Geekbrains

Разработка личного веб-портфолио с использованием современных технологий веб-разработки и анимации, размещенного на GitHub Pages

[**agarkovanatoliy.github.io/portfolio/**](https://agarkovanatoliy.github.io/portfolio/)

Программа: Разработчик

Специализация: Frontend разработка (React)

ФИО: Агарков Анатолий Владимирович

Донецк

2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc184080857)

[1.1 Актуальность темы 3](#_Toc184080858)

[1.2 Цель и задачи дипломной работы 6](#_Toc184080859)

[1.3 Краткое описание проекта 9](#_Toc184080860)

[2. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ 10](#_Toc184080861)

[2.1 HTML5, CSS3, SaSS 10](#_Toc184080862)

[2.2 JavaScript и ES6+ 12](#_Toc184080863)

[2.3 React: описание и применение 14](#_Toc184080864)

[2.4 Bootstrap: фреймворк для визуализации данных 17](#_Toc184080865)

[2.6 Инструменты разработки и деплоя (Vite, GitHub Pages) 24](#_Toc184080866)

[3. ПРОЕКТИРОВАНИЕ ВЕБ-ПОРТФОЛИО 29](#_Toc184080867)

[3.1 Структура веб-портфолио 29](#_Toc184080868)

[3.2 Дизайн интерфейса и пользовательского опыта 30](#_Toc184080869)

[3.3 Разработка адаптивной верстки 32](#_Toc184080870)

[4. РЕАЛИЗАЦИЯ ПРОЕКТОВ 35](#_Toc184080871)

[4.1 Реализация проекта вёрстки лендинга 35](#_Toc184080872)

[4.2 Реализация Todo list с применением React, TypeScript, Vite 36](#_Toc184080873)

[4.3 Деплой на GitHub Pages 43](#_Toc184080874)

[5. ТЕСТИРОВАНИЕ И АНАЛИЗ ПРОЕКТА 48](#_Toc184080875)

[5.1 Тестирование на различных устройствах и браузерах 48](#_Toc184080876)

[5.2 Анализ полученных результатов 49](#_Toc184080877)

[ЗАКЛЮЧЕНИЕ 51](#_Toc184080878)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 53](#_Toc184080879)

# ВВЕДЕНИЕ

# 1.1 Актуальность темы

Актуальность создания онлайн-портфолио для разработчиков в современном мире трудно переоценить. Это не просто способ показать свои работы, а мощный инструмент для построения успешной карьеры, напрямую влияющий на профессиональный рост и привлекательность на рынке труда. В век цифровых технологий, где работодатели проводят первичный отбор кандидатов именно через онлайн-ресурсы, наличие качественно выполненного веб-портфолио становится фактически обязательным условием для успешного трудоустройства. Его значимость обусловлена несколькими ключевыми факторами:

1. ***Профессиональная идентичность и самопрезентация***

В современном мире профессиональная идентичность и самопрезентация играют ключевую роль в сфере трудоустройства и карьерного роста. Веб-портфолио позволяет эффективно демонстрировать свои навыки и опыт. Статичный PDF-документ или просто список выполненных проектов уже не соответствуют современным требованиям. Интерактивное портфолио предоставляет возможность наглядно продемонстрировать функциональность созданных приложений, веб-сайтов или других разработок. Это может включать в себя не только скриншоты, но и интерактивные демоверсии, позволяющие потенциальному работодателю оценить качество кода, архитектуру проекта и пользовательский интерфейс. Возможность разместить исходный код на GitHub или GitLab и предоставить к нему ссылку значительно повышает доверие и позволяет оценить стиль программирования.

***2) Использование актуальных технологий***

Хорошо продуманное онлайн-портфолио – это эффективный инструмент для самомаркетинга. Оно позволяет представить себя не только как опытного специалиста, но и как креативного человека, способного грамотно презентовать свои достижения. Дизайн портфолио, его структура, удобство навигации – все это формирует первое впечатление о разработчике. Современные платформы, такие как Wix, Squarespace, или же собственные разработки на React, Angular или Vue.js, предоставляют широкие возможности для создания уникального и запоминающегося дизайна, отражающего индивидуальный стиль и профессиональные предпочтения. Включение в портфолио блога с собственными статьями, участием в open-source проектах, или описанием интересного личного опыта значительно усилит впечатление и подчеркнёт профессиональный рост и стремление к развитию. Онлайн-портфолио позволяет создать уникальный имидж или бренд. Можно выразить свою индивидуальность и стиль, что помогает выделиться на фоне конкурентов.

***3) Адаптивность и кроссплатформенность***

Сегодня пользователи все чаще обращаются к интернет-ресурсам с различных устройств (смартфоны, планшеты, ноутбуки и ПК). Поэтому крайне важно, чтобы веб-портфолио было адаптировано под разные экраны и браузеры. Современные подходы к адаптивной верстке и использование CSS-фреймворков, таких как Bootstrap, позволяют сделать сайт доступным для более широкой аудитории и поддерживать кроссплатформенность, что повышает удобство и качество пользовательского опыта.

***4) Динамические и интерактивные элементы***

Использование современных анимационных библиотек (например, Animista) и инструментов, таких как React, позволяет создавать динамические и интерактивные интерфейсы. Это улучшает восприятие информации на странице и делает сайт более живым и привлекательным. Возможность добавления слайдеров, анимаций при прокрутке и переходов между страницами делает навигацию на сайте более плавной и интуитивно понятной. Динамическая подгрузка контента, например, при прокрутке страницы, помогает пользователю сосредоточиться на ключевой информации, загружая дополнительные работы или описания по мере необходимости. Интерактивные элементы призваны удерживать внимание пользователя и побуждать его исследовать ваше портфолио более активно. Сложные понятия, идеи или проекты могут быть изложены более понятно с помощью графиков, анимаций и других визуальных средств.

***5) Простота поддержки и масштабируемость***

Веб-портфолио обеспечивает доступность информации для потенциальных работодателей в любое время и в любом месте. Это существенно расширяет географические границы поиска работы и позволяет обращаться в компании по всему миру. Встраивание социальных сетей, таких как LinkedIn, и добавление контактной информации упрощают процесс связи и позволяют кандидатам легко получать обратную связь от потенциальных работодателей. Использование аналитических инструментов для отслеживания трафика на портфолио позволяет оценить эффективность работы и внести необходимые коррективы в его содержание и дизайн.

***6) SEO-оптимизация и доступность в сети***

С развитием цифрового мира важной частью продвижения личного бренда становится оптимизация сайта для поисковых систем (SEO). Современные технологии веб-разработки предоставляют инструменты для улучшения SEO-показателей (метатеги, оптимизация скорости загрузки и т.д.). Это обеспечивает высокую видимость портфолио в сети, привлекает большее количество посетителей и потенциальных клиентов или работодателей.

***7) Подготовка к профессиональной деятельности***

Создание веб-портфолио с использованием современных технологий демонстрирует способность решать реальные задачи и использовать актуальные инструменты, которые востребованы в отрасли. Это подготавливает разработчика к работе в профессиональной среде и позволяет ему применить полученные знания и навыки в контексте реального проекта.

***8) Поддержка карьерного роста и профессионального развития***

Для карьерного роста крайне важно постоянно развивать свои навыки и быть в курсе новых технологий. Разработка веб-портфолио является отличной возможностью для тестирования новых инструментов и подходов, что способствует профессиональному развитию. Портфолио может быть использовано как платформа для демонстрации профессионального прогресса, а также как хранилище личных достижений и проектов.

***9) Гибкость и персонализация***

Создание качественного веб-портфолио с использованием современных технологий – это инвестиция в собственный профессиональный успех. Это не просто тренд, а необходимый инструмент для любого разработчика, стремящегося к построению успешной карьеры в динамично развивающейся сфере информационных технологий. Вложение времени и ресурсов в создание и постоянное обновление собственного онлайн-портфолио многократно окупится в виде увеличения числа предложений о работе и возможности выбора наиболее подходящего варианта. Более того, хорошее портфолио может стать вашей визитной карточкой и существенно помочь в привлечении клиентов для фриланс-работы. Добавление подробной информации о себе, своего опыта, образовании и достижениях помогает создавать личный контакт с потенциальными клиентами или работодателями. Персонализированный опыт помогает потенциальным клиентам или работодателям лучше понять, как вы можете удовлетворить их потребности и решить их проблемы.

# 1.2 Цель и задачи дипломной работы

**Цель дипломной работы**

Дипломная работа посвящена созданию интерактивного и адаптивного веб-портфолио, представляющего собой современный и эффективный инструмент самопрезентации для демонстрации профессиональных навыков и реализованных проектов. Цель проекта – разработка веб-ресурса, который не только визуально привлекателен и удобен в использовании, но и отвечает всем требованиям современных стандартов веб-дизайна и разработки, обеспечивая при этом лёгкость обновления и управления контентом. В рамках данной работы планируется разработка веб-сайта, адаптированного под различные устройства (десктопы, планшеты, смартфоны) с использованием таких современных технологий, как HTML5, CSS3 и JavaScript.

**Задачи дипломной работы**

Для достижения поставленной цели необходимо решить следующие задачи:

1. Анализ современных технологий веб-разработки и их применение:
   * Исследовать существующие инструменты и технологии для создания веб-портфолио.
   * Определить оптимальный стек технологий, который будет использован для реализации проекта, включая HTML5, CSS3, JavaScript, React и дополнительные библиотеки для анимации и визуализации.
2. Проектирование структуры и интерфейса веб-портфолио:
   * Разработать информационную архитектуру и структуру веб-портфолио.
   * Составить макет пользовательского интерфейса, учитывая требования к адаптивности и доступности контента на различных устройствах.
3. Разработка адаптивного дизайна и верстки:
   * Реализовать верстку страницы с использованием HTML5 и CSS3.
   * Обеспечить адаптивность интерфейса для корректного отображения на разных экранах (десктоп, планшет, мобильные устройства).
4. Создание и интеграция компонентов на основе React:
   * Разработать основные компоненты сайта с использованием библиотеки React.
   * Настроить взаимодействие компонентов и управление состоянием приложения.
5. Использование анимаций и визуальных эффектов:
   * Интегрировать библиотеку Animista для создания анимационных эффектов на главной странице.
   * Реализовать анимации переходов и интерактивных элементов интерфейса.
6. Оптимизация и тестирование веб-портфолио:
   * Выполнить оптимизацию кода и ресурсов для повышения скорости загрузки страниц.
   * Провести тестирование на совместимость с различными браузерами и устройствами, а также выявить и устранить возможные ошибки.
7. Деплой проекта на GitHub Pages:
   * Выполнить конфигурацию проекта для публикации на платформе GitHub Pages.
   * Обеспечить корректное отображение и функциональность сайта после деплоя.
8. Документирование процесса разработки:
   * Описать все этапы создания проекта, включая проектирование, разработку и тестирование.
   * Подготовить документацию, содержащую описание используемых технологий, исходный код и иллюстрации процесса разработки.
9. Оценка и анализ полученных результатов:
   * Проанализировать результаты работы, выявить сильные и слабые стороны проекта.
   * Определить перспективы дальнейшего развития и улучшения веб-портфолио.

