**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**

Кафедра «Информатики»

**ЛАБОРАТОРНАЯ РАБОТА №8**

**по дисциплине ВвИТ**

«Бот UI»

Выполнил студент группы БИН2003 Пряхин И. А.

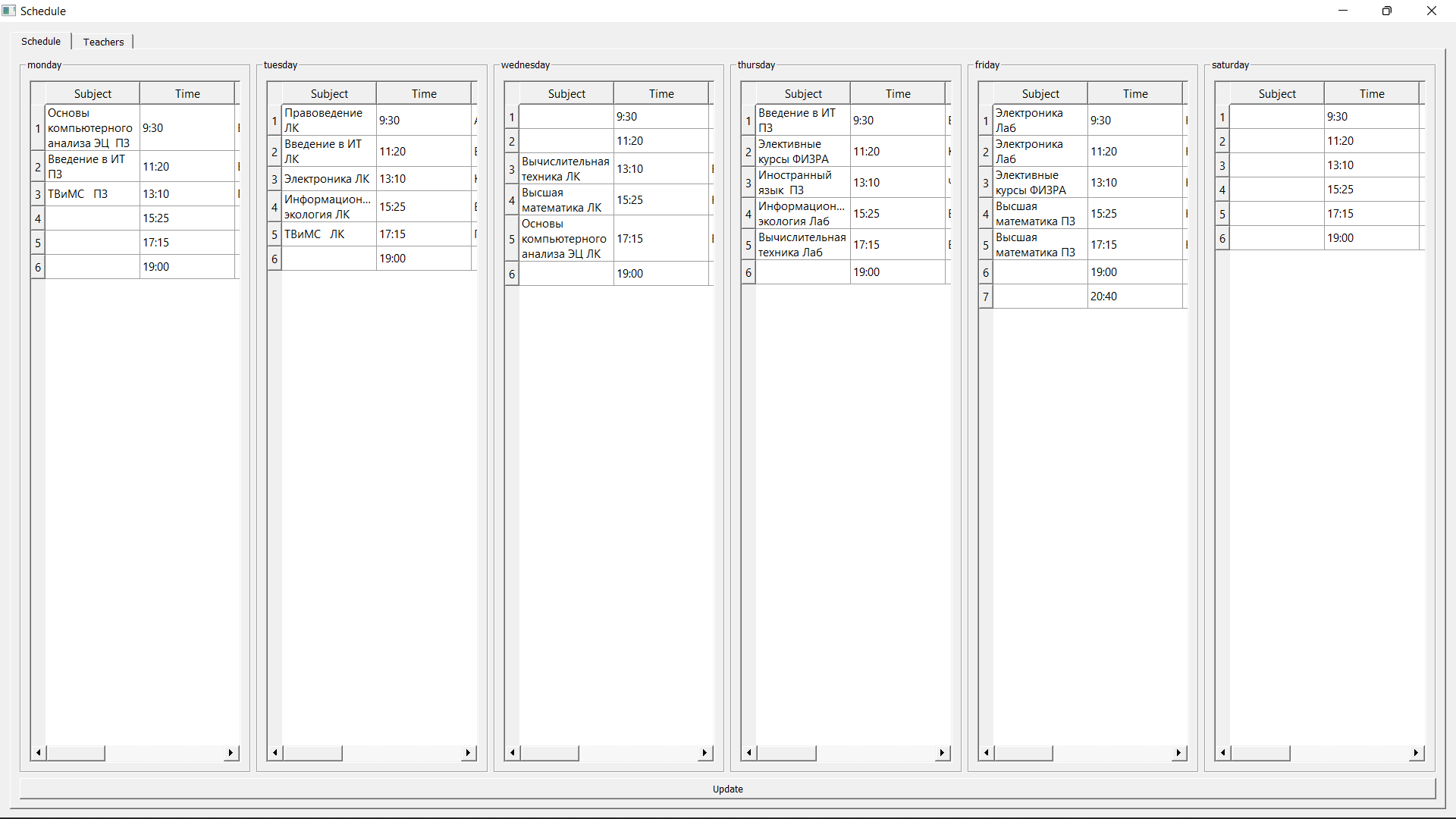
Проверил: Аршинов Е. А.

