Минобрнауки России

Юго-Западный государственный университет

Кафедра программной инженерии

ОТЧЕТ

о преддипломной (производственной) практике

наименование вида и типа практики

на (в) ООО «Предприятие ВТИ-Сервис»

наименование предприятия, организации, учреждения

Студента 4 курса, группы ПО-01б

курса, группы

Украинцева Дмитрия Игоревича

фамилия, имя, отчество

Руководитель практики от предприятия, организации, учреждения

директор

должность, звание, степень

Куркина А. В.

фамилия и. о.

Оценка _____

_____

подпись, дата

Руководитель практики от университета

к.т.н. доцент

должность, звание, степень

Чаплыгин А. А.

фамилия и. о.

Оценка _____

_____

подпись, дата

Члены комиссии

_____ _____

подпись, дата фамилия и. о.

_____ _____

подпись, дата фамилия и. о.

_____ _____

подпись, дата фамилия и. о.

Курск 2024 г.

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

БД – база данных.

ИС – информационная система.

ИТ – информационные технологии.

КТС – комплекс технических средств.

ОМТС – отдел материально-технического снабжения.

ПО – программное обеспечение.

РП – рабочий проект.

СУБД – система управления базами данных.

ТЗ – техническое задание.

ТП – технический проект.

UML (Unified Modelling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения.

1 Анализ предметной области

1.1 Обзор истории CMS

Широкая популярность Интернета обусловнена появлением Всемирной паутины в 1991 году. Изначально считалось, что привлекательность дизайна сайтов не столь важна, как предоставляемая ими информация. Это было связанно с ограниченными возможностями компьютерного оборудования. Веб-сайты были статичными и создавались вручную с использованием HTML-разметки.

С ростом мощности персональных компьютеров и появлением таких технологий как JavaScript (1995) и CSS (1996), интернет стал более наглядным и функциональным.

Параллельно развивалось серверное программирование в 90-е годы, появились языки программирования, такие как PHP (1995), Java (1995), технология Active Server Pages (1996), и система управления базами данных MySQL (1994).

В 2005 году появилась технология AJAX, позволяющая обновлять данные без перезагрузки страницы. Быстрое развитие программного обеспечения привело к разделению веб-сайтов на функциональные блоки: контент (MySQL, HTML), дизайн (CSS) и бизнес-логика (PHP, JavaScript).

В конце 90-х годов наблюдался стремительный рост интернет-контента, что побудило предприятия использовать корпоративные веб-сайты, однако их поддержка в основном выполнялась вручную программистами, затрудняя своевременную публикацию контента. Это создало потребность в системах, автоматизирующих и оптимизирующих процессы работы с контентом, таких как системы управления контентом (CMS).

Первые CMS появились в середине 1990-х годов, разрабатывались организациями самостоятельно и ориентировались на нужды конкретных компаний. В период С 1995 по 1997 годы появились системы управления корпоративным контентом, такие как FileNet, StoryBuilder, Intercloth, Documentum, FatWire, FutureTense и Inso.

С начала 2000-х годов происходит активное создание систем управления веб-контентом (WCMS). В это время утвердилось мнение о том, что современный сайт состоит из двух ключевых компонентов – дизайна и контента. Программисты отвечают за дизайн, а профессионалы в предметной области обеспечивают контент. Это способствовало привлечению большого числа участников к созданию сайтов, что привело к увеличению объема и качества информации в Интернете.

Появление CMS с открытым исходным кодом, таких как Mambo, Drupal, WordPress и Joomla, а также коммерческих CMS, таких как NetCat, Shop-Script, Битрикс: Управление сайтом 3.0 и CS-Cart, сделало создание сайтов доступным для широкого круга пользователей.

WCMS формировались вокруг четырех основных функций: создание контента с использованием редакторов WYSIWYG, управление контентом, публикация контента на сайте и презентация данных для улучшения их визуального представления.

Определились различные группы пользователей, включая дизайнеров, администраторов, команду внедрения и авторов контента.

Уникальность WCMS обусловлена многочисленными шаблонами, плагинами и доступом к CSS.

Тем не менее, первые WCMS имели некоторые недостатки, такие как сложность конфигурации CSS, ограниченную функциональность редакторов WYSIWYG и ограниченный круг пользователей, способных создавать контент.

На дальнейшее развитие WCMS, оказали влияние несколько выжных факторов.

Во-первых, это дальнейшее развитие вычислительной техники. Мощность современных смартфонов превосходит ту, которую имел персональный компьютер 20 лет назад.

Во-вторых, появление в 2005 году концепции и технологии Web 2.0, социальных сетей и облачных вычислений. Web 2.0 расширил возможности Интернета в целом и WCMS в частности. Теперь не только отдельные люди, но

и целые сообщества могли вносить свой вклад в информационные ресурсы, что привело к увеличению объема доступной информации. В результате возникла потребность в более простых инструментах для работы с контентом, таких как вики-разметка и онлайн-редакторы. Это также вызвало рост спроса на интерфейсы, ориентированные на непрофессионалов в информационных системах. Появляются новые более удобные и функциональные версии редакторов WYSIWYG, а установка и первоначальная настройка WCMS стала гораздо быстрее и проще. Рост социальных сетей требует интеграции с ними WCMS, которая происходит через плагины для автоматического связывания и регистрации через социальные сети.

Третий фактор – быстрое развитие мобильных технологий и увеличение трафика с мобильных устройств влияет на тенденции развития веб-сайтов.

1.2 Основные понятия CMS

CMS – программный комплекс, используемый для обеспечения и организации совместного процесса создания, редактирования и контроля содержимого веб-страниц. Содержимое может включать текст, изображения, видео, аудиофайлы, документы, мультимедийные файлы и многое другое.

CMS позволяет нетехническим пользователям легко управлять и обновлять контент на веб-сайте, не требуя навыков программирования или веб-разработки.

CMS, как правило, имеет модульную архитектуру, обеспечивающую легкую интеграцию плагинов и расширений, которые могут быть настроены для удовлетворения конкретных требований или расширения функциональности.

Пользователи CMS, такие как авторы или редакторы, создают контент с помощью WYSIWYG редактора, позволяющего легко форматировать и манипулировать текстом, изображениями и мультимедийными компонентами.

Созданный контент хранится в базе данных вместе с метаданными, такими как информация об авторе, категории и теги, которые облегчают организацию и возможность поиска.

