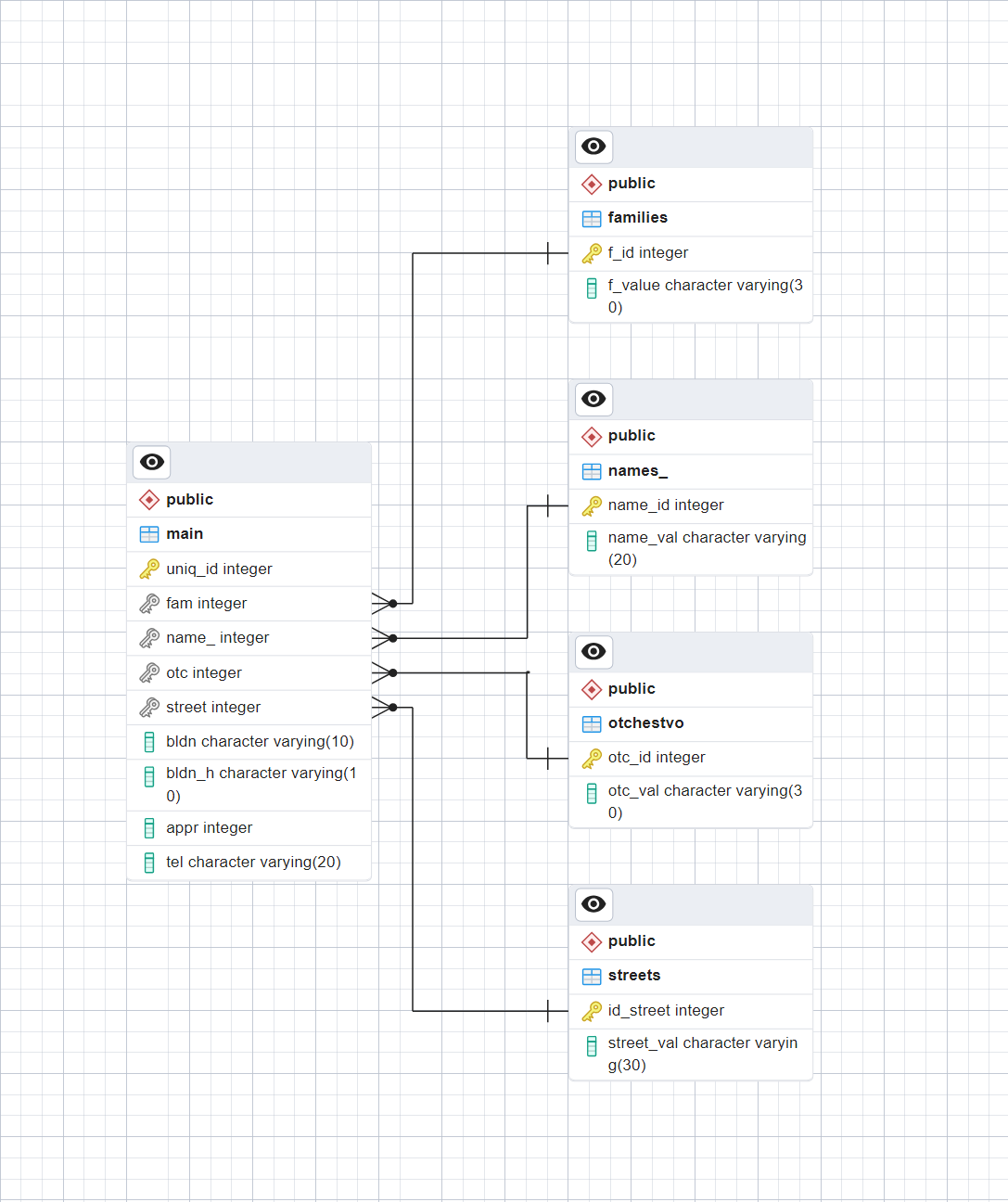


Рисунок — Схема телефонной книги

База данных состоит из таблиц: фамилии, имена, отчества, улицы и телефонная книга. Все таблицы связаны с телефонным справочником с помощью первичного ключа (столбец id). В самом справочнике хранится лишь телефон, номер дома, номер строения, номер квартиры.

# Руководство пользователя

При запуске программы, открывается главное окно, которое для удобства растягивается в любом направлении и закрывается нажатием на крестик в правом верхнем углу. вид главного окна представлен на рисунке 2.

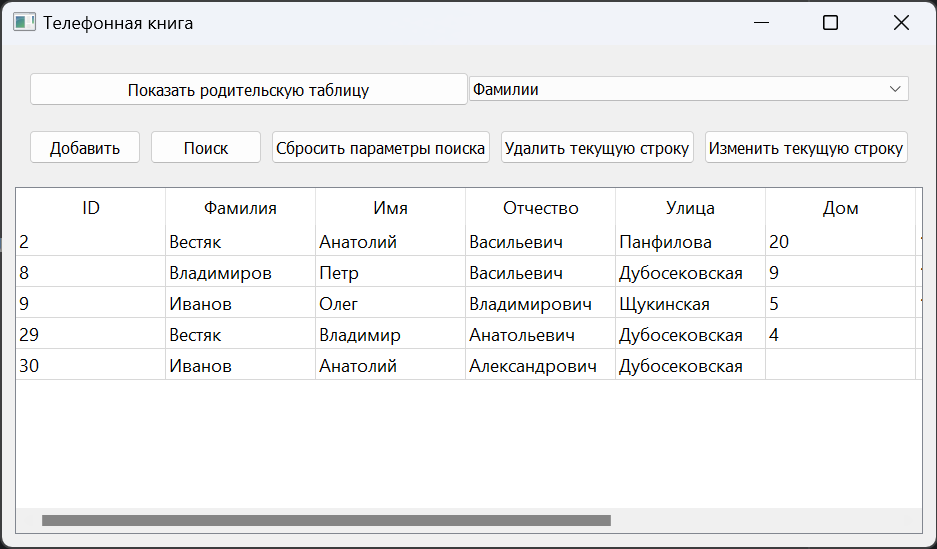


Рисунок — Вид главного окна

## Элементы взаимодействия с базой данных.

Первый элемент программы — выбор родительской таблицы (рис. 3).

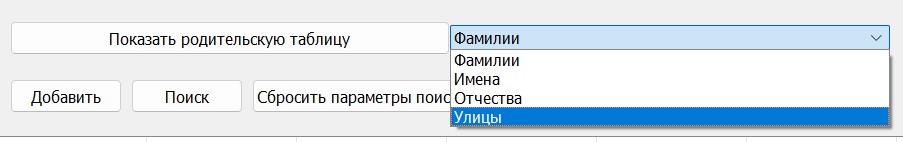


Рисунок — Первый элемент

Он состоит из поля со списком и кнопки. При нажатии на кнопку «Показать родительскую таблицу» открывается дополнительное окно с родительской таблицей, выбранной в поле справа, и доступным для работы с ней функционалом.

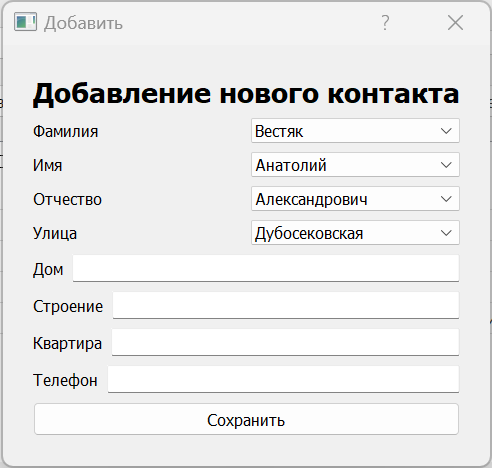
Второй элемент — панель инструментов для редактирования таблицы. (рис. 4).



Рисунок — Второй элемент

Кнопка «Добавить» вызывает открытие окна добавления элемента в таблицу main (рис. 5).

Рисунок — Окно добавления

Здесь в полях со списком доступны для выбора все имеющийся в соответствующих родительских таблицах фамилии, имена, отчества, улицы. Так же доступны поля со свободным текстовым вводом (при желании их можно оставить пустыми). При нажатии на кнопку «Сохранить» данное окно закроется и в таблице main появится новый элемент с введенными здесь значениями.

Кнопка «Поиск» открывает окно поиска (рис. 6).

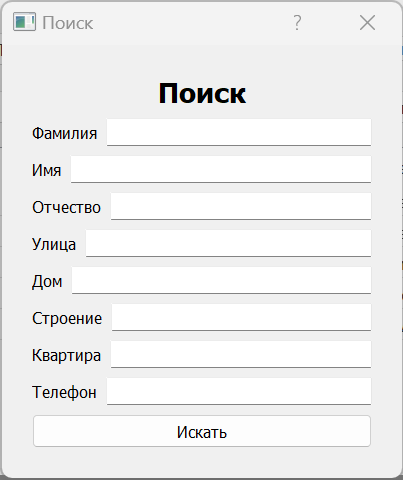
После ввода в доступные поля значений для поиска и нажатия кнопки «Искать» окно поиска и закроется и таблице отобразятся результаты поискового запроса (если по какому-либо параметру поиск не производится, это поле остается пустым). Результаты поискового запроса будут отображаться до тех пор, пока не будет нажата кнопка «Сбросить параметры поиска». Результаты также сбросятся при добавлении нового элемента в таблицу main, при изменении элемента в родительской таблице, при запуске нового поиска с другими значениями.

