Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет   
имени первого Президента России Б.Н. Ельцина»

Институт новых материалов и технологий

Кафедра «Теплофизика и информатика в металлургии»

**Разработка веб-парсера**

**ОТЧЕТ**

**по практической работе №4**

**по дисциплине «Теория информационных процессов и систем»**

Направление 09.03.02 «Информационные системы и технологии» (уровень бакалавриата)

Образовательная программа  
09.03.02/33.02 «Информационные системы и технологии» (СУОС)

Студент

группы НМТ-303902 Е.Н. Кормина

Преподаватель:

профессор, д.т.н. И.А. Гурин

Екатеринбург

2022

**Задание**

Разработать систему для парсинга цен веб-сайта: https://shop.roca.ru

Система должна включать веб-интерфейс для добавления товаров.

Для каждого товара допустимо указать ссылку на соответствующем ресурсе. Запуск парсера осуществляется по кнопке или по таймеру.

При запуске парсера выполняется обход всех ссылок и все цены записываются в базу данных. Пользователю на интерфейсе выводится наименьшая цена по каждому товару и магазин, в котором зафиксирована цена.

Результат представить в виде ссылки на репозиторий и отчета в формате Word. Реализацию отдельных этапов работы выделить в отдельные Commit.

**Ход работы**

Для начала создадим проект. В Visual Studio создаем «Веб-приложение ASP.NET Core (модель-представление-контролер)».

Далее нужно добавить зависимости. Щёлкаем по ним правой кнопкой мыши и выбираем «Управление пакетами NuGet».

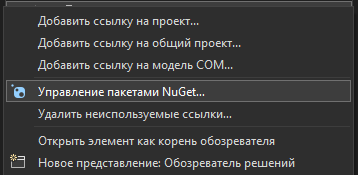


Рисунок 1 - Добавление зависимостей

Здесь нам нужно найти необходимые пакеты, они представлены на рисунке 2.

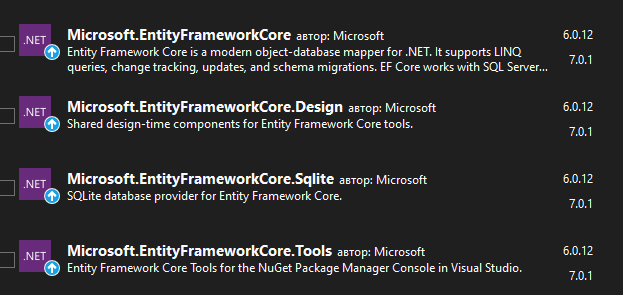


Рисунок 2 - Пакеты необходимые для работы

После этого в файле appsettings.json добавляем строки:

"ConnectionStrings": {

"DefaultConnection": "Data Source=PricesOfProducts.db;"

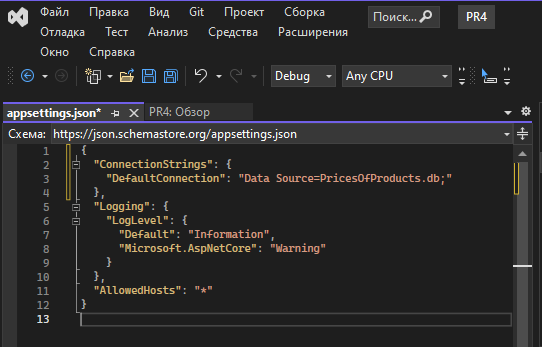
 },

Рисунок 3 - Полученный код appsettings.json

В program.cs прописываем следующее:

var connectionString = builder.Configuration.GetConnectionString("DefaultConnection");

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

options.UseSqlite(connectionString));

Полученный код представлен на рисунке 4.

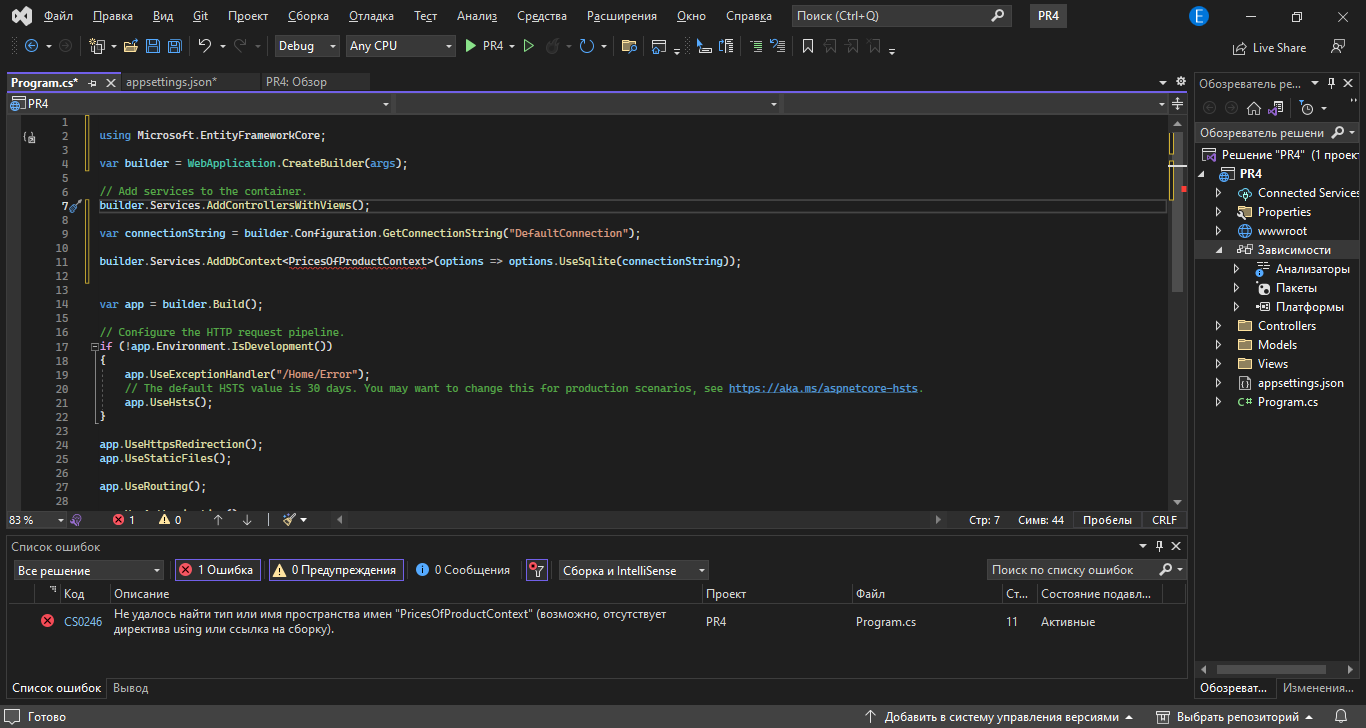


Рисунок 4- Полученный код program.cs

Создаём папку «Data». Там будут храниться данные. Для этого нажимаем на проект и выбираем «Добавить» и «Создать папку».

