Geekbrains

**Разработка MVP сайта-каталога**

**с использованием Vue.js**

Программа: Разработчик

Специализация: Frontend разработка (Vue.js)

Евтеева Анастасия Алексеевна

Москва

2024

**Содержание**

[Введение 2](#_Toc162361907)

[1. Теоретическая часть 4](#_Toc162361908)

[1.1 Основные принципы работы с Vue.js и его преимущества 4](#_Toc162361909)

[1.2 Сравнение Vue.js с другими фронтенд-технологиями 6](#_Toc162361910)

[1.3 Архитектура приложения на Vue.js 6](#_Toc162361911)

[1.4 Возможности использования компонентов и маршрутизации в Vue 7](#_Toc162361912)

[2. Практическая часть 9](#_Toc162361913)

[2.1 Настройка среды разработки 9](#_Toc162361914)

[2.2 Дизайн веб-интерфейса 13](#_Toc162361915)

[2.3 Разработка архитектуры веб-сайта 14](#_Toc162361916)

[2.4 Подключение файлов стилей 16](#_Toc162361917)

[2.5 Создание компонентов на Vue.js 17](#_Toc162361918)

[2.6 Настройка и работа с хранилищем Vuex 19](#_Toc162361919)

[2.7 Использование встроенных директив Vue.js 23](#_Toc162361920)

[2.8 Настройка маршрутизации в проекте 31](#_Toc162361921)

[2.9 Настройка ссылок в проекте. Навигация 34](#_Toc162361922)

[Зключение 37](#_Toc162361923)

[Список использемой литературы 39](#_Toc162361924)

[Приложения 40](#_Toc162361925)

# Введение

Темой дипломной работы, выполненной по специализации "Frontend разработка (Vue.js)", является "Разработка MVP сайта-каталога с использованием Vue.js". Проект представляет собой интерактивный веб-сайт каталог, который будет содержать информацию о различных марках и моделях электроавтомобилей.

Выбор темы проекта обусловлен актуальностью развития технологий фронтенд-разработки, особенно использование современного JavaScript фреймворка Vue для создания динамичных и отзывчивых пользовательских интерфейсов.

Целью данного дипломного проекта является разработка интерактивного веб-сайта каталога на Vue.js, который предоставит удобный инструмент для просмотра и поиска информации.

План работы включает в себя следующие этапы:

1. Проектирование интерфейса и структуры сайта;
2. Разработка архитектуры приложения и выбор подходящих технологий;
3. Интеграция с API для получения информации;
4. Разработка компонентов и функционала сайта на Vue.js..

Дипломный проект направлен на создание удобного и интуитивно понятного интерфейса для просмотра и поиска информации в каталоге, что позволит пользователям быстро и эффективно находить необходимую информацию. Предоставление эффективного и удобного способа доступа к информации о товарах повышает качество обслуживания и уровень удовлетворенности пользователя.

Для реализации проекта планируется использовать **инструменты разработки**, такие как Visual Studio Code, Git[[1]](#footnote-1) для контроля версий, Vue CLI для создания и настройки проекта.

**Технологический стек** проекта будет включать в себя: Vue.js в качестве основного фреймворка для работы с пользовательским интерфейсом, Vuex для управления состоянием приложения, Vue Router для маршрутизации, Fetch API для обмена данными, а также CSS и HTML для стилизации и структурирования веб-страниц.

**Состав команды:**

Евтеева Анастасия Алексеевна (Frontend-разработчик (Vue.js)).

# Теоретическая часть

## 1.1 Основные принципы работы с Vue.js и его преимущества

Vue.js - это современный фреймворк JavaScript, который позволяет создавать интерактивные веб-приложения с помощью простого и интуитивно понятного синтаксиса. Он широко применяется для разработки одностраничных приложений (SPA) и веб-интерфейсов.

Vue.js применяет несколько основных принципов, которые делают работу с этим фреймворком удобной и эффективной:

1. Декларативность: Vue.js позволяет описывать, что должно произойти, на основе состояния приложения, используя HTML-шаблоны. Вы просто говорите приложению, что хотите отобразить в браузере, на основе состояния Vue и он обновляет DOM за вас.
2. Компонентный подход: Во Vue.js приложения строятся из модульных, переиспользуемых компонентов. Вы можете создать компоненты для всего - заголовка, секции навигации, частей формы. Компоненты могут включать другие компоненты, что позволяет создавать сложные интерфейсы из простых частей. Проектирование с использованием компонентов позволяет разделить работу над проектом и делает код более легким для понимания и поддержки.
3. Реактивность: Vue.js позволяет легко и эффективно следить за изменениями стейта приложения и автоматически обновлять DOM при изменении данных.
4. Двунаправленная связь данных: Vue.js обеспечивает простой способ связывания данных между моделью и представлением, что позволяет автоматически обновлять пользовательский интерфейс при изменении данных
5. Директивы: Vue предлагает множество встроенных директив для управления DOM-элементами, обработки событий и условного отображения.
6. Управление состоянием: для управления состоянием приложения в Vue.js часто используется библиотека Vuex, которая предоставляет инструменты для централизованного хранения и управления данными приложения.
7. Маршрутизация: в Vue.js имеется возможность добавления маршрутизации в приложение с помощью плагина Vue Router, который позволяет создавать многостраничные приложения и управлять навигацией.

Преимущества работы с Vue.js включают в себя:

* Простота: Vue.js очень интуитивен и прост в изучении, особенно для разработчиков, знакомых с HTML и JavaScript.
* Гибкость: Vue.js может использоваться для создания как небольших, так и крупных масштабируемых приложений. Он прекрасно работает в сочетании с другими библиотеками и инструментами.
* Производительность: Vue.js обеспечивает высокую производительность приложений благодаря интеллектуальному отслеживанию зависимостей и асинхронному обновлению DOM.
* Поддержка сообщества: Vue.js имеет активное и быстрорастущее сообщество разработчиков, которые постоянно работают над улучшением фреймворка и создают плагины для расширения его функциональности.
* Большая информационная база: есть масса справочных материалов и обучающих ресурсов по работе с этим фреймворком.

В целом, Vue.js характеризуется гибкостью, функциональностью и несложностью в использовании, как для новичков, так и для опытных разработчиков, что обусловливает его непрерывно растущую популярность.

## 1.2 Сравнение Vue.js с другими фронтенд-технологиями

Vue.js – это современный JavaScript-фреймворк, который за короткое время завоевал большую популярность среди разработчиков. Однако на рынке веб-разработки существует несколько других ведущих фреймворков, таких как Angular и React, которые также обладают своими сильными сторонами.

Angular[[2]](#footnote-2) — это полноценный фреймворк, созданный Google. Он может быть сложнее для изучения по сравнению с Vue.js из-за своего объемного API и сложной системы типов TypeScript. В то время как Vue.js обеспечивает гораздо более простой и понятный синтаксис, делая его более доступным для начинающих разработчиков. Но Angular превосходит Vue.js в плане производительности для больших приложений благодаря автоматическому трекингу изменений в модели данных.

React[[3]](#footnote-3) — это библиотека для создания пользовательских интерфейсов, созданная Facebook, которая, в отличие от Angular и Vue.js, не является полноценным фреймворком. Это означает, что разработчики должны решить о выборе архитектуры приложения и других вспомогательных библиотек. Vue.js с другой стороны предоставляет продуманный набор инструментов "из коробки". Однако в отношении производительности оба фреймворка показывают похожие результаты.

Таким образом, выбор фреймворка в значительной степени зависит от конкретных условий и требований проекта, а также от опыта и предпочтений команды разработчиков. Несмотря на отличия, все эти фреймворки способны создавать мощные и высокопроизводительные веб-приложения, каждый со своим уникальным набором преимуществ.

## 1.3 Архитектура приложения на Vue.js

Пример возможной архитектуры веб-сайта на Vue.js:

1. Main App Component (Главный компонент приложения):

Главный компонент приложения, который содержит другие компоненты и управляет их отображением.

1. Компоненты:

Компоненты для многократного использования и повторного использования на разных страницах (заголовок, боковое меню, карусель, формы и т.д.)

1. Маршрутизация (Routing):

Vue Router для организации навигации по страницам приложения, определение маршрутов для различных компонентов и страниц.

1. Управление состоянием (State Management)

Vuex для управления состоянием приложения, таким как данные, которые нужно обмениваться между компонентами. Централизованное хранилище позволит легко управлять данными приложения.

1. API Calls:

- Для обмена данными с сервером используют библиотеку Axios или Fetch API.

## 1.4 Возможности использования компонентов и маршрутизации в Vue

Vue.js примечателен своей системой создания и использования компонентов, а также встроенной библиотекой маршрутизации, которая значительно упрощает разработку сложных одностраничных приложений (SPA).

**Возможности использования компонентов в Vue.js:**

1. Многократное использование:

Компоненты в Vue.js разработаны так, чтобы быть многократно используемыми в разных частях приложения без изменений.

1. Локальное и глобальное регистрирование:

Компоненты могут быть зарегистрированы глобально (доступны во всем приложении) или локально (доступны только в определенных компонентах).

1. Props и события:

Компоненты могут принимать данные через свойства (props) и эмитировать события для обращения наверх по иерархии ($emet).

1. Слоты:

С помощью слотов можно расширять содержание компонентов, передавая вложенные элементы из родительского компонента.

**Возможности маршрутизации:**

1. Базовая маршрутизация:

Vue Router позволяет определить маршруты для вашего приложения и связать их с компонентами.

