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:

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
- 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:

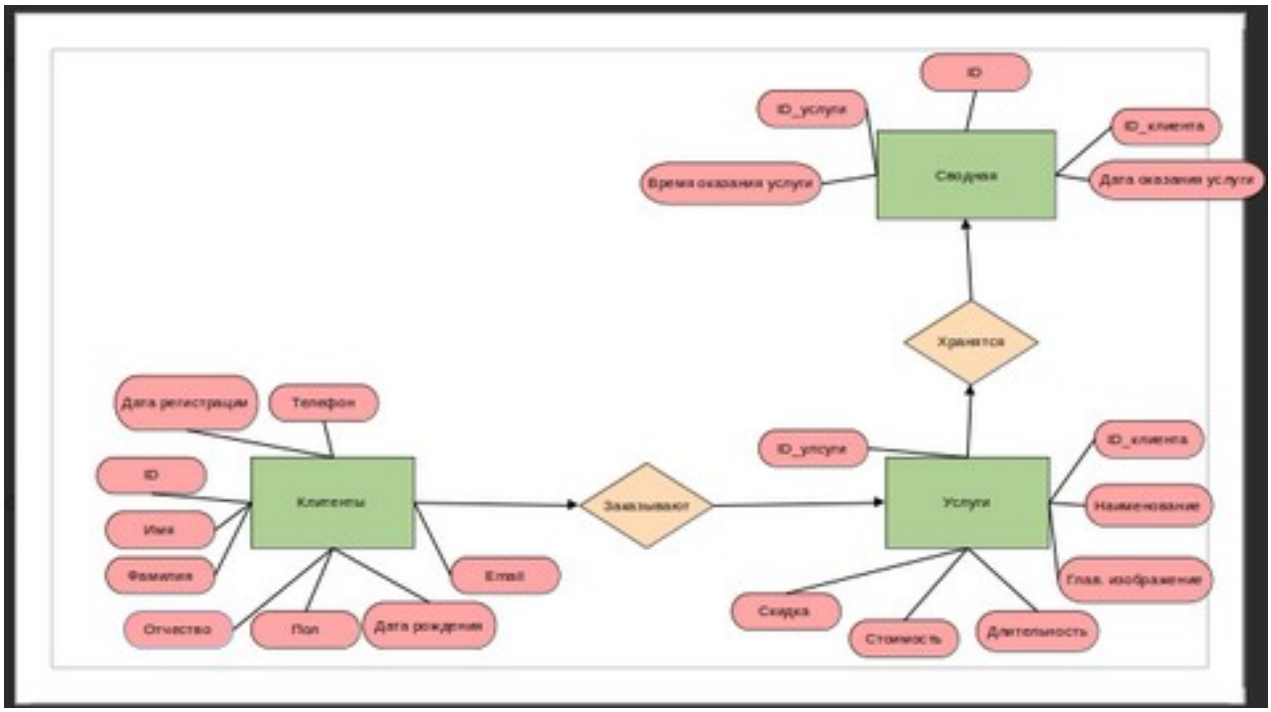
- SqlDraw(сайт для создания er диаграммы)
- Sqlite(для разработки базы данных)
- Qt5 дизайнер(для создания визуальной составляющей интерфейса)
- visual studio code(ide для программирования)

Выбор СУБД.

- Sqlite

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 три сущности: услуга, клиенты, сводная.

Таблица “Сводная” имеет следующие атрибуты:
Id
IdУслуги
IdКлиента
Время оказания услуги
Дата оказания услуги

Таблица “Клиенты” имеет следующие атрибуты:
Id
Фамилия
Имя
Отчество
Номер телефона
Электронная почта
Пол
Дата рождения
Дата регистрации

Таблица “Услуги” имеет следующие атрибуты:
Id
IdКлиента
Название
Главное изображение
Длительность
Стоимость
Скидка

Таблица “Пользователи”, была создана специально для регистрации и логирования пользователя, имеет следующие атрибуты:
Электронная почта
Пароль

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 “Клиенты”...

