Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем
Направление подготовки 09.04.04 Программная инженерия
Направленность (профиль) образовательной программы Управление разработкой программного обеспечения

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. кафедрой
_____ А.В. Бушманов
«_____»_____ 2023 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

На тему: Разработка системы управления контентом для web-сайтов.

Исполнитель
студент группы 157-ом _____ А.А. Ситиков
(подпись, дата)

Руководитель
Доцент, канд. техн. наук _____ Л.В. Никифорова
(подпись, дата)

Руководитель научного
содержания программы
магистратуры
Профессор, доктор техн. наук _____ И.Е. Еремин
(подпись, дата)

Нормоконтроль
Доцент, канд. техн. наук _____ Л.В. Никифорова
(подпись, дата)

Рецензент
Доцент, канд. техн. наук _____ Д.С. Щербань
(подпись, дата)

Благовещенск, 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательная учреждение высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет математики и информатики
Кафедра информационных и управляющих систем

УТВЕРЖДАЮ
Зав. кафедрой
_____ А.В. Бушманов
подпись
«_____»_____2023 г.

З А Д А Н И Е

К магистерской диссертации студента группы 157-ом
Ситикова Алексея Александровича

1. Тема магистерской диссертации: Разработка системы управления контентом для web-сайтов.

(Утверждено приказом от 21.02.2023 № 442-уч)

2. Срок сдачи студентом законченной работы (проекта) 20.06.2023

3. Исходные данные к магистерской диссертации: Исторические описания, проектная документация, фотоснимки, учебная литература, интернет ресурсы.

4. Содержание магистерской диссертации (перечень подлежащих разработке вопросов): анализ предметной области; поиск существующих решений; моделирование объекта исследования

5. Перечень материалов приложения: (наличие чертежей, таблиц, графиков, схем, программных продуктов, иллюстративного материала и т.п.): Нет.

6. Рецензент магистерской диссертации:

7. Дата выдачи задания 30.01.2023

Руководитель выпускной квалификационной работы: Л.В. Никифорова, доцент, доктор техн. наук
Задание принял к исполнению (30.01.2023) _____

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

CMS (системы управления контентом) используются для работы с содержимым web-ресурсов: наполнения, корректировки, удаления ненужной информации, и т.д.

Если необходимо настроить удобное изменение информации некоторых блоков сайта, даже с учетом распределения доступа, CMS будет идеальным вариантом. Основное требование – это интуитивно понятный для пользователя интерфейс и простой способ сохранения информации, с последующим отображением ее на сайте.

Актуальность работы заключается в том, что существует большое количество CMS систем, платных и бесплатных. Но даже рассматривая бесплатные варианты, становится понятно, что существующие методы решений охватывают не все задачи необходимости интеграции системы управления контентом, в частности, большой “пробел” возникает в вопросе встраивания CMS в готовые web-сайты, перенос которых сильно затруднен, либо невозможен на заданные движки.

В связи с этим появляется необходимость в разработке простой CMS с удобной интеграцией на сайт, без функциональной перегруженности.

Разрабатываемая система управления контентом призвана закрыть данные “пробелы” всеми возможными средствами HTML и JavaScript языка, поддерживающиеся на любом web-сайте, просматриваемом через любой web-обозреватель.

Предполагается несколько вариантов встраивания:

– JSON запрос к серверу API – исходный сайт делает обращение к серверу, запрашивая необходимую страницу, которую затем отрисовывает в необ-

ходимом месте. Этот способ хорошо подходит для интеграции на стадии разработки, либо в более продвинутых, современных сайтах с широким библиотечным функционалом;

– Фреймовая вставка – в исходный сайт встраивается iframe, поддерживающийся в рамках HTML языка, который обращается по ссылке к необходимой странице. Данный способ подходит в тех случаях, когда первый вариант встраивания не подходит и имеется доступ к исходному коду страниц, в котором и делается фреймовая вставка.

– Ссылочная вставка – на сайте размещается гиперссылка на страницу рендеринга запрашиваемой страницы. Данная система рендеринга так же будет разработана в рамках диссертационной работы и является непосредственной частью системы. Это универсальный вариант, подходящий в любых ситуациях.

Объект исследования – система управления контентом web-сайтов.

Предмет исследования – методы встраивания редактируемых блоков информации на готовые web-страницы.

Целью магистерской диссертации является: Разработка системы, предназначенной для предоставления пользовательского интерфейса при редактировании блоков информации на сайте.

Для выполнения поставленной цели необходимо решить следующие задачи:

– провести анализ предметной области и аналогов решения;

– выполнить проектирование системы управления контентом;

– разработать систему управления контентом;

– протестировать реализованную систему.

В качестве методов научного исследования используется метод анализа, для детализации изучения предметной области и сравнение, для оценки аналогов решения задач.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ИССЛЕДУЕМОЙ ЗАДАЧИ

1.1. Предметная область и объект проводимого исследования

CMS (от английского Content Management System) – система управления содержимым (контентом) – компьютерная программа или информационная система, которая используется для организации и обеспечения процесса по совместному созданию, управлению и редактированию содержимого сайта [3].

Системы управления контентом (CMS) используются для работы с содержимым web-ресурсов: наполнения, корректировки, удаления ненужной информации, и т.д.

Если необходимо настроить удобное изменение информации некоторых блоков сайта, даже с учетом распределения доступа, CMS будет идеальным вариантом. Основное требование – это интуитивно понятный для пользователя интерфейс и простой способ сохранения информации, с последующим отображением ее на сайте [5].

Система управления контентом – это комплексная программа, которая облегчает создание и управление веб-сайтами любой сложности для людей, не имеющих навыков в программировании.

Применение CMS позволяет существенно ускорить процесс создания и обновления сайта, а также упростить работу с контентом без необходимости знания программирования. Рынок CMS на данный момент очень разнообразен, и каждая система имеет свои уникальные особенности и возможности, подходящие для различных типов сайтов. При выборе подходящей CMS необходимо учитывать потребности вашего сайта, его цели и структуру, а также обращать внимание на удобство работы в системе для конкретных пользователей.

Также стоит обратить внимание на то, что различные CMS могут иметь разный уровень сложности и требования к техническим знаниям для их установки и настройки. Некоторые системы, такие как WordPress, имеют широкое

сообщество пользователей и разработчиков, что делает их более простыми в использовании, а также предоставляет доступ к большому количеству дополнительных плагинов и тем для улучшения функциональности и внешнего вида сайта. Однако более сложные CMS могут предоставлять больше возможностей для разработки крупных и сложных проектов, но требуют больше технических знаний и опыта работы с системами управления сайтом. Поэтому необходимо тщательно выбирать CMS, основываясь на потребностях сайта и уровне технической экспертизы, доступной для его обслуживания. [2].

Основные виды CMS систем можно разбить на несколько классификаций:

– Студийные CMS

Студийные CMS – это тип CMS-систем, которые разрабатываются для создания и управления многофункциональными и сложными веб-сайтами. Эти системы применяются в больших веб-студиях для профессиональной разработки не только легких сайтов-визиток, но и крупных сайтов, электронных коммерций и других сложных проектов.

– Тиражные CMS

CMS, разработанные специализированными компаниями или веб-студиями для последующей массовой продажи конечным пользователям или другим веб-студиям, которые по каким-либо причинам не имеют собственной студийной CMS.

Обычно имеют ограниченные функциональные возможности, по сравнению с другими типами CMS. Они обычно включают только основные функции управления контентом, такие как создание страниц, публикация текстов, изображений и простых форм. Они часто не имеют продвинутых функций, таких как возможность создания сложных форм, комментариев, обратных звонков и других интерактивных функций.

– Open Source CMS (бесплатные)

Системы управления, которые разрабатываются сообществами программистов, поддерживаются ими же и имеют открытый исходный код. Любой программист может взять бесплатную CMS и доработать ее по собственному усмотрению.

Минусом подобных CMS является их массовость и как следствие общеизвестность всех «дыр» в безопасности, что часто приводит к заражению сайтов на Open Source CMS вирусами, что может грозить длительным падением позиций сайта в результатах выдачи поисковиков [7].

1.2. Анализ существующих методов решения аналогичных задач

Перед человеком, который планирует создать сайт, в первую очередь, стоит задача подобрать лучшую CMS, которая наиболее максимально подойдет для его проекта.

В современном мире существует множество Content Management System, различающиеся своими функциональными возможностями и управляемостью. Наиболее богатой функциональностью обладают пять очень распространенных CMS на русскоязычном пространстве интернета:

– Joomla;
– 1С-Битрикс;
– WordPress;
– uCoz;
– SiteEdit.

От правильного выбора CMS для сайта напрямую зависит его судьба, так как появляется зависимость в постоянных обновлениях описанных систем и удобство масштабирования.

Существует большое количество готовых систем управления контентом для сайтов, в том числе и бесплатных. Они могут быть классифицированы в соответствии с тремя типами работы:

1) Локальные системы управления контентом (LCMS). Это программы, работающие на локальном компьютере и обладающие всеми необходимыми плагинами и инструментами для управления контентом. Все изменения вносятся локально и затем загружаются на сервер.

2) SaaS-системы управления контентом (Software as a Service). Это программы, работающие на удаленном сервере и предоставляющие доступ к своим функциям через веб-браузер. Пользователю не нужно заботиться о настройке программного обеспечения, и он может легко управлять контентом через браузер.

3) Платформы управления контентом с открытым исходным кодом (Open Source CMS). Это программы, распространяющиеся бесплатно и имеющие открытый исходный код, который можно изменять и дополнять по желанию. Пользователи могут создавать свои собственные темы и плагины и распространять их онлайн. Системы с открытым кодом, такие как WordPress и Joomla, являются самыми популярными.

Каждый из этих типов CMS имеет свои преимущества и недостатки. Выбор подходящей системы зависит от требований и опыта пользователя в управлении сайтом.

Локальные системы управления контентом могут быть удобны для разработчиков и дизайнеров, так как они имеют прямой доступ к файлам и могут быстро изменять код. Однако, настройка локальной системы управления контентом может быть сложной задачей для новичков в программировании.

SaaS-системы управления контентом позволяют пользователям управлять сайтом из любого места, где есть доступ к интернету, и не требуют установки на компьютер. Однако, они могут быть дороже в использовании и не всегда предоставляют полный доступ к коду.

Платформы управления контентом с открытым исходным кодом наиболее гибкие и доступные для пользователей. Они имеют большое количество плагинов и тем, которые могут быть использованы для расширения функциональности

сайта. С другой стороны, возможности пользователей могут быть ограничены недостатком знаний в программировании, что может повлечь за собой ошибки в настройке сайта.

CMS 1С-Битрикс. Система ориентирована на корпоративные сайты, информационные и справочные порталы, социальные сети, интернет-магазины, сайты СМИ, пригодна для создания других видов веб-ресурсов.

Для хранения данных сайта используется реляционная СУБД. Поддерживаются следующие СУБД: MySQL, Oracle, MS SQL. Продукт работает на Microsoft Windows и UNIX-подобных платформах, включая Linux.

«1С-Битрикс: Управление сайтом» продаётся в одной из семи составленных фирмой-разработчиком редакций (Старт, Стандарт, Эксперт, Малый бизнес, Бизнес, Портал, Большой бизнес), определяющих набор модулей и функциональность системы.

Чтобы работать с системой управления, пользователи, не знакомые с HTML и PHP, должны сначала настроить систему, создать шаблон, основанный на графическом дизайне, определить структуру разделов и страниц, а также подключить модули. При этом могут помочь партнеры компании "1С-Битрикс".

Идеология системы представляет собой разделение логики на модули и компоненты.

Модули в «1С-Битрикс: Управление сайтом» - это набор программных компонентов, отвечающих за работу с различными типами баз данных, а также предоставляющих унифицированный API системы.

Компоненты служат для связи конечного представления информации на сайте с программным ядром системы. Они используют API, созданный модулями, для организации выборки, модификации, управления информацией в базе данных.

Преимущества

Для разработчика одной из особенностей системы "1С-Битрикс: Управление сайтом" является механизм информационных блоков (инфоблоков), который

упрощает создание пользовательских типов контента, например, для разных каталогов. При этом способы работы с инфоблоками в "1С-Битрикс" похожи на способы работы со справочниками в платформе "1С: Предприятие" [11].

Современные версии "1С-Битрикс" также обладают сильным визуальным HTML-редактором, который позволяет размещать на странице обычную HTML-информацию, PHP-код, а также динамические компоненты, управляемые системой управления контентом. Тем не менее, у этого редактора есть некоторые ограничения.

Данную систему обычно критикуют по следующим позициям:

– По сравнению с бесплатными аналогами, Битрикс достаточно медленная и требовательная к ресурсам система. Может генерировать чересчур сложные SQL запросы.

– Некоторая сложность, перегруженность системы для пользователя.

– Нестабильность системы, в частности возможны проблемы после обновления версии CMS.

– Битрикс и рекламируется как очень надёжная система, в ней были обнаружены (и позднее исправлены) ряд потенциальных уязвимостей.

– Компания 1С продвигает "1С-Битрикс" как систему, которая не требует знания программирования, но для тонкой настройки сайта все равно приходится программировать, как минимум, так часто, как в конкурирующих бесплатных CMS. Освоение API "1С-Битрикс" может быть сложным, из-за ориентации в исходном коде и его тяжелом стиле. Кроме того, большинство современных Open Source CMS генерируют документацию с помощью PHP Documentor или Doxygen, что гарантирует ее актуальность и доступность для просмотра исходного кода.Внутреннее устройство Битрикса характеризует ряд устаревших архитектурных решений.

Хотя система управления веб-контентом ориентирована на пользователей, не имеющих опыта в программировании, некоторые простые функции в "1С-

Битрикс" предполагают самостоятельное решение, что может быть сложно без хорошего знания программирования [14].

За последнее время WordPress стал популярен среди пользователей Интернета, и теперь есть много проектов, платных и бесплатных, которые позволяют любому желающему создать свой собственный блог. Блог-системы работают по принципу публикации сообщений или новостей на личной странице, которые затем отображаются в новостной ленте и подразделяются на категории или рубрики. Все посетители блога могут прочитать публикации и оставить комментарии, если владелец блога это разрешил.

Возможности:

– простота установки, настройки;

– поддержкаRSS, Atom, trackback, pingback;

– подключаемые модули (плагины) с уникальной простой системой их взаимодействия с кодом;

– поддержка так называемых «тем», позволяющих легко менять как внешний вид, так и способы вывода данных;

– «темы» реализованы как наборы файлов-шаблонов на PHP, что положительно сказывается на скорости и гибкости;

– громадные библиотеки «тем» и «плагинов»;

– заложенный потенциал архитектуры позволяет легко реализовывать сложные решения;

– наличие ЧПУ (человеко-понятный URL);

– наличие русских переводов.

WordPress может часто обращаться к базе данных, что может привести к ухудшению производительности на слабых хостингах. Для поддержания большого количества посетителей необходимо настроить кэширование и оптимизировать запросы к базе данных [15].

uCoz – это бесплатная система управления сайтом. Её концепция основана на модульной структуре, которые можно использовать как отдельные элементы выстраивая из них полноценный сайт. В рейтинге самых популярных сайтов в России по данным Alexa Internet, uCoz.ru занимает 15-е место.

Основные возможности системы:

– На выбор предоставляется 246 дизайнов (шаблонов) для создания сайта.

– Возможно создать собственный дизайн (шаблон), либо переделать любой стандартный.

– При регистрации выделяется 400 мегабайт дискового пространства. Дисковое пространство увеличивается с ростом числа посетителей и жизни сайта, при желании закачивать большие файлы можно привязать аккаунт Depositfiles.

– Предоставление домена третьего уровня в 21 различной зоне.

– Возможность привязать к сайту собственный домен.

– Неограниченное создание/редактирование MX записей, и создание субдоменов, после прикрепления домена.

– Доступ по FTP.

– WYSIWYG online редактор.

