МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ,

СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамения

Федеральное государственное бюджетное

Образовательно учреждение высшего

Образования

«Московский технический университет связи и

Информатики»

Кафедра «Математическая кибернетика и

Информационные технологии»

Лабораторная работа №8

«Создание визуального интерфейса для базы данных»

Работу выполнил: студент

Группы БВТ2203

Бородин К.Н.

2023

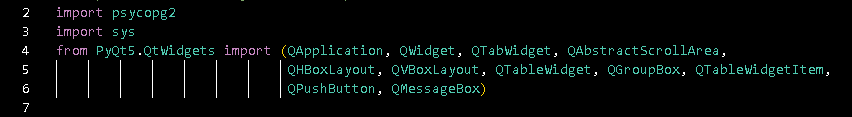
ЗАДАНИЕ.

1. Создать оконное приложение, позволяющее редактировать базу данных с расписанием нашей группы.
2. Использовать библиотеку PyQt5.
3. Использовать адаптер psycopg2.
4. Приложение должно иметь при себе функционал, позволяющий: просматривать базу данных в удобном формате, удалять, добавлять и изменять записи в этой же базе данных.
5. Визуальная часть должна иметь при себе:
6. Минимум три вкладки, в каждой из которых содержится информация из отдельной таблицы в базе данных.
7. Внутри каждой вкладки должны отображаться все поля из таблицы в базе данных в виде колонок.
8. Внутри каждой вкладки должна отображаться кнопка с обновлением информации.
9. Внутри каждой таблицы должны отображаться все поля из таблицы в базе данных в виде колонок.
10. Внутри каждой таблицы после каждой строки записи должны быть отображены кнопки изменения и удаления записи.
11. На вкладки с расписанием дни недели должны быть указаны в отдельных таблицах.

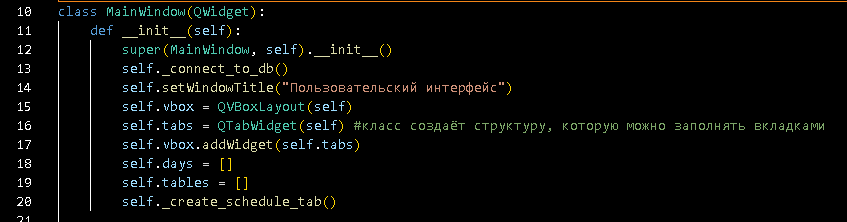
Оглавление

# ХОД РАБОТЫ.

Импортирую необходимые библиотеки, рисунок 1.

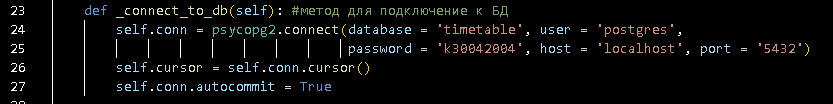
  
Рисунок 1- импорт необходимых библиотек.

Создаю класс MainWindow с конструктором. Также создаю массивы, в которых я буду хранить QWidget для нужных таблиц, а также их характеристики, рисунок 2.

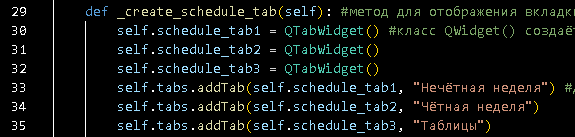
  
Рисунок 2 – создание класса MainWindow с конструктором.

База данных будет использована аналогично той, что была создана в предыдущей лабораторной работе.

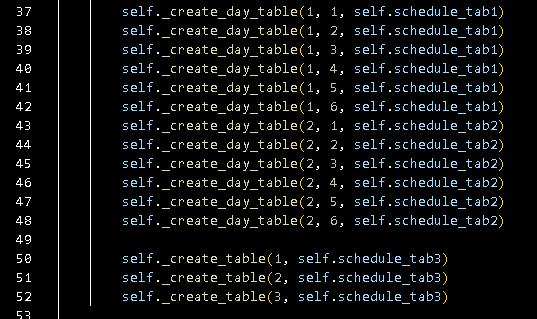
Создаю метод для подключения к базе данных, а также создаю возможность автоматического применения изменения, рисунок 3.

  
Рисунок 3 – создание метода для подключения к базе данных.

