**Отчет: Создание базы данных**  **и интерфейса “Glazki save”**

Государственное бюджетное профессиональное учреждение



Московской области Люберецкий техникум имени героя Советского Союза, летчика космонавта Юрия Алексеевича Гагарина

**Отчет: Создание базы данных**  **и интерфейса “Glazki save”**

Авторы:

Студенты специальности 09.02.07

“Информационные системы и программирование”,

2-ого курса, группы ИС-21

Уракбоев Руслан,

Островский Сергей, Калакуцкий Денис Рецензент:

Тарджиманян Лия Николаевна

Оценка:\_\_\_\_\_

Дзержинский 2023 г

**Содержание:**

1. Введение……………………………………………………………………. .3
2. Логическая структура……………………………………………………… .4
3. Физическая структура……………………………………………………… 5
4. Инструкция пользования……………………………………………………7
5. Разработка интерфейса………………………………………………………9
6. Заключение………………………………………………………………….22
7. Список литературы………………………………………………………………….22

**Введение**

Цели и задачи

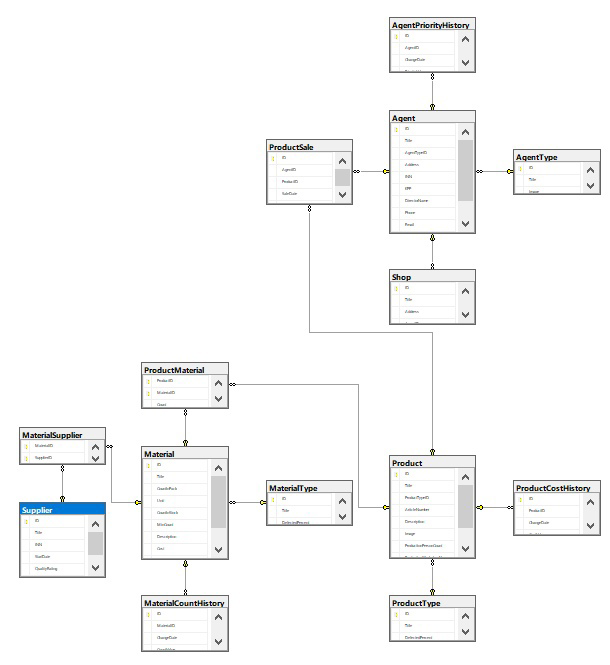
Цель отчета проектирования: закрепление теоретических знаний, а также навыков проектирования БД, полученных при изучении дисциплины «Базы данных».

Задачи:

* Разработать ER-диаграмму по предметной области
* Создать БД по ER-диаграмме
* Наполнить БД данными
* Создать формы для работы с БД
* Связать формы с БД с помощью программирования Инструменты:
* SqlDraw(разработка ER-диаграммы)
* Sqlite (создание БД)
* Qt5 дизайнер (создание форм)
* PyCharm (связка форм и БД) Выбор СУБД.
* Sqlite

**Логическая структура**

Проектирование логической структуры базы данных.



**Физическая структура**

Проектирование физической структуры базы данных.

На базе er-диаграммы создаём базу данных в Sqlite. У нас есть восемь сущностей: Agent, Composition, Employee, Product, Production, Type\_product, prodano, type\_agent

Таблица “Agent” имеет следующие атрибуты: Id\_agent id\_type\_agent Naimenovanie Mail\_agent phone\_agent Adress priority director inn kpp

Таблица “Composition” имеет следующие атрибуты:

id\_composition vendor\_code receipt\_date implementatin\_date id\_agent

Таблица “Employee” имеет следующие атрибуты:

id\_emploee surname name middle\_name

job\_title passport\_series passpoet\_number phone mail

Таблица “Product” имеет следующие атрибуты:

id

naimenovanie id\_type\_product vendor\_code

Number\_people\_product\_ion

Production\_workschop\_number Min\_price\_agent

Таблица “Production” имеет следующие атрибуты:

id\_production

Vendor\_code ID\_emploee

Таблица “Type\_product” имеет следующие атрибуты:

Id\_type\_product Type\_product

Таблица “prodano” имеет следующие атрибуты:

id

vendor\_code id\_agent date\_implementation number\_products

Таблица “type\_agent” имеет следующие атрибуты:

id\_type\_agent type\_agent

База данных “database.db”, была создана специально для регистрации и логирования пользователя, таблица “users” имеет следующие атрибуты:

login password

**Работа с данными**

Организация ввода данных в БД.

