Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности

Кафедра Информационных систем и программирования

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Профиль Защищенные автоматизированные системы управления

**КУРСОВАЯ РАБОТА**

по дисциплине Технологии и методы программирования

(наименование дисциплины)

на тему: «Генеалогическое дерево»

(тема курсовой работы)

Выполнил студент 2 курса группы 18-К-АС1

Хлыст Д.А.

(Ф.И.О.)

Допущен к защите\_\_\_\_\_\_\_

Руководитель (нормоконтролер) работы \_\_\_\_\_\_\_\_\_О.Б. Попова

Защищен \_\_\_\_\_\_\_\_\_ Оценка \_\_\_\_\_\_\_\_\_\_\_

(дата)

Члены комиссии Н.В. Кушнир\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

К.Е. Тотухов\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краснодар

2020

ФГБОУ ВО «Кубанский государственный технологический университет»

(ФГБОУ ВО КубГТУ)

Институт Компьютерных систем и информационной безопасности

Кафедра Информационных систем и программирования

Специальность 10.05.03 Информационная безопасность автоматизированных систем

Профиль Защищенные автоматизированные системы управления

УТВЕРЖДАЮ

Зав. кафедрой \_\_\_\_\_\_\_\_\_М.В. Янаева

«12» февраля 2020 г.

**ЗАДАНИЕ**

на курсовую работу

Студенту: Хлыст Д.А. группы 18-К-АС1 курса 2

(Ф.И.О.) (№ группы и курса)

Тема проекта: «Генеалогическое дерево»

План работы:

1. Изучение предметной области

2. Проектирование

3.  Описание реализованных диаграмм

Объем работы:

а) пояснительная записка 28 с.

Рекомендуемая литература

1.  Йордон. «Объектно-ориентированный анализ и проектирование систем»

2. Прайс Марк C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов

3.  «Автоматизация проектирования вычислительных систем.» ред. М.Брейер

Срок выполнения: с «15» февраля по «11» мая 2020г.

Срок защиты: с «11» мая по «14» июня 2020 г.

Дата выдачи задания «15» февраля 2020г.

Дата сдачи работы на кафедру «01» июня 2020 г.

Руководитель работы \_\_\_\_\_\_\_\_\_ к.т.н., доцент Попова О.Б.

(должность, подпись,)

Задание принял студент Хлыст Д.А Ф.И.О.

**Реферат**

Курсовая работа: 28 страница, 14 рисунков, 8 используемых источников.

Ключевые слова: ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, ПРОЕКТИРОВАНИЕ, ПОСЛЕДОВАТЕЛЬНОСТИ, МОДЕЛЬ, КЛАСС, ГЕНЕАЛОГИЧЕСКОЕ ДЕРЕВО, ACCES, BPMN, ГАНТ, EPC, FURPS+, IDEF0, DFD, ДИАГРАММЫ.

Объектом исследования является программное обеспечение и создания генеалогического дерева, которое способно, по запросу пользователь, выводить данные из базы данных, выполнять поиск, изменять и дополнять значения по желанию.

Цель работы состоит в разработке приложения «Генеалогическое дерево» с использованием диаграмм разного вида, в полной мере описывающих приложение и взаимодействие системы на всех этапах работы.

В результате были получены диаграммы, в полной мере описывающие работу генеалогического дерева. К таким диаграммам относятся: диаграмма Ганта, UML-диаграмма, IDEF0-диаграмма, DFD-диаграмма, EPC-диаграмма, BPMN «As-Is» и BPMN «To be», документ FURPS+.

**Содержание**

[Введение 5](#_Toc39780977)

[1 Формулировка задачи 6](#_Toc39780978)

[2 Диаграмма Ганта 6](#_Toc39780979)

[3 Создание модели As-Is в стандарте IDEF0 7](#_Toc39780980)

[4 Диаграмма потоков данных (DFD) 9](#_Toc39780981)

[5 UML 11](#_Toc39780982)

[6 EPC 12](#_Toc39780983)

[7 BPMN 12](#_Toc39780984)

[8 FURPS+ 13](#_Toc39780985)

[9 Результаты машинного тестирования программы 15](#_Toc39780986)

[10 Системные требования 16](#_Toc39780987)

[11 Руководство пользователя 17](#_Toc39780988)

[Заключение 19](#_Toc39780989)

[Список использованных источников 20](#_Toc39780989)

[Приложение А – Проверка на антиплагиат 21](#_Toc39780990)

[Приложение Б – Диаграмма Ганта 22](#_Toc39780991)

[Приложение В – Листинг программы 23](#_Toc39780992)

**Введение**

Генеалогическое древо – это схема, в которой отображены родственные связи, чаще всего прямых предков и потомков. В приложении также будет указана информация о человеке: дата рождения, дата смерти, братья и сестры, и другие.

При разработке приложения был использован язык программирования высокого уровня C#, а для обеспечения хранения данный была использована реляционная система управления базами данных Microsoft Access.

Таким образом, данное приложение обеспечивает удобное использование и хранение данных.

**1 Формулировка задачи**

Задачей данного курсового проекта является разработка программного обеспечения системы для поддержки генеалогических деревьев. Пользователи системы могут осуществлять поиск полезной информации по дереву:

− находить для указанного члена семьи его детей;

− находить для указанного члена семьи его родителей;

− находить для указанной персоны братьев и сестер, если таковые есть;

− получать список всех предков персоны;

− получать список всех потомков персоны;

− получать список всех родственников персоны;

Приложение по нажатию кнопки «БД» выводит все информацию, хранящуюся в базе данных. При вводе Id пользователь может либо найти информацию о человеке или удалить все данные о нем в БД, это осуществляется кнопками «Найти» и «Удалить». Id пользователя является уникальным, после удаления следующий пользователь получает новый и уникальный идентификатор. При нажатии кнопки «Добавить» открывается новое окно приложения, где необходимо ввести данные о новом пользователи. При нажатии кнопки «Изменить» открывается новое окно приложения, где необходимо ввести Id пользователя, данные которого необходимо изменить.

# 2 Диаграмма Ганта

Диаграмма Ганта — «это популярный тип столбчатых диаграмм (гистограмм), который используется для иллюстрации плана, графика работ по какому-либо проекту. Является одним из методов планирования проектов. Придумал американский инженер Генри Гант. Выглядит это как горизонтальные полосы, расположенные между двумя осями: списком задач по вертикали и датами по горизонтали.

На диаграмме видны не только сами задачи, но и их последовательность.

Ключевым понятием диаграммы Ганта является «веха» — метка значимого момента в ходе выполнения работ, общая граница двух или более задач. Вехи позволяют наглядно отобразить необходимость синхронизации, последовательности в выполнении различных работ. Сдвиг вехи приводит к сдвигу всего проекта. Поэтому диаграмма Ганта не является графиком работ. Кроме того, диаграмма Ганта не отображает значимости или ресурсоемкости работ, не отображает области действия. Для крупных проектов диаграмма Ганта становится чрезмерно тяжеловесной и теряет всякую наглядность.»

