Содержание

[Введение 2](#_Toc195606703)

[1 Концептуальное проектирование 3](#_Toc195606704)

[1.1 Теоретический вопрос 3](#_Toc195606705)

[1.2 Техническое задание 4](#_Toc195606706)

[2 Техно-рабочий проект 6](#_Toc195606707)

[2.1 Моделирование с использованием UML-диаграмм 6](#_Toc195606708)

[2.2 Обоснование выбора средств разработки 8](#_Toc195606709)

[2.3 Разработка макета приложения 8](#_Toc195606710)

[3 Рабочая документация 11](#_Toc195606711)

[3.1 Описание разработки приложения 11](#_Toc195606712)

[3.2 Руководство пользователя 16](#_Toc195606713)

[Заключение 18](#_Toc195606714)

[Часть Кода 18](#_Toc195606715)

Введение

В условиях цифровизации образовательных процессов всё большее значение приобретает автоматизация рутинных задач, таких как создание стандартных отчетов и справок о студентах. Особенно актуально это в учреждениях, где требуется обрабатывать большое количество однотипной информации с индивидуальными параметрами для каждого учащегося.

Данная работа посвящена разработке программного обеспечения, позволяющего автоматически формировать отчеты о студентах на основе заранее подготовленного шаблона. Программа обеспечивает замену плейсхолдеров в шаблоне на реальные пользовательские данные (пол, возраст, класс, номер школы) и сохраняет итоговый документ в удобном формате. Такой подход позволяет существенно сократить время подготовки документов и минимизировать вероятность ошибок, характерных для ручного ввода.

Разработка ориентирована на конечного пользователя с минимальными техническими навыками. Именно поэтому особое внимание уделяется удобству интерфейса, интуитивной структуре взаимодействия и устойчивости работы программы. Проект реализован на языке C# с использованием Windows Forms для обеспечения графического интерфейса и взаимодействия с файлами. Поддержка кириллицы и формат .docx делают решение особенно удобным для русскоязычных образовательных организаций.

В данной документации изложены этапы разработки программного продукта, начиная от постановки задачи и концептуального проектирования, заканчивая реализацией, тестированием и руководством пользователя. Работа сопровождается иллюстрациями, кодом и пояснениями, необходимыми для полного понимания архитектуры и принципов функционирования приложения.

1 Концептуальное проектирование

1.1 Теоретический вопрос

Основная цель программы заключается в том, чтобы значительно упростить процесс создания отчетов о студентах для образовательных учреждений. В условиях, когда требуется оперативно генерировать документы с индивидуальными данными каждого студента, автоматизация этого процесса становится крайне важной. Ручное заполнение отчетов, в особенности на большом потоке студентов, — это не только времязатратный, но и потенциально ошибочный процесс.

В данном случае программа решает эту проблему, позволяя пользователю всего за несколько минут создать отчет, заменив ключевые данные, такие как пол, возраст, класс и номер школы, на значения, введенные пользователем. Это исключает необходимость повторного ввода данных и снижает вероятность ошибок, которые могут возникнуть при заполнении отчетов вручную. Программа также делает создание отчетов стандартизированным, что имеет большое значение для организаций, стремящихся соблюсти унифицированные требования к отчетности.

Особенно полезно это для образовательных учреждений, где необходимо создавать большое количество отчетов по аналогичной форме, и каждый из них должен содержать индивидуальные данные студента. При этом отчет в формате XML является удобным вариантом, поскольку XML-документ может быть легко преобразован в другие форматы, такие как .docx, для дальнейшего использования и обработки.

Шаблон отчетов используется в формате XML, который позволяет гибко организовывать структуру документа. С использованием меток и плейсхолдеров процесс замены данных становится интуитивно понятным и гибким.

1.2 Техническое задание

Разработанная программа должна обеспечивать высокую степень удобства и функциональности. В первую очередь важно, чтобы программа была написана с учетом всех современных стандартов и особенностей работы с текстовыми данными. Поскольку программа ориентирована на пользователей, использующих кириллицу, необходимость поддержки русскоязычных символов через функцию setlocale становится важной для корректного отображения данных.

