**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»**

**Факультет «Сети и системы связи»**

**Лабораторная работа по теме**

**«Расписание»**

Выполнила студентка

Группы БИН2002

Тарыгина Е. М.

Проверил:

Чайка А. Д.

Москва

2021

**Содержание отчета**

[Цель работы 3](#_Toc91097307)

[Выполнение работы 3](#_Toc91097308)

[Результат выполнения 16](#_Toc91097309)

[Вывод 16](#_Toc91097310)

**Цель работы**

Создание оконного приложения с использованием PyQt5.QtWidgets

и PostgreSQL с расписанием для своей группы.

**Выполнение работы**

Создадим проект нашего бота, для чего начнем с базы данных, которая изображена на рисунках 1-2.

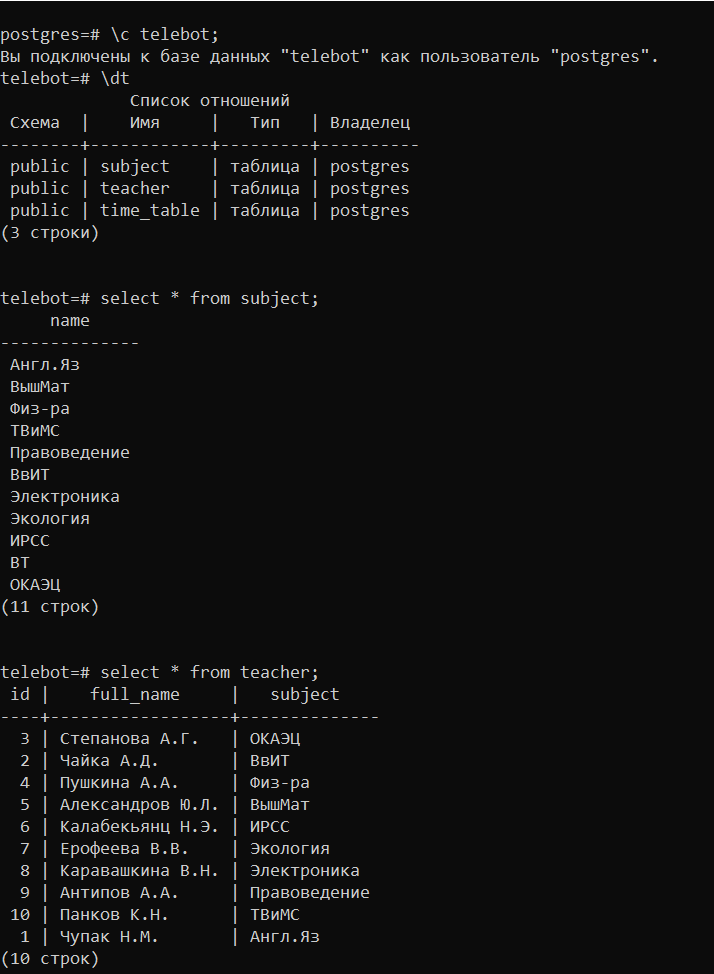


Рисунок 1 – База данных

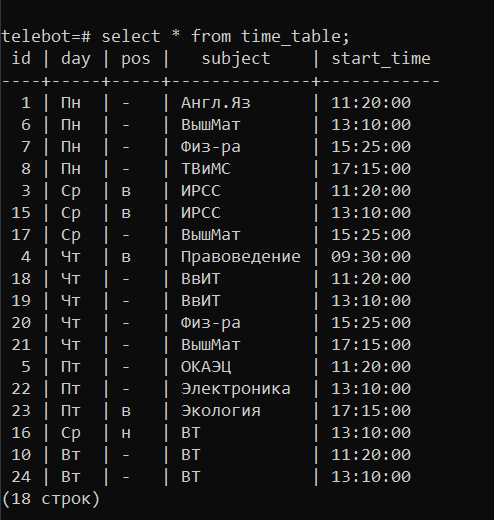


Рисунок 2 – База данных

Программный код изображен на рисунке 3.

