

Колледж космического машиностроения и технологий

**ОТЧЕТ**

**по производственной практике ПП.01.01 по модулю ПМ.01**

**«Разработка программных модулей программного обеспечения**

**для компьютерных систем»**

**по специальности 09.02.03 «Программирование в компьютерных системах»**

Выполнил студент

Группа П1-18

Волков М.А.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Принял преподаватель

Гусятинер Л.Б.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(Оценка)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Подпись)

**Королев, 2021**

**Оглавление**

[**Введение 3**](#_Toc75754906)

[**1. Общие сведения об организации 4**](#_Toc75754907)

[1.1. Организационная структура и управление деятельностью ККМТ. 4](#_Toc75754908)

[1.2. Организационная структура и управление архивного отдела. 6](#_Toc75754909)

[1.3. Основные задачи архивного отдела. 7](#_Toc75754910)

[1.4. Функции архивного отдела. 8](#_Toc75754911)

[1.5. Нормативные документы, которые регламентируют профессиональную деятельность. 9](#_Toc75754912)

[1.6. Программное обеспечение предприятия. 9](#_Toc75754913)

[1.7. Задачи, подлежащие автоматизации. 9](#_Toc75754914)

[**2. Содержание выполняемых видов работ 10**](#_Toc75754915)

[2.1. Разработка спецификаций отдельных компонентов. 10](#_Toc75754916)

[**2.2. Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы HelperDoc 22**](#_Toc75754917)

[2.3. Отладка программного модуля с использованием специализированных программных средств 24](#_Toc75754918)

[**3. Выводы 26**](#_Toc75754919)

[**4. Дневник практики 27**](#_Toc75754920)

[**4. Список использованной литературы 31**](#_Toc75754921)

[**5. Приложения 32**](#_Toc75754922)

# **Введение**

На 3 курсе обучения в ККМТ, студентом группы П1-18 Волковым Максимом Александровичем была пройдена производственная практика по модулю ПМ.01 «Участие в интеграции программных модулей» в архивном отделе Колледжа Космического Машиностроения и Технологий им. А. А. Леонова. Студент получил задание разработать программу, которая создает документ и меняется в зависимости от данных студентов.

# **Общие сведения об организации**

## **Организационная структура и управление деятельностью ККМТ.**

* KKMT является составной частью организационной структуры Университета.
* Структура и штат ККМТ, a также изменения к ним утверждаются приказом ректора «Технологического университета» по представлению директора ККМТ, согласованному с проректором по учебно-методической работе в установленном порядке.
* KKMT имеет следующее структурное деление:

-отделения по специальностям;

-предметные и цикловые комиссии;

-другие учебные подразделения (лаборатории, мастерские, кабинеты), обеспечивающие образовательный процесс.

* Непосредственное управление деятельностью ККМТ осуществляет директор, назначаемый приказом ректора «Технологического университета», имеющий высшее образование и стаж работы не менее 5 лет на педагогических и руководящих должностях в системе профессиональном образования.
* Полномочия директора ККМТ определяются настоящим Положением, трудовым договором и должностной инструкцией, и другими локальными актами «Технологического университета».
* Заместители директора ККМТ;

-заместитель директора по учебной работе;

-заместитель директора по учебно-методической работе;

-заместитель директора по учебно-производственной работе;

-заместитель директора по учебно-воспитательной работе;

-заместитель директора по научно-методической работе.

* Координацию деятельности ККМТ осуществляет первый проректор и проректор по учебно-методической работе.
* C цепью объединения усилий руководства ККМТ, педагогических работников, учебно-вспомогательного персонала ККМТ для обеспечения качества подготовки специалистов создается Педагогический совет ККМТ.
* B состав Педагогического совета ККМТ входят по должности директор ККМТ, его заместители и руководители структурных подразделений ККМТ. В состав Педагогического совета ККМТ могут также входить сотрудники структурных подразделений «Технологического университета», организаций и учреждений, непосредственно связанных с деятельностью ККМТ по подготовке специалистов.
* Председателем Педагогического совета ККМТ является директор ККМТ.
* К компетенциям Педагогического совета KKMT относятся:

-разработка и предоставление Ученому совету «Технологического университета» на утверждение стратегии развития ККМТ;

-решение вопросов учебной, методической и воспитательной работы, вопросов взаимодействия с институтами «Технологического университета» по обеспечению функционирования системы непрерывного многоуровневого профессиональные образования;

-формирование предложений по контрольным цифрам приема студентов в ККМТ, рассмотрение планов работы приемной комиссии, отчетов по итогам приема и проведению профориентационной кампании;

-ежегодные заслушивание отчета директора ККМТ об основных направлениях деятельности;

-рассмотрение вопросов выдвижения кандидатур из состава работников ККМТ на соискание премий, присвоения почетных званий:

-рассмотрение других вопросов, связанных с деятельностью ККМТ, не относящихся к исключительной компетенции Ученого совета «Технологического университета».

Схема организационной структуры ККМТ на рисунке 1.

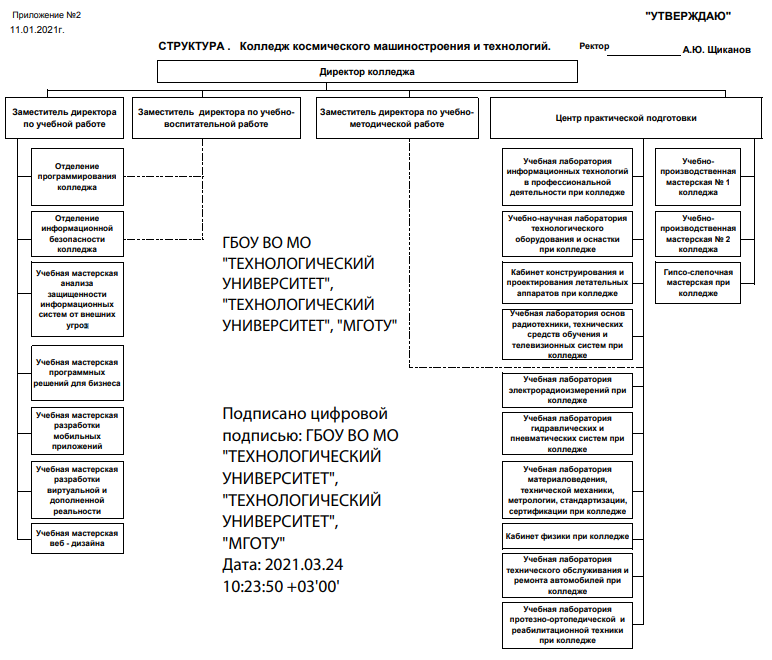


Рис.1 Организационная структура ККМТ

## **Организационная структура и управление архивного отдела.**

* Руководство архивным отделом осуществляет начальник отдела.
* Структура, численность и штат работников архивного отдела, а также их изменения утверждаются приказом ректора Университета по представлению начальника центра кадровой политики и учета персонала по согласованию с курирующим проректором.
* Должностные обязанности работников архивного отдела, включая начальника отдела, определяются должностными инструкциями, утверждаемыми ректором Университета.
* В период временного отсутствия (болезнь, отпуск, командировка) начальника отдела его обязанности исполняет лицо, назначенное приказом ректора Университета, которое несет ответственность за надлежащее их исполнение.

## **Основные задачи архивного отдела.**

Архивный отдел создан для организации, руководства, координации, контроля и совершенствования работы по обеспечению системы приема, учета и хранения документов в Университете. Исходя из поставленных целей, архивный отдел решает следующие задачи:

* Комплектование документами, состав которых предусмотрен разделом 3 настоящего Положения.
* Учет, обеспечение сохранности, создание научно-справочного аппарата, использование документов, хранящихся в архивном отделе.
* Организационно-методическое руководство, координация и контроль деятельности структурных подразделений по вопросам организации архивного дела.
* Подготовка и своевременная передача документов на государственное хранение с соблюдением требований, устанавливаемых государственной архивной службой.
* Внедрение новых методов организации работы, в том числе на основе использования современных информационных технологий.

