

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

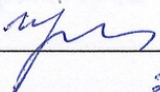
ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК
Кафедра программного обеспечения

РЕКОМЕНДОВАНО К ЗАЩИТЕ В ГЭК
И ПРОВЕРЕНО НА ОБЪЕМ
ЗАИМСТВОВАНИЯ

Заведующий кафедрой

д.п.н., профессор

И.Г. Захарова


29 июня 2018 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Разработка системы мониторинга базовых параметров веб-страниц сайта

02.04.03. Математическое обеспечение и администрирование информационных систем

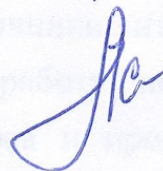
Магистерская программа «Высокопроизводительные вычислительные системы»

Выполнил работу
Студент 2 курса
очной формы обучения



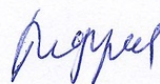
Пермин
Александр
Дмитриевич

Научный руководитель
к.ф.-м.н., доцент



Ступников
Андрей
Анатольевич

Рецензент
к.фил.н., доцент каф. ИС
ФГБОУ ВО «ТюмГУ»



Бидуля
Юлия
Владимировна

Тюмень 2018

Оглавление

Введение	3
Глава 1. Общие подходы к тестированию приложений	5
1.1 Основные определения	5
1.2 Виды тестирования	6
1.2.1 Системное тестирование	6
1.2.2 Приемочное тестирование	7
1.2.3 Модульное тестирование	7
1.2.4 Интеграционное тестирование	8
1.3 Особенности тестирования web-приложения	10
1.4 Требования к разрабатываемому приложению	11
1.5 Аналоги	12
Глава 2. Автоматизация тестирования веб-приложения	14
2.1 Обзор используемых технологий	14
2.2 Алгоритм тестирования веб-страниц.	15
2.2.1 Оценка скорости загрузки страниц веб-приложения.	18
2.2.2 Интеграция с сервисами Google Analytics и Яндекс.Метрика	20
2.3 Формирование и отправка отчетов о прохождении тестирования	22
2.3.1 Формирование отчета	22
2.3.2 Отправка уведомлений на электронный адрес	24
2.3.3 Отправка уведомлений в Telegram	24
Глава 3. Применение разработанной системы мониторинга	25
Заключение	29
Список литературы	30
Приложение А - Пример теста компонента “Breadcrumbs”	32
Приложение Б - Пример теста по открытию фильтра	33
Приложение В - Определение картинок на странице	35
Приложение Г - Функция отправки отчета о тестировании	36
Приложение Д - Функция получения данных из Яндекс.Метрика	37
Приложение Е – Формат файла конфигурации системы мониторинга	38

Введение

В настоящее время веб-приложения являются неотъемлемой частью успешного развития бизнеса. Однако важно не только создать удобный сайт для пользователей, но и проводить мониторинг качества его веб-страниц и состояния работы всего приложения.

Сложность программного обеспечения возрастает с невероятной скоростью. Разработка программного обеспечения, как и любая человеческая деятельность, практически невозможна без ошибок. Насколько хорошо бы не был написан код изначально, в него вносятся изменения, связанные либо с исправлением существующих ошибок, либо с добавлением дополнительной функциональности. Так или иначе данные модификации могут привести к появлению различных дефектов.

Для поддержания качества веб-приложения применяются различные виды тестирования:

- системное тестирование;
- модульное тестирование;
- интеграционное тестирование;
- приемочное тестирование

Для оптимизации временных и человеческих ресурсов, затрачиваемых на проведение тестирования, производится его автоматизация. Часто процесс тестирования выполняется в момент разработки нового функционала или изменения существующего, но также может производиться мониторинг работающего продукта.

Мониторинг веб-приложения подразумевает постоянное отслеживание соответствия параметров контента и скорости функционирования заранее

определённым значениям. Это требует наличия программного продукта, который бы взял на себя обязанность отслеживания данного соответствия.

В связи с необходимостью постоянного мониторинга работы веб-приложения была поставлена следующая цель:

Разработать автоматизированную систему тестирования, которая позволит отслеживать сбои в работе веб-сайта, которые невозможно или трудно отследить на этапе тестирования.

Для достижения поставленной цели требуется решить следующие задачи:

1. Выявить наиболее распространенные проблемы, которые невозможно или трудно отследить на этапе тестирования.
2. Изучить существующие продукты для тестирования веб-приложений.
3. Разработать алгоритм тестирования веб-страниц.
4. Реализовать процедуру оценки скорости веб-страниц.
5. Разработать приложение для тестирования веб-страниц и оповещения о найденных сбоях.

Глава 1. Общие подходы к тестированию приложений

1.1 Основные определения

Тестирование – это одна из техник контроля качества, включающая в себя активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов.

Мониторинг – система постоянного наблюдения за процессами, результаты которого служат для обоснования управленческих решений.

Тестовый случай (Test Case) – это артефакт, описывающий совокупность шагов, конкретных условий и параметров, необходимых для проверки реализации тестируемой функции или её части. [4]

Проверка кода (Code Review) – анализ (инспекция) кода с целью выявить ошибки, недочеты, расхождения в стиле написания кода, в соответствии написанного кода и поставленной задачи.

Линтер (Lint) – программа для проведения статического анализа кода.

Валидация (Validation) – это процесс определения соответствия разрабатываемого ПО ожиданиям и потребностям пользователя, а также требованиям к системе.

Непрерывная интеграция (Continuous Integration, CI) - это практика разработки программного обеспечения, при которой разработчики регулярно объединяют изменения программного кода в центральной репозитории, после чего автоматически выполняется сборка, тестирование и запуск.

Шпионы (Spy) – используются для получения информации о вызовах функций.

Заглушки (Stub) – замена целевых функций для управления форсировать какие-то события в коде или предотвратить вызовы внешних ресурсов.

Моки (Mock) – поддельные методы с заранее запрограммированным поведением и соглашениями.

