МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДРАСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра информационных систем

ДОПУЩЕНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
И.О. заведующего кафедрой
Профессор, д.т.н.
_____ И.Н. Глухих
_____ 2016

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА СИСТЕМЫ ПОДДЕРЖКИ ПРИНЯТИЯ РЕШЕНИЙ ДЛЯ ПРОДВИЖЕНИЯ САЙТОВ НА ОСНОВЕ СИСТЕМ АНАЛИЗА ПОСЕЩАЕМОСТИ И СИНТАКСИЧЕСКОГО АНАЛИЗА САЙТОВ

09.04.03 Прикладная информатика
Магистерская программа «Прикладная информатика в экономике»

Выполнил работу Студент 2 курса очной формы обучения	*(Подпись)*	Ильин Ростислав Вячеславович
Научный руководитель Доцент кафедры ИС, к.т.н., доцент	*(Подпись)*	Карякин Юрий Евгеньевич
Рецензент Генеральный директор ООО «Приоритет»	*(Подпись)*	Адамов Андрей Владимирович

г. Тюмень 2016

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время сайты являются неотъемлемой частью любой компании, предприятия и бизнеса в целом. Для успешного развития сайта, ему необходимы посетители, которые будут приходить и выполнять целевые действия, такие как: покупка товара, либо заказ услуги. Для привлечения новой аудитории и удержания текущей проводится ряд работ, который называется Поисковой оптимизацией (СЕО), которая заключается в оптимизации HTML-кода, текста, структуры и внешних факторов сайта. Существуют различные системы анализа посещаемости сайтов, такие как Яндекс.Метрика, Google Analytics и другие. Все они предоставляют множество инструментов для разностороннего анализа сайта.

Перед началом работ по поисковой оптимизации проводится аудит сайта, по результатам которого выводятся рекомендации по улучшению и корректировки. Как правило, данная работа выполняется в ручном, либо полуавтоматическом режиме, что приводит к большим временным и трудовым затратам. Таким образом, возникает необходимость создания системы автоматического аудита.

Для разработки данной системы необходимо реализовать алгоритм синтаксического анализа страниц сайта, который будет сравнивать основные характеристики сайта с эталонными характеристиками.

Целью данной работы является сокращение временных, трудовых и материальных затрат на проведение подробного аудита сайта.

Для реализации цели, поставленной при написании данной работы, были выделены следующие **задачи**:

1. Провести сравнительный анализ существующих систем анализа посещаемости сайтов.
2. Проанализировать существующие системы сканирования сайта и их особенности.
3. Реализовать алгоритм синтаксического анализа сайта на основе

полученных знаний.

4. Создать веб-приложение, которое будет выполнять синтаксический анализ сайтов и отображать характеристики сайта.

5. Реализовать проверку основного текста страницы на уникальность.

6. Реализовать проверку изображений на уникальность.

7. На основе данных, полученных после синтаксического анализа, предложить пользователю рекомендации по дальнейшему улучшению поисковой оптимизации.

Данная работа состоит из введения, четырех глав, заключения, списка использованной литературы и одного приложения. В работе содержится 32 рисунка.

Первая глава содержит описание основных факторов ранжирования сайтов, также в ней представлен сравнительный анализ различных систем анализа посещаемости сайтов, настольных приложений-сканеров и выделены основные факторы, необходимые для продвижения сайта.

Во второй главе представлен анализ различных платформ, методов и инструментов, необходимых для разработки веб-приложения. В данной главе описаны такие языки программирования, как Python и Ruby, также приведены особенности двух фреймворков: Django и Ruby On Rails.

Третья глава представляет собой описание синтаксического анализа веб-страниц, разбор и нормализацию искомых данных, проверку схожих изображений и проектирование интерфейсов.

Четвертая глава содержит описание реализации веб-приложения и настройки сервера, а именно: разработку каркаса системы, разработку системы блоков и шаблонов, настройку сервера.

ГЛАВА 1. ОБЗОР ПРИМЕНЕНИЯ СИСТЕМ АНАЛИЗА ПОСЕЩАЕМОСТИ САЙТОВ И НАСТОЛЬНЫХ ПРИЛОЖЕНИЙ-СКАНЕРОВ САЙТОВ ДЛЯ ПРОДВИЖЕНИЯ В ПОИСКОВЫХ СИСТЕМАХ

В данной главе представлены основные факторы ранжирования сайтов, сравнительный анализ различных систем анализа посещаемости сайтов, настольных приложений-сканеров и выделены основные факторы, необходимые для продвижения сайта.

Сайт – (от англ. website: web – «паутина, сеть» и site – «место», буквально «место, сегмент, часть в сети») – система электронных документов (файлов данных и кода) частного лица или организации в компьютерной сети под общим адресом (доменным именем или IP-адресом).

Первый в мире сайт под доменом info.cern.ch появился 20 декабря 1990 года. Его создателем был Тим Бернерс-Ли, который опубликовал на сайте описание новой технологии World Wide Web, основанной на протоколе передачи данных http и языке гипертекстовой-разметки HTML. Данный сайт представлял собой страницу с текстовой информацией, не был сложным в архитектуре и понимании, но был прорывом того времени.

С тех пор сайты очень усложнились в своей разработке, архитектуре, использовании современных технологий и инструментов. Появление серверных языков стало следующим этапом в эволюции: на основе этих языков начали делать Framework’и, а на их основе уже системы управления сайтами (CMS – Content Management System). [7]

Данные методы и инструменты используются для разработки различного рода сайтов, как простых, так и более сложных. Для того, чтобы понять, каким образом происходит выдача сайтов в поисковых системах, и какие показатели сайта необходимо улучшить для его продвижения, необходимо рассмотреть понятие ранжирования и проанализировать факторы ранжирования.

1.1. Анализ факторов ранжирования

Для дальнейшего описания необходимо ввести основные понятия ранжирования.

1.1.1 Основные понятия ранжирования

Ранжирование (в поисковых системах) – сортировка сайтов в поисковой выдаче.

Ключевые слова (ключевики) – это слова или словосочетания, помогающие определить содержание страницы для поисковых систем.

Поисковая система (англ. search engine) – это компьютерная система, предназначенная для поиска информации. Одно из наиболее известных применений поисковых систем – веб-сервисы для поиска текстовой или графической информации. [6]

1.1.2 Факторы домена

Ниже представлен список факторов домена [1]:

1. Возраст домена. Фактор учитывается поисковыми системами, но является не столь важным по отношению к другим факторам (разница полгода, либо год для домена не существенна).
2. Использование первым словом в домене ключевого слова. Домен имеет преимущество, если его первое слово является ключевым словом.
3. Наличие ключевых слов в домене 2-го уровня. Фактор не является важным, но наличие ключевого слова в названии домена может положительно отразиться на выдаче поисковых систем.
4. Наличие в имени поддомена ключевого слова. Данный фактор также положительно влияет на ранжирование.
5. Частая смена владельца домена может привести к сбросу истории сайта с отменой всех ссылок на домен.
6. Использование домена с привязкой к стране, в которой производится продвижение, положительно влияет на результаты выдачи в поисковых системах.

1.1.3 Факторы оптимизации сайта

Далее представлены некоторые факторы оптимизации сайта:

1. Ключевое слово в заголовке страницы. Заголовок – это второй по важности контент страницы, поэтому он является сильным сигналом для поисковой системы.

2. Использование первым словом в заголовке ключевого слова. Ключевому слову в начале заголовка придается больше релевантности, чем если бы оно находилось в конце.

3. Ключевое слово в описании. Также не маловажный признак релевантности.

4. Наличие в H1 ключевого слова. H1 называют «вторым заголовком», т.к. он посылает сигнал релевантности в поисковые системы.

5. Частое употребление ключевого слова по отношению к прочему тексту на странице увеличивает релевантность данной страницы.

6. Скорость загрузки сайта. Поисковые системы используют скорость как фактор ранжирования.

7. Уникальность контента. Идентичный или видоизмененный контент может негативно отразиться на видимости сайта в поисковых системах.

8. Оригинальность контента. Если текст на странице скопирован с другого ресурса, это негативно отразится на поисковой выдаче. Также существует вероятность, что в таком случае данный сайт будет отнесен к поддержке ранее индексированного сайта с этим текстом.

9. Оптимизированные изображения. Изображения отсылают в поисковые системы такие данные, как название файлов, описание, подпись, заголовок, альтернативный текст.

10. Размер обновления страниц. Обновления и правки дают понять о свежести страниц; добавление и удаление целых секций более значимо, чем смена/удаление слов.

11. Наличие ключевых слов в H2 и H3 тегах. Данный фактор не

обязателен, но также оказывает положительное влияние на отображение сайта в поиске.

12. Расположение ключевых слов. Наличие ключевых слов в первых ста символах контента является мощным сигналом релевантности для поисковых систем.

13. Наличие полезного вспомогательного контента. Если вспомогательный контент на странице является грамотным и полезным, это также будет сигналом релевантности.

14. Мультимедиа. Наличие изображений, видео и других элементов служит сигналом качества контента.

15. Битые ссылки. Слишком много битых ссылок может означать, что сайт заброшен.

16. Сложность текста. Поисковые системы оценивают сложность текста. Ориентированность текста на специальную аудиторию, либо на аудиторию с базовыми знаниями по данной теме, может показать результат выдачи хуже, чем если бы текст был ориентирован на средний сектор пользователей.

17. Ошибки в html и W3C. Множество html ошибок и сообщения от W3C так же являются негативным показателем для сайта.

18. Длина URL. Слишком длинный URL (более 2000 символов) может негативно отразиться на выдаче.

19. Ключевое слово в URL. Наличие в URL ключевого слова, и разделение слов через символ «-» является положительным знаком для поисковых систем.

20. Приоритет страницы в карте сайта. Он известен из sitemap.xml и может влиять на ранжирование страницы.

21. Списки. Наличие нумерованных либо простых списков, позволяет разбить контент на части, что улучшает понимание этого текста для поисковых машин.

22. Качество верстки. Верстка на странице должна быть предельно

простой, для того, чтобы поисковые системы легко могли найти контент и не сделали ошибочных предположений.

1.1.4 Факторы уровня сайта

Факторы уровня сайта представлены ниже:

1. Контент освещает ценную и уникальную информацию. Поисковые системы негативно реагируют на сайты «штамповки».
2. Страница обратной связи. На сайте должна быть как минимум одна форма обратной связи.
3. «Траст» домена. Доверие к сайту, измеряющееся количеством исходящих ссылок на сайты с высоким доверием, является мощным фактором ранжирования.
4. Обновление сайта. Частота обновления сайта, даты добавления нового контента – это важный фактор в масштабах всего сайта.
5. Время бесперебойной работы сайта. Частые сбои и технические работы на сайте приводят к ухудшению позиций в поисковых системах.
6. Сертификат SSL. Наличие на сайте сертификата является мощным сигналом.
7. Навигация и хлебные крошки. Это стиль интуитивно понятной архитектуры сайта, помогающий пользователям и поисковикам понимать где они находятся.
8. Условия и конфиденциальность. Страницы с условиями предоставления услуг и политикой конфиденциальности говорят о качестве сайта.
9. Удобство интерфейса. Если на сайте трудно ориентироваться, сайт не сможет продвинуться в поисковых системах.
10. Использование на странице инструментов метрики. Если требуется продвижение сайта в поисковых системах Яндекс и Google, то желательно иметь соответствующие инструменты на сайте от этих поисковиков.

1.2 Обзор применения методов аналитики веб-сайтов

Перед улучшением позиций сайта в поисковой выдаче требуется сначала проанализировать сайт, выявить ошибки и недостатки. Для этого необходимо рассмотреть понятие веб-аналитики, области ее применения и ее методов.

1.2.1 Понятие веб-аналитики

Веб-аналитика (англ. Web analytics) – это измерение, сбор, анализ, представление и интерпретация информации о посетителях веб-сайтов с целью их улучшения и оптимизации. Основной задачей веб-аналитики является мониторинг посещаемости веб-сайтов, на основании данных которого определяется веб-аудитория и изучается поведение веб-посетителей для принятия решений по развитию и расширению функциональных возможностей веб-ресурса. [19]

1.2.2 Область применения

Веб-аналитика участвует в различных областях развития сайтов, основные из них представлены ниже [23]:

- развитие и создание нового функционала в зависимости от поведения пользователей на сайте;
- выявление проблем в навигационных схемах, структуре и контенте сайта;
- оценка рекламных кампаний в интернете.

Статистика посещаемости веб-страниц и разделов сайта позволяет проанализировать [28]:

- количество просмотренных пользователем веб-страниц;
- время, проведенное посетителем на веб-странице;
- переходы между веб-страницами;
- ключевые слова и фразы, которые пользователи используют для нахождения сайта в поисковых системах;
- географию посетителей;

- аудиторию сайта (случайные, постоянные посетители и т. д.);
- удобство интерфейса сайта и т. д.

1.2.3 Методы веб-аналитики

Существуют различные методы веб-аналитики, которые служат для измерения эффективности сайта, отслеживания конверсии и улучшения ключевых метрик. Ниже приведены несколько методов веб-аналитики [34]:

- анализ посещаемости сайта, включающий в себя статистику, абсолютные и относительные показатели, тенденции;
- анализ удобства интерфейса (отношение числа успешных действий к ошибкам; время, которое необходимо пользователю для выполнения задания и т.д.);
- анализ поведения пользователя на веб-странице;
- бенчмаркинг. Сравнение как с общими тенденциями, так и с конкурирующими компаниями с помощью различных независимых исследователей (Alexa, GemiusAudience, Google Trends).

1.3 Сравнительный обзор европейских систем анализа посещаемости

Существует большое разнообразие различных систем анализа посещаемости. Каждая из них обладает своими особенностями: достоинствами и недостатками. Далее будут представлены некоторые из них.

1.3.1 Clicky:

Простой и удобный в использовании сервис, предоставляющий инструменты для мониторинга и анализа данных в реальном времени.

Достоинства сервиса:

- понятный интерфейс;
- наличие специального виджета, который позволяет проводить аналитику прямо на сайте: с помощью него можно получить данные о всех действиях пользователя;
- тепловая карта кликов в режиме онлайн, которая отражает действия пользователей, выполнивших заданную на сайте цель;

- предоставление информации и уведомлений о конверсиях, пользователях в сети и т.д.;
- если компания присутствует в сервисе Twitter, то Clicky соберет все упоминания о данной компании;
- доступность на планшетах и телефонах (iPhone и Android), так как в нем не используется flash;
- с помощью функции Spy можно получить информацию о деятельности пользователей в режиме реального времени, просматривая список страниц, которые они посещают, и их местоположение на карте.

Недостатки:

- для крупных сайтов функционала данного сервиса недостаточно;
- отсутствует отслеживание трафика по этапам.

Стоимость: от $10 до $100, для сайтов с посещаемостью в день менее 3-х тысяч – бесплатно.

1.3.2 Piwik

Сервис предоставляет схожий набор инструментов: запросы, браузеры, ссылки, страницы, разрешение экрана, информация о пользователях и многое другое.

Достоинства:

- наличие приложения для iPhone и Android;
- ведение полной история всех данных и отчетов;
- интерфейс отчетов и сводок можно настраивать под себя;
- мультисайтовая аналитика.

Недостатки:

- реакция интерфейса бывает непредсказуема.

Существуют 2 тарифа: Бесплатный и Piwik Pro от $70 в месяц.

1.3.3 Open Web Analytics

Open Source проект, исходный код которого находится в репозитории github, то есть его можно скачать и установить на любой сервер.

Достоинства:

- возможность анализа сразу нескольких сайтов, отсутствие ограничения по количеству данных;
- предоставление максимально подробной информации о посетителях;
- работа тепловой карты в режиме записи движений курсора и отслеживание событий с DOM;
- наличие расширенных версий с дополнительным функционалом для некоторых CMS (WordPress, MediaWiki);
- в случае установки на сервер, все данные хранятся на нем.

Недостатки:

- большой порог вхождения, то есть использование данной программы доступно не каждому;
- отсутствие мобильных приложений;
- экспорт данных возможен только из базы данных.

Стоимость: бесплатно.

1.3.4 **Kissmetrics**

Популярный на западе аналитический сервис.

Достоинства:

- большой уровень детализации активности пользователей;
- возможность проводить отчеты и A/B-тесты множество раз;
- информация о посетителях сайта сохраняется на сервисе, что позволяет в дальнейшем ее использовать;
- предоставление инструментов мониторинга ссылок, к которым добавляются специальные параметры;
- мобильное приложение под iOS;
- хорошая документация и общительное сообщество.

Недостатки:

- в режиме реального времени сервис не всегда предоставляет

верные данные;

- отсутствие Android приложения.

Стоимость: от $150 в месяц.

1.3.5 MixPanel

Данный сервис очень схож с KissMetrics.

Достоинства:

- основной функционал позволяет узнать о каждом посетителе сайта более подробную информацию;
- расширенная сегментация;
- отчеты по вовлеченности и возврату пользователей;
- группировка пользователей по их поведению;
- аналитика в режиме реального времени;
- простой и понятный интерфейс;
- мобильные приложения под iOS и Android.