Рисунок — Окно поиска

Нажатие кнопки «Удалить текущую строку» срабатывает только при наличии выбранной строки в таблице. В этом случае, строка в таблице main, которая была выбрана перед нажатием, удаляется.

Нажатие кнопки «Изменить текущую строку» срабатывает также при наличии выбранной строки. При срабатывании откроется окно изменения, содержащее поля свободного ввода для дома, строения, квартиры и телефона, в которых отображаются первоначальные данные строки (рис. 7).

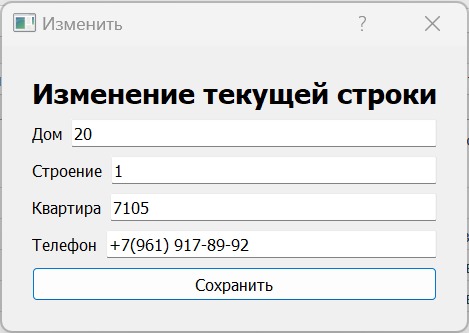
При нажатии кнопки «Сохранить» окно изменения закрывается, и выбранная ранее строка таблицы main обновляется согласно введенным ранее значениям (если не требуется изменение каких-либо элементов, строки ввода остаются теме же).

Рисунок — Окно изменения

Третий элемент главного окна — таблица, в которой отображается текущее состояние таблицы main (рис. 8).

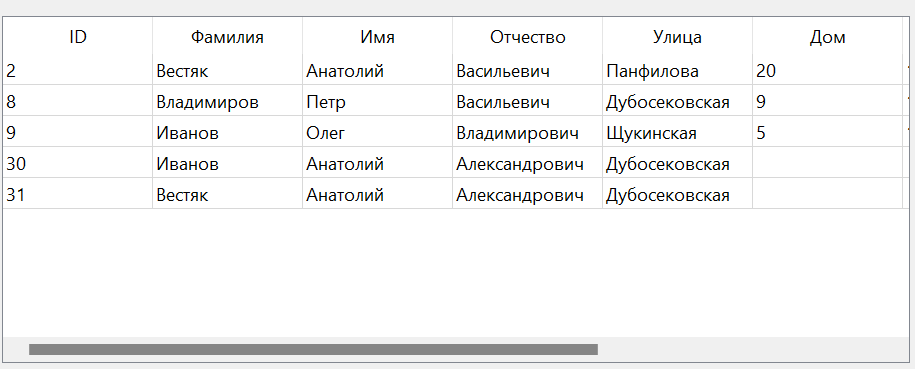


Рисунок — Третий элемент

При одиночном нажатии ЛКМ по ячейке таблицы, она подсвечивается (рис. 9), после чего становятся доступными вышеупомянутые функции «Удалить текущую строку» и «Изменить текущую строку».

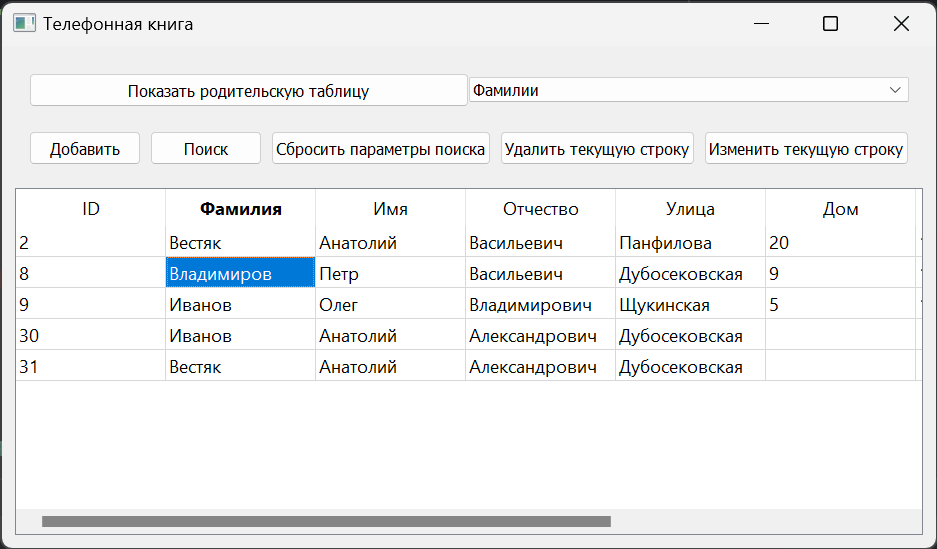


Рисунок — Окно с выделенным элементом

Вид упомянутого ранее окна родительских таблиц представлен на рисунке 10.

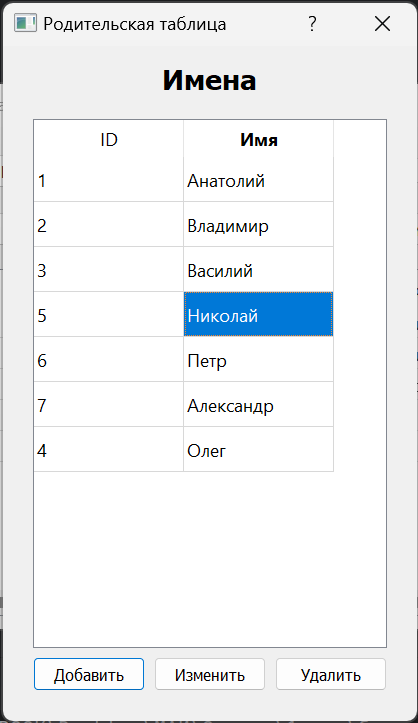
Данное окно имеет похожую структуру, а функционал кнопок аналогичен функционалу кнопок из главного окна за некоторыми исключениями: при попытке удалить элемент родительской таблицы, на который ссылается хотя бы одна строка таблицы main, всплывет предупредительное окно (рис. 11) и действие не произойдет; при попытке добавить копию уже имеющегося элемента (или изменить другой элемент на копирующий) всплывет предупредительное окно (рис. 12) и действие не произойдет.

Рисунок  — Вид окна родительской таблицы

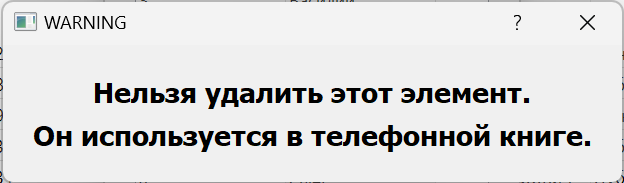


Рисунок — Предупредительное окно 1

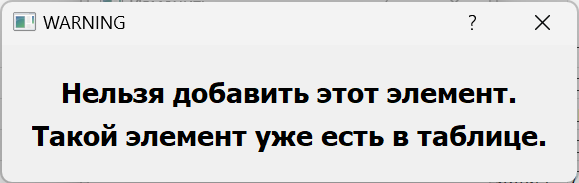


Рисунок — Предупредительное окно 2

# Тестирование

## Тест 1

Проверить добавление элемента в таблицу main.

Ожидаемый результат: В таблицу добавляется элемент (в том числе с пустыми вводными данными).

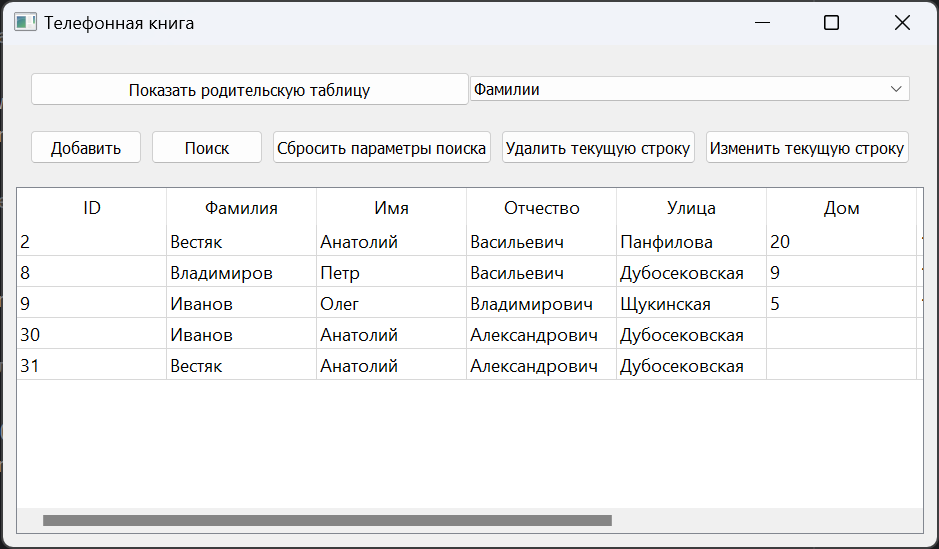


Рисунок — Таблица main до операции добавления

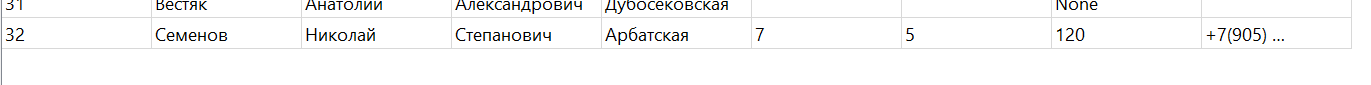
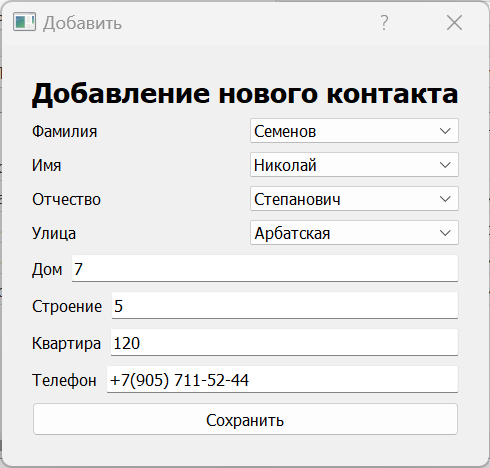


