



**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"**  
**(УНИВЕРСИТЕТ ИТМО)**

**ГРАФИК КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

Студент Морозов А.Д.  
(Фамилия, И., О.)

Факультет ПИИКТ Группа Р41091


Направление (специальность) 09.04.04 Программная инженерия

Руководитель Государев И.Б., доцент  
(Фамилия, И., О., должность)

Дисциплина Проектирование и анализ языков веб-решений

Наименование темы: Анализ времени сборки проекта на JAMstack генераторами статических сайтов в зависимости от числа страниц

№ п/п	Наименование этапа	Дата завершения		Оценка и подпись руководителя
		Планируемая	Фактическая	
1.	Анализ различия статических и динамических сайтов. Анализ языка разметки markdown. Анализ существующих генераторов статических сайтов и их классификации	05.06.22	05.06.22	
2.	Сбор статистики времени сборки проекта каждым выбранным генератором в зависимости от числа генерируемых страниц. Написание отчета. Защита проекта	18.06.22	18.06.22	

Руководитель  И.Б. Государев  
(подпись) (Фамилия И.О.)

Студент  А.Д. Морозов  
(подпись) (Фамилия И.О.)

**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"**  
**(УНИВЕРСИТЕТ ИТМО)**

**АННОТАЦИЯ К КУРСОВОМУ ПРОЕКТУ (РАБОТЕ)**

Студент Морозов А.Д.  
(Фамилия, И., О.)

Факультет ПИИКТ Группа Р41091

Направление (специальность) 09.04.04 Программная инженерия

Руководитель Государев И.Б., доцент  
(Фамилия, И., О., должность)

Дисциплина Проектирование и анализ языков веб-решений

Наименование темы: Анализ времени сборки проекта на JAMstack генераторами статических сайтов в зависимости от числа страниц

**ХАРАКТЕРИСТИКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

**1. Цель и задачи работы** ☒ Предложены студентом ☐ Определены руководителем

Цель работы — провести анализ времени сборки проекта генераторами статических сайтов в зависимости от числа генерируемых статических страниц

Задачи работы:

1. проанализировать различия статических и динамических сайтов;
2. провести анализ языка разметки markdown;
3. провести анализ существующих генераторов статических сайтов и их классификации;
4. собрать статистику времени сборки проекта каждым выбранным генератором в зависимости от числа генерируемых страниц;
5. проанализировать полученные данные и сделать выводы.

**2. Характер работы**

☐ Расчет ☐ Конструирование ☐ Моделирование ☒ Другое

**3. Содержание работы**

Используя программное обеспечение генераторов статических сайтов, оценено время сборки проекта каждым из них в зависимости от числа генерируемых страниц.

Составлена пояснительная записка по проекту.

**4. Выводы**

Требования к проекту реализованы

Руководитель

(подпись)

И.Б. Государев

(Фамилия И.О.)

Студент

(подпись)

А.Д. Морозов

(Фамилия И.О.)

**Факультет программной инженерии и компьютерной техники**

**Направление (специальность) — 09.04.04 Программная инженерия**

**Образовательная программа — Веб-технологии**

**Дисциплина — Проектирование и анализ языков веб-решений**

## **Курсовой проект**

**ТЕМА: «Анализ времени сборки проекта на JAMstack генераторами  
статических сайтов в зависимости от числа страниц»**

**ВЫПОЛНИЛ**

Студент группы

P41091

№ группы



подпись, дата

А.Д. Морозов

ФИО

**ПРОВЕРИЛ**

ученая степень, должность

подпись, дата

ФИО

**САНКТ-ПЕТЕРБУРГ**

**2022 г.**

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>3</b>
<b>1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ.....</b>	<b>5</b>
1.1 Статические и динамические сайты.....	5
1.2 Язык разметки Markdown.....	8
1.3 Генераторы статических сайтов.....	10
1.4 Классификация генераторов статических сайтов .....	11
1.5 Описание эксперимента.....	13
<b>2 ПРАКТИЧЕСКАЯ ЧАСТЬ .....</b>	<b>16</b>
2.1 Реализация сборки проекта генераторами статических сайтов...	16
2.1.1 Gatsby .....	16
2.1.1 Hugo.....	20
2.1.2 Jekyll.....	23
2.1.3 Next.js .....	26
2.1.4 Nuxt.....	29
2.1.5 Нехо .....	32
2.2 Анализ результатов.....	34
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>38</b>
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....</b>	<b>39</b>

## ВВЕДЕНИЕ

На сегодняшний день технологии веб-разработки являются одними из самых быстроразвивающихся. Каждый год появляются новые технологии и инструменты разработки, позволяющие создавать более качественные, безопасные, оптимизированные и как следствие – более быстродействующие веб-ресурсы.

Одним из ключевых факторов роста веб-технологий является развитие open-source проектов, создается огромное количество различных библиотек и фреймворков, хотя и некоторые из них имеют схожесть в функционале. Для успешного внедрения новых технологий, нужно быть в курсе основных тенденций и направлений развития веб-технологий, что не всегда удается при их скорости появления и развития.

Генераторы статических сайтов не являются исключением. Благодаря развитию веб-технологий генераторы статических сайтов набирают всю большую популярность. Это обуславливается тем, что для статических сайтов отсутствует серверная логика, так как генерация файлов проекта (HTML разметки, CSS стилей и JavaScript файлов) происходит заранее, поэтому скорость доступа к ресурсу заметно возрастает.

При каждом малейшем изменении кода проекта генераторы статических сайтов обязаны делать повторную сборку проекта, чтобы пользователи увидели изменения. Зачастую генераторы статических сайтов делают это автоматическом режиме. Однако каждый генератор обладает своими особенностями и набором программного обеспечения, поэтому время сборки в разных генераторах статических сайтов может заметно отличаться.

Цель работы – провести анализ времени сборки проекта генераторами статических сайтов в зависимости от числа генерируемых статических страниц.

Задачи работы:

- 1) проанализировать различия статических и динамических сайтов;
- 2) провести анализ языка разметки markdown;
- 3) провести анализ существующих генераторов статических сайтов и их классификации;
- 4) собрать статистику времени сборки проекта каждым выбранным генератором в зависимости от числа генерируемых страниц;
- 5) проанализировать полученные данные и сделать выводы.

# 1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 1.1 Статические и динамические сайты

Веб-сайт – это набор веб-страниц, содержащие различный текстовый и мультимедийный контент (изображения и видео) и позволяющие получить доступ к этому контенту по запросу пользователя через URL-адрес.

Веб-сайты разделяются на два типа:

- статические;
- динамические.

С появлением интернета одними из самых первых сайтов были статические сайты. Статические сайты позволяли передавать фиксированную информацию (HTML, CSS, JavaScript файлы) с веб-сервера в браузер пользователя, при этом абсолютно все пользователи сайта будут видеть одинаковые страницы. Поэтому статический сайт подходил только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.

С дальнейшим развитием интернета стало популярно использование динамических страниц. Динамические страницы позволяли разработчикам увеличивать функционал сайта, строить страницу из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из базы данных. Но минусом использования динамических страниц была их долгая загрузка, исходя из этого поисковые системы отдавали предпочтение в выдаче статическим сайтам, поскольку они загружаются очень быстро. Также, статические сайты менее затратны, в следствии отсутствия на стороне хостинга поддержке обработки серверной части веб-сайта.

Наглядную разницу между использованием статической и динамической концепций при разработке веб-сайтов можно увидеть в таблице 1:



**Таблица 1.**

**Основные особенности и различия при использовании статических  
и динамических сайтов**

<b>Статический веб-сайт</b>	<b>Динамический веб-сайт</b>
Содержание веб-страниц не может быть изменено во время использования	Содержание веб-страниц может быть изменено во время использования
Быстрая загрузка и хорошие SEO показатели «из коробки»	Требуется оптимизация загрузки страниц
Для управления контентом можно использовать CMS	Для управления контентом можно использовать CMS
Малые затраты на разработку	Более большие затраты, в сравнении с статическими сайтами
Не используются серверные языки программирования для разработки	Для написания серверной логики используются языки программирования (Node.js, PHP, Python, Java и другие)
Один и тот же контент доставляется каждый раз для всех пользователей сайта	Контент может меняться каждый раз при загрузке страницы

Динамические сайты в свою очередь позволяют разрабатывать современные веб-приложения с богатым функционалом, но из-за того, что выполняется большое количество запросов пользователей к серверному программному обеспечению, растет скорость загрузки страницы, а также существует риск в потере надежности ресурса при пиковых нагрузках на сервер.

На сегодняшний день, функционал разработанный на стороне сервера возможно перенести на клиентскую часть, например, комментарии пользователей, поиск по сайту, авторизацию и многое другое. Таким образом

нужный функционал будет подключен через сервисы по HTTP-протоколу через API, а сайт значительно улучшит время загрузки.

Использование статических сайтов дает большое количество преимуществ:

- 1) Бессерверное окружение – нет сервера в обычном понимании, весь проект организуется через CDN (content delivery network);
- 2) Высокая безопасность, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных. Серверная логика может быть абстрагирована в отдельные микросервисы, которые отвечают за свою безопасность самостоятельно. Даже, если подобный микросервис будет взломан, то остальная часть веб-приложения будет работать дальше без ощутимых проблем;
- 3) Высокая скорость загрузки сайта и его доступность для пользователей. Сборка веб-сайта происходит только один раз, при внесении изменений в него. За счет этого веб-приложение получает большой показатель TTFB (time to first byte). Данный показатель отображает время, за которое пользователь получит страницу в браузере при запросе. Показатель TTFB зависит от следующих параметров: загруженность сервера, скорость генерации страницы движком сайта и задержек при передаче данных;
- 4) Устойчивость при нагрузках и простое масштабирование, путем переноса файлов проекта на дополнительные хостинги. Обслуживание стека файлов в большом количестве мест отлично подходит под концепцию CDN, уменьшая стоимость и трудозатраты на масштабирование приложения;
- 5) Модульность проекта – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.

## 1.2 Язык разметки Markdown

Одной из главных особенностей языков разметки является отделение семантики от форматирования. Используя язык разметки, автор описывает смысловую разметку текста, но не указывает, каким должно быть его оформление.

Одним из таких языков разметки является Markdown (расширение файлов md). Данный язык разметки очень прост в освоении, его элементы разметки лаконичны и не мешают восприятию основного содержимого. По сравнению с другими языками разметки синтаксис Markdown прост в использовании. Для записи Markdown можно использовать любой текстовый редактор, но удобнее использовать специализированные редакторы кода, например, VS Code с расширением «Markdownlint» для автоматического просмотра внешнего вида получившейся разметки.

