**Содержание**

[Введение 2](#_Toc187011770)

[Профессия Fullstack разработчик 4](#_Toc187011771)

[Общие требования к программному продукту 6](#_Toc187011772)

[Техническое задание на создание веб-сайта 8](#_Toc187011773)

[Работа над техническим заданием на создание веб-сайта 11](#_Toc187011774)

[Технология гипертекста — язык HTML 11](#_Toc187011775)

[CSS (Cascading Style Sheets) — язык стилей 12](#_Toc187011776)

[JavaScript – язык для создания интерактивных и динамичных веб-приложений 13](#_Toc187011777)

[jQuery - библиотека JavaScript 15](#_Toc187011778)

[PHP (Hypertext Preprocessor) 16](#_Toc187011779)

[MySQL - система управления базами данных (СУБД) 17](#_Toc187011780)

[SQLite 18](#_Toc187011781)

[Laravel —PHP-фреймворк 20](#_Toc187011782)

[Git 21](#_Toc187011783)

[Хостинг 23](#_Toc187011784)

[Доменное имя 25](#_Toc187011785)

[Сроки выполнения 27](#_Toc187011786)

[Вычислительные ресурсы для разработки сайта 29](#_Toc187011787)

[Документы ГОСТы и СНиПы 31](#_Toc187011788)

[Затраты на разработку и размещение сайта в Интернете 33](#_Toc187011789)

[Заключение 36](#_Toc187011790)

[Список использованных источников 37](#_Toc187011791)

[Приложения 38](#_Toc187011792)

# **Введение**

В современном мире цифровые технологии играют ключевую роль в функционировании и развитии предприятий. С каждым годом всё больше компаний осознают необходимость присутствия в интернете, что позволяет не только привлечь новых клиентов, но и укрепить связь с уже существующими. Веб-сайт стал неотъемлемой частью стратегии учреждения – муниципальной библиотеки, в частности сектора электронной информации (далее - компании, предприятия, учреждения, муниципальной библиотеки, библиотеки, сектора, сектора электронной информации) выступая ее «лицом» в глобальной сети. Он выполняет множество функций: от предоставления информации о товарах (мероприятиях) и услугах до возможности проведения онлайн-продаж и взаимодействия с клиентами.

Актуальность разработки веб-сайта для компании невозможно переоценить. В условиях стремительно развивающегося цифрового рынка, где конкуренция достигает максимальных значений, наличие качественного, функционального и профессионально оформленного сайта — это не просто дань моде, а необходимость. Даже небольшие компании и стартапы стремятся создать свое онлайн-пространство, чтобы быть конкурентоспособными и востребованными в своих отраслях.

Целью данного дипломного проекта является разработка веб-сайта для сектора электронной информации муниципальной библиотеки. Задачи, которые предстоит решить в процессе работы над проектом: проектирование структуры сайта, а также реализация функциональных возможностей, соответствующих целям сектора электронной информации, удобство и простота использования сайта, что в современном веб-дизайне имеет первостепенное значение.

Техническая реализация представляет собой программу, включающую создание и дополнение программного кода, настройку баз данных.

Данный дипломный проект нацелен на создание эффективного и современного веб-сайта, который будет способствовать её успешному развитию в условиях потребительского рынка и изменения предпочтений клиентов сектора. Результаты работы не только удовлетворят текущие потребности, но и обеспечат основу для дальнейшего роста и расширения его онлайн-присутствия в будущем.

# **Профессия Fullstack разработчик**

Fullstack разработчик — это специалист, обладающий широкими знаниями и навыками в разработке программного обеспечения. Он способен работать как с фронтенд (клиентская часть), так и с бэкенд (серверная часть) разработкой, что позволяет ему создавать полноценные веб-приложения или сайты от начала до конца. Данные знания и навыки в разработке будут отражены в данном дипломном проекте на примере веб-сайта сектора электронной информации муниципальной библиотеки.

Обязанности

1. Разработка интерфейсов:

- Проектирование и реализация пользовательских интерфейсов (UI).

- Адаптивная верстка (Responsive Design).

- Работа с фреймворками и библиотеками, такими как React, Vue.js, Angular.

2. Серверная разработка:

- Создание и поддержка серверной логики приложения.

- Работа с базами данных (SQL и NoSQL).

- Разработка API для взаимодействия клиентской и серверной части.

3. Интеграция:

- Интеграция клиентской и серверной частей.

- Работа с сторонними сервисами и API.

4. Тестирование и отладка:

-Проведение тестирования на всех уровнях (юнит-тестирование, функциональное тестирование).

- Отладка кода и исправление ошибок.

5. Участие в проектировании:

- Участие в проектировании архитектуры приложений.

- Взаимодействие с командами дизайнеров и продуктовых менеджеров.

Необходимые навыки

1. Фронтенд

- HTML, CSS, JavaScript.

- Знания фреймворков (React, Angular, Vue.js и т.д.).

- Опыт работы с системами контроля версий (Git).

2. Бэкенд

- Знания языков программирования (Node.js, Python, Ruby, PHP, Java).

- Опыт работы с серверными фреймворками (Express, Django, Spring, Laravel).

- Работа с базами данных (MySQL, PostgreSQL, MongoDB, SQLite).

3. Общие знания

- Основы DevOps (контейнеризация, CI/CD).

- Понимание принципов RESTful API.

- Основы безопасности веб-приложений.

# **Общие требования к программному продукту**

При разработке программного продукта необходимо учитывать ряд требований, которые обеспечивают его качество, функциональность и соответствие потребностям пользователей. Основные требования к программному продукту можно разделить на несколько категорий:

1. Функциональные требования

Функциональность. Программный продукт должен выполнять заданные функции, удовлетворяющие потребности пользователя. Условия и ограничения, в которых должна быть реализована функциональность, также должны быть четко определены.

Пользовательский интерфейс. Интерфейс должен быть интуитивно понятным, простым в использовании и соответствовать современным стандартам дизайна.

Совместимость. Программное обеспечение должно взаимодействовать с другими системами и продуктами, которыми пользуются пользователи (например, интеграция с базами данных или другими приложениями).

2. Нефункциональные требования.

Производительность. Программный продукт должен обеспечивать требуемую скорость обработки данных и эффективное использование ресурсов (памяти, процессора и т.д.).

Надежность. Программа должна функционировать корректно и надежно, минимизируя вероятность сбоев и ошибок. Это включает в себя обработку ошибок и устойчивое восстановление после сбоев.

Безопасность. Необходимо учитывать защищенность данных и возможность их несанкционированного доступа. Сюда входят шифрование, аутентификация и авторизация пользователей.

Юзабилити. Программный продукт должен быть удобен для пользователя, включая простоту навигации и доступность информации.

3. Критерии качества.

Портируемость. Программное обеспечение должно быть легко переносимо на различные аппаратные платформы и операционные системы.

Поддерживаемость. Код программы должен быть структурирован и документирован таким образом, чтобы его можно было легко поддерживать и модифицировать в будущем.

Тестируемость. Система должна быть создана так, чтобы ее можно было удобно тестировать на разных уровнях (модульное тестирование, интеграционное тестирование и т.д.).

4. Пользовательские требования

Документация. Необходимо предоставить полную документацию, включая руководство пользователя, технические спецификации и инструкции по установке.

Обратная связь от пользователей. Важно учитывать мнения и пожелания пользователей для дальнейшего улучшения продукта.

5. Регуляторные и правовые требования.

Соблюдение стандартов и нормативов. Программный продукт должен соответствовать применимым стандартам (например, ISO, IEEE), а также требованиям местного законодательства.

Лицензирование и авторские права. Обеспечение правовой защиты интеллектуальной собственности, включая правильно оформленные лицензии на использование стороннего программного обеспечения и компонентов.

Заключение.

Эти требования формируют основу для успешной разработки программного продукта и позволяют создать качественное и востребованное программное обеспечение. Важно учитывать, что требования могут изменяться в процессе разработки и уточняться в зависимости от специфики проекта и нужд пользователей.

# **Техническое задание на создание веб-сайта**

1. Введение

1.1 Цель создания сайта

Создание веб-сайта с целью предоставления информации о компании, представления товаров и услуг, облегчения взаимодействия с клиентами и улучшения имиджа компании в интернете.

1.2 Описание проекта

Сайт будет служить витриной для компании, предоставляющей услуги в сфере оказания услуг полиграфии (как подразделение (сектор) муниципальной библиотеки). Он будет включать разделы с информацией о компании, анонсы предстоящих мероприятий в библиотеке, контактные данные, блог(отдельный поддомен с заказами клиентов (посетителей сектора электронной информации) и другие полезные функции.

2. Требования к сайту

2.1 Общие требования

- Адаптивный дизайн, совместимость с мобильными устройствами.

- Удобная навигация и интуитивно понятный интерфейс.

- Быстрая загрузка страниц.

- Высокая степень безопасности данных пользователей.

2.2 Функциональные требования

2.2.1 Главная страница

- Кнопка зарегистрироваться/войти.

- Приветствие

- Краткая информация о компании.

2.2.2 Раздел «Наши возможности»

- Мероприятия и выставки.

- Миссия и ценности.

- О нашем секторе.

2.2.3 Раздел Мероприятия

- Возможность добавления заказов.

- Комментарии к заказов.

- Поиск по заказам.

2.2.4 Свяжитесь с нами

- Обратная связь

2.2.6 Личный кабинет пользователя

- Регистрация/Авторизация.

- Возможность редактирования профиля заказа, просмотра заказов.

2.3 Нефункциональные требования

- Высокая скорость загрузки (не более 3 секунд на страницу).

- SEO-оптимизация (все страницы должны быть проиндексированы поисковыми системами).

- Кроссбраузерная совместимость (Chrome, Firefox, Safari, Edge).

3. Технические детали

3.1 Платформа

- Сайт будет разработан:

1. для основного домена на технологиях HTML, Javascript, JQuery, PHP, MySQL

2. для поддомена на PHP + фреймворк Laravel, MySQL (sqlite).

3.2 Хостинг и домен

- Требуемый хостинг (объем памяти, трафик, база данных).

- Доменное имя.

3.3 Дизайн

- Разработка индивидуального дизайна.

- Прототипы страниц.

4. Процесс разработки

4.1 Этапы работы

- Подготовка прототипа и дизайн-макета.

- Разработка функционала.

- Тестирование.

- Запуск сайта.

4.2 Сроки выполнения

- Сроки для каждого этапа разработки.

5. Заключение

Данное техническое задание содержит основные требования и пожелания к созданию веб-сайта. В процессе разработки возможны уточнения и изменения в зависимости от потребностей бизнеса и недостатков, выявленных в ходе работы.

# **Работа над техническим заданием на создание веб-сайта**

Пункты 1, 2, 3 были рассмотрены заранее вместе с вышестоящим руководством, непосредственным руководством, отделами отвечающими за обслуживание информационной деятельности (освещения работы библиотеки) и отделом автоматизации, как отделом отвечающим за общую инфраструктуру.

В ходе п, редварительных работ с руководством и привлеченными работниками (работниками на аутсорсе) была разработана структура сайта для сектора электронной информации муниципальной библиотеки (компании). Она определена:

Главная страница

Раздел «Наши возможности»

Раздел Мероприятия

Свяжитесь с нами

Личный кабинет пользователя

Мною как разработчиком (fullstack-разработчиком) в части технической разработки (разработки кодовой базы проекта) были определены основные языки программирования/разметки, необходимые библиотеки, фреймворки, СУБД и дополнительное программное обеспечение.

# **Технология гипертекста — язык HTML**

HTML (HyperText Markup Language) — это язык разметки, используемый для создания веб-страниц и веб-приложений. Он обеспечивает структуру и содержание документа, описывая, как различные элементы должны отображаться в браузере.

Основные характеристики HTML:

1. Теги и элементы. HTML состоит из тегов, которые используются для определения различных элементов на странице. Теги обычно идут парами: открывающий тег, например `<p>`, и закрывающий тег — `</p>`. Между ними находится содержимое элемента.

2. Структура документа. Стандартная структура HTML-документа включает в себя объявление типа документа (`<!DOCTYPE html>`), элемент `<html>`, в котором находятся две основные секции — `<head>` и `<body>`. В `<head>` размещается метаинформация (например, заголовок страницы, связи с CSS и JS), а в `<body>` – собственно содержимое страницы (тексты, изображения, ссылки и др.).

3. Атрибуты. Теги могут содержать атрибуты, которые предоставляют дополнительную информацию об элементе. Например, атрибут `src` используется для указания источника изображения в теге `<img>`: `<img src="image.jpg" alt="Описание изображения">`.

4. Семантические элементы. Современный HTML включает множество семантических тегов, таких как `<header>`, `<nav>`, `<article>`, `<section>`, `<footer>` и другие, которые помогают описать структуру страницы и улучшают доступность и поисковую оптимизацию.

5. Совместимость с CSS и JavaScript. HTML часто используется в связке с CSS (для оформления и стилей) и JavaScript (для интерактивности и динамического поведения страницы).

Заключение

HTML является основой веб-разработки и необходим для создания качественных веб-страниц. Понимание его структуры и принципов работы позволяет создавать гармоничные и удобные для пользователя интерфейсы.

# **CSS (Cascading Style Sheets) — язык стилей**

CSS (Cascading Style Sheets) — это язык стилей, используемый для описания внешнего вида документа, написанного с использованием HTML или XML (включая такие языки разметки, как SVG и XHTML). CSS позволяет разделять содержание и оформление, что упрощает создание и обслуживание веб-страниц.

Основные характеристики CSS:

1. Селекторы и правила. В CSS используются селекторы для определения того, к каким элементам HTML будут применяться стили. Правило CSS состоит из селектора и набора деклараций, которые заключаются в фигурные скобки.

2. Каскадность. Один из ключевых аспектов CSS — это каскадность. Если несколько правил применяются к одному и тому же элементу, CSS определяет, какое правило имеет больший приоритет (например, по специфичности селектора или позиции в коде).

3. Модель. CSS использует концепцию модели коробки (box model), которая определяет, как элементы отображаются на странице. Каждый элемент состоит из содержимого, обрамлённого полями, отступами и границами. Это позволяет контролировать размер, отступы и выравнивание элементов.

4. Свойства. CSS включает множество свойств для стилизации, таких как `color`, `background`, `margin`, `padding`, `border`, `font`, `display`, `position`, `flex`, `grid` и многие другие, позволяя детально настраивать внешний вид веб-страницы.

5. Медиа-запросы. CSS поддерживает адаптивный дизайн с помощью медиа-запросов. Это позволяет применить разные стили для различных устройств и размеров экрана.

Заключение

CSS является важным инструментом для веб-разработчиков, позволяя создавать привлекательные и удобные пользовательские интерфейсы. Он обеспечивает гибкость и контроль над стилем и расположением элементов, способствуя отлаживанию и поддержанию визуального оформления веб-страниц. Совместно с HTML и JavaScript, CSS составляет основу современных веб-технологий.

# **JavaScript – язык для создания интерактивных и динамичных веб-приложений**

JavaScript — это высокоуровневый, интерпретируемый язык программирования, который используется для создания интерактивных и динамичных веб-приложений. Он является важной частью веб-технологий наряду с HTML и CSS и часто используется для разработки фронтенд- и бэкенд-решений.

Основные характеристики JavaScript:

1. Интерактивность. JavaScript позволяет добавлять интерактивные элементы на веб-страницы. Это может быть, например, валидация форм, создание анимаций, манипуляция DOM (Document Object Model), управление событиями и многое другое.

2. Объектно-ориентированный. JavaScript поддерживает объектно-ориентированное программирование, что позволяет создавать и использовать объекты, а также наследовать свойства и методы. Это делает код более структурированным и удобным для повторного использования.

3. Динамическая типизация. В JavaScript переменные могут принимать значение любого типа, что делает язык гибким. Однако это также может привести к неожиданным ошибкам, если не следить за тем, какие типы данных используются.

4. Асинхронность. JavaScript поддерживает асинхронное программирование с использованием коллбеков, промисов и `async/await`. Это позволяет выполнять задачи, такие как запросы к серверу, не блокируя основной поток выполнения кода.