– Визуальный конструктор блоков.

– Версия сайта для PDA.

– Резервное копирование.

– RSS импорт и экспорт.

– Лайтбокс.

– ЧПУ.

– Общая авторизация - uID (unet).

– И др.

Платные возможности

– Снятие копирайта системы

– Снятие рекламного баннера

– Снятие рекламы в панели управления

– Увеличение дискового пространства до 10 Гб

– Возможность скрытия счетчика uCoz (в случае использования встроенного модуля статистики)

– Возможность прикрепления файлов к письмам отправляемым через E-mail формы

После регистрации в системе вы сразу получаете:

– Имя для сайта в интернете, которое выбираете сами.

– Место для сайта

– Систему управления содержимым

– Готовые дизайны для сайта + возможность управлять ими

– uNet профиль

Особенности

uCoz является системой SaaS и имеет характерные для этого типа системы особенности, такие как закрытый исходный код, отсутствие возможности загружать серверные скрипты и базы данных (хотя это возможно в рамках концепции Web 3.0 PaaS), но компенсируется большим количеством встроенных возможностей. Хотя можно использовать статические HTML-страницы, это не рекомендуется, так как существуют некоторые ограничения на их использование и загрузку.

Для дизайнеров, которые не обладают навыками программирования, может быть сложно перенести сайт, созданный на uCoz, на другую платформу. Кроме того, ограничением является невозможность использования языков программирования PHP, Perl и ASP, но корпоративный блог сообщает, что в скором времени использование PHP станет возможным.

Однако, для новичка, которому просто нужна удобная площадка для своей деятельности в сети Интернет, uCoz представляет собой достаточно привлекательный вариант [17].

На основе анализа функциональных возможностей каждой из CMS-систем, становится понятно, что:

– Joomla – бесплатная CMS с ярким и практичным интерфейсом, которая регулярно обновляется разработчиками, но имеет низкую скорость работы.

– 1С-Битрикс – это платный модуль, разработанный известной компанией 1С. В настоящее время он доступен в различных редакциях, и его стоимость может существенно отличаться в зависимости от функциональных возможностей ПО.

– WordPress – бесплатная система, которая изначально была создана для создания блоговых сайтов. Благодаря своей простоте и функциональности она стала очень популярной и сейчас используется для сайтов различных типов.

– uCoz - это бесплатная система управления сайтом и хостинг, которые помогают создать сайт. uCoz содержит модули, которые можно использовать как отдельные элементы, например, блоговую платформу или интернет-магазин, а также в совокупности для создания полноценного сайта.

– SiteEdit - это система создания и управления сайтом. Она предоставляет все возможности для создания сайтов и имеет функции для создания интернет-магазина, форума и доски объявлений. Кроме того, она включает систему продвижения сайта в поисковых системах.

В связи с чем можно выделить основных характеристики каждой системы управления контентом, а именно:

– Удобство работы с контентом. очень важно для веб-мастеров, которые ежедневно публикуют много информации. Для экономии времени необходимо

упростить процесс работы с контентом, включая понятный визуальный интерфейс, быстрое добавление и форматирование изображений, возможность редактирования в режиме HTML и другие функции.

– Простота изменения структуры. Добавление или редактирование разделов и категорий, а также установка меток должны быть легкими задачами. Чем проще механизм структуры системы управления контентом, тем лучше.

– Расширяемость. Важнейший критерий при выборе CMS, поскольку стандартный функционал движка может быть недостаточным. Важно убедиться, что платформу можно расширять всеми необходимыми параметрами без какого-либо ущерба для ее производительности.

– Широкие возможности в отношении дизайна. Идеально, чтобы создание эффектного и привлекательного дизайна было минимально затратным и заняло мало времени.

– Безопасность. Чрезвычайно важна для того, чтобы пользователи и поисковики доверяли CMS, и не подозревали в сомнительности источника.

– Внедряемость. Система должна иметь возможность легко интегрироваться на сайт, используя различные способы встраивания.

На основе данных критериев оценки, можно дать характеристику выбранным, наиболее популярным CMS-системам:

– Joomla предоставляет широкий функционал для работы с контентом сайта, будь то новостная лента, сайт-визитка или иной информационно-направленный web-сайт с динамически обновляемым контентом. Так же является достаточно расширяемым и имеет множество готовых шаблонов для встраивания. Но, в силу изобилия функционала, имеет перегруженный интерфейс, требующий большого времени для изучения данной системы. Так же имеет большую брешь в безопасности в виде множества уязвимостей, из-за которых можно получить несанкционированный доступ к сайту клиента.

– 1С-Битриск является платной системой и в основном наследует все преимущества и недостатки Joomla, за исключением вопроса безопасности, являясь более надёжной.

– WordPress является наиболее популярной бесплатной системой, предоставляющая более упрощенный интерфейс, в отличии от Joomla и 1С-Битрикс, но в остальном во всём им проигрывает. Особенно критичное звено данной системы – безопасность. Архитектура имеет множество “дыр” и уязвимостей, чем злоумышленники часто пользуются.

– uCoz – система, изжившая свою популярность в ~2010 годах, которую сменили более “сильные” конкуренты, предоставляющие функционал намного шире. Кроме простоты создания сайта мало что может предложить. Так же имеет самый низкий уровень безопасности из всех описанных CMS.

– SiteEdit представляет собой систему для создания сайта, имеющий скудный функционал относительно других CMS систем. Так же является платным.

Нагляднее сравнить характеристики CMS систем можно по таблице 1.

Таблица 1 – Анализ CMS систем

	Удобство	Структуризация	Бесплатно	Функционал	Безопасность
Joomla	-	+	+	+	-
1С-Битрикс	-	+	-	+	+
WordPress	+	-	+	-	-
uCoz	+	-	+	-	-
SiteEdit	+	-	-	-	-

Помимо описанных критериев оценок, все эти системы управления контентом имеют общую критическую черту – требование переноса готового сайта под заданную систему, что иногда не представляется возможным. Это является наиболее важным требованием в вопросе интеграции CMS.

В рамках диссертационной работы планируется разработать альтернативную CMS систему, обладающую, по возможности, всеми описанными качествами наиболее популярных систем, и, главное, имеющую возможность встраиваться в готовые web-сайты, без требования к переносу.

2 АЛГОРИТМИЧЕСКОЕ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ РЕШЕНИЯ ЗАДАЧИ

При описании алгоритмов решения задач, необходимо выделить основные функции, пользователей и технологию разработки проектируемой системы

2.1. Функции разрабатываемой системы

Основными функциями системы являются:

- авторизация;
- создание пользователя;
- редактирование прав доступа пользователю;
- создание/редактирование/удаление раздела;
- прием текстовой, графической и файловой информации на сервер.

2.1.1. Авторизация

Функция авторизации позволяет идентифицировать и верифицировать пользователя в системе. На вход подается логин и пароль, после чего идет проверка по базе данных, существует ли данный пользователь. В случае отказа, на экран выводится ошибка. В случае успеха происходит перенаправление на главную страницу в клиентской части системы.

2.1.2. Создание пользователя

Данная функция позволяет вносить новых пользователей в систему. На вход подается имя пользователя, логин и пароль, после чего в базе данных происходит проверка, существует ли данный логин в базе. Если логин занят, то на экран клиентской части системы выводится ошибка. В ином случае создается новая запись в базе данных.

2.1.3. Редактирование прав доступа пользователю

С помощью данной функции реализуется возможность управлением прав доступа к различным разделам системы. На вход подается id пользователя и массив id разделов, к котором предполагается выдача прав доступа. В базе проверяется, существуют ли данные id. В случае успеха, запись заносится в базу. Если id не найдено, то на экран клиентской части системы выводится ошибка.

2.1.4. Создание раздела

Данная функция отвечает за создание нового раздела в базе данных. На вход подается название раздела, маршрут раздела и id родительского элемента. В базе данных проверяется существование указанного id и, в случае успеха, запись заносится в базу. В ином случае на клиентскую часть системы выводится ошибка.

2.1.5. Редактирование раздела

Функция отвечает за изменение названия и/или маршрута раздела. При вызове данной функции происходит обращение к id раздела посредством get-метода. На вход подается название раздела и маршрут раздела. В базе данных проверяется, существует ли данный id и, в случае успеха, запись обновляется. В противном случае на клиентскую часть системы выводится ошибка.

2.1.6. Удаление раздела

С помощью данной функции происходит удаление раздела из базы данных. На вход подается id раздела, после чего проверяется его наличие в базе. В результате, в случае успеха, запись из базы удаляется, в противном случае на клиентскую часть системы выводится ошибка.

2.1.7. Прием текстовой информации на сервер

Данная функция отвечает за запись текстовой информации в формате html в базу данных. На вход подается id раздела и html код страницы. В базе данных проверяется, существует ли указанный id раздела и в случае успеха, запись сохраняется в базу. Иначе на клиентскую часть системы выводится ошибка.

2.1.8. Прием графической информации на сервер

Функция отвечает за приём графического файла в формате blob, после чего объект сохраняется на сервере и генерируется ссылка к этому объекту, возвращаемая в качестве результата запроса. В случае успеха клиентской части возвращается ссылка до графического объекта, иначе выводится ошибка.

2.1.9. Прием файловой информации на сервер

С помощью данной функции происходит прием файловой информации на сервер в формате blob. Объект сохраняется на сервере и генерируется ссылка к этому объекту, возвращаемая в качестве результата запроса. В случае успеха клиентской части возвращается ссылка до файлового объекта, иначе выводится ошибка.

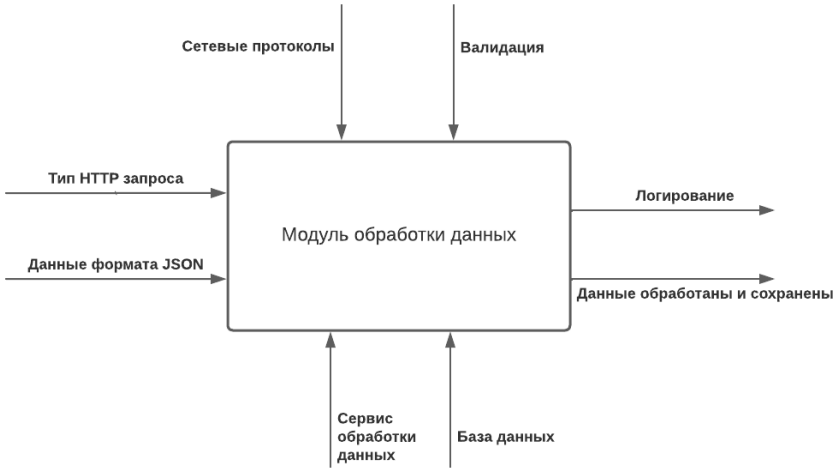

Рисунок 1 – Функциональная модель модуля обработки данных

Функциональная модель модуля обработки данных представлена на рисунке 1. Обобщенный пример цепочки вызовов представлен на рисунке 2.

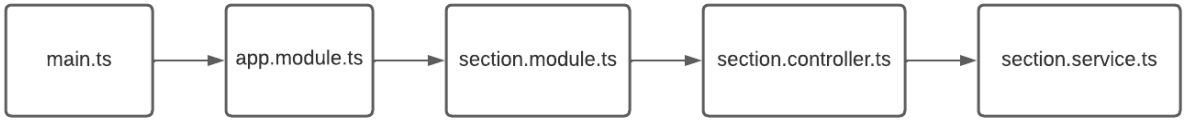

Рисунок 2 – Обобщенная цепочка обработки вызова

Цепочка состоит из пяти элементов:

– главный компонент запуска сервера main.ts;

– компонент регистрации модулей приложения app.module.ts;

– компонент регистрации модулей section.module.ts;

– компонент реакций на обращения к серверу section.controller.ts;

– компонент реализации алгоритмов обработки входных данных section.service.ts;

Диаграмма вариантов использования, также известная как диаграмма сценариев поведения или прецедентов, представляет собой исходное концептуальное представление системы на этапе ее проектирования и разработки. Она включает в себя актеров, варианты использования и отношения между ними, а также общие элементы нотации, такие как примечания и механизмы расширения.

Суть диаграммы заключается в том, что проектируемая система представлена как множество актеров, которые взаимодействуют с системой через варианты использования. Актером может быть любой объект, субъект или система, которые взаимодействуют с моделируемой системой извне. Каждый вариант использования определяет набор действий, которые система должна выполнить при взаимодействии с актером. При этом диаграмма не отражает, каким образом эти действия будут реализованы [8].

Диаграмма вариантов использования представлена на рисунке 3.

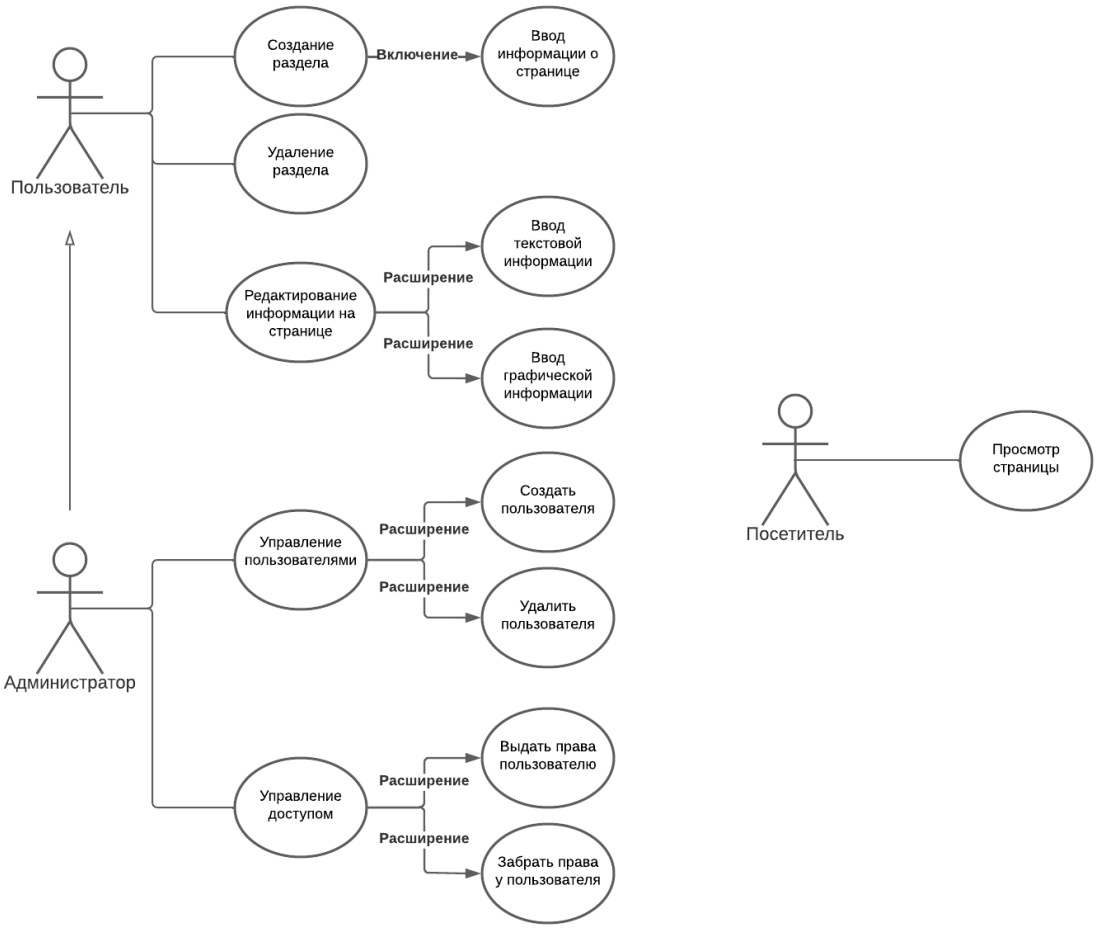

Рисунок 3 – Диаграмма вариантов использования

В рамках описываемой системы актерами выступают:

– администратор;

– пользователь;

– посетитель.