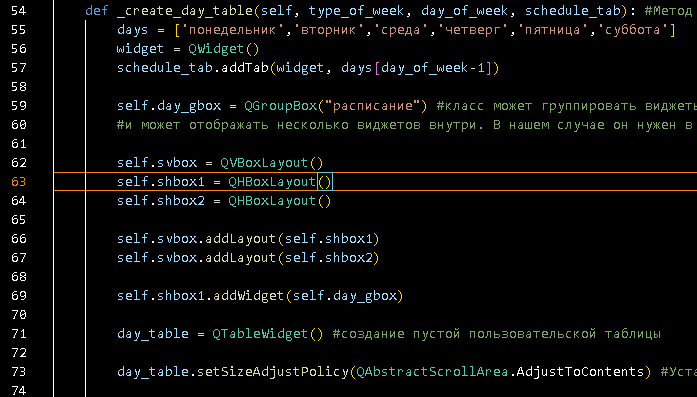
Создаю метод для отображения вкладки с расписание. Создаю необходимые вкладки, рисунок 4.

  
Рисунок 4 – создание метода для отображения вкладок с расписанием.

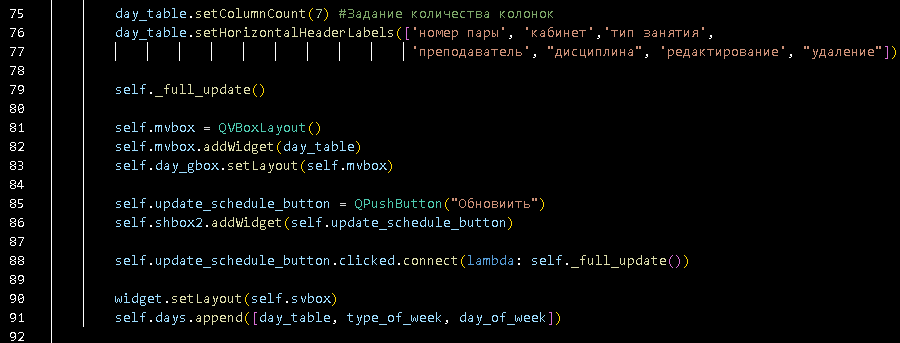
Используя соответствующие методы, создаю вкладки внутри наших вкладок, рисунок 5.

  
Рисунок 5 – создание вкладок внутри вкладок.

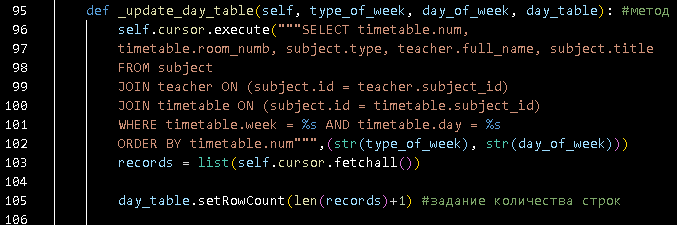
Создаю метод для отображения таблицы с расписанием на день. Добавляю виджеты и регулирую размеры, рисунок 6.

  
Рисунок 6 – метод для отображения расписания на один день.

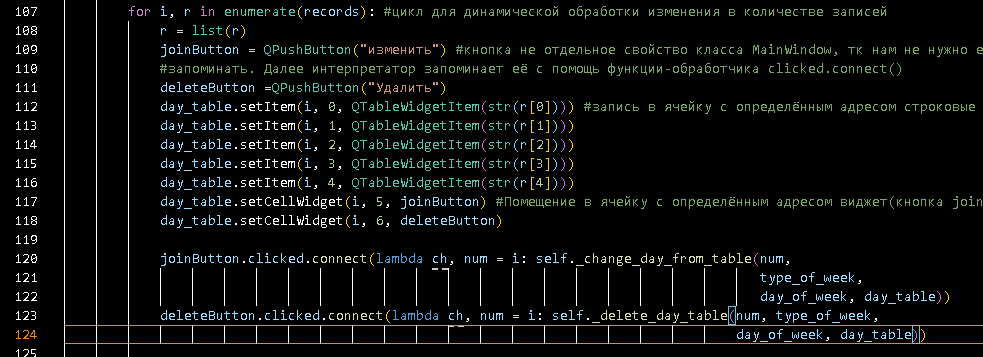
Устанавливаю количество колонок в таблице, присваиваю им имена, обновляю данные. Далее добавляю кнопку обновить и добавляю в массив конфигурацию текущей таблицы, рисунок 7.

