**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Чувашский государственный университет имени И.Н. Ульянова»**

**(ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»)**

**Факультет информатики и вычислительной техники**

**Кафедра вычислительной техники**

ОТЧЕТ

О УЧЕБНОЙ ПРАКТИКЕ

(ОЗНАКОМИТЕЛЬНОЙ ПРАКТИКЕ)

на базе ***кафедры вычислительной техники факультета информатики и вычислительной техники ФГБОУ ВО «ЧГУ им. И.Н. Ульянова»***

|  |  |  |
| --- | --- | --- |
| Обучающийся 1 курса, направление подготовки  «Информатика и вычислительная техника», группа ИВТ-42-23 | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Кузнецов Д.А. |
|  | подпись, дата |  |
|  |  |  |
|  |  |  |
| Руководитель,  ст.преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Юрьева Е.В. |
|  |  |
|  |  |  |
| Руководитель от профильной организации, ассистент кафедры  вычислительной техники | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | Васильев Н.С. |
|  | подпись, дата |  |
|  |  |  |
| Заведующий кафедрой  вычислительной техники, доцент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ подпись, дата | Щипцова А.В. |
|  |  |
|  |  |  |
|  |  |  |

Чебоксары 2024

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 3](#_Toc178714055)

[1 ВВЕДЕНИЕ 4](#_Toc178714056)

[2 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 5](#_Toc178714057)

[2.1. Язык программирования Python 5](#_Toc178714058)

[2.2. Средство кроссплатформенной разработки PyQt 5](#_Toc178714059)

[2.3. Среда разработки PyCharm 6](#_Toc178714060)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 8](#_Toc178714061)

[3.1. Техническое задание 8](#_Toc178714062)

[3.2. Анализ поставленной задачи и выбор пути её решения 8](#_Toc178714063)

[3.3. База данных 9](#_Toc178714064)

[3.4. Главное окно 11](#_Toc178714065)

[3.4.1. Интерфейс главного окна 11](#_Toc178714066)

[3.4.2. Работа с файлом 12](#_Toc178714067)

[3.4.3. Изменение файла 16](#_Toc178714068)

[3.4.4. Проверка корректности данных 18](#_Toc178714069)

[3.4.5. Поиск данных 21](#_Toc178714070)

[4 ЗАКЛЮЧЕНИЕ 22](#_Toc178714071)

[5 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 23](#_Toc178714072)

[6 ПРИЛОЖЕНИЕ А 24](#_Toc178714073)

[Исходный код программы (основные функции) 24](#_Toc178714074)

# ВВЕДЕНИЕ

Цель практики:

* получение профессиональных умений и опыта технологической (проектно-технологической) деятельности;
* закрепление, расширение и углубление теоретических и практических знаний умений и навыков, полученных ранее при изучении дисциплин учебного плана.

Задачи практики:

* ознакомление с базой практики (профильной организацией), выпускаемой продукцией, используемыми программными средствами для решения практических задач;
* изучение технологии создания программных средств вычислительной техники и автоматизированных систем: PyQt – набор инструментов набор инструментов для создания виджетов с графическим интерфейсом для языка программирования Python;
* приобретение и закрепление навыков разработки алгоритмов и программ: проектирование приложений с помощью средств библиотеки PyQt, построение и управление свойствами виджетов, установка и применение библиотек для реализации основного функционала, а также управления базой данных и обработки ошибок.
* приобретение навыков разработки и оформления отчётности и (или) программной документации/документации пользователя;
* оформление отчёта по практике в соответствии с рекомендациями п.п. 6,7 программы практики.

Планируемый результат: десктопное приложение для работы с базой данных телефонных переговоров, включающей в себя записи об абонентах, городах и переговорах. В приложении будут реализованы функции добавления, вставки, редактирования, удаления, а также поиск записей. Для ввода и вывода данных будут использоваться файлы СУБД.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Язык программирования Python

Python — это скриптовый язык программирования. Он универсален, поэтому подходит для решения разнообразных задач и для многих платформ: начиная с iOS и Android и заканчивая серверными операционными системами.

Python – это интерпретируемый язык, а не компилируемый, как C++ или Java. Программа на Python представляет собой обычный текстовый файл. Код можно писать практически в любом редакторе или использовать специальные IDE.

Преимущества языка Python:

* Простота синтаксиса — низкий порог вхождения.
* Расширяемость и гибкость — Python можно легко расширить для взаимодействия с другими программными системами или встроить в программы в качестве компонента.
* Интерпретируемость и кроссплатформенность — интерпретатор Python есть для всех популярных платформ.
* Стандартизированность — у Python есть единый стандарт для написания кода.
* Open Source — у интерпретатора Python открытый код.
* Сильное комьюнити и конференции — вокруг Python образовалось дружественное комьюнити.
* Широта применения — Python используется в web-разработке, работе с данными, автоматизации бизнес-процессов и геймдеве.
* Востребованность на рынке труда и поддержка гигантами IT-сферы.

Основные принципы работы Python:

* Интерпретируемость. Позволяет выполнять код построчно без предварительной компиляции.
* Динамическая типизация. Позволяет не указывать типы данных в коде, что делает его более гибким и удобным для работы.
* Простота и чистота кода. Чем проще и понятнее написан код, тем легче его поддерживать и модифицировать в будущем.
* Кроссплатформенность. Python поддерживает работу на различных операционных системах, таких как Windows, macOS, Linux.
* Переменные и типы данных. Каждая переменная в программе имеет свое уникальное имя и может содержать различные типы данных.
* Структуры данных и операторы. Основа любой программы — это структуры данных, которые позволяют хранить информацию различных типов и структур.
* Функциональные возможности, библиотеки и фреймворки. Функции и модули обеспечивают разнообразные методы организации и структурирования кода.

## Средство кроссплатформенной разработки PyQt

PyQt — это библиотека для языка программирования Python, которая позволяет создавать графические интерфейсы для приложений. Она основана на библиотеке Qt, которая изначально была написана на C++. PyQt позволяет использовать все функциональные возможности Qt в Python-проектах.

Преимущества PyQt:

* Мощные возможности: PyQt предоставляет доступ ко всем функциям библиотеки Qt, включая графический интерфейс, работу с сетью, базами данных и многое другое.
* Кроссплатформенность: приложения, созданные с помощью PyQt, могут работать на различных операционных системах, таких как Windows, macOS и Linux.
* Большое сообщество: PyQt имеет активное сообщество разработчиков, которые создают библиотеки и примеры кода для различных задач.
* Простота использования: PyQt предоставляет удобные инструменты для создания интерфейсов, такие как QMainWindow, QLabel, QPushButton и другие.
* Совместимость с Python: PyQt хорошо интегрируется с Python и позволяет использовать все его возможности.

Принципы работы PyQt:

* Создание окна: сначала необходимо создать окно приложения с помощью класса QMainWindow.
* Добавление элементов интерфейса: затем можно добавить различные элементы интерфейса, такие как кнопки, текстовые поля, изображения и т. д.
* Обработка событий: при нажатии на кнопки или изменении значений текстовых полей необходимо обрабатывать соответствующие события.
* Взаимодействие с пользователем: приложение может взаимодействовать с пользователем, отображая информацию, запрашивая данные и т. п.
* Работа с данными: PyQt позволяет работать с данными, такими как файлы, базы данных и сети.
* Отображение графики: с помощью PyQt можно отображать графику, такую как изображения, видео и анимации.

PyQt — это мощная библиотека для создания графических интерфейсов на Python. Она предоставляет широкие возможности для разработки приложений, которые могут работать на различных операционных системах.