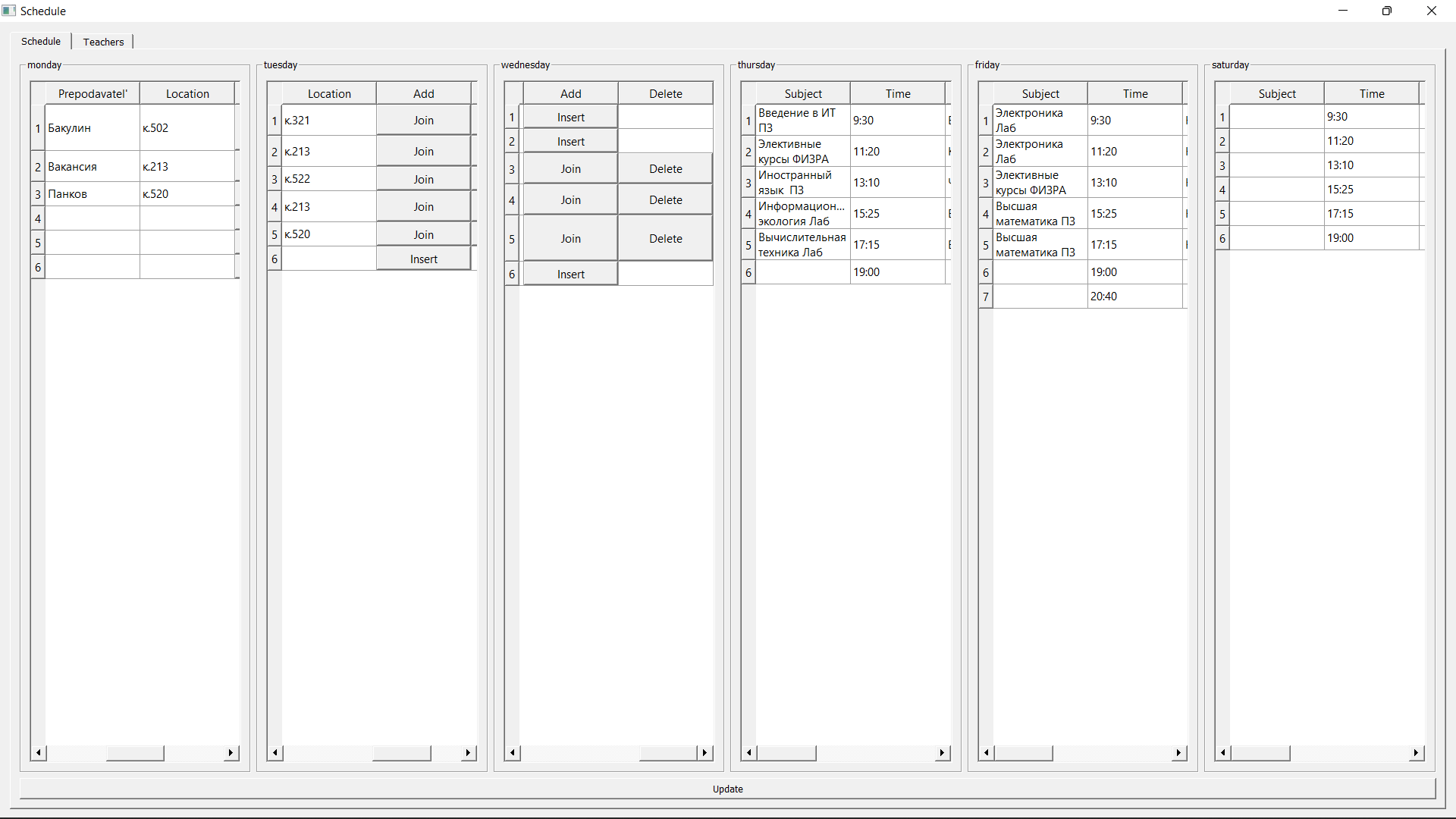
Москва 2021

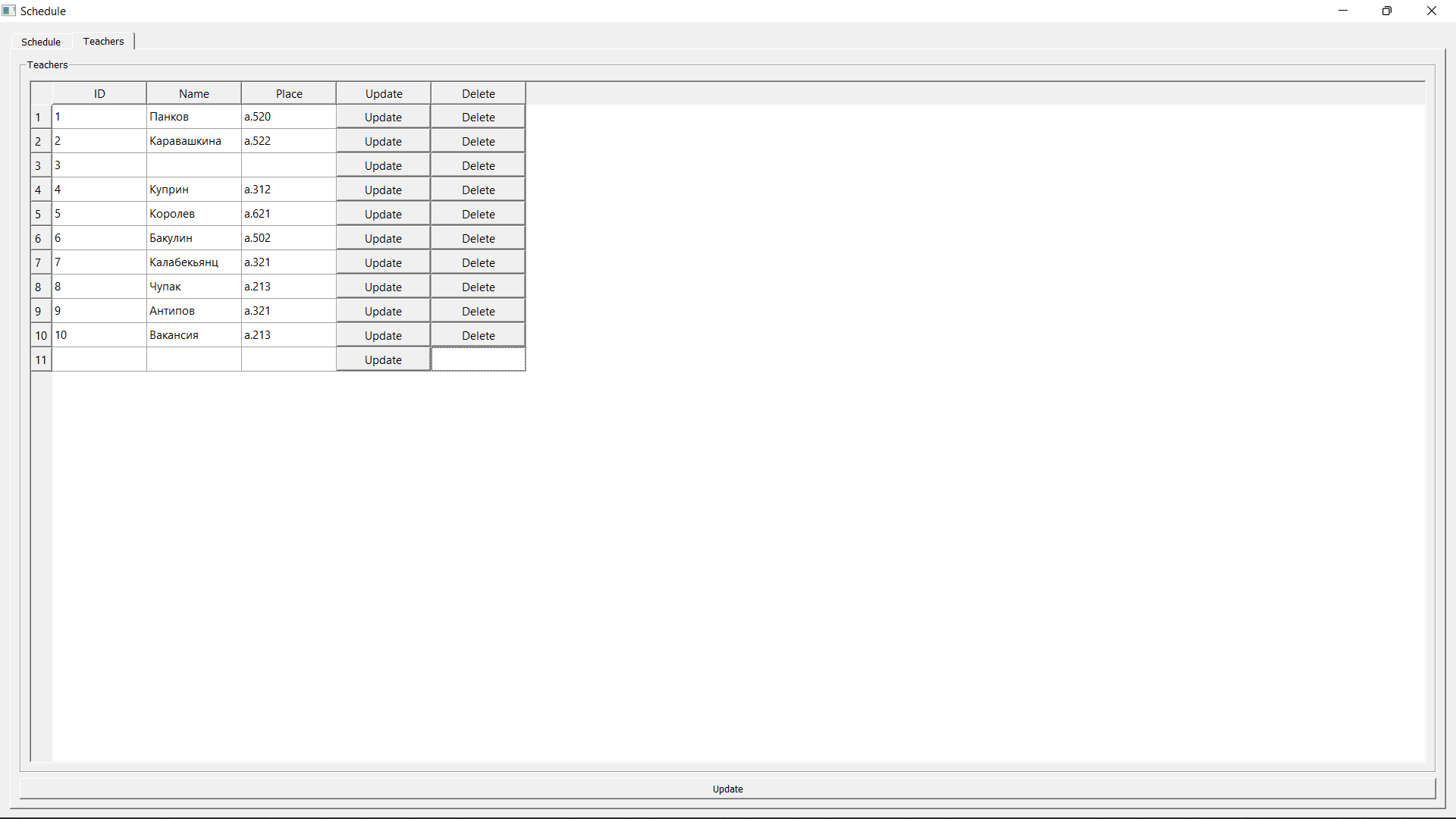
**1. Цель работы:**

Создать свое приложение, которое будет выводит актуальное расписание с преподавателями.

**2. Выполнение работы:**







Код:

import psycopg2

import sys

from datetime import date

from PyQt5.QtWidgets import (QApplication, QWidget,

                             QTabWidget, QAbstractScrollArea,

                             QVBoxLayout, QHBoxLayout,

                             QTableWidget, QGroupBox,

                         QTableWidgetItem, QPushButton, QMessageBox)

from config import DATABASE, USER, PASSWORD

time = ['9:30', '11:20', '13:10', '15:25', '17:15', '19:00', '20:40', '22:10']

days = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday']

class MainWindow(QWidget):

    def \_\_init\_\_(self):

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

        self.week\_type = 'чет' if get\_week\_num() % 2 == 0 else 'неч'

        self.\_connect\_to\_db()

        self.setWindowTitle("Schedule")

        self.vbox = QVBoxLayout(self)

        self.tabs = QTabWidget(self)

        self.vbox.addWidget(self.tabs)

        self.\_create\_shedule\_tab()

        self.\_create\_teachers\_tab()

    def \_connect\_to\_db(self):

        self.conn = psycopg2.connect(database=DATABASE,

                                        user=USER,

                                        password=PASSWORD,

                                        host="localhost",

                                        port="5432")

        self.cursor = self.conn.cursor()

        self.timetable\_table\_name = 'qtimetable'

        self.teachers\_table\_name = 'teachers'

        self.teachers\_names, self.teachers\_places = self.\_fetch\_teachers()