Диаграмма Ганта для проекта «Генеалогическое дерево» находится в «Приложении Б».

**3 Создание модели As-Is в стандарте IDEF0**

Чтобы оценить возможности, разрабатываемой системы необходимо построить её базовую модель, которую можно представить в виде диаграммы As-Is.

Диаграмма As-Is – это функциональная модель системы «как есть», позволяющая узнать, где находятся слабые места, в чём будут состоять преимущества и недостатки, протекающих в ней бизнес-процессов относительно конкурентов. Применение данной модели позволит чётко зафиксировать какие информационные объекты принимают участие в жизненном цикле системы, какая информация будет поступать на вход и что будет получаться на выходе. Модель As-Is, строится с использованием нотации IDEF0.

IDEF0 – это графическая нотация, предназначенная для описания бизнес-процессов. Система, описываемая в данной нотации, проходит через декомпозицию или, иными словами, разбиение на взаимосвязанные функции. Для каждой функции существует правило сторон:

– стрелкой слева обозначаются входные данные;

– стрелкой сверху – управление;

– стрелкой справа – выходные данные;

– стрелкой снизу – механизм.

Учитывая всё вышеперечисленное на рисунке 1 была составлена модель As-Is проекта «Генеалогическое дерево».

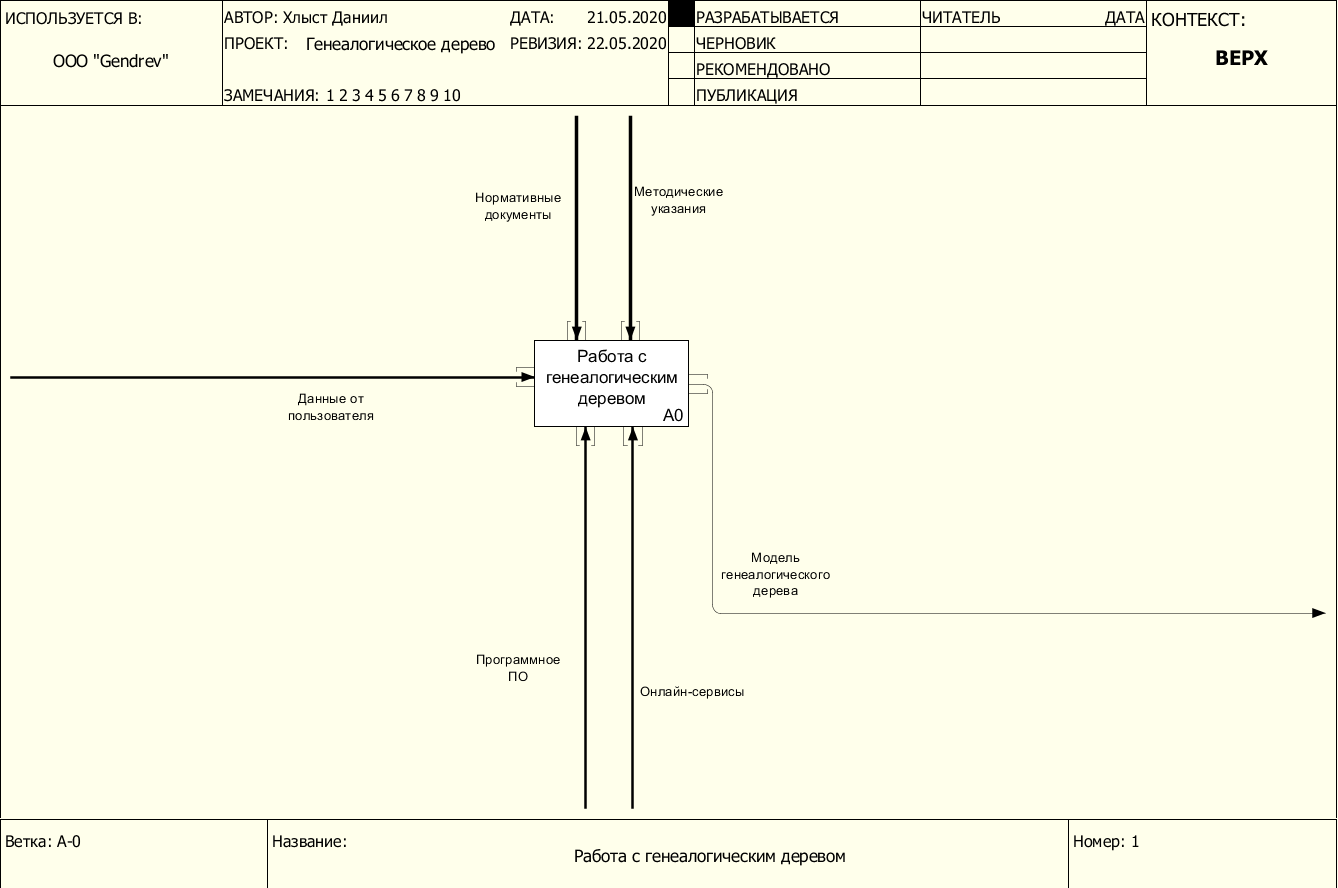


Рисунок 1 – Диаграмма IDEF0 уровня A-0.

Полученная модель системы может быть представлена в более подробном виде путём разбиения на большее количество составных элементов.

На рисунке 2 можно видеть модель кода «Генеалогическое дерево» после декомпозиции.

