Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Вятский государственный университет»

Колледж ВятГУ

**ОТЧЕТ**

**ПО ДОМАШНЕЙ КОНТРОЛЬНОЙ РАБОТЕ №9**

**«Работа с типизированными файлами»**

**ПО МДК 05.02 РАЗРАБОТКА КОДА ИНФОРМАЦИОННЫХ СИСТЕМ**

Выполнил: студент учебной группы

ИСПк-203-52-00

Широкова Анна Михайловна

Преподаватель:

Сергеева Елизавета Григорьевна

Киров

2023

**Цель работы:** получить представление об организации хранения и использования информации посредством типизированных файлов, закрепить навыки создания пользовательского интерфейса.

**Скриншот задания** показан на рисунке 1.

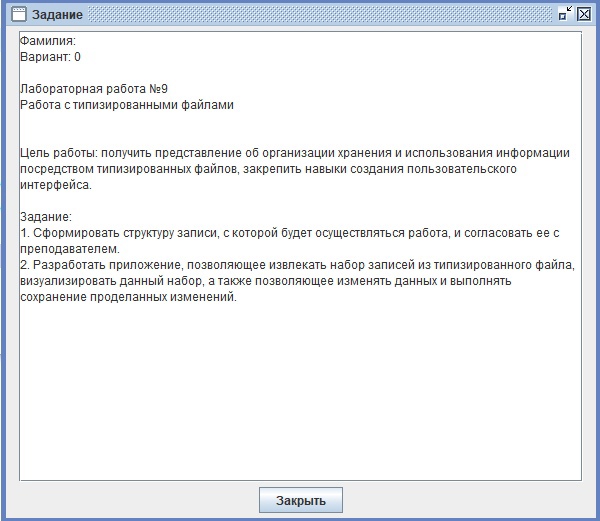


Рисунок 1 – Скриншот задания

**Описание алгоритма:**

Создаем экранную форму при помощи Qt Designer. Подключаемся к базе данных. Создаем функции для работы с базой данных: чтение, запись, удаление и редактирование информации в базе данных.

**Код программы:**

# -\*- coding: cp1251 -\*-

from PyQt5 import QtCore, QtGui, QtWidgets

from PyQt5.QtGui import QIcon

from PyQt5.QtWidgets import \*

from PyQt5.uic import loadUiType

import sqlite3

import sys

menu\_ui, \_ = loadUiType('menu.ui')

groupe\_ui, \_ = loadUiType('groupe.ui')

shedule\_ui, \_= loadUiType('shedule.ui')

teachers\_ui, \_= loadUiType('teachers.ui')

subjects\_ui, \_= loadUiType('subjects.ui')

journal\_ui, \_= loadUiType('journal.ui')

students\_ui, \_= loadUiType('students.ui')

timepair\_ui, \_ = loadUiType('timepair.ui')

class MainAppMenu(QMainWindow,menu\_ui):

def \_\_init\_\_(self, parent=None):

super(MainAppMenu, self).\_\_init\_\_(parent)

QMainWindow.\_\_init\_\_(self)

self.setupUi(self)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/menu-bar.png'))

self.groupe.clicked.connect(self.open\_groupe\_form)

self.btn\_students.clicked.connect(self.open\_students\_form)

self.teacher\_btn.clicked.connect(self.open\_teachers\_form)

self.subjects\_btn.clicked.connect(self.open\_subjects\_form)

self.journal\_btn.clicked.connect(self.open\_journal\_form)

def open\_groupe\_form(self):

self.close()

self.groupe\_form = MainAppGroupe()

self.groupe\_form.show()

def open\_students\_form(self):

self.close()

self.students\_form = MainAppStudents()

self.students\_form.show()

def open\_teachers\_form(self):

self.close()

self.teachers\_form = MainAppTeachers()

self.teachers\_form.show()

def open\_subjects\_form(self):

self.close()

self.subjects\_form = MainAppSubjects()

self.subjects\_form.show()

def open\_journal\_form(self):

self.close()

self.journal\_form = MainAppJournal()

self.journal\_form.show()

class MainAppGroupe(QMainWindow, groupe\_ui):

def \_\_init\_\_(self, parent=None):

super(MainAppGroupe, self).\_\_init\_\_(parent)

QMainWindow.\_\_init\_\_(self)

self.setupUi(self)

self.setWindowIcon(QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/people.png'))

self.move(QApplication.desktop().screen().rect().center() - self.rect().center())

self.handles\_btn()

self.db\_connect()

self.load\_data()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if self.cur:

self.message\_box("Сообщение проверки", "База данных db подключена")

else:

self.message\_box("Сообщение проверки", "База данных db не подключена")

def message\_box(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.btn\_Insert.clicked.connect(self.add\_group)

self.Search\_btn\_sub.clicked.connect(self.search\_group)

self.deletezapic\_2.clicked.connect(self.delete\_group)

self.clears.clicked.connect(self.clear)

self.update\_gr.clicked.connect(self.update\_group)

self.vuvod.clicked.connect(self.load\_data)

self.cleant.clicked.connect(self.clear\_table)

self.back\_to\_menu.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def add\_group(self):