1.2 Виды тестирования

1.2.1 Системное тестирование

Основной задачей системного тестирования является проверка как функциональных, так и нефункциональных требований в системе в целом. При этом выявляются дефекты, такие как неверное использование ресурсов системы, непредусмотренные комбинации данных пользовательского уровня, несовместимость с окружением, непредусмотренные сценарии использования, отсутствующая или неверная функциональность, неудобство использования и т.д. Для минимизации рисков, связанных с особенностями поведения в системы в той или иной среде, во время тестирования рекомендуется использовать окружение максимально приближенное к тому, на которое будет установлен продукт после выдачи.

Можно выделить два подхода к системному тестированию:

- на базе требований

Для каждого требования пишутся тестовые случаи (test cases), проверяющие выполнение данного требования.

- на базе случаев использования

На основе представления о способах использования продукта создаются случаи использования системы. По конкретному случаю использования можно определить один или более сценариев. На проверку каждого сценария пишутся тестовый случаи (test cases), которые должны быть протестированы.

1.2.2 Приемочное тестирование

Приемочное тестирование – это комплексное тестирование, необходимое для определения уровня готовности системы к последующей эксплуатации. Тестирование проводится на основании набора тестовых сценариев, покрывающих основные бизнес-операции системы.

Обычно данный вид тестирования реализуется конечными пользователями системы, однако привлечение опытных тестировщиков сократит время на подготовку к тестированию и позволит повысить качество и надежность проводимых испытаний.

Направления приемочного тестирования

- Операционное тестирование - проверка системы на способность выполнять свою роль в среде эксплуатации согласно бизнес-модели
- Пользовательское тестирование - проверка пригодности системы для внедрения конечными пользователями
- Альфа-тестирование - проверка независимой командой тестирования
- Бета-тестирование - тестирование внешними пользователями, потенциальными клиентами

1.2.3 Модульное тестирование

Модульное тестирование проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы по-отдельности (модули программ, объекты, классы, функции и т.д.). Обычно модульное тестирование проводится, вызывая код, который необходимо проверить и при поддержке сред разработки, таких как фреймворки для модульного тестирования или инструменты для отладки.

Один из наиболее эффективных подходов к модульному тестированию - это подготовка автоматизированных тестов до начала основной разработки программного обеспечения. Это называется разработка от тестирования (test-driven development) или подход тестирования вначале (test first approach). При этом подходе создаются и интегрируются небольшие куски кода, напротив которых запускаются тесты, написанные до начала кодирования. Разработка ведется до тех пор, пока все тесты не будут успешно пройдены.

Каждый модульный тест имеет следующую структуру:

1. Настройка теста
2. Вызов тестируемого метода
3. Утверждение

При использовании модульного тестирования при написании тестов используются шпионы (spy), заглушки (stub) и моки (mock). Они необходимы, когда отдельный модуль необходимо изолировать от внешних ресурсов.

1.2.4 Интеграционное тестирование

Интеграционное тестирование – вид тестирования, при котором на соответствие требований проверяется интеграция модулей, их взаимодействие между собой, а также интеграция подсистем в одну общую систему. Для интеграционного тестирования используются компоненты, уже проверенные с помощью модульного тестирования, которые группируются в множества. Данные множества проверяются в соответствии с планом тестирования, составленным для них, а объединяются они через свои интерфейсы.

Этапы интеграционного тестирования:

- Разработка тест-плана – руководства к действию для тестировщиков;
- Формирование тестовых данных и создание тест-кейсов;
- Реализация сценариев для запуска тест-кейсов;
- Выполнение тест-кейсов и исправление ошибок;
- Повторение цикла тестирования до успешной интеграции

Таким образом, существует достаточно развитая система проведения тестирования, определены многие его виды и этапы.

Для задачи текущего мониторинга наиболее подходящими являются интеграционное и приемочное тестирование, так как проверяется не отдельные подсистемы, а вся система целиком, происходит проверка пригодности системы для конечного пользователя.

1.3 Особенности тестирования web-приложения

Изменения в веб-приложения вносятся при разработке нового функционала и изменении существующего. Обычно в крупных проектах используются практика разработки программного обеспечения - Continuous Integration (CI), при которой команда не реже одного раза в день проводит интеграцию, используя статический анализ кода (различные линтеры, code review - проверка качества кода) и автотесты: модульные (Приложение А), интеграционные (Приложение Б).

Но изменения вносятся не только разработчиками, а также специалистами технической поддержки, поэтому автоматизированного тестирования при написании кода программистами недостаточно. Специалисты технической поддержки могут заменять ссылки, загружать картинки на существующие страницы или создавать новые. Также сайт может получить изменения при обновлении базы данных.

В результате изменений на сайте могут появиться различные дефекты и ошибки, после обновления страниц необходимо проверить их на наличие:

- изображений большого объема;
- битых ссылок;
- редиректов;
- некорректных мета-данных;
- ошибки при выполнении скриптов;
- увеличение времени загрузки страницы.

1.4 Требования к разрабатываемому приложению

Для регулярного отслеживания ситуаций, описанных в предыдущем параграфе, который можно отнести к ошибкам разрабатываемого веб-приложения, очевидно, требуется разработать специальное приложение.

Разрабатываемое приложение должно просто настраиваться, быть легким в использовании обычным пользователем, не имеющим знаний в настройке серверов или веб-разработке.

В приложении должна быть предусмотрена возможность настроить параметры для веб-страниц, для которых нужно проводить мониторинг:

- изображений большого объема;
- битых ссылок;
- редиректов;
- некорректных мета-данных;
- ошибки в скриптах;
- увеличение времени загрузки страницы.

В системе должна быть возможность протестировать несколько страниц, заданных пользователем, или указать какое количество страниц необходимо протестировать. Для этого нужно указать главную страницу сайта, с которого начнется тестирование, а остальной набор страниц формируется из случайных уникальных адресов сайта.

Приложение должно формировать отчеты о прохождении тестирования и иметь возможность отправить их на e-mail адрес или в Telegram.