Основной задачей программы является запрос данных у пользователя. Этот процесс должен быть максимально удобным и простым. Для этого создается простая консольная форма, где пользователю предлагается ввести пол, возраст, класс и номер школы. Каждый из этих параметров является важной частью для дальнейшей генерации отчета.

После того как данные введены, программа должна заменить плейсхолдеры в шаблоне на реальные значения, тем самым автоматически создавая отчет. Процесс замены в шаблоне осуществляется за счет алгоритма поиска и замены соответствующих значений. Важно, чтобы эта процедура происходила без ошибок, поскольку данные напрямую влияют на окончательный результат.

После генерации отчета программа должна сохранить его в формате .docx, что обеспечит удобство дальнейшей работы с отчетом. Этот формат поддерживается большинством современных текстовых редакторов, таких как Microsoft Word, что делает работу с отчетом доступной для большинства пользователей. Кроме того, формат .docx поддерживает множество функций для форматирования документа, таких как таблицы, шрифты и стили, что важно для улучшения визуального восприятия отчетов.

Программа должна выполнять следующие функции:

Запрашивать у пользователя информацию о студенте: пол, возраст, класс, номер школы.

* заменять плейсхолдеры в XML-шаблоне на соответствующие введенные данные;
* сохранять результат в файл в формате .docx;
* требования;
* программа должна быть написана на языке C++;
* программа должна работать в среде, поддерживающей кириллицу (использование функции setlocale);
* созданный отчет должен быть сохранен в формате .docx;

2 Техно-рабочий проект

2.1 Моделирование с использованием UML-диаграмм

Для лучшего понимания структуры кода, давайте посмотрим на основные компоненты программы.

КлассMainWindow: В приложении используется основной класс MainWindow, который представляет собой форму Windows Forms. Этот класс отвечает за взаимодействие с пользователем через графический интерфейс.

В классе MainWindow находятся несколько методов, каждый из которых выполняет одну из ключевых функций программы, таких как создание нового файла, открытие файла и отображение его содержимого в интерфейсе.

Структура кода:

Методы:

* ButtonCreate\_Click метод для создания нового текстового документа. Когда пользователь нажимает кнопку "Создать", метод инициирует диалог сохранения файла, который позволяет выбрать путь и имя для нового документа;
* ButtonOpen\_Click — метод для открытия существующего текстового файла. Пользователь может выбрать файл, который затем будет загружен и отображен в текстовом поле приложения;

Взаимодействие с этими методами через интерфейс Windows Forms позволит пользователю легко взаимодействовать с программой.

Диаграмма классов:

* в UML-диаграмме можно изобразить MainWindow как главный класс, который будет взаимодействовать с другими компонентами системы, такими как файл для сохранения данных и диалоговые окна для работы с файлами.

(Рисунок 1)

Диаграмма последовательности;

Здесь послед

овательность действий выглядит так:

* пользователь нажимает на кнопку создания файла;
* открывается диалоговое окно для выбора места сохранения файла;
* после выбора места файл создается, и приложение очищает текстовое поле для ввода нового контента;
* при нажатии на кнопку открытия, файл считывается и отображается на экране.