ID = self.lineEdit\_2.text()

number = self.lineEdit\_3.text()

speciality = self.lineEdit\_4.text()

if ID and number and speciality != "":

self.cur.execute('SELECT \* FROM groups WHERE ID=?', (ID,))

existing\_record = self.cur.fetchone()

if existing\_record:

self.message\_box("Сообщение", "Запись с таким ID уже существует")

else:

self.cur.execute('''

INSERT INTO groups (ID, number, speciality)

VALUES(?,?,?)''', (ID, number, speciality))

self.db.commit()

self.message\_box("Сообщение", "Запись группы была успешно добавлена")

self.load\_data()

else:

self.message\_box("Сообщение", "Введите данные для записи в базу данных")

def search\_group(self):

ID = self.lineEdit\_seatch.text()

sql = '''SELECT \* FROM groups WHERE ID=?'''

self.cur.execute(sql, [(ID)])

data = self.cur.fetchone()

if ID == "":

self.message\_box("Сообщение", "Введите ID для поиска записи")

elif data:

self.lineEdit\_2.setText(str(data[0]))

self.lineEdit\_3.setText(str(data[1]))

self.lineEdit\_4.setText(str(data[2]))

else:

self.message\_box("Сообщение", "Нет записи в БД")

def delete\_group(self):

ID = self.lineEdit\_2.text()

data = self.cur.fetchone()

sql = '''DELETE FROM groups WHERE ID=?'''

self.cur.execute(sql, [(ID)])

self.db.commit()

self.clear()

self.load\_data()

def clear(self):

self.lineEdit\_2.setText("")

self.lineEdit\_3.setText("")

self.lineEdit\_4.setText("")

self.lineEdit\_seatch.setText("")

def clear\_table(self):

self.table.setRowCount(0)

def update\_group(self):

ID = self.lineEdit\_2.text()

if ID:

ID = self.lineEdit\_2.text()

number = self.lineEdit\_3.text()

speciality = self.lineEdit\_4.text()

self.cur.execute('''

UPDATE groups SET ID=?, number=?, speciality=? WHERE ID=?

''', (ID, number, speciality, ID))

self.db.commit()

self.clear()

self.message\_box("Сообщение", "Запись обновлена")

self.load\_data()

else:

self.message\_box("Сообщение", "Запись не обновлена")

def load\_data(self):

connection = sqlite3.connect('db.sqlite')

cur = connection.cursor()

self.cur.execute('SELECT \* FROM groups LIMIT 300')

rows = self.cur.fetchall()

self.table.setRowCount(len(rows))

for row\_num, row\_data in enumerate(rows):

for col\_num, data in enumerate(row\_data):

self.table.setItem(row\_num, col\_num, QtWidgets.QTableWidgetItem(str(data)))

self.table.resizeColumnsToContents()

pass

class MainAppJournal(QMainWindow, ui):

def \_\_init\_\_(self, parent=None):

super(MainAppJournal, self).\_\_init\_\_(parent)

self.setupUi(self)

self.db\_connect()

self.handles\_btn()

self.loaddata()

self.table\_\_att.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_\_att.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Fixed)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/bl.jpg'))

self.move(QApplication.desktop().screen().rect().center() - self.rect().center())

self.shedule\_btn.clicked.connect(self.open\_shedule\_form)

def open\_shedule\_form(self):

self.close()

self.shedule\_form = MainAppShedule()

self.shedule\_form.show()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if self.cur:

self.messagebox("Сообщение проверки", "База данных db подключена")

else:

self.messagebox("Сообщение проверки", "База данных db не подключена")

def messagebox(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.btn\_Insert\_\_att.clicked.connect(self.Add\_Journal)

self.Search\_btn\_att.clicked.connect(self.Search\_Journal)

self.deletezapic\_\_att.clicked.connect(self.Delete\_Journal)

self.clears\_\_att.clicked.connect(self.Clear)

self.update\_\_\_att.clicked.connect(self.Update\_Journal)

self.vuvod\_\_\_att.clicked.connect(self.loaddata)

self.cleant\_v\_att.clicked.connect(self.clear\_table)

self.back\_to\_menu.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def Add\_Journal(self):

ID = self.lineEdit\_2\_att.text()

if not ID:

self.messagebox("Сообщение", "Введите данные для записи в базу данных")

return

# Проверка повторений записи

sql\_check = "SELECT \* FROM attendance WHERE ID = ?"