1. Управление историей браузера:

Vue Router поддерживает режим «history» HTML5, что позволяет использовать нормальные URL-адреса без хэша (#).

1. Передача параметров:

Vue Router позволяет передавать параметры через URL-адреса маршрутов, например, идентификаторы объектов.

1. Вложенные маршруты:

Vue Router поддерживает вложенные маршруты, что позволяет создавать более сложные структуры приложений.

1. Переходы и анимации:

Благодаря встроенной поддержке переходов в Vue.js, можно легко добавить анимацию при переключении маршрутов.

Вместе эти функции обеспечивают мощные и гибкие инструменты для построения сложных и производительных веб-приложений.

# Практическая часть

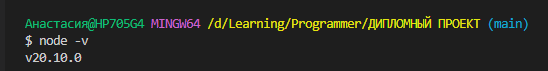
## 2.1 Настройка среды разработки

Для настройки среды разработки сайта на фреймворке Vue.js Вам понадобится выполнить несколько шагов:

1. Установите Node.js:

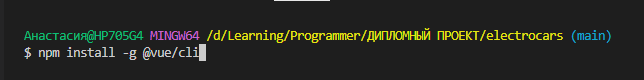
Скачайте и установите Node.js с официального сайта[[4]](#footnote-4): Node.js включает в себя npm (Node Package Manager), который понадобится для установки необходимых пакетов.

Проверьте его работоспособность введя в терминале или командной строке команду. Если все успешной установилось Вы увидите текущую версию Node.js.



1. Установка Vue CLI

Следуем инструкции с официального сайта [[5]](#footnote-5)Vue.js.

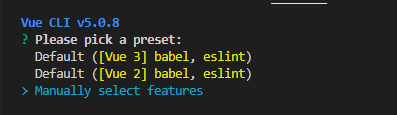


1. Создание проекта

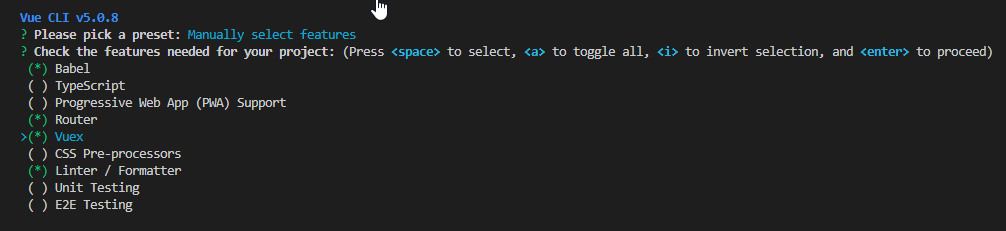
Для создания нового проекта вводим в терминале или командной строке команду:



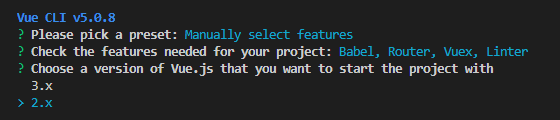
Далее выбираем необходимые опции. Вы можете выбрать как дефолтные пакеты, так и составить пакет самостоятельно. Я выберу последний вариант.

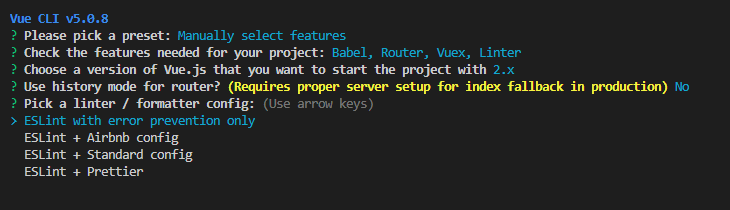


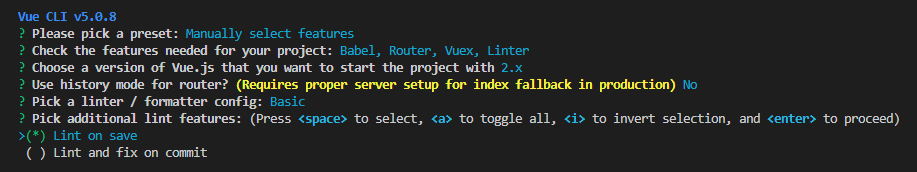
Как говорилось ранее для разработки будут использоваться такие плагины как Vue Router для маршрутизации и Vuex для управления данными. Их возможно установить на данном этапе при создании проекта, выбрав соответствующие опции.



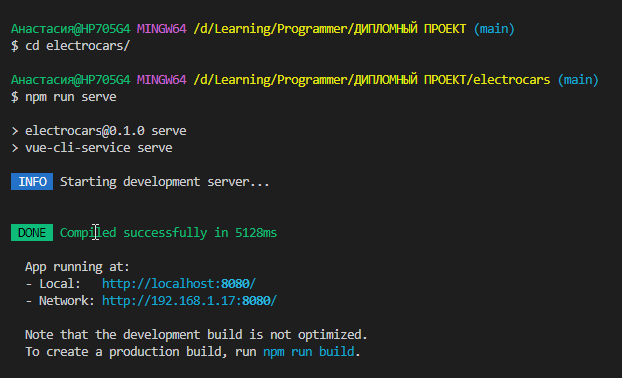
Для своего проекта буду использовать Vue 2.





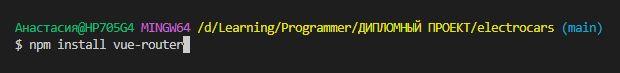


После установки пакета нам необходимо перейти в директорию проекта и запустить сервер. Если все было сделано верно, Вы получите рабочие ссылки на свой проект.



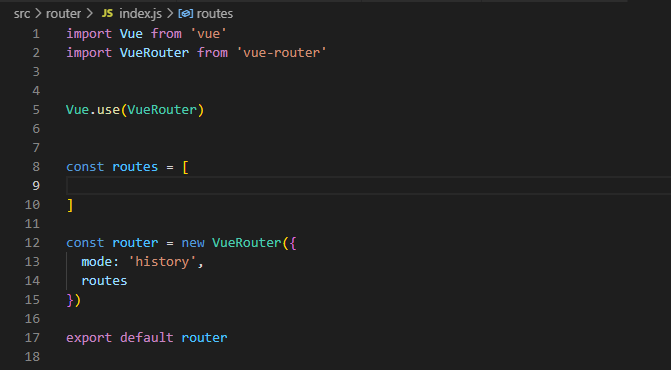
Если же Вы выбрали дефолтные пакеты, то Vue Router и Vuex нужно будет установить отдельно.

Для установки Vue Router вводим в терминале или командной строке команду:

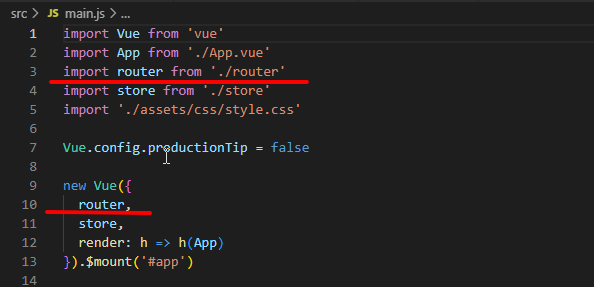


После установки необходимо импортировать модуль в проект.

Для начала в папке «src» создадим папку «Router», в которой разместим файл index.js со следующим содержимым:

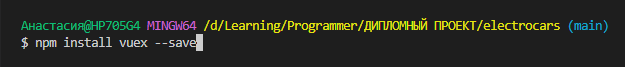


Далее в главном файле main.js добавляем следующие строки:

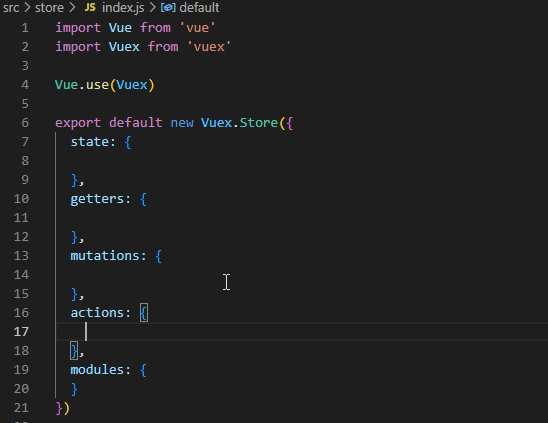


Подключение Vue Router к проекту завершено.

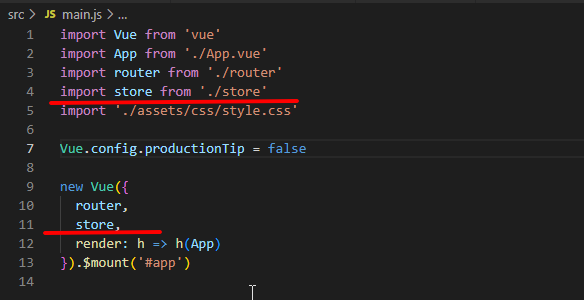
Для установки Vuex вводим в терминале или командной строке команду:



Далее в папке «src» создадим папку «store», в которой разместим файл index.js со следующим содержимым:



Далее в главном файле main.js добавляем следующие строки:



Подключение Vuex к проекту завершено.

## 2.2 Дизайн веб-интерфейса

Для того чтобы начать frontend разработку нам потребуется дизайн нашего будущего сайта или приложения.

