Министерство образования и науки Пермского края

ГБПОУ «Пермский краевой колледж «Оникс»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**ДИПЛОМНЫЙ ПРОЕКТ**

Маркетплейс по продаже товаров

Угольников Даниил Олегович

специальность 09.02.07

«Информационные системы и программирование»

курс 4, группа 2-ИС

форма обучения: очная

Руководитель:

Ведерникова Ирина Дмитриевна

преподаватель ГБПОУ «Пермского

краевого колледжа «Оникс»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

К защите допущен

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2025 г.

Работа защищена с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Протокол ГЭК № \_\_\_\_ от «\_\_\_» \_\_\_\_\_\_ 2025 г.

Председатель ГЭК \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / Л.П.Смирнова /

Пермь 2025

**Оглавление**

[Введение 3](#_Toc199924188)

[1. Понятие маркетплейса и способы их разработки 5](#_Toc199924189)

[2. Программное обеспечение для разработки веб-приложения 8](#_Toc199924190)

[3. Разработка маркетплейса 11](#_Toc199924191)

[4. Тестирование веб-приложения 17](#_Toc199924192)

[5. Рекомендации по сопровождению 19](#_Toc199924193)

[Список литературы 21](#_Toc199924194)

[Приложение 23](#_Toc199924195)

Введение

В последние годы электронная коммерция стремительно развивается, и маркетплейсы становятся важной частью этой трансформации. Эти платформы обеспечивают удобство для продавцов и покупателей, предлагая широкий ассортимент товаров и услуг в одном месте, что значительно упрощает процесс покупки и продажи. На фоне растущего спроса на онлайн-торговлю, создание эффективного маркетплейса становится актуальной задачей для многих компаний, стремящихся выйти на новый уровень и расширить аудиторию.

Актуальность исследования функциональных возможностей маркетплейсов обусловлена их растущей популярностью среди пользователей и предпринимателей. Идея маркетплейсов берет свое начало с первых онлайн-аукционов, таких как eBay, запущенный в 1995 году, а затем получила развитие с появлением крупных платформ, ориентированных на широкий спектр товаров и услуг. Со временем маркетплейсы стали не просто посредниками между продавцом и покупателем, а полноценными экосистемами, объединяющими логистику, финансы и аналитику, что значительно упростило ведение бизнеса.

Создание собственного маркетплейса становится особенно актуальным для компаний, стремящихся расширить аудиторию и автоматизировать процессы продаж. По данным исследований, глобальный рынок электронной коммерции продолжает расти: в 2024 году его объем превысил 6,3 трлн долларов, а маркетплейсы занимают более 60% от всех онлайн-продаж. В России доля маркетплейсов также увеличивается: например, оборот Wildberries в 2024 году вырос на 70% по сравнению с предыдущим годом. Это подтверждает высокий спрос на подобные платформы, их значительное влияние на рынок и необходимость для компаний адаптироваться к новым условиям цифровой торговли [15].

Современные маркетплейсы предлагают широкий спектр функциональных возможностей, включая удобные системы регистрации, интеллектуальные поисковые механизмы с фильтрацией, инструменты для управления товарами и заказами, а также интеграцию с платёжными сервисами и логистическими компаниями. Внедрение новых технологий, таких как искусственный интеллект и анализ больших данных, позволяет маркетплейсам персонализировать пользовательский опыт, прогнозировать спрос и повышать эффективность работы платформ. Кроме того, поддержка мобильных приложений и API делает маркетплейсы еще более гибкими и удобными для пользователей и бизнеса.

Исходя из этого был определен исследовательский аппарат:

Проблема: Как разработать маркетплейс?

Цель: Разработка маркетплейс-площадки для компании «ruMarket» по продаже товаров

Задачи:

1. Изучить информацию о маркетплейсах, их разработке, дать определения технологиям для разработки.
2. Проанализировать программное обеспечение для разработки веб-приложения.
3. Разработать маркетплейс для компании «ruMarket».
4. Провести тестирование готового продукта.
5. Разработать рекомендации по использованию маркетплейса и сопровождению.

Готовый продукт будет сдержать: Современное веб-приложение, реализующее функционал маркетплейса.

1. Понятие маркетплейса и способы их разработки

Маркетплейс – это цифровая платформа, которая объединяет продавцов и покупателей, обеспечивая удобные условия для торговли товарами и услугами. В отличие от обычных интернет-магазинов, где один продавец управляет всей витриной, маркетплейс позволяет множеству продавцов размещать свои товары, а сама платформа выступает в роли посредника, предоставляя инфраструктуру для поиска, оплаты и доставки [13].

Существует несколько классификаций маркетплейсов [22]:

1. По типу участников:

* B2C (business-to-consumer) – платформа связывает компании и конечных покупателей (Ozon, Яндекс.Маркет, Wildberries).
* B2B (business-to-business) – предназначена для сделок между бизнесами (Alibaba, Kompass).
* C2C (consumer-to-consumer) – позволяет пользователям продавать товары друг другу (Avito, eBay).

1. По ассортименту:

* Универсальные – включают товары разных категорий (Amazon, Ozon).
* Нишевые – ориентированы на конкретную отрасль (Lamoda – мода, Booking – аренда жилья).

1. По модели монетизации:

* Комиссия с продаж.
* Платные подписки для продавцов.
* Размещение рекламных объявлений.

Способы разработки маркетплейсов.

Процесс создания маркетплейса может быть реализован разными методами в зависимости от потребностей бизнеса [24]:

1. Готовые решения – используют платформы вроде CS-Cart, WordPress (WooCommerce), Shopify. Этот вариант подходит для небольших проектов, но имеет ограничения в кастомизации.
2. Разработка с нуля – создание уникальной платформы с использованием современных технологий, обеспечивающих гибкость, масштабируемость и безопасность.
3. Гибридный подход – комбинация готовых решений с дополнительными модулями или API-интеграцией.

Для разработки проекта использован подход проектирования с нуля, так как возможность организации большего спектра инструментов и библиотек, чем в готовых решениях.

Определение технологий

Фреймворк — это готовая архитектурная основа для разработки программных решений, предоставляющая разработчику набор инструментов, стандартов и библиотек, упрощающих создание сложных приложений за счёт повторного использования проверенных решений и структурирования кода.

Библиотека — это набор предопределённых функций и классов, упакованных в единое решение для использования в разработке программного обеспечения. Библиотека предоставляет разработчику инструменты для выполнения специфических задач (например, обработки данных, работы с сетью или интерфейсами) без необходимости разработки этих функций с нуля.

База данных — это организованная совокупность данных, которые хранятся и управляются с помощью системы управления базами данных (СУБД). База данных обеспечивает эффективное хранение, поиск и обновление информации, а также поддерживает операции с большими объёмами данных, обеспечивая целостность и безопасность.

Выбор технологий и обоснование их использования

При создании маркетплейса использован стек современных технологий, обеспечивающих надежность, производительность и удобство разработки:

* Next.js 13+ (App Router) - современный React-фреймворк выбран как основа frontend-части.
* NextAuth.js - cистема аутентификации, обеспечивающая: гибкую систему авторизации с поддержкой ролей, безопасное хранение сессий, защита маршрутов на уровне middleware.
* Prisma ORM - современная виртуальная объектная база данных.
* PostgreSQL - надежная реляционная база данных.
* Supabase - платформа для backend-as-service.
* DaData API – сервис для валидации данных.
* Axios – библиотека для HTTP запросов.
* Tailwind CSS – утилитарный CSS-фреймворк.

Данный стек технологий обеспечивает:

* Высокую производительность
* Безопасность данных
* Масштабируемость приложения
* SEO-оптимизацию

В результате проведенного анализа была определена структура будущего продукта (Приложение 5).

1. Программное обеспечение для разработки веб-приложения

Программное обеспечение (ПО) — программа или множество программ, используемых для управления компьютером. Инструментальное программное обеспечение — программное обеспечение, предназначенное для использования в ходе проектирования, разработки и сопровождения программ. Обычно этот термин применяется для акцентирования отличия данного класса ПО от прикладного и системного программного обеспечения [25].

Виды программного обеспечения можно использовать для разработки:

1. Visual Studio Code — это бесплатный, открытый текстовый редактор, разработанный Microsoft, с поддержкой множества языков программирования, плагинов и встроенной интеграцией с Git. Он предлагает множество расширений для различных задач, таких как отладка и форматирование кода [7] (Приложение 6).
2. JetBrains WebStorm— это интегрированная среда разработки (IDE) от компании JetBrains, предназначенная для разработки веб-приложений на JavaScript, HTML, CSS и других веб-технологиях, с мощными инструментами для отладки, рефакторинга и тестирования кода [6] (Приложение 7).
3. Trae IDE — это интегрированная среда разработки от ByteDance с поддержкой ИИ, предназначенная для повышения продуктивности разработчиков. Она предлагает функции редактирования кода, управления проектами, расширений и контроля версий, а также интеграцию с GitHub. Trae IDE поддерживает большинство расширений Visual Studio Code и обеспечивает плавный переход с других редакторов [1] (Приложение 8).
4. Notepad++ — это бесплатный текстовый редактор с открытым исходным кодом, поддерживающий работу с множеством языков программирования и обладающий возможностями подсветки синтаксиса, поиска и замены текста, а также расширяемый с помощью плагинов [3] (Приложение 9).
5. Sublime Text — это текстовый редактор с высокой производительностью, предназначенный для разработки программного обеспечения, поддерживающий множество языков программирования и обладающий возможностями расширения через плагины и настройку пользовательских параметров [5] (Приложение 10).

Сравнение программного обеспечения:

1. Visual Studio Code:

* Преимущества: Множество плагинов, легкость, поддержка многих языков, интеграция с Git, регулярные обновления [8].
* Недостатки: Ограниченные возможности для больших проектов, ограниченная отладка.

1. JetBrains WebStorm:

* Преимущества: Полная интеграция с инструментами для JavaScript, поддержка рефакторинга, мощная отладка, высокая производительность [9].
* Недостатки: Высокая стоимость, требует больше системных ресурсов.

1. Trae IDE:

* Преимущества: Поддерживает бесплатную интеграцию с передовыми ИИ-моделями (GPT-4o и Claude-3.5-Sonnet), поддерживает мультимодальный ввод и режим Builder для пошаговой генерации кода, а также совместима с расширениями VS Code, что делает её мощным и доступным инструментом для ускоренной разработки [10].
* Недостатки: Требует постоянного интернет-соединения для работы с облачными ИИ-моделями и недоступна с российских IP-адресов, что ограничивает её использование для части разработчиков.

1. Notepad++:

* Преимущества: Легковесность, множество плагинов, поддержка множества языков, бесплатность [17].
* Недостатки: Ограниченная функциональность для крупных проектов, нет встроенной отладки.

1. Sublime Text:

* Преимущества: Высокая производительность, поддержка множества плагинов, красивый интерфейс [19].
* Недостатки: Ограниченные возможности по сравнению с более специализированными редакторами, платная версия.