Авторизованные пользователи могут управлять содержимым, выполняя такие действия, как редактирование, просмотр, утверждение или удаление содержимого, а также управление ролями пользователей и разрешениями на доступ.

Когда пользователь запрашивает определенную страницу или ресурс, CMS извлекает соответствующий контент из базы данных, обрабатывает его, используя шаблоны и темы для стилизации, и генерирует окончательный HTML-документ, который затем передается в веб-браузер пользователя.

1.3 Структура CMS

Хотя платформы CMS могут различаться по функциональности, у них есть общие основные компоненты. Эти компоненты включают в себя:

- приложение для управления контентом (CMA). Приложение для управления контентом (CMA) – это пользовательский интерфейс, который позволяет создателям и редакторам контента создавать, изменять и удалять контент с веб-сайта без необходимости наличия технических знаний. Это часть CMS, которую чаще всего используют создатели контента и администраторы;

- приложение доставки контента (CDA). Приложение доставки контента (CDA) отвечает за хранение и доставку контента конечным пользователям. Он извлекает содержимое из базы данных, объединяет его с соответствующими шаблонами и отображает на веб-сайте. Этот процесс происходит в фоновом режиме и невидим для создателей контента и администраторов;

- база данных. База данных хранит и упорядочивает контент и метаданные веб-сайта. Платформы CMS обычно используют базы данных для хранения контента, шаблонов, пользовательской информации и конфигураций.

1.4 Классификация CMS

Существует несколько типов CMS, каждая из которых отличается архитектурой, функциональностью и вариантами использования. Выделяют три основных типа CMS:

- монолитная (связанная) CMS. Монолитная или совмещенная CMS – это традиционная система управления контентом с тесно интегрированными уровнями управления контентом и представления. Этот тип CMS поставляется со встроенными шаблонами и инструментами дизайна для создания и поддержания внешнего вида веб-сайта. Монолитные платформы CMS обычно предлагают более оптимизированный опыт для нетехнических пользователей, но они могут быть менее гибкими, чем безголовые или развязанные варианты CMS;

- безголовая CMS. Безголовая CMS – это система управления контентом, которая не имеет внешнего интерфейса или уровня представления. Вместо этого контент отделен от представления, что позволяет разработчикам выбирать любую интерфейсную технологию для отображения контента. В безголовой CMS контент управляется через API (программные интерфейсы приложений), которые могут обслуживать контент на нескольких устройствах и платформах, что делает его популярным выбором для предприятий с несколькими каналами доставки, такими как веб-сайты, мобильные приложения и устройства IoT;

- развязанная CMS. Развязанная CMS – это гибрид безголовой и традиционной монолитной (связанной) CMS. Как и безголовая CMS, несвязанная CMS отделяет управление контентом от уровня представления. Тем не менее, он также включает в себя встроенные интерфейсные шаблоны и инструменты, позволяющие создавать и предварительно просматривать контент перед запуском. Это позволяет создателям контента иметь больший контроль над представлением своего контента, в то же время используя преимущества гибкости и масштабируемости несвязанной архитектуры.

2 Техническое задание

2.1 Основание для разработки

Основанием для разработки является задание на выпускную квалификационную работу бакалавра «Разработка системы управления содержимым веб-сайтов».

2.2 Цель и назначение разработки

Разрабатываемая программно-информационная система предназначена для совместного создания и управления веб-сайтами, не требуя от пользователей навыков программирования и веб-разработки.

Задачами данной разработки являются:

- разработка пользовательского интерфейса (административной панели) системы;
- разработка структуры базы данных для хранения информации о сущностях системы;
- разработка серверной части системы;
- разработка визуального редактора контента;
- реализация управления пользователями и правами доступа;
- реализация возможности выбора и применения различных тем (шаблонов) веб-сайта;
- реализация возможности совместной работы нескольких пользователей над контентом;
- организация хранилища медиафайлов.

2.3 Требования к программной системе

2.3.1 Требования к данных программной системы

Входными данными для программной системы являются:

- содержимое страниц и записей (постов): текст, медиафайлы;
- категории и теги для организации записей (постов);

- метаданные для страниц: заголовок, описание, ключевые слова;

- пользовательские данные: регистрационная информация (имя пользователя, пароль);

- файлы темы (шаблона) оформления сайта;

- настройки и параметры: название, адрес сайта.

Выходными данными для программной системы будут:

- отображение данных в виде веб-страниц;

- вывод содержимого медиафайлов на веб-страницах;

- использование тем (шаблонов) для оформления содержимомого страниц сайта;

- применение настроек и параметров к сайту.

2.3.2 Функциональные требования к программной системе

На основании анализа предметной области в разрабатываемой программной системе должны быть реализованы следующие функции:

1. Аутентификация и авторизация пользователей.
2. Управление веб-страницами.
3. Управление записями (постами).
4. Управление категориями и тегами записей (постов).
5. Управление меню и навигацией на сайте.
6. Управление темой (шаблоном) сайта.
7. Управление пользователями.
8. Управление настройками сайта.

На рисунке 2.1 представлены функциональные требования к системе в виде диаграммы прецедентов нотации UML.