1.5 Аналоги

Перед разработкой требуемой системы мониторинга были рассмотрены наиболее востребованные её аналоги, представленные на рынке программного обеспечения:

- PageSpeed Insights (<https://developers.google.com/speed/pagespeed/insights/>) [1]
- Test My Site (<https://testmysite.withgoogle.com/intl/ru-ru>)
- Xenu (<https://xenu-link-sleuth.en.softonic.com/>) [2]
- Unicom (<https://validator.w3.org/unicorn/>) [3]

Среди рассмотренных аналогов не было найдено имеющего все необходимые возможности. У некоторых сервисов отсутствует возможность проверки страниц с поддержкой SSL. Существующие сервисы позволяют проверить только одну страницу, нельзя настроить график тестирования и получение уведомлений, либо имеют небольшой список проверяемых дефектов и ошибок.

В таблице 1 представлены результаты наличия в рассмотренных программных продуктах подсистем, решающих те или иные аспекты требуемого функционала.

Таблица 1 – Сравнение аналогов разрабатываемого приложения

	PageSpeed Insights	Test My Site	Xenu	Unicorn
Возможность протестировать несколько веб-страниц	-	-	+	-
Поиск ошибок в скриптах при загрузке страницы	-	-	-	-
Измерение скорости загрузки страниц	+	+	-	-
Формирование отчета о тестировании и отправка уведомления	-	-	-	-

Рассмотренные аналоги частично реализуют необходимый функционал или полностью лишены требуемых модулей системы. Почти во всех системах отсутствует возможность протестировать несколько страниц веб-сайта и сформировать отчет о прохождении тестирования, и отправить уведомление. Для реализации заявленного функционала требуется разработать собственное программное обеспечение.

Глава 2. Автоматизация тестирования веб-приложения

2.1 Обзор используемых технологий

Реализация заявленного функционала в рамках отдельного приложения требует использование различных технологий и программных средств, решающих отдельные подзадачи разрабатываемой системы мониторинга. Ниже перечислены наиболее важные из них.

- Node.js - платформа, основанная на JavaScript движке V8, и предлагающая асинхронное API для работы с сетью и диском.
- JavaScript - прототипно-ориентированный сценарный язык программирования.
- ECMAScript 6 - стандарт языка JavaScript.
- Puppeteer – библиотека, которая позволяет автоматизировать работу с браузером Google Chrome.
- Navigation Timing API – API для сбора данных о производительности веб-приложения.
- Highcharts – библиотека для создания графиков
- Nodemailer - модуль для отправки email сообщений.
- Node-telegram-bot-api – библиотека для взаимодействия с Telegram Bot API для node.js
- Html-pdf – модуль для конвертации из html в pdf формат.
- Googleapis – библиотека для использования Google API.
- yandex-metrika – библиотека для использования API Яндекс.Метрики

2.2 Алгоритм тестирования веб-страниц.

Открытие и тестирование страниц осуществляется при помощи библиотеки Puppeteer. Страница проверяется на наличие следующих ошибок:

- изображений большого объема;
- битых ссылок;
- редиректов;
- некорректных мета-данных;
- ошибки в скриптах;
- увеличение времени загрузки страницы.

Тестирование начинается с запуска приложения и указания основного адреса сайта и настроек: допустимый размер изображений, количество ссылок, которые необходимо проверить, адрес почты, на который необходимо отправить отчет о тестировании.

При открытии основного адреса сайта извлекаются все внутренние ссылки, которые находятся на странице. По каждой ссылке отправляется запрос и проверяется статус ответа. Ссылки, которые ведут на несуществующие или страницы с ошибками сохраняются для записи в отчет.

При загрузке страницы происходит подписка на событие, отлавливающее ошибки, которые произошли при открытии веб-страницы. Такие ошибки могут стать причиной недоступности какого-либо функционала для пользователя или отсутствия части контента веб-страницы.

Также происходит подсчет ресурсов, которые загружаются при открытии страницы, таких как изображения, скрипты, файлы стилей, шрифты и др.

Далее у страницы проверяется наличие основных мета-тегов - заголовка и описания. При отсутствии какого-либо мета-тега эта информация добавляется к отчету тестирования.

После этого все изображения, которые загружаются при открытии страницы проверяются на допустимый размер, который задается в настройках. Список изображений, превышающие заданный размер, со ссылками на них добавляется при формировании отчета (Приложение В).

Если проверено ссылок меньше, чем задано в настройках, тогда берутся уникальные ссылки с этой страницы и описанный выше алгоритм повторяется пока условие на количество ссылок не будет выполнено.

Данные о прохождении тестирования сохраняются в файлы в формате JSON. Из результатов тестирования формируется отчет и отправляется на указанную почту или в Telegram (Рисунок 1).