Недостатки:

- порог вхождения слишком большой, необходимы технические навыки/знания для работы с API.

Стоимость: бесплатно и $150 в месяц.

1.3.6 Reinvigorate

Тепловые карты и данные, получаемые в режиме реального времени, являются особенностями данного сервиса.

Достоинства:

- очень точная тепловая карта с расширенным функционалом;
- подробная статистика по каждому посетителю сайта;
- получение данных о пользователях (и другое) в режиме реального времени;
- каждому пользователю можно присвоить метку для дальнейшего мониторинга конкретного пользователя.

Недостатки:

- отсутствие мобильных приложений;
- неинформативные отчеты.

Стоимость: $10 в месяц и $100 в год.

1.3.7 Woopra

Основным достоинством являются подробные отчеты и информация о пользователях.

Достоинства:

- работа с данными происходит в режиме реального времени;
- пользователям, пришедшим на сайт, можно назначать имена и объединять в группы по разным признакам;
- центр уведомлений;
- мобильные приложения для iOS и Android.

Недостатки:

- большой объем информации может привести к не информативности отчетов;
- для базовой настройки сервиса может понадобиться помощь технического специалиста.

Стоимость: бесплатно и от $90 в месяц.

1.3.8 GoSquared

Простой и понятный для пользователей любого уровня сервис.

Достоинства:

- наличие специального функционала для интернет-магазинов;
- работа с данными в режиме реального времени;
- точность проведенного пользователем времени на сайте;
- возможность общаться с пользователями на сайте по средствам стороннего сервиса (чата);
- наличие модулей для большинства популярных CMS.

Недостатки:

- отсутствие мобильных приложений.

Стоимость: от $22 в месяц.

1.3.9 Chartbeat

Сервис ориентирован по большей части на технических специалистов.

Достоинства:

- возможность получения любой статистики в режиме реального времени;
- наличие статистики по вовлеченности пользователей;
- интеграция с социальными сетями для получения статистики от них (Twitter, Facebook);
- мобильные приложения под iOS и Android.

Недостатки:

- не подходит для работы со статическими данными.

Стоимость: $10 в месяц.

1.3.10 GoingUp

Простой и удобный сервис, предоставляющий CEO инструментарий и веб-аналитику.

Достоинства:

- инструменты для CEO бесплатны;
- тепловая карта кликов.

Недостатки:

- ограниченный функционал;
- отсутствие мобильных приложений.

Стоимость: бесплатно.

1.4 Сравнительный обзор Яндекс.Метрики и Google Analytics

Для анализа статистики сайтов наиболее распространены и популярны Яндекс.Метрика и Google Analytics. Эти системы имеют свой уникальный функционал, но в целом их возможности примерно одинаковы. Ниже

приведено сравнение данных систем анализа сайтов.

1.4.1 Яндекс.Метрика

Данный сервис по сбору статистики посещений сайтов появился в свободном доступе в 2009 году и довольно быстро стал одним из самых популярных сервисов в своем роде. Яндекс.Метрика устанавливается на сайт с помощью специального кода и позволяет отслеживать посещения сайта, группировать его посетителей по определенным категориям и признакам. Также с помощью данного сервиса можно получать данные о том, какие страницы сайта наиболее популярны среди посетителей, а какие – наименее. Кроме этого, Яндекс.Метрика способен измерять конверсию сайта и рекламы, размещенной на нем, что, в свою очередь, позволяет оперативно получать информацию об эффективности продвижения сайта и о популярности сайта среди целевой аудитории.

Таким образом, можно выделить следующие особенности Яндекс.Метрики [29]:

- универсальный интерфейс, понятный даже начинающему пользователю;
- обновление информации каждые 3-5 минут;
- возможность устанавливать и отслеживать различные цели, такие как покупка товара, просмотр определенных страниц и другие;
- получение детальных сведений по посещаемости сайта;
- возможность сортировки посетителей сайта по различным параметрам (пол, возраст и т.д.);
- разделение трафика, группировка по различным источникам;
- в случае неполадок на сайте отчеты можно получить через СМС;
- сервис является бесплатным.

1.4.2 Google Analytics

Данная система является крупным сервисом сбора статистики и анализа посещаемости сайтов. Google Analytics имеет достаточно большие

возможности – с помощью нее можно не только отследить уникальные посещения сайта, но и получить отчеты по многим другим параметрам. Для подключения Google Analytics необходимо установить на сайте счетчик с JavaScript кодом, а обработка информации и составление отчетов происходит автоматически на сайте системы. Отчеты могут содержать абсолютно различные данные, такие как: ключевые слова, среднее время пребывания посетителей на сайте, на странице, анализ посетителей по разным критериям.

Далее приведены некоторые особенности Google Analytics [32]:

- довольно сложный для восприятия интерфейс, так как данная система объемна и содержит большое количество различных режимов, настроек и фильтров;
- анализ конкурирующих сайтов;
- на отечественном трафике вероятны большие погрешности до 40%;
- система является ограниченно бесплатной.

Следует отметить, что обе системы анализа статистики посещаемости обладают очень большими возможностями и высоким потенциалом, однако, для реализации поставленной цели лучше подойдет инструмент Яндекс.Метрика, так как он более понятен пользователю, более оперативно предоставляет отчеты и лучше адаптирован под особенности Рунета.

1.5 Обзор существующих настольных приложений-сканеров сайтов

Также существует большое количество настольных приложений, нацеленных на сканирование сайтов, выявление проблем по основным характеристикам. Далее будет представлен сравнительный анализ самых популярных из них.

Xenu Link Sleuth – изначально предназначалась для поиска битых ссылок на сайте, однако, с помощью неё можно решать ряд других задач, проводя аудит внутренней структуры сайта и находя в нем слабые места.

Возможности программы:

- поиск битых (неработающих) ссылок на заданном ресурсе;
- составление карты сайта;
- поиск страницы с большим временем отдачи;
- нахождение неуникальных заголовков;
- нахождение страниц с большим уровнем вложенности;
- поиск страницы с максимальным количеством исходящих ссылок;
- просмотр страниц, которые имеют наибольшее и наименьшее количество внутренних ссылок на себя;
- нахождение картинок с отсутствующим атрибутом alt.

Netpeak Spider – функционал данного приложения приближен к предыдущему приложению, но имеет свои достоинства.

Возможности программы:

- выгрузка основных SEO-параметров;
- поиск ошибок, битых ссылок, неверных редиректов, дубликатов заголовков и meta-description;
- анализ входящих и исходящих ссылок для каждой страницы;
- расчет внутреннего PageRank (веса страниц);
- гибкие настройки сканирования (в том числе учет запретов в robots.txt), и экспорт результатов в электронные таблицы Excel.

Screaming Frog SEO Spider – программа, позволяющая быстро получить основные данные для анализа сайта. Отличный инструмент для поиска ошибок, дублированного контента, битых ссылок. Позволяет легко и быстро провести экспресс-аудит сайта и сохранить результаты в файл.

Функционал данной программы значительно превосходит свои аналоги:

- Content – тип контента (Text/Html, image/jpeg, text/css и так далее), а также кодировка на данной странице (например, utf8);
- Address – адреса страниц сайта;

- Status Code – код ответа сервера (301, 404 и т.д);
- Status – статус ответа сервера (текстовое обозначение кода ответа);
- Title 1 – заголовок страницы;
- Title 1 Length – длина заголовка страницы;
- Meta Description 1 – мета-описание страницы (тэг <meta type=description>);
- Meta Description Length – длина описания страницы;
- Meta Keyword 1 – список ключевых слов;
- Meta Keywords Length – суммарная длина ключевых слов (в символах);
- h1 – содержимое первого тега <h1> на странице;
- h1 Length – длинна содержимого тега <h1> в символах;
- h2 – содержимое самого первого тега <h2> на странице;
- h2 Length – длинна содержимого тега <h2> в символах;
- Meta Data – данные из файла robots.txt;
- Level – уровень вложенности страницы или число кликов, которое надо сделать, начиная от главной страницы сайта, чтобы попасть на данную страницу;
- Size – размер страницы в байтах;
- Inlinks – количество ссылок, находящихся на странице;
- Outlinks – количество ссылок, ведущих за пределы сайта;
- Hash – помогает определить уникальность страниц, если хэши двух страниц одинаковые, значит содержимое страниц дублируется.

Все из перечисленных программ имеют не только схожий функционал, но и схожие минусы:

- ни одна их них не охватывает весь необходимый спектр задач;
- данные программы не являются кросплатформенными;
- отсутствует возможность отслеживания статистики по

нескольким сайтам одновременно;

- каждый анализ является независимым, из чего следует отсутствие статистики по основным характеристикам (полнота сайта и тенденции его заполнения);
- данные программы не используют информацию из систем анализа посещаемости сайтов (Яндекс.Метрика, Google Analytics).

1.6 Посещаемость сайта

Продвижение сайта и его посещаемость – это два неразрывно связанных между собой компонента. Одно нельзя представить без другого. Посещаемость сайта – это количество посетителей, пришедших на сайт в течение определенного промежутка времени, обычно – в течение суток. [24]

1.6.1 Активные посетители

Хит – это однократная загрузка страницы сайта уникальным посетителем. Активным называется посетитель, выполнивший как минимум пару хитов в течение 10 минут с момента первой загрузки страницы сайта.

1.6.2 Нецелевые посетители

Нецелевые посетители – это люди, зашедшие на сайт с поисковой системы по случайным поисковым запросам, которые не входят в стандартное семантическое ядро сайта. Нецелевой посетитель очень редко становится постоянным посетителем ресурса, и практически никогда не выполняет «целей» сайта.

1.6.3 Целевые посетители

Целевая аудитория – это группа пользователей, для которых предназначен контент сайта. Целевые посетители знают какую информацию они хотят найти или какую услугу приобрести.

Для продвижения сайта очень важно выделить его целевую аудиторию, так как это позволит более точно направить рекламные кампании.

Выделяют три основные группы целевой аудитории сайта:

- посетители, которые заинтересованы в получении какой-либо

информации;

- посетители, которые заинтересованы в выборе товара или услуги;
- посетители, которые заинтересованы в приобретении товара или услуги.

1.6.4 Определение целевой аудитории

Для определения целевой аудитории сайта, необходимо составить примерный портрет целевого посетителя сайта (портрет клиента). [16]

Основные характеристики целевой аудитории: пол, возраст, место жительства, семейное положение, образование, занятость, финансовый статус, социальный статус, а также некоторые другие данные, важные для специфики конкретного сайта.

Различают несколько способов сбора информации для составления портрета целевого посетителя, такие как:

- использование счетчика посещений сайта и анализ лог файла сервера (определение регионов пользователей, время их активности на сайте и др.);
- тестирование и опросы пользователей на сайте (регистрация, опросники на сайте для идентификации уникальных посетителей);
- аудит сайта, исследования, рекламные кампании.

Поисковые запросы являются важным источником информации о целевой аудитории. Произвести оценку величины целевой аудитории можно узнав число поисковых запросов. Данная функция реализована в таких поисковых системах, как Яндекс (Яндекс статистика поиска), Рамблер (Adstat Rambler) и Google (Google Статистика поиска). На основе этих данных можно получить сводные цифры и проанализировать закономерности поведения групп пользователей, а также оценить эффективность рекламной кампании.

При разработке дизайна и структуры сайта всегда учитываются характеристики целевого портрета клиента. Также эти данные используют

при корректировке информационной наполненности сайта, для того, чтобы повысить конверсию сайта.

1.7 Выявление основных факторов сайта, необходимых для его продвижения

В связи с непрерывным ростом сайты давно вышли за грань нескольких страниц. На сегодняшний день до сих пор существуют «простые» сайты, но также существуют информационные порталы, насчитывающие десятки тысяч страниц, анализ их вручную достаточно сложная и трудоемкая задача. Таким образом, существует необходимость создания автоматизированной системы аудита сайтов. Для выявления основных показателей, необходимых для продвижения сайта был проведен анализ множества сайтов, систем анализа (прил.1), и также была проведена консультация с экспертами в сфере продвижения сайтов. Далее представлены основные факторы и показатели с некоторыми комментариями:

- уникальность контента страниц (желательно 500-700 символов);
- заголовок страниц (title) должен быть 80 – 90 символов (по возможности сопоставлен с h1 тегом и ключевыми словами);
- описание страницы (description) должно быть не более 200 символов (по возможности сопоставлено с основным содержимым страницы);
- вложенность страниц: количество кликов до попадания на страницу с главной должно сводиться к минимуму (не более 3х кликов);
- наличие у картинок alt;
- у каждой ссылки в контенте должен быть title;
- должны отсутствовать ссылки, ведущие на страницу с ответом от сервера 404;
- ответ сервера 404 только в случае, если страница не существует;
- проверка сайта на единую точку входа: либо все ссылки сайта начинаются с «www», либо без «www»;

- проверка окончаний ссылок – должен быть выбран один из двух способов: либо с использованием «/» в конце ссылок, либо без;
- определение количества ссылок, которые ведут за пределы сайта;
- проверка title, description, keywords на дублирование;
- проверка страниц сайта на дубликаты (не должно существовать двух страниц с разными url, но с абсолютно одинаковым контентом);
- реализация отображения наличия или отсутствия sitemap.xml;
- реализация отображения наличия или отсутствия robots.txt (правила для индексации);
- история домена – количество раз, когда менялся владелец домена (чем число больше, тем больше вероятность, что Google выполнит сброс истории и отменит все ссылки на домен);
- скорость загрузки не должна превышать трех секунд;
- наличие на странице одного тэга h1, не более двух тэгов h2, трех – h3;
- количество внутренних ссылок, ведущих на страницу сайта (чем больше число, тем важней считается страница для поисковых машин);
- длина URL не должна превышать 2 тысяч символов;
- разделение слов в ЧПУ должно быть реализовано через символ «-».

На основе приведенных выше факторов в дальнейшем будет разработано веб-приложение и его функционал.

ГЛАВА 2. ВЫБОР ПЛАТФОРМЫ, МЕТОДОВ И ИНСТРУМЕНТОВ ДЛЯ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ

В данной главе представлен анализ различных платформ, методов и инструментов, необходимых для разработки веб-приложения. В ней также описаны такие языки программирования, как Python и Ruby, приведены особенности двух фреймворков: Django и Ruby On Rails.

2.1. Основная терминология

Для дальнейшего анализа необходимо ввести основную терминологию. [31]

Язык программирования – формальная знаковая система, предназначенная для записи компьютерных программ. Язык программирования определяет набор синтаксических, лексических, и семантических правил, которые в свою очередь определяют внешний вид программы и действия, которые выполнит исполнитель (обычно – ЭВМ) под её управлением.

Веб-приложение – клиент-серверное приложение, в котором клиентом выступает браузер, а сервером – веб-сервер. Логика веб-приложения распределена между сервером и клиентом, хранение данных осуществляется, преимущественно, на сервере, обмен информацией происходит по сети. Одним из преимуществ данного подхода является тот факт, что клиенты не зависят от конкретной операционной системы пользователя, поэтому веб-приложения являются кросплатформенными сервисами.

Веб-приложения разрабатываются на выбранном языке программирования, для более быстрой и качественной разработки используются фреймворки.

Фреймворк (англ. framework – каркас, структура) – программная платформа, определяющая структуру системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого

проекта. Также часто употребляется слово «каркас», в некоторых книгах оно используется как основное, в том числе без упоминания об англоязычном аналоге. Помимо этого, существует каркасный подход к построению программ, состоящий из двух частей: постоянная часть – каркас, не меняющийся от конфигурации к конфигурации и несущий в себе гнезда, в которых размещается вторая, переменная часть – сменные модули, так называемые точки расширения.

2.2. Выбор языка программирования

Для создания любого веб-приложения необходимо выбрать язык программирования. Их существует огромное множество, и у каждого есть свои достоинства и недостатки. Далее будут представлены особенности двух языков программирования: Python и Ruby.

2.2.1 Обзор возможностей языка Python

Ниже представлены особенности языка Python [6, 30]:

- очень понятная документация с доступными примерами;
- функция является объектом, то есть существует возможность создать переменную, поместить в нее ссылку на функцию и передать другой функции;
- поддержка именованных аргументов функций;
- декораторы – к объектам возможно добавлять аннотации, которые будут влиять на их поведение;
- исключения, произошедшее в главном потоке, обрабатываются обработчиком исключений по умолчанию;
- в стандартной библиотеке предусмотрена распаковка и запаковка архивов;
- для всех платформ реализована поддержка GUI библиотеки Qt.

2.2.2 Обзор возможностей языка Ruby

Далее представлены особенности языка Ruby [35]:

- наличие интерполяции, то есть автоматической подстановки

результата кода в строку;

- встроенная поддержка openssl с доступной документацией;
- поддержка создания внутренних DSL: instance_eval, блоки, вызов функции без скобок;
- механизм вызова shell довольно удобен в использовании. Для получения результата команд необходимо два символа «`». Также данный язык корректно работает с путями, содержащими пробелы;
- существует возможность включения модулей по относительному пути;
- индексирование за пределами коллекции возвращает nil, что позволяет в ряде случаев писать лаконичный код;
- удобный синтаксис подавления ожидаемых ошибок;
- в стандартной библиотеке поставляется шаблонизатор «erb» с очень удобным синтаксисом;
- создание временных папок с возможностью автоматического удаления;
- удобная работа со временем и датой;
- штатная остановка потока снаружи возможна с помощью метода exit.