Решение этих задач позволит создать качественный продукт, который продемонстрирует не только технические навыки и знания автора, но и его умение применять их на практике для решения реальных задач в области веб-разработки.

Гипотеза состоит в том, что использование таких технологий, как React, TypeScript, Animista и Bootstrap, позволяет достичь высокого уровня функциональности и визуальной привлекательности.

# 1.3 Краткое описание проекта

Данный проект представляет собой личное веб-портфолио разработчика, содержащее несколько ПЭТ (пет-проектов), созданных с использованием различных современных технологий. Основная цель портфолио – демонстрация профессиональных навыков и реализованных проектов, а также представление автора как квалифицированного специалиста в области веб-разработки. Портфолио служит не только средой для представления выполненных работ, но и площадкой для самовыражения и креативного подхода к дизайну и функционалу.

Веб-портфолио разработано с использованием HTML5, CSS3 и JavaScript. Особое внимание в проекте уделено визуальным эффектам и анимациям. Для анимации элементов использовалась библиотека Animista, что позволило сделать интерфейс более живым и интерактивным. Важной частью веб-портфолио является его структура: она организована так, чтобы пользователи могли легко навигацировать между различными разделами и находить интересующие их проекты. Проект содержит несколько ПЭТ-проектов, каждый из которых реализован с применением различных технологий, таких как:

* Лендинг для интернет-магазина: использован React, JavaScript, HTML, SaSS. Проект демонстрирует навыки разработки адаптивного интерфейса и реализации пользовательского взаимодействия, а также интеграции с API.
* Простой To-Do список: разработан на React с применением хуков для управления состоянием и с применением TypeScript.

Проект развернут на платформе GitHub Pages, что обеспечивает удобный доступ к веб-портфолио из любой точки мира. Веб-сайт адаптирован для корректного отображения на различных устройствах и браузерах, что делает его доступным для широкой аудитории.

# 2. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ

# 2.1 HTML5, CSS3, SaSS

**HTML5 (HyperText Markup Language, версия 5)**  
HTML5 – это последняя версия стандарта языка разметки гипертекста, используемого для создания и структурирования контента веб-страниц. HTML5 расширяет возможности предыдущих версий, добавляя новые элементы, атрибуты и API, которые упрощают создание интерактивных и мультимедийных веб-приложений. Основные особенности HTML5 включают:

* **новые семантические элементы** (<header>, <footer>, <article>, <section> и др.), которые помогают лучше структурировать веб-страницы и делают их более доступными для поисковых систем и программ чтения с экрана;
* **интеграция мультимедиа**: поддержка встроенного аудио и видео с помощью тегов <audio> и <video>, что позволяет добавлять мультимедийный контент без использования сторонних плагинов, таких как Flash;
* **API для создания интерактивных приложений**: HTML5 предоставляет ряд API для работы с графикой (Canvas), хранения данных (Local Storage), геолокации (Geolocation API) и других задач.

HTML5 упрощает создание веб-страниц, делает их более структурированными и улучшает производительность за счет нативной поддержки мультимедийных и интерактивных возможностей.

**CSS3 (Cascading Style Sheets, версия 3)**  
CSS3 – это последняя версия каскадных таблиц стилей, предназначенных для управления внешним видом и оформлением HTML-документов. CSS3 включает множество новых возможностей, которые делают создание адаптивных и привлекательных интерфейсов более простым и эффективным:

* **новые селекторы и псевдоклассы**: CSS3 расширяет возможности работы с элементами HTML с помощью селекторов, таких как nth-child, nth-of-type, что позволяет обращаться к элементам по более сложным критериям;
* **градиенты, тени и анимации**: CSS3 добавляет поддержку градиентных заливок, теней для текста и блоков, а также простых анимаций и переходов, что позволяет создавать сложные визуальные эффекты без использования JavaScript;
* **медиазапросы (Media Queries)**: это мощный инструмент для создания адаптивных дизайнов. С помощью медиазапросов можно изменять стили в зависимости от характеристик устройства, таких как ширина экрана, ориентация и разрешение;
* **Flexbox и Grid Layout**: эти системы компоновки обеспечивают гибкое расположение элементов на странице, упрощая создание сложных макетов.

CSS3 позволяет создавать более гибкие и эстетически привлекательные веб-интерфейсы, а также улучшает пользовательский опыт за счет адаптивности и плавных анимаций.

**Sass (Syntactically Awesome Style Sheets)**  
Sass – это один из самых популярных препроцессоров CSS, предназначенный для упрощения и улучшения процесса написания стилей. Он добавляет в CSS функциональные возможности, такие как переменные, вложенность, миксины и функции, которые упрощают управление стилями и делают их более структурированными и легко поддерживаемыми:

* **переменные** позволяют хранить значения стилей, такие как цвета, размеры шрифтов и отступы, в одном месте, что упрощает их повторное использование и изменение;
* **вложенность** дает возможность писать стили в контексте родительских элементов, что улучшает читаемость кода и соответствует структуре HTML;
* **миксины** – это повторно используемые блоки кода, которые можно вызывать в разных частях стилей. Это позволяет сократить дублирование и упростить создание адаптивных компонентов.
* **импорт и частичные файлы**: Sass поддерживает разбиение стилей на модули и их последующее объединение, что упрощает структуру проекта и управление кодом.

Sass компилируется в обычный CSS, что позволяет использовать все преимущества препроцессора без потери совместимости с браузерами

# 2.2 JavaScript и ES6+

JavaScript – это высокоуровневый, интерпретируемый язык программирования, который используется для создания интерактивных и динамических элементов на веб-страницах. Язык поддерживается всеми современными веб-браузерами и является неотъемлемой частью стека веб-технологий наряду с HTML и CSS. Основная задача JavaScript – взаимодействие с пользователем и изменение содержимого веб-страниц без перезагрузки, что делает его важным инструментом для разработки интерфейсов, веб-приложений и игр.  
JavaScript поддерживает как императивный, так и функциональный стили программирования и имеет множество встроенных возможностей, таких как:

* манипуляция DOM (Document Object Model): JavaScript позволяет изменять структуру, стиль и содержимое HTML-страниц, что делает его идеальным для создания динамических веб-приложений;
* обработка событий: позволяет реагировать на действия пользователя (нажатия кнопок, ввод текста, движения мыши и т.д.) и выполнять определенные действия;
* асинхронное программирование: с помощью таймеров, коллбеков и Promises JavaScript поддерживает выполнение асинхронных операций, таких как запросы к серверу и обработка ответов;
* поддержка модулей и пакетов: JavaScript может использоваться в связке с модулями и пакетами, например, при работе с Node.js для серверной разработки или при использовании модульных систем в клиентской разработке;

ES6+ (ECMAScript 6 и последующие версии) представляет собой стандарт языка программирования JavaScript, который был принят в 2015 году и значительно расширил возможности языка. С тех пор были выпущены несколько обновлений (например, ES7, ES8 и так далее), каждое из которых добавляло новые функции и улучшения.

ES6 (ECMAScript 2015) и последующие версии стандарта (ES7, ES8, ES9 и т.д.) внесли значительные улучшения и новые функции в JavaScript, расширяя его функциональные возможности и повышая удобство использования. Эти нововведения существенно упростили разработку и сделали код более структурированным и читаемым. Основные особенности ES6+ включают:

* классы и наследование: ES6 добавил синтаксический сахар для работы с классами и наследованием, что делает объектно-ориентированное программирование более интуитивным;
* современные стандарты ECMAScript (ES6 и выше) значительно улучшили возможности JavaScript, сделав его мощным и удобным инструментом как для фронтенд-, так и для бэкенд-разработки;
* переменные с блоковой областью видимости (let, const): позволяют создавать переменные, видимые только в пределах текущего блока, что предотвращает ошибки, связанные с глобальной областью видимости;
* стрелочные функции (=>): упрощают синтаксис функций и сохраняют контекст this текущего объекта, что делает работу с анонимными функциями более понятной;
* шаблонные строки: позволяют вставлять выражения и переменные непосредственно в строки с помощью ${expression}, упрощая работу с текстом и HTML-разметкой;
* деструктуризация: позволяет извлекать значения из массивов и объектов и присваивать их переменным в удобной форме;
* модули (import, export): упрощают структуру проекта и позволяют разделять код на модули, улучшая его повторное использование и поддержку;
* промисы (Promises) и async/await: упрощают асинхронное программирование, делая код более линейным и легким для восприятия. Промисы используются для обработки асинхронных операций, таких как запросы к серверу, а async/await упрощают синтаксис работы с промисами.

# 2.3 React: описание и применение

**React –** это библиотека JavaScript с открытым исходным кодом, разработанная компанией Facebook для создания пользовательских интерфейсов. React широко используется для разработки одностраничных веб-приложений (Single Page Applications, SPA), где интерфейс динамически изменяется без перезагрузки страницы. Главной особенностью React является использование компоненто-ориентированного подхода, который упрощает разработку, тестирование и поддержку веб-приложений.

