**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю.А.Гагарина»**

**КУРСОВАЯ РАБОТА**

по профессиональному модулю (дисциплине)

технология разработки и защиты баз данных

Курс 2 Группа № ИС-21

Тема: БД автоматизированного учёта средств вычислительной техники, планирования и прогнозирования профилактического обслуживания

Выполнил/а/ студент \_\_\_\_\_\_\_\_\_\_\_\_\_ Островский Василий Дмитриевич (подпись) (ФИО полностью)

Руководитель Тарджиманян Лия Николаевна

(подпись) (ФИО полностью)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дзержинский 2023

**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…………………………………………………………………..24

**Введение:**

**Описание предметной области:**

Эта база данных предназначена для автоматизированного учёта средств вычислительной техники, планирования и прогнозирования профилактического обслуживания, чтобы использовать данный интерфейс для БД достаточно просто его запустить и по интуитивно понятному интерфейсу начинать добавлять данные оборудования для предприятия, далее можно отслеживать у какое оборудование закончился срок эксплуатации в отдельной таблице или же добавлять оборудования в таблицу сломанного оборудования.

**Цель:**

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
* Разработать sql запросы чтобы автоматизировать добавление определенных данных в таблицы
* Связать интерфейс с базой данных с помощью программирования

**Инструменты:**

* SqlDraw(сайт для создания er диаграммы)
* Sqlite(для разработки базы данных и добавление данных в таблицы)
* Qt5 дизайнер(для создания визуальной составляющей интерфейса)
* Python(язык программирования для того чтобы связать окружение и таблиц)
* PyQt5(Python библиотека для работы с окружениями)
* Sqlite3(Python библиотека для работы с базами данными)
* Visual studio code(Ide для программирования)

**Выбор СУБД:**

* Sqlite (удобная СУБД с простым функционалом и интерфейсом, что достаточно удобно)

**Техническое задание:**

Обработчик оборудования для предприятий

Моей задачей являлась разработка Разработать БД автоматизированного учёта средств вычислительной техники, планирования и прогнозирования профилактического обслуживания и окружение для этой базы данных, чтобы у администратора этой базы данных была возможность удобно и просто добавлять, смотреть, удалять данные в эту базу данных.

**Данную систему можно разделить на несколько подсистем:**

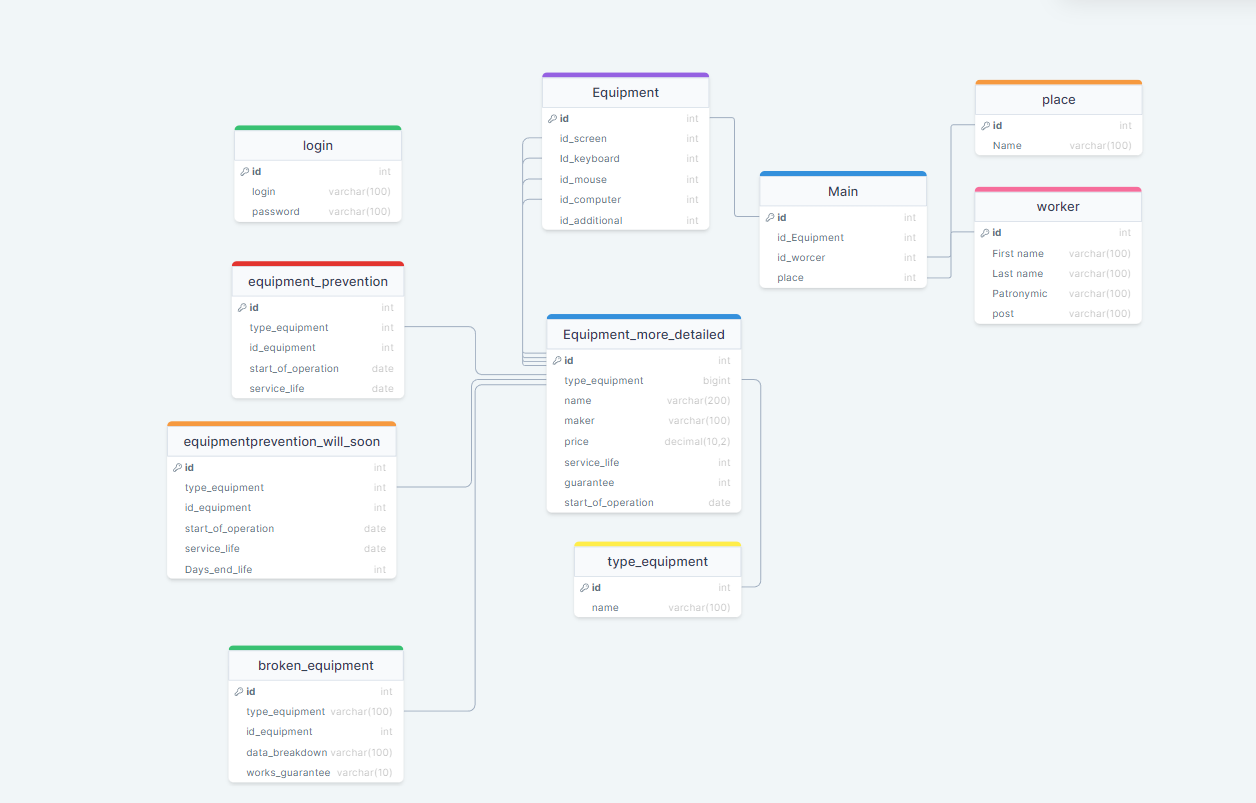
* подсистема по работе с сотрудниками
* подсистема по работе с оборудованием
* подсистема с обработчиком информации по оборудованию
* подсистема по работе с кабинетами предприятия