2.3. Анализ фреймворка Django

Ниже представлены особенности фреймворка Django.

2.3.1 Описание фреймворка

Django (Джанго) – свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MTV. Проект поддерживается организацией Django Software Foundation.

Создание сайта на Django основано на разработке одного или нескольких приложений, которые являются независимыми и самодостаточными. Это одно из основных архитектурных различий данного фреймворка. Как и в большинстве фреймворков, Django пропагандирует

DRY (англ. Don't repeat yourself).

Также важным отличием от других фреймворков является конфигурация URL с помощью регулярных выражений, а не вывод автоматически из структуры моделей контроллеров.

Рекомендованная база данных PostgreSQL. Для работы с БД Django использует собственный ORM, где модели данных описываются классами на языке Python, и по ним генерируется схема базы данных. [17]

2.3.2 Архитектура

«Модель-Представление-Контроллер» (MVC) очень похожа на архитектуру Django.

Шаблон (англ. Template) – презентационная логика, в классической модели MVC это View.

Представление (англ. View) – из модели MVC примерно соответствует уровню, который называется Контроллер. Из-за этого уровня архитектуру Django очень часто называют «Модель-Шаблон-Представление» (MTV).

Изначально Django разрабатывался, как средство для работы новостных ресурсов – это достаточно сильно повлияло на его архитектуру. Он предоставляет набор инструментов, которые помогают в быстрой разработке веб-сайтов информационного характера. Одним из таких инструментов является встроенное приложение для управления контентом сайта – автоматическая генерация административной части сайта. Административное приложение позволяет выполнять набор стандартных операций над объектами (CRUD – create, read, update, delete) и регистрировать все совершенные действия в системе. Данное приложение предоставляет интерфейс для управления пользователями и группами.

2.3.3 Конфигурация сервера

Фреймворк Django разрабатывался для работы на сервере Apache с модулем mod python. Поддержка WSGI позволила работать под управлением mod_wsgi, FastCGI или SCGI на Apache, nginx и других серверах.

Изначально Django поставлялся с базой данных PostgreSQL, однако в

настоящее время данный фреймворк может работать с большим спектром СУБД: MsSQL, MySQL, DB2, SQL Anywhere, SQLite, Firebird и Oracle.

Django содержит собственный веб-сервер, который автоматически определяет изменения в файлах проекта и перезапускается, что ускоряет процесс разработки, но при этом фреймворк работает в однопоточном режиме, что делает его непригодным для загруженных серверов, поэтому его рекомендуется использовать только в процессе разработки.

2.3.4 Возможности

Учитывая факты, описанные выше, можно выделить следующие возможности фреймворка Django:

- автоматически генерируемая административная часть с возможностью перевода на множество языков;
- установка конфигурации URL на основе регулярных выражений;
- собственная ORM с поддержкой транзакций;
- адаптация приложения к языковым и культурным особенностям;
- кеширование;
- авторизация и аутентификация, также возможность подключения сторонних модулей для аутентификации, например, OpenID, LDAP и других;
- встроенная документация по тегам и моделям, доступная через административную часть;
- middleware – система фильтров для построения дополнительных обработчиков запросов;
- generic views – шаблоны функций контроллеров;
- модуль для работы с html формами (построение форм на основе модели базы данных);
- высокая степень абстракции большинства компонент между собой, что позволяет заменять их аналогичными компонентами, например, использовать сторонние шаблонизаторы. Однако замена фундаментальных компонент (таких, как ORM) является довольно сложной задачей.

Помимо встроенных возможностей, поставляемых вместе с фреймворком, существуют пакеты, расширяющие его возможности. Список всех пакетов, их функционал, изменения и разработку удобно отслеживать на специальном ресурсе – www.djangopackages.com

2.4 Анализ фремворка Ruby on Rails

Ниже представлены особенности фреймворка Ruby on Rails.

2.4.1 Описание

Ruby on Rails (ROR) – фреймворк, написанный на языке программирования Ruby, предоставляет архитектурный образец «Модель-Представление-Контроллер» (MVC) для веб-приложений, а также обеспечивает их интеграцию с сервером и базой данных, является открытым ПО и распространяется под лицензией MIT. [37]

2.4.2 Принципы

ROR базируется на следующих принципах разработки приложений:

- использование соглашения по конфигурации, типичных для большинства приложений (принцип Convention over configuration – соглашение выше конфигурации);
- предоставление инструментов для повторного использования, что позволяет минимизировать дублирование кода в приложениях (принцип Don’t repeat yourself – DRY).

2.4.3 Архитектура

ROR использует REST архитектуру для построения веб-приложений. Основными компонентами ROR приложения являются модель, представление и контроллер (model, view and controller).

Модели предоставляют всем компонентам приложения объектно-ориентированное представление данных из БД. Объекты модели способны сохранять и загружать данные в БД, а также реализовывать бизнес-логику.

Для хранения объектов модели в реляционной СУБД в ROR используется библиотека ActiveRecord, ее аналогом (для версии 3) является

DataMapper, также существуют модули для работы с нереляционными БД (Mongoid для MongoDB и т.д).

Представление предоставляет интерфейс пользователя с данными, которые были переданы из контроллера. Далее он передает запросы пользователя для последующих действий с данными в контроллер (не имеет возможность менять модель данных).

Для описания представления используются шаблонизаторы. Основной, который поставляется вместе с ROR называется ERB. Шаблонизаторы в большей части предоставляют синтаксис схожий с HTML, но имеющий дополнительный функционал (Embedded Ruby или ERb).

Контроллер в ROR определяет логику поведения веб-приложения. Контроллер имеет возможность вызывать методы модели и запускать формирование представления.

Маршрутизация и сопоставление URL с действием контроллера задается в файле config/routes.rb.

ГЛАВА 3. СИНТАКСИЧЕСКИЙ АНАЛИЗ И ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСОВ

Данная глава представляет собой описание синтаксического анализа веб-страниц, разбор и нормализацию искомых данных, проверку схожих изображений и проектирование интерфейсов.

Синтаксический анализ сайтов (жарг. парсинг) – последовательный анализ информации, размещённой на веб-страницах, которые представляют собой иерархичный набор данных, структурированный с помощью человеческих и компьютерных языков. На человеческом языке предоставлена информация, знания, ради которых люди и пользуются Интернетом. Компьютерные языки (html, JavaScript, css) определяют, как информация выглядит на мониторе. [8]

3.1. Синтаксический анализ веб-страниц

Для решения задачи нахождения основного контента страницы выделяются следующие подходы:

- извлечение данных с использованием html документа (DOM и текстовый уровень) – синтаксический анализ;
- извлечение данных с использованием отрендеренного с помощью computer vision документа. Данный алгоритм является очень точным, но в тоже время и самым сложным в реализации и самым ресурсоемким;
- извлечение данных с использованием сайта целиком (сравнение однотипных страниц и нахождение различий между ними). Этим занимаются большие поисковики.

Из перечисленных выше способов к извлечению контента страницы подходят способы, использующие только один html документ. Также можно решить проблему определения страниц со списками статей с постраничной навигацией.

Алгоритм синтаксического анализа:

ШАГ 1. Привести верстку сайта к «плоскому» виду: блок, в котором находится текст блока.

ШАГ 2. Найти блоки с повторяющимися строками в атрибутах, пессимизировать их.

ШАГ 3. Найти блоки, в которых размер соотношения верстки к тексту превышает 35%, пессимизировать их.

ШАГ 4. Найти 2 блока, в которых содержится максимальное количество точек, все остальные блоки пессимизировать.

ШАГ 5. Найти блоки с текстом примерно одинаковой длинны, пессимизировать их.

ШАГ 6. Найти 1 блок с самым большим количеством текста, все остальные блоки пессимизировать.

Более подробно алгоритм описан ниже.

3.1.1. Приведение верстки сайта к «плоскому» виду

Необходимо взять все элементы разметки структуры страницы (для простоты – div) и текст, который в них содержится (если он есть). Необходимо получить плоский список DIV элемент –> текст в нем.

Для примера на рисунке 1 представлено меню случайного сайта. Результатом данного шага будет текст, содержащийся в каждом элементе тэга *a:*

«Главная Новости О компании Каталог Мероприятия Контакты»

```
<div class="menu">
    <!-- ... -->
    <a href="/frontpage/">Главная</a>
    <a href="/news/">Новости</a>
    <a href="/about-company/">О компании</a>
    <a href="/catalog/">Каталог</a>
    <a href="/events/">Мероприятия</a>
    <a href="/contacts/">Контакты</a>
    <!-- ... -->
</div>
```

Рисунок 1. Меню случайного сайта

При наличии вложенных div элементов, их содержание отбрасывается. Дочерние div будут обработаны позже.

В качестве примера возьмем блок div с классом *copy* на рисунке 2. Результатом упрощения будет два элемента: в одном текст «site.com», а во втором – «Техподдержка Напишите нам Контакты»

```
<div class="copy">
    (c) site.com
    <div class="copy-right">
        <a href="#">Техподдержка</a>
        <a href="#">Напишите нам</a>
        <a href="#">Контакты</a>
    </div>
</div>
```

Рисунок 2. Блок div с классом copy

Предполагается, что элементы, которые семантически предназначены для разметки структуры (div), не используются для разметки параграфов в тексте.

На рисунке 3 представлено DOM дерево до упрощения структуры.

```
// ШАПКА
<div class="menu">
    <!-- ... -->
    <a href="/frontpage/">Главная</a>
    <a href="/news/">Новости</a>
    <a href="/about-company/">О компании</a>
    <a href="/catalog/">Каталог</a>
    <a href="/events/">Мероприятия</a>
    <a href="/contacts/">Контакты</a>
    <!-- ... -->
</div>

// КОНТЕНТ
<div class="content">
    <!-- ... -->
    Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean
    commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus
    et magnis dis parturient montes, nascetur ridiculus mus. Donec quam
    felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla
    consequat massa quis enim. Donec pede justo, fringilla vel, aliquet
    nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a,
    venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium.
    <!-- ... -->
</div>

// ПРАВЫЙ БЛОК
<div class="news">
    <div class="title">Новости</div>
    <div class="list">
        <!-- ... -->
        <div class="news__item">
            <a href="#" class="news__item__name">
                Названиe нововсти
            </a>
            <div class="news__item__desc">
                Описание новости
            </div>
        </div>
        <div class="news__item">
            <a href="#" class="news__item__name">
                Названиe нововсти
            </a>
            <div class="news__item__desc">
                Описание новости
            </div>
        </div>
        <!-- ... -->
    </div>
</div>

// ПОДВАЛ
<div class="comments">
    <!-- ... -->
    <div class="comment">
        <div class="comment__user_name">
            Иванов Иван Иванович
        </div>
        <div class="comment__text">
            Текст комментарий
        </div>
    </div>
    <div class="comment">
        <div class="comment__user_name">
            Иванов Иван Иванович
        </div>
        <div class="comment__text">
            Текст комментарий
        </div>
    </div>
    <!-- ... -->
</div>

// ПОДВАЛ
<div class="copy">
    <!-- ... -->
    (c) site.com
    <div class="copy_right">
        <a href="#">Техподдержка</a>
        <a href="#">Напишите нам</a>
        <a href="#">Контакты</a>
    </div>
    <!-- ... -->
</div>
```

Рисунок 3. Верстка страницы до упрощения

В итоге из дерева получается «плоский» набор блоков (рисунок 4), в

котором необходимо классифицировать контент.

```
// ШАПКА
<div class="menu">
Главная Новости О компании Каталог Мероприятия Контакты

// КОНТЕНТ
<div class="content">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo
ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis
dis parturient montes, nascetur ridiculus mus. Donec quam felis,
ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa
quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget,
arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo.
Nullam dictum felis eu pede mollis pretium.

// ПРАВЫЙ БЛОК
<div class="title">
Новости
<div class="news__item">
Название нововсти
<div class="news__item__desc">
Описание новости

// ПОДВАЛ
<div class="comment__user_name">
Иванов Иван Иванович
<div class="comment__text">
Текст комментарий

// ПОДВАЛ
<div class="copy">
(c) site.com
<div class="copy_right">
Техподдержка Напишите нам Контакты
```

Рисунок 4. Верстка страницы после преобразования

3.1.2. Поиск повторяющихся строк в атрибутах

Во всех элементах верстки страницы находим элементы с повторяющимися названиями атрибутов (class, id и т.д.)

На рисунке 5 отображены примеры повторяющихся элементов: блоки новостей и комментариев.

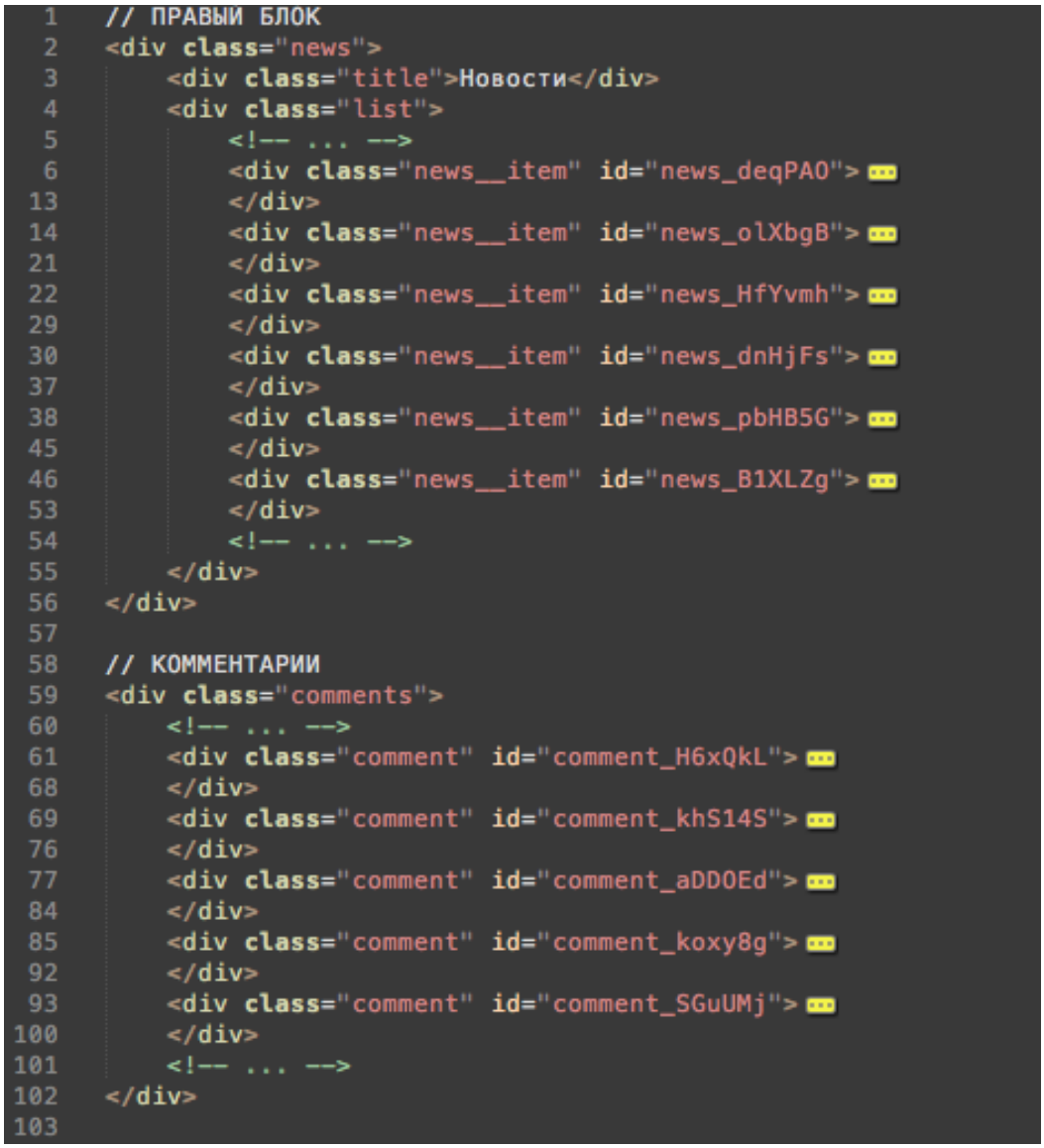


Рисунок 5. Повторяющиеся блоки новостей и комментариев

Все эти элементы и их потомки необходимо пессимизировать, то есть поставить некоторый понижающий коэффициент, который зависит от количества найденных повторений.

Таким образом, все вложенные элементы, являющиеся потомками (включая те, которые попали в набор для классификации), получат отрицательный коэффициент.