**Основные характеристики и возможности React:**

1. Компоненты: React основан на использовании компонентов – автономных и многократно используемых частей интерфейса. Каждый компонент представляет собой отдельный блок, который может включать в себя как визуальную часть (HTML-разметка), так и логику (JavaScript-функции и состояния). Компоненты могут быть функциональными (основанными на функциях) или классовыми (основанными на классах).
2. JSX (JavaScript XML): JSX – это расширение синтаксиса JavaScript, которое позволяет писать разметку (похожие на HTML конструкции) прямо в коде JavaScript. Благодаря JSX разработчики могут легко описывать структуру интерфейса, используя привычный HTML-подобный синтаксис, и при этом использовать мощь JavaScript.
3. Однонаправленный поток данных (props): В React данные передаются от родительских компонентов к дочерним через свойства (props). Это обеспечивает четкую и предсказуемую структуру передачи данных, упрощая отладку и управление состоянием приложения.
4. Состояние компонента (state): React позволяет управлять состоянием компонентов с помощью useState в функциональных компонентах и свойства state в классовых компонентах. Состояние – это объект, который хранит данные, влияющие на внешний вид и поведение компонента. При изменении состояния React автоматически перерисовывает компонент, обеспечивая синхронизацию данных и интерфейса.
5. Виртуальный DOM (Virtual DOM): React использует концепцию виртуального DOM для повышения производительности. Виртуальный DOM – это облегченная копия реального DOM, которая хранится в памяти. При изменении состояния компонента React обновляет виртуальный DOM и вычисляет разницу с предыдущим состоянием, а затем минимально обновляет реальный DOM, что значительно ускоряет рендеринг интерфейсов.
6. Роутинг и навигация: Для создания многостраничных приложений в React используется библиотека React Router, которая позволяет реализовать навигацию по различным маршрутам (URL) без перезагрузки страницы. Это особенно полезно для одностраничных приложений, где разные компоненты отображаются в зависимости от текущего URL.
7. Контекст и глобальное состояние: React предоставляет возможность создавать глобальные состояния с помощью Context API или сторонних библиотек, таких как Redux или MobX. Это позволяет управлять состоянием на уровне всего приложения и передавать данные между компонентами, которые не связаны между собой напрямую.
8. React Hooks: Начиная с версии 16.8, React представил хуки (Hooks) – функции, которые позволяют использовать состояние и другие возможности React в функциональных компонентах. Основные хуки включают:
   * useState – для управления локальным состоянием.
   * useEffect – для выполнения побочных эффектов (например, загрузка данных).
   * useContext – для доступа к данным контекста.

Хуки делают код компонентов более лаконичным и гибким, избавляя от необходимости использования классовых компонентов.

**Применение React:**

React используется для создания интерактивных пользовательских интерфейсов с богатым функционалом. Он подходит для разработки различных типов веб-приложений, таких как:

* Одностраничные приложения (Single Page Applications, SPA): React позволяет создавать приложения, в которых пользователь взаимодействует с одним HTML-документом, а изменение контента происходит без перезагрузки страницы. Это обеспечивает более быстрый отклик и плавный пользовательский опыт.
* Административные панели и информационные панели (Dashboards): React часто используется для создания интерактивных и динамических интерфейсов админ-панелей, где важна быстрая реакция на изменения данных.
* Веб-сервисы и социальные сети: React применяют для создания интерфейсов с интенсивной работой с данными, например, в социальных сетях или сервисах для работы с контентом, таких как Facebook или Instagram.
* E-commerce приложения: React активно используется для разработки интернет-магазинов, где важна быстрая работа с каталогами товаров, корзинами и системой оформления заказов.
* Мобильные приложения: С помощью React Native разработчики могут создавать мобильные приложения на основе React, что позволяет использовать единый стек технологий для веб- и мобильной разработки.

React – мощная и гибкая библиотека, которая позволяет создавать современные, производительные и масштабируемые веб-приложения, обеспечивая комфортную разработку и удобное управление состоянием интерфейса.

# 2.4 Bootstrap: фреймворк для визуализации данных

**Bootstrap –** это популярный фронтенд-фреймворк с открытым исходным кодом, предназначенный для быстрой и удобной разработки адаптивных и кросс-браузерных веб-сайтов и веб-приложений. Он был разработан командой Twitter и впервые выпущен в 2011 году. Bootstrap предлагает широкий набор готовых компонентов, таких как кнопки, формы, таблицы, модальные окна, навигационные панели и многое другое, что позволяет существенно ускорить процесс разработки интерфейсов.

Основное преимущество Bootstrap заключается в том, что он предоставляет разработчикам удобный и понятный инструментарий для создания стилей и компонентов интерфейса с минимальными усилиями и без необходимости написания большого количества CSS-кода. Используя готовые стили и классы Bootstrap, можно легко построить современный и визуально привлекательный интерфейс.

**Основные особенности Bootstrap:**

1. Сетка (Grid System): Bootstrap предоставляет адаптивную систему сеток (grid system), которая позволяет создавать гибкие и адаптивные макеты. Сетка основана на flexbox, что обеспечивает поддержку различных типов макетов, включая:
   * Контейнеры (containers): основа для выравнивания содержимого на странице.
   * Ряды (rows): строки, которые служат для размещения столбцов.
   * Столбцы (columns): отдельные блоки, которые можно объединять или делить на ячейки.

Сетка Bootstrap поддерживает различные размеры экранов (sm, md, lg, xl, xxl), что позволяет создать адаптивные интерфейсы, автоматически подстраивающиеся под размер устройства (мобильные телефоны, планшеты, десктопы и т.д.).

1. Компоненты: Bootstrap включает более 20 готовых компонентов, которые помогают строить интерфейсы с минимальными усилиями:
   * Кнопки (btn): кнопки различных стилей, размеров и с эффектами наведения.
   * Формы (form-group): элементы для создания форм, включая поля ввода, флажки, радио-кнопки и селекторы.
   * Навигация (navbar, nav): компоненты для создания навигационных панелей, меню и хлебных крошек.
   * Карточки (card): для отображения содержимого в структурированных блоках.
   * Модальные окна (modal): всплывающие окна для отображения диалогов или дополнительных сведений.

Эти компоненты можно легко модифицировать с помощью предустановленных классов и атрибутов, а также комбинировать друг с другом для создания сложных интерфейсов.

1. Тема и стилизация: Bootstrap включает готовые стили для типографики, кнопок, форм и других элементов интерфейса, что позволяет добиться единого и современного внешнего вида приложения без дополнительных усилий. Все стили можно переопределить с помощью CSS или подключить собственные темы.
2. Интерактивные компоненты и JavaScript плагины: В состав Bootstrap входят JavaScript-плагины, основанные на jQuery, которые добавляют интерактивность компонентам. Например:
   * Модальные окна (modal): всплывающие окна для отображения дополнительной информации.
   * Слайдеры и карусели (carousel): автоматическая прокрутка изображений или контента.
   * Выпадающие меню (dropdown): интерактивные списки и меню.
   * Всплывающие подсказки и popovers: информационные всплывающие элементы.
3. Адаптивность и кросс-браузерность: Bootstrap изначально разработан для создания адаптивных макетов, поддерживающих отображение на экранах различных размеров. Адаптивность достигается за счет медиа-запросов и гибкой сетки, что обеспечивает корректное отображение интерфейса как на мобильных устройствах, так и на больших экранах.
4. Иконки: Начиная с Bootstrap 5, фреймворк поддерживает набор иконок Bootstrap Icons, которые можно использовать в веб-приложениях для улучшения визуального восприятия и повышения читаемости интерфейсов.

**Применение Bootstrap:**

Bootstrap используется в самых разных проектах – от небольших лендингов и блогов до крупных корпоративных порталов и e-commerce сайтов. Благодаря своей простоте и гибкости, Bootstrap стал одним из наиболее распространенных фреймворков для разработки UI, что делает его идеальным выбором для создания прототипов и MVP (Minimum Viable Product).

**Преимущества Bootstrap:**

* Быстрый старт: для использования Bootstrap достаточно подключить CSS и JavaScript файлы фреймворка и сразу приступить к разработке интерфейса.
* Адаптивность: встроенная поддержка адаптивных макетов упрощает создание кросс-устройственных интерфейсов.
* Сообщество и документация: обширная документация и большое сообщество разработчиков предоставляют множество готовых решений и примеров кода.
* Поддержка тем и кастомизация: разработчики могут легко менять стили компонентов или создавать свои темы с помощью переменных Sass.

Bootstrap – это мощный и гибкий фреймворк, который упрощает процесс создания современных веб-приложений, обеспечивая адаптивность, кросс-браузерность и интуитивно понятную стилизацию интерфейсов. Использование Bootstrap позволяет сосредоточиться на функциональности приложения, а не на разработке базовых стилей и компонентов, что существенно ускоряет разработку и повышает качество интерфейсов.

**2.5 Animista: библиотека для создания анимаций**

**Animista** – это библиотека анимаций CSS, которая предоставляет разработчикам готовые и настраиваемые анимации для веб-проектов. Animista – это не фреймворк, а скорее интерактивный инструмент, с помощью которого можно создавать и настраивать CSS-анимации, а затем экспортировать их в свой проект. Сайт Animista содержит большой каталог анимаций, которые можно визуально настроить и протестировать перед интеграцией в проект.

Основная цель использования Animista – упрощение процесса добавления анимаций в веб-приложения и экономия времени на их разработку. С помощью этой библиотеки разработчики могут создавать анимации для различных элементов интерфейса, таких как кнопки, блоки, изображения и текст, без необходимости писать сложный CSS-код.

**Основные характеристики и возможности Animista:**

1. Большой каталог анимаций: Animista предлагает широкий выбор готовых CSS-анимаций, которые можно применять к различным элементам веб-страницы. В библиотеке присутствуют анимации для появления и исчезновения элементов, масштабирования, вращения, перемещения, изменения цвета и прозрачности и многое другое. Это делает Animista удобным инструментом для добавления анимаций в интерфейс.
2. Настройка анимаций: На сайте Animista (animista.net) разработчики могут визуально настроить параметры анимации, такие как:
   * Длительность (duration): продолжительность анимации.
   * Задержка (delay): время перед началом анимации.
   * Функция временной зависимости (easing): изменение скорости анимации (линейное, ease-in, ease-out и т.д.).
   * Количество повторений (iteration count): сколько раз повторяется анимация.
   * Направление (direction): определяет направление анимации (обычное, обратное или чередующееся).
   * Заполнение (fill mode): как элемент ведет себя до начала или после завершения анимации.

Все эти настройки позволяют кастомизировать анимации под нужды конкретного проекта.

