МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Экономический факультет**

**Кафедра экономики и управления инновационными системами**

Допустить к защите

Заведующий кафедрой

канд. экон. наук, доц.

\_\_\_\_\_\_\_\_\_\_\_ К.О. Литвинский

(подпись)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(БАКАЛАВРСКАЯ РАБОТА)**

**РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ ОЦЕНКИ**

**КАЧЕСТВА СЕТЕВЫХ РЕСУРСОВ**

Работу выполнил В.С. Ерекешов

(подпись)

Направление подготовки 27.03.02. Управление качеством

Направленность (профиль) Управление качеством в социально- экономических системах

Научный руководитель

канд. техн. наук, доц. А.И. Решетняк

(подпись)

Нормоконтролер

канд. экон. наук, доц. Н.Н. Аведисян

(подпись)

Краснодар

2022

**СОДЕРЖАНИЕ**

Введение 3

1. Теоретические основы разработки ПО для оценки качества сетевых

ресурсов 7

* 1. Теоретические основы оценки качества сетевых ресурсов 7
  2. Теоретические основы разработки программного обеспечения 23

1. Технические основы разработки ПО для оценки качества сетевых

ресурсов 27

* 1. Языки программирования и разметки 27
  2. Инструменты и библиотеки 31

1. Процесс разработки 35
   1. Планирование 35
   2. Backend 37
   3. Graphical User Interface 44
   4. Тестирование и сборка 49

Заключение 51

Список использованных источников 53

Приложение А «Графический интерфейс ПО» 59

**ВВЕДЕНИЕ**

Еще двадцать-тридцать лет назад многие люди считали Интернет «прерогативой» для богатых, а коммерческие компании только начинали робкие попытки пролезть в этот пока еще слаборазвитый и малопонятный рынок.

Сейчас же, когда Интернет развивается галопирующим темпом, и большая часть населения нашей планеты уже имеет доступ во всемирную паутину, на рынке информационных интернет-ресурсов развилась серьезная конкуренция, участники которой сражаются за каждую миллисекунду, проведенную пользователем на странице, за каждый бит лишнего трафика.

Как человек не будет приобретать некачественные товар или услугу, так и не будет долго задерживаться на некачественных страницах в Интернете.

Но что вообще есть качество интернет-ресурса? На мой взгляд, существует три главных аспекта качества сайта: увлекательность контента, наличие багов и проработанность SEO.

К сожалению для многих аналитиков и контент-менеджеров, технически измерить качество контента достаточно тяжело, а если речь идет о смысловой нагрузке, то практически невозможно.

С багами, ошибками на сайтах все гораздо проще. Поиском багов занимается QA отдел компании-разработчика сайтов: мануальные тестировщики и разработчики автоматизированных тестов.

Однако, в данной выпускной квалификационной работе мы остановимся на SEO. SEO – это поисковая оптимизация, комплекс мероприятий по внутренней и внешней оптимизации для поднятия позиций сайта в результатах выдачи поисковых систем.

Есть такая поговорка: «Встречают по одежке, а провожают по уму». Так вот, основная проблема любого SEO-менеджера – это встретить, точнее, чтоб тебя встретили, а дальше уже одежка и ум.

Чем качественнее SEO-менеджер настроит поисковую оптимизацию сайта, тем больше пользователей после того, как вобьют запрос в поисковую строку браузера, увидит именно этот сайт, а значит у компании будет больше трафик и выше прибыль.

Но у маленьких компаний не всегда есть средства для того, чтобы воспользоваться услугами SEO менеджера-фрилансера, и тем более нанять его в штат. Однако, если маленькая компания не будет настраивать поисковую оптимизацию своего сайта, она может легко потеряться среди более крупных конкурентов.

Таким образом, изучение механизмов ранжирование интернет ресурсов в браузере и последующая реализация проверки корректности настроек факторов ранжирования, напрямую зависящих от разработчиков данного интернет ресурса в виде программного обеспечения для настольных компьютеров поможет маленьким компаниям провести базовую настройку внутренних факторов ранжирования, тем самым экономя ресурсы компании и позволяя компании поднять свою страницу в результате выдачи браузера и увеличить поток клиентов и, соответственно, прибыль.

Исходя из данных соображений, выбранная проблематика представляется актуальной, имеющей практическое и теоретическое значение.

Объектом исследования является ранжирование сетевых ресурсов в выдаче браузера.

Предметом исследования являются внутренние факторы поисковой оптимизации страницы.

Целью выпускной квалификационной работы является разработка программного обеспечения для оценки качества реализации отдельных факторов SEO.

В соответствии с поставленной целью необходимо решить следующие задачи:

1. Найти и изучить научные источники, раскрывающие принципы ранжирования страниц в браузерах.
2. На базе изученных научных источников выявить требования к технической реализации отдельных внутренних факторов SEO и составить список факторов, основанных на информации в свободном доступе, доступных для реализации в данный момент, учитывая текущее материально-техническое обеспечение.
3. На базе составленного и детально описанного списка доступных для реализации внутренних факторов поисковой оптимизации, на базе доступного материально-технического обеспечения и информации в свободном доступе разработать программное обеспечение для оценки качества реализации отдельных внутренних факторов ранжирования страницы.

Для решения поставленных задач использованы следующие методы исследования:

– теоретические: изучение и анализ специальной литературы;

– эмпирические: эксперимент;

– методы количественного и качественного анализа экспериментальных данных.

Структура выпускной квалификационной работы включает в себя содержание работы, введение, три главы, заключение, список использованных источников.

Во введении обоснована актуальность темы, определены цель и задачи, объект и предмет, методы сбора и обработки теоретической и прикладной информации.

Первая глава посвящена анализу теоретической информации по заявленной теме исследования. В первом параграфе рассматриваются теоретические основы оценки качества сетевых ресурсов. Во втором параграфе рассматриваются теоретические основы разработки программного обеспечения.

Вторая глава представляет собой технические основы разработки данного программного обеспечения. Первый параграф второй главы вводит в основу языков программирования, использованных в данном проекте. Второй параграф рассказывает о библиотеках и зависимостях, использованных при разработке ПО.

Третья глава детально описывает процесс разработки программного обеспечения. В первом параграфе рассматривается планирование процесса разработки. Второй параграф дает детальное описание разработки логической части проекта, она же Backend. Третий параграф дает представление о том, как шел процесс разработки графического пользовательского интерфейса. Четвертый параграф рассказывает о тестировании и сборке проекта в готовый продукт.

**1 Теоретические основы разработки ПО для оценки качества**

**сетевых ресурсов**

**1.1 Теоретические основы оценки качества сетевых ресурсов**

Целью администрации подавляющего большинства страниц в интернете является увеличение трафика и привлечение потенциальных клиентов для монетизации этого трафика, будь то продажа услуг или рекламы.

Возможности Интернета растут едва ли не быстрее, чем количество веб-сайтов и пользователей. Никого не удивляет, что через интернет можно оплачивать коммунальные услуги, заказывать еду онлайн, покупать одежду и даже зарабатывать.

Сегодняшняя глобальная сеть – это влиятельная сила: инструмент продвижения, средство отображения и поиска любой информации, а также огромное пространство для общения. Это далеко не полный перечень возможностей, которые предлагает нам Всемирная паутина.

Возвращаясь к основной функции Интернета как источника информации, следует сказать, каждый пользователь интернета может получить практически любые данные от чего угодно и от кого угодно из сети. Сегодня у каждого есть аккаунт хотя бы в одной социальной сети. Вся информация о том или ином человеке, за очень редким исключением, находится в открытом доступе.

Практика показала, что чем привлекательнее профиль ВКонтакте, Facebook или других социальных сетей, тем больше "лайков" и популярности он имеет у владельца аккаунта. Если каждый может создать для себя определенный имидж через Интернет без затрат и специальных навыков, то, конечно, почти каждая компания, организация и т.д. будет и должна использовать свои собственные и сторонние веб-порталы как способ отображения информации о себе, а также как средство продвижения ваших товаров и услуг и других тем мероприятий.

Есть два основных метода привлечения трафика на страницу: реклама и SEO-оптимизация. В данной работе мы не будем разговаривать о рекламе, мы остановимся на ранжировании.

Для того, чтобы пользователи могли перейти на нужный веб-портал, для продвижения сайта используется поисковая оптимизация - SEO оптимизация (search engine optimization). Этот инструмент представляет собой набор мер по улучшению позиции веб-сайта в результатах запросов поисковых систем. SEO и некоторые другие методы интернет-маркетинга, такие как SMM (маркетинг в социальных сетях -продвижение в социальных сетях) и SEA (search engine advertising - поисковая реклама), часто используемые вместе меры по продвижению страницы организации.

Согласно исследованиям Forrester Research Inc, пользователи вряд ли будут продолжать просматривать более тридцати позиций результатов запросов в браузере. Поэтому сайт организации, в принципе, должен быть на первом месте в поисковых системах, чтобы компанию знали и, более того, интересовались предметом ее деятельности. Здесь начинают работать стереотипы: чем выше местоположение сайта, тем более уважаемым будет этот интернет-ресурс.

Сектор SEO развивается наиболее сильно в последнее десятилетие. На протяжении многих лет различные эксперты, консультанты, организации или учреждения, связанные с областью цифровых коммуникаций и маркетинга, постоянно ставили в общественное обсуждение мнения и данные о тех аспектах, которые являются или могут быть более важными для страницы или веб-сайта, чтобы достичь лучших позиций в поисковых системах, особенно в Google, Искатель по преимуществу во всем мире. Эта глава призвана рассмотреть наиболее важные моменты исследований и исследований, которые содержат статистическую силу и являются репрезентативными для наших исследований.

Говоря о SEO, мы будем ссылаться на оптимизацию интернет-порталов, чтобы достичь лучших позиций в естественных результатах поисковых систем. Органические результаты позиционирования отображаются благодаря математическому алгоритму, который обрабатывает большое количество данных, чтобы показать рейтинг, когда мы выполняем поиск.

Чтобы иметь возможность позиционировать веб-сайт на вершине, учитываются различные методы / стратегии, которые применялись на протяжении многих лет и могут меняться со временем. Кроме того, мы сосредоточимся на анализе наиболее важных переменных для позиционирования в Google, потому что это наиболее используемая поисковая система в мире.

Различные авторы сегментируют или группируют наиболее важные аспекты, которые Google принимает во внимание, чтобы показать свои результаты. Для Уилла Кумбе SEO стоит на четырех столпах:

1. Релевантность: Мы должны убедиться, что каждая страница имеет определенную тему и с большим количеством ключевых слов, связанных с темой. Кроме того, она должен быть полезна для пользователей.
2. Отслеживаемость: C точки зрения сотрудничества мы должны сделать так, чтобы Google мог легко просматривать, сканировать и проверять наши веб-страницы.
3. Взаимодействие: Мы должны показывать Google сигналы, которые показывают, что их пользователи тратят много времени на наши веб-страницы и что они используют этот контент.
4. Авторитет: Наконец, мы должны показать Google, что наш контент заслуживает доверия, и что он зачисляется другими надежными веб-сайтами в нашем секторе и связанных с нашими ключевыми словами.

