**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ**

**Федеральное государственное бюджетное образовательное**

**учреждение высшего образования**

**«Пензенский государственный университет»**

(**ФГБОУ ВО «Пензенский государственный университет») \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

#### Кафедра «Математическое обеспечение и применение ЭВМ»

**«УТВЕРЖДАЮ»**

**Зав. Кафедрой *Макарычев П.П.***

**"\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г.**

**ЗАДАНИЕ**

**на курсовое проектирование по дисциплине**

**«Объектно-ориентированное программирование»**

**Студенту Моисееву Владиславу Андреевичу Группа *20ВП1***

**Тема проекта:** Разработка программы с использованием объектно-ориентированного подхода. ИС «Музей».

**Исходные данные (технические требования) на проектирование**

1. Разработать программу БД “Музей”.
2. Данные по базе А включают: а) ФИО посетителя; б) дата и время посещения;

Данные по базе Б включают: а) название экспоната; б) эпоха, к которой относится экспонат; в) подлинность экспоната; г) дата создания экспоната.

1. Программа должна выполнять следующие действия: создание базы данных, удаление базы данных, сохранение текущей базы данных в файл, добавление записей, удаление записей, поиск записей.
2. Обязательные требования к программе: многомодульность, использование сложных типов данных, использование коллекции для организации базы данных. Старт приложения должен сопровождаться всплывающим окном с информацией об авторе и темой проекта.
3. Визуальный интерфейс приложения реализовать с использованием графических библиотек. Например, WinForms.
4. Среда разработки ПО: Microsoft Visual Studio 2022
5. Язык программирования: C#
6. Программное обеспечение должно быть полностью отлажено и протестировано, и должно функционировать под управлением ОС Windows 10 и выше.

**Объём работ по курсу**

**1. Расчётная часть**

|  |
| --- |
| 1) Анализ требований. |
| 2) Выбор и освоение инструментальных средств анализа и проектирования |
| 3) Определение структуры и функций приложения |
| 4) Разработка диаграмм описания приложения |
| 5) Реализация приложения на языке C# в среде Microsoft Visual Studio |
| 6) Отладка и тестирование приложения |
|  |
|  |

**2. Графическая часть**

|  |
| --- |
| 1. Схема данных |
| 1. Диаграмма вариантов использования |
| 1. Диаграмма классов |

**3. Экспериментальная часть**

|  |
| --- |
| 1) Отладка компонентов приложения и их взаимодействия |
| 2) Функциональное тестирование приложения |
|  |
|  |

**Срок выполнения проекта по разделам**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | Анализ требований | к | 22 февраля | 2022 г. |
| 2 | Определение структуры и функций приложения | к | 1 марта | 2022 г. |
| 3 | Разработка UML диаграмм приложения | к | 15 марта | 2022 г. |
| 4 | Реализация приложения | к | 12 апреля | 2022 г. |
| 5 | Отладка и тестирование приложения | к | 29 апреля | 2022 г. |
| 6 | Оформление пояснительной записки проекта | к | 17 мая | 2022 г. |
| 7 | Защита курсовой работы | к | 24 мая | 2022 г. |

Дата выдачи задания «8»  *февраля 2022 г.*

Дата защиты проекта «24»  *мая 2022 г.*

Руководитель /к.т.н. А.Ю. Афонин/

Задание получил “ 8” *февраля* 2022 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/В.А. Моисеев/

**Оглавление**

[Введение 4](#_Toc104839279)

[1. Постановка задачи 5](#_Toc104839280)

[2. Выбор решения 6](#_Toc104839281)

[2.1. Определение основных модулей программы 6](#_Toc104839282)

[2.2. Определение структуры файла базы данных 9](#_Toc104839283)

[3. Описание разработки программы 10](#_Toc104839284)

[4. Отладка и тестирование 15](#_Toc104839285)

[5. Описание программы 19](#_Toc104839286)

[5.1 Разработка приложения 19](#_Toc104839287)

[5.2 Разработка меню 19](#_Toc104839288)

[6. Руководство пользователя 20](#_Toc104839289)

[Заключение 21](#_Toc104839290)

[**Приложение А** 23](#_Toc104839291)

# Введение

В современном мире, когда информационные технологии активно развиваются, одной из актуальных проблем является проблема хранения, поиска и обработки данных. Практически в любых сферах человеческой деятельности появляется необходимость работать с большим объемом данных.

Целью данной курсовой работы является разработка приложения для учета посетителей музея и представленных экспонатов. В настоящие время в мире проходят тысячи экскурсий ежедневно, неотъемлемой частью которых является посещение музеев, поэтому хранить информацию эффективней всего в электронной базе данных. Разрабатываемое приложение позволяет работникам музеев подключаться к единой базе данных, редактировать ее записи или добавлять их, осуществлять поиск записей по заданным критериям, сортировать по столбцам.

Разработка программы производилась на языке программирования С# в среде разработки Visual Studio 2022.

В пояснительной записке рассмотрены основные этапы разработки приложения с пользовательским интерфейсом: анализ требований, проектирование, реализация и тестирование. Также написаны описание программы и руководство пользователя. В качестве средств моделирования используются диаграммы языка UML.

# 1. Постановка задачи

У разработанного приложения старт должен сопровождаться всплывающим окном с информацией об авторе и темой проекта.

На исходный код должна быть сгенерирована документация с использованием средств автоматического документирования кода.

Установка приложения должна обеспечиваться посредством инсталлятора. На проект должен быть сформирован репозиторий на github.

Тема на курсовой проект: «Музей». В данном проекте должен быть реализован учет посетителей и экспонатов, поиск посетителей по заданному критерию, сортировка по столбцам.

# 2. Выбор решения

## 2.1. Определение основных модулей программы

Проанализировав требования к работе, можно выявить ряд модулей, необходимых для функционирования разрабатываемого приложения:

1. класс главной формы приложения (Form1);
2. класс, осуществляющий выборку данных (Form2);
3. класс, позволяющий удалять данные из таблицы (Form3);
4. класс, инициализирующий приветственную страницу (Form4);
5. класс, позволяющий удалять данные из таблицы (Form5);
6. класс, осуществляющий сортировку по колонкам.

На рисунке 1 представлены взаимоотношения модулей разрабатываемой программы:

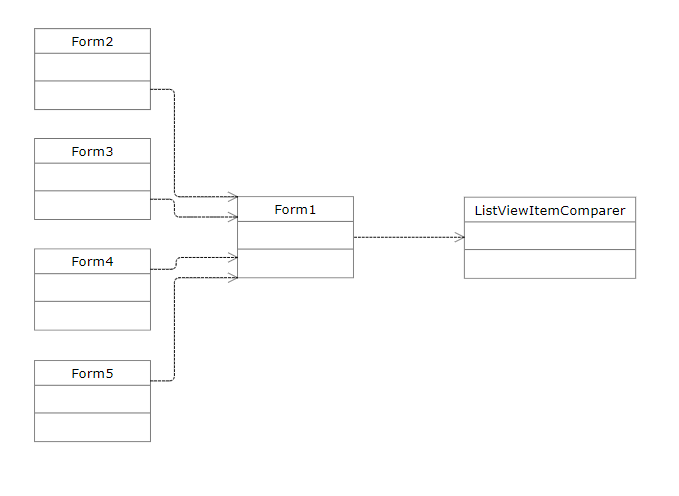


Рисунок 1 – Модель предметной области

Разрабатываемое приложение должно предоставлять ряд функциональных возможностей: просмотр записей из базы данных, их запись, а также их добавление, удаление, фильтрация по определенному критерию, сортировка, поиск. Исходя из описанных выше функциональных требований, можно построить диаграмму вариантов использования, представленную на рисунке 2:

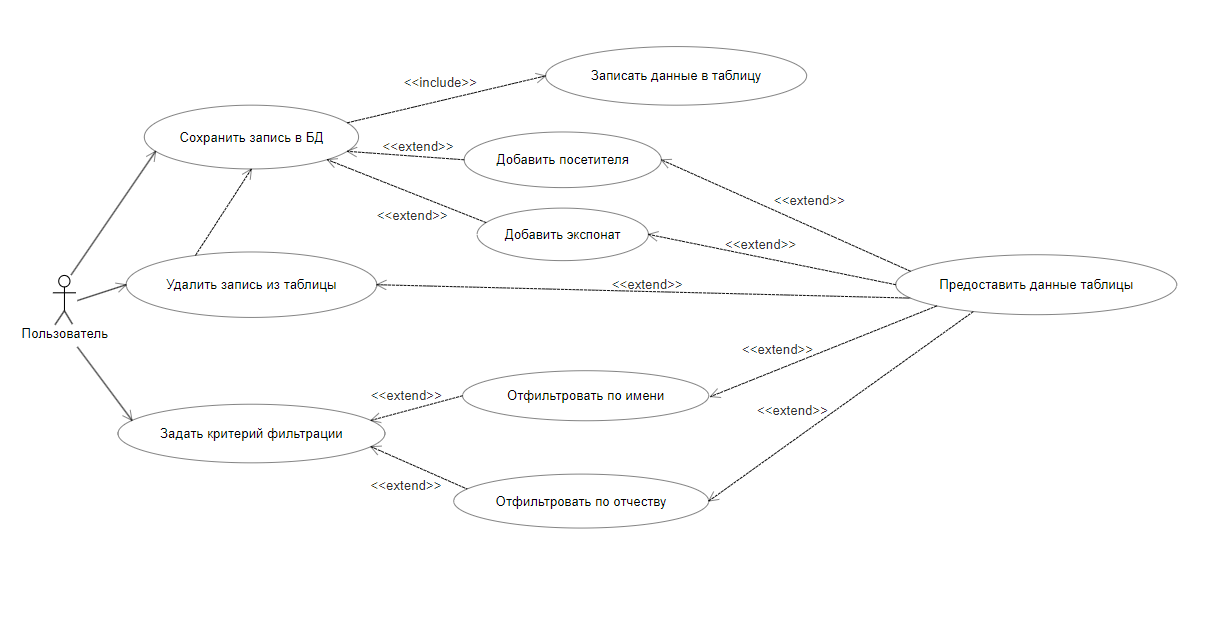


Рисунок 2 – Диаграмма вариантов использования

Из этого рисунка видно, что в системе присутствуют 2 актера – пользователь и программа. Пользователь может осуществлять редактирование записей таблиц базы данных (их добавление, удаление и изменение) и задавать критерии фильтрации. Программа должна предоставлять данные пользователю и сохранять их изменения.

В таблицах 1 и 2 представлены спецификации вариантов использования «Отфильтровать записи таблицы в БД» и «Удалить запись в БД».

Таблица 1 – Спецификация варианта использования «Отфильтровать записи в БД»

|  |
| --- |
| Прецедент: отфильтровать записи таблицы БД |
| ID: 1 |
| Краткое описание: Пользователь вводит критерии фильтрации таблицы БД, и система фильтрует записи |
| Главные актёры: Пользователь |
| Предусловия: Данные из Базы Данных загружены в систему |
| Основной поток:   1. Пользователь задает критерий фильтрации записей 2. Система производит фильтрацию записей 3. Если записи, удовлетворяющие критерию, существуют    1. Система выводит найденные записи |
| Постусловия: Выведены отфильтрованные записи |

Таблица 2 – Спецификация варианта использования «Удалить запись в БД»

|  |
| --- |
| Прецедент: удалить запись из БД |
| ID: 2 |
| Краткое описание: Пользователь выбирает запись для удаления и удаляет ее |
| Главные актёры: Пользователь |
| Предусловия: Данные из Базы Данных загружены в систему |
| Основной поток:   1. Пользователь вводит номер записи 2. Пользователь нажимает кнопку «Удалить» 3. Система удаляет запись |
| Постусловия: Запись удалена |

Рисунок 3 изображает диаграмму последовательности для прецедента «Отфильтровать записи в БД»

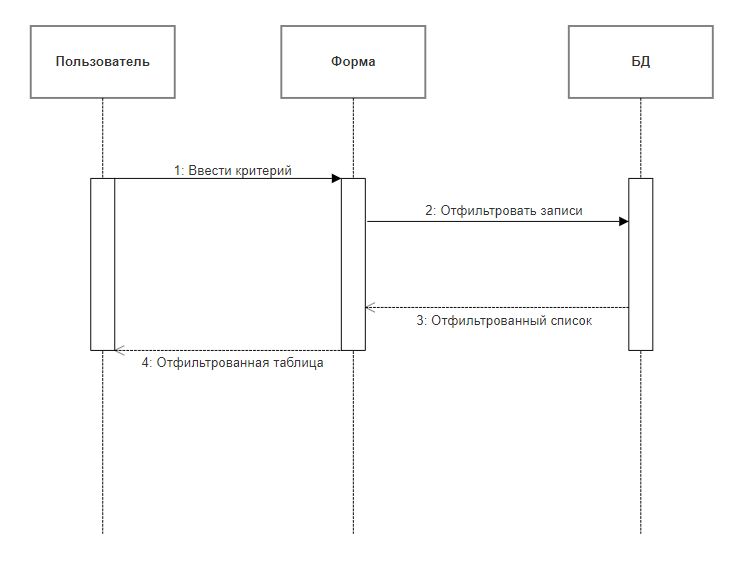


Рисунок 3 – Диаграмма последовательности варианта использования «Отфильтровать записи в БД»

## 2.2. Определение структуры файла базы данных

В разрабатываемом приложении база данных должна включать следующую информацию о пациентах: имя, фамилия, отчество, возраст, район проживания, дата обращения, врач, к которому планируется запись, и диагноз.

# 3. Описание разработки программы

На основе анализа вариантов использования и функциональных требований был разработан пользовательский интерфейс, представленный на рисунке 4-9.

