**Министерство просвещения Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**МПГУ**

**Курсовая работа**

по учебной дисциплине

**«Программирование»**

на тему:

**«Разработка приложения для автоматизации бизнеса»**

**Выполнил:**

Студент 3 курса

**Код и группа подготовки:**

09.03.03 Прикладная информатика

Кожевникова Анастасия Станиславовна

**Научный руководитель:**

Доцент  
Ковалев Евгений Евгеньевич

**Москва**

**2020**

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ](#_heading=h.gjdgxs) **3**

[Глава 1. Язык программирования Python, графическая библиотека и база данных](#_heading=h.30j0zll) **6**

[1.1 Описание предметной области и актуальность программы](#_heading=h.1fob9te) 6

[1.2 Язык программирования Python и модульное программирование](#_heading=h.2et92p0) 9

[1.3 Графическая библиотека PyQt5](#_heading=h.tyjcwt) 11

[1.4 Реляционные базы данных и язык запросов SQL](#_heading=h.3j2qqm3) 13

[Вывод по первой главе](#_heading=h.1y810tw) 14

[Глава 2: Создание приложения](#_heading=h.4i7ojhp) **15**

[2.1 Определение основных этапов разработки и главных функций приложения](#_heading=h.2xcytpi) 15

[2.2 Создание и подключение базы данных](#_heading=h.1ci93xb) 17

[2.3 Создание интерфейса и функциональной части приложения](#_heading=h.3whwml4) 22

[2.4 Тестирование интерфейса и функциональной части приложения](#_heading=h.2bn6wsx) 30

[Вывод по второй главе](#_heading=h.qsh70q) 32

[ЗАКЛЮЧЕНИЕ](#_heading=h.3as4poj) **33**

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ](#_heading=h.1pxezwc) **34**

[**ПРИЛОЖЕНИЕ**](#_heading=h.y684yubl2000) **36**

# **ВВЕДЕНИЕ**

В настоящие дни не только развиваются технологии, но также увеличивается и количество людей. Одни пользуются новыми технологиями, другие же – приходят за помощью к профессионалам. Именно эти люди становятся потенциальными клиентами для различного рода бизнеса. В связи с увеличением рабочей нагрузки на любого рода бизнес, в том числе и на фирмы ИП, процесс автоматизации становится более **актуальным**.

Автоматизация открывает перед людьми огромные возможности, позволяя сократить время рутинной работы, такой как внесение данных о каждом клиенте в Excel или же сортировка заказов на сайте, и заняться более важными делами: продвижением организации на рынке или составлением бизнес-планов.

Кроме того, что автоматизация освобождает время, которое является одним из наиболее важных нематериальных ресурсов во все времена, данный процесс также может стать отличной помощью в ведении бизнеса. Держать всех клиентов и поставщиков, организовывать автоматическое заполнение документов для печати или ФНС, создать отгрузку или получение товара – все это в одном приложении.

Реализация автоматизации возможна на различных языках программирования. Основными требованиями являются возможность создания графического интерфейса и непосредственно возможность подключения базы данных.

Одним из таких языков является объектно-ориентированный язык программирования Python. Данный язык позволяет достаточно легко, по сравнению с другими языками, реализовать графический интерфейс с помощью стандартных библиотек, а также имеет встроенную библиотеку для комфортного взаимодействия с подключенной базой данных, в которых и будут находиться такие основные данные, как информация о клиентах, о заказах, о товаре на складе и т. д.

**Существенный вклад в развитие языка** программирования Python внесли авторы книг, подробно описывающие различные тонкости в работе с языком программирования. По книгам учатся как новички, так и профессионалы, давно знакомые с Python. Одними из самых популярных и востребованных книг для ознакомления с языком программирования являются учебные пособия «Python для чайников» от Джона Пола Мюллера, «Изучаем Python» от Эрика Мэтиза, а также книга российских разработчиков и редакторов Николая Прохоренок и Владимира Дронова «Python 3. Самое необходимое».

**Целью** работы станет разработка приложения для автоматизации бизнеса с помощью языков Python и SQL.

**Предметом** работы будет являться изучение использования Python и SQL технологий для разработки приложения с графическим интерфейсом.

**Объектом** работы станет приложение «***TransAgencyHelper***», с помощью которого можно будет вносить данные по заказам.

**Задачи:**

* Изучить Python-технологии, графические библиотеки;
* Проанализировать требования к приложению, выделить основные функции;
* Разработать графический интерфейс;
* Создать базу данных SQLite;
* Интегрировать базу данных в приложение;
* Протестировать приложение и сделать выводы по полученным результатам;

В работе будут использованы язык программирования Python, язык запросов к базам данных SQL, графическая библиотека PyQt5, а также библиотека интегрирования базы данных sqlite3.

В работе будет использована **методология** объектно – ориентированного программирования, представляющая собой четкую иерархию между компонентами программы, а также **методология** модульного программирования, или декомпозиция программного кода, где большой проект разбивается на отдельные, более понятные и простые модули.

Практической значимостью является то, что **в результате** исследования приложение позволит автоматизировать бизнес-процессы. Пользователем данного приложения может стать любой сотрудник компании, отвечающий за ведение отчетности.

Работа будет содержать введение, 2 главы, вывод, список используемых источников, приложение.

# **Глава 1. Язык программирования Python, графическая библиотека и база данных**

## **1.1 Описание предметной области и актуальность программы**

Бюро Переводов (БП) занимается предоставлением услуг по нотариальному переводу, письменному и устному переводу, легализации документов и апостилю в городе Москва. Так же БП предоставляет дополнительные услуги, в том числе выезд курьера за документами на дом к заказчику, легализация в консульстве и пр.

БП работает как с постоянными клиентами, так и с новыми заказчиками. Каждый вид документа, а также каждая языковая пара рассчитывается по отдельной стоимости, которые впоследствии составляют итоговую стоимость за заказ. К итоговой стоимости по необходимости добавляется стоимость дополнительных услуг, после чего формируется документ об оказанных услугах для предоставления заказчику и получения от него оплаты.

Для непосредственного выполнения работ БП нанимает соответствующих сотрудников, которыми в большинстве случаев являются свободные (наемные) специалисты, выполняющие разовые заказы. Постоянными сотрудниками БП являются *руководитель* и *главный бухгалтер*.

На данный момент БП ведет свою отчетность программе Microsoft Office Excel, что является крайне неудобным, учитывая быстро развивающуюся клиентскую базу, необходимость в быстром расчете стоимости заказа, а также необходимость в наличии быстрого доступа ко всем имеющимся данным по конкретному заказу или клиенту.

Большинство приложений и информационных систем ориентировано на автоматизацию финансовой деятельности или деятельности компаний в целом, но для такого типа бизнеса, как бюро переводов, имеется только пять продуктов автоматизации. Для работы над созданием приложения был проведен сравнительный анализ трех из пяти продуктов – аналогов, наиболее популярных в кругу переводческих компаний.

Сравнительный анализ приведен в таблице 1.

***Таблица 1. Сравнительный анализ продуктов – аналогов***

|  |  |  |  |
| --- | --- | --- | --- |
| **Название /**  **Функции** | **QTRM** | **Protemos** | **УСУ – универсальная система учета** |
| **Количество пользователей** | От 1 до 5 рабочих мест | 1 рабочее место – отдельная лицензия | 1 рабочее место – отдельная лицензия |
| **Стоимость продукта** | 60 000 – 612 400 руб. | 45$/месяц для 1 сотрудника 45$ ~ 3300 руб. | 4 000 руб./ месяц для 1 сотрудника |
| **Внесение заказов** | + | + | + |
| **Добавление заказчиков** | + | + | + |
| **Добавление исполнителей** | – | + | – |
| **Добавление документов** | + | – | – |
| **Языковые пары** | – | + | – |
| **Расчет стоимости заказа** | + | + | + |
| **Отправка документов по почте** | + | + | – |

В таблице представлены самые основные функции, которые могут обеспечить комфортную работу переводческой компании, в отдельных продуктах также есть ряд собственных функций, таких как выставление счетов от исполнителей, выгрузка данных в Excel и др., но, как можно заметить, даже существующие продукты не могут в полной мере охватить весь спектр требуемых для работы с бюро переводов функций. В случае, представленном в работе, когда БП не имеет постоянных исполнителей заказов, а нанимает свободных специалистов для работы, большинство функционала данных программ не актуально, а потому не подходит для работы, ведь представленный функционал автоматически входит в стоимость продукта. БП подобные программы не выгодны, потому как придется платить за тот функционал продукта, который не будет использоваться в ходе работы.

Также для большего понимания актуальности был проведен анализ уже существующих переводческих бюро, многие из которых являются индивидуальными предпринимателями с минимальным штатом постоянных сотрудников.

Необходимо учитывать и то, что в настоящее время намного больше людей интересуется иностранными языками, культурой зарубежных стран или желает путешествовать не только по родной стране, по и по зарубежью. В связи с этим, на бюро переводов ложится большая работа по переводу отдельных документов, либо целому пакету документов, необходимых для получения визы на посещение ряда стран. Именно поэтому становится актуальной разработка приложения для автоматизации рабочего процесса БП с минимально необходимым для данной сферы функционалом.

## **1.2 Язык программирования Python и модульное программирование**

Python – многофункциональный и широко используемый язык программирования. Он используется как для интегрирования на сайты и в приложения, так и для создания программ с нуля. Язык автоматически поставляется с бесплатной операционной системой Linux. Он популярен как среди и индивидуальных разработчиков, так и в больших компаниях для создания серьезных продуктов. Сейчас Python также активно используется в разработке алгоритмов машинного обучения (Machine Learning), имея собственные библиотеки для реализации обучения искусственного интеллекта.

