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 «Математической кибернетики и информационных технологий»

ВВИТ

Лабораторная работа № 8

Выполнила:

студентка группы БВТ2205

Евтушенко Д. А.

**Подключение библиотек.**

import psycopg2

import sys

import datetime

from datetime import date

from PyQt5.QtWidgets import (QApplication, QWidget,

QTabWidget, QAbstractScrollArea,

QVBoxLayout, QHBoxLayout,

QTableWidget, QGroupBox,

QTableWidgetItem, QPushButton,

QMessageBox, QInputDialog)

**Проверка чётности недели: если неделя чётная, то одно расписание, в противном случае другое**

today = date.today()

num = int(today.isocalendar().week)

if (num % 2) == 0:

this\_week = "timetable\_week2"

else:

this\_week = "timetable\_week1"

**Создание окна.**

class MainWindow(QWidget):

def \_\_init\_\_(self):

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

self.\_connect\_to\_db()

**Установка названия окна.**

self.setWindowTitle("BVT2205 Information")

self.vbox = QVBoxLayout(self)

self.tabs = QTabWidget(self)

self.vbox.addWidget(self.tabs)

self.\_create\_schedule\_tab()

self.\_create\_teacher\_tab()

self.\_create\_timetable\_week1\_tab()

self.\_create\_timetable\_week2\_tab()

self.rowSelected = None

self.idSelected = None

**Подключение к БД.**

def \_connect\_to\_db(self):

self.conn = psycopg2.connect(database="timestable\_db",

user="postgres",

password="password",

host="localhost",

port="5433")

self.cursor = self.conn.cursor()

**Таблица с преподавателями.**

def \_create\_teacher\_tab(self):

self.teacher\_tab = QWidget()

self.tabs.addTab(self.teacher\_tab, "Teacher")

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

self.svbox = QVBoxLayout()

self.shbox1 = QHBoxLayout()

self.shbox2 = QHBoxLayout()

self.svbox.addLayout(self.shbox1)

self.svbox.addLayout(self.shbox2)

self.shbox1.addWidget(self.teacher\_gbox)

self.\_create\_teacher\_table()

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

self.shbox2.addWidget(self.update\_teacher\_button)

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

self.shboxa = QHBoxLayout()

self.shbox1.addLayout(self.shboxa)

self.alter\_teacher\_button = QPushButton("Alter")

self.shboxa.addWidget(self.alter\_teacher\_button)

self.alter\_teacher\_button.clicked.connect(lambda ch: self.update\_teacher\_info('Alter'))

self.shboxd = QHBoxLayout()

self.shbox1.addLayout(self.shboxd)

self.delete\_teacher\_button = QPushButton("Delete")

self.shboxd.addWidget(self.delete\_teacher\_button)

self.delete\_teacher\_button.clicked.connect(lambda ch: self.update\_teacher\_info('Delete'))

self.shboxrow = QHBoxLayout()

self.shbox1.addLayout(self.shboxrow)

self.add\_teacher\_button = QPushButton("Add Row")

self.shboxrow.addWidget(self.add\_teacher\_button)

self.add\_teacher\_button.clicked.connect(lambda ch: self.update\_teacher\_info('Add Row'))

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

**Отображение таблицы с преподавателями.**

def \_create\_teacher\_table(self):

self.teacher\_table = QTableWidget()

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

self.teacher\_table.setColumnCount(3)

self.teacher\_table.setHorizontalHeaderLabels(["Full Name", "Subject", ""])

self.\_update\_teacher\_table()

self.mvbox = QVBoxLayout()

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

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

**Создание и заполнение таблицы с преподавателями.**

def \_update\_teacher\_table(self):

