# ВВЕДЕНИЕ

В современном мире интернет-технологии играют ключевую роль в бизнесе, и создание эффективных веб-приложений становится необходимостью для компаний, стремящихся к успешному развитию. Одним из наиболее актуальных направлений в этой области является разработка веб-API, которые обеспечивают взаимодействие между различными системами и позволяют интегрировать функционал интернет-магазинов с другими сервисами.

Данная курсовая работа посвящена разработке веб-API для интернет-магазина DNS, одного из крупнейших ритейлеров электроники и бытовой техники в России. Целью работы является создание RESTful API, которое будет обеспечивать доступ к основным функциональным возможностям интернет-магазина, таким как управление товарами, обработка заказов, а также взаимодействие с пользователями.

В процессе работы будет рассмотрен процесс проектирования архитектуры API, выбор технологий и инструментов для его реализации, а также вопросы безопасности и производительности. Особое внимание будет уделено принципам REST, которые обеспечивают простоту и удобство использования API, а также позволяют легко интегрировать его с различными клиентскими приложениями.

Разработка веб-API для интернет-магазина DNS не только улучшит пользовательский опыт, но и создаст возможности для дальнейшего расширения функционала, интеграции с мобильными приложениями и сторонними сервисами. В результате, реализация данного проекта будет способствовать повышению конкурентоспособности интернет-магазина и улучшению качества обслуживания клиентов.

Таким образом, данная курсовая работа представляет собой важный шаг в изучении современных технологий веб-разработки и их применения в реальных бизнес-проектах, что делает ее актуальной и значимой в условиях быстро меняющегося цифрового мира.

# Глава 1. Теоретическая часть

В начале расскажу немного про само Web API. Мы разберем из чего он состоит, как его конфигурировать и для чего он нужен.

## Web API

Web API (Application Programming Interface) — это интерфейс, который позволяет различным программным приложениям взаимодействовать друг с другом через интернет. Web API предоставляет набор правил и протоколов, которые определяют, как запросы и ответы должны формироваться и обрабатываться, что позволяет разработчикам интегрировать различные системы и сервисы.

## Протоколы и Форматы данных

Web API обычно использует стандартные протоколы, такие как HTTP/HTTPS, для передачи данных. Это позволяет API быть доступными через веб.

Web API поддерживает различные форматы данных для обмена информацией, наиболее распространенными из которых являются JSON (JavaScript Object Notation) и XML (eXtensible Markup Language). JSON стал особенно популярным благодаря своей легкости и простоте.

## REST и SOAP

Существует несколько архитектурных стилей для создания Web API, среди которых наиболее известны:

REST (Representational State Transfer): Это архитектурный стиль, который использует стандартные HTTP-методы (GET, POST, PUT, DELETE) для выполнения операций над ресурсами. RESTful API обычно просты в использовании и хорошо масштабируемы.

SOAP (Simple Object Access Protocol): Это протокол, который использует XML для передачи сообщений. SOAP API более сложны и требуют строгого соблюдения формата сообщений, но обеспечивают более высокий уровень безопасности и надежности.

## Аутентификация и авторизация

Web API часто требует аутентификации пользователей для обеспечения безопасности. Это может быть реализовано с помощью различных методов, таких как OAuth, JWT (JSON Web Tokens) и API-ключи.

## Документация

Хорошо спроектированные Web API обычно сопровождаются документацией, которая описывает доступные эндпоинты, параметры запросов, форматы ответов и примеры использования. Это помогает разработчикам быстро интегрировать API в свои приложения.

# Глава 2. Практическая часть

В данной части, мы будем писать API которая позволит нам сохранять, добавлять и обновлять данные интернет-магазина. В прошлой части мы разобрались, что такое Web API и из чего он состоит.

## Создание проекта

Для начала необходимо создадать проект. Для создания и написания backend мы будем использовать одну из популярных IDE Visual Studio 2022.

В начале открываем IDE(Рисунок 2.1.1).