Дизайн включает в себя разработку внешнего вида веб-сайта - от цветовой схемы и шрифтов до компоновки элементов на странице и графических элементов. Цель дизайна интерфейса состоит в создании привлекательного и удобного для использования внешнего вида, который ориентирован на потребности пользователей.

Возможно у Вас есть уже готовый дизайн, выполненный профессионалом или же Вы рассматриваете шаблонный дизайн, который можно скачать бесплатно на просторах Internet – любой из вариантов подходит, главное подобрать шаблон, который будет соответствовать задумке Вашего проекта.

Я буду использовать Макет[[6]](#footnote-6) в Figma, который как мне кажется подходит для сайта-каталога автомобилей.

## 2.3 Разработка архитектуры веб-сайта

При проектировании архитектуры веб-сайта на Vue.js основное внимание следует уделить компонентному подходу.

Изучив выбранный макет сайта, я выделила следующие компоненты:

1. Главный компонент

В моем случае главный компонент будет содержать в себе навигацию и футер сайта, так как эти элементы дублируются на все страницы и никак не изменяются, в отличие от баннера.

1. Компоненты страниц

При переходе по url страницы в главном компоненте будет меняется только контентная часть. Эту часть мы и вынесем в отдельные компоненты постранично.

У меня получились следующие компоненты страниц:

* HomePage.vue – контент главной страницы;
* BrandsPage.vue – контент страницы каталога;
* ModelsPage.vue – контент страницы марки;
* SingleModelPage.vue – контент страницы карточки автомобиля;
* AboutPage.vue – контент страницы о компании;
* ArticlesPage.vue – контент страницы списка статей;
* SingleArticlePage.vue – контент страницы статьи;
* NotFoundPage.vue – контент страницы ошибки 404.

Созданные страницы разместим в проекте по пути src\views.

1. Вложенные компоненты

Все главные компоненты будут состоять из других компонентов. Выделим основные:

* Header;
* Footer;
* Баннер;
* Хлебные крошки;
* Блок преимуществ;
* Блок «Почему электрокары?»;
* Блок интерьера и экстерьера;
* Форма обратной связи;
* Блок марок/моделей;
* Галерея;
* Блок статей;
* Блок отзывов

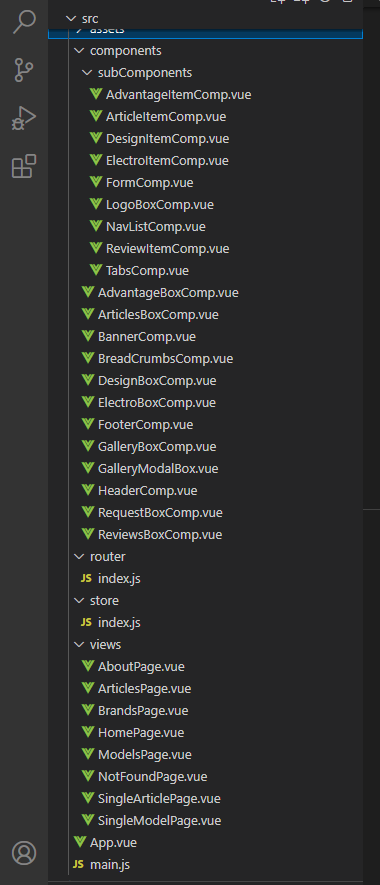
Это основные компоненты, которые я предварительно могу выделить в своем проекте. Их мы разместим по пути src\components.

Эти компоненты могут так же состоять из других компонентов, например:

* «Header» может состоять из подкомпонентов «Логотип» и «Навигация», так как эти элементы так же используются в подвале сайта;
* «Блок статей» будет состоять из отдельных компонентов «Превью статьи»
* «Блок марок/моделей» будет состоять из отдельных компонентов «Превью марки/модели»
* «Блок отзывов» будет состоять из отдельных компонентов «Отзыв».

Эти компоненты мы создадим в проекте по пути src\components\subcomponents.

После создания необходимых компонентов файловая структура проекта выглядит так:



## 2.4 Подключение файлов стилей

Существует несколько способов подключения стилей.

1. Локальное подключение

Можно прописывать стили внутри каждого компонента в соответствующем теге;

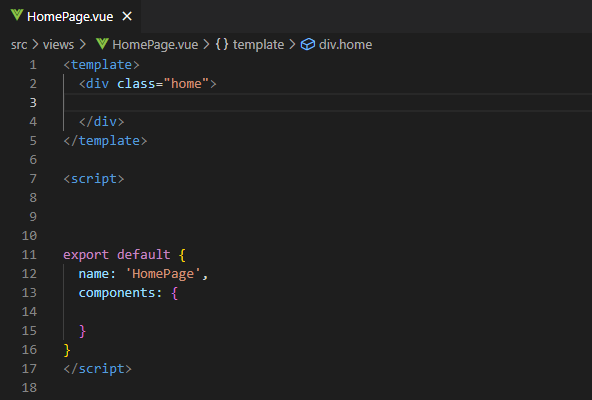
1. Глобальное подключение

Можно вынести стили в отдельную папку и импортировать их в главный файл main.js.

Для подключения стилей в своем проекте я буду использовать второй вариант. Считаю, что если разделить стили и компоненты, то код будет более читаемым и поддерживаемым.

## 2.5 Создание компонентов на Vue.js

Создадим компонент главной страницы. Внутри компонента необходимо добавить следующую структуру:

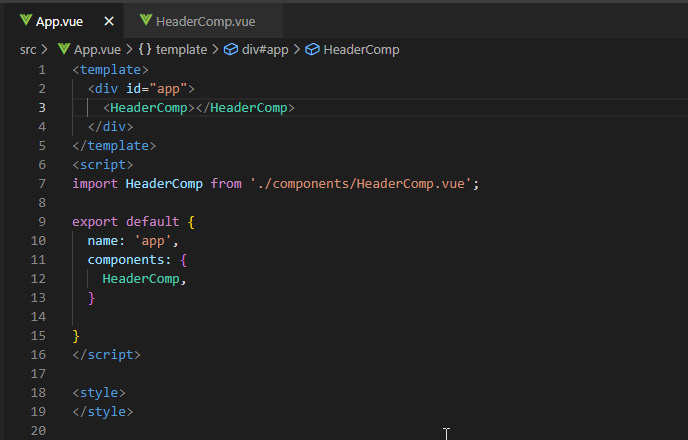


Внутри шаблона (template) добавляем html код страницы. Обязательное условие – наличие общего родительского блока, в моем случае div с классом «home». Внутри тега script необходимо объявить имя компонента, в моем случае это «HomePage». А стили для элемента прописываются внутри тега style.

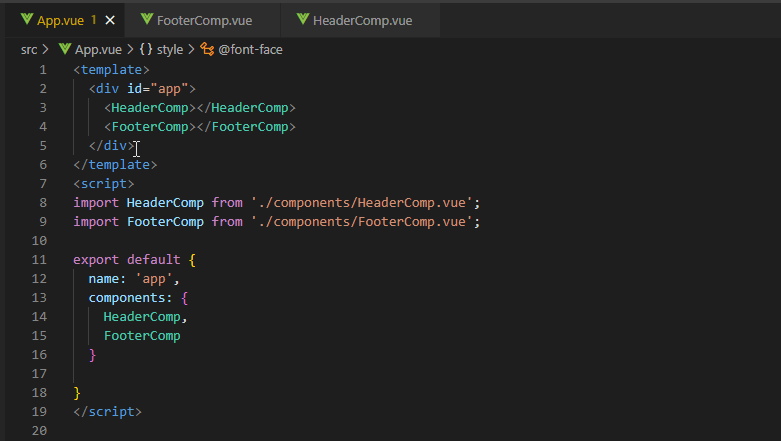
Данная структура является обязательной и все компоненты создаются по данному шаблону.

Теперь создадим компонент «HeaderComp». Так как данный компонент будет отображаться на всех страницах расположим его в корневом компоненте App.vue (в дальнейшем с помощью маршрутизации будем менять контентную часть в зависимости от url).

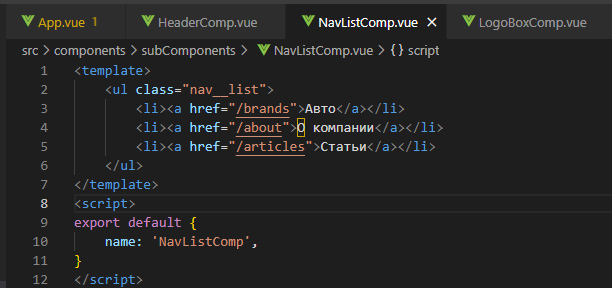
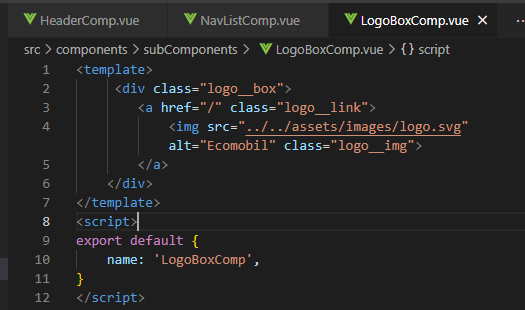
Для подключения компонента его необходимо импортировать в родительский компонент (в данном случае в App.vue) и вызвать в шаблоне с помощью тега.