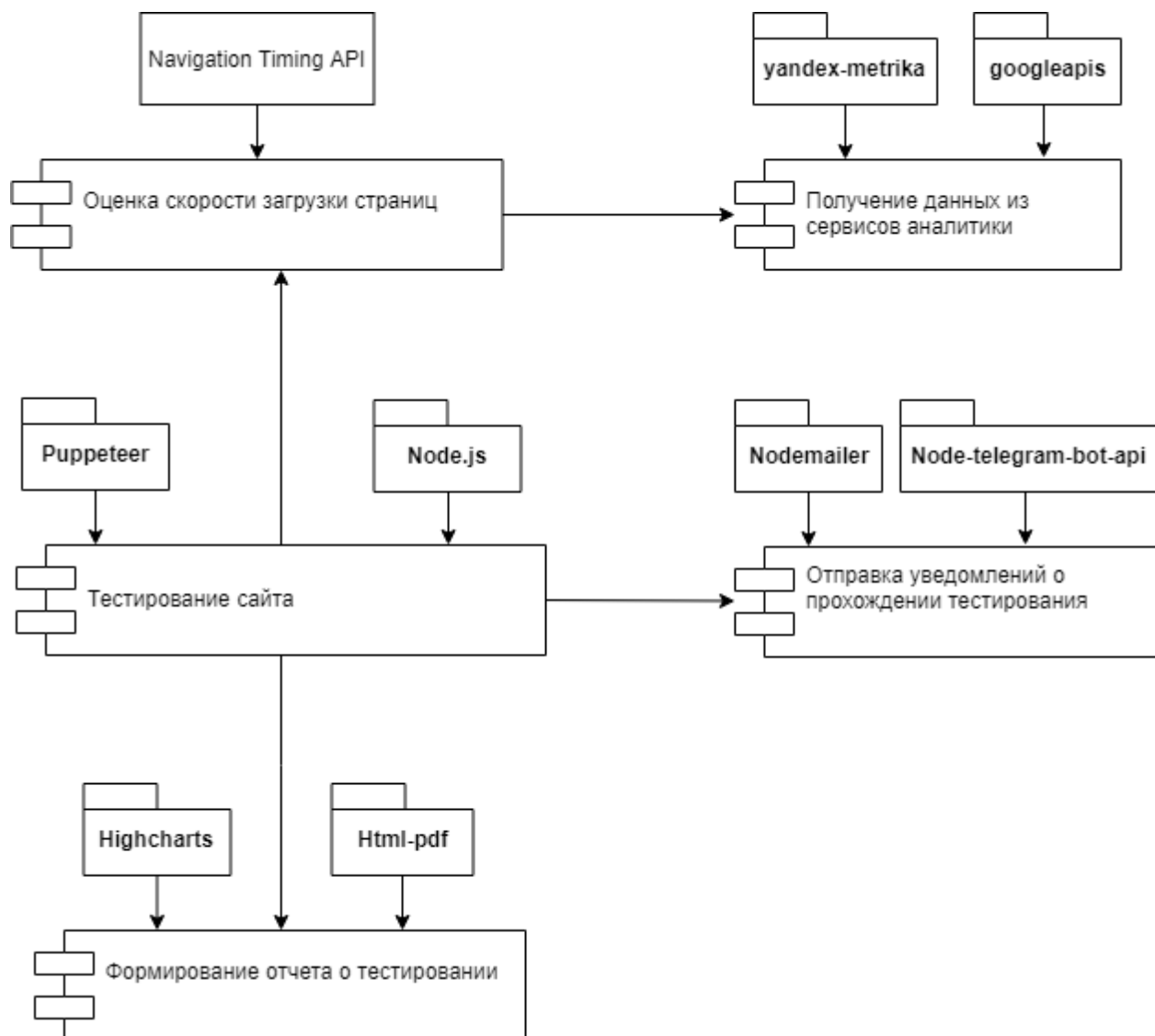


Рисунок 1 – Структура разрабатываемой системы

2.2.1 Оценка скорости загрузки страниц веб-приложения.

Скорость загрузки страницы является важным показателем не только для обычных пользователей, но и некоторых поисковиков, которые могут брать в расчет ранжирования сайтов этот показатель. При длительной загрузке веб-страницы пользователи могут закрыть ее и перейти на сайт, который загружается быстрее.

Данные о производительности веб-страницы можно получить при использовании Navigation Timing API, который может предоставить полную и точную информацию о задержке в получении данных.

Navigation Timing API может быть использован для сбора данных о производительности на стороне клиента при отправке асинхронных запросов на сервер, а также для получения данных, которые нельзя получить с помощью других средств, таких как время выгрузки предыдущей страницы, время DNS запроса, время полной загрузки страницы. [8]

Для вычисления времени полной загрузки веб-страницы необходимо вычесть `navigationStart` из атрибута `loadEventEnd`.

Для получения среднего значения загрузки страницы проводится серия испытаний, число задается в настройках приложения. Но среднее значение чувствительно к «выбросам» - единичные случаи, при которых скорость загрузки имеет большое отклонение от среднего значения.

Чтобы не учитывать такие инциденты, считаются 50-й и 75-й процентиля. 50-й процентиль (медиана) – показывает, что 50% запросов укладывается в это время.

Для оценки скорости загрузки сайта используется не скорость загрузки главной страницы, а вычисляется среднее значение загрузки всех протестированных веб-страниц, что дает более полное представление о производительности сайта.

Кроме использования группы веб-страниц, вместо одной главной, для повышения объективности оценки скорости загрузки сайта можно использовать данные о посещении веб-страниц пользователями. Чем чаще посещается страница, тем важнее чтобы она имела высокую скорость загрузки, так как это повлияет на значительную часть аудитории сайта. Эти данные могут предоставить сервисы трекинга и аналитики, такие как Google Analytics и Яндекс.Метрика, для этого необходимо интегрировать их в разрабатываемое приложение.

2.2.2 Интеграция с сервисами Google Analytics и Яндекс.Метрика

Многие сайты используют инструменты для трекинга и аналитики, такие как Google Analytics и Яндекс.Метрика. Данные сервисы позволяют следить за трафиком сайта, его аудиторией, статистикой переходов по веб-страницам и конверсией.

Данные из этих сервисов можно использовать, чтобы скорректировать оценку скорости загрузки сайта (Рисунок 2). Посещаемость страниц сайта может сильно отличаться, поэтому для оценки скорости загрузки сайта лучше использовать среднее взвешенное значение (Формула 1.1)

$$\bar{x} = \frac{\sum_{i=1}^n (w_i \cdot x_i)}{\sum_{i=1}^n w_i}, \quad (1.1)$$

где x – время загрузки страницы,

w – количество посещений страницы

При вычислении среднего взвешенного с использованием статистики посещений страниц, результат будет учитывать влияние скорости загрузки сайта на большее количество пользователей.

Для того чтобы использовать API сервисов аналитики необходимо зарегистрировать его на OAuth-сервере, чтобы получить токены. Яндекс.Метрика имеет API, совместимое с Google Analytics, что упрощает разработку. Основная часть запроса для Google Analytics “<https://www.googleapis.com/analytics/v3/data/ga>”, а для Яндекс.Метрика имеет вид “<https://api-metrika.yandex.ru/analytics/v3/data/ga>” (Приложение Д).