Познакомиться с Python миру позволил сотрудник голландского института CWI Гвидо ван Россум. Изначально он создавал данный язык программирования для распределенной операционной системы на досуге, заимствовав знания из языка ABC, в разработке которого также принимал участие. Постепенно, благодаря активной и положительной поддержке сообщества программистов в 1991 году Python вышел как официальный язык программирования. Изначально была широко распространена версия Python 2.7, но 3 декабря 2008 вышла обновленная версия Python 3.0, которая с апреля 2020 года стала основной веткой развития Python, более старая версия, хоть и была практически полностью совместима с обновлением, постепенно вышла из пользования ввиду поддержания меньшего числа библиотек и функций самого языка.

При работе с языком программирования Python, как и со многими другими объектно-ориентированными языками, удобно использовать методологию модульного проектирования, или декомпозиции.

**Декомпозиция** – это процесс разбиения большой, громоздкой задачи на более маленькие, управляемые модули и подзадачи. [17]

У модульного подхода к разработке кода громоздких программ существует ряд нескольких преимуществ:

1. **Простота**. За счет разбиения программного кода на модули, облегчается понимание функциональной значимости отдельных частей программы, а также уменьшается вероятность получения большого числа ошибок из-за попыток разобраться в большом количестве функций и методов кода.
2. **Модифицируемость**. Благодаря составлению программы из отдельных модулей, которые практически, а иногда и полностью, не содержат взаимосвязей, появляется возможность изменения отдельных модулей без влияния на остальные части программы, что позволяет работать с программным кодом сразу команде разработчиков из нескольких человек.
3. **Повторное использование кода**. Один модуль, с помощью которого реализована некая функциональность, может быть в дальнейшем использован для другого проекта, в котором содержится подобный функционал, что избавляет от необходимости дублирования и сохраняет время.
4. **Область действия**. Каждый модуль содержит собственное пространство имен, которое не пересекается с другими и помогает избежать ошибок в понимании принадлежности переменных или цели их использования.[17]

## **1.3 Графическая библиотека PyQt5**

Приложение, с которым будет работать пользователь, должно иметь простой и понятный графический интерфейс. Интерфейс бывает как низкоуровневым, представленным в виде обычной командной строки, так и высокоуровневым, содержащим элементы взаимодействия с приложением, такие как: кнопки управления, строки ввода, интерактивное меню и др.

Вместе с развитием технологий низкоуровневый интерфейс в виде командной строки становится все менее актуальным, востребованным и понятным для обычного пользователя, к тому же подобные приложения часто встречаются с ошибкой использования кодировки при запуске на другой машине.

Python, являясь популярным и быстро развивающимся языком, имеет несколько вариантов для создания полноценного высокоуровневого интерфейса. Самыми распространенными являются графические библиотеки tkinter и PyQt5. В работе будет использована вторая библиотека, поэтому именно на ней необходимо остановиться более подробно.

Библиотека, или расширение, языка программирования PyQt5 является версией графического фреймворка Qt, реализующей набор расширений для создания графических интерфейсов приложений. PyQt также включает в себя Qt Designer, использующийся для дизайна графического интерфейса в «ручном» режиме путем добавления на рабочую область элементов интерфейса. [14]

Qt Designer — это инструмент Qt для проектирования и построения графических пользовательских интерфейсов (GUI) с помощью виджетов Qt. С помощью этого инструмента можно создавать и настраивать окна или диалоги в стиле «Что видишь, то и получаешь» (what-you-see-is-what-you-get - WYSIWYG) и тестировать их с использованием различных стилей и разрешений. [9]

Виджеты и формы, созданные с помощью Qt Designer, легко интегрируются с запрограммированным кодом, используя механизм сигналов и слотов Qt, так что для каждого графического элемента легко назначить соответствующее поведение в соответствии с установленными сигналами. Все свойства, заданные в Qt Designer, могут быть динамически изменены в коде. Кроме того, такие функции, как продвижение виджетов и пользовательские плагины, позволяют создавать и использовать свои собственные компоненты с Qt Designer.[14]

Кроме того, расширение PyQt позволяет подключать внешнюю базу данных, работать с запросами к базам данных на языке SQL, а также имеет несколько шаблонов для отображения таблиц с данными в интерфейсе приложения.

## **1.4 Реляционные базы данных и язык запросов SQL**

Реляционные базы данных (РБД) представляют собой базы данных, которые используются для хранения информации и предоставления доступа к взаимосвязанным элементам таблиц.[12] Такие базы данных основаны на реляционной модели, которая интуитивно понятна и имеет наглядный табличный способ представления данных. Каждая строка таблицы, содержащаяся в такой базе данных, являет собой запись с уникальным идентификатором, также называемым ключом. Столбцы таблицы имеют атрибуты, или описание, данных, а каждая запись содержит значение для того или иного атрибута, что позволяет устанавливать связь между элементами данных. Реляционная модель позволяет наиболее эффективно поддерживать целостность данных внутри одной базы. [12]

Для взаимодействия с РБД существуют языки структурированных запросов, помогающие создавать, модифицировать и управлять данными внутри таблиц базы данных. Но одного языка запросов для работы с базой данных недостаточно, необходима также среда управления, поддерживающая отображение баз данных и взаимодействие с ними посредством языка запросов. В работе будет использована СУБД – система управления базами данных – SQLite.

SQLite — это библиотека языка Си, которая реализует быстрый и автономный движок базы данных SQL, отличающийся высокой надежностью и компактностью. SQLite является одной из самых популярных в мире систем управления базами данных, которая встроена во все мобильные телефоны и большинство компьютеров и поставляется в комплекте с огромным количеством других приложений, использующимися людьми каждый день.[11]

В отличие от большинства других баз данных SQL, SQLite не имеет отдельного серверного процесса. SQLite читает и записывает непосредственно в обычные дисковые файлы. Полная база данных SQL с несколькими таблицами, индексами, триггерами и представлениями содержится в одном файле на диске. Формат файла базы данных является кроссплатформенным, что позволяет свободно копировать базу данных между 32-битными и 64-битными системами. [10]

## **Вывод по первой главе**

В первой главе мы познакомились с объектно-ориентированным языком Python, а также разобрали одну и его часто используемых библиотек – PyQt5, с помощью которой во второй главе будет создавать функциональное наполнение приложения. Также было сказано о приложении Qt Designer, благодаря которому во второй главе будет описан процесс создания визуального отображения программы «TransAgencyHelper».

Для сохранения введенных и уже имеющихся данных будут использованы таблицы SQL вместе с СУБД SQLite, которая является популярным выбором в качестве формата файла для базы данных.

Во второй главе будет показано создание минимально необходимой для работы приложения базы данных, а также создание внешнего вида диалоговых окон и добавление функциональных связей между элементами окон.

# **Глава 2: Создание приложения**

## **2.1 Определение основных этапов разработки и главных функций приложения**

Для того, чтобы определить главные функции будущего приложения, необходимо снова обратиться к пункту 1.1 и таблице 1, в которой представлен сравнительный анализ продуктов – аналогов. Исходя из представленной таблицы, а также из информации о БП, можно сделать вывод о необходимых для приложения функциях, а именно:

* Внесение данных о новых заказах;
* Установка языковой пары для заказа;
* Внесение данных по заказчикам;
* Внесение данных по исполнителям;
* Расчет стоимости заказа;

Эти функции являются основными для создания приложения и дальнейшего его использования. Остальные возможности, такие как: установка статуса заказа, ведение прайс-листов, возможность отправки документов по электронной почте и пр. – являются побочными функциями, которые могут быть, а могут и не быть добавлены в функционал приложения.

Теперь, когда необходимые для приложения функции определены, важно также определить основные этапы разработки приложения.

Во-первых необходимо создать простую базу данных, в которую будут записываться все вносимые через приложение данные. Такая база данных нужна не только для первичного внесения данных, но и для того, чтобы при последующей работе с программой данные также сохранялись и не приходилось их заново вносить.

Далее следует работа над интерфейсом приложения, который должен быть минималистичен и понятен в использовании даже для тех, кто не привык работать с подобными программами. Именно интерфейс в большинстве случаев является «визитной карточкой» приложения, по которому будущие пользователи могут ориентироваться, понимая, насколько удобным для них будет использование того или иного продукта. Так как разрабатываемое в рамках работы приложение имеет цель осуществить базовые и необходимые для работы функции, то сделать интерфейс приложения понятным для работы будет намного легче, чем если бы была необходимость реализовывать сразу десятки возможных функций.

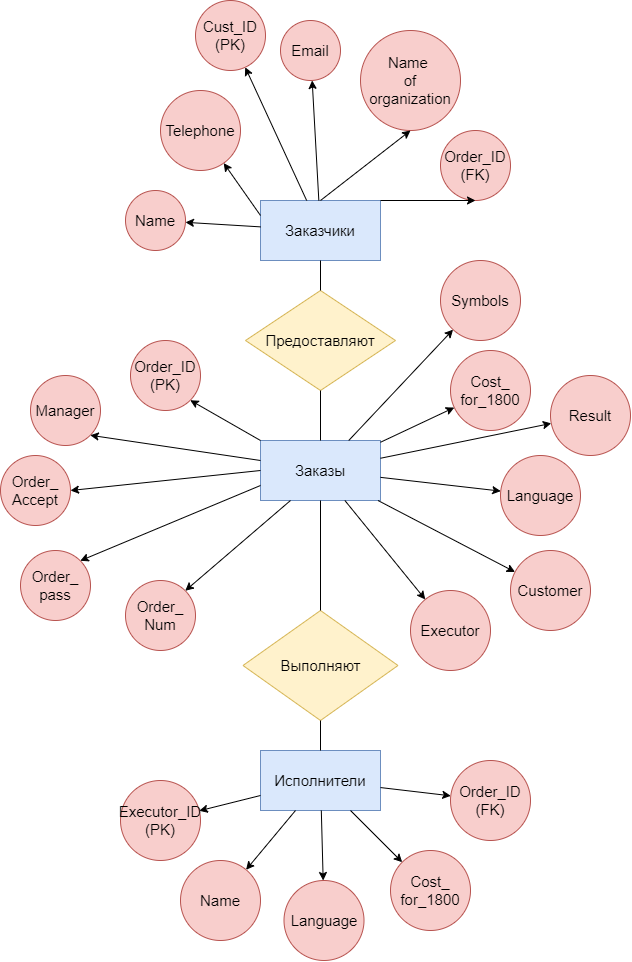
Заключающим этапом разработки является настройка приложения посредством прописанного кода, с помощью которого будут соединены созданные на этапе ранее окна, а также добавлена возможность вносить и редактировать данные. Также на этом этапе будет необходимым проверить правильность работы приложения, а именно: добавление данных, сохранение их при последующих открытиях приложения и корректное отображение всех необходимых вычислений.