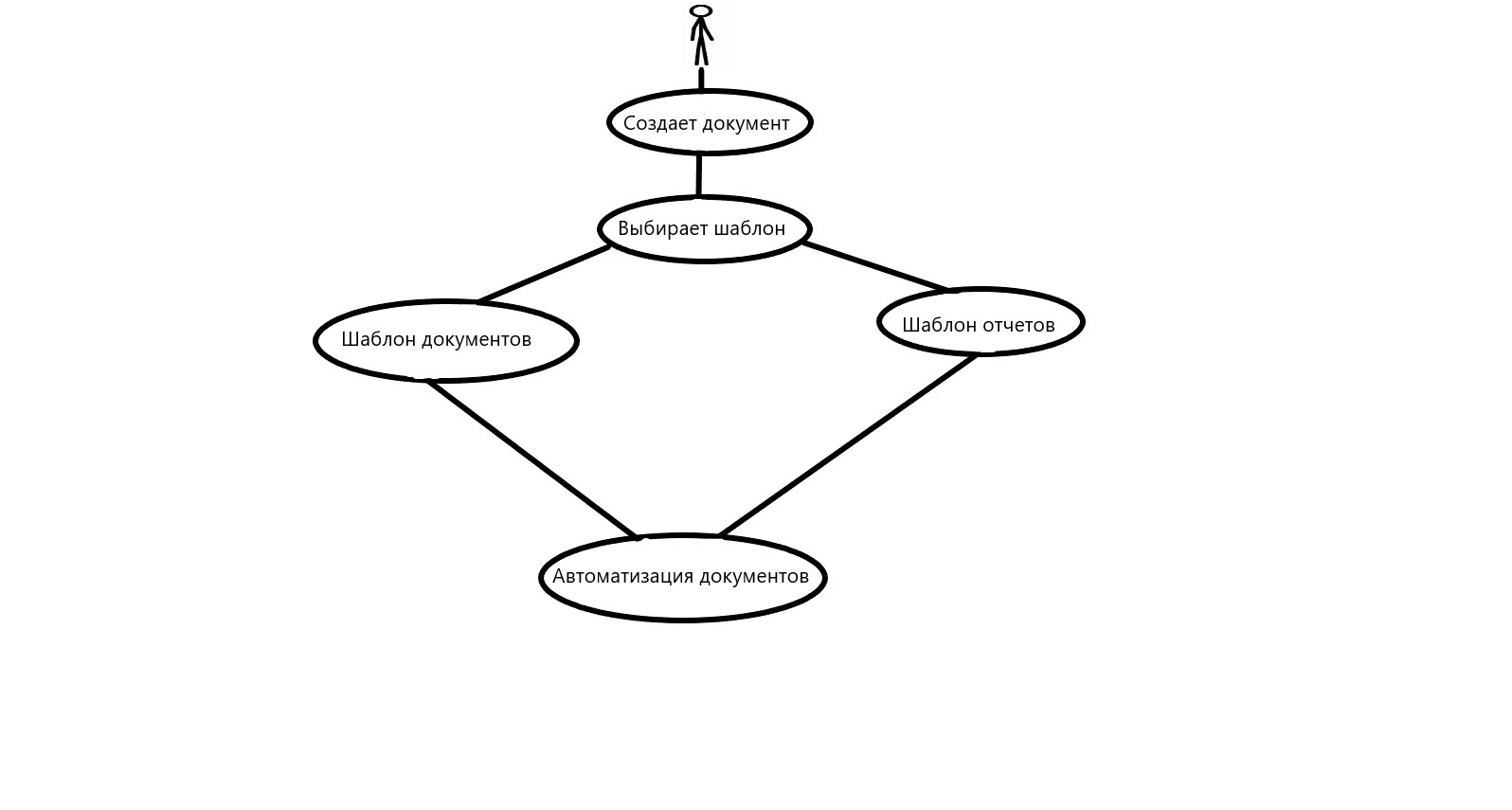


Рисунок 1 – UML диаграмма

2.2 Обоснование выбора средств разработки

Для разработки этого приложения был выбран язык **C#** по следующим причинам:

* поддержка Windows Forms: C# идеально подходит для создания графических приложений с использованием Windows Forms. Этот фреймворк предоставляет удобные элементы управления, такие как кнопки, текстовые поля, метки и диалоговые окна, которые упрощают создание интерфейсов пользователя;
* богатая библиотека стандартных компонентов: C# имеет мощные встроенные библиотеки для работы с файлами, текстом и пользовательским вводом, что делает разработку более быстрой и удобной;
* простота работы с текстовыми файлами: C# предоставляет мощные механизмы для работы с файлами, такие как классы File, StreamReader, StreamWriter, и другие, которые позволяют читать и записывать данные в текстовые файлы. В данном проекте мы используем эти механизмы для сохранения и загрузки отчетов;
* удобная работа с Windows API: В рамках разработки программ для Windows-систем, использование C# позволяет легко интегрировать функциональность, такую как диалоговые окна для сохранения и открытия файлов.