Рисунок 4 – Пользовательский интерфейс

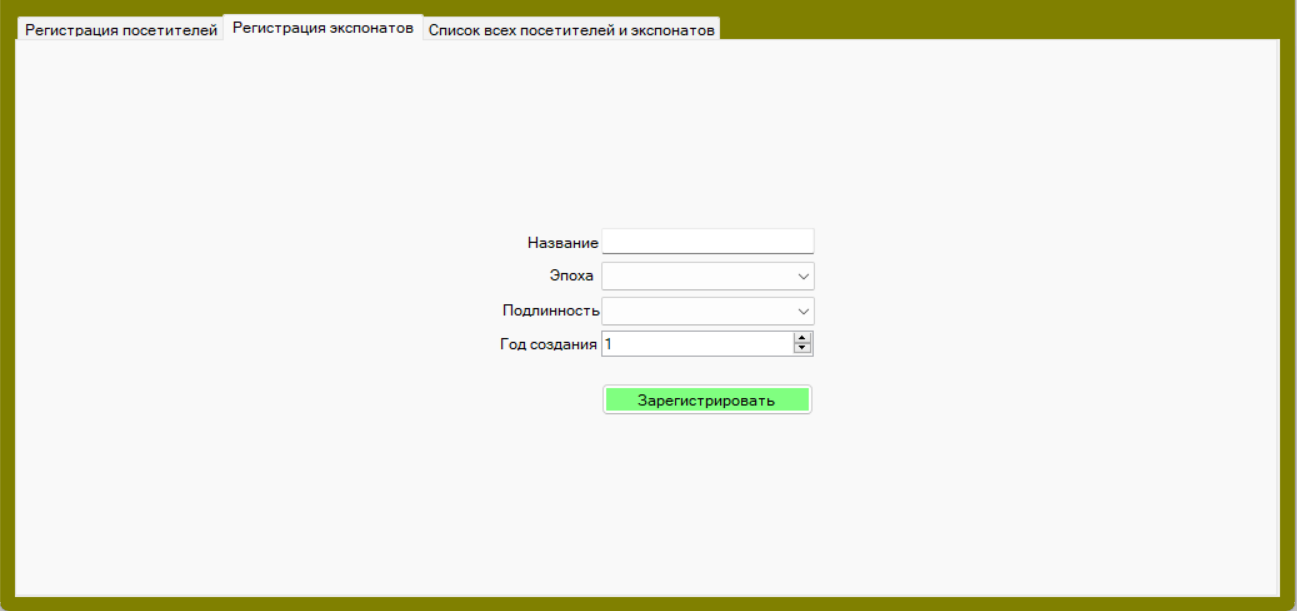
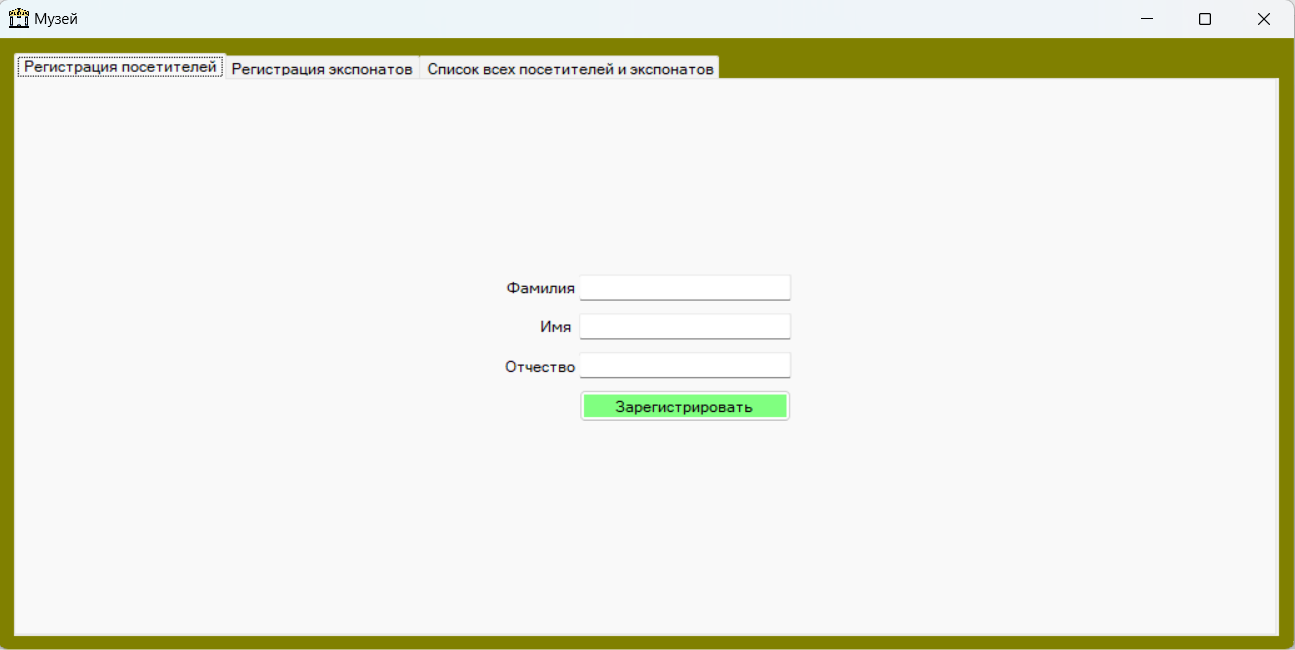


Рисунок 5 – Пользовательский интерфейс

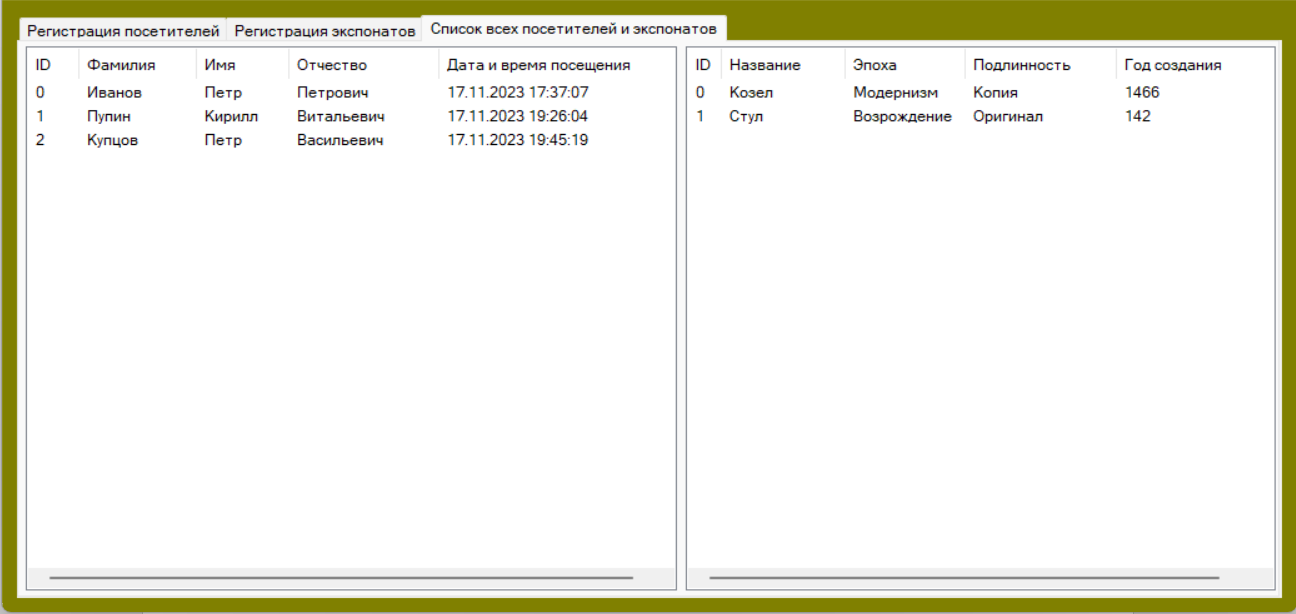


Рисунок 6 – Пользовательский интерфейс



Рисунок 7 – Пользовательский интерфейс

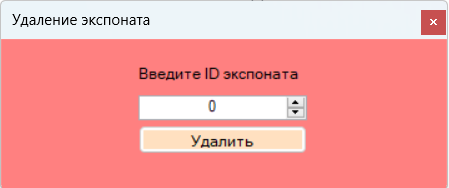


Рисунок 8 – Пользовательский интерфейс

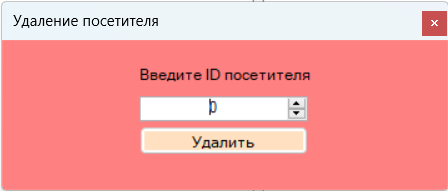


Рисунок 9 – Пользовательский интерфейс

На рисунке 4 представлена форма регистрации новых пользователей. На рисунке 5 представлена форма регистрации новых экспонатов. На рисунке 6 изображены две таблицы, отображающие содержимое баз данных «Посетители» и «Экспонаты». На рисунке 7 отображены критерии фильтрации, а на рисунках 8,9 – формы удаления записей и соответствующих баз данных.

Исходя из модели предметной области и анализа вариантов использования, можно предложить диаграмму классов, представленную на рисунке 10.