import sys  
import psycopg2  
from PyQt5.QtWidgets import (QApplication, QWidget, QTabWidget, QAbstractScrollArea, QVBoxLayout, QHBoxLayout,  
 QTableWidget, QGroupBox, QTableWidget, QGroupBox, QTableWidgetItem, QPushButton,  
 QMessageBox)  
  
  
class Window(QWidget):  
 def \_\_init\_\_(self):  
 super(Window, self).\_\_init\_\_()  
 self.setWindowTitle("Расписание")  
  
 self.\_connect\_to\_db()  
  
 self.tabs = QTabWidget(self)  
  
 self.vbox = QVBoxLayout(self)  
 self.vbox.addWidget(self.tabs)  
 self.\_create\_shedule\_tab()  
  
 def \_connect\_to\_db(self):  
 self.conn = psycopg2.connect(database="telebot",  
 user="postgres",  
 password="1234",  
 host="localhost",  
 port="5432")  
 self.cursor = self.conn.cursor()  
  
 def \_create\_shedule\_tab(self):  
 self.shedule\_tab = QWidget()  
 self.tabs.addTab(self.shedule\_tab, "Расписание")  
 self.teacher\_tab = QWidget()  
 self.tabs.addTab(self.teacher\_tab, "Преподаватели")  
 self.subject\_tab = QWidget()  
 self.tabs.addTab(self.subject\_tab, "Предметы")  
  
 self.monday\_gbox = QGroupBox("Понедельник")  
 self.tuesday\_gbox = QGroupBox("Вторник")  
 self.wednesday\_gbox = QGroupBox("Среда")  
 self.thursday\_gbox = QGroupBox("Четверг")  
 self.friday\_gbox = QGroupBox("Пятница")  
 self.teacher\_gbox = QGroupBox("Перподаватели")  
 self.subject\_gbox = QGroupBox("Предметы")  
  
 self.svbox = QVBoxLayout()  
 self.shbox1 = QVBoxLayout()  
 self.shbox2 = QHBoxLayout()  
 self.svbox2 = QVBoxLayout()  
 self.shbox21 = QVBoxLayout()  
 self.shbox22 = QHBoxLayout()  
 self.svbox3 = QVBoxLayout()  
 self.shbox31 = QVBoxLayout()  
 self.shbox32 = QHBoxLayout()  
  
 self.svbox.addLayout(self.shbox1)  
 self.svbox.addLayout(self.shbox2)  
 self.svbox2.addLayout(self.shbox21)  
 self.svbox2.addLayout(self.shbox22)  
 self.svbox3.addLayout(self.shbox31)  
 self.svbox3.addLayout(self.shbox32)  
  
 self.shbox1.addWidget(self.monday\_gbox)  
 self.shbox1.addWidget(self.tuesday\_gbox)  
 self.shbox1.addWidget(self.wednesday\_gbox)  
 self.shbox1.addWidget(self.thursday\_gbox)  
 self.shbox1.addWidget(self.friday\_gbox)  
 self.shbox21.addWidget(self.teacher\_gbox)  
 self.shbox31.addWidget(self.subject\_gbox)  
  
 self.\_create\_monday\_table()  
 self.\_create\_tuesday\_table()  
 self.\_create\_wednesday\_table()  
 self.\_create\_thursday\_table()  
 self.\_create\_friday\_table()  
 self.\_create\_teacher\_table()  
 self.\_create\_subject\_table()  
  
 self.update\_shedule\_btn = QPushButton("Update")  
 self.update\_teacher\_btn = QPushButton("Update")  
 self.update\_subject\_btn = QPushButton("Update")  
 self.shbox2.addWidget(self.update\_shedule\_btn)  
 self.shbox22.addWidget(self.update\_teacher\_btn)  
 self.shbox32.addWidget(self.update\_subject\_btn)  
 self.update\_shedule\_btn.clicked.connect(self.\_update\_shedule)  
 self.update\_teacher\_btn.clicked.connect(self.\_update\_teacher)  
 self.update\_subject\_btn.clicked.connect(self.\_update\_subject)  
  
 self.shedule\_tab.setLayout(self.svbox)  
 self.teacher\_tab.setLayout(self.svbox2)  
 self.subject\_tab.setLayout(self.svbox3)  
  
 def \_create\_monday\_table(self):  
 self.monday\_table = QTableWidget()  
  
 self.monday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.monday\_table.setColumnCount(6)  
 self.monday\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 self.monday\_table.hideColumn(3)  
  
 self.\_update\_monday\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.monday\_table)  
 self.monday\_gbox.setLayout(self.mvbox)  
  
 def \_create\_tuesday\_table(self):  
 self.tuesday\_table = QTableWidget()  
  
