МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ
Заведующий кафедрой
д.п.н., профессор
_____И.Г. Захарова
29 июня 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

РАЗРАБОТКА ИНСТРУМЕНТА ДЛЯ SEO-ПРОДВИЖЕНИЯ ВЕБ-САЙТОВ НА ОСНОВЕ АНАЛИЗА КОНКУРЕНТОВ

02.04.03. Математическое обеспечение и администрирование информационных систем
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнил работу
Студент 2 курса
очной формы обучения
Угриновский
Назарий
Васильевич

Научный руководитель
к.т.н., доцент
Воробьева
Марина
Сергеевна

Рецензент
к.филол.н., доцент
Бидуля
Юлия
Владимировна

Тюмень 2018

ОГЛАВЛЕНИЕ

Введение 2

Глава 1. Особенности ранжирования в поисковых системах 4

1.1. Обзор существующих решений 5

1.2. Постановка задачи 6

Глава 2. Проектирование программного обеспечения 8

2.1. Описание функциональности 8

2.2. Архитектура системы 9

2.3. Файловая структура 13

2.4. Проектирование базы данных 13

Глава 3. Описание используемых алгоритмов 17

3.1. Метод определения значимых факторов ранжирования 17

Метод ранжирования 17

Прямая расстановка 18

Парное сравнение 18

Метода выявления влияющих факторов 19

3.2. Алгоритм ранжирования сайта среди конкурентов 20

3.3. Алгоритм определения процента содержания запроса 20

3.4. Алгоритм выдачи рекомендаций 21

Глава 4. Программная реализация «Smart Promoter» 23

4.1. Клиентская часть 23

4.2. Серверная часть 31

4.3. Развертывание системы 34

4.4. Результаты тестирования работы сервиса 34

Заключение 39

Список литературы 40

Приложения 43

ВВЕДЕНИЕ

Интернет стал одним из основных способов продвижения услуг и товаров для привлечения новых клиентов. Каждая компания стремится занять свою нишу в виртуальном пространстве из-за огромного количества потенциальных покупателей [7]. Тем не менее у любой компании есть свои конкуренты, которые ранжируются в поисковых системах по определенным правилам. Поисковые системы постоянно улучшают алгоритмы ранжирования и обработки данных для отображения релевантных страниц.

Продвижение в сети – это стратегия и процесс интернет-маркетинга для увеличения объема и качества трафика для веб-сайтов в поисковых системах. Данная процедура является одним из основных и долгосрочных этапов в жизненном цикле веб-сайтов.

На данный момент нет открытой информации относительно алгоритмов анализа сайтов, в связи с этим формулы расчета ранжирования сайта неизвестны. Тем не менее SEO-специалисты могут предположить примерное поведение поисковых систем [1]. На основе общеизвестной информации специалисты поискового продвижения могут продвинуть целевые страницы своего сайта или блога в ТОП-10. Падение рейтинга повлечет за собой понижение позиции сайта в поисковой выдаче.

Целью данной работы является разработка инструмента для SEO-продвижения веб-ресурсов на основе рекомендаций, анализирующего конкурентов и предназначенного для выведения сайта на лидирующие позиции в поисковой системе.

Разрабатываемый веб-сервис предназначен упростить и ускорить процесс продвижения веб-сайтов в сети для сокращения расходов и экономии времени специалистов по продвижению.

Для достижения цели были поставлены следующие задачи:

- изучить особенности ранжирования в поисковых системах;
- рассмотреть существующие решения;

- изучить методы и алгоритмы для определения степени значимости факторов ранжирования;
- реализовать алгоритм ранжирования сайта среди конкурентов;
- реализовать алгоритм выдачи рекомендаций;
- спроектировать разрабатываемое приложение;
- реализовать приложение для автоматизированной выдачи рекомендаций для продвижения веб-сайта;
- апробировать работу приложения.

ГЛАВА 1. ОСОБЕННОСТИ РАНЖИРОВАНИЯ В ПОИСКОВЫХ СИСТЕМАХ

Позиция сайта в поисковых системах определяется посредством внутренних алгоритмов этих систем. Определение позиции называется ранжированием.

Алгоритм анализа сайта поисковыми системами следующий:

- Поиск сайтов через посещение всех найденных ссылок на всех страницах.
- Индексация каждого найденного слова на странице.
- Анализ страницы с помощью заранее определенных правил.
- Ранжирование сайта, основанное на факторах ранжирования.

На ранжирование сайта влияет большое количество факторов. Можно выделить следующие категории факторов [2]:

- Контентные факторы - контент веб-страницы, валидность (соответствие международному стандарту W3C [3]) верстки, концентрация ключевых слов, используемых в продвижении (заголовки, title, meta теги).
- Ссылочные факторы - ссылочная релевантность, ссылки, ведущие на сайт, количество, качество сайтов, хранящих ссылки, качество и количество ссылок, тИЦ, PageRank.
- Поведенческие факторы [19] – действия пользователя в поисковой системе до перехода на сайт и на сайте (CTR, время, проведенное на сайте, кликабельность, показатель отказов).
- Доменные факторы (возраст, название, история домена).
- Брендовые факторы (географическая позиция офиса, упоминание бренда);
- Общие факторы (sitemap, структура, SSL, местонахождение сервера, favicon).

1.1. Обзор существующих решений

На сегодняшний день существует большое количество сервисов, позволяющих проанализировать сайт по определенным критериям.

Сервисы продвижения можно разбить на 2 категории:

- внутренняя оптимизация;
- внешняя оптимизация.

Большинство сервисов проверяет текст на уникальность, определяет процент совпадения текста определенному запросу, проводит валидацию HTML, позволяет «закупать» ссылки для продвижения.

Такие сервисы имеют общую направленность и не позволяют проанализировать поисковую выдачу, в реальном времени.

Тем не менее можно выделить такие инструменты, как:

1. ContentMonster – проверяет уникальность текста, следит за качеством работы авторов. Чем выше уникальность текста на сайте, тем выше рейтинг сайта в поиске, соответственно выше позиция. (URL: https://contentmonster.ru).
2. Орфограф – проверяет орфографию. Поисковые системы умеют проверять текст на наличие пунктуационных и орфографических. Орфографические ошибки наравне с другими негативными факторами влияют на поисковое продвижение, в связи с этим текст должен быть уникальным и не содержать ошибок. (URL: https://www.artlebedev.ru/orfograf/).
3. GoGetLinks – предоставляет биржу для покупки ссылок. Покупка ссылок является одним из способов рекламирования сайта в поисковых системах. Данные метод продвижения подразумевает под собой размещение ссылок на сайт на сторонних ресурсах в целях повышения уровня доверия со стороны поисковых систем. Покупка ссылок относится к внешней оптимизации. (URL: https://gogetlinks.net).
4. Markup Validator – валидирует HTML-код. В результате поиска релевантного контента поисковая система проверяет HTML на

валидность. Невалидный код затрудняет поисковым роботам поиск нужного контента для продвижения. (URL: https://validator.w3.org).

5. Seolib.ru – сервис, который позволяет отслеживать динамику роста позиций сайта, сравнивать позиции в разных поисковиках и регионах, определять CTR сайта в выдаче. (URL: https://seolib.ru).

Наряду с этими инструментами в отдельную категорию могут быть выделены такие сервисы, как Google Analytics (URL: https://www.google.com/analytics/), Яндекс.Метрика (URL: https://metrika.yandex.ru/) и Яндекс.Вебмастер (URL: https://webmaster.yandex.ru/), которые проводят полный аудит факторов ранжирования, основываясь на внутренних алгоритмах поисковых систем.

Google Analytics, также как и Яндекс.Метрика, агрегируют информацию относительно поведения пользователей на сайте посредством внедрения в HTML сайта JS-кода, который централизует действия пользователя и составляет карту переходов между страницами.

Яндекс.Вебмастер позволяет явно добавить сайт на индексацию в поисковой системе Яндекс вручную (например, если на сайт нет никаких внешних ссылок) и просмотреть статистику по внешним и внутренним ссылкам.

Перечисленные сервисы предоставляют узконаправленные услуги для решения частных задач, не давая возможности проследить динамику изменения позиции сайта в поисковой выдаче.

Для получения определенных результатов SEO-специалисту требуется использовать все сервисы взаимосвязано, и на основе полученных результатов вручную продвигать ресурсы среди сайтов-конкурентов.

1.2. Постановка задачи

SEO-продвижение направлено на улучшение соответствия содержимого веб-ресурса алгоритмам ранжирования в поисковых системах, а также на выведение ресурса на первые позиции среди сайтов-конкурентов.

Продвижение заключается в систематическом анализе содержимого сайта на наличие неоптимальных значений факторов ранжирования и их оптимизации.

Автоматизация процесса продвижения подразумевает использование совокупности методов по анализу факторов ранжирования.

Разрабатываемый веб-сервис «Smart Promoter» реализует методы автоматического анализа факторов ранжирования для выявления более приоритетных блоков релевантной информации на веб-странице, проведет комплексную оценку веб-ресурса среди конкурентов, позволит упростить и автоматизировать процесс продвижения веб-сайтов в поисковых система Yandex и Google для сокращения расходов и экономии времени специалистов по продвижению.

В частности, основной функцией «Smart Promoter» является возможность продвинуть страницы сайта на высокие позиции поисковой выдачи по определенному поисковому запросу, основываясь на анализе конкурентных продвигаемых сайтов, считающихся эталонными по этому же запросу. Сайты будут анализироваться как по конкретным критериям факторов ранжирования, так и в общем виде, основываясь на совокупности всех оценок факторов по определенным критериям.

ГЛАВА 2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Одним из важных этапов разработки любой сложной системы является этап проектирования. Данный этап распространяется на все слои разработки приложения, включая проектирование архитектуры, создание прототипа, базы данных, файловой структуры, интерфейса и бизнес-модели.

Проектирование программного обеспечения в общем случае основывается на техническом задании (ТЗ), основанном на перечне требований к инструменту. Требования к продукту составляются на основе пользовательских предпочтений, проведений опросов, или составляются заказчиком.

На этапе проектирования можно получить примерное представление, что из себя будет представлять разрабатываемый инструмент, и, если того требует ТЗ, внести соответствующие доработки и правки.

Сама же разработка в большей степени зависит от выбранной модели и методологии [21].

На данный момент можно выделить следующие подходы разработки программного обеспечения:

- итеративный;
- каскадный;
- спиральный.

В рамках разработки сервиса выбран итеративный подход, достоинством которого в сравнении с другими является наращивание функционала, не привязанного к четкому техническому заданию и срокам.

2.1. Описание функциональности

Сервис «Smart Promoter» позволит неограниченному числу пользователей зарегистрироваться в системе в зависимости от тарификации. На основе тарифа пользователю выделится определенный период на использование сервиса и количество продвигаемых сайтов и страниц.

