

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

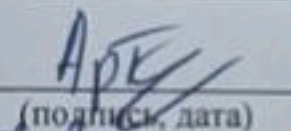
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

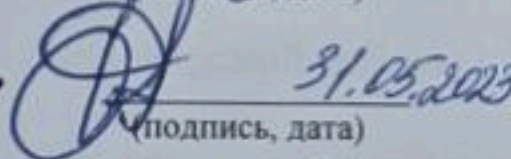
ПРОГРАММНЫЙ АНАЛИЗАТОР
ИНТЕРНЕТ-РЕСУРСОВ

Студент


(подпись, дата)

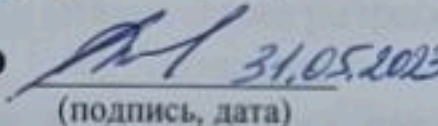
Артемов Кирилл Артемович

Руководитель


(подпись, дата)

Федюнин Р.Н.

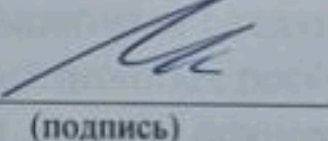
Нормоконтролёр


(подпись, дата)

Бычков А.С.

Работа допущена к защите (протокол зас. кафедры от 02.06.23 г. № 14)

Заведующий кафедрой

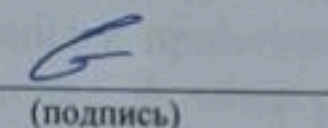

(подпись)

Митрохин М.А.

Работа защищена с отметкой отлично (протокол заседания ГЭК от

06.06.2023г. № 2)

Секретарь ГЭК


(подпись)

Гурин Е.И.

Пенза 2023

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«26» апреля 2023г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Артемов Кирилл Артемович гр. 19ВВВ1 факультета вычислительной техники.

2. Тема работы «Программный анализатор интернет-ресурсов»

Тема утверждена приказом ПГУ №458/о от "26" апреля 2023 г.

3. Руководитель бакалаврской работы к.т.н., доцент Федюнин Р.Н.

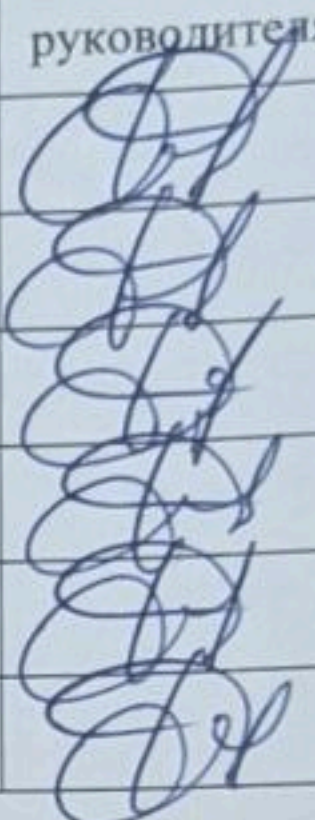
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать приложение «Программный анализатор интернет-ресурсов», выполняющее сбор и компоновку данных о интернет-ресурсах с последующим анализом входных данных. Приложение должно быть спроектировано с применением клиент-серверной архитектуры и состоять из трех связанных между собой частей: клиентская часть, серверная часть, база данных. Клиентская часть программы должна обладать интуитивным графическим интерфейсом. Серверная часть должна включать следующий функционал: сбор, обработку и анализ информации, собранной с информационных ресурсов в сети интернет. Взаимодействие между серверной и клиентской частями должно осуществляться с помощью *REST API*. База данных должна обладать инструментарием для работы с серверной частью, обладать возможностью для работы по системе *CRUD*.

5. Перечень подлежащих разработке вопросов

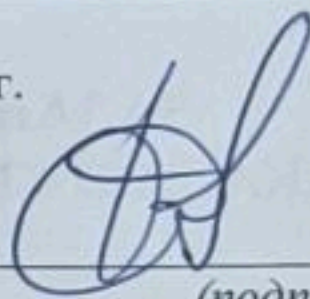
Требуется произвести анализ предметной области, изучить аналоги существующих анализаторов интернет-ресурсов, способы сбора данных, изучить способы проектирования и разработки клиент-серверного приложения с применением *REST API* методологии и *MVC* паттерна. Необходимо изучить функционал фреймворка *Laravel* и *JS*-библиотеки *React*, ознакомиться с разворачиванием и настройкой *VPS*.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	15%	26.04.2023-30.04.2023	
2	Анализ требований	4%	01.05.2023-02.05.2023	
3	Выбор технологий, применяемых при проектировании и разработке	15%	03.05.2023-07.05.2023	
4	Описание алгоритма	13%	08.05.2023-11.05.2023	
5	Разработка приложения «Программный анализатор интернет-ресурсов»	45%	12.05.2023-29.05.2023	
6	Оформление пояснительной записки	8%	30.05.2023-31.05.2023	

Дата выдачи задания "26" апреля 2023 г.

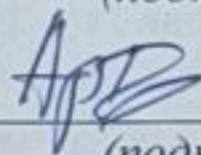
Руководитель бакалаврской работы



Федюнин Р.Н.

(подпись, дата)

Задание к исполнению принял студент



Артемов К.А.

(подпись, дата)

Работу к защите допустить
Декан факультета



02.06.2023

Фионова Л.Р.

(подпись, дата)

Реферат

Бакалаврская работа содержит 92 листа, 31 рисунок, 4 таблицы, 8 источников и 4 приложения.

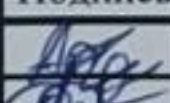
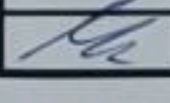
ПРОГРАММНЫЙ АНАЛИЗАТОР ИНТЕРНЕТ-РЕСУРСОВ, REST API, СУБД.

Цель работы – разработать систему, собирающую и структурирующую информацию для анализа интернет-ресурсов.

Объектами разработки являются:

- серверное приложение для сбора и сохранения данных о интернет-ресурсах в СУБД, с последующим их анализом;
- клиентское приложение для отображения результатов анализа данных;
- сущности и связи базы данных для хранения результатов анализа.

В результате проектирования: разработано приложение, полностью соответствующее поставленной задаче, а именно: разработана серверная часть приложения, для сбора и компоновки данных о интернет-ресурсах, разработана база данных, реализовано API для взаимодействия серверной и клиентской части приложения, создана клиентская часть приложения с графическим интерфейсом для и отображения результатов.

					ПГУ1.09.03.01.01.001 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	Программный анализатор интернет-ресурсов Пояснительная записка	Лит.	Лист	Листов
Разраб.		Артемов К.А.					2	
Провер.		Федюнин Р.Н.						
Н. Контр.		Бычков А.С.						
Утверд.		Митрохин М.А.						
					ФВТ, каф. ВТ гр.19BBB1			

Содержание

Перечень сокращений и обозначений.....	5
Введение.....	6
1 Программный анализатор интернет-ресурсов	8
1.1 Предпроектный анализ.....	8
1.1.1 Актуальность.....	8
1.1.2 Пользователи и сценарии.....	9
1.1.3 Анализ аналогов.....	11
1.1.4 Выбор языка программирования	19
1.1.5 Выбор фреймворка для серверной части приложения	21
1.1.6 Хранение данных	24
1.1.7 Выбор СУБД	25
1.1.9 Выбор библиотеки для реализации графического интерфейса	31
1.1.10 Функциональные требования	32
1.1.11 Выводы по предпроектному анализу	33
Выводы по предпроектному анализу:	33
1.2 Проектирование и разработка	33
1.2.1 Реализация <i>VPS</i> сервера	34
1.2.2 Разработка серверной части приложения	36
1.2.3 Разработка БД	40
1.2.4 Реализация основного метода сбора и структурирования информации.....	43
1.2.5 Разработка клиентской части приложения	46
1.3 Анализ результатов.....	48
1.3.1 Тестирование <i>API</i>	48

1.3.2 Тестирование интерфейса.....	51
2 Охрана труда.....	57
2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ.....	57
2.2 Классификация и категорирование помещения, в котором производится разработка	57
2.2.1 Классификация по взрывопожарной и пожарной опасности	57
2.2.2 Классификация по опасности поражения электрическим током	58
2.3 Санитарно-гигиеническая характеристика производства	60
2.4 Расчет условий зрительной работы в производственном помещении, где установлены ПЭВМ.....	61
Заключение	65
Список используемых источников.....	66
Приложение А – UML – диаграммы	67
Приложение А.1 – Диаграмма развертывания.....	68
Приложение А.2 – Диаграмма компонентов.....	69
Приложение А.3 – Диаграмма последовательности	70
Приложение Б – ЕСПД	71
Приложение Б.1 – Схемы программ	72
Приложение В – Листинг программы.....	73
Приложение Г – Презентация	84

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ВКР – выпускная квалификационная работа

ООП – объектно-ориентированное программирование

Патерн – шаблон проектирования

СУБД – система управления базами данных

ЭВМ – электронно-вычислительная машина

ACID (Atomicity, Consistency, Isolation, Durability) — набор требований к транзакционной системе

API (Application Programming Interface) – программный интерфейс приложения

Backend – серверная часть приложения

CRUD – акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. *create*), чтение (*read*), модификация (*update*), удаление (*delete*).

Frontend – клиентская часть приложения

HTML (Hyper Text Markup Language) – язык гипертекстовой разметки

MVC (Model-View-Controller) – «Модель-Представление-Контроллер» или «Модель-Вид-Контроллер»

ORM (Object-Relational Mapping) – объектно-реляционное отображение, или преобразование

REST (Representational State Transfer) – передача репрезентативного состояния или передача «самоописываемого» состояния

SPA (Single Page Application) – приложение, работающее в браузере без перезагрузки страницы

UML (Unified Modeling Language) – унифицирован язык моделирования.

URL (Uniform Resource Locator) – система унифицированных адресов электронных ресурсов

VPS (Virtual Private Server) – виртуальный выделенный сервер

Введение

Тема программных анализаторов интернет-ресурсов относится к сфере информационных технологий и является важной для SEO-аналитики интернет-ресурсов.

Современный программный анализатор интернет-ресурсов должен обладать следующим функционалом:

- сбор информации с сайтов. Анализаторы позволяют собирать информацию с разных интернет-ресурсов, включая сайты, блоги, форумы, новостные порталы и социальные сети;

- анализ данных. Основным функционалом анализаторов является обработка собранных данных, включая классификацию, категоризацию, фильтрацию и анализ с помощью статистических методов;

- визуализация результатов анализа. Визуализация позволяет представить полученные результаты в удобном и понятном виде, например, в виде графиков или таблиц.

Программный анализатор интернет-ресурсов является инструментом, позволяющим анализировать веб-страницы и получать информацию о них, такую как качество контента, скорость загрузки, ключевые слова и т.д. Он может использоваться в различных сферах жизнедеятельности: маркетинг, политические исследования, научные исследования и прочие. Основными пользователями данного инструмента являются: собственники сайтов, интернет-маркетологи, *SEO*-специалисты, аналитики и разработчики.

Преимуществами использования анализаторов являются: автоматизация процесса анализа, увеличение эффективности работы, повышение точности и качества анализа и ускорение процесса принятия решений на основе полученных данных.

При использовании анализаторов существуют следующие ограничения и недостатки: программа не может охватить все нюансы анализируемого сайта, использование анализатора создает дополнительную угрозу безопасности сайту

клиента. При использовании анализаторов пользователь сталкивается со сложностью настройки критериев анализа в зависимости от качества собираемой информации.

Также важным аспектом является соблюдение законодательства по защите персональных данных и правилам использования информации, полученной с интернет-ресурсов.

Вышеописанное доказывает необходимость разработки данных инструментов с учетом описанных недостатков.

В ВКР реализован программный анализатор сайтов. ВКР состоит из двух основных разделов:

- в первом разделе проведен анализ и выбран инструментарий для разработки, реализована программа, с последующим ее тестированием;
- во втором разделе описана безопасность жизнедеятельности.

1 Программный анализатор интернет-ресурсов

1.1 Предпроектный анализ

1.1.1 Актуальность

Тема "Программный анализатор интернет-ресурсов" обладает высокой актуальностью, так как в настоящее время количество информации в интернете продолжает расти и неуклонно увеличивается [1].

Программные анализаторы интернет-ресурсов являются важным инструментом для работы с данными, которые хранятся на интернет-ресурсах. Ручной анализ таких данных может быть крайне трудоемким и неэффективным. В свою очередь, программные анализаторы могут ускорить этот процесс, повысить точность результатов и значительно уменьшить затраты времени.

Программные анализаторы могут использоваться в различных областях [2], таких как маркетинговые исследования, сбор данных для исследований, разведка конкурентной информации, автоматизированный анализ структуры веб-сайтов и т.д.

Помимо этого, программный анализатор интернет-ресурсов также является важным аспектом в области информационной безопасности, поскольку он помогает выявлять потенциальные уязвимости в системах и защищать данные от внешних угроз.

Кроме того, разработка программного анализатора интернет-ресурсов имеет большой практический интерес для студентов, которые изучают программирование, веб-разработку и базы данных. Эта работа также может привлечь внимание потенциальных работодателей и улучшить шансы на рынке трудоустройства.

Таким образом, программный анализатор интернет-ресурсов является не только актуальной темой в настоящее время, но также может быть востребованным в будущем с точки зрения улучшения работы и повышения эффективности бизнес-процессов. Целью ВКР может быть разработка и реализация программного анализатора интернет-ресурсов, который будет

способен автоматически собирать и анализировать данные, связанные с различными параметрами, такими как посещаемость сайтов, динамика изменения трафика, рейтинги и т.д.

1.1.2 Пользователи и сценарии

Для начала, необходимо определиться, какие конкретно пользователи будут пользоваться программным анализатором интернет-ресурсов и каким образом. Можно предположить, что основными пользователями могут быть следующие группы:

- веб-мастера (*webmasters*) и разработчики веб-сайтов, которые могут использовать программный анализатор для определения, насколько хорошо работает их сайт (включая скорость загрузки страниц, количество ошибок, различные метрики продуктивности и т.д.);

- маркетологи, занимающиеся оптимизацией поисковой выдачи (*SEO*), которые используют программный анализатор, чтобы исследовать ключевые слова и их популярность, а также анализировать метаданные страницы и другие данные, влияющие на ранжирование в поисковой выдаче;

- исполнители проектов, которые могут использовать программный анализатор, чтобы оценить производительность веб-сайтов под своим управлением и предоставлять отчеты о проделанной работе;

- обычные пользователи интернет-ресурсов, которые могут использовать программный анализатор, чтобы определить, насколько быстро загружается определенный веб-сайт, какую информацию он содержит и т.д.

Определение пользователей поможет лучше сформулировать пользовательские сценарии.

Начнем с пользовательских сценариев для каждой группы.

Веб-мастера и разработчики веб-сайтов:

- открытие программного анализатора;
- выбор нужного ресурса для анализа;
- получение сводки об основных проблемах, таких как: задержка загрузки страницы, наличие ошибок в коде и проблемы с изображениями;

– проверка рейтинга SEO оптимизации страницы и предложение рекомендаций для его улучшения.

Маркетологи оптимизации поисковой выдачи:

- открытие программного анализатора;
- ввод ключевых слов и проверка рейтинга SEO оптимизации сайта в поисковых системах;
- получение сводки об основных проблемах, таких как: качество контента, заголовков и мета-описаний;
- предложение рекомендаций для улучшения позиций сайта в поисковой выдаче.

Исполнители проектов:

- открытие программного анализатора;
- анализ сайта для определения основных проблем, таких как: качество контента, показатели SEO оптимизации и удобство пользовательского интерфейса;
- предложение конкретных действий для устранения обнаруженных проблем.

Обычные пользователи интернет-ресурсов:

- открытие программного анализатора;
- ввод URL адреса интересующего сайта для анализа;
- получение сводки об основных проблемах, таких как: задержка загрузки страницы и проблемы с отображением страниц на мобильных устройствах;
- предложение рекомендаций по улучшению показателей сайта для повышения качества пользовательского опыта.

Эти сценарии могут быть дополнены, в зависимости от требований проекта и потребностей конечных пользователей.

Выделим общие сценарии:

- пользователь хочет проанализировать определенный интернет-ресурс и получить отчет по его структуре, содержанию и качеству контента;

- пользователь хочет сравнить несколько интернет-ресурсов на основе их структуры, содержания и качества контента;
- пользователь хочет получить отчет об интернет-ресурсе, который был изменен в определенный период времени;
- пользователь хочет проанализировать интернет-ресурс на наличие определенных тегов или ключевых слов;
- пользователь хочет получить отчет об использовании определенных технологий на интернет-ресурсе;
- пользователь хочет настроить программу на прохождение анализа только определенных частей интернет-ресурса, чтобы получить более точный отчет;
- пользователь хочет найти на интернет-ресурсе конкретный элемент, такой как текстовая информация или изображения, на основе заданных параметров;
- пользователь хочет проанализировать данную страницу или веб-сайт на наличие ошибок *SEO*, чтобы оптимизировать его для поисковых систем;
- пользователь хочет получить отчет о трафике на интернет-ресурсе, в том числе, о количестве посетителей, продолжительности пребывания на странице и среднем времени загрузки страницы;
- пользователь хочет сравнить выбранный интернет-ресурс с его аналогами, чтобы понять, какую популярность имеет данный ресурс.

1.1.3 Анализ аналогов

В рамках ВКР можно провести сравнительный анализ различных программных анализаторов [3] на основе их функционала, удобства использования, точности анализа и стоимости подписок.

В таблице 1 приведено краткое сравнение аналогов.

Таблица 1 – Анализ аналогов

Анализатор	Функционал	Удобство использования	Точность анализа	Стоимость подписок
<i>Pr-cy</i>	Анализ трафика, ссылок, ключей	Простой интерфейс	Достаточно точный	От \$15/месяц
<i>MegaIndex</i>	Анализ конкурентов, рекламы	Простой и прямой	Достаточно точный	От \$12.9/месяц
<i>Majestic</i>	Анализ профиля ссылок	Интуитивный интерфейс	Достаточно точный	От \$49.99/месяц
<i>SimilarWeb</i>	Оценка потенциального трафика	Простой интерфейс	Достаточно точный	От \$199/месяц

PR-CY – это мощный инструмент для анализа поисковой оптимизации и маркетинга. Однако, его функциональность немного отличается от других анализаторов. На рисунке 1 представлена главная страница *PR-CY*.

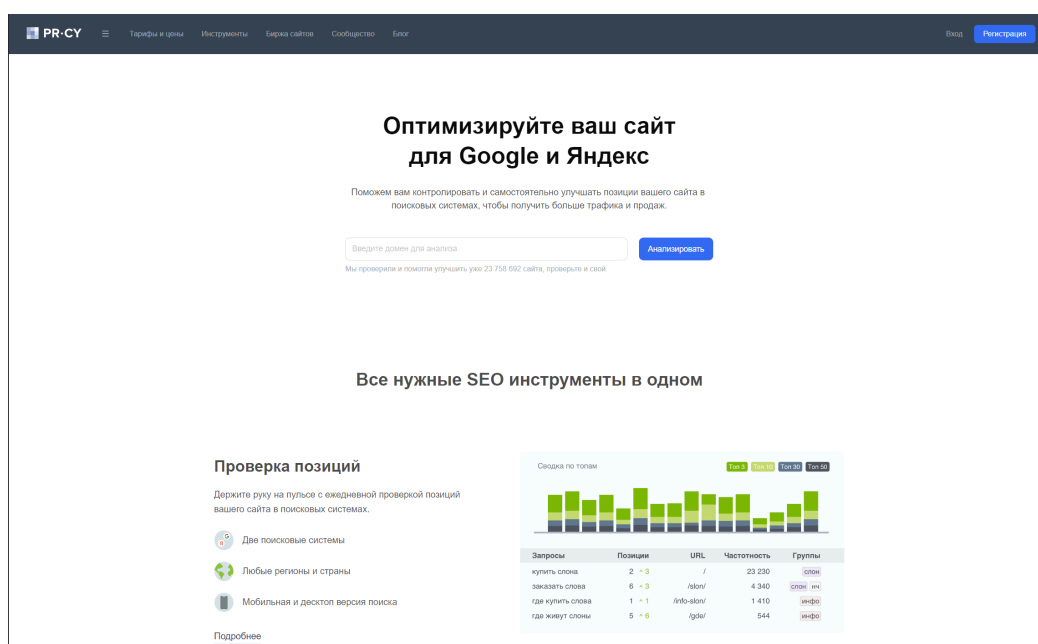


Рисунок 1 – Программный анализатор *PR-CY*

Функциональность. *PR-CY* дает возможность получить информацию о трафике сайта, числе проверенных ссылок, исходных страницах, популярности на блогах и форумах и т. д. А также список ключевых слов, используемых для

продвижения сайта. Однако, инструмент пока не поддерживает анализ социальных сетей.

Удобство использования. Интерфейс *PR-CY* достаточно прост и понятен. Он позволяет быстро найти нужные функции и инструменты.

Точность анализа. Когда речь идет об анализе ссылочного профиля, анализе позиций сайта и ключевых фраз, *PR-CY* может считаться более оптимальным, чем другие аналитические инструменты. Также следует отметить, что анализатор может помочь вам выявить возможные проблемы при SEO оптимизации вашего сайта.

Стоимость подписок. Начальная цена подписки на *PR-CY* составляет 10 000 рублей в месяц.

Преимущества:

- среди конкурентов обладает наиболее точным анализом тИЦ и *PR* сайтов;
- имеет возможность проверки нескольких сайтов одновременно;
- предоставляет рейтинг популярности домена в социальных сетях;
- позволяет определить оптимальную частоту публикации новостей.

Недостатки:

- расчет стоимости услуг поисковой оптимизации является лишь ориентировочным;
- не обладает точными данными по посещаемости сайта.

В целом, *PR-CY* является сильным инструментом для *SEO* анализа и может быть хорошей альтернативой для тех, кто ищет более узкоспециализированные функции для своего бизнеса. Однако, цена на этот инструмент выше, что может не соответствовать бюджету многих пользователей.

MegaIndex – это, также программный анализатор для проверки *SEO* параметров сайта, ключевых слов и конкурентов в интернет-бизнесе. На рисунке 2 представлена главная страница *MegaIndex*.