3.1.3. Поиск блоков, в которых соотношение верстки к тексту превышает 35%

Блок является контентом, если количество разметки в нем достаточно мало. В противном случае, блок контентом не является, пример представлен

на рисунке 6.

```
<div class="htmlblock">
    <ul>
        <!-- ... -->
        <li>
            <span class="num">
                1
            </span>
            Текст элемента
        </li>
        <li>
            <span class="num">
                1
            </span>
            Текст элемента
        </li>
        <li>
            <span class="num">
                1
            </span>
            Текст элемента
        </li>
        <!-- ... -->
    </ul>
</div>
```

Рисунок 6. Пример блока, в котором верстка текста значительно больше чем сам текст

3.1.4. Поиск двух блоков с максимальным количеством точек

Из теории численной лингвистики следует следующее правило: в основной части документа количество точек будет максимально, т.е. в заголовках и меню точки практически не ставятся, в то время как в теле документа их много.

На рисунках 7 и 8 представлены примеры навигационных меню случайных сайтов. Следует отметить, что на них отсутствуют точки, это доказывает, что блок меню не является контентом страницы.

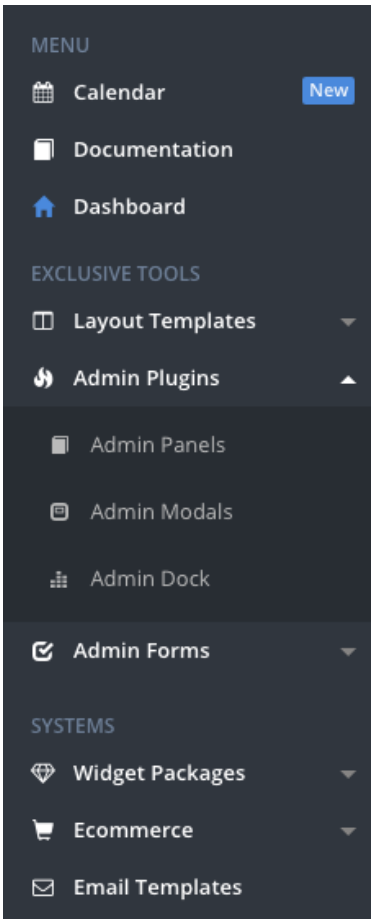


Рисунок 7. Вертикальный блок меню

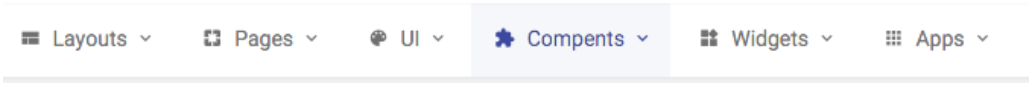


Рисунок 8. Горизонтальный блок меню

Если какие-то меню и списки новых материалов на сайте прошли через предыдущие шаги, то подсчет точек позволит сократить количество претендентов на роль блока с контентом.

3.1.5. Поиск блоков с текстом примерно одинаковой длины

Множество блоков с текстом примерно одной длины, особенно если текст короткий, вероятнее всего является списком новостей, либо каким-то иным списком. Такие блоки необходимо пессимизировать.

Краткий список новостей представлен на рисунке 9. Данный блок необходимо пессимизировать.

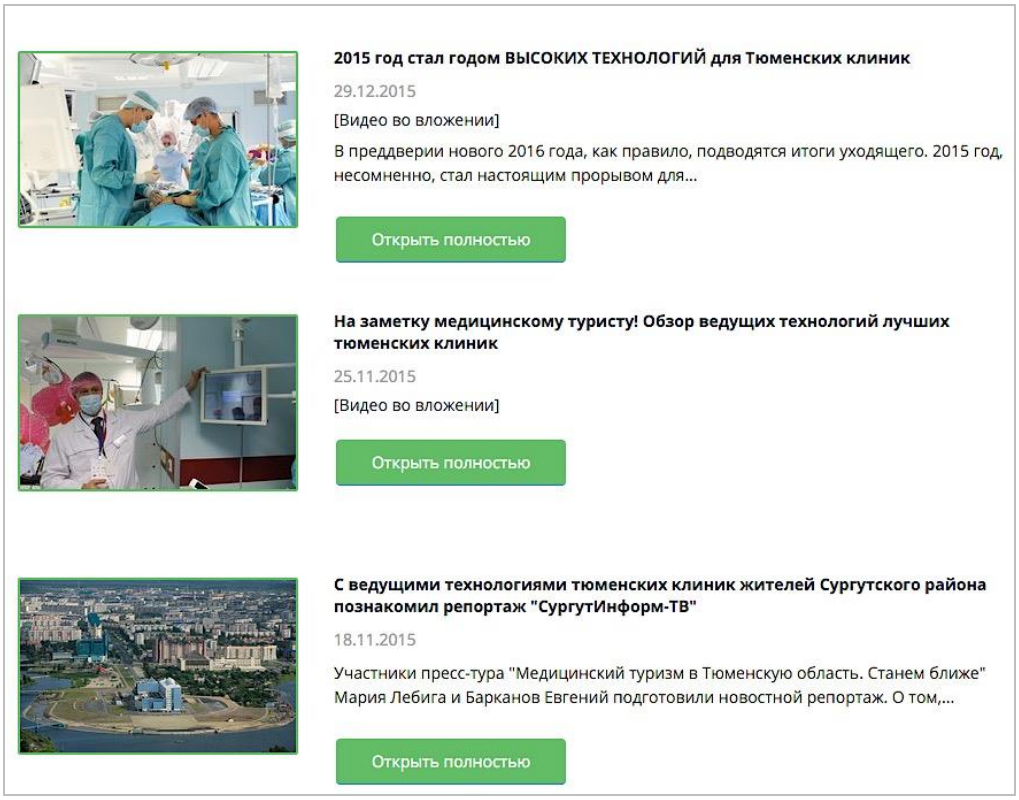


Рисунок 9. Краткий список новостей

3.1.6. Длина текста в элементе

В данном случае можно выделить прямую зависимость – чем длиннее текст в блоке, тем больше вероятность того, что он является контентом.

Вклад данного параметра в конечную оценку элемента довольно существенен. Описанным методом можно решить до 90% случаев синтаксического анализа сайта. Прочие методы способны решить до 95% анализа, но при этом пропустят огромную долю процессорного времени.

Современные подходы и алгоритмы анализа позволяют по одному лишь тексту страницы добиться качества распознания контента до 95%.

Повышение качества анализа достигается учетом оформления текста (CSS) и представления страницы или сравнением нескольких однотипных страниц с сайта, но для решения поставленной задачи достаточно описанного выше алгоритма.

3.2. Разбор и нормализация искомых данных

Используя встроенный в язык Ruby функционал разбора html сущностей, с любой страницы можно получить текст, но для проверки на

уникальность этого недостаточно, его надо нормализовать, придав ему унифицированную форму. [14]

Ниже представлен алгоритм нормализации текста:

ШАГ 1. Нахождение всех кириллических и греческих букв, по написанию похожих на соответствующие английские – это один из основных способов обхода проверки на плагиат.

ШАГ 2. Замена всех простонародных сокращенных форм полными формами.

ШАГ 3. Замена всех высокохудожественных юникодовских символов на простые: кавычки «елочкой», кавычки в виде перевернутых запятых, длинные и полудлинные тире, апострофы, многоточие, а также лигатуры *ff*, *ffi*, *st* и т.д.

ШАГ 4. Замена двух апострофов, стоящих подряд, на кавычки, а двух тире – на одно.

ШАГ 5. Замена всех последовательностей пробельных символов одним пробелом.

ШАГ 6. Удаление из текста всего, что не укладывается в диапазон символов ASCII.

ШАГ 7. Удаление всех управляющих символов, кроме обычного перевода строки.

После проведенных выше операций будет получен нормализованный текст. Далее необходимо разбить его на предложения. Данная задача осложняется следующими факторами:

- предложения могут заканчиваться точкой, многоточием, восклицательным и вопросительным знаком или вообще никак не заканчиваться (например, в конце абзаца);
- точки могут стоять после различных сокращений, которые не являются концом предложения;
- интернет-ссылки: в начале ссылки не всегда указан протокол, что необходимо для ее идентификации (например, статья ссылается на сайт

Amazon.com, в котором содержится точка, но указан протокол);

- необходимо учитывать все домены – несколько основных и довольно большое количество доменов других стран.

Из сказанного выше следует, что результат становится вероятностным. Каждая конкретная точка может быть, а может и не быть концом предложения.

Разбиение текста реализовано с помощью регулярных выражений: находятся все «неправильные» точки, заменяются на другие символы, текст разбивается на предложения по оставшимся, затем символы точек возвращаются назад.

3.2.1. Поиск в интернете

Инструментом для поиска частей текста в интернете был выбран Google. Обычным поиском пользоваться не представляется возможным, так как существует ряд ограничений, в том числе на количество запросов с одного домена.

Google хранит некоторую информацию о расстоянии между словами. Известно, что оптимальные результаты показывает серия из 8 слов.

Ниже представлен конечный алгоритм разбиения и поиска:

ШАГ 1. Разбитие текста на слова.

ШАГ 2. Удаление стоп-слов (служебных, попадающихся чаще всего).

ШАГ 3. Формирование запросов из восьми слов с перекрытием (то есть, первый запрос – слова 1-8, второй 2-9 и так далее. Возможно использование перекрытия в два слова: таким образом можно сэкономить запросы, но данный вариант немного ухудшит качество).

ШАГ 4. Удаление каждого третьего запроса, если текст большой (>50kb), и удаление каждого второго запроса, если текст очень большой (>300 kb). Данное действие влияет на точность результата анализа, но не значительно, так как тексты обычно копируют целыми абзацами, а не отдельными фразами.

ШАГ 5. Отправка запросов на Google (можно одновременно).

ШАГ 6. Получение ответов, их разделение, создание общего списка и удаление из него дубликатов. Далее необходимо отсортировать список полученных адресов по количеству пришедших дубликатов и удалить последние, считая их не показательными. Количество результатов поиска будет чрезмерно огромным, и в этот момент вступает в силу распределение Ципфа.

3.2.2. Закон Ципфа

Закон Ципфа – эмпирическая закономерность распределения частоты слов естественного языка: если все слова языка (или просто достаточно длинного текста) упорядочить по убыванию частоты их использования, то частота n-го слова в таком списке окажется приблизительно обратно пропорциональной его порядковому номеру n (так называемому рангу этого слова). Например, второе по используемости слово встречается примерно в два раза реже, чем первое, третье – в три раза реже, чем первое, и так далее.

График закона Ципфа представлен на рисунке 10.

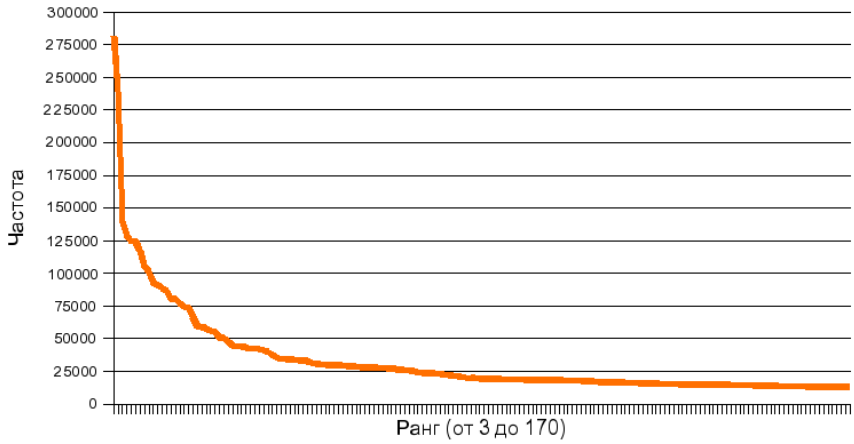


Рисунок 10. Закон Ципфа на графике

Из этого следует, что полностью обработать данные не представляется возможным, и не зависимо от места остановки проверки, качество анализа будет ухудшаться. Поэтому решение, в какой момент необходимо остановиться, было принято, в зависимости от времени, 1 запрос не должен превышать 20 секунд.

3.3. Проверка схожих изображений

Для анализа сайта на уникальность требуется проверка не только его текста, но и изображений, размещенных на нем. Для реализации данной задачи необходимо использовать перцептивные хэш-алгоритмы.

Перцептивные хэш-алгоритмы описывают класс функций, которые позволяют генерировать сравнимые хэши. Характеристики изображения используются для генерации индивидуального (но не уникального) «отпечатка», и эти «отпечатки» в последствии можно сравнивать друг с другом.

Концепция перцептивных хэшей отличается от криптографических хэш-функций, таких как MD5 и SHA1. В криптографии каждый хэш является случайным. Данные, которые используются для генерации хэша, выполняют роль источника случайных чисел, так что одинаковые данные дадут одинаковый результат, а разные данные – разный результат. Из сравнения двух хэшей SHA1 на самом деле можно сделать только два вывода. Если хэши отличаются, значит, данные разные. Если хэши совпадают, то и данные, скорее всего, одинаковые (стоит отметить, что из-за существования вероятности коллизий, одинаковые хэши не гарантируют совпадения данных). В отличие от них, перцептивные хэши можно сравнивать между собой и делать вывод о степени различия двух наборов данных.

Все алгоритмы вычисления перцептивного хэша, обладают одинаковыми базовыми свойствами:

- картинки можно изменять в размере;
- также можно изменять соотношение сторон и даже немного изменять цветовые характеристики (яркость, контраст и т.д.)

В результате изображения все же совпадают по хэшу.

Для получения перцептивного хэша существует несколько распространённых алгоритмов. Одна из простейших хэш-функций отображает среднее значение низких частот.

В изображениях высокие частоты обеспечивают детализацию, а низкие

частоты показывают структуру. Большая, детализированная фотография содержит много высоких частот. В очень маленькой картинке нет деталей, так что она целиком состоит из низких частот.

3.3.1. Алгоритм получения перцептивного хэша

Ниже представлен алгоритм получения перцептивного хэша.

ШАГ 1. Уменьшение размера. Самым быстрым способом избавления от высоких частот является уменьшение изображения. Его необходимо уменьшить до размера 8x8 так, чтобы общее число пикселей составило 64. В данном случае можно пренебречь пропорциями, так как это позволит хэшу соответствовать всем вариантам изображения, независимо от размера и соотношения сторон.

ШАГ 2. Удаление цвета. Изображение необходимо перевести в градации серого, так что хэш уменьшится втрое: с 64 пикселей (64 значения красного, 64 зелёного и 64 синего) до 64 значений цвета в общей сложности.

ШАГ 3. Нахождение среднего. Далее необходимо вычислить среднее значение для всех 64 цветов.

ШАГ 4. Создание цепочки битов. Для каждого цвета требуется получить значение 1 или 0 в зависимости от того, является ли данное значение большим или меньшим по отношению к среднему.

ШАГ 5. Построение хэша. Для этого необходимо перевести 64 отдельных бита в одно 64-битное значение. Порядок не имеет значения, если он сохраняется постоянным.

Итоговый хэш не изменится, если картинку масштабировать, сжать или растянуть. Изменение яркости, контраста, цвета также незначительно повлияет на результат. Главным преимуществом данного алгоритма является то, что он очень быстрый.

Для сравнения двух картинок, необходимо построить хэш для каждой из них и подсчитать количество разных битов с помощью расстояния Хэмминга.

3.3.2. Расстояние Хэмминга для оценки перцептивных хэшей

Расстояние Хэмминга – число позиций, в которых соответствующие символы двух слов одинаковой длины различны. В более общем случае расстояние Хэмминга применяется для строк одинаковой длины любых q-ичных алфавитов и служит метрикой различия (функцией, определяющей расстояние в метрическом пространстве) объектов одинаковой размерности.

$$d_{ij} = \sum_{k=1}^{p} |x_{ik} - x_{jk}|$$

Пример:

$d(1011101, 1001001) = 2$

$d(2173896, 2233796) = 3$

$d(toned, roses) = 3$

Ниже представлены данные, на основе которых следует анализировать сравниваемые изображения:

- если расстояние Хэмминга равно нулю, то скорее всего, сравниваемые изображения одинаковы, либо представляют собой вариации одного изображения;

- если расстояние равно пяти, то данные изображения довольно схожи, но отличаются в некоторых моментах;

- если расстояние равно десяти и более, то изображения, вероятнее всего, различны.

3.3.3. pHash и дискретное косинусное преобразование

Алгоритм перцептивного хэша является простым и довольно быстрым, однако он дает сбои. Например, данный алгоритм не обнаружит дубликат изображения после гамма-коррекции или изменения цветовой гистограммы. Это связано с тем, что цвета меняются в нелинейной масштабе, в следствие чего происходит смещение положения «среднего» и многие биты меняют

свои значения.

Существует более устойчивый алгоритм сравнения изображений – pHash. В данном методе применяется дискретное косинусное преобразование (DCT) для устранения высоких частот.