Первичный ввод записей в базу данных будет производится при помощи импортирования данных из Excel. Далее ввод данных будет производится непосредственно с формы.

**Инструкция пользования**

Описание информационных потребностей пользователей и выбор способов их реализации.

У пользователя могут возникнуть следующие потребности пи пользовании интерфейсом:

* Регистрация на форме
* Авторизация на форме
* Добавление данных в базу данных
* Удаление данных из базы данных
* Поиск информации и ее вывод

Авторизация:

Если Вы уже зарегистрированы для работы с формами(интерфейсом), то Вам необходимо просто внести данные в соответствующие поля и Вас перенесет на страницу “Employee”

Но если Вы не зарегистрированы, то перед началом работы с формами(интерфейсом) нужно зарегистрироваться. Это можно сделать нажав на кнопку “регистрация” и ввести свой адрес электронной почты и пароль в соответствующие ячейки. После всех выполненных действий можете авторизоваться.

Работа с формами бд

Добавление новых записей

По завершению авторизации. если все предыдущие шаги были выполнены верно, Вы попадаете на новую форму “Employee”. Для того, чтобы начать работать с формой необходимо нажать на кнопку “Открыть”, после чего внизу должна открыться таблица базы данных. Через форму можно добавлять новые записи в базу данных. Для этого необходимо заполнить соответствующие пустые ячейки данными, которые необходимо внести в базу данных. После чего нажимаем на кнопку “Добавить”

Удаление записей

Если данные были введены неверно, то их можно удалить из базы данных. Для того, чтобы удалить данные из базы данных необходимо в нижнем окне, где отображается таблица, навести на номер записи, которую Вы хотите удалить и нажать на кнопку “Удалить”.

Поиск записей

Чтобы найти нужную запись, достаточно ввести любую информацию об этой записи и нажмите кнопку “Найти”. В окошко, через которое отображается таблица, будут выделены необходимые данные.

Переключение между формами

Чтобы переключаться между формами предусмотрены кнопки, которые располагаются под блоком “Поиск”. Нажимая на одну из кнопок вы будете переходить на другую форму. Чтобы вернуться, достаточно нажать на крестик.

**Разработка интерфейса.**

Для начала в QT 5 дизайнере необходимо создать внешний вид форм. После создания внешнего вида формы нам необходимо добавить функции на кнопки и поля ввода, чтобы наш интерес исправно функционировал. Назначение команд для функционирования форм

Импорт библиотек import sqlite3 import sys

from PyQt5.QtWidgets import QApplication, QWidget, QTableWidgetItem from PyQt5 import QtWidgets

Импорт форм

import login import vxod

from glazkisave import Ui\_glazkisave from agent import Ui\_Form from composition import Ui\_composition from prodano import Ui\_prodano from product import Ui\_product

db = sqlite3.connect('database.db') cursor = db.cursor()

Создание базы, где будут хранится логины и пароли зарегистрированных пользователей

cursor.execute('''CREATE TABLE IF NOT EXISTS users(login TEXT, password TEXT)''') db.commit()

Регистрирование пользователей. Связка кнопок и кода

class Registration(QtWidgets.QMainWindow, login.Ui\_Form): def \_\_init\_\_(self): super(Registration, self).\_\_init\_\_() self.setupUi(self) self.lineEdit.setPlaceholderText('Введите логин') self.lineEdit\_2.setPlaceholderText('Введите пароль') self.pushButton.pressed.connect(self.reg) # регитрация self.pushButton\_2.pressed.connect(self.login) #переход на вход

def login(self): #показ класса логин (вход) self.login = Login()

self.login.show() self.hide()

def reg(self): #регитрация user\_login = self.lineEdit.text() user\_password = self.lineEdit\_2.text()

if len(user\_login) == 0:

return if len(user\_password) == 0: return

cursor.execute(f'SELECT login FROM users WHERE login = "{user\_login}" ') if cursor.fetchone() is None:

cursor.execute( f'INSERT INTO users

VALUES("{user\_login}","{user\_password}")')

self.label\_2.setText(f'Аккаунт {user\_login} успешно зарегистрирован') db.commit() else:

self.label\_2.setText('Такая запись уже имеется') Авторизация пользователей. Связка кнопок и кода

class Login(QtWidgets.QMainWindow, vxod.Ui\_Form\_1):# вход def \_\_init\_\_(self): super(Login,self).\_\_init\_\_()