self.cur.execute(sql\_check, [(ID)])

data = self.cur.fetchone()

if data:

self.messagebox("Сообщение", "Запись с таким ID уже существует")

return

student\_id = self.lineEdit\_7\_att.text()

shedule\_id = self.lineEdit\_4\_att.text()

pair1 = self.lineEdit\_att.text()

pair2 = self.lineEdit\_6\_\_\_att.text()

pair3 = self.lineEdit\_3\_att.text()

pair4 = self.lineEdit\_3\_\_att.text()

mark = self.lineEdit\_3\_\_att\_2.text()

comment = self.lineEdit\_8\_\_att.text()

self.cur.execute('''

INSERT INTO attendance (ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)''',

(ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment))

self.db.commit()

self.messagebox("Сообщение", "Запись журнала была успешно добавлена")

self.Clear()

self.loaddata()

def Search\_Journal(self):

ID = self.lineEdit\_seatch\_\_att.text()

sql = '''SELECT \* FROM attendance WHERE ID = ?'''

self.cur.execute(sql, [(ID)])

data = self.cur.fetchone()

if ID == "":

self.messagebox("Сообщение", "Введите ID для поиска записи")

elif data:

self.lineEdit\_2\_att.setText(str(data[0]))

self.lineEdit\_7\_att.setText(str(data[1]))

self.lineEdit\_4\_att.setText(str(data[2]))

self.lineEdit\_att.setText(str(data[3]))

self.lineEdit\_6\_\_\_att.setText(str(data[4]))

self.lineEdit\_3\_att.setText(str(data[5]))

self.lineEdit\_3\_\_att.setText(str(data[6]))

self.lineEdit\_3\_\_att\_2.setText(str(data[7]))

self.lineEdit\_8\_\_att.setText(str(data[8]))

else:

self.messagebox("Сообщение", "Нет записи в БД")

def Delete\_Journal(self):

ID = self.lineEdit\_2\_att.text()

sql = '''DELETE FROM attendance WHERE ID = ?'''

self.cur.execute(sql, [(ID)])

self.db.commit()

self.Clear()

self.loaddata()

def Clear(self):

self.lineEdit\_2\_att.setText("")

self.lineEdit\_7\_att.setText("")

self.lineEdit\_4\_att.setText("")

self.lineEdit\_att.setText("")

self.lineEdit\_6\_\_\_att.setText("")

self.lineEdit\_3\_att.setText("")

self.lineEdit\_3\_\_att.setText("")

self.lineEdit\_3\_\_att\_2.setText("")

self.lineEdit\_8\_\_att.setText("")

self.lineEdit\_seatch\_\_att.setText("")

def clear\_table(self):

self.table\_\_att.clearContents()

self.table\_\_att.setRowCount(0)

def Update\_Journal(self):

ID = self.lineEdit\_2\_att.text()

if not ID:

self.messagebox("Сообщение", "Запись не обновлена")

return

student\_id = self.lineEdit\_7\_att.text()

shedule\_id = self.lineEdit\_4\_att.text()

pair1 = self.lineEdit\_att.text()

pair2 = self.lineEdit\_6\_\_\_att.text()

pair3 = self.lineEdit\_3\_att.text()

pair4 = self.lineEdit\_3\_\_att.text()

mark = self.lineEdit\_3\_\_att\_2.text()

comment = self.lineEdit\_8\_\_att.text()

self.cur.execute('''

UPDATE attendance SET ID = ?, student\_id = ?, shedule\_id = ?, pair1 = ?, pair2 = ?, pair3 = ?, pair4 = ?, mark = ?, comment = ?

WHERE ID = ?''',

(ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment, ID))

self.db.commit()

self.Clear()

self.messagebox("Сообщение", "Запись обновлена")

self.loaddata()

def loaddata(self):

self.table\_\_att.clearContents()

connection = sqlite3.connect('db.sqlite')

cur = connection.cursor()

sql = '''SELECT \* FROM attendance LIMIT 500'''

cur.execute(sql)

data = cur.fetchall()

self.table\_\_att.setRowCount(len(data)) # Устанавливаем количество строк таблицы равным количеству записей из базы данных

for row\_index, row\_data in enumerate(data):

for column\_index, cell\_data in enumerate(row\_data):

item = QtWidgets.QTableWidgetItem(str(cell\_data))

self.table\_\_att.setItem(row\_index, column\_index, item)

self.table\_\_att.resizeColumnsToContents() # Растягиваем столбцы таблицы для лучшего отображения данных

class MainAppJournal(QMainWindow, ui):

def \_\_init\_\_(self, parent=None):

super(MainAppJournal, self).\_\_init\_\_(parent)

self.setupUi(self)

self.db\_connect()

