



## **Разработка веб-сайта для кондитерской «От зайки»**

IT-специалист:

Frontend-программист

**Зевина Е.Ю.**

## Оглавление

1. Введение.....	3
1.1. Актуальность темы .....	3
1.2. Цели и задачи работы .....	4
1.3. Обзор основных технологий и инструментов.....	5
2. Теоретический обзор технологий Vue.js .....	6
3. Архитектура и проектирование .....	9
3.1. Обзор требований к веб-сайту .....	9
3.2 Анализ вариантов использования .....	11
3.3 Разработка структуры веб-сайта.....	12
3.4 Создание макетов веб-страниц .....	15
3.5 Выбор архитектурных подходов и паттернов.....	19
3.6 Проектирование компонентной структуры.....	21
4. Верстка и стилизация веб-сайта .....	23
4.1 Разметка HTML .....	23
4.2 Стилизация CSS .....	26
4.2.1 Подключение шрифтов к проекту.....	28
Основные параметры шрифта: .....	28
4.2.2 Анимация и переходы .....	30
4.2.3 Адаптивная верстка .....	31
4.3 Методология БЭМ .....	34
4.4 Препроцессор SCSS (Sass) .....	36
5. Система контроля версий Git.....	39
6. Разработка проекта на Vue.js .....	41
6.1. Структура проекта .....	41

6.2. Настройка и установка окружения разработки на Vue .....	45
6.3 Реализация компонентов и страниц сайта.....	47
6.4 Работа с маршрутизацией и управлением состоянием приложения .....	64
6.4.1 Vue Router .....	64
6.4.2 Vuex .....	68
6.5 Vue Devtools .....	70
7. Заключение .....	72
7.1. Реализованные результаты и их значимость.....	72
7.2. Рекомендации для дальнейшего развития и улучшения .....	73
8. Список использованной литературы.....	75
9. Приложение .....	75

# 1. Введение

## 1.1. Актуальность темы

Актуальность темы проекта по разработке веб-сайта обусловлена современными трендами в сфере цифровизации бизнеса. В современном мире все больше потребителей предпочитают искать информацию о продуктах и услугах онлайн, в том числе и в сфере питания. Веб-сайт является важным инструментом для привлечения новых клиентов, укрепления имиджа компании и повышения удобства обслуживания. Разработка сайта позволит создать современный и динамичный ресурс, который будет привлекать внимание потенциальных клиентов и делать процесс заказа продукции более удобным и эффективным.

Целевая аудитория (ЦА) для сайта может быть достаточно широкой и разнообразной, поэтому важно учитывать интересы и потребности каждой из категорий при разработке и продвижении сайта. ЦА включает в себя людей разного возраста, которые интересуются сладостями, в особенности те, которые знают про японское пирожное моти, и могут быть потенциальными клиентами.

Основные категории целевой аудитории следующие:

1. Любители сладкого: это люди, которые обожают сладости. Они могут быть покупателями кондитерской продукции как для себя, так и в качестве подарков.
2. Родители, которые хотят угодить своим детям угощениями, не жертвуя при этом качеством и натуральностью ингредиентов.
3. Потенциальные партнеры: компании, которые заинтересованы в реализации продукции кондитерской.
4. Гурманы и фуд-блогеры: люди, которые интересуются кулинарией, желают попробовать что-то новое и эксклюзивное, а также делятся своими впечатлениями в социальных сетях.

5. Любители подарков: люди, которые ищут оригинальные и уникальные подарки для близких и друзей, в том числе подарочные коробочки с кондитерскими изделиями.

## **1.2. Цели и задачи работы**

Цели и задачи проекта по созданию вебсайта для кондитерской следующие:

1. Создание современного и привлекательного сайта для кондитерской, соответствующего последним трендам в дизайне. Создание вебсайта позволит увеличить узнаваемость, привлечь новых клиентов и укрепить имидж компании.
2. Обеспечение удобства и функциональности сайта для пользователей, позволяющего легко и быстро ознакомиться с ассортиментом продукции, условиями доставки и оплаты.
3. Разработка интерактивных элементов на сайте, таких как онлайн-заказ продукции, возможность оставить отзывы и подписаться на рассылку об акциях, скидках и новинках, чтобы привлечь больше клиентов и увеличить продажи.
4. Оптимизация веб-сайта для поисковых систем (SEO), чтобы повысить его видимость в поисковых результатах и привлечь больше потенциальных клиентов.
5. Интеграция с социальными сетями для расширения аудитории и увеличения вовлеченности пользователей.
6. Обеспечение мобильной адаптивности сайта, чтобы он отображался корректно на всех устройствах и платформах.
7. Проведение тестирования сайта перед запуском, чтобы гарантировать его стабильную работу и отсутствие ошибок.

### 1.3. Обзор основных технологий и инструментов

- **HTML/CSS/JavaScript:**

Для верстки и стилизации веб-сайта использовались основные языки разметки HTML, каскадные таблицы стилей CSS и язык программирования JavaScript. Эти технологии позволили создать красивый и функциональный дизайн сайта.

- **Vue.js:**

Этот фреймворк был выбран в качестве основной технологии для разработки веб-сайта. Основные принципы Vue.js включают простоту использования, гибкость и производительность. Фреймворк предлагает удобный синтаксис, основанный на директивах, что делает разработку более интуитивной и эффективной. Vue.js также поддерживает компонентный подход к разработке, позволяя создавать независимые и переиспользуемые компоненты, что упрощает поддержку и масштабирование проекта.

- **Vue Router:**

Для управления маршрутизацией на веб-сайте был использован Vue Router. Этот инструмент позволяет создавать маршруты для различных страниц сайта, обеспечивая удобную навигацию пользователя.

- **Vuex:**

Для управления состоянием приложения и обмена данными между компонентами была использована библиотека Vuex.

- **Git:**

Для контроля версий была использована система контроля версий Git. Она позволяет отслеживать изменения в коде, управлять ветками разработки и обеспечивать безопасное хранение кода.

Эти технологии и инструменты были выбраны с целью обеспечить высокое качество разработки веб-сайта для кондитерской и обеспечить его успешное функционирование.

## 2. Теоретический обзор технологий Vue.js

Vue.js — это прогрессивный фреймворк JavaScript, который широко используется для создания интерактивных пользовательских интерфейсов и одностраничных приложений. Был разработан Эваном Ю в 2014 году и за короткое время стал одним из самых популярных инструментов для веб-разработки. Он обладает простым синтаксисом, легким изучением и отличной производительностью, что делает его идеальным выбором для создания динамичных и интерактивных пользовательских интерфейсов.

Одной из ключевых особенностей Vue.js является реактивность. Фреймворк автоматически отслеживает изменения в данных и обновляет пользовательский интерфейс соответствующим образом, что позволяет создавать динамичные и отзывчивые приложения без необходимости ручного управления обновлением данных.

Обоснование выбора Vue.js в качестве фреймворка для веб-приложения кондитерской включает следующие аспекты:

### 1. Простота использования:

предоставляет простой и понятный синтаксис, который упрощает разработку и поддержку кода. Это делает его идеальным выбором для команды разработчиков с разным уровнем опыта.

### 2. Гибкость:

обладает модульной структурой, которая позволяет разработчикам использовать только нужные им функциональности и компоненты. Это дает возможность создания легковесных и масштабируемых приложений, которые могут легко адаптироваться под изменения требований и потребностей.

### 3. Популярность и активное сообщество:

является популярным фреймворком, который имеет большое и активное сообщество разработчиков. Это означает, что есть множество ресурсов, учебных материалов, плагинов и советов, доступных для разработчиков.

Также сообщество постоянно работает над улучшением и дополнением фреймворка.

#### **4. Компонентный подход:**

Одна из ключевых особенностей Vue.js — это его компонентный подход. Это позволяет разработчикам разбить приложение на множество маленьких и переиспользуемых компонентов, что упрощает управление кодом, повторное использование и тестирование. Компонентный подход также способствует созданию интуитивной и привлекательной пользовательского интерфейса.

#### **5. Эффективная реактивность:**

использует эффективный механизм реактивности, который автоматически обновляет DOM-элементы, когда изменяются данные, отслеживаемые в приложении. Это позволяет создавать динамичные и отзывчивые пользовательские интерфейсы без необходимости вручную обновлять DOM.

#### **6. Поддержка и интеграция:**

хорошо поддерживается и предоставляет интеграцию с другими популярными инструментами и библиотеками, такими как Vuex для управления состоянием приложения и Vue Router для маршрутизации.

Это позволяет создавать элегантные и функциональные приложения, которые будут легко масштабироваться и поддерживаться со временем.

Двусторонняя привязка данных в Vue.js — это механизм, который позволяет автоматически синхронизировать данные между моделью и представлением. Это достигается с помощью директивы `v-model`. Например, у нас есть текстовое поле ввода, связанное со свойством `"message"` в экземпляре Vue с помощью директивы `v-model`. Когда значение ввода изменяется, оно автоматически обновляет значение свойства `"message"` в модели. И наоборот, если значение свойства `"message"` в модели изменяется программно, то оно синхронизируется



с текстовым полем ввода. Это означает, что не нужно вручную обновлять представление каждый раз, когда данные изменяются.

Механизм реактивности Vue.js также позволяет создавать динамичные и отзывчивые пользовательские интерфейсы. Можно легко обновлять данные и изменять представление в ответ на эти изменения. Это делает Vue.js идеальным выбором для создания интерактивных веб-приложений.

Vuex — это официальное управление состоянием для Vue.js. Он предоставляет централизованное хранилище для всех компонентов приложения и обеспечивает предсказуемость кода. Vuex предлагает мощные функции, такие как модули, геттеры и более сложные действия и мутации, которые могут быть использованы для управления сложными состояниями в приложении.

## 3. Архитектура и проектирование

### 3.1. Обзор требований к веб-сайту

Для успешной реализации проекта необходимо учитывать следующие основные требования:

- **Интерактивный дизайн:**

Веб-сайт должен иметь современный и привлекательный дизайн, который будет отражать стиль и атмосферу кондитерской. Важно предусмотреть интерактивные элементы, анимации и эффекты для создания приятного пользовательского опыта.

- **Адаптивность:**

Сайт должен быть адаптивным и корректно отображаться на различных устройствах, включая мобильные телефоны, планшеты и настольные компьютеры. Это обеспечит удобство использования сайта для всех пользователей.

- **Каталог продукции:**

На сайте должен быть реализован каталог продукции кондитерской с возможностью просмотра фотографий, описания и цен на товары. Также необходимо предусмотреть фильтрацию и поиск продукции для удобства пользователей.

- **Онлайн-заказы:**

Пользователям должна быть доступна возможность совершать онлайн-заказы продукции с выбором даты и времени доставки или самовывоза. Необходимо предусмотреть форму для заполнения контактной информации и способов оплаты.

- **Контактная информация:**

На сайте должны быть указаны контактные данные кондитерской, включая адрес, телефон и электронную почту. Также желательно предусмотреть форму обратной связи для быстрой связи с клиентами.

- **Социальные сети:**

Важно интегрировать кнопки для быстрой публикации ссылок на продукцию кондитерской в социальных сетях, что поможет расширить аудиторию и привлечь новых клиентов.

Это позволит создать современный, функциональный и привлекательный ресурс, который будет способствовать увеличению продаж и улучшению взаимодействия с клиентами.

## 3.2 Анализ вариантов использования

Use Case — это сценарный план взаимодействия пользователя с программным продуктом, в котором четко прописаны шаги для достижения того или иного результата. Он помогает разработать качественный продукт, от которого пользователь получит ожидаемый результат.

Составить общую картину помогает use case диаграмма. На ней отображаются все участники процесса (в моем случае: пользователь веб-сайта) и все варианты использования ПО.