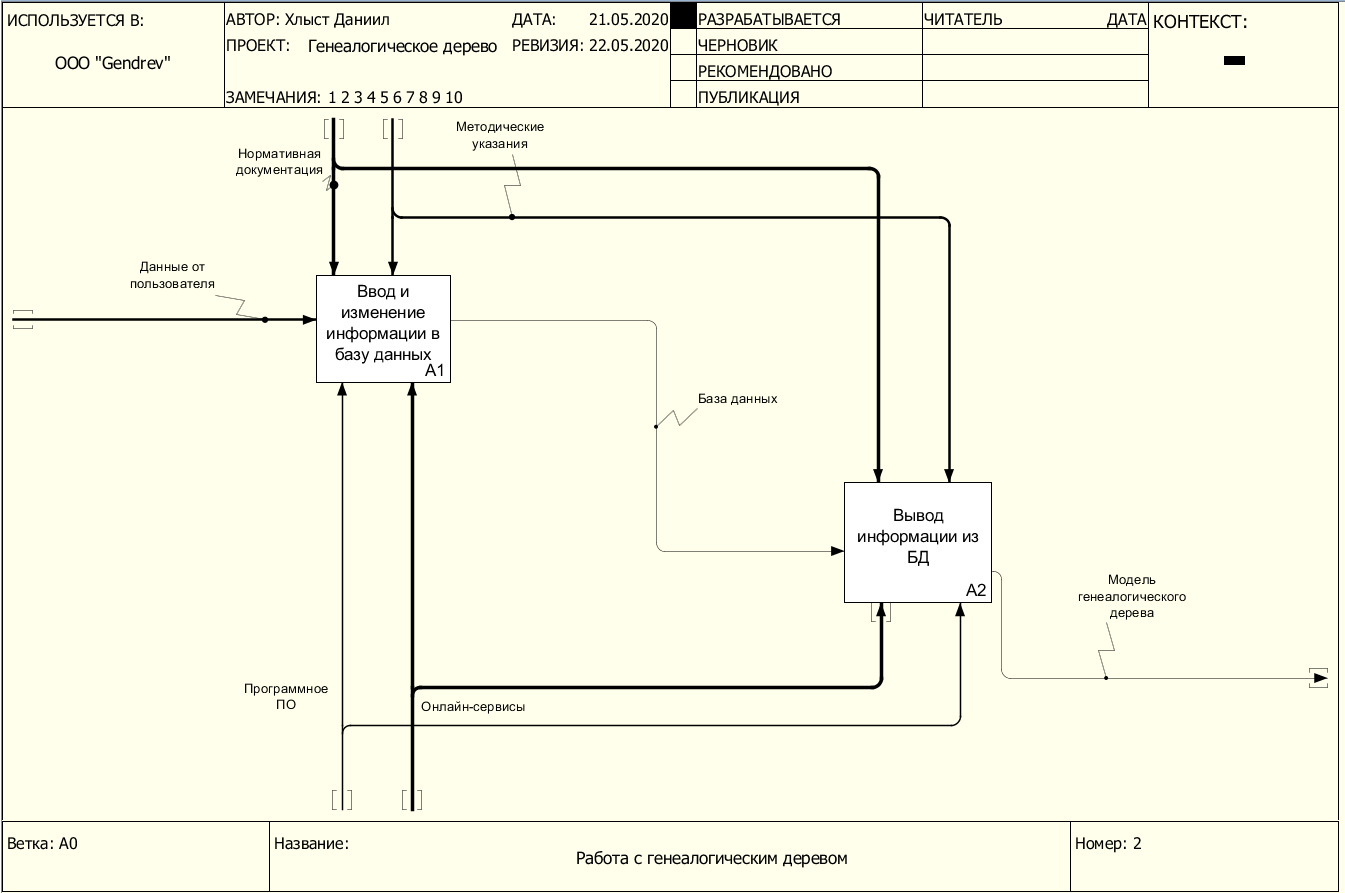


Рисунок 2 – Диаграмма IDEF0 уровня A0.

Для удобного представления работы программы и ее комплектующих данную схему можно рассмотреть более подробно, для этого необходимо разбить процессы A1 и A2. Данные декомпозиции будут представлены на рисунках 3 и 4.

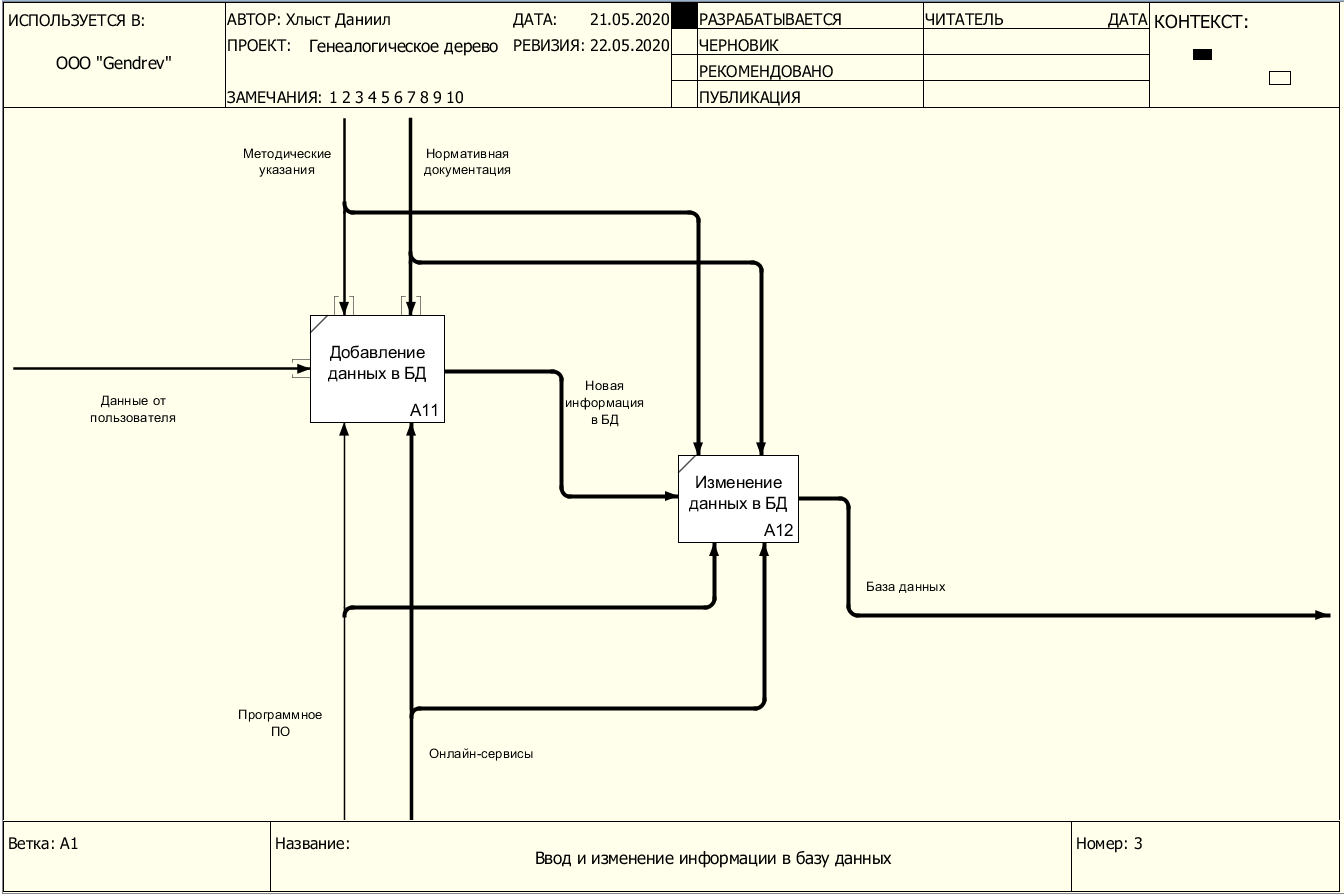


Рисунок 3 – Диаграмма декомпозции А1.

