МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики» (МТУСИ)**

Кафедра «Системное программирование»

**ОТЧЕТ**

по лабораторной работе № 8 дисциплина **«Введение в информационные технологии»**

Выполнил: Студент гр.БВТ2207

Кузьмин И.А.

“ ” 2023г.

Проверил:

“ ” 2023г.

Москва, 2023

# Введение

# Цель работы:

# Задание:

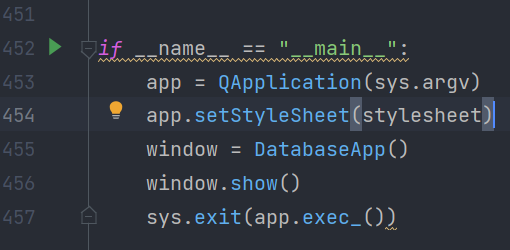
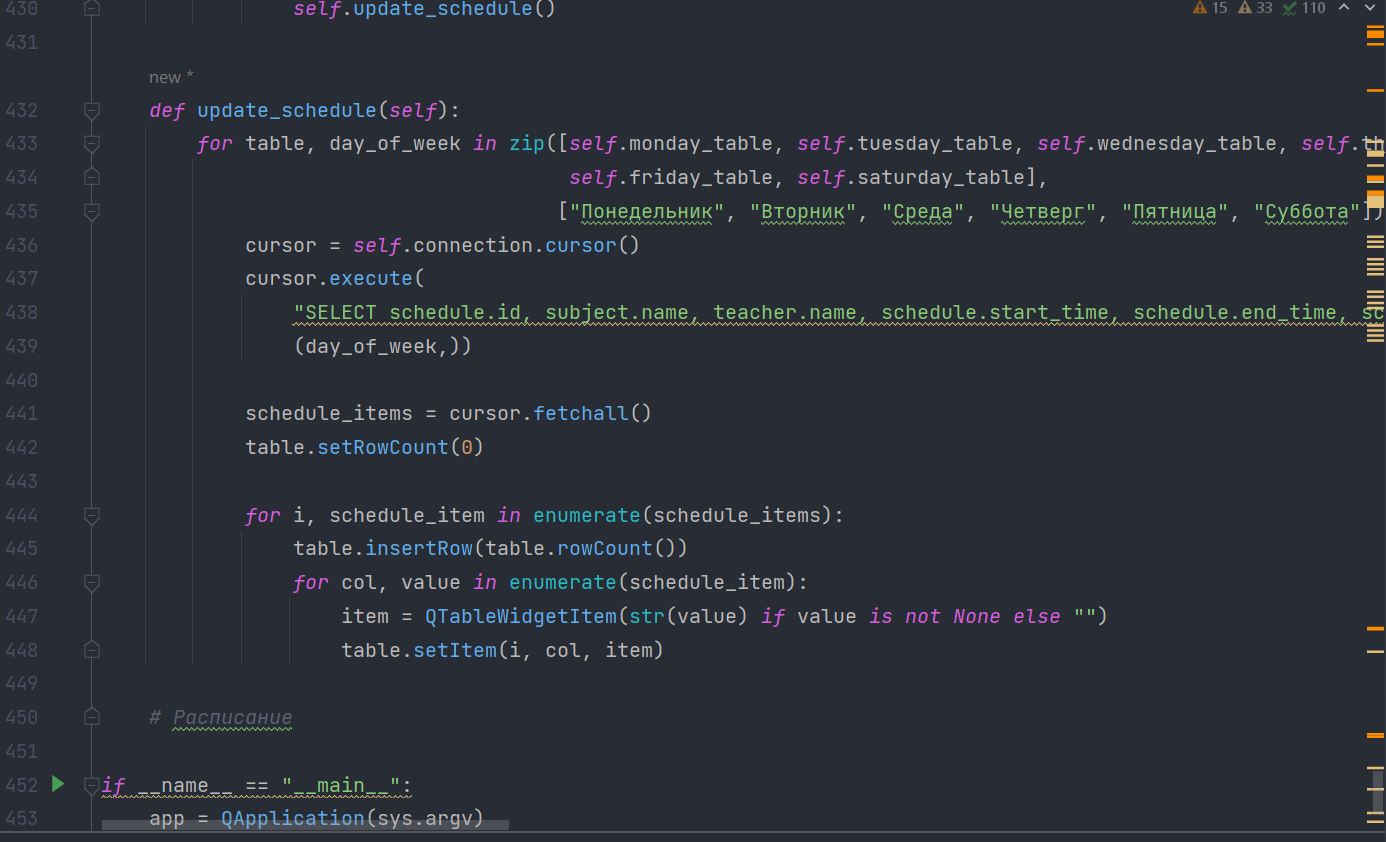
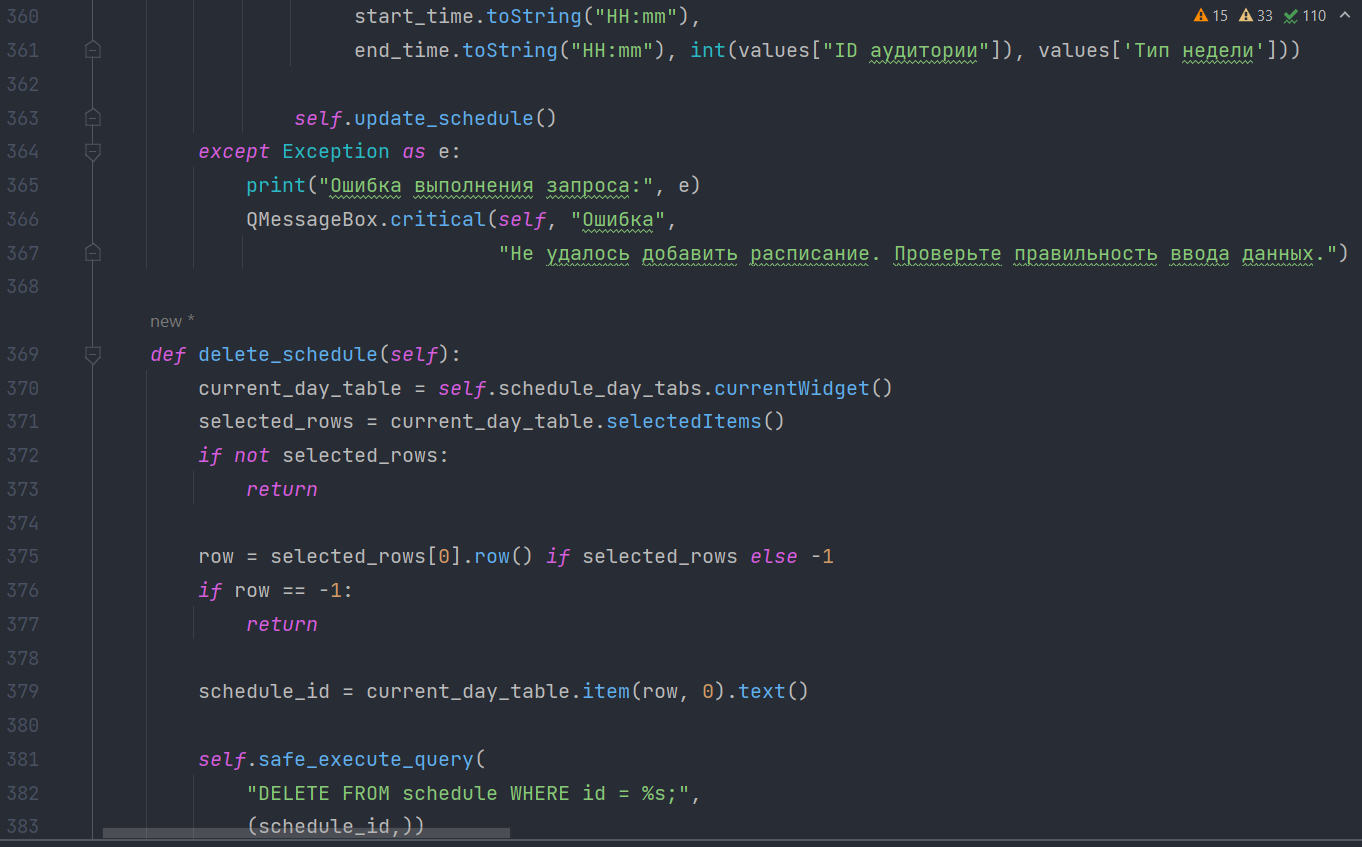
