Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное   
образовательное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

КАФЕДРА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И СИСТЕМ

Разработка приложения для обработки файла по госту сп 1.701-2010

Проект для хакатона «Грамотное решение»

|  |  |
| --- | --- |
|  | Студентка группы 3091  А.Н. Архангельская  «23» апреля 2025 г. |
|  | Студентка группы 3091  Е.Д. Ельнова  23 апреля 2025 г. |

Содержание

[Введение 4](#_Toc196298300)

[1 Постановка задачи 6](#_Toc196298301)

[1.1 Обоснование необходимости разработки 6](#_Toc196298302)

[1.2 Технико-математическое описание задачи 6](#_Toc196298303)

[1.3 Требования к программе 6](#_Toc196298304)

[1.4 Обоснование проектных решений 7](#_Toc196298305)

[1.4.1 Обоснование выбора языков программирования 7](#_Toc196298306)

[1.4.2 Инструментальные средства 7](#_Toc196298307)

[1.4.3 Обоснование выбора среды программирования 7](#_Toc196298308)

[1.4.4 Информационное обеспечение 8](#_Toc196298309)

[1.5 Актуальные взгляды на выбранную тему 8](#_Toc196298310)

[2 Практическая часть 9](#_Toc196298311)

[2.1 Описание логической структуры 9](#_Toc196298312)

[2.2 Разработка программы для работы 9](#_Toc196298313)

[2.2.1 Входные и выходные данные 10](#_Toc196298314)

[2.2.2 Используемые методы и компоненты 11](#_Toc196298315)

[2.2.3 Инструкция 12](#_Toc196298316)

[2.3 Сравнительная характеристика работы 13](#_Toc196298317)

[Приложение А 14](#_Toc196298318)

[Листинг A.2 – код веб-интерфейса fileUpload.js 17](#_Toc196298319)

[Листинг А.3 – веб-интерфейс literature.js 18](#_Toc196298320)

[Листинг А.4 – веб-интерфейс titlePageForm.js 19](#_Toc196298321)

[Листинг А.5 – сервер на С++ (serverStart.cpp) 22](#_Toc196298322)

[Листинг А.6 – обработка файла (formatted.cpp) 23](#_Toc196298323)

Введение

Темой данного проекта является разработка приложения на языке C++ для автоматизированного формирования документов Microsoft Word (.docx) в полном соответствии с требованиями стандарта СТО 1.701-2010. Актуальность проекта обусловлена необходимостью решения проблем, связанных с ручным оформлением технической и научной документации, которое в настоящее время требует значительных временных затрат и часто сопровождается ошибками форматирования.

Основная цель проекта заключается в создании программного решения, позволяющего автоматически генерировать документы с соблюдением всех требований стандарта СТО 1.701-2010. Разрабатываемое приложение будет обеспечивать: автоматическое создание документов с правильной структурой (включая титульные листы, оглавления, списки и таблицы); точное применение всех необходимых параметров форматирования (шрифтов, отступов, межстрочных интервалов); а также проверку готовых документов на соответствие стандарту.

Целевой аудиторией приложения являются: студенты и аспиранты, готовящие учебные работы; научные сотрудники и преподаватели, оформляющие исследовательские материалы; инженеры и технические специалисты, работающие с технической документацией. Для этих категорий пользователей приложение позволит существенно сократить время подготовки документов и минимизировать вероятность ошибок оформления.

Техническая реализация проекта предусматривает использование современных библиотек для работы с форматом .docx, разработку на языке C++ для обеспечения высокой производительности, а также создание модульной архитектуры для возможного расширения функционала в будущем. Особое внимание будет уделено точному соответствию всем требованиям стандарта СТО 1.701-2010 и удобству пользовательского интерфейса.

Проект будет реализован в несколько этапов:

- анализ требований стандарта;

- исследование доступных технологических решений;

- проектирование архитектуры приложения;

- разработка основных функциональных модулей;

- тестирование и отладка системы.

В результате планируется получить готовое к использованию приложение для Windows с сопутствующей документацией и примерами сгенерированных документов.

Реализация данного проекта позволит существенно упростить процесс подготовки документов для образовательных учреждений, научных организаций и промышленных предприятий, где требуется строгое соблюдение стандартов оформления, а также обеспечит единообразие и высокое качество подготавливаемых документов.

## Постановка задачи

### Обоснование необходимости разработки

Необходимо решить проблемы, связанные с ручным оформлением технической и научной документации, которое в настоящее время требует значительных временных затрат и часто сопровождается ошибками форматирования.

# Технико-математическое описание задачи

Требуется разработать программное решение, позволяющее автоматически генерировать документы с соблюдением всех требований стандарта СТО 1.701-2010. При этом нужно обязательно использовать С++ для решения основной задачи, а вспомогательные части могут не придерживаться этого правила (например, веб-интерфейс).

# Требования к программе

Программа должна работать хотя бы на одной из операционных систем. И выполнять минимальные требования по ТЗ хакатона:  
Формирование титульного листа по стандарту (размеры полей, структура, надзаголовочные и подзаголовочные данные);

- поддержка структурных элементов, таких как аннотация/реферат, содержание, разделы, приложения, список литературы;

- поддержка таблиц (структура, заголовки, переносы, оформление);

- поддержка сносок, ссылок, примечаний, формул и рисунков с правильной нумерацией;

- применение правил оформления текста, такие как отступы, шрифты, поля, междустрочные интервалы, размеры заголовков;

- соблюдение правил нумерации страниц;

- работа со шрифтами, например для обычного текста использование Times New Roman, размер 14, черный цвет, междустрочный интервал — 1.5.

# Обоснование проектных решений

Для создание красивого интерфейса выбран веб, а точнее bootstrap5. С помощью него можно легко и быстро создавать красивые формы. Для разработки серверной и функциональной части был выбран C++, так как это прописано в ТЗ и на этом языке значительно быстрее работает сервер.

# Обоснование выбора языков программирования

Код на С++ выполняется быстрее всего, а веб-интерфейс с использованием JavaScript красочен и мобилен.