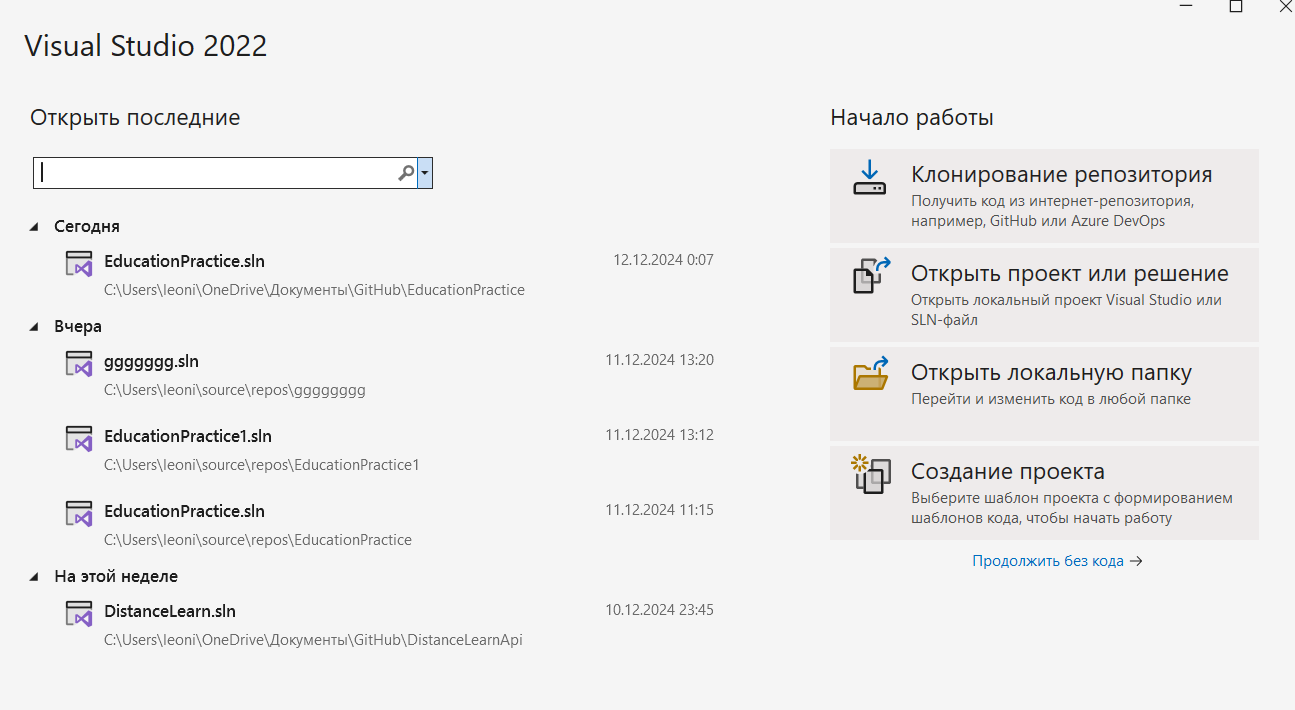


Рисунок 2.1.1 – Visual Studio 2022

Далее создадим проект. Выбираем тип проекта “Пустой шаблон ASP.NET Core (Майкрософт)”, указываем название и расположение проекта на компьютере(Рисунок 2.1.2).

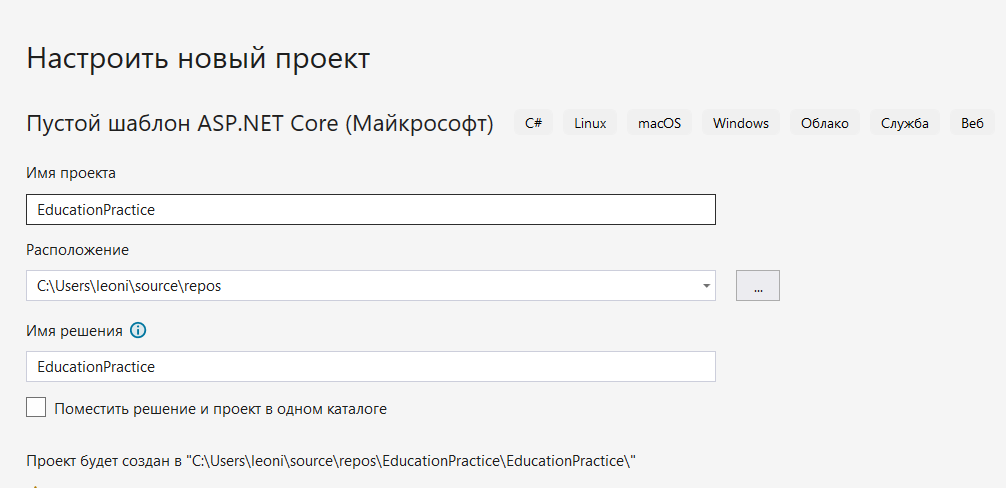


Рисунок 2.1.2 – настройка нового проекта

Затем необходимо сконфигурировать наше приложение указав дополнительные сведения при создании(Рисунок 2.1.3).