 self.tuesday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.tuesday\_table.setColumnCount(6)  
 self.tuesday\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 self.tuesday\_table.hideColumn(3)  
  
 self.\_update\_tuesday\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.tuesday\_table)  
 self.tuesday\_gbox.setLayout(self.mvbox)  
  
 def \_create\_wednesday\_table(self):  
 self.wednesday\_table = QTableWidget()  
  
 self.wednesday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.wednesday\_table.setColumnCount(6)  
 self.wednesday\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 self.wednesday\_table.hideColumn(3)  
  
 self.\_update\_wednesday\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.wednesday\_table)  
 self.wednesday\_gbox.setLayout(self.mvbox)  
  
 def \_create\_thursday\_table(self):  
 self.thursday\_table = QTableWidget()  
  
 self.thursday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.thursday\_table.setColumnCount(6)  
 self.thursday\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 self.thursday\_table.hideColumn(3)  
  
 self.\_update\_thursday\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.thursday\_table)  
 self.thursday\_gbox.setLayout(self.mvbox)  
  
 def \_create\_friday\_table(self):  
 self.friday\_table = QTableWidget()  
  
 self.friday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.friday\_table.setColumnCount(6)  
 self.friday\_table.setHorizontalHeaderLabels(["Позиция", "Предмет", "Время", "ID", "", ""])  
 self.friday\_table.hideColumn(3)  
  
 self.\_update\_friday\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.friday\_table)  
 self.friday\_gbox.setLayout(self.mvbox)  
  
 def \_create\_teacher\_table(self):  
 self.teacher\_table = QTableWidget()  
  
 self.teacher\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.teacher\_table.setColumnCount(5)  
 self.teacher\_table.setHorizontalHeaderLabels(["Имя", "Предмет", "ID", "", ""])  
 self.teacher\_table.hideColumn(2)  
  
 self.\_update\_teacher\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.teacher\_table)  
 self.teacher\_gbox.setLayout(self.mvbox)  
  
 def \_create\_subject\_table(self):  
 self.subject\_table = QTableWidget()  
  
 self.subject\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.subject\_table.setColumnCount(2)  
 self.subject\_table.setHorizontalHeaderLabels(["Предмет", ""])  
  
 self.\_update\_subject\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.subject\_table)  
 self.subject\_gbox.setLayout(self.mvbox)  
  
 def \_update\_monday\_table(self):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = 'Пн' order by start\_time;")  
 records = list(self.cursor.fetchall())  
  
 self.monday\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.monday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.monday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.monday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.monday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 self.monday\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, self.monday\_table))  
 delbtn = QPushButton("Удалить")  
 self.monday\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, self.monday\_table))  
 addbtn = QPushButton("Добавить")  
 self.monday\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.monday\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.monday\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.monday\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.monday\_table.removeCellWidget(i + 1, 4)  
 self.monday\_table.removeCellWidget(i + 1, 5)  
 self.monday\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, self.monday\_table))  
  
 self.monday\_table.resizeRowsToContents()  
  
 def \_update\_tuesday\_table(self):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = 'Вт' order by start\_time;")  
 records = list(self.cursor.fetchall())  
  
 self.tuesday\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.tuesday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.tuesday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.tuesday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.tuesday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 self.tuesday\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, self.tuesday\_table))  
 delbtn = QPushButton("Удалить")  
 self.tuesday\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, self.tuesday\_table))  
 self.tuesday\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.tuesday\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.tuesday\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.tuesday\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.tuesday\_table.removeCellWidget(i + 1, 4)  
 self.tuesday\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.tuesday\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, self.tuesday\_table))  
 self.tuesday\_table.resizeRowsToContents()  
  
 def \_update\_wednesday\_table(self):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = 'Ср' order by start\_time;")  
 records = list(self.cursor.fetchall())  
  
 self.wednesday\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.wednesday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.wednesday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.wednesday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.wednesday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 self.wednesday\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, self.wednesday\_table))  
 delbtn = QPushButton("Удалить")  
 self.wednesday\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, self.wednesday\_table))  
 self.wednesday\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.wednesday\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.wednesday\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.wednesday\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.wednesday\_table.removeCellWidget(i + 1, 4)  
 self.wednesday\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.wednesday\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, self.wednesday\_table))  
  
 self.wednesday\_table.resizeRowsToContents()  
  
 def \_update\_thursday\_table(self):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = 'Чт' order by start\_time;")  
 records = list(self.cursor.fetchall())  
  