# Инструментальные средства

Дополнительные средства, используемые для написания документации, создания презентации и реализации практической части:

- Word ;

- PowerPoint;

- VSCode;

- GitHub;

- Visual Studio.

# Обоснование выбора среды программирования

Visual Studio Code для клиентской части, в этой IDE чаще всего разрабатывают веб-интерфейсы, а также облегчает версионный контроль. Visual Studio использовался для серверной и функциональной, в ней легче всего создать и запустить консольные приложения, в большей части для проведения тестов.

# Информационное обеспечение

Для того, чтобы понимать расположения объектов в приложениии, нужно создать макет в Figma. Это является лучшим выбором для создание макета, так как всё само сохраняется и можно легко сделать красивый прототип с активными элементами. Word для написания записки, PowerPoint для создания презентации, базы данных.

# Актуальные взгляды на выбранную тему

В условиях цифровой трансформации и ужесточения требований к оформлению документации разработка специализированных решений для автоматизированного создания стандартизированных документов становится особенно актуальной. Современные подходы к решению этой задачи развиваются по нескольким ключевым направлениям.

Во-первых, наблюдается постоянное ужесточение требований к стандартизации документации в образовательных, научных и промышленных организациях. Это обусловлено необходимостью обеспечения юридической значимости документов, требованиями к унификации форматов в цифровых архивах, а также международными стандартами научных публикаций. В этом контексте разработка инструментов для автоматического соблюдения норм СТО и ГОСТ приобретает особую важность.

Современные технологические решения в этой области развиваются по пути использования специализированных библиотек (таких как Apache POI, Docx4j, OpenXML SDK), интеграции с системами электронного документооборота (СЭД) и применения облачных сервисов для совместной работы. Особое внимание уделяется поддержке открытых форматов (ODF, OpenXML) и обеспечению кросс-платформенной совместимости.

Перспективным направлением является внедрение элементов искусственного интеллекта в системы обработки документов. Современные исследования демонстрируют эффективность применения машинного обучения для автоматического форматирования, NLP-технологий для анализа содержания и интеллектуальных систем проверки соответствия стандартам.

Архитектурные решения в этой области все чаще строятся по принципам микро-сервисной архитектуры, что обеспечивает модульность, масштабируемость и гибкость разрабатываемых систем. Такой подход позволяет создавать решения, которые могут интегрироваться с различными платформами и адаптироваться к изменяющимся требованиям.

## Практическая часть

# Описание логической структуры

Необходимо сделать веб-интерфейс, включающий в себя: загрузку файла, форму и заполнение списка литературы, затем при нажатии кнопки из index.html делаем запрос на ручку (localehost:8080). Там этот запрос слушает сервер, который принимает отправленные данные. Эти данные передаются в функцию обработки. Там с помощью XML содержимое файла исправляется и дополняется титульным листом, списком литературы и содержанием.

# Разработка программы для работы

Программа представляет собой веб-приложение с клиент-серверной архитектурой, предназначенное для автоматического оформления документов в соответствии с требованиями стандарта СТО 1.701-2010. Клиентская часть реализована как интерактивный веб-интерфейс, включающий модуль загрузки документов (с поддержкой форматов DOCX и RTF), форму ввода метаданных (автор, название, год, тип документа) и редактор списка литературы с возможностью добавления/удаления источников. Интерфейс разработан с использованием современных технологий HTML5, CSS3 и JavaScript (React cdn), обеспечивающих удобное взаимодействие с пользователем.

Серверная часть, работающая на localhost:8080, построена на С++ с использованием cpp-httplib и принимает POST-запросы от клиента. При получении данных сервер выполняет многоэтапную обработку: сначала валидирует входные данные, затем с помощью библиотек python-docx и lxml анализирует и модифицирует XML-структуру документа, добавляя обязательные элементы (титульный лист, оглавление, список литературы) и применяя требуемые стили форматирования.

## Входные и выходные данные

Таблица 1 – Входные и выходные данные formatter.cpp

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Название | Описание | Пример |
| Входные данные | | | |
| const string | docx\_data | Содержимое файла для обработки | Содержимое файла |
| json | metadata | Данные для заполнения титульного листа |  |
| Выходные данные | | | |
| string | ss | Содержимое файла, обновленное после обработки |  |

Этот файл является центром функционала приложения. Он обрабатывает содержимое файла по ГОСТУ.

Таблица 2 – Входные и выходные данные serverStart.cpp

|  |  |  |  |
| --- | --- | --- | --- |
| Тип данных | Название | Описание | Пример |
| Входные данные | | | |
| Текстовый файл | file | Файл для обработки | Курсовик.docx |
| json | metadata | Данные для заполнения титульного листа |  |
| Выходные данные | | | |
| string | result | Исправленный текст |  |

Этот файл является главным. Он запускает сервер, который слушает localhost:8080.

# Используемые методы и компоненты

Таблица 3 – описание компонентов интерфейса

|  |  |
| --- | --- |
| Компонент | Описание |
| App | Основной компонент, который объединяет все три компонента в одну форму. |
| titlePageForm | Форма для получения информации от пользователя (ФИО, номер группы и т.д). |
| fileUpload | Поле для загрузки файла. |
| literature | Элемент список литературы. Состоит из поля ввода и кнопки «Добавить», которая добавляет записанную в поле ввода книгу или источник. |

Таблица 4 – описание функций файла formatted.cpp

|  |  |
| --- | --- |
| Метод | Описание |
| SetPageMargins | Установка полей страницы. |
| SetParagraphSpacing | Установка межстрочного интервала. |
| FormatTables | Обработка текста в таблицах. |
| FormatParagraphs | Выравнивание текста по ширине и очистка форматирования. |
| SetDocumentStyles | Настройка стилей документа. |
| ProcessXml | Обработка XML-файлов. |
| UnzipDocx | Распаковка DOCX. |
| ZipDocx | Упаковка DOCX. |
| format\_document | Основная функция форматирования документа. |

## Инструкция

Загрузка файла.

