**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра «Математическая Кибернетика и Информационные технологии»

Лабораторная работа №7

Выполнил: Студент группы

БВТ2203

Попов Александр

Москва

2022

**Задание:**

Создать оконное приложение, позволяющее редактировать базу данных с расписанием Вашей группы.

**Ход работы:**

import psycopg2  
import sys  
  
from PyQt5.QtWidgets import (QApplication, QWidget,  
 QTabWidget, QAbstractScrollArea,  
 QVBoxLayout, QHBoxLayout,  
 QTableWidget, QGroupBox,  
 QTableWidgetItem, QPushButton, QMessageBox)  
  
  
class MainWindow(QWidget):  
 def \_\_init\_\_(self):  
 super(MainWindow, self).\_\_init\_\_()  
  
 self.\_connect\_to\_db()  
  
 self.setWindowTitle("Shedule")  
  
 self.vbox = QVBoxLayout(self)  
  
 self.tabs = QTabWidget(self)  
 self.vbox.addWidget(self.tabs)  
  
 self.\_create\_shedule\_tab()  
  
 self.\_create\_teachers\_tab()  
  
 self.\_create\_subjects\_tab()  
  
 def \_connect\_to\_db(self):  
 self.conn = psycopg2.connect(database="shedule\_db",  
 user="postgres",  
 password="ph159753da",  
 host="localhost",  
 port="5432")  
  
 self.cursor = self.conn.cursor()  
  
 def \_create\_shedule\_tab(self):  
 self.shedule\_tab = QWidget()  
 self.tabs.addTab(self.shedule\_tab, "Shedule")  
  
 self.schedule\_gbox = QGroupBox("timetable")  
  
 self.svbox = QVBoxLayout()  
 self.shbox1 = QHBoxLayout()  
 self.shbox2 = QHBoxLayout()  
  
 self.svbox.addLayout(self.shbox1)  
 self.svbox.addLayout(self.shbox2)  
  
 self.shbox1.addWidget(self.schedule\_gbox)  
  
 self.\_create\_schedule\_table()  
  
 self.update\_shedule\_button = QPushButton("Update")  
 self.shbox2.addWidget(self.update\_shedule\_button)  
 self.update\_shedule\_button.clicked.connect(self.\_update\_shedule)  
  
 self.shedule\_tab.setLayout(self.svbox)  
  
 def \_create\_schedule\_table(self):  
 self.schedule\_table = QTableWidget()  
 self.schedule\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.schedule\_table.setColumnCount(8)  
 self.schedule\_table.setHorizontalHeaderLabels(  
 ["id", "day", "subject", "start\_time", "week", "room\_numb"])  
  
 self.\_update\_schedule\_table()  
  
 self.mvbox = QVBoxLayout()  
 self.mvbox.addWidget(self.schedule\_table)  
 self.schedule\_gbox.setLayout(self.mvbox)  
  
 def \_update\_schedule\_table(self):  
 self.cursor.execute("SELECT \* FROM timetable ORDER BY week, id")  
 records = list(self.cursor.fetchall())  
  
 self.schedule\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 joinButton = QPushButton("Join")  
 deleteButton = QPushButton("Delete")  
  
 self.schedule\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.schedule\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.schedule\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.schedule\_table.setItem(i, 3, QTableWidgetItem(str(r[3])))  
 self.schedule\_table.setItem(i, 4, QTableWidgetItem(str(r[4])))  
 self.schedule\_table.setItem(i, 5, QTableWidgetItem(str(r[5])))  
 self.schedule\_table.setCellWidget(i, 6, joinButton)  
 self.schedule\_table.setCellWidget(i, 7, deleteButton)  
  
 joinButton.clicked.connect(lambda ch, num=i: self.\_edit\_timetable(num))  
 deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_from\_timetable(num))  
  
 joinButton = QPushButton("Join")  
 self.schedule\_table.setCellWidget(len(records), 6, joinButton)  
 joinButton.clicked.connect(lambda ch, num=len(records): self.\_add\_to\_timetable(num))  
  
 self.schedule\_table.resizeRowsToContents()  
  
