Санкт-Петербургский государственный университет

Математическое обеспечение и
администрирование информационных систем

Кафедра системного программирования

Коекин Ярослав Алексеевич

Разработка технологии управления знаниями ECM-системы для гибкого текстового редактора

Отчёт по учебной практике

Научный руководитель:
к.ф-м.н., доцент Луцив Д. В.
Консультант:
Руководитель отдела производства «ДоксВижн», ООО «ДоксВижн»
Елхов Д. С.

Санкт-Петербург
2021

Содержание

Введение **3**

1 Постановка задачи **5**

2 Обзор **6**

2.1 Существующие аналоги 6

2.2 Вывод 9

3 Архитектура приложения **10**

3.1 Встраивание системы управления знаниями в существующее приложение 10

3.2 Интеграция гибкого текстового веб-редактора с системой Docsvision 11

3.3 Атрибутизация в текстовом редакторе 11

4 Заключение **13**

Список используемой литературы **14**

Введение

ECM (англ. Enterprise Content Management, рус. Управление Корпоративным Контентом) — осуществление деятельности, направленное на хранение, обработку, распространение и управление цифровыми документами в рамках организации. ECM-система — программное обеспечение для управления корпоративным контентом. В список функциональных возможностей такой системы входят: управление документами (экспорт и импорт цифровых документов, контроль версий, архивирование контента), управление потоками работ (поддержка бизнес-процессов, передача контента по маршрутам), документоориентированное взаимодействие (поддержка возможности совместного использования цифровых документов пользователями) и др. Системы такого типа создаются в целях автоматизации и упрощения работы с большим количеством неструктурированной информации различного рода и формата.

Платформа Docsvision – полнофункциональная ECM-платформа, позволяющая упростить работу с корпоративным контентом.

На базе СПбГУ был создан прототип гибкого текстового редактора[1], позволяющий:

- создавать цифровой документ, содержащий текст, графики, таблицы и другие элементы
- объединять элементы документа в независимые блоки и управлять ими отдельно
- задавать внутренние дополнительные атрибуты документа
- создавать шаблоны цифрового документа
- совместно редактировать различные блоки цифрового документа и др.

Интеграция подобного редактора с ECM-системой облегчила бы взаимодействие пользователей при взаимодействии со структурой документа, а также со специфичными знаниями в рамках этой системы.

Основными объектами, управление которыми осуществляется ECM-системами, являются цифровые документы. Вместе с цифровыми документами в системе

[1]Разработка веб-редактора для ведения электронных структурированных документов ECM-системы, 2021 г.

ECM хранятся знания о нём: имена пользователей, номера телефонов, геопозиции и др. Все эти знания являются важными факторами при редактировании и наборе текста цифрового документа в системах ECM.

Для многих ECM-систем справедливо, что такие знания находятся не внутри основного источника всей информации – документе, а в стороне, практически никак с этими источниками не взаимодействуя. Система, знания которого интегрированы внутрь цифрового документа, имеет возможность улучшить опыт пользователей.

Преимущества подобного подхода:

- Удобство создания цифрового документа.
- Наглядность при просмотре и редактировании текста.
- Расширенные возможности управления знаниями о цифровом документе и информацией о данных пользователя.
- Возможность управления бизнес-процессами непосредственно внутри структуры документа, языком составления документа.

Подобные рассуждения являются мотивацией для создания и реализации альтернативного подхода при работе с цифровыми документами в ECM-системах.

1 Постановка задачи

Таким образом, целью данной работы является создание системы управления знаниями для гибкого текстового веб-редактора и его интеграция в веб-платформу DocsVision.