## **2.2 Создание и подключение базы данных**

Для описания таблиц и связей между ними в базе данных выбрана ER-модель, использующаяся для описания концептуальной схемы предметной области. ER-модель сама по себе представляет лишь формальную конструкцию, которая не предписывает никакой визуализации, и только при проектировании непосредственно базы данных ER-диаграмма преобразуется в конкретную схему на основе реляционной модели данных. [7]

Стандартной графической нотацией, с помощью которой происходит визуализация ER-диаграммы, является диаграмма «сущность – связь» (от англ. Entity – Relationship diagram, или ERD).[7]

На рисунке 1 представлена созданная диаграмма для будущего приложения, разработанная в программе Draw.io.



***рис. 1 ER-диаграмма базы данных приложения***

На основе составленной диаграммы в программе SQLite необходимо создать таблицы, соответствующие представленным на диаграмме, а также содержащим описанные атрибуты.

В SQLite есть возможность создать таблицу как с помощью программного кода, так и с помощью простого автоматического добавления. В случае создания этой базы данных воспользуемся созданием таблиц посредством написания соответствующих запросов на языке SQL.

Создание таблиц «Заказы», «Исполнители», «Заказчики», «Стоимость заказа», «Оплата» и «Итог» представлено ниже.

CREATE TABLE "Orders\_Info" (

"Order\_ID" INTEGER PRIMARY KEY AUTOINCREMENT,

"Manager" TEXT,

"Order\_accept" TEXT,

"Order\_pass" date,

"Executor\_ID" TEXT,

"Cust\_ID" TEXT,

"Order\_Num" INTEGER,

"Languages" TEXT,

"Tax" INTEGER,

"Symbols" INTEGER,

"Results" INTEGER )

***Листинг 1. Создание таблицы «Заказы»***

CREATE TABLE "Executors" (

"Executor\_ID" INTEGER PRIMARY KEY AUTOINCREMENT,

"Executor\_name" TEXT,

"Languages" TEXT,

"Cost\_for\_1800s" INTEGER,

"Order\_ID" INTEGER,

FOREIGN KEY ("Order\_ID") REFERENCES Orders\_Info("Order\_ID"))

***Листинг 2. Создание таблицы «Исполнители»***

CREATE TABLE "Customers" (

"Cust\_ID" INTEGER PRIMARY KEY AUTOINCREMENT,

"Cust\_name" TEXT,

"Cust\_phone\_number" TEXT,

"Cust\_email" TEXT,

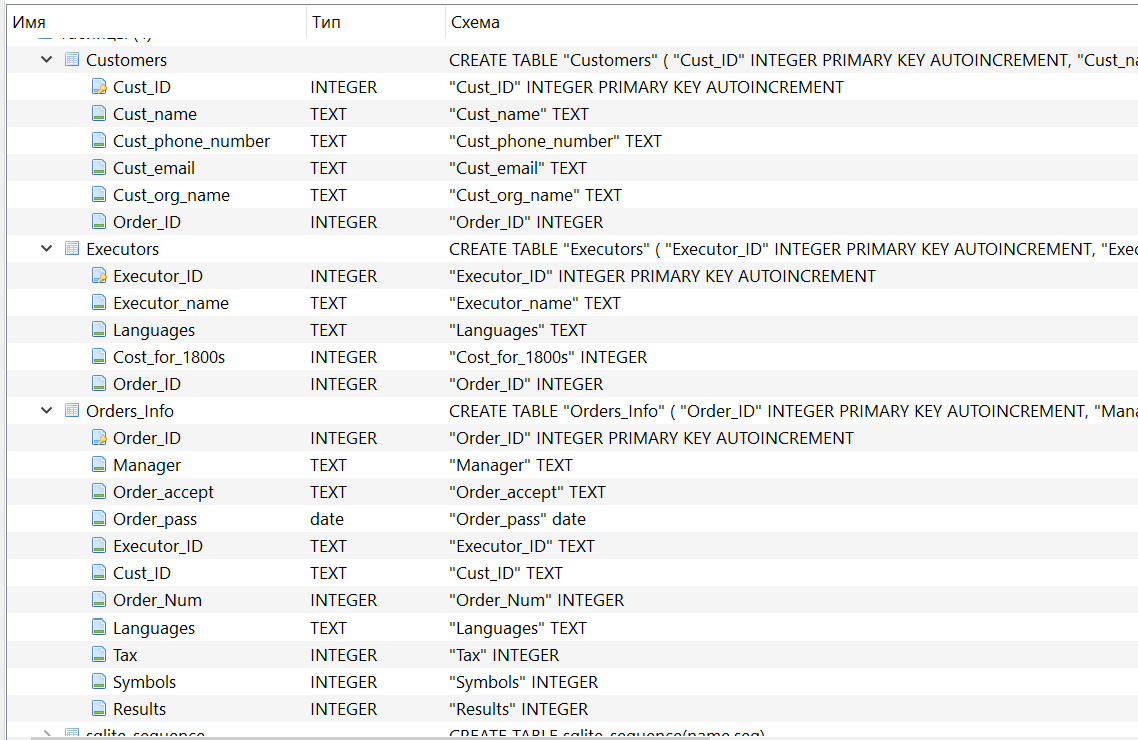
"Cust\_org\_name" TEXT,

"Order\_ID" INTEGER,

FOREIGN KEY ("Order\_ID") REFERENCES Orders\_Info("Order\_ID"))

***Листинг 3. Создание таблицы «Заказчики»***

После выполнения всех запросов, в программе появятся все созданные таблицы, а также их атрибуты, которые можно будет увидеть на вкладке «Структура БД».



***рис. 2 Интерфейс программы SQLite с созданными таблицами***

Для того, чтобы с базой данных можно было работать из приложения, а не напрямую из СУБД SQLite, необходимо внутри программного кода подключить базу данных. Для подключения будут использоваться вложенная в PyQt5 библиотека QtSql, а также отдельный класс данной библиотеки – QSqlQuery.

Программный код выглядит следующим образом:

con = QtSql.QSqlDatabase.addDatabase(**'QSQLITE'**)

con.setDatabaseName(**'TAH.db'**)

con.open()

stm = QtSql.QSqlTableModel()

stm.setTable(**'Orders\_Info'**)

stm.setEditStrategy(QSqlTableModel.OnFieldChange)

stm.setSort(1, QtCore.Qt.AscendingOrder)

stm.select()

model = QtSql.QSqlQueryModel()

model.setQuery(**"""SELECT \* FROM Orders\_Info"""**)

self.mainTable.setModel(model)

***Листинг 4. Подключение базы данных к приложению***

В методе 1-й строки addDatabase устанавливается вид файла базы данных. Это необходимо, потому что существуют различные СУБД, которые по-своему интерпретируют SQL запросы, и для того, чтобы корректно обращаться к базе данных с помощью запросов, необходимо правильно выбрать тип базы данных. Так как в рамках курсовой работы используется СУБД SQLite, то параметр для метода равен «QSQLITE».

Во 2-й и 3-й строке прописывается имя той базы данных, к которой необходимо обратиться, и соответствующая база данных «открывается». Далее выбирается определенная таблица из базы данных, с которой будет происходить непосредственно дальнейшая работа.

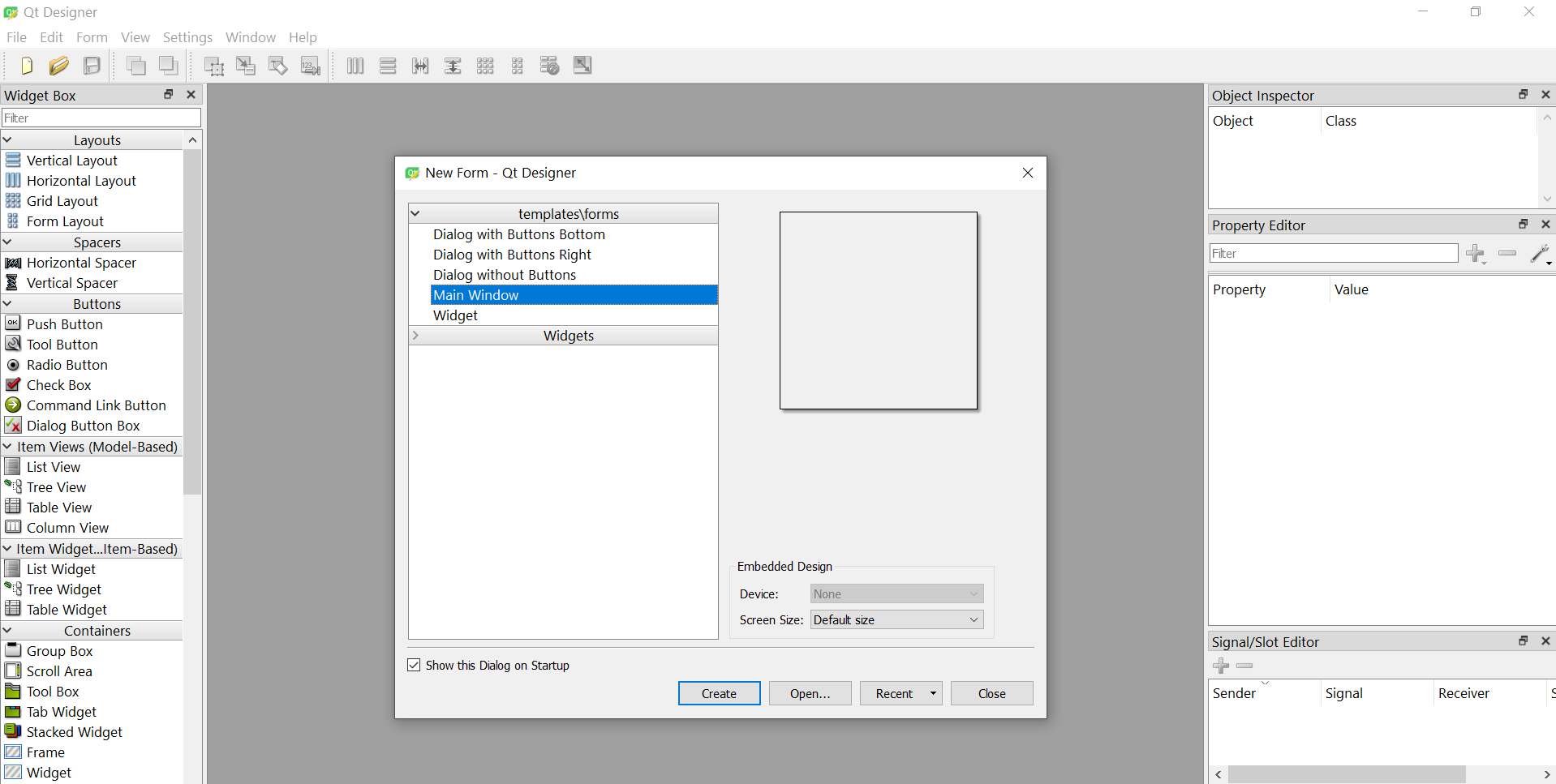
С помощью метода setQuery можно обращаться к выбранной таблице напрямую, используя SQL запросы.