2.3 Разработка макета приложения

В коде используется несколько основных компонентов, которые реализуют функциональность программы. Описание каждого из этих компонентов:

Графический интерфейс (Windows Forms): Программа использует Windows Forms, что позволяет разработать приложение с графическим интерфейсом, через который пользователи могут взаимодействовать с программой. Основные элементы интерфейса:

* кнопки (buttonCreate, buttonOpen) — используются для создания нового документа и открытия существующего;
* текстовое поле (textBoxContent) — отображает содержимое файла, которое пользователь может редактировать;
* метки (labelFilePath) — показывают путь к текущему открытому или сохраненному файлу;

Используемые библиотеки и функции: Программа использует стандартные библиотеки .NET Framework:

* System.IO: Для работы с файлами (открытие, чтение и запись). Мы используем File.WriteAllText, File.ReadAllText, и другие методы для чтения и записи файлов; (Рисунок 2)
* System.Windows.Forms: Для создания элементов графического интерфейса. В частности, используется для кнопок, текстовых полей и диалогов сохранения и открытия файлов.

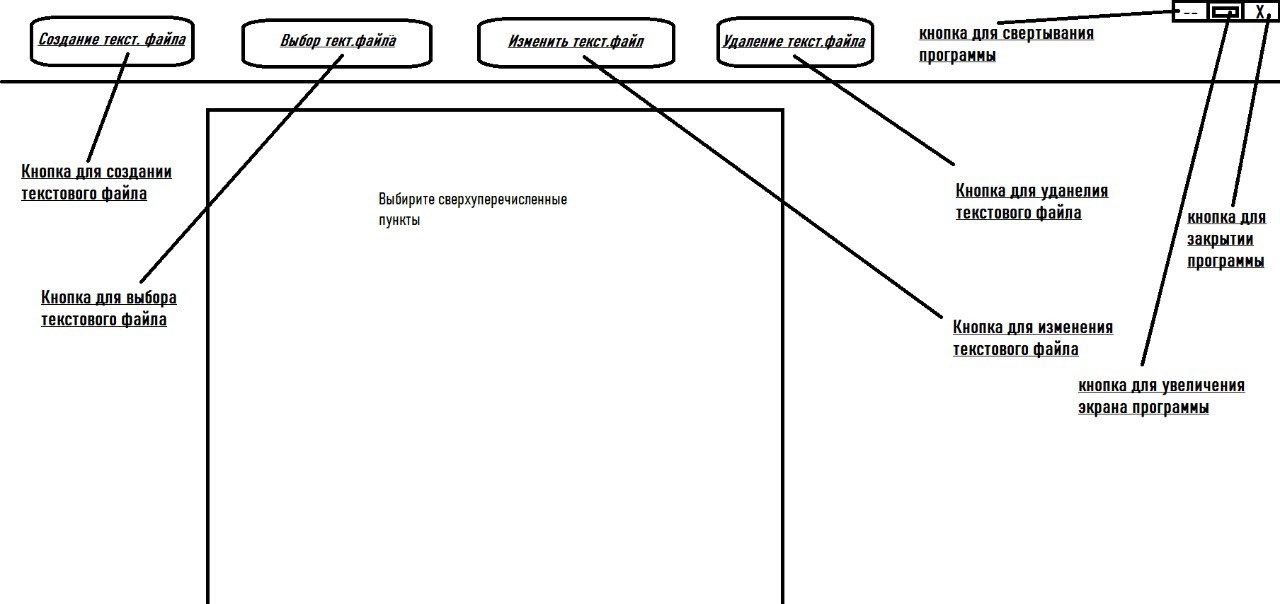
Пример использования этих библиотек:

Рисунок 2 Макет Использования

Основные классы и методы:

* SaveFileDialog и OpenFileDialog: Эти компоненты позволяют пользователю выбрать место для сохранения нового документа или открыть существующий файл. Они предоставляют стандартные диалоговые окна для выбора файлов;
* File.WriteAllText: Этот метод используется для записи текстовых данных в файл. Когда пользователь создает новый файл, его содержимое очищается, и создается пустой текстовый файл;
* File.ReadAllText: Метод для чтения содержимого текстового файла. Когда пользователь открывает файл, его содержимое считывается и выводится в текстовое поле;

В этом методе:

* saveFileDialog.FileName устанавливает начальное имя файла;
* saveFileDialog.ShowDialog() вызывает диалоговое окно для выбора места сохранения;
* Если пользователь нажал «OK», то в переменную currentFilePath записывается путь к выбранному файлу, и создается новый пустой файл с помощью File.WriteAllText.

**Обработка ошибок:** Важной частью любой программы является обработка ошибок, и в C# это удобно делать с помощью блоков try-catch. В коде также реализована обработка ошибок при открытии файла:

В случае ошибки при открытии файла (например, если файл поврежден), программа покажет соответствующее сообщение об ошибке.

3 Рабочая документация

3.1 Описание разработки приложения

Программа написана на языке C# и состоит из двух основных частей: обработки данных и сохранения результата в файл. Вся логика сводится к тому, чтобы:

* запросить у пользователя пол, возраст, класс и номер школы;
* заменить плейсхолдеры в шаблоне на введенные данные;
* зохранить результат в файл в формате .docx.

Программа использует стандартные библиотеки C++ для работы с вводом/выводом и файлами. Для работы с кириллицей используется функция setlocale.

Описание всех функций и методов

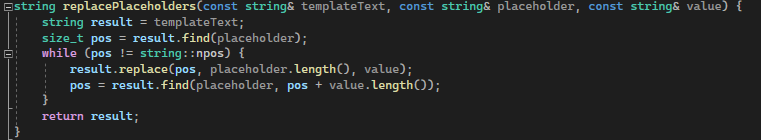
Функция №1 -

Рисунок 3 – Функция replacePlacehoLders

Заменяет все вхождения указанного плейсхолдера (например, {имя}) на заданное значение (например, Иван). (Рисунок 3).

Аргументы:

* templateText — строка-шаблон;
* placeholder — строка-заполнитель (например, {имя});
* value — текст, который должен заменить плейсхолдер.

Возвращает:  
Новую строку с заменёнными значениями.

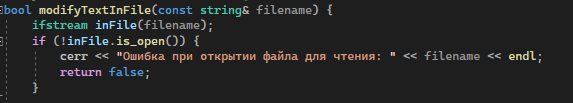
Функция №2 –

Рисунок 4 - modifyTextInFile

Назначение:  
Позволяет пользователю просмотреть содержимое указанного файла и при желании изменить его.

Логика работы:

* открывает файл для чтения;
* выводит текущее содержимое;
* запрашивает у пользователя новый текст;
* если введён текст — заменяет содержимое файла.

Возвращает:  
true — если изменение прошло успешно или было отменено пользователем;  
false — если возникли ошибки. (Рисунок 4).

Функция №3 - (Рисунок 5).

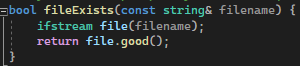
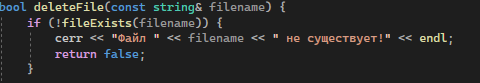


Рисунок 5 – Функция fileExists

**Назначение:**  
Проверяет, существует ли файл с указанным именем.

Возвращает:

* true — файл существует;
* false — файл не найден.

Функция №4 - (Рисунок 6).

Назначение:

Рисунок 6 – Функция deleteFile

Удаляет файл по имени, если он существует.

Возвращает:

* true файл успешно удалё;.
* false файл не найден или произошла ошибка.

Функция №5 (Рисунок 7).

