**МИНИСТЕРСТВО ОБРАЗОВАНИЯ ТУЛЬСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ТУЛЬСКОЙ ОБЛАСТИ   
«ТУЛЬСКИЙ ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»**

**Курсовой проект**

**по МДК 09.01. Проектирование и разработка веб-приложений**

Тема: **Разработка клиентской и серверной части интернет-магазина по продаже автомобильных шин**

Специальность: 09.02.07 Информационные системы и программирование

Группа: **411-П**

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.А. Харитонов

(подпись, дата)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.И. Каргина

(подпись, дата)

Щекино, 2025

**МИНИСТЕРСТВО ОБРАЗОВАНИЯ ТУЛЬСКОЙ ОБЛАСТИ**

**ГОСУДАРСТВЕННОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ТУЛЬСКОЙ ОБЛАСТИ   
«ТУЛЬСКИЙ ЭКОНОМИЧЕСКИЙ КОЛЛЕДЖ»**

**Индивидуальное задание**

**по выполнению курсового проекта   
по МДК 09.01. Проектирование и разработка веб-приложений**

студенту **Харитонову Денису Алексеевичу**

Тема курсового проекта: Разработка клиентской и серверной части интернет-магазина по продаже автомобильных шин

**Структура (содержание) курсового проекта**

ВВЕДЕНИЕ

1. АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Использование интернет-технологий для развития бизнеса

1.2 Постановка задачи. Техническое задание на разработку веб-приложения

1.3 Выбор и обоснование средств разработки

1.3.1 Обоснование выбора средств разработки клиентской части

1.3.2 Обоснование выбора СУБД

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Разработка алгоритма работы веб-приложения.

2.2 Архитектура проектируемого веб-приложения

2.2.1 Проектирование разделов сайта. Карта сайта

2.2.2 Оформление сайта

2.3 Проектирование структуры базы данных

2.4 Тестирование сайта

2.5 Панель администратора

ЗАКЛЮЧЕНИЕ

СПИСОК ЛИТЕРАТУРЫ

ПРИЛОЖЕНИЕ: листинг программы, формы

Графическая часть. Представить структуру сайта.

Список рекомендуемой литературы:

1. Тузовский, А. Ф. Проектирование и разработка web-приложений : учеб. Пособие для СПО / А. Ф. Тузовский. — М. : Издательство Юрайт, 2019. — 218 с.
2. Сысолетин, Е. Г. Разработка интернет-приложений : учеб. пособие для СПО / Е. Г. Сысолетин, С. Д. Ростунцев. — М. : Издательство Юрайт, 2019. — 90 с.
3. Голицына О.Л. и др. Информационные системы и технологии: учебное пособие для среднего профессионального образования / О.Л. Голицына, Н.В. Максимов, И.И. Попов. - Москва: Инфра-М, 2020. – 399с.
4. Диков А.В. Клиентские технологии веб дизайна.HTML5 и CSS3: учебное пособие - Издательство "Лань", 2019. – 188с.
5. Мусаева Т.В. Разработка дизайна веб-приложений: учебник для студентов учреждений среднего профессионального образования. - М.: Академия, 2020. – 256с..
6. Веллинг Л. Разработка WEB-приложений с помощью PHP и MySQL. 2-е издание.: Пер. с англ. / Л. Веллинг - М.: Издательский дом "Вильямс", 2014.-800 с..
7. Глушаков С.В., "Программирование Web-страниц"/С.В.Глушаков, И.А. Жакин, Т.С. Хачиров.- Изд-во "Фолио", М:. 2016. – 390с.
8. Коггзол Дж. PHP 5. Полное руководство. : Пер. С англ. / Дж. Коггзол - М. Издательский дом «Вильямс», 2015.
9. Никсон Р. PHP. Создаем динамические веб-сайты с помощью PHP, MySQL и JavaScript. / Р. Никсон - С-Пб.: "Питер", 2015. - 496 с.
10. Шаньгин В. Ф. Информационная безопасность компьютерных систем и сетей / В.Ф. Шаньгин. - М.: Форум, Инфра-М, 2017. - 416 c.
11. Конверс PHP 5 и MySQL. Библия пользователя / Конверс, др. Т. и. - М.: Вильямс, **2015. - 752** c.

Задание выдал: О.И. Каргина \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О. преподавателя) (подпись) (дата)

Задание принял: Д.А. Харитонов \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О. студента) (подпись) (дата)

|  |  |
| --- | --- |
| ВВЕДЕНИЕ |  |
| 1. АНАЛИТИЧЕСКАЯ ЧАСТЬ |  |
| 1.1 Использование интернет-технологий для развития бизнеса |  |
| 1.2 Постановка задачи. Техническое задание на разработку веб-приложения |  |
| 1.3 Выбор и обоснование средств разработки |  |
| 1.3.1 Обоснование выбора средств разработки клиентской части |  |
| 1.3.2 Обоснование выбора СУБД |  |
| 2. ПРОЕКТНАЯ ЧАСТЬ |  |
| 2.1 Разработка алгоритма работы веб-приложения. |  |
| 2.2 Архитектура проектируемого веб-приложения |  |
| 2.2.1 Проектирование разделов сайта. Карта сайта |  |
| 2.2.2 Оформление сайта |  |
| 2.3 Проектирование структуры базы данных |  |
| 2.4 Тестирование сайта |  |
| 2.5 Панель администратора |  |
| ЗАКЛЮЧЕНИЕ |  |
| СПИСОК ЛИТЕРАТУРЫ |  |
| ПРИЛОЖЕНИЕ: листинг программы, формы |  |
| Графическая часть. Представить структуру сайта. |  |
| Список рекомендуемой литературы: |  |