После регистрации предоставляется возможность добавить тестовый сайт для изучения функционала. Тестовый сайт ничем не отличается от обычного, однако по истечении пробного периода будет запрошена оплата, а сайт переведется в режим «ожидается оплата».

Сервис может работать как автономная единица в контексте SEO-продвижения, так и в совокупности с другими SEO-инструментами.

Для работы с сервисом потребуется доступ к файлам продвигаемого сайта для исправления факторов относительно рекомендаций. Тем не менее, имеется возможность проверить факторы без изменения файлов на сервере в режиме реального времени.

2.2. Архитектура системы

Разрабатываемое приложение реализует трехуровневую клиент-серверную архитектуру [6], представленную на рисунке 1.

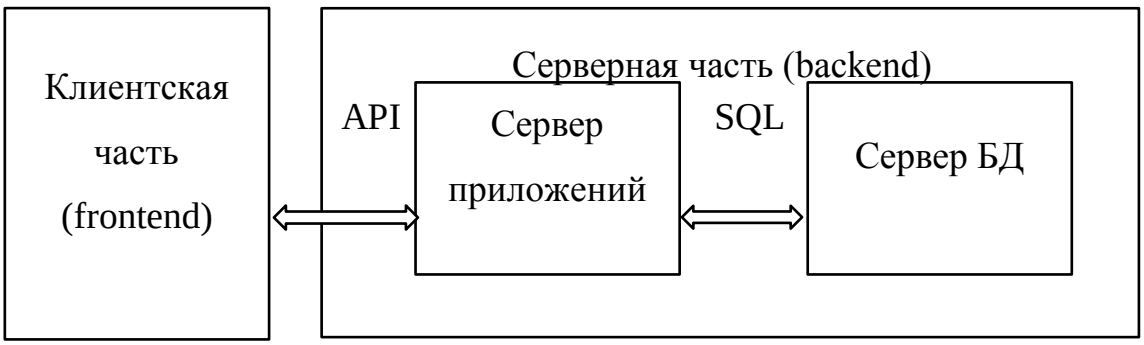

Рисунок 1. Схема трехуровневой архитектуры

Основным преимуществом использования трехуровневой архитектуры является возможность расширения функциональности приложения, так как основные компоненты инкапсулированы друг от друга.

Также для описания архитектуры сервиса как предприятия использовался графический язык Archi [12] в программе Archimate, который предназначен для описания высокоуровневой логической структуры предприятия, его уровней и связей между ними.

С помощью графического языка Archi были описаны такие уровни, как:

1. Бизнес-уровень (см. Рис. 2) представляет информацию о целях, полномочиях и ответственности, основываясь на взаимодействии людей с инструментом.
2. Уровень приложения (см. Рис. 3) описывает приложение на уровне программных компонентов, их связь, работу и процессы взаимодействия с данными.
3. Уровень технологий (см. Рис. 4) описывает аппаратную часть приложения, уровни оборудования.