self.handles\_btn()

self.loaddata()

self.table\_\_att.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_\_att.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Fixed)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/bl.jpg'))

self.move(QApplication.desktop().screen().rect().center() - self.rect().center())

self.shedule\_btn.clicked.connect(self.open\_shedule\_form)

def open\_shedule\_form(self):

self.close()

self.shedule\_form = MainAppShedule()

self.shedule\_form.show()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if self.cur:

self.messagebox("Сообщение проверки", "База данных db подключена")

else:

self.messagebox("Сообщение проверки", "База данных db не подключена")

def messagebox(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.btn\_Insert\_\_att.clicked.connect(self.Add\_Journal)

self.Search\_btn\_att.clicked.connect(self.Search\_Journal)

self.deletezapic\_\_att.clicked.connect(self.Delete\_Journal)

self.clears\_\_att.clicked.connect(self.Clear)

self.update\_\_\_att.clicked.connect(self.Update\_Journal)

self.vuvod\_\_\_att.clicked.connect(self.loaddata)

self.cleant\_v\_att.clicked.connect(self.clear\_table)

self.back\_to\_menu.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def Add\_Journal(self):

ID = self.lineEdit\_2\_att.text()

if not ID:

self.messagebox("Сообщение", "Введите данные для записи в базу данных")

return

# Проверка повторений записи

sql\_check = "SELECT \* FROM attendance WHERE ID = ?"

self.cur.execute(sql\_check, [(ID)])

data = self.cur.fetchone()

if data:

self.messagebox("Сообщение", "Запись с таким ID уже существует")

return

student\_id = self.lineEdit\_7\_att.text()

shedule\_id = self.lineEdit\_4\_att.text()

pair1 = self.lineEdit\_att.text()

pair2 = self.lineEdit\_6\_\_\_att.text()

pair3 = self.lineEdit\_3\_att.text()

pair4 = self.lineEdit\_3\_\_att.text()

mark = self.lineEdit\_3\_\_att\_2.text()

comment = self.lineEdit\_8\_\_att.text()

self.cur.execute('''

INSERT INTO attendance (ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)''',

(ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment))

self.db.commit()

self.messagebox("Сообщение", "Запись журнала была успешно добавлена")

self.Clear()

self.loaddata()

def Search\_Journal(self):

ID = self.lineEdit\_seatch\_\_att.text()

sql = '''SELECT \* FROM attendance WHERE ID = ?'''

self.cur.execute(sql, [(ID)])

data = self.cur.fetchone()

if ID == "":

self.messagebox("Сообщение", "Введите ID для поиска записи")

elif data:

self.lineEdit\_2\_att.setText(str(data[0]))

self.lineEdit\_7\_att.setText(str(data[1]))

self.lineEdit\_4\_att.setText(str(data[2]))

self.lineEdit\_att.setText(str(data[3]))

self.lineEdit\_6\_\_\_att.setText(str(data[4]))

self.lineEdit\_3\_att.setText(str(data[5]))

self.lineEdit\_3\_\_att.setText(str(data[6]))

self.lineEdit\_3\_\_att\_2.setText(str(data[7]))

self.lineEdit\_8\_\_att.setText(str(data[8]))

else:

self.messagebox("Сообщение", "Нет записи в БД")

def Delete\_Journal(self):

ID = self.lineEdit\_2\_att.text()

sql = '''DELETE FROM attendance WHERE ID = ?'''

self.cur.execute(sql, [(ID)])

self.db.commit()

self.Clear()

self.loaddata()

def Clear(self):

self.lineEdit\_2\_att.setText("")

self.lineEdit\_7\_att.setText("")

self.lineEdit\_4\_att.setText("")

self.lineEdit\_att.setText("")

self.lineEdit\_6\_\_\_att.setText("")

self.lineEdit\_3\_att.setText("")

self.lineEdit\_3\_\_att.setText("")

self.lineEdit\_3\_\_att\_2.setText("")

self.lineEdit\_8\_\_att.setText("")

self.lineEdit\_seatch\_\_att.setText("")

def clear\_table(self):

self.table\_\_att.clearContents()

self.table\_\_att.setRowCount(0)

def Update\_Journal(self):

ID = self.lineEdit\_2\_att.text()

if not ID:

self.messagebox("Сообщение", "Запись не обновлена")

return

student\_id = self.lineEdit\_7\_att.text()

shedule\_id = self.lineEdit\_4\_att.text()

pair1 = self.lineEdit\_att.text()

pair2 = self.lineEdit\_6\_\_\_att.text()

pair3 = self.lineEdit\_3\_att.text()

pair4 = self.lineEdit\_3\_\_att.text()

mark = self.lineEdit\_3\_\_att\_2.text()

comment = self.lineEdit\_8\_\_att.text()

self.cur.execute('''

UPDATE attendance SET ID = ?, student\_id = ?, shedule\_id = ?, pair1 = ?, pair2 = ?, pair3 = ?, pair4 = ?, mark = ?, comment = ?