**Функционал:**

* Просмотр списка оборудования,сотрудников,кабинетов,типов оборудования и их атрибутов
* Добавление/удаление/редактирование этих данных
* Просмотр списка сломанного, с истекшим сроком службы или списка отсортированного по количеству дней до истечения срока службы оборудования и их атрибуты

**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 девять сущности: главная, кабинеты, работники, оборудование, оборудование детально, тип оборудования, поломанное оборудование, оборудование с истекшим сроком службы, оборудование у которого скоро истечет срок службы.

Таблица “главная” имеет следующие атрибуты:

Id

Id\_оборудования

Id\_сотрудника

Id\_кабинета

Таблица “кабинеты” имеет следующие атрибуты:

Id

Название

Таблица “работники” имеет следующие атрибуты:

Id

Фамилия

Имя

Отчество

Должность

Таблица “оборудование” имеет следующие атрибуты:

Id

Id\_монитора

Id\_клавиатуры

Id\_мыши

Id\_компьютера

Id\_дополнительного\_оборудования

Таблица “оборудование детально” имеет следующие атрибуты:

Id

Тип\_оборудования

Название

Производитель

Цена

Срок службы

Гарантия

Начало работы

Таблица “тип оборудования” имеет следующие атрибуты:

Id

Название

Таблица “поломанное оборудование” имеет следующие атрибуты:

Id

Тип оборудования

Дата поломки

Действует ли гарантия

Таблица “оборудование с истекшим сроком службы” имеет следующие атрибуты:

Id

Тип оборудования

Начало работы

Срок службы

Таблица “оборудование у которого скоро истечет срок службы” имеет следующие атрибуты:

Id

Тип оборудования

Начало работы

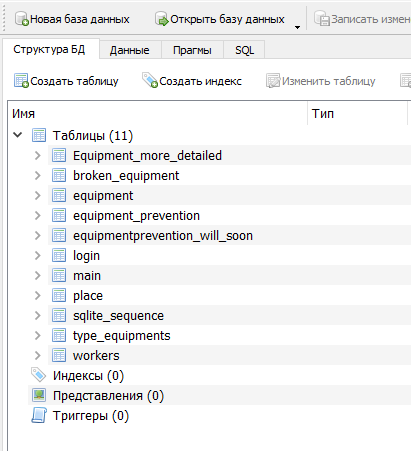
Срок службы

Дней до конца срока службы

**Работа с данными:**

Организация ввода данных в БД.

Первичный ввод записей в базу данных будет производится при помощи записи их напрямую из СУБД Sqlite. Далее ввод данных будет производится непосредственно с формы.



Организация корректировки БД.

Корректироваться данных производится с формы, или непосредственно в Sqlite.

**Инструкция пользования :**

Описание потребностей администраторов и выбор способов их реализации.

