**ПРИЛОЖЕНИЕ А**

**Листинг программы**

**Клиент**

import sys

import traceback

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QDialog, QApplication, QMessageBox, QMainWindow, QMenu, QComboBox

from PyQt5.uic import loadUi

from datetime import datetime

import sys

import traceback

import sqlite3

import requests

app = QApplication(sys.argv)

widget = QtWidgets.QStackedWidget()

login = 'Danya'

password = '1111'

w = 0

h = 0

class Login(QDialog): #Класс окна авторизации

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

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

loadUi("Login.ui",self)

self.pushButton.clicked.connect(self.glmenu)

self.pushButton\_2.clicked.connect(self.glmenu2)

def glmenu(self):

try:

global w, h

lg = self.lineEdit\_2.text()

psw = self.lineEdit.text()

if lg == login and psw == password:

w = 600

h = 506

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

else:

QMessageBox.about(self, "Ошибка", "Неккоректная авторизация")

except:

QMessageBox.about(self, "Ошибка", "Сервер отключен")

mainwindow = Login()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

widget.setFixedWidth(113)

widget.setFixedHeight(83)

def glmenu2(self): #Функция главного меню

try:

global w, h

w = 600

h = 344

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

except:

QMessageBox.about(self, "Ошибка", "Сервер отключен")

mainwindow = Login()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

widget.setFixedWidth(113)

widget.setFixedHeight(83)

class Admin(QDialog): #Класс главного окна

def zapros(self): #функция запроса

zapr = self.lineEdit\_2.text() #переменная введённого условия в запрос

cmbx = self.comboBox.currentText() #переменная выбранного пункта в комбо бокс 1

cmbx2 = self.comboBox\_2.currentText( ) #переменная выбранного пункта

strk = str(cmbx)+"-"+str(cmbx2)+"-"+str(zapr)

cmbx3 = self.comboBox\_3.currentText()

try:

if cmbx3 == 'Таблица: Клиенты':

com = requests.post(

'http://127.0.0.1:5000/zaprs1',

json={'text': strk}

)

itog = com.json()['result']

if itog == 'error':

QMessageBox.about(self, "Ошибка", "Ошибка ввода данных!")

else:

self.tableWidget.clear()

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Клиента","ФИО","Возраст"])

elif cmbx3 == 'Таблица: Сотрудники':

com = requests.post(

'http://127.0.0.1:5000/zaprs2',

json={'text': strk}

)

itog = com.json()['result']

if itog == 'error':

QMessageBox.about(self, "Ошибка", "Ошибка ввода данных!")

else:

self.tableWidget.clear()

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(5)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 4:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Сотрудника","ФИО","Возраст","Должность","Зарплата"])

elif cmbx3 == 'Таблица: Смены':

com = requests.post(

'http://127.0.0.1:5000/zaprs3',

json={'text': strk}

)

itog = com.json()['result']

if itog == 'error':

QMessageBox.about(self, "Ошибка", "Ошибка ввода данных!")

else:

self.tableWidget.clear()

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(2)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 1:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Смены","ID Сотрудника"])

elif cmbx3 == 'Таблица: Номера':

com = requests.post(

'http://127.0.0.1:5000/zaprs4',

json={'text': strk}

)

itog = com.json()['result']

if itog == 'error':

QMessageBox.about(self, "Ошибка", "Ошибка ввода данных!")

else:

self.tableWidget.clear()

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Номера","Класс","Стоимость"])

elif cmbx3 == 'Таблица: Журнал':

com = requests.post(

'http://127.0.0.1:5000/zaprs5',

json={'text': strk}

)

itog = com.json()['result']

if itog == 'error':

QMessageBox.about(self, "Ошибка", "Ошибка ввода данных!")

else:

self.tableWidget.clear()

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(6)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 5:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Записи","ID Клиента","ID Номера","ID Смены","Дата заезда","Дата выезда"])

except pymysql.err.ProgrammingError:

QtWidgets.QMessageBox.warning(self, "Ошибка",

"Пусто!")