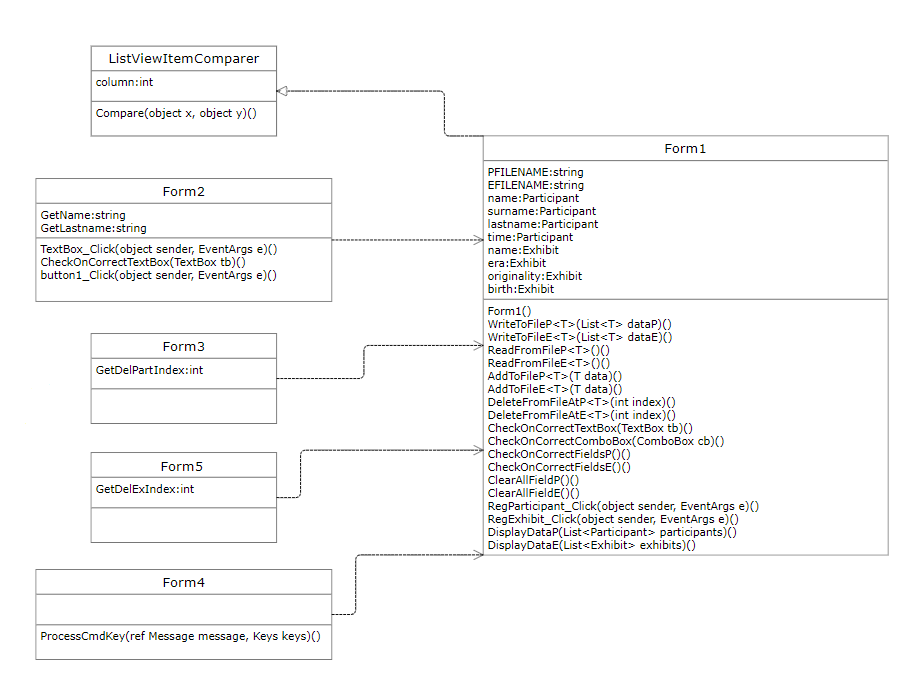


Рисунок 10 – Диаграмма классов

**Название класса:** **Form1**

**Назначение**: класс главной формы приложения

**Члены класса:**

name Participant – поле таблицы имя;

surname Participant - поле таблицы фамилия;

lastname Participant - поле таблицы отчество;

time Participant - поле таблицы дата и время посещения;

name Exhibit - поле таблицы название экспоната;

era Exhibit - поле таблицы эпоха;

originality Exhibit - поле таблицы оригинальность;

birth Exhibit - поле таблицы дата создания;

PFILENAME string – переменная для работы с файлом посетителей.

EFILENAME string – переменная для работы с файлом экспонатов.

**Функции класса:**

public Form1() – инициализация формы;

async Task WriteToFileP<T>(List<T> data) – запись в файл посетителей;

async Task WriteToFileE<T>(List<T> data) – запись в файл экспонатов;

async Task<List<T>> ReadFromFileP<T>() – чтение из файла посетителей;

async Task<List<T>> ReadFromFileE<T>() – чтение из файла экспонатов;

async Task AddToFileP<T>(T data) – добавление записи в файл посетителей;

async Task AddToFileE<T>(T data) – добавление записи в файл экспонатов;

async Task DeleteFromFileAtP<T>(int index) – удаление записи о посещении по ее номеру;

async Task DeleteFromFileAtE<T>(int index) – удаление записи об экспонате по ее номеру;

bool CheckOnCorrectTextBox(TextBox tb) – проверка на ввод корректных данных в текстовые поля;

bool CheckOnCorrectComboBox(ComboBox cb) – проверка на выбор данных из списка;

bool CheckOnCorrectFieldsP – проверка на корректное заполнение всех полей при регистрации посетителя;

bool CheckOnCorrectFieldsE – проверка на корректное заполнение всех полей при регистрации экспоната;

void ClearAllFieldP() – отчистка полей ввода после добавления данных в таблицу посетителей;

void ClearAllFieldE() – отчистка полей ввода после добавления данных в таблицу экспонатов;

async void RegParticipant\_Click(object sender, EventArgs e) – добавление записей в таблицу посетителей;

async void RegExhibit\_Click(object sender, EventArgs e) – добавление записей в таблицу экспонатов;

void DisplayDataP(List<Participant> participants) – вывод данных из файла в таблицу посетителей;

void DisplayDataE(List<Exhibit> exhibits) – вывод данных из файла в таблицу экспонатов;

**Название класса: Form2**

**Назначение**: класс, осуществляющий выборку данных

**Члены класса:**

public string GetName – получение имени;

public string GetLastname – получение отчества.

**Функции класса:**

void TextBox\_Click(object sender, EventArgs e) – изменение цвета поля ввода;

bool CheckOnCorrectTextBox(TextBox tb) – проверка на корректный ввод имени и(или) отчества;

void Button1\_Click(object sender, System.EventArgs e) – фильтрация по заданным критериям.

**Название класса:** **Form3**

**Назначение**: класс, позволяющий удалять данные из таблицы посетителей

**Члены класса:**

public int GetDelPartIndex – переменная для записи ID.

**Название класса:** **Form5**

**Назначение**: класс, позволяющий удалять данные из таблицы экспонатов

**Члены класса:**

public int GetDelExIndex – переменная для записи ID.

**Название класса:** **Form4**

**Назначение**: класс, инициализирующий приветственную страницу

**Функции класса:**

protected override bool ProcessCmdKey(ref Message message, Keys keys) – запуск программы при нажатии на любую клавишу.

**Название класса:** **ListViewItemComparer**

**Назначение**: класс, осуществляющий сортировку по колонкам

**Члены класса:**

public int column – индекс поля.

**Функции класса:**

public int Compare(object x, object y) – сортировка по значению столбца.

# 4. Отладка и тестирование

Проверка функционала программы была осуществлена с помощью функционального тестирования. Функциональное тестирование основывается на функциях, выполняемых системой, и может проводиться на всех уровнях тестирования (компонентном, интеграционном, системном, приемочном). Как правило, эти функции описываются в требованиях, функциональных спецификациях или в виде случаев использования системы. Функциональное тестирование является одним из ключевых видов тестирования, задача которого — установить соответствие разработанного программного обеспечения (ПО) исходным функциональным требованиям заказчика. Проведение функционального тестирования — процесс, позволяющий проверить способность информационной системы работать в определенных условиях и решать задачи, нужные пользователям.

Основные этапы функционального тестирования:

* подготовка (проводится анализ исходных документов о системе. Задачи по этому этапу выполняются совместно с представителями заказчика);
* проведение (функциональное тестирование проходит вручную по подготовленным заранее тестовым сценариям);

отчет (происходит разработка и согласование отчетов о проведенном тестировании со списком обнаруженных отклонений и рекомендациями по улучшению системы). План тестирования приведен в таблице 3. Ниже на рисунках 11 – 24 приведены результаты работы тестов программы.

Таблица 3 - План тестирования

|  |  |  |
| --- | --- | --- |
| **Состав теста** | **Ожидаемый результат** | **Наблюдаемый результат** |
| **Тест 1**  **Добавить запись в таблицу посетителей** | Запись добавлена в таблицу посетителей | Тест пройден.  Запись отобразилась в таблице (рисунок 11, 13) |
| **Тест 2**  **Добавить запись в таблицу экспонатов** | Запись добавлена в таблицу экспонатов | Тест пройден.  Запись отобразилась в таблице (рисунок 12, 14) |
| **Тест 3**  **Удалить вторую запись из таблицы посетителей** | Запись удалена из таблицы без ошибок | Тест пройден. Запись удалена (рисунок 15, 16) |
| **Тест 4**  **Удалить первую запись из таблицы**  **экспонатов** | Запись удалена из таблицы без ошибок | Тест пройден. Запись удалена (рисунок 17, 18) |
| **Тест 5**  **Отсортировать таблицу по фамилиям в алфавитном порядке** | Записи отсортированы в алфавитном порядке. | Тест пройден. Записи отсортированы (рисунок 19) |
| **Тест 6**  **Удалить несуществующую запись** | Вывод сообщения об ошибке, что такой записи не существует. | Тест пройден. Сообщение вывелось, количество записей не изменено (рисунок 20) |
| **Тест 7**  **Отсортировать таблицу по дате посещения** | Записи отсортированы по дате посещения. | Тест пройден. Записи отсортированы (рисунок 21) |
| **Тест 8**  **Фильтрация записей по введенному имени** | Ввод в окно «Имя» имени «Петр». В таблице отображаются только посетители с таким именем. | Тест пройден. В таблице остались записи с заданным критерием (рисунок 22) |
| **Тест 9**  **Ввод некорректных данных** | Данные не сохраняются. Появляется сообщение об ошибке ввода, некорректные поля подсвечены красным. | Тест пройден. Данные не записались. Сообщение вывелось (рисунки 23, 24) |