У администраторов могут возникнуть следующие потребности при пользовании интерфейсом:

* Авторизация на форме
* Добавление данных в базу данных
* Изменение данных в таблице
* Удаление данных из базы данных
* Поиск отсортированной информации и ее вывод
* Просмотр информации по планированию и прогнозированию профилактического обслуживания
* Просмотр информации по действительности гарантии на поломанное оборудование

**Авторизация:**

На форме логин необходимо ввести ваши административные данные от аккаунта, на данной версии окружения есть тестовые данные

Логин:admin

Пароль:admin

Если вы ввели данные правильно вас перекинет на окно “меню” где Вы можете выбрать с какой таблицей Вы хотите работать.

**Работа с формами бд:**

**Просмотр данных:**

Чтобы просмотреть записи достаточно открыть форму с таблицей и нажать кнопку “Открыть”

**Добавление новых записей:**

Чтобы добавить новые записи достаточно открыть форму с таблицей и в поля ввода данных ввести информацию и нажать 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:**

Чтобы просмотреть записи достаточно открыть форму с таблицей, выбрать строку которую вы хотите удалить и нажать кнопку “Удалить”

**Поиск по определенным критериям:**

Чтобы просмотреть записи по определенным критериям достаточно открыть форму с таблицей, открыть таблицу, выбрать атрибут в выпадающем списке справа вверху, ввести данные по которым вы хотите найти записи и нажать кнопку “Поиск”

**Переключение между формами:**

Чтобы переключаться между формами нужно нажимать кнопки с соответствующими названиями в форме “Меню” в котором Вы оказываетесь сразу как авторизуетесь, чтобы вернутся на эту форму на каждой форме внизу есть кнопка “Меню”.

**Просмотр информации по планированию и прогнозированию профилактического обслуживания:**

Чтобы посмотреть у какого оборудования истёк срок эксплуатации нужно перейти на окно “Профилактика обслуживания” и нажать кнопку “открыть”. Добавлять самостоятельно данные туда нельзя так как это делается автоматически.

Чтобы посмотреть у какого скоро истечет срок годности нужно перейти на окно “Скорая профилактика обслуживания” и тут всё по аналогии с окном

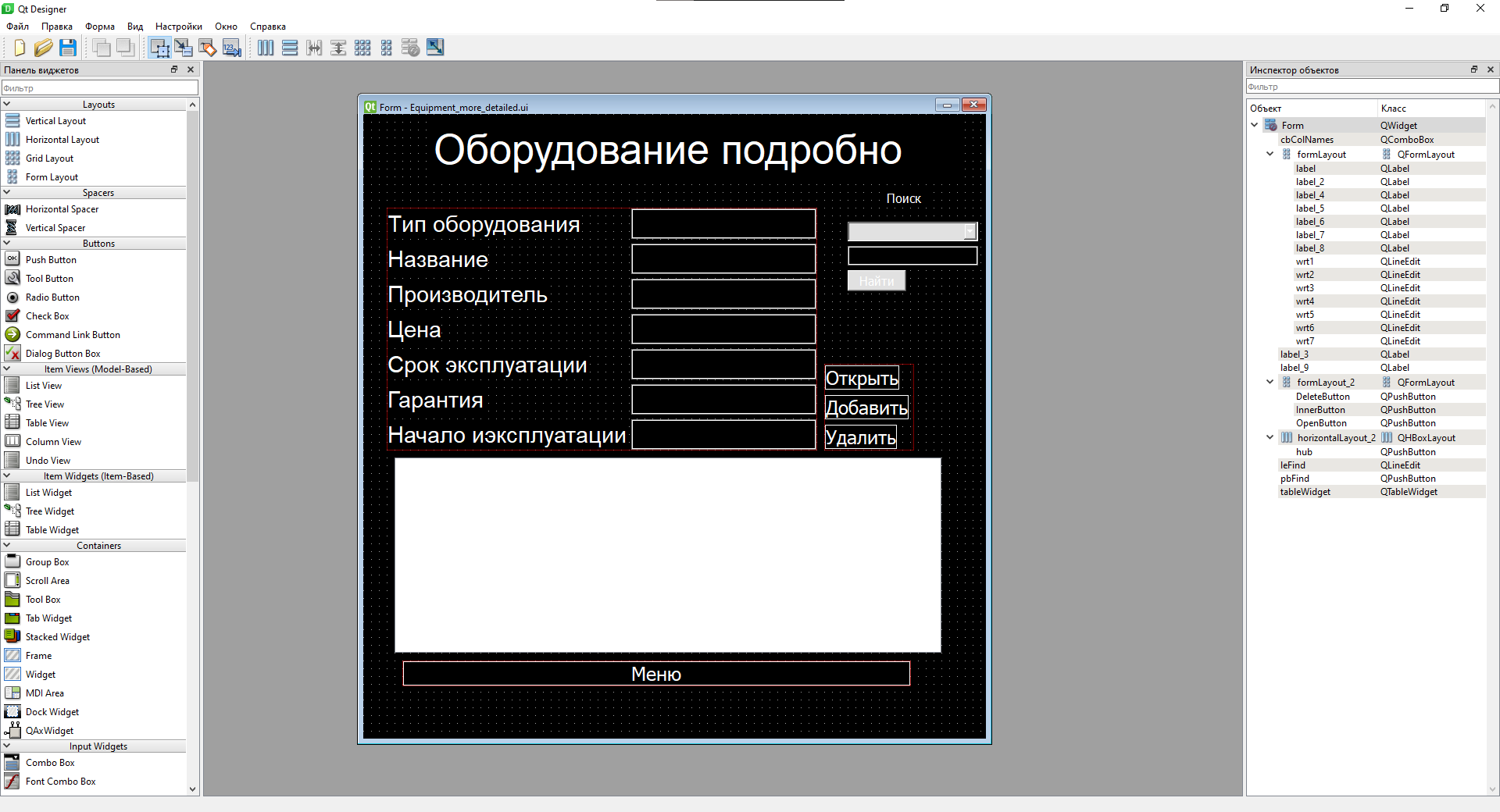
“Профилактика обслуживания”