1. Легкая интеграция: После настройки анимации на сайте Animista, разработчики могут сгенерировать CSS-код и скопировать его в свой проект. Это упрощает интеграцию анимаций и позволяет быстро вносить изменения без необходимости изучения документации или написания анимаций с нуля.
2. Удобный интерфейс: Animista предлагает удобный визуальный интерфейс, где можно в режиме реального времени видеть результат изменения настроек анимации. Это позволяет тестировать и подбирать наиболее подходящие эффекты для элементов интерфейса.
3. Поддержка современных браузеров: Анимации, созданные с помощью Animista, основаны на стандартах CSS3 и поддерживаются всеми современными браузерами. Это обеспечивает совместимость и кросс-браузерную поддержку без необходимости использования дополнительных библиотек или плагинов.
4. Группы анимаций: Animista организует анимации по категориям, что упрощает их поиск и выбор. Категории включают:
   * Появление и исчезновение (entrances, exits).
   * Перемещение (sliding, bouncing).
   * Сдвиг и вращение (flipping, spinning).
   * Изменение размера (scaling).
   * Искажение и растягивание (stretching, wobbling).
   * Всплывание и затемнение (fading, blurring).
   * Эффекты текста (text-shadow, letter-spacing).
5. Микроанимации и взаимодействие: Animista поддерживает микроанимации, которые добавляют интерактивности элементам интерфейса. Микроанимации — это небольшие визуальные эффекты, которые активируются при взаимодействии пользователя с элементами (например, при наведении курсора на кнопку или при нажатии на ссылку).

**Применение Animista:**

Animista применяется для создания динамичных и привлекательных пользовательских интерфейсов. Она идеально подходит для добавления анимаций в следующие случаи:

* Анимации при загрузке: отображение прогресса загрузки или плавное появление контента при загрузке страницы.
* Анимации при наведении: эффекты, которые применяются при наведении курсора на элементы (например, увеличение кнопки или изменение цвета).
* Переходы и трансформации: анимации, которые визуально сопровождают переходы между различными состояниями или страницами.
* Обратная связь: небольшие анимации для указания на успешное выполнение действия (например, анимация при добавлении товара в корзину).
* Эффекты для текста: плавное появление, увеличение или перемещение текста для привлечения внимания.

**Преимущества использования Animista:**

* Простота использования: визуальный интерфейс и генерация кода упрощают процесс создания и интеграции анимаций.
* Экономия времени: разработчикам не нужно вручную прописывать анимации или искать готовые решения, что сокращает время на разработку.
* Гибкость: возможность настройки параметров анимации под конкретные нужды проекта.
* Совместимость: созданные анимации поддерживаются современными браузерами, что гарантирует стабильную работу на всех платформах.

**Animista** – это мощный инструмент для создания анимаций, который помогает разработчикам создавать визуально привлекательные и динамичные интерфейсы с минимальными усилиями. Использование Animista в проектах позволяет оживить элементы интерфейса, улучшить пользовательский опыт и сделать веб-приложение более интерактивным. Использование Animista в проектах позволяет оживить элементы интерфейса, улучшить пользовательский опыт и сделать веб-приложение более интерактивным. Анимации, созданные с помощью данного инструмента, могут привлечь внимание пользователей, повысить их вовлеченность и сделать взаимодействие с приложением более естественным. Благодаря широкому выбору настроек и параметров, разработчики могут легко адаптировать анимации под конкретные нужды и стиль своего проекта, что в свою очередь создает уникальный визуальный опыт.

Как результат, Animista становится незаменимым инструментом для тех, кто хочет добавить акценты и оживить интерфейсы своих приложений, не жертвуя при этом производительностью и стилем.

# 2.6 Инструменты разработки и деплоя (Vite, GitHub Pages)

Современная веб-разработка требует использования инструментов, которые обеспечивают удобную среду разработки и простой деплой проектов. В этой секции мы рассмотрим два таких инструмента: Vite и GitHub Pages. Оба инструмента играют важную роль в процессе создания и развертывания веб-приложений, предоставляя разработчикам мощные возможности для быстрой разработки, сборки и деплоя проектов.

**Vite**

Это инструмент для разработки и сборки веб-приложений, который разработан на базе ESBuild и использует современные возможности браузеров для создания быстрой и интерактивной среды разработки. Он был создан как альтернатива Webpack и Parcel и призван устранить некоторые ограничения и недостатки существующих инструментов. Vite особенно хорошо подходит для работы с приложениями, написанными на Vue.js, React или Svelte, и имеет интуитивно понятную настройку.

**Основные особенности Vite:**

1.Мгновенная загрузка: Vite использует ES-модули для загрузки файлов и делает это непосредственно в браузере без необходимости предварительной сборки проекта. Это позволяет запустить сервер разработки почти мгновенно, даже для больших проектов.

2. Горячая перезагрузка модулей (HMR): Благодаря HMR (Hot Module Replacement) изменения в коде применяются на лету, без полной перезагрузки страницы. Это ускоряет процесс разработки, поскольку разработчики сразу видят результат внесенных изменений в интерфейсе.

3.Быстрая сборка: Vite использует ESBuild на этапе предварительной сборки и Rollup для окончательной сборки проекта. Этот подход позволяет значительно сократить время сборки, особенно по сравнению с традиционными инструментами, такими как Webpack.

4.Поддержка современных стандартов: Vite поддерживает новейшие стандарты JavaScript и CSS, такие как ES6+, CSS-модули и динамический импорт. Это упрощает работу с современными библиотеками и фреймворками, а также позволяет использовать новейшие возможности языка.

5. Расширяемость: Vite поддерживает множество плагинов, которые можно использовать для интеграции дополнительных функций, таких как TypeScript, PWA, интеграция с фреймворками (например, Vue или React), и многое другое.

6.Поддержка нескольких фреймворков: Vite не привязан к конкретному фреймворку и может быть использован с различными библиотеками, такими как Vue, React, Preact, Svelte, и даже с обычными HTML/CSS/JS-проектами.

Применение Vite:

Vite чаще всего используется для создания интерфейсов и клиентской части веб-приложений, где требуется высокая скорость разработки и низкое время отклика. Он особенно полезен для проектов, где важно быстро видеть изменения в коде без долгого ожидания перезагрузки и повторной сборки проекта.

**2.7 GitHub Pages: Платформа для хостинга статических сайтов**

GitHub Pages – это бесплатный сервис для хостинга статических сайтов, предоставляемый платформой GitHub. Он позволяет разработчикам размещать свои веб-проекты (например, портфолио, документацию или блоги) непосредственно из репозитория GitHub. GitHub Pages поддерживает как пользовательские сайты (User Pages), так и сайты для отдельных проектов (Project Pages).

Основные особенности GitHub Pages:

1.Простота развертывания: Развернуть сайт на GitHub Pages можно буквально в несколько кликов. Достаточно создать репозиторий и активировать GitHub Pages в настройках проекта. После этого все изменения, внесенные в основную ветку репозитория, автоматически публикуются на сайте.

2.Автоматическое развертывание: GitHub Pages может автоматически развернуть сайт при каждом пуше изменений в репозиторий. Это особенно удобно для проектов, которые активно развиваются и требуют частых обновлений.

3.Поддержка кастомных доменов: GitHub Pages позволяет привязать собственное доменное имя к проекту. Это полезно для личных сайтов или корпоративных веб-приложений, которые требуют собственного брендинга.

4.Интеграция с Jekyll: GitHub Pages поддерживает генератор статических сайтов Jekyll, что позволяет создавать динамические сайты на базе статических файлов. Jekyll предоставляет шаблоны, плагины и возможности обработки Markdown-файлов, что упрощает создание и ведение блога.

5. Поддержка HTTPS: Все сайты, размещенные на GitHub Pages, автоматически получают сертификат SSL, что обеспечивает безопасное соединение через HTTPS.

Применение GitHub Pages:

GitHub Pages используется для развертывания статических сайтов, которые не требуют серверной логики (например, API или базы данных). Это идеальный выбор для небольших портфолио, сайтов документации или страниц проектов с открытым исходным кодом. Также GitHub Pages подходит для размещения учебных проектов, демонстраций или MVP (Minimum Viable Product).

Vite и GitHub Pages – это мощные и удобные инструменты для разработки и деплоя веб-проектов. Vite обеспечивает быструю и интерактивную среду разработки, в то время как GitHub Pages предоставляет простой и бесплатный способ развертывания статических сайтов. Вместе эти инструменты позволяют разработчикам создавать и размещать свои проекты с минимальными усилиями, что делает их отличным выбором как для небольших пет-проектов, так и для более серьезных приложений.

**2.8 TypeScript**

TypeScript – это строго типизированный язык программирования, который является надстройкой над JavaScript. Он был разработан Microsoft и выпущен в 2012 году. Основная цель TypeScript – улучшить разработку больших и сложных приложений, предоставляя разработчикам инструменты для обнаружения ошибок на этапе компиляции, а не во время выполнения.

**Ключевые особенности TypeScript**

* Статическая типизация: TypeScript позволяет разработчикам явно задавать типы переменных, функций и объектов. Это помогает избежать ошибок, связанных с неправильным использованием типов данных, и облегчает чтение и поддержку кода.
* Поддержка современных возможностей JavaScript: TypeScript поддерживает все возможности современного JavaScript (ES6 и выше), включая стрелочные функции, классы, модули и асинхронные функции. Это позволяет разработчикам использовать последние достижения языка без необходимости беспокоиться о совместимости с различными средами выполнения.
* Компиляция в JavaScript: Код на TypeScript компилируется в чистый JavaScript, который можно выполнять в любом браузере или среде выполнения, поддерживающей JavaScript. Это делает TypeScript совместимым со всеми существующими библиотеками и фреймворками JavaScript.
* Инструменты для разработки: TypeScript предлагает мощные инструменты для разработки, такие как автозаполнение, рефакторинг кода и поддержка IDE. Это значительно повышает продуктивность разработчиков.
* Интерфейсы и типы: TypeScript позволяет создавать интерфейсы и определять типы для объектов, что улучшает организацию кода и делает его более понятным. Разработчики могут создавать собственные типы и использовать их в различных частях приложения.

**Преимущества использования TypeScript**

* Улучшенная поддержка кода: Благодаря статической типизации и более строгим правилам TypeScript позволяет разработчикам легче обнаруживать ошибки и упростить понимание кода. Это особенно полезно в больших командах и проектах.
* Лучшая интеграция с редакторами: TypeScript предлагает улучшенное взаимодействие с редакторами и IDE, что делает разработку более эффективной за счет возможности автозаполнения и подсказок.
* Поддержка масштабируемости: TypeScript подходит для разработки больших и сложных приложений, обеспечивая более структурированный подход к написанию кода. Это позволяет легче управлять изменениями и улучшениями в приложении.