**ВВЕДЕНИЕ**

Современное общество невозможно представить без использования цифровых технологий, которые проникают практически во все сферы жизни. Одной из наиболее значимых областей, активно использующих IT-технологии, является электронная коммерция. Интернет-магазины занимают ключевую роль в торговле, обеспечивая удобный доступ к товарам и услугам для широкого круга потребителей.

В последние годы наблюдается активный рост спроса на покупку автомобильных шин через интернет. Это связано с ростом числа автовладельцев, необходимостью сезонной замены шин, а также желанием покупателей сэкономить время и деньги, избегая посещения физических магазинов. Тем не менее, многие существующие платформы для продажи шин сталкиваются с рядом проблем, таких как сложность навигации, ограниченные функциональные возможности и низкая производительность. Это делает тему разработки современного, производительного и удобного интернет-магазина актуальной для решения как технических, так и коммерческих задач.

Дополнительно, использование актуального стека технологий, включающего **Next.js**, **Prisma**, **NextAuth**, **Zustand** и **Axios**, позволяет обеспечить не только высокую скорость работы и стабильность платформы, но и гибкость в реализации сложных пользовательских сценариев. Такой подход предоставляет возможность интеграции с внешними сервисами, упрощает разработку и последующее масштабирование проекта, что крайне важно в условиях динамично меняющегося рынка.

Целью данной курсовой работы является разработка интернет-магазина по продаже автомобильных шин, включающего клиентскую и серверную части, с использованием современного стека технологий. Интернет-магазин должен обеспечить пользователям удобство в поиске и покупке товаров, высокую производительность, безопасность, а также возможность масштабирования.  
Для достижения поставленной цели необходимо решить следующие задачи:

1. Провести анализ существующих интернет-магазинов автомобильных шин, выделив их основные преимущества и недостатки.
2. Изучить особенности и преимущества выбранного стека технологий для реализации проекта.
3. Разработать архитектуру проекта, включающую клиентскую и серверную части.
4. Реализовать пользовательскую аутентификацию с использованием **NextAuth**.

Объектом исследования является процесс разработки программных решений для электронной коммерции.  
Предметом исследования являются технологии и подходы, используемые при создании клиентской и серверной частей интернет-магазина.

Для достижения целей работы использовались следующие методы:

1. Аналитический метод для изучения существующих решений в области разработки интернет-магазинов.
2. Сравнительный анализ технологий для выбора наиболее подходящего стека.
3. Метод проектирования для создания архитектуры клиентской и серверной частей.

**1. АНАЛИТИЧЕСКАЯ ЧАСТЬ**

**1.1 Использование интернет-технологий для развития бизнеса**

Современные интернет-технологии играют ключевую роль в развитии бизнеса, предоставляя предпринимателям новые возможности для взаимодействия с клиентами, автоматизации процессов и увеличения доходов. Они позволяют значительно повысить эффективность операций, улучшить взаимодействие с клиентами и расширить географию продаж. В области электронной коммерции интернет-магазины становятся важным инструментом для расширения аудитории, повышения продаж и повышения лояльности клиентов. Они дают возможность бизнесу быть доступным для клиентов круглосуточно и не ограничиваться территориально.  
Сегмент автомобильных шин является особенно привлекательным для реализации через интернет-магазины. Это объясняется несколькими факторами, включая высокий спрос на разнообразие моделей, брендов, а также необходимость выбора шин по индивидуальным характеристикам автомобиля. Онлайн-продажи становятся удобным инструментом для клиентов, так как они могут не только найти нужные товары, но и сравнить цены, прочитать отзывы и сделать выбор без необходимости посещать физические магазины.

Интернет-магазины автомобильных шин позволяют минимизировать расходы на аренду физических площадей, оптимизировать работу склада и улучшить логистику. При этом владельцы магазинов могут обеспечить круглосуточный доступ к товару и мгновенное оформление заказа. Такой подход помогает значительно снизить операционные затраты, а также позволяет избежать необходимости большого штата сотрудников для обслуживания покупателей. Использование современных технологий в разработке интернет-магазинов, таких как Next.js, Prisma и NextAuth, позволяет добиться высокой производительности платформы.

**1.2 Постановка задачи. Техническое задание на разработку веб-приложения**

**React: Популярность и производительность:** React обеспечивает быструю работу интерфейса благодаря виртуальному DOM и компонентному подходу, что идеально подходит для интернет-магазина, требующего высокой скорости и отзывчивости

**Tailwind CSS:**

**Адаптивность:** Tailwind позволяет легко создавать адаптивные интерфейсы, что критично для корректного отображения на разных устройствах.