 self.thursday\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.thursday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.thursday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.thursday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.thursday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 self.thursday\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, self.thursday\_table))  
 delbtn = QPushButton("Удалить")  
 self.thursday\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, self.thursday\_table))  
 self.thursday\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.thursday\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.thursday\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.thursday\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.thursday\_table.removeCellWidget(i + 1, 4)  
 self.thursday\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.thursday\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, self.thursday\_table))  
  
 self.thursday\_table.resizeRowsToContents()  
  
 def \_update\_friday\_table(self):  
 self.cursor.execute(  
 "select pos, subject, start\_time, id from time\_table where day = 'Пт' order by start\_time;")  
 records = list(self.cursor.fetchall())  
  
 self.friday\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.friday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.friday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.friday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.friday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 joinbtn = QPushButton("Изменить")  
 self.friday\_table.setCellWidget(i, 4, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_day\_from\_table(j, self.friday\_table))  
 delbtn = QPushButton("Удалить")  
 self.friday\_table.setCellWidget(i, 5, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_time\_table(j, self.friday\_table))  
 self.friday\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.friday\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.friday\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.friday\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.friday\_table.removeCellWidget(i + 1, 4)  
 self.friday\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.friday\_table.setCellWidget(i + 1, 4, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_time\_table(j, self.friday\_table))  
  
 self.friday\_table.resizeRowsToContents()  
  
 def \_update\_teacher\_table(self):  
 self.cursor.execute(  
 "select full\_name, subject, id from teacher;")  
 records = list(self.cursor.fetchall())  
  
 self.teacher\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.teacher\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.teacher\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.teacher\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 joinbtn = QPushButton("Изменить")  
 self.teacher\_table.setCellWidget(i, 3, joinbtn)  
 joinbtn.clicked.connect(lambda checked=None, j=i: self.\_change\_teacher\_table(j, self.teacher\_table))  
 delbtn = QPushButton("Удалить")  
 self.teacher\_table.setCellWidget(i, 4, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_teacher(j, self.teacher\_table))  
 self.teacher\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.teacher\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.teacher\_table.removeCellWidget(i + 1, 4)  
 self.teacher\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.teacher\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.teacher\_table.setCellWidget(i + 1, 3, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_teacher(j, self.teacher\_table))  
  
 def \_update\_subject\_table(self):  
 self.cursor.execute(  
 "select name from subject;")  
 records = list(self.cursor.fetchall())  
  
 self.subject\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 self.subject\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 delbtn = QPushButton("Удалить")  
 self.subject\_table.setCellWidget(i, 1, delbtn)  
 delbtn.clicked.connect(lambda checked=None, j=i: self.\_del\_from\_subject(j, self.subject\_table))  
 self.subject\_table.setItem(i + 1, 0, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 1, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 2, QTableWidgetItem(''))  
 self.subject\_table.setItem(i + 1, 3, QTableWidgetItem(''))  
 self.subject\_table.removeCellWidget(i + 1, 4)  
 self.subject\_table.removeCellWidget(i + 1, 5)  
 addbtn = QPushButton("Добавить")  
 self.subject\_table.setCellWidget(i + 1, 1, addbtn)  
 addbtn.clicked.connect(lambda checked=None, j=i + 1: self.\_add\_subject(j, self.subject\_table))  
  
 def \_change\_day\_from\_table(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 if row[0] == '-' or row[0] == 'н' or row[0] == 'в':  
 if row[2] == '9:30' or row[2] == '11:20' or row[2] == '13:10' or row[2] == '15:25' or row[2] == '17:15':  
 try:  
 self.cursor.execute("update time\_table set pos = '" + row[0] + "' where id = " + row[3] + ";")  
 self.cursor.execute("update time\_table set subject = '" + row[1] + "' where id = " + row[3] + "")  
 self.cursor.execute("update time\_table set start\_time = '" + row[2] + "' where id = " + row[3] + ";")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
 else:  
 QMessageBox.about(self, "Error", "Введите стандартизированое время")  
 else: QMessageBox.about(self, "Error", "Введите положение недели 'в' - верхняя 'н' - няжняя '-' - любая")  
 self.\_update\_monday\_table()  
 self.\_update\_tuesday\_table()  
 self.\_update\_wednesday\_table()  
 self.\_update\_thursday\_table()  
 self.\_update\_friday\_table()  
  