**Применение TypeScript**

TypeScript широко используется в разработке веб-приложений, особенно в сочетании с популярными фреймворками, такими как Angular, React и Vue.js. Многие компании и разработчики выбирают TypeScript для создания надежных и масштабируемых приложений благодаря его возможностям и преимуществам.

* Фреймворки:
  + Angular: TypeScript является основным языком разработки для Angular, что позволяет создавать мощные и эффективные веб-приложения.
  + React: TypeScript можно использовать в проектах React, чтобы добавить типизацию и улучшить поддержку кода.
  + Vue.js: Vue также поддерживает TypeScript, что позволяет разработчикам использовать типизацию в своих приложениях.
* Серверная разработка: TypeScript также может использоваться для разработки серверных приложений с помощью Node.js, что позволяет создавать более надежные и масштабируемые серверные решения.

TypeScript – это мощный инструмент для разработчиков, стремящихся создать качественные и масштабируемые приложения. Благодаря статической типизации, поддержке современных возможностей JavaScript и интеграции с популярными фреймворками, TypeScript становится все более популярным выбором среди разработчиков по всему миру. Он предлагает богатый набор возможностей, которые делают процесс разработки более организованным и продуктивным, что позволяет командам легче управлять сложными проектами.

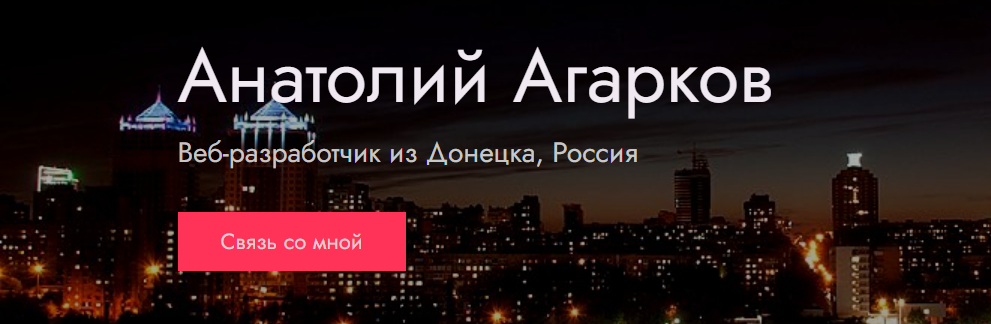
# 3. ПРОЕКТИРОВАНИЕ ВЕБ-ПОРТФОЛИО

# 3.1 Структура веб-портфолио

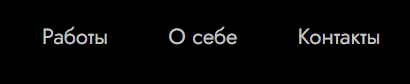
Портфолио представляет собой статический сайт, содержащий несколько секций, которые демонстрируют навыки, проекты и достижения автора. В нем используется современный дизайн и технологии для создания интерактивного пользовательского интерфейса и привлекательного визуального оформления.

**Основная структура веб-портфолио**

1. Главная страница:
   * Заголовок и приветственное сообщение: На главной странице отображается имя автора, а также краткий слоган. Это помогает пользователю быстро понять специализацию автора.



* + Навигационное меню: Навигация расположена в верхней части страницы и позволяет пользователю быстро переходить между различными разделами портфолио (например, «Работы», «О себе», «Контакты»).



1. О себе:
   * Секция содержит информацию о профессиональном опыте и специализации автора. Здесь представлено краткое описание личности, подхода к работе и интересов.
2. Работы:
   * Раздел демонстрирует примеры выполненных проектов, включая пет-проекты и учебные работы.
   * Каждый проект представлен карточкой с коротким демонстрационным видео или фото, названием, кратким описанием и ссылкой на развернутую версию проекта.
3. Контакты:
   * Секция содержит информацию для связи: электронная почта, ссылки на социальные сети (GitHub, Telegram) и контактная форма, позволяющая отправить сообщение напрямую.

Данное веб-портфолио имеет четко структурированную и логически организованную архитектуру. Каждая секция сайта поддерживает цельность дизайна и предоставляет посетителям всю необходимую информацию о навыках и проектах автора. Визуальная привлекательность и анимации делают сайт запоминающимся и удобным для пользователя, а использование современных технологий (React, Bootstrap, Animista, Vite) подчеркивает компетенции автора в области фронтенд-разработки.

# 3.2 Дизайн интерфейса и пользовательского опыта

Веб-портфолио построено на основе современных принципов UI/UX дизайна. Дизайн интерфейса (UI) и пользовательский опыт (UX) ориентированы на удобство восприятия информации, легкость навигации и приятное визуальное оформление. Ниже описаны ключевые аспекты дизайна и UX-подходы, использованные в этом проекте.

**1. Концепция дизайна (UI Design)**

* Минимализм: Дизайн сайта выполнен в минималистичном стиле, без лишних элементов и нагромождений. Это помогает пользователю сосредоточиться на содержимом – навыках и проектах, представленных в портфолио.
* Цветовая палитра: Используется ограниченная цветовая палитра, сочетающая нейтральные и акцентные цвета. Черный цвет фона интернет-портфолио подчеркивает элегантность и современность дизайна, создавая контраст с яркими элементами и изображениями, которые привлекают внимание пользователя. Такой выбор фона позволяет выделить ключевые работы и проекты, делая их более заметными и акцентированными. Кроме того, черный фон часто ассоциируется с экспертностью и профессионализмом, что может произвести положительное впечатление на потенциальных клиентов или работодателей. Этот цвет также создает атмосферу глубины и позволяет использовать светлые и яркие цветовые акценты, обеспечивая гармоничное и сбалансированное восприятие контента. Пользователи, просматривающие портфолио, могут легче сосредоточиться на представленных работах, так как черный фон помогает минимизировать отвлекающие факторы. В результате, такое оформление не только привлекает внимание, но и способствует созданию запоминающегося и визуально привлекательного впечатления о дизайнере или художнике. Кроме того, использование черного фона позволяет более смело экспериментировать с типографикой и графическими элементами, что делает портфолио уникальным и креативным. В целом, черный цвет фона является стратегическим выбором, способствующим созданию визуальной идентичности и выделению работ на фоне других портфолио.
* Типографика: Шрифты подобраны таким образом, чтобы они были легко читаемы. Крупные заголовки используются для обозначения секций, а более мелкий текст – для подробной информации. Это создает визуальную иерархию, которая помогает пользователю ориентироваться на странице.
* Иконки и визуальные элементы: Иконки используются для обозначения навыков, инструментов и социальных сетей. Они упрощают восприятие информации, делая сайт более визуально привлекательным и наглядным.

**2. Пользовательский опыт (UX Design)**

* Простота и ясность: Весь интерфейс логично разделен на секции: «Проекты», «О себе», «Контакты». Это позволяет пользователю легко перемещаться между разделами и находить нужную информацию. Каждая секция имеет свое содержание, которое представлено в сжатой и структурированной форме.
* Адаптивность и отзывчивость: Дизайн сайта полностью адаптивен, что позволяет ему корректно отображаться на любых устройствах – от настольных компьютеров до смартфонов. Для адаптивности используются flexbox и grid-системы, что позволяет элементам автоматически менять свое расположение в зависимости от разрешения экрана.
* Интерактивные элементы: Кнопки, ссылки и карточки проектов анимированы и реагируют на действия пользователя (например, изменение цвета или тени при наведении), что повышает ощущение интерактивности и отзывчивости интерфейса.
* Оптимизация контента: Сайт загружается быстро и обеспечивает плавный скроллинг, что достигается благодаря использованию легковесных фреймворков и инструментов сборки, таких как Vite. Оптимизация скорости загрузки страниц и кода повышает удовлетворенность пользователей и снижает показатель отказов.

Дизайн интерфейса и пользовательского опыта данного веб-портфолио построен на основе современных подходов к разработке интерфейсов. Минимализм, четкая структура и интуитивная навигация делают сайт легким в использовании, а современные технологии и интерактивные элементы добавляют привлекательности и подчеркивают профессионализм автора.

# 3.3 Разработка адаптивной верстки

Адаптивная верстка на сайте реализована через использование современных технологий и подходов, которые обеспечивают корректное отображение контента на различных устройствах. Ниже описаны ключевые элементы, благодаря которым достигается адаптивность этого веб-портфолио.

**1. Медийные запросы (Media Queries)**

* Сайт использует медийные запросы для изменения стилей в зависимости от ширины экрана устройства. Это позволяет применять различные CSS-правила, чтобы адаптировать элементы интерфейса.
* Например, шрифты, отступы, размеры блоков и расположение элементов могут изменяться в зависимости от размера экрана. Это гарантирует, что текст остаётся читаемым, а элементы не перекрываются.

**2. Гибкие сетки (Flexible Grids)**

* Портфолио использует гибкую сеточную систему, что позволяет элементам корректно располагаться на странице и менять свои размеры в зависимости от доступного пространства.
* Эта система включает в себя использование CSS Flexbox и Grid для выстраивания элементов в строки и колонки, обеспечивая гибкость и динамичность верстки.

**3. Bootstrap**

* Использование фреймворка Bootstrap значительно упрощает процесс создания адаптивного дизайна. Bootstrap предлагает готовые компоненты и классы, которые автоматически подстраиваются под разные размеры экранов.
* Элементы, такие как навигационное меню, кнопки и карточки, обладают свойствами адаптивности и изменяют свое отображение в зависимости от устройства.
* На больших экранах меню отображается в горизонтальном формате, что позволяет легко находить нужные разделы.

**4. Изображения и медиа-контент**

Изображения на сайте настроены таким образом, чтобы автоматически масштабироваться в зависимости от ширины экрана. Это достигается за счет использования свойств CSS, таких как max-width: 100%;, что позволяет изображениям адаптироваться к размерам контейнера.

* Это предотвращает искажение изображений и сохраняет их качество на всех устройствах.

**5. Адаптивные шрифты**

* Шрифты на сайте также подстраиваются под размер экрана с помощью медийных запросов. Например, заголовки могут иметь разные размеры на мобильных и настольных устройствах для обеспечения удобочитаемости.

**6. Тестирование на разных устройствах**

* Портфолио тестировалось на различных устройствах и браузерах для обеспечения кроссбраузерной совместимости и корректного отображения интерфейса.
* Инструменты разработчика в браузерах используются для эмуляции мобильных устройств и проверки адаптивности.

Адаптивная верстка сайта [github.com/agarkovanatoliy/portfolio](https://github.com/agarkovanatoliy/portfolio) успешно реализована с использованием медийных запросов, гибких сеток и других современных технологий. Это обеспечивает удобное взаимодействие пользователей с контентом на устройствах различного размера, гарантируя, что информация всегда доступна и легко воспринимается.