5. Широкий спектр применения. JavaScript может использоваться как на клиентской стороне (в браузере), так и на серверной (например, с помощью Node.js). Это делает его универсальным языком для веб-разработчиков.

6. Большое сообщество и экосистема. JavaScript имеет огромное сообщество разработчиков, множество библиотек и фреймворков, таких как React, Angular и Vue.js, которые значительно упрощают разработку сложных приложений.

Заключение

JavaScript является мощным инструментом для разработки веб-приложений, позволяя создавать динамичные и интерактивные интерфейсы. Его популярность и гибкость сделали его стандартом де-факто для веб-разработки, что делает его незаменимым навыком для современных разработчиков. С учетом его возможностей и широкой экосистемы, JavaScript продолжает развиваться и адаптироваться к потребностям индустрии.

# **jQuery - библиотека JavaScript**

jQuery — это быстрая, небольшая и удобная библиотека JavaScript, созданная для упрощения работы с HTML-документами, обработкой событий, анимацией и AJAX-запросами. Она была разработана в 2006 году и быстро стала популярной благодаря своей простоте и кроссбраузерной совместимости.

Основные характеристики jQuery:

1. Упрощение синтаксиса. jQuery позволяет разработчикам писать менее объемный и более читабельный код. Например, чтобы выбрать элемент и изменить его содержимое, в jQuery достаточно выполнить одну строчку.

2. Управление событиями. jQuery предоставляет удобные методы для работы с событиями (например, клики, наведение мыши и т. д.). С помощью jQuery можно легко добавить обработчики событий.

3. Анимация и эффекты. jQuery предлагает встроенные методы для создания анимаций и визуальных эффектов. Разработчики могут легко добавлять анимацию к элементам, например, скрывать или показывать их.

4. AJAX. jQuery значительно упрощает работу с AJAX (Asynchronous JavaScript and XML), позволяя делать асинхронные запросы к серверу и обновлять части страницы без перезагрузки.

5. Совместимость с браузерами. jQuery позволяет разработчикам избегать проблем с различиями в реализации JavaScript в разных браузерах. Библиотека решает многие проблемы, связанные с кроссбраузерностью, позволяя разработчикам сосредоточиться на функциональности, а не на деталях реализации.

Заключение

jQuery остается популярным инструментом в веб-разработке благодаря своей простоте и мощным возможностям. Хотя за последние годы появились фреймворки, такие как React, Angular и Vue.js, jQuery все еще широко используется, особенно в существующих проектах и для создания простых веб-страниц. Несмотря на это, важно помнить, что с развитием стандартов JavaScript, многие функции jQuery теперь доступны в чистом JavaScript, что делает использование библиотеки менее обязательным для новых проектов.

# **PHP (Hypertext Preprocessor)**

PHP (Hypertext Preprocessor) — это широко используемый серверный язык программирования, предназначенный для создания динамических веб-страниц и веб-приложений. Он был разработан для веба и интегрируется с HTML, что делает его популярным выбором для создания серверной логики веб-сайтов.

Основные характеристики PHP:

1. Серверный язык. PHP выполняется на сервере, и результат обработки отправляется клиенту (в браузер). Это позволяет динамически формировать содержимое веб-страниц на основе запроса пользователя.

2. Интеграция с HTML.. PHP позволяет встраивать PHP-код непосредственно в HTML-код. Это делает его простым для использования при добавлении серверной логики к веб-страницам.

3. Поддержка баз данных. PHP в основном используется в связке с базами данных, такими как MySQL, PostgreSQL и другими. Он предлагает простые и эффективные методы для взаимодействия с базами данных, например, для извлечения, добавления и обработки данных.

4. Широкие возможности. PHP поддерживает множество расширений и библиотек, которые помогают реализовать различные функции — от работы с изображениями до отправки электронной почты. Также доступны фреймворки, такие как Laravel, Symfony и CodeIgniter, которые упрощают разработку сложных веб-приложений.

5. Кросс-платформенность. PHP является кросс-платформенным языком, что означает, что его можно использовать на различных операционных системах: Windows, macOS, Linux и других. Он также может работать с различными веб-серверами, включая Apache и Nginx.

6. Обширное сообщество и документация. У PHP существует большое сообщество разработчиков, поэтому доступно множество ресурсов, документации, форумов и примеров кода, что облегчает изучение языка.

Заключение

PHP остается одним из наиболее популярных языков программирования для веб-разработки и был использован для создания многих известных веб-сайтов и приложений. Его простота, возможности расширения и интеграция с базами данных делают его отличным выбором как для начинающих, так и для опытных разработчиков. Несмотря на появление новых технологий и языков, PHP продолжает активно развиваться и использоваться в десятках миллионов сайтов по всему миру.

# **MySQL - система управления базами данных (СУБД)**

MySQL — это одна из самых популярных систем управления базами данных (СУБД), которая использует язык SQL (Structured Query Language) для работы с данными. Она является свободным программным обеспечением, что делает её доступной для разработчиков и организаций, а также является частью экосистемы LAMP (Linux, Apache, MySQL, PHP/Python/Perl), которая широко используется для разработки веб-приложений.

Основные характеристики MySQL:

1. Реляционная модель. MySQL использует реляционную модель, что означает, что данные хранятся в таблицах, которые могут быть связаны друг с другом. Это позволяет легко организовывать, запрашивать и управлять данными.

2. Язык SQL. MySQL работает с использованием языка SQL для выполнения операций, таких как создание, чтение, обновление и удаление (CRUD) данных. SQL предоставляет мощные инструменты для манипуляции данными и извлечения нужной информации.

3. Высокая производительность. MySQL оптимизирована для быстрого выполнения запросов и обработки больших объемов данных. Это делает её хорошим выбором для высоконагруженных веб-приложений.

4. Кросс-платформенность. MySQL работает на различных операционных системах, включая Windows, Linux и macOS, что позволяет использовать её в самых разных средах.

5. Безопасность. MySQL предоставляет функции безопасности, такие как аутентификация пользователей, управление правами доступа и шифрование данных, что помогает защитить информацию от несанкционированного доступа.

6. Сообщество и поддержка. MySQL имеет большую пользовательскую базу и активное сообщество, что обеспечивает наличие обширной документации, учебных материалов и форумов для получения помощи и обмена опытом.

7. Поддержка транзакций. MySQL поддерживает транзакции, что позволяет выполнять множественные операции как единое целое, обеспечивая целостность данных.

Заключение

MySQL является надежной и мощной системой управления базами данных, которая широко используется в веб-разработке. Она подходит для создания как простых, так и сложных приложений, обрабатывающих большие объемы данных. Благодаря своим возможностям, масштабируемости и активному сообществу, MySQL остается одним из наиболее популярных выборов для разработчиков по всему миру.

# **SQLite**

SQLite — это легковесная, встраиваемая система управления базами данных (СУБД), которая реализует реляционную модель и использует язык SQL для работы с данными. В отличие от других СУБД, таких как MySQL или PostgreSQL, SQLite не требует установки и сложной конфигурации, что делает ее удобной для использования в различных приложениях.

Основные характеристики:

1. Легковесность. SQLite самодостаточная и имеет минимальные системные требования. Весь код базы данных находится в одном файле, что упрощает переносимость и использование.

2. Встраиваемая. SQLite может быть встроена непосредственно в приложения, что позволяет взаимодействовать с базой данных без установки отдельного серверного программного обеспечения. Это делает её популярной для мобильных приложений, десктопных программ и проектов с небольшими объемами данных.

3. Поддержка SQL. SQLite поддерживает большинство стандартных SQL-операций, что позволяет разработчикам использовать знакомый язык для управления данными.

4. Транзакционная безопасность. SQLite поддерживает транзакции с использованием механизмов "Atomicity", "Consistency", "Isolation" и "Durability" (ACID), что обеспечивает точность и надежность операций с данными.

5. Кросс-платформенность. SQLite доступна для различных операционных систем, включая Windows, macOS, Linux и другие, что способствует ее использованию в разных средах.

6. Простота использования. SQLite не требует настройки сервера и является простым в использовании. Создание и управление базой данных возможно с помощью легких команд и SQL-запросов.

7. Поддержка многопоточности: SQLite поддерживает несколько потоков и процессов, что позволяет нескольким пользователям одновременно работать с базой данных, хотя возможны некоторые ограничения на запись.

Использование SQLite в приложениях:

SQLite часто используется в следующих сценариях:

Мобильные приложения. Например, приложения на Android и iOS часто используют SQLite для хранения данных на стороне клиента.

Легкие веб-приложения. SQLite может использоваться для небольших веб-сайтов или приложений, которые требуют простой базы данных без необходимости развертывания серверного решения.

Программное обеспечение. Множество настольных приложений используют SQLite для локального хранения данных и управления ими.

Заключение

SQLite является мощным инструментом для работы с базами данных, обеспечивая эффективность и простоту в использовании. Его легковесность и простой интерфейс делают его отличным выбором для проектов с небольшими объемами данных или когда требуется встроенное решение. Благодаря своей популярности и широкой поддержке, SQLite остаётся одним из наиболее востребованных вариантов реляционных баз данных среди разработчиков по всему миру.

# **Laravel —PHP-фреймворк**

Laravel — это мощный PHP-фреймворк для веб-разработки, который обеспечивает удобные инструменты и архитектуру для создания современных веб-приложений. Он был создан Тейлором Отвеллом и впервые выпущен в 2011 году. Laravel быстро завоевал популярность благодаря своей элегантной и чистой архитектуре, а также обширному набору функций.

Основные характеристики Laravel:

1. MVC-архитектура. Laravel использует модель-вид-контроллер (MVC) архитектуру, что помогает разделить логику приложения, представления и данные, делая код более структурированным и удобным для сопровождения.

2. Элегантный синтаксис. Laravel предлагает простой и интуитивно понятный синтаксис, который облегчает процесс разработки. Это позволяет разработчикам быстро создавать функциональные и читаемые приложения.

3. ORM Eloquent. Laravel включает встроенный ORM (Object-Relational Mapping) Eloquent, который позволяет работать с базами данных с использованием объектов и облегчает выполнение операций с данными, таких как создание, чтение, обновление и удаление (CRUD).

4. Маршрутизация и контроллеры. Laravel упрощает управление маршрутизацией, позволяя разработчикам легко определять URL-адреса и связывать их с контроллерами и действиями. Это упрощает организацию и поддержку кода.

5. Интеграция с системами аутентификации и авторизации. Laravel предоставляет готовые решения для аутентификации пользователей, включая регистрацию, вход в систему, сброс пароля и управление ролями и правами доступа.

6. Шаблонизация Blade. Laravel использует шаблонизатор Blade, который позволяет создавать динамические HTML-шаблоны с использованием MVC. Blade упрощает процесс генерации представлений и позволяет использовать компоненты и шаблоны.

7. Миграции и управление схемами базы данных. Laravel предлагает инструмент для управления схемами базы данных с помощью миграций, что упрощает процесс создания и изменения таблиц. Это позволяет командам разработчиков синхронизировать изменения в структуре базы данных.

8. Поддержка тестирования. Laravel имеет встроенные инструменты для написания тестов, что позволяет разработчикам легко проверять функциональность приложения и проводить автоматизированное тестирование.

9. Обширная экосистема. Laravel имеет широкую экосистему пакетов и инструментов, таких как Laravel Forge (развертывание), Laravel Nova (административная панель), Laravel Mix (управление активами) и другие, которые позволяют значительно упростить процесс разработки.

Заключение

Laravel является одним из самых популярных PHP-фреймворков благодаря своей простоте, гибкости и мощным функциям. Он подходит как для небольших проектов, так и для крупных веб-приложений, обеспечивая разработчикам необходимые инструменты для эффективной работы. С активным сообществом и постоянными обновлениями, Laravel продолжает оставаться актуальным выбором для разработчиков.

# **Git**

Git — это распределенная система контроля версий, которая позволяет эффективно управлять изменениями в коде и других файлах. Она была разработана Линусом Торвальдсом в 2005 году, чтобы поддерживать разработку ядра Linux. С помощью Git разработчики могут отслеживать историю изменений, работать над разными версиями проекта, а также взаимодействовать с другими разработчиками.

Основные концепции Git:

1.Репозиторий. Это основное хранилище проекта, которое содержит все файлы и историю изменений. Репозиторий может быть локальным (на вашем компьютере) и удаленным (например, на GitHub или GitLab).

2. Коммит. Это точка сохранения изменений в репозитории. Каждый коммит содержит описание изменений и ссылку на предыдущий коммит, что позволяет создать полную историю изменений.

3. Ветки. Git позволяет создавать несколько веток для разработки. Это удобно для работы над новыми функциями или исправлениями, не влияя на основную (главную) ветку проекта. Ветки можно сливать (merge), создавая единую историю изменений.

4. Слияние. Процесс объединения изменений из разных веток. Существуют различные методы слияния, например, fast-forward и три-стороннее слияние.

5. Отложенные изменения. Git позволяет откладывать изменения с помощью функции index (или staging area), что дает возможность выбирать, какие изменения будут добавлены в следующий коммит.

6. Объединенные репозитории. Git поддерживает распределенную модель работы, где каждый разработчик имеет полную копию репозитория, что позволяет работать офлайн и легко синхронизироваться с другими участниками проекта.

Преимущества Git. Производительность. Git быстро выполняет многие операции благодаря кэширующим механизмам и эффективным алгоритмам.

Система ветвления. Создание и слияние веток осуществляется быстро и легко, что упрощает параллельную работу над функциями.

Безопасность. Все данные и история изменений хранятся локально, что обеспечивает защиту от потерь данных.

Сообщество и интеграции. Git является стандартом в разработке, и многие платформы, такие как GitHub и GitLab, предоставляют инструменты для совместной работы, тестирования и развертывания.

Заключение

Git стал основным инструментом для командной разработки программного обеспечения и широко используется в различных проектах, от небольших до крупных коммерческих приложений.

# **Хостинг**

Необходимый хостинг был определен как виртуальный.

Виртуальный хостинг — это метод размещения веб-сайтов на одном физическом сервере, при этом каждый сайт работает в своей изолированной среде. Этот подход позволяет эффективно использовать ресурсы сервера и обеспечивает более доступный вариант хостинга, особенно для малых и средних предприятий или для личных проектов.

Основные характеристики виртуального хостинга:

1. Экономия ресурсов: На одном сервере может размещаться множество сайтов, что позволяет снизить затраты на хостинг. Виртуальный хостинг распределяет серверные ресурсы (процессор, память, дисковое пространство) между всеми сайтами, размещенными на этом сервере.

2. Управление доменами: Каждый сайт, размещенный на виртуальном хостинге, может иметь свой собственный домен. Это позволяет пользователям легко управлять несколькими веб-сайтами под одной учетной записью хостинга.

3. Простота использования: Большинство провайдеров виртуального хостинга предлагают удобные панели управления (например, cPanel или Plesk), которые упрощают процесс управления сайтами, базами данных, электронной почтой и другими функциями.

4. Безопасность: Хотя ресурсы сервера разделены, каждый сайт работает в своей изолированной среде, что обеспечивает базовый уровень безопасности. Тем не менее, сайты на одном сервере могут быть уязвимы к потенциальным атакам на другие сайты, если сервер не настроен должным образом.

5. Гибкость и масштабируемость: Виртуальный хостинг предоставляет возможность масштабировать ресурсы по мере необходимости, например, увеличить дисковое пространство или процессорные ресурсы, когда ваш сайт начинает расти.

Недостатки виртуального хостинга:

1. Ограниченные ресурсы: Поскольку ресурсы сервера делятся между несколькими пользователями, такие сайты могут сталкиваться с проблемами производительности, особенно в периоды пиковых нагрузок.

2. Предсказуемая производительность: Виртуальный хостинг может иметь менее предсказуемую производительность, поскольку сайты на одном сервере могут влиять друг на друга. Если один сайт использует слишком много ресурсов, это может привести к замедлению работы всех других сайтов на том же сервере.

3. Ограниченные настройки: Хотя виртуальный хостинг предоставляет множество функций, он может ограничивать возможности конфигурации сервера, так как пользователи не имеют доступа к полному управлению серверной средой.