self.setupUi(self) self.lineEdit.setPlaceholderText('Введите логин') self.lineEdit\_2.setPlaceholderText('Введите пароль')

self.pushButton\_2.pressed.connect(self.login) self.pushButton.pressed.connect(self.reg)

def reg(self): self.reg=Registration() self.reg.show() self.hide()

def login(self): try:

user\_login = self.lineEdit.text() user\_password = self.lineEdit\_2.text() if len(user\_login) == 0:

return if len(user\_password) == 0: return

cursor.execute(f'SELECT password FROM users WHERE login =

"{user\_login}"') check\_pass = cursor.fetchall()

cursor.execute(f'SELECT login FROM users WHERE login = "{user\_login}"') check\_login = cursor.fetchall() print (check\_login) print(check\_pass)

if check\_pass[0][0] == user\_password and check\_login[0][0] == user\_login:

self.label\_2.setText(f'Успешная авториазация') self.Emploe = Emploee1() self.Emploe.show()

self.hide() else:

self.label\_2.setText(f'Ошибка авторизации') except Exception as e: self.label\_2.setText(f'Ошибка авторизации')

Форма “Emploee”. Связка

STAFF\_POSTS = ['Зам. директора', 'Директор', 'Менеджер', 'Аналитик',

'Консультант', 'Бухгалтер', 'Экономист' ]

#STAFF\_POSTS - (константа) для выбора должности class Emploee1 (QWidget, Ui\_glazkisave): def \_\_init\_\_(self): super(Emploee1, self).\_\_init\_\_()

self.setupUi(self)

self.comboBox.addItems(STAFF\_POSTS) # выбора должности сотрудники self.pushButton.clicked.connect(self.open) #открытие таблицы сотрудники self.pushButton\_2.clicked.connect(self.insert) #добавить данные сотрудники self.pushButton\_3.clicked.connect(self.deleteRow) #удаление self.pushButton\_8.clicked.connect(self.show\_2)#таблица агент self.pushButton\_10.clicked.connect(self.serch)#поиск сотрудники

self.pushButton\_7.clicked.connect(self.show\_Composition)#таблица композиция self.pushButton\_5.clicked.connect(self.show\_Prodano)#таблица продано self.pushButton\_9.clicked.connect(self.show\_Product)#таблица продукт def open(self): #кнопка добавить

try:

self.conn = sqlite3.connect('Glazki.db') cur = self.conn.cursor() data = cur.execute("select \* from Emploee;") col\_name = [i[0] for i in data.description] data\_rows = data.fetchall() except Exception as e:

print ("Ощибки с подключением к БД") return e

self.tableWidget.setColumnCount(len(col\_name)) self.tableWidget.setHorizontalHeaderLabels(col\_name) self.tableWidget.setRowCount(0) for i, row in enumerate(data\_rows):

self.tableWidget.setRowCount(self.tableWidget.rowCount()+1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def update(self, query="select \* from Emploee"): #обновление данных try:

cur = self.conn.cursor() data = cur.execute(query).fetchall() except Exception as d: print(f"Проблемы с подкл {d}") return d

self.tableWidget.setRowCount(0) #обнуляем все данные из таблицы

#заносим по новой for i, row in enumerate(data):

self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def insert(self): #кнопка добавить row = [self.lineEdit.text(), self.lineEdit\_2.text(), self.lineEdit\_3.text(), self.comboBox.itemText(self.comboBox.currentIndex()), self.lineEdit\_4.text(), self.lineEdit\_5.text(), self.lineEdit\_7.text(), self.lineEdit\_6.text()] try:

cur = self.conn.cursor()

c u r . e x e c u t e ( f " " " i n s e r t i n t o

Emploee(surname,name,middle\_name,job\_title,passport\_series,passpoet\_number,pho ne,mail)

values('{row[0]}','{row[1]}','{row[2]}','{row[3]}','{row[4]}','{row[5]}','{row[6 ]}','{row[7]}')""" )

self.conn.commit() cur.close() except Exception as r: print("Не смогли добавить запись") return r

self.update()#обращаемся к update чтобы сразу увидеть изменения в БД

def deleteRow(self): #удаление записей по id\_emploee id = self.lineEdit\_8.text() conn = sqlite3.connect('Glazki.db') c = conn.cursor()

c.execute("DELETE FROM Emploee WHERE id\_emploee=?", (id,)) conn.commit() conn.close() self.update()

def serch(self): #поиск по таблице

search\_text = self.lineEdit\_9.text()