**Ускоренная стилизация:** Система классов ускоряет процесс разработки без необходимости написания сложных стилей.

**MongoDB:**

**Гибкость схемы:** MongoDB не требует заранее определенной схемы, что упрощает работу с динамическими данными (товары, заказы).

**Масштабируемость:** Идеальна для роста интернет-магазина, поддерживает большие объемы данных.

**Prisma:**

**Удобный ORM:** Prisma упрощает работу с MongoDB и ускоряет разработку, предоставляя удобный интерфейс для взаимодействия с базой данных.

**Next Auth:**

**Безопасность и простота:** Next Auth обеспечивает безопасную аутентификацию, поддерживает различные провайдеры и легко интегрируется с MongoDB, что важно для работы с данными пользователей и заказами.

### 1. **Проектирование интерфейса:**

* **Главная страница:** Должна содержать привлекательную визуальную презентацию товаров, акций и категорий, с возможностью быстрого перехода в каталог, корзину и поиск товаров.
* **Страница "О нас":** Содержит информацию о компании, ее ценностях, а также преимуществах покупки товаров в интернет-магазине.
* **Контакты:** Интерактивная страница с контактными данными, картой, формой обратной связи.
* **Товары:** Каталог с возможностью фильтрации, сортировки и поиска по различным параметрам (цене, бренду, типу товара и т. д.).
* **Корзина:** Страница, где пользователь может просматривать добавленные товары, редактировать количество или удалять их, а также оформлять заказ.
* **Регистрация:** Страница для создания нового аккаунта с авторизацией через email и пароль или через социальные сети (если интеграция предусмотрена).
* **Админ панель:** Интерфейс для администрирования каталога товаров, заказов и управления пользователями.

### 2. **Разработка функционала:**

* **Каталог товаров:** Используя MongoDB и Prisma, можно хранить информацию о товарах в базе данных. Данные должны включать название, описание, цену, изображение и категории товаров.
* **Корзина и оформление заказа:** Для корзины можно создать серверную логику с использованием Next.js API Routes для обработки добавления товаров в корзину и оформления заказа.
* **Система регистрации и аутентификации:** С помощью Next Auth реализуется безопасная авторизация через email и пароль, а также возможность авторизации через социальные сети (Google, Facebook и т. д.). Для хранения данных пользователей можно использовать MongoDB.
* **Платежные и доставочные системы:** Интеграция с API внешних сервисов, таких как Stripe или PayPal, для обработки платежей, а также подключение внешних сервисов для расчета доставки.
* **Административная панель:** Использование React и Tailwind для удобного и интуитивно понятного интерфейса, а также предоставление функционала для администраторов по добавлению, редактированию и удалению товаров и заказов.

### 3. **Тестирование:**

* Проведение тестирования для проверки функционала каждого раздела сайта: корзины, поиска товаров, оформления заказа и других.
* Протестировать корректность отображения сайта на различных устройствах, проверив адаптивность через Tailwind CSS.
* Провести нагрузочное тестирование для проверки работы с большим количеством пользователей и запросов.

### 4. **Запуск проекта:**

* **Хостинг:** Для проекта можно выбрать виртуальный сервер с возможностью масштабирования, такой как Vercel или AWS. Для MongoDB использовать облачное хранилище, например MongoDB Atlas.
* **Безопасность:** Для обеспечения защиты данных пользователей и защиты от DDoS-атак можно применить SSL-сертификаты, настройки брандмауэра и мониторинг трафика.
* **Масштабируемость:** Сервисы должны быть настроены так, чтобы в будущем, если трафик увеличится, их ресурсы можно было легко увеличить.

### 5. **Технические требования:**

* Разработка должна соответствовать принципам безопасности, включая защиту данных пользователей.
* Оптимизация контента для поисковых систем (SEO).
* Адаптивный дизайн с использованием Tailwind CSS, обеспечивающий корректное отображение на всех устройствах.

**1.3 Выбор и обоснование средств разработки**

**1.3.1 Обоснование выбора средств разработки клиентской части**

Для реализации интернет-магазина был выбран актуальный стек технологий, обеспечивающий высокую производительность, надежность и удобство разработки.

Клиентская часть интернет-магазина будет разработана с использованием Next.js, который обеспечивает:

Быструю загрузку страниц за счет серверного рендеринга (SSR) и статической генерации (SSG).

Поддержку маршрутизации из коробки, что упрощает разработку.

Возможность интеграции с современными инструментами для стилизации и управления состоянием, такими как Zustand.

Использование Zustand позволит эффективно управлять состоянием приложения, обеспечивая реактивность и снижение сложности кода. Взаимодействие клиента с сервером будет реализовано с помощью Axios, предоставляющего удобные инструменты для работы с HTTP-запросами.

