МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
ЧАСТНОЕ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«ГРОДНЕНСКИЙ КОЛЛЕДЖ БИЗНЕСА И ПРАВА»

СОЗДАНИЕ, ВЕРСТКА И ТЕСТИРОВАНИЕ САЙТА НА ОСНОВЕ ШАБЛОНА ИЗ СЕРВИСА FIGMA

ОП I.32ПО0018

Руководители практики: П.С.Равданович

Учащийся Д.В.Дударчук

2023

СОДЕРЖАНИЕ

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Введение

Проект "Верстка сайта на основе шаблона из сервиса Figma" является задачей разработки веб-сайта, основанного на дизайн-шаблоне, созданном с использованием популярного инструмента дизайна Figma. Цель проекта - создать функциональный и привлекательный веб-сайт, соответствующий оригинальному дизайн-шаблону, с использованием технологий веб-разработки, таких как HTML, CSS и JavaScript.

Процесс верстки сайта включает в себя анализ дизайн-шаблона, создание структуры HTML-разметки, оформление стилей CSS для внешнего вида и визуального оформления сайта, а также добавление интерактивности с использованием JavaScript для создания динамических элементов и взаимодействия пользователя с сайтом.

Проект также включает тестирование и отладку, чтобы убедиться в корректной работе сайта на различных устройствах, браузерах и разрешениях экрана. Оптимизация и оптимизация производительности также являются важной частью проекта, чтобы обеспечить быструю загрузку страниц и улучшить производительность сайта.

Оформление проекта в виде студенческого отчета позволяет систематизировать и описать в деталях все этапы процесса верстки сайта на основе дизайн-шаблона из сервиса Figma, включая анализ дизайна, создание структуры HTML, оформление стилей CSS, добавление интерактивности с использованием JavaScript, тестирование, оптимизацию и заключение. Этот проект предоставляет возможность приобрести практические навыки веб-разработки и применить их на практике, а также изучить процесс верстки сайта на основе дизайн-шаблона из сервиса Figma.

1.2 Коротко о HTML

HTML (HyperText Markup Language) - это стандартный язык разметки, используемый для создания веб-страниц. Он состоит из набора тегов (markup tags), которые определяют структуру и содержимое веб-страницы. HTML теги представляются в угловых скобках (< >) и могут содержать атрибуты для определения дополнительных свойств элементов.

Основные понятия в HTML:

1. Элементы (Elements): они являются строительными блоками HTML и представляют собой отдельные компоненты веб-страницы, такие как заголовки, абзацы, изображения, ссылки и др. Элементы создаются с использованием HTML тегов и могут содержать внутреннее содержимое, а также атрибуты, определяющие их свойства.

Пример элемента:

```
<p>Это абзац текста</p>
```

2. Теги (Tags): они используются для определения элементов в HTML. Теги имеют открывающую и закрывающую части, и между ними располагается содержимое элемента.

Пример тега:

```
<p>Это абзац текста</p>
```

3. Атрибуты (Attributes): они используются для определения дополнительных свойств элементов. Атрибуты указываются в открывающем теге и содержат имя и значение.

Пример атрибута:

```
<img src "image.jpg" alt "Изображение">
```

4. Структура документа: HTML документ состоит из декларации типа документа (Document Type Declaration или DOCTYPE), открывающего

и закрывающего тегов <html>, <head> и <body>. Внутри тега <head> находится мета-информация о документе, такая как заголовок страницы, мета-теги, стили и скрипты. Внутри тега <body> находится содержимое страницы, такое как текст, изображения, ссылки и другие элементы.

Пример структуры HTML документа:

```
<!DOCTYPE html>
<html>
<head>
  <title>Заголовок страницы</title>
  <!-- Мета-теги, стили и скрипты -->
</head>
<body>
  <!-- Содержимое страницы -->
</body>
</html>
```

HTML позволяет создавать структурированный контент, определять его семантику, создавать ссылки, встроенные изображения, таблицы, формы и многое другое. Он является основным языком разметки для создания.

1.3 Коротко о JavaScript

JavaScript - это высокоуровневый язык программирования, который широко используется для создания интерактивных веб-страниц и приложений. Он является одним из трех основных технологических стандартов веб-разработки, вместе с HTML и CSS.

Основные особенности JavaScript:

1. Интерактивность: JavaScript позволяет создавать динамические веб-страницы, взаимодействовать с пользователями и реагировать на их действия. Это включает обработку событий (например, кликов мыши, нажатий клавиш), изменение содержимого страницы, создание анимаций, валидацию форм и многое другое.