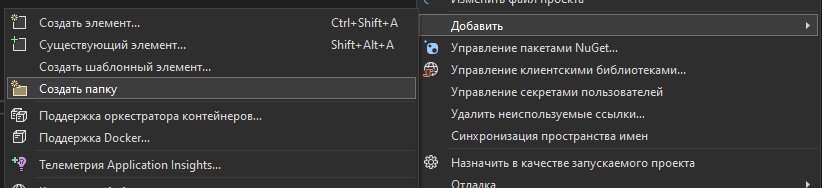


Рисунок 5 - Добавление папки

В этой папке создаём класс «PricesOfProductContext.cs» и прописываем код (рисунок 6).

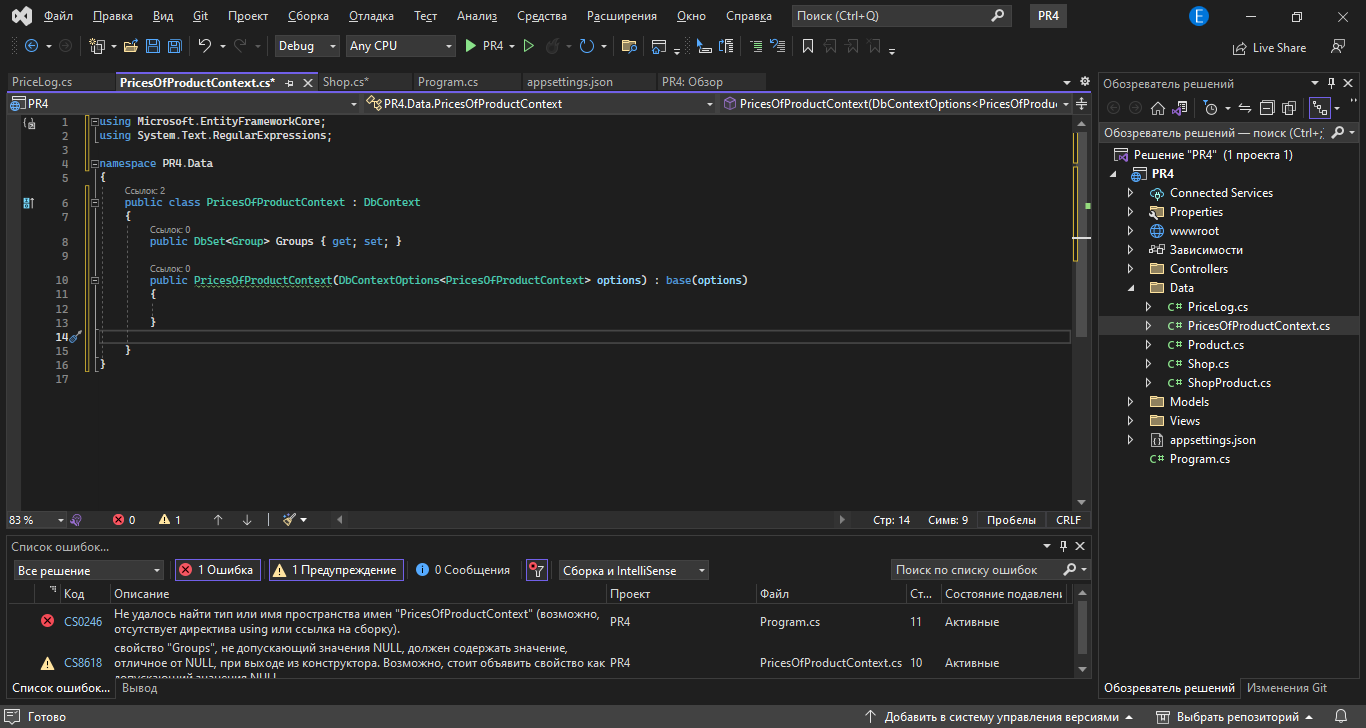


Рисунок 6 - Исходный код PricesOfProductContext.cs

В коде DbSet – это таблицы. Нам так же нужно добавить их в папку «Data» как классы. Нам потребуется 4 таблицы:

Shop – таблица с магазинами ;  
Product – таблица с товарами;

ShopProduct – таблица с ссылками на товары;

PriceLog – таблица с логированием цен (в ней хранятся ключи «Дата и время», «Товар» и «Магазин», а также «Цена»).

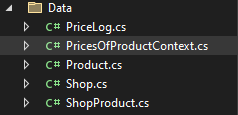


Рисунок 7 - Созданные таблицы

Теперь каждую таблицу нужно заполнить. Заполненные таблицы представлены на рисунках 8–11.

