Geekbrains

Создание интернет-магазина женской одежды.

Программа: Разработчик

Специализация: Frontend разработка (React)

Алфёрова Альбина Владимировна

Москва

2025

Содержание:

Введение.

Дипломный проект написан в рамках специализации Разработчик — Frontend разработка (React).

Целью настоящего дипломного проекта является создание сайта Интернет-магазина как инструмента для продажи и ведения бизнеса.

После установки, наполнения его контентом и размещения на хостинге, сайт начнёт исполнять свои цели, функции и задачи.

К таким функциям относятся: возможность создания, просмотра и редактирования категорий товаров, обслуживания заказа.

Глава 1. Основы создания интернет-магазина

1.1 История создания интернет-магазинов.

Уже в 1981 году Томсон Холидэйс выдвинул идею создания B2B или "Business to Business". Основа этой системы заключалась в корпоративном бизнесе, в сбыте товаров не рядовым потребителям, а фирмам, структурным юридическим лицам. В роли товара могли выступать электронные торговые площадки, b2b-медиа продукты (книги, информационные бюллетени), нацеленные на вопросы менеджмента, логистики.

В ноябре 1990 года Тим Бернерс-Ли из CERN (Европейского центра ядерных исследований) создал первый прототип WWW(World Wide Web)-сервера.

Уже в 1992 году был создан протокол HTTP для работы с гипертекстами. Позже в 1992 году Чарльз Стэк основал первый интернет-магазин books.com по продаже книг.

В 1994 году американец Джефф Безос основал тогда небольшой ресурс под названием Amazon, который начинал с продаж книг на территории США.

По прогнозу Data Insight, в 2025 году объем продаж на российском рынке eCommerce достигнет 10,2 трлн рублей, рост составит 30%; (сайт https://datainsight.ru/eCommerce_2023).

1.2 Преимущества интернет-торговли.

Интернет-магазины обладают целым рядом преимуществ перед традиционными магазинами. Интернет-магазин работает 24 часа 7 дней в неделю.

Пользователь интернет-магазина может быстро узнать актуальную стоимость и наличие товара в магазинах, сравнить описание и характеристики с товарами на других сайтах, просмотреть сведения о скидках, акциях. Использование фильтра поможет найти товар с необходимыми характеристиками.

Происходит экономия личного времени при выборе товара, покупке, его оплате и доставке. Интернет-магазин позволяет сэкономить время пользователей, избежать подводных сложностей и полностью заменить поездку в магазин и общение с консультантом, что приносит личную и экономическую выгоду.

Также создание интернет-магазина — это хороший способ начать свой собственный бизнес. Нет необходимости в аренде склада и торгового помещения.

С интернет-магазином нет привязки к региону. А наличие склада и офиса в регионе сократит затраты по отношению к конкурентам и избавит от расширений. Очень много бизнес моделей с отправкой по всей России.

В настоящее время основная активность происходит в интернете: общаются в интернете, развлекаются в интернете, покупают в интернете, обучаются в интернете. Если бизнеса нет в интернете, значит бизнес неконкурентоспособен. Создание интернет-магазина расширяет возможность получения дохода: интернет -реклама позволяет показывать рекламу целевой аудитории. Эффективность рекламных объявлений повышается, отдача от рекламы значительно ощущается. Упрощает запуск продаж на маркетплейсах , имея актуальный каталог на сайте с легкостью можно разместить на Ozon, Wildberries и др. На основе сбора статистики покупок, можно предлагать покупателям персональную рекламу.

Процесс обновления товаров в магазине намного проще и быстрее чем обновление ассортимента в рознице. Если в розничном магазине обновлением ассортимента занимается отдельно выделенный человек, то с интернет-магазином можно настроить обновление в автоматическом режиме. Можно напрямую обновлять с сайта поставщика - через ссылку XML, CSV, прайс листы.

В интернет-магазине имеется расширенная почтовая рассылка, благодаря которой можно уведомить покупателей о скидках и акциях в любое время. Интернет-магазин позволяет автоматизировать простые процессы. Эффективность работы повышается. Человеческого внимания потребуется меньше, а работа будет проходить быстрее.

Покупатель сам может выбрать удобный способ доставки и оплаты за товар в интернет магазине.

Минимизировать затраты на аренду, сотрудников, если настроена автоматизация, интернет-магазин позволяет заменить сотрудников в некоторых моментах. Большинство компаний закрывают розничные магазины, оставляют только склад и внедряют онлайн продажи.

На сильно конкурентных рынках, интернет-магазин позволяет снизить стоимость продукции за счет минимизации затрат.

Интернет-магазин — это форма электронной торговли, которая позволяет потребителям покупать товары или услуги с помощью веб-браузера. Интернет-магазин (англ. online store или e-shop) —сайт, торгующий товарами посредством сети Интернет Позволяет пользователям онлайн, в своём браузере, или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. При этом продажа товаров осуществляется дистанционным способом. Когда онлайн-магазин настроен на то, чтобы позволить компаниям покупать у других компаний, этот процесс называется онлайн-магазинами бизнес для бизнеса (B2B)

(https://ru.wikipedia.org/wiki/)

1.3 Этапы создания интернет-магазина.

Примерный пошаговый план проектирования сайта для интернет-торговли:

Определите нишу и целевую аудиторию. Решите, вопрос, о том какие товары вы будете продавать и кому.

Проанализируйте конкурентов. Изучите, как работают другие магазины в вашей нише.

Создайте бизнес-план. Включите цели, стратегии маркетинга, бюджет и план развития.

Выбор платформы для интернет-магазинов.

Хостинг и домен. Выберите надежного хостинг-провайдера и зарегистрируйть домен.

На данном этапе исполнитель создаёт схему основных страниц сайта. На ней указано где и какие блоки будут на странице: картинки, кнопки, ссылки для удобного интерфейса.

После утверждения схемы начинается этап дизайна, который включает в себя подбор уникального внешнего вида страниц сайта, цвета, форматирования.

Дизайнер готовит версии для различных устройств: компьютер, планшет, мобильный телефон. Весь материал собирается в дизайн-макет. Убедитесь, что сайт хорошо отображается на мобильных устройствах.

Проверьте навигацию. Сделайте меню удобным и интуитивно понятным.

Создайте интеграцию с платежными системами. Подключите популярные способы оплаты (банковские карты, PayPal и т.д.).

Наполните сайт контентом.

Над проектом работают различные специалисты: Front-End Developer получают дизайн макет и начинают верстать его с использованием современных технологий и решений.

На данном этапе уже можно смотреть эффекты анимации в браузере, работу вёрстки на мобильных устройствах. Также вступает в работу программист Back End Developer. Он

занимается интеграцией вёрстки в CMS и программированием функционала. На данном этапе реализуется логика работы сайта, реализуются все основные пользовательские сценарии.

Затем программист сдаёт проект интернет-магазина тестировщику, который прогоняет различные сценарии с целью найти ошибки в работе функционала. Тестировщик сообщает о найденных багах. Программист правит баги до полного принятия работы тестировщиком.

После того, как функционал интернет-магазина протестирован, начинается работа по наполнению контентом. Для интернет-магазинов наполнение производится с помощью импортов прайс-листов в каком-либо заданном формате. Реже каждый товар заводится по одному. Создание карточек товаров: добавляется описание, фото, цены и характеристики товаров. Блог и новости: здесь размещается полезная информация, которая будет привлекать клиентов.

SEO-оптимизация: оптимизируется контент для поисковых систем.

Когда интернет-магазин наполнен и готов к запуску, исполнитель предоставляет заказчику документацию по проекту. Как минимум, готовятся пользовательские инструкции для администратора и контент-менеджера. Заказчик на выходе должен уметь управлять интернет-магазином через веб-интерфейсы CMS.

Следующий важный этап релиз – выкатывание всех наработок с тестового на рабочий сайт с настройкой сервера и самого сайта.

Важная часть проекта маркетинг и продвижение.

Социальные сети: создайте страницы в социальных сетях и продвигайте магазин.

Контекстная реклама: используйте Google Ads, Яндекс.Директ и другие платформы для привлечения клиентов.

Email-маркетинг: собирайте базы подписчиков и рассылайте новости и предложения.

SEO: работайте над улучшением позиций в поисковых системах.