Согласно Чжан и Cabage (2017, стр. 148-150), стратегии SEO могут быть организованы в трех областях:

1. Содержание: это все те тактики, которые мы можем применить на нашем сайте, и которые мы можем суммировать в трех: качество контента, структура сайта, изменения SEO на странице. Первое касается предоставления пользователям текстов и полезных функций для читателей. Что касается структуры сайта, речь идет о создании организованного веб-сайта с соответствующими разделами/категориями, а также с возможностью легко понять и ориентироваться для пользователей. Наконец, изменения на странице касаются адаптации нашего контента в соответствии со стратегией ключевых терминов, по которым мы хотим найти нашу страницу в результатах поиска.
2. Link Building: Одним из моментов, которые отличали Google, когда он вышел на рынок поисковых систем, был способ анализа отношений между веб-сайтами через любые ссылки, которые могут быть между ними. С помощью ссылки Google может анализировать популярность сайтов, доверие, которое они могут иметь, проверить соответствие темы или обнаружить спам. Таким образом, link building является фундаментальной стратегией для достижения позиций в первых результатах поиска. Он состоит из построения / сбора ссылок с других веб-сайтов на конкретный сайт.
3. Социальный обмен: Также известный как «социальные сигналы». Эта стратегия включает в себя создание контента, который может быть привлекательным для пользователей и делиться им в социальных сетях. Это еще один способ получить ссылки на наш сайт. Google также указал, что относительный авторитет пользователя влияет на взвешивание этих социальных сигналов.

Ранжированием называется упорядочивание результатов поиска по их релевантности [1]. Ранжирование страницы зависит от внешних и внутренних факторов.

Внешние факторы ранжирования – это факторы, которые оценивают взаимодействие ресурса с пользователями интернета. Внешние факторы состоят из цитируемости, социальных факторов и поведенческих факторов.

Цитируемость – это количество ссылок с других ресурсов, ведущих на страницы, содержащие искомую фразу: простое размещение ссылок на других ресурсах и размещение в каталогах. При высоком качестве контента и показателях посещаемости данный фактор может прогнозировать быстрый естественный прирост.

В 2016 году компания Backlinko, специализирующаяся на цифровом маркетинге, опубликовала исследование, основанное на 10 000 ключевых из разных областей и секторов, чтобы попытаться измерить важность / влияние различных переменных позиционирования в топ-10 Google (Dean, 2016). Следуя исследованию Чжана и Кабажа (2017), одна из самых высоких корреляций в этом исследовании связана с цитируемостью. Сайты с наибольшим количеством реферальных доменов (referring domains) лучше позиционируются в Google. Страницы, достигающие первой позиции, получают в среднем 280 доменов ссылок. Результаты данного исследования представлены на рисунке 1.

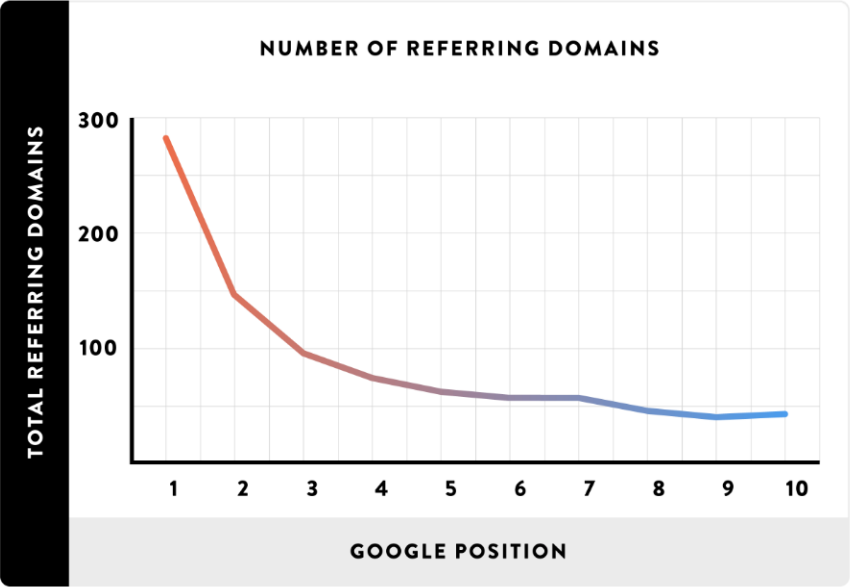


Рисунок 1 – Количество доменов ссылок против позиционирования в топ-10 Google.com

В том же духе, сайты с наибольшим авторитетом ссылок – это те, кто лучше всего позиционируется на первой странице Google, в первой пятерке результатов средний авторитет составляет от 67 до 69 пунктов (из 100). Кроме того, общее количество ссылок, которые получает сайт, превосходит рейтинг позиций, в топ-5 полученные ссылки колеблются между 5000 и 35000. С позиции 6 по 10, шкала полученных ссылок меньше и более однородна, между 5000 и 3000.

В соответствии с важностью полученных ссылок, Эрик Энге (2019) опубликовал исследование, проведенное на 27000 поисковых терминов, чтобы определить, была ли корреляция между количеством ссылок, которые получают лучшие сайты рейтинга. Для исследования были проанализированы информационные и коммерческие поиски из следующих секторов: здравоохранение, финансы, технологии и другие. Кроме того, было рассмотрено общее количество полученных ссылок, Авторитет страницы (AP) и Авторитет домена (AD) (Enge, 2019). Используя корреляцию Спирмена, была обнаружена положительная корреляция 0,293 между полученными ссылками и позиционированием. Также среди индикаторов AD с 0,327 и AP с 0,321.

В целом, мы можем сказать, что это исследование является репрезентативным по количеству проанализированных элементов (страниц и ключевых слов), но имеет ограничение на то, что оно было выполнено автоматически через робота, запрограммированного для этого исследования. Это может быть слабостью, потому что лучший способ проверить каждую из этих переменных – это ручные инструменты, то есть человек, который получает данные из инструмента и смотрит на страницу напрямую, в зависимости от рассматриваемой переменной.

Социальные факторы отражают интерес пользователя к ресурсу. К социальным факторам относится фактор влияния социальных сетей на посещаемость страницы. Активность и регулярность выхода качественного контента, а также большое количество лайков, комментариев и репостов дает знак поисковым системам, что данный ресурс полезен, актуален и релевантен для аудитории.

Поведенческие факторы оценивают поведение пользователей во время посещения страницы и в выдаче поисковых систем, когда он делает выбор между предложенными сайтами в поисковой системе. В этом случае ранжирование формируется, исходя из полезности сайта пользователю. К поведенческим факторам относятся:

1. Время, проведенное на сайте – основополагающий фактор при оценке качества страницы. Данный показатель может улучшаться при помощи точек захвата: изображений, видео, таблиц и графики.
2. Количество просмотров страниц – оценка качества сайта в целом, о чём свидетельствует количество просмотров других документов.
3. Кликабельность (CTR) в поисковой выдаче – учитывает наименование страницы и сниппет. Сниппет – это программный термин, обозначающий небольшую область многократно используемого исходного кода, машинного кода или текста. Обычно это формально определенные оперативные блоки, которые необходимо включить в более крупные программные модули.
4. Показатель возвратов – определяет количество возвратов пользователя на ресурс.
5. Показатель отказов – определяет количество быстрых закрытий страницы. Высокий показатель отказов может дать сигнал, что страница не релевантна: имеющаяся на ней информация не соответствует поисковому запросу пользователя.

В данной работе мы не будем уделять много внимания внешним факторам ранжирования, так как они не зависят от веб-разработчиков, в отличии от внутренних.

Внутренние факторы ранжирования – это факторы, которые относятся к структуре сайтов и находятся под контролем веб-разработчика. Внутренние факторы формируются с созданием сайта и оказывают непосредственное влияние на позиции страниц в поисковых выдачах. Внутренние факторы ранжирования формируются в HTML документе страницы.

HTML (от англ. HyperText Markup Language – «язык гипертекстовой разметки») – стандартизированный язык разметки документов во Всемирной паутине [2]. Язык HTML интерпретируется браузерами; полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

К внутренним факторам относятся:

* анализ контента, то есть текста сайтов, происходит по определению качества контента по многим параметрам, основными из которых будут уникальность текста, новизна, актуальность, релевантность запросу пользователя и естественность (правильная оптимизация статьи);
* теги и метатеги, такие как заголовок title, ключевые слова, описание, теги h1-h6, анализ правильной технической составляющей (корректная верстка, чистый исходный код, настройка редиректов, зеркал сайта, robots.txt), и т. д.;
* удобная навигация и структура, правильная перелинковка (связь внутренних страниц сайта ссылками друг на друга), количество рекламы. Важным параметром является возраст сайта (ресурса с момента его первого попадания в индекс) – чем старше сайт, тем он авторитетнее и выше ранжируется при прочих равных;
* внешние ссылки: позицию в ранжировании увеличивают ссылки на качественные площадки, которые могут дополнить информацию и помочь решить проблемы пользователей, однако избыточное количество ссылок может, напротив, уменьшить позицию.

На внутренних факторах ранжирования мы и сконцентрируем внимание в данной работе.

Факторы ранжирования, как внутренние, так и внешние и являются показателями качества сетевого ресурса.

За оптимизацию факторов ранжирования отвечает поисковая оптимизация [3]. Поисковая оптимизация (англ. search engine optimization, SEO) – это процесс улучшения веб-сайта или веб-страницы с целью повышения качества и количества живого трафика из поисковых систем [4]. Качественное SEO означает, что веб-страница с большей вероятностью будет отображаться выше на странице результатов поисковой системы (SERP).

Одной из разновидностей поисковой оптимизации является On-Page SEO.

On-Page SEO (также известная, как «On-Site SEO») – это практика оптимизации отдельных веб-страниц для повышения рейтинга и получения более релевантного трафика в поисковых системах. On-Page SEO относится как к содержанию, так и к исходному коду HTML страницы, который можно оптимизировать, в отличие от внестраничного SEO, которое относится к ссылкам и другим внешним сигналам [5].

Так как алгоритмы ранжирования в поисковых системах не стоят на месте, а постоянно усложняются и дорабатываются, On-Page SEO была взята в данной работе за основу определения критериев оценки качества сетевых ресурсов, так как эта поисковая оптимизация является одной из самых современных, отвечающих всем требованиям ранжирования современных поисковых систем, таких как Google и Яндекс. Также некоторые критерии были взяты из обычной SEO.

Исходя из возможности технической реализации оценки качества, автором было выделено 13 критериев качества сетевых ресурсов.

Первым критерием по списку и первым в истории развития поисковой оптимизации является фактор наличия ключевых слов. С самого начала развития поисковых систем, для повышения ранжирования страницы в поисковых системах разработчики сайтов используют ключевые слова [6]. Для этого разработчики используют тег <meta> в HTML документе. С помощью этого тега они добавляют на страницу информацию полезную для поисковых систем: собственно, сам перечень ключевых слов, а также краткое описание страницы [7].

В 2012 году Рикардо Каррерас провел исследование SEO, которое он представил в своей докторской диссертации в Мадридском университете Комплутенсе. Исследование состояло в том, чтобы проанализировать 20 критериев позиционирования на страницах, которые были в первой позиции Google.es, для 359 слов различной тематики (2012, стр. 61-67).

Среди наиболее релевантных результатов, которые можно наблюдать в этом исследовании, (2012, стр. 422-454), что 93% проанализированных страниц содержали ключевые слова, которые искали в названии.

