**Федеральное агентство связи**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**ПРАКТИЧЕСКАЯ РАБОТА**

**по дисциплине**

**ВвИТ**

на тему

«**Создание оконного приложения для редактирования расписания**»

Выполнила: студентка гр. БИН2005 Коцерева М.В.

Москва 2021 г.

**Цель работы:**

Создать оконное приложение позволяющее редактировать базу данных с расписанием. Приложение должно иметь функционал позволяющий: просматривать базу данных, удалять, добавлять и изменять записи в этой же базе данных. Также должны быть использованы библиотека PyQt5 и адаптер psycopg2.

**Выполнение:**

**Код программы**

import sys

import psycopg2

from PyQt5.QtWidgets import (QApplication, QWidget,

                             QTabWidget, QAbstractScrollArea,

                             QVBoxLayout, QHBoxLayout,

                             QTableWidget, QGroupBox,

                         QTableWidgetItem, QPushButton, QMessageBox)

from PyQt5.QtCore import Qt

class Window(QWidget):

    def \_\_init\_\_(self):

        super(Window, self).\_\_init\_\_()

        self.setWindowTitle("Расписание")

        self.vbox = QVBoxLayout(self)

        self.\_connect\_to\_db()

        self.tabs = QTabWidget(self)

        self.vbox.addWidget(self.tabs)

        self.\_create\_shedule\_tab()

        self.\_create\_shedule1\_tab()

        self.\_create\_shedule2\_tab()

    def \_connect\_to\_db(self):

        self.conn = psycopg2.connect(database = "raspisanie",

                                     user="postgres",

                                     password="123",

                                     host="localhost",

                                     port="5433")

        self.cursor = self.conn.cursor()

    def \_create\_shedule\_tab(self):

        self.shedule\_tab = QWidget()

        self.monday\_gbox = QGroupBox("Понедельник")

        self.svbox = QVBoxLayout()

        self.shbox1 = QHBoxLayout()

        self.shbox2 = QHBoxLayout()

        self.shbox3 = QHBoxLayout()

        self.svbox.addLayout(self.shbox2)

        self.svbox.addLayout(self.shbox3)

        self.svbox.addLayout(self.shbox1)

        self.shbox2.addWidget(self.monday\_gbox)

        self.monday\_table = QTableWidget()

        self.\_create\_monday\_table()

        self.thuseday\_gbox = QGroupBox("Вторник")

        self.shbox2.addWidget(self.thuseday\_gbox)

        self.thuseday\_table = QTableWidget()

        self.\_create\_thuseday\_table()

        self.we\_gbox = QGroupBox("Среда")

        self.shbox2.addWidget(self.we\_gbox)

        self.we\_table = QTableWidget()

        self.\_create\_we\_table()

        self.th\_gbox = QGroupBox("Четверг")

        self.shbox3.addWidget(self.th\_gbox)

        self.th\_table = QTableWidget()

        self.\_create\_th\_table()

        self.fr\_gbox = QGroupBox("Пятница")

        self.shbox3.addWidget(self.fr\_gbox)

        self.fr\_table = QTableWidget()

        self.\_create\_fr\_table()

        self.sa\_gbox = QGroupBox("Суббота")

        self.shbox3.addWidget(self.sa\_gbox)

        self.sa\_table = QTableWidget()

        self.\_create\_sa\_table()

        self.update\_shedule\_button = QPushButton("Update")

        self.update\_shedule\_button.clicked.connect(self.\_update\_shedule)

        self.shbox1.addWidget(self.update\_shedule\_button)

        self.shedule\_tab.setLayout(self.svbox)

        self.tabs.addTab(self.shedule\_tab, "Расписание")

    def \_create\_shedule1\_tab(self):

        self.shedule1\_tab = QWidget()

        self.svbox1=QVBoxLayout()

        self.teacher\_gbox = QGroupBox("Препадователи")

        self.svbox1.addWidget(self.teacher\_gbox)

        self.teacher\_table = QTableWidget()

        self.\_create\_teacher\_table()

        self.update\_shedule1\_button = QPushButton("Update")

        self.update\_shedule1\_button.clicked.connect(self.\_update\_shedule1)

        self.svbox1.addWidget(self.update\_shedule1\_button)

        self.shedule1\_tab.setLayout(self.svbox1)

