Д. С. Топычканов, И. П. Худасов, Т. Ю. Чернышева
Тюменский государственный университет, г. Тюмень
УДК 004.8

РАЗРАБОТКА АЛГОРИТМА ФОРМАТИРОВАНИЯ СОДЕРЖИМОГО DOCX-ФАЙЛОВ

Аннотация. В настоящее время в государственных учреждениях вся документация пишется с учетом различных требований к оформлению. В статье представлен обзор web-приложения, разработанного для упрощения создания текстовых документов с учетом необходимости строгого соблюдения требований.

Ключевые слова: web-приложение, документ, оформление, правило, требование, PhpWord, PlateJS.

Введение. Форматирование — неотъемлемая часть процесса создания и редактирования документа, когда речь идет о приведении оформления к установленным стандартам. Популярные текстовые редакторы, несмотря на свою широкую функциональность, имеют ряд проблем, с которыми сталкиваются пользователи, к примеру «слетающие» шрифт текста и абзацные отступы [1]. Также пользователь может случайно совершить ошибку при оформлении элемента документа, из-за чего документ не будет соответствовать стандарту, по которому пишется.

Проблематика. Все вышеперечисленные проблемы ведут к дополнительным временным затратам на исправления и корректировку внешнего вида документа, а не его содержания. Обсуждаемое в данной статье web-приложение предлагает решение указанных проблем, предоставляя пользователям инструменты для создания текстовых документов с возможностью настройки пользовательских стилей оформления. Одной из ключевых особенностей является его способность гарантировать соблюдение установленных стандартов оформления документов. Практической задачей является разработка сервиса форматирования содержимого текстовых файлов [2], который избавит пользователей от времязатратной работы с оформлением и позволит сосредоточиться на самой сути документа при его написании.

Материалы и методы. Во время разработки использованы такие ключевые библиотеки, как PhpWord для генерации docx-документов на серверной части и PlateJS для редактирования текстового содержимого документов внутри браузера [3, 4].

Для каждого документа решено создавать отдельный набор правил оформления, который описывает каждый элемент шаблона как отдельную сущность с единственным экземпляром стилей. К примеру, для абзаца может существовать только один размер, цвет, шрифт текста, выравнивание, абзацный отступ и остальные компоненты, которые могут относиться к этому элементу. Такой стиль оформления применяется ко всем абзацам в документе, из-за чего у пользователя нет возможности изменить оформление какого-то конкретного элемента или совершить в нем ошибку. Это делает документ визуально единым и исключает возможность неверного оформления отдельных частей документа.

PlateJS — библиотека для создания редакторов форматированного текста, работающая благодаря атрибуту html-тегов ‘contenteditable’ [5]. Ее использование предполагает использование библиотеки react, для создания компонентов, которые будут использоваться в текстовом редакторе для ввода текста. В нашем случае это набор параграфов, заголовков, таблиц и т. д.

Стилистическое оформление разрабатывается по концепции WYSIWYG — редактора. Для ее реализации используются глобальные css-переменные [6], которые нужны для создания css-класса для каждого элемента ввода. Свойства для каждого элемента ввода уникальны и определенны в css-переменных, которые определяется исходя из набора правил оформления у документа [7].

Генерация документов на серверной части происходит с помощью библиотеки PhpWord. Этот инструмент избавляет разработчика от необходимости генерировать xml файлы документа, структутрировать их в архиве документа и упрощает работу с его тегами и элементами. С помощью него приложение генерирует все существующие элементы документа.

Результаты исследования. Алгоритм работы генерации документа, показанный на рис. 1, является парсером данных с клиентской части приложения. Сначала для переданного документа подгружается его шаблон, из которого берутся стили всех элементов. После чего через phpWord создаются стандартные стили в документе, такие как стандартный текст абзаца, стили заголовков и т. д. Далее парсер запускает алгоритм, в котором через массив прогоняется текст документа [8].