Средняя протяженность текста для всех страниц составила 2108 слов. 69% проанализированных доменов получили более 100 тысяч ссылок. 85% проанализированных веб-сайтов получили PageRank более 3 баллов (из 10) и 61% выше 5, что является значительным результатом, если учесть, что шкала является логарифмической. 87% доменов были старше 7 лет. 66% результатов содержали искомое ключевое слово в теге заголовка H1. 88% изученных страниц содержали искомое слово в URL-адресе[8].

Перечень ключевых слов задаётся тегом <meta>, у которого атрибут «name» имеет значение «keywords». Ключевые слова (самые важные слова из содержания страницы) перечисляются в атрибуте «content» через запятую:

<meta name="keywords" content="важные, ключевые, слова">

На практике этот критерий реализован следующим образом: программа парсит HTML документ страницы, и если она находит данный метатег, и он не пустой, то программа возвращает пользователю положительный ответ, в противном случае ответ будет отрицательным.

Следующим критерием оценки качества является фактор наличия описания страницы. Описание страницы находится в ещё одном метатеге, т. е. оно задаётся похожим образом, только значение атрибута name меняется на description:

<meta name="description" content="краткое описание">

Описание страницы показывается под названием и URL адресом в списке страниц в поисковой поисковой системе [9].

Описание страницы важно не меньше, чем наличие ключевых слов, а вместе они создают основу поисковой оптимизации страницы [10].

Данный критерий на практике реализован так же, как и прошлый, только в этом случае программа будет искать тег <meta>, у которого атрибут «name» имеет значение «description».

Следующий критерий оценки выходит из прошлых двух. Этим критерием является наличие ключевых слов в описании. Само по себе наличие ключевых слов в описании не является важным фактором, но лишним он не будет, поэтому было решено добавить его как критерий в ПО [11].

Реализован данный критерий следующим образом: программа возвращает пользователю процентное соотношение сходства ключевых слов и описания, а также для каждого слова из списка ключевых слов программа возвращает положительный ответ если данное слово есть в описании и отрицательный ответ, если данного слова нет.

Наличие ключевых слов в заголовках h1 также является одним из критериев оценки качества сетевых ресурсов.

В тег h1 обычно заключаются заголовки страниц. Вероятно, поэтому включение ключевых слов в заголовок всегда было традиционной мудростью в SEO.

Джон Мюллер, адвокат поисковой системы Google, подтвердил важность заголовков в 2020 году: «Когда речь заходит о тексте на странице, заголовок является действительно сильным сигналом, говорящим нам, что эта часть страницы посвящена конкретной теме» [12].

Данный критерий на практике реализован следующим образом: программа возвращает процентное соотношение сходства ключевых слов со словами в тегах h1, а также для каждого из ключевых слов она возвращает положительный ответ если данное слово есть в тегах h1.

Следующий критерий похож на предыдущий, но он проверяет сходство ключевых слов с тегами h2. В теги h2 обычно заключаются подзаголовки, поэтому включение ключевых слов в данные теги является хорошей практикой поисковой оптимизации [13].

Реализован данный критерий так же, как и прошлый, только в данном случае программа проверяет сходство ключевых слов не с h1, а с h2 и так же для каждого ключевого слова возвращает положительный или отрицательный ответ исходя из наличия данного слова в тегах h2.

Следующий критерий отражает фактор наличия ключевых слов в первых 150 словах на странице. Наличие ключевых слов в первых 150 словах – это старая тактика поисковой оптимизации, которая все еще имеет значение [14]. Этот фактор важен, потому что поисковые системы уделяют большое внимание терминам, которые появляются на странице, и чем раньше на странице появляются основные термины - тем лучше. Данный фактор позволяет поисковой системе понимать соответствует ли содержание страницы ключевым словам или нет [15].

Реализован данный критерий так же, как и прошлые, но в данном случае программа ищет совпадения в ключевых словах и первых 150 слов на странице, и также возвращает совпадения по каждому слову.

Ещё одним немаловажным критерием является оптимизация структуры URL-адреса страницы.

Унифицированный указатель ресурса (англ. Uniform Resource Locator, URL), в просторечии называемый веб-адресом, представляет собой ссылку на веб-ресурс, которая указывает его местоположение в компьютерной сети и механизм для его получения [16].

Структура URL-адреса является одной из важнейших, но при этом недооцененных частей поисковой оптимизации. Кроме того, URL-адреса в поисковой выдаче Google для мобильных устройств и компьютеров теперь находятся над тегом заголовка. Поэтому URL-адрес сейчас более важен, чем раньше [17].

Когда мы рассматривали различные переменные, связанные с доменом, мы увидели, что доменное имя и URL-адрес выделяются как одни из самых влиятельных для ранжирования. Исследование, проведенное Search Engine Watch (Sentance, 2017) проанализировал первую десятку позиций для 10 конкурентоспособных ключевых слов на английском языке из различных секторов, таких как: бизнес, кредитные карты, сантехника, программное обеспечение, продукты питания и напитки, потеря веса, правительство или туризм. Они подчеркивают, что 63% результатов первой страницы Google.com для проанализированных поисков они имели в URL слова, которые были найдены. Области, где наиболее заметными были слова, которые искали в URL, были «дебет» с 76%, «бизнес» с 73% и «еда и напитки» с 71% [18]. Результаты данного исследования представлены на рисунке 2.

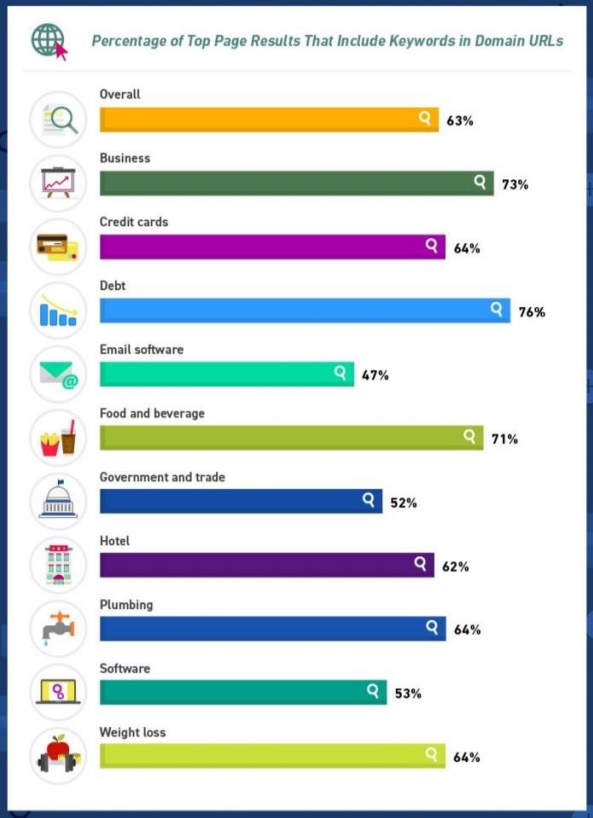


Рисунок 2 – Процент результатов первой страницы Google.com

которые включают искомые слова в URL

Таким образом, чтобы оптимизировать URL-адрес, его необходимо сделать коротким, а также включить в него ключевые слова. Это не значит, что что ваш URL должен содержать только ваши ключевые слова. Вполне нормально добавить к URL-адресу еще одно или два слова [19].

Данный критерий реализован следующим образом: программа возвращает процентное соотношение ключевых слов и слов из URL-адреса, проверяет каждый элемент пути на длину, а так же для каждого ключевого слова возвращает положительный или отрицательный ответ в зависимости от наличия данного слова в URL-адресе страницы.

Ещё одним критерием оценки качества является частота ключевых слов. Данный критерий подразумевает то, как часто определенное ключевое слово попадается на странице [20]. Google может отрицать, что использование одного и того же ключевого слова несколько раз помогает, но опытные SEO специалисты утверждают, что это определенно работает. Утверждается, что поисковая система будет более уверена в том, что содержание страницы соответствует теме, если ключевое слово будет повторяться 10 раз, чем если бы оно повторялось 1 раз [21]. Это не говорит о том, что страницу нужно максимально набивать ключевыми словами, речь идет о упоминание вашего целевого ключевого слова несколько раз, чтобы подтвердить поисковой системе, что ваша страница действительно посвящена этой теме.

Данный критерий реализуется просто: программа проходится по всем словам на странице и высчитывает столько раз конкретное ключевое слово встречается на сайте.

Следующим критерием является использование внешних ссылок. Внешние ссылки на связанные страницы помогают Google определить тему вашей страницы [22]. Это также показывает Google, что ваша страница является центром качественной информации. Reboot Online провели эксперимент, чтобы посмотреть, помогают ли внешние ссылки улучшить рейтинг. Они создали 10 новых сайтов.

Половина веб-сайтов связана с авторитетными сайтами (такими как Оксфордский университет). Другая половина не имела внешних ссылок. Эксперимент показал, что сайты с внешними ссылками превосходят в ранжировании сайты без внешних ссылок [23].

Реализация данного критерия такова: если на странице есть внешние ссылки, программа возвращает положительный ответ, количество этих ссылок и сами ссылки, если внешних ссылок на странице нет, то программа возвращает отрицательный ответ.

Важнейшим фактором поисковой оптимизации является заголовок страницы. Заголовок страницы находится в теге «title» HTML документа сайта. Заголовок сайта важен тем, что он - первое, что видит пользователь, когда поисковая система возвращает ему список сайтов в ответ на запрос [24]. Заголовок страницы даёт общие сведения о странице не только пользователям, но и поисковой системе, поэтому важно, чтоб в теге «title» присутствовало хотя бы одно ключевое слово [25].

Как критерий оценки качества сетевого ресурса данный фактор реализуется следующим образом: программа возвращает процент соответствия заголовка ключевым словам, а также показывает, присутствует ли определённое ключевое слово в теге «title» или нет.

Следующий критерий оценки качества сетевых ресурсов – фактор наличия текущего года в заголовке и в описании. Данный фактор не зависит на ранжирование напрямую, но он зависит на кликабельность [26]. Добавление года в заголовок и описание позволяет пользователю понять, что информация на сайте является актуальной.

Реализуется данный критерий следующим образом: программа проверяет заголовок и описание на наличие текущего года и возвращает положительный ответ, если текущий год присутствует.

Ещё один критерий – внутренние ссылки. Внутренние ссылки – это гиперссылки на другие страницы этого же сайта [27].

Ссылка в HTML выглядит следующим образом:

<a href="https://domain.com">текст ссылки (анкор)</a>

Их называют «внутренними ссылками», потому что они ведут на страницы того же домена.

У этих ссылок три основных функции:

– помогают пользователю легко ходить по сайту;

– определяют иерархию и структуру сайта;

– влияют на рейтинг страницы сайта [28].

Внутренние ссылки сайта влияют на улучшение позиций сайта на странице результатов поиска и помогают в оптимизации конкретных страниц [29]. Благодаря им на сайте создается сеть ссылок, которые упрощают использование сайта пользователями, а также позитивно влияют на скорость и глубину сканирования сайта поисковыми системами.