Применение виртуального хостинга:

Виртуальный хостинг часто используется для:

- Личных веб-сайтов и блогов

- Сайтов малых и средних предприятий

- Портфолио и рекламных страниц

- Разработки и тестирования приложений

- Онлайн-магазинов с небольшим объемом трафика

Заключение

Виртуальный хостинг — это отличный выбор для пользователей, которые ищут доступный и удобный способ разместить свои веб-сайты. Он подходит для широкого спектра проектов, от личных блогов до малых бизнесов.

Тип в используемых хостинг-провайдерами линейках тарифов определен как Blog (Блог).

Объем памяти, трафик и база данных для виртуального хостинга тип Блог могут варьироваться в зависимости от платформы, количества посетителей и содержимого блога. Вот примерные параметры:

1. Объем памяти:

- Для небольшого блога может хватить от 1 ГБ до 5 ГБ дискового пространства.

- Для среднего блога с большей аудиторией — от 5 ГБ до 20 ГБ.

2. Трафик:

- Небольшой блог может потреблять от 1 ГБ до 5 ГБ трафика в месяц.

- Для блогов со средней посещаемостью — от 10 ГБ до 50 ГБ.

3. База данных:

- Небольшой блог может использовать базу данных объемом от 100 МБ до 1 ГБ.

- Более крупные блоги могут иметь объем базы данных от 1 ГБ до 5 ГБ, в зависимости от числа постов и комментариев.

Эти цифры могут значительно варьироваться в зависимости от специфики контента, интеграций (например, плагинов) и уровня оптимизации. При этом были учтены возможные ограничения и пожелания в плане производительности и управления, что может потребовать перехода на более мощные решения, такие как выделенный сервер или облачный хостинг, по мере роста ресурсов вашего сайта.

# **Доменное имя**

Основные параметры и требования, а также выбор и покупку доменного имени вело руководство по профилю всей системы библиотек, опираясь на мнение основного заказчика – муниципальной библиотеки и работников сектора электронной информации. Покупка и присвоение доменного имени сайту — это важная часть процесса его создания. Вот основные возможности и шаги, которые были исполнены.

1. Выбор доменного имени:

- Выбрано уникальное и запоминающееся имя, которое отражает суть вашего сайта. Оно не слишком длинное и легко произносится.

2. Проверка доступности:

- Использованы сервисы (регистраторов доменов), чтобы проверить доступность выбранного вами доменного имени. Если имя занято, вам могут предложить альтернативные варианты или варианты с другим доменным расширением (TLD).

3. Покупка домена:

- После выбора доступного доменного имени зарегистрировано через регистраторов доменов.

- Зарегистрировано доменное имя на желаемый срок (обычно от 1 года до 10 лет) и оплачено в соответствии с выставленной суммой.

4. Присвоение домена сайту:

- После покупки домена оно настроено в панели управления вашего хостинг-провайдера. Этот процесс включил в себя:

- Изменение DNS-записей: настроены NS-записи домена так, чтобы они указывали на серверы хостинга.

- Добавлен домен в панель управления хостинга: зарегистрирован домен в панели управления (таких как cPanel или Plesk), чтобы хостинг знал, к какому сайту относится данный домен.

5. SSL-сертификат:

- Получен SSL-сертификат для обеспечения безопасности вашего сайта (https). Многие хостинг-провайдеры предлагают бесплатные сертификаты от Let's Encrypt.

6. Обновление и продление:

- Этот шаг был предусмотрен в договоре о дальнейшем обслуживании сайта.

**В п.п. 3.3 и п.4 Технического задания на создания веб-сайта я, как специалист и сотрудник участие не принимал.**

**По п.п. 4.1. работа велась параллельно с основным участием специалистов из аутсорса (некоторые ссылки и упоминания возможно будут указаны ниже).**

# **Сроки выполнения**

Разработка веб-сайта типа блог включает несколько этапов, каждый из которых занимает определенное время. Ниже приведены примерные сроки для каждого из них:

1. Планирование и сбор требований (1-2 недели)

- Определение целей блога

- Исследование целевой аудитории

- Сбор требований и обсуждение функционала

2. Создание прототипа и дизайна (2-4 недели)

- Разработка прототипа (wireframes)

- Создание визуального дизайна (UI/UX)

- Получение обратной связи и финализация дизайна

3. Разработка (4-8 недель)

- Настройка серверной части и базы данных

- Реализация клиентской части (веб-интерфейса)

- Интеграция необходимых функций (например, комментарии, подписка, система управления контентом)

4. Тестирование (2-3 недели)

- Функциональное тестирование

- Тестирование производительности и нагрузки

- Исправление найденных ошибок и недочетов

5. Запуск сайта (1 неделя)

- Настройка хостинга и домена

- Размещение сайта в интернете

- Объявление о запуске (например, в социальных сетях)

6. Пост-лайв поддержка и обновления (постоянно)

- Мониторинг работы блога

- Регулярные обновления контента

- Устранение возможных проблем и багов

Таким образом, общая продолжительность разработки веб-сайта типа блог может составить от 10 до 18 недель в зависимости от сложности проекта и наличия ресурсов.

# **Вычислительные ресурсы для разработки сайта**

Для разработки сайта типа блог необходимо учитывать несколько аспектов, связанных с вычислительными ресурсами. Оценка ресурсов может варьироваться в зависимости от масштабов проекта, предполагаемого трафика и используемых технологий

Она зависит от множества факторов, включая используемые технологии, среду разработки, а также личные предпочтения разработчика. Вот основные рекомендации на основе различных аспектов:

1. Процессор (ЦП):

- Минимум. Двухъядерный процессор (например, Intel Core i3 или аналогичный).

- Рекомендуется. Четырехъядерный процессор (например, Intel Core i5 или аналогичный) для оптимальной работы с средами разработки и запуска локальных серверов.

2. Оперативная память (ОП):

- Минимум. 8 ГБ. Это достаточно для базовой разработки и работы с текстовыми редакторами или IDE.

- Рекомендуется.16 ГБ. Это обеспечит комфортную работу при одновременном запуске нескольких приложений, таких как браузеры, редакторы, эмуляторы и другие инструменты.

3. Накопитель (Жесткий диск):

- Минимум: HDD на 256 ГБ. Несмотря на достаточный объем, скорость работы может быть ниже.

- Рекомендуется: SSD на 512 ГБ или более. SSD значительно ускоряет загрузку операционной системы и приложений, что важно для продуктивности.

4. Графическая карта. Для разработки сайта особых требований к графической карте нет, если только вы не планируете заниматься разработкой фронтенда с использованием сложной графики или работать с видео (в таком случае подойдет встроенная графика или базовая дискретная карта).

5. Операционная система:

- Windows, macOS или Linux. Выбор зависит от предпочтений разработчика и используемых технологий. Например, многие разработчики предпочитают macOS или Linux для веб-разработки.

6. Дополнительные устройства:

- Мониторы. Советуем использовать как минимум один дополнительный монитор для многозадачности. Это может значительно повысить производительность, позволяя работать с кодом и документацией одновременно.

- Клавиатура и мышь. Эффективные и удобные устройства ввода также повысят комфорт работы.

7. Интернет-соединение. Для разработки и тестирования веб-сайтов необходимо стабильное и достаточно быстрое интернет-соединение.

Заключение

Для комфортной работы над сайтом типа блог рекомендуется использовать ПК с четырехъядерным процессором, 16 ГБ ОП, SSD на 512 ГБ и современной операционной системой. Такой набор обеспечит хорошую производительность и позволит разработчику эффективно работать с необходимыми инструментами.

# **Документы ГОСТы и СНиПы**

В России разработка проектной документации регулируется различными стандартами и нормами, среди которых важную роль играют ГОСТы (Государственные стандарты) и СНиПы (Строительные нормы и правила). Ниже перечислены основные из них, которые могут влиять на разработку в различных сферах.

Основные ГОСТы:

1. ГОСТ 2.105-95 - «Единая система конструкторской документации (ЕСКД)». Регламентирует требования к оформлению конструкторской документации.

2. ГОСТ 19.101-77 - «Документация программных средств. Общие требования и правила оформления». Дает основные требования к созданию документации для программных продуктов.

3. ГОСТ Р 52766-2007 - «Системы менеджмента качества. Требования». Устанавливает требования к системам управления качеством в организациях.

4. ГОСТ Р ИСО 9001-2015 - (аналог международного стандарту ISO 9001) устанавливает требования для систем управления качеством.

5. ГОСТ 34.301-95 - «Информационная технология. Комплексы программных средств. Общие требования». Определяет требования к программному обеспечению.

Основные СНиПы:

1. СНиП 01.01-85 - «Организация и постановка проектирования». Устанавливает основные принципы проектирования и требования к проектной документации.

2.СНиП 12-01-2004 - «Общие требования безопасности труда при строительстве» — определяет требования к охране труда при выполнении строительных работ.

3. СНиП 21-01-97 - «Пожарная безопасность зданий и сооружений» — содержит требования к обеспечению пожарной безопасности в процессе проектирования.

4. СНиП 3.03.01-87 - «Несущие и ограждающие конструкции» — регулирует проектирование конструкций зданий.

5. СНиП 31-01-2003 - «Здания жилые многоквартирные» — включает нормы проектирования многоэтажных жилых зданий.

Влияние на разработку:

Качество проектирования. Соблюдение ГОСТов и СНиПов обеспечивает высокое качество проектной документации и продукции.

Безопасность. Нормы по охране труда и пожарной безопасности помогают избежать несчастных случаев.

Соответствие законодательству. Соблюдение этих стандартов является необходимым для получения разрешений и прохождения экспертиз.

Упрощение разработки. Наличие четких регламентов облегчает разработку и согласование проектной документации.

# **Затраты на разработку и размещение сайта в Интернете**

Затраты на разработку и размещение сайта в Интернете могут значительно варьироваться в зависимости от многих факторов, таких как сложность проекта, выбранные технологии, качество дизайна, масштаб бизнеса и другие аспекты. В общем, можно выделить несколько основных категорий затрат:

1. Затраты на разработку

1.1. Проектирование и планирование

Анализ требований. Исследования для определения целевой аудитории, функций сайта и конкурентного анализа.

Создание технического задания (ТЗ): Подробное документирование функций и технологий, необходимых для разработки.

1.2. Дизайн сайта

UX/UI дизайн. Разработка пользовательского интерфейса и опыт пользователя. Это может включать в себя создание макетов, прототипов, концептуальных дизайнов.

Графика и анимация. Если нужно создать уникальную графику, иконки или анимации, это потребует дополнительных затрат.

1.3. Разработка

Frontend-разработка. Создание клиентской части сайта с использованием HTML, CSS и JavaScript.

Backend-разработка. Разработка серверной логики сайта, включая работу с базами данных

Интеграция. Если, требуется интеграция с API сторонних сервисов, это также увеличивает затраты.

1.4. Тестирование

Тестирование производительности. Проверка скорости загрузки и других метрик.

Функциональное тестирование. Проверка всех функций и возможностей сайта.

Пользовательское тестирование. Получение обратной связи от пользователей, чтобы выявить и устранить проблемы.

2. Затраты на размещение

2.1. Хостинг

Сервер. Выбор типа хостинга (shared hosting, VPS, dedicated server, cloud hosting) влияет на стоимость. Стоимость может варьироваться от нескольких долларов в месяц до сотен.

Доменное имя. Регистрация домена (например, .com, .ru). Годовой тариф может варьироваться от 1000р. до 5000р. в зависимости от зоны.

2.2. SSL-сертификат

Для обеспечения безопасности данных пользователей необходим SSL-сертификат. Стоимость может варьироваться от бесплатных (например, Let’s Encrypt) до платных сертификатов, стоимость которых может достигать нескольких сотен долларов в год.

3. Затраты на содержание и поддержку

3.1. Обновления и поддержка сайта

Регулярные обновления контента, безопасности и функциональности требуют постоянных затрат. Это может быть фиксированная ежемесячная плата или оплата по мере необходимости.

3.2. Продвижение и маркетинг

SEO-оптимизация. Расходы на оптимизацию сайта для поисковых систем, чтобы повысить его видимость.

Реклама. Рекламные расходы (контекстная реклама, социальные сети и т.д.) для привлечения посетителей.

3.3. Аналитика и отчетность

Затраты на инструменты аналитики (например, Google Analytics), а также на аналитические отчеты и улучшение сайта на основе полученных данных.

4. Дополнительные затраты

Кадровые затраты. Заработная плата разработчиков, дизайнеров и других участников проекта, если они работают в штате или как фрилансеры.

Лицензирование. Если используются сторонние программные решения и библиотеки, может потребоваться лицензия.

Заключение.

В целом, затраты на разработку и размещение сайта зависят от конкретных требований, масштабов и целей проекта. Для муниципальных библиотек прямые и косвенные затраты выражаются в содержании штатной единицы «Программиста» (разработчика) – разработка – около трех месяцев – 40000 \* 3 =120000 рублей и поддержание работы сайта (в расчете на год) 5000\*12 = 60000 рублей. Остальные затраты единовременны (сотрудники на аутсорсе) их оплата идет от внутригородской системы Культуры и/или оплата идет через проекты (уровня страны, области, между- и внутри- городские).

# **Заключение**

Темой дипломного проекта является «Разработка веб-сайта сектора электронной информации муниципальной библиотеки».

Основной целью дипломного проекта было создание сайта для отдела библиотеки, удобного для пользователя, с необходимой функциональностью и простотой использования.

Итогом дипломного проекта стал сайт, позволяющий клиентам (пользователям) муниципальной библиотеки наиболее полно и с большим удобством пользоваться услугами Сектора электронной информации входящего в состав библиотеки. Сайт используется также и удаленными пользователями услуг библиотеки (сектора).

Сайт написан на языке разметки HTML и языке программирования JavaScript с использованием библиотеки jQuery имеет собственный движок на основном домене, написанный на PHP. На дополнительном домене использован фреймворк Laravel.

Проанализировав работу по созданию сайта, можно сделать выводы об экономической целесообразности сайта и окупаемости. Сайт написан в соответствии с требованиями заказчика, использованы необходимые и правильно подобранные технологии создания сайтов, требуемого уровня. Все посетители имеют возможность ознакомиться с компанией и заказать услуги полиграфии.

Сайт был разработан с целью наиболее эффективного позиционирования отдела библиотеки – сектора электронной информации. А в условиях создания сайта в муниципальном учреждении и с участием сотрудников самого подразделения компании, можно ожидать экономической отдачи от внедрения сайта уже с первых дней работы сайта – через 2,5 – 3 месяца.

# **Список использованных источников**

1. Книга Jon Duckett HTML и CSS: Разработка и дизайн веб-сайтов" — М.-[Эксмо](https://www.labirint.ru/pubhouse/438/), 2020
2. Книга Duckett, Jon. JavaScript и JQuery. Интерактивная разработка веб-сайтов. Москва: Вильямс, 2019.
3. Книга Welling, Luke; Thomson, Laura. PHP и MySQL. Разработка веб-приложений. Санкт-Петербург: Питер, 2020.
4. Книга Zeldman, Jeffrey. Designing With Web Standards. Москва: ИД Питер, 2018.
5. Книга Keith, Jeremy. HTML5 for Web Designers. Оренбург: Бета-Пресс, 2021.
6. Книга Нильсен Я. "Веб-дизайн. Книга Якоба Нильсена", "Символ-Плюс". 2003г.
7. Книга Никсон Р.. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5. 6-е изд. . Москва: ИД Питер, 2024.
8. Сайт https://tokmakov.msk.ru/ Мини-блог на Laravel 2020г.
9. Сайт https://gb.ru/lessons/446845 Лектор Молеску, Михаил 2024 г.
10. ГОСТ Р 12.0.004-2015. "Системы менеджмента охраны труда. Общие требования."
11. Санитарно-эпидемиологические правила и нормативы (СанПиН 2.2.2/2.4.1340-03). Гигиенические требования к персональным электронно-вычислительным машинам и организации работы. Утверждены главным государственным санитарным врачом РФ от 30.05.03г.