Trae IDE была выбрана из-за преимущества в наличии интеллектуальных функций, таких как встроенный AI-ассистент и режим Builder, поддержку мультимодального ввода и интеграцию с GitHub, что обеспечивает более интеллектуальную и эффективную разработку.

При разработке так же необходимо использовать систему контроля версий Git, и приложение GitHub Desktop для упрощения контроля изменений (Приложение 11).

1. Разработка маркетплейса

Для разработки веб-приложения разработчику необходимо техническое задание, согласованное с заказчиком. Техническое задание — это официальный документ, в котором подробно описаны цели, задачи, функциональные и нефункциональные требования, этапы разработки и критерии приемки проекта [27]. Техническое задание служит основой для организации и контроля процесса разработки, а также обеспечивает единое понимание проекта всеми участниками: заказчиком, разработчиками и тестировщиками (Приложение 4).

В нём сформулированы цели проекта — создание функциональной площадки для эффективного взаимодействия продавцов и покупателей — и определён объём функциональности: управление учётными записями, добавление и редактирование карточек товаров, поиск и фильтрация, оформление заказов. Особое внимание уделено требованиям к производительности и масштабируемости, обеспечению защиты пользовательских данных, а также адаптивности интерфейса для разных устройств. Чёткая структура технического задания позволяет контролировать ход разработки и гарантировать выпуск качественного продукта без отклонений от согласованных требований.

На начальном этапе были изучены современные требования к разработке маркетплейсов: высокая скорость загрузки, SEO-оптимизация, безопасность аутентификации и масштабируемость. На основании анализа был выбран используемый стек технологий. Также была разработана структура маршрутов, API-ссылок и компонентов, соответствующих различным разделам сайта (каталог, карточка товара, корзина, личный кабинет, страница продавца).

В качестве названия для маркетплейса было выбрано наименование «ruMarket», в честь той страны, в которой он был разработан.

При создании дизайна были разработаны несколько вариантов, отличающихся цветовой схемой:

1. Первый вариант - глубокий оттенок синего с фиолетовым акцентом (Приложение 12).
2. Второй вариант - жженый апельсин (выгоревший оранжевый) (Приложение 13).
3. Третий вариант – индийский зеленый (Приложение 14).

После изучения литературы был выбран дизайн, где глубокий оттенок синего с фиолетовым акцентом является основным цветом (#4438ca). Данный цвет был выбран по нескольким причинам: во-первых, синий традиционно ассоциируется с надёжностью и доверием, что критично для электронной коммерции, где пользователи совершают финансовые операции; во-вторых, лёгкий фиолетовый под тон вносит элемент креативности и современности, подчёркивает технологическую направленность проекта и наконец, этот цвет хорошо смотрится на светлом и тёмном фоне, обеспечивая контраст и читаемость интерфейсных элементов [16] (Приложение 12).

Архитектура веб-приложения

Во время проектирования веб-приложения для удобства и безопасности взаимодействия пользователей с платформой были выделены три основные роли:

* Гость — неавторизованный пользователь, который может просматривать товары и общую информацию о маркетплейсе, но не имеет доступа к оформлению заказов и личному кабинету.
* Авторизованный пользователь — зарегистрированный покупатель, который получает возможность оформлять заказы, сохранять избранные товары, просматривать историю покупок и управлять своими данными.
* Продавец — отдельная роль для пользователей, которым предоставлен доступ к личному кабинету продавца, с возможностью добавлять товары, управлять ассортиментом, отслеживать заказы.

Такое разделение позволило гибко настраивать права доступа и создать удобные пользовательские сценарии для каждой группы.

При разработке структуры проекта был выбран подход с использованием App Router из Next.js, который позволяет эффективно организовать маршруты и упрощает управление как страницами, так и API-ссылками (Приложение 15). Благодаря этой архитектуре удалось добиться более чистого и гибкого кода, а также лёгкости в поддержке и масштабировании приложения. В рамках этого подхода вся клиентская часть была размещена в папке «app», а серверная логика — в папке «api». Это позволило организовать чёткое разделение ответственности и улучшить читаемость кода (Приложение 24)

App Router обеспечил удобную маршрутизацию для различных пользовательских ролей, а также позволил эффективно управлять защищёнными и открытыми маршрутами. В папке app размещены все страницы и компоненты, которые обрабатывают клиентские запросы, взаимодействуя с сервером через API (Приложение 16).

В свою очередь, папка api содержит всю серверную логику, включая обработку запросов для аутентификации, работы с базой данных и взаимодействия с пользователями и продавцами. Такой подход позволил интегрировать серверную логику с клиентской частью, избавив от необходимости использовать отдельные серверы или сложные конфигурации. Это ускорило процесс разработки и повысило производительность приложения.

Например, в папке «api/auth» реализована вся логика, связанная с аутентификацией и управлением сессиями через NextAuth.js, что поддерживает гибкую авторизацию с учётом различных ролей. В папке «api/products» расположены конечные точки для работы с товарами, включая их добавление, редактирование и удаление продавцами. Все запросы к серверу обрабатываются через методы «GET, POST, PUT и DELETE», что соответствует принципам RESTful-архитектуры [4] (Приложение 17).

Такой подход позволил не только чётко разделить ответственность между клиентской и серверной частью, но и улучшил структуру проекта, сделав его более масштабируемым и удобным для дальнейшей разработки и поддержки.

Выбор фреймворка для стилизации

При разработке возникла необходимость использования фреймворка для стилизации. При принятии решения об использовании Tailwind CSS основополагающим фактором стала его парадигма utility-first, которая позволяет стилизовать элементы прямо в разметке, избавляя от постоянного переключения между HTML и CSS-файлами и ускоряя процесс разработки интерфейса [12] (Приложение 18).

Такой подход делает внесение изменений более безопасным и предсказуемым, поскольку каждая утилитная класс-инструкция влияет только на конкретный элемент, не затрагивая стили других частей приложения.

Кроме того, Tailwind CSS обеспечивает высокую консистентность дизайна: единая система утилитных классов и дизайн-токенов гарантирует, что все отступы, цвета и типографика будут строго соответствовать заданному проекту, что особенно важно для масштабируемого маркетплейса с многочисленными страницами и компонентами.

При этом обширная библиотека классов снижает объём пользовательского CSS и минимизирует необходимость в его написании, что поддерживает чистоту и читаемость кода (Приложение 19).

Важным преимуществом стала автоматическая очистка неиспользуемых стилей при сборке (purge), благодаря чему итоговый CSS-файл содержит только необходимые правила, что существенно снижает объём передаваемых данных и ускоряет загрузку страниц (Приложение 20).

Наличие режима JIT (Just-In-Time) дополнительно оптимизирует процесс разработки, генерируя утилиты по требованию и позволяя использовать произвольные значения прямо в классах без предварительной настройки в конфигурации.

Выбор хостинга и способа развертывания

При разработке маркетплейса «ruMarket» активно использовался GitHub с настройкой CI/CD для автоматического развертывания проекта на Vercel. Такой подход значительно ускорил процесс разработки, обеспечивая постоянную интеграцию и непрерывную доставку. Каждый новый коммит в репозиторий автоматически инициировал процесс сборки и развертывания, что позволило избежать ошибок при ручной загрузке и ускорить тестирование новых функций в продакшн-среде (Приложение 21). При разработке возникла необходимость в покупке именного домена для хостинга, что обеспечило большую запоминаемость URL-адресу веб-приложения. Веб-приложение располагается по постоянному URL-адресу: https://ugolnikov-do.ru/ (Приложение 25).

Использование CI/CD с GitHub и Vercel имеет несколько ключевых преимуществ. Во-первых, это автоматизация процессов сборки и развертывания, что снижает вероятность человеческой ошибки и ускоряет вывод новых изменений в продуктивную среду. Во-вторых, подход обеспечивает постоянную интеграцию, что позволяет разработчикам работать с актуальной версией кода и тестировать изменения в реальном времени. В-третьих, настройка загрузки на Vercel позволяет быстро и эффективно публиковать новые версии приложения с минимальными задержками, а также воспользоваться встроенными возможностями платформы для оптимизации производительности и масштабируемости.

Кроме того, GitHub предоставляет удобные инструменты для совместной работы в команде, такие как pull-requests, ревью кода и управление задачами, что позволяет поддерживать высокое качество кода и минимизировать количество ошибок на всех этапах разработки.

Решение проблемы с загрузкой изображений

Во время разработки маркетплейса появилась проблема с загрузкой изображений товаров: на платформе Vercel запрещено сохранять файлы на её файловой системе, поэтому прямая загрузка не работала. Чтобы обойти это ограничение, была интегрирована библиотека Supabase Storage, где организовал безопасное и масштабируемое хранение медиафайлов — теперь при загрузке картинка сразу уходит в облачное хранилище Supabase, а из интерфейса выводится полученная публичная ссылка. Таким образом удалось решить проблему с ограничениями Vercel и обеспечить удобный доступ к изображениям в приложении.

Интеграция с внешними сервисами

Для улучшения качества данных в маркетплейсе был использован сервис DaData, который предоставляет API для валидации и подгрузки информации о юридических лицах и индивидуальных предпринимателях. Это позволяло эффективно обрабатывать данные о компаниях, такие как наименование, адрес, регистрационные данные, а также проверять их корректность на основе ИНН. С помощью DaData при регистрации продавцов их данные автоматически проверялись на наличие ошибок, а также подгружалась актуальная информация по ИНН, что значительно улучшало точность данных в системе [11].

При разработке маркетплейса были использованы различные документации к библиотекам. В ходе работы, были изучены обучающие материалы по фреймворку Next.js и ORM Prisma [2].

1. Тестирование веб-приложения

Для безопасной и структурированной разработки веб-приложения необходимо проводить тестирования продукта при его реализации.

Тестирование — это процесс проверки программного обеспечения на соответствие требованиям, с целью выявления ошибок и обеспечения стабильной работы системы [26]. Существует несколько основных подходов к тестированию программного обеспечения:

* Модульное тестирование — проверка отдельных функций или компонентов программы.
* Интеграционное тестирование — проверка взаимодействия между модулями или компонентами.
* Системное тестирование — проверка всей системы целиком на соответствие требованиям.
* Приёмочное тестирование — проверка готовности продукта к передаче заказчику.
* Функциональное тестирование — проверка, соответствует ли программное обеспечение своим функциональным требованиям.
* Регрессионное тестирование — проверка, не нарушили ли новые изменения существующий функционал.

В рамках данного проекта основное внимание было уделено функциональному и регрессионному тестированию. Такой выбор обусловлен необходимостью обеспечить корректную работу ключевых пользовательских сценариев, а также стабильность приложения при внесении изменений в кодовую базу.

В процессе разработки веб-приложения для автоматизации тестирования были написаны тесты на языке программирования Python. Основными преимуществами Python [18] являются - простота синтаксиса, высокая читаемость, обширная стандартная библиотека, кроссплатформенность (Приложение 22).

В качестве библиотеки для тестирования была выбрана Pytest за её простоту, гибкость и широкие возможности для написания читаемых и масштабируемых тестов [23].