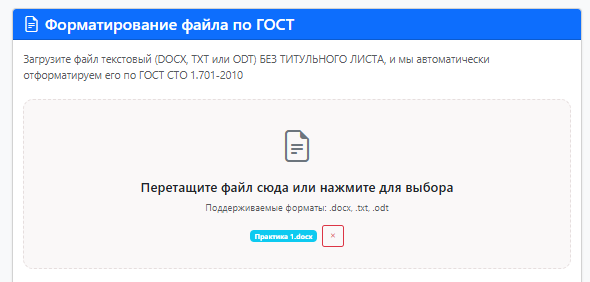


Рисунок 1 – Демонстрация загрузки файла

Заполнение формы.

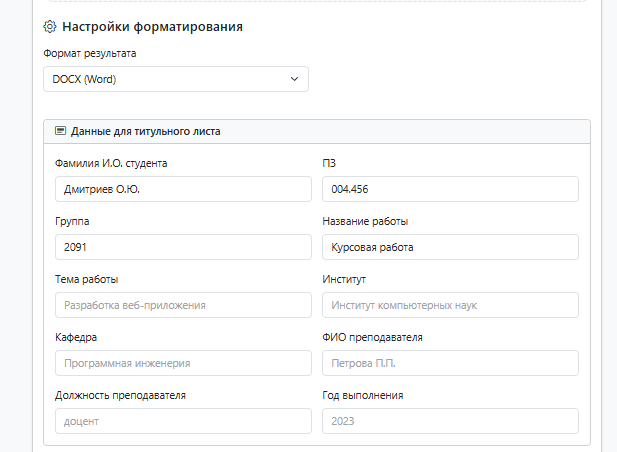


Рисунок 2 – Демонстрация загрузки формы

Записываем список литературы.

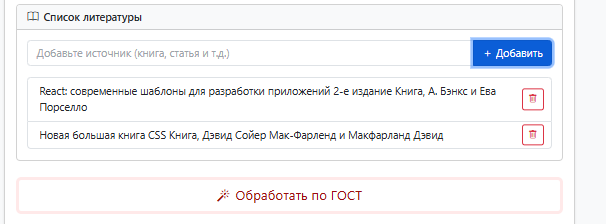


Рисунок 3 – Демонстрация списка литературы

Нажимаем кнопку «Обработать по ГОСТ»

# Сравнительная характеристика работы

Учитывая количество критериев и время, данное на разработку, была выполнена только часть предоставленного ТЗ. Выполненные критерии:

- устанавливает поля страницы (левое 30 мм, правое 10 мм, верх/низ 20 мм);

- задает межстрочный интервал 1.5 строки (360 twips);

- выравнивает основной текст по ширине;

- удаляет жирное выделение и цвет текста в основном тексте;

- устанавливает размер шрифта 12pt (24 twips) для всего текста в таблицах;

- выравнивает текст в таблицах по ширине;

- удаляет жирное выделение и цвет текста в таблицах;

- сохраняет форматирование заголовков таблиц;

- основной шрифт: Times New Roman 14pt;

- шрифт в таблицах: Times New Roman 12pt;

- обновляет стили Normal и табличные стили;

# Приложение А

Листинг А.1 – код веб-интерфейса index.html

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Форматирование файла по ГОСТ</title>

<!-- Bootstrap 5 CSS -->

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Иконки Bootstrap Icons -->

<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.0/font/bootstrap-icons.css">

<!-- React CDN -->

<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>

<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>

<script crossorigin src="https://unpkg.com/@babel/standalone/babel.min.js"></script>

<link rel="stylesheet" href="css/style.css">

</head>

<body class="bg-light">

<div id="root"></div>

<script type="text/babel" src="js/fileUpload.js"></script>

<script type="text/babel" src="js/literature.js"></script>

<script type="text/babel" src="js/titlePageForm.js"></script>

<script type="text/babel">

const { useState, useRef } = React;

// Главный компонент приложения

function App() {

const [file, setFile] = useState(null);

const [format, setFormat] = useState('docx');

const [formData, setFormData] = useState({

studentName: '',

pz: '',

studentGroup: '',

workTitle: '',

workTopic: '',

institute: '',

department: '',

teacherName: '',

teacherPosition: '',

workYear: ''

});

const [literature, setLiterature] = useState([]);

const [result, setResult] = useState(null);

const handleFormatSubmit = async () => {

if (!file) return;

setResult({ status: 'loading', message: 'Идет форматирование файла...' });

try {

// 1. Создаем FormData для отправки файла и данных

const formPayload = new FormData();

// 2. Добавляем файл

formPayload.append('document', file);

// 3. Добавляем метаданные в JSON

formPayload.append('metadata', JSON.stringify({

format: format,

titleData: formData,

literature: literature

}));

console.log([...formPayload.entries()]);

// 4. Отправляем на сервер

const response = await fetch('http://localhost:8080/format', {

method: 'POST',

body: formPayload

// Не устанавливаем Content-Type - браузер сделает это сам с boundary

});

if (!response.ok) {

throw new Error(`Ошибка сервера: ${response.status}`);

}

// 5. Получаем и сохраняем результат

const blob = await response.blob();

const url = URL.createObjectURL(blob);

// 6. Автоматическое скачивание

const a = document.createElement('a');

a.href = url;

a.download = `Отчет\_по\_${formData.workTitle.trim().replace(/\s+/g, '\_')}\_СП\_${formData.studentName.trim().replace(/\s+/g, '\_').replace(/\./g, '\_')}${formData.studentGroup}.${format}`;

a.click();

setResult({

status: 'success',

message: 'Форматирование завершено! Файл автоматически загружается.'