## **Функции архивного отдела.**

В соответствии с возложенными задачами архивный отдел осуществляет следующие функции:

* Планирует, принимает, учитывает и хранит документы структурных подразделений Университета, обработанные в соответствии с требованиями нормативных документов.
* Разрабатывает и внедряет современные унифицированные формы документов, утвержденные нормативными правовыми актами Университета.
* Осуществляет учет и обеспечивает полную сохранность принятых на хранение дел, соблюдение в помещениях архивного отдела условий, необходимых для обеспечения сохранности документов.
* Создает, пополняет и совершенствует научно-справочный аппарат к хранящимся в архивном отделе делам и документам.
* Обеспечивает использование документов:

-информирует курирующего проректора, начальника центра кадровой политики и учета персонала и работников Университета о составе и содержании документов архивного отдела; - выдает в установленном порядке дела, документы или копии документов в целях служебного и научного использования;

-исполняет запросы организаций и заявления граждан об установлении трудового стажа и другим вопросам социально-правового характера, в установленном порядке выдает копии документов и архивные справки;

-ведет учет использования документов, хранящихся в архивном отделе.

* Проводит экспертизу ценности документов, хранящихся в архивном отделе, организует и принимает участие в работе Экспертной комиссии Университета.
* Участвует в работе по составлению номенклатуры дел Университета, оказывает методическую и практическую помощь структурным подразделениям в ее составлении, совместно с общим отделом готовит проект сводной номенклатуры дел Университета.
* Контролирует правильность формирования и оформления дел в делопроизводстве, а также подготовку дел к передаче в архивный отдел.

## **Нормативные документы, которые регламентируют профессиональную деятельность.**

* Номенклатура дел института.
* Положение об институте
* Должностные инструкции персонала института
* Документы по планированию деятельности института техника и цифровых технологий.
* Перечень нормативной и технической документации.

## **Программное обеспечение предприятия.**

* Excel 2013
* Windows 7 Ultimate
* Access 2013
* DB Browser for SQLite Version 3.12.1
* Visual Studio Community 2016
* JetBrains PyCharm Community Edition 2019.2.1

## **Задачи, подлежащие автоматизации.**

* Улучшения интерфейса и реализация ее простоты.
* Создание .exe файла для запуска программы.

# 

# **Содержание выполняемых видов работ**

## **Разработка спецификаций отдельных компонентов.**

При прохождении производственной практики было получено задание на разработку программы, которая строит шаблон Word файла и вставляет данные студентов, из таблицы. Таблицу можно пополнять, изменять и удалять.

**Структура данных HelperDoc.**

Логическая структура базы данных:

document (id, title, type, series, number, loss\_confirmation, exchange\_confirmation, destruction\_confirmation, education\_level, issue\_date, registration\_number, status, student\_id).

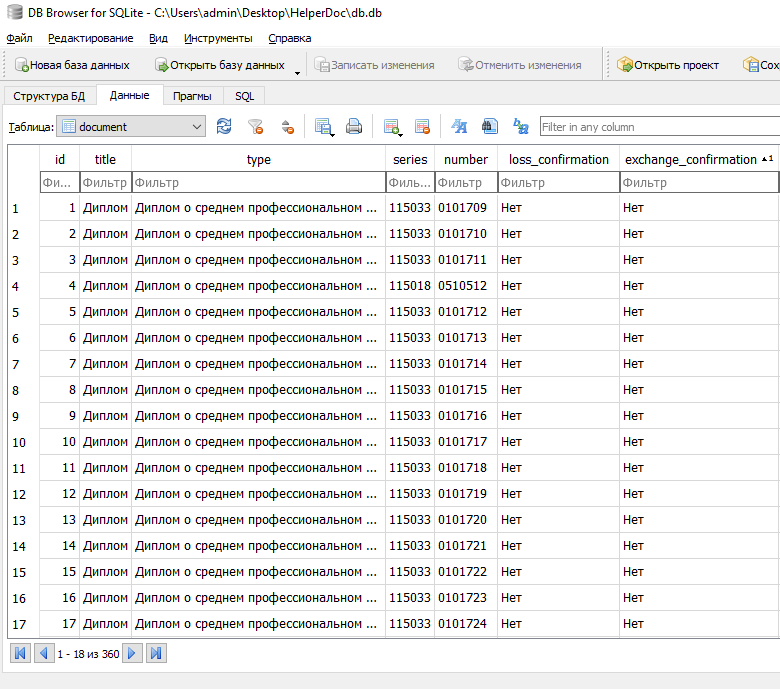


Рис.1 Пример хранения таблицы “document” (1 часть)

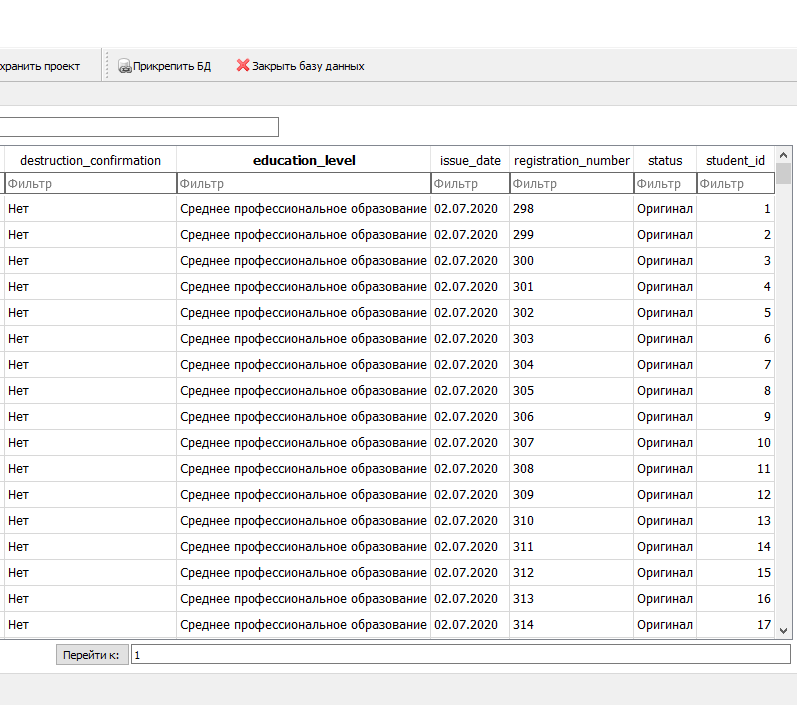


Рис.2 Пример хранения таблицы “document”(2 часть)

education\_program (id, title).

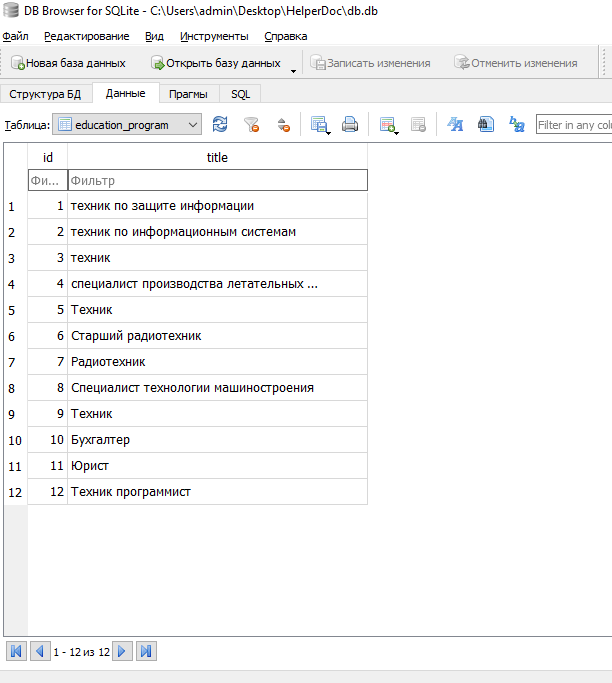


Рис.3 Пример хранения таблицы “education\_program”

profession(code, title).