В качестве ядра для автоматизированного тестирования пользовательского интерфейса был выбран Selenium— инструмент, позволяющий управлять браузером и эмулировать действия пользователя [14]. Selenium обеспечивает кроссбраузерное тестирование, поддерживает различные языки программирования (включая Python) и хорошо интегрируется с такими библиотеками, как Pytest, что делает его оптимальным выбором для тестирования веб-интерфейсов маркетплейса.

Дополнительно были реализованы сценарии, охватывающие ключевые пользовательские действия: регистрация, авторизация, добавление товаров в корзину, оформление заказа и поиск по товарам. Это позволило выявить и устранить критические ошибки на ранних этапах разработки. Тесты регулярно запускались в процессе разработки, что способствовало поддержанию стабильности и качества веб-приложения.

Также была предусмотрена возможность расширения тестов при добавлении нового функционала, что делает систему тестирования гибкой и пригодной для дальнейшего развития проекта.

По итогам проведённого тестирования удалось достичь высокой стабильности и надёжности работы ключевого функционала веб-приложения. Автоматизация позволила существенно сократить время на проверку регрессионных ошибок при внесении изменений в код. Выявленные в ходе тестирования ошибки были оперативно устранены, что повысило общее качество пользовательского опыта. Наличие системы тестирования также создало основу для безопасного масштабирования и внедрения новых функций в будущем. (Приложение 23)

1. Рекомендации по сопровождению

В процессе эксплуатации информационной системы важно обеспечить её стабильную работу, удобство для пользователей и готовность к масштабированию. Для этого разрабатываются рекомендации по сопровождению программного продукта, а также рекомендации для конечных пользователей, которые помогают эффективно взаимодействовать с системой.

Рекомендации для пользователей

Рекомендации для пользователей представляют собой пошаговые инструкции по выполнению основных действий на сайте, адаптированные под разные роли: покупателя, продавца и администратора [20]. Эти инструкции помогают пользователям быстрее освоить функционал платформы, избежать типичных ошибок и повысить удобство работы с системой.

Пользовательские инструкции включают:

* порядок регистрации и входа в систему;
* алгоритм оформления заказа (для покупателя);
* добавление товаров и управление заказами (для продавца);
* контроль и настройка системы (для администратора).

Полные алгоритмы действий приведены в приложении:

* Приложение 1 – Алгоритм действий для покупателя
* Приложение 2 – Алгоритм действий для продавца
* Приложение 3 – Алгоритм действий для администратора

Рекомендации по сопровождению программного продукта

Сопровождение программного продукта включает технические мероприятия и регламенты, обеспечивающие его работоспособность, защиту и развитие [21]. Основные цели сопровождения:

* своевременное обнаружение и устранение ошибок;
* контроль безопасности данных и стабильности работы;
* адаптация системы под изменяющиеся требования (новые категории товаров, способы оплаты и т.д.);
* поддержка пользователей.

В качестве рекомендаций по сопровождению можно выделить:

* регулярное тестирование пользовательских функций (по алгоритмам, приведённым в приложениях);
* резервное копирование данных и базы данных;
* контроль за безопасностью (актуальные сертификаты SSL, защита от SQL-инъекций и XSS);
* поддержка пользователей и обработка обращений через техническую поддержку;
* обновление зависимостей и компонентов системы (фреймворки, библиотеки).

Список литературы

1. ByteDance представила Trae AI — новую IDE с интеграцией нейросетей: [сайт] – URL: <https://sky.pro/media/news/bytedance-predstavila-trae-ai-novuyu-ide-s-integracziej-nejrosetej/>
2. Framer Motion Documentation: [сайт] – URL: <https://motion.dev/docs>
3. Notepad ++, что это такое простыми словами: [сайт] – URL: <https://blog.skillfactory.ru/glossary/notepad/>
4. REST API, что это такое и как работает: [сайт] - URL: <https://practicum.yandex.ru/blog/chto-takoe-rest-api-i-kak-rabotaet/>
5. Sublime Text: что это за программа и где применяется: [сайт] – URL: <https://blog.skillfactory.ru/glossary/sublime-text/>
6. Trae IDE: ускоренная разработка с помощью ИИ: [сайт] – URL: <https://gptonline.ai/ru/trae-ai/>
7. Visual Studio Code: [сайт] – URL: <https://blog.skillfactory.ru/glossary/visual-studio-code/>
8. Visual Studio Code: что это и как пользоваться: [сайт] – URL: <https://blog.skillfactory.ru/glossary/visual-studio-code/>
9. WebStorm: это что и зачем?: [сайт] – URL: <https://thecode.media/webstorm/>
10. WebStorm: функциональность: [сайт] – URL: <https://www.jetbrains.com/ru-ru/webstorm/features/>
11. Документация DaData: [сайт] – URL: <https://dadata.ru/api/>
12. Документация TailWind CSS: [сайт] – URL: <https://tailwindcss.com/docs>
13. Как создать веб-приложение. 5 шагов к успешному запуску: [сайт] – URL: <https://surf.ru/razrabotka-veb-prilozhenij/>
14. Лучшие библиотеки Java для тестирования в 2023 году: [сайт] – URL: <https://habr.com/ru/companies/otus/articles/716444/>
15. Маркетплейсы, почему они актуальны и почему вы не должны их игнорировать: [сайт] – URL:  
    <https://tenchat.ru/media/1311513-marketpleysy-pochemu-oni-aktualny-i-pochemu-vy-ne-dolzhny-ikh-ignorirovat>
16. Надёжный синий: как цвет влияет на восприятие бренда: [сайт] – URL: <https://skillbox.ru/media/design/psihologia-sinego/>
17. Обзор Notepad++: возможности и особенности: [сайт] – URL: <https://sky.pro/wiki/javascript/obzor-notepad-vozmozhnosti-i-osobennosti/>
18. Преимущества и недостатки языка Python: [сайт] – URL: <https://www.hocktraining.com/blog/preimuschestva-yazyka-python>
19. Редактор кода Sublime Text — какие у него есть возможности и почему он популярен у разработчиков: [сайт] – URL: <https://ru.hexlet.io/blog/posts/sublime-text>
20. Рекомендательные системы и алгоритмы рекомендаций: [сайт] – URL: <https://www.uplab.ru/blog/rekomendatelnye-sistemy/>
21. Сопровождение программного обеспечения: [сайт] – URL: <https://www.sviaz-expo.ru/ru/ui/17042/>
22. Типология маркетплейсов: [сайт] – URL: <https://ru.wiki.rademade.com/4-typology-of-marketplaces>
23. Что такое PyTest: [сайт] – URL: <https://blog.skillfactory.ru/glossary/pytest/>
24. Что такое Selenium: [сайт] – URL: <https://blog.skillfactory.ru/glossary/selenium/>
25. Что такое разработка программного обеспечения?: [сайт] – URL: <https://sky.pro/wiki/profession/chto-takoe-razrabotka-programmnogo-obespecheniya/>
26. Что такое тестирование программного обеспечения: [сайт] – URL: <https://rt-solar.ru/events/blog/2955/>
27. Что такое техническое задание: [сайт] – URL: <https://secretmag.ru/enciklopediya/chto-takoe-tekhnicheskoe-zadanie-obyasnyaem-prostymi-slovami.htm>

Приложение

Приложение 1

Алгоритм действий покупателя:

1. Зарегистрироваться или войти в аккаунт
2. Перейти в каталог или воспользоваться поиском
3. Выбрать товар и перейти в его карточку
4. Нажать «Добавить в корзину»
5. Перейти в корзину
6. Проверить состав заказа
7. Нажать «Оформить заказ»
8. Ввести контактные и адресные данные
9. Оплатить заказ
10. Отслеживать статус в личном кабинете

Приложение 2

Алгоритм действий продавца:

1. Зарегистрироваться или войти в аккаунты
2. Пройти верификацию как продавца
3. Заполнить контактную информацию
4. Дождаться подтверждения от администратора
5. Войти в личный кабинет
6. Перейти в раздел «Товары»
7. Нажать «Добавить товар»
8. Заполнить название, описание, цену, фотографии
9. Указать единицу измерения
10. Опубликовать товар
11. Ожидать заказ
12. Перейти в раздел «Заявки»
13. Просмотреть новый заказ
14. Отправить товар
15. Подтвердить отправку товара в заявках
16. Обновить остатки товара (при необходимости)

Приложение 3

Алгоритм действий администратора:

1. Зарегистрироваться или войти в аккаунт администратора
2. Войти в админ-панель
3. Проверить новые заявки от продавцов
4. Подтвердить или отклонить заявки от продавцов
5. Перейти в раздел «Товары»
6. Проверить новые товары
7. Удалить или скрыть нарушающие правила
8. Перейти в раздел «Пользователи»
9. Управлять аккаунтами (удаление, изменение и т.д.)
10. Перейти в раздел «Заказы»
11. Проверить заказы
12. Обработать спорные ситуации
13. Проверить отчёты и статистику
14. Проверить наценку
15. Проверить логи
16. Сделать резервную копию базы
17. Обновить систему (если требуется)

Приложение 4

ГБПОУ «Пермский краевой колледж «ОНИКС»

УТВЕРЖДАЮ

Заказчик: директор, Рогова Марина Николаевна, ГБПОУ «Пермский краевой колледж «Оникс»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Рогова М.Н.

М.П.

02.12.2024

УТВЕРЖДАЮ

Исполнитель: студент(ка) группы 2-ИС Угольников Даниил Олегович специальность 09.02.07 «Информационные системы и программирование», ГБПОУ ПКК «Оникс»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_ / Угольников Д.О. /

02.12.2024

«ruMarket»

Техническое задание

на 11листах

Действует с 02.12.2024

СОГЛАСОВАНО

Заказчик: директор, Рогова Марина Николаевна, ГБПОУ «Пермский краевой колледж «Оникс»

\_\_\_\_\_\_\_\_\_\_\_\_\_ / Рогова М.Н. /

02.12.2024

**1) Вводные данные:**

1. Наименование информационной системы: «ruMarket»

2. Основание для выполнения информационной системы: договор от 02.12.2024

№ 0023;

3. Исполнитель информационной системы: студент(ка) группы 2-ИС Угольников Даниил Олегович

4. Срок исполнения информационной системы: с 01.12.2024 по 28.12.2024

**2) Цель выполнения** информационной системы**:** Целью разработки является создание удобного, масштабируемого веб-приложения для размещения товаров продавцами и их покупки пользователями с использованием современных технологий

**3) Технические требования и стандарты:**

1. **Требования по созданию информационной системой:**

* Использование Next.js для клиентской и серверной части
* Развертывание системы на платформе Vercel

1. **Рекомендации по работе с информационной системой:**

* Регулярное обновление компонентов для устранения уязвимостей
* Ежемесячное тестирование производительности

1. **Требования к надежности:**

* Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением Заказчиком совокупности организационно-технических мероприятий:   
  а) организацией бесперебойного питания технических средств;   
  б) использованием лицензионного программного обеспечения;
* Время восстановления после отказа, вызванного сбоем электропитания технических средств (иными внешними факторами), не фатальным сбоем (не крахом) операционной системы, не должно превышать 30-ти минут при условии соблюдения условий эксплуатации технических и программных средств.
* Время восстановления после отказа, вызванного неисправностью технических средств, фатальным сбоем (крахом) операционной системы, не должно превышать времени, требуемого на устранение неисправностей технических средств и переустановки программных средств.
* Отказы программы возможны вследствие некорректных действий пользователя при взаимодействии с операционной системой.

