Государственное бюджетное профессиональное учреждение



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

**Отчет: Создание базы данных**

**и интерфейса “learn school”**

Авторы:

Студенты специальности 09.02.07

“Информационные системы и программирование”,

2-ого курса, группы ИС-21

Брюханов Никита Анатольевич,

Островский Василий Дмитриевич,

Фомин Игорь Андреевич

Рецензент:

Тарджиманян Лия Николаевна

Оценка:\_\_\_\_\_

Дзержинский 2023 г

**Содержание:**

1. Введение.……………………………………...…………………………….…3
2. Техническое задание……………………………………………………….....4
3. Логическая структура………………………………………………………...5
4. Физическая структура………………………………………………….……..6
5. Работа с данными…..………………………………………………………8
6. Инструкция пользования…….………………………………………….…9
7. Разработка интерфейса…………………………………………………...…11
8. Заключение…………………………………………………………………..19

**Введение**

Цели

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

Задачи:

* Разработать er диаграмму по предметной области
* По разработанной er диаграмме создать базу данных
* Заполнить данными получившуюся базу данных
* Разработать интерфейс для работы с базой данных
* Связать интерфейс с базой данных с помощью программирования

Инструменты:

* SqlDraw(сайт для создания er диаграммы)
* Sqlite(для разработки базы данных)
* Qt5 дизайнер(для создания визуальной составляющей интерфейса)
* visual studio code(ide для программирования)

Выбор СУБД.

* Sqlite

**Техническое задание**

Услуги школы иностранных языков

Вашей задачей является разработка системы для компании, которая оказывает услуги клиентам и продаёт определенные товары.

Данную систему можно разделить на несколько подсистем:

• подсистема по работе с клиентами,

• подсистема по работе с сотрудниками,

• подсистема по работе с товарами,

• подсистема по работе с услугами,

• подсистема для управления компанией.

Основное задание на разработку

Школа языков “Леарн” открывает большое количество дополнительных направлений в рамкахоказаний услуг (например: подготовка к ЕГЭ, тренировки к TOEFL, выездные кемпинги в другие страны и много другое). В связи с этим нам важно, чтобы родители слушателей, а также наши взрослые слушатели могли ознакомиться со списком абсолютно всех услуг на наших компьютерах, пока они ждут преподавателя на очередное занятие. В это же время для администраторов должны быть доступны функции управления списком услуг.

Вам предстоит разработать подсистему для работы с услугами нашей компании, которая должна включать в себя следующий функционал:

• просмотр списка услуг,

• добавление/удаление/редактирование данных об услугах,

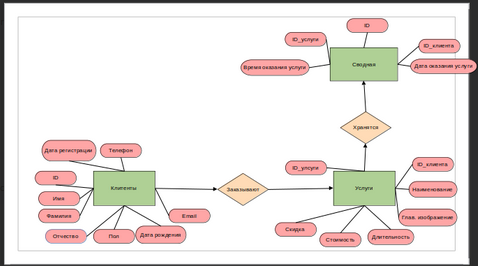
• работа с дополнительными изображениями к услугам,