## Среда разработки PyCharm

PyCharm — это интегрированная среда для написания, редактирования и запуска кода на языке Python. Она предоставляет программистам множество инструментов для разработки, упрощает написание и отладку кода, а также управление проектами.

Преимущества PyCharm:

* Поддержка Python. PyCharm ориентирована на язык Python, что означает наличие множества функций и инструментов, специально разработанных для Python-разработчиков.
* Мощные инструменты разработки. Среда предлагает широкий спектр инструментов для разработки, включая редактор кода с подсветкой синтаксиса, функцией автодополнения и другими полезными возможностями. Это упрощает процесс написания кода и улучшает его качество.
* Интеграция с виртуальными средами. PyCharm позволяет создавать и управлять виртуальными средами. Это изолирует проекты друг от друга и помогает избежать конфликтов зависимостей.
* Поддержка отладки. Среда предоставляет мощные инструменты для отладки кода, включая пошаговое выполнение, точки останова и просмотр значений переменных. Это помогает выявлять и исправлять ошибки в коде.
* Интеграция с системами контроля версий. PyCharm поддерживает популярные системы контроля версий, такие как Git и SVN. Это позволяет эффективно управлять изменениями в коде.
* Поддержка фреймворков и библиотек. Среда поддерживает множество популярных фреймворков и библиотек, используемых в разработке на Python. Это упрощает их использование и интеграцию в проекты.

Принципы работы PyCharm:

* Удобная навигация по коду. PyCharm предоставляет удобные инструменты для навигации по коду, включая поиск и замену, переход к определению и другие функции. Это помогает быстро ориентироваться в проекте.
* Рефакторинг кода. Среда предлагает множество функций для рефакторинга кода, включая переименование переменных, функций и классов, удаление дубликатов и другие операции. Это улучшает структуру и читаемость кода.
* Интеграция с другими инструментами. PyCharm может интегрироваться с другими инструментами разработки, такими как GitKraken, GitHub и другими. Это расширяет функциональность среды и повышает удобство её использования.

# ПРАКТИЧЕСКАЯ ЧАСТЬ

## Техническое задание

Разработать программу для работы с базой данных. Необходимо обеспечить следующую функциональность:

* ввод данных в таблицы;
* чтение данных из файлов;
* запись данных в файлы;
* вывод данных на экран;
* добавление, изменение, удаление данных;
* поиск информации.

Разработать интерактивное меню, позволяющее пользователю выбрать функцию работы с базой данных. Диалоговое меню должно обязательно содержать пункты, указанные выше.

Программа должна корректно работать с любыми исходными данными (реализовать проверку вводимых пользователем данных). Типы полей в базе данных должны соответствовать назначению поля, например, тип поля «Дата договора» – структура дата/время, тип поля «Серия паспорта» - число и т.д.

Вы работаете в коммерческой службе телефонной компании. Компания предоставляет абонентам телефонные линии для междугородних переговоров. Вашей задачей является отслеживание стоимости междугородних телефонных переговоров.

Абонентами компании являются юридические лица, имеющие телефонную точку, ИНН, расчётный счёт в банке. Стоимость переговоров зависит от города, в который осуществляется звонок, и времени суток (день, ночь). Каждый звонок абонента автоматически фиксируется в базе данных. При этом запоминаются город, дата, длительность разговора и время суток.

***Таблицы***

Абоненты (Код абонента, Номер телефона, ИНН, Адрес).

Города (Код города, Название, Тариф дневной, Тариф ночной).

Переговоры (Код переговоров, Код абонента, Код города, Дата, Количество минут, Время суток).

## Анализ поставленной задачи и выбор пути её решения

На данном этапе был проведён анализ поставленного технического задания, результатом которого стал следующий план работ:

1. Разработка интерфейса главного окна приложения. За образец был взят интерфейс программы для редактора таблиц MS Excel (рис. 1).
2. Реализация функционала: вывод таблицы на экран, открытие таблицы из файла базы данных, сохранение таблицы в файл базы данных.
3. Реализация функционала: добавление, изменение и удаление данных.
4. Разработка функционала: поиск данных.

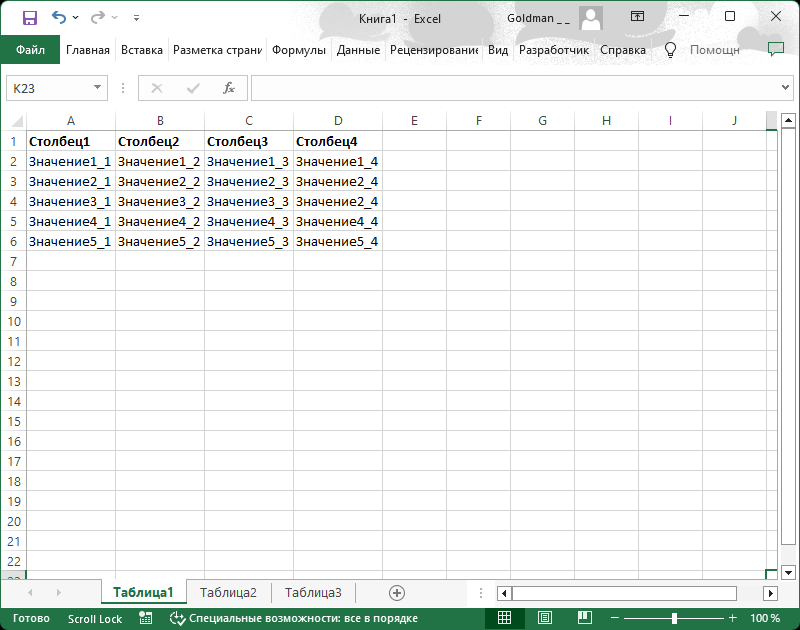


Рисунок 1. Интерфейс MS Excel

## База данных

База данных содержит 3 таблицы (рис. 2): *абоненты* (callers), *города* (cities) и *переговоры* (conversations).

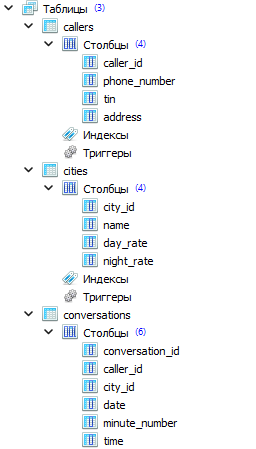


Рисунок 2. Структура базы данных

Таблица *абоненты* содержит 4 поля: *код абонента* (caller\_id), *номер телефона* (phone\_number), *ИНН* (tin), *адрес* (address). Поле caller\_idявляется первичным ключом (PRIMARY KEY) и представляется целым числом (INTEGER). Поле phone\_number представляется целым числом (INTEGER). Поле tin представляется текстом (TEXT), так как ИНН может начинаться с 0. Поле address представляется текстом (TEXT).

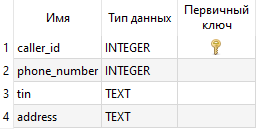


Рисунок 3. Таблица *абоненты*

Таблица *города* содержит 4 поля: *код города* (city\_id), *название* (name), *тариф дневной* (day\_rate), *тариф ночной* (night\_rate). Поле city\_idявляется первичным ключом (PRIMARY KEY) и представляется целым числом (INTEGER). Поле name представляется текстом (TEXT). Поле day\_rate представляется целым числом (INTEGER). Поле night\_rate представляется целым числом (INTEGER).

