Geekbrains

Специальность: Frontend-программист “Цифровые профессии”

Дипломным проект: “Создание веб-приложения из макета Figma для подбора замков по марке автомобиля с использованием Vue.”

Юрьев Алексей Владимирович

Вологда, 2024год.

**Создание веб-приложения из макета Figma для подбора замков по марке автомобиля с использованием Vue**

Оглавление

[**Введение:** 4](#_Toc158241018)

[**Цель дипломного проекта:** 4](#_Toc158241019)

[**Задачи дипломного проекта:** 4](#_Toc158241020)

[**Методы и инструменты:** 5](#_Toc158241021)

[**Ожидаемые результаты:** 5](#_Toc158241022)

[**Веб-приложения получения данных из Интернета.** 6](#_Toc158241023)

[Основы веб-приложения включают: 6](#_Toc158241024)

[**Frontend веб-приложения, его особенности.** 7](#_Toc158241025)

[**Программные продукты и инструменты web-разработки.** 9](#_Toc158241026)

[**Наиболее популярные примеры IDE приведены далее.** 10](#_Toc158241027)

[**Дипломный проект будет разрабатываться в VS Code.** 11](#_Toc158241028)

[Некоторые из ключевых особенностей VS Code включают в себя: 11](#_Toc158241029)

[**Инструменты frontend.** 13](#_Toc158241030)

[**HTML** 13](#_Toc158241031)

[**CSS** 13](#_Toc158241032)

[**JavaScript и Vue.js** 14](#_Toc158241033)

[**Scss** 15](#_Toc158241034)

[**Node.js** 16](#_Toc158241035)

[EmailJS 17](#_Toc158241036)

[**Практическая разработка и тестирование результатов работы веб-приложения.** 18](#_Toc158241037)

[**Установка Node.js** 18](#_Toc158241038)

[**Установка Vue.CLI при Node.js и создание проекта** 18](#_Toc158241039)

[**Figma** 19](#_Toc158241040)

[**Коренвой компонент** 22](#_Toc158241041)

[Дочерние компоненты: 23](#_Toc158241042)

[Main.js - в исходном коде клиентского дополнения 23](#_Toc158241043)

[index.html – Основа сайта. Добавить стиль для векторным значок. 23](#_Toc158241044)

[Store/index.js - Хранилище» — это контейнер, в котором хранится состояние вашего приложения. 24](#_Toc158241045)

[Создал компоненты Header.vue. – шапка 24](#_Toc158241046)

[Создал компоненты main.vue – основа 28](#_Toc158241047)

[Создание сервер и шаблоны emailJS 33](#_Toc158241048)

[Создал компоненты requisition.vue. оставить заявку 34](#_Toc158241049)

[Создал компоненты Protection.vue – о защите 42](#_Toc158241050)

[Создал компоненты Comfort.vue – о комфорте 46](#_Toc158241051)

[**Создал компоненты advantage.vue – о преимуществе** 48](#_Toc158241052)

[Создал компоненты footer.vue - нижний 51](#_Toc158241053)

[Подведение итогов, оценка результатов и заключение 55](#_Toc158241054)

[Список использованный литературы 56](#_Toc158241055)

**Введение:**

Веб-разработка становится всё более актуальной сферой деятельности с каждым годом, и требования к качеству веб-приложений постоянно растут. Современная автомобильная промышленность насчитывает множество моделей и марок автомобилей, каждая из которых имеет свою специфику и уникальные замки. В связи с этим возникает необходимость в разработке эффективного веб-приложения, которое позволяет пользователям легко и быстро подобрать необходимые замки для их автомобилей. Для создания удобного и привлекательного пользовательского интерфейса макет Figma используется для создания интерфейса, а Vue.js - для разработки логики приложения.

## **Цель дипломного проекта:**

Целью данного дипломного проекта является разработка веб-приложения с использованием макета Figma для обеспечения удобного и интуитивно понятного интерфейса пользователей, а также реализация функционала подбора замков по марке автомобиля.

## **Задачи дипломного проекта:**

1. Изучение основных концепций и принципов разработки веб-приложений с использованием Vue.js.
2. Анализ требований и спецификаций для построения приложения для подбора замков по марке автомобиля.
3. Изучение макета Figma и его внедрение в процесс разработки пользовательского интерфейса.
4. Разработка логики и функционала приложения с использованием Vue.js.
5. Тестирование и отладка приложения для обеспечения его корректной работы и отсутствия ошибок.
6. Оформление документации и написание отчета о выполненном проекте.

## **Методы и инструменты:**

* Исследование и анализ существующих технологий и разработческих инструментов для разработки веб-приложений.
* Использование Vue.js для разработки логики и функционала приложения.
* Применение макета Figma для создания привлекательного и интуитивно понятного пользовательского интерфейса.
* Тестирование и отладка приложения с использованием различных инструментов и методик.
* Изучение и применение современных методов разработки веб-приложений.

## **Ожидаемые результаты:**

* Разработка веб-приложения, позволяющего пользователям легко и быстро подобрать замки по марке автомобиля.
* Создание привлекательного и удобного пользовательского интерфейса с использованием макета Figma.
* Реализация логики и функционала приложения с использованием Vue.js.
* Отладка и тестирование приложения для обеспечения его корректной работы.
* Составление документации и написание отчета о выполненном проекте.

# **Веб-приложения получения данных из Интернета.**

Веб-приложение - это программное обеспечение, которое работает через интернет и предоставляет пользователю доступ к различным функциям и сервисам.

## Основы веб-приложения включают:

1. Клиент-серверная архитектура: веб-приложение состоит из клиентской и серверной частей. Клиентская часть - это пользовательский интерфейс, который отображается веб-браузером. Серверная часть - это сервер, который обрабатывает запросы от клиента, выполняет необходимые действия и отправляет обратно результаты.
2. Языки программирования: для разработки веб-приложений используются различные языки программирования, такие как HTML, CSS и JavaScript для фронтенда, а также языки программирования, такие как Java, Python или PHP, для бэкенда.
3. HTTP протокол: веб-приложения взаимодействуют с клиентами через HTTP протокол. Клиенты отправляют HTTP запросы на сервер с помощью URL-адреса, а серверы посылают HTTP ответы с данными обратно клиентам.
4. Базы данных: веб-приложения могут использовать базы данных для хранения данных. Базы данных могут быть реляционными (например, MySQL, PostgreSQL), нереляционными (например, MongoDB, Cassandra) или гибридными.
5. Фреймворки и библиотеки: разработка веб-приложений облегчается использованием фреймворков и библиотек. Фреймворки, такие как Django (для Python) или Ruby on Rails (для Ruby), предоставляют готовые инструменты и архитектуру для разработки приложения. Библиотеки, такие как jQuery или React, предоставляют функции и возможности для работы с пользовательским интерфейсом.

Это основы веб-приложения, их понимание поможет начать разработку и понимать, как работает веб-приложение целиком. Дальнейшее изучение и практическая разработка помогут освоить эту тему более глубоко и узнать о более сложных аспектах разработки веб-приложений.

# **Frontend веб-приложения, его особенности.**

Frontend веб-приложения - это пользовательский интерфейс (UI), с которым взаимодействует пользователь. Он отвечает за представление данных, взаимодействие с пользователем и обработку событий. Особенности frontend веб-приложения могут включать:

1. HTML, CSS и JavaScript: основные технологии, используемые для разработки frontend. HTML отвечает за структуру и содержимое страницы, CSS отвечает за внешний вид и оформление, а JavaScript обеспечивает интерактивность и динамическое поведение страницы.
2. Кросс-браузерность: веб-приложение должно работать одинаково хорошо на разных браузерах, таких как Chrome, Firefox, Safari и других. Для этого разработчики должны учитывать различия в синтаксисе, поддержке функций и поведении браузеров.
3. Responsive Design: frontend приложение должно быть адаптивным и подстраиваться под различные типы устройств и размеры экранов. Это позволяет обеспечить удобство использования и доступность сайта независимо от устройства, на котором пользователь просматривает страницу.
4. Взаимодействие с API: frontend может взаимодействовать с сервером, отправлять запросы и получать ответы через API. Это позволяет приложению получать и обновлять данные в реальном времени.
5. Библиотеки и фреймворки: для упрощения разработки и повышения производительности, разработчики frontend часто используют библиотеки и фреймворки, такие как React, Angular или Vue.js. Они предоставляют готовые инструменты и архитектуру для разработки масштабируемых и модульных приложений.
6. Оптимизация производительности: разработчики frontend должны уделить внимание оптимизации производительности приложения. Это включает в себя уменьшение размера и количество загружаемых ресурсов, оптимизацию кода и использование кеширования для ускорения загрузки страницы.
7. UX/UI дизайн: frontend приложение должно обеспечивать хороший пользовательский опыт (UX) и иметь привлекательный пользовательский интерфейс (UI). Это включает в себя удобную навигацию, понятные инструкции, эффективное использование цветов, шрифтов и графики.

Это некоторые из основных особенностей frontend веб-приложения, которые разработчики должны учитывать при разработке пользовательского интерфейса для веб-приложения.

# **Программные продукты и инструменты web-разработки.**

Любая разработка начинается с выбора среды разработки. Интегрированные среды разработки (IDE) — это программные приложения, которые обеспечивают комплексную среду для разработки, тестирования и развертывания программного обеспечения. IDE обычно включают в себя редактор кода, компилятор или интерпретатор и инструменты отладки, а также другие функции, такие как интеграция контроля версий и инструменты управления проектами.

Разработчики часто предпочитают IDE, потому что они обеспечивают универсальное решение для всех аспектов процесса разработки. Разработчики могут писать и редактировать код, компилировать и запускать код, а также устранять ошибки в одной и той же среде. IDE также часто предоставляют дополнительные функции, такие как завершение кода, подсветка синтаксиса и инструменты отладки, которые могут ускорить и повысить эффективность разработки.