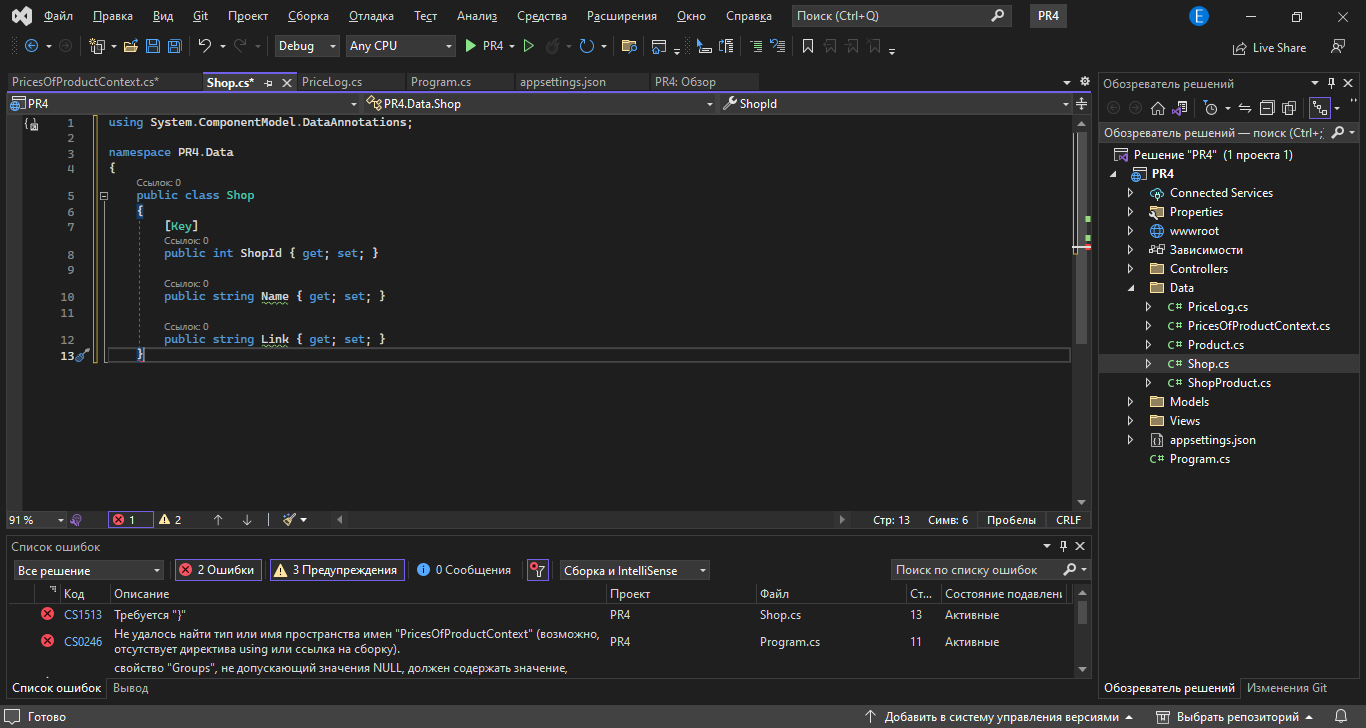


Рисунок 8 - Таблица Shop

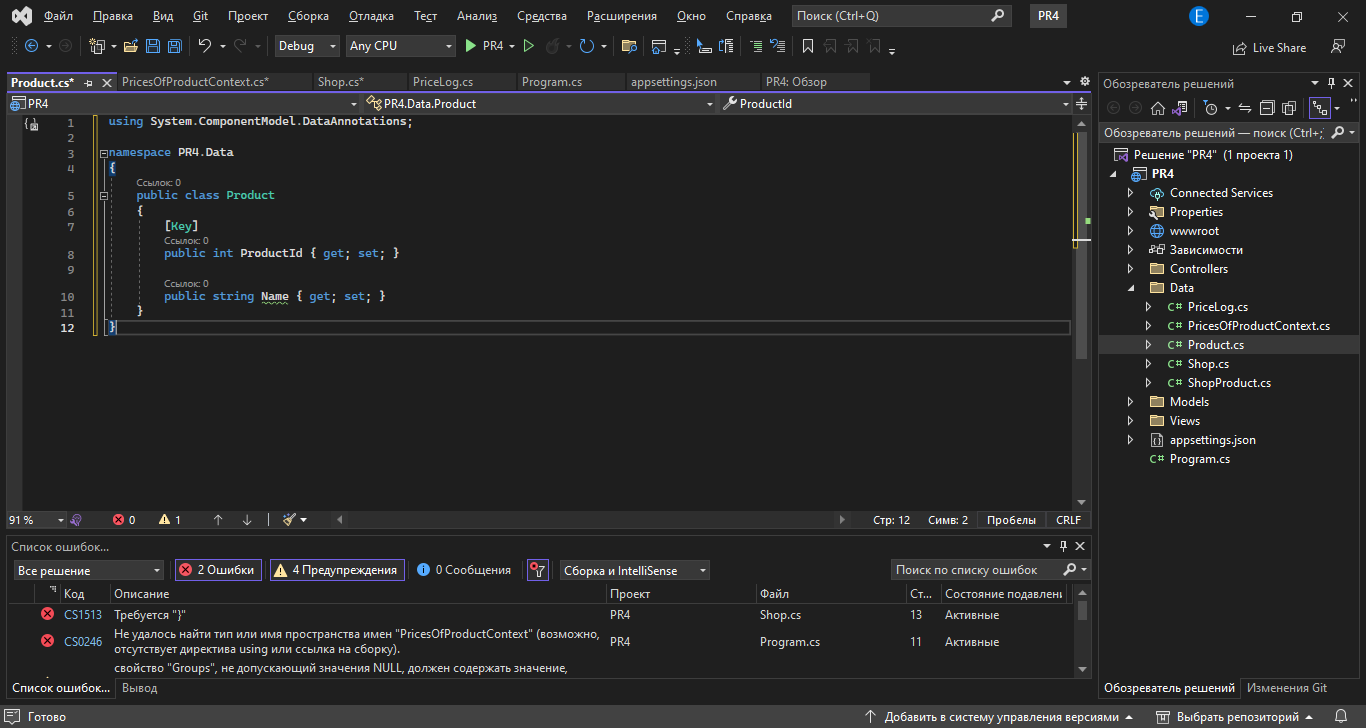


Рисунок 9 - Таблица Product

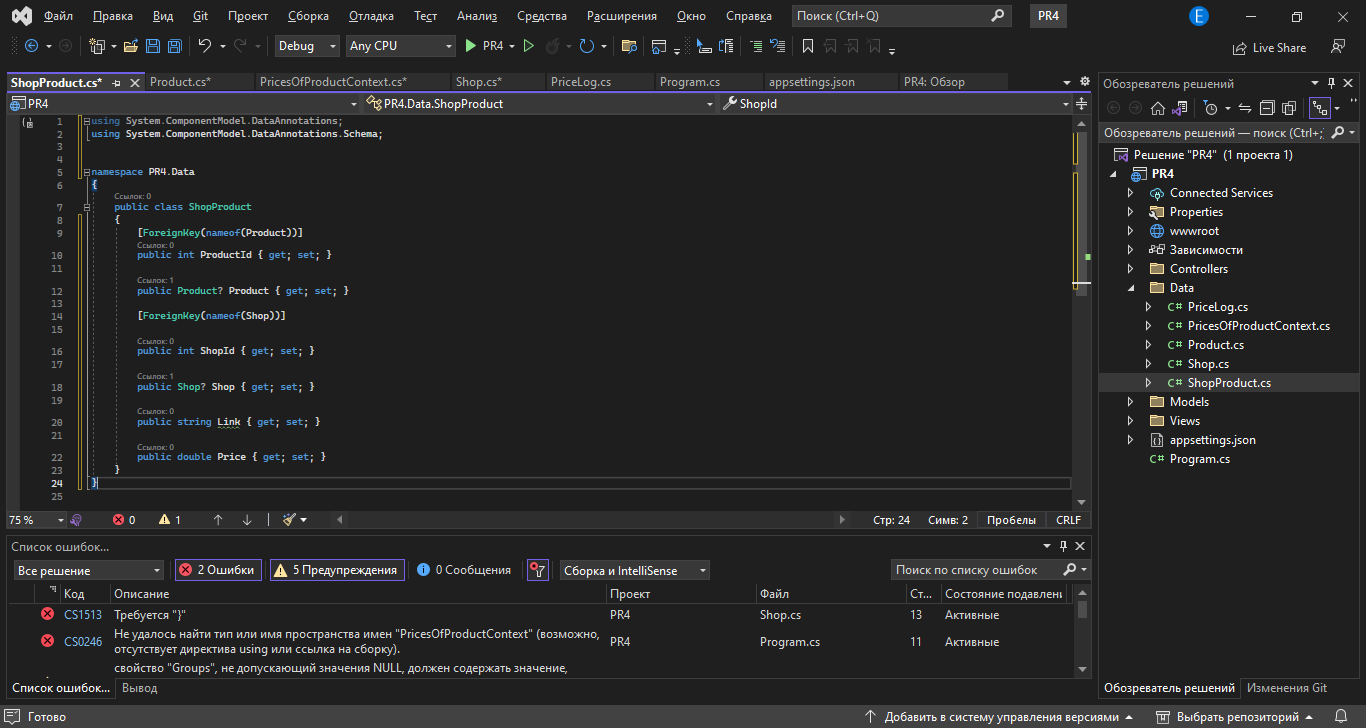


Рисунок 10 - Таблица ShopProduct

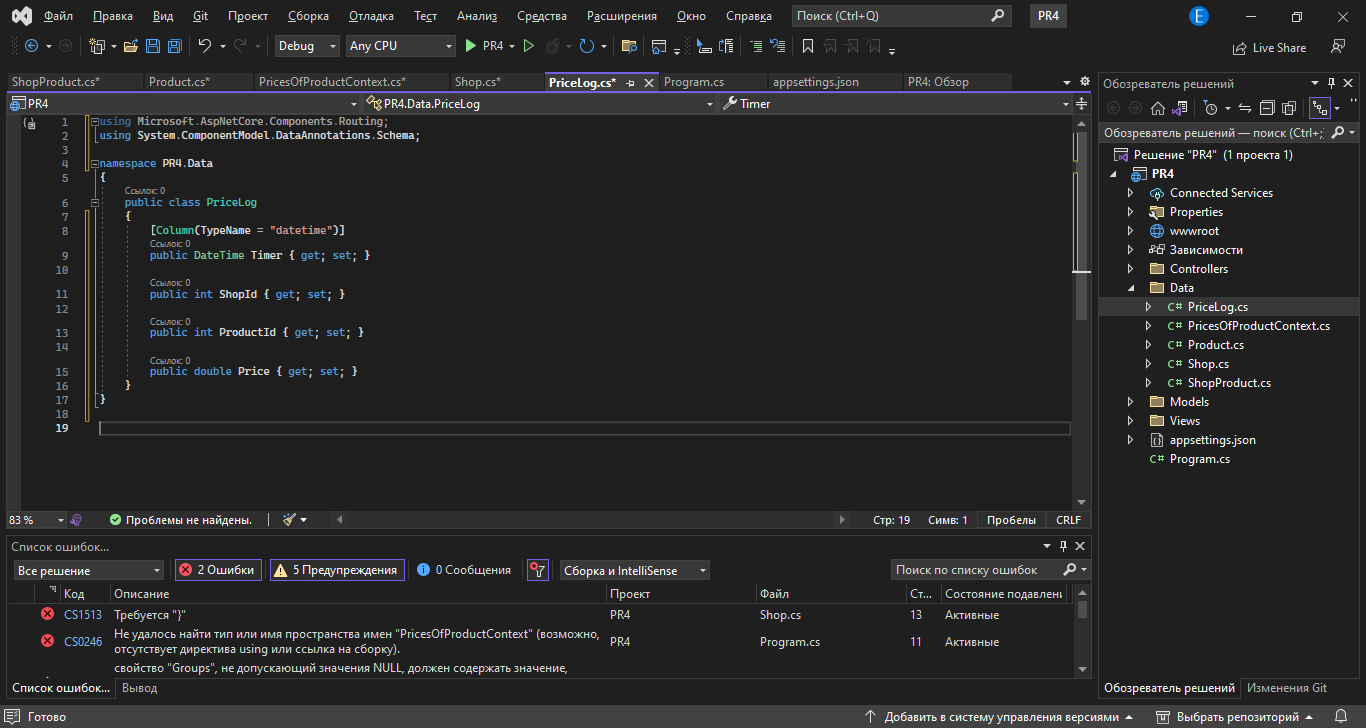


Рисунок 11 - Таблица PriceLog

Так как таблицы ShopProduct и PriceLog имеют составные ключи, это нужно указать в PricesOfProductContext. Туда же нужно добавить все созданные таблицы (Рисунок 12).