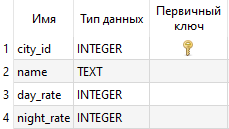


Рисунок 4. Таблица *города*

Таблица *города* содержит 6 полей: *код переговоров* (conversation\_id), *код абонента* (caller\_id), *код города* (city\_id), *дата* (date), *количество минут* (minute\_number), *время суток* (time). Поле conversation\_idявляется первичным ключом (PRIMARY KEY) и представляется целым числом (INTEGER). Поле caller\_idявляется внешним ключом (FOREIGN KEY) и представляется целым числом (INTEGER). Поле city\_idявляется внешним ключом (FOREIGN KEY) и представляется целым числом (INTEGER). Поле date представляется текстом (TEXT), так как содержит знаки “.”. Поле minute\_number представляется целым числом (INTEGER). Поле time представляется текстом (TEXT), так как может принимать два значения: *день* или *ночь*.

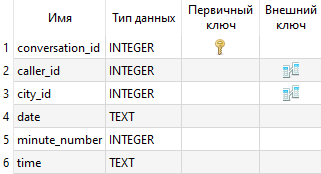


Рисунок 5. Таблица *переговоры*

## Главное окно

### Интерфейс главного окна

Открывая приложение, пользователь видит перед собой пустую таблицу «Абоненты» (рис. 6).

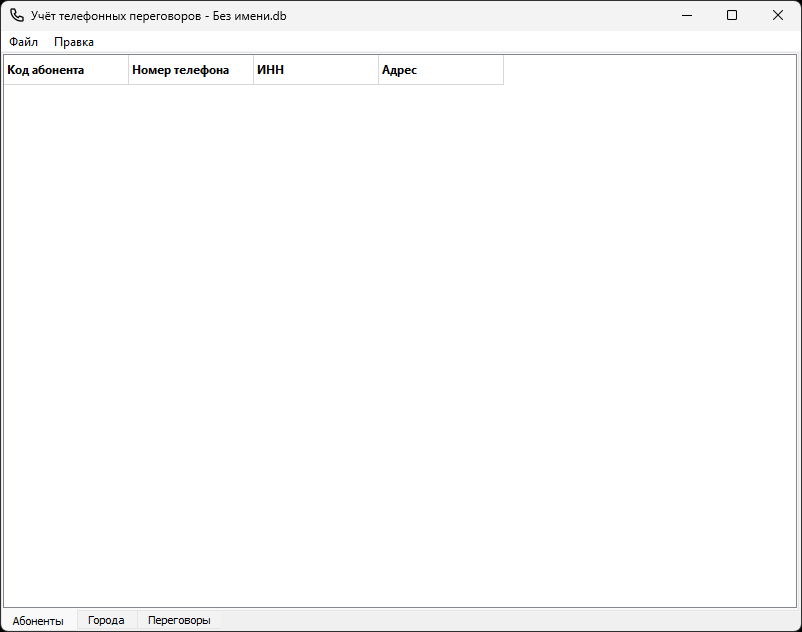


Рисунок 6. Пустая таблица *абоненты*

Для того чтобы открыть другую таблицу (рис. 7, 8), необходимо выбрать её среди вкладок нижней части экрана: *Абоненты*, *Города*, *Переговоры*.

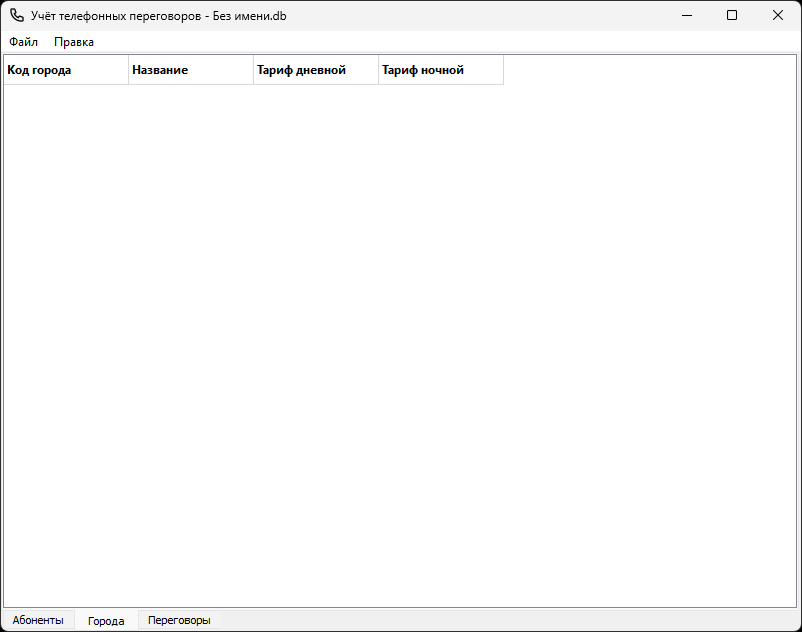


Рисунок 7. Пустая таблица *Города*

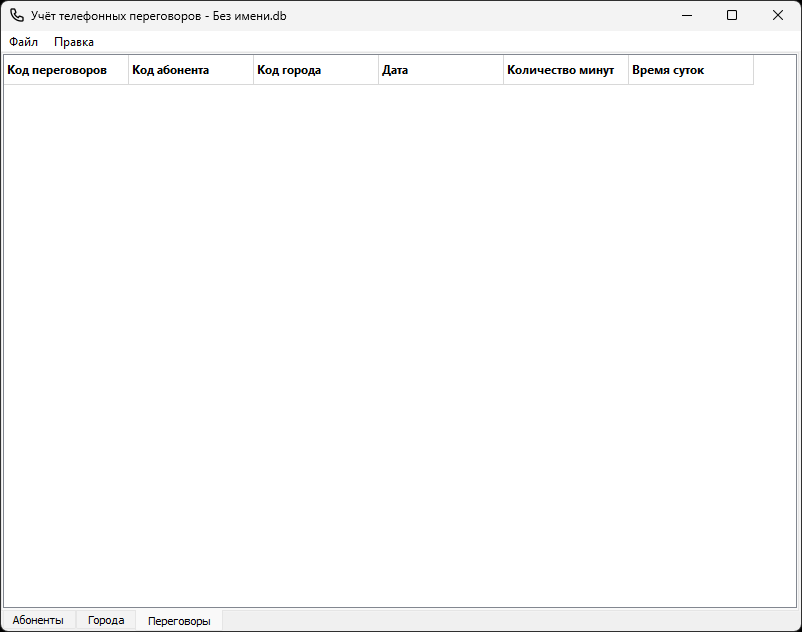


Рисунок 8. Пустая таблица *Переговоры*

### Работа с файлом

Для того чтобы создать новый файл, необходимо выбрать пункт *Файл* верхнего меню, а затем выбрать пункт *Новый* (рис. 9). Для того чтобы открыть существующий файл, необходимо выбрать пункт *Файл* верхнего меню, а затем выбрать пункт *Открыть* (рис. 9).

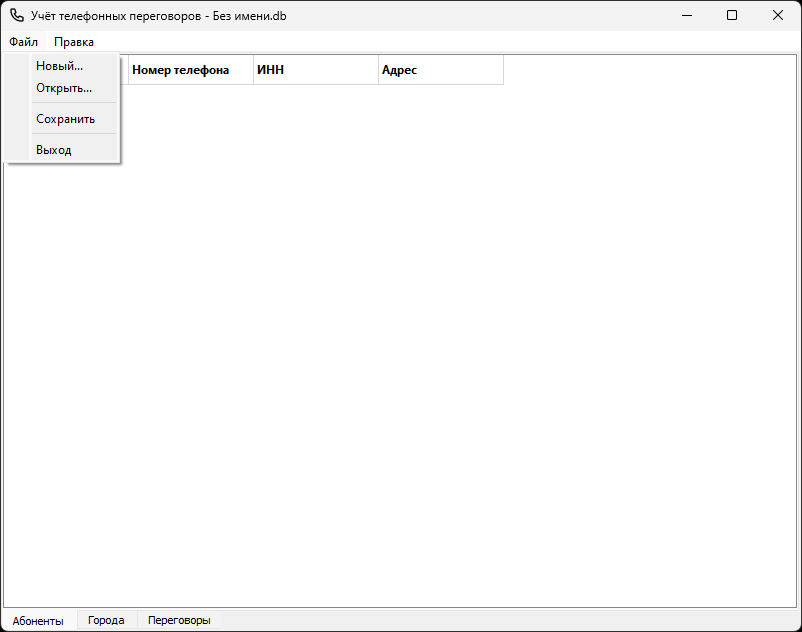


Рисунок 9. Пункт *Файл* верхнего меню

Затем открывается диалоговое окно, где пользователь может выбрать файл базы данных с расширением .db (рис. 10).