  
Рисунок 7 – метод для отображения расписания на один день.

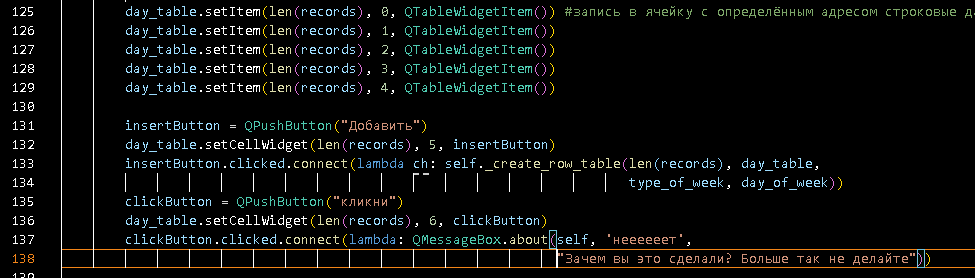
Создаю метод, который обновляет данные на текущий день. Он совершает запрос, устанавливает количество колонок, рисунок 8.

  
Рисунок 8 – метод для обновления расписания на текущий день.

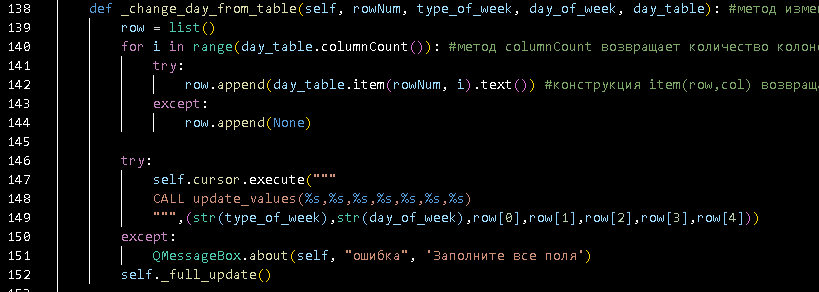
Далее идёт цикл для динамической обработки изменений в количестве записей. Также создаю кнопки изменения и удаления, рисунок 9.

  
Рисунок 9 – метод для обновления расписания на текущий день.

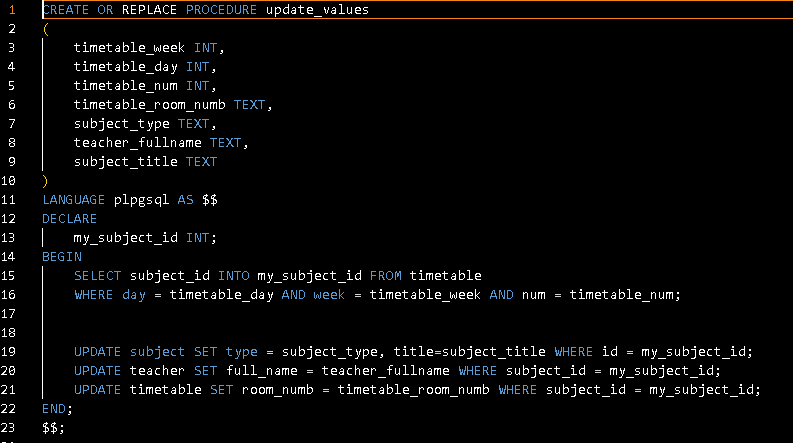
Далее создаю в конце пустую строку, в которой добавляю кнопок добавления и «занятия» пустого пространства, рисунок 10.

  
Рисунок 10 – метод для обновления расписания на день.