except pymysql.err.OperationalError:

QtWidgets.QMessageBox.warning(self, "Ошибка",

"Некорректный ввод данных!")

except pymysql.err.IntegrityError:

QtWidgets.QMessageBox.warning(self, "Ошибка",

"Сотрудник привязан к другой таблице!")

def loadbd(self):

try:

self.comboBox.clear()

cmbx3 = self.comboBox\_3.currentText()

if cmbx3 == 'Таблица: Клиенты':

com = requests.post(

'http://127.0.0.1:5000/myfunction',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Клиента","ФИО","Возраст"])

self.comboBox.addItems(["ID Клиента","ФИО","Возраст"])

elif cmbx3 == 'Таблица: Сотрудники':

com = requests.post(

'http://127.0.0.1:5000/myfunction2',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(5)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 4:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Сотрудника","ФИО","Возраст","Должность","Зарплата"])

self.comboBox.addItems(["ID Сотрудника","ФИО","Возраст","Должность","Зарплата"])

elif cmbx3 == 'Таблица: Смены':

com = requests.post(

'http://127.0.0.1:5000/myfunction3',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(2)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 1:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Смены","ID Сотрудника"])

self.comboBox.addItems(["ID Смены","ID Сотрудника"])

elif cmbx3 == 'Таблица: Номера':

com = requests.post(

'http://127.0.0.1:5000/myfunction4',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Номера","Класс","Стоимость"])

self.comboBox.addItems(["ID Номера","Класс","Стоимость"])

elif cmbx3 == 'Таблица: Журнал':

com = requests.post(

'http://127.0.0.1:5000/myfunction5',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(6)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 5:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Записи","ID Клиента","ID Номера","ID Смены","Дата заезда","Дата выезда"])

self.comboBox.addItems(["ID Записи","ID Клиента","ID Номера","ID Смены","Дата заезда","Дата выезда"])

except:

QMessageBox.about(self, "Ошибка", "Сервер отключен")

mainwindow = Login()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

widget.setFixedWidth(113)

widget.setFixedHeight(83)

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

global w, h

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

loadUi("glmenu.ui",self)

widget.setFixedWidth(w)

widget.setFixedHeight(h)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_5.clicked.connect(self.sotr)

self.pushButton\_2.clicked.connect(self.otr)

self.pushButton\_3.clicked.connect(self.sostotr)

self.pushButton\_4.clicked.connect(self.tran)

self.pushButton.clicked.connect(self.loadbd)

self.pushButton\_6.clicked.connect(self.zapros)

self.pushButton\_7.clicked.connect(self.zhur)

def sotr(self):

sotrr = Sotrudniki()

widget.addWidget(sotrr)

widget.setCurrentIndex(widget.currentIndex()+1)

def otr(self):

otrr = Otryad()

widget.addWidget(otrr)

widget.setCurrentIndex(widget.currentIndex()+1)

def tran(self):

tranr = Transport()

widget.addWidget(tranr)

widget.setCurrentIndex(widget.currentIndex()+1)

def sostotr(self):

sostotrr = Sostotr()

widget.addWidget(sostotrr)

widget.setCurrentIndex(widget.currentIndex()+1)

def zhur(self):

zhur = Zhur()

widget.addWidget(zhur)

widget.setCurrentIndex(widget.currentIndex()+1)

class Zhur(QDialog):

def loadbd(self):

try:

com = requests.post(

'http://127.0.0.1:5000/myfunction5',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(6)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 5:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Записи","ID Клиента","ID Номера","ID Смены","Дата заезда","Дата выезда"])

except:

QMessageBox.about(self, "Ошибка", "Сервер отключен")

mainwindow = Login()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

widget.setFixedWidth(113)

widget.setFixedHeight(83)

def dobavzp(self):

ids = self.lineEdit.text()

fio = self.lineEdit\_2.text()

gen = self.lineEdit\_3.text()

old = self.lineEdit\_4.text()

zv = self.lineEdit\_5.text()

idot = self.lineEdit\_6.text()

strk = str(ids)+"-"+str(fio)+"-"+str(gen)+"-"+str(old)+"-"+str(zv)+"-"+str(idot)

com = requests.post(

'http://127.0.0.1:5000/dobavv4',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def udalzp(self):

ids = self.lineEdit\_7.text()

strk = str(ids)

com = requests.post(

'http://127.0.0.1:5000/ubavv4',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def izmzp(self):

ids = self.lineEdit\_8.text()

cmbx = self.comboBox.currentText()

upd = self.lineEdit\_9.text()

strk = str(ids)+"-"+str(cmbx)+"-"+str(upd)

com = requests.post(

'http://127.0.0.1:5000/izmavv4',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

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

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

loadUi("sotrudniki.ui",self)

widget.setFixedWidth(601)

widget.setFixedHeight(432)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_4.clicked.connect(self.glmenu)

self.pushButton.clicked.connect(self.dobavzp)

self.pushButton\_2.clicked.connect(self.udalzp)

self.pushButton\_3.clicked.connect(self.izmzp)

def glmenu(self):

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

class Sotrudniki(QDialog):

def loadbd(self):

com = requests.post(

'http://127.0.0.1:5000/myfunction2',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(5)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 4:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Сотрудника","ФИО","Возраст","Должность","Зарплата"])

def dobavzp(self):

ids = self.lineEdit.text()

fio = self.lineEdit\_2.text()

gen = self.lineEdit\_3.text()

old = self.lineEdit\_4.text()

zv = self.lineEdit\_5.text()

strk = str(ids)+"-"+str(fio)+"-"+str(gen)+"-"+str(old)+"-"+str(zv)

com = requests.post(

'http://127.0.0.1:5000/dobavv5',

json={'text': strk}

)

self.tableWidget.clear()

self.loadbd()

def udalzp(self):

ids = self.lineEdit\_7.text()

strk = str(ids)

com = requests.post(

'http://127.0.0.1:5000/ubavv5',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def izmzp(self):

ids = self.lineEdit\_8.text()

cmbx = self.comboBox.currentText()

upd = self.lineEdit\_9.text()

strk = str(ids)+"-"+str(cmbx)+"-"+str(upd)

com = requests.post(

'http://127.0.0.1:5000/izmavv5',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

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

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

loadUi("sotrudniki2.ui",self)

widget.setFixedWidth(501)

widget.setFixedHeight(432)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_4.clicked.connect(self.glmenu)

self.pushButton.clicked.connect(self.dobavzp)

self.pushButton\_2.clicked.connect(self.udalzp)

self.pushButton\_3.clicked.connect(self.izmzp)

def glmenu(self):

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

class Transport(QDialog):

def loadbd(self):

com = requests.post(

'http://127.0.0.1:5000/myfunction3',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(2)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 1:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Смены","ID Сотрудника"])

def dobavzp(self):

idsm = self.lineEdit.text()

idsot = self.lineEdit\_2.text()

strk = str(idsm)+"-"+str(idsot)

com = requests.post(

'http://127.0.0.1:5000/dobavv3',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def udalzp(self):

idsm = self.lineEdit\_7.text()

idsot = self.lineEdit\_3.text()

strk = str(idsm)+"-"+str(idsot)

com = requests.post(

'http://127.0.0.1:5000/ubavv3',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

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

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

loadUi("Transport.ui",self)

widget.setFixedWidth(201)

widget.setFixedHeight(403)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_4.clicked.connect(self.glmenu)

self.pushButton.clicked.connect(self.dobavzp)

self.pushButton\_2.clicked.connect(self.udalzp)

def glmenu(self):