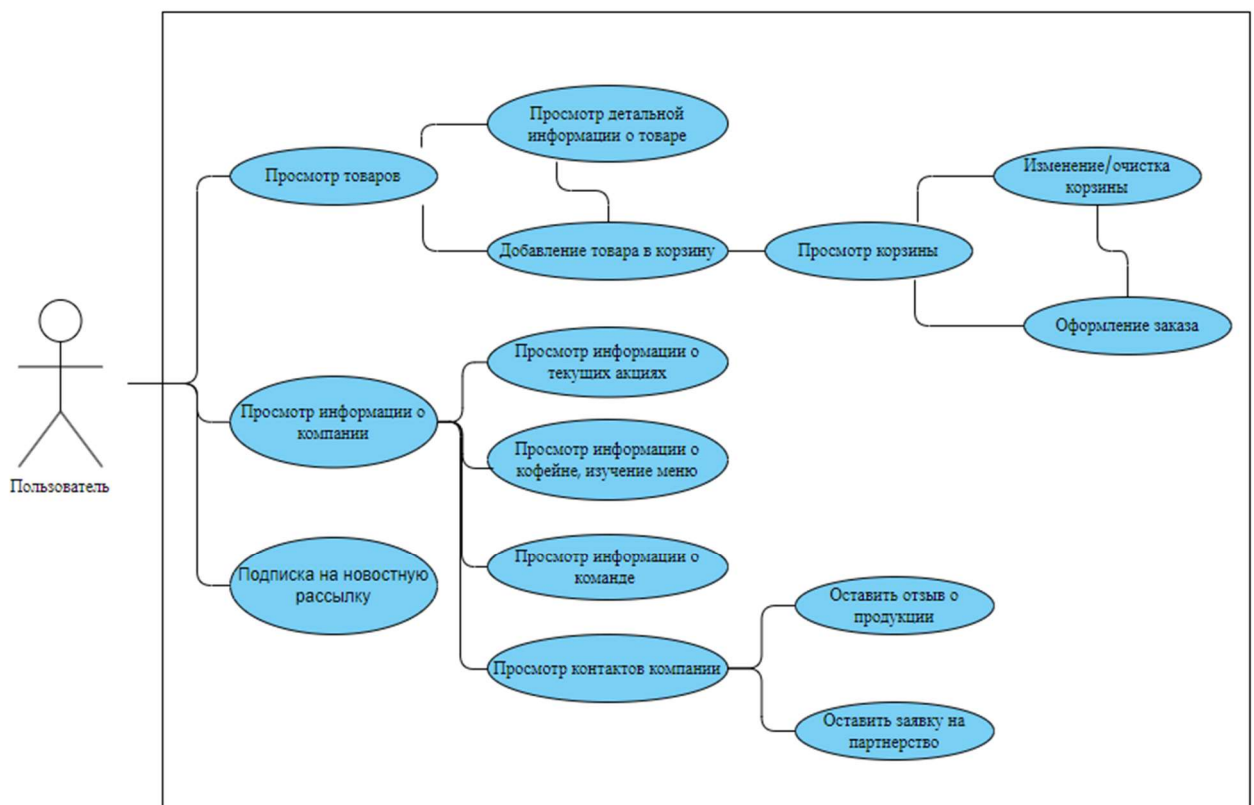


Рис.3.1 Диаграмма use case

### 3.3 Разработка структуры веб-сайта

Разработка структуры играет ключевую роль в обеспечении удобства использования и эффективности работы сайта. Несколько принципов, которые я использовала при проектировании сайта:

- Главная страница должна быть информативной и привлекательной для посетителей. Она должна содержать основную информацию о предлагаемых товарах, акционные предложения, новинки и другие ключевые элементы, чтобы привлечь внимание пользователей.
- Основной товар кондитерской (пирожное моти) должен иметь свою собственную страницу с подробным описанием вкусов, изображениями, ценой и кнопкой "Добавить в корзину". Пользователи должны иметь возможность легко просматривать, фильтровать товары и добавлять их в корзину.
- Корзина должна быть доступна из любой страницы сайта и позволять пользователям просматривать и управлять товарами, которые они добавили для покупки. Также необходимо предусмотреть возможность удаления товаров из корзины.
- Пользователям должно быть удобно оформлять покупку без необходимости регистрации. Необходимо предоставить форму для ввода контактной информации и адреса доставки прямо на странице корзины.
- Структура сайта должна быть логичной и интуитивно понятной.
- Сайт должен быть адаптирован для просмотра на мобильных устройствах, чтобы обеспечить удобство использования пользователями смартфонов и планшетов.

Создание информационной архитектуры позволяет организовать контент веб-сайта в логическую структуру. На данном этапе я определила, какие разделы и подразделы должны быть на сайте, и создала иерархию страниц.

Таблица 3.1. Карта сайта

№п/п	Название страницы	Адрес страницы	Описание
1	Главная страница	/	Главная страница представляет собой визитную карточку кондитерской. Здесь пользователи могут ознакомиться с информацией о компании, узнать о местоположении кофейни, а также увидеть актуальные новости и специальные предложения.
2	Каталог моти	/mochi	На странице каталога пользователи могут ознакомиться с линейкой вкусов пирожных моти. Каждый товар отображается в виде карточки с информацией о нем, ценой и возможностью добавления в корзину необходимого количества.
3	Меню кофейни	/cafe	Пользователи могут ознакомиться с меню кофейни, узнать ее местоположение на карте города и часы работы
4	Новости и акции	/promo	Пользователи могут быть в курсе последних новостей и акций компании. Здесь будут отображаться объявления о предстоящих мероприятиях, новинках в ассортименте, специальных предложениях и промокодах
5	О нас	/about	Пользователи могут узнать более подробную информацию о кондитерской, ее ценностях и команде сотрудников. Здесь может быть представлена информация о сертификациях и достижениях компании.

6	Контакты	/contacts	Пользователи могут найти часы работы кофейни и контактную информацию, такую как телефон, адрес, ссылки на соцсети и электронную почту. Здесь также размещена форма обратной связи для отправки сообщений и вопросов компании, а также запросов на партнерство.
7	Корзина	/cart	Пользователи могут просмотреть список выбранных товаров, редактировать их количество или удалить из корзины. Также здесь отображается общая стоимость товаров и кнопка оформления заказа. На странице также есть форма для оформления заказа, где пользователи заполняют необходимую информацию для доставки и оплаты товара. Здесь можно указать адрес доставки, выбрать способ оплаты и оставить комментарии к заказу.
8	Страница 404	-	Страница 404 поможет пользователю понять, что произошла ошибка, и предложит ему возможные варианты действий

### 3.4 Создание макетов веб-страниц

Графический дизайн сайта — это один из ключевых этапов разработки, который включает в себя создание визуального облика сайта: выбор цветовой гаммы, шрифтов, изображений и других элементов дизайна. Важно помнить, что графический дизайн сайта должен быть не только красивым, легко воспринимаемым и соответствующим бренду компании, но и удобным для пользователей.

Разработка макета веб-сайта помогает визуализировать и оптимизировать пользовательский опыт (UX) и создать привлекательный и интуитивно понятный пользовательский интерфейс (UI) веб-сайта.

Особенности пользовательского интерфейса (UI) и пользовательского опыта (UX), которые также следует учитывать при разработке макета веб-сайта:

- UX (User Experience): UX фокусируется на обеспечении удовлетворения и простоты использования для пользователей. В процессе создания макета я думала о том, как улучшить взаимодействие и навигацию на веб-сайте, создав интуитивно понятные пути, понятные значки и простые действия для достижения целей пользователей.
- UI (User Interface): UI фокусируется на создании чистого, привлекательного и консистентного визуального стиля. Я подобрала цветовую схему, которая соответствует бренду, и постаралась создать понятные и легко различимые элементы интерфейса.

Для выполнения макета сайта я выбрала программу Figma — это удобный инструмент для работы с веб-дизайном. Вдохновение я искала на различных платформах, таких как Behance или Pinterest, собирала и сохраняла идеи, которые нравятся, чтобы использовать их в своем макете.



Создание дизайн-проекта вебсайта в Figma проходило следующие основные этапы:

1. На основании определенной на предыдущем этапе структуры сайта я выделила основные разделы и функциональные элементы, которые должны быть на сайте, и расположила их логически и удобно для пользователя. Проработала дизайн всех страниц вашего веб-сайта и создала компоненты интерфейса, такие как формы, кнопки, карточки, навигационные меню и т. д. Сохранила их в библиотеке компонентов, чтобы легко повторно использовать в макете.
2. Выбрала цветовую схему, которая соответствует бренду компании и помогает создать единый стиль сайта.



*Рис 3.2 Цветовые стили макета сайта*

3. Подобрала подходящие шрифты для текстов на сайте, которые будут четкими и удобными для чтения. Я использовала только два шрифта, чтобы сохранить единый стиль: основной шрифт для подзаголовков и текста (Jost) и дополнительный шрифт для заголовков и акцентных элементов (Blackberry).

#### СТИЛИ ТЕКСТА

Large Title

blackberry / 148

HEADING

Jost / Bold / 108

Heading 1

blackberry / 108

NAV ITEM

Jost / Bold / 66

Heading 2

blackberry / 88

Nav Subitem

Jost / Regular / 24

Heading 3

blackberry / 68

This is body

Jost / Regular / 24

Heading 4

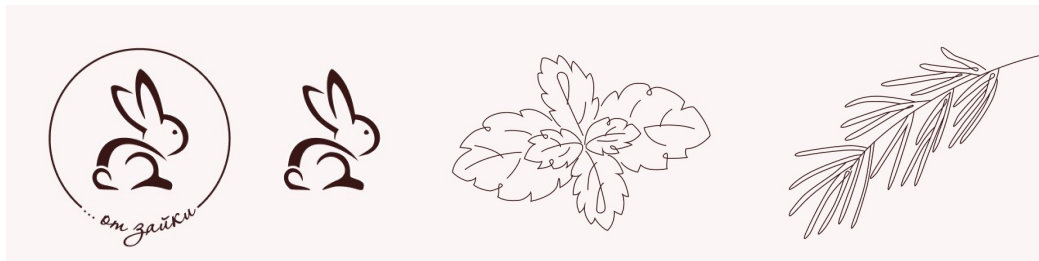
blackberry / 48

This is body - B

Jost / Bold / 24

*Рис 3.3 Стили шрифтов макета сайта*

4. Добавила качественные изображения десертов от профессионального фотографа и графические элементы, которые подчеркивают ключевые моменты и помогают визуально интерпретировать контент на сайте.



*Рис 3.4 Графические элементы сайта*

5. После завершения работы над макетом я произвела опрос потенциальной целевой аудитории, чтобы убедиться, что дизайн сайта соответствует потребностям пользователей. Внесла необходимые изменения и доработки, чтобы достичь оптимального результата.
6. Завершающим этапом был экспорт изображений и компоненты отдельно, чтобы их можно было легко использовать в разработке.

### 3.5 Выбор архитектурных подходов и паттернов

Архитектурный подход MVVM:

- Model (Модель): отвечает за управление данными и бизнес-логикой, как и в MVC. Модель может содержать данные, методы для доступа к данным и методы для обновления данных.
- View (Представление): отвечает за отображение данных пользователю, как и в MVC. В Vue.js это компонент, который отображает данные на основе их состояния.
- ViewModel (Представление-Модель): отвечает за обработку бизнес-логики и предоставление данных представлению. В ViewModel данные из модели подготавливаются для отображения, а также обрабатывается пользовательский ввод и взаимодействие с моделью.

MVVM выбран в качестве архитектурного подхода для проекта по нескольким причинам:

- ✓ хорошо сочетается с Vue.js и упрощает работу с компонентами и управление состоянием приложения через механизмы Vue.
- ✓ ViewModel позволяет легко отделить бизнес-логику от представления и подготовить данные для отображения, повышая переиспользуемость и модульность кода.
- ✓ обладает хорошей поддержкой двустороннего связывания данных, что означает, что представления автоматически обновляются при изменении данных в модели и наоборот.

Выбор MVVM обусловлен его соответствием основным требованиям проекта, таким как простота использования, поддержка компонентного подхода и реактивности, а также возможность эффективно управлять состоянием приложения.

В этом проекте:

- View — это HTML-разметка, которая отображает данные и логику приложения.
- Model — это слой, который содержит бизнес-логику и данные приложения.
- ViewModel — это слой, который связывает модель данных и представление. Он управляет данными и логикой приложения, а также обеспечивает взаимодействие между моделью данных и представлением.

Компоненты страниц (в папке `src/pages`) являются представлениями (View), так как они отображают данные и логику приложения. Vuex — это модель данных (Model), так как он содержит бизнес-логику и данные приложения. Vue Router — это представление модели (ViewModel), так как он связывает модель данных и представление, управляя данными и логикой приложения.

Компоненты Vue позволяют разделить приложение на множество независимых и переиспользуемых частей, каждая из которых сочетает в себе данные, представление и логику, что упрощает разработку и обслуживание кода. Они также обеспечивают четкое разграничение между моделью, представлением и ViewModel, что способствует легкости поддержки и расширения приложения.

Приложение использует архитектуру Single-Page Application (SPA). Это означает, что весь интерфейс приложения загружается на клиентскую сторону один раз и затем обновляется динамически без полной перезагрузки страницы.

### 3.6 Проектирование компонентной структуры

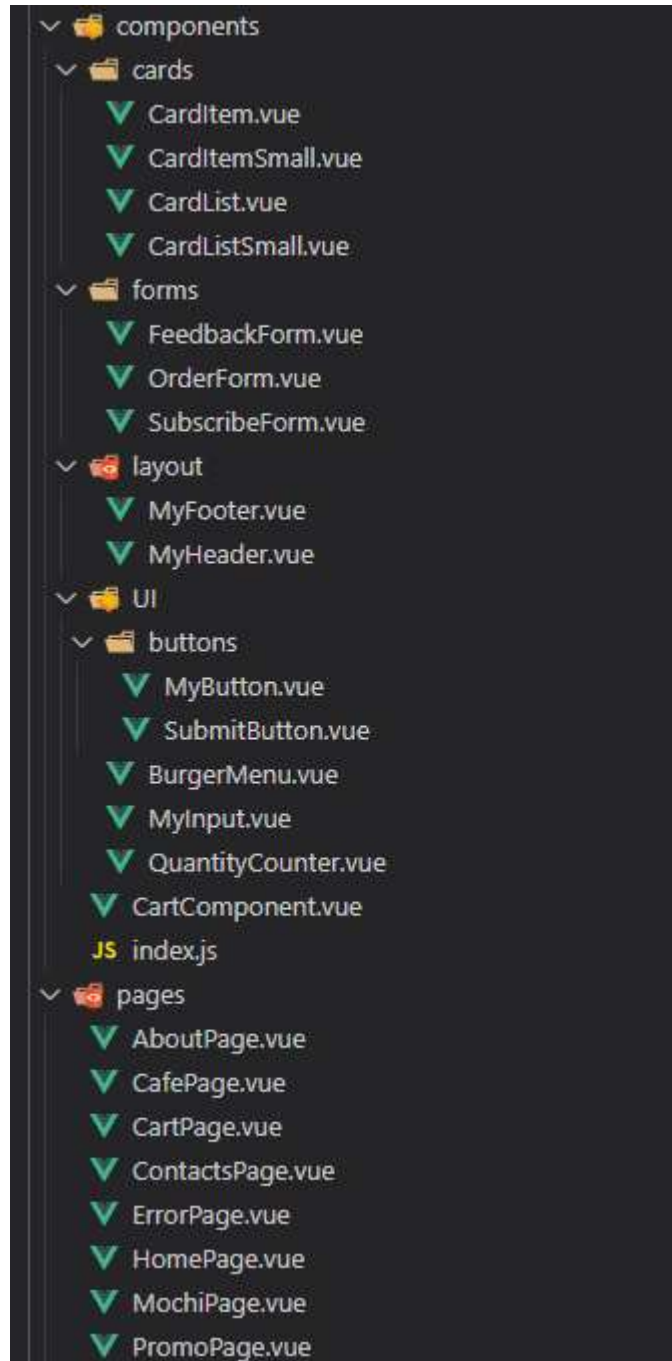
При разработке веб-сайта на платформе Vue.js важным аспектом является проектирование компонентной структуры приложения. Компоненты в Vue.js позволяют создавать модульные и переиспользуемые элементы интерфейса, что способствует удобству разработки, поддержке и масштабируемости проекта.