Для достижения данной цели были поставлены следующие задачи:

1. Изучить аналогичные подходы и системы интеграции знаний.
2. Изучить возможности к расширению системы DocsVision и его возможности по управлению контентом.
3. Разработать архитектуру системы взаимодействия текстового редактора и знаний.
4. Разработать архитектуру интеграции текстового редактора и системы ECM DocsVision.
5. Разработать систему взаимодействия текстового редактора со знаниями ECM-системы:
   - возможность задавать, менять и удалять элементы знаний документа в тексте документа;
   - возможность управлять бизнес-процессами посредством атрибутизации свойств документа.
6. Интегрировать полученное приложение с системой DocsVision.
7. Провести тестирование полученного приложения.

2 Обзор

2.1 Существующие аналоги

Для того, чтобы достичь поставленную цель, необходимо провести анализ существующих текстовых редакторов, которые имеют все необходимые функциональные требования, или которые могут быть расширены до удовлетворения всем требованиям.

Google Docs

Google Docs[2] – популярный WYSIWYG онлайн текстовый веб-редактор. Его функциональные возможности по созданию, редактированию и поддержке введённого пользователями текста обширны. Помимо этого, такой продукт имеет дополнительные возможности по шаблонированию и экспорту документа.

Google Docs интегрирован в экосистему Google (Drive[3], Calendar[4] и др.), которая позволяет иметь удобный доступ к документам и к его настройкам. Такая интеграция позволила также внедрять некоторые сущности из различных систем под управлением Google в текст документа. Например, такой документ позволяет внедрить информацию об электронной почте пользователя, находящегося в списке контактов клиента, даты из Google Calendar, ссылки на существующие файлы и др.

К сожалению, возможности по расширению Google Docs[5] не позволяют использовать данный продукт для интеграции с ECM-системой, а его лицензия не подходит для свободного коммерческого использования.

Notion

Notion[6] – текстовый редактор с широкими возможностями по кастомизации. Продукт нацелен в первую очередь на взаимодействие между людьми в различных командах и предлагает альтернативный подход к ведению текстовых онлайн документов. Такое решение подходит для поддержания репозитория знаний команды, документирования, управления задачами, общения между членами команды и др.

[2]https://www.google.com/docs/about/
[3]https://www.google.com/drive/
[4]https://calendar.google.com/calendar/
[5]https://developers.google.com/docs/api
[6]https://www.notion.so/

Документ в системе Notion – больше, чем обычный текстовый документ. Notion объединяет работу многих сервисов и решений: создание заметок, управление гиперссылками, создание баз данных внутри текста, написание и редактирование текстовой информации и др. Продукт работает с понятием блока, таким образом каждый элемент документа в Notion – состоит из одного или нескольких блоков, которые можно менять и переставлять.

Не смотря на то, что знаниями, которые сохраняются внутри Notion можно делиться в различных частях документа, их необходимо выбирать и настраивать вручную, что может быть неудобно в случае, если информация о таких знаниях уже существует вне системы.

Для расширения списка возможностей, Notion предоставляет Notion API[7]. Данный инструмент позволяет экспортировать атрибуты документа во внешние сервисы, а также наоборот, встраивать новые сущности из различных источников внутрь своей структуры.

К сожалению, многие возможности Notion не применимы к возможностям ECM-систем, а сам продукт является платным.

Coda

Coda[8] – продукт для эффективного коллаборативного взаимодействия внутри команды. Как и Notion, Coda предоставляет широкие возможности по созданию и управлению знаниями внутри своей экосистемы. Так же, как и Notion, Coda оперирует понятием блока внутри своих документов. Каждый блок – самостоятельная единица информации, которая может менять свои свойства, а также взаимодействовать друг с другом.

Так же, как и у предыдущих решений, Coda имеет свой API[9] для расширения своих функциональных возможностей. В его возможности входит следующее:

- Поиск информации внутри документа.
- Создание нового документа и копирование предыдущих.
- Взаимодействие с блоками документа.

[7]https://developers.notion.com/

[8]https://coda.io/

[9]https://coda.io/developers/apis/v1

- Чтение, вставка, удаление информации из документа.