Однако некоторые разработчики предпочитают использовать текстовые редакторы и инструменты командной строки, поскольку они обеспечивают большую гибкость и могут быть настроены в соответствии с конкретными потребностями разработки. В конечном счете, выбор между использованием IDE или текстового редактора зависит от личных предпочтений и требований проекта разработки.

## **Наиболее популярные примеры IDE приведены далее.**

1. Visual Studio Code - это бесплатный редактор исходного кода, который поддерживает множество языков программирования и имеет множество расширений для различных инструментов и платформ.
2. Chrome DevTools - это инструменты разработчика для браузера Chrome. Они включают в себя инструменты для отладки, профилирования, проверки доступности и оптимизации производительности веб-приложений.
3. Node Package Manager (NPM) - это менеджер пакетов для Node.js, который позволяет разработчикам легко устанавливать, управлять и обновлять пакеты Node.js.
4. Vue.js - это прогрессивный фреймворк для создания пользовательских интерфейсов. Он предлагает комбинацию из V-модели (двунаправленное связывание данных), рендеринг на основе шаблонов, декларативный и компонентно-ориентированный подход.

# **Дипломный проект будет разрабатываться в VS Code.**

Одной из главных особенностей VS Code является обширный набор расширений и плагинов, которые позволяют разработчикам настраивать редактор под свои конкретные нужды. VS Code поддерживает широкий спектр языков программирования, и доступны расширения для всего: от подсветки синтаксиса и форматирования кода до отладки и интеграции с Git.

## Некоторые из ключевых особенностей VS Code включают в себя:

1. IntelliSense: эта функция предоставляет интеллектуальные предложения по завершению кода по мере ввода в зависимости от языка, с которым вы работаете.
2. Встроенная поддержка Git: VS Code включает встроенную поддержку Git, что позволяет легко управлять репозиториями кода, не выходя из редактора.
3. Поддержка отладки: VS Code предоставляет встроенный отладчик для ряда языков программирования, позволяющий легко выполнять код, выявлять и исправлять ошибки.
4. Запуск задач: VS Code поддерживает создание пользовательских задач, которые могут автоматизировать общие рабочие процессы, такие как создание и тестирование кода.
5. Встроенный терминал: VS Code включает встроенное окно терминала, позволяющее выполнять команды оболочки непосредственно в редакторе.
6. Настройка темы: VS Code включает в себя ряд встроенных цветовых тем, и вы также можете загружать и устанавливать пользовательские темы, чтобы настроить внешний вид редактора.
7. Поддержка нескольких языков: VS Code поддерживает широкий спектр языков программирования, включая JavaScript, Python, Ruby и многие другие.

VS Code — очень гибкий и настраиваемый редактор кода, предоставляющий ряд мощных функций для разработчиков. Он имеет большое и активное сообщество пользователей и разработчиков, а в Интернете доступно множество ресурсов, которые помогут начать работу и научиться эффективно его использовать.

# **Инструменты frontend.**

## **HTML**

HTML (HyperText Markup Language) – это стандартный язык разметки документов для Всемирной паутины. Документы HTML состоят из различных тегов, которые определяют структуру и содержимое веб-страницы. Теги могут быть пустыми или содержать текст или другие элементы, такие как изображения.

Существует несколько основных типов тегов в HTML:

* Тег разметки - определяет структуру документа и содержит информацию о его заголовке, описании и т.д.
* Теги форматирования - используются для изменения внешнего вида текста, например, изменение размера шрифта, цвета или стиля.
* Теги ссылок - используются для добавления ссылок на другие веб-страницы или ресурсы.
* Теги изображений - используются для вставки изображений на веб-страницу.

Кроме того, HTML поддерживает атрибуты, которые позволяют задавать дополнительные параметры для тегов. Например, можно задать размер изображения, ссылку на альтернативный текст и т.д.

В целом, HTML является простым и понятным языком разметки, который позволяет создавать веб-страницы с различным содержанием и функциональностью.

## **CSS**

CSS (Cascading Style Sheets) – это язык стилей, который используется для оформления и стилизации веб-страниц. С его помощью можно изменять цвета, шрифты, размеры элементов, добавлять тени, градиенты и многое другое. CSS позволяет создать единый стиль для всех страниц сайта и отделить оформление от содержимого, что упрощает поддержку и обновление дизайна.

Основные понятия CSS:

* Селекторы – определяют, к каким элементам на странице будут применяться стили.
* Свойства – задают параметры оформления элементов, такие как цвет, шрифт, размер и т. д.
* Значения – определяют конкретные параметры свойств, например, цвет текста или размер шрифта.

CSS поддерживает каскадирование, то есть стили могут наследоваться от родительского элемента к дочерним. Это позволяет создавать более сложные стили и гибко настраивать оформление сайта.

Один из популярных инструментов для работы с CSS – препроцессор Sass, который расширяет возможности CSS и позволяет использовать переменные, вложенные блоки и другие возможности.

## **JavaScript и Vue.js**

JavaScript (JS) — это язык программирования, который используется для создания интерактивных веб-сайтов и приложений. Он является одним из самых популярных языков программирования в мире и используется в большинстве веб-браузеров. Vue.js — это фреймворк для создания пользовательских интерфейсов на JavaScript, который позволяет создавать динамические и реактивные веб-приложения.

В этой лекции мы познакомимся с основами JavaScript и фреймворка Vue.js. Мы начнем с изучения основ JavaScript, таких как переменные, типы данных, циклы и условные операторы. Затем мы перейдем к изучению Vue.js, включая создание компонентов, использование директив и обработку событий. Мы также рассмотрим, как использовать Vuex для управления состоянием приложения и как интегрировать Vue с другими библиотеками, такими как React и Angular.

После этой лекции вы будете знакомы с основами программирования на JavaScript и сможете начать создавать свои собственные веб-приложения с использованием Vue.js.

## **Scss**

SCSS — это препроцессор для написания CSS кода. SCSS дает дополнительные возможности. С ним можно использовать переменные, вложенные селекторы и медиазапросы, собирать несколько файлов в один, миксины, наследования и другие полезные вещи которые упростят и структурируют работу с CSS кодом.

SCSS это отдельный формат файла, в котором пишутся стили для сайта. Он очень похож на CSS, но в добавок имеет множество улучшений. Буквально SCSS расшифровывается как Super CSS — то есть CSS c супер-силой.

SCSS пишется в отдельных файлах с расширением .scss, например main.scss.

SCSS не подключается в браузер напрямую. Браузеры не умеют с ним работать. SCSS компилируется (собирается) в привычный для браузера CSS код, и уже CSS файл подключается на страницу.

Для того чтобы скомпилировать SCSS в CSS нужно использовать компилятор. инструмент для сборки проекта: Webpack .

Особенности SCSS

* Это отдельный формат файла
* Пишется в отдельных файлах с расширением .scss
* SCSS не подключается в браузер напрямую
* SCSS надо компилировать в CSS

Дает возможность использовать переменные, вложенные селекторы и медиазапросы, собирать несколько файлов в один, миксины, наследования и многое другое чтобы структурировать работу с CSS кодом

## **Node.js**

Node.js — это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом, которая позволяет разработчикам запускать код JavaScript вне веб-браузера. Он построен на основе движка Google V8, который представляет собой высокопроизводительный движок JavaScript, на котором работает браузер Google Chrome.

Node.js позволяет разработчикам писать серверные приложения с использованием JavaScript, что упрощает процесс разработки. Он предоставляет ряд функций, которые делают его популярным выбором для создания масштабируемых высокопроизводительных веб-приложений, в том числе:

Неблокирующая модель ввода-вывода (Non-blocking I/O model): Node.js использует управляемую событиями неблокирующую модель ввода-вывода, которая позволяет ему обрабатывать большое количество подключений с небольшим количеством потоков.

Высокая производительность: Node.js построен на основе движка V8, который обеспечивает высокую производительность кода JavaScript.

Большая экосистема пакетов: Node.js имеет большую и растущую экосистему пакетов с тысячами модулей и пакетов, доступных через диспетчер пакетов Node (npm).

Кроссплатформенная совместимость: Node.js является кроссплатформенным, что означает, что он может работать в Windows, macOS и Linux.

Поддержка приложений реального времени: Node.js хорошо подходит для создания приложений реального времени, таких как приложения для чата и онлайн-игровые платформы, поскольку он обеспечивает связь в реальном времени между сервером и клиентом.

Node.js обычно используется в сочетании с такими веб-фреймворками, как Express.js и Sails.js, для создания серверных приложений. Он также широко используется в сочетании с интерфейсными фреймворками, такими как Vue, React и Angular, для создания полнофункциональных веб-приложений.

## EmailJS

EmailJS — это библиотека JavaScript, которая помогает отправлять письма с использованием только клиентских технологий.

Для работы с EmailJS не требуется сервер. Нужно только подключить его к одному из поддерживаемых сервисов электронной почты, создать шаблон письма и использовать EmailJS для отправки письма.

Среди других возможностей EmailJS:

* добавление вложений;
* проверка CAPTCHA;
* переключение между сервисами электронной почты без изменения кода;
* просмотр истории запроса письма.

# **Практическая разработка и тестирование результатов работы веб-приложения.**

## **Установка Node.js**

Node.js позволяет разработчикам использовать JavaScript для создания серверных приложений, взаимодействия с базами данных, разработки API и других задач. Node.js скачивается и устанавливается по ссылке ниже: <https://nodejs.org/en/download/>

Node.js так же включает в себя npm (Node Package Manager) - менеджер пакетов для языка программирования JavaScript. Он позволяет легко управлять зависимостями и библиотеками, используемыми в проекте. С помощью npm можно устанавливать, обновлять и удалять пакеты JavaScript из центрального репозитория npm или из локального каталога. Каждый пакет в npm содержит файл package.json, который определяет его зависимости, версии и другую информацию.