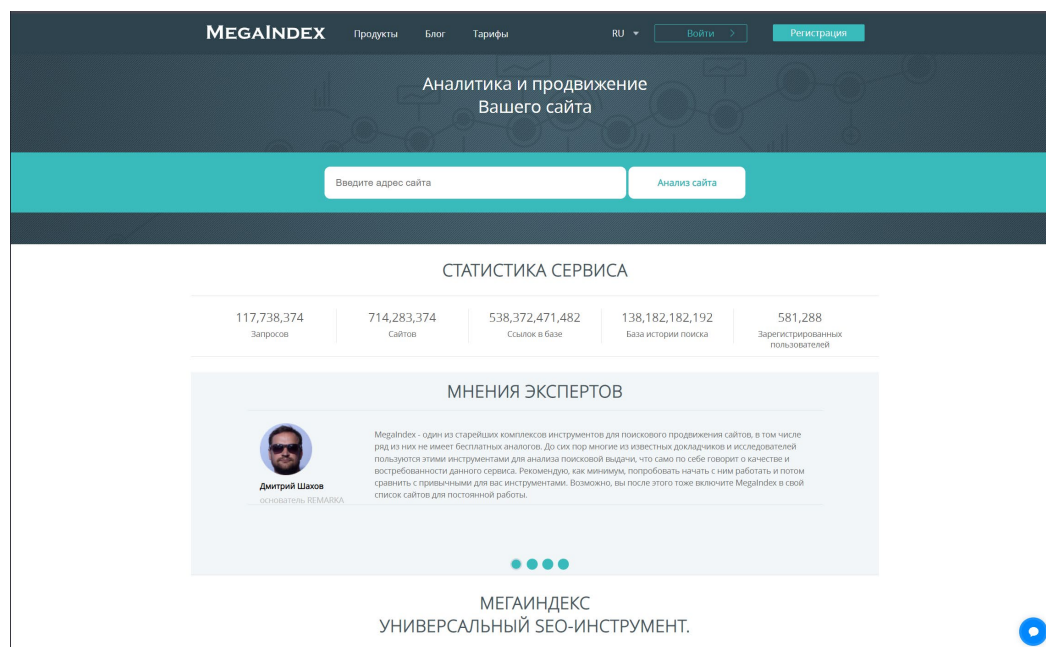


Рисунок 2 – Программный анализатор *MegaIndex*

Сравнение *MegaIndex* и других аналитических инструментов выглядит следующим образом.

Функциональность. *MegaIndex* предоставляет высокую функциональность в сравнении с другими аналитическими инструментами. Он позволяет проводить анализ не только сайта, но и его конкурентов. Основные функции *MegaIndex* включают в себя: анализ сайта, анализ ключевых слов, анализ ссылок и проверка конкурентов.

Удобство использования. Интерфейс *MegaIndex* интуитивно понятен и прост в использовании даже для новичков. Многие функции инструмента легко доступны благодаря наличию меню в верхней части экрана, что позволяет быстро находить нужные инструменты.

Точность анализа. *MegaIndex* может считаться одним из самых точных инструментов, когда речь идет об анализе найденных ключевых слов, мониторинге позиций сайта и сопоставлении сайта с конкурентами.

Стоимость подписок. Цены на подписки на *MegaIndex* относительно низкие и составляют от 899 до 19 999 рублей в месяц.

Преимущества:

– осуществляет мониторинг и анализ всех видов рекламы сайта;

- обладает базой данных в 20 раз больше, чем у конкурентов;
- помогает улучшать *SEO* путем отбора ключевых слов на основании анализа конкурентов;
- способен находить мертвые ссылки, технические ошибки и дубликаты контента.

Недостатки:

- стоимость услуг относительно высока по сравнению с аналогами;
- не всегда обеспечивает корректный анализ сайтов на иностранных языках.

В целом, *MegaIndex* – это мощный аналитический инструмент для *SEO* анализа, который может стать хорошим выбором для оценки сайтов и конкурентов в интернет-бизнесе. Этот инструмент обладает отличной функциональностью и точностью анализа, а его стоимость может быть доступна для большого числа пользователей.

Majestic – это еще один аналитический инструмент для анализа *SEO* параметров сайта, ключевых слов и конкурентов в интернет-бизнесе. На рисунке 3 представлена главная страница *Majestic*.

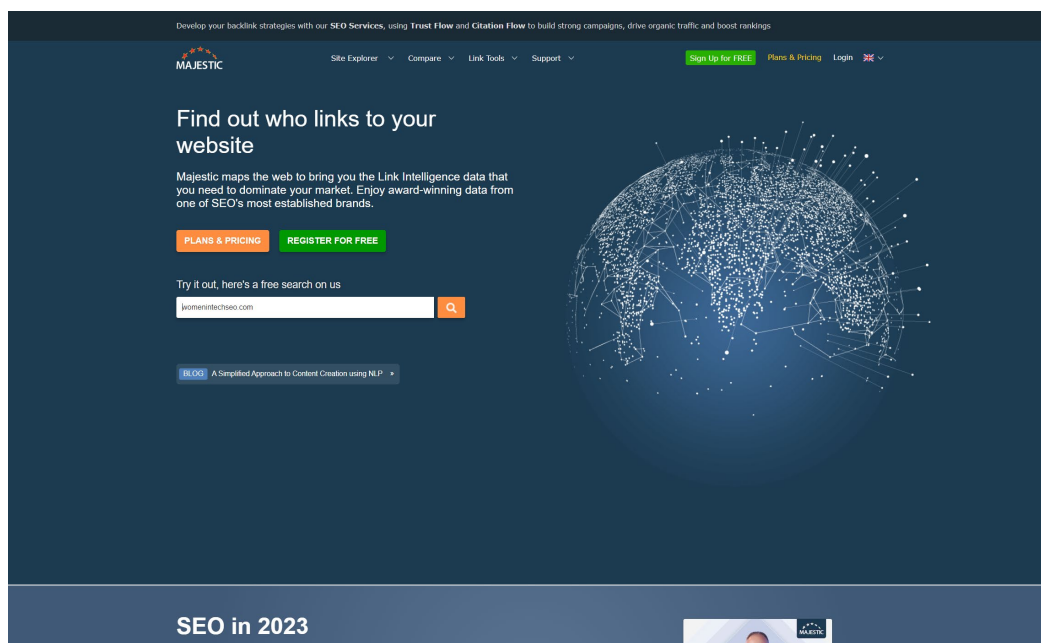


Рисунок 3 – Программный анализатор *Majestic*

Сравнение *Majestic* с остальными выглядит следующим образом.

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

Функциональность. *Majestic* также предоставляет широкий набор функций для анализа *SEO* параметров сайта и его конкурентов. Основные функции *Majestic* включают в себя: анализ ссылочного профиля, анализ содержания страниц, мониторинг позиций сайта, а также мониторинг конкурентов.

Удобство использования. Интерфейс *Majestic* прост и удобен в использовании, и он легко понятен даже для новичков. Основной панелью управления является круговая диаграмма в центре экрана, которая обеспечивает быстрый доступ к основным функциям инструмента.

Точность анализа. *Majestic* является надежным инструментом и позволяет получать достаточно точные данные о ссылочном профиле сайта, позициях в поисковой выдаче и конкурентах.

Стоимость подписок. Стоимость подписок на *Majestic* выше, чем у *MegaIndex*, и составляет от 49,99 до 399,99 долларов в месяц.

Преимущества:

- обладает наиболее великолепной базой данных обратных ссылок;
- выявляет и анализирует новые страницы и обратные ссылки сайта более быстро, чем конкуренты;
- доступ к *API* и автоматическому обновлению;
- пригоден для сравнительного анализа конкурентов.

Недостатки:

- не всегда обеспечивает корректный анализ сайтов на иностранных языках;
- ограниченный перечень параметров и метрик, по которым можно провести анализ.

В целом, *Majestic* – это еще один мощный аналитический инструмент для *SEO* анализа, который может стать хорошим выбором для оценки параметров сайта, ключевых слов и конкурентов в интернет-бизнесе. *Majestic* также обладает простым в использовании интерфейсом и достаточной точностью анализа.

SimilarWeb – это еще один аналитический инструмент для анализа сайтов, который предлагает данные о трафике, привлекаемом сайтом, и его позициях в поисковой выдаче. На рисунке 4 представлена главная страница *SimilarWeb*.

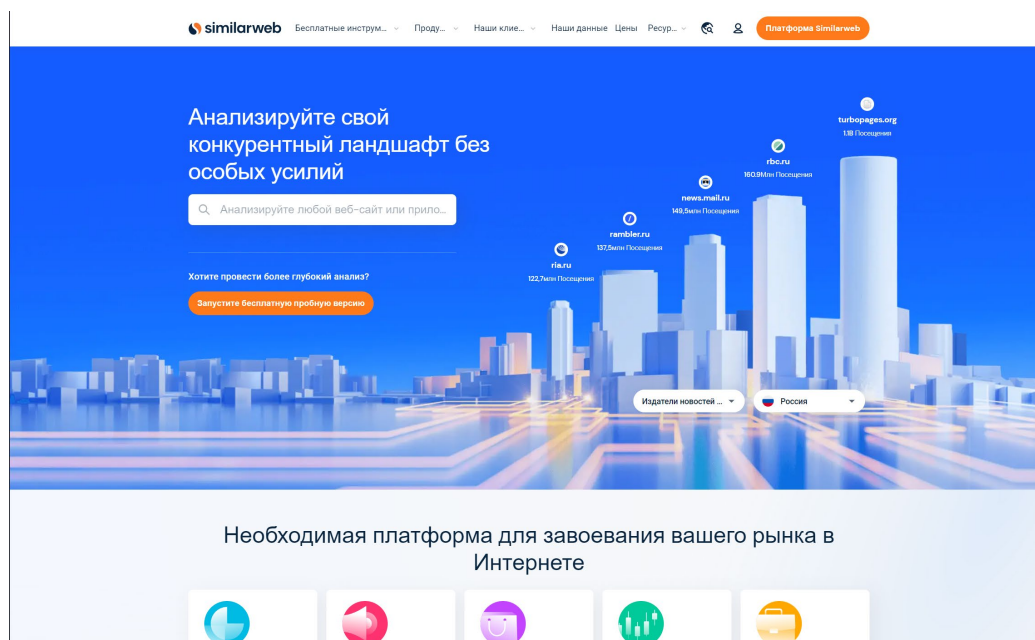


Рисунок 4 – Программный анализатор *SimilarWeb*

Сравнение *SimilarWeb* с остальными выглядит следующим образом.

Функциональность. *SimilarWeb* анализирует чужие сайты, а не только ваши, и предоставляет данные о трафике, источниках трафика, популярности сайта по странам и тематикам и многое другое. Однако, *SimilarWeb* предоставляет менее подробную информацию о позициях сайта в поисковой выдаче.

Удобство использования. Интерфейс *SimilarWeb* достаточно простой и интуитивно понятный. Однако, использование *SimilarWeb* может потребовать некоторого времени для ознакомления с доступными функциями и получения достоверных данных.

Точность анализа. *SimilarWeb* предоставляет достаточно точные данные о трафике, популярности сайта и его привлекательности. Однако, его оценка позиций сайта в поисковой выдаче может не быть столь точной.

Стоимость подписок. Стоимость подписок на *SimilarWeb* также может быть выше, чем у *Majestic* и составляет от 199 до 799 долларов в месяц.

Преимущества:

- позволяет определить популярность сайта в определенной стране и в мире;
- выдает детализированные сведения о посетителях сайта, в том числе географические данные, возраст и пол, категории интересов и другие;
- обладает наиболее полной информацией о рекламных кампаниях конкурентов и целевой аудитории;
- имеет возможность анализа мобильной версии сайта.

Недостатки:

- стоимость услуг относительно высока;
- пригоден только для анализа крупных сайтов.

В целом, *SimilarWeb* является еще одним хорошим аналитическим инструментом для анализа сайтов, который предоставляет достаточно точные данные о трафике, привлекаемом сайтом, и его популярности.

Исходя из результатов, все анализаторы предоставляют достаточно точные данные для анализа интернет-ресурсов. Однако, каждый из них имеет свои сильные и слабые стороны, которые могут влиять на выбор при работе с конкретными задачами.

Например, если главной задачей является анализ трафика, ссылок и ключевых слов, *Pr-cy* является хорошим выбором, так как он стартует с \$15 в месяц, что является более бюджетным вариантом. *MegaIndex* может быть нужен, если интересуется анализ контекстной рекламы. Если же вам необходим анализ профиля ссылок, то *Majestic* является наиболее полным из представленных инструментов, но он также является более дорогим, чем другие варианты. Наконец, *SimilarWeb* широко используется профессиональными маркетологами, так как предоставляет информацию о контенте, пользовательском поведении и других показателях, которые могут быть важны для целей маркетинга. Однако его стоимость начинается от \$199 в месяц.

Таким образом, при выборе инструмента для работы с интернет-ресурсами, необходимо обращать внимание на функционал, удобство

использования, точность анализа и стоимость подписок и выбирать инструмент, который наилучшим образом подходит под ваши цели и задачи.

1.1.4 Выбор языка программирования

Существует множество языков программирования, которые могут использоваться для *backend* разработки веб-приложений, логотипы представлены на рисунке 5.



Рисунок 5 – Языки программирования

Рассмотрим несколько из них и сравним их по некоторым критериям.

Java.

Java является одним из самых популярных языков программирования для *backend* разработки. *Java*-приложения запускаются на виртуальной машине, что обеспечивает высокую степень переносимости. Он также имеет обширную библиотеку, которая помогает разработчикам быстро и легко создавать функциональность.

Плюсы:

- прост в использовании и изучении;
- огромное количество библиотек и фреймворков;
- полная ориентированность на объекты.

Минусы:

- долгая компиляция и запуск приложения;
- не все фреймворки являются совместимыми с последними версиями языка.

Python.

Python имеет очень простой синтаксис и хорошо подходит для быстрой разработки прототипов. Он также имеет большое сообщество и множество библиотек и фреймворков для создания веб-приложений.

Плюсы:

- язык программирования с высоким уровнем абстракции;
- бесплатный и с открытым исходным кодом;
- имеет богатое сообщество разработчиков;
- огромное количество библиотек для создания веб-приложений.

Минусы:

- интерпретируемый язык, что может быть медленным на больших проектах;
- ограничения при работе с многопоточностью.

Ruby.

Ruby также является популярным языком программирования для *backend* разработки на основе фреймворка *Ruby on Rails*. Он имеет простой и понятный синтаксис и хорошо подходит для быстрой разработки прототипов.

Плюсы:

- простой и элегантный синтаксис;
- большое количество готовых библиотек;
- эффективная поддержка многопоточности;
- огромное количество фреймворков для создания веб-приложений.

Минусы:

- медленнее, чем *Java*;
- проблемы с масштабированием.

PHP.

PHP является одним из самых распространенных языков программирования для *backend* разработки веб-приложений. Он хорошо интегрируется с *HTML*, что упрощает создание динамических веб-сайтов.

Плюсы:

- легкий и простой язык программирования;
- огромное количество готовых библиотек и фреймворков;
- большое сообщество и множество ресурсов для обучения.

Минусы:

- ограниченная поддержка многопоточности;
- синтаксические особенности могут затруднять отладку.

Каждый из этих языков программирования имеет свои плюсы и минусы, и выбор зависит от конкретных потребностей проекта. *Java* - оптимальный выбор для крупных и высоконагруженных проектов, *Python* - для быстрой разработки прототипов, *Ruby* - для простоты в разработке, *PHP* - для веб-сайтов, которые работают с базами данных. В любом случае выбор языка зависит от требований конкретного проекта и уровня квалификации разработчиков. Учитывая все факторы, был выбран *PHP*.

1.1.5 Выбор фреймворка для серверной части приложения

Существует множество фреймворков для *PHP*, каждый со своими особенностями и преимуществами. Рассмотрим некоторые из них.



Laravel [4] – это самый популярный *PHP* фреймворк с открытым исходным кодом, построенный на основе архитектуры *MVC*. Он предоставляет легкий и элегантный синтаксис, безопасность и надежность. *Laravel* также предоставляет массу инструментов для управления маршрутизацией, валидацией, авторизацией, приложениями в реальном времени и др.



Symfony – это гибкий и мощный *PHP* фреймворк, который пользуется хорошей репутацией в корпоративной среде. Он предназначен для создания сложных веб-приложений и имеет отличные функции для работы с базами данных. Он предлагает множество инструментов для создания масштабируемых приложений, включая контейнеры зависимостей, компоненты, события и др.



CodeIgniter – это легковесный *PHP* фреймворк, который отлично подходит для создания маленьких и средних проектов. Он имеет небольшой вес и очень быстро работает, что делает его отличным выбором для проектов с ограниченными ресурсами. Он предоставляет базовые инструменты для управления базами данных, маршрутизацией и рендеринга *HTML*.



Yii – это еще один высокопроизводительный фреймворк, написанный на *PHP*. Он поддерживает шаблон проектирования *MVC* и имеет специальные инструменты для обработки форм, работы с базами данных и аутентификации. Он также предоставляет возможность использовать компоненты от других разработчиков, что делает его серьезным конкурентом других фреймворков.



CakePHP – это быстрый и современный *PHP* фреймворк, который был создан с использованием концепции "*convention over configuration*". Это означает, что он имеет множество предустановленных настроек, которые

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		22

определены автоматически, без необходимости указания каждого куска кода. *CakePHP* также предоставляет встроенные инструменты для работы с базами данных, аутентификацией пользователей и шаблонами.

В конечном итоге выбор фреймворка зависит от опыта разработчика и потребностей проекта. Все перечисленные выше фреймворки предоставляют большой набор инструментов и функций для быстрой разработки и безопасности вашего приложения.

Есть несколько причин, по которым был выбран фреймворк *Laravel*.

Во-первых, *Laravel* – это фреймворк с открытым исходным кодом, что означает, что он бесплатен и доступен для всех. Он имеет большое сообщество разработчиков, которое продолжает оттачивать и расширять функциональность фреймворка, а также средства разработки и инструменты.

Во-вторых, *Laravel* имеет четкую и аккуратную архитектуру, что делает его легким в использовании и легко масштабируемым для разработки больших и сложных приложений.

Третья причина, по которой *Laravel* может быть хорошим выбором, это то, что он имеет встроенную поддержку для автозагрузки классов, отправки электронной почты, управления сессией, авторизации и служб очередей.

Наконец, *Laravel* также имеет чрезвычайно хорошую документацию и сообщество на *Stack Overflow* и других форумах, готовые помочь и ответить на ваши вопросы, что делает его проекты более доступными и понятными.

Кроме того, *Laravel* обеспечивает хороший уровень безопасности веб-приложений. Фреймворк предоставляет набор инструментов для защиты приложений от уязвимостей, таких как *SQL*-инъекции, *CSRF*-атаки и других видов взломов.

Laravel также предоставляет мощный механизм маршрутизации, который позволяет определить маршруты запросов и ответов в вашем приложении. Это помогает управлять ссылками на различные страницы и ресурсы внутри приложения.

Еще одной причиной использования *Laravel* является то, что фреймворк предоставляет инструментарий для разработки *REST API*. Это полезно для создания мобильных, веб- и настольных приложений, которые работают с этими *API*.

Laravel также имеет встроенные возможности для тестирования, что облегчает создание и проведение тестов на работоспособность вашего приложения.

Наконец, *Laravel* использует подход *MVC*, что делает его более легким в использовании и обслуживании. Он также предоставляет возможности для настройки базы данных, что сделает ваш код более читаемым и управляемым.

В целом, *Laravel* является мощным и гибким фреймворком, который обеспечивает всестороннюю поддержку для веб-разработки. Его большое сообщество разработчиков и хорошая документация делают его ориентированным на использование в производственных проектах и подходящим выбором для разработки веб-приложений.

1.1.6 Хранение данных

Для организации структурированного хранения данных часто применяют БД. В случае использования *Laravel* в качестве фреймворка, наилучшим выбором будет использование готового решения в виде *Eloquent ORM*. С таким подходом можно взаимодействовать с базой данных, используя классы-модели языка программирования, а не прямое выполнение *SQL*-запросов к базе данных.

Приблизительная схема запроса к БД через *ORM* представлена на рисунке 6.

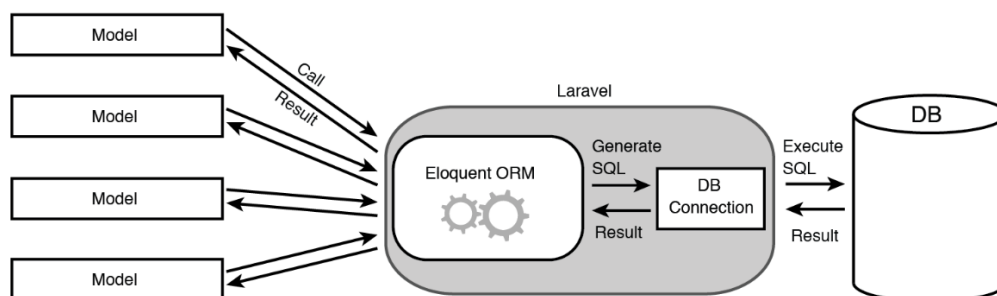


Рисунок 6 – Схема запроса к БД через *Eloquent ORM*

1.1.7 Выбор СУБД

На сегодняшний день существует множество видов баз данных и систем управления ими. Базы данных можно разделить на два основных класса: реляционные (*SQL*) и не реляционные (*NoSQL*). Основное отличие между ними заключается в представлении данных.

В не реляционных базах данных данные хранятся, как правило, в виде словарей, где ключом является первичный ключ сущности, а значением - все остальные поля таблицы. Такой подход позволяет абстрагироваться от строгой типизации и формата данных. Кроме того, *NoSQL*– базы данных могут хранить иерархические структуры данных, в то время как для реализации такой задачи в *SQL*-базах данных потребовалось бы создавать дополнительные таблицы. Самыми популярными *NoSQL* СУБД являются *MongoDB*, *HBase*, *Cassandra*.

Реляционные базы данных также известны как табличные. *SQL*-базы данных разделяются на два вида: строковые и колоночные. Основное отличие заключается в способе хранения данных.

В колоночных базах данных данные одной колонки физически расположены близко друг к другу на жестком диске, что позволяет наиболее эффективно сжимать информацию. Обычно такие базы данных используются для записи логов и в аналитических целях, например, компания Яндекс специально для своего сервиса Яндекс.Метрика разработала СУБД *ClickHouse*. Но существенным недостатком является крайне низкая скорость обновления данных.

В строковых базах данных данные на физическом носителе хранятся построчно, что увеличивает время обработки и выборки, однако позволяет быстро обращаться к нужному элементу по идентификатору, изменять его или удалять. Кроме того, некоторые строковые СУБД более эффективны по сравнению с *NoSQL*.

Ниже перечислены некоторые из наиболее популярных СУБД и краткая информация о них.

СУБД *Oracle* предоставляет широкий спектр возможностей для управления данными и разработки приложений.

Преимущества:

- высокая надежность;
- производительность и масштабируемость;
- широкий спектр возможностей для управления данными и разработки приложений.

Недостатки:

- высокая стоимость;
- сложность в использовании;
- наличие лицензионных ограничений.

MySQL является одной из самых популярных среди веб-приложений в связи с тем, что она открытая схема, бесплатная в использовании и легка в настройке.

Преимущества:

- открытая схема;
- бесплатная в использовании;
- легка в настройке;
- быстрая скорость работы.

Недостатки:

- могут возникнуть сложности при работе с большими объемами данных;
- ограниченный функционал в сравнении с другими СУБД;
- отсутствие устойчивости к сбоям.

PostgreSQL – это СУБД с открытым исходным кодом, которая позиционируется как полнофункциональная альтернатива *Oracle* и *MySQL*. *PostgreSQL* поддерживает *ACID*-транзакции, доступ к объектам базы данных через язык программирования и хранилище данных.

Преимущества:

- открытый исходный код;

- широкий функционал;
- масштабируемость;
- хорошая поддержка языка *SQL*.

Недостатки:

- сложность в настройке;
- высокое потребление ресурсов;
- требовательность к аппаратным ресурсам.

SQL Server является разработкой корпорации *Microsoft* и предназначен для работы на *Windows*-серверах. Она имеет широкие возможности, предназначенные для обработки и анализа данных, а также интеграции с другими продуктами *Microsoft*.

Преимущества:

- широкий функционал;
- интеграция с другими продуктами *Microsoft*;
- хорошая поддержка;
- отличное управление данными.

Недостатки:

- высокая стоимость;
- доступность только на *Windows*-серверах.