1. **Требования к безопасности:**

* Идентификация и аутентификация пользователей
* Разграничение доступа на основе ролей

1. **Требования к квалификации персонала:** Базовые навыки работы с веб-приложениями
2. **Требования к защите информации от несанкционированного доступа:**

– идентификация пользователя;

– проверка полномочий пользователя при работе с системой;

– разграничение доступа пользователей на уровне задач и информационных массивов.

**7. Требования к информационным структурам и методам решения:** Реляционная структура данных (MySQL)

**8. Требования к исходным кодам и языкам программирования:** Использование JavaScript / TypeScript.

**9. Требования к программным средствам, используемым программным продуктом:** Visual Studio Code, PhpStorm для разработки

**10. Технические требования:** Совместимость с последними версиями Chrome, Firefox, Edge

**11. Требования к защите информации и программ:** Использование HTTPS

**12. Требования к документированию:**Предварительный состав программной документации должен включать в себя:

1. техническое задание;   
2. руководство пользователя;

**4)** **Структурно-функциональная схема разработки   
информационной системы:**

Анкета-заявка

Анализ анкеты

Интервью с заказчиком

Пользовательский сценарий

Техническое задание

Проектирование

Разработка ИС

Тестирование ИС

Внедрение ИС

Адаптация ИС

**5) Состав ИС:**

1. **Функциональное содержание отдельных структурных элементов:**

* Prisma + Next API для обработки запросов
* Next.js для пользовательского интерфейса

1. **Листинг ИС:**

Next API: маршруты, контроллеры, модели

Next.js: страницы, компоненты

1. **Интерфейс ИС:** Современный адаптивный интерфейс с поиском

**6) Стадии и этапы разработки**

**Стадии разработки:**

1. разработка технического задания;   
2. рабочее проектирование;   
3. внедрение.

**Этапы разработки:**

1. Описание предметной области

* Краткое введение в предметную область
* Выделить элементы предметной области, их взаимосвязи
* Определить особенности и ограничения предметной области
* Используемые термины и сокращения

2. Цель создания системы

* Сформулировать цель создания
* Назначение системы, существующие аналоги
* целевая аудитория, ожидаемый уровень использования

3. Детализация функций системы

* Изучение потребностей заказчика
* Подготовить описание функций системы

4. Анализ категорий пользователей

* Выделение категорий пользователей
* Определение функциональных требований пользователей каждой категории

5. Определение ограничений

* Анализ аппаратных особенностей и ограничений
* Анализ топологии и особенностей развертывания
* Определение технологических ограничений

6. Формирование и утверждение совокупного списка требований к системе

* Если система предполагает интерактивность в общении с пользователем, то определить функциональные требования (описывают в динамике сценарии взаимодействия посетителя с системой) и структуру данных.
* Выделить специфические требования
* Прочие требования
* Сформировать список требований

7. Выработка архитектурного решения

* Выбор технологической платформы
* Специфическая бизнес-логика должна быть задокументирована в техническом задании максимально подробно.
* Подготовка модульной структуры системы
* Подготовка детализированного описания подсистем

8. Подготовка календарного плана

* Оценка сложности реализации подсистем.
* Выделение работ, построение сетевого графика.
* Оценка сроков выполнения работ

9. Завершающий этап

* Согласование процесса приемки работ
* Компоновка из полученных материалов текста технического задания

7) **Стоимость работы:**

Стоимость ИС можно рассчитать по формуле

Сс/с = Сэ/э + Сам + Ср + С пр

1. Стоимость затраченной электрической энергии (Сэ/э)

При создании **ИС** использовался ПК, вследствие чего расходовалась электроэнергия. Время работы (t) на ПК – 160часов.   
Мощность (Р) ПК – **0,6** кВт. Цена 1 кВт(Ч) – **3,99** руб.

Стоимость электроэнергии равна:

Сэ/э= Р \* t \* Ч = 383.04 руб.

2. Амортизация оборудования (Сам)

Для того чтобы посчитать амортизацию ПК нужно знать:

1. Стоимость ПК (Sпк);

Sпк = 20000 руб.

2. Амортизацию ПК (полный износ в течение 1 года):

Количество часов работы на ПК в день – (Ч); Ч = 10 часов

Количество дней работы на ПК – (Д); Д = 6 дня

Количество недель – (Н); Н = 4 недели

Количество месяцев – (М); М = 9 месяцев

Количество лет – (Л); Л = 5 лет

Подсчитываем количество часов работы на ПК в течении 5 лет (К):

К = Ч \* Д \* Н \* М \* Л = 10 \*6\* 4 \* 9 \* 5 =10800 часов

Чтобы найти амортизацию (А) ПК надо стоимость ПК разделить на количество часов работы в течение 5 лет:

А = Sпк / К = 20000/10800= 1.85 руб./час

3. Находим амортизацию ПК во время работы над программным продуктом:

Сам = А \* t = 296 **руб.**

3. Стоимость работы (Ср)

S 1 час – 200 руб.

t – 160 часов

Ср = S 1 час \* t = 32000руб.

4. Прочие расходы (С пр): отсутствуют

1. Себестоимость ИС:

Сс/с = Сэ/э + Сам + Ср + *С* пр = 32 679.04 руб.

**8) Порядок контроля и приемки**

1. Виды испытаний

* Приемо-сдаточные испытания должны проводиться на объекте Заказчика в оговоренные сроки.
* Приемо-сдаточные испытания программы должны проводиться согласно разработанной Исполнителем и согласованной Заказчиком Программы и методик испытаний.
* Ход проведения приемо-сдаточных испытаний Заказчик и Исполнитель документируют в Протоколе проведения испытаний

2. Общие требования к приемке работы

* На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывает Акт приемки-сдачи программы в эксплуатацию.

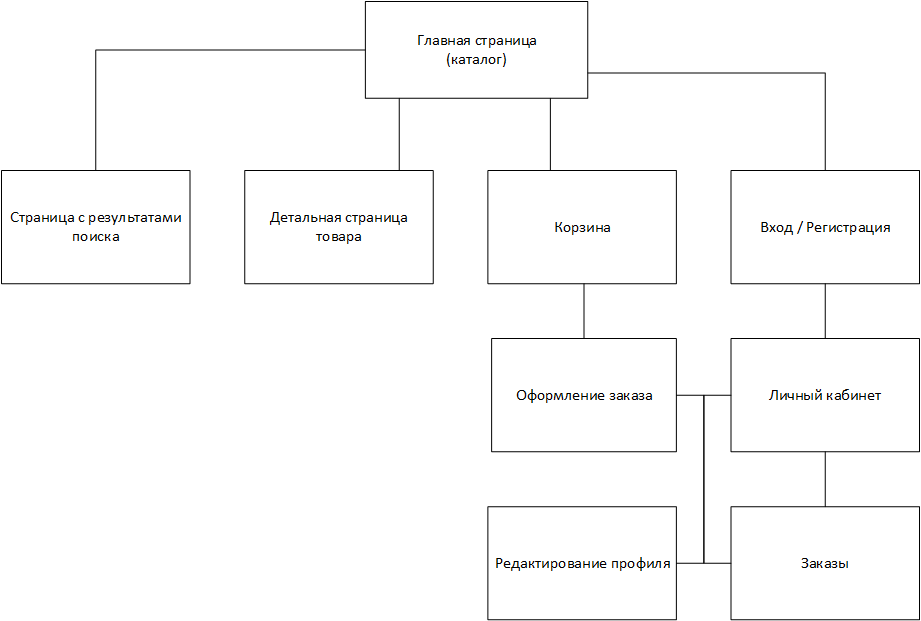
СОСТАВИЛИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование организации, предприятия | Должность  исполнителя | Ф.И.О. | Подпись | Дата |
| ГБПОУ «Пермский краевой колледж «Оникс» | Студент(ка) группы 2-ИС | Угольников Даниил Олегович | Угольников Д.О. | 02.12.2024 |

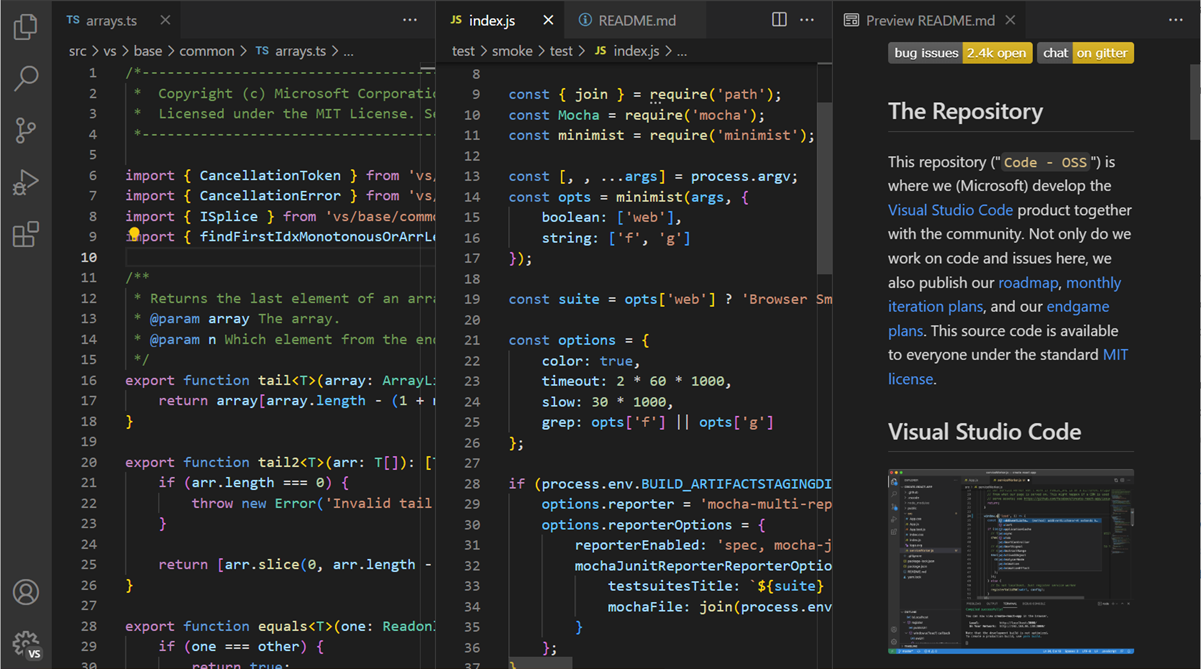
СОГЛАСОВАНО

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование организации, предприятия | Должность  заказчика | Ф.И.О. | Подпись | Дата |
| ГБПОУ «Пермский краевой колледж «Оникс» | Директор | Рогова Марина Николаевна | Рогова М.Н. | 02.12.2024 |