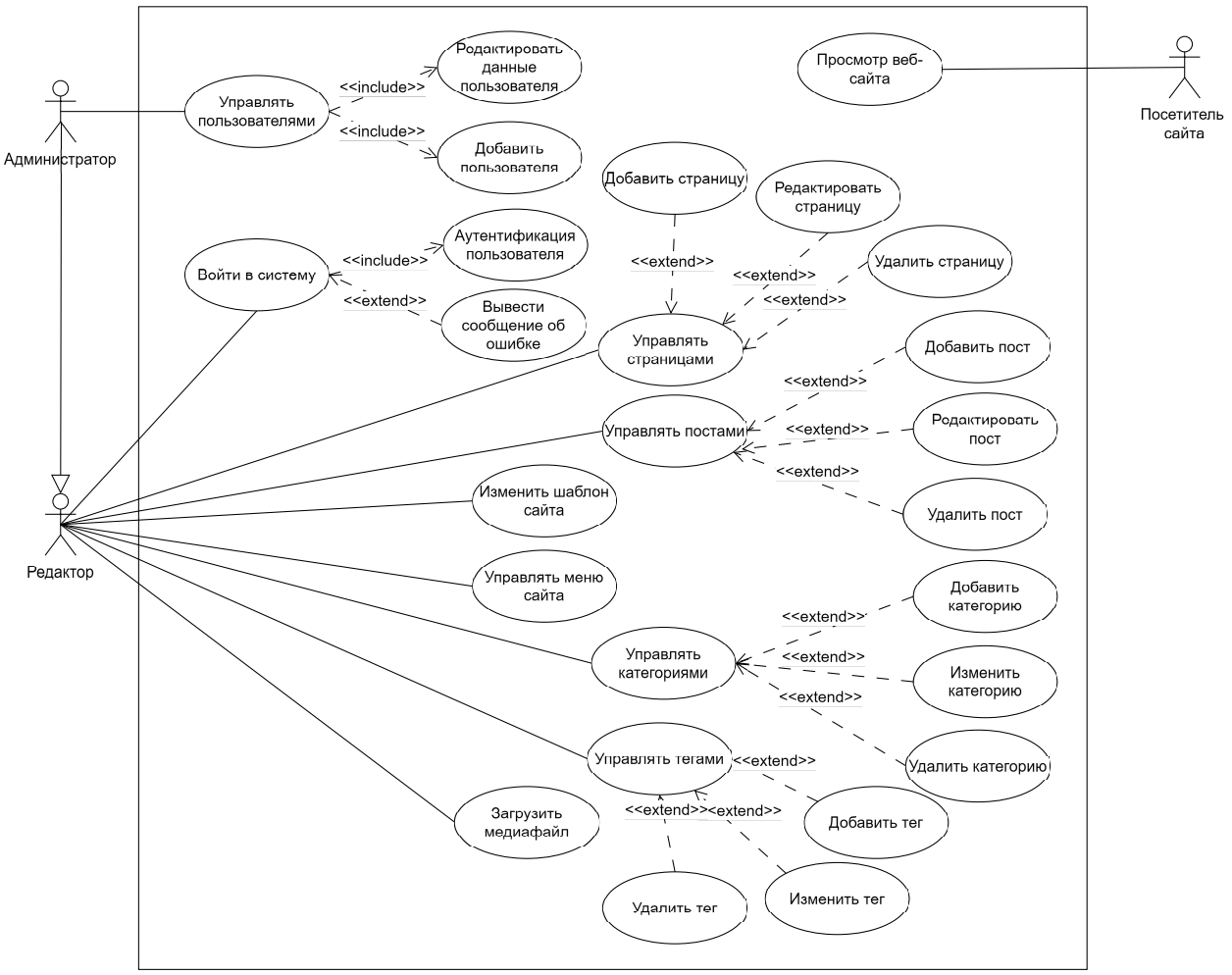

Рисунок 2.1 – Диаграмма прецедентов

2.3.3 Моделирование вариантов использования

2.3.3.1 Вариант использования «Войти в систему»

Заинтересованные лица и их требования: пользователь хочет поулчить доступ к административной панели сайта.

Предусловие: пользователь зарегистрирован в системе и знает свои данные для входа (логин и пароль).

Постусловие: система авторизует пользователя в соответствии с его ролью.

Основной успешный сценарий:

1. Пользователь вводит адрес административной панели сайта в браузере для перехода на страницу входа в систему.

2. Пользователь вводит логин и пароль.

3. Система проверяет корректность введенных данных и аутентифицирует пользователя.

4. Пользователь перенаправляется на главную страницу административной панели сайта.

2.3.3.2 Вариант использования «Управлять пользователями»

Заинтересованные лица и их требования: администратор хочет управлять учетными записями пользователей и их ролями в системе.

Предусловие: пользователь авторизован и имеет права администратора.

Постусловие: изменения в учетных записях пользователей сохранены в системе.

Основной успешный сценарий:

1. Администратор открывает раздел управления пользователями.

2. Администратор выбирает действие (создание, редактирование, удаление пользователя).

3. Администратор вводит необходимую информацию о пользователе.

4. Система сохраняет изменения и обновляет список пользователей.

2.3.3.3 Вариант использования «Управлять страницами сайта»

Заинтересованные лица и их требования: пользователь хочет управлять страницами сайта.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: изменения сохранены и отображаются на сайте.

Основной успешный сценарий:

1. Пользователь открывает раздел управления страницами.

2. Пользователь создает новую страницу или выбирает существующую для редактирования.

3. Пользователь переходит в режим редактирования страницы.

4. Пользователь добавляет новый или выбирает существующуй элемент страницы для редактирования.

5. Пользователь вносит необходимые изменения в содержимое элемента.

6. Пользователь вносит необходимую информацию о странице (название, URL, метаданные).

7. Пользователь сохраняет изменения.

2.3.3.4 Вариант использования «Управлять постами»

Заинтересованные лица и их требования: пользователь хочет упралять постами.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: изменения в постах сохранены и отображаются на соответствующей странице сайта.

Основной успешный сценарий:

1. Пользователь открывает раздел управления постами.

2. Пользователь создает новый пост или выбирает существующий для редактирования.

3. Пользователь вносит необходимые изменения в текст, заголовок, метаданные и т. д..

4. Пользователь сохраняет или публикует новый пост.

2.3.3.5 Вариант использования «Управлять категориями и тегами постов»

Заинтересованные лица и их требования: пользователь хочет управялть категориями и тегами постов.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: изменения в категориях и тегах сохранены и применены к соответствующим постам.

Основной успешный сценарий:

1. Пользователь открывает подраздел управления категориями и тегами постов.

2. Пользователь создает новую категорию/тег или редактирует существующие.

3. Пользователь присваивает их соответствующим постам.

4. Система сохраняет изменения и реорганизует отображение постов.

2.3.3.6 Вариант использования «Управлять меню и навигацией сайта»

Заинтересованные лица и их требования: пользователь хочет управялть навигационным меню сайта.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: изменения в меню сохранены и отображаются на сайте.

Основной успешный сценарий:

1. Пользователь открывает раздел управления меню и навигацией сайта.

2. Пользователь создает новый пункт меню, редактирует существующие.

3. Пользователь изменяет их порядок или структуру.

4. Пользователь сохраняет изменения.

2.3.3.7 Вариант использования «Изменить шаблон»

Заинтересованные лица и их требования: пользователь хочет изменить шаблон дизайна сайта.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: выбранный шаблон применен к страницам сайта.