Рисунок — Введение данных в окне добавления

Рисунок — Результат нажатия на кнопку «Сохранить»

При добавлении элемента с пустыми строками для дома, строения, квартиры, телефона, результат получается таким же.

Вывод: Кнопка «Добавить» и окно добавления корректно выполняют свои функции.

## Тест 2

Проверить удаления элемента из таблицы main.

Ожидаемый результат: Элемент мгновенно удаляется из таблицы.

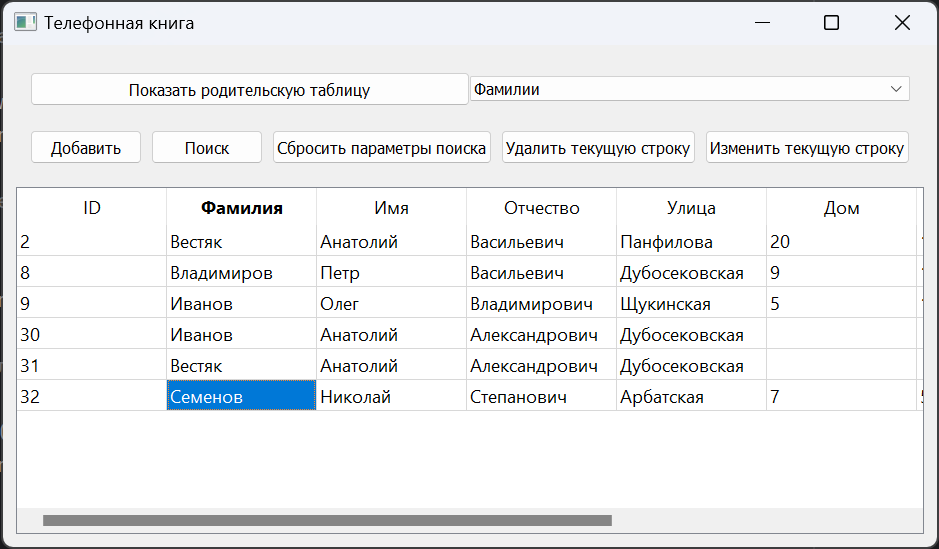


Рисунок — До нажатия кнопки «Удалить текущую строку»

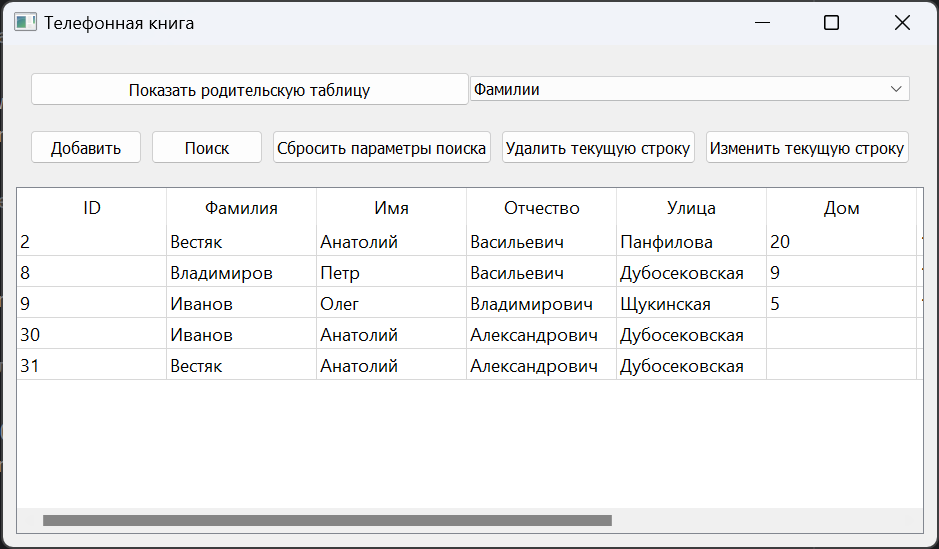


Рисунок — После нажатия кнопки «Удалить текущую строку»

После нажатия кнопки элемент мгновенно удалился из таблицы. При попытке удалить строку без выбранной ячейки ничего не происходит.

Вывод: Кнопка «Удалить текущую строку» работает корректно.

## Тест 3

Проверить изменение элемента в таблице main.

Ожидаемый результат: Элемент изменяется по одному/нескольким параметрам.

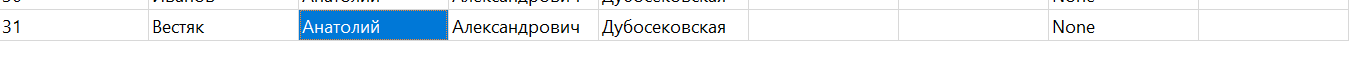


Рисунок — До изменения строки

Рисунок — Окно изменения

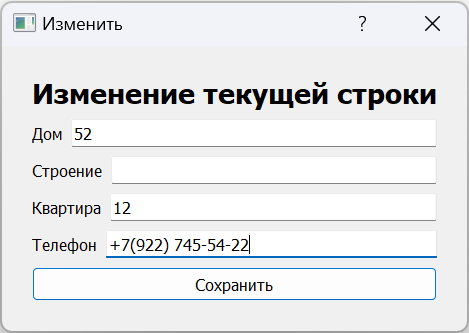




Рисунок — После изменения строки

Введенные данные сохранились в таблице. Те ячейки, которые не подвергались изменению, остались прежними.

Вывод: Кнопка «Изменить текущую строку» и окно изменения строки работают корректно.

## Тест 4

Проверить работу поиска элементов в таблице main.

Ожидаемый результат: Работает поиск по одному/нескольким критериям. Критерии можно изменить или сбросить. Таблицу можно редактировать по показанным результатам поиска.

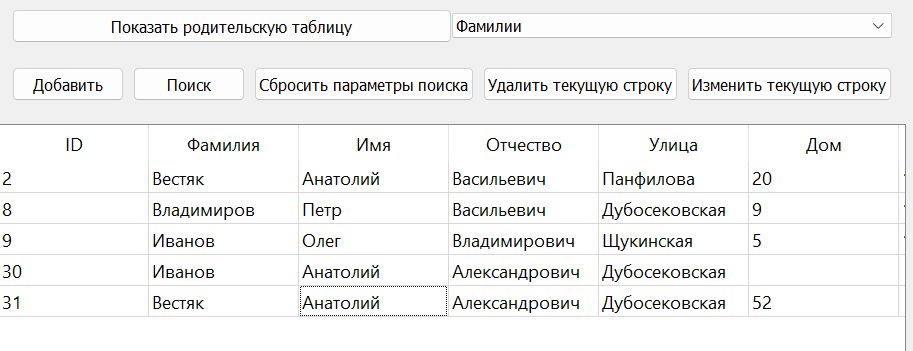
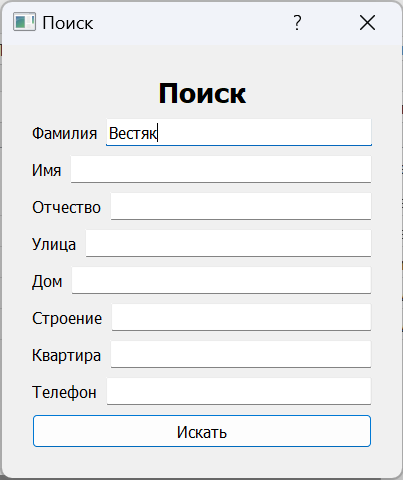


Рисунок — Таблица до поиска

Рисунок — Окно поиска



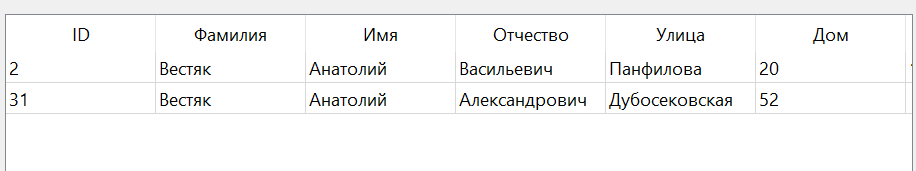


Рисунок — Результаты поиска

Попробуем дополнить поиск еще одним параметром.