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

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

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

for i, r in enumerate(records):

r = list(r)

updateTeach = QPushButton("Select")

self.teacher\_table.setItem(i, 0,

QTableWidgetItem(str(r[1])))

self.teacher\_table.setItem(i, 1,

QTableWidgetItem(str(r[2])))

self.teacher\_table.setItem(len(records), 0, QTableWidgetItem(str('')))

self.teacher\_table.setItem(len(records), 1, QTableWidgetItem(str('')))

self.teacher\_table.setCellWidget(i, 2, updateTeach)

updateTeach.clicked.connect(lambda ch, num=i, id=r[0]: self.select\_row(num, id))

selectTeach = QPushButton("Select")

self.teacher\_table.setCellWidget(len(records),2, selectTeach)

selectTeach.clicked.connect(lambda ch, num=len(records): self.select\_row(num))

self.teacher\_table.resizeRowsToContents()

def update\_teacher\_info(self, query):

**Обработка Alter.**

if query == 'Alter':

self.cursor.execute("select count(full\_name) from teacher")

records = self.cursor.fetchall()

#print (self.rowSelected)

try:

if records[0][0] == self.rowSelected:

raise Exception

elif records[0][0] > self.rowSelected:

self.cursor.execute("SELECT column\_name FROM information\_schema.columns "

"WHERE table\_schema = 'public' AND table\_name = 'teacher' ")

columns = self.cursor.fetchall()

new\_values = []

for temp in columns[1:]:

text, ok = QInputDialog.getText(self, 'Add new teacher', 'Enter {} value:'.format(temp[0]))

if ok and text != "":

new\_values.append(text)

if len(new\_values) == 2:

try:

self.cursor.execute("update teacher "

"set full\_name = %s, subject = %s "

"where id = {};".format(self.idSelected), tuple(new\_values))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Given subject value "

"does not exist in subject table")

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Select a non empty row first")

**Обработка Delete.**

elif query == 'Delete':

self.cursor.execute("select count(full\_name) from teacher")

records = self.cursor.fetchall()

try:

if records[0][0] == self.rowSelected:

raise Exception

elif records[0][0] > self.rowSelected:

print(self.rowSelected)

self.cursor.execute("delete from teacher where id={}".format(self.idSelected))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Select a non empty row first")

**Обработка Add Row.**

elif query == 'Add Row':

self.cursor.execute("select count(full\_name) from teacher")

records = self.cursor.fetchall()

if records[0][0] == self.rowSelected:

print('Can do')

self.cursor.execute("SELECT column\_name FROM information\_schema.columns "

"WHERE table\_schema = 'public' AND table\_name = 'teacher' ")

columns = self.cursor.fetchall()

new\_values = []

for temp in columns[1:]:

text, ok = QInputDialog.getText(self, 'Add new teacher', 'Enter {} value:'.format(temp[0]))

if ok and text != "":

new\_values.append(text)

if len(new\_values) == 2:

try:

self.cursor.execute("insert into "

"teacher (full\_name, subject) "

"values (%s, %s);", tuple(new\_values))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Given subject value "

"does not exist in subject table")

print(new\_values)

else:

QMessageBox.about(self, "Error", "Select an empty row first")

def \_update\_teacher(self):

self.rowSelected = None

self.idSelected = None

self.\_update\_teacher\_table()

**Отображение таблицы с расписанием первой недели.**

def \_create\_timetable\_week1\_tab(self):

self.timetable\_week1\_tab = QWidget()

self.tabs.addTab(self.timetable\_week1\_tab, "Week1")

self.timetable\_week1\_gbox = QGroupBox("Week1")

self.svbox = QVBoxLayout()

self.shbox1 = QHBoxLayout()

self.shbox2 = QHBoxLayout()

self.svbox.addLayout(self.shbox1)

self.svbox.addLayout(self.shbox2)

self.shbox1.addWidget(self.timetable\_week1\_gbox)

self.\_create\_timetable\_week1\_table()

self.update\_timetable\_week1\_button = QPushButton("Update")

self.shbox2.addWidget(self.update\_timetable\_week1\_button)

self.update\_timetable\_week1\_button.clicked.connect(self.\_update\_timetable\_week1)

self.timetable\_week1\_tab.setLayout(self.svbox)

def \_create\_timetable\_week1\_table(self):

self.timetable\_week1\_table = QTableWidget()

self.timetable\_week1\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

self.timetable\_week1\_table.setColumnCount(2)