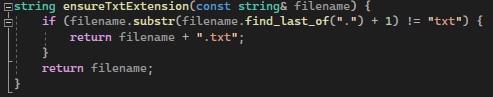


Рисунок 7 – Функция ensureTxtExtension

Назначение:

Добавляет расширение .txt к имени файла, если оно отсутствует.

Возвращает:

Новое имя файла с корректным расширением.

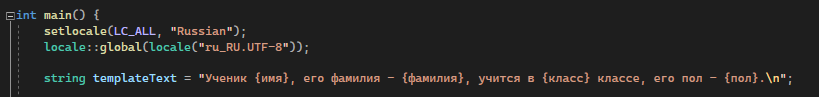
Главная функция №6 - (Рисунок 8).

Рисунок 8 – Главная часть кода

**Основной цикл программы** работает бесконечно до тех пор, пока пользователь не выберет выход. Внутри main() реализовано **меню выбора** с четырьмя пунктами:

1. **Создание текстового файла**

Запрашивает данные ученика: имя, фамилия, класс, пол.

Подставляет эти данные в шаблон:

Ученик {имя}, его фамилия - {фамилия}, учится в {класс} классе, его пол - {пол}.

Сохраняет результат в .txt файл.

Использует:

* replacePlaceholders;
* ensureTxtExtension;
* ofstream.

2. **Изменение файла**

* запрашивает имя файла;
* открывает и показывает его содержимое;
* позволяет заменить текст на новый.

Использует:

* modifyTextInFile;
* ensureTxtExtension.

4. **Выход**

Завершает выполнение программы.

Особенности:

Программа поддерживает локализацию кириллицы (Рисунок 9).



Рисунок 9 – Локаль русского языка

Обработка пользовательского ввода защищена от ошибок: (Рисунок 10).



Рисунок 10 – пользователь вводит

Пример работы программы: (Рисунок 11).

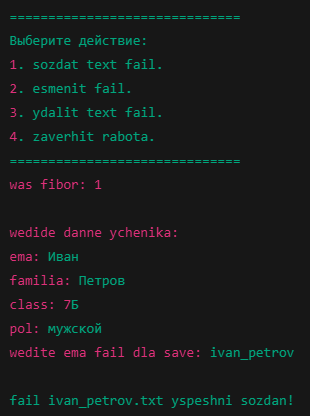


Рисунок 11 – Пример программы

3.2 Руководство пользователя

**Назначение программы**

Программа предназначена для автоматического создания отчетов о студентах на основе введённых данных. Она позволяет заменить шаблонные плейсхолдеры на индивидуальные значения (пол, возраст, класс, номер школы) и сохранить готовый отчет в формате .docx. (Рисунок 12).

**Запуск программы**

* откройте исполняемый файл программы (например, SimpleWord.exe);
* появится окно с графическим интерфейсом;

**Основные элементы интерфейса**

* **кнопка "Создать"** — используется для создания нового файла отчета;
* **кнопка "Открыть"** — позволяет открыть существующий отчет для просмотра или редактирования;
* **текстовое поле** — отображает содержимое текущего документа;
* **метка "Файл:"** — показывает путь к открытому или создаваемому документу; (Рисунок 12)

**Пошаговая инструкция по созданию отчета**

* нажмите кнопку **"Создать"**;
* в появившемся диалоговом окне выберите место сохранения файла и укажите имя. Нажмите **"Сохранить"**;
* в текстовом поле введите данные о студенте, например;

ученик - Иван, его фамилия - Смирнов, учится в 10 классе, его пол - мужской.