В контексте сайта кондитерской, компоненты могут быть разделены на несколько категорий:

1. Компоненты для отображения продукции: в данной категории компоненты для отображения карточек товаров, фотографий продукции, описания и цен. Эти компоненты могут быть переиспользованы на различных страницах сайта, где требуется отображение продукции.
2. Компоненты для форм и взаимодействия с пользователем: эти компоненты включают формы для онлайн-заказов, форму обратной связи и т.д. Они обеспечивают удобство использования сайта для пользователей.
3. Компоненты для макета и навигации: в данной категории компоненты для шапки сайта, навигационного меню, подвала и других элементов макета, которые присутствуют на всех страницах сайта.
4. Компоненты страниц сайта: каждый компонент содержит отдельное содержимое, стили и логику, которые отображаются только на соответствующей странице. Все компоненты страниц включены в маршрутизацию Vue Router для навигации между ними.

При проектировании компонентной структуры Vue.js необходимо учитывать принципы ее модульности, переиспользуемости и легкости поддержки. Разделение компонентов на логические блоки позволяет эффективно организовать код и обеспечить гибкость при дальнейшем развитии проекта.

В своем проекте я постаралась создать компоненты, которые являются независимыми и переиспользуемыми. Разделила большие компоненты на меньшие, чтобы упростить разработку и сделать код более читабельным и понятным.



*Рис.3.5 Компоненты проекта*

## 4. Верстка и стилизация веб-сайта

### 4.1 Разметка HTML

HTML (HyperText Markup Language) – это стандартный язык разметки, используемый для создания структуры веб-страниц. HTML состоит из различных элементов, таких как заголовки, абзацы, списки, изображения, ссылки и т. д. HTML начал свой путь в начале 90-х годов как примитивный язык для создания веб-страниц. В 2014 году официально была завершена работа над новым стандартом - HTML5, который фактически произвел революцию, привнеся в HTML много нового.

HTML основан на тегах, которые используются для определения структуры и содержимого веб-страницы. Теги обычно заключаются в угловые скобки и могут иметь свойства или атрибуты для указания дополнительной информации.

Базовая структура документа:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Документ HTML5</title>
  </head>
  <body>
    <div>Содержание документа HTML5</div>
  </body>
</html>
```

Корневым элементом HTML-страницы является тег `<html>`, который содержит элементы `<head>` (заголовок страницы) и `<body>` (основное содержимое страницы). Внутри элемента `<head>` обычно размещаются метаданные, стили CSS и скрипты JavaScript, а внутри `<body>` – текст, изображения и другие элементы контента.



Как правило, элементы имеют открывающий и закрывающий тег, которые заключаются в угловые скобки. Например:

```
<div>Текст элемента div</div>
```

Элементы также могут состоять из одного тега, например, элемент `<br />`, функция которого - перенос строки.

Каждый элемент внутри открывающего тега может иметь атрибуты. Существуют глобальные или общие для всех элементов атрибуты, как например, `style`, а есть специфические, применяемые к определенным элементам, как например, `type`. В HTML5 были добавлены пользовательские атрибуты (custom attributes). Теперь разработчик или создатель веб-страницы сам может определить любой атрибут, предваряя его префиксом `data-`.

Если же возникают затруднения, насколько правильной является создаваемая разметка html, то ее можно проверить с помощью валидатора по адресу <https://validator.w3.org>

Flexbox - это общее название для модуля Flexible Box Layout, который имеется в CSS3. Данный модуль определяет особый режим компоновки/верстки пользовательского интерфейса, который называется flex layout.

В макете сайта предполагается размещение короткого видео на главной странице сайта. Для воспроизведения видео в HTML5 используется элемент **video**. Чтобы настроить данный элемент, мы можем использовать следующие его атрибуты:

- `src`: источник видео, это может быть какой-либо видеофайл
- `width`: ширина элемента
- `height`: высота элемента
- `controls`: добавляет элементы управления воспроизведением

- `autoplay`: устанавливает автовоспроизведение
- `loop`: задает повторение видео
- `muted`: отключает звук по умолчанию

Атрибут `poster` позволяет установить изображение, которое будет отображаться до запуска видео. Этому атрибуту в качестве значения передается путь к изображению

Главной проблемой при использовании элемента `video` является поддержка различными веб-браузерами определенных форматов. С помощью вложенных элементов `source` можно задать несколько источников видео, один из которых будет использоваться.

```
<video class="cafe__video" autoplay muted loop
poster="@/assets/img/home/video_poster_big.png">
<source src="@/assets/img/home/video.mp4" type="video/mp4">
<source src="@/assets/img/home/video.webm" type="video/webm">
<source src="@/assets/img/home/video.avi" type="video/avi">
Ваш браузер не поддерживает встроенные видео
</video>
```

## 4.2 Стилизация CSS

Любой html-документ, сколько бы он элементов не содержал, будет по сути "мертвым" без использования стилей. Стили или лучше сказать каскадные таблицы стилей (Cascading Style Sheets) или попросту CSS определяют представление документа, его внешний вид.

Стиль в CSS представляет правило, которое указывает веб-браузеру, как надо форматировать элемент. Форматирование может включать установку цвета фона элемента, установку цвета и типа шрифта и так далее.

Определение стиля состоит из двух частей: **селектор**, который указывает на элемент, и **блок объявления стиля** - набор команд, которые устанавливают правила форматирования. Например:

```
div{
  background-color:red;
  width: 100px;
  height: 60px;
}
```

Многие элементы html позволяют устанавливать стили отображения с помощью атрибутов. Однако подобного подхода следует избегать и вместо встроенных атрибутов следует применять стили CSS. Важно четко понимать, что разметка HTML должна предоставлять только структуру html-документа, а весь его внешний вид, стилизацию должны определять стили CSS.

В процессе написания стилей CSS могут возникать вопросы, а правильно ли так определять стили, корректны ли они. И в этом случае мы можем воспользоваться валидатором css, который доступен по адресу <http://jigsaw.w3.org/css-validator/>.

В дополнение к селекторам тегов, классов и идентификаторов нам доступны селекторы псевдоклассов, которые несут дополнительные возможности по выбору нужных элементов.

Список некоторых псевдоклассов:

- **:link**: применяется к ссылкам и представляет ссылку в обычном состоянии, по которой еще не совершен переход
- **:visited**: применяется к ссылкам и представляет ссылку, по которой пользователь уже переходил
- **:active**: применяется к ссылкам и представляет ссылку в тот момент, когда пользователь осуществляет по ней переход
- **:hover**: представляет элемент, на который пользователь навел указатель мыши. Применяется преимущественно к ссылкам, однако может также применяться и к другим элементам, например, к параграфам
- **:focus**: представляет элемент, который получает фокус, то есть когда пользователь нажимает клавишу табуляции или нажимает кнопкой мыши на поле ввода (например, текстовое поле)
- **:not**: позволяет исключить элементы из списка элементов, к которым применяется стиль

Псевдоэлементы обладают рядом дополнительных возможностей по выбору элементов веб-страницы и похожи на псевдоклассы. Список доступных псевдоэлементов:

- **::first-letter**: позволяет выбрать первую букву из текста
- **::first-line**: стилизует первую строку текста
- **::before**: добавляет сообщение до определенного элемента
- **::after**: добавляет сообщение после определенного элемента
- **::selection**: выбирает выбранные пользователем элементы

## 4.2.1 Подключение шрифтов к проекту

Шрифты делятся на несколько типов:

- шрифты с засечками (serif);
- шрифты без засечек (sans-serif);
- моноширинные (monospace).

Основные параметры шрифта:

- font-family — указывает, к какому семейству относится шрифт;
- font-weight — указывает вес шрифта;
- font-style — указывает начертание шрифта.

Чтобы подключить шрифт, нужно в CSS использовать правило `@font-face`. Правило `@font-face src` позволяет задать название локального шрифта, т.е. если у пользователя на компьютере уже установлен нужный шрифт, то будет использоваться именно он, при этом существенно увеличится скорость загрузки и отрисовки страницы.

Сегодня используются четыре формата шрифтов:

- TTF/OTF – работают в большинстве браузеров, кроме IE
- EOT – создан Microsoft, представляет сжатую копию шрифта TTF, поддерживается только в IE
- WOFF – формат представляет собой сжатый шрифт в формате TTF/OTF
- WOFF2 – имеет улучшенное сжатие, по сравнению с первой версией

Как видно нет единого формата, который поддерживается всеми браузерами, поэтому нужно делать подключение нескольких файлов, браузер сам выберет подходящий формат.

Рекомендуется подключать файлы шрифтов по приоритету:

1. WOFF2 для современных браузеров
2. WOFF для браузеров, которые не поддерживают WOFF2
3. TTF для устаревших браузеров
4. EOT для поддержки IE

Файлы шрифтов хранятся в папке `assets/fonts` и подключаются в отдельном файле стилей `fonts.scss`:

```
@font-face {
  font-family: 'Blackberry';
  src: local('Blackberry'),
    url('~@/assets/fonts/blackberry/blackberry.woff2') format('woff2'),
    url('~@/assets/fonts/blackberry/blackberry.woff') format('woff'),
    url('~@/assets/fonts/blackberry/blackberry.ttf') format('ttf'),
    url('~@/assets/fonts/blackberry/blackberry.eot');

  font-weight: normal;
  font-style: normal;
}

@font-face {
  font-family: 'Jost';
  src: local('Jost'),
    url('~@/assets/fonts/jost/Jost-Regular.woff2') format('woff2'),
    url('~@/assets/fonts/jost/Jost-Regular.woff') format('woff'),
    url('~@/assets/fonts/jost/Jost-Regular.ttf') format('ttf'),
    url('~@/assets/fonts/jost/Jost-Regular.eot');

  font-weight: normal; // 400
  font-style: normal;
}

@font-face {
  font-family: 'Jost';
  src: local('Jost'),
    url('~@/assets/fonts/jost/Jost-SemiBold.woff2') format('woff2'),
    url('~@/assets/fonts/jost/Jost-SemiBold.woff') format('woff'),
    url('~@/assets/fonts/jost/Jost-SemiBold.ttf') format('ttf'),
    url('~@/assets/fonts/jost/Jost-SemiBold.eot');
```