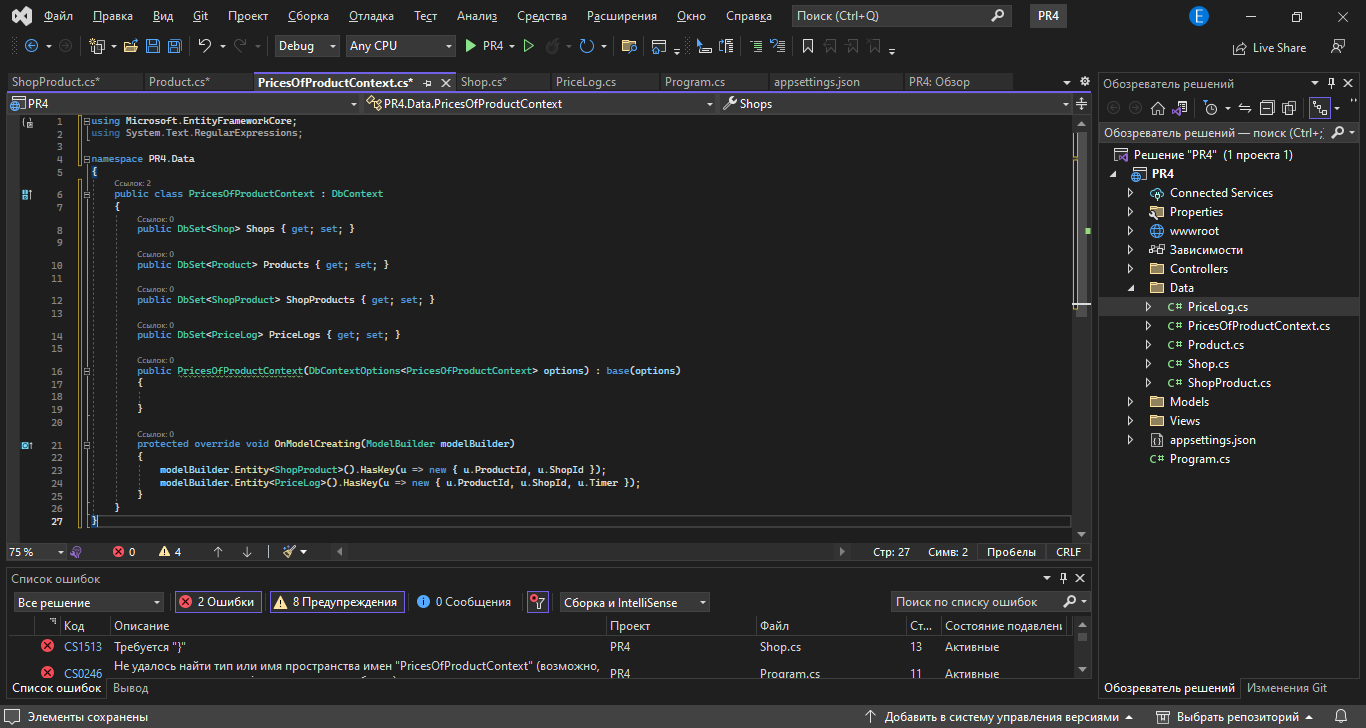


Рисунок 12 - Обновлённый код PricesOfProductContext.cs

Далее переходим в консоль диспетчера пакетов, вводим команду add-migration mig\_init и следом команду update-database. Видим, что добавились следующие файлы (Рисунок 13).

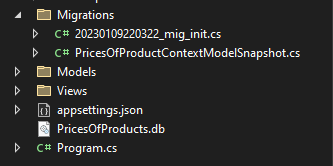


Рисунок 13 - Получившиеся файлы

Теперь добавим контроллеры. Кликаем правой кнопкой мыши по папке «Controllers» – «Добавить» – «Контроллер».

Здесь нас интересует «Контроллер MVC с представлениями, использующий Entity Framework» (рисунок 14). Нажимаем «Добавить». Так нужно сделать для всех таблиц.

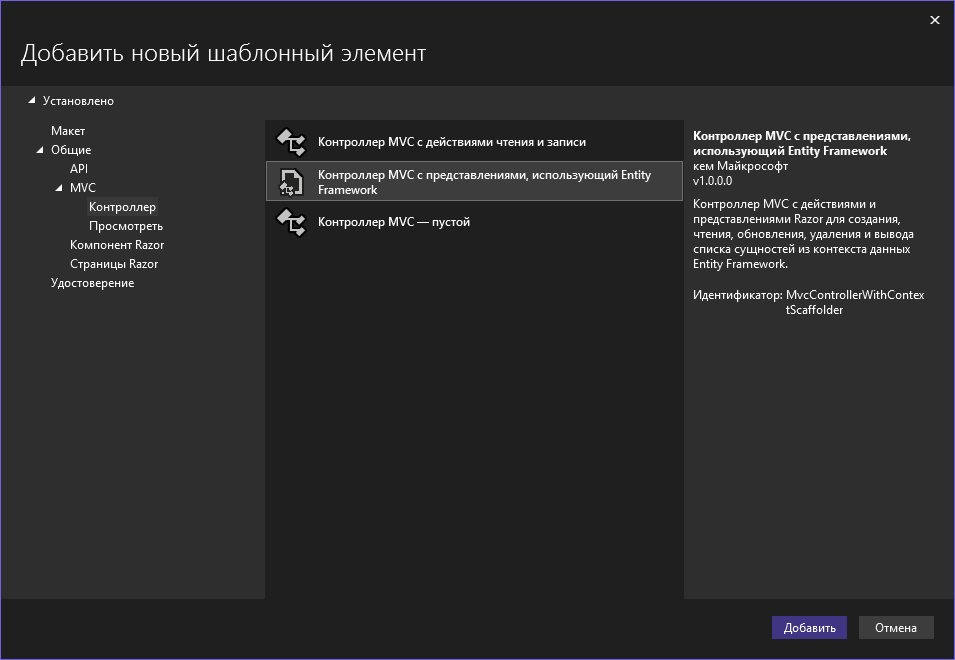


Рисунок 14 - Добавление контроллера

Далее задаём нужные параметры и нажимаем «Добавить» (рисунок 15).

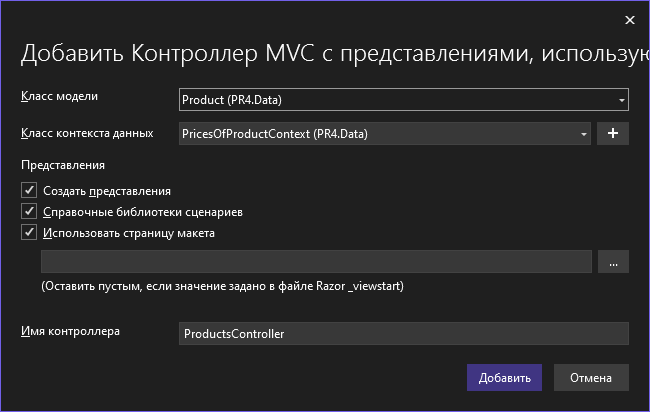


Рисунок 15 - Настройка контроллера

В итоге у нас получаются контроллеры представленные на рисунке 16.

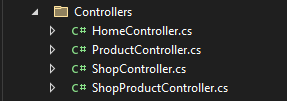


Рисунок 16 - Список контроллеров

Далее запускаем созданный контроллер. Браузер открывает страницу нашего приложения (рисунок 17). Теперь нужно ввести https://localhost:7069**/Shop** и добавить нужные сайты.

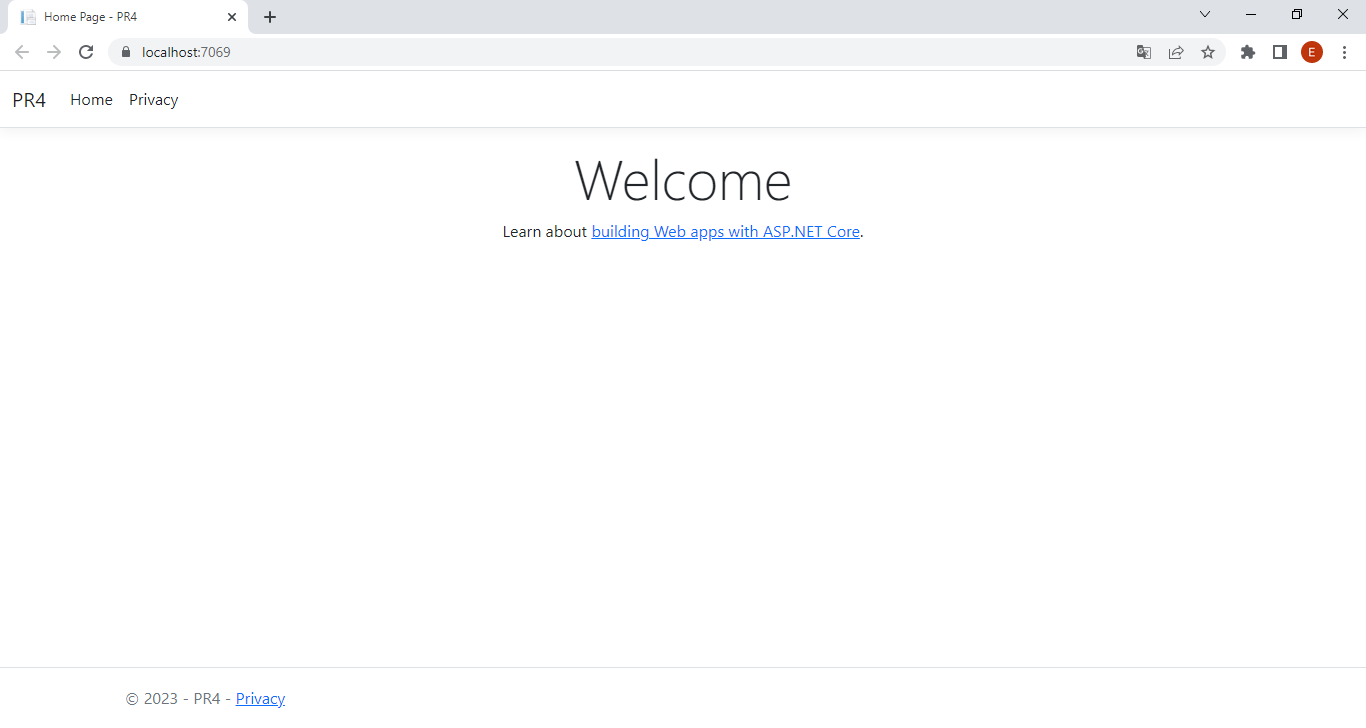


Рисунок 17 – страница нашего приложения

Создав контроллер для товаров, нам нужно ссылать их на конкретный товар, а не на ID товара. Для этого в коде ShopProduct.cs мы добавляем ForeignKey, который и будет ссылаться на название.

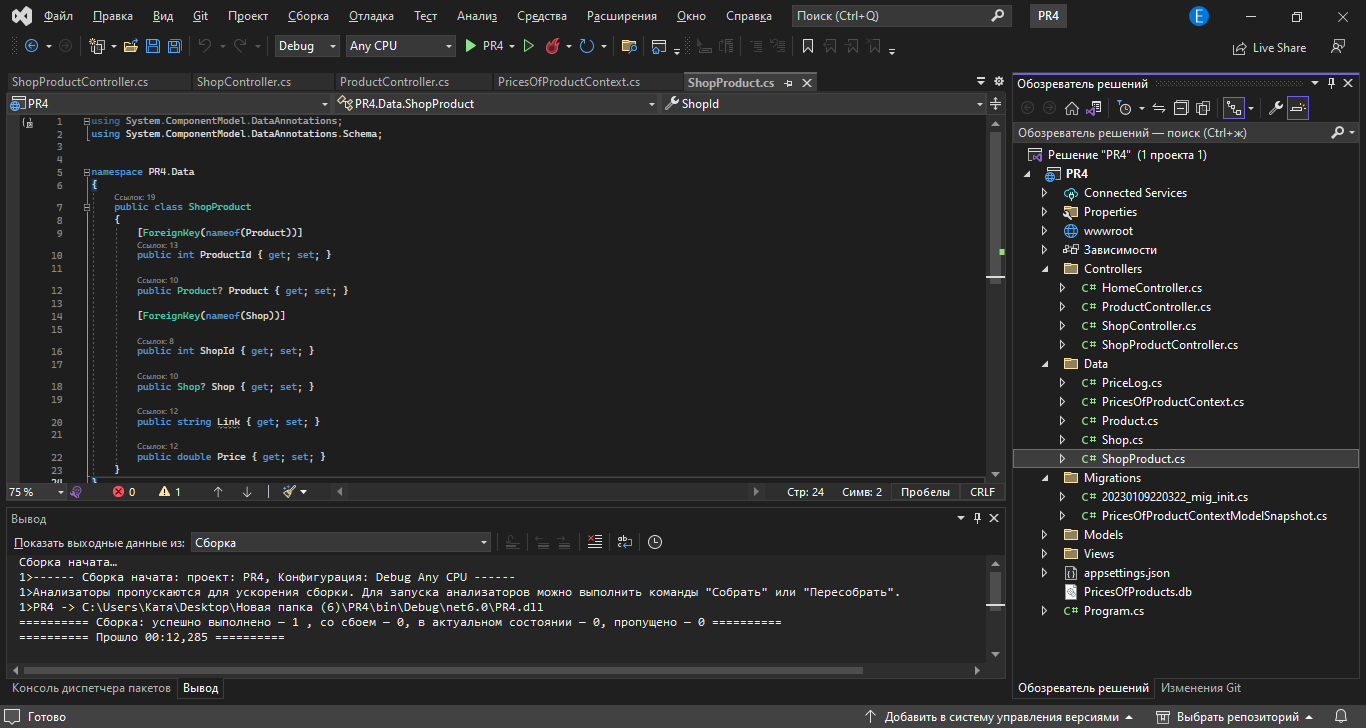


Рисунок 18 - Ссылки на название

Переходим в папку Views. Далее в папку Shared и открываем код \_Layout.cshtml. Там меняем линки строки 23-29, чтобы мы могли переходить по ним для доступа к нужным таблицам. Это должно выглядеть так:

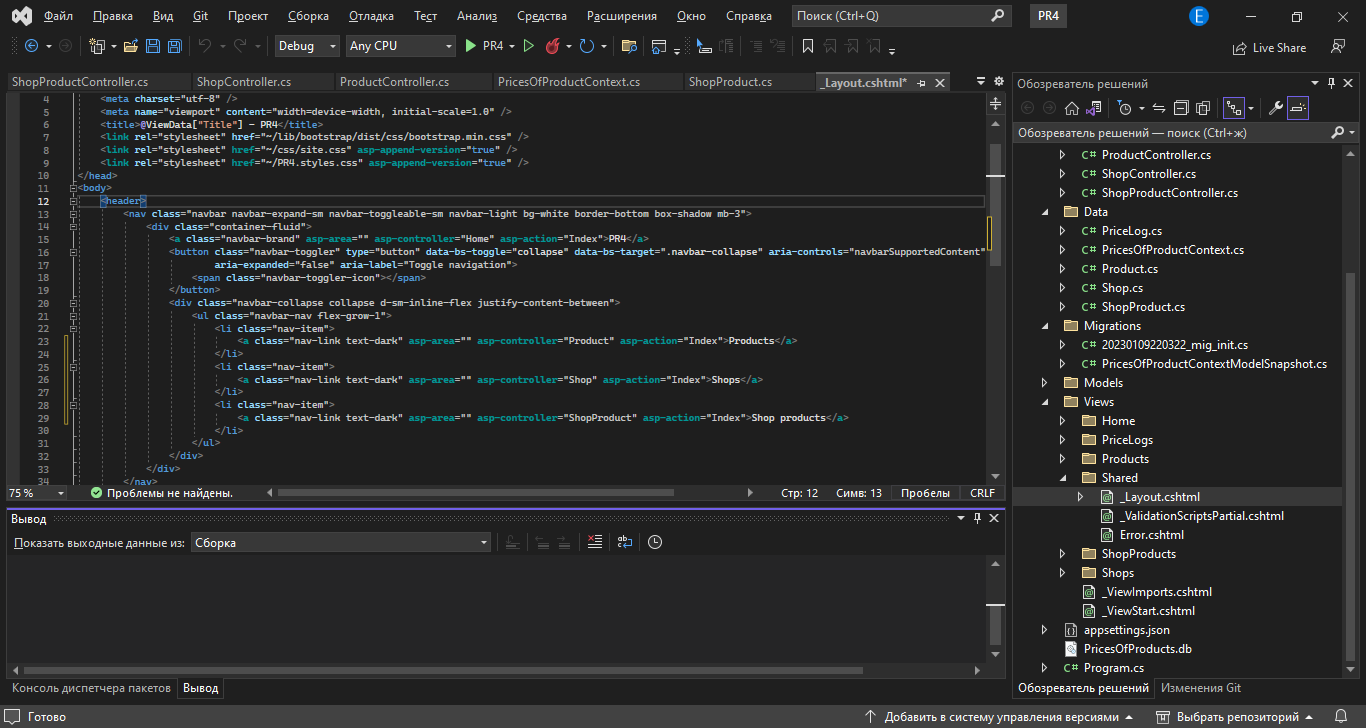


Рисунок 18 - Отображение линков

В контроллере ShopProduct в коде меняем значения ViewData для корректного отображения названий товаров и магазинов (чтобы отображались не ID, а само название). То есть мы заменяем ShopId на Name, аналогично для ProductId (рисунок 19).

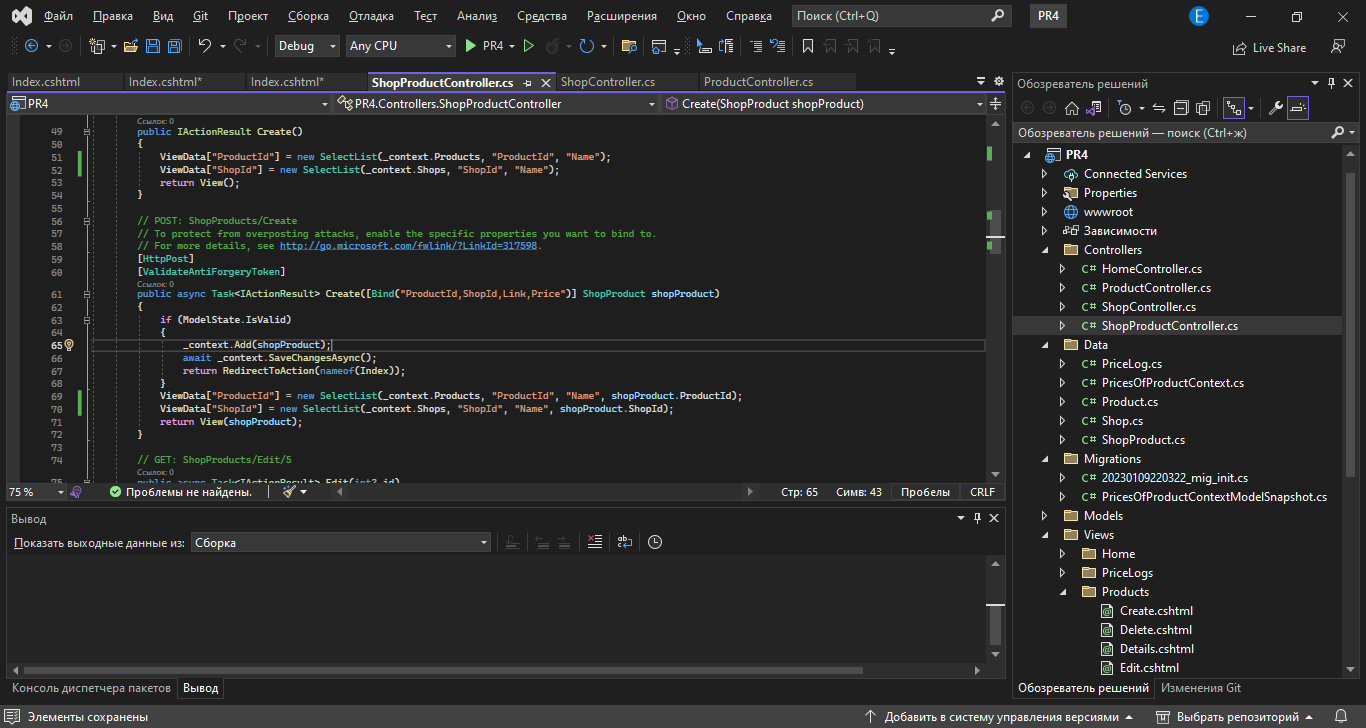


Рисунок 19 - Корректное отображение названий во время добавления

Чтобы на странице со ссылками на товары корректно отображались названия магазина и товара, нужно перейти в папку Views, далее ShopProduct, а затем открыть файл Index.cdhtml. Там меняем item.Product.ProductId на item.Product.Name. Аналогично поступаем с магазином.

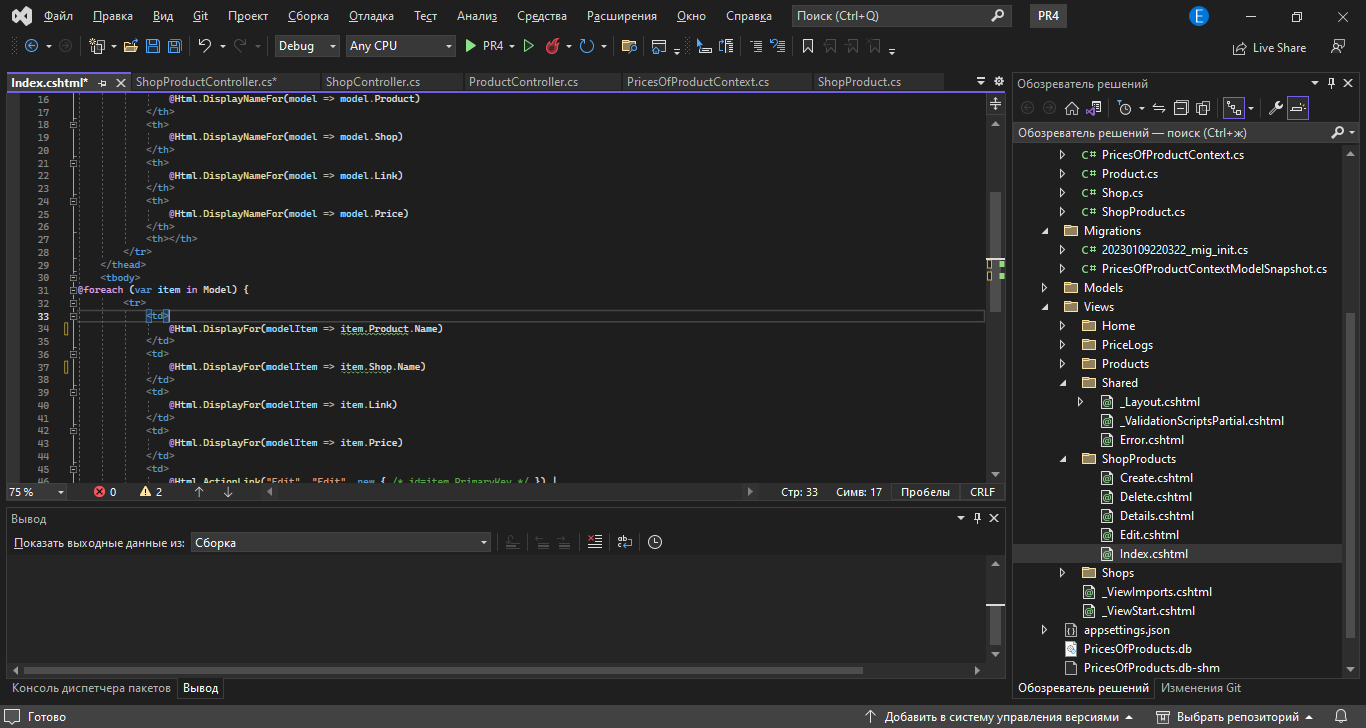


Рисунок 20 - Корректное отображение названий

Так же для лучшего визуала поменяем и название заголовка. Для этого меняем Index на Shop products. Так нужно проделать во всех файлах Index.cshtml для папок Shop и Product.

