Geekbrains

Разработка веб-сайта с использованием Vue.js и его компонентов.

Программа:

Специализация: Разработчик — Frontend разработка (Vue)

ФИО: Новиков Владимир Витальевич

Город Москва

Год 2024

**Содержание**

Введение

Глава 1. Основы разработки веб-сайта

1.1 Что такое веб-сайт, его особенности

1.2 Что такое разработка, кто такой веб-разработчик. Зачем нужна разработка и когда оно начинается

1.3 Этапы разработки веб-сайтов

1.4 Источники требований: что такое техническое задание и как проводить разработку веб-сайтов, если техническое задание отсутствует

Глава 2. Теоретические основы и преимущества использования Vue.js для разработки веб-приложений"

2.1 Обзор литературы по теме исследования

2.2 Анализ существующих решений для разработки веб-сайтов

2.3 Описание фреймворка Vue.js и его компонентов.

2.4 Преимущества использования Vue.js для разработки веб-приложений.

Глава 3. “Разработка веб-сайта с использованием Vue.js: от проектирования до реализации, на примере веб-сайта компании дизайнерских решений “Interno”

3.1 Постановка задачи для разработки веб-сайта

3.2 Выбор инструментов и технологий для реализации проекта

3.3 Проектирование архитектуры веб-сайта

3.4 Разработка клиентской части веб-сайта с использованием Vue.js

Заключение

Список используемой литературы

Приложения

**Введение**

Разработка веб-сайта с использованием Vue.js и его компонентов является актуальной темой в современном мире веб-разработки. Vue.js - это мощный JavaScript фреймворк, который позволяет создавать интерактивные пользовательские интерфейсы. Он предлагает множество преимуществ, таких как высокая производительность, легкость в использовании и поддержка современных веб-стандартов.

Структура работы включает в себя введение, основную часть, заключение и список использованной литературы, приложения. Основная часть состоит из нескольких глав, каждая из которых посвящена отдельному аспекту разработки веб-сайта с использованием Vue.js.

**Тема:** Разработка веб-сайта с использованием Vue.js и его компонентов.

**Цель:** данного дипломного проекта является создание веб-сайта с использованием Vue.js, который будет предоставлять пользователям удобный и интуитивно понятный интерфейс для взаимодействия с контентом. Задачи проекта включают в себя изучение основных концепций Vue.js, разработку архитектуры приложения, создание компонентов и их интеграцию в общую структуру сайта.

**Какую проблему решает**: Данный проект решает проблему удобства, скорости создания современного веб-сайта, построение удобной и понятной архитектур веб-сайта, для веб-разработчика с использованием Vue.js. Проект предоставляет пользователям удобный и интуитивно понятный интерфейс для взаимодействия с контентом, а также обеспечивает высокую производительность и поддержку современных веб-стандартов.

**Объект:** Объектом исследования является процесс разработки веб-сайта с использованием Vue.js. Предметом исследования являются компоненты Vue.js и их применение при создании веб-приложений.

**Методы:** методы исследования включают анализ литературы по теме, практическое применение изученных методов и инструментов, а также тестирование и оптимизацию полученного результата.

**Задачи:**

В данной работе будут рассмотрены следующие задачи:

1.Изучить литературу, касающуюся темы исследования

2. Обзор Vue.js и его возможностей.

3. Создание архитектуры приложения с использованием Vue.js.

4. Разработка компонентов Vue.js для различных функциональных блоков сайта.

5. Интеграция компонентов в общую структуру сайта.

6. Тестирование и оптимизация полученного результата.

**Инструменты:** Vue.js**,** HTML и CSS, JavaScript, Git, NPM, VS Code, Figma

Эти инструменты позволили создать эффективный и масштабируемый веб-сайт, который соответствует современным стандартам веб-разработки.

**Состав команды:** Новиков Владимир Витальевич (Веб-Разработчик)

**Глава 1. Основы разработки веб-сайта**

**1.1 Что такое веб-сайт, его особенности**

Веб-сайт – это набор связанных между собой веб-страниц, размещенных на одном сервере и имеющих общий домен. Веб-сайт может содержать текстовую информацию, графические элементы, видео и аудио контент, а также различные интерактивные элементы, такие как формы обратной связи, опросы и т.д.



Особенностью веб-сайта является его доступность для пользователей через интернет. Веб-сайт может быть доступен круглосуточно и из любой точки мира, где есть подключение к интернету.

Веб-сайт может служить различным целям: продвижение товаров или услуг, привлечение новых клиентов, повышение узнаваемости бренда и т.д. Разработка начинается после того, как определены цели и задачи будущего веб-сайта.  
  
**Основные понятия и термины**

Прежде чем мы погрузимся в мир веб-сайтов, важно понять ключевые термины:

Домен – это уникальное имя, которое используется для идентификации сайта в интернете. Вы можете сравнить доменное имя с адресом вашего дома: это уникальный идентификатор, который помогает другим найти вас в мире интернета. Например, в URL-адресе “[www.example.com](http://www.example.com/)“, “example.com” является доменным именем. Доменные имена регистрируются, и каждый домен уникален, что означает, что два веб-сайта не могут иметь одинаковые доменные имена.

Хостинг – это услуга предоставления ресурсов сервера для хранения файлов и обеспечения их доступности через интернет. Если сравнивать домен с адресом дома, то хостинг можно сравнить с самим домом, где хранятся все ваши вещи. В контексте веб-сайтов “вещами” являются файлы: код, текст, изображения и другой контент, доступ к которому получают пользователи при посещении вашего сайта. Существуют различные типы хостинга, включая общий хостинг, VPS-хостинг, облачный хостинг и выделенный хостинг, и выбор типа зависит от нужд и бюджета владельца.

Осознавая, что такое домен и хостинг, мы можем лучше понять, как работают веб-сайты, и какие шаги необходимы для создания собственного веб-пространства.

HTTPS (что расшифровывается как Hyper Text Transfer Protocol Secure) – это протокол безопасной передачи данных через интернет. HTTPS используется для защиты информации, которая передается между браузером пользователя и веб-сайтом, чтобы предотвратить перехват, изменение или уничтожение данных третьими лицами.

HTTPS основан на протоколе HTTP (Hyper Text Transfer Protocol), который используется для передачи данных в интернете, но добавляет слой шифрования с помощью протокола SSL (Secure Sockets Layer) или TLS (Transport Layer Security).

В адресной строке браузера сайты, использующие HTTPS, обычно отображаются с замком и префиксом “https://”, что говорит пользователям о том, что соединение защищено. Это особенно важно для сайтов, на которых происходит обмен конфиденциальной информацией, такой как данные кредитных карт или личные данные пользователей.

Система управления веб-сайтом (CMS) — это совокупность программ, на основе которых создается сайт. Пользователи могут взаимодействовать с ним именно благодаря движку. С его помощью программный код выводится в виде привычных пользователю страниц.

Система управления даёт владельцу возможность работать с контентом сайт: публиковать статьи, менять номера телефонов и адреса, добавлять товары в каталог.

Контент – это информация, которую содержит веб-сайт, включая тексты, изображения, видео и другие материалы.

Верстка – это процесс создания веб-страницы из HTML-кода и CSS-стилей.

Тестирование – это проверка веб-сайта на наличие ошибок и соответствие требованиям.

Отладка – это процесс исправления ошибок в коде веб-сайта.

Публикация – это процесс размещения веб-сайта в интернете для общего доступа.

Запуск – это момент, когда веб-сайт становится доступным для посетителей.

Поддержка – это процесс обслуживания и обновления веб-сайта после его запуска.

Развитие – это процесс улучшения и расширения функциональности веб-сайта.

Мониторинг – это наблюдение за работой веб-сайта и его показателями.

Аналитика – это анализ данных о посетителях веб-сайта для принятия решений по его улучшению.

Юзабилити – это удобство использования веб-сайта пользователями.

SEO-оптимизация – это процесс оптимизации веб-сайта для улучшения его видимости в поисковых системах.

**Виды веб-сайтов.**

Существует множество видов веб-сайтов, каждый из которых имеет свои особенности и предназначен для определенных целей. Вот некоторые из них:

1. Корпоративные сайты: это веб-сайты компаний, которые используются для представления информации о компании, ее продуктах и услугах, контактных данных и т.д. Они обычно имеют профессиональный дизайн и содержат много информации о компании.

2. Интернет-магазины: это веб-сайты, которые позволяют пользователям покупать товары или услуги онлайн. Они обычно имеют каталог товаров, корзину покупок и систему оплаты.

3. Блоги: это веб-сайты, на которых пользователи могут публиковать свои мысли, идеи, фотографии и видео. Они обычно имеют RSS-ленту, чтобы пользователи могли следить за обновлениями.

4. Социальные сети: это веб-сайты, которые позволяют пользователям общаться друг с другом, делиться информацией и фотографиями. Они обычно имеют профили пользователей, возможность добавлять друзей и оставлять комментарии.

5. Информационные порталы: это веб-сайты, которые предоставляют пользователям информацию по определенной тематике. Они обычно имеют большое количество статей, новостей и других материалов.

6. Сайты-визитки: это веб-сайты, которые представляют одного человека или небольшую компанию. Они обычно содержат краткую информацию о человеке или компании и контактные данные.

7. Форумы: это веб-сайты, на которых пользователи могут обсуждать различные темы. Они обычно имеют систему регистрации пользователей и возможность создавать новые темы для обсуждения.

8. Портфолио: это веб-сайты, на которых художники, дизайнеры и другие творческие люди могут показать свои работы. Они обычно содержат галерею работ и контактные данные.

Каждый вид веб-сайта имеет свои особенности и предназначен для определенных целей. При выборе типа веб-сайта необходимо учитывать цели и потребности вашей компании или проекта.

Также можно разделить веб-сайты по технологии их реалезации:

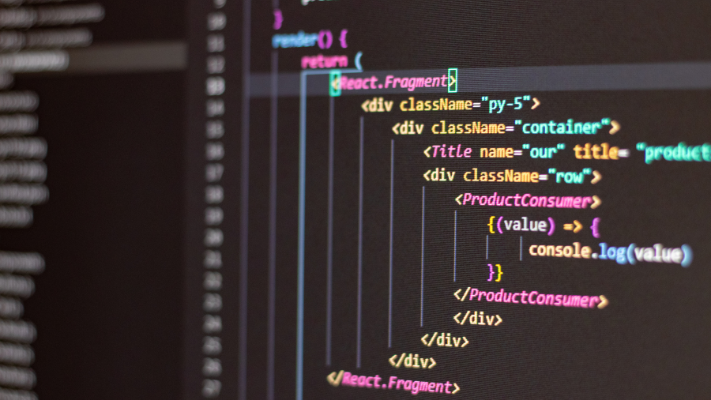
HTML сайты – это ресурсы, написанные на языке HTML Этот сайт легко написать даже в блокноте и вручную сверстать его.HTML-сайты удобны для создания сайта-визитки или лендингов.

CMS сайт – это веб-сайт, созданный с использованием системы управления контентом (CMS). C

Flash сайт – это веб-сайт, созданный с использованием технологии Flash. Это может быть анимированный или интерактивный сайт, который требует специального плагина для просмотра.

Конструктор – это программа или сервис, который помогает создавать веб-сайты без знания программирования. Конструкторы обычно предлагают готовые шаблоны и инструменты для быстрого создания сайтов. MS позволяет легко управлять содержимым сайта, добавлять, редактировать и удалять страницы, изображения, видео и другие элементы.

**1.2 Что такое разработка, кто такой веб-разработчик. Зачем нужна разработка и когда оно начинается**

**Разработка** — это процесс создания нового продукта или услуги. Она включает в себя множество этапов, начиная от идеи и заканчивая запуском готового продукта.  Она включает в себя множество этапов, начиная от идеи и заканчивая запуском готового продукта.

**Разработка веб-сайта** – это процесс создания и настройки веб-сайта, начиная от проектирования дизайна и заканчивая запуском готового продукта.

Первым этапом является анализ требований заказчика и определение целей проекта. Затем следует проектирование структуры и дизайна продукта. После этого происходит верстка страниц и программирование функционала. Далее следует тестирование и отладка продукта, а затем публикация и запуск готового продукта.

Важным аспектом разработки является выбор платформы для создания продукта. Существуют различные платформы, такие как WordPress, Joomla, Drupal и многие другие. Каждая платформа имеет свои преимущества и недостатки, поэтому выбор зависит от конкретных требований проекта.

Также важно учитывать юридические аспекты разработки. Необходимо соблюдать законодательство о защите персональных данных и авторских правах. Кроме того, необходимо обеспечить безопасность продукта от взломов и вирусов.

В целом разработка — это сложный и многогранный процесс, требующий знаний в области программирования, дизайна и маркетинга. Однако правильный подход к этому процессу поможет создать качественный продукт, который будет удовлетворять потребности заказчика и привлекать новых клиентов.

**Веб-разработчик** — это специалист, который занимается созданием и поддержкой веб-сайтов и веб-приложений. Он отвечает за весь процесс разработки, начиная от проектирования и заканчивая запуском готового продукта.

Веб-разработчики могут быть разделены на несколько категорий в зависимости от их специализации и выполняемых задач. Вот некоторые из основных видов веб-разработчиков:

**1. Frontend** –отвечает за верхушку айсберга, то есть за ту его часть, которую мы видим. Пользователь заходит на определенный сайт и видит некую картинку. Можно потыкать на разные кнопочки, лайкнуть фото или положить товар в корзину. Это все – поле ответственности frontend-разработчика. Мы не вникаем в то, как сайт работает изнутри, мы смотрим на красивую и понятную картинку и пользуемся сайтом по назначению. Если интерфейс скучноват или вообще выходит за экран мобильника – кидайте камни во фронтенда.