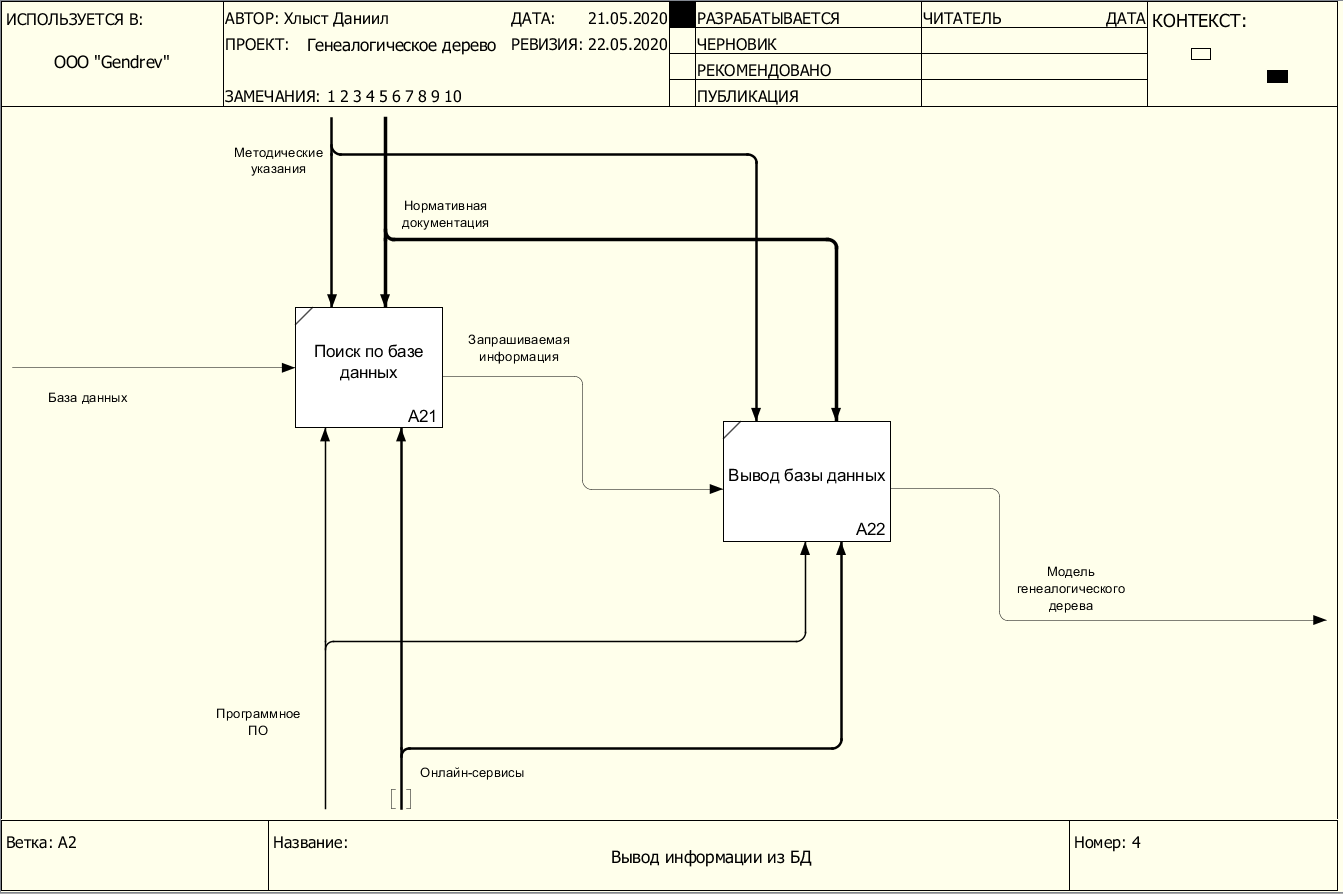


Рисунок 4 – Диаграмма декомпозции А2.

**4 Диаграмма потоков данных (DFD)**

Диаграмма потоков данных (data flow diagram, DFD) — один из основных инструментов структурного анализа и проектирования информационных систем, существовавших до широкого распространения UML. DFD – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных.

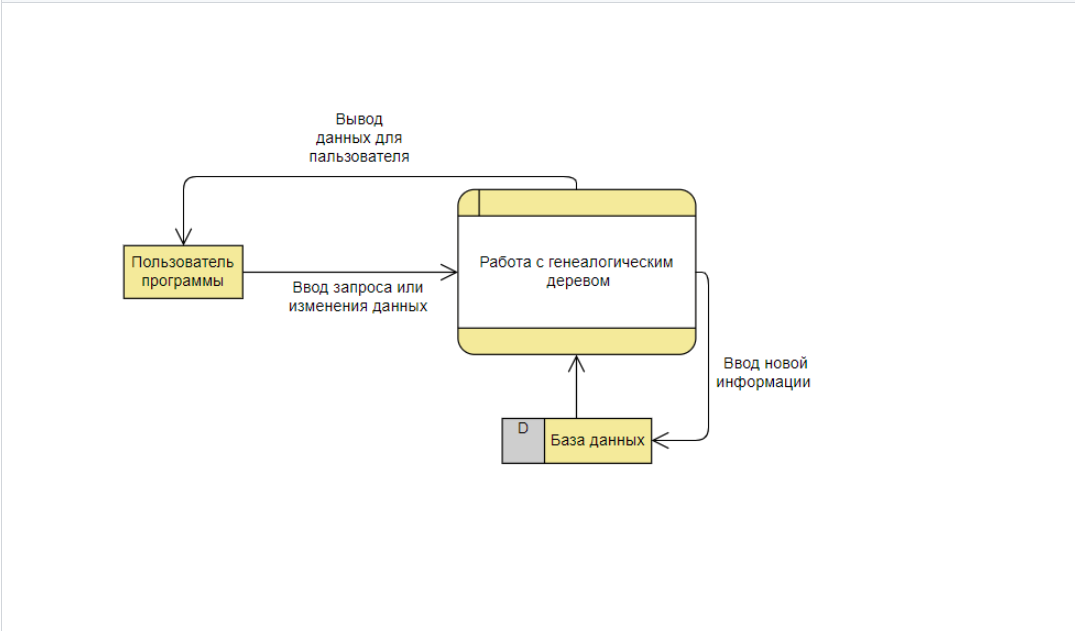


Рисунок 5 – Диаграмма DFD-системы «Генеалогическое дерево».

**5 UML**

UML — унифицированный язык моделирования, язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур.

UML позволяет также разработчикам программного обеспечения достигнуть соглашения в графических обозначениях для представления общих понятий таких как класс, компонент, обобщение и больше