# Проходим по всем строкам таблицы for row in range(self.tableWidget.rowCount()): # Проходим по всем ячейкам строки for column in range(self.tableWidget.columnCount()):

# Получаем текст ячейки cell\_text = self.tableWidget.item(row, column).text() if search\_text.lower() in cell\_text.lower():

self.tableWidget.selectRow(row) return

Показ формы “Agent” def show\_2(self): self.Agen = Agent1() self.Agen.show()

Показ формы “composition” def show\_Composition(self): self.Comp = Composition1() self.Comp.show()

Показ формы “Prodano” def show\_Prodano(self): self.prod = Prodano1() self.prod.show()

Показ формы “Product” def show\_Product(self): self.produc=Product1() self.produc.show() TYPE = ['1', '2', '3', '4', '5', '6']

Форма “Agent” class Agent1 (QWidget, Ui\_Form): # def \_\_init\_\_(self): super(Agent1, self).\_\_init\_\_()

self.setupUi(self)

self.pushButton.clicked.connect(self.open\_1)#открыть таблицу агент

self.comboBox.addItems(TYPE) #выбор self.pushButton\_2.clicked.connect(self.insert\_1) #добавить self.pushButton\_3.clicked.connect(self.deleteRow\_1) self.pushButton\_10.clicked.connect(self.serch\_1)

def open\_1(self): #открыть таблицу агент try:

self.conn = sqlite3.connect('Glazki.db') cur = self.conn.cursor() data = cur.execute("select \* from Agent;") col\_name = [i[0] for i in data.description] data\_rows = data.fetchall() except Exception as e:

print ("Ощибки с подключением к БД") return e

self.tableWidget.setColumnCount(len(col\_name)) self.tableWidget.setHorizontalHeaderLabels(col\_name) self.tableWidget.setRowCount(0) for i, row in enumerate(data\_rows):

self.tableWidget.setRowCount(self.tableWidget.rowCount()+1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def update\_1(self, query="select \* from Agent"): #после добавление сразу видно запись агент try:

cur = self.conn.cursor() data = cur.execute(query).fetchall() except Exception as d: print(f"Проблемы с подкл {d}") return d self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы

#заносим по новой for i, row in enumerate(data):

self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def insert\_1(self): #кнопка добавить

row = [self.comboBox.itemText(self.comboBox.currentIndex()), self.lineEdit.text(), self.lineEdit\_2.text(), self.lineEdit\_3.text(), self.lineEdit\_4.text(), self.lineEdit\_5.text(), self.lineEdit\_7.text(), self.lineEdit\_6.text(), self.lineEdit\_16.text()] try:

cur = self.conn.cursor()

cur.execute(f"""insert into Agent(id\_type\_agent, Naimenovanie, Mail\_agent, phone\_agent, Adress, priority, director, inn, kpp)

values('{row[0]}','{row[1]}','{row[2]}','{row[3]}','{row[4]}','{row[5]}','{row[6

]}','{row[7]}', '{row[8]}')""") self.conn.commit() cur.close() except Exception as r: print("Не смогли добавить запись") return r

self.update\_1()#обращаемся к update чтобы сразу увидеть изменения в БД

def deleteRow\_1(self): #удалить агент id = self.lineEdit\_8.text() conn = sqlite3.connect('Glazki.db') c = conn.cursor()

c.execute("DELETE FROM Agent WHERE id\_agent=?", (id,)) conn.commit() conn.close() self.update\_1()

def serch\_1(self): #поиск агент

search\_text = self.lineEdit\_9.text()

# Проходим по всем строкам таблицы for row in range(self.tableWidget.rowCount()): # Проходим по всем ячейкам строки for column in range(self.tableWidget.columnCount()):

# Получаем текст ячейки

cell\_text = self.tableWidget.item(row, column).text()

if search\_text.lower() in cell\_text.lower(): self.tableWidget.selectRow(row) return