• запись клиента на услугу,

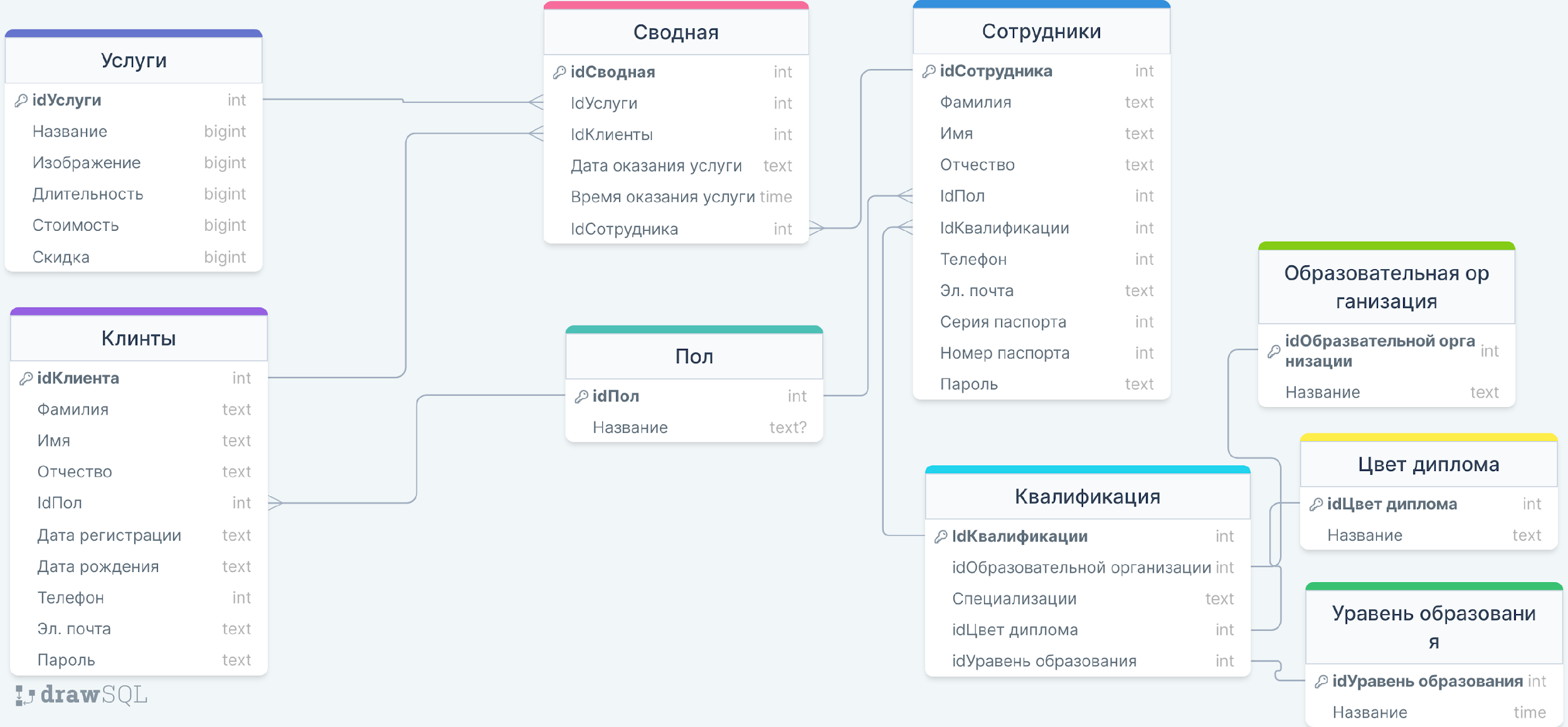
• управление ближайшими записями на услуги школы иностранных языков.

**Логическая структура**

Проектирование логической структуры базы данных.



Так как полученная er-диаграмма не соответствует требованиям технического задания нами была разработана новая ER-диаграмма.



**Физическая структура**

Проектирование физической структуры базы данных.

На базе er-диаграммы создаём базу данных в Sqlite. У нас есть три сущности: услуга, клиенты, сводная.

Таблица “Сводная” имеет следующие атрибуты:

IdСводная

IdУслуги

IdКлиента

IdСотрудника

Время оказания услуги

Дата оказания услуги

Таблица “Клиенты” имеет следующие атрибуты:

IdКлиента

Фамилия

Имя

Отчество

Номер телефона

Электронная почта

Пол

Дата рождения

Дата регистрации

Пароль

Таблица “Услуги” имеет следующие атрибуты:

IdУслуги

IdКлиента

Название

Главное изображение

Длительность

Стоимость

Скидка

Таблица “Пол” имеет следующие атрибуты:

IdПол

Название

Таблица “Цвет диплома” имеет следующие атрибуты:

IdЦвет диплома

Название

Таблица “Уровень образования” имеет следующие атрибуты:

IdУровень образования

Название

Таблица “Образоваетльная организация” имеет следующие атрибуты:

IdОбразовательная организации

Название

Таблица “Сотрудники” имеет следующие атрибуты:

IdСотрудники

Фамилия

Имя

Отчество

Пол

IdКвалификация

Номер телефона

Электронная почта

Серия паспорта

Номер паспорта

Пароль

Таблица “Квалификация” имеет следующие атрибуты:

IdКвалификация

IdОбразовательной организации

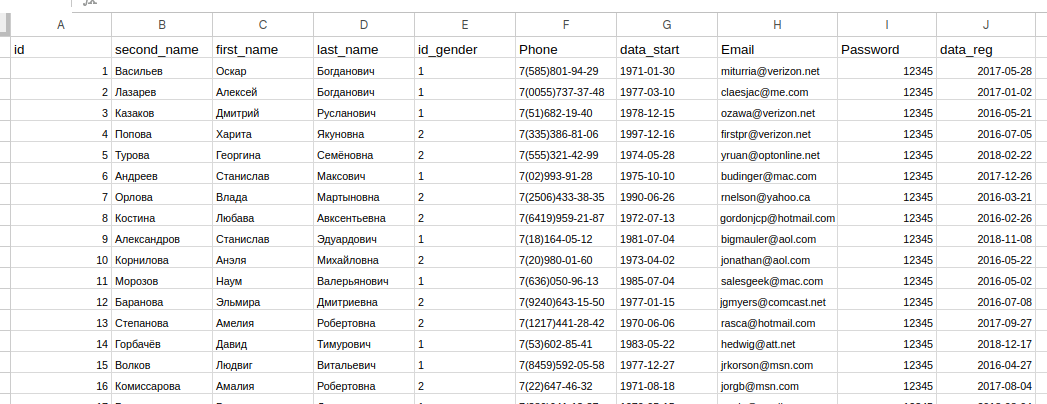
Специализация

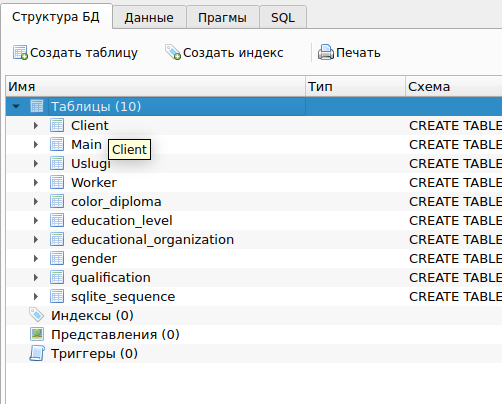
IdЦвет диплома

IdУровень образования

**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.

Организация корректировки БД.

Корректироваться данных производится с формы, или непосредственно в Sqlite.

**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
* Изменение данных в таблице
* Удаление данных из базы данных
* Поиск отсортированной информации и ее вывод

Авторизация:

Если Вы уже зарегистрированы для работы с формами(интерфейсом), то Вам необходимо просто внести данные в соответствующие поля без ошибок и Вас перенесет на страницу, которая доступна вам в связи с вашими провами доступа.

Для того чтобы войти от администратора и начать редоктировать базу данных через формы необходимо ввести данные «0000» в обе строки на странице авторизации.

Но если Вы - являетись клиентом и не зарегистрированы, то перед началом работы с формами(интерфейсом) нужно зарегистрироваться. Это можно сделать нажав на кнопку “регистрация” и заполнить все требываемые данные. После всех выполненных действий можете авторизоваться.

Работа с формами бд (Администратор)

Добавление новых записей

По завершению авторизации. если все предыдущие шаги были выполнены верно, Вы попадаете на новую форму “Клиенты”. Для того, чтобы начать работать с формой необходимо нажать на кнопку “открыт”, после чего внизу должна открыться таблица из 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”

Изменение данных в таблице

Для того, чтобы изменить данные в таблице необходимо навести курсор на ячейку таблицы, в которой Вы хотите произвести изменения и несколько раз нажать на нее, после чего изменить данные.

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”. которая располагается в одной колонке с кнопками “Открыть” и “Добавит”.