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

class Sostotr(QDialog):

def loadbd(self):

com = requests.post(

'http://127.0.0.1:5000/myfunction4',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Номера","Класс","Стоимость"])

def dobavzp(self):

idnom = self.lineEdit.text()

clas = self.lineEdit\_2.text()

price = self.lineEdit\_3.text()

strk = str(idnom)+"-"+str(clas)+"-"+str(price)

com = requests.post(

'http://127.0.0.1:5000/dobavv2',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def udalzp(self):

idnom = self.lineEdit\_7.text()

strk = str(idnom)

com = requests.post(

'http://127.0.0.1:5000/ubavv2',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def izmzp(self):

idnom = self.lineEdit\_8.text()

cmbx = self.comboBox.currentText()

upd = self.lineEdit\_9.text()

strk = str(idnom)+"-"+str(cmbx)+"-"+str(upd)

com = requests.post(

'http://127.0.0.1:5000/izmavv2',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

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

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

loadUi("Sostotr.ui",self)

widget.setFixedWidth(301)

widget.setFixedHeight(432)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_4.clicked.connect(self.glmenu)

self.pushButton.clicked.connect(self.dobavzp)

self.pushButton\_2.clicked.connect(self.udalzp)

self.pushButton\_3.clicked.connect(self.izmzp)

def glmenu(self):

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

class Otryad(QDialog):

def loadbd(self):

com = requests.post(

'http://127.0.0.1:5000/myfunction',

json={'text': 'text'}

)

itog = com.json()['result']

row = 0

column = 0

self.tableWidget.setRowCount(len(itog))

self.tableWidget.setColumnCount(3)

for i in itog:

for k in i:

self.tableWidget.setItem(row, column, QtWidgets.QTableWidgetItem(str(k)))

if column == 2:

row += 1

column = 0

else:

column += 1

self.tableWidget.setHorizontalHeaderLabels(["ID Клиента","ФИО","Возраст"])

def dobavzp(self):

idcl = self.lineEdit.text()

fio = self.lineEdit\_2.text()

old = self.lineEdit\_3.text()

strk = str(idcl)+"-"+str(fio)+"-"+str(old)

com = requests.post(

'http://127.0.0.1:5000/dobavv1',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def udalzp(self):

idcl = self.lineEdit\_7.text()

strk = str(idcl)

com = requests.post(

'http://127.0.0.1:5000/ubavv1',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

def izmzp(self):

idcl = self.lineEdit\_8.text()

cmbx = self.comboBox.currentText()

upd = self.lineEdit\_9.text()

strk = str(idcl)+"-"+str(cmbx)+"-"+str(upd)

com = requests.post(

'http://127.0.0.1:5000/izmavv1',

json={'text': strk}

)

itog = com.json()['result']

self.tableWidget.clear()

self.loadbd()

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

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

loadUi("Otryad.ui",self)

widget.setFixedWidth(301)

widget.setFixedHeight(432)

self.loadbd()

self.tableWidget.verticalHeader().hide()

self.pushButton\_4.clicked.connect(self.glmenu)

self.pushButton.clicked.connect(self.dobavzp)

self.pushButton\_2.clicked.connect(self.udalzp)

self.pushButton\_3.clicked.connect(self.izmzp)

def glmenu(self):

mainwindow = Admin()

widget.addWidget(mainwindow)

widget.setCurrentIndex(widget.currentIndex()+1)

mainwindow = Login()

widget.addWidget(mainwindow)

widget.setFixedWidth(113)

widget.setFixedHeight(83)

widget.show()

widget.setWindowTitle('Шеметов РГ')

app.exec\_()

**Сервер**

import sqlite3

from flask import Flask, request

app = Flask(\_\_name\_\_)

@app.route("/myfunction", methods=['POST']) #Создаем функцию

def myfunction(): #Описываем

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

cur.execute('SELECT \* FROM Clients')

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result} # Передаем значение клиенту

@app.route("/myfunction2", methods=['POST'])

def myfunction2():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

cur.execute('SELECT \* FROM Sotrudniki')

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/myfunction3", methods=['POST'])

def myfunction3():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

cur.execute('SELECT \* FROM Smena')

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/myfunction4", methods=['POST'])

def myfunction4():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

cur.execute('SELECT \* FROM Nomera')