2. Объектно-ориентированное программирование: JavaScript поддерживает объектно-ориентированную парадигму программирования, позволяя создавать и использовать объекты с методами и свойствами. Он также предлагает множество встроенных объектов, таких как объекты для работы с датами, массивами, строками и другими типами данных.
3. Мультипарадигменность: JavaScript поддерживает не только объектно-ориентированное программирование, но и функциональное программирование. Это позволяет разработчикам выбирать подход, который лучше всего соответствует конкретным требованиям и структуре их кода.
4. Кросс-платформенность: JavaScript может выполняться в браузере на стороне клиента (клиентский JavaScript) и на сервере (серверный JavaScript). Это делает его кросс-платформенным языком, который может быть использован как на клиентской стороне веб-приложений, так и на сервере для разработки серверных приложений и API.
5. Богатая экосистема: JavaScript имеет большую и активную сообщество разработчиков, которое создает и поддерживает множество библиотек, фреймворков и инструментов для упрощения разработки. Это включает такие популярные фреймворки, как React, Angular, Vue, библиотеки для работы с AJAX, работы с DOM, обработки данных и другие.

Применение JavaScript включает создание динамических веб-страниц, разработку веб-приложений, взаимодействие с сервером, обработку данных, валидацию форм, создание анимаций, реализацию сложных функциональностей на веб-страницах и многое другое.

JavaScript также широко используется в различных областях разработки, таких как:

1. Фронтенд-разработка: JavaScript является основным языком программирования для создания динамических пользовательских интерфейсов на веб-страницах. С его помощью разработчики могут создавать интерактивные элементы, взаимодействовать с пользователем, создавать анимации и многое другое.
2. Бэкенд-разработка: JavaScript также используется на серверной стороне для создания серверных приложений и API. Серверные фреймворки, такие как Node.js, Express, Koa и другие, позволяют разработчикам создавать масштабируемые и эффективные серверные приложения, основанные на JavaScript.
3. Разработка мобильных приложений: С использованием фреймворков и инструментов, таких как React Native, Ionic, Apache Cordova и других, JavaScript можно использовать для разработки мобильных приложений, работающих на различных платформах, таких как iOS, Android и Windows.
4. Интернет вещей (IoT): JavaScript также находит применение в разработке приложений для управления устройствами Интернета вещей, такими как умные дома, умные города и другие умные устройства.
5. Игровая разработка: JavaScript используется для создания веб-игр и игровых приложений, как на стороне клиента, так и на стороне сервера, с использованием фреймворков и библиотек, таких как Phaser, Three.js, Babylon.js и других.

JavaScript имеет множество возможностей и широкий спектр применения, что делает его одним из наиболее популярных языков программирования в веб-разработке и не только. Он продолжает развиваться, и его экосистема постоянно обновляется с появлением новых инструментов, фреймворков и библиотек для более эффективной разработки веб-приложений и других приложений.

1.4 Коротко о CSS

CSS (Cascading Style Sheets) - это язык стилей, используемый в веб-разработке для определения внешнего вида и оформления веб-страниц. CSS позволяет разработчикам определять стили элементов HTML, таких как цвет, шрифт, размеры, отступы, маргины, выравнивание и другие атрибуты, которые влияют на внешний вид веб-страницы.

Основные принципы работы CSS:

1. Селекторы: CSS использует селекторы, чтобы указывать на элементы HTML, которые нужно стилизовать. Селекторы могут быть классами, идентификаторами, тегами, псевдоклассами, псевдоэлементами и другими, и они позволяют разработчикам точно указать, какие элементы должны быть стилизованы.
2. Свойства: CSS свойства определяют, какие стили будут применены к выбранным элементам. Примеры свойств включают цвет, шрифт, размеры, отступы, маргины, фон, границы и многое другое.
3. Значения: Значения определяют конкретные значения, которые будут применены к свойствам. Например, цвет может быть задан в виде имени цвета (например, "красный"), шестнадцатеричного кода цвета (например, "#FF0000") или функции (например, "rgb(255, 0, 0)").
4. Каскадность: CSS поддерживает каскадность, что означает, что стили могут быть наследованы от родительских элементов и переопределены на более низком уровне. Это позволяет разработчикам создавать гибкие и масштабируемые стили, которые могут быть применены к разным элементам на странице.
5. Бокс-модель: CSS основан на концепции бокс-модели, где каждый элемент HTML представляется как прямоугольный блок с определенными размерами, отступами, границами и внутренним содержимым. С помощью CSS можно контролировать и

настраивать эти аспекты бокс-модели, чтобы создавать различные макеты и дизайны.

