Geekbrains

Разработка личного веб-портфолио с использованием современных технологий веб-разработки и анимации, размещенного на хостинге SpaceWeb

<http://fyodartemm.temp.swtest.ru/>

Программа: Разработчик

Специализация: Frontend разработка (React)

ФИО: Федоров Артем Николаевич

Калининград

2025

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc190128761)

[1.1 Актуальность темы 3](#_Toc190128762)

[1.2 Цель и задачи дипломной работы 5](#_Toc190128763)

[1.3 Краткое описание проекта 8](#_Toc190128764)

[2. ОБЗОР ИСПОЛЬЗУЕМЫХ ТЕХНОЛОГИЙ 9](#_Toc190128765)

[2.1 HTML5, CSS3, SaSS 9](#_Toc190128766)

[2.2 JavaScript и ES6+ 11](#_Toc190128767)

[2.3 React: описание и применение 14](#_Toc190128768)

[2.4 Bootstrap: фреймворк для визуализации данных 18](#_Toc190128769)

[2.6 Инструменты разработки 25](#_Toc190128770)

[3. ПРОЕКТИРОВАНИЕ ВЕБ-ПОРТФОЛИО 28](#_Toc190128771)

[3.1 Структура веб-портфолио 28](#_Toc190128772)

[3.2 Дизайн интерфейса и пользовательского опыта 30](#_Toc190128773)

[3.3 Разработка адаптивной верстки 32](#_Toc190128774)

[4. РЕАЛИЗАЦИЯ ПРОЕКТОВ 36](#_Toc190128775)

[4.1 Реализация проекта вёрстки лендинга 36](#_Toc190128776)

[4.2 Реализация приложения для управления каталогом продуктов с применением HTML, CSS, JavaScript, React, React-Redux и Redux-Toolkit 37](#_Toc190128777)

[5. ТЕСТИРОВАНИЕ И АНАЛИЗ ПРОЕКТА 46](#_Toc190128778)

[5.1 Тестирование на различных устройствах и браузерах 46](#_Toc190128779)

[5.2 Анализ полученных результатов 47](#_Toc190128780)

[ЗАКЛЮЧЕНИЕ 49](#_Toc190128781)

[СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ 51](#_Toc190128782)

# ВВЕДЕНИЕ

# 1.1 Актуальность темы

В условиях стремительного развития цифровых технологий создание онлайн-портфолио для разработчиков становится не просто полезным, а необходимым инструментом для профессионального успеха. В эпоху, когда первое знакомство работодателя с кандидатом происходит через интернет, наличие качественного веб-портфолио превращается в ключевой фактор, влияющий на трудоустройство и карьерный рост. Его значимость обусловлена рядом важных аспектов:

1. **Гибкость и персонализация**

Онлайн-портфолио – это инвестиция в будущее. Оно позволяет разработчику выделиться среди конкурентов, привлечь внимание работодателей и клиентов, а также расширить свои профессиональные возможности. Персонализированный подход, включающий описание опыта, образования и достижений, помогает установить доверие и показать, как специалист может решить задачи потенциальных заказчиков.

2)**Практическая подготовка к профессиональной среде**

Создание портфолио с использованием современных технологий – это не только демонстрация навыков, но и возможность применить их на практике. Такой проект позволяет разработчику подготовиться к реальным задачам, с которыми он столкнется в профессиональной среде.

3)**Простота в обслуживании и расширяемость**

Онлайн-портфолио обеспечивает постоянный доступ к информации для потенциальных работодателей, независимо от их местоположения. Это расширяет возможности для поиска работы на международном уровне. Интеграция с социальными сетями, такими как LinkedIn, и добавление контактной информации упрощают коммуникацию. Кроме того, использование аналитических инструментов позволяет отслеживать посещаемость и вносить улучшения в структуру и дизайн портфолио.

4) **Универсальность и поддержка различных устройств**

Современные пользователи взаимодействуют с веб-ресурсами с различных устройств – от смартфонов до настольных компьютеров. Поэтому важно, чтобы портфолио было адаптировано под разные экраны и браузеры. Использование адаптивной верстки и CSS-фреймворков, таких как Bootstrap, обеспечивает удобство просмотра и повышает доступность сайта для широкой аудитории.

5) **Профессиональная видимость и самовыражение**

Веб-портфолио позволяет разработчику эффективно продемонстрировать свои навыки, опыт и достижения. В отличие от статичных документов, таких как PDF-файлы или простые списки проектов, интерактивное портфолио предоставляет возможность показать свои работы в действии. Это может включать не только скриншоты, но и рабочие демоверсии приложений или сайтов, что позволяет потенциальным работодателям оценить качество кода, архитектуру проекта и пользовательский интерфейс. Кроме того, интеграция с платформами для хранения кода, такими как GitHub или GitLab, повышает доверие и позволяет продемонстрировать стиль программирования.

6) **Развитие карьеры и совершенствование навыков**

Постоянное обновление портфолио помогает разработчику оставаться в курсе новых технологий и тенденций. Это не только способ продемонстрировать свои достижения, но и платформа для экспериментов с новыми инструментами и подходами, что способствует профессиональному развитию.