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. Во вторую строку “пароль” нужно ввести пароль, который Вы указали выше. 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 “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 назание 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”. 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

```
Импортирование необходимых библиотек
import sys
import sqlite3
from PyQt5 import QtWidgets
from   PyQt5.QtWidgets   import   QDialog,   QApplication,
QTableWidgetItem
from PyQt5.uic import loadUi

Подключение к бд
db = sqlite3.connect("school(1).db")
sql = db.cursor()

Логирование
class Login(QDialog):
   def __init__(self):
       super(Login,self).__init__()
       loadUi("login.ui",self)

       self.password.setEchoMode(QtWidgets.QLineEdit.Pass
word)
       self.loginbutton.clicked.connect(self.loginfunctio
n)
       self.createaccbutton.clicked.connect(self.gotocrea
te)
```

```
    def loginfunction(self):
Запрос у пользователя данных для входа
        email=self.email.text()
        password=self.password.text()
Поиск данных, которые ввел пользователь, в бд
        sql.execute(f"SELECT * FROM users WHERE username =
'{self.email.text()}'        AND        password        =
'{self.password.text()}';")
        db.commit()

Цикл на авторизацию
        if sql.fetchone() == None:
            print("Нет такой записи")
        else:
            print('Welcome')
            loginbutton=Client()
            widget.addWidget(loginbutton)
              widget.setCurrentIndex(widget.currentIndex()
+1)



Кнопка, которая отправляет пользователя на страницу
регистрации
    def gotocreate(self):
        createaccbutton=CreateAcc()
        widget.addWidget(createaccbutton)
        widget.setCurrentIndex(widget.currentIndex()+1)

Регистрация
class CreateAcc(QDialog):
    def __init__(self):
        super(CreateAcc,self).__init__()
```

```
        loadUi("createacc.ui",self)
        self.signupbutton.clicked.connect(self.createaccfu
nction)
        self.password.setEchoMode(QtWidgets.QLineEdit.Pass
word)
        self.confirmpass.setEchoMode(QtWidgets.QLineEdit.P
assword)
        self.conn = None

    def createaccfunction(self):
Запрос на ввод данных от пользователя
        email = self.email.text()
Поиск данных, которые ввел пользователь
        password=self.password.text()
        if self.password.text()==self.confirmpass.text():
Запрос на поиск информации, которую ввел пользователь
                    sql.execute(f"SELECT username, password
FROM users WHERE username = '{email}' AND password =
'{password}'")
Цикл на регистрацию
        if sql.fetchone() is None:
Запрос на добавление новых записей в таблицу для
регистрации пользователя
            sql.execute(f"INSERT INTO users VALUES (?,?)",
(email, password))
            db.commit()
            print('You have registered')
            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)


Первая форма “Клиенты”
class Client(QDialog):
    def __init__(self):
        super(Client, self).__init__()
        loadUi("client.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.conn = None
Кнопк открытия БД
    def open_file(self):
        try:
            self.conn = sqlite3.connect('school(1).db')
            cur = self.conn.cursor()
Запрос на вывод таблицы
            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.rowCou
nt() + 1)
            for j, elem in enumerate(row):
                                self.twStaffs.setItem(i,  j,
QTableWidgetItem(str(elem)))
        self.twStaffs.resizeColumnsToContents()
        self.avg_age()
Изменение данных в таблице и ее вывод
       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.rowCou
nt() + 1)
            for j, elem in enumerate(row):
                                self.twStaffs.setItem(i,  j,
QTableWidgetItem(str(elem)))
        self.twStaffs.resizeColumnsToContents()

Добавление данных в таблицу
   def insert_staff(self):
           row  =  [self.leFio.text(),  self.leFio_2.text(),
self.leFio_3.text(),  'м' if self.rbMale.isChecked() else
'ж', self.lePhone.text(), self.sbAge.text(),
               self.leEmail.text(), self.sbAge_2.text()]
```

```
        try:
            cur = self.conn.cursor()
Запрос на добавление новых данных в таблицу
                                cur.execute(f"""insert   into
client(second_name, first_name, last_name, gender, phone,
data_start, Email, data_reg)
                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 e:
            print(f"Исключение1: {e}")
            return e
        self.update_twStaffs()

    def delete_staff(self):
        row = self.twStaffs.currentRow()
        num = self.twStaffs.item(row, 0).text()
        try:
            cur = self.conn.cursor()
Запрос на удаление данных из таблицы
               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()

Поиск информации с сортировкой
    def find_for_val(self):
```

```
        val = self.leFind.text()
                                                col        =
self.cbColNames.itemText(self.cbColNames.currentIndex())
Запрос на сортировку
         self.update_twStaffs(f"select * from client where
{col} like '{val}%'")

    def closeEvent(self, event):
        if self.conn is not None:
            self.conn.close()
        event.accept()
Кнопк, которая переходит на форму “Сводная”
    def gotosvod(self):
        pushButton_2=Swod()
        widget.addWidget(pushButton_2)
        widget.setCurrentIndex(widget.currentIndex()+1)
Кнопк, которая переходит на форму “Услуги”
    def gotouslugu(self):
        pushButton_3=Uslugu()
        widget.addWidget(pushButton_3)
        widget.setCurrentIndex(widget.currentIndex()+1)
```

Формы: “Сводная” и “Услуги” работают аналогично форме “Клиенты”

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