Федеральное государственное автономное образовательное учреждение высшего образования

«ОМСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
Кафедра математических методов и информационных технологий в экономике

ИТОГОВЫЙ ПРОЕКТ

по дисциплине Методы и средства проектирования информационных систем и технологий в медиаиндустрии

Пояснительная записка

Выполнили ст. гр. ИСТ 221

_____А.А. Кутузов

_____Л.Р. Пономарев

"____"_____20__г.

Принял

_____О.Н. Канева

"____"_____20__ г.

Омск 2025

ВВЕДЕНИЕ

В современном мире быстро развивающемся мире человеку требуется постоянно обучаться новым навыкам для поддержания своей конкурентоспособности на рынке рабочей силы. Одним из возможных навыков является работа с базами данных.

Базы данных играют ключевую роль в современном мире, где информация является важнейшим ресурсом. Они позволяют хранить, организовывать и управлять данными, обеспечивая доступ к ним для различных приложений и пользователей.

Целью работы является создание программы для управления базой данных троллейбусного депо, во время работы над которой будут получены навыки работы с базами данных и созданию программ.

Задачами работы являются:

– Разработка базы данных;

– Создание интерфейса программы;

– Создание внутренних систем взаимодействия программ с базой данных;

– Получение навыков работы с *Python* и скрепляющими их решениями.

1 Техническое задание

1.1 Архитектура приложения

Для создания и начала работы с базой данных, необходимо определить ее будущую структуру. Логическая модель базы данных для разрабатываемой БД в стандарте *IDEF1X* [1]:

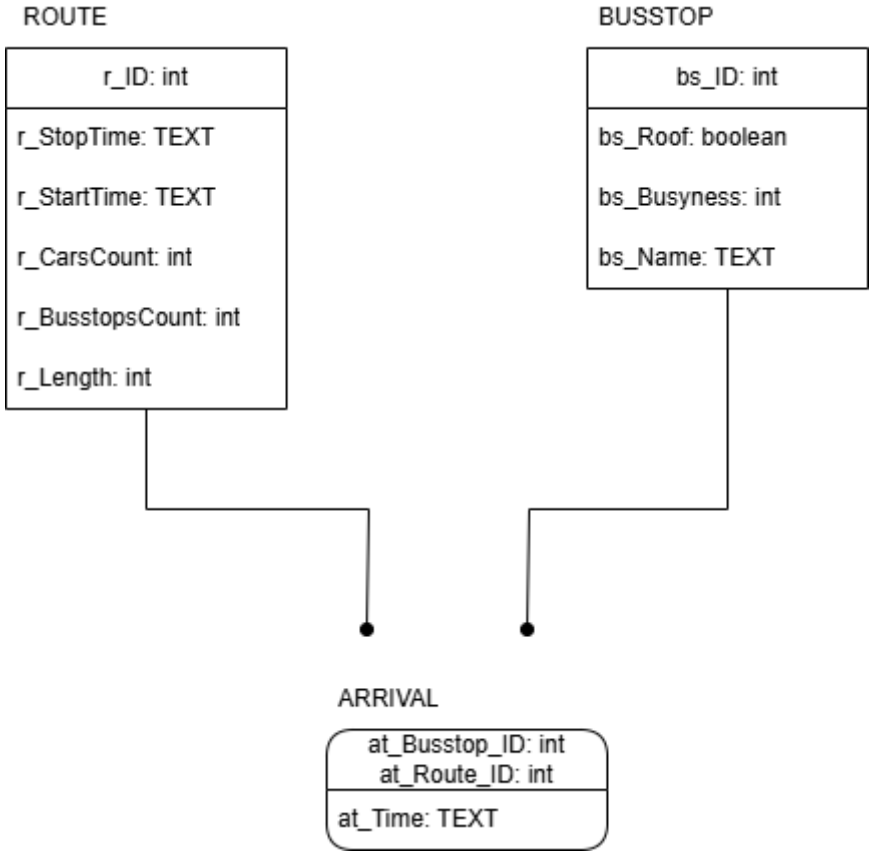

Рисунок 1 – Логическая модель базы данных

При работе с базой данных можно выбрать действие с информацией (добавить, удалить, заменить) и базу данных, с которой предстоит работать (*ROUTE*, *BUSSTOP*, *ARRIVAL*). Выбор происходит посредством нажатия кнопок с необходимым действием. Нажатие кнопок может менять поля, которые предстоит заполнить пользователю. При выборе базы данных также происходит вывод текущей информации в ней.