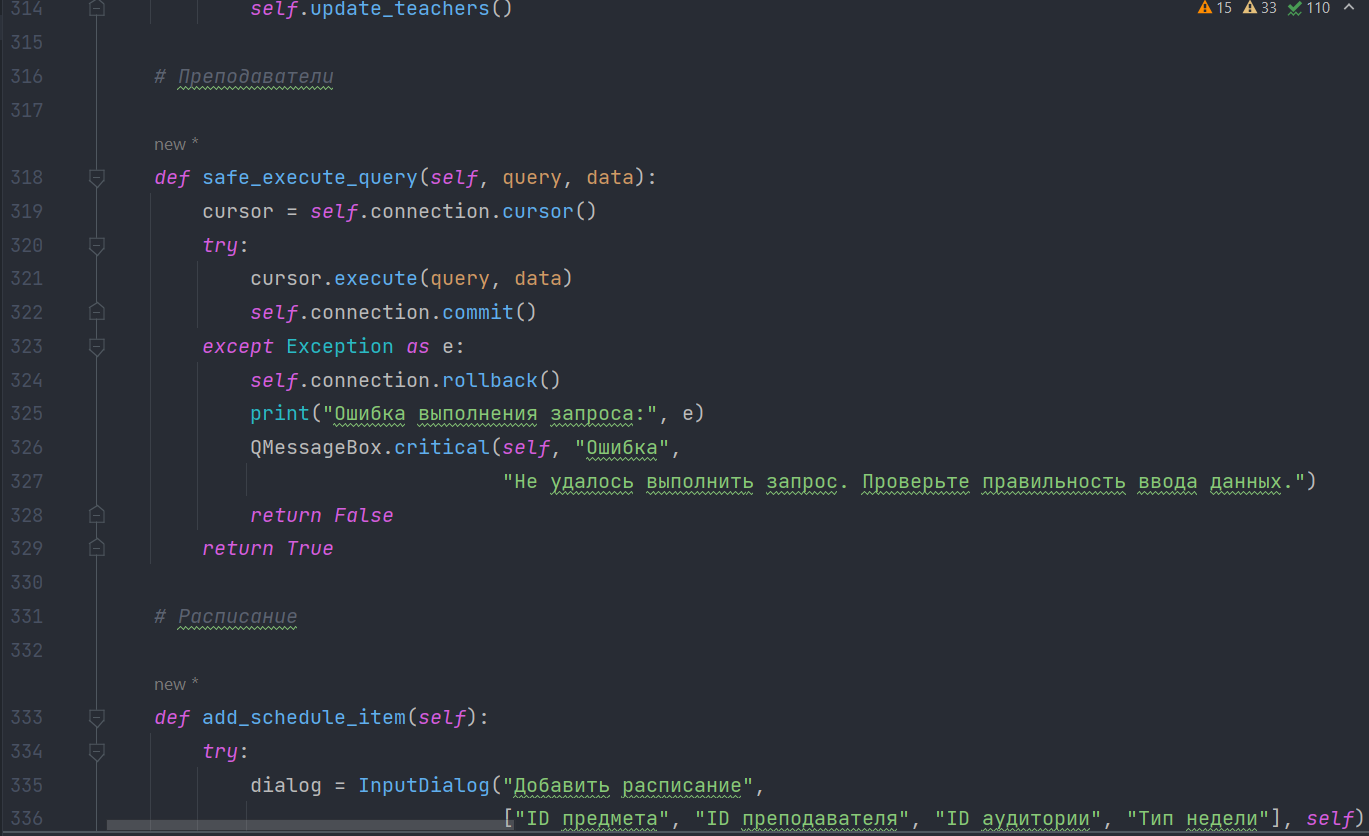
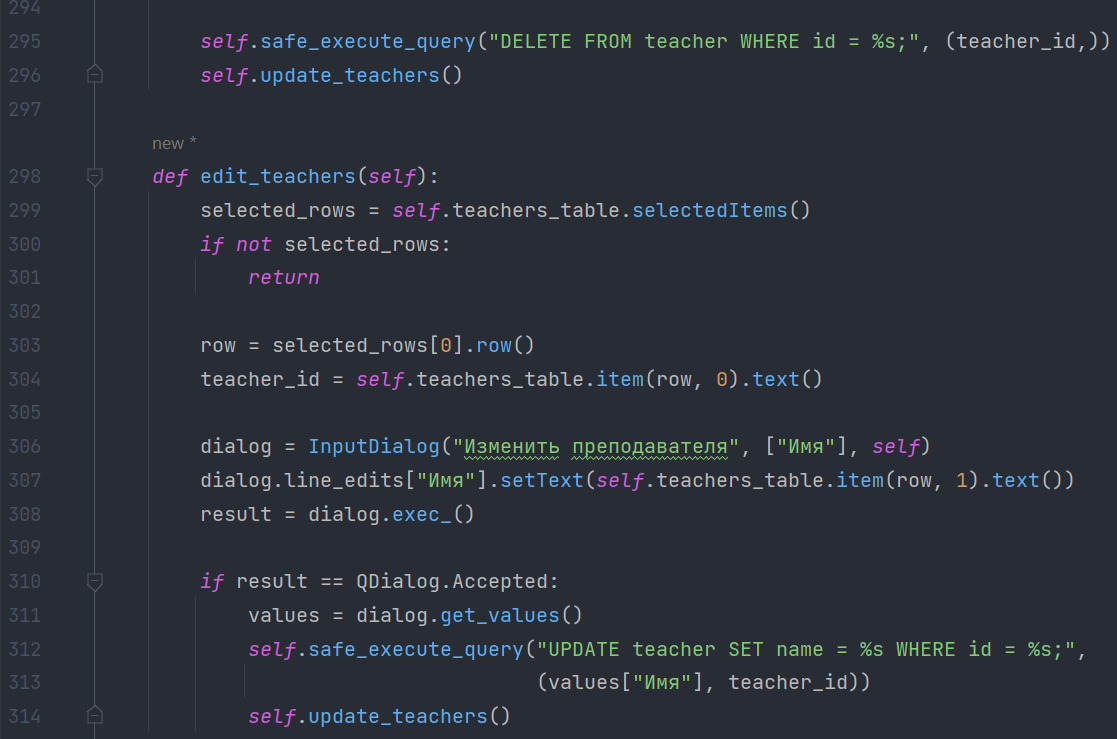
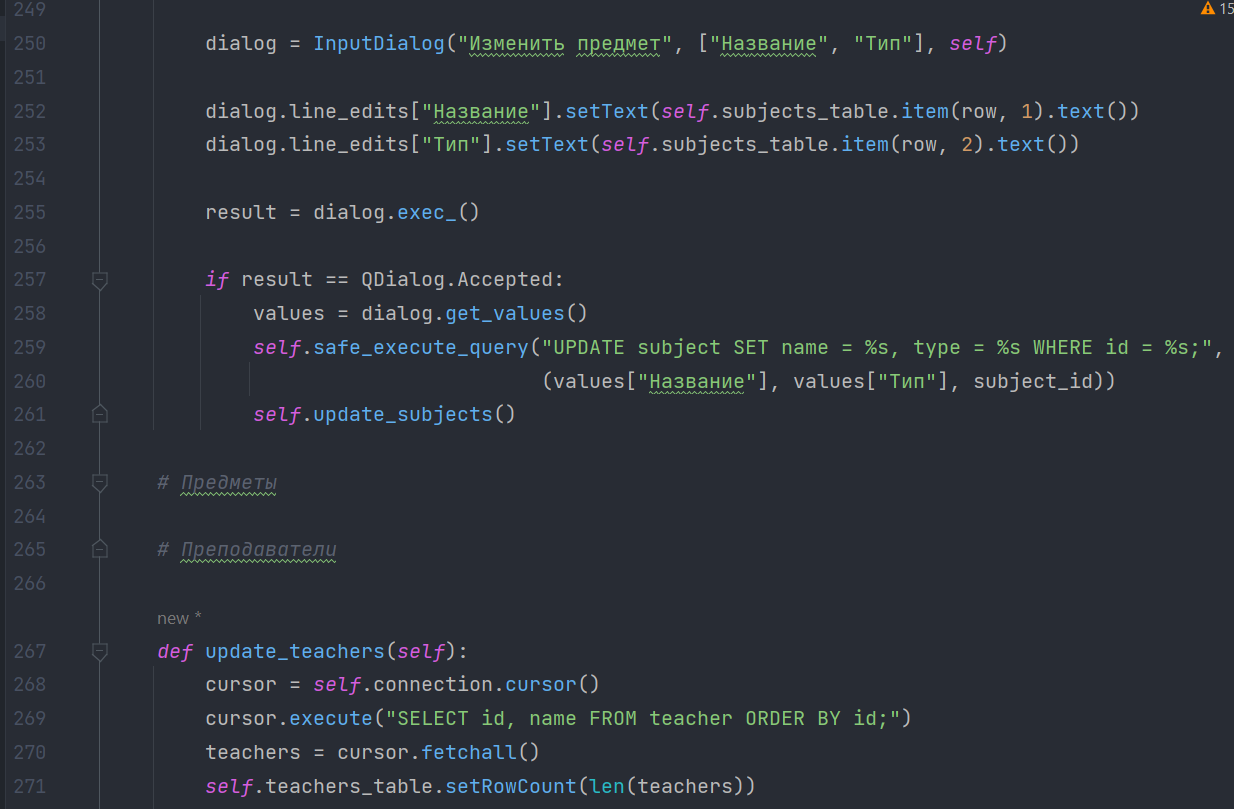
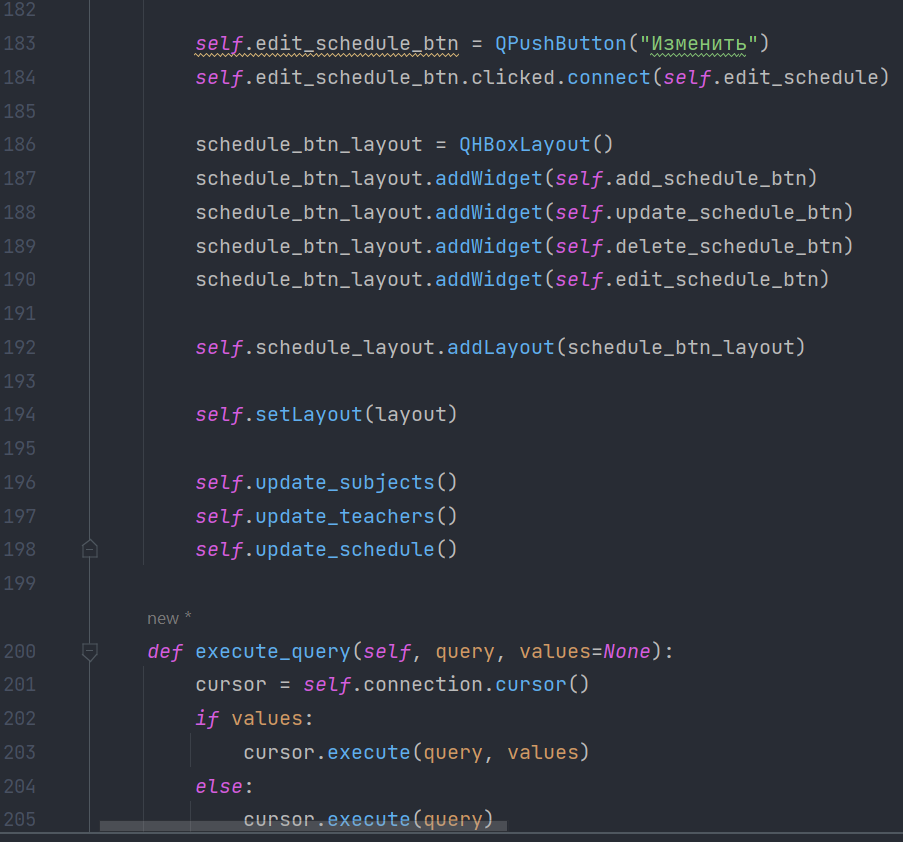