**7) SEO-оптимизация и видимость в сети**

Оптимизация портфолио для поисковых систем (SEO) – важный шаг для повышения его видимости в интернете. Использование метатегов, оптимизация скорости загрузки и других SEO-техник помогает привлечь больше посетителей, включая потенциальных работодателей и клиентов.

8) **Интерактивность и динамические элементы**

Применение анимационных библиотек (например, Animista) и современных технологий, таких как React, позволяет создавать динамичные и интерактивные интерфейсы. Это делает портфолио более привлекательным и удобным для пользователя. Анимации при прокрутке, слайдеры и плавные переходы между страницами улучшают восприятие информации, а динамическая подгрузка контента помогает сосредоточить внимание на ключевых моментах.

9) **Использование передовых технологий**

Онлайн-портфолио – это не только способ показать свои работы, но и возможность заявить о себе как о креативном и технически подкованном специалисте. Использование современных инструментов, таких как React, Angular или Vue.js, а также платформ вроде Wix или Squarespace, позволяет создать уникальный дизайн, отражающий индивидуальность разработчика. Добавление блога, описания участия в open-source проектах или личного опыта делает портфолио более живым и запоминающимся, подчеркивая стремление к профессиональному развитию.

# 1.2 Цель и задачи дипломной работы

**Цель дипломной работы**

Данная дипломная работа направлена на создание интерактивного и адаптивного веб-портфолио, которое станет современным и эффективным инструментом для демонстрации профессиональных компетенций, реализованных проектов и навыков в области веб-разработки. Основная цель проекта заключается в разработке веб-ресурса, сочетающего в себе визуальную привлекательность, удобство использования и соответствие актуальным стандартам веб-дизайна и программирования. Планируется создать сайт, который будет корректно отображаться на различных устройствах (компьютеры, планшеты, смартфоны) и обеспечит простоту обновления и управления контентом. Для реализации проекта будут использованы современные технологии, такие как HTML5, CSS3, JavaScript, а также библиотеки и фреймворки, включая React и Animista.

**Задачи дипломной работы**

Для достижения поставленной цели необходимо выполнить ряд задач, которые охватывают все этапы разработки – от анализа и проектирования до тестирования и внедрения. Основные задачи включают:

1. Изучение актуальных технологий для веб-разработки:
   * Провести анализ существующих инструментов и подходов к созданию веб-портфолио.
   * Определить оптимальный набор технологий, включая HTML5, CSS3, JavaScript, React, а также дополнительные библиотеки для анимации и визуализации, которые обеспечат высокую производительность и функциональность проекта.
2. Проектирование архитектуры и пользовательского интерфейса:
   * Разработать информационную архитектуру сайта, продумав логическую структуру и навигацию.
   * Создать макеты интерфейса, учитывая требования к адаптивности, доступности и удобству использования на различных устройствах.
3. Создание адаптивного дизайна и кроссбраузерной верстки:
   * Выполнить верстку страниц с использованием HTML5 и CSS3, обеспечив кроссбраузерную совместимость.
   * Настроить адаптивный дизайн, чтобы сайт корректно отображался на экранах разных размеров – от мобильных устройств до настольных компьютеров.
4. Разработка компонентов с использованием React:
   * Создать основные компоненты сайта, используя библиотеку React, для обеспечения модульности и повторного использования кода.
   * Настроить управление состоянием приложения и взаимодействие между компонентами для повышения производительности и удобства разработки.
5. Добавление анимаций и интерактивных элементов:
   * Создать основные компоненты сайта, используя библиотеку React, для обеспечения модульности и повторного использования кода.
   * Настроить управление состоянием приложения и взаимодействие между компонентами для повышения производительности и удобства разработки.
6. Оптимизация производительности и тестирование:
   * Провести оптимизацию кода и ресурсов для ускорения загрузки страниц и повышения производительности.
   * Выполнить тестирование на различных устройствах и браузерах, чтобы убедиться в корректной работе сайта и устранить возможные ошибки.
7. Публикация проекта в сети
   * Настроить проект для размещения в интернете, выбрав подходящий хостинг или платформу для публикации.
   * Проверить корректность отображения сайта и его функциональность после загрузки на сервер, чтобы убедиться в его доступности и работоспособности для пользователей
8. Подготовка документации по проекту:
   * Подробно описать все этапы создания проекта, включая проектирование, разработку, тестирование и внедрение.
   * Подготовить документацию, содержащую описание используемых технологий, исходный код, скриншоты и иллюстрации, которые помогут понять логику и структуру проекта.
9. Оценка результатов и определение перспектив:
   * Провести оценку готового продукта, выделив его сильные стороны и возможные недостатки.
   * Определить направления для дальнейшего улучшения и расширения функциональности веб-портфолио.

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

Основная гипотеза работы заключается в том, что использование современных технологий, таких как React, TypeScript, Animista и Bootstrap, позволяет достичь высокого уровня функциональности, производительности и визуальной привлекательности веб-портфолио. Эти инструменты обеспечивают гибкость, масштабируемость и удобство разработки, что делает их идеальным выбором для создания профессионального и адаптивного веб-ресурса.