Также имеется кнопка «Действие», которая изменят базу данных согласно выбранным параметрам (текст на ней зависит от выбранной БД и

действия для неё). При нажатии на неё происходит вывод сообщения с ответом из базы данных и принудительное обновление показа текущей информации в ней.

Существуют два окна: главное и побочное. Главное окно представляет собой стандартный интерфейс программы. Побочное представляет собой диалоговое окно, всплывающее для подтверждения желания удалить информацию из таблицы.

Программа состоит из четырех классов. Для каждого окна программы имеется один функциональный класс и один визуальный. Визуальный класс представляет собой преобразование интерфейса, созданного в программе «*Qt Designer*» в код. Функциональный класс имеет в себе всю логику конкретного окна.

1.2 Дизайн и пользовательский интерфейс

Макет внешнего вида программы продемонстрирован ниже на схеме:

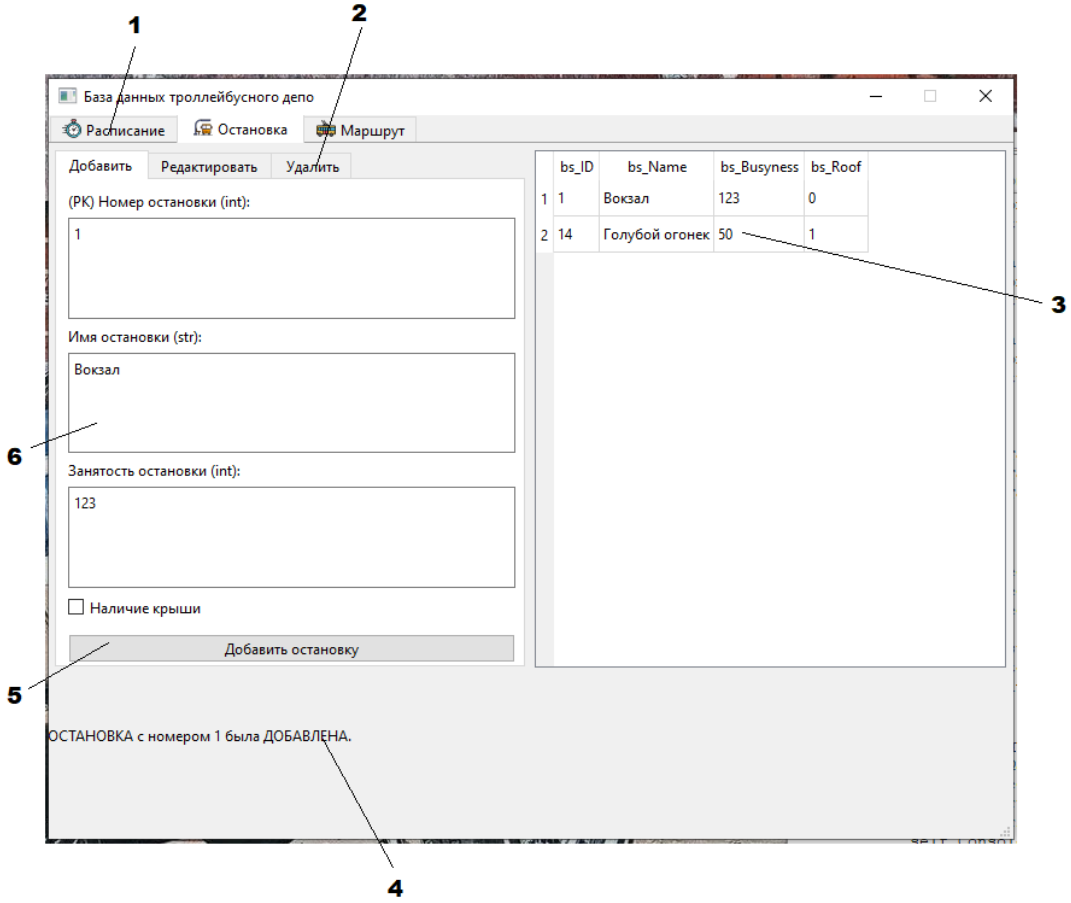

Рисунок 2 – Схема программы

Обозначения на схеме: 1 – кнопки выбора БД; 2 – кнопки выбора действия с текущей БД; 3 – текущие данные в выбранной БД; 4 – информация по поводу последнего сделанного действия; 5 – кнопка действия, нажатие на которую производит попытку изменения БД; 6 – заполняемые поля, необходимые для ввода данных, требующихся для изменения БД.

Ниже приведены примеры интерфейса БД в разных состояниях.

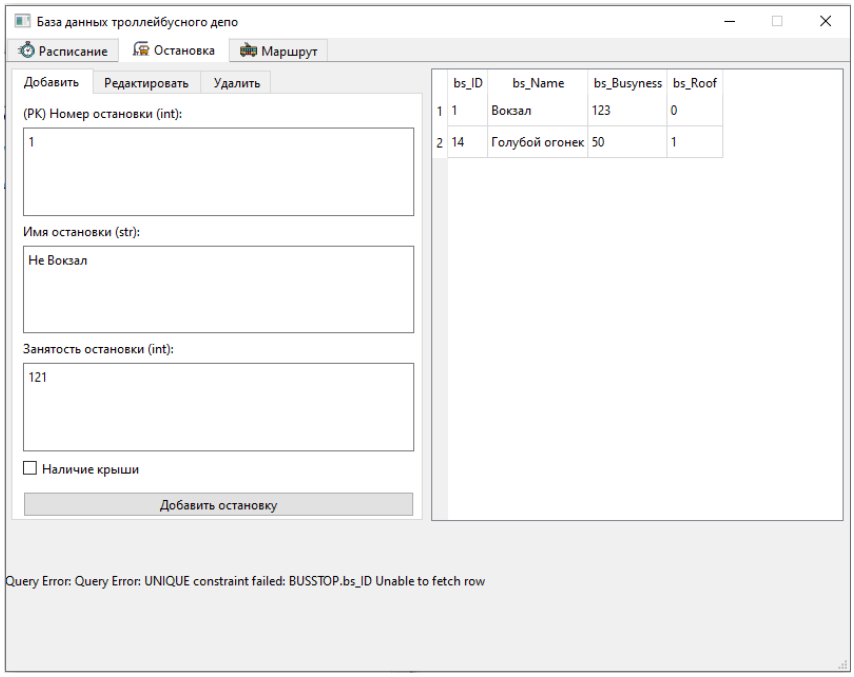

Рисунок 3 – Внешний вид программы (вывод ошибки)