Обслуживание клиентов

После релиза сайт еще раз проверяется тестировщиком и принимается заказчиком. На этом этапе наступает время подписать акты и закрывать работы по созданию интернет-магазина.

Доработки и развитие работающего сайта – это новый проект, который называется «Поддержка».

Техподдержка: обеспечьте быстрый ответ на вопросы клиентов через чат, email или телефон.

Отзывы и рейтинги: стимулируйте клиентов оставлять отзывы и оценивайте товары.

Система лояльности: внедрите программы скидок и бонусов для постоянных клиентов.

Аналитика и оптимизация.

Мониторинг: используйте инструменты аналитики (Google Analytics Яндекс.Метрика) для отслеживания посещаемости и продаж.

Оптимизация: регулярно улучшайте сайт и маркетинговые стратегии на основе собранных данных.

Глава 2. Инструменты разработчика

Понятие клиент- сервер.

Интернет магазин создается на основе WEB-технологий, комплекса технических,

коммуникационных и программных методов. Веб-сайт и его страницы хранятся или размещаются на веб-сервере. Веб-серверы — это компьютеры, обычно работающие под управлением Windows Server или, что чаще, какой-либо разновидности операционной системы Linux, например CentOS. На этих компьютерах работает часть программного обеспечения, называемая веб-сервером. Обычно это Apache, IIS или NGINX.

При этом используются такие основные понятия, как клиенты (АРМ администратора, АРМ оператора, АРМ клиента, смартфон клиента) и серверы (сервер базы данных, сервер с файлами php). Упрощённая схема того, как взаимодействуют взаимодействуют сервер и клиент, может выглядеть следующим образом:

Запросы (HTTP Requests) — сообщения, которые отправляются клиентом на сервер для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок. Ответы (HTTP Responses) — сообщения, которые сервер отправляет в ответ на клиентский запрос. Заголовки HTTP (англ. HTTP Headers) — это строки в HTTP-сообщении, содержащие разделённую двоеточием пару имя-значение. Форма заголовков соответствует общему формату заголовков текстовых сетевых сообщений ARPA (https://selectel.ru/blog/http-request). Клиенты являются обычными пользователями, подключёнными к Интернету посредством устройств (например, компьютер подключён к Wi-Fi, или телефон подключён к мобильной сети) и программного обеспечения, доступного на этих устройствах (браузер, например, Яндекс или Chrome).

Какие инструменты могут понадобиться веб-разработчику:

1. FTP-клиент, например, FileZilla, WinSCP, Total Commander, Cyberduck.

2. Плагины, например, Firebug, Web Developer.

3. Валидатор.

4. Оптимизация изображений: TinyPNG.

5. Совместимости разных версий браузеров: Can I Use.

FTP-клиент

FTP — это протокол передачи файлов. Через него файлы можно скопировать с компьютера на сервер и обратно. Ещё одна из часто используемых возможностей – редактирование файлов на сервере.

Для использования FTP необходима специальная программа. Практически все программы FTP работают таким образом: появляются два окна, где первое – ваши файлы на сервере в интернете, а второе - файлы на компьютере. Можно легко передавать файлы на компьютер или сервер.

Предположим, программа FTP уже выбрана и установлена на компьютер, но для работы необходимо знать FTP-адрес вашего веб-сервера, имя пользователя и пароль. Это важная информация, её можно получить на хостинге и нужно обязательно записать. Если все данные есть, можно попробовать получить доступ к файлам FTP. Действия могут быть разными в зависимости от программы, но в любом случае нужно создать новое соединение с именем пользователя и паролем. Это позволит сохранить настройки и исключить необходимость ввода данных каждый раз при входе в программу.

После создания профиля и ввода всех данных нажмите «Соединиться», после чего программа начнёт соединяться с сервером. Если все настройки верны, через несколько секунд (в зависимости от скорости интернета) можно будет увидеть все свои файлы, находящиеся на сайте.

На веб-сервере находится набор папок и файлов. Подключаясь к серверу, программа отправляет запрос через интернет на специальный сервер и получает данные. Существуют протоколы и с более расширенным функционалом, нежели обычный веб-сёрфинг.

FTP-программа даёт возможность создавать, редактировать, удалять файлы и папки на веб-сервере, а также передавать файлы на сервер и обратно.

Плагин

Плагины — это программные модули или дополнения, которые расширяют функциональность основного программного обеспечения. Они позволяют добавлять новые функции или модифицировать поведение программы без необходимости изменять ее исходный код. Плагины широко используются в различных приложениях, от веб-браузеров до графических редакторов и игровых платформ.

Основные характеристики плагинов:

1. Интеграция с основной программой: Плагины работают в рамках основной программы, используя ее интерфейсы и возможности.
2. Легкость добавления и удаления: Они часто устанавливаются и удаляются отдельно, что позволяет пользователю гибко адаптировать функционал под свои нужды.
3. Разнообразие задач: Плагины могут выполнять самые разные функции — от повышения производительности до добавления новых возможностей.

Например, плагин Firebug позволяет пользователям запускать код JavaScript через командную строку и регистрировать ошибки, возникающие в JavaScript, CSS и XML

. Firebug предоставляет отдельный текстовый редактор для изменения JavaScript и немедленного просмотра результатов в браузере пользователя.

Плагин Web Developer - изначально это расширение создавалось для Firefox, но теперь доступно и в Chrome.

Web Developer добавляет панель инструментов, через которую разработчики могут менять размеры браузера, выделять картинки без атрибутов alt, отключать JavaScript, смотреть данные о метатегах сайта, изучать HTML-код страницы и многое другое.

Преимущества плагинов:

Упрощают добавление новых функций.

Уменьшают нагрузку на разработчиков основного продукта.

Позволяют пользователям адаптировать программное обеспечение под свои потребности.

Недостатки:

Некоторые плагины могут быть несовместимы между собой.

Использование большого количества плагинов может замедлить работу программы.

Риск безопасности, если плагин создан ненадежным источником.

Плагины — это удобный способ применения стандартного решения так, чтобы оно лучше соответствовало конкретным задачам или желаниям, делая его более функциональным и гибким. В рамках дипломного проекта также используется плагин "Smart Basket".

Валидатор.

Чистый валидный код в HTML и CSS повышает скорость загрузки и шансы, что на всех без исключения устройствах и браузерах вёрстка будет выглядеть так, как необходимо. Поэтому любой вёрстке нужно проводить валидацию. Валидаторы W3C — два сервиса, которые помогают держать код в чистоте и порядке.

HTML-валидатор производит несколько проверок кода:

1. Валидация синтаксиса — проверка на наличие синтаксических ошибок.

2. Проверка вложенности тегов — они должны быть закрыты в обратном порядке относительно их открытия.

Валидация DTD — проверка соответствия вашего кода указанному Document Type Definition. Она включает проверку названий тегов, атрибутов и «встраивания» тегов.

4. Проверка на посторонние элементы выявляет всё, что есть в коде, но отсутствует в DTD.

Типограф.

Типографами называются программы или утилиты, позволяющие откорректировать набранный текст в соответствии с правилами типографского набора. Например, для того, чтобы заменить машинописные кавычки на французские, дефисы на тире, а также расставить неразрывные пробелы в тех местах, где они требуются. Добавить спецсимволы, если такие требуются, доступ к типографу по адресу https://www.artlebedev.ru/typograf/

Оптимизация изображений

Оптимизация графики — не менее важный аспект вёрстки, чем чистота кода. Изображения бывают разных форматов. Для вёрстки самое важное — это размер файла. При скачивании изображений в необходимом формате размер файла может быть слишком большим, поэтому после экспорта PNG-файлов рекомендуется оптимизировать графику в TinyPNG. После обработки файл теряет 30–70% веса, но на качество это никак не влияет.

Совместимость разных версий браузеров.

В процессе работы любой веб-разработчик сталкивается с таким понятием, как кроссбраузерность. Оно означает, что любой сайт должен выглядеть одинаково во всех браузерах. Главный виновник этой проблемы — Internet Explorer (IE).

Пути решения проблемы кроссбраузерности:

1.Необходимо проверять, как выглядит сайт во всех популярных браузерах, с Некоторой периодичностью во время вёрстки. Тогда вы сможете оперативно устранять несоответствия.