Форма “Composition” class Composition1(QWidget, Ui\_composition): #композиция def \_\_init\_\_(self): super(Composition1, self).\_\_init\_\_()

self.setupUi(self)

self.pushButton.clicked.connect(self.open\_2)#открыть таблицу композиция self.pushButton\_2.clicked.connect(self.insert\_2) #добавить композиция self.pushButton\_3.clicked.connect(self.deleteRow\_2) #удалить композиция self.pushButton\_10.clicked.connect(self.serch\_2)#найти композиция

def open\_2(self): #открыть таблицу композиция try:

self.conn = sqlite3.connect('Glazki.db') cur = self.conn.cursor()

data = cur.execute("select \* from Composition;") col\_name = [i[0] for i in data.description] data\_rows = data.fetchall() except Exception as e:

print ("Ощибки с подключением к БД") return e

self.tableWidget.setColumnCount(len(col\_name)) self.tableWidget.setHorizontalHeaderLabels(col\_name) self.tableWidget.setRowCount(0) for i, row in enumerate(data\_rows):

self.tableWidget.setRowCount(self.tableWidget.rowCount()+1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def update\_2(self, query="select \* from Composition"): #после добавление композиция try:

cur = self.conn.cursor() data = cur.execute(query).fetchall() except Exception as d: print(f"Проблемы с подкл {d}") return d

self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы

#заносим по новой for i, row in enumerate(data):

self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1) for j, elen in enumerate(row):

self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def insert\_2(self): #кнопка добавить композиция

row = [self.lineEdit.text(), self.lineEdit\_2.text(), self.lineEdit\_3.text(), self.spinBox.text()] try:

cur = self.conn.cursor()

cur.execute(f"""insert into Composition(vendor\_code, receipt\_date, implementatin\_date, id\_agent )

values('{row[0]}','{row[1]}','{row[2]}','{row[3]}')""") self.conn.commit() cur.close() except Exception as r: print("Не смогли добавить запись") return r

self.update\_2()#обращаемся к update чтобы сразу увидеть изменения в БД

def deleteRow\_2(self): #удалить агент id = self.lineEdit\_8.text() conn = sqlite3.connect('Glazki.db') c = conn.cursor()

c.execute("DELETE FROM Composition WHERE id\_composition=?", (id,)) conn.commit() conn.close() self.update\_2()

def serch\_2(self): #поиск композиция search\_text = self.lineEdit\_9.text()

# Проходим по всем строкам таблицы for row in range(self.tableWidget.rowCount()): # Проходим по всем ячейкам строки for column in range(self.tableWidget.columnCount()):

# Получаем текст ячейки cell\_text = self.tableWidget.item(row, column).text() if search\_text.lower() in cell\_text.lower(): self.tableWidget.selectRow(row) return

Форма “Prodano” class Prodano1 (QWidget, Ui\_prodano): #продано def \_\_init\_\_(self): super(Prodano1, self).\_\_init\_\_() self.setupUi(self)

self.pushButton.clicked.connect(self.open\_3)#открыть таблицу композиция self.pushButton\_2.clicked.connect(self.insert\_3) #добавить self.pushButton\_3.clicked.connect(self.deleteRow\_3)#удалить продано self.pushButton\_10.clicked.connect(self.serch\_3)#найти

def open\_3(self): #открыть таблицу агент try:

self.conn = sqlite3.connect('Glazki.db') cur = self.conn.cursor() data = cur.execute("select \* from prodano;") col\_name = [i[0] for i in data.description] data\_rows = data.fetchall() except Exception as e:

print ("Ощибки с подключением к БД") return e

self.tableWidget.setColumnCount(len(col\_name)) self.tableWidget.setHorizontalHeaderLabels(col\_name) self.tableWidget.setRowCount(0) for i, row in enumerate(data\_rows):

self.tableWidget.setRowCount(self.tableWidget.rowCount()+1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def update\_3(self, query="select \* from prodano"): #обнова продано try:

cur = self.conn.cursor() data = cur.execute(query).fetchall() except Exception as d: print(f"Проблемы с подкл {d}") return d

self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы

#заносим по новой for i, row in enumerate(data):

self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def insert\_3(self): #кнопка добавить продано

row = [self.lineEdit.text(), self.lineEdit\_3.text(), self.lineEdit\_2.text(), self.spinBox.text()] try:

cur = self.conn.cursor()

cur.execute(f"""insert into prodano(vendor\_code, id\_agent, date\_implementation, number\_products )

values('{row[0]}','{row[1]}','{row[2]}',{row[3]})""") self.conn.commit() cur.close() except Exception as r: print("Не смогли добавить запись") return r self.update\_3()

def deleteRow\_3(self): #удалить продано id = self.lineEdit\_8.text() conn = sqlite3.connect('Glazki.db') c = conn.cursor()

c.execute("DELETE FROM prodano WHERE id=?", (id,)) conn.commit() conn.close() self.update\_3()

def serch\_3(self): #поиск продано

search\_text = self.lineEdit\_9.text()