**2. Backend** – это главный по внутренностям сайта, ответственный за его серверный отдел. Он принимает задачи с «витрины» сайта, выполняет их и отправляет обратно на «витрину». Если пользователь нажал на картинку с дамской сумочкой, а вместо нее увидел туристический рюкзак – во всем виноват backend-разработчик.

**3. Fullstack** это универсальный работник. Он может делать работу как фронтенда, так и бэкенда, разбирается во всем понемногу, но детально не углубляется в специфику. Работать может за всех, но не слишком быстро. Зато он в курсе всех дел от и до, видит картину в целом и держит руку на пульсе. Как правило, в fullstack-разработчики чаще приходят из backend

Зачем нужна разработка веб-сайта? Разработка веб-сайта необходима для создания функционального и привлекательного онлайн-представительства компании или проекта. Она начинается после того, как определены цели и задачи будущего веб-сайта.

Некоторые из причин, почему разработка веб-сайта может быть необходима:

1. Продвижение товаров или услуг: Веб-сайт может использоваться для привлечения новых клиентов и увеличения продаж.

2. Привлечение новых клиентов: Веб-сайт может помочь в привлечении новых клиентов, предоставляя информацию о компании и ее продуктах или услугах.

3. Повышение узнаваемости бренда: Веб-сайт может помочь повысить узнаваемость бренда, особенно если он хорошо спроектирован и оптимизирован для поисковых систем.

4. Улучшение взаимодействия с клиентами: Веб-сайт может предоставить клиентам удобный способ связаться с компанией, задать вопросы или оставить отзывы.

5. Предоставление информации: Веб-сайт может быть использован для предоставления важной информации о компании, ее истории, миссии, ценностях и многом другом.

6. Увеличение прибыли: Веб-сайт может помочь увеличить прибыль компании, предлагая онлайн-продажи или другие способы монетизации.

7. Расширение географии бизнеса: Веб-сайт позволяет компании расширить свою географическую область, привлекая клиентов из разных стран и регионов.

8. Улучшение имиджа компании: Хорошо спроектированный и информативный веб-сайт может улучшить имидж компании, делая ее более профессиональной и надежной в глазах клиентов.

9. Удобство для клиентов: Веб-сайт может предложить клиентам удобный способ получения информации, заказа товаров или услуг, а также общения с представителями компании.

10. Улучшение коммуникации внутри компании: Веб-сайт может быть использован для улучшения внутренней коммуникации, предоставляя сотрудникам доступ к актуальной информации и ресурсам.

**1.3 Этапы разработки веб-сайтов**

В современном цифровом мире веб—сайт является неотъемлемой частью успешного онлайн—присутствия. Создание сайта с нуля требует тщательного планирования и последовательного выполнения этапов, чтобы создать функциональный и привлекательный ресурс.

Разработка веб-сайта проходит несколько этапов:



Этапы разработки веб-сайтов включают в себя следующие шаги:

1. Анализ требований заказчика и определение целей проекта: На этом этапе команда разработчиков обсуждает с заказчиком его потребности и ожидания от веб-сайта,проводит Анализ целевой аудитории, определяют, какой функционал должен быть реализован, какие страницы должны быть включены, какие элементы дизайна должны присутствовать и т.д.

2. Проектирование структуры и дизайна веб-сайта: На основе собранной информации команда разрабатывает структуру веб-сайта, определяет основные разделы и страницы, а также создает концепцию дизайна. Дизайнеры разрабатывают графические элементы, логотипы, иконки, выбирают цветовую палитру и шрифты. Важным аспектом этого этапа является создание макетов страниц и интерфейса пользователя, чтобы визуализировать и оценить внешний вид сайта.

3. Верстка страниц и программирование функционала: После утверждения дизайна начинается процесс верстки и программирования. Frontend-разработчики создают HTML-страницы, добавляют стили с помощью CSS и реализуют функциональность с помощью JavaScript или других языков программирования, Backend-разработчики реализуют функциональность: обработку форм, авторизацию пользователей, взаимодействие с базой данных.Бэкенд—разработка также включает тестирование и отладку функциональности, чтобы убедиться в ее правильной работе.

4. Тестирование и отладка веб-сайта: После завершения программирования разработчики проводят тестирование и отладку, чтобы проверить работоспособность и корректность функциональности нового ресурса. Они выполняют проверку на различных устройствах и браузерах, чтобы убедиться, что сайт выглядит и функционирует одинаково хорошо во всех условиях. Если возникают ошибки или неполадки, разработчики их исправляют и проводят дополнительные тесты. Также на этом этапе улучшают юзабилити сайта и исправляют выявленные проблемы.

5. Этап запуска и оптимизации является последним этапом в процессе созданиясайта. На этом этапе он готовится к публикации и настройке для обеспечения его эффективной работы и удовлетворения потребностей пользователей. Рассмотрим подробнее этот этап.

Размещение сайта на хостинге или сервере. После завершения работы с сайтом, необходимо разместить его на хостинге или сервере.

Настройка DNS и доменного имени: Для доступа к вашему сайту по доменному имени необходимо настроить DNS (Domain Name System).

Установка аналитических инструментов. Для отслеживания посещаемости и поведения пользователей на сайте

Оптимизация для скорости загрузки страниц. Оптимизация скорости загрузки страниц является важным аспектом успешного проекта. Программисты проводят оптимизацию кода, изображений, стилей и скриптов, чтобы ускорить загрузку страниц. Они минимизируют размер файлов, используют кэширование, сжимают изображения и применяют другие техники. Быстрая загрузка страниц улучшает пользовательский опыт и влияет на ранжирование сайта в поисковых системах.

Оптимизация SEO—параметров: Чтобы ваш ресурс ранжировался в поисковых системах и привлекал органический трафик, нужно провести оптимизацию SEO—параметров. Специалисты выполняют следующие действия:

Используют соответствующие ключевые слова в заголовках, мета—тегах, описаниях страниц и контенте. Это помогает поисковым системам понять, о чем ваш ресурс и что он предлагает.

Создают информативные и уникальные мета—теги для каждой страницы, включая заголовки и описания. Это помогает привлечь внимание пользователей в результатах поиска.

Создают читаемые URL—адреса для страниц, содержащие ключевые слова и отражающие их содержание.

Устанавливают правильные теги заголовков (H1, H2 и т.д.) для организации информации на страницах и выделения ключевых фраз.

Создают внутреннюю ссылочную структуру, чтобы обеспечить удобную навигацию и связь между страницами.

Разрабатывают контент высокого качества, соответствующий запросам пользователей и целям сайта.

Мониторинг и обслуживание: После запуска важно проводить мониторинг и обслуживание сайта. Разработчики следят за его работоспособностью, отслеживают метрики и аналитику, чтобы обнаружить любые проблемы

или потенциальные улучшения. Они также регулярно обновляют контент, добавляют новые функции и исправляют ошибки.

**1.4 Источники требований: что такое техническое задание и как проводить разработку веб-сайтов, если техническое задание отсутствует**

Техническое задание (ТЗ) – это документ, который содержит все требования к будущему проекту, в данном случае к веб-сайту. Оно является основой для разработки и реализации проекта, поскольку определяет цели, задачи, функциональность, структуру и дизайн веб-сайта.

Техническое задание должно быть составлено на начальном этапе разработки веб-сайта и утверждено заказчиком. Оно включает в себя следующие разделы:

1. Цели и задачи проекта: В этом разделе описываются цели и задачи, которые должен решать веб-сайт. Например, увеличение продаж, привлечение новых клиентов, повышение узнаваемости бренда и т.д.

2. Структура и навигация: Здесь описывается структура веб-сайта, его основные разделы и страницы, а также навигация между ними.

3. Дизайн: В этом разделе описывается внешний вид веб-сайта, включая цветовую гамму, шрифты, иконки и другие элементы дизайна.

4. Функциональность: В этом разделе перечисляются все функции, которые должен выполнять веб-сайт, например, форма обратной связи, корзина покупок, поиск по сайту и т.д.

5. Требования к контенту: Здесь описывается, какой контент должен быть размещен на веб-сайте, включая тексты, изображения, видео и другие медиафайлы.

6. Требования к безопасности: В этом разделе указываются требования к защите веб-сайта от взлома, вирусов и других угроз.

7. Требования к производительности: Здесь описываются требования к скорости загрузки страниц, устойчивости к нагрузкам и другим параметрам производительности.

8. Требования к поддержке и развитию: В этом разделе указываются требования к дальнейшей поддержке и развитию веб-сайта, включая обновление контента, добавление новых функций и т.д.

Если техническое задание отсутствует, то разработка веб-сайта может проводиться на основе устных договоренностей с заказчиком или на основе его пожеланий. Однако это может привести к недопониманию и конфликтам в процессе работы.

Несмотря на то, что разработка веб-сайта без технического задания может быть сложнее и требовать больше времени, при правильном подходе и коммуникации с заказчиком можно достичь хорошего результата.

**Глава 2. Теоретические основы и преимущества использования Vue.js для разработки веб-приложений"**

**2.1 Обзор литературы по теме исследования.**

При обучении по специальности Разработчик — Frontend разработка (Vue). Я пользовался не только открытыми источниками и официальным сайтом фреймворка <https://ru.vuejs.org/>, но и специальной литературой такой как Книга «Vue.js. От новичка до профессионала» Дмитрия Горшкова  
 От новичка до профессионала» Дмитрия Горшкова является отличным выбором для изучения разработки веб-сайтов с использованием фреймворка Vue.js. Она охватывает все аспекты разработки, начиная от основ и заканчивая сложными приложениями, что делает её идеальным ресурсом для студентов, работающих над дипломом в этой области.

В книге подробно рассматриваются следующие темы:

1. Основы Vue.js: Введение в фреймворк, его основные концепции и принципы работы.

2. Компоненты: Создание и настройка компонентов, их взаимодействие друг с другом, использование жизненного цикла компонентов.

3. Шаблоны: Работа с шаблонами, синтаксис и возможности шаблонизации в Vue.js.

4. Данные: Обработка данных в компонентах, двухстороннее связывание данных, реактивность.

5. Маршрутизация: Использование Vue Router для создания динамических маршрутов и навигации по приложению.

6. Тестирование: Различные подходы к тестированию Vue.js-приложений, использование таких инструментов, как Jest и Vue Test Utils.

7. Инструменты и библиотеки: Использование Vue CLI для управления проектом, интеграция с другими инструментами и библиотеками, такими как Vuex и Vuex Persisted State.

В целом, книга «Vue.js. От новичка до профессионала» Дмитрия Горшкова является ценным ресурсом для студентов, работающих над дипломом в области веб-разработки с использованием Vue.js. Она предоставляет глубокое понимание фреймворка, его компонентов, а также практические примеры, которые помогут студентам развить свои навыки и создать качественные веб-приложения.

**2.2 Анализ существующих решений для разработки веб-сайтов**  
**Анализ существующих решений для разработки веб-сайтов**

Сегодня веб-разработка является одной из самых динамично развивающихся областей IT-индустрии. Веб-сайты стали неотъемлемой частью нашей жизни, и их качество и функциональность играют важную роль в успехе бизнеса. В связи с этим, выбор правильного инструмента для разработки веб-сайта становится критически важным.

При выборе инструмента для разработки веб-сайтов необходимо учитывать ряд ключевых факторов:

- Простота использования: инструмент должен быть понятным и удобным в использовании, чтобы разработчики могли быстро приступить к работе.

- Гибкость: инструмент должен предоставлять возможность создания различных типов веб-сайтов, от простых одностраничных до сложных многофункциональных порталов.

- Поддержка стандартов: инструмент должен соответствовать современным стандартам веб-разработки, таким как HTML5, CSS3 и JavaScript.

- Производительность: инструмент должен обеспечивать высокую производительность и скорость загрузки веб-сайтов.

- Поддержка сообщества: наличие активного сообщества разработчиков, которое может помочь в решении возникающих проблем.

В настоящее время существует множество инструментов для разработки веб-сайтов. Рассмотрим некоторые из них:

- HTML/CSS: классический подход к веб-разработке, где каждый элемент сайта создается вручную с помощью HTML и стилизуется с помощью CSS. Этот метод требует хороших знаний в области веб-разработки и может быть трудоемким.

- Bootstrap: популярная библиотека для создания адаптивных веб-сайтов с использованием HTML, CSS и JavaScript. Она предоставляет готовые шаблоны и компоненты, что упрощает процесс разработки.

- React: JavaScript-библиотека для разработки пользовательских интерфейсов. Она позволяет создавать интерактивные веб-приложения с высокой производительностью.

- Vue.js: JavaScript-фреймворк, который фокусируется на создании одностраничных приложений (SPA). Он предлагает простой синтаксис и мощные инструменты для разработки веб-приложений.

Проведем сравнительный анализ инструментов для разработки веб-сайтов:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Инструм-ент | Простота исполь-зования | Гибкость | Под-держка стандар-тов | Произ-водитель-ность | Под-держка сооб-щества |
| HTML/CSS | Средняя | Высокая | Высокая | Низкая | Высокая |
| Bootstrap | Высокая | Высокая | Средняя | Средняя | Высокая |
| React | Высокая | Средняя | Высокая | Высокая | Высокая |
| Vue.js | Высокая | Средняя | Средняя | Высокая | Средняя |

Таблица 1. Сравнительный анализ популярных инструментов для разработки веб-сайтов

Выбор инструмента для разработки веб-сайта зависит от конкретных потребностей проекта. Для простых сайтов, где не требуется высокая производительность и сложные функции, HTML/CSS может быть хорошим выбором. Для более сложных проектов, таких как одностраничные приложения, React или Vue.js могут быть более подходящими. Bootstrap может быть использован для создания адаптивных сайтов с минимальными усилиями.

В данной главе мы провели анализ существующих решений для разработки веб-сайтов. Мы рассмотрели основные требования к таким инструментам и сравнили несколько популярных вариантов. Выбор инструмента зависит от конкретных потребностей и целей проекта, но важно учитывать такие факторы, как простота использования, гибкость, поддержка стандартов и производительность.