        # self.teachers\_names = []

        self.class\_names = self.\_fetch\_classes()

    def \_fetch\_teachers(self):

        select\_teachers = f"SELECT id, name FROM {self.teachers\_table\_name}"

        self.cursor.execute(select\_teachers)

        names = dict(self.cursor.fetchall())

        select\_teachers = f"SELECT id, place FROM {self.teachers\_table\_name}"

        self.cursor.execute(select\_teachers)

        places = dict(self.cursor.fetchall())

        return [names, places]

    def \_fetch\_classes(self):

        select\_classes = f"SELECT \* FROM {self.timetable\_table\_name}"

        self.cursor.execute(select\_classes)

        return [class\_name[1] for class\_name in (self.cursor.fetchall())]

    def \_create\_teachers\_table(self, gbox):

        table = QTableWidget()

        table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        table.setColumnCount(5)

        table.setHorizontalHeaderLabels(["ID", "Name", "Place", "Update", "Delete"])

        table.setRowCount(len(self.teachers\_names) + 1)

        for i in range(len(self.teachers\_names)):

            joinButton = QPushButton("Update")

            joinButton.clicked.connect(lambda ch, tbl=table, id=(i + 1):self.\_change\_teacher\_from\_table(tbl, id))

            deleteButton = QPushButton("Delete")

            deleteButton.clicked.connect(lambda ch, t\_id=(i + 1):self.\_delete\_teacher(t\_id))

            table.setItem(i, 0, QTableWidgetItem(str(i + 1)))

            try:

                table.setItem(i, 1, QTableWidgetItem(str(self.teachers\_names[i + 1])))

                table.setItem(i, 2, QTableWidgetItem(str(self.teachers\_places[i + 1])))

            except KeyError:

                table.setItem(i, 1, QTableWidgetItem())

                table.setItem(i, 2, QTableWidgetItem())

            table.setCellWidget(i, 3, joinButton)

            table.setCellWidget(i, 4, deleteButton)

        joinButton = QPushButton("Update")

        joinButton.clicked.connect(lambda ch, tbl=table:self.\_insert\_teacher(tbl.item(i + 1, 1).text(), tbl.item(i + 1, 2).text()))

        table.setItem(i + 1, 0, QTableWidgetItem(''))

        table.setItem(i + 1, 1, QTableWidgetItem(''))

        table.setItem(i + 1, 2, QTableWidgetItem(''))

        table.setCellWidget(i + 1, 3, joinButton)

        table.resizeRowsToContents()

        mvbox = QVBoxLayout()

        mvbox.addWidget(table)

        gbox.setLayout(mvbox)

    def \_create\_teachers\_tab(self):

        self.teachers\_tab = QWidget()

        self.tabs.addTab(self.teachers\_tab, "Teachers")

        gbox = QGroupBox('Teachers')

        svbox = QVBoxLayout()

        shboxes = [QHBoxLayout() for \_ in range(2)]

        [svbox.addLayout(shbox) for shbox in shboxes]

        shboxes[0].addWidget(gbox)

        self.\_create\_teachers\_table(gbox)

        self.teachers\_tab.setLayout(svbox)

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

        shboxes[1].addWidget(update\_shedule\_button)

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

        self.teachers\_tab.setLayout(svbox)

    def \_create\_table(self, table, gbox, weekday):

        table = QTableWidget()

        table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

        table.setColumnCount(6)

        table.setHorizontalHeaderLabels(["Subject", "Time", "Prepodavatel'", "Location", "Add", "Delete"])

        self.\_update\_table(table, weekday)

        mvbox = QVBoxLayout()

        mvbox.addWidget(table)

        gbox.setLayout(mvbox)

    def \_create\_shedule\_tab(self):

        self.shedule\_tab = QWidget()

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

        self.gboxes = [QGroupBox(day) for day in days]

        self.svbox = QVBoxLayout()

        self.shboxes = [QHBoxLayout() for \_ in range(2)]

        [self.svbox.addLayout(shbox) for shbox in self.shboxes]