MongoDB – это документ-ориентированная СУБД с открытым исходным кодом, которая использует *JSON*-подобные документы для хранения данных. Она может работать на кластерах высокой доступности и горизонтальной масштабируемости.

Преимущества:

- гибкость в управлении данными;
- возможность масштабирования;
- быстрые операции записи и чтения.

Недостатки:

- не подходит для сложных транзакций;

– меньшая надежность и устойчивость к сбоям по сравнению с другими СУБД.

Это только некоторые из многочисленных СУБД, которые используются сегодня.

MariaDB – это копия *MySQL*, созданная в результате приобретения *MySQL* корпорацией Oracle. Используя исходный код *MySQL*, команда разработчиков *MariaDB* добавила дополнительные функции и улучшения для увеличения производительности и совместимости.

Вот некоторые преимущества и недостатки *MariaDB* по сравнению с другими СУБД:

– в сравнении с *Oracle*, *MariaDB* более доступна и легка в использовании, а также имеет открытую схему, что делает ее привлекательной для разработчиков;

– *MariaDB* предоставляет большую производительность и скорость работы в сравнении с *MySQL*, а также имеет совместимость в синтаксисе запросов *SQL*, что делает переход с *MySQL* более простым;

– *MariaDB* легче в настройке и имеет более простой пользовательский интерфейс. Однако *PostgreSQL* более функциональна и поддерживает сложные запросы;

– *MariaDB* более гибкая и имеет более открытую архитектуру, чем *SQL Server*;

– *MariaDB* более стандартизирована и имеет большой набор функций, чем *MongoDB*.

MariaDB является качественным выбором для малого и среднего бизнеса, веб-разработки и некоторых финансово-технических задач. Она может быть более простой в использовании и настройке, чем некоторые из более сложных СУБД, и имеет широкий функционал и функциональность. Однако, выбор конкретной СУБД зависит от конкретных требований и потребностей пользователя. Учитывая все плюсы и минусы, выбор был остановлен на *MariaDB*.

1.1.8 Выбор платформы, хостинга, веб-сервера

В качестве хостинга *VPS* был выбран джино, на который была установлена операционная система *Centos*, и веб-сервер *NGINX* [5].

Выбор *VPS* от джино имеет некоторые преимущества:

- высокая производительность - хостинг джино использует серверы нового поколения, оснащенные быстрым *HDD/SSD*, что обеспечивает высокую скорость чтения и записи данных. Выделенные *CPU* и *RAM* гарантируют стабильную производительность и отзывчивость приложений;

- гибкость – у джино есть различные тарифные планы, позволяющие выбрать оптимальный вариант для нужд пользователя. Пользователь может выбирать ОС для установки, выбирать необходимый объем дискового пространства и возможность масштабирования в будущем по мере роста потребностей;

- простота управления – панель управления *VPS* у джино, как правило, интуитивно понятна и позволяет управлять сервером, настроить *DNS*-записи, создавать и управлять базами данных и пользователями. Также имеется доступ к *SSH (Secure Shell)* для более глубоких настроек и управления сервером;

- безопасность – джино производит ежедневное резервное копирование данных пользователей, что позволяет быстро восстановить данные в случае сбоя в системе. Также у джино имеется обширный набор инструментов безопасности, включая защиту от *DDoS*-атак и вирусов.

В целом, джино *VPS* предоставляет надежное решение для хранения и обработки данных, размещения веб-сайтов, приложений и других целей, где требуется выделенная вычислительная мощность и контроль над сервером.

CentOS (Community Enterprise Operating System) – это бесплатная операционная система, основанная на открытом коде *Red Hat Enterprise Linux (RHEL)*. *CentOS* обновляется регулярно, и представляет собой стабильную и надежную платформу для работы как на серверах, так и на настольных ПК.

CentOS – это отличный выбор для веб-проектов, поскольку:

- *CentOS* является бесплатным, что позволяет снизить затраты на развёртывание и поддержку веб-проекта;
- *CentOS* имеет высокую степень стабильности и безопасности, благодаря длительной поддержке и регулярным обновлениям безопасности;
- *CentOS* имеет широкую совместимость с приложениями, что позволяет без проблем работать со многими веб-серверами, базами данных и другими средствами разработки;
- одним из главных преимуществ *CentOS* является его поддержка на уровне ОС, что позволяет успешно работать с различными веб-платформами и сокращает время ответа на проблемы;
- *CentOS* имеет большое сообщество пользователей, которое готово помочь с использованием ОС, так что в случае возникновения проблем всегда можно обратиться за помощью.

В целом, *CentOS* является надежным, стабильным и безопасным выбором для веб-проекта, который обеспечивает высокую производительность и совместимость с широким спектром приложений.

Apache и *NGINX* являются двумя наиболее популярными веб-серверами в мире. Оба веб-сервера имеют свои уникальные преимущества и недостатки, но рассмотрим основные отличия и преимущества *NGINX* в сравнении с *Apache*:

- архитектура. *NGINX* разработан для использования малого числа рабочих процессов и не блокирует соединения при обработке запросов к серверу. В то время как *Apache* использует большое число процессов и потоков для обработки запросов;
- производительность. *NGINX* быстрее, чем *Apache* в обработке статических файлов, особенно при большом количестве одновременных соединений. В данном случае *NGINX* благоприятнее для быстрой настройки конфигурации сервера;
- высокая производительность при высоких нагрузках. *NGINX* эффективнее обрабатывает высокие нагрузки, благодаря особенностям своей

архитектуры, и поэтому используется в крупных проектах и веб-сайтах с высокими показателями трафика;

– простота настройки. *NGINX* легкий и имеет простую структуру, что упрощает его настройку по сравнению с *Apache*.

В результате, если для разработки использовать статический контент, вы ощутите более высокую производительность веб-сервера, а если вам нужно быстро обрабатывать высокие нагрузки, то лучше выбрать *NGINX*.

1.1.9 Выбор библиотеки для реализации графического интерфейса

Из множества возможных вариантов выделяются *React* и *Vue*, которые имеют разные подходы к разработке и отличаются друг от друга в ряде аспектов.

React - это библиотека *JavaScript*, основанная на создании компонентов, которые позволяют отображать пользовательский интерфейс, основанный на состоянии. Разработчикам также предоставляются инструменты для разработки многократно используемых компонентов, что значительно упрощает их повторное использование и тестирование. В отличие от *Vue*, *React* не использует шаблонный язык для отображения, что делает его более гибким и дает больше свободы в разработке. Кроме того, *React* является более быстрым и эффективным, что может быть важным фактором при разработке сложных пользовательских интерфейсов.

Vue, с другой стороны, является относительно новой библиотекой, которая также используется для создания компонентов, но имеет некоторые особенности, которые отличают ее от *React* и делают ее более подходящей для определенных проектов. Например, *Vue* использует шаблонный язык для отображения, что может упростить и ускорить создание пользовательского интерфейса в некоторых случаях. Кроме того, *Vue* является более легкой и более простой в изучении, что может быть полезно для небольших команд или проектов.

В целом, как *React* [6], так и *Vue* могут быть использованы для создания качественного пользовательского интерфейса клиентского приложения. Хотя

они имеют разные подходы к разработке и отличаются в некоторых деталях, у каждой библиотеки есть свои преимущества и недостатки.

Выбор *React* для использования с *Laravel* для разработки веб-приложения можно обосновать несколькими причинами.

React является одним из самых популярных фреймворков для разработки пользовательского интерфейса на данный момент. Это библиотека, которая активно поддерживается и используется в большом количестве крупных проектов. *React* имеет множество преимуществ, таких как высокая производительность, простота в использовании, возможность создания многоразовых компонентов и прочее.

React призван работать вместе с отдельным *API*, что делает связку *React* и *Laravel* очень удачной. *React* приложение может использовать *Laravel API* для получения данных из базы данных и обновления их без перезагрузки страницы. Это позволяет создать более гибкое взаимодействие между клиентской и серверной частями приложения.

Таким образом, использование *React* и *Laravel API* в качестве технологического стека [7] позволяет создать высокопроизводительное и гибкое веб-приложение с удобным взаимодействием между клиентской и серверной частями.

1.1.10 Функциональные требования

Требования к программному анализатору на основе пользовательских сценариев, разработанных ранее.

Программный анализатор должен:

- иметь возможность анализировать интернет-ресурсы и выдавать отчет об их структуре, содержании и качестве контента;
- позволять пользователю сравнить несколько интернет-ресурсов на основе их структуры, содержания и качества контента;
- иметь функционал, который позволил бы пользователю получить отчет о интернет-ресурсе, который был изменен в определенный период времени;

- иметь возможность производить анализ интернет-ресурсов на наличие определенных тегов или ключевых слов;
- позволять пользователю получить отчет об использовании определенных технологий на интернет-ресурсе;
- быть легким в использовании, чтобы пользователь мог легко понимать функционал приложения;
- обладать быстрой скоростью обработки и анализа данных, чтобы пользователь мог быстро получать отчеты о результате;
- обладать высокой точностью и полнотой анализа, чтобы пользователь мог получать наиболее точные и полные отчеты о качестве интернет-ресурсов.

1.1.11 Выводы по предпроектному анализу

Выводы по предпроектному анализу:

- показана актуальность выбранной темы;
- описаны основные типы пользователей;
- описаны функциональные требования к разрабатываемому приложению;
- проведен анализ и выбраны инструменты для разработки.

1.2 Проектирование и разработка

Программный анализатор состоит из *VPS* [8] сервера, клиентского и серверного приложения. Сервер собирает, обрабатывает и хранит информацию о различных интернет-ресурсах. Клиентами могут быть браузер, *postman*, а также *desktop*-приложения и мобильные.

Приведенное на рисунке 7 программное обеспечение позволяет: проводить общий анализ интернет-ресурсов, смотреть блог разработки, в котором описываются значительные нововведения, смотреть инструменты разработанные и находящиеся в разработки, таже смотреть рейтинг сайтов университетов.

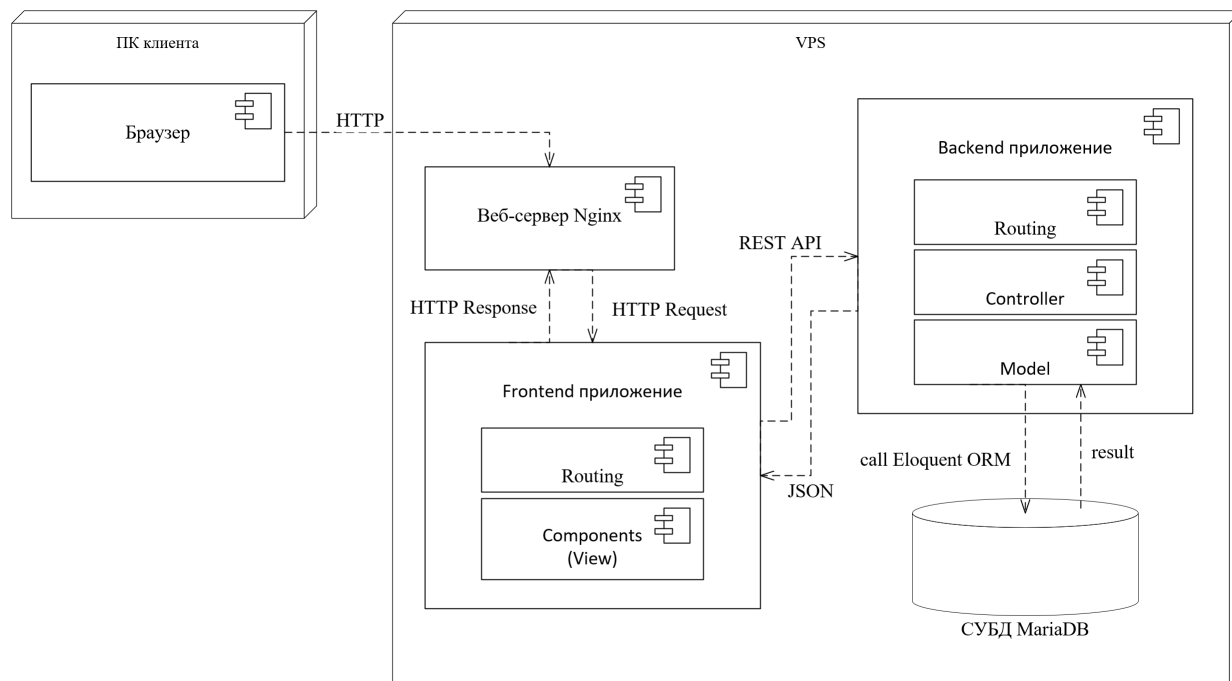


Рисунок 7 – UML диаграмма развертывания

1.2.1 Реализация VPS сервера

В качестве сервера был выбран *VPS* от хостинга джино. Произведено создание с выбором конкретных настроек, был выбран минимальный тариф, где сервер с довольно низкими системными характеристиками, для того чтобы начать пойдет, но в скором времени пришлось переходить на более мощный. В качестве образа был выбран *CentOS 7*, в последствии при смене сервера, обновил до *CentOS 8*. Была выбрана версия *PHP 7.4*. и последняя версия *MariaDB 10.6*. За управление нагрузками, конфигурирование подключений и проксирование клиентского и серверного приложения отвечает веб-сервер *Nginx*. На рисунках 8 и 9 представлены настройки при создании *VPS*.

← Новый сервер

Тип тарифа

Готовый тариф
Использование сбалансированных тарифов
☒

Гибкий тариф
Настройка тарифа в конструкторе
☐

Тип сервера

SSD
Высокая производительность
☒

SSD+HDD
Больше места за меньшую цену
☐

Тариф

Альфа
Достаточно, чтобы попробовать

CPU 1 ядро × 2000 МГц
RAM 1.5 Гб
Диск 10 Гб SSD
Выделенные IP —

129 ₹ в месяц

Бета
Тариф начального уровня

CPU 2 ядра × 2000 МГц
RAM 4 Гб
Диск 20 Гб SSD
Выделенные IP 147 ₹

399 ₹ в месяц

Гамма Рекомендуем
Подойдет для большинства задач

CPU 2 ядра × 2000 МГц
RAM 8 Гб
Диск 50 Гб SSD
Выделенные IP 147 ₹

799 ₹ в месяц

Гамма Плюс +
Только для участников программы Джино.Плюсы +

CPU 3 ядра × 2000 МГц
RAM 10 Гб
Диск 70 Гб SSD
Выделенные IP 147 ₹

999 ₹ в месяц

Дельта
Для самых требовательных

CPU 4 ядра × 2000 МГц
RAM 12 Гб
Диск 100 Гб SSD
Выделенные IP 147 ₹

1599 ₹ в месяц

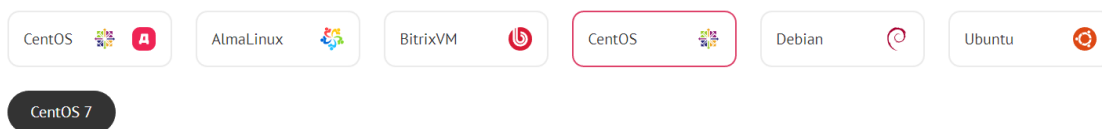
Эпсилон
Максимальная производительность

CPU 6 ядер × 2000 МГц
RAM 16 Гб
Диск 130 Гб SSD
Выделенные IP 147 ₹

1999 ₹ в месяц

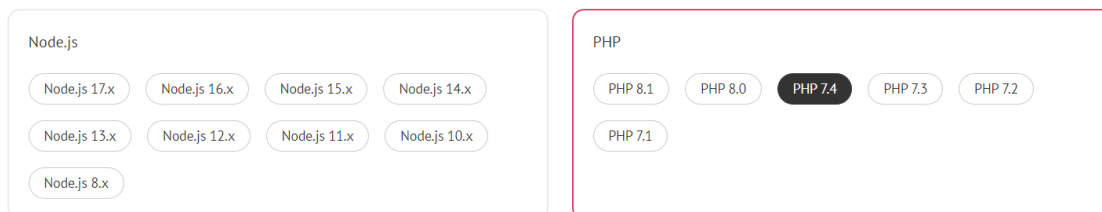
Рисунок 8 – Создание VPS джино

Образ



Приложения

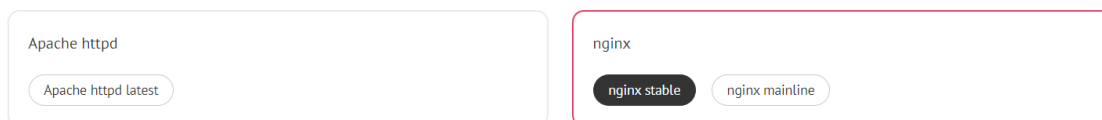
Языки программирования ^



Хранение и обработка данных ^



Веб-серверы ^



Другое v

Рисунок 9 - Создание VPS джино продолжение

После создания сервера была выполнена настройка *Nginx* и установлен *phpMyAdmin* для графического отображения БД. Настройка *Nginx* выполняется в файле *config* все запросы в приоритете идут на *frontend* приложение. Также добавлено проксирование по определенным *URL* на *backend* приложение и *phpMyAdmin*.

1.2.2 Разработка серверной части приложения

Backend приложение разрабатывается на фреймворке *Laravel*. Для начала было необходимо поставить *Composer* на *VPS*. Далее был создан проект

командой «*composer create-project laravel/laravel backend*». Было создано базовое приложение с минимальным функционалом. Для проверки можно было перейти по следующему URL «*https://getindex.ru/back/*». Базовая страница приведена на рисунке 10.

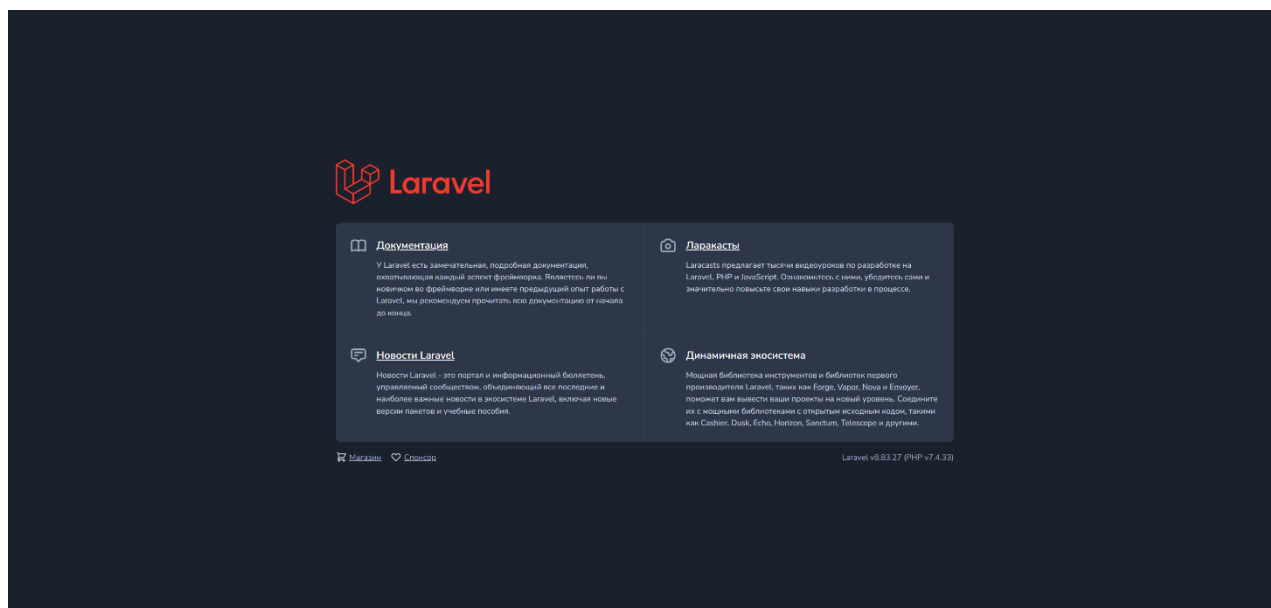


Рисунок 10 – Стартовая страница *Laravel*

После этого было добавлено несколько внешних библиотек:

– *Laravel Sanctum*– легковесная система аутентификации для *SPA*, позволяет каждому пользователю создавать несколько токенов *API* для своей учетной записи.

– *Laravel api-tester* – позволяет просмотреть все маршруты *API* и протестировать их.

Все настройки приложения задаются в конфигурационном файле «.env».

Ниже приведена часть файла .env:

```
APP_NAME=back
APP_KEY= ****
APP_DEBUG=true
APP_URL=https://getindex.ru/back

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=seo_site
DB_USERNAME=*****
DB_PASSWORD=*****
```

APP_NAME – имя приложения используется в различных файлах конфигурации, *APP_KEY* отвечает за секретный ключ всего проекта, *APP_DEBUG* – режим отладки, показывает информацию об ошибках конечному пользователю, *APP_URL* – ссылка приложения используется в некоторых библиотеках, конфигурациях и маршрутизации. *DB_CONNECTION*, *DB_HOST*, *DB_PORT*, *DB_DATABASE*, *DB_USERNAME*, *DB_PASSWORD* необходимы для подключения *Laravel* к СУБД.

Принципы проектирования *MVC* являются одним из фундаментальных элементов *Laravel*.

MVC паттерн разбивает приложение на три основных компонента:

- *Model* (Модель) - отвечает за представление данных и бизнес-логику;
- *View* (Представление) - отвечает за отображение данных пользователю;
- *Controller* (Контроллер) - связующий элемент между *Model* и *View*, обрабатывает пользовательский ввод и убеждается, что изменения взаимодействия с *View* отражаются в *Model*, а также управляет направлением потока данных в приложении.

Laravel использует этот паттерн, предоставляя на более высоком уровне абстракции некоторые стандартные функции, такие как маршрутизация *HTTP*-запросов, миграции базы данных, управление сессиями и т. д.

Единственное нужно учесть, что клиентская часть является отдельным приложением, поэтому в *Laravel* остается *Model* и *Controller*, а *View* реализовывается в приложении *React*.

Входной точкой в *backend* приложение являются запросы от *frontend* приложения. Запросы обрабатываются механизмом маршрутизации *Laravel*.

Laravel предоставляет мощный механизм маршрутизации для *API*. Он позволяет создавать маршруты для обработки запросов к *API*. Маршруты обрабатываются контроллерами.

Маршруты для *API* в *Laravel* определяются в файле *routes/api.php*. Этот файл служит для определения всех маршрутов, относящихся к *API*. Для задания маршрута используется метод *Route::* с указанием *HTTP*-метода (*get*, *post*, *put*,

delete и т.д.) и пути к маршруту. Например, *Route::get('user',[AuthController::class,'user'])* задает маршрут к методу *user* контроллера *AuthController*, который будет обрабатывать *GET*-запросы к */users*.

Контроллеры используются для обработки запросов к *API*. В *Laravel* контроллеры определяются как классы *PHP*, который содержит методы для обработки конкретных маршрутов. Методы контроллера должны возвращать данные в формате *JSON*, так как *API* работает с данными в этом формате.

Контроллеры *API* часто используются для обработки запросов на создание, чтение, обновление и удаление данных (*CRUD*). Они также могут выполнять дополнительные задачи, такие как аутентификация пользователей, проверка прав доступа и обработка ошибок.

В *Laravel*, создание контроллеров *API* можно осуществить с помощью следующей команды *Artisan*: «*php artisan make:controller BlogController --api*».