**2.3 Описание фреймворка Vue.js и его компонентов.**  
В настоящее время веб-разработка является одной из наиболее динамично развивающихся отраслей IT-индустрии. Веб-сайты стали неотъемлемой частью нашей жизни, и их качество и функциональность играют важную роль в успехе бизнеса. В связи с этим, выбор правильного инструмента для разработки веб-сайта становится критически важным. Одним из таких инструментов является Vue.js – JavaScript-фреймворк, который фокусируется на создании одностраничных приложений (SPA).

 **Vue.js** – это прогрессивный JavaScript-фреймворк для создания пользовательских интерфейсов. Он был создан Эваном Ю в 2014 году и с тех пор стал одним из самых популярных инструментов для разработки веб-приложений. Vue.js предлагает простой синтаксис и мощные инструменты для разработки веб-приложений.

**Компоненты Vue.js**. Одной из ключевых особенностей Vue.js являются компоненты. Компоненты – это отдельные части приложения, которые могут быть повторно использованы и объединены для создания более сложных интерфейсов. Каждый компонент имеет свой собственный набор данных, логику и шаблон.

**Хуки жизненного цикла** Vue.js - это функции, которые вызываются в определенных точках жизненного цикла компонента Vue.js. Вот список основных хуков жизненного цикла:

1. beforeCreate : Этот хук вызывается перед созданием экземпляра компонента. Здесь можно выполнить предварительную инициализацию данных и логики.

2. created: Этот хук вызывается после создания экземпляра компонента, но до связывания данных. Здесь можно выполнить инициализацию, которая должна произойти только один раз.

3. beforeMount : Этот хук вызывается перед первым монтированием компонента в DOM. Здесь можно выполнить подготовительные операции, такие как загрузка данных или инициализация сторонних библиотек.

4. mounted : Этот хук вызывается сразу после первого монтирования компонента в DOM. Здесь можно выполнять операции, которые зависят от наличия DOM, такие как манипуляции с элементами или установка обработчиков событий.

5. beforeUpdate : Этот хук вызывается перед обновлением компонента. Здесь можно подготовиться к обновлению, например, обновить состояние, которое должно быть синхронизировано с DOM.

6. updated : Этот хук вызывается непосредственно после обновления компонента. Здесь можно выполнять любые операции, которые должны быть выполнены после обновления.

7. beforeDestroy : Этот хук вызывается перед уничтожением компонента. Здесь можно освободить ресурсы, которые были выделены в течение жизненного цикла компонента.

8. destroyed : Этот хук вызывается, когда экземпляр компонента полностью уничтожен. Здесь можно выполнить окончательную очистку и освобождение ресурсов.

Хуки жизненного цикла позволяют разработчикам управлять различными этапами жизненного цикла компонента и реагировать на изменения в состоянии компонента или его окружения.

**Шаблоны и директивы.** Шаблоны в Vue.js используются для определения внешнего вида компонента. Они могут содержать HTML, а также специальные директивы, которые позволяют управлять отображением и поведением элементов. Некоторые из наиболее распространенных директив включают v-if, v-for, v-model и v-bind.

**Двухстороннее связывание данных.** Vue.js поддерживает двухстороннее связывание данных, что означает, что изменения в модели данных автоматически отражаются в представлении, и наоборот. Это достигается с помощью директивы v-model.

**Маршрутизация в Vue.js** осуществляется с помощью библиотеки Vue Router. Она позволяет создавать динамические маршруты и навигацию по приложению.

**Тестирование в Vue.js** выполняется с помощью библиотеки Jest и Vue Test Utils. Эти инструменты позволяют писать тесты для проверки функциональности компонентов и других частей приложения.

**Инструменты и библиотеки.**Vue.js предоставляет ряд инструментов и библиотек, которые облегчают разработку веб-приложений. Например, Vue CLI используется для управления проектом, а Vuex – для управления состоянием приложения.

Vue.js – это мощный и гибкий инструмент для разработки веб-приложений. Он предлагает простой синтаксис, мощные инструменты и поддержку сообщества. Благодаря своим компонентам, шаблонам и директивам, Vue.js позволяет создавать сложные и интерактивные веб-приложения.

**2.4 Преимущества использования Vue.js для разработки веб-приложений.**

Vue.js является одним из самых популярных JavaScript-фреймворков для разработки веб-приложений. Он предлагает множество преимуществ, которые делают его привлекательным выбором для разработчиков. В этой главе мы рассмотрим основные преимущества использования Vue.js для разработки веб-приложений.

1. Простота использования.

Одно из главных преимуществ Vue.js - его простота использования. Фреймворк предлагает простой синтаксис, который легко понять и использовать даже начинающим разработчикам. Vue.js использует HTML-компоненты, которые позволяют разработчикам создавать сложные пользовательские интерфейсы без необходимости изучать новые языки или сложные API.

2. Гибкость.

Vue.js очень гибкий фреймворк, который позволяет разработчикам выбирать уровень абстракции, соответствующий их потребностям. Он может использоваться как для создания простых одностраничных приложений, так и для разработки сложных многофункциональных веб-приложений. Кроме того, Vue.js легко интегрируется с другими фреймворками и библиотеками, такими как React и Angular.

3. Двухстороннее связывание данных.

Vue.js поддерживает двухстороннее связывание данных, что означает, что изменения в модели данных автоматически отражаются в представлении, и наоборот. Это достигается с помощью директивы v-model. Двухстороннее связывание данных упрощает разработку и уменьшает количество кода, необходимого для поддержания синхронизации данных между моделью и представлением.

4. Производительность.

Vue.js известен своей высокой производительностью. Он оптимизирован для быстрой загрузки и рендеринга веб-страниц, что делает его идеальным выбором для создания высокопроизводительных веб-приложений. Vue.js также поддерживает виртуальный DOM, который позволяет значительно ускорить обновление DOM без потери производительности.

5. Активное сообщество.

Vue.js имеет активное и дружелюбное сообщество разработчиков, которое постоянно работает над улучшением фреймворка и созданием новых инструментов и библиотек. Сообщество Vue.js предоставляет множество ресурсов, таких как документация, учебники, форумы и конференции, которые помогают разработчикам учиться и развиваться.

Vue.js предлагает множество преимуществ для разработки веб-приложений. Он прост в использовании, гибок, поддерживает двухстороннее связывание данных и обладает высокой производительностью. Активное сообщество разработчиков Vue.js постоянно работает над улучшением фреймворка, что делает его привлекательным выбором для многих разработчиков.

**Глава 3. “Разработка веб-сайта с использованием Vue.js: от проектирования до реализации, на примере веб-сайта компании дизайнерских решений “Interno”**

**3.1 Постановка задачи для разработки веб-сайта на примере “Interno”**

Веб-сайт дизайна интерьеров и мебели является важным инструментом для привлечения клиентов и демонстрации портфолио дизайнера. Целью данного проекта является разработка веб-сайта, который будет представлять услуги дизайнера интерьеров и предлагать каталог мебели.

**1.Основными целями проекта являются**:

- Создание современного и привлекательного веб-сайта для представления услуг дизайнера интерьеров.

- Обеспечение удобной навигации по сайту для пользователей.

- Разработка каталога мебели с возможностью фильтрации по различным параметрам.  
 - Создание Раздела статей и новостей, для привлечения нового потока пользователей.

**2.Задачи проекта включаю**т:

- Разработка структуры сайта и макета страниц.

- Выбор и интеграция необходимых технологий и инструментов для реализации проекта.

- Создание контента для сайта (тексты, изображения).

- Тестирование и оптимизация сайта для улучшения пользовательского опыта.

**3. Описание целевой аудитории**

Целевой аудиторией веб-сайта являются люди, заинтересованные в услугах дизайнера интерьеров и покупке мебели. Это могут быть как частные лица, так и компании, которые ищут профессиональные услуги по дизайну интерьера и мебели.

**4. Анализ конкурентов**

Перед началом разработки веб-сайта необходимо провести анализ конкурентов. Это позволит определить сильные и слабые стороны конкурентов, а также выявить потребности целевой аудитории. Анализ конкурентов поможет определить уникальное предложение нашего веб-сайта и выделиться среди других игроков на рынке.  
Для анализа конкурентов были выбраны 3 топ позиции в поисковой системе google.com  
<https://geometrium.com/><https://balcon.studio/><https://salon.ru/interiors>.  
 Наш анализ выглядит следующим образом:

1. Сайт https://salon.ru/interiors:

- Предлагает широкий спектр услуг по дизайну интерьеров, включая проектирование, авторский надзор и комплектацию объектов.

- Имеет удобную навигацию по сайту, что позволяет быстро найти нужную информацию.

- Отсутствует возможность онлайн-заказа мебели.

- Недостаток: нет каталога мебели.

2. Сайт https://geometrium.com/:

- Специализируется на современном дизайне интерьеров и мебели.

- Имеет удобную навигацию по сайту, что позволяет быстро найти нужную информацию.

- Есть возможность онлайн-заказа мебели.

- Предоставляет каталог мебели с возможностью фильтрации по различным параметрам.

- Сильная сторона: современный дизайн и функциональность сайта.

3. Сайт https://balcon.studio/:

- Предлагает услуги по дизайну интерьеров и проектированию мебели.

- Удобная навигация по сайту, есть возможность онлайн-заказа.

- Нет каталога мебели.

- Слабая сторона: отсутствие каталога мебели.

По результатам анализа конкурентов можно сделать вывод, что сайт https://geometrium.com/ имеет наибольшее количество преимуществ и может служить примером для разработки нашего веб-сайта. На него и будем ориентироваться дальнейшей работе.

**5. Структура сайта**

Структура сайта должна быть понятной и удобной для пользователей. Основные разделы сайта могут включать:

- Главная страница с привлекательным баннером и демонстрацией проектов дизайнера интерьеров и каталогом мебели. Дополнительными статьями по теме дизайна.

- Страница проектов, где представлены различные виды готовых решений дизайна интерьера.

-Страница новостей, где будет предоставлена актуальная информация из мира дизайна.

- Каталог мебели с возможностью фильтрации по различным параметрам.

- Подвал сайта с информацией для связи.

**6. Технологии и инструменты**

Для реализации проекта будут использоваться следующие технологии и инструменты:

- Фреймворк Vue.js для создания интерактивных веб-приложений.

- Библиотека Bootstrap для создания адаптивного дизайна.

- HTML и CSS для создания структуры и внешнего вида веб- страниц,

- JavaScript для добавления интерактивности и логики в веб- страницы.

- Git для отслеживания изменений в коде проекта

- NPM для установки, управления и распространения пакетов JavaScript

- VS Code. редактор кода, который используется для написания и редактирования кода проекта.

-Figma для работы с дизайном проекта

**7 Создание контента**

Контент сайта должен быть информативным и привлекательным для пользователей. Он должен включать тексты, изображения , которые демонстрируют услуги дизайнера интерьеров и мебель. Важно убедиться, что контент соответствует потребностям целевой аудитории и помогает достичь поставленных целей проекта.

**8 Тестирование и оптимизация**

После завершения разработки веб-сайта необходимо провести тестирование и оптимизацию. Тестирование поможет выявить ошибки и недочеты в работе сайта, а оптимизация позволит улучшить пользовательский опыт и повысить эффективность работы сайта.

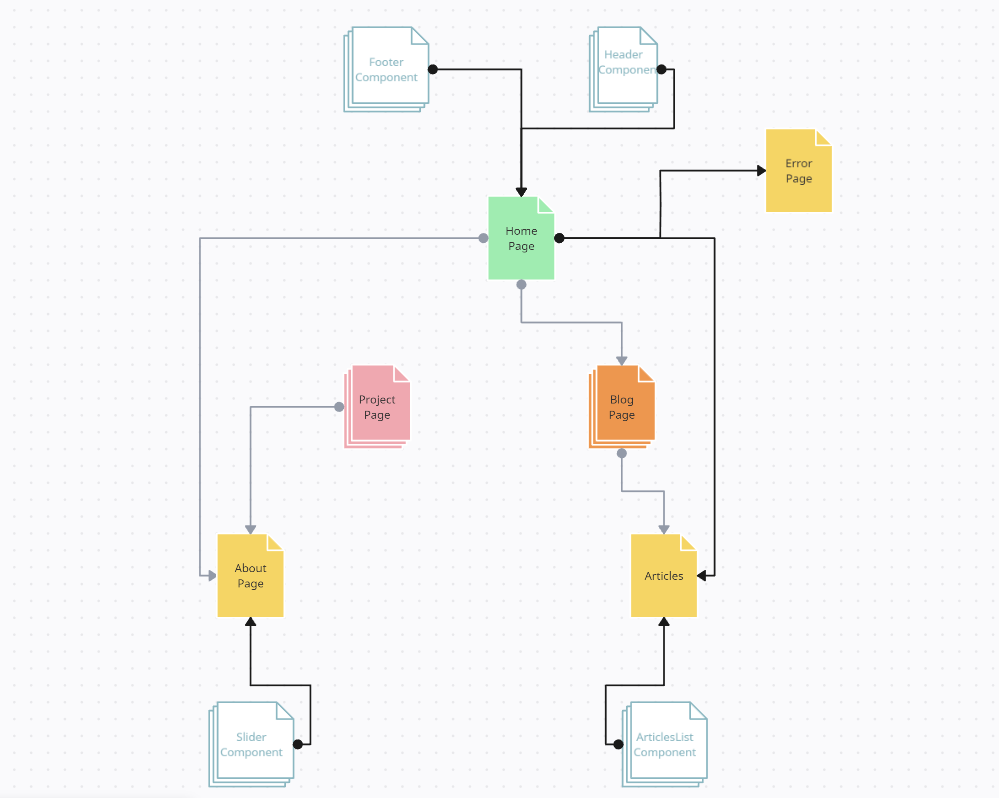
В данной главе мы определили цели и задачи проекта, описали целевую аудиторию и провели анализ конкурентов. Мы также определили структуру сайта и выбрали необходимые технологии и инструменты для реализации проекта. Следующим шагом будет разработка веб-сайта согласно поставленным задачам и требованиям.