Реализованы внутренние ссылки как критерий следующим образом: если на странице есть внутренние ссылки, программа возвращает положительный ответ, количество этих ссылок и сами ссылки, если внутренних ссылок на странице нет, то программа возвращает отрицательный ответ.

Альтернативный текст для изображения является последним в списке критериев оценки качества сетевых ресурсов. Альтернативный текст используется в HTML коде для описания внешнего вида изображения и его функции на странице. Альтернативный текст находится в атрибуте «alt» тега «img» [30].

Аспекты важности альтернативного текста:

1. Принцип веб-доступности: слабовидящим пользователям, использующим программы чтения с экрана, необходим атрибут «alt», чтобы лучше понять изображение на странице.
2. Альтернативный текст будет отображаться вместо изображения, если файл изображения не может быть загружен.
3. Альтернативный текст обеспечивает лучший контекст / описание изображения для поисковых роботов, помогая им правильно индексировать изображение [31].

Пример альтернативного текста:

<img src="pupdanceparty.gif" alt="Щенки танцуют">

Реализация данного критерия: Программа ищет количество всех тегов img и возвращает его, а также возвращает количество тегов img с атрибутом alt и количество тегов img без данного атрибута. Отдельно программа вернет список с адресами изображений и alt текстов всех тегов img, у которых есть атрибут alt.

**1.2 Теоретические основы разработки программного обеспечения**

Программное обеспечение (ПО) – программа или множество программ, используемых для управления компьютером [32].

Программное обеспечение – важнейшая составляющая информационных технологий, оно включает компьютерные программы и данные, предназначенные для решения определённого круга задач, эти программы хранятся на машинных носителях. Программное обеспечение представляет собой либо данные для использования в других программах, либо алгоритм, реализованный в виде последовательности инструкций для процессора.

Разработка программного обеспечения (англ. software development) – деятельность по созданию нового программного обеспечения [33].

Жизненный цикл разработки программного обеспечения, для краткости SDLC, представляет собой четко определенную, структурированную последовательность этапов разработки программного обеспечения для разработки предполагаемого программного продукта [34].

Деятельность SDLC:

SDLC предлагает ряд шагов, которые необходимо выполнить для эффективной разработки программного продукта. SDLC включает в себя следующие этапы:

1. Анализ требований. Отвечает на вопрос «Какие проблемы требуют решений?».
2. Планирование. Отвечает на вопрос «Что необходимо сделать?».
3. Проектирование и дизайн. Отвечает на вопрос «Как добиться поставленных целей?».
4. Разработка ПО. Регулирует процесс создания продукта.
5. Тестирование. Обеспечивает качественную работу продукта.
6. Развертывание. Регулирует использование финального продукта.

Этап 1. Анализ требований.

На этом этапе SDLC необходимо получить обратную связь и поддержку от соответствующих внутренних и внешних заинтересованных сторон. Необходимо будет задуматься о том, кто станет вашими потенциальными пользователями. В целом необходим ответить на следующий вопрос: «Какие проблемы требуют решений?».

Этап 2. Планирование.

Как только анализ требований будет выполнен, следующим шагом будет четкое определение и документирование требований к продукту и их утверждение от клиента или аналитиков рынка. Это делается с помощью документа SRS (Спецификация требований к программному обеспечению), который содержит все требования к продукту, которые должны быть спроектированы и разработаны в течение жизненного цикла проекта.

SRS предназначен для того, чтобы установить базу для соглашения между заказчиком и разработчиком (или подрядчиками) о том, как должен функционировать программный продукт [35].

Этап 3. Проектирование.

SRS – это ориентир для разработчиков продукта, чтобы предложить лучшую архитектуру для продукта, который будет разработан. На основании требований, указанных в SRS, обычно предлагается несколько подходов к проектированию архитектуры продукта, которые документируются в спецификации DDS – документ проектирования.

Эта DDS рассматривается всеми важными заинтересованными сторонами и на основе различных параметров, таких как оценка рисков, надежность продукта, модульность дизайна, бюджетные и временные ограничения, для продукта выбран наилучший подход к проектированию.

Подход к проектированию четко определяет все архитектурные модули продукта, а также его связь и представление потока данных с внешними и сторонними модулями (если таковые имеются). Внутренний дизайн всех модулей предлагаемой архитектуры должен быть четко определен с мельчайшими деталями в DDS.

Этап 4. Разработка продукта.

На этом этапе SDLC начинается фактическая разработка и сборка продукта. Программный код генерируется в соответствии с DDS на этом этапе. Если дизайн выполнен детально и организованно, генерация кода может быть выполнена без особых хлопот.

Разработчики должны следовать руководящим принципам кодирования, определенным их организацией, и для генерации кода используются такие инструменты программирования, как компиляторы, интерпретаторы, отладчики и т. д. Для кодирования используются различные языки программирования высокого уровня, такие как C, C ++, Pascal, Java и PHP. Язык программирования выбирается в зависимости от типа разрабатываемого программного обеспечения.

Этап 5. Тестирование.

Этот этап обычно является подмножеством всех этапов, так как в современных моделях SDLC тестирование в основном затрагивает все этапы SDLC. Однако этот этап относится только к этапу тестирования продукта, на котором дефекты продукта регистрируются, отслеживаются, исправляются и повторно тестируются до тех пор, пока продукт не достигнет стандартов качества, определенных в SRS.

Этап 6. Развертывание.

Как только продукт протестирован и готов к развертыванию, он официально выпускается на соответствующем рынке.

Иногда развертывание продукта происходит поэтапно в соответствии с бизнес-стратегией этой организации. Продукт может быть сначала выпущен в ограниченном сегменте и протестирован в реальной бизнес-среде (UAT-Пользовательское тестирование).

Затем, основываясь на отзывах, продукт может быть выпущен как есть или с предлагаемыми улучшениями в сегменте таргетинга.

После того, как продукт выпущен на рынок, его обслуживание выполняется для существующей клиентской базы. Само обслуживание может заключаться в поддержке, доработке, обновлении продукта и т.д.

Данная система будет взята за основу при проведении разработки программного обеспечения для оценки качества сетевых ресурсов. Однако, следует заметить, что система взята с некоторыми изменениями, что обуславливает уровень разработки, так как разрабатываемое программное обеспечение не является коммерческим и не требует выхода на рынок.

В третьей главе данной работы мы подробно остановимся на всех этапах, кроме последнего, так как концепция разработки ПО по оценке качества сетевых ресурсов в рамках данной работы не предполагает выход на рынок.

**2 Технические основы разработки ПО для оценки качества**

**сетевых ресурсов**

**2.1 Языки программирования и разметки**

Язык программирования – формальный язык, предназначенный для записи компьютерных программ [36]. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит ЭВМ под её управлением.

В наше время существует огромное количество различных языков программирования как похожих друг на друга, так и различающихся между собой.

Все языки имеют какую-то направленность, т.е. способны решать лишь определенные задачи. Поэтому чаще всего они классифицируются по типу решаемых задач.

В рамках данной работы производство программного обеспечения выполнено на языках программирования Python и Javascript, а также на языке разметки HTML и языке описания внешнего вида документа CSS.

Процесс написания логики программного обеспечения был выполнен на языке Python.

Python – высокоуровневый язык программирования общего назначения, обладающий большой стандартной библиотекой, содержащей множество полезных функций и поддерживающий несколько парадигм программирования [37]. Язык ориентирован на повышение производительности разработчика и читаемости кода.

Python имеет простой и доступный для быстрого изучения синтаксис, освоить который сможет любой человек, у которого есть компьютер и доступ в Интернет.

Python поддерживает несколько парадигм программирования: структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. В языке присутствует динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Программный код на Python организовывается в функции и классы, которые могут объединяться в модули, а они в свою очередь могут быть объединены в пакеты. Python обычно используется как интерпретируемый, но может быть скомпилирован в байт-код Java и в MSIL (в рамках платформы .NET) [38].

По производительности интерпретируемый Python похож на все остальные подобные языки, но возможность компиляции в байт-код позволяет добиться большей производительности [39].

По сравнению с Ruby и некоторыми другими языками, в Python отсутствует возможность модифицировать встроенные классы, такие, как int, str, float, list и другие.

В Python присутствует глобальная блокировка интерпретатора (GIL) –при своей работе основной интерпретатор постоянно использует большое количество потоко-небезопасных данных. В основном это словари, в которых хранятся атрибуты объектов, и обращения к внешнему коду, поэтому, во избежание разрушения этих данных при совместной модификации из разных потоков, перед началом исполнения нескольких инструкций (обычно ста) поток интерпретатора захватывает GIL, а по окончании освобождает.

Существует несколько реализаций языка – CPython (основная), Jython, PyS60, IronPython, Stackless, Python for .NET, PyPy, python-safethread, Unladen Swallow, tinypy [40].

JavaScript – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации ECMAScript (стандарт ECMA-262) [41].

В рамках данной работы язык Javascript используется как связующее звено между логикой, написаной на Python и графическим интерфейсом, написанном на языке разметки HTML и языке описания внешнего вида документа CSS.

Современный JavaScript – это «безопасный» язык программирования. Он не обеспечивает низкоуровневый доступ к памяти или ЦП, поскольку изначально был создан для браузеров, которым такой доступ не требуется [42].

Возможности JavaScript во многом зависят от среды, в которой он работает. Например, Node.js поддерживает функции, позволяющие JavaScript читать и записывать произвольные файлы, выполнять сетевые запросы и т. д.

Встроенный в браузер JavaScript может делать все, что связано с манипулированием веб-страницей, взаимодействием с пользователем и веб-сервером [43].

Например, встроенный в браузер JavaScript может:

* добавить на страницу новый HTML, изменить содержимое существуемого, изменить стили;
* реагировать на действия пользователя, запускать щелчки мыши, движения указателя, нажатия клавиш;
* отправлять запросы по сети на удаленные серверы, скачивать и выгружать файлы (так называемые технологии AJAX и COMET );
* получать и устанавливать cookie, задавать вопросы посетителю, показывать сообщения;
* запоминать данные на стороне клиента («локальное хранилище»).

Язык разметки гипертекста или HTML – это стандартный язык разметки для документов, предназначенных для отображения в веб-браузере [44].

Веб-браузеры получают HTML документы с веб-сервера или из локального хранилища и визуализации документы на мультимедийные веб-страницы [45]. HTML описывает структуру веб-страницы семантически и изначально включает подсказки для внешнего вида документа.

Элементы HTML – это строительные блоки HTML-страниц. С помощью конструкций HTML изображения и другие объекты, такие как интерактивные формы, могут быть встроены в отображаемую страницу. HTML предоставляет средства для создания структурированных документов путем обозначения структурной семантики для текста, такого как заголовки, абзацы, списки, ссылки, цитаты и другие элементы. Элементы HTML выделяются тегами, записанными с использованием угловых скобок.

Разметка HTML состоит из нескольких ключевых компонентов, включая так называемые теги (и их атрибуты), символьные типы данных, символьные ссылки и ссылки на сущности. HTML-теги чаще всего идут парами, например, и, хотя некоторые из них представляют собой пустые элементы и, например, не являются парными. Первый тег в такой паре – это начальный тег, а второй – конечный тег (их также называют открывающими тегами и закрывающими тегами).

Каскадные таблицы стилей (CSS) – это язык таблиц стилей, используемый для описания представления документа, написанного на языке разметки, таком как HTML [46]. CSS – это краеугольная технология всемирной паутины, наряду с HTML и JavaScript.