});

} catch (error) {

setResult({

status: 'error',

message: `Ошибка: ${error.message}`

});

console.error('Ошибка при отправке:', error);

}

};

return (

<div className="container py-5">

<div className="row justify-content-center">

<div className="col-lg-8">

<div className="card shadow-sm">

<div className="card-header text-white bg-primary">

<h2 className="h4 mb-0"><i className="bi bi-file-earmark-text me-2"></i>Форматирование файла по ГОСТ</h2>

</div>

<div className="card-body">

<div className="mb-4">

<p className="text-muted">Загрузите файл текстовый (DOCX, TXT или ODT) БЕЗ ТИТУЛЬНОГО ЛИСТА, и мы автоматически отформатируем его по ГОСТ СТО 1.701-2010</p>

</div>

<FileUploadArea

onFileSelect={setFile}

onFileClear={() => setFile(null)}

fileName={file?.name}

/>

<div className="mb-4">

<h5 className="mb-3"><i className="bi bi-gear me-2"></i>Настройки форматирования</h5>

<div className="row">

<div className="col-md-6 mb-3">

<label className="form-label">Формат результата</label>

<select

className="form-select"

value={format}

onChange={(e) => setFormat(e.target.value)}

>

<option value="docx">DOCX (Word)</option>

<option value="pdf">PDF</option>

</select>

</div>

</div>

</div>

<TitlePageForm

formData={formData}

onFormChange={setFormData}

/>

<LiteratureList

literature={literature}

setLiterature={setLiterature}

/>

<div className="d-grid gap-2 mt-4">

<button

className="btn btn-custom btn-lg"

onClick={handleFormatSubmit}

disabled={!file}

>

<i className="bi bi-magic me-2"></i>Отформатировать по ГОСТ

</button>

</div>

{result && (

<div className={`alert mt-4 alert-${result.status === 'error' ? 'danger' : result.status === 'success' ? 'success' : 'info'} d-flex align-items-center`}>

{result.status === 'loading' ? (

<div className="spinner-border spinner-border-sm me-2" role="status"></div>

) : result.status === 'success' ? (

<i className="bi bi-check-circle-fill me-2"></i>

) : (

<i className="bi bi-exclamation-triangle-fill me-2"></i>

)}

<span>{result.message}</span>

</div>

)}

</div>

<div className="card-footer text-muted small">

<i className="bi bi-info-circle me-1"></i>Система автоматически применяет требования ГОСТ СТО 1.701-2010

</div>

</div>

</div>

</div>

</div>

);

}

// Рендерим приложение

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

</script>

</body>

</html>

# Листинг A.2 – код веб-интерфейса fileUpload.js

function FileUploadArea({ onFileSelect, onFileClear, fileName }) {

const fileInputRef = useRef();

const [isDragging, setIsDragging] = useState(false);

const handleClick = () => fileInputRef.current.click();

const handleChange = (e) => {

if (e.target.files[0]) {

onFileSelect(e.target.files[0]);

}

};

const handleDragOver = (e) => {

e.preventDefault();

setIsDragging(true);

};

const handleDragLeave = () => setIsDragging(false);

const handleDrop = (e) => {

e.preventDefault();

setIsDragging(false);

if (e.dataTransfer.files[0]) {

onFileSelect(e.dataTransfer.files[0]);

}

};

return (

<div

className={`upload-area mb-4 ${isDragging ? 'border-primary bg-light' : ''}`}

onClick={handleClick}

onDragOver={handleDragOver}

onDragLeave={handleDragLeave}

onDrop={handleDrop}

>

<input

type="file"

ref={fileInputRef}

className="d-none"

accept=".docx,.txt,.odt"

onChange={handleChange}

/>

<div className="preview-icon mb-3">

<i className={`bi ${fileName ? 'bi-file-earmark-text' : 'bi-cloud-arrow-up'}`}></i>

</div>

<h5 className="mb-2">Перетащите файл сюда или нажмите для выбора</h5>

<p className="text-muted small">Поддерживаемые форматы: .docx, .txt, .odt</p>

{fileName && (

<div className="file-info">

<span className="badge bg-info">{fileName}</span>

<button

className="btn btn-sm btn-outline-danger ms-2"

onClick={(e) => {

e.stopPropagation();

onFileClear();

}}

>

<i className="bi bi-x"></i>

</button>

</div>

)}

</div>

);

}

# Листинг А.3 – веб-интерфейс literature.js

function LiteratureList({ literature, setLiterature }) {

const [newSource, setNewSource] = useState('');

const handleAddSource = () => {

if (newSource.trim()) {

setLiterature([...literature, newSource.trim()]);

setNewSource('');

}

};

const handleRemoveSource = (index) => {

const updated = [...literature];

updated.splice(index, 1);

setLiterature(updated);

};

return (

<div className="card mt-3">

<div className="card-header bg-light">

<h6 className="mb-0"><i className="bi bi-book me-2"></i>Список литературы</h6>

</div>

<div className="card-body">

<div className="mb-3">

<div className="input-group">

<input

type="text"

className="form-control"

value={newSource}

onChange={(e) => setNewSource(e.target.value)}

placeholder="Добавьте источник (книга, статья и т.д.)"

/>

<button

className="btn btn-primary"

type="button"

onClick={handleAddSource}

>

<i className="bi bi-plus"></i> Добавить

</button>

</div>

</div>

{literature.length > 0 ? (

<ul className="list-group">

{literature.map((source, index) => (

<li key={index} className="list-group-item literature-item">

{source}

<button

className="btn btn-sm btn-outline-danger remove-literature"

onClick={() => handleRemoveSource(index)}

>

<i className="bi bi-trash"></i>

</button>

</li>

))}

</ul>

) : (

<p className="text-muted">Нет добавленных источников</p>

)}

</div>

</div>

);

}

# Листинг А.4 – веб-интерфейс titlePageForm.js

function TitlePageForm({ formData, onFormChange }) {

const handleChange = (e) => {

onFormChange({

...formData,

[e.target.id]: e.target.value

});

};

return (

<div className="card mt-3">

<div className="card-header bg-light">

<h6 className="mb-0"><i className="bi bi-card-heading me-2"></i>Данные для титульного листа</h6>

</div>

<div className="card-body">

<div className="row g-3">

<div className="col-md-6">

<label className="form-label">Фамилия И.О. студента</label>

<input

type="text"

className="form-control"

id="studentName"

placeholder="Иванов И.И."

value={formData.studentName}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">ПЗ</label>

<input

type="text"

className="form-control"

id="pz"

placeholder="004.457"

value={formData.pz}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Группа</label>

<input

type="text"

className="form-control"

id="studentGroup"

placeholder="ПИ-123"

value={formData.studentGroup}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Название работы</label>

<input

type="text"

className="form-control"

id="workTitle"

placeholder="Курсовая работа"

value={formData.workTitle}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Тема работы</label>