WHERE ID = ?''',

(ID, student\_id, shedule\_id, pair1, pair2, pair3, pair4, mark, comment, ID))

self.db.commit()

self.Clear()

self.messagebox("Сообщение", "Запись обновлена")

self.loaddata()

def loaddata(self):

self.table\_\_att.clearContents()

connection = sqlite3.connect('db.sqlite')

cur = connection.cursor()

sql = '''SELECT \* FROM attendance LIMIT 500'''

cur.execute(sql)

data = cur.fetchall()

self.table\_\_att.setRowCount(len(data)) # Устанавливаем количество строк таблицы равным количеству записей из базы данных

for row\_index, row\_data in enumerate(data):

for column\_index, cell\_data in enumerate(row\_data):

item = QtWidgets.QTableWidgetItem(str(cell\_data))

self.table\_\_att.setItem(row\_index, column\_index, item)

self.table\_\_att.resizeColumnsToContents() # Растягиваем столбцы таблицы для лучшего отображения данных

class MainAppStudents(QMainWindow,students\_ui):

def \_\_init\_\_(self, parent=None):

super(MainAppStudents, self).\_\_init\_\_(parent)

QMainWindow.\_\_init\_\_(self)

self.setupUi(self)

self.db\_connect()

self.handles\_btn()

self.table\_st.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_st.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Fixed)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/graduated.png'))

self.move(QApplication.desktop().screen().rect().center()- self.rect().center())

self.table\_st.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch) #авт.растягивание колонок

self.loaddata()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if(self.cur):

self.messagebox("Сообщение проверки", "База данных db подключена")

else:

self.messagebox("Сообщение проверки", "База данных db не подключена")

def messagebox(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.btn\_Insert\_st.clicked.connect(self.Add\_Students)

self.Search\_btn\_st.clicked.connect(self.Search\_Students)

self.deletezapic\_st.clicked.connect(self.Delete\_Students)

self.clears\_st.clicked.connect(self.Clear)

self.update\_st.clicked.connect(self.Update\_Students)

self.vuvod\_st.clicked.connect(self.loaddata)

self.cleant\_st.clicked.connect(self.clear\_table)

self.student\_btn.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def Add\_Students(self):

ID = self.lineEdit\_2\_st.text()

surname = self.lineEdit\_3\_st.text()

name = self.lineEdit\_4\_st.text()

patronymic = self.lineEdit\_st.text()

gender = self.lineEdit\_6\_st\_2.text()

number = self.lineEdit\_6\_st.text()

id\_groupe = self.lineEdit\_5\_st.text()

mail = self.lineEdit\_7\_st.text()

phone = self.lineEdit\_8\_st.text()

if ID and surname and name and patronymic and gender and number and id\_groupe and mail and phone != "":

self.cur.execute('SELECT \* FROM students WHERE ID=?', (ID,))

data = self.cur.fetchone()

if data:

self.messagebox("Ошибка", "Запись с таким ID уже существует")

else:

self.cur.execute('''

INSERT INTO students (ID, surname, name, patronymic, gender, number, id\_groupe, mail, phone)

VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)

''', (ID, surname, name, patronymic, gender, number, id\_groupe, mail, phone))

self.db.commit()

self.messagebox("Сообщение", "Запись группы была успешно добавлена")

self.loaddata()

else:

self.messagebox("Сообщение", "Введите данные для записи в базу данных")

def Search\_Students(self):

ID = self.lineEdit\_seatch\_st.text()

sql =('''SELECT \* FROM students WHERE ID=? ''')

self.cur.execute(sql,[(ID)])

data = self.cur.fetchone()

if ID=="":

self.messagebox("Сообщение", "Введите ID для поиска записи")

elif (data):

self.lineEdit\_2\_st.setText(str(data[0]))

self.lineEdit\_3\_st.setText(str(data[1]))

self.lineEdit\_4\_st.setText(str(data[2]))

self.lineEdit\_st.setText(str(data[3]))

self.lineEdit\_6\_st\_2.setText(str(data[4]))

self.lineEdit\_6\_st.setText(str(data[5]))

self.lineEdit\_5\_st.setText(str(data[6]))

self.lineEdit\_7\_st.setText(str(data[7]))

self.lineEdit\_8\_st.setText(str(data[8]))

else:

self.messagebox("Сообщение", "Нет записи в БД")

def Delete\_Students(self):

ID = self.lineEdit\_2\_st.text()

sql = 'DELETE FROM students WHERE ID=?'

