МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Чувашский государственный университет имени И.Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра компьютерных технологий

КУРСОВОЙ ПРОЕКТ

по дисциплине «СОВРЕМЕННЫЕ ЯЗЫКИ ПРОГРАММИРОВАНИЯ»

на тему «Разработка онлайн-магазина по продаже брендовой одежды»

Выполнил: студент

группы ЗКТ-45-21

Башкиров Денис Владимирович

Проверил: доцент кафедры компьютерных технологий, к.ф.-м.н., Тихонов Сергей Владимирович

Чебоксары, 2025

**Содержание**

[**Введение** 3](#_Toc219205437)

[**ГЛАВА 1. Проектирование структуры веб-приложения** 4](#_Toc219205438)

[1.1. Анализ предметной области 4](#_Toc219205439)

[1.2. Концептуальное проектирование 9](#_Toc219205440)

[1.3. Логическое проектирование 12](#_Toc219205441)

[1.4. Физическое проектирование 17](#_Toc219205442)

[**ГЛАВА 2. Описание функционирования веб-приложения** 23](#_Toc219205443)

[2.1. Назначение и перечень функций веб-приложения 23](#_Toc219205444)

[2.2. Программная реализация веб-приложения 25](#_Toc219205445)

[2.2. Демонстрация веб-приложения 43](#_Toc219205446)

[**Заключение** 53](#_Toc219205447)

[**Список используемых источников** 54](#_Toc219205448)

**Введение**

Объектом исследования курсового проекта является веб-приложение для автоматизации процессов продажи брендовой одежды через интернет-магазин.

Целью курсового проекта является разработка веб-приложения онлайн-магазина по продаже брендовой одежды на базе технологий ASP.NET Core Blazor Server.

Данная тема актуальна, так как в настоящее время интернет-торговля является одним из наиболее быстро развивающихся сегментов экономики. Рост популярности онлайн-покупок связан с удобством выбора товаров, возможностью сравнения цен, экономией времени и доступностью товаров независимо от географического расположения покупателя. Особенно это актуально для сферы продажи брендовой одежды, где покупателям важно иметь доступ к широкому ассортименту товаров от различных производителей, возможность фильтрации по брендам, категориям, размерам и ценам. Автоматизация процессов управления товарами, заказами и пользователями значительно повышает эффективность работы магазина и улучшает качество обслуживания клиентов [1].

Задачи курсового проекта:

1. Провести анализ предметной области, выполнить концептуальное, логическое и физическое проектирование веб-приложения онлайн-магазина по продаже брендовой одежды.

2. Разработать программу для автоматизации работы интернет-магазина с функциями каталога товаров, корзины покупок, системы заказов, авторизации пользователей и управления контентом.

# **ГЛАВА 1. Проектирование структуры веб-приложения**

## 1.1. Анализ предметной области

Поиск и изучение информации о потребностях заказчика, а также выяснение смысла данной информации для дальнейшей работы с проектом, называется анализом предметной области. Анализ предметной области – это первая фаза проектирования предметной области [2].

В качестве предметной области в данной курсовой работе рассматривается деятельность интернет-магазина по продаже брендовой одежды.

Выбор данной области был связан с потребностями рынка. С ростом объема информации обусловлено увеличение объема продаж, усложнение структуры ассортимента, необходимость управления большим количеством товарных позиций от различных брендов.

Каждому интернет-магазину нужен спрос на предоставляемые товары, поэтому в настоящий момент недостаточно просто продавать товар, чтобы удержаться на рынке. Сейчас важным фактором предприятия является современный веб-сайт с удобным интерфейсом для покупателей, предоставляющий возможность быстрого поиска нужных товаров, фильтрации по различным параметрам, оформления заказов и управления личным кабинетом. Также необходима административная панель для управления товарами, заказами и пользователями.

Реализация таких решений даёт возможность магазину увеличить покупательский спрос, прибыль и сэкономить время как клиентов, так и персонала магазина. Поэтому интернет-магазин брендовой одежды должен выполнять следующие основные функции:

* управление продажами (каталог товаров, корзина покупок, оформление заказов);
* управление товарным ассортиментом (добавление, редактирование, удаление товаров, брендов, категорий);
* управление заказами (просмотр, изменение статусов, обработка заказов);
* управление пользователями и кадровый учет (регистрация, авторизация, роли, личные кабинеты);
* информационная (обеспечение клиентов информацией о товарах, брендах, доступности).

Данные функции реализуются в большинстве современных интернет-магазинов.

В настоящее время существует множество готовых решений для создания интернет-магазинов. Рассмотрим несколько примеров успешных интернет-магазинов брендовой одежды.

Данные функции реализуются в большинстве разных производств.

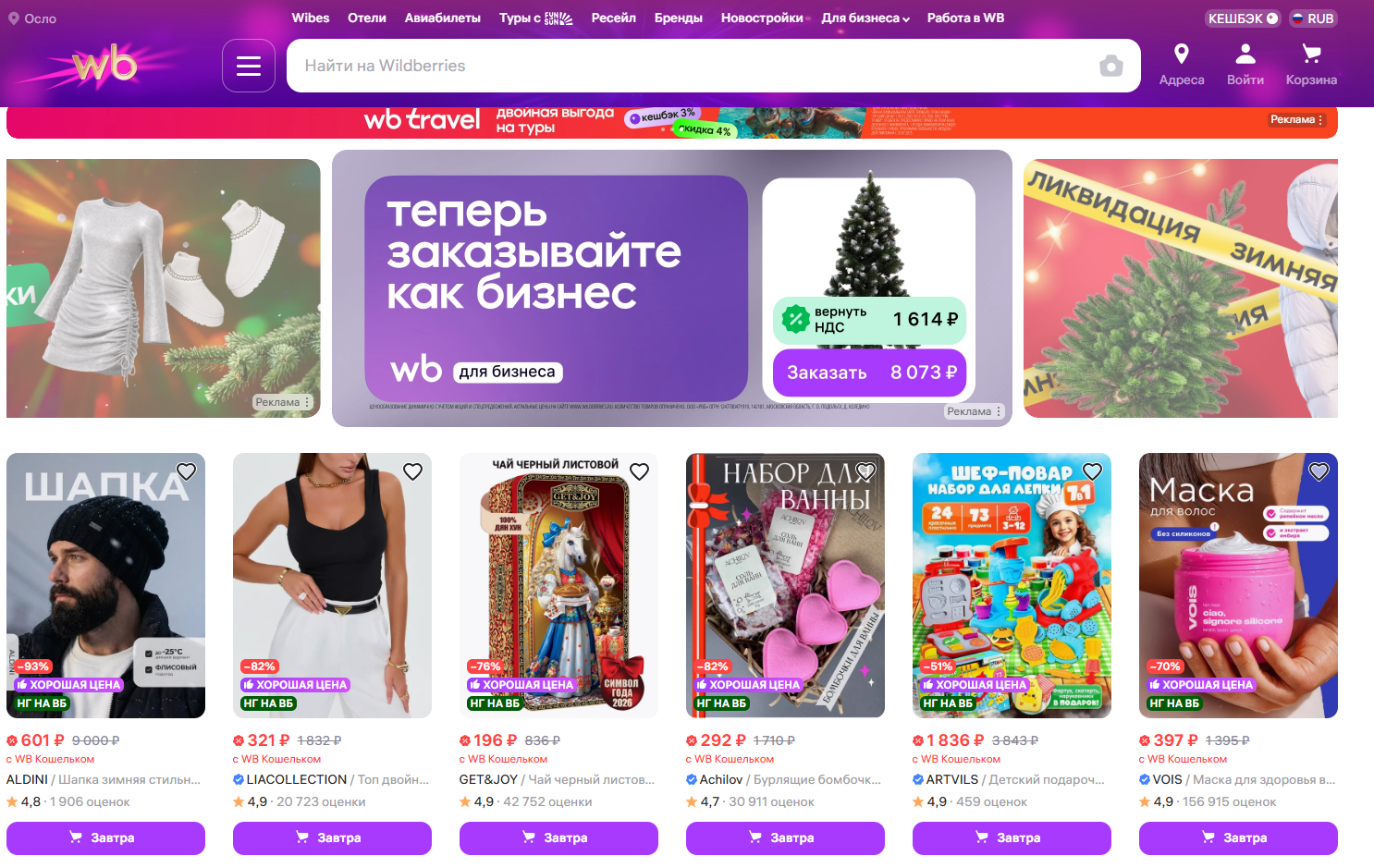
Одним из популярных примеров является сайт «Wildberries» (https://www.wildberries.ru/). Данный интернет-магазин предоставляет широкий ассортимент товаров различных категорий, включая брендовую одежду. На сайте реализованы функции каталога с фильтрацией по различным параметрам, корзины покупок, личного кабинета пользователя, отслеживания заказов. (рис. 1.1).

Рисунок 1.1 − Главная страница сайта «Wildberries»

Другим примером является сайт «Lamoda» (https://www.lamoda.ru/). Этот интернет-магазин специализируется на продаже модной одежды и обуви. На сайте реализованы функции каталога товаров с детальными описаниями, изображениями, размерами, фильтрации по брендам, категориям, размерам, цветам и ценам, корзины покупок, оформления заказов с выбором способа доставки, системы скидок и акций, личного кабинета с историей заказов. (рис. 1.2).

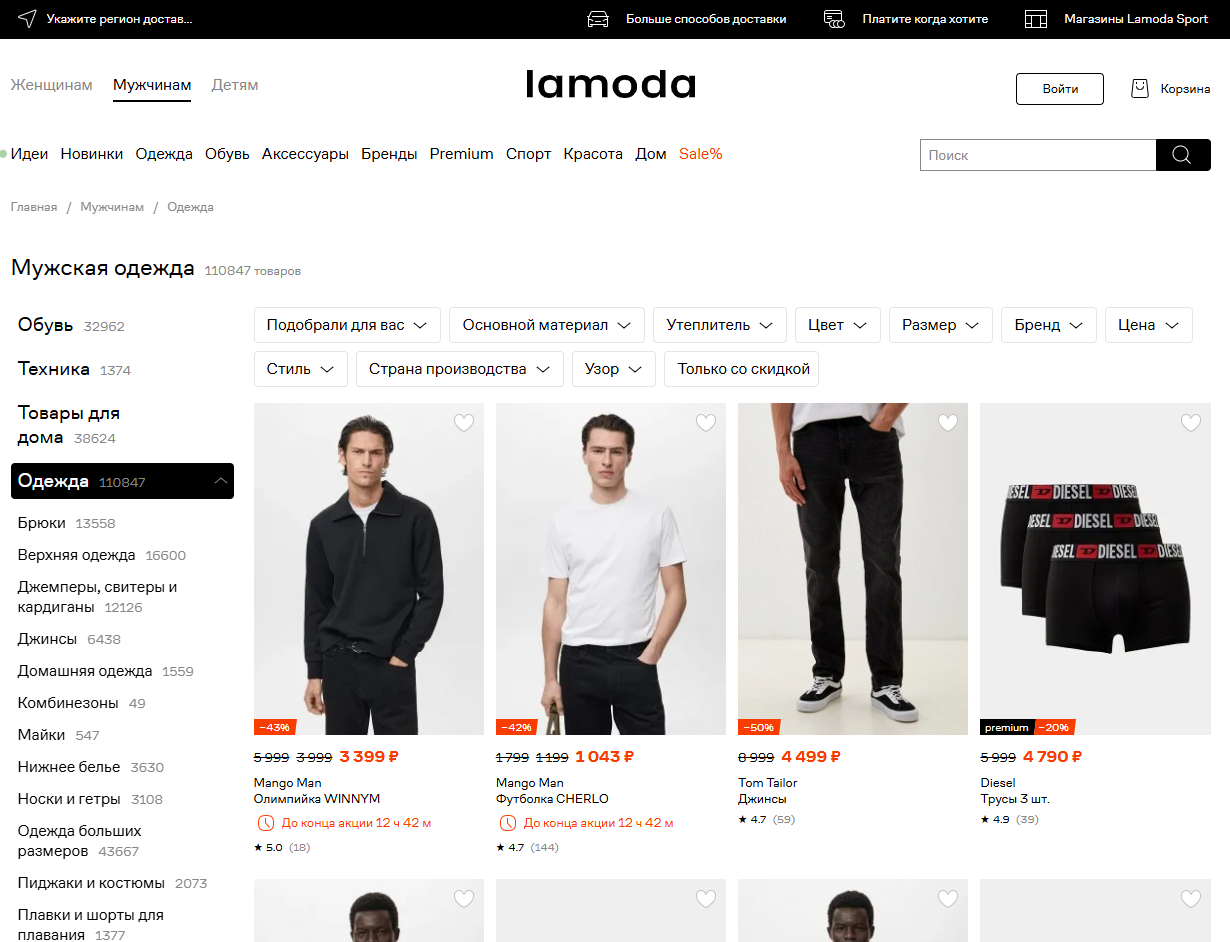


Рисунок 1.2 − Главная страница сайта «Lamoda»

Также можно рассмотреть интернет-магазин «Brandshop» (https://brandshop.ru/), который специализируется на продаже брендовой одежды и обуви. На сайте реализованы функции каталога с подробными карточками товаров, фильтрации по брендам и категориям, корзины покупок, системы авторизации и регистрации, управления заказами, программы лояльности. (рис. 1.3).

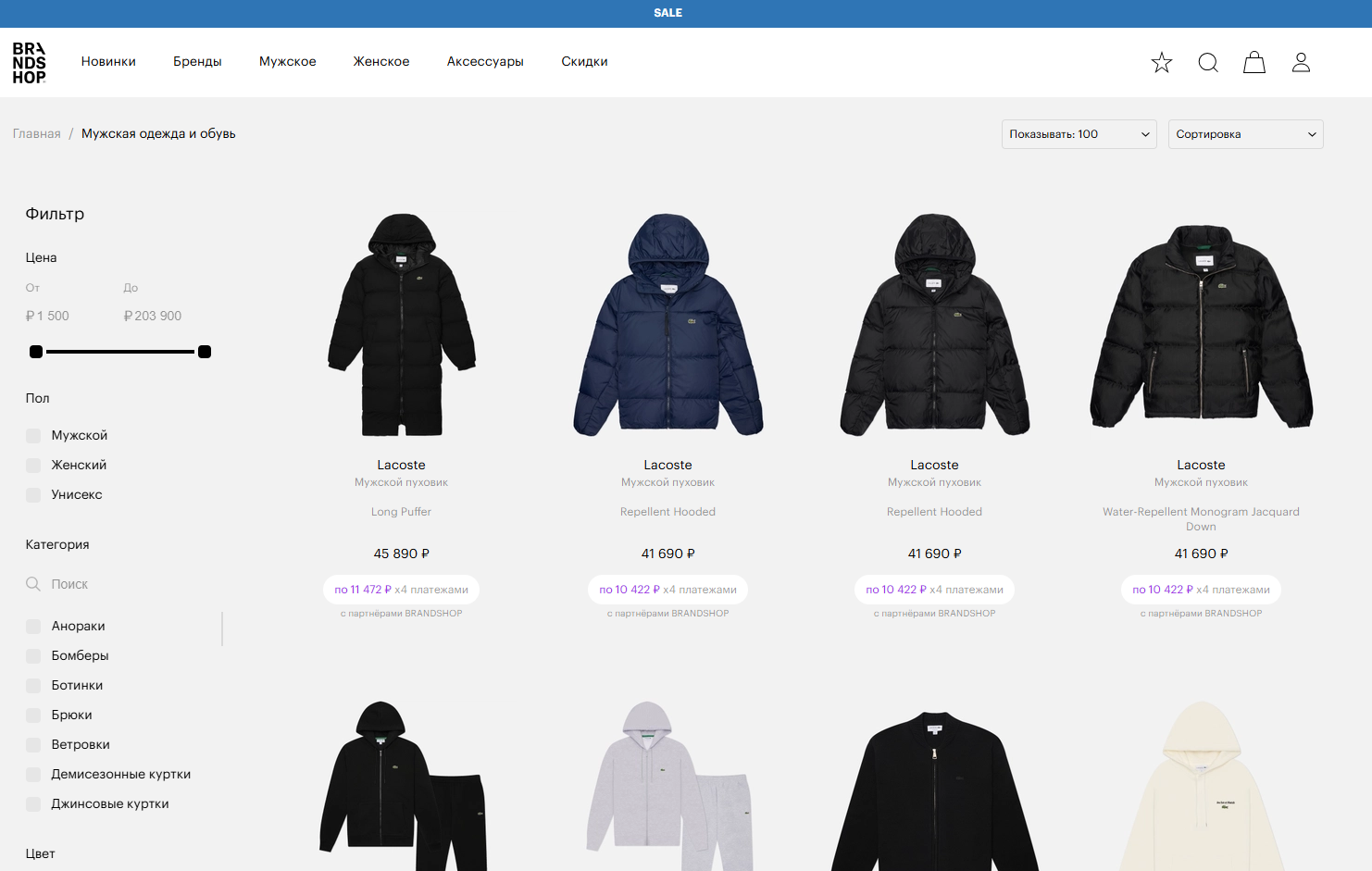


Рисунок 1.3 − Главная страница сайта «Brandshop»

При сравнении рассмотренных интернет-магазинов было выявлено, что все они имеют схожий функционал: каталог товаров с фильтрацией, корзину покупок, систему заказов, авторизацию пользователей. Основные отличия заключаются в дизайне интерфейса, дополнительных функциях (например, программа лояльности, системы рекомендаций) и масштабе ассортимента.

На основе проведённого анализа предметной области был выделен перечень функций, которые будут реализованы в курсовом проекте:

1. Каталог товаров с фильтрацией по брендам, категориям и поиском по названию.
2. Детальный просмотр информации о товаре.
3. Корзина покупок с возможностью изменения количества товаров.
4. Оформление заказов с указанием адреса доставки.
5. Система авторизации и регистрации пользователей с дополнительными полями (пол, дата рождения, предпочитаемый размер, город).
6. Система ролей и разграничение прав доступа (Администратор, Менеджер, Покупатель).
7. Административная панель для управления товарами (CRUD-операции).
8. Административная панель для управления брендами и категориями (CRUD-операции).
9. Административная панель для управления заказами.
10. Экспорт и импорт данных в формате Excel.
11. Личный кабинет пользователя с историей заказов.

## 1.2. Концептуальное проектирование

Концептуальное проектирование заключается в формализованном описании предметной области, которое должно быть таким, чтобы с одной стороны, можно было проанализировать корректность схемы БД, а с другой стороны не должно быть привязано к конкретной СУБД.

Любая БД создаётся для решения определенных прикладных задач: в простейшем случае – накопление и выдача данных, в более сложном – решение задач, использующих данные из БД в качестве исходных данных.