# 1.3 Краткое описание проекта

Данный проект представляет собой персональное веб-портфолио, разработанное для демонстрации профессиональных навыков и реализованных проектов. Основная цель портфолио – показать уровень компетенций автора в области веб-разработки, а также представить его как специалиста, способного создавать современные и функциональные решения. Портфолио включает несколько пет-проектов, выполненных с использованием различных технологий, что позволяет продемонстрировать широкий спектр навыков и подходов к решению задач.

Портфолио создано с использованием современных веб-технологий, таких как HTML5, CSS3 и JavaScript. Особое внимание уделено визуальной составляющей: для добавления анимаций и интерактивных элементов использовалась библиотека Animista, что сделало интерфейс более динамичным и привлекательным. Структура портфолио продумана таким образом, чтобы пользователи могли легко находить нужные разделы и проекты, а навигация была интуитивно понятной.

Ключевые проекты, представленные в портфолио:

1. Приложение для управления каталогом продуктов:
   * Создано с применением React-Redux и Redux Toolkit.
   * Позволяет выполнять базовые операции с товарами: добавление, редактирование, удаление и просмотр списка продуктов. Интерфейс приложения интуитивно понятен и удобен в использовании.
2. Лендинг для интернет-магазина:
   * Разработан с использованием React, JavaScript, HTML и Sass.
   * Проект демонстрирует навыки создания адаптивных интерфейсов, реализации пользовательского взаимодействия и интеграции с внешними API.

Каждый из проектов в портфолио подчеркивает различные аспекты профессиональных навыков автора, такие как работа с современными фреймворками, управление состоянием приложения, создание адаптивного дизайна и интеграция с внешними сервисами.

Портфолио служит не только инструментом для демонстрации выполненных работ, но и площадкой для творческого самовыражения. Оно отражает индивидуальный стиль автора и его подход к решению задач, что делает его уникальным и запоминающимся.

# 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.

Но также есть и отрицательные моменты, о них ниже:

Одно из самых раздражающих — подъем состояния наверх через колбеки, которые нужно прокидывать вниз с самого верхнего уровня. Прокидывать приходится не только колбеки, но и любые данные. Получается, что множество промежуточных компонентов выступают в качестве прокси — они пропускают сквозь себя данные, которыми не пользуются.

Второй раздражающий фактор — рендеринг и логика смешаны в одном месте, что быстро раздувает компоненты и усложняет понимание. Сюда же добавляются неконтролируемые побочные эффекты вперемешку с обновлением данных.

Для решения этих проблем появились менеджеры состояния, в том числе Redux — официальный менеджер, который поддерживает сама компания Facebook.

Как работают Redux и Toolkit:

Redux — это очень простая библиотека, предназначенная исключительно для управления состоянием. Ее разрабатывали под использование в React, но на самом деле от React она не зависит — можно использовать ее с чем угодно.

Для связи Redux с React понадобится Redux Toolkit, через который мы произведем всю необходимую интеграцию. Redux Toolkit не просто склеивает Redux и React. Он меняет способ работы с Redux до неузнаваемости, привносит множество полезных абстракций — например, широко используемые мидлвары.

Цель этой библиотеки — сделать хранение, извлечение и использование данных максимально удобными, убрав шаблонный код, характерный для приложений на Redux.

Redux Toolkit вводит много новых понятий, которые тяжело рассматривать в отрыве от общей картины. Поэтому сначала мы без погружения рассмотрим пример, в котором все собрано в одном месте, а уже в следующих уроках разберем по косточкам каждый элемент.

Основные понятия Redux — actions, dispatcher, store*.*

Store — это состояние веб-компонента, которое хранит в себе всю информацию (или ту которую вы решили сохранить в него). В дальнейшем стор будет доступен из любого компонента вашего приложения.

Action — действие, описывает что нужно сделать. Согласно принципам функционального программирования, мы не можем изменять объект напрямую, поэтому нам нужны экшены, чтобы передать их в диспатчер и «сказать», что нужно сделать.

Dispatcher— сообщает хранилищу о каком-то действии (action) и передает ему обновленную информацию.

# 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

Современная веб-разработка требует использования инструментов, которые обеспечивают удобную среду разработки и простой деплой проектов. В этой секции мы рассмотрим Vite.

Данный инструмент играет 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.