 def \_add\_to\_timetable(self, num):  
 row = list()  
 for i in range(self.schedule\_table.columnCount()):  
 try:  
 row.append(self.schedule\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["id", "day", "subject", "start\_time", "week", "room\_numb"]  
 self.cursor.execute(  
 f"INSERT INTO timetable({columns[0]}, {columns[1]}, {columns[2]}, {columns[3]}, {columns[4]}, {columns[5]}) values('{row[0]}', '{row[1]}', '{row[2]}', '{row[3]}', '{row[4]}', '{row[5]}')")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_edit\_timetable(self, num):  
 row = list()  
 for i in range(self.schedule\_table.columnCount()):  
 try:  
 row.append(self.schedule\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["id", "day", "subject", "start\_time", "week", "room\_numb"]  
 for i in range(1, 5):  
 self.cursor.execute(f"UPDATE timetable SET {columns[i]} = '{row[i]}' WHERE id = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_delete\_from\_timetable(self, num):  
 row = list()  
 for i in range(self.schedule\_table.columnCount()):  
 try:  
 row.append(self.schedule\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 self.cursor.execute(f"DELETE from timetable where id = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_create\_teachers\_tab(self):  
 self.teachers\_tab = QWidget()  
 self.tabs.addTab(self.teachers\_tab, "Teachers")  
  
 self.teachers\_gbox = QGroupBox("teacher")  
  
 self.tvbox = QVBoxLayout()  
 self.thbox1 = QHBoxLayout()  
 self.thbox2 = QHBoxLayout()  
  
 self.tvbox.addLayout(self.thbox1)  
 self.tvbox.addLayout(self.thbox2)  
  
 self.thbox1.addWidget(self.teachers\_gbox)  
  
 self.\_create\_teachers\_table()  
  
 self.update\_teachers\_button = QPushButton("Update")  
 self.thbox2.addWidget(self.update\_teachers\_button)  
 self.update\_teachers\_button.clicked.connect(self.\_update\_shedule)  
  
 self.teachers\_tab.setLayout(self.tvbox)  
  
 def \_create\_teachers\_table(self):  
 self.teachers\_table = QTableWidget()  
 self.teachers\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.teachers\_table.setColumnCount(5)  
 self.teachers\_table.setHorizontalHeaderLabels(["id", "full\_name", "subject"])  
  
 self.\_update\_teachers\_table()  
  
 self.ttvbox = QVBoxLayout()  
 self.ttvbox.addWidget(self.teachers\_table)  
 self.teachers\_gbox.setLayout(self.ttvbox)  
  
 def \_update\_teachers\_table(self):  
 self.cursor.execute("SELECT \* FROM teacher ORDER BY id")  
 records = list(self.cursor.fetchall())  
  
 self.teachers\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 joinButton = QPushButton("Join")  
 deleteButton = QPushButton("Delete")  
  
 self.teachers\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.teachers\_table.setItem(i, 1, QTableWidgetItem(str(r[1])))  
 self.teachers\_table.setItem(i, 2, QTableWidgetItem(str(r[2])))  
 self.teachers\_table.setCellWidget(i, 3, joinButton)  
 self.teachers\_table.setCellWidget(i, 4, deleteButton)  
  
 joinButton.clicked.connect(lambda ch, num=i: self.\_edit\_teachers\_table(num))  
 deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_from\_teachers\_table(num))  
  
 joinButton = QPushButton("Join")  
 self.teachers\_table.setCellWidget(len(records), 3, joinButton)  
 joinButton.clicked.connect(lambda ch, num=len(records): self.\_add\_to\_teachers\_table(num))  
  
 self.teachers\_table.resizeRowsToContents()  
  
 def \_edit\_teachers\_table(self, num):  
 row = list()  
 for i in range(self.teachers\_table.columnCount()):  
 try:  
 row.append(self.teachers\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["id", "full\_name", "subject"]  
 for i in range(1, 2):  
 self.cursor.execute(f"UPDATE teacher SET {columns[i]} = '{row[i]}' WHERE id = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_delete\_from\_teachers\_table(self, num):  
 row = list()  
 for i in range(self.teachers\_table.columnCount()):  
 try:  
 row.append(self.teachers\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 self.cursor.execute(f"DELETE from teacher where id = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_add\_to\_teachers\_table(self, num):  
 row = list()  
 for i in range(self.teachers\_table.columnCount()):  
 try:  
 row.append(self.teachers\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["id", "full\_name", "subject"]  
 self.cursor.execute(  
 f"INSERT INTO teacher({columns[0]}, {columns[1]}, {columns[2]}) values('{row[0]}', '{row[1]}', '{row[2]}')")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_create\_subjects\_tab(self):  
 self.subjects\_tab = QWidget()  
 self.tabs.addTab(self.subjects\_tab, "Subjects")  
  
