МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Скриптовые языки программирования»

Тема «Web-сайт «Интернет-магазин электроники»»

**Исполнитель**

студент(ка) 2 курса 2 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.А. Хващевский

подпись, дата

**Руководитель**

ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Кантарович

должность, ученая степень, ученое звание подпись, дата

Допущен(а) к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Кантарович

подпись дата инициалы и фамилия

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра информационных систем и технологий

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_ В.В. Смелов

подпись инициалы и фамилия

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

**ЗАДАНИЕ**

**на курсовое проектирование**

по дисциплине«Скриптовые языки программирования»

Курс \_\_\_ Группа: \_\_\_\_ Специальность: 1-40 05 01 «Информационные системы и технологии»

Студент: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Тема: Web-сайт «**Интернет-магазин электроники»

**2. Срок выполнения курсового проекта**: с 01 сентября 2022 г. по 16 декабря 2022 г.

**3. Технические требования :**

3.1 Работа с create-react-app, Virtual DOM, JSX:

* работа c React Hooks (применение правил использования и пишем свои);
* тестирование компонент с помощью Jest и ReactTestingLibrary;
* подключение стилей с css modules;
* переиспользование кода с помощью наследования, декораторов и кастомных хуков;
* связь с DOM: keys & refs;
* подключение сторонних компонентов.

3.2 Использование Redux.js: функциональный подход, Redux dev tools:

* Store как иммутабельный объект: Redux + Immutable.js;
* настраивание синхронного потока с Redux, переиспользование созданных ранее компонент;
* использование React-redux для связи компонентов с логикой;
* использование мемоизированных селекторов reselect;
* применение Side-effects в Redux: создание и использование Middlewares.
* получение данных от сервера.
* использование асинхронных экшенов с redux-thunk;
* применение Redux ToolKit;

3.3 Веб-сайт должен содержать несколько веб-страниц;

3.4 Верстка сайта должна быть адаптивной и кроссбраузерной;

3.5 Для тестирования использовать Linter;

3.6 Проект и пояснения к проекту должны быть размещены на GitHub.

**4. Содержание пояснительной записки**

1. Титульный лист;

2. Задание на курсовое проектирование;

3. Введение;

4. Постановка задачи;

5. Проектирование веб-сайта;

6. Реализация структуры веб-сайта;

7. Тестирование веб-сайта;

8. Заключение;

9. Список использованных литературных источников;

10. Приложения (полный исходный текст программы разработанного приложения с подробными комментариями).

**5. Форма представления на GitHub выполненного курсового проекта:**

– теоретическая часть пояснительной записки курсового проекта должна быть представлена в формате MS Word;

– оформление записки должно быть согласно правилам;

– листинги представляются в приложении.

**Календарный план**

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта |
| 1 | Задание на курсовое проектирование. Титульный лист | 14.09.2022 |
| 2 | Введение. Постановка задачи | 22.09.2022 |
| 3 | Проектирование веб-сайта | 28.09.2022 |
| 4 | Реализация структуры веб-сайта на HTML5 и внешнего оформления на Sass/СSS3 | 09.10.2022 |
| 5 | Тестирование веб-сайта | 30.10.2022 |
| 6 | Заключение | 02.11.2022 |
| 7 | Список использованных литературных источников. Приложения | 02.11.2022 |
| 6 | Подготовка и оформление пояснительной записки курсового проекта | 03.12.2022 |
| 7 | Сдача и защита курсового проекта | 10.12.2022 |

**5. Дата выдачи задания** «01» сентября 2022 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Жиляк

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

[Введение……………………………………………………….………………..........5](#_Toc122653388)

[1. Постановка задач и обзор аналогичных решений 6](#_Toc122653389)

[1.1 Актуальность задачи 6](#_Toc122653390)

[1.2 Обзор аналогичных решений 6](#_Toc122653391)

[1.3 Постановка задачи 8](#_Toc122653392)

[1.4 Вывод по разделу 9](#_Toc122653393)

[2. Проектирование приложения 10](#_Toc122653394)

[2.1 Выбор средств реализации 10](#_Toc122653395)

[2.2 Проектирование UML-диаграммы 10](#_Toc122653396)

[2.3 Проектирование базы данных 11](#_Toc122653397)

[2.4 Проектирование страниц 14](#_Toc122653398)

[2.5 Выбор стилевого оформления 16](#_Toc122653399)

[2.6 Разработка пользовательских элементов 17](#_Toc122653400)

[2.7 Вывод по разделу 18](#_Toc122653401)

[3. Реализация приложения 19](#_Toc122653402)

[3.1 Реализация серверной части 19](#_Toc122653403)

[3.2 Реализация клиентской части 21](#_Toc122653404)

[3.3 Реализация работы с базой данных 27](#_Toc122653405)

[3.4 Вывод по разделу 27](#_Toc122653406)

[4. Тестирование приложения 29](#_Toc122653407)

[4.1 Адаптивность приложения 29](#_Toc122653408)

[4.2 Ручное тестирование 31](#_Toc122653409)

[4.3 Вывод по разделу 34](#_Toc122653410)

[5. Руководство пользователя 35](#_Toc122653411)

[5.1 Заключение по разделу 37](#_Toc122653412)

[Заключение 38](#_Toc122653413)

[Список использованных источников 39](#_Toc122653414)

[ПРИЛОЖЕНИЯ 40](#_Toc122653415)

[ПРИЛОЖЕНИЕ А 40](#_Toc122653416)

[ПРИЛОЖЕНИЕ Б 43](#_Toc122653417)

[ПРИЛОЖЕНИЕ С 46](#_Toc122653418)

[ПРИЛОЖЕНИЕ D 48](#_Toc122653419)

[ПРИЛОЖЕНИЕ E 50](#_Toc122653420)

[ПРИЛОЖЕНИЕ F 52](#_Toc122653421)

[ПРИЛОЖЕНИЕ G 54](#_Toc122653422)

Введение

С каждый днем Интернет интегрируется в нашу жизнь все глубже и глубже. Практически в каждой сфере жизни присутствует Интернет. Особенно сильно он переплетается с нашей повседневной жизнью после вспышки коронавируса. Люди изолировались в домах. Они стали реже выходить в магазины и чаще стали пользоваться интернет-магазинами. Больше не нужно ехать в магазин и стоять в очередях. Все это можно получить у себя дома, всего лишь нажимая кнопки на телефоне либо же на компьютере.

Для большей заинтересованности потенциального покупателя многие сайты начали войну веб-дизайна и юзабельности. Каждое приложение стремилось стать наиболее адоптированным и приятным на вид, интуитивно понятным и производительным.

Тема моей курсовой работы «Интернет-магазин гаджетов». В этом приложении присутствует выбор различных товаров и возможность оформления покупки. Приложение должно быть максимально простым в дизайне, чтобы каждый мог зайти и сразу понять куда нажимать для достижения своих целей.

В курсовом проекте основными задачами будет создание макета приложения, структуры React с адаптивной версткой, подключение базы данных к приложению, использование сторонних библиотек для расширения функционала приложения, использование хуков для удобства в написании кода, создание функционала приложения, а именно:

* Возможность входа/регистрации.
* Добавление товара в корзину.
* Управление товарами в корзине.
* Оформление покупки.
* Верификация пароля.
* Сортировка товаров по цене и скидке.
* Сортировка товаров по выбранной категории.
* Выпадающее меню.

Целью проекта является создание удобного React-приложения для пользователей, которые смогут покупать товары через интернет.

1. Постановка задач и обзор аналогичных решений

1.1 Актуальность задачи

Актуальность интернет-магазина заключается с появлением и развитием электронных денег и электронных платежных систем. Все большее и большее людей предпочитают электронные расчеты, потому что это не только экономит много времени, позволяет производить точный расчет, но и экономит деньги, ведь цены в интернет магазине ниже, чем в обычном, это обусловлено низкими расходами на содержание интернет-магазина чем обычного.

1.2 Обзор аналогичных решений

При поисках идей для верстки сайта стоит учитывать то, что привлекательность сайта напрямую зависит от его простоты и его приятной цветовой палитрой. В качестве аналога для старта очень хорошо подошел сайт «Хайтек» [5], представленный на рисунке 1.1.

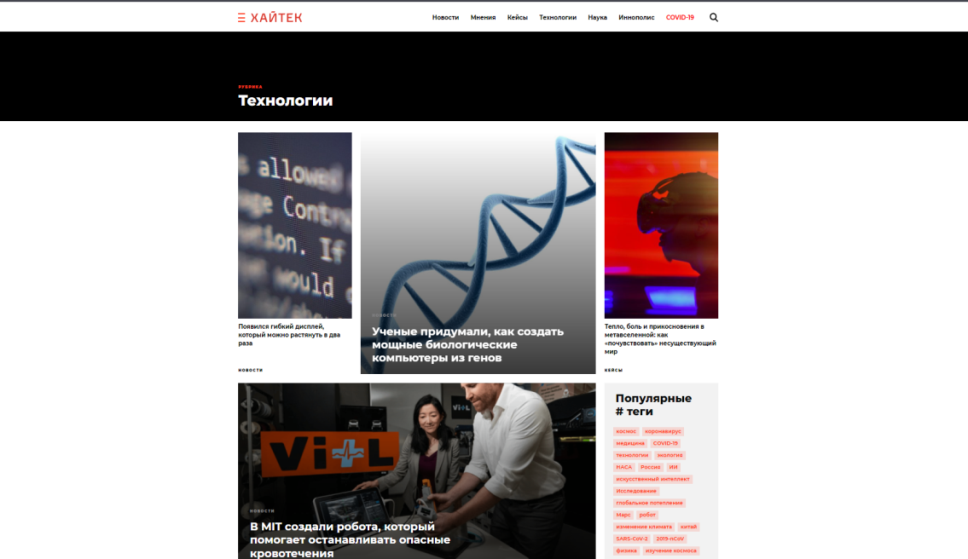


Рисунок 1.1 – Сайт «Хайтек»

Основные идеи, взятые отсюда:

* Акцентный цвет, который делает сайт запоминающимся.
* Отсутствие лишней информации.
* Хедер, представленный на рисунке 1.2.

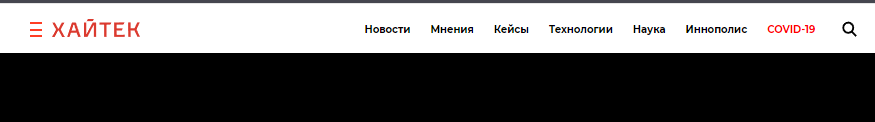


Рисунок 1.2 – Хедер сайта «Хайтек»

Основная страница со всеми товарами была оформлена, как и у большинства интернет-магазинов. Основным примером для аналога был взят сайт «5 элемент»[6] представленный на рисунке 1.3. Тут присутствует вся важная информация для того, чтобы пользователь мог по первому взгляду выбирать нужный себе товар. Так же из функционала тут присутствует сортировка по категориям, добавление в корзину, переход на страницу с просмотром информации о конкретном товаре.

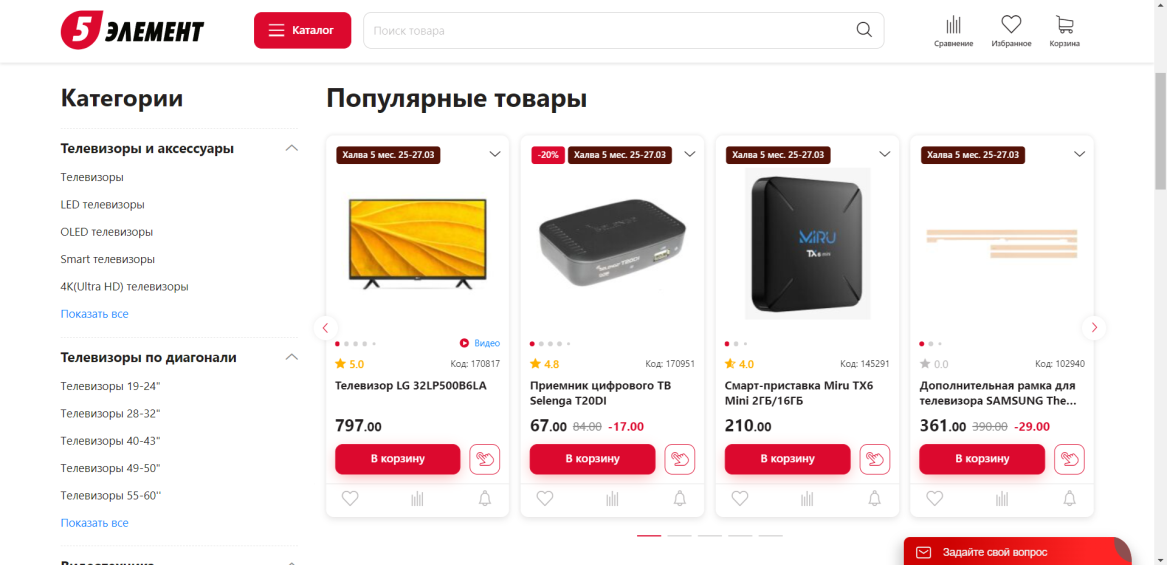


Рисунок 1.3 – Страница с товарами в магазине «5 элемент»

Страницу с подробной информацией о товарах я так же создал, используя идею аналога «5 элемент»[6], представленный на рисунке 1.4.

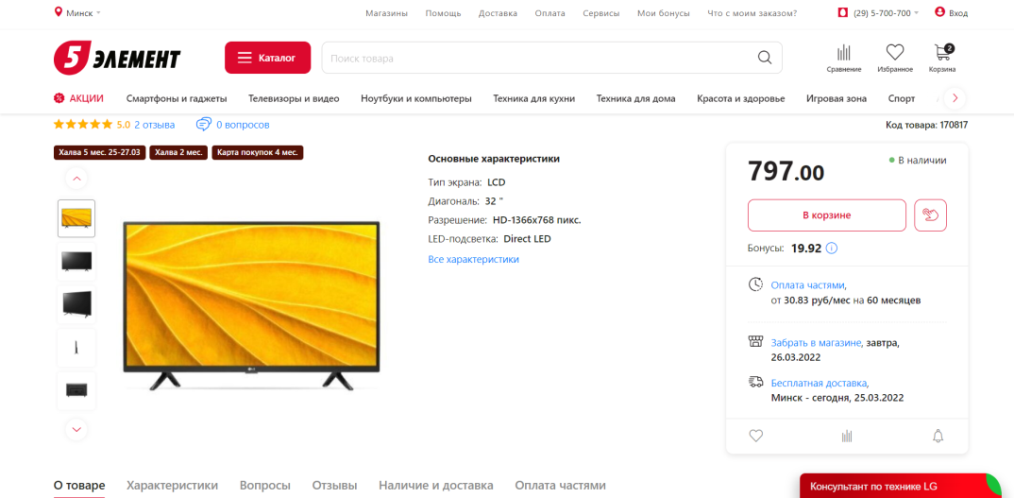


Рисунок 1.4 – Страница с товаром в магазине «5 элемент»

Тут указаны дополнительная информация, касаемо характеристик и цены товара. На этой странице минимум лишней информации и остается лишь нужная информация, которая необходима пользователю.

Страницу с корзиной была сделана, используя аналог магазина «5 элемент»[6], представленную на рисунке 1.5. Здесь присутствует возможность изменения количества товаров и оформление покупки.

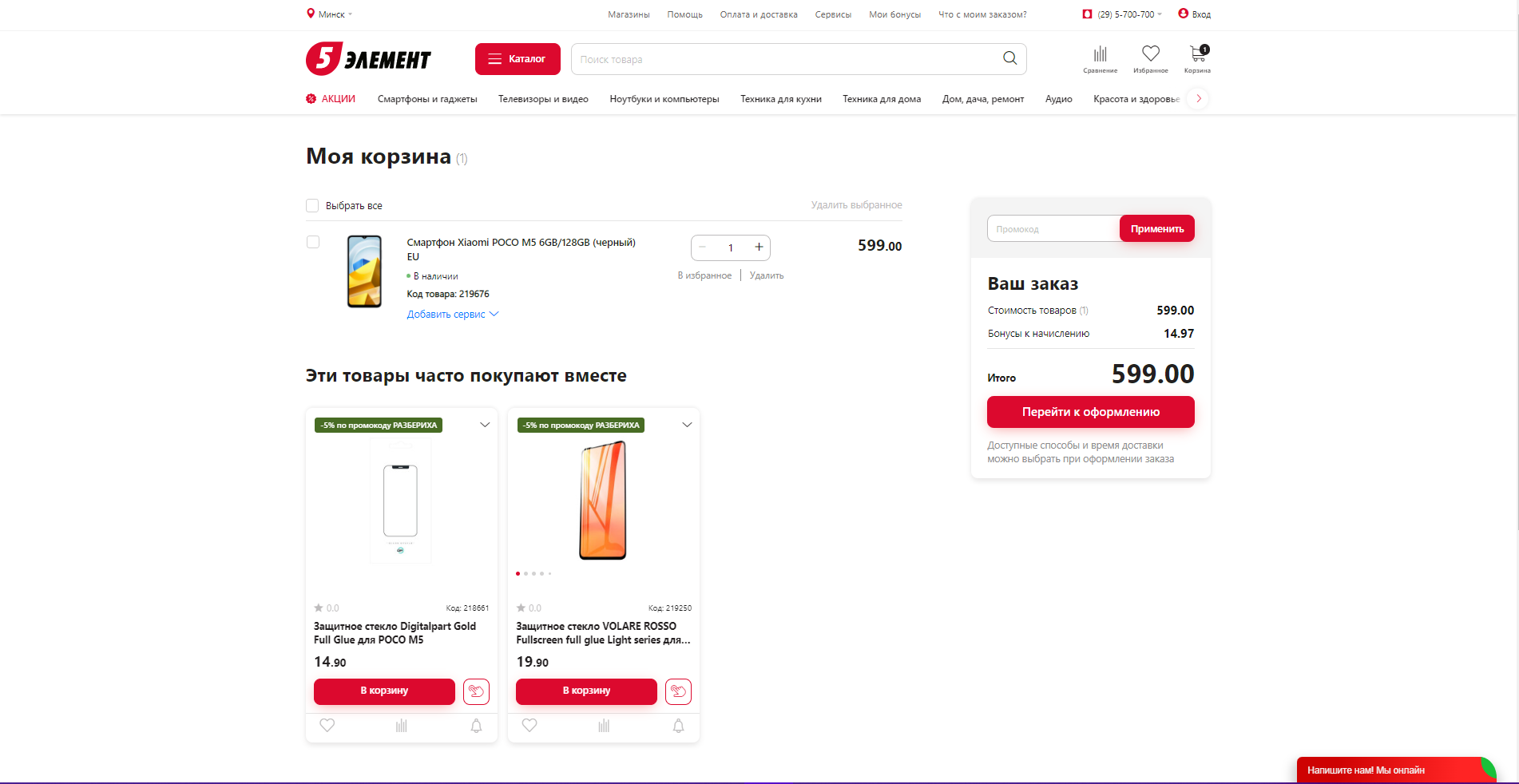


Рисунок 1.5 – Страница корзины в магазине «5 элемент»

1.3 Постановка задачи

Основной задачей будет создание приложения, в котором пользователи смогут покупать электротехнику. Для отображения товаров, возможности регистрации/входа, возможности просмотра истории покупок будет использоваться база данных PostgreSQL. Благодаря БД информация на сайте будет динамически обновляться при изменении в БД и это дает больший спектр возможности в использовании приложения.

При создании приложения будут добавлены следующий функционал: регистрация, вход, сортировка, добавление товара в корзину, оформление покупки, просмотр личной информации, включающей в себя Имя, Фамилию, количество покупок и сам список покупок.

Архитектура приложения будет состоять из трёх частей: Клиент, Сервер, База Данных.

Клиентская часть включает в себя оформление прототипа приложения, на основе которого будет создаваться основной дизайн приложения, UI-дизайн, который будет отвечать за оформление дизайна функциональной части приложения, адаптивность приложения, которая будет делать приложение максимально удобным при разных разрешениях экрана устройства.

Серверная часть включает в себя построение логики приложения, на основе которого будет разрабатываться функционал для пользователя. Основным функционалом будет возможность регистрация или вход пользователя, оформление заказа, добавление в корзину, сортировка по категориям.

База данных будет использоваться для хранения и использование данных в приложении. База будет давать возможность взятия данных о пользователе, о товаре.

1.4 Вывод по разделу

В данном разделе были поставлены задачи, которые будут выполняться в течение разработки проекта. Актуальностью проекта обусловлена необходимость разработки приложения. При обзоре аналогичных решений рассматривался функционал аналогичных интернет-магазинов электроники, из которых в проект было добавлено много полезных функций. Были затронуты средства, которыми нужно будет пользоваться в течение разработки приложения.

2. Проектирование приложения

2.1 Выбор средств реализации

При создании приложения были выбраны и использованы средства, которые описаны ниже.

Redux — это библиотека для управления состоянием приложения.

Redux Toolkit — это набор инструментов для упрощения работы с Redux. Redux Toolkit позволяет создавать более простые и понятные редьюсеры, а также упрощает работу с асинхронными запросами.

React Router — это библиотека для маршрутизации в React-приложениях. React Router позволяет создавать маршруты, которые будут отображаться в зависимости от URL-адреса. Благодаря React Router можно создавать SPA (Single Page Application), которые не перезагружают страницу при переходе по ссылкам.

Axios — это библиотека для работы с HTTP-запросами. Axios позволяет создавать запросы к API, которые будут возвращать промисы, которые можно обрабатывать с помощью async/await.

Node.js — это JavaScript-окружение, которое позволяет запускать JavaScript-код на сервере. Node.js позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

Express — это фреймворк для Node.js, который позволяет создавать сервер, который будет обрабатывать запросы от клиента и возвращать данные клиенту.

PostgreSQL — это объектно-реляционная система управления базами данных (СУБД), которая позволяет хранить данные в виде таблиц. PostgreSQL позволяет создавать базы данных, которые будут хранить данные приложения.

JSON — это текстовый формат обмена данными, который основан на JavaScript. JSON позволяет передавать данные между клиентом и сервером. JSON позволяет передавать данные в виде объектов, массивов, строк, чисел, логических значений и null.

2.2 Проектирование UML-диаграммы

В приложении пользователь может иметь одну из двух ролей: зарегистрированный пользователь, гость.

Если пользователь не зарегистрировался, то его возможностями являются:

* Регистрация (ввод данных для регистрации).
* Авторизация (ввод логина и пароля для входа, так же есть возможность выхода из аккаунта).
* Просмотр всех товаров (Сортировка по категории, сортировка по ценам и скидкам).

Если пользователь зарегистрировался и вошел в сервис, то его возможностями являются:

* Просмотр всех товаров (Сортировка по категории, сортировка по ценам и скидкам).
* Корзина (Добавление в корзину, оформление покупки, изменение количества товаров).
* Личная страница (Просмотр имени и фамилии, просмотр истории покупок).

Разграничение по ролям делает приложение более работоспособным, так как не будут возникать конфликты между ролями и ошибки с предоставлением услуг пользователю. Если пользователь попытается сделать то, что не входит в его полномочия, то система автоматически пресекает подобные действия и требует от пользователя, чтобы он зарегистрировался.

UML-диаграмма пользователя, которая показывает роли в системе и их возможности при использовании приложения представлена на рисунке 2.1.

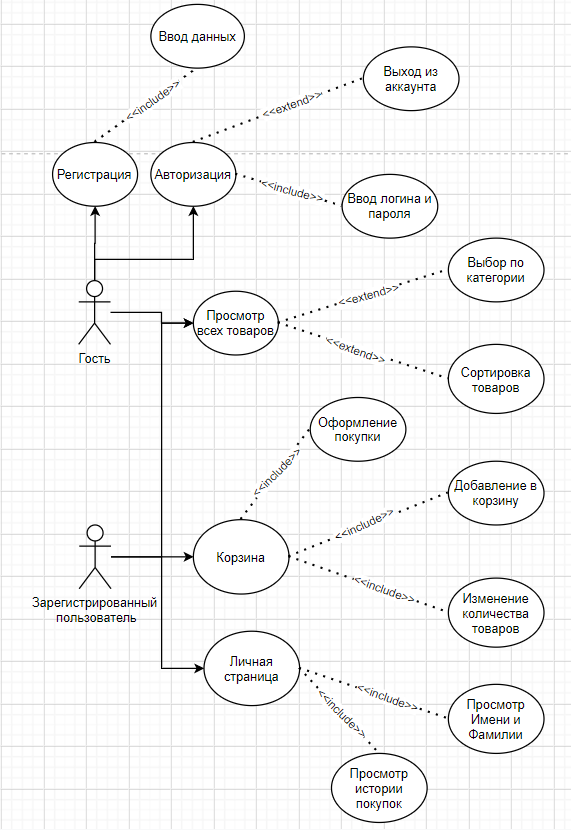


Рисунок 2.1 – UML-диаграмма приложения

2.3 Проектирование базы данных

В качестве СУБД была использована PostgreSQL, которая является одной из самых популярных в мире. Она очень удобна в использовании.

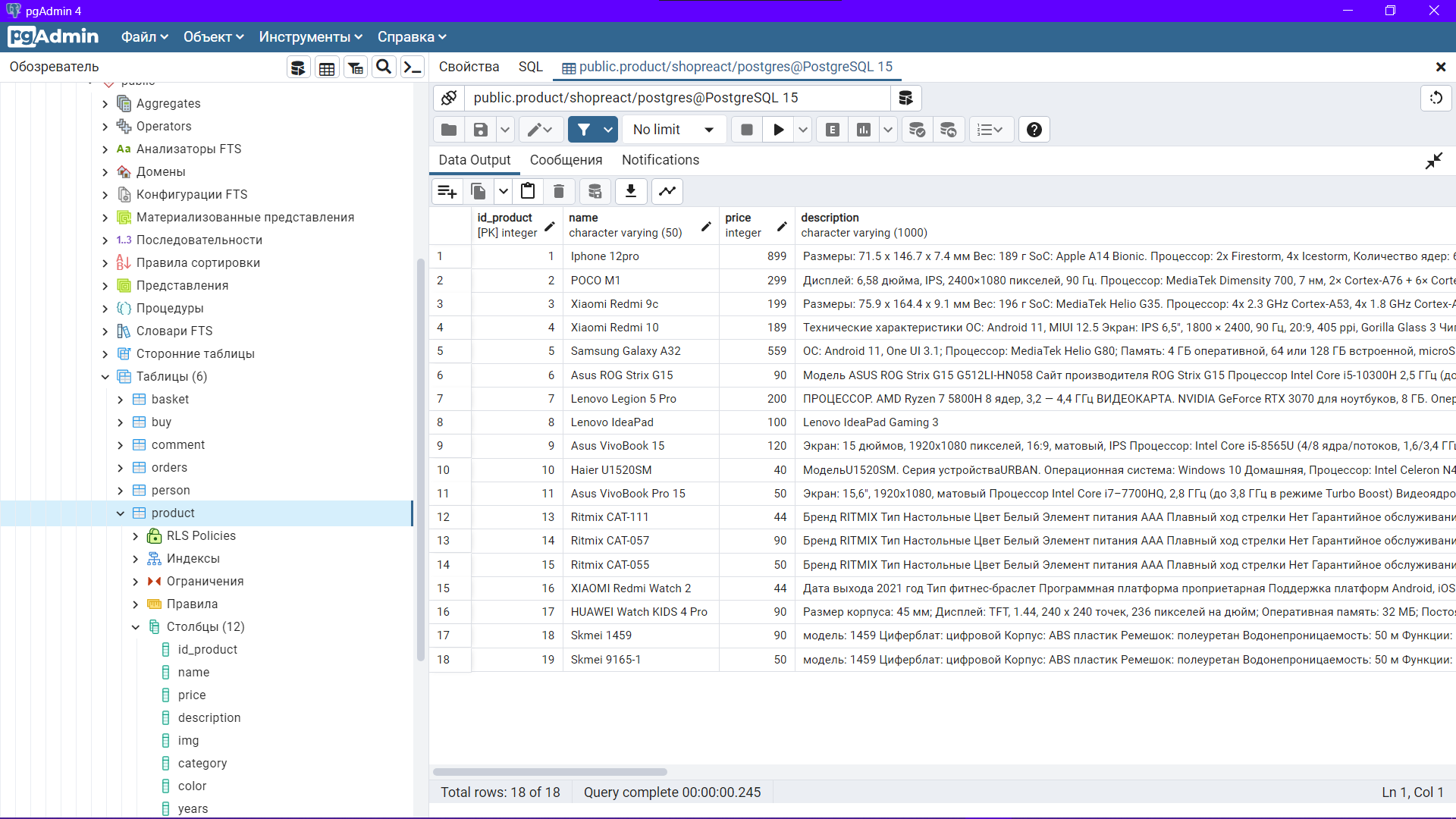


Рисунок 2.2 – PostgreSQL

Для построения базы данных изначально была построена логическая схема, которая будет визуально показывать таблицы и их связи.



Рисунок 2.3 – Диаграмма базы данных

Таблица «buy» содержит в себе информацию о покупке, которая отображается в личном кабинете пользователя.

Таблица «person» содержит в себе информацию о пользователе.

Таблица «basket» содержит в себе информацию о корзине.

Таблица «orders» содержит в себе информацию о позиции, которая отображается в корзине.

Таблица «product» содержит в себе информацию о товаре.

Получение данных из БД возможно благодаря запросам. Подключение к БД происходит благодаря библиотеке “pg”, которая отвечает за подключение сервера к БД. При подключении требуется имя пользователя, которому будет доступно подключение к бд, и по умолчанию это “postgres”, пароль от бд, хост, порт, название БД.

Для проверки получения данных было использовано приложение Postman, которые помогает отправлять и получать данные с сервера. В приложении можно ввести url для подключения к серверу, выбор режима “get”,”set”, которые отвечают за отправку и получение данных соответственно, выбор типа данных, который мы будем получать при отправке или получении.

Подключение к серверу осуществляется с помощью команды “npm dev run”, которую надо вводить в терминале программы Visual Studio Code и если выведет сообщение “server started on port 3001”, то сервер запущен и может работать с данными. Пример подключения приведен на рисунке 2.3

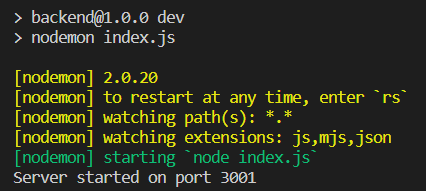


Рисунок 2.4 – Подтверждение подключения сервера к базе данных

Вся логика описана в отдельной папке “server”, структура которой представлена на рисунке 2.1

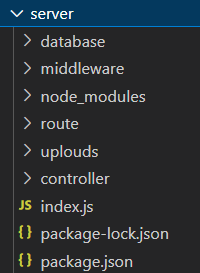


Рисунок 2.5 – Структура серверной части

В папке database прописана логика подключения логики использования базы данных к самой базе данных. В папке middleware будет прописана логика перехвата ошибок. В папке uploads хранятся изображения, которые берет база данных для предоставления их серверу. В папке controllers предоставлена логика получения информации из базы данных.

2.4 Проектирование страниц

Клиентская часть состоит из структуры приложения, внешнего вида, функционала для взаимодействия с приложением. Данная часть предназначена для установки связи между серверной частью приложения и пользователем. В клиентской части приложения установлено React-приложение.

В папке route представлена логика переадресации пользователя в приложении. Библиотека Route предоставляет возможность пользователю переключаться между страницами, предоставляя объекты, параметром которого является ссылка на нужную страницу. Так же при переходе на другую страницу предоставляет метод, который выполняется при переходе.

Структура клиентской части предоставлена ниже.

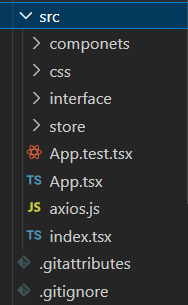


Рисунок 2.6 – Структура клиентской части

В папке components находится каталог компонентов приложения, из которых оно будет состоять. Компонентами и являются страницами приложения.

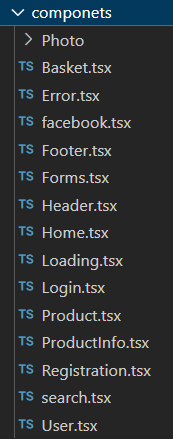


Рисунок 2.7 – Компоненты приложения

Компонент Basket является корзиной магазина. В корзине мы можем изменять количество определенного товара, удалять все товары, оформлять покупку.

Компонент Footer содержит в себе нижнюю часть приложения, в которой есть логотип, кнопки для перехода по ссылкам на соцсети, информация о компании.

В компоненте Forms содержится форма для оформления покупки. В ней нужно ввести имя, номер телефона, выбрать адрес. При желании можно отменить оформление.

В компоненте Header содержится логотип, кнопки для перехода в корзину, форму входа, личную страницу. Так же после определенного разрешения экрана все меню превращается в бургер-меню.

Компонент Home содержит в себе рекламный баннер, а так же другой компонент Products, в котором находится каталог товаров.

В компоненте Login присутствует форма для входа пользователя в случае, если он уже зарегистрирован. Данные с поля отправляются на базу данных, проверяются и позволяют пользователю войти в приложение, как зарегистрированный пользователь.

В компоненте Product реализован каталог товаров. Основные возможности при выборе товаров:

* Сортировка по цене.
* Сортировка по скидке.
* Фильтрация по типу товара.
* Фильтрация по цвету товара.
* Фильтрация по марке товара.

Так же в компоненте присутствуют карточки товаров и возможность их перелистывать в случае, если они не вмещаются в экран.

В компоненте ProductInfo присутствует информация о товаре, так же есть возможность добавления товара в корзину либо же вызвать форму для оформления покупки.

В компоненте Registration присутствует форма для регистрации, с которой введенная информация будет отправляться на сервер, а затем на базу данных. В поле с паролем нужно будет соблюдать корректность при введении, чтобы пароль не был слишком простым.

В последнем компоненте User пользователю будет дана информация о его аккаунте, количестве покупок и более подробная информация о самих покупках.

При создании приложения использовался компонентный подход программирования. В React он используется наиболее часто. Компонентный подход является основополагающим при создании приложений с помощью react. Компоненты - это главные строительные блоки, которые в своей композиции помогают реализовать сложные системы. В основе каждого компонента, в идеале, всегда лежит какой-то обособленный набор функционала, своего рода микро-решение микро-задачи. Существует несколько разных способов организации компонентов и каждый из них может быть хорош в конкретной ситуации. Все дело в том, что компоненты бывают разные, как и задачи, которые они решают. Получается, что в зависимости от функциональности компонента, его предназначения, нужно выбирать подходящий дизайн его реализации.

2.5 Выбор стилевого оформления

Сайт будет верстаться блочным способом. Этот способ наиболее актуальный на сегодняшний день. Каждый блок – элемент, в который можно вкладывать другие элементы. Блок и его содержание можно позиционировать, стилизовать при помощи CSS. Достоинством этого способа является простота кода, который легко читается. Так же благодаря неограниченности вложения элементов их можно группировать в другие блоки и работать над ними. К примеру вы хотите при нажатии на кнопку скрыть несколько элементов, но для каждого элемента писать скрипт долго. Вместо этого можно все нужные элементы вложить в другой элемент и создавать скрипт уже для общего элемента. Недостатков практически нет. Элементы div и span поддерживаются даже в старых браузерах. Блоки легко адаптировать, скрывать и изменять.

Сайт будет стилизован по идее минимализма – убрать все лишнее и выразить самое нужное. Для цветовой палитры я выбрал 3 цвета: черный(#000000), белый(#FFFFFF) и красный(#A80000). Эти цвета хорошо сочетаются. Малое количество цветов позволяет избавиться от информационного шума. Акцентный цвет на странице делает его запоминающимся и при вспоминании этого сайта сразу возникает ассоциация с акцентным цветом. Задний фон белый – это послужит лучшему выделению основной информации.

Выбор упал на шрифт “Montserrat”. Этот шрифт один из самых популярных сегодня. Он не имеет лишних элементов на символах, что создает очень хорошую читабельность сайта. Его пример на рисунке 1.5



Рисунок 2.8 – Шрифт Montserrat

Для логотипа был выбран шрифт “Quantico”. Этот шрифт при латинской раскладке придает символам строгую структуру, которая ассоциируется с технологиями, пример на рисунке 2.9



Рисунок 2.9 – Логотип

Логотип сайта напрямую связан со сферой интернет-магазина, а именно технологии. HITECH - (HIGHT TECHNOLOGY). HI выделен красным, а TECH белым, чтобы было понятно, что логотип составлен из двух слов. Логотип окрашен акцентными цветами, которые будут преследовать пользователя на протяжении всего присутствия в сайте. Дизайн логотипа максимально строгий, что ассоциируется с конструкциями различных гаджетов, а именно угловатостью.

2.6 Разработка пользовательских элементов

На первой странице присутствует форма, показанная на рисунке 2.10

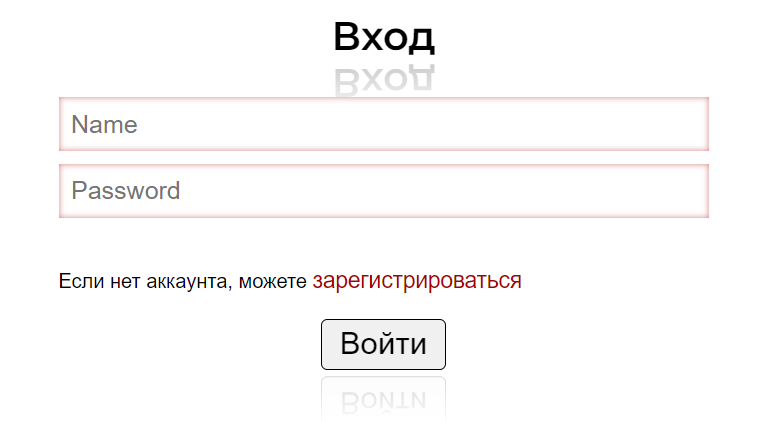


Рисунок 2.10 – Форма входа

Так же присутствует форма регистрации, представленная на рисунке 2.11

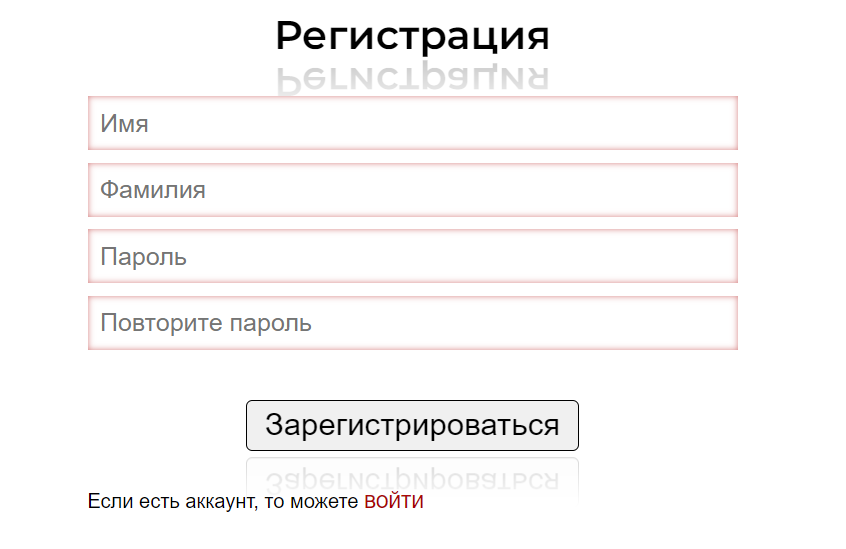


Рисунок 2.11 – Форма регистрации

На главной странице присутствует сортировка по категориям, представленная на рисунке 2.12

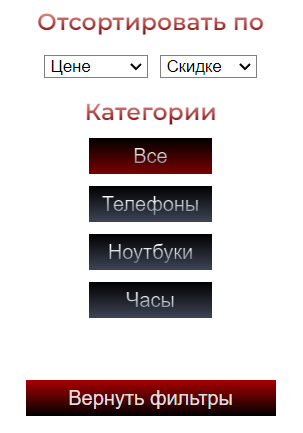


Рисунок 2.12 – Категории и сортировка

Так же присутствует каталог товаров.

Каталог товаров представлен на рисунке 2.13

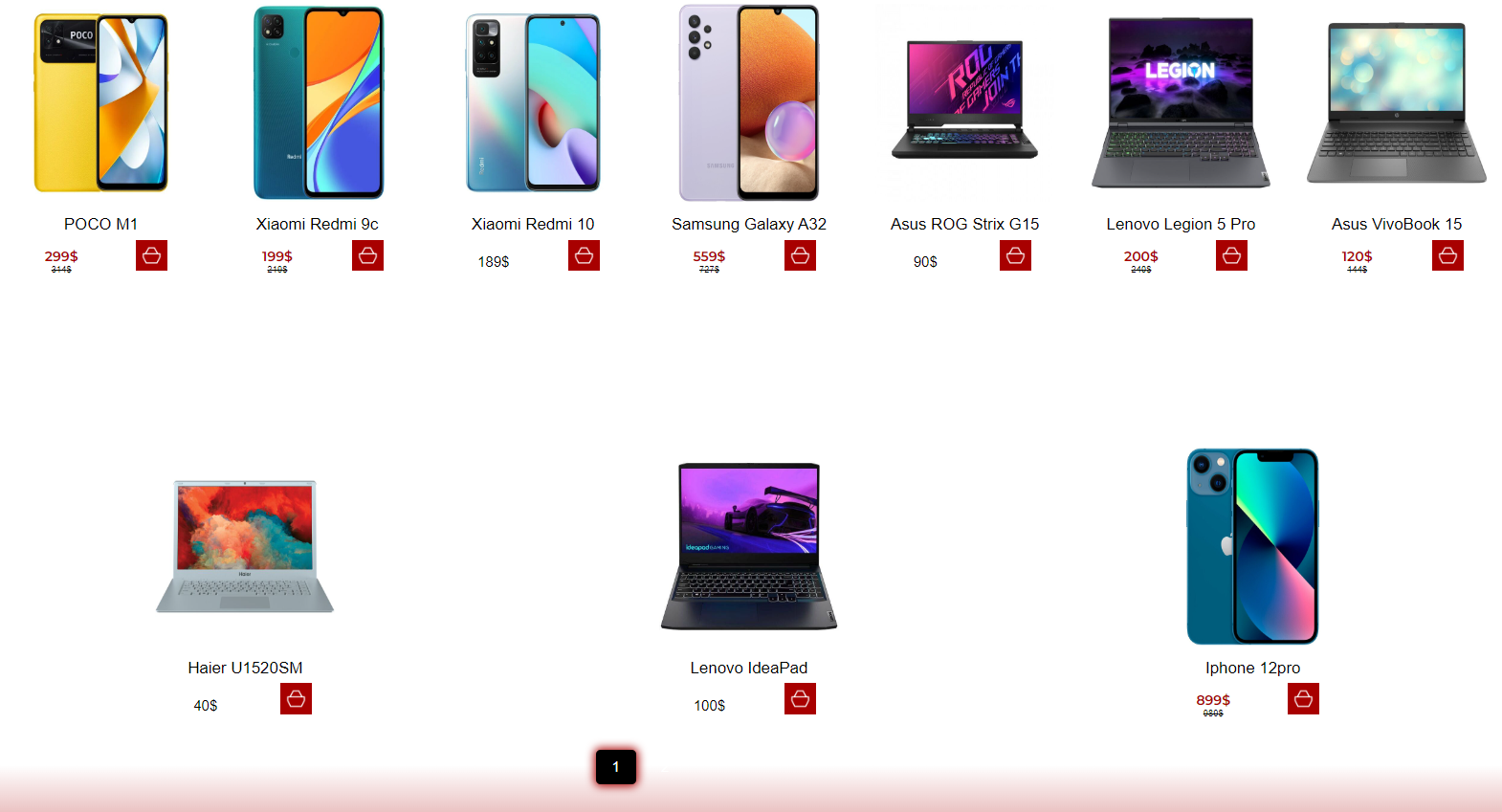


Рисунок 2.13 – Каталог товаров

Как и в любом магазине, пользователь сможет добавлять товары в корзину и использовать ее для изменения количества покупок, а так же оформление покупки товаров. Корзина представлена на рисунке 2.14, а так же оформление на рисунке 2.15.

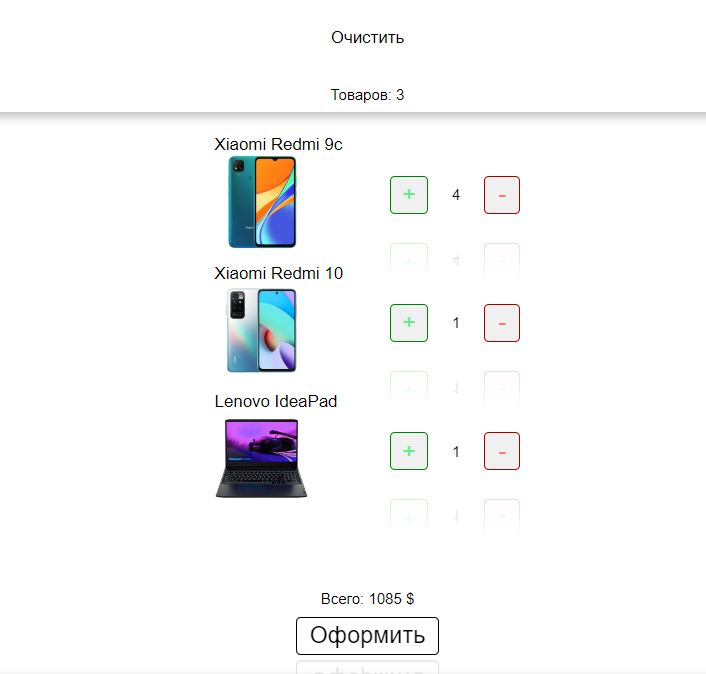


Рисунок 2.14- Корзина

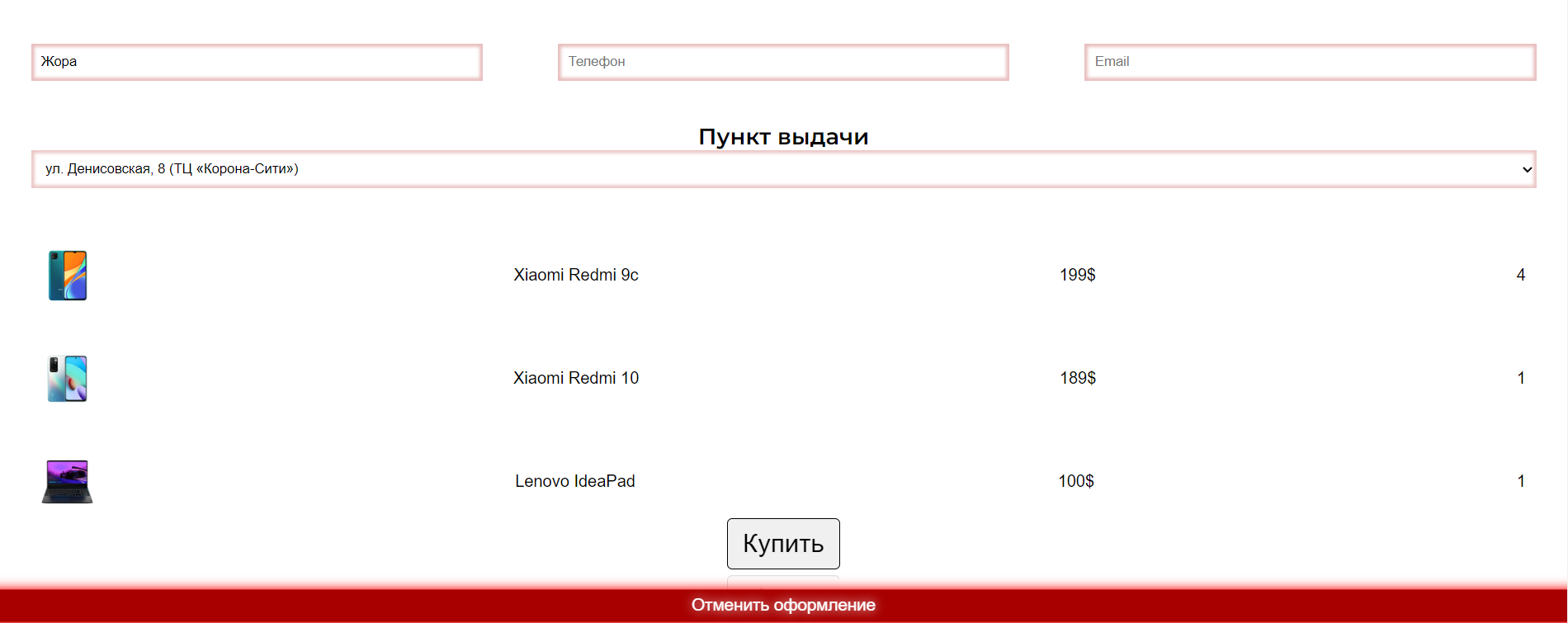


Рисунок 2.15 – Оформление покупки

Это все пользовательские функции, которые есть в приложении

2.7 Вывод по разделу

В данном разделе разрабатывалась структура приложения. Была описана диаграмма базы данных, логика страниц, функционал страниц. Был выбран стиль дизайна, который будет присутствовать на всех страницах. Каждая страница будет иметь свои свойства и функции, которыми пользователь сможет пользоваться. Отталкиваясь от данного раздела можно переходить к реализации приложения.

3. Реализация приложения

3.1 Реализация серверной части

Получение данных из БД возможно благодаря запросам. Подключение к БД происходит благодаря библиотеке “pg”, которая отвечает за подключение сервера к БД. При подключении требуется имя пользователя, которому будет доступно подключение к бд, и по умолчанию это “postgres”, пароль от бд, хост, порт, название БД. Листинг подключения предоставлен в листинге 3.1

|  |
| --- |
| import pkg from "pg";  const { Pool } = pkg;  const pool = new Pool({  user: "postgres",  password: "135790Egor",  host: "localhost",  port: 5432,  database: "shopreact",  });  export default pool; |

Листинг 3.1 – Подключение к базе данных

Для проверки получения данных было использовано приложение Postman, которые помогает отправлять и получать данные с сервера. В приложении можно ввести url для подключения к серверу, выбор режима “get”,”set”, которые отвечают за отправку и получение данных соответственно, выбор типа данных, который мы будем получать при отправке или получении.

Подключение к серверу осуществляется с помощью команды “npm dev run”, которую надо вводить в терминале программы Visual Studio Code и если выведет сообщение “server started on port 3001”, то сервер запущен и может работать с данными. Пример подключения приведен на рисунке 3.1

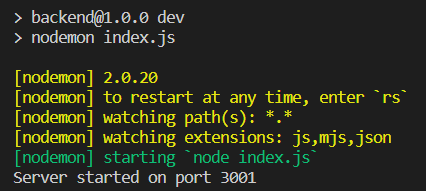


Рисунок 3.1 – Подтверждение подключения сервера к базе данных

Вся логика описана в отдельной папке “server”, структура которой представлена на рисунке 3.2

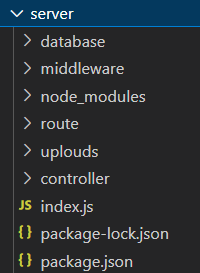


Рисунок 3.2 – Структура серверной части

В папке database прописана логика подключения логики использования базы данных к самой базе данных. В папке middle предоставления информации.

|  |
| --- |
| import jwt from "jsonwebtoken";  export const checkAuth = (req, res, next) => {  try {  const token = req.headers.authorization.replace("Bearer ", "");  const decoded = jwt.verify(token, "secret");  req.userData = decoded.name;  next();  } catch (err) {  return res.json({  message: "",  });}; |

Листинг 3.2 – Middleware

В папке uploads хранятся изображения, которые берет база данных для предоставления их серверу.

В папке controllers предоставлена логика получения информации из базы данных.

|  |
| --- |
| async GetProducts(req, res) {  try {  const products = await pool.query("SELECT \* FROM product");  var product;  const Path = "http://localhost:3001";  for (let i = 0; i < products.rows.length; i++) {  const CommentForProduct = await pool.query(  "SELECT \* FROM comment WHERE id\_product = $1",  [products.rows[i].id\_product]  );  product = {  id\_product: products.rows[i].id\_product.toString(),  name: products.rows[i].name,  img: `${Path}/${products.rows[i].category}/${products.rows[i].img}`,  price: products.rows[i].price,  category: products.rows[i].category,  color: products.rows[i].color,  description: products.rows[i].description,  sale: products.rows[i].sale,  count: products.rows[i].count,  count\_in\_shop: products.rows[i].count\_in\_shop,  unique\_property: products.rows[i].unique\_property,  };  products.rows[i] = product;  }  res.json(products.rows);  } catch (err) {  res.json({  message: "Неверные данные",});}} |

Листинг 3.3 – Метод получения информации о товарах

3.2 Реализация клиентской части

Клиентская часть состоит из структуры приложения, внешнего вида, функционала для взаимодействия с приложением. Данная часть предназначена для установки связи между серверной частью приложения и пользователем. В клиентской части приложения установлено React-приложение.

В папке route представлена логика переадресации пользователя в приложении. Библиотека Route предоставляет возможность пользователю переключаться между страницами, предоставляя объекты, параметром которого является ссылка на нужную страницу. Так же при переходе на другую страницу предоставляет метод, который выполняется при переходе.

|  |
| --- |
| import { Router } from "express";  import ProductController from "../сontroller/product.controller.js"  const router = Router();  router.get("/product/getproducts", ProductController.GetProducts);  router.get("/product/getproduct/:id", ProductController.GetProduct);  router.post("/product/addbasket", ProductController.AddToBasket);  router.delete("/product/deletebasket", ProductController.DeleteFromBasket);  router.get("/product/getbasket/:id", ProductController.GetBasket);  router.post("/product/addorder", ProductController.AddOrder);  router.get("/product/getorders/:id", ProductController.GetOrders);  router.delete("/product/deleteallbasket", ProductController.DeleteAllBasket);  router.get("/product/gettoppersons", ProductController.GetPersons);  router.put("/product/updateorder", ProductController.UpdateOrder);  export default router; |

Листинг 3.4 – Переключения между страницами

Структура клиентской части предоставлена ниже.

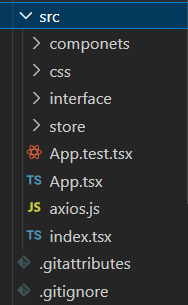


Рисунок 3.3 – Структура клиентской части

В папке components находится каталог компонентов приложения, из которых оно будет состоять.

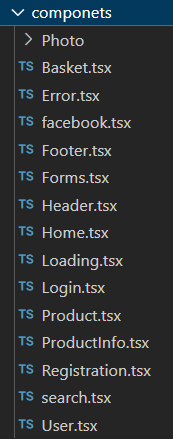


Рисунок 3.4 – Компоненты приложения

Компонент Basket.tsx является корзиной магазина. В корзине мы можем изменять количество определенного товара, удалять все товары, оформлять покупку. Листинг компонента предоставлен в приложении А.

Компонент Footer содержит в себе нижнюю часть приложения, в которой есть логотип, кнопки для перехода по ссылкам на соцсети, информация о компнии.

|  |
| --- |
| const Footer: FC = () => {    return (  <div className='footer'>  <div className="wrapper">  <div className="logo">  <Link to={'/'}> <img id='logo' src={require( "./Photo/hitech.png")}/></Link>  </div>  <div className="adress\_contact">  <div className="adress">  <h3>Адрес</h3>  <p>  Беларусь, г.Минск, улица Фогеля, 1в,183  </p>  </div>  <div className="contact">  <h3>Контакт</h3>  <p>  +375 12 123 12 12  </p>  </div>  </div>  <div className="contacts">  <button className="button"><i id='fotlog facebook' className="button\_\_icon fab fa-facebook fa-2x">f</i></button>  <button className="button"><i id='fotlog' className="button\_\_icon fab fa-twitter ">VK</i></button>  <button className="button"><i id='fotlog' className="button\_\_icon fab fa-instagram ">TG</i></button>  </div>  </div>  </div > )}  export default Footer |

Листинг 3.5 – Структура компонента Footer

В компоненте Forms содержится форма для оформления покупки. В ней нужно ввести имя, номер телефона, выбрать адрес. При желании можно отменить оформление. Листинг Forms предоставлен в приложении Б.

В компоненте Header содержится логотип, кнопки для перехода в корзину, форму входа, личную страницу. Так же после определенного разрешения экрана все меню превращается в бургер-меню. Листинг Header предоставлен в приложении C.

Компонент Home содержит в себе рекламный баннер, а так же другой компонент Products, в котором находится каталог товаров. Листинг содержится в приложении D

В компоненте Login присутствует форма для входа пользователя в случае, если он уже зарегистрирован. Данные с поля отправляются на базу данных, проверяются и позволяют пользователю войти в приложение, как зарегистрированный пользователь. Листинг Login предоставлен в приложении E.

В компоненте Product реализован каталог товаров. Основные возможности при выборе товаров:

* Сортировка по цене.
* Сортировка по скидке.
* Фильтрация по типу товара.
* Фильтрация по цвету товара.
* Фильтрация по марке товара.

Так же в компоненте присутствуют карточки товаров и возможность их перелистывать в случае, если они не вмещаются в экран. Листинг Product предоставлен в приложении E.

В компоненте ProductInfo присутствует информация о товаре, так же есть возможность добавления товара в корзину либо же вызвать форму для оформления покупки.

|  |
| --- |
| return (  <div className='animate'>  <div className="wrapper product\_fulss">  <div className="block\_img">  <h1 className="title">  {productID?.name}  </h1>  <img src={productOne?.img} alt="" />  </div>  <div className="block\_info">    <div className="description\_full">  {productID?.sale !== 0 ? (<p>Скидка: <span>{productID?.sale}%</span> </p>) : (  <></>  )}  <p>Категория: <span>{productID?.category}</span> </p>  <p>Год выпуска: <span>{productID?.years}</span> </p>  <p className='color'>Цвет:  <span className="block"  >{productID?.color}  </span> </p>  <p>Цена: <span>{productID?.price}$</span> </p>  <p>Описание: <span>{productID?.description}</span> </p>  </div>  <div className="block\_buttons">  <button onClick={() => AddToBasket(productID)} ><p id='categ'>Добавить в корзину</p></button>  <button onClick={() => handleBuyOrder(productID)}><p id='categ'>Купить</p></button>  </div></div>  </div>{FormBool && <Forms changeBool={changeBool}  products={[productID]}  cost={productOne?.price}/>}  <Footer /></div >)}  export default ProductInfo |

Листинг 3.6 – Структура компонента ProductInfo

В компоненте Registration присутствует форма для регистрации, с которой введенная информация будет отправляться на сервер, а затем на базу данных. В поле с паролем нужно будет соблюдать корректность при введении, чтобы пароль не был слишком простым.

|  |
| --- |
| class Search extends Component{  constructor(props:any){  super(props)  this.state={  value:''  }  this.handleSubmit = this.handleSubmit.bind(this)  this.handleChange = this.handleChange.bind(this)  }  handleChange(event:any){  this.setState({  value:event.target.value  })  }  handleSubmit(event:any){  this.setState({  value:event.target.value  })  }  render(): React.ReactNode {  return(  <div className="well-blosd">  <h3 className="lead">Quick shop</h3>  <div className="input-group">  <form onClick={this.handleSubmit} >  <input onChange={this.handleChange}  type='text'  className="form-control"  />  </form>  <span className="input-group-btn">  <button className="btn btn-default">  <span className="glyphicon glyphicon-search"></span></button></span>  </div>  </div>)}}  const mapDispatchToProps={}  export default connect(null, mapDispatchToProps)( Search); |

Листинг 3.7 – Структура компонента Registration

В последнем компоненте User пользователю будет дана информация о его аккаунте, количестве покупок и более подробная информация о самих покупках. Листинг компонента представлен в приложении F.

При создании приложения использовался компонентный подход программирования. В React он используется наиболее часто. Компонентный подход является основополагающим при создании приложений с помощью react. Компоненты - это главные строительные блоки, которые в своей композиции помогают реализовать сложные системы. В основе каждого компонента, в идеале, всегда лежит какой-то обособленный набор функционала, своего рода микро-решение микро-задачи. Существует несколько разных способов организации компонентов и каждый из них может быть хорош в конкретной ситуации. Все дело в том, что компоненты бывают разные, как и задачи, которые они решают. Получается, что в зависимости от функциональности компонента, его предназначения, нужно выбирать подходящий дизайн его реализации.

В листинге 3.8 представлен один из компонентов приложения, который отвечает за вывод ошибки в случае ошибки подключения к базе данных.

|  |
| --- |
| import { FC } from 'react'  import Footer from './Footer';  const Error: FC = () => {  return (<div className="animate">  <div className='error'>  <p className='er'>Ошибка в базе данных</p>  </div> < Footer /></div>)}  export default Error |

Листинг 3.8 – Структура компонента ошибки

3.3 Реализация работы с базой данных

Реализация работы с базой данных будет проводиться в самом React-приложении. В компонентах мы будем импортировать функции из серверной части и использовать их при необходимости взятия данных. К примеру для того, чтобы заполнить каталог товарами, был использован цикл map, пример которого представлен в листинге 3.9.

|  |
| --- |
| {state?.map((item: any, index: any) => {  const { id\_product, name, img, count } = item;  return (  <div className="product" key={index}><div className="img"><div className='name'>  <Link to={`/product/:${id\_product}`}><p>{name}</p></Link></div><Link to={`/product/:${id\_product}`}><div className='backet\_img' ></div> <img id='backet\_img' src={img} alt="" /></Link></div>  <div className="buttons\_add\_del">  <button id='plus' className='add' onClick={() => AddToBasket(item)}>+</button> <p className='count'><p>{count || 0}</p></p> <button className='add' id='minus' onClick={() => DeleteToBasket(item)}>-</button></div></div>)})} |

Листинг 3.9– Пример использования базы данных

До этого мы в начале объекта, в котором описывается структура компонента, были добавлены переменные, которые берут значения из базы данных с помощью функции useAppSelector. Код в листинге 3.10.

|  |
| --- |
| import { useAppSelector } from '../store/store';  . . .  const Basket: FC = () => {  const { id } = useParams();  const { basket, sum } = useAppSelector(state => state.product);  const { user } = useAppSelector(state => state.auth);  const [state, setState] = useState<any>([])  const [FormBool, setFormBool] = useState<boolean>(false) |

Листинг 3.10– Взятие данных из базы данных

3.4 Вывод по разделу

В данном разделе была описана реализация клиентской части, серверной части и части базы данных. В клиентской части были описаны компоненты, которые будут выводиться пользователю для того, чтобы пользователь мог работать с приложением. В серверной части была описана реализация передачи информации между пользователем и базы данных. Часть базы данных была реализована в клиентской части в виде функций, которые импортируются из серверной части и используются для передачи данных пользователю.

4. Тестирование приложения

4.1 Адаптивность приложения

Сегодня большинство пользователей предпочитают использовать мобильные устройства и с каждым днем количество таких пользователей увеличивается. Особенность мобильных устройств в том, что практически у каждой модели свое разрешение экрана. Чтобы пользователю было приятно находится в приложении, нужно сделать его адаптивным.

Для адаптивности приложения был использован инструмент @media, который задает компонентам определенные стили при определенном разрешении экрана. Наиболее часто использовалась flex-верстка, так как она адаптируется под ширину экрана.

Далее ставился вопрос о тестировании приложения в разных браузерах. При открытии в других браузерах приложение работало корректно.

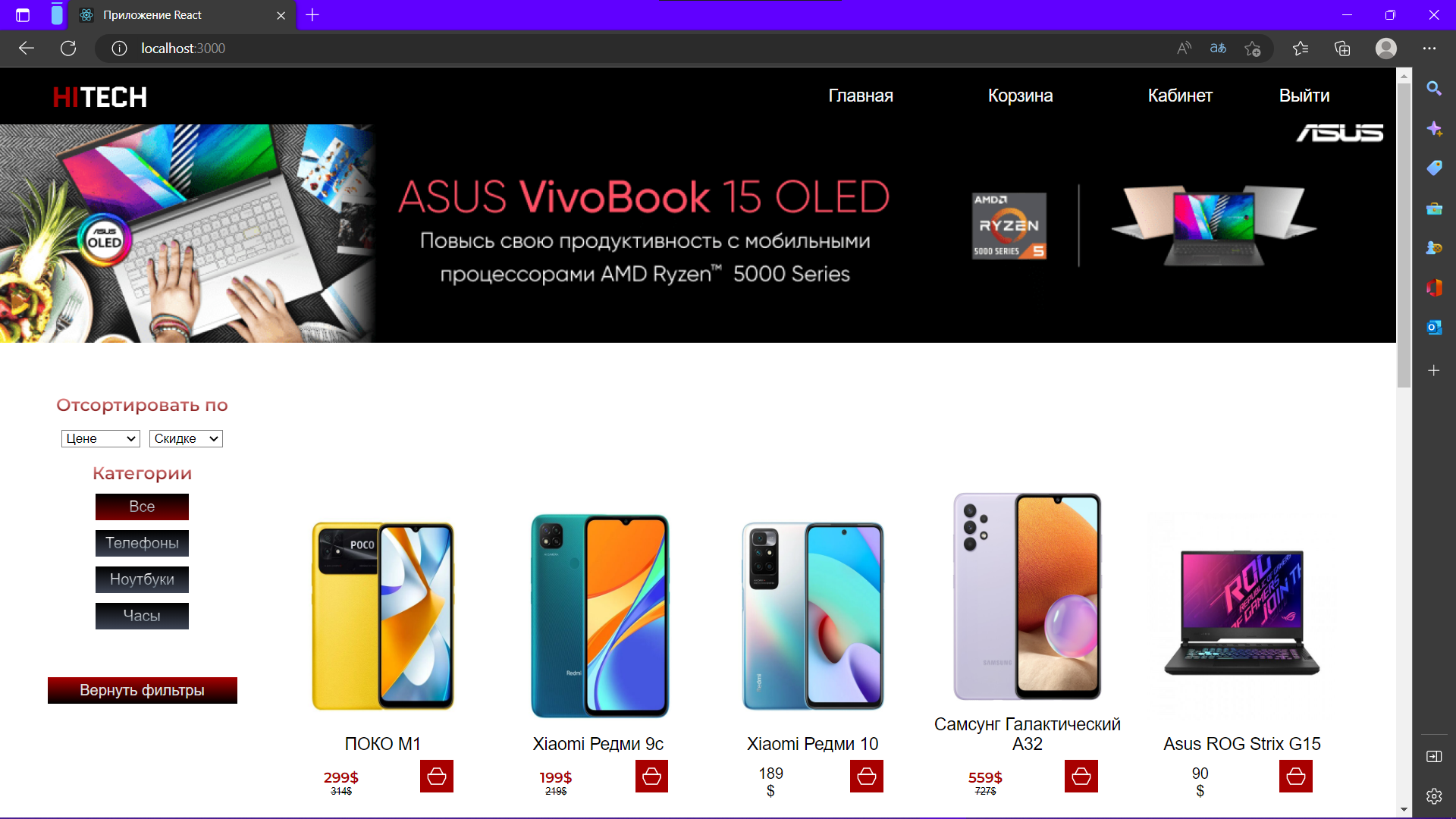


Рисунок 4.1 – Работа приложения в Explorer

Пример адаптивности страницы каталога приведен рисунке 4.2

|  |  |  |
| --- | --- | --- |
|  |  | |
|  | |

Рисунок 4.2 – Главная страница в мобильной и компьютерной версии

Пример адаптивности страницы товара приведен рисунке 4.3

|  |  |
| --- | --- |
|  |  |
|  |  |

Рисунок 4.3 – Страница товара в мобильной и компьютерной версии

Пример адаптивности страницы корзины приведен рисунке 4.4

|  |  |
| --- | --- |
|  |  |

Рисунок 4.4 – Страница корзины в мобильной и компьютерной версии

Чтобы элементы приложения выстраивались вертикально и не вылезали за рамки экрана, было использовано свойство flex-direction:column, которое выстраивает элементы в колонку и не дает им вылезти за пределы своего блока. Все кнопки были стилизированы так, чтобы при наведении на них менялись свойства. Это делается с помощью слова :hover, которое задает другие свойства объекту при наведении.

Адаптивность приложения под разные устройства является важнейшим показателем. Пользователю не будет интересен функционал приложения, красивое оформление дизайна, если приложение не можем адаптироваться под устройства любого размера. Поэтому приложение прошло проверки на адаптивность через разные браузеры и разрешения экрана.

После проверки не было замечено никаких изменений в тексте, картинках, кнопках, которые могли бы разрушить вид приложения.

4.2 Ручное тестирование

Тестирование frontend части включает в себе проверку правильности осуществления перехода между страницами, которые совершают пользователи системы, также проверку правильности отправления ajax запросов, их обработки frontend сервером, а также правильности отображение представлений.

В ручное тестирование приложения входит проверка правильности осуществления перехода между страницами, верификация пароля при вводе, чтобы пароль не был слишком простым, форматирование номера телефона после ввода, изменение личных данных пользователя и корзины при заходе на страницу под другим пользователем.

Рассмотрим верификацию пароля при регистрации. Если строка пароля будет пустая, то будет выводиться уведомление о том, что пароль нужно ввести. Если пароль слишком короткий, то будет выдавать ошибку. Пример пустой строки пароля и ошибки на рисунке 4.5.

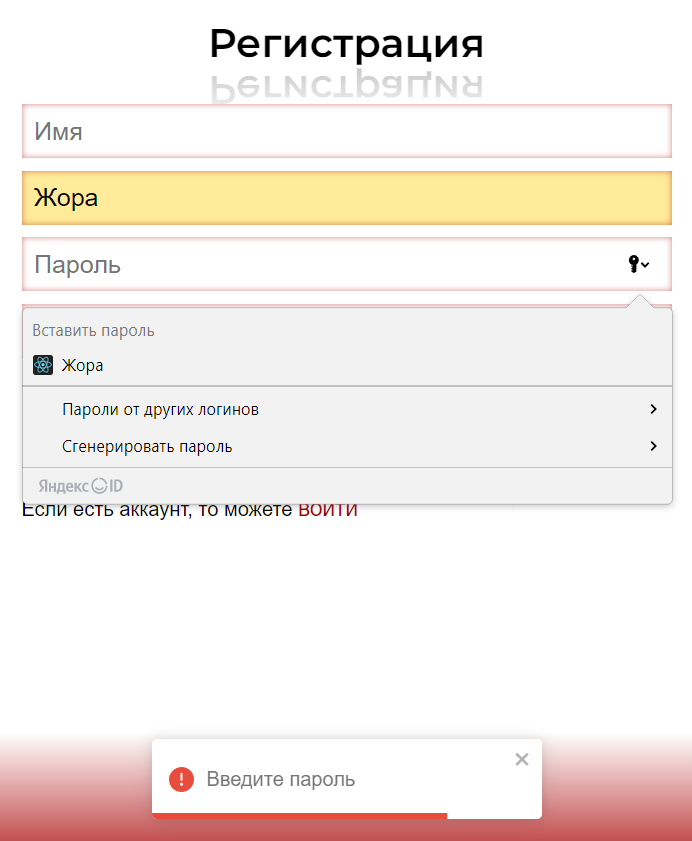


Рисунок 4.5 – Форма пароля пустая

Если строка имени или фамилии будет пустая, то будет выводиться ошибка с предупреждением. Пример на рисунке 4.6.

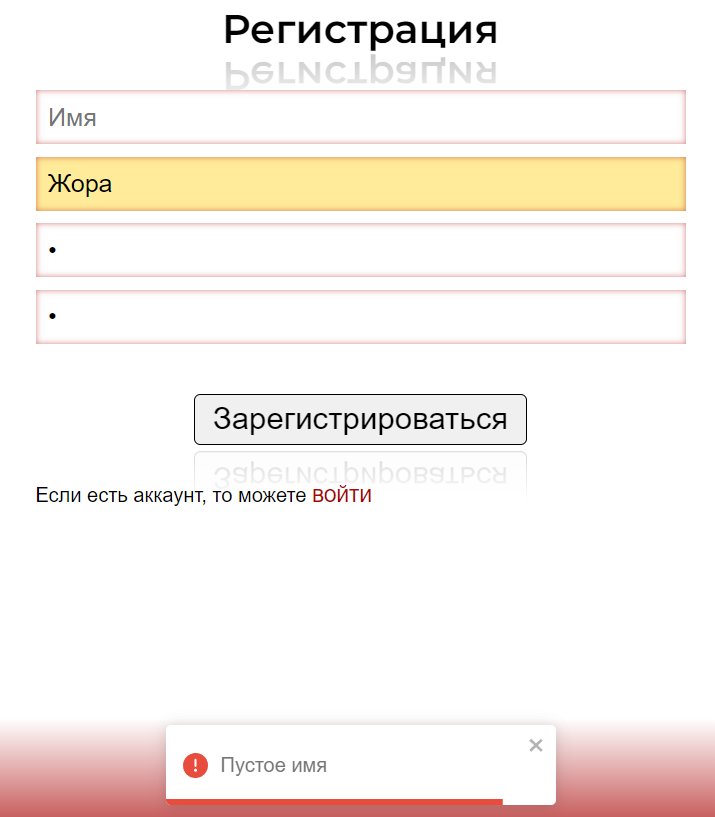


Рисунок 4.6 – Форма имени пустая

Пароль должен состоять больше чем из пяти символов, а имя больше чем из трех, иначе будет показываться ошибка, аналогична предыдущим примерам.

После заполнения формы правильными данными осуществится переход на личную страницу пользователя с уведомлением, что регистрация прошла успешно. Пример на рисунке 4.7.

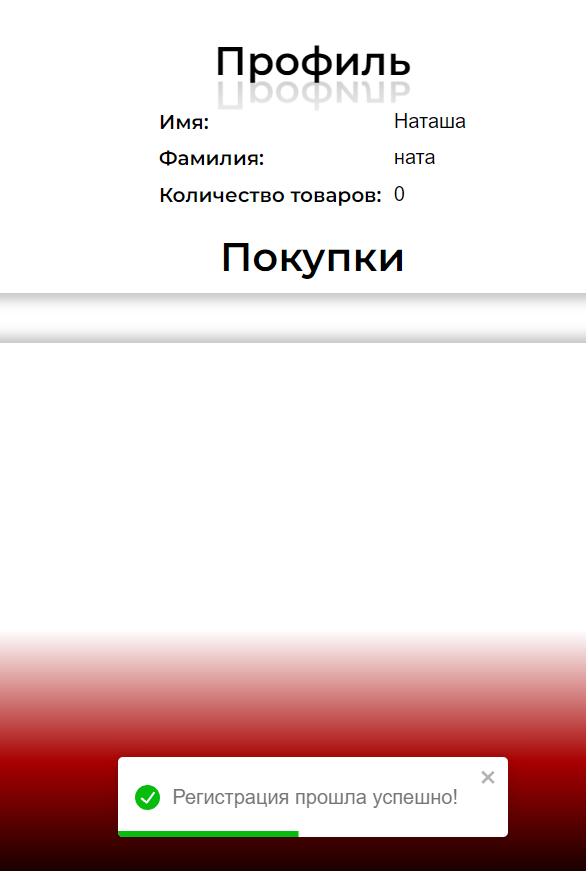


Рисунок 4.7 – Успешная регистрация

Попробуем войти под несуществующим пользователем. У пользователя это не выйдет, так как в базе данных нет такого пользователя или соответствующего к нему пароля и система выдаст ошибку входа, которая представлена на рисунке 4.8.

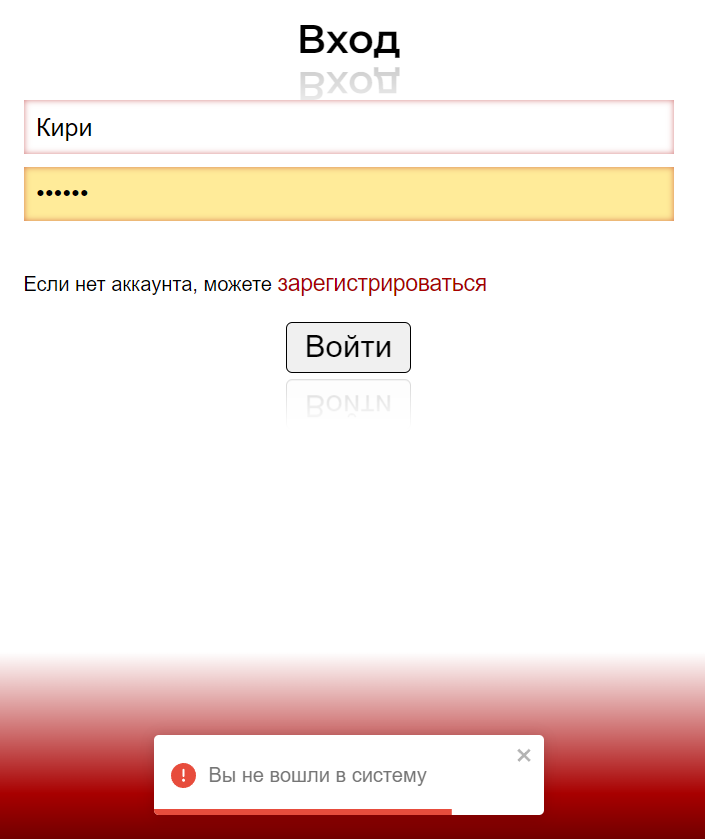


Рисунок 4.8 – Оповещение об ошибке при входе

Правильно введенные данные представлены на рисунке 4.9.

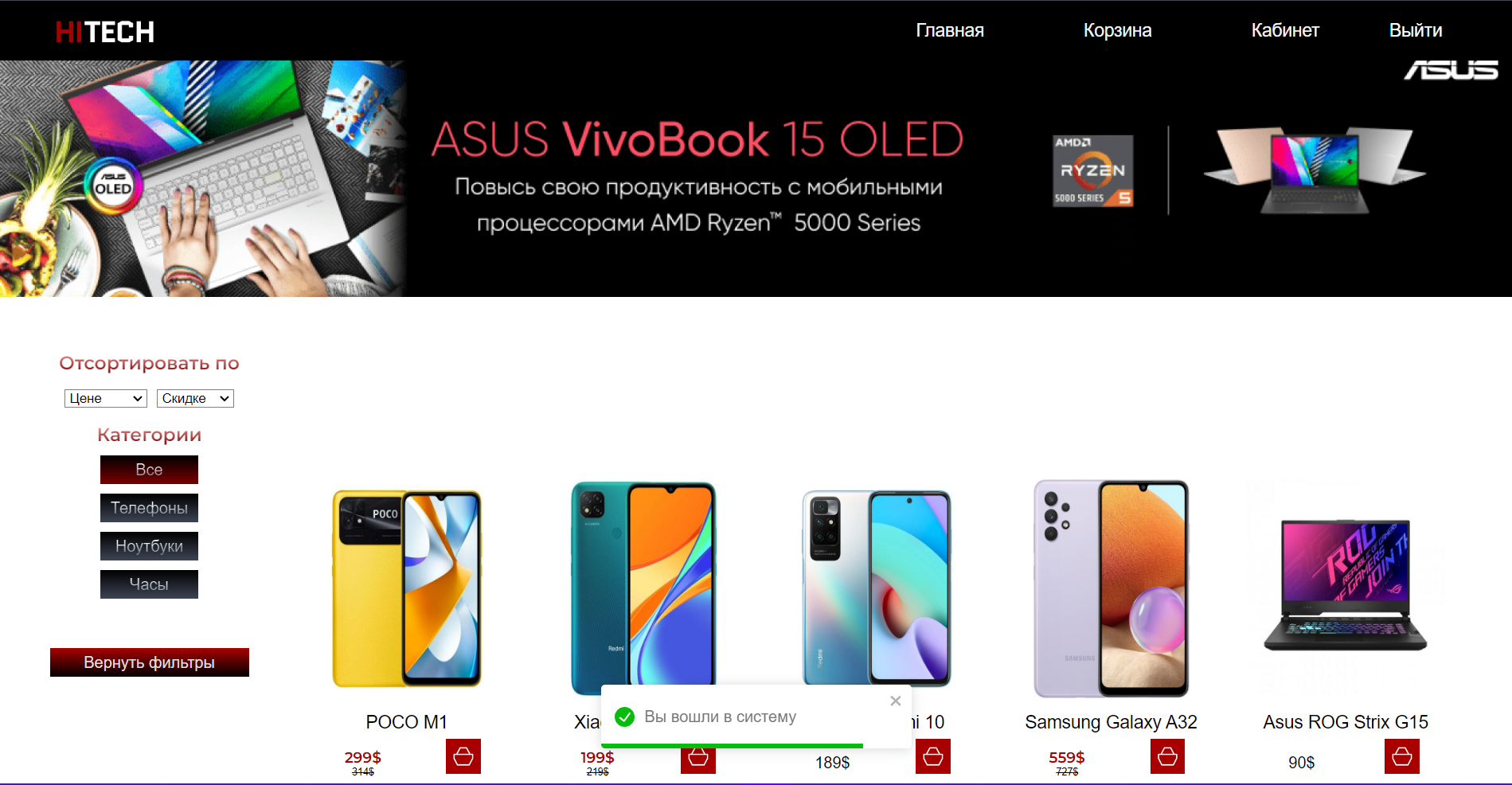


Рисунок 4.9 – Вход с правильными данными и уведомление о входе

4.3 Вывод по разделу

В данном разделе было проведено тестирование приложения. В него входило:

Проверка адаптивности.

Проверка кроссплатформенности.

Проверка работоспособности функций приложения вручную.

В итог можно вывести то, что приложение выглядит правильно при любом разрешении экрана, функции правильно выполняются в других браузерах, уведомления пользователю показывают информацию соответствующую выполненным шагам пользователя.

5. Руководство пользователя

В самом начале пользователь оказывается на главной странице и для использования приложения со всеми функциями нужно войти или зарегистрироваться. Эту возможность предоставляет кнопка “Вход” в шапке страницы в правом верхнем углу, представленной на рисунке 5.1

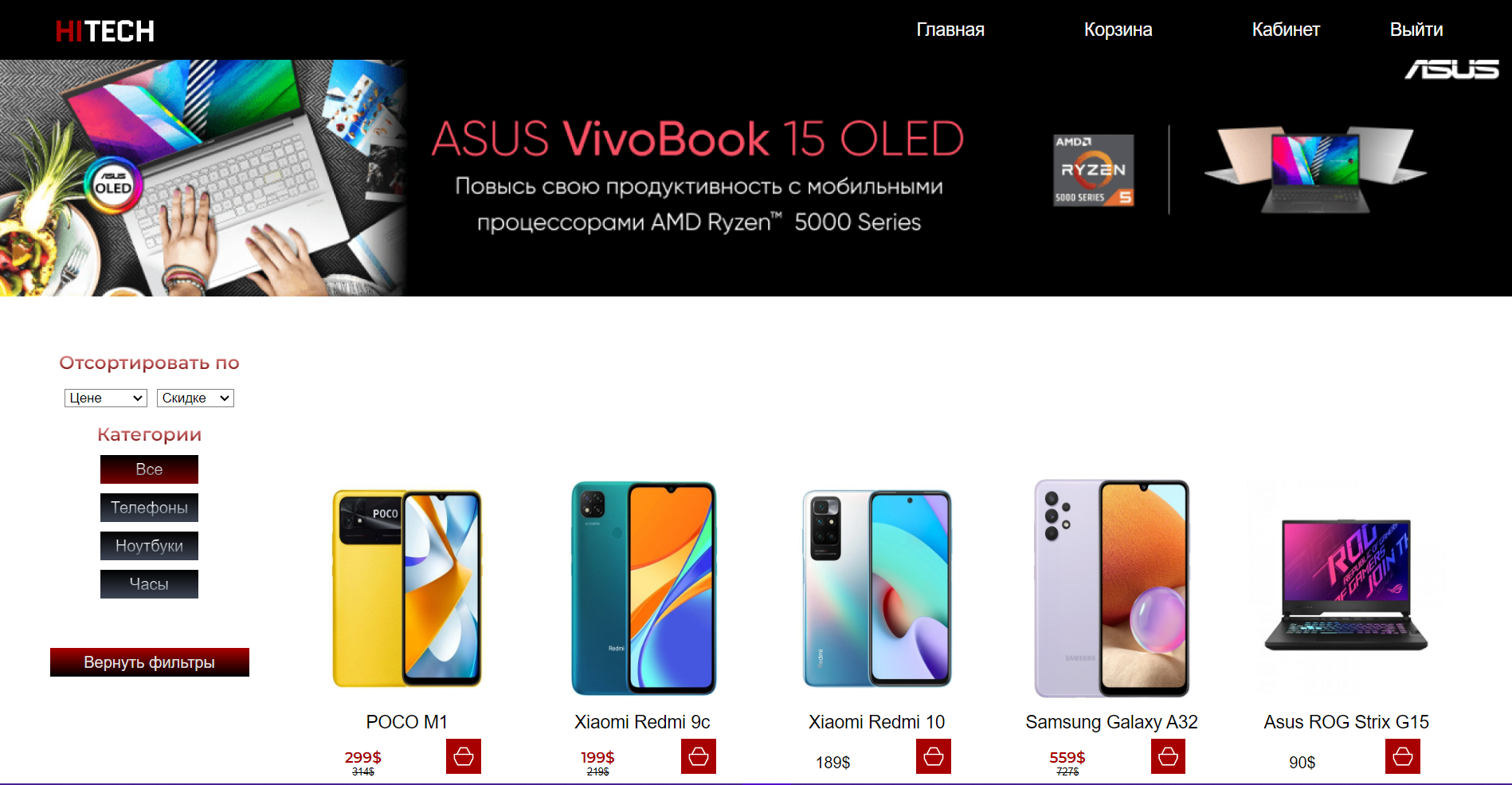


Рисунок 5.1 – Главная страница

Для входа в качестве зарегистрированного пользователя нужно ввести данные в форме страницы входа, представленной на рисунке 5.2

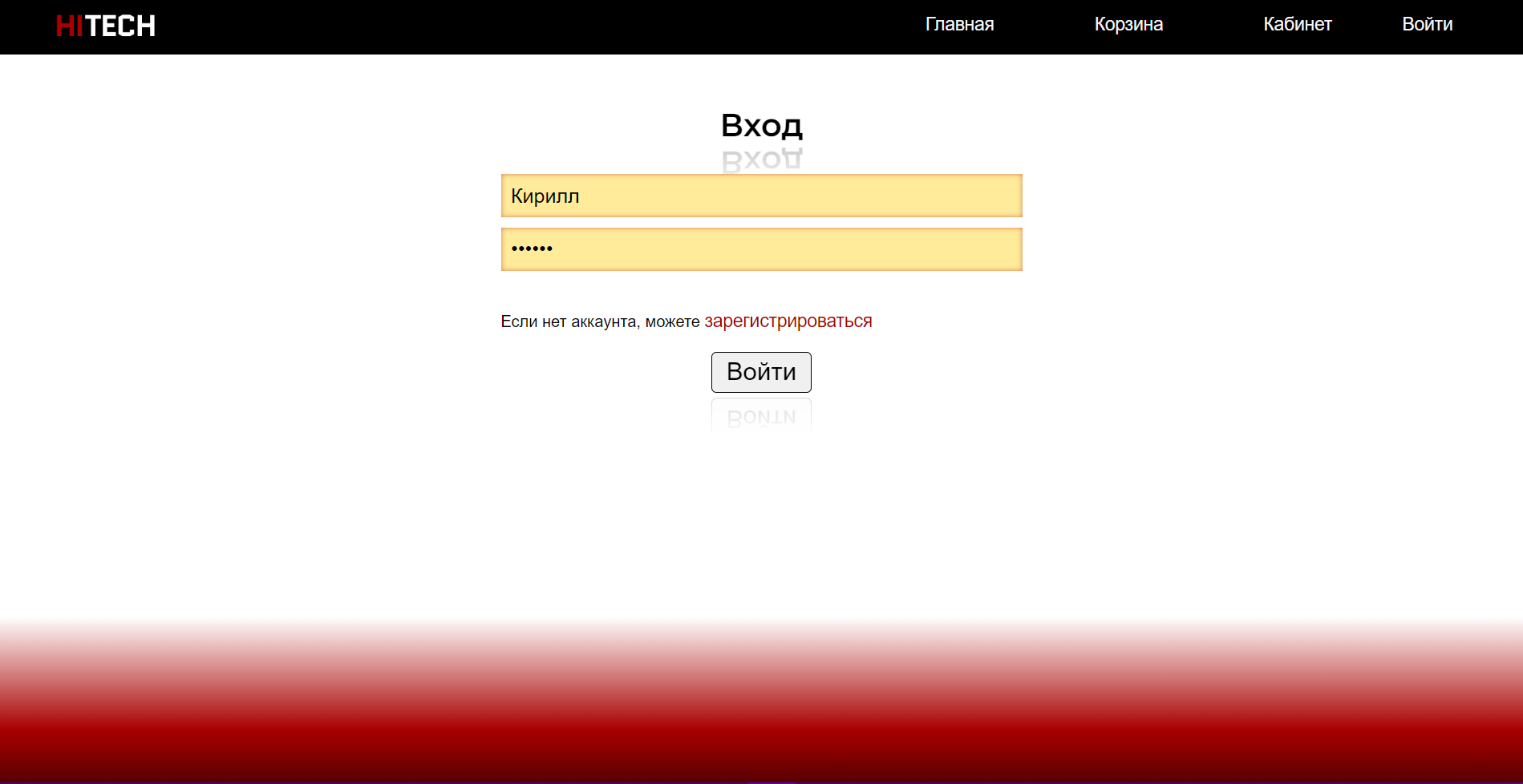


Рисунок 5.2 – Страница входа

Если пользователь не зарегистрирован, то он может зарегистрироваться, нажав на слово “зарегистрироваться”, которое перенаправит пользователя на страницу с регистрацией, представленной на рисунке 5.3.

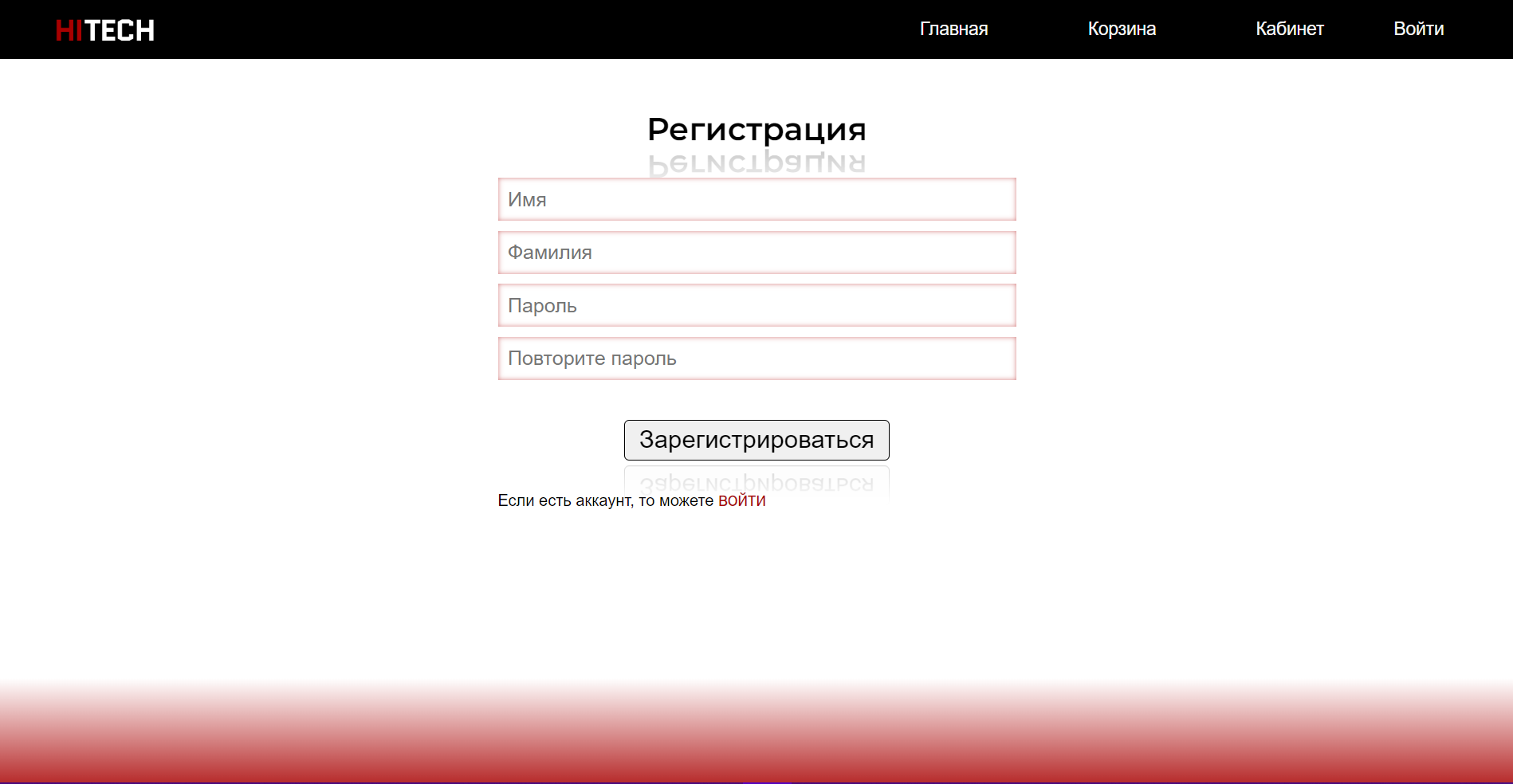


Рисунок 5.3 – Страница регистрации

После успешной регистрации пользователю будет открыт доступ к личному кабинету, работе с корзиной, просмотр информации о товаре, которые представлены на рисунках 5.4, 5.5, 5.6 соответственно.

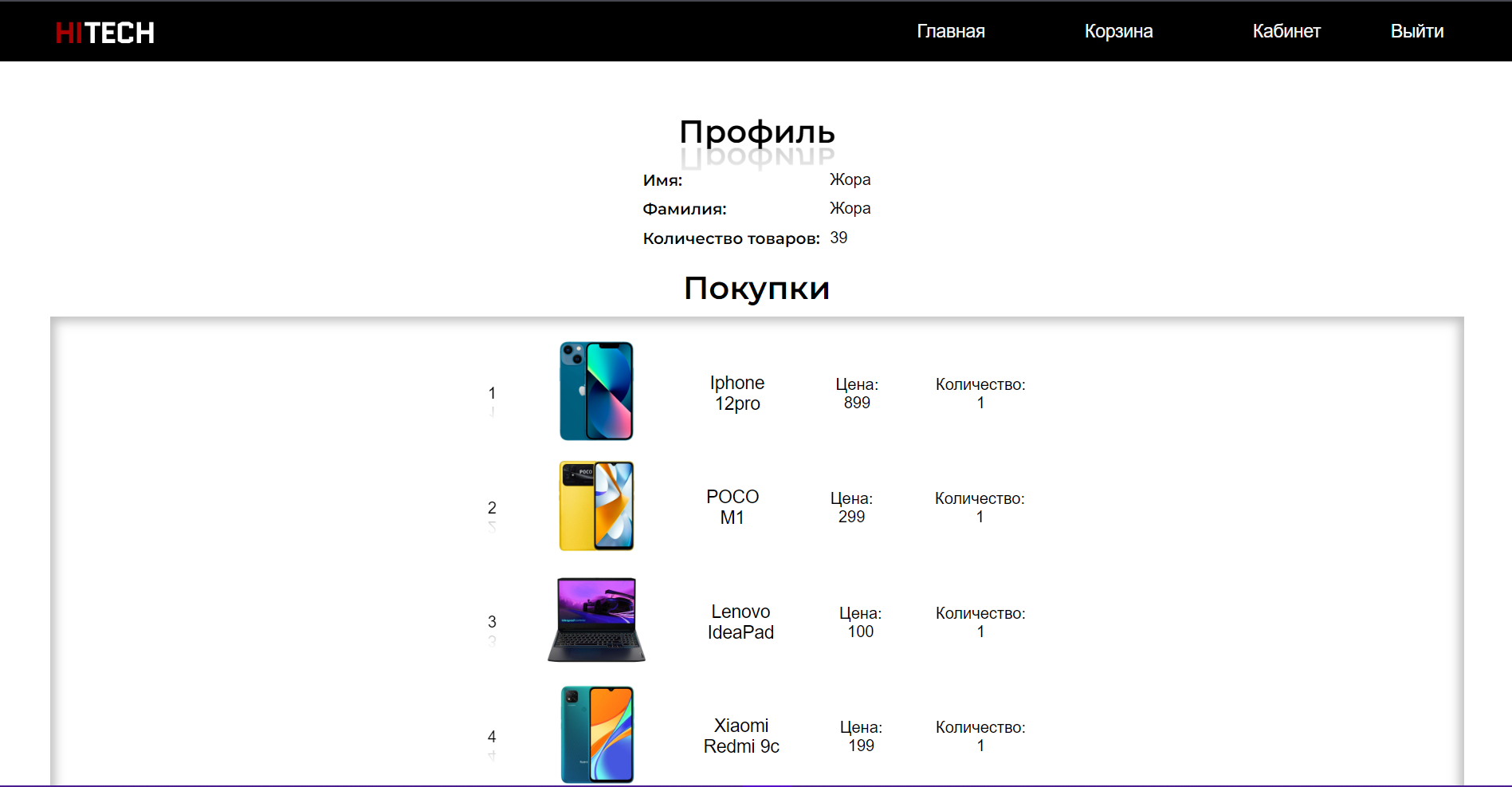


Рисунок 5.4 – Личная страница пользователя

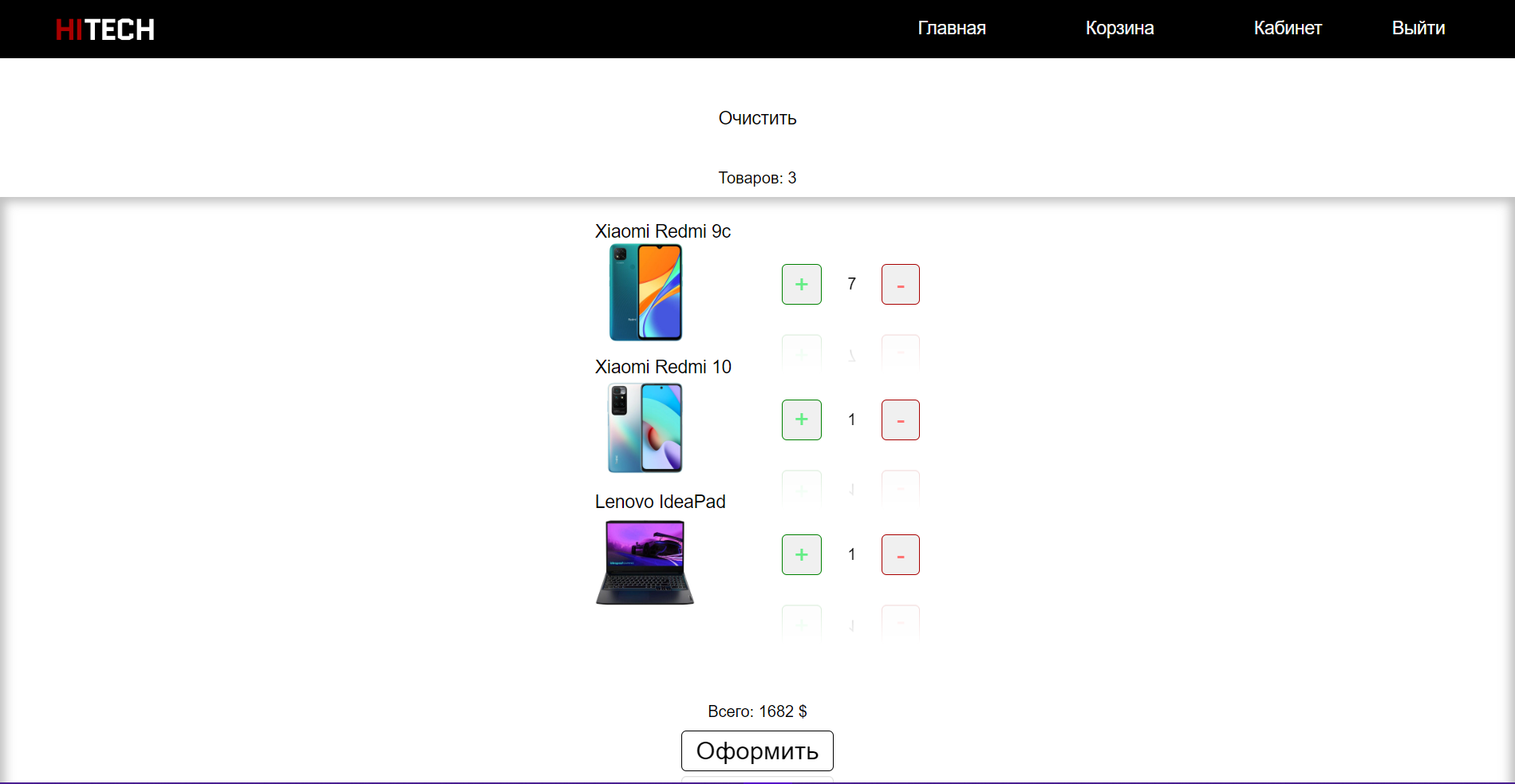


Рисунок 5.5 – Страница корзины

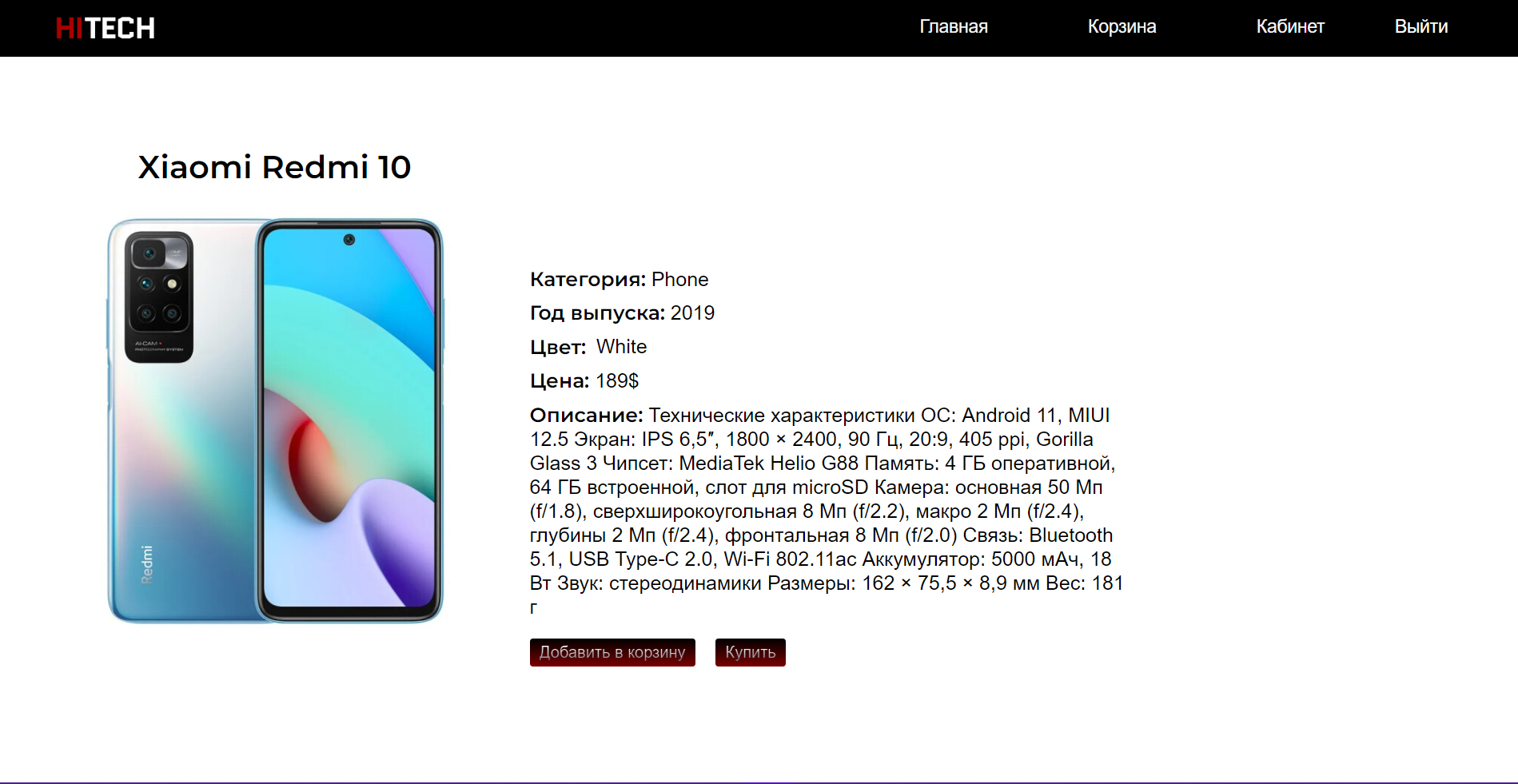


Рисунок 5.6 – Страница с информацией о товаре

5.1 Заключение по разделу

В данном разделе было предоставлено руководство для пользователя, который впервые вошел в приложение. В случае, если пользователь не знает, какие функции выполняет приложения, то он сможет воспользоваться руководством.

Заключение

В ходе курсового проекта были изучены основные возможности при использовании React-библиотеки. Благодаря сторонним библиотекам были достигнуты те цели, которые были поставлены в начале проекта. Благодаря компоненту Toastify была реализована возможность уведомлять пользователя о событиях. В ходе проекта основными использованными хуками были классы, внутри которых была возможность использовать состояния и пропсы, а так же прописывать логику функций, которые были использованы внутри render. Использование usestate() позволило выполнять функции сразу же при загрузке страницы. Были исследованы способы подключения приложения к базе данных, благодаря которой пользователь мог динамически работать с информацией.

Были реализованы следующие задачи:

* Создание адаптивного приложения
* Разработать макет приложения
* Разработать структуру приложения
* Разработка функций для взаимодействия пользователя с приложением
* Разработка взаимодействия приложения с базой данных
* Тестирование приложения

В итоге был разработано рабочее приложение, которое дало возможность выбирать и заказывать товары через интернет. Приложение специально разрабатывалось простым и понятным для пользователя, чтобы привлекать большее количество пользователей. У приложения большая перспектива, если продолжать разрабатывать и добавлять новые возможности.

Список использованных источников

1. Документация по React JS [Электронный ресурс]. - Режим доступа : <https://reactjs.org>/. – Дата обращения 05.12.2022.
2. Документация по Figma [Электронный ресурс]. - Режим доступа: <https://help.figma.com>. – Дата обращения 01.12.2022.
3. Документация по Redux/Redux-tolkit [Электронный ресурс]. - Режим доступа : <https://redux-toolkit.js.org/> . – Дата обращения 05.12.2022.
4. Документация по React-toastify [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/package/react-toastify> – Дата обращения 05.12.2022.
5. Сайт-аналог «Хайтек» [Электронный ресурс]. – Режим доступа: <https://hightech.fm/> - Дата обращения 28.11.2022
6. Сайт-аналог «5 элемент»[Электрон.ный ресурс]. – Режим доступа: <https://5element.by/> - Дата обращения 28.11.2022.

ПРИЛОЖЕНИЕ А

Листинг компонента Basket

|  |
| --- |
| import { FC, useEffect, useState } from 'react';  import { useDispatch } from 'react-redux';  import { Link, useNavigate, useParams } from 'react-router-dom';  import { useAppSelector } from '../store/store';  import { AddBasket, DeleteAllBasket, DeleteBasket, GetBasket } from './../store/CreateProduct';  import Footer from './Footer';  import Forms from './Forms';  const Basket: FC = () => {  const { id } = useParams();  const { basket, sum } = useAppSelector(state => state.product);  const { user } = useAppSelector(state => state.auth);  const [state, setState] = useState<any>([])  const [FormBool, setFormBool] = useState<boolean>(false)  const dispatch = useDispatch();  const navigate = useNavigate();  useEffect(() => {  setState(basket)  dispatch(GetBasket((id)));  }, [basket]);  useEffect(() => {  window.scrollTo(0, 0);  dispatch(GetBasket((id)));  setState(basket)  }, []);  useEffect(() => {  if (!user) {  navigate('/login');  }  }, [user]);  const DeleteBasketAll = () => dispatch(DeleteAllBasket(Number(user?.id\_person)))  const changeBool = (value: boolean) => setFormBool(value)  const handleBuyOrder = () => setFormBool(true)  const AddToBasket = (item: any) => {  const basket = {  id\_product: Number(item.id\_product),  id\_person: Number(user?.id\_person),  count: Number(item.count)  }  dispatch(AddBasket(basket))  }  const DeleteToBasket = (item: any) => {  const basket = {  id\_product: Number(item.id\_product),  id\_person: Number(user?.id\_person),  id\_basket: Number(item.id\_basket)  }  dispatch(DeleteBasket(basket))  }  return (  <div  className='animate'>  {state?.length === 0 ? (<div className="empty">  <p>Карзина пуста,<Link to="/product"> заполните ее</Link></p>  </div>) : (  <> <h1 className='title\_basket'>  </h1>  <button  className='button\_delete\_all'  onClick={DeleteBasketAll}  >Очистить</button>  <p className="count\_products">  <span> Товаров: {state?.length || 0}</span>  </p>  <div className="wrapper bas noselect">  <div className="title\_list ">  <p>Всего: {sum} $</p>  <button id='kupit' onClick={() => handleBuyOrder()}>Оформить</button>  </div>  <div className="basket">  {state?.map((item: any, index: any) => {  const { id\_product, name, img, count } = item;  return (  <div className="product" key={index}>    <div className="img">  <div className='name'>  <Link to={`/product/:${id\_product}`}>  <p>{name}</p>  </Link></div>  <Link to={`/product/:${id\_product}`}>  <div className='backet\_img' ></div> <img id='backet\_img' src={img} alt="" />  </Link>    </div>    <div className="buttons\_add\_del">  <button id='plus' className='add' onClick={() => AddToBasket(item)}>+</button>  <p className='count'><p>{count || 0}</p></p>  <button className='add' id='minus'  onClick={() => DeleteToBasket(item)}  >-</button>  </div>  </div>  )  })}  </div>    </div>  {FormBool && <Forms changeBool={changeBool} cost={sum}  products={state}  />}  </>  )}  <Footer />  </div>  )  }  export default Basket |

ПРИЛОЖЕНИЕ Б

Листинг компонента Forms

|  |
| --- |
| return (  <div className="onsova\_form">    <div className="froms">    <div className="osnova\_form">    <div className="form\_pages">  <input  type="text"  placeholder="Имя"  value={name}  className="input\_form"  onChange={(e: ChangeEvent<HTMLInputElement>) => setName(e.target.value) }  />      <input  type="phone"  placeholder="Телефон"  value={mobile}  onChange={(e: ChangeEvent<HTMLInputElement>) => {  TryMobile(e)  }}  maxLength={13} />  <input  type="text"  placeholder="Email"  value={gmail}  onChange={(e: ChangeEvent<HTMLInputElement>) => TryEmail(e)}  />      </div>    <div className="form\_pages two">  <h1>Пункт выдачи</h1>  <select>  <option value="Denis">ул. Денисовская, 8 (ТЦ «Корона-Сити»)</option>  <option value="Korzh">ул. Корженевского, 26 (ТЦ "Корона")</option>  <option value="Nezav">пр. Независимости, 154 (ТЦ "Корона")</option>  </select>  <div>  </div>  </div >  <div className="form\_pages tree">  <div className="block">  {products?.length === 1 ? <>  {products?.map((item: any) => {  const { img, name, price, count } = item;  return (  <div className="block\_buy" key={name}>  <img src={img} alt="" />  <p>{name}</p>  <p>{price}$</p>  <p>{count}</p>  </div>  )  })}</> : (  <>{products?.map((item: any) => {  const { img, name, price, count } = item;  return (  <div className="block\_buy" key={name}>  <img src={img} alt="" />  <p>{name}</p>  <p>{price}$</p>  <p>{count}</p>  </div>  )  })}</>  )}  </div>  <div >  <button id='kupit' className="buyblock" onClick={() => handleClick(products)}>Купить</button>  </div>    </div >  <div className="close" onClick={() => changeBool(false)}>  <span>Отменить оформление</span>  </div></div></div></div>);};  export default Forms; |

ПРИЛОЖЕНИЕ С

Листинг компонента Header

|  |
| --- |
| const Header: FC<IHeader> = () => {  const [state, setState] = useState<boolean>(false)  const { user } = useAppSelector(state => state.auth);  const dispatch = useAppDispatch();  const isAuth = useAppSelector(CheckIsAuth);  const [user\_id, setUser\_id] = useState<number>(user?.id\_person)  useEffect(() => { //Принимает функцию, которая содержит код.  setUser\_id(user?.id\_person)  }, [user]);  const handleExit = () => {  dispatch(LogoutUser())  toast.warning("Вы вышли из системы")  setState(false)  dispatch(Reset())  }    const PathBasket = `/basket/:${user\_id}`;  const PathProfile = `/profile/:${user\_id}`;  return (  <div className='header'>  <div className="wrapper">  <div className="burger">  <div id="nav-icon1"  className={state ? 'open' : 'after'}  onClick={() => setState(!state)}>  {state ? <p>Закрыть</p>:<p>Меню</p>}  </div>  </div>  <div className="logo">  <Link to={'/'}> <img id='logo' src={require( "./Photo/hitech.png")}/></Link>  </div>  <nav  className={state ? 'nav active' : 'nav'}  >  <ul>  <li  >  <Link to="/"  onClick={() => setState(false)}  >Главная</Link>  </li>    <li>  <Link to={PathBasket}  onClick={() => setState(false)}>Корзина</Link>  </li>  <li>  <Link to={PathProfile}  onClick={() => setState(false)}>Кабинет</Link>  </li>  <li className='li\_end'>  {isAuth  ?  <a  className='button\_b'  onClick={() => {  handleExit();  }}  >  Выйти  </a>  :  <Link  onClick={() => setState(false)}  className='button\_a'  to="/login">  Войти  </Link>}</li></ul></nav></div></div >)}  export default Header |

ПРИЛОЖЕНИЕ D

Листинг компонента Home

|  |
| --- |
| const Home: FC<IHeader> = () => {  const [state, setState] = useState<boolean>(false)  const { user } = useAppSelector(state => state.auth);  const dispatch = useAppDispatch();  const isAuth = useAppSelector(CheckIsAuth);  const [user\_id, setUser\_id] = useState<number>(user?.id\_person)  useEffect(() => { //Принимает функцию, которая содержит код.  setUser\_id(user?.id\_person)  }, [user]);  const handleExit = () => {  dispatch(LogoutUser())  toast.warning("Вы вышли из системы")  setState(false)  dispatch(Reset())}  const PathBasket = `/basket/:${user\_id}`;  const PathProfile = `/profile/:${user\_id}`;  return (  <div className='header'>  <div className="wrapper">  <div className="burger">  <div id="nav-icon1"  className={state ? 'open' : 'after'}  onClick={() => setState(!state)}>  {state ? <p>Закрыть</p>:<p>Меню</p>}  </div>  </div>  <div className="logo">  <Link to={'/'}> <img id='logo' src={require( "./Photo/hitech.png")}/></Link>  </div>  <nav  className={state ? 'nav active' : 'nav'}>  <ul>  <li>  <Link to="/"  onClick={() => setState(false)}  >Главная</Link>  </li>  <li>  <Link to={PathBasket}  onClick={() => setState(false)}>Корзина</Link>  </li>  <li>  <Link to={PathProfile}  onClick={() => setState(false)}>Кабинет</Link>  </li>  <li className='li\_end'>  {isAuth  ?  <a  className='button\_b'  onClick={() => {  handleExit();  }}  >  Выйти  </a>  :  <Link  onClick={() => setState(false)}  className='button\_a'  to="/login">  Войти  </Link> }  </li>  </ul>  </nav>  </div>  </div >)}  export default Home; |

ПРИЛОЖЕНИЕ E

Листинг компонента Login

|  |
| --- |
| const Login: FC = () => {  const { message, token, user } = useAppSelector(state => state.auth);  const [name, setName] = useState<string>('');  const [password, setPassword] = useState<string>('');  const navigate = useNavigate();  const dispatch = useAppDispatch();  useEffect(() => {  window.scrollTo(0, 0);  }  , []);  useEffect(() => {  if (message) {  if (message === "Вы вошли в систему") toast.success(message);  else {  toast.error("Вы не вошли в систему")  }  }  return () => {  dispatch(DeleteMessage());  }  }, [message]);  useEffect(() => {  if (token) {  navigate(`/`);}}, [token]);  const Login = () => {  if (name.trim() === "") {  toast.error("Имя пустое");  return;}  if (password.trim() === "") {  toast.error("Пароль пустой");  return;}  dispatch(LoginUser({ name, password }));}  const TryNameRegexp = (e: ChangeEvent<HTMLInputElement>) => { const regexp = /^[a-zA-Zа-яА-Я ]+$/;  const name = e.target.value;  if (regexp.test(name)) {  setName(name);}  if (name === "") {  setName(name);}}  const TryPasswordRegexp = (e: ChangeEvent<HTMLInputElement>) => {const regexp = /^[a-zA-Z0-9]+$/;  const password = e.target.value;  if (regexp.test(password)) {  setPassword(password);}  if (password === "") {  setPassword(password);}}  return (  <div className='animate'>  <div className="wrapper">  <div className="form">  <h1 className="title">  Вход  </h1>  <div className="input">  <input type="text" placeholder="Name"  value={name}  onChange={(e: ChangeEvent<HTMLInputElement>) => TryNameRegexp(e)}/>  <input type="password" placeholder='Password'  value={password}  onChange={(e: ChangeEvent<HTMLInputElement>) => TryPasswordRegexp(e)}  />  </div>  <div className="message">  </div>  <p className='account'>  Если нет аккаунта, можете <Link to='/register'>зарегистрироваться</Link>  </p>  <div className="buttons">  <button  onClick={Login}>  Войти  </button>  </div>  </div>  </div>  < Footer />  </div>)}  export default Login |

ПРИЛОЖЕНИЕ F

Листинг компонента User

|  |
| --- |
| const User: FC = () => {  const { orders } = useAppSelector(state => state.product);  const { id } = useParams();  const { user } = useAppSelector(state => state.auth);    const [order, setOrder] = useState<any[]>(orders);  const [show, setShow] = useState<boolean>(false);  const navigate = useNavigate();  const dispatch = useAppDispatch();  useEffect(() => {  if (!user) {  navigate('/login');  }  }, [user]);  useEffect( () => {  dispatch(GetOrders(id));  setOrder(orders);  setShow(!show);  })  return (  <div className="animate">  <div className='wrapper'>  <h1 className='title-profile'>  Профиль  </h1>  <div className="profile">  <div className="information">  <div className="info\_about\_user">  <div className="name\_info"><p>Имя:</p>  <p>Фамилия:</p><p>Количество товаров:</p></div><div className="value\_info"><p>  {user?.name || "No name"}</p><p>  {user?.surname || "No surname"}  </p><p>  {order.length}</p></div></div></div></div>  <div>  <h1 className='title\_my\_orders'>  Покупки  </h1><div className="wrapper bas">  <div className="basket">  {order?.map((item: any, index: any) => {  const { id\_product, name, img, count, price, dateorders, id\_buy, isbuy } = item;  return (<div className="product" key={index}>  <div className="id">  <p>{index + 1}</p></div><div className="img">  <Link to={`/product/:${id\_product}`}>  <img src={img} alt="" />  </Link></div>  <div className="name">  <Link to={`/product/:${id\_product}`}>  <p>{name}</p>  </Link><p>Цена: {price}</p><p>Количество: {count}</p> </div>  </div>)})}</div></div></div></div><Footer /></div >)}  export default User; |

ПРИЛОЖЕНИЕ G

Макет приложения