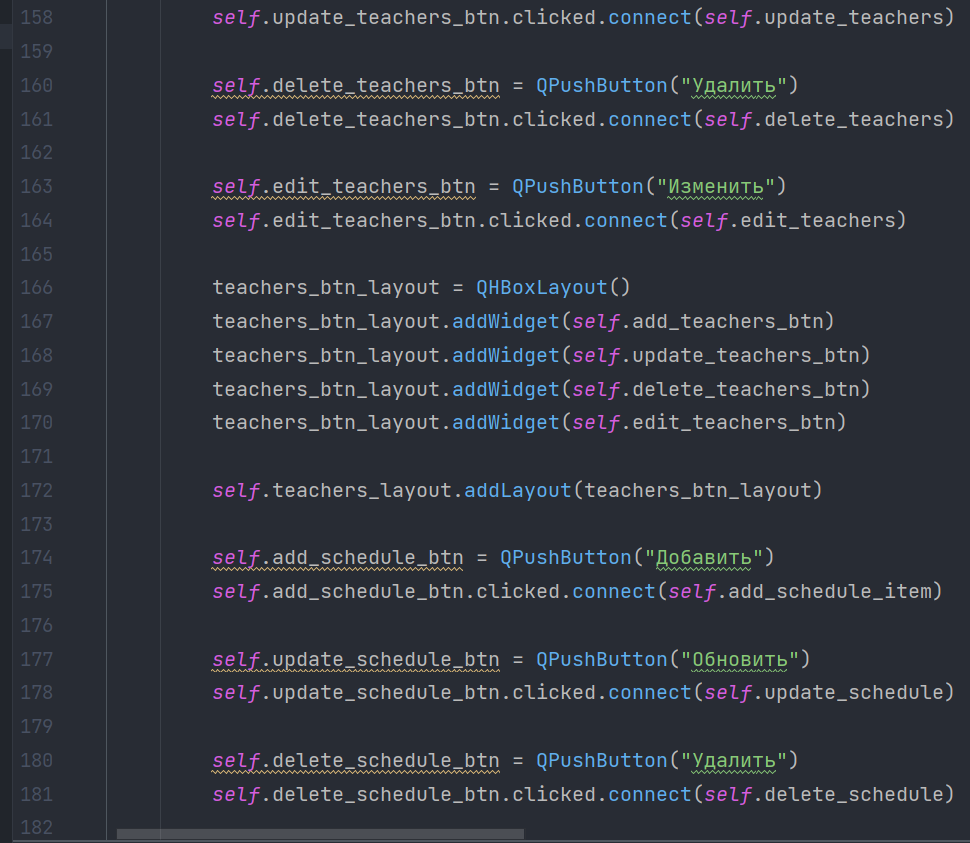
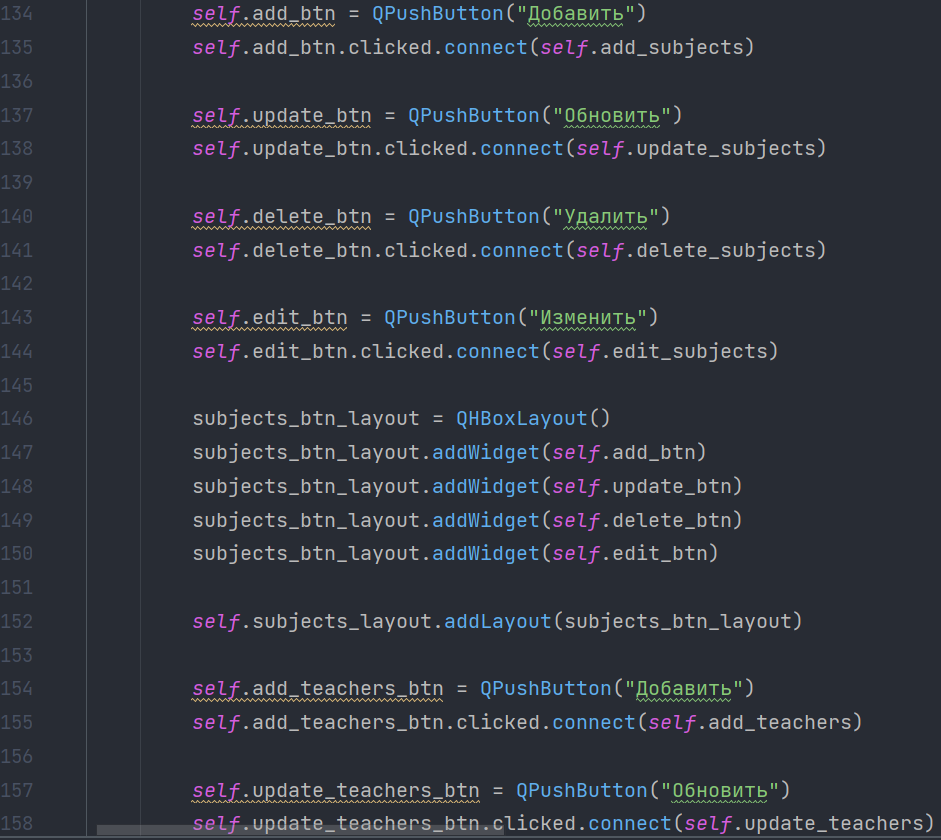
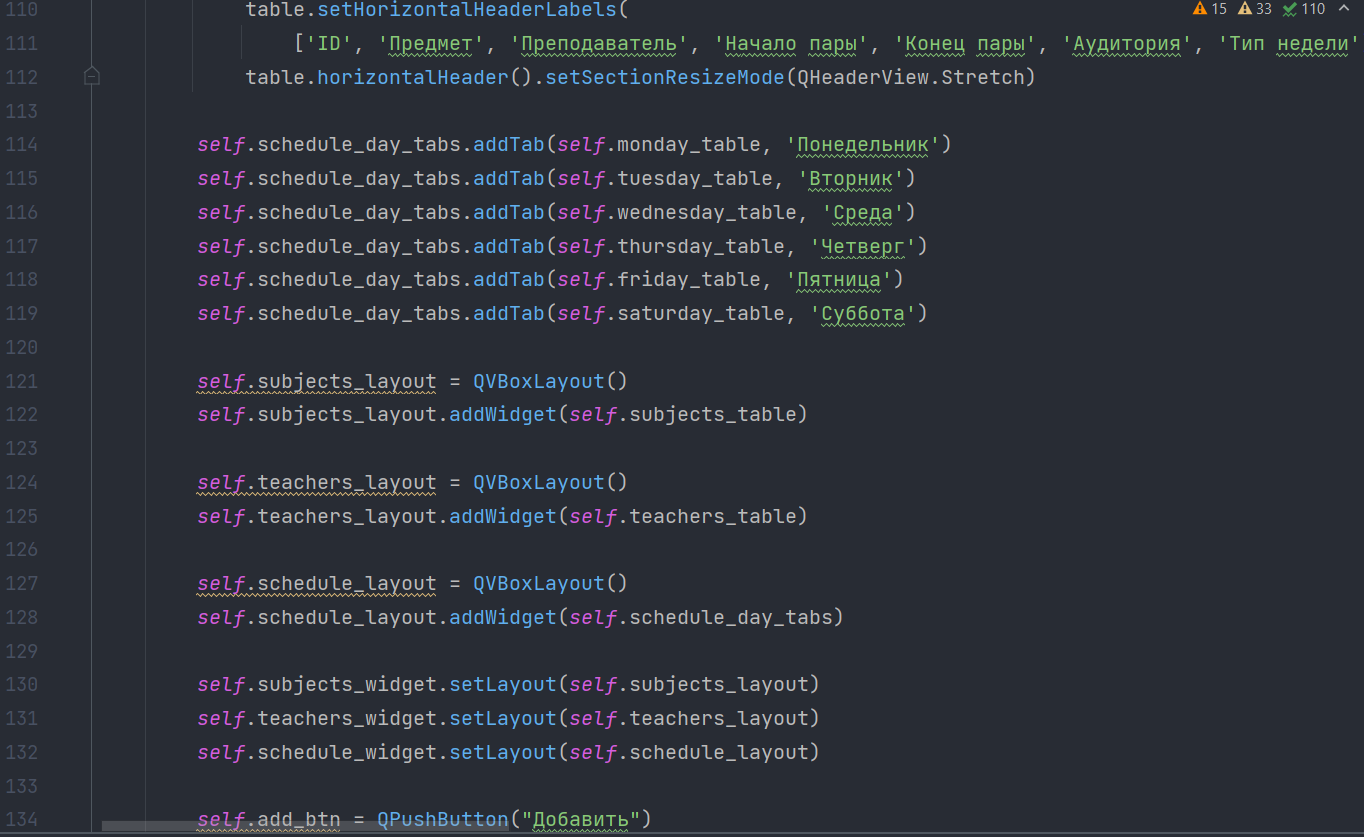
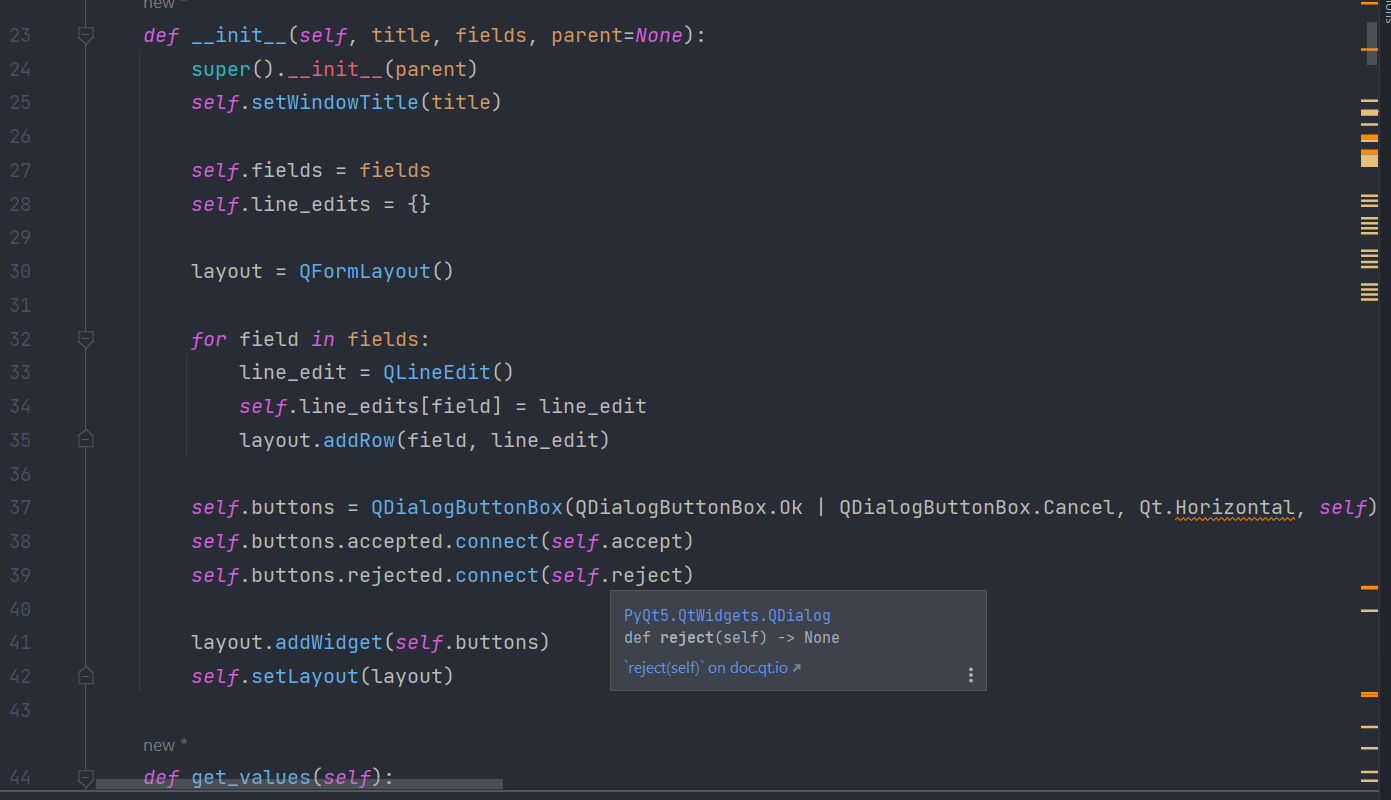