Для концептуального проектирования используется диаграмма вариантов использования.

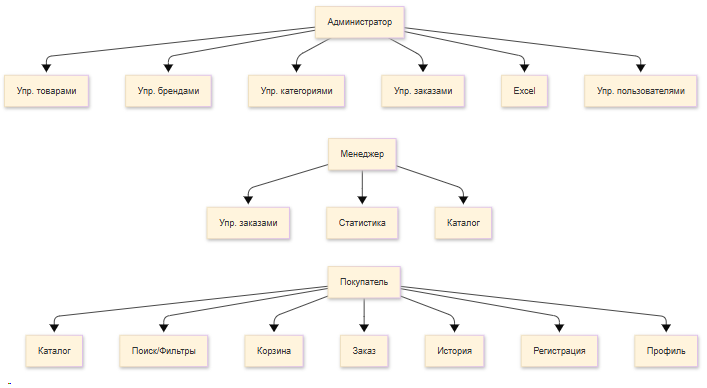
Диаграмма вариантов использования (Use Case Diagram) состоит из графической диаграммы, описывающей действующие лица и прецеденты, а также спецификации, представляющей собой текстовое описание конкретных последовательностей действий (потока событий), которые выполняет пользователь при работе с системой.

Главное назначение диаграммы вариантов использования – формализация функциональных требований к системе с помощью действующих лиц (потенциальных пользователей системы) и вариантов использования (задачи, с которыми действующее лицо обращается к разрабатываемому продукту).

На диаграмме использования изображаются:

* Действующие лица – группы лиц или систем, взаимодействующих с нашей системой;
* Варианты использования (прецеденты) – сервисы, которые наша система предоставляет;
* Отношения между элементами диаграммы.

Выделим свойства и структуру базы данных для автоматизации интернет-магазина брендовой одежды.

 Рисунок 1.4 − Диаграмма использования

Действующие лица:

* Администратор – имеет полный доступ ко всем функциям системы, управляет товарами, брендами, категориями, заказами и пользователями;
* Менеджер – управляет заказами, просматривает статистику, имеет доступ к управлению товарами;
* Покупатель (Customer) – просматривает каталог, добавляет товары в корзину, оформляет заказы, управляет личным кабинетом.

Рассмотрим варианты использования для каждого действующего лица.

**Администратор:**

1. Управление товарами (добавление, редактирование, удаление товаров).
2. Управление брендами (добавление, редактирование, удаление брендов).
3. Управление категориями (добавление, редактирование, удаление категорий).
4. Управление заказами (просмотр всех заказов, изменение статусов заказов).
5. Экспорт и импорт данных в Excel.
6. Просмотр информации о пользователях.
7. Управление ролями пользователей.

**Менеджер:**

1. Просмотр и управление заказами (изменение статусов заказов).
2. Просмотр каталога товаров.
3. Просмотр статистики по заказам.

**Покупатель:**

1. Просмотр каталога товаров.
2. Поиск товаров по названию.
3. Фильтрация товаров по брендам и категориям.
4. Просмотр детальной информации о товаре.
5. Добавление товаров в корзину.
6. Управление корзиной (изменение количества, удаление товаров).
7. Оформление заказа.
8. Просмотр истории своих заказов.
9. Регистрация и авторизация в системе.
10. Управление профилем пользователя.

Результатом этого этапа проектирования является построение первичной информационной структуры базы данных, которая называется концептуальной схемой базы данных или инфологической моделью.

Концептуальная схема базы данных содержит сгруппированные атрибуты предметной области по признакам функциональной зависимости.

В первом приближении для решения выделенных задач необходимо хранение данных о следующих объектах:

1. Товар (Product) – информация о товарах магазина.
2. Бренд (Brand) – информация о брендах одежды.
3. Категория (Category) – информация о категориях товаров.
4. Заказ (Order) – информация о заказах покупателей.
5. Элемент заказа (OrderItem) – информация о товарах, входящих в заказ.
6. Пользователь (ApplicationUser) – информация о пользователях системы (расширение стандартной модели Identity).

## 1.3. Логическое проектирование

Логическое проектирование – создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели данных логическая модель – набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи.

Преобразование концептуальной модели в логическую модель, как правило, осуществляется по формальным правилам. Этот этап может быть в значительной степени автоматизирован.

При создании моделей данных используется метод семантического моделирования. Семантическое моделирование основывается на значении структурных компонентов или характеристик данных, что способствует правильности их интерпретации (понимания, разъяснения). В качестве инструмента семантического моделирования используются различные варианты диаграмм “сущность-связь” [7].

Существуют различные варианты отображения ER-диаграммы, но все варианты диаграмм “сущность-связь” исходят из одной идеи – рисунок всегда нагляднее текстового описания. ER-диаграммы используют графическое изображение сущностей предметной области, их свойств (атрибутов), и взаимосвязей между сущностями [6].

Основные преимущества ER-моделей:

* наглядность;
* модели позволяют проектировать базы данных с большим количеством объектов и атрибутов;
* ER-модели реализованы во многих системах автоматизированного проектирования баз данных.

Основные элементы ER-моделей:

* объекты (сущности);
* атрибуты объектов;
* связи между объектами.

Сущность – объект предметной области, имеющий атрибуты.

Связь между сущностями характеризуется:

* типом связи (1:1, 1:N, N:М);
* классом принадлежности.

Результатом логического проектирования является логическая модель данных, состоящая из ER-диаграммы или диаграммы отношений, а также из реляционной схемы..

Рассмотрим ER-диаграмму "Товар принадлежит бренду" (рис. 1.5).



Рисунок 1.5 – ER-диаграмма “ Товар принадлежит бренду”

Связь ПРИНАДЛЕЖИТ имеет тип M:1, так как много товаров могут принадлежать одному бренду. Сущность ТОВАР имеет обязательный класс принадлежности, так как каждый товар обязательно принадлежит бренду. Сущность БРЕНД имеет необязательный класс принадлежности, так как не каждый бренд обязательно имеет товары в системе.

Рассмотрим ER-диаграмму “ Товар принадлежит категории” (рис. 1.6).



Рисунок 1.6 – ER-диаграмма “ Товар принадлежит категории”

Связь ПРИНАДЛЕЖИТ имеет тип M:1, так как много товаров могут принадлежать одной категории. Сущность ТОВАР имеет обязательный класс принадлежности, так как каждый товар обязательно принадлежит категории. Сущность КАТЕГОРИЯ имеет необязательный класс принадлежности, так как не каждая категория обязательно имеет товары в системе.

Рассмотрим ER-диаграмму “ Пользователь оформляет заказ” (рис. 1.7)

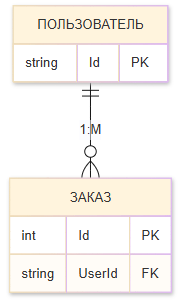


Рисунок 1.7 – ER-диаграмма “Пользователь оформляет заказ”

Связь ОФОРМЛЯЕТ имеет тип 1:M, так как один пользователь может оформить несколько заказов. Сущность ПОЛЬЗОВАТЕЛЬ имеет необязательный класс принадлежности, так как не каждый пользователь обязательно оформляет заказы. Сущность ЗАКАЗ имеет обязательный класс принадлежности, так как каждый заказ обязательно принадлежит пользователю.

Рассмотрим ER-диаграмму “Заказ состоит из элементов заказа” (рис. 1.8)

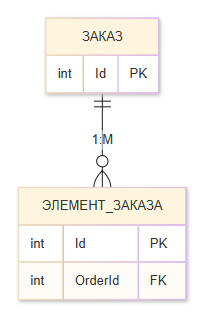


Рисунок 1.8 – ER-диаграмма “Заказ состоит из элементов заказа”

Связь СОСТОИТ ИЗ имеет тип 1:M, так как один заказ может содержать несколько элементов заказа. Сущность ЗАКАЗ имеет необязательный класс принадлежности, так как заказ может быть пустым (хотя на практике это маловероятно). Сущность ЭЛЕМЕНТ\_ЗАКАЗА имеет обязательный класс принадлежности, так как каждый элемент заказа обязательно принадлежит заказу.

Рассмотрим ER-диаграмму “Элемент заказа содержит товар” (рис. 1.9).

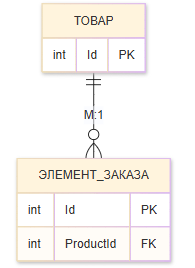


Рисунок 1.9 – ER-диаграмма “Элемент заказа содержит товар”

Связь СОДЕРЖИТ имеет тип M:1, так как много элементов заказа могут содержать один товар. Сущность ЭЛЕМЕНТ\_ЗАКАЗА имеет обязательный класс принадлежности, так как каждый элемент заказа обязательно содержит товар. Сущность ТОВАР имеет необязательный класс принадлежности, так как не каждый товар обязательно входит в заказы.

Формирование отношений на основе диаграмм ER-типа

Правило 4. Если степень связи 1:M (M:1) и класс принадлежности М-связной сущности – обязательный, то, независимо от КП другой сущности, достаточно сформировать два отношения: по одному для каждой сущности с соответствующими ключами. При этом ключ односвязной сущности добавляется в качестве атрибута в отношение для многосвязной сущности.

Правило 6. Если степень связи M:M, то необходимо формирование трех отношений: по одному для каждой сущности с соответствующими ключами и одно, выражающее связь и содержащее только ключи из каждой сущности.

Сформируем набор предварительных отношений на основе ER-диаграмм с указанием первичного ключа для каждого отношения:

1. Связь "Товар принадлежит бренду" (Рисунок 1.1), по правилу №4 формируем два отношения:
   * Бренд (Id, ...);
   * Товар (Id, BrandId, ...).
2. Связь "Товар принадлежит категории" (Рисунок 1.2), по правилу №4 формируем два отношения:
   * Категория (Id, ...);
   * Товар (Id, CategoryId, ...).
3. Связь "Пользователь оформляет заказ" (Рисунок 1.3), по правилу №4 формируем два отношения:
   * Пользователь (Id, ...);
   * Заказ (Id, UserId, ...).
4. Связь "Заказ состоит из элементов заказа" (Рисунок 1.4), по правилу №4 формируем два отношения:
   * Заказ (Id, ...);
   * ЭлементЗаказа (Id, OrderId, ...).
5. Связь "Элемент заказа содержит товар" (Рисунок 1.5), по правилу №4 формируем два отношения:
   * Товар (Id, ...);
   * ЭлементЗаказа (Id, ProductId, ...).

Набор предварительных отношений с указанием предполагаемого первичного ключа и основных атрибутов для каждого отношения:

1. Бренд (Id, Name, Description, Country, CreatedAt).
2. Категория (Id, Name, Description, CreatedAt).
3. Товар (Id, Name, Description, Price, Sizes, Color, StockQuantity, IsAvailable, BrandId, CategoryId, CreatedAt, UpdatedAt).
4. Пользователь (Id, UserName, Email, FirstName, LastName, Gender, BirthDate, PreferredSize, City, RegistrationDate).
5. Заказ (Id, OrderNumber, UserId, OrderDate, Status, TotalAmount, DeliveryAddress, Comment, RecipientPhone, RecipientName).
6. ЭлементЗаказа (Id, OrderId, ProductId, Quantity, UnitPrice, Size).

## 1.4. Физическое проектирование

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД [4].

На этапах концептуального и логического проектирования не принимаются во внимание конкретные возможности целевой СУБД.

В качестве СУБД для данной курсовой работы был выбран Microsoft SQL Server. SQL Server Management Studio (SSMS) – это графический набор средств для разработки сценариев на T-SQL и управления всеми компонентами SQL Server. Management Studio является основным инструментом любого разработчика или администратора MS SQL Server [5]. Перечислим основные возможности среды SQL Server Management Studio:

* Подключение к любому компоненту SQL Server.
* Обозреватель объектов.
* Создание и редактирование сценариев.
* Настройка свойств сервера, баз данных и других объектов.

На этапе логического проектирования были разработаны ER-диаграммы. Далее нужно на их основе определить требования к структуре таблиц и физически реализовать таблицы в СУБД.

Рассмотрим структуру таблиц базы данных.

Таблица 1.1 – Структура таблицы «Brands»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Id | int | NOT NULL | Первичный | — | Identity(1,1) | — |
| Name | nvarchar(100) | NOT NULL | — | — | — | — |
| Description | nvarchar(500) | NULL | — | — | — | — |
| Country | nvarchar(50) | NULL | — | — | — | — |
| CreatedAt | datetime2 | NOT NULL | — | GETDATE() | — | — |

Таблица 1.2 – Структура таблицы «Categories»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Id | int | NOT NULL | Первичный | — | Identity(1,1) | — |
| Name | nvarchar(100) | NOT NULL | — | — | — | — |
| Description | nvarchar(500) | NULL | — | — | — | — |
| CreatedAt | datetime2 | NOT NULL | — | GETDATE() | — | — |

Таблица 1.3 – Структура таблицы «Products»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Id | int | NOT NULL | Первичный | — | Identity(1,1) |  |
| Name | nvarchar(200) | NOT NULL | — | — | — | — |
| Description | nvarchar(2000) | NULL | — | — | — | — |
| Price | decimal(18,2) | NOT NULL | — | — | Price > 0 | — |
| Sizes | nvarchar(100) | NULL | — | — | — | — |
| Color | nvarchar(50) | NULL | — | — | — | — |
| StockQuantity | int | NOT NULL | — | 0 | StockQuantity >= 0 | — |
| IsAvailable | bit | NOT NULL | — | 1 | — | — |
| BrandId | int | NOT NULL | Внешний | — | — | Brands.Id |
| CategoryId | int | NOT NULL | Внешний | — | — | Categories.Id |
| CreatedAt | datetime2 | NOT NULL | — | GETDATE() | — | — |
| UpdatedAt | datetime2 | NULL | — | — | — | — |

Таблица 1.4 ­– Структура таблицы «Orders»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Id | int | NOT NULL | Первичный | — | Identity(1,1) | — |
| OrderNumber | nvarchar(50) | NOT NULL | Уникальный | — | — | — |
| UserId | nvarchar(450) | NOT NULL | Внешний | — | — | AspNetUsers.Id |
| OrderDate | datetime2 | NOT NULL | — | GETDATE() | — | — |
| Status | nvarchar(50) | NOT NULL | — | 'Новый' | — | — |
| TotalAmount | decimal(18,2) | NOT NULL | — | — | TotalAmount >= 0 | — |
| DeliveryAddress | nvarchar(500) | NULL | — | — | — | — |
| Comment | nvarchar(1000) | NULL | — | — | — | — |
| RecipientPhone | nvarchar(20) | NULL | — | — | — | — |
| RecipientName | nvarchar(100) | NULL | — | — | — | — |

Таблица 1.5 – Структура таблицы «OrderItems»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| Id | int | NOT NULL | Первичный | — | Identity(1,1) | — |
| OrderId | int | NOT NULL | Внешний | — | — | Orders.Id |
| ProductId | int | NOT NULL | Внешний | — | — | Products.Id |
| Quantity | int | NOT NULL | — | 1 | Quantity > 0 | — |
| UnitPrice | decimal(18,2) | NOT NULL | — | — | UnitPrice >= 0 | — |
| Size | nvarchar(20) | NULL | — | — | — | — |

Таблица 1.6 – Структура таблицы «AspNetUsers»

Таблица AspNetUsers является частью системы ASP.NET Core Identity и содержит стандартные поля для хранения пользователей, а также дополнительные поля, расширяющие модель ApplicationUser:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Описание** |
| Id | nvarchar(450) | NOT NULL | Первичный | Уникальный идентификатор пользователя |
| UserName | nvarchar(256) | NULL | — | Имя пользователя |
| Email | nvarchar(256) | NULL | — | Email пользователя |
| FirstName | nvarchar(50) | NULL | — | Имя пользователя |
| LastName | nvarchar(50) | NULL | — | Фамилия пользователя |
| Gender | nvarchar(20) | NULL |  | Пол пользователя |
| BirthDate | datetime2 | NULL |  | Дата рождения |
| PreferredSize | nvarchar(10) | NULL |  | Предпочитаемый размер одежды |
| City | nvarchar(100) | NULL |  | Город пользователя |
| RegistrationDate | datetime2 | NOT NULL |  | Дата регистрации |

Таблица 1.7 – Структура таблицы «Поставка»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_поставки | Целое | NOT NULL | Первичный | — | — | — |
| Id\_поставщика | Целое | NOT NULL | Внешний | — | — | Id\_поставщика в таблице «Поставщик» |
| Дата | Дата | NOT NULL | — | — | — | — |
| Время | Время | NOT NULL | — | — | — | — |

Таблица 1.8 – Структура таблицы «СоставПоставки»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Столбец** | **Тип данных** | **NULL** | **Ключ** | **Значение по умолчанию** | **Ограничение** | **Ссылка** |
| id\_поставки | Целое | NOT NULL | Первичный/Внешний | — | — | id\_поставки в таблице «Поставка» |
| id\_ткани | Целое | NOT NULL | — | — | id\_ткани в таблице «Ткань» |
| Количество | Целое | NOT NULL | — | — | >20 | — |
| Вес | Целое | NOT NULL | — | — | — | — |

Свяжем таблицы и создадим схему баз данных, содержащую эти таблицы (рис. 1.22)

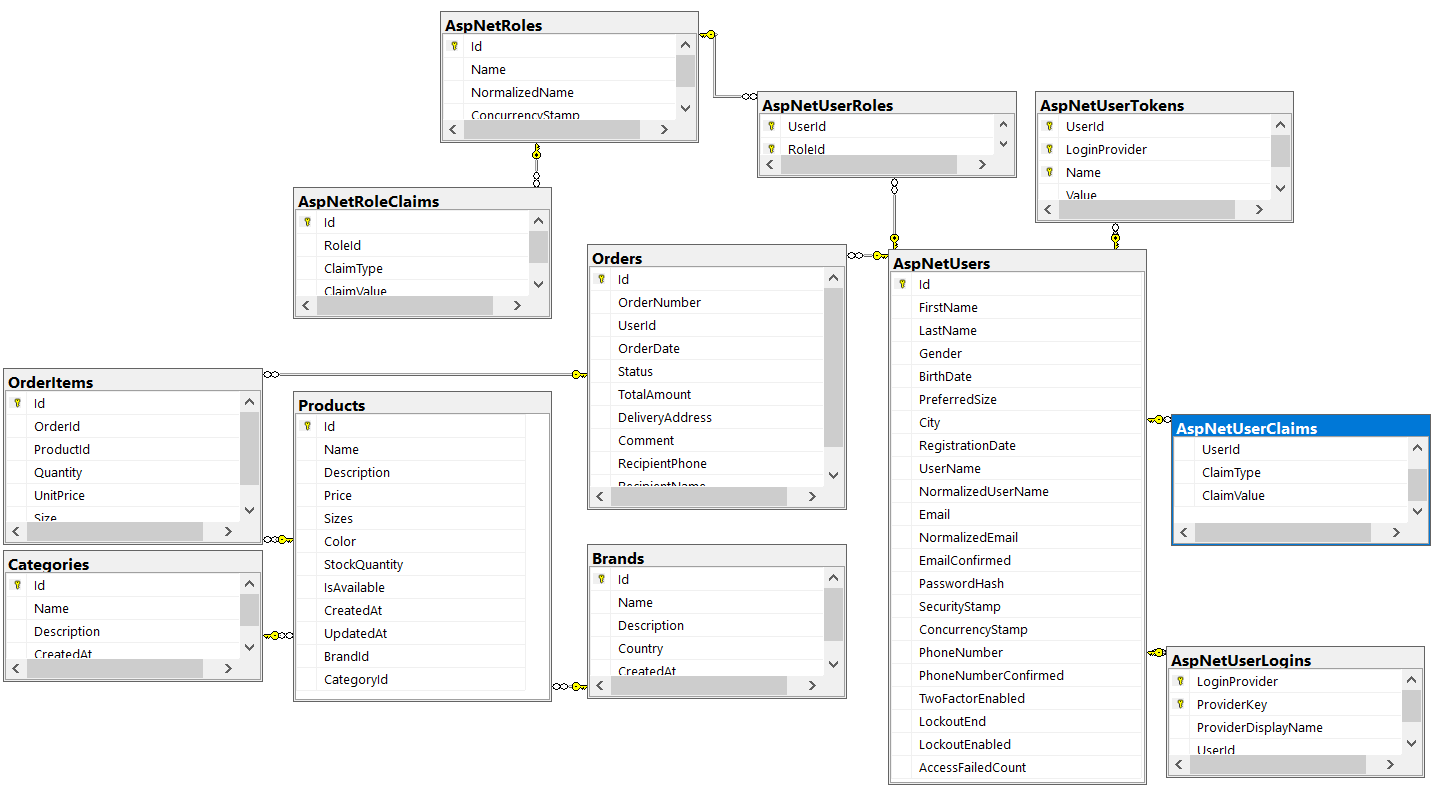


Рисунок 1.22 – Схема данных

Выводы

На этапе анализа предметной области были изучены существующие решения в области интернет-торговли, выявлены функциональные требования к разрабатываемому приложению.