**1.3.2 Обоснование выбора СУБД**

Для хранения данных о товарах, заказах и пользователях в проекте выбран инструмент Prisma, который представляет собой эффективное решение для работы с реляционными базами данных. Prisma является ORM (Object-Relational Mapping) инструментом, который значительно упрощает работу с базами данных, повышая скорость разработки и снижая сложность взаимодействия с данными. В частности, для этого проекта выбрана база данных PostgreSQL, которая является одной из самых популярных и надежных реляционных систем управления базами данных. Одним из основных преимуществ использования Prisma является его способность работать с различными реляционными базами данных, такими как PostgreSQL, MySQL, SQLite и другими. Это делает Prisma гибким инструментом, который может быть применен в широком спектре проектов и позволяет легко переключаться между различными СУБД без необходимости переписывать значительную часть кода.

Prisma также предоставляет удобную возможность автоматической генерации моделей на основе схемы базы данных. Это значительно ускоряет процесс разработки, поскольку позволяет создавать модели данных с минимальными усилиями. Вместо того чтобы вручную писать код для создания таблиц и связей между ними, Prisma автоматически генерирует необходимую структуру на основе заранее заданных схем, что существенно снижает вероятность ошибок и упрощает поддержку проекта. Еще одним важным преимуществом Prisma является интуитивно понятный синтаксис для написания запросов. Он предоставляет простой и удобный способ взаимодействия с базой данных, что делает его доступным не только для опытных разработчиков, но и для начинающих. Использование Prisma позволяет писать запросы, которые легко читаются и поддерживаются, благодаря чему снижается сложность работы с базой данных, а код становится более прозрачным и надежным. Это особенно важно при работе с большими объемами данных, таких как информация о товарах, заказах и пользователях, где важно поддерживать высокую производительность и точность запросов.

**ПРОЕКТНАЯ ЧАСТЬ**

**2.1 Разработка алгоритма работы веб-приложения.**

### Основные этапы работы

#### 1. Инициализация и загрузка главной страницы

* Пользователь заходит на сайт.
* Сервер обрабатывает запрос и отдает статическую или серверно-рендеренную (SSR) версию главной страницы.
* Клиент получает данные о популярных товарах, акциях и категории шин.

#### 2. Просмотр каталога товаров

* Пользователь переходит на страницу "Товары".
* Отправляется запрос к серверу на получение списка шин.
* Сервер взаимодействует с базой данных MongoDB через Prisma и возвращает отфильтрованный список товаров.
* Клиентская часть рендерит полученные данные, отображая карточки товаров.

#### 3. Просмотр детальной информации о товаре

* При клике на товар пользователь переходит на страницу с подробным описанием.
* Отправляется запрос на сервер для получения полной информации о товаре.
* Сервер извлекает данные из базы и передает их клиенту.
* Пользователь видит характеристики шин, цену, доступность и возможность добавления в корзину.

#### 4. Добавление товара в корзину

* Пользователь нажимает кнопку "Добавить в корзину".
* Товар добавляется в локальное состояние клиента или в базу данных при авторизованном сеансе.
* Если пользователь зарегистрирован, данные корзины сохраняются в базе и синхронизируются между устройствами.

#### 5. Регистрация и авторизация

* Пользователь переходит на страницу "Регистрация" и вводит данные.
* Запрос передается на сервер, где NextAuth обрабатывает регистрацию.
* Если регистрация успешна, создается новая запись в базе данных и пользователь перенаправляется в личный кабинет.
* При входе в аккаунт происходит аутентификация через NextAuth, возвращается JWT-токен, который используется для последующих запросов.

#### 6. Оформление заказа

* Пользователь переходит в корзину, проверяет товары и нажимает "Оформить заказ".
* Отправляется запрос на сервер для формирования заказа.
* Сервер проверяет наличие товаров, вычисляет сумму и создаёт запись заказа в базе.
* Если требуется оплата, пользователь перенаправляется на платёжный шлюз.

#### 7. Администрирование (Админ-панель)

* Администратор входит в систему через защищенный маршрут.
* Получает доступ к управлению товарами, заказами и пользователями.
* Может добавлять, редактировать и удалять товары, а также отслеживать заказы.

**2.2 Архитектура проектируемого веб-приложения**

Архитектура приложения включает в себя клиентскую и серверную части, взаимодействующие между собой посредством API. Основные технологии: React, Tailwind CSS, Prisma, MongoDB, Next.js, NextAuth, React Icons.

#### 1. Основные модули и их функции:

1. **Главная страница** – отображает популярные товары, акции, содержит навигацию по магазину.
2. **О нас** – содержит информацию о компании, её деятельности и преимуществах.
3. **Контакты** – предоставляет контактные данные и форму обратной связи.
4. **Каталог товаров** – список автомобильных шин с возможностью фильтрации, поиска и сортировки.
5. **Карточка товара** – содержит детальное описание товара, цену, изображения и возможность добавления в корзину.
6. **Корзина** – отображает список добавленных товаров, позволяет изменять количество, удалять товары и оформлять заказ.
7. **Регистрация / Вход** – реализована аутентификация и авторизация через NextAuth с поддержкой JWT-токенов.
8. **Админ-панель** – управление товарами, пользователями и заказами. Доступна только администраторам.