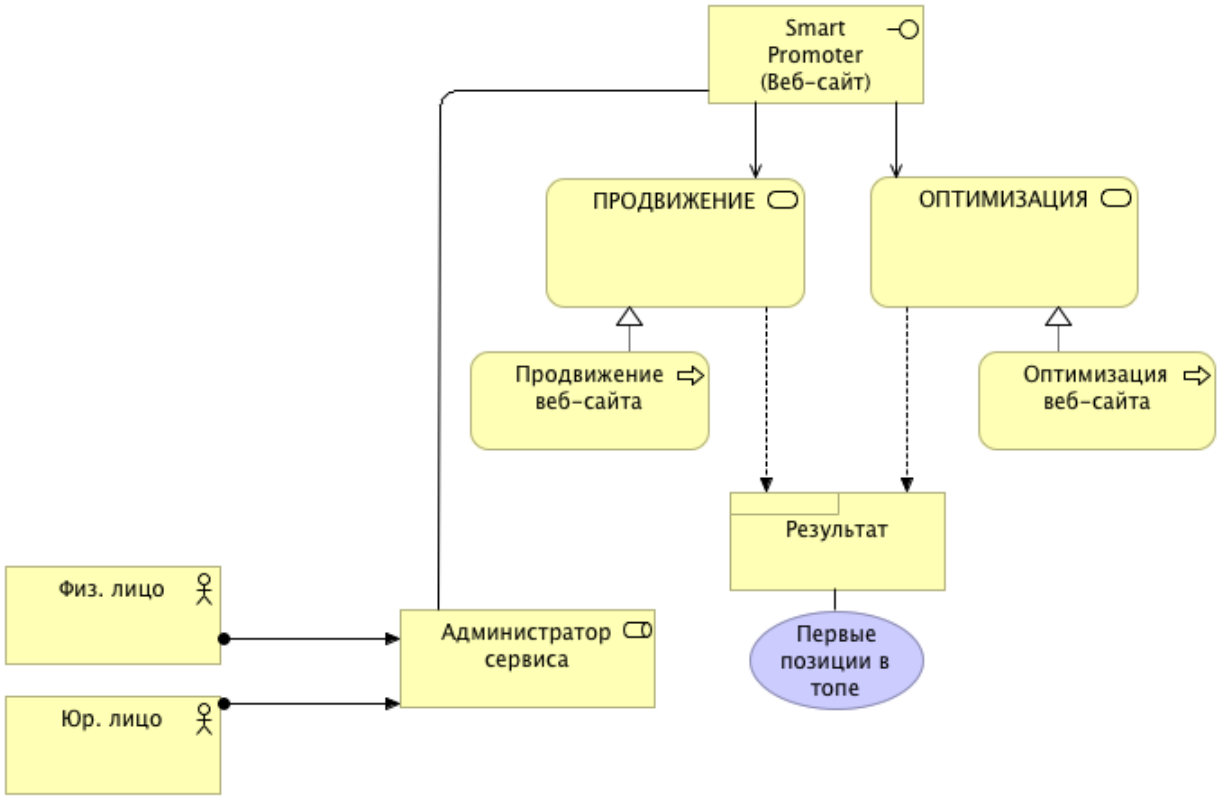

Рисунок 2. Схема Archimate – бизнес-уровень

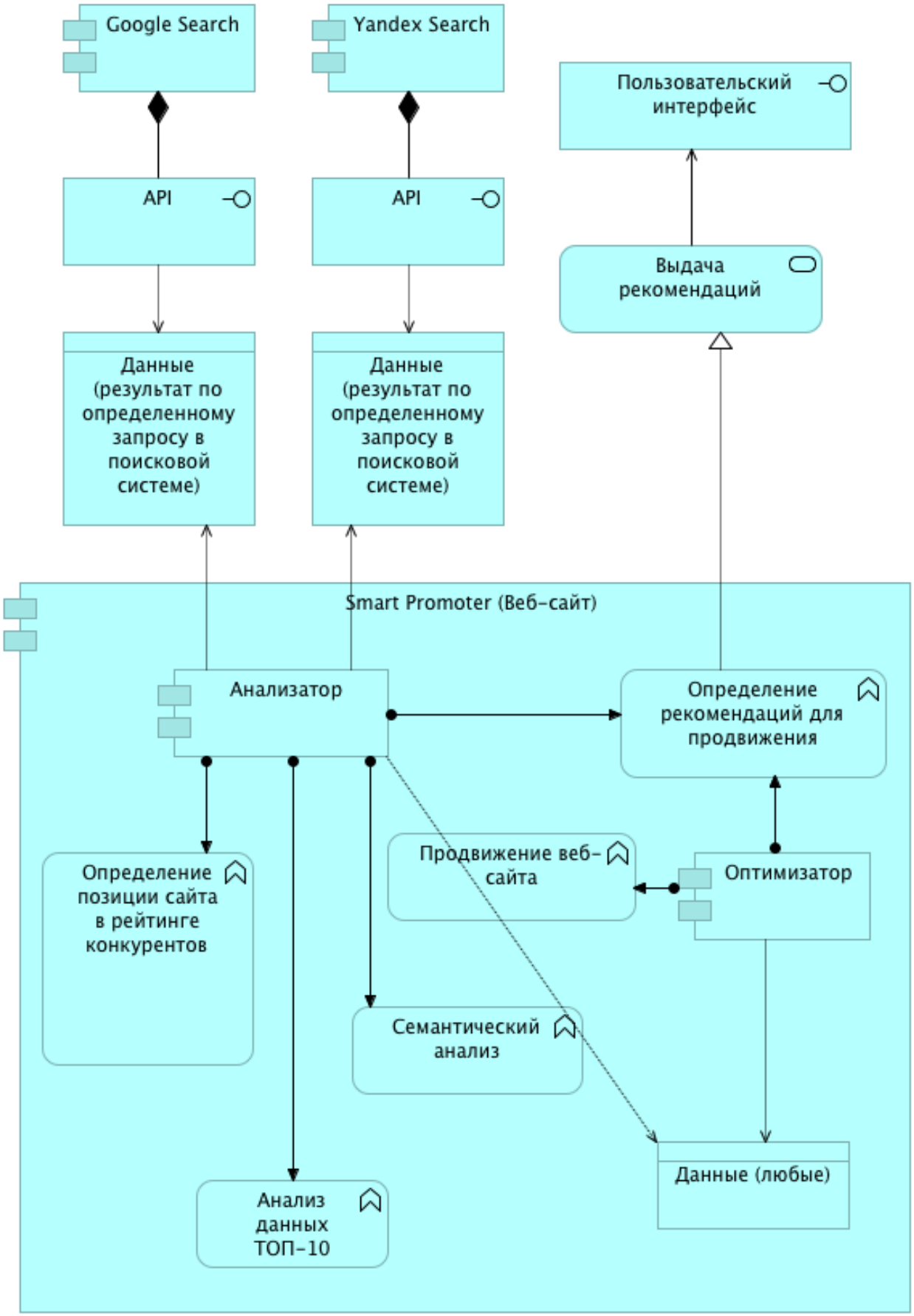

Рисунок 3. Схема Archimate – уровень приложения

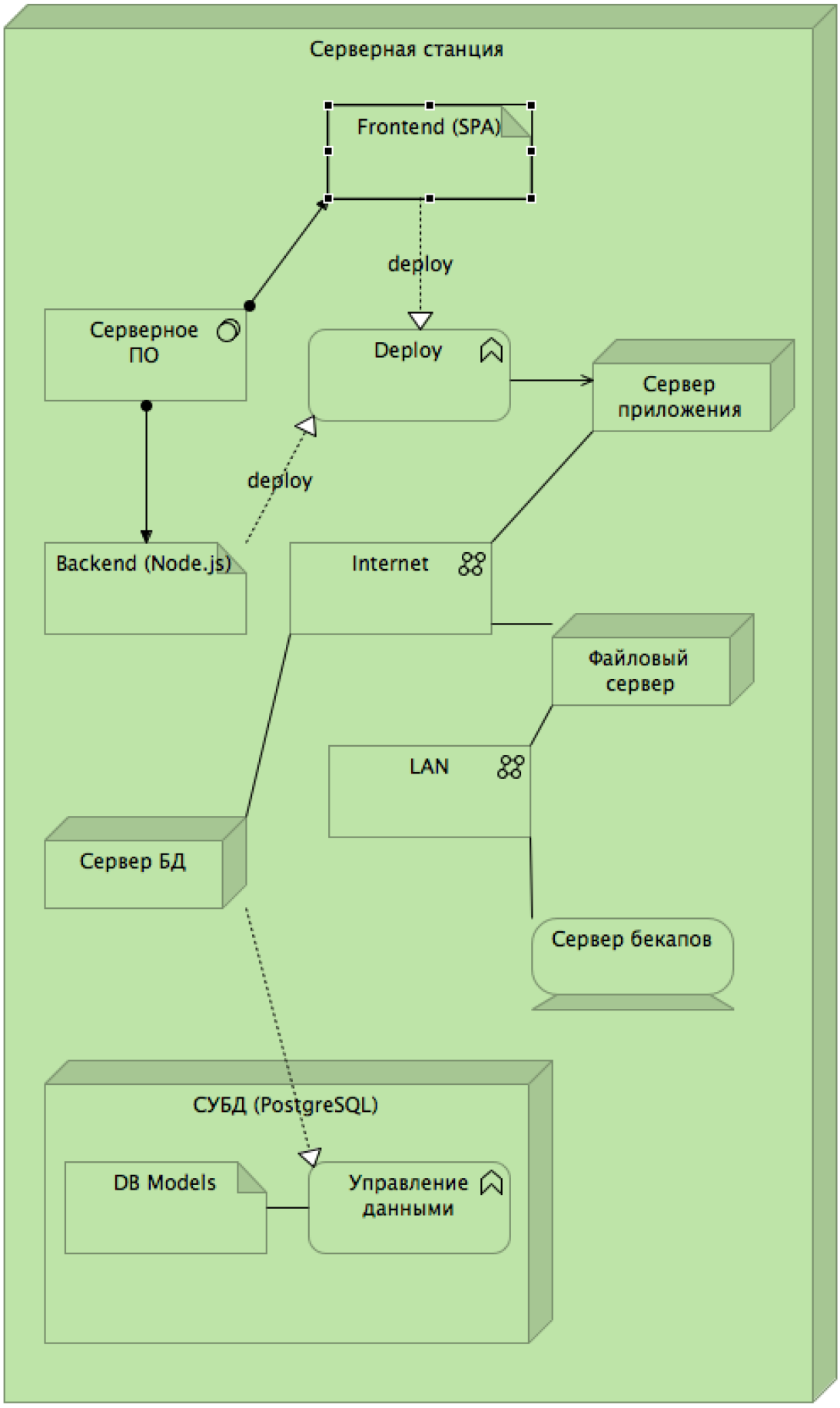

Рисунок 4. Схема Archimate – уровень технологий

2.3. Файловая структура

Файловая архитектура проекта зависит от назначения функционала, в связи с этим структуры frontend (клиент) и backend (сервер) сильно разнятся.

Разработка интерфейса проходит 2 этапа:

- этап проектирования и прототипирования;
- этап интеграции с бизнес-логикой.

Проектирование интерфейса позволяет реализовать общую концепцию приложения, подверженную быстрым модификациям.

Структура состоит из исходных файлов, хранящихся в директории src, и файлов после сборки в директории build. Такой подход позволяет независимо работать с исходными и результирующими данными, разделяя их друг от друга, проводить обработку файлов (сжатие, модификацию и склейку файлов). Данный подход реализован и опубликован в публичном репозитории на https://github.com [10] (см. Приложение).

Структура серверной части в основном зависит от используемого фреймворка, тем не менее большинство фреймворков гибко настраиваемы. AdonisJS предоставляет минимальный набор директорий, основываясь на парадигме проектирования MVC (Model-View-Controller) и не требует следовать определенной схеме расположения исходных файлов.

2.4. Проектирование базы данных

В качестве СУБД выбрана PostgreSQL, которая позволяет хранить JSON и JSONB (бинарный JSON). Тип данных JSON реализует валидацию приходящих данных. JSONB в свою очередь проходит внутреннюю предобработку для оптимального хранения. В связи с этим PostgreSQL может выступать как реляционная, так и документно-ориентированная СУБД.

Используемый веб-фреймворк AdonisJS предоставляет QueryBuilder для работы с различными базами данных, в частности с PostgreSQL.

Для работы с данными была разработано 19 таблиц, описание которых приведено в таблице 1.

Таблица 1. Описание таблиц БД

Таблица	**Описание**
Users	Хранит список зарегистрированных в системе пользователей.
Sites	Содержит список продвигаемых сайтов, принадлежащих конкретным пользователям. Также здесь хранится рейтинг сайта относительно конкурентов, качество сайта в целом, домен и статус.
Systems	Является справочником и содержит список поисковых систем, в которых может быть произведена оптимизация сайта.
Histories	Содержит историю изменения «качества» сайта относительно времени.
Tariffs	Таблица «Tariffs» является справочником и содержит наименование тарифа, цену и период действия.
Requests	Таблица «Requests» хранит список продвигаемых страниц определенного сайта, запрос продвижения, регион, ключевые слова.
Factors	Таблица «Factors» является справочником и содержит наименования факторов.
Regions	Таблица «Regions» является справочником и содержит список регионов страны. Информация о регионе необходима, так как поисковые системы умеют работать с геопозицией.
SiteStatuses	Таблица «SiteStatuses» является справочником и хранит в себе наименования статуса сайта.

Таблица	Описание
Countries	Таблица «Countries» является справочником и хранит в себе список стран.
UserStatuses	Таблица «UserStatuses» является справочником и содержит наименования статусов пользователей.
Criterions	Таблица «Criterions» является справочником и содержит список критериев факторов.
SiteFactors	Таблица «SiteFactors» является сводной таблицей и объединяет таблицы «Sites» и «Factors».
FactorSiteCriterioons	Таблица «FactorSiteCriteries» является сводной таблицей и объединяет «SiteFactors» и «Criterions».
Roles	Таблица «Roles» является справочником и содержит список ролей.
Permissions	Таблица «Permissions» является справочником и содержит список прав.
UsersPermissions	Таблица «UsersPermissions» является сводной таблицей и объединяет «Users» и «Permissions».
UsersRoles	Таблица «UsersRoles» является сводной таблицей и объединяет «Users» и «Roles».
PermissionsRoles	Таблица «PermissionsRoles» является сводной таблицей и объединяет «Roles» и «Permissions».

Схема базы данных представлена на рисунке 5.

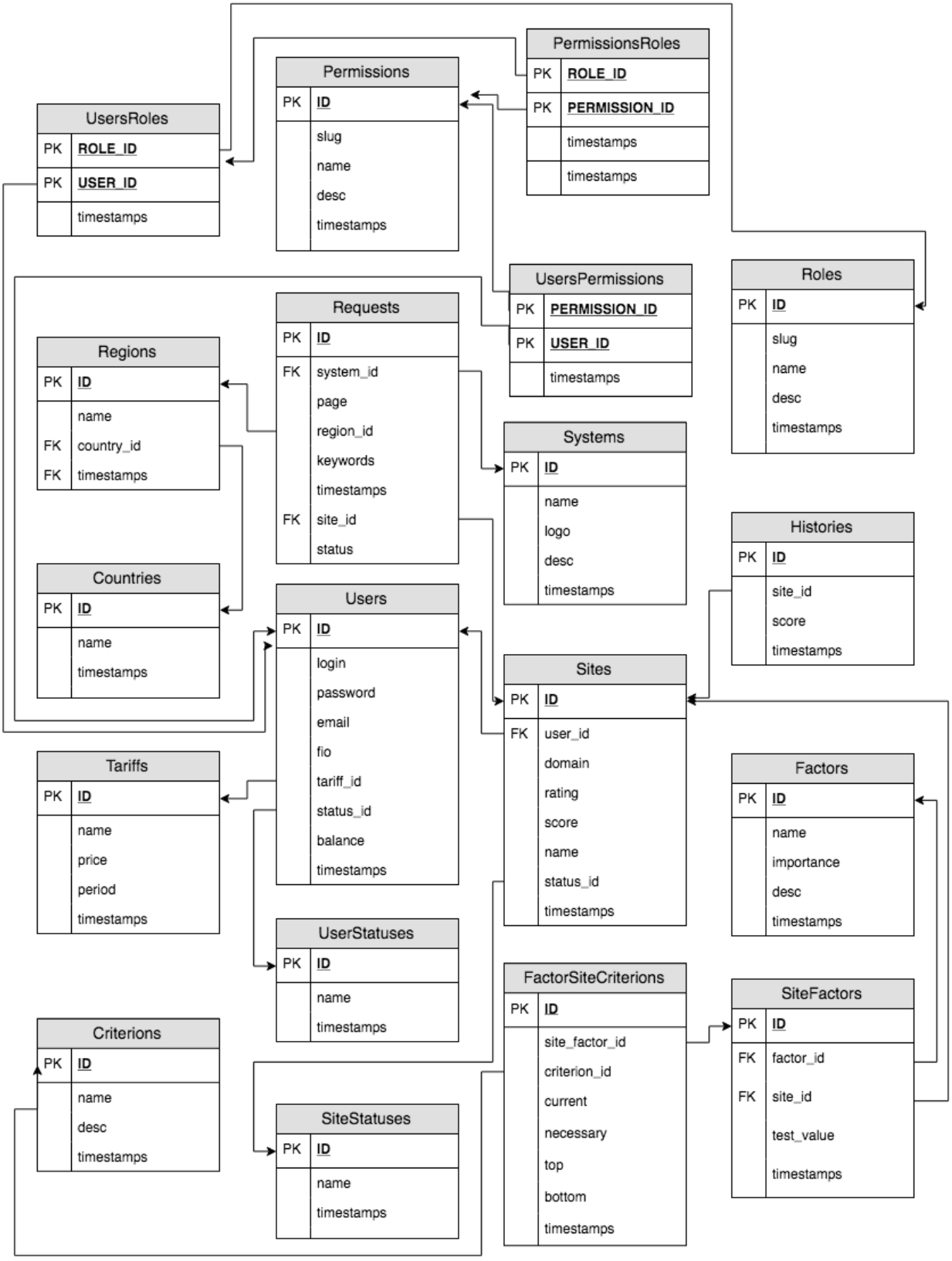

Рисунок 5. Схема БД

ГЛАВА 3. ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ АЛГОРИТМОВ

Для работы сервиса «Smart Promoter» были разработаны следующие алгоритмы:

1. Метод определения значимых факторов ранжирования.
2. Алгоритм ранжирования сайта среди конкурентов.
3. Алгоритм определения процента содержания запроса.
4. Алгоритм выдачи рекомендаций.

Модель ранжирования сайта среди конкурентов и алгоритм выдачи рекомендаций основываются на обработке факторов ранжирования, которые, в свою очередь, могут быть получены благодаря методу определения значимых факторов ранжирования.

3.1. Метод определения значимых факторов ранжирования

Одной из задач при проектировании разрабатываемого приложения «Smart Promoter» является выбор метода определения значимых факторов ранжирования и их весовых коэффициентов.

На данный момент существует большое количество методов определения факторов ранжирования, все они основаны на экспертной оценке. Тем не менее можно выделить самые распространенные из них, которые и были при выполнении данной работы:

- метод ранжирования [4];
- прямая расстановка [8];
- парное сравнение [5].

Метод ранжирования

1. В данном методе эксперт упорядочивает оцениваемые факторы по убыванию относительно степени важности.
2. После определения степени важности определяется согласованность мнений, которую можно получить с помощью коэффициента конкордации Кендалла, вычисляемого по формуле (1).

$$W = \frac{12S}{n^2(m^3 - m)} \quad (1)$$

$$S = \sum_{i=1}^{m} \left(\left(d_i - \bar{d} \right)^2 \right) \quad (2)$$

где n – число экспертов,

m – число ранжируемых факторов,

S – сумма квадратов отклонений всех оценок рангов каждого фактора от среднего значения,

d_i - сумма рангов i-го показателя по всем экспертам.

Коэффициент конкордации Кендалла W может варьироваться в пределах 0 < W < 1 и показывает единогласие экспертов (W =1) или разногласие экспертов (W = 0).

3. После вычисления W для каждого фактора строится матрица преобразования рангов, каждый элемент которой делится на показатель суммы по столбцу.
4. Для получения ранга определяется среднее арифметическое, и на его основе присваивается числовой ранг: чем больше среднее значение, тем выше ранг.

Прямая расстановка

В данном методе, основанном на формуле 3, экспертам предоставляется список факторов, для которых требуется определить числовые коэффициенты k_i, при этом сумма коэффициентов не должна превышать 1.

$$\sum k_i = 1 \quad (3)$$

Парное сравнение

Метод предлагает экспертам сравнить оцениваемые факторы попарно. На основе сравнения по каждому эксперту строится матрица.

Метода выявления влияющих факторов

Для выбора метода выявления влияющих факторов был рассмотрен метод определения коэффициентов «влияющих факторов» [8]. В ходе расчетов авторами статьи было определено наличие кореллированности коэффициентов, рассчитана зависимость коэффициентов от способа усреднения и выявлено, что получаются схожие результаты.

В связи с тем, что итоговые результаты близки, было решено выбрать в качестве метода определения влияющих факторов метод ранжирования.

Для апробации метода выявления влияющих факторов опрошены 10 экспертов, представители тюменских фирм, которым было предложено оценить 39 предполагаемых контентных факторов.

На основе полученных оценок экспертов выбраны следующие факторы:

- ключевые слова (meta keywords);
- мета-описание (meta description);
- заголовок (title);
- H1 (заголовок первого уровня);
- H2 (заголовок второго уровня);
- P (параграф).

Для каждого фактора были определены критерии оценки:

- для всех 6 факторов:
    - длина в символах;
    - процент содержания запроса;
    - частота букв;
    - расстояние между одинаковыми ключевыми словами;
    - месторасположение ключевой фразы к началу строки.
- для h1, h2, p:
    - количество;
    - длина первого элемента;
    - средняя длина всех элементов.
- для p:

- количество непустых абзацев.

3.2. Алгоритм ранжирования сайта среди конкурентов

Метод определения позиции продвигаемого сайта в списке конкурентов основывается на реализации модели суммарной оценки сайта [13].

Модель предусматривает вычисление максимального значения каждого фактора по каждому критерию среди 10 анализируемых сайтов, находящихся на лидирующих позициях (4).

$$Y_i = f(a_i)_{max} \quad (4)$$

где i – порядковый номер фактора ранжирования.

Далее вычисляется общая оценка сайта (5).

$$F(\varphi_i) = \frac{f(\alpha_i)}{\gamma_i} * k \quad (5)$$

где $f(\alpha_i)$ - значение фактора анализируемого сайта,

γ_i – числовое значение фактора эталонного сайта,

k – предложенный экспертами коэффициент (в нашем случае k = 10).

Полученные значения суммируются и приводятся к общему интервалу с помощью линейной нормализации в пределах [0, 1] (6).

$$\tilde{x_i} = \frac{x_i - x_{min\ i}}{x_{max\ i} - x_{min\ i}} * 100\% \quad (6)$$

где x_i – i-е значение исходной выборки,

$\tilde{x_i}$ – нормализованное значение.

Данная модель применена для автоматического нахождения значения качества сайта на основе поискового запроса.

3.3. Алгоритм определения процента содержания запроса

Для определения процента вхождения ключевых слов запроса в анализируемый текст используется формула 7.

$$\nu\ (\%) = \frac{M}{L} * 100\% \quad (7)$$

где ν – частотность вхождения слов,

M – количество повторений слова,

L – количество слов в тексте.

Для определения однокоренных слов выбрана библиотека Az.js, разработанная на языке JavaScript, которая проводит анализ морфологии и выдает список слов, схожих по корню со словом, полученном на входе [22].

Помимо морфологического анализа библиотека Az.js имеет функционал для разбивки текста на токены.

3.4. Алгоритм выдачи рекомендаций

Для выдачи рекомендаций был разработан алгоритм выдачи рекомендаций, который позволяет проанализировать первые 10 позиций сайтов в поисковой выдаче по определенному запросу.

Алгоритм выдачи рекомендаций работает следующим образом:

1. Для каждого анализируемого фактора по всем критериям строится диаграмма конкурентов, отображающая значения критериев сайтов конкурентов по определенному фактору, включая значение продвигаемого сайта.
2. Диаграмма содержит коридор рекомендуемого значения (см. Приложение), имеющий верхнюю и нижнюю границу.
3. Верхняя граница коридора определяется по формуле 8.

$$S = A + \frac{|\ A - Max\ |}{4} \quad (4)$$

где A – среднее значение по одному фактору,

Max – максимальное значение из всех факторов.

4. Нижняя граница определяется формулой 10.

$$S = A - \frac{|\ A - Min\ |}{4} \quad (5)$$

где A – среднее значение по одному фактору,

Min – минимальное значение из всех факторов.

5. На основе полученных данных вычисляется рекомендуемое значение, ориентируясь на которое, пользователь системы приводит значения продвигаемого сайта к выдаваемым рекомендациям. Такой подход рассмотрен авторами статьи [18].
6. Совокупность значений критериев продвигаемого сайта влияет на значение их фактора.

ГЛАВА 4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ «SMART PROMOTER»

Следующим этапом после проектирования приложения идет программная реализация, которая делится на серверную и клиентскую, и развертывание с последующей эксплуатацией.

Разрабатываемый сервис «Smart Promoter» в силу выбранной архитектуры основную бизнес-логику реализует на стороне сервера, клиент же отвечает за отображение данных и минимальные вычисления.

4.1. Клиентская часть

При разработке интерфейса были спроектированы и сверстаны следующие макеты:

- Страница авторизации и регистрации (см. Рис. 6);
- Личный кабинет со списком продвигаемых сайтов (см. Рис. 7);
- Страница с формой добавления сайта на продвижение (см. Рис. 8);
- Страница с формой добавления конкретной страницы на продвижение (см. Рис. 9);
- Карточка сайта (см. Рис. 10);
- Страница общей статистики (см. Рис. 11);
- Страница с диаграммой по конкретному запросу (см. Рис. 12).

Остановимся на каждой странице интерфейса подробно:

1. Страница авторизации сочетает интерфейс авторизации и регистрации в одном месте (см. Рис. 6).

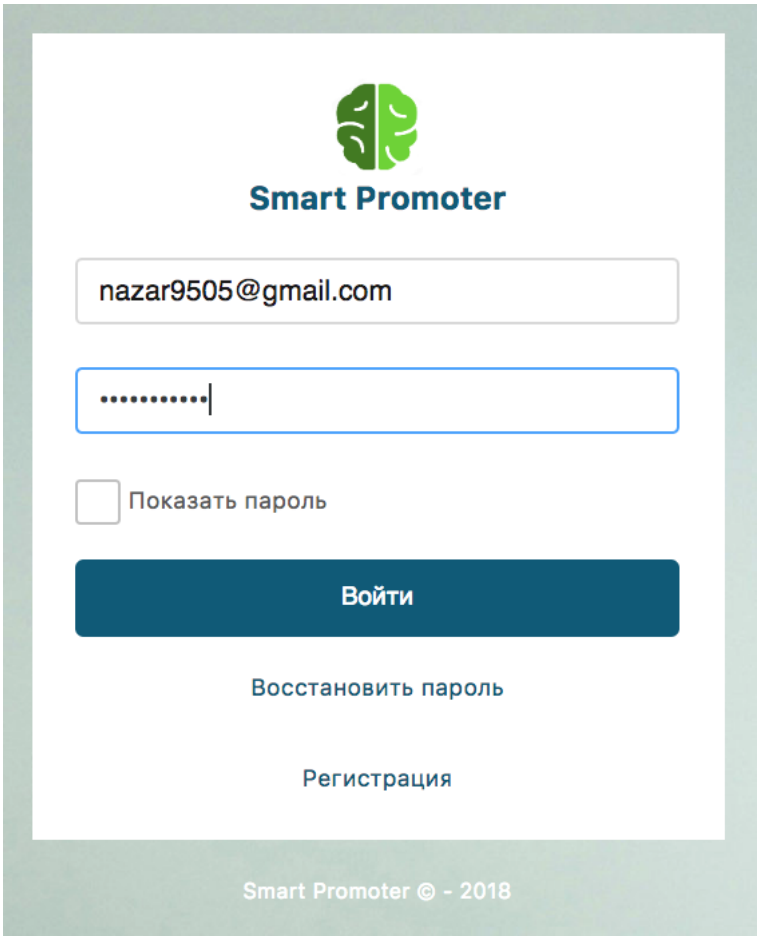

Рисунок 6. Страница авторизации

2. Личный кабинет (см. Рис. 7) отображает таблицу, содержащую список добавленных на продвижение сайтов, привязанных к конкретному пользователю.

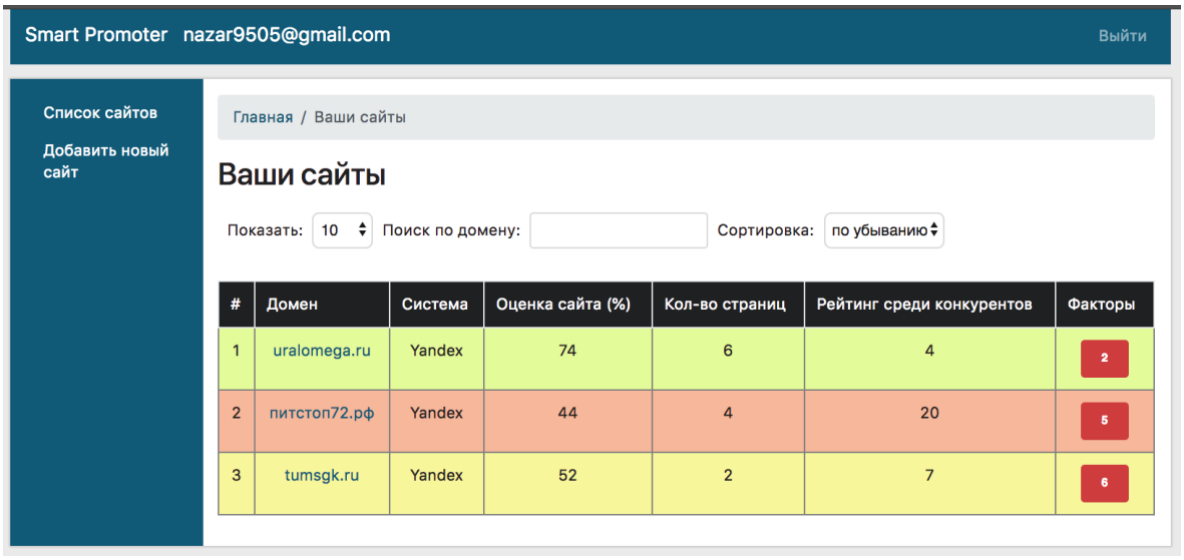

Рисунок 7. Личный кабинет со списком продвигаемых сайтов

3. Страница с формой добавления сайта позволяет зарегистрировать Интернет-ресурс в системе для последующего продвижения (см. Рис. 8).

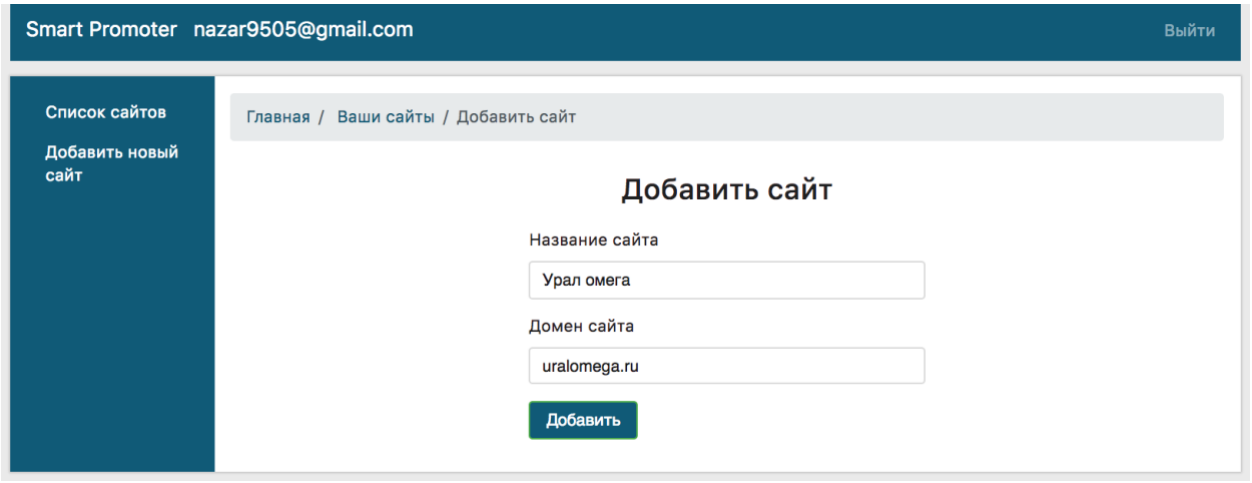

Рисунок 8. Страница с формой добавления сайта на продвижение

4. Страница с формой добавления страницы (см. Рис. 9) позволяет зарегистрировать конкретную страницу для продвижения.

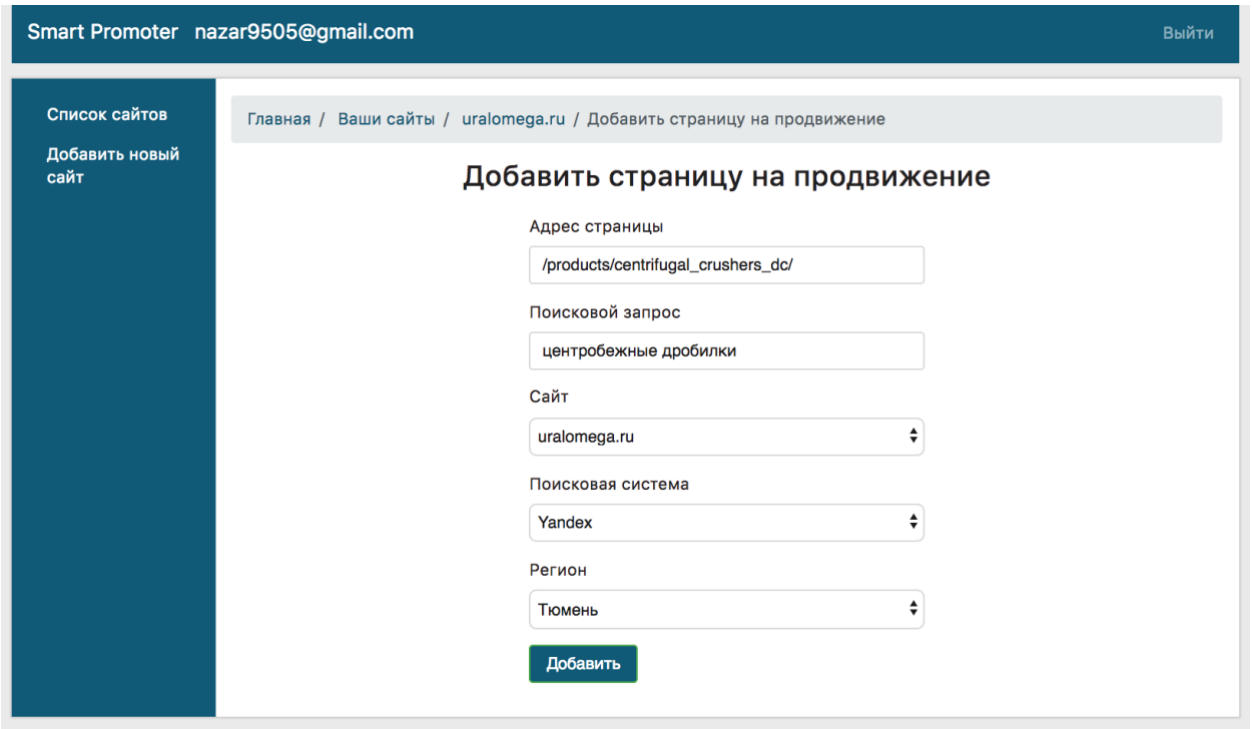

Рисунок 9. Страница с формой добавления конкретной страницы на продвижение

5. Карточка сайта (см. Рис. 10) отображает детальную информацию о зарегистрированном в системе сайте, а именно:
    - программно составить ТОП-10 конкурентов по тематике сайта;
    - просмотреть список продвигаемых страниц

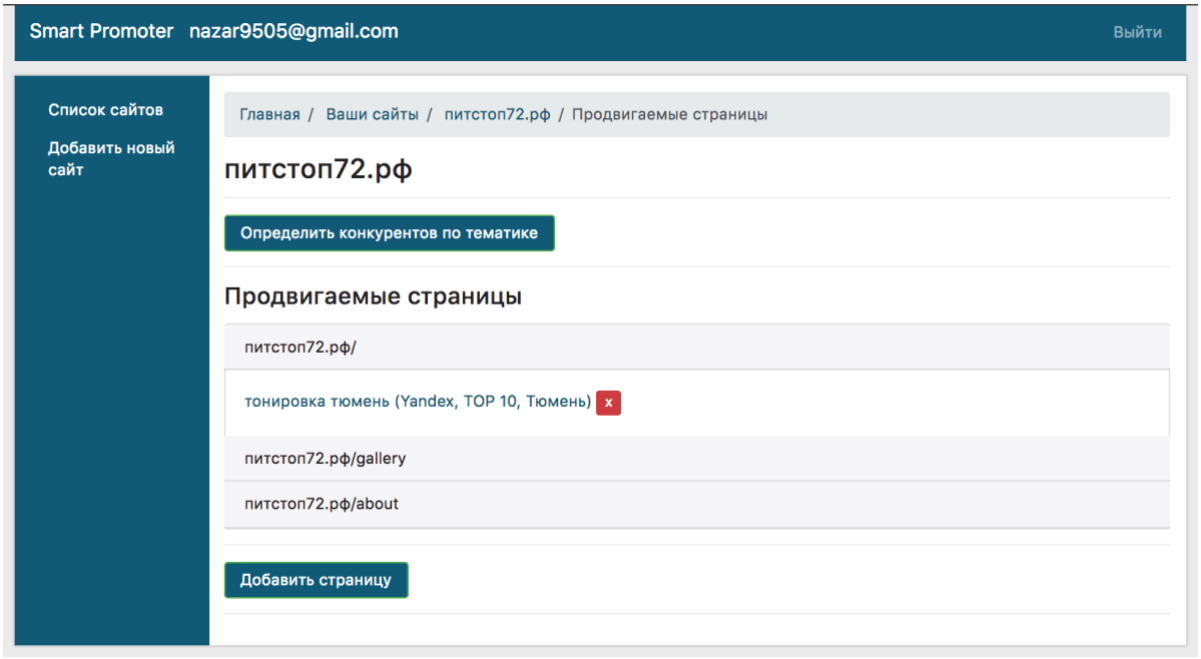

Рисунок 10. Карточка сайта

6. Страница общей статистики (см. Рис. 11) позволяет просмотреть общую информацию о всех зарегистрированных в системе сайтах, получить список сайтов-конкурентов по определенным поисковым запросам.

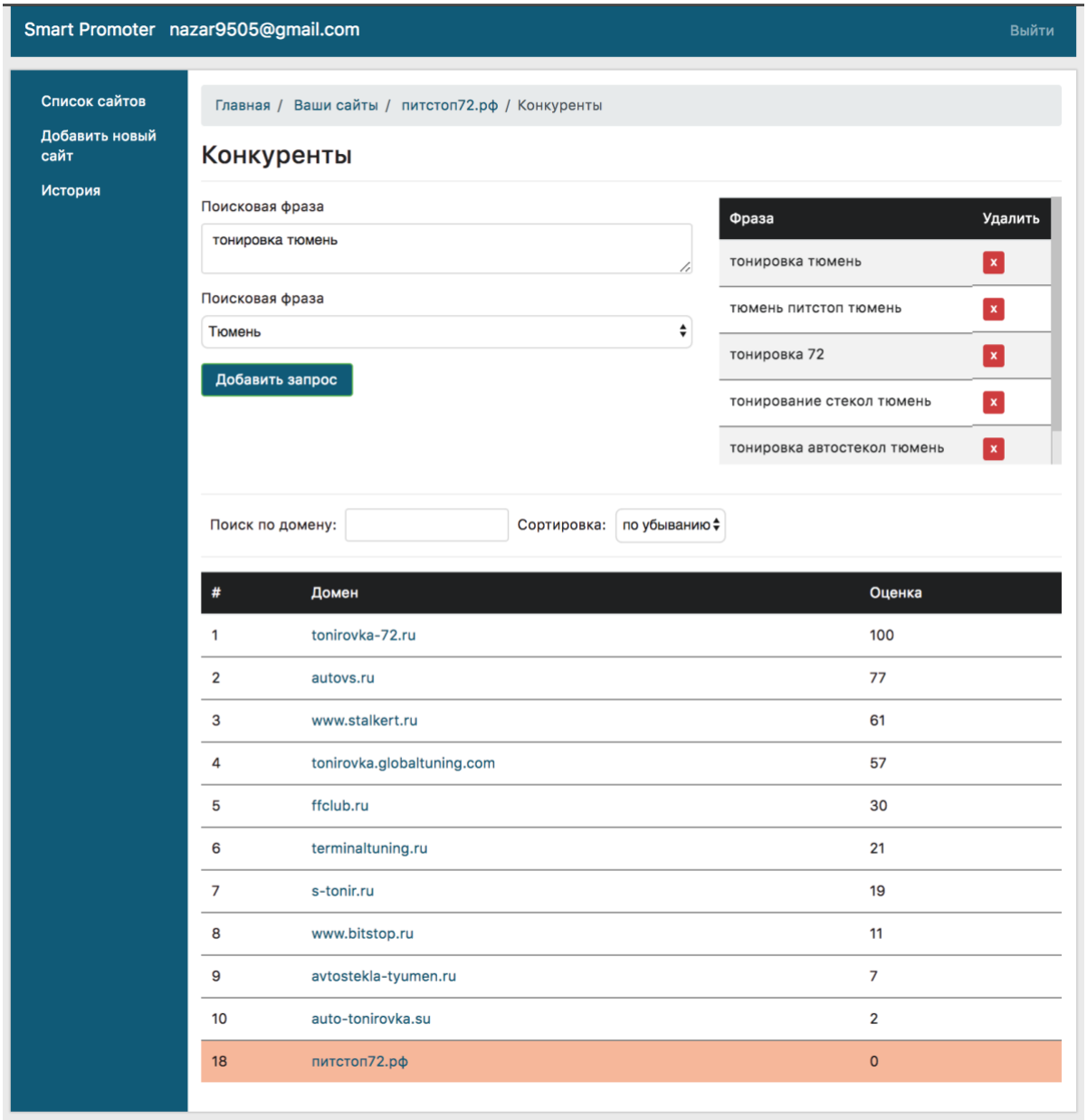

Рисунок 11. Страница с общей статистики

7. Страница с диаграммой по конкретному запросу (см. Рис. 12) позволяет просматривать информацию по продвижению страницы, основываясь на факторах ранжирования. На данной странице строится диаграмма, определяющая продвигаемый сайт среди конкурентов.

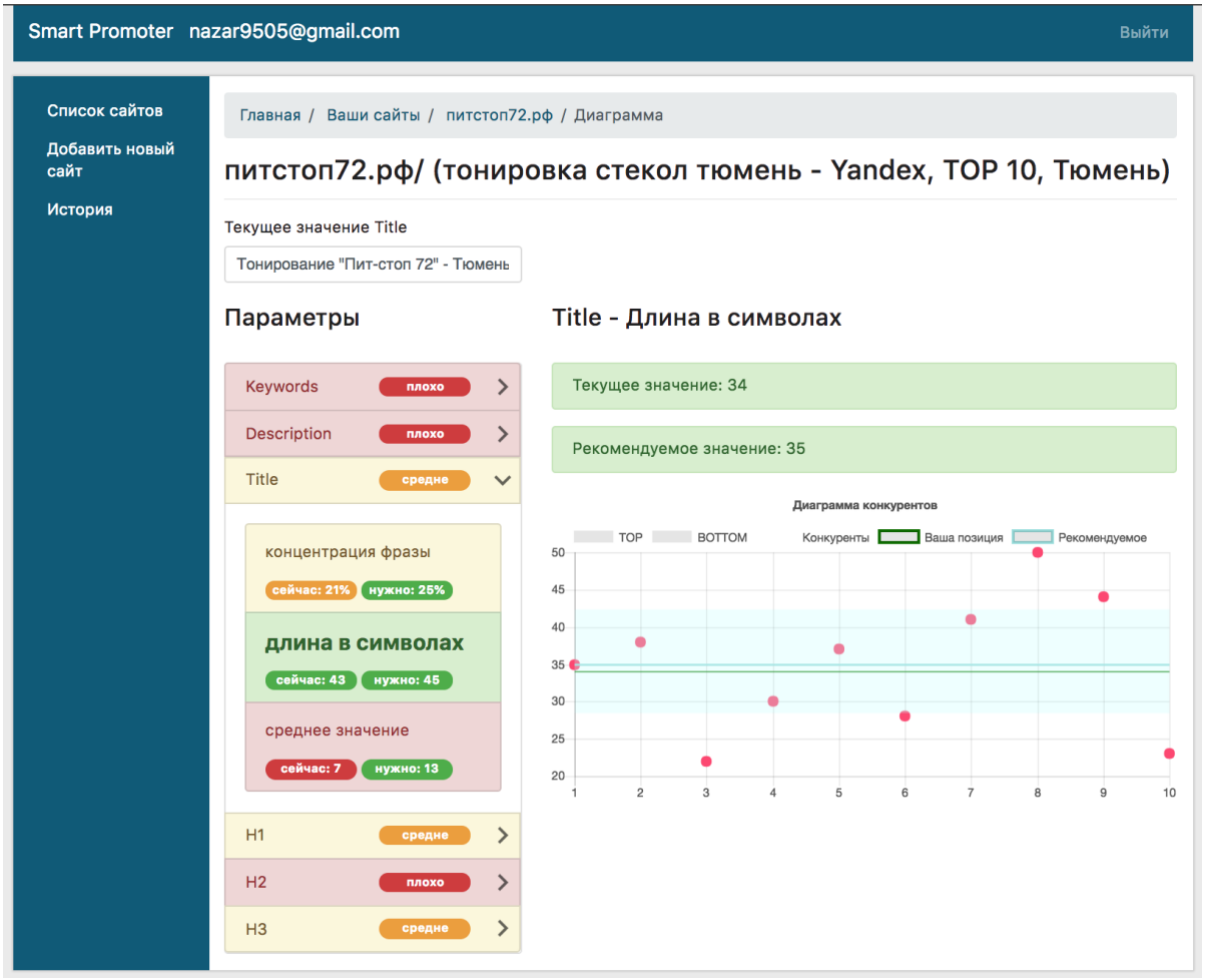

Рисунок 12. Страница с диаграммой по конкретному запросу

В роли task-менеджера в пределах клиентской части рассматривались такие инструменты, как:

- Grunt;
- Gulp.

В качестве менеджера, используемого в проекте, был выбран инструмент задач Gulp. Преимущество Gulp над Grunt в том, что Gulp не требует создания временных файлов между процессами. Файл, после выполнения текущего процесса, сразу передается следующему процессу, не тратя время на сохранение файла.

Gulp использует Orchestrator, который помогает запускать процессы параллельно, ускоряя тем самым работу с файловой системой.

Gulp - интерфейс для подключения и настройки различных плагинов, которые выполняют определенные операции с файлами:

- преобразование файлов (сжатие картинок, минификация js и css, объединение в один файл);
- слежение за изменениями файлов и реакция на эти изменения;
- автоматизированное тестирование;
- автоматическое генерирование документации;
- компиляция с одного языка на другой:

LESS -> CSS, TypeScript -> JavaScript, ES6 -> ES5.

В контексте данного проекта используются плагины, представленные в таблице .

Таблица 2. Плагины, используемые в Gulp

Наименование	**Описание**
gulp-sass	компиляция файлов scss
gulp-autoprefixer	добавление префиксов под разные браузеры
gulp-livereload	автоматическое обновление страницы в браузере при изменениях
gulp-clean-css	минификация css
gulp-jsmin	минификация js
gulp-changed	слежение только за измененными файлами
gulp-concat	склеивание файлов
gulp-rename	переименование файлов
gulp-connect	локальный веб-сервер
gulp-watch	перезапуск задач при наличии измененных файлов

Наименование	Описание
gulp-clean	удаление файлов и директорий
gulp-file-include	HTML-шаблонизатор
gulp.spritesmith	генерация спрайтов
gulp-debug	отладочная информация по отслеживаемым файлам
gulp-if	использования условий в задачах
gulp-sourcemaps	генерация структуры файлов, понятной браузеру для отладки
webpack-stream	разновидность Webpack для gulp

В ходе разработки был написан конфигурационный файл, который агрегирует следующие задачи:

- компиляция scss в css, минификация, склейка, построение sourcemaps;
- компиляция ES6 в ES5, построение sourcemaps, минификация и склейка (с помощью webpack);
- шаблонизация html;
- Перенос шрифтов, видео в директорию сборки;
- Оптимизация изображений, построение спрайтов;
- Преобразование svg в единый файл;
- Запуск локального сервера, поддерживающий livereload.

Шаблон для сборки клиентской части был вынесен в отдельный репозиторий на Github под свободной лицензией (MIT).

В разрабатываемом приложении используется соглашение по именованию CSS-селекторов по методологии БЭМ (Блок-Элемент-Модификатор), разработанной и поддерживаемой компанией Яндекс. БЭМ

позволяет разрабатывать интерфейс на уровне компонентов, тем самым предоставляя возможность избежать дублированная кода.

Блок - автономная простая или составная единица интерфейса, которая может быть пере использована в любом месте страницы неограниченное количество раз.

Элемент – составная контекстно-зависимая часть блока, которая должна находиться только внутри блока.

Модификатор – свойство блока/элемента, описывающее изменения в поведении или внешнем виде.

Элементы отделяются от блоков двумя подчеркиваниями __. Модификаторы отделяются от элементов двумя дефисами --.

В будущем предполагается перенос фронтенда на фреймворк Vue.js/React.js.

Плюс использования frontend фреймворка, реализующего VirtualDOM [14], в том, что рендеринг HTML происходит на клиенте независимо от backend, получая данные посредством API. Генерировать HTML на стороне сервера и передавать по сети – дорогостоящая задача [15].

4.2. Серверная часть

Серверная часть отвечает за обработку данных на стороне сервера и доступ к базе данных. Производительность сервера зависит от провайдера, предоставляющего услуги хостинга.

В роли платформы был выбран Node.js. Основным преимуществом перед другими платформами является событийный, неблокирующий поток ввода/вывода, реализованный на движке JavaScript с открытым исходным кодом V8, разрабатываемом компанией Google.

В качестве фреймворка рассматривались следующие проекты:

- Express.js
- Loopback
- Sails

Первоначально сервис разрабатывался на express.js, но пришлось отказаться от него по ряду причин:

- Большинство функционала пришлось бы разрабатывать вручную, другие же фреймворки предоставляют все «из коробки».
- Отсутствие своей ORM, из-за чего требовалось выбрать стороннюю библиотеку (выбор пал на Sequelize).
- Отсутствие модуля авторизации/аутентификации, из-за чего требовалось выбрать сторонний (выбор пал на Passport.js).

Loopback в отличие от Sails хорошо подходит для разработки API, как отдельного сервиса, в связи с этим он был выбран в качестве используемого фреймворка [11].

Особенности фреймворка Loopback:

- Встроенный API-интерфейс и документация API для работы со Swagger;
- Встроенные элементы управления доступом (ACL);
- Встроенная ORM/ODM с различными драйверами;
- Подробная документация;
- Реализация Best Practice;
- Встроенная библиотека загрузки файлов;
- Отличная масштабируемость;
- Возможность разработки микросервисов;
- Документированный код ядра.

Особенности фреймворка Sails:

- Обеспечивает хорошую организацию кода;
- Встроенная поддержка WebSockets;
- Поддержка различных баз данных;
- Валидация данных;
- Генерация кода для контроллеров, моделей и маршрутов;
- Множество готовых функций безопасности, например, CSRF и совместимость с Lusca;

- Встроенная библиотека загрузки файлов;
- Хорошая документация;
- Встроенная поддержка Redis;
- Гибкая и модульная архитектура с хуками и плагинам.

Для работы с Яндекс.XML была разработана библиотека Yandex-xml [9], которая была опубликована в публичном хранилище пакетов npm.js (см. Приложение). Программный код разработанной библиотеки Yandex-xml приведен в Листинге 3 (см. Приложение). Данная библиотека написана на ES6, в связи с этим может быть подключена в проект на node.js, либо после компиляции в ES5 через webpack/babel в клиентский скрипт. Работает через async/await.

Yandex-xml может быть установленная в любой проект с помощью пакетных менеджеров npm/yarn. В качестве зависимостей использует request-promise и xml2js.

Формат возвращаемых данных – массив json-объектов, содержащих следующие поля:

- url;
- domain;
- title.

С помощью библиотеки request-promise по полю url запрашивается html-код страницы. С помощью библиотеки cheerio можно работать с версткой на уровне объектов.

Библиотека Yandex-xml поддерживает конфигурирование с помощью опций, описанных в официальной документации Яндекс.XML.

В качестве системы контроля версиями используется инструмент Git, позволяющий фиксировать любые изменения проекта и хранить для случаев, при которых может возникнуть необходимость возврата к более ранним версиям. В качестве хостинга удаленного репозитория выбран BitBucket из-за возможности хранения проекта в бесплатном закрытом репозитории.

В будущем планируется перейти на GitLab в связи с тем, что данное хранилище является OpenSource-проектом и может быть развернуто на локальном сервере.

4.3. Развертывание системы

Проект разворачивается в докере на VDS сервере. Преимущество использования Docker в том, что данный инструмент позволяет инкапсулировать окружение в виде отдельного процесса посредством виртуализации [16].

Также Docker позволяет быстро создать легковесный образ приложения и опубликовать в удаленный реестр образов DockerHub, что позволит быстро и легко разворачивать приложение при наличии Интернета.

Использование docker-образов в контексте разработки сервиса «Smart Promoter» хорошо вписывается в рамки такой концепции развертывания приложения, как CI/CD (непрерывная интеграция и доставка). CI/CD предоставляет возможность

VDS сервер (виртуальный) имеет превосходство перед физическим сервером тем, что предоставляет максимальные возможности для настройки. С одной стороны, такой подход занимает больше времени, но является более гибким. VDS сервер поставляется с минимальным набором ПО – обычно это пустая операционная система GNU/Linux. Также такой сервер требует гораздо меньше финансовых затрат и позволяет широко сконфигурировать аппаратное обеспечение.

4.4. Результаты тестирования работы сервиса

Для проверки работоспособности сервиса по продвижению «Smart Promoter» был протестирован сайт питстоп72.рф, добавленный на индексацию в поисковую систему через сервис Яндекс.Вебмастер и находящийся первоначально на 44 позиции на дату 02.06.2018.

В ходе проверки функционала сервиса был составлен и проанализирован список сайтов конкурентов, полученный из поисковой выдачи по запрос «тонировка стекол Тюмень», на основе которого выдавались рекомендации для последующего их применения (см. Табл. 2).

Таблица 2. Плагины, используемые в Gulp

Домен	**Описание**
tonirovka-72.ru	Тонировка автомобиля – тонировка стекол атермальной пленкой в Тюмени
http://tyumen.terminaltuning.ru/tonirovka	Тонировка стекол авто в Тюмени, атермальная
http://kuzov72.ru/toning.htm	Тонировка стекол в Тюмени. Выгодные цены, высокое качество.
http://www.avtobest72.ru/tonirovka	тонировка стекол автомобиля по госту недорого в Тюмени
https://www.tonirovka72-tyumen.ru	Тонировка-72 в Тюмени и Тюменской области
Центр тонирования стекол	Центр тонирования стекол
http://автостекла-тюмень.рф	Автостекла Тюмень в Тюмени Тонировка
http://tmn.ultravisionfilm.ru	Тонировочная пленка для авто в Тюмени
http://www.bitstop.ru/contacts/tyumen	Ремонт, тонировка автостекла в Тюмени
http://s-tonir.ru/index.php/goroda-i-regiony/163-tyumen	Тюмень - Съемная силиконовая статическая тонировка

Для тестирования сайта «питстоп72.рф» выбраны 3 фактора и 3 критерия по каждому фактору.

Факторы:

- Title;
- Description;
- Keywords.

Критерии:

- Концентрация фразы;
- Длина в символах;
- Частота букв.

Далее один фактор приводится к рекомендуемым значениям, после чего по истечении недели аналогичная операция производится над другим фактором.

Результаты вычислений значений по фактору «Title» представлены в таблице 3. По итогу приведения значений факторов к рекомендуемым позиция сайта изменилась с 44 (4 страница поисковой выдачи) на 37 (3 страница поисковой выдачи).

Таблица 3. Числовые коэффициенты 3 критериев по фактору «Title» сайта питстоп72.рф

	Концентрация фразы		Длина в символах		Частота букв	
	Текущее	Нужное	Текущее	Нужное	Текущее	Нужное
До (02.06.2018)	12.5	32	57	45	7.12	9
После (10.06.2018)	37.5	34	43	45	8.5	9

Результаты вычислений значений по фактору «Description» представлены в таблице 4. Позиция сайта изменилась с 37 (3 страница поисковой выдачи) на 32 (3 страница поисковой выдачи).

Таблица 4. Числовые коэффициенты 3 критериев по фактору «Description» сайта питстоп72.рф

	Концентрация фразы		Длина в символах		Частота букв	
	Текущее	Нужное	Текущее	Нужное	Текущее	Нужное
До (1-10.06.2018)	80	46.15	47	121	9.4	9
После (17.06.2018)	56	33.33	125	119	9.4	9

Результаты вычислений значений по фактору «Keywords» и критериям представлены в таблице 5. Позиция изменилась с 32 (3 страница поисковой выдачи) на 29 (2 страница поисковой выдачи).

Таблица 5. Числовые коэффициенты 3 критериев по фактору «Keywords» сайта питстоп72.рф

	Концентрация фразы		Длина в символах		Частота букв	
	Текущее	Нужное	Текущее	Нужное	Текущее	Нужное
До (17.06.2018)	55.56	30	78	128	8.17	8
После (25.06.2018)	41.67	30	107	128	8.92	8

Изменение позиции сайта и числовые значения критериев анализируемых факторов с течением времени записываются в журнал логов. Данная особенность функционала в будущем позволит отслеживать динамику изменения значений факторов продвигаемого сайта относительно позиции.

На основе проведенного тестирования можно сделать вывод, что изменение фактора по одному критерию прямо пропорционально влияет на значение других критериев.

В связи с этим приведение всех критериев одновременно к нужным и точным значениям является довольно сложной задачей.

Оптимальным будет являться значение, не превышающее 5-10% от рекомендуемого.

Также в результате тестирования выявлено, что приведение критериев фактора к рекомендуемому напрямую влияет на позицию сайта.

Стоит учесть, что сайты конкурентов также предпринимают меры по продвижению. Для лучшего результата при использовании сервиса требуется учесть все факторы по всем критериям и поддерживать их в актуальном состоянии раз в 1-2 недели.

ЗАКЛЮЧЕНИЕ

В ходе написания магистерской работы был реализован сервис «Smart Promoter» для SEO-продвижения веб-сайтов на основе анализа конкурентов и выдачи рекомендаций.

Были выполнены следующие задачи:

- изучены особенности ранжирования в поисковых системах, а именно системы Яндекс;
- рассмотрены и проанализированы существующие решения;
- изучены методы и алгоритмы для определения степени значимости факторов ранжирования, и выбран метод для использования в рамках разработки сервиса.
- реализован алгоритм ранжирования сайта среди конкурентов;
- реализован алгоритм выдачи рекомендаций;
- спроектировано разрабатываемое приложение;
- выбрана архитектура и технологии;
- реализовано приложение для автоматизированной выдачи рекомендаций для продвижения веб-сайта;
- проведено тестирование разработанного сервиса «Smart Promoter».

Библиотека Yandex-XML [9] и Шаблон для frontend-разработки [10] были разработаны и опубликованы в публичном хранилище репозиториев разработчиков, и интегрированы в сервис «Smart Promoter».

В будущем планируется доработать алгоритмы и методы анализа сайтов конкурентов, расширить список значимых факторов и критериев, автоматизировать процесс подгонки значений под рекомендации, добавить механизм проверки сайта на валидность HTML и историю продвижения для отображения динамики изменения позиции сайта относительно времени для последующего информирования пользователя.

СПИСОК ЛИТЕРАТУРЫ

1. Поисковая оптимизация. Электронный ресурс URL: https://ru.wikipedia.org/wiki/Поисковая_оптимизация (дата обращения 17.06.2017).
2. Google Ranking Factors. Электронный ресурс URL: https://www.hobo-web.co.uk/google-ranking-factors-quality-rating-criteria/
3. HTML Validation of Context-Free Languages / Anders Moller, Mathias Schwarz // International Conference on Foundations of Software Science and Computational Structures – 2011 (дата обращения 26.04.2018).
4. Метод ранжирования. Электронный ресурс URL: http://studme.org/53270/ekonomika/metod_ranzhirovaniya (дата обращения 18.06. 2017).
5. Метод попарного сравнения. Электронный ресурс URL: http://www.refsru.com/referat-23194-3.html (дата обращения 18.06.2017).
6. Преимущества трехуровневой архитектуры клиент-сервер / Дочкин А.С. // Научно-технический прогресс: актуальные и перспективные направления будущего – 2016, С. 41-42. (дата обращения 01.07.2017).
7. Значение SEO для эффективных продаж в интернете / К. И. Фаустова // Территория науки, 2015, № 3, С. 139 – 144. (дата обращения 14.06.2017).
8. Сравнительный анализ методов определения весовых коэффициентов «влияющих факторов» / Коробов В.Б. // Социология: методология, методы, математическое моделирование, 2005. № 20. С. 54-73. (дата обращения 10.07.2017).
9. Угриновский Н.В. Библиотека Yandex-XML. Электронный ресурс URL: https://www.npmjs.com/package/yandex-xml.
10. Угриновский Н.В. Шаблон для frontend-разработки. Электронный ресурс URL: https://github.com/fredure/empty_frontend_project_structure.
11. Contrasting Enterprise Node.js Frameworks: Hapi vs. Kraken vs. Sails.js vs. Loopback. Электронный ресурс URL: https://medium.com/capital-one-

developers/contrasting-enterprise-node-js-frameworks-hapi-vs-kraken-vs-sails-js-vs-loopback-51c6712cc3c7 (дата обращения 3.12.2017).

12. Моделирование архитектуры предприятия с archi. / Точилкина Т.Е. // Экономика и менеджмент инновационных технологий. 2014. (дата обращения 29.12.2017).
13. Модель суммарной оценки сайта в сети Интернет на основе факторов ранжирования / Пестеров П.В., Янишевская А.Г. // Инженерный вестник Дона – 2015. (дата обращения 24.01.2018).
14. Building User Interfaces Using Virtual DOM / Marianne Grov // UIO Department of informatics – 2015. (дата обращения 30.03.2018).
15. Virtual DOM coverage for effective testing of dynamic web applications / Yunxiao Zou, Zhenyu Chen // ResearchGate – 2014. (дата обращения 30.03.2018).
16. Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research / Vincenzo Ferme // ResearchGate – 2016. (дата обращения 15.04.2018).
17. 130 факторов ранжирования. Электронный ресурс URL: http://convertmonster.ru/blog/seo-blog/osnovnye-principy-ranzhirovanija-sajta-v-poiskovyh-sistemah/ (дата обращения: 19.02.2018)
18. Оптимизация алгоритмов ранжирования методами машинного обучения / Гулин Андрей, Карпович Павел, Расковалов Денис, Сегалович Илья // Яндекс на РОМИП, 2009. (дата обращения 02.05.2018).
19. Система семантической оптимизации содержимого веб-сайтов на основе пользовательских предпочтений / П. И. Банокин, В. Н. Вичугов // Известия Томского политехнического университета. Инжиниринг георесурсов, 2012, - Т. 321, № 5: Управление, вычислительная техника и информатика. — С. 93-97. (дата обращения 25.01.2018).
20. Understanding DevOps & bridging the gap from continuous integration to continuous delivery / Manish Virmani // Fifth International Conference on

the Innovative Computing Technology – 2015. (дата обращения 15.04.2018).

21. A comparison of software cost, duration, and quality for waterfall vs. iterative and incremental development: A systematic review / Susan M. Mitchell, Carolyn B. Seaman, 2012. (дата обращения 09.10.2017).
22. Az.js: Javascript-библиотека для обработки текстов на русском языке. Электронный ресурс URL: https://habr.com/post/303308/ (дата обращения: 02.11.2017).

ПРИЛОЖЕНИЯ

Листинг №1. Файловая структура разработки клиентской части приложения.

```
project/
+--assets/                    # файлы проекта до сборки
    +--img/                    # изображения до сжатия
    +--video/                  # видео
    +--fonts/                  # шрифты
    +--sass/                   # sass
    ¦  +--vendor/              # библиотеки, reset.css
    ¦  +--components/          # блоки sass, библиотеки
    ¦  +--mixins.scss          # миксины sass
    ¦  +--variables.scss       # переменные sass
    ¦  +--main.scss            # @include components, vendor
    +--js/                     # js
    ¦  +--components/          # плагины
    ¦  +--main.js
    +--svg/
    +--pages/                  # html
    ¦  +--/blocks              # partial блоки (header, footer, etc.)
    ¦     +--demomenu.html     # демо меню со всеми страеницами и uikit
    ¦  +--index.html           # html
    ¦  +--...
+--build/                     # файлы проекта после сборки
    +--css/
    ¦  +--styles.min.css       # min css
    +--js/
    ¦  +--custom.min.js        # min js
    +--img/
    ¦  +--svg/                 # sprite svg
    +--video/                  # видео
    +--fonts/                  # шрифты
    +--index.html              # index.html
    +--...                     # тут же остальные html файлы после сборки
+--gulpfile.js
+--package.json
+--.gitignore                 # ignore node_modules | ignore build
```

Листинг №2. Построение диаграммы конкурентов с коридором рекомендуемого значения фактора

```
function genArr(value) {
      return new Array(10).fill(value);
}

function average(arr) {
      return arr.reduce((p, c) => p + c, 0) / arr.length;
}

function minCorr(avg, arr){
      var min = Math.min(...arr);
      return avg - (Math.abs(avg - min) / power);
}

function maxCorr(avg, arr){
      var max = Math.max(...arr);
      return avg + (Math.abs(avg - max) / power);
}

var avgValue = average(criterions);
var minCorrValue = minCorr(avgValue, criterions);
var maxCorrValue = maxCorr(avgValue, criterions);

var need = Math.round(minCorrValue + ((maxCorrValue  - minCorrValue) / 2));


var chartData = {
    labels: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    datasets: [
      {
        label: "TOP",
            radius: 0,
            data: genArr(maxCorrValue),
        borderColor: "transparent",
        fill: false,
        steppedLine: false,
        tension: 0,
            fillBetweenSet: 1,
            borderWidth: 0.5,
        fillBetweenColor: "rgba(208, 255, 255, 0.3)"
    },
    {
      radius: 0,
        label: "BOTTOM",
            data: genArr(minCorrValue),
        borderColor: "transparent",
        borderWidth: 0.5,
        fill: false,
        steppedLine: false,
        tension: 0.5
    },
    {
        label: 'Конкуренты',
        data: criterions,
        backgroundColor: 'rgba(255, 99, 132, 0)',
        borderColor: 'rgba(255,99,132,1)',
            borderWidth: 0,
            showLine: false,
```

```
            pointBackgroundColor: 'rgba(255,99,132,1)',
            pointRadius: 5,
            pointHoverBackgroundColor: 'rgba(255,99,132,1)',
        fill: true,
      },
      {
            label: 'Ваша позиция',
            borderColor: 'green'
      },
      {
            label: 'Рекомендуемое',
            borderColor: '#aae4e4'
      }]
};

var chart = new Chart($('#chart').get(0), {
      type: 'line',
      data: chartData,
      options: chartOptions,
});
```

Листинг №3. Библиотека для работы с Yandex.XML.

```
// Copyright (c) 2018 Nazar Ugrinovsky
const
  xml2js = require('xml2js'),
  rp = require('request-promise');
class YandexXML {
  /**
    * Constructor
    * @constructor
    * @param  {String} user
    * @param  {String} key
    */
  constructor(user, key){
    this.user = user;
    this.key = key;
    this.URL = "https://yandex.com/search/xml";
  }
  /**
    * Search
    * @param  {Object} options
    * @return {Array<Object>} results
    */
  async search(options) {
    try {
      const xml = await this.request(options);
      const res = await this.beautify(xml);
      return res;
    } catch (error) {
        throw new Error(error.message);
    }
  }
  /**
    * Http request to Yandex
    * @param  {Object} options
    * @return {String} https answer
    */
  async request(options) {
    try {
      const params = {
        uri: this.URL,
        qs: {
          user: this.user,
          key: this.key,
          ...options
        }
      };
      return await rp(params);
    } catch (error) {
      throw new Error(error.message);
    }
  }
  /**
    * Getting the necessary fields
    * @param  {String} xml
    * @return {Array<Object>} beautify results
    */
  async beautify(xml) {
    try {
      const res = await this.convert(xml);

      if (
```

```
        !res.yandexsearch ||
        !res.yandexsearch.response ||
        !res.yandexsearch.response[0]
      ) {
        throw new Error("Invalid response of Yandex");
      }
      if (res.yandexsearch.response[0].error) {
        throw new Error(res.yandexsearch.response[0].error[0]);
      }
      if                  (!res.yandexsearch.response[0].results                  ||
!res.yandexsearch.response[0].results[0]) {
          throw new Error("No results");
      }
      if (
        !res.yandexsearch.response[0].results[0].grouping ||
        !res.yandexsearch.response[0].results[0].grouping[0] ||
        !res.yandexsearch.response[0].results[0].grouping[0].group
      ) {
          throw new Error("Invalid answer");
      }
      const groups = res.yandexsearch.response[0].results[0].grouping[0].group;
      const elements = [];
      groups.forEach((el, i, arr) => {
        const doc = el.doc[0];
        const element = {
          title: doc.title[0]._,
          domain: doc.domain[0],
          url: doc.url[0],
          headline: (doc.headline) ? doc.headline[0]._ : ''
        };
        elements.push(element);
      });
      return elements;
    } catch (error) {
        throw new Error(error.message);
    }
  }
  /**
    * Convert xml string to js object
    * @param  {String} xml
    * @return {Object} converted xml to js object
    */
  convert(xml) {
    return new Promise((resolve, reject) => {
      xml2js.parseString(xml, (err, res) => {
        if (!err) {
            resolve(res);
        } else {
            reject(err);
        }
      });
    });
  }
}
module.exports = YandexXML;
```

Профиль Угриновского Н.В. на github.com, отображающий разработанные библиотеки