Для того чтобы получить статистику просмотров определенной страницы, нужно указать дополнительные параметры запроса:

- `dimensions=ga:pagePath` – группировка данных по адресу страницы;
- `metrics=ga:pageviews` – метрика, которая позволяет получить данные о статистике посещаемости веб-страниц;
- `filters=ga:pagePath=/realty/` - ограничивает данные, возвращаемые в результате запроса, указанной веб-страницей;
- `start-date=2018-06-04` – дата начала отчетного периода;
- `end-date=2018-06-13` – дата окончания отчетного периода.

Ответ от API возвращается в кодировке UTF-8 в виде JSON-файла.

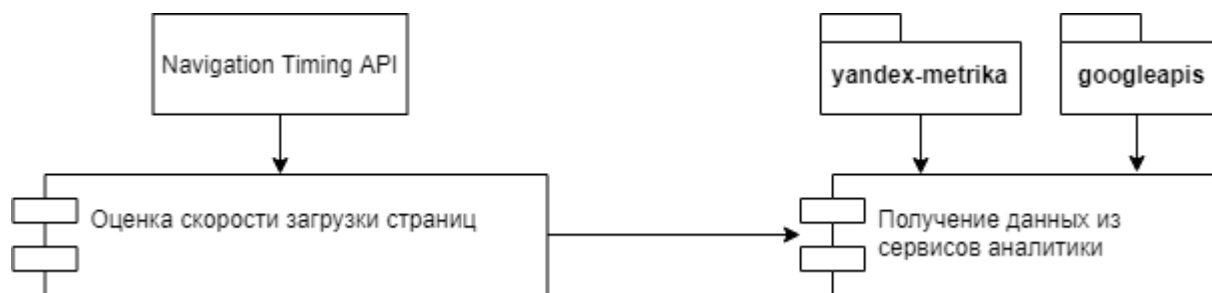


Рисунок 2 – Оценка скорости загрузки страниц

2.3 Формирование и отправка отчетов о прохождении тестирования

2.3.1 Формирование отчета

Отчет формируется в html формате из данных, которые были собраны в процессе тестирования веб-страницы (Рисунок 4). В отчете содержится следующая информация:

- количество изображений, превышающих допустимый объем и ссылки на них;
- количество битых ссылок;
- наличие редиректов;
- наличие некорректных мета-данных;
- ошибки в скриптах;
- размер дополнительных ресурсов (скрипты, файлы стилей, шрифты и др.)

Данная информация записывается для каждой протестированной веб-страницы. Для отображения дополнительных ресурсов строится круговая диаграмма (Рисунок 3).

Ресурсы, загружаемые для страницы www.etagi.com

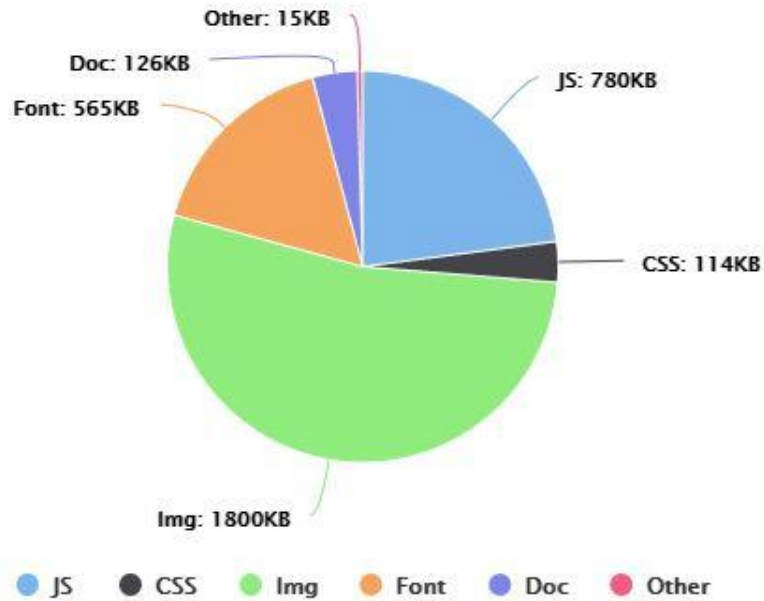


Рисунок 3 – Диаграмма загружаемых ресурсов страницы

Также в отчет заносится информация об общей скорости загрузки сайта по протестированным страницам с учетом статистики посещаемости этих страниц, если был подключен один из сервисов аналитики (Google Analytics или Яндекс.Метрика). Скорость загрузки характеризуется средним значением (средневзвешенным), а также процентилями (50%, 75%).

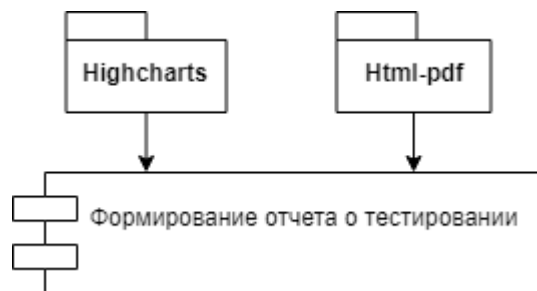


Рисунок 4 – Компонент формирования отчета

2.3.2 Отправка уведомлений на электронный адрес

Для отправки электронной почты используется модуль Nodemailer по протоколу SMTP. Сформированный отчет отправляется на почту, указанную в настройках (Приложение Г). В настройках отправки указываются данные авторизации, отправитель, заголовок письма, а также сам отчет в формате html.

2.3.3 Отправка уведомлений в Telegram

Для отправки уведомления в Telegram используется библиотека Node-telegram-bot-api. Для этого создается бот, заполняется имя, ник и получается токен бота, который будет использоваться при отправке уведомлений. Сформированный отчет конвертируется из html в pdf-формат при помощи библиотеки html-pdf и отправляется уведомление с прикрепленным pdf-файлом (Рисунок 5).