Язык markdown поддерживает множество конструкций, из которых можно выделить следующие:

1. Написание заголовков разного уровня;
2. Полужирное написание текста;
3. Курсивное написание текста;
4. Моноширинное написание;
5. Описание нумерованных и маркированных списков;
6. Вставка таблиц;
7. Вставка изображений и ссылок;
8. Реализация цитирования блоков текста;
9. Вставка фрагментов кода и комментариев;
10. Реализация метаданных внутри файла.

Пример реализации таких конструкций представлен в таблице 2:

Таблица 2.

*Реализация элементов разметки в Markdown*

Конструкция	Назначение
#Заголовок1 ###Заголовок2 ###Заголовок3	Заголовки разных уровней (возможны от 1 до 6)
**полужирный текст**	Полужирное написание текста
*курсивный текст*	Курсивное написание текста
`monospace`	Моноширинное написание
1. текст 1 2. текст 2	Нумерованные списки
* текст 1 * текст 2	Маркированные списки
![alt text](ссылка или путь к изображению)	Изображение
[google](https://google.com)	Ссылка
> Блок с текстом для цитаты	Цитирование
```\nprint('Hello world');\n```	Фрагмент кода
<!-- комментарий -->	Комментарии
---\n\ntitle: курсовой проект\nauthor: Алексей Морозов\n\n---	Написание метаданных в начале файла (язык YAML)

Для написания блогов и документации (README-файлов) очень часто прибегают к использованию языка разметки Markdown. Также файлы

Markdown легко конвертируются в HTML-код, для чего используются генераторы статических сайтов.

### 1.3 Генераторы статических сайтов

Устройство генераторов статических сайтов (static site generators - SSG) представляет из себя генерацию HTML, JavaScript, CSS файлов из динамического контента. В большинстве случаев разработка с использованием SSG осуществляется в командной строке, но растет количество SSG на основе браузера.

Одна из главных возможностей использования SSG это наличие в них лёгкой и быстрой разработки темы проекта, позволяющей настраивать внешний интерфейс и месторасположение содержимого контента. Большинство из созданных SSG опирается на существующие инструменты для тематизации и не создают свои собственные, а также дают возможность расширять тематизацию с помощью расширений.

На сайте JAMstack можно оценить популярность генераторов статических сайтов по количеству оценок в их репозиториях. Исходя из данных, в пятерку крупнейших генераторов входят: Next.js, Hugo, Gatsby, Jekyll (один из первых SSG от создателя GitHub), Nuxt и Hexo.

Основными критериями при выборе SSG являются:

- 1) Доступность (наличие веб-платформы, а не только командной строки);
- 2) Поддержка нескольких языков шаблонов;
- 3) Наличие локального сервера для разработки и тестирования продукта;
- 4) Поддержка используемых разработчиком форматов данных (JSON, YAML, TOML);

- 5) Расширяемая архитектура (плагины, расширения и дополнительный функционал);
- 6) Хорошая производительность и автоматическая сборка.

Стоит подробнее остановиться на последнем пункте. Сборка проекта осуществляется каждый раз, при внесении в него даже самых малых изменений, но время, затрачиваемое на сборку достаточно велико. В связи с, чем больше будет расти проект по мере своего существования, тем больше будет время, затраченное на его сборку, при внесении изменений в шаблон. Также, для всех генераторов статических сайтов время сборки может отличаться.

## **1.4 Классификация генераторов статических сайтов**

Классификация инструментов генерации статических сайтов, исходя из сценариев использования их в разных ситуациях и наличия нужного функционала выглядит следующим образом:

- предпочтительные сценарии использования;
- язык и экосистема;
- языки разметки, шаблонов, метаданных и данных;
- подключаемые модули и плагины;
- дополнительные инструменты разработки.

Инструменты по первому пункту разделяются в зависимости от написанной документации разработчиком, в которой он описывает сценарии использования генератора, а также его характеристики. Также инструменты в этом пункте можно разделить по требованиям к конечному проекту пользователя.

По второму пункту инструменты генерации делятся по языку программирования, на котором они написаны. Так, по данным официального сайта JAMstack, можно найти инструменты генерации статических сайтов

практически на любом языке программирования, но одним из ведущих языков остается JavaScript, так как, из десяти самых популярных инструментов шесть написаны именно на нем.

В третьем пункте инструменты генерации делятся на два типа: инструменты, использующие широко распространённый набор языков (Markdown – язык разметки, JSX/Liquid – язык шаблонов, YAML – для разметки метаданных и JSON – для данных сайта) и инструменты, которые используют специализированные наборы языков (AsciiDoc/Textile – языки разметки, ERB, Go, Jinja и другие – языки шаблонов, API/GraphQL – управление метаданными и данными сайта).

Инструменты в четвертом пункте делятся на генераторы по двум группам. В первой инструменты, которые предоставляют богатый набор функционала, тем оформления и набор модулей, но менее настраиваемые, а во второй, инструменты, которые дают пользователю малый функционал изначально, но дают возможность настроить функциональность инструмента под нужды собственно проекта.

В последнем пункте выделяются инструменты по наличию у них из «коробки» дополнительных наборов инструментов, функций и систем интеграции с другими сервисами.

Таким образом инструменты генерации статических сайтов являются одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать страницы заранее и настроить их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретный разрабатываемый проект.

## 1.5 Описание эксперимента

Для анализа времени сборки проекта разными генераторами статических сайтов с официального сайта JAMstack было выбрано 6 самых популярных инструментов: Gastby, Hugo, Jekyll, Next.js, Nuxt и Нехо.

Тесты по времени сборки проекта проводились последовательно на одном и том же программном и аппаратном обеспечении:

- операционная система Windows 10;
- восьмиядерный процессор AMD Ryzen 7 5800x с тактовой частотой 3800 ГГц;
- оперативная память Crucial DDR4 с тактовой частотой 3200 МГц;
- твердотельный накопитель Samsung SSD с скоростью записи 3000 Мбайт/сек.

Все тесты для выбранных генераторов статических сайтов включали в себя следующие условия:

- в качестве источника данных для построения статических страниц использовались файлы Markdown, содержащие несколько текстовых блоков, заголовков, ссылок и метаданных, без использования изображений;
- перед проведением каждого теста кэш и сформированные страницы после предыдущего текста очищались, для того чтобы генераторы находились в одинаковых условиях;
- построение каждого проекта проводилось индивидуально, исходя из представленной документации по каждому генератору статических сайтов;
- результатом проведения теста являются сформированные статичные HTML-страницы из контента в файлах Markdown, к которым можно получить доступ при запуске локального сервера.



Для того, чтобы все файлы Markdown не содержали одну и ту же разметку, при их генерации каждый заголовок имел свой уникальный номер, начиная от 0 и до 9999. Таким образом оценка времени сборки производилась для следующего количества страниц: 1, 5, 10, 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000, 9000 и 10000.

## Структура файлов markdown (рисунок 1):

```
textfile0.md > ...
1 |--
2 title: "Learning about SSG №0"
3 date: "2022-06-16"
4 ---
5
6 # Статика
7
8 ## Статические сайты
9
10 С появлением интернета одними из самых первых сайтов были - статические сайты. Статические сайты позволяли передавать фиксированную информацию (HTML, CSS, JavaScript файлы) с веб-сервера в браузер пользователя, при этом абсолютно все пользователи сайта будут видеть одинаковые страницы. Поэтому статический сайт подходил только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.
11
12 С дальнейшим развитием стало популярно использование динамических страниц. Динамические страницы позволяли разработчикам увеличивать функционал сайта, строить страницу из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из базы данных. Но минусом использования динамических страниц была их долгая загрузка, исходя из этого поисковые системы отдавали предпочтение в выдаче статическим сайтам, поскольку они загружаются очень быстро.
13
14
15 Статические сайты на сегодняшний день имеют широкую область применения и не уступают динамическим в функциональном плане и обладают обширным количеством инструментов для построения полноценной бессерверной архитектуры.
16
17 Использование статических сайтов дает большое количество преимуществ:
18 1) Бессерверное окружение – нет сервера в обычном понимании, весь проект организуется через CDN;
19 2) Высокая безопасность, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных;
20 3) Высокая скорость загрузки приложения и его доступность для пользователей;
21 4) Устойчивость при нагрузках и простое масштабирование, путем переноса файлов проекта на дополнительные хостинги CDN;
22 5) Модульность проекта – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.
23
24 ## Генераторы статических сайтов (SSG)
25
26 Инструменты генерации статических сайтов являются одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать страницы заранее и настроить их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретный разрабатываемый проект.
27
28 Самыми популярными SSG являются:
29 1) Gatsby (https://www.gatsbyjs.com/);
30 2) Hugo (https://gohugo.io/);
31 3) Jekyll (https://jekyllrb.com/);
32 4) Next.js (https://nextjs.org/);
33 5) Nuxt.js (https://nuxtjs.org/);
34 6) Hexo (https://hexo.io/ru/);
35
```

Рисунок 1 – Пример структуры файла markdown с разметкой для генерации статических страниц.

## **2 ПРАКТИЧЕСКАЯ ЧАСТЬ**

### **2.1 Реализация сборки проекта генераторами статических сайтов**

#### **2.1.1 Gatsby**

На сегодняшний день Gatsby стал гораздо большим, чем просто генератором статических сайтов, сегодня, Gatsby – это довольно обширная платформа, позволяющая развертывать большие проекты. Gatsby построен на языке программирования JavaScript, поэтому для построения проекта за основу берётся библиотека React.

Большим преимуществом Gatsby является возможность получать данные из обширного количества источников, например: CMS, внешние API, файлы markdown, файлы csv. Таким образом не возникает потребности в использовании базы данных.

Также стоит упомянуть о большом количестве всевозможных плагинов, которые есть на платформе, что, несомненно, тоже является преимуществом по сравнению с другими генераторами. Использование таких плагинов позволяет быстрее организовать разработку проекта и добавить много функционала, но замедляет сборку приложения. Однако по окончании сборки проекта веб-сайт получает хорошую производительность и скорость доступа.

Проект, построенный для тестов на Gatsby, имел структуру, изображенную на рисунке 2. Папки «.cache» и «public» появлялись автоматически после проведения теста и содержали в себе информацию о кэше и сгенерированных статических страницах. В папке «src/posts» хранились файлы markdown (в примере на рисунке один файл - textfile0).

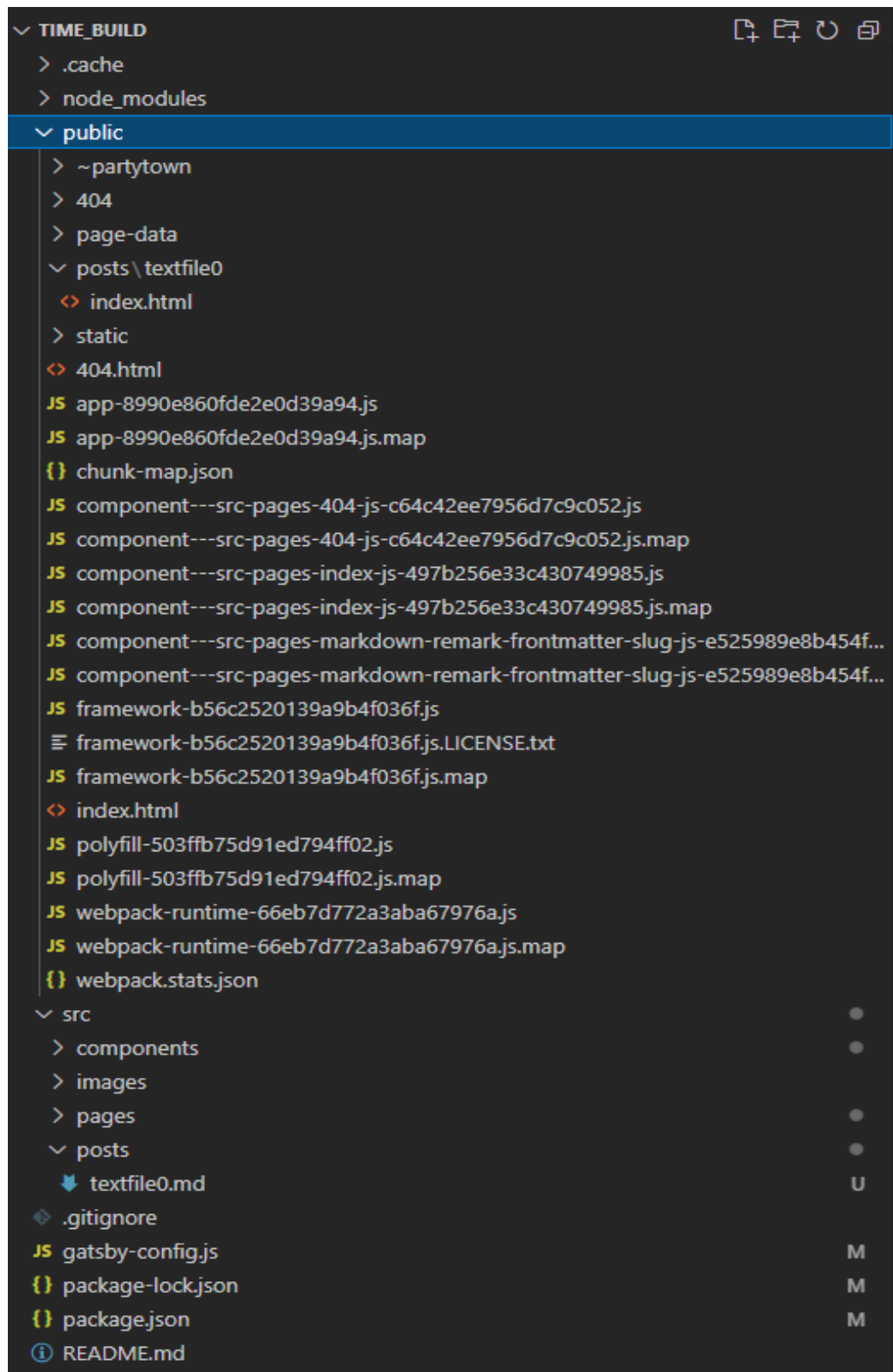


Рисунок 2 – Структура проекта для тестов на примере Gatsby.

Запуск теста для генерации textfile0 в статичную страницу (рисунок 3):