В контроллере *API* обработка различных типов запросов осуществляется с помощью методов, которые принимают экземпляр класса *Illuminate\Http\Request* в качестве аргумента. Например, для обработки *GET*-запросов, метод *index* может быть определен следующим образом:

```
public function index()
{
    $blogs =
BlogModel::leftjoin('blog_types','blog.id_type', '=',
'blog_types.id')->where('blog.is_deleted', 0)->select('blog.*',
'blog_types.name')->get();
    return response()->json($blogs, 200);
}
```

Кроме того, *Laravel* также предлагает множество инструментов для управления маршрутизацией и организации *API*. Например, можно использовать *middleware*, чтобы добавлять промежуточные действия перед выполнением запросов к маршрутам, авторизацию и многое другое.

На рисунке 11 представлена *UML* диаграмма компонентов, можно просмотреть различные запросы, их маршрутизацию, и какие контроллеры и модели взаимодействуют.

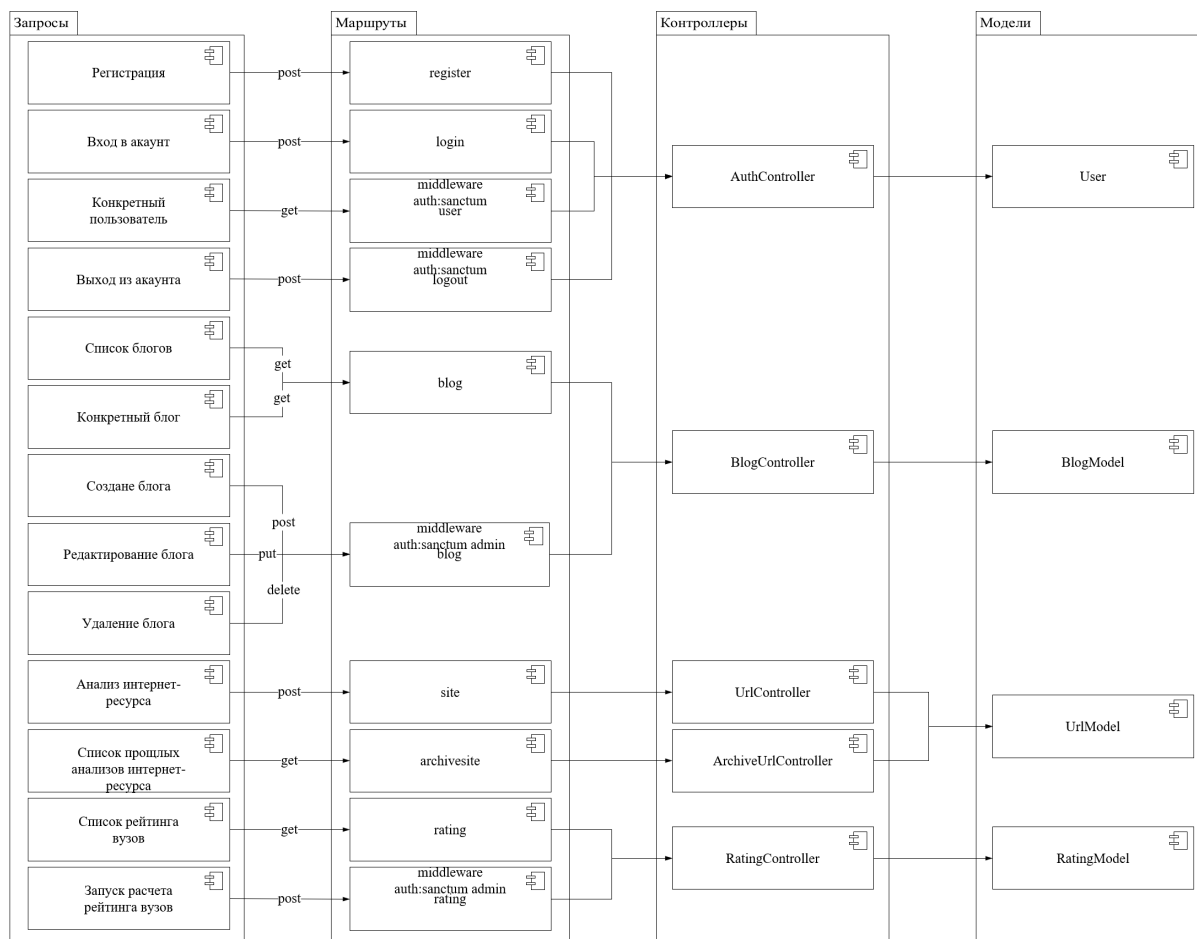


Рисунок 11 – *UML* диаграмма компонентов

1.2.3 Разработка БД

Laravel позволяет чрезвычайно просто взаимодействовать с БД, будь то сырой *SQL*, гибкий построитель запросов или *Eloquent ORM*.

Eloquent ORM – это объектно-реляционная система управления базами данных, встроенная в *Laravel*. Она позволяет управлять базой данных через объекты и методы, что упрощает написание кода и повышает его читаемость.

Для создания моделей используется *Artisan* - специальная утилита, встроенная в *Laravel*. Она автоматически генерирует классы моделей для работы с базой данных. Для создания модели достаточно запустить команду *PHP artisan make:model ИмяМодели, -m* и *Laravel* автоматически создаст класс модели и файл миграции.

Пример простой модели *BlogModel* с которой и началась вся разработка:

```
class BlogModel extends Model
{
```

```

use HasFactory;

protected $table = "blog";
protected $dateFormat = 'U';
protected $dates = [
    'created_at',
    'updated_at'
];
protected $fillable = [
    'id',
    'title',
    'content',
    'created_at',
    'updated_at',
    'id_type',
    'is_deleted'
];
}

```

Кодом будет наглядней так как если строить *UML* диаграмму, то можно увидеть только поля самого класса, используемые фреймворком, а необходимые поля сущности для *BlogModel* находятся в поле *fillable*.

Поле *table* описывает к какой таблице БД подключаться, так как даты хранятся в БД в формате *Unix*, во все моделях появились поля *dateFormat* и *dates*, которые описывают формат времени и поля в БД, относящиеся к датам. Поле *fillable* описывает поля сущности из БД, которые могут использоваться. На рисунке 12 приведена *UML* диаграмма класса *BlogModel*.

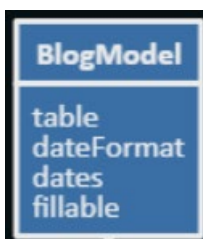


Рисунок 12 – *UML* диаграмма класса *BlogModel*

Миграции – это механизм, позволяющий создавать и изменять структуру базы данных. Они позволяют управлять изменениями в базе данных без необходимости написания *SQL*-скриптов. Для создания миграций в *Laravel* используется *Artisan*. Достаточно запустить команду *php artisan make:migration ИмяМиграции*, и *Laravel* автоматически создаст файл миграции. Затем вам

нужно заполнить методы *up()* и *down()*, описывающие изменения структуры базы данных.

Файл миграции для той же модели *BlogModel* выглядит следующим образом:

```
class CreateBlogTable extends Migration
{
    public function up()
    {
        Schema::create('blog', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->text('content');
            $table->integer('created_at');
            $table->integer('updated_at');
            $table->integer('id_type')->default(0);
            $table->tinyInteger('is_deleted')->default(0);
        });
    }
    public function down()
    {
        Schema::dropIfExists('blog');
    }
}
```

Также есть такие классы для работы с БД как сидеры и фабрики.

Сидеры – это классы, используемые для заполнения базы данных начальными данными (*seed*-ами). Они могут быть полезны при разработке или тестировании, когда вам нужно заполнить базу данных данными для проверки функциональности.

Фабрики – это классы, которые позволяют создавать элегантный код для генерации «фабричных» объектов, используемых для тестирования. Таким образом, можно быстро создавать большие наборы тестовых данных, не переживая о деталях их создания.

В *Laravel* доступны множество других функций и возможностей для работы с базами данных. В частности, можно использовать фасады для работы с таблицами, создавать отношения между моделями и многое другое.

Вся структура БД представлена на рисунке 13 в виде физической модели.

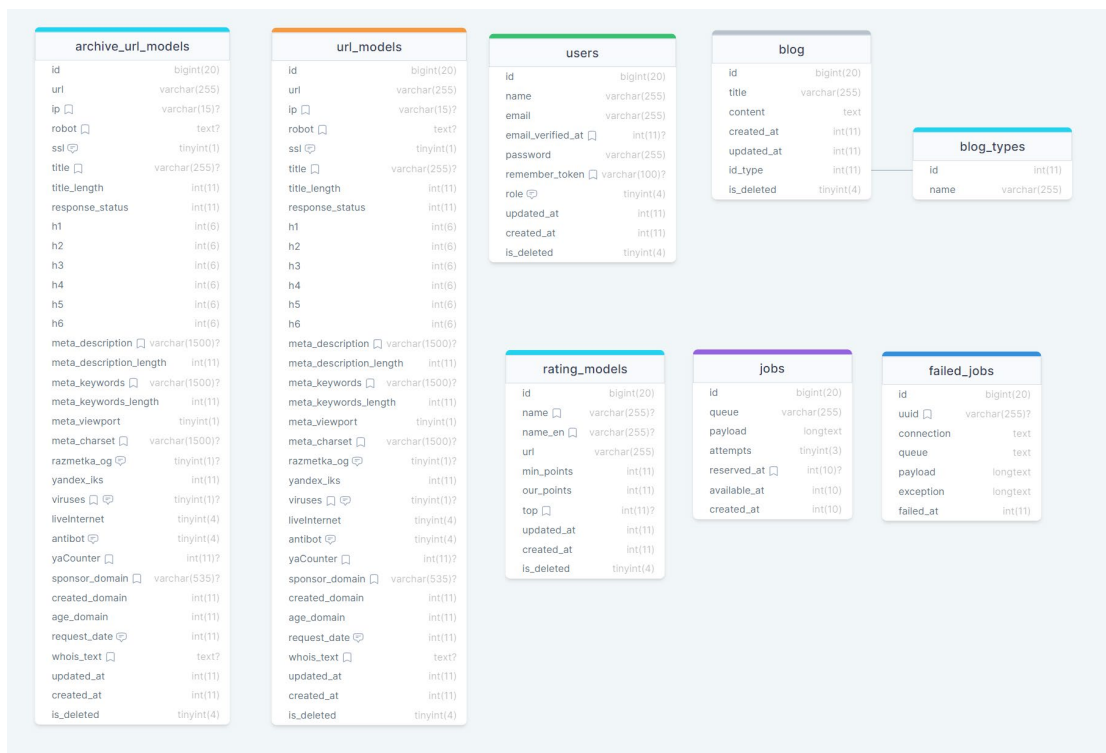


Рисунок 13 – Физическая модель базы данных

1.2.4 Реализация основного метода сбора и структурирования информации

На вход принимается *URL*, из которого вычленяется домен. Анализируется главная страница интернет-ресурса. Сначала проверяется как давно сканировался этот ресурс, если прошло менее дня с момента анализа, то пользователю отдается прошлый результат сканирования, это сделано для снижения нагрузки на сервер. Если прошло более дня – запускается метод *parsAndCreate(\$domen)* описанный в модели *UrlModel*.

Метод *parsAndCreate* вызывает другие методы, основной из которых *getPage(['url'=>\$domen])* в этом методе используется *CURL*.

CURL – это библиотека для передачи данных по многим протоколам, включая *HTTP*, *FTP* и многие другие. *PHP* содержит обертку над этой библиотекой, которая позволяет доставать данные с помощью функции *curl_exec()* и получать информацию о запросах с помощью функции *curl_getinfo()*.

Ниже представлен частичный код обращения к *CURL*.

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
```

```

curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
...
$content = curl_exec($ch);
$info = curl_getinfo($ch);
...
curl_close($ch);

```

Функция *curl_setopt()* используется для настройки параметров *CURL*, таких как *URL*-адрес, опции соединения и т. д. Параметр *CURLOPT_RETURNTRANSFER* устанавливает флаг, указывающий *CURL* сохранять *HTTP*-ответ в переменной, а не выводить его на экран.

Функция *curl_exec()* выполняет *HTTP*-запрос к заданному *URL*-адресу и возвращает содержимое ответа в виде строки. Она принимает в качестве параметра объект *CURL* или его дескриптор, настроенный на выполнение запроса.

Функция *curl_getinfo()* возвращает информацию о последнем запросе *CURL* в формате массива. Она может быть полезна для отладки и анализа ответов сервера.

Далее в методе *parsAndCreate* происходит разбор и анализ всего контента, на различное содержание тегов и их количество например: заголовки *h* разного уровня или мета теги. Также по контенту проверяется наличие счетчиков, микроразметки. Из ответа сервера можно узнать: были ли перенаправления с введенного домена на другую страницу, наличие *ssl* сертификата, сам статус ответа сервера, кодировку содержимого и многое другое.

С помощью метода *lookup* из *PhpWhois* – это *PHP* библиотека, получается информация о доменном имени, такая как имя владельца, контактная информация, даты регистрации, *DNS*, и многое другое, с помощью *WHOIS* протокола. Библиотека содержит встроенные справочные данные для различных доменных зон, чтобы упростить процесс извлечения информации *WHOIS* из различных источников.

С помощью метода *getRobot(\$domen)* проверяется наличие файла *robot.txt* и берется его содержимое. Метод основан на функции *file_get_contents* в *PHP* –

это функция позволяет получать содержимое файла в виде строки. Она принимает в качестве аргумента путь к файлу или *URL*.

Для получения значения Яндекс ИКС был разработан отдельный класс, его работа заключается в том, что по адресу *'https://www.yandex.ru/cycounter?'. \$domain* можно получить картинку со значения Яндекс ИКС и далее класс вычленяет из картинки цифры, с помощью методов *PHP* для работы с изображениями.

После получения разбора все данные складываются в БД. На рисунке 14 представлен алгоритм работы описанного метода.

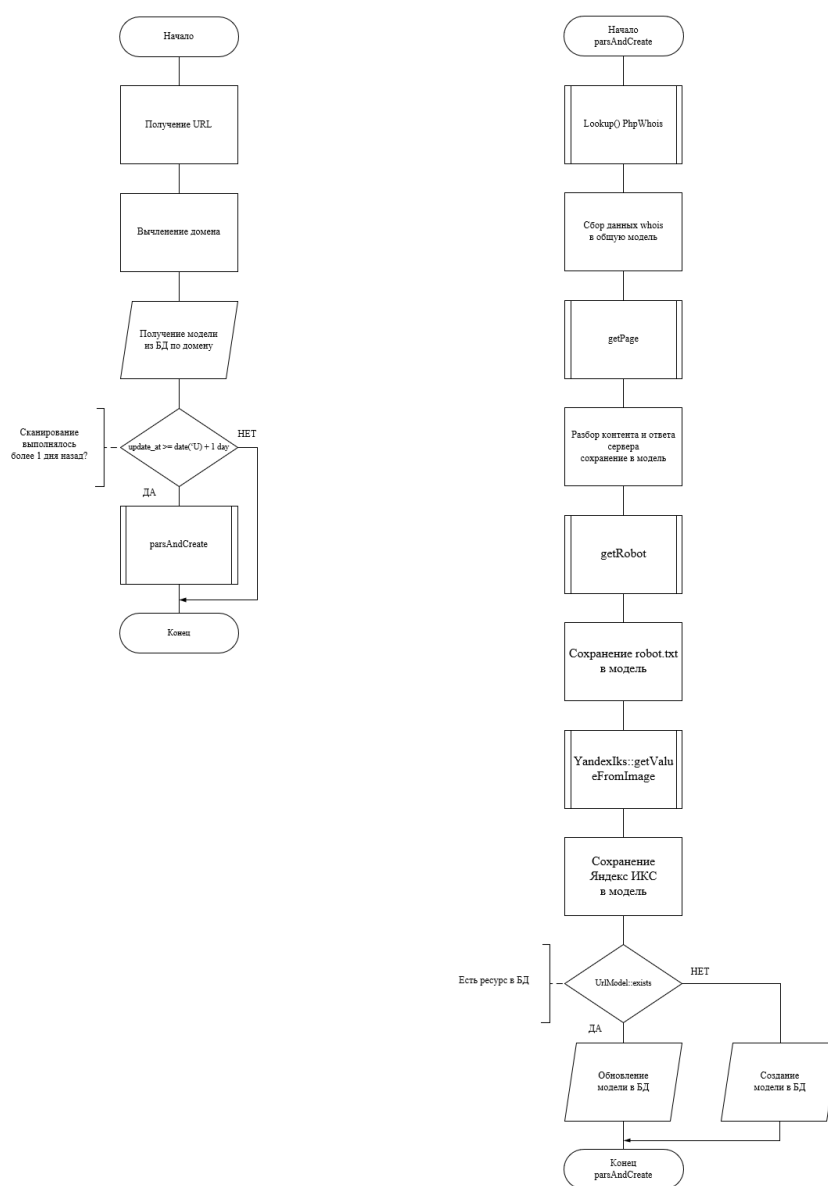


Рисунок 14 – Алгоритм анализа интернет-ресурса

1.2.5 Разработка клиентской части приложения

Разработка началась с подготовки окружения. Установлена последняя версия *Node.js* и создан новый проект *React* с помощью *Create React App*.

Следующим этапом была реализована маршрутизация. Маршрутизация – это процесс обработки *URL* и определения, какой компонент должен быть отображен на странице. Когда пользователь вводит *URL*, *React* должен отобразить соответствующий компонент в зависимости от маршрутизации.

Для реализации маршрутизации в *React* была использована библиотека *React Router*. Она позволяет определить маршруты, указать компоненты, которые должны быть отображены для каждого маршрута, и обрабатывать изменения *URL*.

Ниже часть кода, который показывает, как реализована маршрутизация в проекте:

```
import React, { useEffect, useState } from 'react';
import { BrowserRouter, Routes, Route } from "react-router-dom";
<BrowserRouter>
  <Routes>
    <Route path="/" element={<Main />} />
    <Route path="/blog" element={<Blog />} />
    <Route path="/contact" element={<Contact />} />
    <Route path="/tools" element={<Tools />} />
    <Route path="/register" element={<Register />} />
    <Route path="/login" element={<Login />} />
    <Route path="/analytic" element={<PageAnalytics />} />
    <Route path="/profile" element={<Profile />} />
    <Route path="/myip" element={<MyIp />} />
    <Route path="/rating" element={<Rating />} />
    <Route path="*" element={<Page404 />} />
  </Routes>
</BrowserRouter>
```

Компонент *BrowserRouter* создан для обеспечения маршрутизации в приложении. Затем были созданы компоненты, разложенные по различным файлам.

Компонент *Route* используется для указания компонентов, которые должны отображаться при переходе по определенным маршрутам.

На главной странице использовался компонент, содержащий форму поле для ввода *URL*-адреса, кнопка анализа и результатов программного анализа.

```

<form onSubmit={submit}>
  <input onChange={e => setSite(e.target.value)}
  id="floatingSite" type="text"/>

  <button type="submit"></button>
</form>

```

Во время отправки формы вызывается метод *submit*, в котором идет обращение к *API backend* приложения с помощью *await fetch*, если ответ пришел, то с помощью *useDispatch* из библиотеки *React-Redux* устанавливается состояние для хранения введенного *URL*-адреса.