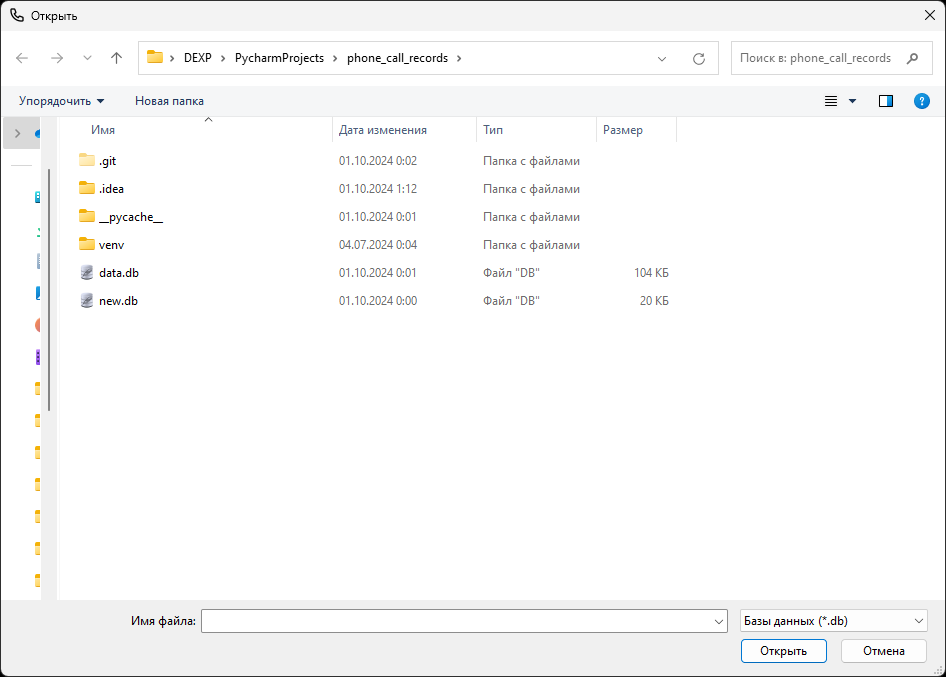


Рисунок 10. Диалоговое окно выбора файла

При открытии файла отправляются запросы в базу данных:

SELECT \* FROM callers (для таблицы *Абоненты*)

SELECT \* FROM cities (для таблицы *Города*)

SELECT \* FROM conversations (для таблицы *Переговоры*)

Пример полученных данных:

Таблица *Абоненты*

[  
(1, 89761729535, 573484469643, Елец),  
(2, 89192990338, 230214223686, Нижний Тагил), …

]

Таблица *Города*

[  
(1, Абакан, 2, Елец),  
(2, Ангарск, 15, 20), …

]

Таблица *Переговоры*

[  
(1, 301, 10, 02.09.2018, 135, день),  
(2, 124, 58, 19.01.2009, 40, ночь), …

]

Затем данные копируются в таблицы главного окна: *Абоненты* (рис. 11), *Города* (рис. 12) и *Переговоры* (рис. 13).