Подобно Notion, Coda имеет ряд возможностей, конфликтующие с возможностями ECM-системы.

slate.js

Slate[10] — фреймворк для создания богатых текстовых редакторов с открытым исходным кодом под лицензией MIT. Slate имеет следующие преимущества.

- Ориентированность на расширяемость — Slate предоставляет ядро и механизмы для добавления новых функциональных возможностей к этому ядру.

- Модель документа внутри редактора представляет собой структурированное дерево в формате JSON и основывается на объектной модели документа DOM, что, в свою очередь, предоставляет возможность создания сложных вложенных структур.

- Использование атомарных команд при модифицировании текста. Каждое изменение структуры документа создаёт команду внутри Slate. Такой подход позволяет явно следить за изменениями документа и облегчает разработку инструментов, зависящих от них (например, реализация истории состояний документа или совместное редактирование текста).

- Представление пользовательского интерфейса происходит с использованием технологии React[11], что даёт доступ к обширной документации инструмента, а также опыт крупного сообщества технологии в случае возникновения проблем при разработке интерфейса [6]. Более того, это даёт возможность облегчённого взаимодействия с теми системами, которые так же основаны на этой технологии. К подобным системам относится, например, веб-решение DocsVision.

Slate предоставляет текстовый редактор с минимальным количеством функциональных возможностей, но даёт инструменты для полной его кастомизации [1]. В сообществе существуют библиотеки различных расширений для Slate с

[10] `https://github.com/ianstormtaylor/slate`
[11] `https://reactjs.org/`

открытым исходным кодом под лицензией MIT[12], в которую входят расширения для форматирования и выравнивания текста, расширения для работы с таблицами, изображениями и другими структурами, расширения для сериализации/десериализации структуры документа Slate в HTML, расширения для идентификации структурных единиц и другие.

Данная технология уже зарекомендовала себя при создании гибкого текстового редактора для ECM-системы.

2.2 Вывод

Таким образом, так как расширения Google Docs являются не применимыми для задач ECM-системы, а Notion и Coda имеет многие возможности, избыточные по отношению к идеями ECM-систем, было решено реализовывать интеграцию системы знаний на базе гибкого текстового редактора, разработанный на основе технологии Slate.js.

[12]например, `https://github.com/udecode/slate-plugins`

3 Архитектура приложения

3.1 Встраивание системы управления знаниями в существующее приложение

По своей структуре приложение имеет клиент-серверную архитектуру, где на стороне клиента в первую очередь находится текстовый редактор, а на стороне серверной части — бизнес-логика взаимодействия между основными сущностями: пользователями, документами, блоками и атрибутами документа. Обновлённая архитектура приложения представлена на рис. 1.