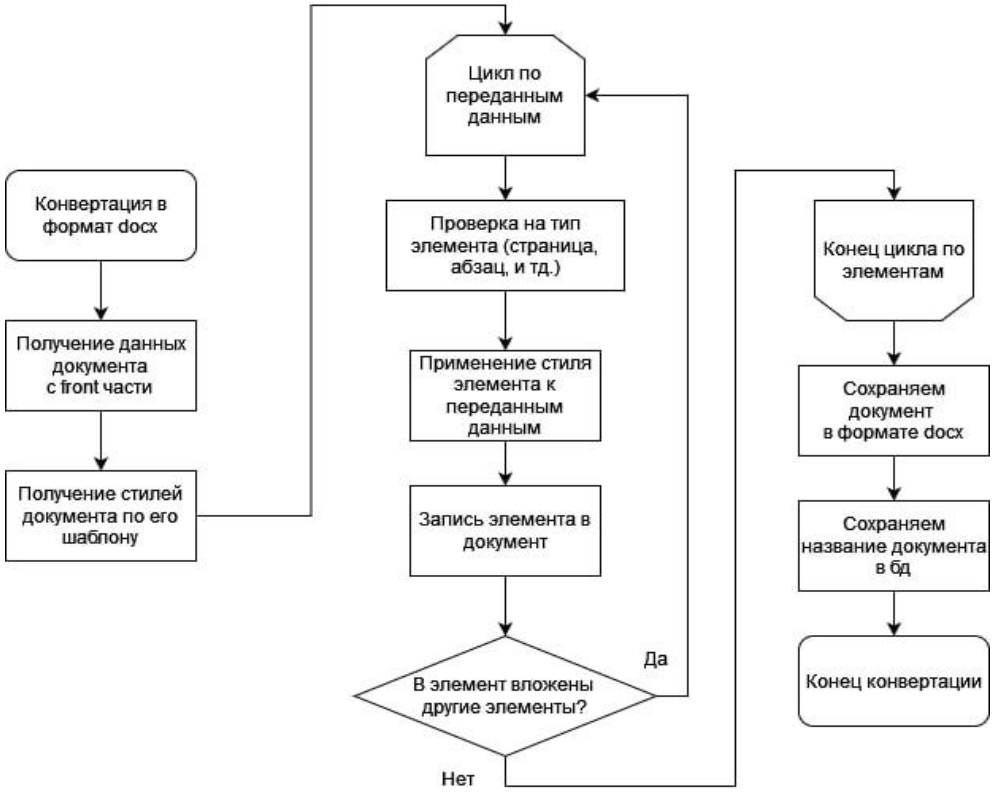

Рис. 1. Алгоритм парсинга данных с клиентской части

Алгоритм проверяет тип элемента и на его основе добавляет его контент в документ. Для каждого элемента создается секция, в которую после добавляется элемент класса phpWord, такой как абзац, таблица, список, изображение, формула и заголовок. Для этого элемента устанавливаются свои правила оформления, после чего элемент сохраняется в документ. После завершения итерации по элементам, алгоритм завершает свою работу, сохраняя документ на сервере, и возвращает его ссылку на клиентскую часть.

Со стороны пользователя создается набор правил оформления, который может быть использован множество раз. Для каждого элемента ввода настраиваются свои настройки оформления (рис. 2). После этого этапа создается отдельный документ (рис. 3), с выбором созданного шаблона. В документе редактирование происходит аналогично всем редактором — пользователь указывает курсором на поле ввода и по умолчанию вводит текст с использованием базового элемента «параграф». Если требуется новый элемент ввода, то он добавляется в документ через панель элементов. После чего пользователь сохраняет и скачивает документ в формате docx.