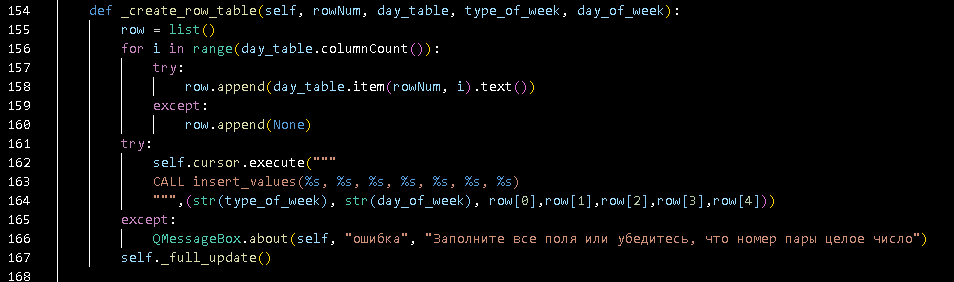
Создаю метод для обновления одной строки в таблице. Метод будет считывать данные, введённые пользователем, а затем на их основе совершать запрос для обновления в базу данных. После этого будет совершаться обновление всех таблиц, рисунок 11.

  
Рисунок 11 – метод для обновления записи в таблице.

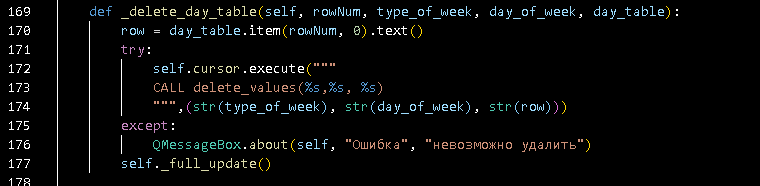
В ходе работы этой функции используется процедура, написанная на языке PostgreSQL. Такая функция используется для каскадного обновления ошибок, чтобы избежать ошибки. Реализация представлена на рисунке 12.

  
Рисунок 12 – функция для каскадного обновления информации.

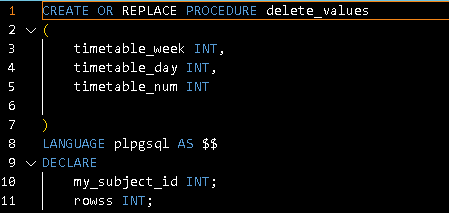
Далее создаю метод для добавления информации в базу данных. Функция будет считывать введённые данные, а затем, используя процедуру из прошлой лабораторной работы, добавлять значения в базу данных, рисунок 13.

  
Рисунок 13 – метод для добавления данных в базе данных.

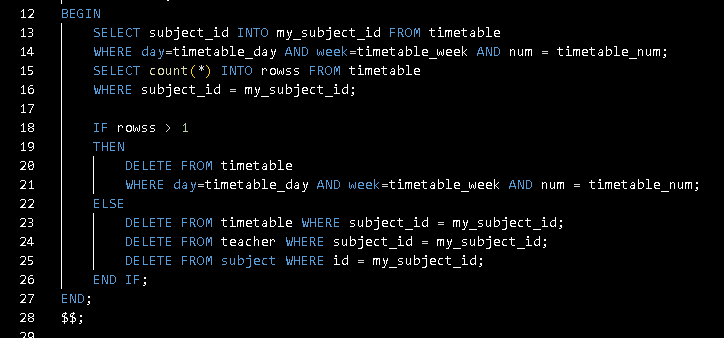
Создаю метод для удаления данных из базы данных, рисунок 14.

  
Рисунок 14 – метод для удаления данных из базы данных.

Для корректного удаления будет создаваться процедура, на языке postgresql. Её создание представлено на рисунке 15.

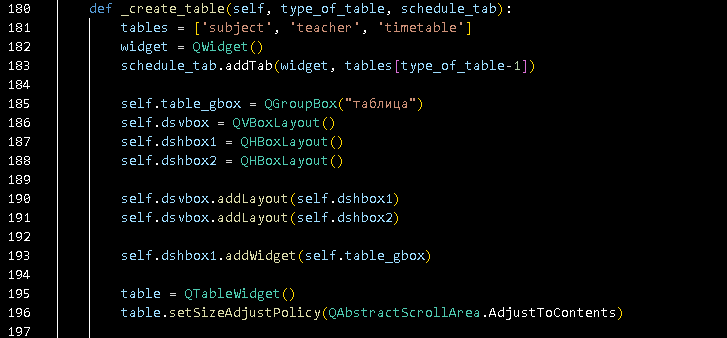
  
Рисунок 15 – создание процедуры для удаления данных.

Далее я пишу команды для удаления при различных условиях, рисунок 16.