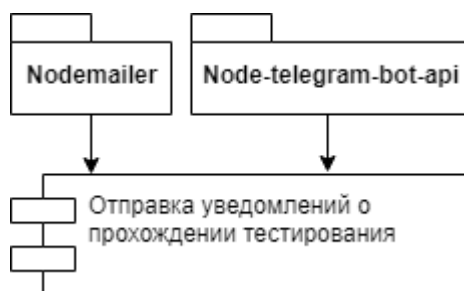


Рисунок 5 – Компонент отправки уведомлений

Глава 3. Применение разработанной системы мониторинга

Для запуска приложения пользователь заносит необходимые настройки приложения в JSON-формате (Приложение E):

- адрес сайта, который необходимо протестировать;
- список страниц для тестирования или их количество;
- список параметров, по которым нужно проводить мониторинг;
- допустимый размер изображений;
- количество серий, для оценки скорости загрузки веб-страницы;
- данные авторизации в системах трекинга и аналитики;
- данные для отправки уведомлений.

После заполнения необходимы настроек, выполняется скрипт, который проводит тестирование веб-сайта. Время выполнения зависит выбранного количества страниц для тестирования и количества серий, необходимых для оценки скорости загрузки веб-страницы. После прохождения тестирования информация сохраняется в JSON-файлы, а также отправляется уведомлении (Рисунок 6, Рисунок 7).

Представлен отчет о тестировании сайта “www.etagi.com”. Были протестированы 10 наиболее посещаемых страниц сайта за период с 4 по 13 июня. Для оценки скорости загрузки страниц было проведено по 50 серий открытия веб-страниц сайта.

На главной странице сайта было обнаружено изображение, превышающее допустимый размер (700KB). На странице “/analytics/” была обнаружена битая ссылка на изображение (статус ответа - 404), а также ошибка выполнения скрипта при загрузке страницы (Рисунок 8).

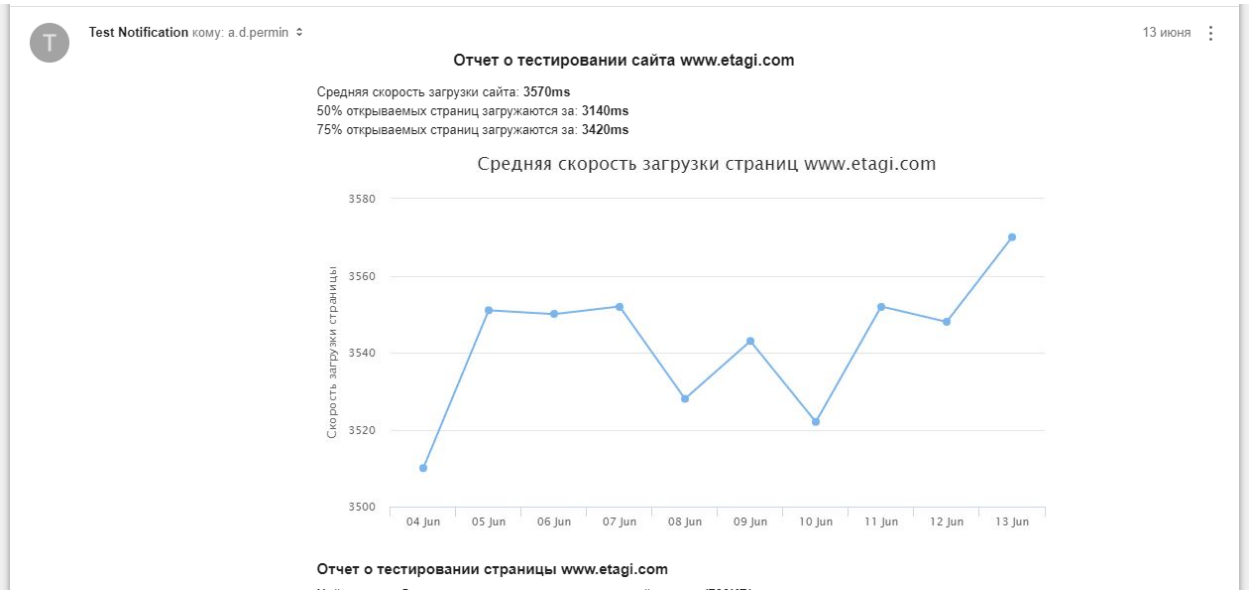


Рисунок 6 – Отчет о тестировании (информация о сайте)