```

font-weight: 600;
font-style: normal;
}

@font-face {
  font-family: 'Jost';
  src: local('Jost'),
  url('~@/assets/fonts/jost/Jost-Bold.woff2') format('woff2'),
  url('~@/assets/fonts/jost/Jost-Bold.woff') format('woff'),
  url('~@/assets/fonts/jost/Jost-Bold.ttf') format('ttf'),
  url('~@/assets/fonts/jost/Jost-Bold.eot');

  font-weight: bold; // 700
  font-style: normal;
}

```

#### 4.2.2 Анимация и переходы

Переход (transition) представляет анимацию от одного стиля к другому в течение определенного периода времени. Для создания перехода необходимы прежде всего два набора свойств CSS: начальный стиль, который будет иметь элемент в начале перехода, и конечный стиль - результат перехода. Анимация предоставляет большие возможности за изменением стиля элемента. При переходе у нас есть набор свойств с начальными значениями, которые имеет элемент до начала перехода, и конечными значениями, которые устанавливают после завершения перехода. Тогда как при анимации мы можем иметь не только два набора значений - начальные и конечные, но и множество промежуточных наборов значений. Анимация опирается на последовательную смену ключевых кадров (keyframes). Каждый ключевой кадр определяет один набор значений для анимируемых свойств. И последовательная смена таких ключевых кадров фактически будет представлять анимацию. По сути, переходы применяют ту же модель - так же неявно определяются два ключевых кадра - начальный и конечный, а сам переход представляет переход от начального к конечному ключевому кадру. Единственное отличие анимации в данном случае состоит в том, что для анимации можно определить множество промежуточных ключевых кадров.

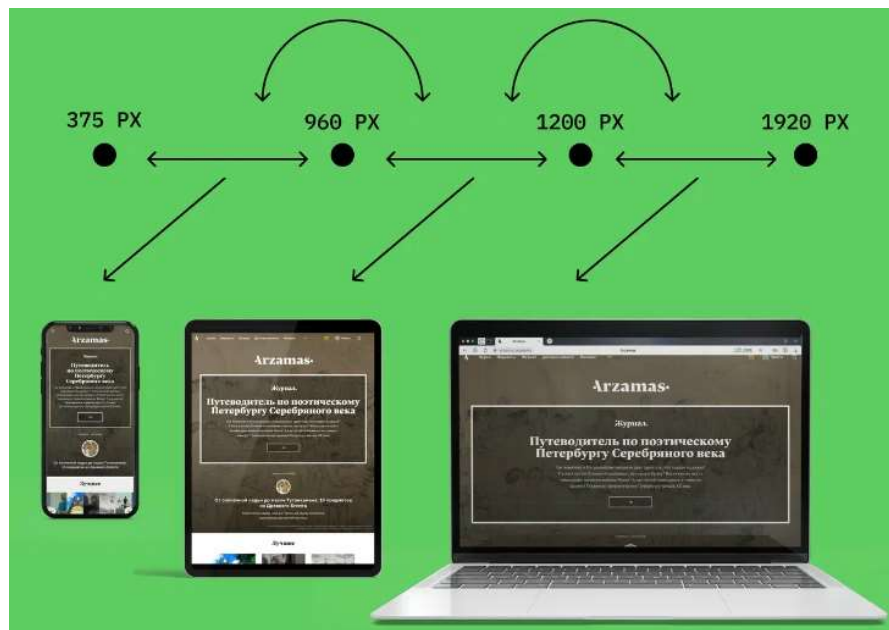
### 4.2.3 Адаптивная верстка



Рис.4.1 Визуализация сайта на различных устройствах

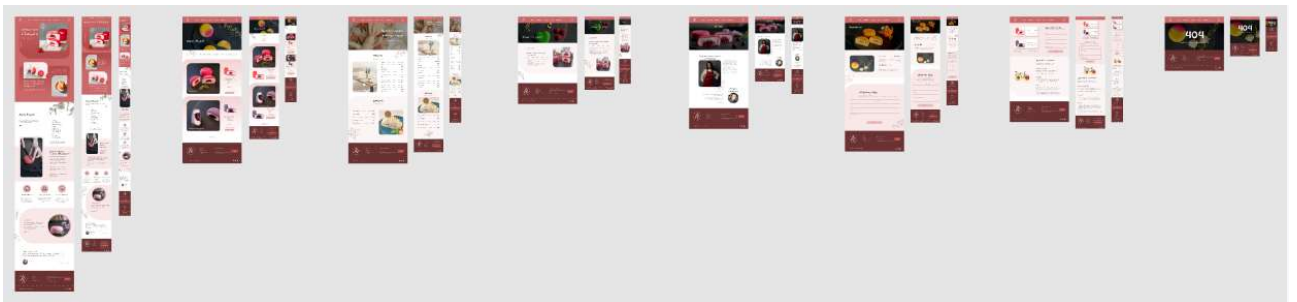
Сейчас все большее распространение находят различные гаджеты - смартфоны, планшеты, "умные часы" и другие устройства, которые позволяют выходить в интернет, просматривать содержимое сайтов. По некоторым оценкам уже чуть ли не половина интернет-трафика генерируется подобными гаджетами, разрешение экрана которых отличается от разрешения экранов стандартных компьютеров. Подобное распространение гаджетов несет новые возможности по развитию веб-сайтов, привлечения новых посетителей, продвижению информационных услуг и т.д. Но вместе с тем появляются и новые проблемы. Главная проблема заключается в том, что стандартная веб-страница будет по-разному выглядеть для разных устройств с разным разрешением экрана. И для решения проблемы совместимости веб-страниц с самыми различными разрешениями самых различных устройств возникла концепция адаптивного дизайна. Ее суть заключается в том, чтобы должным образом масштабировать элементы веб-страницы в зависимости от ширины экрана.





*Рис 4.2 Разрешения экранов для адаптивной вёрстки*

Для верстки адаптивного сайта мной были разработаны макеты страниц сайта в трех размерах (разрешениях экрана):



*Рис 4.3 Макеты страниц проекта для адаптивной вёрстки*

Другим важным элементом в построении адаптивного дизайна являются правила Media Query, которые позволяют определить стиль в зависимости от размеров браузера пользователя.

## Применяемые функции в CSS3 Media Query:

- **aspect-ratio**: отношение ширины к высоте области отображения (браузера)
- **device-aspect-ratio**: отношение ширины к высоте экрана устройства
- **max-width/min-width** и **max-height/min-height**: максимальная и минимальная ширина и высота области отображения (браузера)
- **max-device-width/min-device-width** и **max-device-height/min-device-height**: максимальная и минимальная ширина и высота экрана мобильного устройства
- **orientation**: ориентация (портретная или альбомная)

```
// Desktop view - middle point
@media (max-width: 1644px) {}
// Tablet view
@media (max-width: 1200px) {}
// Mobile view
@media (max-width: 960px) {}
// Mobile - middle point
@media (max-width: 768px) {}
```

## 4.3 Методология БЭМ

БЭМ (Блок, Элемент, Модификатор) — компонентный подход к веб-разработке. В его основе лежит принцип разделения интерфейса на независимые блоки. Он позволяет легко и быстро разрабатывать интерфейсы любой сложности и повторно использовать существующий код, избегая «Copy-Paste».

### Блок

Функционально независимый компонент страницы, который может быть повторно использован. В HTML блоки представлены атрибутом `class`.

Особенности:

- Название блока характеризует смысл («что это?» — «меню»: `menu`, «кнопка»: `button`), а не состояние («какой, как выглядит?» — «красный»: `red`, «большой»: `big`).
- Блок не должен влиять на свое окружение, т. е. блоку не следует задавать внешнюю геометрию (в виде отступов, границ, влияющих на размеры) и позиционирование.
- В CSS по БЭМ также не рекомендуется использовать селекторы по тегам или `id`

### Элемент

Составная часть блока, которая не может использоваться в отрыве от него.

Особенности:

- Название элемента характеризует смысл («что это?» — «пункт»: `item`, «текст»: `text`), а не состояние («какой, как выглядит?» — «красный»: `red`, «большой»: `big`).
- Структура полного имени элемента соответствует схеме: имя-блока\_\_имя-элемента. Имя элемента отделяется от имени блока двумя подчеркиваниями (`__`).

## Модификатор

Сущность, определяющая внешний вид, состояние или поведение блока либо элемента.

Особенности:

- Название модификатора характеризует внешний вид («какой размер?», «какая тема?» и т. п. — «размер»: `size_s`, «тема»: `theme_islands`), состояние («чем отличается от прочих?» — «отключен»: `disabled`, «фокусированный»: `focused`) и поведение («как ведет себя?», «как взаимодействует с пользователем?» — «направление»: `directions_left-top`).
- Имя модификатора отделяется от имени блока или элемента одним подчеркиванием (`_`).

## 4.4 Препроцессор SCSS (Sass)

SCSS (Sassy CSS) – это препроцессор CSS, который добавляет некоторые дополнительные функции и возможности к стандартному CSS. SCSS позволяет использовать переменные, вложенные правила, миксины, условия и другие функции, что делает процесс написания стилей более эффективным и удобным.

### 1. Переменные в SCSS:

Переменные позволяют задать значение, которое можно использовать многократно в различных частях стилей. В своем проекте я создала специальный файл для хранения переменных vars.scss с таким содержимым:

```
$width: 1644px;  
$pink-50: #fbf5f5;  
$pink-100: #f8e8e8;  
$pink-200: #f2d3d3;  
$pink-300: #e9b8b8;  
$pink-400: #db8e8e;  
$pink-500: #ca6969;  
$pink-600: #B44E4E;  
$pink-700: #973e3e;  
$pink-800: #7e3636;  
$pink-900: #6a3232;  
$pink-950: #381717;
```

При помощи этих переменных мне легко использовать цвета, которые использованы в макете Figma:

Colors	...		
All variables	23	Name	Value
Oyster Pink		50	FBF5F5
Mercury		100	F8E8E8
		200	F2D3D3
		300	E9B8B8
		400	DB8E8E
		500	CA6969
		600	B44E4E
		700	973E3E
		800	7E3636
		900	6A3232
		950	381717

Рис.4.4 Переменные для цветов в макете Figma

## 2. Вложенные правила:

SCSS позволяет писать вложенные стили, что упрощает и улучшает читаемость кода. Это упрощает стилизацию элементов с классами, именованными согласно методологии БЭМ. Например:

```
.quantity {  
  
  &__counter {  
    display: flex;  
    gap: 14px;  
  }  
  
  &__input {  
    width: 100px;  
    height: 50px;  
    background-color: transparent;  
    border: 2px solid $pink-900;  
    border-radius: 10px;  
    @include header-6($pink-950);  
    text-align: center;  
  
    &::placeholder {  
      color: $pink-800;  
    }  
  }  
  
  &__btn {  
    width: 80px;  
    height: 50px;  
    @include header-5($pink-600);  
  }  
}
```

### 3. Миксины:

Миксины в SCSS – это фрагменты стилей, которые можно повторно использовать в коде. Они могут содержать любые стили и аргументы для настройки. Например, миксины для стилизации шрифтов:

```
@mixin header-1($color) {  
  color: $color;  
  font-family: 'blackberry', serif;  
  font-size: 108px;  
  line-height: 111%;  
}  
  
@mixin nav-item($color) {  
  color: $color;  
  font-weight: 600;  
  font-size: 26px;  
  text-transform: uppercase;  
}
```

SCSS компилируется в стандартный CSS с помощью специальных инструментов, таких как Sass или Node-sass, и затем подключается к HTML-странице.

Для организации структуры кода в SCSS и удобного подключения различных файлов стилей можно использовать возможность импорта файлов в один общий файл SCSS. Например, это позволяет легко передавать данные о базовой стилизации, о переменных и миксинах в любой файл стилей с расширением \*.scss:

```
@import "vars";  
@import "base";  
@import "fonts";
```

Кроме того, в SCSS можно использовать переменные для указания путей к файлам. Например, такой подход позволит более гибко управлять путями к импортируемым файлам и избежать дублирования:

```
$components-path: 'components/';  
@import '#{ $components-path }header';  
@import '#{ $components-path }footer';
```

Использование SCSS позволяет значительно упростить и ускорить процесс написания стилей, делая их более модульными, гибкими и поддерживаемыми.

## 5. Система контроля версий Git

Git — это распределенная система управления версиями, которая используется для отслеживания изменений в коде и совместной работы над проектами. Вот основные концепции и команды Git:

### 1. Репозиторий (repository):

Репозиторий — это место, где хранится весь исторический код проекта. Репозиторий может быть локальным (на вашем компьютере) или удаленным (например, на GitHub).

### 2. Клонирование репозитория (clone):

Чтобы начать работу с проектом, вы можете клонировать удаленный репозиторий на свой компьютер с помощью команды `git clone <url>`.

### 3. Коммиты (commits):

Коммит — это сохраненные изменения в коде. Чтобы создать коммит, используйте команду `git commit -m "Описание изменений"`.

### 4. Ветки (branches):

Ветка — это отдельная линия разработки, в которой вы можете внести изменения без влияния на основную версию кода. С помощью команды `git branch <branch-name>` создается новая ветка, и с помощью команды `git checkout <branch-name>` вы переключаетесь на нее.

### 5. Слияние коммитов (merge):

Чтобы внести изменения из одной ветки в другую, используйте команду `git merge <branch-name>`.

### 6. Удаленные репозитории (remote repositories):

Удаленные репозитории позволяют вам работать над проектом с другими разработчиками. Для добавления удаленного репозитория используйте команду `git remote add <name> <url>`.

### 7. Загрузка и скачивание изменений (push и pull):

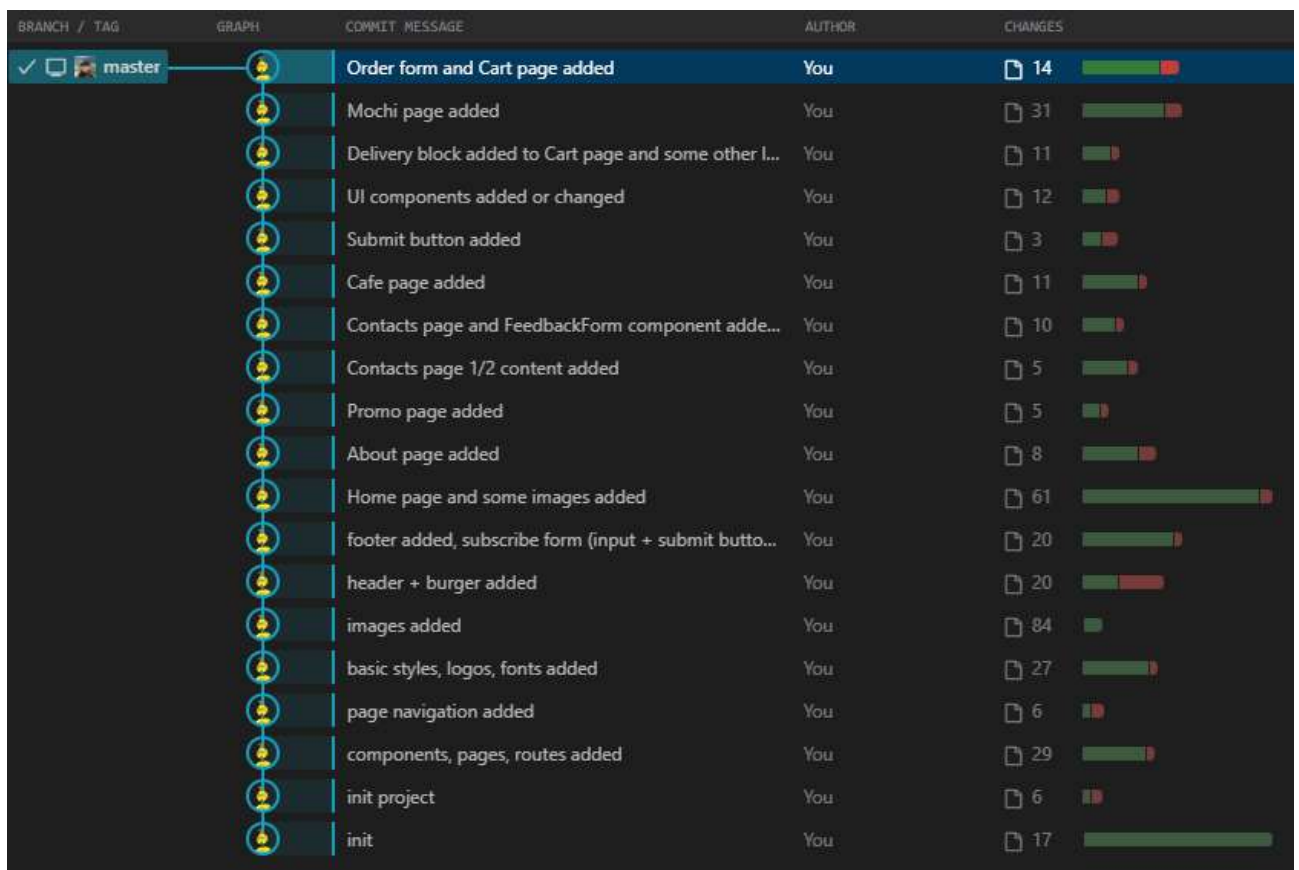


Чтобы отправить свои коммиты в удаленный репозиторий, используйте команду `git push`. Для загрузки изменений из удаленного репозитория на свой компьютер используйте команду `git pull`.

#### 8. Журнал коммитов (log):

С помощью команды `git log` вы можете посмотреть историю коммитов в репозитории.

Это только некоторые из основных команд Git. Система Git предоставляет множество возможностей для управления версиями кода, ветвлением и совместной работой над проектами. Изучение Git может значительно улучшить процесс разработки и совместной работы над проектами.



BRANCH / TAG	GRAPH	COMMIT MESSAGE	AUTHOR	CHANGES
✓ master		Order form and Cart page added	You	14
		Mochi page added	You	31
		Delivery block added to Cart page and some other I...	You	11
		UI components added or changed	You	12
		Submit button added	You	3
		Cafe page added	You	11
		Contacts page and FeedbackForm component adde...	You	10
		Contacts page 1/2 content added	You	5
		Promo page added	You	5
		About page added	You	8
		Home page and some images added	You	61
		footer added, subscribe form (input + submit butto...	You	20
		header + burger added	You	20
		images added	You	84
		basic styles, logos, fonts added	You	27
		page navigation added	You	6
		components, pages, routes added	You	29
		init project	You	6
		init	You	17

Рис. 5.1 Git log проекта

## 6. Разработка проекта на Vue.js

### 6.1. Структура проекта

Структура проекта веб-приложения на Vue может выглядеть следующим образом:

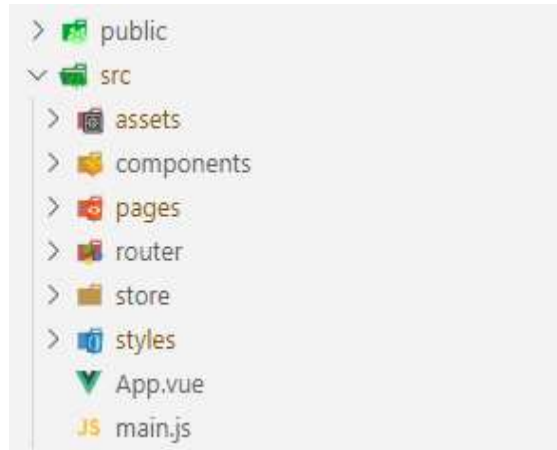


Рис.6.1 Структура проекта Vue.js

1. **public** - папка с публичными ресурсами, здесь содержится основной файл `index.html`, он является отправной точкой для веб-приложения

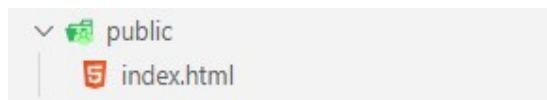


Рис 6.2. Папка *public*

2. **src** - главная папка проекта, содержащая остальные файлы и папки:
  - **assets** - папка с ресурсами, такими как шрифты и изображения (разложены по папкам в соответствии со страницей сайта и общими изображениями для всего сайта)

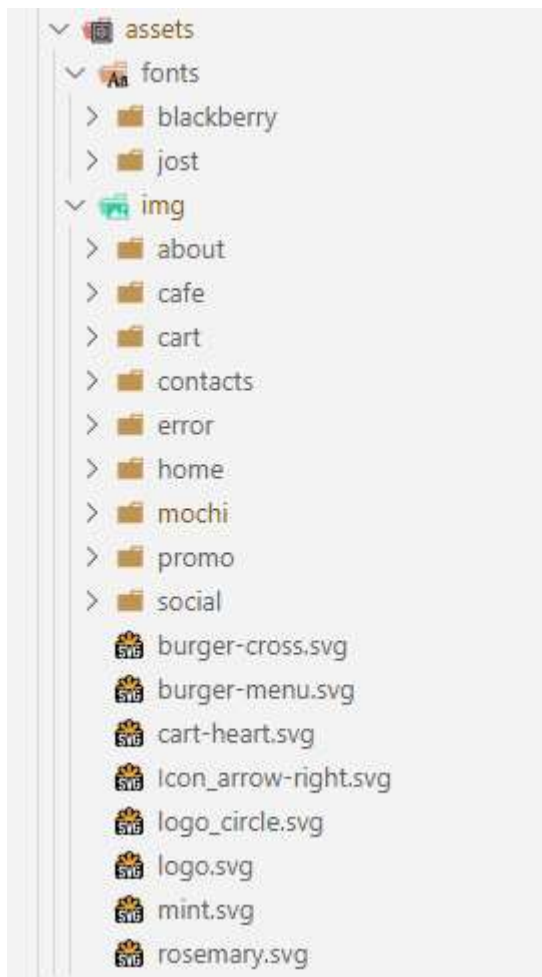


Рис 6.3. Папка assets

- **components** - папка для Vue-компонентов, содержит компоненты Vue, которые могут быть повторно использованы в представлениях или других компонентах. Также здесь расположен файл `index.js`, в который импортированы все компоненты приложения.

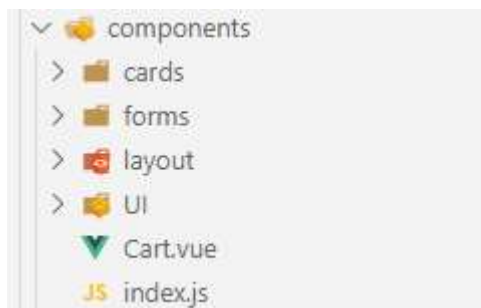
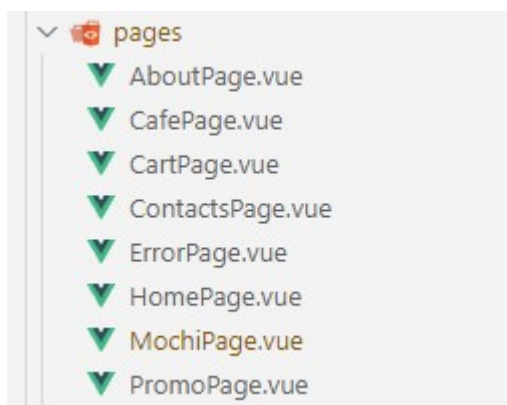


Рис 6.4. Папка components

- **pages** - папка с Vue-компонентами представлений (страниц), которые используются маршрутизатором для отображения различных страниц в приложении



*Рис 6.5. Папка pages*

- **router** - папка с файлом маршрутизации (index.js), где определены маршруты и соответствующие компоненты для них



*Рис 6.6. Папка router*

- **store** - папка для хранилища Vuex, где определены состояние, мутации и действия приложения



*Рис 6.7. Папка store*

- **styles** - папка с файлами стилей CSS (в своем проекте я использовала препроцессор SCSS (Sass). Файлы стилей разделены по страницам и компонентам приложения и собираются в общий файл стилей styles.scss.

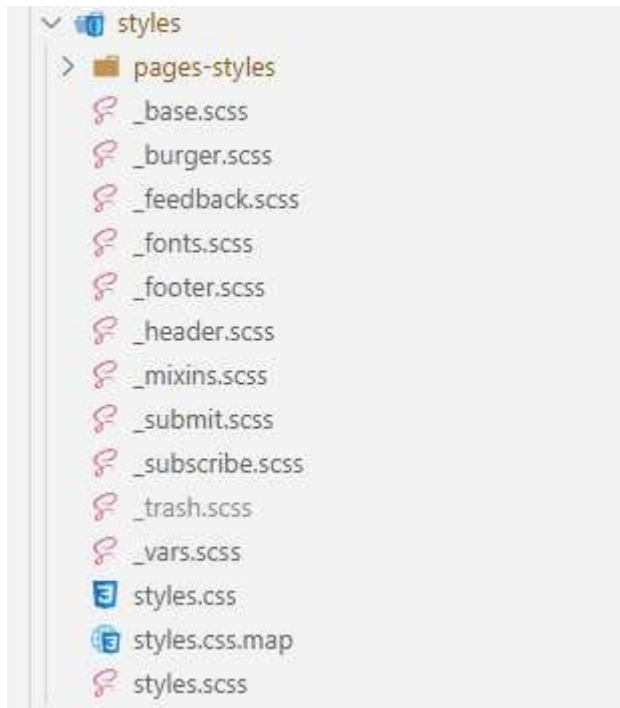


Рис 6.8. Папка styles

- **App.vue** - главный компонент приложения, содержащий остальные компоненты приложения и маршрутизацию
- **main.js** - точка входа приложения, где инициализируется Vue-приложение и подключаются необходимые плагины и компоненты

## 6.2. Настройка и установка окружения разработки на Vue

Для начала разработки веб-сайта на платформе Vue.js необходимо установить и настроить окружение разработки. В этом разделе приведены основные шаги по установке необходимых инструментов.

1. **Установка Node.js и npm:** Vue.js требует наличие Node.js и npm (Node Package Manager) для управления зависимостями проекта. Node.js можно скачать с официального сайта (<https://nodejs.org/>) и установить с помощью установщика.
2. **Установка Vue CLI:** Vue CLI - инструмент командной строки, который позволяет создавать и управлять проектами на Vue.js. Установить Vue CLI с помощью npm следующей командой:

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

3. **Создание нового проекта:** после установки Vue CLI нужно создать новый проект с помощью команды:

```
vue create otzayki_zevina_gb
```

где `otzayki_zevina_gb` - название проекта

3. **Запуск локального сервера:** после создания проекта можно запустить локальный сервер для разработки с помощью команды:

```
npm run serve
```

Это позволяет видеть изменения в реальном времени и тестировать функционал веб-сайта.

4. **Установка дополнительных пакетов:** требуется установить дополнительные пакеты, такие как Vue Router для маршрутизации или Vuex для управления состоянием приложения во время настройки нового проекта при его создании или впоследствии отдельными командами:

```
npm install vue-router
```

```
npm install vuex
```

## 6.3 Реализация компонентов и страниц сайта

Компоненты в Vue.js имеют следующую структуру:

```
<template>
  <!-- HTML-шаблон компонента -->
</template>

<script>
  export default {
    // Данные и логика компонента
  }
</script>

<style>
  / * CSS-стили для компонента * /
</style>
```

*Рис.6.9 Структура компонента Vue.js*

Реализация основных компонентов веб-приложения кондитерской:

### I. Компоненты форм

#### 1. SubscribeForm.vue

Данный компонент представляет собой форму подписки на рассылку новостей и акций. Он содержит следующие элементы:

- заголовок формы "Не пропускайте важное!"
- поле ввода электронной почты для подписки.
- кнопка "Отправить", которая становится активной только после ввода электронной почты.
- текст "получать информацию об акциях и событиях", который объясняет, что получит пользователь после подписки.

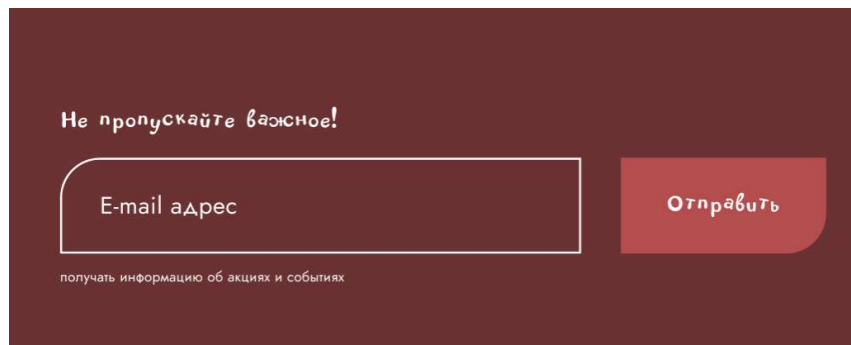
В JavaScript-части кода определены данные компонента, методы жизненного цикла и обработчики событий. В частности, метод `sendEmail()` отвечает за отправку данных формы на сервер и очистку полей формы после отправки.



В CSS-части кода используются стили, определенные в файле `subscribe.scss`, для стилизации формы подписки.

Использует компоненты `MyInput.vue` и `SubmitButton.vue`.

Таким образом, данный компонент позволяет пользователям подписаться на рассылку новостей и акций, отправляя свои электронные адреса на сервер.



*Рис.6.10 Форма подписки на рассылку*

## 2. `FeedbackForm.vue`

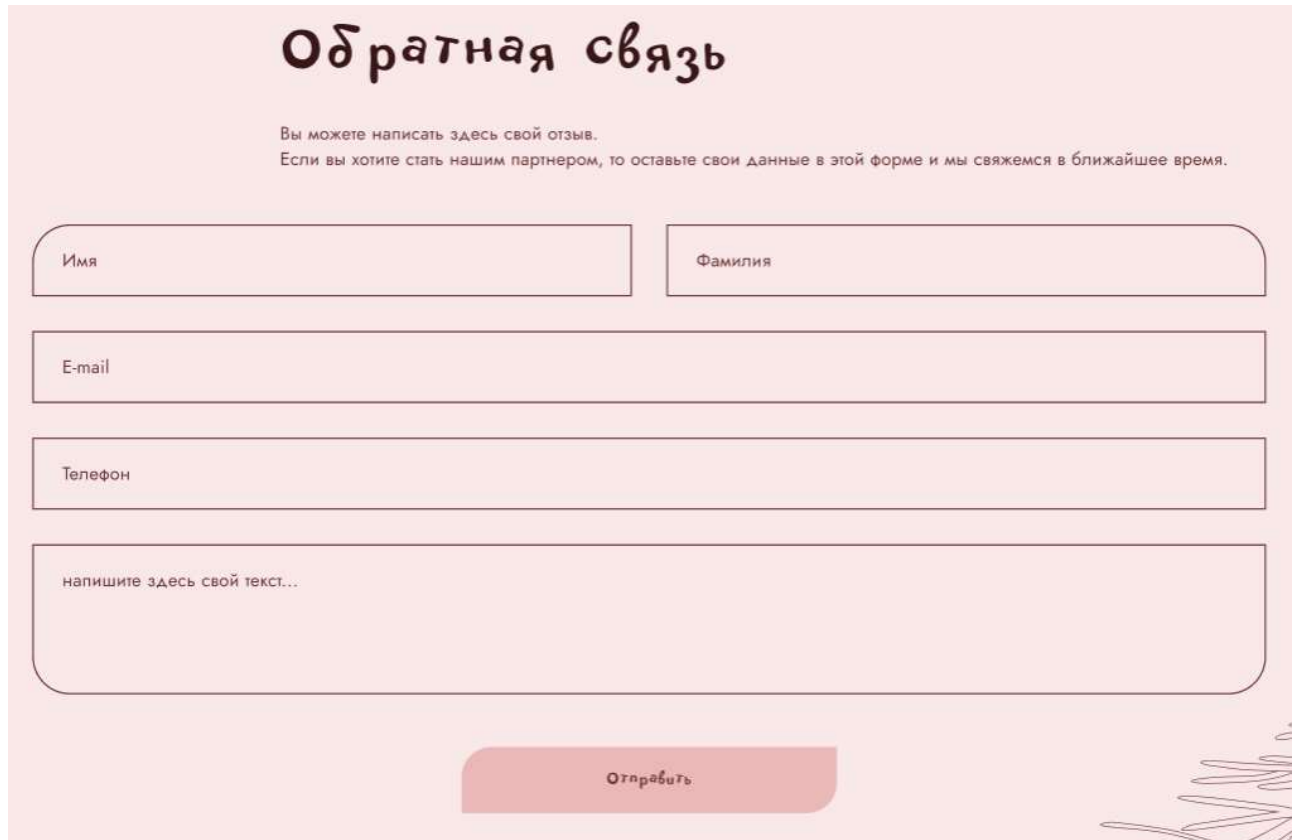
Данный компонент представляет собой форму обратной связи, которая позволяет пользователям оставлять отзывы, комментарии или связываться с компанией по вопросам партнерства. Он содержит следующие элементы:

- заголовок формы "Обратная связь".
- текстовое описание, объясняющее, что пользователь может написать в форме.
- поля ввода для имени, фамилии, электронной почты, телефона и сообщения.
- кнопка "Отправить", которая становится активной только после заполнения всех обязательных полей.

В JavaScript-части кода определены данные компонента, методы жизненного цикла и обработчики событий. В частности, метод `submitForm()` отвечает за отправку данных формы на сервер и очистку полей формы после отправки.

В CSS-части кода используются стили, определенные в файле `feedback.scss`, для стилизации формы обратной связи.

Использует компоненты `MyInput.vue` и `SubmitButton.vue`.



The image shows a web form titled "Обратная связь" (Feedback) on a light pink background. Below the title, there is a short instruction in Russian: "Вы можете написать здесь свой отзыв. Если вы хотите стать нашим партнером, то оставьте свои данные в этой форме и мы свяжемся в ближайшее время." (You can write your review here. If you want to become our partner, leave your data in this form and we will contact you in the near future). The form consists of several input fields: "Имя" (Name) and "Фамилия" (Surname) are side-by-side; "E-mail" and "Телефон" (Phone) are stacked vertically below them. There is a large text area for "напишите здесь свой текст..." (write your text here...). At the bottom center is a red "Отправить" (Send) button. The bottom right corner has a faint line-art illustration of reeds.

*Рис.6.11 Форма обратной связи*

### 3. OrderForm.vue

Данный компонент представляет собой форму заказа, которая позволяет пользователям оформить заказ, указав свои данные. Он содержит следующие элементы:

- заголовок формы "Оформление заказа".
- текстовое описание, объясняющее, что пользователь может написать в форме.
- поля ввода для имени, электронной почты и адреса доставки.
- кнопка "Оформить заказ", которая отправляет данные формы на сервер.

В JavaScript-части кода определены данные компонента и метод `submitForm()`, который отвечает за отправку данных формы на сервер и очистку полей формы после отправки.

В CSS-части кода используются стили, которые определяются в файле `order-form.scss`, для стилизации формы заказа.

Использует компоненты `MyInput.vue` и `SubmitButton.vue`.



*Рис.6.12 Форма оформления заказа*

## II. Компоненты макета

### 1. `MyHeader.vue`

Данный компонент представляет собой верхний колонтитул веб-сайта. Он содержит логотип, навигационное меню и ссылку на корзину.

Компонент использует `Vue Router` для управления переходом по ссылкам и компонент `BurgerMenu.vue` для отображения навигационного меню при просмотре сайта с мобильных устройств.



*Рис.6.13 Хедер (шапка) сайта*

## 2. MyFooter.vue

Компонент представляет собой нижний колонтитул веб-сайта. Он содержит информацию о компании, контактные данные и ссылки на социальные сети. В компоненте также есть форма подписки на рассылку новостей.

Верхняя часть футера содержит логотип компании, контактные данные и адрес офиса. В нижней части футера расположены информация о правах на контент, иконки социальных сетей.

Компонент использует Vue Router для управления переходом по ссылкам и компонент `SubscribeForm.vue` для отображения формы подписки.

В целом, компонент `MyFooter.vue` предоставляет посетителям сайта информацию о компании и контактные данные, а также предлагает возможность подписаться на рассылку новостей и связаться с компанией через социальные сети.



Рис.6.14 Футер (подвал) сайта

### III. Компоненты товаров

#### 1. CardItem.vue и CardList.vue

Этот компонент представляет карточку товара. Внутри компонента есть изображение товара, название товара, дополнительное изображение, описание, цена и кнопка "Добавить в корзину". Используются данные из объекта `item`, такие как изображения, название, описание и цена.

В скрипте компонента импортируются дочерние компоненты `QuantityCounter` и `SubmitButton`. Пропсы содержат объект `item`, который является обязательным. Есть вычисляемое свойство `getCartItemQuantity`, которое получает значение из геттера хранилища. Есть метод `addToCart`, который добавляет товар в корзину. Используется SCSS для подключения стилей из файла `carditem.scss`.

Этот компонент используется для отображения информации о товаре с возможностью добавления его в корзину с помощью кнопки.



Рис.6.15 Карточка товара для каталога

## 2. CardItemSmall.vue и CardListSmall.vue

Этот компонент представляет небольшую карточку товара для отображения списка товаров в корзине. В компоненте есть изображение товара, кнопка "Удалить из корзины", название товара, цена, счетчик количества товара и возможность обновления этого количества.

Используются данные из объекта `item`, такие как изображение, название и цена. В скрипт секции компонента импортируется дочерний компонент `QuantityCounter`. Пропс содержит объект `item`, который является обязательным. В `data` компонента инициализируется переменная `quantity`, которая устанавливается из геттера хранилища. Также есть вычисляемое свойство `getCartItemQuantity`, которое также получает значение из геттера хранилища. Есть методы `removeFromCart`, который удаляет товар из корзины, и `updateQuantity`, который обновляет количество товара. Используется SCSS для подключения стилей из файла `cardsmall.scss`.

Этот компонент позволяет отобразить информацию о товаре в небольшой карточке, а также управлять количеством товара в корзине.

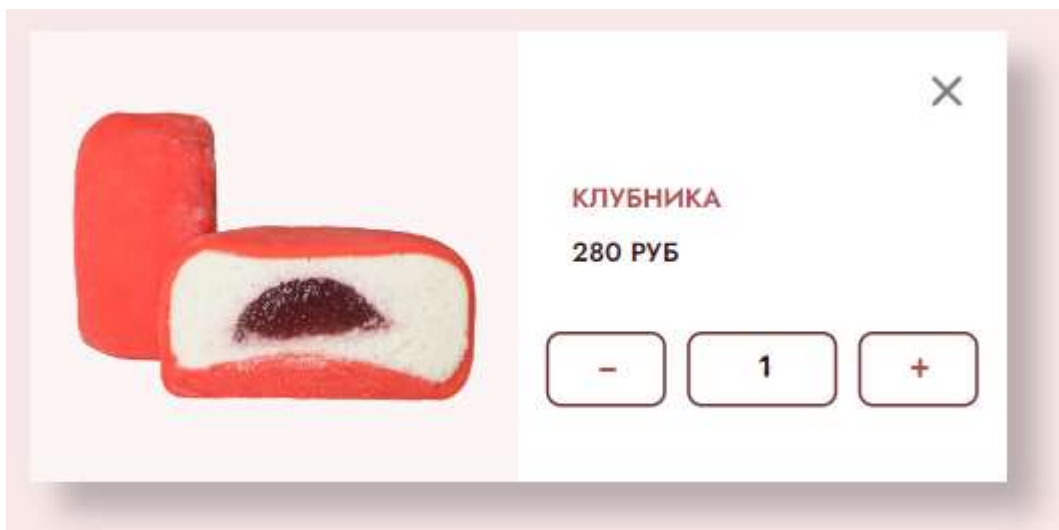


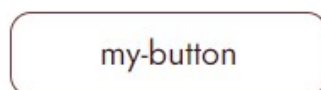
Рис.6.16 Карточка товара для корзины

## IV. Компоненты дизайна UI

### 1. MyButton.vue

Данный компонент представляет собой кнопку с названием "my-button". Он состоит из одного div-элемента с классом "my-button". Внутри этого div-элемента находится слот. Слот в компоненте позволяет размещать внутри него любой HTML-код, который будет отображаться на месте этого компонента. Это позволяет гибко настраивать внешний вид кнопки в зависимости от потребностей.

Стили для компонента задаются в файле mybutton.scss, который импортируется в компонент.

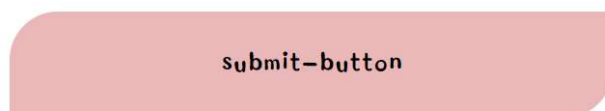


*Рис.6.17 Кнопка my-button*

### 2. SubmitButton.vue

Данный компонент представляет собой кнопку с названием "submit-button". Он состоит из одного div-элемента с классом "submit-button". Внутри этого div-элемента находится слот. Слот в компоненте позволяет размещать внутри него любой HTML-код, который будет отображаться на месте этого компонента. Это позволяет гибко настраивать внешний вид кнопки в зависимости от потребностей.

Стили для компонента задаются в файле submit.scss, который импортируется в компонент.



*Рис.6.18 Кнопка my-button*

### 3. BurgerMenu.vue

Данный компонент представляет собой бургер-меню, которое используется для навигации по сайту. Он содержит кнопку, которая при нажатии меняет изображение с иконки меню на иконку крестика, и список с ссылками на различные разделы сайта.