Дискретное косинусное преобразование (англ. Discrete Cosine Transform, DCT) является одним из ортогональных преобразований. Вариант косинусного преобразования для вектора действительных чисел применяется в алгоритмах сжатия информации с потерями, например, MPEG и JPEG. Это преобразование тесно связано с дискретным преобразованием Фурье и является гомоморфизмом его векторного пространства.

Далее описан алгоритм реализации данного метода:

ШАГ 1. Уменьшение размера. Необходимо уменьшить изображение до размера 32x32. Данное действие производится с целью упрощения DCT, а не для устранения высоких частот, как это было в алгоритме перцептивного хэша.

ШАГ 2. Удаление цвета. Аналогично, цветовые каналы удаляют, чтобы упростить дальнейшие вычисления.

ШАГ 3. Запуск дискретного косинусного преобразования. DCT разбивает картинку на набор частот и векторов.

ШАГ 4. Сокращение DCT. Первоначальный блок имеет размер 32x32, но на данном этапе необходимо сохранить лишь левый верхний блок 8x8, так как в нём содержатся самые низкие частоты изображения.

ШАГ 5. Вычисление среднего значения. Далее требуется вычислить среднее значение DCT на блоке 8x8, также необходимо исключить из расчёта самый первый коэффициент, чтобы убрать из описания хэша пустую информацию, например, одинаковые цвета.

ШАГ 6. Сокращение DCT. Необходимо присвоить каждому из 64 DCT-значений 0 или 1 в зависимости от того, является ли данное значение большим или меньшим по отношению к среднему. Данный вариант выдержит гамма-коррекцию или изменение гистограммы;

ШАГ 7. Построение хэша. Необходимо найти 64-битное значение. В данном алгоритме существует возможность посмотреть, на что похож отпечаток визуально. Для этого необходимо присвоить каждому пикселю значения +255 и -255, в зависимости от того равно оно 1 или 0, и преобразовать DCT 32x32 (с нулями для высоких частот) обратно в изображение 32x32.

Как и в хэше по среднему, значения pHash можно сравнивать между собой, например, с помощью алгоритма расстояния Хэмминга (сравнивается значение каждого бита и подсчитывается количество отличий).

Хэш-функция по среднему работает гораздо быстрее pHash. Однако, если с изображением осуществляли какие-либо манипуляции, например, добавление текста или изменение других деталей, то данный алгоритм может допускать ошибки. Алгоритм pHash медленнее, но более точно работает с изображениями, которые были модифицированы (при условии, что на изображении модифицировалось не более 25% площади).

3.4. Проектирование интерфейсов для отображения необходимой информации пользователю

При проектировании программного обеспечения, немаловажную роль отводят созданию пользовательского интерфейса. Его важность заключается в том, что по интерфейсу пользователь, работающий с программой, оценивает ее в целом. Графический интерфейс предоставляет пользователю возможность удобной работы с программой, не требуя от него специальных навыков программирования. [2]

3.4.1. Интерфейсы количественного типа

Интерфейс краткого отображения количественной информации с иконками показан на рисунке 11.

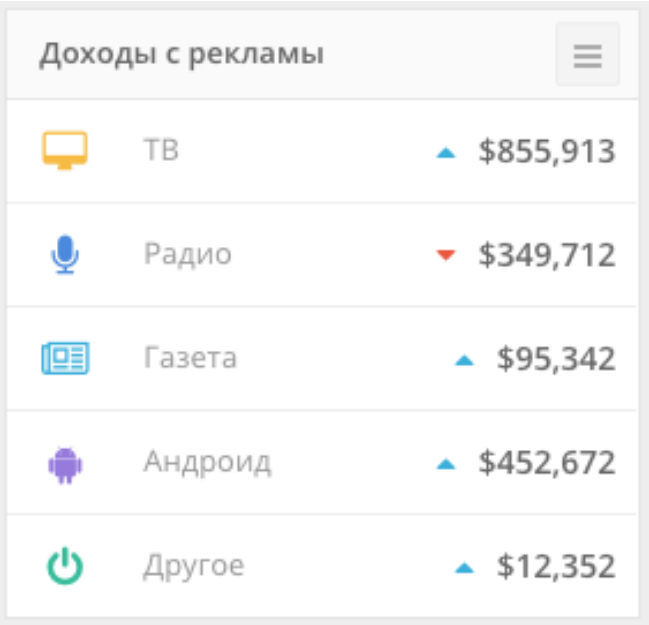


Рисунок 11. Представление виджета доходов

Интерфейс краткого отображения количественной информации в виде списка и кнопкой «подробнее» показан на рисунке 12.

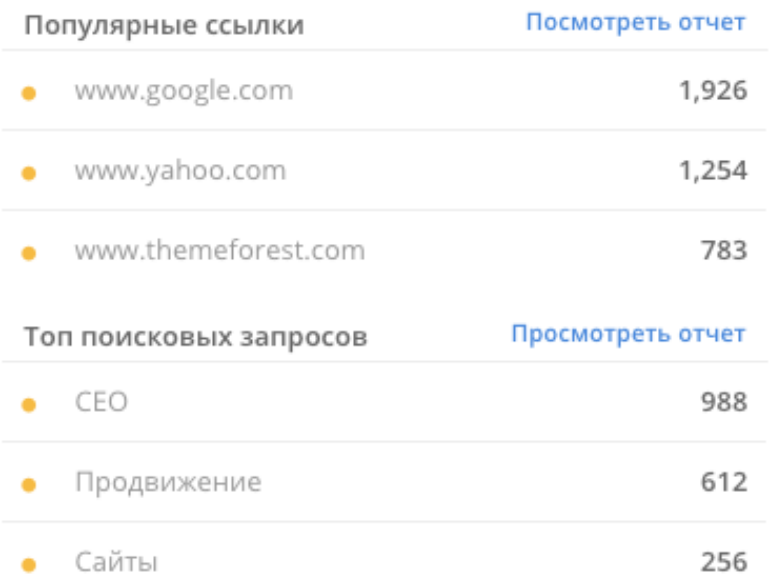


Рисунок 12. Список популярных ссылок и топ поисковых запросов

Интерфейс краткого отображения количественной информации в виде списка (с иконками) и кнопкой «подробнее» показан на рисунке 13.

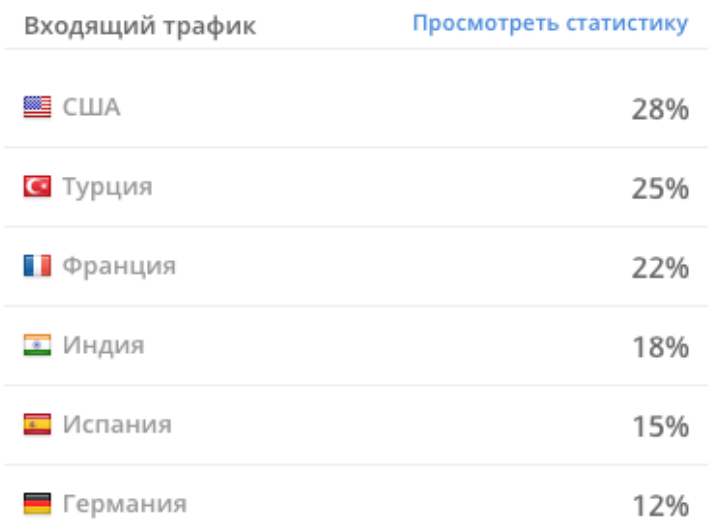


Рисунок 13. Входящий трафик

Интерфейс отображения количества комментарией (отзывов) в виде «плитки» с иконкой и текстом показан на рисунке 14.

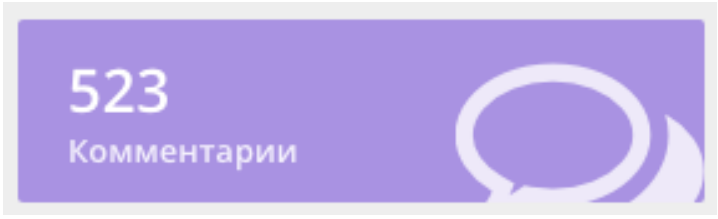


Рисунок 14. Количество комментариев

Интерфейс отображения количества твитов (отзывов) в виде «плитки» с иконкой и текстом показан на рисунке 15.

Рисунок 15. Количество твитов

3.4.2. Интерфейсы графического типа

На рисунке 16 отображена посещаемость пользователей в виде столбчатой диаграммы.

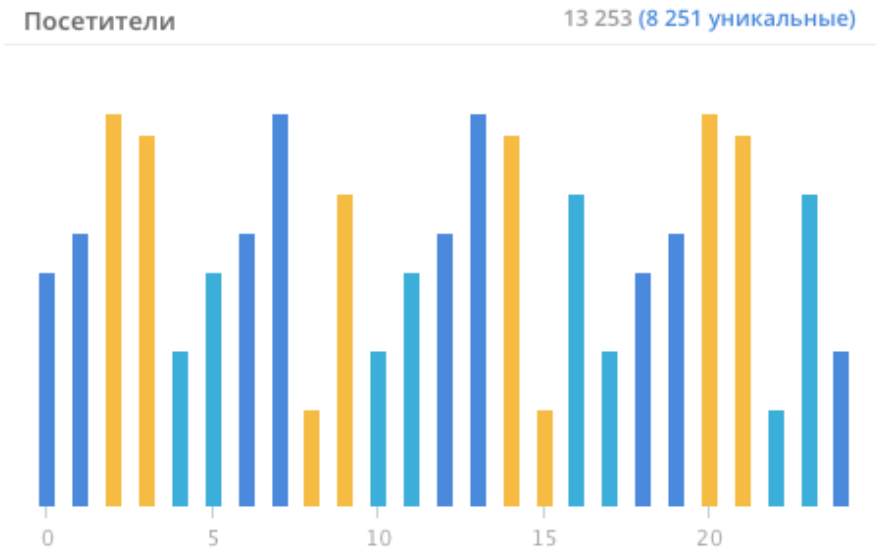


Рисунок 16. Посетители в виде вертикальной диаграммы

На рисунке 17 отображена посещаемость пользователей дифференцируемой по гендерному признаку.

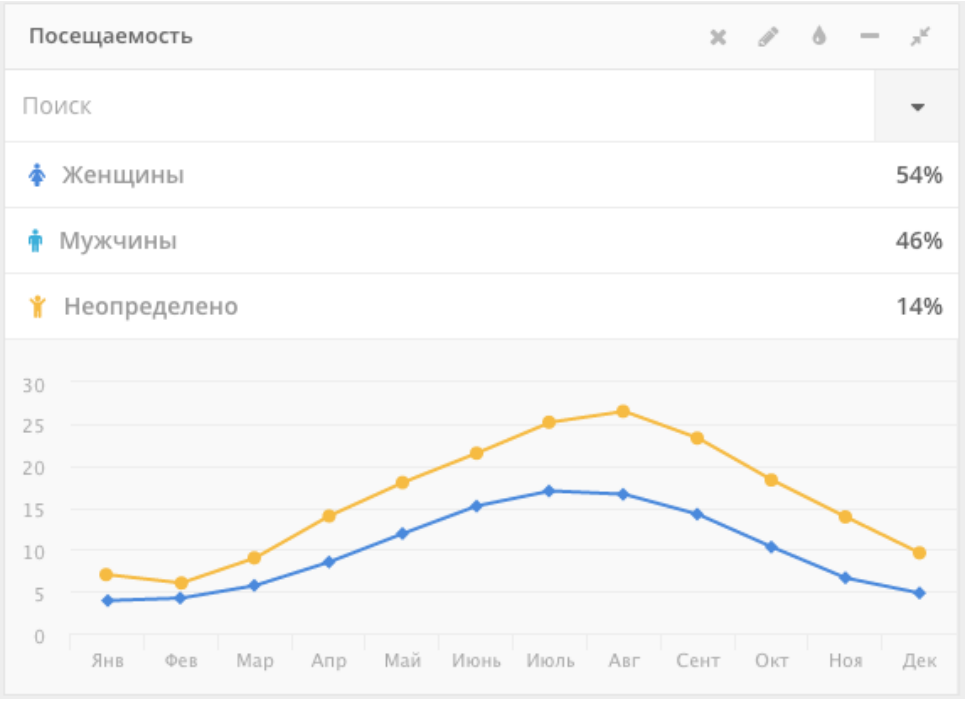


Рисунок 17. Посетители сайта: мужчины, женщины и без указания пола

На рисунке 18 отображена круговая диаграмма браузеров, используемых пользователями изучаемого сайта.

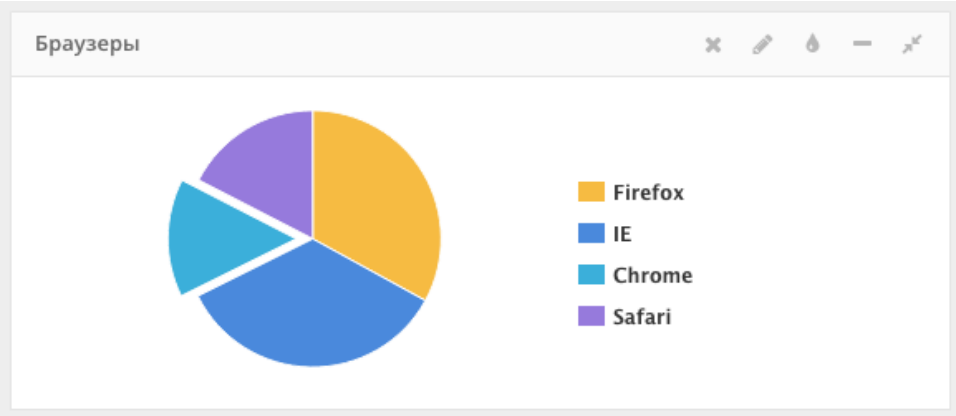


Рисунок 18. Браузеры, используемые посетителями сайта

На рисунке 19 отображена круговая диаграмма количества посетителей, зашедших на сайт через социальные сети: Facebook, Twitter, Google+.

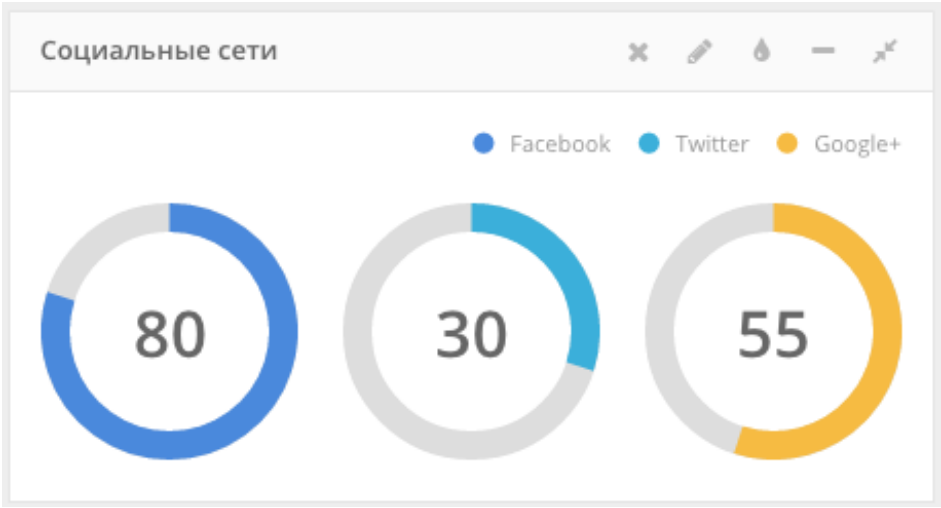


Рисунок 19. Количество посетителей, пришедших через социальные сети

На рисунке 20 отображена диаграмма активности пользователей из разных городов/регионов.

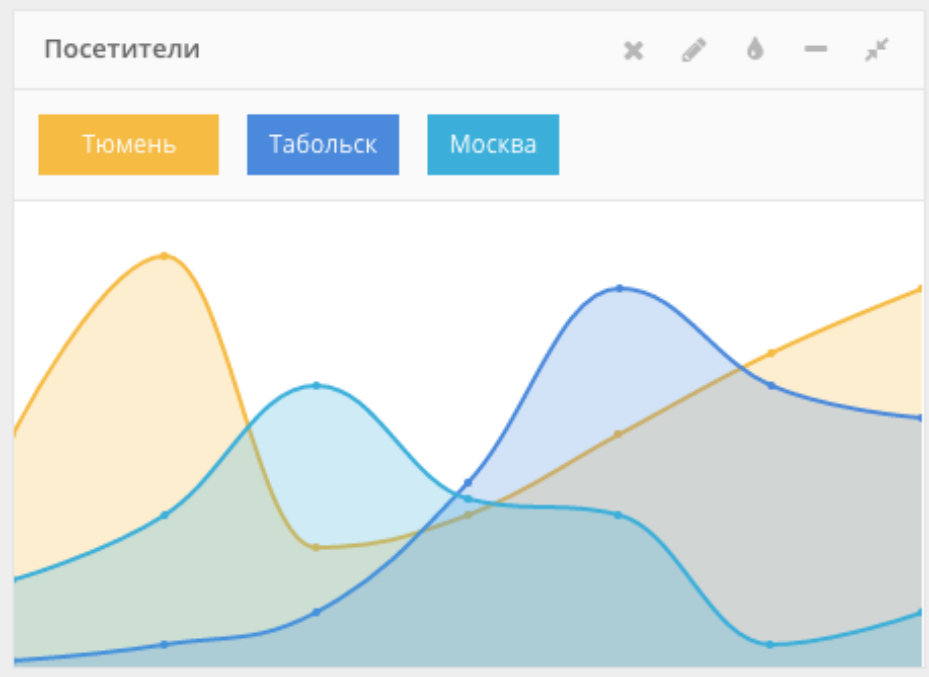


Рисунок 20. Активность пользователей из разных городов/регионов

На рисунке 21 отображена диаграмма активности пользователей из разных городов/регионов (в виде карты).