```

const response = await fetch
('https://getindex.ru/back/api/V1/site', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    url
  })
});

```

Далее происходит перезагрузка страницы с перенаправлением на компонент страницы анализа. Из сохраненного состояния вытягивается введенный *URL*-адрес, и дальше начинается установка всех состояний компонентов, доставая значения из *json* пришедшего от *backend* приложения.

На рисунке 15 графически показана структура клиентской части.

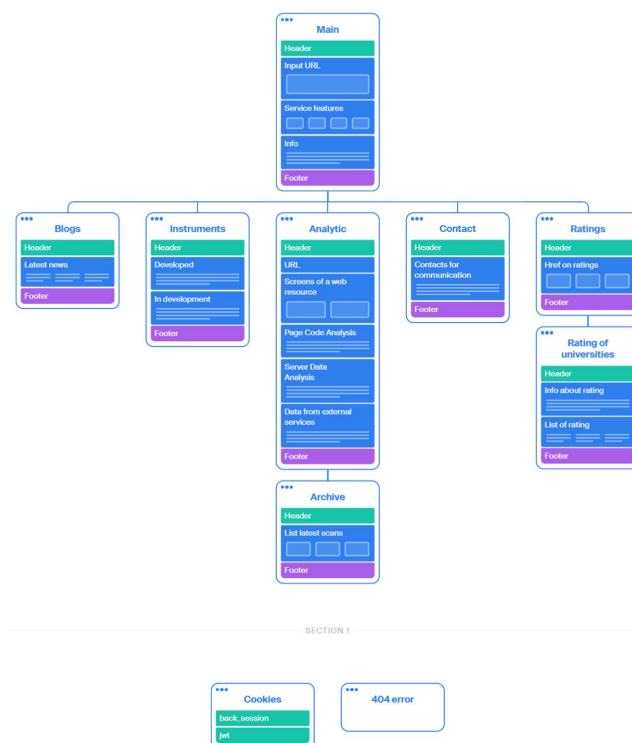


Рисунок 15 – Структура клиентской части приложения

1.3 Анализ результатов

1.3.1 Тестирование API

Для тестирования и отладки API можно воспользоваться программой *Postman*, терминальной командой *curl* или же открыть страницу *backend* приложения *.../api*, представленную на рисунке 16. На странице отображается весь список доступных маршрутов, что упрощает отладку, т.к. сразу видно доступность API и URL-адреса.

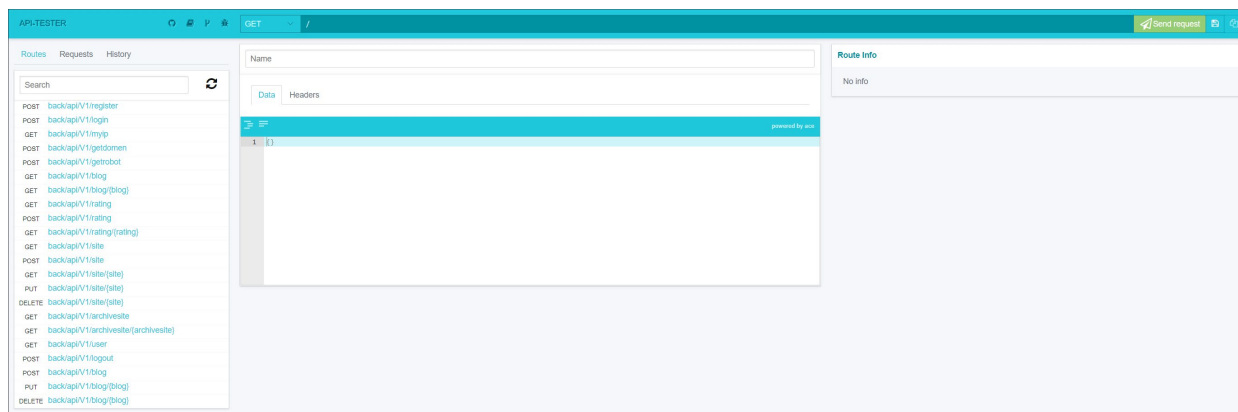


Рисунок 16 – Начальный экран API-Tester

Можно протестировать работу с блогами. На рисунке 17 представлен результат запроса списка блогов.

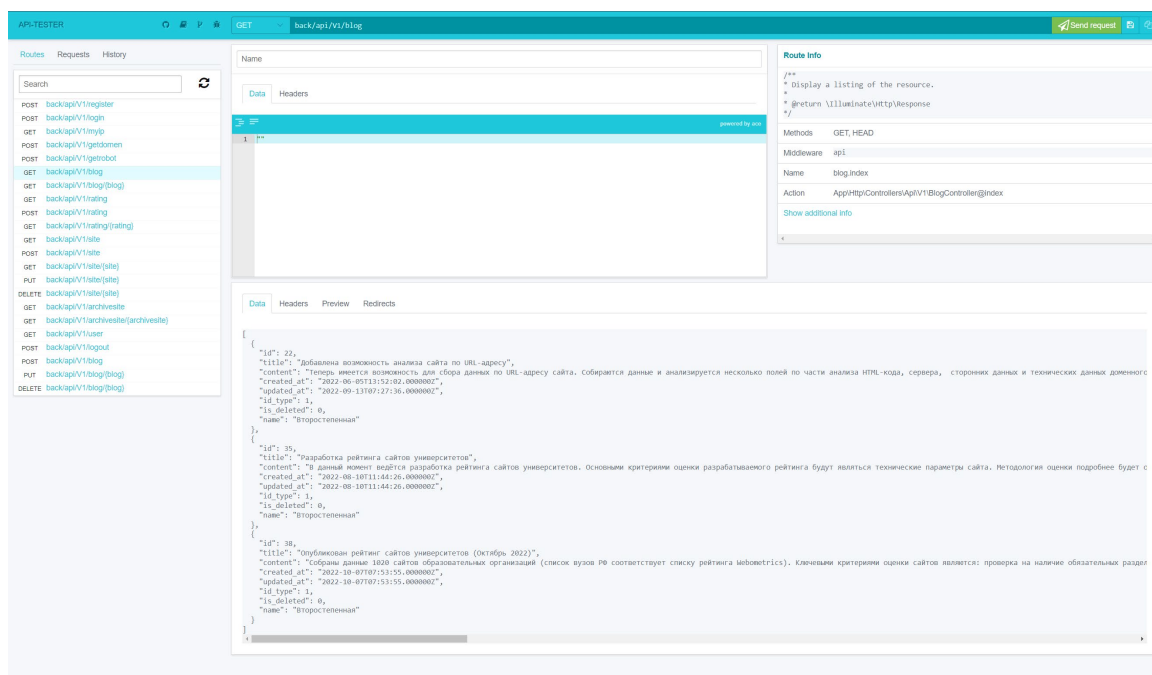


Рисунок 17 – Результат запроса блогов из базы данных

После этого можно попробовать добавить блог в систему. Для этого выбираем нужный маршрут и заполнить форму необходимую для *API*. На рисунке 18 представлен ответ *API* на запрос добавления блога в БД.

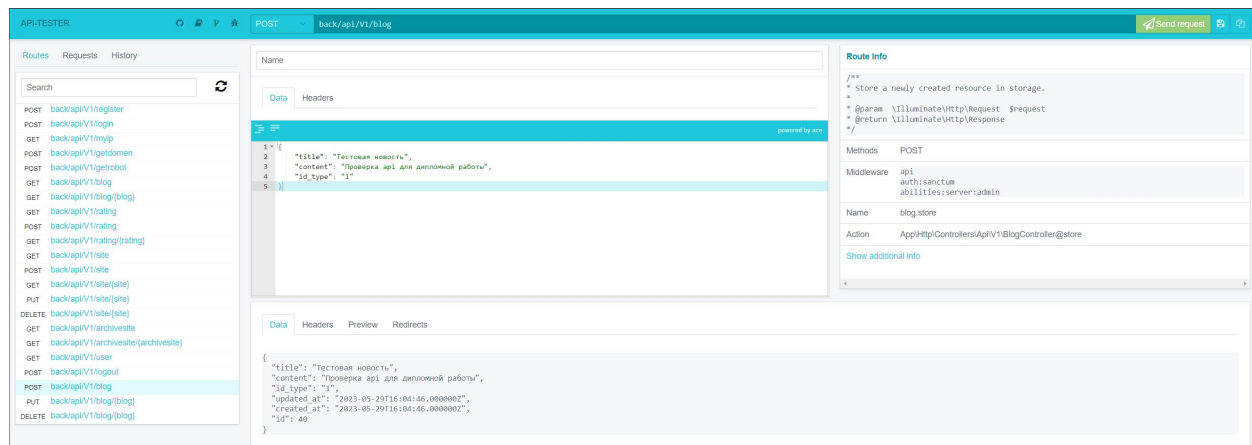


Рисунок 18 – Ответ *API* на запрос добавления блога в БД

Теперь можно попробовать снова запросить список блогов. На рисунке 19 представлен повторный запрос блогов.

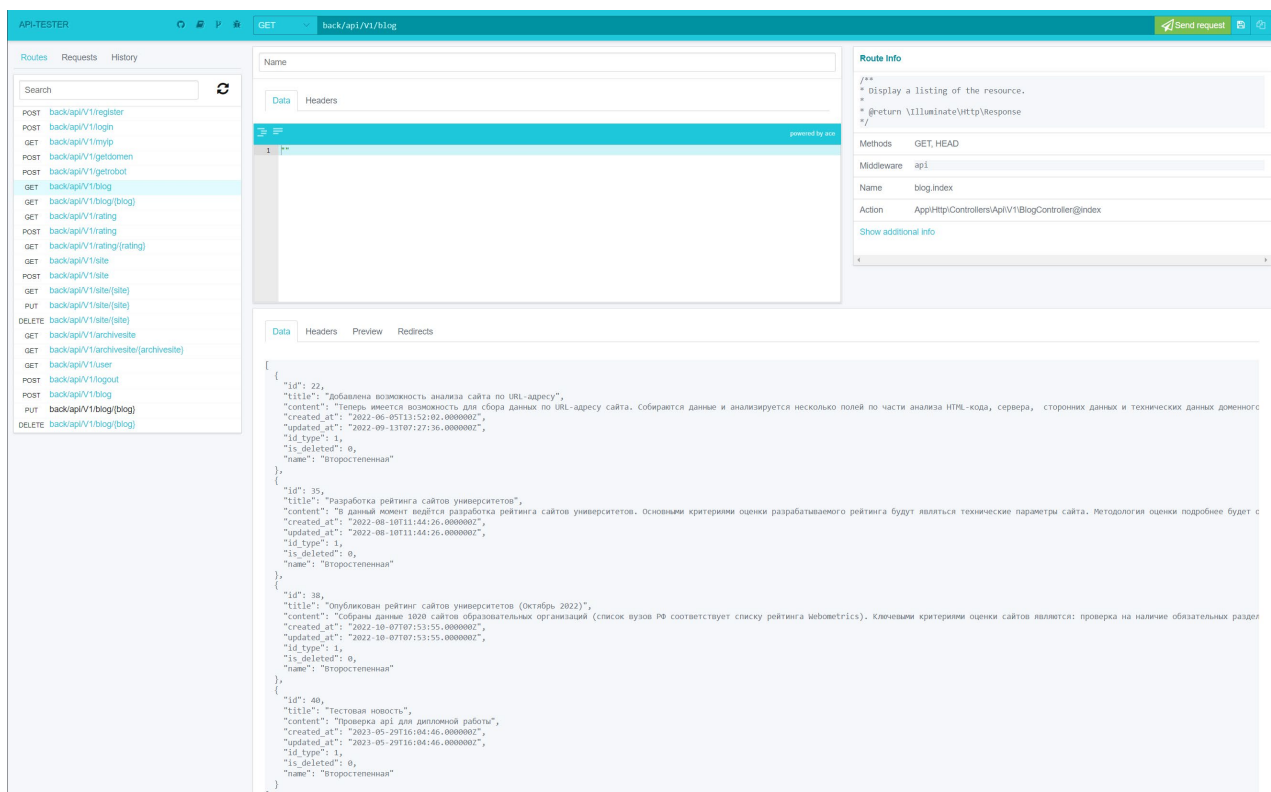


Рисунок 19 – Результат повторного запроса блогов

Также доступно удаление блогов из БД. При удалении блог не удаляется, а лишь помечается удаленным. На рисунке 20 представлен *API* на удаление блога.

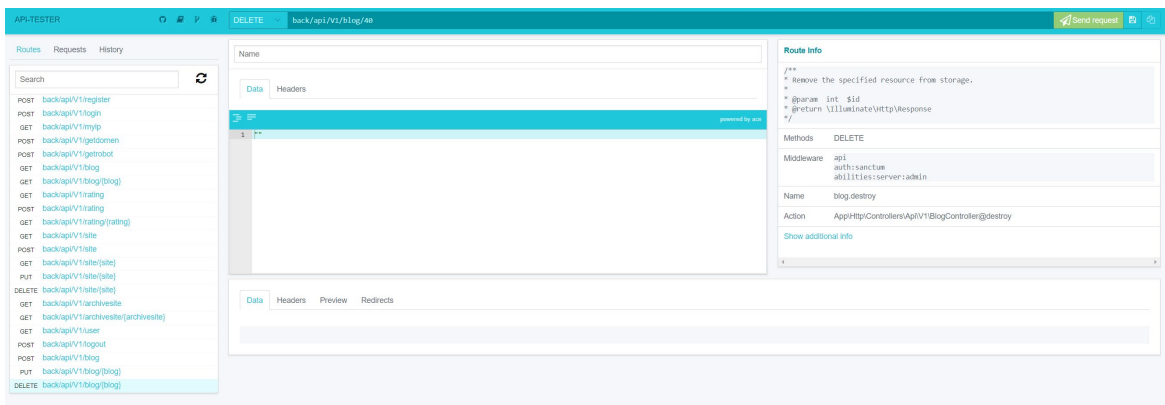


Рисунок 20 – *API* на удаление блога

Проверить правильность работы удаления можно повторно запросив список блогов в системе или посмотреть в БД. Результат повторного запроса списка блогов представлен на рисунке 21, а на рисунке 22 запись удаленного блога в БД.

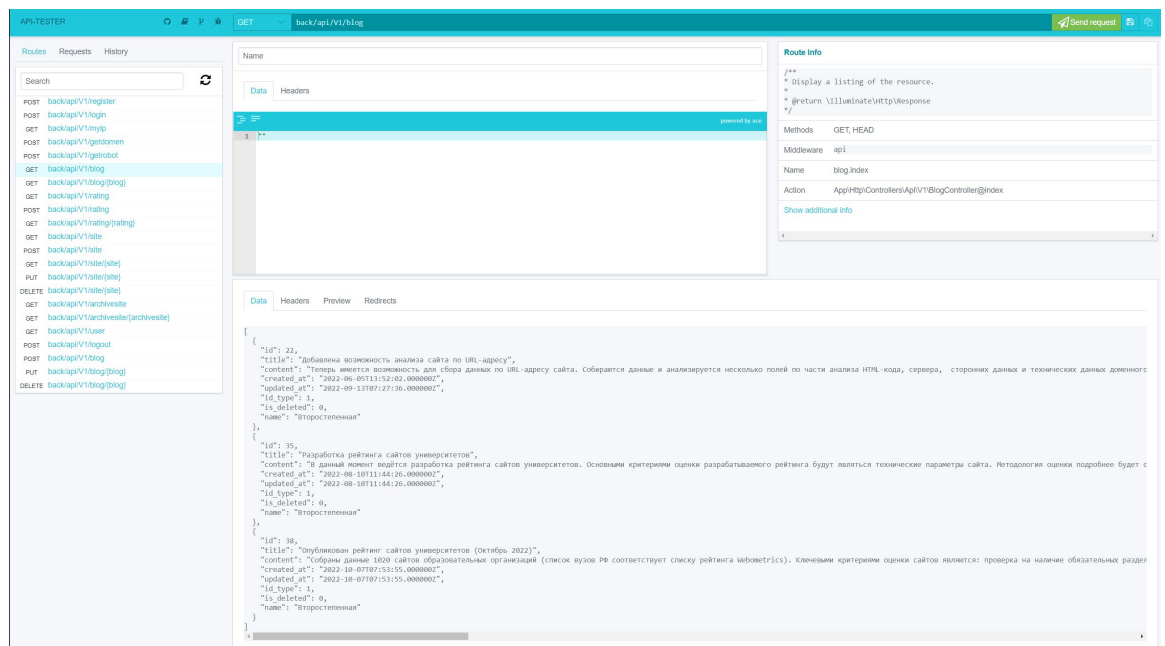


Рисунок 21 – Результат повторного запроса блогов после удаления

	id	title	content	created_at	updated_at	id_type	is_deleted
☐	40	Тестовая новость	Проверка апи для дипломной работы	1685376286	1685377185	1	1

Рисунок 22 – Запись удаленного блога в БД

Теперь попробуем основной механизм анализа интернет-ресурсов, просканируем сайт кафедры. На рисунке 23 представлен результат анализа *vt.pnzgu.ru*.

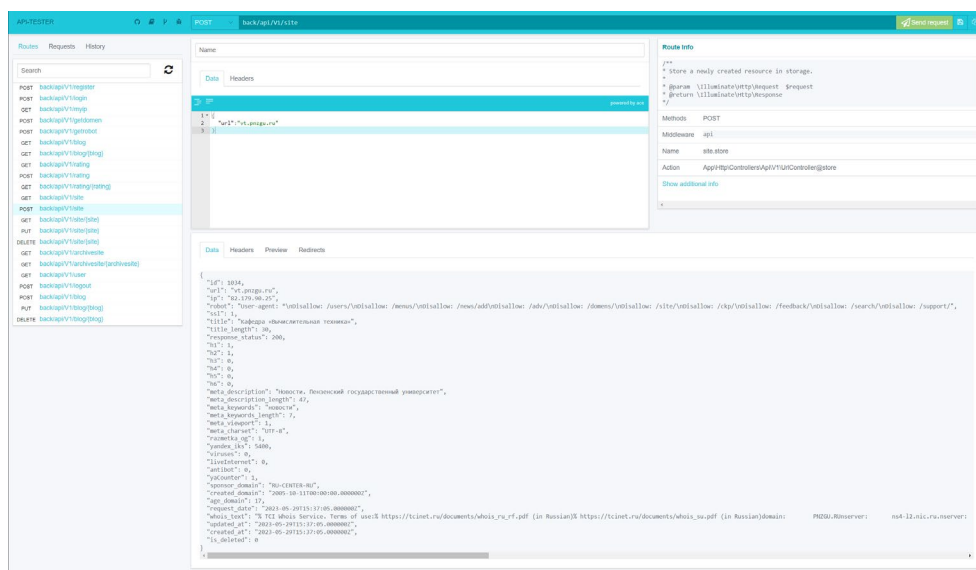


Рисунок 23 – Результат анализа *vt.pnzgu.ru*

Для оценки правильности работы перейдем на страницу <https://vt.pnzgu.ru> и посмотрим ее код через инструменты разработчика в браузере. На рисунке 24 можем увидеть и сравнить такие теги как *title*, *meta_description* и многие другие.

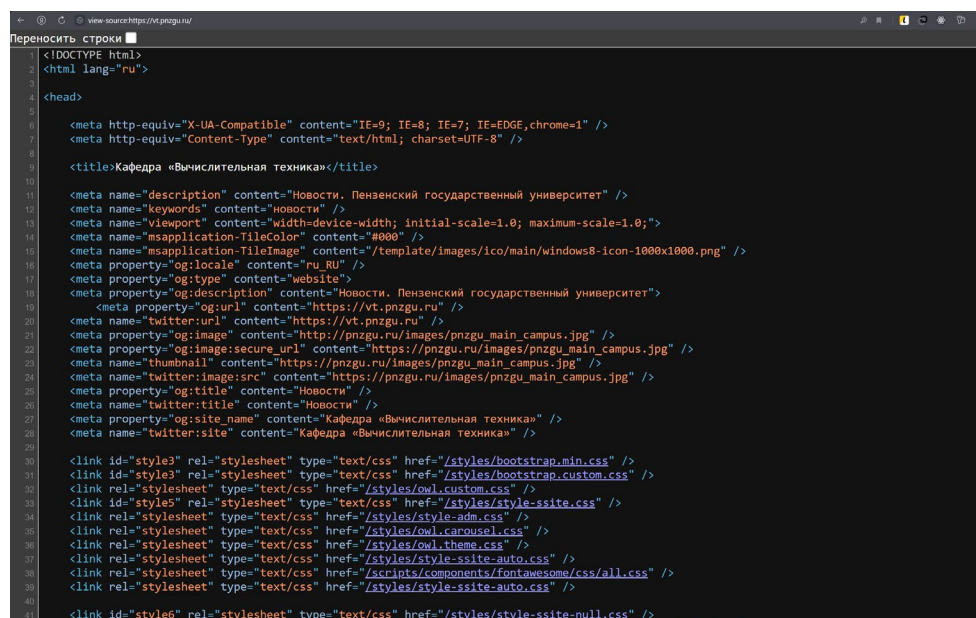


Рисунок 24 – Код страницы *vt.pnzgu.ru*

Можно убедиться, что *backend* приложение и *API* работают верно.

1.3.2 Тестирование интерфейса

В клиентском приложении протестируем основной механизм анализа интернет-ресурсов, и посмотрим рейтинг вузов. Ниже на рисунке 25 представлена главная страница, встречающая пользователей.

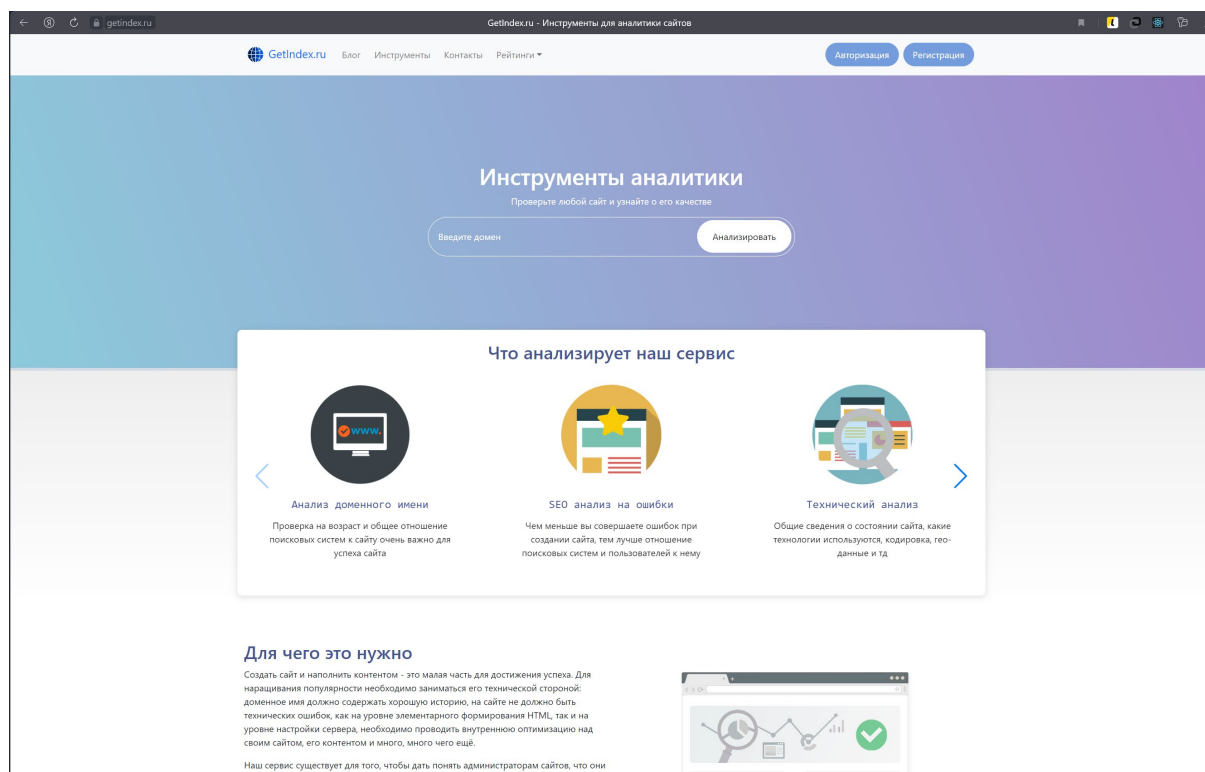


Рисунок 25 – Главная страница

Что бы просканировать интернет-ресурс, нужно ввести его *URL*-адрес и нажать кнопку анализировать. Ниже на рисунках 26, 27, 28, 29 представлены результаты сканирования, они разбиты на разделы: анализ кода страницы, анализ данных сервера и других технических показателей и данные сторонних сервисов.

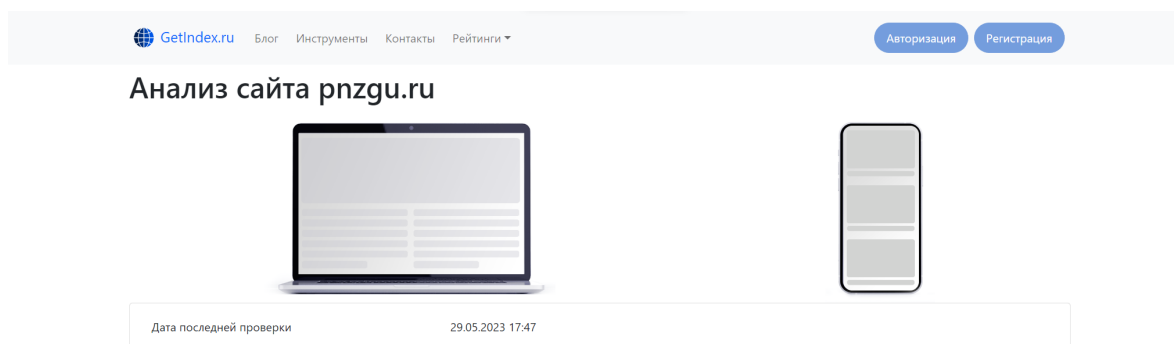


Рисунок 26 – Результаты сканирования

Ter title

Пензенский государственный университет
36 символов

Справка. **Ter title** - обязательный **тег** для HTML-страницы, он используется в поисковых системах, а также в браузерных вкладках. Обязательно размещайте ключевые слова вашего контента в **title**, это положительно воспринимается в поисковых системах. Название вашего сайта писать в **ter title** необязательно, поисковые системы его учитывают. Рекомендуемая длина **ter title** - не более 60-80 символов. Больше писать не рекомендуется - поисковые системы просто обрежут название, а пользователям большие заголовки во вкладке браузера также не понадобятся.

✔ Meta ter description

Новости. Пензенский государственный университет
47 символов

Справка. Мета тег «Description» - это описание страницы для поисковых роботов, прописывается между тегом «head», пользователи не видят его содержимое. Рекомендуемая длина, для поисковой системы Google - 150-300 символов. Для поисковой системы Яндекс - 140-160 символов. Этот показатель косвенно влияет на SEO - через показатель CTR (от англ. «click-through-rate») - показатель кликабельности.

- ✓ Meta ter keywords

7 символов

Справка. Мета-тег «Keywords» – это ключевые слова, описывающие содержание интернет-страницы. Раньше этот тег массово использовался для SEO-продвижения, сейчас его важность заметно снизилась, но он также прописывается между тегом «head».

✔ Наличие OpenGraph микроразметки

имеется

Справка. Микроразметка — это способ оптимизации семантики сайта, который позволяет роботам идентифицировать тип информации на странице и лучше структурировать ее в сниппетах выдачи или в соцсетях.

Open Graph – стандарт микроразметки, который позволяет формировать превью сайта при публикации в социальных сетях.

Стандарт Open Graph был разработан Facebook. Сейчас Open Graph используют Facebook, Яндекс, Вконтакте, Google+, Twitter, Pinterest и др.

✓ Теги H1-H6

Тег H1 имеется

Справка. Заголовки в HTML-коде h1-h6 дают общее понимание структурированности контента для пользователей и поисковых систем. H1 заголовков на странице рекомендуется использовать не более одного раза. Другие H2-H6 заголовки уже в соответствии с контентом представленном на странице. Не добавляйте ссылки на теги h1-h6, это не рекомендуется.

h1: 1	h4: 6
h2: 3	h5: Теги не найдены
h3: Теги не найдены	h6: Теги не найдены

✓ Кодировка

UTF-8

Справка. Мета tag charset. Рекомендуем использовать кодировку UTF-8 для корректного отображения контента на сайте. Некоторые системы управления сайтом (CMS), а также библиотека AJAX поддерживают именно кодировку UTF-8.

- Ter viewport

имеется

Справка. Мета tag viewport отвечает за корректное отображения сайта на кокретных экранах устройств. Большой процент аудитории сегодня именно пользователи мобильных устройств и наличия тега, для правильного отображения сайта обязательно.

Рисунок 27 – Результаты сканирования кода страницы

Анализ данных сервера и других технических показателей

✓ Наличие сертификата SSL (https)
 имеется

Справка. SSL-сертификат – это цифровой сертификат, удостоверяющий подлинность веб-сайта и позволяющий использовать зашифрованное соединение. Отсутствие сертификата ведёт к возможности злоумышленникам перехватить данные вашего сайта, в том числе данные от авторизации: логины, пароли и другие данные.

✓ Код ответа сервера
 200

Справка. Существует 5 групп классов от ответа сервера:

- Информационные 100 - 199
- Успешные 200 - 299
- Перенаправления 300 - 399
- Клиентские ошибки 400 - 499
- Серверные ошибки 500 - 599

Для корректного отображения сайта и анализа в нашем веб-сервисе ваш сайт должен возвращать код сервера - 200.

✓ Наличие файла robots.txt
 имеется

Справка. Robots.txt — это текстовый файл, который содержит параметры индексирования сайта для роботов поисковых систем. Параметры прописываемые в данном файле, как разрешают доступ для роботов сторонних систем, так их и ограничивают. Это файл и эти настройки влияют на индексацию вашего сайта в поисковых системах.

[Показать содержимое файла robots.txt](#)

✓ IP-адрес сервера
 82.179.90.25

✓ Регистратор
 RU-CENTER-RU

✓ Возраст домена (лет)
 17

✓ Дата регистрации домена
 11.10.2005 04:00

Рисунок 28 – Результаты сканирования данных сервера и других технических показателей

Данные сторонних сервисов

✓ Яндекс Вирусы
 Информация о вредоносном коде отсутствует

Справка. Яндекс предлагает проверку страницы сайта на вредоносный код (Safe Browsing). Необязательно, что вы являетесь злоумышленником, возможно, ваш сайт взломали и поместили на эту страницу вредоносный код.

Также вы можете проверить страницу на безопасность непосредственно в [веб-сервисе Яндекс](#)

✓ Яндекс ИКС
 5400

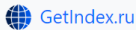
Справка. Поисковая система Яндекс ввела свой собственный параметр для оценки сайта - индекс качества сайта. Индекс качества сайта — это показатель того, насколько полезен ваш сайт для пользователей с точки зрения Яндекса. Чем больше Яндекс ИКС - тем более качественный ваш сайт. Небольшие блоги и сайты одностраничники имеют менее 200 Яндекс ИКС. Информационные порталы, сайты больших изданий и СМИ имеют показатель более 500-1000 Яндекс ИКС.

✗ Сервис статистики liveinternet
 Данные отсутствуют

✓ Сервис статистики Яндекс Метрика
 имеется

Рисунок 29 – Результаты сканирования данных сторонних сервисов

Перейдем на страницу рейтинга вузов. Ниже на рисунке 30 представлен рейтинг вузов, отсортированный по месту в топе.


Блог
Инструменты
Контакты
Рейтинги ▾
Авторизация
Регистрация

Рейтинг сайтов университетов

Цель данного рейтинга, заменить существующий аналог (Ranking Web of Universities, webometrics.info), который прекратил оценку сайтов Российской Федерации с февраля 2022 года.

Ключевыми критериями оценки сайтов являются: проверка на наличие обязательных разделов по требованиям предъявляемым к сайтам образовательных организаций, анализ кода главной страницы, анализ данных сервера, история доменного имени, данные сторонних сервисов и т.д. В следующей версии рейтинга методология оценки будет изменена, за счёт увеличения технических параметров оценки.

Текущая версия: **Октябрь 2022**

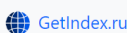
Нашли ошибку, неточные данные или у вас остались вопросы? Напишите нам: adv@getindex.ru

Список оцениваемых сайтов:

Наименование университета	Наименование университета на английском языке	URL-адрес ресурса	Количество обязательных разделов по требованиям Министерства образования РФ	Технический анализ	Общая оценка
Национальный исследовательский университет Высшая школа экономики	National Research University Higher School of Economics	https://www.hse.ru/	15	136	151
Уральский федеральный университет	Ural Federal University	https://urfu.ru/	15	125	140
Московский государственный институт электроники и математики (технический университет)	Moscow State Institute of Electronics and Mathematics	https://miem.hse.ru/	0	128	128
Национальный исследовательский Томский Политехнический Университет	Tomsk Polytechnic University	https://tpu.ru/	15	111	126
Российский университет дружбы народов	Peoples' Friendship University of Russia	https://www.rudn.ru/	15	111	126
Сибирский федеральный университет	Siberian Federal University	http://www.sfu-kras.ru/	15	111	126
Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации	Russian Presidential Academy of National Economy and Public Administration	https://www.ranepa.ru/	15	106	121
Сибирский институт управления Филиал Российской академии народного хозяйства и государственной службы	Siberian Institute of Management Branch Russian Academy of National Economy and Public Administration under the President of the Russian Federation	https://siu.ranepa.ru/	15	106	121
Санкт-Петербургский Политехнический Университет Петра Великого	Peter the Great St.Petersburg Polytechnic University	https://www.spbstu.ru/	15	105	120
Казанский (приволжский) Федеральный Университет	Kazan (Volga Region) Federal University	https://kpfu.ru/	15	105	120
Южно-Уральский государственный университет	South Ural State University	https://www.susu.ru/	15	105	120
Первый Московский государственный медицинский университет И М Сеченова (Сеченовский университет)	I M Sechenov First Moscow State Medical University (Sechenov University)	https://www.sechenov.ru/	15	105	120
Российский государственный университет нефти и газа ИМ Губкина	Gubkin Russian State University of Oil and Gas	https://www.gubkin.ru/	15	104	119
Национальный исследовательский технологический университет	National University of Science & Technology MISIS (Moscow Institute of Steel and Alloys)	https://misis.ru/	15	103	118
Московский технологический университет	Moscow Technological University	https://www.mirea.ru/	15	103	118
Военно-медицинская Академия С М Кирова	Russian Medical Military Academy	https://vmeda.mil.ru/	15	102	117
Военно-космическая академия А Ф Можайского	Military Space Academy A F Mozhaisky	https://vka.mil.ru/	15	102	117
Военная академия связи С М Будённого	Military Academy of Communication	https://vas.mil.ru/	15	102	117
Университет ИТМО	ITMO University	https://itmo.ru/	15	101	116
Южный федеральный университет	Southern Federal University (Rostov State University)	https://sfedu.ru/	15	101	116

Рисунок 30 – Страница рейтинг вузов

По поиску найдем сайты вузов Пензы. Ниже на рисунке 31 представлены результаты поиска.


[Блог](#)
[Инструменты](#)
[Контакты](#)
[Рейтинги ▾](#)

[Авторизация](#)
[Регистрация](#)

Рейтинг сайтов университетов

Цель данного рейтинга, заменить существующий аналог (Ranking Web of Universities, webometrics.info), который прекратил оценку сайтов Российской Федерации с февраля 2022 года.

Ключевыми критериями оценки сайтов являются: проверка на наличие обязательных разделов по требованиям предъявляемым к сайтам образовательных организаций, анализ кода главной страницы, анализ данных сервера, история доменного имени, данные сторонних сервисов и т.д. В следующей версии рейтинга методология оценки будет изменена, за счёт увеличения технических параметров оценки.

Текущая версия: **Октябрь 2022**

Нашли ошибку, неточные данные или у вас остались вопросы? Напишите нам: adv@getindex.ru

Список оцениваемых сайтов:

Наименование университета	Наименование университета на английском языке	URL-адрес ресурса	Количество обязательных разделов по требованиям Министерства образования РФ	Технический анализ	Общая оценка
Пензенский государственный университет	Penza State University	https://www.pnzgu.ru/	15	99	114
Пензенский государственный технологический университет	Penza State Technological University	http://www.penzgtu.ru/	15	78	93
Пензенский государственный университет архитектуры и строительства	Penza State University of Architecture and Construction	http://do.pguas.ru/	15	61	76
Пензенская государственная сельскохозяйственная академия	Penza State Academy of Agriculture	https://pgau.ru/	14	61	75

Рисунок 31 – Рейтинг вузов Пензы

2 Охрана труда

В этом разделе излагаются условия, необходимые для обеспечения безопасности и здоровья пользователей ПК, которые работают с программным обеспечением. Кроме того, здесь будут описаны методы снижения вредных факторов, которые могут негативно сказаться на здоровье.

2.1 Характеристика потенциальных опасностей и вредностей, которые могут возникнуть при работе с ЭВМ