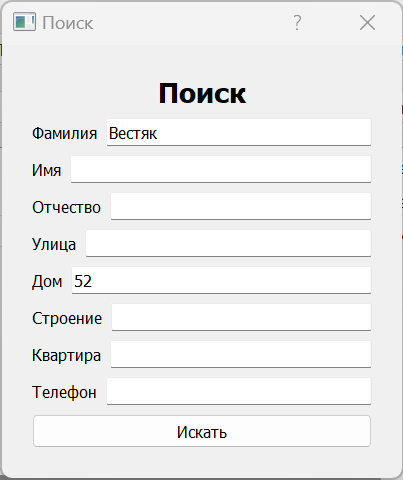


Рисунок — Дополнение поиска

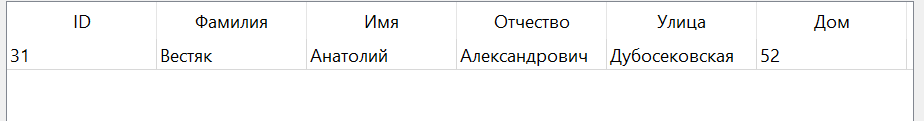


Рисунок — Результат поиска по двум параметрам

После нажатия кнопки «Сбросать параметры поиска» таблица возвращается в изначальное состояние. При работе с результатами поиска добавление элемента, удаление и изменения строки работают корректно.

Вывод: Кнопки «Поиск» и «Сбросить параметры поиска», а также окно поиска работают корректно.

## Тест 5

Проверить вывод родительских таблиц.

Ожидаемый результат: Выбранная таблица соответствует выводимой. Таблицы фамилий, имен, отчеств выводятся корректно.



Рисунок — Выбор родительской таблицы

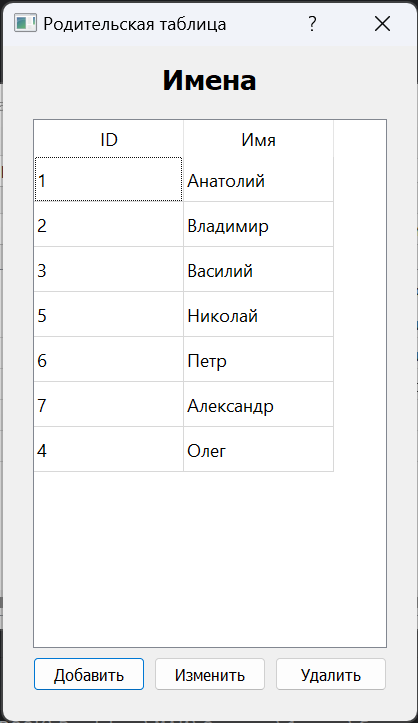
При выборе других таблиц вывод также происходит корректно.

Рисунок — Выбранная родительская таблица

Вывод: Кнопка «Показать родительскую таблицу» и поле выбора родительской таблицы работают корректно.

## Тест 6

Проверить добавление элемента в родительскую таблицу.

Ожидаемый результат: Добавляется элемент со введенным значением. При попытке добавить уже существующий элемент выводится диалоговое окно некорректного добавления элемента.

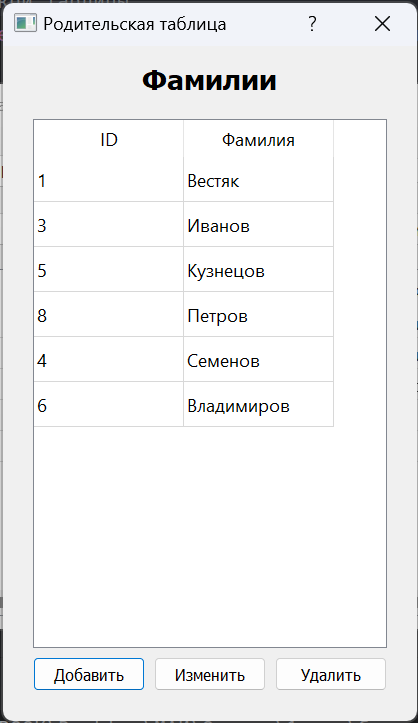
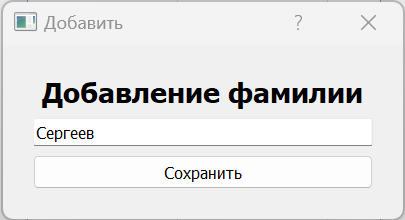


Рисунок — Окно добавления для родительской таблицы

Рисунок — Родительская таблица до изменения



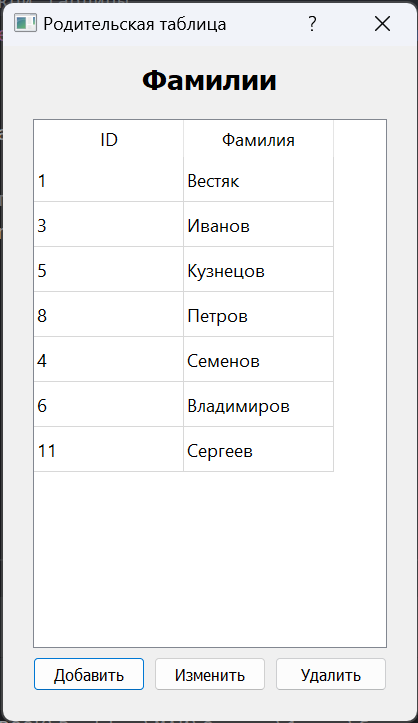


Рисунок — Родительская таблица после изменения

При попытке ввести тот же элемент повторно, всплывает предупреждающее диалоговое окно.

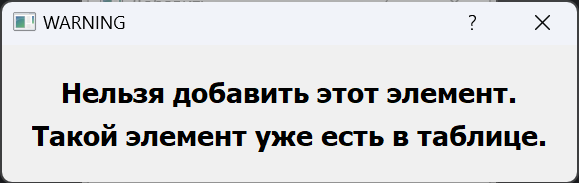
Аналогичные результаты получим при тех же действиях во всех остальных родительских таблицах.

Рисунок — Предупреждающее окно

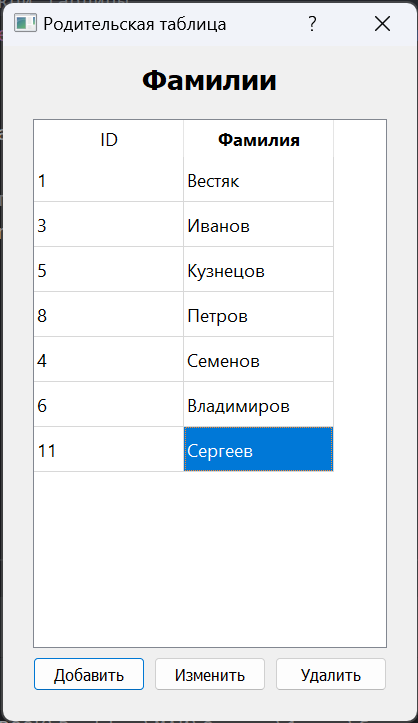
Вывод: Для каждой родительской таблицы корректно работает кнопка «Добавить», окно добавления элемента и предупреждающее окно.

## Тест 7

Проверить удаление элемента из родительской таблицы.

Ожидаемый результат: Удаляется выбранный элемент. При попытке удалить элемент, используемый в таблице main, выводится диалоговое окно некорректного удаления элемента.

Рисунок — Родительская таблица до удаления



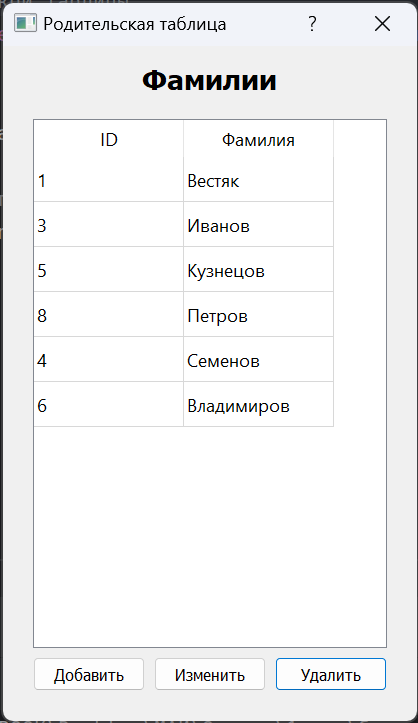


Рисунок — Родительская таблица после удаления

При попытке удаления элемента, используемого в таблице main, выводится диалоговое окно.

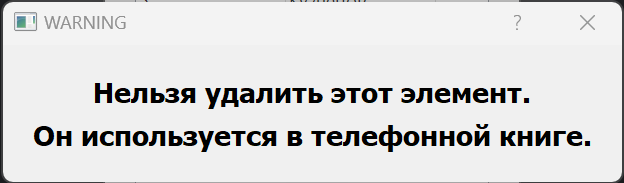


Рисунок — Предупреждающее окно

Аналогичные результаты получим при тех же действиях во всех остальных родительских таблицах.