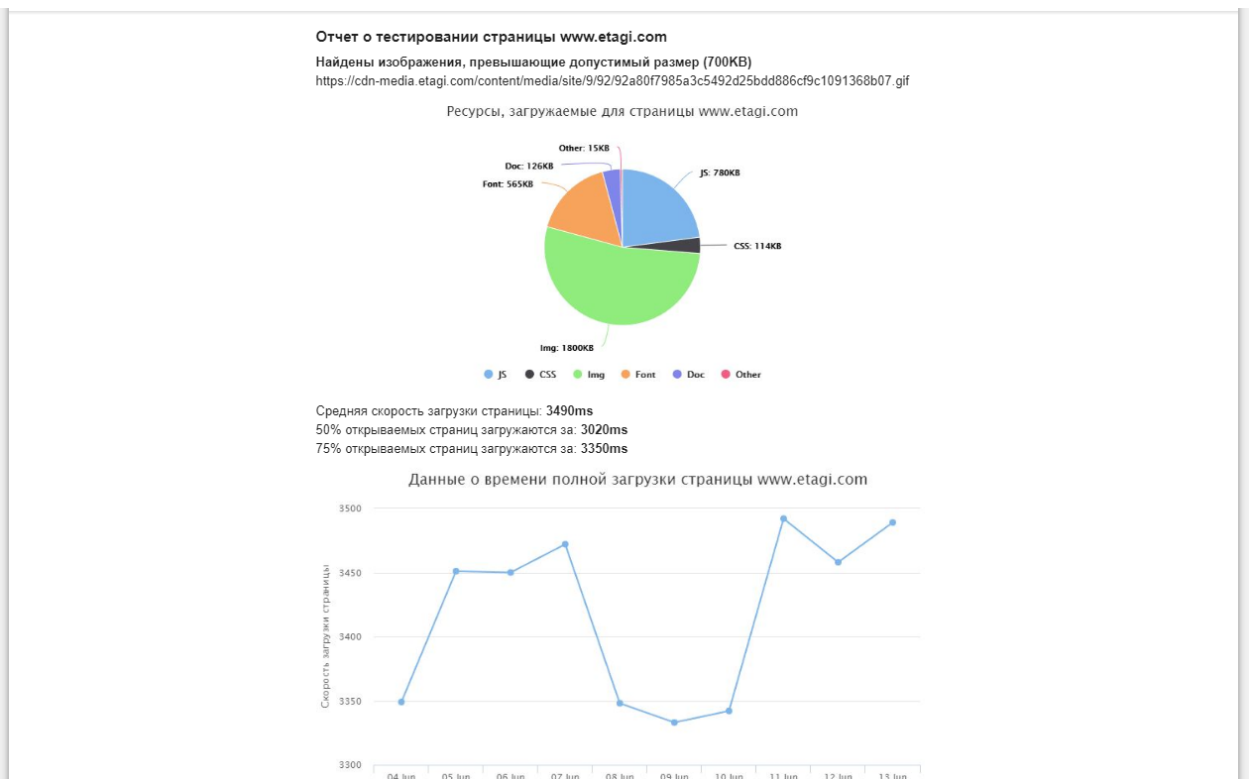


Рисунок 7 – Отчет о тестировании (информация о главной странице)

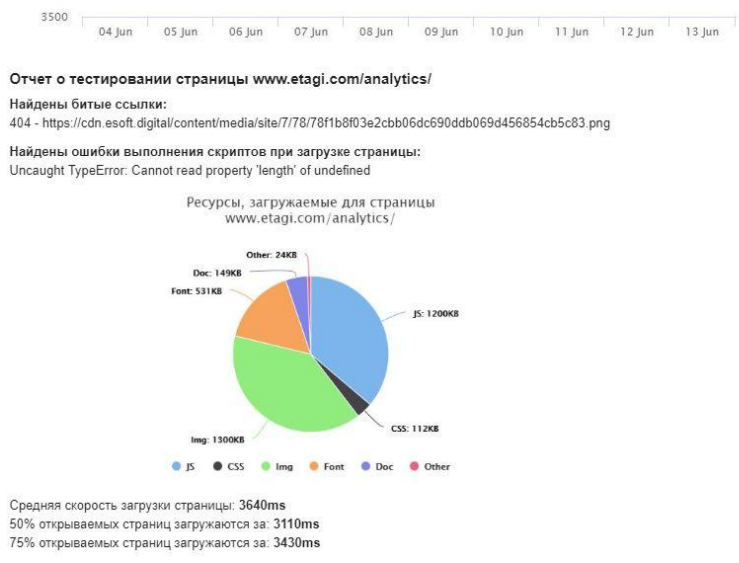


Рисунок 8 – Отчет о тестировании (информация о странице “/analytics/”)

Также отчет может содержать другие данные о результатах тестирования (Рисунок 9).



Рисунок 9 – Отчет о тестировании (информация о странице “/test/”)

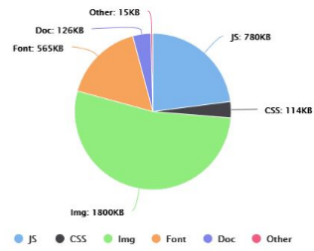
Если результат оценки скорости страницы будет иметь большое отклонение, на диаграмме этот участок будет выделен (Рисунок 10).

Отчет о тестировании страницы www.etagi.com

Найдены изображения, превышающие допустимый размер (700KB)

<https://cdn-media.etagi.com/content/media/site/9/92/92a80f7985a3c5492d25bdd886cf9c1091368b07.gif>

Ресурсы, загружаемые для страницы www.etagi.com



Средняя скорость загрузки страницы: 3569ms

50% открываемых страниц загружаются за: 3020ms

75% открываемых страниц загружаются за: 3350ms

Данные о времени полной загрузки страницы www.etagi.com

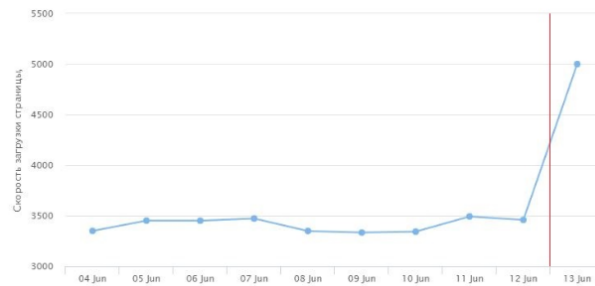


Рисунок 10 – Отчет о тестировании (информация о странице)

Заключение

В ходе выполнения выпускной квалификационной работы выявлены наиболее распространенные проблемы, которые невозможно или трудно отследить на этапе тестирования: наличие изображений большого объема, битых ссылок, редиректов, некорректных мета-данных, ошибок при выполнении скриптов, увеличение времени загрузки страницы.

Изучены существующие продукты для тестирования веб-приложений: PageSpeed Insights, Test My Site, Xenu, Unicorn.

Разработан алгоритм тестирования веб-страниц и реализована процедура оценки скорости веб-страниц с использованием систем трекинга и аналитики Google Analytics и Яндекс.Метрика.

Реализована система мониторинга веб-страниц сайта с отправкой уведомлений на электронную почту или в Telegram и протестирована на реальных данных сайта www.etagi.com.