**3.2 Проектирование архитектуры веб-сайта на примере “Interno”**

Архитектура клиентской части веб-сайта представляет собой набор компонентов, которые взаимодействуют друг с другом и с серверной частью для обеспечения функциональности веб-сайта. Компоненты могут быть как простыми (например, кнопки, поля ввода), так и сложными (например, формы, таблицы).

Проектируя архитектура веб-сайта “Interno” я ориентировался на удобство, доступность, скоростью освоения технологиями веб-разработки.

Наша архитектура выглядит следующим образом:

Приложение №2.

Как мы видим из архитектуры наш проект состоит из 6 страниц и 4 компонентов.  
 - Главная страница HomePage с привлекательным баннером и демонстрацией проектов дизайнера интерьеров и каталогом мебели. Дополнительными статьями по теме дизайна.

- Страница Project, где представлены различные виды готовых решений дизайна интерьера.

- Страница Blog, где пользователь может ознакомиться с нашими идеями

- Страница Articles, где будет предоставлена актуальная информация из мира дизайна.

- Страница About , где показываются конкретные проекты и более подробная инофрмация о них

- Страница Error 404, куда будет перекидывать пользователя в случаи написание неправильного URL адреса.

- Компонент Footer сайта с информацией для связи.

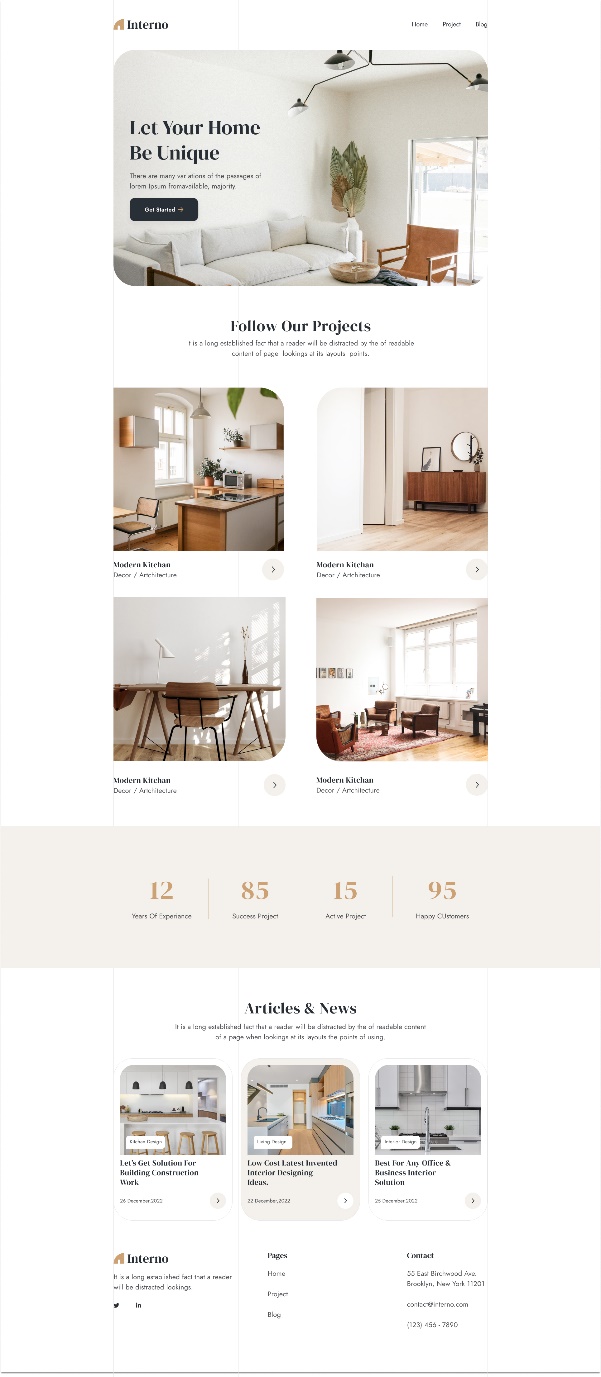
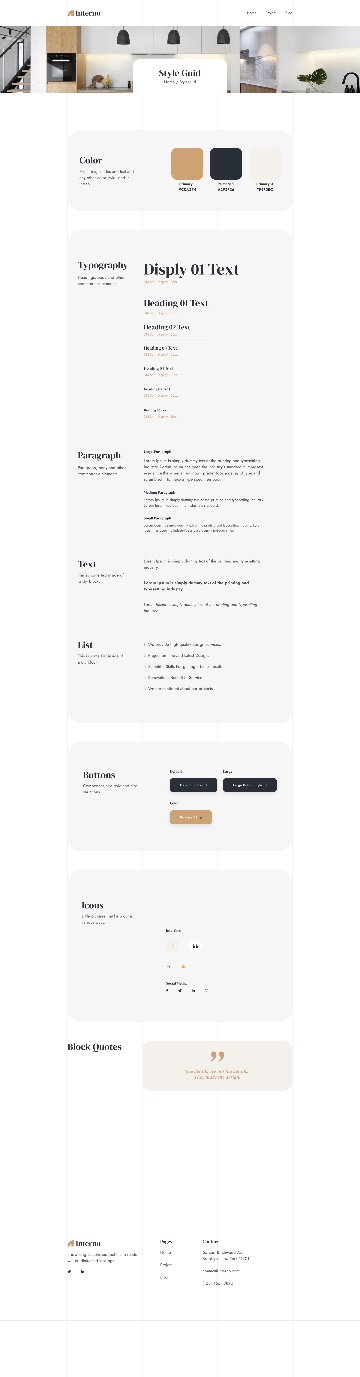
- Компонент Header сайта с удобной навигацией

- Компонент Slider сайта где находится вся логика отображения слайдера.

- Компонент ArticlesList где находятся актуальная статьи которые обновляются с сервера.

При создании архитектуры мы отталкивались от предоставленной информации по проекту и дизайна.

Дизайн-макеты был разработан в онлайн cервисе Figma.



Приложение №3

Также при составлении архитектуры сайта я отталкивался от современного подхода к реализации данной задачи, я подобрал:

1. Доменное имя : Interno-design.com.

2. Хостинг : Google Cloud Platform , который поддерживает Node.js и предоставляет SSH-доступ для установки и настройки сервера.

3. Фреймворк : Vue.js будет использоваться для создания клиентской части веб-сайта.

5. База данных : Я использовал массивы внутри компонентов.

6. Репозиторий : на GitHub для управления кодом проекта.

7. Дизайн : Figma для создания дизайна интерфейса, который затем экспортируется в виде компонентов для интеграции в проект Vue.js.

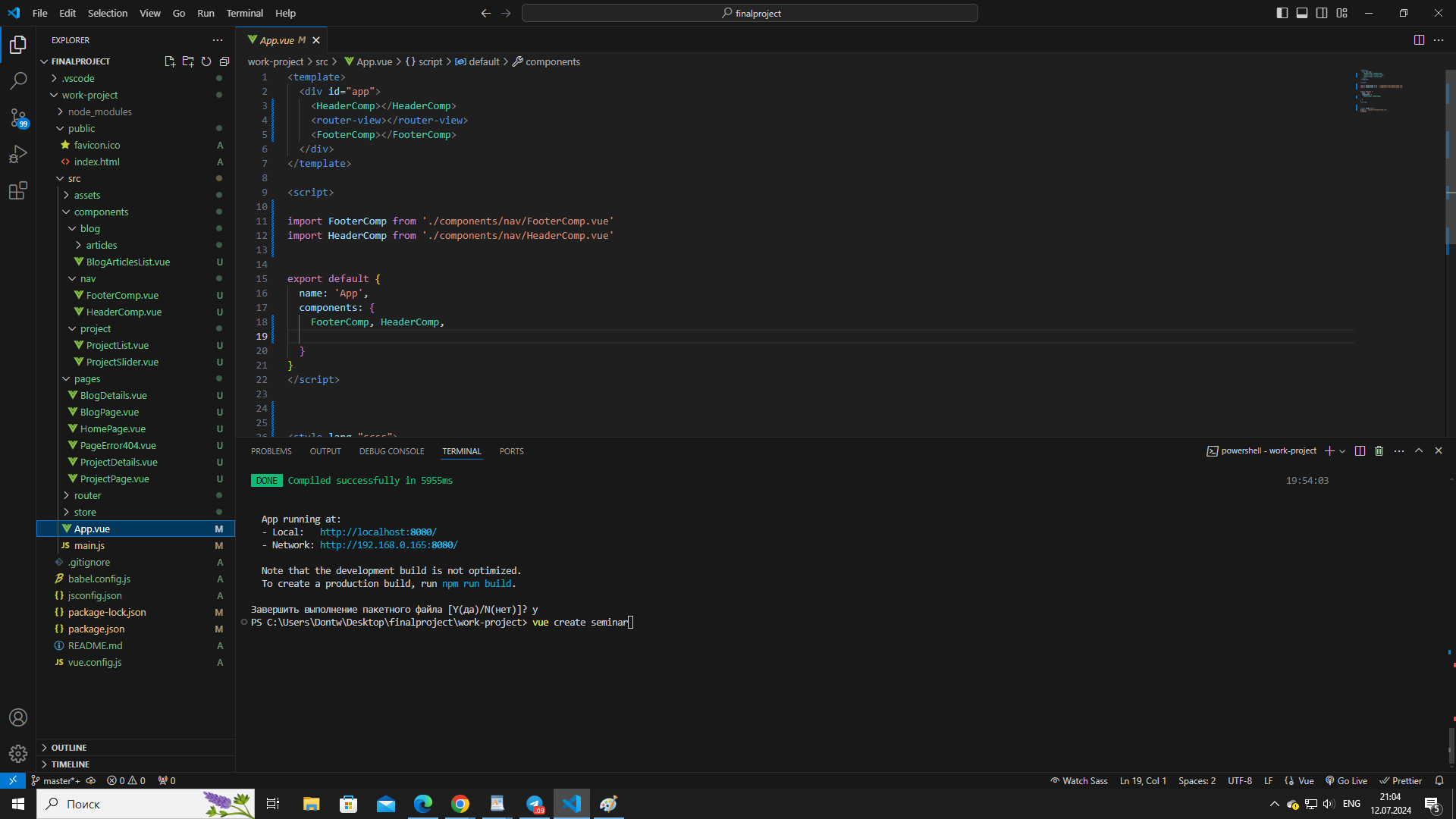
**3.3 Разработка клиентской части веб-сайта с использованием Vue.js**

В современном мире веб-разработка является одной из наиболее динамично развивающихся областей информационных технологий. С каждым годом появляются новые инструменты и технологии, которые позволяют создавать более эффективные и удобные веб-приложения. Одной из таких технологий является Vue.js - JavaScript-фреймворк, который позволяет разработчикам быстро и легко создавать интерактивные пользовательские интерфейсы.

Начнем с установки Vue CLI

npm install -g @vue/cli

Создаем проект  
 vue create work-project

После установки переходим в компонент App.vue где и будет производить основную работу.

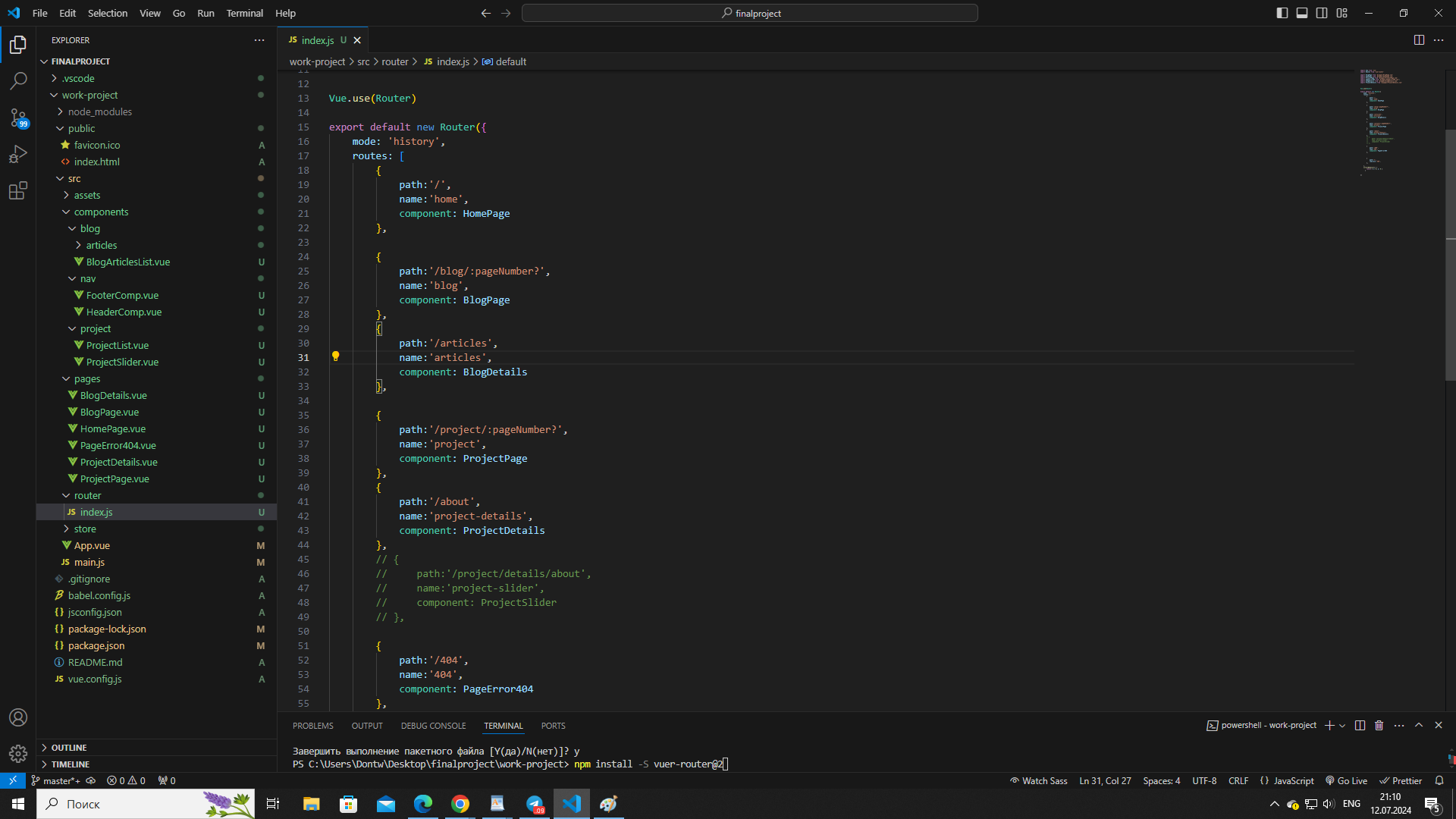
Приложение № 4

В компоненте App.vue импортируем необходимые зависимости:

import FooterComp from './components/nav/FooterComp.vue'

import HeaderComp from './components/nav/HeaderComp.vue'