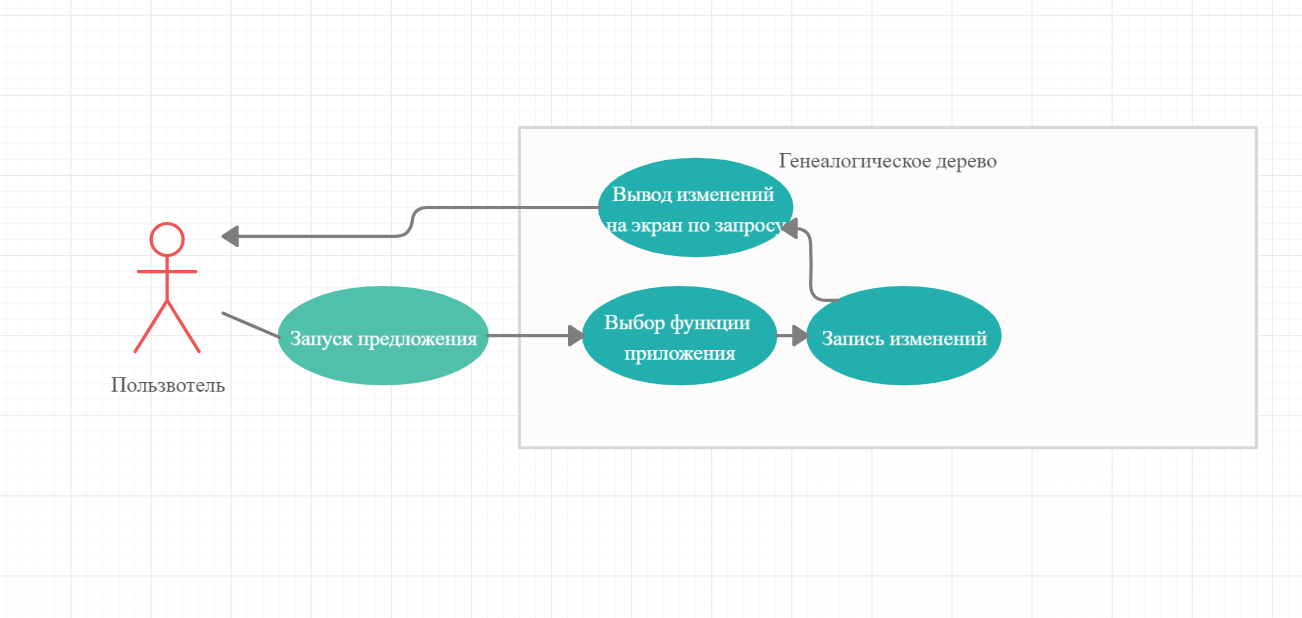
сконцентрироваться на проектировании и архитектуре. 

Рисунок 6 – UML-диаграмма системы «Генеалогическое дерево».

# 6 EPC

EPC-диаграммы — один из видов блок-схем, предназначенный для бизнес-моделирования. Название происходит от английского event-driven process chain, что переводится как событийная цепочка процессов. Организации используют EPC-диаграммы для планирования потоков работ бизнес-процессов. EPC-диаграммы используют символы нескольких видов, чтобы показать структуру потока управления (последовательность решений, функции, события и другие элементы) бизнес-процесса.

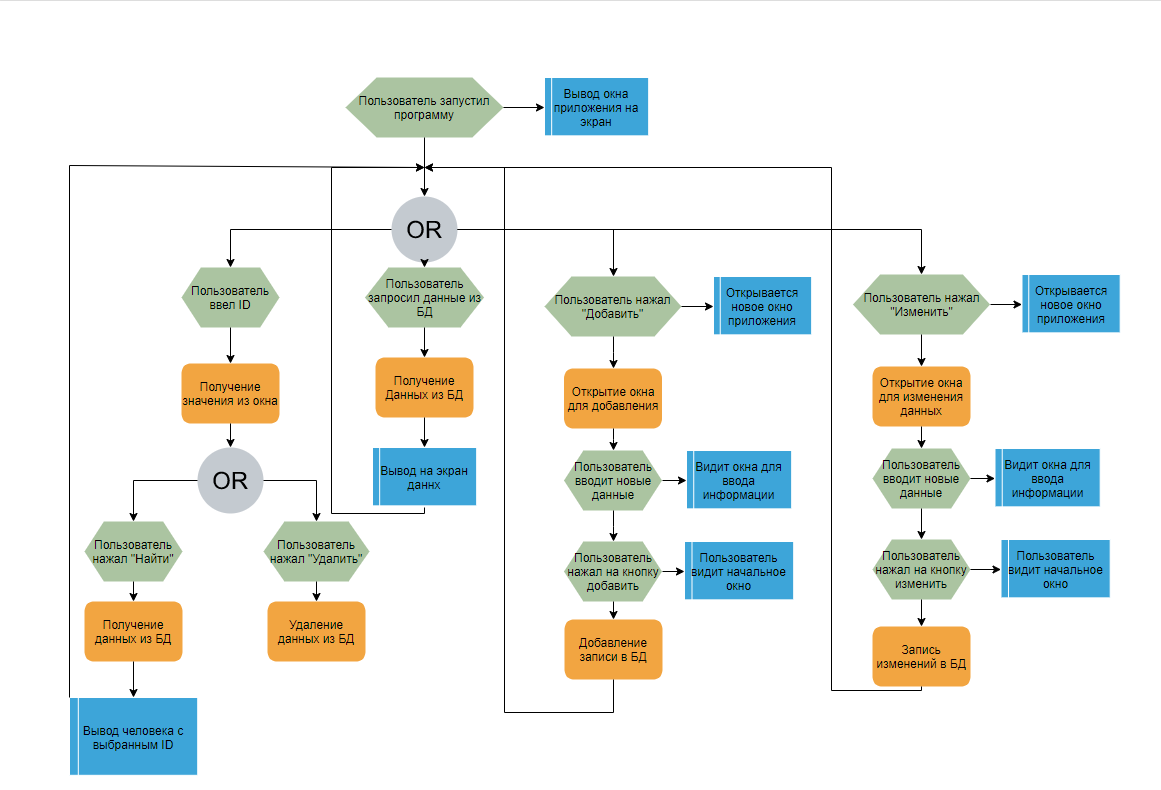
EPC-метод был разработан Августом-Вильгельмом Шеером в рамках работ над созданием методологии ARIS (Архитектура интегрированных информационных систем) в начале 1990-х годов. 

Рисунок 7 – EPC-диаграмма системы «Генеалогическое дерево».

# 7 BPMN

BPMN (Business Process Management Notation) – это язык моделирования бизнес-процессов, который является промежуточным звеном между формализацией/визуализацией и воплощением бизнес-процесса.

То есть - это схемы протекания бизнес-процесса. Они необходимы для выстраивания соответствий процессов для пояснения участникам, а также облегчит автоматизацию в будущем.

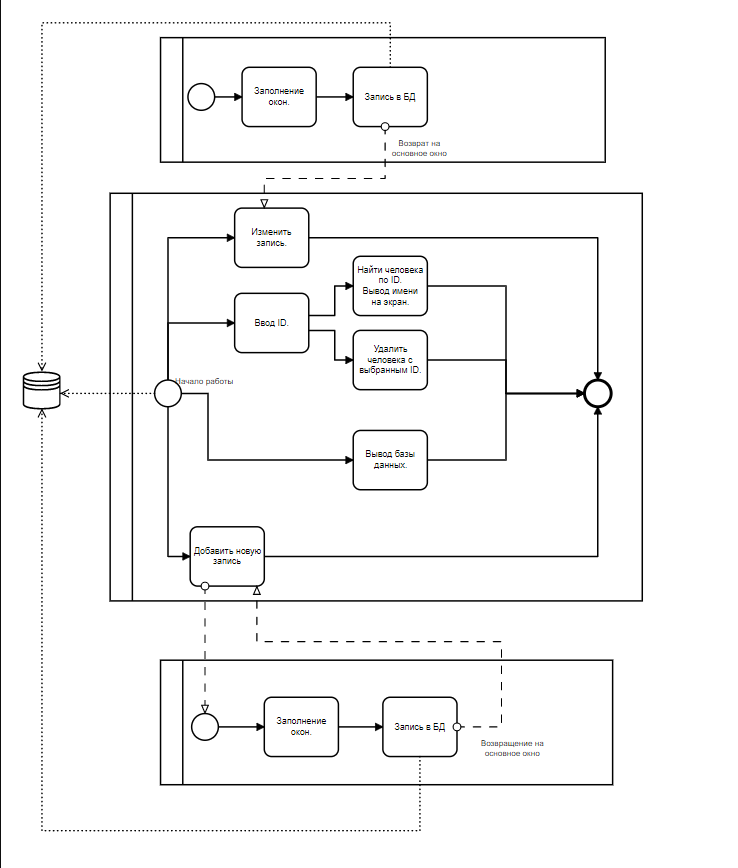


Рисунок 8 – Диаграмма BPMN «As-Is» и «To be».