Каждый из них выполняет свои функции при работе в системе, однако администратор имеет наивысший ранг доступа к функциональным частям системы, поэтому он наследует все функциональные возможности.

2.2. Обоснование выбора программно-технического обеспечения

В настоящее время у программистов есть множество средств разработки и проектирования, из которых они могут выбрать. В каждом случае выбор средств зависит от того, какие задачи нужно решить. Программисты выбирают наиболее подходящий набор инструментов для конкретной задачи.

В рамках данной работы были использованы средства разработки семейства JetBrains:

– Среда разработки IntelliJ IDEA для серверной части.
– Среда разработки WebStorm для клиентской части системы.
– IDE DataGrip для работы с базой данных.

2.2.1. Среда разработки IntelliJ IDEA

JetBrains IntelliJ IDEA - это интегрированная среда разработки (IDE) на языке Java, которая предлагает широкие возможности для создания высококачественных приложений. Она содержит интеллектуальный редактор, мощные средства для обработки кода, поддержку технологий J2EE, возможность интеграции с системами управления версиями и среду тестирования, уникальный инструмент для проверки и оптимизации кода, а также инновационный конструктор графических интерфейсов. Считается одной из лучших IDE для Java и широко используется в разработке приложений и веб-сервисов. IntelliJ IDEA предлагает также множество удобных функций, которые позволяют сократить время на выполнение рутинных операций: автодополнение кода, быстрый переход к определениям методов, рефакторинг кода, автоматическое форматирование кода, подсветка синтаксиса и т.д. Все эти функции позволяют ускорить процесс разработки и уменьшить количество ошибок [5].

Преимущества:

– Умное автодополнение, инструменты для анализа качества кода, удобная навигация, расширенные рефакторинги и форматирование для Java, Groovy,

Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML и многих других языков.

Поддержка всех популярных фреймворков и платформ, включая Java EE, Spring Framework, Grails, Play Framework, GWT, Struts, Node.js, AngularJS, Android, Flex, AIR Mobile и многих других.

– Интеграция с серверами приложений, включая Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty и Virgo.

– Инструменты для работы с базами данных и SQL файлами, включая удобный клиент и редактор для схемы базы данных.

– Интеграция с коммерческими системами управления версиями Perforce, Team Foundation Server, ClearCase, Visual SourceSafe.

– Инструменты для запуска тестов и анализа покрытия кода, включая поддержку всех популярных фреймворков для тестирования [18].

2.2.2. Среда разработки WebStorm

JetBrains WebStorm – интегрированная среда разработки (IDE) на языке JavaScript, которая предназначена для разработки веб-приложений. Она содержит редактор кода, инструменты для отладки и тестирования JavaScript, HTML и CSS, а также возможности для работы с системами контроля версий. WebStorm позволяет разработчикам работать с широким набором фреймворков и библиотек, включая AngularJS, React, Vue.js, Ember и другие. Она является одной из наиболее популярных IDE для веб-разработки и используется как в небольших, так и в больших проектах во многих отраслях промышленности.

WebStorm предлагает ряд удобных функций, которые позволяют сократить время на выполнение рутинных операций, такие как автодополнение кода, быстрый переход к определениям методов, подсветка синтаксиса и т.д. Она также

поддерживает технологии Live Edit и Spy-js, которые позволяют получить быструю обратную связь при разработке приложений без необходимости перезагрузки страницы.

WebStorm также содержит ряд интеллектуальных функций, таких как улучшенное авто-импортирование, подсказки по параметрам функций, рендеринг JSX-компонентов и многое другое, что позволяет разработчикам сосредоточиться на написании проекта вместо того, чтобы тратить время на ручное написание кода. Кроме того, WebStorm удобен для работы в команде, так как он интегрируется с популярными системами контроля версий, такими как Git, SVN, Mercurial, Perforce и другие, а также имеет встроенную поддержку сборщика проектов таких как Gulp, Grunt и npm.

IDE WebStorm обладает мощной и гибкой настройкой, что позволяет разработчикам настроить его под свои потребности. Кроме того, он предлагает широкий набор тем оформления, чтобы разработчики могли выбрать подходящий для себя дизайн интерфейса.

Преимущества:

– Модификация файлов .css, .html, .js с одновременным просмотром результатов;

– отладка кода на JavaScript;

– удалённое развёртывание по протоколам FTP, SFTP, на монтированных сетевых дисках и т. д. с возможностью автоматической синхронизации;

– интеграция с системами управления версиями: Subversion, Git, GitHub [16].

2.2.3. IDE DataGrip

DataGrip - это интегрированная среда разработки (IDE), созданная компанией JetBrains, специально для работы с базами данных. Она поддерживает множество СУБД (систем управления базами данных), включая MySQL, PostgreSQL,

Oracle, SQL Server, SQLite и многое другое. Предоставляет широкий набор инструментов для работы с базами данных, включая подключение к БД, создание и редактирование запросов, авто-заполнение кода и многое другое. Она также поддерживает большое количество форматов данных, таких как JSON, XML, HTML, CSV и многое другое.

С помощью DataGrip можно выполнять запросы в реальном времени, изменять данные с помощью встроенного редактора или использовать внешние приложения для редактирования и просмотра данных без нужды переключения между режимами. Также, IDE предоставляет аналитические функции, такие как поиск и устранение ошибок в запросах, автоматический вывод таблиц и отображение взаимосвязи между таблицами в базе данных, а также возможности для тестирования баз данных.

DataGrip обладает мощной системой настройки и интеграции, что позволяет интегрировать ее с другими инструментами разработки и системами контроля версий, такими как Git, SVN, Mercurial и другие. Это значительно упрощает процесс разработки, тестирования и внедрения баз данных.

Возможности:

– Генерирование кода и элементы БД

JetBrain DataGrip – платформа для обработки элементов БД. В случае, когда пользователь имеет дело с таблицами и производит в них различные действия, включающие редактирование колонок, использование графического интерфейса DataGrip будет очень уместным. Такие корректировки предполагают создание конкретного скрипта. Так, корректировки производятся непосредственно в базе, или же прямо в редактор копируется сгенерированный DDL запрос, и дальнейшая работа осуществляется впоследствии с самим кодом.

– Автоматическое дополнение

DataGrip ориентирована на поддержание автоматического дополнения кода, что положительно отражается на скорости написания запросов. Во время

набора кода IDE распознает контекст и выполняет привычные для юзера операции – оказывает помощь при прописании кода, учитывая конкретные ключевые слова и наименования элементов базы данных. Наряду с этим программа принимает во внимание зависимости при прописывании JOIN, делает подсказки, касающиеся типов параметров при осуществлении функций, описывает табличную структуру в предложениях INSERT.

– Навигация с помощью кода и переименование

Инструмент IDE распознает, какие элементы БД применяются в коде. Поэтому если изменить название одного из них в запросе, соответствующие изменения произойдут и в базе. Присутствует функция поиска, где применялся элемент или символ в части запроса. Кроме того, есть опция переноса от использования к точке обновления. Когда пользователь проделывает те же действия в отношении объекта, уже сгенерированного в БД, курсор переносит его в окошко структуры БД. Когда запрашивается конкретное имя элемента, не существующего в базе (к примеру, не корректно указали наименование таблицы или строки), IDE информирует о наличии ошибки и выдает варианты решений.

– Операции с данными

Редактор таблиц в JetBrain DataGrip в состоянии отфильтровывать информацию, а также текстовую навигацию в таблице. Кроме того, если есть связь между внешними "кеями", есть возможность переноситься в табличные строки, ссылающиеся на эти ключи, и обратно.

– Осуществление запросов

Чтобы произвести часть запроса, следует отметить код, а затем запустить его. В окошке результата запроса можно воспользоваться большим количеством опций Table Editor – с его помощью можно корректировать данные, а также в нем присутствует текстовая навигация [11].

2.3. Алгоритм компьютеризированного решения задачи

Разрабатываемая система имеет клиент-серверную архитектуру. Это означает, что каждый модуль системы будет выполнять свою функцию. Система будет функционировать по принципу REST (Representational State Transfer – «передача репрезентативного состояния») API, что позволит ей легко масштабироваться, наращивая всё новые функциональные компоненты [6]. Схема взаимодействия модулей представлена на рисунке 4.

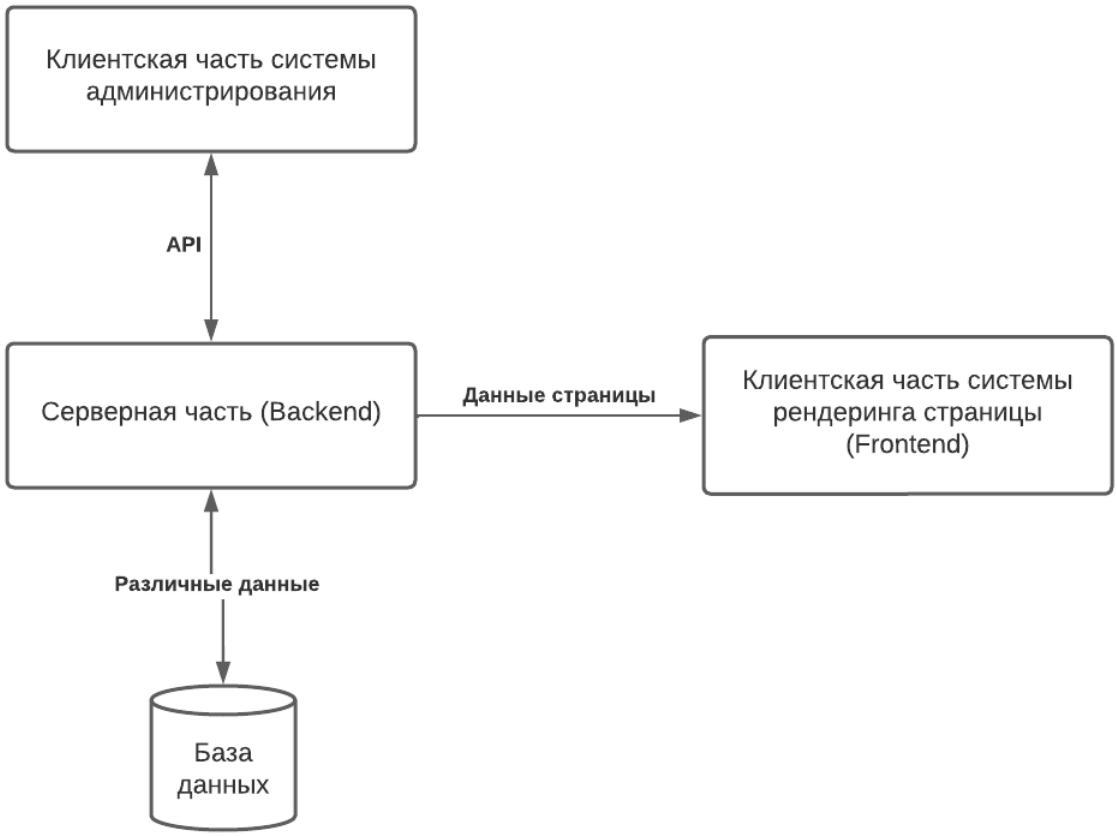

Рисунок 4 – Схема взаимодействия компонентов проектируемой системы

При загрузке страницы происходит проверка авторизации, если авторизация не произошла, то происходит перенаправление на страницу авторизации пользователя. После ввода логина и пароля происходит проверка данных в БД и

в случае наличия пользователя происходит авторизация. Алгоритм работы модуля авторизации представлен на рисунке 5.

После этого появляется главное окно, содержащее актуальное дерево иерархии разделов. Заголовок раздела, ссылку на него и форму редактирования содержания страницы данного раздела (рисунок 6). Схема алгоритма создания нового раздела отображена на рисунке 7.

По истечению сессионного времени необходимо будет снова пройти авторизацию, то есть произойдет перенаправление на страницу авторизацию с необходимостью ввода логина и пароля.

Данный алгоритм является наиболее важным, так как отвечает за распределением функциональных возможностей пользователей.

В ходе реализации планируется использовать алгоритм шифрование base64 для хранения и передачи пароля на сервер.