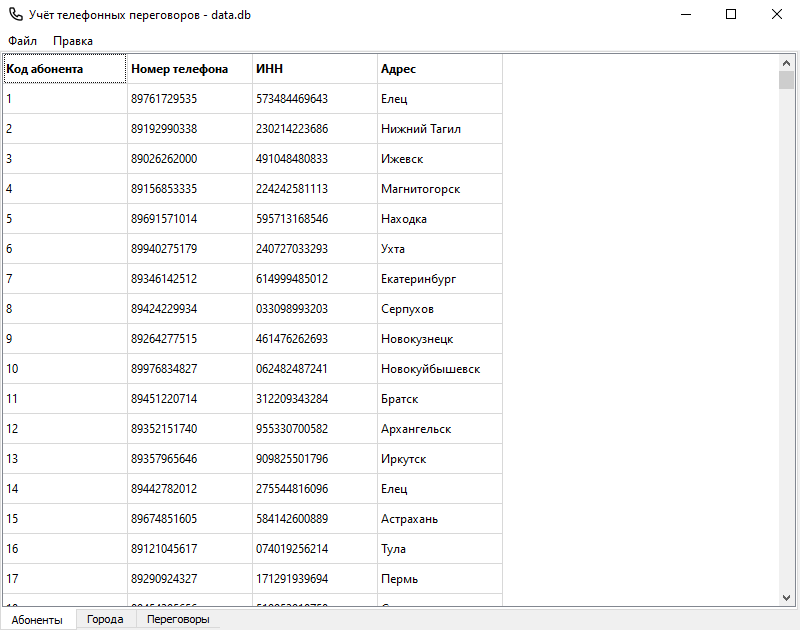


Рисунок 11. Данные таблицы *Абоненты*

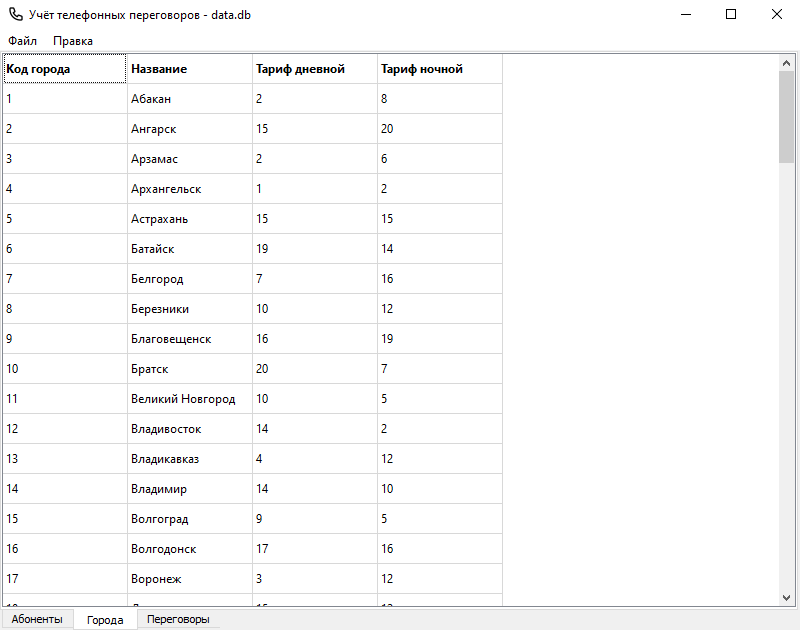


Рисунок 12. Данные таблицы *Города*

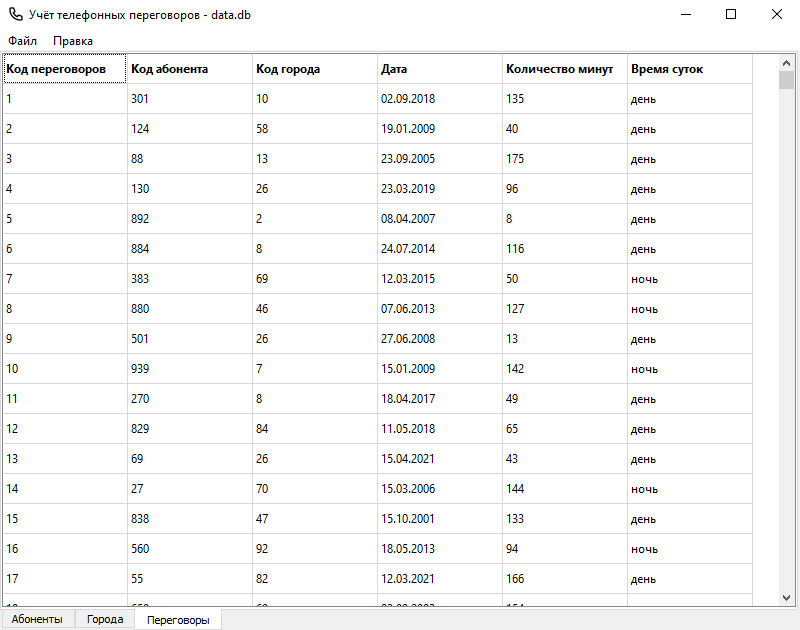


Рисунок 13. Данные таблицы *Переговоры*

Для того чтобы сохранить файл, необходимо выбрать пункт *Файл* верхнего меню, а затем выбрать пункт *Сохранить* (рис. 9).

Затем откроется диалоговое окно сохранения файла (рис. 11). Сохранённый файл будет иметь расширение .db.

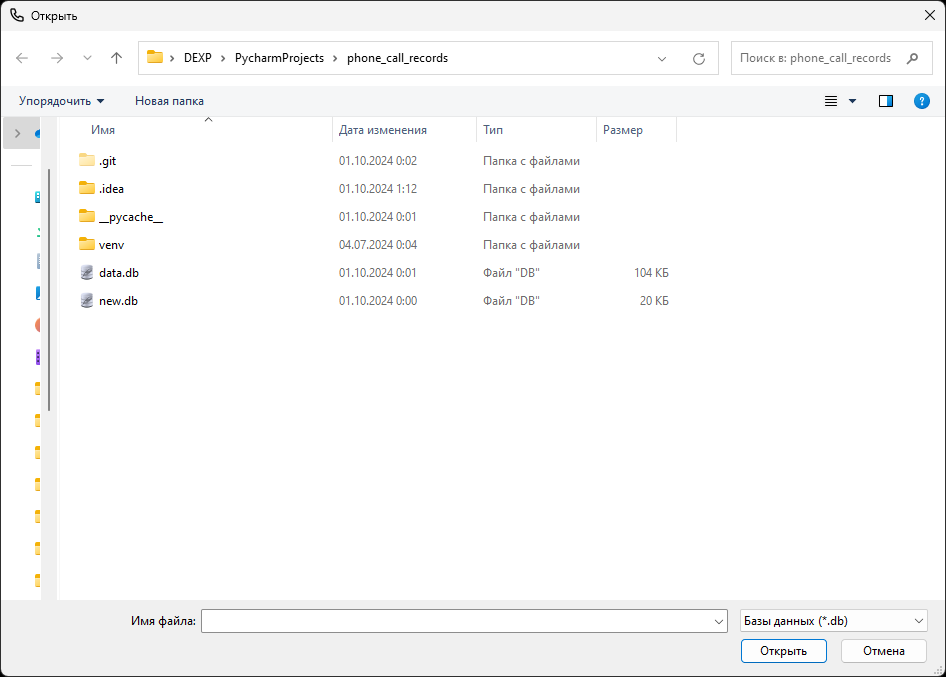


Рисунок 14. Диалоговое окно сохранения файла

При сохранении данных отправляются следующие запросы в базы данных:

DELETE FROM callers

INSERT INTO callers  
(caller\_id, phone\_number, tin, address)   
VALUES (?, ?, ?, ?) (для таблицы *Абоненты*)

DELETE FROM cities

INSERT INTO cities  
(city\_id, name, day\_rate, night\_rate)  
VALUES (?, ?, ?, ?) (для таблицы *Города*)

DELETE FROM conversations

INSERT INTO conversations  
(conversation\_id, caller\_id, city\_id, date, minute\_number, time)  
VALUES (?, ?, ?, ?, ?, ?) (для таблицы *Переговоры*)

Вместо знаков “?” подставляются данные, полученные из таблиц главного окна.

### Изменение файла

Для того чтобы добавить строку, необходимо выбрать пункт *Правка* верхнего меню, а затем выбрать пункт *Добавить строку* (рис. 15). Для того чтобы удалить строку, необходимо выбрать пункт *Правка* верхнего меню, а затем выбрать пункт *Удалить строку* (рис. 15).

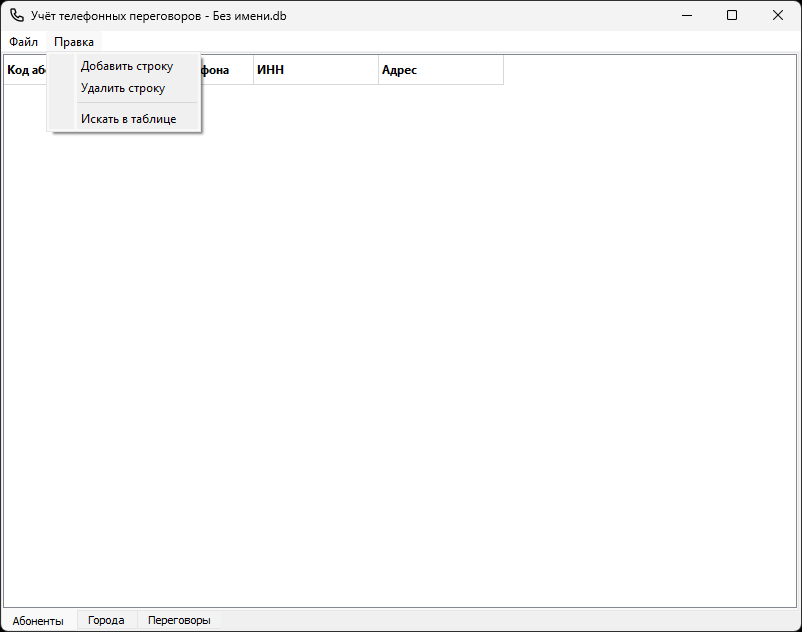


Рисунок 15. Пункт *Правка* верхнего меню

При добавлении строки используются значения по умолчанию. Для таблицы *Абоненты* – “1, 89000000000, 000000000000, Введите адрес” (рис. 16).