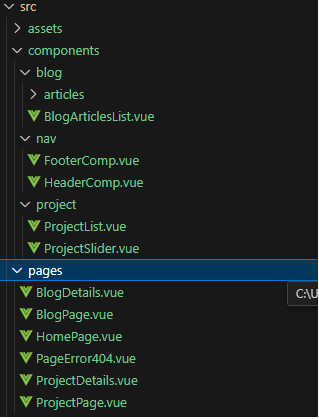
Далее устанавливаем vue router.  
npm install -S vuer-router@2  
Создаем папку router и в ней создаем файл index.js



Приложение №5

В файле index.js мы прописываем все наши маршруты - после какого URL запроса или клика на сайте, будет направляться пользователь

В папке SRC создаем дополнительную папку Pages, где будут храниться все наши страницы.



Для удобства работы со стилями устанавливаем препроцессор SASS.

npm install -g sass

И подключаем его к нашему главное файлу App.vue

<style lang="scss">

@import './assets/styles/style.css'

</style>

Переходим в файл main.js подключаем в нем router и импортируем bootstrap она нам пригодиться для создания слайдера.

import Vue from 'vue'

import App from './App.vue'

import store from './store'

import router from './router'

import 'bootstrap/dist/css/bootstrap.css'

import bootstrap from 'bootstrap/dist/js/bootstrap.bundle'

Vue.config.productionTip = false

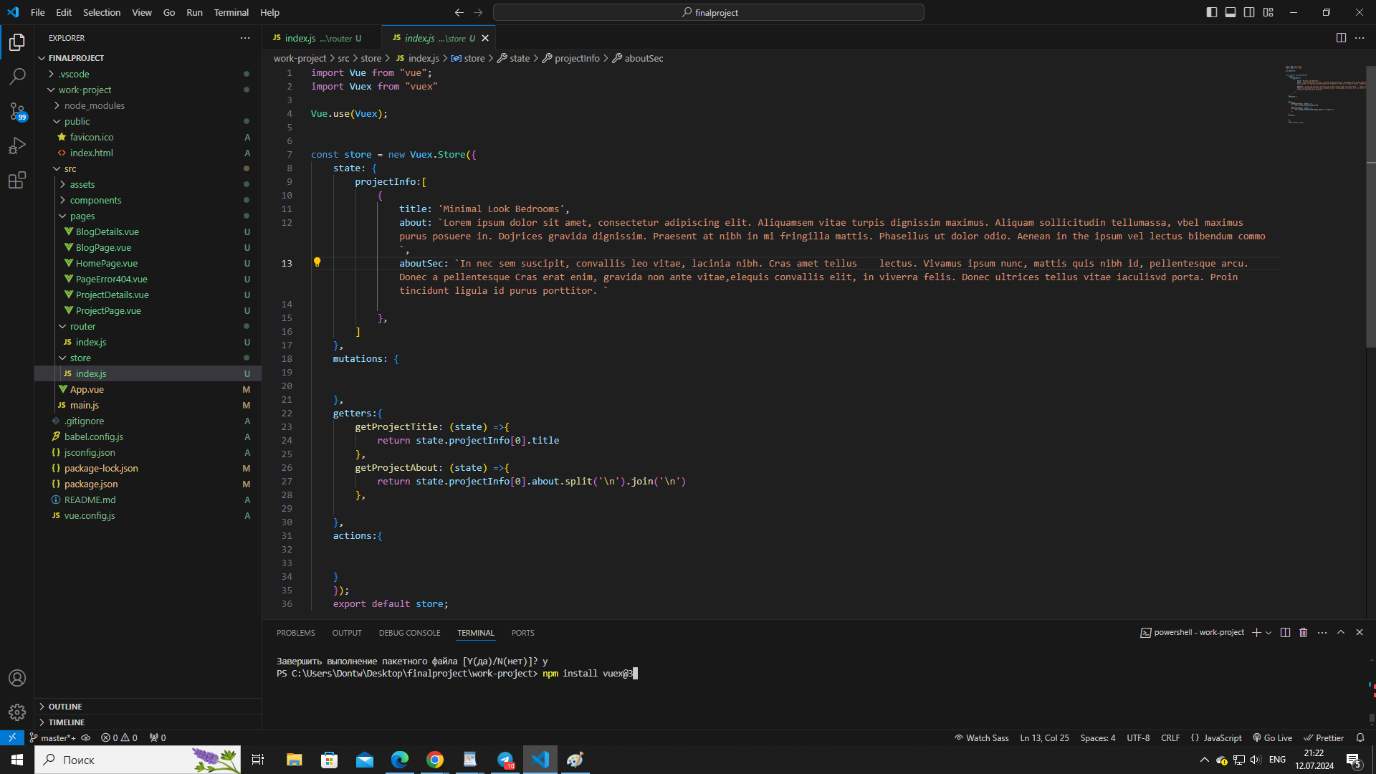
new Vue({

store,bootstrap,router,

render: h => h(App),

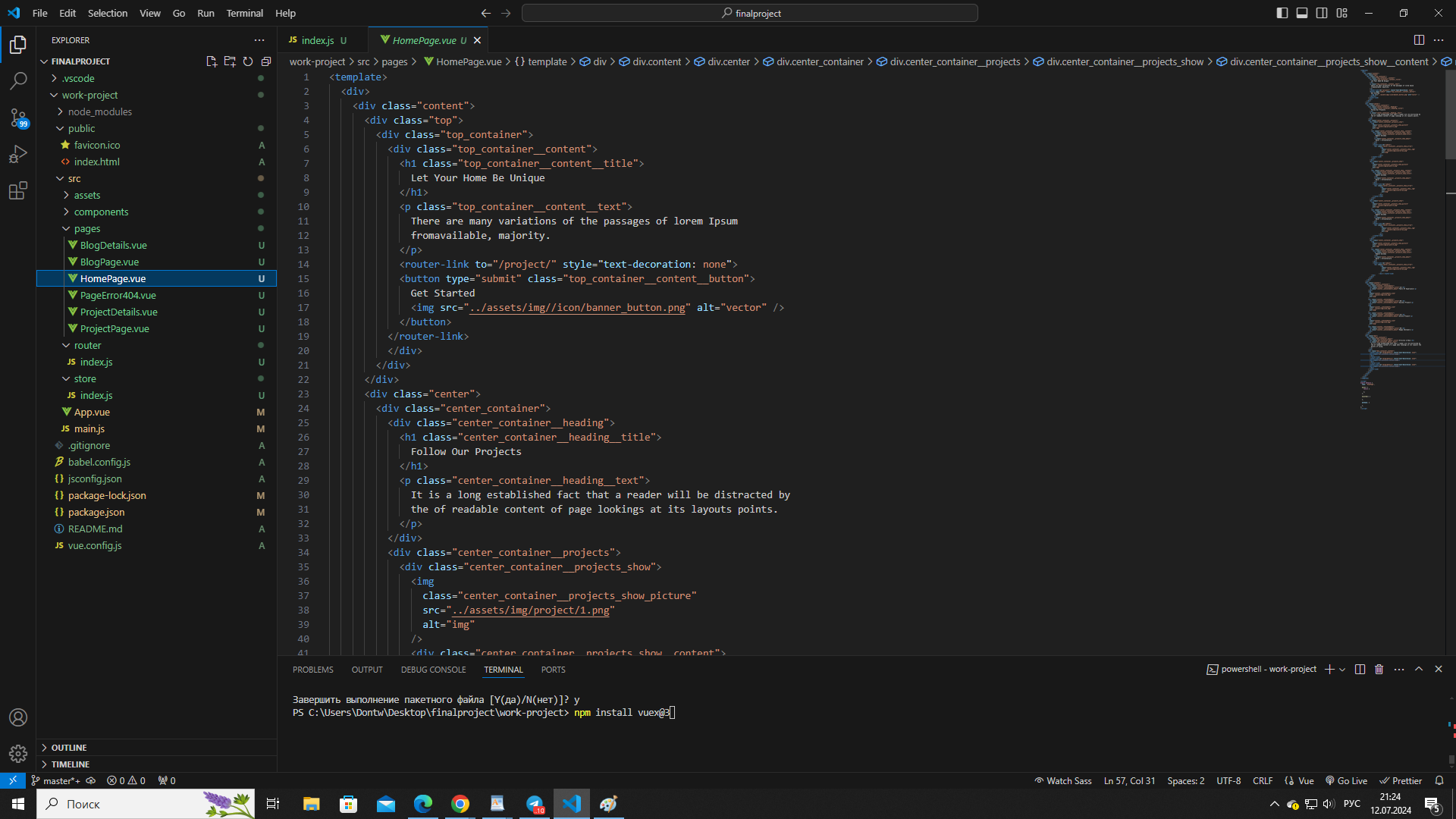
}).$mount('#app')

Далее устанавливаем VueX  
npm install vuex@3

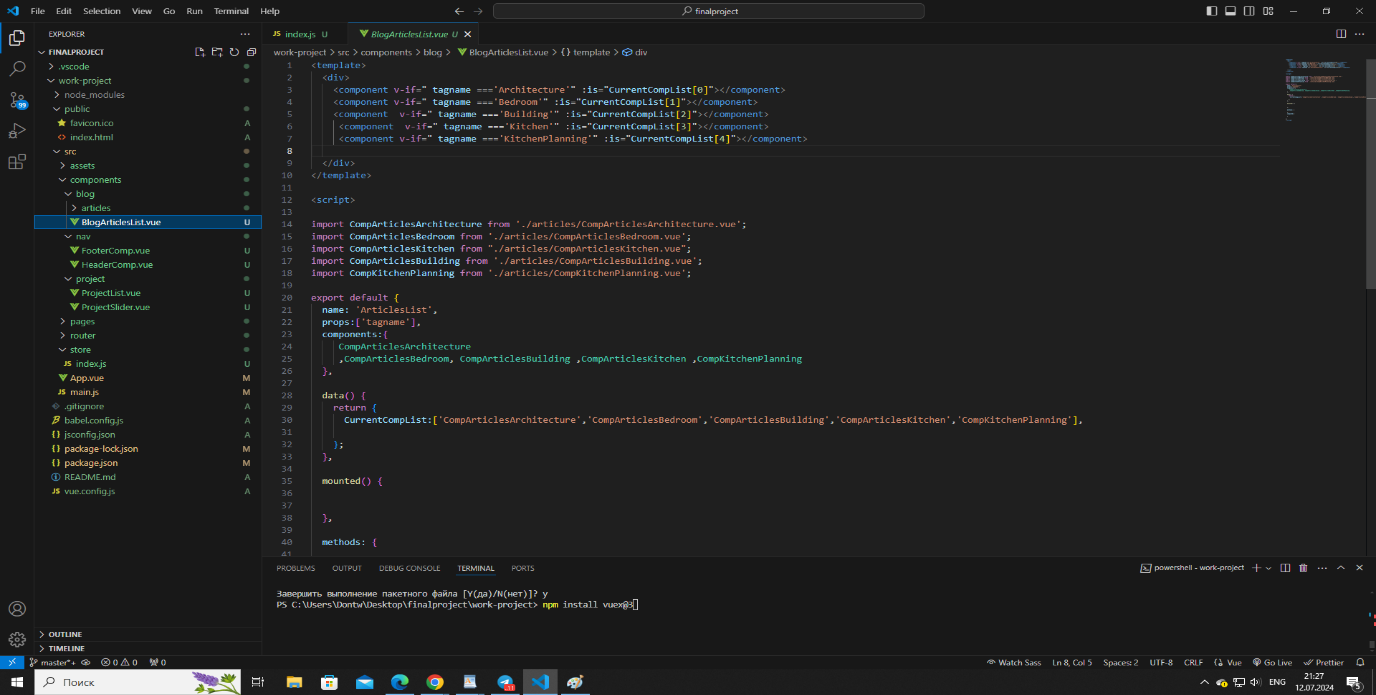


Приложение № 6

VueX используем для хранения данных мутаций и различной логики которая понадобиться нам в проекте