Рисунок 5 – Блок-схема работы модуля авторизации

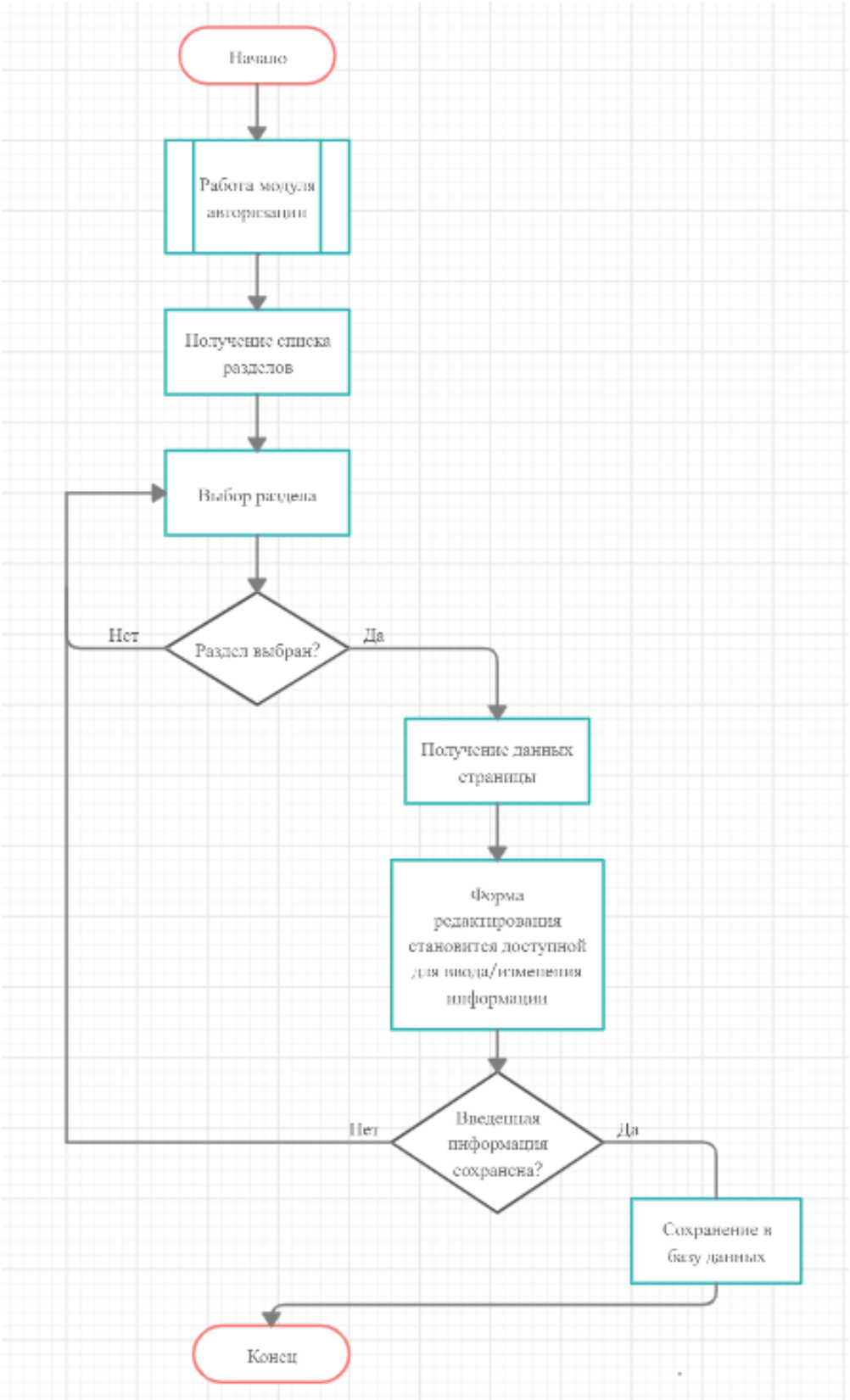

Рисунок 6 – Блок-схема работы с содержанием страницы

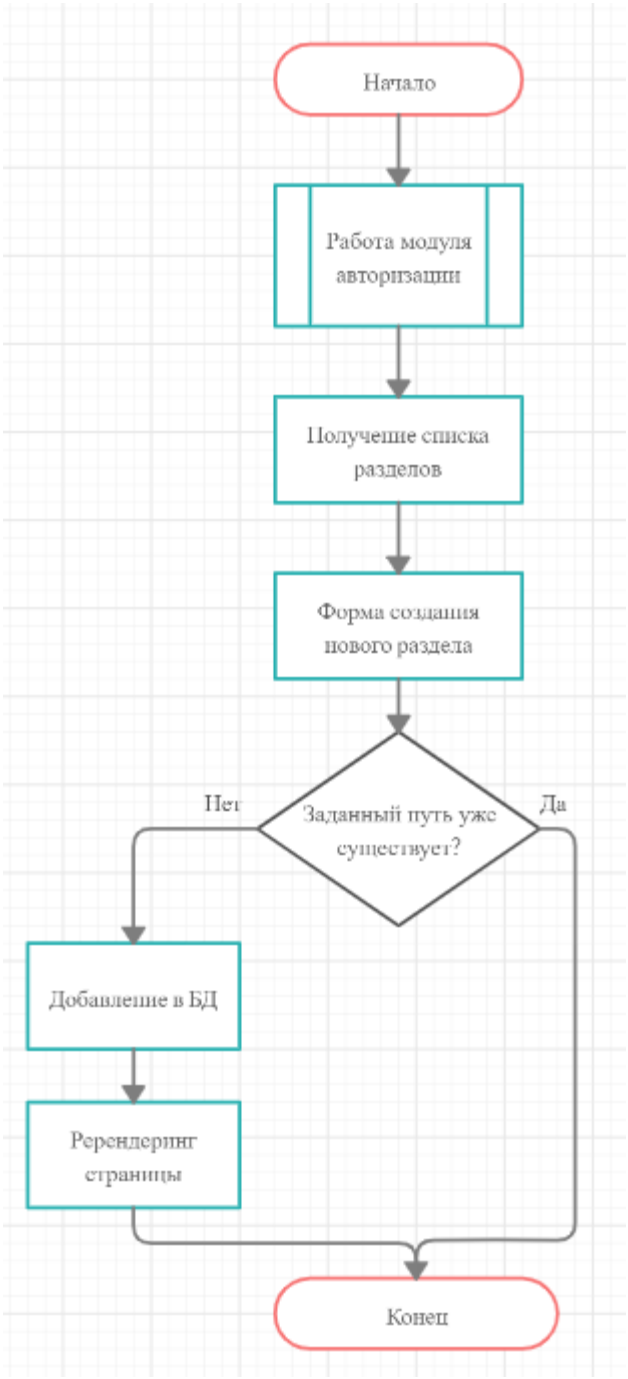

Рисунок 7 – Блок-схема создания нового раздела

2.4. Проектирование базы данных

В ходе анализа предметной области и функциональных потребностей проектируемой системы, были выделены основные сущности базы данных, представленные в таблице 2.

Таблица 2 – Сущности

Название таблицы	Описание таблицы
Files	Содержит информацию с загруженными файлами
Sections	Содержит информацию о существующих разделах
Users	Содержит информацию с пользователями
Users_Sections	Таблица, для связи многие-ко-многим

Сущность «Files» предназначена для организации хранения файлов на сервере. Атрибуты сущности представлены в таблице 3.

Таблица 3 – Атрибуты сущности «Files»

Название атрибута	Описание атрибута	Тип
Id	Ключевое поле	Числовой
Name	Наименование файла	Текстовый

Сущность «Sections» отвечает за хранение секций информации на странице, представляющую собой самостоятельный объект в рамках алгоритма работы системы. Атрибуты сущности представлены в таблице 4.

Таблица 4 – Атрибуты сущности «Sections»

Название атрибута	Описание атрибута	Тип
Id	Ключевое поле	Числовой

Html_code	Хранит информацию о странице	Текстовый
Name	Наименование раздела	Текстовый
Route	Маршрут до раздела	Текстовый
Parent_section	Ключ родительского раздела	Числовой

Сущность «Users» содержит информацию о пользователях, имеющих доступ к редактированию информации на сайтах. Атрибуты сущности представлены в таблице 5.

Таблица 5 – Атрибуты сущности «Users»

Название атрибута	Описание атрибута	Тип
Id	Ключевое поле	Числовой
Name	Имя пользователя	Текстовый
Password	Пароль	Текстовый

Логическая модель базы данных представлена на рисунке 8.

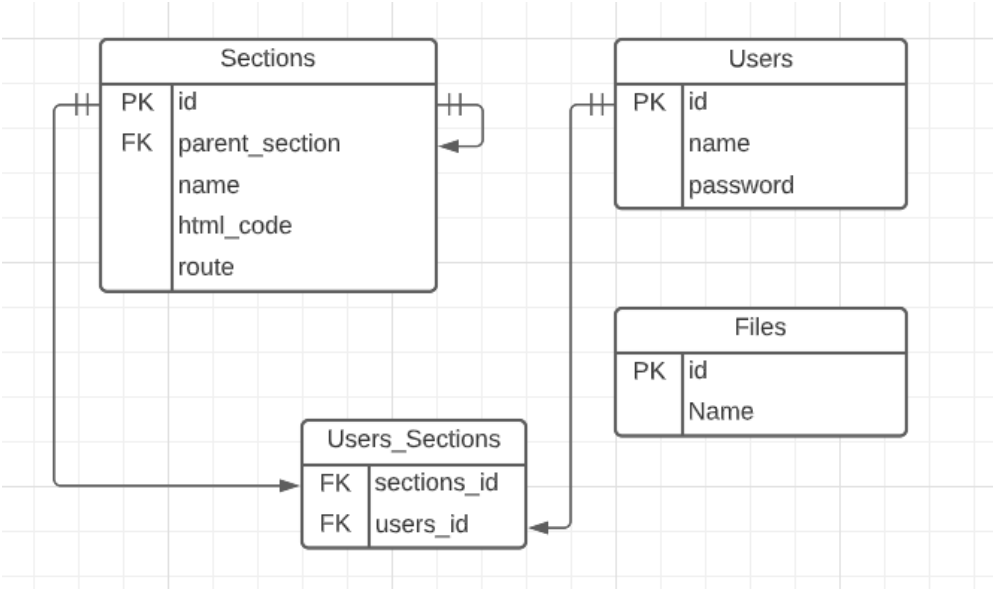

Рисунок 8 – Логическая модель базы данных

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ПРЕДЛАГАЕМОГО АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ

Так как предполагается использование клиент-серверной архитектуры, можно выделить три этапа практической разработки программного продукта:

– разработка базы данных;
– разработка серверной части программного продукта;
– разработка клиентской части программного продукта.

3.1. Разработка базы данных

В ходе этапа проектирования было решено использовать СУБД PostgreSQL, следовательно, необходима реализация с использованием технологий Nest.js, в частности, библиотекой Sequelize.

Конфигурация будет иметь вид, представленный на рисунке 9.

```
SequelizeModule.forRoot( options: {
  dialect: 'postgres',
  host: process.env.POSTGRES_HOST,
  port: Number(process.env.POSTGRES_PORT),
  username: process.env.POSTGRES_USER,
  password: process.env.POSTGRES_PASSWORD,
  database: process.env.POSTGRES_DB,
  models: [User, Section, UserSections],
  autoLoadModels: true
}),
```

Рисунок 9 – Конфигурация Sequelize

Можно видеть, что основные поля конфигурации вынесены во внешний property-файл расширения .env. Его содержимое имеет вид, представленный на рисунке 10.

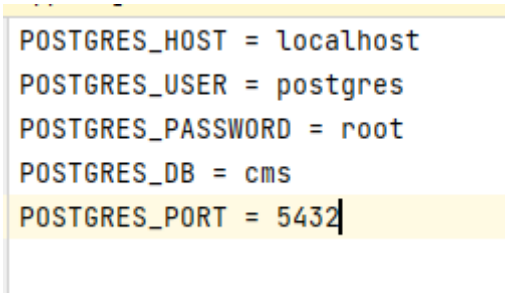

Рисунок 10 – Внешний файл properties

Таблицы базы данных описываются в разделе models, конфигурации представленной на рисунке 1. Так объект Users, автоматически генерируемый из Nest.js в БД PostgreSQL представлена на рисунке 11.

```
@Table( options: {tableName: 'users'})
export class User extends Model<User> {
  @Column( options: {type: DataType.INTEGER, unique: true, autoIncrement: true, primaryKey: true})
  id: number;

  @Column( options: {type: DataType.STRING})
  name: string;

  @Column( options: {type: DataType.STRING, unique: true, allowNull: false})
  login: string;

  @Column( options: {type: DataType.STRING, unique: true, allowNull: false})
  password: string;

  @BelongsToMany( associatedClassGetter: () => Section, through: () => UserSections)
  sections: Section[];
}
```

Рисунок 11 – Объект Users

Таблица Sections имеет более сложную структуру. Поле childrenSection имеет тип данных массива, предназначенную для хранения массива id “дочерних” элементов. Так же присутствует декоратор @BelongsToMany, реализующий связь многие-ко-многим с таблицей users. Скриншот представлен на рисунке 12.

Рисунок 12 – Объект Sections

Таблица для связи многие-ко-многим так же вынесена в отдельную модель и имеет вид, представленную на рисунке 13.

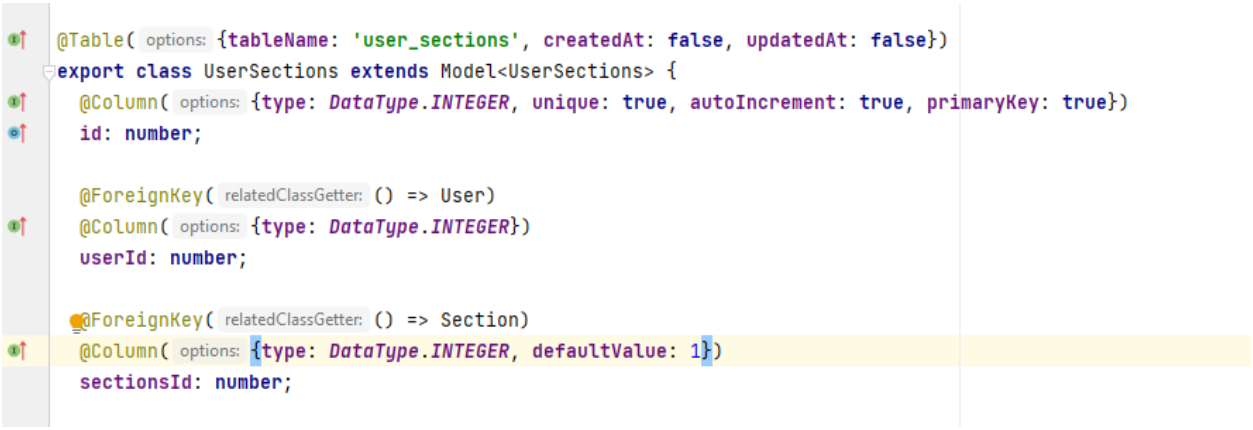

Рисунок 13 – Связи таблиц

3.2. Разработка серверной части программного продукта

При проектировании серверной части программного продукта, на этапе выбора технологического решения, был выбран фреймворк Nest.js, использующий язык программирования TypeScript в качестве основного языка.

Точкой входа является файл main.ts, где запускаются все зависимые модули из файла app.module.ts.

App.module.ts содержит все зарегистрированные вложенные модули, отвечающие за логику работы каждого компонента, таких как section.module.ts и user.module.ts. В section.module.ts вызываются все необходимые для этого компонента скрипты, такие как SectionsController и SectionsService, а так же экспортируемые и импортируемые скрипты.

Листинг SectionsService представлен на рисунке 14.

```
@Injectable()
export class SectionsService {

  constructor(@InjectModel(Section) private sectionRepository: typeof Section,
              private userService: UsersService) {}

  async createSection(dto: CreateSectionDto){
    const section = await this.sectionRepository.create(dto);
    return section
  }

  async createChildSection(dto: CreateSectionDto, idParent){
    const section = await this.sectionRepository.findByPk(idParent)
    if(section){
     const newSection = await this.sectionRepository.create(dto);
     newSection.parentSection = idParent;
     await newSection.save();
     console.log(newSection.id);
     section.childrenSection.push(123);
     console.log(section.childrenSection);
     await section.changed();
     await section.save();
     return newSection;
```

Рисунок 14 – Листинг SectionsService

На рисунке 14 можно видеть описание логики работы таких функций, как:

– создание секции;
– создание дочернего элемента;
– получение списка секций;
– получение секции по id;
– присвоение прав доступа к секции для пользователя;
– получение секций в виде дерева.

SectionsController служит для привязки необходимых функций к url адресу сервера, что подразумевается в структурном подходе парадигмы REST API. Скрипт представлен на рисунке 15.

```
@Controller( prefix: 'sections')
export class SectionsController {

  constructor(private sectionsService: SectionsService) {}

  @Post()
  create(@Body() sectionDto: CreateSectionDto){
    return this.sectionsService.createSection(sectionDto)
  }

  @Post( path: ':id')
  createChildrenSection(@Body() sectionDto: CreateSectionDto, @Param( property: 'id') idParent: number){
    return this.sectionsService.createChildSection(sectionDto,idParent);
  }

  @Get()
  getAllSections(){
    return this.sectionsService.getAllSections();
  }

  @Get( path: ':id')
  getSectionById(@Param( property: 'id') id: number){
    return this.sectionsService.getSectionById(id);
  }
}
```

Рисунок 15 – Листинг SectionsController

На рисунке 8 видно, что входной точкой в логику является раздел /sections/ из url адреса сервера. Запросы деляется на @Get и @Post. Get возвращают данные по запросу без входных данных, либо с параметром описываемом в самом url

запросе, как например “/section/1” для getSectionById, параметром которого является id равное 1.

Post запросы принимают данные в формате JSON на обработку и дальнейшее сохранение изменений в базе данных.

В случае с UserService и UserController ситуация аналогичная, их листинг представлен на рисунке 16 и 17.

```
@Injectable()
export class UsersService {

  constructor(@InjectModel(User) private userRepository: typeof User) {}

  async createUser(dto: CreateUserDto){
    const user = await this.userRepository.create(dto);
    return user
  }

  async getAllUsers(){
    const users = await this.userRepository.findAll();
    return users
  }

  async getUserById(id){
    const user = await this.userRepository.findOne( options: {where: id});
    return user
  }
}
```

Рисунок 16 – Листинг UserService

Рисунок 17 – Листинг UserController

3.3. Разработка клиентской части программного продукта

Клиентская часть написана с использованием фреймворка Vue.js и библиотеки Vuetify.js.

Точкой входа в приложение является компонент main.js, где создается экземпляр Vue компонента, а также глобально регистрируются все необходимые модули, листинг представлен на рисунке 18.

lit-cms-frontend-admin / src / main.js

```
1  import Vue from 'vue'
2  import './plugins/axios'
3  import App from './App.vue'
4  import vuetify from './plugins/vuetify';
5  import ACMS from "./components/ACMS/index";
6  import Store from '@/configurations/store'
7  import router from './router'
8
9
10 Vue.config.productionTip = false;
11 Vue.component('a-c-m-s', ACMS);
12
13 new Vue({
14   vuetify,
15   data: Store,
16   router,
17   render: h => h(App)
18 }).$mount('#app')
19
```

Рисунок 18 – Листинг main.js

Дальнейшая работа предполагается в программном разделе v-app, находящимся в файле app.vue, где вызывается vue-router, отвечающий за отрисовку необходимых страниц по запрашиваемому url адресу. Код представлен на рисунке 19.

lit-cms-frontend-admin / src / App.vue

```
<template lang="pug">
  v-app
    v-content
        router-view

</template>

<script>


    import AHeader from "./components/AHeader/index";
    import AFooter from "./components/AFooter/index";
    import AText from "./components/AText/index";

    import APanel from "./components/APanel/index";
    import ASection from "./components/ASections/index";


    export default {
    name: 'App',


        props: {

    },
    components: {
```

Рисунок 19 – Листинг v-app

При написании кода для разметки страниц использовался шаблонизатор PUG, позволяющий уменьшить предполагаемый код в два раза за счёт использования табуляции в качестве метки конца тега, вместо закрывающих тегов в исходном виде html.