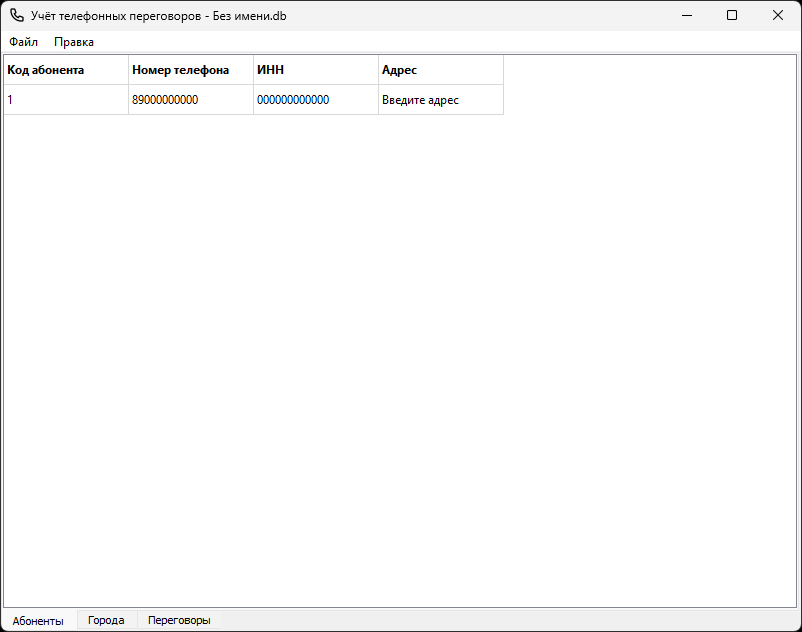


Рисунок 16. Значения по умолчанию для таблицы *Абоненты*

Для таблицы *Города* – “1, Введите город, 0, 0” (рис. 17).

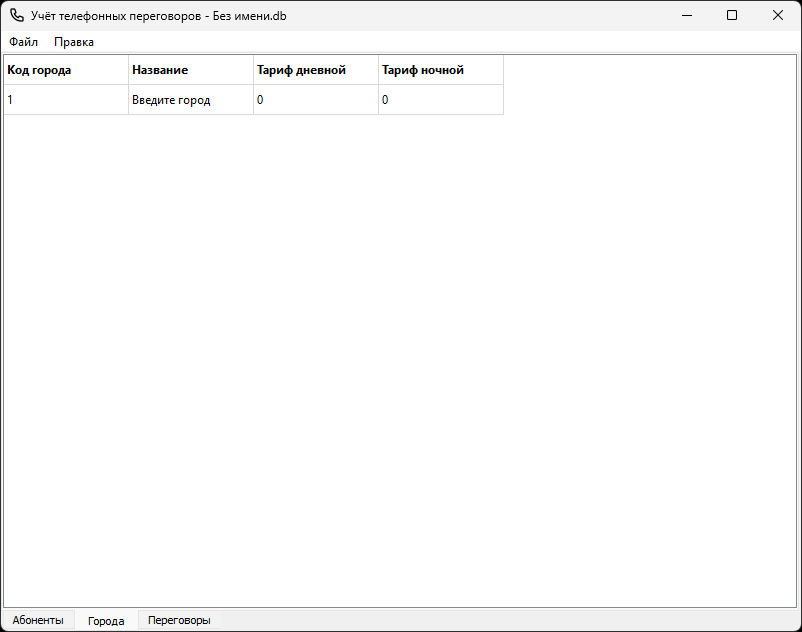


Рисунок 17. Значения по умолчанию для таблицы *Города*

Для таблицы *Переговоры – “*1, 1, 1, 01.01.01, 0, день*”* (рис. 18).

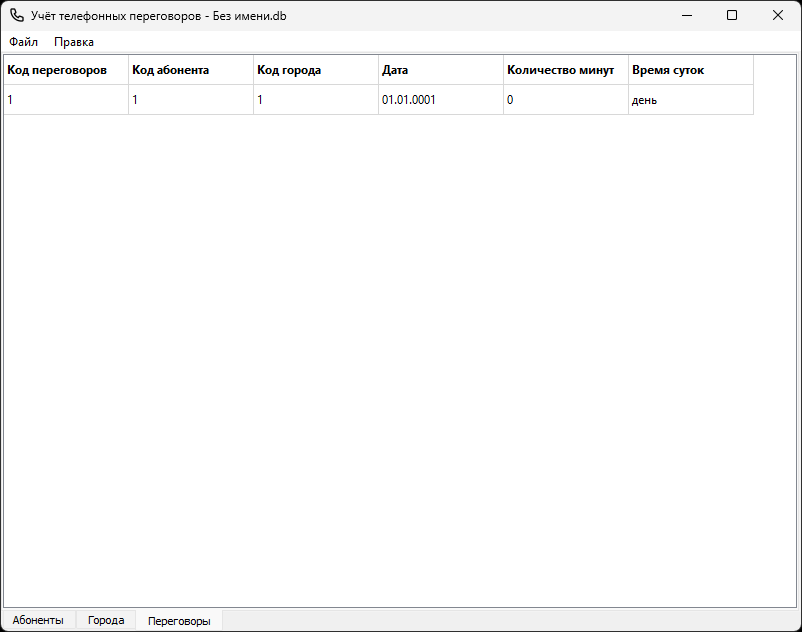


Рисунок 18. Значения по умолчанию для таблицы *Переговоры*

Причём следующий код для каждой таблицы будет получаться в соответствии с максимальным кодом из таблицы. Например, если в таблице максимальным кодом является 10, то при добавлении строки код будет иметь значение 11 (рис. 19).

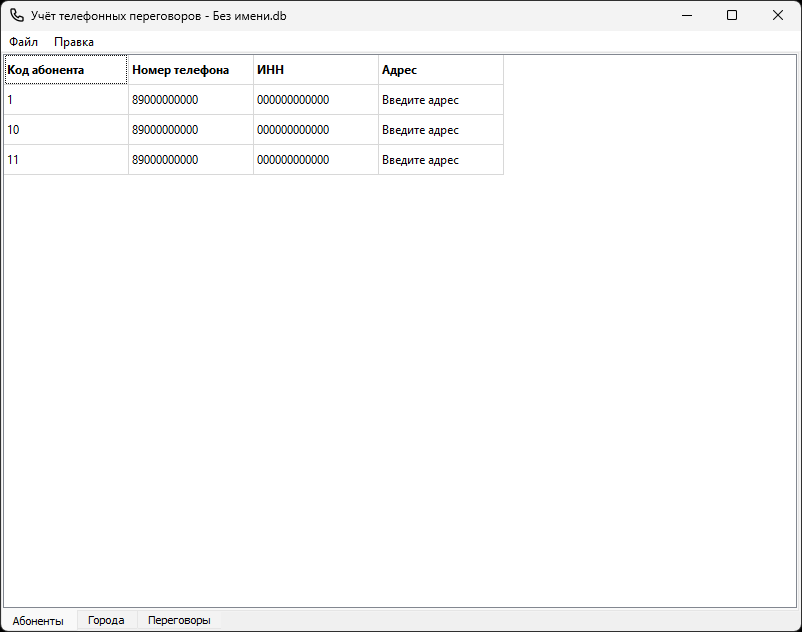


Рисунок 19. Пример получения следующего кода

### Проверка корректности данных

Для того чтобы изменить данные в таблице, необходимо выбрать нужную ячейку таблицы и вписать требуемое значение. При этом происходит проверка типа: если пользователь ввёл недопустимое значение, то ячейка таблицы примет красный цвет (рис. 20), а сохранение в файл будет недоступно (рис. 21).