Основной успешный сценарий:

1. Пользователь открывает раздел управления шаблонами административной панели.

2. Пользователь выбирает одну из доступных тем (шаблонов).

3. Пользователь нажимает соответствующую кнопку для активации темы (шаблона).

4. Система сохраняет изменения и обновляет шаблон сайта.

2.3.3.8 Вариант использования «Загрузить медиафайл»

Заинтересованные лица и их требования: пользователь хочет иметь возможность загрузки и удаления медиафайлов.

Предусловие: пользователь авторизован в системе и имеет соответствующие права доступа.

Постусловие: медиафайлы загружены и доступны для использования на сайте.

Основной успешный сценарий:

1. Пользователь открывает раздел управления медиафайлами административной панели.

2. Пользователь нажимает соответствующую кнопку для загрузки файла и выбирает файл.

3. Система сохраняет файл в соответствующую папку на сервере.

2.3.4 Требования пользователя к интерфесу программной системы

Интрефейс пользователя административной панели разрабатываемой системы управления содержимым должен включать следующие компоненты:

1. Навигационное меню.
2. Редактор контента.
3. Интерфейс управления страницами сайта.
4. Интерфейс управления записями (постами).
5. Интерфейс управления виджетами.
6. Интерфейс управления меню и навигацией на сайте.
7. Интерфейс управления пользователями.
8. Интерфейс управления темами (шаблонами) сайта.

9. Интерфейс управления медиафайлами.

Навигационное меню предназначено для перемещения по разделам и функциям CMS.

Интерфейс редактора контента включает панели инструментов с кнопками для форматирования текста, добавления ссылок, вставки изображений, видео, таблиц и т. д.

Интерфейс управления страницами сайта отображает список страниц сайта, включает кнопки для создания, редактирования, удаления страниц, поля для ввода данных страницы при добавлении или редактировании страницы.

Интерфейс управления записями (постами) отображает список записей, включает кнопки для создания, редактирования, удаления записей, поля для ввода данных при добавлении или редактировании записи.

Интерфейс управления медиафайлами отображает список загруженных файлов и включает кнопки для загрузки, просмотра, редактирования и удаления медиафайлов.

Интерфейс управления пользователями отображает список пользователей системы и включает кнопки для создания, редактирования и удаления учетных записей пользователей.

Интерфейс управления шаблонами отображает список доступных тем (шаблонов) и кнопку для активации выбранной темы.

Интерфейс управления меню и навигацией включает возможность управления навигационными меню и ссылками на страницы.

При реализации пользовательского интерфейса должны быть использованы следующие технологии:

- язык разметки веб-страниц HTML – для описания структуры страниц веб-интерфейса;

- каскадные таблицы стилей CSS – для стилизации элементов интерфейса;

- язык программирования JavaScript – для создания интерактивного интерфейса;

- библиотека jQuery языка программирования JavaScript – для обмена данными с сервером и обновления элементов интерфейса.

2.4 Нефункциональные требования к программной системе

2.4.1 Требования к надежности

В приложении не должно возникать критических ошибок, приводящих к экстренному завершению работы.

2.4.2 Требования к программному обеспечению

Для реализации программной системы должены быть использованы следующие технологии:

- язык программирования PHP – для разработки серверной части;
- СУБД MySQL – для хранения данных и организации данных;
- веб-сервер Apache HTTP Server – для обработки клиентских запросов.

2.4.3 Требования к аппаратному обеспечению

Для работы приложения, установленного на компьютер, необходимо дисковое пространство не менее 100 Мб, свободная оперативная память в размере не менее 1024 Мб, видеокарта с не менее 1024 Мб видеопамяти, клавиатура, мышь, установленная операционная система Windows, macOS или Linux, архитектура процессора x86 (Windows) или x64 (Windows, macOS, Linux).

Для доступа к административной панели системы, потребуется браузер Google Chrome, Mozilla Firefox или Microsoft Edge.

2.4.4 Требования к оформлению документации

Разработка программной документации и программного изделия должна производиться согласно ГОСТ 19.102-77 и ГОСТ 34.601-90. Единая система программной документации.

3 Технический проект

3.1 Общая характеристика организации решения задачи

Необходимо спроектировать и разработать серверную и клиентскую части программно-информационной системы.

Система управления содержимым веб-сайта состоит из различных компонентов, предназначенных для управления, создания, редактирования и публикации контента на веб-сайте. Основные компоненты разрабатываемой CMS:

1. Интерфейс управления (административная панель).
2. База данных.
3. Редактор контента.
4. Шаблоны и темы.

При разработке программы должно быть уделено внимание следующим ключевым аспектам:

1. Простота использования.
2. Масштабируемость.
3. Безопасность.
4. Производительность.

3.2 Обоснование выбора технологии проектирования

Выбранные для разработки программно-информационной системы языки и технологии предоставляют функции для создания эффективных и функциональных кроссбраузерных веб-приложений, позволяя создавать легко поддерживаемые и масштабируемые программные продукты.

3.2.1 Описание используемых технологий и языков программирования

3.2.1.1 Язык программирования PHP

PHP (Hypertext Preprocessor) – распространённый интерпретируемый язык общего назначения с открытым исходным кодом, который создавался специально для ведения веб-разработок, и код на нём встраивается непосредственно в HTML-код. Синтаксис языка берёт начало из языков C, Java и Perl и лёгок для изучения. Основная цель языка – помочь веб-разработчикам создавать динамически генерируемые веб-страницы.

Главная область применения PHP – написание скриптов, работающих на стороне сервера; таким образом, PHP способен выполнять все то, что выполняет любая другая программа CGI, например, обрабатывать данные форм, генерировать динамические страницы или отсылать и принимать cookies.

PHP отличается от JavaScript тем, что PHP-скрипты выполняются на сервере и генерируют HTML, который посылается клиенту.

Существуют три основных области применения PHP:

- создание скриптов для выполнения на стороне сервера;
- создание скриптов для выполнения в командной строке;
- создание оконных приложений, выполняющихся на стороне клиента.

PHP доступен для большинства операционных систем, включая Linux, многие модификации Unix (такие как HP-UX, Solaris и OpenBSD), Microsoft Windows, macOS, RISC OS и многие другие. Также в PHP включена поддержка большинства современных веб-серверов, таких как Apache, IIS и многих других.

Таким образом, PHP предоставляет свободу выбора операционной системы и веб-сервера. Более того, появляется выбор между использованием процедурного или объектно-ориентированного программирования (ООП) или же их сочетания.

Использование PHP не ограничивается выводом HTML. Возможности PHP включают вывод файлов различных типов, таких как изображения или

PDF-файлы, шифрование данных и отправку электронной почты. Можно выводить любой текст, например JSON или XML. PHP может автоматически генерировать эти файлы и сохранять их в файловой системе вместо вывода на печать, формируя серверный кеш для динамического содержимого.

Одним из значительных преимуществ PHP является поддержка широкого круга баз данных. Можно воспользоваться модулем, специфичным для отдельной базы данных (таким как mysql) или использовать уровень абстракции от базы данных, такой как PDO, или подсоединиться к любой базе данных, поддерживающей Открытый Стандарт Соединения Баз Данных (ODBC), с помощью одноимённого модуля ODBC.

PHP также поддерживает взаимодействие с другими сервисами через такие протоколы, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (на платформах Windows) и многих других.

3.2.1.2 Язык программирования JavaScript

JavaScript – это интерпретируемый язык программирования высокого уровня, который в основном используется в качестве языка сценариев для веб-разработки. Это одна из трех основных технологий Всемирной паутины наряду с HTML и CSS.

Язык программирования JavaScript позволяет создавать интерактивные веб-страницы и является неотъемлемой частью веб-приложений. В то время как HTML определяет структуру и макет веб-страницы, а CSS придает ей стиль, JavaScript делает ее интерактивной, обеспечивая динамическое содержание и взаимодействие с пользователем.

Веб-браузеры имеют встроенные механизмы для интерпретации и выполнения скриптов JavaScript, что позволяет языку работать непосредственно в браузере (фронтенд) без компилятора. Эта особенность JavaScript делает его языком клиентской стороны, хотя он также может использоваться на стороне сервера (бэкенд) с помощью таких сред, как Node.js.

Язык JavaScript поддерживает объектно-ориентированное программирование с прототипным наследованием, а также императивный и функци-

ональный стили программирования. В нем есть API для работы с текстом, массивами, датами, регулярными выражениями и объектной моделью документа (DOM), но он не включает никаких средств ввода-вывода, таких как сеть, хранилище или графические средства, полагаясь для этого на среду хоста, в которую он встроен.

3.2.1.3 Библиотека jQuery

jQuery – это быстрая, небольшая и многофункциональная библиотека языка программирования JavaScript, которая предоставляет множество полезных функций и инструментов для создания интерактивных и функциональных веб-приложений.

Одной из основных функций jQuery является возможность манипулировать HTML элементами на странице. С помощью этой библиотеки можно легко добавлять новые элементы, изменять их атрибуты или стили, а также удалять ненужные элементы.

jQuery позволяет легко обрабатывать различные виды событий на веб-странице. Например, можно обрабатывать клики по кнопкам, наведение курсора на элементы и многое другое.

jQuery упрощает использование AJAX-запросов, позволяя разработчикам отправлять асинхронные запросы на сервер без перезагрузки всей страницы.

3.2.1.4 Технология AJAX

AJAX (аббревиатура от Asynchronous JavaScript and XML) – это технология взаимодействия с сервером без перезагрузки страницы. Поскольку не требуется каждый раз обновлять страницу целиком, повышается скорость работы с сайтом и удобство его использования.

В работе технологии можно выделить 4 основных этапа:

1. Пользователь вызывает AJAX. Обычно это реализуется с помощью какой-либо кнопки, предлагающей получить больше информации.

2. Система отправляет на сервер запрос и всевозможные данные. Например, может потребоваться загрузка определенного файла либо конкретных сведений из базы данных.

3. Сервер получает ответ от базы данных и отправляет информацию в браузер.

4. JavaScript получает ответ, расшифровывает его и выводит пользователю.

Для обмена данными на странице создается объект XMLHttpRequest, он будет выполнять функцию посредника между браузером и сервером. Запросы могут отправляться в одном двух типов – GET и POST. Серверная часть обрабатывает поступающие данные и на их основании создает новую информацию, которая будет отправлена клиенту.

AJAX применяет асинхронную передачу данных. Такой подход позволяет пользователю совершать различные действия во время «фонового» обмена информации с сервером.

В качестве ответа сервер использует простой текст, XML и JSON.

3.3 Проектирование архитектуры программной системы

3.3.1 Описание сущностей программной системы

Исходя из требований изложенных в технческом задании, можно выделить следующие основные сущности проектируемой системы:

- «Пользователь»;
- «Страница»;
- «Пост»;
- «Категория»;
- «Тег»;
- «Пункт меню».

В состав сущности «Пользователь» можно включить атрибуты, представленные в таблице 3.1.

Таблица 3.1 – Атрибуты сущности «Пользователь»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
username	String	true	Логин
name	String	true	Имя пользователя
password_hash	String	true	Хэш пароля

В состав сущности «Страница» можно включить атрибуты, представленные в таблице 3.2.

Таблица 3.2 – Атрибуты сущности «Страница»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
title	String	true	Название страницы
content	Text	true	Содержимое страницы
parent_page_id	Integer	false	Идентификатор родительской страницы

В состав сущности «Пост» можно включить атрибуты, представленные в таблице 3.3.

Таблица 3.3 – Атрибуты сущности «Пост»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
title	String	true	Название поста
content	Text	true	Содержимое поста
author_id	Integer	true	Идентификатор автора поста
updated_datetime	DateTime	true	Дата создания/обновления поста

В состав сущности «Категория» можно включить атрибуты, представленные в таблице 3.4.

Таблица 3.4 – Атрибуты сущности «Категория»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
name	String	true	Название категории
parent_category_id	Text	false	Идентификатор родительской категории

В состав сущности «Тег» можно включить атрибуты, представленные в таблице 3.5.

Таблица 3.5 – Атрибуты сущности «Тег»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
name	String	true	Название тега

В состав сущности «Пункт меню» можно включить атрибуты, представленные в таблице 3.6.

Таблица 3.6 – Атрибуты сущности «Пункт меню»

Поле	Тип	Обязательное	Описание
1	2	3	4
_id	Integer	true	Уникальный идентификатор
menu_id	Integer	true	Идентификатор области меню
text	String	true	Текст ссылки
url	String	true	URL-адрес ссылки
parent_menu_item_id	Integer	false	Идентификатор родительского пункта меню
order_num	Integer	true	Порядковый номер пункта меню в пределах области меню

3.3.2 Компоненты программной системы

Диаграмма компонентов используется для визуализаци программной системы, ее структурных компонентов и связей (зависимостей) между компонентами. Компоненты могут быть программными модулями, библиотеками, пакетами и другими элементами, которые реализуют определенные функции системы. На рисунке 3.1 изображена диаграмма компонентов проектируемой системы.

Разрабатываемая программно-информационная система состоит из следующих основных компонентов:

1. Frontend (Пользовательский интерфейс) – включает файлы, которые описывают пользовательский интерфейс административной панели CMS, а также папки с файлами шаблонов, которые определяют внешний вид и структуру веб-страниц сайта, включая HTML, CSS и JavaScript.

2. Редактор контента – компонет системы который предоставляет инстументы для создания, редактирования и форматирования контента веб-страниц.

3. Backend (Серверная часть) включет следующие компоненты:

- Controllers (Контроллеры) – обрабатывает входящие HTTP-запросы клиента, вызывают методы моделей и определяют соответствующие представления для отображения данных;

- Models (Модели) – управляют данными и бизнес логикой, взаимодействуют с базой данных для получения, обновления и удаления данных;

- Views (Отображения) – отвечают за формирование HTML-страниц на основе данных полученных из моделей.

4. База данных – хранит структурированные данные в виде записей в таблицах, каждая таблица представляет определенную сущность системы.

5. Веб-сервер – программное обеспечение, которое принимает HTTP-запросы от клиентов и отвечает на них, предоставляя нужные ресурсы, такие как HTML-страницы, изображения, видео и другие данные.

Описание компонентов программной системы:

1. Page – отвечает за создание и управление статическими страницами и организацию разделов сайта.

2. Post – используется для управления динамическим контентом, таким как статьи в блоге, новости, обновления и другие материалы, которые публикуются регулярно.

3. Category – используется для организации контента на сайте, содержит функции для структурирования постов по темам или разделам.

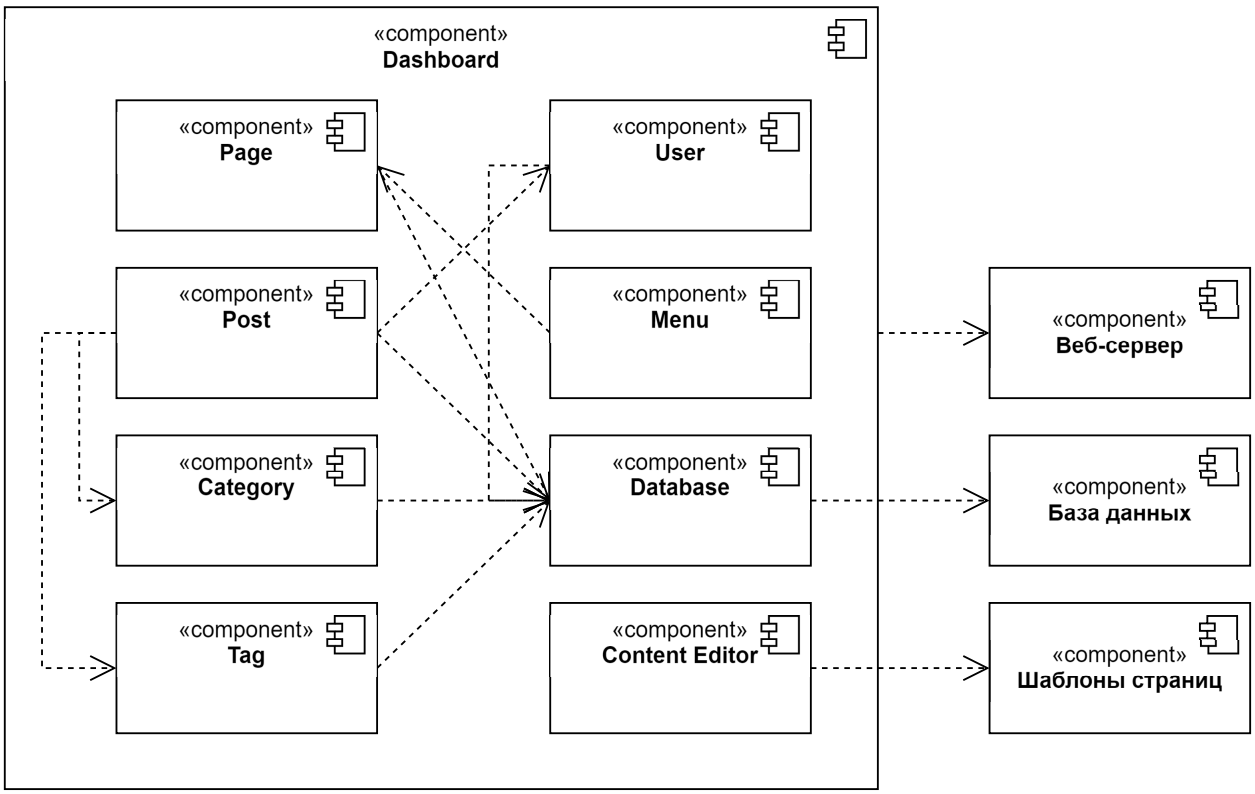

Рисунок 3.1 – Диаграмма компонентов

4. Tag – используется для дополнительной классификации контента, предоставляет более гибкую систему организации, чем категории, позволяя распределять посты по ключевым словам или темам.

5. User – этот компонент управляет информацией о пользователях сайта. Включает создание, редактирование и удаление учетных записей, управление ролями и правами доступа.

6. Menu – обеспечивает навигацию по сайту. Этот компонент позволяет создавать и управлять навигационными меню, которые могут содержать ссылки на страницы, посты, категории и другие элементы сайта.

7. Content Editor – представляет инструмент для создания и редактирования текстового и мультимедийного контента на сайте. Обычно он включает в себя визуальный интерфейс который позволяет пользователям форматировать текст, вставлять изображения, видео, ссылки и другие элементы без необходимости писать HTML код.

8. Database – обеспечивает создание и управление данными которые храняться в таблицах БД, отвечает за выполнение запросов к базе данных для получения, добавления, обновления и удаления информации.

3.3.3 Классы программной системы

На рисунке 3.3 представлена диаграмма классов программной системы.

3.4 Проектирование пользовательского интерфейса программной системы

На основании требований к пользовательскому интерфейсу представленных в пункте 2.3.4 технического задания, был разработан интерфейс администранивной панели системы и интерфейс редактора контента. Для создания пользовательского интерфейса используется язык разметки HTML и веб-фреймворк Bootstrap 5.

На рисунке 3.3 представлен макет главного окна однистративной панели CMS. Макет содержит следующие элементы:

1. Навигационное меню для перехода в соответствующий раздел панели управления.
2. Область отображения содержимого текущего раздела.

На рисунке 3.4 представлен макет раздела управления страницами. Макет содержит следующие элементы:

1. Кнопку «Добавить страницу» для добавленя новой страницы.
2. Список страниц сайта.
3. Название соответствующей страницы.
4. Кнопки управления страницей.

На рисунке 3.5 представлен макет раздела управления областями меню. Макет содержит следующие элементы:

1. Список областей меню сайта.
2. Кнопку «Изменить» для перехода на страницу управления пунктами выбранной области меню.

На рисунке 3.6 представлен макет раздела управления пунктами меню. Макет содержит следующие элементы:

1. Кнопку «Добавить» для добавленя нового пункта меню.
2. Список пунктов меню.
3. Кнопку для просмотра дочерних пунктов меню.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Фримен, А. Практикум по программированию на JavaScript / А. Фримен. – Москва : Вильямс, 2013. – 960 с. – ISBN 978-5-8459-1799-7. – Текст : непосредственный.

2. Бретт, М. PHP и MySQL. Исчерпывающее руководство / М. Бретт. – Санкт-Петербург : Питер, 2016. – 544 с. – ISBN 978-5-496-01049-8. – Текст : непосредственный.

3. Веру, Л. Секреты CSS. Идеальные решения ежедневных задач / Л. Веру. – Санкт-Петербург : Питер, 2016. – 336 с. – ISBN 978-5-496-02082-4. – Текст : непосредственный.

4. Гизберт, Д. PHP и MySQL / Д. Гизберт. – Москва : НТ Пресс, 2013. – 320 с. – ISBN 978-5-477-01174-2. – Текст : непосредственный.

5. Голдстайн, А. HTML5 и CSS3 для всех / А. Голдстайн, Л. Лазарис, Э. Уэйл. – Москва : Вильямс, 2012. – 368 с. – ISBN 978-5-699-57580-0. – Текст : непосредственный.

6. Дэкетт, Д. HTML и CSS. Разработка и создание веб-сайтов / Д. Дэкетт. – Москва : Эксмо, 2014. – 480 с. – ISBN 978-5-699-64193-2. – Текст : непосредственный.

7. Макфарланд, Д. Большая книга CSS / Д. Макфарланд. – Санкт-Петербург : Питер, 2012. – 560 с. – ISBN 978-5-496-02080-0. – Текст : непосредственный.

8. Лоусон, Б. Изучаем HTML5. Библиотека специалиста / Б. Лоусон, Р. Шарп. – Санкт-Петербург : Питер, 2013 – 286 с. – ISBN 978-5-459-01156-2. – Текст : непосредственный.

9. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

10. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

11. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

12. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

13. Титтел, Э. HTML5 и CSS3 для чайников / Э. Титтел, К. Минник. – Москва : Вильямс, 2016 – 400 с. – ISBN 978-1-118-65720-1. – Текст : непосредственный.