#### 2. Взаимодействие между компонентами

* Клиентская часть (React + Next.js) отправляет запросы к API-серверу для получения данных о товарах, пользователях, корзине и заказах.
* Серверная часть (Next.js API Routes) обрабатывает запросы, взаимодействует с базой данных через Prisma и возвращает ответы в формате JSON.
* NextAuth.js обеспечивает авторизацию пользователей, храня учетные данные и доступы в базе MongoDB.
* MongoDB служит основным хранилищем данных, в котором хранятся:
* **Пользователи** (id, имя, email, хешированный пароль, роль).
* **Товары** (id, название, описание, цена, изображения, остатки на складе, категории).
* **Корзина** (id пользователя, список товаров, общая сумма заказа).
* **Заказы** (id пользователя, список товаров, статус, дата оформления).

#### 3. Backend (Next.js API Routes)

API-роуты обрабатывают запросы и выполняют CRUD-операции через Prisma ORM:

* /api/auth – регистрация, вход, выход пользователя (NextAuth.js).
* /api/products – управление товарами (получение списка, добавление, обновление, удаление).
* /api/cart – управление корзиной (добавление, удаление, изменение количества товаров).
* /api/orders – управление заказами (создание, получение истории заказов).

#### 4. Клиентская часть (React + Next.js)

Использует Tailwind CSS для стилизации, React Icons для графических элементов. Компоненты:

* **Header и Footer** – глобальная навигация и информация о магазине.
* **Карточка товара** – отображает информацию о шине, кнопки добавления в корзину.
* **Формы аутентификации** – регистрация, вход, восстановление пароля.
* **Модальные окна** – для оформления заказа, подтверждения действий.

#### 5. Аутентификация (NextAuth + JWT)

* Регистрация нового пользователя – создание записи в базе, хеширование пароля, генерация JWT-токена.
* Авторизация – проверка учетных данных, выдача токена.
* Middleware – проверка прав доступа к защищенным маршрутам.

#### 6. Дополнительные модули

* Валидация данных на стороне клиента и сервера.
* Управление состоянием (React Context API/Redux).
* Типизация с использованием TypeScript.
* Переменные окружения (dotenv) для хранения конфигурационных данных.

**2.2.1 Проектирование разделов сайта. Карта сайта**

При разработке веб-приложения интернет-магазина автомобильных шин важно продумать удобную навигацию, обеспечивающую быстрый доступ пользователей к нужной информации. Для этого был разработан перечень ключевых разделов сайта и их взаимосвязь.

#### Основные разделы сайта

Веб-приложение состоит из следующих страниц:

1. **Главная** – основная страница сайта, отображает популярные товары, акции и навигацию по магазину.
2. **О нас** – информация о компании, её миссии и преимуществах.
3. **Контакты** – содержит контактные данные, форму обратной связи и карту с местоположением.
4. **Товары** – каталог автомобильных шин с возможностью поиска, фильтрации и сортировки.
5. **Карточка товара** – детальная информация о выбранной шине, цена, изображения и кнопка добавления в корзину.
6. **Корзина** – отображает добавленные товары, позволяет изменять их количество и оформлять заказ.
7. **Регистрация / Вход** – система аутентификации пользователей с использованием NextAuth.
8. **Админ-панель** – защищённый раздел для управления товарами, пользователями и заказами.

#### Карта сайта

Структура страниц веб-приложения представлена в следующем виде:

├── Главная

├── О нас

├── Контакты

├── Товары

│ ├── Поиск

│ ├── Фильтрация

│ ├── Карточка товара

├── Корзина

├── Регистрация / Вход

├── Админ-панель

├── Управление товарами

├── Управление пользователями

├── Управление заказами

**2.2.2 Оформление сайта**

Цели оформления сайта

1. Обеспечение удобного и интуитивно понятного интерфейса — пользователи должны легко находить нужные товары, а также быстро оформлять заказы, минимизируя количество шагов для совершения покупки.
2. Создание современного адаптивного дизайна — сайт должен корректно отображаться на разных устройствах (ПК, планшеты, мобильные), обеспечивая комфортное взаимодействие на любых экранах.
3. Использование лаконичного и профессионального стиля — оформление должно соответствовать тематике автомобильных шин, быть строгим и современным, с акцентом на практичность и функциональность.
4. Оптимизация скорости загрузки страниц — для повышения удобства использования, сайт должен загружаться быстро, что достигается за счет минимизации лишних анимаций и избыточной графики.
5. Поддержка темной и светлой темы (опционально) — возможность переключения цветовой схемы в зависимости от предпочтений пользователя, улучшая удобство использования.

* Задачи оформления сайта:  
  Реализовать стилизацию с использованием Tailwind CSS для удобства работы с классами и быстрого прототипирования.  
  Поддерживать адаптивность с помощью flexbox и grid, чтобы элементы корректно располагались на разных устройствах.  
  Разработать единый стиль для всех страниц, включая шрифты, цвета, кнопки и формы, обеспечивая визуальное единство и удобство восприятия.  
  Использовать React Icons для добавления наглядных значков, улучшая интерфейс и пользовательский опыт.  
  Проработать взаимодействие элементов, включая анимации кнопок, эффектов при наведении и плавного появления контента.
* Основные принципы оформления  
  Цветовая палитра: Использование нейтральных темных оттенков, таких как черный и серый, с яркими акцентами, например, желтым или синим, для выделения ключевых элементов (кнопки, ссылки).  
  Шрифты: Использование sans-serif шрифтов, таких как Inter или Roboto, для улучшения читаемости и обеспечения современного вида.  
  Адаптивность: Поддержка мобильной и десктопной версии с корректной расстановкой элементов, чтобы пользователи на разных устройствах могли комфортно взаимодействовать с сайтом.  
  Минимализм: Меньше лишних элементов, простота и интуитивность интерфейса. Кнопки и формы должны быть легко различимы и доступными для взаимодействия.