<input

type="text"

className="form-control"

id="workTopic"

placeholder="Разработка веб-приложения"

value={formData.workTopic}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Институт</label>

<input

type="text"

className="form-control"

id="institute"

placeholder="Институт компьютерных наук"

value={formData.institute}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Кафедра</label>

<input

type="text"

className="form-control"

id="department"

placeholder="Программная инженерия"

value={formData.department}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">ФИО преподавателя</label>

<input

type="text"

className="form-control"

id="teacherName"

placeholder="Петрова П.П."

value={formData.teacherName}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Должность преподавателя</label>

<input

type="text"

className="form-control"

id="teacherPosition"

placeholder="доцент"

value={formData.teacherPosition}

onChange={handleChange}

/>

</div>

<div className="col-md-6">

<label className="form-label">Год выполнения</label>

<input

type="number"

className="form-control"

id="workYear"

placeholder="2023"

min="2000"

max="2100"

value={formData.workYear}

onChange={handleChange}

/>

</div>

</div>

</div>

</div>

);

}

# Листинг А.5 – сервер на С++ (serverStart.cpp)

#include <iostream>

#include <fstream>

#include "include/json.hpp"

#include "include/httplib.h"

#include "formatter.h"

using json = nlohmann::json;

using namespace httplib;

using namespace std;

int main() {

Server svr;

// Включить CORS

svr.set\_default\_headers({

{"Access-Control-Allow-Origin", "\*"},

{"Access-Control-Allow-Methods", "POST, GET, OPTIONS"},

{"Access-Control-Allow-Headers", "Content-Type"}

});

svr.Options("/format", [](const Request&, Response& res) {

res.status = 200;

});

svr.Post("/format", [](const Request& req, Response& res) {

cout << "Received POST to /format\n";

if (!req.has\_file("document") || !req.has\_file("metadata")) {

cout << "Missing files in request\n";

res.status = 400;

res.set\_content("Неверный формат запроса", "text/plain");

return;

}

try {

auto file = req.get\_file\_value("document");

auto metadata\_json = req.get\_file\_value("metadata");

cout << "File size: " << file.content.length() << " bytes\n";

cout << "Metadata: " << metadata\_json.content << "\n";

json metadata = json::parse(metadata\_json.content);

string result = format\_document(file.content, metadata);

res.set\_content(result, "application/vnd.openxmlformats-officedocument.wordprocessingml.document");

res.set\_header("Content-Disposition", "attachment; filename=formatted.docx");

}

catch (const exception& e) {

cerr << "Error: " << e.what() << "\n";

res.status = 500;

res.set\_content("Ошибка обработки: " + string(e.what()), "text/plain");

}

});

svr.set\_base\_dir("./front"); // Папка, где хранятся ваши HTML файлы

cout << "Сервер запущен на http://localhost:8080\n";

if (!svr.listen("0.0.0.0", 8080)) {

cerr << "Не удалось запустить сервер!\n";

return 1;

}

return 0;

}

# Листинг А.6 – обработка файла (formatted.cpp)

#include "formatter.h"

#include <fstream>

#include <sstream>

#include <iostream>

#include <vector>

#include <tinyxml2.h>

#include <minizip/mz.h> // Базовые определения

#include <minizip/mz\_os.h> // Функции ОС

#include <minizip/mz\_strm.h> // Потоки

#include <minizip/mz\_zip.h> // ZIP-архивы

#include <minizip/mz\_zip\_rw.h> // Чтение/запись ZIP

using namespace std;

using namespace tinyxml2;

struct DocxFile {

string path;

vector<char> data;

};

// Константы форматирования

const int LEFT\_MARGIN = 1701; // 30 мм

const int RIGHT\_MARGIN = 567; // 10 мм

const int TOP\_MARGIN = 1134; // 20 мм

const int BOTTOM\_MARGIN = 1134; // 20 мм

const int LINE\_SPACING = 360; // 1.5 строки (360 twips)

const int TABLE\_FONT\_SIZE = 24; // 12pt (24 twips)

// Установка полей страницы

void SetPageMargins(XMLDocument& doc) {

XMLElement\* body = doc.FirstChildElement("w:document")->FirstChildElement("w:body");

if (!body) return;

XMLElement\* sectPr = body->FirstChildElement("w:sectPr");

if (!sectPr) {

sectPr = doc.NewElement("w:sectPr");

body->InsertEndChild(sectPr);

}

XMLElement\* pgMar = sectPr->FirstChildElement("w:pgMar");

if (!pgMar) {

pgMar = doc.NewElement("w:pgMar");

sectPr->InsertFirstChild(pgMar);

}

pgMar->SetAttribute("w:left", LEFT\_MARGIN);

pgMar->SetAttribute("w:right", RIGHT\_MARGIN);

pgMar->SetAttribute("w:top", TOP\_MARGIN);

pgMar->SetAttribute("w:bottom", BOTTOM\_MARGIN);

}

// Установка межстрочного интервала для абзаца

void SetParagraphSpacing(XMLElement\* pPr, XMLDocument& doc) {

if (!pPr) return;

XMLElement\* spacing = pPr->FirstChildElement("w:spacing");

if (!spacing) {

spacing = doc.NewElement("w:spacing");

pPr->InsertEndChild(spacing);

}

spacing->SetAttribute("w:line", LINE\_SPACING);

spacing->SetAttribute("w:lineRule", "auto");

}

// Форматирование текста в таблицах (шрифт 12pt)

void FormatTables(XMLDocument& doc) {

XMLElement\* body = doc.FirstChildElement("w:document")->FirstChildElement("w:body");

if (!body) return;

// Ищем все таблицы в документе

for (XMLElement\* tbl = body->FirstChildElement("w:tbl"); tbl; tbl = tbl->NextSiblingElement("w:tbl")) {

// Обрабатываем все строки в таблице

for (XMLElement\* row = tbl->FirstChildElement("w:tr"); row; row = row->NextSiblingElement("w:tr")) {

// Обрабатываем все ячейки в строке

for (XMLElement\* cell = row->FirstChildElement("w:tc"); cell; cell = cell->NextSiblingElement("w:tc")) {