```
> time_build@1.0.0 build
> gatsby build

success compile gatsby files - 0.703s
success load gatsby config - 0.016s
success load plugins - 0.626s
success onPreInit - 0.003s
success initialize cache - 0.052s
success copy gatsby files - 0.117s
success Compiling Gatsby Functions - 0.185s
success onPreBootstrap - 0.199s
success createSchemaCustomization - 0.005s
success Checking for changed pages - 0.003s
success source and transform nodes - 0.078s
info Writing GraphQL type definitions to C:/Users/mad38/time_build/.cache/schema.gql
success building schema - 0.250s
success createPages - 0.001s
success createPagesStatefully - 0.070s
info Total nodes: 38, SitePage nodes: 4 (use --verbose for breakdown)
success Checking for changed pages - 0.002s
success onPreExtractQueries - 0.001s
success extract queries from components - 0.193s
success write out redirect data - 0.001s
success onPostBootstrap - 0.001s
info bootstrap finished - 4.543s
success write out requires - 0.005s
success Building production JavaScript and CSS bundles - 4.181s
success Building HTML renderer - 4.386s
success Execute page configs - 0.016s
success Caching Webpack compilations - 0.490s
success run queries in workers - 0.047s - 4/4 84.48/s
success Merge worker state - 0.002s
success Rewriting compilation hashes - 0.002s
success Writing page-data.json files to public directory - 0.011s - 4/4 355.22/s
success Building static HTML for pages - 2.506s - 4/4 1.60/s
success onPostBuild - 0.000s
info Done building in 17.3045156 sec

Pages
├─ src/pages/404.js
│  └─ /404/
│     └─ /404.html
├─ src/pages/index.js
│  └─ /
└─ src/pages/{MarkdownRemark.frontmatter__slug}.js
   └─ /posts/textfile0/

(SSG) Generated at build time
D (DSG) Deferred static generation - page generated at runtime
∞ (SSR) Server-side renders at runtime (uses getServerData)
λ (Function) Gatsby function
```

Рисунок 3 – Пример запуска теста для генерации файла textfile0.md в статичную страницу с использованием Gatsby.

Результат отображения в браузере сгенерированной страницы (textfile8000.md, рисунок 4):

## Learning about SSG №8000

June 16, 2022

### Статика

#### Статические сайты

С появлением интернета одним из самых первых сайтов были - **статические сайты**. Статические сайты позволяли передавать фиксированную информацию (*HTML, CSS, JavaScript файлы*) с веб-сервера в браузер пользователя, при этом абсолютно все пользователи сайта будут видеть одинаковые страницы. Поэтому статический сайт подходил только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.

С дальнейшим развитием стало популярно использование **динамических страниц**. Динамические страницы позволяли разработчикам увеличивать функционал сайта, строить страницу из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из базы данных. Но минусом использования динамических страниц была их долгая загрузка, исходя из этого поисковые системы отдавали предпочтение в выдаче статическим сайтам, поскольку они загружаются очень быстро.

**Статические сайты** на сегодняшний день имеют широкую область применения и не уступают динамическим в функциональном плане и обладают обширным количеством инструментов для построения полноценной бессерверной архитектуры.

Использование статических сайтов дает большое количество преимуществ:

1. **Бессерверное окружение** – нет сервера в обычном понимании, весь проект организуется через CDN;
2. **Высокая безопасность**, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных;
3. **Высокая скорость загрузки** приложения и его доступность для пользователей;
4. **Устойчивость** при нагрузках и простое масштабирование, путем переноса файлов проекта на дополнительные хостинги CDN;
5. **Модульность проекта** – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.

#### Генераторы статических сайтов (SSG)

**Инструменты генерации статических сайтов** являются одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать страницы заранее и настроить их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретный разрабатываемый проект.

Самыми популярными SSG являются:

1. [\*Gatsby\*](#);
2. [\*Hugo\*](#);
3. [\*Jekyll\*](#);
4. [\*Next.js\*](#);
5. [\*Nuxt\*](#);
6. [\*Hexo\*](#);

Рисунок 4 – Сгенерированная HTML-страница на основе файла markdown в браузере на Gatsby.

Таблица 3.

#### Результаты проведения тестов для Gatsby

Количество страниц	Время, сек
1	17,31
5	17,07
10	16,87
50	17,35
100	17,43

200	17,85
300	18,21
400	18,49
500	18,81
600	19,26
700	19,25
800	19,81
900	20,19
1000	20,53
2000	23,88
3000	26,96
4000	30,99
5000	33,93
6000	37,28
7000	39,93
8000	44,57
9000	52,21
10000	63,14

### 2.1.1 Hugo

Hugo – один из наиболее популярных и быстрых генератор статических сайтов с открытым исходным кодом. Для быстрой реализации проектов Hugo имеет множество готовых шаблонов, в частности по работе с SEO и такими элементами веб-сайтов, как: комментарии, формы, работы с статистикой сайта и другим.

Скорость работы генератора обусловлена использованием языка программирования Go, что в свою очередь позволяет генерировать статические страницы за доли секунд. Также использование Hugo возможно на большинстве операционных систем: Windows, macOS, Linux, FreeBSD и других. Hugo предоставляет обширное количество тем, позволяющих настроить отображение контента под любой вкус.

Структура проекта на Hugo после проведенного теста, представлена на рисунке 5, а проведение теста на рисунке 6. Тут папка «content» используется для хранения markdown файлов, а папка «public» для хранения

сгенерированных страниц. В папке «themes» храниться информация о выбранной теме для проекта.

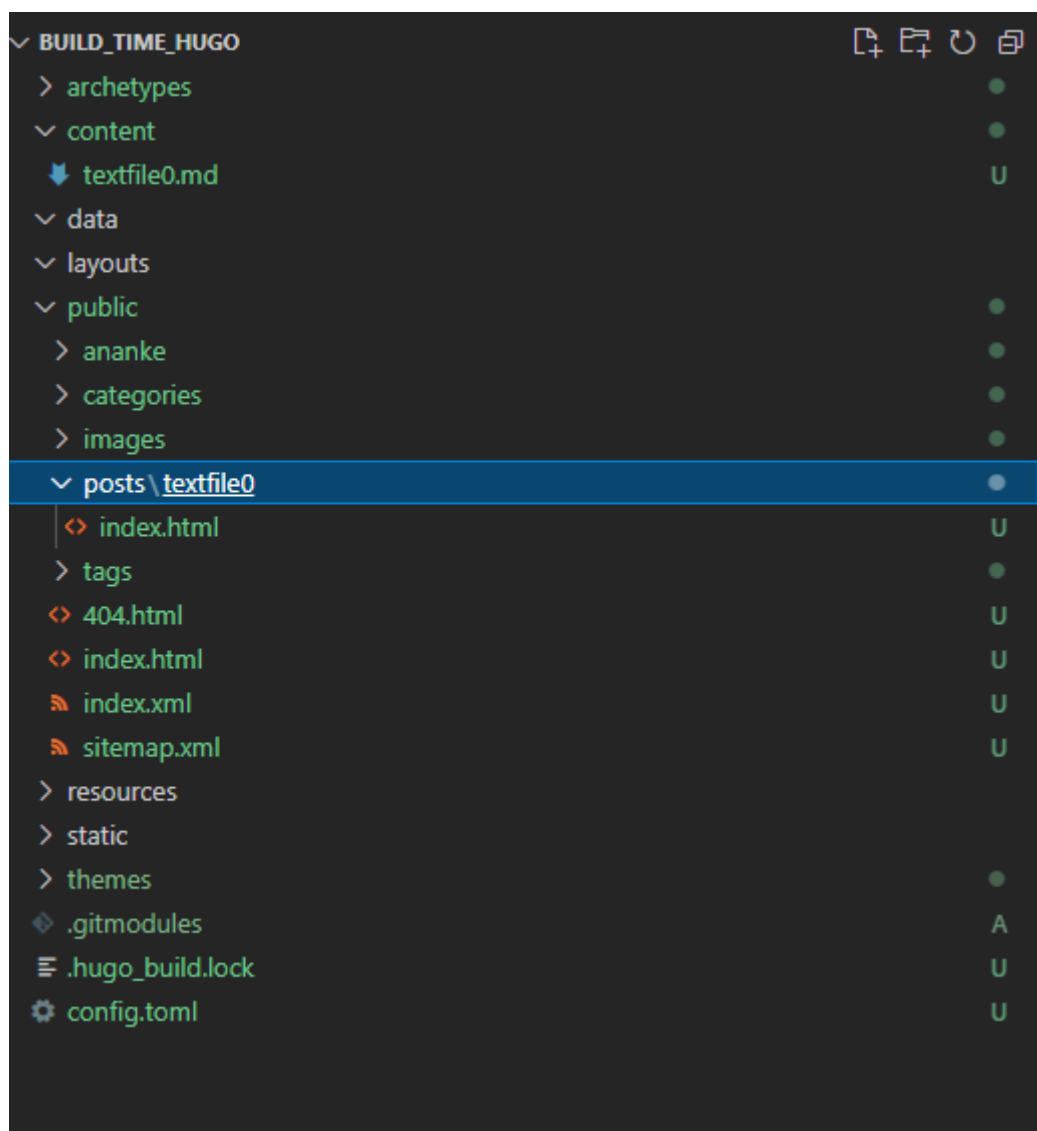


Рисунок 5 – Структура проекта на генераторе Нуго после проведения теста.