Поиск по определенным критериям

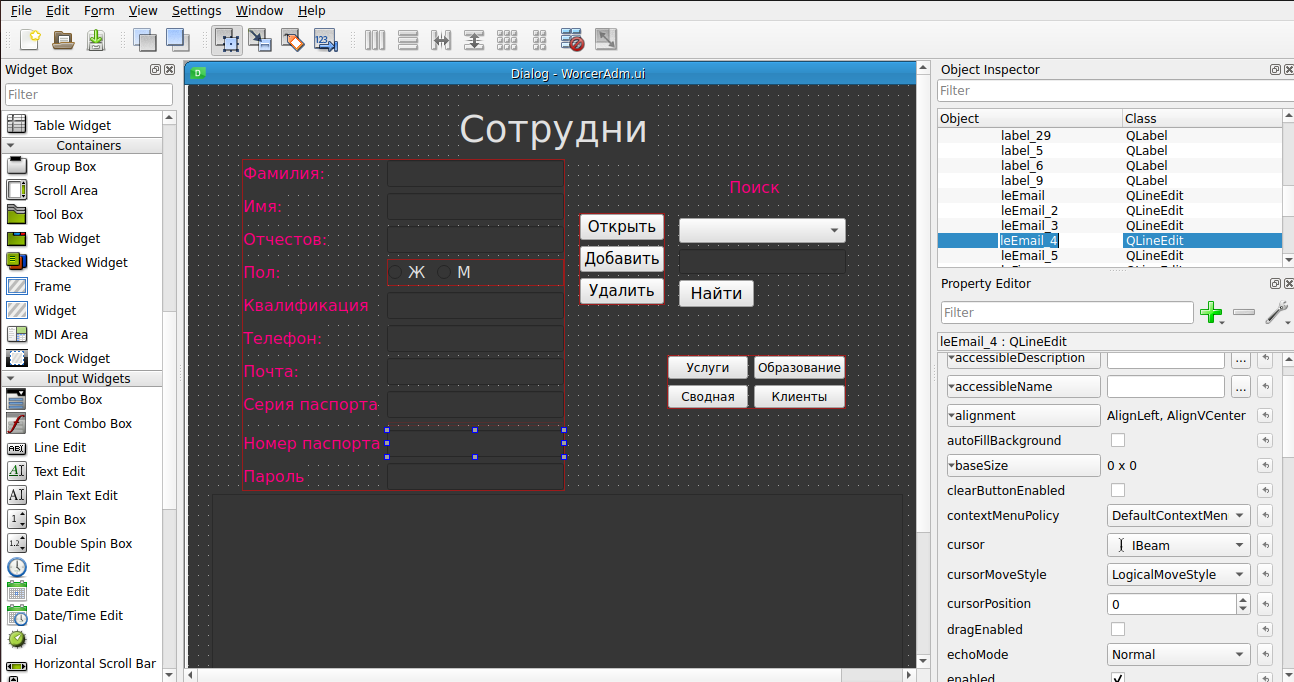
Также через формы можно выводить данные, которые нужно отсортировать по тому или иному критерию. Для этого в разделе поиска нажимаем на выпадающее окно, где представлены названия столбцов, и выбираем название столбца, по значением которого нам нужно вывести данные. Ниже, в окошке для ввода данных нужно указать значения, по которым мы будем сортировать и нажимаем кнопку “Найти”. В окошко, через которое отображается таблица, будут выведены необходимые отсортированные данные.