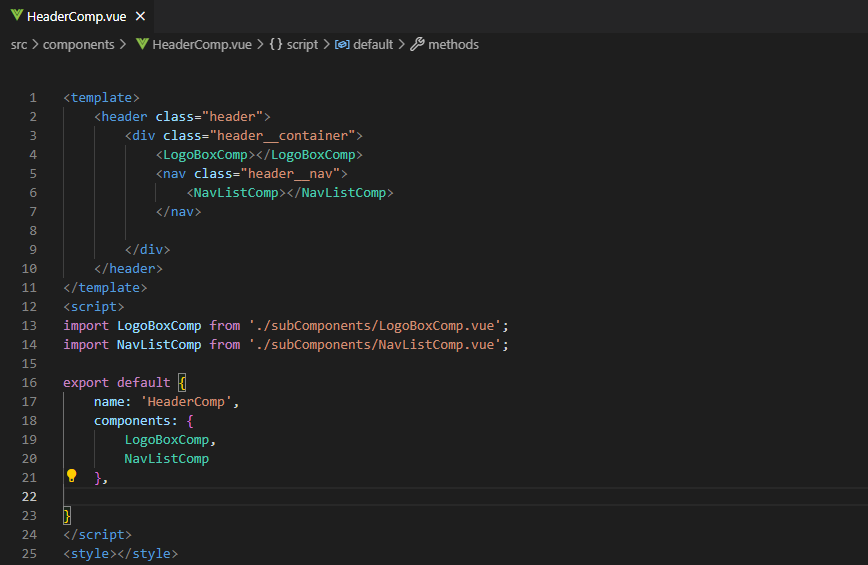


Теперь так же создаем компонент «FooterComp» и добавляем его в корневой компонент App.vue.



Таким же образом создадим вложенные компоненты «LogoBox» - блок с логотипом и «NavListComp» - блок навигации. Данные компоненты будем импортировать и подключать в компонент «HeaderComp».



Аналогичным образом создаем все остальные компоненты веб-приложения.

## 2.6 Настройка и работа с хранилищем Vuex

В своем проекте я будут подгружать данные для страниц из базы CMS (Content Management System) по API[[7]](#footnote-7). Этот метод управления контентом известен как headless[[8]](#footnote-8) CMS.

Headless CMS представляет собой CMS без встроенного пользовательского интерфейса для отображения контента. Основная идея заключается в разделении контента (данных) от его представления (дизайна). Это позволяет создавать содержимое один раз и затем использовать его на различных платформах и устройствах, не привязываясь к конкретному интерфейсу.

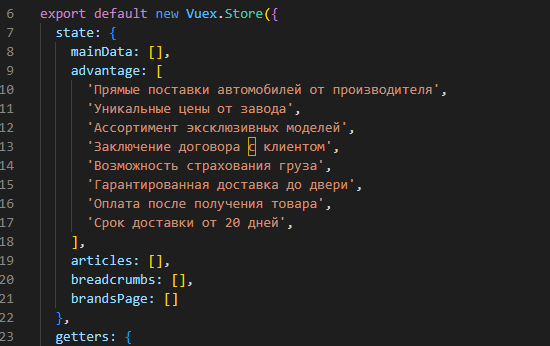
Для того чтобы сделать запрос на сервер, загрузить данные и получить их из хранилища необходимо создать директивы:

1. State
2. Mutations
3. Actions
4. Getters

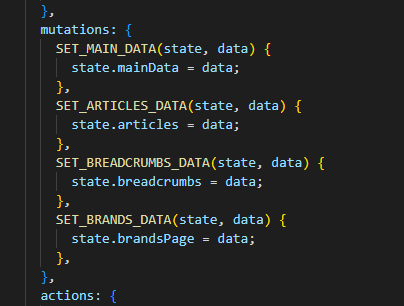
Для моего проекта потребуется создать несколько запросов к API:

1. Запрос на получение контента главной страницы (баннер, отзывы)
2. Запрос на получение данных по маркам/моделям
3. Запрос на получение данных по статьям
4. Запрос для формирования «хлебных крошек»

Создадим в директиве State пустые массивы, в которых в дальнейшем будут подгружаться данные.



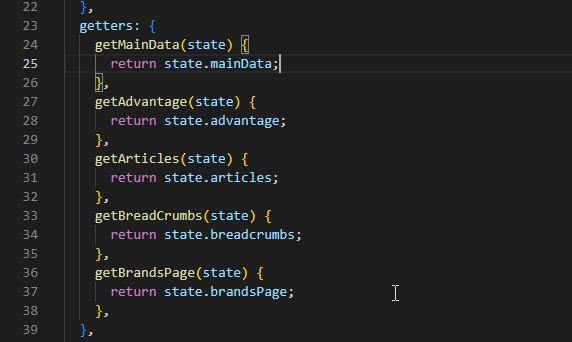
Для того, чтобы внести изменения в состояние (state) хранилища создадим соответствующие мутации (mutations):



В директиве «actions» создадим методы для осуществления запросов к серверу:



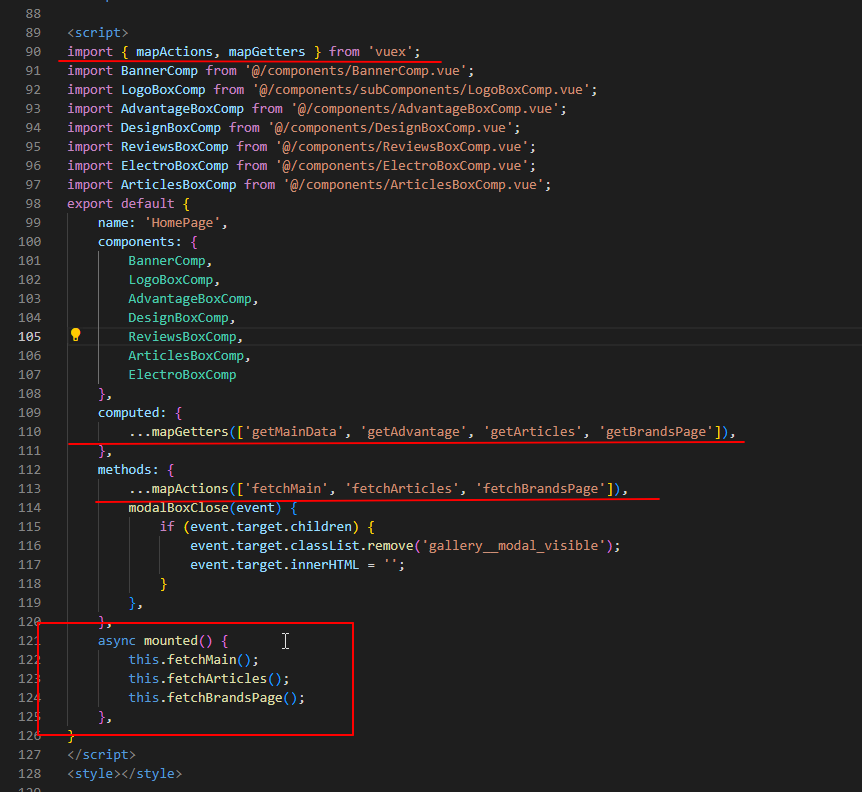
Чтобы в дальнейшем использовать полученные данные реализуем getters, которые в дальнейшем мы будем использовать в компонентах:



В главном компоненте Главной страницы будут отображаться отзывы, фотографии автомобилей и превью статей. Для их отображения необходимо будет осуществить 3 запроса к серверу:

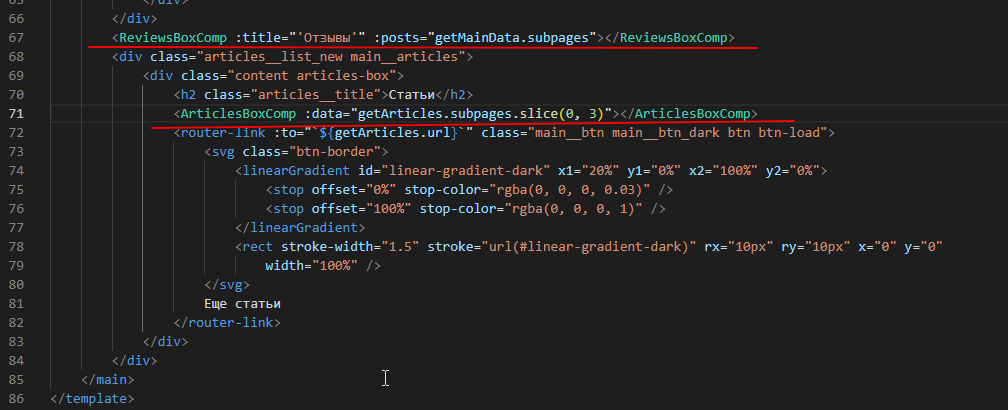
1. <https://api.site-77.ru/>
2. <https://api.site-77.ru/brands>
3. <https://api.site-77.ru/articles>

Импортируем необходимые директивы хранилища в компонент Главной страницы, объявляем actions в методах и вызываем их в асинхронном хуке «mounted», объявляем getters в свойстве computed:



Вызываем нужные геттеры внутри шаблона компонента.

Затем передаем эти данные внутрь компонентов через props. Это позволит компонентам быть более универсальными, и в случае необходимости мы сможем использовать их для отображения другой информации.



## 2.7 Использование встроенных директив Vue.js

Для создания компонентов и работы с ними используем встроенные директивы Vue.js. Они значительно ускоряют разработку и сокращают код.