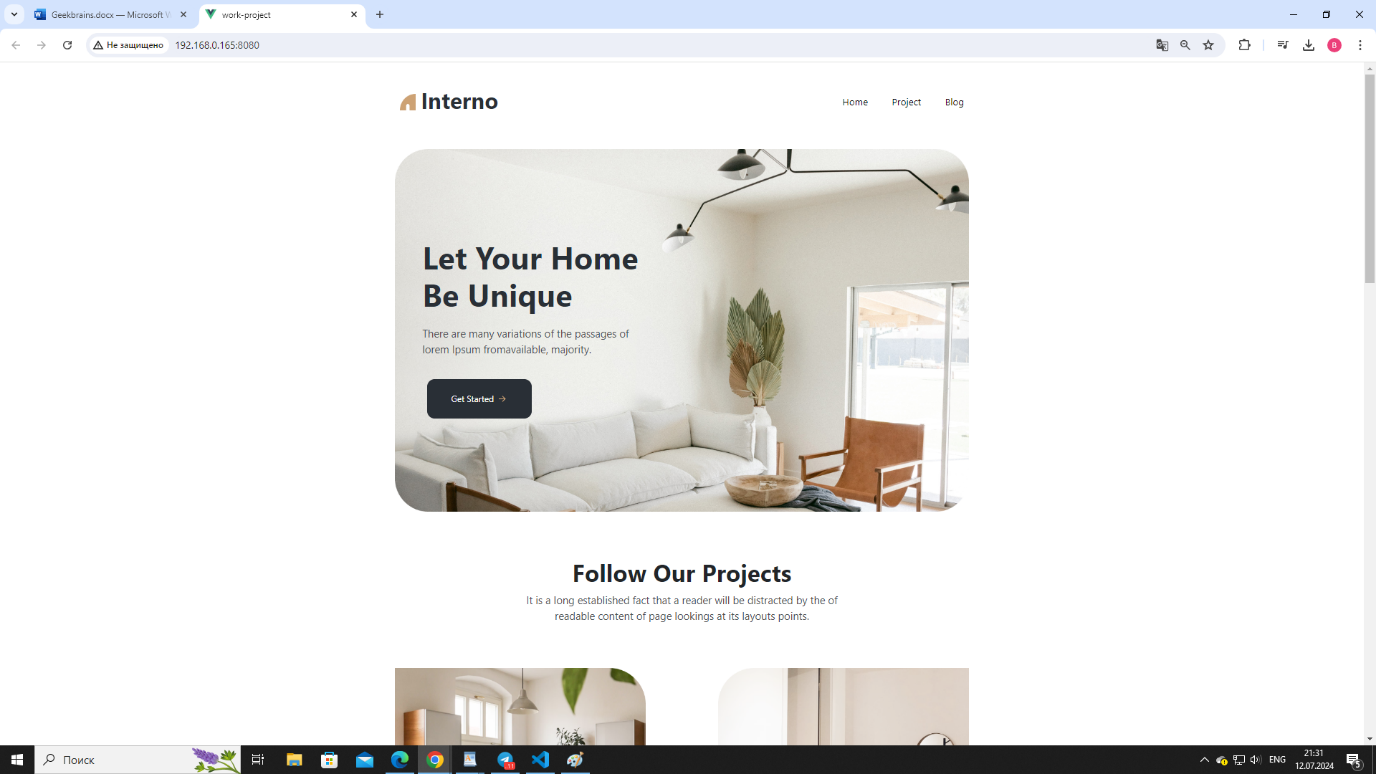
Приступаем к верстке отдельных страниц приведу пример страницы HomePage.  


Приложение №7

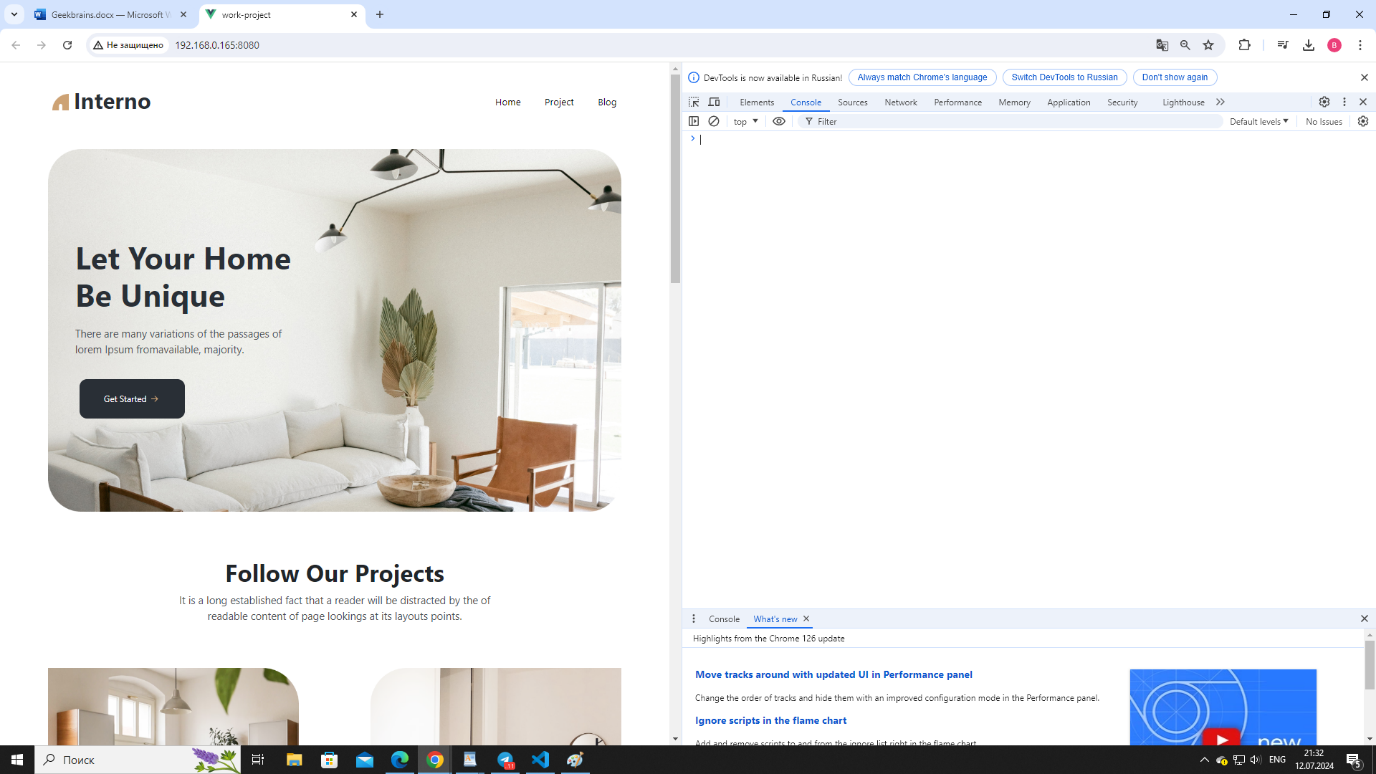
Разносим весь на проект на отдельные компоненты которые будут взаимодействовать между собой через Import  


Приложение №8

Когда все логика, стили, верстка закончены запускаем наш проект командой  
npm run serve

Переходим по адресу:  
- Network: <http://192.168.0.165:8080/>

Приложение № 9

Проверям консоль на наличие ошибок:  


Приложение № 10

Наш проект работает и не выдает ошибок.  
Провели ручное тестирование на наличие багов. После чего остается подключать наш сайт к серверной части и выкладывать в открытый доступ.  
Разработка клиентской части веб-сайта с использованием Vue.js завершена.

**Заключение.**

В ходе выполнения дипломной работы был проведен анализ теоретических основ разработки веб-сайтов с использованием фреймворка Vue.js и компонентов. Были изучены основные принципы работы с этим инструментом, его преимущества и недостатки, а также возможности интеграции с другими технологиями.

Была разработана структура веб-сайта, включающая в себя различные компоненты, такие как навигационное меню, формы обратной связи, галереи изображений и другие. Каждый компонент был тщательно протестирован и оптимизирован для обеспечения максимальной производительности и удобства использования.

Результаты исследования показали, что использование Vue.js позволяет значительно ускорить процесс разработки веб-сайтов благодаря своей гибкости и простоте использования. Кроме того, возможность создания компонентов делает процесс разработки более структурированным и масштабируемым.

Практическая значимость работы заключается в том, что разработанный веб-сайт может быть использован для различных целей, таких как продвижение товаров или услуг компании, информирование пользователей о новостях и событиях, а также для организации онлайн-встреч и конференций.

Для дальнейшего развития проекта рекомендуется провести дополнительное тестирование на большом количестве пользователей, чтобы выявить возможные проблемы с производительностью и удобством использования. Также стоит рассмотреть возможность интеграции с другими технологиями, такими как серверные языки программирования и базы данных.

В целом, цель дипломной работы была достигнута – был разработан функциональный веб-сайт с использованием Vue.js и его компонентов. Все поставленные задачи выполнены, а гипотеза подтверждена.

**Список используемой литературы**

Книга: Дмитрий Горшков . «Vue.js. От новичка до профессионала». Город: Москва, 2019г.

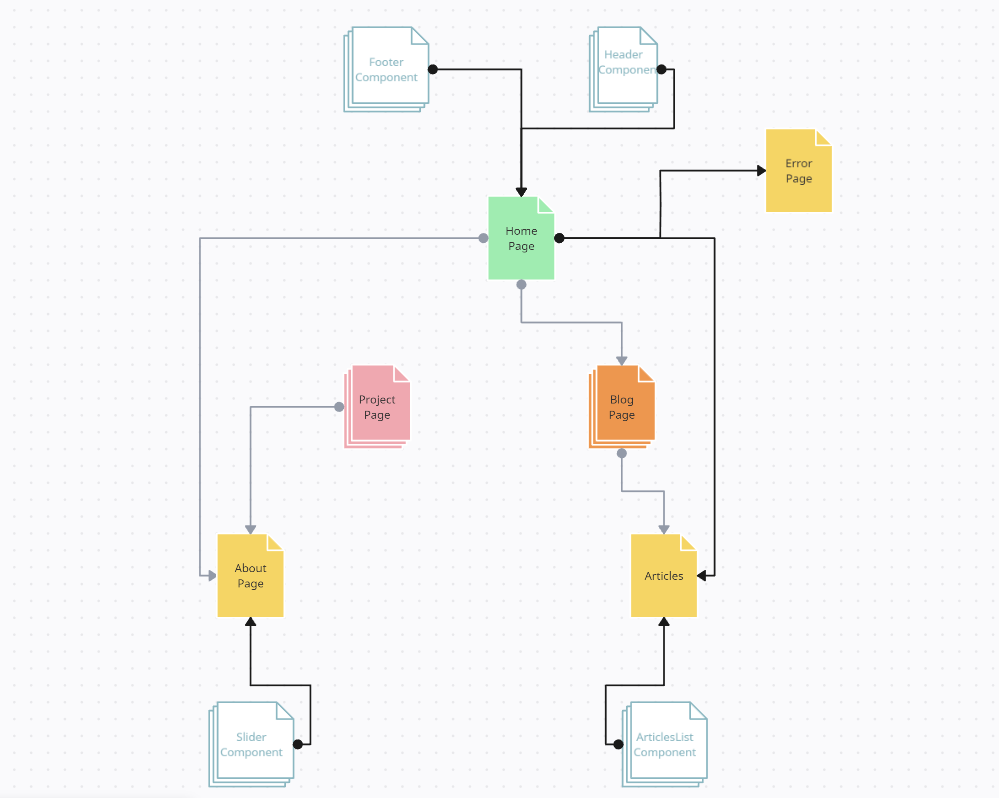
Книга: Erik Hanchett. "Vue.js: Up & Running". 2018г.

**Приложения**

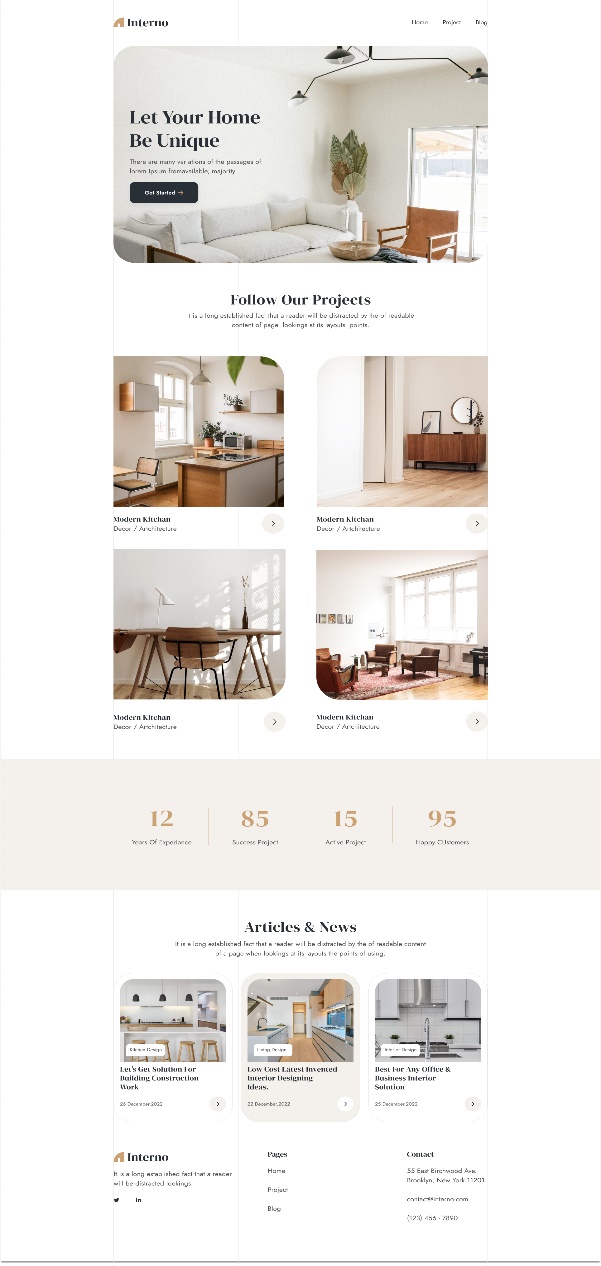
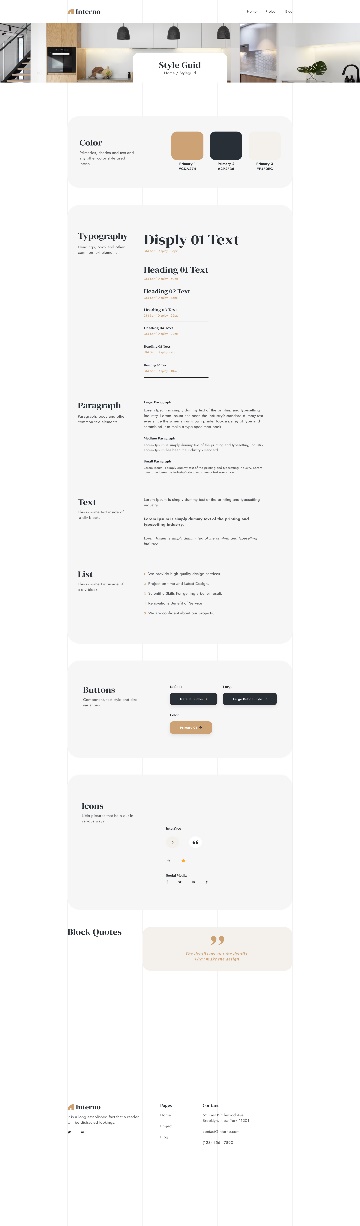
**Таблица 1.**

