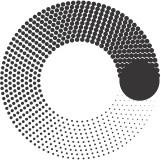
**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ВЫСШАЯ ШКОЛА ПЕЧАТИ И МЕДИАИНДУСТРИИ**

***Институт Принтмедиа и информационных технологий Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»,**

КУРСОВОЙ ПРОЕКТ

**Дисциплина: Back-end разработка**

**Тема:** Онлайн магазин электроники

**Выполнил: студент группы 221-374**

Кузин Артём Александрович

**Дата, подпись**

(Дата) (Подпись)

**Проверил:**

(Фамилия И.О., степень, звание)

**Дата, подпись**

(Дата) (Подпись)

**Замечания: \_**

**Москва 2024**

Оглавление

[ВВЕДЕНИЕ 3](#_Toc163512628)

[ГЛАВА 1 Проектирование 4](#_Toc163512629)

[1.1 Описание предметной области 4](#_Toc163512630)

[1.2 Выбор инструментов разработки 4](#_Toc163512631)

[ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ 7](#_Toc163512632)

[2.1 Проектирование и физическая реализация базы данных 7](#_Toc163512633)

[2.2 Разработка API 14](#_Toc163512634)

[2.3 Разработка UI и реализация Fetch API 28](#_Toc163512635)

[ЗАКЛЮЧЕНИЕ 37](#_Toc163512636)

[Библиографический список 38](#_Toc163512637)

# ВВЕДЕНИЕ

Тема курсового проекта – разработка на базе кроссплатформенных WEB технологий информационной системы онлайн магазина электроники, обеспечивающей удобную покупку техники.

В современном мире электронная коммерция становится все более популярной и востребованной среди потребителей. С развитием интернет-технологий и расширением возможностей онлайн платформ, возникла необходимость в создании удобных и функциональных информационных систем для онлайн магазинов. Удобная и качественная информационная система позволит пользователям без особых проблем купить тот товар, который их интересует.

Цель данного проекта заключается в создании высококачественной информационной системы, которая обеспечит пользователям удобный и эффективный процесс покупки техники через онлайн магазин. Информационная система позволит клиентам легко найти необходимые товары, совершить покупку с минимальными усилиями и удобно произвести оплату.

Исходя из поставленной цели и описанной проблемы, были сформированы следующие задачи

* + 1. Изучение предметной области;
    2. Изучение технологий для разработки WEB API;
    3. Изучение методов работы с запросами HTTP;
    4. Выбор оптимальных инструментов разработки для создания системы;
    5. Проектирование базы данных, определение структуры и связей между таблицами;
    6. Реализация физической модели данных.

# ГЛАВА 1 Проектирование

# Описание предметной области

Онлайн магазин – это информационная система, позволяющая пользователям совершать различные покупки, не выходя из дома. При этом пользователи хотят пользоваться удобным, информативным и понятным сервисом.

Для этого была создана информационная система, упрощающая процесс покупки и других нужд пользователя.

Приложение будет использоваться обычными пользователями, а также администраторами информационной системы.

Пользователь системы может посмотреть список товаров в интернет-магазине, выбрать интересующий его товар, количество и оформить заказ. Вся информация о заказе будет автоматически добавлена в базу данных.

Функционал для администраторов приложения предоставляет больший функционал для работы с информационной системы. Администратор имеет возможность добавлять, изменять или удалять данные о товарах в магазине, пользователях, просматривать заказы пользователей.

# Выбор инструментов разработки

Выбор оптимальных инструментов для разработки информационной системы имеет решающее значение для успешной реализации проекта и обеспечения комфортной работы разработчиков и пользователей системы.

Важным критерием выбора является функциональность и удобство работы с базой данных. В реализуемом проекте был выбран такой инструмент, как SQL Server 2020 Express - это бесплатная и легкая в использовании версия управляемой реляционной системы управления базами данных Microsoft SQL Server. Она предлагает множество возможностей SQL Server, таких как управление данными, создание запросов, хранение и обработку информации, но в урезанной по сравнению с полными версиями функциональностью.

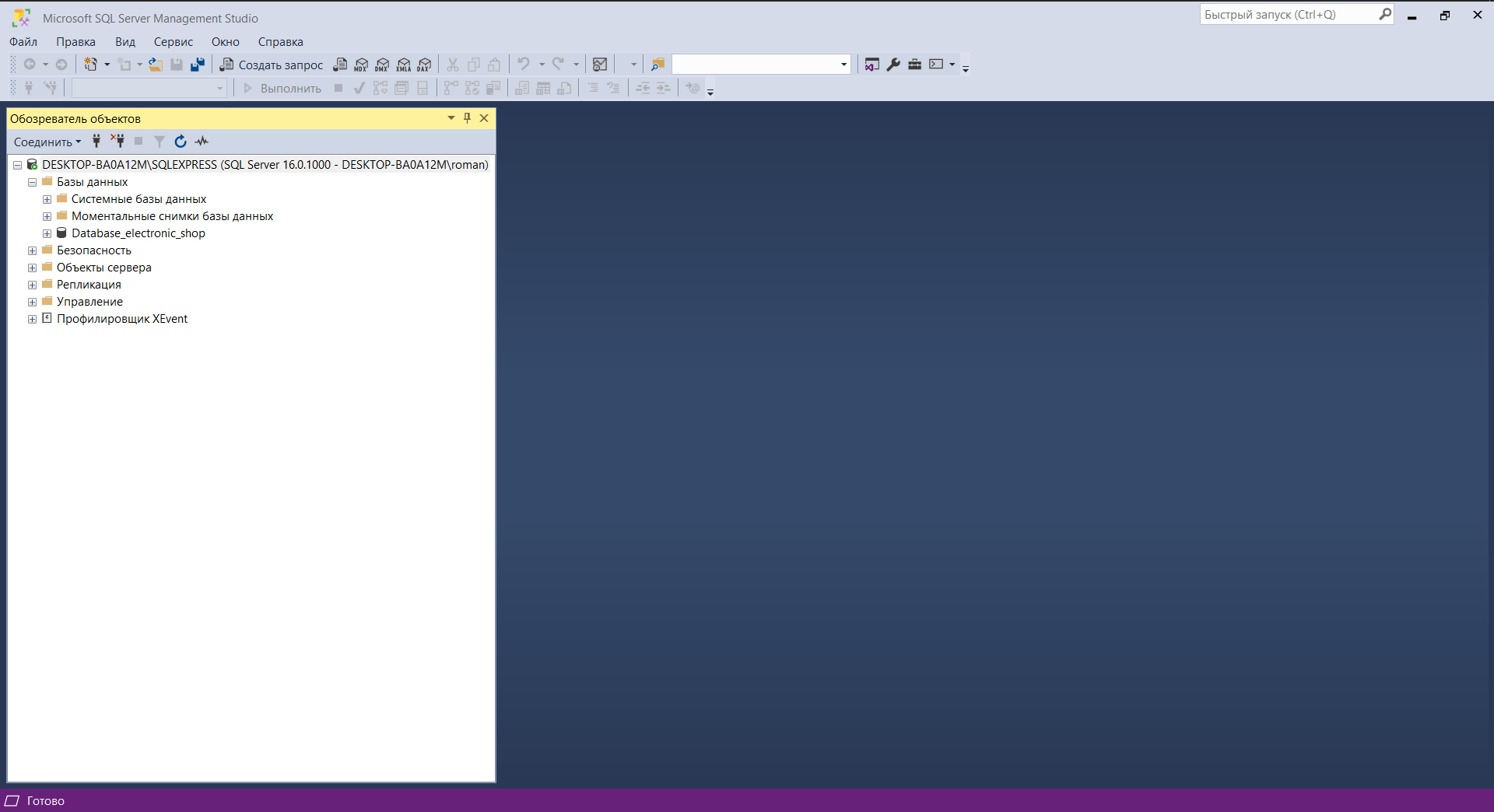


Рисунок 1 – Оболочка SQL Server 2020 Express

Для реализации проекта было выбрано WEB-приложение. Данный формат проекта был выбран потому, что люди пользуются различными операционными системами, а доступ к оформлению заказа должен быть с разных платформ (IOS, Android). Для создания информационной системы использовался язык программирования C#. Язык является мощным и гибким языком программирования, разработанным компанией Microsoft. Он широко используется для создания разнообразных приложений, включая веб-приложения. C# это объектно-ориентированный язык программирования, который также поддерживает компонентно-ориентированное программирование.

В качестве фреймворка для создания API был выбран ASP.NET Core.

Для разработки web-приложения «онлайн магазин электроники» была выбрана такая среда программирования, как Microsoft Visual Studio, так как с помощью неё есть возможность разрабатывать множество типов приложения под разные платформы и обладает хорошей интеграцией с другими компонентами Microsoft.

Также, среда разработки Microsoft Visual Studio предоставляет возможность работы с фреймворком ASP.NET Core, который обеспечивает множество дополнительных инструментов и библиотек для создания эффективного API.

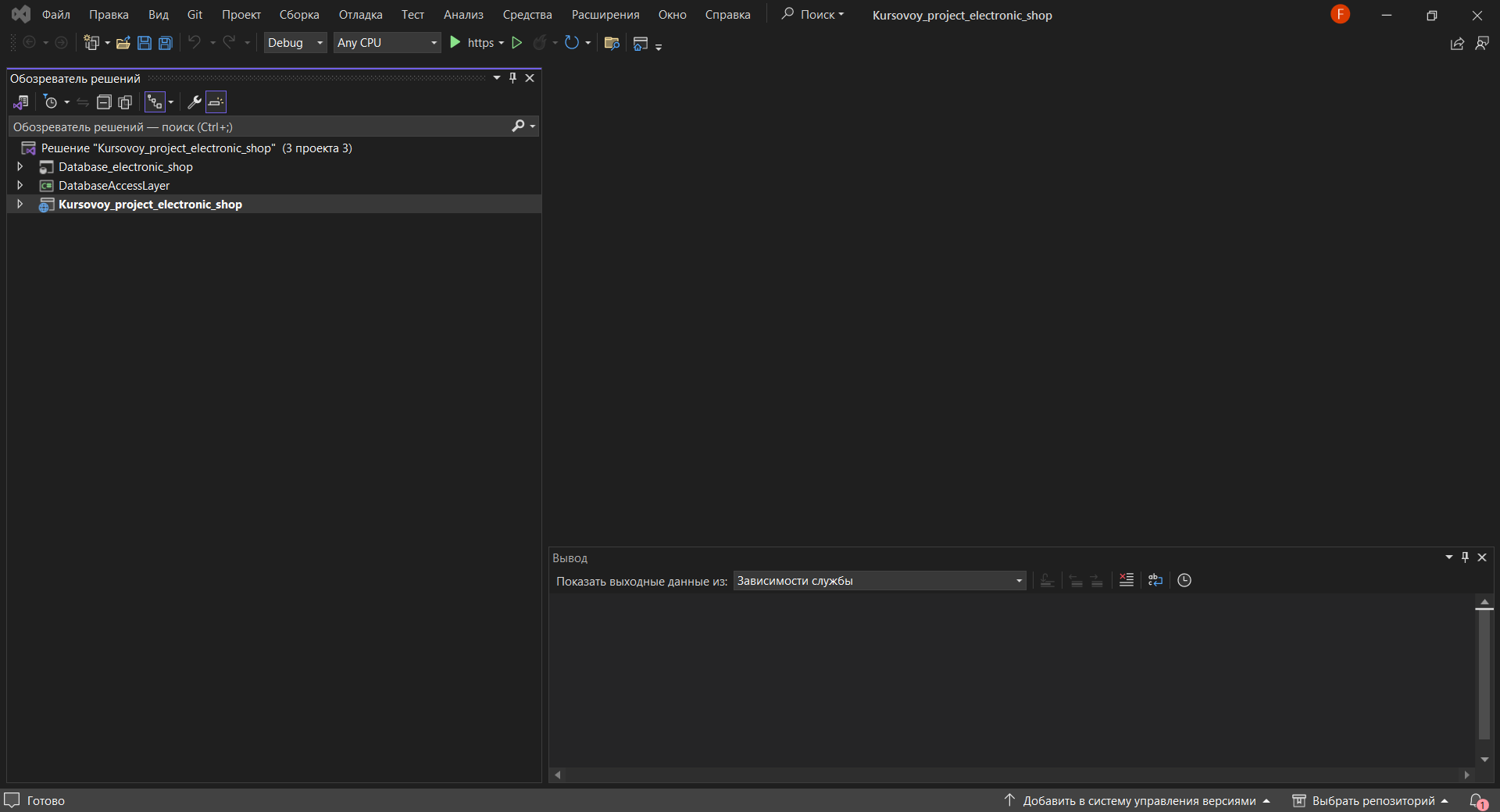


Рисунок 2 – Оболочка Microsoft Visual Studio

# ГЛАВА 2. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ

**«Онлайн магазин электроники»**

# Проектирование и физическая реализация базы данных

В рамках данного проекта была спроектирована и создана база данных, содержащая семь связанных таблиц. Таблица реляционной базы данных – это совокупность связанных данных, хранящихся в структурированном виде в базе данных. Данная таблица показана на рисунке ниже.

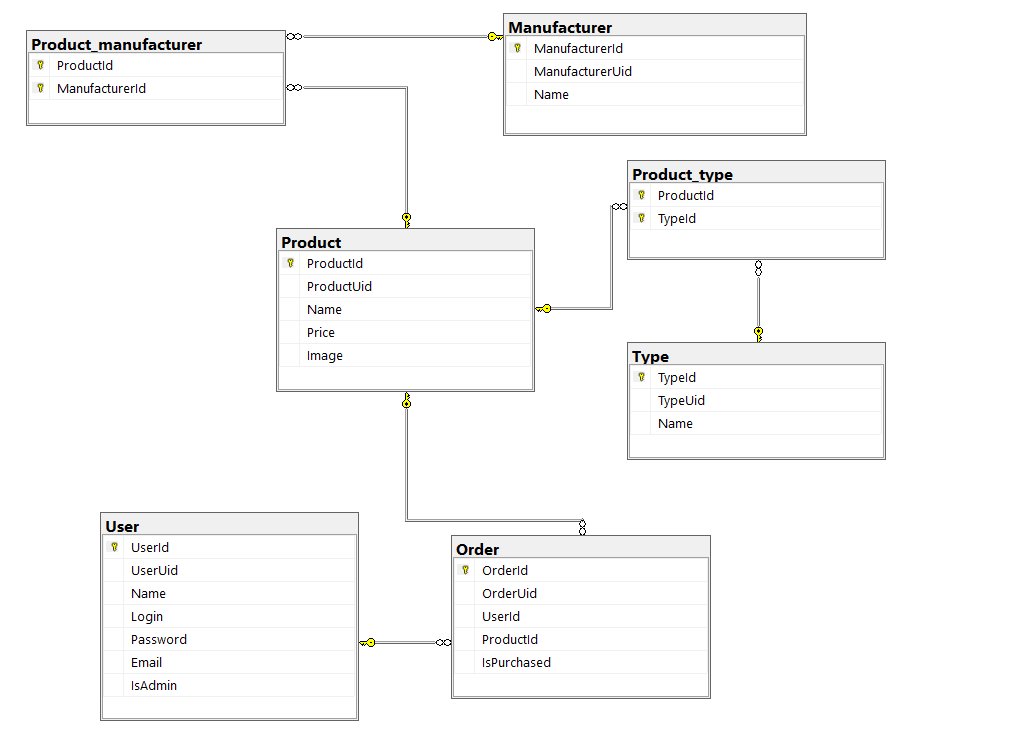


Рисунок 3 – Диаграмма базы данных

Далее будут представлены структуры всех таблиц с пояснением к каждой и скриншотом её представления в системе SQL Server Management Studio 20.

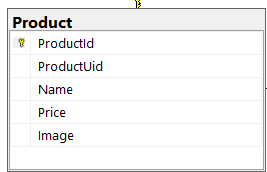


Рисунок 4 – Таблица товаров

* ProductId – уникальный идентификатор товара в базе данных
* ProductUid – уникальный идентификатор товара на сайте
* Name – название товара
* Price – цена товара
* Image – картинка товара

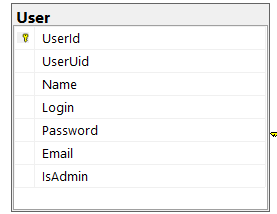


Рисунок 5 – Таблица пользователей

* UserId – уникальный идентификатор пользователя в базе данных
* UserUid – уникальный идентификатор пользователя на сайте
* Name – Имя пользователя
* Login – Логин пользователя
* Password – пароль пользователя
* Email – электронная почта пользователя
* IsAdmin – флаг, который указывает является ли пользователь администратором

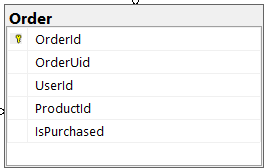


Рисунок 6 – Таблица заказов

* OrderId – уникальный идентификатор заказа в базе данных
* OrderUid – уникальный идентификатор заказа на сайте
* UserId – идентификатор пользователя
* ProductId – идентификатор товара
* isPurchased – флаг, который указывает является ли заказ оплаченым

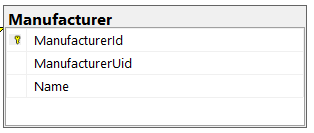


Рисунок 7 – Таблица производителей

* ManufacturerId – уникальный идентификатор производителя в базе данных
* ManufacturerUid – уникальный идентификатор производителя на сайте
* Name – название производителя



Рисунок 8 – Таблица типов товаров

* TypeId – уникальный идентификатор типа товара в базе данных
* TypeUid – уникальный идентификатор типа товара на сайте
* Name – название типа товара

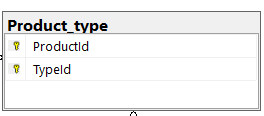


Рисунок 9 – Таблица продуктов и типов товаров

Таблица Product\_type является связующей таблицей и содержит информацию о товарах и их типах. Она содержит такие поля как:

* ProductId – идентификатор товара
* TypeId – идентификатор типа товара

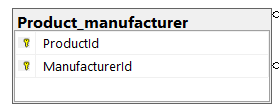


Рисунок 10 – Таблица продуктов и их производителей

Таблица Product\_manufacturer является связующей таблицей и содержит информацию о товарах и их производителях. Она содержит такие поля как:

* ProductId – идентификатор товара
* ManufacturerId – идентификатор производителя товара

С помощью SQL Server 2020 Express была реализована база данных для информационной системы «онлайн магазин электронной техники»

Следом была разработана диаграмма классов:

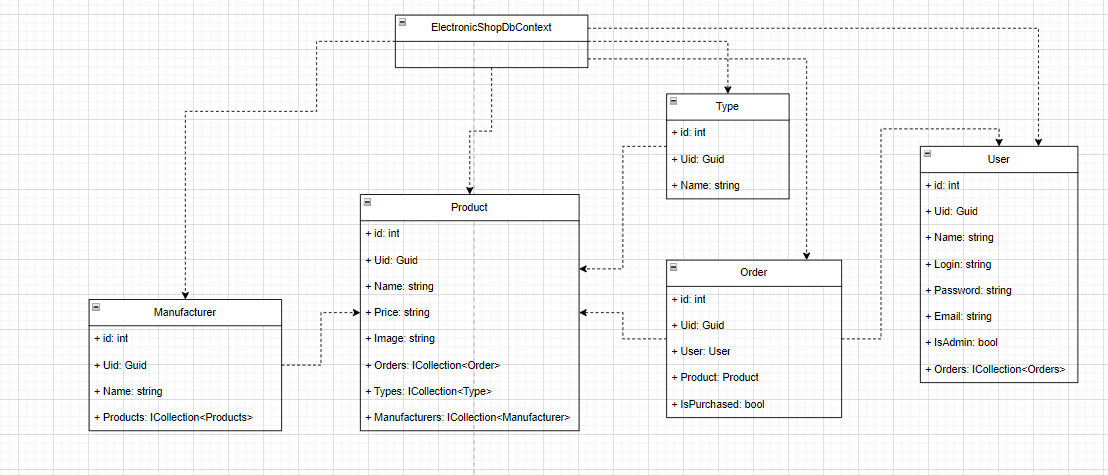


Рисунок 11 – Диаграмма классов

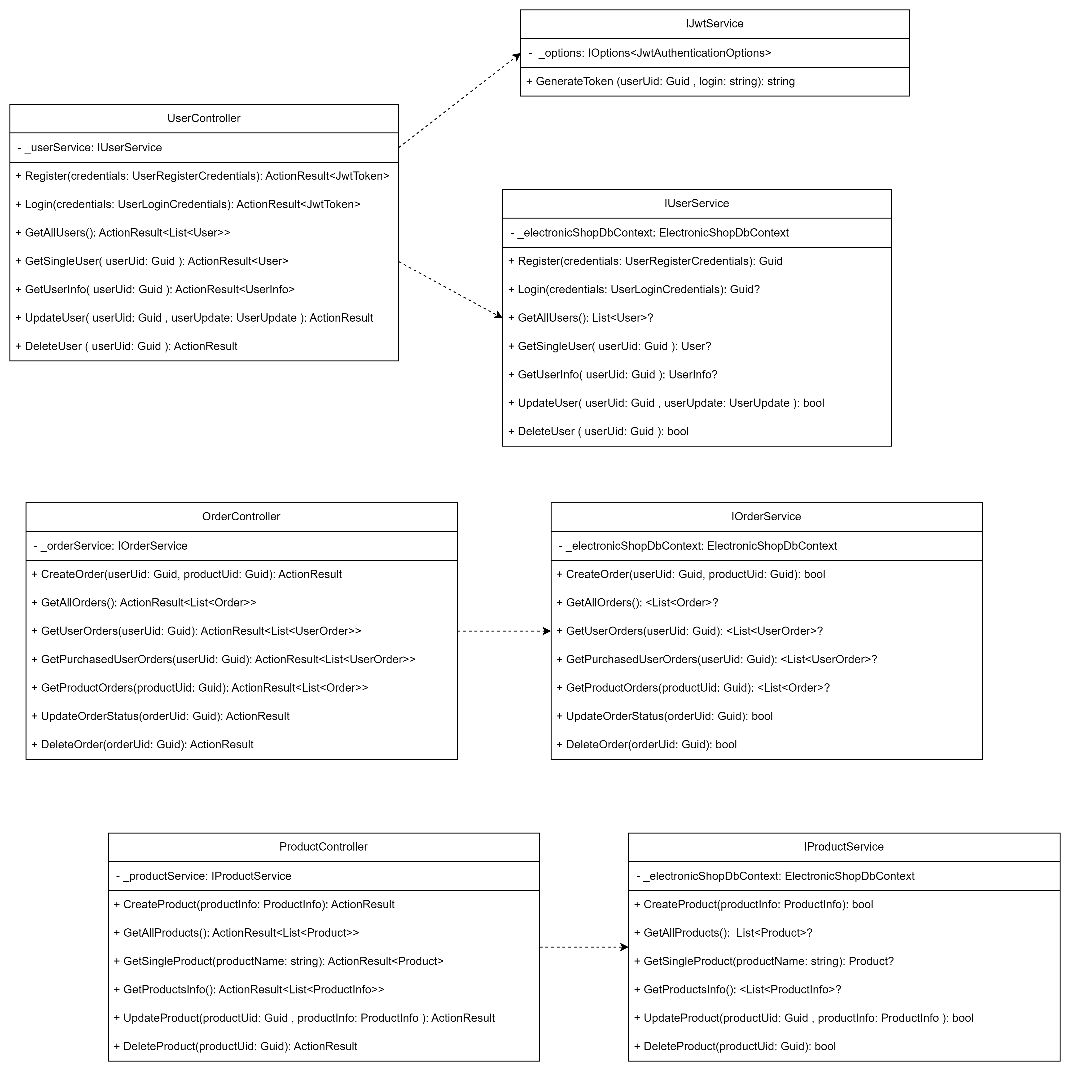


Рисунок 12.1 – API

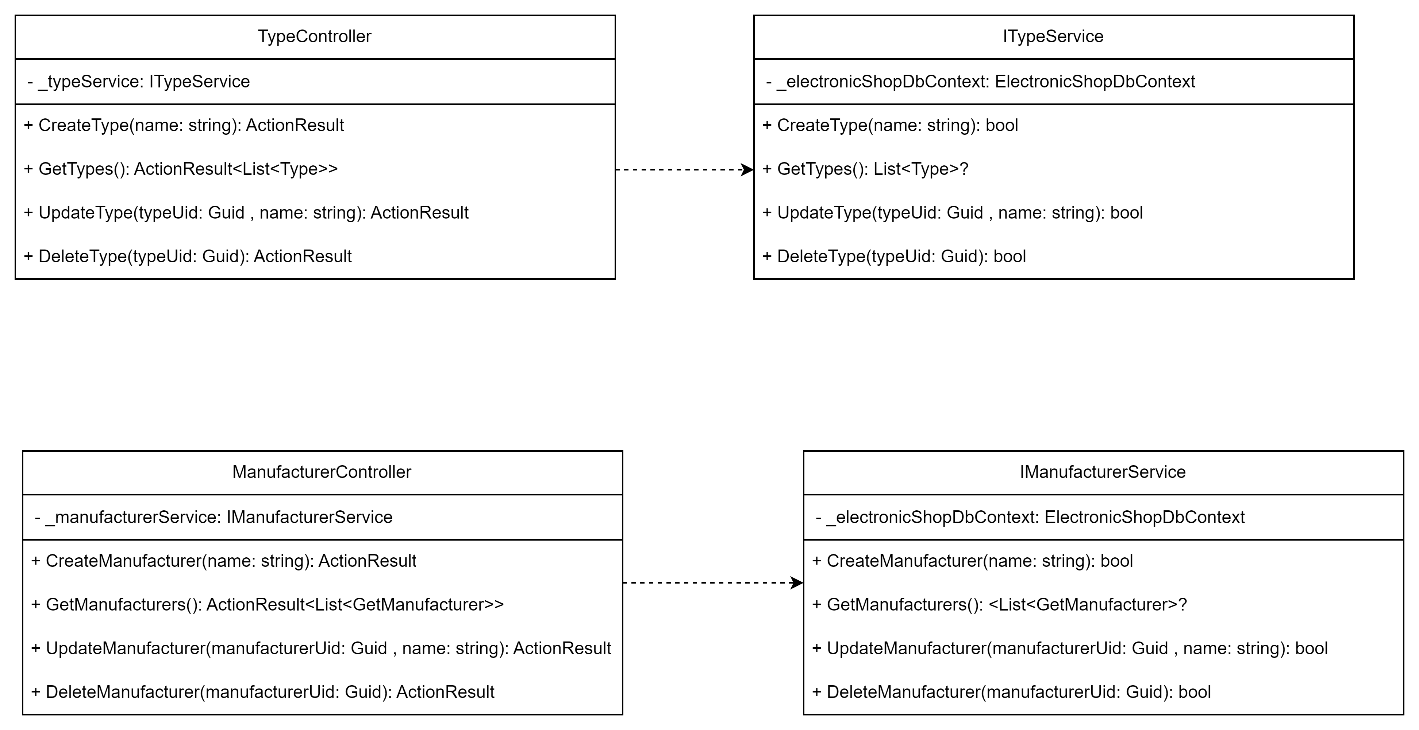


Рисунок 12.2 – API

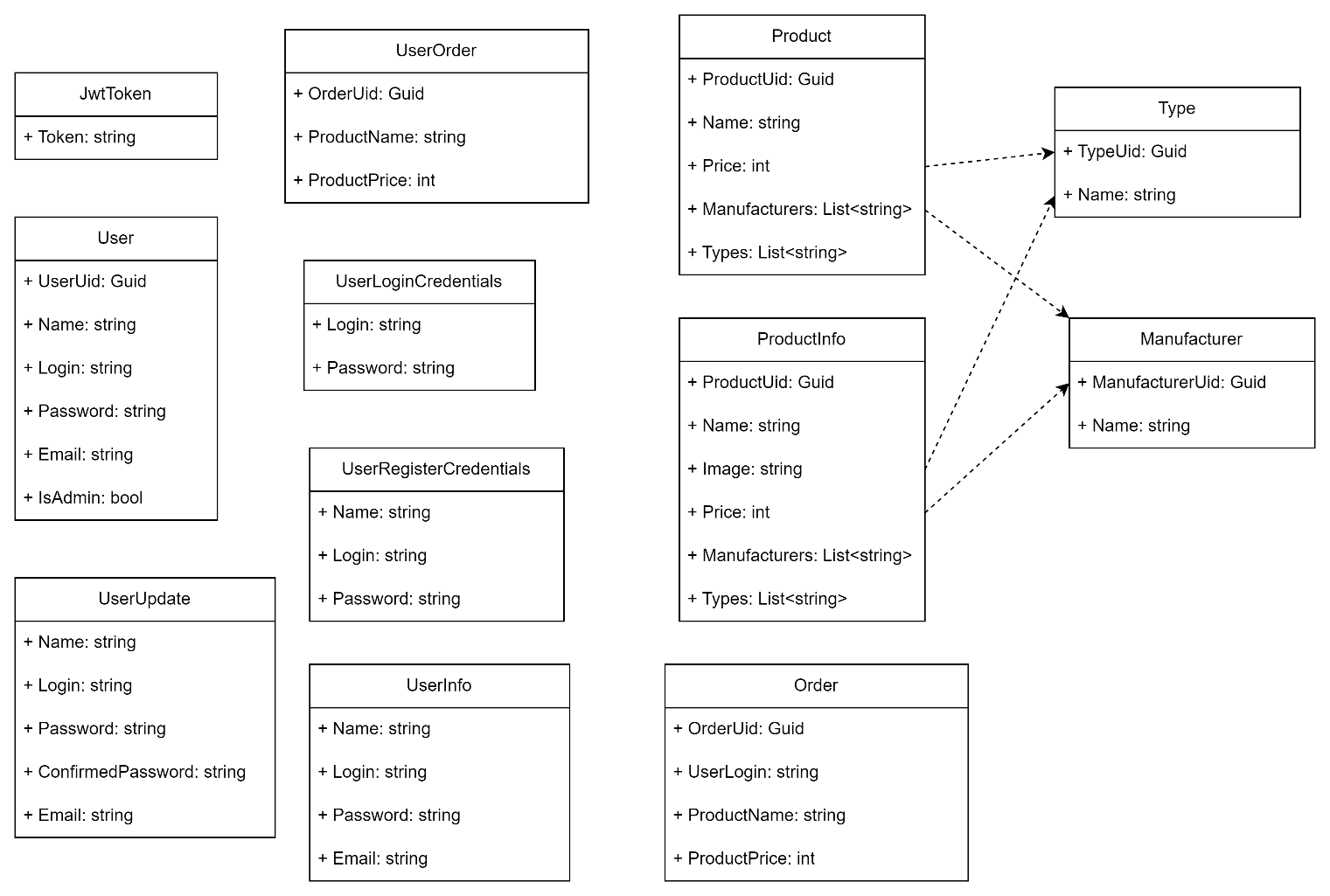


Рисунок 13 – Контрактные модели

# Разработка API

Для понимания того, как проходила разработка данного проекта необходимо сначала изучить его ключевые компоненты, которые были реализованы на языке C#.

В архитектуре ASP.NET Core одним из основных компонентов является контроллер, который отвечает за обработку запросов. При поступлении запроса, система маршрутизации определяет соответствующий контроллер для его обработки и передаёт ему данные запроса. Контроллер обрабатывает запрос и возвращает результат выполненной операции.

Другим важным компонентом является сервис, который выполняет определённые операции и обработку данных. Сервис взаимодействует с базой данных, файловой системой и другими сервисами для получения или обновления данных. Контроллер вызывает методы сервиса для работы с данными.

Контрактные модели (или модели представления) – ещё один компонент, используемый для передачи данных между контроллером и клиентом. Они определяют структуру данных, передаваемую через сеть, и содержат только необходимую информацию. Это упрощает взаимодействие между сервером и клиентом.

И, наконец, класс DbContext представляет собой подключение к базе данных. DbContext обеспечивает взаимодействие с базой данных, выполнение операции CRUD и содержит информацию о структуре данных. Этот класс облегчает работу с данными и позволяет легко управлять таблицами, столбцами и связями в базе данных.

Далее будут показаны примеры реализации сервисов, контроллеров, контрактных моделей и класса DbContext:

На рисунке ниже представлен контроллер для пользователя. Метод Register() реализует HTTP POST запрос. Он принимает на вход контрактную модель UserRegisterCredentials, которая содержит в себе данные для регистрации. После регистрации пользователю присваиватется специальный JWT-токен. Данный токен используется для безопасной передачи данных между клиентом и сервером с помощью шифрования.



Рисунок 14 – метод Register()

Метод Login() реализует HTTP POST запрос. Он принимает на вход контрактную модель UserLoginCredentials, которая содержит в себе данные авторизации. После авторизации пользователю так же присваиватется специальный JWT-токен.

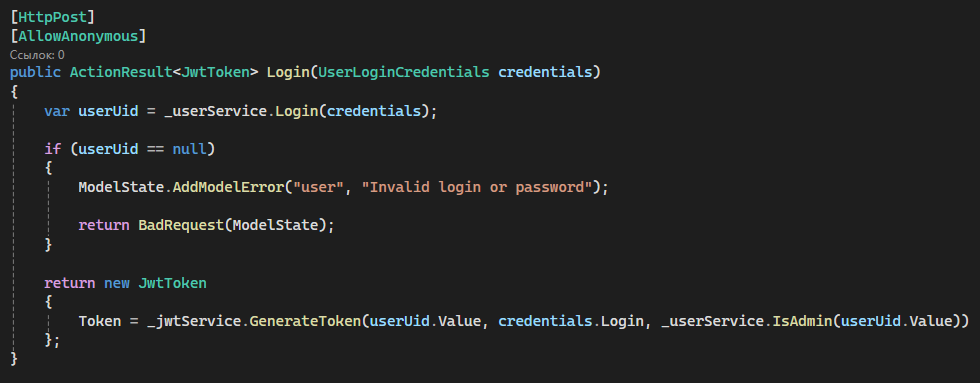


Рисунок 15 – метод Login()

Ниже представлен код реализации метода GenerateTocken(), который создаёт Jwt-токен.



Рисунок 17 – метод GenerateTocken()

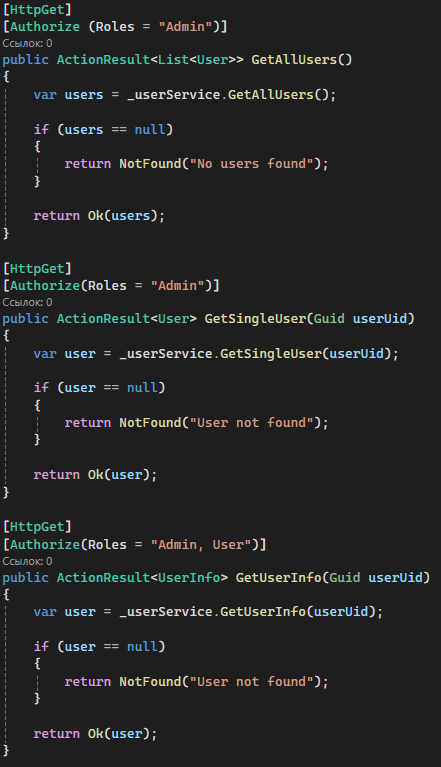
Такие методы, как GetAllUsers(), GetSingleUser(), GetUserInfo используют HTTP GET запрос. Эти методы используются администратором информационной системы для получения различной информации о пользователях.  
 

Рисунок 18 – методы GetAllUsers(), GetSingleUser(), GetUserInfo

Также администратор имеет возможность изменять информацию о пользователе. Для этого существует метод UpdateUser(). Этот метод использует HTTP PUT запрос. В качестве аргументов метод принимает uid пользователя и контрактную модель UserInfo, в которой содержатся данные для обновления.

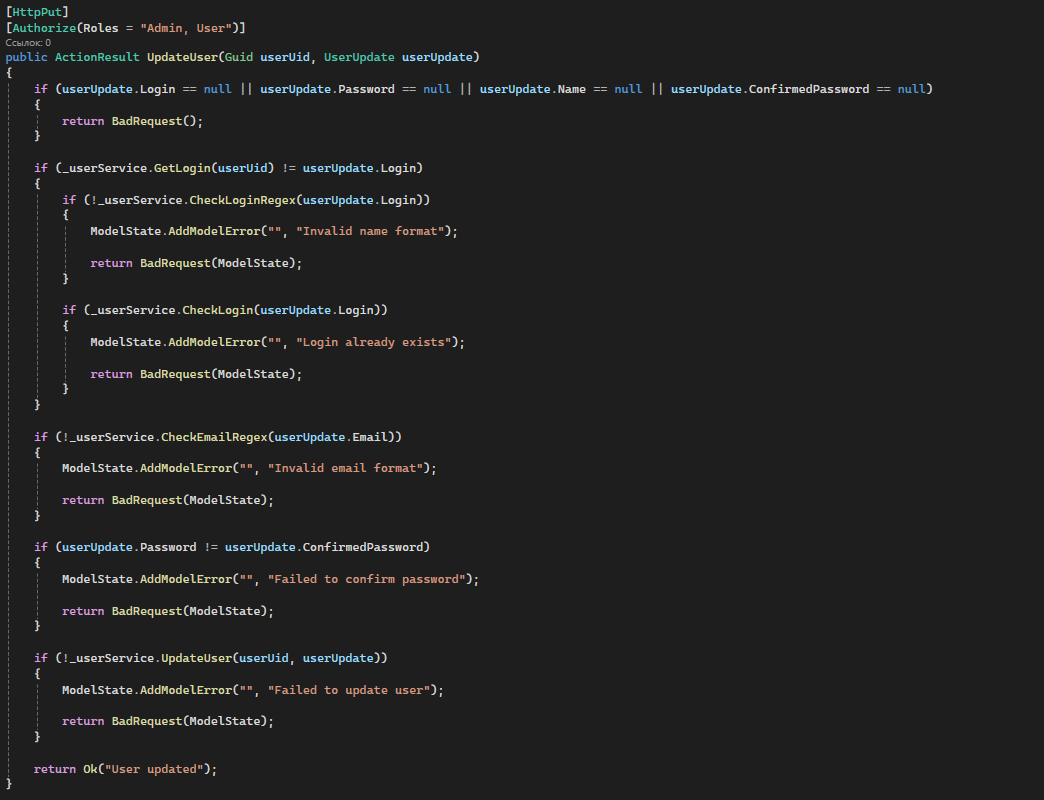


Рисунок 19 – метод UpdateUser()

Также администратор информационной системы может удалить пользователя. Для этого существует метод DeleteUser(). Этот метод использует HTTP DELETE запрос. В качестве параметров он принимает uid пользователя, который нужно удалить.

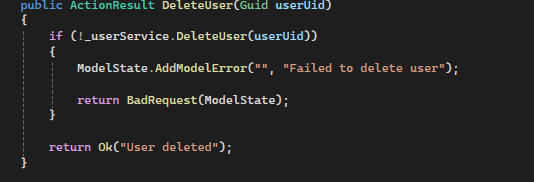


Рисунок 20 – метод DeleteUser()

Администратор информационной системы может также работать с другими данными, например со списком товаров на сайте. Так, с помощью метода CreateProduct(), администратор может добавить новый товар на сайт. Этот метод использует HTTP POST запрос и принимает в качестве аргумента контрактную модель, в которой находятся необходимые данные, такие как название, цена, производитель.



Рисунок 21 – метод CreateProduct()

Метод GettAllProducts() позволит администраторы вывести на экран все товары.

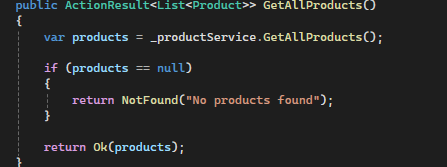


Рисунок – 22 метод GetAllProducts()

Также у администратора есть возможность изменять характеристики товаров. Для этого существует метод UpdateProduct(). Этот метод использует запрос HTTP PUT, принимает в качестве аргумента uid товара и контрактную модель, в которой находится вся необходимая для замены информация.

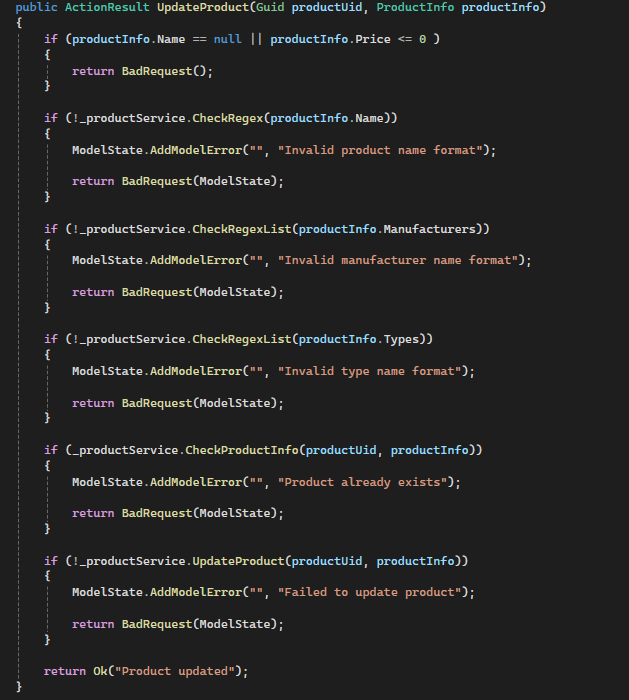


Рисунок 23 – метод UpdateProduct()

Администратор также может удалить товар. Для этого существует метод DeleteProduct(). Метод использует запрос HTTP DELETE и принимает в качестве аргумента uid товара.

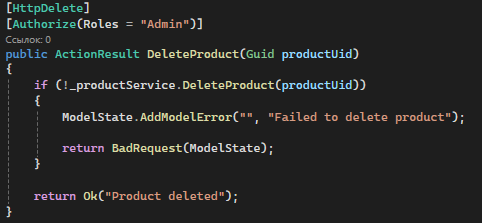


Рисунок 24 – метод DeleteProduct()

Далее будут представлены некоторые контрактные модели:

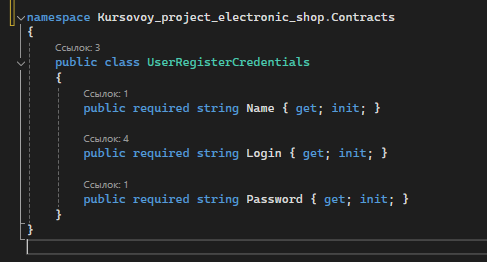


Рисунок 25 – контрактная модель регистрации пользователя

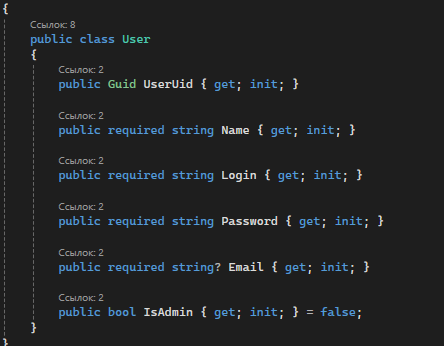


Рисунок 26 – контрактная модель пользователя

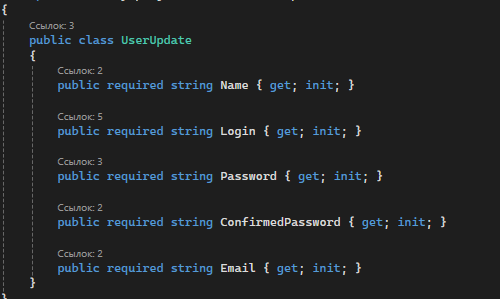


Рисунок 27 – контрактная модель изменения данных о пользователе

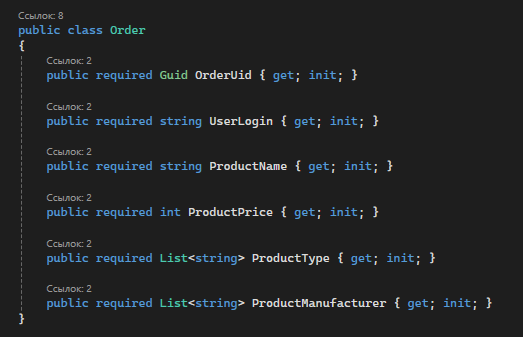


Рисунок 28 – контрактная модель заказа

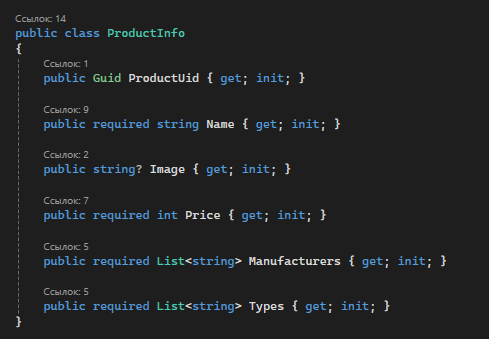


Рисунок 29 – контрактная модель информации о товаре

Как уже говорилось ранее, класс ElectronicShopDbContext используется для взаимодействия с базой данных.

Fluent API в ASP.NET Core используется для гибкой настройки отображения сущностей базы данных на объекты модели в Entity Framework Core.

Этот интерфейс позволяет детально настроить связи между сущностями, определяя типы связей (один-к-одному, один-ко-многим, многие-ко-многим), настройку каскадного удаления, управление именами и типами столбцов в базе данных и другие аспекты.

ORM (Object-Relational Mapping) является технологией программирования, которая создает абстракцию между реляционными базами данных и объектно-ориентированными языками программирования. ORM облегчает работу с данными, предоставляя высокоуровневую модель данных и автоматическое отображение объектов на таблицы в базе данных. Это позволяет программистам работать с данными на уровне объектов, без необходимости писать сложные SQL-запросы и работать с низкоуровневыми деталями структуры данных.

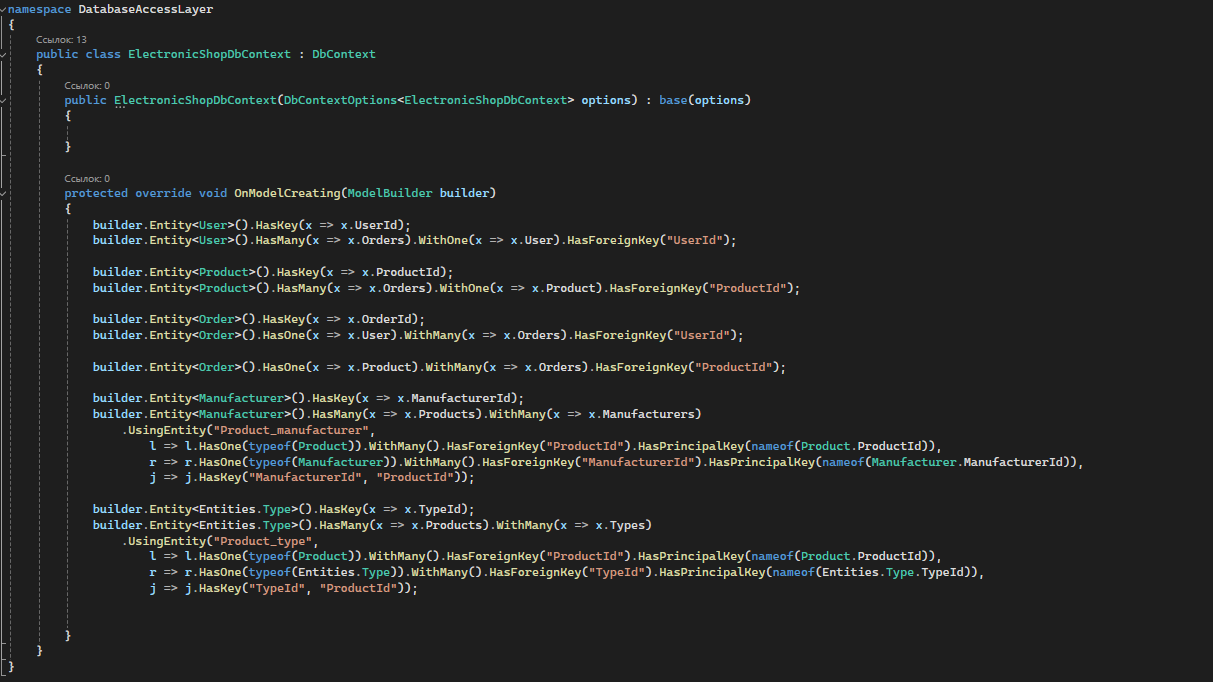


Рисунок 30 – класс ElectronicShopDbContext

Далее будут представлены классы для сущностей в базе данных:

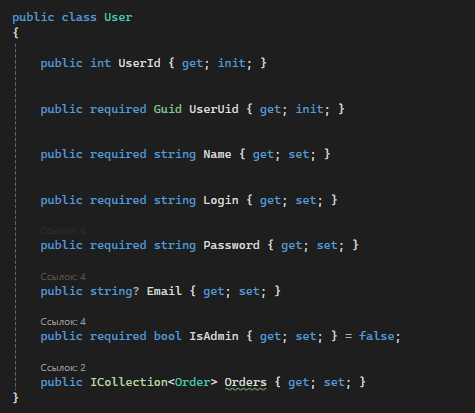


Рисунок 31 – класс для сущности User

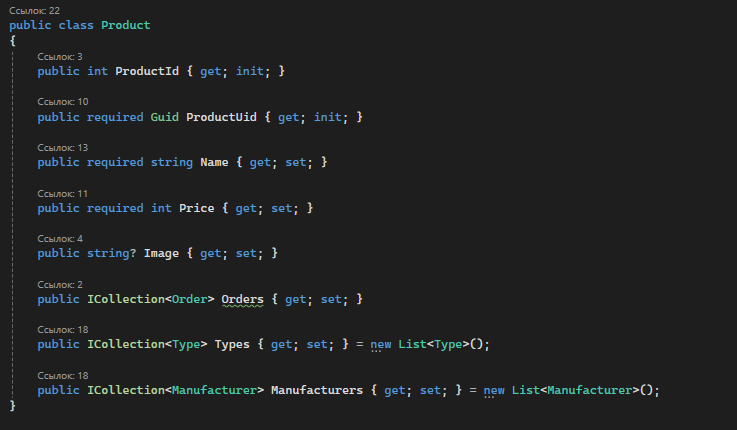


Рисунок 32 – класс для сущности Product

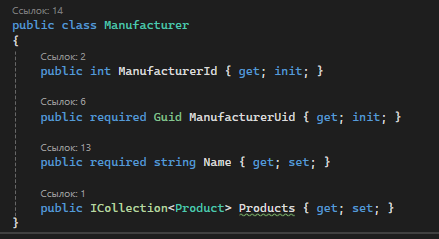


Рисунок 33 – класс для сущности Manufacturer

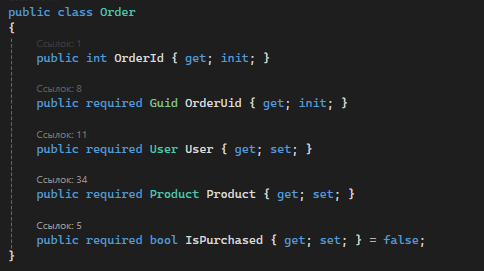


Рисунок 34 – класс для сущности Order

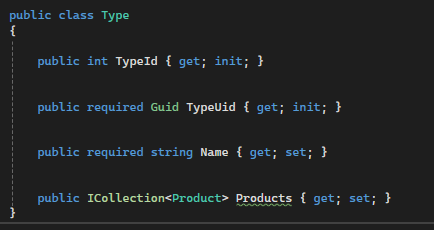


Рисунок 35 – класс для сущности Type

# Разработка UI и реализация Fetch API

Для разработки пользовательского интерфейса был выбран HTML, UI фреймворк Bootstrap и необходимые скрипты JavaScript для обработки событий.

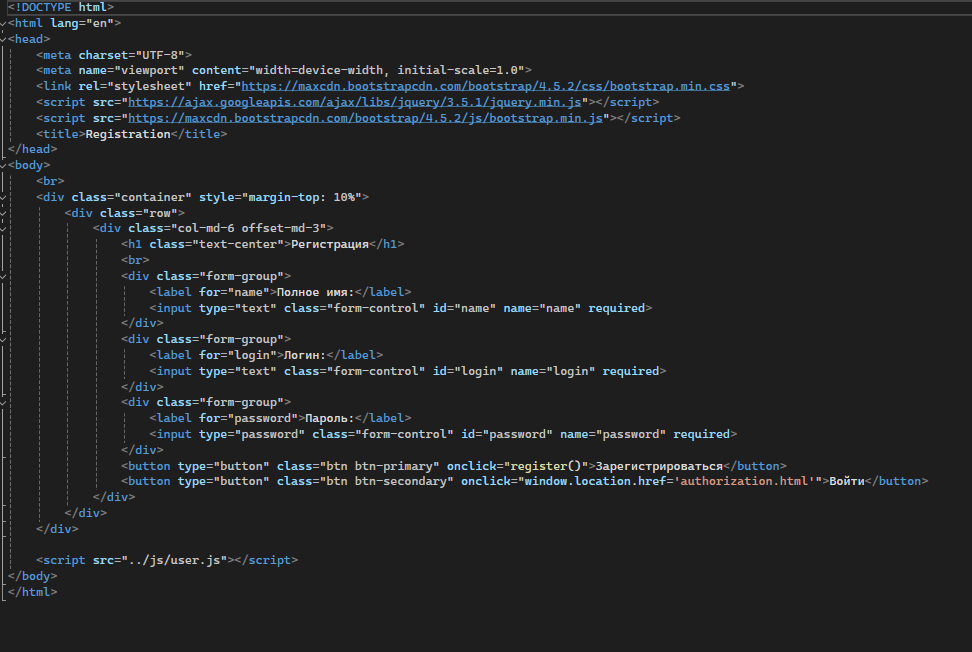


Рисунок 36 – страница регистрации пользователя

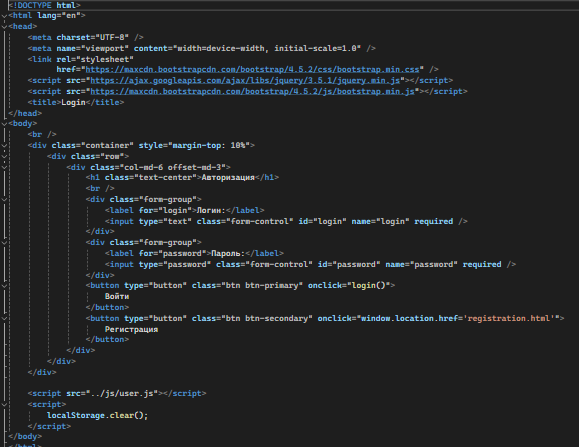


Рисунок 37 – страница авторизации пользователя

Далее будут представлены фрагменты кода, отвечающие за работу с запросами и ответами HTTP:

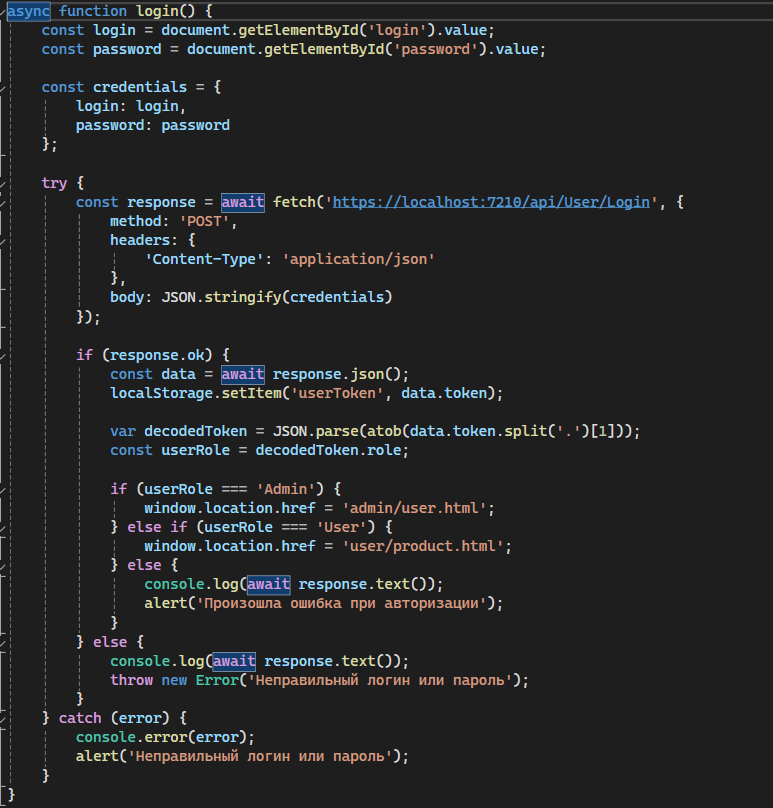


Рисунок 38 – функция авторизации пользователя



Рисунок 39 – функция регистрации пользователя



Рисунок 40 – функция получения информации о пользователе



Рисунок 41 – функция получения списка всех пользователей



Рисунок 42 – функция удаления пользователя



Рисунок 43 – функция получения списка всех товаров



Рисунок 44 – функция создания товара

Далее будут предоставлены скриншоты работающей информационной системы:

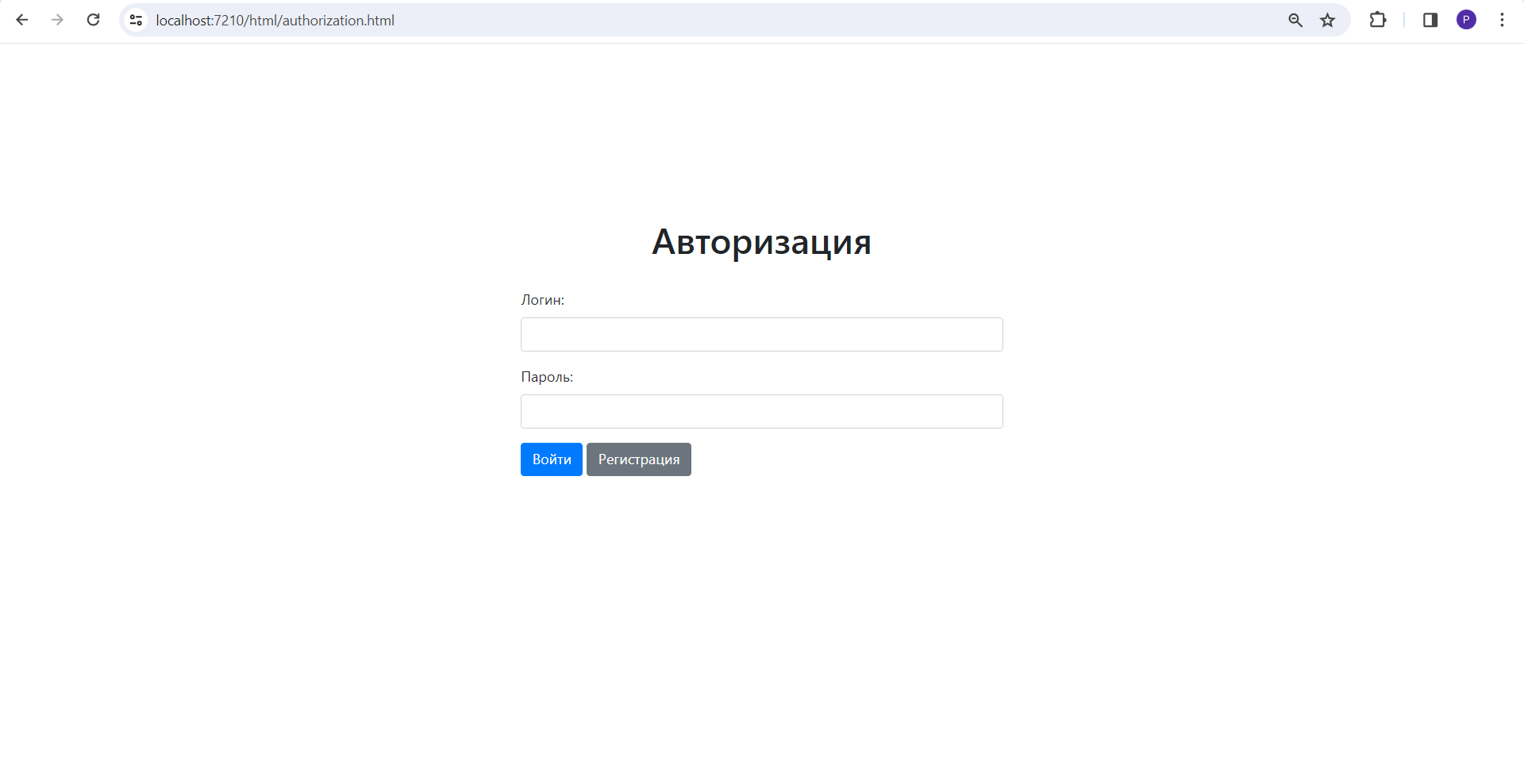


Рисунок 45 – страница авторизации

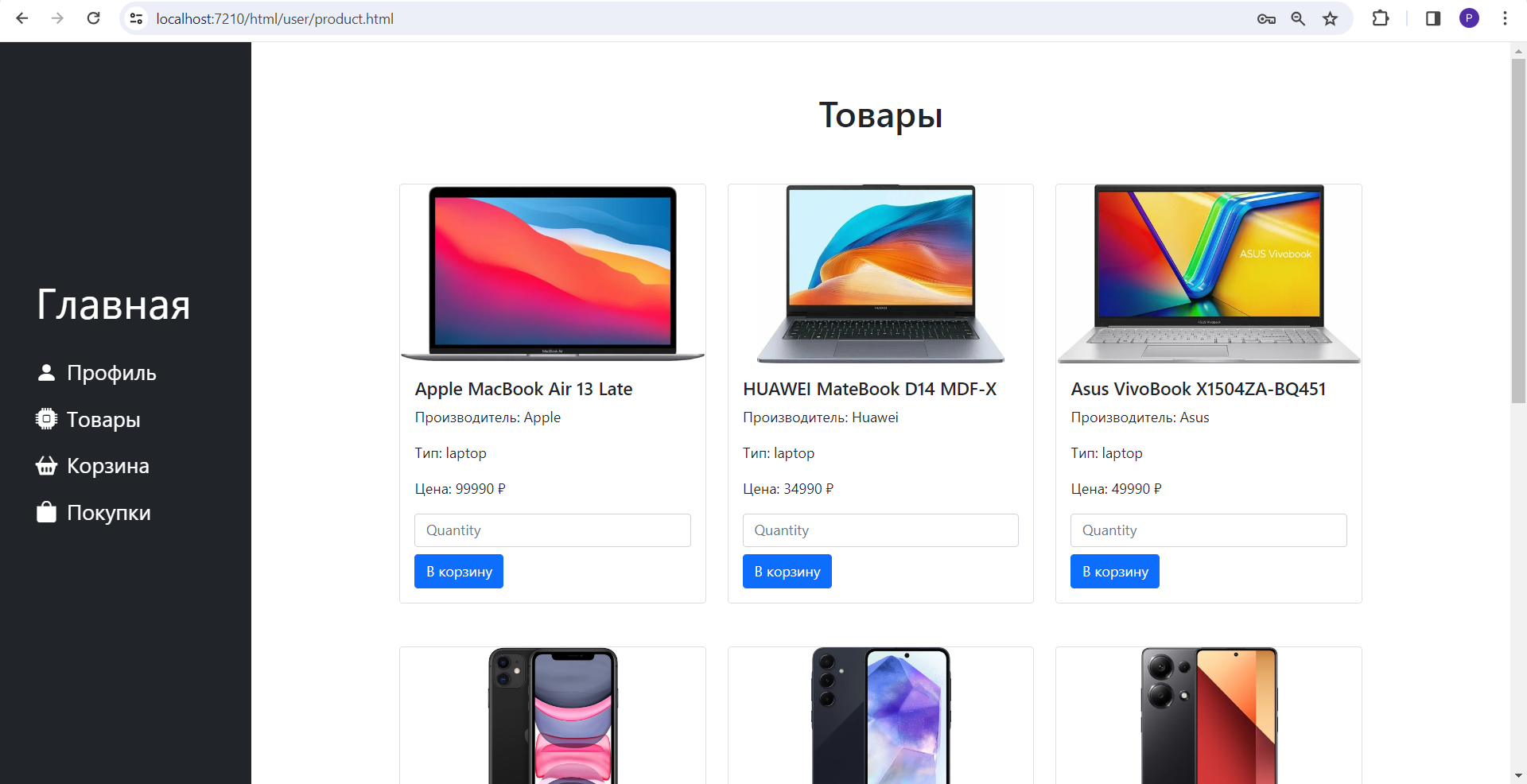


Рисунок 46 – страница товаров

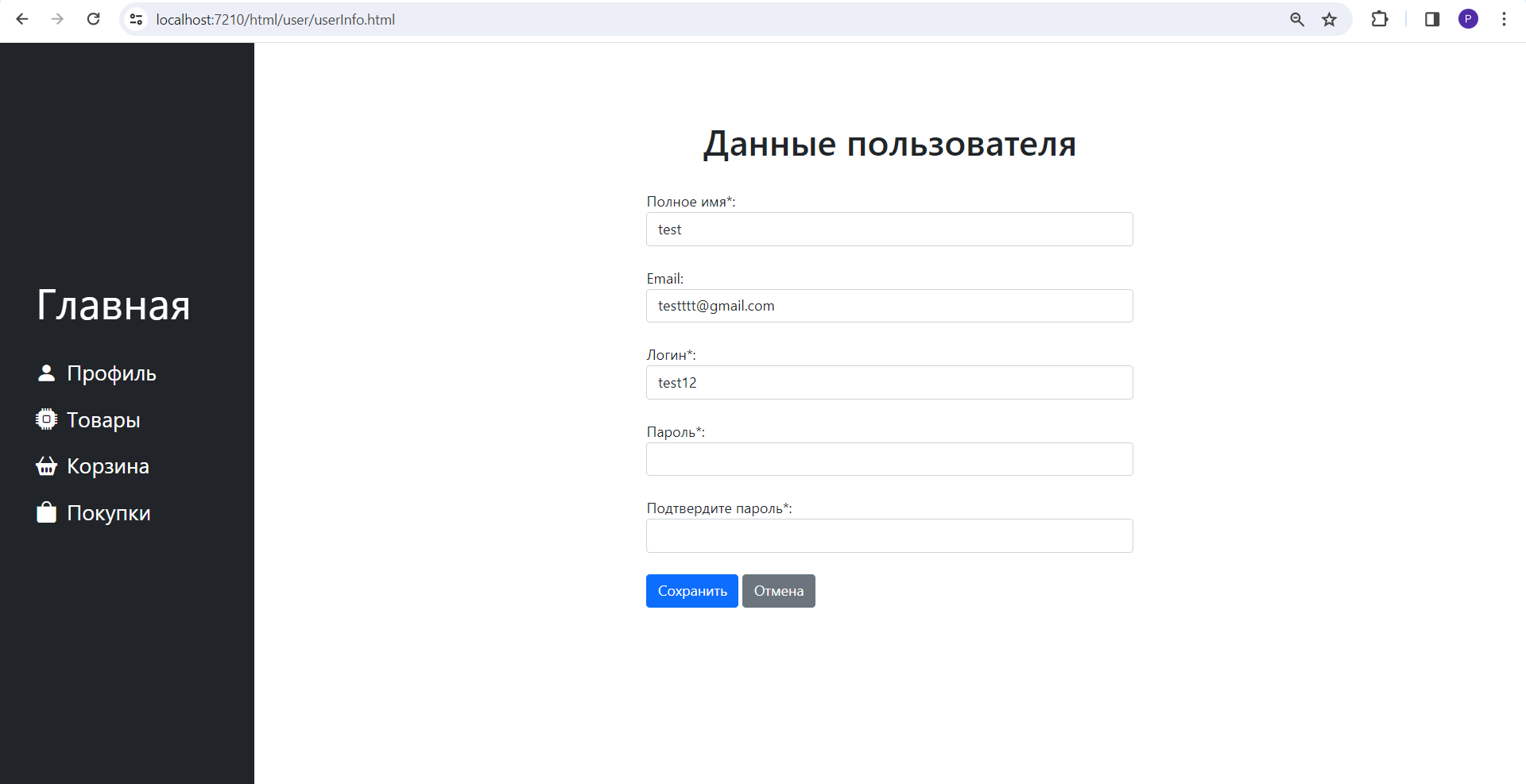


Рисунок 47 – страница профиля

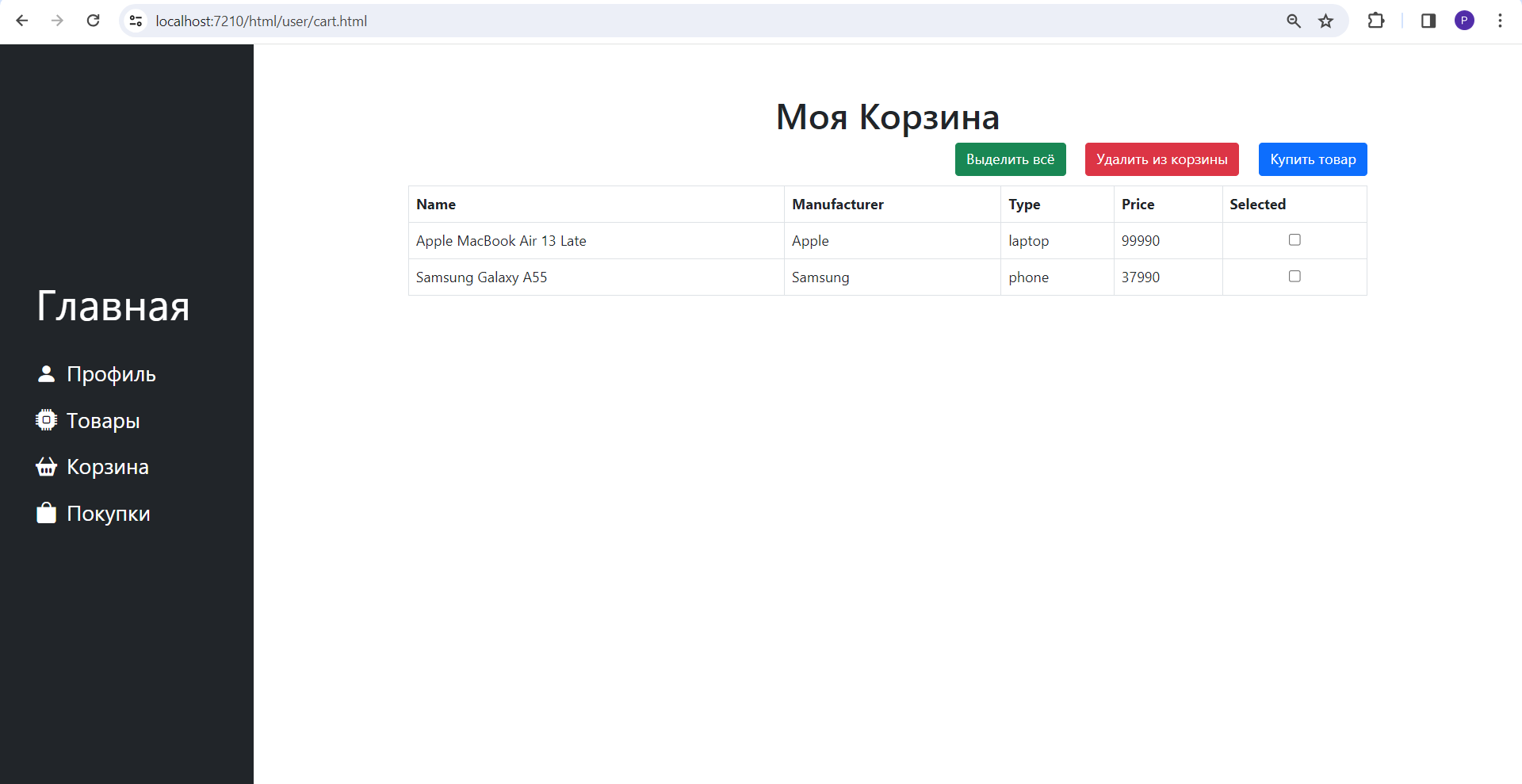


Рисунок 48 – страница корзины

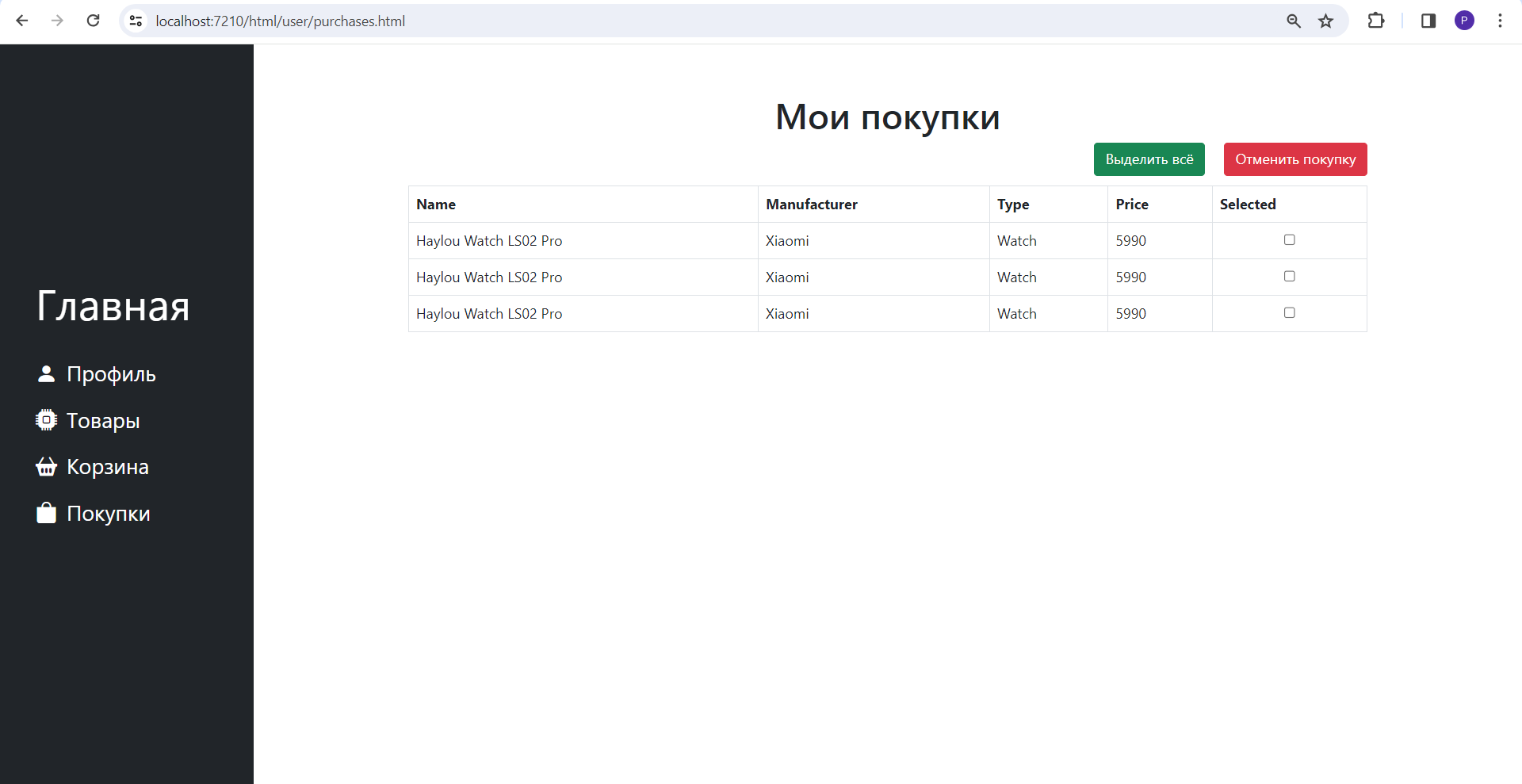


Рисунок 49 – страница покупок

# ЗАКЛЮЧЕНИЕ

Итак, была создана информационная система “онлайн магазин электронной техники”, основанная на кроссплатформенных WEB технологиях, которая предоставляет функциональность для покупки электронной техники онлайн.

При реализации проекта использовались средства разработки от Microsoft Visual Studio для создания API на платформе ASP.NET Core. Кроме того, в проекте применялась система управления базами данных SQL Server 2019 Express. В рамках проекта были приобретены навыки в проектировании WEB API, работе с фреймворком ASP.NET Core и Fetch API.

В результате, поставленная цель на проект была достигнута.

# Библиографический список

1. Официальная документация Entity Framework [Электронный ресурс].

URL: [https://learn.microsoft.com/ru-ru/ef/](https://learn.microsoft.com/ru-ru/ef/%20)

1. Официальная документация ASP.NET Core [Электронный ресурс].

URL: [https://learn.microsoft.com/ru-ru/ef/](https://learn.microsoft.com/ru-ru/ef/%20)

1. Официальная документация API Fluent [Электронный ресурс].

URL:<https://learn.microsoft.com/ru-ru/ef/ef6/modeling/code-first/fluent/relationships>

1. Официальная документация Fetch API [Электронный ресурс].

URL: [https://developer.mozilla.org/en-US/docs/Web/API/Fetch\_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API%20)

1. JWT Аутентификация в ASP.NET [Электронный ресурс].

URL: <https://dev.to/fabriziobagala/jwt-authentication-in-aspnet-13ma>