Каждый день персональные электронно-вычислительные устройства помогают людям решать широкий спектр задач, но в то же время они могут вызвать ущерб не только для пользователя, но и для окружающих. Для уменьшения риска вреда, связанного с использованием ПК, важно строго соблюдать рекомендованные правила безопасности, которые могут включать в себя нормирование времени работы на компьютере, оптимальное расположение рабочего места, соответствие уровней освещения и шума внутри помещения установленным нормам.

2.2 Классификация и категорирование помещения, в котором производится разработка

2.2.1 Классификация по взрывопожарной и пожарной опасности

По взрывопожарной и пожарной опасности помещения подразделяются на категории А, Б, В1-В4, Г и Д, а здания - на категории А, Б, В, Г и Д.

Категория помещений и зданий определяется на основании горючих веществ и материалов, а также их пожароопасных свойств и количества. Это также зависит от планировочных решений помещений и характеристик процессов, которые проводятся в них. Ниже приведены категории помещения и характеристик веществ и материалов, находящихся в помещении.

Категория «А» относится к помещениям с повышенной взрывопожароопасностью, где находятся горючие газы и

легковоспламеняющиеся жидкости с температурой вспышки, не превышающей 28 °С. В таких помещениях может образовываться взрывоопасная парогазовоздушная смесь, которая при возгорании может вызвать избыточное давление взрыва, превышающее 5 кПа. Также, в категории «А» включены вещества и материалы, способные взрываться и гореть при взаимодействии с кислородом, воздухом, водой или друг с другом. Если таких веществ и материалов находится в помещении в большом количестве, то может возникнуть избыточное давление взрыва в помещении, превышающее 5 кПа.

Категория «Б» относится к помещениям, где существует взрывопожароопасность из-за наличия горючих пыли и волокон, легковоспламеняющихся жидкостей с температурой воздуха более 28 °С, а также горючих жидкостей, которые могут образовать взрывоопасные пылевоздушные или паровоздушные смеси. Если эти смеси взорвутся, то избыточное давление взрыва в помещении может превысить 5 кПа.

Категории «В1-В4» относятся к группе пожароопасности, где находятся горючие и трудногорючие жидкости, вещества и материалы, которые способны гореть при взаимодействии с водой, кислородом воздуха или друг с другом.

Категория «Г» относится к помещениям с умеренной пожароопасностью, где присутствуют негорючие вещества и материалы в горячем, раскаленном состоянии. Также в этой категории могут находиться помещения, где в процессе обработки этих материалов выделяется лучистое тепло, искры и пламя.

Категория «Д» относится к помещениям с пониженной пожароопасностью, где присутствуют материалы, находящиеся в холодном состоянии, а также негорючие вещества.

2.2.2 Классификация по опасности поражения электрическим током

Условия повышенной опасности поражения людей электрическим током:

– наличие влажной среды в помещении, где выделяется конденсат, в том числе, когда относительная влажность превышает 75%. Это условие способствует усилению проводимости электрического тока, что может увеличить риск поражения электрическим током;

– наличие проводящей пыли, которая может оседать на проводах, внутри аппаратов или машин, создавая небезопасные условия для изоляции и охлаждения. Несмотря на то, что такая пыль не вызывает опасности пожара или взрыва, она может усилить риск поражения электрическим током;

– наличие металлических, земляных, железобетонных или кирпичных оснований, способствующих передаче электрического тока, что увеличивает риск поражения;

– наличие повышенной температуры в помещении, когда температура превышает 35°C продолжительно или 40°C кратковременно, вне зависимости от времени года.

Кроме того, существуют условия особой опасности поражения людей электрическим током, такие как:

– наличие сырости, включающей дождь, снег, частое опрыскивание и покрытие влагой потолка, стен и других поверхностей;

– наличие химически активной среды в помещении, что может увеличить риск поражения электрическим током;

– одновременное наличие двух или более условий повышенной опасности.

Таким образом, на основе этих условий можно выделить три категории опасности помещений: безопасные, с повышенной опасностью и особо опасные.

Для минимизации рисков поражения электрическим током необходимо работать в помещениях, которые соответствуют следующим критериям безопасности:

– температура воздуха должна быть в пределах рекомендованного диапазона от 18°C до 20°C, при этом не должна опускаться ниже 30°C;

– полы должны быть выполнены из диэлектрического материала для уменьшения риска возникновения электрического заряда и выхода изоляции из строя;

– помещение должно регулярно проветриваться для поддержания нормального уровня кислорода и уменьшения вероятности возникновения пожара или короткого замыкания;

– влажность воздуха не должна превышать 60%, так как высокий уровень влажности может привести к образованию конденсата на электрических приборах и аппаратах, что увеличивает риск поражения электрическим током.

2.3 Санитарно-гигиеническая характеристика производства

Необходимые санитарно-гигиенические стандарты включают в себя такие параметры, как сухость помещения, температура и количество света. Для обеспечения безопасных условий работы, существуют определенные стандарты, которые необходимо соблюдать, а именно:

- каждое рабочее место должно быть в объеме не менее 15 м³;
- высота помещения не менее 3,2 м;
- ширина дверей не менее 0,8 м;
- ширина коридоров не менее 1,4 м.

Окна должны быть расположены таким образом, чтобы максимально обеспечивать естественное освещение рабочих мест. Каждое окно должно открываться и регулярно очищаться от грязи и пыли. В случае недостаточного освещения естественным светом, помещение должно быть оснащено электрическими осветительными приборами. Фонари должны быть изготовлены из армированного стекла и защищены металлическими сетками. Оптимальный диапазон температур в помещении - от 18°C до 21°C, при влажности от 40% до 60%.

Руководители производства должны обеспечить своих работников чистой питьевой водой, и рекомендуется ежедневно менять воду. Помещения, в которых уровень шума превышает норму, должны быть оснащены специальными знаками. Каждый работник должен быть обучен и знать правила безопасности при работе в шумных условиях.

					ПГУ1.09.03.01.01.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

Помещения, где уровень шума превышает установленную норму, должны быть обязательно маркированы специальными знаками, чтобы предупредить работников о возможных рисках и предупредить их принимать необходимые меры предосторожности. Каждый работник, который работает в шумных условиях, должен быть проинструктирован и обязан знать правила безопасности при работе в таких условиях. Он должен быть осведомлен о необходимости использования средств индивидуальной защиты, таких как наушники или слуховые протекторы, чтобы защитить свой слух от повреждения. Работники также должны знать, как правильно использовать эти средства и следить за их состоянием, чтобы эффективно уменьшить воздействие шума на слух.

2.4 Расчет условий зрительной работы в производственном помещении, где установлены ПЭВМ

Условия зрительной работы в помещении, где используются персональные электронные вычислительные машины (ПЭВМ), являются важным вопросом в области охраны труда и безопасности в производственном окружении. В данном разделе ВКР необходимо произвести расчет условий зрительной работы в производственном помещении с установленными ПЭВМ.

Для расчета условий зрительной работы в производственном помещении с ПЭВМ были выбраны следующие параметры:

- площадь помещения - 100 кв. м.;
- количество ПЭВМ - 10 шт.;
- уровень освещенности - 500 лк.

Для определения условий зрительной работы необходимо рассчитать следующие показатели:

- нормативный уровень освещенности;
- коэффициент естественного освещения;
- коэффициент использования освещения;

- коэффициент тени;
- коэффициент блеска;
- коэффициент контрастности;
- расчет величин абсолютной и относительной пространственной яркости.

Для проведения расчета была выбрана методика, основанная на рекомендациях ГОСТ 12.2.003-91 "Эргономические требования к работам на персональных электронных вычислительных машинах". Ниже приведены таблицы 2, 3, 4 из вышеуказанного ГОСТа.

Таблица 2 – Уровень освещенности

Категория сложности работы	Уровень освещения, лк	Предельно допустимое значение, лк
Легкая	400-500	750
Средняя	300-400	500
Тяжелая	200-300	300

Таблица 3 – Коэффициент тени

Категория сложности работы	Коэффициент тени, %	Предельно допустимое значение, %
Легкая	30	50
Средняя	20	30
Тяжелая	10	20

Таблица 4 – Коэффициент блеска

Категория сложности работы	Коэффициент блеска, %	Предельно допустимое значение, %
Легкая	15-20	40
Средняя	10-15	30
Тяжелая	5-10	20

Нормативный уровень освещенности для производственного помещения с ПЭВМ составляет 300-500 лк, таким образом, условия зрительной работы в данном помещении соответствуют норме.

Коэффициент естественного освещения (КЕО) - отношение естественного светового потока, попадающего в помещение, к световому потоку, обеспечиваемому искусственным освещением:

КЕО = 0%, так как в описанном помещении искусственные источники света являются основными.

Коэффициент использования освещения (КИО) - отношение светового потока, попадающего на рабочую поверхность, к общему световому потоку:

КИО = 40%, при условии использования люминесцентных ламп (70 лм/Вт) или светодиодных светильников (100 лм/Вт) и установки их на оптимальной высоте (2-2,5 метра от пола).

Коэффициент тени (КТ) - отношение мощности теней к общей мощности светового потока:

КТ = 10%, при условии расположения компьютеров на достаточном расстоянии друг от друга, чтобы избежать попадания теней от одной рабочей зоны на другую.

Коэффициент блеска (КБ) - отношение светового потока, отраженного от блестящих поверхностей, к общему световому потоку на рабочей поверхности:

КБ = 5%, при условии использования матовых поверхностей.

Коэффициент контрастности (КК) - отношение яркостей самых ярких и самых темных участков рабочей поверхности:

КК = 10:1.

Расчет величин абсолютной и относительной пространственной яркости – это дополнительное измерение освещенности, требующее специальной аппаратуры и необходимое при проектировании сложных производственных помещений.

Из расчетов следует, что условия зрительной работы в производственном помещении с ПЭВМ соответствуют нормам, установленным для данного типа

помещений. Показатель освещенности превышает минимальный порог (500 лк), установленный для комфортной визуальной работы за ПЭВМ. При этом, при правильной установке светильников и оптимальной высоте их размещения, энергопотребление может быть снижено. Коэффициенты тени и блеска также находятся в безопасных границах. Коэффициент контрастности является оптимальным для работы за монитором. Таким образом, работа в данном производственном помещении с ПЭВМ будет комфортной и безопасной для зрительного аппарата.

					ПГУ1.09.03.01.01.001 ПЗ	Лист
						64
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе реализации данной работы было изучено основы *SEO* анализа, настройка *VPS*, методы сбора и структурирования информации об интернет-ресурсах.

Также были получены навыки разработки *WEB*-приложений с использованием фреймворка *Laravel* на языке *PHP*, и *JavaScript*-библиотеки *React*. Были получены навыки работы с *MariaDB*, навыки разработки собственного *API* и *backend* и *frontend* приложений как два отдельных проекта, взаимодействующих по *REST API* и соответствующие паттерну *MVC*. Также были усовершенствованы навыки разработки программ с использованием объектно-ориентированного подхода.

Во время выполнения данной работы были разработаны: алгоритм сбора и структурирования информации об интернет-ресурсах, клиент-серверное веб-приложение, с разделением клиентской и серверной части на два приложения с применением *REST API*. Также была спроектирована БД для хранения полученных и обработанных данных. И все это было реализовано на *VPS* и выложено в общий доступ.

В дальнейшей разработке данной программы будут:

- добавлен механизм сканирования страницы в виде графического изображения с видом как на ПК, так и мобильном устройстве;
- решен вопрос с получением таких данных, как визиты, просмотры, отказы и т.п.;
- реализованы отдельные инструменты, включенные в общий анализ;
- добавлены различные рейтинги;
- добавлены механизмы подписки для автоматизированного сканирования ресурса с в соответствии с заданным графиком;
- добавлен механизм сравнения ресурсов между собой.

Список используемых источников

1. Шамина И.С., Носаченко А.В. «*Total SEO*. Полное практическое руководство по продвижению сайтов», 2021. – 604с.
2. Энж Эрик, Стрикчиола Джесси, Спенсер Стефан. «*SEO: искусство раскрутки сайтов*», 2017. – 816с.
3. Программные продукты *PR-CY*, *MegaIndex*, *Majestic*, *SimilarWeb* - руководства пользователя, статьи.
4. Документация к фреймворку *Laravel*. [Электронный ресурс] / Режим доступа: <https://laravel.su/docs/8.x/> свободный. (дата обращения 29.05.2023).
5. Документация по веб серверу Nginx: [Электронный ресурс] / Режим доступа: <https://nginx.org/> свободный. (дата обращения 29.05.2023).
6. Документация к библиотеке *React*. [Электронный ресурс] / Режим доступа: <https://ru.react.js.org/docs/getting-started.html> свободный. (дата обращения 29.05.2023).
7. Никсон Робин. «Создаем динамические веб-сайты с помощью *PHP*, *MySQL*, *JavaScript*, *CSS* и *HTML5*», 2019. – 816с.
8. Документация по *VPS* серверам джино. [Электронный ресурс] / Режим доступа: <https://jino.ru/spravka/vps/#vps-articles> свободный. (дата обращения 29.05.2023).

Приложение А

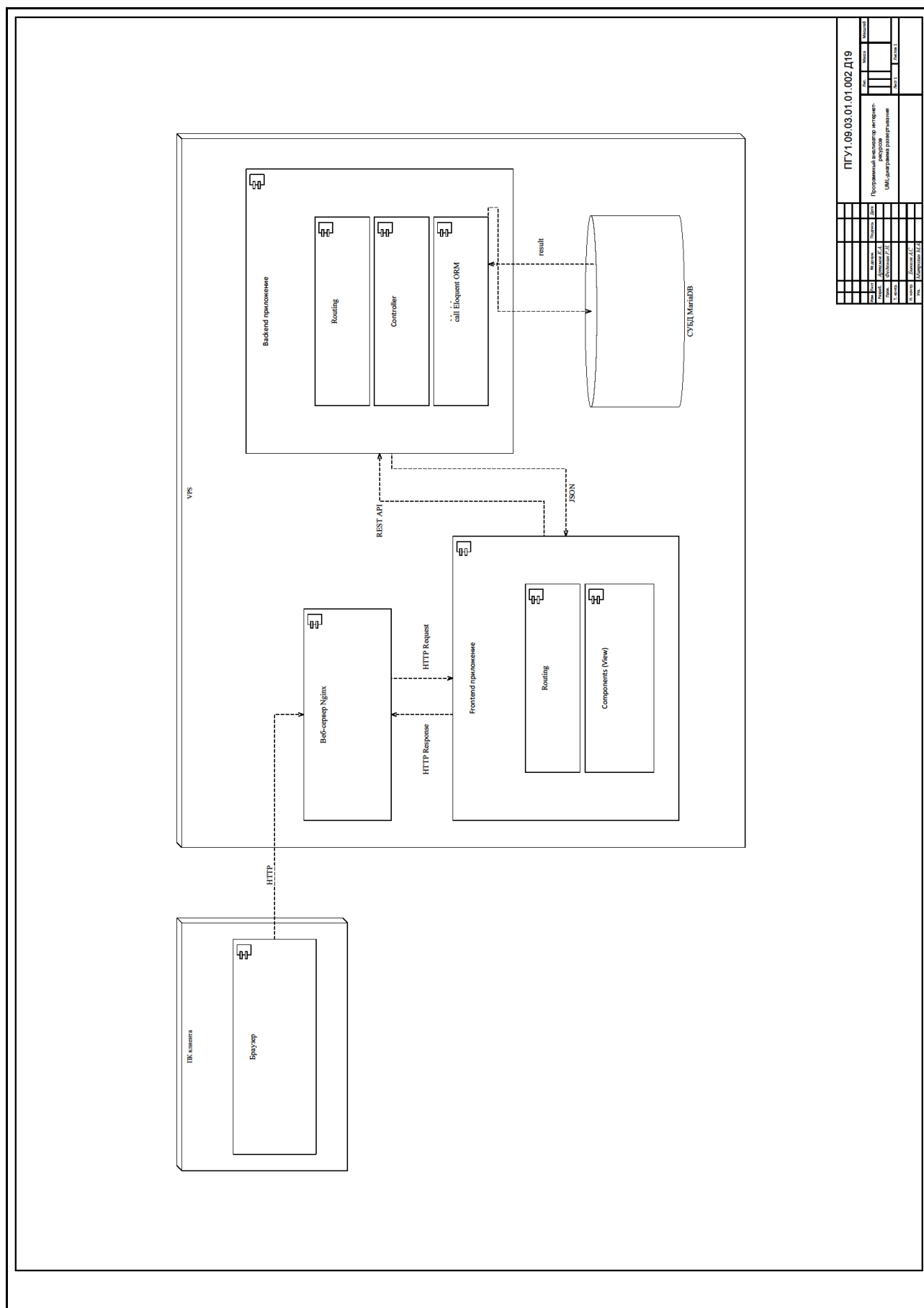
(Обязательное)

Программный анализатор интернет-ресурсов

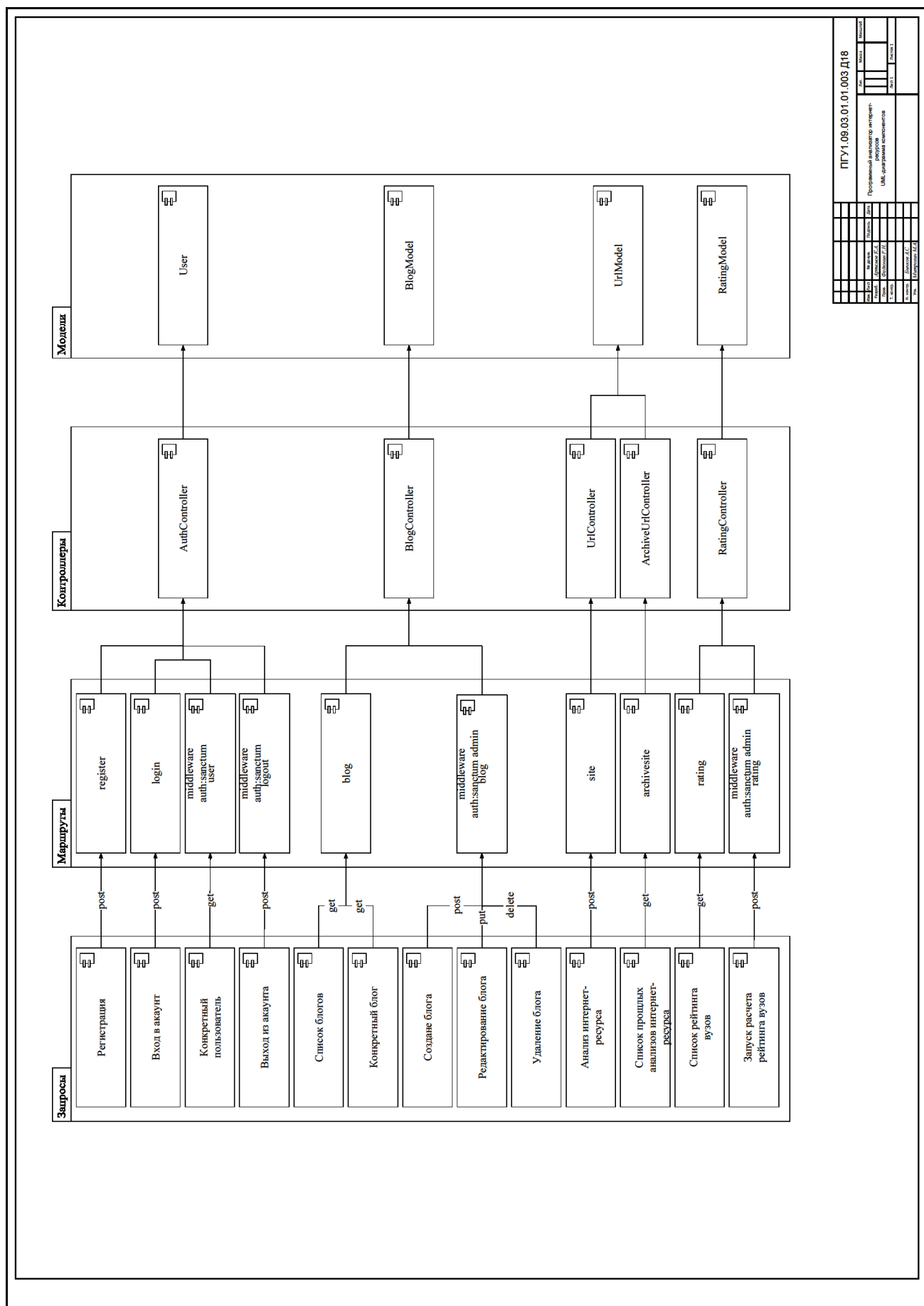
UML-диаграммы

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		67

Приложение А.1 – Диаграмма развертывания

[illegible]

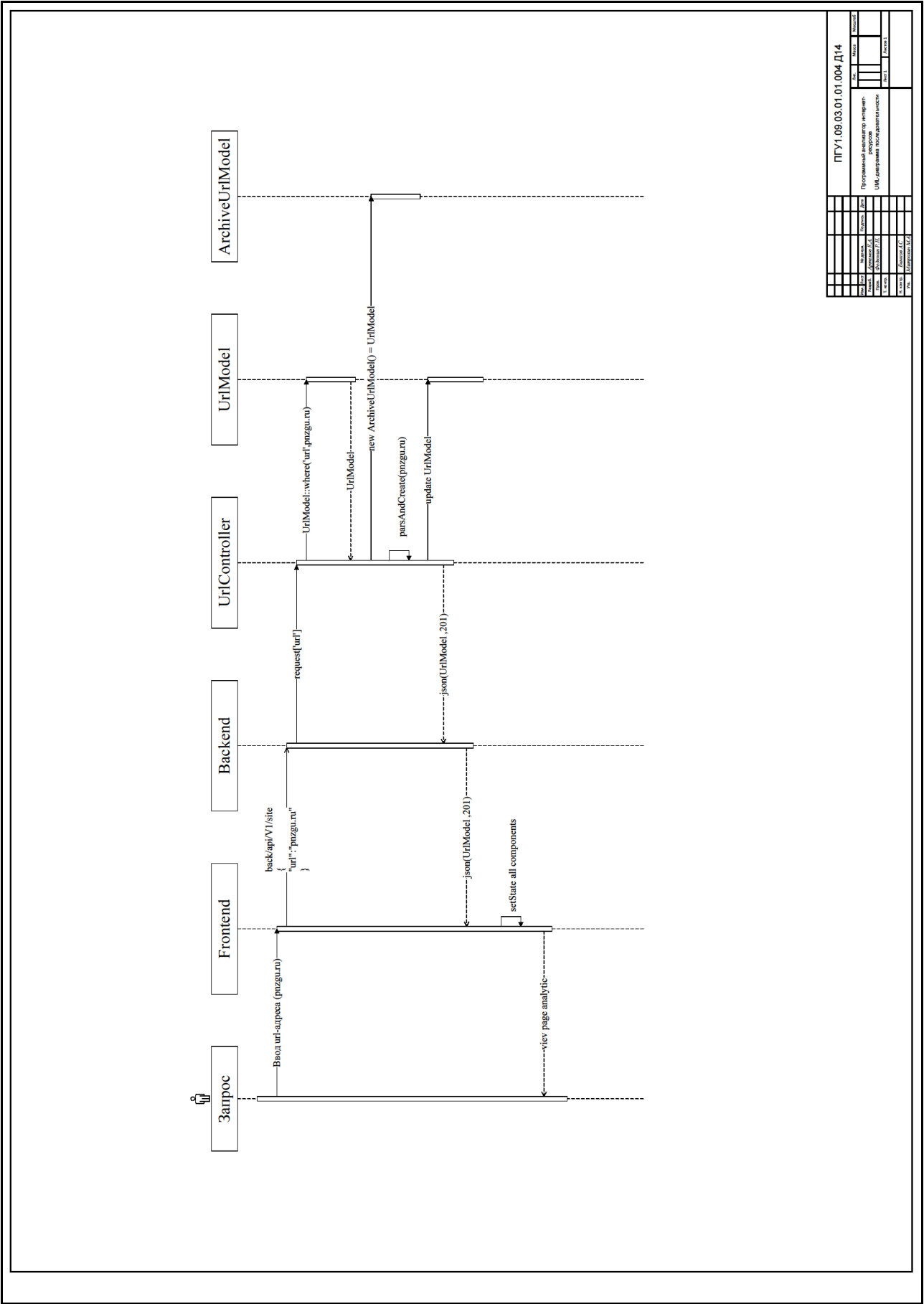
Приложение А.2 – Диаграмма компонентов



ПГУ1.09.03.01.01.003 Д18									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Иванов	Иван	Петров	Петр	Сидоров	Сидор	Климов	Клим	Васильев	Василий
Программный инженер интернет-ресурсов									
ИМ, диаграмма компонентов									
Лист 1									
Итого: 10 стр.									

Изм.	Лист	№ докум.	Подпись	Дата

Приложение А.3 – Диаграмма последовательности



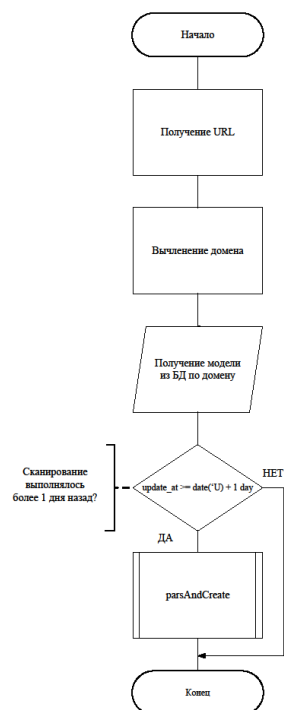
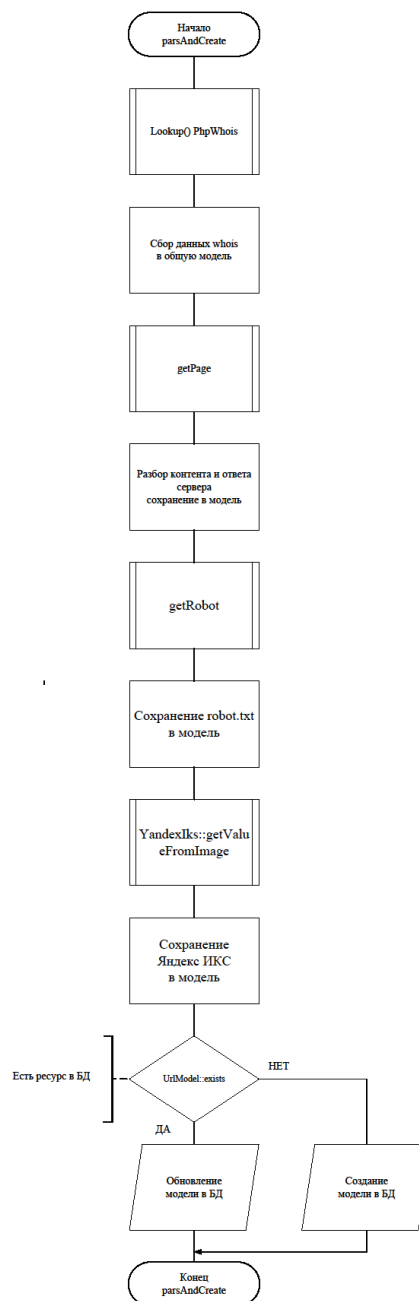
Приложение Б

(Обязательное)

Программный анализатор интернет-ресурсов
ЕСПД

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		71

Приложение Б.1 – Схемы программ



ПГУ1.09.03.01.01.005 Д2									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.09.03.01.01.001 ПЗ

Лист

72

Приложение В

(Обязательное)

Программный анализатор интернет-ресурсов

Листинг программы

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		73

Файл App.jsx