2. Некоторые теги HTML, например, заголовки, параграфы, списки, изначально имеют определённый набор свойств и значений. Они могут определяться каждым браузером по-разному. Чтобы вид HTML-страницы не зависел от того, с помощью какого браузера её просматривают, используется сброс стилей CSS. Обычно, если используется сброс стилей, их определяют в отдельном файле, который подключают на веб-страницу при помощи тега <link>.

3.Воспользоваться хаками. Хак — исправление ошибки или добавление новой функции посредством использования другой недокументированной или некорректно реализованной особенности. Необходимость добавлять хаки возникает в основном для IE, но ими пользоваться не рекомендуется, так как это противоречит спецификации CSS Чтобы исправить несоответствия отображения в IE, лучше воспользоваться решением из следующего пункта.

4. Объявить условный комментарий. Он представляет собой обычный HTML комментарий, где в квадратных скобках указывается условие, которое понимает

только браузер IE, а все остальные браузеры игнорируют.

Отладчик браузера.

В отладчике браузера можно:

- просматривать HTML-код элемента;

- инспектировать любые сайты в интернете;

- точно определять структуру сайта;

- вносить изменения в код;

Размещение сайта в интернете.

- Выбираем любой бесплатный хостинг.

- Выбираем бесплатный тариф.

- Регистрируемся (при необходимости).

- Нажимаем «Создать сайт» (придумываем имя сайта).

- Нажимаем кнопку «Управлять сайтом».

- Выбираем файловый менеджер.

- Загружаем всё, что создали, в файловый менеджер.

2.2 Инструменты для написания сайта Интернет магазина

При написании интернет-магазина использовалась технология REACT

React — это JS-библиотека, коллекции предварительно написанных фрагментов кода, которые можно повторно использовать для выполнения стандартных функций JavaScript. Библиотеки и компоненты предлагают готовые к использованию элементы, такие как кнопки, значки, формы, меню и шаблоны макетов. В отличие от множества других UI фреймворков, этот инструмент сокращает время на разработку. React сочетает в себе

простоту кода, логику отображения, производительность и масштабируемость. Еще один плюс — возможность переиспользовать код компонента в разных частях интерфейса и даже в других проектах.

В React используется синтаксис JSX, расширение языка JavaScript, который выглядит как обычный html. При создании веб-приложений код пишется на чистом JS. Этот UI-фреймворк избавляет от необходимости взаимодействовать с DOM за счет того, что создается дополнительный слой абстракции.

React представлен с открытым исходным кодом, и его можно без ограниченийиспользовать для разработки коммерческих проектов.

Разработчику, который будет использовать этот инструмент, нет необходимости изучать упаковщики rollup, parcel или другие, даже если сборка проекта будет происходит с их помощью. Опционально совместно с JS-библиотекой может использоваться JSX, но это не обязательно.

Развитое комьюнити — еще одна особенность UI-фреймворка. В процессе работы могут возникнуть вопросы, ответы на которые почти всегда находятся в профессиональных сообществах.

Преимущества разработки интернет-магазина на React : Интернет-магазин, разработанный с использованием инструментов из экосистемы React — это площадка с высокой производительностью и логично выстроенным интерфейсом. Такой проект легко масштабировать, при этом разработчики могут сосредоточиться на проектировании архитектуры, а рутинные задачи будет выполнять Реакт. Преимущества JavaScript-библиотеки применительно к разработке интернет-магазина:

Универсальность. Инструмент одинаково хорошо подходит для использования на сервере и на мобильных платформах (iOS или Android). Для мобильной разработки применяется UI-фреймворк React Native.

Множество готовых решений. Использование готовых компонентов, которые можно адаптировать под индивидуальные проекты, позволяет экономить ресурсы и разрабатывать интернет-магазины в короткие сроки.

Декларативность. В отличие от императивного подхода с пошаговыми инструкциями, декларативный предполагает описание конечного пользовательского интерфейса в разных состояниях. Это экономит время разработчика.

Virtual DOM. Для улучшение кроссбраузерной совместимости и производительности Реакт реализует систему DOM, независимую от браузера. Для клиента это означает быструю загрузку страниц и отзывчивость интернет-магазина в целом.

Компонентная архитектура. Нужный компонент создается один раз, и затем многократно используется для разных задач. Благодаря компонентному подходу достигается простота масштабируемости. При расширении функциональности веб-проекта и добавлении новых элементов интерфейс выглядит целостным и завершенным, при этом его логика не усложняется.

Минимальная вероятность ошибок. Особенность JS-библиотеки React — нисходящая передача данных (сверху вниз). Это облегчает отладку и сводит к минимуму вероятность ошибок.

Кроме того, за библиотекой React стоит огромное сообщество, она всесторонне протестирована, есть стабильная поддержка и регулярные обновления. Переход на новые

версии проходит без проблем. Почти на для всех возможных задач есть готовые компоненты, а если у клиента уникальная ситуация, под которую в библиотеке нет решения, его можно найти в независимой библиотеке и интегрировать в React.

Сейчас React — это не просто библиотека, а целая экосистема, с помощью которой можно создавать интернет-магазины любой сложности. Огромное количество документации и развитое сообщество позволяет разобраться в основах за несколько дней. На React Native написано много популярных мобильных приложений: Skype, Нетфликс, Walmart, Pinterest, Bloomberg, Uber Eats, Wix, Airbnb.

Технологии, которые будут использоваться в проекте:

Приложение - React (Hooks)

Глобальное управление состоянием - Redux, Redux Toolkit

Маршрутизация - React Router

Стили — CSS

3.1 Структура каталога

Структура каталогов верхнего уровня будет выглядеть следующим образом:

assets- глобальные статические файлы, такие как изображения, svg, логотип компании и т.д.

components - глобальные общие/повторно используемые компоненты, такие как макеты (обертки), компоненты форм, кнопки

services - JavaScript модули

store - глобальное хранилище Redux

utils - утилиты, помощники, константы и т.п.

views - можно также назвать pages, здесь будет содержаться большая часть приложения. Лучше сохранять привычные соглашения, где это возможно, поэтому src содержит все, index.js является точкой входа, а App.js устанавливает аутентификацию и маршрутизацию.

3.3 Файл index.js в приложении интернет-магазина

Файл index.js в приложении интернет-магазина является входной точкой (entry point). Он отвечает за инициализацию и рендеринг React-приложения в браузере. В библиотеке React, render - это метод жизненного цикла компонента, который вызывается для отображения компонента на странице. Он возвращает React-элемент представляющий содержимое компонента. Метод render определяет, что должно быть отображено на экране в ответ на изменение состояния или свойств компонента.