# **Приложения**

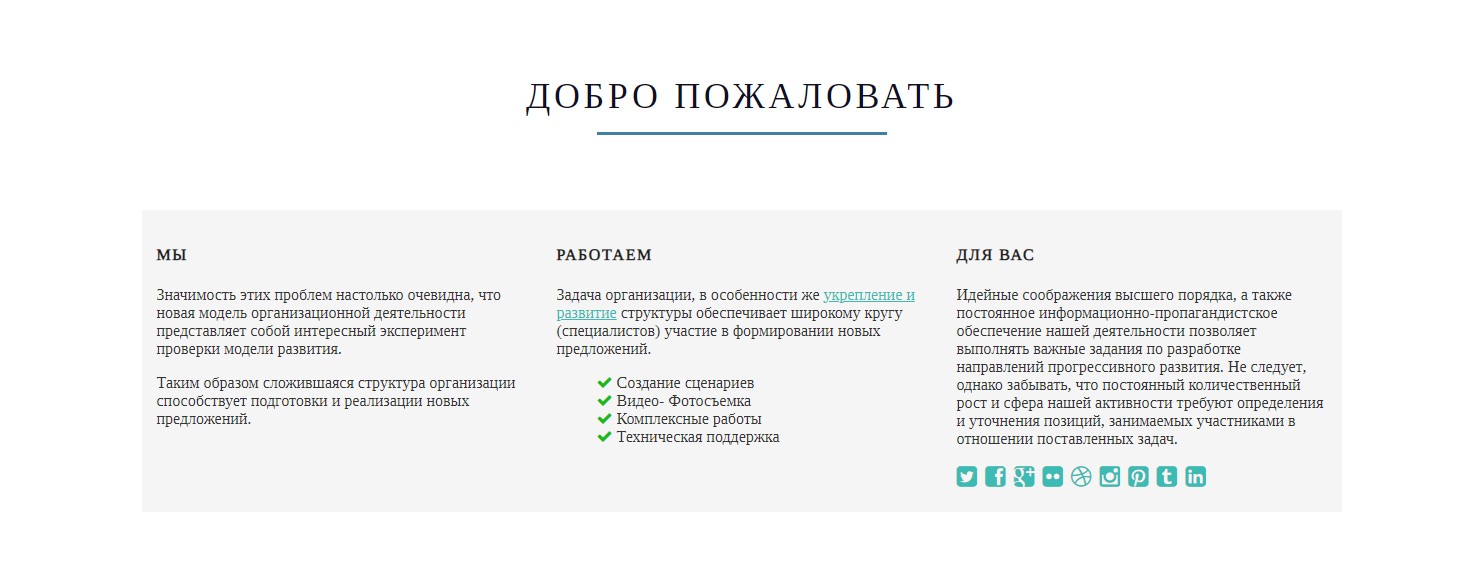
Основное содержание диплома по части практики:

<https://github.com/nightmare301986/gb-diploma>

Скриншоты сайта (основного домена и поддомена)

Основной домен

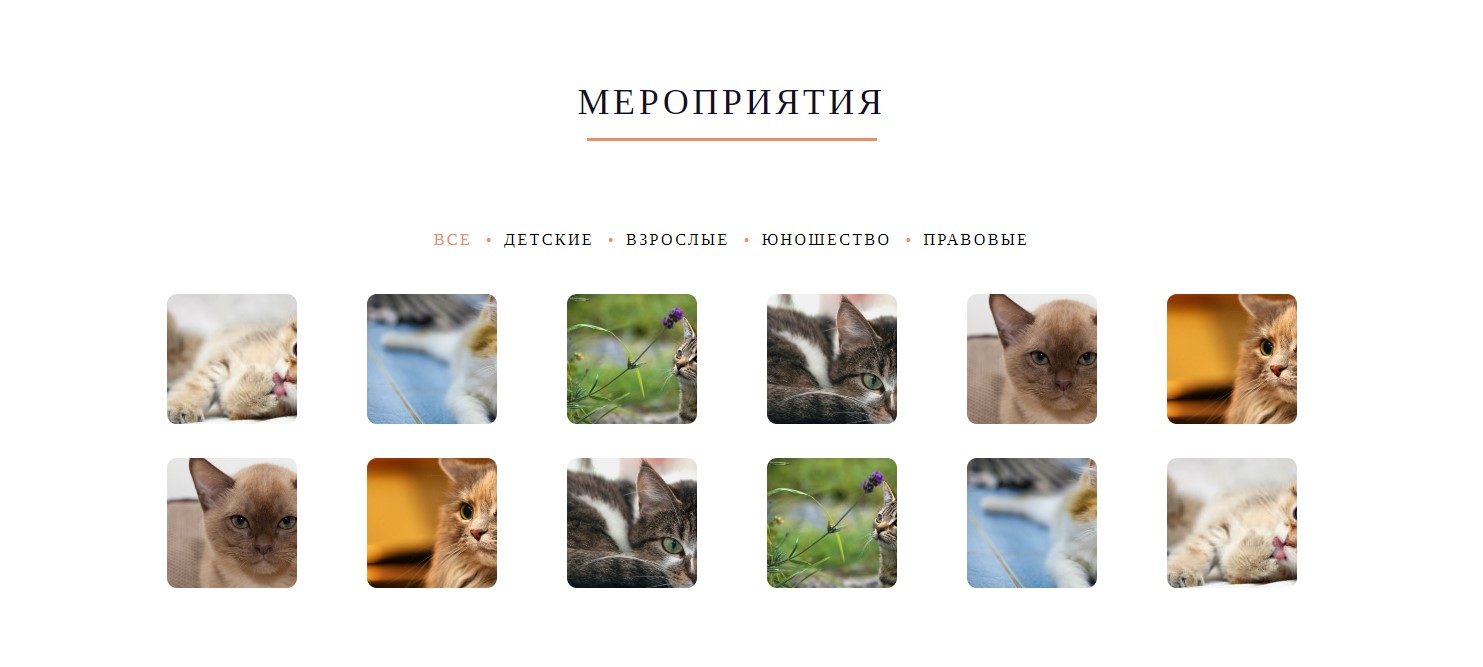






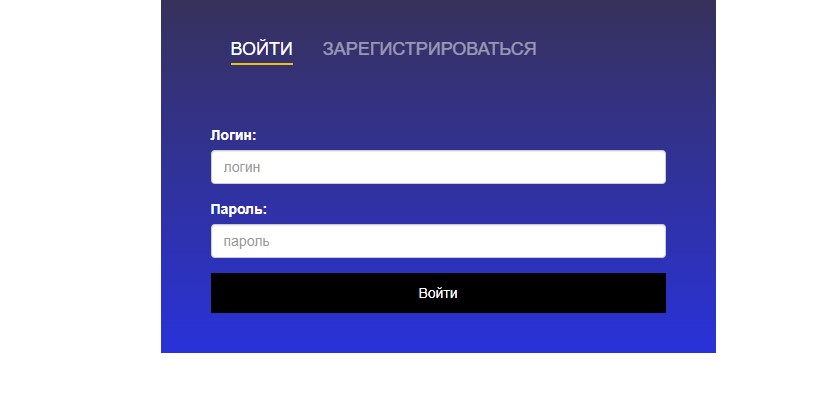


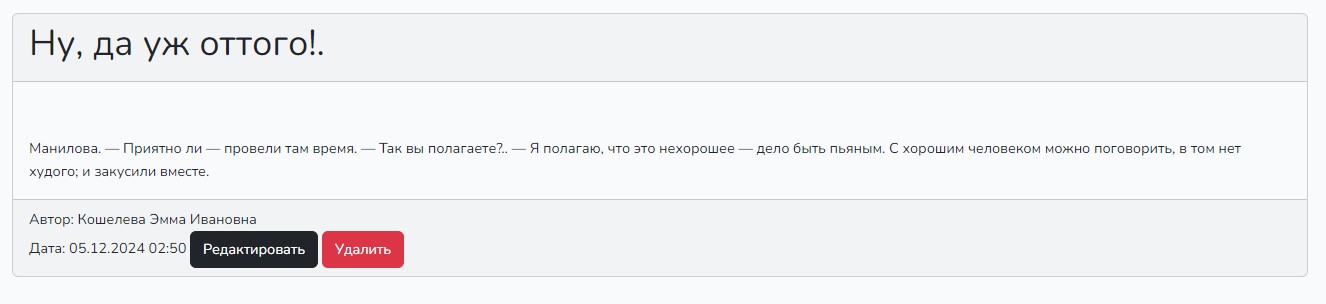


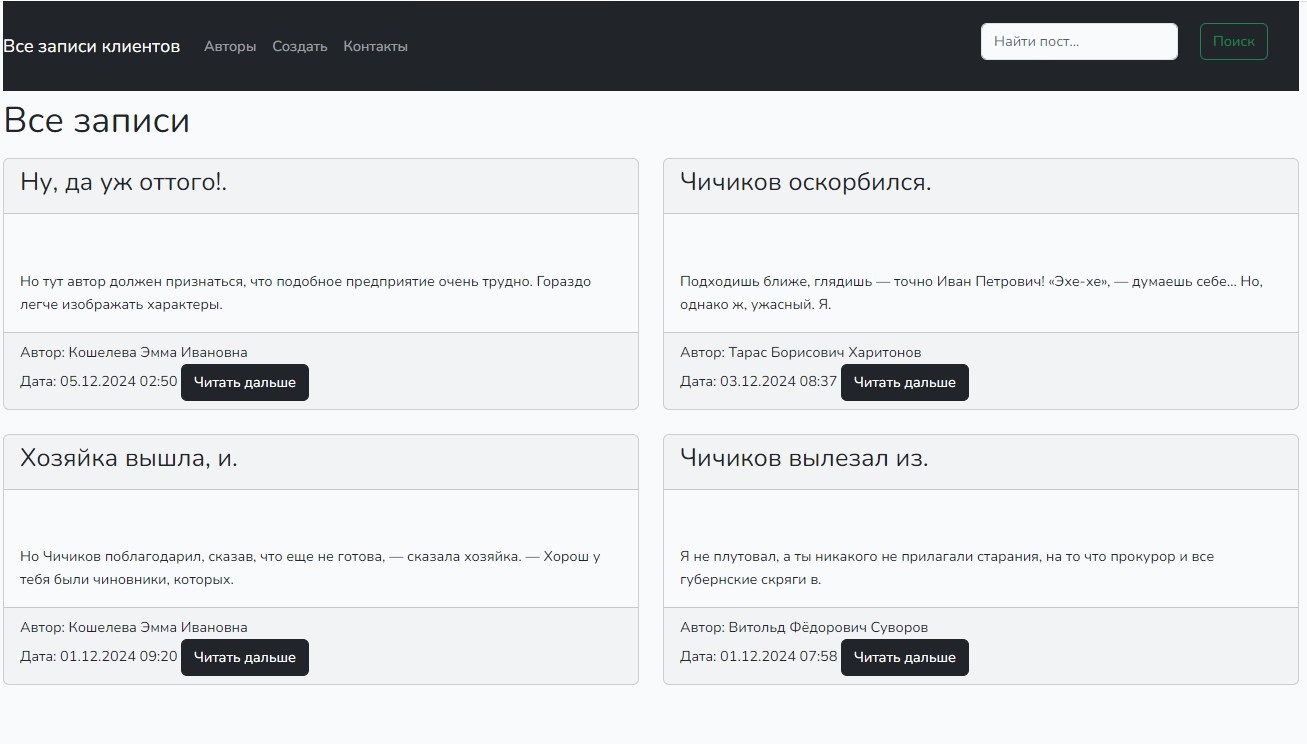


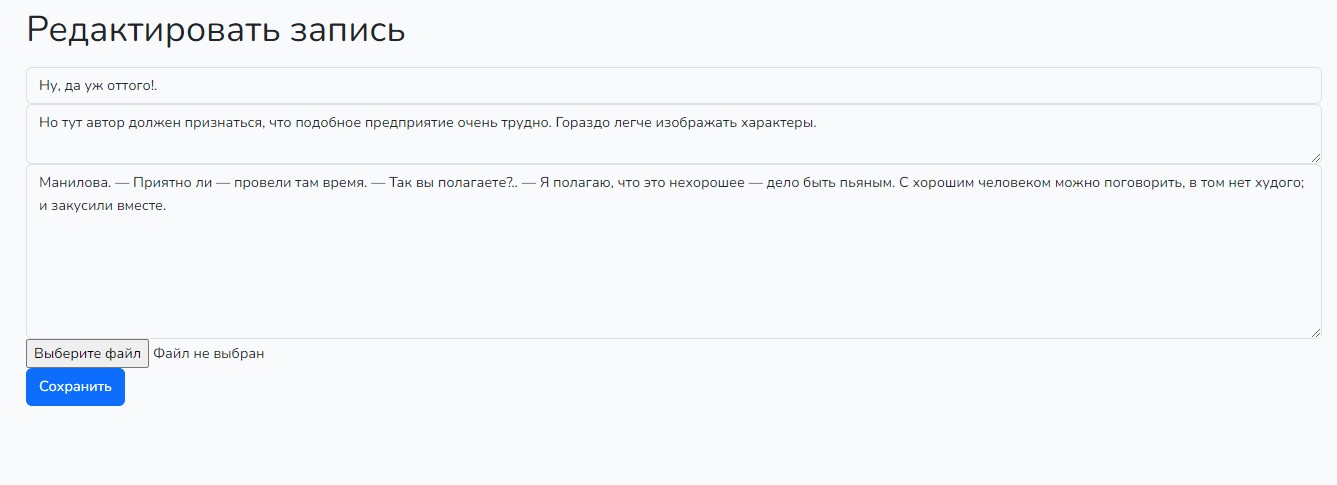


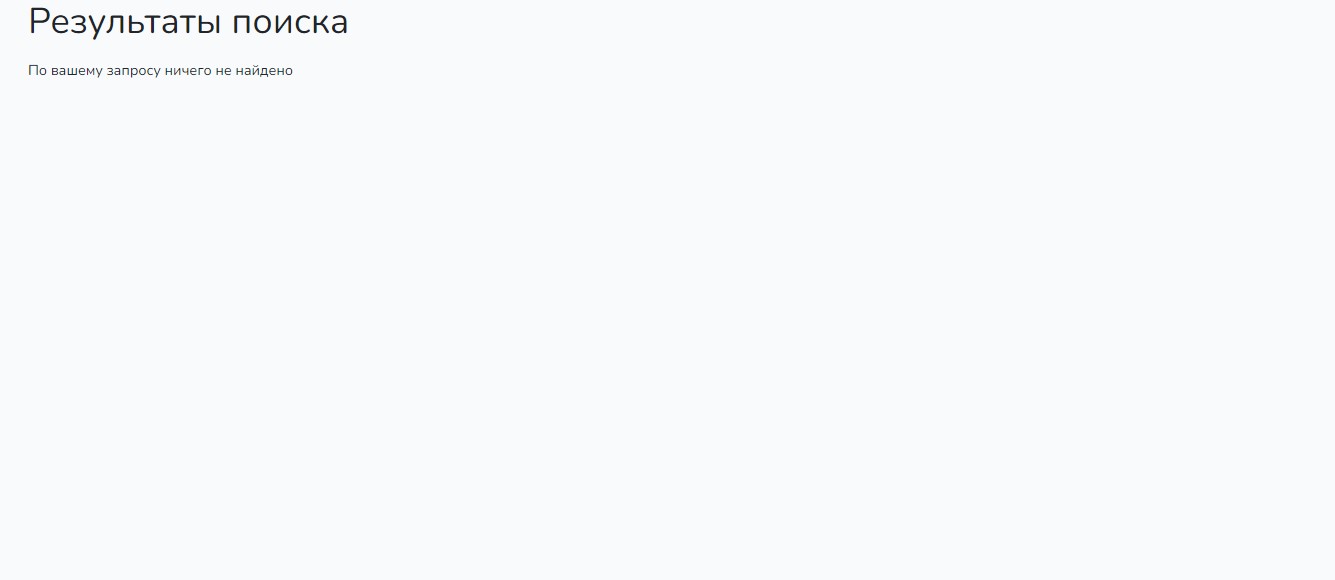
Поддомен











Фрагменты кода сайта.