# 3. ПРОЕКТИРОВАНИЕ ВЕБ-ПОРТФОЛИО

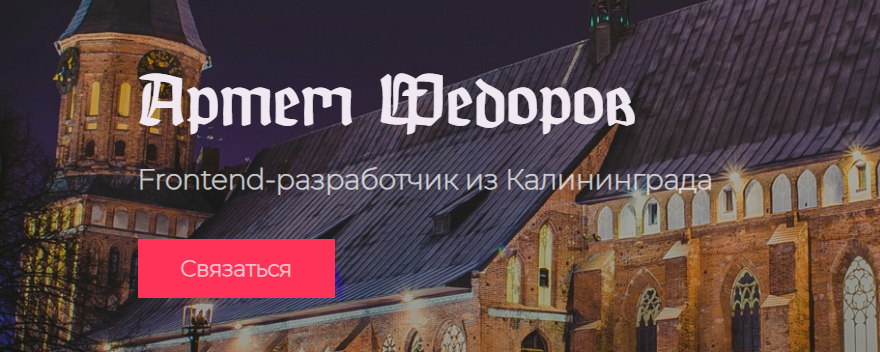
# 3.1 Структура веб-портфолио

Веб-портфолио представляет собой статический сайт, состоящий из нескольких разделов, каждый из которых направлен на демонстрацию профессиональных навыков, реализованных проектов и достижений автора. Основная цель структуры – обеспечить удобство навигации и представить информацию в максимально понятной и визуально привлекательной форме.

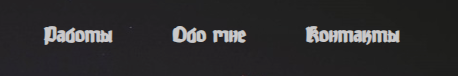
Сайт разработан с использованием современных технологий, которые позволяют создать интерактивный и динамичный интерфейс. Особое внимание уделено дизайну, чтобы портфолио не только выполняло свою функциональную роль, но и оставляло приятное впечатление у посетителей.

**Основная структура веб-портфолио**

1. Главная страница:
   * Заголовок и приветственное сообщение: На главной странице размещено имя автора и краткий слоган, который сразу дает понять посетителю специализацию и направление деятельности.



* + Навигационное меню: В верхней части страницы расположено меню, позволяющее быстро переходить между ключевыми разделами портфолио, такими как «Работы», «Обо мне» и «Контакты».



1. Обо мне:
   * В этом разделе представлена информация о профессиональном опыте, ключевых навыках и специализации автора. Здесь также можно найти краткое описание личности, подходов к работе и профессиональных интересов, что помогает создать целостное представление о кандидате.
2. Работы:
   * Этот раздел посвящен демонстрации выполненных работ, включая пет-проекты и учебные задания.
   * Каждый проект представлен в виде карточки, которая содержит:

* Название проекта.
* Краткое описание.
* Демонстрационные материалы (фото, видео или скриншоты).
* Ссылку на подробную страницу проекта или его исходный код.

1. Контакты:

В этом разделе представлена вся необходимая информация для связи с автором. Ссылки на профили в социальных сетях и профессиональных платформах, таких как:

* + GitHub
  + GitVerse
  + Telegram

Данное веб-портфолио обладает четкой и логически продуманной структурой. Каждый раздел сайта выполнен в едином стиле, что обеспечивает целостность дизайна и позволяет посетителям легко находить информацию о навыках и проектах автора. Визуальная привлекательность, дополненная анимациями, делает портфолио запоминающимся и удобным в использовании. Применение современных технологий, таких как React, Bootstrap и Animista демонстрирует высокий уровень владения инструментами фронтенд-разработки и подчеркивает профессионализм автора.

# 3.2 Дизайн интерфейса и пользовательского опыта

Разработка веб-портфолио базируется на современных подходах к созданию интерфейсов и обеспечению комфортного взаимодействия пользователя с сайтом. Основное внимание уделено тому, чтобы сделать портфолио не только визуально привлекательным, но и удобным в использовании. Ниже рассмотрены ключевые аспекты, которые легли в основу дизайна и пользовательского опыта.

**1. Визуальная концепция и стилистика (UI Design)**

Минимализм как основа дизайна: В оформлении сайта использован минималистичный подход, который позволяет избежать перегруженности интерфейса. Это помогает пользователю сосредоточиться на ключевой информации – проектах и навыках автора.

Цветовая палитра: В дизайне портфолио использована сдержанная цветовая гамма, сочетающая нейтральные и акцентные оттенки. Основной фон выполнен в черном цвете, что придает сайту элегантность и современность. Черный фон создает выразительный контраст с яркими элементами, такими как изображения и графические акценты, что делает их более заметными и привлекательными для пользователя. Такой выбор фона не только подчеркивает ключевые работы и проекты, но и способствует созданию атмосферы профессионализма и экспертности. Черный цвет часто ассоциируется с серьезностью и глубиной, что может произвести положительное впечатление на потенциальных работодателей или клиентов. Кроме того, черный фон позволяет использовать светлые и яркие цветовые акценты, что обеспечивает гармоничное и сбалансированное восприятие контента. Это помогает пользователю сосредоточиться на представленных работах, минимизируя отвлекающие факторы. В результате, портфолио становится не только визуально привлекательным, но и функциональным, позволяя посетителям легко находить нужную информацию. Использование черного фона также открывает широкие возможности для экспериментов с типографикой и графическими элементами. Это делает дизайн портфолио уникальным и креативным, выделяя его на фоне других подобных сайтов. В целом, выбор черного цвета в качестве основного фона – это стратегическое решение, которое способствует созданию запоминающегося визуального образа и укреплению профессиональной идентичности автора.

Типографика: Шрифты подобраны с учетом читаемости и визуальной иерархии. Крупные заголовки используются для разделов, а более мелкий текст – для деталей. Это помогает пользователю быстро находить нужную информацию.