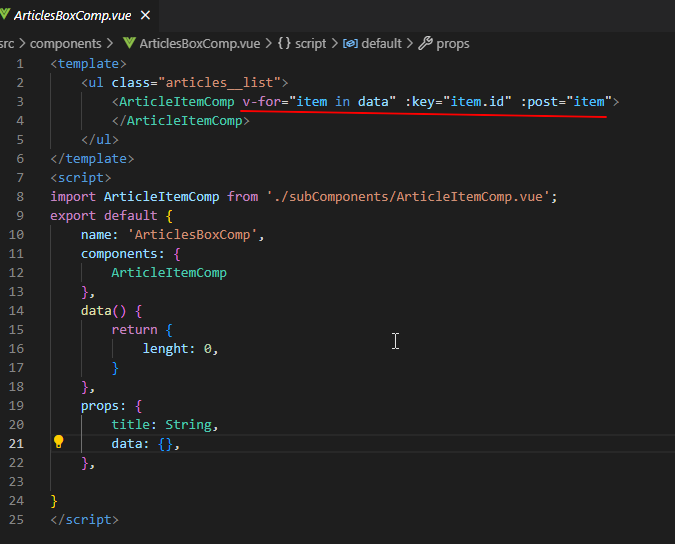
Рассмотрим варианты их использования на примере данного проекта.

1. **v-for**

В проекте множество повторяющихся элементов, которые можно создать с использованием цикла:

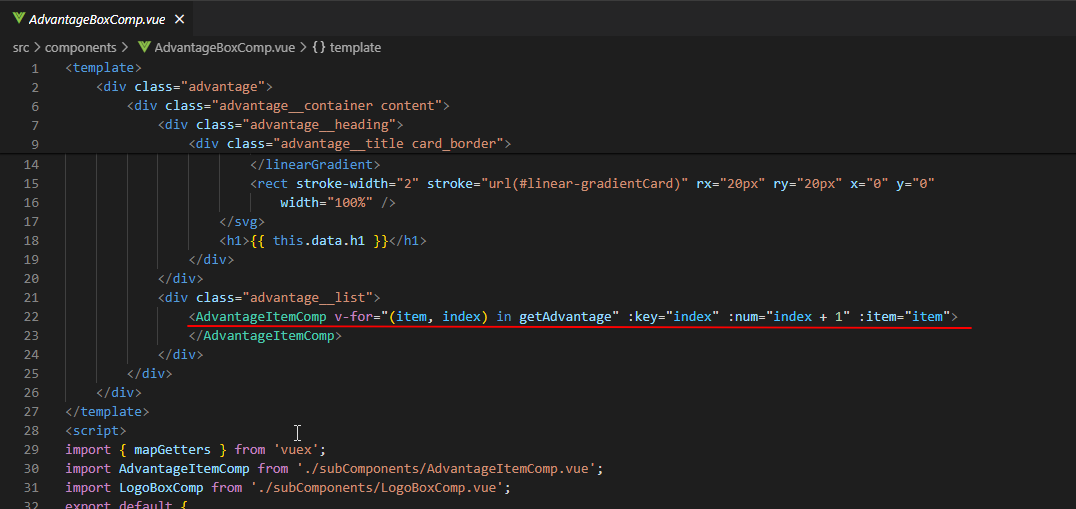
* На главной странице
  + Перечень преимуществ
  + Табы
  + Отзывы
  + Превью статей
* Хлебные крошки
* Страница Каталога
  + Перечень марок/моделей
* Страница модели
  + Перечень технических характеристик
  + Фотографии галереи

Создадим компонент с превью статей с использованием данной директивы:



С помощью директивы мы создаем компонент «ArticleItemComp» в цикле, идя по полученному из родительского компонента объекту «data» и передавая внутрь компонента отдельной статьи данные по статье «:post=”item”».

Если нам необходимо при создании списка элементов получить их индексы мы можем воспользоваться данной конструкцией:



В данном случае я передаю через props значение «:num» с использованием индекса элементов для отображения нумерации:



1. **v-on**

Данная директива заменяет собой стандартные события в js.

Осуществим вызов мобильного меню при клике на иконку «бургер» использовав событие «click».



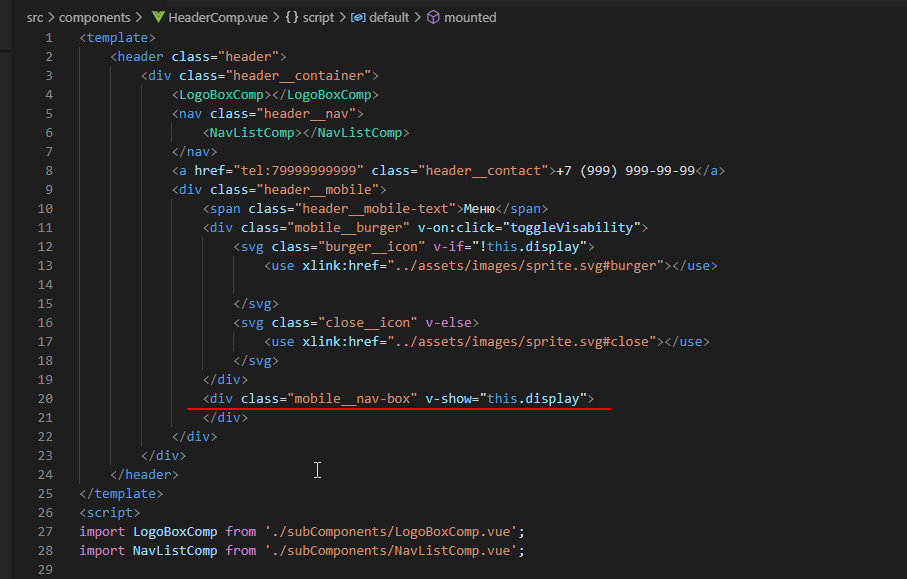
При клике по иконке вызывается метод «toggleVisability», который изменяет состояние отображения:



1. **v-show**

Мы может отобразить или скрыть элемент при каком-то условии. Как раз для отображения мобильного меню нам потребуется данная директива.

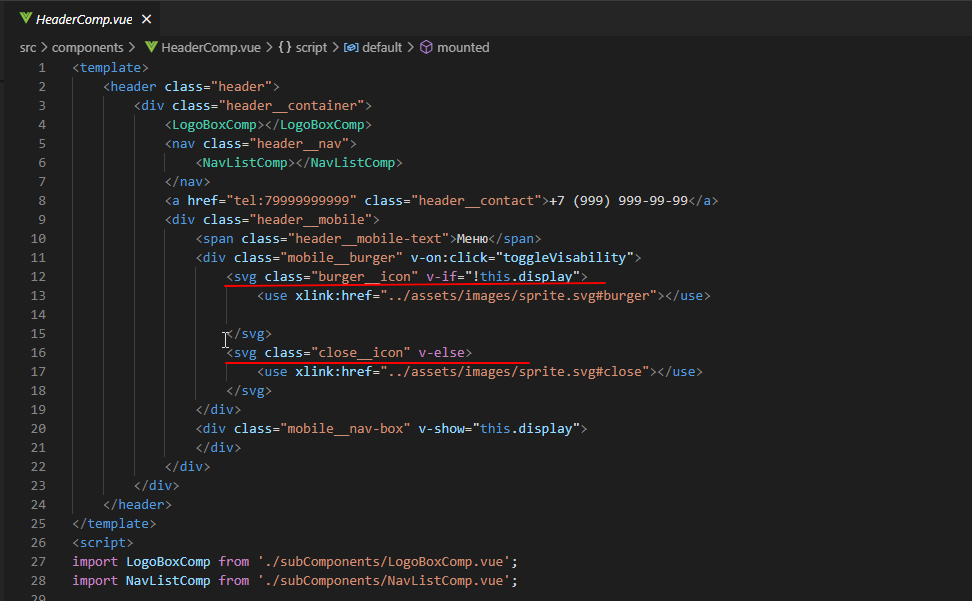
После вызова метода «toggleVisability» значение переменной «display» изменилось на противоположное и если значение «display» равняется true, то с помощью директивы v-show мы его отобразим:



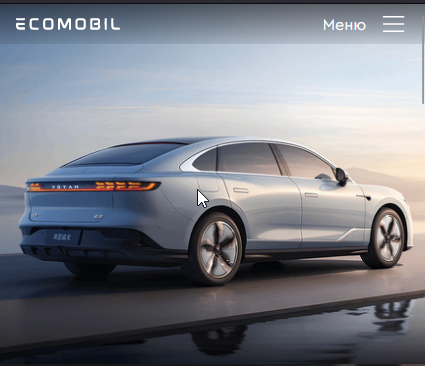
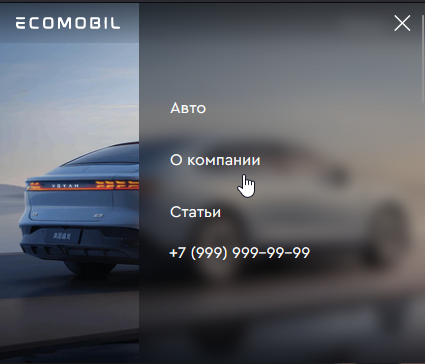
1. **v-if / v-else**

Мы можем отобразить различные элементы в зависимости от условий.

К примеру, в мобильном меню отображается иконка «бургер», но при клике на нее я хочу, чтобы данная иконка заменялась на «крестик».