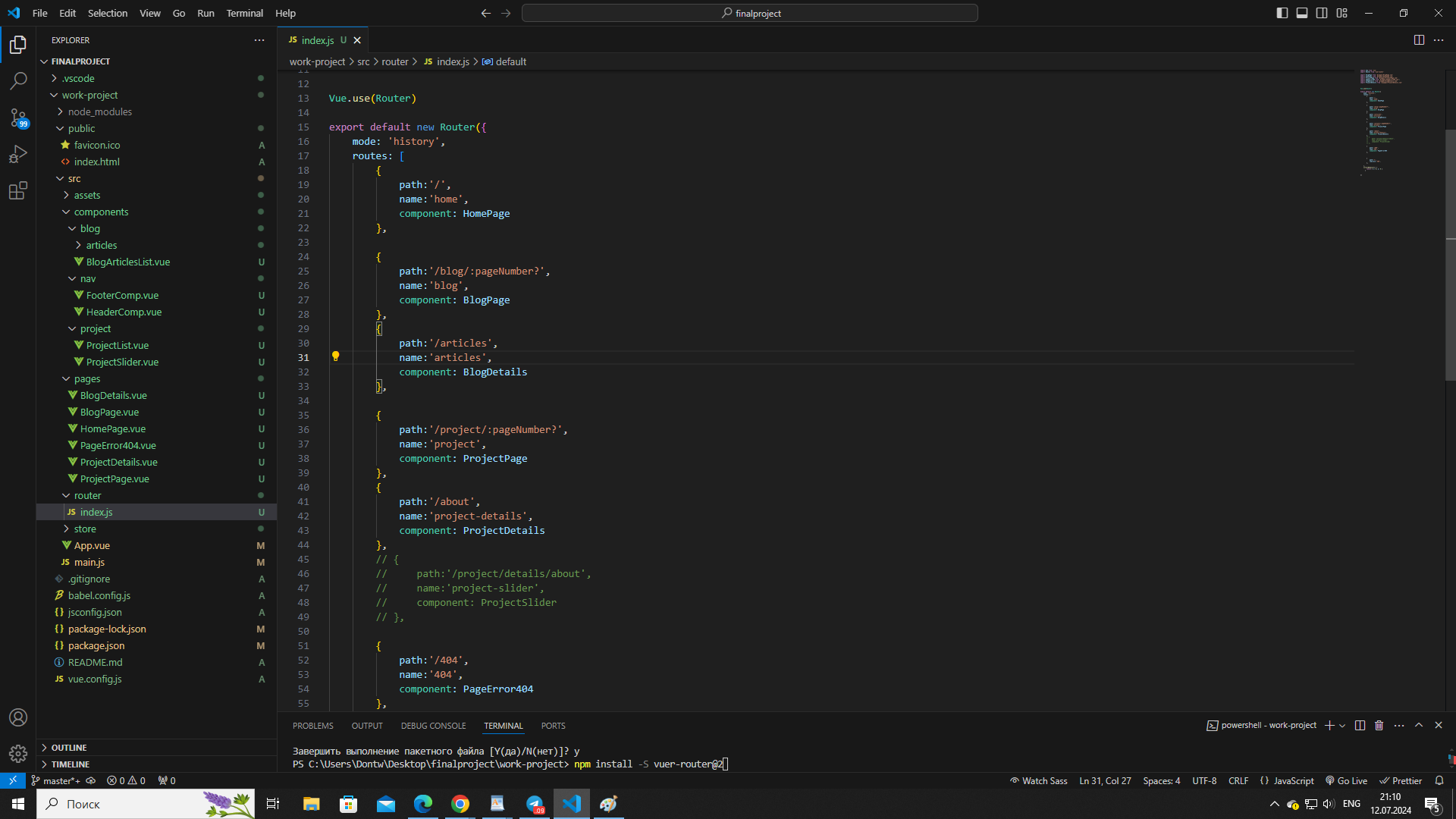
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Инструм-ент | Простота исполь-зования | Гибкость | Под-держка стандар-тов | Произ-водитель-ность | Под-держка сооб-щества |
| HTML/CSS | Средняя | Высокая | Высокая | Низкая | Высокая |
| Bootstrap | Высокая | Высокая | Средняя | Средняя | Высокая |
| React | Высокая | Средняя | Высокая | Высокая | Высокая |
| Vue.js | Высокая | Средняя | Средняя | Высокая | Средняя |