```
C:\Users\mad38\Hugo\build_time_hugo>hugo -D
Start building sites ...
hugo v0.101.0-466fa43c16709b4483689930a4f9ac8add5c9f66 windows/amd64 BuildDate=2022-06-16T07:09:16Z VendorInfo=gohugoio
```

	EN
Pages	8
Paginator pages	0
Non-page files	0
Static files	1
Processed images	0
Aliases	0
Sitemaps	1
Cleaned	0

```
Total in 108 ms
```

Рисунок 6 - Пример запуска теста для генерации файла `textfile0.md` в статичную страницу с использованием Нуго.



## Learning about SSG №0

### Статика

#### Статические сайты

С появлением интернета одними из самых первых сайтов были - **статические сайты**. Статические сайты позволяли передавать фиксированную информацию (*HTML, CSS, JavaScript файлы*) с веб-сервера в браузер пользователя, при этом абсолютно все пользователи сайта будут видеть одинаковые страницы. Поэтому статический сайт подходил только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.

С дальнейшим развитием стало популярно использование **динамических страниц**. Динамические страницы позволяли разработчикам увеличивать функционал сайта, строить страницу из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из базы данных. Но минусом использования динамических страниц была их долгая загрузка, исходя из этого поисковые системы отдавали предпочтение в выдаче статическим сайтам, поскольку они загружаются очень быстро.

**Статические сайты** на сегодняшний день имеют широкую область применения и не уступают динамическим в функциональном плане и обладают обширным количеством инструментов для построения полноценной бессерверной архитектуры.

Использование статических сайтов дает большое количество преимуществ:

1. **Бессерверное окружение** – нет сервера в обычном понимании, весь проект организуется через CDN;
2. **Высокая безопасность**, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных;
3. **Высокая скорость загрузки** приложения и его доступность для пользователей;
4. **Устойчивость** при нагрузках и простое масштабирование, путем переноса файлов проекта на дополнительные хостинги CDN;
5. **Модульность проекта** – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.

### Генераторы статических сайтов (SSG)

**Инструменты генерации статических сайтов** являются одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать страницы заранее и настроить их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретный разрабатываемый проект.

Самыми популярными SSG являются:

1. [Gatsby](#);
2. [Hugo](#);
3. [Jekyll](#);
4. [Next.js](#);
5. [Nuxt](#);
6. [Hexo](#);

*Рисунок 7 – Сгенерированная HTML-страница на основе файла markdown в браузере на Hugo.*

**Таблица 4.**

### Результаты проведения тестов для Hugo

Количество страниц	Время, сек
--------------------	------------

1	0,07
5	0,08
10	0,08
50	0,10
100	0,11
200	0,14
300	0,17
400	0,21
500	0,23
600	0,26
700	0,30
800	0,31
900	0,35
1000	0,36
2000	0,69
3000	0,98
4000	1,27
5000	1,59
6000	1,90
7000	2,19
8000	2,49
9000	4,23
10000	3,32

### 2.1.2 Jekyll

Jekyll – один из наиболее старых генераторов статических сайтов, от создателя GitHub, но все еще пользующийся большой популярностью. Jekyll позиционирует себя исключительно как обычный генератор статических сайтов, не использующий большое количество плагинов, которые бы увеличивали время сборки. Jekyll написан на языке Ruby, что в свою очередь дает ему хорошую оптимизацию и быструю сборку, однако документация по данному генератору достаточно скудная, а установка и настройка проекта немного затруднительная.

На рисунках 8 и 9 представлена структура проекта и пример генерации страниц. В папке «\_posts» хранятся файлы markdown, в папке «static»

сгенерированная статика после проведения теста и в папке «.jeekyll-cache» - кэш сгенерированных страниц.

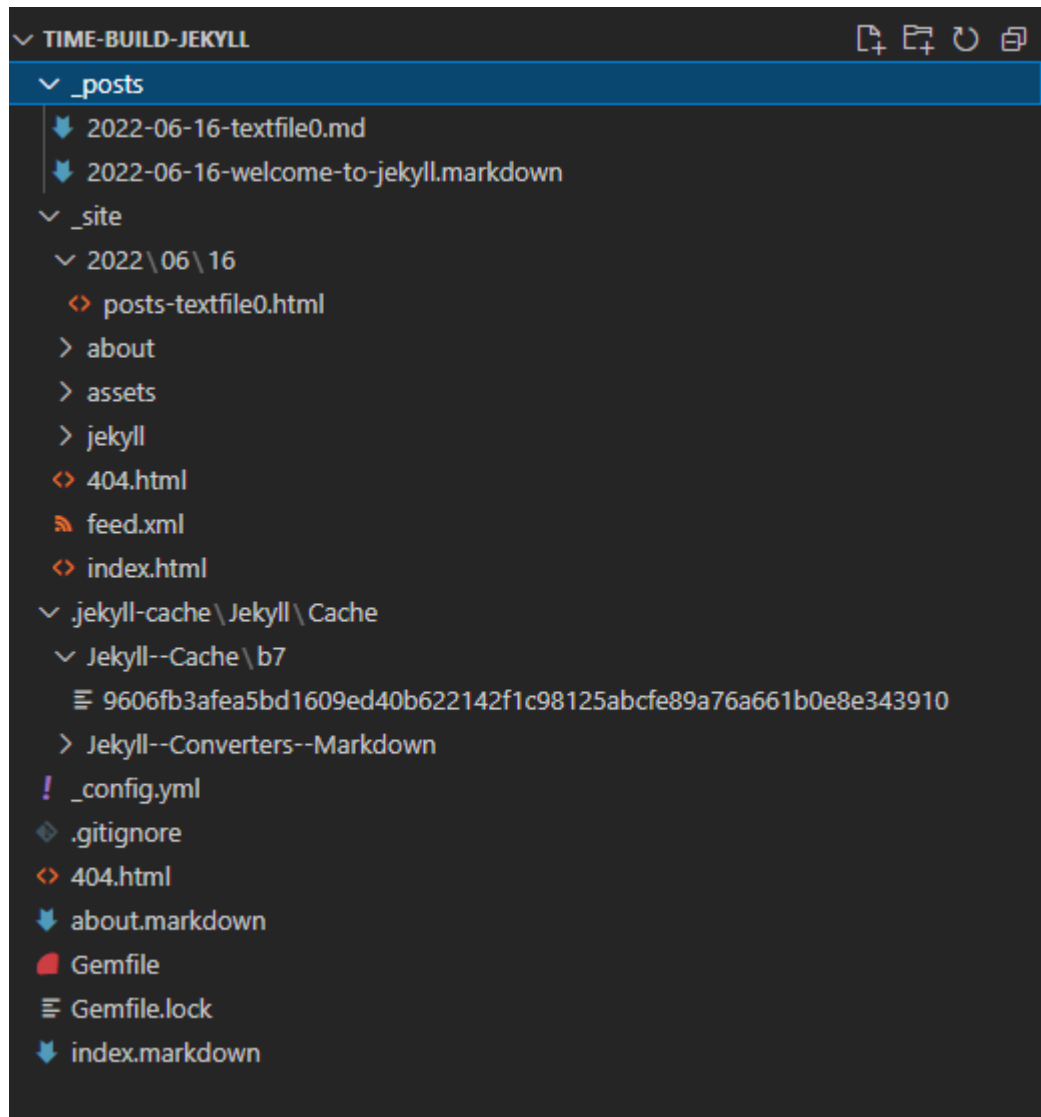


Рисунок 8 – Структура проекта на генераторе Jekyll после проведения теста.

```
C:\Users\mad38\time-build-jeekyll>jeekyll build
Configuration file: C:/Users/mad38/time-build-jeekyll/_config.yml
      Source: C:/Users/mad38/time-build-jeekyll
      Destination: C:/Users/mad38/time-build-jeekyll/_site
Incremental build: disabled. Enable with --incremental
Generating...
      Jekyll Feed: Generating feed for posts
                   done in 0.369 seconds.
Auto-regeneration: disabled. Use --watch to enable.