Компонент router-view считывает запрос из url адреса и отрисовывает необходимые страницы на основе описанной конфигурации в файле router.js, представленном на рисунке 20.

lit-cms-frontend-admin / src / router / index.js

```
import Auth from '../components/Auth'
import AMain from '../components/AMain'

Vue.use(VueRouter);

const routes = [
  {
    path: '/',
    name: 'auth',
    component: Auth,
  },
  /* {
     path: '/sections',
     name: 'section',
     component: Sections
   }, */
  {
    path: '/section',
    name: 'section',
    component: AMain
  },
  {
    path: '*',
    name: 'section',
    component: AMain
  },

```

Рисунок 20 – Листинг router.js

Компонент главной страницы AMain содержит несколько компонентов, файлово декомпозированных по разным частям файловой структуры проекта. Главное окно страницы, представленное на рисунке 21. Для удобства на рисунке 21 нанесены три метки, каждая из которых отвечает за отдельные компоненты.

Под цифрой 1 представлен компонент редактора текста со всеми функциональными кнопками для манипуляций и форматирования текстовой, графический и табличной информации. Метка 2 указывает на компонент отрисовки всех доступных для редактирования разделов. Напротив метки 3 располагается

наименование выбранного раздела и функциональные кнопки быстрого перехода в режим отображения на странице отрисовки и генерация фреймовой вставки.

Часть листинга окна представлено на рисунке 22.

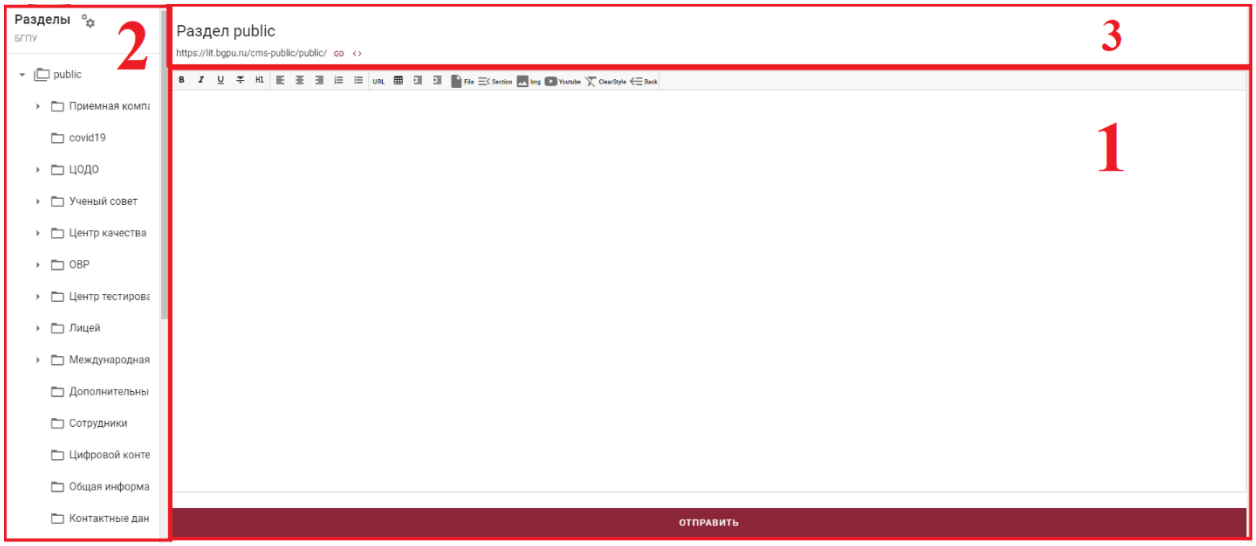

Рисунок 21 – Главная страница

lit-cms-frontend-admin / src / components / AMain / **index.vue** Edit

```
<template lang="pug">
    div
        v-navigation-drawer(permanent='' clipped='' app='')
            a-menu(@Pushing = "Push" @Linking = "Linked" :sections = 'sections' ref='aMenu' @update = "getSections")
        v-app-bar(app='', :color='this.$root.COLOR', dark='' clipped-left='')
            img(src='@/assets/logo2.png' onclick="location.href = '/auth'" align='left', tag='span', style='cursor: pointer')
            v-spacer
            v-toolbar-title Система контроля содержимого
        br
        v-list-item-title.headline.ma-2 Раздел {{section}}
            v-text-field(id="frameGen" readonly v-if="showTempString" value='<iframe style="margin: 0; padding: 0; height: 100em; width: 100%;" scrolling="auto" id="fr
            v-list-item-subtitle {{link}}
                v-btn(@click="openHref" icon='' x-small='' v-if="link").ma-2
                    v-icon(:color='this.$root.COLOR' ) mdi-link
                v-btn(@click="iframeGen()" icon='' x-small='' v-if="link && user==='admin'")
                    v-icon(:color='this.$root.COLOR' small='') mdi-code-tags
            //v-list-item-subtitle(@click="openHref") {{link}}
                 //v-btn(@click="test") {{link}}
        a-c-m-s(v-model='CurHtml.html' ref='aCms')
        br
        v-btn(tile='' v-if="link" @click="Submit" :color='this.$root.COLOR' block='' large='' :disabled="dialog" :loading="dialog").textColor Отправить
        .text-center
            v-dialog(v-model='dialog', hide-overlay='', persistent='', width='300' )
                v-card(dark='' :color='this.$root.COLOR')
                    v-card-text
                        | Пожалуйста, подождите
                        v-progress-linear.mb-0(indeterminate='', color='white')

            v-snackbar(v-model='snackbar')
                | {{ text }}
        //v-list-item(three-line='')
```

Рисунок 22 – Листинг главной страницы

Компонент вывода разделов описан в теге “a-menu”, который принимает на вход props’ы и ссылки на методы, вызываемые через ref, обрабатываемый в разделе описания функций methods. Вызов объекта происходит на этапе получения данных для отрисовки, с помощью callback, в случае успеха. Листинг представлен на рисунке 23.

```
getSections() {
    Api.getSections().then(value => {
            this.sections = value.treeSection;
            this.$refs.aMenu.tester();

            console.log('Запросил');
        },
        () => {
            console.log('Ошибка');
        });
},
```

Рисунок 23 – Метод получения данных

Меню, вызываемое из главного окна страницы, принимает на вход данные, реактивно обрабатываемые в коде разметки страницы через встроенный во фреймворк Vue.js оператор v-for, который предоставляет работу с каждым объектом из массива принимаемых объектов. Код представлен на рисунке 24.

```
v-list(dense='', nav='')
    //v-list-item(v-for='section in sections', :key='section.id', @click="getChildren(section.route)" /*:to="{ name: 'section', params: {id: section}}"*/)
            v-list-item-content
                v-list-group
                    template(v-slot:activator='')
                        v-list-item-title {{ section.title }}

    //v-list-group(v-for='section in sections', :key='section.id', @click="getChildren(section.route)" v-if="true")
        template( v-slot:activator='')
            v-list-item-title {{section.title}}
        v-list-group(no-action='', sub-group='' v-for='child in childrens', :key='child.id')
            template(v-slot:activator='')
                v-list-item-content
                    v-list-item-title {{child.title}}
            v-list-item(v-for='section in sections', :key='section.id' @click="")
                v-list-item-content
                    v-list-item-title {{ section.title }}

    //div(v-else="false")
        v-list-item(@click="") {{section.title}}
```

Рисунок 24 – Листинг отрисовки разделов

В ходе работы, по клику на нужный раздел, происходит передача данных родительскому элементу, “AMain”, через оператор $emit, после чего родительский элемент передает эти данные дочернему “ACMS” с помощью props. Листинг представлен на рисунке 25.

```
a-c-m-s(v-model='CurHtml.html' ref='aCms')
br
v-btn(tile='' v-if="link" @click="Submit" :color='this.$root.COLOR' block='' large='' :disabled="dialog" :loading="dialog").textColor Отправить
.text-center
    v-dialog(v-model='dialog', hide-overlay='', persistent='', width='300' )
        v-card(dark='' :color='this.$root.COLOR')
            v-card-text
                | Пожалуйста, подождите
                v-progress-linear.mb-0(indeterminate='', color='white')

    v-snackbar(v-model='snackbar')
        | {{ text }}
```

Рисунок 25 – Листинг передачи данных редактору текста

Компонент “ACMS” служит для управления текстовой, графической и табличной информацией. В качестве входных данных использует такие props, как:

– value – обрабатываемый текст;

– hide – функция скрытия редактора;

– height – управление высотой редактора;

Код представлен на рисунке 26.

```
props: {
    value: {
        type: String
    },
    hide: {
        type: Object,
        default: () => ({})
    },
    height: {
        type: [Number, String],
        default: 600
    }
},
```

Рисунок 26 – Листинг внешних объектов управления

Переменная value реактивно обрабатывается в формате HTML с помощью “слушателя” страницы каждые 100 миллисекунд. Это позволяет моментально форматировать вводную информацию и отображать её по принципу “ WYSIWYG – what you see is what you get”. Слушатель представлен на рисунке 27.

```
watch: {
    value: {
        handler (val) {
            if (val) {
                setTimeout(() => {
                    if (!document.getElementById('editor').innerHTML) {
                        document.getElementById('editor').innerHTML = val;
                    }
                }, 100);
            }
        },
        immediate: true
    }
},
```

Рисунок 27 – Листинг функции изменения

Кнопки, расположенные в верхней части редактора, вызывают соответствующие функции из раздела methods, некоторые из которых вызывают новые формы, где необходимо ввести уточняющую информацию, в зависимости от вызываемой функции.

Например, форма вставки изображения, представленная на рисунке 28, запрашивает такие поля ввода, как:

– текст – тот текст, который будет выведен в качестве параметра alt тега img, в случае не отрисовки изображения;

– файл – сам файл изображения;

– ориентация – метод расположения изображения в тексте;

– размер – управляется ползунком.

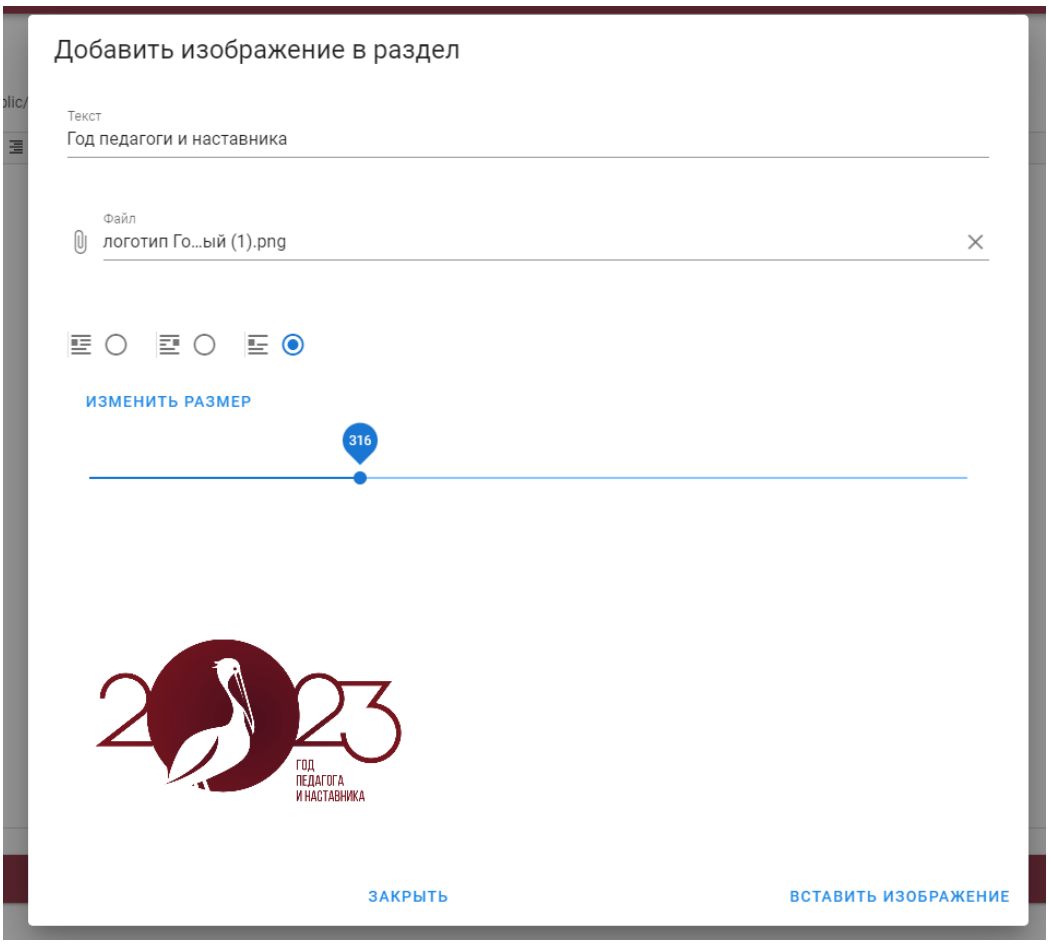

Рисунок 28 – Форма вставки изображения

Листинг формы вставки изображения представлен на рисунке 29. Код отрисовки предварительного просмотра представлен на рисунке 30. Функция замены метки под изображения с последующей вставки в страницу представлен на рисунке 31.

```
<template lang="pug">
    v-dialog(v-model="mutableDialog" :max-width="600+widthImg")
        v-card.noscroll
            v-card-title
                span.headline Добавить картинку в раздел {{asection}}
            v-card-text
                v-container
                    v-row
                        v-col(cols='12')
                            v-text-field(label='Текст' v-model="text")
                        v-col(cols='12')
                            v-file-input(label="Файл" v-model="files")
                        v-col(cols='12' v-if="files")
                            v-radio-group(row='' v-model="formatImgText")
                                v-divider(vertical='')
                                v-icon(left='') mdi-format-float-left
                                v-radio(value='left')
                                v-divider(vertical='')
                                v-icon(left='') mdi-format-float-right
                                v-radio(value='right')
                                v-divider(vertical='')
                                v-icon(left='') mdi-format-float-none
                                v-radio(value='none')
                            v-btn(color='blue darken-1', text='', @click="LoadImg") Изменить размер
                        v-col(cols='12' v-if="resize")
                            v-container(fluid='')
                                v-slider(v-model='widthImg', min='10', max='1000', step='1'  thumb-label="always")
                                v-navigation-drawer(:width='widthImg', :value='true', stateless='')
                        v-col(cols='12' v-if="resize")
                            v-img(:src="srcImg" :width='widthImg')
```

Рисунок 29 – Листинг формы вставки изображения

```
methods: {
    ImgLink() {
        Api.upFile(this.files).then(result => {
                this.$emit('Insert', `<img style="float: ${this.formatImgText}; margin: 0 13px 5px 0; "
                src=${Api.api}/file/get?id=${result} width=${this.widthImg} alt=${this.text}></img>`);
                this.text = '';
                this.files = null;
                this.srcImg = '';
                this.formatImgText = 'none';
                this.width = 160;
            },
            (error => {
                console.log(error)
            })
        );

        this.resize = false;
        this.mutableDialog = !this.mutableDialog;
    },
```

Рисунок 30 – Листинг кода предварительного просмотра изображения

```
,,
showNewImg() {
    new Promise((resolve) => {
        this.selectedNode = document.getSelection().focusNode;
        if (this.selectedNode.data === undefined) {
            this.exec('insertHTML', true, '<div> </div>');
            this.selectedNode = document.getSelection().focusNode;
        }
        resolve();
    }).then(() => {
        let offset = document.getSelection().focusOffset;
        this.selectedNode.data = this.selectedNode.data.substr(0, offset) + "+" + this.selectedNode.data.substr(offset, this.selectedNode.data.length);
        this.$refs.aNewImg.showDialog();
    })
},
```

Рисунок 31 – Листинг вставки изображения

Форма вставки файла, в общем случае, имеет схожий функционал, имеющий меньше параметров на входе. После появления окна, пользователю необходимо выбрать файл на компьютере и задать текст для вставляемой ссылки на файл. После загрузки файла на сервер, клиенту возвращается id файла, после чего формируется сама ссылка на этот файл. Форма и код формы представлены на рисунке 32 и 33.