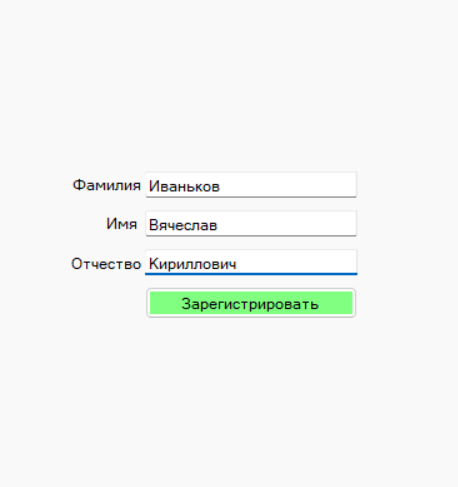


Рисунок 11 – Добавление записи о посетителе

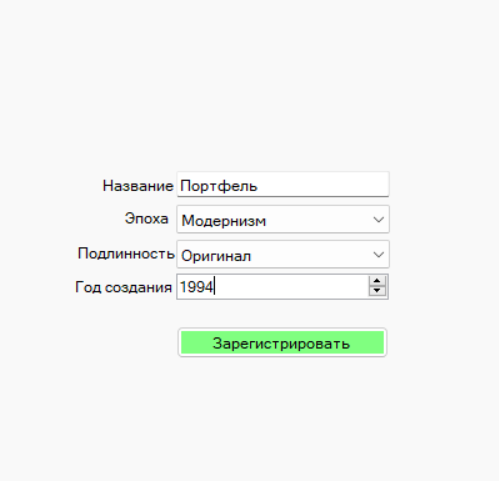


Рисунок 12 – Добавление записи об экспонате

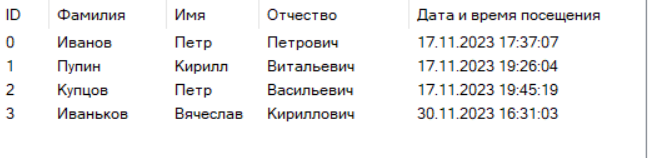


Рисунок 13 – Отображение данных в таблице о посещениях

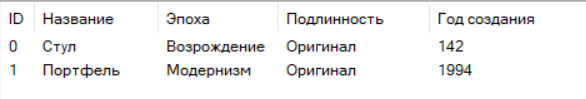


Рисунок 14 – Отображение данных в таблице об экспонатах

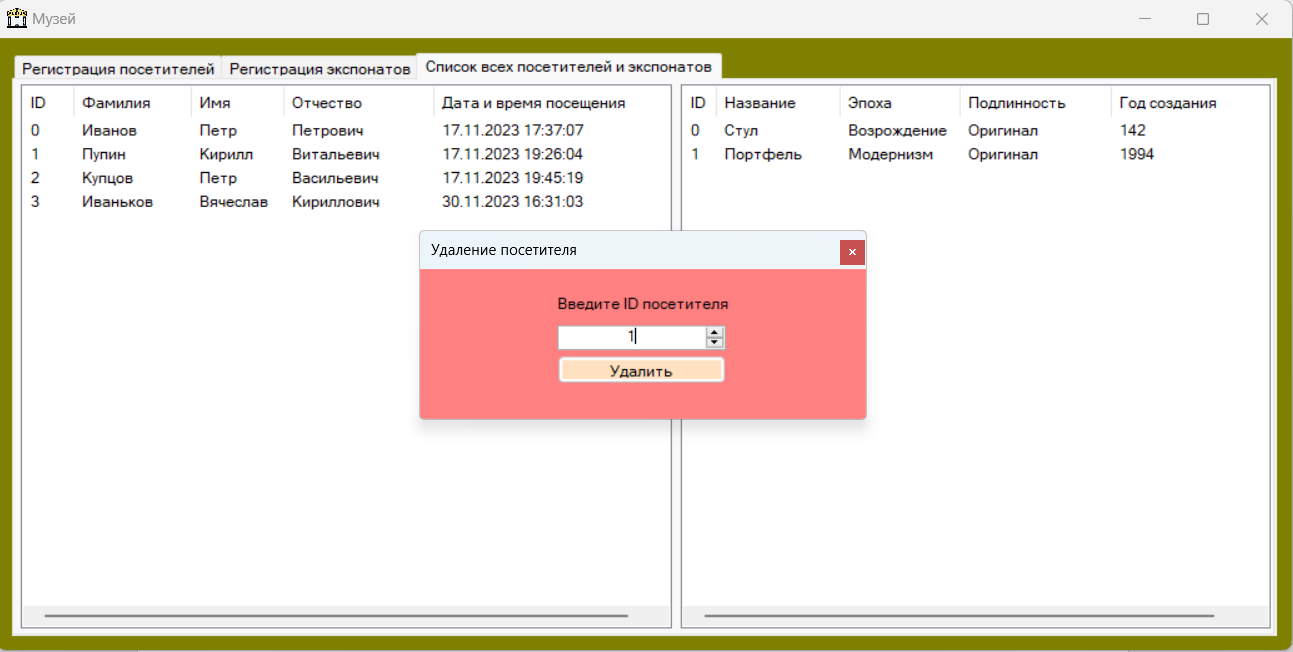


Рисунок 15 – Удаление второй записи в таблице посещений

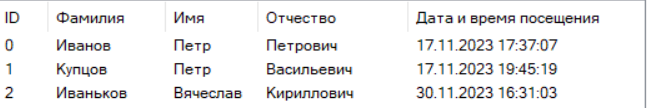


Рисунок 16 – Таблица после удаления записи

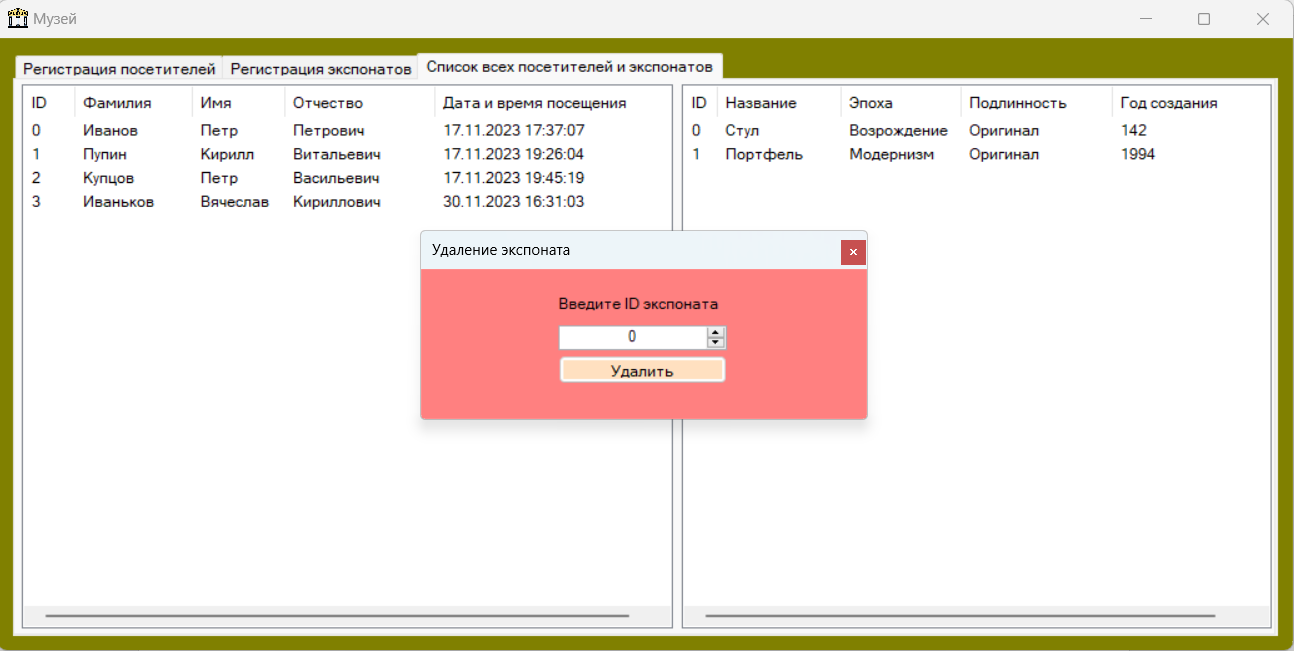


Рисунок 17 – Удаление первой записи в таблице экспонатов

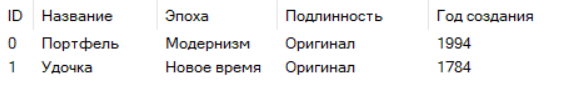


Рисунок 18 – Таблица после удаления записи

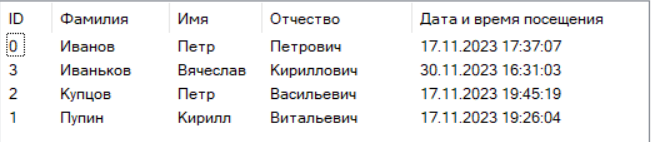


Рисунок 19 – Сортировка данных по фамилиям

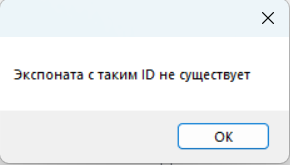


Рисунок 20 – Удаление несуществующей записи

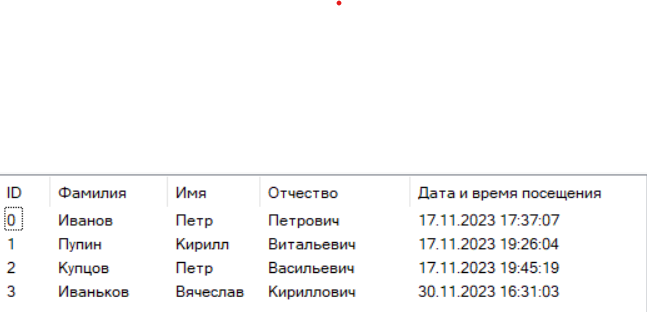


Рисунок 21 – Сортировка по дате посещения

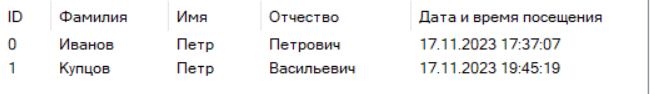


Рисунок 22 – Фильтрация по имени посетителя

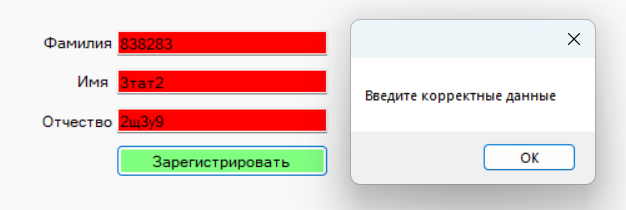


Рисунок 23 – Сообщение о вводе некорректных данных

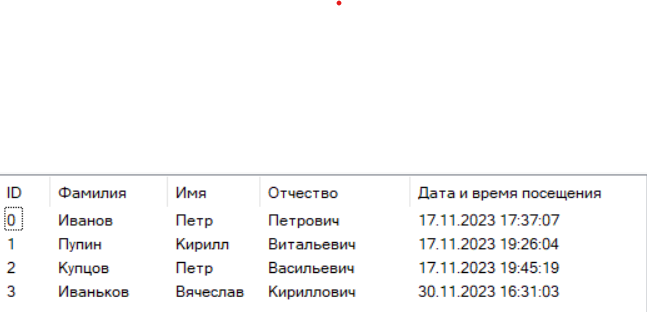


Рисунок 24 – Некорректная запись не записана

В ходе выполнения тестирования программное средство удачно прошло все тесты и показало корректную работу.

# 5. Описание программы

## 5.1 Разработка приложения

Приложение «Музей» является основным модулем программы. При запуске программы происходит вывод заставки - приветствия, после чего программа готова к выполнению своих функций. Пользователь может работать с таблицами и использовать соответствующие пункты меню. В этот момент управление переходит в функции, реализованные в файлах «Form1.cs», «Form2.cs», «Form3.cs», «Form4.cs», «Form5.cs».

## 5.2 Разработка меню

Главное меню приложения включает в себя три вкладки: «Регистрация посетителей», «Регистрация экспонатов» и «Список всех посетителей и экспонатов». В «Регистрации посетителей» представлена форма создания новой записи о посетителе, а также кнопка «Зарегистрировать». В «Регистрации экспонатов» представлена форма создания новой записи об экспонате, а также кнопка «Зарегистрировать». Вкладка «Список всех посетителей и экспонатов» вызывает таблицу с данными, хранящимися в файлах.