# FURPS+

FURPS+ — это одна из распространенных классификаций требований к информационным системам. Название произошло от первых букв слов Functionality, Usability, Reliability, Performance, Supportability (Функциональность, Удобство использования, Надежность, Производительность, Поддерживаемость).

Стоит отметить, что классификация FURPS+ содержит не только требования, но и ограничения, которые выделены в названии знаком «плюса». В рамках рассматриваемой классификации выделяют четыре вида ограничений:

1) ограничения проектирования (Design). В качестве примера может служить требование использовать реляционную базу данных в качестве основной;

2) ограничения разработки (Implementation). В рамках данных ограничений может указываться конкретная методология, конкретный язык программирования;

3) ограничения интерфейсов (Interface), к которым относят ограничения на конкретные форматы данных;

4) физические ограничения (Physical), включающие все возможные ограничения внешней среды. В качестве такого ограничения может служить уровень влажности, температурный режим.

F — изменение данных в БД по запросу пользователя;

U — интуитивно понятный интерфейс, доступность и удобство в получении информации;

R — режим доступности системы, отказоустойчивость;

P — время отклика системы 0,1 сек, эффективность работы;

S — простота установки;

+ — никаких ограничений;

# 9 Результаты машинного тестирования программы

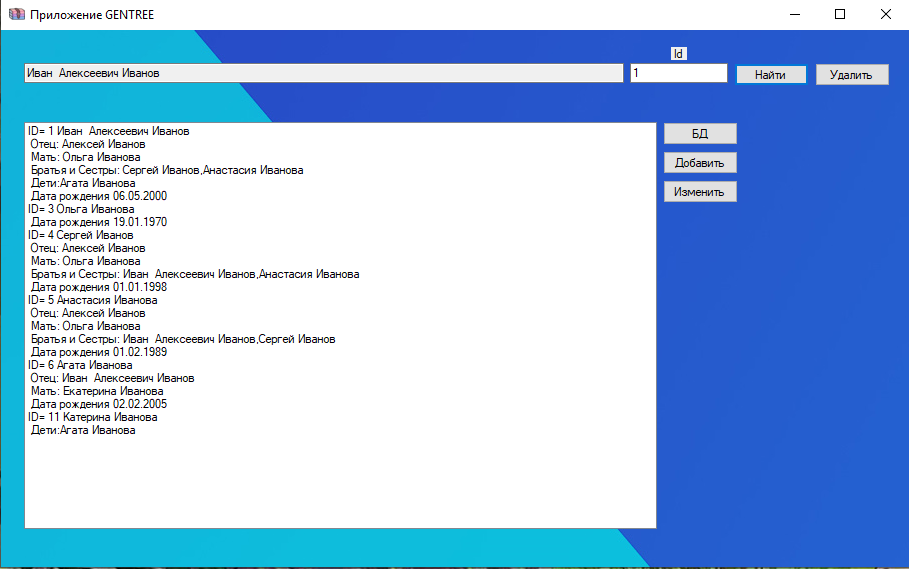


Рисунок 9 – Запуск программу и запрос БД.

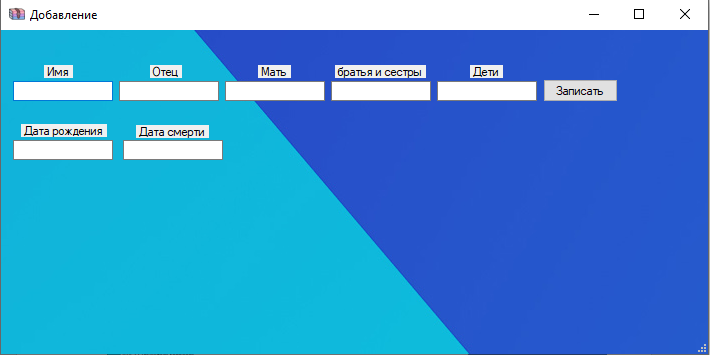


Рисунок 10 – Запрос на добавление.



Рисунок 11 – Запрос на изменение.

**10 Системные требования**

Таблица 1 – Системные требования программы

|  |  |
| --- | --- |
| Процессор | 2.5 ГГц |
| Оперативная память | 150 Мб |
| Монитор | 1920 x 1080 |
| Свободное место на носителе | 15 Мб |
| Устройства взаимодействия с пользователем | Клавиатура и мышь |
| Программное обеспечение | Visual Studio 2020 года последней версии |

**11 Руководство пользователя**

Запуск программы можно осуществить следующим способом: открыть непосредственно в Visual Studio, открыть из папки проекта файл формата .sln или открыть исполняемый файл из папки проекта.

1. Запуск непосредственно в Visual Studio:

1) Открываем Visual Studio.

2) Нажимаем на вкладку «Файл» => «Открыть» => «Решение или

проект» (рисунок 12) .

3) В диалоговом окне находим папку проекта и нажимаем на файл в формате .sln, жмем кнопку «Открыть» (рисунок 10).

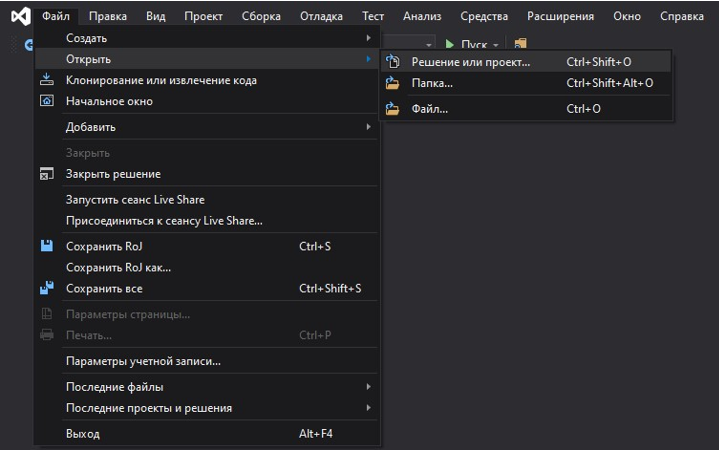
4) Нажимаем на кнопку «Пуск» (рисунок 11). 

Рисунок 12 – Открытие проекта.