        self.tabs.addTab(self.shedule1\_tab, "Препадователи")

    def \_create\_shedule2\_tab(self):

        self.shedule2\_tab = QWidget()

        self.svbox1=QVBoxLayout()

        self.sub\_gbox = QGroupBox("Предметы")

        self.svbox1.addWidget(self.sub\_gbox)

        self.sub\_table = QTableWidget()

        self.\_create\_sub\_table()

        self.update\_shedule2\_button = QPushButton("Update")

        self.update\_shedule2\_button.clicked.connect(self.\_update\_shedule2)

        self.svbox1.addWidget(self.update\_shedule2\_button)

        self.shedule2\_tab.setLayout(self.svbox1)

        self.tabs.addTab(self.shedule2\_tab, "Предметы")

    def \_create\_monday\_table(self):

        self.monday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.monday\_table.setColumnCount(7)

        self.monday\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_monday\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.monday\_table)

        self.monday\_gbox.setLayout(self.mvbox)

    def \_create\_thuseday\_table(self):

        self.thuseday\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.thuseday\_table.setColumnCount(7)

        self.thuseday\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_thuseday\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.thuseday\_table)

        self.thuseday\_gbox.setLayout(self.mvbox)

    def \_create\_we\_table(self):

        self.we\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.we\_table.setColumnCount(7)

        self.we\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_we\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.we\_table)

        self.we\_gbox.setLayout(self.mvbox)

    def \_create\_th\_table(self):

        self.th\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.th\_table.setColumnCount(7)

        self.th\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_th\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.th\_table)

        self.th\_gbox.setLayout(self.mvbox)

    def \_create\_fr\_table(self):

        self.fr\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.fr\_table.setColumnCount(7)

        self.fr\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_fr\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.fr\_table)

        self.fr\_gbox.setLayout(self.mvbox)

    def \_create\_sa\_table(self):

        self.sa\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.sa\_table.setColumnCount(7)

        self.sa\_table.setHorizontalHeaderLabels(["ID", "Day", "Subject", "Location", "Time"," ",""])

        self.\_update\_sa\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.sa\_table)

        self.sa\_gbox.setLayout(self.mvbox)

    def \_create\_teacher\_table(self):

        self.teacher\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.teacher\_table.setColumnCount(5)

        self.teacher\_table.setHorizontalHeaderLabels(["ID","Teacher", "Subject",""])

        self.\_update\_teacher\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.teacher\_table)

        self.teacher\_gbox.setLayout(self.mvbox)

    def \_create\_sub\_table(self):

        self.sub\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        self.sub\_table.setColumnCount(5)

        self.sub\_table.setHorizontalHeaderLabels([ "Subject",""])

        self.\_update\_sub\_table()

        self.mvbox = QVBoxLayout()

        self.mvbox.addWidget(self.sub\_table)

        self.sub\_gbox.setLayout(self.mvbox)

    def \_update\_monday\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='понедельник'")

        records = list(self.cursor.fetchall())

        self.monday\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.monday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.monday\_table.item(i,0).setFlags(self.monday\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            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])))

            self.monday\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.monday\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.monday\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.monday\_table))

        addButton = QPushButton("Add")

        self.monday\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.monday\_table))

        self.monday\_table.resizeRowsToContents()

    def \_update\_thuseday\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='вторник'")

        records = list(self.cursor.fetchall())

        self.thuseday\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.thuseday\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.thuseday\_table.item(i,0).setFlags(self.thuseday\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.thuseday\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.thuseday\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            self.thuseday\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))

            self.thuseday\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.thuseday\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.thuseday\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day1\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.thuseday\_table))

        addButton = QPushButton("Add")

        self.thuseday\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.thuseday\_table))

        self.thuseday\_table.resizeRowsToContents()

    def \_update\_we\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='среда'")

        records = list(self.cursor.fetchall())

        self.we\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.we\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.we\_table.item(i,0).setFlags(self.we\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.we\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.we\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            self.we\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))

            self.we\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.we\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.we\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day2\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.we\_table))

        addButton = QPushButton("Add")

        self.we\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.we\_table))

        self.we\_table.resizeRowsToContents()

    def \_update\_th\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='четверг'")

        records = list(self.cursor.fetchall())

        self.th\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.th\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.th\_table.item(i,0).setFlags(self.th\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.th\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.th\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            self.th\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))