self.timetable\_week1\_table.setHorizontalHeaderLabels(["Day", "Lessons", ""])

self.\_update\_timetable\_week1\_table()

self.mvbox = QVBoxLayout()

self.mvbox.addWidget(self.timetable\_week1\_table)

self.timetable\_week1\_gbox.setLayout(self.mvbox)

def \_update\_timetable\_week1\_table(self):

self.cursor.execute(

"select day, string\_agg(table\_column, '\n\n') as table\_row from (select day, timetable\_week1.subject ||' | '|| room\_numb ||' | '|| start\_time ||'-'|| finish\_time ||' | '|| full\_name as table\_column from timetable\_week1, teacher where teacher.subject = timetable\_week1.subject order by start\_time)timetable\_week1 group by 1 order by case when day = 'Monday' then 1 when day = 'Tuesday' then 2 when day = 'Wednesday' then 3 when day = 'Thursday' then 4 else 5 end;")

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

self.timetable\_week1\_table.setRowCount(len(records))

for i, r in enumerate(records):

r = list(r)

self.timetable\_week1\_table.setItem(i, 0,

QTableWidgetItem(str(r[0])))

self.timetable\_week1\_table.setItem(i, 1,

QTableWidgetItem(str(r[1])))

self.timetable\_week1\_table.resizeRowsToContents()

def \_update\_timetable\_week1(self):

self.rowSelected = None

self.idSelected = None

self.\_update\_timetable\_week1\_table()

**Отображение таблицы с расписанием второй недели.**

def \_create\_timetable\_week2\_tab(self):

self.timetable\_week2\_tab = QWidget()

self.tabs.addTab(self.timetable\_week2\_tab, "Week2")

self.timetable\_week2\_gbox = QGroupBox("Week2")

self.svbox = QVBoxLayout()

self.shbox1 = QHBoxLayout()

self.shbox2 = QHBoxLayout()

self.svbox.addLayout(self.shbox1)

self.svbox.addLayout(self.shbox2)

self.shbox1.addWidget(self.timetable\_week2\_gbox)

self.\_create\_timetable\_week2\_table()

self.update\_timetable\_week2\_button = QPushButton("Update")

self.shbox2.addWidget(self.update\_timetable\_week2\_button)

self.update\_timetable\_week2\_button.clicked.connect(self.\_update\_timetable\_week2)

self.timetable\_week2\_tab.setLayout(self.svbox)

def \_create\_timetable\_week2\_table(self):

self.timetable\_week2\_table = QTableWidget()

self.timetable\_week2\_table.setSizeAdjustPolicy(QAbstractScrollArea.AdjustToContents)

self.timetable\_week2\_table.setColumnCount(2)

self.timetable\_week2\_table.setHorizontalHeaderLabels(["Day", "Lessons", ""])

self.\_update\_timetable\_week2\_table()

self.mvbox = QVBoxLayout()

self.mvbox.addWidget(self.timetable\_week2\_table)

self.timetable\_week2\_gbox.setLayout(self.mvbox)

def \_update\_timetable\_week2\_table(self):

self.cursor.execute(

"select day, string\_agg(table\_column, '\n\n') as table\_row from (select day, timetable\_week2.subject ||' | '|| room\_numb ||' | '|| start\_time ||'-'|| finish\_time ||' | '|| full\_name as table\_column from timetable\_week2, teacher where teacher.subject = timetable\_week2.subject order by start\_time)timetable\_week2 group by 1 order by case when day = 'Monday' then 1 when day = 'Tuesday' then 2 when day = 'Wednesday' then 3 when day = 'Thursday' then 4 else 5 end;")

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

self.timetable\_week2\_table.setRowCount(len(records))

for i, r in enumerate(records):

r = list(r)

self.timetable\_week2\_table.setItem(i, 0,

QTableWidgetItem(str(r[0])))

self.timetable\_week2\_table.setItem(i, 1,

QTableWidgetItem(str(r[1])))

self.timetable\_week2\_table.resizeRowsToContents()

def \_update\_timetable\_week2(self):

self.rowSelected = None

self.idSelected = None

self.\_update\_timetable\_week2\_table()