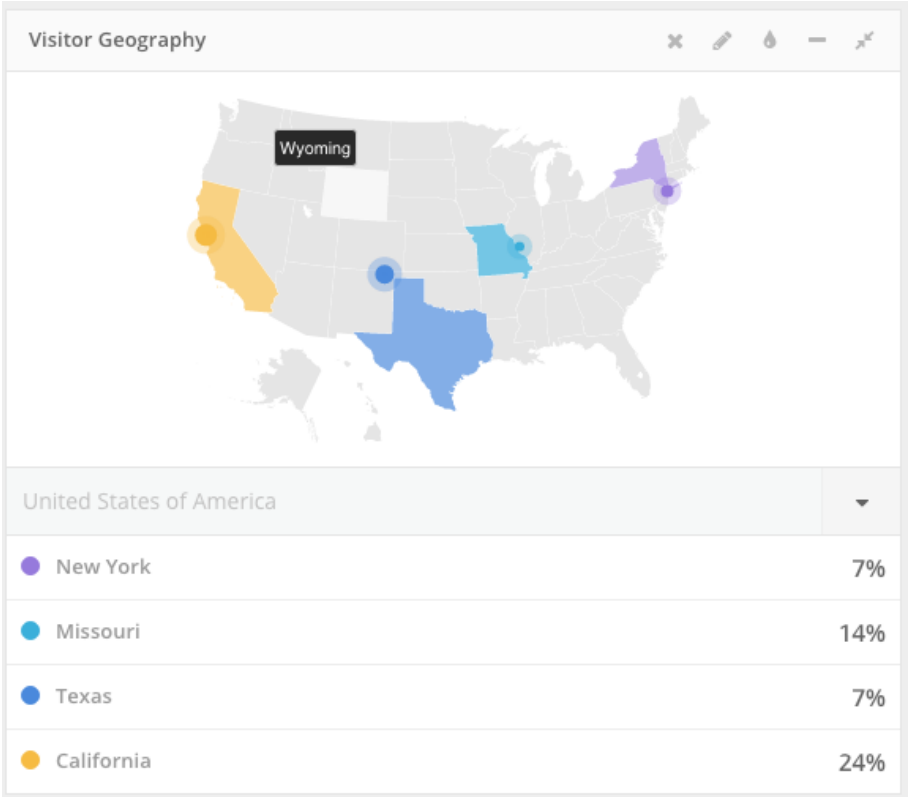


Рисунок 21. Активность пользователей из разных городов/регионов в виде карты

Прототип административной панели со списком сайтов отображен на

рисунке 22.

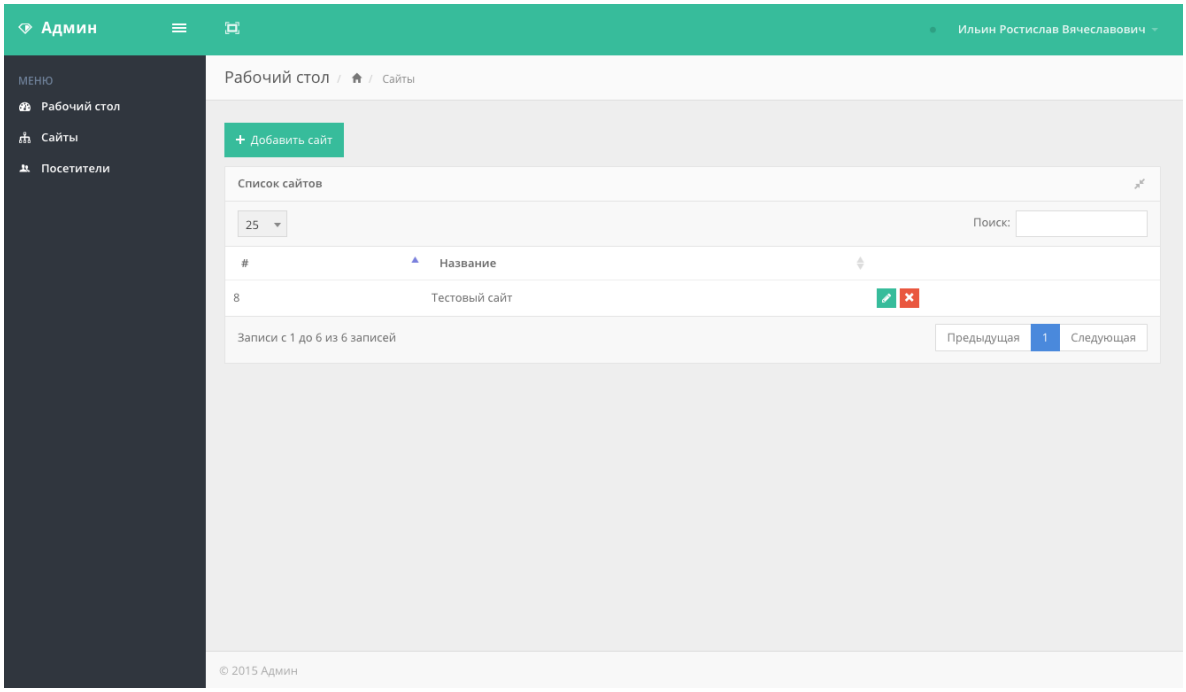


Рисунок 22. Прототип административной панели

В концепте данного прототипа существует следующий функционал:

- отображение всех необходимых и важных показателей на рабочем столе (dashboard);
- возможность добавления сайтов, список сайтов их удаление и редактирование;
- просмотр статистики посещений пользователей.

ГЛАВА 4. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ И НАСТРОЙКА СЕРВЕРА

Данная глава содержит описание реализации веб-приложения и настройки сервера, а именно: разработка каркаса системы, разработка системы блоков и шаблонов, настройка сервера.

4.1. Разработка каркаса для системы

Веб-приложение будет реализовано на платформе Ruby on Rails с использованием языка Ruby и базы данных.

Как уже указывалось ранее, Ruby on Rails (ROR) – опопулярный веб-фреймворк, в основе которого лежит принцип DRY (don’t repeat yourself) и модель MVC (рисунок 23).

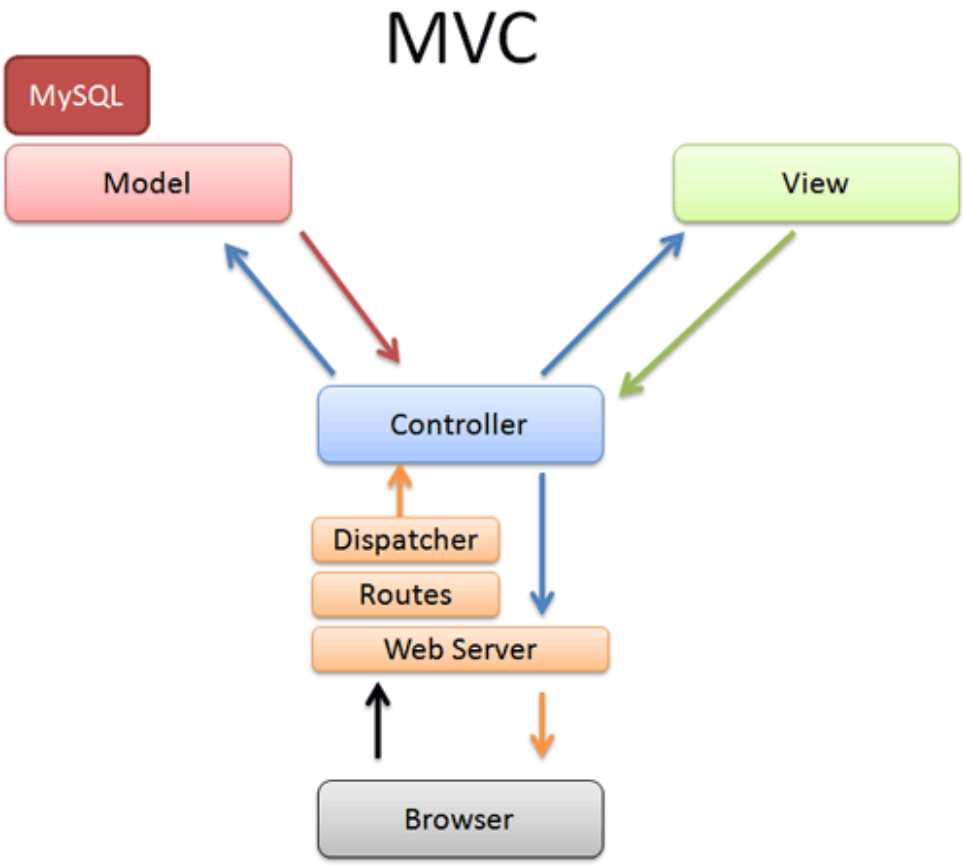


Рисунок 23. Представление модели MVC

Одной из важных особенностей фреймворка ROR является библиотека RubyGems – система управления пакетами для языка Ruby, каждый пакет

системы называется gem, его можно установить и использовать, он является самодостаточным. [37]

Перед началом разработки необходимо установить систему управления пакетами Ruby – RVM (ruby version manager) создать gemset web в котором будут находиться все необходимые gem'ы для веб-разработки. Алгоритм установки приведен далее:

1. Установка curl и git
2. Установка RVM
3. % bash << (curl -s https://rvm.beginrescueend.com/install/rvm)
4. После установки для удобства работы необходимо добавить путь до RVM в .bash_profile либо .zshconfig (в зависимости от оболочки терминала).
5. Список всех версий ruby – rvm list know
6. Установка ruby версии 2.3.0 – rvm install ruby-2.3.0
7. Использование определенной версии ruby – rvm use ruby-2.3.0
8. Создание gemset'а – rvm use 2.3.0@web --create
9. Просмотр списка gemset'ов – rvm gemset list

RVM поддерживает ряд команд с gemset'ами: создание (create), экспорт (export), импорт (import), удаление (delete), очистка (empty).

Ruby on Rails, как и большинство продуктов на Ruby, является gem'ом, поэтому для работы с ним достаточно установить этот gem:

gem install rails 2.3.6.1

После установки rails для создания нового приложения необходимо ввести следующую команду:

rails new my_project

Данная команда создаст новый rails проект в текущей директории.

Структура проекта представлена на рисунке 24.

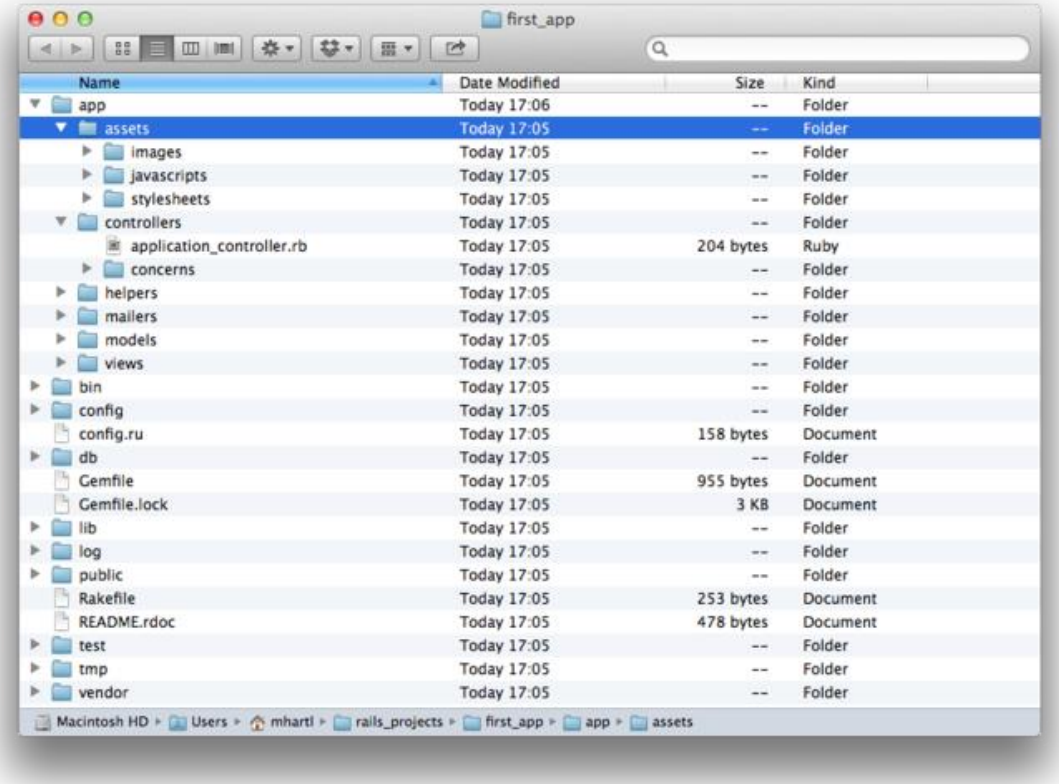


Рисунок 24. Структура Rails проекта

Каталог содержит 3 основных папки, которые отражают модель MVC:

- controllers;
- models;
- views.

В ходе исследования разработки приложения был определен список необходимых gem’ов:

source 'https://rubygems.org'

- ruby '2.3.6.1' – версия Ruby
- gem 'rails', '4.2.5.1' – версия Ruby on Rails
- gem 'pg', '~> 0.17.1' – адаптер для работы базы данных PostgreSQL
- gem 'sass-rails', '~> 5.0' – синтаксический сахар для быстрой разработки таблиц стилей

- gem 'uglifier', '>= 1.3.0' – минификатор javascript кода
- gem 'coffee-rails', '~> 4.1.0' – синтаксический сахар для быстрой разработки javascript с использованием синтаксиса coffeecript
- gem 'jquery-rails' – библиотека для работы с DOM
- gem 'jquery-turbolinks' – библиотека адаптер эмулятора SPA (single page application) с библиотекой jquery
- gem 'jquery-ui-rails' – вспомогательная библиотека интерфейсов для jquery
- gem 'turbolinks' – библиотека, эмулирующая работу SPA (single page application) для ссылок
- gem 'turboboost' – библиотека, эмулирующая работу SPA (single page application) для форм
- gem 'jbuilder', '~> 2.0' – конструктор json представлений
- gem 'sdoc', '~> 0.4.0', group: :doc – генерация документации
- gem 'pghero' – мониторинг базы данных PostgreSQL
- gem 'devise' – авторизация и регистрация пользователей
- gem 'cancancan', '~> 1.10' – распределение прав доступа на основе RBAC
- gem 'carrierwave' – адаптер загрузки изображений
- gem 'rails-i18n' – мультиязычность
- gem 'translit' – траслитерация русских названий, необходимая для загрузки изображений с русскими названиями файлов
- gem 'will_paginate' – постраничная навигация
- gem 'curb' – библиотека CURL для запросов к API
- gem 'truncate_html' – определяет длину html кода и обрезает ее в

читаемом виде

- gem 'mini_magick', '~> 4.2.10' – гем для работы с изображениями (изменение размеров, обрезание)
- gem 'exception_handler' – улучшенная обработка исключений
- gem 'gretel' – хлебные крошки
- gem 'rails-api' – гем для быстрой разработки API
- gem 'apipie-rails' – генерация документации к API
- gem 'magnific-popup-rails', '~> 1.1.0' – интерфейсный гем для увеличения изображений и всплывающих окон

4.2. Система блоков и шаблонов

Одной из сложных задач при разработке является создание форм и таблиц, проверка форм на правильность заполнения и показ ошибок.

Для реализации этой задачи была разработана система блоков – каждый элемент на сайте, который можно было использовать повторно, оборачивался в блок и вызывался с помощью хелпера render_admin_block.

Список реализованных блоков представлен далее:

1) cells;
2) dropdown;
3) field_for_model;
4) form_for_model;
5) link;
6) panel;
7) pnotify;
8) row;
9) table_for_model.

Каждый из блоков имеет следующую структуру:

- input
    - default

        - _default.html.erb
        - _default_example.md
    - email
        - _email.html.erb
        - _email_example.md

Первый уровень – это название блока, второй уровень – тип блока, третий уровень содержит в себе ряд технологий? с помощью которых реализован текущий блок.

Данный подход похож на методолгию БЭМ от компании Яндекс.