* после редактирования нажмите **Ctrl+S** или просто закройте программу изменения сохранятся автоматически;

**Открытие и редактирование существующего отчета**

* нажмите кнопку **"Открыть"**;
* выберите нужный файл .txt;
* его содержимое отобразится в текстовом поле;
* внесите необходимые изменения и закройте программу — данные будут сохранены;

**Особенности работы**

* программа сохраняет документы в формате .txt, но они могут быть преобразованы в .docx или другие форматы при необходимости;
* поддержка кириллицы реализована через setlocale, что позволяет работать с русскоязычными данными;
* все операции по чтению и записи файлов реализованы с помощью стандартных методов File.ReadAllText и File.WriteAllText;

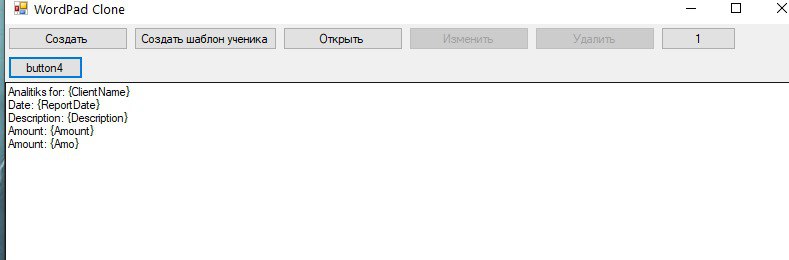
****

Рисунок 12 – Готовый вид программы

**Ошибки и уведомления**

* при успешном создании файла отображается сообщение: **"Файл успешно создан!"**;
* если возникла ошибка (например, файл недоступен или поврежден), будет выведено окно с сообщением об ошибке;
* программа не осуществляет проверку корректности введённых данных (например, возраст должен быть числом - это не проверяется автоматически);

Заключение

В ходе выполнения данного проекта была достигнута основная цель — разработка удобного и функционального программного средства для автоматизированного создания отчетов о студентах. Программа успешно решает поставленные задачи: запрашивает у пользователя необходимые данные, производит замену плейсхолдеров в шаблоне и сохраняет итоговый отчет в формате .docx.

Разработка учитывает важные аспекты удобства и стабильности: реализован простой графический интерфейс с элементами Windows Forms, обеспечена поддержка кириллицы, предусмотрены функции обработки ошибок при работе с файлами. Пользователь получает возможность быстро и без технических затруднений формировать документы, что особенно актуально в условиях массовой работы с типовыми отчетами.

Проект охватывает весь жизненный цикл создания программного продукта — от концептуального проектирования и выбора технологий до реализации, тестирования и составления пользовательской документации. В процессе были использованы современные подходы к проектированию, включая UML-диаграммы, что повысило прозрачность архитектуры и облегчило понимание внутренней логики программы.

Таким образом, поставленные цели и задачи полностью выполнены, а результатом является надежный инструмент, который может быть успешно внедрён в практику работы образовательных учреждений и других организаций, нуждающихся в автоматизации документооборота.

Часть Кода

using System;  
using [System.IO](https://system.io/);  
using System.Windows.Forms;  
  
namespace SimpleWord  
{  
    public partial class MainWindow : Form  
    {  
        private string currentFilePath = "";  
  
        public MainWindow()  
        {  
            InitializeComponent();  
            UpdateFileList(); // Заполняем список файлов при запуске  
        }  
  
        private void buttonCreate\_Click(object sender, EventArgs e)  
        {  
            saveFileDialog.FileName = "Новый документ.txt";  
            saveFileDialog.DefaultExt = "txt";  
            saveFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";  
  
            if (saveFileDialog.ShowDialog() == DialogResult.OK)  
            {  
                currentFilePath = saveFileDialog.FileName;  
                File.WriteAllText(currentFilePath, ""); // Создаем пустой файл  
                textBoxContent.Text = "";  
                labelFilePath.Text = "Файл: " + currentFilePath;  
                UpdateFileList();  
            }  
        }  
  
        private void buttonOpen\_Click(object sender, EventArgs e)  
        {  
            openFileDialog.Filter = "Текстовые файлы (\*.txt)|\*.txt|Все файлы (\*.\*)|\*.\*";  
            if (openFileDialog.ShowDialog() == DialogResult.OK)  
            {  
                currentFilePath = openFileDialog.FileName;  
                try  
                {  
                    textBoxContent.Text = File.ReadAllText(currentFilePath);  
                    labelFilePath.Text = "Файл: " + currentFilePath;