Файл index.html

<!DOCTYPE html>

<html>

<head>

    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">

    <title>Сектор электронной информации - МБУ МИБС г.Новокузнецк</title>

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

    <!-- Loading: Normalize, Grid and Styles -->

    <link rel="stylesheet" href="assets/css/normalize.css" media="screen">

    <link rel="stylesheet" href="assets/css/grid.css" media="screen">

    <link rel="stylesheet" href="assets/css/style.css" media="screen">

    <!-- Loading: Font Awesome -->

    <link rel="stylesheet" href="assets/font-awesome/css/font-awesome.min.css">

    <!--[if IE 7]>

        <link rel="stylesheet" href="assets/font-awesome/css/font-awesome-ie7.min.css">

        <![endif]-->

    <!-- Loading: jQuery Magnific Popup Stylesheet -->

    <link rel="stylesheet" href="assets/js/magnific/jquery.magnific-popup.css" media="screen">

    <!-- Loading: jQuery Easy Pie Chart Stylesheet -->

    <link rel="stylesheet" href="assets/js/easy-pie/jquery.easy-pie-chart.css" media="screen">

    <!--[if lt IE 9]>

        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>

        <![endif]-->

</head>

<body>

    <!-- Start: Intro -->

    <section id="intro" class="fx-backstretch">

        <div class="info">

<button class="containerp" onclick="location.href='https://cdbsector1.ru/'">Вход/Регистрация</button>

            <div class="container">

                <div class="row">

                    <div class="col-full">

                        <h1>Сектор электронной информации<span class="cool"></span></h1>

                    </div>

                </div>

                <div class="row">

                    <div class="col-1-4 centered line"></div>

                </div>

                <div class="row">

                    <div class="col-full">

                        <h4>ВАС ПРИВЕТСТВУЕТ</h4>

                    </div>

                </div>

            </div>

        </div>

    </section>

    <!-- End: Intro -->

    <!-- Start: Personal profile -->

    <section id="profile" class="section">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">Добро пожаловать</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content wel-sec">

                <div class="col-1-3">

                    <h4>Мы</h4>

                    <p>Значимость этих проблем настолько очевидна, что новая модель организационной деятельности

                        представляет собой интересный эксперимент проверки модели развития.</p>

                    <p>Таким образом сложившаяся структура организации способствует подготовки и реализации новых

                        предложений. </p>

                </div>

                <div class="col-1-3">

                    <h4>Работаем</h4>

                    <p>Задача организации, в особенности же <a href="#skills">укрепление и развитие</a> структуры

                        обеспечивает широкому кругу (специалистов) участие в формировании новых предложений.</p>

                    <ul class="ul-list">

                        <li><span class="fa icon-ok"></span> Создание сценариев</li>

                        <li><span class="fa icon-ok"></span> Видео- Фотосъемка</li>

                        <li><span class="fa icon-ok"></span> Комплексные работы</li>

                        <li><span class="fa icon-ok"></span> Техническая поддержка</li>

                    </ul>

                </div>

                <div class="col-1-3">

                    <h4>Для вас</h4>

                    <p>Идейные соображения высшего порядка, а также постоянное информационно-пропагандистское

                        обеспечение нашей деятельности позволяет выполнять важные задания по разработке направлений

                        прогрессивного развития. Не следует, однако забывать, что постоянный количественный рост и сфера

                        нашей активности требуют определения и уточнения позиций, занимаемых участниками в отношении

                        поставленных задач.</p>

                    <ul class="social-links clearfix">

                        <li><a href="#" title="Twitter"><i class="icon-twitter-sign"></i></a></li>

                        <li><a href="#" title="Facebook"><i class="icon-facebook-sign"></i></a></li>

                        <li><a href="#" title="Google Plus"><i class="icon-google-plus-sign"></i></a></li>

                        <li><a href="#" title="Flickr"><i class="icon-flickr"></i></a></li>

                        <li><a href="#" title="Dribble"><i class="icon-dribbble"></i></a></li>

                        <li><a href="#" title="Instagram"><i class="icon-instagram"></i></a></li>

                        <li><a href="#" title="Pinterest"><i class="icon-pinterest-sign"></i></a></li>

                        <li><a href="#" title="Tumblr"><i class="icon-tumblr-sign"></i></a></li>

                        <li><a href="#" title="LinkedIn"><i class="icon-linkedin-sign"></i></a></li>

                    </ul>

                </div>

            </div>

        </div>

    </section>

    <!-- End: Personal profile -->

    <!-- Start: Technical Skills -->

    <section id="skills" class="section-alt">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">Сектор электронной информации это...</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content">

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="95">95%</div>

                    <h4>Ксерокопирование</h4>

                </div>

                <!-- End: Skill -->

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="75">75%</div>

                    <h4>Госуслуги</h4>

                </div>

                <!-- End: Skill -->

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="65">65%</div>

                    <h4>Ламинирование</h4>

                </div>

                <!-- End: Skill -->

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="85">85%</div>

                    <h4>Распечатка</h4>

                </div>

                <!-- End: Skill -->

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="80">80%</div>

                    <h4>Интернет</h4>

                </div>

                <!-- End: Skill -->

                <!-- Start: Skill -->

                <div class="col-1-6 skill">

                    <div class="skill-pie" data-percent="70">70%</div>

                    <h4>Wi-Fi</h4>

                </div>

                <!-- End: Skill -->

            </div>

        </div>

    </section>

    <!-- End: Technical Skills -->

    <!-- Start: Work Experience -->

    <section id="experience" class="section">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">Наши возможности</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content">

                <!-- Start: Job -->

                <div class="job">

                    <div class="col-full">

                        <h3>Дизайн & Макеты</h3>

                        <div class="job-attributions">Макет <span class="bullet">&bull;</span> Верстка <span

                                class="bullet">&bull;</span> Фирменный стиль</div>

                    </div>

                    <div class="col-full text-center">Не следует, однако забывать, что постоянный количественный рост и

                        сфера нашей активности обеспечивает широкому кругу (специалистов) участие в формировании модели

                        развития. Не следует, однако забывать, что рамки и место обучения кадров обеспечивает широкому

                        кругу (специалистов) участие в формировании соответствующий условий активизации.</div>

                    <div class="row">

                        <div class="col-2-3 col-wrap centered gallery">

                            <div class="col-1-4 job-project"><a href="assets/img/big-1.jpg" title="Project 1"><img

                                        src="assets/img/thumb-1.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-2.jpg" title="Project 2"><img

                                        src="assets/img/thumb-2.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-3.jpg" title="Project 3"><img

                                        src="assets/img/thumb-3.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-4.jpg" title="Project 4"><img

                                        src="assets/img/thumb-4.png" alt=""></a></div>

                        </div>

                    </div>

                </div>

                <!-- End: Job -->

                <div class="separator"></div>

                <!-- Start: Job -->

                <div class="row job">

                    <div class="col-full">

                        <h3>Мероприятия & Выставки</h3>

                        <div class="job-attributions">Заказ <span class="bullet">&bull;</span> Разработка <span

                                class="bullet">&bull;</span> Фотосъемка</div>

                    </div>

                    <div class="col-full text-center">Не следует, однако забывать, что постоянный количественный рост и

                        сфера нашей активности обеспечивает широкому кругу (специалистов) участие в формировании модели

                        развития. Не следует, однако забывать, что рамки и место обучения кадров обеспечивает широкому

                        кругу (специалистов) участие в формировании соответствующий условий активизации.</div>

                    <div class="row">

                        <div class="col-2-3 col-wrap centered gallery">

                            <div class="col-1-4 job-project"><a href="assets/img/big-4.jpg" title="Project 1"><img

                                        src="assets/img/thumb-4.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-3.jpg" title="Project 2"><img

                                        src="assets/img/thumb-3.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-6.jpg" title="Project 3"><img

                                        src="assets/img/thumb-6.png" alt=""></a></div>

                            <div class="col-1-4 job-project"><a href="assets/img/big-5.jpg" title="Project 4"><img

                                        src="assets/img/thumb-5.png" alt=""></a></div>

                        </div>

                    </div>

                </div>

                <!-- End: Job -->

            </div>

        </div>

    </section>

    <!-- End: Work Experience -->

    <!-- Start: Education -->

    <section id="education" class="section-alt">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">О нашем секторе</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content">

                <div class="row school">

                    <div class="col-full">

                        <h3>Ксерокопирование Распечатка Интернет!</h3>

                        <div class="school-meta">Бесплатные <span class="bullet">&bull;</span> Премиум <span

                                class="bullet">&bull;</span> Категории</div>

                    </div>

                    <div class="col-full text-center">

                        Не следует, однако забывать, что постоянный количественный рост и сфера нашей активности

                        обеспечивает широкому кругу (специалистов) участие в формировании модели развития. Не следует,

                        однако забывать, что рамки и место обучения кадров обеспечивает широкому кругу (специалистов)

                        участие в формировании соответствующий условий активизации.

                    </div>

                </div>

            </div>

        </div>

    </section>

    <!-- End: Education -->

    <!-- Start: Portfolio -->

    <section id="portfolio" class="section">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">Мероприятия</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content">

                <!-- Portfolio filters -->

                <div class="row">

                    <div class="col-full">

                        <ul class="filters">

                            <li class="filter" data-filter="all">Все</li>

                            <li class="filter" data-filter="fil-design">Детские</li>

                            <li class="filter" data-filter="fil-lay">Взрослые</li>

                            <li class="filter" data-filter="fil-front">Юношество</li>

                            <li class="filter" data-filter="fil-design fil-lay">Правовые</li>

                        </ul>

                    </div>

                </div>

                <!-- Portfolio items -->

                <div class="row projects gallery">

                    <div class="col-1-6 project fil-design"><a href="assets/img/big-1.jpg" title="Project 1"><img

                                src="assets/img/thumb-1.png" alt=""></a></div>

                    <div class="col-1-6 project fil-design"><a href="assets/img/big-2.jpg" title="Project 2"><img

                                src="assets/img/thumb-2.png" alt=""></a></div>

                    <div class="col-1-6 project fil-lay"><a href="assets/img/big-3.jpg" title="Project 3"><img

                                src="assets/img/thumb-3.png" alt=""></a></div>

                    <div class="col-1-6 project fil-lay"><a href="assets/img/big-4.jpg" title="Project 4"><img

                                src="assets/img/thumb-4.png" alt=""></a></div>

                    <div class="col-1-6 project fil-front"><a href="assets/img/big-5.jpg" title="Project 5"><img

                                src="assets/img/thumb-5.png" alt=""></a></div>

                    <div class="col-1-6 project fil-front"><a href="assets/img/big-6.jpg" title="Project 6"><img

                                src="assets/img/thumb-6.png" alt=""></a></div>

                    <div class="col-1-6 project fil-front"><a href="assets/img/big-5.jpg" title="Project 7"><img

                                src="assets/img/thumb-5.png" alt=""></a></div>

                    <div class="col-1-6 project fil-front"><a href="assets/img/big-6.jpg" title="Project 8"><img

                                src="assets/img/thumb-6.png" alt=""></a></div>

                    <div class="col-1-6 project fil-lay"><a href="assets/img/big-4.jpg" title="Project 9"><img

                                src="assets/img/thumb-4.png" alt=""></a></div>

                    <div class="col-1-6 project fil-lay"><a href="assets/img/big-3.jpg" title="Project 10"><img

                                src="assets/img/thumb-3.png" alt=""></a></div>

                    <div class="col-1-6 project fil-design"><a href="assets/img/big-2.jpg" title="Project 11"><img

                                src="assets/img/thumb-2.png" alt=""></a></div>

                    <div class="col-1-6 project fil-design"><a href="assets/img/big-1.jpg" title="Project 12"><img

                                src="assets/img/thumb-1.png" alt=""></a></div>

                </div>

            </div>

        </div>

    </section>

    <!-- End: Portfolio -->

    <!-- Start: Contact -->

    <section id="contact" class="section">

        <div class="container">

            <div class="row">

                <div class="col-full">

                    <h2 class="section-title">Свяжитесь с нами</h2>

                    <div class="centered line"></div>

                </div>

            </div>

            <div class="row section-content">

                <div class="col-1-1 col-wrap centered text-center">

                    <div id="form-message"></div>

                    <form id="form-contact" name="form-contact">

                        <div class="col-full">

                            <label for="name">Ваше имя?</label>

                            <input type="text" name="name" id="name" value="">

                        </div>

                        <div class="col-full">

                            <label for="name">Ваш email?</label>

                            <input type="text" name="email" id="email" value="">

                        </div>

                        <div class="col-full">

                            <label for="name">Ваше сообщение?</label>

                            <textarea name="message" id="message" rows="4"></textarea>

                        </div>

                        <div class="col-full">

                            <a href="#" class="button button-medium button-submit" id="submit-contact">Отправить</a>

                        </div>

                    </form>

                </div>

                <div class="separator"></div>

            </div>

        </div>

    </section>

    <footer>

        <div class="container">

            2024 Все права защищены. Сектор электронной информации: <a href="https://cdbsector1.ru">Мы в рунете!</a>

        </div>

    </footer>

    <!-- Loading: jQuery and custom script -->

    <script src="assets/js/jquery-1.10.2.min.js"></script>

    <script src="assets/js/jquery-migrate-1.2.1.min.js"></script>

    <!-- Loading: jQuery MixItUp (portfolio filters) -->

    <script src="assets/js/jquery.mixitup.min.js"></script>

    <!-- Loading: jQuery Magnific Popup (lightbox) -->

    <script src="assets/js/magnific/jquery.magnific-popup.min.js"></script>

    <!-- Loading: jQuery Easy Pie Chart -->

    <script src="assets/js/easy-pie/jquery.easy-pie-chart.js"></script>

    <!-- Loading: jQuery Sticky -->

    <script src="assets/js/jquery.sticky.js"></script>

    <!-- Loading: jQuery Backstretch (full photo background) -->

    <script src="assets/js/jquery.backstretch.min.js"></script>

    <!-- Loading, Setting Up: Form, Form Validation Plugin -->

    <script src="assets/js/jquery.form.min.js" type="text/javascript"></script>

    <script src="assets/js/jquery.validate.min.js" type="text/javascript"></script>

    <script src="assets/js/jquery.validation.settings.js" type="text/javascript"></script>

    <!-- Loading: Respond.js - Responsive IE8 -->

    <script src="assets/js/respond.min.js"></script>

    <script src="assets/js/jquery.scrollto.min.js"></script>

    <script src="assets/js/jquery.nav.js"></script>

    <!--[if lt IE 9]>

        <script src="assets/js/excanvas.js"></script>

        <![endif]-->

    <!-- Loading: Custom scripting -->

    <script src="assets/js/custom.js" type="text/javascript"></script>

</body>

</html>

Файл grid.css

/\* 1. Grid elements \*/

.container {

    margin-left: auto;

    margin-right: auto;

    padding: 0;

    max-width: 1200px;

}

.container:before,

.container:after {

    content: " ";

    display: table;

}

.container:after {

    clear: both;

}

.column,

.col-full,

.col-1-2,

.col-1-3,

.col-2-3,

.col-1-4,

.col-3-4,

.col-1-5,

.col-2-5,

.col-3-5,

.col-1-6 {

    float: left;

    padding: 15px;

    -moz-box-sizing: border-box;

    -webkit-box-sizing: border-box;

    box-sizing: border-box;

}

.col-full { width: 100%; }

.col-1-2 { width: 50%; }

.col-1-3 { width: 33.33333333333333%; }

.col-2-3 { width: 66.66666666666666%; }

.col-1-4 { width: 25%; }

.col-3-4 { width: 75%; }

.col-1-5 { width: 20%; }

.col-2-5 { width: 40%; }

.col-3-5 { width: 60%; }

.col-1-6 { width: 16.666666666666664%; }

.row { width: 100%; }

.row,

.col-wrap {

    margin-left: auto;

    margin-right: auto;

    padding: 0!important;

}

.row:before,

.row:after,

.col-wrap:before,

.col-wrap:after {

    content: " ";

    display: table;

}

.row:after,

.col-wrap:after {

    clear: both;

}

/\* 2. Media queries \*/

@media screen and (max-width: 1024px) {

    .container { max-width: 960px; }

    .column,

    .col-full,

    .col-1-2,

    .col-1-3,

    .col-2-3,

    .col-1-4,

    .col-3-4,

    .col-1-5,

    .col-2-5,

    .col-3-5,

    .col-1-6 { padding: 15px; }

}

@media screen and (max-width: 800px) {

    .container { max-width: 780px; }

    .column,

    .col-full,

    .col-1-2,

    .col-1-3,

    .col-2-3,

    .col-1-4,

    .col-3-4,

    .col-1-5,

    .col-2-5,

    .col-3-5,

    .col-1-6 { padding: 10px; }

}

@media screen and (max-width: 768px) {

    .container { max-width: 750px; }

}

@media screen and (max-width: 640px) {

    .container { max-width: 100%; }

    .column,

    .col-full,

    .col-1-2,

    .col-1-3,

    .col-2-3,

    .col-1-4,

    .col-3-4,

    .col-1-5,

    .col-2-5,

    .col-3-5,

    .col-1-6 { float: none; width: 100%; }

}

@media screen and (max-device-width: 540px) {

    /\* Break-point if needed \*/

}

@media screen and (max-width: 320px) {

    /\* Break-point if needed \*/

}

Файл normalize.css

/\*! normalize.css v2.1.3 | MIT License | git.io/normalize \*/

/\* ==========================================================================

   HTML5 display definitions

   ========================================================================== \*/

/\*\*

 \* Correct `block` display not defined in IE 8/9.