CSS разработан для разделения представления и содержимого, включая макет, цвета и шрифты. Такое разделение может улучшить доступность контента, обеспечить большую гибкость и контроль в спецификации характеристик представления, позволяет нескольким веб-страницам совместно использовать форматирование, указав соответствующий CSS в отдельном файле .css, что снижает сложность и повторение структурного контента как а также возможность кэширования файла .css для повышения скорости загрузки страницы между страницами, которые совместно используют файл, и его форматирования [47].

Разделение форматирования и содержимого также позволяет представлять одну и ту же страницу разметки в разных стилях для разных методов визуализации, таких как на экране, в печати, голосом (через браузер на основе речи или программы чтения с экрана). В CSS также есть правила для альтернативного форматирования, если доступ к контенту осуществляется на мобильном устройстве.

Каскадирование имен происходит из указанной схемы приоритета, чтобы определить, какое правило стиля применяется, если конкретному элементу соответствует несколько правил. Эта каскадная схема приоритетов предсказуема.

**2.2 Инструменты и библиотеки**

В качестве пакетного менеджера в Python автоматически представлен pip.

Pip – система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python [48].

Система управления пакетами, или менеджер пакетов – набор программного обеспечения, позволяющего управлять процессом установки, удаления, настройки и обновления различных компонентов программного обеспечения.

В данном случае, система управления пакетами «Pip» будет использована нами для загрузки и установки необходимых для работы библиотек и пакетов.

Система управления пакетами «Pip» – это базовый, встроенный в Python менеджер пакетов. Это говорит нам о том, что «Pip» не требует отдельной установки и доступен всем, у кого установлен Python.

При разработке Python-приложений может возникнуть ряд проблем, связанных с использованием библиотек различных версий. Для решения данных проблем используется подход, основанный на построении виртуальных окружений – своего рода песочниц, в рамках которых запускается приложение со своими библиотеками, обновление и изменение которых не затронет другие приложение, использующие те же библиотеки.

Для создания виртуального окружения решено было использовать модуль venv.

Venv – встроенный модуль, это значит, что он не требует установки, однако, стоит сказать, что данный модуль присутствует только в Python3.

Для создания виртуального окружения с именем env с помощью venv необходимо выполнить следующую команду: python -m venv env

В результате будет создан каталог env.

Активация виртуального окружения в Linux выполняется командой: source env/bin/activate; в Windows: env\Scripts\activate.bat.

Деактивация выполняется командой deactivate (работает как в Windows, так и в Linux).

Для отправки HTTP запросов была использована библиотека Requests.

Данная библиотека устанавливается командой «pip install requests». Помимо библиотеки Requests, с помощью данной команды так же установятся библиотеки Urllib3, Idna, Charset и Certifi

Requests позволяют очень легко отправлять запросы HTTP / 1.1. Нет необходимости вручную добавлять строки запроса к вашим URL-адресам или кодировать данные POST. Сохранение активности и пулы HTTP-соединений на 100% автоматические, благодаря urllib3.

Urllib3 – HTTP-клиент для Python, который скачивается автоматически во время скачивания библиотеки Request через pip.

Функции urllib3:

* безопасность потоков;
* пул соединений;
* проверка SSL / TLS на стороне клиента;
* загрузка файлов с многокомпонентной кодировкой;
* помощники для повторных запросов и обработки перенаправлений HTTP;
* поддержка кодирования gzip, deflate и brotli;
* поддержка прокси для HTTP и SOCKS;
* 100% тестовое покрытие.

Протокол IDNA расшифровывается как интернационализированные доменные имена в приложениях. А сама библиотека необходима для поддержки данного протокола.

Charset – это универсальный детектор кодировки символов.

Библиотека Certifi предоставляет тщательно подобранную коллекцию корневых сертификатов Mozilla для проверки надежности сертификатов SSL при проверке идентичности хостов TLS.

Для извлечения данных из HTML кода страниц была использована библиотека Beautiful Soup.

Beautiful Soup – это библиотека Python для извлечения данных из файлов HTML и XML. Он работает с парсером, обеспечивая идиоматические способы навигации, поиска и изменения дерева синтаксического анализа. Данная библиотека скачивается командой «pip install beautifulsoup4».

Библиотека Beautiful Soup – это основная библиотека для написания логики данного программного обеспечения. С помощью этой библиотеки будут находиться все необходимые нам элементы HTML документа обрабатываемой страницы.

Datetime – встроенная библиотека для работы с временем.

Библиотека Eel – это небольшая библиотека Python для создания простых автономных приложений с графическим интерфейсом HTML / JS, подобных Electron, с полным доступом к возможностям и библиотекам Python.

Eel размещает локальный веб-сервер, а затем позволяет аннотировать функции в Python, чтобы их можно было вызывать из Javascript, и наоборот.

Eel разработан, чтобы упростить написание коротких и простых приложений с графическим интерфейсом. Эту библиотеку можно скачать командой «pip install eel»

Re – встроенная библиотека для работы с регулярными выражениями.

Модуль «transliterate» представляет собой двунаправленный транслитератор текста для Python.

Этот модуль транслитерирует unicode строки в соответствии с правилами, указанными в языковых пакетах, то есть заменяет все русские буквы на английские и наоборот по правилам транслитерации.

В этом модуле, для транслитерации букв кириллицы буквами латинского алфавита используются правила, описанные в ISO 9:1995 и ГОСТ 7.79-2000.

Данный модуль можно скачать и установить с помощью команды «python – m pip install – U transliterate».

**3 Процесс разработки**

**3.1 Планирование**

Как уже было сказано ранее в главе 1.2, жизненный цикл разработки программного обеспечения, коротко SDLC, предлагает ряд шагов, которые необходимо выполнить для эффективной разработки программного продукта. SDLC включает в себя следующие этапы:

1. Анализ требований. Отвечает на вопрос «Какие проблемы требуют решений?».
2. Планирование. Отвечает на вопрос «Что необходимо сделать?».
3. Проектирование и дизайн. Отвечает на вопрос «Как добиться поставленных целей?».
4. Разработка ПО. Регулирует процесс создания продукта.
5. Тестирование. Обеспечивает качественную работу продукта.
6. Развертывание. Регулирует использование финального продукта.

Давайте пройдемся по каждому из этих этапов, исключение составит только шестой этап – развертывание, так как концепция разработки программного обеспечения в рамках данной выпускной квалификационной работы не предполагает выход на рынок.

Первый этап – анализ требований.

На этом этапе необходимо будет задуматься о том, кто станет вашими потенциальными пользователями. В целом необходим ответить на следующий вопрос: «Какие проблемы требуют решений?».

Данное программное обеспечение разрабатывается в первую очередь для администраторов и разработчиков сетевых ресурсов, обычному пользователю данное программное обеспечение не будет интересно, так как оно анализирует техническую часть сетевого ресурса и даёт ответ техническим языком, понятным только администраторам и разработчикам сетевых ресурсов.

Отвечая на вопрос: «Какие проблемы требуют решений?», будет сказано, что данное программное обеспечение необходимо для решения проблем, связанных с ранжированием страницы в поисковых системах, что является основным фактором привлечения трафика на страницу, который в свою очередь приносит доход владельцу страницы.

Можно сказать, что проблема низкого положения страницы в поисковой системе является основополагающей при ведении бизнеса в сети Интернет, поэтому мною было решено взять на рассмотрение именно эту проблему.

Второй этап – планирование.

Как только мы определились с проблемой, для которой необходимо решение и целевой аудиторией программного обеспечения, следующим шагом стало четкое определение и документирование требований к продукту и их утверждение от клиента или аналитиков рынка. Это делается с помощью документа SRS (Спецификация требований к программному обеспечению), который содержит все требования к продукту, которые должны быть спроектированы и разработаны в течение жизненного цикла проекта.

Требования к продукту были описаны в главе 1.1, и в разделе 2. Краткое повторение требований:

1. В программном обеспечении представлено 13 параметров оценки качества сетевых ресурсов: фактор наличия ключевых слов, фактор наличия описания, фактор наличия ключевых слов в описании, фактор наличия тегов h1 и ключевых слов в них, фактор наличия тегов h2 и ключевых слов в них, фактор наличия заголовка и ключевых слов в них, фактор наличия внутренних ссылок, фактор наличия внешних ссылок, фактор наличия текущего года в заголовке, фактор наличия ключевых слов в первых 150 словах на странице, фактор оптимизации URL адреса, количество повторений ключевых слов на странице, фактор наличия alt-текста в тегах img.
2. Написано программное обеспечения на двух языках программирования: Javascript и Python, а также на языке разметки HTML и языке стилизации CSS.
3. Используемые библиотеки и инструменты: Pip, Venv, Requests, Urllib3, Idna, Chardet, Certifi, Beautifulsoup, Datetime, Re, Eel.

Четвертый этап – разработка.

Есть два пути разработки:

1. Сначала пишется GUI, потом логика.
2. Сначала пишется логика, потом GUI.

Логика, она же Backend, – это программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части.

Графический интерфейс пользователя (англ. graphical user interface, GUI) – система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т. п.) [49].

В отличие от интерфейса командной строки, в GUI пользователь имеет произвольный доступ (с помощью устройств ввода – клавиатуры, мыши, джойстика и т. п.) ко всем видимым экранным объектам (элементам интерфейса) и осуществляет непосредственное манипулирование ими [50].

Мной был выбран второй путь разработки.

* 1. **Backend**

В логике я прописал все перечисленные ранее критерии оценки качества сетевых ресурсов. Я начал разработку логики с создания виртуального окружения, сделал я с помощью встроенной библиотеки Venv. Для начала я создал папку, в которой будет храниться проект, и через командную строку перешел в неё. В командной строке я ввёл команду: python -m venv env

Этой командой я создал виртуальное окружение.

Дальше необходимо – это виртуальное окружение активировать. Активировал я его с помощь команды: env\Scripts\activate.bat

Следующим моим шагом стало создание файла logics.py, в неё я и прописал всю логику программы.

Первым делом я скачал с помощью пакетного менеджера pip две библиотеки – Beautifulsoup и Requests.

После того, как данные библиотеки скачались, я их импортировал в файл logics.py. Я так же импортировал ещё две библиотеки: re и datetime.

Разработка велась по принципу ООП, поэтому для начала я создал класс QualityControl. В инициализаторе класса всего одна переменная self.url, в которую и будет передаваться ссылка. Также в инициализатор было добавлено подключение к странице через передаваемый url и подключение beautifulsoup.

Первым прописанным критерием стал фактор наличия ключевых слов. С помощью функции .find\_all(‘meta’) библиотеки Beautifulsoup, я нашел все мета теги на странице. Дальше в цикле for с помощью условного оператора if я пробежался по всем мета тегам, нашел мета тег с атрибутом «name=«keywords»« и всё содержимое разделил методом «re.split()», в котором в качестве разделителя использовались символы «()»«;^&.,:/\\|–—-» и добавил его в список. Функция возвращает полученный список.