Данный подход к оформлению позволяет не только создать привлекательный, но и функциональный интерфейс для интернет-магазина, что улучшает взаимодействие пользователей с сайтом и способствует росту продаж.

**2.3 Проектирование структуры базы данных**

### 1. **Описание сущностей и их атрибутов:**

* **Пользователи (Users)**: информация о зарегистрированных пользователях.
  + Поля:
    - id (уникальный идентификатор)
    - email
    - password (хешированный)
    - role (например, пользователь или администратор)
    - address (адрес для доставки)
    - phone (номер телефона)
    - created\_at (дата регистрации)
* **Товары (Products)**: информация о шинах.
  + Поля:
    - id (уникальный идентификатор)
    - name (название товара)
    - brand (бренд)
    - description (описание)
    - price (цена)
    - stock (количество в наличии)
    - category (категория, например, летние, зимние)
    - image (ссылка на изображение)
* **Корзина (Cart)**: информация о товарах, которые добавлены в корзину.
  + Поля:
    - id (уникальный идентификатор)
    - user\_id (ссылка на пользователя)
    - items (массив объектов, содержащих product\_id, quantity, total\_price)
    - total\_price (общая сумма)
* **Заказы (Orders)**: информация о заказах пользователей.
  + Поля:
    - id (уникальный идентификатор)
    - user\_id (ссылка на пользователя)
    - order\_items (массив объектов, содержащих product\_id, quantity, price)
    - shipping\_address (адрес доставки)
    - status (статус заказа: "обрабатывается", "отправлено", "доставлено")
    - created\_at (дата заказа)
    - total\_price (общая сумма заказа)
* **Отзывы (Reviews)**: отзывы на товары.
  + Поля:
    - id (уникальный идентификатор)
    - user\_id (ссылка на пользователя)
    - product\_id (ссылка на товар)
    - rating (оценка товара, например, от 1 до 5)
    - comment (текст отзыва)
    - created\_at (дата создания отзыва)

### 2. **Реляции и связи между сущностями:**

* **Пользователи и заказы**: один пользователь может иметь несколько заказов, но каждый заказ связан только с одним пользователем.
* **Товары и корзина**: один товар может быть в нескольких корзинах, но каждая корзина может содержать несколько товаров.
* **Пользователи и отзывы**: один пользователь может оставить несколько отзывов, каждый отзыв связан с одним товаром.

### 3. **Используемая база данных и технология:**

* Описание использования **MongoDB** как NoSQL базы данных.
* Использование **Prisma** для работы с базой данных. Можно упомянуть, что Prisma будет использоваться для взаимодействия с базой данных, написания запросов и миграций.
* Структура коллекций в MongoDB.
* Применение схем данных в Prisma для упрощенного взаимодействия с коллекциями базы данных.

### 4. **Типы данных и ограничения:**

* Для всех полей указать типы данных (например, String, Number, Date, Boolean).
* Описание обязательных полей и ограничений (например, email должен быть уникальным, price должен быть больше нуля).
* Указание индексов для часто запрашиваемых полей (например, индекс для email в таблице пользователей или для product\_id в отзывах).

### 5. **Миграции и управление схемой базы данных:**

* Процесс миграций базы данных с использованием Prisma для обновления структуры схемы в процессе разработки.

**2.4 Тестирование сайта**

Тестирование является важным этапом разработки веб-приложений, особенно для интернет-магазинов, где стабильность работы, безопасность и производительность являются критически важными аспектами. В данном разделе рассматриваются методы и подходы, использованные для тестирования интернет-магазина по продаже автомобильных шин. Основной целью тестирования было обеспечение бесперебойной работы всех функциональных компонентов сайта, включая регистрацию пользователей, оформление заказов, а также безопасность данных и производительность при повышенных нагрузках.

Цель тестирования  
Основной целью тестирования было обеспечение надежной и безопасной работы интернет-магазина, с целью гарантировать пользователям бесперебойный и безопасный процесс покупок. Это включает проверку функциональности сайта, его производительности и безопасности данных пользователей.

Для достижения данной цели, были проведены различные виды тестирования, включая функциональное, безопасность и производительность. Это позволило выявить и устранить потенциальные уязвимости, повысить стабильность работы и улучшить пользовательский опыт.