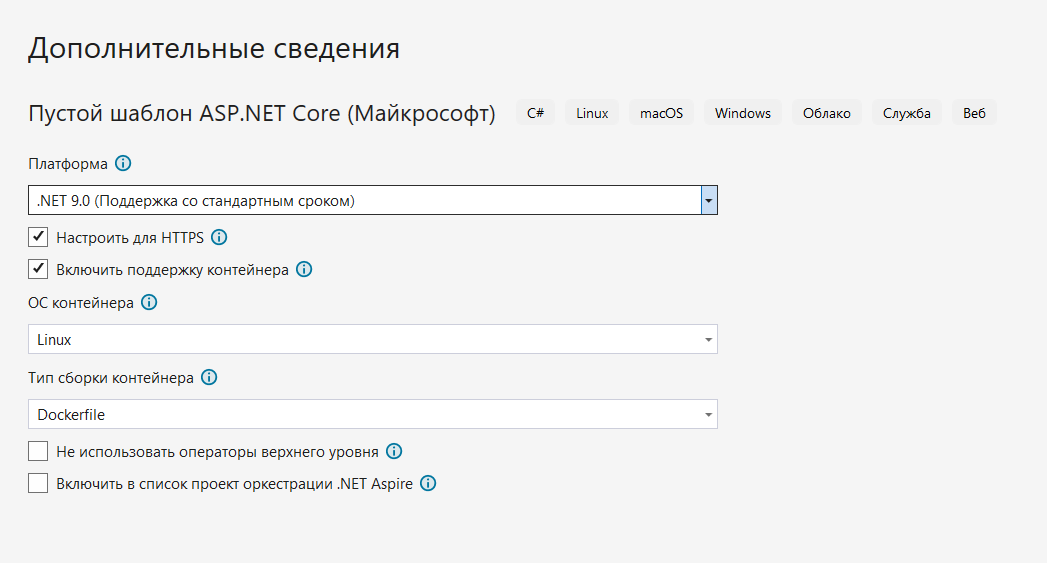


Рисунок 2.1.3 – указание дополнительных настроек приложения

По итогу у нас должно открыться следующее окно, указывающее на успешное создание проекта(Рисунок 2.1.4).