 self.subjects\_gbox = QGroupBox("subject")  
  
 self.fvbox = QVBoxLayout()  
 self.fhbox1 = QHBoxLayout()  
 self.fhbox2 = QHBoxLayout()  
  
 self.fvbox.addLayout(self.fhbox1)  
 self.fvbox.addLayout(self.fhbox2)  
  
 self.fhbox1.addWidget(self.subjects\_gbox)  
  
 self.\_create\_subjects\_table()  
  
 self.update\_subjects\_button = QPushButton("Update")  
 self.fhbox2.addWidget(self.update\_subjects\_button)  
 self.update\_subjects\_button.clicked.connect(self.\_update\_shedule)  
  
 self.subjects\_tab.setLayout(self.fvbox)  
  
 def \_create\_subjects\_table(self):  
 self.subjects\_table = QTableWidget()  
 self.subjects\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)  
  
 self.subjects\_table.setColumnCount(3)  
 self.subjects\_table.setHorizontalHeaderLabels(["name"])  
  
 self.\_update\_subjects\_table()  
  
 self.ffvbox = QVBoxLayout()  
 self.ffvbox.addWidget(self.subjects\_table)  
 self.subjects\_gbox.setLayout(self.ffvbox)  
  
 def \_update\_subjects\_table(self):  
 self.cursor.execute("SELECT \* FROM subject ORDER BY name")  
 records = list(self.cursor.fetchall())  
  
 self.subjects\_table.setRowCount(len(records) + 1)  
  
 for i, r in enumerate(records):  
 r = list(r)  
 joinButton = QPushButton("Join")  
 deleteButton = QPushButton("Delete")  
  
 self.subjects\_table.setItem(i, 0, QTableWidgetItem(str(r[0])))  
 self.subjects\_table.setCellWidget(i, 1, joinButton)  
 self.subjects\_table.setCellWidget(i, 2, deleteButton)  
  
 joinButton.clicked.connect(lambda ch, num=i: self.\_edit\_subjects\_table(num))  
 deleteButton.clicked.connect(lambda ch, num=i: self.\_delete\_from\_subjects\_table(num))  
  
 joinButton = QPushButton("Join")  
 self.subjects\_table.setCellWidget(len(records), 1, joinButton)  
 joinButton.clicked.connect(lambda ch, num=len(records): self.\_add\_to\_subjects\_table(num))  
  
 self.schedule\_table.resizeRowsToContents()  
  
 def \_edit\_subjects\_table(self, num):  
 row = list()  
 for i in range(self.subjects\_table.columnCount()):  
 try:  
 row.append(self.subjects\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["name"]  
 self.cursor.execute(f"UPDATE subject SET {columns[0]} = '{row[0]}' WHERE name = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_delete\_from\_subjects\_table(self, num):  
 row = list()  
 row2 = list()  
 row3 = list()  
 for i in range(self.subjects\_table.columnCount()):  
 try:  
 row.append(self.subjects\_table.item(num, i).text())  
 except:  
 row.append(None)  
 for i in range(self.schedule\_table.columnCount()):  
 try:  
 row2.append(self.schedule\_table.item(i, 2).text())  
 except:  
 row2.append(None)  
 for i in range(self.teachers\_table.columnCount()):  
 try:  
 row3.append(self.teachers\_table.item(i, 2).text())  
 except:  
 row3.append(None)  
 if row[0] in row3 or row[0] in row2:  
 QMessageBox.about(self, "!", "Данный предмет нельзя удалить")  
 return  
 print(row, num)  
 try:  
 self.cursor.execute(f"DELETE from subject where name = '{row[0]}'")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_add\_to\_subjects\_table(self, num):  
 row = list()  
 for i in range(self.subjects\_table.columnCount()):  
 try:  
 row.append(self.subjects\_table.item(num, i).text())  
 except:  
 row.append(None)  
 print(row, num)  
 try:  
 columns = ["name"]  
 self.cursor.execute(f"INSERT INTO subject({columns[0]}) values('{row[0]}')")  
 self.conn.commit()  
 except:  
 QMessageBox.about(self, "Error", "Enter all fields")  
  
 def \_update\_shedule(self):  
 self.\_update\_schedule\_table()  
 self.\_update\_teachers\_table()  
 self.\_update\_subjects\_table()  
  
  
app = QApplication(sys.argv)  
win = MainWindow()  
win.show()  
sys.exit(app.exec\_())