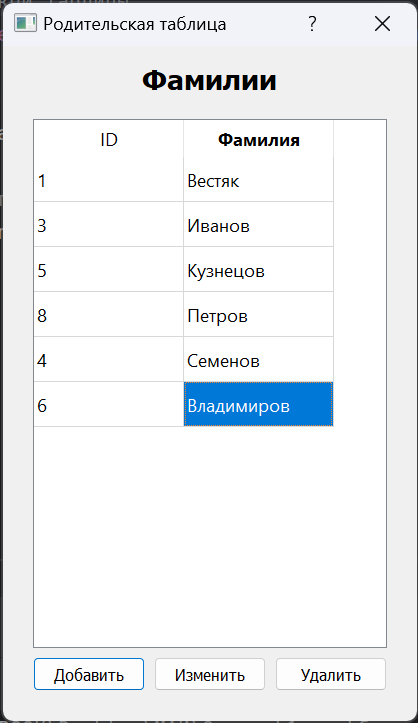
Вывод: Для каждой родительской таблицы корректно работает кнопка «Удалить» и предупреждающее окно.

## Тест 8

Проверить изменение элемента родительской таблицы.

Ожидаемый результат: Выбранный элемент изменяется в соответствии с введенным значением. При попытке изменить элемент на уже существующий выводится диалоговое окно некорректного изменения элемента.

Рисунок — Родительская таблица до изменения



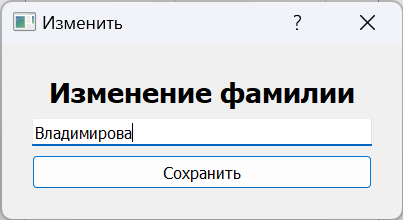
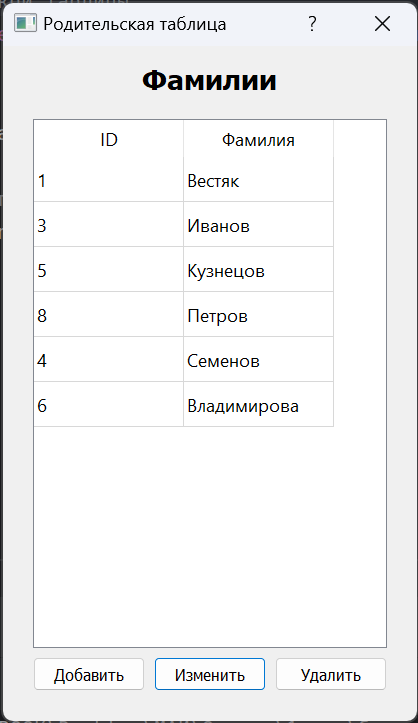


Рисунок — Родительская таблица после изменения

Рисунок — Окно изменения элемента

При изменении элемента, используемого в таблице main, его конечное значение также мгновенно отражается в дочерней таблице.

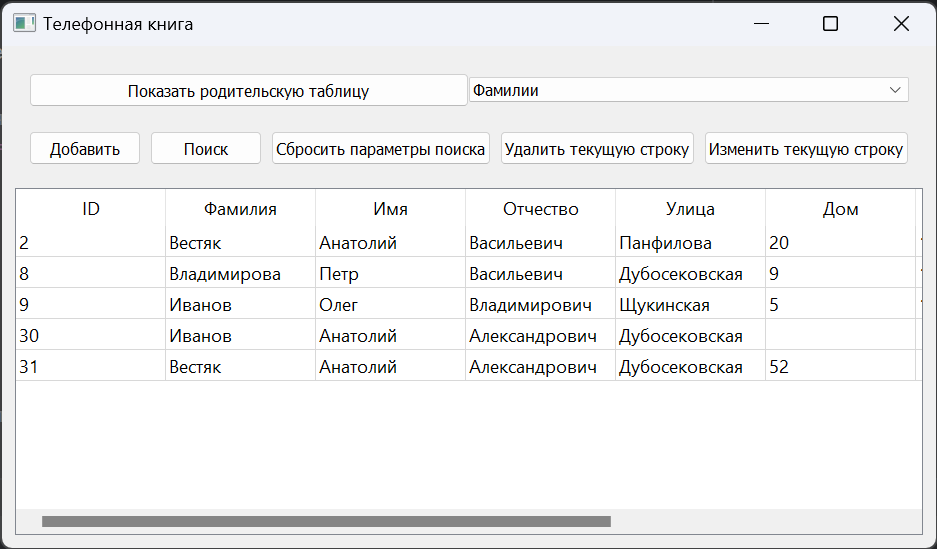


Рисунок — Результат изменения для таблицы main

Аналогичные результаты получим при тех же действиях во всех остальных родительских таблицах.

Вывод: Для каждой родительской таблицы корректно работает кнопка «Изменить» и окно изменения элемента родительской таблицы.

# Вывод

В результате выполнения лабораторной работы был изучен язык SQL и его применение для создания приложения с использованием архитектуры клиент-сервер, библиотекa psycopg2 для работы с PostgreSQL с помощью языка Python и фреймворк QT. Была создана база данных, реализующая телефонный справочник, а также программа клиент для связи с базой данных. Программа показала свою работоспособность в результате проведения тестирования.

