Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Институт повышения квалификации
и переподготовки

Кафедра «Информатика»

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

ПОСОБИЕ
для слушателей специальности переподготовки 9-09-0611-02 «Веб-дизайн и компьютерная графика» заочной формы обучения

Гомель 2025

УДК 004.42:004.738(075.8)
ББК 32.973.43я73
Р17

Рекомендовано советом института повышения квалификации и переподготовки ГГТУ им. П. О. Сухого
(*протокол № 2 от 23.06.2025 г.*)

Составитель *В. Н. Леонова*
Рецензент: ст. преподаватель кафедры «Информатика» ГГТУ им. П. О. Сухого
Н. В. Самовендюк

Разработка веб-приложений : пособие для слушателей специальности переподготовки 9-09-0611-02 «Веб-дизайн и компьютерная графика» заоч. формы обучения / В. Н. Леонова. – Гомель : ГГТУ им. П. О. Сухого, 2025. – 137 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 2 Gb RAM ; свободное место на HDD 16 Mb ; ATL Lunux 10.1 ; Adobe Acrobat Reader. – URL: http://elib.gstu.by. – Загл. с титул. экрана.

Р17

Данное пособие предназначено для ознакомления с основными принципами разработки веб-приложений. Курсом предусмотрено изучение систем управления контентом (CMS), критериев их выбора, а также приобретение практических навыков по созданию сайта на выбранной платформе. Особое внимание уделено вопросам использования фреймворков, техники оптимизации веб-ресурсов, подбора подходящего доменного имени и хостинга, а также методам защиты веб-приложений от угроз информационной безопасности.

Для слушателей специальности переподготовки 9-09-0611-02 «Веб-дизайн и компьютерная графика» ИПКиП.

УДК 004.42:004.738(075.8)
ББК 32.973.43я73

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном мире веб-приложения стали неотъемлемой частью повседневной жизни, обеспечивая пользователям доступ к услугам, информации и развлечениям. Их можно встретить на любом современном устройстве: телефоне, планшете, компьютере. Чаще всего это приложения: браузеры, чаты, «облачные хранилища», онлайн-игры — и таких приложений множество.

С учетом растущей значимости веб-приложений, разработка и изучение их основ становятся особенно актуальными.

Данное учебно-методическое пособие разработано в рамках дисциплины «Разработка веб-приложений» для слушателей специальности «Веб-дизайн и компьютерная графика» и предназначено для ознакомления с основными аспектами создания веб-приложений и простых сайтов, а также для понимания ключевых отличий между ними. В нем рассматриваются используемые технологии веб-разработки и взаимодействие их компонентов, что позволяет лучше понять архитектуру современных веб-решений.

Курс включает в себя:

- знакомство с основами веб-разработки и современными технологиями;
- методы проектирования и архитектуры, обеспечивающие создание эффективных и масштабируемых приложений;
- особенности применения фреймворков при разработке веб-сайтов;
- ознакомление с видами систем управления контентом (CMS) и критериями выбора подходящей системы для проекта;
- принцип создания сайта на CMS: запуск, задание базовых настроек, использование плагинов и шаблонов;
- принцип настройки прав доступов на сайт;
- знакомство с доменом и хостингом, а также рынком услуг;
- техническая оптимизация сайта;
- основы безопасности при разработке веб-сайтов с рекомендациями по защите веб-приложений.

В результате прохождения курса слушатели смогут овладеть теоретическими знаниями и использовать их на практике.

Понимание ключевых технологий и принципов проектирования позволит им создавать качественные и функциональные решения, соответствующие современным требованиям.

1. ВЕБ-ПРИЛОЖЕНИЯ И ВЕБ-СЕРВИСЫ

Веб-приложения и веб-сервисы отличаются по задачам, техническим особенностям и архитектуре. Их разработка требует учета специфики функционала и инфраструктуры, что обеспечивает эффективное выполнение поставленных целей и надежную работу системы.

1.1. Веб-приложения

Веб-приложение – это программное обеспечение, которое запускается на веб-сервере и доступно через веб-браузер. Веб-приложения не нужно устанавливать локально на компьютер пользователя. Для работы веб-приложения необходимо подключение к интернету, так как данные обрабатываются на сервере.

Веб-приложения запускаются в браузере, но не каждое приложение, которое требует браузера и сетевого соединения, обязательно является веб-приложением (например, некоторые простые сайты).

Веб-приложение определяется не только необходимостью сетевого соединения и браузера, но и уровнем интерактивности, динамичности и функциональности, которые оно предоставляет пользователю.

Задачи веб-приложений и примеры использования

Веб-приложения выполняют различные задачи, в зависимости от своих целей и предназначения. Ими являются: веб-почта, онлайн-калькуляторы, социальные сети, магазины электронной коммерции, онлайн-формы, потоковое видео, веб-приложения для совместной работы, например: Фигма (Figma.com), Канва (Canva.com) – графические онлайн-редакторы, GoogleDisk (google.com/drive) – облачное хранилище данных и пр.).

Учитывая их доступность, веб-приложения используются для задач, которые ранее требовали настольного программного обеспечения, таких как обработка текстов, создания электронных таблиц, и графических или видео-редактирование.

Основные группы задач, которые могут быть реализованы с помощью веб-приложений:

1. Управление данными: хранение информации и отображение данных.

Хранение информации позволяет пользователям сохранять и управлять данными, например, на веб-сайтах для заметок или бюджетирования.

Отображение данных предоставляет интерфейс для обработки и анализа информации, например, в виде аналитических панелей.

2. Общение и сотрудничество: социальные сети и документы для командной работы.

Социальные сети создают возможность взаимодействия между людьми (например, Ok, VK, YouTube, Facebook, Twitter и пр.)

Документы для командной работы: позволяют нескольким пользователям совместно редактировать документы (например, Google Docs - google.com/docs/).

3. Электронная коммерция: онлайн-магазины и платежные системы.

С помощью онлайн-магазинов осуществляется покупка товаров и услуг через интернет (например, Wildberries.by, Ozon.by).

Платежные системы обеспечивают безопасную оплату и обработку транзакций онлайн.

4. Образование и обучение: онлайн-курсы, тестирование и оценка.

Онлайн-курсы — платформы для удаленного обучения, которые предоставляют курсы и видеоуроки.

Тестирование и оценка упрощают процесс проведения экзаменов и онлайн-тестов.

5. Развлечения и медиа: стриминг видео и музыки, веб-игры.

Стриминг видео и музыки позволяют пользователям просматривать контент в реальном времени, например, на YouTube или Rutube.

Веб-игры — игры, которые можно играть прямо в браузере, например, онлайн-настольные и многопользовательские игры).

6. Информационные и новостные ресурсы: блоги и новостные сайты, форумы.

Блоги и новостные сайты — обеспечивают доступ к актуальным новостям и статьям.

Форумы и обсуждения — платформы для обмена мнениями и получения информации.

7. Сервисы и инструменты: календари и расписания, инструменты для разработки.

Календари и расписания позволяют пользователям планировать мероприятия и управлять своим временем, например, Google Calendar (google.com/calendar).

Инструменты для разработки — это платформы, позволяющие создавать, разрабатывать и тестировать программный код.

Веб-приложения обширны и многообразны, и их задачи могут варьироваться в зависимости от специфики работы и потребностей пользователей. Каждая задача требует различных технологий и подходов к разработке.

Технические особенности веб-приложений

Веб-приложения обладают рядом уникальных технических характеристик, которые определяют их функциональность, производительность и безопасность:

- Веб-приложения не требуют установки на устройство пользователя.
- Кросс-платформенность: веб-приложения доступны на любых устройствах с браузером.
- Централизованные обновления: изменения в веб-приложениях могут быть быстро внедрены и доступны всем пользователям одновременно без обновления вручную.
- Экономическая эффективность: одна и та же основа веб-приложений может использоваться для разных платформ.

Ключевые отличия между сайтом и веб-приложением

Между сайтом и веб-приложением существуют существенные различия:

Веб-сайт — это набор связанных веб-страниц, доступных через интернет, который может содержать текст, изображения, видео и другие медиа. Основная цель простого веб-сайта — предоставить информацию. Пример: блоги, корпоративные сайты, портфолио.

Веб-сайты могут быть частью веб-приложения, если они включают интерактивные элементы и функциональность, позволяющую пользователю взаимодействовать с ним.

Веб-приложение — это более интерактивная и динамичная форма веб-сайта, которая позволяет пользователям выполнять определенные действия, такие как ввод данных, взаимодействие с базами данных, выполнение расчетов и т.д. Веб-приложения часто требуют более сложной логики и могут включать функционал, аналогичный настольным приложениям. Таким образом, все веб-приложения — это

веб-сайты, но не все веб-сайты являются веб-приложениями. Например, новостной сайт или блог — это веб-сайты, а такие сервисы, как онлайн-банкинг, Google Таблицы (google.com/sheets) или Facebook (и другие социальные сети) — это веб-приложения.

С точки зрения веб-строения и архитектуры интернет-ресурсов, социальные сети являются отдельными веб-приложениями или платформами, которые функционируют как самостоятельные веб-сервисы. Они обладают следующими характеристиками:

1. Социальные сети — это сложные веб-приложения (Web Applications), предоставляющие пользователям интерфейс для взаимодействия, обмена контентом, коммуникации и создания пользовательского контента.

2. Клиент-серверная архитектура социальный сетей работает по модели «клиент — сервер», где браузер пользователя (клиент) взаимодействует с серверами соцсети для получения данных, отправки сообщений, публикации постов и т.д.

3. Социальные сети используют собственную инфраструктуру серверов, баз данных и API (интерфейсы программирования приложений), позволяющие интеграцию с внешними системами, сайтами и приложениями.

4. Каждая соцсеть имеет свой домен и размещается на своих серверах или облачных платформах (например, vk.com, ok.ru и т.д.).

5. Веб-строение предполагает возможность интеграции соцсетей через виджеты, плагины или API для отображения ленты новостей, авторизации через соцсети и т.п.

Таблица 1 - Основные отличия веб-сайта от веб-приложения

Характеристика	Веб-сайт	Веб-приложение
Технологии	HTML, CSS	HTML, CSS, JavaScript, часто серверные технологии
Цель	Информирование	Выполнение задач
Технологии	HTML, CSS	HTML, CSS, JavaScript, часто серверные технологии
Взаимодействие	Ограничено, в основном только для чтения	Высокая интерактивность; пользователи могут выполнять действия
Сложность	В целом – проще; базовая навигация	Более сложный; требует ввода данных пользователем и обработки
Функциональность	Страницы с текстом, изображениями и ссылками	Формы, учетные записи пользователей и обновления
Пользовательский интерфейс	Стандартизированный; последовательный макет	Настраиваемый; адаптированный к предпочтениям и задачам пользователя

Архитектура веб-приложений

Веб-приложения работают с использованием архитектуры клиент-сервер, а также слои API и базы данных. Пользователь взаимодействует с клиентской частью (Frontend), которая включает в себя пользовательский интерфейс, в то время как серверная часть управляет логикой и данными приложения.

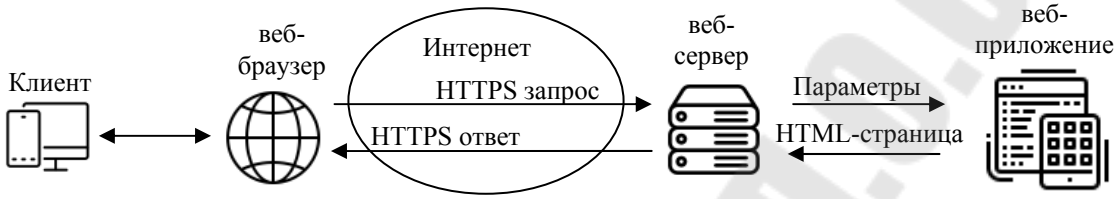

Рис. 1 – Общая схема взаимодействия пользователя (клиента) с веб-приложением

Типичными этапами архитектуры клиент-сервер являются (где клиент — это браузер или приложение, а сервер — веб-сервер и сервер приложений):

1. Пользователь открывает веб-приложение через браузер или приложение на своем устройстве, отправляя запрос на веб-сервер.
2. Веб-сервер перенаправляет запрос на сервер веб-приложений.
3. Сервер веб-приложений обрабатывает задачу и генерирует результаты.

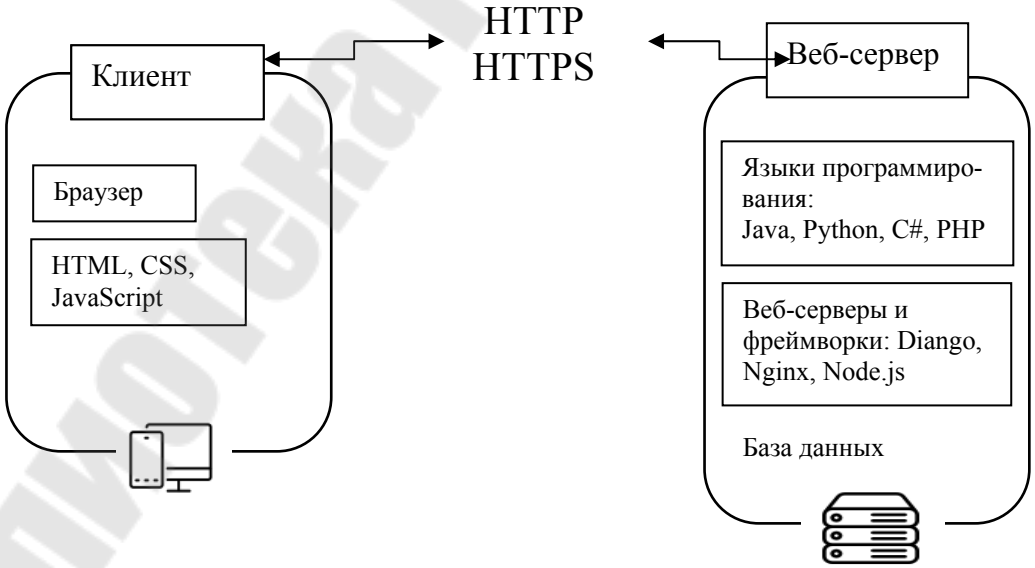

Рис. 2 – Схема взаимодействия клиентской и серверной частей веб-приложения

4. Сервер веб-приложений отправляет результаты обратно на веб-сервер.

5. Веб-сервер доставляет информацию на устройство пользователя, отображая ее на экране.

Клиентская часть – это то, что видит пользователь. Она обрабатывает пользовательский интерфейс и взаимодействие с пользователем.

1. Используемые технологии: HTML/CSS, JavaScript.

HTML/CSS – это основные технология для разметки и стилизации страниц.

JavaScript – это язык для создания динамического контента и взаимодействия с сервером, например, с использованием фреймворков как React или Vue.js.

2. Функции: обработка событий, таких как нажатия кнопок и ввод данных; отображение информации на веб-странице, полученной от сервера.

Серверная часть - это часть приложения, которая обрабатывает запросы от клиента и управляет логикой приложения.

1. Технологии: серверные языки программирования (такие как Python (с Flask или Django), PHP, Node.js, Ruby (с Ruby on Rails)) и веб-серверы (Apache или Nginx).

2. Функции: обработка запросов от клиентов и уровень бизнес-логики, включая аутентификацию, обработку данных и управление сессиями.

Слой API (Application Programming Interface) — это интерфейс, который позволяет различным приложениям взаимодействовать между собой.

1. Технологии: RESTful API – широко применяемый стиль архитектуры, использующий HTTP-запросы и GraphQL – современный подход, позволяющий запрашивать только те данные, которые нужны.

2. Функции: предоставление интерфейса для клиентских приложений для взаимодействия с серверной частью и обработка запросов на получение, изменение, добавление или удаление данных.

Слой базы данных — это компонент системы, отвечающий за хранение, управление и доступ к данным.

1. База данных — это место, где хранятся все данные приложения.

2. Технологии: реляционные базы данных (такие как MySQL, PostgreSQL) и нереляционные базы данных (такие как MongoDB, Firebase).

3. Функции: хранение данных в структурированном или неструктурированном виде и обеспечение надежного доступа и работы с данными, включая управление транзакциями и обработку запросов.

Архитектура веб-приложения может быть различной в зависимости от специфики проекта, однако вышеописанные компоненты обычно присутствуют практически во всех веб-приложениях.

1.2. Веб-сервисы

Веб-сервис — это программный интерфейс, который позволяет взаимодействовать различным приложениям через сеть.

Веб-приложение направлено на работу с пользователем и имеет пользовательский интерфейс. Впротивоположность этому, веб-сервис работает либо с другими веб-сервисами, либо с веб-приложениями. Обмен при этом происходит точно так же, как и в случае с приложениями, то есть по схеме запрос-ответ. В качестве клиента может выступать любая программа, которая правильно сформирует HTTP-запрос и расшифрует полученный HTTP-ответ.

Примеры веб-сервисов: API, облачные приложения, интегрированные инструменты.

Использование встроенных Google Таблиц на сайте является применением веб-сервиса. Встраивание таблиц на сайт позволяет отображать данные из Google Таблиц прямо на веб-странице, что требует доступа к интернету и использования серверных ресурсов Google. Данные в Google Таблицах могут автоматически обновляться, что делает приложение более динамичным и интерактивным.

Примеры использования веб-сервисов:

1. Картографические сервисы: встраивание карт и маршрутов из Google Maps в веб-сайты позволяет пользователям находить местоположения и получать навигационные указания в реальном времени.

2. Социальные сети: встраивание ленты социальных сетей с помощью API позволяет показывать на сайте свежие твиты из Twitter и пины из Pinterest.

3. Системы оплаты: интеграция кнопок оплаты ЮMoney, WebMoney, QIWI Кошелек и других систем позволяет пользователям совершать транзакции прямо на сайте без перехода на сторонние ресурсы.

4. Веб-формы: Создание опросов или регистрационных форм, которые можно встраивать из Яндекс Форм и Google Форм (Google Таблиц) на сайт для сбора данных и обратной связи от пользователей.

5. Чаты и поддержка: использование чат-ботов, например Chatbot API, для предоставления поддержки клиентам прямо на сайте.

6. Запись видео и стриминг: видеохостинг YouTube позволяет записывать видео, проводить онлайн-трансляции и встраивать видео на сайт для просмотра контента без необходимости покидать страницу.

Каждый из этих примеров демонстрирует, как веб-сервисы могут улучшить функциональность веб-сайта, обеспечивая пользователям удобство и доступ к дополнительной информации.

Задачи веб-сервисов: обмен данными между приложениями; интеграция различных систем; упрощение разработки за счёт использования готовых API.

Технические особенности: протоколы передачи: HTTP/HTTPS; форматы данных: JSON, XML; удобство интеграции: использование стандартов, облегчающих взаимодействие между разными системами.

Архитектура веб-сервисов

Архитектура веб-сервиса основана на модели клиент — сервер, с дополнительными компонентами взаимодействия и обмена данными. Типичными этапами архитектуры клиент-сервер являются:

1. Клиент: отправляет запросы к веб-сервису.
2. Сервер веб-сервиса: обрабатывает запросы и выполняет бизнес-логические функции.
3. База данных: хранит данные, с которыми работает веб-сервис.
4. Протоколы передачи данных: определяют, как данные передаются между клиентом и сервером (например, HTTP, SOAP, REST).

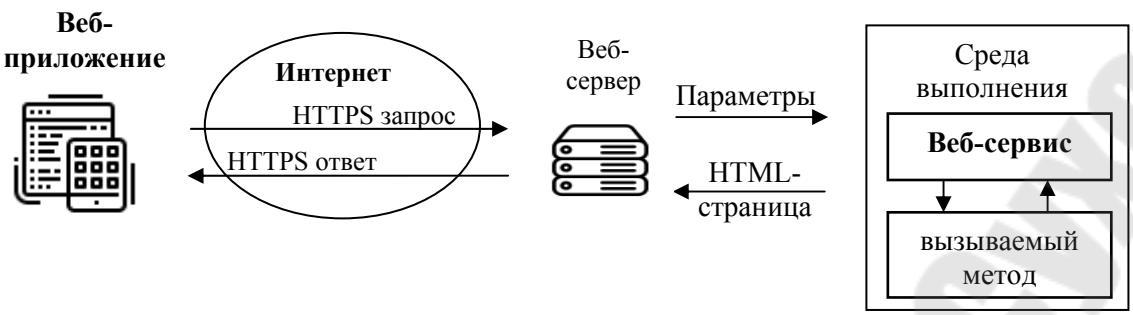

Рис. 3 – Схема взаимодействия веб-приложения и веб-сервиса

Информация с веб-сервиса возвращается к клиенту и отображается в пользовательском интерфейсе (например, на странице сайта или в приложении), где пользователь может ознакомиться с полученными результатами.

Недостатки веб-сервисов:

- Зависимость от интернет-соединения: в случае проблем с подключением их работа может быть нарушена.
- Безопасность: поскольку веб-сервисы передают данные через интернет, существует риск утечки данных или кибератак. Необходимо уделять особое внимание защите данных.
- Ограничения по функциональности из-за используемых технологий или протоколов.

Веб-приложения и веб-сервисы играют ключевую роль в современном программировании и веб-разработке. Понимание их архитектуры, задач и особенностей применения поможет разработчикам создавать гибкие и эффективные решения.

2. ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЙ

Современные системы управления контентом (CMS) широко используются для создания и поддержки веб-сайтов различного масштаба и назначения. В то время как большинство пользователей предпочитает готовые CMS, такие как WordPress, Joomla или Drupal, существует также возможность разработки индивидуальных систем, полностью адаптированных под конкретные требования бизнеса. Такой подход позволяет создать уникальную платформу, максимально соответствующую нуждам заказчика, и обеспечивает гибкость в дальнейшем развитии и расширении сайта.

2.1. Проектирование систем управления веб-сайтом

Проектирование систем управления веб-сайтом (CMS) включает в себя создание архитектуры и функциональности самой системы, которая будет использоваться для управления контентом на сайте. Это может включать:

1. Определение требований.

Первым шагом в проектировании системы управления веб-сайтом является определение требований. Это включает в себя:

1.1. Функциональные требования: какие функции должна выполнять система? Например, возможность добавления и редактирования страниц, загрузка медиафайлов, управление пользователями и правами доступа.

1.2. Нефункциональные требования: каковы требования к производительности, безопасности и доступности? Например, система должна поддерживать большое количество пользователей одновременно или обеспечивать защиту от несанкционированного доступа.

1.3. Целевая аудитория: кто будет использовать систему? Это могут быть администраторы сайта, редакторы контента или конечные пользователи.

2. Архитектура системы.

После определения требований необходимо разработать архитектуру системы. Это включает в себя выбор технологий и компонентов, которые будут использоваться для реализации CMS:

2.1. Frontend (визуальная часть, например, страница сайта или приложение, с которым взаимодействует пользователь) – это часть

системы, с которой взаимодействуют пользователи. Она должна быть интуитивно понятной и удобной для навигации. Для разработки интерфейса используются такие технологии, как: HTML, CSS и JavaScript, а также различные фреймворки (например, React или Vue.js).

2.2. Backend (невидимая часть системы, отвечающая за обработку данных, бизнес-логику и взаимодействие с базой данных): серверная часть системы отвечает за обработку запросов пользователей и взаимодействие с базой данных. Для разработки backend могут использовать языки программирования такие как PHP, Python, Node.js. Важно также выбрать подходящую базу данных (например, MySQL или PostgreSQL) для хранения контента.

2.3. API: разработка API (интерфейс программирования приложений) позволяет интегрировать систему с другими сервисами и приложениями. Это может быть полезно для обмена данными с внешними системами или мобильными приложениями.

3. Пользовательский интерфейс.

Проектирование пользовательского интерфейса (UI) является важным аспектом разработки CMS. Интерфейс должен быть простым и интуитивно понятным:

3.1. Навигация — удобная навигация помогает пользователям быстро находить нужные функции.

3.2. Редактор контента — обеспечение удобного редактора для создания и редактирования страниц, который позволит пользователям видеть изменения в реальном времени.

3.3. Адаптивный дизайн: интерфейс должен корректно отображаться на различных устройствах — от настольных компьютеров до мобильных телефонов.

4. Безопасность CMS:

4.1. Аутентификация и авторизация — реализация надежных механизмов аутентификации пользователей (например, через пароли или двухфакторную аутентификацию) и управление правами доступа.

4.2. Защита от атак — система должна быть защищена от распространенных угроз, таких как SQL-инъекции, XSS (межсайтовый скриптинг) и CSRF (межсайтовая подделка запроса). Регулярные обновления системы и использование проверенных библиотек помогут минимизировать риски.

5. Масштабируемость: при проектировании системы управления веб-сайтом важно учитывать возможность ее масштабирования:

5.1. Горизонтальное масштабирование: возможность добавления новых серверов для обработки увеличивающегося объема трафика.

5.2. Оптимизация производительности: использование кэширования данных и оптимизация запросов к базе данных помогут улучшить производительность системы при увеличении нагрузки.

Дополнительно, стоит также учесть:

- разработку структуры базы данных;
- проектирование пользовательского интерфейса для администраторов и конечных пользователей;
- определение модульной архитектуры и интеграции с другими системами – это позволит легко добавлять новые функции и расширять систему в будущем;
- обеспечение безопасности и производительности системы.

Проектирование систем управления веб-сайтами — это сложный процесс, требующий внимательного подхода к определению требований, архитектуре системы, пользовательскому интерфейсу, безопасности и масштабируемости.

Этапы разработки веб-сайта

Проектирование систем управления веб-сайтом и этапы разработки веб-сайта — тесно связаны между собой.

Этапы разработки веб-сайта охватывают весь процесс создания сайта от начала до конца. Это может включать:

- анализ требований: определение целей сайта, целевой аудитории и функциональных требований;
- этап проектирования: создание макетов, прототипов и архитектуры информации;
- разработка – программирование сайта, включая фронтенд и бэкенд;
- тестирование: проверка функциональности, производительности и безопасности сайта;
- размещение сайта на сервере и его публикация в интернете;
- обеспечение технической поддержки, обновление контента и функциональности.

Таким образом, проектирование систем управления веб-сайтом является неотъемлемой частью более широкого процесса разработки веб-приложений и веб-сайтов. Проектирование CMS фокусируется на создании системы для управления контентом, тогда как этапы разработки охватывают весь процесс создания сайта от концепции до запуска и дальнейшей поддержки.

Кастомизация и расширение функциональности CMS

Разработка Frontend и Backend в контексте систем управления контентом (CMS) может показаться излишней, поскольку многие CMS уже предоставляют готовые решения для этих компонентов. Однако есть несколько причин и способов, как можно и нужно разрабатывать Frontend и Backend даже при использовании существующих CMS:

1. Кастомизация интерфейса (Frontend).

Несмотря на то, что многие CMS предлагают стандартные шаблоны и темы, разработчики могут создавать кастомизированные (адаптированные) интерфейсы для улучшения пользовательского опыта:

1.1. Создание собственных тем – разработка уникальных тем, которая соответствует бренду или специфическим требованиям проекта. Это включает в себя создание пользовательских стилей, макетов и компонентов.

1.2. Использование фреймворков – интеграция современных JavaScript-фреймворков (например, React, Vue.js, Angular) к выбранной CMS для создания более интерактивного и отзывчивого интерфейса.

1.3. Разработка адаптивного интерфейса, который будет хорошо выглядеть на всех устройствах.

2. Расширение функциональности (Backend).

Несмотря на то, что, CMS предоставляет базовые функции, часто требуется дополнительная функциональность:

2.1. Создание дополнительных плагинов — многие CMS позволяют разработать собственные плагины или модули для добавления новых функций. Это может быть полезно для реализации специфических бизнес-требований.

2.3. Интеграция с внешними сервисами – разработка API или использование существующих API для интеграции с другими системами (например, CRM или платежными системами).

2.4. Оптимизация производительности — если стандартные функции не удовлетворяют требованиям по производительности необходимо оптимизировать запросы к базе данных или реализовать функцию кэширования.

3. Управление данными.

Разработка Backend может включать в себя управление данными более эффективно:

3.1. Кастомизация базы данных — в некоторых случаях может потребоваться изменение структуры базы данных или добавление новых таблиц для хранения специфических данных.

3.2. Создание пользовательских моделей данных: если стандартные модели данных не подходят для проекта, можно создать свои собственные модели.

4. Безопасность.

Разработка Frontend и Backend также может быть направлена на улучшение безопасности:

4.1. Аудит безопасности — внесение изменения в существующий код для повышения уровня безопасности.

4.2. Реализация дополнительных мер безопасности: добавление двухфакторной аутентификации или улучшение защиты от атак.

5. Обновления и поддержка.

Поддержка актуальности технологий: обновления Frontend и Backend позволяет использовать современные технологии и подходы, что повышает производительность и безопасность сайта.

Хотя готовые решения в CMS могут значительно упростить процесс разработки веб-сайта, кастомизация Frontend и Backend остается важной частью работы над проектом. Это позволяет адаптировать систему под конкретные требования бизнеса, улучшить пользовательский опыт и обеспечить безопасность. Разработчики могут использовать возможности существующих платформ как основу для создания уникальных решений, которые отвечают современным требованиям пользователей и бизнеса.

2.2. Проектирование базы данных

База данных — это структурированное хранилище данных, необходимое для эффективного функционирования веб-приложений и веб-сервисов. Правильное проектирование базы данных обеспечивает надежность, масштабируемость и быстродействие системы, а также облегчает последующее сопровождение и развитие.

Этапы проектирования базы данных

1. Анализ требований.

На этом этапе определяется набор данных, необходимых для работы системы, её функциональность и взаимодействие с пользователями. Важным аспектом является выявление ключевых бизнес-правил и сценариев использования.

2. Моделирование данных.

Создается концептуальная модель, которая отражает основные сущности и связи между ними.

Пример концептуальной модели базы данных для интернет-магазина:

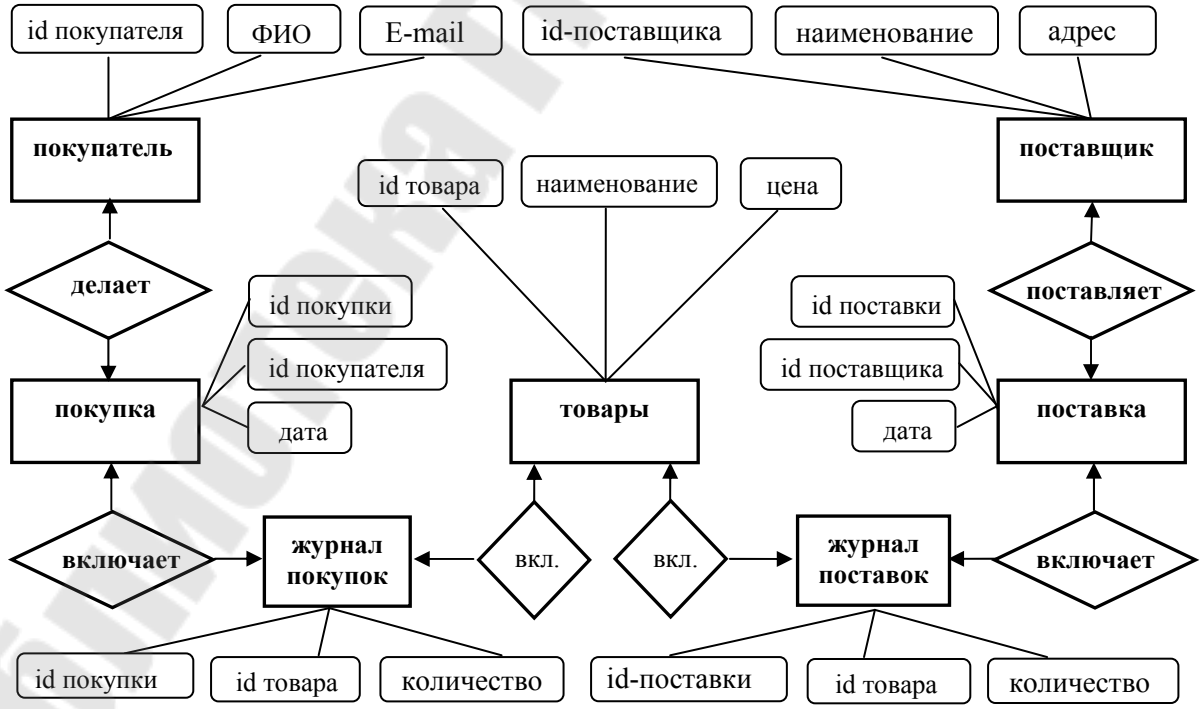

Рис. 4 – Концептуальная модель базы данных интернет-магазина

Затем переходят к логической модели, где каждую сущность преобразуют в таблицы, соотношения — в внешние ключи.

3. Нормализация – это процесс организации таблиц для устранения избыточности и обеспечения целостности данных.

4. Выбор типа базы данных - решение о типе системы (реляционная, документоориентированная, графовая и др.) в зависимости от требований проекта.

5. Проектирование структуры таблиц - определение таблиц, ключей (первичных и внешних), индексов для быстрого поиска и целостности данных.

Пример преобразования концептуальной модели базы данных в таблицы: определение таблиц, полей, первичных ключей (PK) и связей (FK).

Далее – концептуальная модель преобразовывается в систему управления базами данных (СУБД): MySQL, PostgreSQL, MongoDB и др.

Покупатель
Id покупателя (PK)
ФИО
E-mail

Поставщик
Id поставщика (PK)
Наименование
Адрес

Покупка
Id покупки (PK)
Id покупателя (FK)
Дата

Поставка
Id поставки (PK)
Id поставщика (FK)
Дата

Товар
Id товара (PK)
Наименование
Цена

Журнал покупок	
Id покупки (FK)	} PK
Id товара (FK)	
Количество	

Журнал поставок	
Id поставки (FK)	} PK
Id товара (FK)	
Количество	

Рис. 5 – Преобразование концептуальной модели базы данных в таблицы.

6. Обеспечение целостности и безопасности – внедрение правил для поддержания корректности данных и механизмов защиты информации.

7. Оптимизация производительности – планирование индексов, кэширования для повышения скорости работы с данными.

8. Масштабируемость и расширяемость – проектирование с учетом возможного роста объема данных и нагрузки.

9. Резервное копирование и восстановление – разработка стратегий для защиты данных от потери или повреждения.

10. Тестирование и документация – проверка структуры на предмет ошибок и подготовка документации для поддержки и развития системы.

Примеры проектов, для которых разрабатываются базы данных:

Таблица 2. Таблица баз данных для проектов:

Тип проекта	Описание примерной базы данных
Онлайн-магазин	Каталог товаров, пользователи, заказы, отзывы, платежи
Социальная сеть	Пользователи, друзья, посты, комментарии, сообщения
Бронирование гостиниц	Отели, номера, бронирования, отзывы, платежи
Образовательный портал	Пользователи, курсы, уроки, оценки, сертификаты
Финансовые сервисы	Банковские счета, транзакции, клиенты, кредиты

Проектирование баз данных для веб-приложений и веб-сервисов требует тщательного планирования и учета различных факторов, включая выбор СУБД, проектирование схемы, безопасность, оптимизацию производительности и интеграцию с другими приложениями. Хорошо спроектированная база данных обеспечивает устойчивую работу системы и комфортное взаимодействие пользователей с приложением.

Базы данных интегрированные в CMS

Системы управления контентом (CMS) поставляются с готовыми или предустановленными базами данных, которые уже настроены для хранения контента, настроек и других данных. Такие базы данных позволяют быстро начать работу с CMS без необходимости создавать структуру базы данных с нуля. Обычно эти базы данных проектируются под стандартные задачи и обеспечивают совместимость и удобство использования в рамках конкретной CMS.

Преимущества использования готовых баз данных для CMS: скорость внедрения — не тратится время на проектирование с нуля и стандартизация — проверенные схемы, совместимые с обновлениями CMS. Примеры баз данных популярных CMS:

Таблица 3. Таблица типовой структуры баз данных популярных CMS

CMS	Типовая структура базы данных	Характеристика
WordPress	Таблицы: wp_users, wp_posts, wp_comments, wp_terms, wp_postmeta	Традиционная реляционная схема для статей, комментариев, пользователей и мета-данных
Joomla	Таблицы: #__users, #__content, #__categories, #__extensions	Структура для контента, категорий, плагинов и модулей
Drupal	Таблицы: users, node, taxonomy_term_data, field_data_*, config	Модульная схема, поддерживающая расширения и дополнительные типы контента
OpenCart / WooCommerce	Таблицы: customers, orders, products, categories	Структурированные базы данных для интернет-магазинов

3. CMS - СИСТЕМА УПРАВЛЕНИЯ КОНТЕНТОМ

CMS (Content Management System) — это система, которая используется для быстрого создания сайтов и управления контентом. В мире существует множество популярных CMS, каждая из которых предлагает уникальные функции и возможности.

Количество CMS постоянно растет, так как новые платформы разрабатываются, а существующие обновляются. На данный момент можно выделить несколько сотен различных CMS, включая как популярные, так и менее известные решения.

Наиболее известные и широко используемые CMS:

- WordPress – популярная CMS в мире, используемая для создания блогов, корпоративных сайтов и интернет-магазинов.
- Joomla! – мощная и гибкая CMS, подходящая для создания различных типов сайтов.
- Drupal – гибкая платформа, часто используемая для сложных и масштабируемых проектов.
- Magento – специализированная CMS для создания интернет-магазинов.
- OpenCart – популярная платформа для электронной коммерции.
- Shopify – облачная платформа для создания интернет-магазинов.
- Blogger – платформа для ведения блогов от Google.
- 1С-Битрикс – мощная платформа для управления контентом и электронной коммерцией, популярная в России и странах СНГ, предлагающая широкий спектр инструментов для бизнеса.

- Tilda Publishing – конструктор сайтов с интуитивно понятным интерфейсом, позволяющий быстро создавать адаптивные сайты и лендинги без необходимости программирования.
- Wix – конструктор сайтов с элементами CMS, позволяющий создавать сайты без программирования.

Кроме этих популярных платформ, существует множество специализированных и нишевых CMS, которые могут быть предназначены для конкретных задач или отраслей.

В целом, точное количество существующих CMS трудно определить из-за постоянного появления новых решений и исчезновения старых. Однако можно с уверенностью сказать, что их много — от десятков до сотен различных систем управления контентом по всему миру.

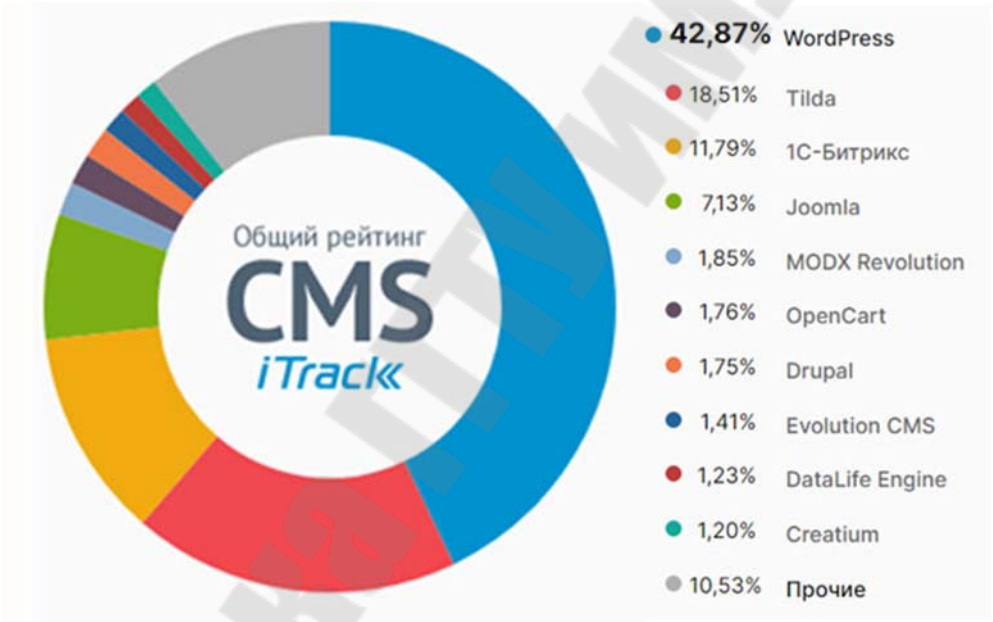

Рис. 6 – Рейтинг популярных CMS в мире по версии сервиса iTrack

Владельцы большинства сайтов выбирают CMS из-за ряда преимуществ:

- CMS позволяет разработать с нуля проект любой сложности;
- для работы с системой не обязательно обладать навыками программирования;
- CMS позволяет создать сайт намного быстрее;

Многие CMS можно использовать бесплатно: все необходимые инструменты движков доступны.

3.1. Классификация CMS

CMS можно классифицировать по разным критериям: типу лицензии, назначению, сложности и другим параметрам.

1. По типу лицензии: бесплатные и платные.

Бесплатные (Open Source - программное обеспечение с открытым исходным кодом доступно для свободного использования, изменения и распространения):

- WordPress – самая популярная CMS в мире, изначально создавалась для блогов, но сейчас поддерживает любые сайты.
- Joomla – мощная система с большими возможностями, но сложнее в освоении, чем WordPress.
- Drupal – гибкая и безопасная CMS, подходит для сложных проектов (госсайты, корпоративные порталы).
- OpenCart, WooCommerce (для WordPress) – специализированные CMS для интернет-магазинов.
- MODX – удобна для разработчиков, позволяет глубоко кастомизировать сайт.

Платные (Проприетарные - программное обеспечение (ПО), которое является собственностью конкретной компании или лица. Проприетарное ПО обычно имеет закрытый исходный код, который нельзя изменять, копировать или распространять его без разрешения владельца. Это программное обеспечение часто поставляется с лицензией, которая ограничивает его использование и модификацию):

1. Специализированные CMS:

- 1С-Битрикс – популярна в России, Беларуси и других стран СНГ, интегрируется с 1С, есть техподдержка. Пользователи устанавливают эту CMS на своем сервере или используют облачное решение. Это дает им полный контроль над сайтом, включая возможность настройки дизайна, функционала и интеграции с другими системами (например, CRM – Система управления взаимоотношениями с клиентами).
- Яндекс.Маркет, Wildberries, Ozon, AliExpress и другие. Эти платформы представляют собой системы управления контентом, которые специально разработаны для электронной коммерции. Продавцы регистрируются на этих платформах и создают свои магазины в рамках уже существующей экосистемы. Они не имеют полного кон-

троля над дизайном или функционалом сайта; вместо этого они используют предоставленные инструменты для управления своими товарами.

2. Конструкторы сайтов:

Software as a Service, SaaS – это модель предоставления программного обеспечения, при которой приложения размещаются в облаке и доступны пользователям через интернет, облачные CMS. Примеры популярных веб-конструкторов:

- Tilda – простой конструктор для лендингов и небольших сайтов.
- Wix – удобный визуальный редактор, но ограниченная гибкость.

2. По назначению (специализация CMS):

2.1. Универсальные CMS – подходят для разных типов сайтов: WordPress, Joomla, Drupal.

2.2. CMS для интернет-магазинов (E-commerce):

- OpenCart – простая и бесплатная система для создания интернет-магазинов.
- WooCommerce – плагин для WordPress, превращающий сайт в полноценный магазин.
- Яндекс.Маркет – маркетплейс и платформа для продаж через Яндекс, популярная в России.

2.3. CMS для корпоративных сайтов: Битрикс24 – российская платформа для бизнеса, включающая инструменты для электронной коммерции и CRM; Drupal (для сложных проектов).

2.4. CMS для блогов и медиа: WordPress (изначально создавался для блогов).

2.5. CMS для форумов: phpBB – классическая система для форумов и другие.

3.2. Обзор популярных CMS

Обзор популярных CMS, их преимущества и недостатки, какие задачи эти CMS решают.

CMS WordPress (wordpress.com/ru/) – самая распространенная платформа для запуска сайтов и публикации материалов в интернете. Система считается одной из самых безопасных и отказоустойчивых,

что сделало ее популярным решением для электронной коммерции, создания блогов, форумов, одностраничников, информационных сайтов.

Изначально WordPress использовали для создания личных блогов и информационных сайтов, однако сегодня его функционал расширился, появились сотни новых плагинов и расширений. Теперь на этом движке можно разработать лендинг, новостной или корпоративный ресурс, онлайн-магазин и любой другой проект.

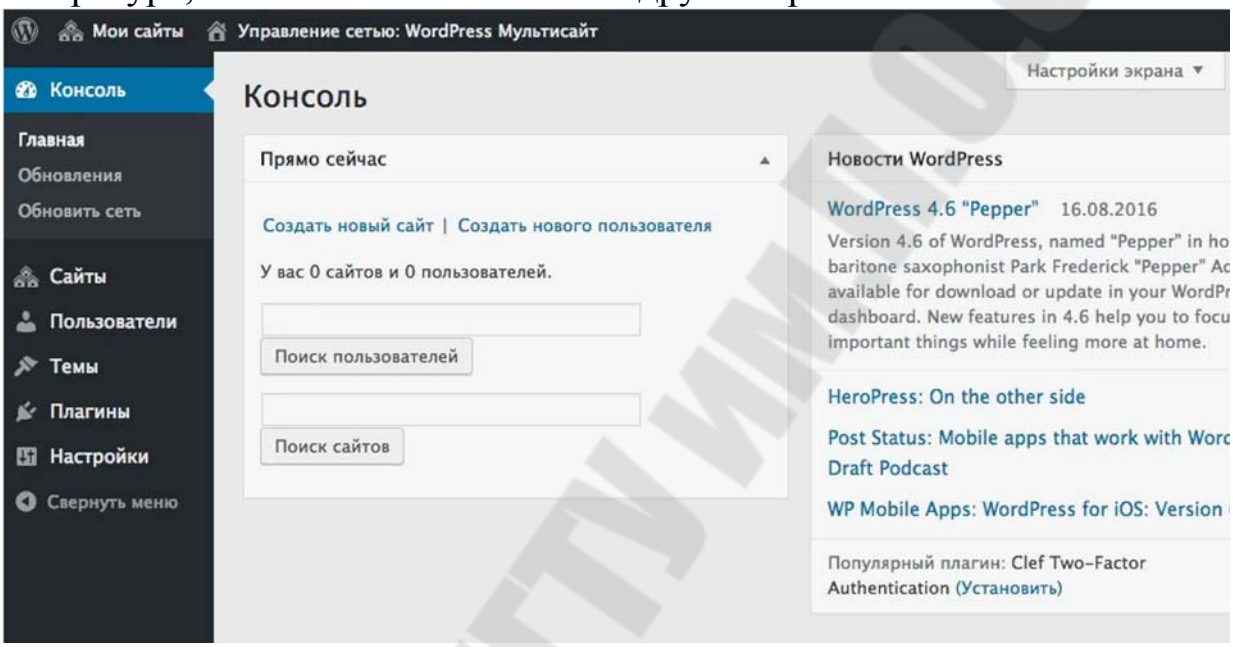

Рис. 7 – Панель управления WordPress

Преимущества CMS WordPress:

- общедоступный программный код;
- легкая установка – WordPress доступен для быстрой установки на большинстве популярных хостингов;
- интуитивно понятный интерфейс;
- большое количество бесплатных и платных тем и плагинов;
- адаптивность тем для мобильных устройств;
- интеграции с сторонними сервисами: Яндекс Метрика (metrica.yandex.com), Google Analytics (analytics.google.com), Unisender (unisender.com), социальные сети, Битрикс24 (bitrix24.by) и др.;
- многоязычность: встроенная поддержка более 70 языков;
- инструменты для поискового продвижения;
- настройка ролей пользователей с различными уровнями доступа;

- регулярные обновления для улучшения безопасности, производительности и функциональности;

- сообщество пользователей WordPress, где можно попросить помощи у других специалистов, найти плагины и варианты решения проблемы;

- создание различных типов сайтов – от блогов до интернет-магазинов и корпоративных порталов.

Недостатки WordPress:

- из-за популярности WordPress часто становится целью хакеров. Требуется регулярное обновление и установка плагинов для защиты;

- многие функции реализуются через плагины, что увеличивает риск возникновения конфликтов и усложняет поддержку;

- при использовании большого количества плагинов сайт работает медленнее, что влияет на скорость загрузки страницы;

- для правильной настройки и поддержки сайта нужны базовые знания в области веб-разработки и администрирования;

- для очень крупных или сложных проектов WordPress может быть менее подходящим по сравнению с специализированными системами (например, Битрикс).

Несмотря на это, WordPress считается одной из лучших CMS систем.

CMS Joomla! (joomla.org) – бесплатная система с гибкими настройками сайта. Благодаря широкому функционалу на Joomla! можно построить как сайт-визитку или блог, так и полноценный интернет-магазин.

Хотя разработчики добавили в Joomla! обширную библиотеку расширений и готовых решений, пользователям рекомендуется освоить языки HTML и CSS для более полного использования возможностей платформы.

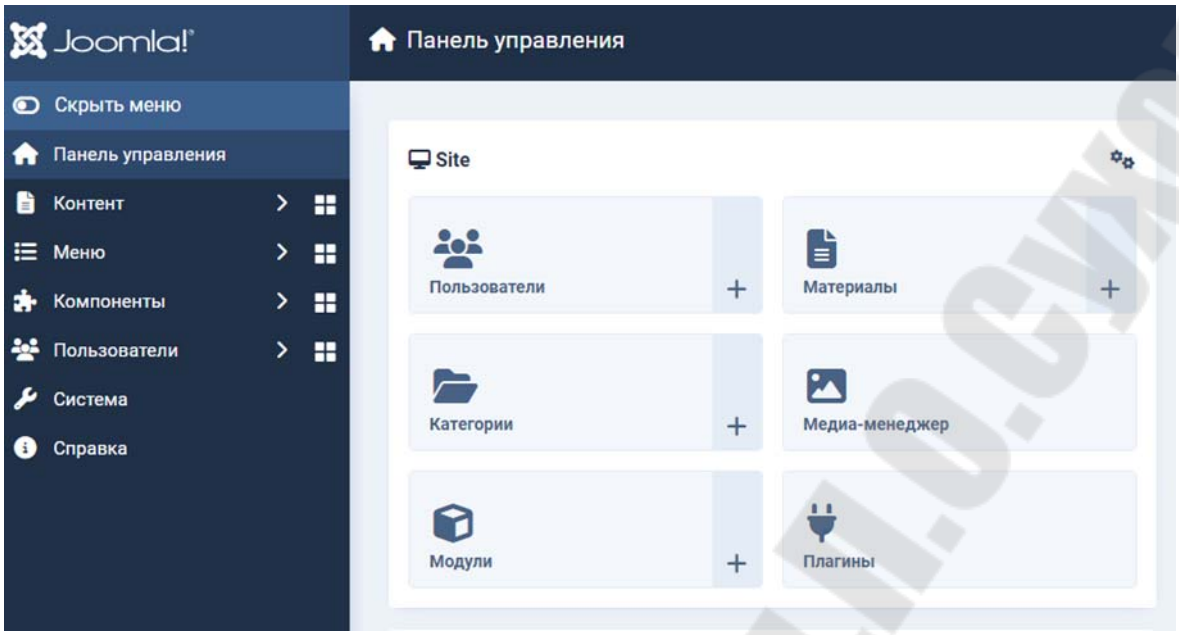

Рис. 8 – Панель управления Joomla!

Преимущества CMS Joomla!:

- общедоступный программный код;
- установка Joomla! в пару кликов на многих популярных хостингах;
- широкий спектр инструментов для SEO-оптимизации;
- собственная система отладки;
- регулярные обновления Joomla! для устранения уязвимостей и улучшения безопасности;
- адаптивность современных шаблонов Joomla! на мобильных устройствах;
- управление пользователями с различными уровнями доступа;
- платформа поддерживает интеграцию с различными сторонними сервисами через API и плагины: CRM-системы, платформы для email-маркетинга, социальные сети, платежные системы;
- большое количество свободных и коммерческих плагинов для расширения функционала;
- интуитивно понятный интерфейс для управления контентом;
- встроенные функции для создания многоязычных сайтов;
- у Joomla! есть активное сообщество разработчиков и пользователей, что обеспечивает доступ к обширной документации, форумам и ресурсам поддержки;
- платформа подходит для создания различных типов сайтов, включая блоги, корпоративные сайты, интернет-магазины и социальные сети.

Недостатки CMS Joomla!:

- начинающим освоение Joomla может быть непростым;
- количество доступных шаблонов для Joomla! ограниченное;
- для быстрой загрузки сайта необходима дополнительная seo-оптимизация;
- отсутствие техподдержки.

CMS Drupal (Drupal.org) – это мощная и гибкая система управления контентом (CMS), которая используется для создания и управления веб-сайтами различной сложности, от простых блогов до сложных корпоративных порталов и социальных сетей. Она известна своей модульной архитектурой, что позволяет пользователям настраивать функциональность сайта под свои нужды. Drupal поддерживает многоязычность, имеет развитую систему управления пользователями и предлагает множество инструментов для SEO и безопасности.

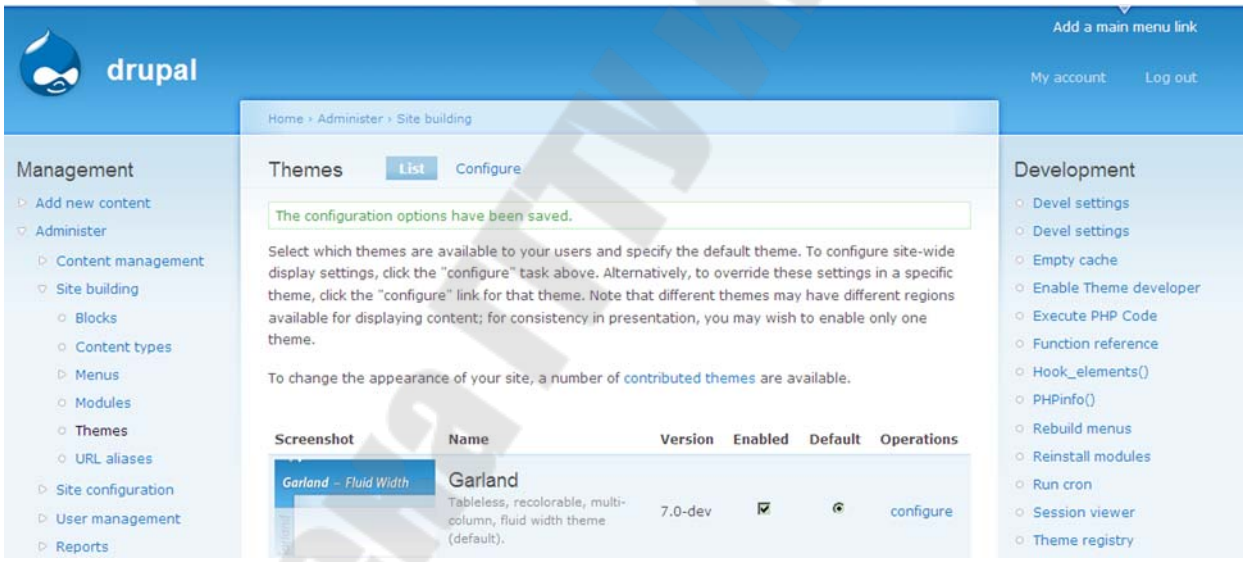

Рис. 9 – Административная панель Drupal

Преимущества Drupal:

- подходит для создания сайтов любой сложности — от небольших до крупных корпоративных порталов;
- использование модульной структуры — в Drupal используется система модулей, которые позволяют добавлять функциональность без изменения ядра системы. Есть тысячи готовых модулей для различных задач.

- считается одной из самых безопасных CMS, что делает его популярным для государственных и корпоративных сайтов;
- позволяет управлять несколькими сайтами из одной административной панели;
- гибкая система ролей и прав доступа - возможность точно настроить права для разных групп пользователей;
- поддержка сложных структур данных – свободно работает с большими объемами контента, сложными связями и пользовательскими типами данных;
- встроенная поддержка мультиязычных сайтов с возможностью легко управлять переводами;
- поддержка API и интеграций;
- активное сообщество разработчиков обеспечивает постоянное развитие, обновления и поддержку.

Недостатки Drupal:

- полное освоение всех возможностей требует времени и усилий;
- для работы с Drupal может потребоваться более мощный хостинг по сравнению с другими CMS;
- для настройки и разработки сложных решений часто требуется помощь квалифицированных разработчиков;
- наличие меньшего количества тем и плагинов;
- обновления иногда могут приводить к несовместимости с установленными модулями или изменению функционала.

Drupal – это мощный инструмент для создания сложных веб-сайтов, но его использование требует определенных знаний и навыков, что может быть препятствием для некоторых пользователей.

1С-Битрикс (1c-bitrix.ru) – коммерческая система, которая применяется для создания сайтов, работающих с большим количеством данных. Ее активно используют для управления торговыми площадками, т. к. она поддерживает разные единицы измерения, позволяет управлять огромными каталогами, имеет встроенные платежные системы.

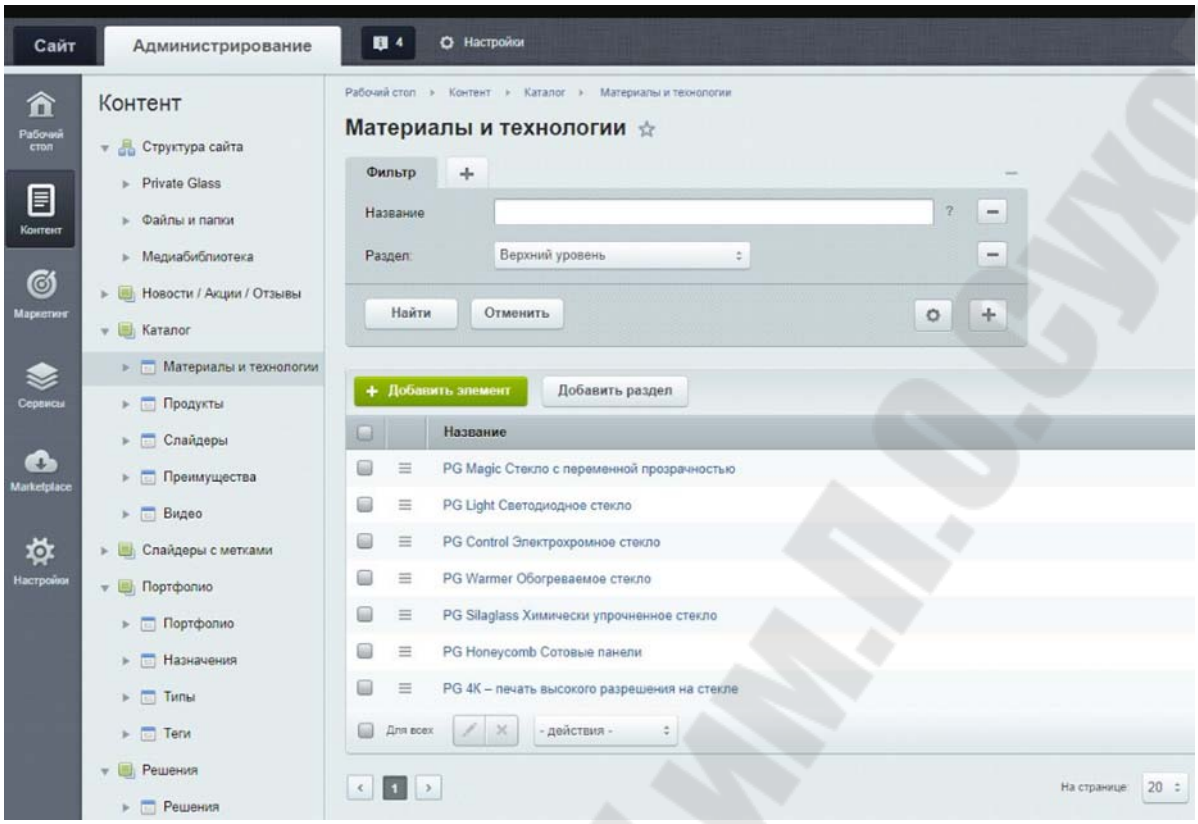

Рис. 10 – Административная панель Битрикс

Ключевые особенности Битрикс:

- большое количество встроенных инструментов для управления контентом, интернет-магазинами, email-рассылками, CRM, маркетингом и аналитикой;
- платформа обеспечивает высокий уровень безопасности, включая защиту от DDoS-атак и регулярные обновления для устранения уязвимостей;
- поддержка многоязычности;
- поддержка интеграций с 1С, платежными сервисами, службы доставки, кассы, торговые площадки (eBay, VK, Яндекс Маркет), «1С: Предприятие» и другие дополнения;
- Битрикс позволяет настраивать функционал под конкретные потребности бизнеса благодаря множеству модулей и компонентов;
- платформа подходит как для корпоративных сайтов, так и для интернет-магазинов, порталов и других типов ресурсов;
- у Битрикс есть официальная поддержка и обширная документация, что облегчает решение возникающих вопросов.

Недостатки:

- дорогая лицензия;
- для новичков интерфейс может показаться сложным, а полное освоение всех возможностей платформы требует времени;
- для корректной работы Битрикс может потребоваться более мощный хостинг по сравнению с другими CMS;
- для настройки и разработки сложных решений может потребоваться помощь квалифицированных разработчиков, что увеличивает затраты;
- из-за высокой стоимости и сложности освоения многие малые компании предпочитают более простые решения, такие как WordPress.

Несмотря на минусы, 1С-Битрикс остается в топе популярных систем, так как сайты на нем работают стабильно. Платформа предлагает широкий функционал для создания и управления интернет-магазинами и корпоративными сайтами, а также обеспечивает высокую степень безопасности и поддержку различных интеграций. Это делает 1С-Битрикс привлекательным выбором для бизнеса, который ценит надежность и масштабируемость своих веб-решений.

OpenCart (OpenCart.com) – специализированная система управления контентом (CMS), так как она разработана для создания и управления интернет-магазинами.

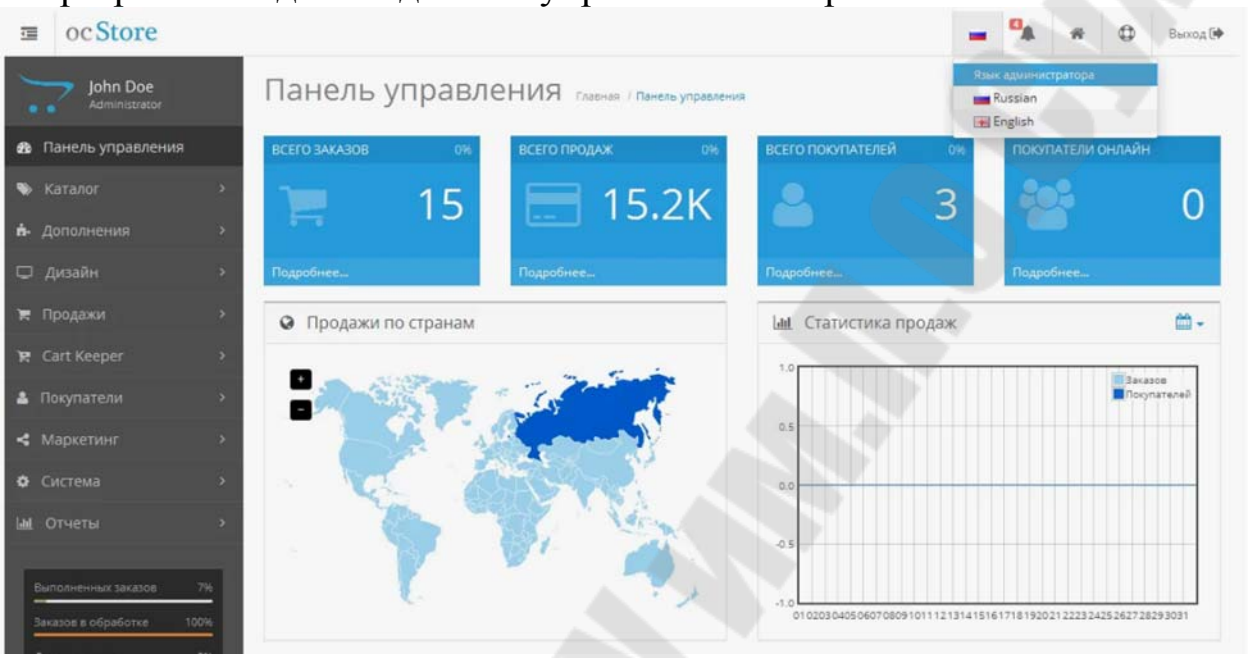

Рис. 11 – Админ-панель OpenCart

Она предлагает множество функций и инструментов, специально предназначенных для электронной коммерции, таких как управление товарами, обработка заказов, интеграция с платежными системами и многое другое. Это делает OpenCart более узкоспециализированным решением по сравнению с более универсальными CMS, такими как WordPress или Drupal.

В отличие от 1С-Битрикс, система управления контента не имеет технической поддержки. Если возникнут трудности при работе с CMS, то их придется решать самостоятельно.

Ключевые особенности OpenCart:

- В удобном кабинете администратора визуализирована активность пользователей: легко отслеживать количество и статус заказов, собирать аналитику и общаться с потенциальными покупателями через чат.

- Каталогизация с сложной структурой — администраторы могут детально настроить фильтрацию карточек и добавлять описания с характеристиками товаров.

- Настройка дизайна: OpenCart имеет огромное количество плагинов для уникализации ресурса.

Преимущества CMS OpenCart:

- поддержка многоязычности;
- высокая скорость загрузки;
- бесплатная лицензионная версия;
- инструкции по запуску ресурса для новичков.

Недостатки CMS OpenCart:

- нет службы поддержки;
- не все плагины доступны бесплатно;
- нет заранее настроенных параметров и конфигураций.

Несмотря на некоторые сложности, этот движок остается востребованным благодаря своей гибкости и мощным возможностям. Недостатки можно считать незначительными, если разобраться в системе самостоятельно.

OpenCart применяют только для создания сайтов интернет-магазинов. Реализация других проектов затруднительна.

Tilda (tilda.cc/ru) – это популярный веб-конструктор сайтов, который позволяет пользователям без специальных знаний в веб-разработке создавать красивые и функциональные сайты. Также идеально подходит для продвинутых пользователей и веб-дизайнеров. Он ориентирован на создание лендингов, портфолио, блогов и интернет-магазинов.

Особенности Tilda:

- Tilda использует систему блоков, что позволяет пользователям легко добавлять и настраивать различные элементы на странице (тексты, изображения, формы и т.д.) с помощью перетаскивания.
- Платформа предлагает множество готовых шаблонов, которые можно использовать в качестве основы для создания сайта. Шаблоны адаптированы под разные ниши и цели.
- Все сайты, созданные на Tilda, автоматически адаптируются под различные устройства (мобильные телефоны, планшеты и десктопы).

- Tilda поддерживает интеграцию с различными сервисами (CRM-системы, почтовые рассылки, аналитика), что позволяет расширять функциональность сайта.

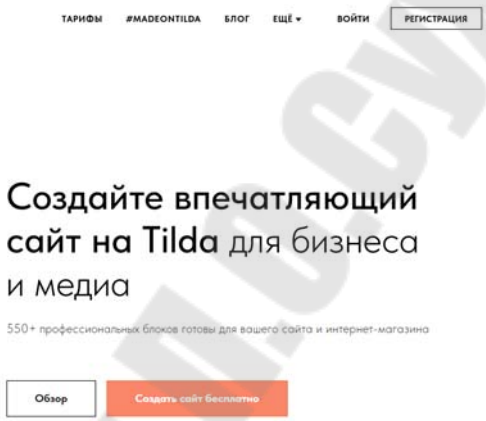

Рис. 12 – Веб-конструктор сайтов Тильда

- Платформа предоставляет инструменты для настройки SEO (метатеги, заголовки и т.д.), что помогает улучшить видимость сайта в поисковых системах.

- Для более опытных пользователей Tilda предлагает возможность добавления собственного HTML-кода и CSS для кастомизации дизайна.

Достоинства Tilda:

- Интуитивно понятный интерфейс позволяет быстро освоить платформу.

- Высокое качество шаблонов и блоков делает сайты привлекательными и современными.

- Возможность создания различных типов сайтов;

- Tilda активно поддерживается на русском языке, что делает её удобной для русскоязычных пользователей.

- Возможность использования бесплатного тарифа с ограниченными функциями для тестирования платформы.

Недостатки Tilda:

- Ограниченные возможности кастомизации: несмотря на то, что есть возможность добавления кода, пользователи могут столкнуться с ограничениями в дизайне по сравнению с полноценными CMS.

- Для доступа к более продвинутым функциям (например, подключение собственного домена или расширенные интеграции) необходимо перейти на платный тариф.

- Зависимость от платформы: все сайты хранятся на серверах Tilda, что может вызывать опасения у пользователей по контролю над своим контентом.

- В некоторых случаях сайты могут загружаться медленнее по сравнению с самописными решениями или другими CMS из-за особенностей платформы.

Конструктор Tilda используется для:

- создание лендингов для продвижения продуктов или услуг;

- разработки портфолио для фрилансеров и творческих специалистов;

- создания блогов и новостных сайтов;

- построения интернет-магазинов с возможностью интеграции с платежными системами;

- разработки одностраничных сайтов для мероприятий или акций.

Tilda — это отличный инструмент для тех, кто хочет быстро создать красивый сайт без глубоких знаний в веб-разработке, но при этом готов мириться с некоторыми ограничениями платформы.

Nethouse (nethouse.ru) — бесплатный веб-конструктор с возможностью докупить дополнительные опции. Он разработан в первую очередь для запуска онлайн-магазинов, но подходит для одностраничников и персональных блогов. Создатели движка добавили удобный редактор каталогов и готовые решения для интеграции систем эквайринга.

Функциональные особенности NetHouse:

- Библиотека функций для онлайн-магазинов – многоуровневый каталог, поисковая строка, оформление покупки за 1 клик, рекомендации товаров.

- Домен и почта – разработчики дарят домен и корпоративный почтовый ящик.

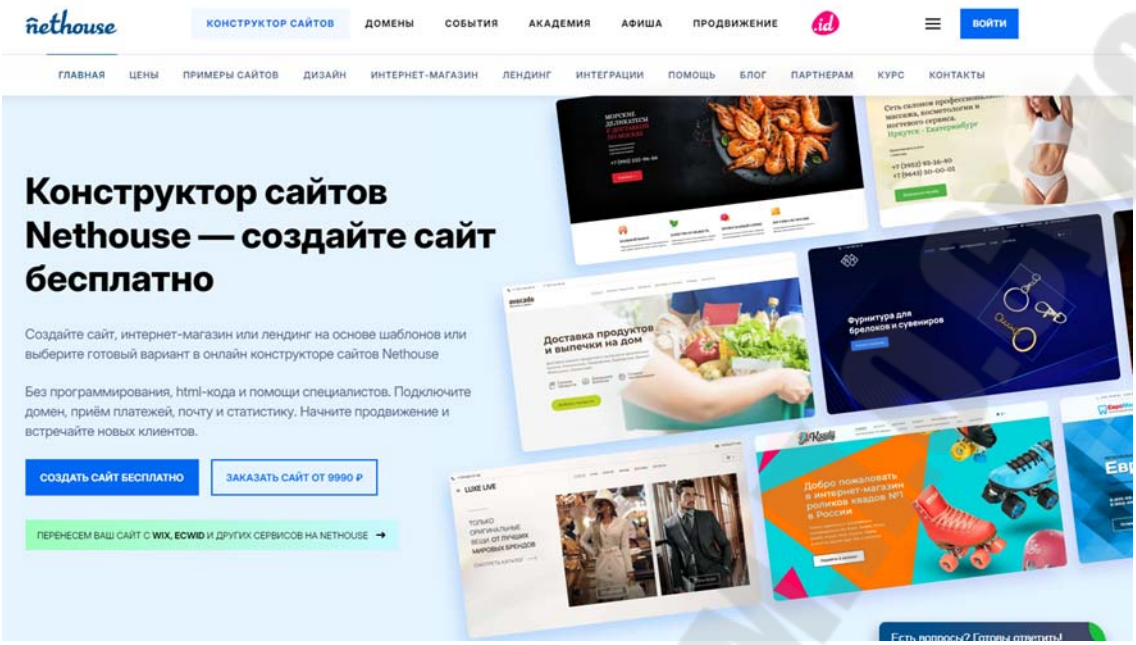

Рис. 13 – Веб-конструктор сайтов Nethouse

- Автоматическая выгрузка карточек на Яндекс.Маркет – это поможет сэкономить время на рутинные действия и быстро увеличить охват.

- Для работы не требуются навыки программирования и постоянный контроль контента. Она имеет достаточно функций для автоматизации размещения товарных карточек и других материалов.

Достоинства движка:

- интеграция с Гугл Аналитикой и Яндекс Метрикой;
- простой интерфейс;
- встроенный сервис для SEO-оптимизации.

Недостатки:

- минимальный функционал для одностраничников, персональных блогов;
- слабый редактор для проработки дизайна.

Nethouse подходит индивидульным предпринимателям, малому бизнесу и фрилансерам для запуска небольших онлайн-магазинов, страниц, рекламирующих мероприятия.

Wix (ru.wix.com) - популярный конструктор сайтов, который нельзя назвать полноценной CMS.

Платформа имеет мощный редактор, который позволяет детально проработать дизайн страницы. А при желании пользователь может создать собственный вариант оформления, уникальные элементы, а также корпоративный логотип.

Особенности Wix:

- Интуитивно понятный интерфейс: Wix использует метод перетаскивания, что позволяет легко добавлять и перемещать элементы на странице без необходимости в программировании.

- Платформа предлагает большое количество готовых шаблонов, которые можно использовать для создания различных типов сайтов — от личных блогов до интернет-магазинов.

- Все шаблоны автоматически адаптируются под мобильные устройства, а также есть возможность редактирования мобильной версии сайта.

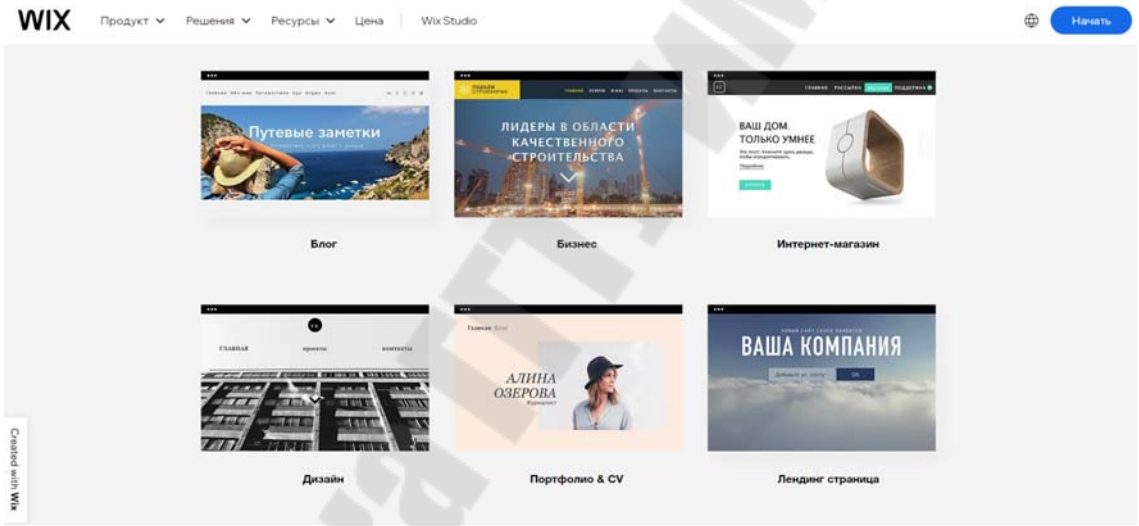

Рис. 14 – Конструктор Wix

- Wix предоставляет базовые инструменты для оптимизации сайта под поисковые системы, включая возможность редактирования метатегов и настройки URL.

- В Wix множеством приложений для расширения функциональности сайта (формы обратной связи, чаты, аналитика и т.д.).

- Все сайты на Wix размещаются на серверах компании, что упрощает процесс управления хостингом.

- Возможность генерации готового сайта с помощью ИИ-конструктора за несколько минут.

Возможности веб-конструктора Wix:

- встроенный магазин веб-приложений и автоматическая установка плагинов;
- универсальность платформы;
- открытая платформа разработки;
- конкурентная стоимость подписки;
- онлайн-учебник от разработчиков.

Недостатки:

- невозможны миграция и масштабирование;
- мало опций для разработки нишевых площадок.

Все системы управления контентом (CMS) активно развиваются и адаптируются к современным требованиям пользователей. На сегодняшний день многие из них обладают схожими функциональными возможностями, что позволяет пользователям выбирать платформу, наиболее соответствующую их потребностям.

Разработчики CMS постоянно работают над улучшением своих продуктов, устраняя недостатки и добавляя новые функции. Это включает в себя оптимизацию производительности, улучшение интерфейса, расширение возможностей интеграции с другими сервисами и повышение уровня безопасности.

Таким образом, пользователи могут рассчитывать на то, что выбранная ими CMS будет не только актуальной, но и способной эффективно решать задачи, стоящие перед их проектами. Конкуренция между платформами способствует постоянному совершенствованию и внедрению инноваций, что в конечном итоге приносит пользу всем пользователям.

3.3. Критерии выбора CMS

От выбора платформы для разработки зависит как сама возможность реализации необходимой функциональности, так и трудоёмкость дальнейшего развития веб-проекта.

Для создания сайта, как правило, выбирается одна из платформ: CMS, фреймворк или конструктор. У каждого из типов платформ есть свои плюсы и минусы.

Разработка на конструкторе

SaaS‑платформы еще часто называют «конструкторами сайтов». Пример популярных конструкторов для простых сайтов это Тильда (tilda.cc/ru) и Wix (Wix.com); «Деал бай» (Deal.by) и «Пульс Цен» (Pulscen.by) – для электронной коммерции. Основное преимущество этого варианта – хороший уровень качества за очень небольшие деньги.

Но SaaS решения недостаточно гибкие: они допускают только минимальную настройку бизнес-логики и имеют ограничения по настройке визуального оформления. Если требования к оформлению или к функциональности проекта достаточно высокие, то «конструктор сайтов» по характеристикам уже может не подойти, так как реализация некоторых функциональных возможностей на этих платформах просто невозможна технически.

Разработка на CMS

CMS имеют некую аналогию с конструкторами сайтов (SaaS) – оба вида платформ являются набором готовых решений и функций для создания веб-ресурсов. При этом системы управления проектами требуют определенных знаний и навыков веб-разработки.

Наиболее правильный подход разработки на CMS, если проект достаточно типовой. То есть в CMS уже есть все нужные модули, а те процессы, которые встроены в CMS, почти полностью соответствуют ожиданиям разработчика и заказчика.

Много небольших и средних сайтов строится именно на CMS‑платформах, так как это наиболее целесообразный с экономической точки зрения подход: требования к таким сайтам с технической точки зрения невысокие, а встроенные в CMS модули обычно удовлетворяют выдвигаемым к ним бизнес-требованиям.

Разработка на фреймворке

Фреймворк – это программный продукт, который служит основой для сайта. Он включает в себя базовые компоненты для разработки веб-приложений, но обычно не содержит в себе гото-

вых программных модулей для реализации конкретных бизнес-процессов.

Разработка на фреймворке – наиболее правильный подход, если видение проекта не особо укладывается в возможности и процессы присутствующих на рынке CMS. То есть тогда, когда в случае использования CMS её придётся существенно дорабатывать.

Разработчики, при создании сайта на фреймворке, создают не только публичную часть сайта, но и проектируют базу данных, разрабатывают алгоритмы для модулей системы, а также создают административный интерфейс для управления проектом. Необходимость серьёзных затрат на программирование делает разработку более дорогой, но и результат получается более индивидуальным.

Этот вид платформ используют почти все крупные веб-проекты: на CMS построена лишь очень малая часть действительно серьёзных проектов, проекты с серьёзной посещаемостью, а также подавляющее большинство веб-приложений и веб-сервисов основаны на фреймворках. Коробочных решений для уникальных бизнес-процессов просто не существует, а использование не очень подходящих CMS в качестве основы для последующей кастомизации очень сильно усложняет разработку.

Скорость работы и устойчивость к нагрузкам у созданных на фреймворках решений в разы, а иногда и на порядок выше (в сравнении с CMS), сопровождаемость лучше, а стоимость владения при этом не сильно отличается.

Разработка с нуля

Разработка «с нуля» может быть правильной только в том случае, если создание проекта – это ключевая задача компании, времени под этот проект выделено крайне много, бюджет серьёзный и в команде проекта есть очень компетентные разработчики. Создание хороших проектов «с нуля» занимает много времени, однако этот метод позволяет разрабатывать действительно впечатляющие решения.

Критерии выбора

Выбор стоит делать на основании экономической составляющей разработки.

Функциональность, которая встроена в CMS, на фреймворке реализовывать дороже и дольше. Но разработка сложных функций на CMS или переписывание базовых процессов CMS стоят дороже и занимает больше времени, чем та же работа выполненная сразу на фреймворке.

Добиться от сложного проекта на CMS высокой скорости работы стоит дороже, чем сделать это на фреймворке. При высоких требованиях к устойчивости к нагрузкам, производительности или к отказоустойчивости стоит выбрать решения на фреймворках или необходимо закладывать стоимость работ по оптимизации CMS в бюджет проекта.

Запуск первой пилотной (неполной) версии проекта на CMS всегда быстрее, чем запуск аналогичной версии на фреймворке.

Если выбор по разработке проекта падает на конкретную CMS, очевидно, что следует выбирать ту CMS, которая будет наиболее экономичной и сможет эффективно решить поставленные задачи.

3.4. Популярные CMS в Беларуси

Беларусь в целом вписывается в мировые тенденции. Наиболее популярные CMS среди белорусских пользователей — это WordPress, Joomla, Drupal, 1С-Битрикс и другие специализированные системы.

В тройку самых используемых систем в Беларуси, входит коммерческая CMS — «1С-Битрикс». На ней в доменах .BY и .БЕЛ работает более 7% от общего числа используемых CMS.

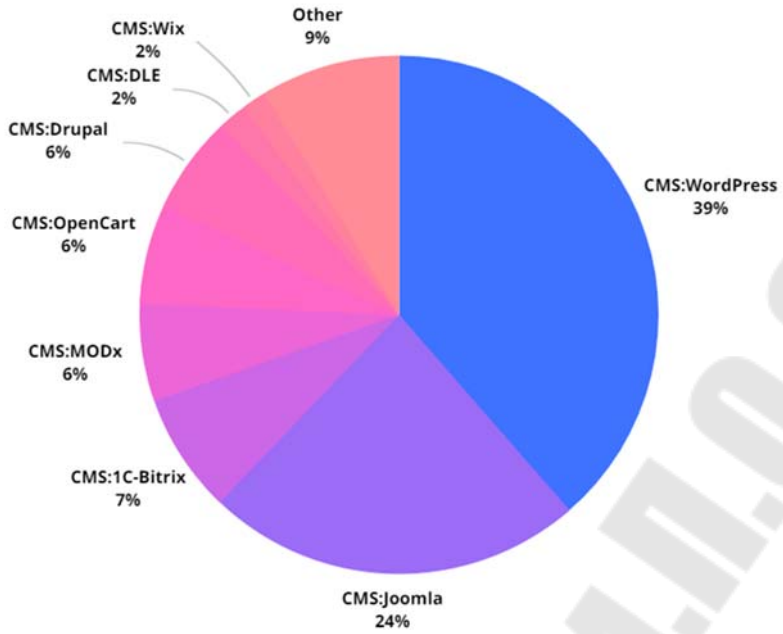

Рис. 15 – Популярные CMS в Беларуси

Приблизительный рейтинг CMS в Беларуси:

1) WordPress – 39%,
2) Joomla – 24%,
3) 1C-Bitrix – 7%,
4) MODx – 6%, OpenCart – 6%, Drupal – 6%,
5) DLE – 2%, Wix – 2%, Tilda, Site.pro и др.

В зависимости от типа сайтов, технологии варьируются. Сайты предприятий и организаций, чаще всего создаются на базе CMS: WordPress, Joomla или Drupal. ВУЗы, также могут использовать эти платформы в том числе и систему управления образовательными электронными курсами CMS Moodle.

Многие школы и садики применяют специализированную систему CMS Web.Perspective.

Корпоративные сайты, как правило, разрабатываются на 1С-Битрикс или Drupal, а интернет-магазины могут использовать как Битрикс, так и самописные специализированные решения.

Индивидуальные предприниматели предпочитают использовать одностраничники и сайты-визитки с использованием CMS WordPress, CMS Joomla, веб-конструкторов.

К веб-конструкторам в РБ можно отнести:

- Сервисы, которые предоставляют создание сайта на конструкторах вида Tilda – Хостер hoster.by (предоставляет конструктор сайтов Tilda), ИИ конструктор сайтов – site.pro и т.д.

- CMS «Web.Perspective» – детские садики, школы.

- CMS 1С-Битрикс (включает в себя функционал веб-конструктора).

Для освоения веб-конструкторов прилагается инструкция в видео и текстовом формате, а также техническая поддержка хостинга, на котором находится сайт.

Различные виды специализированных веб-сайтов, где пользователи создают свои магазины (или страницы, например для объявлений на Kufar.by) в рамках уже существующей экосистемы:

- торговые площадки (маркетплейсы) «Деал бай» (deal.by), «Пульс Цен» (pulscen.by) и т.д.;

- доски объявлений, городские порталы, электронные газеты.

Системы управления контентом (CMS) играют ключевую роль в современной веб-разработке, предоставляя пользователям возможность легко создавать, редактировать и управлять содержимым своих сайтов. Популярные CMS, такие как WordPress, Joomla, Drupal и 1С-Битрикс, предлагают разнообразные функции и возможности, которые могут удовлетворить потребности как новичков, так и опытных разработчиков.

WordPress остается самой популярной CMS в мире благодаря своей простоте использования, обширной библиотеке плагинов и тем, а также активному сообществу. Это идеальный выбор для блогеров, малых бизнесов и крупных корпоративных сайтов.

Joomla! предлагает более сложные возможности для управления контентом и подходит для создания многофункциональных сайтов. Она может быть отличным выбором для пользователей, которым нужны более продвинутые функции без необходимости глубокого программирования.

Drupal выделяется своей гибкостью и безопасностью, что делает его предпочтительным выбором для крупных проектов с высокими требованиями к безопасности и функциональности. Однако его сложность может быть препятствием для новичков.

1С-Битрикс — это мощная платформа для создания корпоративных сайтов и интернет-магазинов. Она предлагает широкий спектр инструментов для управления контентом, интеграции с CRM-

системами и аналитики. Битрикс особенно популярен благодаря своей локализации и поддержке специфических бизнес-процессов. Однако его стоимость может быть выше по сравнению с другими CMS, что стоит учитывать при выборе.

Кроме того, конструкторы сайтов, такие как Tilda и Wix, предоставляют альтернативный подход к созданию веб-сайтов. Они позволяют пользователям без технических знаний быстро создавать привлекательные сайты с помощью визуальных редакторов.

В конечном счете, выбор CMS зависит от конкретных потребностей проекта, уровня технической подготовки пользователя и бюджета. Каждая из платформ имеет свои достоинства и недостатки, поэтому важно тщательно оценить их перед принятием решения. Независимо от выбора, современные CMS значительно упрощают процесс создания и управления веб-сайтами, делая его доступным для широкой аудитории.

4. ОСОБЕННОСТИ ПРИМЕНЕНИЯ ФРЕЙМВОРКОВ ПРИ РАЗРАБОТКЕ ВЕБ-САЙТОВ

4.1.Фреймворки

Фреймворк (framework) переводится с английского языка как «каркас, структура». Под этим термином понимается комплекс IT-инструментов, которые позволяют сделать разработку и поддержку сложных веб-проектов с высокой нагрузкой более простыми.

За счет того, что фреймворк включает в себя базовые программные модули, его можно охарактеризовать как промежуточный вариант по гибкости и уровню сложности между созданием кода от самого начала и применением CMS. Если в работе необходимо использовать специфичные компоненты, они создаются программистами.

Определенная часть фреймворка неизменна, она сохраняет свое постоянство независимо от конфигурации. Другая часть является переменной, она включает в себя модули и компоненты, необходимые для кастомизации окончательной системы.

Веб-фреймворк — это платформа для создания сайтов и веб-приложений, облегчающая разработку и объединение разных компонентов большого программного проекта в единую систему. За счёт широких возможностей в реализациии бизнес-логики и высокой производительности фреймворки особенно хорошо подходят для создания сложных сайтов, бизнес-приложений и веб-сервисов.

Типы фреймворков

Бэкенд-фреймворки

К этим фреймворкам относятся разработки, функции которых осуществляются на сервере. Как правило, в зоне их ответственности лежат отдельные части ресурса, от которых зависит работоспособность всего сайта или приложения.

Пример бэкенд-фреймворков: Django — Python (Django — это фреймворк для разработки бэкенда (серверной части веб-приложений), он основан на языке программирования Python), Laravel — PHP, JS — JavaScript, Ruby on Rails — Ruby.

Функциональность данных фреймворков недостаточно разнообразна.

Бэкэнд-фреймворки отлично подходят для создания простой интернет-страницы, формы ее могут быть различны. Также платформы прекрасно справляются с задачей формирования выходных данных и гарантируют безопасность ресурса, если возникнет хакерская атака.

Фронтенд-фреймворки

Функционал этого вида фреймворков завязан на браузере. В их зоне ответственности лежит визуальная составляющая ресурса. Они не принимают участия в организации внутренней логики работы приложения или сайта.

С помощью таких платформ осуществляется улучшение интерфейса пользователя, создаются анимации и лендинги. Эту группу составляют фреймворки: Angular, Vue.js, Svelte, React (эта платформа по своей сути приравнивается к библиотеке, но функционал настолько широк, что ее часто сравнивают с иными веб-фреймворками).

Все эти фреймворки базируются на JavaScript.

Фулстек-фреймворки несут ответственность не только за серверные, но и за клиентские функции. Эту группу составляют:
Meteor, который базируется на JavaScript, что позволяет разработчику использовать один и тот же код. Фреймворки Next.js и Nuxt, работа которых базируется на React.js и Vue.js. Их, как правило, используют разработчики с опытом.

Фреймворки и микрофреймворки

Фреймворки по своему объему классифицируются на:

1. Микрофреймворки. Они призваны решать узкие задачи. Чтобы создать проект на их базе, необходимо привлечь дополнительные приложения, позволяющие разработать небольшой ресурс. Обычно специалисты совмещаются в работе микрофреймворки и большие фреймворки.

2. Большие фреймворки используются, когда нужно задействовать большое количество функций для решения всех поставленных задач.

Популярные фреймворки

1. Ruby on Rails. Ресурс создан на базе Ruby, с его помощью разрабатываются веб-приложения.

2. Laravel и Symfony. Фреймворки на PHP — это платформы, посредством которых создаются веб-приложения.

3. Svelte — платформа, созданная на языке JavaScript для создания веб-приложений.

4. Django — этот фреймворк построен на Python. С его помощью разрабатываются веб-приложения.

5. Bootstrap — фреймворк, базирующийся на HTML, CSS, JavaScript, посредством его создаются сайты и приложения. Этот ресурс нередко относят к классу библиотеки.

6. Angular — создан с использованием TypeScript, контролируется в системе Google. Чаще всего его используют крупные корпорации.

7. Vue.js — фреймворк на JavaScript, который применяют для создания интерфейсов.

8. JQuery — фреймворк, который активно используют разработчики на протяжении более чем 15 лет для создания небольших проектов. Он не подходит для крупных ресурсов, так как количество дополнительных элементов JavaScipt в нем минимально.

9. React.js — базируется на JavaScript. Этим фреймворком пользуются на протяжении 10 лет с целью спроектировать клиентский интерфейс.

10. Express.js — написан на Node.js. Его применяют для создания сайтов и приложений.

11. Flutter — платформа, которая позволяет работать на смартфоне на базе Андроид.

12. Flask – написан на Python, включает в себя 7 кодов.

13. Spring. Back-end работает на Java. Особенно активно применяется на таких сайтах, как Wix (веб-конструктор).

14. Codelgniter. Один из простейших фреймворков, который был создан в 2006 году.

15. YiiFramework на базе языка PHP имеет широкий функционал. Чтобы написать код на этой платформе уйдет всего несколько минут.

16. NW.JS. Эта платформа отличается универсальностью. Фреймворк может работать вместе с платформами Linux, Windows и MacOS, а также создавать коды на JavaScript, HTML и CSS.

17. ASP.net. Этот фреймворк разработан на базе Майкрософт и поддерживается Виндовс. Он занимается созданием точных кодов HTML и разработкой относительно сложных и полноценных онлайн-ресурсов.

18. Phalcon. Этот фреймворк был создан в 2012 году. Он работает на языках PHP, Zephir и C, что позволяет легко и быстро создать код.

19. SemanticUI – фреймворк, пользующийся особенной популярностью у начинающих специалистов. Платформа позволяет создать код на самом простом языке HTML. Этим и объясняется столь высокая востребованность.

Архитектура фреймворка

Архитектура — метод, используемый для создания кода. Нередко разработчики, которым нужно создать сайт или приложение на фреймворке, применяют архитектуру «Model — View — Controller», MVC, или «модель — представление — контроллер».

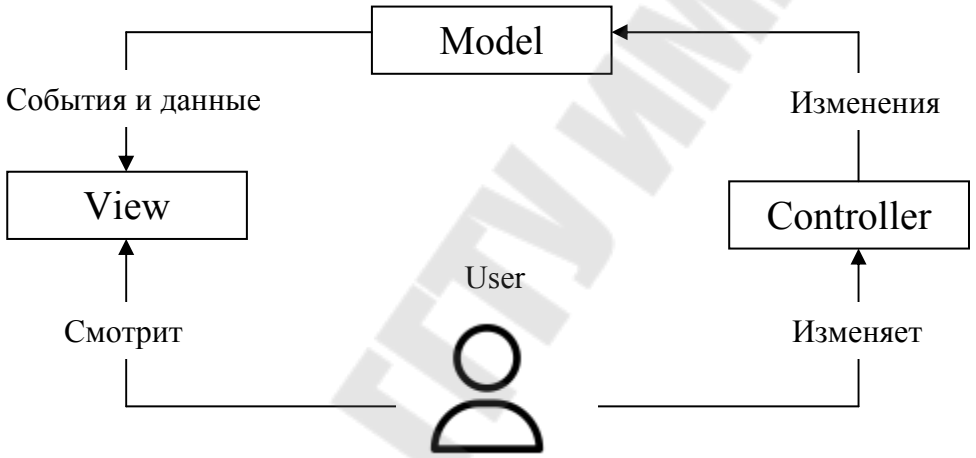

Рис.16 – Архитектура «Model — View — Controller»

Аббревиатура MVC имеет следующую расшифровку:

- Модель (Model) — компонент, в зоне ответственности которого лежат данные будущего сайта или приложения, а также определение структуры ресурса. К примеру, при создании приложения для планирования времени, кодом модели являются календарь и перечень поставленных задач, которые нужно выполнить.
- Представление (View) — компонент, в зоне ответственности которого лежит обеспечение взаимодействия пользователя с ресурсом. От кода зависит визуальное оформление и панель управления — то есть то, куда нужно нажать, чтобы выполнить задачу, где осуществляется набор текста, какие элементы можно поменять местами.

- Контроллер (Controller) — компонент, который несет ответственность за взаимодействие модели и представления. Если говорить простым языком, контролер – это центр управления ресурсом. От кода зависит, какая реакция будет наблюдаться со стороны проекта на различные действия пользователя.

Во время работы с фреймворком происходит взаимодействие разработчика с отдельным компонентом. Главное удобство таких платформ заключается в том, что специалист может вносить изменения лишь там, где это требуется.

Особенности веб-фреймворков

Особенности применения фреймворков веб-разработчиками:

1. Веб-кэширование – эта функция дает возможность сохранять различные файлы, не перегружая при этом сервер. Его можно применять в разных системах, но нужно соблюдать определенные условия. Такой фреймворк работает на базе сервера. Пример – кэш-контент в поисковой системе Google.

2. Скаффолдинг — это веб-фреймворки, которые осуществляют автоматическую генерацию типичных частей ресурса. Кроме того, платформы могут сделать это даже со всей структурой проекта, что особенно важно для разработчиков-новичков. Фреймворк в значительной мере сокращает время, потраченное на работу, и делает кодовую базу стандартизированной.

3. Система веб-шаблонов — это комплекс, состоящий из разнообразных методик и ПО, которые могут быть использованы с целью создать и развернуть веб-страницу. Обработка шаблонов происходит посредством шаблонизаторов, которые применяются во фреймворке как инструмент, который отвечает за публикацию ресурса.

4. Безопасность — термин, обозначающий комплекс средств, которые позволяют идентифицировать, разрешить или отклонить доступ к функционалу веб-фреймворка. Также эти средства дают возможность определить профили, использующие ресурс, во избежание кликджекинга (тип атаки на веб-приложения, механизм обмана платформы с целью получить конфиденциальные данные о пользователях и доступ к их устройствам).

Фреймворки позволяют разрабатывать не только сайты, но и приложения, а также блоги, форумы, системы управления контентом (CMS).

Преимущества фреймворков

Преимущества фреймворков – высокая кастомизуемость (изменение программного обеспечения согласно требованиям конкретного пользователя (заказчика)), хорошая сопровождаемость и расширяемость в сложных проектах, высокая производительность, надёжность и информационная безопасность.

Разработка на фреймворке позволяет добиться высокой сопровождаемости проекта в отличие от самописных решений и от нетиповых проектов на коробочных CMS.

Возможна и относительно проста реализация любых бизнес-процессов, а не только тех, которые изначально заложены в систему. Также проекты на базе фреймворков легко масштабируемы и модернизируемы.

Решения на фреймворках работают значительно быстрее и выдерживают большую нагрузку, чем любые CMS и многие самописные системы. Именно поэтому почти все популярные и посещаемые сайты и интернет-магазины работают не на коробочных CMS, а на фреймворках. По уровню информационной безопасности приложения на фреймворках значительно превосходят самописные системы и CMS. CMS «из коробки» тоже весьма безопасны, но, как правило, готовые проекты на фреймворках всё же превосходят проекты на CMS благодаря более частому соблюдению заложенной методологии и использованию более продвинутых инженерных практик, а также за счёт относительно более высокой компетенции у разработчиков на фреймворках.

Недостатки фреймворков

Недостатки фреймворков – длительная разработка, требуется качественная аналитика и высокая экспертиза.

Сроки разработки типовой функциональности на фреймворках больше, чем при использовании CMS. Фреймворки содержат только базовые компоненты бизнес-логики уровня приложения, поэтому многие функции реализовываются индивидуально.

Для разработки на фреймворке требуется понимание бизнес-процессов, которые требуется реализовать. Например, если в CMS уже есть некий предустановленный процесс обработки заказов в ин-

тернет-магазине, то фреймворки такого не предоставляют, а значит приходится продумывать и реализовывать этот процесс собственными силами.

Рекомендации по выбору фреймворка для веб-проекта

При выборе конкретной платформы для разработки часто возникают сложности. Бывает непросто принять взвешенное решение, в этом случае будет полезен следующий алгоритм выбора:

- Оценка функциональных и нефункциональных требований проекта, таких как производительность, безопасность, масштабируемость и поддержка.
- Ознакомление с наиболее распространенными фреймворками в данной области (например, Laravel, Symfony, Django для Python и т.д.) и их особенностями.
- Проверка активности сообщества вокруг фреймворка: наличие документации, форумов, обучающих материалов и готовых решений. Чем больше сообщество, тем легче найти помощь.
- Изучение производительности фреймворка в контексте требований. Некоторые фреймворки могут быть более оптимизированными для определенных задач.
- Оценка уровня знаний команды в различных фреймворках. Если команда уже знакома с определенным инструментом, это может значительно ускорить процесс разработки.
- Убедиться, что выбранный фреймворк поддерживает масштабирование приложения в будущем. Это особенно важно для проектов с потенциальным ростом нагрузки.
- Изучение механизмов безопасности, предлагаемых фреймворком. Необходимо убедиться, что он предоставляет средства для защиты от распространенных уязвимостей (например, SQL-инъекций, XSS).
- Ознакомление с лицензией фреймворка и проверка ее соответствия требованиям (особенно если проект коммерческий).
- Узнать о возможностях тестирования и отладки в рамках фреймворка. Наличие встроенных инструментов для тестирования может упростить процесс разработки.

Следуя этим рекомендациям, можно выбрать подходящий фреймворк для веб-проекта, который будет соответствовать всем тре-

бованиям и обеспечит успешную разработку и поддержку приложения в будущем.

Экономическая эффективность и целесообразность использования фреймворков

С точки зрения бизнеса разработка на фреймворке почти всегда экономически эффективнее и качественнее по результату, нежели написание проекта на чистом языке программирования без использования каких-либо платформ. Разработка без использования платформы может быть правильным решением только в двух случаях — либо проект совсем простой, не требующий дальнейшего развития и сопровождения, либо очень нагруженный и требует низкоуровневой оптимизации (например, веб-сервисы с десятками или сотнями тысяч обращений в секунду). Во всех других случаях разработка на программной платформе будет быстрее и качественнее.

Если сравнивать фреймворки с другими классами платформ — SaaS (веб-конструктор), CMS — то фреймворки значительно эффективнее использовать в проектах со сложной бизнес-логикой и с высокими требованиями к скорости работы, надёжности и безопасности. Для простых и типовых проектов стоимость разработки на фреймворке будет выше и займёт она больше времени, нежели запуск простого проекта на SaaS или CMS.

4.2. Фрейворки в популярных CMS

Многие современные системы управления контентом (CMS) основаны на различных фреймворках или предлагают возможность интеграции сторонних фреймворков, что делает разработку сайтов более эффективной и структурированной.

Пример популярных CMS поддерживающих фреймворки, для расширения функциональности и настройки: CMS WordPress, CMS Joomla!, CMS Drupal, CMS Magento и другие.

Ключевые моменты применения фреймворков в системах CMS:

1. В некоторых CMS встроен собственный фреймворк для разработки и настройки функциональности системы.
2. CMS могут интегрироваться с другим фреймворком, включая загрузку файлов CSS и использование классов для построения сетки и стилизации компонентов.

3. CMS поддерживают сторонние шаблоны и плагины, которые используют фреймворки.

4. В CMS встроен собственный набор инструментов и принципов разработки, которые можно рассматривать как «внутренний фреймворк» или «микрофреймворк».

Примеры использования фреймворков в популярных CMS:

CMS Joomla! Joomla! использует собственный фреймворк, который часто просто называют – Joomla Framework. Этот фреймворк был разработан специально для системы управления контентом Joomla и предоставляет разработчикам набор инструментов и библиотек для создания расширений и шаблонов.

Основные характеристики фреймворка Joomla!:

1. Структура: фреймворк основан на архитектуре MVC (Model-View-Controller), что обеспечивает четкое разделение логики приложения, пользовательского интерфейса и данных.

2. Расширяемость: позволяет разработчикам создавать собственные компоненты, модули, плагины и шаблоны, которые могут быть легко интегрированы в систему.

3. Безопасность: включает в себя множество функций для обеспечения безопасности, таких как система аутентификации и механизмы защита от атак.

4. Интернационализация: поддерживает многоязычность, что позволяет создавать сайты для разных языков.

5. Комьюнити: имеет активное сообщество разработчиков и пользователей, что облегчает процесс поиска поддержки и получения информации о лучших практиках.

Фреймворк Joomla! значительно упрощает создание мощных веб-приложений, делает разработку более удобной и организованной.

Интеграция CMS Joomla с фреймворками: Интеграция Joomla! с различными фреймворками, например, Bootstrap, позволяет расширить функциональность сайта, обеспечить более гибкую настройку и повысить его производительность.

Bootstrap — это популярный фреймворк CSS, который позволяет создавать адаптивные, отзывчивые интерфейсы. Он включает в себя готовые шаблоны и компоненты, такие как кнопки, формы, таблицы и сетки.

Joomla! поддерживает интеграцию с Bootstrap, начиная с версии 3.x. Это позволяет разработчикам использовать сеточную систему Bootstrap для создания макетов страниц. Использование встроенной сетки на основе Bootstrap в Joomla! является применением фреймворка. Это облегчает создание адаптивных и современных сайтов с минимальными усилиями. Внедрение Bootstrap в шаблоны Joomla! предоставляет доступ к широкому спектру функций и возможностей для творчества у разработчиков и дизайнеров.

Применение библиотеки Bootstrap в Joomla!:

1. Использование шаблонов включающих поддержку с Bootstrap. Пример популярных шаблонов, активно применяющие фреймворк Bootstrap: Cassiopeia (установочный шаблон Joomla! 4, 5), Helix Ultimate, YOOtheme и др.

2. Встраивание Bootstrap-файлов CSS и JavaScript в шаблон Joomla.

3. Использование классов Bootstrap при создании контента – встраивание HTML-кода на страницу с классами Bootstrap для создания сетки.

4. Стилизация компонентов: использование встроенных классов Bootstrap для таких компонентов: как кнопки, формы и навигация.

Помимо Bootstrap существует множество других аналогичных фреймворков, которые можно интегрировать с шаблонами в CMS, таких как Joomla!. Вот некоторые из них:

- Foundation – мощный фреймворк, который предлагает гибкие сетки и множество компонентов для создания адаптивных веб-сайтов.
- Bulma – CSS-фреймворк на основе Flexbox, который предлагает простую и современную структуру для создания адаптивных интерфейсов. Bulma легко интегрируется в шаблоны Joomla! через подключение соответствующих CSS-файлов.
- Tailwind CSS – утилитарный CSS-фреймворк, который позволяет создавать кастомные дизайны с помощью классов. Можно подключить Tailwind к шаблону сайта и использовать его классы для стилизации элементов.
- Materialize CSS – фреймворк, основанный на принципах Material Design от Google. Он предоставляет готовые компоненты и стили, которые можно легко интегрировать в шаблон.
- UIKit – легкий и модульный фреймворк для разработки интерфейсов, который предлагает множество компонентов и стилей. Его

можно подключить к шаблону Joomla! аналогично другим фреймворкам.

Интеграция с сторонними расширениями для Joomla, использующих фреймворки:

1. Галереи: Phoca Gallery – использует собственный фреймворк Phoca.

2. Формы:

- RSForm! Pro – использует собственный фреймворк RS Framework.
- ChronoForms –использует свой фреймворк ChronoForms.

3. Карты:

- Joomla Maps – использует API Google Maps или Leaflet.
- Google Maps Plugin – использует API Google Maps.

4. Другие:

- Akeeba Backup (резервное копирование) – использует собственную библиотеку Akeeba Engine.
- JCE Editor (визуальный редактор) – использует API Joomla и собственные разработки.

Важно отметить, что многие расширения разрабатываются с использованием собственных фреймворков или API Joomla, и не всегда полагаются на широко известные PHP-фреймворки, такие как Laravel или Symfony.

CMS WordPress

WordPress не является фреймворком в классическом понимании. Он предоставляет API (набор функций, классов и хуков) для разработки тем и плагинов, но не навязывает строгую архитектуру или набор компонентов, как это делают фреймворки. API WordPress можно рассматривать как основу для разработки, но не как полноценный фреймворк.

Многие плагины WordPress разрабатываются с использованием API WordPress, JavaScript-библиотек (например, Leaflet для карт) или собственных архитектур, и не полагаются на классические PHP-фреймворки. Визуальные редакторы часто имеют собственные фреймворки для построения интерфейса.

WordPress поддерживает интеграцию с Bootstrap как через сторонние темы (шаблоны), так и через создание собственного шаблона с использованием данного фреймворка. Разработчики могут использовать классы Bootstrap для создания адаптивных макетов, форм и интерфейсов.

Для упрощения процесса разработки на WordPress разработано много готовых тем, основанных на Bootstrap: Astra, GeneratePress, OceanWP и др.

Помимо Bootstrap WordPress поддерживает интеграцию с другими фреймворками.

Пример популярных фреймворков и библиотек, которые можно использовать:

- Foundation – фреймворк от ZURB, обеспечивающий гибкость и быстроту в разработке адаптивных интерфейсов. Может быть интегрирован аналогично Bootstrap, с использованием нужных стилей и скриптов.
- Bulma – CSS-фреймворк на основе Flexbox, легкий и простой в использовании. Интеграция через подключение CSS-файлов и использование классов в HTML.
- Tailwind CSS – CSS-фреймворк, позволяющий создавать компоненты с нуля. Встраивается в шаблон с помощью подключения CDN (Content Delivery Network - Сеть доставки контента).
- Semantic UI – фреймворк, основанный на естественном языке, который упрощает создание разметки. Интеграция происходит через подключение необходимых стилей и скриптов.

Основные подходы к разработке плагинов WordPress

Использование полноценных PHP-фреймворков, таких как Laravel или Symfony, встречается реже в плагинах WordPress. Это связано с тем, что WordPress уже предоставляет API, а добавление еще одного фреймворка может увеличить размер плагина и усложнить его поддержку. Однако фреймворки могут использоваться для решения конкретных задач внутри плагина. Например, для работы с базой данных, маршрутизации запросов или создания REST API.

Разработка плагинов WordPress обычно основывается на API WordPress, JavaScript-библиотеках и собственных архитектурах. Использование полноценных PHP-фреймворков встречается реже и оправдано в случаях, когда требуется решить специфическую задачу, которую фреймворк упрощает.

«Собственная архитектура и API» подразумевает, что разработчики создали свой набор инструментов и правил для построения плагина, что можно считать упрощенным аналогом фреймворка в рамках конкретного плагина.

Пример плагинов WordPress, использующие фреймворки или библиотеки:

- Плагины для работы с базами данных – могут использовать Eloquent (фреймворк ORM из Laravel) через сторонние библиотеки для упрощения работы с базой данных WordPress.
- Плагины, использующие React/Vue.js – многие современные плагины для создания сложных интерфейсов в админ-панели WordPress используют JavaScript-фреймворки, такие как React или Vue.js.
- Плагины, построенные на основе Composer – некоторые плагины используют Composer для управления зависимостями и могут включать в себя компоненты из различных PHP-фреймворков.

Важно отметить, что чаще всего плагины использует собственный набор инструментов и принципов разработки, которые можно рассматривать как «внутренний фреймворк» или «микрофреймворк».

И Joomla!, и WordPress предлагают разработчикам мощные инструменты через встроенные фреймворки, а также возможность интеграции сторонних фреймворков. Это позволяет разработчикам выбирать наиболее подходящие инструменты для создания адаптивных и функциональных веб-сайтов, расширяя возможности кастомизации и управления контентом.

4.3. Инструменты и программы для разработки веб-страниц на фреймворках

Для разработки страниц с использованием фреймворков обычно используют интегрированные среды разработки (IDE) или редакторы кода, которые поддерживают работу с фреймворками и позволяют писать более структурированный и удобный код. Популярные программы для этого — Visual Studio Code, WebStorm, Sublime Text и другие.

Примеры программ для работы с фреймворками:

- Visual Studio Code, WebStorm — очень популярный редактор с множеством расширений для работы с различными фреймворками (React, Angular, Vue и др.)
- Sublime Text — легкий редактор с поддержкой плагинов.

Пример создания веб-страниц с использованием фреймворков:

Bootstrap — это популярный CSS-фреймворк, разработанный компанией Twitter. Он предоставляет готовые стили, компоненты и сеточную систему для быстрого создания красивых и адаптивных веб-страниц, позволяет сделать современный дизайн сайта без необходимости писать всё с нуля, используя только стандартные HTML и CSS. Фреймворк Bootstrap:

- обеспечивает набор готовых стилей для кнопок, форм, карточек, навигации и других элементов интерфейса;
- позволяет легко создавать адаптивные макеты, которые хорошо выглядят на любых устройствах;
- включает JavaScript-компоненты для интерактивных элементов (модальные окна, карусели и др.).

Пример простого HTML-кода страницы с использованием популярного фреймворка **Bootstrap** (в <head> подключается CSS-файл Bootstrap через CDN). Этот код создает простую страницу с использованием фреймворка Bootstrap.

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width,
initial-scale=1.0" />
  <title>Пример страницы на Bootstrap</title>
  <!-- Подключение Bootstrap CSS -->
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/
bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <div class="container text-center mt-5">
    <h1>Привет, мир!</h1>
    <p>Это пример страницы на Bootstrap.</p>
    <button class="btn btn-primary">Кнопка</button>
  </div>

  <!-- Подключение Bootstrap JS -->
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bo
otstrap.bundle.min.js"></script>
</body>
</html>
```

Привет, мир!

Это пример страницы на Bootstrap.

Кнопка

Рис. 17 – HTML-кода страницы с использованием популярного фреймворка Bootstrap

Пример создания простой html-страницы на фреймворке **React:**

```
import React from 'react';

function App() {
  return (
    <div style={{ textAlign: 'center', marginTop: '50px' }}>
      <h1>Привет, мир!</h1>
      <p>Это страница на React.</p>
      <button style={{ padding: '10px 20px', fontSize: '16px' }}>Кнопка</button>
    </div>
  );
}

export default App;
```

Рис. 18 – Html-страница на React

Это компонент App, который возвращает JSX — расширение синтаксиса JavaScript для описания интерфейса.

Для запуска этого кода нужно создать проект с помощью Create React App или другого сборщика.

Create React App — это шаблон или скелет проекта, который автоматически создаёт структуру папок и файлов приложения. Позволяет сразу начать писать React-код, не тратя время на настройку окружения.

Пример создания простой html-страницы на фреймворке **Vue 3**

```
<template>
  <div style="text-align: center; margin-top: 50px;">
    <h1>Привет, мир!</h1>
    <p>Это страница на Vue.</p>
    <button style="padding: 10px 20px; font-size: 16px;">Кнопка</button>
  </div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>
/* Можно добавить стили здесь */
</style>
```

Рис. 19 – Html-страница на Vue 3

Пример компонента Vue, который содержит шаблон (<template>), скрипт (<script>) и стили (<style>). Для запуска этого кода нужно использовать специальную среду разработки.

Приведённые примеры на React и Vue создают простую веб-страницу, которая в конечном итоге рендерится в HTML:

- В React и Vue пишутся компоненты с помощью специального синтаксиса (JSX для React или шаблонный синтаксис для Vue).
- Эти компоненты затем компилируются или транслируются в обычный HTML, CSS и JavaScript, которые браузер может понять и отобразить.
- В результате пользователь видит страницу с заголовком, текстом и кнопкой — всё это создаётся через фреймворк, но отображается как стандартный HTML.

Иными словами, эти фреймворки позволяют создавать динамические и сложные интерфейсы, но в конечном итоге всё отображается как обычная HTML-страница в браузере.

5. ОСОБЕННОСТИ НАПИСАНИЯ ПРОГРАММНОГО КОДА

В современном программировании особое значение приобретает качество написанного кода. Чистый и хорошо структурированный код способствует более эффективной работе команды и облегчает дальнейшее развитие проекта.

Особенности написания программного кода – это широкая тема, включающая в себя множество аспектов, связанных с созданием качественного программного обеспечения.

Написания программного кода могут включать в себя:

- чистый и читабельный код с правильным форматированием и комментариями;
- архитектура программного обеспечения и паттерны проектирования: как организованы модули и компоненты;
- производительность: оптимизация кода для скорости и эффективности, выбор правильных алгоритмов и структур данных;
- безопасность: предотвращение уязвимостей, защита от атак;
- масштабируемость: написание кода, который можно легко расширять и адаптировать к изменяющимся требованиям;
- поддержка: написание кода, который легко поддерживать и отлаживать;
- тестирование: написание кода, который можно легко протестировать;
- специфические требования проекта: учет требований заказчика, стандартов компании и других факторов;
- выбор языка программирования и фреймворков: обоснование использования того или иного языка/фреймворка для решения конкретной задачи.

Умение писать красивый код, будь то программный код или код на HTML — один из технических навыков разработчиков.

Чистый код — код, который легко читается, понятен любому разработчику и прост в поддержке. Это помогает создавать более качественные, поддерживаемые и масштабируемые приложения.

Принципы чистого кода

Принципы чистого кода — это набор правил и рекомендаций, направленных на создание понятного, легко поддерживаемого и мас-

штабируемого программного обеспечения. Следование этим принципам помогает разработчикам писать код, который легко читается, тестируется и модифицируется, что в конечном итоге повышает качество продукта и ускоряет процесс разработки.

Основные правила для написания чистого кода:

- Читаемость: использование понятных имен переменных и функций.
- Простота: избегание излишней вложенности.
- Эффективность: оптимизация кода.
- Отсутствие избыточности: уборка лишних элементов.
- Стремление к простоте: стремление к простым решениям.
- Последовательность: соблюдение стиля кода в команде.
- Тестируемость: код должен быть легко тестируемым.
- Документированность: хорошая документация для понимания логики кода.
- Модульность: разделение кода на независимые модули или функции.
- Ясность намерений: названия должны отражать назначение элементов.
- Обработка ошибок: код должен корректно обрабатывать возможные ошибки.
- Принципы SOLID: следование принципам объектно-ориентированного проектирования, которые помогают разработчикам создавать более гибкий, поддерживаемый и понятный код.

Рекомендации.

1. Пояснения к коду:

- Необходимо писать короткие пояснения к сложным участкам кода.
- Следует обновлять устаревшие комментарии.
- Комментарии должны присутствовать только у рабочего кода.
- Не рекомендуется добавлять избыточные комментарии.
- В комментариях необходимо объяснять цели, которые лежали в основе решения.

2. Надежность кода:

- Обработка ошибок и тестирование кода в различных ситуациях.

3. Организация кода:

- Продумывание структуры программы в начале работы, а не в конце.

- Упрощение логики: необходимо выносить уровни в отдельные функции.

- Минимизирование уровней вложенности: каждый уровень вложенности усложняет код.

- Избегание дублирования: необходимо не повторять один и тот же код, а выносить его в отдельные функции.

- Избегание двусмысленных названий: необходимо использовать сокращения, принятые в программировании.

- Форматирование кода: необходимо использовать отступы и пробелы, чтобы код выглядел аккуратно и легко читался.

Пример фрагмента чистого кода HTML с комментариями:

```
<!DOCTYPE html>
<!-- Объявление типа документа, указывающее браузеру, что это HTML5 -->
<html lang="ru">
<!-- Корень документа, язык установлен как русский -->
<head>
  <meta charset="UTF-8" />
  <!-- Установка кодировки символов -->
  <title>Пример страницы</title>
  <!-- Заголовок страницы, отображается в вкладке браузера -->
</head>
<body>
  <!-- Основная часть документа, видимая пользователю -->

  <h1>Заголовок страницы</h1>
  <!-- Заголовок первого уровня -->

  <p>Это пример простого параграфа текста.</p>
  <!-- Параграф текста -->

  <!-- Комментарий: ниже – кнопка для взаимодействия -->
  <button>Нажми меня</button>
</body>
</html>
```

Рис. 20 – Пример фрагмента чистого кода HTML с комментариями

- Перепроверка кода: необходимо обязательно перечитывать написанное и проводить редактирование. Это поможет оптимизировать код и позволит другим разработчикам понимать его без лишних пояснений.

- Соблюдение договоренностей: необходимо договориться о том, как вы будете писать код, если работа происходит в команде (какие отступы использовать, как называть переменные и т.д.).

4. Производительность кода:

- Оптимизирование медленных участков и придерживание стандартов кодирования.

- Анализ решений и ошибок: необходимо разбирать, что получилось хорошо, а что стоит улучшить.

- Использование полезных инструментов для настройки среды программирования.

5. Документация:

- чтение документации: это поможет писать код правильно и эффективно;

- создание документации для созданного кода, чтобы другие разработчики могли быстро разобраться с проектом.

6. Следование стандартам: необходимо придерживаться общепринятых стандартов кодирования для выбранного языка программирования, тогда код будет более совместимым с другими проектами и облегчит работу в команде.

7. Использование систем контроля версий: необходимо хранить свой код в системах контроля версий (например, Git git-scm.com). Это позволит отслеживать изменения, возвращаться к предыдущим версиям и работать в команде без конфликтов.

Инструменты для написания чистого кода

Инструменты для написания чистого кода помогают автоматизировать анализ, форматирование, оптимизацию и тестирование кода для повышения его качества и читаемости.

Популярные инструменты:

- IDE (Интегрированные среды разработки): помогают писать, редактировать и отлаживать код (например, PyCharm – (jetbrains.com/pycharm/), Visual Studio Code – code.visualstudio.com/).

- Линтеры: анализируют код, находят ошибки, стилистические проблемы и потенциальные баги (например, Pylint – pylint.pycodestyle.info/, ESLint – eslint.org).

- Форматеры кода: автоматически приводят код к единому стилю и делают его более читаемым (например, Black – black.readthedocs.io, Prettier – prettier.io).

- Инструменты для автоматического тестирования: подтверждают, что код работает правильно и не ломается, если что-то изменить (например, Pytest – pytest.org, Jest – jestjs.io).

- Плагины и скрипты: для разных сред программирования существуют инструменты, которые помогут привести текст программы к единому стандарту, сделать его легко читаемым и понятным.

Написание чистого кода — важный навык для любого веб-программиста. Это помогает создавать более качественные, поддерживаемые и масштабируемые приложения.

6. НАПИСАНИЕ АВТОРСКИХ РЕШЕНИЙ

Разработка авторских решений для систем управления контентом (CMS) позволяет улучшать функциональность сайта и улучшать пользовательский опыт. Это может включать в себя создание дополнительных плагинов, тем (шаблонов), компонентов или модулей. Они обычно разрабатываются сторонними разработчиками и могут быть установлены на различные платформы для добавления новых функций или оптимизации существующих.

Разработчики дополнительных расширений для различных систем управления контентом (CMS) лицензированы и представлены на официальных сайтах соответствующих CMS. Среди доступных расширений можно найти большое количество некоммерческих решений, что позволяет пользователям легко расширять функциональность своих сайтов без дополнительных затрат.

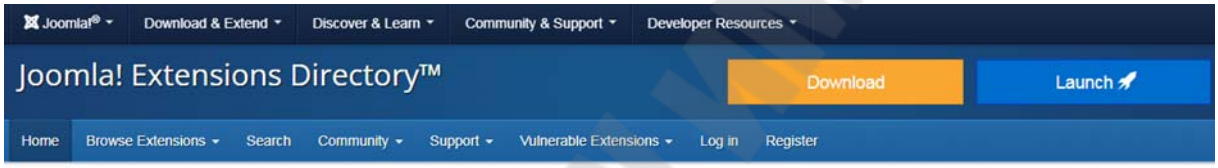

Рис. 21 – Раздел расширений «Extensions Directory» для CMS Joomla на сайте-разработчика

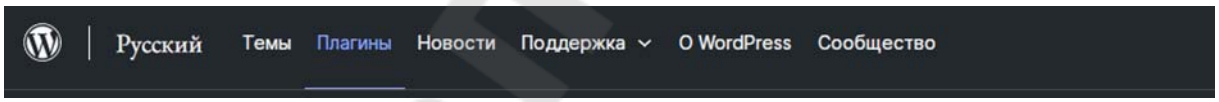

Рис. 22 – Раздел расширений для CMS Wordpress на сайте-разработчика

Преимущества расширений:

1. Улучшение функциональности: плагины могут добавить новые возможности и функции на сайт, делая его более удобным для пользователей и привлекательным для поисковых систем.

2. Экономия времени и ресурсов: вместо того, чтобы самостоятельно разрабатывать новые функции, можно использовать готовые плагины, которые сэкономят время и силы.

3. Легкость обновления: обновление плагина обычно проще, чем обновление кода сайта, что делает процесс поддержки более простым и безопасным.

Виды плагинов:

Расширения для CMS представлены в самом широком ассортименте:

- формы обратной связи и регистрации;
- галереи изображений и видео;
- социальные кнопки и виджеты;
- системы управления контентом;
- seo-плагины;
- модули для интеграции с другими сервисами и др.

6.1. Примеры авторских решений для CMS WordPress и Joomla

Популярные плагины для WordPress:

- Contact Form 7: этот плагин позволяет легко создавать и настраивать формы обратной связи для сайта.
- WooCommerce: мощное расширение для создания интернет-магазина на базе WordPress, предлагающее множество функций и интеграций с платежными системами.
- Akeeba Backup – компонент для резервного копирования сайтов, позволяющий гибко настраивать создание бэкапов.
- Elementor – плагин WordPress, который расширяет возможности стандартного редактора.
- Yoast SEO, Clearfy Pro, W3 Total Cache, WP Super Cache – популярные SEO-плагины для WordPress.
- Robin image optimizer —плагин для оптимизации изображений.
- WPForms – плагин для создания контактных форм: оплаты, опросов, информационных бюллетеней и регистрации.

Популярные плагины для Joomla:

- Akeeba Backup – компонент для резервного копирования сайтов, позволяющий гибко настраивать создание бэкапов.
- JCE (Joomla Content Editor) – продвинутый визуальный редактор контента.
- Helix Ultimate, Helix3 – популярные фреймворки для шаблонов от разработчика JoomShaper.
- SP Page Builder – мощный конструктор посадочных страниц (лендингов) на CMS.
- K2 (сайта CCK (Content Construction Kit) – компонент для вставки статей в любое место сайта.

- Sourcerer – системный плагин для Joomla, позволяющий размещать PHP и любой другой код, в том числе стили CSS и JavaScript, непосредственно в контент сайта.

- JCH Optimize – компонент для Joomla, ускоряющий загрузку веб-страниц сайта без изменений в содержимом исходных файлов CMS.

- JComments – мощная и простая в использовании бесплатная система комментариев для Joomla, использующая технологию AJAX.

- JoomShopping: компонент Joomla для реализации функционала полноценного интернет-магазина.

- Xmap – генератор карты сайта для Joomla (sitemap.xml).

- Cache Cleaner – добавляет в админку кнопку для быстрой очистки кэша.

- Phoca Gallery, Sigplus, Simple Image Gallery – популярные компонент для создания галерей изображений в Joomla.

6.2. Начало работы с плагином

Перед тем, как начать работу с плагинами необходимо проверить:

- Совместимы ли плагины с версией CMS – авторы всегда указывают версию системы, с которой плагин будет работать гарантированно корректно. При этом утилиты должны обновляться вместе с CMS. Есть плагины, которым не требуется совместимость с последними версиями CMS, но это исключения.

- Рейтинг плагина и количество скачиваний. Рейтинг формируют пользователи, которые уже установили этот инструмент себе на сайт и могут поделиться мнением. Лучше не скачивать плагины с низким рейтингом.

- Отзывы. Перед установкой стоит почитать отзывы пользователей о преимуществах и недостатках плагина.

Пример рейтинга для расширения Phoca Gallery для Joomla! с сайта https://extensions.joomla.org/.

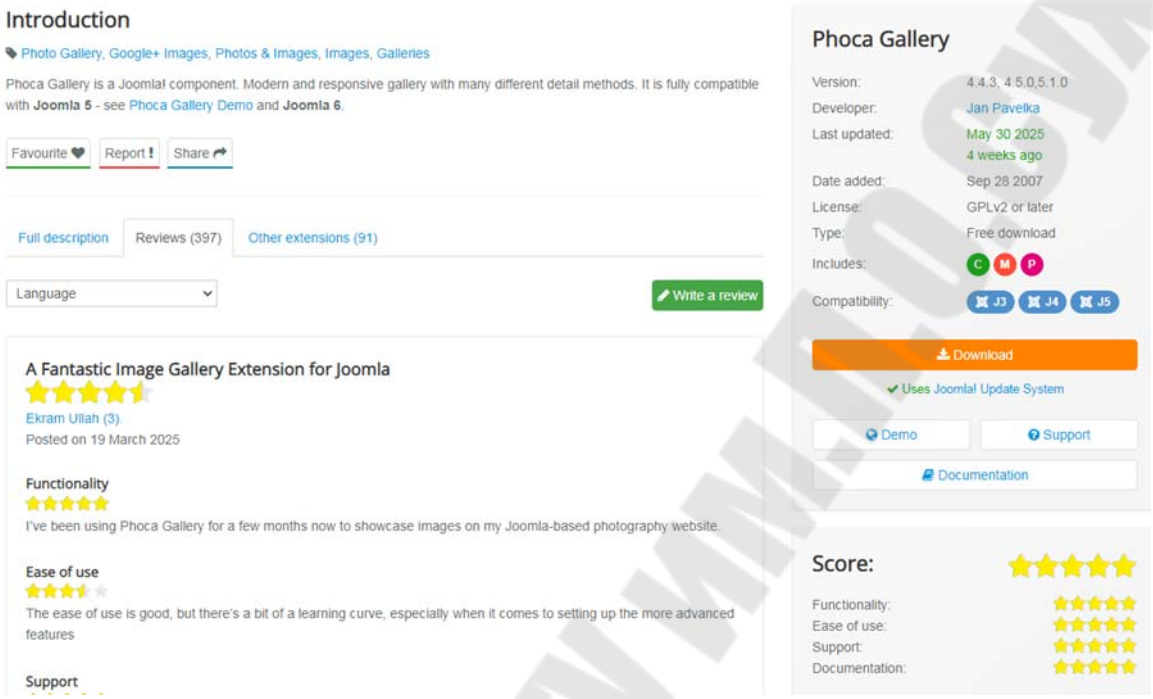

Рис. 23 – Рейтинг расширений

Перед установкой плагина необходимо сделать резервную копию сайта. И если установка пойдёт не по плану, сайт можно будет откатить до прежней версии.

6.3. Коммерческие и некоммерческие расширения

Для CMS существуют коммерческие и некоммерческие расширения. Они отличаются по нескольким ключевым аспектам:

1. Лицензирование:

Некоммерческие расширения предоставляются под лицензиями, которые позволяют использовать, изменять и распространять их бесплатно. Пользователи могут свободно устанавливать и использовать такие расширения на своих сайтах.

Коммерческие расширения требуют оплаты за их использование. Лицензии на коммерческие продукты часто ограничивают возможность их распространения или модификации. Пользователи должны приобрести лицензию для доступа к функциональности.

2. Поддержка и обновления:

Для некоммерческих расширений поддержка может быть ограниченной, так как разработчики часто работают над проектами на добровольной основе. Обновления могут выходить реже, и пользователи могут быть вынуждены полагаться на форумы сообщества для получения помощи.

Коммерческие расширения обычно сопровождаются официальной технической поддержкой от разработчиков и регулярными обновлениями. Пользователи могут рассчитывать на более быструю реакцию на проблемы и наличие документации.

3. Функциональность:

Некоммерческие расширения могут предлагать базовую функциональность, которая удовлетворяет потребности многих пользователей. Однако иногда они могут быть менее мощными или функциональными по сравнению с коммерческими аналогами.

Коммерческие расширения часто предлагают более сложные функции, дополнительные возможности и интеграции, которые могут быть полезны для бизнеса или профессиональных пользователей.

4. Качество и тестирование:

Качество некоммерческих расширений может варьироваться, поскольку они могут разрабатываться разными людьми с разным уровнем опыта. Тестирование может быть неполным.

Коммерческие расширения обычно проходят более строгие тестирования и проверки качества, так как разработчики заинтересованы в поддержании хорошей репутации и удовлетворении клиентов.

5. Целевая аудитория:

Некоммерческие расширения часто ориентированы на пользователей, которые ищут бесплатные решения или только начинают свой путь в веб-разработке.

Коммерческие расширения обычно нацелены на более серьезных пользователей и организации, которые готовы инвестировать в качественные инструменты для улучшения своих сайтов.

Эти различия могут помочь пользователям выбрать подходящие расширения в зависимости от их потребностей и бюджета. Создание авторских решений для CMS — это отличный способ улучшить функциональность сайтов и удовлетворить потребности пользователей. Это требует внимания к деталям, опыта разработки и понимания специфики платформы.

7. СОЗДАНИЕ САЙТА НА CMS

Для создания и предварительного тестирования сайтов созданы специальные пакеты программ – локальные серверы, которые устанавливаются на компьютер. В них входит такое же программное обеспечение, которое устанавливается на хостингах.

На локальном сервере размещают файлы сайта. После этого к страницам сайта можно обращаться через браузер.

Популярные локальные серверы: Denver, OpenServer, XAMPP. Общие характеристики данных серверов: легкая установка; дружелюбный интерфейс, бесплатность.

Порядок создания сайта на CMS:

1. Скачивание дистрибьютива с сайта-разработчика.
2. Создание базы данных и развертывание CMS на сервере.
3. Задание базовых настроек CMS.
4. Наполнение сайта контентом:
   - создание категорий;
   - создание материалов;
   - создание дополнительных пунктов в меню и связывание материалов;
   - вывод материалов с помощью категорий и подкатегорий;
5. Установка сторонних расширений (плагинов).

7.1. Скачивание дистрибьютива и установка CMS

Для установки системы управления сайтом (CMS) необходимо скачать установочный файл с сайта-разработчика. У каждой CMS этапы развертывания и первичных настроек могут существенно отличаться.

Пример этапов развертывания сайта на CMS Joomla!

Необходимо перейти на сайт разработчика конкретной CMS и скачать установочный файл.

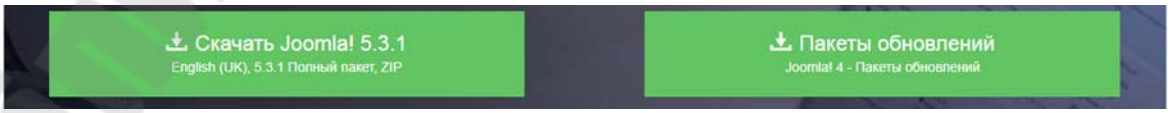

Рис. 24 – Скачивание Joomla с официального сайта разработчика downloads.joomla.org

После скачивания выполняется распаковка файла и создание базы данных на локальном хостинге. Затем запускается установочный скрипт, который проводит настройку CMS и подключение к базе данных.

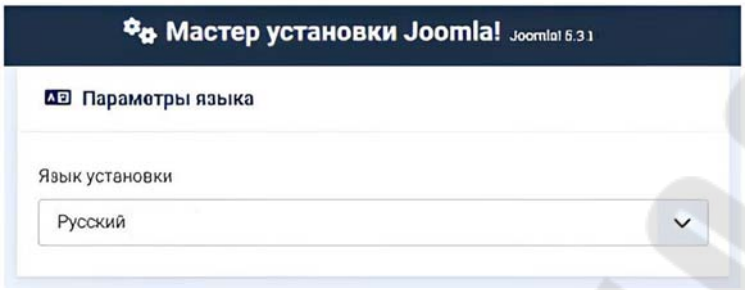

Рис. 25 – Развертывание CMS на локальном хостинге

После успешной установки становится доступна панель управления сайтом. Вход в админ-панель осуществляется по специальной ссылке (loc-domen/administrator/, где loc-domen – это локальное имя сайта). Эта внутренняя часть не видна посетителям и доступна только администратору (super user), когда сайт находится в интернете.

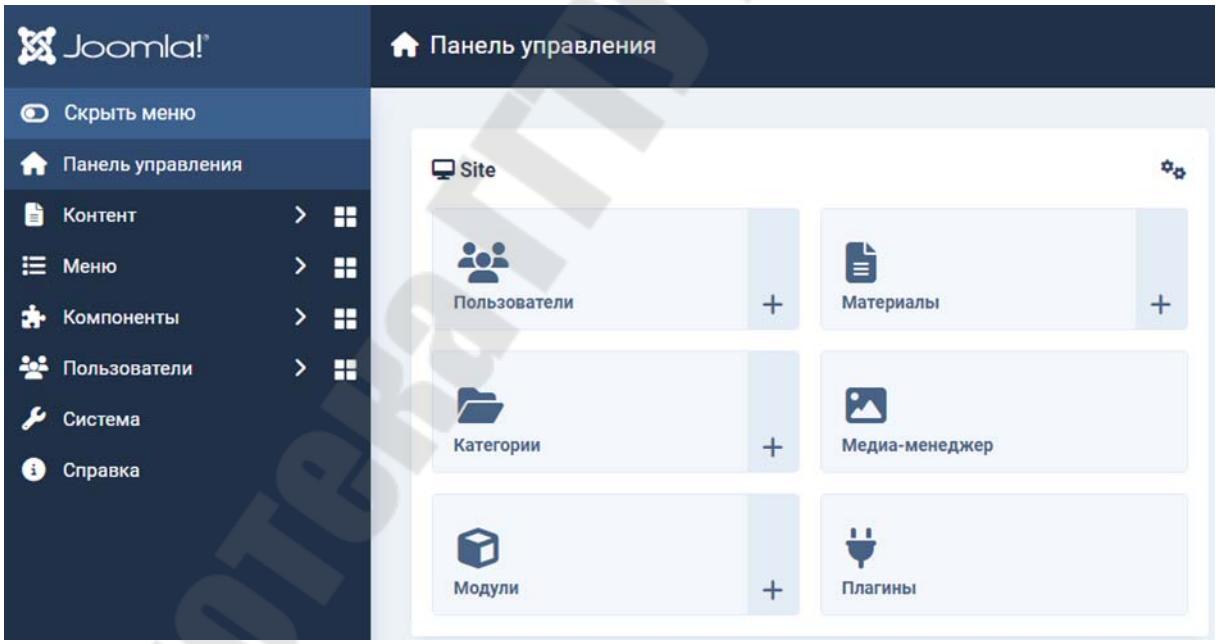

Рис. 26 – Фрагмент Панели управления CMS Joomla!

7.2. Задание базовых настроек CMS

После успешной установки CMS необходимо выполнить первичную настройку, чтобы обеспечить корректную работу сайта.

Первичная настройка включает: название и краткое описание сайта, настройка мета-тегов (<description>,<robots>,<keywords>), создание «человекопонятных» URL-адресов, язык интерфейса и сайта.

1. Название и краткое описание сайта:

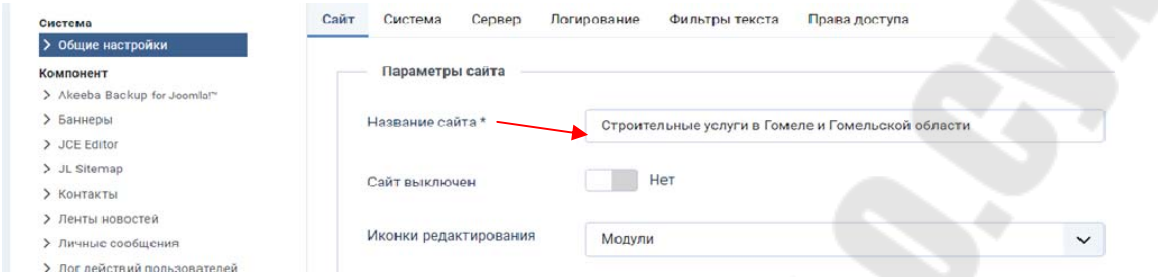

Рис. 27 – Первичная настройка CMS: заполнение названия сайта

2. Настройка необходимых seo-параметров сайта: заполнение метатегов <description> (краткое описание), <robots> (чтение сайта), <keywords> (ключевые слова), SEF, перенаправления URL, суффикс.

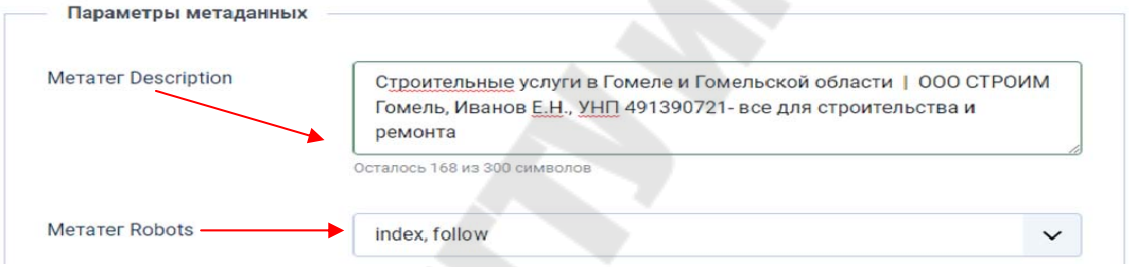

Рис. 28 – Фрагмент первичной настройки CMS: заполнения метатегов

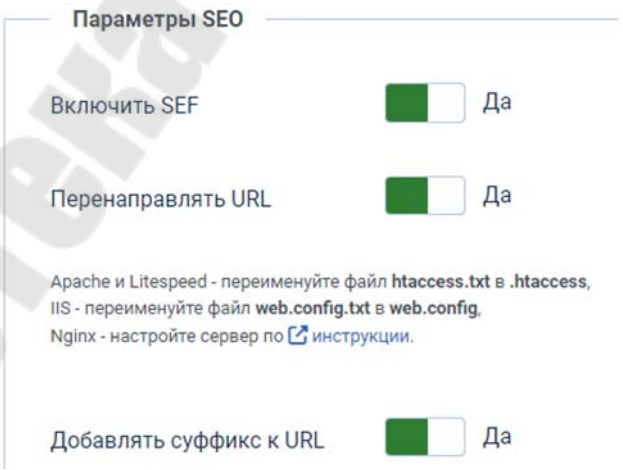

Рис. 29 – Настройка seo-параметров

2.1. Включение SEF – «человекопонятных» URL-адресов (ЧПУ).

Без настройки URl-страницы может выглядеть так: site.by/products/view?category=electronics&product_id=987

После настройки: domen.by/products/electronics/monitor.html

2.2. Подключение перенаправления URL (в Joomla! – удаление из URL цепочки звена /index.php/). Также, потребуется изменить файл htaccess.txt на .htaccess

Без настройки URL страницы будет такой: site.by/index.php/page.html После настройки: site.by/page.html.

2.3. Добавление суффикса <html> к URL (например, site.by/page.html). Если не включить добавление суффикса, то сайт будет работать, а URL-страницы будет такой: site.by/page.
Это необязательная настройка, хотя широко используемая.

3. Установка языка интерфейса и сайта.

Язык интерфейса устанавливается в процессе установки CMS. Настроить отображение языка можно в разделе:
Система/ Управление/ Установленные языки:

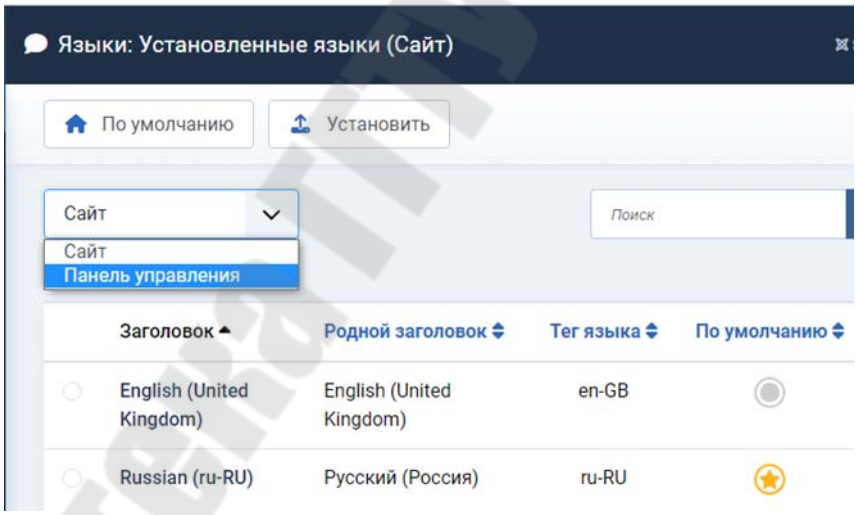

Рис. 30 – Выбор языка сайта и панели управления

На этом этапе также могут настраивать:
- загрузку логотипа и фавикона сайта;
- отключают ненужные модули;
- указывают позицию главного меню;
- настраивают вывод материалов («*Панель управления/Общие настройки/Материалы*»), т.е. будут ли выводиться на странице до-

полнительные сведения: иконки принтера, дата, количества просмотров и пр.

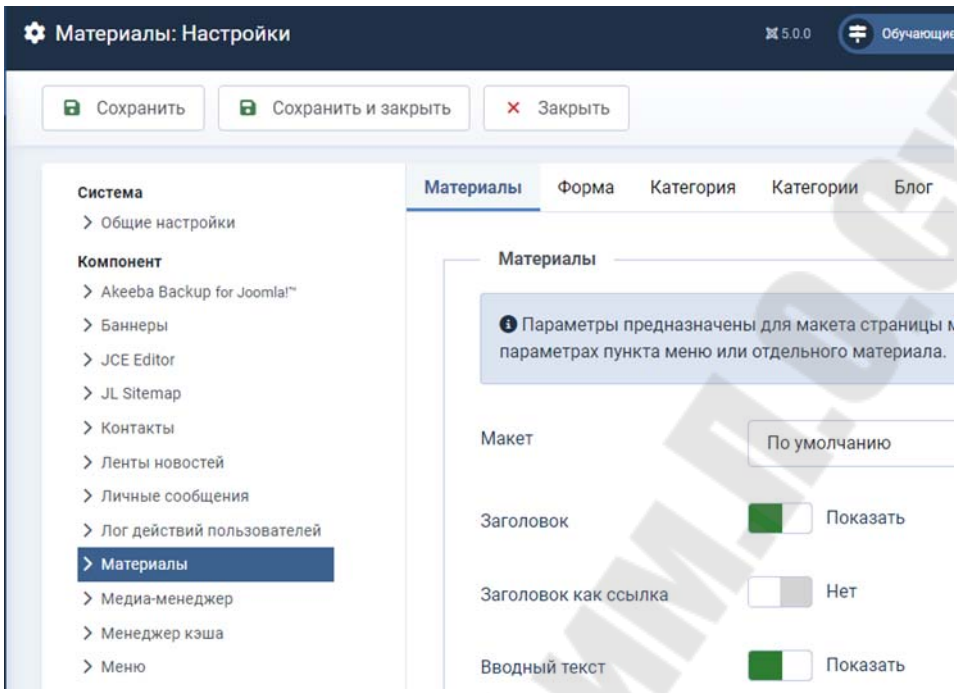

Рис. 31 – Настройки раздела «Материалы».

7.3. Наполнение сайта контентом

После подготовительных работ сайт заполняется: создаются материалы, категории, дополнительные разделы меню.

Создание материалов. Из материалов формируются страницы сайта. Пример создания нового материала:

В админ панели Joomla! нужно перейти в раздел «Материалы» и нажать на кнопку «+Создать»: *Контент/Материалы/ «+Создать»*

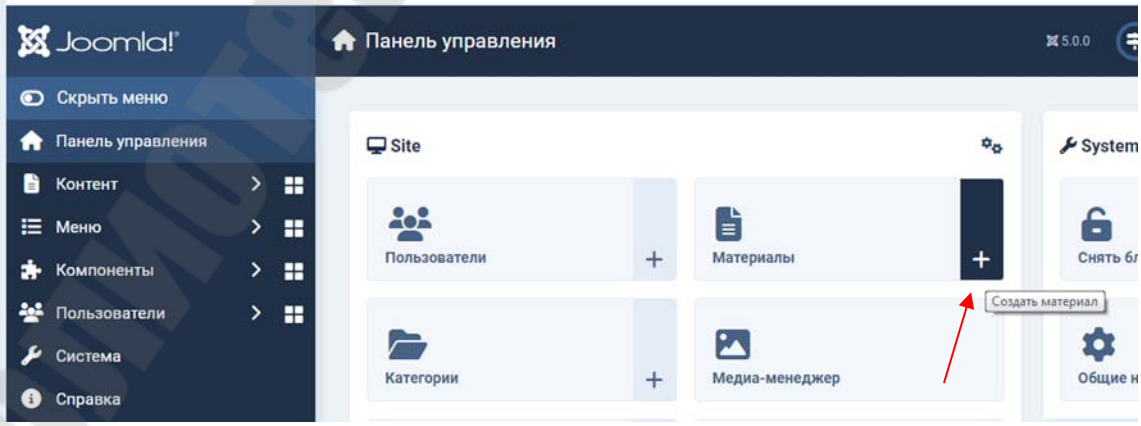

Рис. 32 – Создание материала

При нажатии на кнопку «Создать», открывается редактор материалов с пустой страницей. Материал заполняется контентом (текст, изображение, видео, галерея и т.д.). Если у материала нет категории, то она не присваивается (установка по умолчанию).

Пример создания материала:

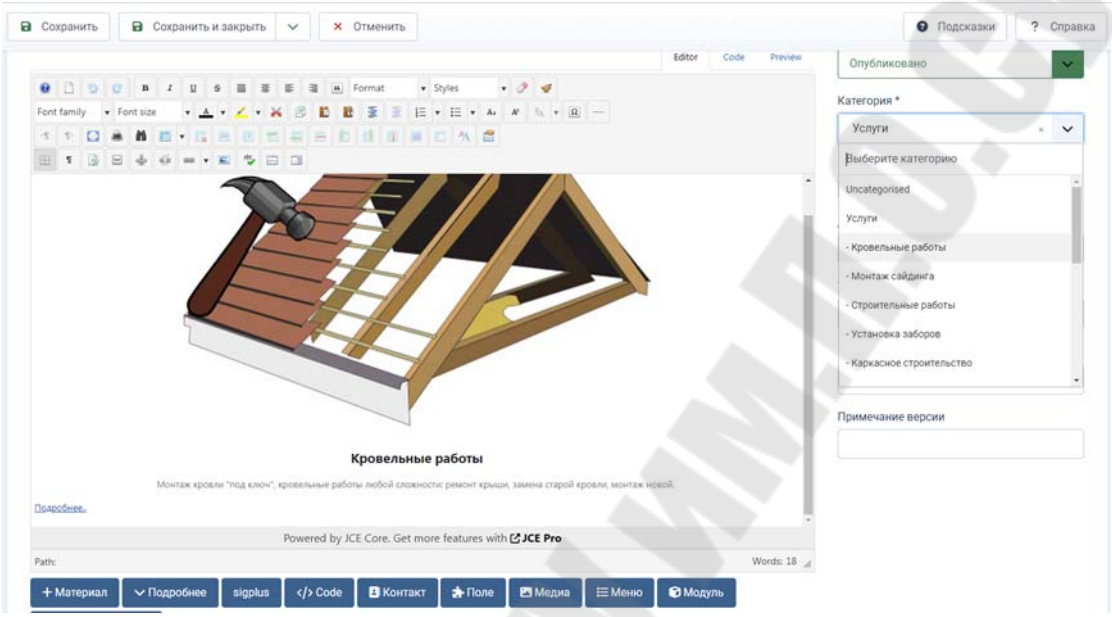

Рис. 33 – Создание материала: заполнение контентом и присвоение категории

Так создаются все нужные материалы. Заголовки материалов совпадают с их названиями.

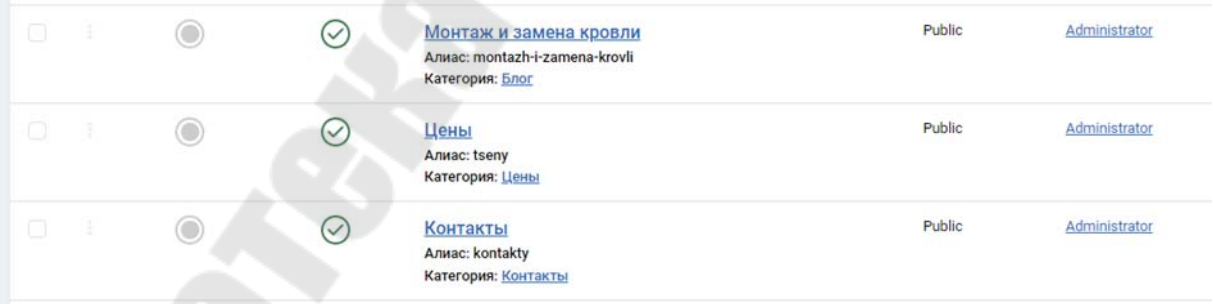

Рис. 34 – Материалы сайта

После создания материалов создаются дополнительные пункты меню, к которым они прикрепляются. Соответственно, новые пункты меню также будут отображаться на сайте.

Связывание материалов с пунктами меню

Изначально, в меню (оно называется Главным) – только одна страница, т.е. она является главной (к ней, в дальнейшем, прикрепляется домен).

Создание пункта меню

Пример создания пункта меню «Контакты» и прикрепление к нему созданной страницы.

В админ-панели Joomla! необходимо перейти в «*Меню/Пункты меню/Создать*»:

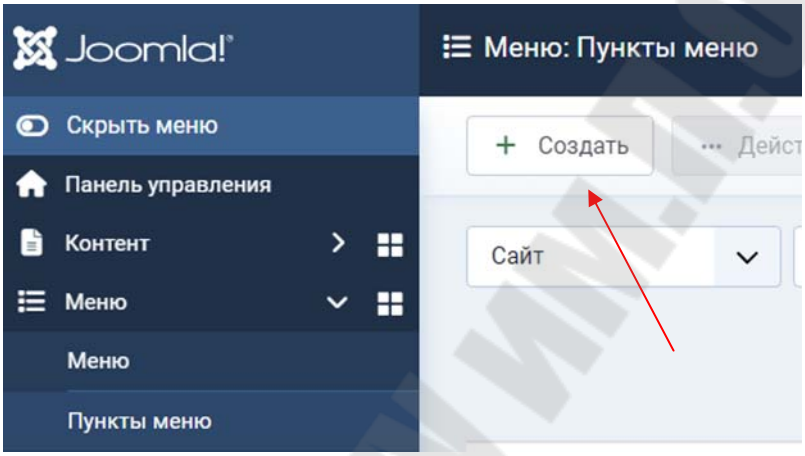

Рис. 35 – Создание пункта меню

Далее, необходимо пошагово настроить данный пункт меню:

- заполнить заголовок (указать название);
- выбрать: «*Тип пункта меню*»/«*Материалы/Материал: Контакты*»;
- завершить настройку нажатием «Сохранить».

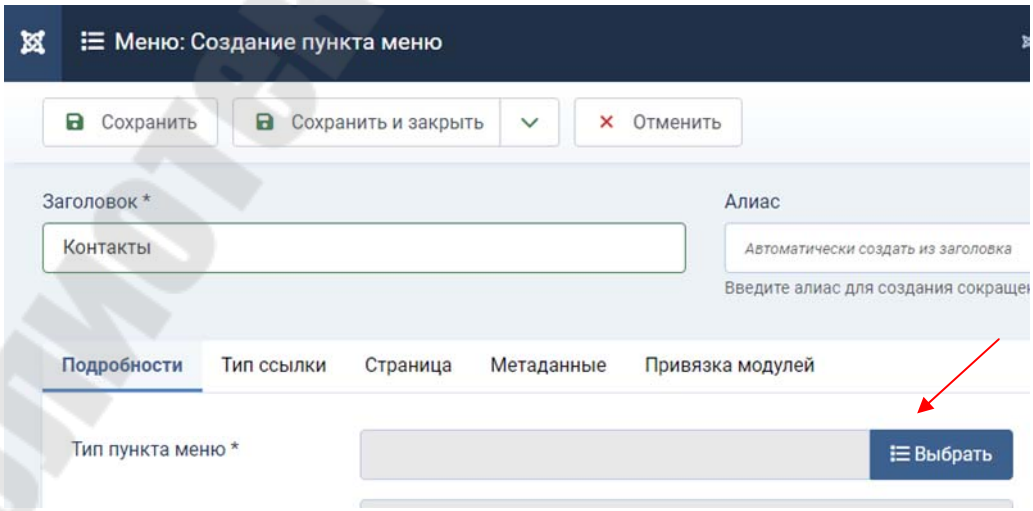

Рис. 36 – Настройка меню «Контакты»

После завершения настройки пункт меню и материал будут связаны, т.е. после перехода по данному пункту посетитель увидит прикрепленный материал.

Вывод материалов с помощью категорий

С помощью категорий информация с нескольких страниц выводится на одной html-странице.

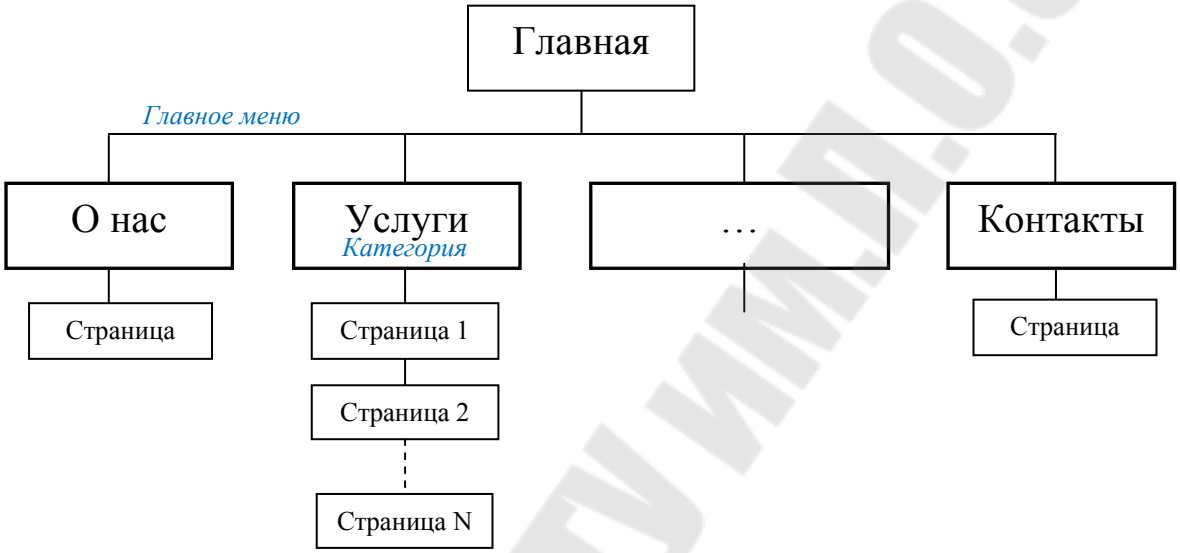

Рис. 37 – Схема простого сайта с одной категорий

Порядок работ:

1. В админ-панели Joomla! необходимо перейти в *Контент/Категории/Создать*». Ввести название категории, сохранить настройку.

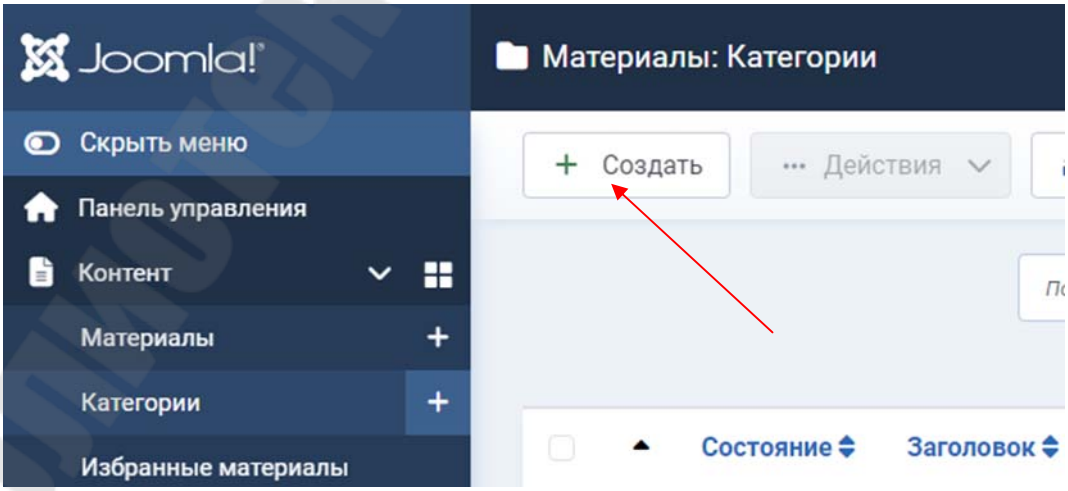

Рис. 38 – Создание категорий

2. Создание материала и присвоение ему категории.

В админ панели Joomla! нужно перейти в раздел «Материалы» и нажать на кнопку «+Создать»: *Контент/Материалы/ «+Создать»*. Заполнить заголовок, метаданные, контент. Материалу присвоить нужную категорию. Сохранить настройки (если материалы уже созданы, то им присваивается нужная категория).

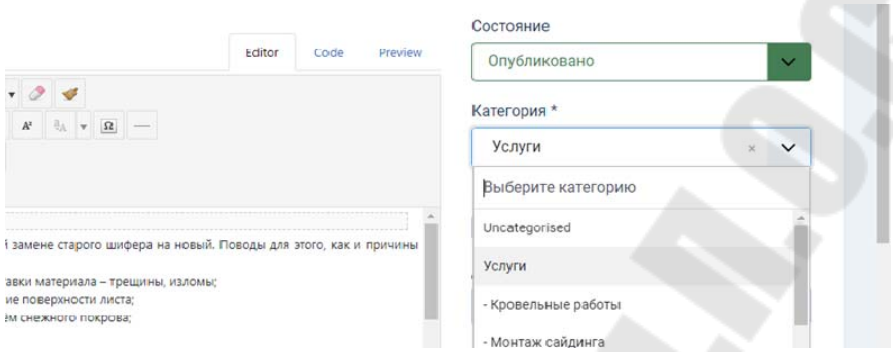

Рис. 39 – Присвоение материалу категории

3. Создание пункта в меню и присвоение ему нужной категории.

В админ-панели Joomla! необходимо перейти в «*Меню/Пункты меню/Создать*»:

Далее, необходимо пошагово настроить данный пункт меню:

- заполнить заголовок пункта меню (обычно заголовок совпадает с названием категории);

- выбрать: «*Тип пункта меню»/«Материалы: Блог категории/Название категории*»:

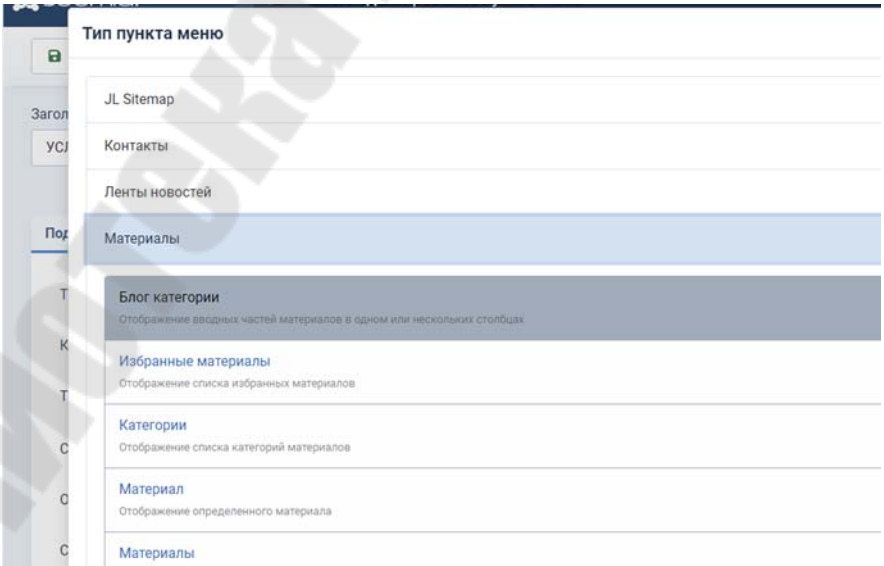

Рис. 40 – Настройка типа пункта меню

Настроить вывод материалов в разделе «Блог» и завершить настройку нажатием «Сохранить».

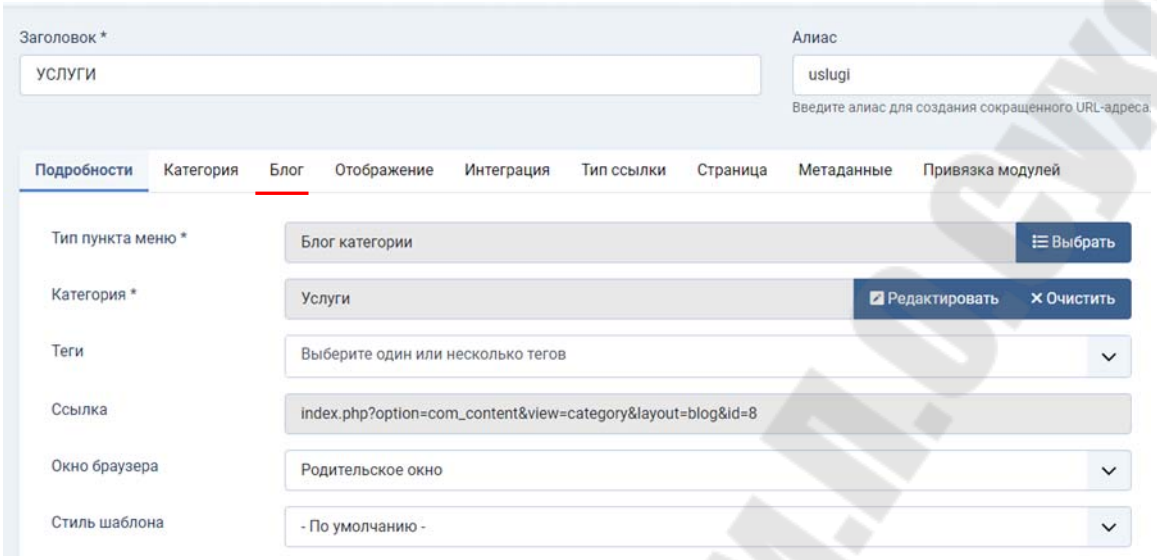

Рис. 41 – Завершение настройки пункта меню

После завершения настроек после перехода по пункту меню на странице будут выводиться материалы только одной категории.

Вывод материалов с помощью нескольких категорий:
Категория может содержать несколько других подкатегорий.

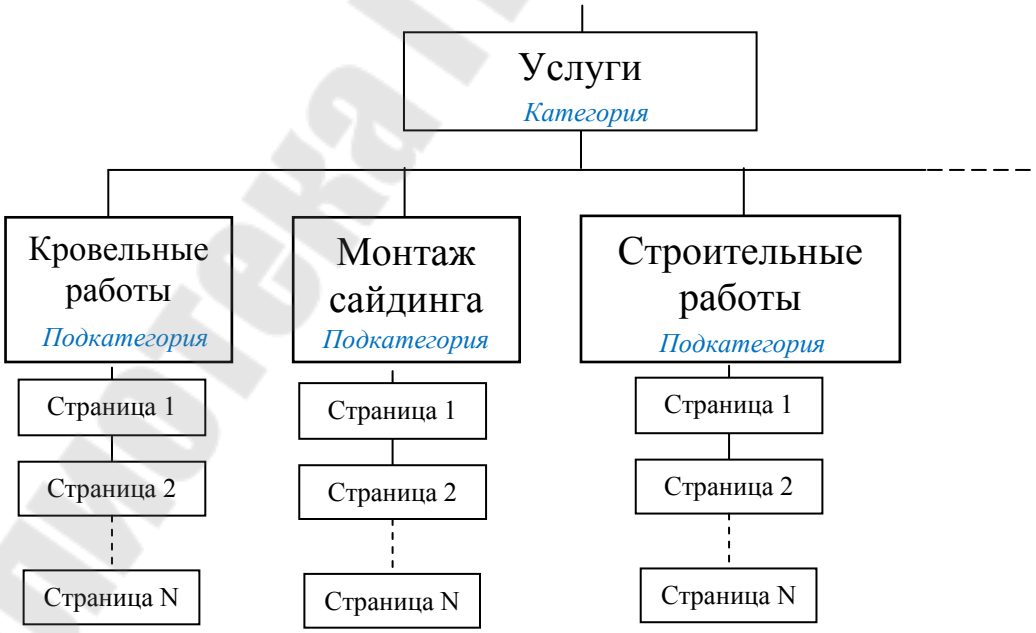

Рис. 42 – Схема вывода подкатегорий

В данном случае на html-странице будут выводиться несколько ссылок, ведущие на каждую из этих подкатегорий. Их можно оформить текстом, иконкой, выводить в меню второго уровня.

Пример вывода материалов на странице категории «Услуги» (*Главная/ Услуги/)*. Переход на другие категории представлен не меню, а изображениями.

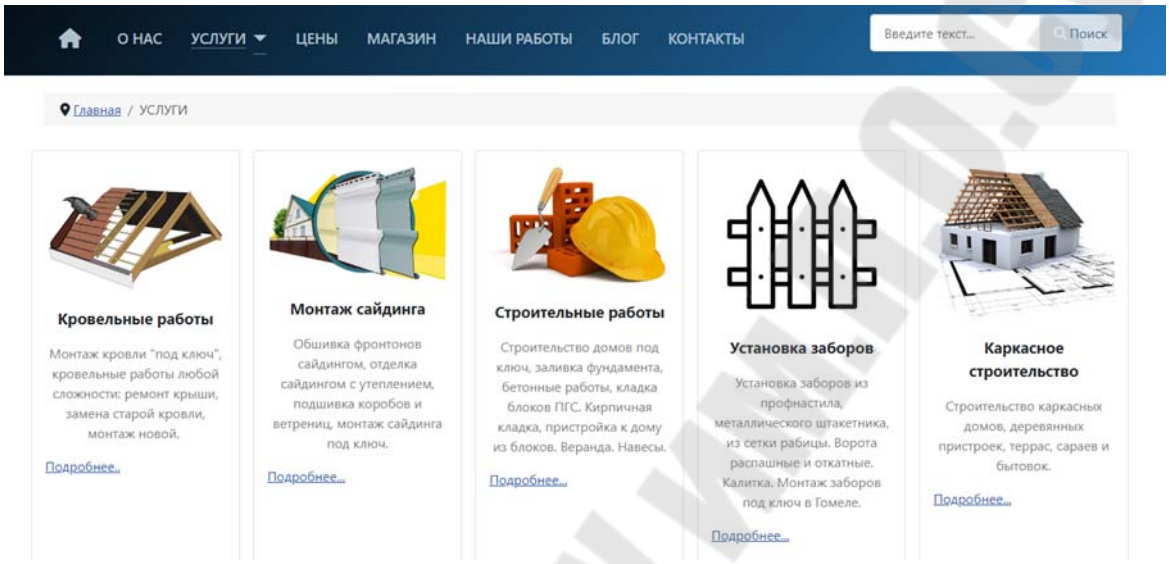

Рис. 43 – Пример вывода материалов подкатегорий «Услуги»

Порядок настройки:

1. Создание подкатегории (*Контент/Категории/Создать*).

В настройках необходимо указать родительский пункт подкатегории:

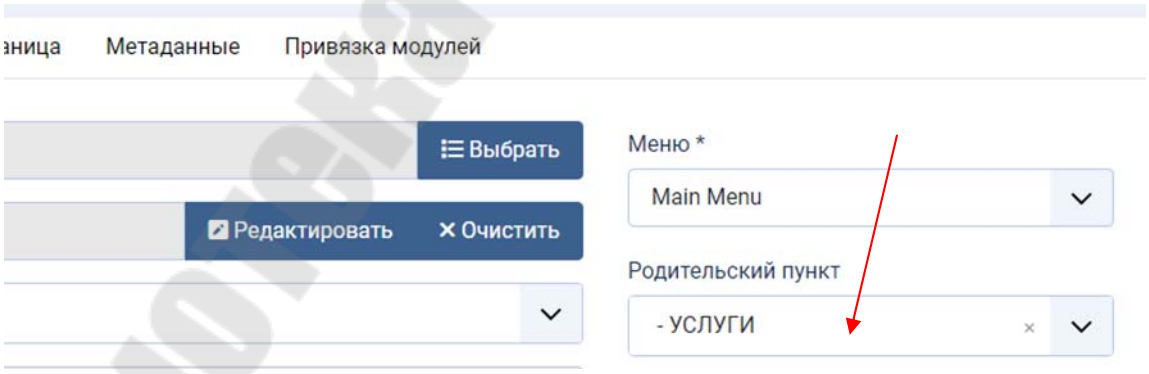

Рис. 44 – Указание родительского пункта подкатегории

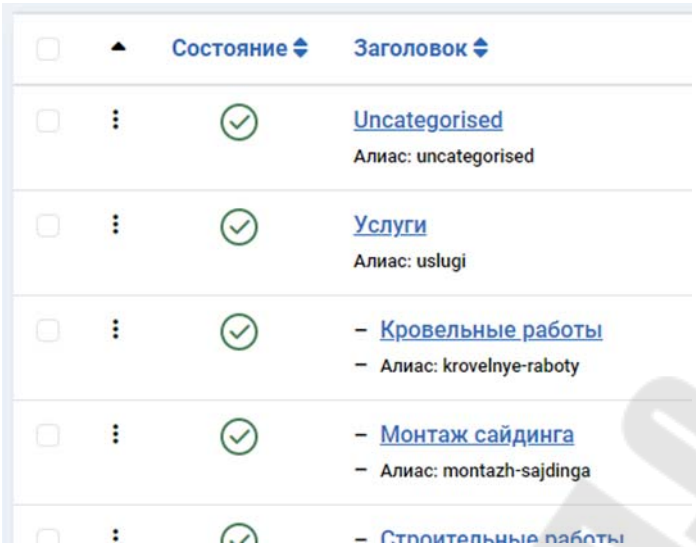

Рис. 45 – Иерархия категорий

2. Создание материала и присвоение подкатегории.

Создание материала: *Контент/Материалы/Создать*. В настройках материала необходимо присвоить нужную категорию (если материалы уже созданы, то им присваивается нужная подкатегория).

3. Создание пункта (или подпункта) в меню и присвоение ему нужной категории.

Создание пункта меню: *Меню/Пункты меню/Создать*. В настройках создаваемого меню необходимо указать родительский пункт, расположенный выше иерархически. А также настроить вывод материалов указанной категории в разделе «Блог».

В настройках вывода данной категории выставлено:

- общее количество материалов на странице – 10;
- количество столбцов – 2.

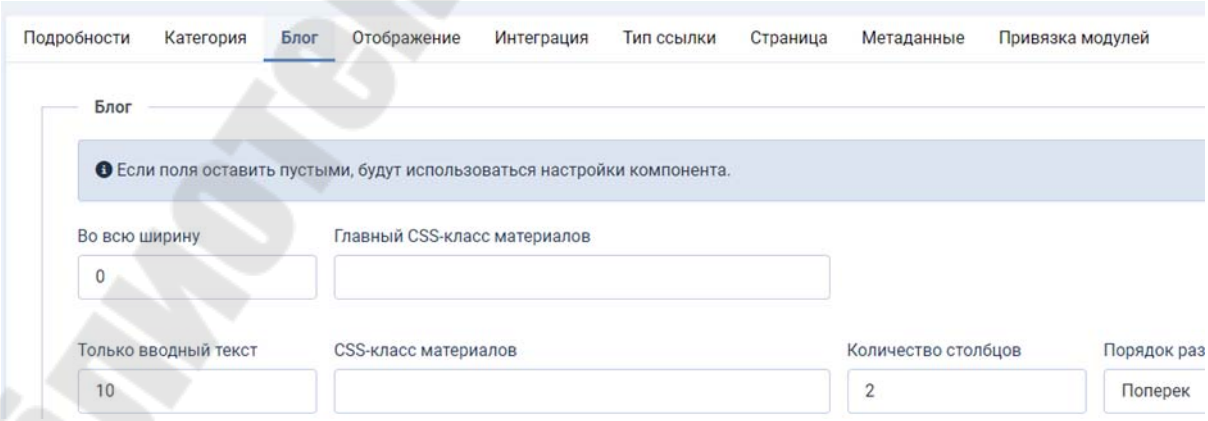

Рис. 46 – Настройка вывода материалов «Блога»

Пример вывода внешнего вида материалов подкатегории «Кровельные работы»:

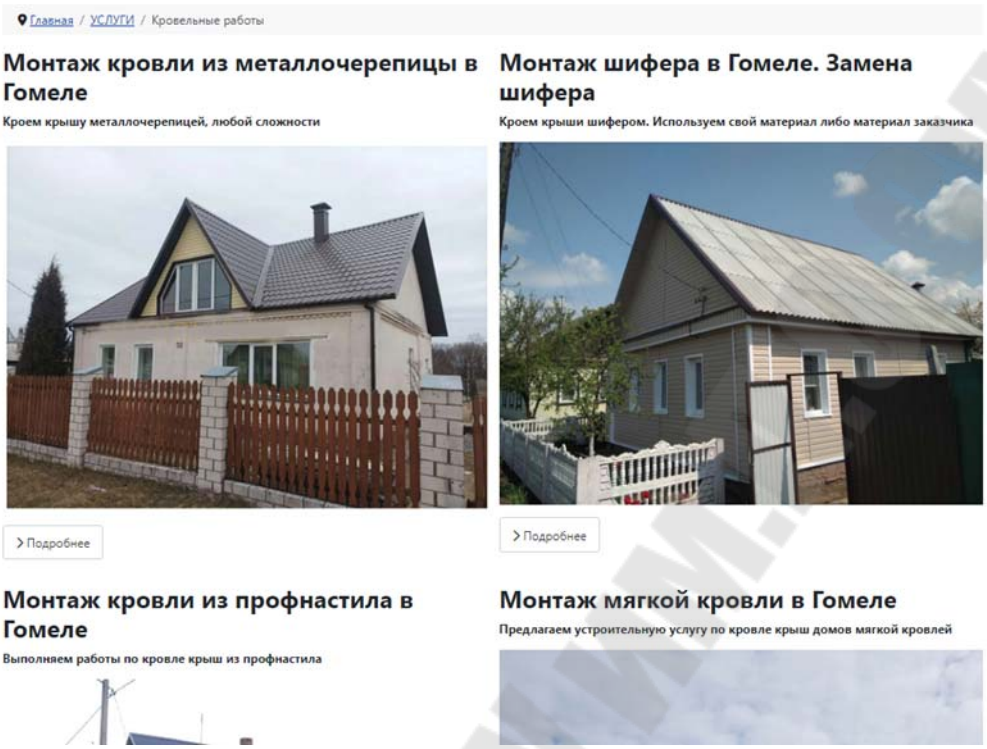

Рис. 47 – Вывод материалов подкатегории в два столбца

Вывод нескольких категорий достаточно сложен и требует множества настроек.

Пример иконографическоего отображения разных категорий на главной странице строймаркета.

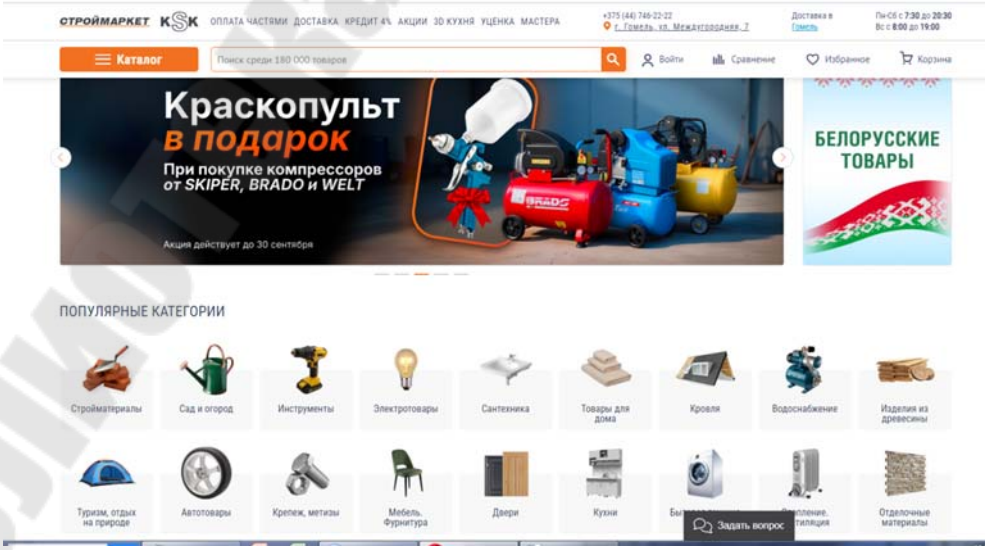

Рис. 48 – Вывод категорий на главной странице

7.4. Установка дополнительных расширений

Для расширения функциональности сайта устанавливаются дополнительные плагины, например такие, как: редактор, галерея, слайдер, шаблоны или темы. Сторонние расширения указываются на сайтах разработчиков конкретных CMS. Эти расширения позволяют улучшить работу сайта, добавить новые возможности и изменить внешний вид. Установка обычно включает загрузку соответствующих файлов и их активацию через административную панель CMS.

Чтобы попасть в соответствующий раздел панели управления, необходимо пройти по ссылке: *Система/ Установка/ Расширения*

Далее, выбрать, загрузить и установить скачанный файл стороннего расширения. После установки появится системное сообщение об успешной или неуспешной установке.

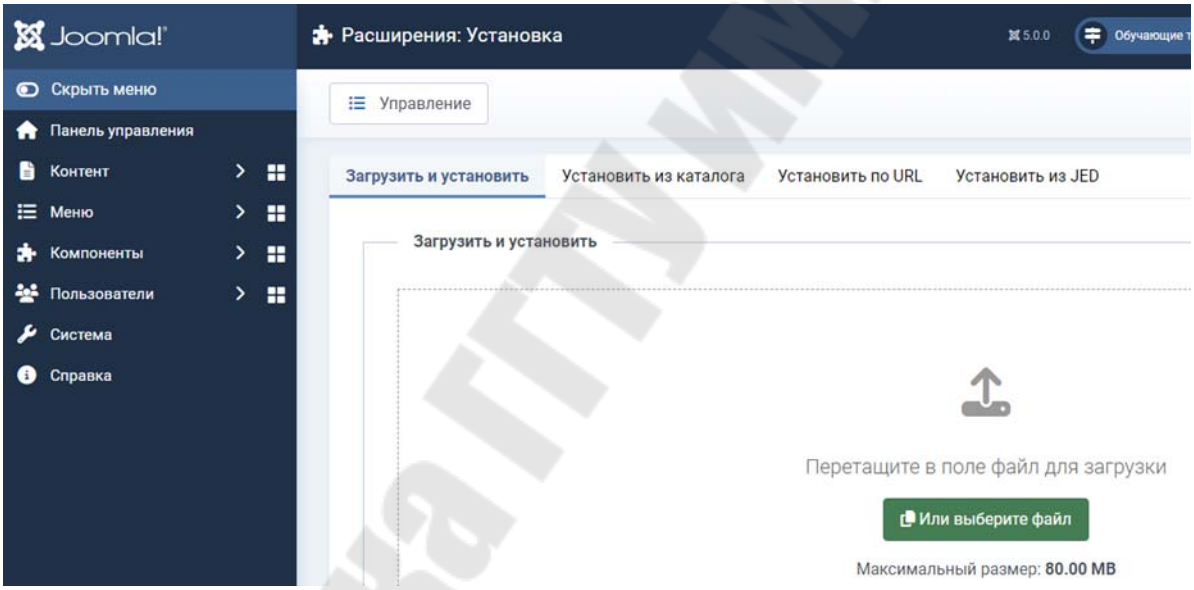

Рис. 49 – Установка сторонних расширений

Установленные плагины доступны для настройки и отображаются в разделе «Компоненты» админ-панели.

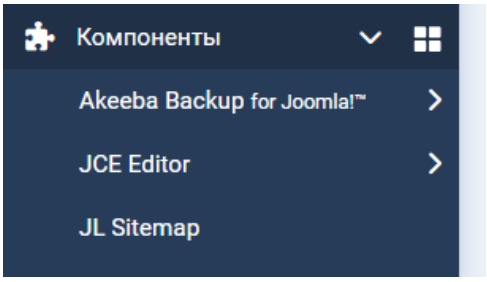

Рис. 50 – Пример установленных расширений в разделе «Компоненты»

Также, управление (отключение и удаление) сторонними расширениями доступно по ссылке: *Система/ Управление/ Плагины*

Пример рабочей области установленных расширений:

- JCE Editor – удобный редактор для создания материалов. На этапе первичных настроек встроенный редактор CMS заменяется на данное стороннее расширение, т.к. оно очень удобно при создании материалов.

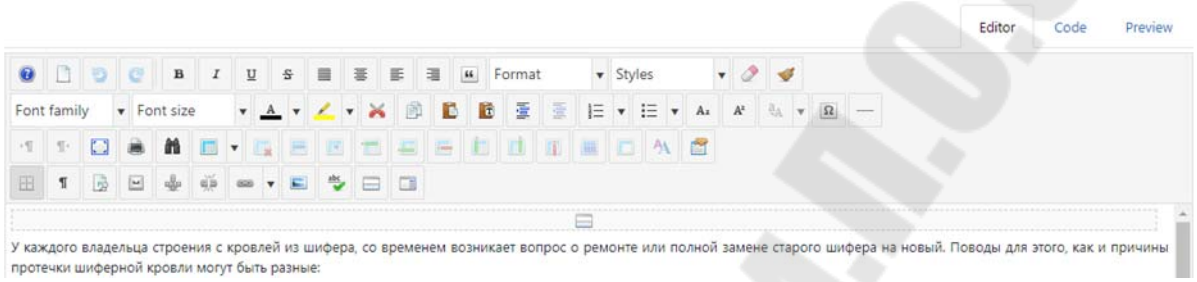

Рис. 51 – Расширение JCE Editor (редактор)

- JL Sitemap – карта сайта для поисковых роботов (используется для внешней оптимизации). Карта Sitemap может устанавливаться до или после публикации сайта в интернет.

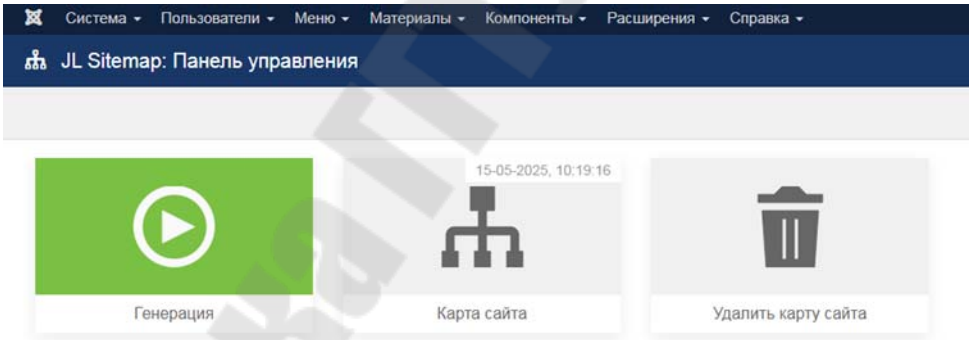

Рис. 52 – Фрагмент установленного расширения – JL Sitemap

- Sigplus – плагин, предназначенный для создания красивых и адаптивных галерей.

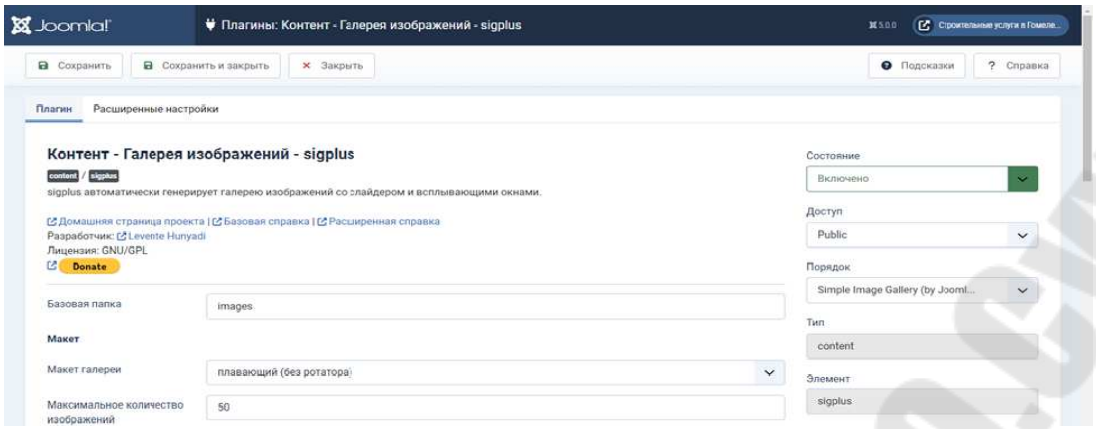

Рис. 53 – Рабочая область расширения Sigplus

Пример вывода изображений с помощью галереи Sigplus:

Рис. 54 – Внешний вид галлереи – Sigplus

- Akeeba Backup – плагин, предназначенный для создания копий сайта (бэкапов).

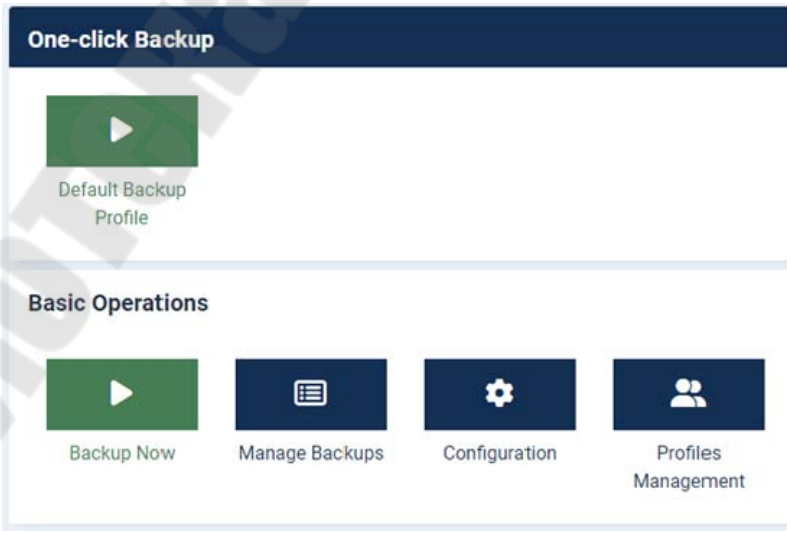

Рис. 55 – Рабочая область Akeeba Backup

Примеры полезных плагинов для CMS Joomla! и WordPress

Таблица 4: Полезные плагины для CMS Joomla! и WordPress:

CMS Joomla	CMS WordPress	Назначение
Editor-JCE	Elementor	Редактор контента
Regular Labs - Sourcerer	Insert Headers and Footers	Вставка кода
Phoca Gallery, Sigplus	Envira Gallery, NextGEN Gallery	Галерея
Helix3, Helix Ultimate	Astra, OceanWP	Тема/шаблон
JL Sitemap	Yoast SEO	Карта сайта для seo
Akeeba backup	Akeeba Backup	Создание бэкапов

Аналогичные сторонние расширения разработаны и для других CMS.

Разработка и применение шаблонов для организации клиентской части ресурса

Для изменения внешнего вида клиентской части сайта устанавливают специально разработанный шаблон, предназначенный для данного проекта, а если его нет, то кастомизируют сторонний шаблон либо встроенный.

Шаблоны для сайта — это готовые дизайн-макеты, которые используются для оформления и структурирования внешнего вида клиентской части сайта. Это заранее подготовленный набор файлов (HTML, CSS, JS, изображения), который определяет внешний вид и структуру страниц сайта. Такой шаблон позволяет быстро и последовательно оформить сайт без необходимости писать всё с нуля.

Пример установочного шаблона для Joomla! 5 «Cassiopeia».

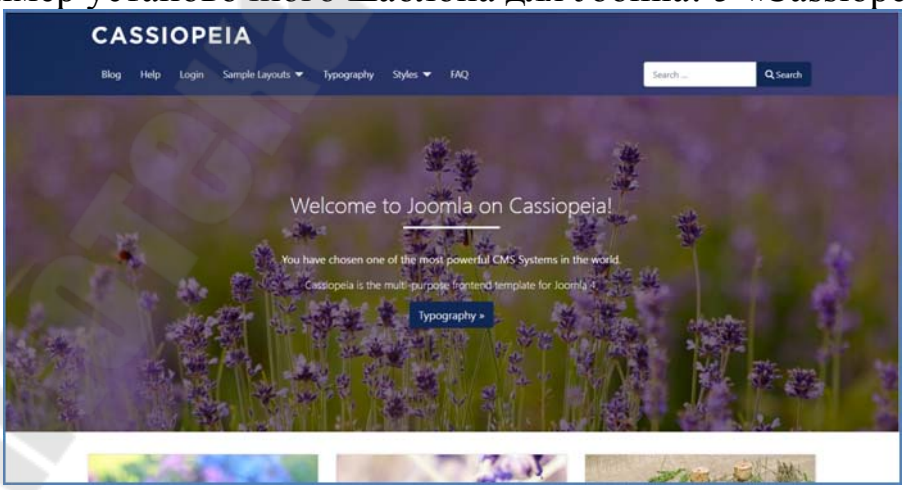

Рис. 56 – Установочный шаблон Joomla! 5 «Cassiopeia»

Кастомизация — это настройка и адаптация уже существующих решений. В настройку текущей темы шаблона входит: изменение цвета, шрифтов, расположения элементов, добавление пользовательского CSS и внесение правки в файлы шаблона для создания уникального дизайна.

Пример стороннего шаблона для CMS Joomla! – Helix3 Framework. Данный шаблон кастомизируют, изменяют под проект: настраивают стили, шрифты, расположение элементов, добавляют пользовательский CSS.

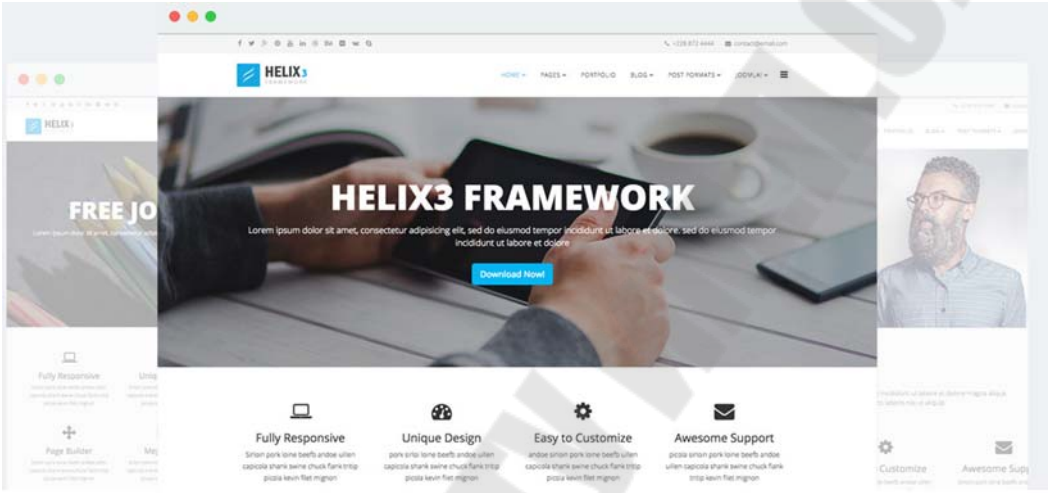

Рис. 57 – Шаблон Free Helix3 для Joomla 5

Пример готовых шаблонов CMS от разработчиков Rockettheme и JoomShaper для организации клиентской части ресурса сайта.

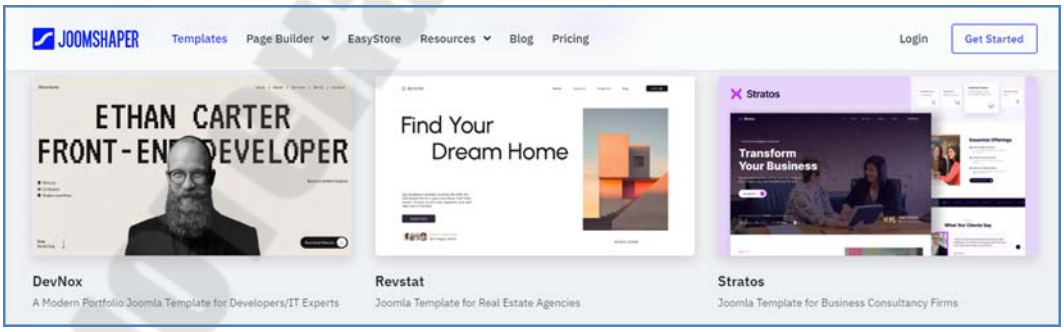

Рис. 58 – Шаблоны для CMS от joomshaper.com

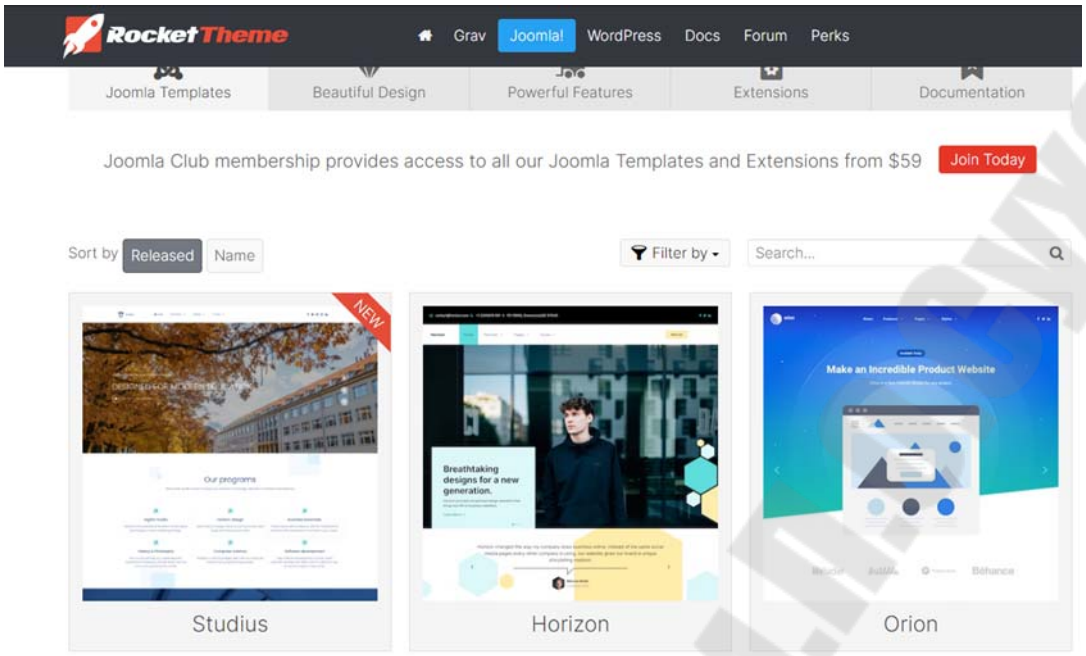

Рис. 59 – Шаблоны для CMS от rockettheme.com

7.5. SEO-оптимизация контента

Во время заполнения материалов проводится внутренняя оптимизация. Основные этапы внутренней оптимизации:

1. Заполнение метатегов <title>, <description>, <keywords>.

2. Оптимизация контента:

- использование в заголовках, подзаголовках и тексте ключевых слов и фраз, по которым продвигается страница;

- структурирование текста — заключение заголовков в теги <h1> - <h6>, выделение ключевых слов и фраз с помощью жирного и курсивного шрифта, использование абзацев и списков и т.д.;

- создание уникальных, качественных и полезных материалов, избегая дублирования;

- оптимизация изображений: уменьшение размера, заполнение атрибутов <alt> и <title> с ключевыми словами или синонимами, использование современных форматов изображений;

- улучшение внутренней перелинковки — добавление ссылок между релевантными страницами.

3. Настройка производительности сайта.

Пример структурированного текста шаблона Joomla Cassiopeia:

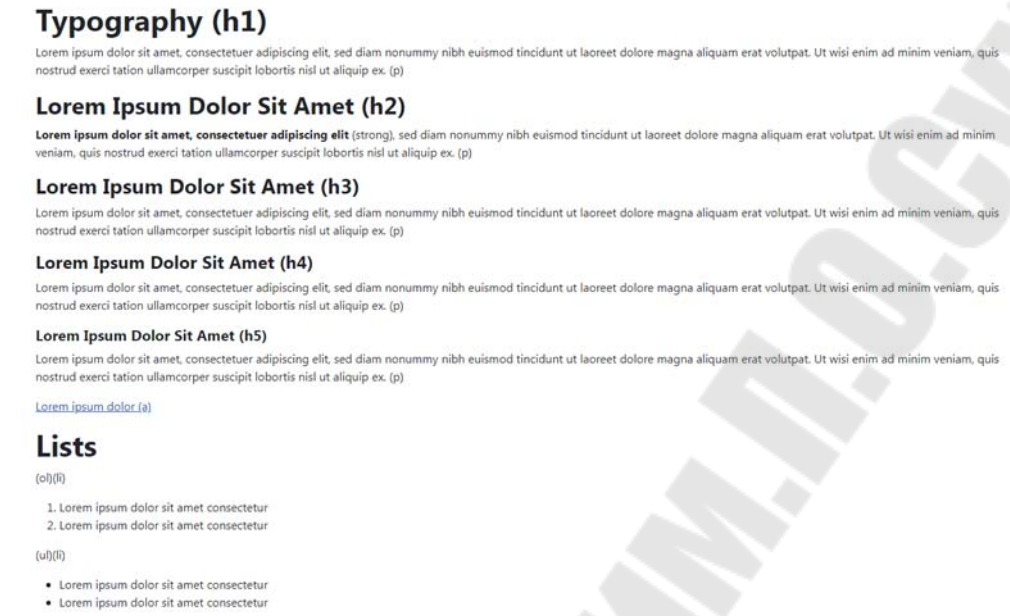

Рис. 60 – Структурированный текст шаблона Joomla Cassiopeia

7.6. Настройки пользователей и безопасности

Если на сайте предполагаются регистрации, то настраиваются: роли пользователей и уровни доступа:

Пример раздела сайта Joomla для управления пользователями: *Пользователи/ «+Создать»*

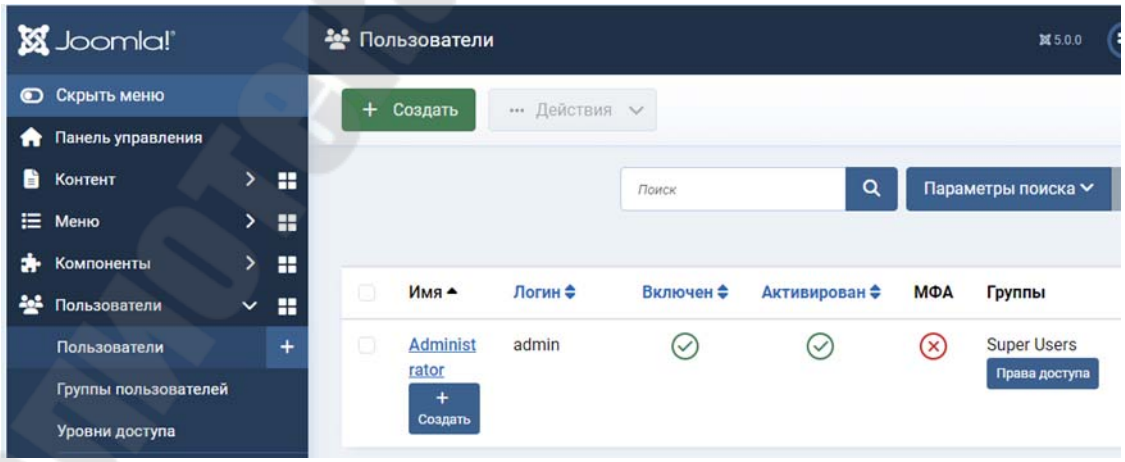

Рис. 61 – Раздел сайта Joomla для управления пользователями

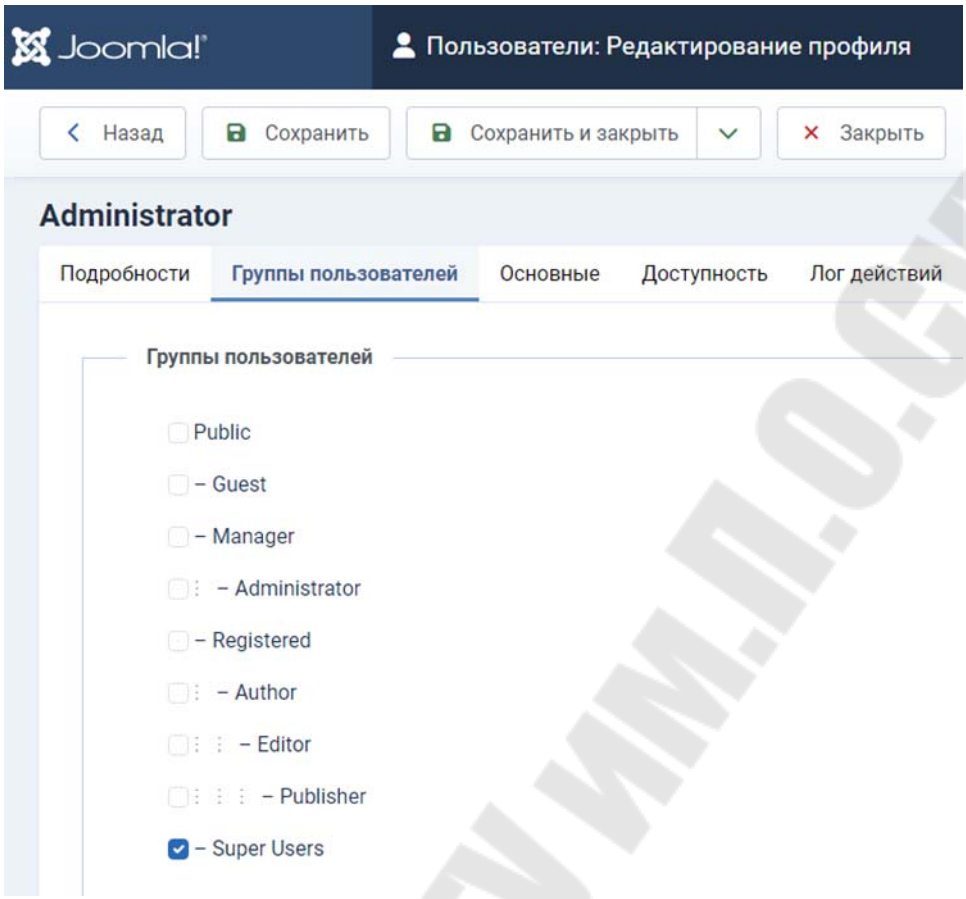

Рис. 62 – Настройка прав для пользователя

Основные роли пользователей в Joomla!

В Joomla! права доступа управляются через группы пользователей и разрешения. Можно создавать собственные группы и настраивать их права по необходимости.

Примеры ролей пользователей:

Гостевой пользователь (*Guest*) — не авторизованный пользователь, имеет минимальные права, обычно только просмотр публичного контента.

Менеджер (*Manager*) — имеет расширенные права управления контентом и пользователями, может управлять разделами сайта.

Администратор (*Administrator*) — полные права на управление сайтом, настройками, расширениями, пользователями и т.п.

Пользователь (*Registered*) — зарегистрированный пользователь, может просматривать ограниченный контент, участвовать в комментариях и т.п.

Автор (*Author*) — может создавать собственные статьи и материалы, но не может публиковать их без одобрения.

Редактор (*Editor*) — может редактировать и публиковать материалы как свои, так и чужие, управлять контентом сайта.

Публикатор (*Publisher*) — это группа пользователей, которая обладает правами публиковать, редактировать и управлять контентом, но обычно не имеет полного доступа к системным настройкам сайта.

Суперпользователь (*Super User*) — полный контроль над сайтом, все возможные права без ограничений.

Для эффективной защиты веб-ресурса от спама и действий вредоносных ботов рекомендуется использовать комплекс мер, включающих двухфакторную авторизацию, аутентификацию через надежные сервисы вроде Gmail, интеграцию системы CAPTCHA, а также специализированные программы для защиты и управления сайтом, например – «admin tools» для Joomla! и WordPress.

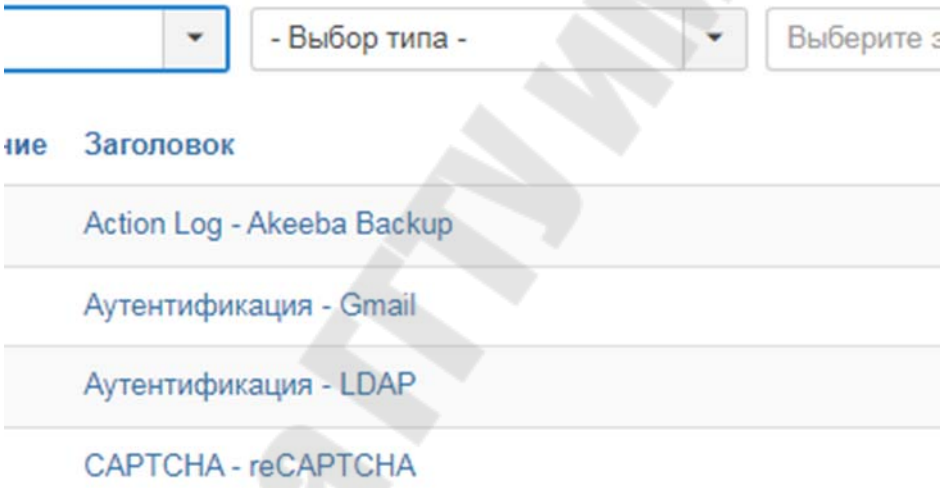

Рис. 63 – Фрагмент встроенных функций CMS Joomla!

7.7. Проверка работоспособности проекта

После заполнения сайта материалами, сайт тестируют на корректность работы и отображения в браузерах, роботоспособность ссылок и адаптивность шаблона для мобильных устройств.

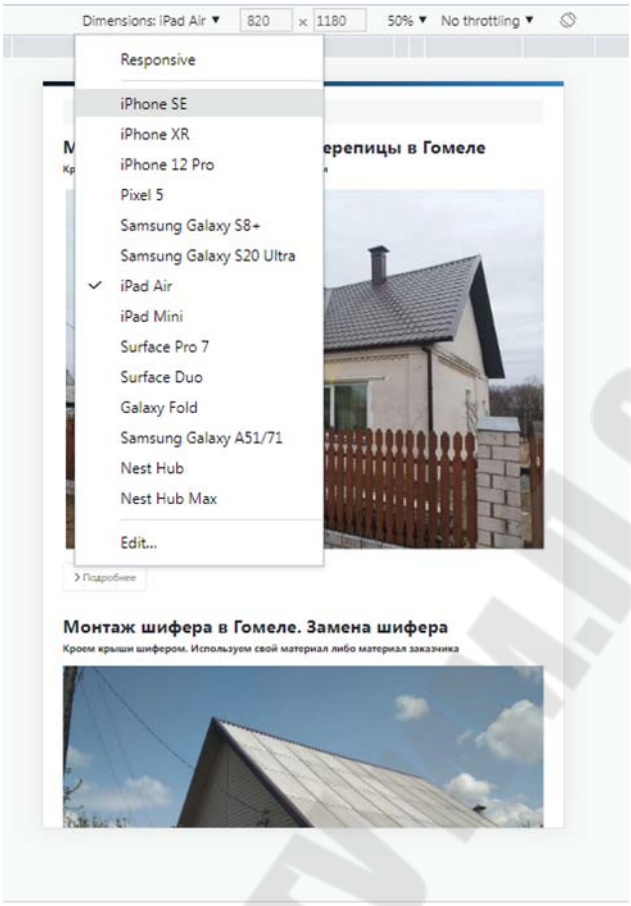

Рис. 64 – Тестирование сайта на разных устройствах

7.8. Публикация сайта в интернет

После завершения тестирования и окончательной проверки сайта подбирается доменное имя и хостинг-провайдер. Для переноса файлов сайта на сервер используется FTP-клиент, например, FileZilla. С помощью этого программного обеспечения файлы сайта загружаются на хостинг, обеспечивая его доступность в сети.

Рис. 65 – FTP-клиент FileZilla (filezilla-project.org)

7.9. Подключение HTTPS (SSL-сертификат)

Для обеспечения безопасности передачи данных на сайт необходимо подключить SSL-сертификат.

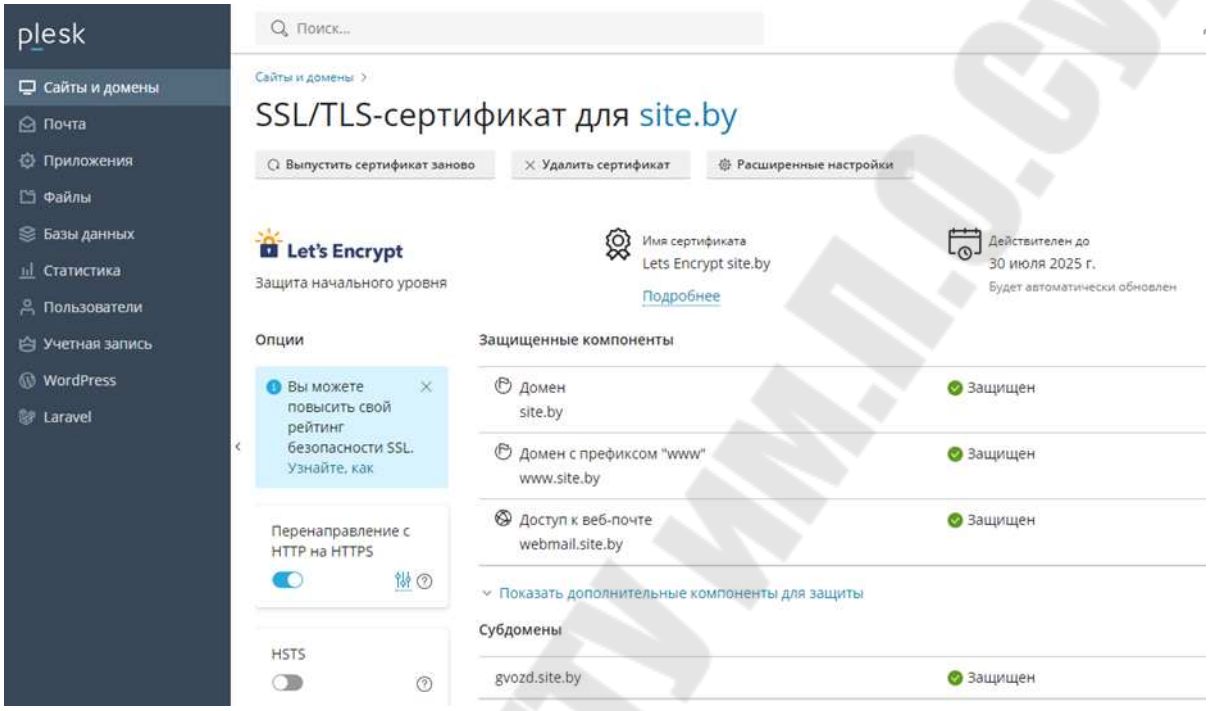

Рис. 66 – Пример настройки сертификата на хостинге и подключение перенаправления с HTTP на HTTPS

Это шифрует передаваемую информацию между браузером пользователя и сервером, защищая личные данные, пароли и платежные реквизиты от перехвата и несанкционированного доступа.

Пример домена до подключения сертификата HTTPS: http://site.by После подключения вид будет такой: https://site.by

Подключение сертификата к домену сайта осуществляется на хостинге. Перед этим вносятся необходимые записи в файл robots.txt (site.by/robots.txt) и настраивается редирект с www на без (либо наоборот в файле .htaccess).

Использование HTTPS повышает доверие посетителей и способствует улучшению позиций в поисковых системах.

7.10. Техническая оптимизация сайта

После переезда на хостинг скорость сайта может снизится. Несмотря на использование современного оборудования и хорошую настройку хостинга, снижение скорости сайта может быть связано с рядом факторов, не связанных напрямую с аппаратной частью. Это может произойти из-за:

- проблем с сетью или DNS (медленное обновление DNS);
- избыточные проверки безопасности или фильтры могут замедлять обработку запросов;
- ограничения хостинга на лимиты CPU (использование процессора), RAM (памяти) или I/O (трафик) – обычно отображаются в тарифах; временные ограничения на выполнение скриптов и др.

Рис. 67 – Пример установленных ограничений хостинга

Проблемы со стороны сайта:

- отсутствие настройки кэширования;
- неоптимизированный код и тяжелые скрипты;
- использование большого количества плагинов;
- неиспользуемые или устаревшие расширения;
- неоптимизированные изображения;
- большие файлы CSS/JS, отсутствие минификации и объединения файлов;
- некоторые настройки сайта, например кэширование, лучше всего настраивать после переезда на новый сервер или домен;
- использование стороннего шаблона.

Действия для устранения проблем:

- провести диагностику скорости с помощью инструментов (google pagespeed insights, webpagetest и др.)
- проверить работу dns и сетевое соединение;
- обратиться к хостинг-провайдеру за консультацией по ресурсам и настройкам сервера;
- включить и правильно настроить кэширование и сжатие на сайте, оптимизировать изображения и скрипты.

Настройки производительности сайта:

1. Настройка кэширование страниц и данных – использование внутренних настроек CMS для быстрой загрузки страниц сайта (включение кэширования). Кэш сохраняет копии часто запрашиваемых ресурсов на определенное время (например, страниц сайта, изображений, результатов запросов к базе данных), что сокращает время загрузки страницы в браузере для посетителя.

2. Минификация CSS и JS файлов – использование плагинов для улучшения процесса загрузки страниц сайта. Специализированные плагины объединяют JavaScript и CSS-файлы в один, что снижает количество HTTP-запросов браузера и ускоряет загрузку страницы. В результате сайт открывается быстрее, что положительно влияет на его посещаемость.

Пример плагинов для Joomla, влияющие на ускорения работы сайта:

- «JCH Optimize», «Jbetolo», «ScriptMerge», «JCC – JS CSS Control» - объединяют Javascript и CSS-файлы в один файл.
- «NoNumber Cache Cleaner» – очищает кэш одним кликом по ссылке в Панели Управления.
- «CDN for Joomla!» – это расширение, позволяет создать простую интеграцию с сетью доставки контента (CDN) и другие.

Аналогичные расширения для оптимизации загрузки страниц имеются и для других CMS: Wordpress, Drupal, DLE и пр.

3. Настройки резервного копирования с помощью хостинга (функция cron — планировщика задач) или сторонних расширений (например, Akeeba backup).

4. Проверка и исправление ошибок в коде, устранение «битых» ссылок и неработающих скриптов.

5. Удаление неиспользуемых материалов, модулей и пр.

6. Регулярный мониторинг производительности сайта с помощью специальных инструментов (например, Google PageSpeed Insights) для выявления узких мест и дальнейшей оптимизации.

Техническая оптимизация сайта (настройка производительности) способствует SEO-оптимизации и является её важной частью.

Тестирование:

После завершения технической оптимизации сайт тестируют на производительность и мобильность с помощью специальных сервисов.

Пример результата тестирования сайта с помощью сервиса pagespeed.web.dev (Google):

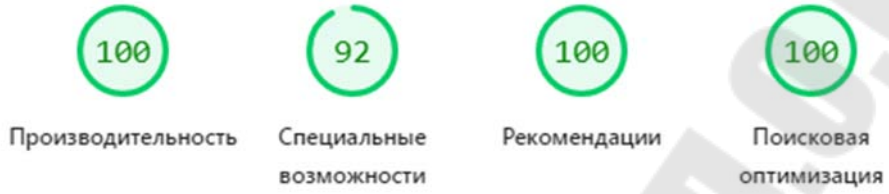

Рис. 68 – Фрагмент данных сайта после теста

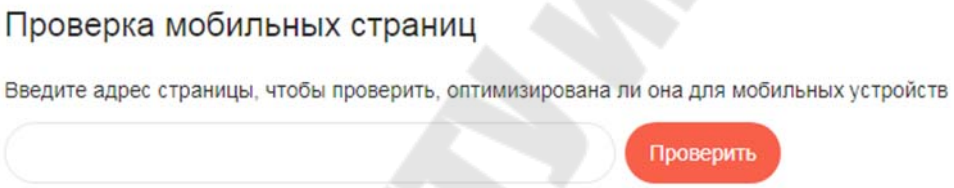

Рис. 69 – Проверка мобильности страниц с помощью Яндекс Вебмастера

После настройки производительности, сайт добавляется в поисковые системы.

8. ДОМЕН И ХОСТИНГ

8.1. Пространство доменных имен

Пространство доменных имен — это система, которая помогает организовать и упорядочить адреса сайтов в интернете. Каждый сайт имеет свой уникальный адрес, который называется доменным именем. Например: google.com, wikipedia.org, yandex.by.

Когда в браузере вводится доменное имя, происходит следующее:

1. Браузер отправляет запрос на сервер, который хранит информацию о доменах.
2. Сервер ищет, к какому IP-адресу (уникальному числу, которое идентифицирует сервер в интернете) относится это доменное имя.
3. Браузер получает IP-адрес и подключается к серверу, где размещен сайт.

Структура и иерархия доменных имен

Доменное имя состоит из нескольких частей, например - www.yandex.by:

1. Субдомен (необязателен): «www».
2. Название сайта (доменное имя): «yandex».
3. Доменная зона: «.by», «.com», «.ru», «.org» и другие.

Домен верхнего уровня (top-level domain, TLD): последний сегмент имени, который показывает, к какой категории или стране относится сайт (доменная зона), например:

- .com и .biz – домены часто используется для коммерческих сайтов не зависимо от региона;
- .ai, .app – сервисы использующие для своих услуг искусственный интеллект;
- .рф и .ru — для сайтов из России;
- .бел и .by – для сайтов белорусских организаций и т.д.

Таблица 5 – Тип и назначение доменных имен

Тип	Примеры	Назначение
Национальные	.бел, .by, .ru, .ua, .kz	Привязка к стране
Общие	.com, .org, .net, .site	Универсальные
Специализированные	.gov, .int	Международные государственные домены

Домен второго уровня: Примеры доменов второго уровня: gstu.by, yandex.by, rutube.ru, vk.com и др.

Домен третьего уровня: используется для создания поддоменов и разделения ресурсов.

Например, домен gstu.by – домен второго уровня.
А portal.gstu.by, edu.gstu.by, ipk.gstu.by – домены третьего уровня.

В белорусском сегменте интернета домены третьего уровня широко используются для сайтов организаций в регионах. При этом сайт успешно индексируется поисковыми системами и конкурирует на региональном уровне.

Пример доменных имен третьего уровня: Министерство образования Республики Беларусь (edu.gov.by), Калинковичский завод железобетонных изделий (kgbi.gomel.by), Жилищно-коммунальное хозяйство Гомельской области (ugkh.gomel.by), Городской центр культуры (gck.gomel.by), Гомельская мебельная фабрика «Прогресс» (progres.gomel.by), Техника в рассрочку от Белтелеком (tehnika.gomel.by), УП «Городская аварийная служба» (gas.minsk.by), Мебель (mogilev.nebo.by и brest.nebo.by) и т.д.

Домен четвертого уровня: Применяется реже. В основном используется для организаций, например: Гомельское областное управление МЧС (gomel.mchs.gov.by), Национальный статистический комитет Республики Беларусь (gomel.belstat.gov.by).

Регистраторы доменных имен

Регистраторы доменных имен — это компании, которые управляют процессом регистрации доменных имен и обеспечивают их поддержку.

Регистраторы выступают посредниками между владельцами доменных имен и организацией, управляющей доменным пространст-

вом. Они помогают в регистрации, переносе и управлении доменными именами.

Пример наиболее популярных белорусских регистраторов доменов: ООО «Активные технологии» activecloud.by, ООО «Надежные программы» hoster.by, ООО «ТриИнком» hb.by и другие.

Исключение представляет компания Белтелком beltelecom.by (оператор Республики Беларусь по предоставлению услуг электросвязи), которая предоставляет широкий спектр услуг хостинга, а также домены третьего уровня для сайтов своих клиентов, например вида: site.gomel.by и др.).

- Плюсы: предоставления доменов третьего уровня – разовый платеж; положительно воспринимается поисковыми системами, хорошо подходит для региональных организаций; возможность парковки к сайту собственного домена второго уровня.

- Минус – сайт не сможет переехать на другой хостинг, т.к. потеряет свое доменное имя (доменное имя третьего уровня), если он представлен в интернете под данным доменом.

Перечень аккредитованных белорусских регистраторов доменов можно посмотреть на официальном сайте доменных зон .BY и .БЕЛ https://cctld.by/recorder/

Популярные российские регистраторы доменов:

- Rucenter (nic.ru), Рег.ру (Reg.ru), Timeweb (timeweb.com), Бегет (beget.com), Макхост (mchost.ru), Спринтхост (sprinthost.ru) и другие.

Популярные зарубежные регистраторы доменов:

- GoDaddy (godaddy.com, США), Hostinger (hostinger.com, европейский), IONOS (ionos.com, Германия), Cloudways (cloudways.com, Мальта), Hawk Host (hawkhost.com, Канада).

Обладателями доменных имен могут быть физические лица и организации).

- Физические лица: могут регистрировать домены для личных сайтов и проектов.

- Организации: регистрируют домены для корпоративных сайтов, интернет-магазинов и т.д.

Регистрация доменного имени

Вначале необходимо выбрать для своего сайта доменное имя и зону (например, biblioklub.by) и убедиться, что оно не занято. Желательно чтобы в доменное имя входило ключевое слово темы ресурса, что положительно повлияет на продвижение в поисковых системах и нахождению сайта по данному запросу, например:

Сайт по продаже моноблоков использует в домене ключевое слово «моноблок» – (monoblok.by), интернет-магазин строительных материалов СтройКа – «стройка» (stroi-ka.by); чистка ковров в Гомеле – «чистка ковров» (chistka-kovrov-gomel.myinfo.by), синоним чистки – «отраим» (otdraim.by) и др.

Известные сервисы включают названия в домены своих брендов, например: Куфар (kufar.by), 5 элемент (5element.by), КСК (ksk.by), кондитерская фабрика «Спартак» (spartak.by) и т.д.

Организации в основном используют в домене аббревиатуру: ГГТУ им. П.О.Сухого (gstu.by), КПУП «Гомельское городское ЖКХ» (orjkh-gomel.by), газета «Гомельская правда» (gp.by) и т.д.

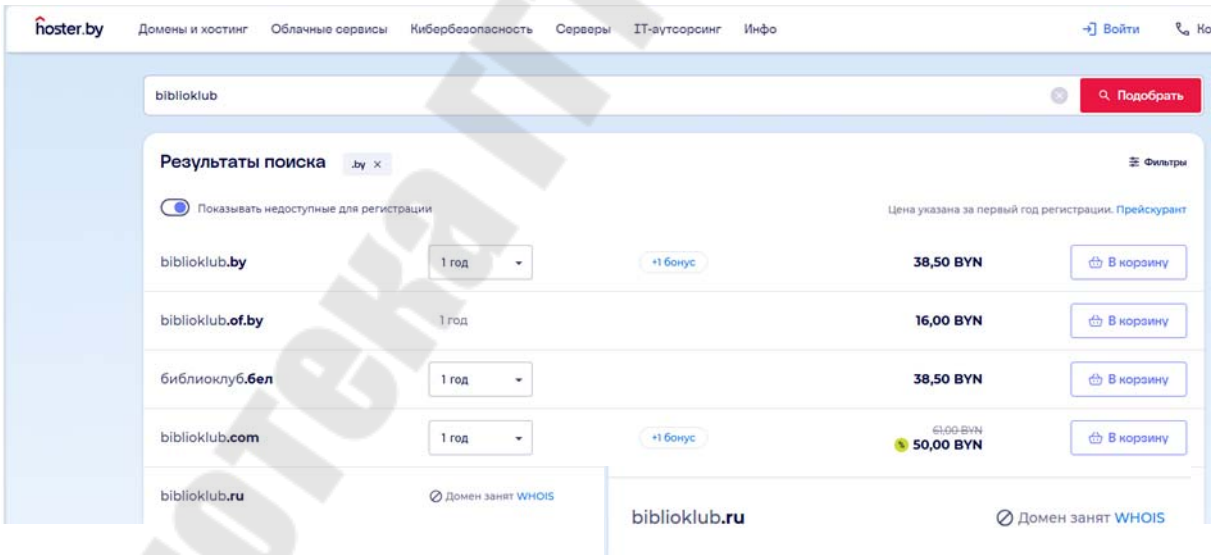

Рис. 70 – Подбор доменного имени на hoster.by

На любом регистраторе доменных имен в разделе «Домены» можно посмотреть, доступен ли выбранный домен или занят. Далее выбирается регистратор доменных имен, в которой оплачивается домен.

В данном примере доменное имя biblioklub.by – свободно, а в российском сегменте занято. Данную информацию можно проверить через сервис WHOS. Данный раздел находится у всех регистраторов доменных имен.

WHOIS (от англ. who is – «кто это?») — сетевой протокол прикладного уровня. Основное применение — получение регистрационных данных о владельцах доменных имён, IP-адресов и автономных систем. Протокол подразумевает архитектуру «клиент-сервер» и используется для доступа к публичным серверам баз данных (БД) регистраторов IP-адресов и регистраторов доменных имён.

Пример проверки регистрации домена на сервисе WHOIS:

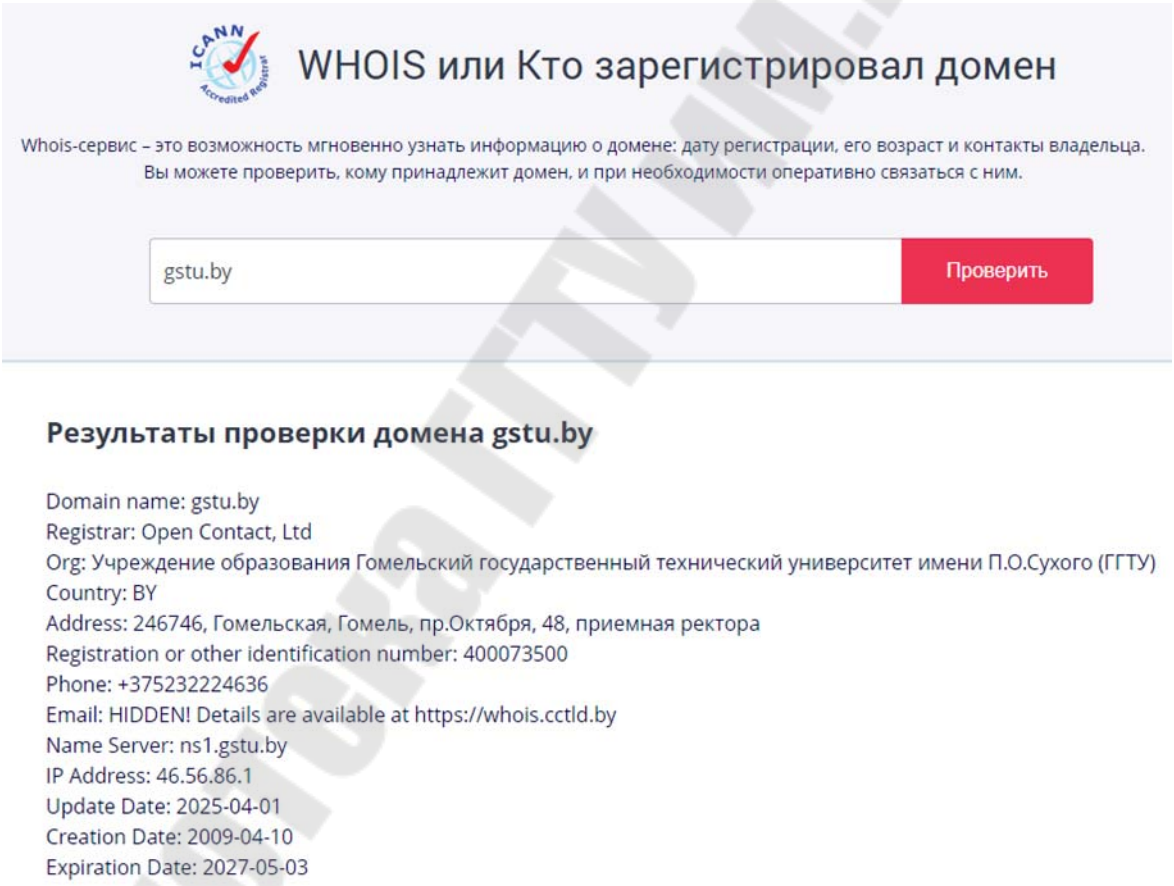

Рис. 71 – Пример работы WHOIS

В данном примере получены данные о домене gstu.by через WHOIS whois.hoster.by: регистратор домена, владелец, адрес, DNS-серверы, дата покупки и дата окончания.

Управление доменными именами

После регистрации и покупке домена, владельцу доступно:

- Изменение данных о домене – владельцы могут изменять контактную информацию и настройки домена через панель управления регистратора.
- Перенос домена – Процесс передачи доменного имени от одного регистратора к другому, который может потребовать подтверждения со стороны текущего владельца.
- Поддержка и обслуживание доменных имен – регистраторы предлагают услуги по управлению доменными именами, включая техническую поддержку, обновление данных и помощь при переносе.

Правильное управление доменными именами важно для обеспечения доступности ресурсов и защиты бренда.

Подключение домена к хостингу

Чтобы сайт заработал, нужно соединить домен и хостинг. Домен и хостинг можно приобрести на одном сервисе, либо на разных.

Общий принцип подключения домена к хостингу:

1. Выбор хостинга: необходимо выбрать хостинг и зарегистрироваться в нем.
2. Получение DNS-серверов (DNS/Domain Name System – это система, которая переводит удобные для человека доменные в IP-адреса, необходимые компьютерам для обмена данными в интернете).

После регистрации хостинг предоставляет информацию о DNS-серверах выбранного хостинга. Данные доступны в письме клиенту и в личном кабинете:

Пример DNS сайтов:

Таблица 6 – Примеры DNS сайтов

DNS hoster.by	DNS beget.com	DNS gstu.by
ns1.datacenter.by ns2. datacenter.by	ns1.beget.com ns2.beget.com	ns1.gstu.by собственный сервер

3. Внесение DNS-записей хостинга у регистратора домена:

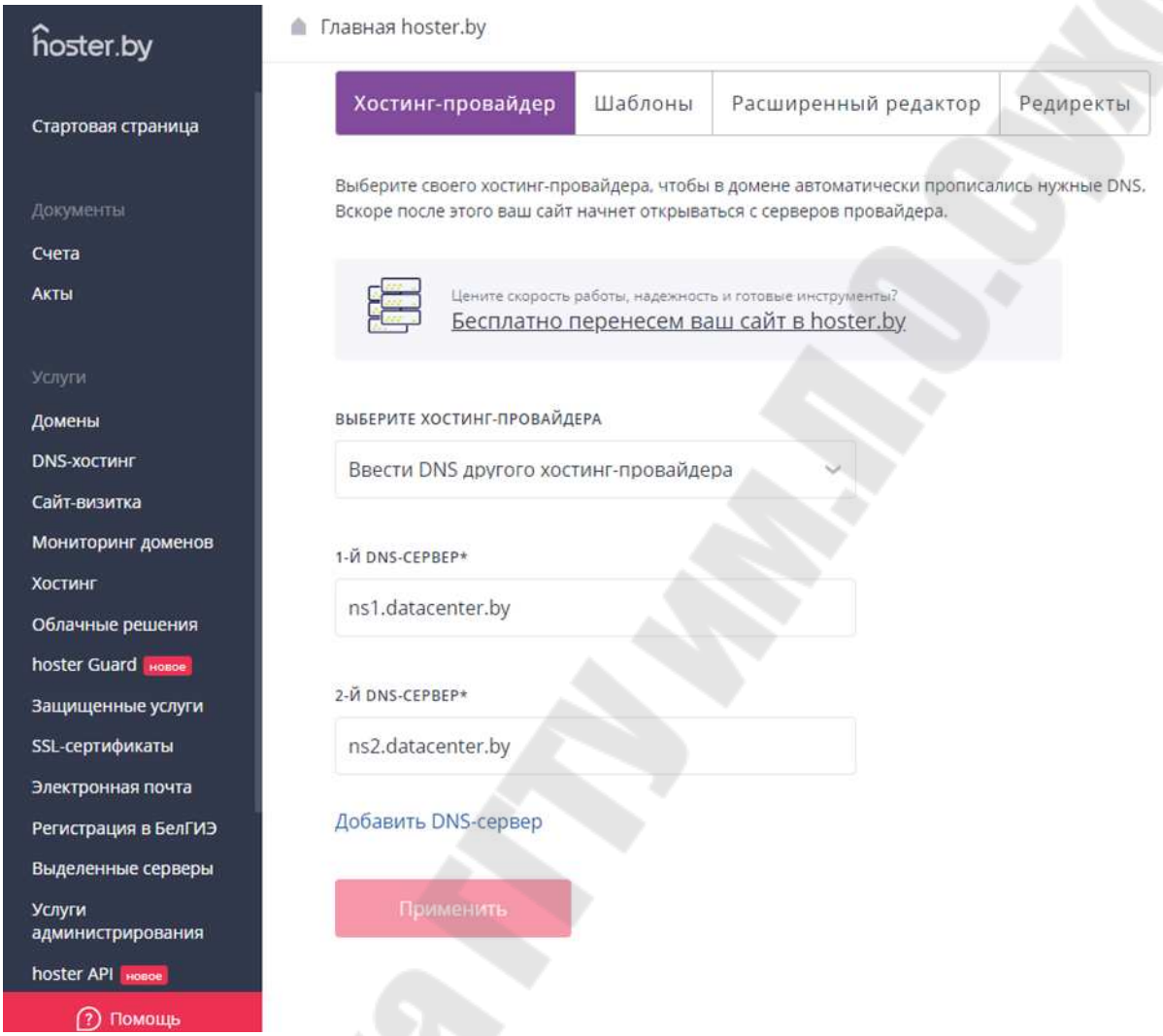

Рис. 72 – Ввод данных DNS в личном кабинете

Порядок действий:

- Необходимо войти в панель управления регистратора домена (где приобретался домен).
- Найти раздел управления DNS или настройки домена.
- Ввести DNS-серверы, предоставленные хостингом.
- Сохранить изменения.

4. Настройка сайта на хостинге: В раздел «Добавить домен» или «Управление доменами» необходимо добавить свой домен и далее следовать инструкциям для его привязки к аккаунту.

5. Загрузка файлов сайта на хостинг: для загрузки файлов сайта на хостинг используется FTP-клиент или файловый менеджер в соответствующую директорию (обычно это папка public_html или www).

В данном примере домен и хостинг приобретены у разных организаций. В аккаунте регистратора доменного имени hoster.by в разделе «Домены» прописаны данные DNS хостинга Белтелекома.

6. Проверка работы: обычно сайт становится доступным всего через несколько минут. Необходимо открыть приобретенный домен в браузере, чтобы убедиться, что он загружается корректно. В течение трех дней сайт будет доступен из любой точки мира.

Регистрация сайта в БелГИЭ

Если планируется создать сайт, который будет предоставлять услуги или информацию, связанную с коммерческой деятельностью, его необходимо зарегистрировать в БелГИЭ.

РУП «БелГИЭ» (belgie.by) – государственная инспекция Республики Беларусь по электросвязи Министерства связи и информатизации Республики Беларусь.

Регистрировать сайт необходимо индивидуальным предпринимателям, юридическим лицам, их филиалам и представительствам, созданных в соответствии с законодательством Республики Беларусь, с местонахождением в Республике Беларусь осуществляющие деятельность по реализации товаров, выполнению работ, оказанию услуг на территории Республики Беларусь.

Регистрация интернет-ресурса утверждена законодательно на основании Постановления Совмина № 644 от 29.04.2010. В Постановлении указано, что все сайты, созданные в РБ, должны быть зарегистрированы в государственном регистре информационных сетей, систем и ресурсов национального сегмента сети Интернет.

Процедура регистрации единоразовая. Данной процедурой обычно занимается хостинг, на котором расположен сайт. Также возможна самостоятельная регистрация.

Регистрация позволяет гарантировать, что сайт соответствует стандартам безопасности и защиты прав пользователей, а также помогает обеспечить соответствие требованиям белорусского законодательства в области информационных технологий и защиты информации.

Перспективы развития пространств доменных имен

Вот уже более десяти лет в белорусском сегменте интернета доступна возможность регистрации доменов на кириллице в доменной зоне «.бел». Это позволяет пользователям создавать сайты, ориентированные на отечественную аудиторию, что способствует лучшему восприятию и удобству использования ресурсов для пользователей, говорящих на русском и белорусском языках.

Домен **.БЕЛ** был делегирован Республике Беларусь 12 февраля 2015 года. За 10 лет работы доменной зоны .БЕЛ зарегистрировано около 15000 доменов второго уровня.

Пример доменов на кириллице:

- СМАЙЛ.Бел (смайл.бел) – техника для дома.
- Трава.бел (трава.бел) - доставка цветов.
- ЗАО «МТБ банк» (халвамтбанк.бел) и др.

Использование кириллических доменов способствует развитию локального контента и упрощает навигацию по сайтам для пользователей, предпочитающих кириллицу.

Домен **.BY** – национальный домен верхнего уровня для Республики Беларусь. Регистрация доменов в Беларуси была разрешена с 1994 года, а свободная регистрация доменов второго уровня для предприятий открылась в апреле 2000 г.

Техническим администратором BY с 2022 года является ООО «Белорусские облачные технологии».

С развитием интернет-технологий все больше появляется дополнительных доменных зон, например: универсальные (.online, .top, .best, .store, .help и др.), географические (.moscow, .city, .center и др.), отели/кофе/рестораны (.beer, .cafe, .food и др.), красота (.moda, .yoga, .spa и др.), образование и обучение (.courses, .school, .info, .academy и др.) и другие.

Подбор доменного имени теперь можно осуществлять не только с помощью простого ввода желаемого домена, но и с использованием функций искусственного интеллекта (AI, ИИ), что значительно упрощает процесс и помогает находить более креативные и подходящие варианты.

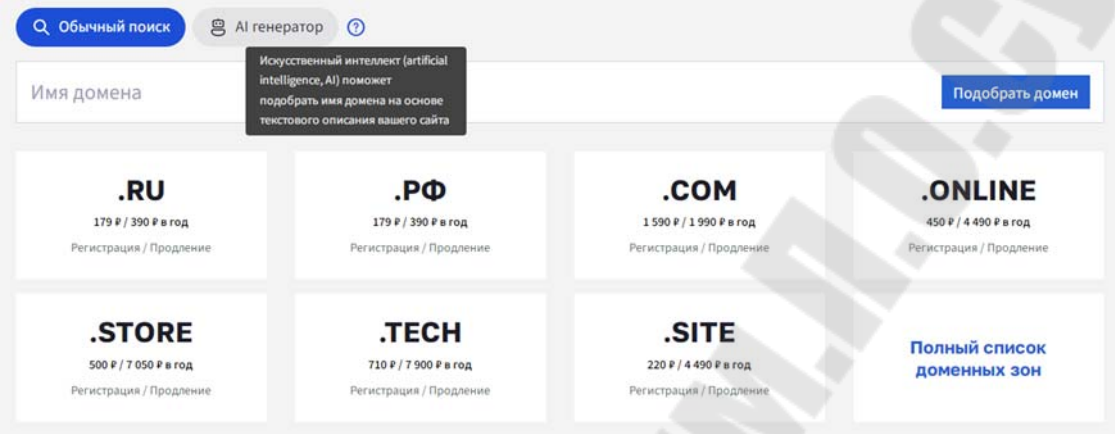

Рис. 73 – Подбор домена с помощью AI-генератора

С ростом новых технологий, пространство доменных имен будет продолжать развиваться, чтобы справляться с новыми вызовами и потребностями пользователей.

8.2 Хостинг

Хостинг — это услуга, позволяющая размещать веб-сайты и приложения на серверах, доступных в Интернете, тем самым обеспечивая их доступность пользователям. Хостинг играет ключевую роль в разработке и функционировании веб-приложений и сайтов, предоставляя необходимые ресурсы (хранилище, вычислительная мощность, доступ к сети) для их работы и взаимодействия с пользователями.

Сайт представляет собой совокупность текстов, видеофайлов, изображений, кода, баз данных и другой информации. Для того чтобы пользователи могли получить доступ к этим ресурсам, сайт должен быть размещен на компьютере с специальным программным обеспечением и постоянным подключением к интернету. Такой компьютер называется сервером. Хостинг-компания располагает достаточным количеством серверов для обслуживания большого числа клиентов.

Физически серверы располагаются на специальных стойках в отдельном помещении. Оно называется дата-центром, центром обра-

ботки данных (ЦОД) или центром хранения и обработки данных (ЦХОД). В нём обеспечивают бесперебойную подачу электричества, связь, обновление оборудования, безопасность. У хостера может быть свой ЦОД. Но чаще он арендует у дата-центра его серверы или место под размещение своих серверов.

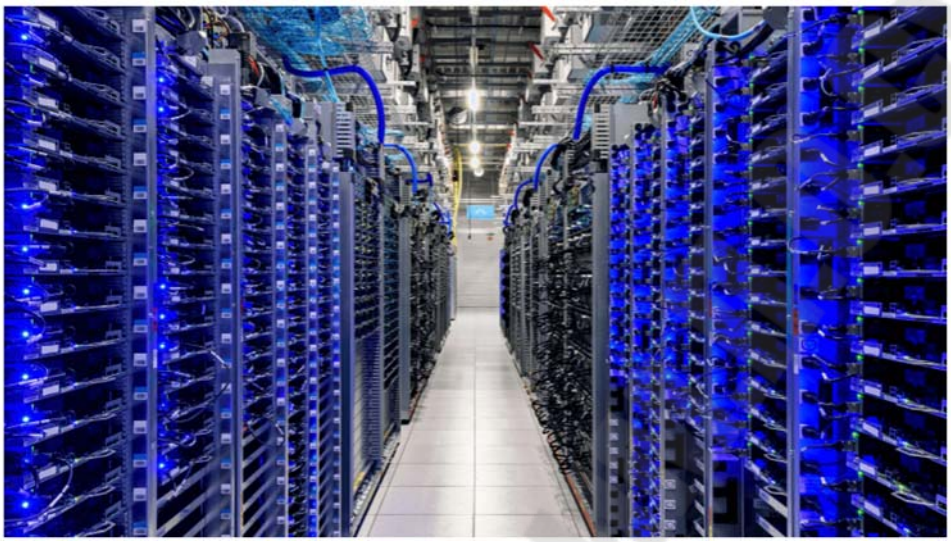

Рис. 74 – Дата-центр Google

Пример белорусских центров обработки данных:

- ООО «Белорусские облачные технологии» becloud.by - провайдер облачных решений на базе собственного дата-центра.
- РУП «Белтелеком» datacenter.by - оператор Республики Беларусь по предоставлению услуг электросвязи, центр обработки данных.

Дата-центр может располагаться в любой точке мира. Это важно учитывать при выборе хостинга.

Основные концепции хостинга:

- Доступность: услуги хостинга обеспечивают круглосуточный доступ к веб-ресурсам.
- Ресурсы: хостинг предлагает различные ресурсы, такие как процессоры, память, дисковое пространство и пропускная способность сети.
- Серверы: веб-сайты размещаются на физических или виртуальных серверах, которые обрабатывают запросы пользователей.

Как работает хостинг: хостинг включает в себя размещение файлов веб-сайта на сервере. Когда пользователь вводит адрес сай-

та в браузере, сервер обрабатывает запрос и отправляет данные о веб-странице на устройство пользователя.

Типы хостинга

В зависимости от задачи компания может выбрать один из четырёх видов хостинга: виртуальный хостинг, виртуальный сервер, выделенный хостинг или облачный сервер.

Виртуальный (веб-хостинг) – ресурсы сервера разделяются между несколькими веб-сайтами, при этом администрирование сервера хостер берёт на себя. Это наиболее экономичный вариант для малых и средних сайтов.

Специальное программное обеспечение следит за тем, чтобы мощности машины распределялись между клиентами более-менее равномерно, однако это происходит с некоторыми ограничениями.

Преимущества: лёгкий старт и простота управления; низкая стоимость.

Недостатки:

- Ограничены контроль и функциональность. Сервер предоставляют «как есть» и настраивают для всех клиентов одинаково. Нельзя, например, поставить дополнительное программное обеспечение или изменить файлы конфигурации.

- Ограничены объём и скорость передачи данных между пользователями и вашим сайтом.

- Могут быть проблемы с производительностью. Ресурсы сервера общие, клиенты используют их одновременно. Поэтому сайт может начать виснуть из-за того, что в какой-то момент ему достанется мало оперативной памяти.

- Общий IP. На виртуальном хостинге под одним IP-адресом могут находиться сотни сайтов. Если один из них попадает в черный список (фильтр поисковых систем, блокировка), это может повлиять на работу и других сайтов, размещенных на том же IP.

Выделенный виртуальный сервер (VPS) – виртуальный сервер, который делит физические ресурсы с другими, но предлагает больше контроля и индивидуальные настройки, чем виртуальный хостинг.

На одном физическом сервере запускают изолированные друг от друга процессы, которые имитируют работу самостоятельного компьютера. Виртуальные серверы обозначают аббревиатурами VPS (virtual private server) и VDS (virtual dedicated server). Технические различия между ними есть, но это несущественно для большинства пользователей. Клиент получает те же привилегии, что и при работе с выделенным сервером. Это и установка необходимой операционной системы, и своё программное обеспечение. Виртуальные серверы не влияют на работу друг друга.

Преимущества: защищённая среда; контроль и возможность тонкой настройки; стоимость ниже, чем у выделенного сервера; скорость работы выше, чем у виртуального хостинга.

Недостатки: дороже виртуального хостинга; для администрирования необходимы знания.

Выделенный сервер (хостинг) — пользователь получает полный контроль над одним сервером, который не делится с другими. Это идеальный вариант для крупных проектов с высокими требованиями к ресурсам.

Пример распределения ресурсов между сайтами на разных типах (тарифах) хостинга.

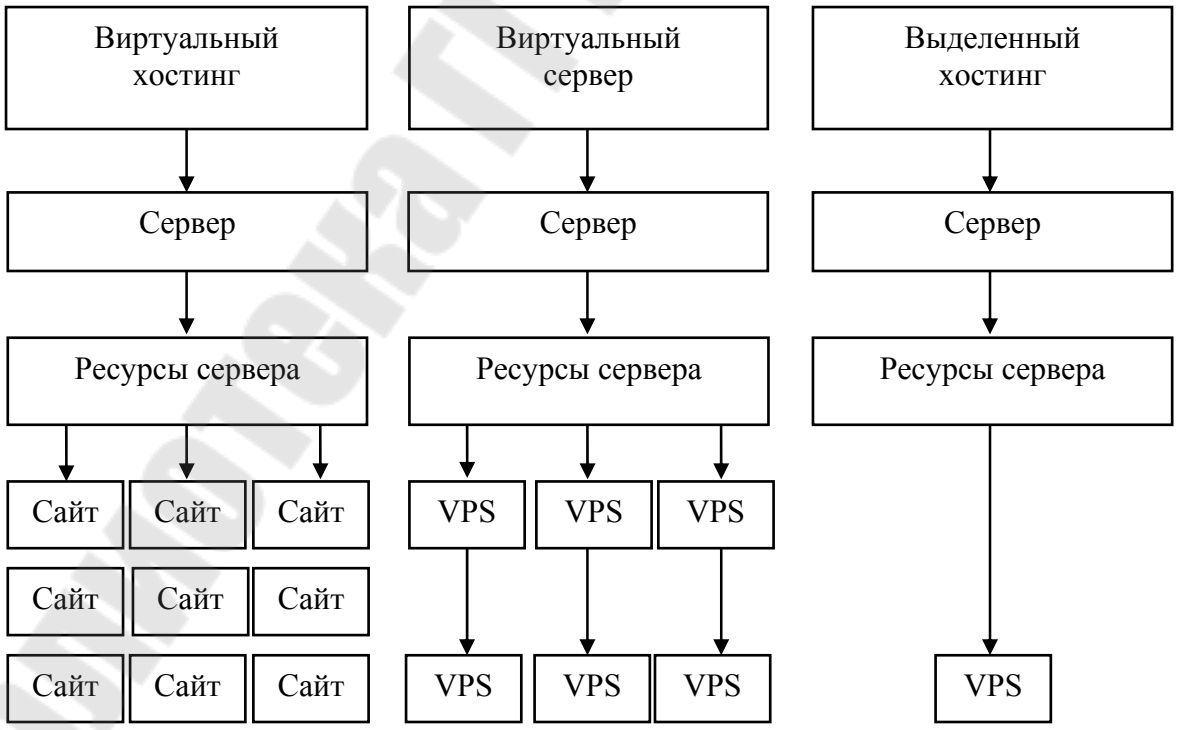

Рис. 75 – Типы хостинга

Преимущества выделенного хостинга: высокая производительность — на сервере проект только одного клиента.

Недостатки: высокая стоимость — чем мощнее сервер, тем выше цена; для поддержки хостинга нужна высокая квалификация и большие затраты.

Облачный хостинг сайта - это система из группы серверов, которые страхуют друг друга. Если один из них отключится, другие перераспределят между собой нагрузку, и сайт продолжит работать. Подходит ресурсам с посещаемостью около 10 тысяч пользователей в сутки.

Такой вид серверов удобно масштабировать. Если компании нужна дополнительная оперативная память или дисковое пространство, их можно быстро отрегулировать в личном кабинете хоста, а также быстро отключить, чтобы не платить за ресурсы, которые простаивают.

У такого варианта аренды сервера есть два минуса:

- Облачные серверы сложнее настроить, поэтому для работы с ними нужен опытный администратор. Это связано с тем, что хостер чаще всего сам устанавливает операционную систему, и некоторые функции могут быть недоступны.
- Сложно следить за безопасностью данных — они находятся на разных серверах. Для того чтобы выстроить защиту, нужно обратиться к специалистам хостинга, либо настроить ее самостоятельно, но для этого требуются специализированные знания.

Цена облачной технологии выше, чем у виртуального хостинга, но ниже, чем у выделенного реального. При этом по набору функций облачный хостинг ближе именно к собственному серверу.

Собственный хостинг — это решение, которое некоторые организации выбирают для управления своими веб-ресурсами и данными. Вместо использования услуг сторонних хостинг-провайдеров, компании могут создать собственный сервер или группу серверов, установив на них специальное программное обеспечение для хостинга. Это позволяет им иметь полный контроль над инфраструктурой, настройками безопасности и производительностью.

Создание собственного хостинга может быть особенно выгодным для организаций с высокими требованиями к безопасности и конфиденциальности данных, таких как финансовые учреждения или

компании, работающие с чувствительной информацией. Кроме того, собственный хостинг может обеспечить большую гибкость в настройках и масштабируемости, позволяя адаптировать ресурсы под конкретные нужды бизнеса.

Однако стоит отметить, что создание и поддержка собственного хостинга требует значительных технических знаний и ресурсов. Организациям необходимо учитывать затраты на оборудование, программное обеспечение, а также на техническую поддержку и обслуживание серверов.

Популярные хостинги

Провайдеры хостинга предоставляют необходимые ресурсы и поддержку для размещения веб-сайтов, а также управление серверами.

Популярные провайдеры хостинга в Беларуси: Хостер бай (hoster.by), ActiveCloud (activecloud.by), Белтелком (beltelecom.by) и др.

Популярные провайдеры хостинга в России: Спринтхост (sprinthost.ru), Бегет (beget.com), RU-CENTER (nic.ru), Макхост (mchost.ru), Рег.ру (reg.ru), Timeweb (timeweb.com).

Зарубежные провайдеры хостинга: Bluehost, SiteGround, HostGator, DigitalOcean, AWS (Amazon Web Services) и др.

Критерии выбора хостинга

Чтобы выбрать хостинг, нужно оценить свои потребности, подобрать несколько компаний и сравнить их.

Оценка потребностей нужна, чтобы понять, какой тип сервера подойдёт для проекта. Если сайт простой или лендинг, то есть то, что не требует особых ресурсов для работы, — подойдет виртуальный хостинг.

Если проект предполагает массивные нагрузки и большой ресурс железа, а также нестандартный софт, то стоит обратить внимание на виртуальный сервер.

Облачный хостинг хорош для клиентов, которым критически важна постоянная работа сайта. Например, для интернет-магазинов. Выделенные серверы часто используют крупные компании, которым

нужен максимум ресурса сервера. Также на них часто размещают крупные игровые проекты.

Подобрать хостинг можно в поисковой выдаче по запросу «рейтинг хостингов», «лучшие хостинги для сайта» и т.д. или на специальных сайтах с рейтингами.

Сравнивать хостинги можно по таким параметрам как: скорость, надежность - время без простоев, цена, местоположение хостера, качество работы службы поддержки.

На самом хостинге необходимо выбрать подходящий тариф для сайта, который включает:

- объем дискового пространства: необходимое место для хранения файлов сайта;
- количество баз MySQL;
- пропускная способность (трафик): сколько данных может передаваться между вашим сайтом и пользователями;
- цена: сколько стоит хостинг в месяц или год (аренда на год стоит обычно дешевле);
- дополнительные услуги: наличие дополнительных услуг, таких как резервное копирование, защита от DDoS-атак, установка защищенного соединения, различные настройки.

Чтобы облегчить выбор тарифа многие хостеры предлагают CMS-хостинг (оптимизация раздела хостинга под конкретную CMS) либо конструктор сайтов.

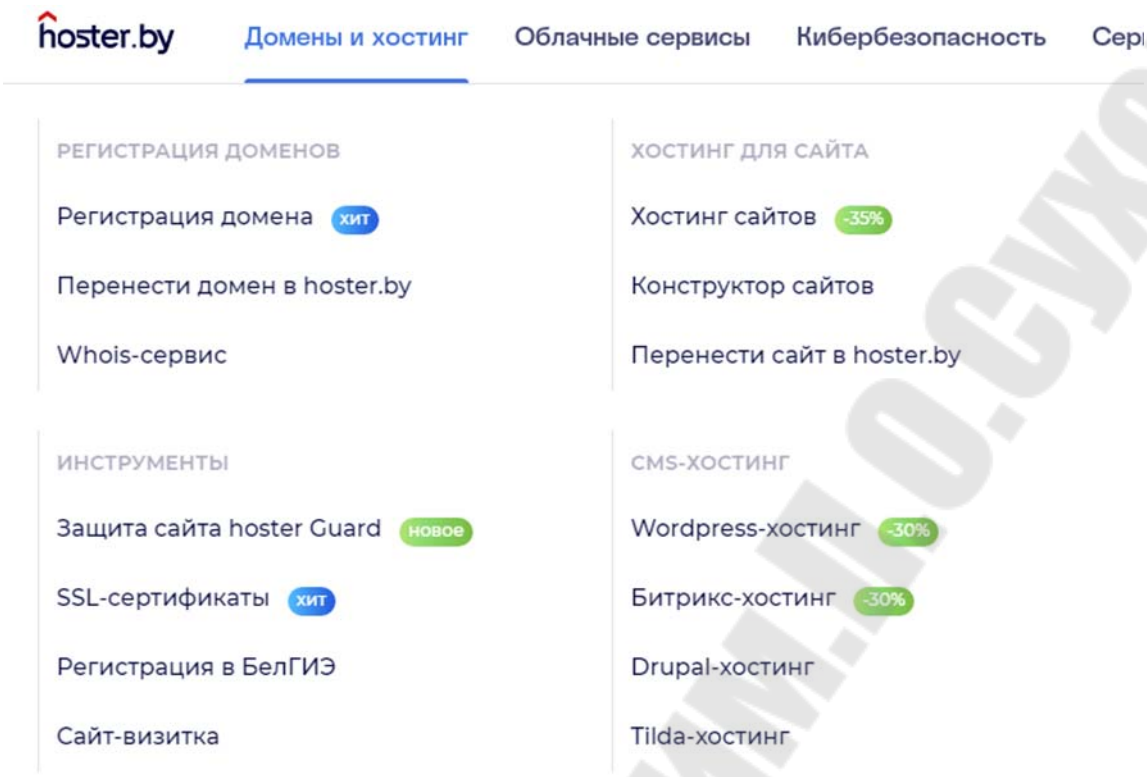

Рис. 76 – Хостинг для популярных CMS на hoster.by

1	Предоставление дискового пространства в разрезе пакетов, в месяц :
1.1	"Хостинг Старт"
	- объем дискового пространства - 2 ГБ
	- 10 почтовых ящиков
	- 3 базы данных MySQL
1.2	"Хостинг 10"
	- объем дискового пространства - 10 ГБ
	- 20 почтовых ящиков
	- 10 базы данных MySQL

Рис. 77 – Выбор дискового пространства на серверах, принадлежащих РУП «Белтелеком»

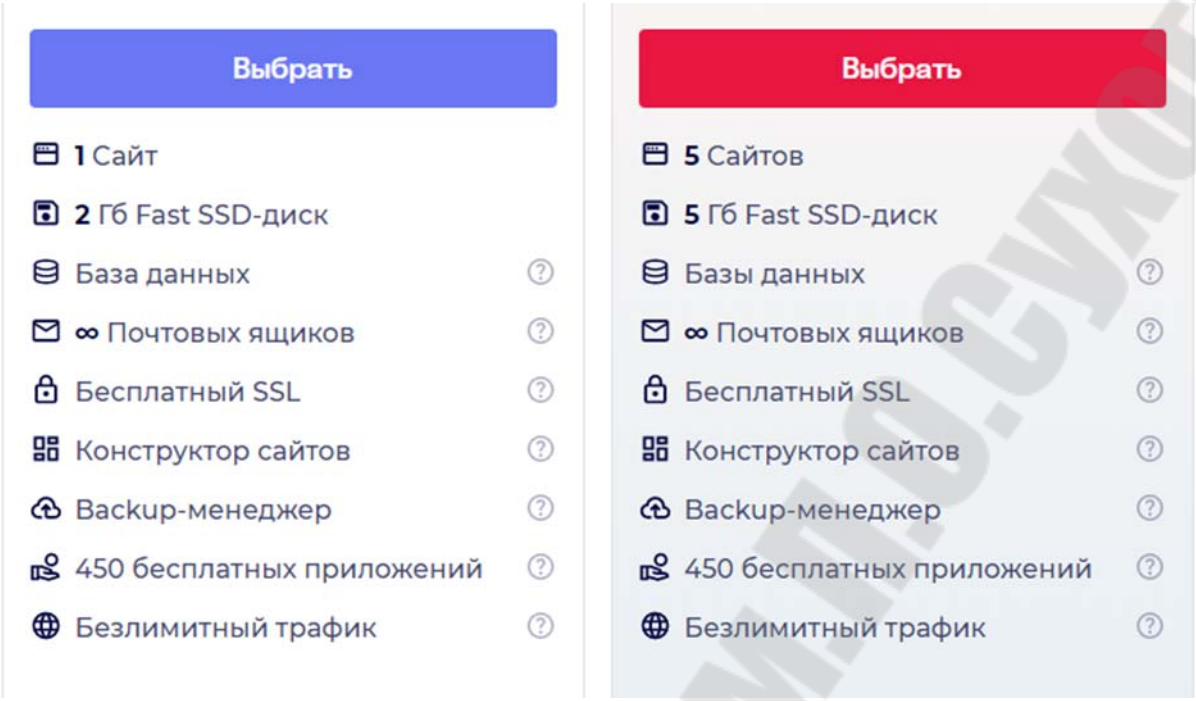

Рис. 78 – Пример технических характеристик тарифов хостинга на hoster.by

Регистрация и настройка хостинга

Этапы регистрации хостинга:

1. Регистрация на сайте хостинга – необходимо выбрать хостера и пройти процесс регистрации.

2. Выбрать тариф и оплатить хостинг – оплата хостинга доступна различными способами, включая банковские карты и систему ЕРИП.

3. Получение данных доступа – после оплаты хостинга на указанную почту придет письмо с ссылкой на панель управления хостингом, пароль и логин для входа, данные DNS, контактные данные технической поддержки.

4. Настройка домена и хостинга – необходимо настроить DNS записи для правильной работы домена с хостингом.

5. Запуск сайта - перенос сайта с локального хостинга на полноценный сервер, либо развертывание нового сайта на хостинге с использованием предлагаемых CMS.

Управление хостингом

Прежде всего, услуга хостинга – это сдача в аренду части ресурсов сервера, пространства для запуска сайта клиента или веб-приложения. Панель управления хостингом может быть установлена на выделенном сервере или на виртуальной машине (VDS/VPS). Как правило, панель содержит наборы инструментов с базовым и расширенным функционалом – для управления доменными именами и создания субдоменов, пользователей, адресов e-mail, сайтов, баз данных, резервных копий файлов и т.д. Кроме того, административная панель интегрируется с другим специализированным ПО, которое устанавливается на веб-сервере и обеспечивает работу сайта и приложений.

Панели управления упрощают процесс управления хостингом, предоставляя инструменты для редактирования файлов, настройки баз данных и управления сайтами.

Популярные панели управления хостингом:

- cPanel: одна из самых распространенных панелей управления, предлагающая интуитивно понятный интерфейс для управления веб-хостингом, включая управление доменами, базами данных, почтовыми ящиками и файлами.
- Plesk: многофункциональная панель управления, поддерживающая как Windows, так и Linux-серверы. Она предлагает широкий спектр инструментов для управления веб-сайтами, приложениями и безопасностью.
- DirectAdmin: простая и легкая в использовании панель управления, которая предлагает основные функции для управления хостингом, включая управление доменами, базами данных и почтой.
- ISPmanager: удобная панель управления для веб-хостинга, поддерживающая Linux-серверы. Она предлагает интуитивно понятный интерфейс и набор инструментов для управления доменами, сайтами, базами данных и почтовыми ящиками. ISPmanager также включает функции для мониторинга ресурсов и настройки безопасности.

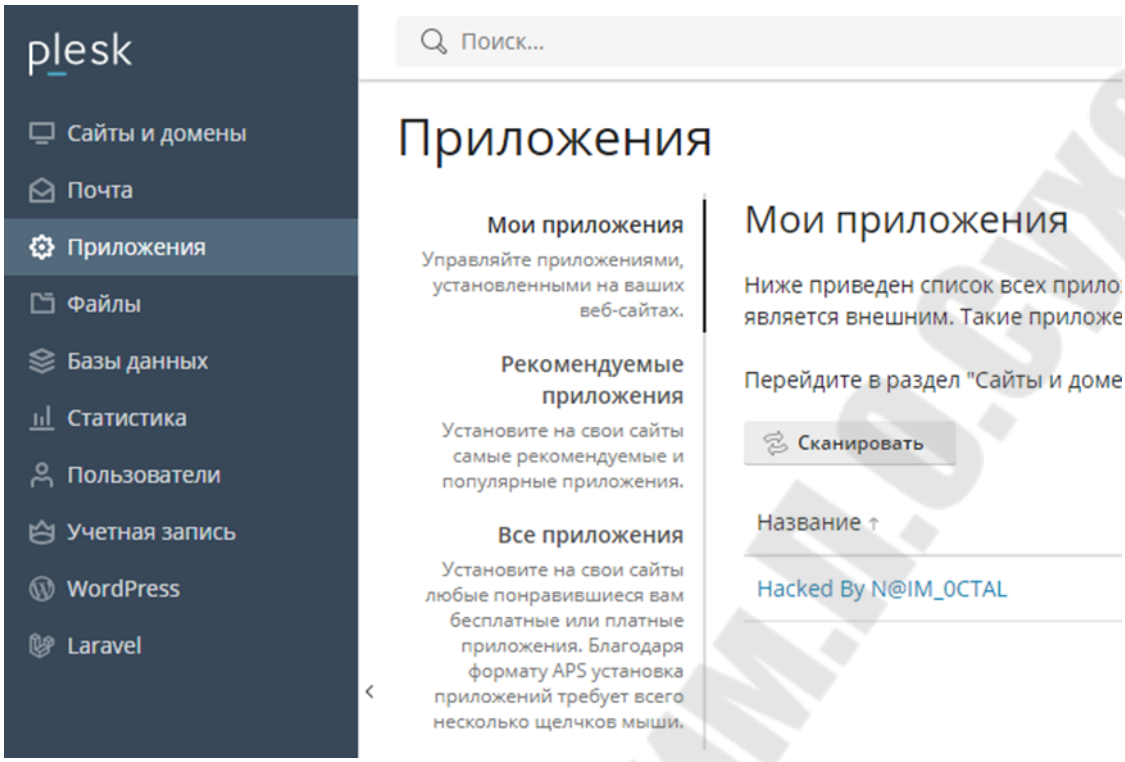

Рис. 79 – Панель управления хостингом Plesk (Белтелеком).

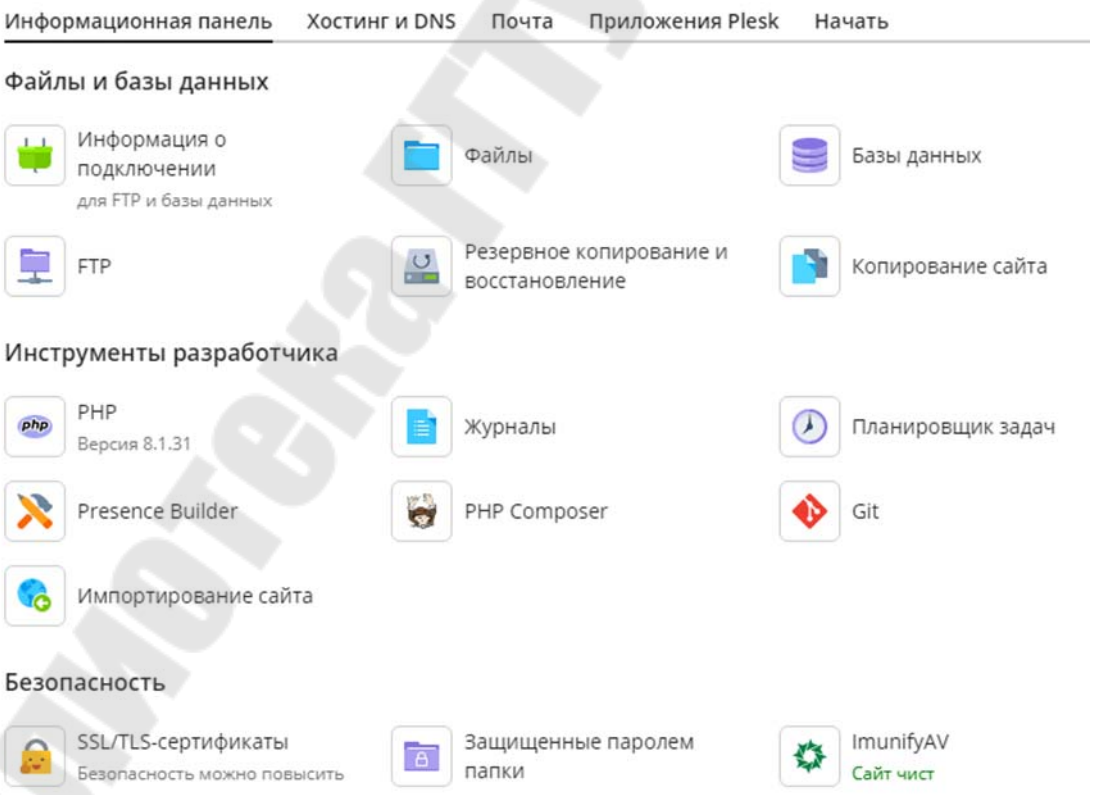

Рис. 80 – Фрагмент раздела панели управления хостингом Plesk.

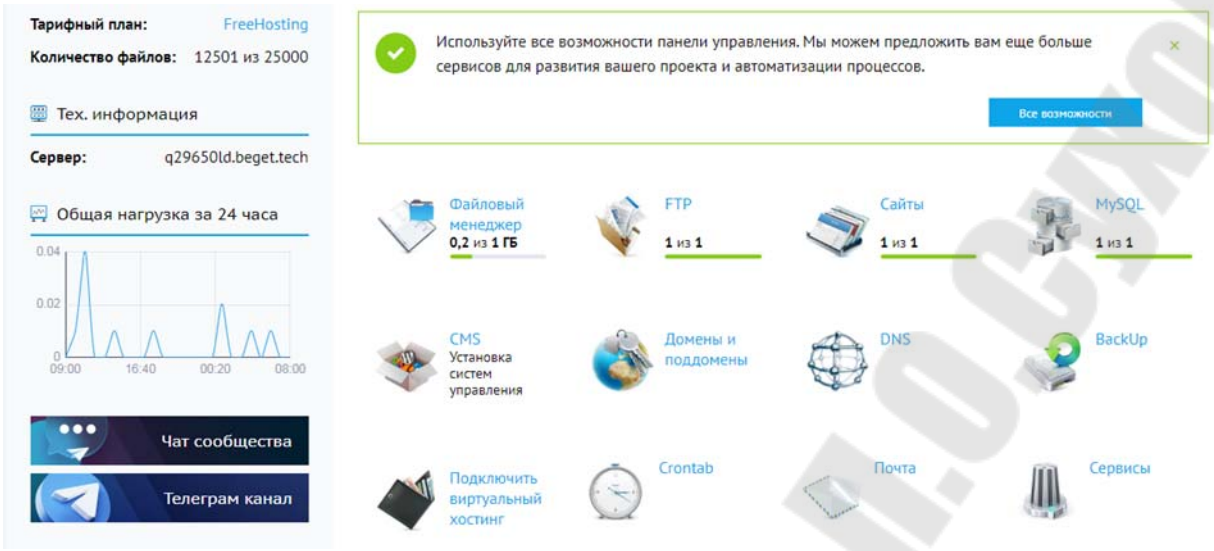