// Обрабатываем все абзацы в ячейке

for (XMLElement\* p = cell->FirstChildElement("w:p"); p; p = p->NextSiblingElement("w:p")) {

// Обрабатываем все текстовые элементы в абзаце

for (XMLElement\* r = p->FirstChildElement("w:r"); r; r = r->NextSiblingElement("w:r")) {

XMLElement\* rPr = r->FirstChildElement("w:rPr");

if (!rPr) {

rPr = doc.NewElement("w:rPr");

r->InsertFirstChild(rPr);

}

// Устанавливаем размер шрифта 12pt (24 twips)

XMLElement\* sz = rPr->FirstChildElement("w:sz");

if (!sz) {

sz = doc.NewElement("w:sz");

rPr->InsertEndChild(sz);

}

sz->SetAttribute("w:val", "24");

XMLElement\* szCs = rPr->FirstChildElement("w:szCs");

if (!szCs) {

szCs = doc.NewElement("w:szCs");

rPr->InsertEndChild(szCs);

}

szCs->SetAttribute("w:val", "24");

// Удаляем жирное выделение и цвет

if (rPr->FirstChildElement("w:b")) {

rPr->DeleteChild(rPr->FirstChildElement("w:b"));

}

if (rPr->FirstChildElement("w:bCs")) {

rPr->DeleteChild(rPr->FirstChildElement("w:bCs"));

}

if (rPr->FirstChildElement("w:color")) {

rPr->DeleteChild(rPr->FirstChildElement("w:color"));

}

}

// Устанавливаем выравнивание по ширине для абзацев в таблице

XMLElement\* pPr = p->FirstChildElement("w:pPr");

if (!pPr) {

pPr = doc.NewElement("w:pPr");

p->InsertFirstChild(pPr);

}

XMLElement\* jc = pPr->FirstChildElement("w:jc");

if (!jc) {

jc = doc.NewElement("w:jc");

pPr->InsertEndChild(jc);

}

jc->SetAttribute("w:val", "both");

// Устанавливаем интервал для абзацев в таблице

SetParagraphSpacing(pPr, doc);

}

}

}

}

}

// Выравнивание текста по ширине и очистка форматирования

void FormatParagraphs(XMLDocument& doc) {

XMLElement\* body = doc.FirstChildElement("w:document")->FirstChildElement("w:body");

if (!body) return;

for (XMLElement\* p = body->FirstChildElement("w:p"); p; p = p->NextSiblingElement("w:p")) {

// Пропускаем абзацы внутри таблиц (они обрабатываются отдельно)

if (p->Parent() && p->Parent()->ToElement() &&

strcmp(p->Parent()->ToElement()->Name(), "w:tbl") == 0) {

continue;

}

XMLElement\* pPr = p->FirstChildElement("w:pPr");

if (!pPr) {

pPr = doc.NewElement("w:pPr");

p->InsertFirstChild(pPr);

}

// Устанавливаем интервал для ВСЕХ абзацев

SetParagraphSpacing(pPr, doc);

// Пропускаем заголовки (стили Heading1-Heading6)

bool isHeading = false;

XMLElement\* pStyle = pPr->FirstChildElement("w:pStyle");

if (pStyle) {

const char\* val = pStyle->Attribute("w:val");

if (val && string(val).find("Heading") != string::npos) {

isHeading = true;

}

}

// Пропускаем приложения

bool isAppendix = false;

XMLElement\* r = p->FirstChildElement("w:r");

if (r) {

XMLElement\* t = r->FirstChildElement("w:t");

if (t) {

const char\* text = t->GetText();

if (text && string(text).find("Приложение") != string::npos) {

isAppendix = true;

}

}

}

if (!isHeading && !isAppendix) {

// Устанавливаем выравнивание по ширине

XMLElement\* jc = pPr->FirstChildElement("w:jc");

if (!jc) {

jc = doc.NewElement("w:jc");

pPr->InsertEndChild(jc);

}

jc->SetAttribute("w:val", "both");

// Удаляем жирное выделение и цвет

for (XMLElement\* r = p->FirstChildElement("w:r"); r; r = r->NextSiblingElement("w:r")) {

XMLElement\* rPr = r->FirstChildElement("w:rPr");

if (rPr) {

// Удаляем жирное выделение

if (rPr->FirstChildElement("w:b")) {

rPr->DeleteChild(rPr->FirstChildElement("w:b"));

}

if (rPr->FirstChildElement("w:bCs")) {

rPr->DeleteChild(rPr->FirstChildElement("w:bCs"));

}

// Удаляем цвет текста

if (rPr->FirstChildElement("w:color")) {

rPr->DeleteChild(rPr->FirstChildElement("w:color"));

}

}

}

}

}

}

// Создание нумерации страниц

void CreatePageNumbers(XMLDocument& doc) {

XMLElement\* footer = doc.NewElement("w:ftr");

footer->SetAttribute("xmlns:w", "http://schemas.openxmlformats.org/wordprocessingml/2006/main");

XMLElement\* p = doc.NewElement("w:p");

// Настройки абзаца

XMLElement\* pPr = doc.NewElement("w:pPr");

// Устанавливаем интервал для колонтитула

SetParagraphSpacing(pPr, doc);

// Выравнивание по центру

XMLElement\* jc = doc.NewElement("w:jc");

jc->SetAttribute("w:val", "center");

pPr->InsertEndChild(jc);

p->InsertFirstChild(pPr);

// Номер страницы

XMLElement\* r = doc.NewElement("w:r");

XMLElement\* fldChar = doc.NewElement("w:fldChar");

fldChar->SetAttribute("w:fldCharType", "begin");

r->InsertEndChild(fldChar);

p->InsertEndChild(r);

r = doc.NewElement("w:r");

fldChar = doc.NewElement("w:fldChar");

fldChar->SetAttribute("w:fldCharType", "separate");

r->InsertEndChild(fldChar);

p->InsertEndChild(r);

r = doc.NewElement("w:r");

XMLElement\* instrText = doc.NewElement("w:instrText");

instrText->SetAttribute("xml:space", "preserve");