# Приложение А

## main.py

import sys  
from PyQt5.QtWidgets import QApplication, QMainWindow, QDialog  
from PyQt5.QtWidgets import QTableWidgetItem  
from MainWindowDesign import Ui\_MainWindow  
from InsertWindowDesign import Ui\_InsertDialogWindow  
from ParentTableDialogWindowDesign import Ui\_ParentTableWindow  
from NoDeletableWindowDesign import Ui\_NoDeletableDialogWindow  
from InsertParentWindowDesign import Ui\_InsertParentDialogWindow  
from NoUniqueDialogWindow import Ui\_NoUniqueDialogWindow  
from SearchWindowDesign import Ui\_SearchDialogWindow  
from UpdateWindow import Ui\_UpdateWindow  
from UpdateParentWindowDesign import Ui\_UpdateParentDialogWindow  
from Connection import DataBase  
class PhoneBook(QMainWindow):  
 def \_\_init\_\_(self):  
 super(PhoneBook, self).\_\_init\_\_()  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
 self.insert\_window = QDialog()  
 self.ui\_insert\_dialog = Ui\_InsertDialogWindow()  
 self.ui\_insert\_dialog.setupUi(self.insert\_window)  
 self.search\_window = QDialog()  
 self.ui\_search\_dialog = Ui\_SearchDialogWindow()  
 self.ui\_search\_dialog.setupUi(self.search\_window)  
 self.update\_window = QDialog()  
 self.ui\_update\_dialog = Ui\_UpdateWindow()  
 self.ui\_update\_dialog.setupUi(self.update\_window)  
 self.parent\_window = QDialog()  
 self.ui\_parent\_dialog = Ui\_ParentTableWindow()  
 self.ui\_parent\_dialog.setupUi(self.parent\_window)  
 self.nodeletable\_window = QDialog()  
 self.ui\_nodeletable\_dialog = Ui\_NoDeletableDialogWindow()  
 self.ui\_nodeletable\_dialog.setupUi(self.nodeletable\_window)  
 self.insert\_parent\_window = QDialog()  
 self.ui\_insert\_parent\_dialog = Ui\_InsertParentDialogWindow()  
 self.ui\_insert\_parent\_dialog.setupUi(self.insert\_parent\_window)  
 self.update\_parent\_window = QDialog()  
 self.ui\_update\_parent\_dialog = Ui\_UpdateParentDialogWindow()  
 self.ui\_update\_parent\_dialog.setupUi(self.update\_parent\_window)  
 self.nounique\_window = QDialog()  
 self.ui\_nounique\_dialog = Ui\_NoUniqueDialogWindow()  
 self.ui\_nounique\_dialog.setupUi(self.nounique\_window)  
 self.database = DataBase()  
 self.view\_main\_data()  
 self.ui.InsertButton.clicked.connect(self.insert\_button\_clicked)  
 self.ui.DeleteButton.clicked.connect(self.delete\_button\_clicked)  
 self.ui.UpdateButton.clicked.connect(self.update\_button\_clicked)  
 self.ui.SearchButton.clicked.connect(self.search\_button\_clicked)  
 self.ui.StopSeacrhButton.clicked.connect(self.stop\_search\_button\_clicked)  
 self.ui.ShowParentTableButton.clicked.connect(self.show\_parent\_table\_window\_button\_clicked)  
 self.ui\_insert\_dialog.InsertSaveButton.clicked.connect(self.insert\_window\_save\_button\_clicked)  
 self.ui\_parent\_dialog.ParentTableDeleteButton.clicked.connect(self.parent\_window\_delete\_button\_clicked)  
 self.ui\_parent\_dialog.ParentTableInsertButton.clicked.connect(self.parent\_window\_insert\_button\_clicked)  
 self.ui\_parent\_dialog.ParentTableUpdateButton.clicked.connect(self.parent\_window\_update\_button\_clicked)  
 self.ui\_insert\_parent\_dialog.SaveParentButton.clicked.connect(self.insert\_parent\_window\_save\_button\_clicked)  
 self.ui\_update\_parent\_dialog.SaveParentButton.clicked.connect(self.update\_parent\_window\_save\_button\_clicked)  
 self.ui\_search\_dialog.SearchBeginButton.clicked.connect(self.search\_begin\_button\_clicked)  
 self.ui\_update\_dialog.SaveButton.clicked.connect(self.update\_window\_save\_button\_clicked)  
 self.current\_id = 0  
 self.updating\_row = -1  
 self.selected\_parent = "Фамилии"  
 self.current\_parent\_id = 0  
 self.updating\_parent\_row = -1  
 def view\_main\_data(self):  
 data = self.database.get\_full\_table()  
 rows\_number = 0  
 for row in data:  
 self.ui.DataTable.setRowCount(rows\_number + 1)  
 for i in range(len(row)):  
 item = QTableWidgetItem()  
 item.setText(str(row[i]))  
 self.ui.DataTable.setItem(rows\_number, i, item)  
 rows\_number += 1  
 def open\_insert\_window(self):  
 self.clear\_insert\_window()  
 self.fill\_insert\_window\_comboboxes()  
 self.insert\_window.show()  
 def insert\_window\_save\_button\_clicked(self):  
 new\_element = [self.ui\_insert\_dialog.FamilyComboBox.currentText(),  
 self.ui\_insert\_dialog.NameComboBox.currentText(),  
 self.ui\_insert\_dialog.OtcComboBox.currentText(),  
 self.ui\_insert\_dialog.StreetComboBox.currentText(),  
 self.ui\_insert\_dialog.BldnInput.text(),  
 self.ui\_insert\_dialog.Bldn\_hInput.text(),  
 self.ui\_insert\_dialog.ApprInput.text(),  
 self.ui\_insert\_dialog.PhoneInput.text()]  
 self.database.insert\_element\_to\_main(new\_element)  
 self.ui.DataTable.clearContents()  
 self.view\_main\_data()  
 self.insert\_window.close()  
 def insert\_button\_clicked(self):  
 self.open\_insert\_window()  
 def delete\_button\_clicked(self):  
 row = self.ui.DataTable.currentRow()  
 if row != -1:  
 item = self.ui.DataTable.item(row, 0)  
 self.database.delete\_element\_from\_main(item.text())  
 self.ui.DataTable.removeRow(row)  
 def show\_parent\_table\_window\_button\_clicked(self):  
 self.selected\_parent = self.ui.ShowParentComboBox.currentText()  
 parent = self.ui.ShowParentComboBox.currentText()  
 self.ui\_parent\_dialog.ParentTableWindowName.setText(parent)  
 self.view\_parent\_data(parent)  
 self.parent\_window.show()  
 def view\_parent\_data(self, parent):  
 item = QTableWidgetItem()  
 item.setText("ID")  
 self.ui\_parent\_dialog.ParentTable.setHorizontalHeaderItem(0, item)  
 item = QTableWidgetItem()  
 if parent == 'Фамилии':  
 data = self.database.get\_families\_table()  
 item.setText("Фамилия")  
 self.ui\_parent\_dialog.ParentTable.setHorizontalHeaderItem(1, item)  
 elif parent == 'Имена':  
 data = self.database.get\_names\_table()  
 item.setText("Имя")  
 self.ui\_parent\_dialog.ParentTable.setHorizontalHeaderItem(1, item)  
 elif parent == 'Отчества':  
 data = self.database.get\_otchestvo\_table()  
 item.setText("Отчество")  
 self.ui\_parent\_dialog.ParentTable.setHorizontalHeaderItem(1, item)  
 else:  
 data = self.database.get\_streets\_table()  
 item.setText("Улица")  
 self.ui\_parent\_dialog.ParentTable.setHorizontalHeaderItem(1, item)  
 rows\_number = 0  
 for row in data:  
 self.ui\_parent\_dialog.ParentTable.setRowCount(rows\_number + 1)  
 for i in range(len(row)):  
 item = QTableWidgetItem()  
 item.setText(str(row[i]))  
 self.ui\_parent\_dialog.ParentTable.setItem(rows\_number, i, item)  
 rows\_number += 1  
 def parent\_window\_delete\_button\_clicked(self):  
 parent = self.ui\_parent\_dialog.ParentTableWindowName.text()  
 row = self.ui\_parent\_dialog.ParentTable.currentRow()  
 if row != -1:  
 item = self.ui\_parent\_dialog.ParentTable.item(row, 1)  
 deletable = self.database.element\_deletable(item.text())  
 if deletable:  
 item = self.ui\_parent\_dialog.ParentTable.item(row, 0)  
 self.database.delete\_element\_from\_parent\_table(parent, item.text())  
 self.ui\_parent\_dialog.ParentTable.clearContents()  
 self.view\_parent\_data(parent)  
 else:  
 self.nodeletable\_window.show()  
 def parent\_window\_insert\_button\_clicked(self):  
 self.open\_insert\_parent\_window()  
 def fill\_update\_parent\_window(self):  
 if self.selected\_parent == "Фамилии":  
 self.ui\_update\_parent\_dialog.UpdateWindowName.setText("Изменение фамилии")  
 elif self.selected\_parent == "Имена":  
 self.ui\_update\_parent\_dialog.UpdateWindowName.setText("Изменение имени")  
 elif self.selected\_parent == "Отчества":  
 self.ui\_update\_parent\_dialog.UpdateWindowName.setText("Изменение отчества")  
 else:  
 self.ui\_update\_parent\_dialog.UpdateWindowName.setText("Изменение улицы")  
 item = self.ui\_parent\_dialog.ParentTable.item(self.updating\_parent\_row, 1)  
 self.ui\_update\_parent\_dialog.ParentInput.setText(item.text())  
 def parent\_window\_update\_button\_clicked(self):  
 self.updating\_parent\_row = self.ui\_parent\_dialog.ParentTable.currentRow()  
 if self.updating\_parent\_row != -1:  
 item = self.ui\_parent\_dialog.ParentTable.item(self.updating\_parent\_row, 0)  
 self.current\_parent\_id = int(item.text())  
  
 self.fill\_update\_parent\_window()  
 self.update\_parent\_window.open()  
 def update\_parent\_window\_save\_button\_clicked(self):  
 element = self.ui\_update\_parent\_dialog.ParentInput.text()  
 if self.database.element\_unique(self.selected\_parent, element):  
 self.database.update\_element\_in\_parent(self.selected\_parent, element, self.current\_parent\_id)  
 self.ui\_parent\_dialog.ParentTable.clearContents()  
 self.view\_parent\_data(self.selected\_parent)  
 self.update\_parent\_window.close()  
 self.ui.DataTable.clearContents()  
 self.view\_main\_data()  
 else:  
 self.nounique\_window.show()  
 def open\_insert\_parent\_window(self):  
 self.clear\_insert\_parent\_window()  
 if self.selected\_parent == "Фамилии":  
 self.ui\_insert\_parent\_dialog.InsertWindowName.setText("Добавление фамилии")  
 elif self.selected\_parent == "Имена":  
 self.ui\_insert\_parent\_dialog.InsertWindowName.setText("Добавление имени")  
 elif self.selected\_parent == "Отчества":  
 self.ui\_insert\_parent\_dialog.InsertWindowName.setText("Добавление отчества")  
 else:  
 self.ui\_insert\_parent\_dialog.InsertWindowName.setText("Добавление улицы")  
 self.insert\_parent\_window.show()  
 def insert\_parent\_window\_save\_button\_clicked(self):  
 element = self.ui\_insert\_parent\_dialog.ParentInput.text()  
 if self.database.element\_unique(self.selected\_parent, element):  
 self.database.insert\_element\_to\_parent\_table(self.selected\_parent, element)  
 self.ui\_parent\_dialog.ParentTable.clearContents()  
 self.view\_parent\_data(self.selected\_parent)  
 self.insert\_parent\_window.close()  
 else:  
 self.nounique\_window.show()  
 def fill\_insert\_window\_comboboxes(self):  
 data = self.database.get\_families()  
 for i in range(len(data)):  
 self.ui\_insert\_dialog.FamilyComboBox.addItem(data[i][0], userData=None)  
 data = self.database.get\_names()  
 for i in range(len(data)):  
 self.ui\_insert\_dialog.NameComboBox.addItem(data[i][0], userData=None)  
  