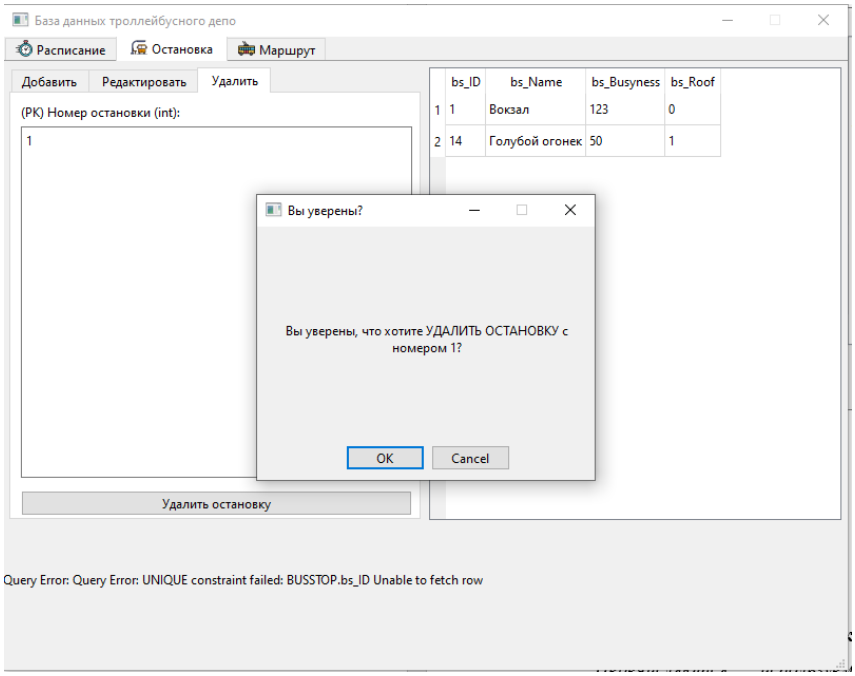

Рисунок 4 – Внешний вид программы (функция удаления)

2 Реализация приложения

2.1 Используемые библиотеки

В программе использовались следующие библиотеки:

– ***PyQt6***

Требовалась для вывода информации на экран в виде интерфейса и работы с базой данных.

PyQt6 – это кроссплатформенный фреймворк для разработки графических интерфейсов пользователя (*GUI*) на языке программирования *Python*. Он является привязкой к *Qt*, мощной и зрелой библиотеке *C++*, предоставляющей широкий набор инструментов и компонентов для создания разнообразных приложений, от простых настольных приложений до сложных профессиональных программ. Предлагает широкий набор инструментов и компонентов, позволяющих создавать сложные и интерактивные приложения для различных платформ [2].

– ***PyInstaller***

Требовался для создания .exe-файла приложения.

PyInstaller – это инструмент *Python*, позволяющий упаковывать *Python*-приложения в отдельные, автономные исполняемые файлы для различных операционных систем (*Windows, macOS, Linux*). Он упрощает распространение приложений, избавляя пользователей от необходимости устанавливать *Python* и зависимости скрипта [3].

2.2 Описание реализации

Приложение запускается через класс «*MainWindow*».

В конструкторе происходит инициализация: подгружается визуальная часть из сопутствующего класса, назначаются функции для кнопок действия, создается база данных и настраивается обновление таблиц.

База данных создается на основе «*QSQLITE for SQLite*» [4]. Название «*Depot*». В функции создания базы данных также создаются заполняемые таблицы: *ARRIVAL*, *BUSSTOP* и *ROUTE*.

Основными функциями, которые позволяют взаимодействовать с таблицами в БД, являются «*Execute*» и «*ExecuteWithPrepare*». Первая принимает на вход *SQL*-команду, которую сразу же отправляет. Вторая принимает неполную *SQL*-команду и массив аргументов, на основе которых создает полную команду с последующей отправкой. Обе функции вызывают ошибку в случае обнаружения ошибки в получившейся команде [5].

Функции настраивания обновления таблиц представляют собой отправку имеющихся таблиц в соответствующие им виджеты «*TableView*». Благодаря использованию *QtSql* данное действие можно сделать одной строчкой.

У всех кнопок имеются следующие общие особенности:

– Используется «*try*» с «*except*», что позволяет отслеживать ошибки.

– В конце действия всегда обновляется прикрепленная таблица.

– В конце действия всегда показывается сообщение об обратной связи.

Разберем особенности функционирования кнопок.

1) Кнопка «Добавление». Для её использования требуется заполнить несколько полей. Большая их часть преобразовывается в *INT*, у некоторых меняется формат на «*HH*:*mm*».

После нажатия кнопки происходит отправка *SQL* команды *INSERT* в *ExecuteWithPrepare* вместе с информацией с полей. В случае неправильного заполнения полей или в случае дублирования главного ключа БД выводится ошибка и действие откатывается.