Иконки и графические элементы: Для визуального обозначения навыков, инструментов и социальных сетей используются иконки. Они делают интерфейс более интуитивно понятным и визуально привлекательным.

**2. Удобство взаимодействия (UX Design)**

Логичная структура: Сайт разделен на несколько секций, таких как «Работы», «Обо мек» и «Контакты». Каждая из них имеет четкую структуру, что позволяет пользователю легко находить нужную информацию.

Адаптивность: Дизайн сайта адаптирован под различные устройства – от смартфонов до настольных компьютеров. Для этого используются современные технологии, такие как Flexbox и Grid, которые обеспечивают гибкость верстки.

Интерактивность: Элементы интерфейса, такие как кнопки и карточки, реагируют на действия пользователя (например, изменение цвета при наведении). Это делает взаимодействие с сайтом более живым и приятным.

Оптимизация производительности: Для обеспечения быстрой загрузки сайта используются легковесные инструменты, такие как Vite. Это позволяет минимизировать время ожидания и повысить удовлетворенность пользователей

# 3.3 Разработка адаптивной верстки

Адаптивность – один из ключевых аспектов современного веб-дизайна. В данном проекте она реализована с использованием передовых технологий, которые обеспечивают корректное отображение сайта на любых устройствах.

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

Для обеспечения корректного отображения сайта на различных устройствах активно применяются медиазапросы. Это мощный инструмент CSS, который позволяет адаптировать стили в зависимости от ширины экрана пользовательского устройства. Благодаря медиазапросам, можно гибко настраивать внешний вид элементов интерфейса, чтобы они выглядели одинаково хорошо как на больших мониторах, так и на компактных мобильных устройствах.

Например, с помощью медиазапросов можно изменять размеры шрифтов, отступы между блоками, а также расположение элементов на странице. Это особенно важно для обеспечения удобочитаемости текста и предотвращения наложения элементов друг на друга. Таким образом, пользователь всегда получает комфортный опыт взаимодействия с сайтом, независимо от того, с какого устройства он заходит.

Медиазапросы также позволяют скрывать или показывать определенные элементы интерфейса в зависимости от размера экрана. Например, на мобильных устройствах можно упростить навигационное меню, чтобы оно занимало меньше места, а на десктопах – отобразить его в полном объеме.

Использование медиазапросов делает сайт более гибким и отзывчивым, что является важным аспектом современного веб-дизайна.

**2. Гибкие сетки (Flexible Grids)**

Для создания адаптивной и отзывчивой верстки в портфолио используется гибкая сеточная система. Этот подход позволяет элементам интерфейса автоматически подстраиваться под доступное пространство, сохраняя при этом четкую структуру и пропорции.

Основу гибкой сетки составляют современные CSS-технологии, такие как Flexbox и Grid. Flexbox используется для выравнивания элементов в строках или колонках, что особенно полезно для создания гибких макетов. Grid, в свою очередь, позволяет создавать сложные многоколоночные структуры, которые легко адаптируются под разные размеры экранов.

Благодаря гибким сеткам, элементы интерфейса могут менять свои размеры и расположение в зависимости от устройства пользователя. Например, на мобильных устройствах блоки могут выстраиваться вертикально, а на десктопах – располагаться в несколько колонок. Это обеспечивает удобство восприятия информации и улучшает пользовательский опыт.

**3. Bootstrap**

Для ускорения разработки и обеспечения высокой степени адаптивности в проекте используется популярный фреймворк Bootstrap. Этот инструмент предоставляет готовые компоненты и классы, которые автоматически адаптируются под разные размеры экранов, что значительно упрощает процесс создания отзывчивого дизайна.

Например, такие элементы, как навигационное меню, кнопки и карточки, уже имеют встроенные свойства адаптивности. Это позволяет им корректно отображаться на любых устройствах без необходимости написания дополнительного кода.

На больших экранах навигационное меню отображается в горизонтальном формате, что делает его удобным для использования. На мобильных устройствах оно может сворачиваться в "гамбургер"-меню, что экономит место на экране и улучшает юзабилити.

Использование Bootstrap не только ускоряет процесс разработки, но и обеспечивает высокое качество и стабильность работы интерфейса на всех устройствах.

**4. Оптимизация изображений и медиа-контента**

Для обеспечения корректного отображения изображений на всех устройствах в портфолио используется адаптивная верстка. Изображения настроены таким образом, чтобы автоматически масштабироваться в зависимости от ширины экрана. Это достигается за счет применения CSS-свойств, таких как max-width: 100%, которые позволяют изображениям адаптироваться к размерам контейнера

Такой подход предотвращает искажение изображений и сохраняет их качество на любых устройствах – от смартфонов до настольных компьютеров. Это особенно важно для визуальной привлекательности портфолио, так как изображения играют ключевую роль в демонстрации проектов и работ автора.

**5. Адаптивные шрифты**

Шрифты на сайте также адаптируются под размер экрана с помощью медиазапросов. Например, заголовки могут иметь разные размеры на мобильных устройствах и десктопах, что обеспечивает удобочитаемость и комфортное восприятие текста.