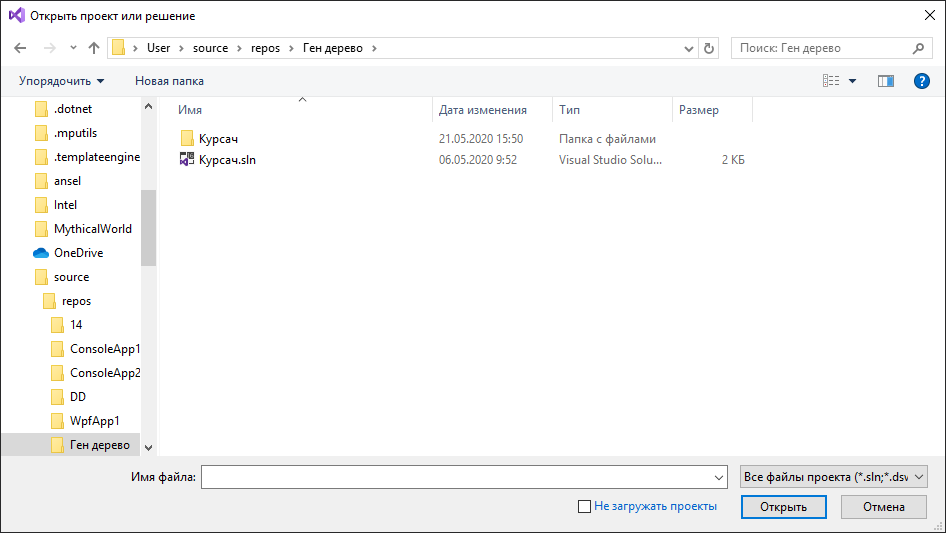


Рисунок 13 – Выбор файла.

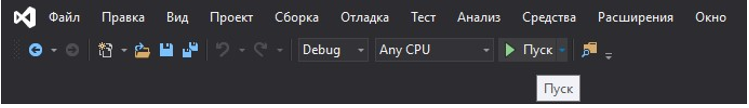


Рисунок 14 – Запуск программы.

Если необходимо получить все данные из БД, то нужно нажать кнопку «БД». При вводе в окно с надписью Id можно получить данные о человека при нажатии «Найти», или удалить пользователя при нажатии «Удалить». Если нажать кнопку «Добавить», то открывается новое окно, где необходимо ввести данные о новом человеке, id присваивается автоматически. Если нажать кнопку «Изменить», то открывается новое окно, где необходимо ввести id и новые данные человека.

**Заключение**

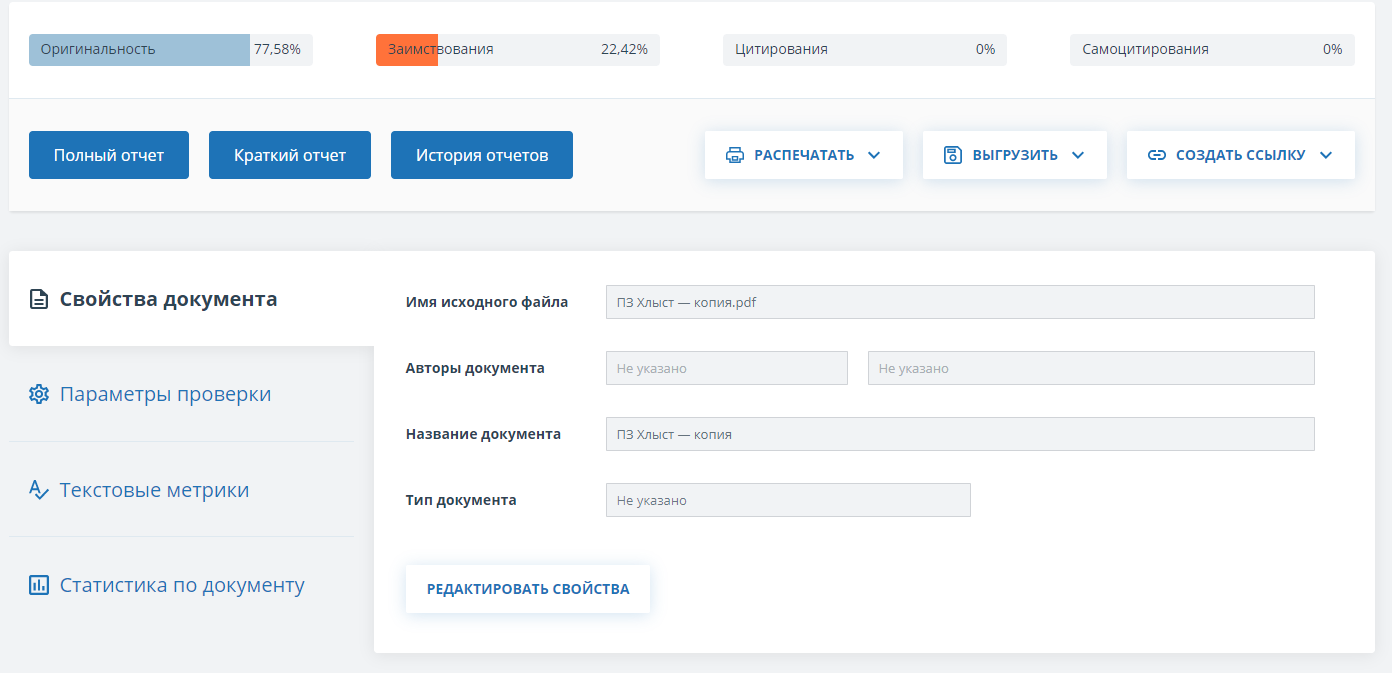
В результате выполнения данного курсового проекта была спроектирована система «Генеалогическое дерево» на языке высокого уровня C#, позволяющая наглядно продемонстрировать работу всех её компонентов. Полученные диаграммы обладают простой и понятной, структурой, описывающей каждый аспект системы с различных сторон.

Код программы был отлажен вручную. Были получены важные знания и практические навыки как в области использования объектно-ориентированных языков программирования в целом, так и в области построения диаграмм проектирования, отображающих поведение различных организационных структур.