C:\Users\mad38\time-build-jeekyll>
```

Рисунок 9 - Пример запуска теста для генерации файла *textfile0.md* в статичную страницу с использованием Jekyll.

## Статика

### Статические сайты

С появлением интернета одним из самых первых сайтов были - **статические сайты**. Статические сайты позволяли передавать фиксированную информацию (*HTML, CSS, JavaScript файлы*) с веб-сервера в браузер пользователя, при этом абсолютно все пользователи сайта будут видеть одинаковые страницы. Поэтому статический сайт подходил только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.

С дальнейшим развитием стало популярно использование **динамических страниц**. Динамические страницы позволяли разработчикам увеличивать функционал сайта, строить страницу из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из баз данных. Но минусом использования динамических страниц была их долгая загрузка, исходя из этого поисковые системы отдавали предпочтение в выдаче статическим сайтам, поскольку они загружаются очень быстро.

**Статические сайты** на сегодняшний день имеют широкую область применения и не уступают динамическим в функциональном плане и обладают обширным количеством инструментов для построения полноценной бессерверной архитектуры.

Использование статических сайтов дает большое количество преимуществ: 1) **Бессерверное окружение** – нет сервера в обычном понимании, весь проект организуется через CDN; 2) **Высокая безопасность**, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных; 3) **Высокая скорость загрузки** приложения и его доступность для пользователей; 4) **Устойчивость** при нагрузках и простое масштабирование, путем переноса файлов проекта на дополнительные хостинги CDN; 5) **Модульность проекта** – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.

### Генераторы статических сайтов (SSG)

**Инструменты генерации статических сайтов** являются одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать страницы заранее и настраивать их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретный разрабатываемый проект.

Самыми популярными SSG являются: 1) *Gatsby*; 2) *Hugo*; 3) *Jekyll*; 4) *Next.js*; 5) *Next*; 6) *Nuxt*.

Рисунок 10 – Сгенерированная HTML-страница на основе файла markdown в браузере на Jekyll.

Таблица 5.

### Результаты проведения тестов для Jekyll

Количество страниц	Время, сек
1	0,37
5	0,38
10	0,40
50	0,45
100	0,51
200	0,63
300	0,74
400	0,86
500	0,98
600	1,07
700	1,19
800	1,34
900	1,44
1000	1,56
2000	2,76
3000	3,97
4000	5,14
5000	6,36
6000	7,60
7000	8,79
8000	9,94
9000	11,12
10000	12,34

### 2.1.3 Next.js

Next.js можно назвать одним из самых крупных и поддерживаемых генераторов статических сайтов, который уже перерос в фреймворк. Исходя из названия, Next.js написан на JavaScript и также как и Gatsby использует React для проектирования приложения. Отличительной чертой фреймворка Next.js является поддержка гибридного рендеринга страниц: серверного (server side rendering) и статического (static rendering), а также возможности использования одновременно обоих подходов.

Next.js имеет одну из самых подробных документация и примеров использования среди выбранных генераторов. Готовые шаблоны, интеграция по API с множеством других сервисов и множество других возможностей делают из Next.js сильный инструмент разработки, преимущественно использующийся для крупных проектов и являющийся прямым конкурентом Gatsby.

На рисунках 11 и 12 представлена структура проекта на Next.js и тест генерации статической страницы. Как видно из рисунка 11 проект имеет очень большую структуру и количество используемых компонентов. В папке «\_posts» хранятся страницы markdown, а в папке «.next/» код созданных статических страниц и кэша.

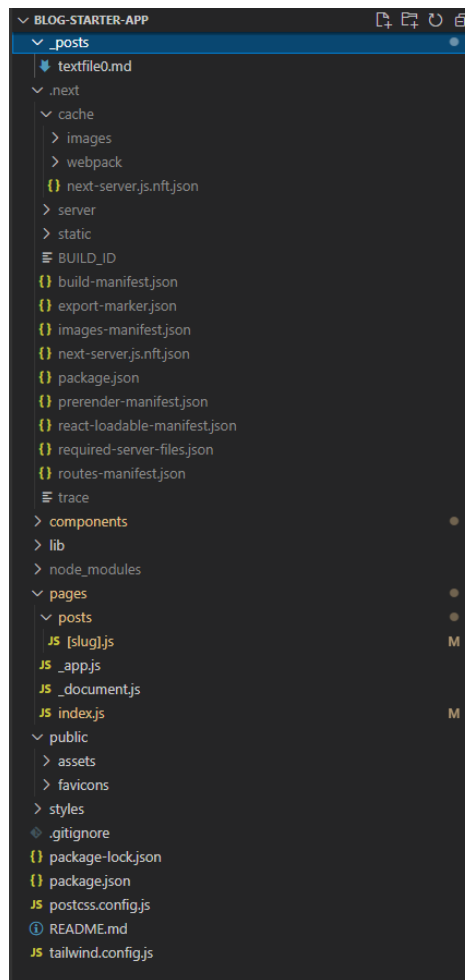
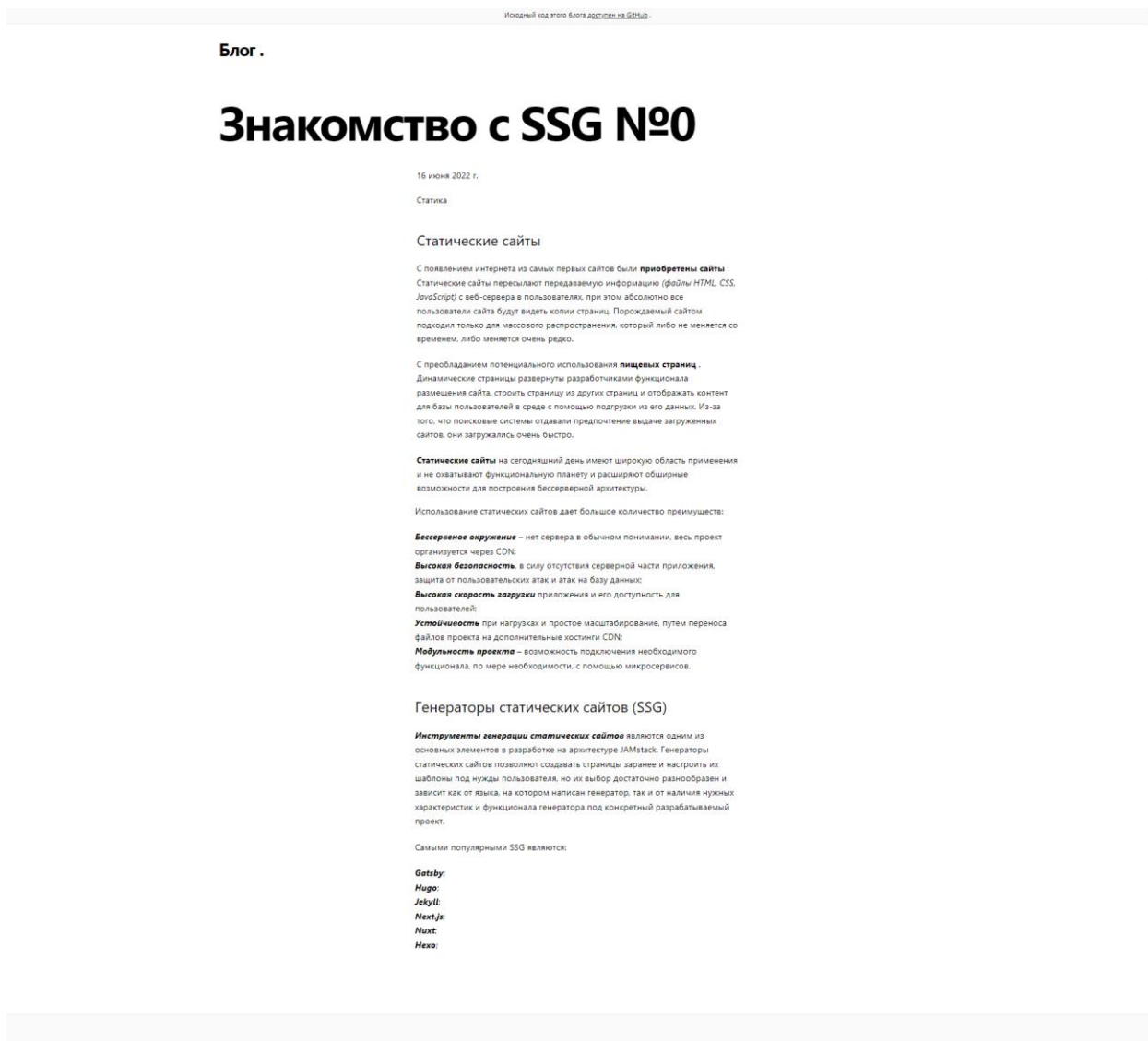


Рисунок 11 – Структура проекта на генераторе Next.js после проведения теста.

```
C:\Users\mad38\blog-starter-app>npm run build | gnomon
0.5100s
0.0002s > build
0.0001s > next build
0.4988s
0.1157s info - Checking validity of types...
1.8413s info - Checking validity of types... lint to begin setup
0.0002s info - Creating an optimized production build...
0.5864s info - Compiled successfully
0.1600s info - Collecting page data...
0.0002s info - Generating static pages (0/4)
0.2112s info - Generating static pages (1/4)
0.1713s info - Generating static pages (2/4)
0.1513s info - Generating static pages (3/4)
0.0143s info - Generating static pages (4/4)
0.0144s info - Finalizing page optimization...
0.0001s
0.0001s Page
0.0001s [
0.0001s   • /
0.0001s     /_app
0.0001s     o /404
0.0001s     • /posts/[slug] (413 ms)
0.0001s       css/cf121497002c184d.css
0.0000s       /posts/textfile0 (413 ms)
0.0000s + First Load JS shared by all
0.0000s   chunks/framework-5f4595e5518b5600.js
0.0000s   chunks/main-f65e66e62fc5ca80.js
0.0000s   chunks/pages/_app-2ea30a6251a11f8a.js
0.0000s   chunks/webpack-69bfa6990bb9e155.js
0.0000s   css/ef22ae15e7dd5510.css
0.0002s
0.0001s o (Static) automatically rendered as static HTML (uses no initial props)
0.0000s • (SSG) automatically generated as static HTML + JSON (uses getStaticProps)
0.3483s
0.0003s
Total 4.6300s
```

Рисунок 12 - Пример запуска теста для генерации файла textfile0.md в статичную страницу с использованием Next.js.



*Рисунок 13 – Сгенерированная HTML-страница на основе файла markdown в браузере на Next.js.*

**Таблица 6.**

**Результаты проведения тестов для Next.js**

Количество страниц	Время, сек
1	4,63
5	4,43
10	4,59
50	4,70
100	4,71
200	4,89
300	5,14
400	5,46
500	5,55
600	5,74

700	6,03
800	6,17
900	6,33
1000	6,67
2000	8,25
3000	9,58
4000	10,47
5000	11,05
6000	13,97
7000	15,55
8000	17,12
9000	18,48
10000	19,63

#### 2.1.4 Nuxt

Nuxt – генератор статических сайтов, который написан на JavaScript, но в отличие от Gatsby и Next.js использует фреймворк Vue. Nuxt имеет подробную документацию и интуитивно понятную установку. Nuxt также, как и Next.js дает возможность выбора статического и серверного рендеринга. Для серверного рендеринга nuxt имеет большое количество конфигурационных заготовок. Одними из основных возможностей и преимуществ nuxt являются: асинхронность данных и система их маршрутизации, наличие шаблонов для разработки, автоматическое разделение кода, интеграция с линтерами кода, поддержка TypeScript и множество других.

На рисунках 14 и 15 представлена структура проекта на Nuxt и проведение теста. Как видно из рисунка 14 nuxt имеет не такой большой размер проекта как Next.js или Gatsby. В папке «content/articles» храниться информация о markdown файлах, а за их генерацию отвечает файл \_slug.vue.



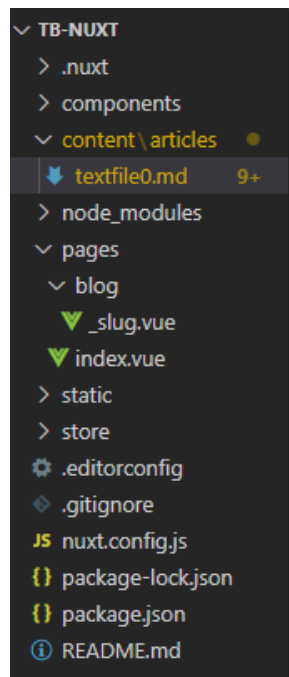


Рисунок 14 – Структура проекта на генераторе Nuxt после проведения теста.