**Просмотр информации по действительности гарантии на поломанное оборудование:**

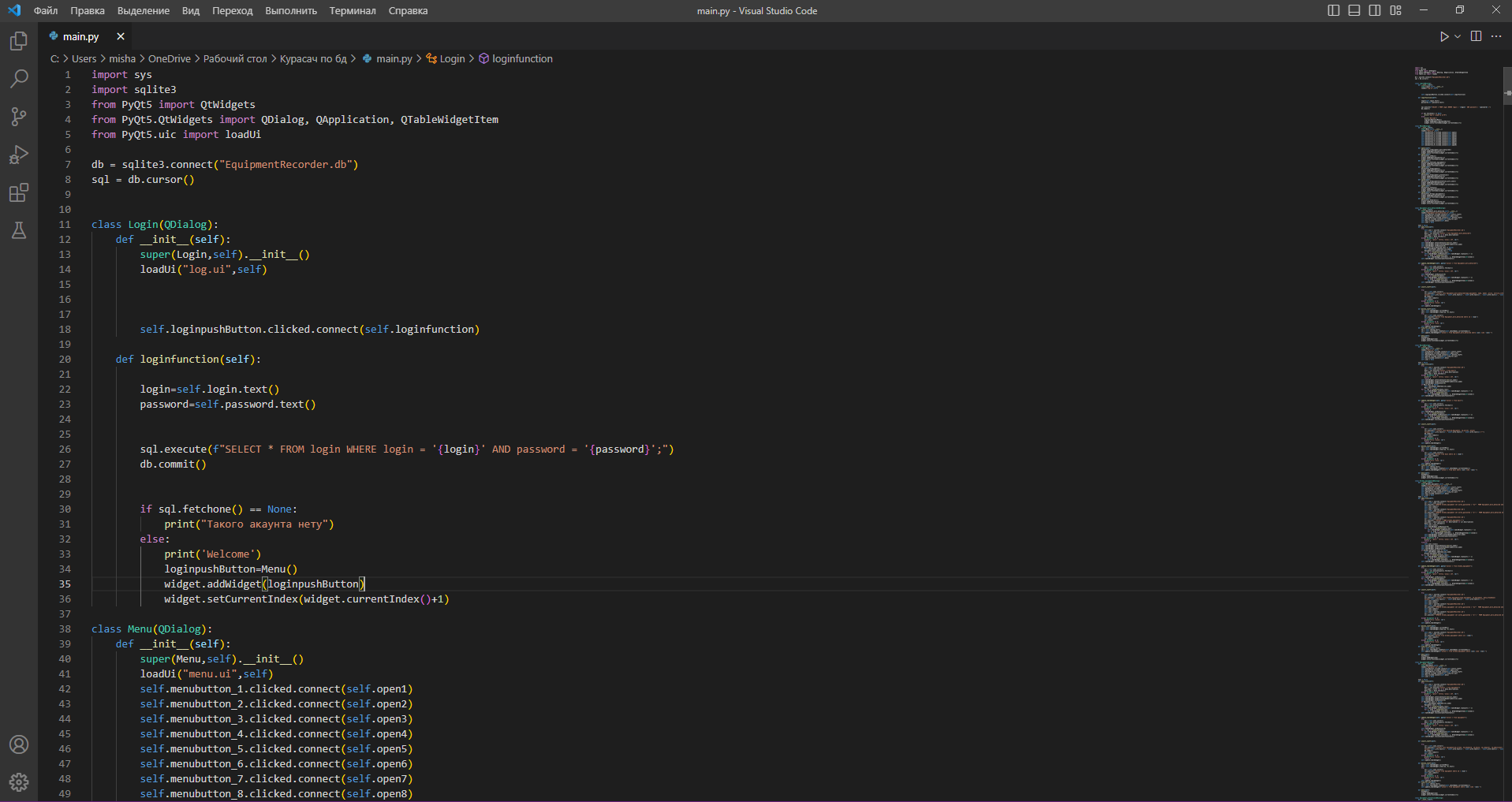
Чтобы посмотреть информацию о поломанном оборудовании нужно перейти на окно “Поломанное оборудование” и по аналогии с обычной формой взаимодействовать с таблицей, но проверка на то есть ли еще гарантия на это оборудование происходит автоматически .