Импорт необходимых модулей

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { BrowserRouter } from 'react-router-dom'; // Импортируем BrowserRouter
import './index.css';
import App from './App';
```

Для чего нужен импорт:

React: библиотека для создания компонентов.

ReactDOM: используется для рендеринга компонентов в HTML.

BrowserRouter из react-router-dom: позволяет управлять страницами (маршрутизацией) в одностраничном приложении (SPA).

index.css: подключаем глобальные стили.

App.js: главный компонент приложения, который содержит всю логику и структуру.

Создание корневого элемента

const root = ReactDOM.createRoot(document.getElementById('root'));

Находит элемент в index.html и подключает к нему React-приложение.

Использует ReactDOM.createRoot() — это новый способ рендеринга, который работает с React 18 и улучшает производительность.

3. Рендеринг главного компонента App

root.render(

<BrowserRouter> {/* Оборачиваем в Router */}

<React.StrictMode>

<App />

</React.StrictMode>

</BrowserRouter>

);

При написании данного кода происходит следующее:

1. Оборачиваем приложение в <BrowserRouter>, чтобы оно могло использовать маршрутизацию (react-router-dom).

2. <React.StrictMode> — помогает находить потенциальные ошибки в коде (активен только в режиме разработки).

3. Рендерится <App />, который является главным компонентом приложения.

Таким образом, index.js в интернет-магазине выполняет следующие функции:

- Опеделяет точку входа в приложение (где именно в HTML будет рендериться React).

- Настраивает маршрутизацию (BrowserRouter позволяет переключаться между страницами).

- Применяет глобальные стили (index.css).

- Включает режим строгой проверки (React.StrictMode) для отлова багов.

- Без этого файла невозможно запустить приложение

3.4 Описание кода App.js.

Импорт и структуры компонента:

Импорты:

Header, Footer, Items, Categories, ShowFullItem, и Register — это компоненты, которые будут использоваться в главном компоненте App.

Routes и Route — это элементы из библиотеки react-router-dom, предназначенные для маршрутизации в приложении, чтобы переходить между различными страницами.

Компонент App является классом, который наследует от React.Component. Этот компонент управляет состоянием приложения и рендерит различные подкомпоненты в зависимости от текущего состояния.

Состояние (state):

orders: массив заказов, которые добавляет пользователь.

currentItems: массив товаров, которые отображаются в данный момент. Изначально он равен всем товарам из массива items.

items: массив объектов товаров. Каждый товар включает:

id: уникальный идентификатор товара.

img: путь к изображению товара.

title: название товара.

text: описание товара.

category: категория товара (например, "school", "wedding").

price: цена товара.

showFullItem: логическое значение, которое контролирует, показывать ли полную информацию о товаре.

fullItem: объект, который содержит данные о выбранном товаре, если он был выбран для отображения.

Методы:

addToOrder(item): Добавляет товар в корзину (orders), если его еще нет в корзине. Для этого проверяется, есть ли товар с таким же id в массиве заказов.

deleteOrder(id): Удаляет товар из корзины по его id.

chooseCategory(category): Фильтрует товары по категории. Если категория "all", то показываются все товары, если другая категория, то отображаются только те товары которые соответствуют выбранной категории.

onShowItem(item): Устанавливает состояние, при котором показывается полная информация о товаре. Если товар выбран, то его данные сохраняются в fullItem, и отображается компонент ShowFullItem.

JSX-разметка:

В рендере компонента:

Header: отображает заголовок и корзину с товарами.

Routes: используется для маршрутизации. Здесь определен только один маршрут для страницы регистрации (/register), где отображается компонент Register.

Categories: отображает список категорий товаров (например, "school", "wedding"). При выборе категории будет фильтроваться список товаров.

Items: отображает список товаров. При клике на товар показывается полная информация о нем.

ShowFullItem: компонент, который отображает подробную информацию о выбранном товаре (если showFullItem равно true).

Footer: отображает нижний колонтитул страницы.

Описание компонентов:

Categories: Компонент, который позволяет пользователю выбрать категорию товара для фильтрации. Он вызывает метод chooseCategory.

Items: Компонент для отображения списка товаров. Каждый товар можно добавить в корзину с помощью метода addToOrder, а также нажать на товар, чтобы увидеть его полную информацию через метод onShowItem.

ShowFullItem: Компонент, который отображает полную информацию о товаре, выбранном пользователем. Этот компонент отображается только при активном состоянии showFullItem.

Register: Страница регистрации пользователя, которая отображается при переходе по маршруту /register.

Как работает фильтрация и отображение товаров:

При выборе категории в компоненте Categories вызывается метод chooseCategory, который обновляет currentItems — это массив товаров, которые должны быть отображены в данный момент (отфильтрованные по категории).

Если выбран "all", показываются все товары.

Переходы по страницам:

С помощью react-router-dom добавлен маршрут /register, который отображает страницу регистрации. Это расширяет функциональность приложения, позволяя пользователю перейти на страницу для регистрации.

3.4 Описание компонента Categories.js

Импортируем React, потому что работаем с React-компонентами.

```
const Categories = ({ chooseCategory }) => {
```

Объявляем функциональный компонент Categories.

{ chooseCategory } — это деструктуризация, она сразу "извлекает" chooseCategory из props.

Список категорий:

```
const categories = [
    { key: "all", name: "Всё" },
    { key: "school", name: "Выпускной" },
    { key: "wedding", name: "Свадьба" },
    { key: "holiday", name: "Отдых" },
    { key: "celebration", name: "Торжество" },
    { key: "concert", name: "Концерт" },
    { key: "party", name: "Вечеринка" }
];
```

Это просто статический массив объектов. В каждом объекте есть: key — уникальный идентификатор, name — название категории, которое будет видно на экране.

Отображение списка категорий:

```
return (
<div className="categories">
        {categories.map((el) => (
        <div key={el.key} onClick={() => chooseCategory(el.key)}>
        {el.name}
    </div>
))}
</div>
);
```

Оборачиваем всё в div с классом categories. Это нужно для стилизации (CSS).

map() — перебираем массив categories. map() создаёт новый div для каждой категории. Это аналог this.state.categories.map() в классовом компоненте.

Атрибут key

key={el.key}

React требует key для списков, чтобы правильно обновлять элементы, key должен быть уникальным, поэтому мы используем el.key из массива.

Обработчик клика onClick

onClick={() => chooseCategory(el.key)}

Когда пользователь кликает на категорию, вызывается

chooseCategory(el.key).

chooseCategory — это функция, переданная из родительского компонента.

el.key передаёт идентификатор выбранной категории.

Экспорт компонента:

export default Categories;

Делаем компонент доступным для использования в других файлах.

Теперь его можно импортировать в родительский компонент:

import Categories from "./Categories";

<Categories chooseCategory={(category) => console.log("Выбрали:", category)} />;

3.5 Описание компонента Header.js

Компонент Header:

- Отображает навигационное меню (логотип, ссылки "Главная", "Контакты", "Кабинет").
- Позволяет открыть/закрыть корзину.
- Показывает список товаров в корзине или сообщение "Товаров нет".
- Подсчитывает общую сумму товаров в корзине.

Импорты:

import React, { useState } from 'react'

import Order from './Order'

import { IoIosBasket } from "react-icons/io";

import { Link } from 'react-router-dom'; // Импорт Link

useState — хук для управления состоянием (cartOpen).

Order — отдельный компонент, который рендерит товар в корзине.

IoIosBasket — иконка корзины из react-icons/io.

Link — компонент из react-router-dom для перехода между страницами без перезагрузки.

Функция showOrders(props)

```
const showOrders = (props) => {
	let summa = 0
	props.orders.forEach(el => summa += Number.parseFloat(el.price)) // Считаем сумму всех товаров
	return (<div>
	{props.orders.map(el => (
	<Order onDelete={props.onDelete} key={el.id} item={el} /> // Отображаем
	каждый товар через компонент Order
))}

<p className='summa'>Сумма: {new Intl.NumberFormat().format(summa)}₽</p> // Выводим сумму товаров
	</div>)
}
```

Перебирает props.orders (массив товаров в корзине).

Считает общую сумму (summa).

Рендерит список товаров через компонент <Order />, передавая в него onDelete (функцию удаления).

Отображает сумму в корзине, отформатированную с разделителями (Intl.NumberFormat()).

Пример входных данных (props.orders):

```
[
    { id: 1, name: "ПЛАТЬЕ МАКСИ В АЛОМ ЦВЕТЕ", price: "5000" },
    { id: 2, name: "ГЛАВНЫЙ ДЕНЬ", price: "4500" }
]
```

Выведет:

Сумма: 9500₽

Функция showNothing()

```
const showNothing = () => {
    return (<div className='empty'>
    <h2>Товаров нет</h2>
    </div>)
}
```

Если корзина пуста, отображает сообщение "Товаров нет".

Константа Header:

```
const Header = (props) => {
    let [cartOpen, setCartOpen] = useState(false); // Состояние для открытия/закрытия корзины
```

cartOpen — true, если корзина открыта, иначе false.

setCartOpen — функция для изменения cartOpen.

Отображение логотипа и навигации:

```
<header>
    <div>
        <span className="logo">women's stories</span>
        <ul className='nav'>
        <li><Link to="/" style={{ cursor: 'pointer' }}>Главная</Link></li>
        <li>Контакты</li>
        <li>
        <Link to="/register" style={{ cursor: 'pointer' }}>Кабинет</Link>
        </li>
        </ul>
```

Логотип "women's stories".

Меню с Link (чтобы переходить между страницами без перезагрузки).

"Главная", "Контакты", "Кабинет".

Кнопка корзины:

```
<IoIosBasket onClick={() => setCartOpen(prevState => !prevState)}
    className={`shop-cart-button ${cartOpen && 'active'}`} />
    onClick={() => setCartOpen(prevState => !prevState)}
```

Переключает cartOpen между true и false при клике.

Если корзина закрыта → откроется. Если открыта → закроется.

className={shop-cart-button ${cartOpen && 'active'}}

Добавляет класс "active", если cartOpen === true.

Открытие/закрытие корзины:

{cartOpen && (

{props.orders.length > 0 ? showOrders(props) : showNothing()}

)}

Если cartOpen === true, рендерится (корзина).

Если props.orders.length > 0, отображаем showOrders(props).

Если в корзине пусто, вызываем showNothing().

Пример работы:

Если в props.orders есть товары, выведется их список и сумма.

Если корзина пуста, выведется "Товаров нет".

Презентационный блок:

Этот блок содержит изображение.

Экспорт компонента:

export default Header;

Это позволяет использовать <Header /> в других файлах.

Таким образом, весь компонент выполняет следующие функции в части:

1. Навигационное меню (Главная, Контакты, Кабинет).

2. Корзина (можно открыть/закрыть кликом на иконку).

3. Если в корзине есть товары → показываются товары и сумма.

4. Если корзина пуста → сообщение "Товаров нет".

Этот компонент используется в родительском компоненте App.js:

import React, { useState } from 'react';

import Header from './Header';

3.6 Описание компонента Item.js

Этот код предназначен для создания карточки товара:

import React, { Component } from "react";

Импортируется React и класс Component.

export class Item extends Component { render() { ... } }

Это классовый компонент Item.

Он принимает props (данные о товаре).

Возвращает JSX-разметку для отображения товара.

- Оборачивает весь контент карточки товара.

- Стилизуется через CSS (.item { ... }).

img – изображение товара

<img

src={process.env.PUBLIC_URL + "/" + this.props.item.img}

onClick={() => this.props.onShowItem(this.props.item)}

alt={this.props.item.alt || "img"}

/>

src={process.env.PUBLIC_URL + "/" + this.props.item.img}

Формирует путь к картинке в папке public.

process.env.PUBLIC_URL — это путь к папке public/ в проекте React.

this.props.item.img — это название файла картинки.

onClick={() => this.props.onShowItem(this.props.item)}

Вызывает функцию onShowItem при клике на картинку (например, чтобы открыть карточку товара).

alt={this.props.item.alt || "img"}

Добавляет альтернативный текст (если alt не передан, будет img).

Название, описание и характеристики

{this.props.item.title}

{this.props.item.text}

{this.props.item.property}

{this.props.item.price}

Выводит основные данные товара:

title – название товара

text – краткое описание

property – характеристика

price – цена (жирным)

+ Кнопка добавления в корзину

<div className="add-to-cart" onClick={() => this.props.onAdd(this.props.item)}>+

Кнопка, которая добавляет товар в корзину:

className="add-to-cart" → стилизовутся в CSS.

onClick={() => this.props.onAdd(this.props.item)}

При клике вызывается onAdd (функция, добавляющая товар в корзину).

Таким образом:

- Этот компонент создаёт карточку товара с: картинкой, названием, описанием, ценой и кнопкой "Добавить в корзину".

- При клике на картинку можно открыть подробное описание товара.

- При клике на + товар добавляется в корзину.

3.7 Описание компонента Items.js

Импорты:

import React from 'react';

import Item from './Item';

import React from 'react' – подключаем библиотеку React

import Item from './Item' – импортируем компонент Item, который отвечает за отображение одного товара.

Определение функционального компонента Items:

const Items = ({ items, onShowItem, onAdd }) => {

Создаём функциональный компонент Items (используем стрелочную функцию).

Принимаем props (свойства) через деструктуризацию, синтаксическое

удобство, которое облегчает извлечение данных из массивов или объектов,

позволяя прямо «распаковывать» их значения в переменные:

items – массив товаров.

onShowItem – функция, которая вызывается при клике на товар (открывает карточку товара).

onAdd – функция, которая добавляет товар в корзину.

```
<main>
    {items.map(el => (
    <Item
    key={el.id}
    item={el}
    onShowItem={onShowItem}
    onAdd={onAdd}
    />
))}
</main>
```

<main> – используется как контейнер для списка товаров.

- items.map(el => (...)) – перебираем массив товаров (items) и создаём компонент <Item /> для каждого товара.

Внутри <Item /> передаём свойства (props):

key={el.id} – уникальный ключ.

item={el} – передаём сам товар (el содержит title, text, category, price, img).

onShowItem={onShowItem} – передаём функцию, которая срабатывает при клике на товар.

onAdd={onAdd} – функция, добавляющая товар в корзину.

Пример работы map:

Например, items выглядит так:

```
[
 { id: 1, img: "img/141.jpg", title: "ПЛАТЬЕ МАКСИ В АЛОМ ЦВЕТЕ", text: "Летящее игривое, оно станет вашей желанной одеждой", category: "school", price: "5000₽" },
{ id: 2, img: "img/33.jpg", title: "ГЛАВНЫЙ ДЕНЬ", text: "Платье свадебное из кружева с фигурным силуэтом", category: "wedding", price: "4500₽" } ]
```

В таком случае React создаст:

```
<Item key={1} item={{ id: 1, img: "img/141.jpg", title: "ПЛАТЬЕ МАКСИ В АЛОМ ЦВЕТЕ", text: "Летящее игривое, оно станет вашей желанной одеждой", category: "school", price: "5000₽", img: "jacket.jpg"price: "5000₽" }} />
<Item key={2} item={{ id: 2, timg: "img/33.jpg", title: "ГЛАВНЫЙ ДЕНЬ", text: "Платье свадебное из кружева с фигурным силуэтом", category: "wedding", price: "4500₽" }} />
```

Экспорт компонента:

```
export default Items;
```

Делаем экспорт, чтобы этот компонент можно было использовать в других файлах .

Таким образом компонент Items выполняет следующие функции:

- Получает список товаров (items) через props.

- Перебирает массив items и создаёт компонент <Item /> для каждого товара.

- Передаёт в <Item /> нужные свойства (item, onShowItem, onAdd).

- Выводит все товары внутри <main>.

3.8 Описание компонента Order.js

React-компонент Order отображает элемент заказа в виде карточки с изображением, названием, ценой и кнопкой удаления.

Импорт необходимых модулей

```
import React, { Component } from 'react'
import { FaTrash } from "react-icons/fa";
```

React, { Component } — импортируется React и базовый класс Component для создания классового компонента.

FaTrash — иконка корзины (мусорной корзины) из библиотеки react-icons/fa.

Объявление и рендеринг компонента

```
export class Order extends Component {
  render() {
    return (
      <div className='item'>
        <img
          src={process.env.PUBLIC_URL + "/" + this.props.item.img}
          alt={this.props.item.alt || "img"}
        />
        <p>{this.props.item.title}</p>
```

```
        <b>{this.props.item.price}</b>
        <FaTrash className='delete-icon' onClick={() => this.props.onDelete(this.props.item.id)} />
      </div>
    )
  }
}
```

Компонент Order наследует Component и переопределяет метод render() который отвечает за отрисовку интерфейса.

Получение данных через this.props.item:

this.props.item.img — путь к изображению товара.

this.props.item.alt — альтернативный текст (если alt нет, используется "img").

this.props.item.title — название товара.

this.props.item.price — цена товара.

Обработчик удаления

```
<FaTrash className='delete-icon' onClick={() => this.props.onDelete(this.props.item.id)} />
```

Это иконка корзины (<FaTrash />).

При клике на нее вызывается this.props.onDelete(this.props.item.id), передавая id товара в функцию onDelete, которую родительский компонент должен передавать в props.

Экспорт

```
export default Order
```

Экспорт компонента по умолчанию (default), чтобы его можно было импортировать в других файлах.

Пример использования компонента.

Допустим, у нас есть список товаров, и мы хотим использовать компонент Order:

```
import React from 'react'
import Order from './Order'
const orders = [
  { id: 1, img: "product1.jpg", title: "Товар 1", price: "$10" },
  { id: 2, img: "product2.jpg", title: "Товар 2", price: "$20" },
]
const App = () => {
const handleDelete = (id) => {
console.log(`Удалить товар с ID: ${id}`)
  }
  return (
    <div>
      {orders.map((item) => (
        <Order key={item.id} item={item} onDelete={handleDelete} />
      ))}
    </div>
  )
}
```

export default App

Данный компонент предназначен для выполнения следующих действий:

Order рендерится для каждого товара.

При клике на корзину вызывается handleDelete, которая выводит id товара в консоль.

3.9 Описание компонента Register.js

Этот компонент Register представляет собой форму регистрации для интернет-магазина, созданную с использованием React и хуков (useState).

Импортируется React и useState — хук для управления состоянием формы.