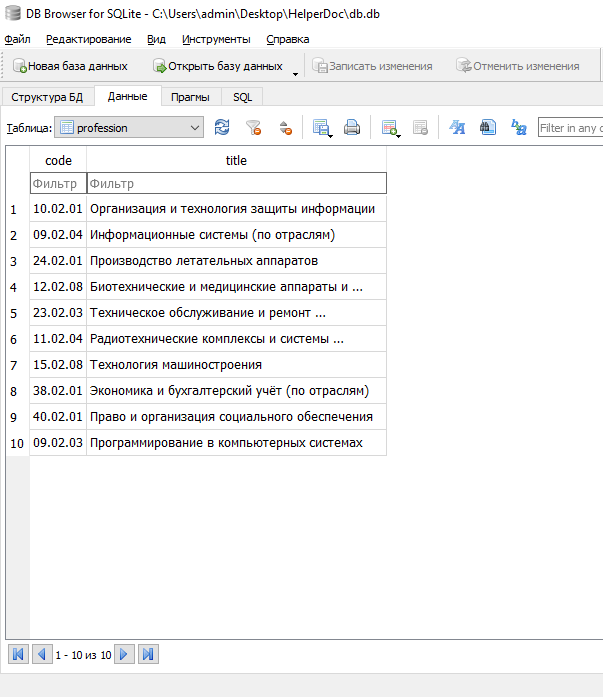
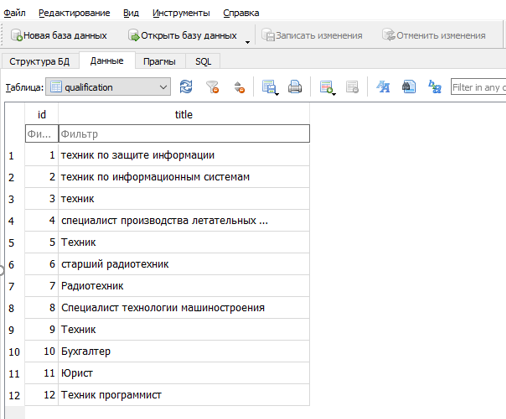


Рис.4 Пример хранения таблицы “profession”

qualification(id, title).

  
Рис.5 Пример хранения таблицы “qualification”

sqlite\_sequence(name, seq).

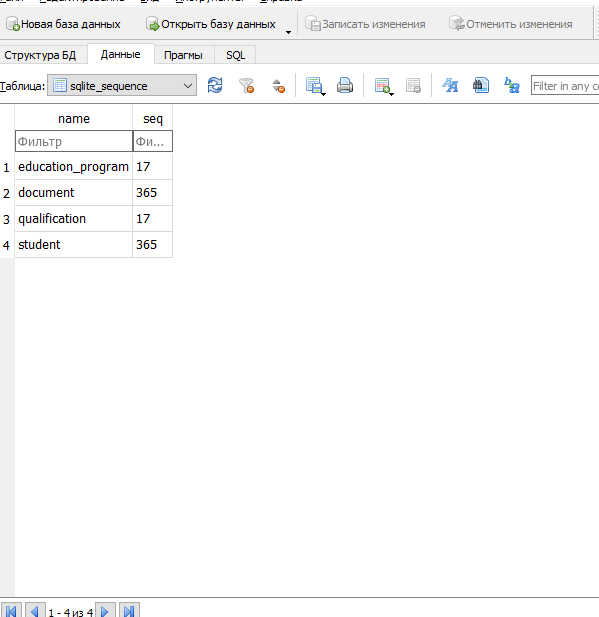


Рис.6 Пример хранения таблицы “sqlite\_sequence”

Student(id, name, last\_name, middle\_name, birth\_date, gender, snills, county\_code, education\_form, education\_receipt\_form, admission\_year, granduation\_year, study\_period, profession\_code, qualification\_code, qualification\_id, edication\_program\_id).

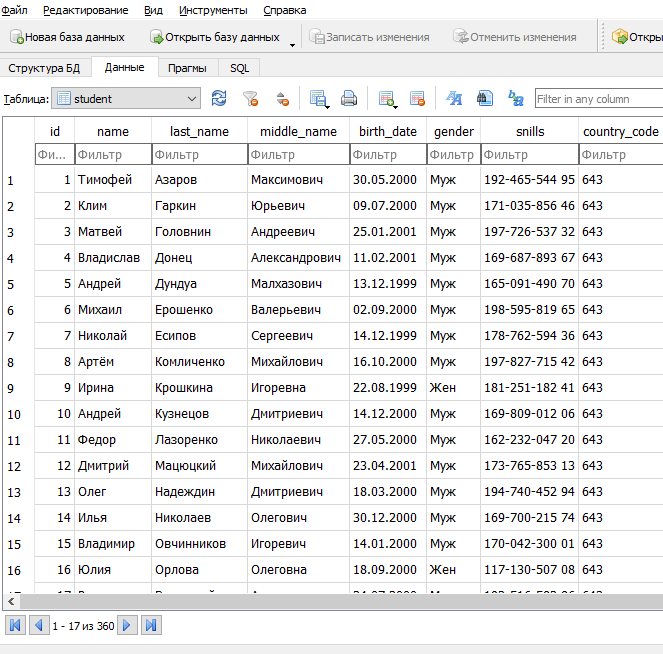


Рис.7 Пример хранения таблицы “student”(1 часть)

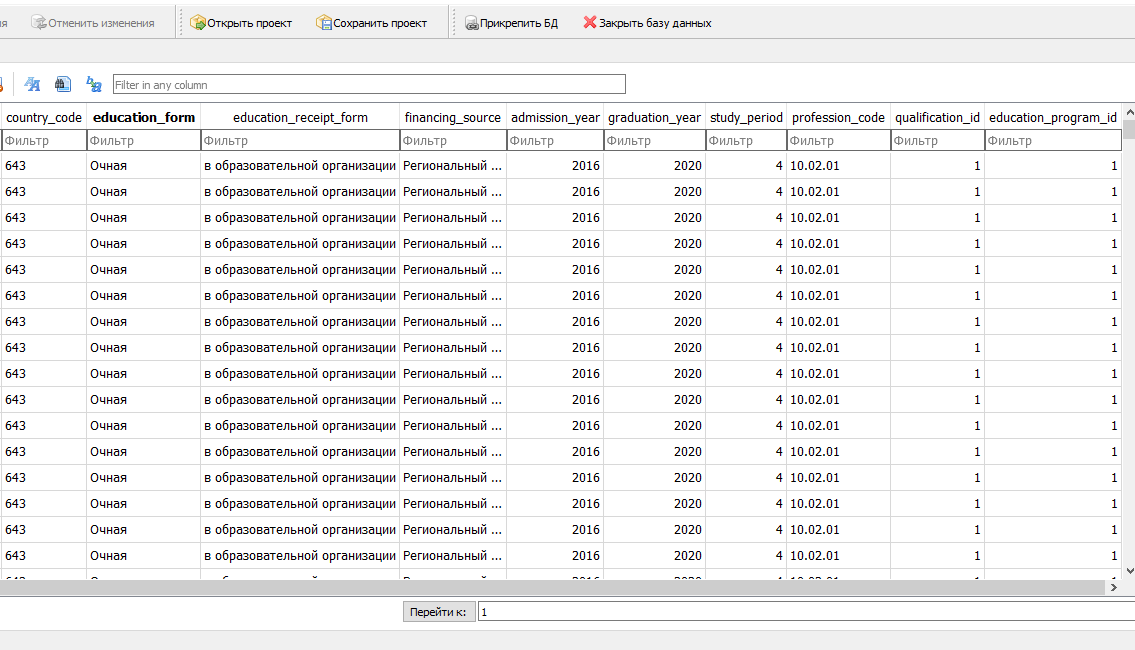


Рис.8 Пример хранения таблицы “student”(2 часть)

Для проектирования базы данных был выбран SQLite. Для хранения данных используются таблицы. База данных включает несколько взаимосвязанных таблиц.