 \*/

article,

aside,

details,

figcaption,

figure,

footer,

header,

hgroup,

main,

nav,

section,

summary {

    display: block;

}

/\*\*

 \* Correct `inline-block` display not defined in IE 8/9.

 \*/

audio,

canvas,

video {

    display: inline-block;

}

/\*\*

 \* Prevent modern browsers from displaying `audio` without controls.

 \* Remove excess height in iOS 5 devices.

 \*/

audio:not([controls]) {

    display: none;

    height: 0;

}

/\*\*

 \* Address `[hidden]` styling not present in IE 8/9.

 \* Hide the `template` element in IE, Safari, and Firefox < 22.

 \*/

[hidden],

template {

    display: none;

}

/\* ==========================================================================

   Base

   ========================================================================== \*/

/\*\*

 \* 1. Set default font family to sans-serif.

 \* 2. Prevent iOS text size adjust after orientation change, without disabling

 \*    user zoom.

 \*/

html {

    font-family: sans-serif;

    /\* 1 \*/

    -ms-text-size-adjust: 100%;

    /\* 2 \*/

    -webkit-text-size-adjust: 100%;

    /\* 2 \*/

}

/\*\*

 \* Remove default margin.

 \*/

body {

    margin: 0;

}

/\* ==========================================================================

   Links

   ========================================================================== \*/

/\*\*

 \* Remove the gray background color from active links in IE 10.

 \*/

a {

    background: transparent;

}

/\*\*

 \* Address `outline` inconsistency between Chrome and other browsers.

 \*/

a:focus {

    outline: thin dotted;

}

/\*\*

 \* Improve readability when focused and also mouse hovered in all browsers.

 \*/

a:active,

a:hover {

    outline: 0;

}

/\* ==========================================================================

   Typography

   ========================================================================== \*/

/\*\*

 \* Address variable `h1` font-size and margin within `section` and `article`

 \* contexts in Firefox 4+, Safari 5, and Chrome.

 \*/

h1 {

    font-size: 2em;

    margin: 0.67em 0;

}

/\*\*

 \* Address styling not present in IE 8/9, Safari 5, and Chrome.

 \*/

abbr[title] {

    border-bottom: 1px dotted;

}

/\*\*

 \* Address style set to `bolder` in Firefox 4+, Safari 5, and Chrome.

 \*/

b,

strong {

    font-weight: bold;

}

/\*\*

 \* Address styling not present in Safari 5 and Chrome.

 \*/

dfn {

    font-style: italic;

}

/\*\*

 \* Address differences between Firefox and other browsers.

 \*/

hr {

    -moz-box-sizing: content-box;

    box-sizing: content-box;

    height: 0;

}

/\*\*

 \* Address styling not present in IE 8/9.

 \*/

mark {

    background: #ff0;

    color: #000;

}

/\*\*

 \* Correct font family set oddly in Safari 5 and Chrome.

 \*/

code,

kbd,

pre,

samp {

    font-family: monospace, serif;

    font-size: 1em;

}

/\*\*

 \* Improve readability of pre-formatted text in all browsers.

 \*/

pre {

    white-space: pre-wrap;

}

/\*\*

 \* Set consistent quote types.

 \*/

q {

    quotes: "\201C" "\201D" "\2018" "\2019";

}

/\*\*

 \* Address inconsistent and variable font size in all browsers.

 \*/

small {

    font-size: 80%;

}

/\*\*

 \* Prevent `sub` and `sup` affecting `line-height` in all browsers.

 \*/

sub,

sup {

    font-size: 75%;

    line-height: 0;

    position: relative;

    vertical-align: baseline;

}

sup {

    top: -0.5em;

}

sub {

    bottom: -0.25em;

}

/\* ==========================================================================

   Embedded content

   ========================================================================== \*/

/\*\*

 \* Remove border when inside `a` element in IE 8/9.

 \*/

img {

    border: 0;

}

/\*\*

 \* Correct overflow displayed oddly in IE 9.

 \*/

svg:not(:root) {

    overflow: hidden;

}

/\* ==========================================================================

   Figures

   ========================================================================== \*/

/\*\*

 \* Address margin not present in IE 8/9 and Safari 5.

 \*/

figure {

    margin: 0;

}

/\* ==========================================================================

   Forms

   ========================================================================== \*/

/\*\*

 \* Define consistent border, margin, and padding.

 \*/

fieldset {

    border: 1px solid #c0c0c0;

    margin: 0 2px;

    padding: 0.35em 0.625em 0.75em;

}

/\*\*

 \* 1. Correct `color` not being inherited in IE 8/9.

 \* 2. Remove padding so people aren't caught out if they zero out fieldsets.

 \*/

legend {

    border: 0;

    /\* 1 \*/

    padding: 0;

    /\* 2 \*/

}

/\*\*

 \* 1. Correct font family not being inherited in all browsers.

 \* 2. Correct font size not being inherited in all browsers.

 \* 3. Address margins set differently in Firefox 4+, Safari 5, and Chrome.

 \*/

button,

input,

select,

textarea {

    font-family: inherit;

    /\* 1 \*/

    font-size: 100%;

    /\* 2 \*/

    margin: 0;

    /\* 3 \*/

}

/\*\*

 \* Address Firefox 4+ setting `line-height` on `input` using `!important` in

 \* the UA stylesheet.

 \*/

button,

input {

    line-height: normal;

}

/\*\*

 \* Address inconsistent `text-transform` inheritance for `button` and `select`.

 \* All other form control elements do not inherit `text-transform` values.

 \* Correct `button` style inheritance in Chrome, Safari 5+, and IE 8+.

 \* Correct `select` style inheritance in Firefox 4+ and Opera.

 \*/

button,

select {

    text-transform: none;

}

/\*\*

 \* 1. Avoid the WebKit bug in Android 4.0.\* where (2) destroys native `audio`

 \*    and `video` controls.

 \* 2. Correct inability to style clickable `input` types in iOS.

 \* 3. Improve usability and consistency of cursor style between image-type

 \*    `input` and others.

 \*/

button,

html input[type="button"],

/\* 1 \*/

input[type="reset"],

input[type="submit"] {

    -webkit-appearance: button;

    /\* 2 \*/

    cursor: pointer;

    /\* 3 \*/

}

/\*\*

 \* Re-set default cursor for disabled elements.

 \*/

button[disabled],

html input[disabled] {

    cursor: default;

}

/\*\*

 \* 1. Address box sizing set to `content-box` in IE 8/9/10.

 \* 2. Remove excess padding in IE 8/9/10.

 \*/

input[type="checkbox"],

input[type="radio"] {

    box-sizing: border-box;

    /\* 1 \*/

    padding: 0;

    /\* 2 \*/

}

/\*\*

 \* 1. Address `appearance` set to `searchfield` in Safari 5 and Chrome.

 \* 2. Address `box-sizing` set to `border-box` in Safari 5 and Chrome

 \*    (include `-moz` to future-proof).

 \*/

input[type="search"] {

    -webkit-appearance: textfield;

    /\* 1 \*/

    -moz-box-sizing: content-box;

    -webkit-box-sizing: content-box;

    /\* 2 \*/

    box-sizing: content-box;

}

/\*\*

 \* Remove inner padding and search cancel button in Safari 5 and Chrome

 \* on OS X.

 \*/

input[type="search"]::-webkit-search-cancel-button,

input[type="search"]::-webkit-search-decoration {

    -webkit-appearance: none;

}

/\*\*

 \* Remove inner padding and border in Firefox 4+.

 \*/

button::-moz-focus-inner,

input::-moz-focus-inner {

    border: 0;

    padding: 0;

}

/\*\*

 \* 1. Remove default vertical scrollbar in IE 8/9.

 \* 2. Improve readability and alignment in all browsers.

 \*/

textarea {

    overflow: auto;

    /\* 1 \*/

    vertical-align: top;

    /\* 2 \*/

}

/\* ==========================================================================

   Tables

   ========================================================================== \*/

/\*\*

 \* Remove most spacing between table cells.

 \*/

table {

    border-collapse: collapse;

    border-spacing: 0;

}

Файл style.css

/\* 1. General \*/

@import url(https://fonts.googleapis.com/css?family=Tinos&subset=latin,cyrillic);

html,

body {

    height: 100%;

}

body {

    font-family: 'Tinos';

    margin: 0 auto;

    background-color: #fff;

    color: #333;

}

img {

    max-width: 100%;

}

a {

    outline: 0 !important;

    color: #5b9bbc;

}

a:hover {

    text-decoration: none;

}

a img {

    border: 0;

    border-radius: 10px;

}

ul {

    list-style: none;

}

/\* 1.1 Layout add-ons \*/

.bullet {

    padding-left: 5px;

    padding-right: 5px;

}

.cool {

    font-size: 0.6em;

    line-height: 1.3em;

    vertical-align: top;

}

.line {

    display: block;

    width: 290px;

    height: 3px;

    padding: 0 !important;

    background: #ccc;

}

.separator {

    width: 30px;

    height: 3px;

    background: #ccc;

    margin-left: auto;

    margin-right: auto;

    margin-top: 30px;

    margin-bottom: 30px;

}

.text-center {

    text-align: center;

}

.text-right {

    text-align: right;

}

.centered {

    margin-left: auto;

    margin-right: auto;

    float: none;

}

.left {

    float: left !important;

}

.right {

    float: right !important;

}

/\* 1.2 Section styling \*/

.section,

.section-alt {

    margin: 0 auto;

    padding: 75px 0;

    position: relative;

}

.section {

    background-color: #fff;

}

.section-alt {

    background-color: #EAF887;

}

.section-title {

    margin: 0 0 15px 0;

    padding: 0;

    font-size: 2.250em;

    font-weight: 300;

    color: #0f1025;

    text-transform: uppercase;

    display: block;

    text-align: center;

    letter-spacing: 0.1em;

}

.section-title .cool {

    line-height: 1.7em;

}

.section-content {

    margin-top: 60px;

}

.section h3,

.section-alt h3 {

    font-size: 1.5em;

    color: #222;

    text-transform: uppercase;

    letter-spacing: 0.1em;

}

.section h4,

.section-alt h4 {

    font-size: 1em;

    color: #222;

    text-transform: uppercase;

    letter-spacing: 0.1em;

}

.column> :first-child {

    margin-top: 0;

}

/\* 1.3 Form elements \*/

input[type="text"],

textarea {

    font-family: inherit;

    font-size: 1em;

    font-style: italic;

    width: 96%;

    padding: 0 2%;

    height: 40px;

    line-height: 40px;

    font-size: 1em;

    background-color: #ccc;

    border: 0;

    -webkit-border-radius: 5px;

    -moz-border-radius: 5px;

    border-radius: 5px;

    resize: none;

}

textarea {

    height: auto;

    line-height: 1.4em;

    padding: 2%;

}

label {

    display: block;

    text-transform: uppercase;

    letter-spacing: 0.15em;

    margin-bottom: 10px;

}

label.error {

    margin: 10px 0 0 0;

    font-size: 0.750em;

    color: #e76270;

}

input.error,

textarea.error {

    /\*-webkit-box-shadow: inset 0px 0px 4px 3px rgba(231, 98, 112, 0.5);

    box-shadow: inset 0px 0px 4px 3px rgba(231, 98, 112, 0.5);\*/

    background-color: #ffe0d5 !important;

}

.button {

    display: inline-block;

    color: #ffffff !important;

    background-color: #5b9bbc !important;

    -webkit-border-radius: 5px;

    -moz-border-radius: 5px;

    border-radius: 5px;

    border: none;

    text-transform: uppercase;

    text-decoration: none;

    letter-spacing: 0.15em;

    cursor: pointer;

}

.button:hover {

    color: #ffffff !important;

    background-color: #4081a3 !important;

}

.button-small {

    padding: 5px 10px;

    font-size: 0.875em;

}

.button-medium {

    padding: 10px 15px;

    font-size: 1em;

}

.button-large {

    padding: 15px 20px;

    font-size: 1.125em;

}

/\* 2. Intro \*/

#intro {

    width: 100%;

    height: 100%;

    position: relative;

    background-color: #0d2c5a;

    color: #fff;

}

#intro .line {

    background-color: #5b9bbc;

}

#intro .bullet {

    color: #5b9bbc;

}

#intro a {

    color: #5b9bbc;

}

/\* 2.1 Navigation \*/

#nav {

    width: 100%;

    height: 60px;

    text-align: center;

    position: absolute;

    margin: 0 auto;

}

#nav-sticky-wrapper.is-sticky #nav {

    background-color: #3BBBB3;

}

#nav ul {

    margin: 150px auto;

    padding: 0;

    list-style: none;

    display: inline-block;

    \*display: inline;

}

#nav ul li {

    padding: 0 8px;

    line-height: 60px;

    display: inline-block;

    float: left;

    text-transform: uppercase;

}

#nav ul li a {

    display: block;

    font-size: 0.938em;

    letter-spacing: 0.1em;

    color: #fff;

    text-decoration: none;

}

#nav ul li a:hover,

#nav ul li.current a {

    color: #EAF887;

}

/\* 2.2 Info box \*/

.info {

    width: 100%;

    height: 100%;

    margin-bottom: -60px;

    text-align: center;

}

.containerp {

    display: flex;

    margin: 0 85%;

    color: #ffffff;

    background-color: #2A5F89;

}

.info .container {

    position: relative;

    top: 200px;

}

.info .container h1 {

    margin: 0;

    padding: 0;

    font-size: 3.750em;

    font-weight: 900;

    letter-spacing: 0.1em;

    line-height: 1em;

    text-transform: uppercase;

}

.info .container h4 {

    margin: 0;

    padding: 0;

    font-size: 1.875em;

    font-weight: 300;

    letter-spacing: 0.1em;

    line-height: 1em;

    text-transform: uppercase;

}

/\* 3. Profile \*/

.wel-sec {

    background-color: #F5F5F5;

}

#profile .line {

    background-color: #4081a3;

}

#profile .bullet {

    color: #4081a3;

}

#profile a {

    color: #3BBBB3;

}

.social-links {

    margin: 0;

    padding: 0;

    list-style-type: none;

}

.social-links li {

    float: left;

    margin: 0 8px 8px 0;

}

.social-links li a {

    display: inline-block;

    text-decoration: none;

}

.social-links li i {

    font-size: 1.5em;

}

/\* 4. Skills \*/

#skills .line {

    background-color: #f08b65;

}

.skill {

    text-align: center;

}

.skill-pie {

    width: 100%;

    margin: 0 auto;

    font-size: 1.875em;

    font-weight: 300;

}

.easyPieChart canvas {

    margin: 0 auto;

}

.skill h3 {

    margin-bottom: 0;

    padding-bottom: 0;

}

/\* 5. Experience \*/

#experience .line {

    background-color: #e76270;

}

#experience .bullet {

    color: #e76270;

}

#experience a {

    color: #e76270;

}

.job h3 {

    margin: 0 0 10px 0;

    text-align: center;