На этапе концептуального проектирования определены действующие лица системы (Администратор, Покупатель) и их функции, составлена диаграмма вариантов использования.

На этапе логического проектирования выделены основные сущности (Бренд, Категория, Товар, Пользователь, Заказ, ЕдиницаЗаказа), построены ER-диаграммы, определены связи между сущностями и сформирован набор отношений для хранения данных.

На этапе физического проектирования определена структура таблиц базы данных, выбрана СУБД Microsoft SQL Server.

# **ГЛАВА 2. Описание функционирования** **веб-приложения**

## 2.1. Назначение и перечень функций веб-приложения

В процессе выполнения курсового проекта было создано веб-приложение онлайн-магазина по продаже брендовой одежды и разработан его интерфейс для удобства в работе пользователей с данными о товарах, заказах, брендах, категориях и пользователях.

При реализации функционала данного приложения использованы следующие технологии:

* СУБД Microsoft SQL Server для создания базы данных;
* ASP.NET Core Blazor Server для создания веб-приложения;
* Entity Framework Core для работы с базой данных;
* ASP.NET Core Identity для системы авторизации и аутентификации;
* EPPlus и ClosedXML для экспорта и импорта данных в Excel.

ASP.NET Core Blazor Server – это фреймворк для создания интерактивных веб-приложений с использованием C#. Blazor Server позволяет создавать веб-интерфейсы с использованием Razor компонентов, которые выполняются на сервере, а взаимодействие с пользователем происходит через SignalR соединение. Функции веб приложения:

1. **Каталог товаров с фильтрацией** – просмотр списка товаров с возможностью фильтрации по брендам, категориям и поиску по названию.
2. **Детальный просмотр товара** – просмотр подробной информации о товаре, включая описание, цену, размеры, цвета, наличие на складе.
3. **Корзина покупок** – добавление товаров в корзину, изменение количества товаров, удаление товаров из корзины.
4. **Оформление заказов** – создание заказов на основе товаров из корзины с указанием адреса доставки, имени и телефона получателя.
5. **Система авторизации и регистрации** – регистрация новых пользователей с дополнительными полями (пол, дата рождения, предпочитаемый размер, город, телефон), авторизация существующих пользователей.
6. **Система ролей и разграничение прав доступа** – три роли пользователей:
   * Администратор – полный доступ ко всем функциям системы;
   * Менеджер – управление заказами, просмотр товаров;
   * Покупатель – просмотр каталога, оформление заказов, управление личным кабинетом.
7. **Административная панель для управления товарами** – CRUD-операции (создание, чтение, обновление, удаление) для товаров.
8. **Административная панель для управления брендами** – CRUD-операции для брендов.
9. **Административная панель для управления категориями** – CRUD-операции для категорий.
10. **Административная панель для управления заказами** – просмотр всех заказов, изменение статусов заказов, просмотр детальной информации о заказах.
11. **Экспорт и импорт данных в Excel** – экспорт данных о товарах, брендах и категориях в файлы Excel, импорт данных из файлов Excel.
12. **Личный кабинет пользователя** – просмотр истории заказов, детальной информации о заказах, управление профилем пользователя.
13. **Главная страница** – отображение популярных товаров, информации о магазине.
14. **Инициализация данных при первом запуске** – автоматическое создание ролей, администратора по умолчанию, начальных данных о брендах, категориях и товарах.

## 2.2. Программная реализация веб-приложения

Работа над программой начинается с создания проекта ASP.NET Core Blazor Server и установки необходимых NuGet пакетов. (рис. 2.1).

* Microsoft.EntityFrameworkCore.SqlServer – для работы с SQL Server;
* Microsoft.AspNetCore.Identity.EntityFrameworkCore – для системы Identity;
* EPPlus – для работы с Excel файлами;
* ClosedXML – для работы с Excel файлами.

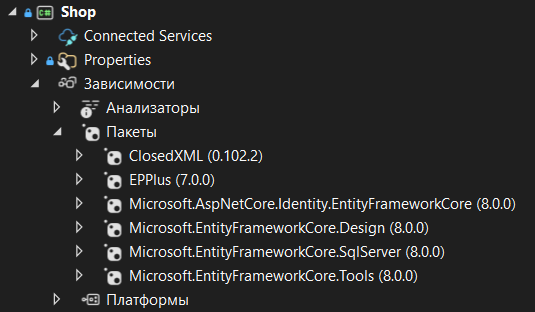


Рисунок 2.1 − Пакеты поставщика

В папке Models создаём необходимые классы, содержащие свойства для хранения данных. Рассмотрим модель Product. (рис. 2.2-2.3).



Рисунок 2.2 − Фрагмент экранной копии, показывающий модель Product

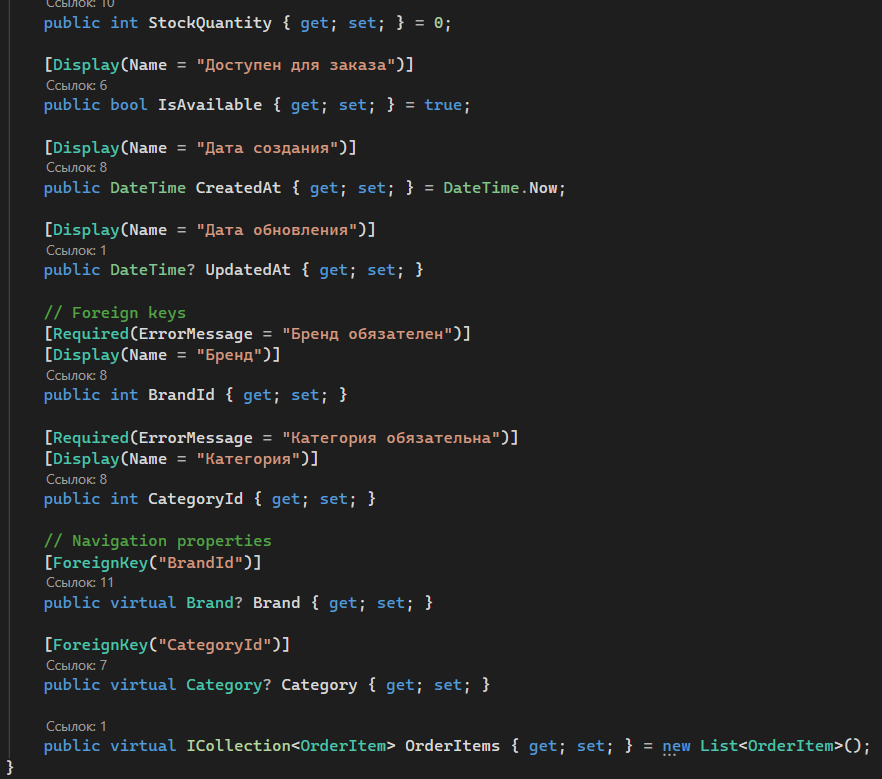


Рисунок 2.3 − Фрагмент экранной копии, показывающий модель Product

Аналогичным образом создаём оставшиеся модели данных: Brand, Category, Order, OrderItem, ApplicationUser. (рис. 2.4 ).

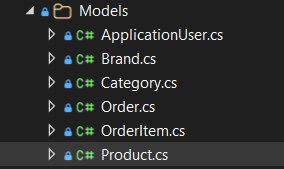


Рисунок 2.4 − Фрагмент экранной копии, показывающий модели данных

После создания моделей нужно обратиться к контексту базы данных. Контекст базы данных DbContext – это основной класс, который координирует функциональные возможности EF для определённой модели данных. Этот класс является производным от класса Microsoft.EntityFrameworkCore.DbContext. Производный класс DbContext указывает сущности, которые включаются в модель данных. В проекте соответствующий класс называется ApplicationDbContext. Данный класс создаётся в папке Data и содержит следующий код (рис. 2.5).

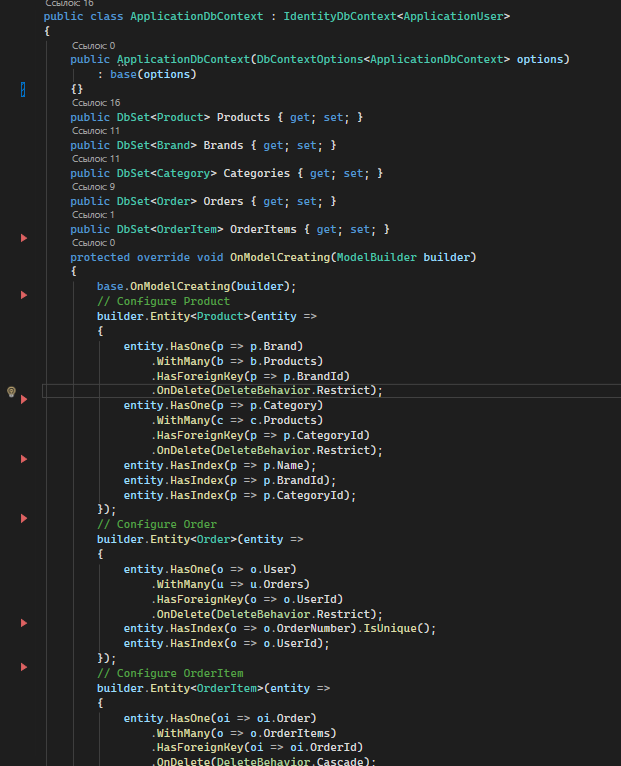
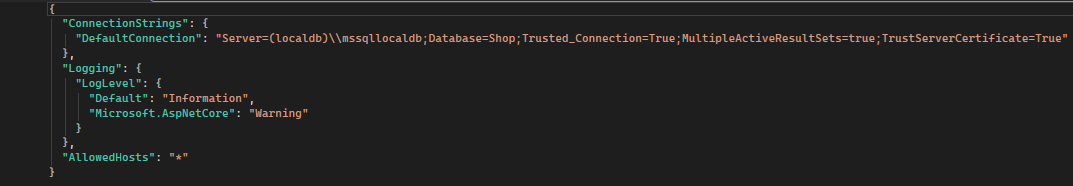
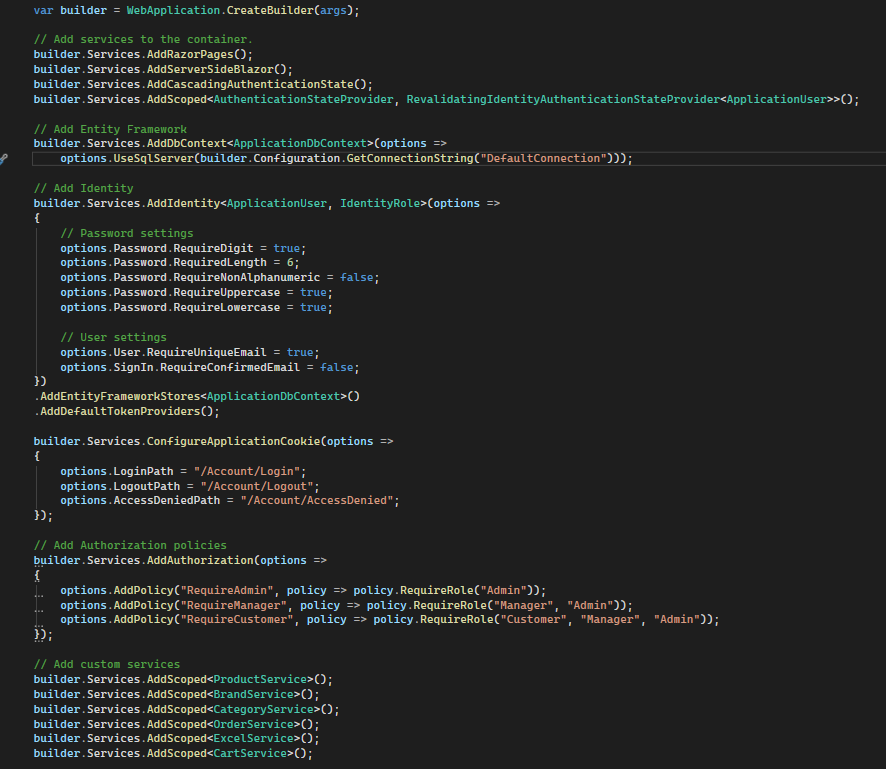


Рисунок 2.5 – Фрагмент экранной копии, класс ApplicationDbContext

Далее необходимо выполнить миграцию и обновить базу данных. Для этого в файле appsettings.json прописываем строку подключения к базе данных. (рис. 2.6)

 Рисунок 2.6 – Файл настроек подключения к базе данных

Для создания базы данных используется метод EnsureCreated(), который автоматически создаёт базу данных и таблицы при первом запуске приложения. Настройка Identity и авторизации. В файле Program.cs настраиваем Identity с использованием пользовательской модели ApplicationUser. Также настраиваем политики авторизации для различных ролей (рис. 2.7)

 Рисунок 2.7 – Фрагмент Identity с использованием модели ApplicationUser и ролей

Создаём сервисы для работы с данными. В папке Services создаём классы ProductService, BrandService, CategoryService, OrderService, ExcelService, CartService. Рассмотрим класс ProductService (рис. 2.8-2.9):