**Структура данных ListOfStudent.**

Логическая структура базы данных:

StudentList(ID, FirstName, LastName, MiddleName, EducationForm, Snills, BirthDate)

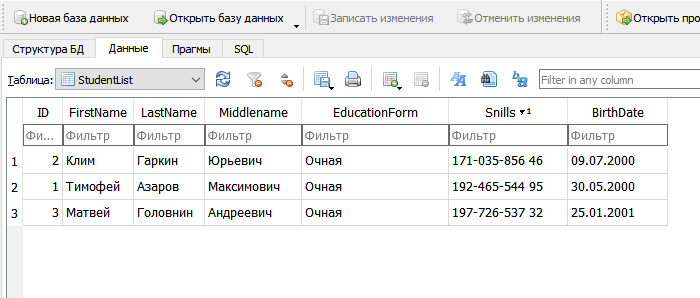


Рис.9 Пример хранения таблицы “StudentList”

sqlite\_sequence(name, seq)

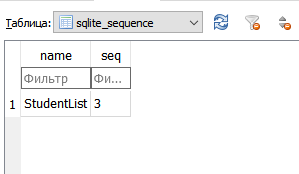


Рис.10 Пример хранения таблицы “sqlite\_sequence”

Для проектирования базы данных был выбран SQLite. Для хранения данных используются таблицы.

**Структура программы HelperDoc**

Программа имеет главную форму, цель которой служит для отображения информации из таблицы СУБД SQLite и для различных манипуляцией с таблицей СУБД SQLite. Есть специальное выделенное место, в котором заполняются данные студентов. Надо заполнять данные с соответствием названием колонки. В главной форме присутствует поиск, который ищет написанные в специальную строку данные. Есть 4 кнопки: Удаление, Создать документ, Очистить и Добавить. Кнопка “Удаление” служит для удаления ячейки данных. Кнопка “Создать документ” создает документ по выбранной строке из таблицы. Кнопка “Очистить” отчищает поля, которые хотели добавить в таблицу. Кнопка “Добавить” служит для добавления данных из поля в таблицу.

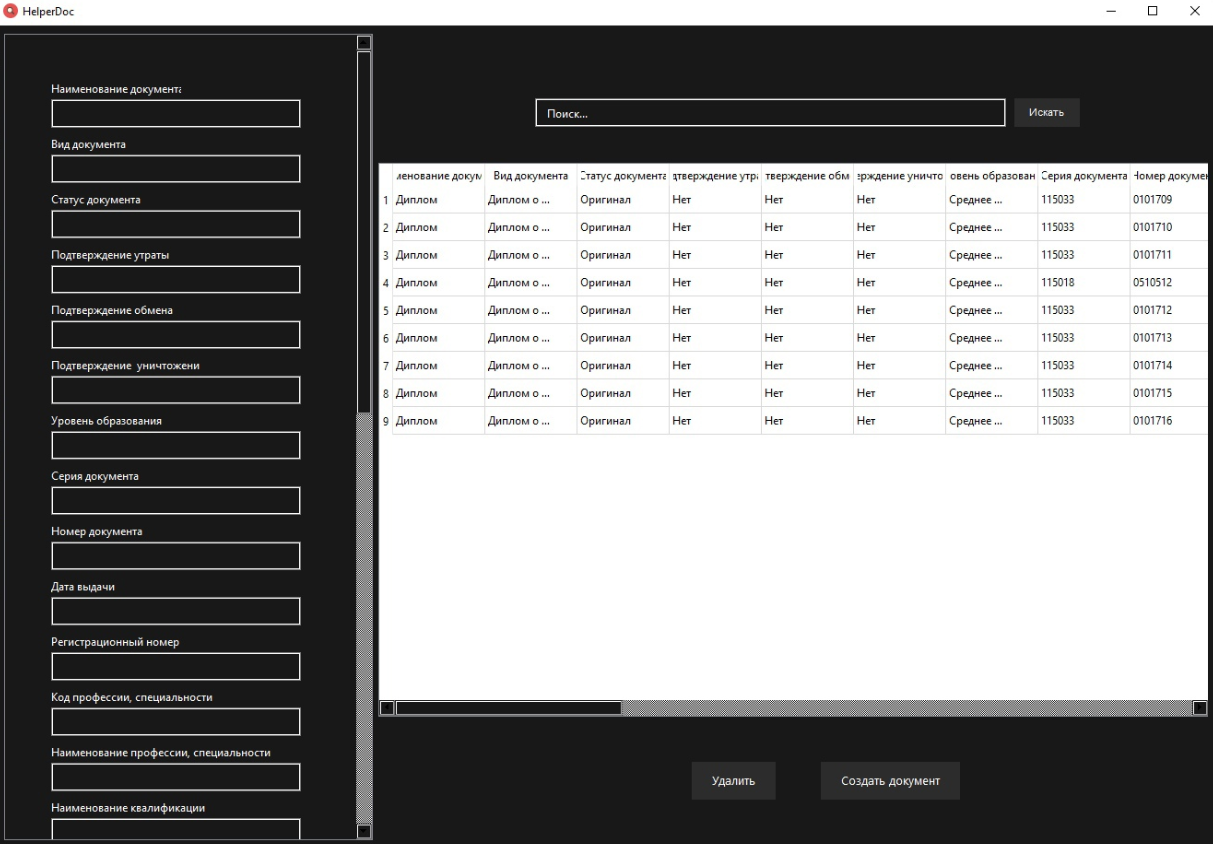


Рис.11 Главная форма c поле ввода HelperDoc

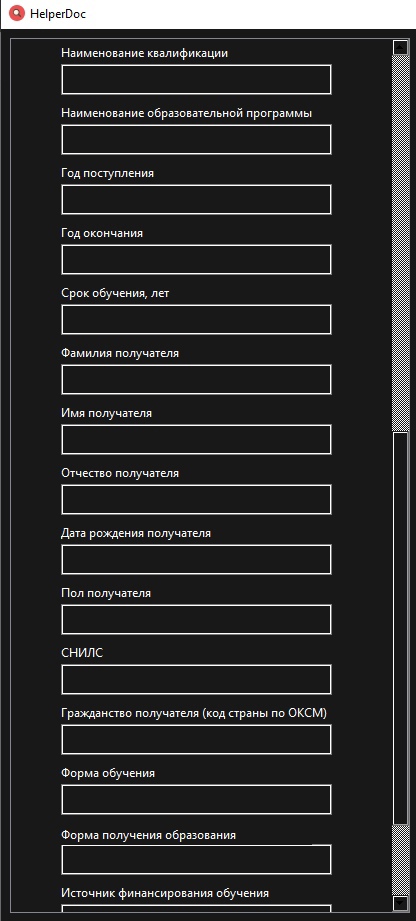


Рис.12 Поле ввода HelperDOC (1 часть)

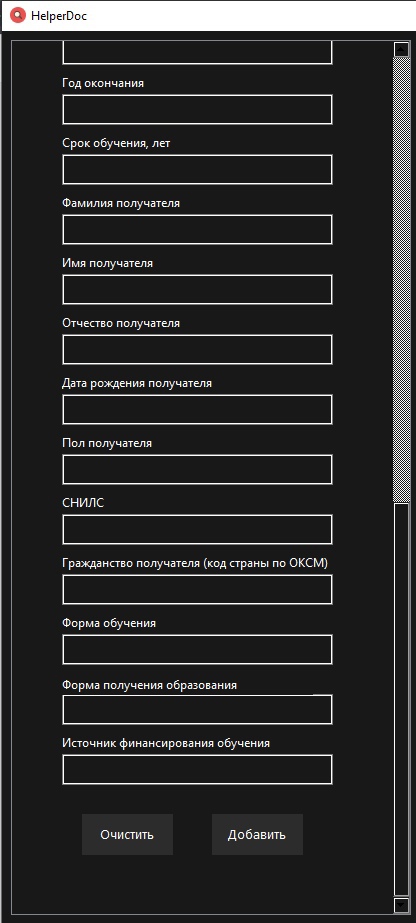


Рис.13 Поле ввода HelperDoc (2 часть)

**Структура программы ListOfStudents**

Программа имеет главную форму, цель которой служит для отображения информации из таблицы СУБД SQLite и для различных манипуляцией с таблицей СУБД SQLite. Есть меню, в котором заполняются данные студентов. Надо заполнять данные с соответствием названием ярлыка. Есть 4 кнопки, которые отвечают за добавление данных из таблицы, удаление данных из таблицы, изменение данных таблицы и выход из программы.