self.cur.execute(sql, [(ID)])

self.db.commit()

self.loaddata()

self.Clear()

def Clear(self):

self.lineEdit\_2\_st.setText("")

self.lineEdit\_3\_st.setText("")

self.lineEdit\_4\_st.setText("")

self.lineEdit\_st.setText("")

self.lineEdit\_6\_st\_2.setText("")

self.lineEdit\_6\_st.setText("")

self.lineEdit\_5\_st.setText("")

self.lineEdit\_7\_st.setText("")

self.lineEdit\_8\_st.setText("")

self.lineEdit\_seatch\_st.setText("")

def clear\_table(self):

self.table\_st.setRowCount(0)

def Update\_Students(self):

ID = self.lineEdit\_2\_st.text()

if(ID):

ID = self.lineEdit\_2\_st.text()

surname = self.lineEdit\_3\_st.text()

name = self.lineEdit\_4\_st.text()

patronymic = self.lineEdit\_st.text()

gender = self.lineEdit\_6\_st\_2.text()

number = self.lineEdit\_6\_st.text()

id\_groupe = self.lineEdit\_5\_st.text()

mail = self.lineEdit\_7\_st.text()

phone = self.lineEdit\_8\_st.text()

self.cur.execute('''

UPDATE students SET ID=?, surname=?, name=?, patronymic=?, gender=?, number=?, id\_groupe=?, mail=?, phone=? WHERE ID=?

''',(ID, surname, name, patronymic, gender, number, id\_groupe, mail, phone,ID))

self.db.commit()

self.Clear()

self.messagebox("Сообщение", "Запись обновлена")

self.loaddata()

else:

self.messagebox("Сообщение", "Запись не обновлена")

def loaddata(self):

self.cur.execute('SELECT \* FROM students LIMIT 300')

rows = self.cur.fetchall()

self.table\_st.setRowCount(len(rows))

for row\_num, row\_data in enumerate(rows):

for col\_num, data in enumerate(row\_data):

self.table\_st.setItem(row\_num, col\_num, QtWidgets.QTableWidgetItem(str(data)))

self.table\_st.resizeColumnsToContents()

class MainAppSubjects(QMainWindow, subjects\_ui):

def \_\_init\_\_(self, parent=None):

super(MainAppSubjects, self).\_\_init\_\_(parent)

QMainWindow.\_\_init\_\_(self)

self.setupUi(self)

self.db\_connect()

self.handles\_btn()

self.table\_sub.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_sub.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Fixed)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/sub.ico'))

self.move(QApplication.desktop().screen().rect().center() - self.rect().center())

self.table\_sub.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch) # авт.растягивание колонок

self.loaddata()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if self.cur:

self.messagebox("Сообщение проверки", "База данных db подключена")

else:

self.messagebox("Сообщение проверки", "База данных db не подключена")

def messagebox(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.Insert\_dis.clicked.connect(self.Add\_Subjects)

self.Search\_btn\_sub.clicked.connect(self.Search\_Subjects)

self.deletezapic\_sub.clicked.connect(self.Delete\_Subjects)

self.clears\_sub.clicked.connect(self.Clear)

self.update\_sub.clicked.connect(self.Update\_Subjects)

self.vuvod\_sub.clicked.connect(self.loaddata)

self.cleant\_sub.clicked.connect(self.clear\_table)

self.subject\_btn.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def Add\_Subjects(self):

ID = self.lineEdit2.text()

indexx = self.lineEdit3.text()

name = self.lineEdit4.text()

if ID and indexx and name != "":

# Проверка на повторение записей

self.cur.execute("SELECT \* FROM subjects WHERE ID=?", (ID,))

existing\_record = self.cur.fetchone()

if existing\_record:

self.messagebox("Сообщение", "Запись с таким ID уже существует")

else:

self.cur.execute('''

INSERT INTO subjects (ID, indexx, name)

VALUES(?, ?, ?)''', (ID, indexx, name))

self.db.commit()

self.messagebox("Сообщение", "Запись успешно добавлена")

self.loaddata() # Update the table

else:

self.messagebox("Сообщение", "Введите данные для записи в базу данных")

def Search\_Subjects(self):

ID = self.lineEdit\_seatch\_sub.text()

sql = '''SELECT \* FROM subjects WHERE ID=? '''

self.cur.execute(sql, [(ID)])

data = self.cur.fetchone()

if ID == "":

self.messagebox("Сообщение", "Введите ID для поиска записи")

elif data:

self.lineEdit2.setText(str(data[0]))

self.lineEdit3.setText(str(data[1]))

self.lineEdit4.setText(str(data[2]))

else:

self.messagebox("Сообщение", "Нет записи в БД")

def Delete\_Subjects(self):

ID = self.lineEdit2.text()

if ID:

sql = '''DELETE FROM subjects WHERE ID=?'''