# 6. Руководство пользователя

Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой, оно содержит описания шагов, которые необходимо выполнить для достижения пользователем конкретной цели.

Программа «Музей» предназначена для хранения данных о посетителях и экспонатах. Программа имеет интуитивно понятный интерфейс и поддерживает такие операции, как работа с записями, а именно: добавление новой записи, удаление записи, фильтрация записей по имени и отчеству.

Для добавления новой записи о посетителе в таблицу необходимо заполнить значениями поля «Фамилия», «Имя», «Отчество» на вкладке меню «Регистрация посетителя», после чего нажать на кнопку «Зарегистрировать». Новая запись появится в таблице.

Для добавления новой записи об экспонате в таблицу необходимо заполнить значениями поля «Название», «Эпоха», «Подлинность», «Год создания» на вкладке меню «Регистрация», после чего нажать на кнопку «Зарегистрировать». Новая запись появится в таблице.

Для удаления записи о посетителе в таблице нажмите сочетание клавиш Ctrl и P, введите номер записи, которую нужно удалить и нажмите «ОК». Запись о посещении будет удалена.

Для удаления записи об экспонате в таблице нажмите сочетание клавиш Ctrl и E, введите номер записи, которую нужно удалить и нажмите «ОК». Запись об экспонате будет удалена.

Для поиска посетителя по имени и фамилии нужно нажать сочетание клавиш Ctrl и S, внесите нужные вам значения в поле «Имя» и(или) «Отчество» и нажмите «ОК». После этого отобразятся подходящие под заданные критерии записи. Если таких нет – выведется пустая таблица. Чтобы снять фильтр нужно зайти на страницу регистрации и вернуться обратно.

Для сортировки нужно нажать на название столбца, по которому хотите отсортировать.

# Заключение

В ходе выполнения курсового проекта была разработана программа «Музей», позволяющая пользователям работать с базами данных, хранящими записи о посетителях и экспонатах. Приложение предоставляет основные инструменты для следующих действий: добавление записей, фильтрация и сортировка записей, удаление записей. Сконструированное программное средство отвечает всем поставленным требованиям. Результаты тестирования показывают корректную работу приложения. Разработка осуществлялась в среде Visual Studio 2022 на языке С#.