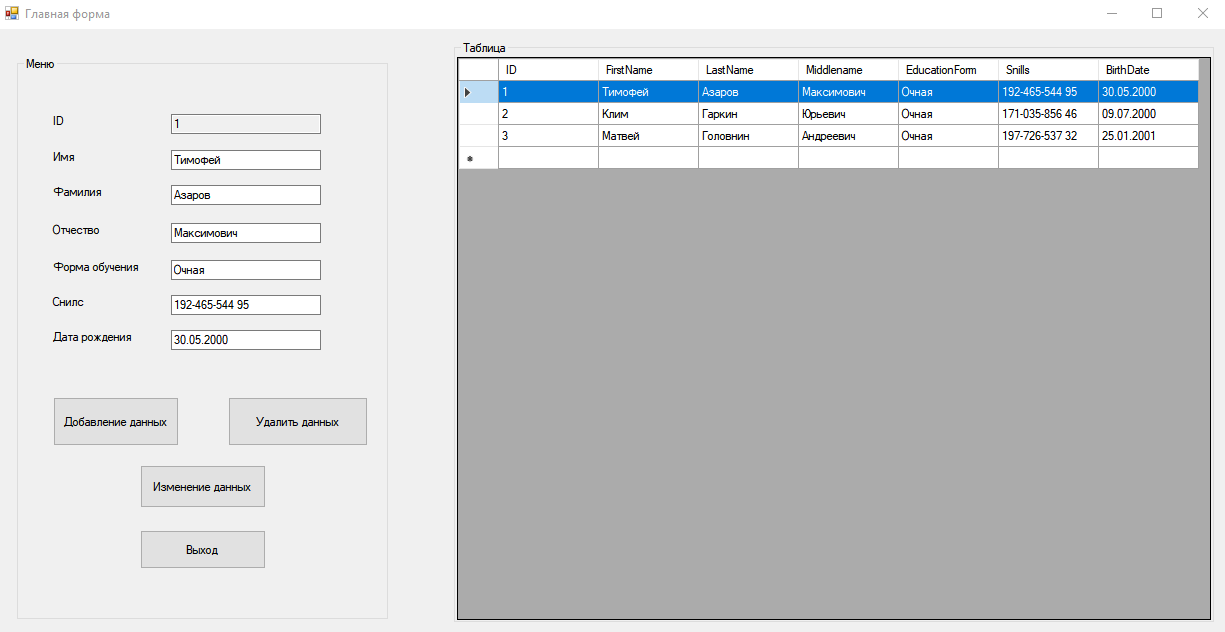
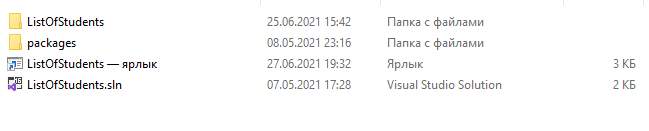


Рис.14 Главная форма ListOfStudent

**Процесс запуска программы ListOfStudents**

Для начала нужно скачать саму программу. Когда скачалась программа, будет папка ListOfStudents, открываем ее и перед нами предстает 2 папки, специальное .sln расширение для открытия проекта и ярлык к .exe.

Рис.15 Демонстрация файлов ListOfStudent

Чтобы посмотреть проект, нужно нажать на файл с расширением .sln, если нужно запустить настольное приложение, то нужно нажать на ярлык.

Процесс запуска программы HelperDoc

Для начала нужно скачать саму программу. Когда скачалась программа, будет папка HelperDoc, открываем ее и перед нами представляется несколько файлов, чтобы запустить программу, нужно нажать на main.py с помощью Shift + правая кнопка мыши и после нажать копировать как путь.

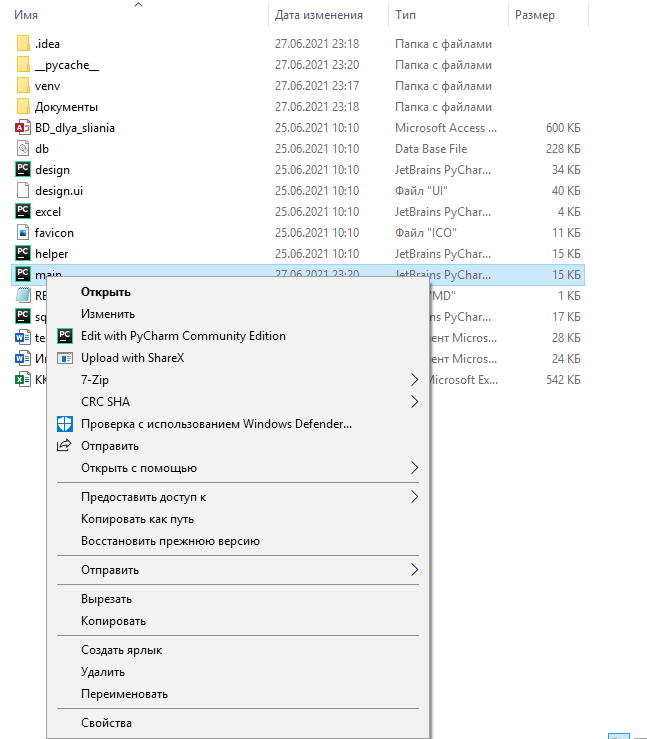


Рис.16 Демонстрация файлов HelpDoc

После нужно зайти в командную строку, нажимаем win + R и перед нам предстает окно, вводим туда cmd. Нам предстает строка, вводим cd, нажимаем пробел и ctrl+v, там выводится "C:\Users\admin\Desktop\HelperDoc\main.py", удаляем на конце main.py, чтобы сделать путь к папке. После пишем python main.py и программа запускается.

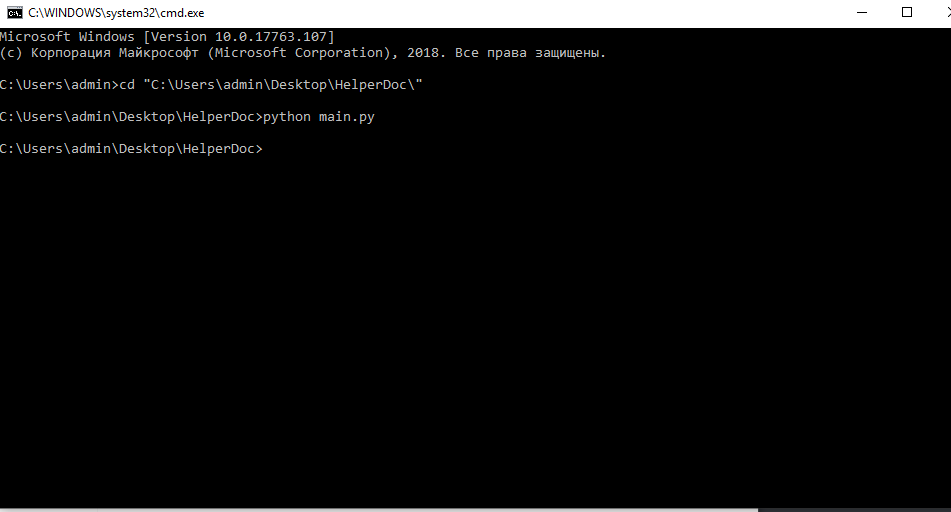


Рис.17 Демонстрация командной строки.