В директории созданного проекта необходимо проинициализировать npm командой:

|  |
| --- |
| npm init |

## **Установка Vue.CLI при Node.js и создание проекта**

Выбрал Vue.js за его компонентный подход к созданию пользовательского интерфейса, использование директив и реактивных свойств для управления динамическим контентом, а также удобной интеграции с data JSON.

|  |
| --- |
| npm install -g @vue/cli |

Данный тип установки позволяет использовать компонентный подход в разработке веб приложения.

После установки создается папка в которой преднастроены все необходимые модули.

Запуск разработки производится командой:

|  |
| --- |
| npm run serve |

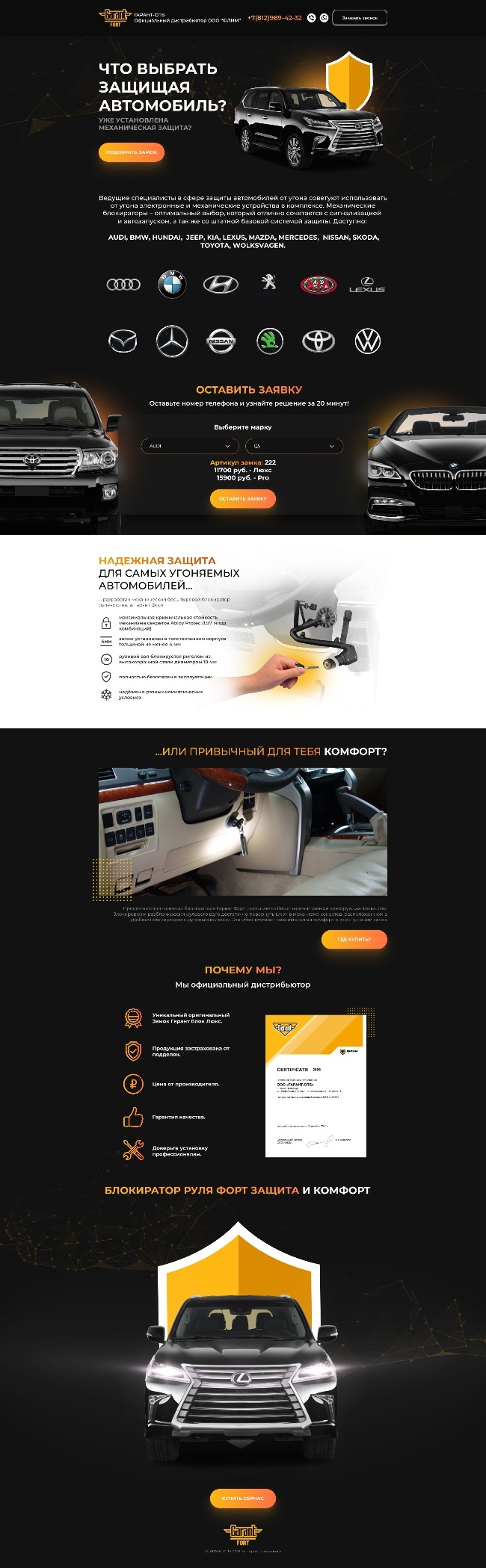
## **Figma**

Процесс создания проекта в Figma можно разделить на несколько этапов:

* Изучение макета: Откройте макет в Figma. Изучите все элементы и компоненты, используемые в макете. Определите, какие элементы являются статическими (например, фоновые изображения), а какие - динамическими (например, текстовые поля, кнопки).
* Импортирование ресурсов: Загрузите все необходимые ресурсы для вашего проекта, такие как изображения, иконки, шрифты и прочее.
* Создание прототипов: Создайте прототипы всех страниц и компонентов вашего проекта. Это поможет вам понять, как все элементы будут взаимодействовать друг с другом и как будет выглядеть конечный продукт.
* Разработка дизайна: Начните создавать дизайн вашего проекта, используя элементы и компоненты из макета. Убедитесь, что вы сохраняете общий стиль и цветовую схему макета.

**Макета на ссылку:** [**https://www.figma.com/file/c8tU9MfTZppHWuMi7RNWJj/Гарант-ФОРТ?type=design&node-id=202-2403&mode=design&t=yjOEPj4E6sANbD31-0**](https://www.figma.com/file/c8tU9MfTZppHWuMi7RNWJj/Гарант-ФОРТ?type=design&node-id=202-2403&mode=design&t=yjOEPj4E6sANbD31-0)

**Десктопная версия**

****

И еще должен быть адаптированные на планшетные и мобильные версиях

Планшетная версия мобильная версия

Разбивает раздела на 7 частей( Header, Main, Requisiton, Protection, Comfort, advantage, footer)

## **Коренвой компонент**

App.vue — корневой компонент приложения Vue.js. Это начальная точка приложения, где визуализируются все остальные компоненты. Это однофайловый компонент, который служит контейнером для других компонентов и определяет базовую структуру приложения.

<template>

  <HeaderBlock />

  <MainBlock />

  <Requisition />

  <Protection />

  <Comfort />

  <Advantage />

  <Footer />

</template>

<script>

import HeaderBlock from './components/header.vue'

import MainBlock from './components/main.vue'

import Requisition from './components/requisition.vue';

import Protection from './components/protection.vue';

import Comfort from './components/comfort.vue';

import Advantage from './components/advantage.vue';

import Footer from './components/footer.vue';

export default {

  name: 'App',

  components: {

    HeaderBlock,

    MainBlock,

    Requisition,

    Protection,

    Comfort,

    Advantage,

    Footer,

  }

}

</script>

## Дочерние компоненты:

- HeaderBlock (Шапка с навигатором)

- MainBlock (Основа)

- Requisition ( отправить заявку)

- Comfort ( комфорт секции)

- Advantage (Преимущество секции)

- Footer (нижний)

## Main.js - в исходном коде клиентского дополнения

Добавить файл scss(шрифт, вариаблы, стиль) и store – хранилище

import { createApp } from 'vue'

import App from './App.vue'

import store from './store'

import '@/scss/fonts.scss'

import '@/scss/\_vars.scss'

import '@/scss/style.scss'

createApp(App).use(store).mount('#app')

## index.html – Основа сайта. Добавить стиль для векторным значок.

<!DOCTYPE html>

<html lang="">

    <head>

        <meta charset="utf-8">

        <meta http-equiv="X-UA-Compatible" content="IE=edge">

        <meta name="viewport" content="width=device-width,initial-scale=1.0">

        <link rel="shortcut icon" href="./favicon.svg" type="image/x-icon">

        <link rel="stylesheet" href="./css/svg.css">

        <title>Garant fort</title>

    </head>

    <body>

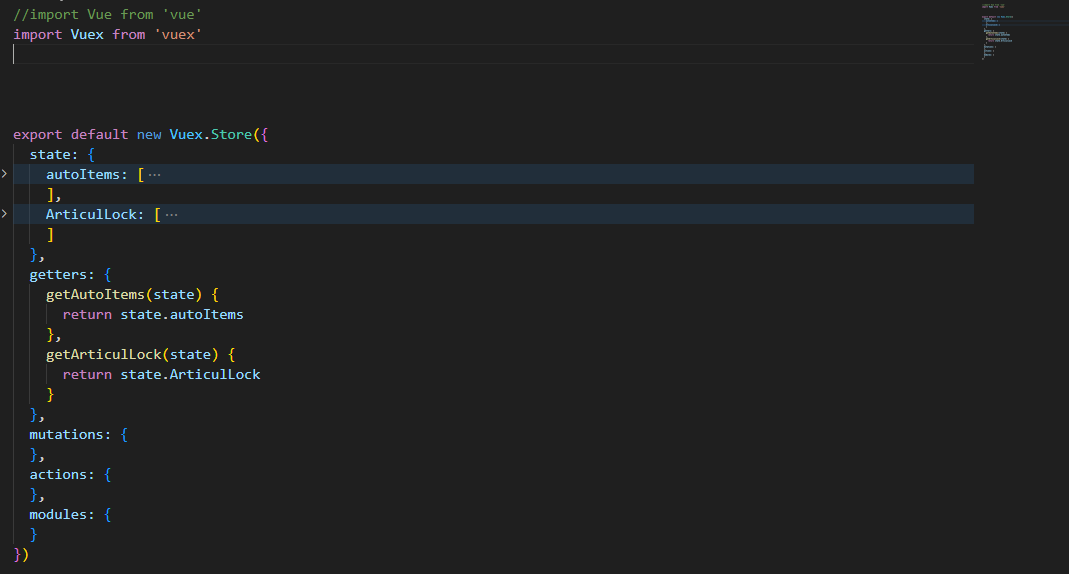
        <div id="app"></div>

        <!-- built files will be auto injected -->

    </body>

</html>

## Store/index.js - Хранилище» — это контейнер, в котором хранится состояние вашего приложения.



Заполнил данных база autoItems и articulLock.

AutoItems – этот отвечает за марок и модели автомобили.

articulLock -этот отвечает стоимость и какой номер замков для соответствующего автомобиля.

## Создал компоненты Header.vue. – шапка

<template>

    <header class="header" id="header">

        <div class="container nav">

            <div class="nav\_left">

                <svg width="132.000000" height="70" viewBox="0 0 132 70" fill="none">

                    <path class="logo" />

                </svg>

                <div class="nav\_title">

                    <h6>{{ title }}</h6>

                    <h6>{{ subtitle }}</h6>

                </div>

            </div>

            <div class="nav\_right">

                <a v-bind:href="'tel:' + tel">

                    <h4>{{ tel }}</h4>

                </a>

                <a v-bind:href="'viber://chat?number=' + mobTel">

                    <svg width="35" height="35" viewBox="0 0 35 35" fill="none" xmlns="http://www.w3.org/2000/svg">

                        <path class="viber" />

                    </svg>

                </a>

                <a v-bind:href="'whatsapp://send?phone=' + mobTel">

                    <svg width="35" height="35" viewBox="0 0 35 35" fill="none" xmlns="http://www.w3.org/2000/svg">

                        <path class="whastapp" />

                    </svg>

                </a>

                <button>

                    {{ btn\_call }}

                </button>

            </div>

            <div class="nav\_gambuger">

                <input id="nav\_gambuger\_toggle" type="checkbox" />

                <label class="nav\_gambuger\_btn" for="nav\_gambuger\_toggle">

                    <span></span>

                </label>

                <ul class="nav\_gambuger\_box">

                    <li> <a class="nav\_gambuger\_box\_tel" v-bind:href="'tel:' + tel">

                            <h4>{{ tel }}</h4>

                        </a></li>

                    <li><a class="nav\_gambuger\_box\_viber" v-bind:href="'viber://chat?number=' + mobTel">

                            <svg width="35" height="35" viewBox="0 0 35 35" fill="none" xmlns="http://www.w3.org/2000/svg">

                                <path class="viber" />

                            </svg>

                        </a>

                        <a v-bind:href="'whatsapp://send?phone=' + mobTel">

                            <svg width="35" height="35" viewBox="0 0 35 35" fill="none" xmlns="http://www.w3.org/2000/svg">

                                <path class="whastapp" />

                            </svg>

                        </a>

                    </li>

                    <li> <button>

                            {{ btn\_call }}

                        </button></li>

                    <li>

                        <h5 class="nav\_gambuger\_title">Адреса магазинов</h5>

                        <p class="nav\_gambuger\_subtitle"> Наш офис:</p>

                        <p class="nav\_gambuger\_text">Санкт-Петербург, ул. Софийская<br> д.8 к.1</p>

                        <p class="nav\_gambuger\_subtitle"> Официальный<br>

                            установочный центр: </p>

                        <p class="nav\_gambuger\_text">Ул, Шоссейная 1Г, Бугры</p>

                        <p class="nav\_gambuger\_subtitle"> Отдел продаж:</p>

                        <p class="nav\_gambuger\_text">+7(812) 989-42-32</p>

                        <p class="nav\_gambuger\_subtitle">Оптовый отдел:</p>

                        <p class="nav\_gambuger\_text">+7(903) 095-61-77</p>

                        <p class="nav\_gambuger\_subtitle">E-mail:</p>

                        <p class="nav\_gambuger\_text">garant-spb-78@mail.ru</p>

                    </li>

                </ul>

            </div>

        </div>

    </header>

</template>

<script>

export default {

    name: 'headerBlock',

    data() {

        return {

            title: 'Гарант-СПБ',

            subtitle: 'Официальный дистрибьютор ООО "ФИЛМ"',

            btn\_call: 'Заказать звонок',

            tel: '+7(812)989-42-32',

            mobTel: '79270000000',

        }

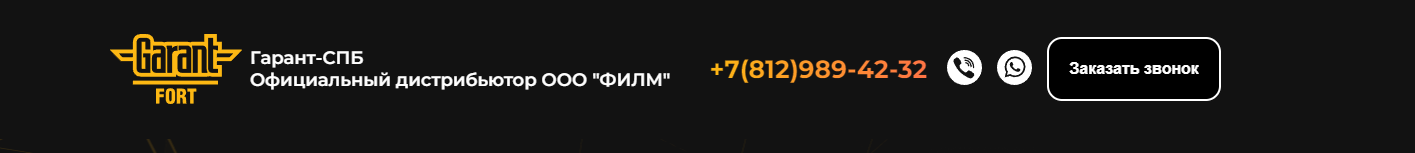
    },

}

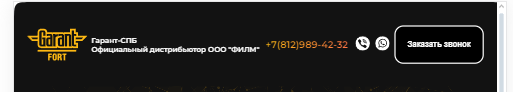
</script>

Так результат сайта

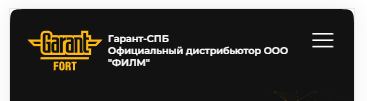
**Десктопная версия**



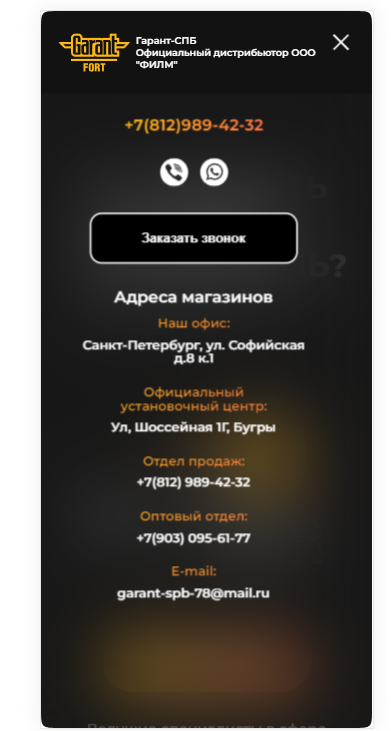
Планшетная версия



Мобильная версия



Не только гамбургер - это тип меню, который обычно отображается в виде трех линий, расположенных горизонтально. Чтобы открыть его, просто нажмите на значок меню в верхнем правом углу экрана.



## Создал компоненты main.vue – основа

<template>

    <section class="main">

        <div class="container">

            <div class="main-content">

                <div class="main-top">

                    <div class="main-top-content">

                        <h1>{{ title }}</h1>

                        <h4>{{ subtitle }}</h4>

                        <button class="btn" onclick="location.href='#requisition'">

                            <h6>{{ btn\_pickUp }}</h6>

                        </button>

                    </div>

                    <div class="main-top-image">

                        <img src="@/assets/top-image.png">

                    </div>

                    <div class="main-top-button">

                        <button class="btn" onclick="location.href='#requisition'">

                            <h6>{{ btn\_pickUp }}</h6>

                        </button>

                    </div>

                </div>

                <div class="main-content">

                    <h4 class="main-content-title">{{ info\_content }}</h4>

                    <div class="main-content-autoTitle">

                        <h4><span v-for="auto in getAutoItems" :key="auto.id">{{ auto.name }}<span

                                    v-if="auto.id < 22">,&#32;

                                </span>

                                <span v-else>.</span></span>

                        </h4>

                    </div>

                    <div class="main-content-autoImg">

                        <img v-for="auto in   getAutoItems  " :key="auto.id" :src="require(`@/assets/marok/${auto.logo}`)"

                            alt="">

                    </div>

                </div>

            </div>

        </div>

    </section>

</template>

<script>

import { mapGetters } from 'vuex';

export default {

    name: 'mainBlock',

    data() {

        return {

            title: 'ЧТО ВЫБРАТЬ ЗАЩИЩАЯ АВТОМОБИЛЬ?',

            subtitle: 'УЖЕ УСТАНОВЛЕНА МЕХАНИЧЕСКАЯ ЗАЩИТА?',

            btn\_pickUp: 'ПОДОБРАТЬ ЗАМОК',

            info\_content: 'Ведущие специалисты в сфере защиты автомобилей от угона советуют использовать от угона электронные и механические устройства в комплексе. Механические блокираторы – оптимальный выбор, который отлично сочетается с сигнализацией и автозапуском, а так же со штатной базовой системой защиты. Доступно:',

        }

    },

    computed: {

        ...mapGetters(['getAutoItems'])

    }

}

</script>

Отмечаю, что у div main-content-autoTitle, main-content-autoImg Чтобы получить данные из Store.

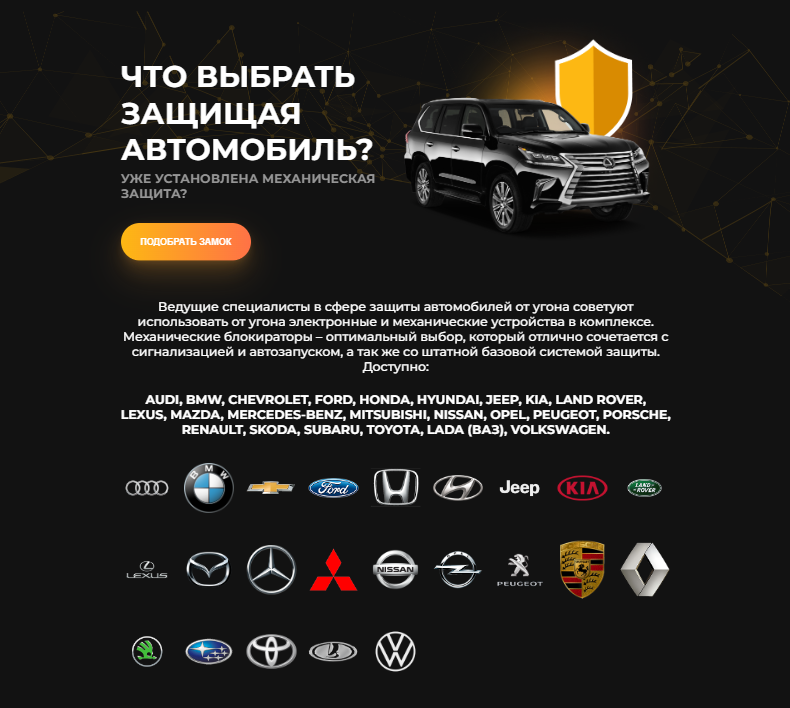
autoTitle – отвечает за название марок автомобилей.

autoImg – отвечает за картинки марок автомобилей.

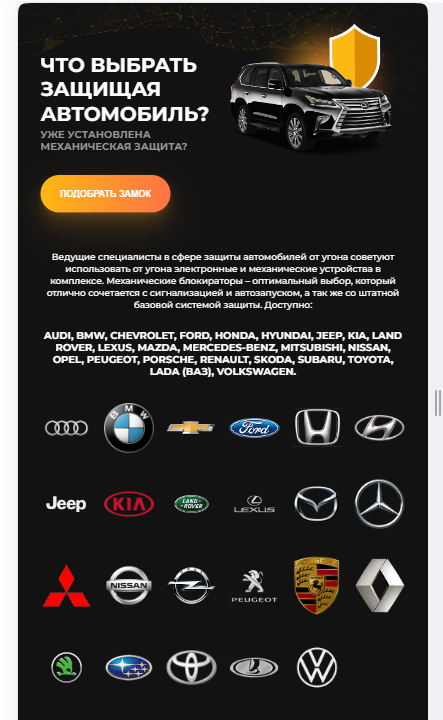
Кнопка « подобрать замок» перейдет на компонента requisition (отставить заявку)

Так результат сайта

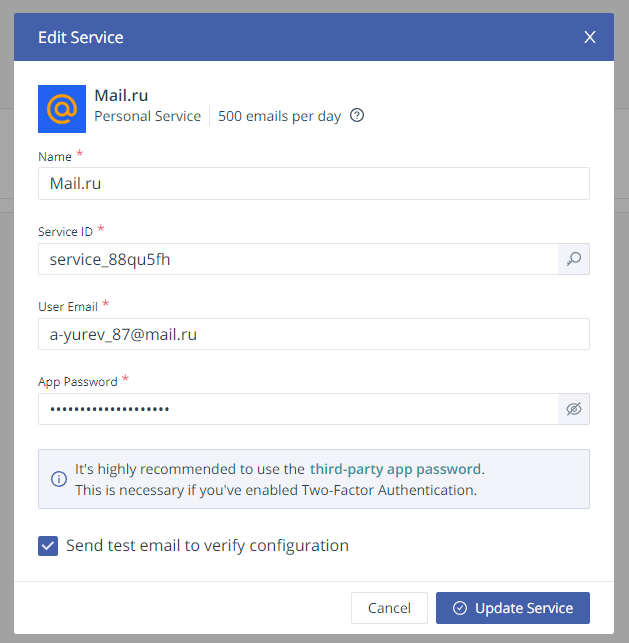
**Десктопная версия**



Планшетная версия мобильная версия

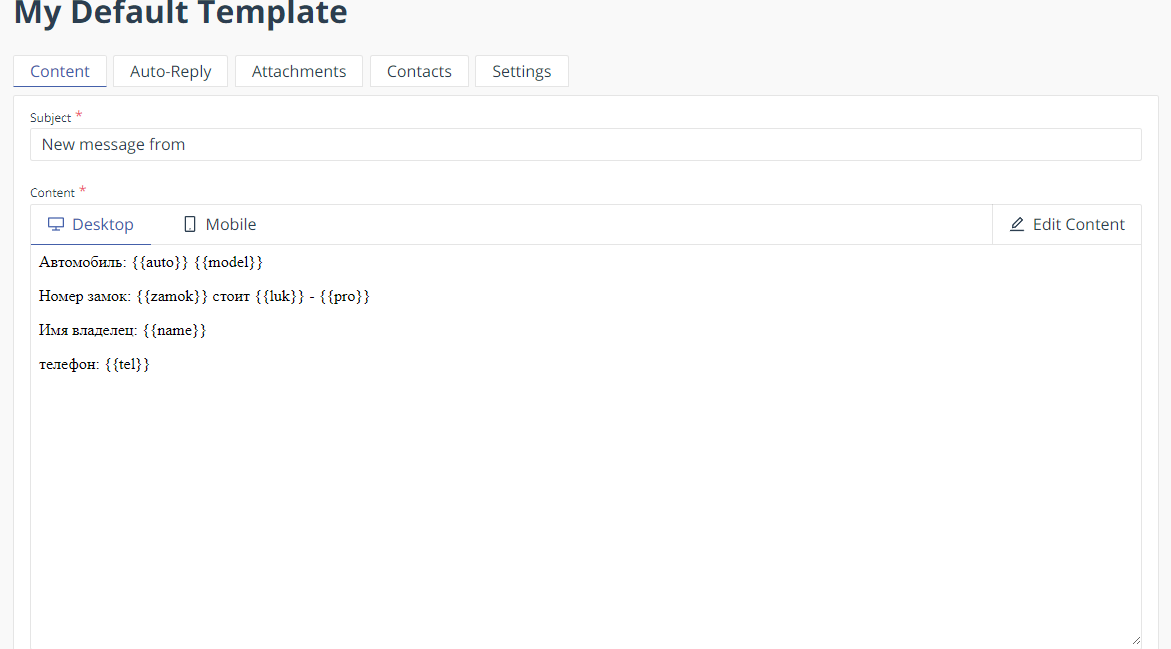
 

## Создание сервер и шаблоны emailJS



Установил серверу к майлу, чтоб сообщает заявку к тому.

Оформил шаблоны писем.



## Создал компоненты requisition.vue. оставить заявку

<template>

    <section id="requisition" class="requisition">

        <div class="container">

            <div class="requisition-content">

                <h2>{{ submitYourApplication }}</h2>

                <h4>{{ submitNumTelefon }}</h4>

                <div class="requisition-block">

                    <h4>Выберите марку</h4>

                    <div class="requisition-ListBlok">

                        <div class="requisition-ListBlok-left">

                            <select v-model="selectedMarka">

                                <option v-for="auto in getAutoItems" :key="auto.id" v-bind:value="auto.id">

                                    <h6>{{ auto.name }}</h6>

                                </option>

                            </select>

                        </div>

                        <div class="requisition-ListBlok-right">

                            <select v-model="selectedModel">

                                <option v-for="models in getAutoModel(selectedMarka)" :key="models.id"

                                    v-bind:value="models.id">

                                    <h6>{{ models.name }}</h6>

                                </option>

                            </select>

                        </div>

                    </div>

                    <div class="requisition-lock">

                        <h4><span class="requisition-lock-gradition">Артикул замка:

                            </span>{{ getArticulLock[selectedMarka].idModels[selectedModel].indexLock }}</h4>

                        <h4>{{ getArticulLock[selectedMarka].idModels[selectedModel].Lux }} руб. - Люкс</h4>

                        <h4>{{ getArticulLock[selectedMarka].idModels[selectedModel].Pro }} руб. - Pro</h4>

                    </div>

                    <button class="btn" onclick="location.href='#openModal'">

                        <h6>{{ btn\_requisition }}</h6>

                    </button>

                </div>

            </div>

            <div class="requisition-modal" id="openModal">

                <div class="requisition-modal-dialog">

                    <div class="requisition-modal-content">

                        <div class="requisition-modal-header">

                            <a href="#close" title="Close" class="close">

                                <img src="@/assets/close.svg">

                            </a>

                        </div>

                        <div class="requisition-modal-body">

                            <div class="requisition-modal-title">

                                <h4>Ваши данные под надежной защитой.<br>

                                    Заполните форму ниже:</h4>

                            </div>

                            <div class="input">

                                <input v-model="setName" type="text" name="text" class="requisition-modal-name"

                                    placeholder="Ваше имя" required>

                            </div>

                            <div class="input">

                                <input v-model="setTel" type="tel" name="tel" class="requisition-modal-tel" value="+7"

                                    pattern="\+7\s?[\(]{0,1}9[0-9]{2}[\)]{0,1}\s?\d{3}[-]{0,1}\d{2}[-]{0,1}\d{2}"

                                    placeholder="+7(\_\_\_)\_\_\_-\_\_-\_\_" required>

                            </div>

                            <button class="btn" v-on:click="onlyNumbers()">

                                <h6>отправить заявку</h6>

                            </button>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </section>

</template>

<script type="module">

import emailjs from 'emailjs-com';

import { mapGetters } from 'vuex';

export default {

    name: 'requisitionBlock',

    data() {

        return {

            submitYourApplication: 'Оставть заявку',

            submitNumTelefon: 'Оставьте номер телефона и узнайте решение за 20 минут!',

            selectedMarka: "0",

            selectedModel: "0",

            btn\_requisition: "Оставить заявку",

            failedValidation: {

                setTel: null

            },

            setTel: "",

            setName: ""

        }

    },

    computed: {

        ...mapGetters(['getAutoItems', 'getArticulLock']),

    },

    methods: {

        getAutoModel(num) {

            let arrayModels = this.getAutoItems[num].models;

            return arrayModels;

        },

        onlyNumbers() {

            this.failedValidation.setTel = /^\+7?([0-9]{3})\)?([0-9]{3})([0-9]{2})([0-9]{2})$/.test(this.setTel) ? true : false;

            console.log(this.failedValidation)

            this.secondReg();

        },

        secondReg() {

            if (this.failedValidation.setTel) {

                this.sendEmail();

            }

            else {

                // если проверка не пройдена

                alert('Неверный номер телефон')

            }

        },

        sendEmail() {

            try {

                emailjs.init('iV6Pm357YzCffuisj');

                emailjs.send('service\_88qu5fh', 'template\_6vrtg3k', {

                    auto: this.getAutoItems[this.selectedMarka].name,

                    model: this.getAutoItems[this.selectedMarka].models[this.selectedModel].name,

                    zamok: this.getArticulLock[this.selectedMarka].idModels[this.selectedModel].indexLock,

                    luk: this.getArticulLock[this.selectedMarka].idModels[this.selectedModel].Lux,

                    pro: this.getArticulLock[this.selectedMarka].idModels[this.selectedModel].Pro,

                    name: this.setName,

                    tel: this.setTel

                });

                alert('успешно отправить данных');

            } catch (error) {

                alert('Ошибки не отправлено');

                console.log({ error })

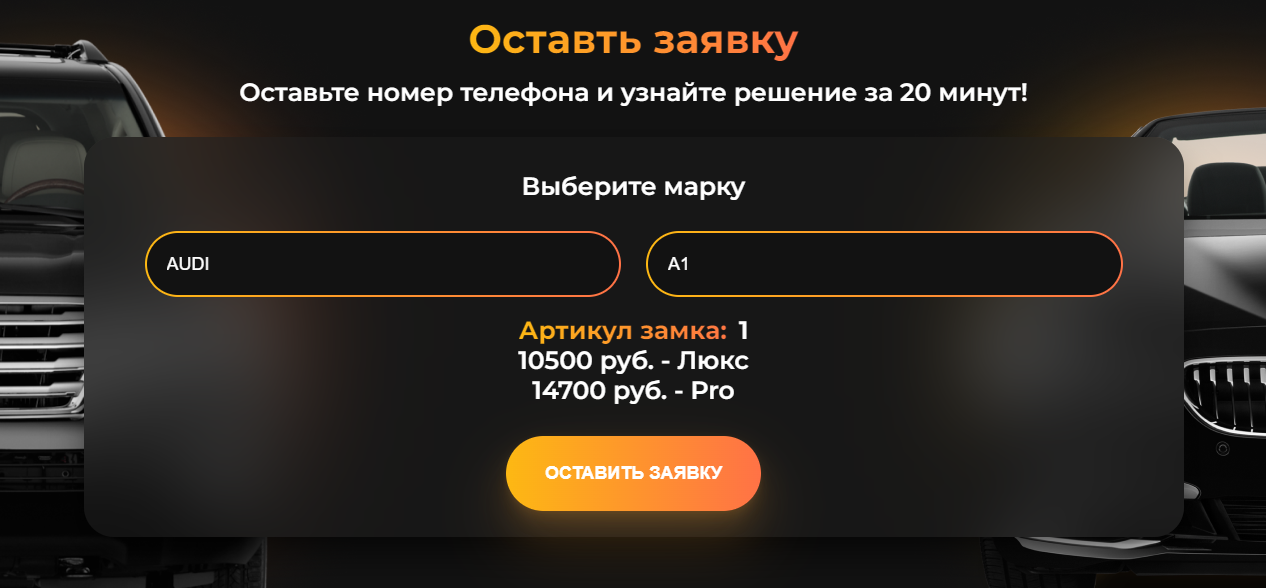
            }

        }

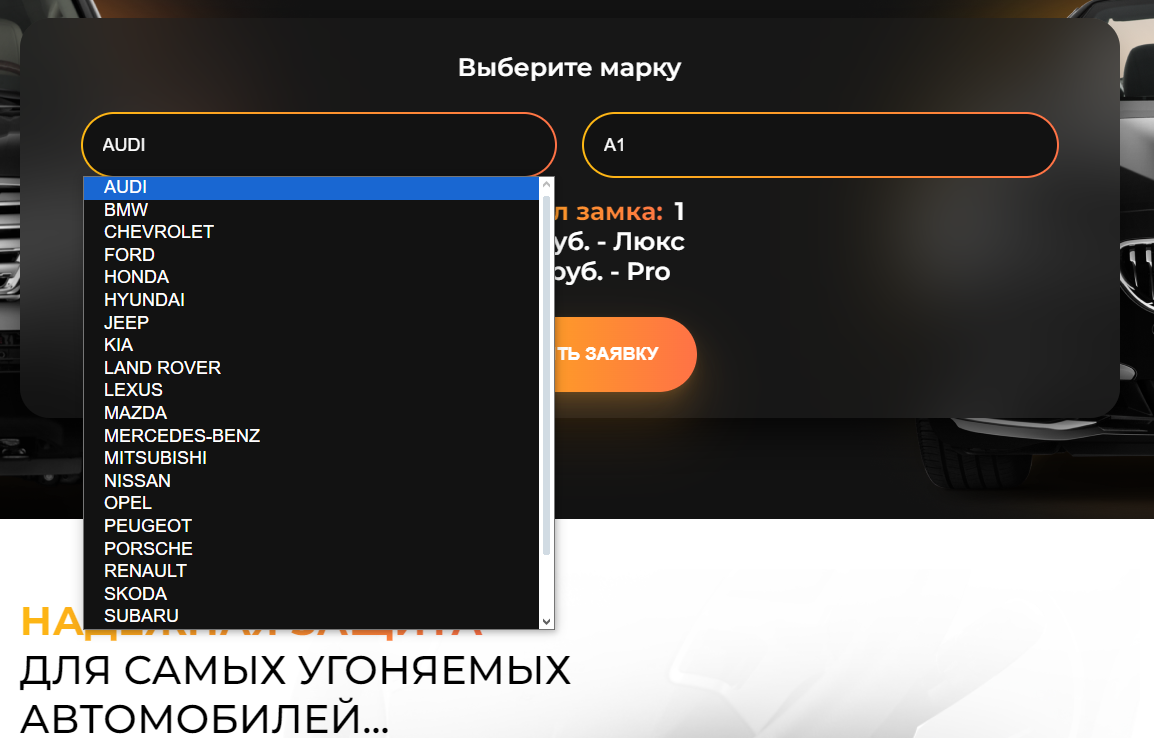
    }

}

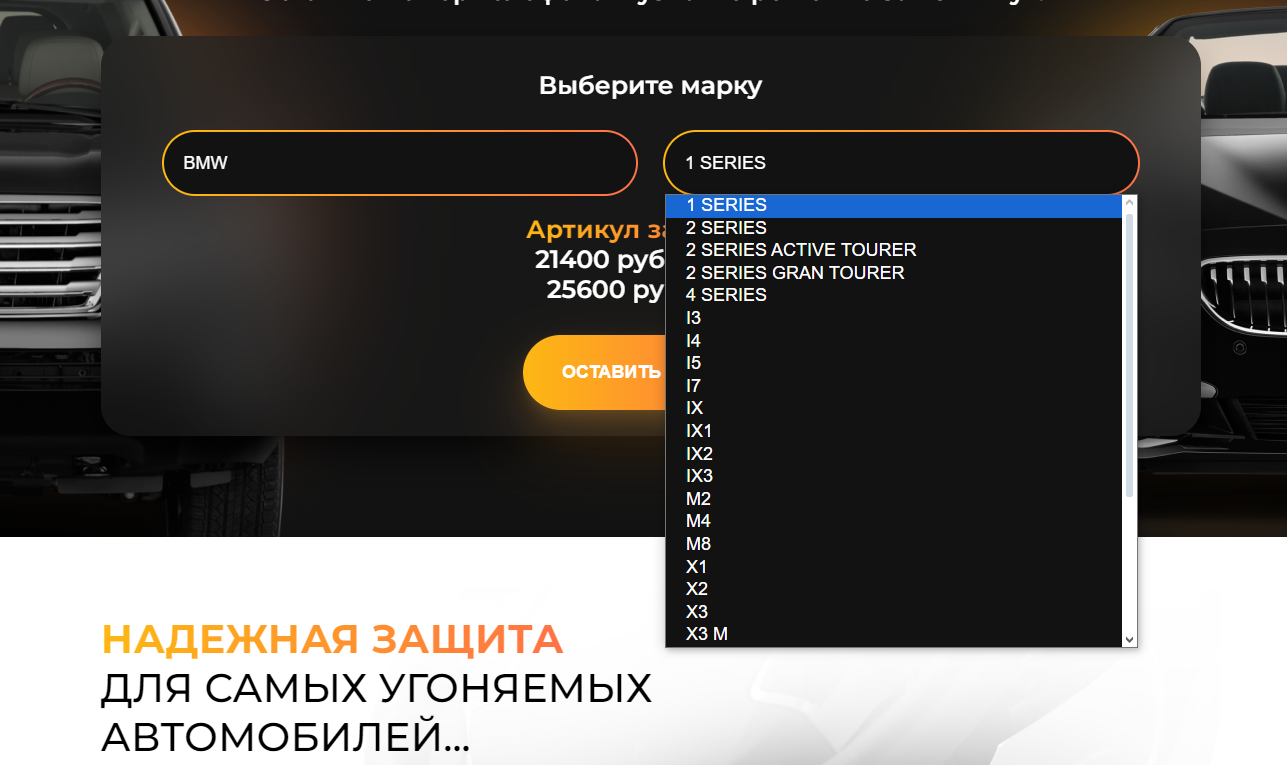
Отмечаю, что два списка ( левый – марок автомобиля, правый – модель автомобили) получить данных от Store. Можно подобрать какого марка и модели автомобили, чтоб получить о стоимость замок автомобиля. И оставить заявку.



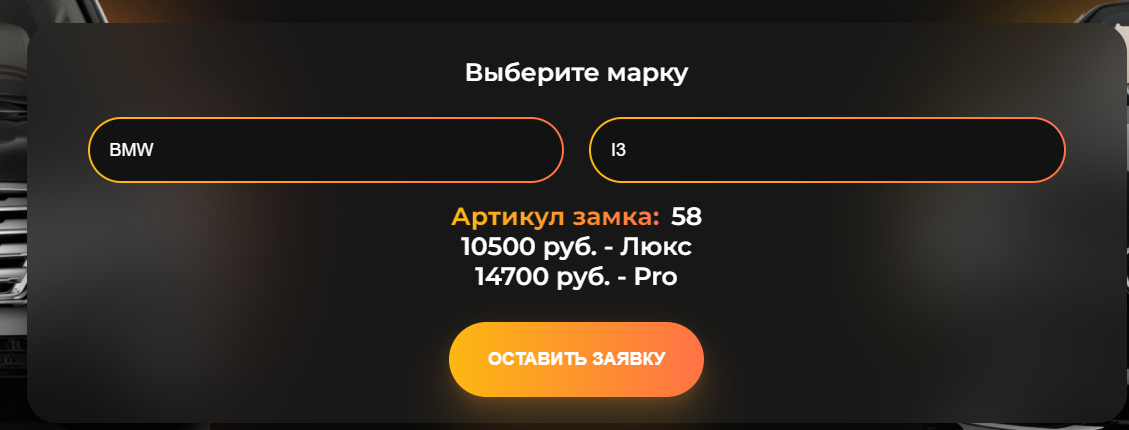
Выпадающий список левый.