XMLText\* text = doc.NewText(" PAGE ");

instrText->InsertEndChild(text);

r->InsertEndChild(instrText);

p->InsertEndChild(r);

r = doc.NewElement("w:r");

fldChar = doc.NewElement("w:fldChar");

fldChar->SetAttribute("w:fldCharType", "end");

r->InsertEndChild(fldChar);

p->InsertEndChild(r);

footer->InsertEndChild(p);

doc.InsertEndChild(footer);

}

// Настройка стилей документа

// Настройка стилей документа

void SetDocumentStyles(XMLDocument& doc) {

XMLElement\* styles = doc.FirstChildElement("w:styles");

if (!styles) return;

// 1. Настройка стиля Normal (основной стиль документа)

XMLElement\* styleNormal = nullptr;

for (XMLElement\* style = styles->FirstChildElement("w:style"); style; style = style->NextSiblingElement("w:style")) {

const char\* styleId = style->Attribute("w:styleId");

if (styleId && strcmp(styleId, "Normal") == 0) {

styleNormal = style;

break;

}

}

if (!styleNormal) {

styleNormal = doc.NewElement("w:style");

styleNormal->SetAttribute("w:type", "paragraph");

styleNormal->SetAttribute("w:styleId", "Normal");

styleNormal->SetAttribute("w:default", "1");

styles->InsertEndChild(styleNormal);

}

// Настройки абзаца для Normal

XMLElement\* pPr = styleNormal->FirstChildElement("w:pPr");

if (!pPr) {

pPr = doc.NewElement("w:pPr");

styleNormal->InsertEndChild(pPr);

}

// Устанавливаем интервал для стиля Normal

SetParagraphSpacing(pPr, doc);

// Выравнивание по ширине

XMLElement\* jc = pPr->FirstChildElement("w:jc");

if (!jc) {

jc = doc.NewElement("w:jc");

pPr->InsertEndChild(jc);

}

jc->SetAttribute("w:val", "both");

// Настройки шрифта для Normal

XMLElement\* rPr = styleNormal->FirstChildElement("w:rPr");

if (!rPr) {

rPr = doc.NewElement("w:rPr");

styleNormal->InsertEndChild(rPr);

}

XMLElement\* rFonts = rPr->FirstChildElement("w:rFonts");

if (!rFonts) {

rFonts = doc.NewElement("w:rFonts");

rPr->InsertFirstChild(rFonts);

}

rFonts->SetAttribute("w:ascii", "Times New Roman");

rFonts->SetAttribute("w:hAnsi", "Times New Roman");

XMLElement\* sz = rPr->FirstChildElement("w:sz");

if (!sz) {

sz = doc.NewElement("w:sz");

rPr->InsertEndChild(sz);

}

sz->SetAttribute("w:val", "28"); // 14pt

// Удаляем жирное выделение в стиле Normal

if (rPr->FirstChildElement("w:b")) {

rPr->DeleteChild(rPr->FirstChildElement("w:b"));

}

if (rPr->FirstChildElement("w:bCs")) {

rPr->DeleteChild(rPr->FirstChildElement("w:bCs"));

}

// Удаляем цвет текста в стиле Normal

if (rPr->FirstChildElement("w:color")) {

rPr->DeleteChild(rPr->FirstChildElement("w:color"));

}

// 2. Настройка стилей таблиц

for (XMLElement\* style = styles->FirstChildElement("w:style"); style; style = style->NextSiblingElement("w:style")) {

const char\* type = style->Attribute("w:type");

if (!type) continue;

// Обрабатываем стили таблиц и ячеек

if (strcmp(type, "table") == 0 || strcmp(type, "tableCell") == 0) {

// Настройки форматирования текста

XMLElement\* rPr = style->FirstChildElement("w:rPr");

if (!rPr) {

rPr = doc.NewElement("w:rPr");

style->InsertFirstChild(rPr);

}

// Размер шрифта 12pt (24 twips)

XMLElement\* sz = rPr->FirstChildElement("w:sz");

if (!sz) {

sz = doc.NewElement("w:sz");

rPr->InsertEndChild(sz);

}

sz->SetAttribute("w:val", "24");

XMLElement\* szCs = rPr->FirstChildElement("w:szCs");

if (!szCs) {

szCs = doc.NewElement("w:szCs");

rPr->InsertEndChild(szCs);

}

szCs->SetAttribute("w:val", "24");

// Удаляем жирное выделение

if (rPr->FirstChildElement("w:b")) {

rPr->DeleteChild(rPr->FirstChildElement("w:b"));

}

if (rPr->FirstChildElement("w:bCs")) {

rPr->DeleteChild(rPr->FirstChildElement("w:bCs"));

}

// Удаляем цвет текста

if (rPr->FirstChildElement("w:color")) {

rPr->DeleteChild(rPr->FirstChildElement("w:color"));

}

}

// Для стилей абзацев, которые могут использоваться в таблицах

if (strcmp(type, "paragraph") == 0) {

const char\* styleId = style->Attribute("w:styleId");

if (styleId && (strstr(styleId, "Table") != nullptr || strstr(styleId, "table") != nullptr)) {

XMLElement\* rPr = style->FirstChildElement("w:rPr");

if (!rPr) {

rPr = doc.NewElement("w:rPr");

style->InsertFirstChild(rPr);

}

// Устанавливаем размер шрифта 12pt

XMLElement\* sz = rPr->FirstChildElement("w:sz");

if (!sz) {

sz = doc.NewElement("w:sz");

rPr->InsertEndChild(sz);

}

sz->SetAttribute("w:val", "24");

XMLElement\* szCs = rPr->FirstChildElement("w:szCs");

if (!szCs) {

szCs = doc.NewElement("w:szCs");

rPr->InsertEndChild(szCs);

}

szCs->SetAttribute("w:val", "24");

}

}

}

// 3. Настройка стилей для заголовков таблиц (если есть)

const char\* tableStyleNames[] = { "Table Grid", "Table Normal", "Table", "Light Shading", "Medium Shading" };

for (const char\* styleName : tableStyleNames) {

for (XMLElement\* style = styles->FirstChildElement("w:style"); style; style = style->NextSiblingElement("w:style")) {

const char\* nameAttr = style->Attribute("w:name");

if (nameAttr && strcmp(nameAttr, styleName) == 0) {

XMLElement\* rPr = style->FirstChildElement("w:rPr");

if (!rPr) {

rPr = doc.NewElement("w:rPr");

style->InsertFirstChild(rPr);