**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:



#ипортирование библиотек

import sys

import sqlite3

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QDialog, QApplication, QTableWidgetItem

from PyQt5.uic import loadUi

#соеденение с базой данных

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

sql = db.cursor()

#класс в котором открывается окно авторизации и разные функции

class Login(QDialog):

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

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

#загрузка ui формы

loadUi("log.ui",self)

#подключаем кнопку к фунции

self.loginpushButton.clicked.connect(self.loginfunction)

#Проверка есть ли такие данные в базе

def loginfunction(self):

#переносим в переменную данные из строк ввода на форме

login=self.login.text()

password=self.password.text()

#сверяем данные из переменных с базой таблицей login с помощью sql запроса

sql.execute(f"SELECT \* FROM login WHERE login = '{login}' AND password = '{password}';")

#сохраняем изменения

db.commit()

#проверяем если запрос сработал то он должен что то вывести на что мы и проверяем, Если нечего не вывело то таких данных нету, если вывило то есть , так и делаем проверку есть ли такие данные

if sql.fetchone() == None:

print("Такого акаунта нету")

else:

print('Welcome')

#переход на другую форму с помощью вызова класса

loginpushButton=Menu()

widget.addWidget(loginpushButton)

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

код “Меню”

**class Menu(QDialog):**

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

**super(Menu,self).\_\_init\_\_()**

**#вызываем ui форму**

**loadUi("menu.ui",self)**

**#подключаем кнопки**

**self.menubutton\_1.clicked.connect(self.open1)**

**self.menubutton\_2.clicked.connect(self.open2)**

**self.menubutton\_3.clicked.connect(self.open3)**

**self.menubutton\_4.clicked.connect(self.open4)**

**self.menubutton\_5.clicked.connect(self.open5)**

**self.menubutton\_6.clicked.connect(self.open6)**

**self.menubutton\_7.clicked.connect(self.open7)**

**self.menubutton\_8.clicked.connect(self.open8)**

**self.menubutton\_9.clicked.connect(self.open9)**

**#функции вызова других классов**

**def open1(self):**

**menubutton\_1=Equipment\_more\_detailed()**

**widget.addWidget(menubutton\_1)**

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

**def open2(self):**

**menubutton\_2=Main()**

**widget.addWidget(menubutton\_2)**

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

**def open3(self):**

**menubutton\_3=Broken\_equipment()**

**widget.addWidget(menubutton\_3)**

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

**def open4(self):**

**menubutton\_4=Equipment()**

**widget.addWidget(menubutton\_4)**

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

**def open5(self):**

**menubutton\_5=Equipment\_prevention()**

**widget.addWidget(menubutton\_5)**

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

**def open6(self):**

**menubutton\_6=Equipmentprevention\_will\_soon()**

**widget.addWidget(menubutton\_6)**

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

**def open7(self):**

**menubutton\_7=Place()**

**widget.addWidget(menubutton\_7)**

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

**def open8(self):**

**menubutton\_8=Type\_equipments()**

**widget.addWidget(menubutton\_8)**

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

**def open9(self):**

**menubutton\_9=Workers()**

**widget.addWidget(menubutton\_9)**

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

Код стандартного класса с функциями вывода, добавления, удаление, поиска

**class Equipment\_more\_detailed(QDialog):**

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

**super(Equipment\_more\_detailed,self).\_\_init\_\_()**

**#вызываем окно**

**loadUi("Equipment\_more\_detailed.ui",self)**

#Подключаем кнопки к функциям

**self.InnerButton.clicked.connect(self.insert\_staff)**

**self.OpenButton.clicked.connect(self.open\_file)**

**self.DeleteButton.clicked.connect(self.delete\_staff)**

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

**self.hub.clicked.connect(self.menu)**

**self.conn = None**

**#переменная для проверки вызывалась ли функции вывода**

**test = False**

**def open\_file(self):**

**try:**

**#соединение с бд**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**#запрос на вывод всей базы данных**

**data = cur.execute("select \* from Equipment\_more\_detailed")**

**#установление того сколько столбцов и строк в таблице и вывод их в переменнеы**

**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**

**#задание параметров столбцов и срок для виджета вывода**

**self.tableWidget.setColumnCount(len(col\_name))**

**self.tableWidget.setHorizontalHeaderLabels(col\_name)**

**self.tableWidget.setRowCount(0)**

**#добавление колонок в выпадающий список для поиска**

**if Equipment\_more\_detailed.test == False:**

**self.cbColNames.addItems(col\_name)**

**#передача в переменную true чтобы больше не использовать эту строчку кода**

**Equipment\_more\_detailed.test = True**

**#вывод всех строк и столбцов с помощью двух циклов**

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

**self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)**

**for j, elem in enumerate(row):**

**self.tableWidget.setItem(i, j, QTableWidgetItem(str(elem)))**

**self.tableWidget.resizeColumnsToContents()**

**#функция которая записывает все изменения в бд**

**def update\_tableWidget(self, query="select \* from Equipment\_more\_detailed"):**

**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.tableWidget.setRowCount(0)**

**for i, row in enumerate(data):**

**self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)**

**for j, elem in enumerate(row):**

**self.tableWidget.setItem(i, j, QTableWidgetItem(str(elem)))**

**self.tableWidget.resizeColumnsToContents()**

**#функция которая добавляет изменения в бд**

**def insert\_staff(self):**

**try:**

**#просто запрос на добавление данных в бд и как данные используются значения из строк ввода на форме**

**cur = self.conn.cursor()**

**cur.execute(f"""insert into Equipment\_more\_detailed(type\_equipment, name, maker, price, service\_life, guarantee, start\_of\_operation)**

**values('{self.wrt1.text()}', '{self.wrt2.text()}', '{self.wrt3.text()}', '{self.wrt4.text()}', '{self.wrt5.text()}', '{self.wrt6.text()}', '{self.wrt7.text()}')""")**

**#сохранение изменений**

**db.commit()**

**self.conn.commit()**

**cur.close()**

**except Exception as e:**

**print(f"Исключение1: {e}")**

**return e**

**self.update\_tableWidget()**

**#удаления строк из таблицы**

**def delete\_staff(self):**

**#добавляем в переменные данные об выделенной строке и значение ip этой строки**

**row = self.tableWidget.currentRow()**

**num = self.tableWidget.item(row, 0).text()**

**try:**

**cur = self.conn.cursor()**

**#удаление строки с ip который мы выделили**

**cur.execute(f"delete from Equipment\_more\_detailed where id = {num}")**

**self.conn.commit()**

**cur.close()**

**except Exception as e:**

**print(f"Исключение: {e}")**

**return e**

**self.update\_tableWidget()**

**#функция поиска по значениям из поля ввода по атрибуту из выподающего списка**

**def find\_for\_val(self):**

**#заносим в переменную значение из поля ввода**

**val = self.leFind.text()**

**#заносим в переменную значение выбранного атрибута из выпадающего списка**

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

**#запрос на вывод данных в которых в выбранном атрибуте написано значение из поля ввода**

**self.update\_tableWidget(f"select \* from Equipment\_more\_detailed where {col} like '{val}'")**

**#вызов окна меню**

**def menu(self):**

**#вызов класса меню**

**hub=Menu()**

**widget.addWidget(hub)**

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

Все остальные таблицы работают по такому же принципу кроме: “оборудования с истекшим сроком эксплуатации”, “оборудования с скорым истечением срока эксплуатации”, “поломанное оборудование так как у них сильно отличается функционал”

Класс “сломанного оборудования” очень сильно похож на прошлые, но с некоторыми изменениями

**def open\_file(self):**

**#тут используется сразу несколько sql запросов из-за чего нужно после каждого запроса закрывать и заново открывать соединение с бд**

**try:**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**#запрос на обновление атрибута works\_guarantee на нет если дата покупки оборудования + гарантийный срок будет больше сегодняшней даты**

**cur.execute("""UPDATE broken\_equipment set works\_guarantee = "нет" FROM Equipment\_more\_detailed where DATETIME(start\_of\_operation, '+12 month') < CURRENT\_TIMESTAMP AND Equipment\_more\_detailed.id= broken\_equipment.id\_equipment ;""")**

**#сохранение изменений и закрытие подключения к бд, и открытие этого соединения**

**self.conn.commit()**

**self.conn.close()**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**#запрос на обновление атрибута works\_guarantee на да если дата покупки оборудования + гарантийный срок будет меньше сегодняшней даты**

**cur.execute("""UPDATE broken\_equipment set works\_guarantee = "есть" FROM Equipment\_more\_detailed where DATETIME(start\_of\_operation, '+12 month') > CURRENT\_TIMESTAMP AND Equipment\_more\_detailed.id= broken\_equipment.id\_equipment ;""")**

**#сохранение изменений и закрытие подключения к бд, и открытие этого соединения**

**self.conn.commit()**

**self.conn.close()**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**#вывод всех данных из таблицы**

**cur.execute("""SELECT \* FROM broken\_equipment;""")**

**col\_name = [description[0] for description in cur.description]**

**data\_rows = cur.fetchall()**

**self.conn.close()**

**self.tableWidget.setRowCount(0)**

**#два цикл перебора данных для нормального вывода в виджет**

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

**self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)**

**for j, elem in enumerate(row):**

**self.tableWidget.setItem(i, j, QTableWidgetItem(str(elem)))**

**self.tableWidget.setHorizontalHeaderLabels(col\_name)**

**self.tableWidget.resizeColumnsToContents()**

**except Exception as e:**

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

**return e**

**finally:**

**self.conn.close()**

**self.tableWidget.setColumnCount(len(col\_name))**

**self.tableWidget.setHorizontalHeaderLabels(col\_name)**

**self.tableWidget.setRowCount(0)**

**if Broken\_equipment.test == False:**

**self.cbColNames.addItems(col\_name)**

**Broken\_equipment.test = True**

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

**self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)**

**for j, elem in enumerate(row):**

**self.tableWidget.setItem(i, j, QTableWidgetItem(str(elem)))**

**self.tableWidget.resizeColumnsToContents()**

**def update\_tableWidget(self, query="select \* from broken\_equipment"):**

**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.tableWidget.setRowCount(0)**

**for i, row in enumerate(data):**

**self.tableWidget.setRowCount(self.tableWidget.rowCount() + 1)**

**for j, elem in enumerate(row):**

**self.tableWidget.setItem(i, j, QTableWidgetItem(str(elem)))**

**self.tableWidget.resizeColumnsToContents()**

**#тут используется сразу несколько sql запросов из-за чего нужно после каждого запроса закрывать и заново открывать соединение с бд**

**def insert\_staff(self):**

**try:**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**cur.execute(f"""insert into broken\_equipment(type\_equipment, id\_equipment, data\_breakdown)**

**values('{self.wrt1.text()}', '{self.wrt2.text()}', '{self.wrt3.text()}')""")**

**self.conn.commit()**

**self.conn.close()**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**cur.execute("""UPDATE broken\_equipment set works\_guarantee = "нет" FROM Equipment\_more\_detailed where DATETIME(start\_of\_operation, '+12 month') < CURRENT\_TIMESTAMP AND Equipment\_more\_detailed.id= broken\_equipment.id\_equipment ;""")**

**self.conn.commit()**

**self.conn.close()**

**self.conn = sqlite3.connect('EquipmentRecorder.db')**

**cur = self.conn.cursor()**

**cur.execute("""UPDATE broken\_equipment set works\_guarantee = "есть" FROM Equipment\_more\_detailed where DATETIME(start\_of\_operation, '+12 month') > CURRENT\_TIMESTAMP AND Equipment\_more\_detailed.id= broken\_equipment.id\_equipment ;""")**

**except Exception as e:**

**print(f"Исключение1: {e}")**

**return e**

**self.update\_tableWidget()**

Главные различия в том что тут сразу несколько запросов

Код “таблиц оборудования с истекшим сроком эксплуатации”, “оборудования с скорым истечением срока эксплуатации” схож между собой.

код оборудования с истекшим сроком эксплуатации

class Equipment\_prevention(QDialog):

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

super(Equipment\_prevention,self).\_\_init\_\_()

loadUi("equipment\_prevention.ui",self)

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

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

self.hub.clicked.connect(self.menu)

self.conn = None

**#переменная для проверки вызывалась ли функции вывода**

test = False

def open\_file(self):

#тут тоже используется сразу несколько запросов, но полностью отсутствуют функции добавление,удаления данных так как это делается автоматически

try:

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

cur = self.conn.cursor()

cur.execute("DELETE FROM equipment\_prevention;")

self.conn.commit()

self.conn.close()

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

cur = self.conn.cursor()

cur.execute("INSERT INTO equipment\_prevention( type\_equipment ,id\_equipment,start\_of\_operation,service\_life) SELECT type\_equipment, id, start\_of\_operation, service\_life from Equipment\_more\_detailed where DATETIME(start\_of\_operation, '+12 month') < CURRENT\_TIMESTAMP;")

self.conn.commit()

self.conn.close()

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

cur = self.conn.cursor()

cur.execute("""SELECT \* FROM equipment\_prevention;""")

col\_name = [description[0] for description in cur.description]

data\_rows = cur.fetchall()

self.conn.close()

self.tableWidget.setRowCount(0)

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

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

for j, elem in enumerate(row):

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

self.tableWidget.setHorizontalHeaderLabels(col\_name)

self.tableWidget.resizeColumnsToContents()

except Exception as e:

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

return e

finally:

self.conn.close()

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

self.tableWidget.setHorizontalHeaderLabels(col\_name)

self.tableWidget.setRowCount(0)

if Equipment\_prevention.test == False:

self.cbColNames.addItems(col\_name)

Equipment\_prevention.test = True

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

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

for j, elem in enumerate(row):

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

self.tableWidget.resizeColumnsToContents()

def update\_tableWidget(self, query="select \* from equipment\_prevention"):

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.tableWidget.setRowCount(0)

for i, row in enumerate(data):

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

for j, elem in enumerate(row):

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

self.tableWidget.resizeColumnsToContents()

def find\_for\_val(self):

val = self.leFind.text()

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

self.update\_tableWidget(f"select \* from equipment\_prevention where {col} like '{val}'")

#вызов меню

def menu(self):

self.conn.close

hub=Menu()

widget.addWidget(hub)

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

код “оборудования с скорым истечением срока эксплуатации отличается лишь запросами”

def open\_file(self):

try:

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

cur = self.conn.cursor()

cur.execute("DELETE FROM equipmentprevention\_will\_soon;")

self.conn.commit()

self.conn.close()

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

cur = self.conn.cursor()

cur.execute("INSERT INTO equipmentprevention\_will\_soon(type\_equipment, id\_equipment, start\_of\_operation, service\_life, Days\_end\_life) SELECT type\_equipment, id, start\_of\_operation, service\_life, ROUND(julianday(DATETIME(start\_of\_operation, '+12 month')) - julianday('now')) FROM Equipment\_more\_detailed WHERE DATETIME(start\_of\_operation, '+12 month') > CURRENT\_TIMESTAMP;")

self.conn.commit()

self.conn.close()

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

cur = self.conn.cursor()

cur.execute("""SELECT \* FROM equipmentprevention\_will\_soon;""")

а всё остальное такое же как и в “оборудования с истекшим сроком эксплуатации”

**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 я разработал, была создана база данных, которая была заполнена данными, также был разработан интерес непосредственно под саму базу данных, был создан файл main.py, в где мы при помощи языка python связали формы и базу данных, назначили действия на кнопки и поля ввода, расположенных на форме.