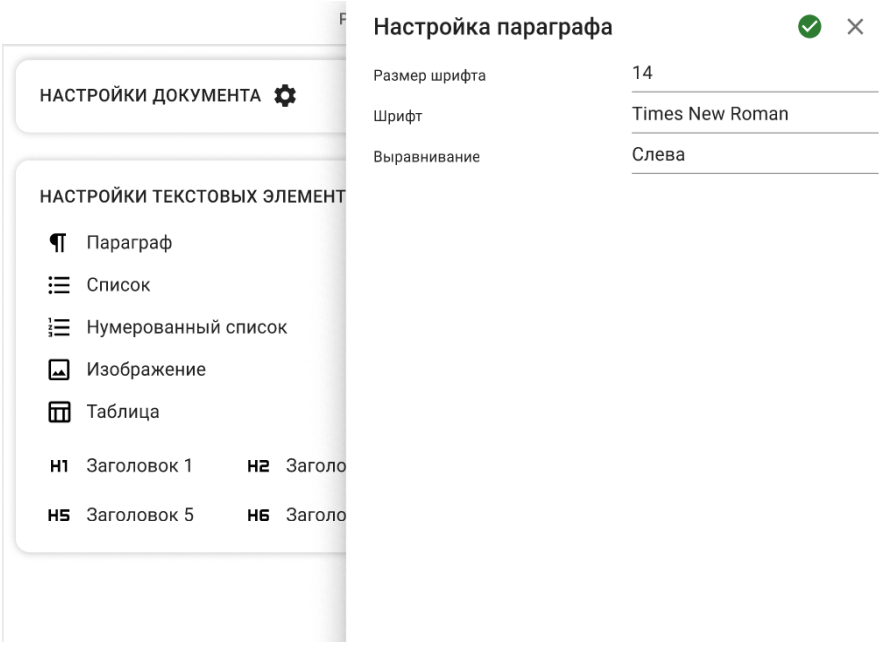

Рис. 2. Интерфейс редактирования правил оформления

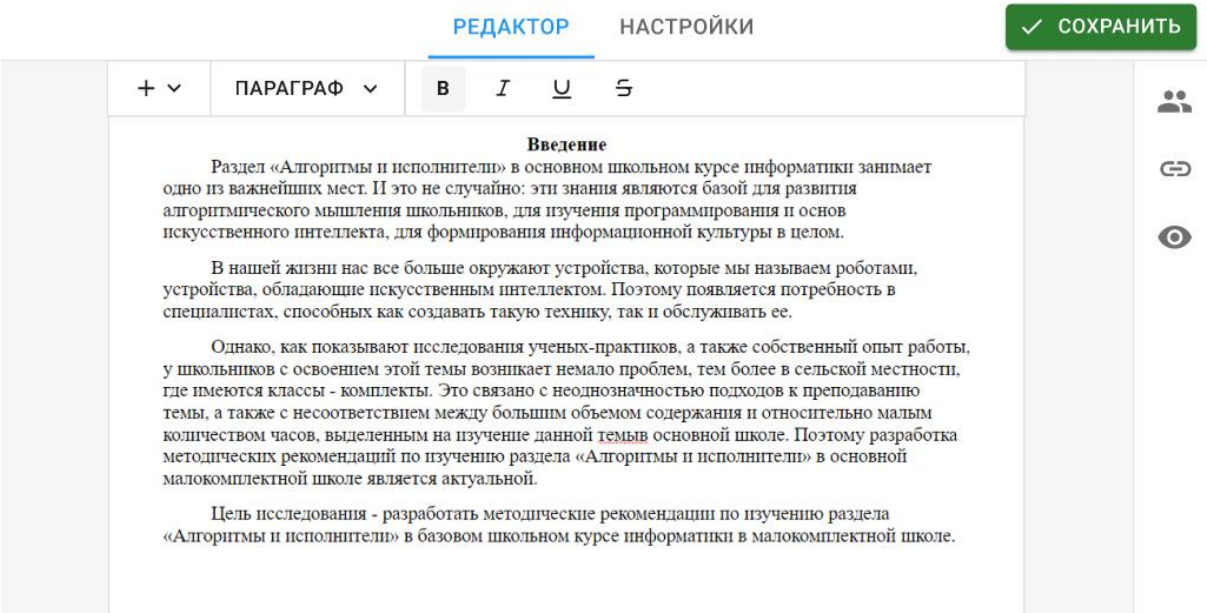

Рис. 3. Интерфейс редактора документа

Заключение. В данной статье рассмотрена работа с документацией, ее написание, стилизация и форматирование в docx. Проанализированы средства для работы с текстом в браузерах и создание документов на основе шаблонов. В дальнейшем планируется дорабатывать функционал web-приложения, например, добавить функционал импорта документа в сервис и реализовать выгрузку в формат odt.

СПИСОК ЛИТЕРАТУРЫ

1. Пудовина А.И. Проблемы документоведения и документационного обеспечения управления в свете современных стандартов / А.И. Пудовина, А.И. Чигрина, А.Р. Шаисламов. Уфа: Изд-во БГПУ, 2015. — С. 26-30. — URL: https://bspu.ru/tpl/sveden/files/education/RPD/FOS/metod_mat3_46.03.02_da_ddou_fgos3+.pdf (дата обращения: 10.05.2024). — Текст: электронный.
2. Стариченко Б.Е. Программа автоматизации контроля оформления текстовых документов / Б.Е. Стариченко, М.А. Устинов. — Текст: непосредственный // Педагогическое образование в России. — 2018. — № 8. — С. 163-168.
3. PhpWord: Библиотека для создания и форматирования документов: [сайт]. — URL: https://phpword.readthedocs.io/en/latest/intro.html (дата обращения: 10.05.2024). — Текст: электронный.
4. PlateJs: Библиотека для создания редакторов форматированного текста: [сайт]. — URL: https://platejs.org (дата обращения: 10.05.2024). — Текст: электронный.
5. contenteditable: Глобальный атрибут html: [сайт]. — URL: https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/contenteditable (дата обращения: 10.05.2024). — Текст: электронный.
6. Использование переменных в CSS: [сайт]. — URL: https://developer.mozilla.org/ru/docs/Web/CSS/Using_CSS_custom_properties (дата обращения: 10.05.2024). — Текст электронный.
7. Ефимов А.С. Текстовый процессор с устойчивым форматированием / А.С. Ефимов. — Текст: непосредственный // Новейшие технологии освоения месторождений углеводородного сырья и обеспечение безопасности экосистем каспийского шельфа. — 2023. — С. 373-375.
8. Топычканов Д.С. Модель текстового процессора с поддержкой настраиваемых правил оформления документов и конвертации в формат docx / Д.С. Топычканов, И.П. Худасов. — Текст: непосредственный // Сборник трудов XV Всероссийской научно-практической конференции для студентов и учащейся молодежи «Прогрессивные технологии и экономика в машиностроении» / Юргинский технологический институт. — Томск: Изд-во Томского политехнического университета, 2024. — № 3. — С. 237-239.