        [self.shboxes[0].addWidget(day\_box) for day\_box in self.gboxes]

        self.tables = [QTableWidget() for \_ in range(6)]

        for i, table in enumerate(self.tables):

            self.\_create\_table(table, self.gboxes[i], i)

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

        self.shboxes[1].addWidget(self.update\_shedule\_button)

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

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

    def \_update\_table(self, table, weekday):

        global time

        what\_we\_need = f"WHERE weekday = {weekday} AND week = '{self.week\_type}';"

        select\_day = f"SELECT \* FROM {self.timetable\_table\_name} {what\_we\_need}"

        self.cursor.execute(select\_day)

        records = sorted(list(self.cursor.fetchall()), key=lambda elem: elem[4])

        table.setRowCount(len(records) + 1)

        empty = ['None', 'удалена', '', 'тут могла быть ваша пара']

        for i, r in enumerate(records):

            if str(r[1]) not in empty:

                joinButton = QPushButton("Join")

                deleteButton = QPushButton("Delete")

                joinButton.clicked.connect(lambda ch, wd=weekday, tbl=table, class\_num=i:self.\_change\_day\_from\_table(tbl, wd, class\_num))

                deleteButton.clicked.connect(lambda ch, tbl=table, wd=weekday, num = r[4]:self.\_delete\_class(wd, num))

                table.setItem(i, 0, QTableWidgetItem(str(r[1])))

                table.setItem(i, 1, QTableWidgetItem(str(time[i])))

                try:

                    table.setItem(i, 2, QTableWidgetItem(str(self.teachers\_names[r[5]])))

                    table.setItem(i, 3, QTableWidgetItem(str(self.teachers\_places[r[5]])))

                except KeyError:

                    table.setItem(i, 2, QTableWidgetItem(str(self.teachers\_names[10])))

                    table.setItem(i, 3, QTableWidgetItem((self.teachers\_places[10])))

                table.setCellWidget(i, 4, joinButton)

                table.setCellWidget(i, 5, deleteButton)

            else:

                table.setItem(i, 0, QTableWidgetItem(''))

                table.setItem(i, 1, QTableWidgetItem(str(time[i])))

                insert\_button = QPushButton("Insert")

                insert\_button.clicked.connect(lambda ch, tbl=table, wd=weekday, num=i:self.\_change\_day\_from\_table(tbl, wd, num))

                table.setCellWidget(i, 4, insert\_button)

        insert\_button = QPushButton("Insert")

        insert\_button.clicked.connect(lambda ch, tbl=table:self.\_insert\_class(tbl.item(i + 1, 0).text(), weekday, i + 1, tbl.item(i + 1, 0).text()))

        table.setItem(i + 1, 0, QTableWidgetItem(''))

        table.setItem(i + 1, 1, QTableWidgetItem(str(time[i + 1])))

        table.setCellWidget(i + 1, 4, insert\_button)

        table.resizeRowsToContents()

    def \_change\_day\_from\_table(self, table, weekday, class\_num):

        try:

            text = table.item(class\_num, 0).text()

            try:

                pr\_id = int(table.item(class\_num, 2).text())

                if pr\_id > len(self.teachers\_names):

                    return QMessageBox.about(self, "Error", "Такого id не существует")

            except:

               return QMessageBox.about(self, "Error", "Введите ID цифрами")

            update\_day = f"UPDATE {self.timetable\_table\_name} SET class\_name = %s, pr\_id = %s WHERE weekday = %s AND class\_num = %s AND week = '{self.week\_type}'"

            self.cursor.execute(update\_day, (text, pr\_id, weekday, class\_num))

            self.conn.commit()

        except:

            QMessageBox.about(self, "Error", "sql error")

    def \_change\_teacher\_from\_table(self, table, id):

        try:

            update\_teacher = f"UPDATE {self.teachers\_table\_name} SET name = %s WHERE id = %s"

            self.cursor.execute(update\_teacher, (str(table.item(id - 1, 1).text()), str(id), ))

            update\_teacher = f"UPDATE {self.teachers\_table\_name} SET place = %s WHERE id = %s"

            self.cursor.execute(update\_teacher, (str(table.item(id - 1, 2).text()), str(id), ))

            self.conn.commit()

        except:

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

    def \_insert\_class(self, class\_name, weekday, class\_num, pr\_id):

        try:

            insert\_data = f"""

            INSERT INTO {self.timetable\_table\_name} (class\_name, week, weekday, class\_num, pr\_id)