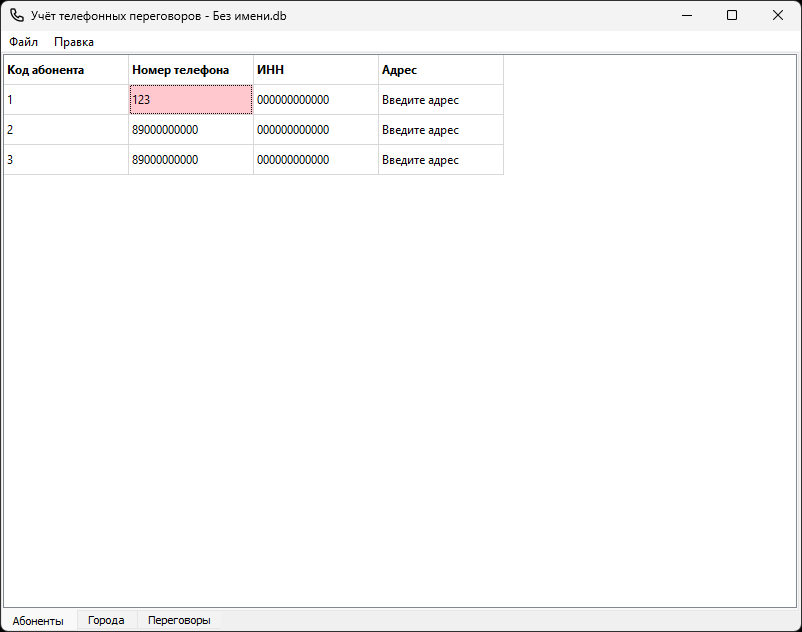


Рисунок 20. Ячейка красного цвета с неверным значением

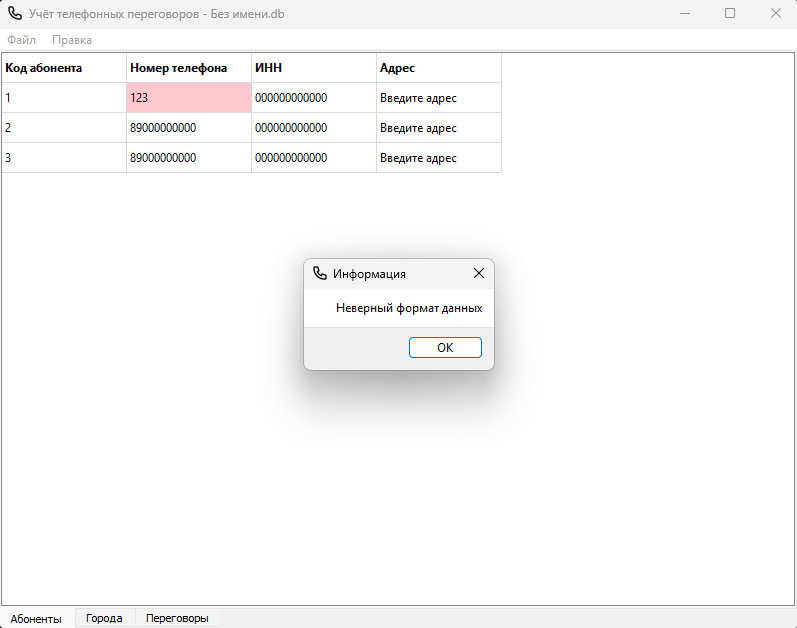


Рисунок 21. Ошибка сохранения файла

Для кода также производится дополнительная проверка на уникальность, то есть, если код повторится, то ячейка таблицы примет красный цвет и при попытке сохранить файл выведется соответствующее сообщение (рис. 22).

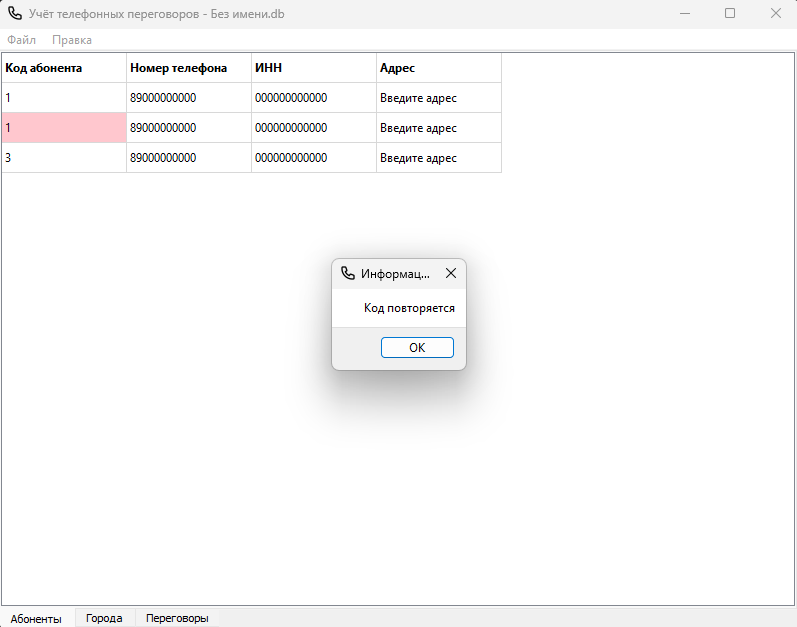


Рисунок 22. Ошибка с повторяющимся кодом

Для полей *код абонента* и *код города* в таблице *Переговоры* проверяется наличие этих кодов в связанных таблицах, то есть *Абоненты* и *Города* соответственно. Если код, не был найден в связанной таблице, то ячейка таблицы примет красный цвет и при сохранении файла выведется соответствующее сообщение (рис. 23).

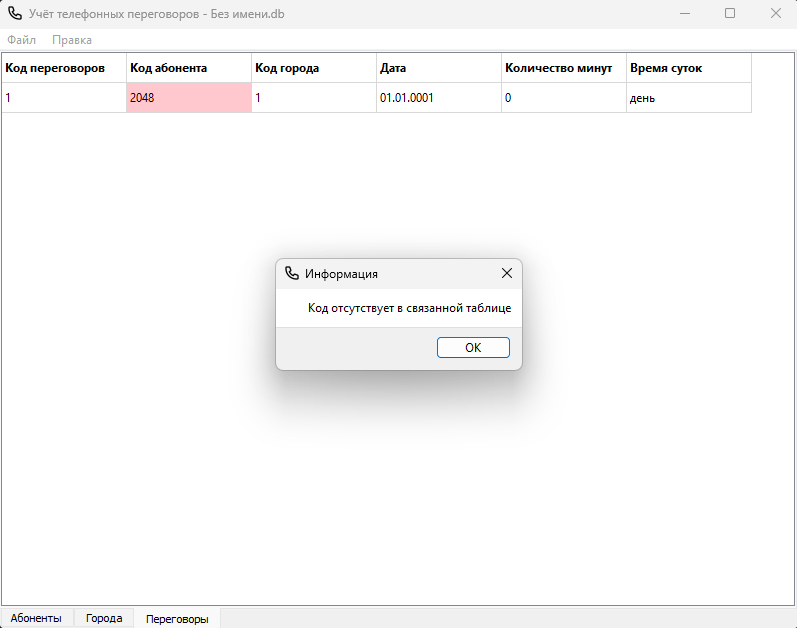


Рисунок 23. Ошибка, вызванная отсутствием кода в связанной таблице

### Поиск данных

Для осуществления поиска данных по таблице, необходимо выбрать пункт *Правка* верхнего меню, а затем выбрать пункт *Искать в таблице* (рис. 15). Затем над таблицей появится строка поиска (рис. 24).