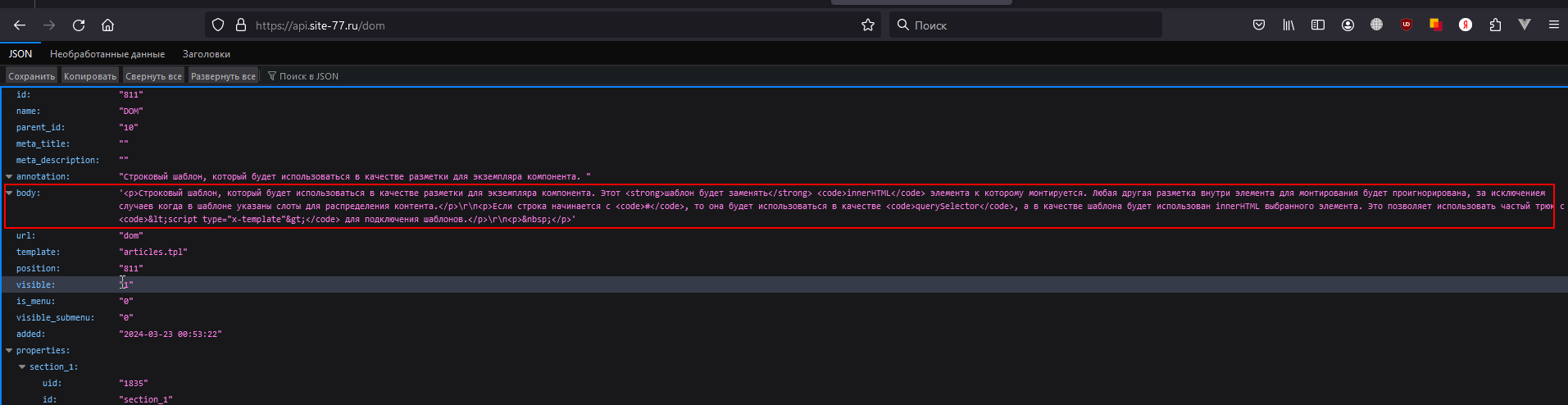
Результат:

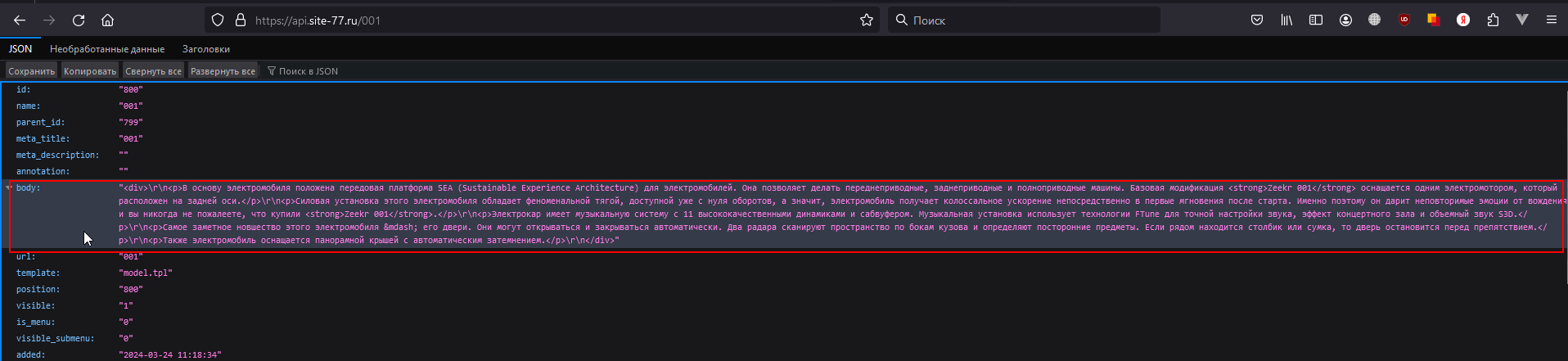
1. **v-html**

Так как контентную часть сайта я получаю по API из CMS, то иногда данные приходят в виде html кода (параграфы, заголовки и т.д).

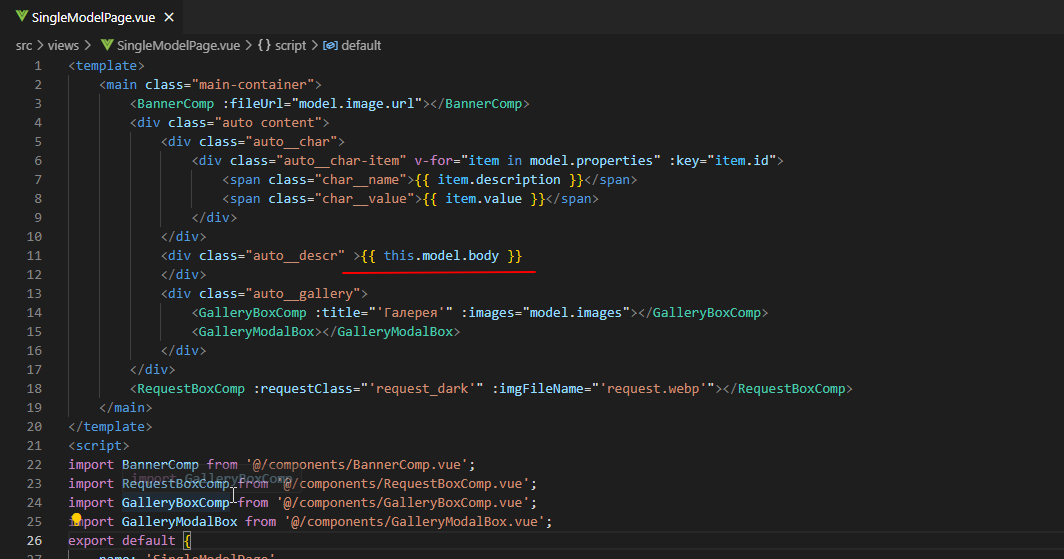
Ответ по старице статьи:



Ответ по старице модели:



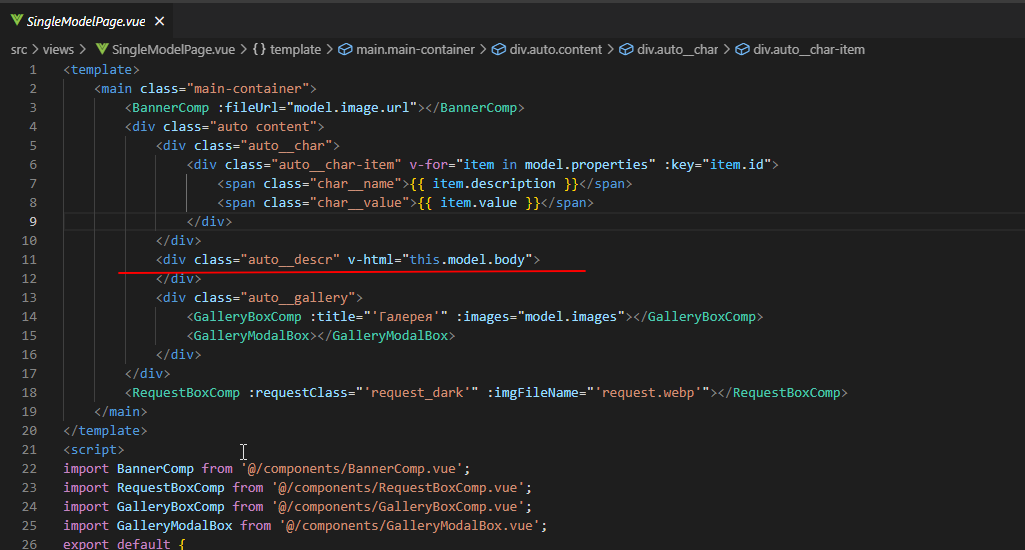
Выведем полученные данные на странице как обычно:



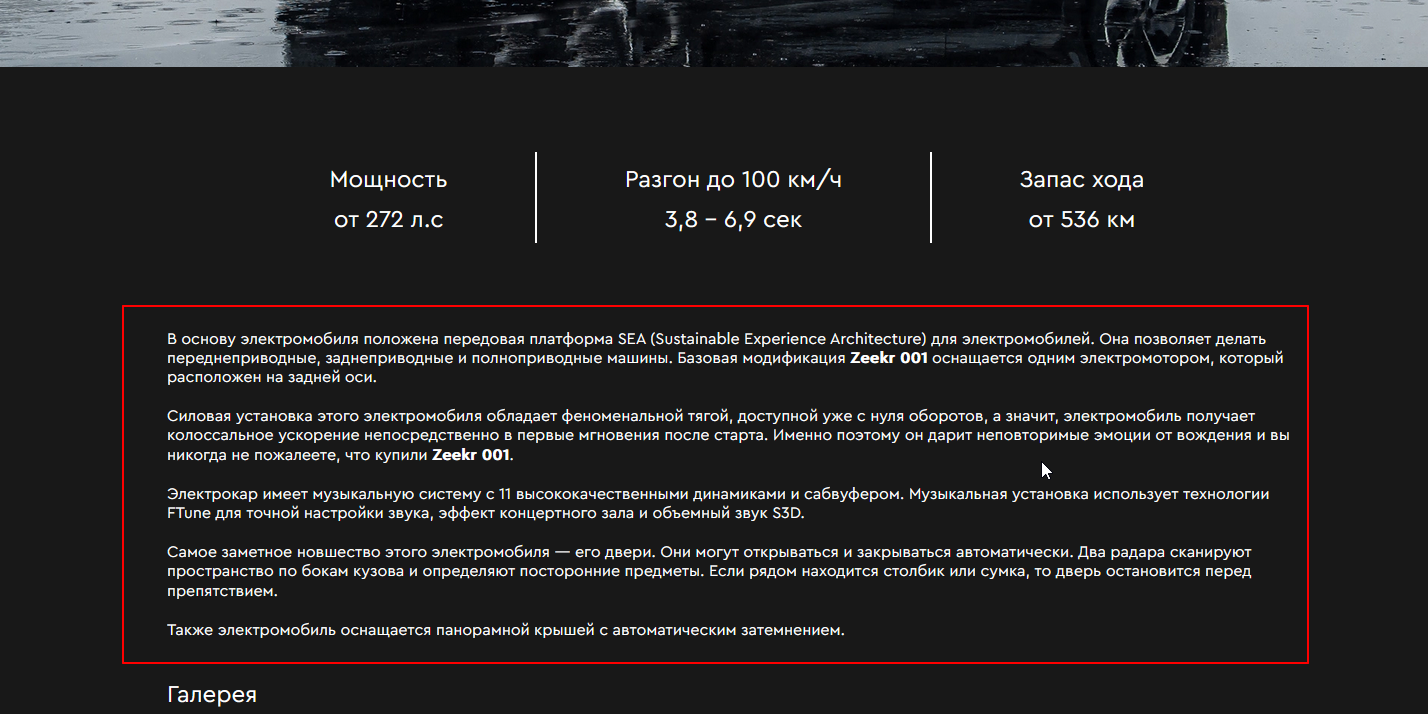
В итоге получим следующее:



Для корректного отображения данных используем директиву v-html:



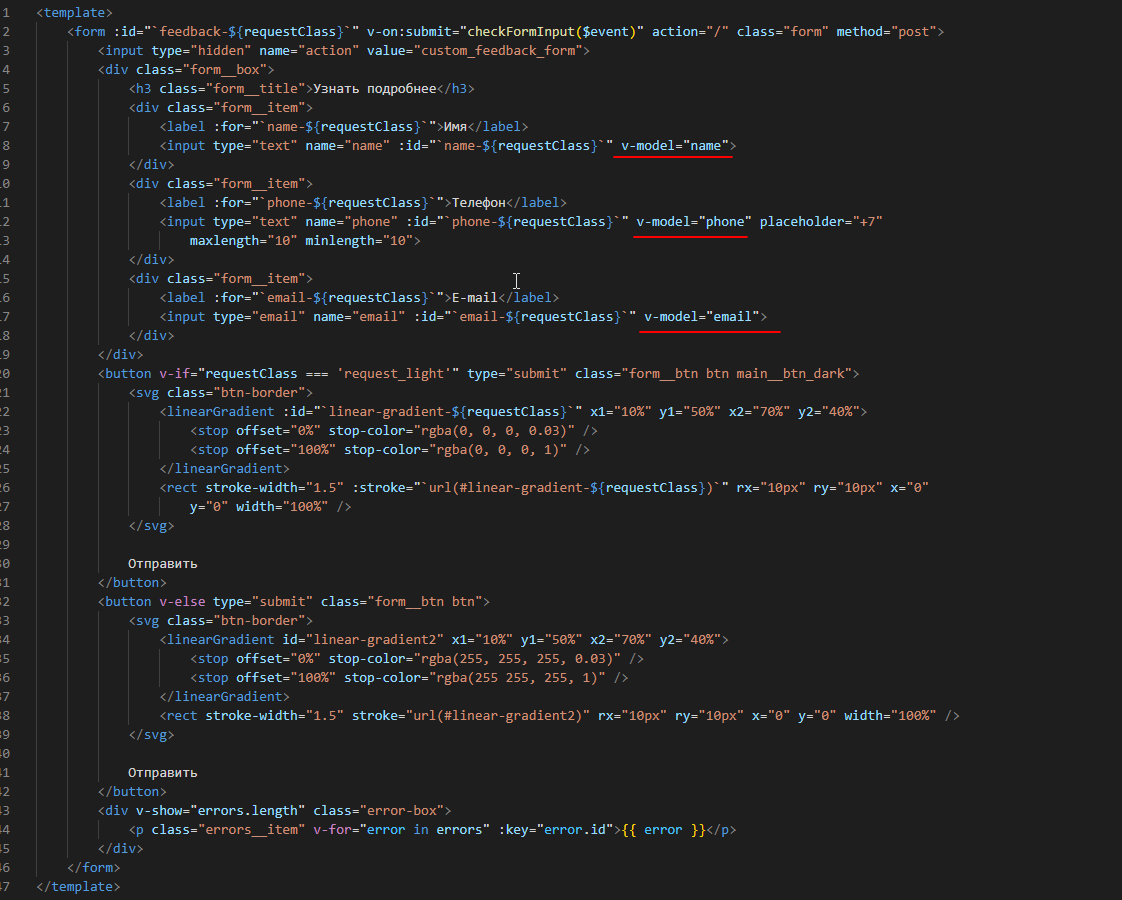
Итоговый результат:



1. **v-model**

Данная директива полезна при работе с формами.

Настроим простую проверку введенных в форму данных. Припишем имена переменных, в которые будут передаваться значения полей с помощью v-model и создадим метод для проверки наличия данных в полях.



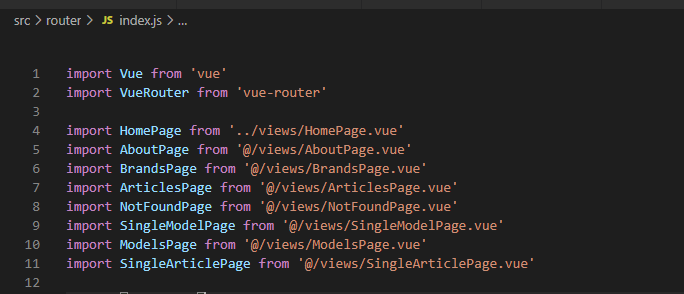


Таким образом, при попытке отправки формы по клику на кнопку «Отправить» будет вызываться функция «checkFormInput», которая будет проверять полученные значения.

## 2.8 Настройка маршрутизации в проекте

В предыдущем разделе мы создали основные компоненты страниц. Теперь необходимо настроить переходы на данные страницы в соответствии с их url.

Для того, чтобы настроить маршруты переходим в ранее созданный файл index.js в папке «Router». Импортируем компоненты страниц, для которых будем настраивать переходы.



В каждом маршруте пропишем путь, имя маршрута и вызываемый компонент при переходе по данному маршруту.

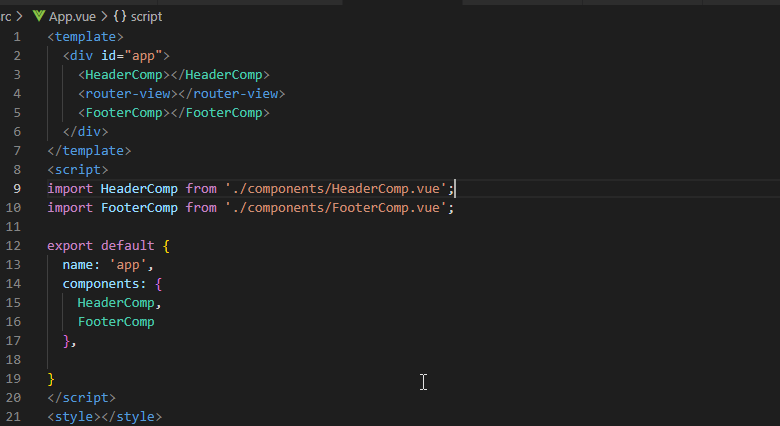
Для того, чтобы пользователь попадал на страницу «404» при переходе по несуществующему пути (а не только по пути /404) необходимо осуществить redirect с этих страниц на страницу с ошибкой. Для этого в свойстве «path» нужно указать «\*».

После внесенных изменений файл index.js будет выглядеть так:



После определения основных маршрутов, нам необходимо указать место, где будет выводится компонент соответствующий маршруту. Для этого прописываем тег «Router-view».

Так как в моем проекте статичными будут только шапка сайта и подвал, то размещу данный тег в основном компоненте App.vue таким образом:



В моем проекте предполагается наличие вложенных маршрутов, например:

1. Основная страница Каталога «Авто» с перечислением марок -> Страница с описанием Марки и перечислением возможных моделей -> Страница конкретной модели данной марки;
2. Страница списка статей «Статьи» -> Страница статьи

В Vue Router есть возможность описания подобных маршрутов. Данные маршруты можно описать двумя способами:

1. Как базовые маршруты
2. С использованием опции «children»

Отличие между ними в том, что при использовании свойства «children» необходимо указывать «Router-view» в родительском компоненте. Это нужно для того, чтобы не перезагружать страницу целиком, а только отдельные части родительского компонента.

Так как в моем случае изображения баннеров, крошки и заголовки на баннерах будут отличаться, данный способ мне не подходит. Описывать данные маршруты буду как обычно.

Так как url у каждой марки, каждой модели и статьи должен быть уникальным потребуется так же описание маршрутов как «динамических».