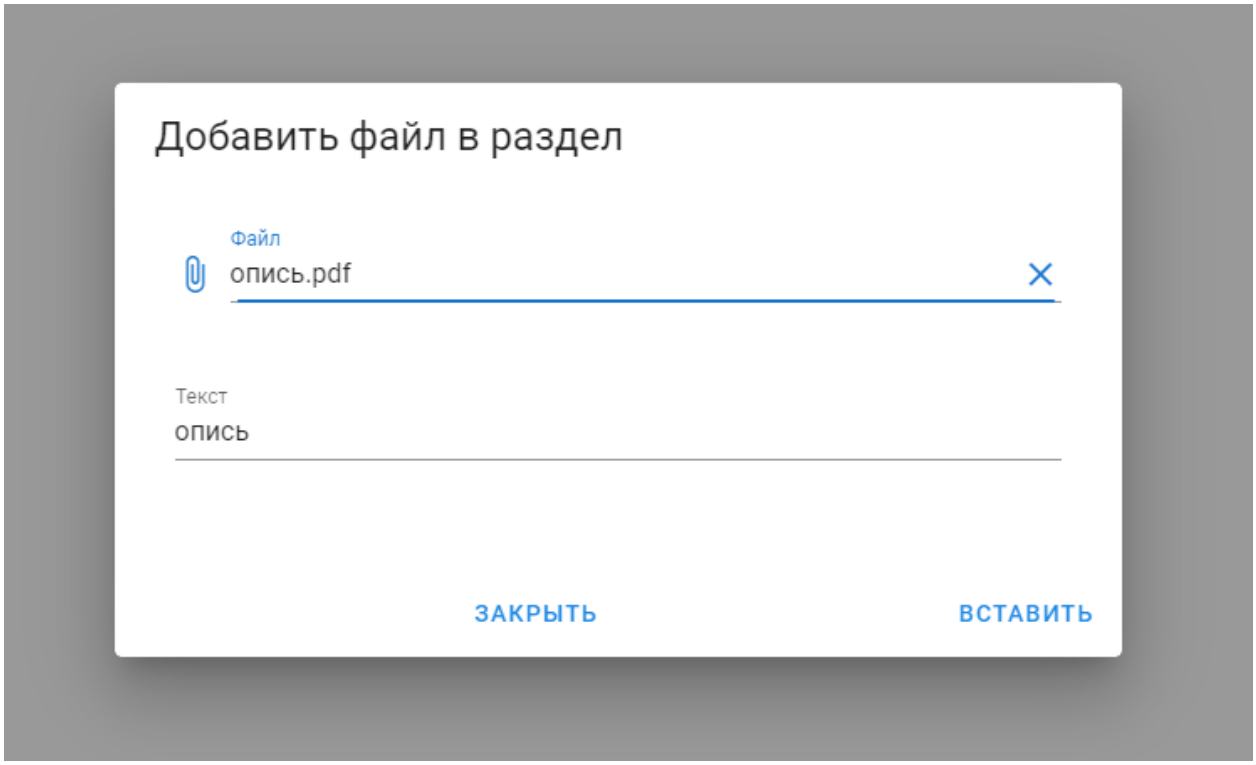

Рисунок 32 – Форма загрузки файла

lit-cms-frontend-admin / src / components / ANewFile / **index.vue**

```
<template lang="pug">
    v-dialog(v-model="mutableDialog" max-width="600px")
        v-card.noscroll
            v-card-title
                span.headline Добавить файл в раздел {{asection}}
            v-card-text
                v-container
                    v-row
                        v-col(cols='12')
                          //v-file-input(label="Файл" v-model="files" @change="text = files.name")
                          v-file-input(label="Файл" v-model="files" @change="text = files.name.split('.')[0]")
                        v-col(cols='12')
                          v-text-field(label='Текст' v-model="text")



          v-card-actions
                v-spacer
                v-btn(color='blue darken-1', text='', @click="showDialog") Закрыть
                v-spacer
                v-btn(color='blue darken-1', text='', @click="FileLink") Вставить
```

Рисунок 33 – Листинг формы загрузки файла

Кнопка вызова формы вставки ссылки на секцию имеет в себе библиотечный компонент v-treeview, который принимает на вход древовидную структуру объектов, имеющих дочерние объекты. Компонент автоматически обрабатывает всю вложенную структуру и отрисовывает её на экране. Внешний вид и код представлены на рисунках 34 и 35.

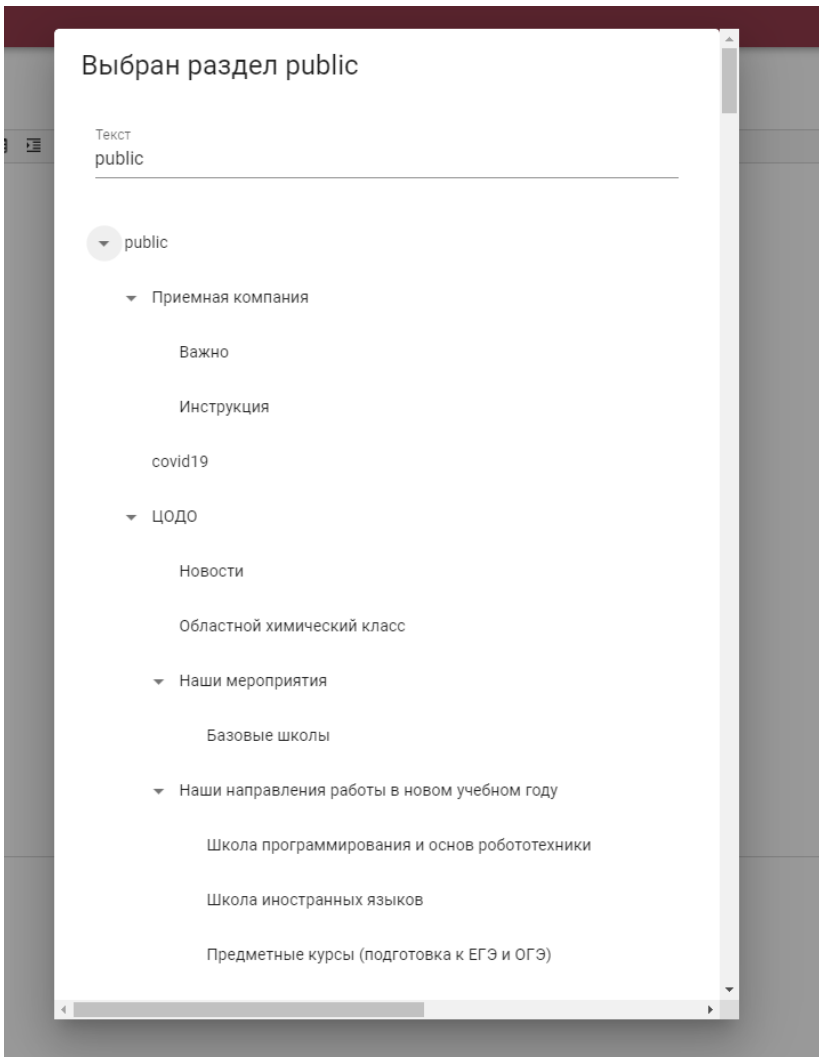

Рисунок 34 – Форма вставки ссылки на раздел

lit-cms-frontend-admin / src / components / ALinkOnSection / **index.vue**

```
<template lang="pug">
    v-dialog(v-model="mutableDialog" max-width="600px")
        v-card.noscroll
            v-card-title
                span.headline Выбран раздел {{CurNameSection}}
            v-card-text
                v-container
                    v-row
                        v-col(cols='12')
                            v-text-field(label='Текст' v-model="text")

                        v-treeview(:items='sections', item-children='treeSection', item-key='id', activatable='', open-on-click='' open-all='')
                            template(slot='label', slot-scope='{ item }')
                                div(@click='onClick(item)') {{ item.name }}



            v-card-actions
                v-spacer
                v-btn(color='blue darken-1', text='', @click="showDialog") Закрыть
                v-spacer
                v-btn(color='blue darken-1', text='', @click="getFullPath()") Вставить


</template>
```

Рисунок 35 – Листинг формы вставки ссылки на раздел

Форма вставки видео с youtube принимает на вход ссылку любого формата (длинную и короткую), после чего размещает фрейм с видеозаписью на странице. Форма вставки видео и листинг представлены на рисунках 36 и 37.

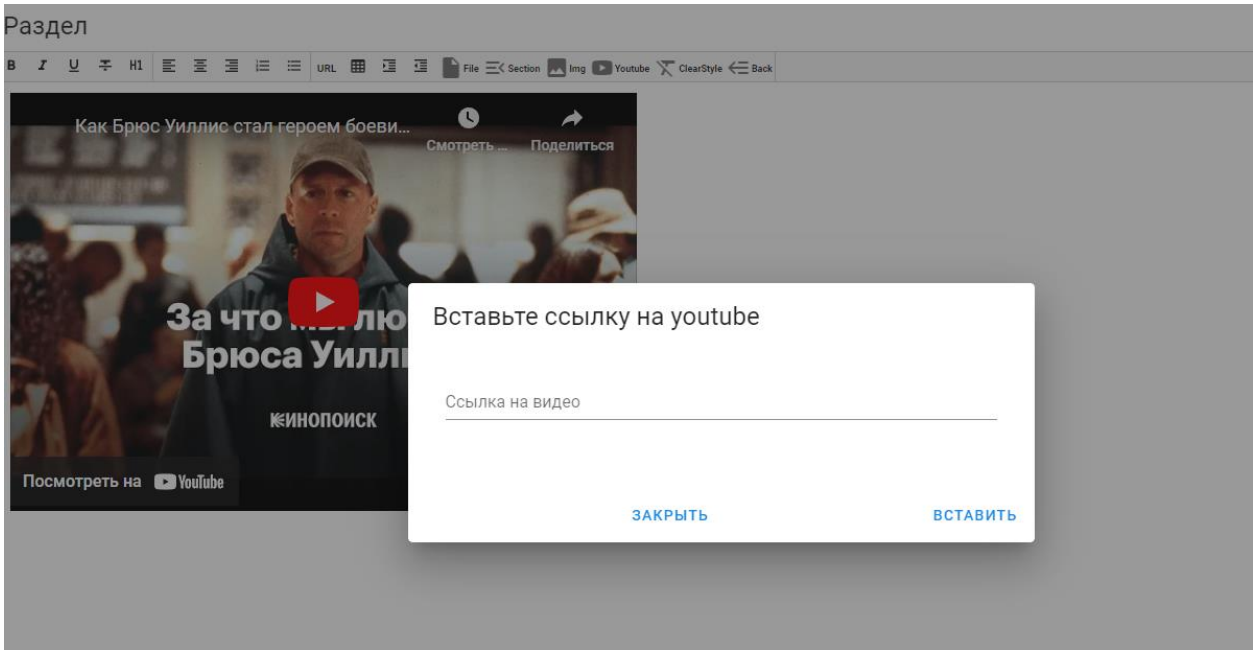

Рисунок 36 – Форма вставки видео по ссылке

```
methods: {
    submit() {
        this.parse = this.url.split('/');
        if (this.parse[2] == "youtu.be") {
            this.code = this.parse[this.parse.length - 1];
            this.$emit('Insert', `<iframe width="600" height="400" src=https://www.youtube.com/embed/${this.code}
            frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>`);
        } else if (this.parse[2] == "www.youtube.com") {
            this.code = this.url.split('=');
            this.code = this.code[this.code.length - 1];
            this.$emit('Insert', `<iframe width="600" height="400" src=https://www.youtube.com/embed/${this.code}
            frameborder="0" allow="accelerometer; autoplay; encrypted-media; gyroscope; picture-in-picture" allowfullscreen></iframe>`);
        } else alert("Видео не найдено. Скопируйте корректный url");
        this.url = null;
        this.code = null;
        this.mutableDialog = !this.mutableDialog;
    },

    showDialog() {
        this.mutableDialog = !this.mutableDialog;
    },
},
```

Рисунок 37 – Листинг формы вставки видео по ссылке

Все описанные выше методы используют API сервера, вызываемые из конфигурационного файла service.js, где описаны url обращения к серверу. Для взаимодействия с ним описаны специальные заголовки, необходимые для корректной работы запроса, а также перехватчик http-кода ошибок авторизации, осуществляющая переход на форму авторизации в случае недостаточности прав доступа. Код представлен на рисунке 38.

lit-cms-frontend-admin / src / service / apiService.js

```
11
12  const baseURL = `${api}`;
13
14  const API = axios.create({
15      withCredentials: true,
16      baseURL,
17
18      headers: {
19          "X-Requested-With": "XMLHttpRequest",
20          "Access-Control-Allow-Origin": "*",
21          "Accept": "application/json",
22          "Access-Control-Allow-Methods": "GET,POST,PUT,DELETE,OPTIONS",
23          "Access-Control-Allow-Headers": "Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With",
24          //'Content-Type': 'multipart/form-data'
25          "Set-Cookie": "SameSite=None;"
26          //"Set-Cookie": "cross-site-cookie=name; SameSite=None; Secure"
27      }
28  });
29
30  API.interceptors.response.use(({data}) => data
31      , (error) => {
32          if (error.response && error.response.status === 403 || (error.response && error.response.status === 401)) {
33              router.replace({
34                  name: 'auth',
35                  query: {redirect: router.currentRoute.fullPath},
36              })
37          }
38      });
39
```

Рисунок 38 – Листинг настройки взаимодействия с сервером

3.4. Руководство пользователя

При входе в систему через web-обозреватель, появляется окно авторизации, где необходимо ввести логин и пароль. Скриншот формы представлен на рисунке 39.

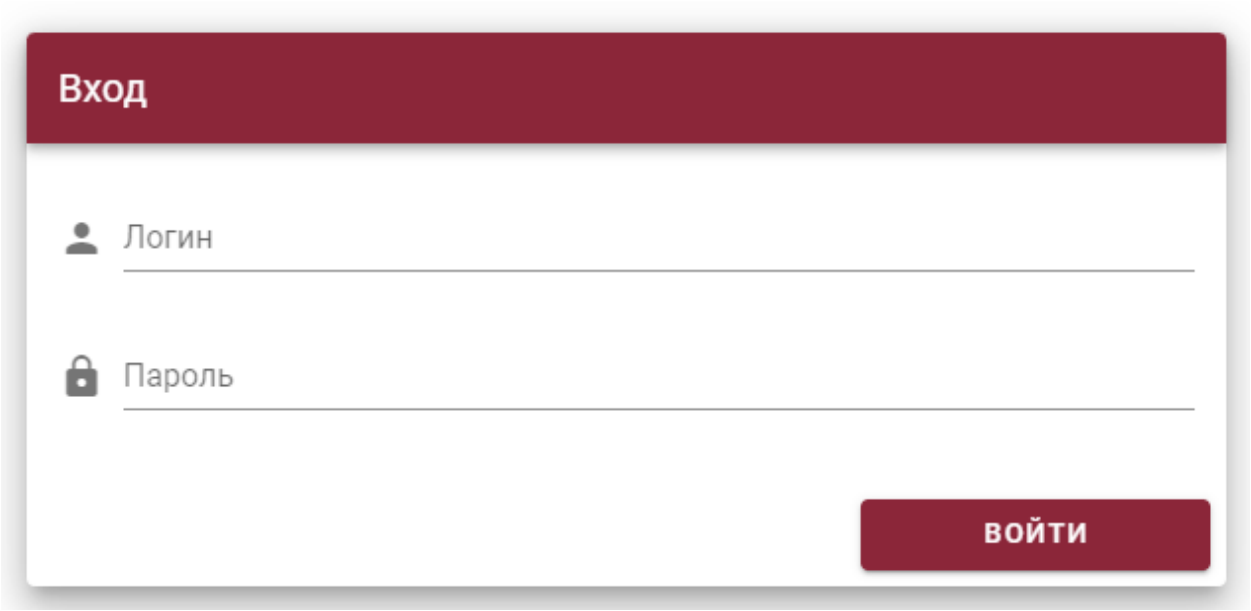

Рисунок 39 – Окно авторизации

По нажатию кнопки “войти” появляется главное окно, содержащее дерево разделов, название выбранного раздела, окно редактирования содержимым. Главная страница представлена на рисунке 40.