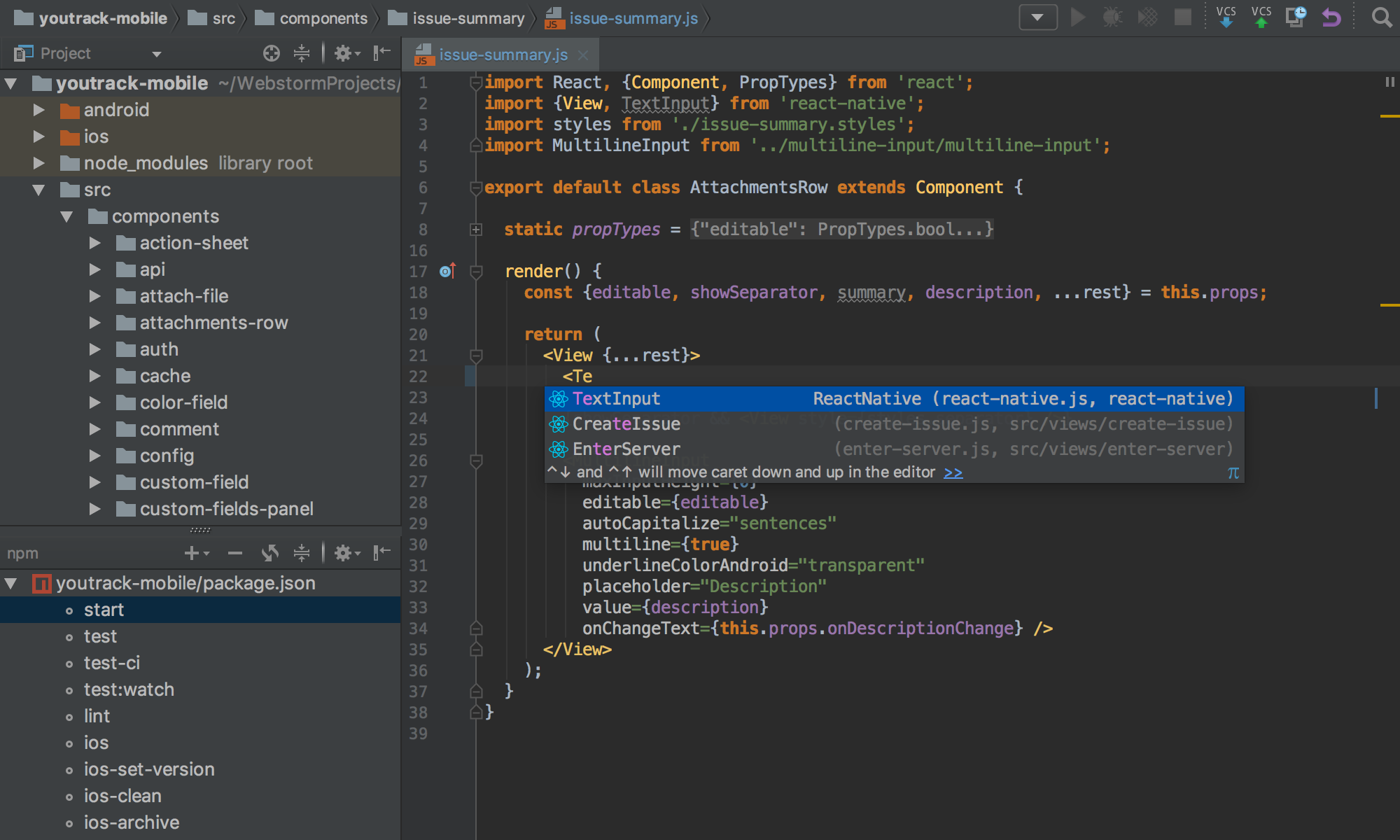
Приложение 5



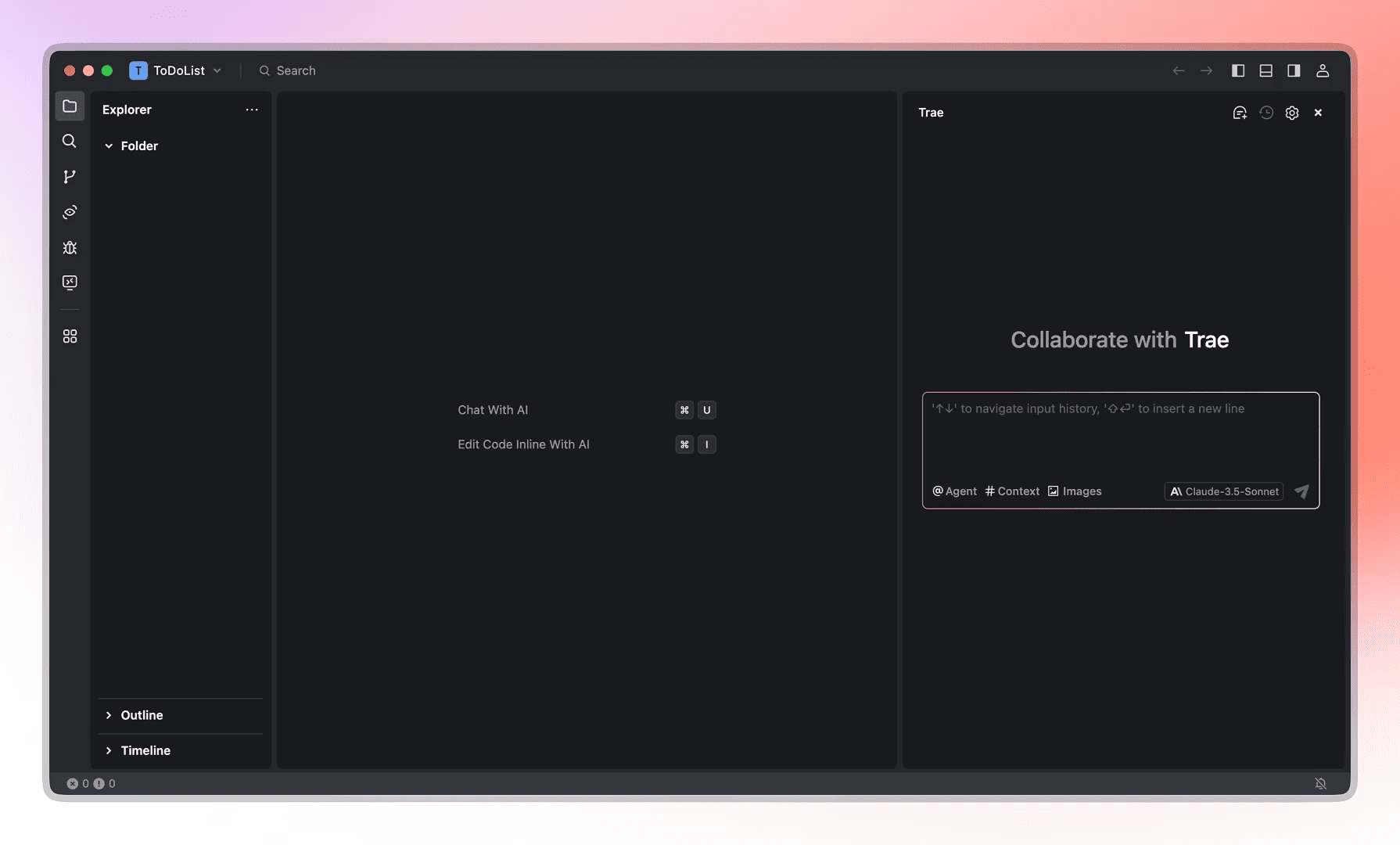
Приложение 6



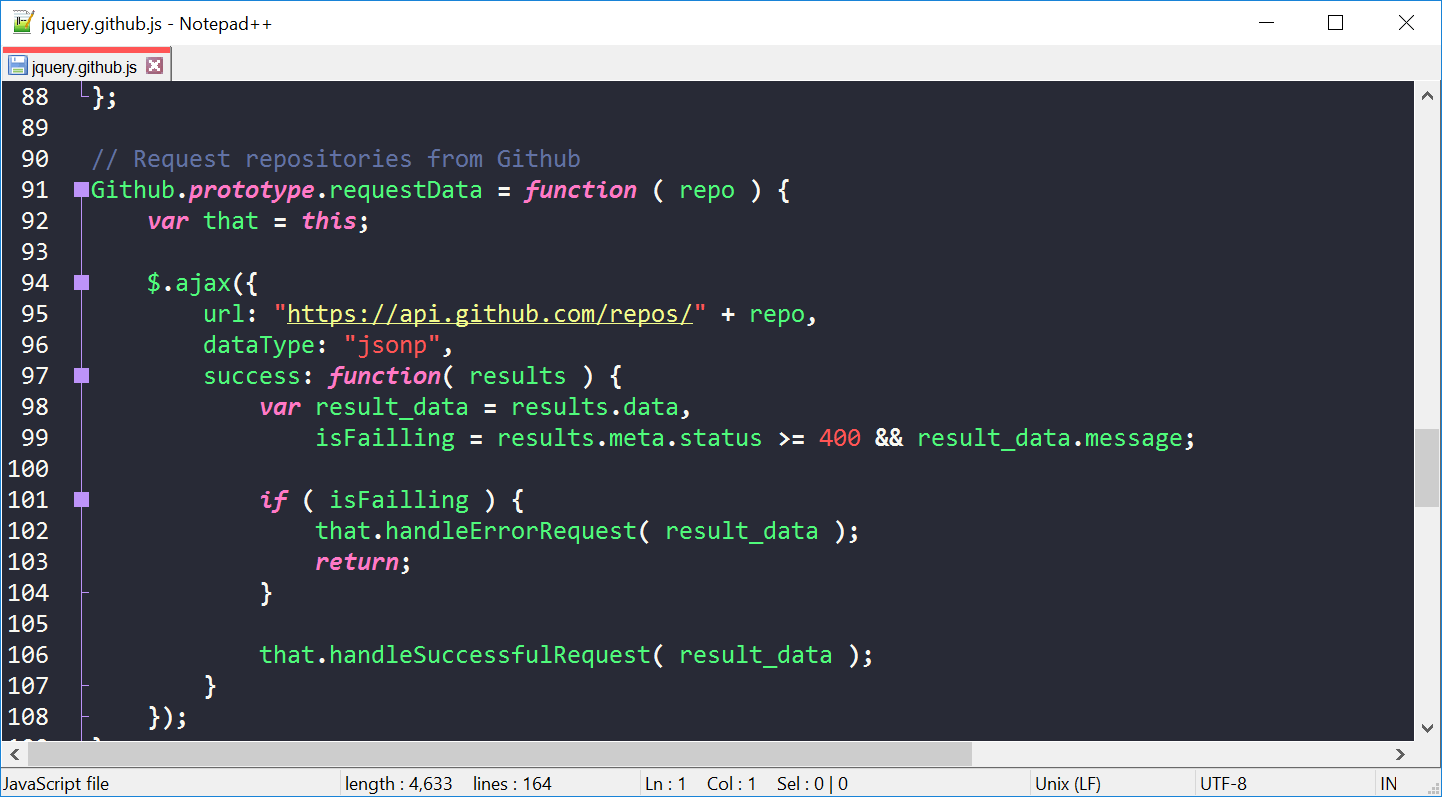
Приложение 7



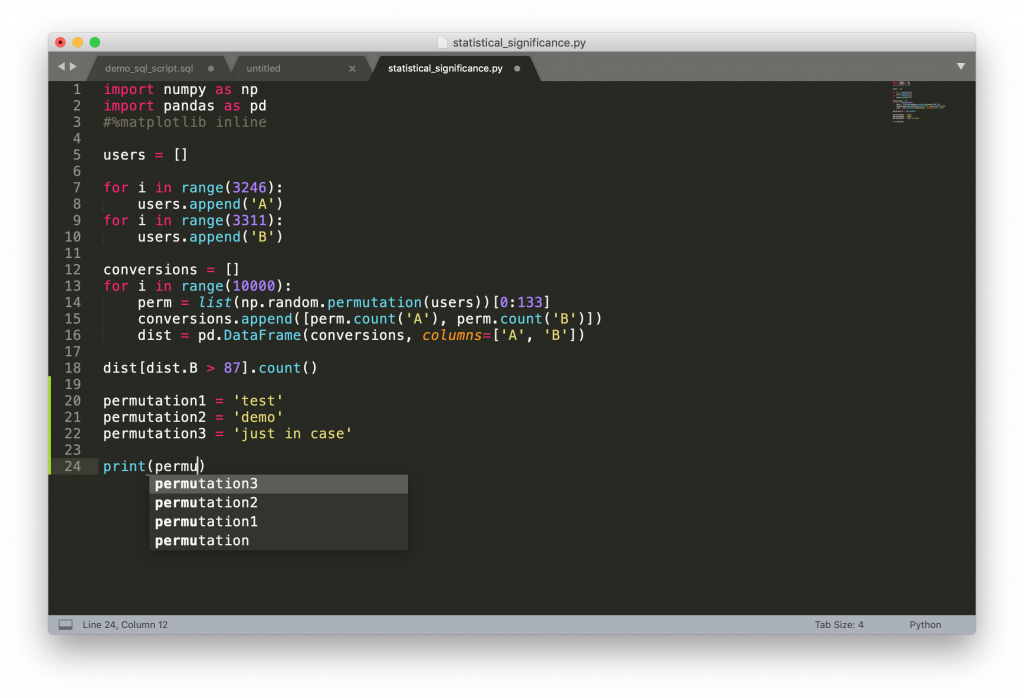
Приложение 8



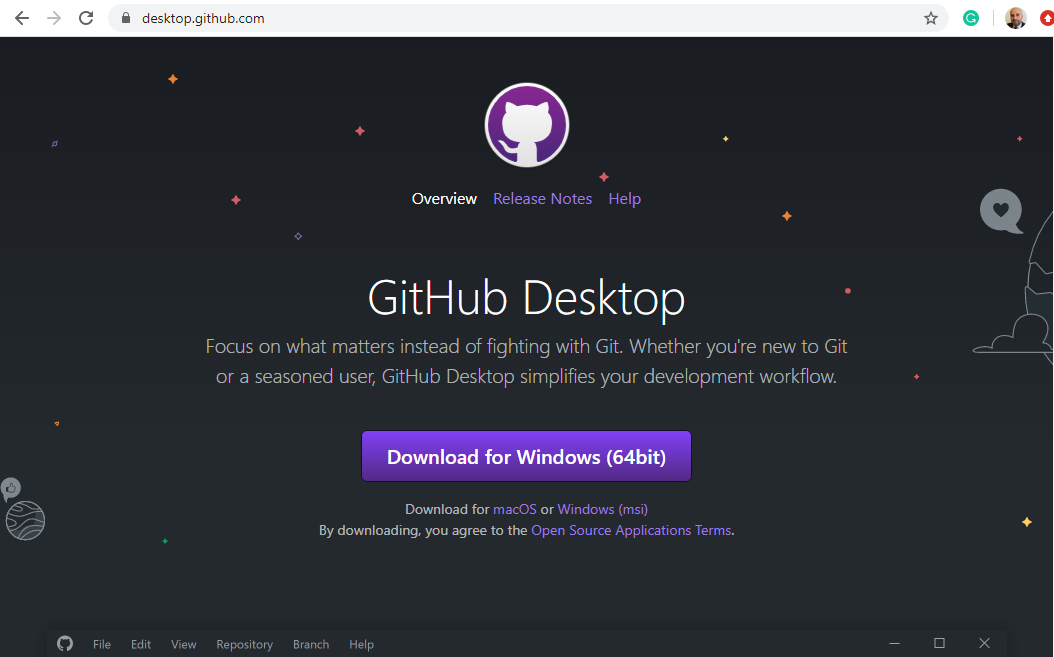