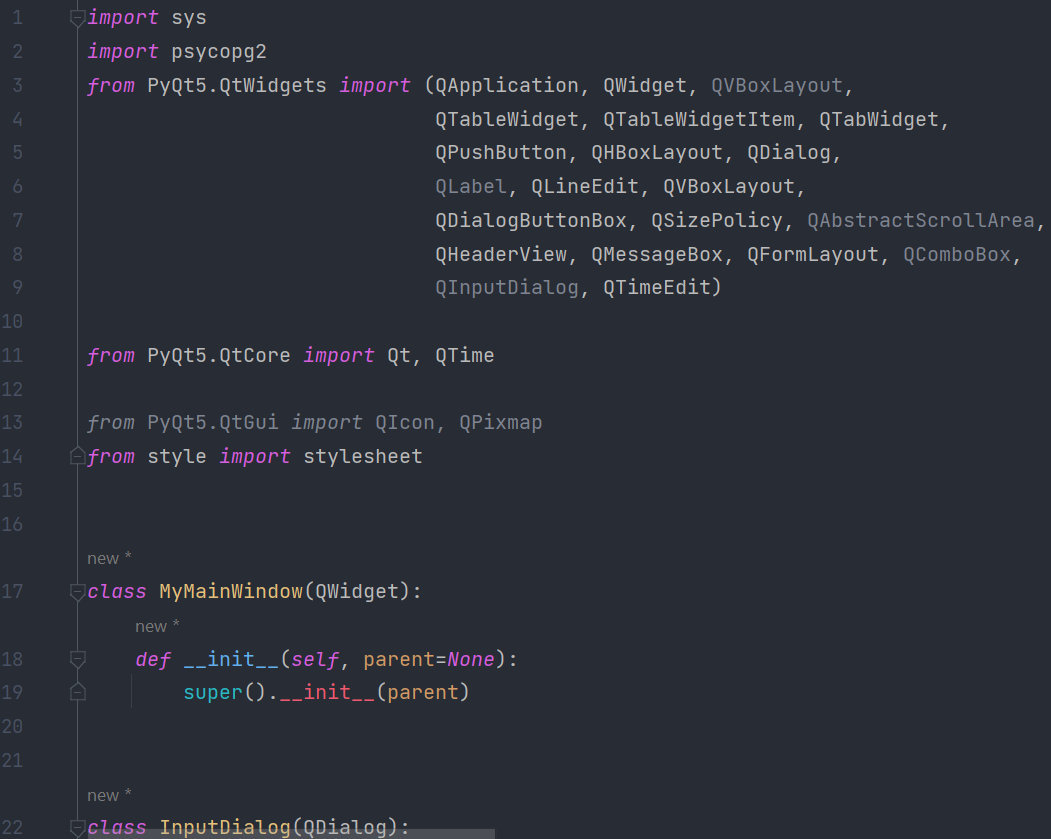
Написать программу, которая будет выводить графический интерфейс, внутри которого можно будет менять данные БД.

.

# Реализация

В процессе выполнения лабораторной работы использованы следующие информационные источники [1-2].

**Практическая реализация**

****

Данный код представляет собой программу на языке Python, использующую библиотеку PyQt5 для создания графического интерфейса пользователя (GUI). Он также использует библиотеку psycopg2 для взаимодействия с базой данных PostgreSQL.

Давайте рассмотрим основные компоненты кода:

1. Импорты:

- `sys`: модуль системных операций.

- `psycopg2`: модуль для работы с базой данных PostgreSQL.

- `QApplication`, `QWidget`, `QVBoxLayout`, `QTableWidget`, `QTableWidgetItem`, `QTabWidget`, `QPushButton`, `QHBoxLayout`, `QDialog`, `QLabel`, `QLineEdit`, `QDialogButtonBox`, `QSizePolicy`, `QAbstractScrollArea`, `QHeaderView`, `QMessageBox`, `QFormLayout`, `QComboBox`, `QInputDialog`, `QTimeEdit`: классы и виджеты библиотеки PyQt5 для создания графического интерфейса.

2. `MyMainWindow` класс:

- Пустой класс, наследующийся от `QWidget`, который представляет главное окно приложения. В данный момент он не содержит никакой функциональности.

3. `InputDialog` класс:

- Класс для создания диалогового окна с вводом данных.

- Инициализирует окно с заданным заголовком и полями ввода на основе переданных аргументов.

- Метод `get\_values` возвращает значения, введенные пользователем в поля ввода.

4. `DatabaseApp` класс:

- Класс, представляющий главное окно приложения.

- Инициализирует подключение к базе данных PostgreSQL и создает интерфейс с помощью библиотеки PyQt5.

- Метод `initUI` создает различные виджеты (таблицы, кнопки и т.д.) и размещает их внутри главного окна.

- Метод `execute\_query` выполняет SQL-запросы к базе данных с использованием соединения.

Данный код относится к разделам "Предметы", "Преподаватели" и "Расписание". Давайте рассмотрим эти функции:

1. \*\*Функции, относящиеся к разделу "Предметы"\*\*:

- `update\_subjects`: Получает предметы из базы данных и обновляет таблицу `self.subjects\_table` с информацией о предметах.

- `add\_subjects`: Открывает диалоговое окно `InputDialog`, где пользователь может ввести данные о новом предмете. После подтверждения введенных данных, функция выполняет SQL-запрос для добавления предмета в базу данных и затем вызывает `update\_subjects` для обновления таблицы.

- `delete\_subjects`: Удаляет выбранный предмет из базы данных и обновляет таблицу `self.subjects\_table`.

- `edit\_subjects`: Открывает диалоговое окно `InputDialog` с предварительно заполненными данными выбранного предмета. После внесения изменений и их подтверждения функция выполняет SQL-запрос для обновления предмета в базе данных и затем вызывает `update\_subjects`.

2. \*\*Функции, относящиеся к разделу "Преподаватели"\*\*:

- `update\_teachers`: Получает преподавателей из базы данных и обновляет таблицу `self.teachers\_table` с информацией о преподавателях.

- `add\_teachers`: Открывает диалоговое окно `InputDialog`, где пользователь может ввести данные о новом преподавателе. После подтверждения введенных данных, функция выполняет SQL-запрос для добавления преподавателя в базу данных и затем вызывает `update\_teachers` для обновления таблицы.

- `delete\_teachers`: Удаляет выбранного преподавателя из базы данных и обновляет таблицу `self.teachers\_table`.

- `edit\_teachers`: Открывает диалоговое окно `InputDialog` с предварительно заполненными данными выбранного преподавателя. После внесения изменений и их подтверждения функция выполняет SQL-запрос для обновления преподавателя в базе данных и затем вызывает `update\_teachers`.

3. \*\*Функции, относящиеся к разделу "Расписание"\*\*:

- `add\_schedule\_item`: Открывает диалоговое окно `InputDialog`, где пользователь может ввести данные о новом элементе расписания (пред

мете, преподавателе, аудитории и т.д.). После подтверждения введенных данных, функция выполняет SQL-запрос для добавления элемента расписания в базу данных и затем вызывает `update\_schedule` для обновления таблицы расписания.

- `delete\_schedule`: Удаляет выбранный элемент расписания из базы данных и обновляет таблицу расписания.

- `edit\_schedule`: Открывает диалоговое окно `InputDialog` с предварительно заполненными данными выбранного элемента расписания. После внесения изменений и их подтверждения функция выполняет SQL-запрос для обновления элемента расписания в базе данных и затем вызывает `update\_schedule`.