Для дальнейшего использования базы данных необходимо также разместить данный фрагмент кода в том классе или методе программы, в котором требуется подключение к базе данных.

## **2.3 Создание интерфейса и функциональной части приложения**

Как уже было сказано выше, интерфейс любой программы должен быть интуитивно понятен и прост в использовании, а также содержать минимально необходимое количество наполнения для более легкой навигации по приложению. Если приложение громоздкое, заполнено текстом малого шрифта и множеством полей, то пользователю будет сложнее ориентироваться, труднее понимать, куда именно необходимо нажать.

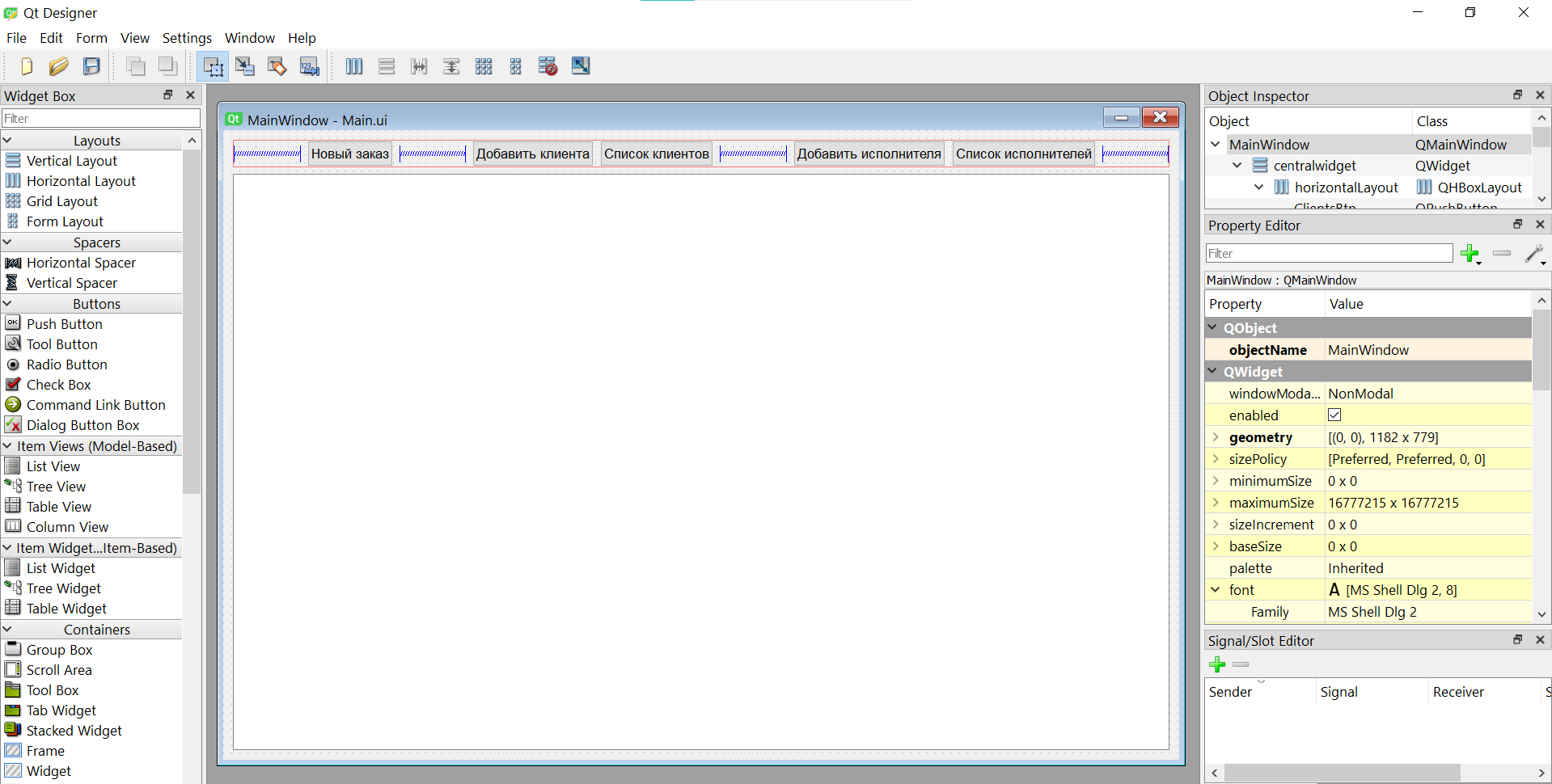
В первой главе кратко было рассказано о таком инструменте создания графического интерфейса, как Qt Designer. Его начальное окно содержит панель выбора шаблона будущего приложения. Для начинающих пользователей советуется выбирать Main Window, а профессионалы могут настраивать шаблоны под свои требования, создавая собственные и модифицируя имеющиеся с помощью программного кода.



***рис. 3 Начальное окно приложения***

Слева представлена панель элементов, которые размещаются на главном окне в выбранном пользователем порядке. Справа представлены сверху вниз: список элементов внутри главного окна, меню редактирования атрибутов каждого элемента, меню создания «сигналов» для элементов, которые также являются функциями внутри программного кода, отвечающими за какое-либо действие. Сигналы могут добавляться как в программе Qt Designer, так и при непосредственном редактировании конечного кода окна.

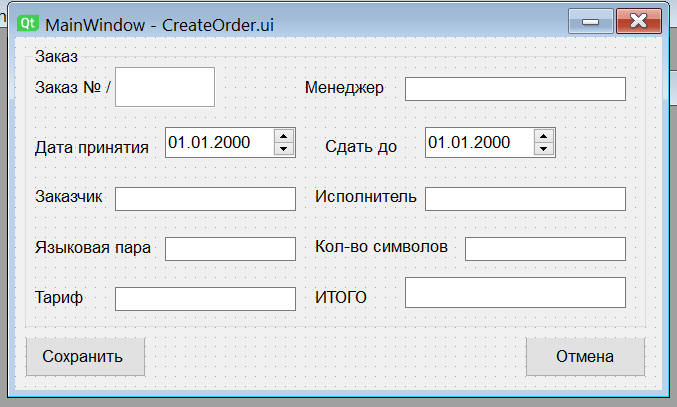
На рис. 4 представлено созданное в программе главное окно для будущего приложения.



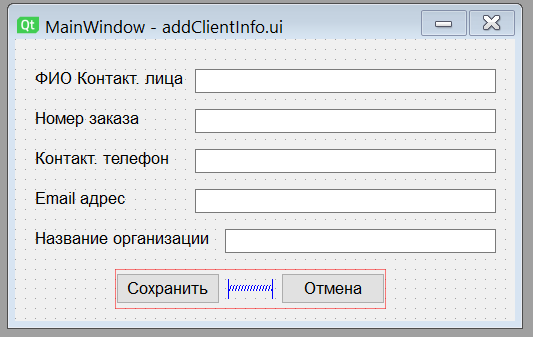
***рис. 4 Главное окно для приложения «TransAgencyHelper»***

С помощью инструмента Qt Designer также создается еще пять окон:

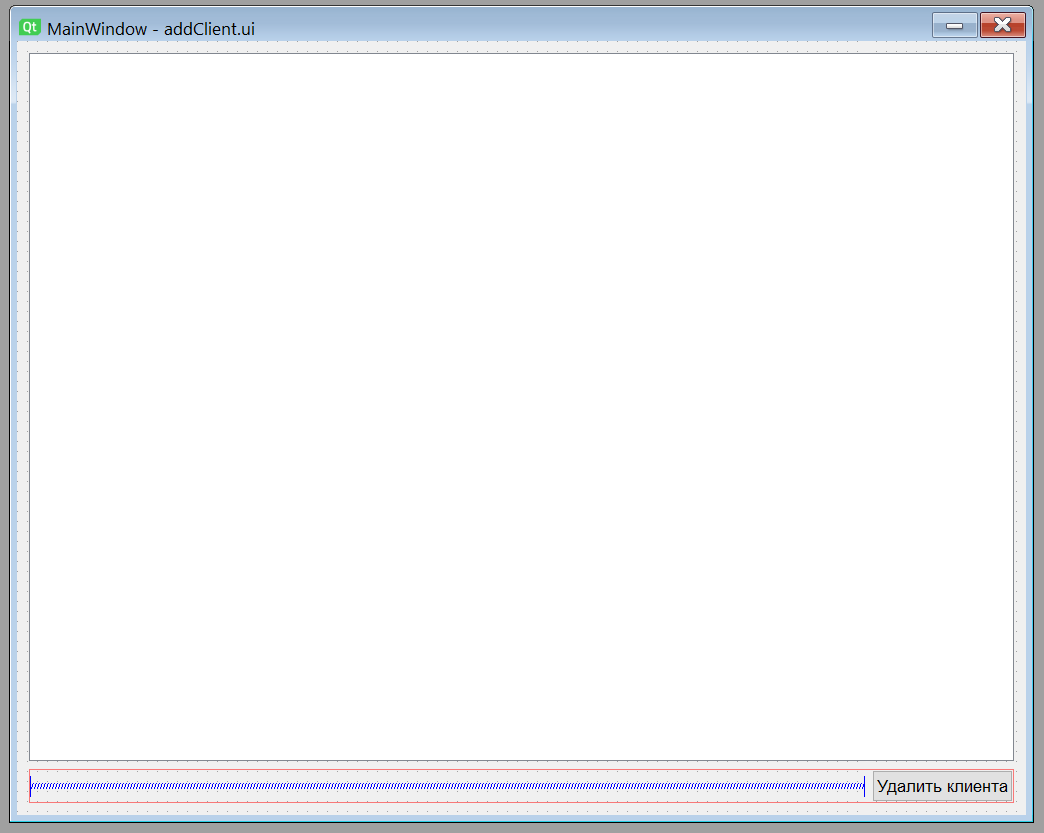
1. Добавление информации по заказу
2. Добавление информации по клиенту
3. Отображение таблицы клиентов
4. Добавление информации по исполнителю
5. Отображение таблицы исполнителей



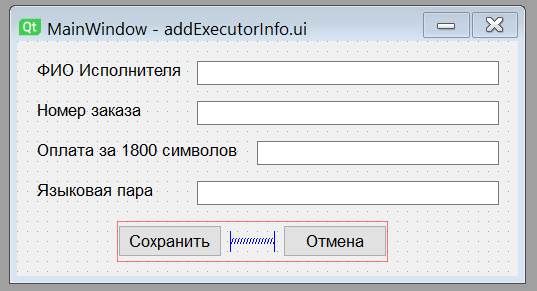
***рис. 5 «Добавление информации по заказу»***



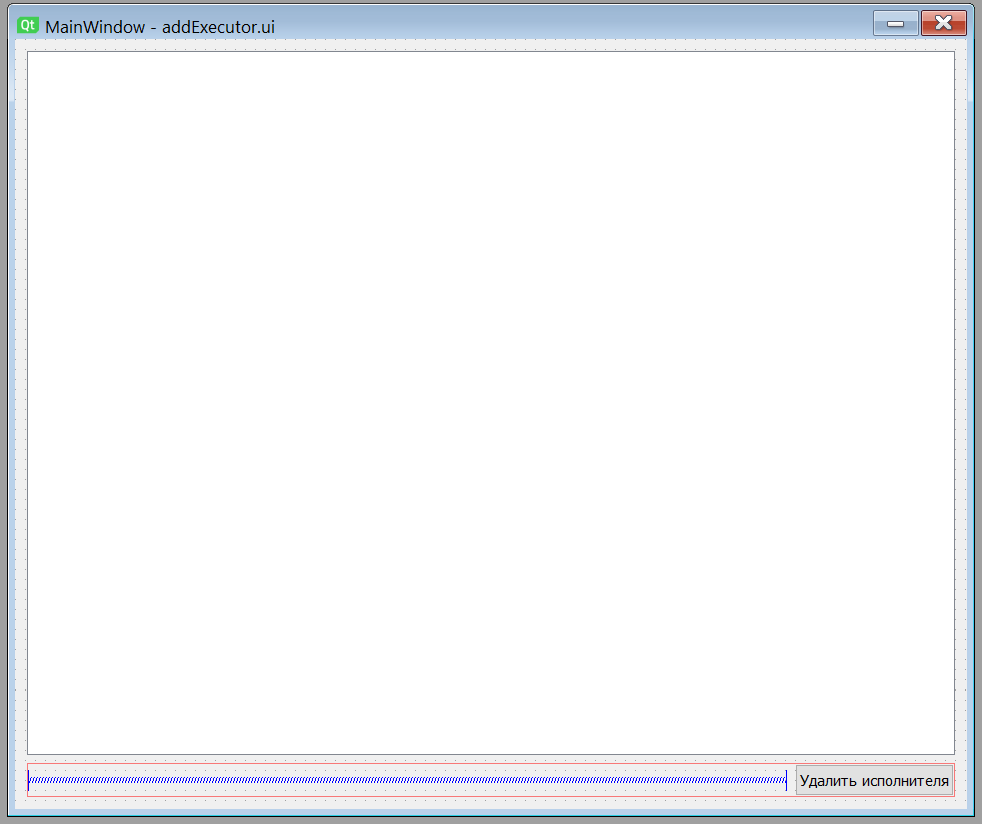
***рис. 6 «Добавление информации по клиенту»***



***рис. 7 «Отображение таблицы клиентов»***



***рис. 8 «Добавление информации по исполнителю»***



***рис. 9 «Отображение таблицы исполнителей»***

Как видно из представленных изображений, поля, на которых в дальнейшем будут располагаться таблицы, на данный момент не содержат никакой информации. Отображение данных из базы данных, а также отображение новых созданных заказов создается непосредственно в коде программы.

Теперь, когда интерфейс программы настроен, необходимо добавить функциональную часть, благодаря которой будут считываться данные при нажатии на кнопки и появятся данные из базы данных.

Для работы созданы следующие файлы:

* Main3.py – основной файл, благодаря которому запускается всё созданное приложение, содержит методы для вывода дополнительных окон при нажатии на кнопки добавления заказа, клиента или исполнителя;
* Main.py – файл главного окна приложения, содержит визуальное отображение элементов главного окна, т. е. пять кнопок в верхней части окна и пространство таблицы в остальной части, а также содержит функциональные методы для отображения на пустом пространстве таблицы базы данных с названием «Orders\_Info»;
* CreateOrder.py – файл визуального отображения элементов для создания заказа, включает в себя элементы, передающие информацию по заказу в SQL запрос для добавления строки нового заказа в базу данных;
* addClient.py – файл визуального всех имеющихся данных по клиентам из таблицы «Customers» базы данных;
* addClientInfo.py – файл визуального отображения элементов для создания нового клиента, который включается в себя элементы, передающие атрибуты в SQL запрос для добавления строки нового клиента в базу данных;
* addExecutor.py – файл визуального всех имеющихся данных по клиентам из таблицы «Executors» базы данных;
* addExecutorInfo.py – файл визуального отображения элементов для создания нового исполнителя, который включается в себя элементы, передающие атрибуты в SQL запрос для добавления строки нового исполнителя в базу данных;
* TAH.db – файл базы данных, содержащий таблицы и элементы таблиц, т. е. введенные данные

Программный код для добавления новой строки данных в таблицу выглядит следующим образом:

**def** add\_to\_table(self):

con = QtSql.QSqlDatabase.addDatabase(**'QSQLITE'**)

con.setDatabaseName(**'TAH.db'**)  
con.open()

stm = QtSql.QSqlTableModel()

stm.setTable(**'Customers'**)  
stm.setSort(1, QtCore.Qt.AscendingOrder)

stm.select()  
query = QSqlQuery()

FIOValue = self.clientFIOLine.text()

PhoneValue = self.clientTelLine.text()

OrdNumValue = self.clientOrdNumLine.text()

EmailValue = self.clientEmailLine.text()

OrganValue = self.clientOrganLine.text()

**if** FIOValue != **"" and** PhoneValue != **"" and** OrdNumValue != **""**

**and** EmailValue != **"" and** OrganValue != **""**:  
 query.prepare(**"INSERT INTO Customers(Cust\_name,**

**Cust\_phone\_number, Cust\_email, Cust\_org\_name, Order\_ID)"**

**"VALUES (?, ?, ?, ?, ?)"**)

query.addBindValue(FIOValue)  
 query.addBindValue(PhoneValue)  
 query.addBindValue(EmailValue)  
 query.addBindValue(OrganValue)  
 query.addBindValue(OrdNumValue)  
  
 query.exec()  
 self.clientFIOLine.clear()  
 self.clientOrdNumLine.clear()  
 self.clientTelLine.clear()  
 self.clientEmailLine.clear()  
 self.clientOrganLine.clear()

***Листинг 5. Добавление новой записи в таблицу базы данных***

В первой части кода происходит подключение базы данных, а именно таблицы «Customers». Из элементов окна, в которые пользователь будет вводить данных, в переменные сохраняются их значения, которые после проверки на наличие значения становятся атрибутами в SQL запросе типа INSERT. При успешном добавлении строки в таблицу, введенные с клавиатуры значения стираются из полей ввода для комфортного и быстрого ввода данных следующей строки.