Типы тестирования

1. Функциональное тестирование  
   Мы проверяли ключевые функции сайта, включая:
   * Регистрация и авторизация пользователей: Проверялись корректность работы форм регистрации, входа в личный кабинет, а также функции восстановления пароля.
   * Корзина товаров: Тестировались функции добавления и удаления товаров из корзины, возможность изменения количества товаров и корректность расчета стоимости.
   * Оформление заказа: Проверялась возможность успешного оформления заказа, правильность отображения данных заказа, а также корректность работы с платежной системой.
2. Тестирование безопасности  
   Важно обеспечить защиту данных пользователей, особенно при работе с чувствительной информацией, такой как пароли и платежные данные. Мы сосредоточились на следующих аспектах:
   * Проверка безопасности данных пользователей: Для безопасного хранения паролей использовался алгоритм хеширования с добавлением соли.
   * Контроль прав доступа: Проверяли, что только администратор имеет доступ к панели управления товарами и заказами, а обычные пользователи могут только просматривать товары, добавлять их в корзину и оформлять заказы.
   * Защита от атак: Тестировали уязвимости на стороне сервера, включая защиту от SQL-инъекций, CSRF и XSS-атак.
3. Тестирование производительности  
   Мы провели несколько типов тестирования производительности:

* Тестирование скорости загрузки: С помощью инструментов, таких как Google Lighthouse, проводились тесты на скорость загрузки страниц. Были сделаны оптимизации изображений и сокращение количества запросов для повышения скорости работы сайта.
* Нагрузочное тестирование: С помощью JMeter проводились тесты с моделированием высокой нагрузки (несколько тысяч пользователей одновременно). Целью было проверить, как сайт будет вести себя при большом числе запросов и обеспечить стабильную работу даже при пиковых нагрузках.

Методы тестирования  
Для эффективного проведения тестирования использовались как ручные, так и автоматизированные методы:

1. Ручное тестирование  
   Ручное тестирование включало проверку функциональных компонентов сайта: регистрация, авторизация, добавление товаров в корзину, оформление заказов, а также визуальный осмотр интерфейса на различных устройствах (мобильные, планшеты, ПК).
2. Автоматизированноетестирование  
   Для ускорения процесса тестирования были использованы автоматизированные скрипты для проверки основных сценариев пользовательских действий, таких как добавление товара в корзину и оформление заказа.

Результаты тестирования  
После проведения всех тестов было выявлено несколько багов, включая ошибки валидации на формах, проблемы с правами доступа на серверной стороне и некорректную работу с сессиями пользователей. Все выявленные ошибки были исправлены, после чего было проведено повторное тестирование. В результате все функциональные компоненты были протестированы на совместимость, безопасность и производительность.

После исправления ошибок валидации данных и усиления безопасности, сайт был проверен снова, что позволило обеспечить его стабильную и безопасную работу. Также был улучшен процесс оформления заказов и взаимодействие с платежной системой.

**2.5 Панель администратора**

### 1. **Доступ и отображение панели администратора:**

* **Условия доступа:**
  + Панель администратора доступна только для пользователей с ролью администратора. При входе в систему через **NextAuth** проверяется роль пользователя, и если это администратор, появляется пункт "Админ панель" в навигационном меню.
* **Компоненты панели:**
  + На странице админ-панели отображаются различные элементы управления товарами, такие как список существующих товаров, кнопки для создания нового товара, редактирования или удаления.

### 2. **Создание карточки товара:**

* **Форма для добавления товара:**
  + В админ-панели есть кнопка **"Создать карточку"**, которая открывает форму с несколькими полями для ввода:
    - **Название товара (name)**: текстовое поле для ввода названия шины.
    - **Описание товара (description)**: большое текстовое поле для подробного описания товара, например, его характеристик.
    - **Цена товара (price)**: числовое поле для ввода цены товара.
    - **Фото товара (image)**: возможность прикрепить изображение товара (кнопка для выбора файла).
* **Кнопка "Разместить товар":**
  + После заполнения всех обязательных полей и прикрепления изображения товара появляется кнопка **"Разместить товар"**.
  + При нажатии на кнопку происходит отправка данных на сервер через API, где товар сохраняется в базе данных (MongoDB) и становится доступным для просмотра на сайте.
  + Серверная часть взаимодействует с **Prisma** для добавления данных о товаре в коллекцию **Products**.

### 3. **Редактирование карточек товара:**

* В админ-панели выводится список всех товаров, где для каждого товара предусмотрены кнопки для редактирования и удаления.
* **Кнопка "Редактировать":**
  + При нажатии на кнопку редактирования открывается форма с уже заполненными данными товара (название, описание, цена, изображение), которые можно изменить.
  + После внесения изменений администратор может нажать кнопку **"Сохранить изменения"**, чтобы обновить товар в базе данных.
* Сервер обновляет запись товара в коллекции **Products** в базе данных MongoDB с новыми данными.

### 4. **Удаление карточек товара:**

* **Кнопка "Удалить":**
  + Рядом с каждым товаром в списке есть кнопка **"Удалить"**.
  + При нажатии на эту кнопку товар удаляется из базы данных через API, и товар больше не отображается на сайте.

### 5. **Управление отображением товаров:**

* **Список товаров:**
  + В админ-панели отображается таблица или список с товарами, где указаны их названия, цена, описание и действия (редактировать/удалить).
* **Пагинация:**
  + Если товаров много, администратор может увидеть их с пагинацией (разбиение на страницы), чтобы удобнее было работать с большими объемами данных.