```
C:\Users\mad38\tb-nuxt>npm run build | gnomon
0.5115s
0.0002s > tb-nuxt@1.0.0 build
0.0001s > nuxt build
0.9554s
0.8338s i Parsed 1 files in 0.3 seconds
0.0003s i Production build
0.0003s i Bundling for server and client side
0.0210s i Target: static
0.0003s i Using components loader to optimize imports
0.0011s i Discovered Components: .nuxt/components/readme.md
0.0511s ✓ Builder initialized
0.7411s ✓ Nuxt files generated
3.9215s i Compiling Client
0.0056s ✓ Client: Compiled successfully in 3.92s
0.3755s i Compiling Server
0.3755s ✓ Server: Compiled successfully in 375.12ms

Hash: 1b0745e7c806f79ea38c
Version: webpack 4.46.0
Time: 3923ms
Built at: 17.06.2022 13:29:32
    Asset      Size  Chunks             Chunk Names
  ../server/client.manifest.json  12 KiB
  274bbfe.js    6.63 KiB    3 [emitted] [immutable] components/tutorial
  2e51669.js    1.85 KiB    2 [emitted] [immutable] components/nuxt-logo
  3bb9189.js   31.6 KiB    8 [emitted] [immutable] vendors/app
  7a17e31.js   94.3 KiB    9 [emitted] [immutable] vendors/content/plugin.js
  933727a.js    5.62 KiB    4 [emitted] [immutable] content/plugin.js
  LICENSES    407 bytes
  a0e25d0.js    6.91 KiB    6, 3 [emitted] [immutable] pages/index
  ad8da1a.js    2.35 KiB    7 [emitted] [immutable] runtime
  c48b1ae.js    693 bytes    5 [emitted] [immutable] pages/blog/_slug
  c8c20c5.js    193 KiB    1 [emitted] [immutable] commons/app
  e79b17a.js   55.1 KiB    0 [emitted] [immutable] app
+ 2 hidden assets
Entrypoint app = ad8da1a.js c8c20c5.js 3bb9189.js e79b17a.js

Hash: a1c031a87be3d923c9bc
Version: webpack 4.46.0
Time: 375ms
Built at: 17.06.2022 13:29:32
    Asset      Size  Chunks             Chunk Names
components/nuxt-logo.js    5.89 KiB    1 [emitted] components/nuxt-logo
components/tutorial.js    7.81 KiB    2 [emitted] components/tutorial
pages/blog/_slug.js    1.86 KiB    3 [emitted] pages/blog/_slug
pages/index.js    9.52 KiB    4, 2 [emitted] pages/index
server.js    89.6 KiB    0 [emitted] app
server.manifest.json    547 bytes
+ 5 hidden assets
0.0022s ✓ Server: Compiled successfully in 375.12ms
0.7411s i Ready to run nuxt generate
0.0002s

Total 8.1650s
```

Рисунок 15 - Пример запуска теста для генерации файла textfile0.md в статичную страницу с использованием Nuxt.

Статика

Статические сайты

С появлением интернета одним из самых первых сайтов были – статические сайты. Статические сайты позволяют передавать фиксированную информацию (HTML, CSS, JavaScript файлы) с веб-сервера в браузер пользователей, при этом абсолютно все пользователи сайта будут видеть одинаковые странички. Поэтому статический сайт подходит только для распространения фиксированного контента, который либо не меняется со временем, либо меняется очень редко.

С дальнейшим развитием стало популярно использование динамических страниц. Динамические страницы позволяют разработчикам (увеличить функционал сайта, строить страничку из множества других страниц и отображать контент для пользователя в реальном времени с помощью подгрузки его из базы данных). Но минусом использования динамических страниц была их долгая загрузка, исходя из этого появившиеся системы отдавали предпочтение в выборе статических сайтов, поскольку они загружаются очень быстро.

Статические сайты на сегодняшний день имеют широкую область применения и не уступают динамическим в функциональности плане и обладают обширным количеством инструментов для построения полноценной беспроблемной архитектуры.

Использование статических сайтов дает большое количество преимуществ:

- 1. **Беспроблемная архитектура** – нет сервера в обычном понимании, весь проект организуется через CDN;
- 2. **Высокая безопасность**, в силу отсутствия серверной части приложения, защита от пользовательских атак и атак на базу данных;
- 3. **Высокая скорость загрузки приложения и его доступность для пользователей**;
- 4. **Удобственность** при нагрузках и простоем масштабировании, путем сорочки файлов проекта на дополнительные копии CDN;
- 5. **Модальность времени** – возможность подключения необходимого функционала, по мере необходимости, с помощью микросервисов.

Генераторы статических сайтов (SSG)

Использование генерации статических сайтов является одним из основных элементов в разработке на архитектуре JAMstack. Генераторы статических сайтов позволяют создавать странички заранее и встроить их шаблоны под нужды пользователя, но их выбор достаточно разнообразен и зависит как от языка, на котором написан генератор, так и от наличия нужных характеристик и функционала генератора под конкретной разрабатываемый проект.

Самыми популярными SSG являются:

- 1. **Gatsby**;
- 2. **Next**;
- 3. **Jekyll**;
- 4. **SvelteKit**;
- 5. **Vue**;
- 6. **Nuxt**;

Рисунок 16 - Сгенерированная HTML-страница на основе файла  
markdown в браузере на Nuxt.

Таблица 7.

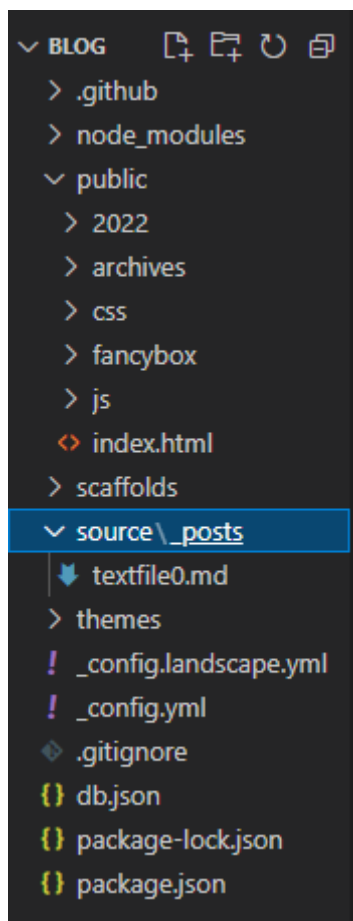
Результаты проведения тестов для Nuxt

Количество страниц	Время, сек
1	8,16
5	8,24
10	8,21
50	8,50
100	8,66
200	8,96
300	9,29
400	9,59
500	9,94
600	10,34
700	10,64
800	10,87
900	11,30
1000	11,59
2000	14,54
3000	17,56
4000	20,68
5000	23,93
6000	27,76
7000	29,66
8000	33,99
9000	36,54
10000	41,08

## 2.1.5 Нехо

Нехо – один из самых удобных в настройке генераторов статических сайтов, написанный на JavaScript. Нехо обладает быстрой скоростью генерации небольшого количества страниц и позволяет использовать возможности плагинов Octopress. Нехо предоставляет мощные API для безграничной расширяемости. Доступны различные плагины для поддержки большинства шаблонизаторов (EJS, Pug, Nunjucks и многих других). Легко интегрируется с существующими NPM-пакетами (Babel, PostCSS, Less/Sass и т.д.).

Структура проекта на Нехо и реализация теста представлены на рисунках 17 и 18 соответственно. Папка «public» хранит кэш и сгенерированные страницы, а папка «source/\_posts» хранит файлы markdown.



*Рисунок 17 – Структура проекта на генераторе Нехо после проведения теста.*