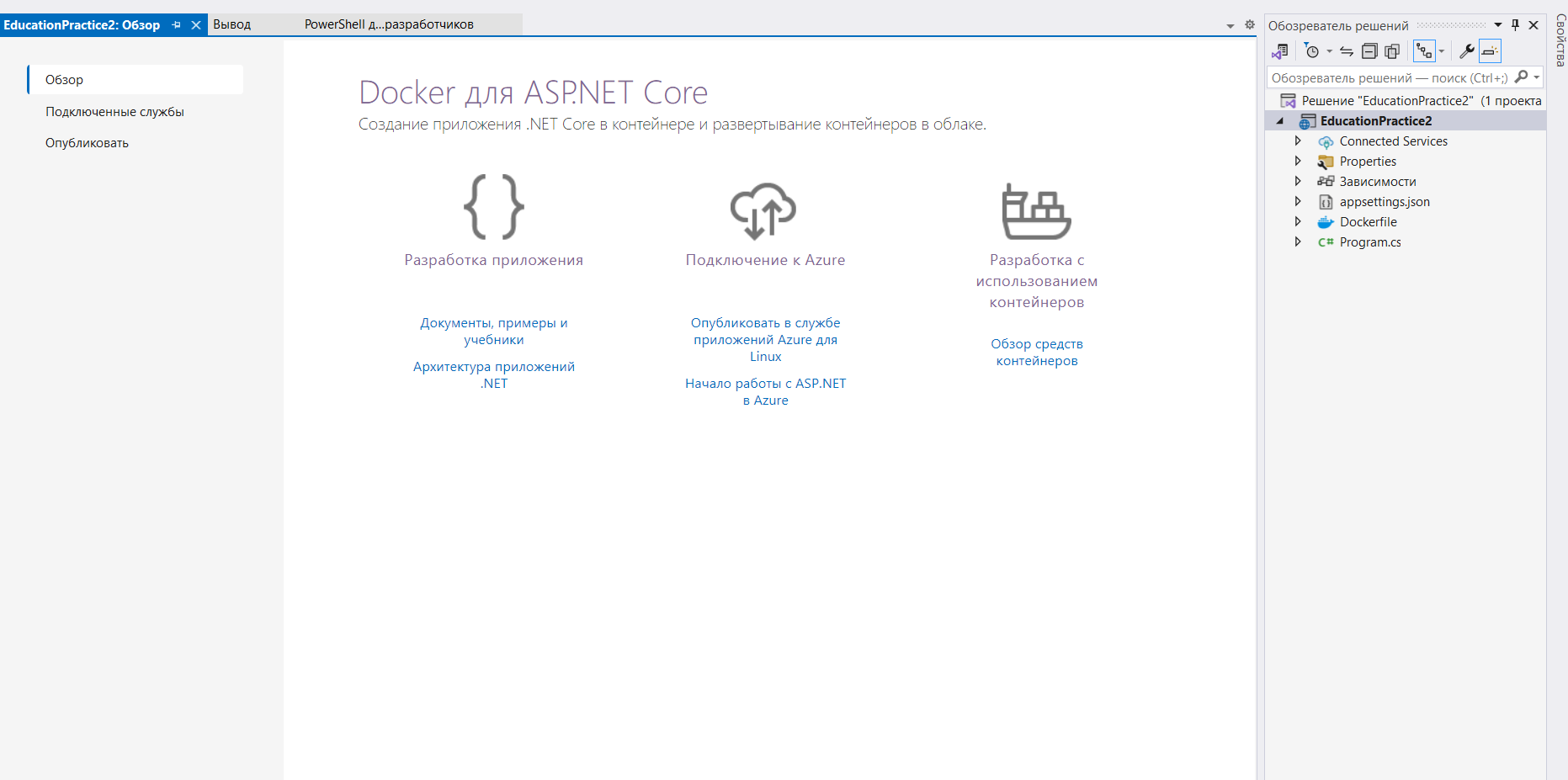


Рисунок 2.1.4 – окно успешно созданного проекта

## Загрузка библиотеки

Проект создан, теперь необходимо загрузить библиотеку, чтобы мы смогли работать с базой данных.

В проектах для .NET есть удобный пакетный менеджер. Называется Nuget. При помощи него мы можем найти практически любую библиотеку созданную для .NET.

Вообще существует несколько способов открытия Nuget в IDE Visual Studio 2022, будем использовать самый простой. Чтобы открыть Nuget, необходимо выполнить следующие шаги. Необходимо найти в обозревателе решений, нужно вам решение, затем нажатв правой кнопкой мыши по решению проекта, выбрать «Управление пакетами Nuget для решения…»(Рисунок 2.2.1).

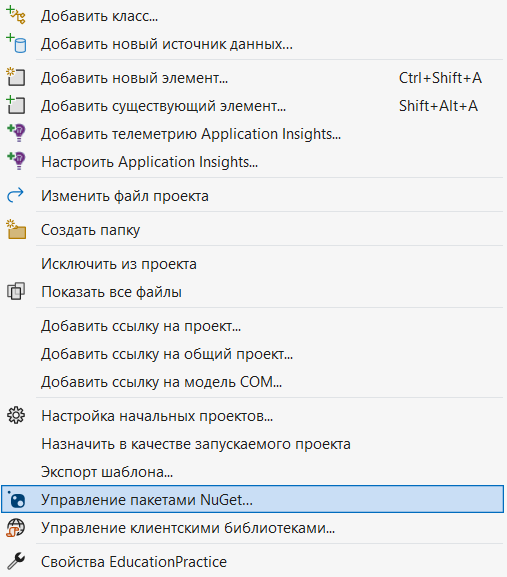


Рисунок 2.2.1 – Открытие Nuget в Visual Studio

После нажатия на кнопку откроется новая вкладка, и первое, что вы увидите это все пакеты установленные в проекте(Рисунок 2.2.2). Но так как, пакетов в нашем проекте не установлено вообще, появится надпись «Пакеты не установлены».

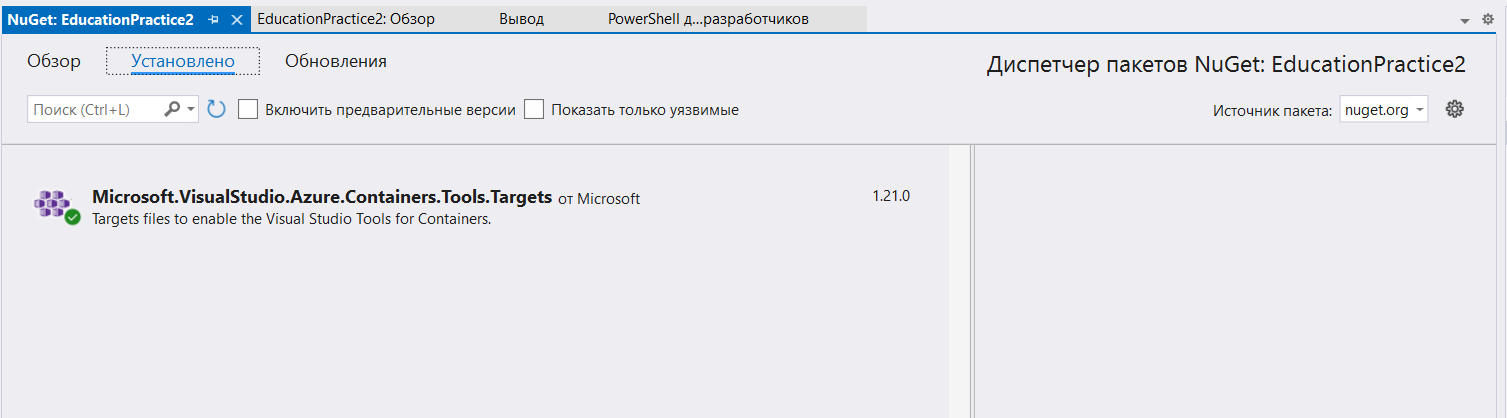


Рисунок 2.2.2 – Вкладка с установленными пакетами для решения.