# 4. РЕАЛИЗАЦИЯ ПРОЕКТОВ

# 4.1 Реализация проекта вёрстки лендинга

Проект [github.com/agarkovanatoliy/portfolio](https://github.com/agarkovanatoliy/portfolio). Макет этого проекта взят из Figma, включает в себя основные элементы, такие как заголовки, изображения, кнопки и блоки с информацией, которые были точно воспроизведены в коде. Figma позволяет легко работать с цветами, шрифтами и отступами, что способствует точности в реализации. Каждый из этих элементов был тщательно воспроизведён в коде, что позволило достичь высокой степени соответствия оригинальному дизайну. Использование Figma значительно облегчило процесс работы с цветовыми палитрами, типографикой и отступами, что в итоге способствовало повышению точности реализации каждого компонента.

HTML-код организован с использованием семантических тегов, таких как <header>, <main>, <section>, и <footer>, что улучшает доступность и SEO-оптимизацию страницы.

Включена навигационная панель, позволяющая пользователям быстро перемещаться между различными разделами лендинга.

Стили оформлены с помощью Sass, что позволяет задать цвета, шрифты и отступы в соответствии с дизайном в Figma. Применение Sass в проекте не только упростило процесс стилизации, но и предоставило возможность использовать переменные, вложенные селекторы и функции, что сделало код более модульным и легче поддерживаемым. Использованы медийные запросы для реализации адаптивности. Это решение важно для обеспечения комфорта пользователей и улучшения общего пользовательского опыта, так как помогает им быстрее находить необходимую информацию.

Сайт использует адаптивную вёрстку, что позволяет корректно отображать контент на различных устройствах – от мобильных телефонов до десктопов.

Применены базовые техники SEO для повышения видимости сайта в поисковых системах, такие как использование метатегов.

# 4.2 Реализация Todo list с применением React, TypeScript, Vite

Проект [agarkovanatoliy.github.io/todo-list-project/](https://agarkovanatoliy.github.io/todo-list-project/) представляет собой приложение «To-Do List» (список дел), разработанное с использованием современного стека технологий: Vite, React и TypeScript. В проекте реализованы базовые функции управления задачами, такие как добавление, удаление, редактирование и отметка выполненных дел. Вся функциональность поддерживается состояниями компонентов, управляемыми с помощью React-хука useState.

**Структура проекта и инструменты разработки**

Vite – это современный инструмент для создания и разработки веб-приложений, который обеспечивает быструю сборку и перезагрузку проекта в реальном времени. В данном проекте Vite используется для инициализации и управления сборкой приложения. Его преимущества включают:

* Мгновенная сборка: Благодаря использованию Vite, приложение компилируется и разворачивается быстро, что сокращает время разработки.
* Горячая перезагрузка (Hot Module Replacement, HMR): Обновления кода вносятся мгновенно без перезагрузки всей страницы, что улучшает процесс разработки.

Файл vite.config.ts содержит базовую конфигурацию Vite для работы с React и TypeScript.

Сборка проекта с использованием Vite происходит в несколько шагов, начиная с инициализации проекта и заканчивая генерацией оптимизированных файлов для продакшн-развертывания. Рассмотрим, как происходит сборка Vite-проекта, какие команды используются и что они делают:

1. Инициализация проекта с помощью Vite

Для начала создается новый проект с использованием Vite. Это можно сделать с помощью команды: npm create vite@latest

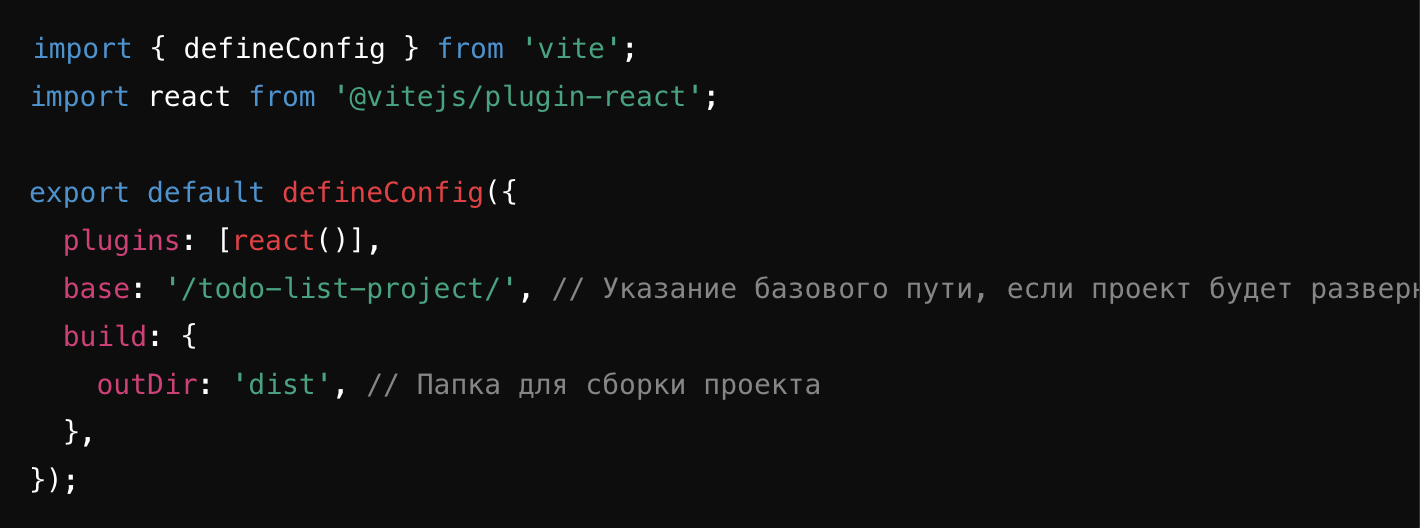
Команда инициализирует новый проект и предложит выбрать имя директории и шаблон проекта (например, react-ts для React + TypeScript). После создания проекта необходимо установить зависимости:

cd project-directory

npm install

2. Конфигурация Vite

Vite использует файл конфигурации vite.config.js или vite.config.ts, который позволяет задать параметры сборки, настройки плагинов и пути. Типичный vite.config.ts для React и TypeScript выглядит следующим образом:



Параметр base указывает базовый URL для вашего приложения, особенно если оно будет размещено в поддиректории, например, на GitHub Pages (/todo-list-project/).

3. Запуск проекта в режиме разработки

Для запуска проекта в режиме разработки используется команда: npm run dev

Эта команда:

* Запускает локальный сервер на основе Vite.
* Обеспечивает горячую перезагрузку модулей (Hot Module Replacement, HMR), что позволяет вносить изменения в код и видеть их сразу же, без перезагрузки страницы.
* Создает оптимизированный билд с минимальным временем запуска, что ускоряет процесс разработки.

4. Сборка проекта для продакшена

Когда разработка завершена, необходимо собрать проект для развертывания. Для этого используется команда: npm run build

Эта команда запускает процесс сборки, который включает следующие этапы:

1. Обработка и оптимизация ресурсов: Vite обрабатывает все модули и ресурсы (JavaScript, TypeScript, CSS, изображения и т.д.), объединяя их в несколько бандлов.
2. Минификация кода: Код минимизируется для уменьшения размера файлов, удаляются неиспользуемые участки кода (dead code elimination).
3. Генерация итоговых файлов: Все сгенерированные файлы помещаются в директорию, указанную в конфигурации (по умолчанию это папка dist). Эти файлы включают:
   * index.html – главный HTML-файл проекта.
   * Минифицированные JS и CSS файлы, такие как index.[hash].js и style.[hash].css.
   * Дополнительные файлы ресурсов, например, изображения и шрифты.
4. Создание манифеста: В Vite создается манифест (manifest.json), содержащий информацию обо всех ресурсах и их зависимостях. Это полезно для интеграции с серверными фреймворками.
5. Просмотр итогового билда

После успешной сборки можно проверить, как приложение будет выглядеть в продакшене, с помощью команды: npm run preview

Эта команда запускает локальный сервер и отображает итоговую собранную версию приложения, что позволяет убедиться, что все работает корректно перед развертыванием.

6. Деплой проекта

Для развертывания проекта на продакшн-сервере или статическом хостинге, таком как GitHub Pages, необходимо загрузить содержимое папки dist. Для деплоя на GitHub Pages используется утилита gh-pages:

1. Устанавливаем gh-pages: npm install gh-pages --save-dev
2. Добавляем скрипт для деплоя в package.json:

"scripts": {

"build": "vite build",

"deploy": "gh-pages -d dist"

}

1. Выполняем команду деплоя:

npm run deploy

Эта команда собирает проект и загружает содержимое папки dist на удаленный репозиторий GitHub Pages.

7. Обработка ошибок и оптимизация сборки

Если в процессе сборки или деплоя возникают ошибки, необходимо проверить конфигурацию vite.config.ts, убедиться, что все зависимости установлены корректно, и проверить корректность путей для статических ресурсов.

Для оптимизации сборки можно добавить дополнительные плагины и настройки, например:

* Оптимизация изображений.
* Поддержка prefetching для асинхронной загрузки ресурсов.
* Кэширование для ускорения загрузки страниц.

React

React – это библиотека для создания пользовательских интерфейсов. В проекте React используется для создания компонентов и управления состоянием приложения.

Основные компоненты включают:

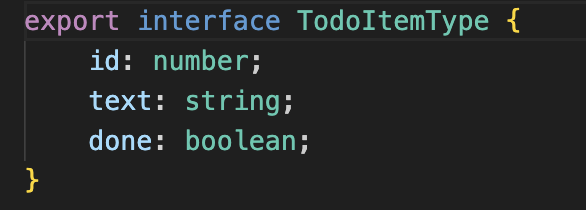
* App.tsx: Главный компонент, содержащий логику отображения списка дел.
* TodoItem.tsx: Компонент, представляющий отдельную задачу со своим состоянием.
* Toda.tsx: Форма для добавления новых задач.

TypeScript

TypeScript – это надстройка над JavaScript, обеспечивающая строгую типизацию. TypeScript помогает избежать ошибок на этапе разработки, улучшая качество кода и его читаемость.

* Типизация: В проекте определены типы для данных, таких как задачи (Task), и функции для управления состоянием задач, что упрощает работу с компонентами и их взаимодействие.

Пример определения типа задачи:



Компоненты и функциональность

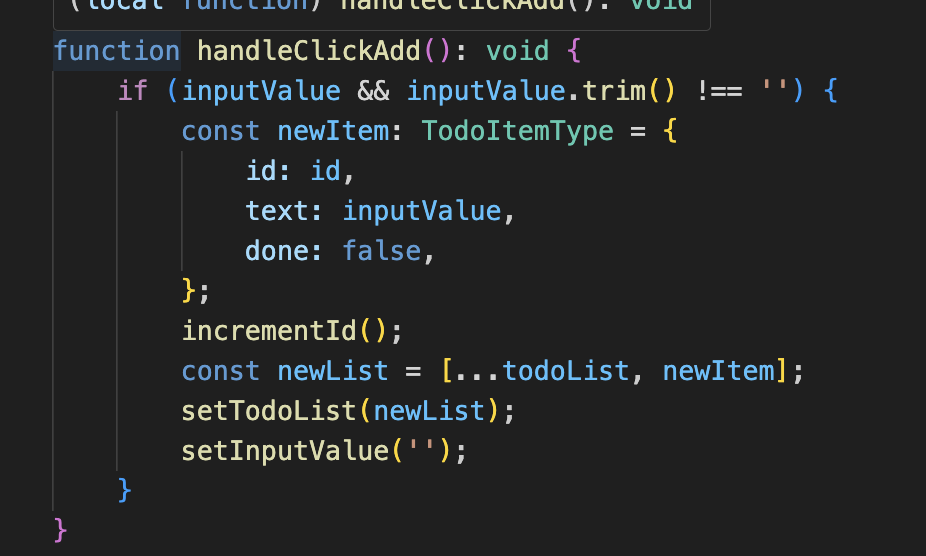
Состояния и хук useState

В проекте используется хук useState для управления состоянием задач. useState позволяет создавать и изменять состояние внутри функциональных компонентов. В проекте он используется следующим образом:

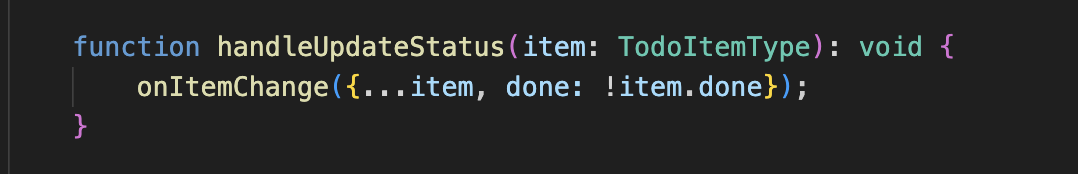
* Список задач: Состояние списка задач управляется с помощью useState и представляет собой массив объектов todoList:



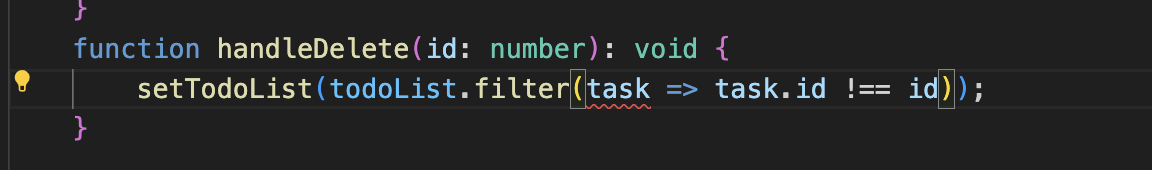
* Добавление новой задачи: Когда пользователь вводит текст и отправляет форму, новое состояние добавляется к списку задач. Это реализуется через функцию добавления задачи:



* Изменение состояния задачи (отметка выполнения): При нажатии на чекбокс задачи вызывается функция изменения состояния, которая обновляет статус done:



* Удаление задачи: Удаление задачи осуществляется путем фильтрации списка задач и обновления состояния:



**Основные компоненты проекта**

1. App.tsx — главный компонент, управляющий всеми остальными. Компонент App является центральным или корневым компонентом любого приложения, разработанного с использованием React. Его основное предназначение — объединить все дочерние компоненты и управлять их взаимодействием, состояниями и поведением на уровне всего приложения.
2. Todo.tsx — компонент формы для добавления новых задач. Он использует локальное состояние (useState) для хранения текста новой задачи и передает его в основной компонент:
3. TodoItem.tsx — компонент, представляющий отдельную задачу. Он принимает пропсы task, toggleTask и removeTask, что позволяет реализовать отметку выполнения и удаление.

**Деплой проекта на GitHub Pages**

Проект развернут на GitHub Pages с использованием конфигурации Vite:

1. Настройка деплоя: В vite.config.ts добавлена корректная настройка base для указания пути развертывания:



1. Скрипт для деплоя: В package.json добавлен скрипт для автоматического деплоя на GitHub Pages с помощью gh-pages:

"scripts": {

"build": "vite build",

"deploy": "gh-pages -d dist"

}

Таким образом проект  [agarkovanatoliy.github.io/todo-list-project/](https://agarkovanatoliy.github.io/todo-list-project/) демонстрирует использование современных технологий, таких как Vite, React и TypeScript, для создания простого и функционального приложения "To-Do List". Реализация с применением хуков React (useState) обеспечивает эффективное управление состоянием и делает код более лаконичным и легко поддерживаемым.

# 4.3 Деплой на GitHub Pages

В дипломной работе были имплементированы как деплои статических проектов, так и проектов на Vite.

Шаги деплоя статического проекта на GitHub Pages

1. Создание репозитория на GitHub

Для начала нужно создать новый репозиторий на GitHub:

1. Перейдите на [GitHub](https://github.com/) и авторизуйтесь.
2. Нажмите на кнопку New для создания нового репозитория.
3. Задайте имя репозитория (например, my-static-project).
4. Выберите, будет ли репозиторий публичным (рекомендуется для GitHub Pages) или приватным.
5. Нажмите на кнопку Create repository.

2. Клонирование репозитория и добавление файлов проекта

После создания репозитория нужно склонировать его на локальный компьютер и добавить файлы вашего статического проекта:

1. Откройте терминал (или командную строку) и выполните команду:

git clone https://github.com/username/my-static-project.git

Замените username на ваше имя пользователя на GitHub, а my-static-project — на имя вашего репозитория.

1. Перейдите в папку репозитория:

cd my-static-project

1. Скопируйте или переместите все файлы вашего статического проекта (например, index.html, style.css, папки с изображениями и т.д.) в эту папку.
2. После этого выполните команды:

git add .

git commit -m "Initial commit with static project"

git push origin main

3. Создание ветки gh-pages и деплой проекта

Теперь, когда ваш проект загружен в репозиторий, необходимо создать ветку gh-pages и выполнить деплой. Есть два основных способа: вручную или с помощью утилиты gh-pages.

Способ 1: Деплой вручную

1. Создайте ветку gh-pages:

git branch gh-pages

1. Переключитесь на эту ветку:

git checkout gh-pages

1. Удалите все файлы, кроме файлов вашего проекта (index.html, style.css и т.д.), чтобы в этой ветке находился только собранный проект.
2. Выполните команды для добавления и пуша:

git add .

git commit -m "Deploy static project to GitHub Pages"

git push origin gh-pages

Способ 2: Использование библиотеки gh-pages

Для автоматизации деплоя можно использовать библиотеку gh-pages, которая позволяет проще и быстрее развернуть проект на GitHub Pages.

1. Установите библиотеку gh-pages:

npm install gh-pages --save-dev

1. Откройте файл package.json и добавьте следующие строки:

"homepage": "https://username.github.io/my-static-project",

"scripts": {

"predeploy": "npm run build",

"deploy": "gh-pages -d build"

}

* + Замените username на ваше имя пользователя, а my-static-project — на имя вашего репозитория.
  + Скрипт predeploy выполняет сборку проекта перед деплоем (если используется статический генератор или фреймворк).
  + Скрипт deploy загружает содержимое папки build в ветку gh-pages.

1. Выполните команду:

npm run deploy

Эта команда создаст новую ветку gh-pages, если ее не существует, и загрузит собранный проект в нее.

4. Настройка GitHub Pages

1. Перейдите на страницу вашего репозитория на GitHub.
2. Перейдите на вкладку Settings (Настройки).
3. В меню слева выберите раздел Pages.
4. В разделе Source выберите ветку gh-pages и папку /root.
5. Нажмите кнопку Save (Сохранить).

После этого GitHub начнет обработку файлов и ваш сайт станет доступен по адресу: https://username.github.io/my-static-project.

5. Проверка и отладка сайта

После завершения настройки и деплоя, сайт должен быть доступен по указанному URL. Если проект не отображается или возникают ошибки (например, белый экран или отсутствующие стили), проверьте следующее:

1. Пути к ресурсам. Убедитесь, что пути к изображениям, стилям и скриптам заданы правильно. Рекомендуется использовать относительные пути или абсолютные с указанием base.
2. Правильность структуры проекта. Убедитесь, что все необходимые файлы, такие как index.html, находятся в корневой директории.
3. Проверка консоли браузера. Посмотрите на ошибки в консоли браузера, чтобы понять, какие ресурсы не могут быть найдены или загружены.

Основные шаги деплоя проекта на GitHub Pages с Vite

1. Создание и настройка репозитория на GitHub

Перед началом деплоя необходимо создать репозиторий на GitHub:

1. Перейдите на GitHub и создайте новый репозиторий.
2. Задайте имя репозитория. В случае с деплоем на GitHub Pages, рекомендуется называть репозиторий как username.github.io, если вы хотите развернуть его на уровне домена, или username.github.io/project-name, если проект будет находиться в поддиректории.
3. После создания репозитория склонируйте его на ваш компьютер, если это еще не сделано:

git clone https://github.com/username/ project-name.git

2. Установка необходимых зависимостей

Если проект еще не содержит всех нужных зависимостей, необходимо установить их. Например, если вы используете create-react-app или vite, сначала убедитесь, что зависимости установлены:

npm install

Для деплоя на GitHub Pages понадобится установить библиотеку gh-pages:

npm install gh-pages --save-dev

3. Настройка скриптов в package.json

Необходимо добавить необходимые скрипты для сборки и деплоя проекта на GitHub Pages. Откройте файл package.json и добавьте следующие строки:

"homepage": " https://github.com/username/project-name/",

"scripts": {

"predeploy": "npm run build",

"deploy": "gh-pages -d dist"

}

Параметр "homepage" задает базовый URL для приложения. В данном случае, это https://github.com/username/project-name/.