Чтобы посмотреть проект, нужно открыть файлы с расширением .py

### **2.2. Разработка кода программного продукта на основе готовых спецификаций на уровне модуля для программы HelperDoc**

Модуль для загрузки данных в визуальную таблицу в Листинге 2. Данный модуль загружает данные из базы данных в визуальную Таблицу.

**Листинг 1. Модуль загрузки данных.**

def load\_data(self):

"""

Загружает все данные из бд в GUI

"""

all\_id = self.get\_all\_id()

table\_row = 0

self.ui.table.setRowCount(len(all\_id))

for pk in all\_id:

data = self.helper.create\_row(pk)

for i in range(self.ui.table.columnCount()):

self.ui.table.setItem(table\_row, i, QtWidgets.QTableWidgetItem(str(data[i])))

table\_row += 1

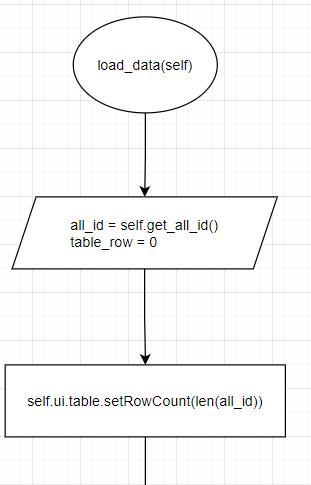


Рис. 18. Блок-схема листинга 1. Часть 1

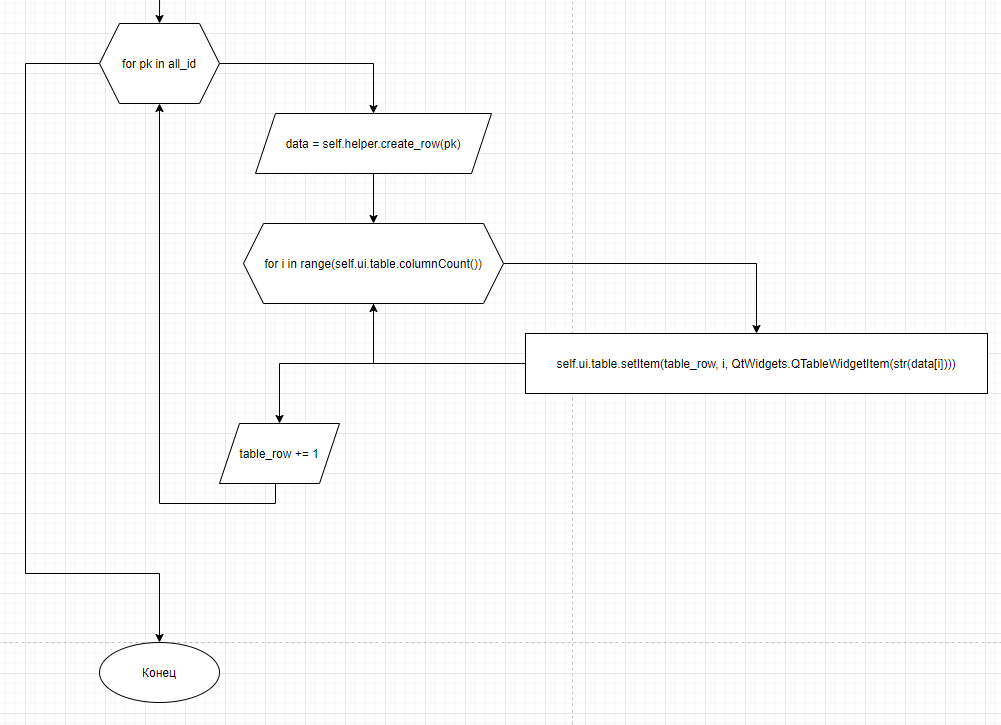


Рис. 19. Блок-схема листинга 1. Часть 2

## **2.3. Отладка программного модуля с использованием специализированных программных средств**

**Отладка HelperDoc**

Отладка проводилась с помощью Дебагера в Pycharm.

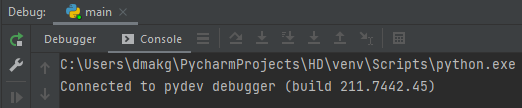
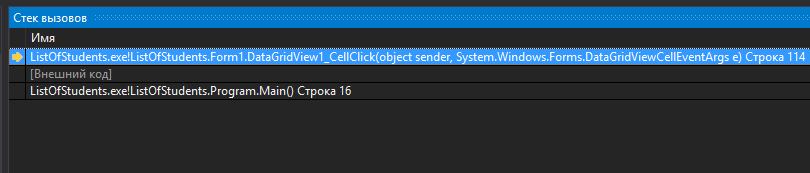


Рис. 20. Успешная сборка проекта HelperDoc

**Отладка ListOfStudents**

Рис. 22 Стек вызовов ListOfStudents

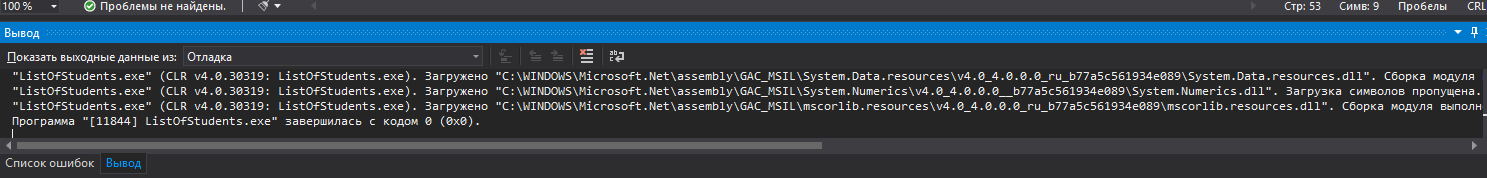


Рис. 21. Успешная сборка проекта ListOfStudents

# **Выводы**

Полученные навыки:

* Работа в Access
* Работа с базами данными в C# через SQLite
* Работа с Git

Полученные умения:

* Создание макросов на VBA
* Разработка программ в Visual Studio на языке C#
* Разработка программ в Visual Studio на языке VB.NET