```
C:\Users\mad38\blog>hexo generate | gnomon
0.7505s INFO Validating config
0.1967s INFO Start processing
0.1583s INFO Files loaded in 325 ms
0.0004s INFO Generated: archives/2022/index.html
0.0001s INFO Generated: archives/index.html
0.0001s INFO Generated: index.html
0.0001s INFO Generated: archives/2022/06/index.html
0.0002s INFO Generated: 2022/06/16/textfile0/index.html
0.0451s INFO 5 files generated in 159 ms
0.0003s
Total 1.1534s
```

Рисунок 18 - Пример запуска теста для генерации файла `textfile0.md` в статичную страницу с использованием *Нехо*.

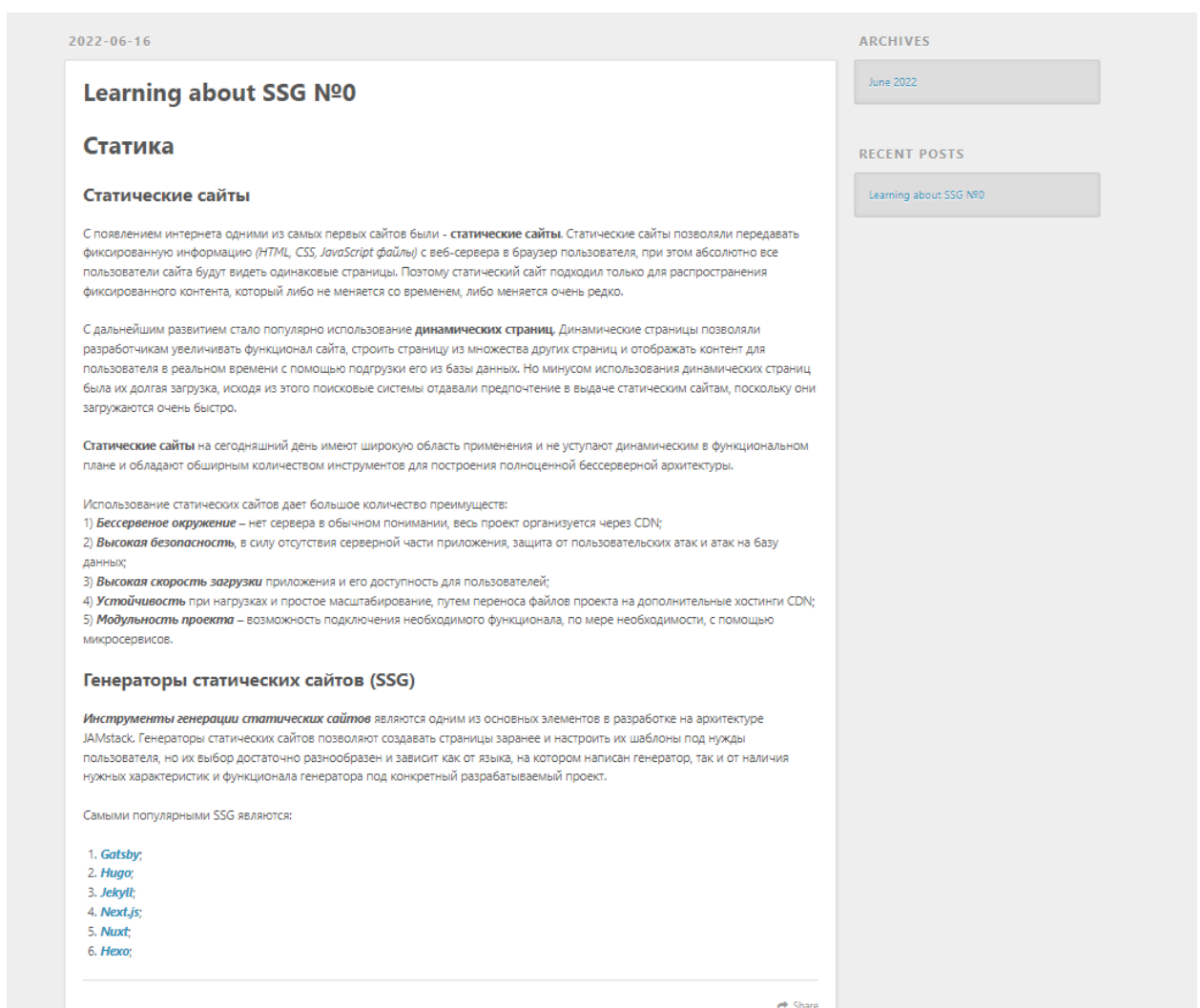


Рисунок 19 - Сгенерированная HTML-страница на основе файла `markdown` в браузере на *Nuxt*.

Таблица 8.

**Результаты проведения тестов для Нехо**

Количество страниц	Время, сек
1	1,15
5	1,20
10	1,25
50	1,42
100	1,68
200	2,07
300	2,43
400	2,81
500	3,28
600	3,62
700	4,03
800	4,44
900	4,79
1000	7,09
2000	12,17
3000	16,57
4000	21,19
5000	25,87
6000	30,25
7000	32,69
8000	36,66
9000	39,83
10000	44,23

**2.2 Анализ результатов**

Для анализа получившихся результатов времени сборки каждым генератором составим сводную таблицу данных (таблица 9). Из таблицы видно, что самые большие показатели времени у Gatsby, а самые маленькие у Hugo. Близким по значениям к Gatsby при количестве страниц свыше 1000 является фреймворк Nuxt и генератор Нехо. Для наглядности данных построим два графика с количеством страниц <1000, как для небольшого корпоративного сайта или блога и 1000-10000 страниц, как для достаточно крупного информационного ресурса или веб-приложения.

Таблица 9.

**Сводная таблица данных по проведенным тестам для каждого генератора статических сайтов**

	Gatsby	Hugo	Jekyll	Next.js	Nuxt	Нexo
Страницы	Время, сек					
1	17,31	0,07	0,37	4,63	8,16	1,15
5	17,07	0,08	0,38	4,43	8,24	1,20
10	16,87	0,08	0,40	4,59	8,21	1,25
50	17,35	0,10	0,45	4,70	8,50	1,42
100	17,43	0,11	0,51	4,71	8,66	1,68
200	17,85	0,14	0,63	4,89	8,96	2,07
300	18,21	0,17	0,74	5,14	9,29	2,43
400	18,49	0,21	0,86	5,46	9,59	2,81
500	18,81	0,23	0,98	5,55	9,94	3,28
600	19,26	0,26	1,07	5,74	10,34	3,62
700	19,25	0,30	1,19	6,03	10,64	4,03
800	19,81	0,31	1,34	6,17	10,87	4,44
900	20,19	0,35	1,44	6,33	11,30	4,79
1000	20,53	0,36	1,56	6,67	11,59	7,09
2000	23,88	0,69	2,76	8,25	14,54	12,17
3000	26,96	0,98	3,97	9,58	17,56	16,57
4000	30,99	1,27	5,14	10,47	20,68	21,19
5000	33,99	1,59	6,36	11,05	23,93	25,87
6000	37,28	1,90	7,60	13,97	27,76	30,25
7000	39,93	2,19	8,79	15,55	29,66	32,69
8000	44,57	2,49	9,94	17,12	33,99	36,66
9000	52,21	4,23	11,12	18,48	36,54	39,83
10000	63,14	3,32	12,34	19,63	41,08	44,23

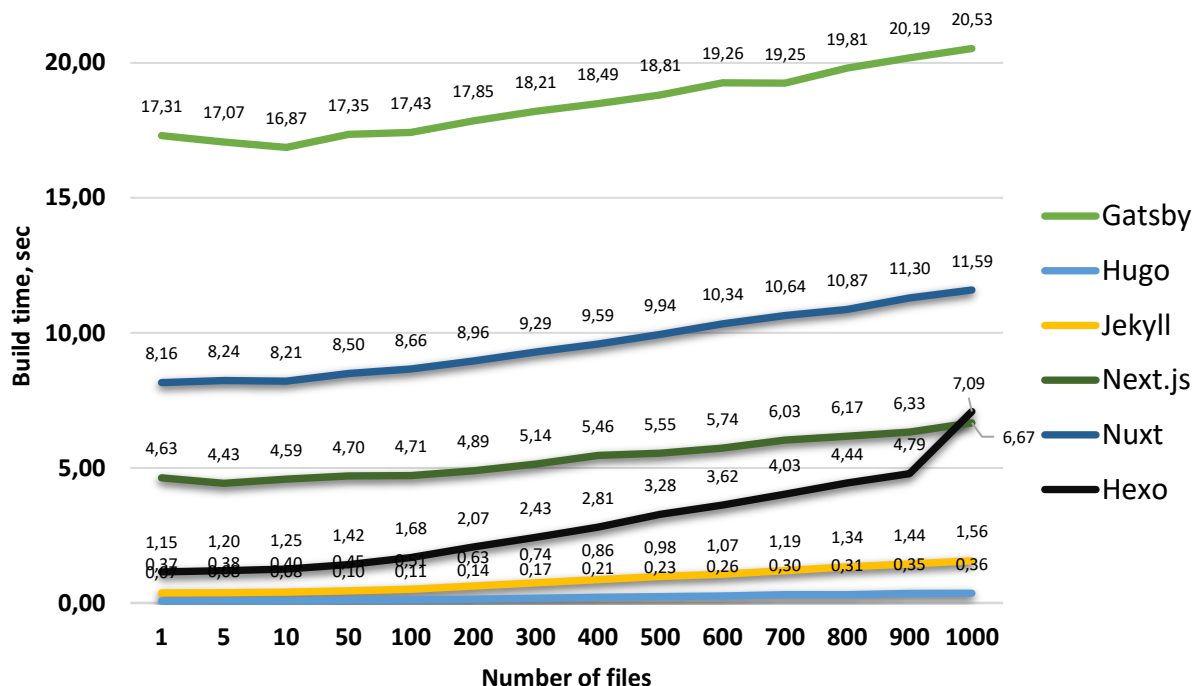


Рисунок 20 – График времени сборки в зависимости от количества страниц (до 1000 страниц) по всем выбранным генераторам.

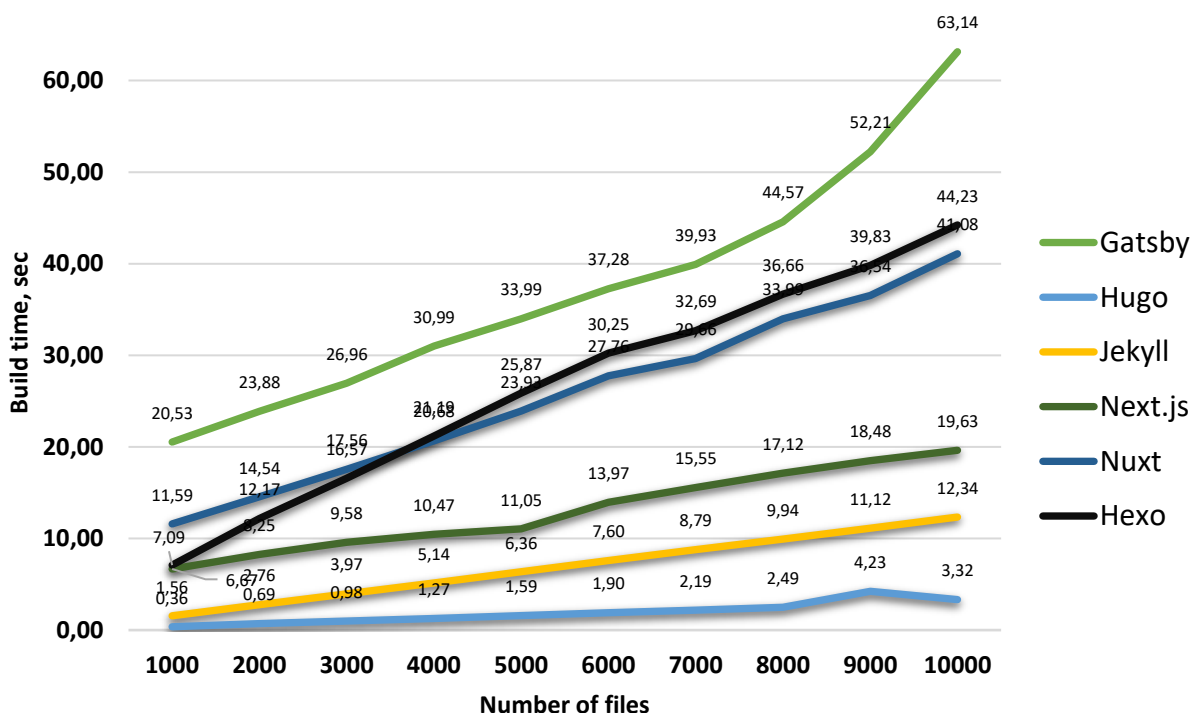
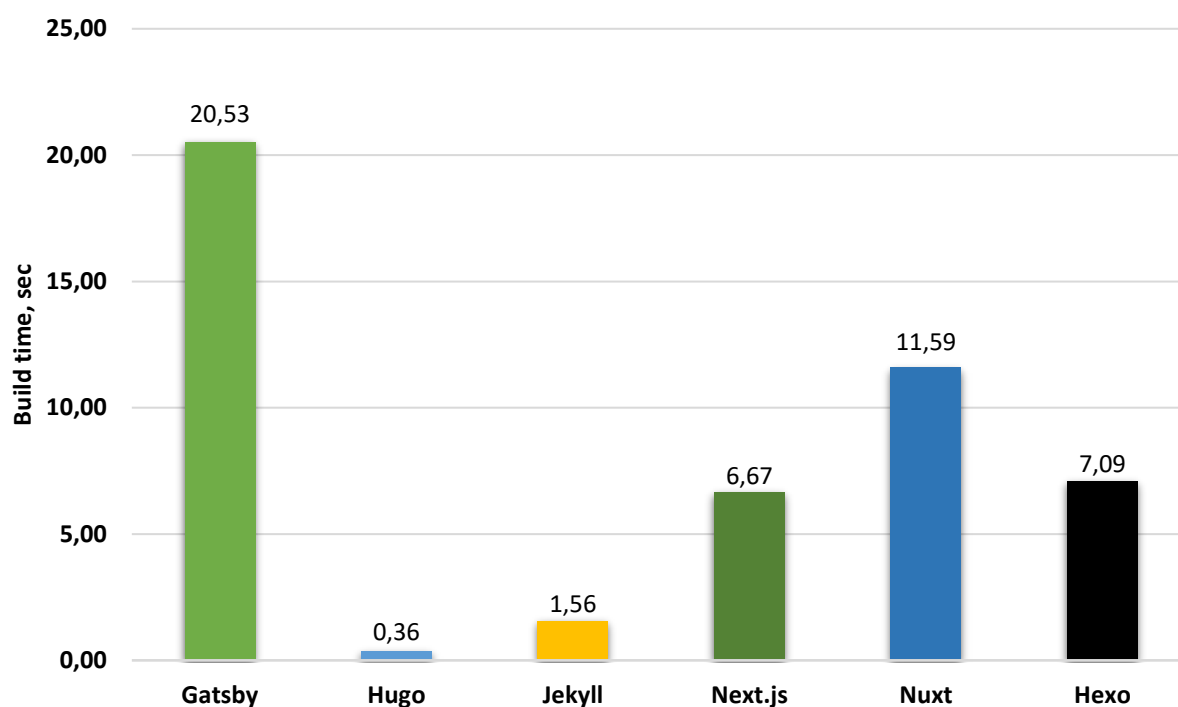


Рисунок 21 – График времени сборки в зависимости от количества страниц (до 1000 страниц) по всем выбранным генераторам.

Из графиков наглядно видно, насколько Gatsby сильно отстает по показателям от остальных генераторов. Однако стоит уточнить, что Gatsby

обладает хорошими кэширующими плагинами, которые позволяют значительно сократить время сборки, но не более чем на половину от данного времени. Также, наглядно видно, как сильно увеличивается время сборки у генератора Нехо по сравнению с остальными.

Для оценки времени сборки по фиксированному количеству страниц (1000) построим диаграмму (рисунок 22):



*Рисунок 22 – Диаграмм времени сборки для 1000 страниц по всем выбранным генераторам.*

Таким образом видно, что выбирая между тремя комплексными платформами, использующими JavaScript: Gatsby, Next.js, Nuxt, Next.js показывает наилучшие результаты в данных текстах, во много раз опережая Gatsby и почти наполовину от Nuxt. Если сравнивать остальные три генератора, с не таким богатым набором плагинов и функционала, как у предыдущих, то лидером без сомнения является Hugo, как и по результатам тестов, так и по возможностям для разработки.



## ЗАКЛЮЧЕНИЕ

В ходе реализации курсового проекта были решены все поставленные задачи. Были изучены различия в применении и реализации статических и динамических сайтов. Проведен анализ возможностей и спецификации языка разметки Markdown. Проведен анализ существующих генераторов статических сайтов и их классификации.