Следом откроем раздел «Обзор» и в поиске найдем пакет «Microsoft.EntityFrameWorkCore»(Рисунок 2.2.3).

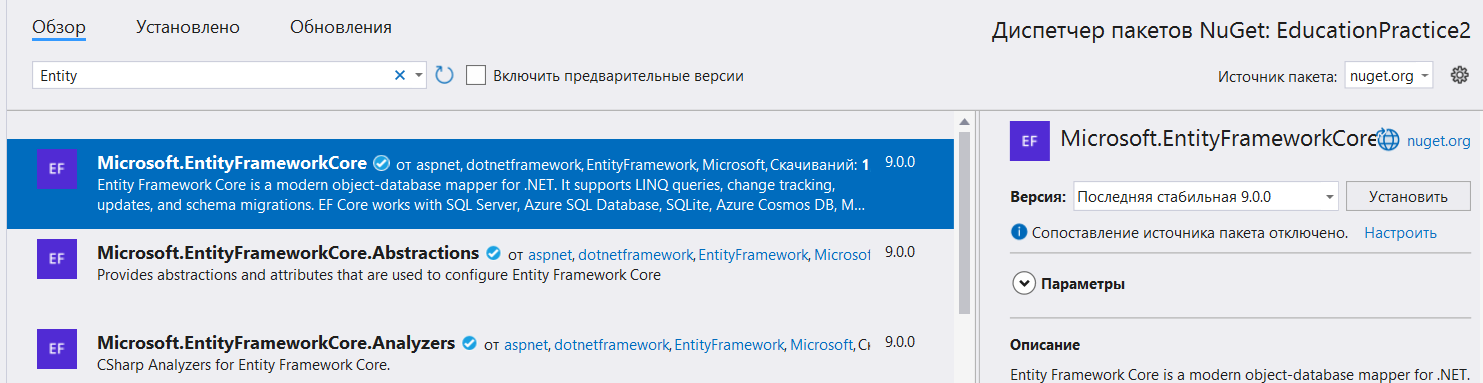


Рисунок 2.2.3 – Поиск MicrosoftEntityFrameWorkCore пакета

Затем просто кликаем по кнопке установить и просто везде нажимаем «Применить»(Рисунок 2.2.4 – 2.2.5).

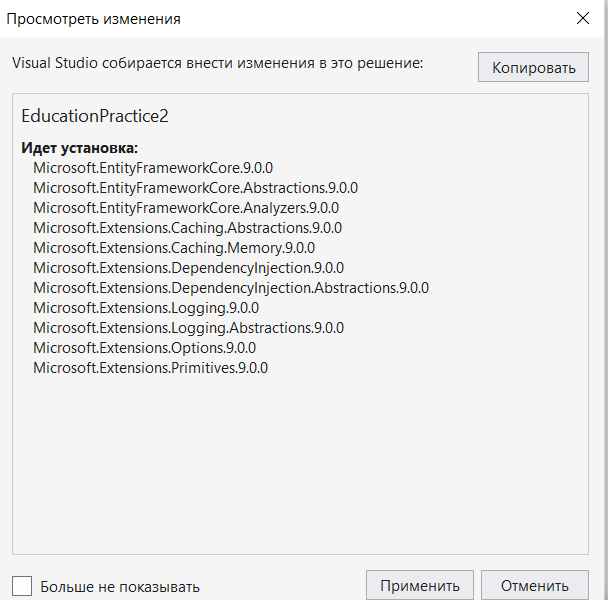


Рисунок 2.2.4 – Установка пакета

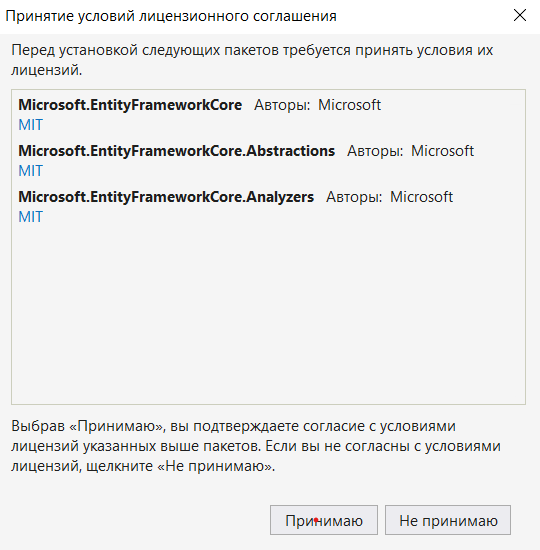


Рисунок 2.2.5 – принятие условий лицензионного соглашения

Также по мимо «Microsoft.EntityFrameWorkCore», нам также необходимо установить следующий пакет: Microsoft.EntityFrameWorkCore.Tols и Server Данный пакет позволит нам настроить и использовать базу данных при работе с нашим приложением.