```
import React, { useState } from "react";
```

Создание компонента и состояние

```
const [formData, setFormData] = useState({
  firstName: "",
  lastName: "",
  gender: "",
  email: "",
  password: ""
});
```

formData — объект, хранящий данные формы: имя, фамилия, пол, email и пароль.

setFormData — функция для обновления formData.

useState({...}) — начальное состояние (поля пустые).

Функция для обработки изменений в полях

```
const handleChange = (e) => {
  setFormData({ ...formData, [e.target.name]: e.target.value });
};
```

Эта функция обновляет соответствующее поле в formData при вводе данных пользователем.

[e.target.name] динамически изменяет нужное поле (firstName, email и т. д.).

e.target.value берет новое значение из поля ввода.

Функция отправки формы

```
const handleSubmit = (e) => {
  e.preventDefault();
  console.log("Submitted Data:", formData);
};
```

e.preventDefault(); предотвращает перезагрузку страницы при отправке формы.

Выводит данные формы в консоль (можно заменить на отправку на сервер).

Разметка формы

```
<div style={{ maxWidth: "400px", margin: "auto", padding: "20px", fontFamily:
"Arial, sans-serif" }}>
```

Ограничивает ширину до 400px, центрирует форму (margin: auto),

добавляет отступ (padding: 20px).

Поля формы

<label>Имя</label>

<input type="text" name="firstName" value={formData.firstName} onChange={handleChange}

required />

- Поле для имени (name="firstName"), связанное с formData.firstName.
- onChange={handleChange} обновляет состояние при изменении.
- required делает поле обязательным.

Выбор пола (radio-кнопки)

<div className="gender" style={{ display: "flex", gap: "10px", alignItems: "center" }}>

<label>Пол:</label>

<input type="radio" name="gender" value="Ж" onChange={handleChange}

required /> <label>Ж</label>

<input type="radio" name="gender" value="M" onChange={handleChange}

required /> <label>M</label>

- Радиокнопки для выбора пола (Ж и M).
- name="gender" делает их взаимозависимыми.
- value="Ж" / value="M" задают значение при выборе.
- onChange={handleChange} обновляет formData.gender.

Поля Email и Пароль

<label>Email</label>

<input type="email" name="email" value={formData.email}

onChange={handleChange} required />

Поле ввода email с валидацией (type="email").

<input

 type="password"

 name="password"

 value={formData.password}

 onChange={handleChange}

 required

 pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"

 title="Пожалуйста, используйте 8 или более символов, включая как минимум 1

цифру и смесь заглавных и строчных букв"

/>

- Поле пароля с паттерном валидации:
  - (?=.*\d) — хотя бы одна цифра.
  - (?=.*[a-z]) — хотя бы одна строчная буква.
  - (?=.*[A-Z]) — хотя бы одна заглавная буква.
  - {8,} — минимум 8 символов.

Кнопка отправки

<button type="submit">Зарегистрироваться</button>

- Нажатие вызывает handleSubmit.

Экспорт компонента

```
export default Register;
```

• Дает возможность импортировать Register в другие файлы.

Компонент Register.js используется в файле App.js:

```
import React from "react";
import Register from "./Register";
const App = () => {
  return (
    <div>
      <h1>Регистрация в магазине</h1>
      <Register />
    </div>
  );
};
export default App;
```

Этот компонент выполняет следующие функции:

- Отображение формы регистрации

- Позволяет пользователю ввести имя, фамилию, пол, email и пароль.

- Реагирует на изменения в полях и сохраняет их в useState.

- Валидирует пароль по требованиям безопасности.

- При нажатии "Зарегистрироваться" отправляет данные в консоль.

3.10 Описание компонента ShowFullItem.js

Этот React-компонент отображает развернутый вид товара с подробной информацией и возможностью добавления в корзину.

Импорт необходимых модулей

```
import React, { Component } from 'react'
```

• Импортируется React и базовый класс Component, так как это классовый компонент.

Создание компонента

```
export class ShowFullItem extends Component {
  render() {
    return (
      <div className='full-item'>
```

• ShowFullItem — классовый компонент, который получает props от родителя.

• Оборачивается в div с классом 'full-item'.

Отображение изображения товара

```
<img
  src={"./" + this.props.item.img}
  onClick={() => this.props.onShowItem(this.props.item)}
  alt={this.props.item.alt || "img"}
/>
```

• Выводит изображение товара.

• src={"./" + this.props.item.img} — путь к изображению (файл находиться в папке src).

• Обработчик onClick:

При клике на изображение вызывается this.props.onShowItem(this.props.item).

Этот метод закрывает/сворачивает развернутый товар.

Отображение информации о товаре

{this.props.item.title}

{this.props.item.text}

{this.props.item.property}

{this.props.item.price}

- this.props.item.title — название товара.
- this.props.item.text — описание товара.
- this.props.item.property — дополнительные характеристики
- this.props.item.price — цена (жирный текст).

Кнопка добавления в корзину

<div className="add-to-cart" onClick={() => this.props.onAdd(this.props.item)}>+

- className="add-to-cart" — стилизуется как кнопка.
- onClick={() => this.props.onAdd(this.props.item)}:
  - При клике вызывается onAdd, передавая весь объект item.
  - Эта функция добавляет товар в корзину.

Экспорт компонента

export default ShowFullItem

- Позволяет использовать компонент в других файлах.

В файле App.js есть список товаров и функции onShowItem и onAdd:

import React, { useState } from "react";

```
import ShowFullItem from "./ShowFullItem";

const App = () => {
  const [selectedItem, setSelectedItem] = useState(null);
  const items = [
    { id: 1, img: "product1.jpg", title: "Товар 1", text: "Описание", property: "Цвет:
красный", price: "$100" },
    { id: 2, img: "product2.jpg", title: "Товар 2", text: "Описание", property: "Цвет:
синий", price: "$120" }
  ];
  const handleShowItem = (item) => {
    setSelectedItem(item); // Развернуть товар
  };
  const handleAddToCart = (item) => {
    console.log("Добавлено в корзину:", item);
  };
  return (
    <div>
      {selectedItem && <ShowFullItem item={selectedItem}
onShowItem={handleShowItem} onAdd={handleAddToCart} />}
    </div>
  );
};
```

export default App;

ShowFullItem выполняет следующие функции:

- Выводит изображение, название, описание, категорию и цену товара.

- Позволяет свернуть товар (клик по изображению вызывает onShowItem).

- Добавляет товар в корзину (onClick на + вызывает onAdd).

Этот компонент используется для отображения карточки товара в полном размере.

Заключение.

В ходе выполнения дипломного проекта была разработана и реализована современная платформа интернет-магазина на основе React. Проект продемонстрировал возможности использования технологий фронтенд-разработки для создания удобного и функционального веб-приложения.

Использование React дало ряд преимуществ, таких как высокая производительность, удобство работы с компонентами и их повторное использование, а также эффективное управление состоянием приложения. Благодаря виртуальному DOM обновления страниц происходят быстрее, что положительно сказывается на пользовательском опыте.

Кроме того, React обеспечивает поддержку современного подхода к разработке, позволяя легко интегрировать сторонние библиотеки и сервисы. Это делает приложение гибким, масштабируемым и удобным в дальнейшей поддержке.

Разработка интернет-магазина с использованием React позволила создать интуитивно понятный интерфейс, оптимизированный для различных устройств, что особенно важно в условиях активного развития мобильной коммерции. Итоговый продукт обладает всеми необходимыми функциями для удобной работы пользователей, а также возможностью дальнейшего расширения функционала.

Таким образом, данный дипломный проект подтвердил актуальность использования React в разработке интернет-магазинов и показал

перспективность его дальнейшего применения в сфере eCommerce.

Список используемой литературы:

1.Руководство для начинающих по HTML и CSS Пошаговое руководство с

примерами и упражнениями — Кевин Уилсон

2. Martine Dowden Michael Gearon Tiny.CSS.Projects

3. Кириченко А.В, Дубовик Е.В. Справочник HTML КРАТКО # БЫСТРО # ПОД

РУКОЙ

4.https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD

5.https://integrion.biz/articles/zachem-nuzhen-internet-magazin

6.https://selectel.ru/blog/http-request/#:~:text=%D0%97%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D1%8B%20(HTTP%20Requests)%20%E2%80%94%20%D1%81%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D1%8F,%D0%B2%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82%20%D0%BD%D0%B0%20%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%81%D0%BA%D0%B8%D0%B9%20%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81.

7.https://official.satbayev.university/download/document/11006/%D0%91%D0%B0%D0%B9%D0%B4%D1%83%D0%BB%D0%BE%D0%B2%20%D0%90.%D0%A0.%20%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0%20%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82-%D0%BC%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD%D0%B0%20%D1%81%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC%20CMS%20WordPress%202019.pdf

8. https://codebra.ru/ru/lessons-css/selectors/12/1

9.https://ru.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF%D0%BE%D0%B3%D1%80%D0%B0%D1%84

10.https://community.exolve.ru/blog/35-browser-plugins-for-chrome-to-help-developers/

11. https://images.app.goo.gl/YA2HLhYNwmyj2oYE9

Приложения

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<link rel="icon" href="%PUBLIC_URL%/favicon.ico" />

<meta name="viewport" content="width=device-width, initial-scale=1" />

<meta name="theme-color" content="#000000" />

<meta name="description" content="Web site created using create-react-app" />

<link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />

<title>React App</title>

</head>

<body>

<noscript>You need to enable JavaScript to run this app.</noscript>

<div id="root"></div>

</body>

</html>
```