 def \_change\_teacher\_table(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 try:  
 self.cursor.execute("update teacher set full\_name = '" + row[0] + "' where id = " + row[2] + ";")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_del\_from\_time\_table(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 try:  
 self.cursor.execute("delete from time\_table where id = " + row[3] + ";")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
 self.\_update\_monday\_table()  
 self.\_update\_tuesday\_table()  
 self.\_update\_wednesday\_table()  
 self.\_update\_thursday\_table()  
 self.\_update\_friday\_table()  
  
 def \_del\_from\_teacher(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 try:  
 self.cursor.execute("delete from teacher where id = " + row[2] + ";")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
 self.\_update\_teacher\_table()  
  
 def \_del\_from\_subject(self, rown, a):  
 row = list()  
 A = list()  
 B = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 for i in range(self.teacher\_table.rowCount()):  
 try:  
 A.append(self.teacher\_table.item(i, 1).text())  
 except:  
 row.append(None)  
 if row[0] in A:  
 QMessageBox.about(self, "Error",  
 "Нельзя удалить предмет, пока он находится в расписании или в преподавателях")  
 else:  
 try:  
 self.cursor.execute("delete from subject where name = '" + row[0] + "';")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Нельзя удалить предмет, пока он находится в расписании или в преподавателях")  
 self.\_update\_subject\_table()  
  
 def \_add\_time\_table(self, rown, a):  
 row = list()  
 A = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 for i in range(self.subject\_table.rowCount()):  
 try:  
 A.append(self.subject\_table.item(i, 0).text())  
 except:  
 row.append(None)  
 if row[1] in A and row[1] != '':  
 if row[0] == '-' or row[0] == 'н' or row[0] == 'в':  
 if row[2] == '9:30' or row[2] == '11:20' or row[2] == '13:10' or row[2] == '15:25' or row[2] == '17:15':  
 try:  
 if a == self.monday\_table:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('Пн', '" + row[0] + "', '" + row[  
 1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 elif a == self.tuesday\_table:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('Вт', '" + row[0] + "', '" + row[  
 1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 elif a == self.wednesday\_table:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('Ср', '" + row[0] + "', '" + row[  
 1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 elif a == self.thursday\_table:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('Чт', '" + row[0] + "', '" + row[  
 1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 elif a == self.friday\_table:  
 self.cursor.execute(  
 "insert into time\_table(day, pos, subject, start\_time) values('Пт', '" + row[0] + "', '" + row[  
 1] + "', '" + row[2] + "');")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
 self.\_update\_monday\_table()  
 self.\_update\_tuesday\_table()  
 self.\_update\_wednesday\_table()  
 self.\_update\_thursday\_table()  
 self.\_update\_friday\_table()  
 else: QMessageBox.about(self, "Error", "Введите стандартизированое время")  
 else: QMessageBox.about(self, "Error", "Введите положение недели 'в' - верхняя 'н' - няжняя '-' - любая")  
 else: QMessageBox.about(self, "Error", "Такого предмета нет в БД")  
  
 def \_add\_teacher(self, rown, a):  
 A = list()  
 row = list()  
 for i in range(self.subject\_table.rowCount()):  
 try:  
 A.append(self.subject\_table.item(i, 0).text())  
 except:  
 row.append(None)  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 if row[1] in A:  
 try:  
 self.cursor.execute("insert into teacher(full\_name, subject) values('" + row[0] + "', '" + row[1] + "');")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
 self.\_update\_teacher\_table()  
 else: QMessageBox.about(self, "Error", "Такого предмета нет в БД")  
  
 def \_add\_subject(self, rown, a):  
 row = list()  
 for i in range(a.columnCount()):  
 try:  
 row.append(a.item(rown, i).text())  
 except:  
 row.append(None)  
 try:  
 self.cursor.execute("insert into subject(name) values('" + row[0] + "');")  
 self.conn.commit()  
 except:  
 QMessageBox(self, "Error", "Enter all fields")  
 self.\_update\_subject\_table()  
  
 def \_update\_shedule(self):  
 self.\_update\_monday\_table()  
 self.\_update\_tuesday\_table()  
 self.\_update\_wednesday\_table()  
 self.\_update\_thursday\_table()  
 self.\_update\_friday\_table()  
  
 def \_update\_teacher(self):  
 self.\_update\_teacher\_table()  
  
 def \_update\_subject(self):  
 self.\_update\_subject\_table()  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app = QApplication(sys.argv)  
 win = Window()  
 win.show()  
 sys.exit(app.exec\_())

Рисунок 3 – Программный код

# **Результат выполнения**

Результат выполнения изображен на рисунке 4.