Переключение между формами

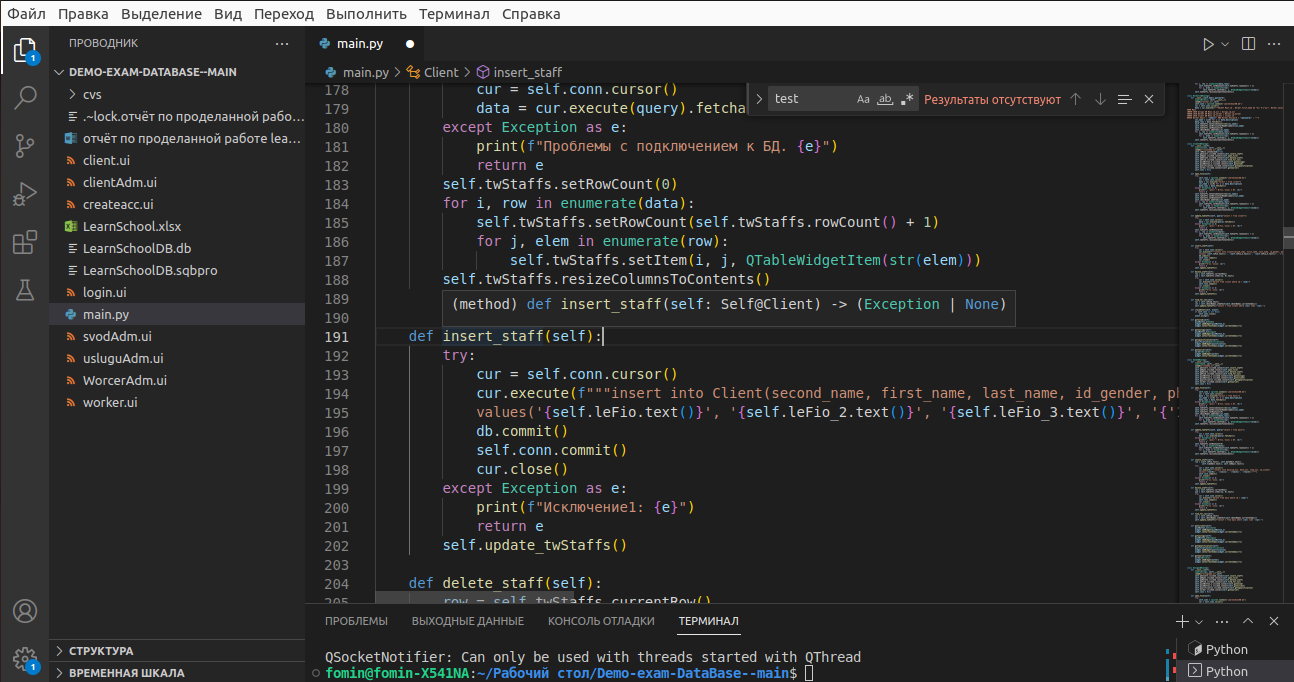
Чтобы переключаться между формами предусмотрены кнопки, которые располагаются под блоком “Поиск” или под фиджетом, который проецирут бд. Нажимая на одну из кнопок Вы будете переходить на другую форму. Все формы работаю аналогично форме “Клиенты”.

**Разработка интерфейса.**

Для начала в QT 5 дизайнере необходимо создать внешний вид форм, перенести на окно необходимые кнопки, поля для ввода данных, текстовые значения. После создания внешнего вида формы нам необходимо добавить функции на кнопки и поля ввода, чтобы наш интерес исправно функционировал.



Назначение команд для функционирования форм



Подключение необходимых библиотек для работы с базами данных и формами

import sys

import sqlite3

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QDialog, QApplication, QTableWidgetItem

from PyQt5.uic import loadUi

Подключение к базе данных

db = sqlite3.connect("LearnSchoolDB.db")

sql = db.cursor()

Форма Авторизации

class Login(QDialog):

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

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

loadUi("login.ui",self)

self.password.setEchoMode(QtWidgets.QLineEdit.Password)

self.loginbutton.clicked.connect(self.loginfunction)

self.createaccbutton.clicked.connect(self.gotocreate)

Функция логирования пользователя

def loginfunction(self):

email=self.email.text()

password=self.password.text()

Вход для администратора

if email == "0000" and password == "0000":

print("Вы зашли как админ")

loginbutton=Client()

widget.addWidget(loginbutton)

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

Поиск данных в таблице сотрудники для авторизации и открытия личного профиля

sql.execute(f"SELECT \* FROM Worker WHERE Email = '{email}' AND Password = '{password}';")

db.commit()

if sql.fetchone() == None:

Поиск данных в таблице клиенты для авторизации и открытия личного профиля

sql.execute(f"SELECT \* FROM Client WHERE Email = '{email}' AND Password = '{password}';")

if sql.fetchone() == None:

print("Такого пользователя нет")

else:

Функция на кнопку входа клиенту

print('Welcome')

loginbutton=ClientLog(email,password)

widget.addWidget(loginbutton)

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

Функция на кнопку входа сотруднику

else:

print('Welcome')

loginbutton=Workers(email,password)

widget.addWidget(loginbutton)