**Список использованных источников**

1. Отношения классов — от UML к коду [Электронный ресурс]. URL: <https://habr.com/ru/post/150041/>[newtonsoft.com](https://vk.com/away.php?to=http%3A%2F%2Fnewtonsoft.com&cc_key=" \t "_blank) (03.06.2022 г.)
2. Асинхронное программирование с использованием ключевых слов async и await [Электронный ресурс]. URL: [docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/](https://vk.com/away.php?to=http%3A%2F%2Fdocs.microsoft.com%2Fru-ru%2Fdotnet%2Fcsharp%2Fprogramming-guide%2Fconcepts%2Fasync%2F&cc_key=) (03.06.2022 г.)
3. Регулярные выражения .NET [Электронный ресурс]. URL: [docs.microsoft.com/ru-ru/dotnet/standard/base-types/regular-expressions](https://vk.com/away.php?to=http%3A%2F%2Fdocs.microsoft.com%2Fru-ru%2Fdotnet%2Fstandard%2Fbase-types%2Fregular-expressions&cc_key=)(03.06.2022 г.)
4. Создание приложения Windows Forms на C# в Visual Studio [Электронный ресурс]. URL: [docs.microsoft.com/ru-ru/visualstudio/ide/create-csharp-winform-visual-studio?view=vs-2022](https://vk.com/away.php?to=http%3A%2F%2Fdocs.microsoft.com%2Fru-ru%2Fvisualstudio%2Fide%2Fcreate-csharp-winform-visual-studio%3Fview%3Dvs-2022&cc_key=) (03.06.2022 г.)
5. Албахари Джозеф, Албахари Бен C# Карманный справочник - М., Издательский дом «Вильямс», «O’REILLY», 2016.
6. Шилдт Герберт, Полный справочник по C#, - М., Издательский дом «Вильямс», 2004.

# **Приложение А**

**КОД ПРОГРАММЫ**

**(обязательное)**

**Файл Form1.cs**

using System;

using System.IO;

using System.Linq;

using System.Drawing;

using Newtonsoft.Json;

using System.Collections;

using System.Windows.Forms;

using System.Threading.Tasks;

using System.Collections.Generic;

using System.Text.RegularExpressions;

namespace CourseWork

{

public partial class Form1 : Form

{

/// <summary>Файл посещений</summary>

const string PFILENAME = "Participants.json";

/// <summary>Файл экспонатов</summary>

const string EFILENAME = "Exhibits.json";

struct Participant

{

/// <summary>Имя посетителя</summary>

public string name;

/// <summary>Фамилия посетителя</summary>

public string surname;

/// <summary>Отчество посетителя</summary>

public string lastname;

/// <summary>Дата и время посещения</summary>

public string time;

}

struct Exhibit

{

/// <summary>Название экспоната</summary>

public string name;

/// <summary>Эпоха, к которой относится</summary>

public string era;

/// <summary>Оригинальность экспоната</summary>

public string originality;

/// <summary>Дата создания</summary>

public string birth;

}

public Form1()

{

Task.Run(() => File.Open(PFILENAME, FileMode.OpenOrCreate).Close());

Task.Run(() => File.Open(EFILENAME, FileMode.OpenOrCreate).Close());

using (var form4 = new Form4())

{

form4.ShowDialog();

}

InitializeComponent();

}

/// <summary>Запись в файл посетителей</summary>

/// <typeparam name="T"></typeparam>

/// <param name="dataP">The data p.</param>

async Task WriteToFileP<T>(List<T> dataP)

{

using (var streamWriterP = new StreamWriter(PFILENAME, false))

{

await streamWriterP.WriteAsync(await Task.Run(() => JsonConvert.SerializeObject(dataP)));

}

}

/// <summary>Запись в файл экспонатов</summary>

/// <typeparam name="T"></typeparam>

/// <param name="dataE">The data e.</param>

async Task WriteToFileE<T>(List<T> dataE)

{

using (var streamWriterE = new StreamWriter(EFILENAME, false))

{

await streamWriterE.WriteAsync(await Task.Run(() => JsonConvert.SerializeObject(dataE)));

}

}

/// <summary>Чтение из файла посетителей</summary>

/// <typeparam name="T"></typeparam>

async Task<List<T>> ReadFromFileP<T>()

{

using (var streamReaderP = new StreamReader(PFILENAME))

{

return await Task.Run(async () => JsonConvert.DeserializeObject<List<T>>(await streamReaderP.ReadToEndAsync()) ?? new List<T>());

}

}

/// <summary>Чтение из файла экспонатов</summary>

/// <typeparam name="T"></typeparam>

async Task<List<T>> ReadFromFileE<T>()

{

using (var streamReaderE = new StreamReader(EFILENAME))

{

return await Task.Run(async () => JsonConvert.DeserializeObject<List<T>>(await streamReaderE.ReadToEndAsync()) ?? new List<T>());

}

}

/// <summary>Добавление записи в файл посетителей</summary>

/// <typeparam name="T"></typeparam>

/// <param name="data">The data.</param>

async Task AddToFileP<T>(T data)

{

var datas = await ReadFromFileP<T>();

datas.Add(data);

await WriteToFileP(datas);

}

/// <summary>Добавление записи в файл экспонатов</summary>

/// <typeparam name="T"></typeparam>

/// <param name="data">The data.</param>

async Task AddToFileE<T>(T data)

{

var datas = await ReadFromFileE<T>();

datas.Add(data);

await WriteToFileE(datas);

}

/// <summary>Удаление записи из файла посетителей</summary>

/// <typeparam name="T"></typeparam>

/// <param name="index">The index.</param>

async Task DeleteFromFileAtP<T>(int index)

{

var datas = await ReadFromFileP<T>();

datas.RemoveAt(index);

await WriteToFileP(datas);

}

/// <summary>Удаление записи из файла экспонатов</summary>

/// <typeparam name="T"></typeparam>

/// <param name="index">The index.</param>

async Task DeleteFromFileAtE<T>(int index)

{

var datas = await ReadFromFileE<T>();

datas.RemoveAt(index);

await WriteToFileE(datas);

}

/// <summary>Проверка корректности ввода в текстовые поля</summary>

/// <param name="tb">The tb.</param>

bool CheckOnCorrectTextBox(TextBox tb) => !Regex.IsMatch(tb.Text, @"^\s\*$") && Regex.IsMatch(tb.Text, @"^[а-яА-Яa-zA-Z ]+$") || (tb.BackColor = Color.Red) != Color.Red;

/// <summary>Проверка выбора значения в выпадающем списке</summary>

/// <param name="cb">The cb.</param>

bool CheckOnCorrectComboBox(ComboBox cb) => !(cb.SelectedItem is null) || (cb.BackColor = Color.Red) != Color.Red;

/// <summary>Проверка на корректность всех полей при регистрации посетителя</summary>

bool CheckOnCorrectFieldsP =>

CheckOnCorrectTextBox(Firstname) &

CheckOnCorrectTextBox(Surname) &

CheckOnCorrectTextBox(Lastname);

/// <summary>Проверка на корректность всех полей при регистрации экспоната</summary>

bool CheckOnCorrectFieldsE =>

CheckOnCorrectTextBox(ExNameBox) &

CheckOnCorrectComboBox(ExEraBox) &

CheckOnCorrectComboBox(ExOriginalityBox);

/// <summary>Очистка всех полей в форме регистрации посетителя</summary>

void ClearAllFieldP()

{

Firstname.Text = Surname.Text = Lastname.Text = default;

}

/// <summary>Очистка всех полей в форме регистрации экспоната</summary>

void ClearAllFieldE()

{

ExNameBox.Text = default;

ExEraBox.Text = ExOriginalityBox.Text = default;

}

/// <summary>Добавление записи о посещение</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

async void RegParticipant\_Click(object sender, EventArgs e)

{

if (CheckOnCorrectFieldsP)

{

await AddToFileP(new Participant

{

name = Firstname.Text,

surname = Surname.Text,

lastname = Lastname.Text,

time = DateTime.Now.ToString()

});

ClearAllFieldP();