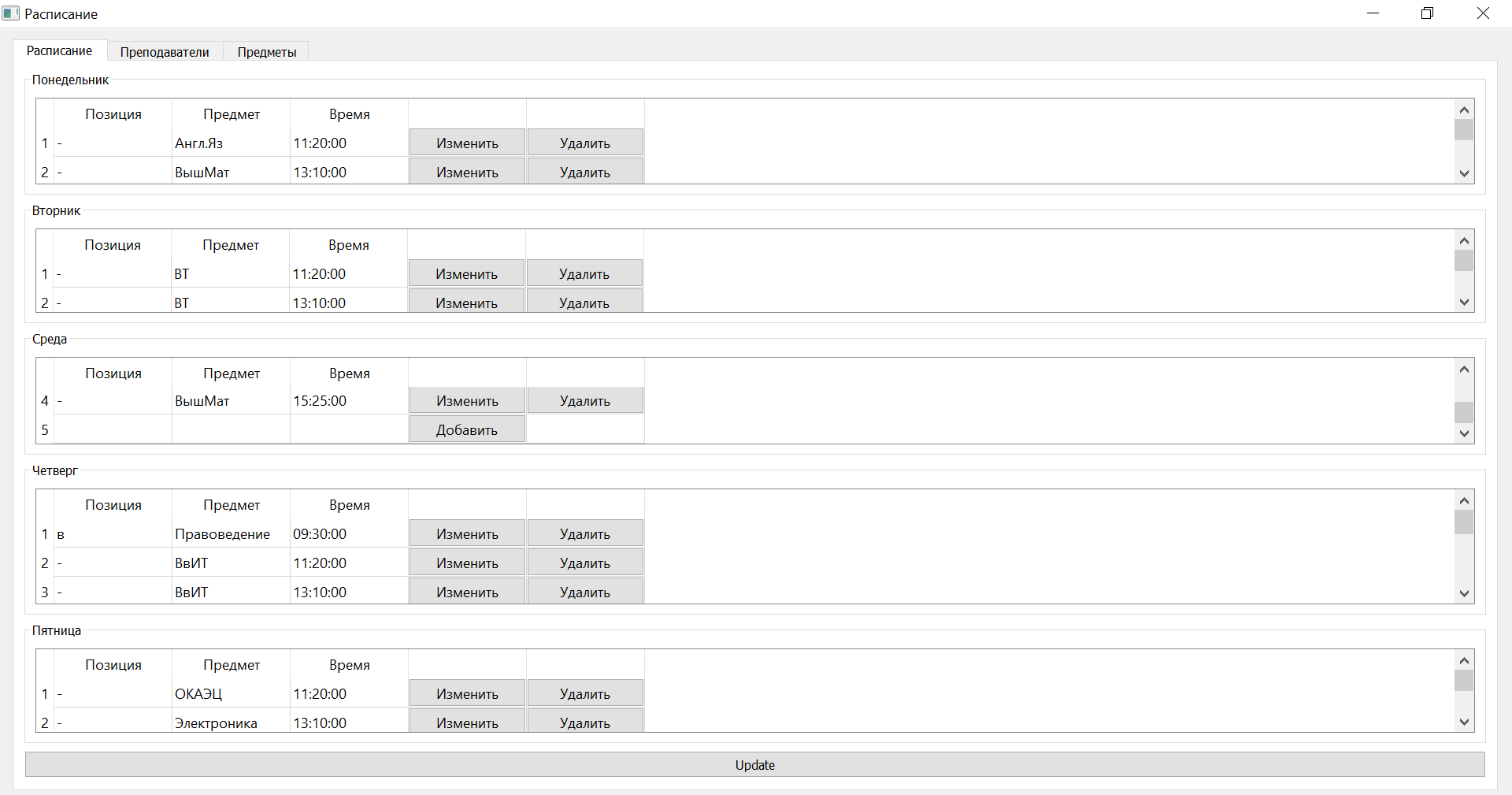


Рисунок 4 – Результат выполнения

# **Вывод**

Мы успешно создали оконное приложение с использованием PyQt5.QtWidget и PostgreSQL с расписанием для своей группы.