            self.th\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.th\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.th\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day3\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.th\_table))

        addButton = QPushButton("Add")

        self.th\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.th\_table))

        self.th\_table.resizeRowsToContents()

    def \_update\_fr\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='пятница'")

        records = list(self.cursor.fetchall())

        self.fr\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.fr\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.fr\_table.item(i,0).setFlags(self.fr\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.fr\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.fr\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            self.fr\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))

            self.fr\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.fr\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.fr\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day4\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.fr\_table))

        addButton = QPushButton("Add")

        self.fr\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.fr\_table))

        self.fr\_table.resizeRowsToContents()

    def \_update\_sa\_table(self):

        self.cursor.execute("SELECT id, day, subject,room\_numb, start\_time FROM timetable WHERE day='суббота'")

        records = list(self.cursor.fetchall())

        self.sa\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.sa\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.sa\_table.item(i,0).setFlags(self.sa\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.sa\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.sa\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            self.sa\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))

            self.sa\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))

            joinButton = QPushButton("Join")

            self.sa\_table.setCellWidget(i, 5, joinButton)

            deleteButton = QPushButton("Delete")

            self.sa\_table.setCellWidget(i, 6, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_day5\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_timetable\_from\_table(num, self.sa\_table))

        addButton = QPushButton("Add")

        self.sa\_table.setCellWidget(i+1, 5, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_timetable\_table(i+1,self.sa\_table))

        self.th\_table.resizeRowsToContents()

    def \_update\_teacher\_table(self):

        self.cursor.execute("SELECT id, full\_name, subject FROM teacher")

        records = list(self.cursor.fetchall())

        self.teacher\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.teacher\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            self.teacher\_table.item(i,0).setFlags(self.teacher\_table.item(i,0).flags() ^ Qt.ItemIsEditable)

            self.teacher\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))

            self.teacher\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))

            joinButton = QPushButton("Join")

            deleteButton = QPushButton("Delete")

            self.teacher\_table.setCellWidget(i, 3, joinButton)

            self.teacher\_table.setCellWidget(i, 4, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_teacher\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_teacher\_from\_table(num))

        addButton = QPushButton("Add")

        self.teacher\_table.setCellWidget(i+1, 3, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_teacher\_table(i+1))

        self.teacher\_table.resizeRowsToContents()

    def \_update\_sub\_table(self):

        self.cursor.execute("SELECT name FROM subject")

        records = list(self.cursor.fetchall())

        self.sub\_table.setRowCount(len(records)+1)

        for i, r in enumerate(records):

            r=list(r)

            print(r)

            self.sub\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))

            joinButton = QPushButton("Join")

            deleteButton = QPushButton("Delete")

            self.sub\_table.setCellWidget(i, 1, joinButton)

            self.sub\_table.setCellWidget(i, 2, deleteButton)

            joinButton.clicked.connect(lambda ch, num=i: self.\_change\_sub\_from\_table(num))

            deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_sub\_from\_table(num))

        addButton = QPushButton("Add")

        self.sub\_table.setCellWidget(i+1, 1, addButton)

        addButton.clicked.connect(lambda ch, num=i: self.\_add\_sub\_table(i+1))

        self.sub\_table.resizeRowsToContents()

    def \_change\_day\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.monday\_table.columnCount()):

            try:

                row.append(self.monday\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0]))) #DELETE FROM timetable WHERE day=%s, subject=%s,room\_numb=%s ,start\_time=%s

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_day1\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.thuseday\_table.columnCount()):

            try:

                row.append(self.thuseday\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_day2\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.we\_table.columnCount()):

            try:

                row.append(self.we\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_day3\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.th\_table.columnCount()):

            try:

                row.append(self.th\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_day4\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.fr\_table.columnCount()):

            try:

                row.append(self.fr\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            print(row)

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_day5\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.sa\_table.columnCount()):

            try:

                row.append(self.sa\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE timetable SET day=%s, subject=%s ,room\_numb=%s ,start\_time=%s WHERE id=%s",(str(row[1]),str(row[2]),str(row[3]), str(row[4]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_teacher\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.teacher\_table.columnCount()):

            try:

                row.append(self.teacher\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        try:

            self.cursor.execute("UPDATE teacher SET full\_name=%s, subject=%s WHERE id=%s",(str(row[1]),str(row[2]),int(row[0])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_change\_sub\_from\_table(self, rowNumb):

        row = list()

        try:

            row.append(self.sub\_table.item(rowNumb, 0).text())

        except:

            row.append(None)

        self.cursor.execute("SELECT \* FROM subject")