Приложение 9



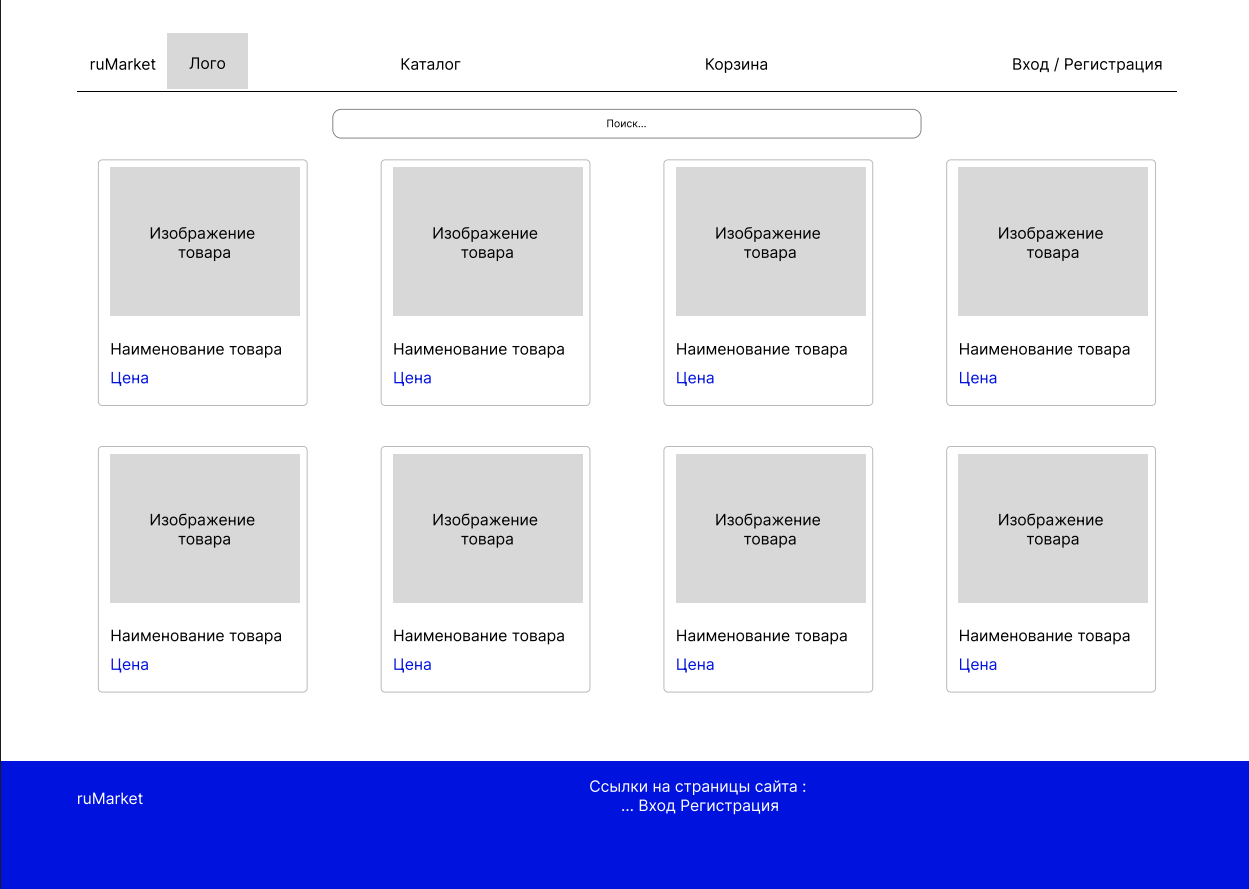
Приложение 10



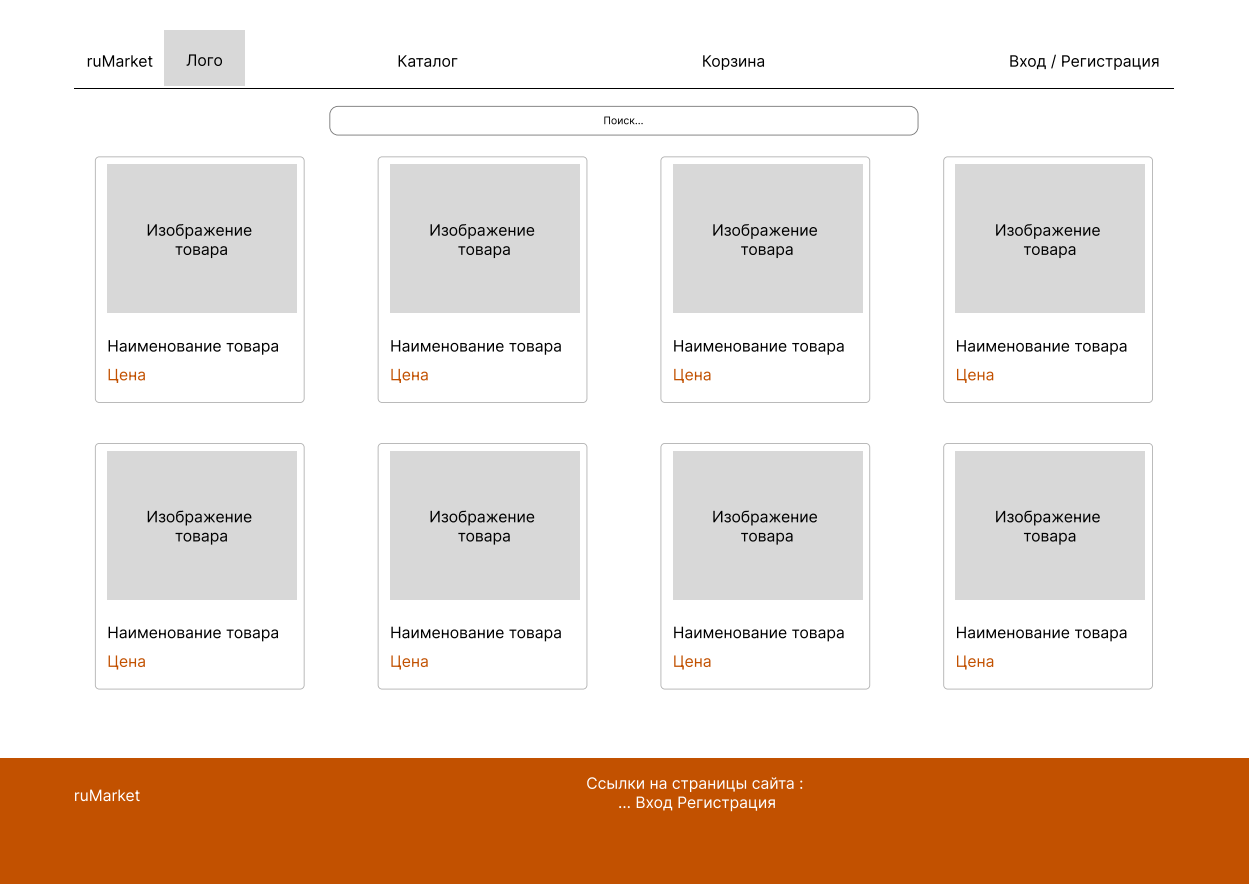
Приложение 11



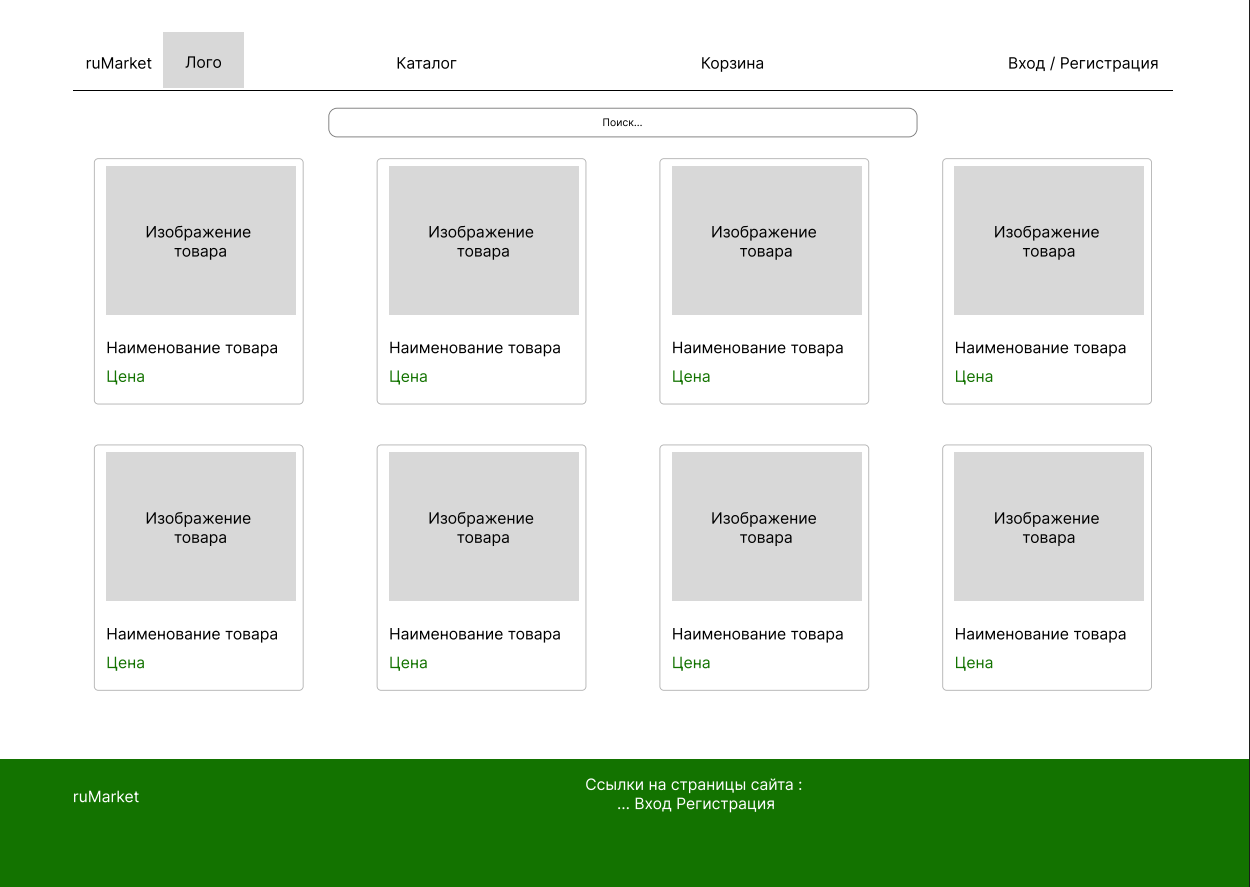
Приложение 12



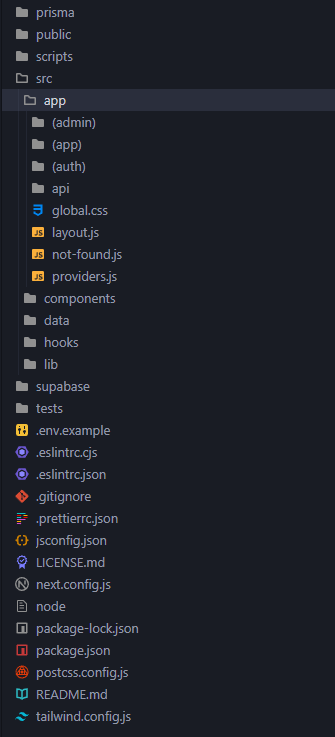