```
import './App.css';
import React, { useEffect, useState } from 'react';
import { BrowserRouter, Routes, Route } from "react-router-dom";
import 'bootstrap/dist/css/bootstrap.min.css';
...
return (

  <BrowserRouter>
    <div className="wrapper d-flex flex-column justify-content-between">
      <Header />
      <div className="flex-grow-1">
        <Routes>
          <Route path="/" element={<Main />} />
          <Route path="/blog" element={<Blog />} />
          <Route path="/contact" element={<Contact />} />
          <Route path="/tools" element={<Tools />} />
          <Route path="/register" element={<Register />} />
          <Route path="/login" element={<Login />} />
          <Route path="/analytic" element={<PageAnalytics />} />
          <Route path="/profile" element={<Profile />} />
          <Route path="/myip" element={<MyIp />} />
          <Route path="/rating" element={<Rating />} />
          <Route path="*" element={<Page404 />} />
        </Routes>
      </div>
      <Footer />
    </div>
  </BrowserRouter>
)
```

Файл Main.jsx

```
import React, { useState } from 'react';
import { Navigate } from "react-router-dom";
import { useDispatch, useSelector } from 'react-redux';
import { setCount } from '../reducers/reposReducer';

const Main = () => {

  const dispatch = useDispatch()
  const count = useSelector(state => state.repos.count)

  const [url, setSite] = useState('');
  const [redirect, setRedirect] = useState(false);
```

```

const submit = async (e) => {
  e.preventDefault();
  const response = await
fetch('https://getindex.ru/back/api/V1/site', {
  method: 'POST',
  headers: { 'Content-Type': 'application/json' },
  body: JSON.stringify({
    url
  })
});
  if (response.ok) {
    const contentt = await response.json();
    console.log(contentt);
    if (contentt.response_status !== 404) {
      dispatch(setCount(url));
      setRedirect(true);
    } else {
      alert("Данного URL-адреса не существует.
Ошибка:" + contentt.response_status);
    }
  } else {
    alert("Сайт недоступен для сканирования. Ошибка:" +
response.status);
  }
}

if (redirect) {
  return <Navigate to="/analytic" />
}

return (
  ...
  <form onSubmit={submit}>
    <input onChange={e => setSite (e.target.value)}
id="floatingSite" type="text" />
    ...
    <button type="submit"></button>
  </form>
  ...
);

```

Файл PageAnalytics.jsx

```

import React from 'react';
import { useSelector } from 'react-redux';
import { Navigate } from "react-router-dom";

const PageAnalytics = () => {
  const count = useSelector(state => state.repos.count)
  if (!count) {
    return <Navigate to="/" />
  }
  return (

```

```

        <div className="container analytics">
            <h1>Анализ сайта {count}</h1>
            <PageAnalyticsStats />
            <PageAnalyticsHtml />
            <PageAnalyticsServer />
            <PageAnalyticsService />
        </div>

    );
};

    Файл PageAnalyticsStats.jsx

import React, { useEffect, useState } from 'react';
import { useSelector } from 'react-redux';
import Moment from 'react-moment';

const PageAnalyticsStats = () => {
    const count = useSelector(state => state.repos.count);
    const [updated_at, setUpdatedat] = useState('');
    useEffect(() => {
        (
            async () => {
                const response = await
fetch('https://getindex.ru/back/api/V1/site?url=' + count, {
                    method: 'POST',
                    headers: { 'Content-Type':
'application/json' },
                    body: JSON.stringify({
                        updated_at
                    }),
                    credentials: 'include',
                });

                const content2 = await response.json();
                setUpdatedat(content2.updated_at);
            }
        )();
    });

    return (
        ...
        <div className="d-flex align-items-center col-12 col-md-
4">
            Дата последней проверки
        </div>
        <div className="col-12 col-md-8">
            <Moment format="DD.MM.YYYY HH:mm">{updated_at ?
updated_at : <span className="text-danger">Данные
отсутствуют</span>}</Moment>
        </div>
        ...
    );
};

```

Файл .example

```
APP_NAME=back
APP_KEY= ****
APP_DEBUG=true
APP_URL=https://getindex.ru/back
```

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=seo_site
DB_USERNAME=*****
DB_PASSWORD=*****
```

Файл routes/api.php

```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::post('register', [AuthController::class, 'register']);
Route::post('login', [AuthController::class, 'login'])->name('login');
Route::get('myip', [InstrumentsController::class, 'myip']);
Route::post('getdomen', [InstrumentsController::class, 'getdomen']);
Route::post('getrobot', [InstrumentsController::class, 'getrobot']);
Route::apiResources([
    'blog'=>BlogController::class,
], ['only' => ['index', 'show']]
);
Route::apiResources([
    'rating'=>RatingController::class,
], ['only' => ['index', 'show', 'store']]);
Route::apiResources([
    'site'=>UrlController::class,
]);
Route::apiResources([
    'archivesite'=>ArchiveUrlController::class,
], ['only' => ['index', 'show']]
);
Route::middleware(['auth:sanctum'])->group(function() {
    Route::get('user', [AuthController::class, 'user']);
    Route::post('logout', [AuthController::class, 'logout']);
});
Route::middleware(['auth:sanctum', 'abilities:server:admin'])->group(function() {
    Route::apiResources([
        'blog'=>BlogController::class,
    ], ['only' => ['store', 'update', 'destroy']]);
});
```

Файл UrlController.php

```
namespace App\Http\Controllers\Api\V1;
use App\Http\Controllers\Controller;
use App\Models\UrlModel;
use Illuminate\Http\Request;
```

```

class UrlController extends Controller
{
    public function store(Request $request)
    {
        if(!str_contains($request['url'], '://'))
        $search_url="http://".$request['url'];
        else $search_url=$request['url'];
        // Получение домена по url
        $domain = parse_url($search_url, PHP_URL_HOST);
        $url =UrlModel::where('url', $domain)->first();
        if (is_null($url) || $url->updated_at->format('U')<
date('U',strtotime("-1 day"))){
            $url = new UrlModel();
            $url = $url->parsAndCreate($domain);
        }
        if($url!=false)
            return response()->json($url,201);
        else
            return response()->json(['error'=> true,
'message'=>'Not found'], 501);
    }
    public function show($url)
    {
        if(!str_contains($url, '://')) $search_url="http://".$url;
        else $search_url=$url;
        // Получение домена по url
        $domain = parse_url($search_url, PHP_URL_HOST);
        $url =UrlModel::where('url', $domain)->first();//
        if($url!=false)
            return response()->json($url,201);
        else
            return response()->json(['error'=> true,
'message'=>'Not found'], 501);
    }
}

```

Файл UrlModel.php

```

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use phpwhois\phpwhois\src;
use Whois;

class UrlModel extends Model
{
    use HasFactory;
    protected $table = "url_models";
    protected $dateFormat = 'U';
    protected $dates = [
        'created_at',
        'updated_at',
        'request_date',
        'created_domain',
    ];
}

```

```

//public $timestamps = false;
protected $fillable = [
    'id',
    'url',
    'ip',
    'robot',
    'ssl',
    'title',
    'title_length',
    'response_status',
    'h1',
    'h2',
    'h3',
    'h4',
    'h5',
    'h6',
    'meta_description',
    'meta_description_length',
    'meta_keywords',
    'meta_keywords_length',
    'meta_viewport',
    'meta_charset',
    'razmetka_og',
    'yandex_iks',
    'viruses',
    'liveInternet',
    'antibot',
    'created_at',
    'updated_at',
    'is_deleted',
    'yaCounter',
    'sponsor_domain',
    'created_domain',
    'age_domain',
    'request_date',
    'whois_text',
];

public function parsAndCreate($domen){
    $sponsor=NULL;
    $ip=NULL;
    $created=NULL;
    if(!is_null($domen)){
        $domen2 = str_replace("www.", "", $domen);
        $whois = new Whois;
        $result = $whois->lookup ($domen2,false);

        ...
        // Основной парсинг страницы по url
        $html = $this->getPage(['url'=>$domen]);
        // ip
        if(array_key_exists('primary_ip',$html['data']['info'])){
            $ip=$html['data']['info']['primary_ip'];
        }
        // Взятие файла robot.txt

```

```

        $robot = $this->getRobot($domain);
        // Определение ssl сертификата
    ...
        // Статус ответа
        $response_status =
$html['data']['info']['http_code'];
        // Подсчет заголовков
    ...
        //Содержимое тегов meta
    ...
        // Наличие Микроразметки Schema.org или OpenGraph
(OG)
    ...
        // Наличие счетчика "LiveInternet" или "yadro.ru"
    ...
        // Наличие счетчика "yandex.ru/metrika" или
"yaCounter"
    ...
        // яндекс икс
        $yandex_iks =
YandexIks::getValueFromImage($domain);
        if(!$yandex_iks){
            $yandex_iks =
YandexIks::getValueFromImage($domain2);
            if(!$yandex_iks)
                $yandex_iks= 0;
        }
        //проверка на вирусы Yandex Safe Brouser
        if($ssl == 1){
            $ysf=$this-
>isYandexSafeBrowsing(['url'=>"https://".$domain]);
        }else{
            $ysf=$this-
>isYandexSafeBrowsing(['url'=>"http://".$domain]);
        }
        $url_model = $this::where('url', $domain)->first();
        if (is_null($url_model)){
            return $this::create([
                ...
            ]);
        }else{
            $archive_url = new ArchiveUrlModel();
            $copy_url_model=$url_model->original;
            unset($copy_url_model['id']);
            $archive_url ->create($copy_url_model);
            $url_model -> update([
                ...
            ]);
            return $url_model;
        }
    }
    return false;
}

```

```

    }
    public function getRobot($url){
        $robotsUrl = "http://".$url ."/robots.txt";
        $robotostxt =@file_get_contents($robotsUrl);
...
        return $robotostxt;
    }
    private static function getPage($params = []){
        if($params){
            if(!empty($params["url"])){
                $url = $params["url"];
                $ch = curl_init();
                curl_setopt($ch, CURLOPT_URL, $url); // отправляем
на
                curl_setopt($ch, CURLOPT_RETURNTRANSFER, true); //
возвратить то что вернул сервер
...
                $content = curl_exec($ch);
                $info      = curl_getinfo($ch);
                $error = false;
                if($content === false){
                    $data = false;
                }else{
                    $data["content"] = $content;
                }
                curl_close($ch);
                return [
                    "data"      => $data,
                    "error" => $error
                ];
            }
        }
        return false;
    }
    private static function isYandexSafeBrowsing($params = []){
        if($params){
            if(!empty($params["url"])){
                $url = $params["url"];
                $curl = curl_init();
                curl_setopt_array($curl, array(
CURLOPT_URL =>
"https://sba.yandex.net/v4/threatMatches:find?key=****",
CURLOPT_RETURNTRANSFER => true,
CURLOPT_ENCODING => "",
CURLOPT_MAXREDIRS => 10,
CURLOPT_TIMEOUT => 30,
CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
CURLOPT_CUSTOMREQUEST => "POST",
CURLOPT_POSTFIELDS => ' {
    "client": {
        "clientId":      "getindex.ru",
        "clientVersion": "0.1"
    },

```



```

    $iks_y = 10;
    $iks_w = 52;
    $iks_h = 11;
    // сюда вырезается нужная часть изображения
    $dst_img = imagecreatetruecolor($iks_w, $iks_h);
    // нужна часть изображения, на которой могут быть цифры
    imagecopy($dst_img, $src_img, 0, 0, $iks_x, $iks_y,
    $iks_w, $iks_h);
    $arr = [];
    for ($i = 0; $i < $iks_w; ++$i) {
        $arr[$i] = '';
        for ($j = 0; $j < $iks_h; ++$j) {
            $arr[$i] .= 5855577 == imagecolorat($dst_img, $i,
    $j) ? '.' : 'x';
        }
    }
    // подчистить пустые ряды
    for ($i = 0; $i < $iks_w; ++$i) {
        if ('.....' == $arr[$i]) unset($arr[$i]);
    }
    $iks = '';
    $current_symbol = '';
    $current_len = 0;
    foreach ($arr as $v) {
        $current_symbol .= $v;
        $current_len += 11;
        if (88 == $current_len) { // все символы имеют ширину
8
            foreach (self::$reference as $num => $symb) {
                if (similar_text($symb, $current_symbol) > 80)
{
                    $iks .= $num;
                    break;
                }
            }
            $current_symbol = '';
            $current_len = 0;
        } elseif (55 == $current_len) { // кроме 1 – у него
ширина 5
            if (similar_text(self::$reference[1],
    $current_symbol) > 50) {
                $iks .= 1;
                $current_symbol = '';
                $current_len = 0;
            }
        }
    }
    return $iks;
}
}

```

Приложение Г

(Обязательное)

Программный анализатор интернет-ресурсов

Презентация

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		84

Программный анализатор интернет-ресурсов

Автор работы:

студент гр. 19BBB1 Артемов Кирилл Артемович

Научный руководитель:

к.т.н., доцент Федюнин Роман Николаевич

Пенза, 2023

Цель работы и объекты

Цель работы – разработать клиент-серверную приложение для сбора, структурирования и анализа информации интернет-ресурсов

Объектом проекта является клиент-серверное приложение, выполняющие следующие функции:

- сбор данных об интернет-ресурсе;
- анализ собранных данных;
- передача данных от программы-сервера программе клиенту посредством API;
- получение и отображение данных хранящихся на сервере через клиентскую программу.

Зачем анализировать интернет-ресурсы?

Программные анализаторы используются в различных областях:

- маркетинговые исследования
- сбор данных для исследований
- разведка конкурентной информации
- автоматизированный анализ структуры веб-сайтов

Программный анализатор полезен как для собственников сайтов, так и для интернет-маркетологов, SEO-специалистов, аналитиков и разработчиков.

Также анализатор является важным аспектом в области информационной безопасности.

Сравнение аналогов

Анализатор	Функционал	Удобство использования	Точность анализа	Стоимость подписок
<u><i>Pr-cy</i></u>	Анализ трафика, ссылок, ключей	Простой интерфейс	Достаточно точный	От \$15/месяц
<u><i>MegaIndex</i></u>	Анализ конкурентов, рекламы	Простой и прямой	Достаточно точный	От \$12.9/месяц
<u><i>Majestic</i></u>	Анализ профиля ссылок	Интуитивный интерфейс	Достаточно точный	От \$49.99/месяц
<u><i>SimilarWeb</i></u>	Оценка потенциального трафика	Простой интерфейс	Достаточно точный	От \$199/месяц

Используемые технологии



ДЖИНО



React



Redux



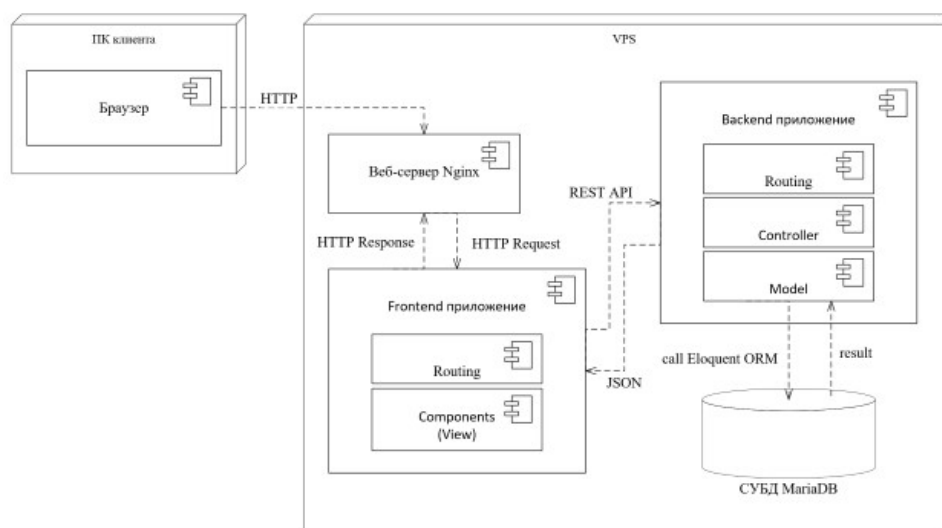
NGINX



MariaDB



Архитектура приложения



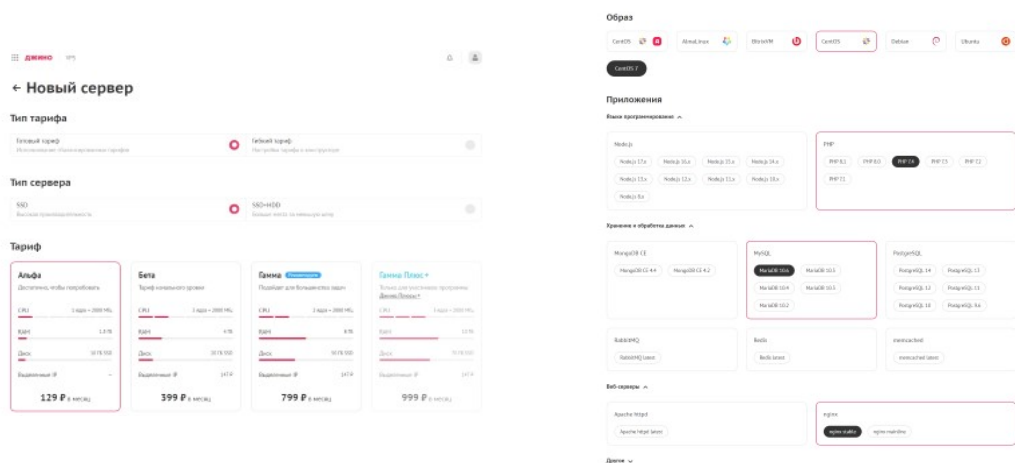
Изм.	Лист	№ докум.	Подпись	Дата

ПГУ1.09.03.01.01.001 ПЗ

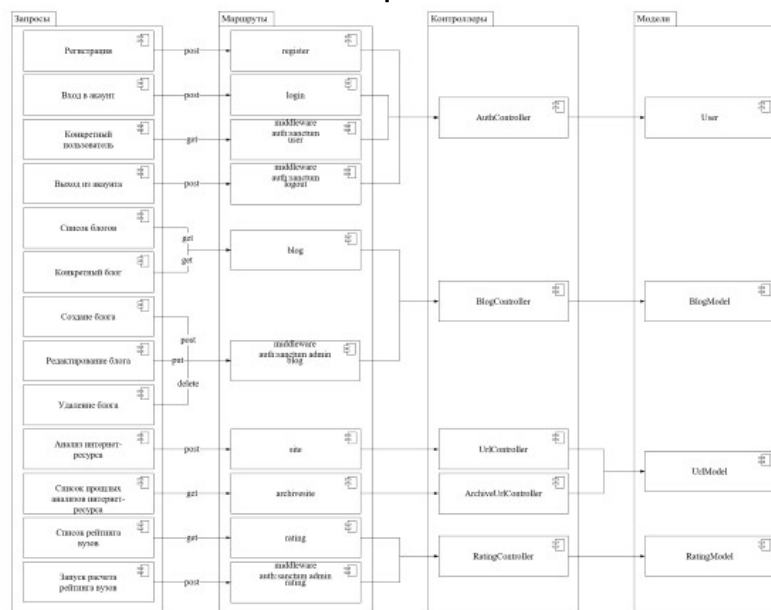
Лист

87

VPS и вебсервер



Backend приложение



The diagram illustrates the architecture of a data science project. On the left, four 'Model' boxes point to a central 'Eloquent ORM' box. The 'Eloquent ORM' box is connected to a 'DB Connection' box, which in turn is connected to a 'DB' cylinder. Arrows indicate data flow: 'Query' from models to ORM, 'Result' from ORM to DB, and 'Execute SQL' from DB to ORM. On the right, a grid of 12 database schema diagrams is shown, each representing a different table with its columns and data types.

Model	Query	Result	Execute SQL	DB
Model	↓	↑	↓	↑
Model	↓	↑	↓	↑
Model	↓	↑	↓	↑
Model	↓	↑	↓	↑

The database schema diagrams on the right are as follows:

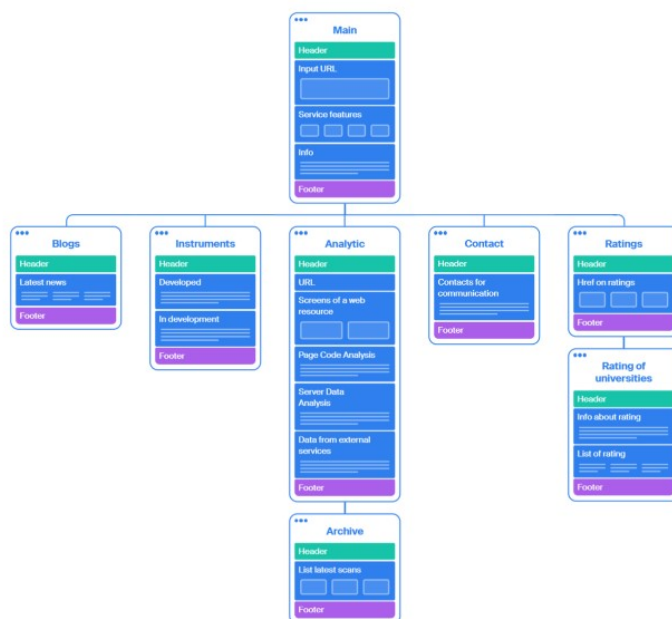
- archive_url_models**: id (bigint(20)), url (varchar(255)), ip (varchar(150)), robot (text), url_id (tinyint(4)), title (varchar(255)), title_length (tinyint(4)), response_status (tinyint(4)), meta_description (varchar(1500)), meta_keywords (varchar(1500)), meta_keywords_length (tinyint(4)), meta_sitemap (tinyint(4)), meta_charset (varchar(1500)), razmarka_ag (tinyint(4)), yandex_ks (tinyint(4)), vikont (tinyint(4)), livelimited (tinyint(4)), antbot (tinyint(4)), yaCounter (tinyint(4)), sponsor_domain (varchar(150)), created_domain (tinyint(4)), egs_domain (tinyint(4)), request_date (tinyint(4)), whole_text (text), updated_at (tinyint(4)), created_at (tinyint(4)), is_deleted (tinyint(4)).
- url_models**: id (bigint(20)), url (varchar(255)), ip (varchar(150)), robot (text), url_id (tinyint(4)), title (varchar(255)), title_length (tinyint(4)), response_status (tinyint(4)), meta_description (varchar(1500)), meta_keywords (varchar(1500)), meta_keywords_length (tinyint(4)), meta_sitemap (tinyint(4)), meta_charset (varchar(1500)), razmarka_ag (tinyint(4)), yandex_ks (tinyint(4)), vikont (tinyint(4)), livelimited (tinyint(4)), antbot (tinyint(4)), yaCounter (tinyint(4)), sponsor_domain (varchar(150)), created_domain (tinyint(4)), egs_domain (tinyint(4)), request_date (tinyint(4)), whole_text (text), updated_at (tinyint(4)), created_at (tinyint(4)), is_deleted (tinyint(4)).
- users**: id (bigint(20)), name (varchar(255)), email (varchar(255)), email_verified_at (tinyint(4)), password (varchar(255)), remember_token (varchar(100)), role (tinyint(4)), updated_at (tinyint(4)), created_at (tinyint(4)), is_deleted (tinyint(4)).
- blog**: id (bigint(20)), title (varchar(255)), content (text), created_at (tinyint(4)), updated_at (tinyint(4)), is_type (tinyint(4)), is_deleted (tinyint(4)).
- blog_types**: id (tinyint(4)), name (varchar(100)).
- rating_models**: id (bigint(20)), name (varchar(255)), name_en (varchar(255)), url (varchar(255)), min_points (tinyint(4)), our_points (tinyint(4)), top (tinyint(4)), updated_at (tinyint(4)), created_at (tinyint(4)), is_deleted (tinyint(4)).
- jobs**: id (bigint(20)), qrcode (varchar(255)), payload (longtext), attempts (tinyint(4)), reserved_at (tinyint(4)), available_at (tinyint(4)), created_at (tinyint(4)).
- failed_jobs**: id (bigint(20)), uuid (varchar(255)), connection (text), payload (text), exception (longtext), failed_at (tinyint(4)).

```

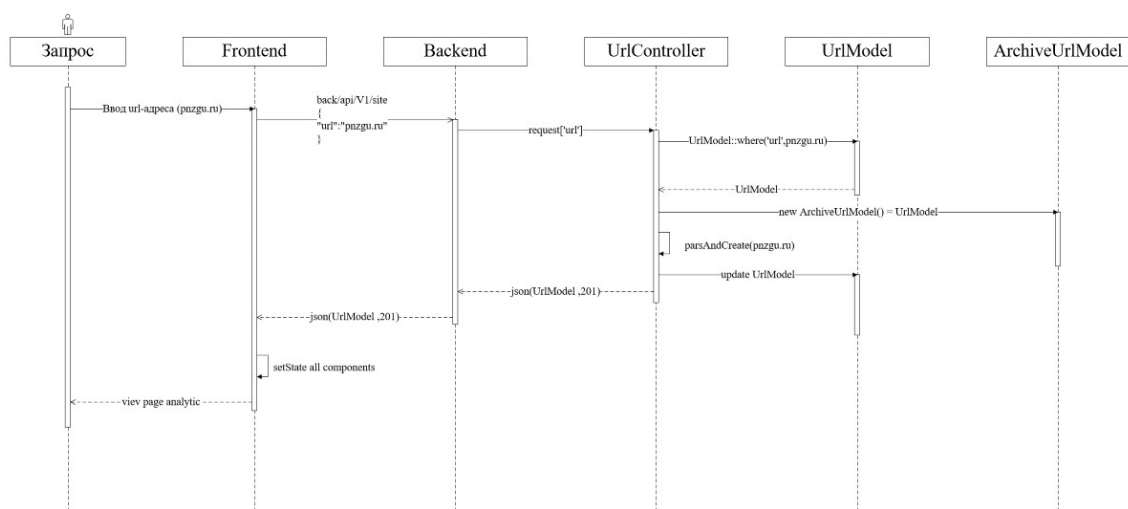
graph TD
    Start([Inicio]) --> Read[Leer valores LRI,]
    Read --> Calc[Calcular número de iteraciones]
    Calc --> Read2[/Leer nuevo número de LRI en pantalla/]
    Read2 --> Dec1{¿<math>abs(x_{n+1} - x_n) < 10^{-5}</math>?}
    Dec1 -- SI --> End1([Fin])
    Dec1 -- No --> Print1[/mostrar pascalCode/]
    Print1 --> Dec1

    Start2([Inicio pascalCode]) --> Read3[/Leer los Logaritmo Pq/Pq(n)</math>]
    Read3 --> Calc2[Calcular número de iteraciones y número de iteraciones]
    Calc2 --> Print2[/pqr/pqr</math>]
    Print2 --> Read4[/Leer los iteraciones y el número de iteraciones y el número de iteraciones</math>]
    Read4 --> Print3[/grf/grf</math>]
    Print3 --> Read5[/Comparar los iteraciones y el número de iteraciones</math>]
    Read5 --> Print4[/Yendo al grf/grf de los iteraciones</math>]
    Print4 --> Read6[/Comparar los iteraciones y el número de iteraciones</math>]
    Read6 --> Dec2{¿<math>abs(x_{n+1} - x_n) < 10^{-5}</math>?}
    Dec2 -- SI --> End2([Fin])
    Dec2 -- No --> Print5[/mostrar pascalCode</math>]
    Print5 --> Dec2
  
```

Frontend приложение



UML – диаграмма последовательностей



```

<script type="text/javascript">
    (function() {
        <!--[if IE]--><script src="/js/html5shiv.js"></script>

        meta http-equiv="X-UA-Compatible" content="IE=9; IE=8; IE=7; IE=6; chrome=" />
        meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

        <title>Кабула - «Шаренгачная техника»</title>

        meta name="description" content="Искусство, Русская государственная пивоварня" />
        meta name="keywords" content="искусство" />
        meta name="viewport" content="initial-scale=width; initial-scale=1.0; maximum-scale=1.0;" />
        meta name="application:color" content="#fff" />
        meta name="application:theme" content="/images/logo/cdn/images/widescreen/widescreen-100x100.png" />
        meta property="og:locale" content="ru_RU" />
        meta property="og:type" content="website" />
        meta property="og:description" content="Искусство, Русская государственная пивоварня" />
        meta property="og:url" content="https://vpr.ru/" />
        meta name="twitter:url" content="https://vpr.ru/" />
        meta property="og:image" content="https://prague.ru/images/prig_main_capus.jpg" />
        meta property="og:image:secure_url" content="https://prague.ru/images/prig_main_capus.jpg" />
        meta name="channel:url" content="https://prague.ru/images/prig_main_capus.jpg" />
        meta name="twitter:image:src" content="https://prague.ru/images/prig_main_capus.jpg" />
        meta property="og:title" content="Искусство" />
        meta name="twitter:title" content="Искусство" />
        meta property="og:site_name" content="Кабула «Шаренгачная техника»" />
        meta name="twitter:site" content="Кабула «Шаренгачная техника»" />

        class id="typel" rel="stylesheet" type="text/css" href="/styles/defaultmain.css" />
        class id="typel" rel="stylesheet" type="text/css" href="/styles/shortpagecustom.css" />
        class id="typel" rel="stylesheet" type="text/css" href="/styles/defaultmain.css" />
        class id="typel" rel="stylesheet" type="text/css" href="/styles/vizyle-site.css" />
        class rel="stylesheet" type="text/css" href="/styles/style_main.css" />
        class rel="stylesheet" type="text/css" href="/styles/defaultmain.css" />
        class rel="stylesheet" type="text/css" href="/styles/vizyle_theme.css" />
        class rel="stylesheet" type="text/css" href="/styles/vizyle-site-main.css" />
        class rel="stylesheet" type="text/css" href="/resources/content/fonts/open_sans/all.css" />
        class rel="stylesheet" type="text/css" href="/styles/vizyle-theme-main.css" />
        class id="typel" rel="stylesheet" type="text/css" href="/css/mixins/defaultmain.css" />
    })();
</script>

```

Генератор идей
Генератор сценариев
Генератор персонажей
Генератор диалогов
Генератор диалогов

Инструменты аналитики

Получите анализ сайта и узнайте его сильные стороны

Анализ сайта

Анализировать

Что анализирует наш сервис

Анализ дизайна сайта

Проверка удобства и общего внешнего вида сайта, соответствие требованиям юзабилити

SEO анализ на сайте

Проверка мета-тегов, структуры сайта, наличия и качества контента, соответствия требованиям поисковых систем

Технический анализ

Общая информация о состоянии сайта, скорости загрузки, безопасности, мобильности

Для чего это нужно

Сайт — это визитная карточка компании. Если сайт не отвечает требованиям юзабилити, не соответствует требованиям поисковых систем, не имеет качественного контента, то компания теряет клиентов. Наш сервис поможет вам выявить проблемы и улучшить сайт, чтобы вы могли привлечь больше клиентов и увеличить продажи.

Рейтинг сайтов университетов

Цель данного рейтинга, заложенная разработчиком сайта ([Ranking Web of Universities, webometrics.info](#)), который определяет рейтинг сайтов Российской Федерации с февраля 2002 года:

«Специальные критерии оценки сайтов включают: проверка на наличие обязательных разделов по требованиям, предъявляемым к сайтам образовательных организаций; наличие кода главной страницы, анализ данных сервера, история доменного имени, данные сторонних сервисов и т.д. В следующей версии рейтинга методология оценки будет изменена, за счет усиления прозрачности процедуры оценки».

Титул статьи: **Октябрь 2022**

Наши партнеры, чьи сайты вошли в топ-10, оставляют авторам! Напишите нам: pap@webdms.ru

Список оцениваемых сайтов:

гипс:

Наименование университета	Наименование университета по англоязычному языку	URL-адрес ресурса	Количество обязательных разделов по требованиям Министерства образования РФ	Технический анализ	Общая оценка
Пензенский государственный университет	Penza State University	http://www.pspu.ru/	15	99	114
Пензенский государственный технологический университет	Penza State Technological University	http://www.pstu.ru/	15	78	93
Пензенский государственный университет архитектуры и строительства	Penza State University of Architecture and Construction	http://psuas.ru/	15	61	76
Пензенская государственная сельскохозяйственная академия	Penza State Academy of Agriculture	http://psaa.ru/	14	61	75

Результаты работы

- Разработан и отлажен алгоритм анализа интернет-ресурсов;
- Создан и настроен VPS под управлением nginx, обслуживающий оба приложения
- Разработано и отлажено серверное приложение, способное анализировать интернет-ресурсы;
- Разработано и отлажено клиентское приложение для взаимодействия с сервером.

Спасибо за внимание!

					ПГУ1.09.03.01.01.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		92