## Конфигурация приложения

После успешной установки пакетов можно настроить наше приложение, чтобы оно могло использовать БД.(Рисунок 2.3.1).

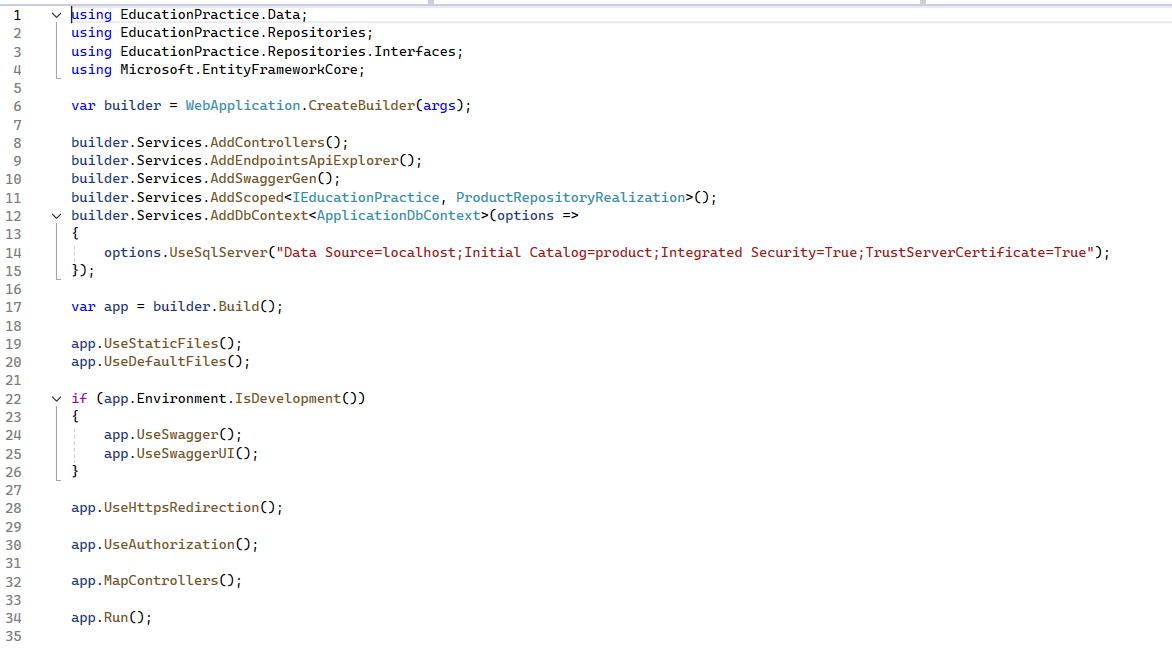


Рисунок 2.3.1 – Конфигурация Program.cs

На рисунке выше мы объвляем и подключаемся к БД, а также добавили «Swagger» для того, чтобы мы могли управлять базой данных.

## Создание БД.

После конфигурации мы реализуем саму базу данных

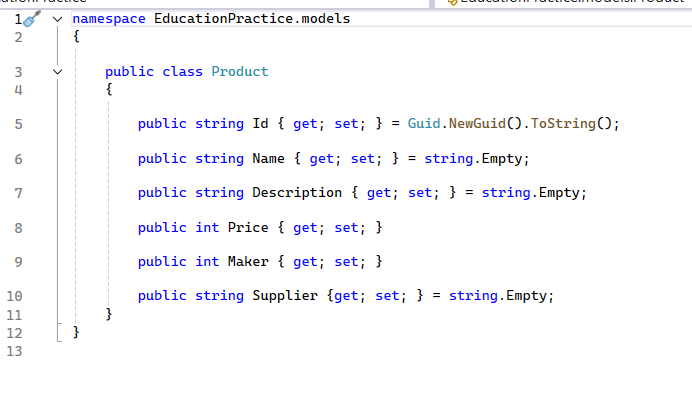


Рисунок 2.4.1 – Добавление столбцов

Следующим шагом нам необходимо реализовать репозиторий и интерфейс (Рисунок 2.4.2).