self.cur.execute(sql, [(ID)])

self.db.commit()

self.Clear()

self.loaddata() # Update the table

else:

self.messagebox("Сообщение", "Введите ID для удаления записи")

def Clear(self):

self.lineEdit2.setText("")

self.lineEdit3.setText("")

self.lineEdit4.setText("")

self.lineEdit\_seatch\_sub.setText("")

def clear\_table(self):

self.table\_sub.setRowCount(0)

def Update\_Subjects(self):

ID = self.lineEdit2.text()

if ID:

self.cur.execute("SELECT \* FROM subjects WHERE ID=?", (ID,))

existing\_record = self.cur.fetchone()

if existing\_record:

indexx = self.lineEdit3.text()

name = self.lineEdit4.text()

self.cur.execute('''

UPDATE subjects SET ID=?, indexx=?, name=? WHERE ID=?

''', (ID, indexx, name, ID))

self.db.commit()

self.Clear()

self.messagebox("Сообщение", "Запись обновлена")

self.loaddata() # Update the table

else:

self.messagebox("Сообщение", "Записи с указанным ID не существует")

else:

self.messagebox("Сообщение", "Запись не обновлена")

def loaddata(self):

self.table\_sub.setRowCount(0)

sql = '''SELECT \* FROM subjects LIMIT 50'''

tablerow = 0

for row in self.cur.execute(sql):

self.table\_sub.insertRow(tablerow)

self.table\_sub.setItem(tablerow, 0, QtWidgets.QTableWidgetItem(str(row[0])))

self.table\_sub.setItem(tablerow, 1, QtWidgets.QTableWidgetItem(row[1]))

self.table\_sub.setItem(tablerow, 2, QtWidgets.QTableWidgetItem(row[2]))

tablerow += 1

self.table\_sub.resizeColumnsToContents()

class MainAppTeachers(QMainWindow, teachers\_ui):

def \_\_init\_\_(self, parent=None):

super(MainAppTeachers, self).\_\_init\_\_(parent)

QMainWindow.\_\_init\_\_(self)

self.setupUi(self)

self.db\_connect()

self.handles\_btn()

self.table\_tea.setEditTriggers(QtWidgets.QAbstractItemView.NoEditTriggers)

self.table\_tea.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Fixed)

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/training.png'))

self.move(QApplication.desktop().screen().rect().center() - self.rect().center())

self.table\_tea.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stretch) # авт.растягивание колонок

self.loaddata()

def db\_connect(self):

self.db = sqlite3.connect('db.sqlite')

self.cur = self.db.cursor()

if self.cur:

self.messagebox("Сообщение проверки", "База данных db подключена")

else:

self.messagebox("Сообщение проверки", "База данных db не подключена")

def messagebox(self, title, message):

mess = QtWidgets.QMessageBox()

mess.setWindowTitle(title)

mess.setText(message)

mess.setStandardButtons(QtWidgets.QMessageBox.Ok)

mess.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/message.png'))

mess.exec\_()

def handles\_btn(self):

self.btn\_Insert\_tea.clicked.connect(self.Add\_Teachers)

self.Search\_btn\_tea.clicked.connect(self.Search\_Teachers)

self.deletezapic\_tea.clicked.connect(self.Delete\_Teachers)

self.clears\_tea.clicked.connect(self.Clear)

self.update\_tea.clicked.connect(self.Update\_Teachers)

self.vuvod\_tea.clicked.connect(self.loaddata)

self.cleant\_tea.clicked.connect(self.clear\_table)

self.teachers\_btn.clicked.connect(self.open\_main\_menu)

def open\_main\_menu(self):

self.close()

self.main\_menu = MainAppMenu()

self.main\_menu.show()

def Add\_Teachers(self):

ID = self.lineEdit\_2\_tea.text()

surname = self.lineEdit\_3\_tea.text()

name = self.lineEdit\_4\_tea.text()

patronymic = self.lineEdit\_tea.text()

mail = self.lineEdit\_8\_tea.text()

phone = self.lineEdit\_6\_tea.text()

subject\_id = self.lineEdit\_5\_tea.text()

if ID and surname and name and patronymic and mail and phone and subject\_id != "":

self.cur.execute("SELECT \* FROM teachers WHERE ID=?", (ID,))

existing\_data = self.cur.fetchone()

if existing\_data:

self.messagebox("Ошибка", "Запись с таким ID уже существует")

return

self.cur.execute(

'''INSERT INTO teachers (ID, surname, name, patronymic, mail, phone, subject\_id)

VALUES(?,?,?,?,?,?,?)''', (ID, surname, name, patronymic, mail, phone, subject\_id))

self.db.commit()

self.messagebox("Сообщение", "Запись успешно добавлена")

self.loaddata() # Update the table

else:

self.messagebox("Сообщение", "Введите данные для записи в базу данных")

def Search\_Teachers(self):