}

// Устанавливаем размер шрифта 12pt

XMLElement\* sz = rPr->FirstChildElement("w:sz");

if (!sz) {

sz = doc.NewElement("w:sz");

rPr->InsertEndChild(sz);

}

sz->SetAttribute("w:val", "24");

XMLElement\* szCs = rPr->FirstChildElement("w:szCs");

if (!szCs) {

szCs = doc.NewElement("w:szCs");

rPr->InsertEndChild(szCs);

}

szCs->SetAttribute("w:val", "24");

break;

}

}

}

}

// Обработка XML-файлов

string ProcessXml(const vector<char>& xmlData, const string& filename) {

XMLDocument doc;

if (doc.Parse(xmlData.data(), xmlData.size()) != XML\_SUCCESS) {

cerr << "Ошибка парсинга: " << filename << endl;

return "";

}

if (filename == "word/styles.xml") {

SetDocumentStyles(doc);

}

else if (filename == "word/document.xml") {

SetPageMargins(doc);

FormatParagraphs(doc);

FormatTables(doc); // Добавлена обработка таблиц

}

else if (filename == "word/footer1.xml") {

CreatePageNumbers(doc);

}

XMLPrinter printer;

doc.Print(&printer);

return printer.CStr();

}

// Распаковка DOCX

vector<DocxFile> UnzipDocx(const string& filename) {

vector<DocxFile> files;

int32\_t err = MZ\_OK;

// Создаём объект для чтения ZIP (без аргументов!)

void \*reader = mz\_zip\_reader\_create();

if (!reader) {

cerr << "Ошибка создания ZIP-ридера" << endl;

return files;

}

// Открываем архив

err = mz\_zip\_reader\_open\_file(reader, filename.c\_str());

if (err != MZ\_OK) {

cerr << "Ошибка открытия файла: " << filename << endl;

mz\_zip\_reader\_delete(&reader);

return files;

}

// Переходим к первому файлу в архиве

err = mz\_zip\_reader\_goto\_first\_entry(reader);

if (err != MZ\_OK) {

cerr << "Архив пуст или повреждён" << endl;

mz\_zip\_reader\_delete(&reader);

return files;

}

// Читаем все файлы

do {

mz\_zip\_file \*file\_info = NULL;

err = mz\_zip\_reader\_entry\_get\_info(reader, &file\_info);

if (err != MZ\_OK) {

cerr << "Ошибка чтения информации о файле" << endl;

continue;

}

// Подготавливаем структуру для хранения данных

DocxFile file;

file.path = file\_info->filename;

file.data.resize(file\_info->uncompressed\_size);

// Читаем содержимое файла

err = mz\_zip\_reader\_entry\_read(reader, file.data.data(), file.data.size());

if (err < 0) {

cerr << "Ошибка чтения файла: " << file.path << endl;

continue;

}

files.push\_back(file);

} while (mz\_zip\_reader\_goto\_next\_entry(reader) == MZ\_OK);

// Закрываем и освобождаем ресурсы

mz\_zip\_reader\_close(reader);

mz\_zip\_reader\_delete(&reader);

return files;

}

// Упаковка DOCX

bool ZipDocx(const vector<DocxFile>& files, const string& filename) {

int32\_t err = MZ\_OK;

// Создаём объект для записи ZIP

void \*writer = mz\_zip\_writer\_create();

if (!writer) {

cerr << "Ошибка создания ZIP-райтера" << endl;

return false;

}

// Открываем архив для записи

err = mz\_zip\_writer\_open\_file(writer, filename.c\_str(), 0, 1);

if (err != MZ\_OK) {

cerr << "Ошибка создания архива: " << filename << endl;

mz\_zip\_writer\_delete(&writer);

return false;

}

// Добавляем файлы в архив

for (const auto& file : files) {

mz\_zip\_file file\_info = {};

file\_info.filename = file.path.c\_str();

file\_info.compression\_method = MZ\_COMPRESS\_METHOD\_DEFLATE;

file\_info.zip64 = MZ\_ZIP64\_AUTO;

// Явное приведение типа к void\*

err = mz\_zip\_writer\_add\_buffer(writer,

const\_cast<void\*>(static\_cast<const void\*>(file.data.data())),

file.data.size(),

&file\_info);

if (err != MZ\_OK) {

cerr << "Ошибка добавления файла: " << file.path << endl;

continue;

}

}

// Закрываем архив

mz\_zip\_writer\_close(writer);

mz\_zip\_writer\_delete(&writer);

return true;

}

string format\_document(const string& docx\_data, const nlohmann::json& metadata) {

string inputPath = "temp\_input.docx";

string outputPath = "temp\_output.docx";

ofstream out(inputPath, ios::binary);

out.write(docx\_data.c\_str(), docx\_data.size());

out.close();

vector<DocxFile> files = UnzipDocx(inputPath);

if (files.empty()) {

throw runtime\_error("Не удалось распаковать документ");

}

bool hasFooter = false;

for (auto& file : files)

if (file.path == "word/footer1.xml")

hasFooter = true;

if (!hasFooter) {

DocxFile footer;

footer.path = "word/footer1.xml";

string content = R"(<?xml version=\"1.0\" encoding=\"UTF-8\" standalone=\"yes\"?><w:ftr xmlns:w=\"http://schemas.openxmlformats.org/wordprocessingml/2006/main\"></w:ftr>)";

footer.data.assign(content.begin(), content.end());

files.push\_back(footer);

}

for (auto& file : files) {

string newContent = ProcessXml(file.data, file.path);

if (!newContent.empty()) {

file.data.assign(newContent.begin(), newContent.end());

}

}

if (!ZipDocx(files, outputPath)) {

throw runtime\_error("Ошибка упаковки документа");

}

ifstream in(outputPath, ios::binary);

ostringstream ss;

ss << in.rdbuf();

return ss.str();

}