Использование адаптивных шрифтов позволяет сохранить визуальную иерархию и структуру контента, независимо от того, с какого устройства пользователь просматривает сайт. Это делает портфолио более профессиональным и удобным для использования.

**6. Тестирование на различных устройствах и браузерах**

Для обеспечения кроссбраузерной совместимости и корректного отображения интерфейса портфолио было протестировано на различных устройствах и браузерах. Это включает в себя проверку работы сайта на смартфонах, планшетах, ноутбуках и настольных компьютерах.

Для тестирования использовались инструменты разработчика в браузерах, которые позволяют эмулировать различные устройства и проверять адаптивность интерфейса. Это помогает выявить и устранить возможные ошибки, связанные с отображением контента на разных экранах.

Адаптивная верстка портфолио успешно реализована с использованием медиазапросов, 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

Данный проект представляет собой веб-портфолио, макет которого был разработан в Figma. В макете учтены все основные элементы интерфейса, такие как заголовки, изображения, кнопки и информационные блоки, которые были точно перенесены в код. Использование Figma значительно упростило процесс работы с цветовыми палитрами, типографикой и отступами, что позволило достичь высокой точности в реализации дизайна.

Каждый элемент интерфейса был тщательно воспроизведен в коде, что обеспечило полное соответствие оригинальному макету. Figma также помогла в организации работы с компонентами, что сделало процесс разработки более структурированным и эффективным.

HTML-код проекта организован с использованием семантических тегов, таких как <header>, <main>, <section> и <footer>. Это не только улучшает читаемость кода, но и способствует повышению доступности и SEO-оптимизации страницы.

Навигационная панель, включенная в проект, позволяет пользователям быстро перемещаться между разделами портфолио. Это делает взаимодействие с сайтом более удобным и интуитивно понятным.

Для стилизации проекта использовался Sass – мощный препроцессор CSS, который позволяет задавать цвета, шрифты и отступы в соответствии с дизайном, созданным в Figma. Применение Sass значительно упростило процесс стилизации, предоставив возможность использовать такие функции, как переменные, вложенные селекторы и миксины. Это сделало код более модульным, легко поддерживаемым и масштабируемым. Например, переменные для цветов и шрифтов позволяют быстро вносить изменения в дизайн, не затрагивая весь код.

Сайт использует адаптивную верстку, что обеспечивает корректное отображение контента на различных устройствах – от мобильных телефонов до настольных компьютеров. Для реализации адаптивности применены медиазапросы, которые позволяют изменять стили в зависимости от ширины экрана.

В проекте активно применяются базовые техники SEO, направленные на повышение видимости сайта в поисковых системах. Эти теги помогают поисковым системам лучше понимать содержание страницы, что способствует улучшению ее ранжирования в результатах поиска.

# 4.2 Реализация приложения для управления каталогом продуктов с применением HTML, CSS, JavaScript, React, React-Redux и Redux-Toolkit

Данный проект представляет собой приложение для управления каталогом продуктов, разработанное с использованием современного стека технологий: HTML, CSS, JavaScript, React, React-Redux и Redux Toolkit. В приложении реализованы базовые функции управления продуктами, такие как добавление новых позиций, редактирование существующих, удаление ненужных и просмотр полного списка товаров.

Функциональность приложения поддерживается за счет управления состоянием с помощью Redux Toolkit, что обеспечивает предсказуемость и удобство работы с данными. Для взаимодействия между компонентами и состоянием используется React-Redux, что делает код более структурированным и легко поддерживаемым.

**Структура проекта и инструменты разработки**

Проект представляет собой приложение для управления каталогом продуктов, разработанное с использованием современных веб-технологий. Основная цель приложения – предоставить пользователю удобный интерфейс для добавления, редактирования, удаления и просмотра продуктов. Для реализации проекта использованы следующие технологии:

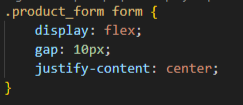
* HTML – язык разметки, который используется для создания структуры веб-страниц. В проекте HTML применяется для семантической разметки, что улучшает доступность и SEO-оптимизацию страницы.
* CSS – язык стилей, который отвечает за визуальное оформление интерфейса. В проекте CSS используется для создания адаптивного дизайна, анимаций и визуальных эффектов.
* JavaScript – язык программирования, который добавляет интерактивность и динамику на веб-страницы. В проекте JavaScript используется для реализации логики взаимодействия с пользователем.
* Node.js – среда выполнения JavaScript на стороне сервера, которая использовалась для настройки локального сервера разработки и управления зависимостями проекта.
* React – библиотека для создания пользовательских интерфейсов, которая позволяет разрабатывать компонентный и модульный код. В проекте React используется для создания компонентов, таких как карточки продуктов, формы добавления и редактирования.
* React-Redux – библиотека, которая обеспечивает интеграцию React с Redux, позволяя управлять состоянием приложения на глобальном уровне.
* Redux Toolkit – инструмент для упрощения работы с Redux, который предоставляет готовые решения для управления состоянием, такие как создание слайсов и настройка хранилища (store).