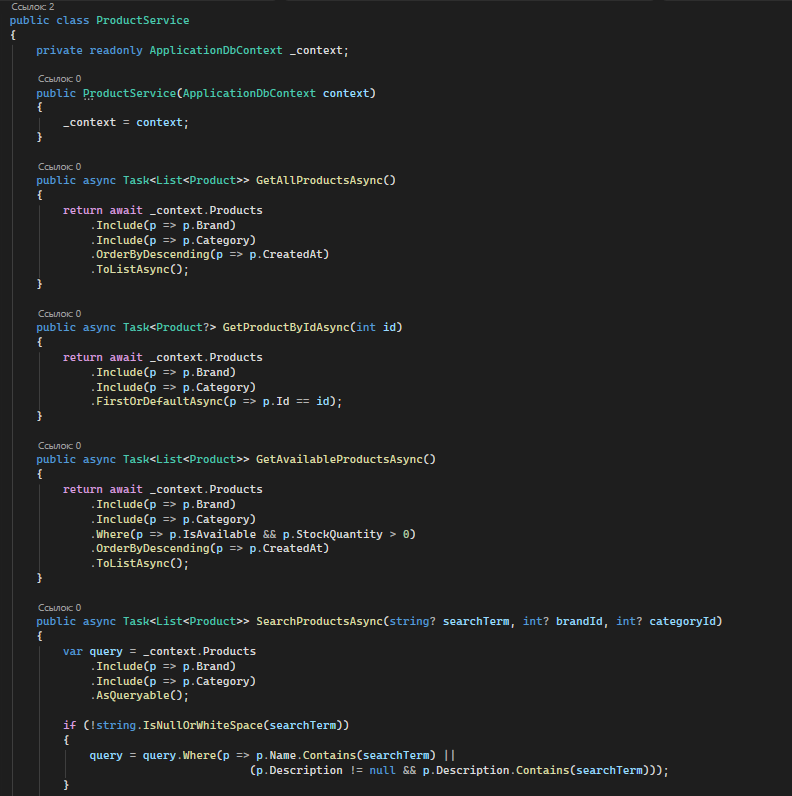
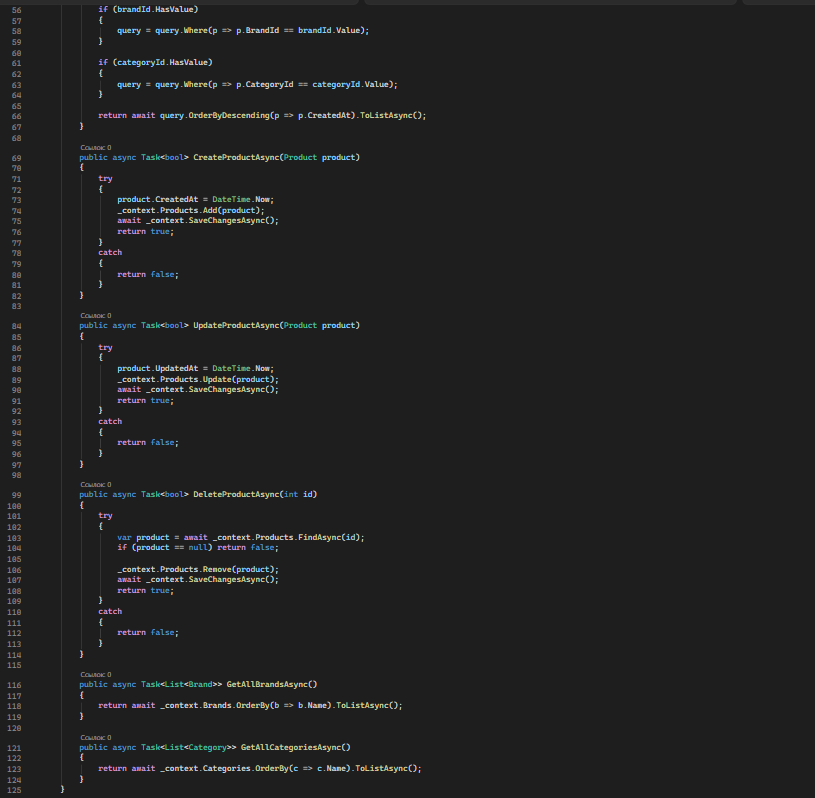
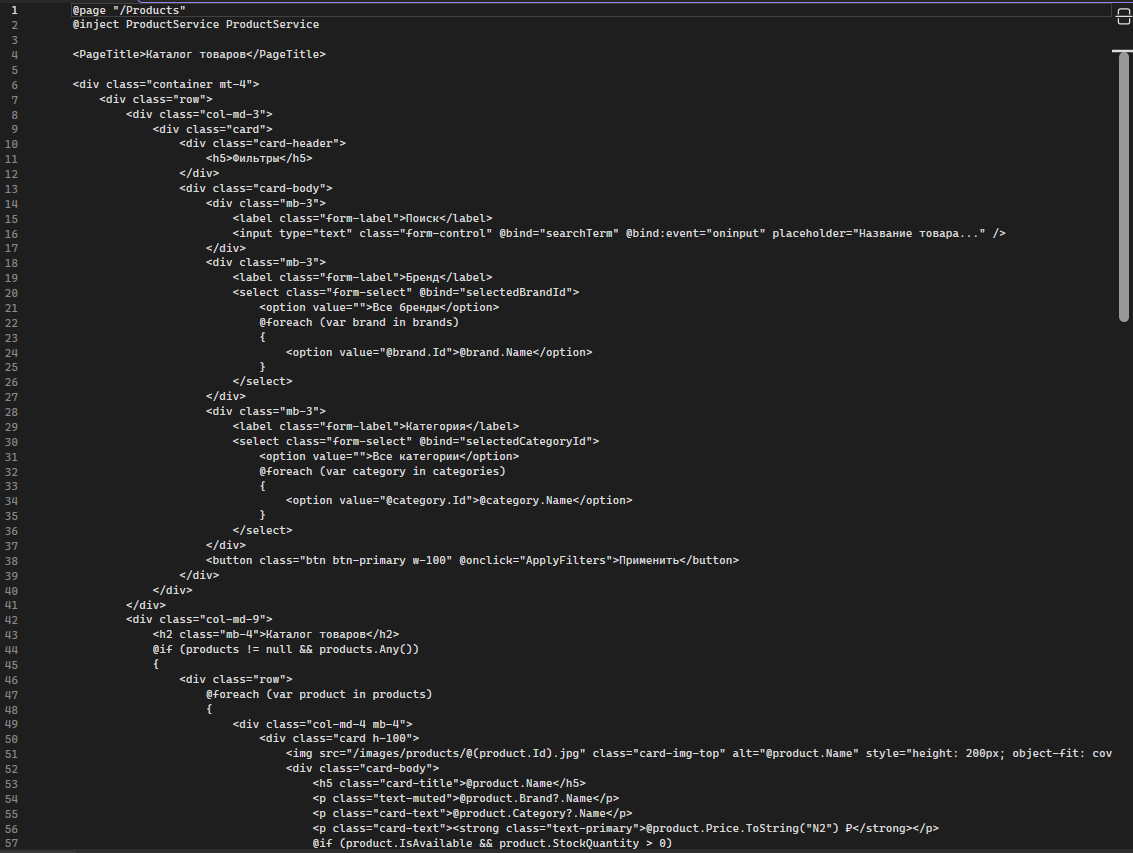
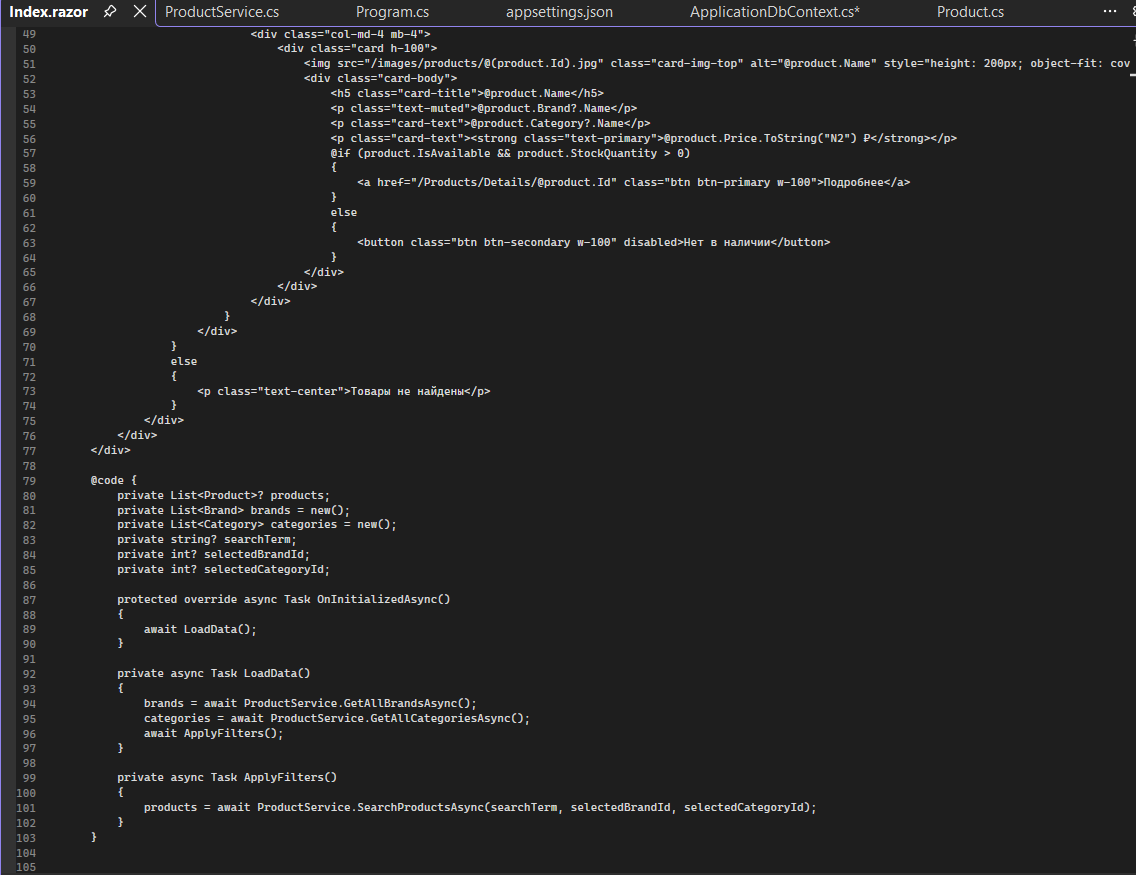


Рисунок 2.8 – Фрагмент класс ProductService

 Рисунок 2.9 – Фрагмент класс ProductService

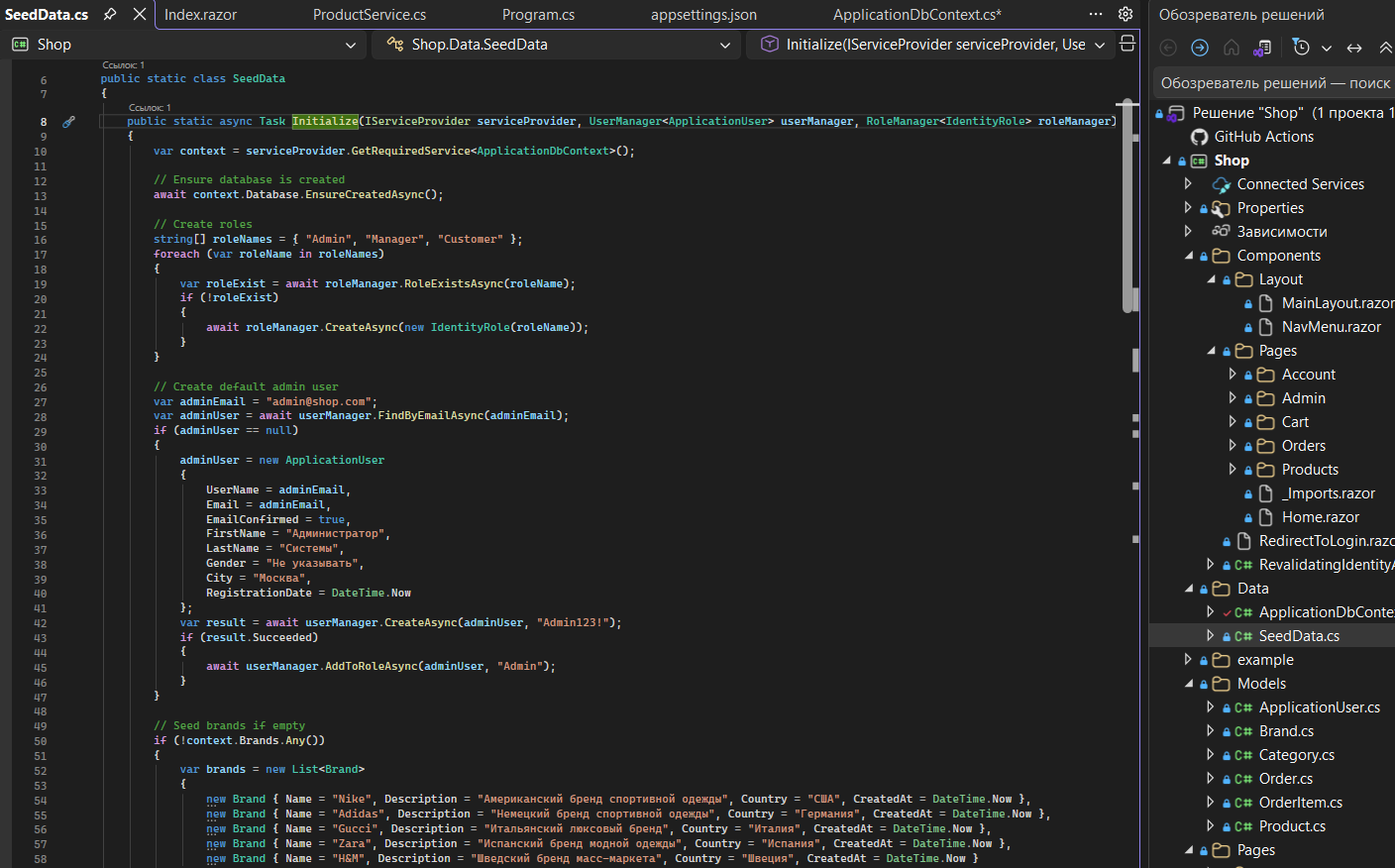
Создаём Blazor компоненты для отображения страниц. В папке Components/Pages создаём компоненты для различных страниц приложения. Рассмотрим компонент каталога товаров Products/Index.razor (рис. 2.10-2.11):

Рисунок 2.10 – Фрагмент каталога товаров

 Рисунок 2.11 – Фрагмент каталога товаров

Реализация инициализации данных при первом запуске приложения. Инициализация пользователей и ролей выполняется в классе SeedData, который содержит метод Initialize. Этот метод вызывается при запуске приложения и используется для добавления начальных данных в базу данных.

Внутри метода Initialize создаются три роли: "Admin", "Manager", "Customer". После создания ролей создаётся администратор по умолчанию (email: admin@shop.com, пароль: Admin123!) и назначается ему роль "Admin". Также добавляются начальные данные о брендах (Nike, Adidas, Gucci, Zara, H&M), категориях (Футболки, Джинсы, Куртки, Обувь, Аксессуары) и товарах. (рис. 2.12):

Рисунок 2.12 – Фрагмент SeedData.Initialize добавления начальных данных в базу данных

Экспорт и импорт данных из файлов Excel. Для реализации экспорта и импорта данных используется библиотека EPPlus. В классе ExcelService реализованы методы ExportProductsToExcelAsync и ImportProductsFromExcelAsync.

Метод ExportProductsToExcelAsync используется для экспорта данных из таблицы Products в файл Excel. Используя библиотеку EPPlus, создаётся новый файл Excel и заполняется данными из базы данных. Затем файл отправляется обратно клиенту в виде потока байтов. (рис. 2.13)

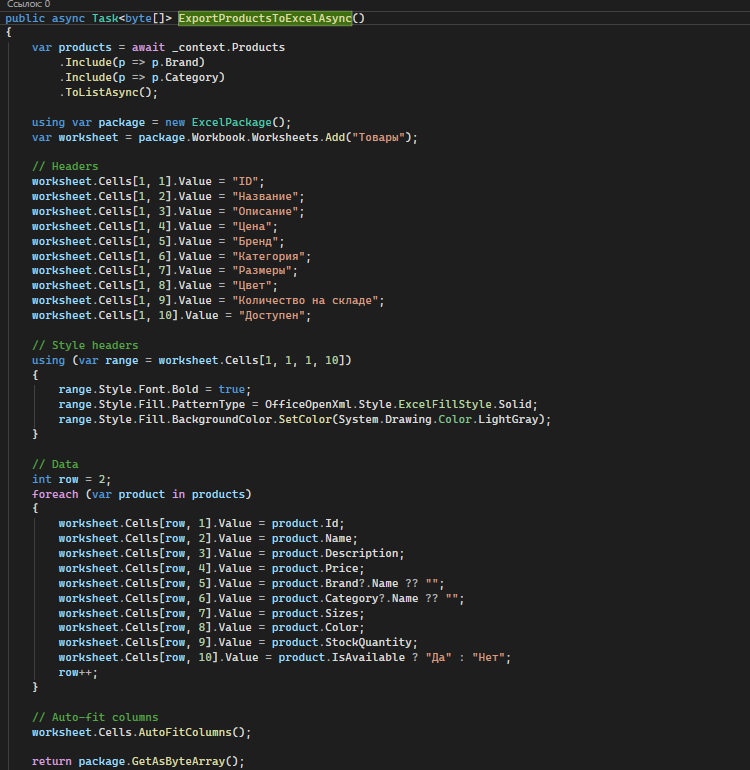
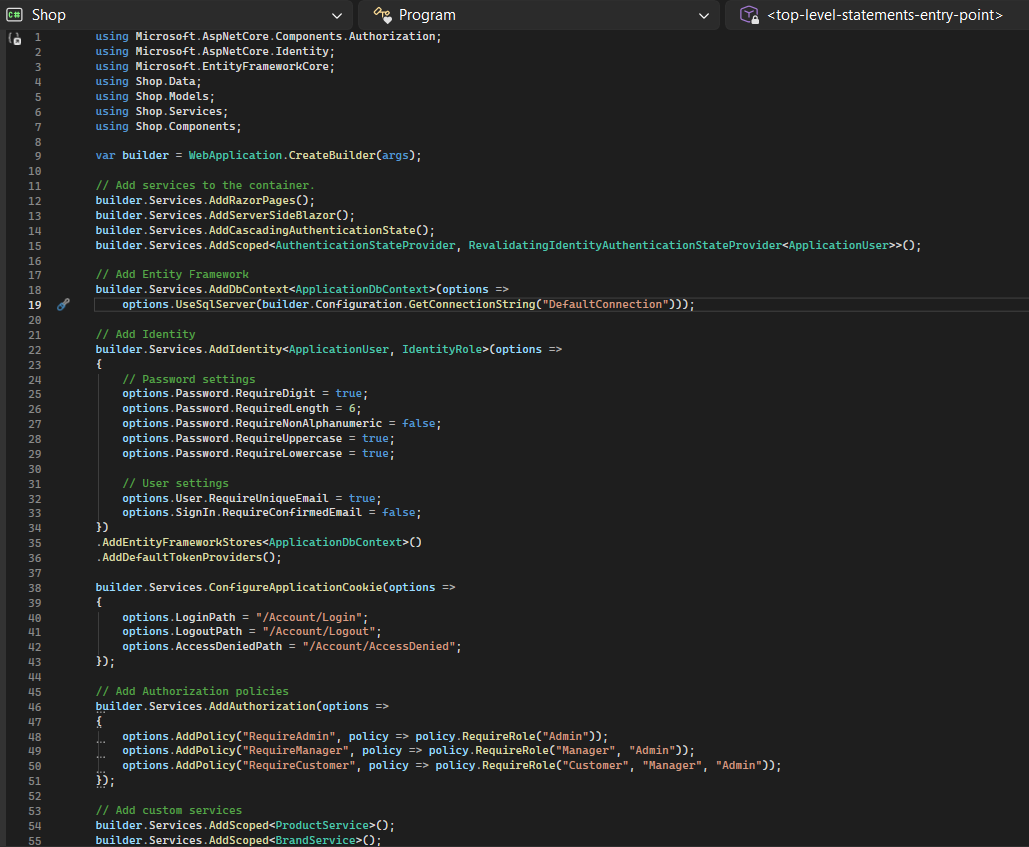
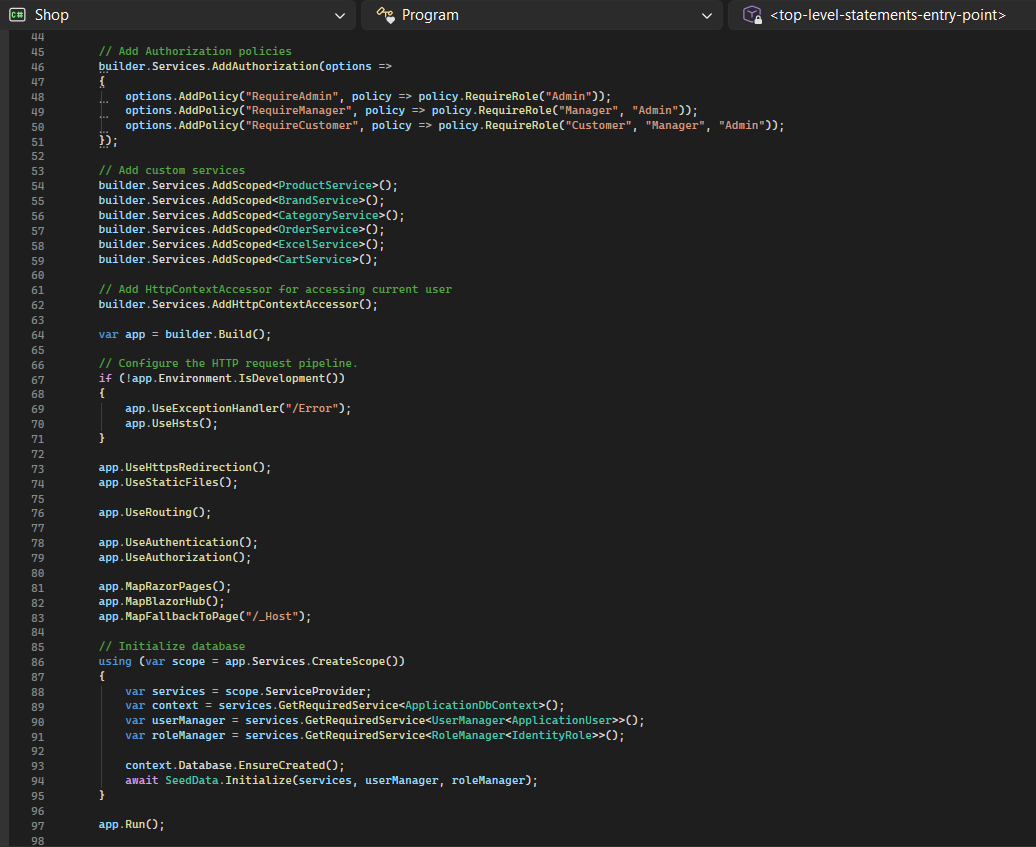


Рисунок 2.12 – Фрагмент ExportProductsToExcelAsync

Файл Program.cs является точкой входа в приложение и содержит настройку и конфигурацию приложения. Создаётся экземпляр WebApplication с использованием WebApplication.CreateBuilder(args). Настраиваются сервисы: Razor Pages, Blazor Server, Entity Framework Core, Identity, авторизация, пользовательские сервисы. Настраивается конвейер обработки HTTP-запросов: обработчики исключений, HTTPS-переадресация, статические файлы, маршрутизация, авторизация. Выполняется инициализация базы данных и начальных данных. (рис. 2.14-2.15)

Рисунок 2.14 – Фрагмент Program.cs является точкой входа с подключением сервисов

Рисунок 2.15 – Фрагмент Program.cs является точкой входа с подключением сервисов

После запуска приложения база данных Shop создаётся автоматически со всеми необходимыми таблицами (рис. 2.16).