# **4. Дневник практики**

**Записи выполненных работах.**

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Краткое содержание работ | Отметка о выполнении работы | Подпись руководителя практики |
| 30.04 | Дан образец заполнения таблицы и Word документ, с которым нужно работать. Создание планировалось на C# Windows Forms. Сделал Grid, в качестве СУБД, был выбран SQLite для работы с таблицей. Подключен SQLite к C#. |  |  |
| 03.05 | Добавлена кнопка добавление данных. Вписаны данные для тестировки работы с таблицей. Появились баги, не добавлялись данные при нажатии кнопки. |  |  |
| 04.05 | Фикс бага с добавлением данных, планирование добавить кнопки прокрутки таблицы, удаления и изменения данных. |  |  |
| 05.05 | Добавление кнопки удаления и изменения данных в C# Windows Forms для слияния. Добавлены кнопки прокрутки таблицы по строке. |  |  |
| 06.05 | Изучал тему дистанционного обучения, принцип работы Git. Работа с курсовым проектом: сделал описание предметной области. |  |  |
| 07.05 | Искал схожие программы для курсового проекта, изучал тему дистанционного обучения, принцип работы Git. Работа с курсовым проектом: Сделана вся теоретическая часть, дорабатывая документ. |  |  |
| 10.05 | Работа с курсовым проектом. Сделана вся эксплуатационная часть. Написал заключение. |  |  |
| 11.05 | Работа с курсовым проектом: добавлено описание модулей, описание тестовых наборов модулей. Сделана диаграмма классов, проектирование сценария описал выбор инструментов и составлена диаграмма прецедентов. |  |  |
| 12.05 | Изучал Lazarus, Free Pascal и SQLite. Работа с курсовым проектом: проведена проверка имя группы, заполнения вида репозитория и описал применение средств отладки. Сделана вся проектная часть. |  |  |
| 13.05 | Вместо программы на C#, была выбран Access в качестве БД для слияния Word. |  |  |
| 14.05 | Сделана колонка для родительных падежей для слияния Word, но для этого нужно самому вписывать ФИО в родительном падеже. |  |  |
| 17.05 | Проведение проверки слияния Word с помощью Access таблицы проведена успешно, кроме дат. |  |  |
| 18.05 | В качестве таблицы думал над вариантом сделать таблицу не в Access, в Excel, из-за того, что в слиянии Word дата и время разделялись “\”, а нужно “.”, в итоге в Excel такая же проблема и снова выбор выпал на Access. Заполнял таблицу Excel для слияния Word. |  |  |
| 19.05 | Поиск автоматизации родительного падежа в слиянии Word, найдены макросы на VBA, которые не работают. |  |  |
| 20.05 | Найдена специальная библиотека padeg.dll для автоматического вписывания в слияние Word. Попытка неудачная, padeg.dll имеет 32 bit, запускается только через командную консоль. Находил только платную padeg.dll для 64 bit. |  |  |
| 21.05 | Завершение курсового проекта по теме: Приложение “Таск-анализатор репозиториев”. Клиентная часть. |  |  |
| 24.05 | Неудачная реализация родительного падежа через VBA макрос, чтобы автоматически вписывались в слияние Word, т.к. макросы не используются для слияния Word. |  |  |
| 25.05 | Разбор билета КТП УП.01.01 18-20. Как БД использовался Access, в качестве вывода использовался C#. |  |  |
| 26.05 | Поиск информации о маске ввода в Word: Замена “\” на “.” в дате. Замена символов в слиянии Word. Замена маски ввода Access в типа данных “Дата и время”. |  |  |
| 27.05 | Удаление колонок: ФИО (в родительном падеже). Добавление колонок: Форма обучения. Поправка данных в слиянии Word. |  |  |
| 31.05 | Добавление колонок: Приложение к диплому, регистрационный номер, примечание. Поправка данных в слиянии Word. |  |  |
| 01.06 | Добавление колонок: Дата приказа о зачислении, Номер приказа о зачислении, Дата приказа об окончании, Номер приказа об окончании. Поправка данных в слиянии Word. Удаление колонок: Приказа о зачислении, Приказа об окончании. Заново делал таблицу для сброса счетчика. Удалял свои данные для примера. |  |  |
| 02.06 | Посмотрел совместимость слияния Word вместе с таблицами Excel, которые были сегодня даны. |  |  |
| 03.06 – 11.06 | Изучение языка Visual Basic на платформе .NET. |  |  |
| 14.06  ­­–  16.06 | Изучил компиляцию в C#; CIL(Common Intermediate Language); CLR(Common Language Runtime); JIT(Just-In-Time)-компилятор; CTS(Common Type System); CLS (Common Language Specification). |  |  |
| 17.06 | Изучил назначение сборщика мусора .NET, управление свободным списком . |  |  |
| 18.06 | Изучил сборщик мусора .NET на основе трассировки. |  |  |
| 21.06 | Изучил как приостанавливать потоки для каждой фазы. |  |  |
| 22.06 – 28.06 | Оформление отчета по производственной практике. |  |  |

1. **Список использованной литературы**
2. Изучение VB.NET и C#. <https://metanit.com/>
3. Изучение C# и .NET структуры. <https://professorweb.ru>
4. Изучение программирования на C# и изучение Git. <https://stepik.org>

# **Приложения**

**Приложение 1**



**Рис. 15 Организационная структура предприятия.**

**Код программы «ListOfStudents»**

**Приложение 2.**

**Листинг 2. Form1.cs**

using System;

using System.Data;

using System.Windows.Forms;

using System.Data.SQLite;

namespace ListOfStudents

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private static SQLiteConnection sql\_con; // Подключаем базу данных SQLite.

private static SQLiteCommand sql\_cmd; // представляет реализацию интерфейса, для выполняния запросов к базе данных.

private static SQLiteDataAdapter DB; // Считывает результаты базы данных.

private static DataSet DS = new DataSet(); // Предоставляет кэш памяти.

private static DataTable DT = new DataTable(); // Создает таблицу в памяти.

private static string dbPath = Application.StartupPath + "\\" + "DBStudent.db"; // Прокладывает путь к базе данных.

// Загружает дату в наш конструктор.

private void Form1\_Load(object sender, EventArgs e)

{

LoadData();

}

// Подключает дата сурс, который в свою очередь позволяет получать и задавать данные. Все это записывается в путь к базе данных.

private void SetConnection()

{

sql\_con = new SQLiteConnection("Data Source= " + dbPath);

}

//Создает универсальную функцию, чтобы выполнить запросы команды Create.

private void ExecuteQuery(string query)

{

SetConnection();

sql\_con.Open();

sql\_cmd = sql\_con.CreateCommand();

sql\_cmd.CommandText = query;

sql\_cmd.ExecuteNonQuery();

sql\_con.Close();

}

// Создаем доступ к базе данных SQLite, извлекаем данные из таблицы и заполняем набор данных.

private void LoadData()

{

SetConnection();

sql\_con.Open();

sql\_cmd = sql\_con.CreateCommand();

string CommandText = "select \* from StudentList";

DB = new SQLiteDataAdapter(CommandText, sql\_con);

DS.Reset();

DB.Fill(DS);

DT = DS.Tables[0];

dataGridView1.DataSource = DT;

sql\_con.Close();

}

// Выполняет удаление данных

private void Deletebutton\_Click\_1(object sender, EventArgs e)

{

string query = "DELETE FROM StudentList WHERE ID = '" + IDtextBox.Text + "' ;";

ExecuteQuery(query);

LoadData();

}

// Выполняет запись данных

private void Addbutton\_Click(object sender, EventArgs e)

{

string query = "INSERT INTO StudentList (`FirstName`, `LastName`, `MiddleName`, `EducationForm`, `Snills`, `BirthDate`)" +

" VALUES('" + FirstNametextBox.Text + "','" + LastNametextBox.Text + "','" + MiddleNametextBox.Text + "','" + EducationFormtextBox.Text +

"','" + SnillstextBox.Text + "','" + BirthDatetextBox.Text + "')";

ExecuteQuery(query);

LoadData();

}

// Выполняет изменение данных

private void Insertbutton\_Click(object sender, EventArgs e)

{

string query = "UPDATE StudentList SET `FirstName` = '" + FirstNametextBox.Text + "', `LastName` = '" + LastNametextBox.Text +

"', `MiddleName` = '" + MiddleNametextBox.Text + "', `EducationForm` = '" + EducationFormtextBox.Text + "', `Snills` = '" +

SnillstextBox.Text + "', `BirthDate` = '" + BirthDatetextBox.Text + "' WHERE `ID` = '" + IDtextBox.Text + "' ;";

ExecuteQuery(query);

LoadData();

}

//Выполняет выход из программы

private void Exitbutton\_Click(object sender, EventArgs e)

{

Application.Exit();

}

//При нажатие определенной ячейки, выделяется ряд

private void DataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex == -1)

return;

dataGridView1.CurrentRow.Selected = true;

IDtextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["ID"].FormattedValue.ToString();

FirstNametextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["FirstName"].FormattedValue.ToString();

LastNametextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["LastName"].FormattedValue.ToString();

MiddleNametextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["MiddleName"].FormattedValue.ToString();

EducationFormtextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["EducationForm"].FormattedValue.ToString();

SnillstextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["Snills"].FormattedValue.ToString();

BirthDatetextBox.Text = dataGridView1.Rows[e.RowIndex].Cells["BirthDate"].FormattedValue.ToString();

}

}

}

**Приложение 3.**

**Листинг 3. Program.cs**

using System;

using System.Windows.Forms;

namespace ListOfStudents

{

static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

**Класс программы «HelperDoc»**

**Приложение 4.**

**Листинг 4. sqlite.py**

import sqlite3

class Sqliter:

def \_\_init\_\_(self, name\_db):

"""

Подключение к бд

:param name\_db: имя базы данных

"""

self.connection = sqlite3.connect(name\_db)

self.cursor = self.connection.cursor()

self.table = str()

self.table\_profession = "profession"

self.table\_qualification = "qualification"

self.table\_education\_program = "education\_program"

self.table\_student = "student"

self.table\_document = "document"

@staticmethod

def is\_valid(table):

"""

Валидна ли таблица

:param table: название таблицы

:return: True - если название таблицы подходит; False - если нет

"""

is\_valid = False

if len(table) != 0 or type(table) == str:

new\_table = "".join(table.split('\_'))

is\_valid = new\_table.isalnum()

return is\_valid

def connect\_table(self, table):

"""

Функция create\_table создаёт таблицу в БД

:param table: имя таблицы

:return: True - если подключились к таблице, False - если нет

"""

if table is None:

table = self.table

with self.connection:

is\_valid = self.is\_valid(table)

if is\_valid:

self.table = table

return is\_valid

def get\_qualification\_id(self, title):

"""

Вернет id записи от таблицы qualification.