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

Функция на кнопку регистрации для создания личного профиля клиенту

def gotocreate(self):

createaccbutton=CreateAcc()

widget.addWidget(createaccbutton)

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

Регистрация клиента

class CreateAcc(QDialog):

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

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

loadUi("createacc.ui",self)

self.signupbutton.clicked.connect(self.createaccfunction)

self.conn = None

def createaccfunction(self):

email = self.leEmail.text()

password=self.leEmail\_2.text()

Поис введенных данных в базе данных

sql.execute(f"SELECT Email, Password FROM Client WHERE Email = '{email}' AND Password = '{password}'")

Если данные,которые ввел пользователь ввел не найдены в бд то происходит функция регистрации

if sql.fetchone() is None:

sql.execute(f"""insert into Client(second\_name, first\_name, last\_name, id\_gender, phone, data\_start, Email, Password, data\_reg)

values('{self.leFio.text()}', '{self.leFio\_2.text()}', '{self.leFio\_3.text()}', '{'1' if self.rbMale.isChecked() else '2'}', '{self.sbAge.text()}', '{self.lePhone.text()}', '{self.leEmail.text()}', '{self.leEmail\_2.text()}', '{self.sbAge\_2.text()}')""")

db.commit()

print('You have registered')

Функция на кнопку регистрации, чтобы после регистрации пользователя возвращало на страницу авторизации

signupbutton=Login()

widget.addWidget(signupbutton)

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

else:

print('Такая запись уже существует')

for i in sql.execute('SELECT \* FROM users'):

print(i)

Форма клиента, которая отражает личные данные пользователя

class ClientLog(QDialog):

def \_\_init\_\_(self,email,password):

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

loadUi("client.ui",self)

self.conn = sqlite3.connect('LearnSchoolDB.db')

cur = self.conn.cursor()

Запрос на вывод данных в личный профиль клиенту

data = cur.execute(f"""SELECT Client.id, Client.first\_name AS "Имя клиента", Client.second\_name AS "Фамилия клиента", Client.last\_name AS "Отчество клиента", Uslugi.name\_usl AS "Название услуги", Worker.first\_name AS "Имя учителя", Worker.second\_name AS "Фамилия учителя",Worker.last\_name AS "Отчество учителя", Main.date\_usl AS "дата начала", Main.time\_usl AS "Время начала"

FROM Main

INNER JOIN Uslugi ON Main.id\_usl = Uslugi.id\_usl

INNER JOIN Worker ON Main.id\_worker = Worker.id\_worker

INNER JOIN Client ON Main.id\_client = Client.id

WHERE Client.Email = "{email}" AND Client.Password = "{password}" ; """)

col\_name = [i[0] for i in data.description]

data\_rows = data.fetchall()

self.twStaffs.setColumnCount(len(col\_name))

self.twStaffs.setHorizontalHeaderLabels(col\_name)

self.twStaffs.setRowCount(0)

self.cbColNames.addItems(col\_name)

for i, row in enumerate(data\_rows):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

Форма сотрудники, которая отражает личные данные пользователя

class Workers(QDialog):

def \_\_init\_\_(self,email,password):

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

loadUi("worker.ui",self)

self.conn = sqlite3.connect('LearnSchoolDB.db')

cur = self.conn.cursor()

Запрос на вывод данных в личный профиль сотруднику

data = cur.execute(f"""SELECT Main.id , Worker.first\_name AS "Имя учителя", Worker.second\_name AS "Фамилия учителя",Worker.last\_name AS "Отчество учителя", Client.id AS "id клиента", Client.second\_name AS "Имя клиента", Client.first\_name AS "Фамилия клиента", Client.last\_name AS "Отчество клиента", Uslugi.name\_usl AS "Название услуги", Main.date\_usl AS "дата начала", Main.time\_usl AS "Время начала"