Categories.js

```
import React from "react";

const Categories = ({ chooseCategory }) => {

const categories = [
```

```
{ key: "all", name: "Всё" },
{ key: "school", name: "Выпускной" },
{ key: "wedding", name: "Свадьба" },
{ key: "holiday", name: "Отдых" },
{ key: "celebration", name: "Торжество" },
{ key: "concert", name: "Концерт" },
{ key: "party", name: "Вечеринка" }
];
return (
<div className="categories">
{categories.map((el) => (
<div key={el.key} onClick={() => chooseCategory(el.key)}>
{el.name}
</div>
))}
</div>
);
};
export default Categories;
```

Footer.js

```
import React from 'react'
export default function Footer() {
```

```
return (
<footer>Все права защищены &copy;</footer>
)
}
```

Header.js

```
import React, { useState } from 'react'
import Order from './Order'
import { IoIosBasket } from "react-icons/io";
import { Link } from 'react-router-dom'; // Импорт Link
const showOrders = (props) => {
let summa = 0
props.orders.forEach(el => summa += Number.parseFloat(el.price))
return (<div>
{props.orders.map(el => (
<Order onDelete={props.onDelete} key={el.id} item={el} />
))}
<p className='summa'>Сумма: {new Intl.NumberFormat().format(summa)}₽</p>
</div>)
}
const showNothing = () => {
return (<div className='empty'>
```

```
<h2>Товаров нет</h2>

</div>)

}

const Header = (props) => {

let [cartOpen, setCartOpen] = useState(false);

return (

<header>

<div>

<span className="logo">women's stories</span>

<ul className='nav'>

<li><Link to="/" style={{ cursor: 'pointer' }}>Главная</Link></li>

<li>Контакты</li>

<li>

<Link to="/register" style={{ cursor: 'pointer' }}>Кабинет</Link>

</li>

</ul>

<IoIosBasket onClick={() => setCartOpen(prevState => !prevState)} className={`shop-cart-button
${cartOpen && 'active'}`} />

{cartOpen && (

<div className='shop-cart'>

{props.orders.length > 0 ? showOrders(props) : showNothing()}

</div>

)}
```

```
</div>

<div className='presention'></div>

</header>

);

};

export default Header;
```

Item.js

```
import React, { Component } from "react";

export class Item extends Component {

render() {

return (

<div className="item">

<img

src={process.env.PUBLIC_URL + "/" + this.props.item.img}

onClick={() => this.props.onShowItem(this.props.item)}

alt={this.props.item.alt || "img"}

/>

<p><strong>{this.props.item.title}</strong></p>

<p>{this.props.item.text}</p>

<p>{this.props.item.property}</p>

<b>{this.props.item.price}</b>

<div className="add-to-cart" onClick={() => this.props.onAdd(this.props.item)}>+</div>
```

```
</div>
);
}
}
export default Item;
```

Items.js

```
import React from 'react';
import Item from './Item';
const Items = ({ items, onShowItem, onAdd }) => {
return (
<main>
{items.map(el => (
<Item
key={el.id}
item={el}
onShowItem={onShowItem}
onAdd={onAdd}
/>
))}
</main>
);
};
```

```
export default Items;
```

Order.js

```
import React, { Component } from 'react'
import { FaTrash } from "react-icons/fa";
export class Order extends Component {
render() {
return (
<div className='item'>
<img
src={process.env.PUBLIC_URL + "/" + this.props.item.img}
alt={this.props.item.alt || "img"}
/>
<p>{this.props.item.title}</p>
<b>{this.props.item.price}</b>
<FaTrash className='delete-icon' onClick={() => this.props.onDelete(this.props.item.id)} />
</div>
)
}
}
export default Order
```

Register.js

```
import React, { useState } from "react";
```

```
const Register = () => {

const [formData, setFormData] = useState({

firstName: "",

lastName: "",

gender: "",

email: "",

password: ""

});

const handleChange = (e) => {

setFormData({ ...formData, [e.target.name]: e.target.value });

};

const handleSubmit = (e) => {

e.preventDefault();

console.log("Submitted Data:", formData);

};

return (

<div style={{ maxWidth: "400px", margin: "auto", padding: "20px", fontFamily: "Arial, sans-serif"
}}>

<h2>Заполните ваши данные</h2>

<form onSubmit={handleSubmit}>

<label>Имя</label>

<input type="text" name="firstName" value={formData.firstName} onChange={handleChange}
required />
```

<label>Фамилия</label>

<input type="text" name="lastName" value={formData.lastName} onChange={handleChange} required />

<div className="gender" style={{ display: "flex", gap: "10px", alignItems: "center" }}>

<label>Пол:</label>

<input type="radio" name="gender" value="Ж" onChange={handleChange} required /> <label>Ж</label>

<input type="radio" name="gender" value="М" onChange={handleChange} required /> <label>М</label>

<h3>Login details</h3>

<label>Email</label>

<input type="email" name="email" value={formData.email} onChange={handleChange} required />

<label>Password</label>

<input

type="password"

name="password"

value={formData.password}

onChange={handleChange}

required

pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"

title="Пожалуйста, используйте 8 или более символов, включая как минимум 1 цифру и

смесь заглавных и строчных букв"

/>

<small>Пожалуйста, используйте 8 или более символов, включая как минимум 1 цифру и

смесь заглавных и строчных букв</small>

<button type="submit">Зарегистрироваться</button>

</form>

);

};

export default Register;

ShowFullItem.js

import React, { Component } from 'react'

export class ShowFullItem extends Component {

render() {

return (

<img

src={"./" + this.props.item.img} onClick={() => this.props.onShowItem(this.props.item)}

alt={this.props.item.alt || "img"}

/>

{this.props.item.title}

{this.props.item.text}

```
<p>{this.props.item.property}</p>

<b>{this.props.item.price}</b>

<div className="add-to-cart" onClick={() => this.props.onAdd(this.props.item)}>+</div>

</div>

</div>

)

}

}

export default ShowFullItem
```