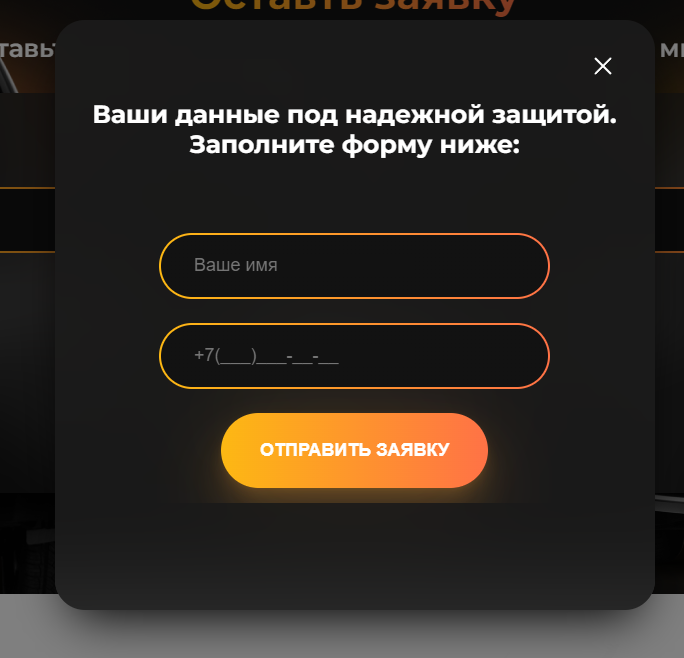
Подобрал марок BMW, то правый список подобреет каких-то моделях по BMW из данных store.



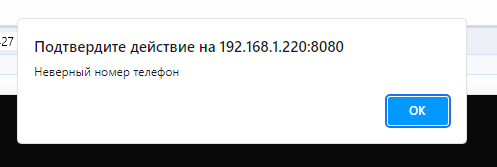
Выбрать модели I3, получить данных о стоимостей замок для него автомобиля.



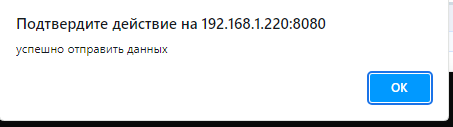
И нажмите кнопку оставить заявку



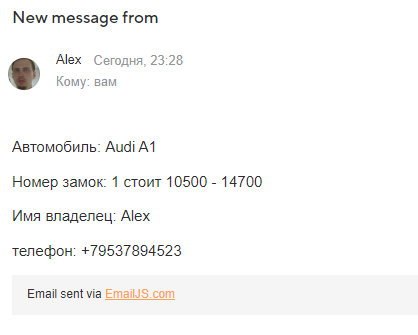
Введите имя и телефон. Если не соответствует номер телефон, то вывод сообщения об ошибках.



Если вводит правильно номер телефон, то успешно сообщения об отправке.



О поступлении сообщения в майле от заказчике.



## Создал компоненты Protection.vue – о защите

<template>

    <section class="protection">

        <div class="container ">

            <div class="protection-background">

                <div class="protection-header">

                    <h2><span>Надежная защита</span><br>для самых угоняемых автомобилей...</h2>

                    <h6>... разработан механический бесштыревой блокиратор рулевого вала Гарант Форт</h6>

                </div>

                <div class="protection-colonna">

                    <div class="protection-content">

                        <ul class="protection-list">

                            <li v-for="list in   protectionList  " :key="list.id">

                                <div class="img"><img :src="require('@/assets/' + list.img)"></div>

                                <h6>{{ list.title }}</h6>

                            </li>

                        </ul>

                    </div>

                    <div class="protection-empty"></div>

                </div>

            </div>

        </div>

    </section>

</template>

<script >

export default {

    name: 'ProtectionBlock',

    data() {

        return {

            protectionList: [

                {

                    img: 'zamok.svg',

                    title: 'максимальная криминальная стойкость механизма секретов Abloy Protec (1,97 млрд комбинаций)'

                },

                {

                    img: '4mm.svg',

                    title: 'замок установлен в толстостенном корпусе толщиной не менее 4 мм'

                },

                {

                    img: '10.svg',

                    title: 'рулевой вал блокируется ригелем из высокопрочной стали диаметром 10 мм'

                },

                {

                    img: 'protection.svg',

                    title: 'полностью безопасен в эксплуатации'

                },

                {

                    img: 'freez.svg',

                    title: 'надёжен в разных климатических условиях'

                },

            ]

        }

    },

}

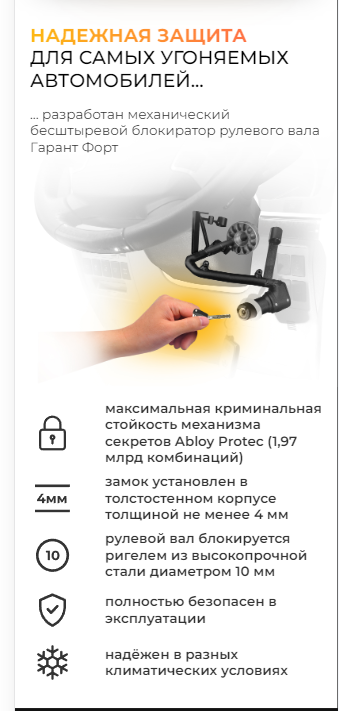
</script>

У data переменной protectionList составит список с изображением при помощей v-for

**Десктопная версия**



Планшетная версия мобильная версия

## Создал компоненты Comfort.vue – о комфорте

<template>

    <section class="comfort">

        <div class="container">

            <div class="comfort-content">

                <h2><span>...или привычный для тебя</span>комфорт?</h2>

                <div class="image">

                    <img src="@/assets/setki.svg" alt="setki" class="setki">

                    <img src="@/assets/comfort.png" alt="" class="img">

                </div>

                <h6>Простота использования блокиратора Гарант Форт достигается бесштыревой схемой конструкции замка. Для

                    блокировки и разблокировки рулевого вала достаточно повернуть ключ в механизме секретов, расположенном в

                    удобном месте рядом с рулевым колесом. Это обеспечивает максимальный комфорт в эксплуатации замка</h6>

                <div class="button">

                    <button class="btn">

                        Где купить?