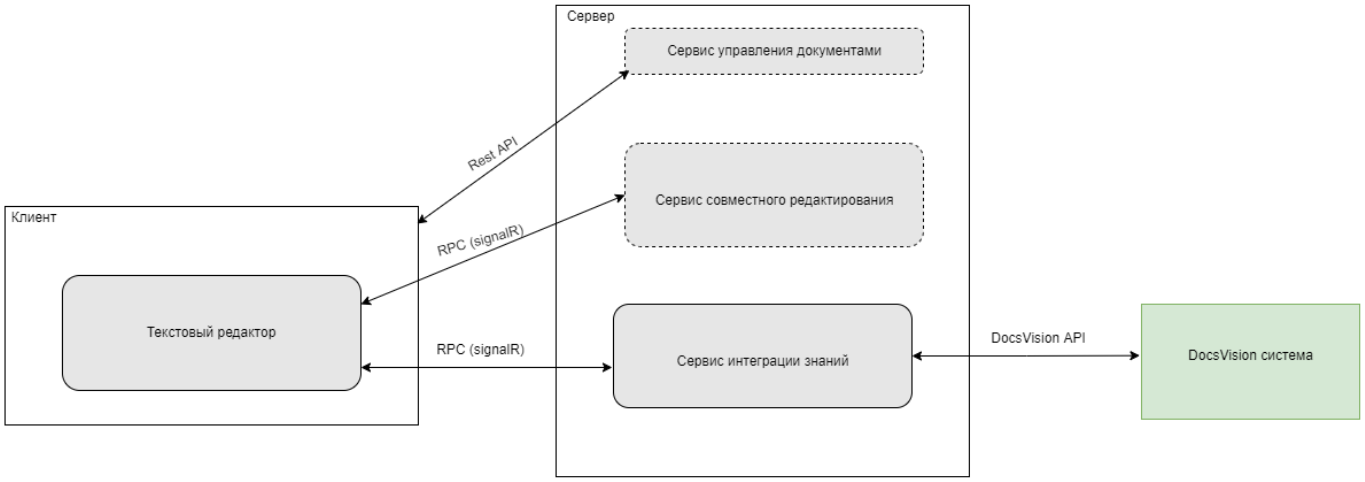

Рис. 1: Архитектура приложения

На изображении можно увидеть, как сервис, отвечающий за связывания системы знаний будет интегрирован с текстовым редактором с одной стороны, и с системой DocsVision с другой. Между клиентским приложением и серверным приложением-маршрутизатором связь устанавливается с помощью библиотеки SignalR [4]. Само же серверное приложение взаимодействует с DocsVision системой через DocsVision API. Сам же текстовый веб-редактор является частью клиентского приложения. Помимо доступа к текстовому редактору, на стороне клиента находятся интерфейсы для регистрации пользователей и просмотра информации о документах и блоках.

3.2 Интеграция гибкого текстового веб-редактора с системой Docsvision

Благодаря выбору технологий разработки веб-редактора, его встраивание в веб-решение системы является задачей тривиальной. Гибкий текстовый редактор был разработан на технологии Slate.js, который, в свою очередь, позволяет интегрироваться в React-приложения, коим и является веб-приложение системы DocsVision.

Таким образом, интегарция может быть произведена путём интегрирования React-компонента в веб-решение системы DocsVision.

3.3 Атрибутизация в текстовом редакторе

С точки зрения структуры документа, гибкий текстовый редактор работает с двумя типами объектов: промежуточный и текстовый. Промежуточные объекты могут состоять из других промежуточных объектов или из множества текстовых объектов. Текстовые объекты, в свою очередь, помимо самого текста могут содержать различные добавочные метки.

Документ состоит из блоков, каждый из которых, в свою очередь, имеет объединение различных промежуточных объектов. Листьями такого дерева обязательно будут текстовые объекты. Встраивание собственной структуры блока в систему slate.js происходит путём наследования базового промежуточного элемента.

Структура документа позволяет задавать дополнительные атрибуты для каждого объекта блока, что позволяет добавлять дополнительный смысл в те или иные структурные единицы документа. Так выражается знание о том, что данный текст, например, является идентификатором пользователя, или геопозицией, или временным интервалом и т. д.

Гибкий текстовый редактор позволяет переопределять отображение структурной единицы документа. Для этого, на этапе рендеринга электронной страницы можно явно указать (с помощью технологии React) вид отображаемого элемента. Для согласованности с единым стилем системы, было принято решение воспользоваться готовыми решениями отображения элементов у веб-библиотеки DocsVision.