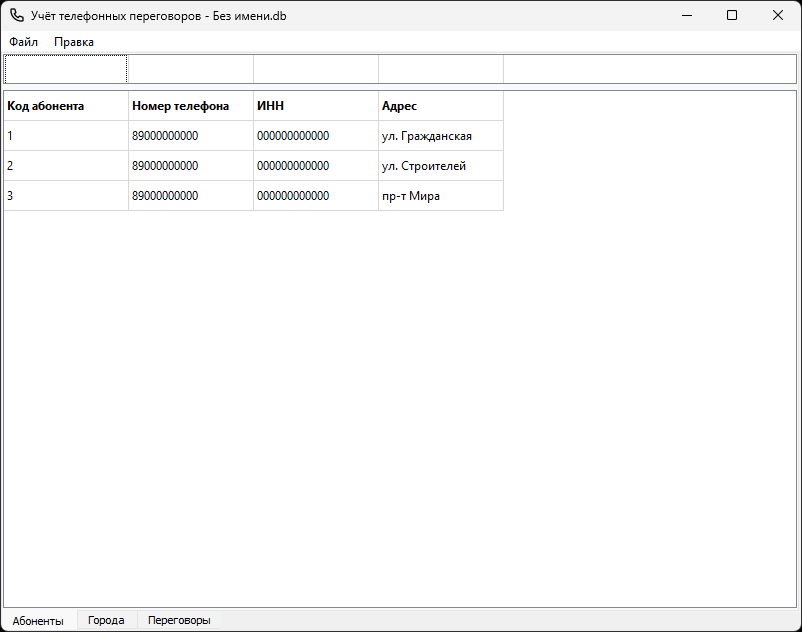


Рисунок 24. Строка поиска

При вводе запроса в строку поиска, в таблице остаются те значения, которые соответствуют запросу. Например, если ввести строку “ул”, то в таблице останутся лишь те записи, которые содержат строку “ул” (рис. 25).

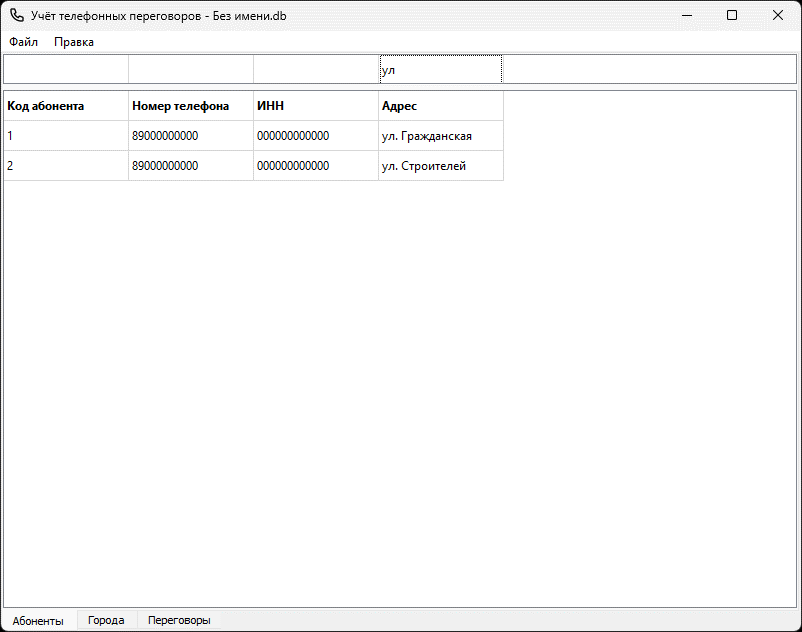


Рисунок 25. Записи, соответствующие запросу “ул”

# ЗАКЛЮЧЕНИЕ

В ходе прохождения учебной практики был приобретён опыт разработки кроссплатформенных приложений с использованием набора библиотек PyQt и языка программирования Python.

В ходе разработки приложения был проведён анализ, проектирование и работа с базами данных, активно использовались средства тестирования и отладки.

Работа над проектом поспособствовала улучшению навыков планирования рабочего дня, а также эффективного распределения задач и ресурсов.

Прохождение практики позволило увидеть, какие реальные задачи стоят перед разработчиками десктопных приложений в настоящее время.

Техническое задание было реализовано в полном объёме. Требования к заданию были учтены.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Python [Электронный ресурс]. URL: <https://docs.python.org/3.10/> (дата обращения 06.07.2024).
2. Документация PyQt [Электронный ресурс]. URL: <https://doc.qt.io/qtforpython-5/> (дата обращения 06.07.2024).
3. Документация SQLite [Электронный ресурс]. URL: <https://www.sqlite.org/docs.html> (дата обращения 06.07.2024).
4. Документация PyCharm [Электронный ресурс]. URL: <https://www.jetbrains.com/help/pycharm/getting-started.html> (дата обращения 06.07.2024).
5. ГОСТ 7.32-2017 Отчёт о научно-исследовательской работе (дата обращения 06.07.2024).

# ПРИЛОЖЕНИЕ А

## Исходный код программы (основные функции)

import sys

import sqlite3

from consts import \*

from PyQt5.QtGui import QIcon, QFont

from PyQt5.QtCore import Qt

from PyQt5.QtWidgets import QApplication, QMainWindow, QFileDialog, QTableWidgetItem, QMessageBox

from main\_window import Ui\_MainWindow as MainWindowUi

class MainWindow(MainWindowUi, QMainWindow):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.setupUi(self)

self.setWindowIcon(QIcon('icon.ico'))

self.action\_new.triggered.connect(self.action\_on\_new)

self.action\_open.triggered.connect(self.action\_on\_open)

self.action\_save.triggered.connect(self.action\_on\_save)

self.action\_exit.triggered.connect(self.close)

self.action\_add.triggered.connect(self.action\_on\_add)

self.action\_delete.triggered.connect(self.action\_on\_delete)

self.action\_search.triggered.connect(self.action\_on\_search)

self.tables = [self.table\_callers, self.table\_cities, self.table\_conversations]

self.searches = [self.search\_callers, self.search\_cities, self.search\_conversations]

self.results = [self.result\_callers, self.result\_cities, self.result\_conversations]

for table, search, result in zip(self.tables, self.searches, self.results):

table.itemChanged.connect(self.check\_table\_value)

search.cellChanged.connect(self.update\_result)

result.cellChanged.connect(self.edit\_table)

self.action\_on\_new()

self.action\_on\_search(self.action\_search.isChecked())

def action\_on\_add(self):

index = self.tab\_widget.currentIndex()

if self.action\_search.isChecked():

for table in (self.tables[index], self.results[index]):

row = table.rowCount()

insert\_into\_table(row, table, index)

else:

table = self.tables[index]

row = table.rowCount()

insert\_into\_table(row, table, index)

def action\_on\_delete(self):

if self.action\_search.isChecked():

table = self.results[self.tab\_widget.currentIndex()]

else:

table = self.tables[self.tab\_widget.currentIndex()]

row = table.currentRow()

item\_id = table.item(row, ID\_COLUMN).text()

if row != 0:

table.removeRow(row)

if self.action\_search.isChecked():

table = self.tables[self.tab\_widget.currentIndex()]

for row in range(1, table.rowCount()):

if table.item(row, ID\_COLUMN).text() == item\_id:

table.removeRow(row)

break

def action\_on\_search(self, checked):

status = self.check\_for\_errors()

if status == DATA\_CORRECT or not checked:

if checked:

for table in self.tables:

table.hide()

for table, search, result in zip(self.tables, self.searches, self.results):

search.show()

search.cellChanged.disconnect(self.update\_result)

fill\_table\_with\_data(search, [['' for \_ in range(table.columnCount())]], with\_headers=False)

search.cellChanged.connect(self.update\_result)

result.cellChanged.disconnect(self.edit\_table)

result.show()

data = [get\_table\_headers(table)] + get\_table\_data(table)

fill\_table\_with\_data(result, data)

result.cellChanged.connect(self.edit\_table)

else:

for table in self.tables:

table.show()

for search, result in zip(self.searches, self.results):

search.hide()

result.hide()

else:

self.action\_search.setChecked(False)

message = ERROR\_MESSAGES.get(status)

self.show\_message(message)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

ex = MainWindow()

ex.show()

sys.exit(app.exec())