App.js

```
import React from 'react';

import Header from "./components/Header";

import Footer from "./components/Footer";

import Items from "./components/Items";

import Categories from "./components/Categories";

import ShowFullItem from "./components/ShowFullItem";

import { Routes, Route } from 'react-router-dom';

import Register from './components/Register';

class App extends React.Component {

constructor(props) {

super(props)

this.state = {
```

```
orders: [],
currentItems: [],
items: [
{ id: 1, img: "img/141.jpg", title: "ПЛАТЬЕ МАКСИ В АЛОМ ЦВЕТЕ", text: "Летящее игривое, оно станет вашей желанной одеждой", category: "school", price: "5000₽" },
{ id: 2, img: "img/33.jpg", title: "ГЛАВНЫЙ ДЕНЬ", text: "Платье свадебное из кружева с фигурным силуэтом", category: "wedding", price: "4500₽" },
{ id: 3, img: "img/91.jpg", title: "СКОРО В ОТПУСК", text: "Платье из шелка в оранжевом оттенке свободного силуэта", category: "holiday", price: "4200₽" },
{ id: 4, img: "img/231.jpg", title: "ПРАЗНИК К НАМ ПРИШЕЛ", text: "Платье длинное бархатное в бордовом цвете", category: "celebration", price: "6000₽" },
{ id: 5, img: "img/22.jpg", title: "ИДЕМ НА КОНЦЕРТ", text: "Платье Vivian серо серебристого оттенка", category: "concert", price: "6100₽" },
{ id: 6, img: "img/622.jpg", title: "НА ВЕЧЕРИНКУ", text: "Маленькое черное платье облегающего силуэта", category: "party", price: "5300₽" },
// остальные продукты
],
showFullItem: false,
fullItem: {}
}
this.state.currentItems = this.state.items
this.addToOrder = this.addToOrder.bind(this)
```

```
this.deleteOrder = this.deleteOrder.bind(this)

this.chooseCategory = this.chooseCategory.bind(this)

this.onShowItem = this.onShowItem.bind(this)

}

render() {

return (

<div className="wrapper">

<Header orders={this.state.orders} onDelete={this.deleteOrder} />

<Routes>

<Route path="/" element={

<>

<Categories chooseCategory={this.chooseCategory} />

<Items onShowItem={this.onShowItem} items={this.state.currentItems}

onAdd={this.addToOrder} />

{this.state.showFullItem && <ShowFullItem onAdd={this.addToOrder}

onShowItem={this.onShowItem} item={this.state.fullItem} />}

</>

} /> {/* Этот маршрут для страницы регистрации */}

<Route path="/register" element={<Register />} /> {/* Этот маршрут для страницы регистрации

*/}

</Routes>

<Footer />
```

```
</div>

)

}

onShowItem(item) {

this.setState({ fullItem: item })

this.setState({ showFullItem: !this.state.showFullItem })

}

chooseCategory(category) {

if (category === 'all') {

this.setState({ currentItems: this.state.items })

return

}

this.setState({

currentItems: this.state.items.filter(el => el.category === category)

})

}

deleteOrder(id) {

this.setState({ orders: this.state.orders.filter(el => el.id !== id) })

}

addToOrder(item) {

let isInArray = false

this.state.orders.forEach(el => {
```

```
if (el.id === item.id)
isInArray = true
})
if (!isInArray)
this.setState({ orders: [...this.state.orders, item] })
}
}
export default App;
```

index.css

```
@importurl('https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,100..900;1,100..900&d
splay=swap');
* {
margin: 0;
padding: 0;
}
body {
background: #fff;
color: #222;
font-family: 'Roboto:ital', sans-serif;
font-weight: 300;
}
.wrapper {
```

```
width: 1280px;

margin: 50px auto;

}

header {

position: relative;

}

header .logo {

font-weight: 600;

font-size: 26px;

}

header ul.nav {

float: right;

list-style: none;

}

header ul.nav li {

display: inline;

margin-left: 25px;

cursor: pointer;

transition: oposity 500ms ease;

}

header ul.nav li:hover {

opacity: 0.5;
```

```
}
header .presention {
margin: 50px 0;
background: url('./img/44.jpg') no-repeat;
width: 100%;
height: 700px;
background-position: center center;
background-blend-mode: multiply;
background-color: rgb(228, 222, 214);
position: relative;
}
header .presention::after {
content: "Лучшие товары";
position: absolute;
top: 100px;
left: 50px;
width: 300px;
font-size: 40px;
font-weight: 600;
color: #fff;
}
header .presention::before {
```

```
content: "на все случаи";
position: absolute;
top: 200px;
left: 50px;
width: 300px;
font-size: 18px;
font-weight: 300;
color: #fff;
}
header .shop-cart {
position: absolute;
top: 30px;
right: 0;
width: 450px;
padding: 20px;
background: #fafafa;
box-shadow: 4px 5px 15px -7px #606060;
z-index: 1000;
padding-bottom: 0;
}
header .shop-cart-button {
float: right;
```

```
cursor: pointer;
transition: color 500ms ease;
}
header .shop-cart-button:hover,
header .shop-cart-button:active {
color: #dc3d3d;
}
header .shop-cart .item {
width: 100%;
float: left;
margin-bottom: 20px;
}
header .shop-cart .item .delete-icon {
color: #ca5252;
float: right;
position: relative;
top: -25px;
cursor: pointer;
transition: color 500ms ease;
}
header .shop-cart .item .delete-icon:hover {
color: #d83030;
```

```
transform: scale(1.5);
}
header .shop-cart .empty .h2 {
font-size: 20px;
margin-bottom: 20px;
}
header .shop-cart .item .h2 {
font-size: 20px;
margin-bottom: 20px;
}
header .shop-cart .item .b {
color: #797979;
font-weight: 600;
}
header .shop-cart .item img {
width: 70px;
float: left;
margin-right: 20px;
}
header .shop-cart .summa {
float: left;
width: 100%;
```

```
font-weight: 600;

font-size: 20px;

margin-bottom: 20px;

}

.categories div {

display: inline-block;

background: #f2f2f2;

border-radius: 50px;

padding: 10px 20px;

margin-bottom: 25px;

margin-right: 15px;

cursor: pointer;

border: 1px solid transparent;

transition: all 500ms ease;

font-weight: 600;

}

.categories div:hover {

border-color: silver;

transform: scale(1.1);

}

.full-item {

position: fixed;
```

```
top: 0;

left: 0;

right: 0;

bottom: 0;

background: rgba(0, 0, 0, 0.8);

z-index: 9999;

overflow: auto;

}

.full-item>div {

width: 600px;

position: relative;

margin: 10% auto;

padding: 40px 30px;

background: #fff;

border-radius: 9px;

}

main {

display: flex;

/* width: 100px; */

flex-wrap: wrap;

justify-content: space-between;

}
```

```
main .item {
width: 30%;
margin-bottom: 100px;
background: rgb(230, 227, 223);
position: relative;
padding-bottom: 20px;
}
main .item img,
.full-item img {
width: 100%;
border-radius: 10px 10px 0 0;
transition: transform 500ms ease;
cursor: pointer;
}
main .item img:hover,
.full-item img:hover {
transform: scale(1.05);
}
main h2,
.full-item h2,
main p,
.full-item p {
```

```
margin: 10px 20 px;

color: #333;

}

main b,

.full-item b {

color: red;

margin: 20px;

font-size: 20px;

}

p {

font-size: 20px;

margin: 20px;

}

main .add-to-cart,

.full-item .add-to-cart {

position: absolute;

right: 20px;

bottom: 20px;

background: #ca5252;

width: 35px;

height: 35px;

text-align: center;
```

```
line-height: 35px;
color: #fff;
border-radius: 50%;
cursor: pointer;
font-weight: 600;
transition: transform 500ms ease;
}
main .add-to-cart:hover,
.full-item .add-to-cart:hover {
transform: scale(1.5) translateY(-5px);
}
footer {
text-align: center;
margin-top: 100px;
}
body {
font-family: Arial, sans-serif;
/* max-width: 400px; */
margin: auto;
padding: 20px;
}
label,
```

```
input {
display: block;
width: 100%;
margin-bottom: 10px;
}
.gender {
display: flex;
gap: 10px;
align-items: center;
}
button {
background-color: #4CAF50;
color: white;
padding: 10px;
border: none;
cursor: pointer;
}
button:hover {
background-color: #45a049;
}
h3 {
margin-bottom: 20px;
```

```
}

button {

margin-top: 20px;

font-size: 16px;

}

h2 {

margin-bottom: 20px;

}

small {

font-size: 14px;

}
```

index.js

```
import React from 'react';

import ReactDOM from 'react-dom/client';

import { BrowserRouter } from 'react-router-dom'; // Импортируем BrowserRouter

import './index.css';

import App from './App';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<BrowserRouter> {/* Оборачиваем в Router */}

<React.StrictMode>

<App />
```

```
</React.StrictMode>

</BrowserRouter>

);
```