Также я написал 2 вспомогательных метода. Первый метод с помощью условного оператора «if» возвращает «True» если передаваемый ему список не пустой, и «False» в противном случае. Данный метод необходим будет в дальнейшем, при реализации GUI. Второй метод, названный мною «keywords\_sorted()» сортирует список ключевых слов, исключая из него дубли. Выглядит это следующим образом: в метод «set()» передается список ключевых слов, в котором данный список и сортируется. Однако, у переданного списка тип данных меняется на «множество», и нам необходимо вернуть ему тип данных «список». Для этого мы помещаем всю эту конструкцию в метод «list()». В конечном итоге, полная реализация данного метода выглядит следующим образом: «list(set(keywords))»

Следующим критерием стал фактор наличия описания. Процесс написания такой же, как и в прошлом критерии, только программа ищет метатеги с именем «description».

Третий критерий, наличие ключевых слов в описании, реализован следующим образом: Программа методом set() меняет тип данных на «множество» списку ключевых слов и списоку слов в описании. Дальше, программа собирает слова, которые совпали из двух этих множеств в результирующий список.

Также к этому критерию были написаны метод, который возвращает процент совпадений ключевых слов в полученном результирующем списке и метод, возвращающий словарь, в котором ключом является ключевое слово, а значением является булевый тип данных (True или False) в зависимости от того, есть ли это слово в результирующем списке.

Метод, возвращающий процент совпадений ключевых слов в полученном результирующем списке, выглядит следующим образом: если список совпавших в описании ключевых слов и список ключевых слов не пустые, то программа делит длину списка совпавших в описании ключевых слов на длину списка отсортированных ключевых слов, после чего округляет полученный результат до целых чисел, умножает его на 100, и переводит в тип данных «string», добавляя в конец знак процента и возвращая полученную строку. Если же один из используемых списков пуст, то программа возвращает строку «0%».

Метод, возвращающий словарь с ключевыми словами в качестве ключа и булевыми данными в качестве значения, взаимосвязан с прошлым методом. Он в цикле «for» пробегается по каждому слову из сортированного списка ключевых слов, вызываемого методом «keywords\_sorted», добавляет это слово как ключ в словарь и, если это слово присутствует в списке совпавших в описании ключевых слов, то значением у этого ключа будет «True», если же данное слово отсутствует, то значением ключа будет «False».

Реализация многих следующих критериев очень похожа на реализацию уже описанных трех. Например, реализация критерия наличия тегов h1 и ключевых слов в этих тегах, который является пятым в списке реализованных мною критериев, отличается тем, что если раньше мы искали методом find\_all() тег «meta» и после искали нужный нам конкретный тег с нужным нам атрибутом «keywords» или «description» перебором в цикле, то сейчас мы просто найдем все теги «h1» методом «find\_all(«h1»)» и не будем дальше перебором искать конкретный тег, потому что в данном критерии нам необходимо работать со всеми тегами «h1» страницы.

Реализация метода, возвращающего процент совпадений ключевых слов в теге «h1», и метода, возвращающего словарь с ключевыми словами в качестве ключа и булевыми данными в качестве значения, аналогична реализации этих методов в предыдущих критериях.

Пятый критерий, критерий наличия тегов «h2» и ключевых слов в этих тегах, отличается только тегом. Если в прошлом критерии мы искали тег «h1» методом «find\_all(«h1»)», то в этом критерии мы ищем тег «h2» методом «find\_all(«h2»)». Во всем остальном реализация одинаковая.

Шестой критерий, критерий наличия заголовка и ключевых слов в нем, реализуется аналогично четвертому и пятому, отличие лишь в том, что теперь мы ищем тег «title» методом «find\_all(«title»)».

Для реализации седьмого (наличие внутренних ссылок) и восьмого (наличие внешних ссылок) критериев мною был написан вспомогательный метод, который ищет ссылки на странице. Данный метод сначала находит все теги «a» методом «find\_all(«a»)», после чего извлекает из этих тегов значение атрибута «href», которое и является ссылкой и добавляет его в список. В итоге получается список всех ссылок, имеющихся на странице.

Для нахождения внутренних ссылок я написал метод, который в цикле «for» проверяет элементы списка всех ссылок страницы следующим образом: если ссылка начинается на «/», то она автоматически добавляется в список внутренних ссылок, если же не начинается, то элемент разбивается методом «re.split()» на протокол («http» или «https») , доменное имя (хост) и URL-путь. Метод сравнивает доменное имя элемента с доменным именем текущей страницы, которое он получает из атрибутов класса и так же разбивает методом «re.split()». Если доменное имя элемента равно доменному имени текущей страницы, элемент добавляется в список внутренних ссылок. В итоге, данный метод возвращает список внутренних ссылок.

Для нахождения внешних ссылок мною был написан метод, который схож с методом нахождения внутренних ссылок. Данный метод в цикле «for» проверяет элементы списка всех ссылок страницы следующим образом: если ссылка начинается на «/», то она отбрасывается и цикл переходит к следующему элементу, если же не начинается, то элемент разбивается методом «re.split()» на протокол («http» или «https») , доменное имя (хост) и URL-путь.

Метод сравнивает доменное имя элемента с доменным именем текущей страницы, которое он получает из атрибутов класса и так же разбивает методом «re.split()». Если доменное имя элемента не равно доменному имени текущей страницы, элемент добавляется в список внешних ссылок. В итоге, данный метод возвращает список внешних ссылок.

Девятым по счету реализованным критерием стал критерий наличия текущего года в заголовке. При реализации этого критерия, первым делом я методом «find\_all(«title»)» нашел заголовок, после чего методом «re.split()» на отдельные слова. После этого, я в цикле «for» проверил каждое слово на то, равно ли оно текущему году или нет. Если хоть одно слово будет равно текущему году, то цикл останавливается и «True» становится возвращаемым значением метода, если же ни одно слово не равно текущему году, то возвращаемым значением метода становится «False». Текущий год мы узнаем с помощью метода «datetime.now.year» библиотеки «datetime».

Десятый критерий, критерий наличия ключевых слов в первых ста пятидесяти словах, очень похож в своей реализации на первые шесть критериев.

Для начала мною был написан метод «all\_words», который собирает все слова на странице. Сначала он находит весь текст методом «soup.body», после чего методом «re.split()» разбивает текст на отдельные слова и сохраняет их в списке. Теперь, чтобы получить совпадения по ключевым словам в первых 150 словах страницы, мы должны передать «all\_words[:151]» в метод, который ищет совпадения (он описан в начале главы). Квадратные скобки здесь работают как обрезание списка, они имеют вид «[X:Y:Z]», где X – это левая граница обрезанного списка, Y – правая граница, а Z – шаг. Значение X у нас пустое, потому что нам необходимо обрезать список начиная с самого первого элемента, Y равен 151, потому что список обрезается до элемента, стоящего перед элементом Y, значение Z так же пустое, потому что стандартный шаг равен одному элементу. Метод, показывающий есть ли совпадение по конкретному ключевому, реализуется так же, как описано в начале главы.

В одиннадцатом по счету критерии – критерии оптимизации URL адреса страницы, необходимо реализовать две проверки: проверку на длину пути и проверку на наличие ключевых слов в URL адресе.

Первая проверка реализована следующим образом: программа разбивает URL на протокол, хост и путь, путь разбивается по отдельным элементам, после чего в цикле «for» программа проверяет длину отдельных элементов пути, и если длина элемента больше десяти, то элемент добавляется в список длинных элементов. Если список длинных элементов пустой, то программа возвращает текст: «Все части пути имеет оптимальную длину», если же список полных элементов не пустой, то программа возвращает текст: «Слишком длинные части пути: и сам список после текста».

Для реализации второй проверки я написал метод, который разбивает URL адрес на отдельные слова методом «re.split()», а так же я написал вспомогательный метод транслитерации ключевых слов. Данный метод в цикле «for» проходится по ключевым словам и транслитерирует их на латиницу с помощью метода «translit()» из библиотеки «transliterate». В итоге мы получаем список ключевых слов, переведенных на латиницу. Этот метод нам необходим для проверки ключевых слов в URL адресе страницы, так как в большинстве русскоязычных сайтов URL написан на латинице, а ключевые слова на кириллице. Однако, для случаев, когда URL адрес сайта так же написан на кириллице, я добавил транслитерацию URL адреса в методе поиска и обработки этого адрес. Сам этот метод поиска и обработки выглядит следующим образом: программа берет введенный URL адрес, транслитерирует его на латиницу (если адрес уже написан на латинице, то программа просто пропускает этап транслитерации), после чего разбивает его на отдельные слова и сохраняет в список. Потом, как и во многих других реализованных критериях, этот список транслитерированных слов URL адреса передается вместе со списком транслитерированных ключевых слов в метод поиска совпадения ключевых слов, который возвращает нам список совпавших ключевых слов. После чего список совпавших ключевых слов и список транслитерированных ключевых слов передается в метод расчета процента совпадения, а также в метод, который рассчитывает, какое ключевое слово есть в URL адресе, а какого нет.

Критерий количества повторений ключевых слов на странице я реализовал двенадцатым. На вход поступают два списка: список с отсортированными ключевыми словами и список со всеми словами на странице. Оба эти списка мы реализовали ранее. Здесь необходимо сформировать словарь, в котором ключ – это ключевое слово, а значение – количество повторений ключевого слова на странице. Формируется он следующим образом: в цикле «for» программа проходит по каждому ключевому слову и методом count() проверяет сколько раз слово повторяется в списке всех слов, после чего переносит пару «слово – количество повторений» в словарь.

Последним, тринадцатым по счету, стал критерий наличия alt текста в тегах «img». Данный критерий реализован следующим образом: программа ищет все теги «img» на странице методом «find\_all('img')» и формирует их в отдельный список всех изображений, так же она ищет все теги «img» с атрибутом «alt» методом «find\_all('img', alt=True)» и формирует их в отдельный список изображений с alt текстом. После чего, методом «len()» определяется длина каждого списка и создается переменная «img\_without\_alt\_num», в которую записывается разность длины списка всех изображений и длины списка изображений с alt текстом. Если переменная «img\_without\_alt\_num» равна нулю, метод возвращает текст: «Все изображения имеют alt текст», если же она больше нуля, то метод возвращает текст: «Количество изображений без alt текста: «и саму переменную «img\_without\_alt\_num».

Резюмируя процесс реализации внутренней части данного программного обеспечения, можно сказать, что, в целом, процесс реализации прошел без особых трудностей и не требует широких познаний в области программирования и структуры HTML документа страницы. Больше половины реализованных критериев (семь штук) имеют практически одинаковую схему реализации: те же методы поиска и структуризации данных, те же методы расчета процента наличия ключевых слов, и те же методы расчета совпадения по конкретным ключевым словам. Остальные шесть критериев хоть и имеют уникальную схему реализации, однако реализация данных критериев с уникальной схемой реализации также не вызывала особых сложностей.

**3.3 Graphical User Interface**

Как уже было сказано в главе 2.2, для реализации графического интерфейса была использована связка из библиотеки для языка программирования Python под названием «Eel», языка программирования JS, а также языка разметки HTML и формального языка описания внешнего вида документа CSS.

Библиотека «Eel» послужила для инициализации окна программного обеспечения, а также для передачи возвращаемых данных из методов логики.

Для начала, был создан документ «main.py», в который были ипортированы библиотека «Eel» и класс «QualityControl» с методами логики из документа «logic.py».