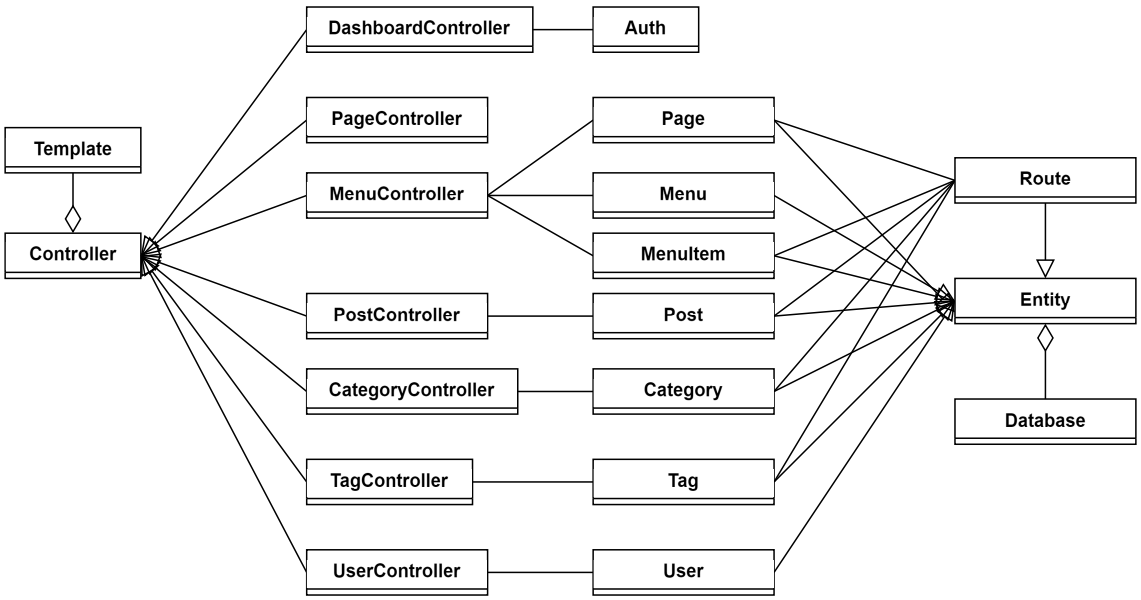

Рисунок 3.2 – Диаграмма классов

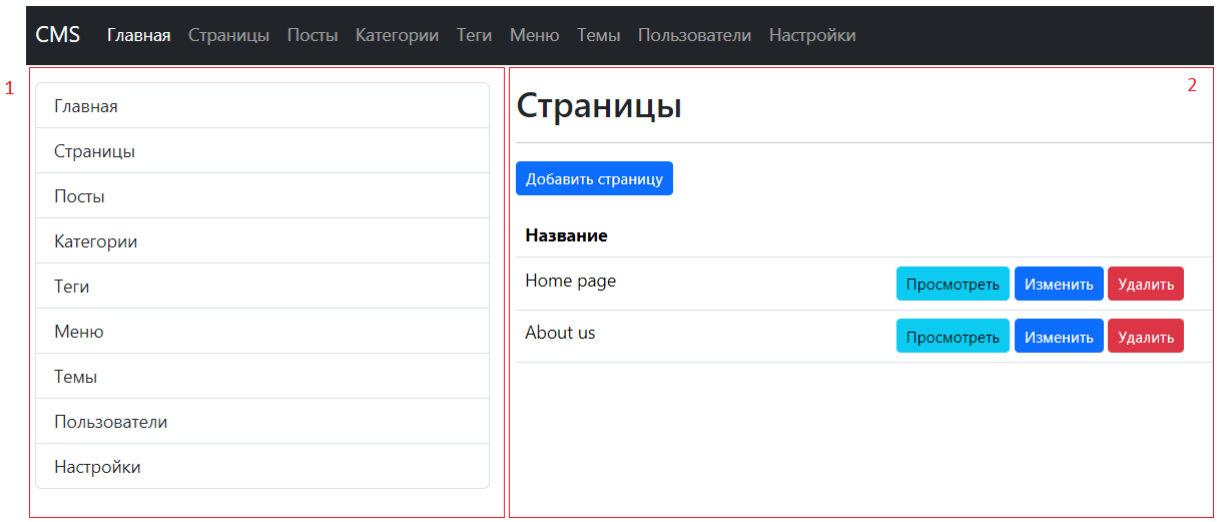

Рисунок 3.3 – Макет главного окна панели управления

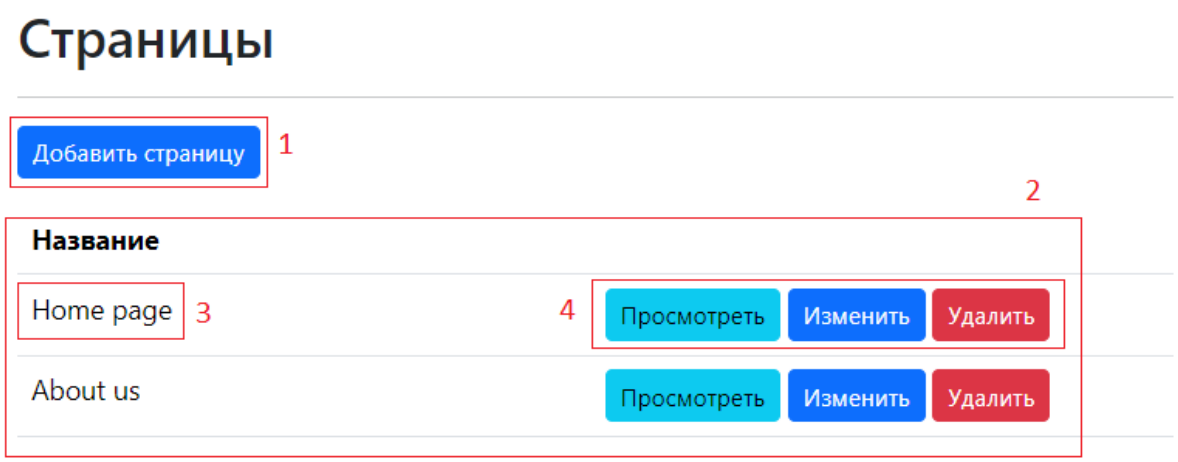

Рисунок 3.4 – Макет раздела управления страницами

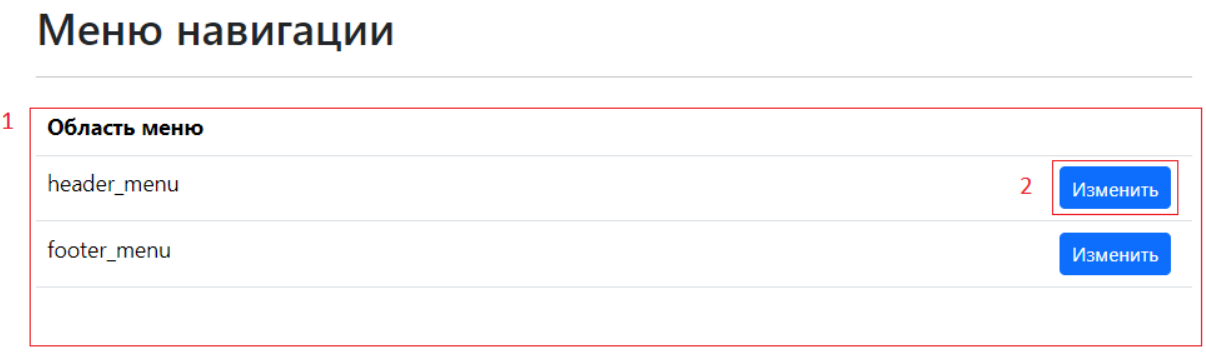

Рисунок 3.5 – Макет раздела управления областями меню

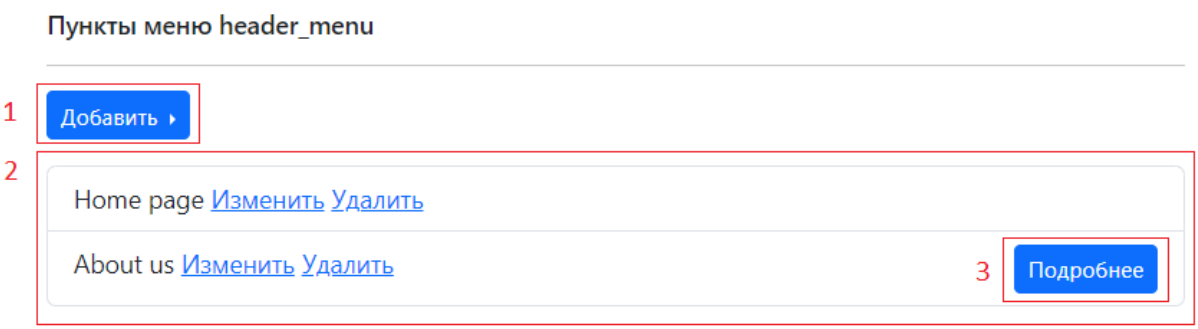

Рисунок 3.6 – Макет раздела управления пунктами меню