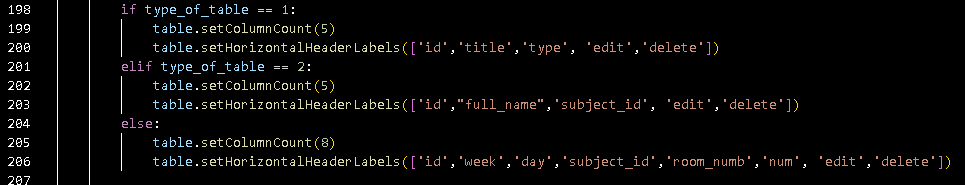
  
Рисунок 16 – тело процедуры.

Теперь аналогичные методы создаю для визуализации таблиц в оконном приложение.

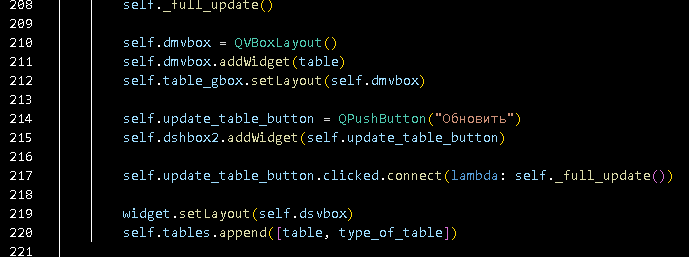
Создаю метод, который будет создавать таблицы, рисунок 17.

  
Рисунок 17 – метод для создания таблиц.

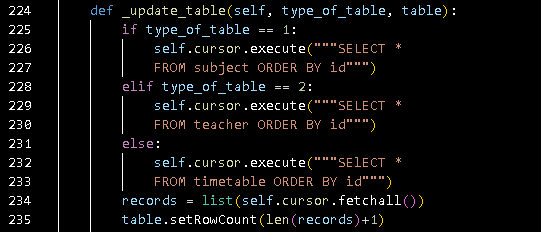
Далее, создаю сами таблицы и даю корректные названия столбцам, рисунок 18.

  
Рисунок 18 – метод для создания таблиц.

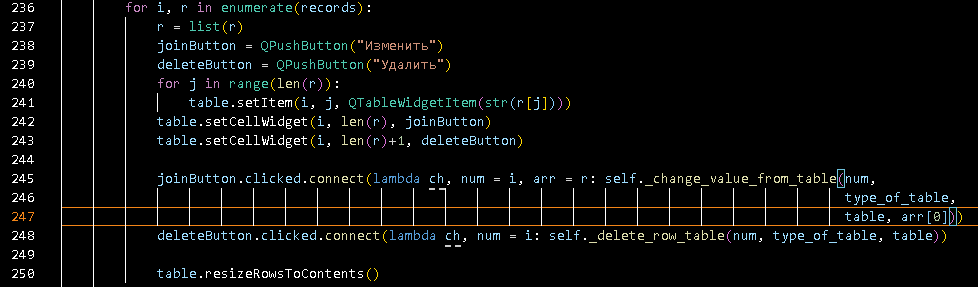
Далее создаю кнопку «обновить» для обновления информации во всех таблицах. Делаю соответствующие выравнивания, рисунок 19.

  
Рисунок 19 – метод для создания таблиц.

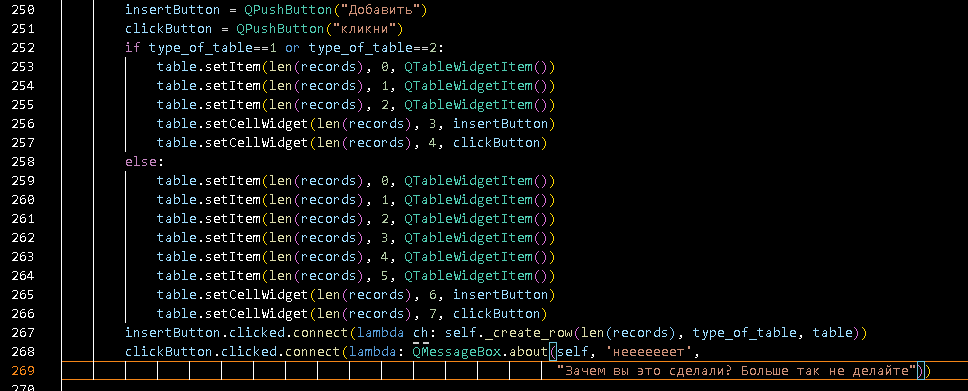
Далее создаю метод для обновления данных в таблицах, рисунок 20.