Проведен эксперимент, в рамках которого было оценено время сборки статических страниц выбранными генераторами статических сайтов в зависимости от их количества. Результаты эксперимента показали, что среди трех больших платформ, таких как Gatsby, Next.js и Nuxt самое лучшее время сборки показывает Next.js (19,63 секунд для 10000 страниц).

Сравнивая генераторы для небольших проектов с скромным функционалом: Hugo, Jekyll и Нехо, лидером во всех тестах является Hugo (0,36 секунд для 1000 страниц), результаты времени сборки которого в большинстве показали менее одной секунды.

## СПИСОК ИСОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Никитинская В. М., Сокурено Ю. А., Государев И. Б. ОБЗОР СОВРЕМЕННЫХ ПРАКТИК СОЗДАНИЯ СТАТИЧЕСКИХ ВЕБ-САЙТОВ //АЛЬМАНАХ НАУЧНЫХ РАБОТ МОЛОДЫХ УЧЕНЫХ УНИВЕРСИТЕТА ИТМО. – 2018. – С. 229-231 URL: <https://elibrary.ru/item.asp?id=36987043> (дата обращения: 10.06.2022).
2. Хорошевич П. А. ИСПОЛЬЗОВАНИЕ ЯЗЫКА РАЗМЕТКИ MARKDOWN ДЛЯ РАЗРАБОТКИ СРЕДСТВ ПРЕДСТАВЛЕНИЯ УЧЕБНОЙ ИНФОРМАЦИИ //Актуальные проблемы и направления цифровой трансформации образования. – 2021. – С. 224-228. URL: <https://www.elibrary.ru/item.asp?id=48170009> (дата обращения: 10.06.2022).
3. Никитинская В. М., Государев И. Б. КЛАССИФИКАЦИЯ ИНСТРУМЕНТОВ ГЕНЕРАЦИИ СТАТИЧЕСКИХ ВЕБ-САЙТОВ //АЛЬМАНАХ НАУЧНЫХ РАБОТ МОЛОДЫХ УЧЕНЫХ УНИВЕРСИТЕТА ИТМО. – 2019. – С. 216-219 URL: <https://elibrary.ru/item.asp?id=42382818> (дата обращения: 10.06.2022).
4. Rinaldi B. Static site generators. – O'Reilly Media, 2015 URL: [https://link.springer.com/chapter/10.1007/978-1-4842-1464-0\\_3](https://link.springer.com/chapter/10.1007/978-1-4842-1464-0_3) (дата обращения: 10.06.2022).
5. JAMstack [Электронный ресурс] URL: <https://jamstack.org/> (дата обращения: 10.06.2022).
6. Gatsby [Электронный ресурс] URL: <https://www.gatsbyjs.com/> (дата обращения: 10.06.2022).
7. Hugo [Электронный ресурс] URL: <https://gohugo.io/> (дата обращения: 10.06.2022).
8. Jekyll [Электронный ресурс] URL: <https://jekyllrb.com/> (дата обращения: 10.06.2022).

9. Next.js [Электронный ресурс] URL: <https://nextjs.org/> (дата обращения: 10.06.2022).
10. Nuxt [Электронный ресурс] URL: <https://nuxtjs.org/> (дата обращения: 10.06.2022).
11. Нехо [Электронный ресурс] URL: <https://hexo.io/ru/> (дата обращения: 10.06.2022).

**"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"**  
**(УНИВЕРСИТЕТ ИТМО)**

**ОТЗЫВ РУКОВОДИТЕЛЯ**  
**о выполнении курсового проекта (работы)**

Студент \_\_\_\_\_ Морозов А.Д.  
(Фамилия, И., О.)

Факультет \_\_\_\_\_ ПИиКТ \_\_\_\_\_ Группа \_\_\_\_\_ Р41091

Направление (специальность) \_\_\_\_\_ 09.04.04 Программная инженерия

Руководитель \_\_\_\_\_ Государев И.Б., доцент  
(Фамилия, И., О., должность)

Дисциплина \_\_\_\_\_ Проектирование и анализ языков веб-решений

Наименование темы: \_\_\_\_\_ Анализ времени сборки проекта на JAMstack генераторами статических  
сайтов в зависимости от числа страниц

**ОЦЕНКА КУРСОВОГО ПРОЕКТА (РАБОТЫ)**

№ п/п	Показатели	Оценка			
		5	4	3	0
1.	Проект создан обучающимся самостоятельно				
2.	Созданные элементы сайта раскрывают тематику и название фирмы				
3.	Проект технологически грамотный				
4.	Оформление отвечает требованиям к отчету				
5.	Во время защиты обучающийся показал умение кратко, доступно представить результаты работы, умение анализировать, аргументировать свою точку зрения, делать обобщение и выводы, адекватно ответить на поставленные вопросы.				
ИТОГОВАЯ ОЦЕНКА					

**Отмеченные достоинства:**

В отчете студента отражены полученные в ходе выполнения проекта навыки, соответствующие компетенциям по данной тематике. Студент продемонстрировал способность к исследованиям и анализу. Студент показал себя личностью пунктуальной, ответственной, готовой к изучению нового материала. В процессе работы студент подтвердил навыки в области проектирования веб-ресурсов на JAMstack. Студент оценил время сборки каждым генератором статических сайтов и провел анализ общей статистики, при помощи которой была выполнена цель курсового проекта.

---

---

---

---

---

**Отмеченные недостатки:**

---

---

---

---

---

**Заключение:**

Студент подтвердил навыки, полученные за время обучения по указанной специальности.

---

---

---

---

---

Руководитель

---

  
(подпись)И.Б. Государев

« \_\_\_\_ » \_\_\_\_\_ 20\_\_