MessageBox.Show("Пользователь зарегистрирован");

return;

}

MessageBox.Show("Введите корректные данные");

}

/// <summary>Добавление записи об экспонате</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

async void RegExhibit\_Click(object sender, EventArgs e)

{

if (CheckOnCorrectFieldsE)

{

await AddToFileE(new Exhibit

{

name = ExNameBox.Text,

era = ExEraBox.SelectedItem as string,

originality = ExOriginalityBox.SelectedItem as string,

birth = Convert.ToString(ExBirthBox.Value)

});

ClearAllFieldE();

MessageBox.Show("Экспонат зарегистрирован");

return;

}

MessageBox.Show("Введите корректные данные");

}

/// <summary>Вывод данных о посещениях из файла в таблицу</summary>

/// <param name="participants">The participants.</param>

void DisplayDataP(List<Participant> participants)

{

Display.Items.Clear();

var i = 0;

foreach (var participant in participants)

{

var listViewItemP = new ListViewItem(i.ToString());

listViewItemP.SubItems.Add(participant.surname);

listViewItemP.SubItems.Add(participant.name);

listViewItemP.SubItems.Add(participant.lastname);

listViewItemP.SubItems.Add(participant.time);

Display.Items.Add(listViewItemP);

++i;

}

}

/// <summary>Вывод данных об экспонатах из файла в таблицу</summary>

/// <param name="exhibits">The exhibits.</param>

void DisplayDataE(List<Exhibit> exhibits)

{

ExDisplay.Items.Clear();

var j = 0;

foreach (var exhibit in exhibits)

{

var listViewItemE = new ListViewItem(j.ToString());

listViewItemE.SubItems.Add(exhibit.name);

listViewItemE.SubItems.Add(exhibit.era);

listViewItemE.SubItems.Add(exhibit.originality);

listViewItemE.SubItems.Add(exhibit.birth);

ExDisplay.Items.Add(listViewItemE);

++j;

}

}

/// <summary>Отображение всех данных о посещениях</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

async void Display\_SelectedIndexChangedP(object sender, EventArgs e) => DisplayDataP(await ReadFromFileP<Participant>());

/// <summary>Отображение всех данных об экспонатах</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

async void Display\_SelectedIndexChangedE(object sender, EventArgs e) => DisplayDataE(await ReadFromFileE<Exhibit>());

/// <summary>Класс сортировки</summary>

class ListViewItemComparer : IComparer

{

public int column;

public int Compare(object x, object y) => string.Compare((x as ListViewItem).SubItems[column].Text, (y as ListViewItem).SubItems[column].Text);

}

/// <summary>Сортировка посетителей при нажатии на колонку</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="ColumnClickEventArgs" /> instance containing the event data.</param>

void Display\_ColumnClick(object sender, ColumnClickEventArgs e) => Display.ListViewItemSorter = new ListViewItemComparer { column = e.Column };

/// <summary>Сортировка экспонатов при нажатии на колонку</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="ColumnClickEventArgs" /> instance containing the event data.</param>

void ExDisplay\_ColumnClick(object sender, ColumnClickEventArgs e) => ExDisplay.ListViewItemSorter = new ListViewItemComparer { column = e.Column };

/// <summary>Фильтр по имени и/или отчеству</summary>

async void ShowSearch()

{

using (var form2 = new Form2())

{

if (form2.ShowDialog() == DialogResult.OK)

{

var queryResult =

from participant in await ReadFromFileP<Participant>()

where string.IsNullOrEmpty(form2.GetName) || participant.name == form2.GetName

where string.IsNullOrEmpty(form2.GetLastname) || participant.lastname == form2.GetLastname

select participant;

DisplayDataP(queryResult.ToList());

}

}

}

/// <summary>Удаление записи о посещении</summary>

async void ShowDeleteP()

{

using (var form3 = new Form3())

{

if (form3.ShowDialog() == DialogResult.OK)

{

try

{

await DeleteFromFileAtP<Participant>(form3.GetDelPartIndex);

DisplayDataP(await ReadFromFileP<Participant>());

}

catch (ArgumentOutOfRangeException)

{

MessageBox.Show("Пользователя с таким ID не существует");

}

}

}

}

/// <summary>Удаление записи об экспонате</summary>

async void ShowDeleteE()

{

using (var form5 = new Form5())

{

if (form5.ShowDialog() == DialogResult.OK)

{

try

{

await DeleteFromFileAtE<Exhibit>(form5.GetDelExIndex);

DisplayDataE(await ReadFromFileE<Exhibit>());

}

catch (ArgumentOutOfRangeException)

{

MessageBox.Show("Экспоната с таким ID не существует");

}

}

}

}

/// <summary>Вызов форм фильтрации и удаления </summary>

/// <param name="message">The message.</param>

/// <param name="keys">The keys.</param>

protected override bool ProcessCmdKey(ref Message message, Keys keys)

{

if (tabControl1.SelectedTab == ListOfAllParticipants)

{

switch (keys)

{

case Keys.Control | Keys.E:

ShowDeleteE();

return true;

case Keys.Control | Keys.P:

ShowDeleteP();

return true;

case Keys.Control | Keys.S:

ShowSearch();

return true;

}

}

return base.ProcessCmdKey(ref message, keys);

}

void TextBox\_Click(object sender, EventArgs e) => (sender as Control).BackColor = Color.White;

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

{

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void numericUpDown1\_ValueChanged(object sender, EventArgs e)

{

}

private void Display\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void ExDisplay\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

**Файл Form2.cs**

using System;

using System.Drawing;

using System.Text.RegularExpressions;

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form2 : Form

{

/// <summary>Инициализация формы</summary>

public Form2() => InitializeComponent();

/// <summary>Переменная для записи имени</summary>

public string GetName => ParticipantName.Text;

/// <summary>Переменная для записи отчества</summary>

public string GetLastname => ParticipantLastname.Text;

/// <summary>Проверка на корректность ввода</summary>

/// <param name="tb">The tb.</param>

bool CheckOnCorrectTextBox(TextBox tb) => string.IsNullOrEmpty(tb.Text) || !Regex.IsMatch(tb.Text, @"^\s\*$") && Regex.IsMatch(tb.Text, @"^[а-яА-Яa-zA-Z ]+$") || (tb.BackColor = Color.Red) != Color.Red;

/// <summary>Функция изменения цвета поля ввода</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

void TextBox\_Click(object sender, EventArgs e) => (sender as TextBox).BackColor = Color.White;

/// <summary>Фильтрация по заданным критериям</summary>

/// <param name="sender">The source of the event.</param>

/// <param name="e">The <see cref="EventArgs" /> instance containing the event data.</param>

void Button1\_Click(object sender, EventArgs e) => DialogResult = CheckOnCorrectTextBox(ParticipantName) & CheckOnCorrectTextBox(ParticipantLastname) ? DialogResult.OK : DialogResult.None;

}

}

**Файл Form3.cs**

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form3 : Form

{

/// <summary>Инициализация формы</summary>

public Form3() => InitializeComponent();

/// <summary>Переменная для записи ID</summary>

public int GetDelPartIndex => (int)DelPartIndex.Value;

}

}

**Файл Form4.cs**

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form4 : Form

{

/// <summary>Инициализация формы</summary>

public Form4() => InitializeComponent();

/// <summary>Запуск программы при нажатии любой клавиши</summary>

/// <param name="message">The message.</param>

/// <param name="keys">The keys.</param>

protected override bool ProcessCmdKey(ref Message message, Keys keys) => (DialogResult = DialogResult.OK) == DialogResult.OK;

}

}

**Файл Form5.cs**

using System.Windows.Forms;

namespace CourseWork

{

public partial class Form5 : Form

{

/// <summary>Инициализация формы</summary>

public Form5() => InitializeComponent();

/// <summary>Переменная для сохранения ID</summary>

public int GetDelExIndex => (int)DelExIndex.Value;

}

}