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/myfunction5", methods=['POST'])

def myfunction5():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

cur.execute('SELECT \* FROM Zhurnal')

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/zaprs1", methods=['POST'])

def zaprs1():

con = sqlite3.connect('GOST.db')

rest = request.json['text']

rest = rest.split('-')

zapr = rest[2]

cmbx = rest[0]

cmbx2 = rest[1]

try:

if cmbx == 'ID Клиента':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Clients Where IdCl {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ФИО':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Clients Where FIO LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Возраст':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Clients Where Old {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

con.close()

return {'result': itog}

except:

itog = 'error'

con.close()

return {'result': itog}

@app.route("/zaprs2", methods=['POST'])

def zaprs2():

con = sqlite3.connect('GOST.db')

rest = request.json['text']

rest = rest.split('-')

zapr = rest[2]

cmbx = rest[0]

cmbx2 = rest[1]

try:

if cmbx == 'ID Сотрудника':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Sotrudniki Where IdSotr {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ФИО':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Sotrudniki Where FIO LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Возраст':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Sotrudniki Where Old {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Должность':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Sotrudniki Where Dolzhn LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Зарплата':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Sotrudniki Where Zarplata {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

con.close()

return {'result': itog}

except:

itog = 'error'

con.close()

return {'result': itog}

@app.route("/zaprs3", methods=['POST'])

def zaprs3():

con = sqlite3.connect('GOST.db')

rest = request.json['text']

rest = rest.split('-')

zapr = rest[2]

cmbx = rest[0]

cmbx2 = rest[1]

try:

if cmbx == 'ID Смены':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Smena Where IdSmen {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ID Сотрудника':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Smena Where IdSot {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

con.close()

return {'result': itog}

except:

itog = 'error'

con.close()

return {'result': itog}

@app.route("/zaprs4", methods=['POST'])

def zaprs4():

con = sqlite3.connect('GOST.db')

rest = request.json['text']

rest = rest.split('-')

zapr = rest[2]

cmbx = rest[0]

cmbx2 = rest[1]

try:

if cmbx == 'ID Номера':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Nomera Where IdNom {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Стоимость':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Nomera Where Price {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Класс':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Nomera Where Class LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

con.close()

return {'result': itog}

except:

itog = 'error'

con.close()

return {'result': itog}

@app.route("/zaprs5", methods=['POST'])

def zaprs5():

con = sqlite3.connect('GOST.db')

rest = request.json['text']

rest = rest.split('-')

zapr = rest[2]

cmbx = rest[0]

cmbx2 = rest[1]

try:

if cmbx == 'ID Записи':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where IdZap {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ID Клиента':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where IdCl {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ID Номера':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where IdNom {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'ID Смены':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where IdSmen {cmbx2} {zapr}"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Дата заезда':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where DateZaezd LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

elif cmbx == 'Дата выезда':

cur = con.cursor()

sqlquery = f"SELECT \* FROM Zhurnal Where DateViezd LIKE '{zapr}%'"

result = cur.execute(sqlquery)

con.commit()

itog = cur.fetchall()

con.close()

return {'result': itog}

except:

itog = 'error'

con.close()

return {'result': itog}

@app.route("/dobavv1", methods=['POST'])

def dobavv1():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idcl = rest[0]

fio = rest[1]

old = rest[2]

cur.execute(f"INSERT INTO Clients VALUES({idcl},'{fio}', {old})")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/ubavv1", methods=['POST'])

def ubavv1():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

idcl = rest

cur.execute(f"DELETE FROM Clients WHERE IdCl = {idcl}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/izmavv1", methods=['POST'])

def izmavv1():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idcl = rest[0]

cmbx = rest[1]

upd = rest[2]

if cmbx == 'ФИО':

cur.execute(f"UPDATE Clients SET FIO = '{upd}' WHERE IdCl = {idcl}")

elif cmbx == 'Возраст':

cur.execute(f"UPDATE Clients SET Old = {upd} WHERE IdCl = {idcl}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/dobavv2", methods=['POST'])

