**НОВОСИБИРСКАЯ АКАДЕМИЯ ДИЗАЙНА**

**И ПРОГРАММИРОВАНИЯ**

**ДИПЛОМНАЯ РАБОТА**

по специальности «Веб-дизайн»

**Тема:** Разработка интернет-магазина с использованием Next.js и Strapi

Работу выполнил

студент

Фамилия: Геращенко

Имя: Илья

Отчество: Владимирович

Проверил:

Чернов Е.Н.

Барнаул 2021

# Оглавление

[Введение 3](#_Toc72761824)

[Глава 1. Основные принципы работы и создания 5](#_Toc72761825)

[интернет-магазина 5](#_Toc72761826)

[§1. Смысл понятия “интернет-магазин” 5](#_Toc72761827)

[§2. Принцип работы интернет-магазина 5](#_Toc72761828)

[§3. Роль веб-сайта в успешности интернет-магазина 6](#_Toc72761829)

[Глава 2. Практика. Разработка интернет-магазина на Next.js 8](#_Toc72761830)

[§1. Создание Next.js приложения и предварительная настройка 8](#_Toc72761831)

[§2. Разработка клиентской части сайта 14](#_Toc72761832)

[§3. Разработка серверной части сайта 25](#_Toc72761833)

[Заключение 68](#_Toc72761834)

[Список литературы 69](#_Toc72761835)

# Введение

В современно мире интернет-магазины являются неотъемлемой частью жизни абсолютно каждого человека. Сотни тысяч пользователей ежедневно совершают онлайн транзакции на различных сайтах. Это способствует развитию интернет-магазинов. Сегодня у каждой фирмы, производителя, заведения и т.п. есть собственный веб-сайт. В таких условиях появляется конкуренция, а вследствие этого возрастают требования к функционалу интернет-порталов. И именно для их удовлетворения могут использоваться Next.js и Starpi.

Эти технологии вмещают в себя весь необходимый функционал для разработки современных сайтов любой сложности и направленности.

**Целью** данной работы является разработка с использованием Next.js и Strapi быстрого и удобного интернет-магазина, который удовлетворял бы всем современным требованиям.

**Объектом** исследования стали Next.js и Strapi и использование их при разработке веб-сайта.

**Предметом** же дипломной работы будет выступать сам разработанный в её ходе интернет-магазин.

**Гипотезой** выступает результат проделанного исследования. По предположениям, в итоге должен быть создан полностью рабочий лёгкий в использовании и удовлетворяющий все требования пользователя интернет-магазин. Реализован же он должен быть с применением Next.js и Strapi.

В ходе данной работы, будут обязательны для выполнения следующие задачи:

* Изучить все необходимые для разработки технологии, используя доступную информацию из сети интернет;
* Выбрать шаблон будущего сайта;
* Произвести настройку рабочей среды “WebStorm” для того, чтобы упростить процесс написания сайта;
* Настроить линтеры и форматеры;
* Разработать в отдельности каждый из компонентов сайта;
* Сделать необходимые анимации на сайте с использованием соответствующего фреймворка;
* Сделать навигацию по сайту без загрузки и открытия новых страниц, используя диалоговые окна Material-UI;
* Создать корзину товаров;
* Реализовать регистрацию и авторизацию, а также валидацию данных, вводимых пользователем, с помощью технологий Yup и Formik.

Перечисленные задачи, решаются при помощи списка **методов,** приведенного ниже:

* Изучение официальной документации, приложенной к используемой технологии;
* Анализ и использование информации, полученной в ходе обучения или из сторонних источников;
* Использование компонентного подхода к разработке сайта.

# Глава 1. Основные принципы работы и создания

# интернет-магазина

**§1. Смысл понятия “интернет-магазин”**

Интернет-магазин (англ. online shop 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. [1]

Из данного определения можно понять, что интернет-магазин – это сайт, что позволяет дистанционно покупать некие товары и получать их с помощью доставки. Пожалуй, основной смысл использования данных сервисов – это именно дистанционная покупка, когда у покупателя нет возможности прийти в магазин самому и выбрать товар в живую.

**§2. Принцип работы интернет-магазина**

И теперь, когда стал понятен сам термин, следует перейти к принципу работы интернет-магазинов. На самом деле он очень прост и пошагово его можно описать следующим образом:

* Покупатель заходит на сайт интернет-магазина, ищет товар и отправляет его в корзину;
* Оформляя заказ, клиент указывает свои контактные данные и способ оплаты;
* Менеджер связывается с клиентом для подтверждения заказа и уточнения контактных данных, пункта доставки товара, либо же пользователь указывает всё выше перечисленное на сайте в профиле или при оформлении товара и тогда связь с менеджером не требуется;
* Клиент оплачивает заказ на этом этапе, или делает это по прибытию товара;
* Клиент получает заказ и производит оплату, если не сделал этого ранее.

Именно это и является основным принципом работы любого интернет-магазина. [2]

**§3. Роль веб-сайта в успешности интернет-магазина**

На сколько бы очевидно это не звучало, но веб-сайта играет главенствующую в успехи интернет-магазина.

Современный пользователь интернет-ресурсов очень требовательный, поэтому разработчикам приходится приложить максимум усилий, чтобы создать действительно конкурентоспособный интернет-магазин. Такой сайт должен быть удобным, не раздражающим в раскрасках, интуитивно понятным, быстрым. [3]

А именно удобство и функционал сайта. И для реализации выше перечисленного будут использоваться Next.js и Strapi.

Next.js в связке с Material-UI позволит реализовать современный интерфейс, который будет понятен и удобен для пользователя. Вся frontend чачасть будет базироваться именно на использовании Next.js.

Почему был выбран именно Next, а не React? На это есть несколько причин. Основная – это скорость загрузки Next.js приложений - она значительно быстрее, чем React. Это происходит благодаря встроенному рендерингу на стороне сервера. Еще из плюсов можно отметить: Быстрое добавление плагинов для настройки Next.js в соответствии с потребностями вашей страницы, встроенную поддержку маршрутизации страниц, CSS, JSX и TypeScript, а также поддержку функции экспорта статических сайтов. [4]

Для Backend разработки был выбран Strapi в связке с базой данных MongoDB.

Strapi представляет собой фреймворк для управления контентом, работающий на Node.js. Это open source-проект, полностью бесплатный. Система разворачивается локально на собственном сервере компании, что обеспечивает безопасность данных.

CMS можно настраивать и масштабировать с помощью системы плагинов. Здесь предусмотрено множество встроенных возможностей: удобная административная панель, управление аутентификацией и доступами, инструменты для работы с контентом, генератор API и прочее.

Главные особенности и преимущества Strapi – это **открытый исходный код,** система разработана энтузиастами и поддерживается сотнями участников GitHub, которые развивают ее в соответствии с новыми требованиями и технологиями. Она всегда будет доступна и бесплатна.

Панель администратора, как и API, легко настраивается. Функционал расширяется за счет пользовательских плагинов в считанные секунды.

Размещение на собственном сервере владельца системы гарантирует конфиденциальность и обеспечивает повышенный уровень защиты данных (в том числе в соответствии с европейским стандартом GDPR). [5]

MongoDB — документно-ориентированная система управления базами данных с открытым исходным кодом. Для хранения данных используется JSON-подобный формат. Эта СУБД отличается высокой доступностью, масштабируемостью и безопасностью.

СУБД MongoDB полагается на концепции базы данных, коллекций и документов. [6]

# Глава 2. Практика. Разработка интернет-магазина на Next.js

§1. Создание Next.js приложения и предварительная настройка

Перед началом работы необходимо создать Next.js приложение, которое будет являться отправной точкой в разработке магазина.

Для этого необходимо установить Node, открыть Git Bash в выбранной директории, выполнить команду **npx-create-next-app** и выбрать название для приложения (рис. 1).

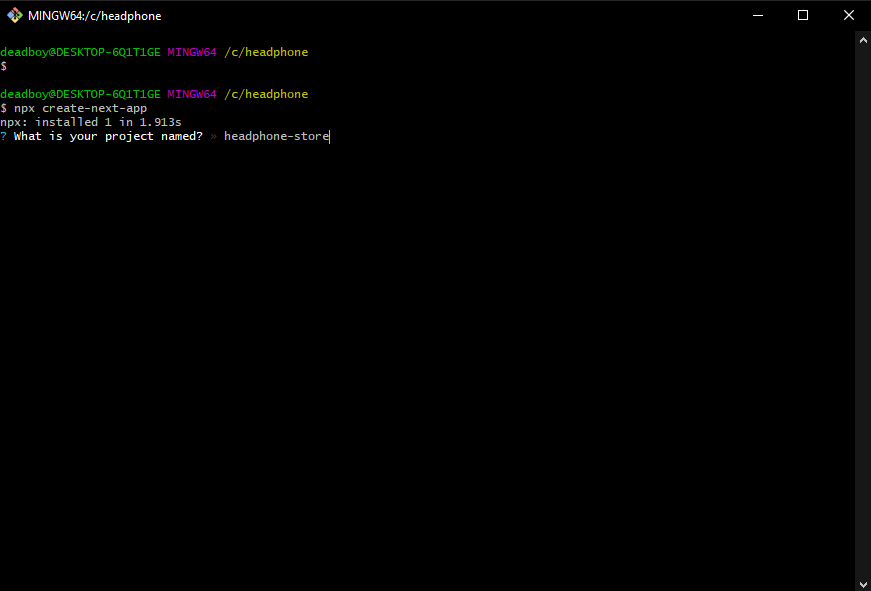


Рис. 1. Создание Next.js приложения

В файле package.json прописываются следующие зависимости зависимости (рис. 2):

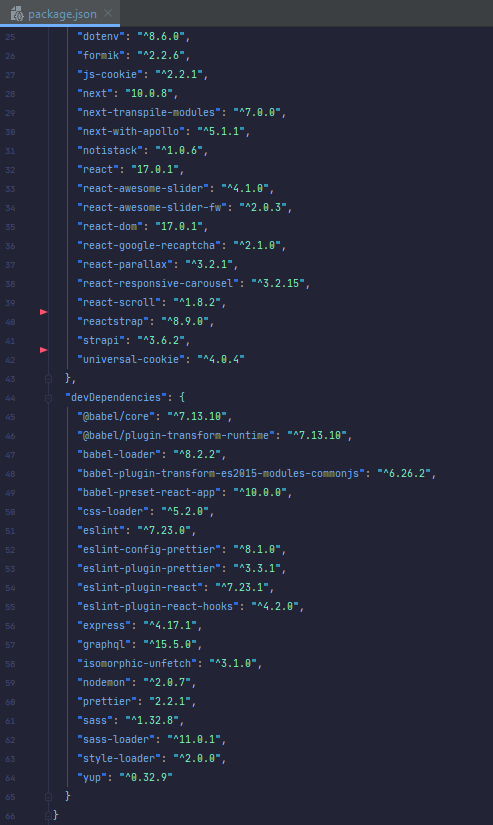


Рис. 2. Зависимости клиентской части

В терминале прописывается команда **npm install**, чтобы все пакеты установились.

Для автоматического форматирования кода используются ESLint и Prettier. Для их работы нужно создать EditorConfig, выбрав используемые языки (рис. 3).

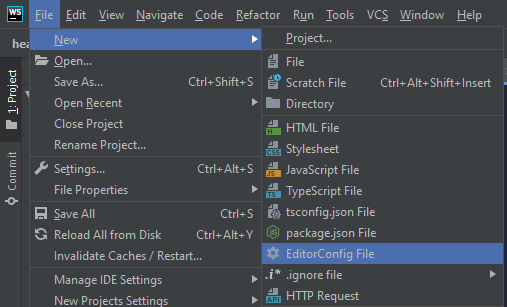


Рис. 3. Создание EditorConfig

Правила написания кода и форматирования текста указываются в файлах .eslintrc и .prettierrc (рис. 4-5).

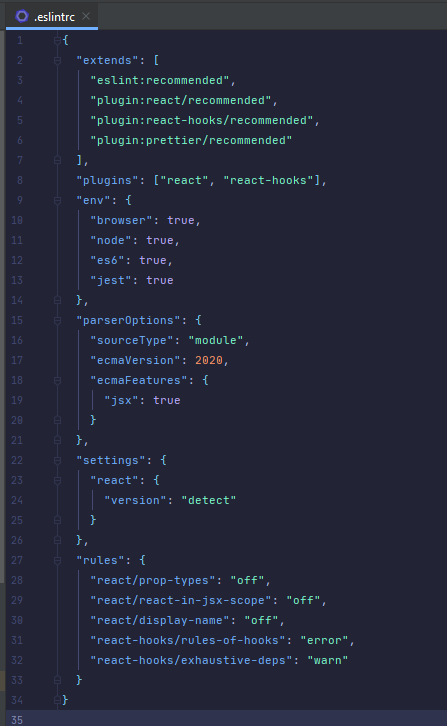


Рис. 4. Файл .eslintrc.

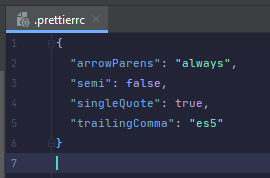


Рис. 5. Файл .prettierrc.

Для удобного хранения проекта и доступа к нему в любом месте следует загрузить его в GitHub репозиторий. Сначала надо пройти регистрацию на GitHub (рис. 6).