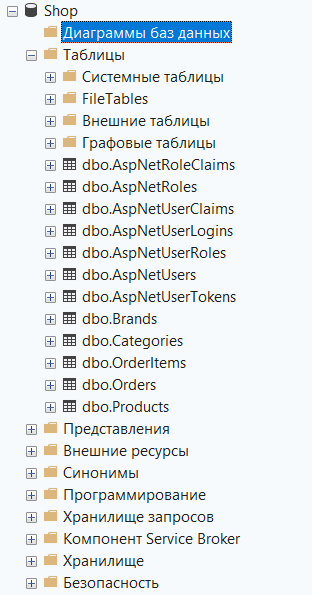


Рисунок 2.16 – Фрагмент экранной копии, показывающий созданную базу данных и её структуру

Проверяем правильность заполнения моделей с помощью диаграммы базы данных. Для этого в SQL Server Management Studio переходим в раздел "Диаграммы баз данных" и создаём диаграмму, которая отображает все таблицы и связи между ними (рис. 2.17).

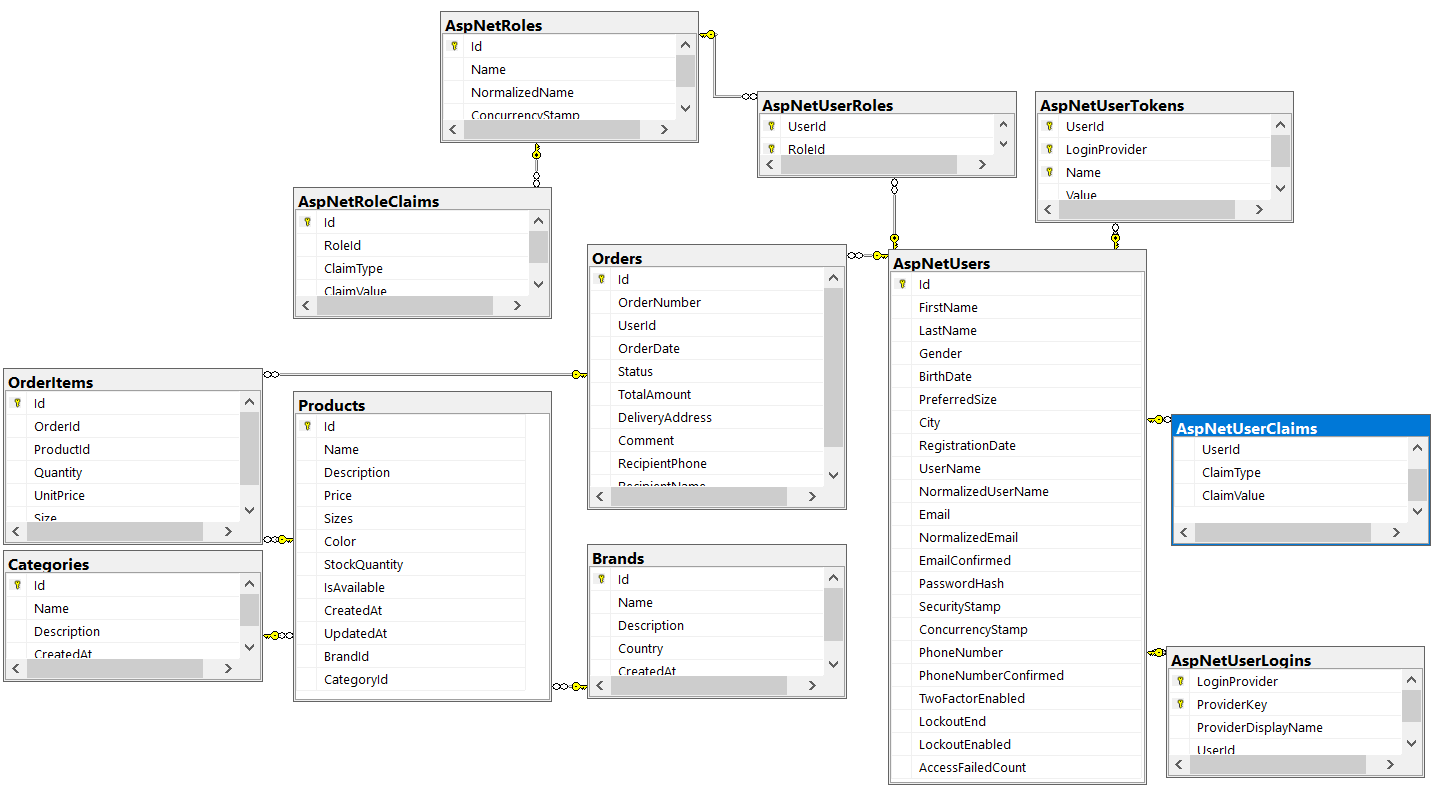


Рисунок 2.17 – Фрагмент экранной копии, диаграмма базы данных

Чтобы созданные элементы были видны, нужно организовать переходы между страницами. В папке Pages изменяем код \_Layout.cshtml для настройки навигации и структуры страниц (рис. 2.18).



Рисунок 2.18 – Фрагмент экранной копии, демонстрирующий код \_Layout.cshtml

Дополнительные поля при регистрации: Добавление дополнительных полей при регистрации пользователя, такие как имя, фамилия, пол, дата рождения, предпочитаемый размер, город, телефон. Это достигается путем обновления модели ApplicationUser и представления регистрации Register.cshtml (рис. 2.19).

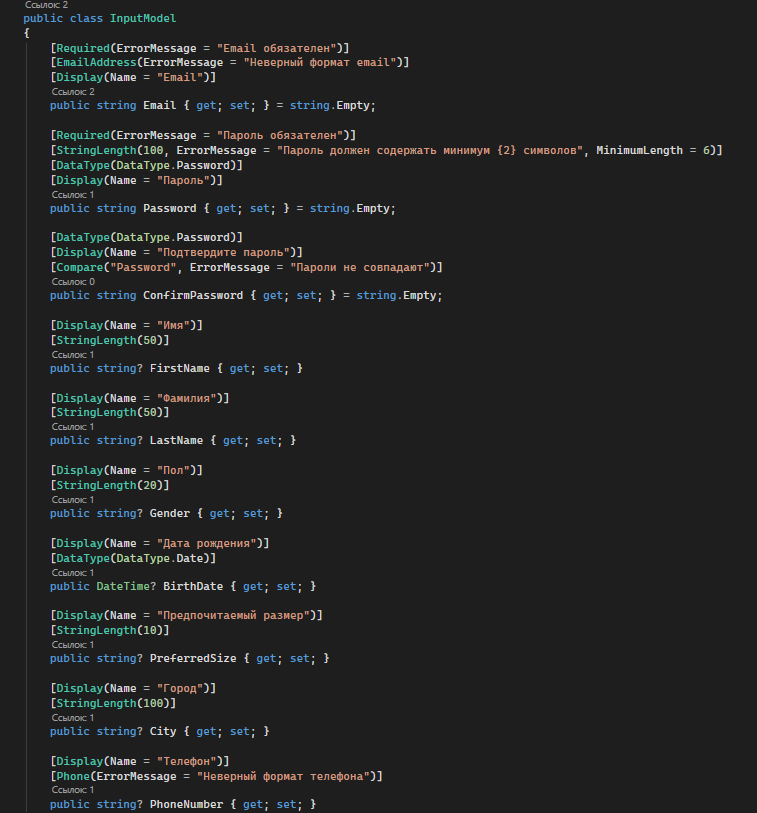


Рисунок 2.19 – Фрагмент экранной копии, добавление новых полей регистрации

Роли и политики (Authorization): Система ролей и политик, которая позволяет управлять доступом пользователей к различным частям приложения. Это достигается путем использования атрибута [Authorize] и его вариаций для ограничения доступа к определенным страницам и компонентам (рис. 2.20).

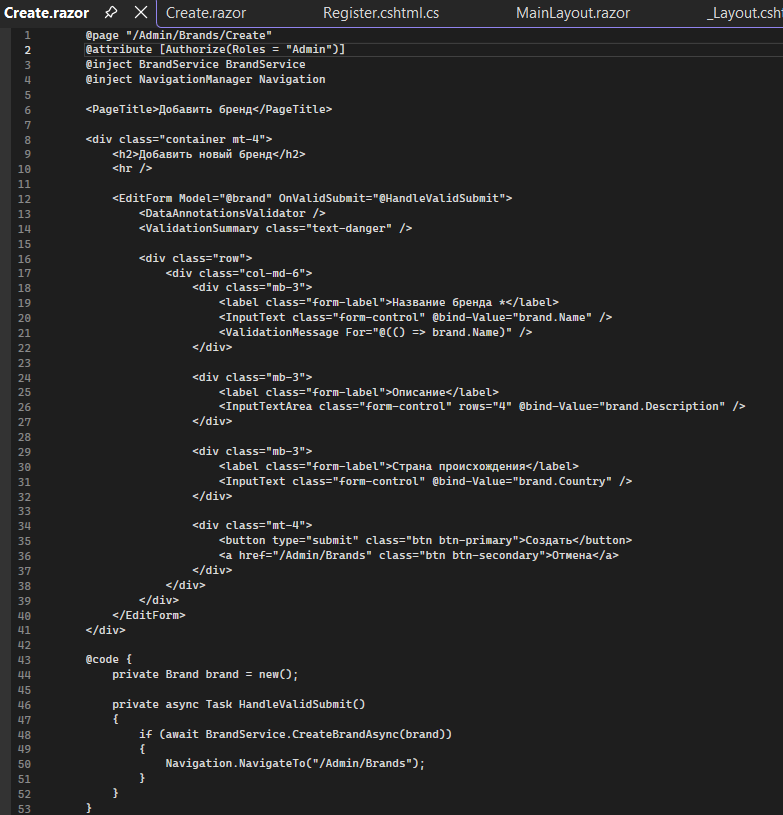


Рисунок 2.20 – Фрагмент экранной копии, использование атрибута “[Authorize]”

Разграничение прав доступа: система разграничения прав доступа, которая позволяет определить, какие действия может выполнять каждый пользователь в зависимости от его роли. Это достигается путем создания пользовательских политик авторизации, которые проверяют роль пользователя перед выполнением действия (рис. 2.21).

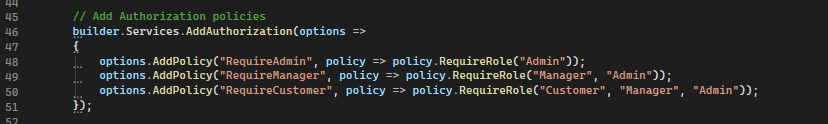


Рисунок 2.21 – Фрагмент экранной копии, добавление политики

## 2.2. Демонстрация веб-приложения

Проверка доступов неавторизованных пользователей (рис. 2.22 – 2.26).