**Отображение расписания по дням.**

def \_create\_schedule\_tab(self):

self.day = 'Monday'

self.schedule\_tab = QWidget()

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

self.schedule\_gbox = QGroupBox("{}".format(self.day))

self.svbox = QVBoxLayout()

self.shbox1 = QHBoxLayout()

self.shbox2 = QHBoxLayout()

self.svbox.addLayout(self.shbox1)

self.shbox1.addWidget(self.schedule\_gbox)

self.\_create\_schedule\_table()

self.svbox.addLayout(self.shbox2)

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

self.shbox2.addWidget(self.update\_schedule\_button)

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

self.shboxm = QHBoxLayout()

self.svbox.addLayout(self.shboxm)

self.monday\_schedule\_button = QPushButton("Monday")

self.shboxm.addWidget(self.monday\_schedule\_button)

self.monday\_schedule\_button.clicked.connect(lambda ch: self.btnstate('Monday'))

self.shboxt = QHBoxLayout()

self.shboxm.addLayout(self.shboxt)

self.tuesday\_schedule\_button = QPushButton("Tuesday")

self.shboxt.addWidget(self.tuesday\_schedule\_button)

self.tuesday\_schedule\_button.clicked.connect(lambda ch: self.btnstate('Tuesday'))

self.shboxw = QHBoxLayout()

self.shboxm.addLayout(self.shboxw)

self.wednesday\_schedule\_button = QPushButton("Wednesday")

self.shboxw.addWidget(self.wednesday\_schedule\_button)

self.wednesday\_schedule\_button.clicked.connect(lambda ch: self.btnstate('Wednesday'))

self.shboxth = QHBoxLayout()

self.shboxm.addLayout(self.shboxth)

self.thursday\_schedule\_button = QPushButton("Thursday")

self.shboxth.addWidget(self.thursday\_schedule\_button)

self.thursday\_schedule\_button.clicked.connect(lambda ch: self.btnstate('Thursday'))

self.shboxf = QHBoxLayout()

self.shboxm.addLayout(self.shboxf)

self.friday\_schedule\_button = QPushButton("Friday")

self.shboxf.addWidget(self.friday\_schedule\_button)

self.friday\_schedule\_button.clicked.connect(lambda ch: self.btnstate('Friday'))

self.shboxa = QHBoxLayout()

self.shbox1.addLayout(self.shboxa)

self.alter\_lesson\_button = QPushButton("Alter")

self.shboxa.addWidget(self.alter\_lesson\_button)

self.alter\_lesson\_button.clicked.connect(lambda ch: self.update\_lesson('Alter'))

self.shboxd = QHBoxLayout()

self.shbox1.addLayout(self.shboxd)

self.delete\_lesson\_button = QPushButton("Delete")

self.shboxd.addWidget(self.delete\_lesson\_button)

self.delete\_lesson\_button.clicked.connect(lambda ch: self.update\_lesson('Delete'))

self.shboxrow = QHBoxLayout()

self.shbox1.addLayout(self.shboxrow)

self.add\_row\_button = QPushButton("Add Row")

self.shboxrow.addWidget(self.add\_row\_button)

self.add\_row\_button.clicked.connect(lambda ch: self.update\_lesson('Add Row'))

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

**Отображение расписания на день.**

def \_create\_schedule\_table(self):

self.schedule\_table = QTableWidget()

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

self.schedule\_table.setColumnCount(5)

self.schedule\_table.setHorizontalHeaderLabels(["Subject", "Room numb", "Start time", "Finish time", "", ""])

self.\_update\_schedule\_table()

self.mvbox = QVBoxLayout()

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

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

def btnstate(self, wday):

self.day = wday

def \_update\_schedule\_table(self):

self.cursor.execute(

"SELECT subject, room\_numb, start\_time, finish\_time, id FROM {} WHERE day = '{}'".format(this\_week, self.day))