def dobavv2():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

clas = rest[1]

price = rest[2]

cur.execute(f"INSERT INTO Nomera VALUES({idnom},'{clas}', {price})")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/ubavv2", methods=['POST'])

def ubavv2():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

idnom = rest

cur.execute(f"DELETE FROM Nomera WHERE IdNom = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/izmavv2", methods=['POST'])

def izmavv2():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

cmbx = rest[1]

upd = rest[2]

if cmbx == 'Класс':

cur.execute(f"UPDATE Nomera SET Class = '{upd}' WHERE IdNom = {idnom}")

elif cmbx == 'Стоимость':

cur.execute(f"UPDATE Nomera SET Price = {upd} WHERE IdNom = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/dobavv3", methods=['POST'])

def dobavv3():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

clas = rest[1]

cur.execute(f"INSERT INTO Smena VALUES({idnom},{clas})")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/ubavv3", methods=['POST'])

def ubavv3():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

clas = rest[1]

cur.execute(f"DELETE FROM Smena WHERE IdSmen = {idnom} and IdSot = {clas}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/dobavv4", methods=['POST'])

def dobavv4():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

ids = rest[0]

fio = rest[1]

gen = rest[2]

old = rest[3]

zv = rest[4]

idot = rest[5]

cur.execute(f"INSERT INTO Zhurnal VALUES({ids},{fio},{gen},{old},'{zv}','{idot}')")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/ubavv4", methods=['POST'])

def ubavv4():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

idnom = rest

cur.execute(f"DELETE FROM Zhurnal WHERE IdZap = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/izmavv4", methods=['POST'])

def izmavv4():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

cmbx = rest[1]

upd = rest[2]

if cmbx == 'ID Клиента':

cur.execute(f"UPDATE Zhurnal SET IdCl = {upd} WHERE IdZap = {idnom}")

elif cmbx == 'ID Номера':

cur.execute(f"UPDATE Zhurnal SET IdNom = {upd} WHERE IdZap = {idnom}")

elif cmbx == 'ID Смены':

cur.execute(f"UPDATE Zhurnal SET IdSmen = {upd} WHERE IdZap = {idnom}")

elif cmbx == 'Дата заезда':

cur.execute(f"UPDATE Zhurnal SET DateZaezd = '{upd}' WHERE IdZap = {idnom}")

elif cmbx == 'Дата выезда':

cur.execute(f"UPDATE Zhurnal SET DateViezd = '{upd}' WHERE IdZap = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/dobavv5", methods=['POST'])

def dobavv5():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

ids = rest[0]

fio = rest[1]

gen = rest[2]

old = rest[3]

zv = rest[4]

cur.execute(f"INSERT INTO Sotrudniki VALUES({ids},'{fio}',{gen},'{old}',{zv})")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/ubavv5", methods=['POST'])

def ubavv5():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

idnom = rest

cur.execute(f"DELETE FROM Sotrudniki WHERE IdSotr = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

@app.route("/izmavv5", methods=['POST'])

def izmavv5():

conn = sqlite3.connect('GOST.db')

cur = conn.cursor()

rest = request.json['text']

rest = rest.split('-')

idnom = rest[0]

cmbx = rest[1]

upd = rest[2]

if cmbx == 'ФИО':

cur.execute(f"UPDATE Sotrudniki SET FIO = '{upd}' WHERE IdSotr = {idnom}")

elif cmbx == 'Возраст':

cur.execute(f"UPDATE Sotrudniki SET Old = {upd} WHERE IdSotr = {idnom}")

elif cmbx == 'Должность':

cur.execute(f"UPDATE Sotrudniki SET Dolzhn = '{upd}' WHERE IdSotr = {idnom}")

elif cmbx == 'Зарплата':

cur.execute(f"UPDATE Sotrudniki SET Zarplata = {upd} WHERE IdSotr = {idnom}")

result = cur.fetchall()

conn.commit()

conn.close()

return {'result': result}

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

app.run()