После импорта всех необходимых нам зависимостей, первым делом методом «eel.init('web')» было инициализировано окно программного обеспечения. После инициализации окна для каждого метода логики был написан одноименный метод, который принимал на вход переменную «url», инициализировал класс «QualityControl» с принятой переменной в качестве атрибута и возвращал значение метода логики.

Ключевой особенностью этих методов является декоратор «@eel.expose», который необходим для передачи значений из Backend во Frontend часть данного программного обеспечения.

Последним, что было сделано в этом документе, стало добавление метода «eel.start('main.html', size=(700, 700))», где «main.html» – это название HTML документа, который необходимо запускать при старте работы программного обеспечения, и в который необходимо передавать данные из Backend части ПО, а «size=(700, 700)» - это указание размера запускаемого окна по ширине и высоте в пикселях.

Следом после создания и заполнения документа «main.py», была создана папка «web», в которой были созданы три документа: «main.html», «style.css» и «logo.ico».

Файл «main.html» – это главный HTML файл программного обеспечения, в котором производится создания и разметка элементов графического интерфейса.

Файл «style.css» – это CSS файл, подключаемый к HTML документу, в котором создаются стили для элементов ПО.

Файл «logo.ico» – это логотип программного обеспечения в виде иконки, созданный в приложении «Photoshop» и имеющий вид трех букв «SQC» зеленого цвета на прозрачном фоне.

Файл «main.html» в IDE «Pycharm», который я использовал при создании программного обеспечения, создается сразу с базовой структурой, то есть уже имеет теги «<DOCTYPE html>», «<html lang="en">», «<head>», «<body>» и теги «<meta charset="UTF-8">» и «<title>« в теге «<head>».

Первым делом, я поменял язык с английского на русский, чтобы при запуске ПО не вызывался системный попап с переводчиком, и поменял заголовок на «Site Quality Control», который будет показываться в верхней части окна ПО.

Следом, я подключил в теге «<head>» все необходимые нам зависимости.

Сначала я подключил JS версию библиотеки «Eel» с помощью тега «<script src="eel.js"></script>». Следом, тегом «<link rel="icon" type="image/png" href="/logo.ico">» я подключил логотип. Потом я подключил CSS файл тегом «<link rel="stylesheet" href="main.css">». Следом за CSS файлом я подключил шрифт «Roboto». И последним я подключил набор функций «JQuery» для более удобной работы с JavaScript.

После подключения всех необходимых зависимостей, в теге «<body>» я создал поле для ввода ссылки на проверяемую страницу, кнопку «Начать», которая запускает обработку проверяемой страницы и общий контейнер «<div>», в котором будет выводиться результат обработки страницы. В общем контейнере я создал отдельные контейнеры для каждого метода логики.

После чего я перешел к созданию стилей в документе «style.css». Так как я не обладаю серьезными навыками веб-дизайнера, я не стал заострять особое внимание на стилистики ПО.

В данном файле я подключил шрифт для всего текста на странице, написал градиентный фон для приложения с помощью специализированных ресурсов, немного стилизовал поле для ввода и кнопку, и, наконец, написал два класса: «red» и «green», в которых текст становится красным и зеленым соответственно.

После стилизации программного обеспечения, я написал JS скрипт, который перенесет логику в GUI. В теге «<body>» я создал тег «<script>», в котором и написан весь JavaScript код.

Реализацию переноса логики я начал с написания функции «get\_url», которая получает ссылку из поля для ввода и возвращает ее. Данная функция нужна будет для каждого метода логики, так как инициализация класса требует передачи ссылки на обрабатываемую страницу.

После чего я создал функцию «display», в которую буду передавать данные, и JQuery функцию, которая начинает работать после клика на кнопку «Начать», и в которой я написал метод очистки страницы от старых данных, а также передал в нее функцию «display». То есть, после нажатия на кнопку «Начать», страница очищается, и уже после очистки страницы начинает работать функция «display».

В самой функции «display» я создал переменную «url», которая хранит в себе значение функции «get\_url». После чего я создал переменные для каждого метода логики. Шаблон создания такой переменной имеет вид:

«let {variableName} = await eel.{logic\_method\_name}(url)();», где {variableName} – это название переменной, а {logic\_method\_name} – это название Python метода из логики ПО, названия указываются без фигурных скобок.

Так же я создал для каждого контейнера метода переменную, которая хранит в себе ссылку на этот контейнер.

Теперь необходимо значение из каждой переменной метода логики вывести в соответствующий контейнер. Вывод значения имеет следующий вид:

«{containerVariableName}.innerHTML = {logicVariableName};», где {containerVariableName} - название переменной, которая хранит в себе ссылку на контейнер, а {logicVariableName} - название переменной, которая хранит в себе данные метода логики, названия указываются без фигурных скобок.

Для переменных, которые хранят в себе булевые данные, я написал функцию, которая с помощью условного оператора «if» в зависимости от «true» или «false» отправляет определенный поясняющий текст в контейнер метода и окрашивает этот текст в зеленый либо красный цвет.

Аналогично я сделал для методов, которые хранят в себе процент наличия ключевых слов: если процент больше условной отметки (в некоторых случаях 49%, в некоторых 9%), то процент окрашивается в зеленый, если же нет, то в красный. Также, если процент равен нулю, то переменная, которая хранит в себе соответствие ключевых слов, по конкретным словам, соответствующего критерия, не передается в контейнер.

Последнее что было сделано в разработке графического интерфейса пользователя – это группировка данных по критериям проверки в виде аккордеона.

Аккордеон представляет собой список, каждый элемент которого может быть развернут или свернут [52]. Использование аккордеона позволяет реализовать удобную навигацию для страниц с большим содержанием.

Реализуется аккордеон как кнопка, которая после нажатия скрывает или раскрывает определенный контейнер.

В HTML добавляется тег «<button>» с классом «accordion» и контейнер «<div>» с классом «area».

В CSS добавляются стили: цвета, значки, ховеры.

Так же в HTML в теге «<script>» добавляется JavaScript код, который работает следующим образом: при нажатии на кнопку с классом «accordion», контейнер «<div>» с классом «area» показывается, а при повторном нажатии контейнер скрывается.

Если описывать этот скрипт точнее, то он сначала находит все элементы с классом «accordion» и формирует из них список. После чего, в цикле проходится по каждому найденному элементу и регистрирует обработчик событий методом «.addEventListener(“click”, finction()».

Данный обработчик событий работает следующим образом: если следующий элемент после кнопки (который и является контейнером для аккодреона) открыт, то есть имеет максимальную высоту, то его высота становится «null» или, попросту, нулевой, т.е. элемент скрывается. Если же его высота равна нулю, то она становится максимальной, т.е. элемент раскрывается и страница скроллится к этому элементу.

Финальный вид разработанного в ходе данной ВКР графического пользовательского интерфейса можно увидеть в приложении А.

В приложении А на рисунке А.1 представлен графический интерфейс в свернутом виде, а на рисунке А.2 в развернутом со всей обработанной информацией. Также следует обратить внимание, что на рисунке А.1 белый фон элемента «Ключевые слова» в данном случае является ховером, т.е. появляется при наведении курсора на элемент.

По итогу реализации GUI важно отметить, что изначально данный вариант графического интерфейса не подразумевался как финальный. Он был создан исключительно для того, чтоб показать концепт того, как может выглядеть пользовательский интерфейс на данном этапе разработки.

**3.4 Тестирование и сборка**

Проект такого уровня, как данное программное обеспечение, не имеет смысла тестировать во всех фазах тестирования программы, девяносто процентов тестов были проведены вручную в процессе разработки на выборке из нескольких страниц с разной степенью соответствия проверяемых критериев, благодаря чему найденные ошибки исправлялись сразу же.

После окончания разработки я увеличил выборку и провел финальное тестирование готового продукта с целью найти возможные упущения и исправить их. Также я подготовил тестовый HTML документ, в котором собирал комбинации из разных критериев и проверял ПО на этом документе.

Когда тестирование закончилось, я приступил к сборке. В данном случае, под сборкой подразумевается объединение всех файлов в один проект и создание исполняемого файла данного программного обеспечения.

Исполняемый файл необходим для того, чтоб любой человек, не имея на своем компьютере язык программирования Python, все равно смог запустить ПО.

Создание исполняемого файла происходит с помощью библиотеки «Pyinstaller».

Первым делом, данную библиотеку необходимо установить. Сделать это можно командой: «pip install pyinstaller». После чего, необходимо открыть в консоли папку проекта и ввести следующую команду:

«python -m eel main.py web --onefile --windowed --name sqc --icon sqc.icns»

Данная команда позволяет собрать проект так, как необходимо в данном случае разработчику или какие ему ставят требования.

Эта команда имеет много опций. Например, можно собрать проект в один исполняемый файл или же в исполняемый проект со всеми элементами и файлами настройки. Основные опции представлены ниже.

Опция «--onefile» позволяет собрать проект в один исполняемый файл, минуя папки с зависимостями.

Опция «--windowed» необходима для приложений с графическим интерфейсом. Она отключает консоль при запуске программы.

Опция «--name» дает исполняемому файлу название. Как видно из примера, я назвал исполняемый файл «sqc».

Опция «--icon» меняет стандартный значок исполняемого файла на тот, который необходим разработчику.

**ЗАКЛЮЧЕНИЕ**

В век доступного массовому пользователю Интернета важность высококачественной настройки поисковой оптимизации очевидна и бесспорна. В данной выпускной квалификационной работе были представлены и реализованы на практике в виде программного обеспечения технические критерии оценки качества сетевых ресурсов.

Основные задачи ВКР – изучение принципов ранжирования страниц, изучение требования к реализации отдельных внутренних факторов SEO, создание плана разработки ПО по оценке качества сетевых ресурсов, разработка Backend’а и GUI программного обеспечение, тестирование готового продукта.

На основе анализа изученных принципов ранжирования страниц были выделены тринадцать критериев оценки качества реализации поисковой оптимизации страницы, которые возможно реализовать с учетом имеющихся материально-технической базы, опыта программирования и информации в свободном доступе.

Для описания проектируемой системы и поддержки реализации был описан план разработки.

В логической части программного обеспечения спроектированы выделенные ранее критерии оценки качества сетевых ресурсов, а в графической части реализовано отображение полученной и обработанной информации в человекопонятный вид.

В конце работы было проведено тестирование на выборке из нескольких страниц с разной степенью соответствия проверяемых критериев, благодаря чему были найдены и исправлены некоторые ошибки, допущенные при разработке.

Дальнейшее развитие разработанного программного обеспечения необходимо по направлению к увеличению его функциональности: количеству обрабатываемый критериев, улучшение информативности обработанной информации, разработке дизайна, а также дальнейшей интеграции в него систем монетизации.