### 6. **Техническая реализация:**

* **Frontend (React, Tailwind CSS, React Icons):**
  + Панель администратора реализована с помощью **React** для динамичного взаимодействия с пользователем.
  + **Tailwind CSS** используется для стилизации элементов интерфейса, таких как формы, кнопки и таблицы.
  + **React Icons** может быть использован для отображения иконок на кнопках (например, для кнопок "Редактировать", "Удалить").
* **Backend (MongoDB, Prisma, Next.js API):**
  + **MongoDB** хранит информацию о товарах в коллекции **Products**.
  + **Prisma** используется для взаимодействия с базой данных и выполнения операций (создание, редактирование, удаление товаров).
  + Для взаимодействия с сервером используется API, реализованное на **Next.js**, где определены эндпоинты для добавления, обновления и удаления товаров.

### 7. **Проверка и валидация данных:**

* Все данные, которые вводятся в форму для создания или редактирования товара, проходят валидацию на сервере. Например:
  + Проверяется, что поля **"Название"** и **"Описание"** не пустые.
  + Проверяется, что **"Цена"** — это положительное число.
  + Проверяется, что изображение загружено в правильном формате (например, .jpg, .png).

### 8. **Безопасность:**

* Все действия в админ-панели защищены аутентификацией через **NextAuth**. Администратор должен быть авторизован для выполнения каких-либо изменений в базе данных.
* Использование JWT-токенов для безопасного взаимодействия между клиентом и сервером.

**ЗАКЛЮЧЕНИЕ**

В рамках разработки клиентской и серверной части интернет-магазина по продаже автомобильных шин был выбран оптимальный технологический стек, включающий React, Tailwind CSS, MongoDB, Prisma и NextAuth, что позволило создать высокоэффективное, адаптивное и безопасное веб-приложение. React обеспечил быстрый интерфейс с использованием виртуального DOM и компонентного подхода, что улучшило производительность и отзывчивость сайта. Tailwind CSS позволил быстро создавать адаптивные интерфейсы, подходящие для корректного отображения на различных устройствах. MongoDB и Prisma предложили гибкость и масштабируемость для работы с динамическими данными, такими как товары и заказы. NextAuth обеспечил безопасную систему аутентификации пользователей, что критически важно для интернет-магазина.

Проект включает несколько функциональных страниц: Главная, О нас, Контакты, Товары, Корзина, Регистрация и Админ панель, каждая из которых выполняет свою роль, обеспечивая удобство навигации и доступ к различным функциям сайта. Особое внимание уделено проектированию интерфейса, обеспечивающего удобство для пользователей и простоту администрирования для менеджеров магазина. Создана административная панель для добавления, редактирования и удаления товаров и заказов.

Система обработки корзины и оформления заказа использует серверную логику на базе Next.js API Routes, что позволяет эффективно управлять данными пользователей и обеспечивать бесперебойную работу магазина. Проект прошел тестирование, включая проверку функционала, адаптивности интерфейса и нагрузочное тестирование для обеспечения стабильной работы при большом числе пользователей.

**ПРИЛОЖЕНИЕ  
СПИСОК ЛИТЕРАТУРЫ**

1. <https://nextjs.org/>
2. <https://tailwindcss.com/>
3. <https://react-icons.github.io/react-icons/>
4. <https://www.prisma.io/>
5. <https://next-auth.js.org/>
6. <https://www.mongodb.com/>
7. <https://gendalf.ru/news/review/sistema-vzaimodeystviya-v-1s/?srsltid=AfmBOopRJXqDZSzN6GvklFNK15nYFB5K-8vlFhEhQholJabV5DVxbyXY>
8. <https://blog.iteam.ru/kak-postroit-vzaimodejstvie-mezhdu-sotrudnikami-kompanii/>
9. <https://huntflow.media/miris-miris-miris/>
10. <https://next-auth.js.org/>
11. <https://nextjs.org/learn/dashboard-app/adding-authentication>
12. <https://create.t3.gg/ru/usage/next-auth/>
13. <https://next-auth.js.org/getting-started/example>
14. <https://e-queo.com/blog/expertnie-stati/kak-organizovat-vnutrenuyu-komunikatsiyu-v-kompanii/?srsltid=AfmBOopDswpioL2oxRLU60rsHP5wCO8OGmB5hnk_2d8O_4l-_CuodRxz>
15. <https://potok.io/blog/hr-overview/vzaimodejstvie-mezhdu-otdelami/>
16. <https://blog.iteam.ru/kak-postroit-vzaimodejstvie-mezhdu-sotrudnikami-kompanii/>
17. <https://b-rs.ru/company/article/sistema-vzaimodeystviya>
18. <https://cyberleninka.ru/article/n/sistema-vzaimodeystviya-organizatsii-truda-i-proizvodstva-na-predpriyatii>
19. <https://1crm.ru/help/sistema-vzaimodeystviya/>
20. <https://ru.legacy.reactjs.org/>
21. <https://react.dev/>
22. <https://developer.mozilla.org/ru/docs/Learn_web_development/Core/Frameworks_libraries/React_getting_started>
23. <https://www.tailwindapp.com/>
24. <https://habr.com/en/companies/kuper/articles/737474/>