  
Рисунок 20 – метод для обновления данных в таблице.

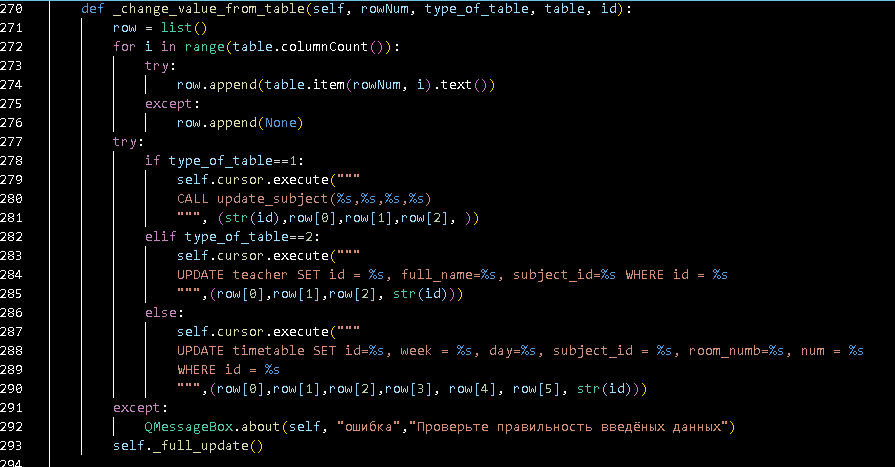
Далее для каждой строки добавляю кнопки изменения и удаления, рисунок 21.

  
Рисунок 21 – метод для обновления данных в таблице.

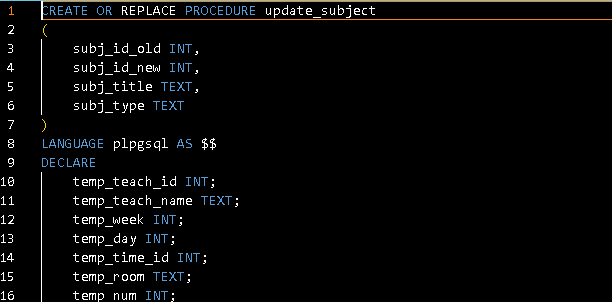
Теперь для каждой строки вношу соответствующие значения. Также добавляю возможность вставлять новые значения: добавляю кнопки в эту строку, рисунок 22.

  
Рисунок 22 – метод для обновления данных в таблице.

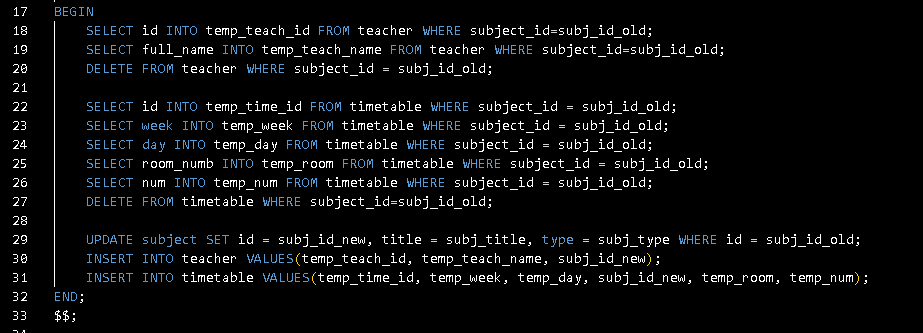
Далее создаю метод для обновления значений. В соответствии с каждым типом таблицы буду делать соответствующий запрос, рисунок 23.

  
Рисунок 23 – метод для обновления данных в таблице.

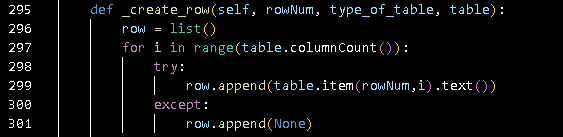
Для корректного обновления данных буду использоваться соответствующую процедуру на языке PostgreSQL. В начале задаю переменные, рисунок 24.