HTML и CSS

HTML: Используется для создания структуры компонентов. Например, в компонентах приложения AddProduct.js и UpdateProduct.js используются HTML-формы.



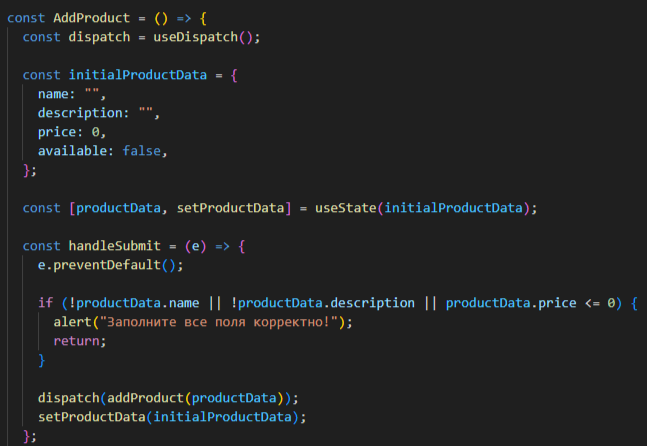
CSS: Используется для стилизации компонентов. Например, в my\_style.css задаются стили для элементов интерфейса.



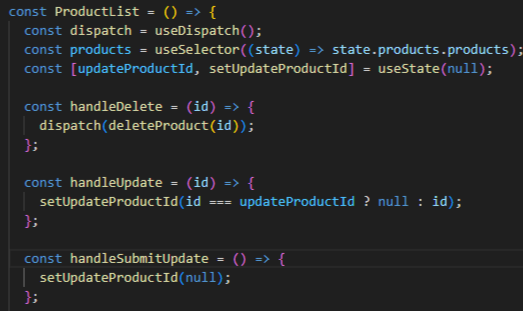
React используется для создания компонентов и управления их состоянием. В вашем приложении React применяется следующим образом:

Компоненты:

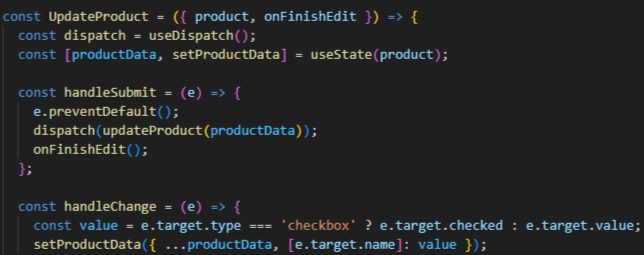
* AddProduct: Компонент для добавления нового продукта.



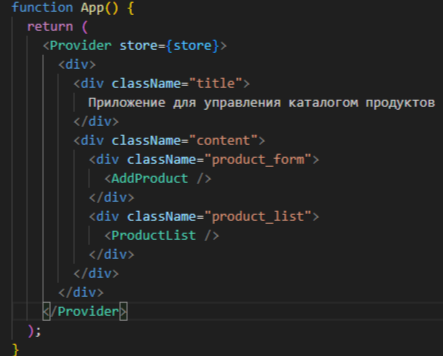
* + useState: Управляет состоянием формы (название, описание, цена, доступность).
  + useDispatch: Отправляет действие addProduct для добавления продукта в глобальное состояние.
* ProductList: Компонент для отображения списка продуктов.



* + useSelector: Получает список продуктов из глобального состояния.
  + useDispatch: Отправляет действие deleteProduct для удаления продукта.
* UpdateProduct: Компонент для редактирования продукта.



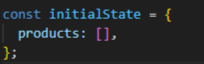
* + useState: Управляет состоянием формы редактирования.
  + useDispatch: Отправляет действие updateProduct для обновления продукта.
* App: Главный компонент, который объединяет все остальные компоненты.



* + Provider: Предоставляет доступ к хранилищу Redux для всех компонентов.

Redux используется для управления глобальным состоянием приложения. В вашем проекте Redux применяется следующим образом:

* Состояние продуктов хранится в глобальном состоянии Redux.



Обновление состояния

* Действия (actions) и редьюсеры (reducers) используются для обновления состояния.
  + addProduct: Добавляет новый продукт в список.
  + deleteProduct: Удаляет продукт из списка.
  + updateProduct: Обновляет данные продукта.



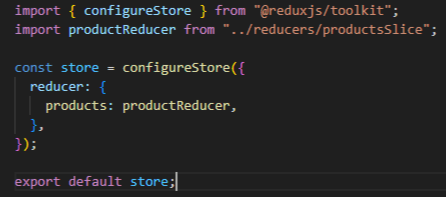
Redux Toolkit упрощает работу с Redux. В вашем проекте он используется для:

Создания слайсов

* В файле productsSlice.js создан слайс для управления состоянием продуктов.

Настройки хранилища

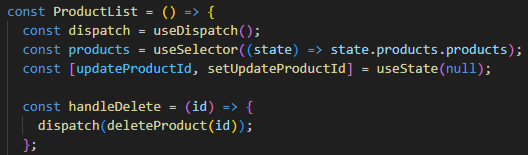
* В файле store.js создается хранилище Redux.