* Скрипт predeploy автоматически запустит сборку проекта (npm run build) перед тем, как будет запущен деплой.
* Скрипт deploy загружает содержимое папки dist на ветку gh-pages в удаленном репозитории.

4. Настройка vite.config.js (если используется Vite)

Если ваш проект создан с помощью Vite, необходимо добавить базовый путь для корректного отображения на GitHub Pages. javascript

import { defineConfig } from 'vite';

import react from '@vitejs/plugin-react';

export default defineConfig({

plugins: [react()],

base: '/my-CV/', // Указываем базовый URL для проекта

build: {

outDir: 'dist',

}

});

Параметр base указывает путь, с которого будет начинаться проект в браузере. В данном случае, это /project-name/.

5. Сборка проекта

Перед деплоем необходимо собрать проект. Выполните команду:

npm run build

Эта команда создаст папку dist, содержащую все необходимые для работы сайта файлы (HTML, JavaScript, CSS и т.д.).

6. Деплой проекта

После успешной сборки проекта выполните команду для деплоя на GitHub Pages:

npm run deploy

Эта команда выполнит следующие действия:

* Создаст новую ветку gh-pages, если она еще не существует.
* Загрузит все содержимое папки dist в эту ветку.
* Настроит GitHub Pages для использования этой ветки в качестве источника данных для сайта.

# 5. ТЕСТИРОВАНИЕ И АНАЛИЗ ПРОЕКТА

Данный раздел посвящен тестированию проекта [agarkovanatoliy.github.io/portfolio/](https://agarkovanatoliy.github.io/portfolio/) на различных устройствах и браузерах, а также анализу полученных результатов. Тестирование помогает убедиться, что сайт отображается корректно и одинаково на всех поддерживаемых платформах, соответствует современным стандартам и обеспечивает положительный пользовательский опыт.

# 5.1 Тестирование на различных устройствах и браузерах

В рамках тестирования проекта было проведено тестирование на различных устройствах, включающих десктопы, планшеты и смартфоны, а также на популярных веб-браузерах. Тестирование позволило выявить, насколько адаптивная вёрстка и функциональность сайта соответствуют ожиданиям и требованиям пользователей.

Список устройств и браузеров для тестирования:

1. Десктопы:
   * Операционная система: Windows 10, Windows 11, macOS Monterey.
   * Браузеры: Google Chrome, Mozilla Firefox, Microsoft Edge, Safari.
2. Планшеты:
   * Устройства: iPad Pro, Samsung Galaxy Tab.
   * Браузеры: Safari (iOS), Google Chrome (Android).
3. Смартфоны:
   * Устройства: iPhone 12, iPhone 14, Samsung Galaxy S21, Google Pixel 6.
   * Браузеры: Safari (iOS), Google Chrome (Android).

Проведенные тесты:

* Тестирование адаптивной вёрстки: Проверка корректного отображения контента на экранах различных размеров и разрешений. Была проведена симуляция размеров экранов с помощью инструмента DevTools в браузерах, а также тестирование на реальных устройствах.
* Кроссбраузерное тестирование: Проверка отображения сайта на разных браузерах для выявления несовместимости или отличий в рендеринге элементов.
* Проверка интерактивных элементов: Тестирование всех кнопок, ссылок, анимаций, формы обратной связи и навигационных элементов.
* Тестирование производительности: Оценка скорости загрузки сайта на различных устройствах и проверка размера загружаемых ресурсов.

Результаты тестирования:

* Сайт корректно отображается на всех протестированных браузерах и устройствах, за исключением небольших отличий в размерах шрифтов и интервалах в некоторых версиях браузера Internet Explorer.
* Анимации и интерактивные элементы работают корректно, все переходы и кнопки функционируют без сбоев.
* На смартфонах и планшетах навигация адаптируется под сенсорный ввод, что обеспечивает комфортное взаимодействие.
* Слайдер фотографий, реализованный на Bootstrap, корректно масштабируется в зависимости от размеров экрана, поддерживая как горизонтальную, так и вертикальную ориентацию.
* Время загрузки сайта не превышает 2 секунд на большинстве устройств, что является отличным показателем для сайтов такого типа.

# 5.2 Анализ полученных результатов

Анализ результатов тестирования показал, что сайт [agarkovanatoliy.github.io/portfolio/](https://agarkovanatoliy.github.io/portfolio/) успешно прошел проверку на совместимость с различными устройствами и браузерами, обеспечивая хорошую производительность и положительный пользовательский опыт. Тем не менее, были выявлены некоторые области, которые могут быть улучшены для повышения общей стабильности и удобства использования:

1. Адаптивная вёрстка: Сайт в целом хорошо адаптируется к различным размерам экранов, однако на некоторых устройствах (например, iPad в горизонтальном режиме) наблюдаются незначительные смещения элементов, которые можно устранить путем дополнительной настройки стилей CSS.
2. Кроссбраузерная совместимость: Небольшие отличия в отображении шрифтов и размеров элементов были замечены в браузере Internet Explorer. Несмотря на устаревание данного браузера, возможно, стоит учесть данные отличия для поддержки пользователей, использующих его на корпоративных устройствах.
3. Производительность: Хотя сайт загружается быстро на большинстве устройств, на устройствах с низкой производительностью загрузка анимаций может занимать больше времени. Рекомендуется оптимизировать размер изображений и использовать метод отложенной загрузки (lazy loading) для уменьшения времени загрузки.
4. Пользовательский опыт (UX): Навигация и доступность интерактивных элементов соответствует современным стандартам. Тем не менее, можно улучшить переходы между разделами сайта, добавив анимации переходов и плавные эффекты при смене контента.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения проекта [agarkovanatoliy.github.io/portfolio/](https://agarkovanatoliy.github.io/portfolio/) были рассмотрены и успешно внедрены современные технологии для создания интерактивного и адаптивного веб-резюме. Проект разрабатывался с использованием HTML5, CSS3, SASS, JavaScript, а также фреймворков и библиотек React, что позволило не только создать удобный и привлекательный интерфейс, но и обеспечить высокую производительность и кроссбраузерную совместимость.

В ходе анализа теоретической базы и практического исследования были сделаны следующие выводы:

1. Теоретические и практические выводы:  
   Использование передовых веб-технологий, таких как React и TypeScript, позволяет значительно улучшить структуру кода и его поддержку в будущем. Деплой проекта на GitHub Pages обеспечивает удобный способ публикации и обновления сайта, а также предоставляет бесплатный хостинг, что актуально для личных проектов и портфолио.
2. Оценка проведенного исследования и его результаты:  
   Проект прошел тщательное тестирование на различных устройствах и браузерах, что подтвердило его корректную работу и адаптивность. Основные цели были достигнуты — сайт получился полностью адаптивным, визуально привлекательным и соответствующим современным стандартам веб-разработки.
3. Практическая значимость работы:  
   Разработанный проект обладает практической значимостью, так как он может использоваться в качестве личного портфолио для демонстрации навыков и компетенций в веб-разработке. Проект, размещенный на GitHub Pages, легко доступен для просмотра потенциальными работодателями или клиентами, что упрощает процесс презентации навыков разработчика. Использованные в проекте технологии позволяют его легко дорабатывать и масштабировать под различные задачи, такие как добавление новых страниц, интеграция API или настройка взаимодействия с базами данных.
4. Общий итог – достижение цели, выполнение задач, доказательство гипотезы:  
   Целью данного проекта было создание интерактивного веб-портфолио с использованием современных технологий, обеспечивающего адаптивность и удобство использования на любых устройствах. Все поставленные задачи были успешно выполнены, а гипотеза о том, что использование таких технологий, как React, TypeScript, Animista и Bootstrap, позволяет достичь высокого уровня функциональности и визуальной привлекательности, была доказана.
5. Предложения по совершенствованию объекта исследования:  
   В будущем проект можно расширить, добавив следующие улучшения:
   * Внедрение серверной части с использованием Node.js для реализации серверной логики и взаимодействия с базами данных.
   * Добавление функционала для оставления отзывов или комментариев пользователями, что повысит интерактивность сайта.
   * Оптимизация анимаций и изображений для уменьшения времени загрузки и повышения производительности на устройствах с низкими ресурсами.
   * Интеграция API для отображения актуальной информации, такой как статистика по просмотрам резюме или публикация актуальных проектов.

Создание данного веб-портфолио позволило не только продемонстрировать навыки и знания в области фронтенд-разработки, но и подтвердить, что использование современных фреймворков и инструментов разработки значительно упрощает процесс создания удобных и адаптивных интерфейсов. Реализация проекта стала хорошим опытом для дальнейшего развития в области веб-разработки и внедрения более сложных и функциональных веб-приложений.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Бёрч, Д. HTML5 и CSS3. Полное руководство для начинающих. М.: Вильямс, 2019.
2. Ли Бенджамин. Sass и Compass. Введение в CSS-препроцессоры. СПб: Питер, 2018.
3. Флэнаган Дэвид. JavaScript: Подробное руководство. 6-е изд. СПб.: Символ-Плюс, 2019.
4. Вассерман Эрик. React.js: Быстрый старт. М.: Эксмо, 2020.
5. Freeman Eric, Robson Elisabeth. Head First HTML and CSS. O'Reilly Media, 2012.
6. Документация библиотеки Animista: [**Animista - CSS Animations On Demand**](https://animista.net/).
7. Официальная документация React: [**https://react.dev/**](https://react.dev/)
8. Официальная документация Bootstrap: [**https://getbootstrap.com/docs**](https://getbootstrap.com/docs)
9. Официальная документация TypeScript: [**https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html**](https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html)
10. Руководство по Vite: [**https://vite.dev/guide/**](https://vite.dev/guide/)
11. Официальная документация по GitHub Pages: [**GitHub Pages Documentation**](https://docs.github.com/en/pages)
12. MDN Web Docs: [**MDN Web Docs - HTML**](https://developer.mozilla.org/en-US/docs/Web/HTML), [**MDN Web Docs - CSS**](https://developer.mozilla.org/en-US/docs/Web/CSS), [**MDN Web Docs - JavaScript**](https://developer.mozilla.org/en-US/docs/Web/JavaScript)
13. Официальная документация по Git и GitHub: [**Git Documentation**](https://git-scm.com/doc), [**GitHub Documentation**](https://docs.github.com/en)