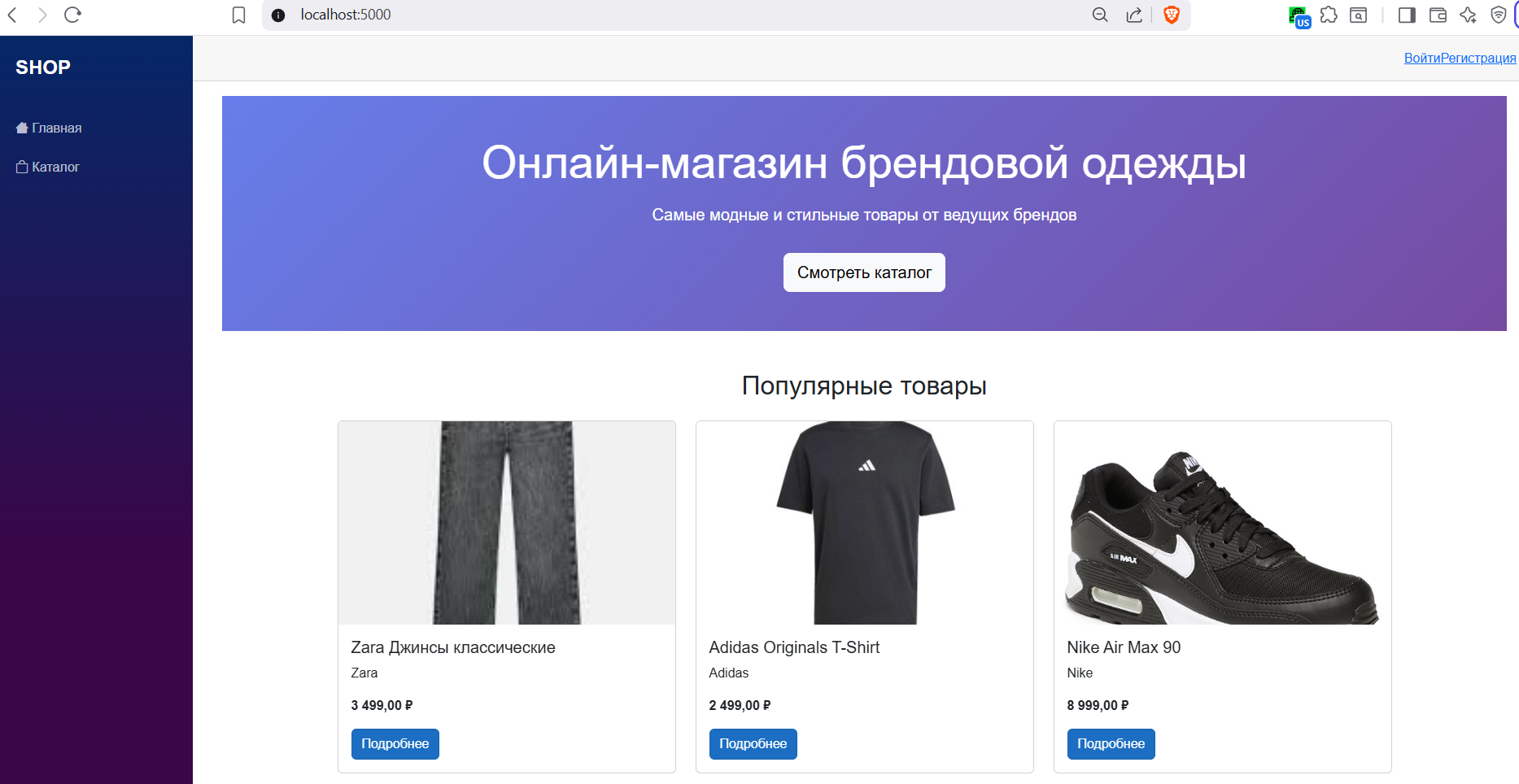


Рисунок 2.22 – Фрагмент экранной копии, Главная

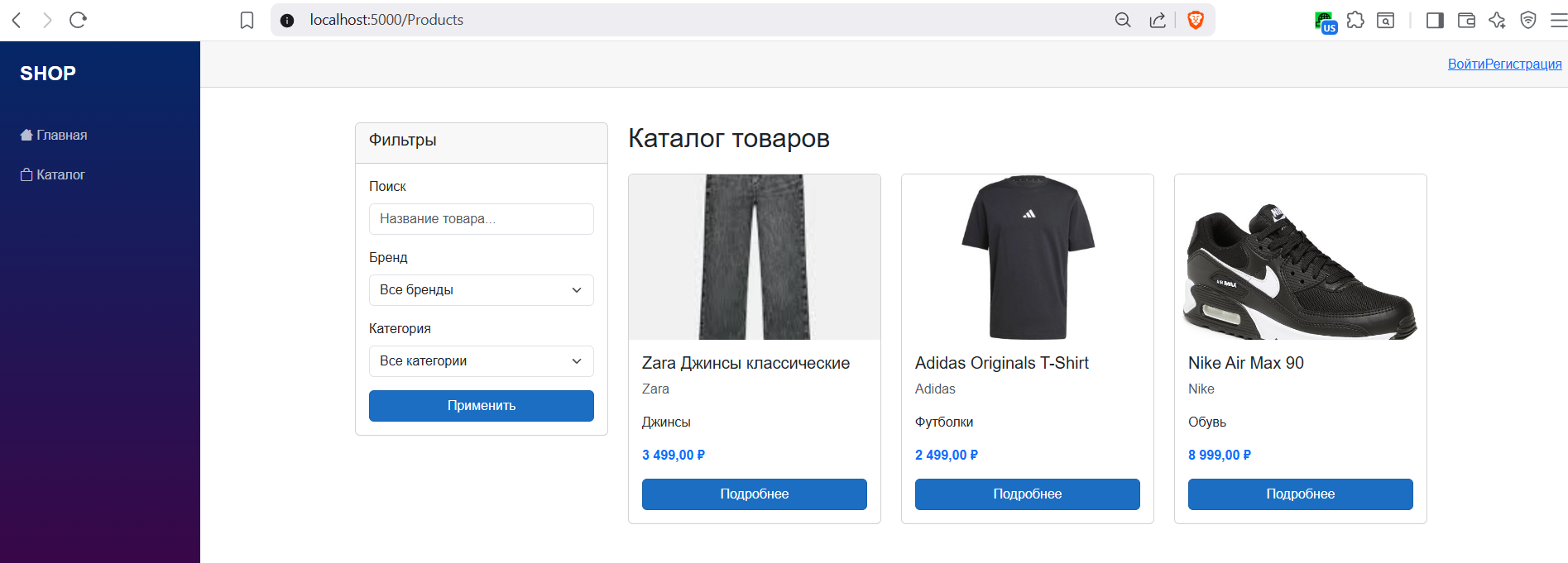


Рисунок 2.23 – Фрагмент экранной копии, Каталог

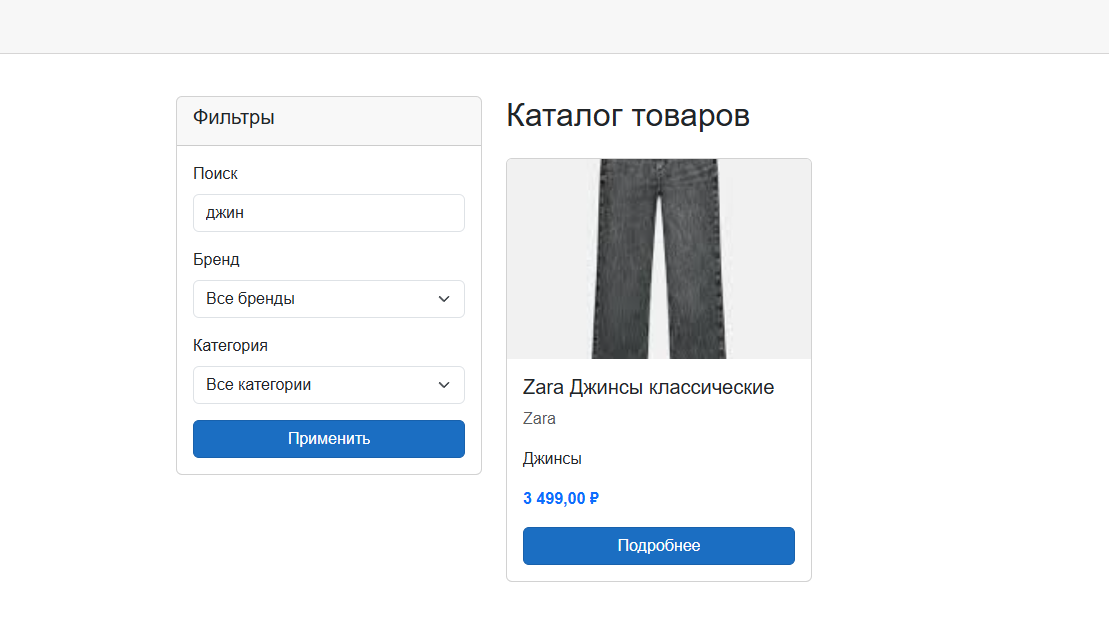


Рисунок 2.24 – Фрагмент экранной копии, Каталог, фильтр – Поиск

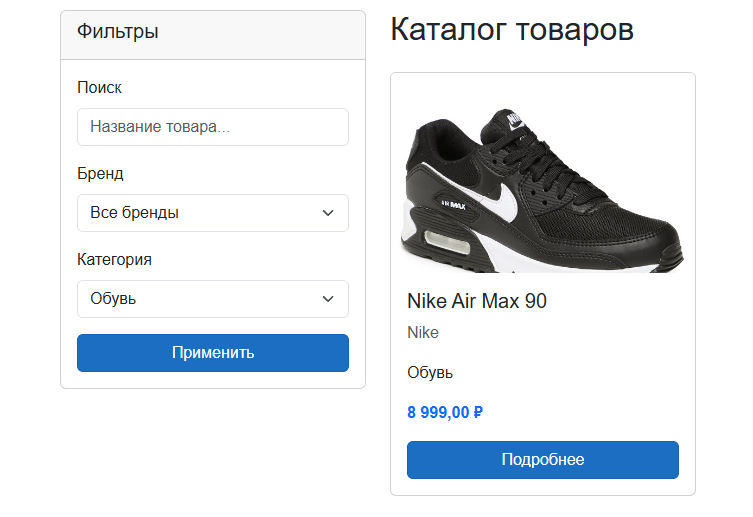


Рисунок 2.25 – Фрагмент экранной копии, Каталог, фильтр - Категория

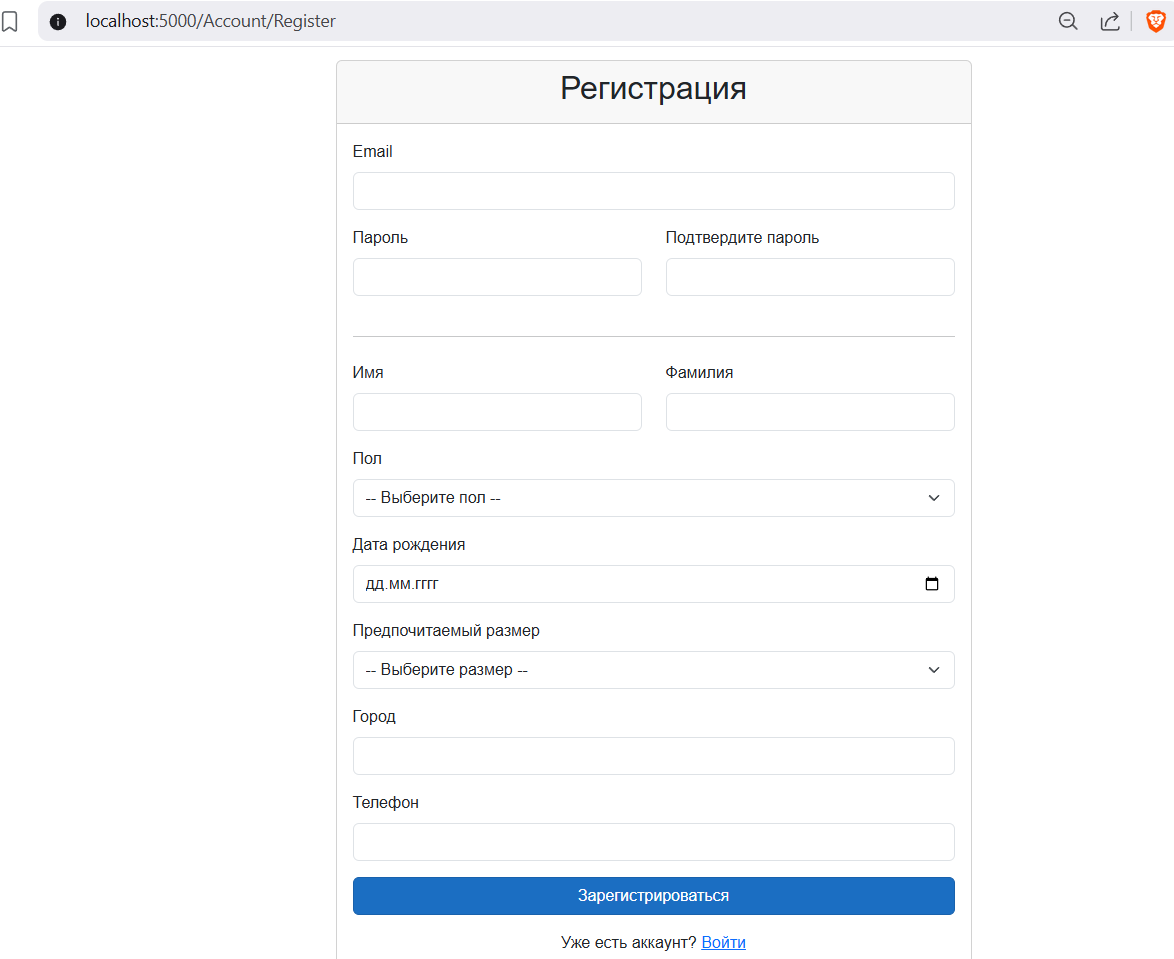


Рисунок 2.26 – Фрагмент экранной копии, Регистрация

Проверка доступов авторизованных пользователей (рис. 2.23 – 2.33).