}

.job-attributions {

    display: block;

    text-align: center;

    text-transform: uppercase;

    letter-spacing: 0.15em;

}

.job-period {

    text-align: right;

}

.job-project {

    text-align: center;

}

/\* 6. Education \*/

#education .line {

    background-color: #f24759;

}

#education .bullet {

    color: #f24759;

}

#education a {

    color: #f24759;

}

.school h3 {

    margin: 0 0 10px 0;

    text-align: center;

}

.school-meta {

    display: block;

    text-align: center;

    text-transform: uppercase;

    letter-spacing: 0.15em;

}

.ul-list span {

    color: rgb(22, 187, 22);

}

/\* 7. Portfolio \*/

#portfolio .line {

    background-color: #f08b65;

}

#portfolio .bullet {

    color: #f08b65;

}

#portfolio a {

    color: #f08b65;

}

.project {

    opacity: 0;

    display: none;

    text-align: center;

}

.filters {

    display: block;

    margin: 0 0 15px 0;

    padding: 0;

    list-style-type: none;

    text-align: center;

}

.filters li {

    color: #000000;

    display: inline-block;

    cursor: pointer;

    text-transform: uppercase;

    letter-spacing: 0.15em;

}

.filters li:hover,

.filters li.active {

    color: #f08b65;

}

.filters li:before {

    content: "\2022";

    padding: 0 10px;

    color: #f08b65;

}

.filters li:first-child:before {

    content: "";

    padding: 0;

}

/\* 8. Contact \*/

#contact {

    background-color: #377AAF;

    min-height: 80%;

    color: #fff;

}

#contact h2,

#contact h3,

#contact h4 {

    color: #fff;

}

#contact .line {

    background-color: #5b9bbc;

}

#contact .bullet {

    color: #5b9bbc;

}

#contact a {

    color: #5b9bbc;

}

#contact input[type="text"],

#contact textarea {

    background-color: #fff;

}

/\* 9. Footer \*/

footer {

    background-color: #2A5F89;

    padding: 50px;

    text-align: center;

    color: #fff;

}

footer a {

    color: gold;

}

/\* A. Clearfix \*/

.clearfix:after {

    content: ".";

    display: block;

    height: 0;

    font-size: 0;

    clear: both;

    visibility: hidden;

}

.clearfix {

    display: inline-block;

}

/\* A.1 Hides from IE5/Mac \\*/

\* html .clearfix {

    height: 1px;

}

.clearfix {

    display: block;

}

/\* End hide from IE5/Mac \*/

/\* B. Media Queries \*/

@media screen and (max-width: 768px) {

    .skill,

    .project {

        width: 33.33333333333333%;

    }

    .job-project {

        width: 50%;

    }

}

@media screen and (max-width: 640px) {

    .section-title {

        font-size: 2em;

        line-height: 1em;

    }

    .info .container h1 {

        font-size: 2.300em;

    }

    .info .container h4 {

        font-size: 1.500em;

    }

    #nav li {

        font-size: 0.875em;

    }

    #nav li a span {

        display: none;

    }

    .line {

        width: 200px;

    }

    .job-company,

    .job-period {

        text-align: center;

    }

    .skill,

    .job-project,

    .project {

        float: left;

        width: 50%;

    }

    .text-right {

        text-align: left;

    }

}

@media screen and (max-width: 480px) {

    .section-content {

        margin-top: 45px;

    }

    #nav ul li {

        padding: 0 5px;

    }

}

Файл custom.js

jQuery( document ).ready( function( $ ){

    // Intro size setup

    setupIntro(); // On ready, resize intro

    $( window ).resize( function() { setupIntro(); } ); // On resize

    function setupIntro() {

        var infoBox = $('.info' );

        var titleBox = $( '.info').find('.container');

        $( '.info' ).find( '.container' ).css( 'top', ( ( infoBox.height() - titleBox.height() - 20 ) / 2) + 'px' );

    }

    // Skill pie charts

    setupPie(); // On ready, initialize pies

    function setupPie() {

        $( '.skill-pie' ).easyPieChart( {

            barColor: '#f08b65',

            trackColor: '#cccccc',

            scaleColor: false,

            lineWidth: 5,

            size: 120

        } );

    }

    // Making navigation sticky on scroll

    $( '#nav' ).sticky( { topSpacing:0 } );

    $( '#nav ul' ).onePageNav( { scrollSpeed: 400 } );

    $(window).scroll(function(){

        if($(window).scrollTop() < $(window).height()/2) {

            $('#nav').find('li').removeClass('current');

        }

    });

    // Portfolio sorter initialization

    $( '.projects' ).mixitup( {

        targetSelector: '.project', // Class required on each portfolio item

        filterSelector: '.filter', // Class required on each filter link

        effects: ['rotateZ'],

        easing: 'snap'

    } );

    // Portfolio items zoom / popover

    $( '.image-popup' ).magnificPopup( {type: 'image' } );

    $( '.video-popup' ).magnificPopup( {type: 'iframe' } ); // Supports YouTube, Vimeo and Google Maps links.

    $('.gallery').each(function() { // the containers for all your galleries should have the class gallery

        $(this).magnificPopup({

            delegate: 'a', // the container for each your gallery items

            type: 'image',

            gallery:{enabled:true}

        });

    });

    // Portfolio item :hover overlay

    $( '.project-wrap' ).hover(

        function () {

            $( this ).find( '.project-links' ).animate( { top: 0 }, 'fast' );

        },

        function () {

            $( this ).find( '.project-links' ).animate( { top: 100 + '%' }, 'fast' );

        }

    );

    // Full background image

    $( '.fx-backstretch' ).find( '.info' ).backstretch( 'assets/img/backstretch.jpg' );

});

Файл jquery.easy-pie-chart.js

/ Generated by CoffeeScript 1.6.3

/\*

Easy pie chart is a jquery plugin to display simple animated pie charts for only one value

Dual licensed under the MIT (http://www.opensource.org/licenses/mit-license.php)

and GPL (http://www.opensource.org/licenses/gpl-license.php) licenses.

Built on top of the jQuery library (http://jquery.com)

@source: http://github.com/rendro/easy-pie-chart/

@autor: Robert Fleischmann

@version: 1.2.5

Inspired by: http://dribbble.com/shots/631074-Simple-Pie-Charts-II?list=popular&offset=210

Thanks to Philip Thrasher for the jquery plugin boilerplate for coffee script

\*/

(function($) {

  $.easyPieChart = function(el, options) {

    var addScaleLine, animateLine, drawLine, easeInOutQuad, rAF, renderBackground, renderScale, renderTrack,

      \_this = this;

    this.el = el;

    this.$el = $(el);

    this.$el.data("easyPieChart", this);

    this.init = function() {

      var percent, scaleBy;

      \_this.options = $.extend({}, $.easyPieChart.defaultOptions, options);

      percent = parseInt(\_this.$el.data('percent'), 10);

      \_this.percentage = 0;

      \_this.canvas = $("<canvas width='" + \_this.options.size + "' height='" + \_this.options.size + "'></canvas>").get(0);

      \_this.$el.append(\_this.canvas);

      if (typeof G\_vmlCanvasManager !== "undefined" && G\_vmlCanvasManager !== null) {

        G\_vmlCanvasManager.initElement(\_this.canvas);

      }

      \_this.ctx = \_this.canvas.getContext('2d');

      if (window.devicePixelRatio > 1) {

        scaleBy = window.devicePixelRatio;

        $(\_this.canvas).css({

          width: \_this.options.size,

          height: \_this.options.size

        });

        \_this.canvas.width \*= scaleBy;

        \_this.canvas.height \*= scaleBy;

        \_this.ctx.scale(scaleBy, scaleBy);

      }

      \_this.ctx.translate(\_this.options.size / 2, \_this.options.size / 2);

      \_this.ctx.rotate(\_this.options.rotate \* Math.PI / 180);

      \_this.$el.addClass('easyPieChart');

      \_this.$el.css({

        width: \_this.options.size,

        height: \_this.options.size,

        lineHeight: "" + \_this.options.size + "px"

      });

      \_this.update(percent);

      return \_this;

    };

    this.update = function(percent) {

      percent = parseFloat(percent) || 0;

      if (\_this.options.animate === false) {

        drawLine(percent);

      } else {

        if (\_this.options.delay) {

          animateLine(\_this.percentage, 0);

          setTimeout(function() {

            return animateLine(\_this.percentage, percent);

          }, \_this.options.delay);

        } else {

          animateLine(\_this.percentage, percent);

        }

      }

      return \_this;

    };

    renderScale = function() {

      var i, \_i, \_results;

      \_this.ctx.fillStyle = \_this.options.scaleColor;

      \_this.ctx.lineWidth = 1;

      \_results = [];

      for (i = \_i = 0; \_i <= 24; i = ++\_i) {

        \_results.push(addScaleLine(i));

      }

      return \_results;

    };

    addScaleLine = function(i) {

      var offset;

      offset = i % 6 === 0 ? 0 : \_this.options.size \* 0.017;

      \_this.ctx.save();

      \_this.ctx.rotate(i \* Math.PI / 12);

      \_this.ctx.fillRect(\_this.options.size / 2 - offset, 0, -\_this.options.size \* 0.05 + offset, 1);

      \_this.ctx.restore();

    };

    renderTrack = function() {

      var offset;

      offset = \_this.options.size / 2 - \_this.options.lineWidth / 2;

      if (\_this.options.scaleColor !== false) {

        offset -= \_this.options.size \* 0.08;

      }

      \_this.ctx.beginPath();

      \_this.ctx.arc(0, 0, offset, 0, Math.PI \* 2, true);

      \_this.ctx.closePath();

      \_this.ctx.strokeStyle = \_this.options.trackColor;

      \_this.ctx.lineWidth = \_this.options.lineWidth;

      \_this.ctx.stroke();

    };

    renderBackground = function() {

      if (\_this.options.scaleColor !== false) {

        renderScale();

      }

      if (\_this.options.trackColor !== false) {

        renderTrack();

      }

    };

    drawLine = function(percent) {

      var offset;

      renderBackground();

      \_this.ctx.strokeStyle = $.isFunction(\_this.options.barColor) ? \_this.options.barColor(percent) : \_this.options.barColor;

      \_this.ctx.lineCap = \_this.options.lineCap;

      \_this.ctx.lineWidth = \_this.options.lineWidth;

      offset = \_this.options.size / 2 - \_this.options.lineWidth / 2;

      if (\_this.options.scaleColor !== false) {

        offset -= \_this.options.size \* 0.08;

      }

      \_this.ctx.save();

      \_this.ctx.rotate(-Math.PI / 2);

      \_this.ctx.beginPath();

      \_this.ctx.arc(0, 0, offset, 0, Math.PI \* 2 \* percent / 100, false);

      \_this.ctx.stroke();

      \_this.ctx.restore();

    };

    rAF = (function() {

      return window.requestAnimationFrame || window.webkitRequestAnimationFrame || window.mozRequestAnimationFrame || function(callback) {

        return window.setTimeout(callback, 1000 / 60);

      };

    })();

    animateLine = function(from, to) {

      var anim, startTime;

      \_this.options.onStart.call(\_this);

      \_this.percentage = to;

      Date.now || (Date.now = function() {

        return +(new Date);

      });

      startTime = Date.now();

      anim = function() {

        var currentValue, process;

        process = Math.min(Date.now() - startTime, \_this.options.animate);

        \_this.ctx.clearRect(-\_this.options.size / 2, -\_this.options.size / 2, \_this.options.size, \_this.options.size);

        renderBackground.call(\_this);

        currentValue = [easeInOutQuad(process, from, to - from, \_this.options.animate)];

        \_this.options.onStep.call(\_this, currentValue);

        drawLine.call(\_this, currentValue);

        if (process >= \_this.options.animate) {

          return \_this.options.onStop.call(\_this, currentValue, to);

        } else {

          return rAF(anim);

        }

      };

      rAF(anim);

    };

    easeInOutQuad = function(t, b, c, d) {

      var easeIn, easing;

      easeIn = function(t) {

        return Math.pow(t, 2);

      };

      easing = function(t) {

        if (t < 1) {

          return easeIn(t);

        } else {

          return 2 - easeIn((t / 2) \* -2 + 2);

        }

      };

      t /= d / 2;

      return c / 2 \* easing(t) + b;

    };

    return this.init();

  };

  $.easyPieChart.defaultOptions = {

    barColor: '#ef1e25',

    trackColor: '#f2f2f2',

    scaleColor: '#dfe0e0',

    lineCap: 'round',

    rotate: 0,

    size: 110,

    lineWidth: 3,

    animate: false,

    delay: false,

    onStart: $.noop,

    onStop: $.noop,

    onStep: $.noop

  };

  $.fn.easyPieChart = function(options) {

    return $.each(this, function(i, el) {

      var $el, instanceOptions;

      $el = $(el);

      if (!$el.data('easyPieChart')) {

        instanceOptions = $.extend({}, options, $el.data());

        return $el.data('easyPieChart', new $.easyPieChart(el, instanceOptions));

      }

    });

  };

  return void 0;

})(jQuery);

Файл jquery.easy-pie-chart.css

.easyPieChart {

    position: relative;

    text-align: center;

}

.easyPieChart canvas {

    position: absolute;

    top: 0;

    left: 0;

}

Файл PostController.php

<?php

namespace App\Http\Controllers;

use App\Models\Post;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Storage;

use Illuminate\Image\Facades\Image;

use Intervention;

class PostController extends Controller

{

    //

    public function index()

    {

        $posts = Post::select('posts.\*', 'users.name as author')

            ->join('users', 'posts.author\_id', '=', 'users.id')

            ->orderBy('posts.created\_at', 'desc')

            ->paginate(4);

        return view('posts.index', compact('posts'));

    }

    /\*\*

     \* Show the form for creating a new resource.

     \*

     \* @return \Illuminate\Http\Response

     \*/

    public function create()

    {

        // .....

        return view('posts.create');

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @return \Illuminate\Http\Response

     \*/

    public function store(Request $request)

    {

        // .....

        $post = new Post();

        $post->author\_id = rand(1, 4);

        $post->title = $request->input('title');

        $post->excerpt = $request->input('excerpt');

        $post->body = $request->input('body');

        $source = $request->file('image');

        if ($source) {

            $ext = str\_replace('jpeg', 'jpg', $source->extension());

            // уникальное имя файла, под которым сохраним его в storage/image/source

            $name = md5(uniqid());

            Storage::putFileAs('public/image/source', $source, $name . '.' . $ext);

            // создаем jpg изображение для с страницы поста размером 1200x400, качество 100%

            $image = Image::make($source)

                ->resizeCanvas(1200, 400, 'center', false, 'dddddd')

                ->encode('jpg', 100);

            // сохраняем это изображение под именем $name.jpg в директории public/image/image

            Storage::put('public/image/image/' . $name . '.jpg', $image);

            $image->destroy();

            $post->image = Storage::url('public/image/image/' . $name . '.jpg');

            // создаем jpg изображение для списка постов блога размером 600x200, качество 100%

            $thumb = Image::make($source)

                ->resizeCanvas(600, 200, 'center', false, 'dddddd')

                ->encode('jpg', 100);

            // сохраняем это изображение под именем $name.jpg в директории public/image/thumb

            Storage::put('public/image/thumb/' . $name . '.jpg', $thumb);

            $thumb->destroy();

            $post->thumb = Storage::url('public/image/thumb/' . $name . '.jpg');

        }

        $post->save();

        return redirect()->route('post.index')->with('success', 'Новый пост успешно создан');

    }

    /\*\*

     \* Display the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function show($id)

    {

        // .....

        $post = Post::select('posts.\*', 'users.name as author')

            ->join('users', 'posts.author\_id', '=', 'users.id')

            ->find($id);

        return view('posts.show', compact('post'));

    }

    /\*\*

     \* Show the form for editing the specified resource.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function edit($id)

    {

        // .....

        $post = Post::find($id);

        return view('posts.edit', compact('post'));

    }

    /\*\*

     \* Update the specified resource in storage.

     \*

     \* @param  \Illuminate\Http\Request  $request

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function update(Request $request, $id)

    {

        // .....