 data = self.database.get\_otchestvo()  
 for i in range(len(data)):  
 self.ui\_insert\_dialog.OtcComboBox.addItem(data[i][0], userData=None)  
 data = self.database.get\_streets()  
 for i in range(len(data)):  
 self.ui\_insert\_dialog.StreetComboBox.addItem(data[i][0], userData=None)  
 def clear\_insert\_window(self):  
 self.ui\_insert\_dialog.FamilyComboBox.clear()  
 self.ui\_insert\_dialog.NameComboBox.clear()  
 self.ui\_insert\_dialog.OtcComboBox.clear()  
 self.ui\_insert\_dialog.StreetComboBox.clear()  
 self.ui\_insert\_dialog.BldnInput.clear()  
 self.ui\_insert\_dialog.Bldn\_hInput.clear()  
 self.ui\_insert\_dialog.ApprInput.clear()  
 self.ui\_insert\_dialog.PhoneInput.clear()  
 def clear\_insert\_parent\_window(self):  
 self.ui\_insert\_parent\_dialog.ParentInput.clear()  
 def search\_button\_clicked(self):  
 self.open\_search\_window()  
 def open\_search\_window(self):  
 self.search\_window.open()  
 def clear\_search\_window(self):  
 self.ui\_search\_dialog.FamiyInput.clear()  
 self.ui\_search\_dialog.NameInput.clear()  
 self.ui\_search\_dialog.OtcInput.clear()  
 self.ui\_search\_dialog.StreetInput.clear()  
 self.ui\_search\_dialog.BldnInput.clear()  
 self.ui\_search\_dialog.Bldn\_hInput.clear()  
 self.ui\_search\_dialog.ApprInput.clear()  
 self.ui\_search\_dialog.PhoneInput.clear()  
 def view\_search\_data(self, data):  
 rows\_number = 0  
 for row in data:  
 self.ui.DataTable.setRowCount(rows\_number + 1)  
 for i in range(len(row)):  
 item = QTableWidgetItem()  
 item.setText(str(row[i]))  
 self.ui.DataTable.setItem(rows\_number, i, item)  
 rows\_number += 1  
  
 def search\_begin\_button\_clicked(self):  
 data = [self.ui\_search\_dialog.FamiyInput.text(),  
 self.ui\_search\_dialog.NameInput.text(),  
 self.ui\_search\_dialog.OtcInput.text(),  
 self.ui\_search\_dialog.StreetInput.text(),  
 self.ui\_search\_dialog.BldnInput.text(),  
 self.ui\_search\_dialog.Bldn\_hInput.text(),  
 self.ui\_search\_dialog.ApprInput.text(),  
 self.ui\_search\_dialog.PhoneInput.text()]  
 results = self.database.search\_elements\_in\_main(data)  
 self.ui.DataTable.clearContents()  
 self.view\_search\_data(results)  
 self.search\_window.close()  
 def stop\_search\_button\_clicked(self):  
 self.ui.DataTable.clearContents()  
 self.view\_main\_data()  
 def update\_button\_clicked(self):  
 self.updating\_row = self.ui.DataTable.currentRow()  
 if self.updating\_row != -1:  
 item = self.ui.DataTable.item(self.updating\_row, 0)  
 self.current\_id = int(item.text())  
 self.set\_elements\_to\_update(self.updating\_row)  
 self.update\_window.open()  
 def set\_elements\_to\_update(self, row):  
 item = self.ui.DataTable.item(row, 5)  
 self.ui\_update\_dialog.BldnInput.setText(item.text())  
 item = self.ui.DataTable.item(row, 6)  
 self.ui\_update\_dialog.Bldn\_hInput.setText(item.text())  
 item = self.ui.DataTable.item(row, 7)  
 self.ui\_update\_dialog.ApprInput.setText(item.text())  
 item = self.ui.DataTable.item(row, 8)  
 self.ui\_update\_dialog.PhoneInput.setText(item.text())  
 def update\_window\_save\_button\_clicked(self):  
 elements = [self.ui\_update\_dialog.BldnInput.text(),  
 self.ui\_update\_dialog.Bldn\_hInput.text(),  
 self.ui\_update\_dialog.ApprInput.text(),  
 self.ui\_update\_dialog.PhoneInput.text()]  
 self.database.update\_element\_in\_main(self.current\_id, elements)  
 item = self.ui.DataTable.item(self.updating\_row, 5)  
 item.setText(elements[0])  
 item = self.ui.DataTable.item(self.updating\_row, 6)  
 item.setText(elements[1])  
 item = self.ui.DataTable.item(self.updating\_row, 7)  
 item.setText(elements[2])  
 item = self.ui.DataTable.item(self.updating\_row, 8)  
 item.setText(elements[3])  
 self.update\_window.close()  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 window = PhoneBook()  
 window.show()  
 sys.exit(app.exec())

## Connection.py

import psycopg2  
DBNAME = 'db\_212'  
USER = 'postgres'  
PASSWORD = '7280'  
HOST = 'localhost'  
class DataBase:  
 def \_\_init\_\_(self):  
 self.connection = psycopg2.connect(dbname=DBNAME, user=USER, password=PASSWORD, host=HOST)  
 self.curs = self.connection.cursor()  
 def \_\_del\_\_(self):  
 self.connection.commit()  
 self.curs.close()  
 self.connection.close()  
 def get\_full\_table(self):  
 self.curs.execute(" select uniq\_id, f\_value, name\_val, otc\_val, street\_val, bldn, bldn\_h, appr, tel \  
 from main join families on main.fam=families.f\_id join names\_ on main.name\_=names\_.name\_id \  
 join otchestvo on main.otc=otchestvo.otc\_id join streets on main.street=streets.id\_street")  
 databasejoin = self.curs.fetchall()  
 return databasejoin  
 def get\_families\_table(self):  
 self.curs.execute("SELECT f\_id, f\_value FROM families")  
 databasefamilies = self.curs.fetchall()  
 return databasefamilies  
 def get\_names\_table(self):  
 self.curs.execute("SELECT name\_id, name\_val FROM names\_")  
 databasenames = self.curs.fetchall()  
 return databasenames  
 def get\_otchestvo\_table(self):  
 self.curs.execute("SELECT otc\_id, otc\_val FROM otchestvo")  
 databaseotc = self.curs.fetchall()  
 return databaseotc  
 def get\_streets\_table(self):  
 self.curs.execute("SELECT id\_street, street\_val FROM streets")  
 databasestreets = self.curs.fetchall()  
 return databasestreets  
 def insert\_element\_to\_main(self, new\_element\_data):  
 # Пример:  
 # self.curs.execute("INSERT INTO main VALUES(DEFAULT, 6, 6, 4, 1, '9', '1', 12, '+7(911) 208-56-00')")  
 data\_message = "INSERT INTO main VALUES(DEFAULT, "  
 data = self.get\_families\_table()  
 for rows in data:  
 if rows[1] == new\_element\_data[0]:  
 data\_message += str(rows[0]) + ", "  
 break  
 data = self.get\_names\_table()  
 for rows in data:  
 if rows[1] == new\_element\_data[1]:  
 data\_message += str(rows[0]) + ", "  
 break  
 data = self.get\_otchestvo\_table()  
 for rows in data:  
 if rows[1] == new\_element\_data[2]:  
 data\_message += str(rows[0]) + ", "  
 break  
 data = self.get\_streets\_table()  
 for rows in data:  
 if rows[1] == new\_element\_data[3]:  
 data\_message += str(rows[0]) + ", "  
 break  
 if new\_element\_data[6] == '': new\_element\_data[6] = 'NULL'  
 data\_message += "'" + new\_element\_data[4] + "', '" + \  
 new\_element\_data[5] + "', " + \  
 new\_element\_data[6] + ", '" + \  
 new\_element\_data[7] + "')"  
 self.curs.execute(data\_message)  
 def get\_families(self):  
 self.curs.execute("SELECT f\_value FROM families")  
 families = self.curs.fetchall()  
 return families  
  