Список литературы

1. About PageSpeed Insights [Электронный ресурс]. URL:<https://developers.google.com/speed/docs/insights/about> (дата обращения: 20.12.2017 г.)
2. Find broken links on web sites [Электронный ресурс]. URL:<http://home.snafu.de/tilman/xenulink.html> (дата обращения: 20.12.2017 г.)
3. Unicorn [Электронный ресурс]. URL:<https://validator.w3.org/unicorn/> (дата обращения: 20.12.2017 г.)
4. Тестирование. Фундаментальная теория / Хабр [Электронный ресурс] URL:<https://habrahabr.ru/post/279535/> (дата обращения: 20.03.2018 г.)
5. Node.js [Электронный ресурс]. URL:<https://nodejs.org/en/> (дата обращения: 10.01.2018 г.)
6. Headless browsers: что, как и почему. Виталий Слободин. Конференция HolyJS 2017 Moscow
7. Nodemailer. [Электронный ресурс]. URL:<https://nodemailer.com/about/> (дата обращения: 10.01.2018 г.)
8. Navigation Timing API - Интерфейсы веб API | MDN. [Электронный ресурс]. URL:
https://developer.mozilla.org/ru/docs/Web/API/Navigation_timing_API
(дата обращения: 13.02.2018 г.)
9. JavaScript, Node, Puppeteer: автоматизация Chrome и веб-скрапинг. [Электронный ресурс]. URL:<https://habrahabr.ru/company/ruvds/blog/341348/> (дата обращения: 27.12.2018 г.)

10. Puppeteer | Tools for Web Developers [Электронный ресурс]
[URL:https://developers.google.com/web/tools/puppeteer/](https://developers.google.com/web/tools/puppeteer/) (дата обращения: 25.12.2018 г.)
11. Telegram Bot API. [Электронный ресурс].
[URL:https://core.telegram.org/bots/api](https://core.telegram.org/bots/api) (дата обращения: 15.03.2018 г.)
12. Highcharts JS API Reference [Электронный ресурс].
URL:<https://api.highcharts.com/highcharts/> (дата обращения: 15.02.2018 г.)
13. Googleapis. [Электронный ресурс]
[URL:http://google.github.io/google-api-nodejs-client](http://google.github.io/google-api-nodejs-client) (дата обращения: 20.03.2018 г.)
14. API Метрики — Запрос к API — Технологии Яндекса.
[Электронный ресурс]. URL:
<https://tech.yandex.ru/metrika/doc/api2/ga/queries/requestjson-docpage/>
(дата обращения: 20.03.2018 г.)

Приложение А - Пример теста компонента “Breadcrumbs”

```
import React from 'react';

import {expect} from 'chai';

import {mount} from 'enzyme';

import Breadcrumbs from './Breadcrumbs';

import configureStore from '../core/configureStore';

describe('<Breadcrumbs />', () => {

  let component;

  beforeEach(() => {

    const store = configureStore({});

    component = mount(<Breadcrumbs {...props}/>);

  });

  describe('Breadcrumbs main items test suite', () => {

    it('should render custom last breadcrumb when userTitle is set', () => {

      props.userTitle = 'USER';

      component = mount(<Breadcrumbs {...props}/>);

      expect(component.find('li').last().text()).to.equal('USER');

    });

  });

});
```

Приложение Б - Пример теста по открытию фильтра

```
var sh = Object.create(require('../Helpers/specHelpers.js'));

var page = Object.create(require('../Pages/Page.js'));

var filter = Object.create(require('../Widgets/FilterSwitcher_Widget.js'));

var allPages = [

  page.realty,

  page.zastr,

  page.realtyOut,

  page.realtyRent,

  page.commerce,

];

describe('Проверка событий в dataLayer при открытии расширенного фильтра', function() {

  sh.getCities().forEach(function(city) {

    allPages.forEach(function(p) {

      it(p.getPath(city), function() {

        p.get(city);

        p.error404.isPresent().then(function(res) {

          if (res) {

            console.log(('404 на ' + p.getPath(city)).red.bold);

          } else {

            filter.openFilter().then(function() {

              expect(filter.getEvent()).toBe(true, 'Событие ' + filter.event + ' должно быть в dataLayer');

            });

          }

        });

      });

    });

  });

});
```

```
    expect(filter.filterOpened.isPresent()).toBe(true);

  }).then(function() {

    return filter.closeFilter().then(function() {

      expect(filter.filterOpened.isPresent()).toBe(false);

    });

  });

  });

  });

  });

  });

  });
```

Приложение В - Определение картинок на странице

```
import {maxImageSize} from './settings';

const puppeteer = require('puppeteer');

(async () => {

  const browser = await puppeteer.launch();

  const page = await browser.newPage();

  page.setViewport({ width: 1280, height: 926 });

  let images = {};

  page.on('response', async (response) => {

    const matches = /\.*\.(jpg|png|svg|gif)$/.exec(response.url());

    if (matches && (matches.length === 2)) {

      const buffer = await response.buffer();

      if (buffer.length > maxImageSize) {

        images[response] = buffer;

      }

    }

  });

  await page.goto('https://www.etagi.com/');

  await browser.close();

})();
```

Приложение Г - Функция отправки отчета о тестировании

```
const nodemailer = require('nodemailer');

export const sendReport = (report, emailTo) => {

  const transporter = nodemailer.createTransport({

    service: 'gmail',

    auth: {

      user: '****@gmail.com',

      pass: '*****'

    }

  });

  const mailOptions = {

    from: '***',

    to: emailTo,

    subject: report.title,

    html: report.body

  };

  transporter.sendMail(mailOptions);

};
```

Приложение Д - Функция получения данных из Яндекс.Метрика

```
const YMetrikaRequest = require( 'yandex-metrika' );

export const getPageViews = (ouathToken, counterId, params) => {

  const api = new YMetrikaRequest(ouath_token);

  const url = 'https://api-metrika.yandex.ru/analytics/v3/data/ga';

  api.request(url, 'GET', params).then(data => {

    return data.totalsForAllResults['ga:pageviews'];

  });

};
```


Приложение Е – Формат файла конфигурации системы мониторинга

```
{  
  
  "site": "www.etagi.com",  
  
  "pagesCount": 10,  
  
  "pagesList": [],  
  
  "maxImageSize": 700,  
  
  "seriesCount": 50,  
  
  "email": "test@mail.ru",  
  
  "yandexMetrika": {  
  
    "oauthToken": "05dd3dd84ff948fdae2bc4fb91f13e22bb1f289ceef0037",  
  
    "counterId": "47055435"  
  
  },  
  
}
```