FROM Main

INNER JOIN Uslugi ON Main.id\_usl = Uslugi.id\_usl

INNER JOIN Worker ON Main.id\_worker = Worker.id\_worker

INNER JOIN Client ON Main.id\_client = Client.id

WHERE Worker.Email = "{email}" AND Worker.Password = "{password}" ; """)

col\_name = [i[0] for i in data.description]

data\_rows = data.fetchall()

self.twStaffs.setColumnCount(len(col\_name))

self.twStaffs.setHorizontalHeaderLabels(col\_name)

self.twStaffs.setRowCount(0)

self.cbColNames.addItems(col\_name)

for i, row in enumerate(data\_rows):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

Форма клиента для администратора

class Client(QDialog):

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

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

loadUi("clientAdm.ui",self)

self.rbMale.setChecked(True)

self.pbInsert.clicked.connect(self.insert\_staff)

self.pbOpen.clicked.connect(self.open\_file)

self.pbDelete.clicked.connect(self.delete\_staff)

self.pbFind.clicked.connect(self.find\_for\_val)

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

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

self.qualification.clicked.connect(self.gotoqualification)

self.worker.clicked.connect(self.gotoworcer)

self.conn = None

Открытие таблицы Клиенты

def open\_file(self):

try:

self.conn = sqlite3.connect('LearnSchoolDB.db')

cur = self.conn.cursor()

data = cur.execute("select \* from client")

col\_name = [i[0] for i in data.description]

data\_rows = data.fetchall()

except Exception as e:

print(f"Проблемы с подключением к БД. {e}")

return e

self.twStaffs.setColumnCount(len(col\_name))

self.twStaffs.setHorizontalHeaderLabels(col\_name)

self.twStaffs.setRowCount(0)

self.cbColNames.addItems(col\_name)

for i, row in enumerate(data\_rows):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

Изменение данных (сортировки)

def update\_twStaffs(self, query="select \* from client"):

try:

cur = self.conn.cursor()

data = cur.execute(query).fetchall()

except Exception as e:

print(f"Проблемы с подключением к БД. {e}")

return e

self.twStaffs.setRowCount(0)

for i, row in enumerate(data):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

Запрос на добавление новых клиентов в таблицу

def insert\_staff(self):

try:

cur = self.conn.cursor()

cur.execute(f"""insert into Client(second\_name, first\_name, last\_name, id\_gender, phone, data\_start, Email, Password, data\_reg)

values('{self.leFio.text()}', '{self.leFio\_2.text()}', '{self.leFio\_3.text()}', '{'1' if self.rbMale.isChecked() else '2'}', '{self.sbAge.text()}', '{self.lePhone.text()}', '{self.leEmail.text()}', '{self.leEmail\_2.text()}', '{self.sbAge\_2.text()}')""")

db.commit()

self.conn.commit()

cur.close()

except Exception as e:

print(f"Исключение1: {e}")

return e

self.update\_twStaffs()

Запрос на удаление данных из таблицы

def delete\_staff(self):

row = self.twStaffs.currentRow()

num = self.twStaffs.item(row, 0).text()

try:

cur = self.conn.cursor()

SQL запрос

cur.execute(f"delete from client where id = {num}")

self.conn.commit()

cur.close()

except Exception as e:

print(f"Исключение: {e}")

return e

self.update\_twStaffs()

Сортировка данных по определенным значением

def find\_for\_val(self):

val = self.leFind.text()

col = self.cbColNames.itemText(self.cbColNames.currentIndex())

self.update\_twStaffs(f"select \* from client where {col} like '{val}'")

Кнопки, которые переходят на другие формы

def gotosvod(self):

pushButton\_2=Swod()

widget.addWidget(pushButton\_2)

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

def gotouslugu(self):

pushButton\_3=Uslugu()

widget.addWidget(pushButton\_3)

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

def gotoqualification(self):

qualification=Qualification()

widget.addWidget(qualification)

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

def gotoworcer(self):

worker=Worcer()

widget.addWidget(worker)

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

Все остальные формы работают аналогично форме “Клиенты”.

**Заключение**

В ходе проделанной работы были закреплены теоретические знаний, а также навыки проектирования БД, полученные при изучении дисциплины «Базы данных».

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 интерес непосредственно под саму базу данных, был создан файл main.pu, в где мы при помощи языка python связали формы и базу данных, назначили действия на кнопки и поля ввода, расположенных на форме.