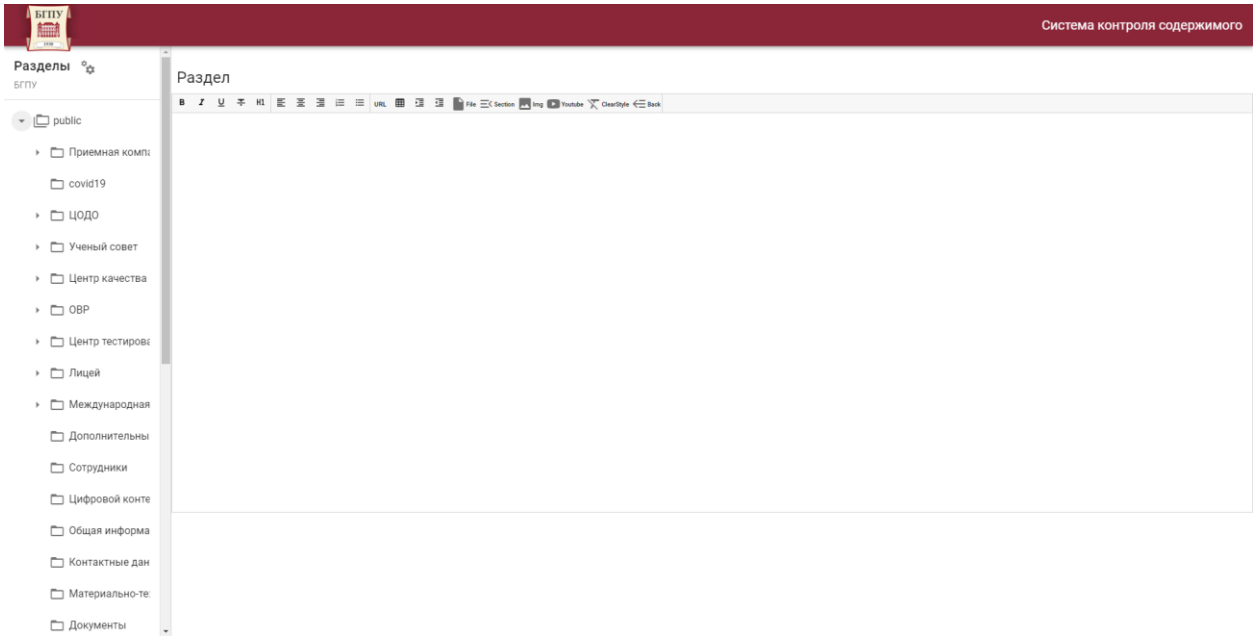

Рисунок 40 – Главная страница

По нажатию на раздел происходит загрузка и отображение содержимого страницы с возможностью редактирования в поле текстового редактора. Скриншот примера страницы представлен на рисунке 41.

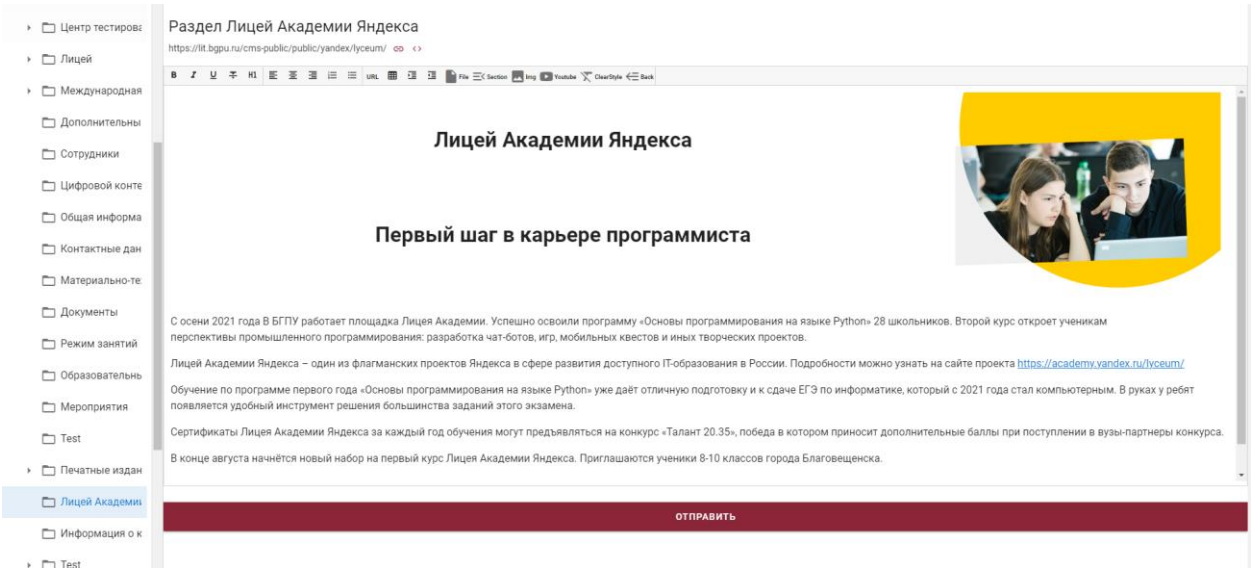

Рисунок 41 – Раздел «Лицей Академии Яндекса»

При нажатии на кнопку добавления раздела, появляется форма с обязательными полями, необходимыми для создания записи в базе. Форма представлена на рисунке 42.

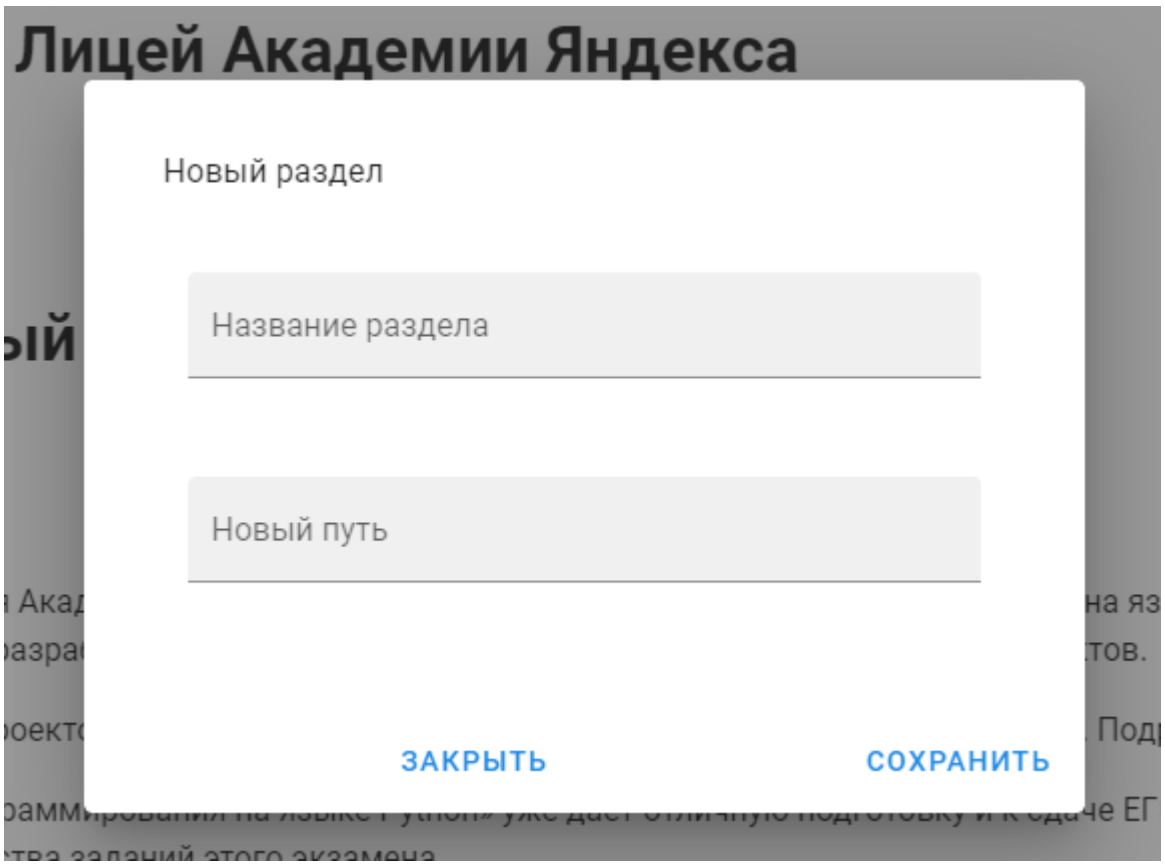

Рисунок 42 – Форма создания нового раздела

После создания нового раздела, происходит переход к его редактированию с целью наполнения текстовой, графической или табличной информацией. Управление видами информации осуществляется с помощью кнопочной панели инструментов, расположенной над текстовым редактором. Скриншот панели инструментов представлен на рисунке 43.

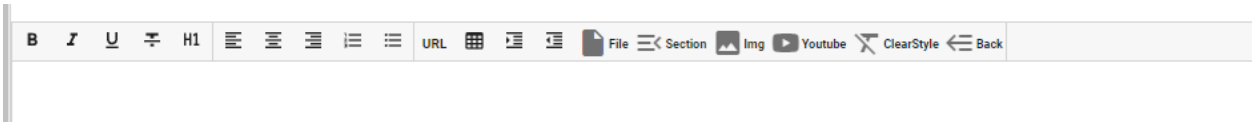

Рисунок 43 – Панель инструментов

Помимо “классических” способов форматирования текста жирным начертанием, курсивом, подчеркиванием, расположенных в первых рядах панели, име-

ется возможность управлением положения текста на странице. Следом располагается функция вставки таблицы, с выбранным количеством ячеек. Форма представлена на рисунке 44.

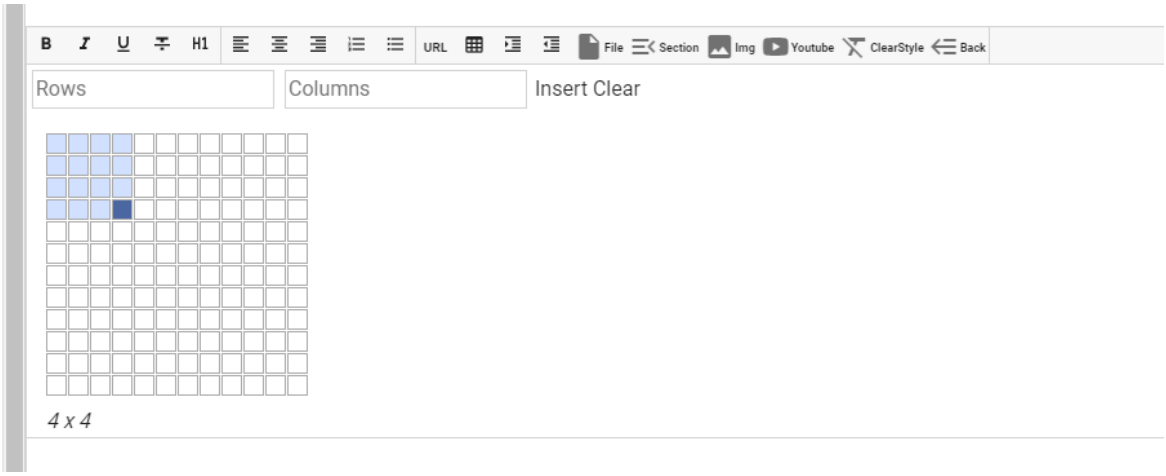

Рисунок 44 – Форма вставки таблицы

При нажатии на кнопку загрузки файла, появляется окно с полями вызова проводника для выбора файлов и текстового поля для наименования строки встраивания ссылки. Скриншот представлен на рисунке 45.

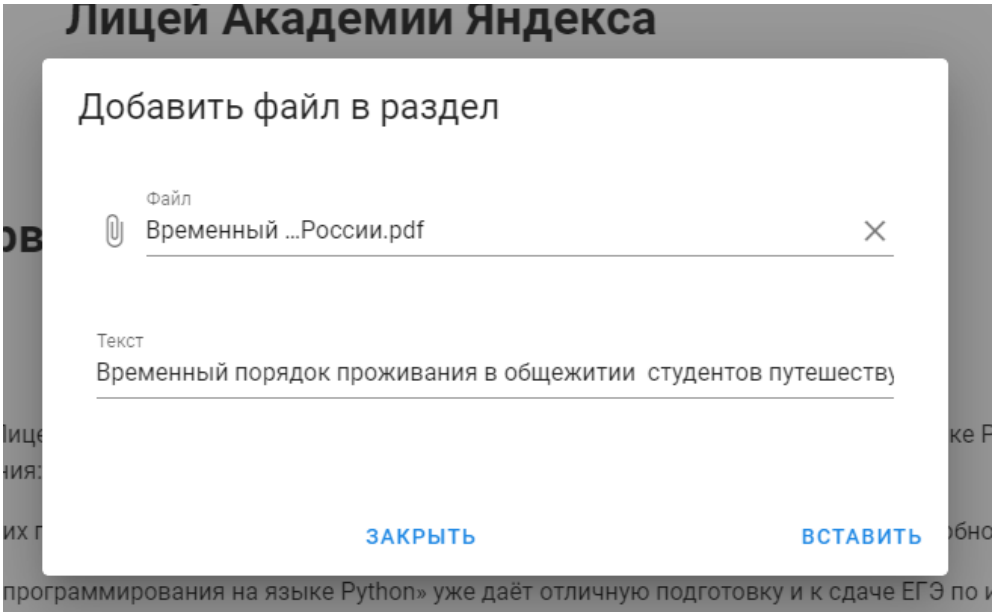

Рисунок 45 – Форма загрузки файла

При клике на кнопку встраивания ссылки на раздел, появляется форма с деревом доступных для редактирования разделов. Форма представлена на рисунке 46.