CSS также поддерживает медиазапросы, позволяющие создавать адаптивные веб-страницы, которые могут корректно отображаться на различных устройствах и экранах, таких как мобильные устройства, планшеты, настольные компьютеры и другие. Медиазапросы позволяют адаптировать стили в зависимости от ширины экрана или других параметров устройства, что позволяет создавать удовлетворительный пользовательский опыт на разных устройствах.

Каскадность и наследование в CSS также позволяют создавать структурированный и эффективный код. Стили могут быть определены на глобальном уровне, что позволяет применять их к нескольким элементам на странице одновременно, и при необходимости они могут быть переопределены на уровне отдельных элементов или классов.

Одним из основных преимуществ CSS является его способность отделить оформление от содержимого. Это позволяет разработчикам создавать гибкие и масштабируемые стили, которые могут быть легко изменены и адаптированы без изменения структуры HTML-кода. Это также делает сайты более доступными и поддерживаемыми, так как структура и содержимое остаются независимыми от стилей.

В целом, CSS является мощным инструментом для оформления веб-страниц, позволяющим создавать эстетически привлекательные и функциональные веб-дизайны. Знание CSS является важным навыком для веб-разработчиков, позволяющим создавать современные и профессиональные веб-сайты.

1.5 Коротко о сервисе Figma

Figma - это инструмент для дизайна интерфейсов, который предоставляет возможность создавать веб-дизайн, мобильные приложения, иконки, прототипы

и другие графические проекты. Figma является популярным сервисом в индустрии веб-дизайна и используется многими профессиональными дизайнерами для создания высококачественных и современных пользовательских интерфейсов.

Основные особенности и возможности Figma включают:

1. Коллаборация в режиме реального времени: Figma позволяет нескольким дизайнерам работать над одним проектом одновременно, обеспечивая возможность командной работы и обмена мнениями.
2. Векторное редактирование: Figma позволяет создавать векторные элементы, такие как иконки и логотипы, с возможностью редактирования и масштабирования без потери качества.
3. Библиотеки компонентов: Figma позволяет создавать библиотеки компонентов, которые могут быть повторно использованы на различных страницах и проектах, что обеспечивает консистентность дизайна и упрощает его обновление.
4. Прототипирование: Figma предоставляет возможность создавать интерактивные прототипы с анимациями, переходами и состояниями элементов, что позволяет тестировать пользовательский опыт и взаимодействие на ранних стадиях проектирования.
5. Интеграция с другими инструментами: Figma интегрируется с другими популярными инструментами для дизайна и разработки, такими как Sketch, Zeplin, InVision и др., что облегчает рабочий процесс и взаимодействие между различными членами команды.
6. Безопасность и доступность: Figma предоставляет возможность настройки прав доступа к проектам, шифрует данные и обеспечивает резервное копирование проектов, что обеспечивает защиту данных и сохранность проектов.

Figma является мощным и гибким инструментом для создания профессионального веб-дизайна, который позволяет дизайнерам работать эффективно и совместно, создавая высококачественные и современные пользовательские интерфейсы.

2 ПРОЦЕСС ВЕРСТКИ

2.1 Планирование

Первым этапом процесса проектирования было изучение шаблона из сервиса Figma. Были проанализированы все основные элементы дизайна, такие как цветовая палитра, шрифты, компоненты и композиция страницы. Был также изучен взаимодействие между различными блоками на странице, такими как шапка, основной блок, портфолио, форма обратной связи и подвал.

На основе изученного шаблона были определены основные элементы дизайна, которые должны быть включены в создаваемый веб-сайт. Это включало выбор цветовой палитры, определение типов шрифтов и их размеров, выбор компонентов, таких как кнопки, иконки и формы, и определение композиции страницы, включая расположение блоков и элементов.

Следующим шагом было создание макета страницы, основываясь на определенных элементах дизайна. Был разработан макет, включающий шапку, основной блок, портфолио, форму обратной связи и подвал, в соответствии с композицией страницы из шаблона. Был учтен адаптивный дизайн, чтобы сайт корректно отображался на различных устройствах и экранах.

2.2 Дизайн

На основе принятых решений по дизайну были созданы компоненты, такие как заголовки, тексты, кнопки, формы, изображения и другие элементы

интерфейса. Были учтены размеры, цвета, стили и другие аспекты, соответствующие выбранному дизайну.