            VALUES (%s, %s, %s, %s, %s);

            """

            self.cursor.execute(insert\_data, (class\_name, self.week\_type, str(weekday), str(class\_num), str(pr\_id), ))

            self.conn.commit()

            self.\_update\_shedule()

        except:

            QMessageBox.about(self, "Error", "sql error")

    def \_insert\_teacher(self, name, place):

        insert\_data = f"""

        INSERT INTO {self.teachers\_table\_name} (name, place)

        VALUES (%s, %s);

        """

        self.cursor.execute(insert\_data, (name, place, ))

        self.conn.commit()

        self.\_update\_shedule()

    def \_delete\_class(self, weekday, class\_num):

        update\_day = f"UPDATE {self.timetable\_table\_name} SET class\_name = %s WHERE weekday = %s AND class\_num = %s AND week = '{self.week\_type}'"

        self.cursor.execute(update\_day, ('', weekday, class\_num))

        self.conn.commit()

        self.\_update\_shedule()

    def \_delete\_teacher(self, teacher\_id):

        delete\_day = f"DELETE FROM {self.teachers\_table\_name} WHERE id = %s;"

        self.cursor.execute(delete\_day, (str(teacher\_id), ))

        self.conn.commit()

        self.\_update\_shedule()

    def \_update\_shedule(self):

        self.teachers\_names, self.teachers\_places = self.\_fetch\_teachers()

        self.class\_names = self.\_fetch\_classes()

        self.tabs.removeTab(1)

        self.tabs.removeTab(0)

        self.\_create\_shedule\_tab()

        self.\_create\_teachers\_tab()

def get\_week\_num():

    first\_day = date(2021, 8, 30)

    today = date.today()

    delta = (today - first\_day).days

    week\_number = (delta // 7) + 1

    return week\_number

app = QApplication(sys.argv)

win = MainWindow()

win.show()

sys.exit(app.exec\_())

database.py :

import psycopg2

import json

from config import DATABASE, USER, PASSWORD

conn = psycopg2.connect(database=DATABASE,

                        user=USER,

                        password=PASSWORD,

                        host="localhost",

                        port="5432")

cur = conn.cursor()

table\_name = 'qtimetable'

filename = 'data.json'

def create\_database(conn, cur, table\_name):

    create\_rasp\_table = f"""

    CREATE TABLE {table\_name} (

    id SERIAL,

    class\_name VARCHAR(256),

    week VARCHAR(256),

    weekday INTEGER,

    class\_num INTEGER,

    pr\_id INTEGER

    );

    """

    create\_teachers\_table = f"""

    CREATE TABLE teachers (

    id SERIAL,

    name VARCHAR(256),

    place VARCHAR(256)

    );

    """

    cur.execute(create\_rasp\_table)

    cur.execute(create\_teachers\_table)

    # cur.close()

    conn.commit()

def post\_timetable\_to\_db(conn, cur, filename):

    with open(filename, encoding='UTF-8') as file:

        data = json.loads(file.read())

    print(data)

    insert\_data = f"""

    INSERT INTO {table\_name} (class\_name, week, weekday, class\_num, pr\_id)

    VALUES (%s, %s, %s, %s, %s);

    """

    for i, item in enumerate(data):

        class\_num = str(i % 5)

        weekday = str((i // 5) % 6)

        # print(i, class\_num, weekday)

        cur.execute(insert\_data, (str(item['para']), str(item['week']), weekday, class\_num, item['pr\_id'], ))

    cur.execute(f'SELECT \* FROM {table\_name};')

    conn.commit()

def post\_teachers\_to\_db(conn, cur):

    teachers = [

        ['Панков', 'к.520'],

        ['Сретенская', 'к.522'],

        ['Аршинов', 'к.301'],

        ['Куприн', 'к.312'],

        ['Королев', 'к.621'],

        ['Бакулин', 'к.502'],

        ['Калабекьянц', 'к.321'],

        ['Чупак', 'к.213'],

        ['Антипов', 'к.502'],

        ['Вакансия', 'к.213']

    ]

    insert\_data = f"""

    INSERT INTO teachers (name, place)

    VALUES (%s, %s);

    """

    for teacher in teachers:

       cur.execute(insert\_data, (teacher[0], teacher[1], ))

    cur.close()

    conn.commit()

# create\_database(conn, cur, table\_name)

post\_timetable\_to\_db(conn, cur, filename)

post\_teachers\_to\_db(conn, cur)