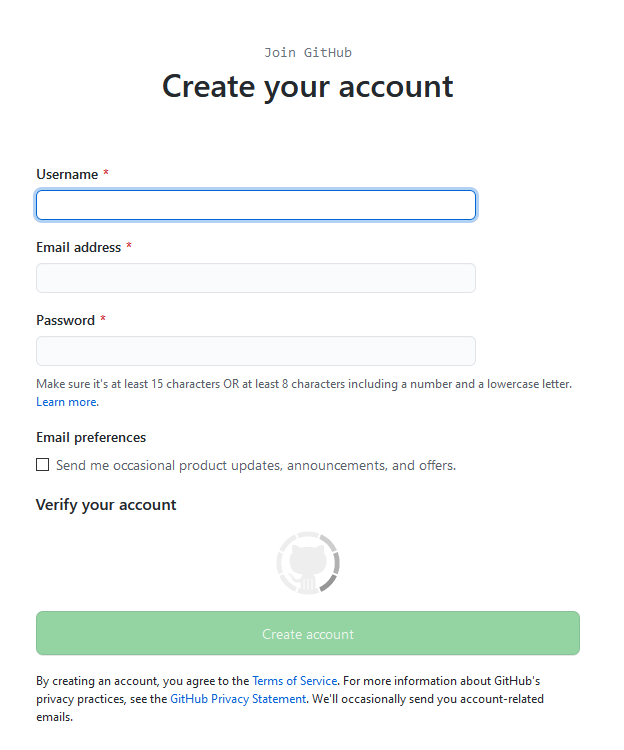


Рис. 6. Регистрация на GitHub

Нажать New для создания репозитория (рис. 7).

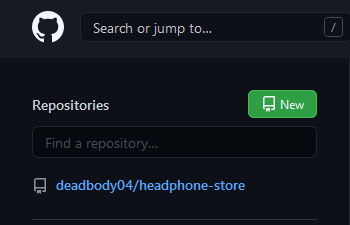


Рис. 7. Кнопка создания репозитория на GitHub

Надо дать ему название и выбрать необходимые настройки (рис. 8).

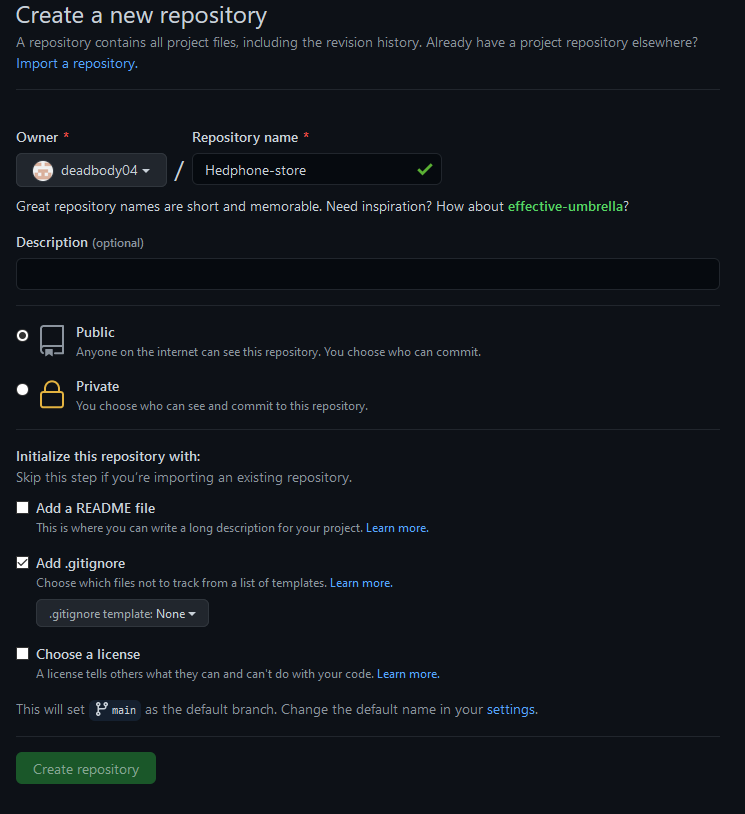


Рис. 8. Создание нового репозитория на GitHub

Теперь в папке с проектом прописываем команды **git init**, **git add**, **git commit –m**, после этого проект будет добавлен в репозиторий.

Нужно настроить SSH. Для этого вводится команда **ssh-keygen-trsa-b4096 –C "электронная почта"**. Сгенерированный ключ нужно вставить в поле Key в настройках GitHub во вкладке SSHandGPGkeys.

## **§2. Разработка клиентской части сайта**

После того, как было создано приложение, можно переходить к разработке сайта. В первую очередь создается Header сайта (рис. 9).

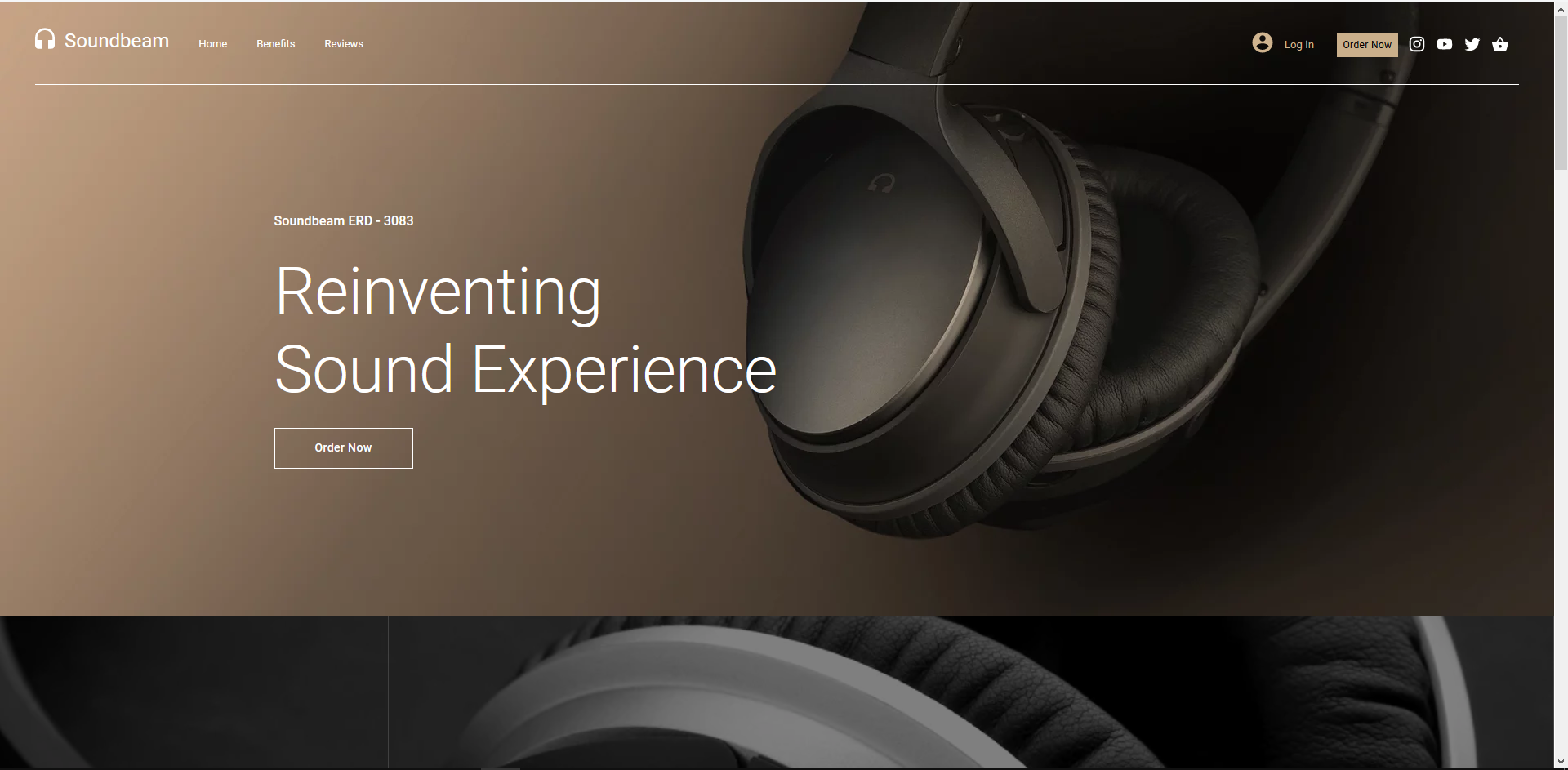


Рис. 9. Header сайта

Наиболее интересной здесь является кнопка входа, по нажатию на неё должно открываться полноэкранное окно, предлагающее авторизоваться, либо зарегистрироваться. Смена окна входа на окно регистрации производиться с помощью подмены содержимого на общем фоне, после нажатия на соответствующую кнопку (рис. 10).





Рис. 10. Открытие диалогового окна

То, какая сейчас страница открыта, хранится в хуке состояния и изменяется при нажатии на ссылку, которая должна открыть другую страницу, таким образом, происходит переключение.

После находится секция с parallax эффектом (рис. 11).

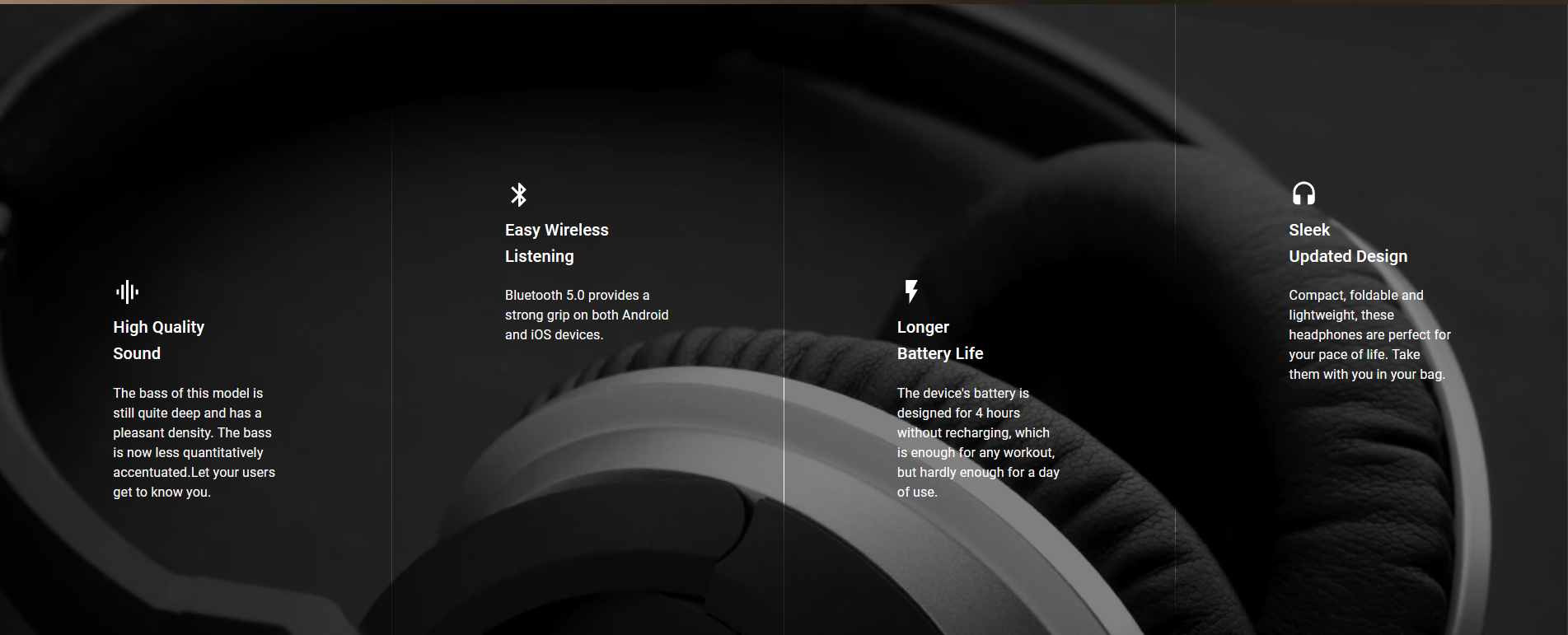


Рис. 11. Parallax эффект

Данный эффект реализовывается с помощью фреймворка react-parallax. Вся секция оборачивается в тег “Parallax” с атрибутами “bgImage” и “strength”. Первый отвечает за фоновое изображение, а второй за скорость эффекта (рис. 12).



Рис. 12. Тег Parallax

Еще ниже идет секция с изображениями спонсоров, а под ней описание товара, которое должно завлекать пользователя (рис. 13).

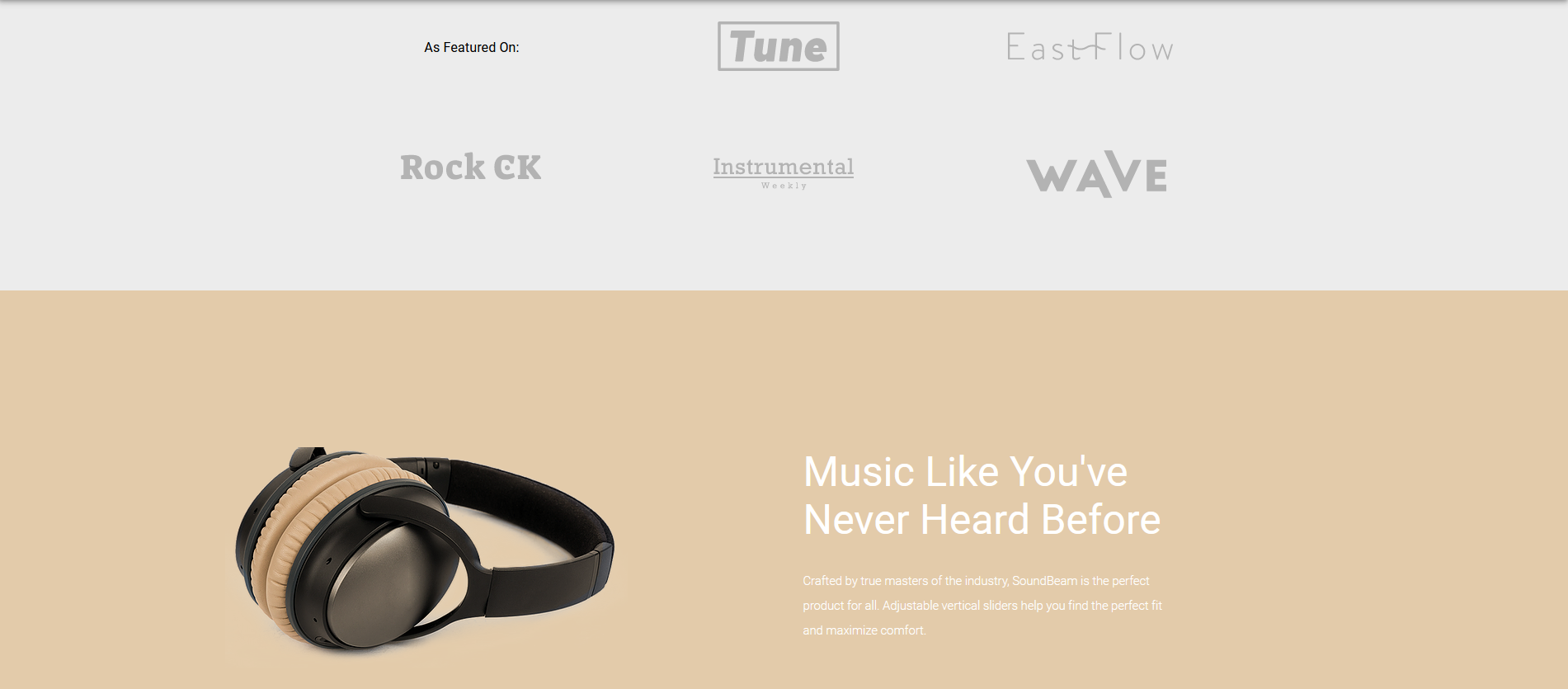


Рис. 13. Секция спонсоров и секция описания

Листая далее, пользователь видит слайдер, который реализуется с помощью фреймворка react-responsive-carousel (рис. 14).

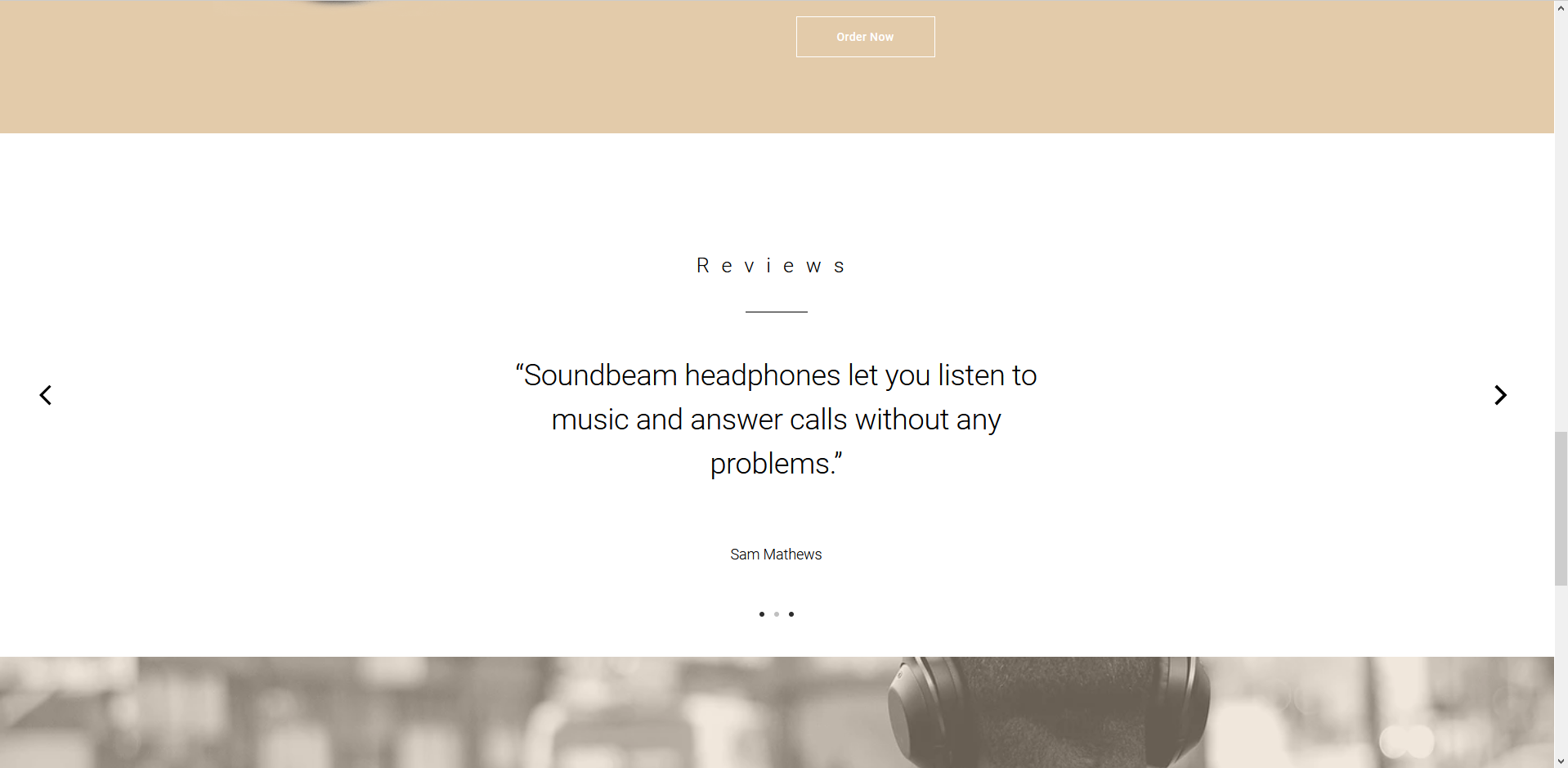


Рис. 14. Слайдер отзывов

Он позволяет завлечь потенциального покупателя тем, что предоставляет краткие отзывы от известных в индустрии людей. А его дизайн привлекает внимание своей минималистичностью.

Чтобы дольше удерживать внимание пользователя ниже расположена еще одна секция с Parallax эффектом, реализованным абсолютно аналогично предыдущему (рис. 15).



Рис. 15. Еще один Parallax эффект

В самом же конце расположена секция с полезными ссылками и ниже footer сайта (рис. 16).

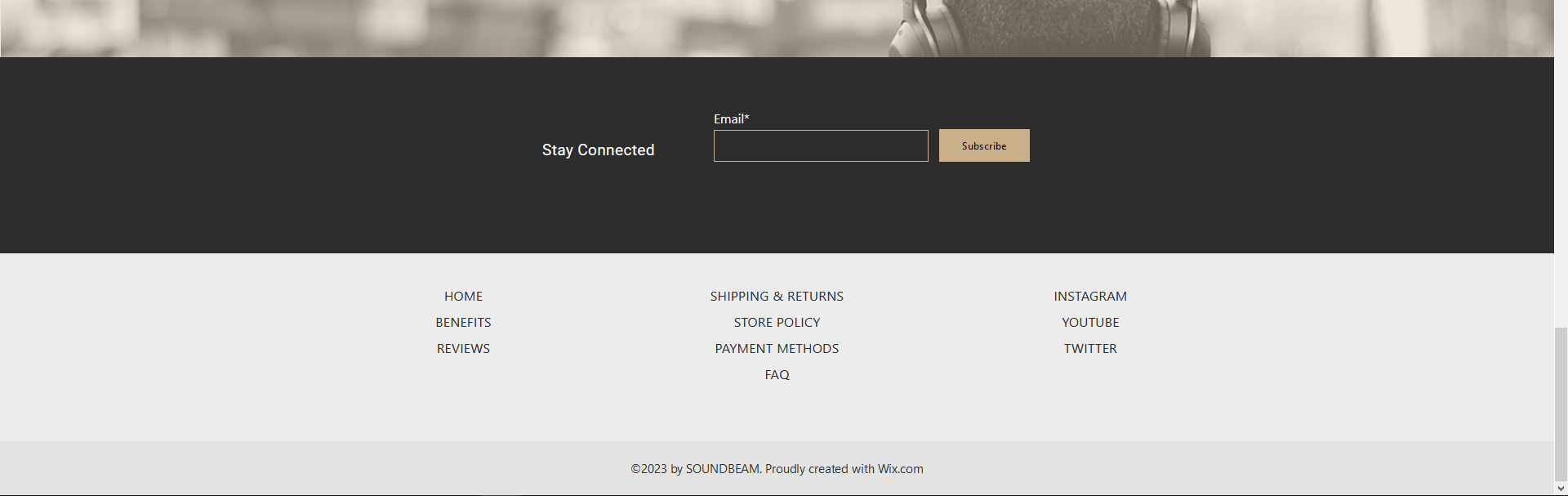


Рис. 16. Нижняя часть сайта

После этого началась разработка корзины сайта, а также возможность добавления товара в неё (рис. 17-18).

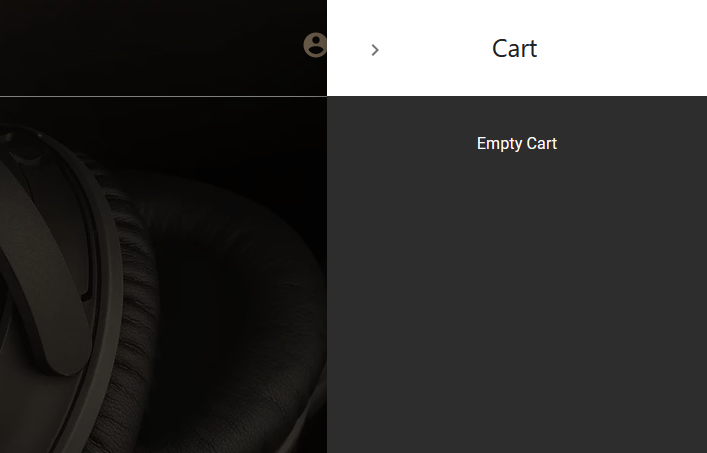


Рис. 17. Пустая корзина

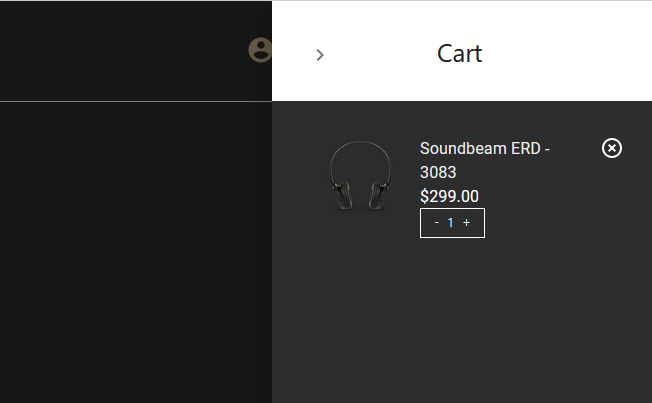


Рис. 18. Товар в корзине

Для реализации интерфейса корзины была использована навигационная панель Material-UI (рис. 19).

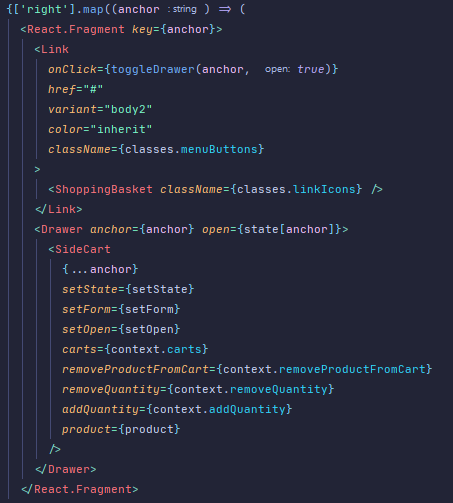


Рис. 19. Material-UI корзина

Для хранения информации о товарах был создан store. Для добавления товара в корзину используется функция addProductToCart (рис. 20).

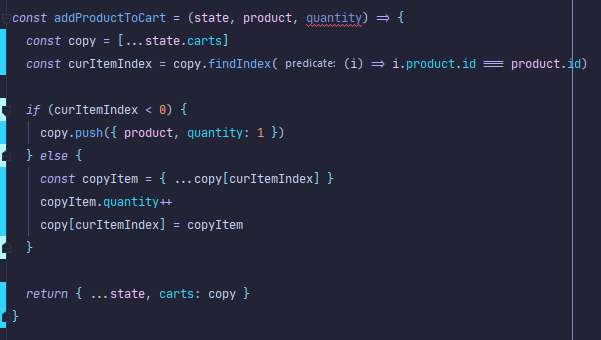


Рис. 20. Функция addProductToCart

Для удаления - removeProductFromCart (рис. 21).

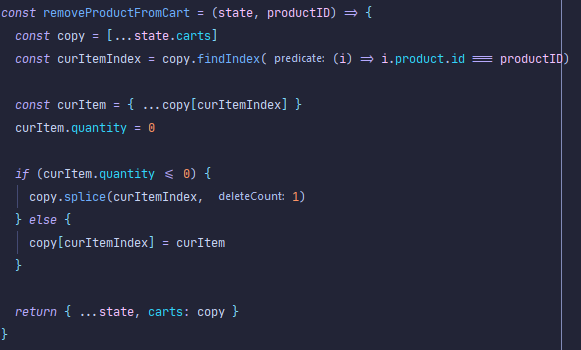


Рис. 21. Функция removeProductFromCart

А для изменения кол-ва товара созданы функции removeQuantity и addQuantity (рис. 22).

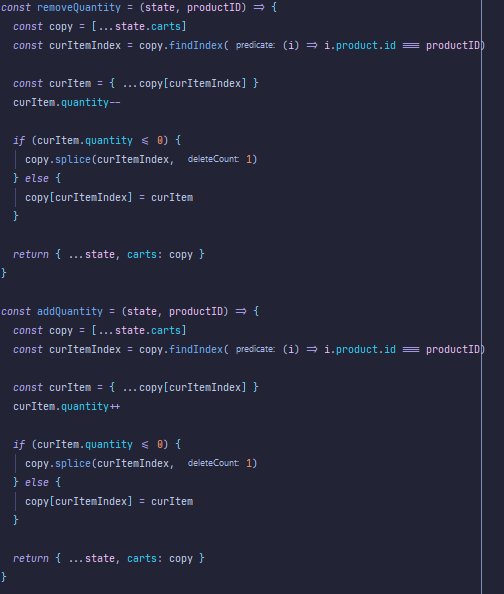


Рис. 22. Функции removeQuantity и addQuantity

По нажатию на кнопку вызывается соответствующая её назначению функция. Например, так выглядит кнопка добавления товара (рис. 23):

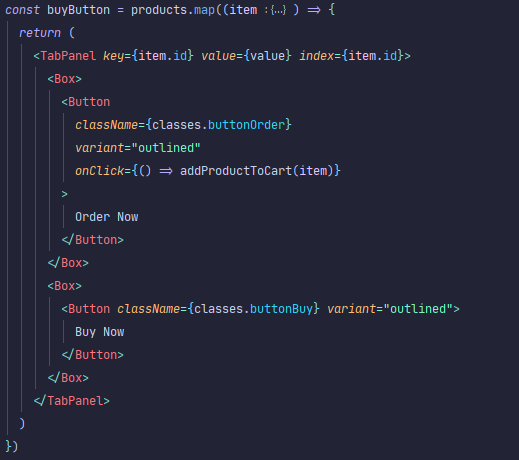


Рис. 23. Кнопка покупки товара

Товар же на странице формируется динамически из базы данных (рис. 24-25).

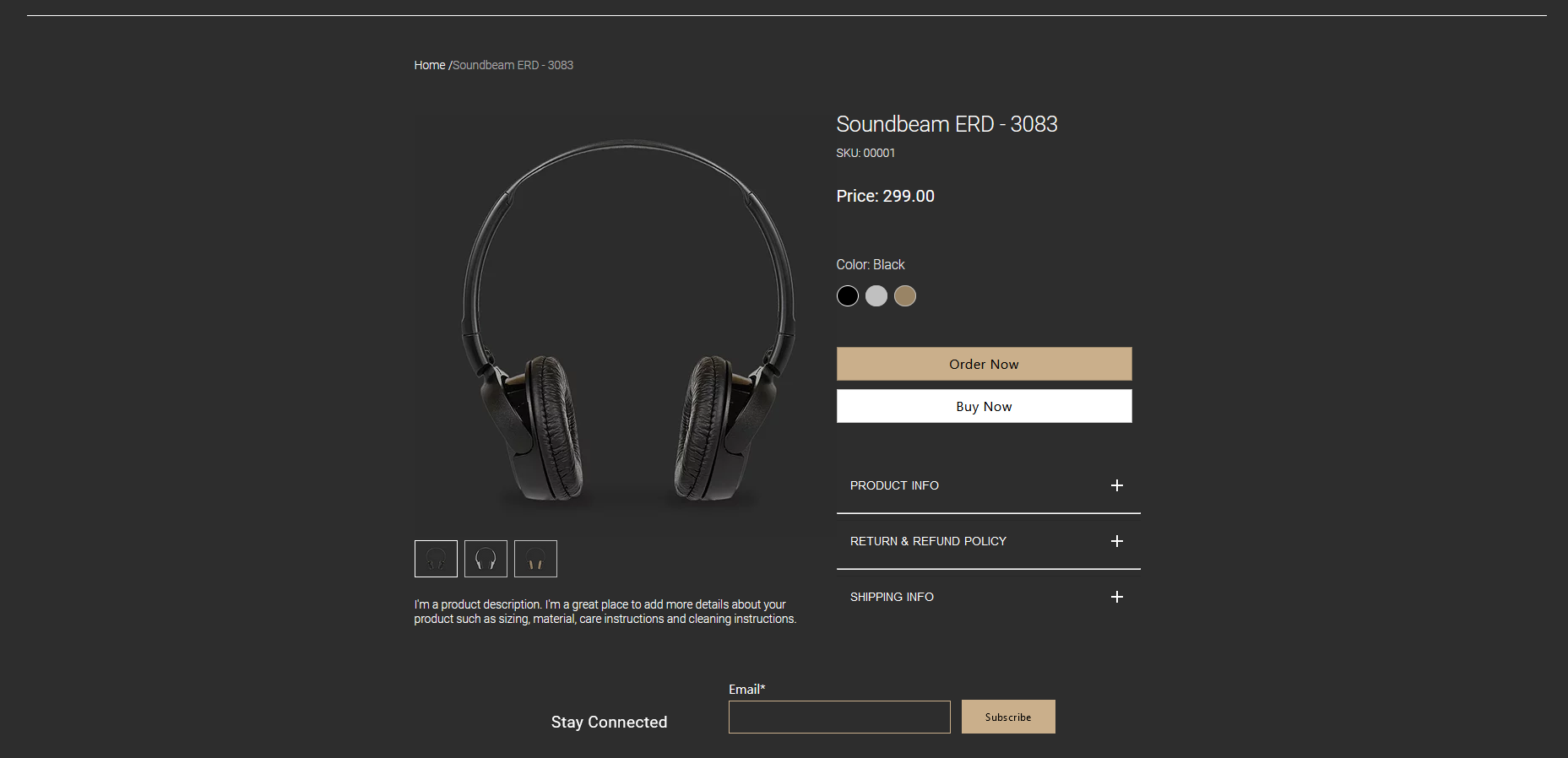
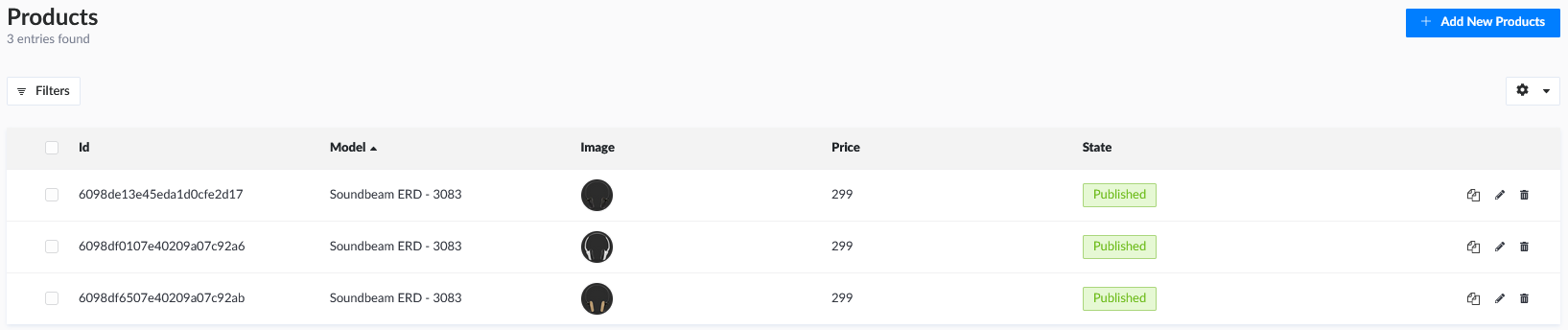


Рис. 24. Страница товара

Рис. 25. База данных

Подсчет итоговой стоимости реализуется при помощи функции reduce, которая возвращает сумму всех элементов массива carts, в котором хранится информация о товарах, добавленных в корзину (рис. 26).

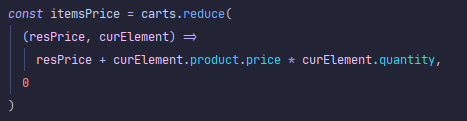


Рис. 26. Подсчет итоговой суммы

Самым последним этапом разработки клиента было создание страницы корзины (рис. 27).

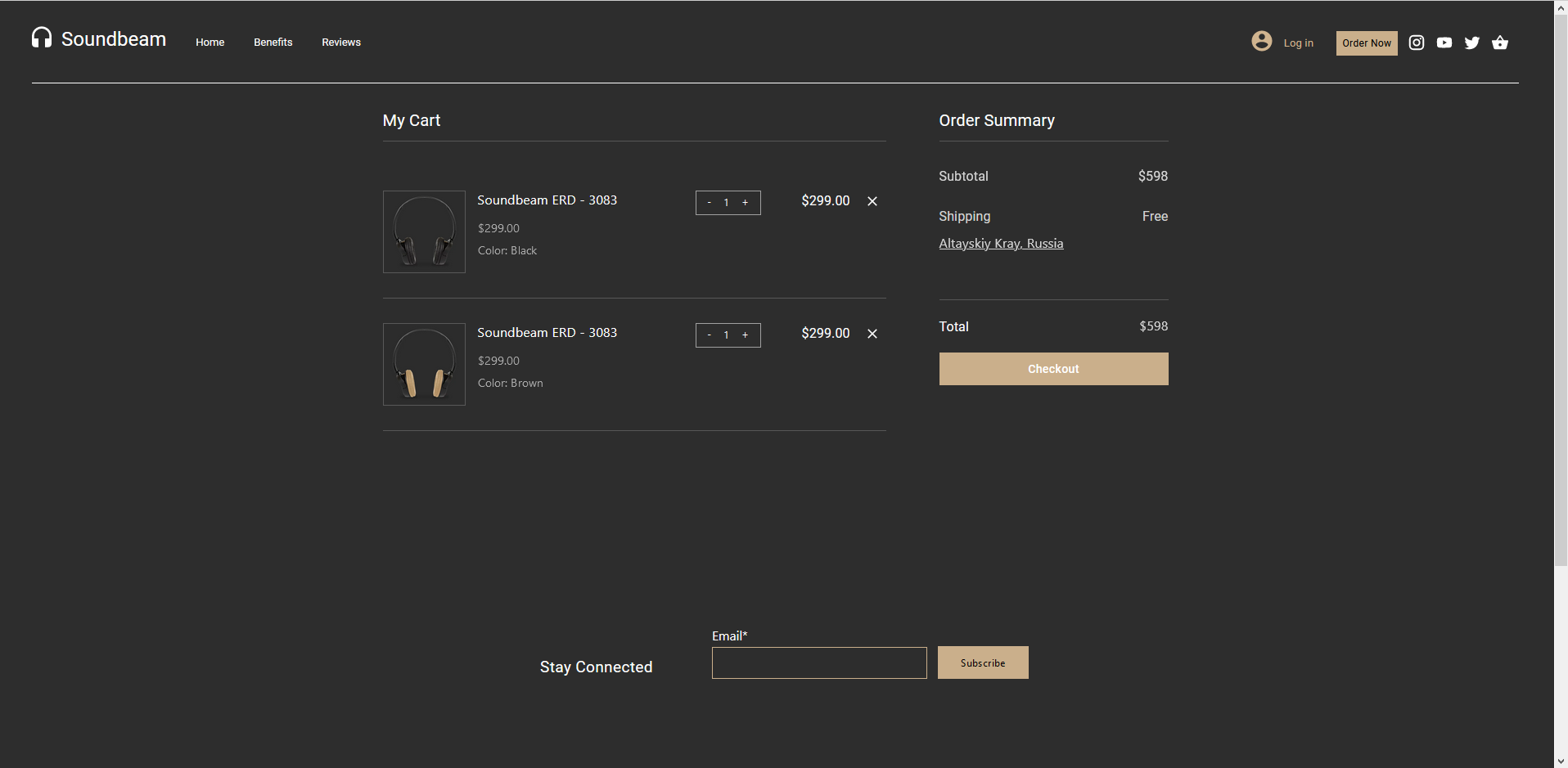


Рис. 27. Страница корзины

В ней используются все те же самые функции, что и в корзине, которая открывалась при помощи навигационного меню, никаких изменений, кроме интерфейса нет.

После того, как была закончена разработка, вся клиентская часть должна быть сохранена в папке “client” и загружена на платформу Vercel.

Для этого надо будет зарегистрироваться, используя свой GitHub аккаунт (рис. 28).

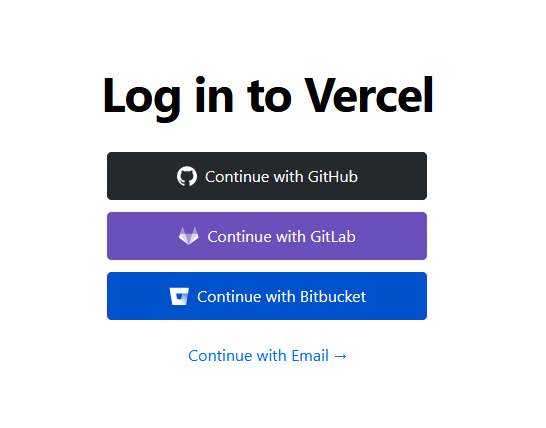


Рис. 28. Vercel

Затем надо будет добавить проект из репозитория (рис. 29). А в его настройках указать дерикторию в которой находится клиентская часть (рис. 30).

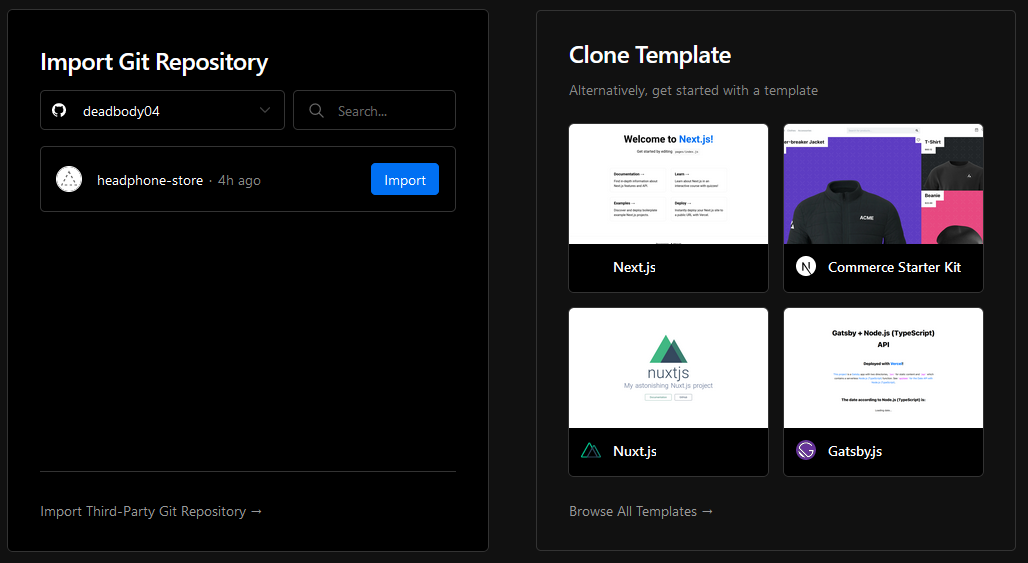


Рис. 29. Импорт проекта

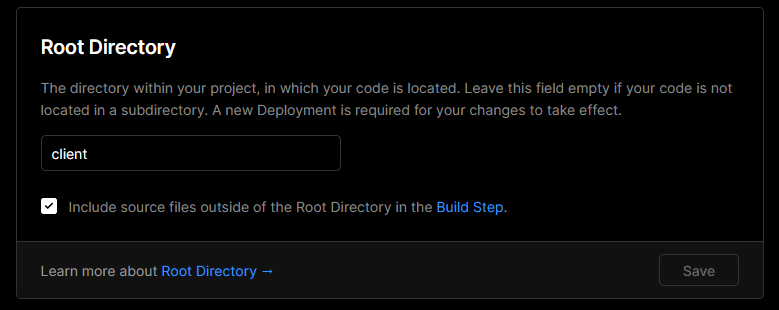


Рис. 30. Настройка проекта

После этого vercel будет автоматически деплоить проект, когда в репозиторий буду вноситься изменения (рис. 31).

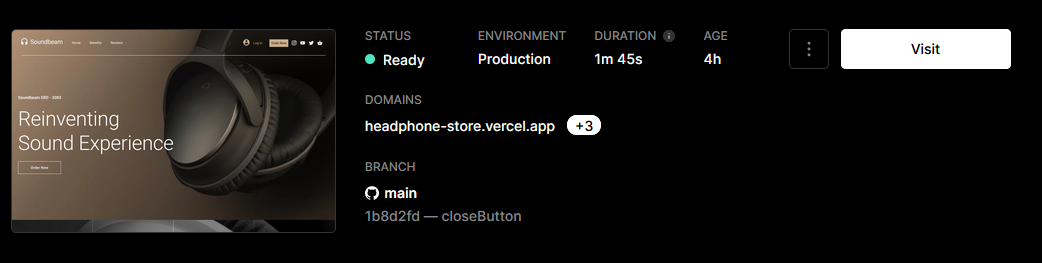


Рис. 31. Добавленный на vercel проект

## **§3. Разработка серверной части сайта**

Серверная часть разрабатывается при помощи фреймворка Strapi. В качестве базы данных будет использоваться MongoDB.

Регистрируем аккаунт MongoDB и создаем кластер, выбрав Frankfurt (рис 32).

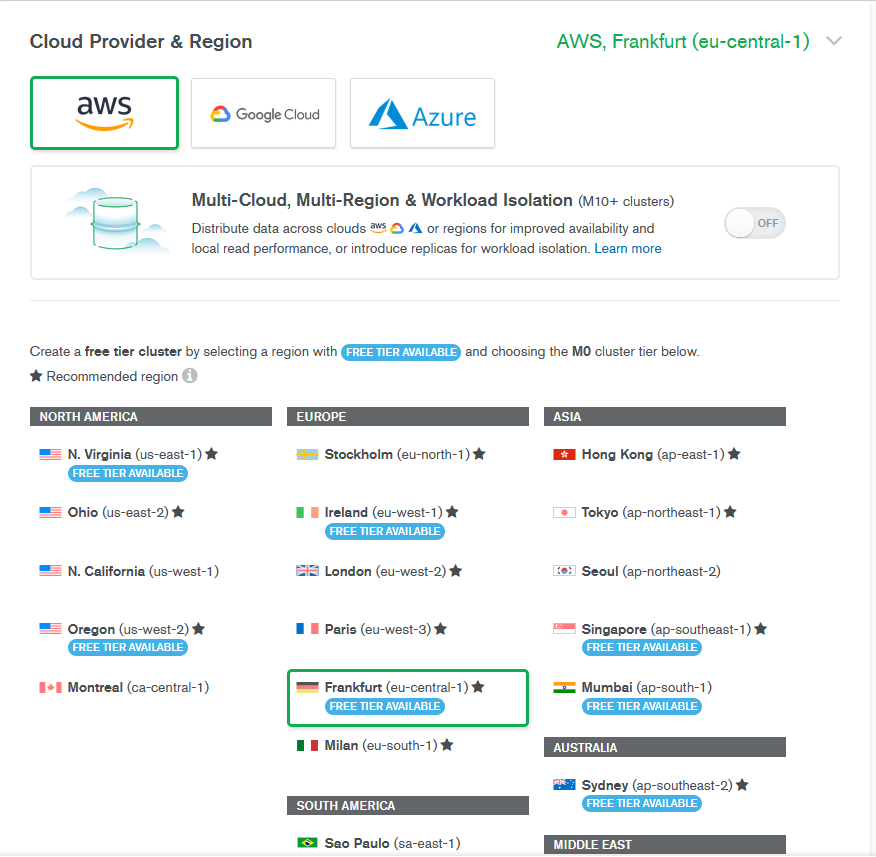


Рис. 32. Создание кластера

Нужно создать Strapi приложение. Находиться оно должно в папке “server”, рядом с папкой “client”. Приложение создаётся командой **npx create-strapi-app** в GitBash, а затем выбирается мануальная установка (рис 33).

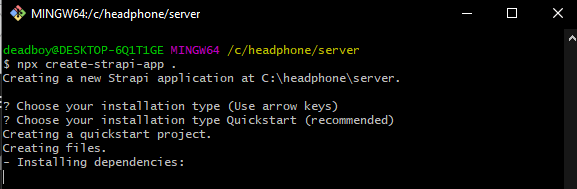


Рис. 33. Создание Strapi приложения

В итоге должна получится подобная структура (рис. 34).

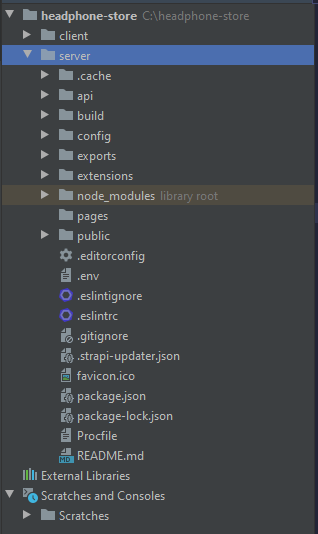


Рис. 34. Структура приложения

Необходимо установить зависимости (рис. 35).

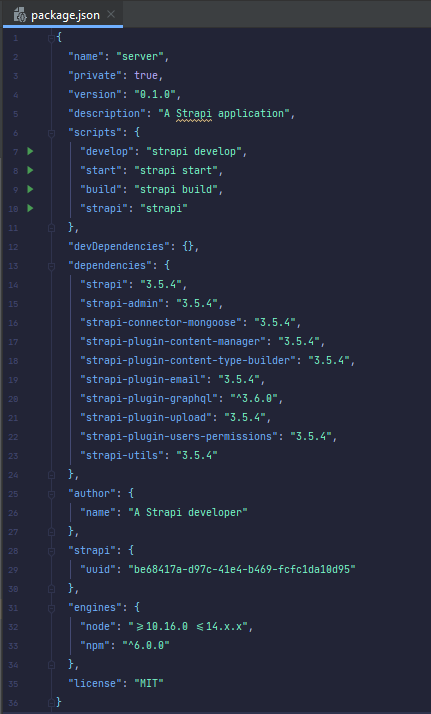


Рис. 35. Файл Package.json для сервера

Надо настроить Procfile, добавив в него одну команду (рис. 36). Это необходимо, чтобы не было проблем с развертыванием приложения в Heroku.

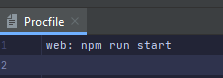


Рис. 36. Procfile

Серверная часть будет развертываться в Heroku. Нужно зарегистрироваться на нем и добавить проект из своего репозитория (рис. 37-38).

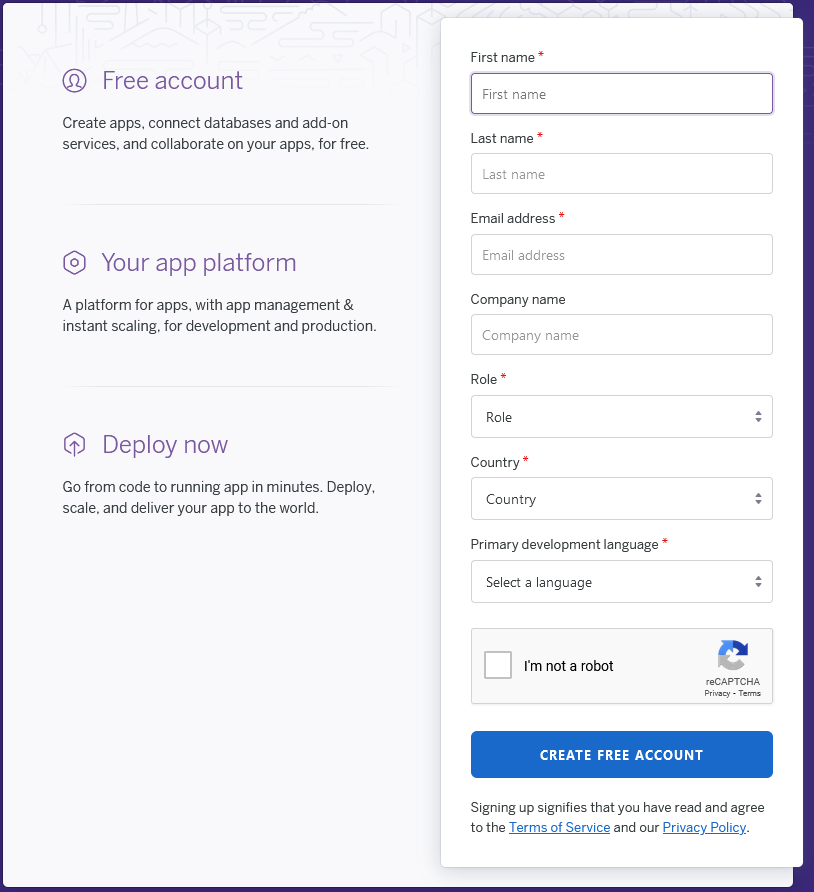
Рис. 37. Регистрация Heroku



Рис. 38. Импорт проекта

Далее, в настройках добавляются необходимые для работы билдпаки (рис. 39).

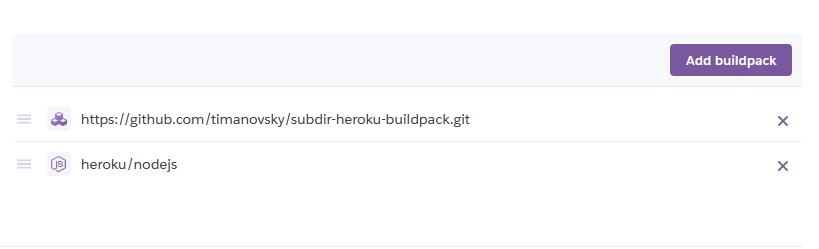


Рис. 39. Добавление билдпаков

В конце включается автоматический деплой (рис. 40).

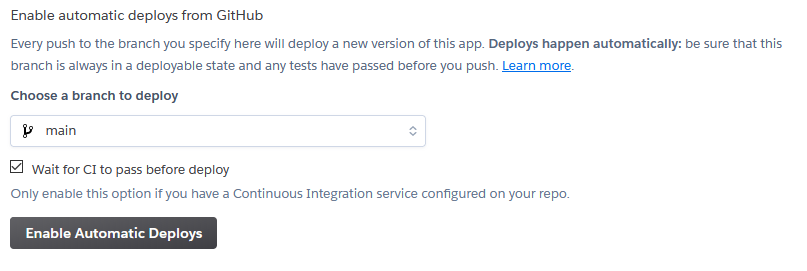


Рис. 40. Автоматический деплой

После надо будет создать next.config.js, добавив в него переменные окружения, и настроить ApolloClient с помощью которого, в связке с GraphQL будут осуществляться запросы и мутацию к серверу (рис. 41-42).

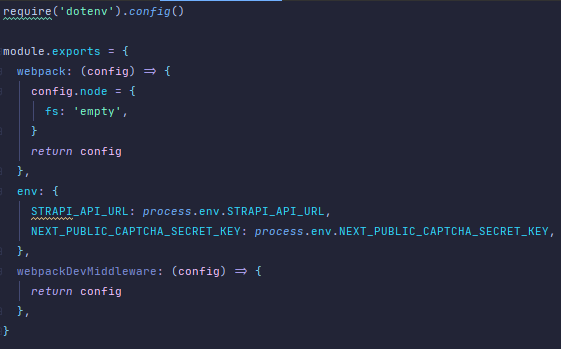


Рис. 41. next.config.js



Рис. 42. Настройка ApolloClient

Для связи с базой данных нужно внести изменения в файлы database.js и server.js (рис. 43 - 44).

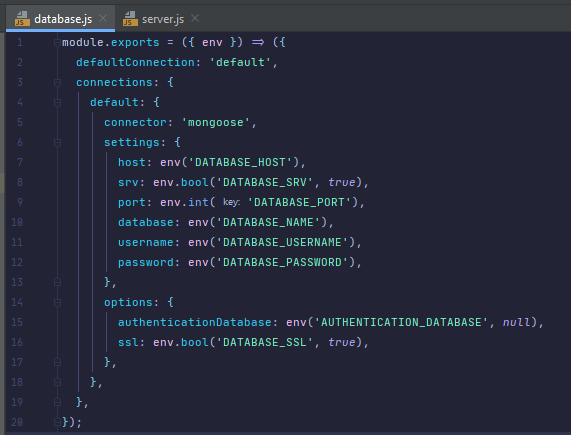


Рис. 43. Файл database.js

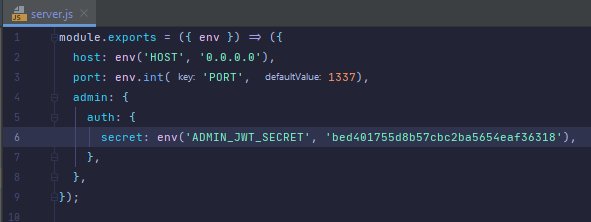


Рис. 44. Файл server.js

Все запросы и мутации хранятся в папке graphql (рис. 44-49).

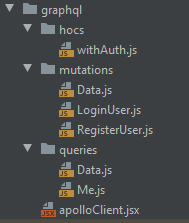


Рис. 44. Структура папки graphql

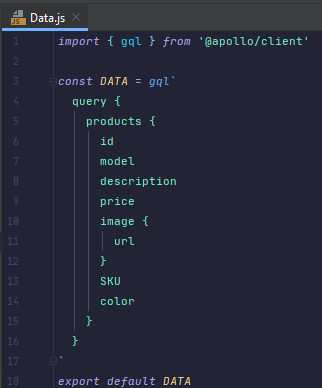


Рис. 45. Файл DATA



Рис. 46. Файл Me

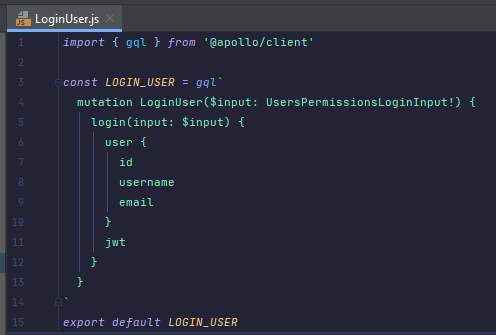


Рис. 47. Файл LoginUser

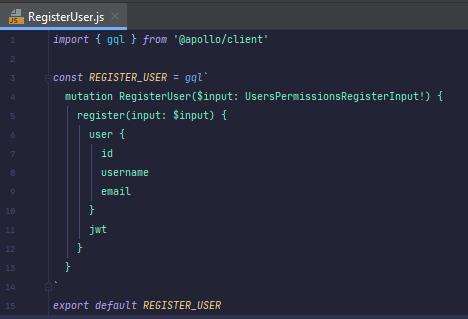


Рис. 48. Файл RegisterUser



Рис. 49. Файл withAuth

Еще понадобится создать файлы .env.development и .env.production (рис. 49 - 50).



Рис. 49. файл .env.development



Рис. 50. Файл .env. production

Для регистрации пользователей создаётся store, как и для корзины (рис. 51).

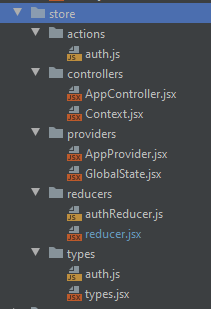


Рис. 51. Структура store

Все описания событий хранятся в папке types в файле auth.js (рис. 52). Сами события находятся в файле auth.js в папке actions (рис. 53). За обновление store отвечает authReducer.js (рис. 54). Для того чтобы использовать store в компонентах создаются файлы AppProvider.jsx и AppController.jsx (рис. 55-56). В файл app.js вносятся следующие изменения (рис. 57).

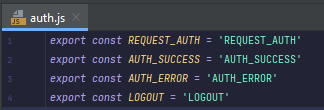


Рис. 52. Описание событий

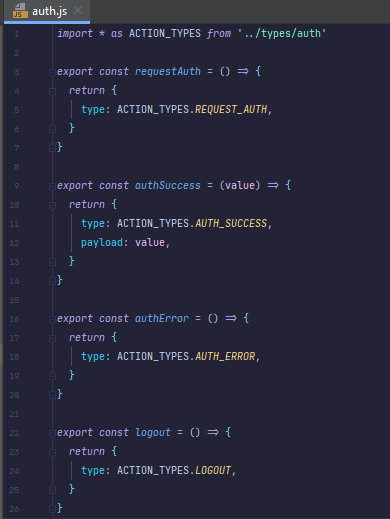


Рис. 53. События



Рис. 54. Файл authReducer.js



Рис. 55. Файл AppProvider.jsx

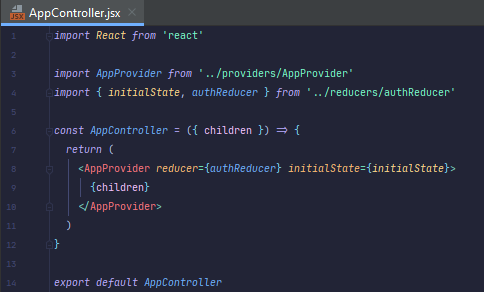


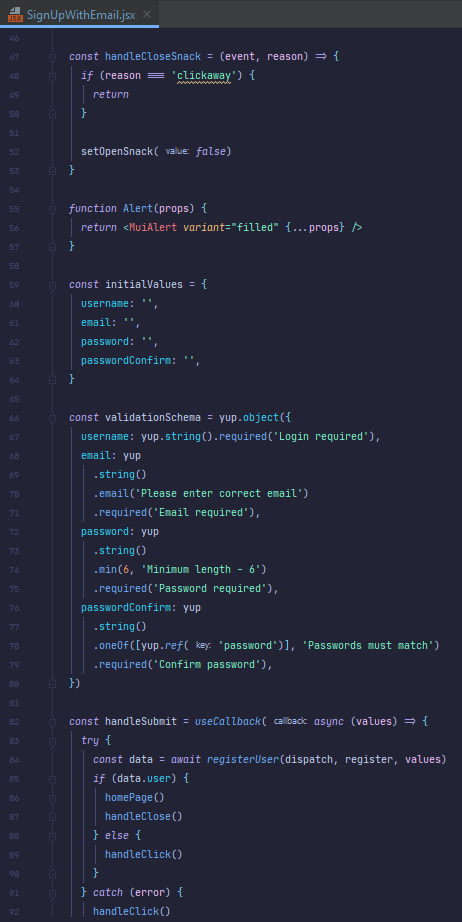
Рис. 56. Файл AppController.jsx



Рис. 57. Файл app.js

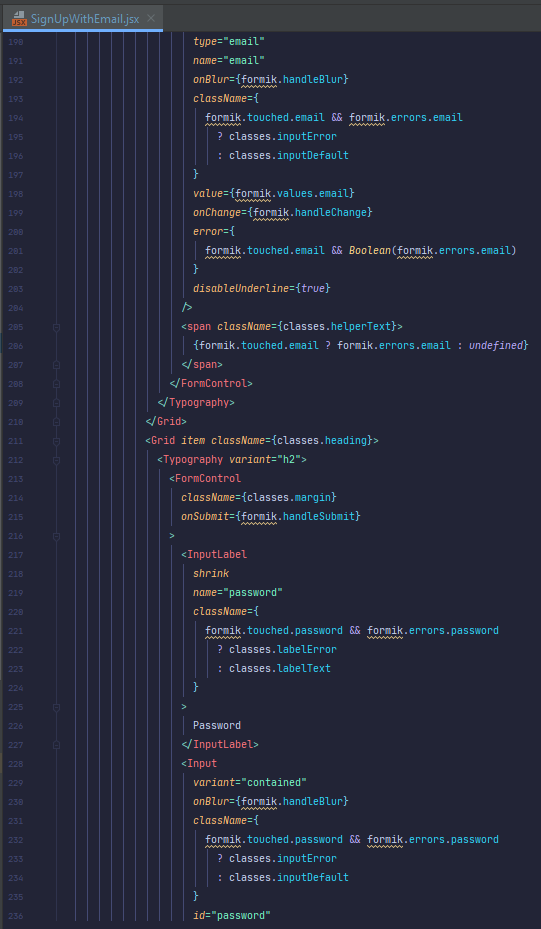
Чтобы пользователь мог без проблем зарегистрироваться необходимо реализовать валидацию вводимых данных с помощью Formik и Yup, а также сделать уведомления об ошибках во время регистрации, используя Snackbar и MuiAlert из пакета lab – Material-UI (рис. 58-59).

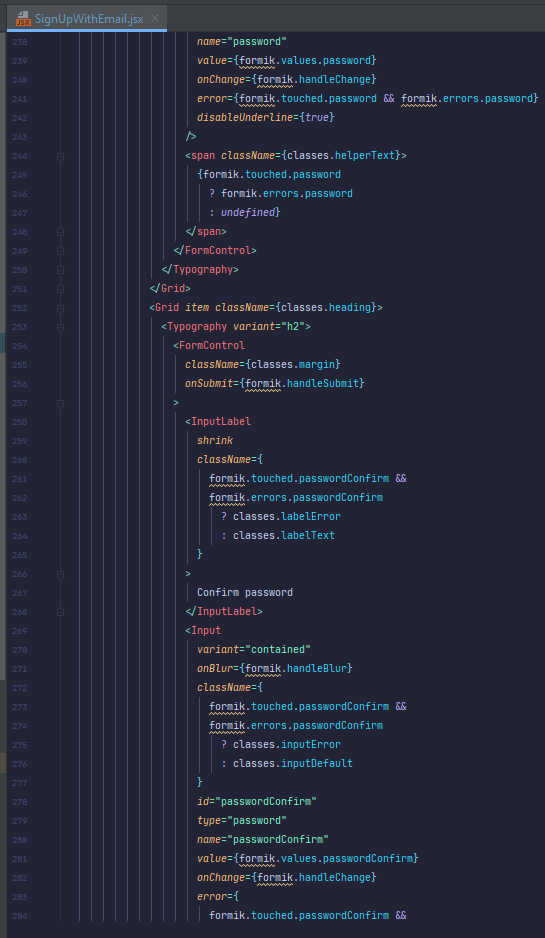












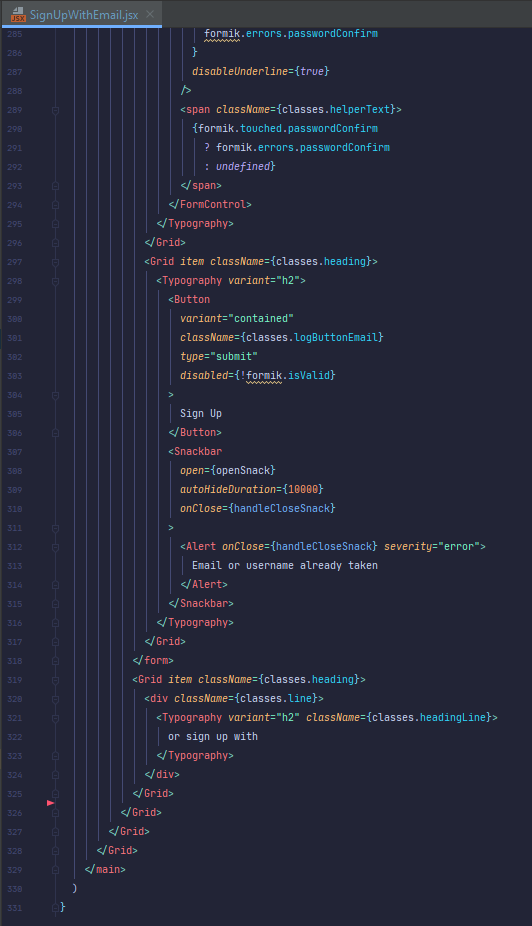
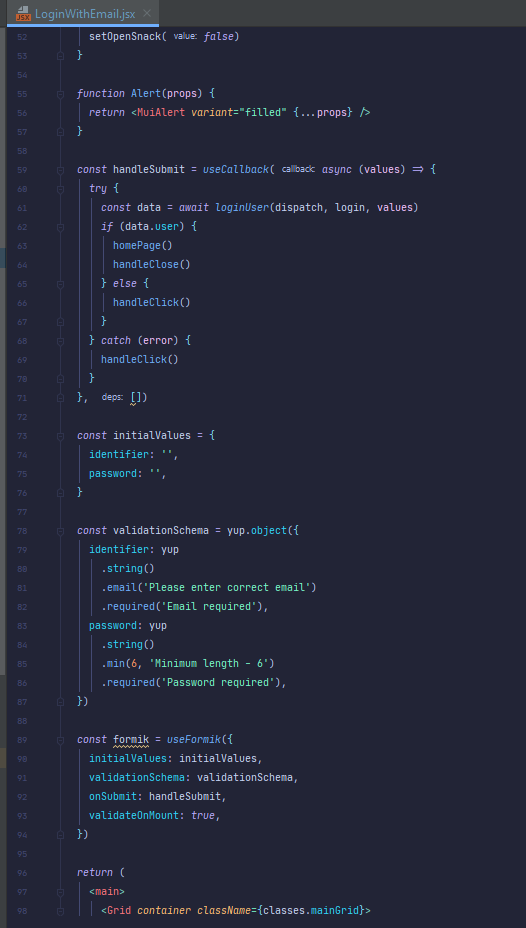
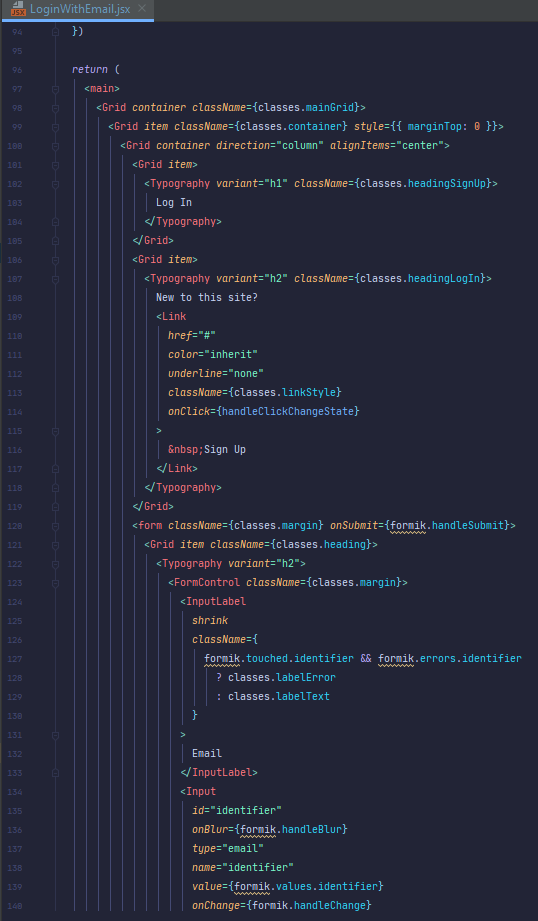


Рис. 58. Страница регистрации









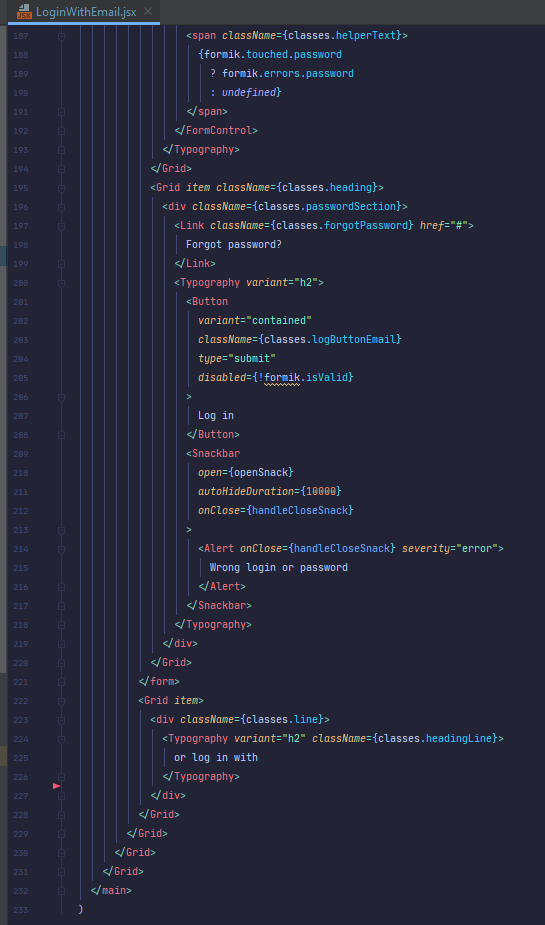


Рис. 59. Страница входа

Теперь, чтобы вводимые пользователем данные оказались в store, а также на сервере, нужен файл auth.js, который находится в директории utils в папке \_mocks\_ (рис. 60). Еще в этом файле будет создаваться кука с токеном, чтобы пользователю не приходилось после регистрации дополнительно заходить на страницу входа и вводить свои данные заново.



Рис. 60. Файл auth.js

Для удобства пользователя добавляется социальная аутентификация. В настройках Strapi во вкладке Providers выбираются сервисы, через которые будет доступна регистрация и авторизация (рис. 61).

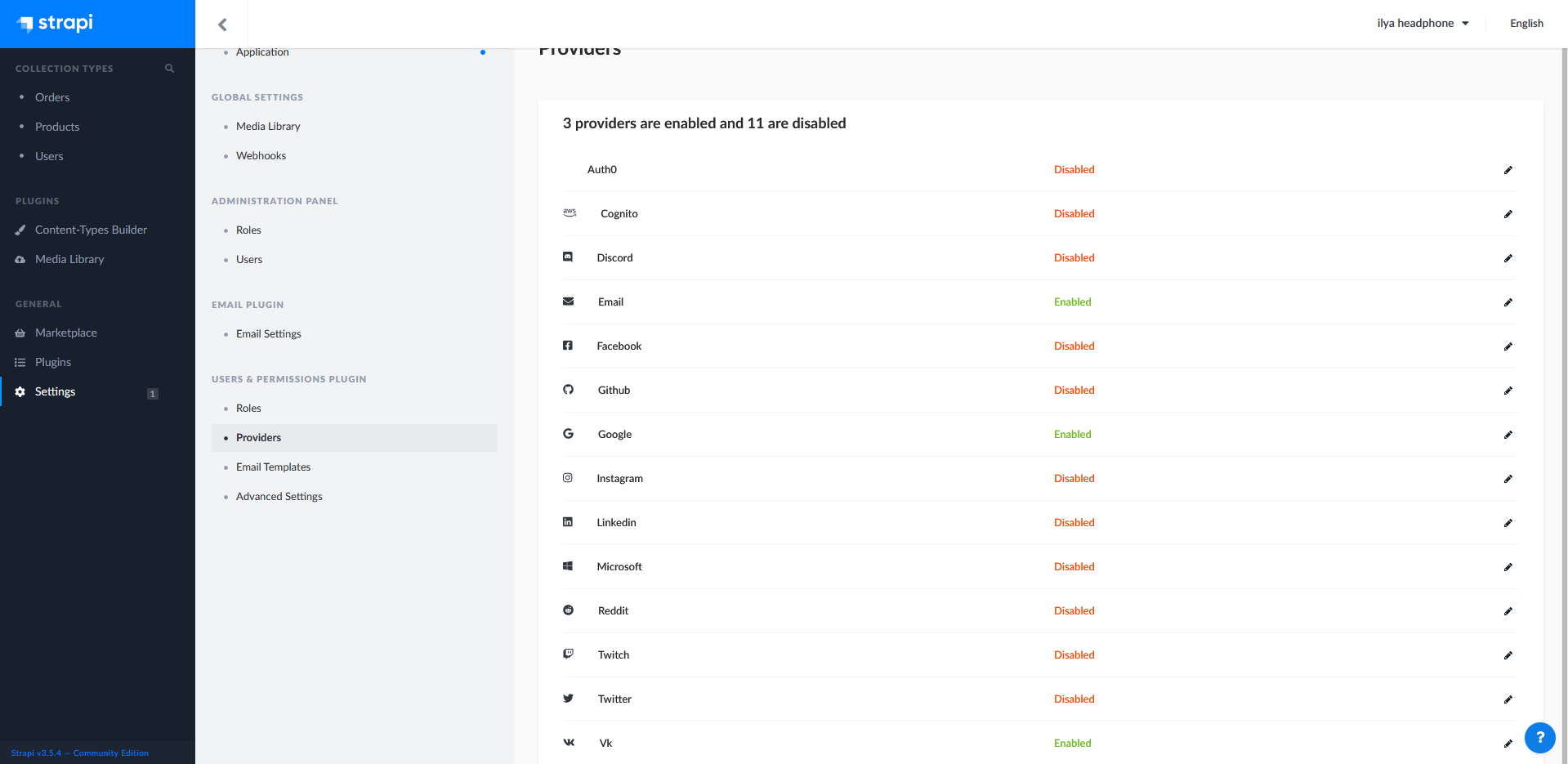


Рис. 61. Strapi провайдеры

Для каждого провайдера необходимо заполнить поля Client id и Client Secret. Значения, которые туда вводятся можно получить на сайтах самих провайдеров (рис. 62- 65).

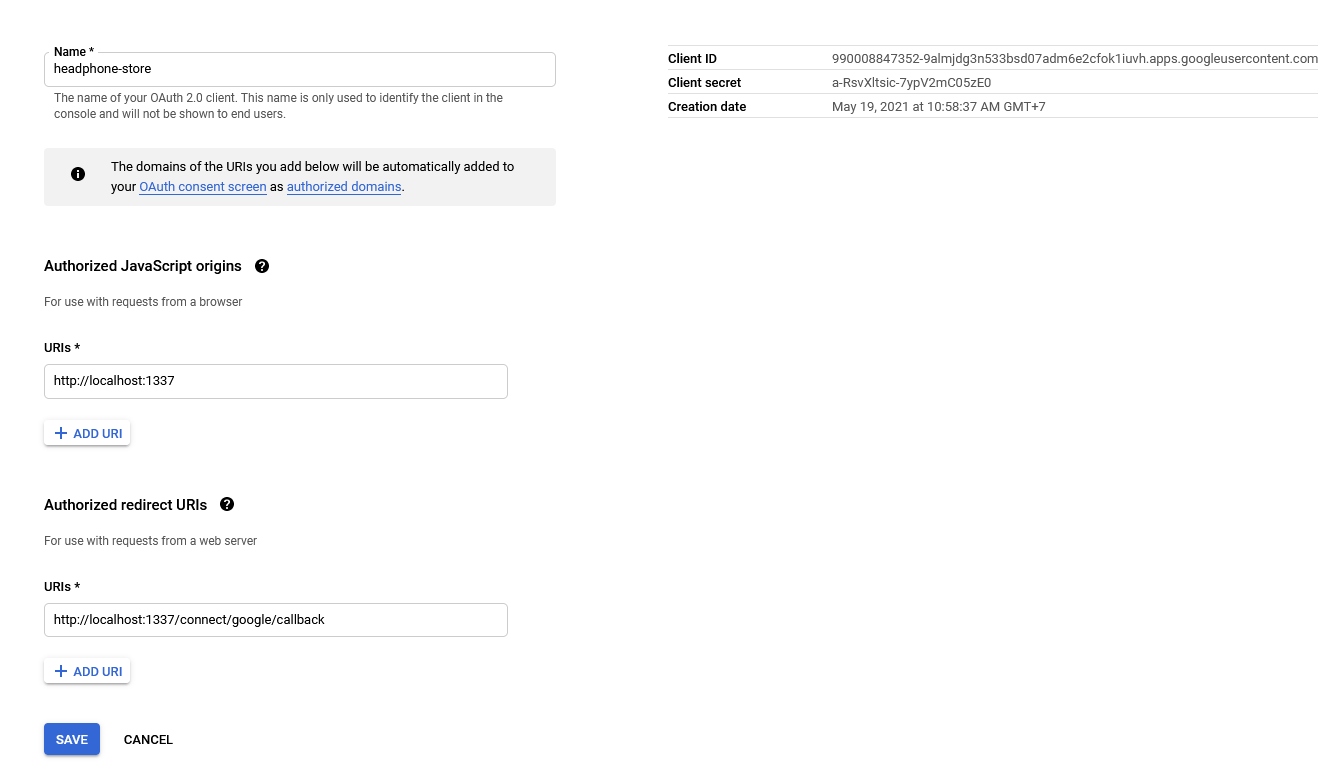


Рис. 62. Создание приложения Google console

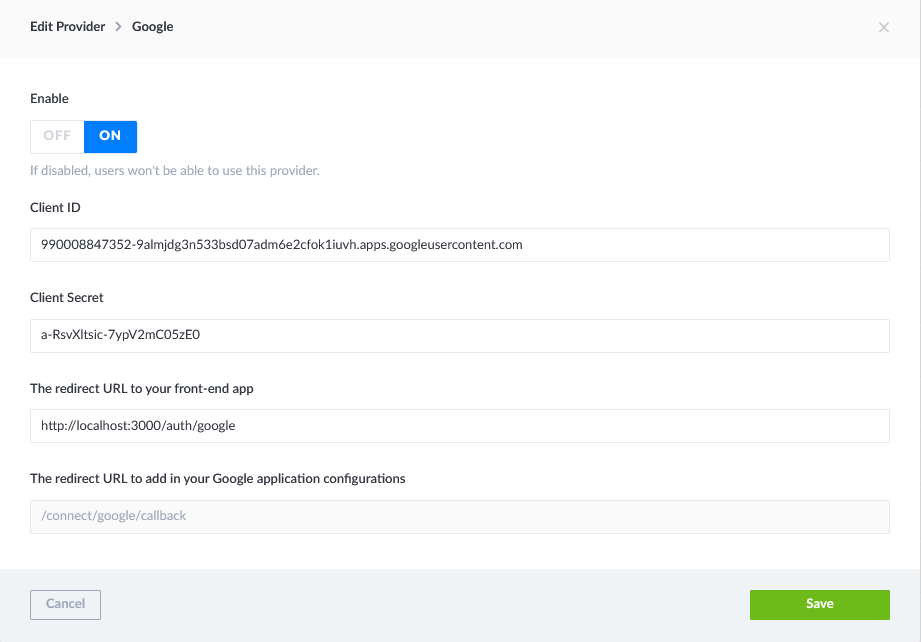


Рис. 63. Google провайдер в Strapi

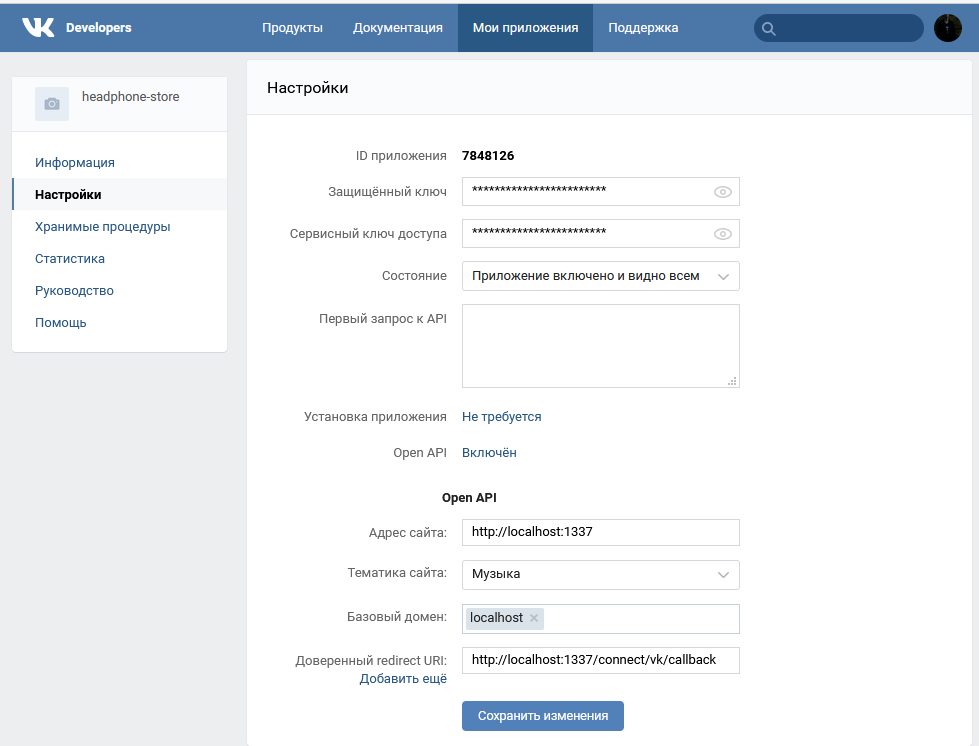


Рис. 64. Создание приложения Vk Developers

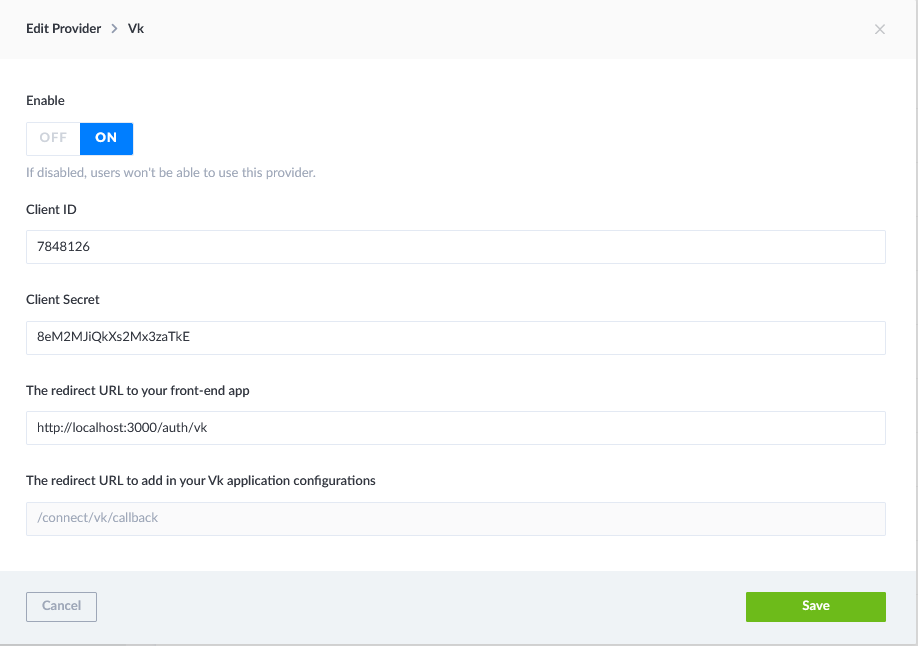


Рис. 65. Vk-провайдер в Strapi

Чтобы связать кнопки регистрации через сервисы с базой данных, необходимо создать файл [provider].jsx (рис. 66).

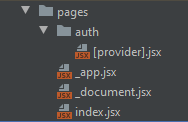


Рис. 66. [provider].jsx

Он должен содержать следующий код (рис. 67):



Рис. 67. [provider].jsx

Нужно добавить страницы, на которых сможет зарегистрироваться, либо войти в аккаунт, используя предложенные сервисы (рис. 68-69).



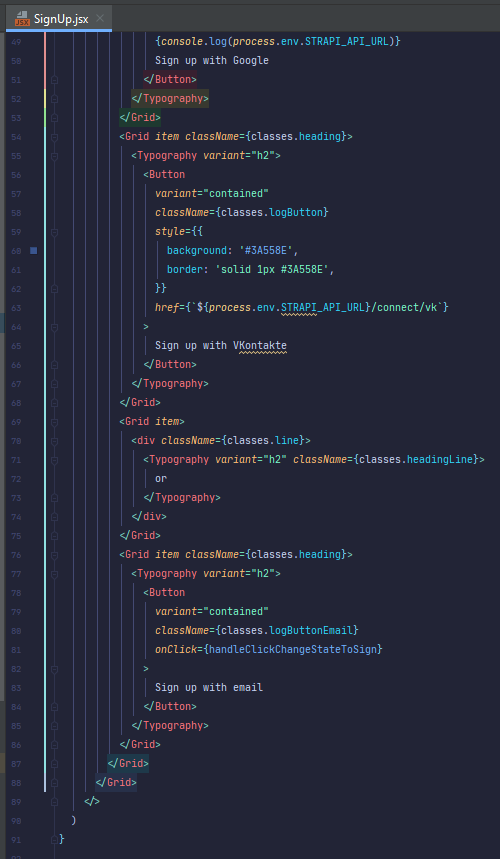


Рис. 68. Страница регистрации через сторонние сервисы



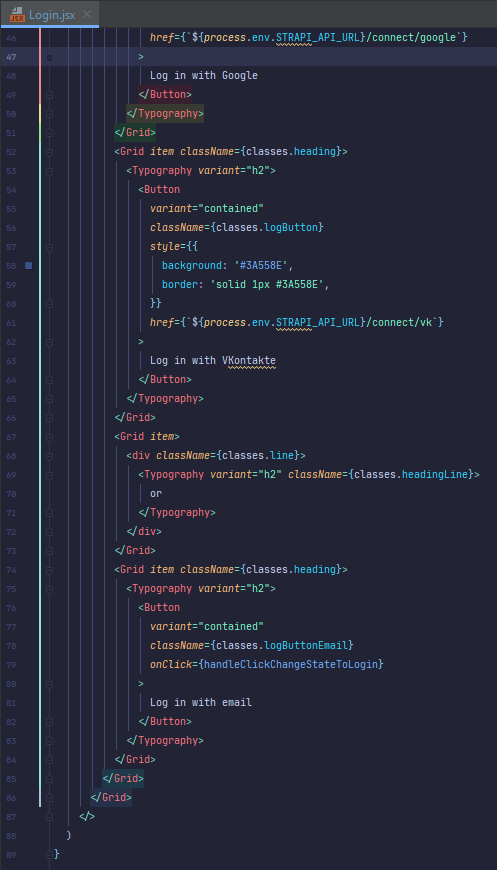


Рис. 69. Страница входа через сторонние сервисы

В итоге это выглядит так (рис. 70-71):

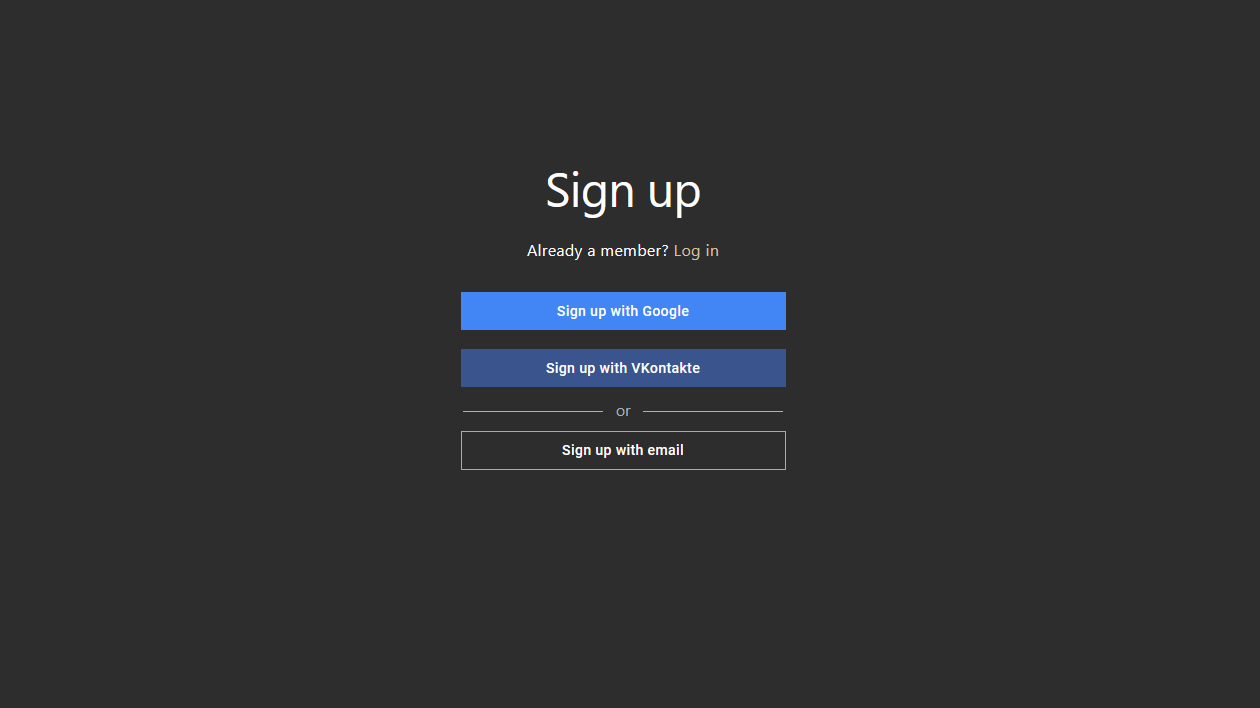


Рис. 70. Вид страницы регистрации

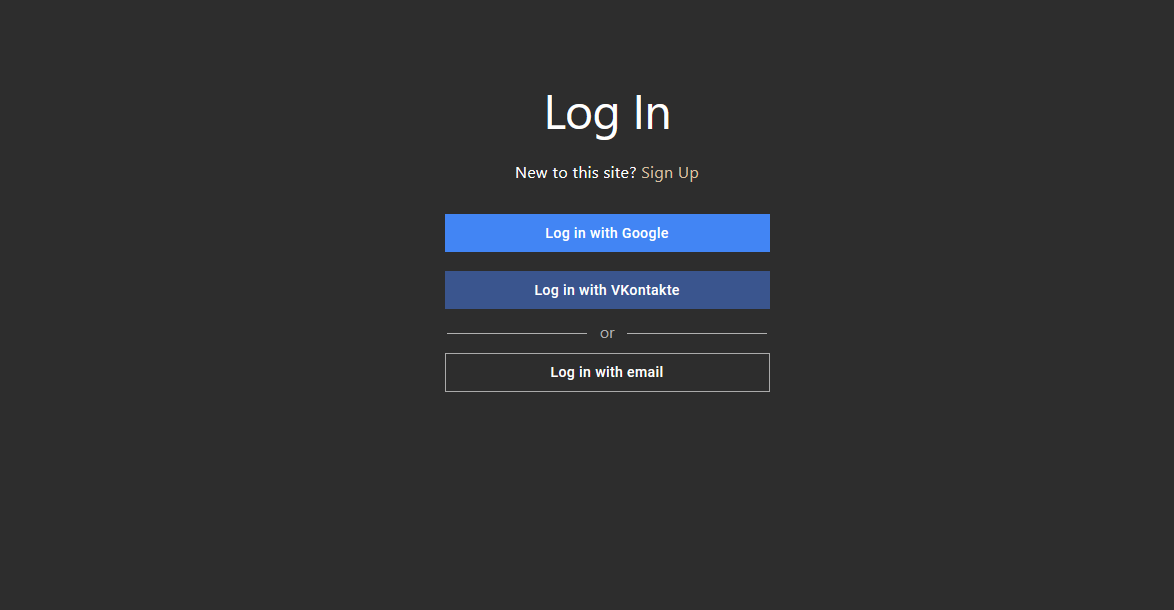


Рис. 71. Вид страницы входа

В конце была добавлена страница ввода контактных данных пользователя для связи с ним и отправки товара (рис. 72).

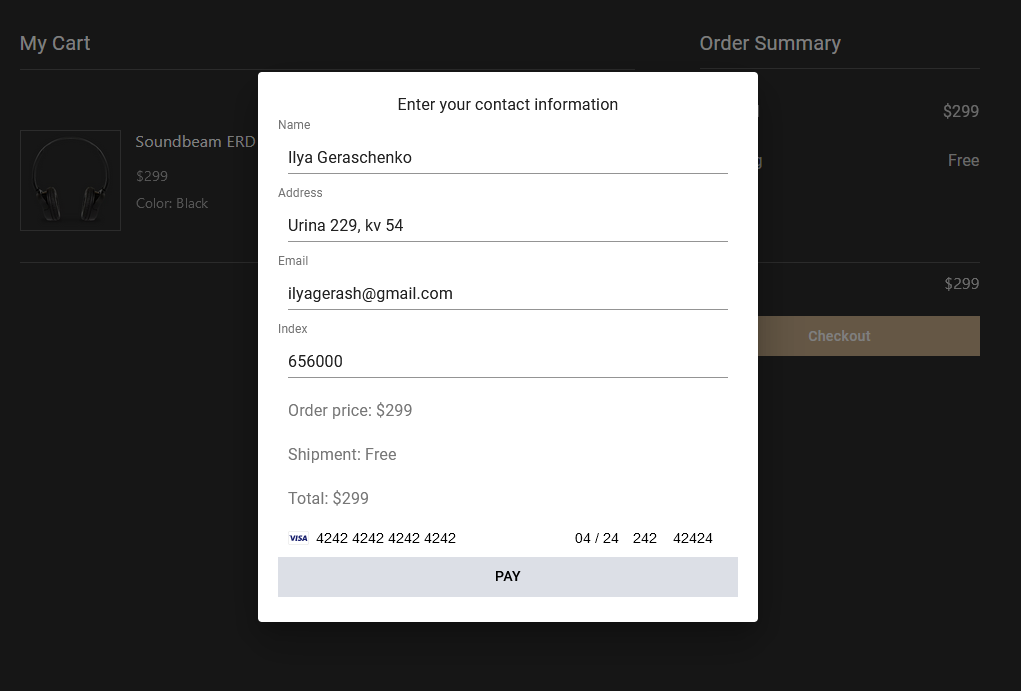


Рис. 72. Страницы ввода контактных данных

# Заключение

Итогом проделанной работы стал интернет-магазин, который отвечает современным требованиям, а главное полностью выполняет свой функционал.

На практике были использованы технологии Next.js и Strapi. Они полностью покрыли все потребности, и позволили с легкостью и в небольшие сроки разработать сайт, а это значит, что, выдвинутая в начале гипотеза, подтвердилась.

В заключение можно отметить, что технологии, которые применялись, продолжают активно развиваться и совершенствоваться, а значит, в будущем разработка веб-приложений на их основе будет лишь упрощаться, а Next.js и Strapi все чаще будут внедряться в проекты различного рода и сложности.

# Список литературы

1. https://ru.wikipedia.org/wiki/Интернет-магазин
2. https://www.imagecms.net/blog/e-commerce/shema-raboty-internet-magazina
3. https://neoseo.com.ua/dizayn-i-vybor-cms-internet-magazina
4. https://pxstudio.pw/blog/chto-takoe-next-js-i-dlya-chego-on-nuzhen
5. https://habr.com/ru/post/520440/
6. https://mcs.mail.ru/blog/osobennosti-mongodb-kogda-baza-dannyh-vam-podhodit