# Проходим по всем строкам таблицы for row in range(self.tableWidget.rowCount()): # Проходим по всем ячейкам строки for column in range(self.tableWidget.columnCount()):

# Получаем текст ячейки cell\_text = self.tableWidget.item(row, column).text() if search\_text.lower() in cell\_text.lower(): self.tableWidget.selectRow(row) return

Форма “Product” class Product1(QWidget, Ui\_product): #продукт def \_\_init\_\_(self): super(Product1, self).\_\_init\_\_() self.setupUi(self)

self.pushButton.clicked.connect(self.open\_4)#открыть продукт self.pushButton\_2.clicked.connect(self.insert\_4)#добавить продукт self.pushButton\_3.clicked.connect(self.deleteRow\_4)#удалить продукт self.pushButton\_10.clicked.connect(self.serch\_4)#найти продукт

def open\_4(self): #открыть таблицу продукт try:

self.conn = sqlite3.connect('Glazki.db') cur = self.conn.cursor() data = cur.execute("select \* from Product;") col\_name = [i[0] for i in data.description] data\_rows = data.fetchall() except Exception as e:

print ("Ощибки с подключением к БД") return e

self.tableWidget.setColumnCount(len(col\_name)) self.tableWidget.setHorizontalHeaderLabels(col\_name) self.tableWidget.setRowCount(0) for i, row in enumerate(data\_rows):

self.tableWidget.setRowCount(self.tableWidget.rowCount()+1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j,QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def update\_4(self, query="select \* from Product"): #обнова продукт try:

cur = self.conn.cursor() data = cur.execute(query).fetchall() except Exception as d: print(f"Проблемы с подкл {d}") return d

self.tableWidget.setRowCount(0) #обнулмяем все данные из таблцы

#заносим по новой for i, row in enumerate(data):

self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1) for j, elen in enumerate(row): self.tableWidget.setItem(i, j, QTableWidgetItem(str(elen))) self.tableWidget.resizeColumnsToContents()

def insert\_4(self): #кнопка добавить продукт

row = [self.lineEdit.text(), self.spinBox\_3.text(), self.lineEdit\_2.text(), self.spinBox.text(), self.spinBox\_2.text() , self.lineEdit\_3.text()] try:

cur = self.conn.cursor()

cur.execute(f"""insert into Product(naimenovanie, id\_type\_product, vendor\_code, Number\_people\_product\_ion, Production\_workschop\_number, Min\_price\_agent )

values('{row[0]}','{row[1]}','{row[2]}',{row[3]}, '{row[4]}', '{row[5]}')""") self.conn.commit() cur.close() except Exception as r:

print("Не смогли добавить запись")

return r self.update\_4()

def deleteRow\_4(self): #удалить продукт id = self.lineEdit\_8.text() conn = sqlite3.connect('Glazki.db') c = conn.cursor()

c.execute("DELETE FROM Product WHERE id=?", (id,)) conn.commit() conn.close() self.update\_4()

def serch\_4(self): #поиск продукт

search\_text = self.lineEdit\_9.text()

# Проходим по всем строкам таблицы for row in range(self.tableWidget.rowCount()): # Проходим по всем ячейкам строки for column in range(self.tableWidget.columnCount()):

# Получаем текст ячейки cell\_text = self.tableWidget.item(row, column).text() if search\_text.lower() in cell\_text.lower():

self.tableWidget.selectRow(row) return

App = QtWidgets.QApplication([]) window = Login() window.show() App.exec()

**Заключение**

В ходе проделанной работы были выполнены все поставленные задачи: была разработана er-диаграмма по предметной области, по er диаграмме, которую мы разработали была создана база данных, которая была заполнена данными, также был разработан интерфейс непосредственно под саму базу данных, был создан файл main.py, в котором мы при помощи языка python связали формы и базу данных, назначили действия на кнопки и поля ввода, которые располагаются на форме.

Список литературы

<https://youtube.com/playlist?list=PLzMcBGfZo4-lB8MZfHPLTEHO9zJDDLpYj> - англоязычный корткий курс, состоящий из видеоуроков

<https://tproger.ru/translations/python-gui-pyqt/> - обучающий сайт с различными полезными материалами