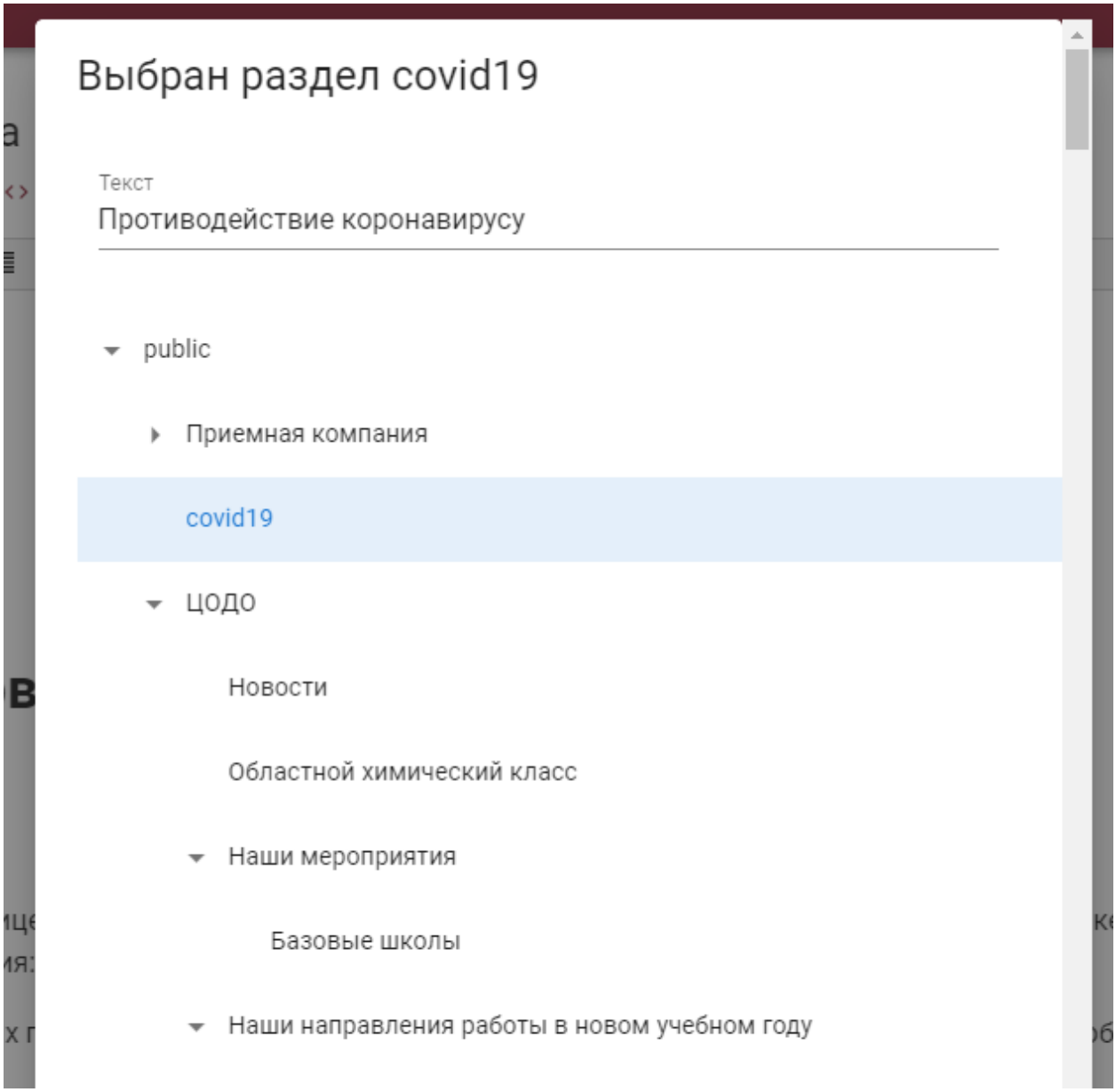

Рисунок 46 – Форма вставки ссылки на раздел

При нажатии на функцию встраивания изображения вызывается форма выбора графической информации через проводник и поле ввода текстовой информации для параметра “alt”, тега “img”, отвечающая за альтернативный вывод текстовой информации в случае ошибки загрузки изображения. Скриншот представлен на рисунке 47.

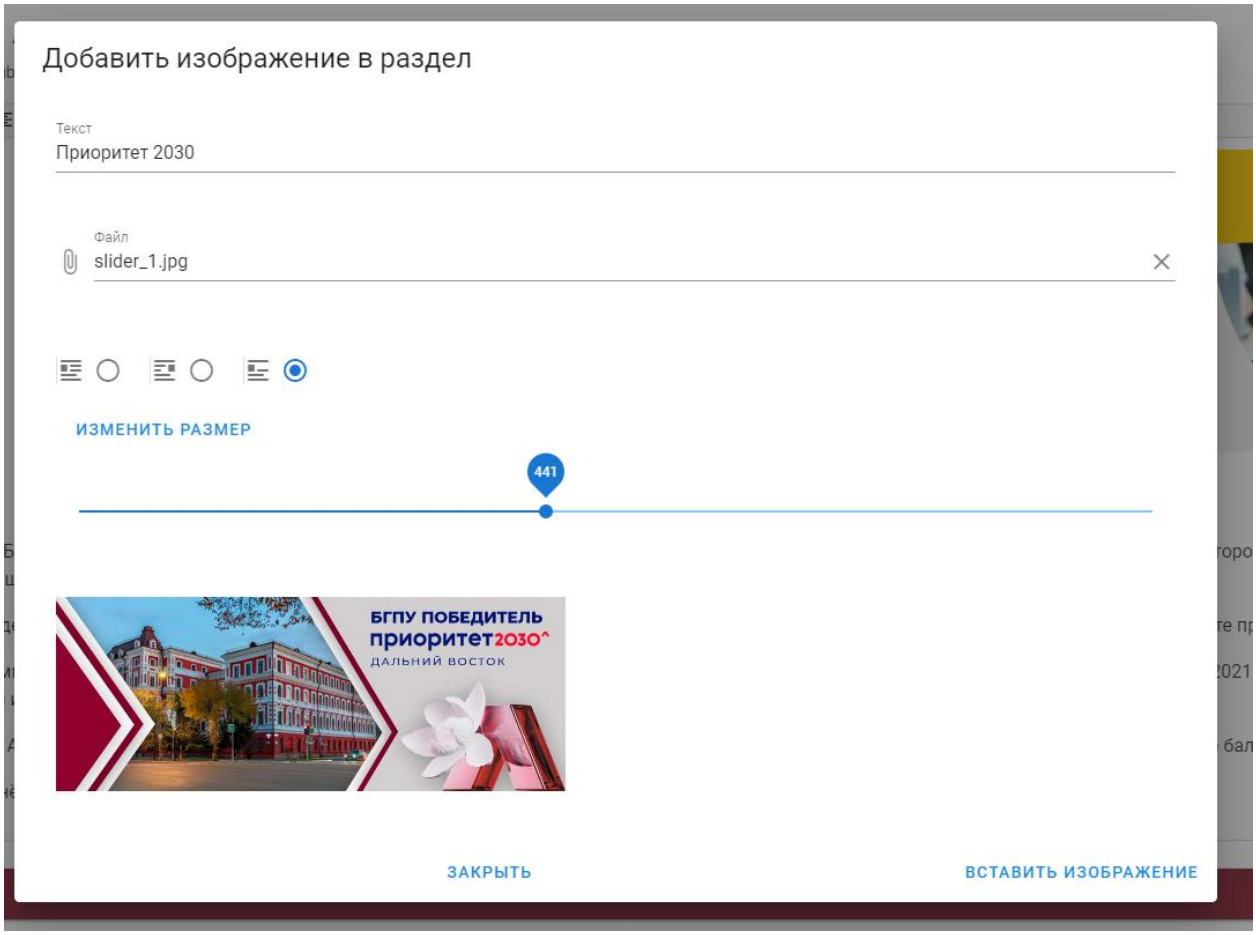

Рисунок 47 – Форма загрузки изображения

Две последние функциональные кнопки из панели инструментов имеют функции очистки стиля выбранного блока информации и кнопка встраивания ссылок на предыдущий переход из web-обозревателя, созданный для удобства переходов при вызове из фрейма.

3.5. Руководство администратора

3.5.1. Введение

3.5.1.1. Область применения

Настоящий документ предназначен для сотрудников эксплуатирующей организации, в должностные обязанности которых входят функции по администрированию информационных систем.

3.5.1.2. Требования к уровню подготовки администратора

Для штатной эксплуатации системы необходимо привлечение следующих групп персонала:

– системный администратор;

– администратор баз данных.

– Основные обязанности системного администратора:

– установка, настройка и мониторинг работоспособности системы управления контентом.

– Основные обязанности администратора баз данных:

– мониторинг целостности базы данных, своевременное создание резервных копий базы.

Системный администратор должен обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию программных средств, а также должен иметь профессиональные знания и практический опыт в области системного администрирования. Обязательны знакомство и практический опыт установки и администрирования серверных операционных систем семейства Windows, Linux, программного обеспечения Apach/Nginx, Docker.

Администратор баз данных должен обладать высоким уровнем квалификации и практическим опытом выполнения работ по установке, настройке и администрированию СУБД.

Роли системного администратора и администратора баз данных могут быть совмещены в одну роль.

3.5.2. Условия применения

Серверная часть системы управления контентом может функционировать и выполнять нужные задачи при условии соответствия требованиям, которые предъявляются к техническому, системному и прикладному программному обеспечению.

Общая структура комплекса технических средств должна включать:

– сервер, на котором будет обеспечиваться выполнение приложения;
– систему хранения данных;
– коммутационное оборудования для соединения с сетью.

Возможные варианты системного программного обеспечения, которые могут поддерживать функционирование серверной части системы управления контентом перечислены в таблице 6.

Таблица 6 – Возможные варианты системного программного обеспечения

Вид программного обеспечения	**Вариант**
Операционная система	64-разрядные операционные системы, поддерживаемые программными продуктами рассматриваемых вариантов (например CentOS 7 или Windows Server)
СУБД	PostgreSQL
Среда выполнения	Node.js
Среда запуска	Docker

Дистрибутив серверной части системы управления контентом представляет собой скомпилированный main.js файл и обязательные для запуска прикладные компоненты.

3.5.3. Установка и настройка дистрибутива

Предполагается предварительная установка и настройка ПО Docker на сервере, посредством которого описываются дальнейшие шаги по развертыванию серверной части системы управления контентом.

3.5.3.1. Создание Docker файла.

Для запуска с помощью ПО docker необходимо создать docker-файл. Его листинг представлен в таблице 7.

Таблица 7 – Листинг docker файла.

Dockerfile
FROM NODE:12.13-alpine WORKDIR /app COPY package*.json ./ RUN npm install COPY ./dist ./dist CMD [“npm”, “run”, “starr”]

3.5.3.2. Создание Docker-compose файла.

Следующим шагом необходимо создать исполняемый docker-compose файл, который и будет запускать среду с сервером на исполнение. Листинг представлен в таблице 8.

Таблица 8 – Листинг docker-compose файла.

Docker-compose файл
version ‘3.0’ service: main: container_name: main build: context env_file: - .development.env volumes: - .:/app - /app/node_modules ports: - {внутренний порт}:{внешний порт} - {внутренний порт}:{внешний порт} command: npm run start depends_on: postgres restart: always postgres: container_name: postgres image: postgres:12 environment: PG_DATA: /var/lib/postgresql/data ports: - {внешний порт}:5432 volumes: - pgdata: /var/lib/postgresql/data restart: always

3.5.3.3. Запуск

Необходимо перейти в директорию с docker-compose файлом и инициировать запуск командой docker-compose up. Логирование укажет на факт запуска, после чего можно вести обращение к серверу.

ЗАКЛЮЧЕНИЕ

В ходе работы были проанализированы наиболее популярные CMS-системы, выделены критерии оценки и охарактеризованы каждая из систем. На основе этого анализа были выделены основные качества разрабатываемой системы, описаны принципы работы и выдвинуто авторское решение поставленных задач.

На этапе проектирования были описаны основные функции разрабатываемой системы, выделены средства разработки, описана схема взаимодействия компонентов и очередь вызовов функций в системе, а также спроектирована инфологическая модель базы данных.

В ходе этапа разработки была реализована база данных, серверная часть и клиентская часть системы управления контентом. При разработке были использованы ведущие программные технологии, основанные на web-разработке, позволяющие обеспечить кроссплатформенность решения задачи, а также повышает производительность за счёт оптимизации работы кода.

Описано руководство пользователя по имеющемуся функционалу системы в рамках клиентской части и руководство администратора по развёртыванию системы в локальной сети предприятия на основе docker-технологии.

Платформа для упаковки Docker позволит упростить развертывание и масштабирование приложения, обеспечит изоляцию отдельных компонентов системы и быстрый их запуск за счёт контейнеризации.

В настоящее время система тестируется на базе ФГБОУ ВО «БГПУ», где используется ссылочная и фреймовая интеграция на официальный сайт образовательного учреждения. Потенциально, система имеет универсальную направленность и может быть внедрена на любое предприятие, в любой момент времени.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Грамотная клиент-серверная архитектура: как правильно проектировать и разрабатывать web API [Электронный ресурс]. – Режим доступа: https://tproger.ru/articles/web-api/. – 04.12.2022.

3. Колисниченко Д.Н. Выбираем лучший движок для сайта / Д.Н. Колисниченко – СПб: BHV-СПб, 2020 г. – 288 с.

4. Ниткин С.В. Разработка приложений типа клиент/сервер для Linux / Ниткин С.В., Жакин И.А., Хачиров Т.С. – Минск: Издательство Премьера, 2019 г. – 390с.

5. Серов, К. В. Введение в UML / К. В. Серов, Д. А. Мовчан, Н. М. Мухин. – СПб: Юрайт, 2021 г. – 496 с.

6. Шакалов, А.Ю. Web-дизайн: HTML, JavaScipt и CSS. Карманный справочник / А.Ю. Шакалов - Минск : КУДИЦ-ПРЕСС, 2021 г. – 320с.

7. IT Система SBMT. / Институт бизнеса и менеджмента технологий БГУ. [Электронный ресурс]. – www.sbmt.bsu.by. – Режим доступа: http://it.sbmt.by/about/indexru.htm. – Дата доступа: 08.10.2022.

8. Nest.js/Основы сервера [Электронный ресурс] – Режим доступа: https://www.codeflow.site/ru/article/nestjs – 01.12.2022.

9. Nest.js/Быстрый старт [Электронный ресурс] – Режим доступа: https://docs.nestjs.com/ru/getting-started/quick-start/ – 01.12.2022.

10. Карпова, Т.С. Базы данных: модели, разработка, реализация / Т.С. Карпова. – М. : НОУ «ИНТУИТ», 2022 г. – 241 с.

11. Баженова, И.Ю. SQL и процедурно-ориентированные языки / И.Ю. Баженова. – М. : НОУ «ИНТУИТ», 2020 г. – 167 с.

12. Гущин, А.Н. Базы данных / А.Н. Гущин. – М. : Директ-Медиа, 2021 г. – 266с.

13. Фрост, Р. Базы данных, Проектирование и разработка / Р. Фрост, Дж. Дей, С. Крейг. – М. : НТ Пресс, 2020 г. – 592 с.

14. Гарретт, Дж. Веб-дизайн / Дж. Гаррет. – СПб. : Символ-Плюс, 2022 г. – 198с.

15. Дунаев, В. HTML, скрипты и стили / В. Дунаев. – СПб. : Питер, 2023 г. – 118с.

16. Макдональд, М. Создание веб-сайтов. Основное руководство / М. Макдональд – М. : Эксмо, 2021 г. – 309с.

17. Гото, Г. Web-дизайн / Г. Гото, Э. Котлер. – СПб.: Символ Плюс, 2020 г. – 416с.

18. Ломов, А. HTML, CSS, скрипты: практика создания сайтов / А. Ломов – СПб.: БХВ-Петербург, 2021 г. – 416с.

19. Серых, Ю.А. Современный веб-дизайн. Эпоха веб 3.0 / Ю.А. Серых. – М. : «Диалектика», 2022 г. – 368с.

20. Структура сайта: основные виды [Электронный ресурс]: офиц. Сайт. – Режим доступа: http://sait-sozdat.ru/etap-sozdanie-sait/proekt-sait/sait-struktura-osnovnye-vidy.php. - 15.03.2023.

21. Ситиков, А. Создание системы управления контентом для web-сайтов / А. Ситиков. – Молодой ученый №47, 2022 г. – 28с.

22. Ситиков, А. Создание системы управления контентом для web-сайтов / А. Ситиков. – Молодёжь XXI века: шаг в будущее, 2022 г. – 218с.