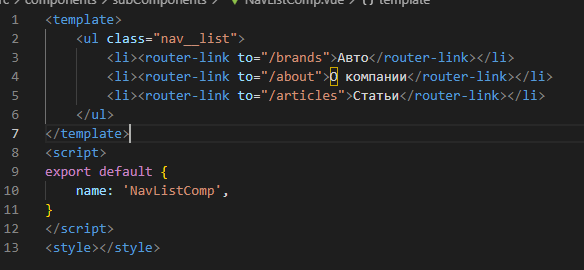
Для создания вложенных маршрутов необходимо создать маршрут на одном уровне с родительским и указать в свойстве «path» полный путь. Так как маршрут (к примеру, у марки) будет динамически изменятся, то в пути перед динамическим сегментом нужно указать «:»:



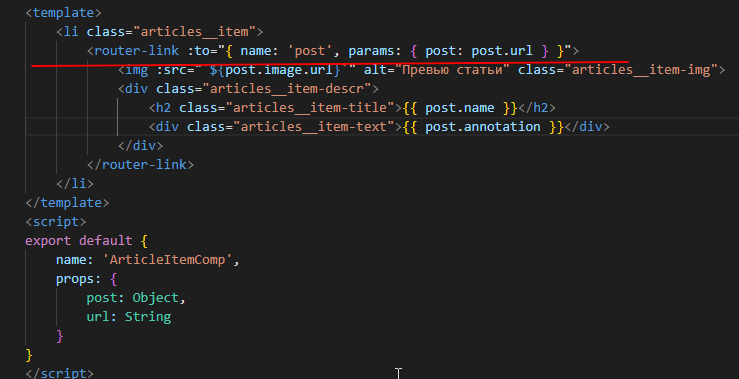
## 2.9 Настройка ссылок в проекте. Навигация

Для того чтобы переходы по маршрутам не сопровождались полной перезагрузкой страницы, необходимо заменить стандартный тег ссылки «a» на тег предоставляемый Vue Router – «Router-link», так же необходимо заменить атрибут «href» на «to».

Компонент навигации после внесенных изменений выглядит:

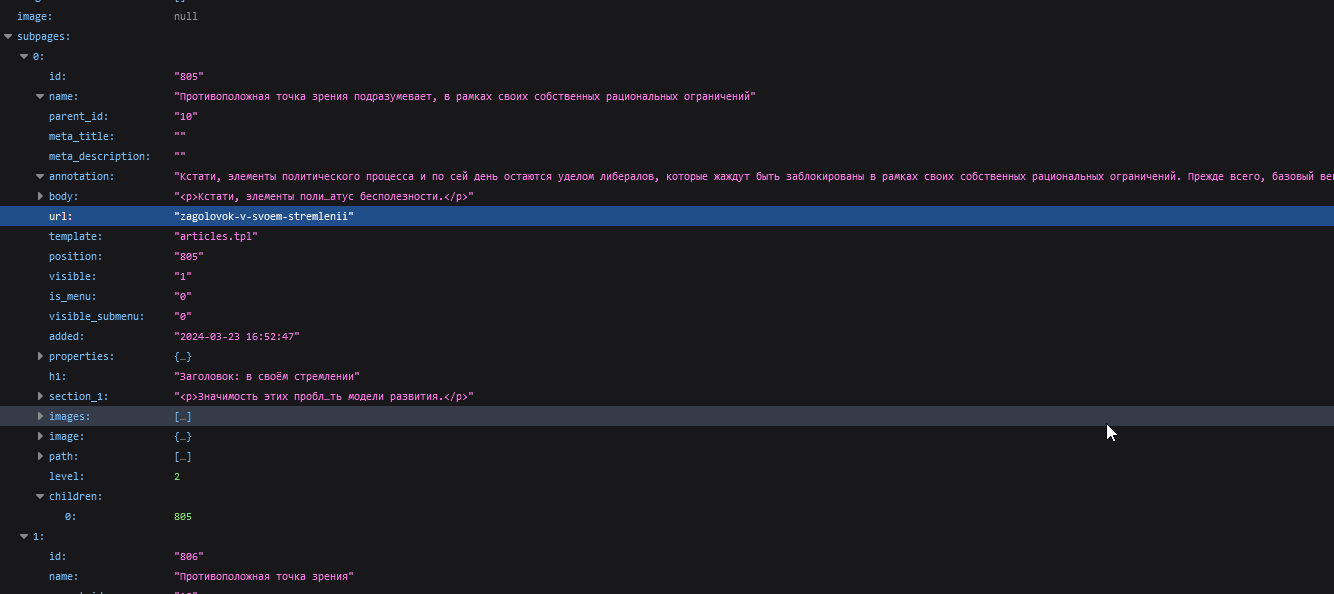


При указании путей для динамических маршрутов в атрибуте «to» нужно указать имя маршрута и значение динамического параметра маршрута. Последний параметр я получаю из объекта с данными по имеющимся маркам, моделям и статьям. Компонент превью статьи будет выглядеть следующим образом:

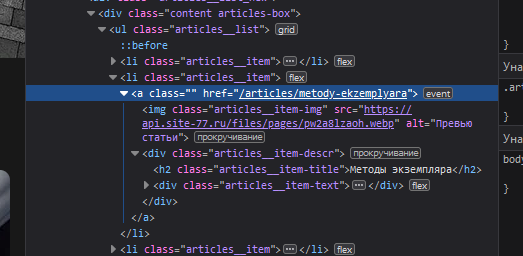


Внутрь компонента мы через props передаем объект с данными по конкретной статье. Указываем имя маршрута и динамический параметр, который мы прописывали ранее в файле index.js в папке Router.

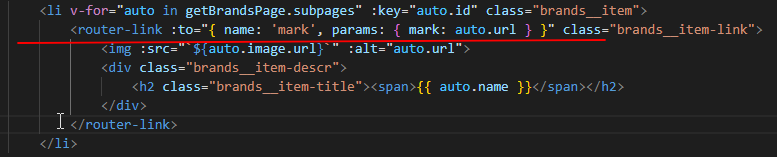
Пример значения параметра, который я получаю из базы по API:

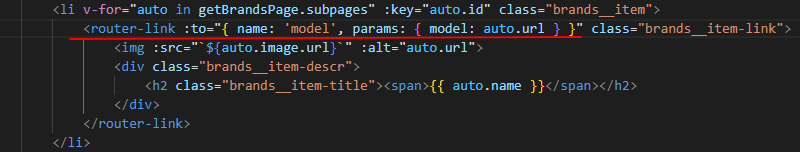


Таким образом мы получаем полный путь до статьи:



Аналогичным образом проделываем с ссылками на страницы Марок и Моделей.





# Заключение

В процессе этого исследования мы рассмотрели основы разработки MVP сайта-каталога с использованием Vue.js.

Теоретическое изучение материала позволило нам оценить преимущества и особенности использования Vue.js в разработке веб-приложений.

Пользовательский интерфейс, разработанный при помощи данного фреймворка, демонстрирует высокую производительность и удобство в использовании.

Практическая часть исследования проиллюстрировала особенности реализации и применения различных компонентов Vue.js при создании сайта-каталога.

В результате выполнения дипломной работы были получены важные теоретические знания и практические навыки, касающиеся работы с Vue.js, что обеспечило прочную базу для дальнейших исследований. Результаты исследования свидетельствуют о высокой эффективности применения Vue.js в разработке сайтов-каталогов.

Практическое применение этого исследования имеет большую значимость.

Знания и навыки, приобретенные в ходе работы, можно использовать для разработки веб-приложений на Vue.js, улучшения существующих проектов или других исследовательских задач.

В планах на дальнейшие исследования – расширение функционала разработанного сайта-каталога, добавление отсутствующих элементов, а также изучение возможностей интеграции Vue.js с другими технологиями.

Основная задача исследования – разработка MVP сайта-каталога на Vue.js – была выполнена успешно. Как показала практика, Vue.js обладает всеми необходимыми инструментами для создания современных и удобных веб-приложений.

Что касается проекта, который был рассмотрен в данной работе, он имеет определенный потенциал к совершенствованию. В дальнейшем можно предложить расширение функционал, создание более продвинутого пользовательского интерфейса, а также интеграция с другими технологиями и сервисами.

Выводы данного исследования позволяют нам утверждать, что Vue.js является эффективным инструментом для создания современных, производительных и интерактивных web-приложений.

# Список использемой литературы

Интернет-сайты:

<https://angular.io/docs>

<https://ru.react.js.org/docs/getting-started.html>

<https://nodejs.org/>

<https://ru.vuejs.org/v2/guide/>

<https://vuex.vuejs.org/>

[https://Router.vuejs.org/](https://router.vuejs.org/)

<https://cli.vuejs.org/ru/>

<https://api.site-77.ru/>

https://habr.com/ru/articles/518160/

# Приложения

1. https://github.com/Stassik/GraduationProject [↑](#footnote-ref-1)
2. https://angular.io/docs [↑](#footnote-ref-2)
3. https://ru.react.js.org/docs/getting-started.html [↑](#footnote-ref-3)
4. https://nodejs.org/ [↑](#footnote-ref-4)
5. https://cli.vuejs.org/ru/ [↑](#footnote-ref-5)
6. https://www.figma.com/file/IGjuti65kHoQ30eOZ9TWRJ/ECOMOBIL-(Copy)-(2)-(Copy)?type=design&node-id=48-5&mode=design&t=S3tuj0CXDr4fHUFC-0 [↑](#footnote-ref-6)
7. https://api.site-77.ru/ [↑](#footnote-ref-7)
8. https://habr.com/ru/articles/518160/ [↑](#footnote-ref-8)