 def get\_names(self):  
 self.curs.execute("SELECT name\_val FROM names\_")  
 names = self.curs.fetchall()  
 return names  
 def get\_otchestvo(self):  
 self.curs.execute("SELECT otc\_val FROM otchestvo")  
 otc = self.curs.fetchall()  
 return otc  
 def get\_streets(self):  
 self.curs.execute("SELECT street\_val FROM streets")  
 streets = self.curs.fetchall()  
 return streets  
 def delete\_element\_from\_main(self, uniq\_id):  
 self.curs.execute("DELETE FROM main WHERE uniq\_id = " + uniq\_id)  
 def element\_deletable(self, element):  
 database = self.get\_full\_table()  
 for rows in database:  
 for items in rows:  
 if element == items:  
 return False  
 return True  
 def delete\_element\_from\_parent\_table(self, parent, element\_id):  
 if parent == 'Фамилии':  
 self.curs.execute("DELETE FROM families WHERE f\_id = " + element\_id)  
 elif parent == 'Имена':  
 self.curs.execute("DELETE FROM names\_ WHERE name\_id = " + element\_id)  
 elif parent == 'Отчества':  
 self.curs.execute("DELETE FROM otchestvo WHERE otc\_id = " + element\_id)  
 else:  
 self.curs.execute("DELETE FROM streets WHERE id\_street = " + element\_id)  
 def insert\_element\_to\_parent\_table(self, parent, element):  
 if parent == 'Фамилии':  
 self.curs.execute("INSERT INTO families VALUES(DEFAULT, '" + element + "')")  
 elif parent == 'Имена':  
 self.curs.execute("INSERT INTO names\_ VALUES(DEFAULT, '" + element + "')")  
 elif parent == 'Отчества':  
 self.curs.execute("INSERT INTO otchestvo VALUES(DEFAULT, '" + element + "')")  
 else:  
 self.curs.execute("INSERT INTO streets VALUES(DEFAULT, '" + element + "')")  
 def element\_unique(self, parent, element):  
 if parent == 'Фамилии':  
 data = self.get\_families()  
 elif parent == 'Имена':  
 data = self.get\_names()  
 elif parent == 'Отчества':  
 data = self.get\_otchestvo()  
 else:  
 data = self.get\_streets()  
 for elements in data:  
 if element == elements[0]:  
 return False  
 return True  
 def search\_elements\_in\_main(self, parameters):  
 data\_message = " SELECT uniq\_id, f\_value, name\_val, otc\_val, street\_val, bldn, bldn\_h, appr, tel \  
 FROM main JOIN families ON main.fam=families.f\_id JOIN names\_ ON main.name\_=names\_.name\_id\  
 JOIN otchestvo ON main.otc=otchestvo.otc\_id JOIN streets ON main.street=streets.id\_street WHERE "  
 data\_message += "f\_value LIKE '%" + parameters[0] + "%' AND "  
 data\_message += "name\_val LIKE '%" + parameters[1] + "%' AND "  
 data\_message += "otc\_val LIKE '%" + parameters[2] + "%' AND "  
 data\_message += "street\_val LIKE '%" + parameters[3] + "%' AND "  
 data\_message += "bldn LIKE '%" + parameters[4] + "%' AND "  
 data\_message += "bldn\_h LIKE '%" + parameters[5] + "%' AND "  
 if parameters[6]:  
 data\_message += "appr = " + parameters[6] + " AND "  
 data\_message += "tel LIKE '%" + parameters[7] + "%'"  
 self.curs.execute(data\_message)  
 databasesearch = self.curs.fetchall()  
 return databasesearch  
 def update\_element\_in\_main(self, uniq\_id, elements):  
 data\_message = "UPDATE main SET bldn = '" + elements[0] + "', bldn\_h = '" + elements[1] + "', "  
 if elements[2] == "None" or elements[2] == "":  
 data\_message += "appr = NULL, "  
 else:  
 data\_message += "appr = " + elements[2] + ", "  
 data\_message += "tel = '" + elements[3] + "' WHERE uniq\_id = " + str(uniq\_id)  
 self.curs.execute(data\_message)  
 def update\_element\_in\_parent(self, parent, element, element\_id):  
 data\_message = ''  
 if parent == 'Фамилии':  
 data\_message += "UPDATE families SET f\_value = '" + element + "' WHERE f\_id = " + str(element\_id)  
 elif parent == 'Имена':  
 data\_message += "UPDATE names\_ SET name\_val = '" + element + "' WHERE name\_id = " + str(element\_id)  
 elif parent == 'Отчества':  
 data\_message += "UPDATE otchestvo SET otc\_val = '" + element + "' WHERE otc\_id = " + str(element\_id)  
 else:  
 data\_message += "UPDATE streets SET street\_val = '" + element + "' WHERE id\_street = " + str(element\_id)  
 self.curs.execute(data\_message)

## MainWindowDesign.py

from PyQt5 import QtCore, QtGui, QtWidgets  
class Ui\_MainWindow(object):  
 def setupUi(self, MainWindow):  
 MainWindow.setObjectName("MainWindow")  
 MainWindow.resize(688, 502)  
 self.centralwidget = QtWidgets.QWidget(MainWindow)  
 self.centralwidget.setObjectName("centralwidget")  
 self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.frame = QtWidgets.QFrame(self.centralwidget)  
 self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)  
 self.frame.setFrameShadow(QtWidgets.QFrame.Raised)  
 self.frame.setObjectName("frame")  
 self.horizontalLayout\_2 = QtWidgets.QHBoxLayout(self.frame)  
 self.horizontalLayout\_2.setContentsMargins(-1, -1, -1, 0)  
 self.horizontalLayout\_2.setSpacing(0)  
 self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")  
 self.ShowParentTableButton = QtWidgets.QPushButton(self.frame)  
 self.ShowParentTableButton.setObjectName("ShowParentTableButton")  
 self.horizontalLayout\_2.addWidget(self.ShowParentTableButton)  
 self.ShowParentComboBox = QtWidgets.QComboBox(self.frame)  
 self.ShowParentComboBox.setObjectName("ShowParentComboBox")  
 self.ShowParentComboBox.addItem("")  
 self.ShowParentComboBox.addItem("")  
 self.ShowParentComboBox.addItem("")  
 self.ShowParentComboBox.addItem("")  
 self.horizontalLayout\_2.addWidget(self.ShowParentComboBox)  
 self.verticalLayout.addWidget(self.frame)  
 self.frame\_2 = QtWidgets.QFrame(self.centralwidget)  
 self.frame\_2.setFrameShape(QtWidgets.QFrame.StyledPanel)  
 self.frame\_2.setFrameShadow(QtWidgets.QFrame.Raised)  
 self.frame\_2.setObjectName("frame\_2")  
 self.horizontalLayout = QtWidgets.QHBoxLayout(self.frame\_2)  
 self.horizontalLayout.setObjectName("horizontalLayout")  
 self.InsertButton = QtWidgets.QPushButton(self.frame\_2)  
 self.InsertButton.setObjectName("InsertButton")  
 self.horizontalLayout.addWidget(self.InsertButton)  
 self.SearchButton = QtWidgets.QPushButton(self.frame\_2)  
 self.SearchButton.setObjectName("SearchButton")  
 self.horizontalLayout.addWidget(self.SearchButton)  
 self.StopSeacrhButton = QtWidgets.QPushButton(self.frame\_2)  
 self.StopSeacrhButton.setObjectName("StopSeacrhButton")  
 self.horizontalLayout.addWidget(self.StopSeacrhButton)  
 self.DeleteButton = QtWidgets.QPushButton(self.frame\_2)  
 self.DeleteButton.setObjectName("DeleteButton")  
 self.horizontalLayout.addWidget(self.DeleteButton)  
 self.UpdateButton = QtWidgets.QPushButton(self.frame\_2)  
 self.UpdateButton.setObjectName("UpdateButton")  
 self.horizontalLayout.addWidget(self.UpdateButton)  
 self.verticalLayout.addWidget(self.frame\_2)  
 self.DataTable = QtWidgets.QTableWidget(self.centralwidget)  
 self.DataTable.setObjectName("DataTable")  
 self.DataTable.setColumnCount(9)  
 self.DataTable.setRowCount(0)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(2, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(3, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(4, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(5, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(6, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(7, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.DataTable.setHorizontalHeaderItem(8, item)  
 self.DataTable.horizontalHeader().setSortIndicatorShown(True)  
 self.DataTable.verticalHeader().setVisible(False)  
 self.DataTable.verticalHeader().setCascadingSectionResizes(False)  
 self.DataTable.verticalHeader().setDefaultSectionSize(30)  
 self.verticalLayout.addWidget(self.DataTable)  
 MainWindow.setCentralWidget(self.centralwidget)  
 self.retranslateUi(MainWindow)  
 QtCore.QMetaObject.connectSlotsByName(MainWindow)  
 def retranslateUi(self, MainWindow):  
 \_translate = QtCore.QCoreApplication.translate  
 MainWindow.setWindowTitle(\_translate("MainWindow", "Телефонная книга"))  
 self.ShowParentTableButton.setText(\_translate("MainWindow", "Показать родительскую таблицу"))  
 self.ShowParentComboBox.setItemText(0, \_translate("MainWindow", "Фамилии"))  
 self.ShowParentComboBox.setItemText(1, \_translate("MainWindow", "Имена"))  
 self.ShowParentComboBox.setItemText(2, \_translate("MainWindow", "Отчества"))  
 self.ShowParentComboBox.setItemText(3, \_translate("MainWindow", "Улицы"))  
 self.InsertButton.setText(\_translate("MainWindow", "Добавить"))  
 self.SearchButton.setText(\_translate("MainWindow", "Поиск"))  
 self.StopSeacrhButton.setText(\_translate("MainWindow", "Сбросить параметры поиска"))  
 self.DeleteButton.setText(\_translate("MainWindow", "Удалить текущую строку"))  
 self.UpdateButton.setText(\_translate("MainWindow", "Изменить текущую строку"))  
 self.DataTable.setSortingEnabled(False)  
 item = self.DataTable.horizontalHeaderItem(0)  
 item.setText(\_translate("MainWindow", "ID"))  
 item = self.DataTable.horizontalHeaderItem(1)  
 item.setText(\_translate("MainWindow", "Фамилия"))  
 item = self.DataTable.horizontalHeaderItem(2)  
 item.setText(\_translate("MainWindow", "Имя"))  
 item = self.DataTable.horizontalHeaderItem(3)  
 item.setText(\_translate("MainWindow", "Отчество"))  
 item = self.DataTable.horizontalHeaderItem(4)  
 item.setText(\_translate("MainWindow", "Улица"))  
 item = self.DataTable.horizontalHeaderItem(5)  
 item.setText(\_translate("MainWindow", "Дом"))  
 item = self.DataTable.horizontalHeaderItem(6)  
 item.setText(\_translate("MainWindow", "Строение"))  
 item = self.DataTable.horizontalHeaderItem(7)  
 item.setText(\_translate("MainWindow", "Квартира"))  
 item = self.DataTable.horizontalHeaderItem(8)  
 item.setText(\_translate("MainWindow", "Номер"))