**Список использованных источников**

1. Ларман, Крэг. Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / КрэгЛарман. - Москва: Гостехиздат, 2017. - 736 c.
2. Роберт А. Максимчук. UML для простых смертных / Роберт А. Максимчук, Эрик Дж. Нейбург. - Москва: СИНТЕГ, 2014. - 272 c.
3. Йордон, Эдвард. Объектно-ориентированный анализ и проектирование систем / Эдвард Йордон , Карл Аргила. - М.: ЛОРИ, 2014. - 264 c.
4. SoloLearn – C# Tutorial. [Электронный ресурс]: - Режим доступа: <https://www.sololearn.com/Course/CSharp/> (Дата обращения 13.03.2020).
5. Википедия. [Электронный ресурс]: - Режим доступа: [https://ru.wikipedia.org](https://ru.wikipedia.org/wiki/Тестовые_функции_для_оптимизации) (Дата обращения 17.09.2019).
6. GitHub – yarajtf/intercom. [Электронный ресурс]: - Режим доступа: <https://github.com/yarajtf/intercom> (Дата обращения 06.05.2020).
7. Comindware – Нотация BPMN 2.0 [Электронный ресурс]: - Режим доступа: <https://comindware.com/ru/blog-нотация-bpmn-2-0-элементы-и-описание/> (Дата обращения 28.02.2020)
8. SysAna– Требования к системе: классификация FURPS+ [Электронный ресурс]: - Режим доступа: <https://sysana.wordpress.com/2010/09/16/furps/> (Дата обращения 03.03.2020)

# Приложение А – Проверка на антиплагиат



# Приложение Б – Диаграмма Ганта

# 

# Приложение В – Листинг программы

Таблица 2 – Характеристика переменных к заданию

|  |  |  |  |
| --- | --- | --- | --- |
| Имя переменной | Смысл переменной | Назначение переменной | Ограничения |
| textBox1.Text | Значение на панели | Исходная | Целочисленная, строковая |
| textBox2.Text | Значение на панели | Исходная | Строковая |
| textBox3.Text | Значение на панели | Исходная | Строковая |
| textBox4.Text | Значение на панели | Исходная | Строковая |
| textBox5.Text | Значение на панели | Исходная | Строковая |
| textBox6.Text | Значение на панели | Исходная | Целочисленная |
| textBox7.Text | Значение на панели | Исходная | Целочисленная |
| T | Проверка истинности | Исходная | Булевая |
| str | Начало командной сторки | Исходная | Строковая |

Код программы:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.OleDb;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Курсач

{

public partial class Form1 : Form

{

public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=kursovoymdb.mdb;";

private OleDbConnection myConnection;

public Form1()

{

InitializeComponent();

myConnection = new OleDbConnection(connectString);

myConnection.Open();

}

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

{

}

private void Form1\_FormClosing(object sender, FormClosingEventArgs e)

{

myConnection.Close();

}

private void Button1\_Click(object sender, EventArgs e)

{

string query = "SELECT id, Name, F, M, BS, CH, Data, dedData FROM BD ORDER BY id";

OleDbCommand command = new OleDbCommand(query, myConnection);

OleDbDataReader reader = command.ExecuteReader();

listBox1.Items.Clear();

while (reader.Read())

{

listBox1.Items.Add("ID= " + reader[0].ToString() + " " + reader[1].ToString());

if (reader[2].ToString() !="") listBox1.Items.Add(" Отец: " + reader[2].ToString());

if (reader[3].ToString() !="") listBox1.Items.Add(" Мать: " + reader[3].ToString());

if (reader[4].ToString() !="") listBox1.Items.Add(" Братья и Сестры: " + reader[4].ToString());

if (reader[5].ToString() !="") listBox1.Items.Add(" Дети:" + reader[5].ToString());

if (reader[6].ToString() !="") listBox1.Items.Add(" Дата рождения " + reader[6].ToString());

if (reader[7].ToString() !="") listBox1.Items.Add(" Дата смерти " + reader[7].ToString());

}

reader.Close();

}

private void TextBox1\_TextChanged(object sender, EventArgs e)

{

}

private void Button2\_Click(object sender, EventArgs e)

{

string query = "SELECT Name FROM BD WHERE id = " + textBox1.Text;

OleDbCommand command = new OleDbCommand(query, myConnection);

textBox2.Text = command.ExecuteScalar().ToString();

}

private void Button1\_Click\_1(object sender, EventArgs e)

{

Form2 form = new Form2();

this.Hide();

form.ShowDialog();

this.Show();

}

private void Button3\_Click(object sender, EventArgs e)

{

string query = "DELETE FROM BD WHERE id =" + textBox1.Text;

OleDbCommand command = new OleDbCommand(query, myConnection);

command.ExecuteNonQuery();

MessageBox.Show("Информация удалена");

}

private void Button4\_Click(object sender, EventArgs e)

{

Form3 form = new Form3();

this.Hide();

form.ShowDialog();

this.Show();

}

private void Button5\_Click(object sender, EventArgs e)

{

}

private void Label1\_Click(object sender, EventArgs e)

{

}

}

}

public partial class Form2 : Form

{

public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=kursovoymdb.mdb;";

private OleDbConnection myConnection;

public Form2()

{

InitializeComponent();

myConnection = new OleDbConnection(connectString);

myConnection.Open();

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void TextBox1\_TextChanged(object sender, EventArgs e)

{

}

private void Button1\_Click(object sender, EventArgs e)

{

string query = "INSERT INTO BD (Name, F, M, BS, CH, Data, dedData) VALUES ('" + textBox1.Text + "', '" + textBox2.Text + "', '" + textBox3.Text + "','" + textBox4.Text + "','" + textBox5.Text + "','" + textBox6.Text + "','" + textBox7.Text + "')";

OleDbCommand command = new OleDbCommand(query, myConnection);

command.ExecuteNonQuery();

this.Close();

}

private void Label1\_Click(object sender, EventArgs e)

{

}

private void TextBox6\_TextChanged(object sender, EventArgs e)

{

}

}

}

public partial class Form3 : Form

{

public static string connectString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=kursovoymdb.mdb;";

//public static string connectString = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source=kursovoymdb.mdb;";

private OleDbConnection myConnection;

public Form3()

{

InitializeComponent();

myConnection = new OleDbConnection(connectString);

myConnection.Open();

}

private void Button1\_Click(object sender, EventArgs e)

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "" && textBox5.Text != "") return;

bool T = false;

string str = "UPDATE BD SET ";

if(textBox1.Text!="")

{

str += "Name = '" + textBox1.Text + "'";

T = true;

}

if (textBox2.Text != "")

{

str += T ? ", " : " ";

str += "F = '" + textBox2.Text + "' ";

T = true;

}

if (textBox3.Text != "")

{

str += T ? "," : "";

str += "M = '" + textBox3.Text + "' ";

T = true;

}

if (textBox4.Text != "")

{

str += T ? "," : "";

str += "BS = '" + textBox4.Text + "' ";

T = true;

}

if (textBox5.Text != "")

{

str += T ? "," : "";

str += "CH = '" + textBox5.Text + "' ";

T = true;

}

if (textBox5.Text != "")

{

str += T ? "," : "";

str += "Data = '" + textBox7.Text + "' ";

T = true;

}

if (textBox5.Text != "")

{

str += T ? "," : "";

str += "dedData = '" + textBox8.Text + "' ";

}

str += "WHERE id = " + textBox6.Text;

OleDbCommand command = new OleDbCommand(str, myConnection);

command.ExecuteNonQuery();

this.Close();

}

private void Form3\_Load(object sender, EventArgs e)

{

}

private void TextBox6\_TextChanged(object sender, EventArgs e)

{

}

private void TextBox1\_TextChanged(object sender, EventArgs e)

{

}

private void TextBox7\_TextChanged(object sender, EventArgs e)

{

}

private void Label8\_Click(object sender, EventArgs e)

{

}

private void TextBox8\_TextChanged(object sender, EventArgs e)

{

}

}