  
Рисунок 24 – создание процедуры на языке PostgreSQL.

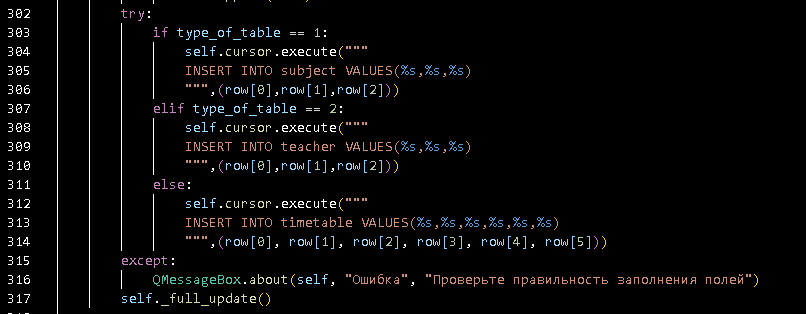
Далее я сохраняю нужные значения в переменных, значения удаляю, а потом их вношу, но с другим айди. Это нужно для корректного каскадного обновления информации, рисунок 25.

  
Рисунок 25 – процедура на языке PostgreSQL.

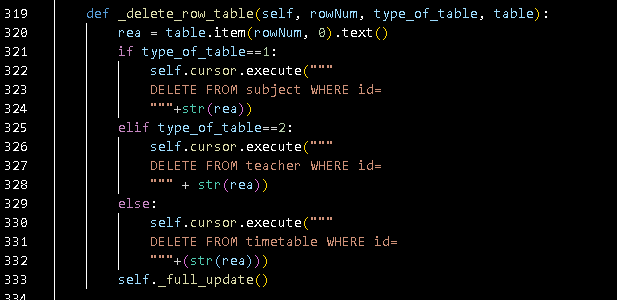
Далее создаю метод для внесения новых данных в базы данных. Сначала сохраняю данные в массив, рисунок 26.

  
Рисунок 26 – метод для внесения данных в базу данных.

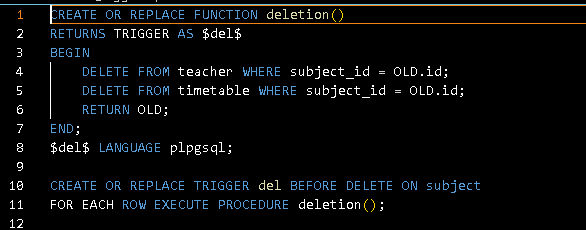
Далее делаю соответствующие запросы, или, в случае их невозможности, вывожу ошибку. После обновляю данных во всех таблицах, рисунок 27.

  
Рисунок 27 – метод для внесения данных в базу данных.

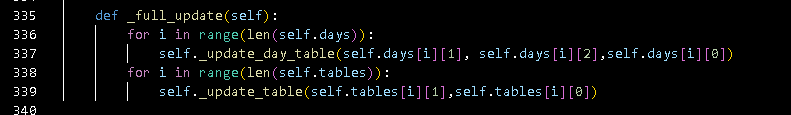
Создаю метод для удаления данных, рисунок 29.

  
Рисунок 29 – метод для обновления данных.

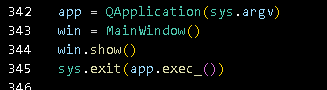
Чтобы совершать каскадное удаление, создаю триггер и триггерную функцию на языке PostgreSQL, рисунок 30.

  
Рисунок 30 – триггер и триггерная функция.

И, наконец, создаю метод для полного обновления данных. Он будет брать данные из массива, созданного в самом начале, и вызывать функции для обновления, рисунок 31.

  
Рисунок 31 – метод для обновления данных во всех таблицах.

Создаю возможность для запуска оконного приложения, рисунок 32.

  
Рисунок 32 – возможность для запуска оконного приложения.

# ЗАКЛЮЧЕНИЕ.

В ходе работы было создано оконное приложение, позволяющее редактировать базу данных с расписанием нашей группы. Были использованы требуемых библиотеки. Приложение имеет функционал, позволяющий: просматривать базу данных в удобной формате, удалять, добавлять и изменять записи в этой же базе данных. Визуальная часть имеет все необходимые характеристики.