Рис. 81 – Фрагмент раздела панели управления хостингом (Бегет). Используется на платном и бесплатном тарифах

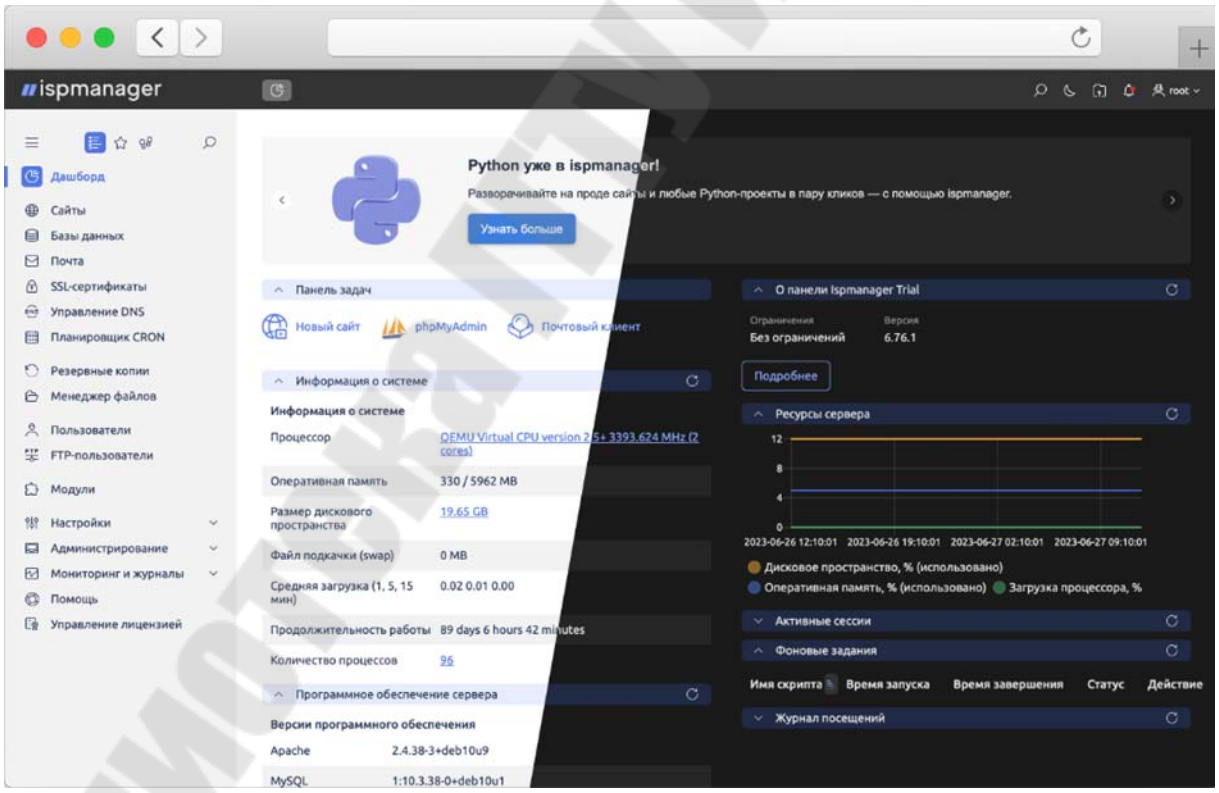

Рис. 82 – Панель управления хостингом Ispmanager.

Правильный выбор хостинга обеспечивает надежность работы веб-ресурсов, ускоряет загрузку и улучшает общую производительность сайтов.

Наиболее используемые инструменты в панели управления:

- Файловый менеджер – необходим для загрузки файлов сайта;
- FTP – программа для передачи файлов между локальным компьютером и сервером, позволяющая загружать и скачивать файлы, а также управлять ими на сервере.
- MySQL – система управления базами данных, используемая для хранения и обработки данных веб-приложений.
- Поддомены – дополнительные домены, которые создаются в рамках основного домена, позволяя организовать структуру сайта.
- SSL/TLS – криптографические сертификаты, которые обеспечивают безопасное соединение между веб-сервером и браузером пользователя. Они шифруют данные, передаваемые между клиентом и сервером, что защищает информацию от перехвата и атак. Наличие SSL/TLS-сертификата также повышает доверие пользователей к сайту и может положительно сказаться на его позициях в поисковых системах.
- CMS (Система управления контентом) – программное обеспечение, которое позволяет пользователям разворачивать предлагаемые CMS на хостинге (в рамках тарифа): Wordpress, Joomla, Laravel, Moodle и др..
- Crontab – инструмент для планирования выполнения задач на сервере в определенное время или с заданной периодичностью, что позволяет автоматизировать рутинные процессы.

Перед приобретением домена и хостинга – важно знать

- Доменное имя нужно периодически продлевать. Его можно продлить на год или больше. Если домен не оплачен, то сайт станет недоступен. И если не оплатить долг в указанный период – можно окончательно потерять домен.
- Хостинг можно оплатить как на несколько месяцев, так и на несколько лет, а также продлить его на выбранный период. При неоплате хостинга сайт будет недоступен.
- Необходимо выбирать надежного провайдера домена и хостинга.

Нахождение сайта на территории РБ

Под нахождением сайта на территории Беларуси предполагается, что физически сервер, на котором расположен данный интернет-ресурс, находится на территории РБ (Указ № 60 «О мерах по совершенствованию использования национального сегмента сети Интернет»).

В Беларуси существуют определенные законодательные требования, которые обязывают хостинг-провайдеров размещать свои серверы на территории страны. Основные причины и законы, касающиеся этого вопроса, включают:

1. Законодательство о защите информации: В соответствии с Законом Республики Беларусь «О защите персональных данных» (от 2016 года) обработка и хранение персональных данных граждан Беларуси должны осуществляться с соблюдением определенных требований. Это включает в себя необходимость хранения таких данных на территории страны.

2. Законодательство о связи: Закон «О связи» (от 2005 года) устанавливает правила для операторов связи и провайдеров интернет-услуг, включая требования к размещению оборудования и серверов.

3. Законодательство о СМИ: В соответствии с Законом «О средствах массовой информации» (от 1995 года) и другими нормативными актами, интернет-ресурсы, которые могут быть отнесены к СМИ, также подлежат регистрации и должны соблюдать местные законы.

4. Контроль за контентом: Размещение хостинга на территории Беларуси позволяет государственным органам легче контролировать контент, размещаемый на сайтах, что связано с обеспечением безопасности и соблюдением законодательства.

5. Соблюдение норм по кибербезопасности: В соответствии с законодательством о кибербезопасности, размещение серверов в стране позволяет лучше защищать информацию от внешних угроз и атак.

Таким образом, размещение белорусского хостинга на территории Республики Беларусь является обязательным требованием для соблюдения законодательства в области защиты данных, связи и обеспечения безопасности информации.

Перспективы развития хостинга и новые технологии

Перспективы развития хостинга и новые технологии продолжают изменять рынок веб-услуг. Ключевые направления и технологий, которые могут оказать значительное влияние на будущее хостинга:

- Облачный хостинг – облачные технологии становятся все более популярными благодаря своей гибкости и масштабируемости. Облачный хостинг позволяет пользователям легко увеличивать или уменьшать ресурсы в зависимости от потребностей, что делает его идеальным для динамичных бизнесов.

- Искусственный интеллект и машинное обучение – AI и ML могут использоваться для оптимизации работы серверов, автоматизации процессов управления хостингом, а также для улучшения безопасности путем выявления аномалий в трафике и предотвращения атак.

- Периферийные вычисления (Edge Computing) – с увеличением числа IoT-устройств (Internet of Things, или «Интернет вещей») и растущей потребностью в быстрой обработке данных, периферийные вычисления становится ключевым направлением в архитектуре обработки данных. Эта модель предполагает размещение вычислительных ресурсов и анализа данных ближе к источнику их генерации, а не в централизованных дата-центрах. Такой подход позволяет обрабатывать данные на «краю» сети, что значительно снижает задержки и повышает производительность приложений.

- Улучшенная безопасность – с ростом киберугроз хостинг-провайдеры будут продолжать внедрять новые технологии безопасности, такие как многофакторная аутентификация, автоматизированные системы обнаружения вторжений и расширенные SSL/TLS-сертификаты.

- Виртуализация – виртуальные частные серверы (VPS) остаются популярными благодаря своей доступности и возможности изоляции ресурсов. Технологии виртуализации продолжают развиваться, предлагая более эффективные способы управления ресурсами.

- Автоматизация – автоматизация процессов развертывания, управления и мониторинга серверов становится стандартом в индустрии хостинга. Это позволяет снизить затраты на обслуживание и повысить эффективность работы.

- Использование SSD-накопителей – перспективы развития хостинга связаны с ростом требований к скорости и надежности сайтов. Использование SSD-накопителей становится ключевым фактором, обеспечивая значительно более быстрый доступ к данным по сравнению с HDD. Это позволяет повысить производительность и снизить время загрузки страниц, что важно для улучшения пользовательского опыта и SEO. В будущем ожидается расширение возможностей SSD, внедрение NVMe-технологий (протокол и интерфейс для быстрого обмена данными между компьютером и твердотельными накопителями (SSD)) и автоматизация управления ресурсами для более эффективного хостинга.

С ростом облачных технологий и услуг на основе ИИ будущее хостинга станет более динамичным и адаптивным, предоставляя пользователям новые возможности для развития.

9. ВНЕШНЯЯ ОПТИМИЗАЦИЯ САЙТА

После завершения технической оптимизации сайт регистрируют в поисковых системах и добавляют на сайт системы аналитики (счетчики). А также регистрируют сайт в специальных каталогах.

Сервисы для веб-мастеров от поисковых систем:

Таблица 7 – Сервисы поисковых систем для регистрации сайта

Сервисы	Гугл	Яндекс
Регистрация сайта	Центр вебмастеров - Google Search Console	ЯндексВебмастер
Метрика (счетчики)	Google Analytics	ЯндексМетрика
Каталог	Google My Business	Яндекс.Бизнес

Также, естественные ссылки на сайт с тематических площадок увеличивают позиции и доверие поисковых систем.

Дальнейшее развитие сайта

Дальнейшему развитию сайта способствуют:

- регулярное добавление свежих материалов и актуализация существующей информации;
- мониторинг информации о техническом и seo-состоянии сайта с помощью поисковых систем;
- поддержка обновлений сайта и сторонних расширений;
- создание бэкапов (копий) сайта;
- анализ пользовательского поведения и сбор обратной связи для улучшения удобства и функциональности;
- внедрение новых технологий и инструментов для повышения производительности и безопасности ресурса.

10. ОСНОВЫ БЕЗОПАСНОСТИ ПРИ РАЗРАБОТКЕ ВЕБ-САЙТОВ

Безопасность кода — это фундаментальная составляющая устойчивости цифровых продуктов. Из-за постоянного появления новых технологий, которые пишут обычные люди, неизбежны баги и уязвимости. Поэтому полностью защитить приложение от атак невозможно.

Также на пути данных от клиента к серверу и обратно находится большое количество протоколов — они тоже содержат уязвимости и могут быть источниками угроз. Поэтому задача разработчика — предусмотреть «дыры» в безопасности и написать код, устойчивый к большинству атак.

Атаки делятся на те, которые направлены на клиентскую часть, и те, которые направлены на серверную. Но это распределение условно, ведь есть атаки, которые направлены и на клиент, и на сервер. А некоторые атаки направлены на сервер, но реализуются на клиенте. Например, взлом базы данных, когда киберпреступник через поля ввода добирается до информации в базе данных.

Примеры атак:

MITM — группа атак с участием посредника, когда киберпреступник тайно вмешивается в обмен данными, чтобы получить или изменить информацию. К таким атакам относятся XSS-атаки и SQL-инъекции.

XSS-атаки — попытка захвата данных, когда киберпреступник внедряет собственный JavaScript-код в веб-приложение и затем использует в личных целях. Такую атаку используют, чтобы захватить учётные записи, украсть личные данные, выдать себя за пользователей или получить доступ к информации.

SQL-атака — атака на базу данных веб-приложения. При работе с базой разработчики используют SQL-запрос. Когда хакер добавляет к таким запросам вредоносный код, происходит SQL-атака, или SQL-инъекция.

CSRF-атака — подделка межсайтовых запросов. При атаке пользователя обманом заставляют делать в веб-приложении то, что он не хотел. Например, пользователя переводят на поддельный сайт, чтобы он перевёл деньги киберпреступнику или ввёл логин и пароль.

DoS и DDoS-атаки — попытка вывести веб-приложение из строя. Такие атаки происходят через настолько большой поток трафи-

ка, что веб-приложение не может его обработать и выходит из строя. DoS-атака организовывается с одного источника, а DDoS — с двух и более.

Атаки с загрузкой файлов — попытка получить информацию или нарушить работу веб-приложения, когда злоумышленник отправляет файл с вредоносным кодом через поле для загрузки.

Манипуляция URL — изменение URL в адресной строке браузера. Если веб-приложение не защищено, то киберпреступник добирается до приватной информации, например, личных данных или сведений о заказанных товарах. Ещё к этой атаке относится поиск страниц, скрытых от пользователей.

Оценка безопасность приложения

Оценить безопасность веб-приложения можно с помощью специальных сервисов и программ: OWASP ZAP, Arachni, Burp Suite или других. Они помогают срежиссировать разные атаки и узнать об уязвимостях при разработке или в готовом веб-приложении.

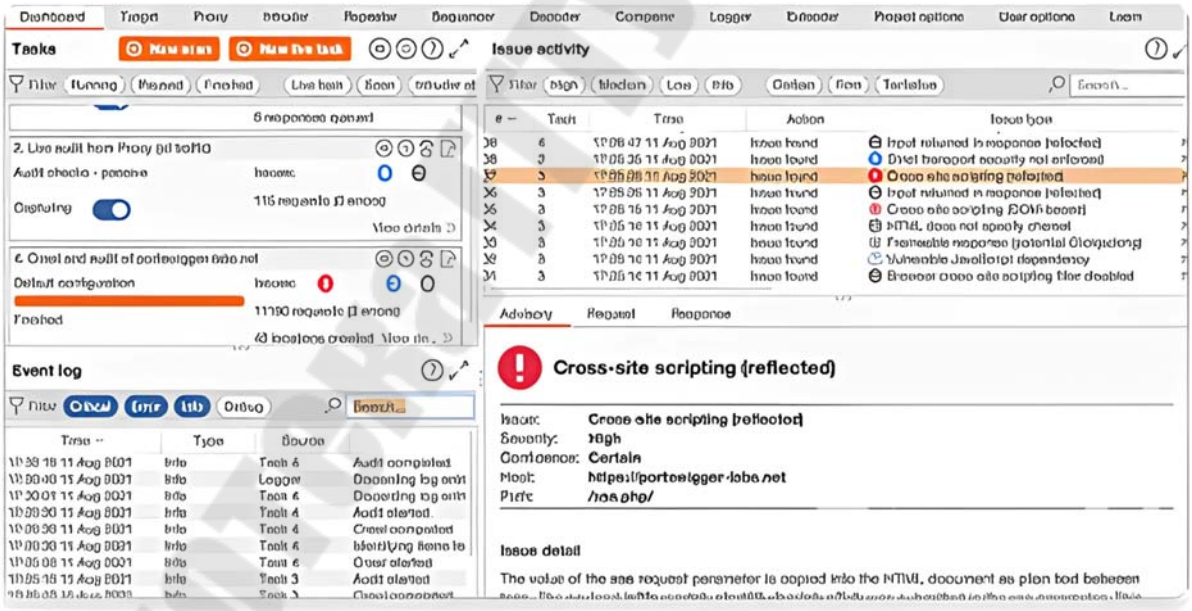

Рис. 83 – Проверка веб-приложения на уязвимость

На скриншоте показаны результаты тестирования в Burp Suite — найдено несколько уязвимостей, и в правом нижнем углу описано, как тестировали определённую атаку.

Такие сканеры проверяют приложение на все основные атаки, в том числе XSS, SQL-инъекции, CSRF и загрузку вредоносных файлов. После тестирования создается подробный отчёт об уязвимости веб-приложения — он пригодится при составлении карты задач для повышения безопасности.

Защита сайта на CMS

При разработке сайта на движке необходимо выбирать популярные CMS с частыми обновлениями — их разработчики постоянно работают над исправлением уязвимостей. Однако обновления могут не только закрывать старые слабые места в безопасности, но и обнажать новые.

Разработчики популярных CMS предупреждают пользователей об уязвимостях на официальных сайтах и в релизах CMS.

Рекомендации по защите сайтов на платформе CMS:

- Необходимо регулярно обновлять CMS и установленные расширения (темы/ шаблоны, плагины).
- Необходимо проводить регулярное создание копий сайтов (бэкапов) до и после обновлений CMS.
- Необходимо скачивать сторонние расширения для CMS только с официальных сайтов разработчиков. Загружая программы с ресурсов, распространяющих их незаконно, веб-мастер рискует столкнуться с вредоносным кодом, встроенным в шаблоны и расширения для бесплатного скачивания.
- Необходимо использовать минимальное количество сторонних расширений, особенно тех, у которых отсутствуют обновления, поскольку это увеличивает риск взлома.
- Можно изменить стандартную URL-ссылку для входа в админ-панель, чтобы скрыть ее от злоумышленников.
- Необходимо ограничить количество попыток входа в админ-панель сайта для предотвращения атак методом перебора паролей.
- Желательно на сайте использовать двухфакторную аутентификацию.
- Можно защитить базу данных при установке CMS путем выбора собственного префикса таблиц. Например, в wordpress по умолчанию задан префикс wp_, но его можно изменить свой, напрмиер: academy_ или ipk_.

- Необходимо на сайт установить системы аналитики. Встроенные системы аналитики могут сигнализировать владельцу о внешнем заражении сайта.

- На сайте необходимо подключить защищенное соединение HTTPS. Оно обеспечивает безопасность данных при передаче и защищает систему от несанкционированного доступа.

- Желательно установить на сайт специальные программы для защиты сайта от взлома и хакерских атак (например, Admin Tools от Akeeba).

Защита со стороны клиента

Рекомендации по защите сайтов со стороны клиента:

- Защита и валидация поля ввода – поля ввода очень уязвимы, ведь злоумышленники могут отправлять через них вредоносный код, поэтому нельзя разрешать пользователям вводить символы, которые запускают JavaScript или PHP: слеш, угловые скобки и знак вопроса. Нельзя давать возможность отправлять форму, если ни одно поле не заполнено.

- Создание системы с авторизацией – необходимо создавать разные уровни доступа, например, чтобы одни пользователи могли только читать страницы, другие — читать и редактировать, а третьи — редактировать код. Такое разделение защищает от SQL-инъекций, кражи cookies и других атак. Главное — не создавать «суперадминистратора» с максимальным количеством прав, иначе при взломе этого пользователя вся информация станет доступна киберпреступнику. Например, владельцу компании не нужны доступы к коду веб-приложения, а простому сотруднику — к бухгалтерским отчётам.

- Защита сайтов со стороны хостинга – подключение специальных приложений со стороны хостинга, которые защищают сайты, фильтруя входящий поток пользователей, и анализируют HTTP-трафик, блокируя ботов и выявляя аномалии трафика. Эти приложения нельзя использовать как единственный способ защиты: они не защищают от всех атак, так как есть множество способов их переобхода.

- Защита файлов при загрузке – разрешение пользователям загружать на сайт только необходимые типы файлов. В CMS данная функция уже предусмотрена, чтобы разрешить загрузку файла в на-

стройках необходимо установить разрешение или прописать расширение файла.

Рис. 84 – Фрагмент разрешенных расширений для загрузки насайт

Такая же защита создается также на сервере.

- Необходимо установить максимальный размер файлов, чтобы снизить нагрузку на сервер — это поможет от DoS- и DDoS-атак.

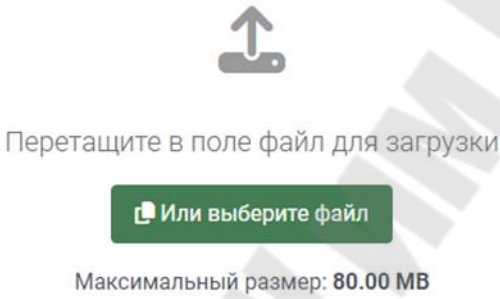

Рис. 85 – Максимальный размер файлов для загрузки на сайт

- Загруженные файлы хранятся вне корневой директории, (например, на поддомене). Тогда загруженный вредоносный файл не навредит основному коду веб-приложения.

- Использование политики безопасности контента: в ней указывается, какой код выполнить, а какой заблокировать. Например, запрет на выполнение встроенного кода: даже если киберпреступник добавит вредоносный код, то он не выполнится и не навредит. Также можно разрешить выполнение скрипта только из определённой директории или только с собственного домена.

Заключение

В данном учебном пособии рассмотрены основные аспекты разработки веб-приложений, изучены ключевые принципы проектирования и реализации веб-ресурсов. Подробно описаны популярные платформы CMS, такие как WordPress, Joomla!, Drupal и другие.

Материал охватывает вопросы выбора подходящей CMS, позволяющей существенно сократить сроки разработки путем предостав-

ления готовых инструментов для администрирования сайта, интеграции компонентов и расширения функциональности посредством модулей и плагинов. Подробно освещены этапы планирования проекта, настройка безопасности, оптимизация производительности и продвижение ресурса в Интернете.

Рассматриваются также вопросы выбора архитектуры приложения, взаимодействие клиентской и серверной частей, работа с базами данных и обеспечение доступности контента пользователям различных устройств и браузеров. Практическое применение полученных знаний позволит укрепить теоретическую базу и сформировать необходимые профессиональные компетенции.

ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Безопасность на уровне кода: руководство по защите веб-приложений для начинающих.

[Электронный ресурс. URL:https://habr.com/ru/companies/runity/articles/878066/ (дата обращения: 01.01.2024)].

2. Руководство по написанию чистого и читабельного кода для начинающих разработчиков [Электронный ресурс. URL:https://clck.ru/3KYHdX (дата обращения: 01.01.2024)].

3. Платформы для создания сайтов: CMS, фреймворки и SaaS‑решения [Электронный ресурс. URL: https://web-creator.ru/articles/platforms (дата обращения: 13.01.2024)].

4. Рейтинг CMS [Электронный ресурс. URL:https://itrack.ru/research/cmsrate/(дата обращения: 2.02.2024)].

5. На каких cms работают белорусские сайты: исследование hoster.by [Электронный ресурс. URL:Https://ratingbynet.by/na-kakikh-cms-rabotayut-belorusskie-sayty-issledovanie-hoster-by/(дата обращения: 2.03.2024)].

6. Что такое плагины и как их использовать [Электронный ресурс. URL:https://clck.ru/3MrXke/(дата обращения: 7.03.2024)].

7. Критерии выбора CMS с открытым исходным кодом

CMS или фреймворк: что выбрать для разработки сайта? [Электронный ресурс. URL:https://malevich1.ru/cms-ili-frejmvork-chto-vyibrat-dlya-razrabotki-sajta/ (дата обращения: 10.03.2024)].

1. Что такое хостинг для сайта и как его выбрать [Электронный ресурс. URL:https://clck.ru/3MrXhi (дата обращения: 25.03.2024)].

2. Регистрация сайта в БелГИЭ [Электронный ресурс. URL:https://hoster.by/service/solutions/belgie/ (дата обращения: 1.04.2024)].

3. Что такое хостинг для сайта и как его выбрать [Электронный ресурс. URL:https://clck.ru/3MrXfF (дата обращения: 23.04.2024)].

4. Фреймворк: особенности, преимущества, архитектура [Электронный ресурс. URL:https://clck.ru/3MrXge (дата обращения: 5.05.2024)].

5. Фреймворки в веб‑разработке [Электронный ресурс. URL:https://clck.ru/3MrXyS (дата обращения: 5.02.2025)].

6. Joomla! Framework [Электронный ресурс. URL:https://framework.joomla.org (дата обращения:1.03.2025)].

7. Информация о домене или IP: WHOIS [Электронный ресурс. URL:https://whois.cctld.by (дата обращения: 14.05.2025)].

8. FTP-клиент [Электронный ресурс. URL:https://filezilla.ru (дата обращения: 3.06.2025)].

9. Государственное предприятие «БелГИЭ» [Электронный ресурс. URL:https://belgie.by (дата обращения: 25.07.2025)].

10. Белтелеком, услуги хостинга [Электронный ресурс. URL:https://beltelecom.by/private/hosting (дата обращения: 14.08.2025)].

11. Центр обработки данных РУП «Белтелеком» [Электронный ресурс. URL:https://datacenter.by (дата обращения: 5.08.2025)].

12. Хостер [Электронный ресурс. URL:https://hoster.by (дата обращения: 13.08.2025)].

13. Регистрация сайтов в Беларуси: вопросы и ответы [Электронный ресурс. URL:https://hoster.by/clients/news/413/ (дата обращения: 03.09.2025)].

РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЙ

Пособие
для слушателей специальности переподготовки
9-09-0611-02 «Веб-дизайн и компьютерная графика»
заочной формы обучения

Составитель **Леонова** Вероника Николаевна

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 04.12.25.
Рег. № 83Е.
http://www.gstu.by