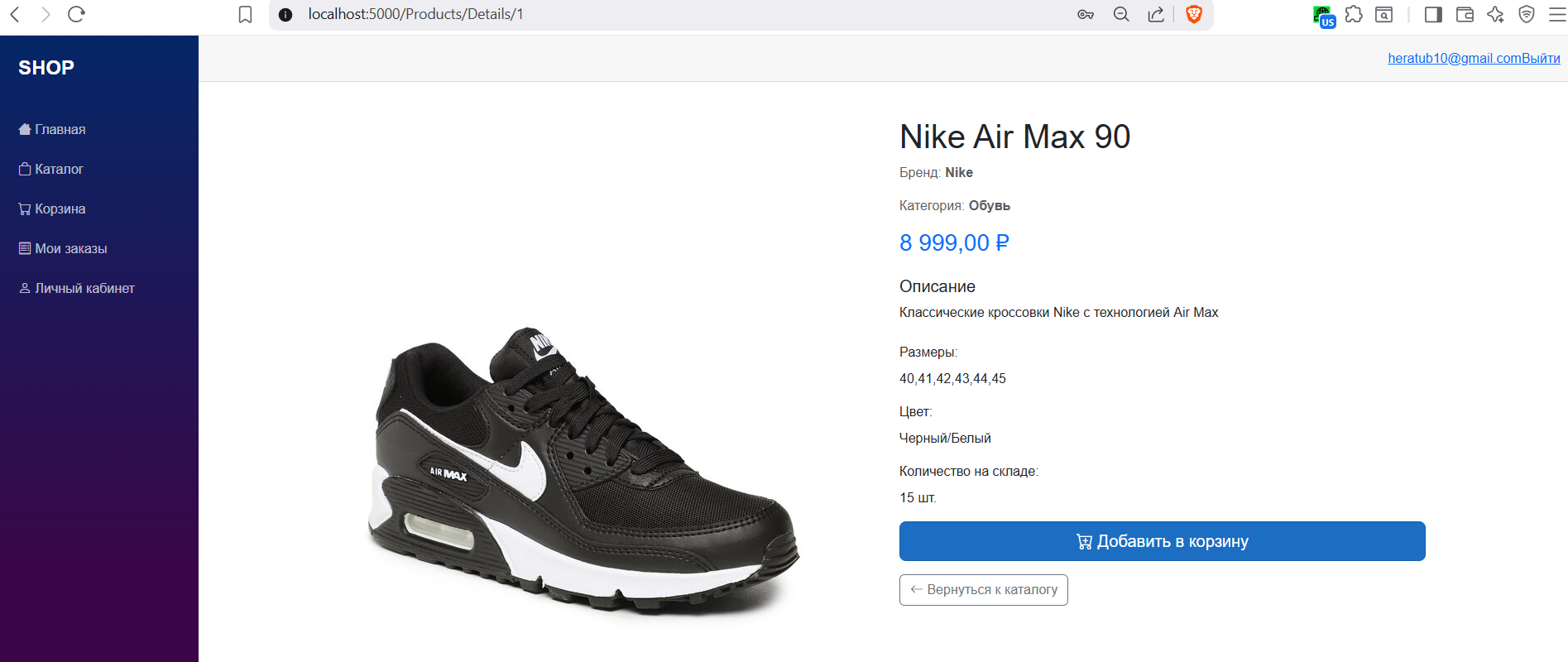


Рисунок 2.27 – Фрагмент экранной копии, Товар

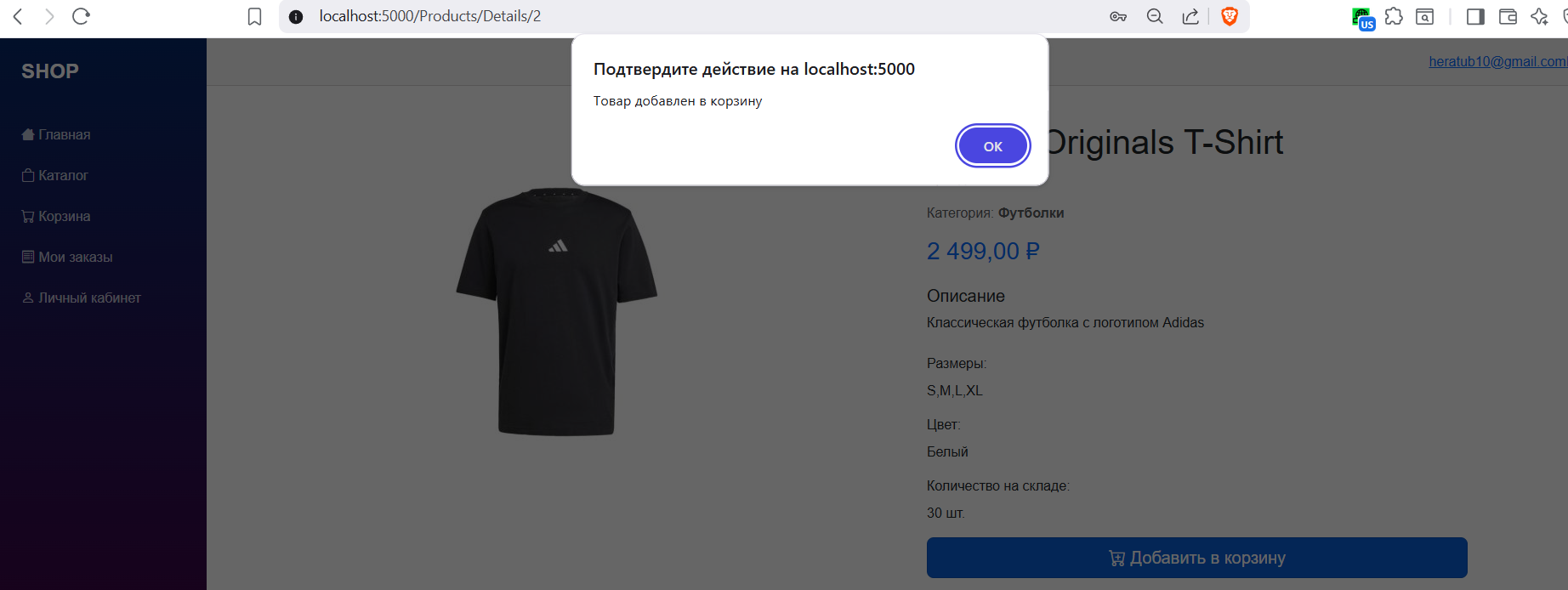


Рисунок 2.28 – Фрагмент экранной копии, Товар добавлен в карзину

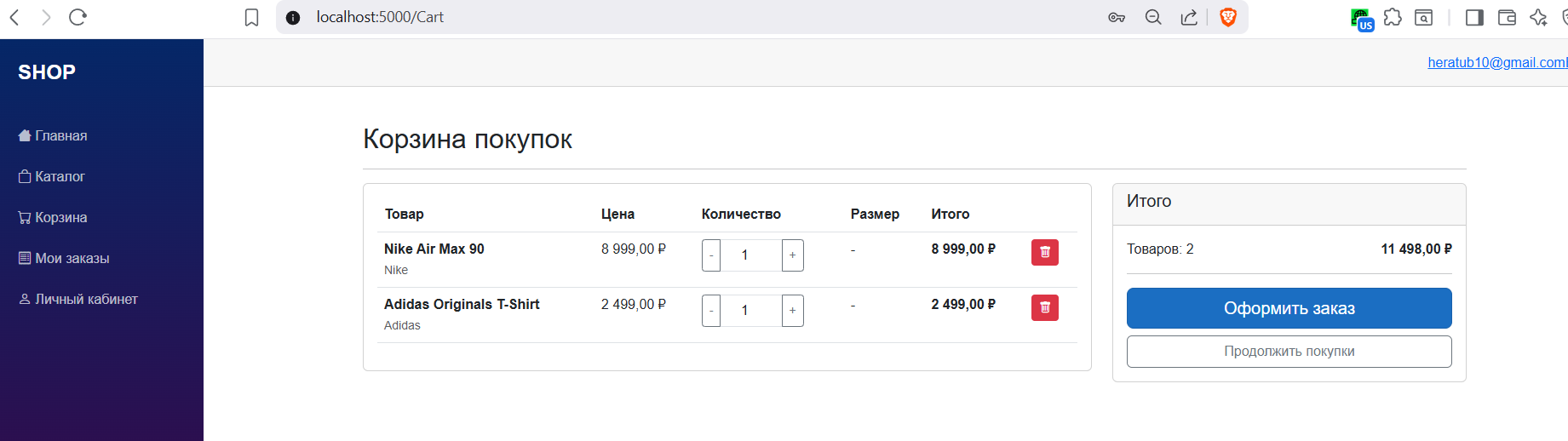


Рисунок 2.29 – Фрагмент экранной копии, Корзина

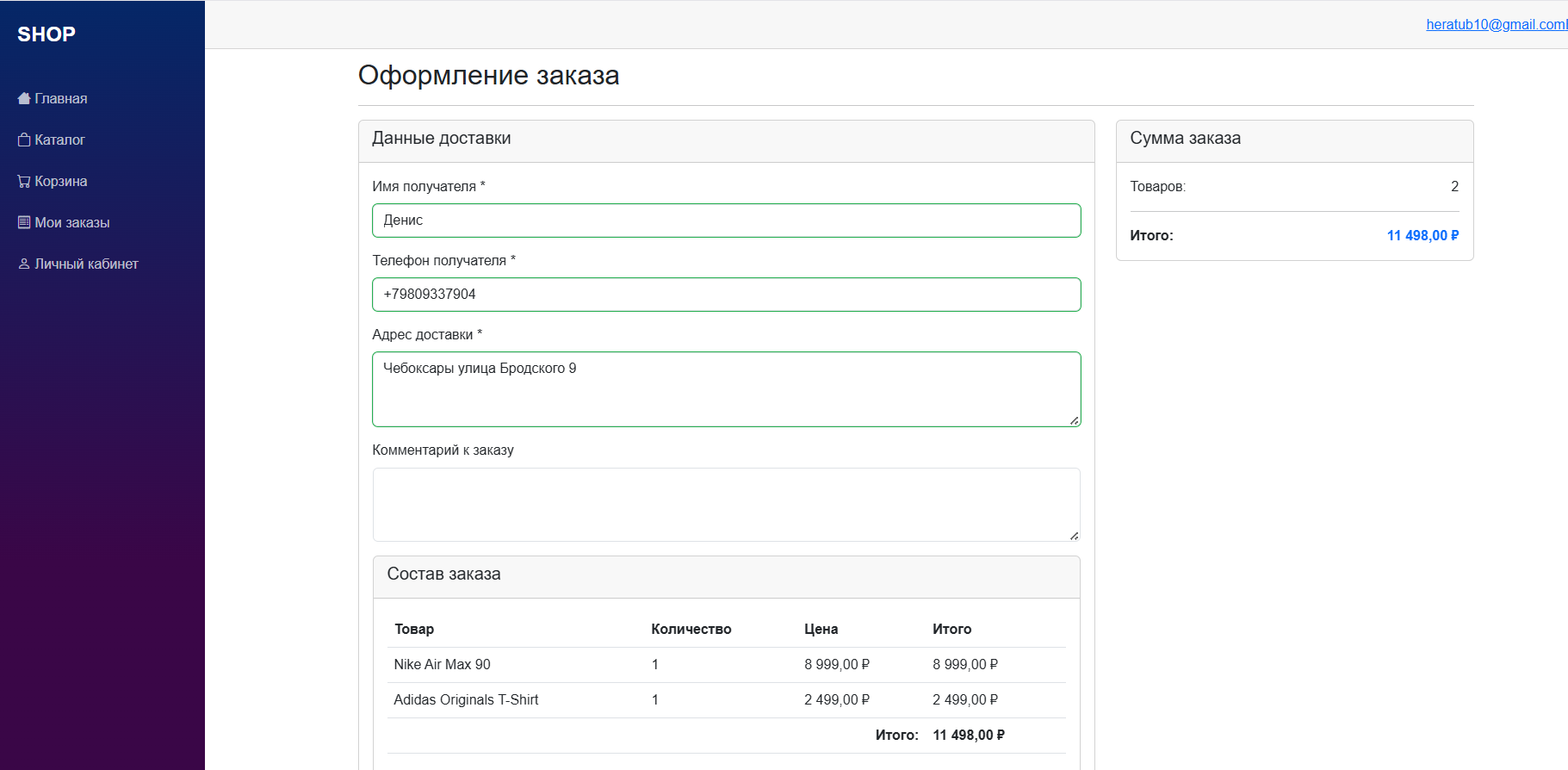


Рисунок 2.30 – Фрагмент экранной копии, Оформление заказа

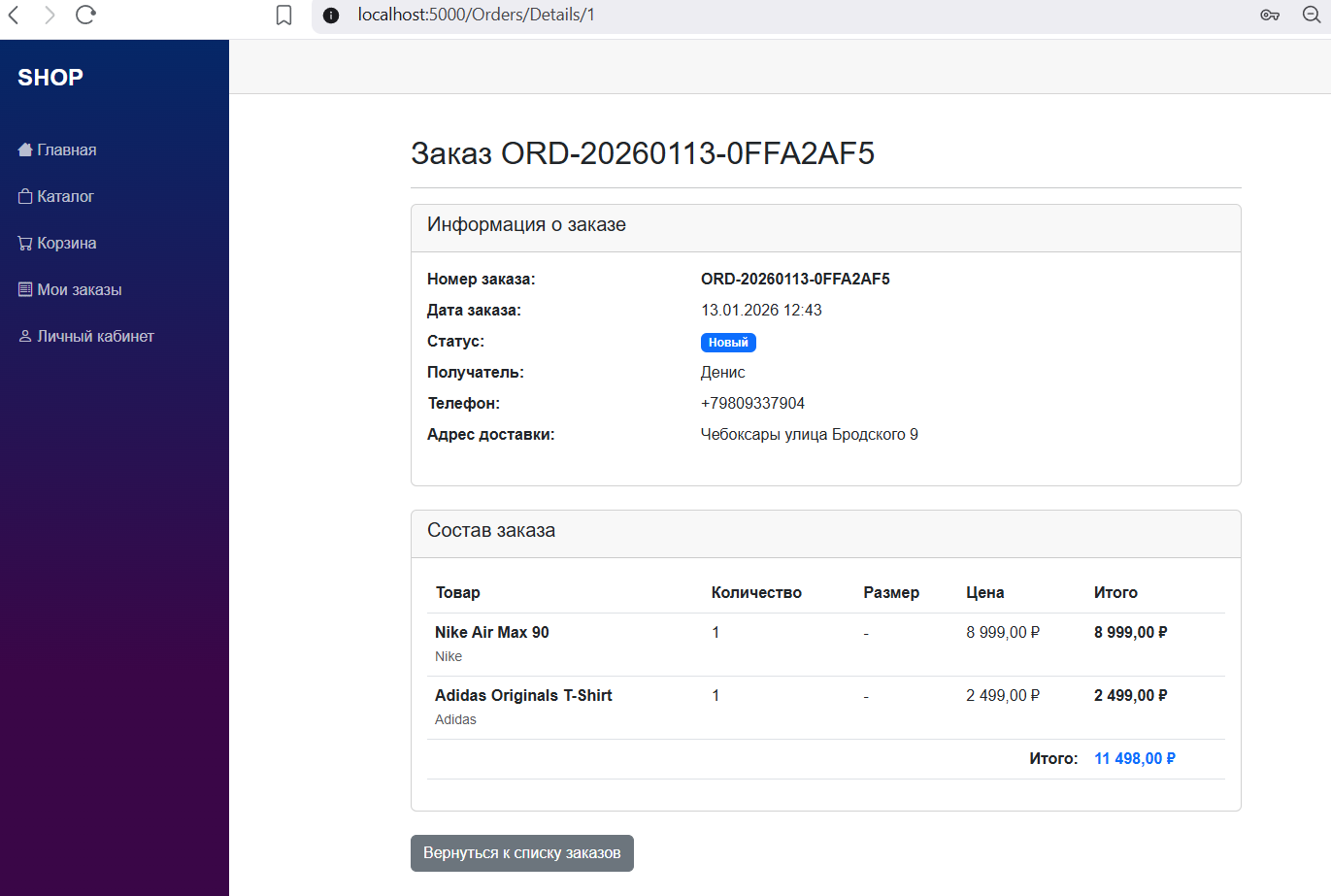


Рисунок 2.31 – Фрагмент экранной копии, Заказ

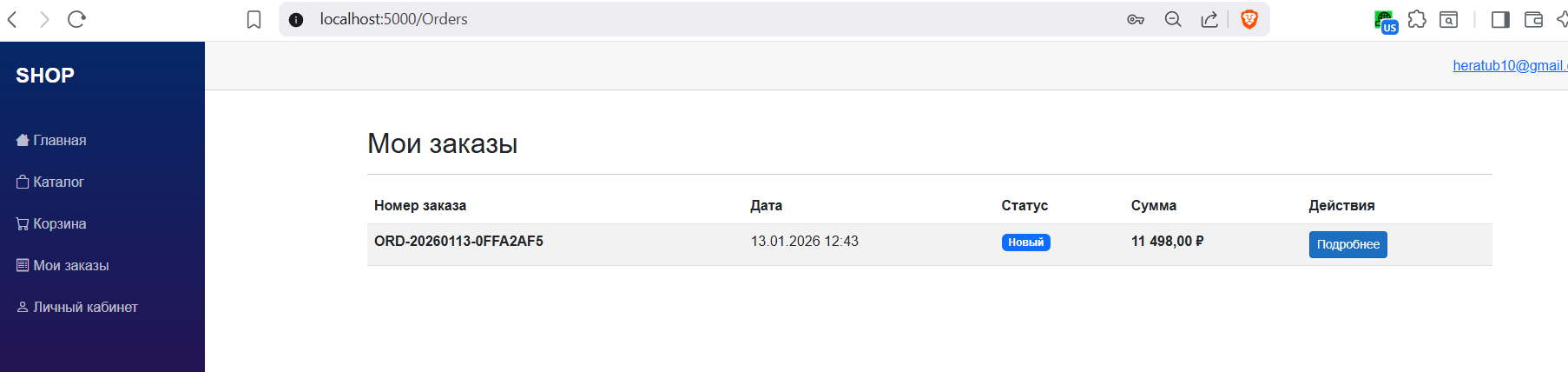


Рисунок 2.32 – Фрагмент экранной копии, Мои заказы

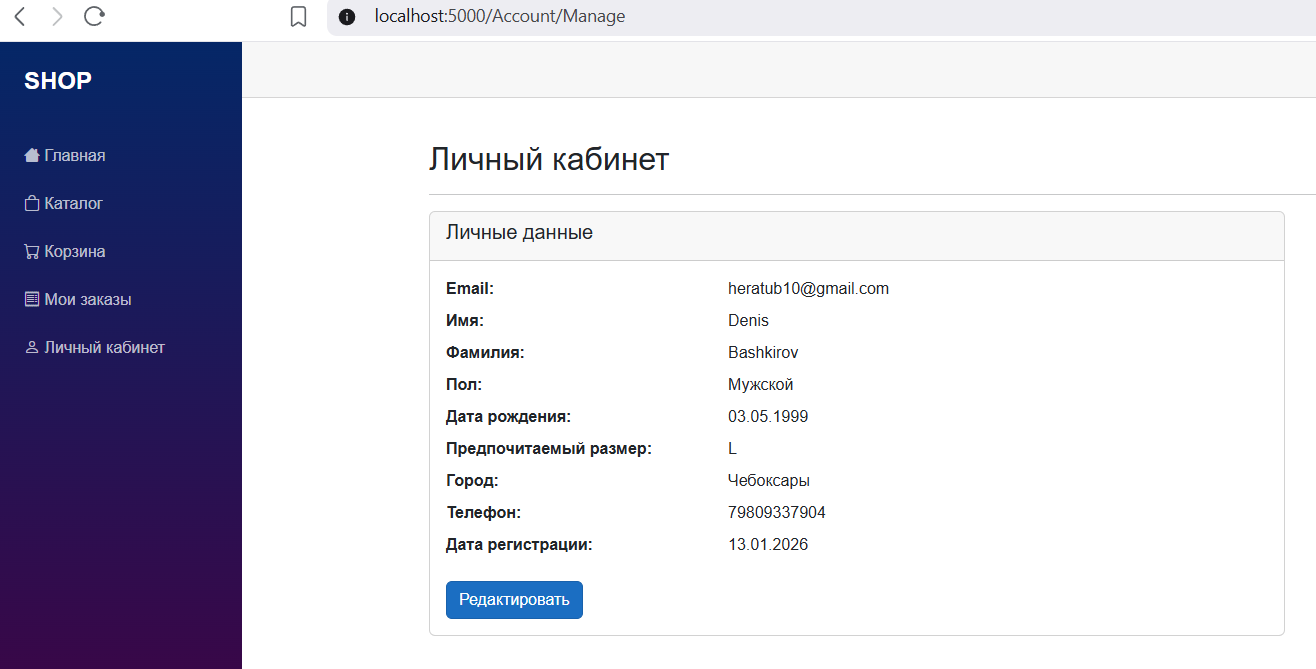


Рисунок 2.33 – Фрагмент экранной копии, Личный кабинет

Проверка доступов администратора(рис. 2.34 – 2.26).