        Sp = list(self.cursor.fetchall())

        row.append(str(Sp[rowNumb]))

        row[1]=row[1].replace(",","")

        row[1]=row[1].replace("(","")

        row[1]=row[1].replace(")","")

        row[1]=row[1].replace("'","")

        print(row)

        try:

            self.cursor.execute("UPDATE subject SET name=%s WHERE name=%s",(str(row[0]),str(row[1])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_delete\_timetable\_from\_table(self, rowNumb, table):

        row = list()

        for i in range(table.columnCount()):

            try:

                row.append(table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(row)

        try:

            self.cursor.execute("DELETE FROM timetable WHERE id=%s AND day=%s AND subject=%s AND room\_numb=%s AND start\_time=%s",(str(row[0]),str(row[1]),str(row[2]),str(row[3]),str(row[4])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_delete\_teacher\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.teacher\_table.columnCount()):

            try:

                row.append(self.teacher\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(row)

        try:

            self.cursor.execute("DELETE FROM teacher WHERE id=%s AND full\_name=%s AND subject=%s",(int(row[0]), str(row[1]),str(row[2])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_delete\_sub\_from\_table(self, rowNumb):

        row = list()

        for i in range(self.sub\_table.columnCount()):

            try:

                row.append(self.sub\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(row)

        try:

            self.cursor.execute("DELETE FROM  subject WHERE name=%s",(str(row[0]),))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_add\_timetable\_table(self, rowNumb, table):

        self.cursor.execute("SELECT id FROM timetable")

        records = list(self.cursor.fetchall())

        b=list()

        for a in range (len(records)):

            b.append(str(records[a]))

            b[a]=b[a].replace(",","")

            b[a]=b[a].replace("(","")

            b[a]=b[a].replace(")","")

            b[a]=b[a].replace("'","")

        print (type(b[0]))

        id=max(b,key=lambda i: int(i))

        row = list()

        for i in range(table.columnCount()):

            try:

                row.append(table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(id, row)

        try:

            self.cursor.execute("INSERT INTO timetable (id, day, subject, room\_numb, start\_time) VALUES (%s, %s, %s, %s, %s)",(int(id)+1,str(row[1]),str(row[2]),str(row[3]),str(row[4])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_add\_teacher\_table(self, rowNumb):

        self.cursor.execute("SELECT id FROM teacher")

        records = list(self.cursor.fetchall())

        b=list()

        for a in range (len(records)):

            b.append(str(records[a]))

            b[a]=b[a].replace(",","")

            b[a]=b[a].replace("(","")

            b[a]=b[a].replace(")","")

            b[a]=b[a].replace("'","")

        print (type(b[0]))

        id=max(b,key=lambda i: int(i))

        row = list()

        for i in range(self.teacher\_table.columnCount()):

            try:

                row.append(self.teacher\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(id, row)

        try:

            self.cursor.execute("INSERT INTO teacher (id, full\_name, subject) VALUES (%s, %s, %s)",(int(id)+1,str(row[1]),str(row[2])))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_add\_sub\_table(self, rowNumb):

        row = list()

        for i in range(self.sub\_table.columnCount()):

            try:

                row.append(self.sub\_table.item(rowNumb, i).text())

            except:

                row.append(None)

        print(id, row)

        try:

            self.cursor.execute("INSERT INTO subject (name) VALUES (%s)",(str(row[0]),))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "Enter all fields")

    def \_update\_shedule(self):

        self.monday\_table.clear()

        self.\_create\_monday\_table()

        self.thuseday\_table.clear()

        self.\_create\_thuseday\_table()

        self.we\_table.clear()

        self.\_create\_we\_table()

        self.th\_table.clear()

        self.\_create\_th\_table()

        self.fr\_table.clear()

        self.\_create\_fr\_table()

        self.sa\_table.clear()

        self.\_create\_sa\_table()

    def \_update\_shedule1(self):

        self.teacher\_table.clear()

        self.\_create\_teacher\_table()

    def \_update\_shedule2(self):

        self.sub\_table.clear()

        self.\_create\_sub\_table()

if \_\_name\_\_ == '\_\_main\_\_':

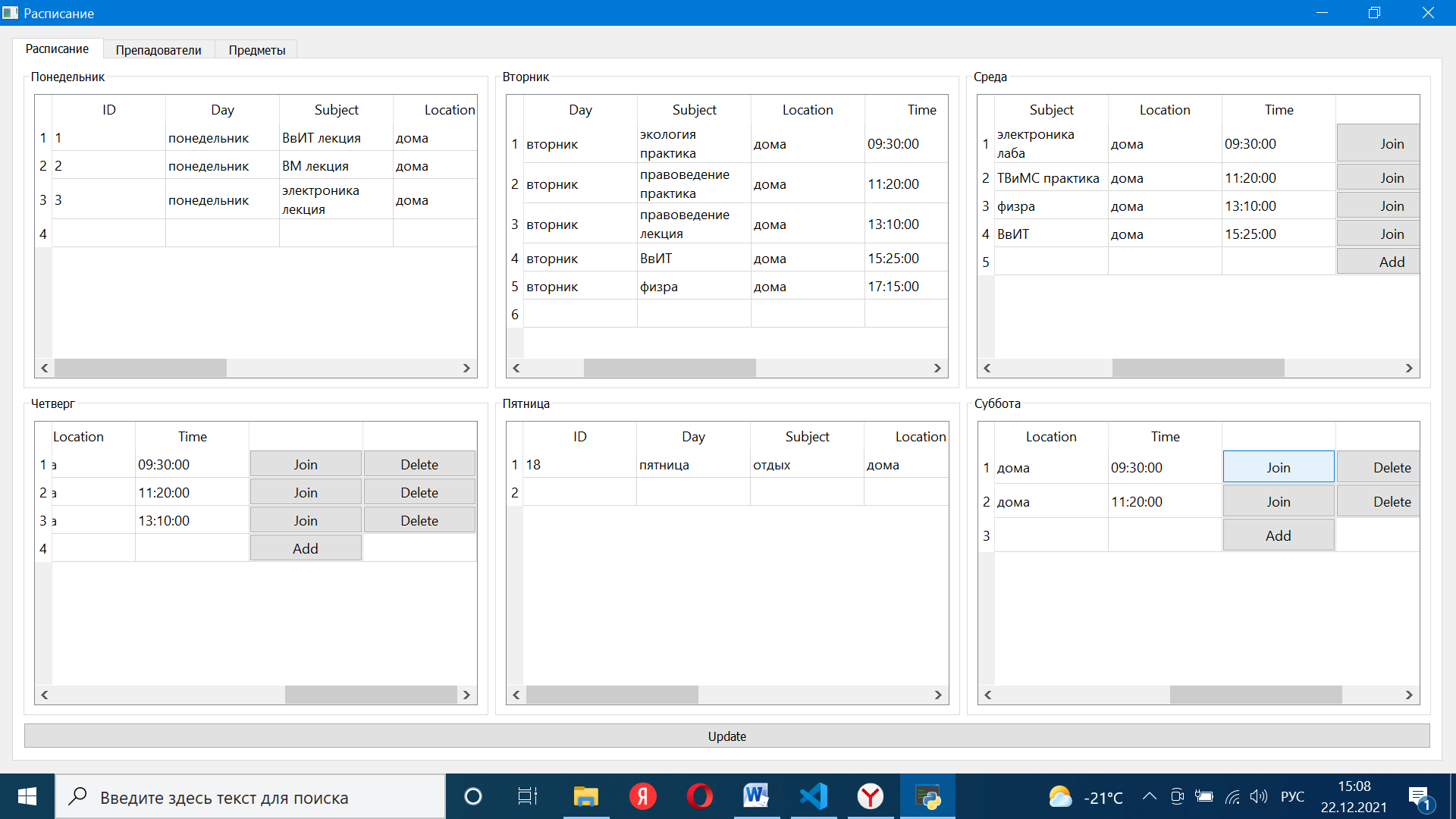
    app =QApplication(sys.argv)