                    </button>

                </div>

            </div>

        </div>

    </section>

</template>

<script>

export default {

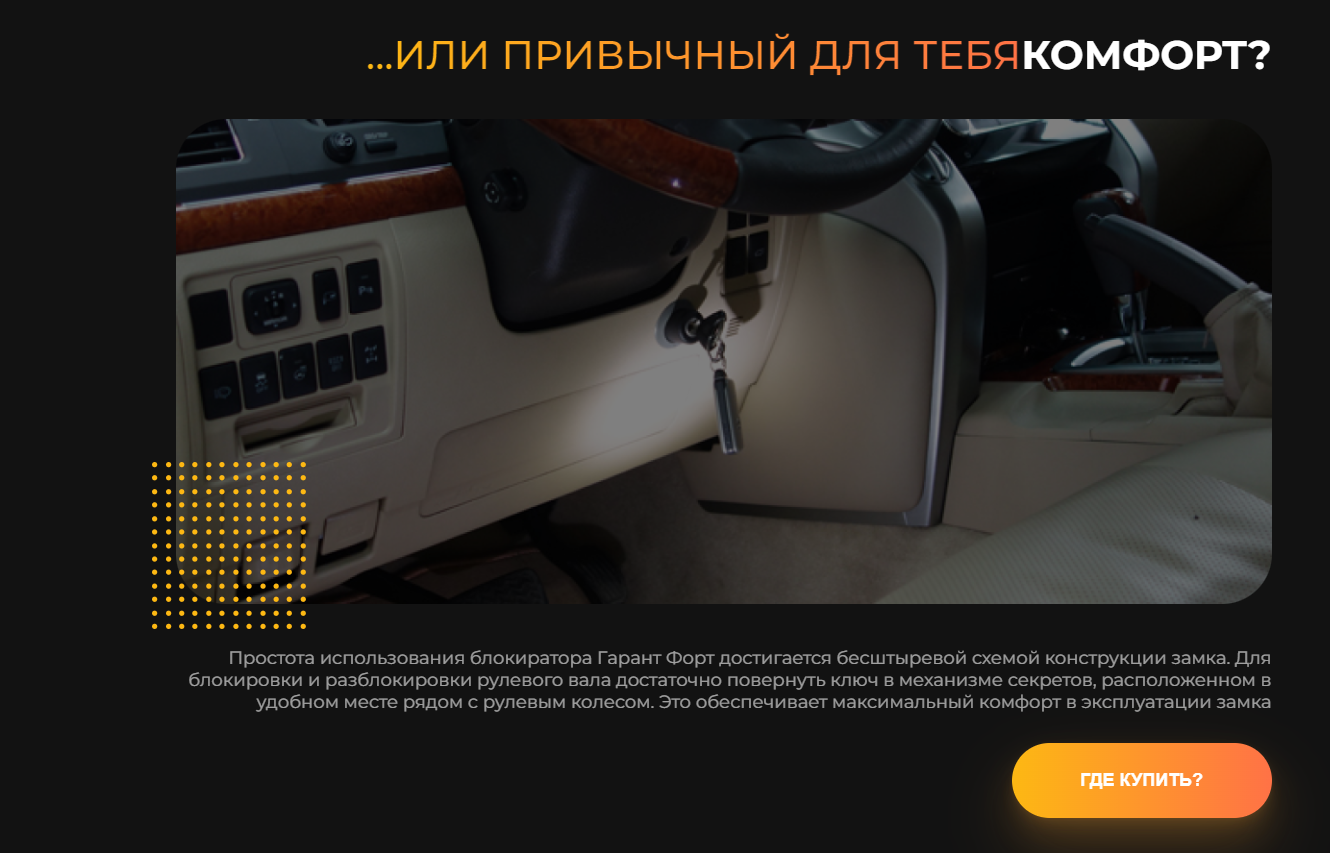
    name: 'ComfortBlock',

}

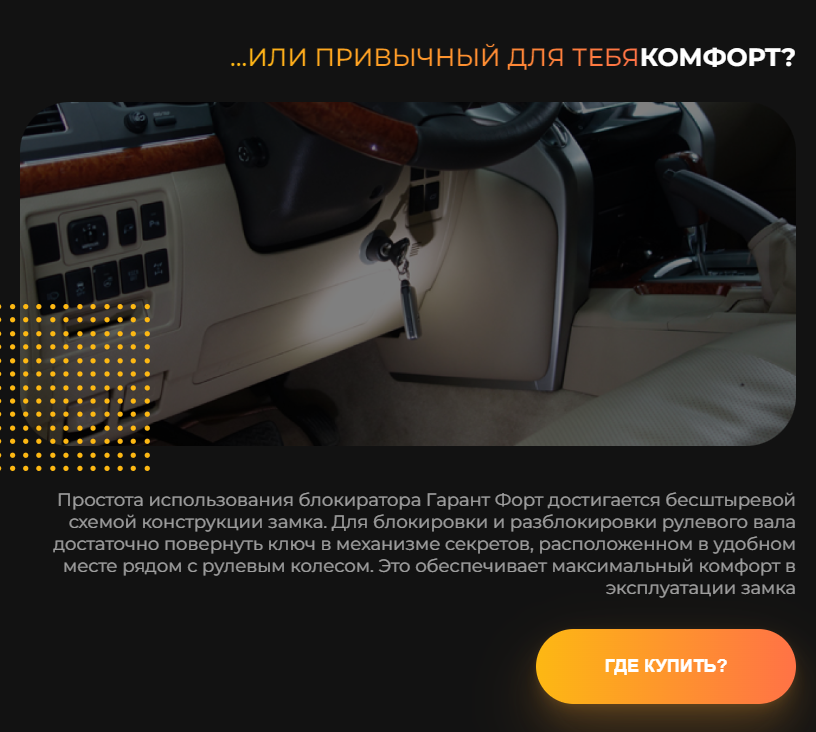
</script>

Так результат

**Десктопная версия**



Планшетная версия мобильная версия



## **Создал компоненты advantage.vue – о преимуществе**

<template>

    <section class="advantage">

        <div class="container">

            <div class="advantage-outContent">

                <h2>Почему мы?</h2>

                <h3>Мы официальный дистрибьютор</h3>

                <div class="advantage-colonna">

                    <div class="advantage-content">

                        <ul class="advantage-lists">

                            <li v-for="list in advantageLists" :key="list.id">

                                <img :src="require('@/assets/' + list.img)" alt="">

                                <h5>{{ list.text }}</h5>

                            </li>

                        </ul>

                    </div>

                    <div class="advantage-image">

                        <img src="@/assets/certificate.png" alt="" class="sertificate">

                        <img src="@/assets/minisetki.svg" alt="" class="minisetki">

                    </div>

                </div>

            </div>

        </div>

    </section>

</template>

<script>

export default {

    name: 'AdvantageBlock',

    data() {

        return {

            advantageLists: [

                {

                    img: 'iconMedal.svg',

                    text: 'Уникальный оригинальный Замок Гарант Блок Люкс.'

                },

                {

                    img: 'iconProtection.svg',

                    text: 'Продукция застрахована от подделок.'

                },

                {

                    img: 'iconRub.svg',

                    text: 'Цена от производителя.'

                },

                {

                    img: 'iconLike.svg',

                    text: 'Гарантия качества.'

                },

                {

                    img: 'iconTools.svg',

                    text: 'Доверьте установку профессионалам.'

                },

            ]

        }

    },

}

</script>

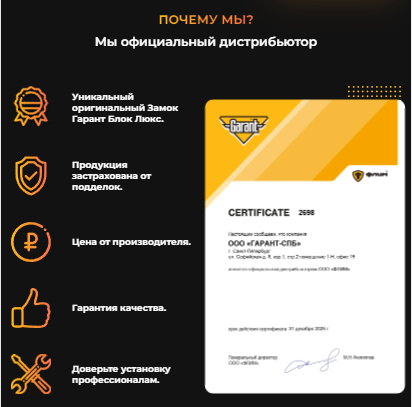
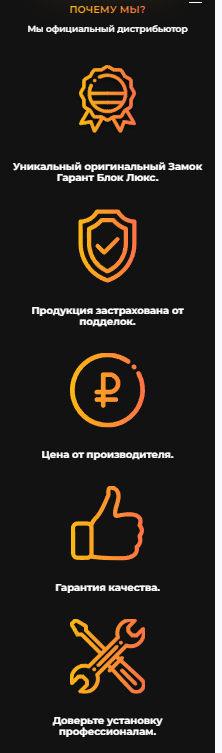
У data переменной advantageList составит список с изображением при помощей v-for

Так результат

**Десктопная версия**



Планшетная версия мобильная версия



## Создал компоненты footer.vue - нижний

<template>

    <footer class="footer">

        <div class="container">

            <div class="footer-title">

                <h2><span>Блокиратор руля форт защита</span>и комфорт</h2>

            </div>

        </div>

        <div class="footer-image">

            <img class="footer-image-warp" src="@/assets/footer.png" alt="">

            <img src="@/assets/headlights.png" alt="" class="footer-image-blink">

        </div>

        <div class="container">

            <div class="footer-button">

                <button class="btn" onclick="location.href='#requisition'">

                    <h6>Купить сейчас</h6>

                </button>

            </div>

        </div>

        <div class="footer-bottom">

            <a href="#header"><img src="@/assets/footer-logo.svg"></a>

            <p class="text">©ГАРАНТ-СПБ 2021 Все права защищены</p>

        </div>

    </footer>

</template>

<script>

export default {

    name: 'FooterBlock',

}

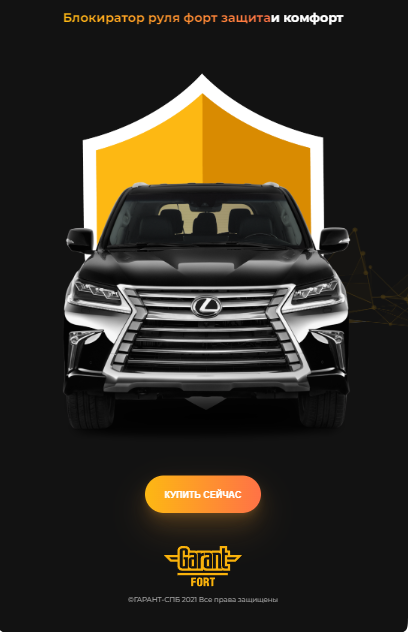
</script>

Так результат

**Десктопная версия**



Планшетная версия мобильная версия



# Подведение итогов, оценка результатов и заключение

В результате успешно выполненного дипломного проекта было создано полноценное веб-приложение из макета Figma для подбора замков по марке автомобиля с использованием Vue

В данном дипломном проекте был разработан веб-агрегатор информации с использованием технологий HTML, CSS, JavaScript, Vue.js, Node.js, Figma.

Данный дипломный проект направлен на создание удобного и эффективного веб-приложения для подбора замков по марке автомобиля с использованием макета Figma и Vue.js. Разработка такого приложения поможет пользователям сэкономить время и упростить процесс поиска нужных замков, а также обеспечит им приятный и привлекательный интерфейс.

В целом, проект был успешно завершен и достиг своей цели – создания веб-агрегатора информации с использованием современных технологий и инструментов разработки.

# Список использованный литературы

**HTML, CSS, JS, Node.js, Vue.js, Figma, PIXSO**

1. W3Schools - https://www.w3schools.com/

2. Codecademy - https://www.codecademy.com/learn/learn-html и по JS - https://www.codecademy.com/learn/introduction-to-javascript

3. FreeCodeCamp - https://www.freecodecamp.org/

4. MDN Web Docs - https://developer.mozilla.org/en-US/docs/Learn

5. CSS-Tricks - https://css-tricks.com/

6. HTML Dog - https://htmldog.com/

7. Learn to Code HTML & CSS - https://learn.shayhowe.com/html-css/

8. TutorialsPoint - https://www.tutorialspoint.com/index.htm

9. SoloLearn - https://www.sololearn.com/

10. Документация на русском языке JS - https://learn.javascript.ru/

11. https://skillbox.ru/media/code/chto\_takoe\_html/

12. https://gb.ru/posts/chto-takoe-css-obyasnyaem-prostymi-slovami

13. https://developer.mozilla.org/ru/docs/Learn/JavaScript/First\_steps/What\_is\_JavaScript

14. Официальная документация Node.js: <https://nodejs.org/en/docs/>

15. Официальная документация VUE.CLI <https://v3.vuex.vuejs.org/>

16. Официальная документация Vue.JS <https://ru.vuejs.org/>

17. Руководство по Figma <https://slashdesigner.ru/figma-guide?ysclid=lsbjiplb4c228170156>