Также для отображения записей выбранной таблицы в указанном для этого поле, используется следующий код:

con = QtSql.QSqlDatabase.addDatabase(**'QSQLITE'**)

con.setDatabaseName(**'TAH.db'**)  
con.open()  
stm = QtSql.QSqlTableModel()

stm.setTable(**'Customers'**)  
stm.setEditStrategy(QSqlTableModel.OnFieldChange)  
stm.setSort(1, QtCore.Qt.AscendingOrder)

stm.select()

model = QtSql.QSqlQueryModel()

model.setQuery(**"""SELECT Cust\_ID, Cust\_name, Cust\_phone\_number, Cust\_email, Cust\_org\_name, Order\_ID FROM Customers"""**)  
model.setHeaderData(0, QtCore.Qt.Horizontal, **'№/'**)  
model.setHeaderData(1, QtCore.Qt.Horizontal, **'Фамилия Имя'**)  
model.setHeaderData(2, QtCore.Qt.Horizontal, **'Контактный телефон'**)  
model.setHeaderData(3, QtCore.Qt.Horizontal, **'Электронная почта'**)  
model.setHeaderData(4, QtCore.Qt.Horizontal, **'Организация'**)  
model.setHeaderData(5, QtCore.Qt.Horizontal, **'Номер заказа'**)  
self.clientTable.setModel(model)

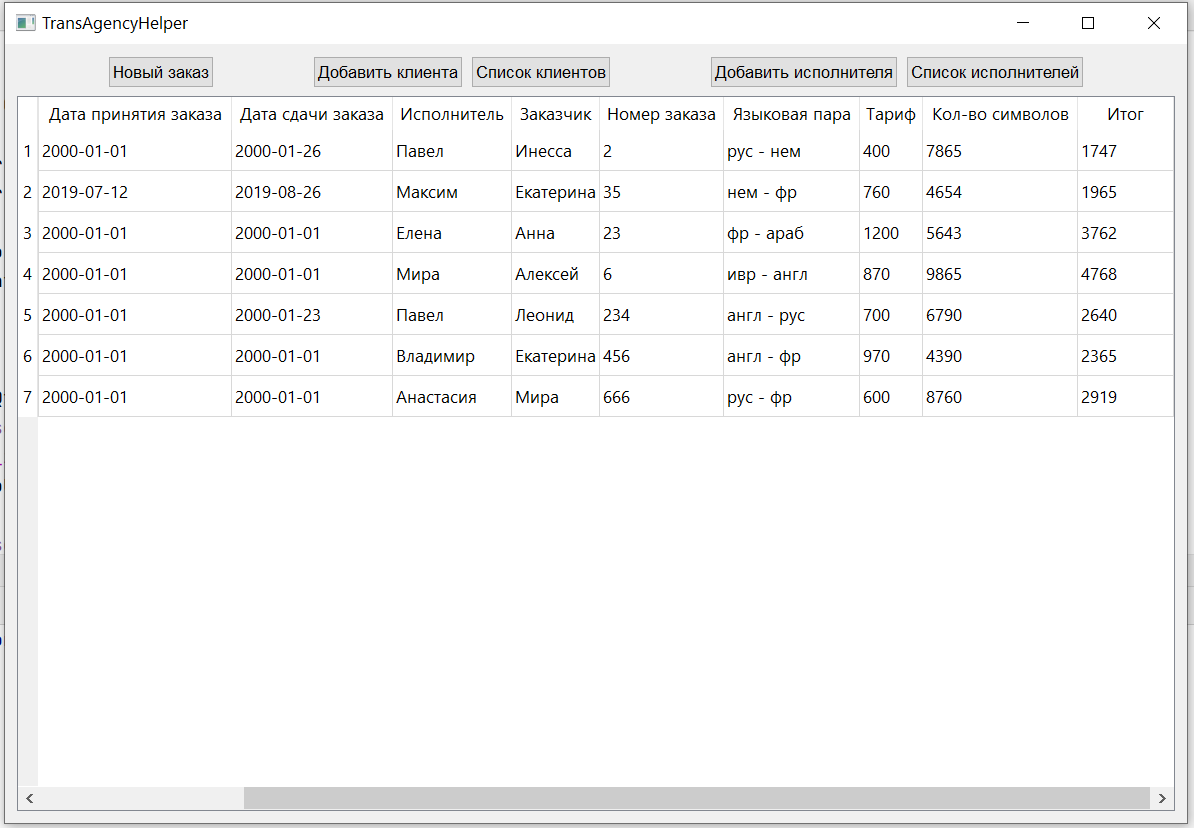
***Листинг 6. Отображение записей из таблицы базы данных***

В данном программном коде сначала происходит подключение к базе данных, выбор соответствующей таблицы «Customers», затем с помощью SQL запроса типа SELECT выбираются необходимые для отображения поля данных и оформляются в соответствии с видом таблицы.

Программный код для таблиц заказов и исполнителей имеет практически такой же вид, за исключением выбранных для отображения полей и самой таблицы, с которой необходимо работать.

## **2.4 Тестирование интерфейса и функциональной части приложения**

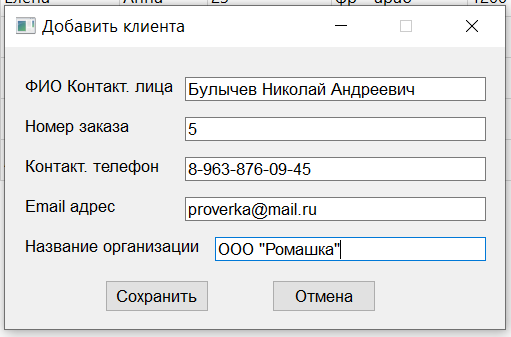
Готовое приложение запускается из основного файла **main3.py**, и главное окно теперь имеет новый вид, отличающийся от макета в конструкторе, стали доступны все кнопки, а также в главном поле появились записи из базы данных с информацией об уже введенных заказах.



***рис. 10 Главный экран приложения***

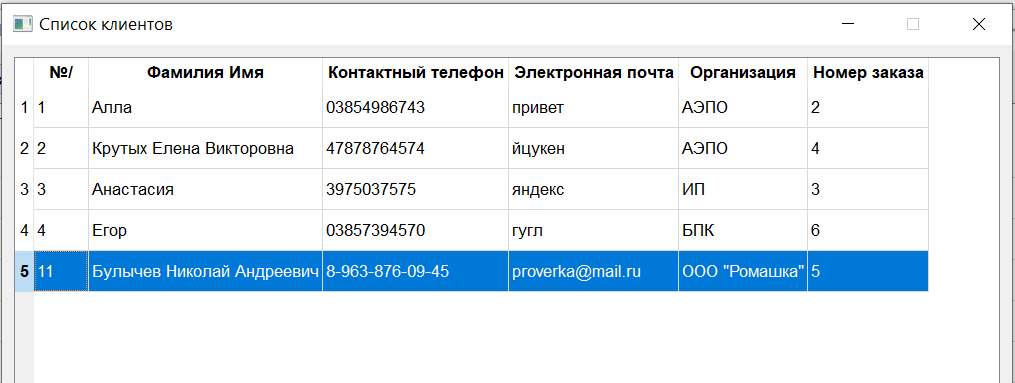
В качестве тестирования приложения добавим нового клиента и проверим его отображение в списке клиентов из базы данных.

На рис. 11 представлены данные, которые будут являться информацией о клиенте, а также посредством кода станут атрибутами для запроса INSERT.



***рис. 11 Данные нового клиента***

Нажимаем кнопку «Сохранить», а затем переходим в окно, где отображаются все клиенты компании.



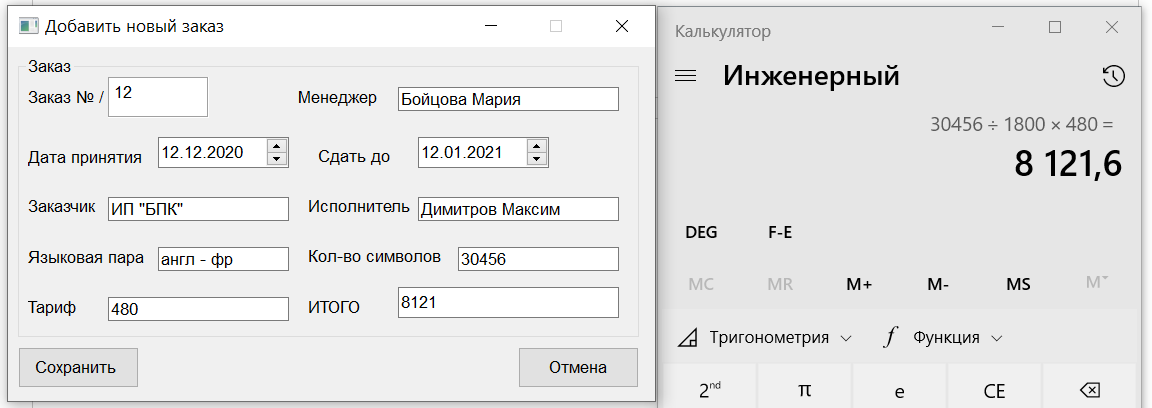
***рис. 12 Новый клиент в списке клиентов***

Как можно увидеть на рис. 12, новый клиент успешно был добавлен в нашу таблицу клиентов, и находится под порядковым номером 11.

Также для тестирования необходимо проверить правильность расчета итоговой стоимости заказа. Стоимость заказа рассчитывается по формуле ***Кол-во Символов / 1800 \* Тариф***.

При добавлении нового заказа и нажатии кнопки «Сохранить» (рис. 13), итоговая стоимость автоматически появляется в соответствующем поле под словом «ИТОГ».

Добавим данные нового заказа и проверим правильность рассчитанной суммы итога.



***рис. 13 Добавление нового заказа и расчет стоимости***

Для проверки также был произведен расчет на калькуляторе, окно которого видно на представленном изображении (рис. 13). Учитывая, что созданное приложение всегда округляет числа в меньшую сторону, можно утверждать, что итоговая сумма просчитана верно.

## **Вывод по второй главе**

Во второй главе была создана необходимая для записи данных база данных посредством СУБД SQLite. С помощью инструмента Qt Designer было создано визуальное отображение каждого окна программы, а также при помощи модификации кода, работа с которым проходила в среде разработки PyCharm, был описан функциональный модуль работы программы.

Созданное приложение было протестировано на правильность отображения визуальной составляющей, на правильность добавления записей в таблицы, на стабильное подключение к базе данных, а также на верность математической составляющей программы.

# **ЗАКЛЮЧЕНИЕ**

В результате исследовательской работы было разработано приложение для организации Бюро Переводов, имеющее следующие возможности:

* Добавление нового заказа;
* Добавление новых клиентов;
* Добавление новых исполнителей;
* Установка языковой пары заказа;
* Расчет стоимости созданного заказа;

Исходя из перечня реализованных возможностей, можно сделать вывод, что наличие предполагаемых минимальных функций для программы было выполнено. Также следует отметить, что при небольших изменениях в SQL запросах и выборе подключенной базы данных позволят легко настроить приложение для работы любой организации, потребностью которых является наличие программы для простого учета клиентов, заказов и расчета стоимостей.