2) Кнопка «Изменение маршрута». Для её использования требуется такое же количество полей, что и для «Добавление».

После нажатия кнопки происходит поиск выбранного ключа в таблице через *SELECT*. Дело в том, что используемая библиотека «*SQLITE*», не может выводить ошибку в случае попытки изменения несуществующего объекта. В связи с этим было принято решение добавить проверку, которая бы смогла

вывести данную ошибку. Такое можно было бы исправить переходом на более продвинутую базу данных, где необходимая ошибка выводится автоматически

В случае, если ключ найден, происходит отправка *SQL* команды *UPDATE* в *ExecuteWithPrepare* вместе с информацией с полей. В случае неправильного заполнения полей выводится ошибка.

3) Кнопка «Удаление». Для её использования нужно ввести лишь главные ключи, по которым будет найден удаляющейся объект.

В зависимости от таблицы, происходит следующее:

– В случае с таблицей *ROUTE* или *BUSSTOP* после нажатия кнопки происходит поиск одинаковых ключей в таблице *ARRIVAL* через *SQL* команду *SELECT*. Дело в том, что по оригинальной задумке таблицы *ARRIVAL* и *ROUTE*/*BUSSTOP* должны были быть связаны таким образом, чтобы нельзя было удалить маршрут/остановку, если он связан с каким-либо из времен прибытия. Но используемая библиотека «*SQLITE*» не поддерживает связность таблиц, в связи с чем приходится проверять наличие время прибытия вручную. На более продвинутой базе данных это бы превратилось в автоматическую ошибку.

– В случае с таблицей *ARRIVAL* описанное выше не происходит.

Далее, вне зависимости от таблицы, происходит создание диалогового окна. В конструктор диалогового окна передается: *SQL*-запрос, массив параметров для *SQL*-запроса, текст-предупреждение, текст сообщения об успехе, ссылка на модель (для обновления таблицы). После создания окна происходит его запуск.

В окне на подтверждение удаления параметры конструктора записываются в переменные, обновляется предупреждающий текст в соответствии с переданным фрагментом.

В случае нажатия на кнопку «*OK*», происходит отправка *SQL*-запроса через родителя при помощи функции «*ExecuteWithPrepare*». После обновляется модель и закрывается всплывающее окно.

В случае нажатия на кнопку «*Cancel*» всплывающее окно закрывается.

Диаграмма классов программы приведена ниже.

Рисунок 5 – Диаграмма классов

ЗАКЛЮЧЕНИЕ

В рамках проекта было реализовано приложение для работы с базой данных троллейбусного депо. Для него был реализован интерфейс в «*Qt Designer*», который впоследствии был переведен в скрипт формата «*py*». Были реализованы функции работы с базой данных: добавление, редактирование и удаление – для каждой из трех таблиц: *ARRIVAL*, *BUSSTOP*, *ROUTE*. В связи с недоработкой используемой библиотеки «*QSQLITE*» были добавлены исключения, которые симулируют стандартные для работы с БД ошибки при определенных условиях.

Были получены практические навыки по работе с «*Qt Designer*», библиотекой «*PyQt*», библиотекой «*QSQLITE*». Данный опыт пригодится при разработке будущих приложений и в построении дальнейшей карьеры.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Методика IDEF1X // bigor.bmstu.ru URL: http://bigor.bmstu.ru/?cnt/?prn=y/?doc=190_CAD/7003.mod (дата обращения: 29.03.2025).

2 PyQt // Python Wiki URL: https://wiki.python.org/moin/PyQt (дата обращения: 28.05.25).

3 PyInstaller Manual // PyInstaller 6.13.0 documentation URL: https://pyinstaller.org/en/stable/index.html (дата обращения: 28.05.25).

4 SQL Database Drivers // Qt SQL URL: https://doc.qt.io/qt-6/sql-driver.html (дата обращения: 28.05.25).

5 Introduction to SQL Commands // Akamai Cloud URL: https://www.linode.com/docs/guides/sql-commands/ (дата обращения: 28.05.25)