После создания компонентов дизайна были оформлены стили с использованием CSS. Были заданы цвета, шрифты, размеры, отступы, рамки и другие свойства стилей, чтобы придать компонентам визуальное оформление в соответствии с выбранным дизайном.

Для обеспечения корректного отображения сайта на различных устройствах был разработан адаптивный дизайн. Были определены точки разрыва (breakpoints) для адаптации компонентов дизайна на различных экранах, таких как мобильные устройства, планшеты и настольные компьютеры. Были применены медиа-запросы и другие техники CSS, чтобы обеспечить оптимальное отображение сайта на разных устройствах.

После создания дизайна было проведено тестирование на различных устройствах и браузерах, чтобы убедиться в корректности отображения компонентов дизайна и их функциональности. Были исправлены все выявленные проблемы и ошибки, чтобы гарантировать качественное отображение и пользовательский опыт на всех устройствах.

Процесс создания дизайна веб-сайта на основе шаблона из сервиса Figma включал изучение шаблона, проектирование макета страницы, принятие решений по дизайну, создание компонентов дизайна, оформление стилей, создание адаптивного дизайна и тестирование. В результате был разработан привлекательный и функциональный дизайн сайта, который соответствует заданным требованиям и обеспечивает оптимальное отображение на различных устройствах.

2.3 Верстка

Целью данного проекта было создание веб-сайта на основе готового дизайн-шаблона из сервиса Figma. Веб-сайт разработан с использованием технологий HTML, CSS и JavaScript, и представляет собой одностраничный сайт

с адаптивным дизайном, содержащий различные блоки информации и стилевые компоненты.

Первым шагом процесса верстки было изучение дизайн-шаблона из сервиса Figma. Была проведена детальная анализ макета, включая расположение компонентов, размеры элементов, цветовую схему, шрифты, а также интерактивные элементы, такие как кнопки и ссылки. Важной частью этого шага было также определение структуры и организации контента на сайте, а также понимание общей концепции и стиля дизайна.

На основе анализа дизайн-шаблона была разработана структура HTML-документа. Были определены основные блоки информации и контент, такие как заголовки, параграфы, списки и изображения. Были созданы соответствующие HTML-элементы и добавлены необходимые атрибуты и классы для структурирования и организации контента на странице.

После создания структуры HTML было разработано оформление стилей с использованием CSS. Были определены стили для каждого компонента дизайна, такие как цвета, шрифты, размеры, отступы и границы. Были использованы селекторы, классы и псевдоклассы для точного определения стилей для каждого компонента. Были также созданы медиа-запросы и другие техники CSS, чтобы обеспечить адаптивность и корректное отображение сайта на различных устройствах.

Для придания динамичности сайту был использован JavaScript. Были добавлены скрипты для реализации интерактивных элементов, таких как анимации, модальные окна, слайдеры и другие функциональности. Были использованы современные технологии JavaScript, такие как ES6+ стандарт, чтобы обеспечить оптимальную производительность.

ES6+ — это сокращение от "**ECMAScript 6** и более новые версии", и оно относится к последним версиям стандарта ECMAScript - языка программирования, на котором основан JavaScript. ECMAScript - это стандарт, разработанный и поддерживаемый организацией Ecma International, который определяет синтаксис, семантику и особенности языка JavaScript.

ES6, также известный как ECMAScript 2015, был выпущен в 2015 году и представил множество новых возможностей и улучшений в языке JavaScript, включая:

Лексические блоки и стрелочные функции: Введение нового синтаксиса для определения функций, включая более краткие и понятные стрелочные функции, а также возможность использования лексических блоков, что позволяет определять локальные области видимости переменных.

Константы (const) и переменные (let): Введение ключевых слов const и let для объявления констант и переменных соответственно, вместо использования var. const создает переменные, значения которых не могут быть изменены, а let создает переменные с блочной областью видимости.

Деструктуризация: Возможность извлекать значения из объектов и массивов с помощью более краткого и удобного синтаксиса.

Расширенные возможности работы с объектами и массивами: Включая новые методы, такие как spread оператор, rest параметры, и возможность определения свойств объекта с помощью динамических вычисляемых имен.

Классы: Введение синтаксиса классов, который облегчает создание объектно-ориентированного кода в JavaScript.

Promise: Введение нового объекта Promise, предоставляющего мощные средства работы с асинхронным кодом и управления асинхронными операциями.

Модули: Возможность организации кода в модули, что позволяет более удобно организовывать и поддерживать большие проекты.