Для выполнения цели исследовательской работы, которая заключалась в разработке приложения, были решены следующие поставленные задачи:

* Выделены основные функции и требования к приложению;
* Изучены Python – технологии и графические библиотеки;
* Разработан графический интерфейс;
* Создана база данных на основе СУБД SQLite;
* База данных была интегрирована в приложение;
* Приложение было протестировано;
* На основе полученных результатов сделаны выводы;

Таким образом, созданное в ходе работы приложение может являться как самостоятельным простым приложением для учета необходимых данных, так и шаблоном для разработки более функционального приложения под потребности конкретной организации.

# **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

**Литература:**

1. Базы Данных: учеб. пособие для студ. учреждений сред. проф. образования / Э. В. Фуфаев, Д. Э. Фуфаев, 7-е издание. – Москва, Издательский центр «Академия», 2012
2. Изучаем программирование на Python / Пол Бэрри; [пер. с англ. М. А. Райтмана]. – Москва, Издательство «Э», 2017
3. Легкий способ выучить Python / Зэд Шоу; [пер. с англ. М. А. Райтмана]. – Москва, Издательство «Эксмо», 2019
4. Объектно – ориентированное программирование / П. Б. Хорев, 4-е издание. – Москва, Издательский центр «Академия», 2004
5. Понимание SQL / Мартин Грубер, [пер. с англ. Лебедева В. Н., под ред. Булычева В. Н.]. – Москва, 1993
6. Совершенный код. Мастер – класс / Пер. с англ. — М. : Издательство «Русская редакция», 2010

**Электронные ресурсы:**

1. ER-модель [Электронный ресурс] // Свободная энциклопедия «ВикипедиЯ». URL: [https://ru.wikipedia.org/wiki/ER-модель](https://ru.wikipedia.org/wiki/ER-%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)
2. PyCharm: The Python IDE for Professional Developers [Электронный ресурс] // Официальный сайт JetBrains. URL: <https://www.jetbrains.com/pycharm/>
3. Python v3.8.7 Документация [Электронный ресурс] // Официальный сайт Python. URL: <https://docs.python.org/3.8/>
4. PyQt5 Документация [Электронный ресурс] // Официальный сайт лицензии и документации по PyQt5. URL: <https://www.riverbankcomputing.com/static/Docs/PyQt5/>
5. Qt Designer Документация [Электронный ресурс] // Официальный сайт Qt5. URL: <https://doc.qt.io/qt-5/qtdesigner-manual.html>
6. SQL Документация [Электронный ресурс] // Официальный сайт SQL. URL: <https://www.sql.ru/docs/>
7. SQLite Документация [Электронный ресурс] // Официальный сайт SQLite. URL: <https://www.sqlite.org/docs.html>
8. Архитектура приложений на Python — модули и пакеты [Электронный ресурс] // URL: <http://chel-center.ru/python-yfc/2020/07/12/30488/>
9. Документация работы с SQLite в Python [Электронный ресурс] // Официальный сайт Python.

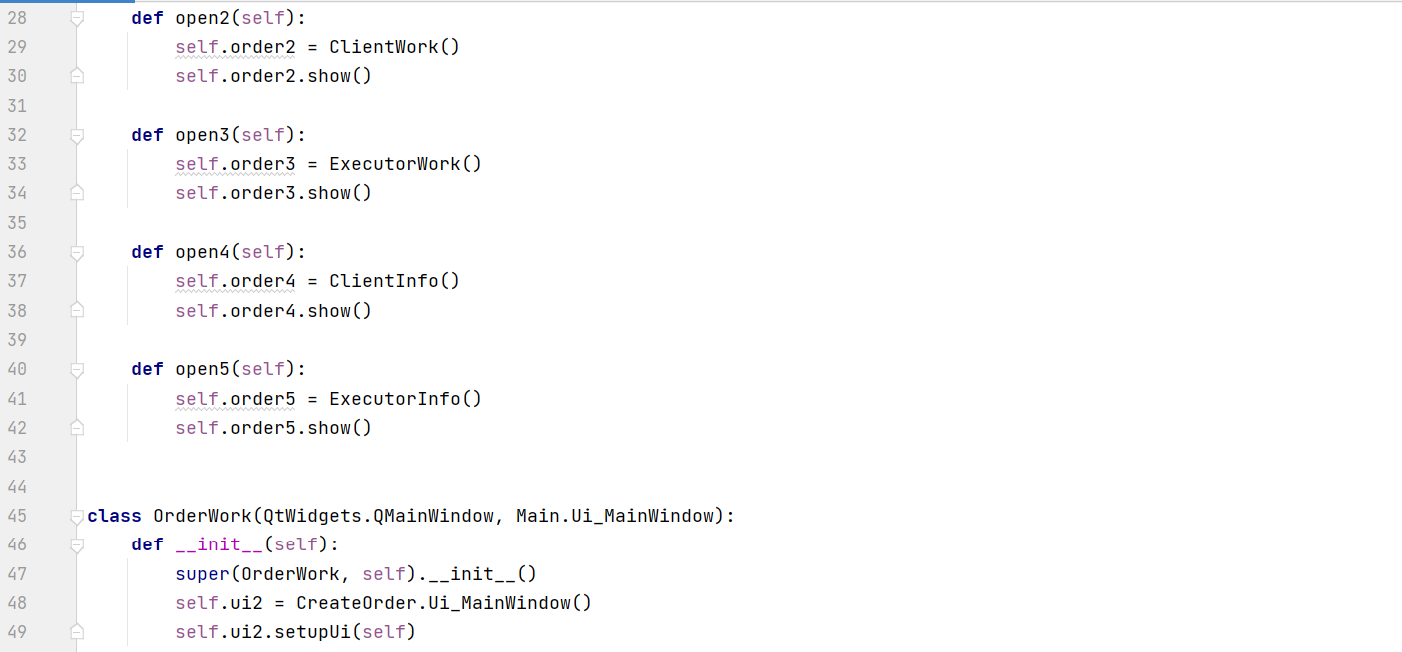
URL: <https://docs.python.org/3/library/sqlite3.html>

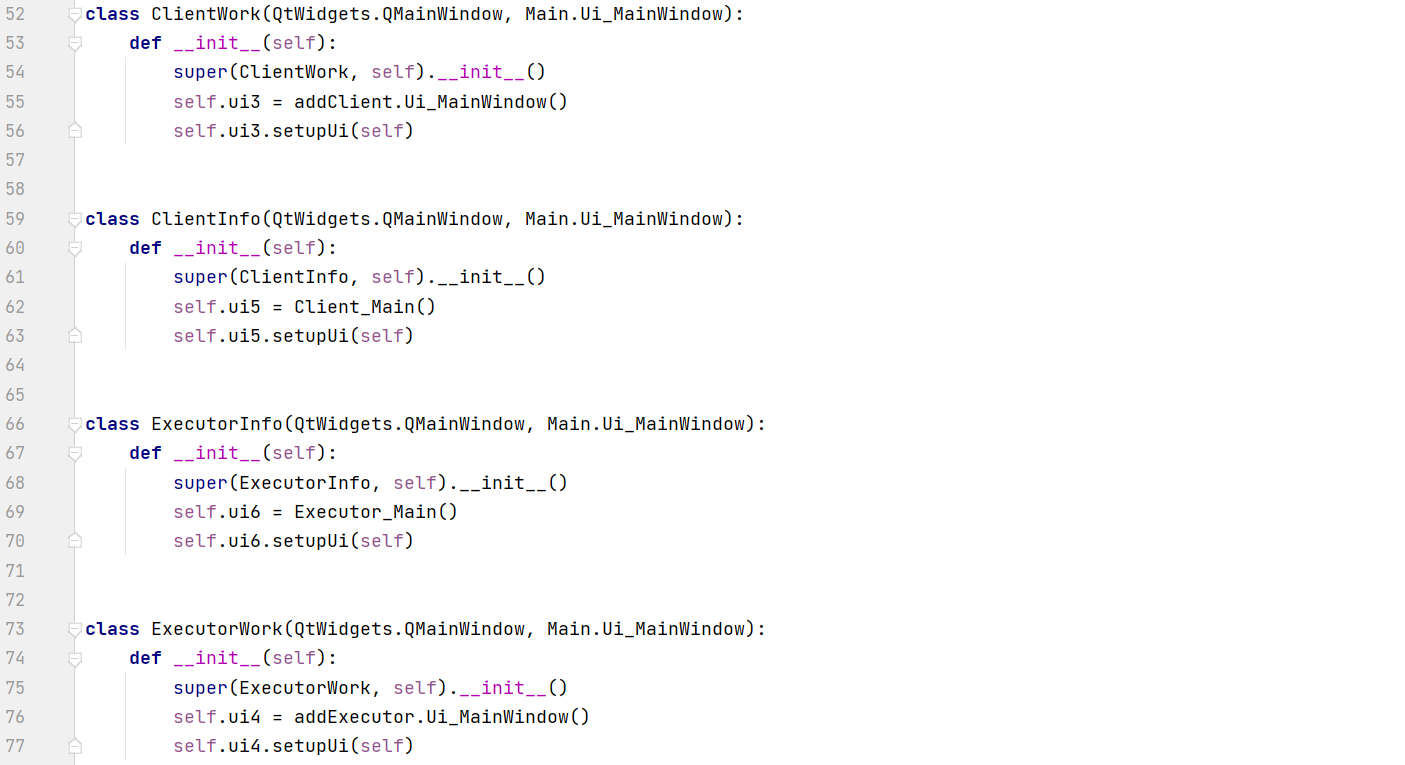
1. Объектно-ориентированное Программирование в Python [Электронный ресурс] // Неофициальный русифицированный сайт Python. URL: <https://python-scripts.com/object-oriented-programming-in-python>
2. Реляционная База Данных. Документация Oracle [Электронный ресурс] // Официальный сайт Oracle. URL: <https://www.oracle.com/ru/database/what-is-a-relational-database/>
3. Руководство по PyQt5 [Электронный ресурс] // Неофициальный русифицированный сайт по Python. URL: <https://python-scripts.com/pyqt5>
4. Руководство по Qt Designer [Электронный ресурс] // Официальный сайт Qt Designer. URL: <http://doc.crossplatform.ru/qt/4.5.0/designer-manual.html>
5. СУБД SQLite [Электронный ресурс] // Сообщество IT-специалистов. URL: <https://habr.com/ru/post/149356/>
6. Техническая документация по SQL Server [Электронный ресурс] // URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15>
7. Учимся проектированию Entity Relationship — диаграмм [Электронный ресурс] // Сообщество IT-специалистов «Хабр». 16 февраля 2019 г. URL: <https://habr.com/ru/post/440556/>

# **ПРИЛОЖЕНИЕ**

***main3.py***









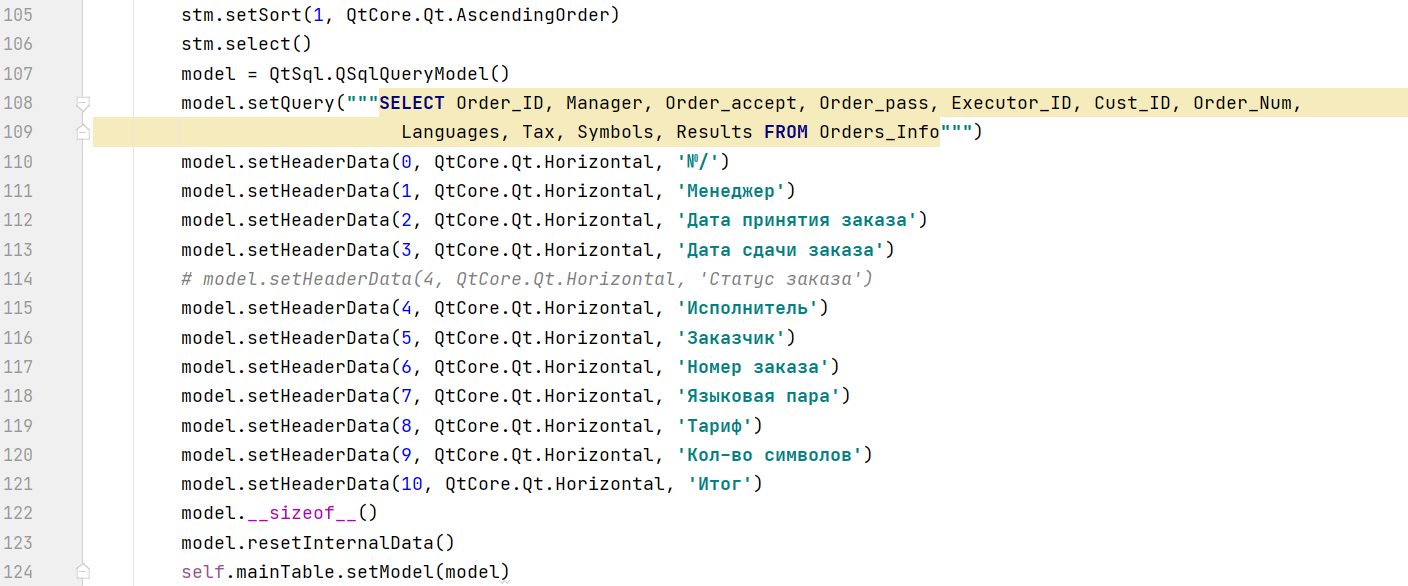
***Main.py***





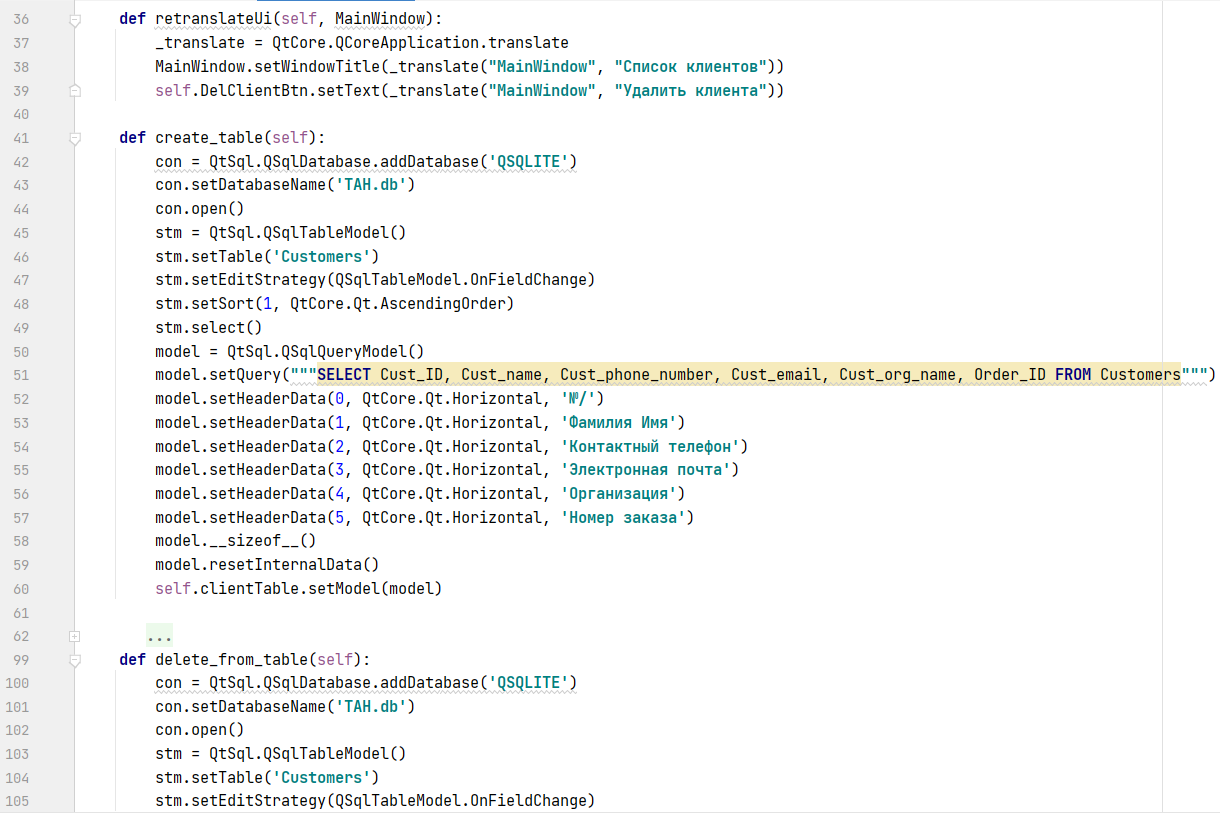


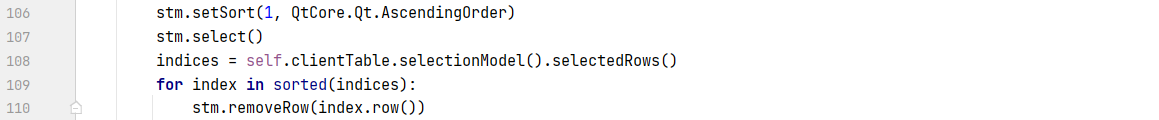




***addClient.py***

******





***addClientInfo.py***

******

******

******

******

***addExecutor.py***

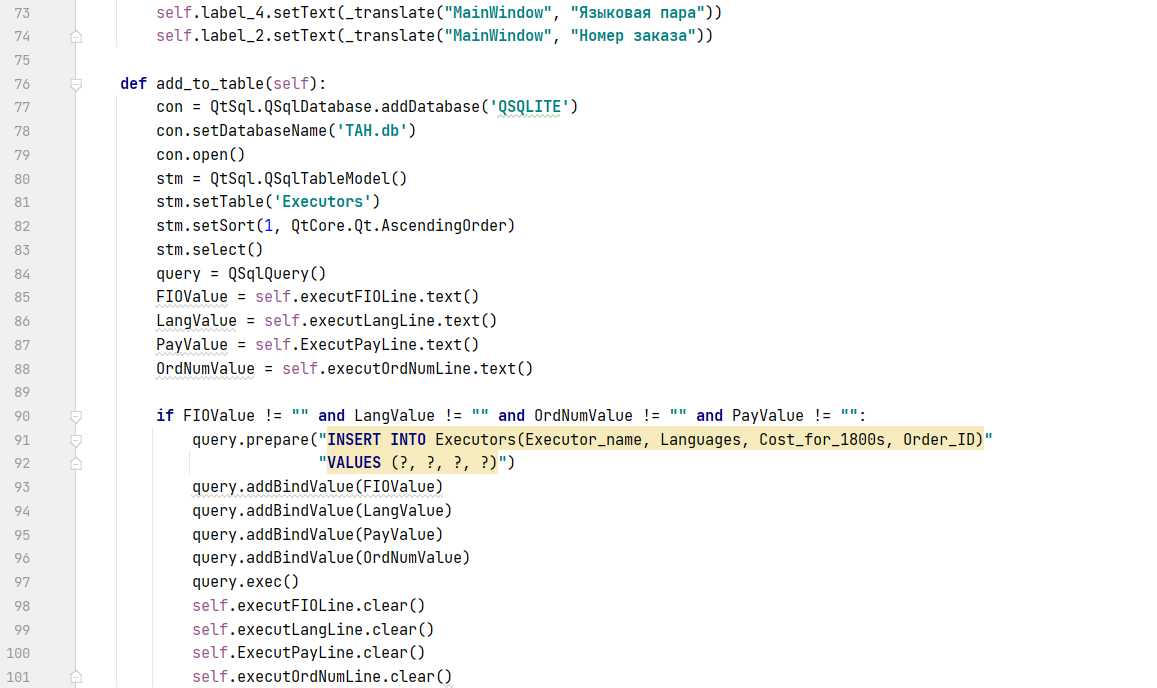
******

******

***addExecutorInfo.py***

******

******

******

***CreateOrder.py***