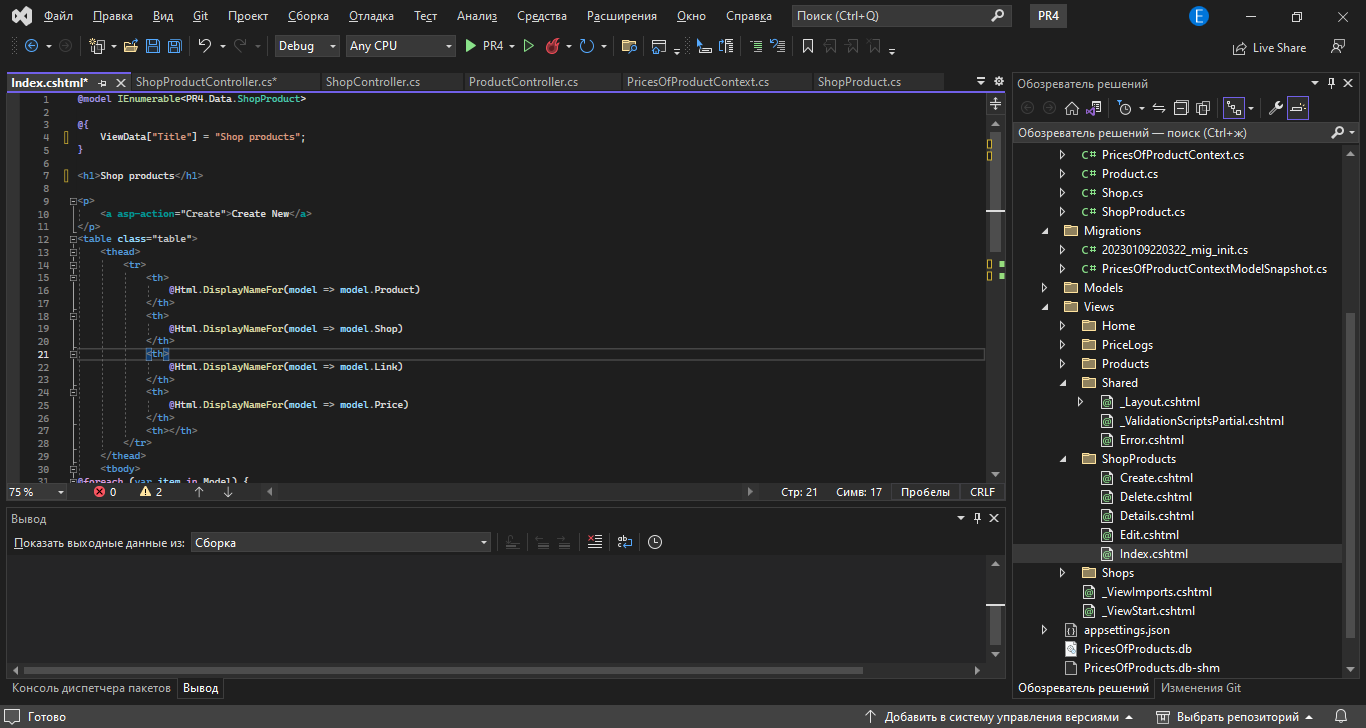


Рисунок 21 - Корректное отображение заголовка

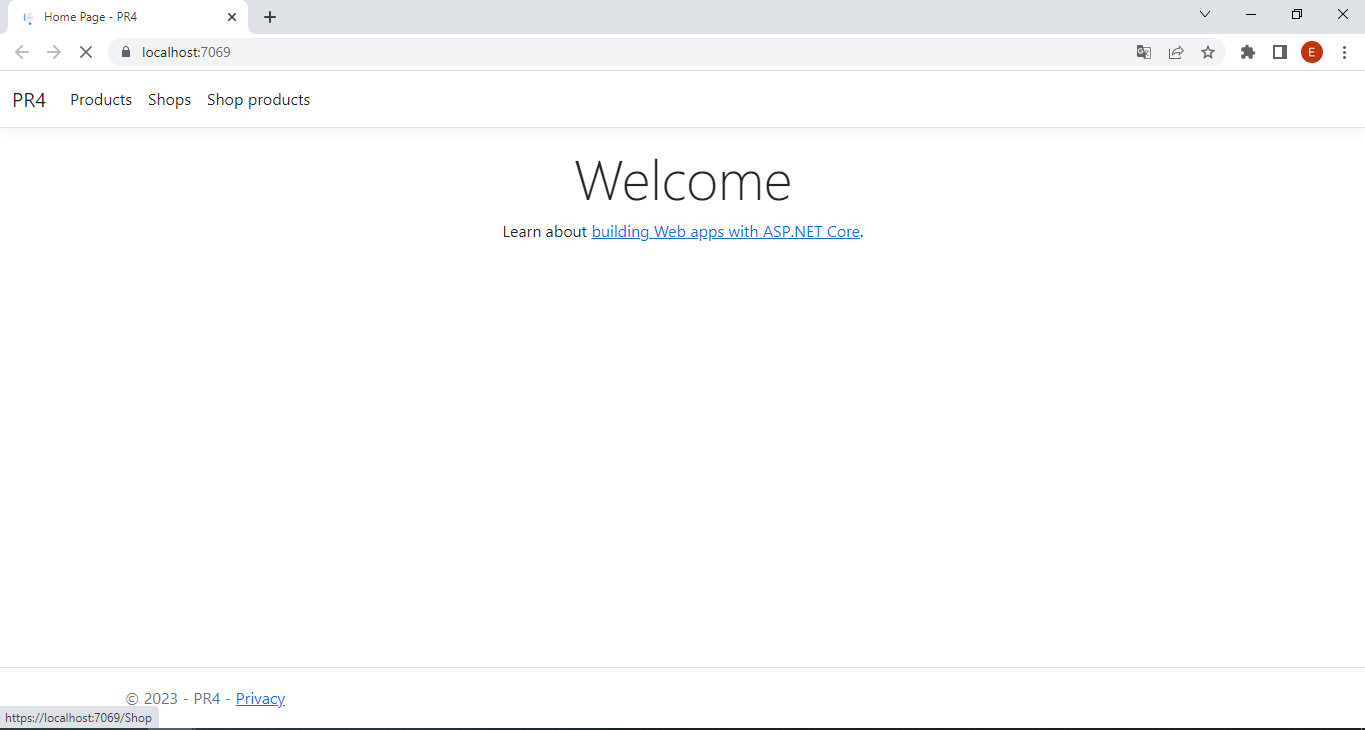


Рисунок 22 - [https://localhost:7069**/**Shop](https://localhost:7069/Shop)

Ссылка на репозиторий github: https://github.com/Kormina/PR4

Вывод: в данной практической работе мы изучили процесс парсинга цен. Поработали в Visual Studio, используя, «Веб-приложение ASP.NET Core (модель-представление-контролер)» и пакеты NuGet и создали свое приложение. Получили навыки парсинга. Парсинг будет полезен в современном мире с большим количеством информации. С его помощью можно упростить поиск и сбор информации, ее редактирование и выявление ошибок, а также парсинг поможет следить за ее изменениями и аткуальностью.