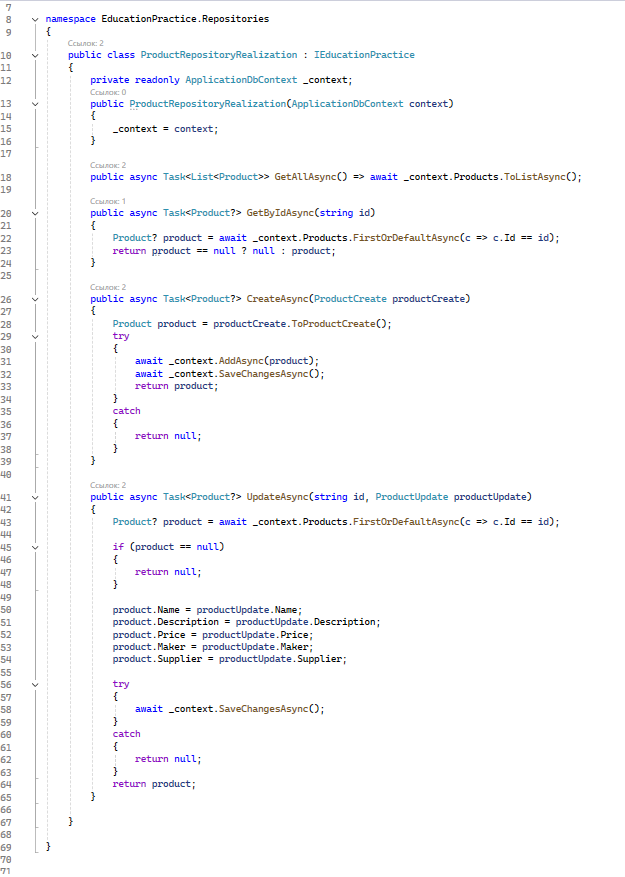


Рисунок 2.4.2 – Репозиторий

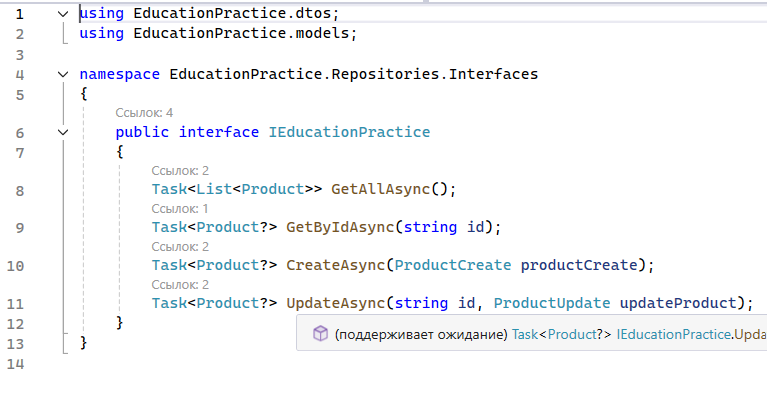


Рисунок 2.4.3 – Интерфейс

## Создание dtos и контроллера и AppDb

Следующим нашим шагом будет создание dtos. DTO (Data Transfer Object) — это паттерн проектирования, который используется для передачи данных между слоями приложения, особенно в контексте веб-приложений и API. DTO представляют собой простые объекты, которые содержат только данные и не включают в себя бизнес-логику или поведение. Они служат для упрощения передачи данных между клиентом и сервером, а также между различными компонентами приложения.(Рисунок 2.5.1. , 2.5.2, 2.5.3).