ID = self.lineEdit\_seatch\_tea.text()

sql = '''SELECT \* FROM teachers WHERE ID=? '''

self.cur.execute(sql, [(ID)])

data = self.cur.fetchone()

if ID == "":

self.messagebox("Сообщение", "Введите ID для поиска записи")

elif data:

self.lineEdit\_2\_tea.setText(str(data[0]))

self.lineEdit\_3\_tea.setText(str(data[1]))

self.lineEdit\_4\_tea.setText(str(data[2]))

self.lineEdit\_tea.setText(str(data[3]))

self.lineEdit\_8\_tea.setText(str(data[4]))

self.lineEdit\_6\_tea.setText(str(data[5]))

self.lineEdit\_5\_tea.setText(str(data[6]))

else:

self.messagebox("Сообщение", "Нет записи в БД")

def Delete\_Teachers(self):

ID = self.lineEdit\_2\_tea.text()

data = self.cur.fetchone()

sql = '''DELETE FROM teachers WHERE ID=?'''

self.cur.execute(sql, [(ID)])

self.db.commit()

self.Clear()

self.loaddata()

def Clear(self):

self.lineEdit\_2\_tea.setText("")

self.lineEdit\_3\_tea.setText("")

self.lineEdit\_4\_tea.setText("")

self.lineEdit\_tea.setText("")

self.lineEdit\_8\_tea.setText("")

self.lineEdit\_6\_tea.setText("")

self.lineEdit\_5\_tea.setText("")

self.lineEdit\_seatch\_tea.setText("")

def clear\_table(self):

self.table\_tea.setRowCount(0)

def Update\_Teachers(self):

ID = self.lineEdit\_2\_tea.text()

if ID:

ID = self.lineEdit\_2\_tea.text()

surname = self.lineEdit\_3\_tea.text()

name = self.lineEdit\_4\_tea.text()

patronymic = self.lineEdit\_tea.text()

mail = self.lineEdit\_8\_tea.text()

phone = self.lineEdit\_6\_tea.text()

subject\_id = self.lineEdit\_5\_tea.text()

self.cur.execute('''

UPDATE teachers SET ID=?, surname=?, name=?, patronymic=?, mail=?, phone=?, subject\_id=? WHERE ID=?

''', (ID, surname, name, patronymic, mail, phone, subject\_id, ID))

self.db.commit()

self.Clear()

self.messagebox("Сообщение", "Запись обновлена")

self.loaddata() # Update the table

else:

self.messagebox("Сообщение", "Запись не обновлена")

def loaddata(self):

self.table\_tea.clearContents()

connection = sqlite3.connect('db.sqlite')

cur = connection.cursor()

sql = '''SELECT \* FROM teachers LIMIT 100'''

cur.execute(sql)

data = cur.fetchall()

row\_count = len(data)

self.table\_tea.setRowCount(row\_count)

for row\_num, row\_data in enumerate(data):

for col\_num, col\_data in enumerate(row\_data):

item = QtWidgets.QTableWidgetItem(str(col\_data))

self.table\_tea.setItem(row\_num, col\_num, item)

self.table\_tea.resizeColumnsToContents()

class MainFormTimePair(QMainWindow, timepair\_ui):

def \_\_init\_\_(self, parent=None):

super(MainFormTimePair, self).\_\_init\_\_(parent)

self.setupUi(self)

self.setWindowTitle("TimePair")

self.table.setColumnCount(3)

self.table.setHorizontalHeaderLabels(["ID", "Start Time", "End Time"])

self.setWindowIcon(QtGui.QIcon('C:/Users/natal/OneDrive/Рабочий стол/Code\_rename/photo/clock.jpg'))

self.load\_data()

def load\_data(self):

connection = sqlite3.connect('db.sqlite')

cursor = connection.cursor()

cursor.execute("SELECT \* FROM pairs")

data = cursor.fetchall()

self.table.setRowCount(len(data))

for row\_index, row\_data in enumerate(data):

for column\_index, cell\_data in enumerate(row\_data):

item = QtWidgets.QTableWidgetItem(str(cell\_data))

self.table.setItem(row\_index, column\_index, item)

self.table.resizeColumnsToContents()

cursor.close()

connection.close()

def main():

app = QApplication(sys.argv)

window = MainAppMenu()

window.setFixedSize(890, 545)

window.show()

app.exec\_()

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

main()

**Вывод:** в ходе этой работы автор усовершенствовал свои навыки в работе на языке программирования Python; научился пользовать системой управления базой данных MySQL; узнал, как подключать базу данных к Python и взаимодействовать с ней.