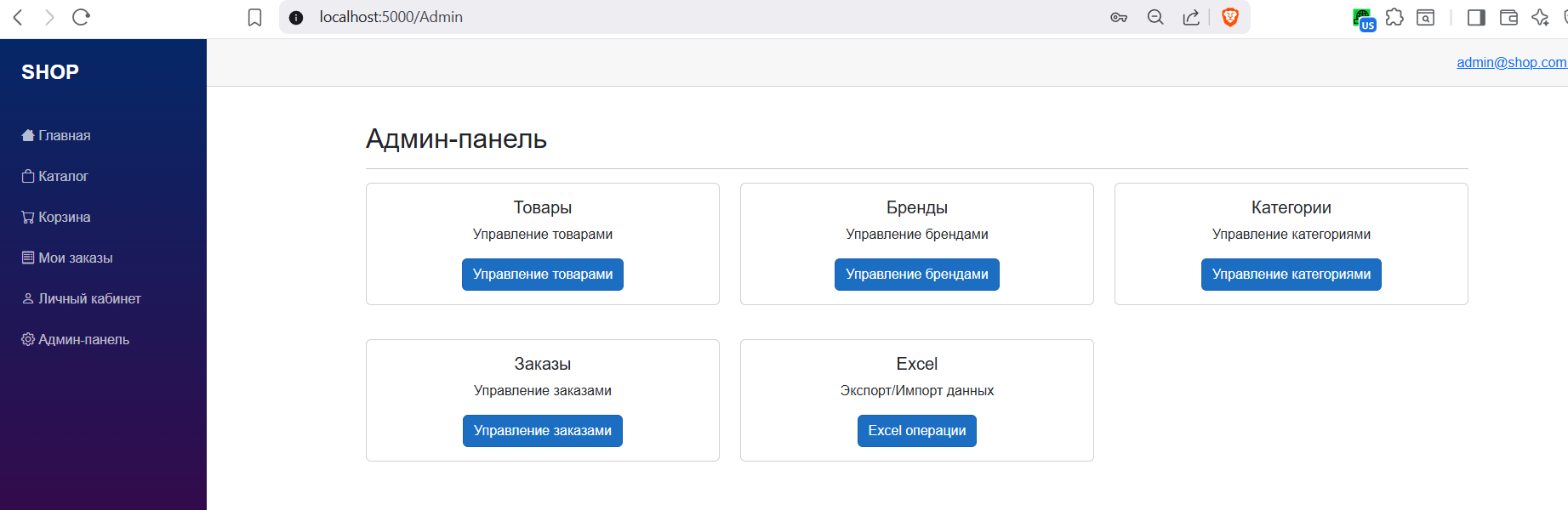


Рисунок 2.40 – Фрагмент экранной копии, Админ панель

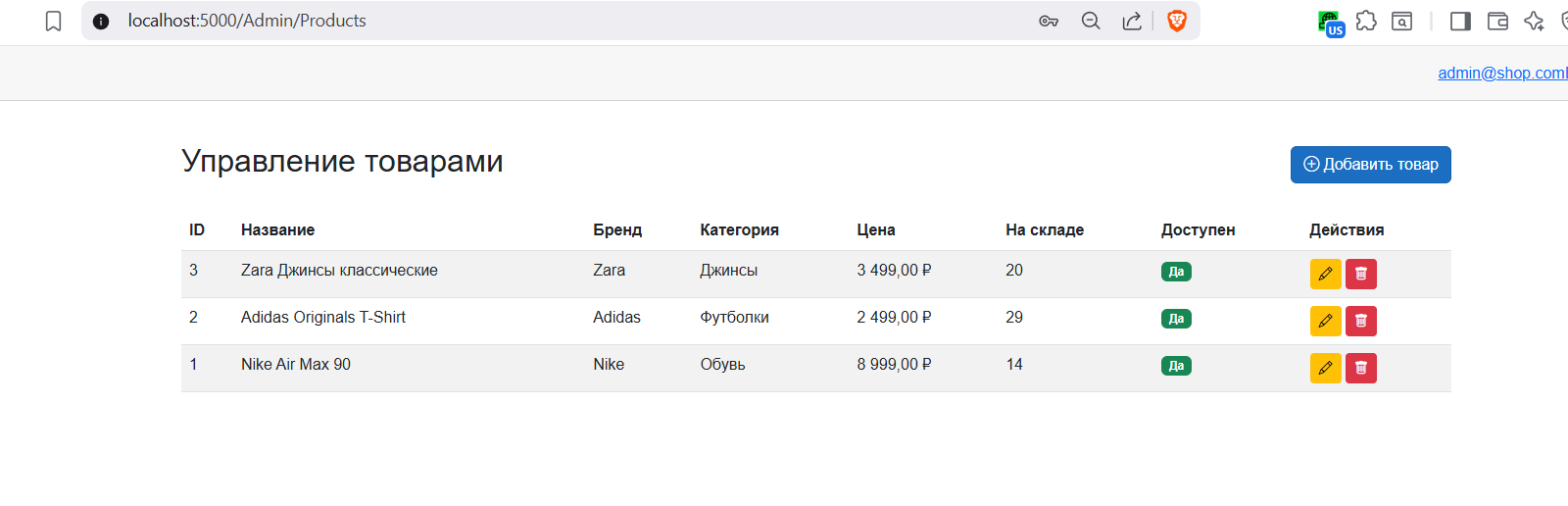


Рисунок 2.40 – Фрагмент экранной копии, Управление товарами

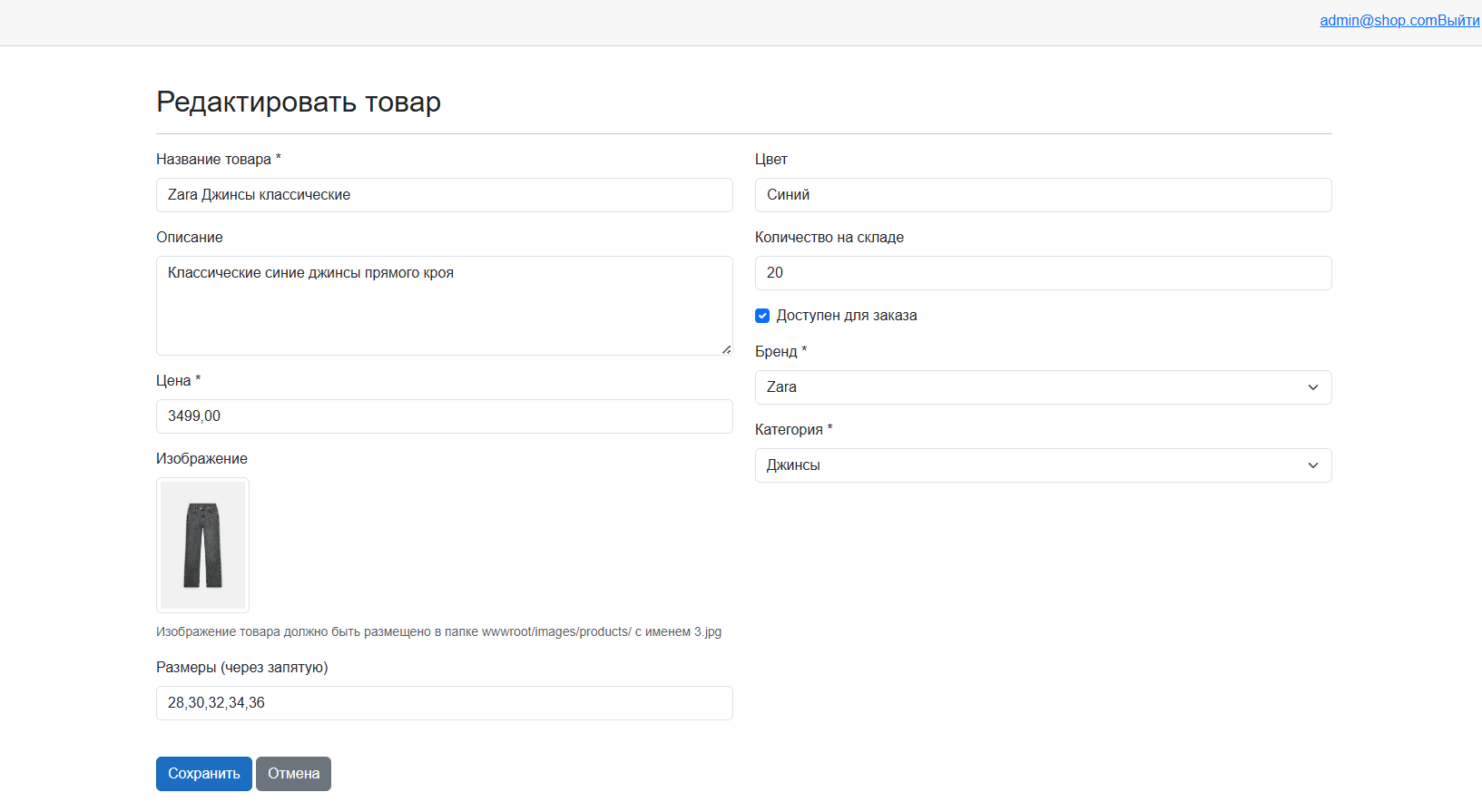


Рисунок 2.40 – Фрагмент экранной копии, Редактировать товар

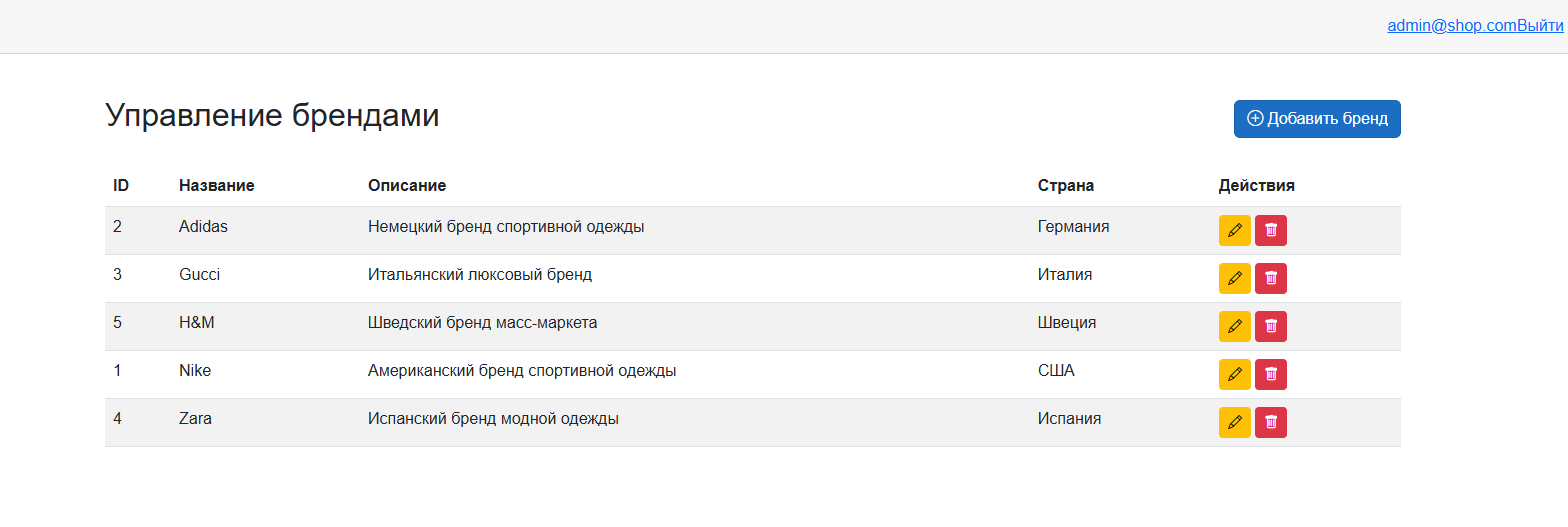
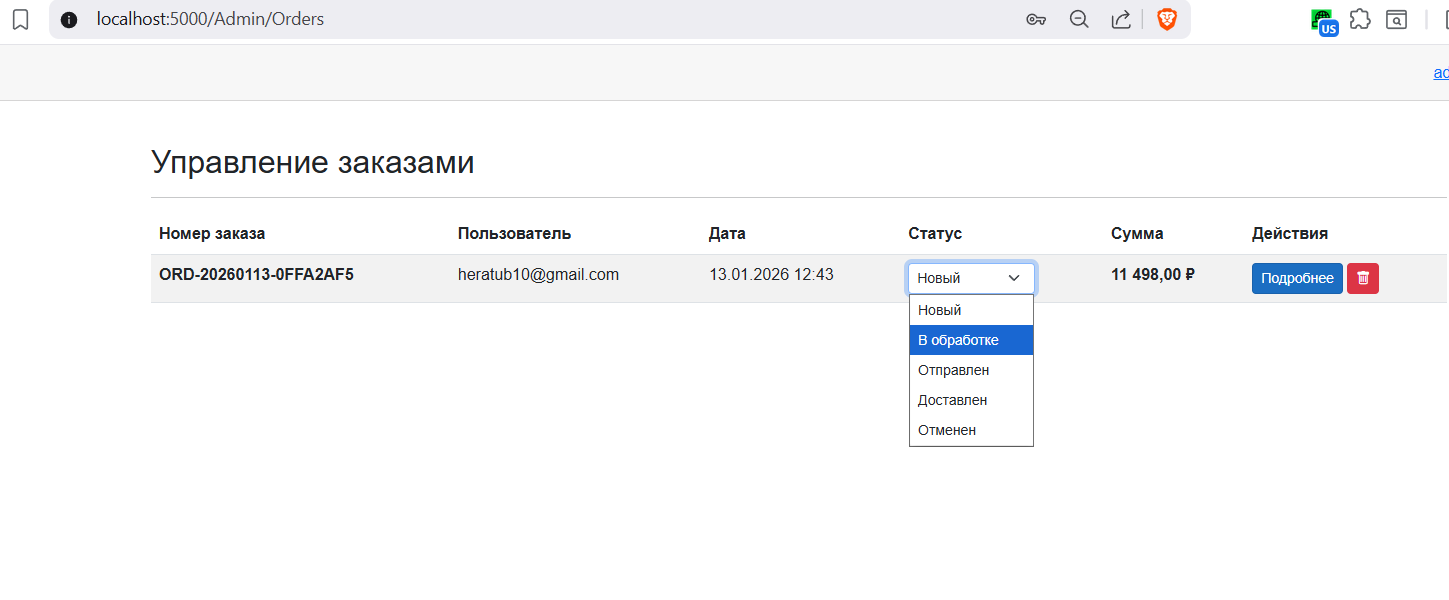
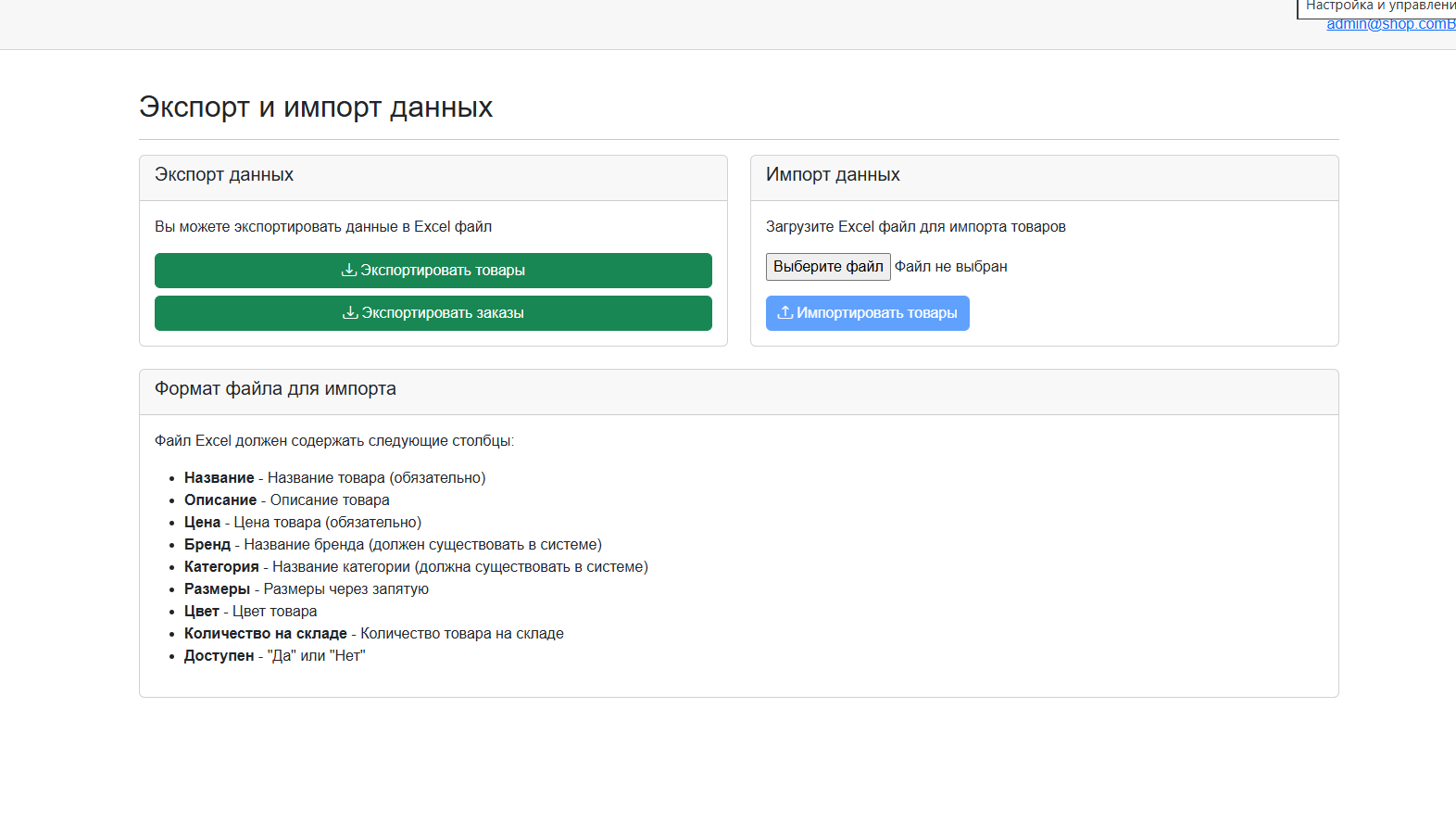
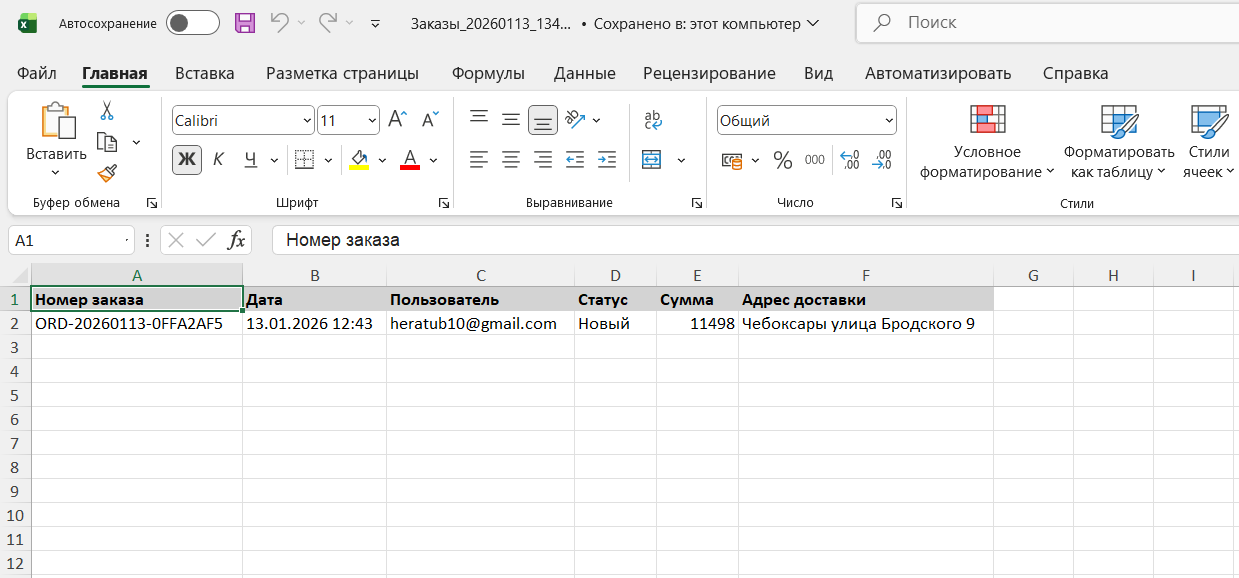
 Рисунок 2.40 – Фрагмент экранной копии, Управление брендами

 Рисунок 2.40 – Фрагмент экранной копии, Управление категориями

 Рисунок 2.40 – Фрагмент экранной копии, Управление заказами

 Рисунок 2.40 – Фрагмент экранной копии, Экспорт и импорт

 Рисунок 2.40 – Фрагмент экранной копии, Экспорт заказов

Выводы

Создано веб-приложение, в котором предусмотрены многие функции, позволяющие пользователю эффективно взаимодействовать с базой данных: каталог товаров с фильтрацией, корзина покупок, оформление заказов, система авторизации с ролями, административная панель для управления товарами, брендами, категориями и заказами, экспорт и импорт данных в Excel, личный кабинет пользователя.

В процессе разработки были выполнены следующие этапы:

* Создана база данных Shop с автоматической инициализацией структуры через метод EnsureCreated();
* Настроена система авторизации и аутентификации с использованием ASP.NET Core Identity;
* Реализованы дополнительные поля при регистрации пользователей (имя, фамилия, пол, дата рождения, предпочитаемый размер, город, телефон);
* Настроены политики авторизации для разграничения прав доступа по ролям (Admin, Manager, Customer);
* Реализована инициализация начальных данных при первом запуске приложения (роли, администратор, бренды, категории, товары);
* Созданы Blazor компоненты для интерактивных страниц каталога, корзины, заказов;
* Реализован экспорт и импорт данных в формате Excel с использованием библиотеки EPPlus;
* Настроена навигация между страницами через \_Layout.cshtml;
* Проведено заполнение всех страниц тестовыми данными и проверка работоспособности всех функций приложения.

# **Заключение**

Для курсового проекта в качестве предметной области выбрана деятельность интернет-магазина по продаже брендовой одежды с целью проектирования для неё веб-приложения, которое будет удобно и практично для работы в данной сфере и которое облегчит управление товарами, заказами и пользователями.

Во время выполнения курсового проекта решены следующие задачи:

* На этапе проведения анализа предметной области рассмотрены существующие интернет-магазины (Wildberries, Lamoda, Brandshop). С их помощью определен перечень функций, которые должны быть реализованы в веб-приложении онлайн-магазина по продаже брендовой одежды.
* На этапе концептуального проектирования выделены основные действующие лица интернет-магазина: Администратор, Менеджер, Покупатель. Определены их функции и составлена диаграмма вариантов использования, а также составлен список объектов, данные которых нужно хранить в системе.
* На этапе логического проектирования выделены основные сущности (Бренд, Категория, Товар, Пользователь, Заказ, ЭлементЗаказа). Исходя из них, построены ER-диаграммы. Из разработанных ER-диаграмм, согласно правилам, сформирован предварительный набор отношений для хранения данных. После этого добавлены неключевые атрибуты в сформированные отношения.
* На этапе физического проектирования определены основные таблицы, составлена схема базы данных, диаграмма бизнес-процессов IDEF0 и диаграмма потоков данных DFD.

При проектировании веб-приложения предусмотрены многие функции, позволяющие пользователю эффективно взаимодействовать с ним: каталог товаров с фильтрацией, корзина покупок, оформление заказов, система авторизации с ролями и разграничением прав доступа, административная панель для управления товарами, брендами, категориями и заказами, экспорт и импорт данных в Excel, личный кабинет пользователя с историей заказов.

Веб-приложение разработано с использованием современных технологий: ASP.NET Core Blazor Server, Entity Framework Core, ASP.NET Core Identity, Microsoft SQL Server, что обеспечивает надежность, безопасность и производительность системы.

Цель курсового проекта достигнута, все поставленные задачи выполнены.

# **Список используемых источников**

1. Дикман, Л. ASP.NET Core в действии / Лок Э. – М.: ДМК Пресс, 2021. – 906 с.
2. Троelsen, Э. Язык программирования C# 10 и платформа .NET 6 / Троelsen Э., Джепикс Ф. – М.: Вильямс, 2022. – 1440 с.
3. Фримен, А. ASP.NET Core MVC 2 с примерами на C# для профессионалов / Фримен А. – М.: Вильямс, 2019. – 1040 с.
4. Майкрософт. Документация по Entity Framework Core [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/ef/core/, свободный. Дата обращения: 15.01.2025
5. Майкрософт. Документация по ASP.NET Core Blazor [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/blazor/, свободный. Дата обращения: 15.01.2025
6. Майкрософт. Документация по ASP.NET Core Identity [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/aspnet/core/security/authentication/identity, свободный. Дата обращения: 15.01.2025
7. Википедия. ASP.NET Core [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/ASP.NET\_Core, свободный. Дата обращения: 15.01.2025
8. SQL Server Documentation [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/sql/sql-server/, свободный. Дата обращения: 15.01.2025
9. EPPlus Documentation [Электронный ресурс] – Режим доступа: https://github.com/EPPlusSoftware/EPPlus, свободный. Дата обращения: 15.01.2025
10. Майкрософт. Документация по проектированию баз данных [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/database-design/, свободный. Дата обращения: 15.01.2025
11. Фаулер, М. Архитектура корпоративных программных приложений / Фаулер М. – М.: Вильямс, 2020. – 544 с.

# 