Приложение 13



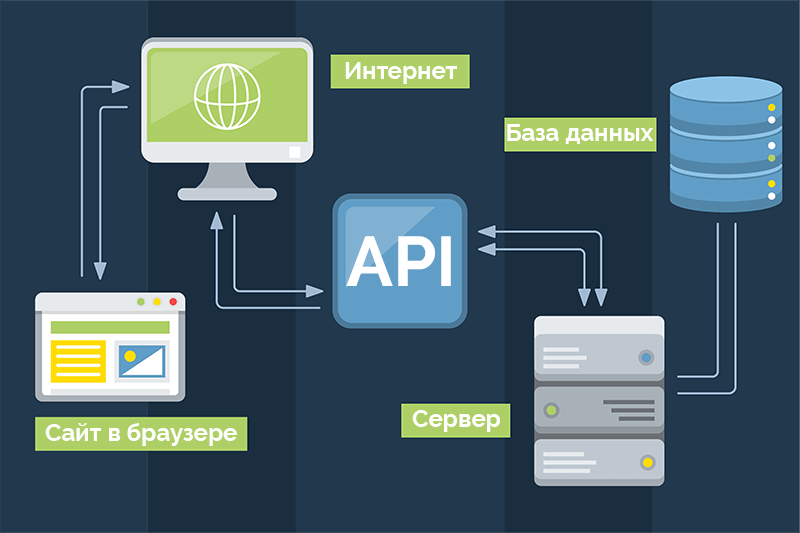
Приложение 14



Приложение 15



Приложение 16



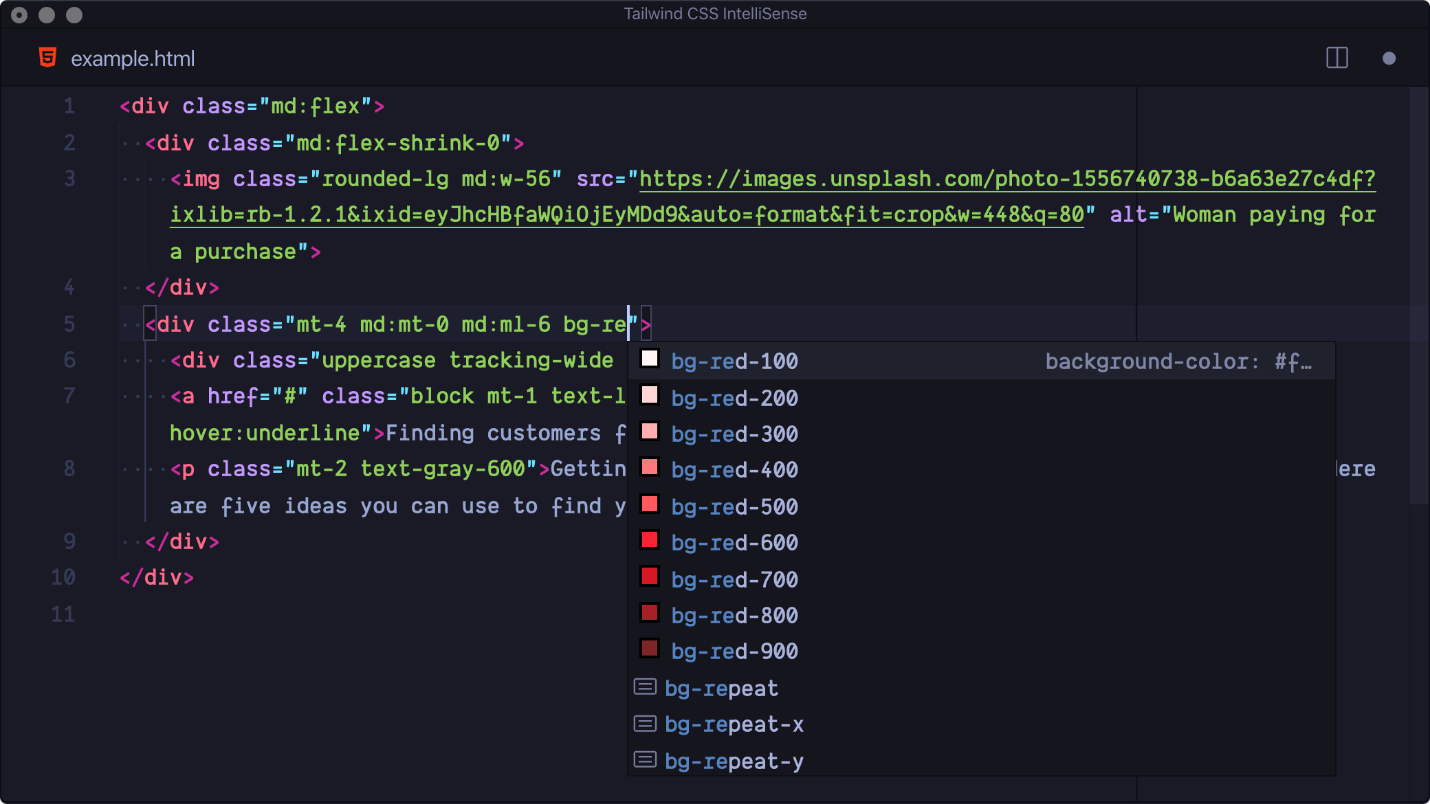
Приложение 17



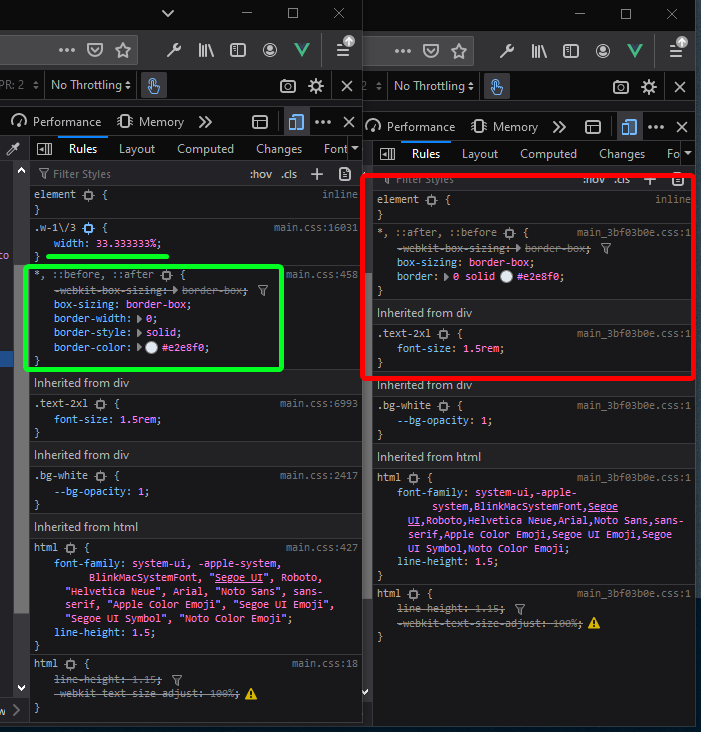
Приложение 18



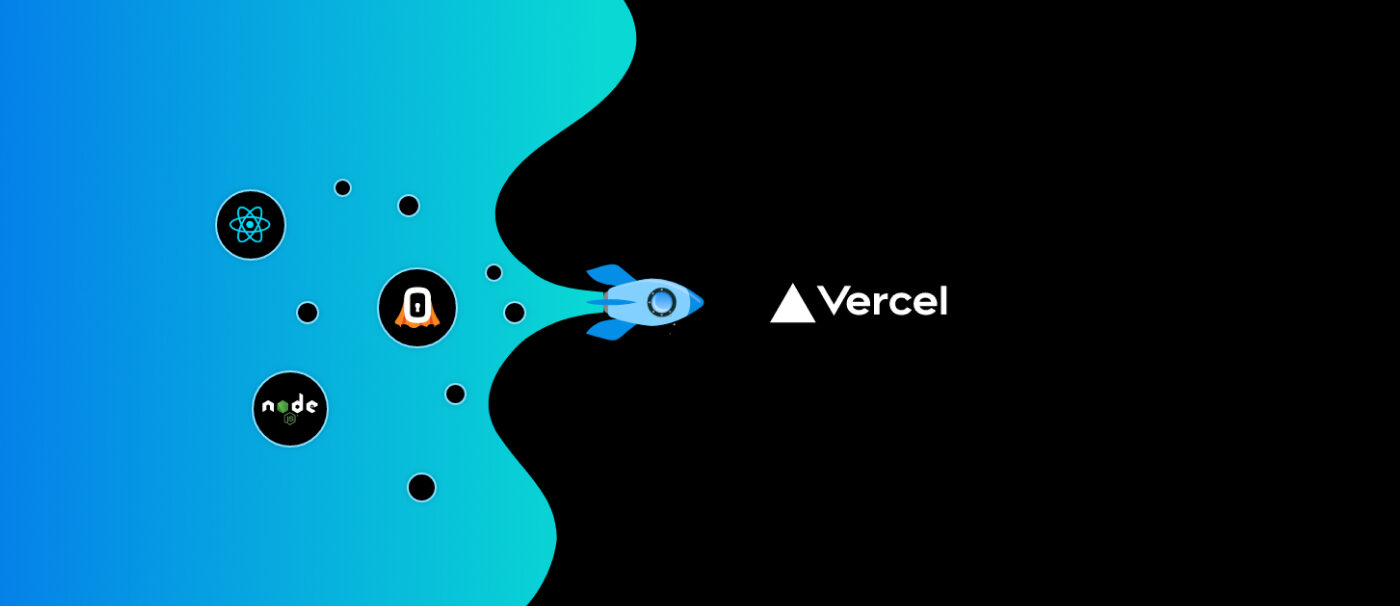