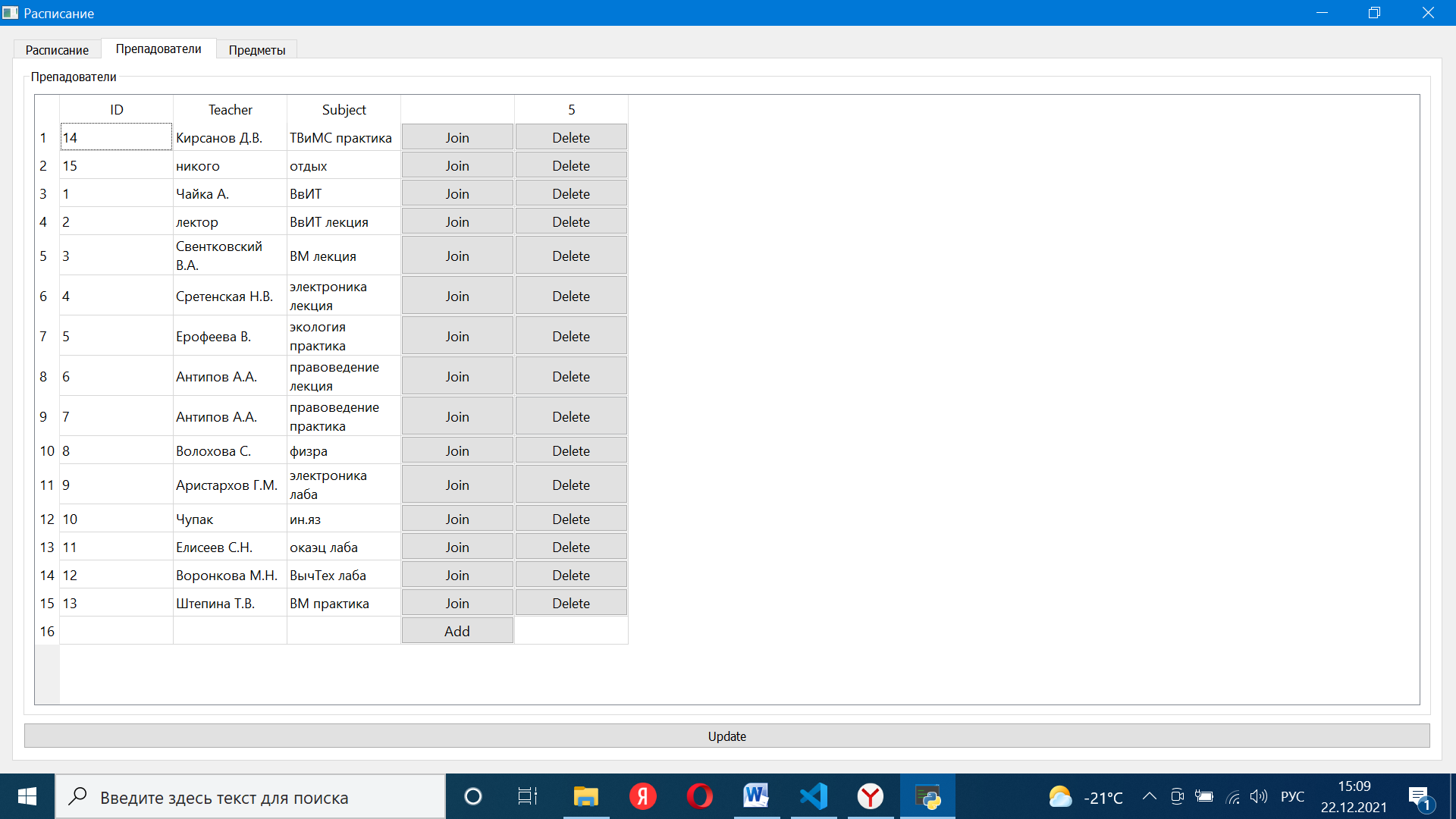
    win = Window()