Использование разработанных блоков позволяет ускорить процесс создания страниц и свести его к созданию конфигурационных файлов для моделей:

```
class UserModelConfiguration < BaseModelConfiguration

  def construct

    @name = User

    @model_name = 'user'

    # signatures

    @signatures = {

        # title and breadcrumbs

        create_element: 'Создание пользователя',

        edit_element: 'Редактирование пользователя',

        elements: 'Пользователи',

        # buttons

        add_element: 'Добавить пользователя'

    }

    # links

    @links = {
```

```
        elements: 'admin_users_url',
        create_element: 'new_admin_user_url',
        edit_element: 'edit_admin_user_path',
        destroy_element: 'admin_user_path'
}
# properties of table
@table_properties = [{
                    caption: {text: '#', class: 'id' },
                    block: {
                        name: 'cells/text',
                        content: {
                            value: 'content[:row].id'
                        }
                    }
                }]
# field of form
@form_fields = [{
                caption: {text: 'E-mail', class: 'email' },
                block: {
                    name: 'field_for_model/input',
                    view: 'email',
                    content: {
                        value: 'email',
                        tooltip_text: 'Пример: example@gmail.com'
```

```
                }
            },
            validates: {
                rules: {
                    required: true,
                    regex: "^[-a-z0-9!#$%&'*+/=?^_`{|}~]+(?:\\.[-a-z0-9!#$%&'*+/=?^_`{|}~]+)*@(?:[a-z0-9]([-a-z0-9]{0,61}[a-z0-9])?[.]+)+(?:aero|arpa|asia|biz|cat|com|coop|edu|gov|info|int|jobs|mil|mobi|museum|name|net|org|pro|tel|travel|[a-z][a-z])$".html_safe
                },
                messages: {
                    required: 'Поле «E-mail» обязательно для заполнения',
                    regex: 'Некорректно указан E-mail'
                }
            }
        }, {
            caption: {text: 'Пароль', class: 'password' },
            block: {
                name: 'field_for_model/input',
                view: 'password',
                content: {
                    value: 'password',
                    tooltip_text: 'Пароль должен быть не менее 8 символов',
```

```
                    help_text: 'Если не хотите менять пароль, оставьте
поле <u>пустым</u>',
                }
            },
            validates: {
                use_for: { actions: [:new] },
                rules: { required: true,  minlength: 8 },
                messages: {
                    required: 'Поле «Пароль» обязательно для
заполнения',
                    minlength: 'Длина пароля должна быть не менее 8
символов',
                }
            }
        }, {
            caption: {text: 'Имя', class: 'first_name' },
            block: {
                name: 'field_for_model/input',
                content: {
                    value: 'first_name'
                }
            },
            validates: {
                rules: { required: true },
                messages: {
```

```
                    required: 'Поле «Имя» обязательно для
заполнения'
                }
            }
        }, {
            caption: {text: 'Фамилия', class: 'second_name' },
            block: {
                name: 'field_for_model/input',
                content: {
                    value: 'second_name'
                }
            },
            validates: {
                rules: { required: true },
                messages: {
                    required: 'Поле «Фамилия» обязательно для
заполнения'
                }
            }
        }, {
            caption: {text: 'Отчество', class: 'last_name' },
            block: {
                name: 'field_for_model/input',
                content: {
                    value: 'last_name'
```

```
                }
            },
            validates: {
                rules: { required: true },
                messages: {
                    required: 'Поле «Отчество» обязательно для
заполнения'
                }
            }
        }, {
            caption: {text: 'Статус', class: 'user_status' },
            block: {
                name: 'field_for_model/select',
                content: {
                    value: 'user_status_id',
                    options: 'UserStatus.all.map{ |datum| [datum.name,
datum.id] }'
                }
            },
            validates: {
                rules: { required: true },
                messages: {
                    required: 'Поле «Статус» обязательно для
заполнения',
                }
```

```
                }
            }, {
                caption: {text: 'Роль', class: 'user_role' },
                block: {
                    name: 'field_for_model/select',
                    content: {
                        value: 'user_role_id',
                        options: 'UserRole.all.map{ |datum| [datum.ru_name,
datum.id] }'
                    }
                },
                validates: {
                    rules: { required: true },
                    messages: {
                        required: 'Поле «Роль» обязательно для
заполнения',
                    }
                }
            }]
    end
end
```

Результат данного конфигурационного файла представлен на рисунках 25-27.

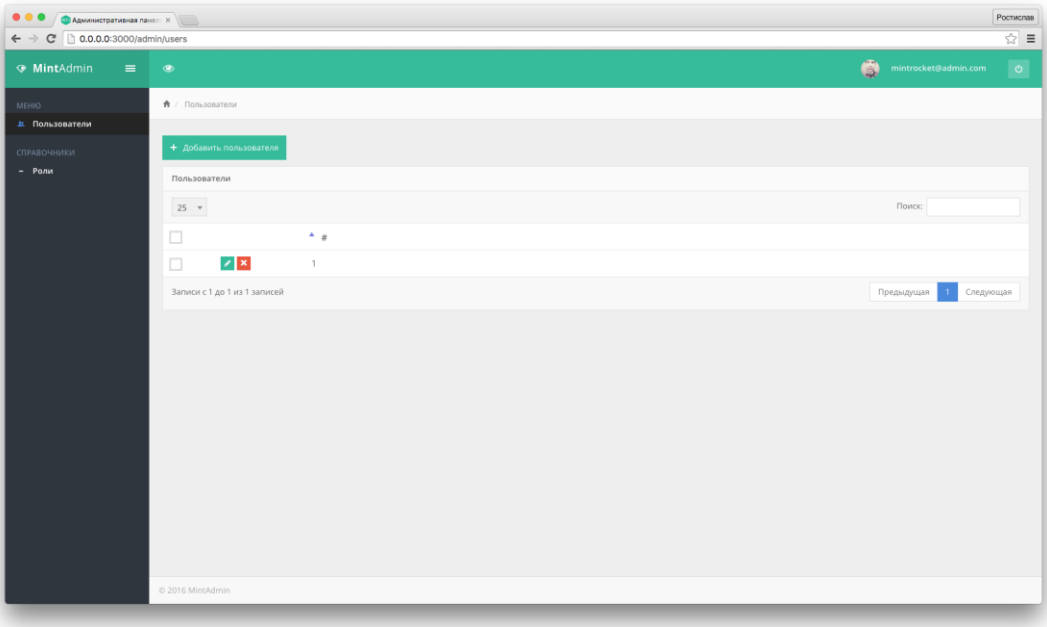


Рисунок 25. Список пользователей

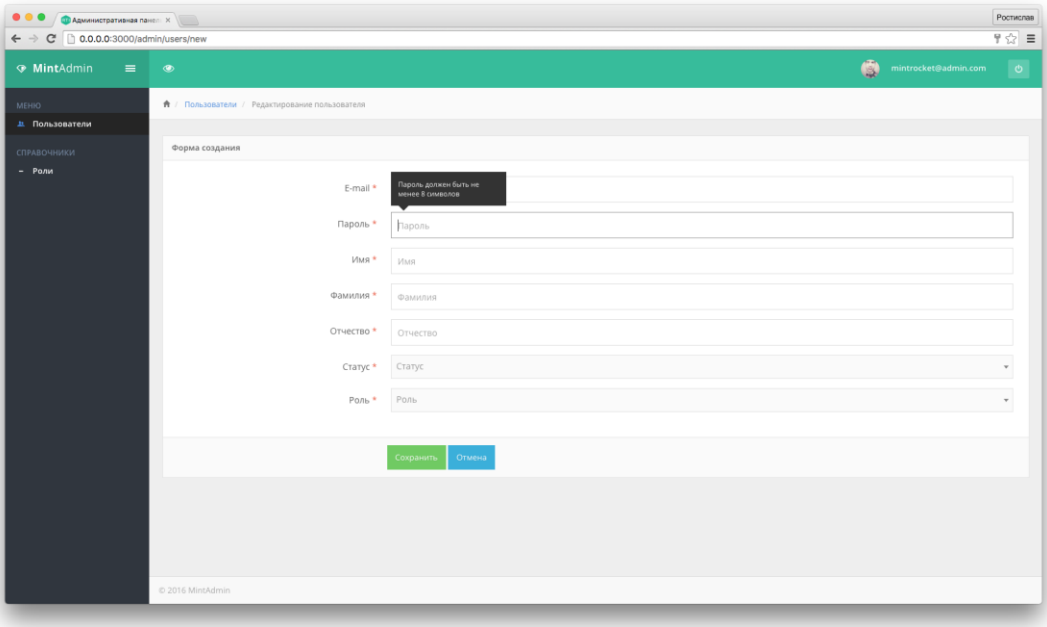


Рисунок 26. Форма создания пользователя

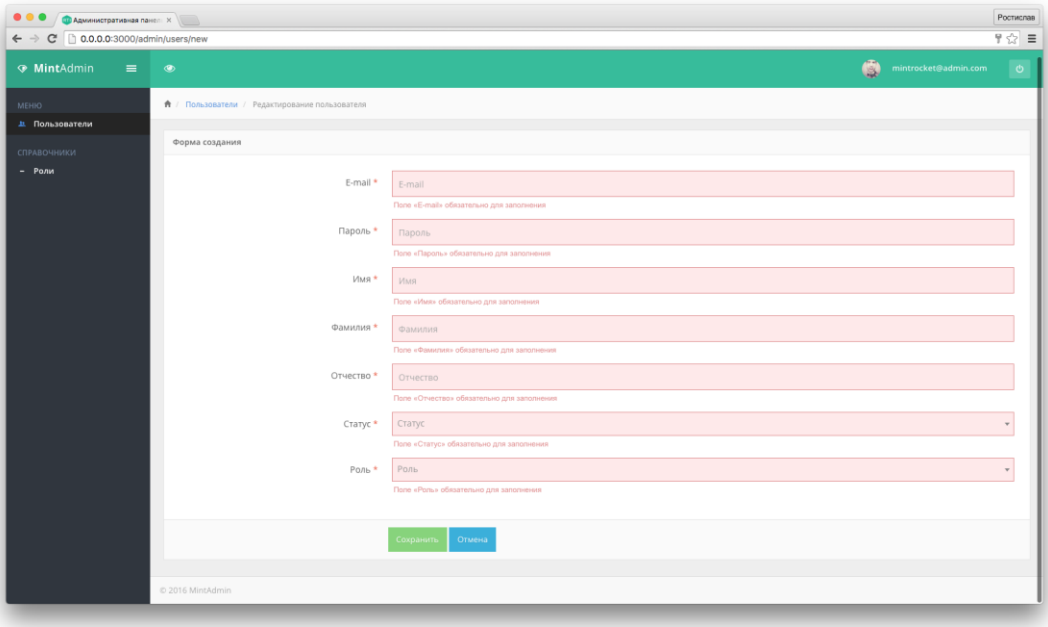


Рисунок 27. Форма создания пользователя в момент сохранения записи без указания обязательных полей

4.3. Настройка сервера и процесса поставки приложения из системы контроля версий

В данном параграфе представлен список необходимых работ для запуска rails приложения на сервере.

4.3.1. Базовые настройки Ubuntu

apt-get update && apt-get upgrade – обновление системы

apt-get install sudo – установка sudo

Чтобы в случае аварийной перезагрузки ошибки были по возможности исправлены автоматически необходимо выставить флаг в конфигурации системы:

nano /etc/default/rcS

=> FSCKFIX=yes

У вновь созданных пользователей домашняя директория должна быть доступна для чтения только самим этим пользователям, для этого необходимо установить флаг в конфигурации создания нового пользователя:

nano /etc/adduser.conf

=> DIR_MODE=0750

Для работы с системами контроля версий необходимо установить git:

apt-get install git-core

Установка многофункциональной оболочки для работы с терминалом позволит ускорить процесс работы:

apt-get install zsh

wget https://github.com/robbyrussell/oh-my-zsh/raw/master/tools/install.sh -O install.sh

zsh install.sh

chsh -s /bin/zsh

Примечание:

Настройка zsh происходит в файле ~/.zshrc. Необходимо сделать псевдоним для zshconfig=”mate ~/.zshrc”

Если требуется установка только для какого-то определенного пользователя:

chsh -s /bin/zsh <username>

Добавляем нового пользователя:

adduser <username>

addgroup <username> sudo

adduser <username>

Выходим из системы и авторизуемся под <username>.

Проводим настройку и установку zsh для этого пользователя так же, как и для предыдущего.

Для работы со временем необходимо установить точное время по Гринвичу и настроить часовой пояс:

sudo apt-get -y install ntp

sudo dpkg-reconfigure tzdata

4.3.2. Установка PostgreSQL

Для работы с БД необходимо установить дистрибутив PostgreSQL и его клиентскую реализацию:

sudo apt-get install postgresql-9.4 postgresql-client-9.4

Примечание:

Для того, чтобы проверить, что всё установилось как положено, необходимо выполнить следующую команду:

ps -ef | grep postgre

Она найдет в списке процессов OS запущенный PostgreSQL.

В целях безопасности устанавливаем пароль для пользователя postgres:

sudo passwd postgres

Все дальнейшие действия должны выполняться из-под пользователя postgres, т.к. только он имеет права на создание/редактирование пользователей, баз данных и таблиц:

su postgres

База данных должна называться так же, как созданная роль пользователя, для взаимодействия с данным БД:

createuser --interactive --pwprompt

createdb <datebasename>

Примечание:

После данных манипуляций появится возможность авторизоваться под пользователем, чье ИМЯ в линукс системе совпадает с ролью в базе данных и таблицей базы данных:

psql

4.3.3. Установка nginx и адапетера Passenger

Установка полного пакета nginx:

apt-get install nginx-extras

Для настройки процесса поставки из CVS на сервер необходимо создать пользователя deploy и отключить возможность авторизации с паролем в целях безопасности:

adduser deploy

passwd -l deploy

Проект нужно расположить на сервер, для этого создаем папку в любой

удобной директории. Обычно это директория /var/www/example.com

```
sudo mkdir -p /var/www/example.com
sudo chown -R deploy:deploy /var/www/example.com
sudo chmod -R 755 /var/www
```

Чтобы сервер начал слушать запросы и отправлять их в папку с проектом, необходимо создать конфигурационный файл в папке /etc/nginx/sites-avalible/site-config

```
server {
   server_name  www.example.com;
   return 301 $scheme://example.com$request_uri;
}
server {
   listen 80;
   server_name example.com;
   client_max_body_size 1G;
   keepalive_timeout 5;
   root /var/www/example.com/current/public;
   passenger_enabled on;
   location ~* ^/assets/ {
      expires 1y;
      add_header Cache-Control public;
      add_header Last-Modified "";
      add_header ETag "";
      break;
   }
}
```

У nginx’a существует 2 папки с конфигурационными файлами:
Sites-available: содержит в себе список всех конфигураций, но эти конфигурации еще не влияют на работу сервера. Для того, чтобы они начали исполняться, необходимо создать ссылку конфигурационного файла во

вторую папку nginx’a – Sites-enabled:

sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/

После всех выполненных работ nginx может слушать и принимать запросы на сервер, но так как приложение написано на Ruby, для корректной работы нужно установить и настроить адаптер Passenger:

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys 561F9B9CAC40B2F7

sudo apt-get install -y apt-transport-https ca-certificates

sudo sh -c 'echo deb https://oss-binaries.phusionpassenger.com/apt/passenger trusty main > /etc/apt/sources.list.d/passenger.list'

sudo apt-get update

В конфигурации nginx нужно добавить 2 строки, которые будут ссылать на Passenger:

/etc/nginx/nginx.conf

=> passenger_root /some-filename/locations.ini;

=> passenger_ruby /usr/bin/passenger_free_ruby;

sudo service nginx restart – перезапуск сервереа

sudo passenger-config validate-install – проверка passenger

4.3.4. Настройка процесса поставки приложения из системы контроля версий на сервер

Для работы с сервером через ssh необходимо создать ssh ключ:

ssh-keygen -t rsa -C rostixman@gmail.com

ssh-add -L | pbcopy – копируем публичную часть ключа

ssh-add ~/.ssh/path_to_rsa - если ключа не видно, добавляем “агента”

Теперь на стороне сервера заходим под пользователем deploy, создаем папку .ssh в домашней директории и добавляем в нее ключ:

sudo su deploy

cd ~

mkdir .ssh

echo “ssh-rds ….” >> .ssh/authorized_keys

chmod 700 .ssh

chmod 600 .ssh/authorized_keys

Проверяем работоспособность ключа:

ssh deploy@0.0.0.0 'hostname; uptime'

Создаем ssh ключ под пользователем deploy:

ssh-keygen -t rsa -C "info@example.com"

cat ~/.ssh/id_rsa.pub

ssh -T git@bitbucket.org

4.4. Построение логической модели данных

На рисунке 28 отображена логическая модель данных, на которой отображены сущности, их атрибуты и связи между ними. Диаграмма выполнена в методологии IDEF1X. Методология основана на подходе Чена и позволяет построить модель данных, эквивалентную реляционной модели в третьей нормальной форме.