**Приложение №1**

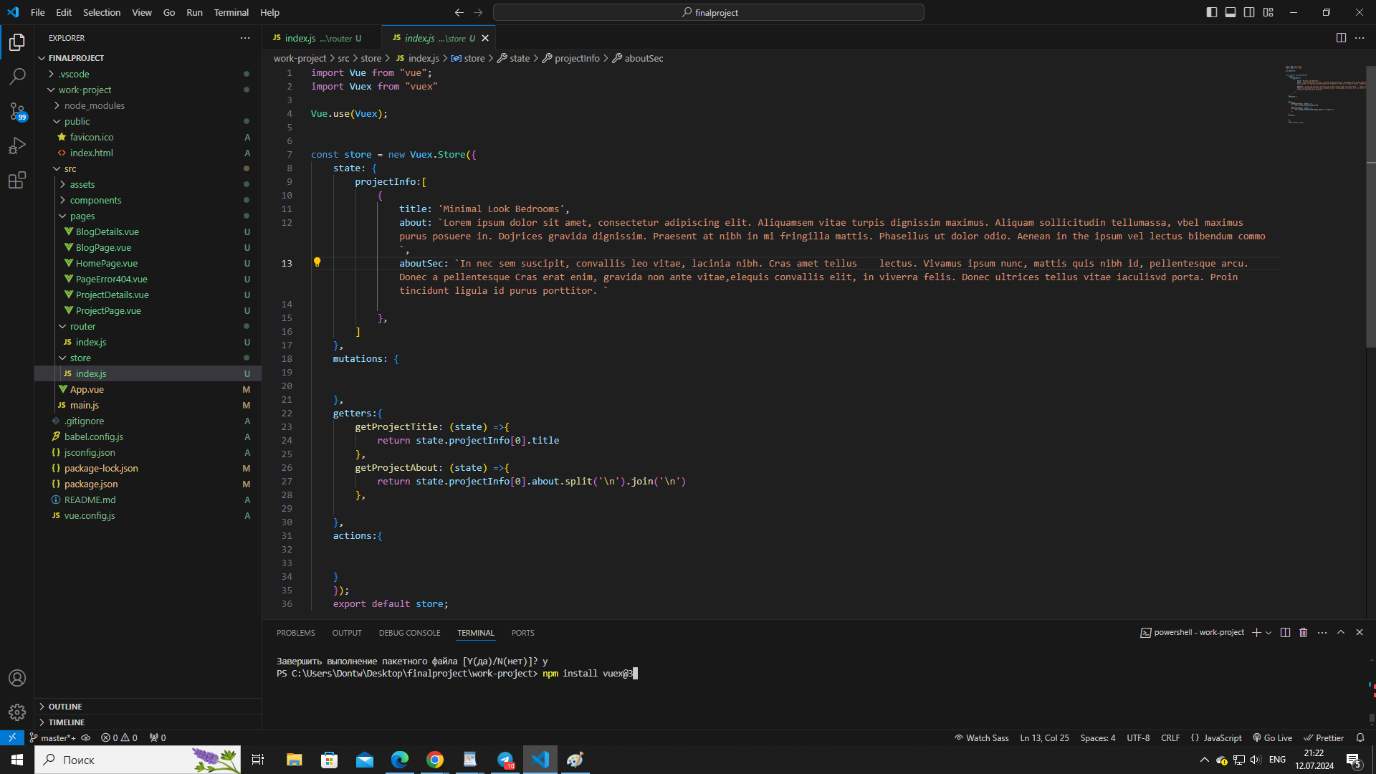


**Приложение №2**

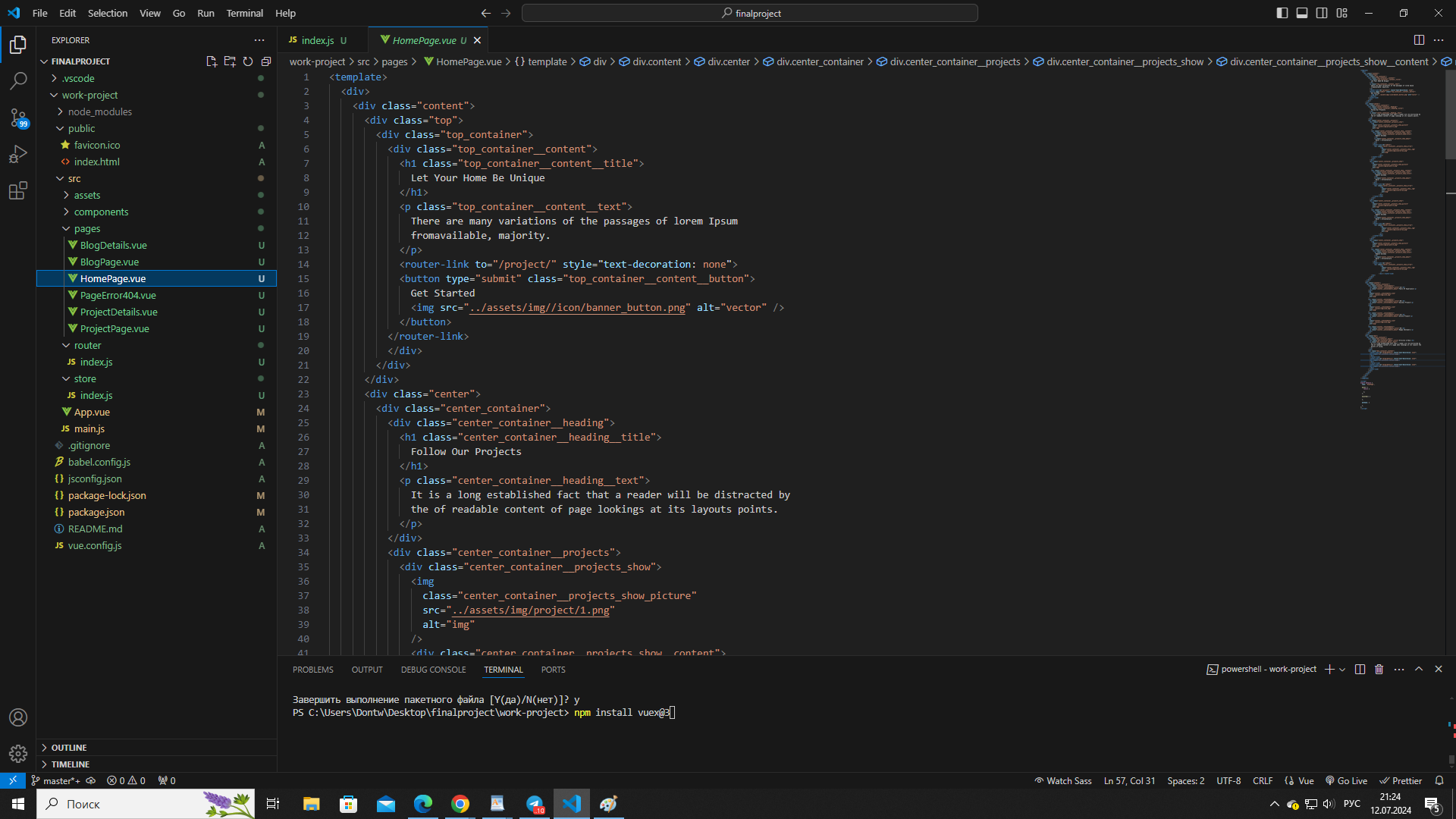


**Приложение №3**

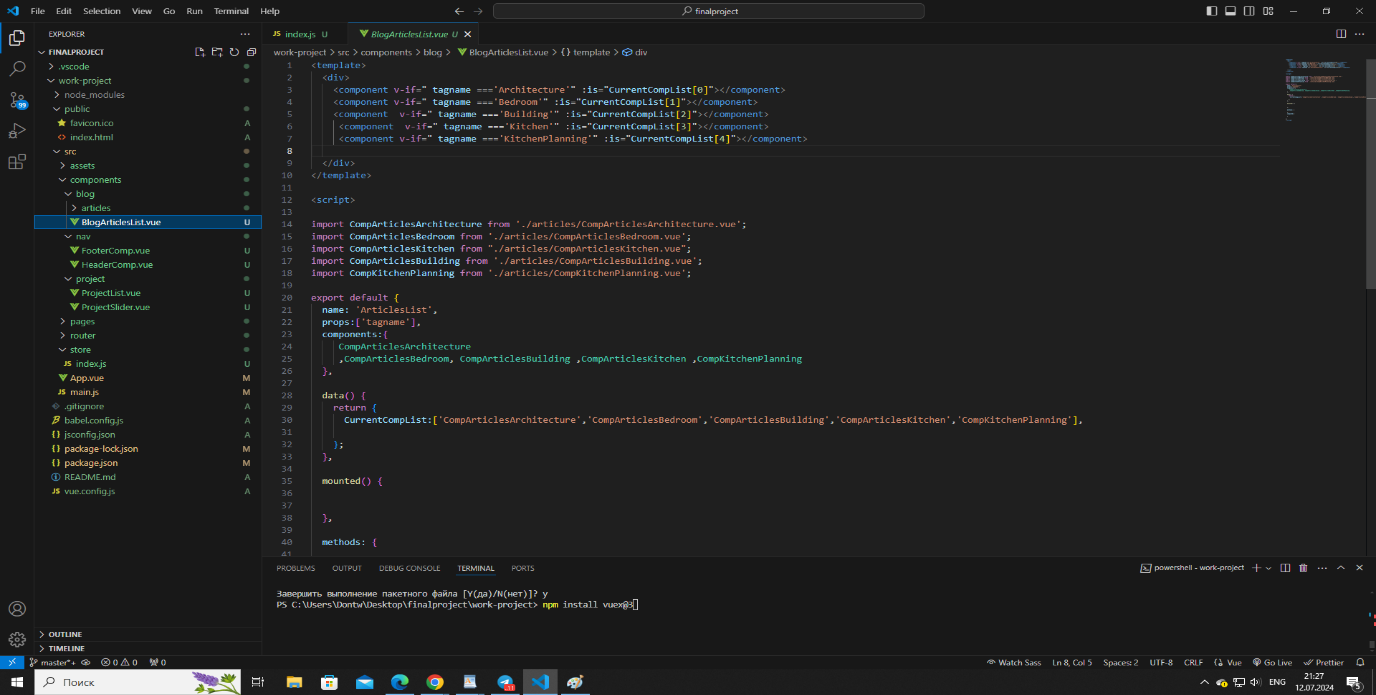
**Приложение №4**



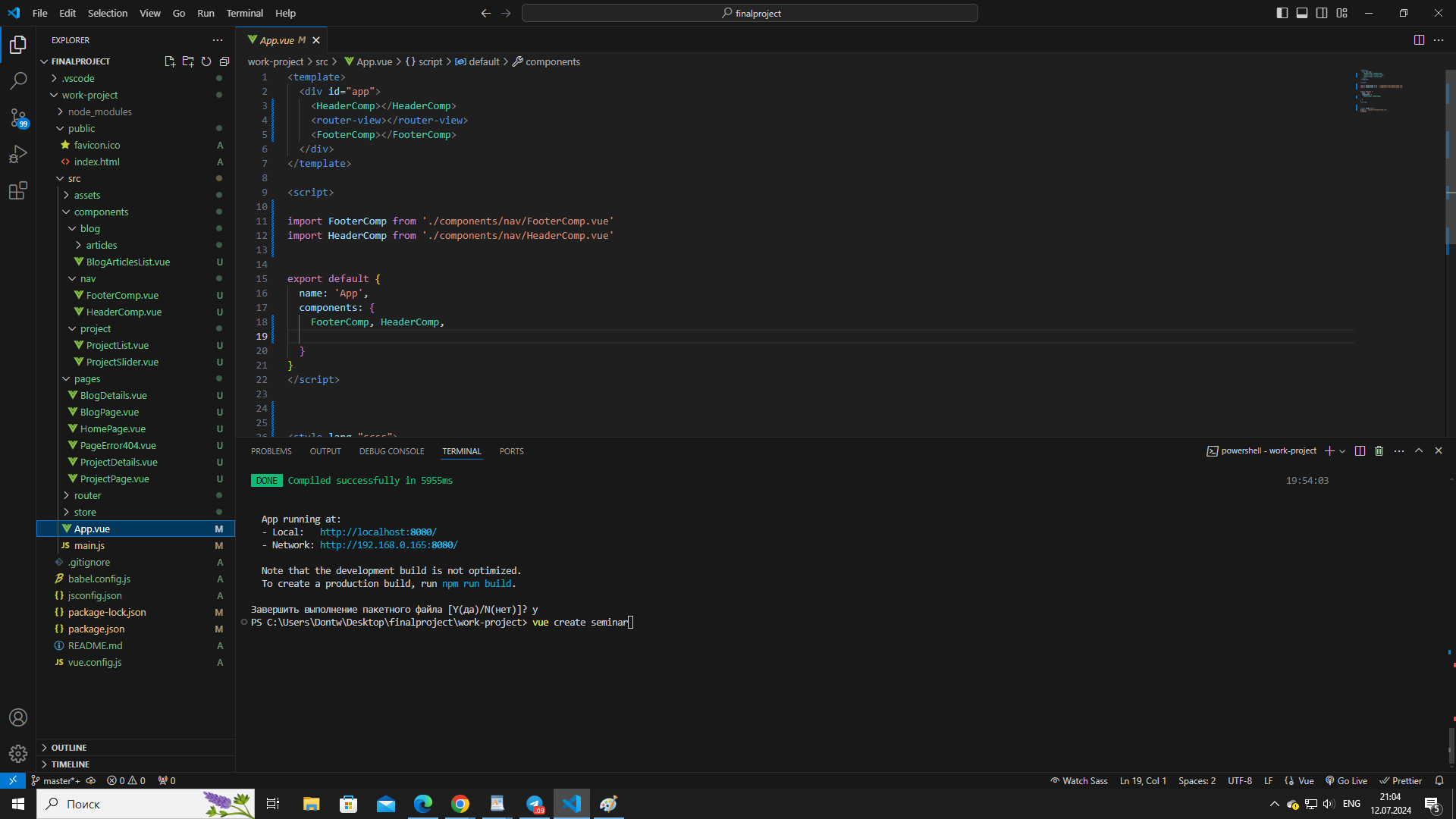
**Приложение №5**



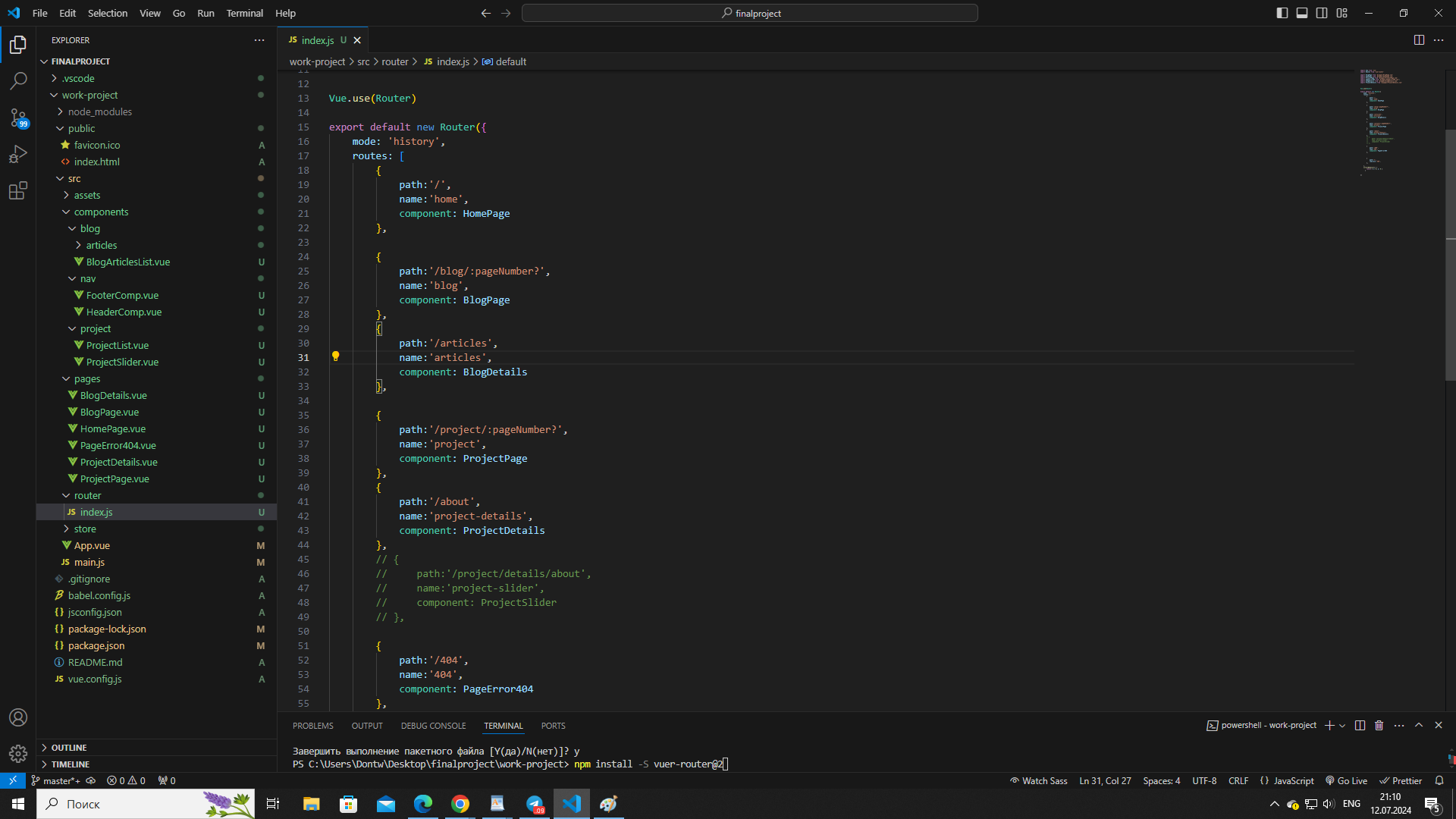
**Приложение №6**



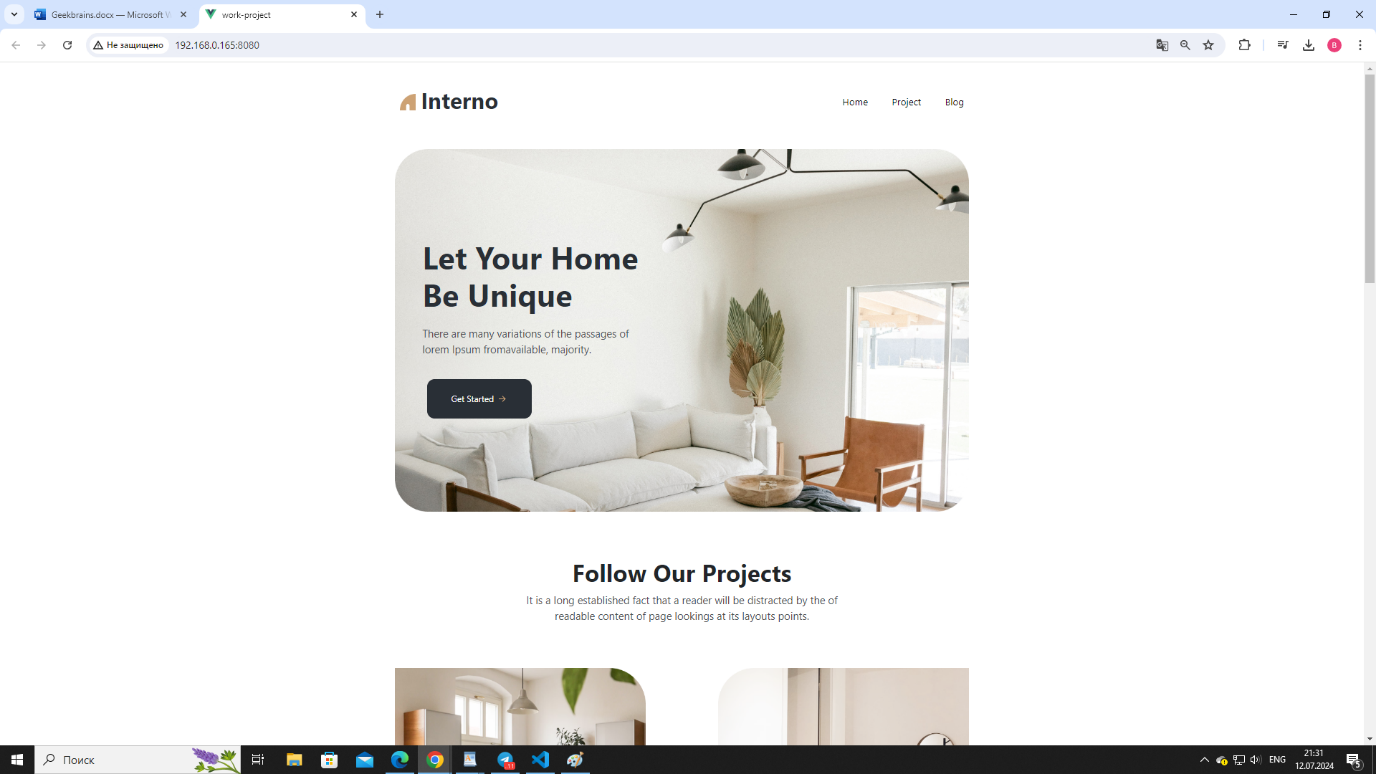
**Приложение №7**



**Приложение №8**



**Приложение №9**



**Приложение №10**