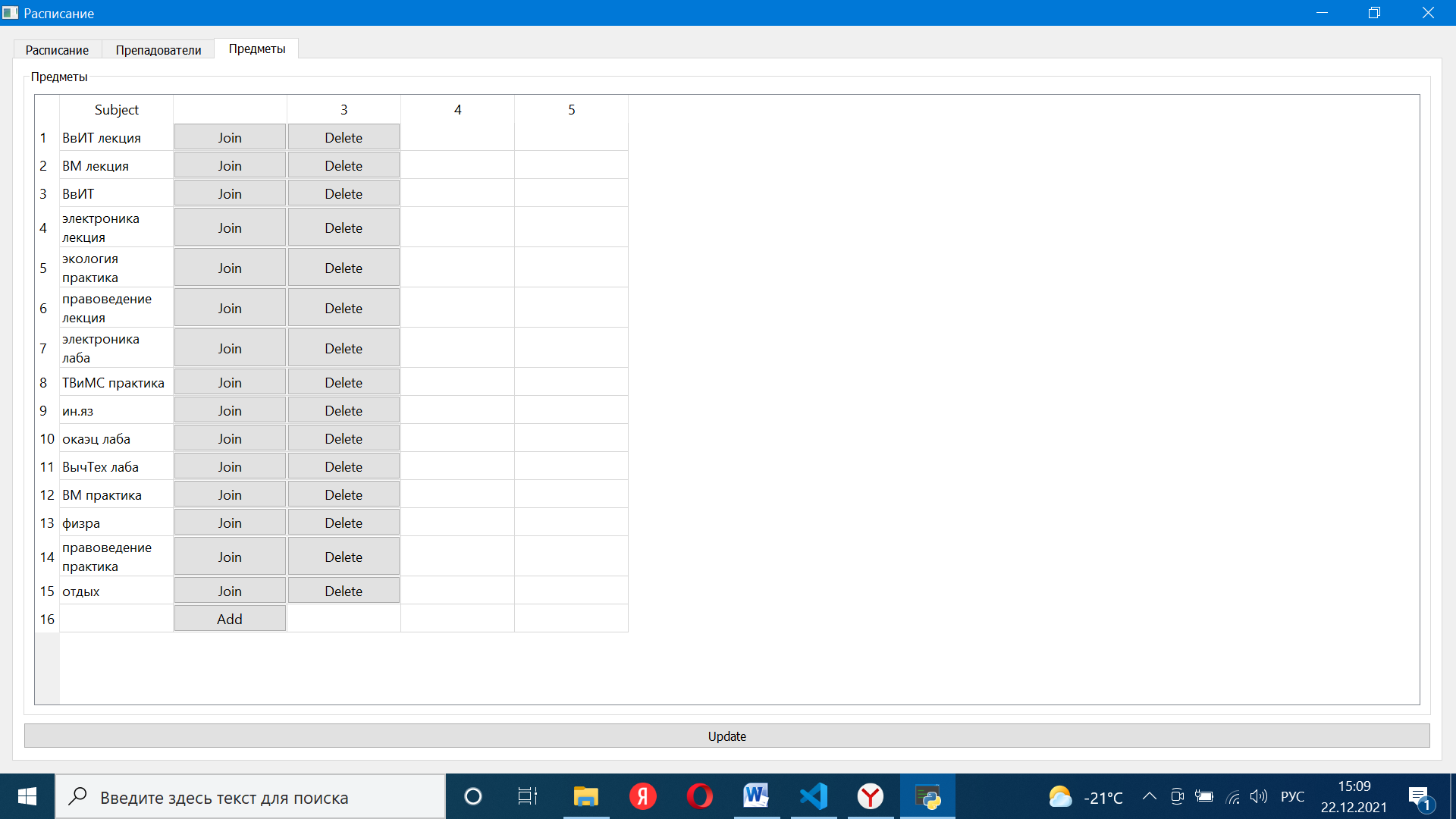
    win.show()

    sys.argv(app.exec\_())

**Результат работы**







**Вывод:**

Мы создали оконное приложение позволяющее редактировать базу данных с расписанием. В работе были использованы библиотека PyQt5 и адаптер psycopg2. В приложение можно: просматривать базу данных, обновлять, удалять, добавлять и изменять записи в этой же базе данных, для этого были созданы кнопки: Update, Delete, Add и Join. Приложение содержит 3 вкладки, в каждой из которых содержится информация из отдельной таблицы в базе данных. Внутри каждой вкладки информация отображается в виде таблиц и отображается кнопка с обновлением информации. Внутри каждой таблицы после каждой строки записи отображены кнопки изменения и удаления записи, а так же пустая строка с кнопкой для добавления новой записи. На вкладке с расписанием дни недели указаны в отдельных таблицах.