React-Redux связывает React с Redux. В вашем проекте он используется для:

Доступа к состоянию

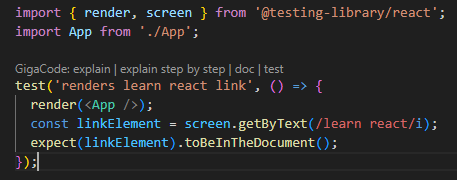
* Хук useSelector позволяет компонентам получать данные из хранилища.



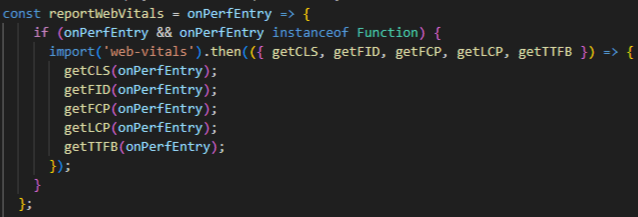
Диспетчеризации действий

* Хук useDispatch позволяет компонентам отправлять действия для обновления состояния.

В файле App.test.js используется библиотека @testing-library/react для тестирования компонента App.



В файле reportWebVitals.js используется библиотека web-vitals для отслеживания метрик производительности.



В моем приложении:

* React используется для создания компонентов и управления локальным состоянием.
* Redux – для управления глобальным состоянием (списком продуктов).
* Redux Toolkit – для упрощения работы с Redux (создание слайсов и настройка хранилища).
* React-Redux – для интеграции React с Redux (доступ к состоянию и диспетчеризация действий).
* HTML и CSS – для создания структуры и стилизации интерфейса.
* Тестирование – для проверки корректности работы приложения.
* Web Vitals – для отслеживания производительности.

Этот стек технологий позволяет создавать мощные, масштабируемые и легко поддерживаемые приложения.

# 5. ТЕСТИРОВАНИЕ И АНАЛИЗ ПРОЕКТА

Данный раздел посвящен тестированию приложения на различных устройствах и браузерах, а также анализу полученных результатов. Тестирование позволяет убедиться, что приложение корректно отображается и функционирует на всех поддерживаемых платформах, соответствует современным стандартам разработки и обеспечивает комфортный 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 <http://fyodartemm.temp.swtest.ru/> 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 <http://fyodartemm.temp.swtest.ru/> были рассмотрены и успешно внедрены современные технологии для создания интерактивного и адаптивного веб-приложения для управления каталогом продуктов. Проект разрабатывался с использованием HTML5, CSS3, JavaScript, а также библиотек React и Redux Toolkit, что позволило не только создать удобный и привлекательный интерфейс, но и обеспечить высокую производительность, кроссбраузерную совместимость и эффективное управление состоянием приложения.

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

1. Применение современных веб-технологий, таких как React, способствует улучшению структуры кода и упрощает его поддержку в дальнейшем. Размещение проекта на SpaceWeb предоставляет удобный способ публикации и обновления сайта, а также предлагает бесплатный хостинг, что особенно выгодно для личных проектов и портфолио.
2. Проект был протестирован на различных устройствах и браузерах, что подтвердило его корректную работу и адаптивность. Все поставленные цели были успешно достигнуты: приложение получилось полностью адаптивным, визуально привлекательным и соответствующим современным стандартам веб-разработки.
3. Разработанный проект имеет практическую ценность, так как может служить личным портфолио для демонстрации навыков и компетенций в веб-разработке. Размещение проекта в сети Интернет делает его доступным для просмотра потенциальными работодателями или клиентами, что упрощает процесс презентации профессиональных умений. Использованные в проекте технологии обеспечивают гибкость для дальнейшей доработки и масштабирования, включая добавление новых страниц, интеграцию API или подключение баз 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, 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:
   * 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 продемонстрировать навыки и знания в области фронтенд-разработки, а также подтвердить, что применение современных фреймворков и инструментов значительно упрощает процесс создания удобных и адаптивных интерфейсов. Реализация проекта стала ценным опытом для дальнейшего профессионального роста в сфере веб-разработки и создания более сложных и функциональных приложений.

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Официальная документация React: [**https://react.dev/**](https://react.dev/)
2. Бёрч, Д. HTML5 и CSS3. Полное руководство для начинающих. М.: Вильямс, 2019.
3. Вассерман Эрик. React.js: Быстрый старт. М.: Эксмо, 2020.
4. Ли Бенджамин. Sass и Compass. Введение в CSS-препроцессоры. СПб: Питер, 2018.
5. Флэнаган Дэвид. JavaScript: Подробное руководство. 6-е изд. СПб.: Символ-Плюс, 2019.
6. Freeman Eric, Robson Elisabeth. Head First HTML and CSS. O'Reilly Media, 2012.
7. Официальная документация Bootstrap: [**https://getbootstrap.com/docs**](https://getbootstrap.com/docs)
8. 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)**.**
9. Документация библиотеки Animista: [**Animista - CSS Animations On Demand**](https://animista.net/).
10. Руководство по Vite: [**https://vite.dev/guide/**](https://vite.dev/guide/)