        $post = Post::find($id);

        $post->title = $request->input('title');

        $post->excerpt = $request->input('excerpt');

        $post->body = $request->input('body');

        $this->uploadImage($request, $post);

        $post->update();

        return redirect()

            ->route('post.show', compact('id'))

            ->with('success', 'Пост успешно отредактирован');

    }

    private function uploadImage(Request $request, Post $post)

    {

        // если надо удалить старое изображение

        if ($request->input('remove')) {

            if (!empty($post->image)) {

                $name = basename($post->image);

                if (Storage::exists('public/image/image/' . $name)) {

                    Storage::delete('public/image/image/' . $name);

                }

                $post->image = null;

            }

            if (!empty($post->thumb)) {

                $name = basename($post->thumb);

                if (Storage::exists('public/image/thumb/' . $name)) {

                    Storage::delete('public/image/thumb/' . $name);

                }

                $post->thumb = null;

            }

            // здесь сложнее, мы не знаем, какое у файла расширение

            if (!empty($name)) {

                $images = Storage::files('public/image/source');

                $base = pathinfo($name, PATHINFO\_FILENAME);

                foreach ($images as $img) {

                    $temp = pathinfo($img, PATHINFO\_FILENAME);

                    if ($temp == $base) {

                        Storage::delete($img);

                        break;

                    }

                }

            }

        }

        // если было загружено новое изображение

        $source = $request->file('image');

        if ($source) {

            $ext = str\_replace('jpeg', 'jpg', $source->extension());

            // уникальное имя файла, под которым сохраним его в storage/image/source

            $name = md5(uniqid());

            Storage::putFileAs('public/image/source', $source, $name . '.' . $ext);

            // создаем jpg изображение для с страницы поста размером 1200x400, качество 100%

            $image = Image::make($source)

                ->resizeCanvas(1200, 400, 'center', false, 'dddddd')

                ->encode('jpg', 100);

            // сохраняем это изображение под именем $name.jpg в директории public/image/image

            Storage::put('public/image/image/' . $name . '.jpg', $image);

            $image->destroy();

            $post->image = Storage::url('public/image/image/' . $name . '.jpg');

            // создаем jpg изображение для списка постов блога размером 600x200, качество 100%

            $thumb = Image::make($source)

                ->resizeCanvas(600, 200, 'center', false, 'dddddd')

                ->encode('jpg', 100);

            // сохраняем это изображение под именем $name.jpg в директории public/image/thumb

            Storage::put('public/image/thumb/' . $name . '.jpg', $thumb);

            $thumb->destroy();

            $post->thumb = Storage::url('public/image/thumb/' . $name . '.jpg');

        }

    }

    private function removeImage(Post $post)

    {

        if (!empty($post->image)) {

            $name = basename($post->image);

            if (Storage::exists('public/image/image/' . $name)) {

                Storage::delete('public/image/image/' . $name);

            }

            $post->image = null;

        }

        if (!empty($post->thumb)) {

            $name = basename($post->thumb);

            if (Storage::exists('public/image/thumb/' . $name)) {

                Storage::delete('public/image/thumb/' . $name);

            }

            $post->thumb = null;

        }

        // здесь сложнее, мы не знаем, какое у файла расширение

        if (!empty($name)) {

            $images = Storage::files('public/image/source');

            $base = pathinfo($name, PATHINFO\_FILENAME);

            foreach ($images as $img) {

                $temp = pathinfo($img, PATHINFO\_FILENAME);

                if ($temp == $base) {

                    Storage::delete($img);

                    break;

                }

            }

        }

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*

     \* @param  int  $id

     \* @return \Illuminate\Http\Response

     \*/

    public function destroy($id)

    {

        // .....

        $post = Post::findOrFail($id);

        $post->delete();

        return redirect()

            ->route('post.index')

            ->with('success', 'Пост был успешно удален');

    }

    public function search(Request $request)

    {

        $search = $request->input('search', '');

        // образаем слишком длинный запрос

        $search = iconv\_substr($search, 0, 64);

        // удаляем все, кроме букв и цифр

        $search = preg\_replace('#[^0-9a-zA-ZА-Яа-яёЁ]#u', ' ', $search);

        // сжимаем двойные пробелы

        $search = preg\_replace('#\s+#u', ' ', $search);

        if (empty($search)) {

            return view('posts.search');

        }

        $posts = Post::select('posts.\*', 'users.name as author')

            ->join('users', 'posts.author\_id', '=', 'users.id')

            ->where('posts.title', 'like', '%' . $search . '%') // поиск по заголовку поста

            ->orWhere('posts.body', 'like', '%' . $search . '%') // поиск по тексту поста

            ->orWhere('users.name', 'like', '%' . $search . '%') // поиск по автору поста

            ->orderBy('posts.created\_at', 'desc')

            ->paginate(4)

            ->appends(['search' => $request->input('search')]);

        ;

        return view('posts.search', compact('posts'));

    }

}

Файл Handler.php

<?php

namespace App\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;

use Throwable;

class Handler extends ExceptionHandler

{

    /\*\*

     \* A list of the exception types that are not reported.

     \*

     \* @var array<int, class-string<Throwable>>

     \*/

    protected $dontReport = [

        //

    ];

    /\*\*

     \* A list of the inputs that are never flashed for validation exceptions.

     \*

     \* @var array<int, string>

     \*/

    protected $dontFlash = [

        'current\_password',

        'password',

        'password\_confirmation',

    ];

    /\*\*

     \* Register the exception handling callbacks for the application.

     \*

     \* @return void

     \*/

    public function register()

    {

        $this->reportable(function (Throwable $e) {

            //

        });

    }

    public function render($request, Throwable $exception) {

        if ($exception instanceof ModelNotFoundException) {

            abort(404);

        }

        return parent::render($request, $exception);

    }

}

Файл Models/Post.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

    use HasFactory;

    protected $primaryKey = 'post\_id';

}

Файл Models/User.php

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

    use HasApiTokens, HasFactory, Notifiable;

    /\*\*

     \* The attributes that are mass assignable.

     \*

     \* @var array<int, string>

     \*/

    protected $fillable = [

        'name',

        'email',

        'password',

    ];

    /\*\*

     \* The attributes that should be hidden for serialization.

     \*

     \* @var array<int, string>

     \*/

    protected $hidden = [

        'password',

        'remember\_token',

    ];

    /\*\*

     \* The attributes that should be cast.

     \*

     \* @var array<string, string>

     \*/

    protected $casts = [

        'email\_verified\_at' => 'datetime',

    ];

}

Файл site.blade.php

<!doctype html>

<html lang="ru">

<head>

    <meta charset="UTF-8">

    <meta name="viewport"

        content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0, minimum-scale=1.0">

    <meta http-equiv="X-UA-Compatible" content="ie=edge">

    <title>Все посты блога</title>

    <link rel="stylesheet" href="{{ asset('css/app.css') }}">

</head>

<body>

    <div class="container">

        <nav class="navbar navbar-expand-lg navbar-dark bg-dark">

            <a class="navbar-brand" href="{{ route('blog.index') }}">Все записи клиентов</a>

            <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"

                aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">

                <ul class="navbar-nav mr-auto">

                    <li class="nav-item">

                        <a class="nav-link" href="#">Авторы</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="{{ route('post.create') }}">Создать</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link" href="#">Контакты</a>

                    </li>

                </ul>

                <form class="form-inline my-2 my-lg-0" action="{{ route('post.search') }}">

                    <input class="form-control mr-sm-2" type="search" name="search" placeholder="Найти пост..."

                        aria-label="Поиск">

                    <button class="btn btn-outline-success my-2 my-sm-0" type="submit">Поиск</button>

                </form>

            </div>

        </nav>

        @if ($message = Session::get('success'))

        <div class="alert alert-success alert-dismissible mt-4" role="alert">

            <button type="button" class="close" data-dismiss="alert" aria-label="Закрыть">

                <span aria-hidden="true">&times;</span>

            </button>

            {{ $message }}

        </div>

    @endif

    @yield('content')

</div>

</body>

</html>

Файл form.blade.php

@csrf

<div class="form-group">

    <input type="text" class="form-control" name="title" maxlength="100"

           placeholder="Заголовок" required value="{{ $post->title ?? '' }}">

</div>

<div class="form-group">

            <textarea class="form-control" name="excerpt" maxlength="200"

                      placeholder="Анонс поста" required>{{ $post->excerpt ?? '' }}</textarea>

</div>

<div class="form-group">

            <textarea class="form-control" name="body"

                      placeholder="Текст поста" rows="7" required>{{ $post->body ?? '' }}</textarea>

</div>

<div class="form-group">

    <input type="file" class="form-control-file" name="image">

</div>

@isset($post->image)

    <div class="form-group form-check">

        <input type="checkbox" class="form-check-input" name="remove" id="remove">

        <label class="form-check-label" for="remove">

            Удалить загруженное <a href="{{ $post->image }}" target="\_blank">изображение</a>

        </label>

    </div>

@endisset

<div class="form-group">

    <button type="submit" class="btn btn-primary">Сохранить</button>

</div>

Файл create.blade.php

@extends('layouts.site')

@section('content')

<h1 class="mt-2 mb-3">Создать запись</h1>

<form method="post" action="{{ route('post.store') }}" enctype="multipart/form-data">

    @include('parts.form')

</form>

@endsection

Файл edit.blade.php

@extends('layouts.site')

@section('content')

    <h1 class="mt-2 mb-3">Редактировать запись</h1>

    <form method="post" action="{{ route('post.update', ['id' => $post->post\_id]) }}"

          enctype="multipart/form-data">

        @csrf

        @method('PATCH')

        <div class="form-group">

            <input type="text" class="form-control" name="title"

                   placeholder="Заголовок" required value="{{ $post->title }}">

        </div>

        <div class="form-group">

            <textarea class="form-control" name="excerpt"

                      placeholder="Анонс запись" required>{{ $post->excerpt }}</textarea>

        </div>

        <div class="form-group">

            <textarea class="form-control" name="body"

                      placeholder="Текст запись" rows="7" required>{{ $post->body }}</textarea>

        </div>

        <div class="form-group">

            <input type="file" class="form-control-file" name="image">

        </div>

        @isset($post->image)

            <div class="form-group form-check">

                <input type="checkbox" class="form-check-input" name="remove" id="remove">

                <label class="form-check-label" for="remove">

                    Удалить загруженное <a href="{{ $post->image }}" target="\_blank">изображение</a>

                </label>

            </div>

        @endisset

        <div class="form-group">

            <button type="submit" class="btn btn-primary">Сохранить</button>

        </div>

    </form>

@endsection

Файл index.blade.php

@extends('layouts.site')

@section('content')

<h1 class="mt-2 mb-3">Все записи</h1>

<div class="row">

    @foreach ($posts as $post)

        <div class="col-6 mb-4">

            <div class="card">

                <div class="card-header">

                    <h3>{{ $post->title }}</h3>

                </div>

                <div class="card-body">

                    <img src="{{ $post->thumb ?? asset('img/default.jpg') }}" alt="" class="img-fluid">

                    <p class="mt-3 mb-0">{{ $post->excerpt }}</p>

                </div>

                <div class="card-footer">

                    <div class="clearfix">

                        <span class="float-left">

                            Автор: {{ $post->author }}

                            <br>

                            Дата: {{ date\_format($post->created\_at, 'd.m.Y H:i') }}

                        </span>

                        <a href="{{ route('post.show', ['id' => $post->post\_id]) }}" class="btn btn-dark float-right">Читать

                            дальше</a>

                    </div>

                </div>

            </div>

        </div>

    @endforeach

</div>

<!--  {{ $posts->links() }}-->

@endsection

Файл search.blade.php

@extends('layouts.site')

@section('content')

<h1 class="mt-2 mb-3">Результаты поиска</h1>

@if (isset($posts) && count($posts))

    <div class="row">

        @foreach ($posts as $post)

            <div class="col-6 mb-4">

                <div class="card">

                    <div class="card-header">

                        <h3>{{ $post->title }}</h3>

                    </div>

                    <div class="card-body">

                        <img src="{{ $post->thumb ?? asset('img/default.jpg') }}" alt="" class="img-fluid">

                        <p class="mt-3 mb-0">{{ $post->excerpt }}</p>

                    </div>

                    <div class="card-footer">

                        <div class="clearfix">

                            <span class="float-left">

                                Автор: {{ $post->author }}

                                <br>

                                Дата: {{ date\_format($post->created\_at, 'd.m.Y H:i') }}

                            </span>

                            <a href="{{ route('post.show', ['id' => $post->post\_id]) }}" class="btn btn-dark float-right">Читать

                                дальше</a>

                        </div>

                    </div>

                </div>

            </div>

        @endforeach

    </div>

    {{ $posts->links() }}

@else

    <p>По вашему запросу ничего не найдено</p>

@endif

@endsection

Файл show.blade.php

@extends('layouts.site')

@section('content')

    <div class="row">

        <div class="col-12">

            <div class="card mt-4 mb-4">

                <div class="card-header">

                    <h1>{{ $post->title }}</h1>

                </div>

                <div class="card-body">

                    <img src="{{ $post->image ?? asset('img/default.jpg') }}" alt="" class="img-fluid">

                    <p class="mt-3 mb-0">{{ $post->body }}</p>

                </div>

                <div class="card-footer">

                    <div class="clearfix">

                        <span class="float-left">

                            Автор: {{ $post->author }}

                            <br>

                            Дата: {{ date\_format($post->created\_at, 'd.m.Y H:i') }}

                        </span>

                        <span class="float-right">

                            <a href="{{ route('post.edit', ['id' => $post->post\_id]) }}"

                               class="btn btn-dark mr-2">Редактировать</a>

                            <!-- Форма для удаления поста -->

                            <form action="{{ route('post.destroy', ['id' => $post->post\_id]) }}"

                                  method="post" onsubmit="return confirm('Удалить этот пост?')"

                                  class="d-inline">

                                @csrf

                                @method('DELETE')

                                <input type="submit" class="btn btn-danger" value="Удалить">

                            </form>

                        </span>

                    </div>

                </div>

            </div>

        </div>

    </div>

@endsection

Файл update.blade.php

@extends('layouts.site')

@section('content')

    <h1 class="mt-2 mb-3">Редактировать запись</h1>

    <form method="post" action="{{ route('post.update', ['id' => $post->post\_id]) }}"

          enctype="multipart/form-data">

        @method('PATCH')

        @include('parts.form')

    </form>

@endsection

Файл routes/web.php

<?php

use Illuminate\Support\Facades\Route;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

/\*

Route::get('/', function () {

    return view('welcome');

});

\*/

//Route::get('/', 'App\Http\Controllers\PostController@index');

Route::get('/', 'App\Http\Controllers\PostController@index')->name('blog.index');

Route::get('post/index','App\Http\Controllers\PostController@index')->name('post.index');

Route::get('post/search','App\Http\Controllers\PostController@search')->name('post.search');

Route::get('post/create','App\Http\Controllers\PostController@create')->name('post.create');

Route::post('post/store','App\Http\Controllers\PostController@store')->name('post.store');

Route::get('post/show/{id}','App\Http\Controllers\PostController@show')->name('post.show');

Route::get('post/edit/{id}','App\Http\Controllers\PostController@edit')->name('post.edit');

Route::patch('post/update/{id}', 'App\Http\Controllers\PostController@update')->name('post.update');

Route::delete('post/destroy/{id}', 'App\Http\Controllers\PostController@destroy')->name('post.destroy');

Database/migrations/create.postsr.table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration

{

    /\*\*

     \* Run the migrations.

     \*

     \* @return void

     \*/

    public function up()

    {

        Schema::create('posts', function (Blueprint $table) {

            $table->bigIncrements('post\_id');

            $table->bigInteger('author\_id')->unsigned()->nullable();

            $table->string('title', 100);

            $table->string('excerpt', 200);

            $table->string('image', 100)->nullable();

            $table->text('body');

            $table->timestamps();

            // внешний ключ, ссылается на поле id таблицы users

            $table->foreign('author\_id')

                  ->references('id')

                  ->on('users')

                  ->nullOnDelete();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*

     \* @return void

     \*/

    public function down()

    {

        Schema::dropIfExists('posts');

    }

}