- `update\_schedule`: Обновляет таблицы расписания для каждого дня недели (`self.monday\_table`, `self.tuesday\_table`, и т.д.) с информацией о расписании из базы данных.

4. \*\*Другие функции\*\*:

- `safe\_execute\_query`: Функция для выполнения SQL-запросов к базе данных с обработкой возможных исключений. Если запрос не может быть выполнен, функция выполняет откат транзакции, выводит сообщение об ошибке и возвращает `False`. В противном случае, возвращает `True`.

5. \*\*Основная часть программы\*\*:

- `if \_\_name\_\_ == "\_\_main\_\_":` обеспечивает запуск кода только при его непосредственном выполнении, а не при импорте.

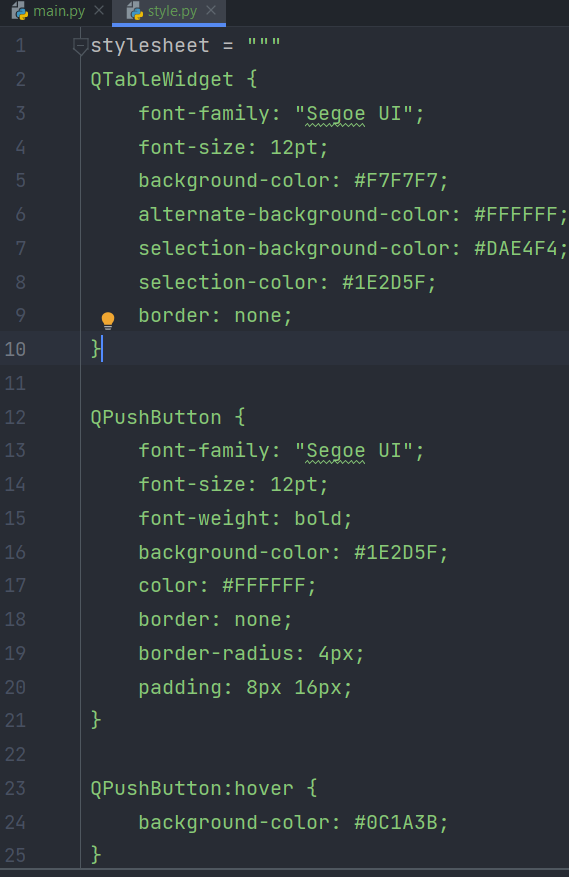
- `app = QApplication(sys.argv)`: Создает экземпляр `QApplication` для создания приложения.

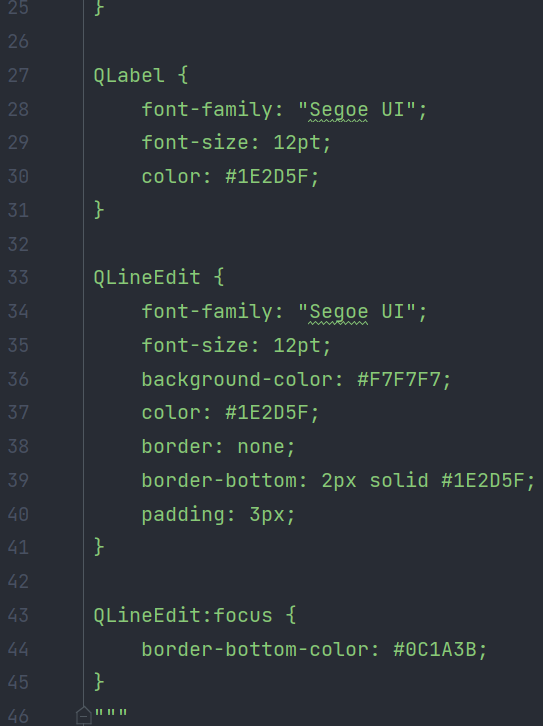
- `app.setStyleSheet(stylesheet)`: Применяет стилевое оформление, заданное в модуле `stylesheet`.

- `window = DatabaseApp()`: Создает экземпляр `DatabaseApp`, представляющий главное окно приложения.

- `window.show()`: Отображает главное окно приложения.

- `sys.exit(app.exec\_())`: Запускает главный цикл приложения и ожидает его завершения.

Далее у нас идёт файл style.py -  


  
  
  
Данный код представляет собой стилевое оформление (CSS-стиль) для виджетов приложения. Он задает определенные свойства и внешний вид виджетов, используемых в графическом интерфейсе приложения.

В стилевом оформлении определены следующие правила:

- `QTableWidget`: Определяет стиль для таблиц виджета `QTableWidget`. Устанавливает шрифт, размер шрифта, цвет фона, цвет выделенной строки и границы.

- `QPushButton`: Определяет стиль для кнопок виджета `QPushButton`. Устанавливает шрифт, размер шрифта, жирность шрифта, цвет фона, цвет текста, границы, радиус скругления и отступы.

- `QPushButton:hover`: Определяет стиль для состояния наведения на кнопку. Устанавливает цвет фона для наведенного состояния.

- `QLabel`: Определяет стиль для виджета `QLabel`. Устанавливает шрифт, размер шрифта и цвет текста.

- `QLineEdit`: Определяет стиль для виджета `QLineEdit`. Устанавливает шрифт, размер шрифта, цвет фона, цвет текста, границы и отступы. Также определяет стиль для фокусированного состояния виджета, изменяя цвет границы.

Это стилевое оформление задает единообразный и согласованный внешний вид для виджетов приложения, используя цвета и шрифты, соответствующие заданным значениям.

# Заключение

# Ссылка на гитхаб со всеми лабораторными работами:

# [FriedCerebrum/Labs (github.com)](https://github.com/FriedCerebrum/Labs)

# Список использованных источников