Компонент использует Vue Router для управления переходом по ссылкам.



Рис.6.19 Навигационное меню (мобильная версия) в свернутом виде

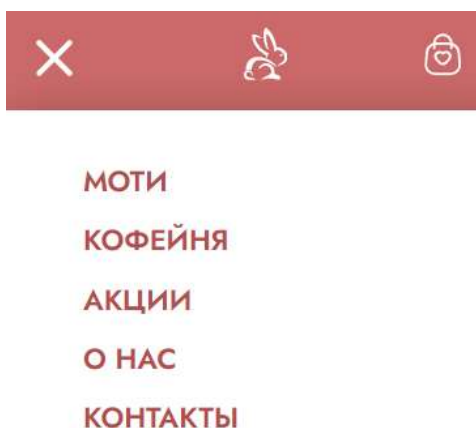


Рис.6.20 Навигационное меню (мобильная версия) в развернутом виде

### 4. MuInput.vue

Данный компонент представляет собой элемент ввода, который может быть использован для ввода текста или чисел. Он принимает значение модели через свойство 'modelValue' и тип ввода через свойство 'type'. При изменении значения ввода, компонент вызывает метод 'updateInput', который обновляет значение модели, передавая его через событие 'update:modelValue'. Компонент использует Vue.js для управления состоянием и событиями.

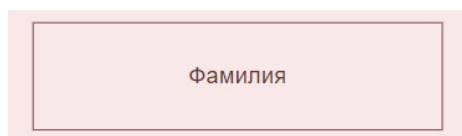


Рис.6.21 Компонент ввода (пример использования)



## 5. QuantityCounter.vue

Данный компонент представляет собой простой счетчик, который позволяет пользователю увеличивать или уменьшать количество чего-либо. Компонент состоит из трех элементов: двух кнопок с текстом "-" и "+", а также одного поля ввода типа "number". Кнопки "-" и "+" связаны с методами `decreaseQuantity` и `increaseQuantity` соответственно. При нажатии на кнопку "-" количество уменьшается на единицу, если оно больше нуля. При нажатии на кнопку "+" количество увеличивается на единицу. Поле ввода типа "number" имеет модель `v-model.number`, которая связана с переменной `quantity` в данных компонента. Это означает, что значение поля ввода будет автоматически обновляться при изменении значения переменной `quantity` и наоборот.

Стили для компонента задаются в файле `quantity.scss`, который импортируется в компонент.



*Рис.6.22 Компонент счетчика*

## V. Компонент корзины

### **Cart.vue**

Этот компонент представляет собой отображение корзины покупок. Он состоит из нескольких элементов:

1. Если корзина пуста (`cartItems.length === 0`), то отображается сообщение "Корзина пуста". В противном случае отображается список карточек товаров (`CardListSmall`) с товарами из объекта `cartItems`, и карточки могут обновить количество товаров.

2. Под списком карточек есть блок с кнопками управления корзиной. При нажатии на кнопку "Очистить корзину" вызывается метод `clearCart`, которым происходит очистка корзины. При нажатии на кнопку "в меню вкусов" происходит переход на страницу `/mochi`.

3. Ниже кнопок отображается блок с итоговой суммой покупки. Сначала вычисляется общая сумма `totalAmount`, умножив цену каждого товара на его количество. Это значение отображается на странице.

4. Основная логика компонента происходит в блоке скрипта. Ему нужны компоненты `CardListSmall` и `MyButton`, которые импортируются в начале секции. В разделе `computed` вычисляются `cartItems` (товары в корзине) и `totalAmount` (итоговая сумма покупки). Методы `clearCart` и `handleQuantityUpdated` отвечают за очистку корзины и обновление общей суммы после изменения количества товаров соответственно.

5. Используется SCSS, чтобы подключить стили из файла `cartcomponent.scss`, которые применяются к элементам компонента.

Этот компонент позволяет управлять содержимым корзины покупок, отображая список товаров, их количество, итоговую стоимость, а также позволяет очистить корзину и перейти на страницу с меню вкусов.



*Рис.6.23 Компонент корзины*

## VI. Компоненты страниц сайта

### 1. HomePage.vue



Рис.6.24 Главная страница

Данный компонент представляет собой веб-страницу, которая содержит несколько разделов.

Верхний раздел содержит изображения и текстовое описание десерта "моти".

В следующем разделе представлен список вкусов десерта с кнопкой для перехода на страницу с полным списком вкусов.

Третий раздел содержит видео и информацию о кофейне.

Далее идет раздел, который содержит ряд преимуществ компании-кондитерской и ее десертов.

Ниже расположен раздел с текущей акцией. Здесь находится текстовое описание акции и кнопка для перехода на страницу с подробностями акции.

В самом низу страницы представлен отзыв о продукции компании и кнопка для перехода на страницу с формой обратной связи с предложением оставить свой отзыв о компании, кофейне или продукции.

Стили для компонента задаются в файле `home.scss`, который импортируется в компонент.

## 2. MochiPage.vue



Рис.6.25 Страница каталога

Верхний раздел компонента содержит изображение десерта и название страницы.

Основной раздел содержит каталог вкусов десерта моти, представленный в виде карточек товара с изображением, подробным описанием и ценой. Здесь же реализована возможность добавить желаемой количество десертов в корзину.

Есть возможность фильтрации карточек по категориям вкусов.

Стили для компонента задаются в файле `mochi.scss`, который импортируется в компонент.

### 3. CafePage.vue



Данный компонент содержит информацию о напитках и десертах в кафе. Он состоит из двух основных блоков: "drinks" и "desserts".

Верхний раздел содержит изображение десерта и название страницы.

В блоке "drinks" находится информация о напитках. Он содержит заголовок "Напитки", подзаголовок и основной контент, который включает в себя изображение и информацию о напитках: название и цена.

В блоке "desserts" находится информация о десертах. Он содержит заголовок "Десерты", подзаголовок и основной контент, который включает в себя изображение и информацию о десертах: название и цена.

Стили для компонента задаются в файле cafe.scss, который импортируется в компонент.

Рис.6.26 Страница меню кофейни



## 4. PromoPage.vue

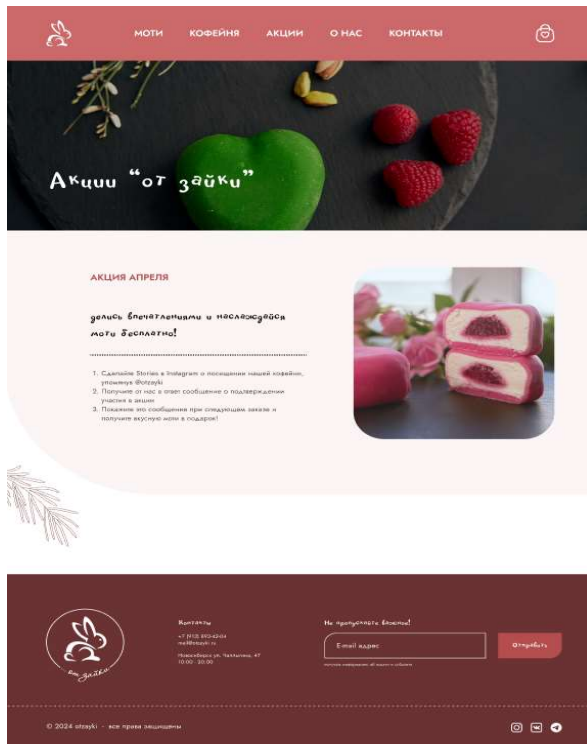


Рис.6.27 Страница акций

Данный компонент представляет собой веб-страницу, которая содержит информацию о текущих акциях.

Верхний раздел содержит изображение десерта и название страницы.

Акция представлена в виде карточки с изображением, названием и подробным описанием условий в основном разделе компонента.

Стили для компонента задаются в файле promo.scss, который импортируется в компонент.

## 5. AboutPage.vue



Рис.6.28 Страница о команде

Данный компонент представляет собой веб-страницу, которая содержит информацию о создателе и главном кондитере, а также о менеджере по работе с клиентами и партнерами.

Верхний раздел содержит изображение десерта и название страницы.

Основной раздел содержит div с классом "about-content". Внутри этого div находятся два div с классами "marina" и "olga". В первом div находится информация о создателе и главном кондитере, а во втором div находится информация о менеджере.

## 6. ContactsPage.vue



Рис.6.29 Страница контактов

Данный компонент представляет собой веб-страницу, которая содержит информацию о контактах кофейни "От зайки". Верхний раздел содержит изображение десерта моти.

Нижний раздел содержит изображение десертов, а также информацию о контактах кофейни, включая адрес, время работы и ссылки на социальные сети.

В конце страницы находится элемент с классом "feedback-form", который содержит форму обратной связи.

Стили для компонента задаются в файле contacts.scss, который импортируется в компонент.

## 7. CartPage.vue

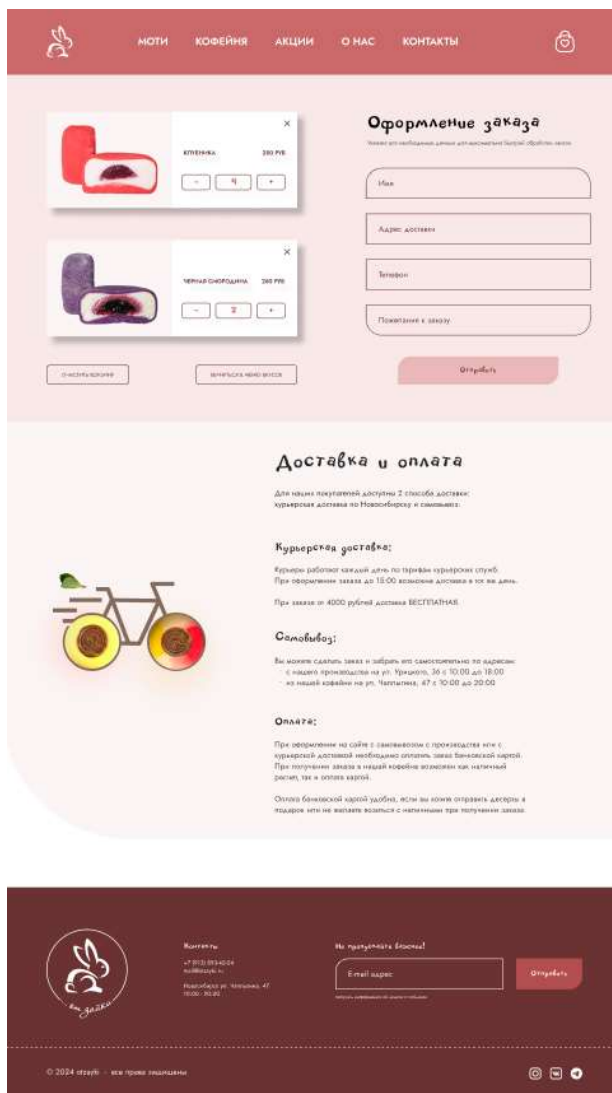


Рис.6.30 Страница корзины

## 8. ErrorPage.vue

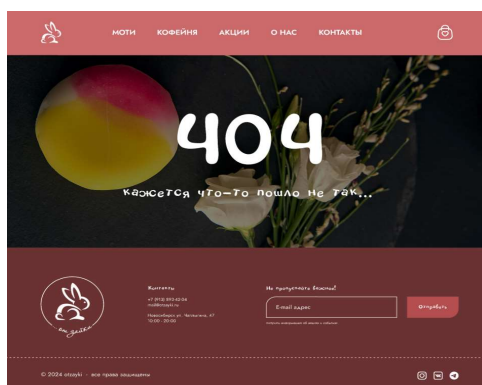


Рис.6.31 Страница ошибки

Данный компонент представляет собой веб-страницу, которая отображает корзину пользователя и позволяет оформить заказ. Цикл `v-for` перебирает элементы корзины и отображает информацию о каждом продукте, включая его название, цену и количество. Ниже идет информация о общей сумме заказа. Справа форма для оформления заказа, которая содержит поля для ввода имени, электронной почты, адреса доставки и примечаний к заказу. В конце формы находится кнопка "Отправить", которая вызывает метод `submitOrder` при нажатии. Он используется для обработки отправки заказа. В компоненте используются методы из `Vuex` для получения данных из хранилища и изменения состояния

Данный компонент представляет собой веб-страницу, которая содержит сообщение об ошибке 404.

В основном разделе находится информационное сообщение, которое расположено на фоне изображения десерта. Пользователю доступно навигационное меню из шапки для решения сложившейся проблемы



## 6.4 Работа с маршрутизацией и управлением состоянием приложения

### 6.4.1 Vue Router

Vue Router является официальным маршрутизатором для приложений Vue.js, который позволяет создавать одностраничные приложения (SPA) с помощью управления путями и отображением различных компонентов на основе URL-адресов.

Определение маршрутов производится в файле настройки маршрутизации (router/index.js). Создается экземпляр VueRouter и определяется объект routes, который будет содержать маршруты:

```
import { createRouter, createWebHistory } from 'vue-router'
import HomePage from "@/pages/HomePage.vue";
import MochiPage from "@/pages/MochiPage.vue";
import CafePage from "@/pages/CafePage.vue";
import ContactsPage from "@/pages/ContactsPage.vue";
import AboutPage from "@/pages/AboutPage.vue";
import PromoPage from "@/pages/PromoPage.vue";
import CartPage from "@/pages/CartPage.vue";
import ErrorPage from "@/pages/ErrorPage.vue";

const routes = [
  {
    path: '/',
    name: 'home',
    component: HomePage
  },
  {
    path: '/mochi',
    name: 'mochi',
    component: MochiPage
  },
  {
    path: '/cafe',
    name: 'cafe',
    component: CafePage
  },
]
```

```

    {
      path: '/contacts',
      name: 'contacts',
      component: ContactsPage
    },
    {
      path: '/about',
      name: 'about',
      component: AboutPage
    },
    {
      path: '/promo',
      name: 'promo',
      component: PromoPage
    },
    {
      path: '/cart',
      name: 'cart',
      component: CartPage
    },
    {
      path: '/*',
      name: 'error404',
      component: ErrorPage
    }
  ]

  const router = createRouter({
    history: createWebHistory(process.env.BASE_URL),
    routes
  })

  export default router;

```

В компонент App.vue добавлен компонент <router-view>. Этот компонент будет отображать компонент, связанный с текущим маршрутом:

```
<template>
  <div class="wrapper">
    <my-header></my-header>
    <router-view></router-view>
    <my-footer></my-footer>
  </div>
</template>

<script>
import MyHeader from "@/components/layout/MyHeader.vue";
import MyFooter from "@/components/layout/MyFooter.vue";

export default {
  components: {MyFooter, MyHeader}
}
</script>

<style lang="scss">
@import "@/styles/styles.css";
</style>
```

Навигация выполняется программно с помощью метода `$router.push`, когда пользователь отправляет форму или событие. Например, навигационное меню для мобильной версии сайта в компоненте `BurgerMenu.vue` выглядит следующим образом:

```
<template>
  <div class="burger">
    <div @click="isHidden = !isHidden" class="burger__button">
      
      
    </div>
    <ul v-show="!isHidden" @click="isHidden = !isHidden" class="burger-menu">
      <li @click="$router.push('/mochi')" class="burger-menu__item"><a href="#">Моти</a></li>
      <li @click="$router.push('/cafe')" class="burger-menu__item"><a href="#">Кофейня</a></li>
      <li @click="$router.push('/promo')" class="burger-menu__item"><a href="#">Акции</a></li>
      <li @click="$router.push('/about')" class="burger-menu__item"><a href="#">О нас</a></li>
      <li @click="$router.push('/contacts')" class="burger-menu__item"><a href="#">Контакты</a></li>
    </ul>
  </div>
</template>

<script>
export default {
  data() {
    return {
      isHidden: true,
    },
  },
}
</script>

<style lang="scss" scoped>
@import "@/styles/burger";
</style>
```

### 6.4.2 Vuex

Vuex - это библиотека управления состоянием для Vue.js, которая помогает управлять данными и состоянием приложения централизованно. Для работы с Vuex необходимо создать файл `store/index.js`, определить состояние, мутации, действия и геттеры. Vuex позволяет эффективно управлять данными и обеспечивает их доступность в различных компонентах приложения.

Базовый шаблон файла `store/index.js` для установки хранилища Vuex выглядит следующим образом:

```
import { createStore } from 'vuex'

export default createStore({
  state: () => ({
    // состояние хранилища
  }),
  mutations: {
    // мутации для изменения состояния
  },
  actions: {
    // действия для выполнения асинхронных операций
  },
  getters: {
    // геттеры для получения вычисляемых свойств
  }
});
```

- `state` - объект, который содержит глобальное состояние вашего приложения.
- `mutations` - функции, которые изменяют состояние синхронно. Внутри мутаций вы изменяете состояние путем манипуляции его свойств.

- actions - функции, которые выполняют асинхронные операции. Внутри действий вы вызываете мутации для изменения состояния. Действия полезны, когда вам нужно вызвать API-запросы или выполнить асинхронные операции, прежде чем изменить состояние.
- getters - функции, которые позволяют получать вычисляемые свойства на основе состояния. Геттеры аналогичны вычисляемым свойствам компонентов Vue.

Можно добавлять дополнительные модули, плагины и расширения в хранилище Vuex в соответствии с требованиями проекта.

Основная идея Vuex - централизация состояния приложения и обеспечение предсказуемого управления состоянием через вызов мутаций или действий. Хранилище Vuex помогает легко контролировать и обрабатывать изменения состояния, а также обеспечивает централизованный доступ к состоянию в компонентах.

## 6.5 Vue Devtools

Vue Devtools - это браузерное расширение, которое предоставляет ряд полезных инструментов для разработки и отладки приложений Vue.js. Он помогает разработчикам исследовать, отлаживать и тестировать приложения Vue с удобством и эффективностью.

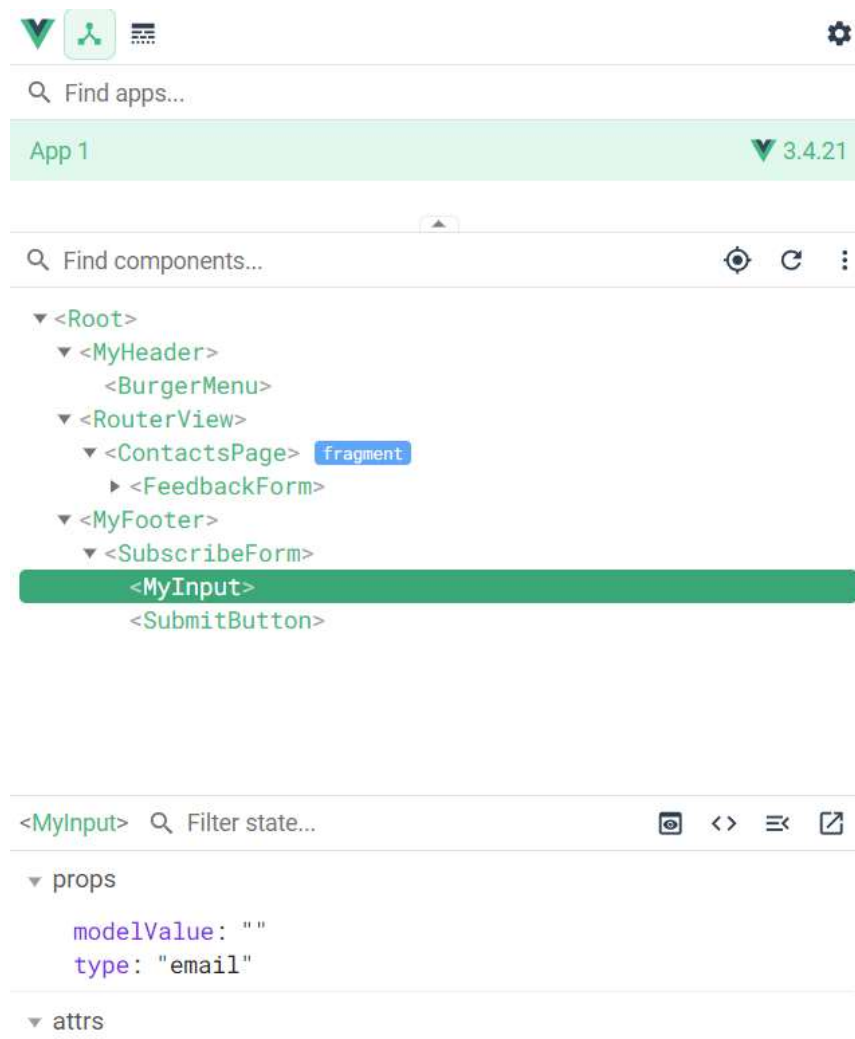


Рис.6.32 Расширение Vue Devtools в браузере Google Chrome

Вот подробнее о возможностях и способах тестирования приложения при помощи Vue Devtools:

1. Инспектирование компонентов: Vue Devtools позволяет просматривать компоненты и их состояние в иерархическом списке. Можно получить доступ к

данным, пропсам, вычисляемым свойствам и методам компонента, а также изменять их для тестирования различных сценариев.

2. Изменение и отладка состояния: можно непосредственно изменять или вставлять значения в состояние компонентов, чтобы проверить, как изменения данных могут влиять на отображение и функциональность приложения. Можно также использовать функции временной отладки для исследования и отслеживания изменений состояния в рабочем приложении.

3. Анализ иерархии компонентов: Vue Devtools позволяет просматривать иерархию компонентов приложения и изучать взаимодействие родительских и дочерних компонентов. Можно отслеживать передачу данных и событий между компонентами, что может помочь в устранении ошибок и оптимизации приложения.

4. Визуализация производительности: Vue Devtools предоставляет графики производительности, показывающие время, затраченное на создание, обновление и удаление компонентов вашего приложения. Это позволяет оптимизировать работу с производительностью и выявить компоненты, которые занимают слишком много времени.

5. Отслеживание и фильтрация событий: можно использовать Vue Devtools для отслеживания событий, реагирующих на действия пользователя или изменения данных. Можно фильтровать события по типу, компоненту или собственной пользовательской логике, что помогает в отладке и тестировании определенных сценариев использования.



## **7. Заключение**

### **7.1. Реализованные результаты и их значимость**

Я горжусь тем, что достигла нескольких ключевых целей, которые ставила перед собой в начале проекта:

1. **Качество продукта:** смогла создать качественное веб-приложение для кондитерской, которое сочетает в себе привлекательный пользовательский интерфейс, быструю и отзывчивую производительность, и интуитивно понятные функциональности. Веб-приложение предоставляет информацию о десертах и кофейне, о проводимых акциях и новостях кондитерской.
2. **Функциональность:** реализовала важные функциональные возможности, такие как просмотр и выбор товаров, оформление заказов, а также возможность оставлять отзывы и контактировать с кондитерской. Все функции были реализованы согласно требованиям проекта и предоставляют удобный и понятный пользовательский опыт.
3. **Дизайн и пользовательский интерфейс:** веб-приложение имеет привлекательный и эстетичный дизайн, сочетающий в себе элементы яркие сочные цвета и минималистический стиль. Интерфейс легок в использовании, интуитивно понятен и дружелюбен к пользователю. Я стремилась к созданию приятной и привлекательной среды для пользователей, где они могут наслаждаться покупками и взаимодействием с нашей кондитерской.

## **7.2. Рекомендации для дальнейшего развития и улучшения**

Важно продолжать улучшать и развивать сайт со временем, отслеживая последние тенденции и требования пользователей. Регулярное обновление и поддержка помогут сохранить сайт актуальным и конкурентоспособным.

Несколько направлений для дальнейшего развития проекта:

1. Анализ данных и обратная связь пользователей: анализ данных о посещаемости сайта, поведении пользователей и конверсии. Использование аналитических инструментов, такие как Google Analytics, для измерения эффективности сайта. Собирать обратную связь от пользователей через формы обратной связи и реагировать на ее основании.

3. Улучшение оптимизации: оптимизировать производительность сайта, чтобы он загружался быстро. Упростить код, улучшить кэширование, сжатие файлов и оптимизировать изображения. Проверить сайт на наличие ошибок скриптов, неиспользованного кода и медленных запросов.

4. SEO-оптимизация: улучшить поисковую оптимизацию сайта, чтобы увеличить видимость в поисковых системах. Использовать уникальные мета-теги, оптимизировать URL-адреса, добавить теги заголовков и осуществить правильное использование ключевых слов на странице.

5. Добавление нового функционала: исследуйте новые возможности и тренды веб-разработки и добавить новый функционал, который будет полезен пользователям. Это могут быть новые интерактивные элементы, службы социальных сетей, интеграция с платежными системами и другое.

6. Поддержка и безопасность: поддерживать сайт, обновлять версии фреймворков и плагинов, исправлять возникающие ошибки и обновлять безопасность. Обеспечить резервное копирование данных и регулярное обновление системы.

7. Обновление контента: планировать и обновлять контент на сайте регулярно. Оптимизировать тексты, добавлять новые материалы и информацию, чтобы сайт оставался актуальным и привлекательным для посетителей.

8. Тестирование: непрерывно тестировать сайт, чтобы выявлять и исправлять ошибки и проблемы. Проводить тестирование функциональности, проверку кросс-браузерности и тестирование производительности.

9. Взаимодействие с пользователями: внимательно слушать отзывы и предложения пользователей, обращайтесь к ним и внедряйте улучшения, которые помогут повысить удовлетворенность и лояльность пользователей.

10. Использование стандартов и лучших практик: соблюдать стандарты веб-разработки и использовать лучшие практики для создания качественного кода и повышения эффективности сайта.

## 8. Список использованной литературы

### Интернет-ресурсы:

1. <https://learn.javascript.ru/>
2. <https://ru.vuejs.org/>
3. <https://vue3js.cn/vuex/ru/>
4. <https://developer.mozilla.org/ru/docs/Web>
5. <https://htmlbook.ru/>
6. <https://metanit.com/web>
7. <https://ru.bem.info/>
8. <https://git.github.io/git-scm.com/book/ru/v2>
9. <https://momentpravdi.ru/cto-takoe-vuejs-v-javascript/>
10. <https://practicum.yandex.ru/blog/kak-adaptirovat-sayt-pod-mobilnye-ustroystva/>
11. <https://developer.mozilla.org/ru/>
12. <https://mentorium.frontend-design.ru/articles/include-fonts/>

### Youtube каналы:

1. <https://www.youtube.com/@UlbiTV>
2. <https://www.youtube.com/@YandexforFrontend>
3. <https://www.youtube.com/@VladilenMinin>
4. <https://www.youtube.com/@JavaScriptNinja>

## 9. Приложение

Ссылка на удаленный репозиторий проекта на Github:

[https://github.com/zevina/Otzayki\\_Zevina\\_GB](https://github.com/zevina/Otzayki_Zevina_GB)

Ссылка на проект на платформе Vercel:

<https://otzayki-zevina-gb.vercel.app/>

Макет сайта (постранично для трех разрешений экрана) размещен ниже: