МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ФАКУЛЬТЕТ АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ

КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ



**Курсовой проект**

По дисциплине: «Программное обеспечение информационных систем»

Тема: «Интернет-магазин смартфонов»

|  |  |
| --- | --- |
| Факультет: АВТФ | Преподаватель: Менжулин С. А. |
| Группа: АВТ-018 |  |
| Студент: Наумова Д.К. |  |

Новосибирск

2023

**РЕФЕРАТ**

В условиях быстро меняющегося рынка смартфонов и усиливающейся конкуренции создание информационной системы для интернет-магазина становится крайне актуальным. Современные требования рынка предполагают эффективное использование технологий для улучшения управления бизнес-процессами, повышения качества обслуживания клиентов и оперативного принятия управленческих решений. В этом контексте разработка индивидуализированной информационной системы становится ключевым инструментом для достижения стратегических целей интернет-магазинов.

Актуальность создания информационной системы заключается в том, что современные требования рынка не только касаются качественных продуктов, но и включают в себя применение передовых технологий для оптимизации бизнес-процессов, улучшения обслуживания клиентов и оперативного принятия управленческих решений.

Цель данного проекта заключается в создании информационной системы, способной оптимизировать операционные процессы интернет-магазина, увеличивать производительность персонала и улучшать обслуживание клиентов.

Для достижения этой цели в рамках проекта предусмотрены следующие задачи:

* Анализ предметной области
* Определение функциональности веб-приложения для магазина смартфонов.
* Проектирование информационной системы.
* Реализация системы.
* Тестирование системы.

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Разработать информационную систему на языке C# в соответствии с индивидуальным вариантом задания. Жизненный цикл всех информационных объектов в работе должны быть управляем из интерфейса пользователя или администратора информационной системы.

Оглавление

[РЕФЕРАТ 2](#_Toc154400668)

[ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ 3](#_Toc154400669)

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ 5](#_Toc154400670)

[ВВЕДЕНИЕ 6](#_Toc154400671)

[1 ПРЕДМЕТНАЯ ОБЛАСТЬ 7](#_Toc154400672)

[1.1 Анализ предметной области 7](#_Toc154400673)

[1.2 Требования к проекту 8](#_Toc154400674)

[2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 10](#_Toc154400675)

[2.1 Диаграмма классов предметной области 10](#_Toc154400676)

[2.2 Информационные объекты системы 11](#_Toc154400677)

[2.3 Диаграмма состояний 17](#_Toc154400678)

[2.4 Диаграмма прецедентов 18](#_Toc154400679)

[2.5 Диаграмма компонент 20](#_Toc154400680)

[3 ПРАКТИЧЕСКАЯ ЧАСТЬ 20](#_Toc154400681)

[3.1 Структура информационной системы 20](#_Toc154400682)

[3.2 Реализация системы 23](#_Toc154400683)

[4 ТЕСТИРОВАНИЕ 27](#_Toc154400684)

[ЗАКЛЮЧЕНИЕ 33](#_Toc154400685)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc154400686)

[ПРИЛОЖЕНИЯ 36](#_Toc154400687)

[ПРИЛОЖЕНИЕ А 36](#_Toc154400688)

[ПРИЛОЖЕНИЕ Б 37](#_Toc154400689)

[ПРИЛОЖЕНИЕ В 47](#_Toc154400690)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

1. ASP.NET Core MVC - Advanced Server Pages .NET Core Model-View-Controller, фреймворк для создания веб-приложений.
2. HTML - HyperText Markup Language, стандартный язык разметки для создания веб-страниц.
3. CSS - Cascading Style Sheets, язык таблиц стилей, используемый для описания внешнего вида документа, написанного на языке разметки.
4. JS - JavaScript, программный язык, используемый для создания интерактивных эффектов внутри веб-браузеров.
5. SQL - Structured Query Language, язык программирования для управления данными в системах управления базами данных.
6. API - Application Programming Interface, интерфейс программирования приложений, набор определений и протоколов для создания и интеграции программного обеспечения.
7. ORM - Object-Relational Mapping, техника программирования для преобразования данных между несовместимыми типами систем.
8. EF - Entity Framework, фреймворк ORM для .NET.

**ВВЕДЕНИЕ**

Развитие современных технологий оказывает значительное воздействие на различные сферы человеческой жизни, приводя к переходу к цифровому формату. Этот переход обусловлен постоянным увеличением объема обрабатываемой и хранимой информации. С ростом масштабов систем, таких как библиотеки, больницы, интернет-магазины, где тысячи записей постоянно обновляются, эффективное управление данными без использования электронных устройств становится невозможным.

На малых по размерам системах, таких как классные журналы и телефонные справочники, физические носители информации могут быть достаточно эффективными. Однако, в контексте разрастающихся организаций, их потребностей в обработке и хранении данных, становится явным, что для оптимизации управления системой требуется более удобное и эффективное решение — хранение данных в базах данных.

Внедрение баз данных автоматизирует процессы хранения и обработки данных, обеспечивая их безопасность. Аспекты безопасности, такие как целостность, доступность и конфиденциальность данных, становятся приоритетными. В сравнении с физическими носителями, базы данных обеспечивают более надежную защиту: они имеют резервные копии для восстановления данных, позволяют точно определить значения полей таблиц, и ограничивают доступ через уникальные логины и пароли пользователей.

Кроме того, базы данных оптимизируют поиск информации и предоставляют персонализированные справки в соответствии с требованиями пользователя. Благодаря автоматизации и оптимизации работы с данными, системы баз данных стали неотъемлемой частью многих бизнес-процессов, в том числе в коммерческих и некоммерческих организациях.

**1. ПРЕДМЕТНАЯ ОБЛАСТЬ**

**1.1 Анализ предметной области**

Система управления интернет-магазином смартфонов представляет программное решение, направленное на оптимизацию процессов продажи мобильных устройств через веб-платформу. Основными участниками данной системы являются клиенты, стремящиеся приобрести современные смартфоны, и администраторы магазина, ответственные за эффективное управление всеми аспектами онлайн-торговли.

Система должна предоставлять возможность клиентам просматривать ассортимент смартфонов, выбор бренда, осуществлять сортировку по цене, и производить оформление заказа. Для администраторов предусмотрены инструменты управления каталогом товаров.

Для эффективной работы администратора с системой необходимо создать модули управления каталогом смартфонов, обеспечивающие оперативное добавление новых моделей и обновление информации о товарах, также удаление уже не актуальных товаров.

Внедрение данной системы позволит улучшить пользовательский опыт и увеличить эффективность управления интернет-магазином. В условиях динамичного рынка мобильных технологий успешная автоматизация процессов продажи смартфонов является стратегическим инструментом для привлечения и удержания клиентов, а также обеспечения конкурентоспособности в сфере электронной коммерции.

## 1.2 Требования к проекту

Информационная система "Интернет-магазин смартфонов" предназначена для эффективного управления процессами онлайн-торговли мобильными устройствами, с целью повышения эффективности продаж.

Система должна содержать следующую информацию:

* Каталог смартфонов, содержащий информацию о доступных моделях, их характеристиках и ценах.
* История заказов клиента.

Система должна предоставлять следующие возможности:

* Предоставление информации о товаре.
* Осуществление поиска товаров по брендам.
* Регистрация и авторизация;
* Оформление заказа товара;
* Сортировка товаров по цене.
* Система должна сохранять историю заказов клиентов.
* Административные функции: администраторы имеют доступ к управлению каталогом.

Требования к графическому интерфейсу:

* Единый стиль: все страницы веб-приложения оформлены в едином стиле для создания единообразного визуального восприятия.
* Интуитивная навигация: система предоставляет простой и интуитивно понятный пользовательский интерфейс.
* Многостраничное веб-приложение: система должна быть реализована в виде многостраничного веб-приложения для удобства взаимодействия пользователя с различными функциональными модулями.

Требования к базе данных:

При проектировании реляционной базы данных необходимо помнить об эффективности структуры БД.

Эффективная БД должна:

* обеспечивать быстрый доступ к данным в базе;
* минимизировать избыточность данных;
* обеспечивать целостность данных в базе. Изменение одних объектов должно автоматически изменять связанные с ними другие объекты.

Таблицы эффективной БД должны находиться в третьей нормальной форме.

**2. ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ**

В качестве средств проектирования жизненного цикла (ЖЦ) информационной системы «Интернет-магазин смартфонов» были выбраны следующие инструменты:

* Диаграмма классов предметной области «Интернет-магазин смартфонов» - общая структура иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними;
* Диаграмма состояний для сущностей системы - диаграмма состояний из теории автоматов со стандартизированными условными обозначениями, определяет состояние системы в динамике (в разные моменты времени);
* Диаграмма прецедентов - диаграмма, отражающая отношения между акторами (действующими лицами, пользователями системы) и прецедентами (функции, которые они могут выполнять);
* Диаграмма компонент - показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами.

**2.1 Диаграмма классов предметной области**

Диаграмма классов представлена на рисунке 1. Каждый класс будет иметь свойства, соответствующие полям в таблицах базы данных. Взаимосвязи между классами будут отражать внешние ключи и связи между таблицами. Класс «Пользователь» имеет доступ к работе со всеми классами системы.

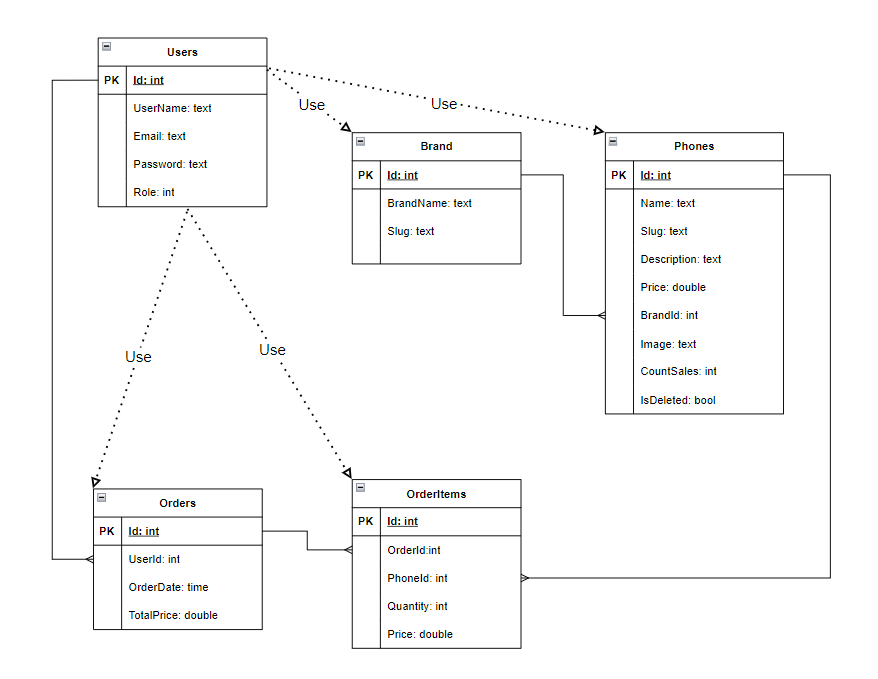


Рис. 1 – Диаграмма классов предметной области

## 2.2 Информационные объекты системы

К информационным объектам системы относятся таблицы, хранимые в базе данных. База данных содержит в себе следующие таблицы:

* таблица «Phone» - список смартфонов;
* таблица «Brand» - список брендов смартфонов;
* таблица «OrderItems» - список заказанных товаров;
* таблица «Orders» - список заказов всех пользователей;
* таблица «Users» - список пользователей

Структура таблиц «Phone», «Brands», «OrderItems», «Orders» и «Users» представлена в таблицах 1, 2, 3, 4, 5 и 6 соответственно.

Таблица 1. Структура таблицы «Phones».

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип поля** | **Допустимость NULL** |
| Id | integer | not null |
| Name | text | not null |
| Slug | text |  |
| Description | text | not null |
| Price | double precision | not null |
| BrandId | integer | not null |
| Image | text |  |
| CountSales | integer | not null |
| IsDeleted | boolean | not null |
| **Индексы:** | | |
| "PK\_Phones" PRIMARY KEY, btree ("Id") | | |
| "IX\_ Phones \_ BrandId " btree ("BrandId ") | | |
| **Ограничения внешнего ключа:** | | |
| "FK\_ Phones\_Brands\_BrandId" FOREIGN KEY ("BrandId") REFERENCES "Brands"("Id") ON DELETE CASCADE | | |
| **Ссылки извне:** | | |
| TABLE ""OrderItem"" CONSTRAINT "FK\_OrderItem\_ Phones\_ PhoneId" FOREIGN KEY ("PhoneId") REFERENCES "Phones"("Id") ON DELETE CASCADE | | |

Таблица 2. Структура таблицы «Brands».

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип поля** | **Допустимость NULL** |
| Id | integer | not null |
| BrandName | text |  |
| Slug | text |  |
| **Индексы:** | | |
| "PK\_ Brands" PRIMARY KEY, btree ("Id") | | |
| **Ссылки извне:** | | |
| TABLE ""Phones"" CONSTRAINT "FK\_ Phones \_Brands\_ BrandId" FOREIGN KEY ("BrandId") REFERENCES " Brands"("Id") ON DELETE CASCADE | | |

Таблица 3. Структура таблицы «OrderItems».

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип поля** | **Допустимость NULL** |
| Id | integer | not null |
| OrderId | integer | not null |
| PhoneId | integer | not null |
| Quantity | integer | not null |
| Price | double precision | not null |
| **Индексы:** | | |
| "PK\_OrderItem" PRIMARY KEY, btree ("Id")  "IX\_OrderItem\_PhoneId" btree ("PhoneId")  "IX\_OrderItem\_OrderId" btree ("OrderId") | | |
| **Ограничения внешнего ключа:** | | |
| "FK\_OrderItem\_Phones\_PhoneId " FOREIGN KEY ("PhoneId ") REFERENCES "Phones"("Id") ON DELETE CASCADE  "FK\_OrderItem\_Orders\_OrderId" FOREIGN KEY ("OrderId") REFERENCES "Orders"("Id") ON DELETE CASCADE | | |

Таблица 4. Структура таблицы «Orders».

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип поля** | **Допустимость NULL** |
| Id | integer | not null |
| UserId | integer | not null |
| OrderDate | timestamp with time zone | not null |
| TotalPrice | double precision | not null |
| **Индексы:** | | |
| "PK\_Orders" PRIMARY KEY, btree ("Id")  "IX\_Orders\_UserId" btree ("UserId") | | |
| **Ограничения внешнего ключа:** | | |
| "FK\_Orders\_Users\_UserId" FOREIGN KEY ("UserId") REFERENCES "Users"("Id") ON DELETE CASCADE | | |
| **Ссылки извне:** | | |
| TABLE ""OrderItems"" CONSTRAINT "FK\_OrderItem\_Orders\_OrderId" FOREIGN KEY ("OrderId") REFERENCES "Orders"("Id") ON DELETE CASCADE | | |

Таблица 5. Структура таблицы «Users».

|  |  |  |
| --- | --- | --- |
| **Название поля** | **Тип поля** | **Допустимость NULL** |
| Id | integer | not null |
| UserName | text | not null |
| Email | text | not null |
| Password | text | not null |
| Role | integer | not null |
| **Индексы:** | | |
| "PK\_Users" PRIMARY KEY, btree ("Id") | | |
| **Ссылки извне:** | | |
| TABLE ""Orders"" CONSTRAINT "FK\_Orders\_Users\_UserId" FOREIGN KEY ("UserId") REFERENCES "Users"("Id") ON DELETE CASCADE | | |

Связи между таблицами изображены на рисунке 2.

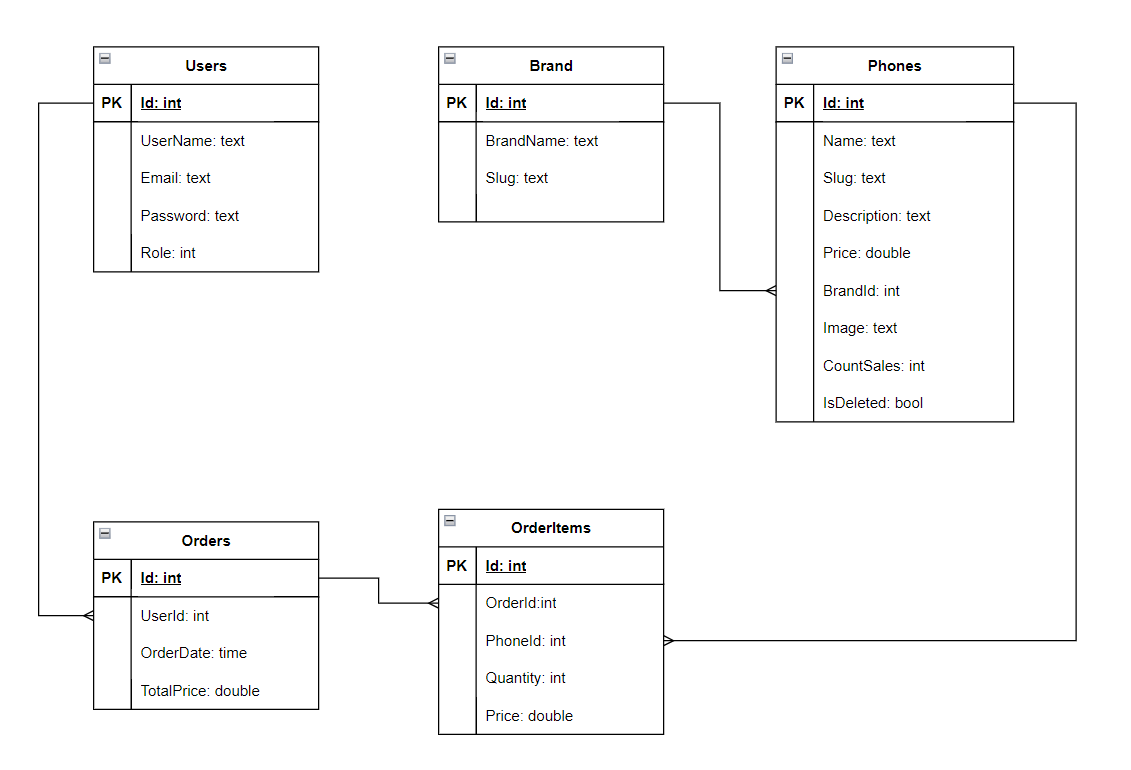


Рис. 2 - Связи между таблицами базы данных.

**2.3 Диаграмма состояний**

Диаграмма состояний изображена на рисунке 3. После перехода клиента на функциональную страницу веб-сайта начинает формироваться SQL-запрос к БД. Когда SQL-запрос сформирован он передается серверу для выполнения. Когда запрос выполнен, то он уничтожается, и клиенту выводится результат сформированного запроса в виде списка доступных смартфонов. Сервер переходит в состояние ожидания запроса.

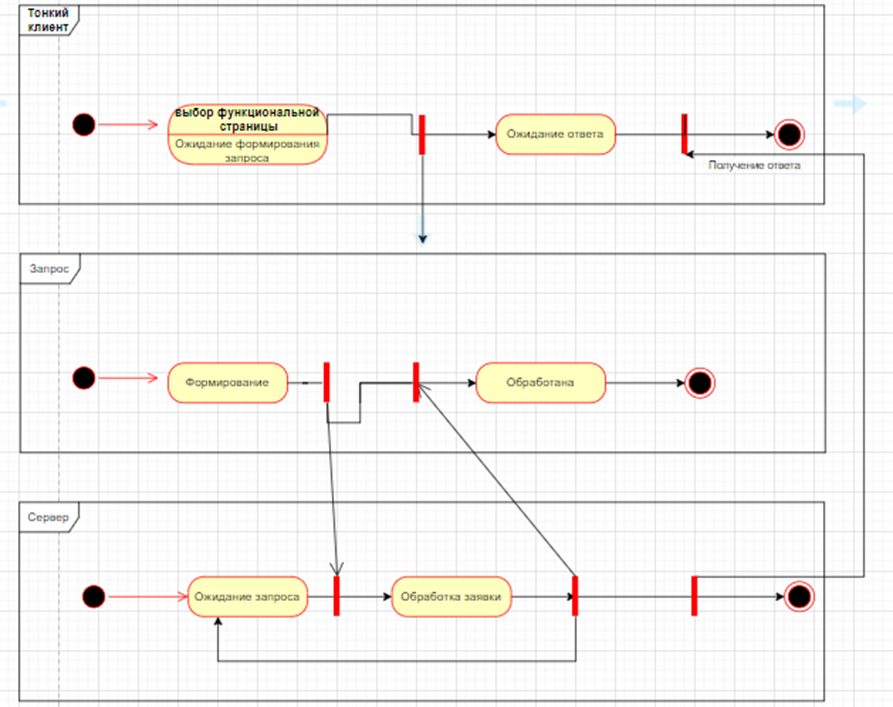


Рис. 3 – Диаграмма состояний

**2.4 Диаграмма прецедентов**

Диаграмма прецедентов отображена на рисунке 4. Данная диаграмма содержит 4 актора: «Пользователь», «Администратор», «Сервер», «База данных». «Пользователь» взаимодействует с «Сервером» и ему доступны функции выбора страницы веб-сайта, просмотра товаров (смартфонов), выбор бренда для вывода смартфонов определенного бренда, выбор метода сортировки смартфонов по цене, регистрации/авторизации, просмотра истории заказов, оформление заказа, получения ответа от «Сервера». «Администратору» предоставляется возможность добавлять, изменять, удалять товары из каталога, также ему доступны все функции обычного пользователя, кроме оформление заказа. «Сервер» формирует запрос и отправляет его «Базе данных». Взаимодействие с данными и выполнение запросов осуществляется на стороне «Базы данных». Таким образом результат запроса будет сформирован на стороне «Базы данных» и отправлен «Серверу», который в свою очередь осуществляет графическое отображение результата запроса «Пользователю».

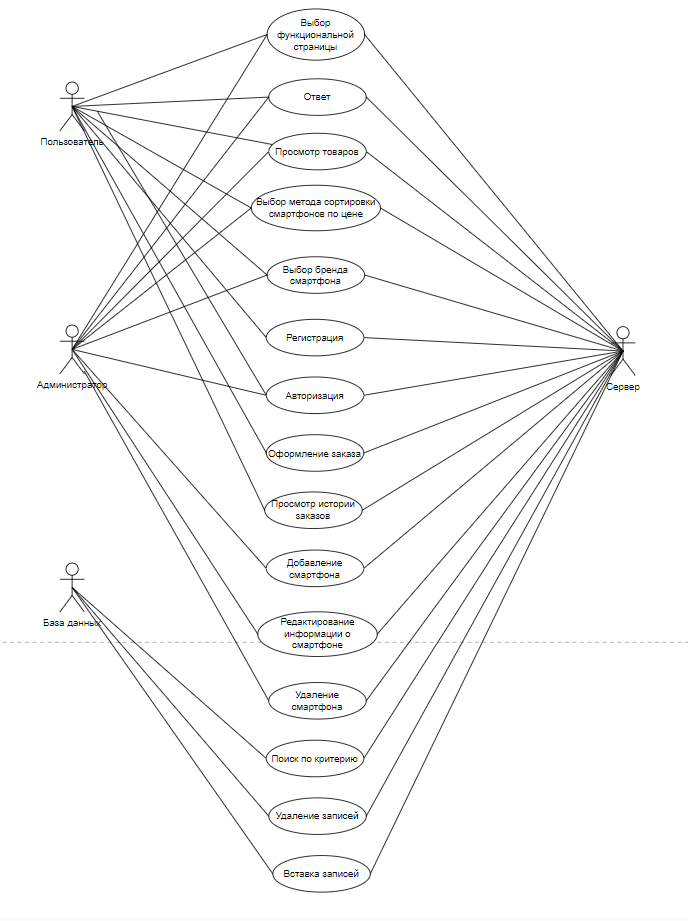


Рис. 4 – Диаграмма прецедентов

**2.5 Диаграмма компонент**

На рисунке 5 изображена диаграмма компонент реализации системы. Пользовательский интерфейс представлен в виде многостраничного сайта с разделением страниц по функционалу. В качестве языка реализации используется C#. Для пользовательского графического интерфейса используются каскадные таблицы стилей css. В качестве СУБД выбран PostgreSQL, так как он бесплатен, осуществляет функции хранения данных в виде реляционных БД и функции манипулирования ими посредством запросов SQL.

Реализация системы «Интернет-магазин смартфонов» включает в себя:

Frontend – Пользовательский интерфейс представлен в виде тонкого клиента (веб-сайт).

Backend – Серверная часть и сервер БД.

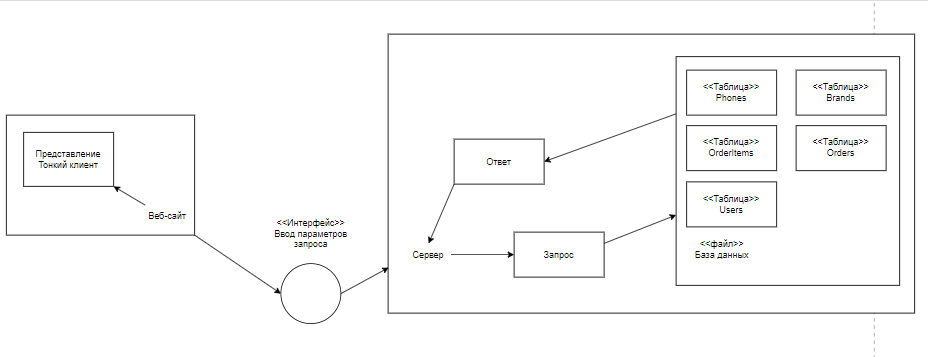


Рис. 5 – Диаграмма компонент реализации системы

**3. ПРАКТИЧЕСКАЯ ЧАСТЬ**

**3.1 Структура информационной системы**

Для хранения данных системы была создана база данных «PhoneStore», содержащая следующие таблицы: «Brands», «CartItems», «OrderItems», «Orders», «Phones» и «Users». Подробная структура таблиц описана в разделе 2.2.

Заполним таблицы и выведем их содержимое (рисунки 6-10).

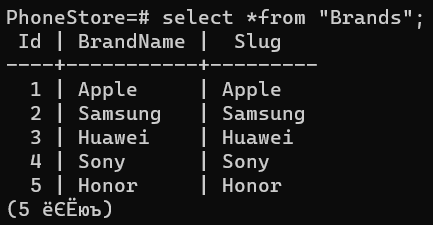


Рис. 6 – Фрагмент содержимого таблицы «Brands»

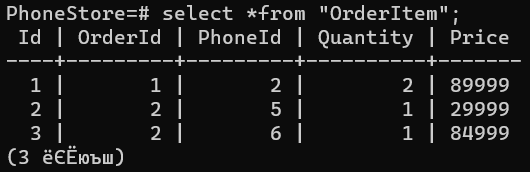


Рис. 7 – Содержимое таблицы «OrdersItem»

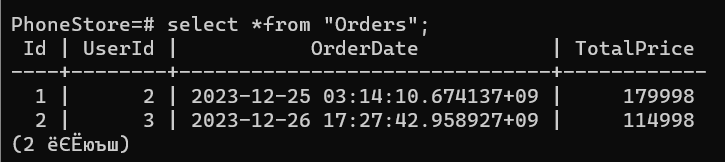


Рис. 8 – Содержимое таблицы «Orders»

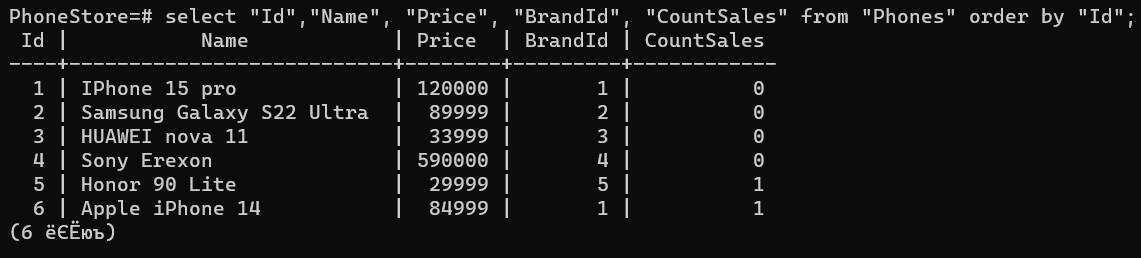


Рис. 9 – Содержимое таблицы «Phones»

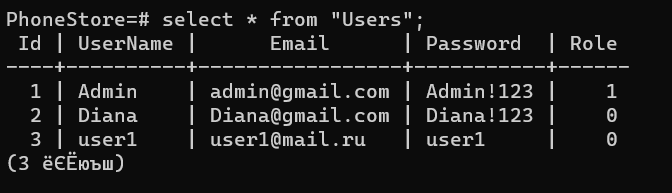


Рис. 10 – Содержимое таблицы «Users»

**3.2 Реализация системы**

Разработка системы осуществляется в среде разработки Visual Studio 2022. В качестве шаблона для реализации многостраничного веб-сайта использован «Веб-приложение ASP.NET Core (модель-представление-контроллер)». Данный проект реализован на платформе .NET версии 6.0.

Для обеспечения единого стиля страниц веб-сайта используется мастер-страница \_Layout.cshtml, на которой располагаются элементы, которые должны быть размещены на каждой странице. Такой подход позволяет создать шаблон, который упрощает внесение изменений в интерфейс веб-сайта: нет необходимости менять каждую страницу по отдельности, достаточно изменить только один файл мастер-страницы и изменения автоматически применятся ко всем связанным с этим шаблоном страницам содержания.

Для реализации графического интерфейса были также применены каскадные таблицы стилей, позволяющие установить внешний вид элементов и библиотека bootstrap, позволяющая задавать элементам в разметке классы, описывающие их стилизацию.

При разработке мастер-страницы было добавлено меню с элементами, реализующими навигацию по веб-сайту. Макет меню представлен на рисунке 11.

В ASP.NET Core приложении входной точкой служит файл Program.cs, откуда начинается выполнение программы. Он содержит все необходимые настройки для работы веб-сервера, обработки HTTP-запросов, работой с базой данных. Полный код файла Program.cs приведен в приложении А.

Разработка приложения была построена на модели MVC (Model-View-Controller) — архитектурном паттерне, который разделяет приложение на три основных компонента для упрощения управления и разработки. "Model" (модель) отвечает за бизнес-логику и данные, "View" (представление) за отображение данных пользователю, а "Controller" (контроллер) служит связующим звеном между моделью и представлением, обрабатывая пользовательские запросы, взаимодействуя с моделью и выбирая представление для отображения ответа. Полный код всех контроллеров приведен в приложении Б.

В разработке был задействован Entity Framework — это ORM (Object-Relational Mapper), который позволяет разработчикам работать с базой данных, используя объектно-ориентированные модели. Это означает, что вместо написания SQL-запросов, можно использовать объекты C#, которые отражают структуру базы данных, делая код более интуитивно понятным и уменьшая вероятность ошибок. ORM автоматически генерирует SQL-запросы, что позволяет исключить написание отдельных хранимых пользовательских функций, что положительно сказывается на масштабируемости приложения.

В приложении также была настроена авторизация по ролям. Администратор имеет доступ к админ-функциональности, в то время как обычный пользователь – нет. Для возможности заказа товара необходимо быть авторизованным.

Ниже на рисунках 11-13 представлены макеты страниц сайта.



Рис. 11 – Содержимое меню мастер-страницы.

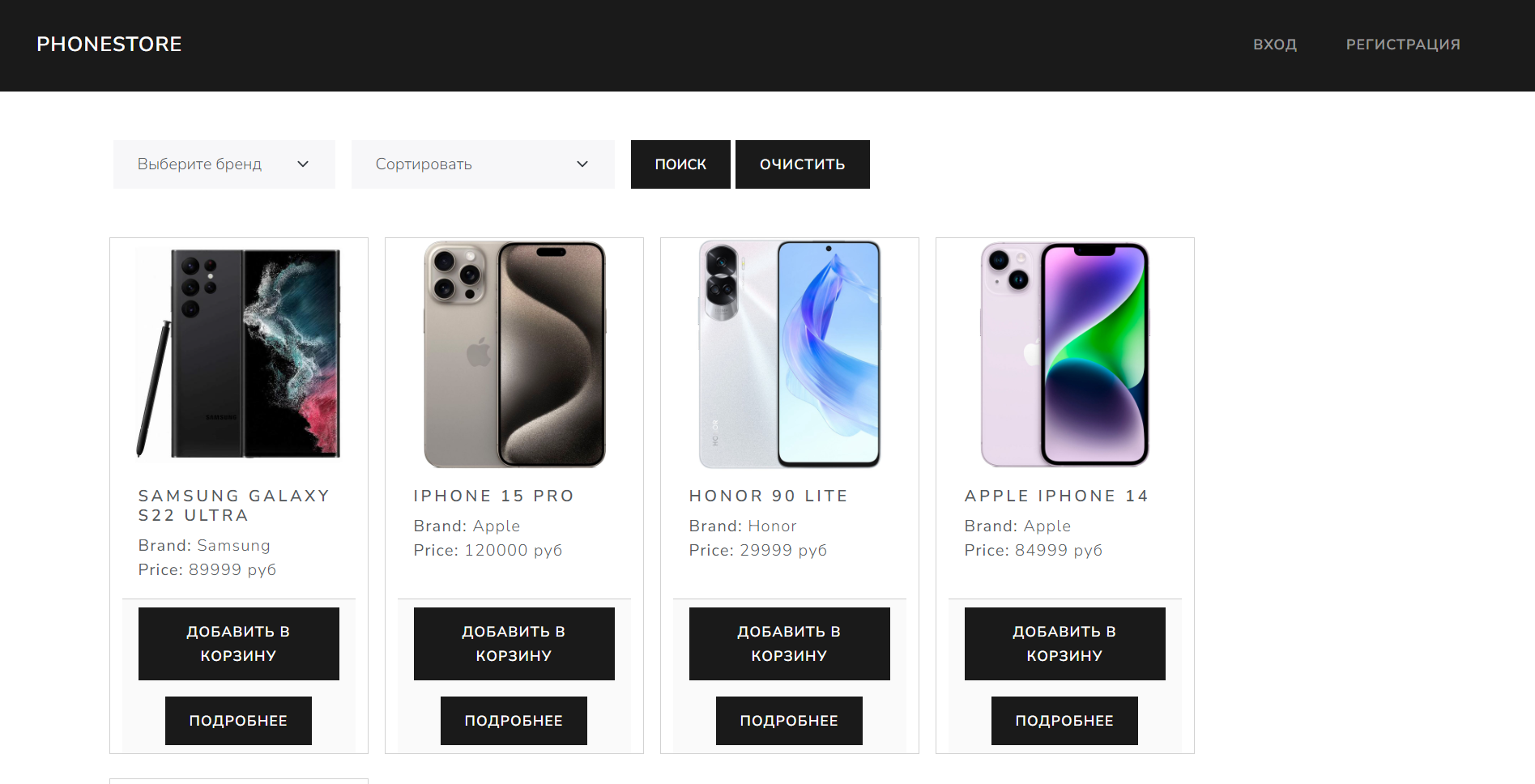


Рис. 12 – Содержимое главной страницы.

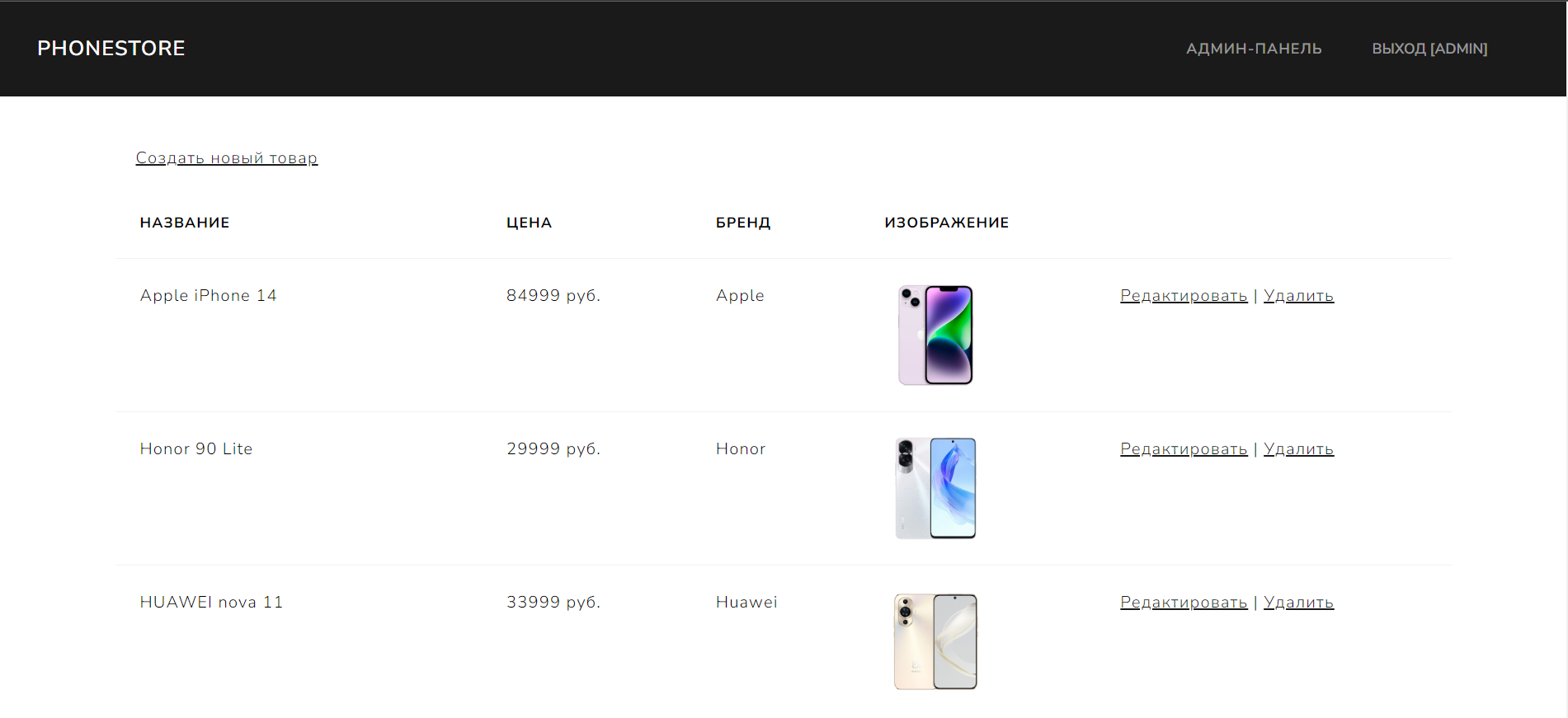


Рис. 13 – Содержимое страницы админ-панели.

**4. ТЕСТИРОВАНИЕ**

Запустим проект и протестируем систему «Интернет-магазин смартфонов» (рисунок 14).

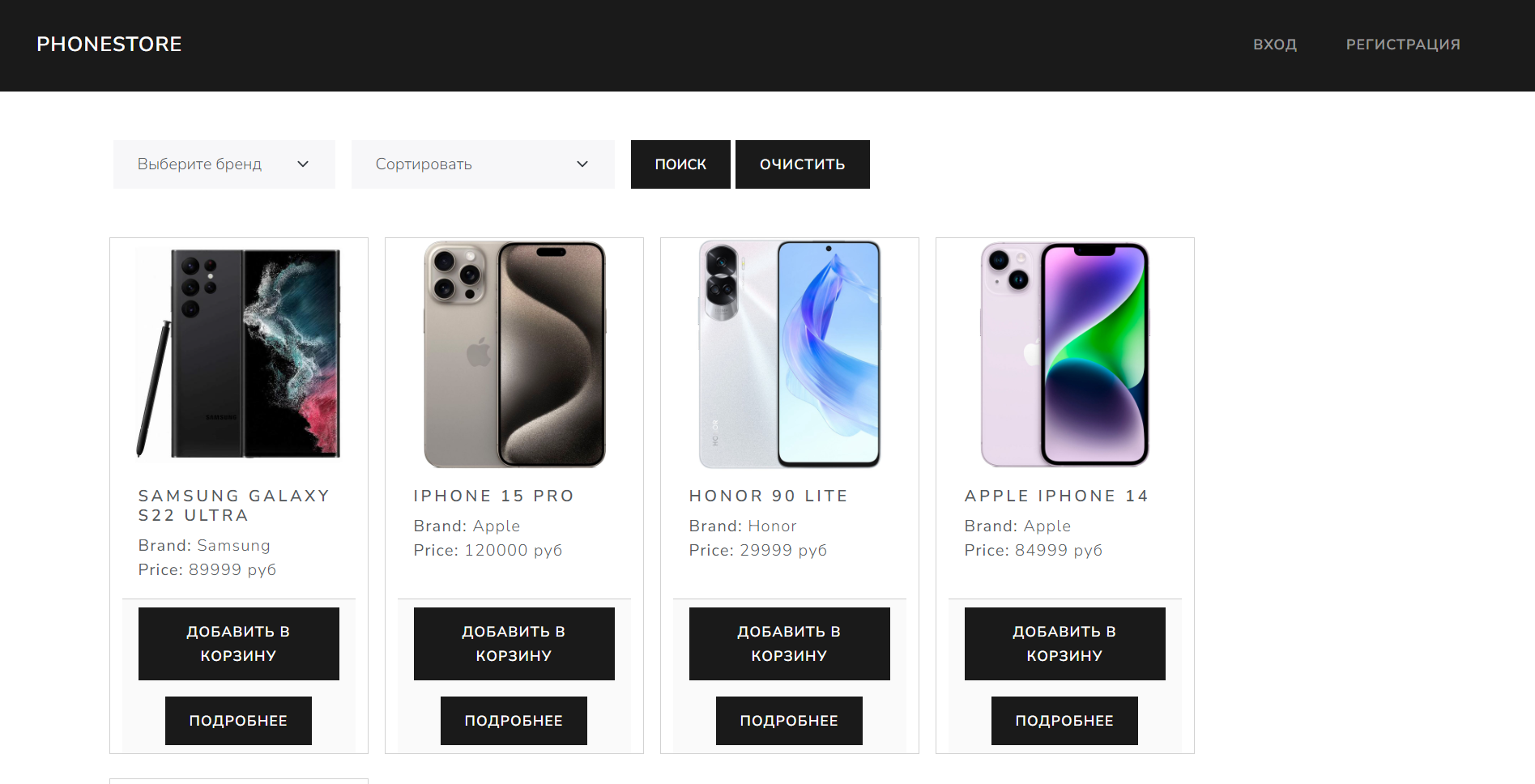


Рис. 14 – Главная страница.

Перейдем по кнопке «Регистрация» и введем данные для регистрации (рисунок 15).

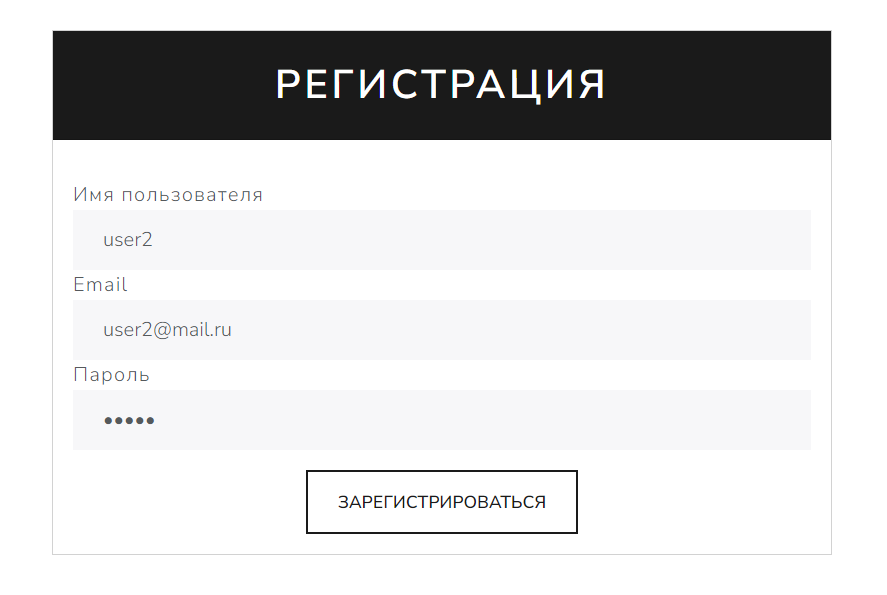


Рис. 15 – Страница регистрации.

После регистрации перейдем по кнопке «Вход» и введем данные для входа (рисунок 16).

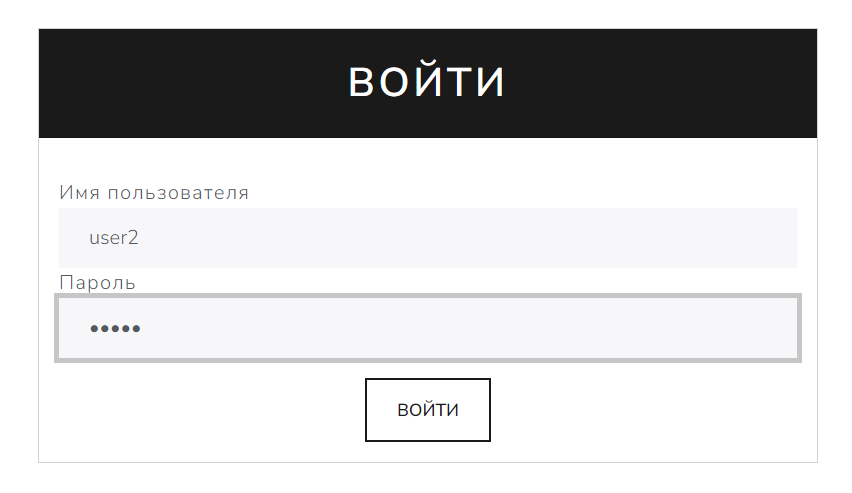


Рис. 16 – Страница авторизации.

После успешного входа можем заметить, что в меню появились кнопки с выходом, просмотра корзины и просмотра предыдущих заказов (рисунок 17).



Рис. 17 – Меню сайта.

Теперь выберем смартфон и добавим один из них в корзину. После добавления товара сводка о имеющихся товаров в корзине увеличивается (рисунки 18 – 19).



Рис.18 – Краткая сводка о имеющихся товарах в корзине.

Перейдем в корзину и на ее странице получим возможность увеличить/уменьшить количество товаров одной модели. После можно оформить заказ.

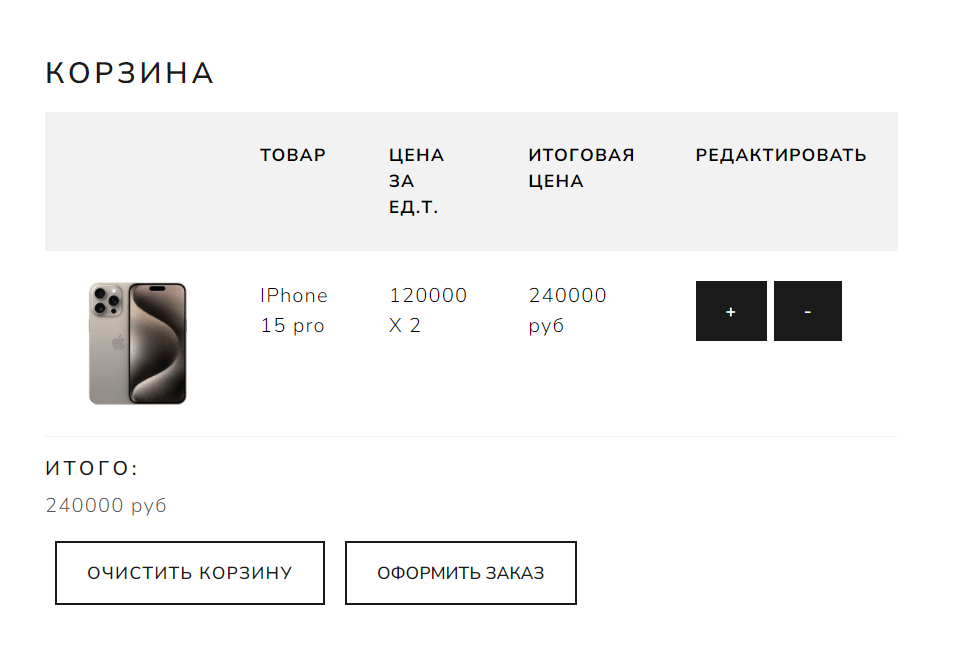


Рис. 19 – Страница корзины.

После оформления заказа можно будет просмотреть заказ на странице истории заказов пользователя, перейдя по кнопке «История заказов» (рисунок 20).

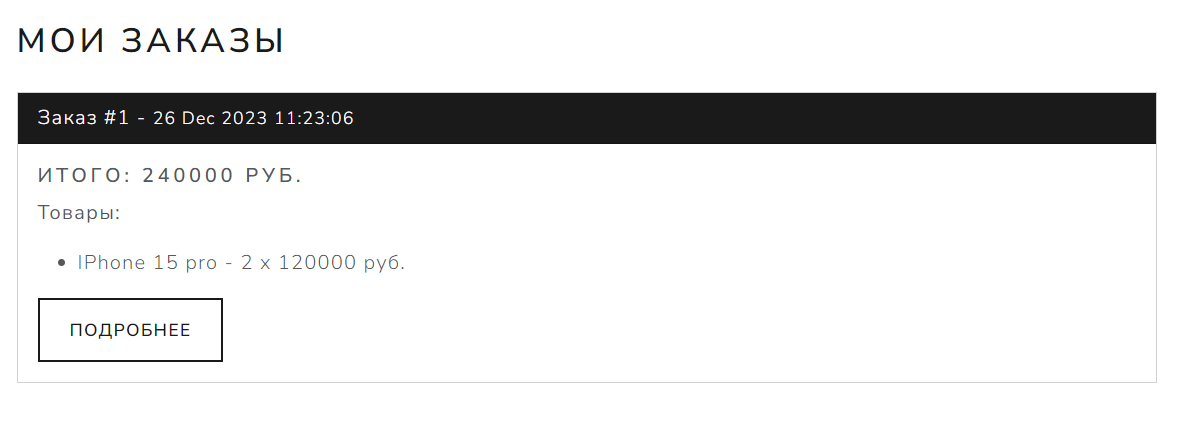


Рис. 20 – Страница с заказами пользователя.

Перейдя по кнопке «Подробнее» можно детальнее ознакомиться с заказом (рисунок 21).

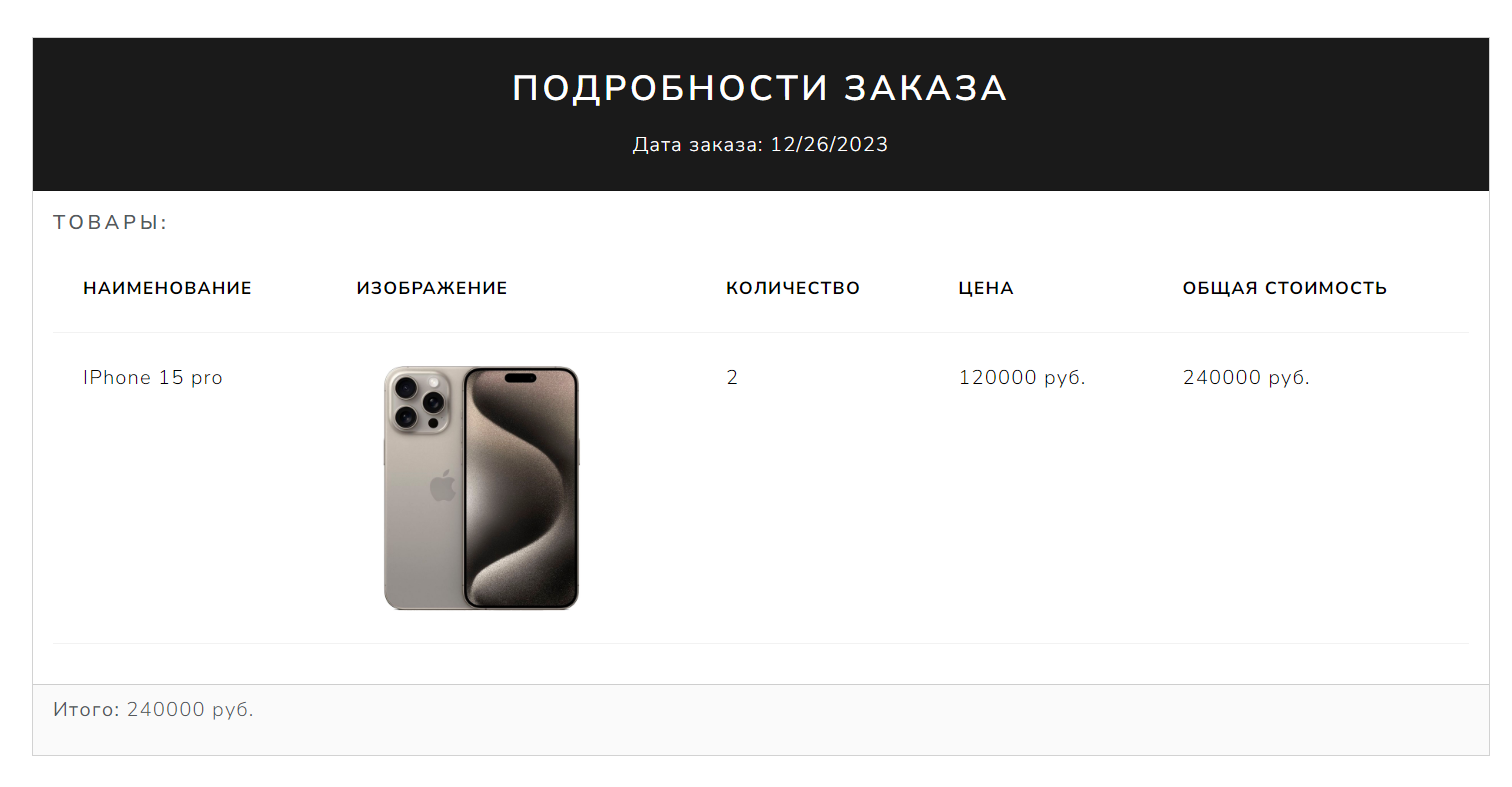


Рис. 21 – Страница с деталями заказа.

Вернемся на главную странницу и перейдем по кнопке «Подробнее» одного из смартфонов. (рисунок 22)

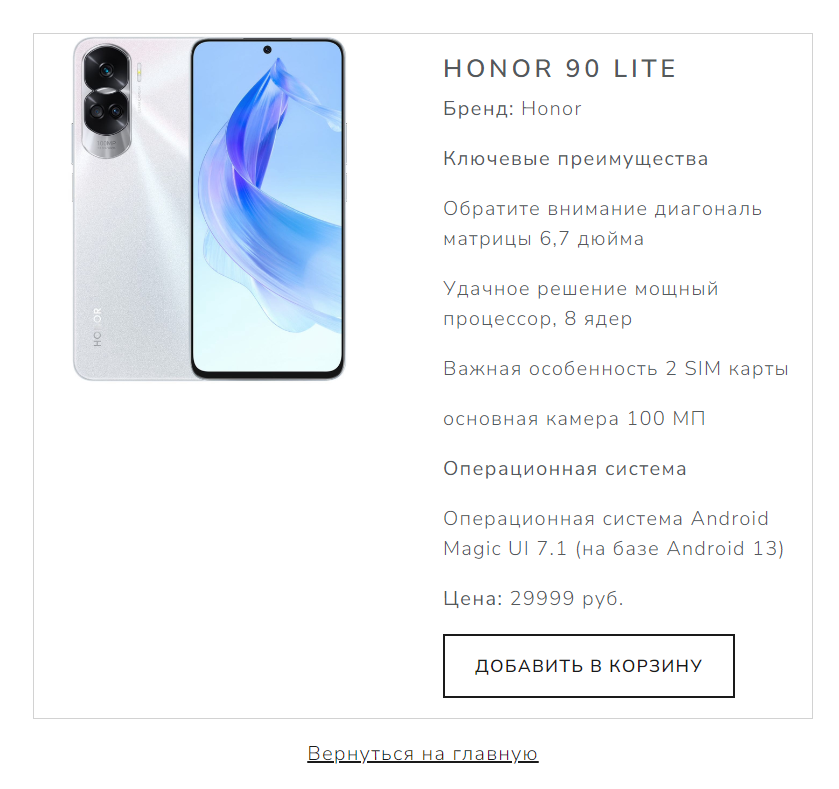


Рис. 22 – Карта товара.

Протестируем фильтрацию и сортировку. Поменяем значение в выпадающем списке «Выберите бренд» на «Apple» и увидим, что на странице будут представлены только товары бренда «Apple». (рисунок 23)

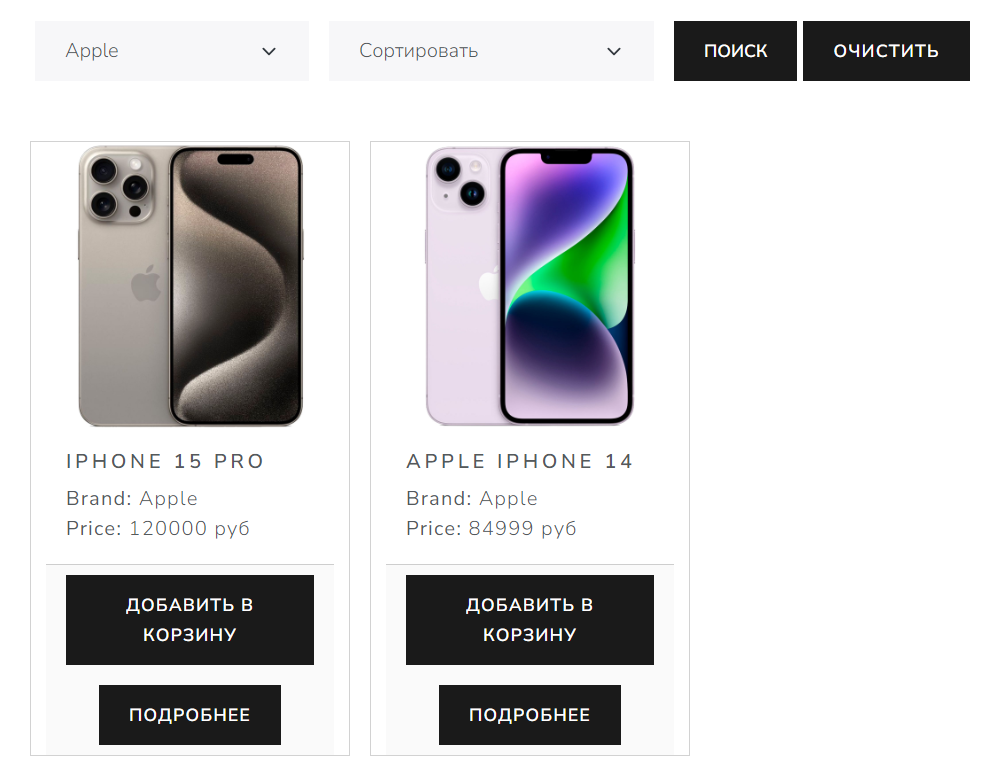


Рис. 23 – Результат фильтрации по брендам.

Теперь поменяем значение в выпадающем списке «Сортировать» и увидим, что сортировка работает. (рисунок 24-25)

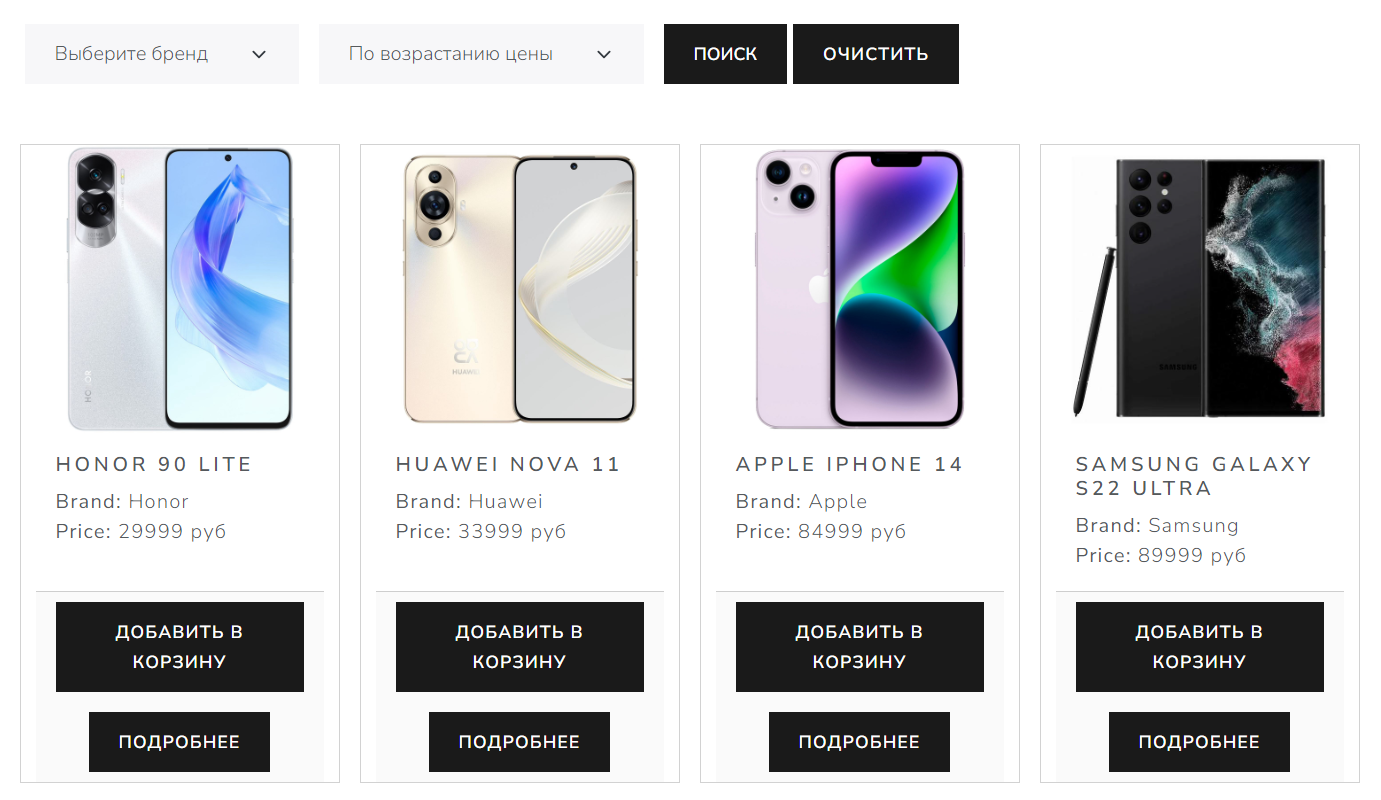


Рис. 24 – Результат работы сортировки по возрастанию цены.

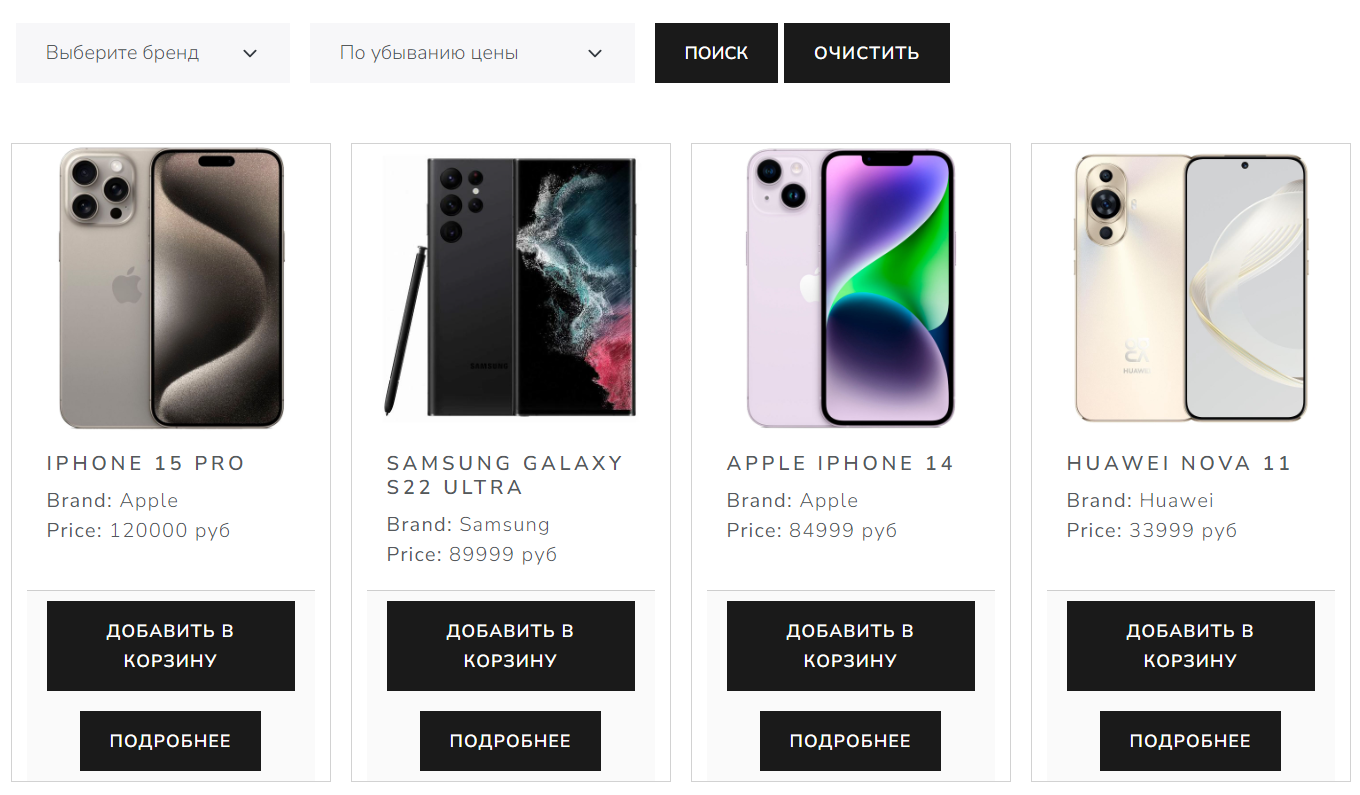


Рис. 25 – Результат работы сортировки по убыванию цены.

Поменяем значение в выпадающем списке «Выберите бренд» и добавим сортировку по цене. (рисунок 26)

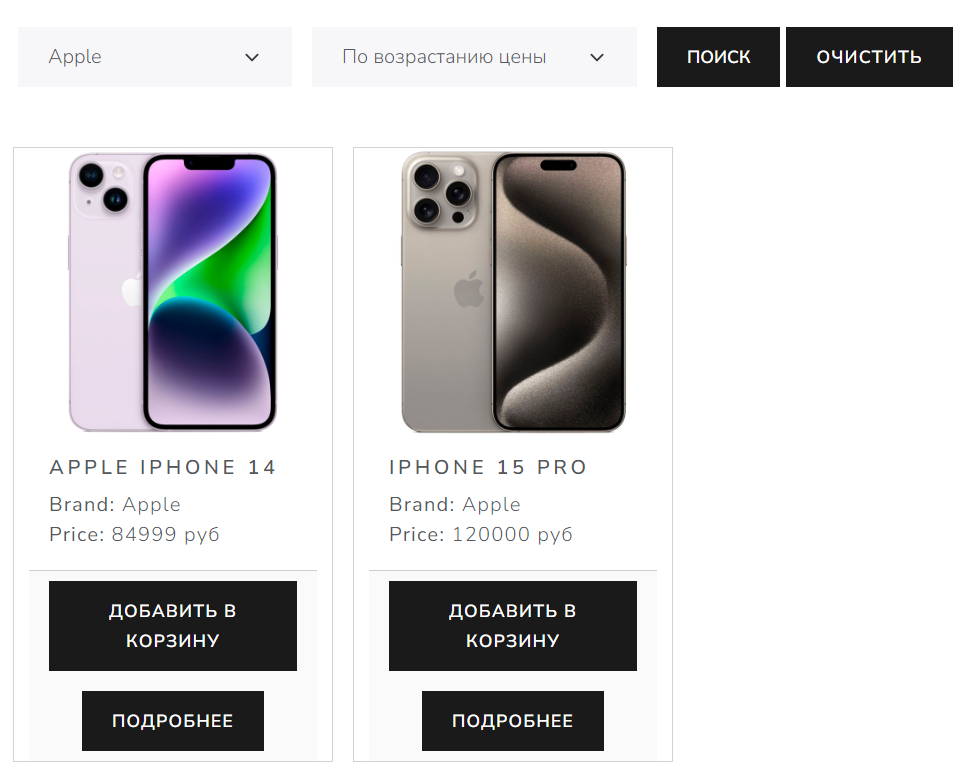


Рис. 26 – Результат совместной работы сортировки и фильтрации.

Теперь войдем на сайт в качестве администратора. В меню увидим появившуюся кнопку «Админ-панель» (рисунок 27).



Рис. 27 – Меню сайта.

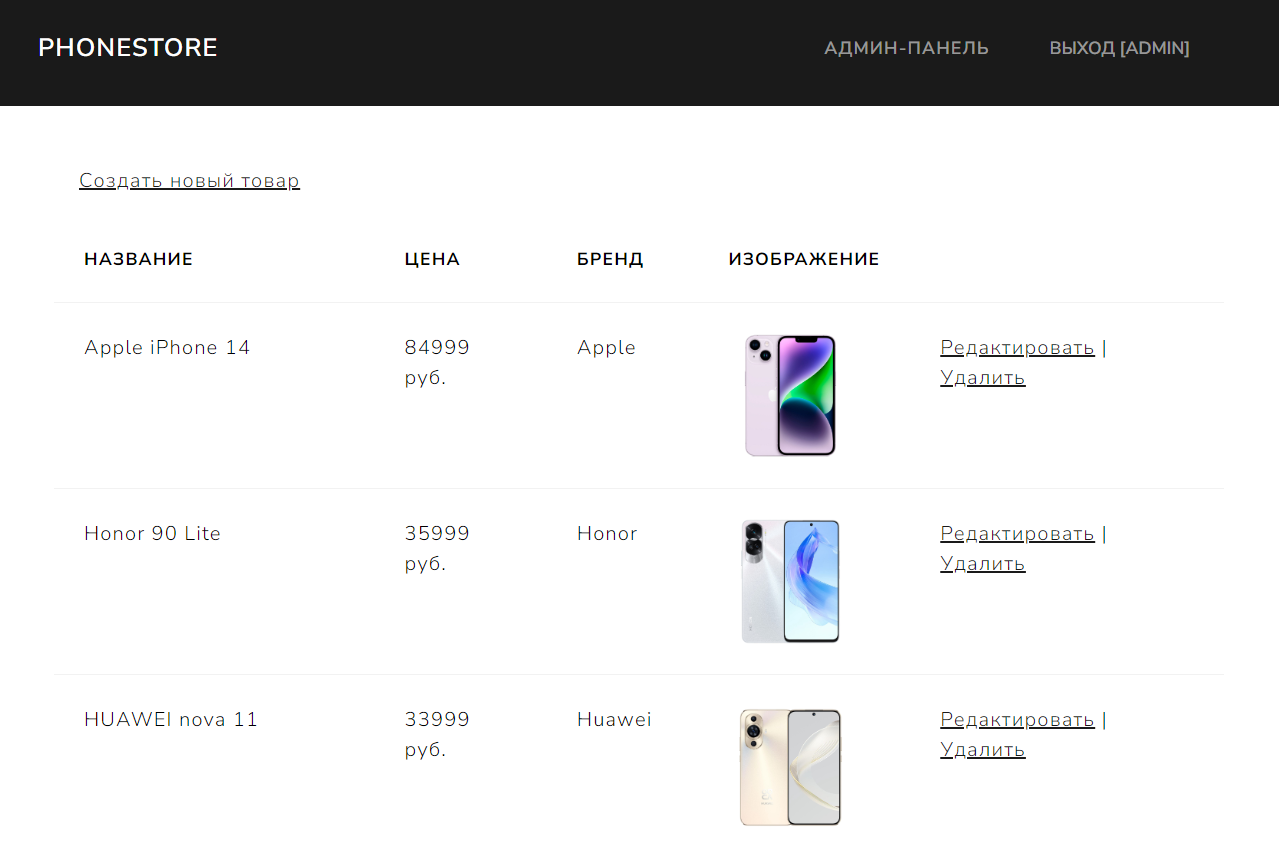


Рис. 27– Содержимое страницы админ-панели.

Отредактируем один из товаров, поменяем цену (рисунок 28-29).

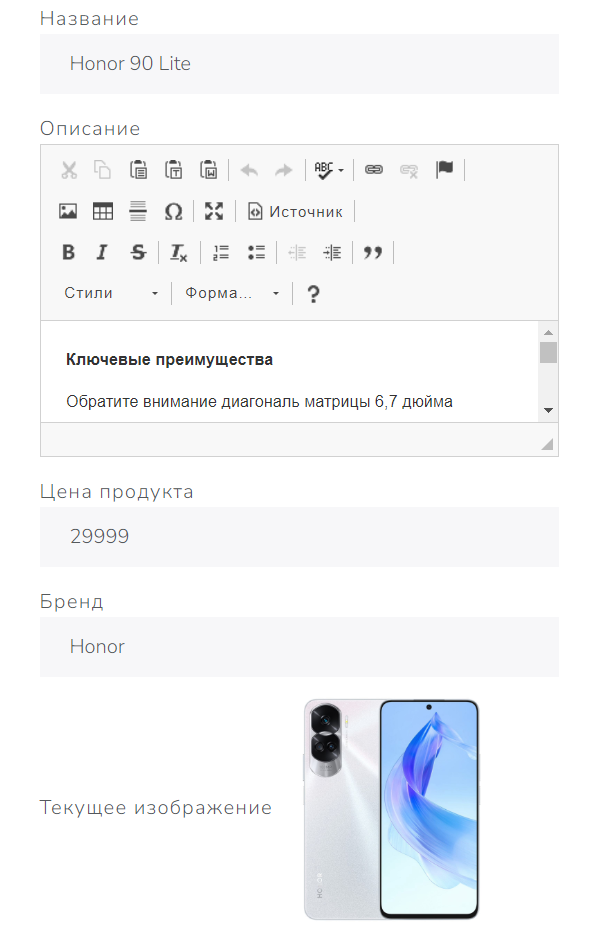


Рис. 28– Форма редактирования.

Найдя соответствующий товар в каталоге, можно убедиться, что внесенные изменения успешно применились (рисунок 29).

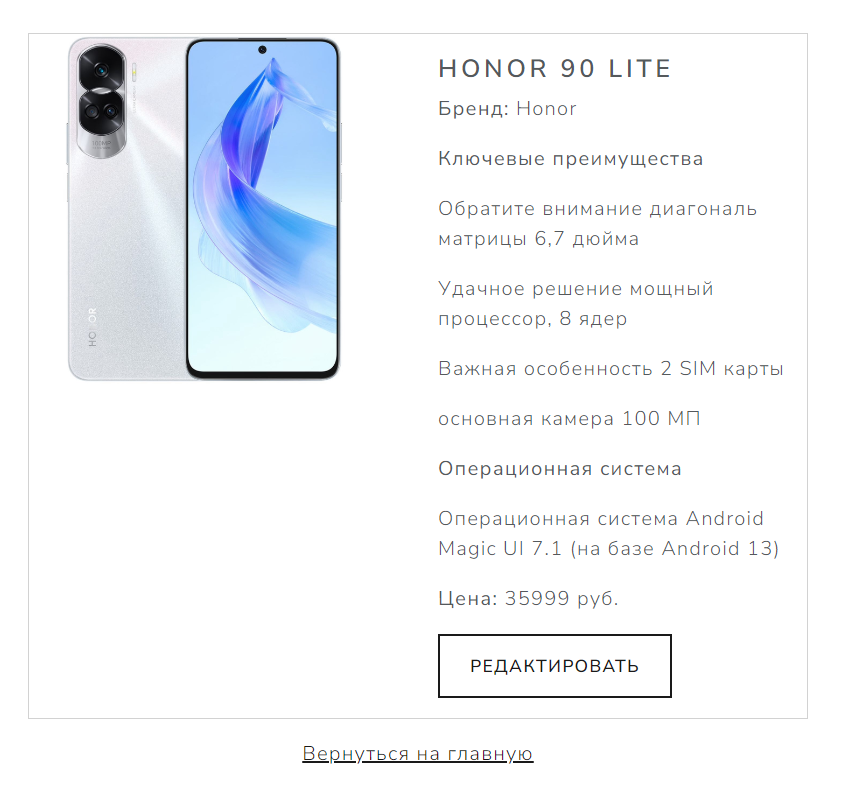


Рис. 29 – Результат редактирования.

Теперь попробуем создать новый товар (рисунок 30-31). Но не будем добавлять изображение товара.

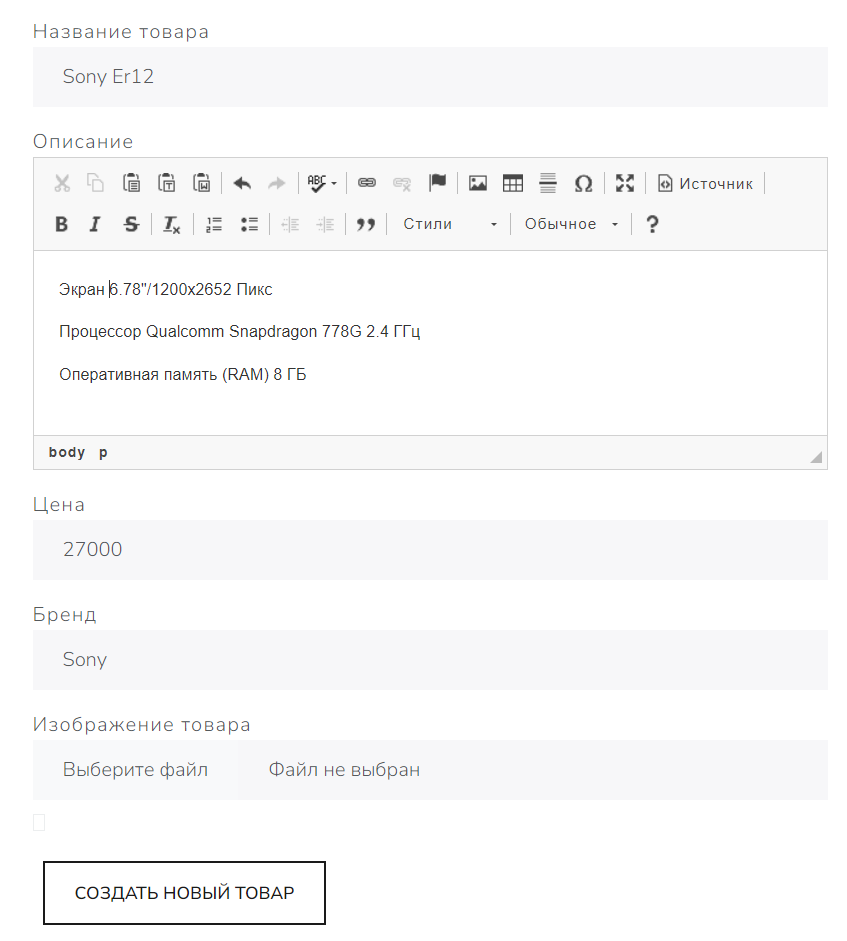


Рис. 30 – Форма добавления товара.

Товар был успешно добавлен в каталог.

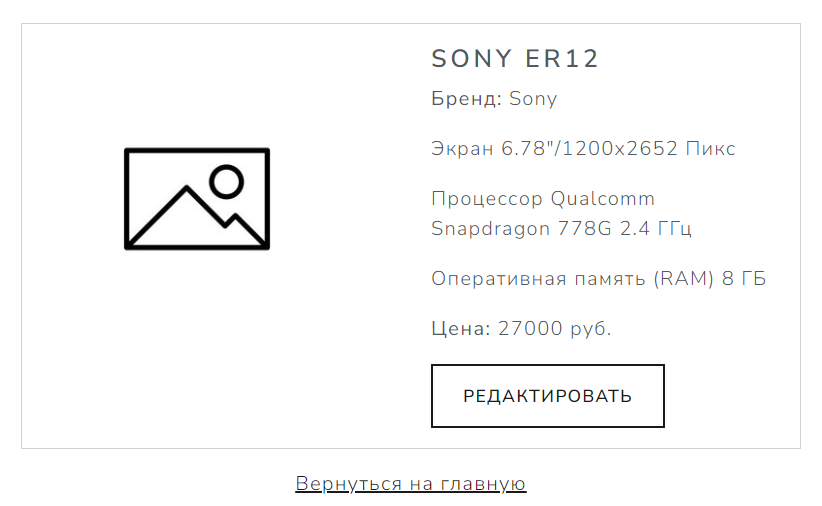


Рис. 31 – Результат добавления товара.

**ЗАКЛЮЧЕНИЕ**

Внедрение интернет-магазина смартфонов с использованием современных технологий разработки, таких как ASP.NET Core, предоставляет уникальные преимущества, делая процесс покупки более удобным и эффективным. Автоматизация рутинных задач и предоставление клиентам возможности выбора товаров в любое время и из любого места создают оптимальное виртуальное пространство для совершения покупок.

Технология ASP.NET Core позволяет абстрагироваться от элементов программирования низкого уровня, предоставляя гибкий инструментарий на платформе .NET. Разработчик имеет доступ к тем же средствам управления и технологиям, что и при создании десктопных приложений, сохраняя гибкость в принятии решений и обеспечивая возможность создания полноценных веб-приложений для развёртывания на веб-сервере.

Преимущество использования MVC и Entity Framework в ASP.NET Core заключается в четком разделении ответственности между компонентами системы, что упрощает поддержку и масштабирование приложения. Это также способствует более легкому тестированию, поскольку различные части приложения могут быть протестированы независимо. Entity Framework ускоряет разработку, автоматизируя большую часть кода доступа к данным, и улучшает качество кода за счет использования сильно типизированных объектов вместо строковых запросов.

Гибкость разработки веб-приложения и возможность подключения источника данных независимо от поставщика делают технологию ASP.NET Core хорошим инструментом для разработки программного обеспечения информационных систем в виде веб-сайта.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Джеймс Чамберс, Дэвид Пэкетт, Саймон Тиммс ASP.NET Core. Разработка приложений. [Текст] / Джеймс Чамберс, Дэвид Пэкетт, Саймон Тиммс — 3-е изд.. — Санкт-Петербург: Питер, 2018 — 464 c.
2. Адам Фримен ASP.NET Core MVC с примерами на C# для профессионалов [Текст] / Адам Фримен — 6-е изд.. — Москва: Диалектика, 2017 — 992 c.
3. Умрихин, Е. Д. Разработка веб-приложений с помощью ASP.Net Core MVC. [Текст] / Е. Д. Умрихин — 1-е изд.. — Санкт-Петербург: БХВ, 2023 — 416 c.
4. Лок Эндрю ASP.NET CORE в действии [Текст] / Лок Эндрю — 2-е изд.. — Санкт-Петербург: ДМК Пресс, 2021 — 906 c.
5. Е. Моргунов: PostgreSQL. Основы языка SQL – СПб: БХВ-Петербург, 2018. — 336 с.

**ПРИЛОЖЕНИЯ**

## ПРИЛОЖЕНИЕ А

using Microsoft.AspNetCore.Authentication.Cookies;

using Microsoft.AspNetCore.Localization;

using Microsoft.EntityFrameworkCore;

using PhoneStore.Context;

using System.Globalization;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddControllersWithViews();

builder.Services.AddDbContext<AppDbContext>(options =>

{

options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"));

});

builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)

.AddCookie(options =>

{

options.LoginPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");

options.AccessDeniedPath = new Microsoft.AspNetCore.Http.PathString("/Account/Login");

});

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession(options =>

{

options.IdleTimeout = TimeSpan.FromMinutes(30);

options.Cookie.IsEssential = true;

});

var app = builder.Build();

app.UseSession();

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

var supportedCultures = new[] { new CultureInfo("en-US") };

app.UseRequestLocalization(new RequestLocalizationOptions

{

DefaultRequestCulture = new RequestCulture("en-US"),

SupportedCultures = supportedCultures,

SupportedUICultures = supportedCultures

});

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.MapControllerRoute(

name: "Areas",

pattern: "{area:exists}/{controller=Products}/{action=Index}/{id?}");

app.MapControllerRoute(

name: "products",

pattern: "/products/{brandSlug?}",

defaults: new {controller = "Products", action = "Index"});

app.MapControllerRoute(

name: "products",

pattern: "/products/{id?}",

defaults: new { controller = "Products", action = "Details" });

app.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

var context = app.Services.CreateScope().ServiceProvider.GetRequiredService<AppDbContext>();

SeedData.SeedDatabase(context);

app.Run();

**ПРИЛОЖЕНИЕ Б**

**HomeController.cs**

namespace PhoneStore.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

private readonly AppDbContext \_context;

public HomeController(ILogger<HomeController> logger, AppDbContext context)

{

\_logger = logger;

\_context = context;

}

public async Task<IActionResult> Index(string brandId, string sortOrder, string sterm)

{

var query = \_context.Phones

.Include(p => p.Brand)

.Where(p => !p.IsDeleted);

int? brandIdValue = null;

if (!string.IsNullOrEmpty(brandId) && int.TryParse(brandId, out var parsedBrandIdValue))

{

brandIdValue = parsedBrandIdValue;

query = query.Where(p => p.BrandId == brandIdValue);

}

switch (sortOrder)

{

case "price\_asc":

{

query = query.OrderBy(p => p.Price);

break;

}

case "price\_desc":

{

query = query.OrderByDescending(p => p.Price);

break;

}

}

var phones = await query.ToListAsync();

var brands = await \_context.Brands.ToListAsync();

var viewModel = new PhoneViewModel

{

Phones = phones,

Brands = brands,

STerm = sterm,

BrandId = brandIdValue ?? 0,

SortOrder = sortOrder,

};

return View(viewModel);

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**AccountController.cs**

namespace PhoneStore.Controllers

{

public class AccountController : Controller

{

private AppDbContext \_context;

public AccountController(AppDbContext context)

{

\_context = context;

}

[HttpGet]

public IActionResult Create() => View();

[HttpPost]

public async Task<IActionResult> Create(User user)

{

if (ModelState.IsValid)

{

var existingUserName = await \_context.Users.FirstOrDefaultAsync(u => u.UserName == user.UserName);

if (existingUserName != null)

{

ModelState.AddModelError("", "Пользователь с таким именем уже существует");

return View(user);

}

var existingUserEmail = await \_context.Users.FirstOrDefaultAsync(u => u.Email == user.Email);

if (existingUserEmail != null)

{

ModelState.AddModelError("", "Пользователь с такой почтой уже существует");

return View(user);

}

user.Password = user.Password;

await \_context.Users.AddAsync(user);

await \_context.SaveChangesAsync();

TempData["Success"] = "Вы успешно зарегистрировались! Войдите в аккаунт.";

return RedirectToAction("Index", "Home");

}

return View(user);

}

[HttpGet]

public IActionResult Login() => View();

[HttpPost]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var user = await \_context.Users.FirstOrDefaultAsync(u => u.UserName == model.UserName && u.Password == model.Password);

if (user != null)

{

var claims = new List<Claim>

{

new Claim(ClaimsIdentity.DefaultNameClaimType, user.UserName),

new Claim(ClaimsIdentity.DefaultRoleClaimType, user.Role.ToString()),

new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),

new Claim(ClaimTypes.Name, user.UserName)

};

var claimsIdentity = new ClaimsIdentity(claims, CookieAuthenticationDefaults.AuthenticationScheme);

var claimsPrincipal = new ClaimsPrincipal(claimsIdentity);

await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme, claimsPrincipal);

return RedirectToAction("Index", "Home");

}

ModelState.AddModelError("", "Неверное имя пользователя или пароль");

}

return View(model);

}

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

await HttpContext.SignOutAsync(CookieAuthenticationDefaults.AuthenticationScheme);

return RedirectToAction("Index", "Home");

}

}

}

**CartController.cs**

namespace PhoneStore.Controllers

{

[Authorize]

public class CartController : Controller

{

private readonly AppDbContext \_context;

public CartController(AppDbContext context)

{

\_context = context;

}

private string GetUserId()

{

return User.FindFirstValue(ClaimTypes.NameIdentifier);

}

public IActionResult Index()

{

string userId = GetUserId();

List<CartItem> cart = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}") ?? new List<CartItem>();

CartViewModel cartVM = new()

{

CartItems = cart,

GrandTotal = cart.Sum(x => x.Quantity \* x.Price)

};

return View(cartVM);

}

public async Task<IActionResult> Add(int id)

{

Phone phone = await \_context.Phones.FindAsync(id);

string userId = GetUserId();

List<CartItem> cart = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}") ?? new List<CartItem>();

CartItem cartItem = cart.FirstOrDefault(c => c.PhoneId == id && c.UserId == userId);

if (cartItem == null)

{

cart.Add(new CartItem(phone, userId));

}

else

{

cartItem.Quantity += 1;

}

HttpContext.Session.SetJson($"Cart\_{userId}", cart);

return Redirect(Request.Headers["Referer"].ToString());

}

public async Task<IActionResult> Decrease(int id)

{

Phone phone = await \_context.Phones.FindAsync(id);

string userId = GetUserId();

List<CartItem> cart = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}");

CartItem cartItem = cart.Where(c => c.PhoneId == id).FirstOrDefault();

if (cartItem.Quantity > 1)

{

--cartItem.Quantity;

}

else

{

cart.RemoveAll(p => p.PhoneId == id);

}

if (cart.Count == 0)

{

HttpContext.Session.Remove($"Cart\_{userId}");

}

else

{

HttpContext.Session.SetJson($"Cart\_{userId}", cart);

}

return RedirectToAction("Index");

}

public async Task<IActionResult> Remove(int id)

{

string userId = GetUserId();

List<CartItem> cart = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}");

cart.RemoveAll(p => p.PhoneId == id);

if (cart.Count == 0)

{

HttpContext.Session.Remove($"Cart\_{userId}");

}

else

{

HttpContext.Session.SetJson($"Cart\_{userId}", cart);

}

return RedirectToAction("Index");

}

public int GetCartItemCount()

{

string userId = GetUserId();

List<CartItem> cart = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}") ?? new List<CartItem>();

return cart.Sum(item => item.Quantity);

}

public IActionResult GetTotalItemInCart()

{

int cartItem = GetCartItemCount();

return Ok(cartItem);

}

public IActionResult Clear()

{

string userId = GetUserId();

HttpContext.Session.Remove($"Cart\_{userId}");

return RedirectToAction("Index");

}

}

}

**OrdersController.cs**

namespace PhoneStore.Controllers

{

public class OrdersController : Controller

{

private readonly AppDbContext \_context;

public OrdersController(AppDbContext context)

{

\_context = context;

}

[HttpPost]

public async Task<IActionResult> Create(string pickupPoint)

{

var userId = int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier));

var cartItems = HttpContext.Session.GetJson<List<CartItem>>($"Cart\_{userId}");

if (cartItems != null)

{

foreach (var item in cartItems)

{

var phone = await \_context.Phones.FindAsync(item.PhoneId);

if (phone != null)

{

phone.CountSales += item.Quantity;

\_context.Update(phone);

}

}

await \_context.SaveChangesAsync();

}

CartViewModel cartVM = new()

{

CartItems = cartItems,

GrandTotal = cartItems.Sum(x => x.Quantity \* x.Price)

};

var order = new Order

{

UserId = userId,

OrderDate = DateTime.UtcNow,

TotalPrice = Math.Round(cartVM.GrandTotal, 2),

};

foreach (var item in cartVM.CartItems)

{

order.OrderItems.Add(new OrderItem

{

PhoneId = item.PhoneId,

Quantity = item.Quantity,

Price = item.Price

});

}

\_context.Orders.Add(order);

await \_context.SaveChangesAsync();

HttpContext.Session.Remove($"Cart\_{userId}");

TempData["Success"] = "Спасибо за покупку!";

return RedirectToAction("Index", "Home");

}

public async Task<IActionResult> Index()

{

var userId = int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier));

var orders = await \_context.Orders

.Where(o => o.UserId == userId)

.Include(o => o.OrderItems)

.ThenInclude(oi => oi.Phone)

.ToListAsync();

return View(orders);

}

public async Task<IActionResult> Details(int id)

{

var userId = int.Parse(User.FindFirstValue(ClaimTypes.NameIdentifier));

var order = await \_context.Orders

.Include(o => o.OrderItems)

.ThenInclude(oi => oi.Phone)

.FirstOrDefaultAsync(o => o.Id == id && o.UserId == userId);

if (order == null)

{

return NotFound();

}

return View(order);

}

}

}

**ProductsController.cs**

namespace PhoneStore.Controllers

{

public class ProductsController : Controller

{

private readonly AppDbContext \_context;

public ProductsController(AppDbContext context)

{

\_context = context;

}

public async Task<IActionResult> Index(string brandSlug = "", int p = 1, string sortOrder = "")

{

int pageSize = 4;

ViewBag.PageNumber = p;

ViewBag.PageRange = pageSize;

ViewBag.BrandSlug = brandSlug;

ViewBag.PageTarget = "/products";

ViewBag.AdditionalQueryParameters = $"&sortOrder={sortOrder}&brandSlug={brandSlug}";

IQueryable<Phone> query = \_context.Phones.Include(p => p.Brand).Where(p => !p.IsDeleted);

if (!string.IsNullOrEmpty(brandSlug))

{

Brand category = await \_context.Brands.Where(c => c.Slug == brandSlug).FirstOrDefaultAsync();

if (category == null) return RedirectToAction("Index");

query = query.Where(p => p.BrandId == category.Id);

}

switch (sortOrder)

{

case "asc":

query = query.OrderBy(p => p.Price);

break;

case "desc":

query = query.OrderByDescending(p => p.Price);

break;

default:

query = query.OrderByDescending(p => p.Id);

break;

}

ViewBag.TotalPages = (int)Math.Ceiling((decimal)query.Count() / pageSize);

var products = await query.Skip((p - 1) \* pageSize).Take(pageSize).ToListAsync();

return View(products);

}

[HttpGet("phone/{id}")]

public async Task<IActionResult> Details(int id)

{

var phone = await \_context.Phones.Include(p => p.Brand).Where(p => !p.IsDeleted).FirstOrDefaultAsync(p => p.Id == id);

if (phone == null) return NotFound();

return View(phone);

}

}

}

**ПРИЛОЖЕНИЕ В**

@model PhoneStore.Models.ViewModels.PhoneViewModel

@{

ViewData["Title"] = "Главная";

}

<**form** **asp-controller**="Home" **asp-action**="Index" method="get" class="row row-cols-lg-auto g-3 align-items-center">

<div class="col-12">

<label class="visually-hidden" for="brandId">Brands</label>

<select class="form-select" id="brandId" name="brandId">

<**option** selected>Выберите бренд</**option**>

@foreach (var brand in Model.Brands)

{

<**option** selected="@(brand.Id == Model.BrandId)" **value**="@brand.Id">@brand.BrandName</**option**>

}

</select>

</div>

<div class="col-12">

<label class="visually-hidden" for="sortOrder">Сортировка</label>

<select class="form-select" id="sortOrder" name="sortOrder">

<**option** selected>Сортировать</**option**>

<**option** **value**="price\_asc" selected="@("price\_asc" == Model.SortOrder)">По возрастанию цены</**option**>

<**option** **value**="price\_desc" selected="@("price\_desc" == Model.SortOrder)">По убыванию цены</**option**>

</select>

</div>

<div class="col-12">

<button type="submit" class="btn btn-primary">Поиск</button>

<a href="/Home/Index" class="btn btn-primary">Очистить</a>

</div>

</**form**>

<br></br>

<div class=" w-100 d-flex flex-wrap">

<div class="row">

@foreach (var item in Model.Phones)

{

<div class="card mx-2 mb-4" style="width: 16rem;">

@if (string.IsNullOrEmpty(item.Image))

{

<img style="width:100%; height:230px" src="/images/noimage.jpg" class="card-img-top" alt="...">

}

else

{

<img style="width:100%; height:230px" src="/images/@item.Image" class="card-img-top" alt="...">

}

<div class="card-body">

<h5 class="card-title">@item.Name</h5>

<p class="card-text">

<b>Brand: </b>@item.Brand.BrandName<br />

<b>Price: </b>@item.Price pуб<br />

</p>

</div>

<div class="card-footer text-center">

@if (!User.IsInRole("Admin"))

{

<**a** class="btn btn-primary mx-auto" **asp-controller**="Cart" **asp-action**="Add" **asp-route-id**="@item.Id">Добавить в корзину</**a**>

<p></p>

}

<**a** class="btn btn-primary mx-auto" **asp-controller**="Products" **asp-action**="Details" **asp-route-id**="@item.Id">Подробнее</**a**>

</div>

</div>

}

</div>

</div>

@section Scripts{

<script>

async function add(phoneId) {

var usernameEl = document.getElementById("username");

if (usernameEl == null) {

window.location.href = "/Identity/Account/Login";

}

try {

var response = await fetch(`/Cart/AddItem?phoneId=${phoneId}`);

if (response.status == 200) {

var result = await response.json();

var cartCountEl = document.getElementById("cartCount");

cartCountEl.innerHTML = result;

window.location.href = "#cartCount";

}

}

catch (err) {

console.log(err);

}

}

</script>

}