В частности, необходима дальнейшая разработка и расширение разработанного программного обеспечения, а конкретнее, внедрение более сложных систем анализа данных с целью обеспечения более объективной оценки качества системы ранжирования и предоставления более точных рекомендаций по повышению уровня ранжирования страницы в выдаче браузера.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Яковлев, А. Б. Раскрутка и продвижение сайтов: основы, секреты, трюки / А. Б. Яковлев. – Санкт-Петербург : БХВ-Петербург, 2017. – 26 с. – ISBN 978-5-94157-751.
2. Ашманов, И. Б. Оптимизация и продвижение сайтов в поисковых системах / Б. И. Ашманов. – Санкт-Петербург : Питер, 2019. – 282 с. – ISBN 654-3-892746-092.
3. Агеева, Ю. А. SEO , SMO И SMM: Что лучше для продвижения товаров и услуг? / Ю. А. Агеева // Научное сообщество студентов XXI столетия : экономические науки: материалы XX Международной студенческой научно-практической конференции № 5. – Новосибирск : СибАК, 2018. – С. 8-10.
4. Бабаев, А. П. Секреты эффективного продвижения сайтов / А. П. Бабаев. – Санкт-Петербург : Питер, 2016. – 272 с. – ISBN 835-6-892047-734.
5. Байков, В. Д. Интернет. Поиск информации. Продвижение сайтов / В. Д. Байков. – Санкт-Петербург : Питер, 2019. – 245 с. – ISBN 255-2-35724-244.
6. Бэттелл, Д. Поиск. Как компания Google и ее конкуренты переписали законы бизнеса и изменили нашу культуру / Д. Бэттелл. – Москва : Добрая книга, 2019. – 368 с. – ISBN 234-2-5828-235.
7. Вебер, Л. Эффективный маркетинг в Интернете. Социальные сети, блоги, Twitter и другие инструменты продвижения в Сети /  
    Л. Вебер. – Москва : Манн, Иванов и Фербер, 2017. – 320 с. – ISBN 842-1-27273-148.
8. Веселкова, Т. В. Эффективная эксплуатация сайта. Практическое пособие / Т. В. Веселкова. – Москва : Дашков и К, 2018. – 176 с. – ISBN 724-65-3592-19.
9. Вирин, Ф. Б. Интернет-маркетинг. Полный сборник практических инструментов / Ф. Б. Вирин. – Москва : Эксмо, 2018. – 224 с. – ISBN 432-2-32848-136.
10. Геддс, Б. Google AdWords: исчерпывающее руководство / Б, Геддс. – Москва : Манн, Иванов и Фербер, 2018. – 624 с. – ISBN 253-5-35932-113.
11. Ашманов, И. А. Оптимизация и продвижение сайтов в поисковых системах / И. А. Ашманов. – Санкт-Петербург : Питер, 2019. – 83 с. – ISBN 356-3-26895-234.
12. Гроховский, Л. А. SEO: руководство по внутренним факторам / Л. А. Гроховский. – Москва : Манн, Иванов и Фербер, 2016. – 133 с. – ISBN 398-6-86924-464.
13. Шафер, С. HTML, XHTML и CSS. Библия пользователя   
    / С. Шафер. – Санкт-Петербург : Питер, 2017. – 40 с. – ISBN 754-4-56934-367.
14. Севостьянов, И. Б. Поисковая оптимизация. Практическое руководство по продвижению сайта в Интернете / И. Б. Севостьянов. – Санкт-Петербург : Питер, 2016. – 59 с. – ISBN 893-4-69258-472.
15. SEO: в чем разница? // Webopedia: [сайт]. – 2018. – URL: <https://www.webopedia.com/insights/sem-vs-seo/> (дата обращения: 14.05.2022).
16. Гуров, Ф. Л. Продвижение бизнеса в сети Интернет. Все о PR и рекламе в Сети / Ф. Л. Гуров. – Москва : Манн, Иванов и Фербер, 2018. – 172 с. – ISBN 865-6-45735-674.
17. Джонс, Г. Кликология. Психология онлайн-шоппинга для привлечения покупателей / Г. Джонс. – Москва : Эксмо, 2018. – 320 с. – ISBN 903-4-35672-572.
18. On-page SEO. // Moz : [сайт]. – 2019. – URL: <https://moz.com/beginners-guide-to-seo/on-page-seo> (дата обращения: 13.05.2022).
19. Джонс, К. 140 технологий раскрутки сайтов / К. Джонс. – Москва : Рид Групп, 2017. – 352 с. – ISBN 356-2-85673-285.
20. Ванюшкин, А. С. Методы и алгоритмы извлечения ключевых слов / А. С. Ванюшкин. Москва : Эксмо, 2018. – 320 с. – ISBN 934-4-26245-542.
21. Евдокимов, Н. В. Основы контентной оптимизации. Эффективная интернет-коммерция и продвижение сайтов с Интернете / Н. В. Евдокимов. – Москва : Вильямс, 2019. – 160 с.– ISBN 466-5-63455-674.
22. Загуменов, А. П. Как раскрутить и разрекламировать Web-сайт в сети Интернет / А. П. Загуменов. – Москва : ДМК Пресс, 2015. – 384 с. – ISBN 643-5-23545-357.
23. Монти, Р. Google: теги заголовков – сильный сигнал / Р. Монти. –   
    Москва : Вильямс, 2019. – 123 с.– ISBN 865-9-53357-643.
24. Конолли, Д. Веб адреса в HTML 5 / Д. Конолли. – Москва : Рид Групп, 2017. – 342 с. – ISBN 452-1-235631-124.
25. Зиссер, Ю. А. Маркетинг on-line. Как превратить сайт компании в эффективный инструмент продаж / Ю. А. Зиссер. – Минск : Гревцов Паблишер, 2017. – 304 с. – ISBN 345-3-453-45.
26. Крохина, О. И. Первая книга SEO-копирайтера. Как написать текст для поисковых машин и пользователей / О. И. Крохина. – Москва : Инфра-Инженерия, 2017. – 216 с. – ISBN 754-5-624684-89.
27. Маннинг, К. Д. Введение в информационный поиск / К. Д. Маннинг. – Москва : Вильямс, 2019. – 528 с. – ISBN 7527-3-52582-83.
28. Маршалл, П. Контекстная реклама, которая работает: Библия Google AdWords / П. Маршалл. – Москва : Манн, Иванов и Фербер, 2018. – 464 с. – ISBN 421-9-23953-41.
29. Ахарон, Ш. Исходящие ссылки, используемые в качестве сигнала ранжирования / Ш. Ахарон. – Москва : Рид Групп, 2020. – 12 с. – ISBN 532-4-53111-42.
30. Фримен, Э. Изучаем HTML, XHTML и CSS   
    / Э. Фримен. – Санкт-Петербург : Питер, 2017. – 656 с. – ISBN 345-4-3512-342.
31. Питер, Л. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений / Л. Питер. – Москва :  
    Вильямс, 2018. – 272 с. – ISBN 352-8-5773-32.
32. Соммервилл, И. Инженерия программного обеспечения / И. Соммервилл. – Москва : Рид Групп, 2020. – 18 с. – ISBN 422-56-5641-3.
33. ГОСТ 19.101-2010. Единая система программной документации. Виды программ и программных документов. Общие требования и правила составления : национальный стандарт Российской Федерации : издание официальное : утвержден и введен в действие Постановлением Государственного комитета стандартов Совета Министров от 20 мая 2010 года №1268-ст : введен впервые : дата введения 2010-08-01 / подготовлен Информационным агентством России, филиалов Российской книжной палаты, Российской государственный библиотекой. – Москва : Стандартинформ, 2010. – 15с.
34. ГОСТ Р 50949-01-2012. Средства отображения информации индивидуального пользования. Методы измерений и оценки эргономических параметров и параметров безопасности. Общие требования и правила составления : национальный стандарт Российской Федерации : издание официальное : утвержден и введен в действие Постановлением Госстандарта России от 25 декабря 2011 года №576-ст : введен впервые : дата введения 2012-02-01 / подготовлен Автономной некоммерческой организацией «Научно-технический Центр сертификации электрооборудования», Московским государственным институтом электроники и математики (технический университет) с участием ВНИИ стандарта, Научного Центра социально-производственных проблем охраны труда – Москва : Стандартинформ, 2012. – 25 с.
35. Братищенко, В. В. Проектирование информационных систем / В. В. Братищенко. – Иркутск : БГУЭП, 2018. – 84 с. – ISBN 39-5311-31-5.
36. Брауде, Э. Дж. Технология разработки программного обеспечения / Э. Дж. Брауде. – Санкт-Петербург : Питер, 2019. – 656 с. ISBN 390-2-41470-36.
37. Гавриков, М. М. Теоретические основы разработки и реализации языков программирования / М. М. Гавриков. – Москва : Манн, Иванов и Фербер, 2018. – 42 с. – ISBN 499-2-38918-41.
38. Сваруп, К. Программирование на языке Python / К. Сваруп. – Москва : ДМК Пресс, 2020. – 56 с. – ISBN 321-5-3379-8.
39. Бизли, Д. Справочное руководство по языку программирования Python / Д. Бризли. – Санкт-Петербург : Питер, 2016. – 456 с. – ISBN 83-7-3092-452.
40. Лутц, М. Язык Python / М. Лутц. – Москва : Манн, Иванов и Фербер, 2019. – 152 с. – ISBN 929-8-42145-89.
41. Кеннет, Р. Автостопом по Python / Р. Каннет. – Санкт-Петербург : Питер, 2017. – 46 с. – ISBN 47-4-3422-32.
42. Уайт, М. Справочник программиста JavaScript / М. Уайт. – Санкт-Петербург : Питер, 2019. – 16 с. – ISBN 32-2-90876-38.
43. Закас, Н. Высокопроизводительный JavaScript / Н. Закас. – Москва : Рид Групп, 2020. – 33 с. – ISBN 43-6-57921-21.
44. Флэнаган, Д. Карманный справочник. Сделайте веб-страницы интерактивными! /Д. Флэгнан. – Москва : Вильямс, 2015. – 198 с. – ISBN 54-2-58161-34.
45. Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений / П. Лабберс, О. Брайан. – Москва : Вильямс, 2018. – 58 с. – ISBN 83-1-36902-26.
46. Шафер, С. HTML, XHTML и CSS. Библия пользователя / С. Шафер. – Москва : Диалектика, 2018. – 656 с. – ISBN 92-7-91039-40.
47. Титтел, Э. HTML, XHTML и CSS для чайников / Э. Титтел, Д. Ноубл. – Москва : Диалектика, 2016. – 35 с. – ISBN 43-1-38961-27.
48. Макфарланд, Д. Новая большая книга CSS / Д. Макфарланд. – Санкт-Петербург : Питер, 2017. – 101 с. – ISBN 89-5-78256-83.
49. Марлет, К. Python / К. Марлет. – Москва : Вильямс, 2016. – 230 с. – ISBN 34-6-246863-26.
50. Графический интерфейс пользователя // Wiki : [сайт]. – 2018. – URL: <https://www.wikiru.wiki/blog/en/Graphical_interface> (дата обращения: 09.05.2022).
51. Графический интерфейс против командной строки // Wiki : [сайт]. – 2018. – URL: https://www.wikiru.wiki/blog/en/Graphical\_interface (дата обращения: 10.05.2022).
52. Аккордеон. Турбо-страницы для отдельный сайтов // Турбо-страницы Яндекс : [сайт]. – 2019. URL: <https://yandex.ru/dev/turbo/doc/rss/elements/accordion.html> (дата обращения: 10.05.2022).

**ПРИЛОЖЕНИЕ А**

**Графический интерфейс ПО**



Рисунок А.1 – Графический интерфейс ПО в свернутом виде



Рисунок А.2 – Графический интерфейс ПО в частично развернутом виде