И другие улучшения, такие как усовершенствованный синтаксис для работы с функциями, операторы для работы с итераторами и генераторами, возможность работы с двоичными данными и т. д.

ES6+ также включает все последующие версии стандарта ECMAScript, такие как ES7, ES8, ES9 и так далее, каждая из которых вносит новые функциональности и улучшения в JavaScript.

2.4 Тестирование

Первый этап тестирования включал подготовку к тестированию. Были определены цели и задачи тестирования, разработаны тестовые сценарии и тестовые данные. Также был проведен анализ требований и дизайна шаблона для определения ожидаемого функционала и внешнего вида веб-сайта.

Было проведено тестирование функциональности веб-сайта. Были проверены все интерактивные элементы, такие как кнопки, ссылки, формы обратной связи и другие элементы в соответствии с тестовыми сценариями. Были также проведены тесты на валидацию форм и проверка корректного отображения контента на различных устройствах и экранах.

Было проведено тестирование совместимости веб-сайта с различными веб-браузерами, такими как Google Chrome, Mozilla Firefox, Microsoft Edge, Safari и другими популярными браузерами. Было проверено корректное отображение веб-сайта на различных браузерах, а также функциональность и взаимодействие с элементами сайта.

Было проведено тестирование адаптивности веб-сайта на различных устройствах и экранах, включая настольные компьютеры, ноутбуки, планшеты и смартфоны. Было проверено корректное отображение и функциональность веб-сайта на различных устройствах, а также адекватность адаптивного дизайна в соответствии с оригинальным шаблоном из Figma.

Было проведено тестирование производительности веб-сайта, включающее проверку времени загрузки страницы, скорости отклика, оптимизации ресурсов и других аспектов производительности. Было также проведено т естирование работы сайта при различных условиях нагрузки, таких как большое количество пользователей, медленное интернет-соединение и другие факторы.

Было проведено тестирование безопасности веб-сайта, включая проверку на наличие уязвимостей, защиту от вредоносных атак, проверку на корректную

обработку пользовательских данных, защиту от XSS, CSRF и других типов атак. Были также проведены тесты на безопасность передачи данных между клиентом и сервером, такие как HTTPS-шифрование и корректность обработки пользовательских авторизационных данных.

Было проведено тестирование пользовательского опыта (UX), включающее оценку удобства использования веб-сайта, навигации, взаимодействия с элементами сайта, читаемости контента и других аспектов, влияющих на удовлетворенность и удобство использования сайта для конечных пользователей.

После проведения всех вышеуказанных тестов были получены следующие результаты:

- Функциональность сайта полностью соответствует требованиям и тестовым сценариям.
- Сайт корректно отображается и функционирует на различных веб-браузерах, устройствах и экранах.
- Адаптивный дизайн сайта адекватно реагирует на изменение размеров экрана и различные устройства.
- Производительность сайта соответствует ожиданиям и не вызывает заметных задержек при загрузке страницы.
- Безопасность сайта проверена, уязвимости отсутствуют, защита данных пользователей обеспечена.
- Пользовательский опыт сайта оценен как удовлетворительный, сайт удобен в использовании и навигации.

Выводы: На основе проведенного тестирования веб-сайта https://gcbipdaniil.github.io/ по шаблону из сервиса Figma, можно сделать вывод о том, что сайт соответствует ожиданиям и требованиям, предъявляемым к нему. Веб-сайт успешно прошел тестирование функциональности, совместимости, адаптивности, производительности, безопасности и пользовательского опыта. Однако, рекомендуется продолжить тестирование веб-сайта при различных условиях нагрузки, таких как большое количество одновременных

пользователей, интенсивное использование ресурсов сервера и другие сценарии реальной эксплуатации, чтобы убедиться в его стабильности и надежности.

Также, в дальнейшем тестировании рекомендуется обратить внимание на возможные обновления и изменения веб-сайта, чтобы удостовериться, что они не вносят новых ошибок или проблем.

В результате проведенного тестирования веб-сайта https://gcbipdaniil.github.io/ по шаблону из сервиса Figma было подтверждено, что сайт соответствует требованиям и ожиданиям. Были проведены тесты на функциональность, совместимость, адаптивность, производительность, безопасность и пользовательский опыт. Однако, рекомендуется проводить регулярное тестирование и мониторинг, а также учитывать обратную связь пользователей для дальнейшего совершенствования сайта. Тестирование является важной частью процесса разработки веб-сайтов и позволяет обнаруживать и исправлять ошибки и проблемы, которые могут возникнуть в процессе эксплуатации сайта. Однако, тестирование не является одноразовым процессом, и рекомендуется проводить его на разных этапах разработки, а также после внесения изменений или обновлений на сайте.