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_qualification} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_qualification(self, title: str):

"""

Функция add добавляет данные в таблицу qualification

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_qualification}` (title) VALUES (?)", (title,))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_qualification: Такой пользователь уже существует")

return is\_valid

def del\_qualification(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_qualification} WHERE `id` = {pk}')

self.save()

def get\_education\_program\_id(self, title):

"""

Вернет id записи от таблицы student.

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_education\_program} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_education\_program(self, title: str):

"""

Функция add добавляет данные в таблицу education\_program

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_education\_program}` (title) VALUES (?)", (title,))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_education\_program: Такой пользователь уже существует")

return is\_valid

def del\_education\_program(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_education\_program} WHERE `id` = {pk}')

self.save()

def get\_profession\_id(self, title):

"""

Вернет id записи из таблицы profession

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `code` FROM {self.table\_profession} WHERE `title` = \'{title}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_profession(self, code: str, title: str):

"""

Функция add добавляет данные в таблицу profession

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_profession}` (code, title) VALUES (?,?)", (code, title))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_profession: Такой пользователь уже существует")

return is\_valid

def del\_profession(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_profession} WHERE `code` = \'{pk}\'')

self.save()

def get\_student\_id(self, name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

command = f'SELECT `id` FROM {self.table\_student} WHERE `name` = ? AND \

`last\_name` = ? AND `middle\_name` = ? AND \

`birth\_date` = ? AND `gender` = ? AND \

`snills` = ? AND `country\_code` = ? AND \

`education\_form` = ? AND \

`education\_receipt\_form` = ? AND \

`financing\_source` = ? AND \

`admission\_year` = ? AND \

`graduation\_year` = ? AND \

`study\_period` = ? AND \

`profession\_code` = ? AND \

`qualification\_id` = ? AND \

`education\_program\_id` = ?'

args = (

name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

)

data = self.cursor.execute(command, args).fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_student(self, name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id

):

"""

Функция add добавляет данные в таблицу student

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_student}` (name, last\_name, middle\_name, birth\_date, \

gender, snills, country\_code, education\_form, education\_receipt\_form, financing\_source, \

admission\_year, graduation\_year, study\_period, profession\_code, qualification\_id, \

education\_program\_id) VALUES (?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)",

(name, last\_name, middle\_name, birth\_date, gender,

snills, country\_code, education\_form, education\_receipt\_form,

financing\_source, admission\_year, graduation\_year, study\_period,

profession\_code, qualification\_id, education\_program\_id))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_student: Такой пользователь уже существует")

return is\_valid

def del\_student(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_student} WHERE `id` = {pk}')

self.save()

def get\_document\_id(self, title, type, series, number, loss\_confirmation, exchange\_confirmation,

destruction\_confirmation, education\_level, issue\_date, registration\_number,

status, student\_id):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

data = self.cursor.execute(

f'SELECT `id` FROM {self.table\_document} WHERE `title` = \'{title}\' AND \

`type` = \'{type}\' AND `series` = \'{series}\' AND \

`number` = \'{number}\' AND `loss\_confirmation` = \'{loss\_confirmation}\' AND \

`exchange\_confirmation` = \'{exchange\_confirmation}\' AND \

`destruction\_confirmation` = \'{destruction\_confirmation}\' AND \

`education\_level` = \'{education\_level}\' AND `issue\_date` = \'{issue\_date}\' AND \

`registration\_number` = \'{registration\_number}\' AND `status` = \'{status}\' AND \

`student\_id` = \'{student\_id}\'').fetchall()

if len(data) != 0:

return data[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def add\_document(self, title, type, series, number, loss\_confirmation, exchange\_confirmation,

destruction\_confirmation, education\_level, issue\_date, registration\_number,

status, student\_id):

"""

Функция add добавляет данные в таблицу profession

:return:

"""

is\_valid = False

try:

with self.connection:

self.cursor.execute(f"INSERT INTO `{self.table\_document}` (title, type, series, number, \

loss\_confirmation, exchange\_confirmation, destruction\_confirmation, education\_level, issue\_date, \

registration\_number,status, student\_id) VALUES (?,?,?,?,?,?,?,?,?,?,?,?)", (title, type, series, number,

loss\_confirmation,

exchange\_confirmation,

destruction\_confirmation,

education\_level,

issue\_date,

registration\_number,

status, student\_id))

self.save()

is\_valid = True

except sqlite3.IntegrityError:

print("add\_document: Такой пользователь уже существует")

return is\_valid

def del\_document(self, pk):

with self.connection:

self.cursor.execute(f'DELETE FROM {self.table\_document} WHERE `id` = {pk}')

self.save()

def get\_id(self, table: str, name\_pk: str, data: dict):

"""

Вернет id студента от таблицы student

:return:

"""

try:

with self.connection:

command = f"SELECT `{name\_pk}` FROM `{table}`"

count = len(data)

if count == 0:

return -1

command += " WHERE "

column\_values = list()

for key, value in data.items():

count -= 1

if value is not None:

command += f"`{key}` = ?"

column\_values.append(value)

if count:

command += " AND "

need\_id = self.cursor.execute(command, tuple(column\_values)).fetchall()

if len(need\_id) != 0:

return need\_id[0][0]

return -1

except sqlite3.IntegrityError:

return -1

def get\_last\_id(self, table, name\_pk):

"""

Вернет последнее id записи из таблицы которую вы передадите

:return:

"""

try:

with self.connection:

return self.cursor.execute(f"SELECT MAX({name\_pk}) FROM {table}").fetchall()[0][0]

except sqlite3.IntegrityError:

return -1

def select(self, table: str, need: str, data: dict):

"""

Вернет выбранные данные (need) из выбранной таблицы (table).

Выборка происходит благодаря переданным данным (data)

:return:

"""

try:

with self.connection:

command = f"SELECT {need} FROM `{table}`"

if len(data) == 0:

return self.cursor.execute(command).fetchall()

command += " WHERE "

column\_values = list()

count = len(data)

for key, value in data.items():

command += f"`{key}` = ?"

column\_values.append(value)

count -= 1

if count:

command += " AND "

return self.cursor.execute(command, tuple(column\_values)).fetchall()

except sqlite3.IntegrityError:

return -1

# def get

def update(self, table: str, pk: dict, data: dict):

"""

Обновит выбранные данные (data) из выбранной таблицы (table).

Выборка происходит благодаря id таблицы (pk)

:param: все словари имеют следующий вид => {'имя\_колоны': значение\_колоны}

:return:

"""

try:

with self.connection:

column\_values = list()

command = f"UPDATE `{table}`"

key\_pk = list(pk.keys())[0]

command += f" SET `{key\_pk}` WHERE "

column\_values.append(pk[key\_pk])

count = len(data)

for key, value in data.items():

column\_values.append(value)

command += f"`{key}` = ?"

count -= 1

if count:

command += ", "

return self.cursor.execute(command, tuple(column\_values)).fetchall()

except sqlite3.IntegrityError:

return -1

# Функция save сохраняет изменения в БД

def save(self):

self.connection.commit()

# print(f"{self.cursor.rowcount} отредактированно строк")

# Функция close закрывает БД

def close(self):

self.connection.close()