Изображение встраивания React-компонентов на этапе рендеринга доку-

мента представлено на рис. 2

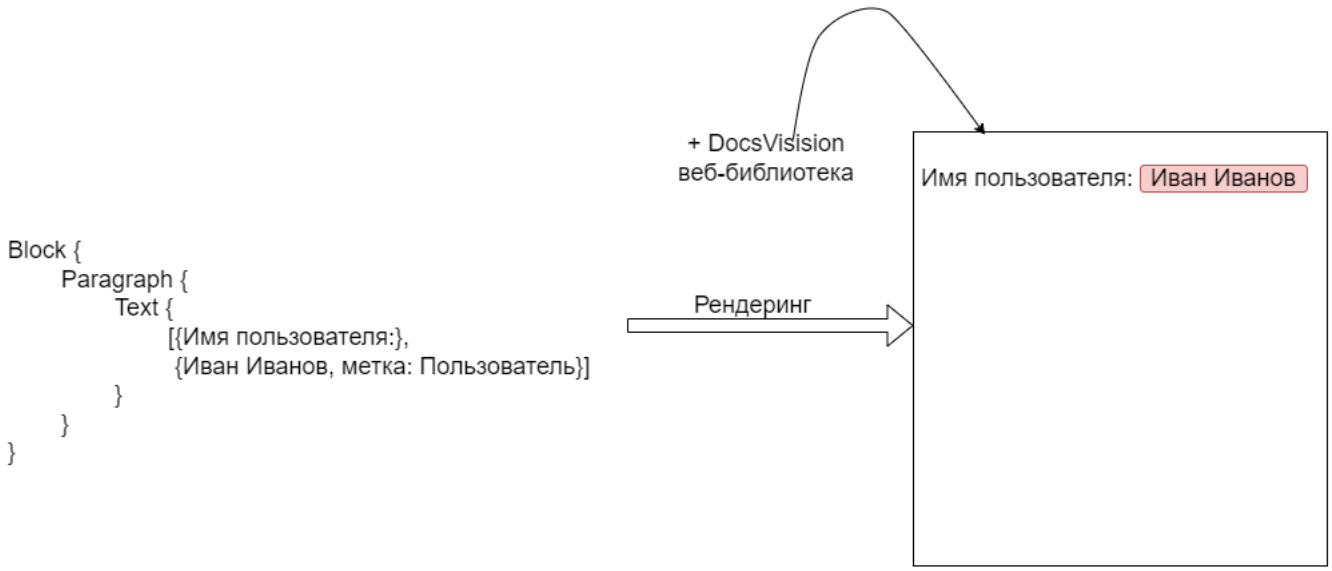

Рис. 2: Встраивание React-компонентов веб-библиотеки DocsVision

4 Заключение

В ходе данной работы были достигнуты следующие результаты.

- Проанализированы существующие решения для интеграции систем знаний: Google Docs, Notion, Coda и гибкий текстовый веб-редактор на основе Slate.js. Последний был выбран в качестве продукта, способный нативно встроиться в экосистему ECM-платформы Docsvision.
- Спроектирована архитектура приложения: серверная часть приложения, взаимодействие с существующими сервисами текстового веб-редактора.
- Спроектирована архитектура веб-редактора: взаимодействия с атрибутами, протокол общения между сервисами, встраивание в веб-решение DocsVision.

Список используемой литературы

[1] Slate documentation. — URL: `https://docs.slatejs.org/` (дата обращения: 14.12.2021)

[2] ASP.NET Core documentation. — URL: `https://docs.microsoft.com/aspnet/core/` (дата обращения: 17.12.2021)

[3] Dino Esposito. *Programming ASP.NET Core (1st Edition)* — Microsoft Press, 2018

[4] ASP.NET Core SignalR documentation. — URL: `https://docs.microsoft.com/aspnet/core/signalr` (дата обращения: 25.04.2021)

[5] Entity Framework Core documentation. — URL: `https://docs.microsoft.com/ef/core` (дата обращения: 22.12.2021)

[6] React documentation. — URL: `https://reactjs.org/docs` (дата обращения: 22.12.2021)

[7] TypeScript Documentation. — URL: `https://www.typescriptlang.org/docs/` (дата обращения: 23.12.2021)

[8] Google Docs API Documentation. — URL: `https://developers.google.com/docs/api` (дата обращения: 23.12.2021)

[9] Notion API Documentation. — URL: `https://developers.notion.com/` (дата обращения: 23.12.2021)

[10] Coda API Documentation. — URL: `https://coda.io/developers/apis/v1` (дата обращения: 23.12.2021)