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

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

self.schedule\_gbox.setTitle(self.day)

for i, r in enumerate(records):

r = list(r)

# updateButton = QPushButton("Update")

# deleteButton = QPushButton("Delete")

# addRow = QPushButton("Add Row")

selectRow = QPushButton("Select")

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(len(records), 0, QTableWidgetItem(str('')))

self.schedule\_table.setItem(len(records), 1, QTableWidgetItem(str('')))

self.schedule\_table.setItem(len(records), 2, QTableWidgetItem(str('')))

self.schedule\_table.setItem(len(records), 3, QTableWidgetItem(str('')))

self.schedule\_table.setCellWidget(i, 4, selectRow)

selectRow.clicked.connect(lambda ch, num=i, id=r[4]: self.select\_row(num, id))

selectRow = QPushButton("Select")

self.schedule\_table.setCellWidget(len(records), 4, selectRow)

selectRow.clicked.connect(lambda ch, num=len(records): self.select\_row(num))

self.schedule\_table.resizeRowsToContents()

def select\_row(self, numRow, \*numId):

self.rowSelected = numRow

if numId:

self.idSelected = numId[0]

print (self.idSelected)

print (self.rowSelected)

def update\_lesson(self, query):

**Обработка команды Alter.**

if query == 'Alter':

print('alter')

self.cursor.execute("select count(day) from {} where day = %s".format(this\_week), (self.day,))

records = self.cursor.fetchall()

print (self.rowSelected)

try:

if records[0][0] == self.rowSelected:

raise Exception

elif records[0][0] > self.rowSelected:

new\_values = []

self.cursor.execute("SELECT column\_name FROM information\_schema.columns "

"WHERE table\_schema = 'public' AND table\_name = '{}' ".format(this\_week))

columns = self.cursor.fetchall()

for temp in columns[2:]:

text, ok = QInputDialog.getText(self, 'Alter in timetable', 'Enter {} value:'.format(temp[0]))

if ok and text != "":

new\_values.append(text)

if len(new\_values) == 4:

try:

print(new\_values)

self.cursor.execute("update {} set subject = %s, room\_numb= %s, start\_time = %s, finish\_time = %s where id= {}".format(this\_week, self.idSelected), tuple(new\_values))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Given subject value "

"does not exist in subject table")

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Select a non empty row first")

**Обработка команды Delete.**

elif query == 'Delete':

self.cursor.execute("select count(day) from {} where day = %s".format(this\_week), (self.day,))

records = self.cursor.fetchall()

print (self.rowSelected)

try:

if records[0][0] == self.rowSelected:

raise Exception

elif records[0][0] > self.rowSelected:

self.cursor.execute("delete from {} where id={}".format(this\_week, self.idSelected))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Select a non empty row first")

**Обработка команды Add Row.**

elif query == 'Add Row':

self.cursor.execute("select count(day) from {} where day = %s".format(this\_week), (self.day,))

records = self.cursor.fetchall()

if records[0][0] == self.rowSelected:

print('Can do')

self.cursor.execute("SELECT column\_name FROM information\_schema.columns "

"WHERE table\_schema = 'public' AND table\_name = '{}' ".format(this\_week))

columns = self.cursor.fetchall()

new\_values = [self.day]

for temp in columns[2:]:

text, ok = QInputDialog.getText(self, 'Add in timetable', 'Enter {} value:'.format(temp[0]))

if ok and text != "":

new\_values.append(text)

if len(new\_values) == 5:

try:

self.cursor.execute("insert into "

"{}(day, subject, room\_numb, start\_time, finish\_time) "

"values (%s, %s, %s, %s, %s);".format(this\_week), tuple(new\_values))

self.conn.commit()

except:

self.conn.commit()

QMessageBox.about(self, "Error", "Given subject value "

"does not exist in subject table")

print(new\_values)

else:

QMessageBox.about(self, "Error", "Select an empty row first")

def \_update\_schedule(self):

self.rowSelected = None

self.idSelected = None

self.\_update\_schedule\_table()

app = QApplication(sys.argv)

win = MainWindow()

win.show()

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