Тестирование позволяет выявлять потенциальные проблемы, такие как неправильное отображение элементов на разных устройствах или браузерах, некорректная работа функциональности, низкая производительность сайта, уязвимости безопасности и другие аспекты, которые могут оказать негативное влияние на пользовательский опыт и функциональность сайта.

В данном отчете были описаны основные этапы и подходы к тестированию веб-сайта https://gcbipdaniil.github.io/ по шаблону из сервиса Figma. В процессе тестирования были использованы различные методы, такие как тестирование функциональности, совместимости, адаптивности, производительности и безопасности. Были также представлены результаты тестирования и рекомендации по дальнейшим действиям для обеспечения стабильной и надежной работы сайта.

Тестирование является важным этапом в процессе разработки веб-сайта, поскольку позволяет выявлять и исправлять ошибки и проблемы до их выхода в продакшн, что способствует повышению качества и надежности сайта. Также тестирование позволяет проверить соответствие сайта требованиям и стандартам разработки, а также учесть предпочтения и ожидания пользователей. Однако, тестирование не является единственным средством обеспечения качества сайта, и рекомендуется проводить его в сочетании с другими методами и подходами, такими как код-ревью, автоматизированное тестирование, анализ производительности и безопасности, сбор обратной связи от пользователей и др.

2.5 Заключение

В данном проекте, основанном на верстке сайта по шаблону из сервиса Figma, были выполнены следующие этапы: создание дизайн-макета в Figma, верстка сайта с использованием HTML, CSS и JavaScript, а также тестирование сайта на различных этапах разработки.

В процессе создания проекта была проведена детальная верстка макета с использованием современных технологий и методологий разработки, таких как адаптивный дизайн, оптимизация производительности, доступность и безопасность. Были также реализованы функциональные элементы, такие как интерактивные элементы, анимации и другие элементы дизайна.

После создания сайта был проведен процесс тестирования, который включал проверку функциональности, совместимости с различными браузерами и устройствами, адаптивности на разных экранах, производительности и безопасности. Были использованы различные методы тестирования, включая ручное тестирование, автоматизированное тестирование и анализ кода.

В результате тестирования были выявлены и исправлены ошибки и проблемы, такие как неправильное отображение на определенных устройствах, некорректная работа функциональных элементов, а также недостатки в производительности и безопасности. Были также представлены рекомендации

по дальнейшим действиям, таким как доработка функциональности, оптимизация производительности и обеспечение безопасности сайта.

В заключение, процесс создания, верстки и тестирования данного проекта на основе шаблона из сервиса Figma был выполнен с использованием современных технологий и методологий разработки. Были проведены проверки функциональности, совместимости, адаптивности, производительности и безопасности сайта. Выявленные ошибки и проблемы были исправлены, и представлены рекомендации по дальнейшим действиям для обеспечения стабильной и надежной работы сайта.

2.6 Источники

При создании отчета о процессе тестирования проекта на основе шаблона из сервиса Figma, были использованы следующие источники:

1. Официальная документация Figma (https://www.figma.com/docs/) - официальная документация и руководства по использованию сервиса Figma, которые предоставляют информацию о функциональных возможностях, инструментах и процессе работы с Figma.
2. MDN Web Docs (https://developer.mozilla.org/) - ресурс от Mozilla Developer Network, который содержит обширную документацию и руководства по веб-технологиям, таким как HTML, CSS, JavaScript и другим технологиям, используемым при создании проекта.
3. Stack Overflow (https://stackoverflow.com/) - популярный вопросно-ответный ресурс, где можно найти множество ответов на вопросы, связанные с разработкой веб-приложений, включая вопросы о тестировании и отладке.
4. Блоги и статьи экспертов веб-разработки - различные блоги и статьи на тему веб-разработки, в которых авторы делятся своим

опытом, лучшими практиками и советами по разработке, верстке и тестированию веб-приложений.

5. Документация и руководства по инструментам и технологиям, используемым в проекте - такие как документация по библиотекам и фреймворкам, использованным в проекте, инструкции и руководства по использованию инструментов тестирования, отладки и оптимизации производительности.

Эти источники были использованы для изучения и освоения различных аспектов процесса тестирования проекта на основе шаблона из сервиса Figma и обеспечения качественного и надежного функционирования сайта.