В данной методологии все сущности делятся на зависимые и независимые. Сущность независимая, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность зависимая, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности. Независимая сущность изображается в виде обычного прямоугольника, зависимая – в виде прямоугольника с закругленными углами.

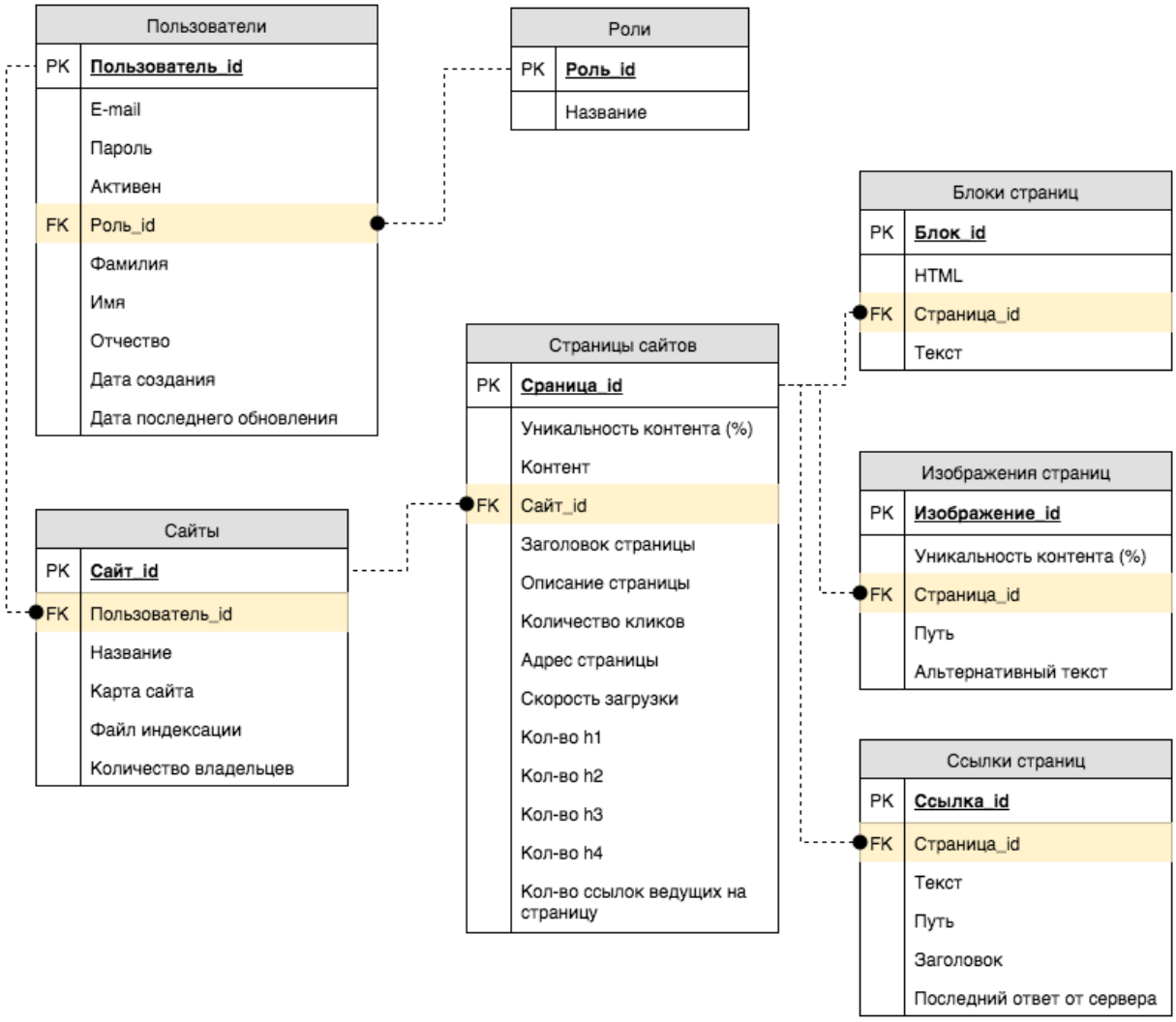


Рисунок 28. Логическая модель данных (IDEF1x)

4.5. Выходные данные

На рисунке 29 представлен результат синтаксического анализа сайта.

#	Характеристика	Значение
1	Количество выделенных блоков	7
2	Количество блоков с повторяющимися атрибутами	4
3	Количество блоков, в которых верстка превышает 35%	3
4	Количество блоков, содержащих мало предложений	4
5	Количество блоков с текстом примерно одинаковой длины	2
6	Количество блоков, содержащих мало текста	5

Рисунок 29. Результат синтаксического анализа

На рисунке 30 представлен результат алгоритма преобразования текста.

#	Характеристика	Значение
1	Количество замененных символов	45
2	Количество сокращений, приведенных к полной форме	4
3	Количество замененных литературных символов	2
4	Количество замененных апострофов, стоящих вместе	3
5	Количество замененных подряд идущих пробелов	10
6	Количество удаленных символов, не входящих в таблицу ASCII	0
7	Количество удаленных управляющих символов	0

Рисунок 30. Результат преобразования текста

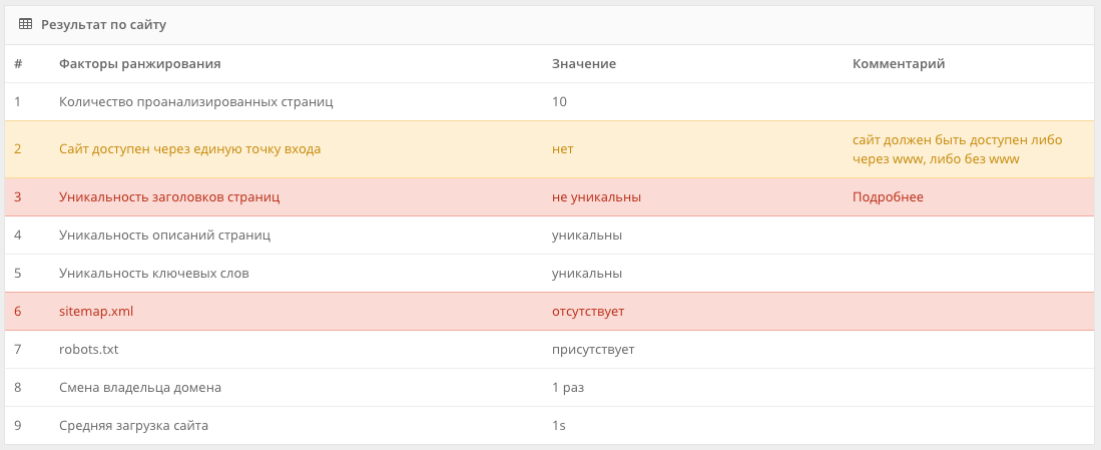
На рисунке 31 представлен результат анализа одной страницы сайта.

Результат по странице

#	Факторы ранжирования	Значение	Комментарий
1	Уникальность контента	50%	Желательно иметь уникальность больше 80%
2	Заголовок страницы	Продажа квартир в Тюмени	
3	Длина заголовка страницы	25 символов	Длина должна быть 80-90 символов
4	Описание страницы	Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus.	
5	Длина описания страницы	191 символ	Длина не должна превышать 200 символов
6	Количество кликов до попадания на страницу	3	Не должно превышать 3 клика
7	URL	www.recom72.com/prodaja-kvartir-tyumen	
8	Слова в URL	prodaja, kvartir, tyumen	@thehawk
9	Скорость загрузки	0.2s	Не должна превышать 3 секунд
10	Тэг h1	Продажа квартир; Продажа квартир	По возможности должен совпадать с заголовком страницы
11	Количество тэгов h1	2	h1 должен быть 1 на странице
12	Тэг h2	Продажа в Тюмени; Продажа в Москве	
13	Количество тэгов h2	2	Должно быть не больше 2-х h2 тэгов на странице
14	Тэг h3	---	
15	Количество тэгов h3	0	Должно быть не больше 3-х h3 тэгов на странице

Рисунок 31. Результат анализа одной страницы сайта

На рисунке 32 представлен результат анализа сайта.



Результат по сайту

#	Факторы ранжирования	Значение	Комментарий
1	Количество проанализированных страниц	10	
2	Сайт доступен через единую точку входа	нет	сайт должен быть доступен либо через www, либо без www
3	Уникальность заголовков страниц	не уникальны	Подробнее
4	Уникальность описаний страниц	уникальны	
5	Уникальность ключевых слов	уникальны	
6	sitemap.xml	отсутствует	
7	robots.txt	присутствует	
8	Смена владельца домена	1 раз	
9	Средняя загрузка сайта	1s	

Рисунок 32. Результат анализа сайта

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была достигнута основная цель, решены частные задачи и получены следующие результаты и выводы:

1. Выполнен сравнительный анализ существующих систем анализа посещаемости сайтов: все системы обладают очень большими возможностями и высоким потенциалом. Также были определены системы, максимально подходящие для решения поставленных задач.

2. Проанализированы существующие системы сканирования сайтов, выявлены их особенности, принц работы и проведен сравнительный анализ. Выявлены основные факторы, влияющие на продвижение сайта в поисковых системах.

3. На основе полученных знаний о системах мониторинга и сканирования сайтов реализован синтаксический анализ сайтов с использованием работы с html документом страницы. Изучены альтернативные методы решения синтаксического анализа их плюсы и минусы.

4. Разработано веб-приложение, в котором реализован алгоритм синтаксического анализа страниц сайтов, проверка всех ключевых CEO показателей сайта и их отображение в системе мониторинга, авторизация пользователей, добавление/редактирование и мониторинг нескольких сайтов. Приложение построено на клиент-серверной архитектуре, с использованием модели MVC и фреймворка Ruby on Rails.

5. В процессе реализации веб-приложения были изучены методики настройки веб-серверов на базе nginx; настроена база данных PostgreSQL с возможностью удаленного подключения. Настроен процесс поставки/развертывания (deploy) локальной копии веб-приложения на сервер.

6. Реализована проверка основного текста страницы на уникальность. Изучена работа с API поисковой системы Google для множественных

запросов и получения результата.

7. Реализована проверка изображений на уникальность: преобразование изображения в перцептивный хэш, сравнение хэшей с помощью расстояния Хэмминга.

Результаты исследования опубликованы в сборнике научных статей «Математическое и информационное моделирование», результаты апробированы на XX международной конференции «Информационно-вычислительные технологии и их приложения», а также получен грант по программе «УМНИК» по Тюменский области.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Бокарев Т.А. Способы продвижения компании в сети Интернет//Маркетинг и маркетинговые исследования. – 2009. – № 4.
2. Брауде Э. Технологии разработки программного обеспечения. – СПб: Питер, 2004. – 655 с.
3. Волченков Е. Программная инженерия. Стандартизация пользовательского интерфейса. – М, 2002.
4. Дейтел Г. Введение в операционные системы. В двух томах/Пер. с англ. Теплицкого Л.А., Ходулева А.Б., Штаркмана В.С. под ред. Штаркмана В.С. – М: Мир, 1987.
5. Клещев А.С., Грибова В.В. Методы и средства разработки пользовательского интерфейса: современное состояние. – 2001.
6. Лутц М. Изучаем Python. 4-е издание. – Символ-Плюс, 2010. – 1280 с.
7. Мазуркевич А. PHP: настольная книга программиста. – Мн.: Новое знание, 2006. – 495 с.
8. Миргород В.С. Метод интеллектуального анализа интернет страниц/В.С. Миргород, И.С. Личканенко, Д.М. Мазур, Р.А. Родригес Залепинос//Информатика и компьютерные технологии – ДонНТУ, 2012. – №8.
9. Успенский И.В. Интернет как инструмент маркетинга. – СПб: БХВ, 2006.
10. Chia-Hui Ch. A survey of Web Information Extraction/Ch. Chia-Hui, K. Mohammed, R.G. Moheb, F. S. Khaled//IEEE Transactions on Knowledge and Data Engineering – 2006. – №18/10. – С. 1411-1428.
11. Crescenzi V., Mecca G. Automatic Information Extraction from Large Websites//Journal of the ACM, Vol. 51, 2004. – №5 – С. 731-779.
12. Gottlob G., Koch C. Logic-based Web Information Extraction – SIGMOD Record, Vol. 33, 2004. – №2 – С. 87-94.

13. Laender A.H.F., Ribeiro-Neto B.A., Juliana S.Teixeria. A brief survey of web data extraction tools. – ACM SIGMOD Record 31(2), 2002. – С. 84-93.

Электронные ресурсы:

14. Беленький А. Текстомайнинг. Извлечение информации из неструктурированных текстов. – Режим доступа: http://www.compress.ru/article.aspx?id=19605&iid=905

15. Википедия «Система поддержки принятия решений». – Режим доступа: https://en.wikipedia.org/wiki/Decision_support_system

16. Википедия «Целевая аудитория сайта». – Режим доступа: https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B0%D1%83%D0%B4%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%B0%D0%B9%D1%82%D0%B0

17. Документация Django. – Режим доступа: http://djbook.ru/rel1.8/

18. Дюк В.А., Асеев М.Г. Поиск if-then правил в данных: проблемы и перспективы. – Режим доступа: www.datadiver.nw.ru/Articles/Problems.htm

19. Инструменты и основные показатели веб-аналитики. – Режим доступа: http://lpgenerator.ru/blog/2015/06/23/instrumenty-i-osnovnye-pokazateli-veb-analitiki/

20. Интернет ресурс/библиотека Bootstrap. – Режим доступа: http://getbootstrap.com/

21. Интернет ресурс/библиотека jQuery. – Режим доступа: http://jquery.page2page.ru/index.php5/%D0%97%D0%B0%D0%B3%D0%BB%D

22. Как увеличить посещаемость сайта бесплатно. Никакого SEO! – Режим доступа: http://great-world.ru/kak-uvelichit-poseshhaemost-sajta-besplatno/

23. Когда веб-аналитика не нужна? – Режим доступа: http://www.cmsmagazine.ru/library/items/web-analytics/when-web-analytics-is-not-needed/

24. Коэффициент конверсии – основы и методы применения. – Режим доступа: http://altblog.ru/excellent-analytics-tip5-conversion-rate-basics-best-practices/

25. Некрестьянов И. Обнаружение структурного подобия HTML-документов. – Режим доступа: http://rcdl.ru/doc....

26. Обзор современных web-технологий построения приложений серверной стороны. – Режим доступа: http://www.lightnet.obninsk.ru/Review/Webreview/257_300.shtml

27. Основы языка программирования Python за 10 минут. – Режим доступа: http://habrahabr.ru/post/31180/

28. Поведенческие факторы и их значение для продвижения. – Режим доступа: http://uguide.ru/news/povedencheskie_faktory/2014-10-01-143

29. Различие Seo-продвижения для Яндекс и Google. – Режим доступа: http://www.artrix.ru/articles/razlichie-seo-prodvizheniya-dlya-yandex-i-google/

30. Самоучитель Python. – Режим доступа: http://pythonworld.ru/samouchitel-python

31. Семь принципов создания современных веб-приложений. – Режим доступа: http://habrahabr.ru/post/242429/

32. Сравнение Яндекс.Метрики и Google Analytics. – Режим доступа: http://pr-cy.ru/lib/seo/Sravnenie-Yandeks-Metriki-i-G..

33. Тестирование вашего сайта: анализируем основные характеристики перед тем, как приступить к продвижению. – Режим доступа: http://www.internet-expert.ru/tools/articles/120/

34. Эффективность web-аналитики за последние несколько лет. – Режим доступа: http://seo-noob.ru/effektivnost-web-analitiki-za-poslednie-neskolko-let

35. Язык Ruby. – Режим доступа: http://acm.mipt.ru/twiki/bin/view/Ruby/WebHome

36. Logical methods in computer science. – Режим доступа: www.lmcs-online.org

37. Ruby on Rails Tutorial. Изучение Rails на Примерах. – Режим доступа: http://railstutorial.ru/chapters/4_0/beginning

ПРИЛОЖЕНИЯ

Приложение 1

Сводная таблица систем анализа посещения сайтов

	Google	Clicky	Piwik	Open Web Analytics	Kissmetrics	Mixpanel	Reinvigorate	Woopra	GoSquared	Chartbeat	Going Up
Мобильное приложение	Да	Да	Да	нет	Да	Да	нет	Да	нет	Да	нет
Статистика в реальном времени	Да	Да	Да	нет	Да	Да	Да	Да	Да	Да	нет
Тепловая карта кликов	нет	Да	нет	Да	нет	нет	Да	нет	нет	нет	Да
Цели и конверсии	Да	Да	Да	Да	Да	Да	нет	Да	нет	нет	нет
Распространенность	49,50%	0,50%	1,30%	<0.1%	0,10%	0,20%	<0.1%	0,20%	<0.1%	0,20%	<0.1%
Бесплатная пробная версия			30 дней		14 дней		14 дней		14 дней	30 дней	
Стоимость в месяц	Бесплатно	Бесплатно или $9,99	$65	Бесплатно	$150	Бесплатно или $150	$10	Бесплатная версия или $79,95	$28	$9,95	Бесплатно