Приложение 19



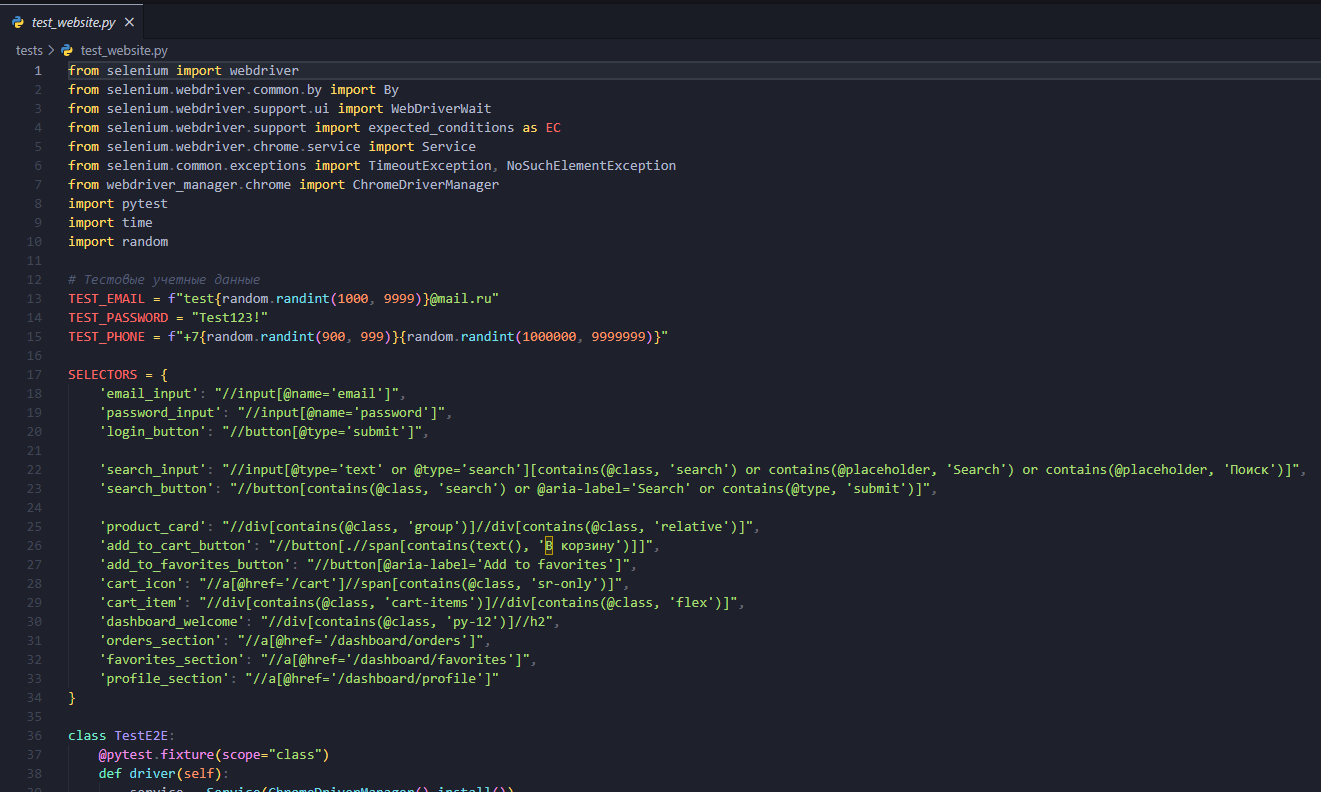
Приложение 20



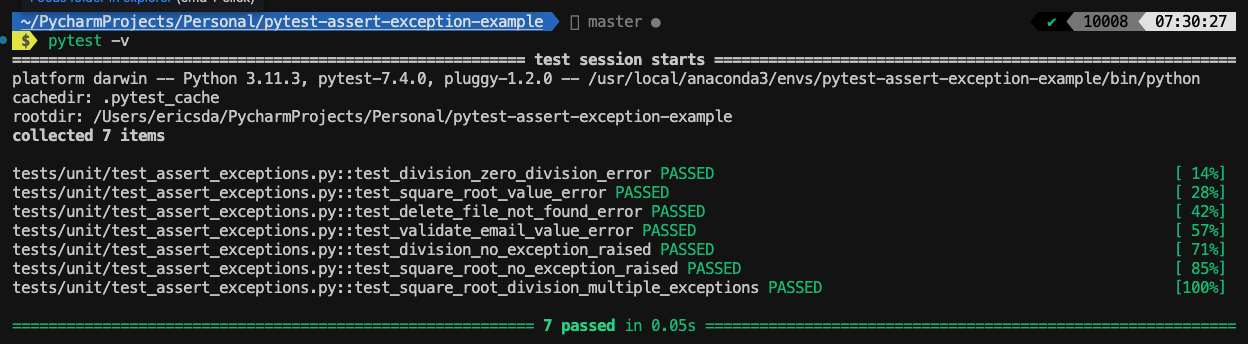
Приложение 21



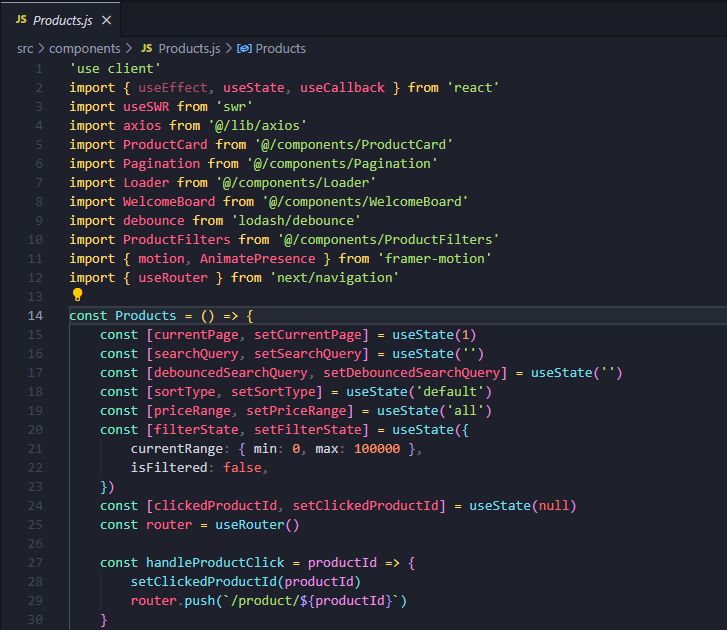
Приложение 22



Приложение 23



Приложение 24



Приложение 25