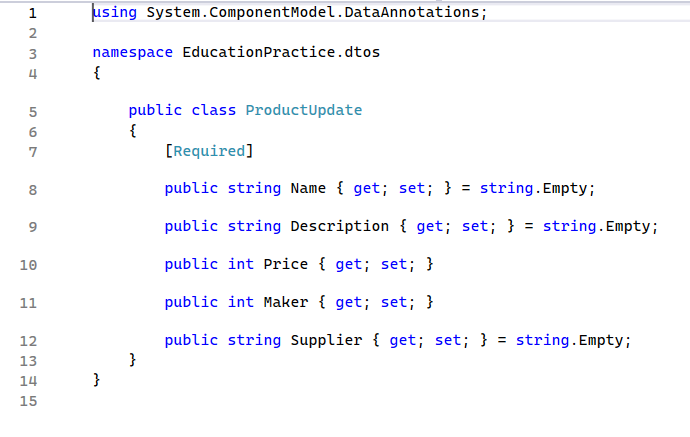


Рисунок 2.5.1 – Обновление

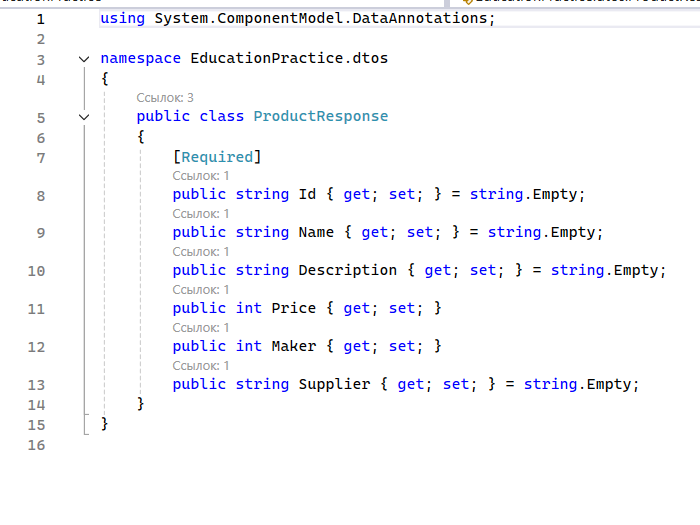


Рисунок 2.5.2 – Запрос

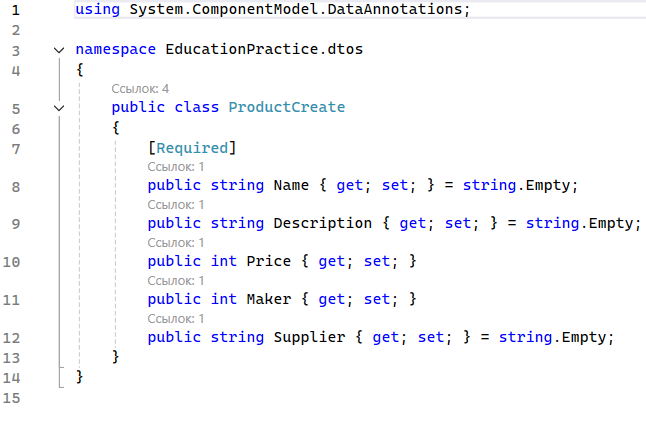
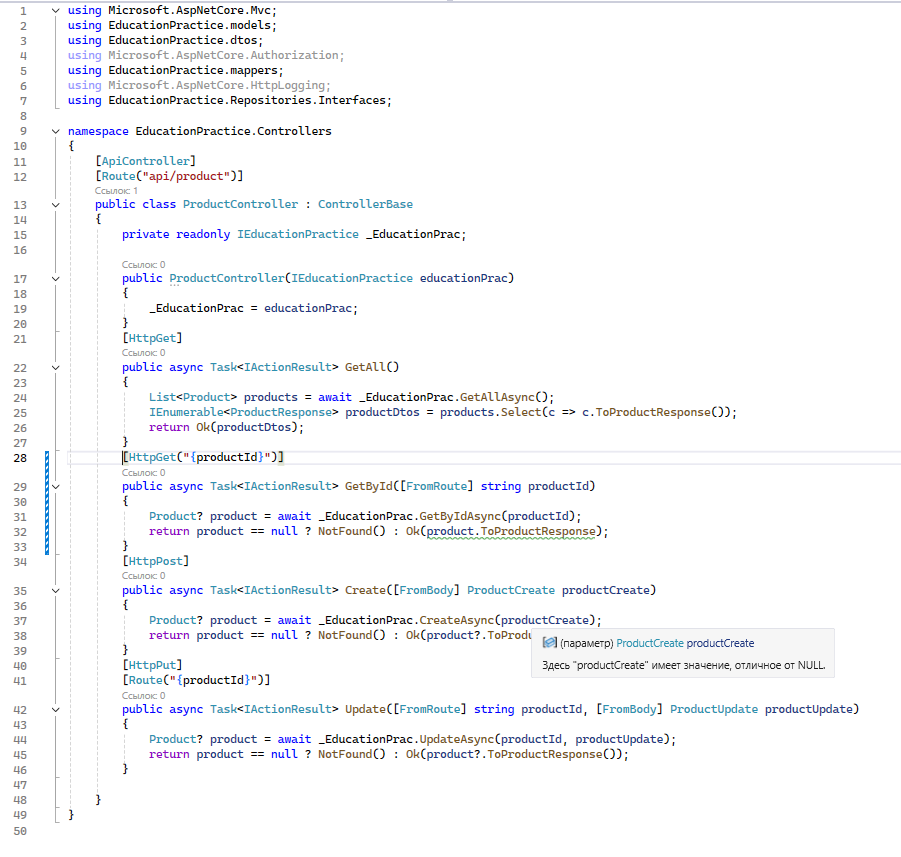


Рисунок 2.5.3 – Создание

Рисунок 2.5.4 – Создание контроллера

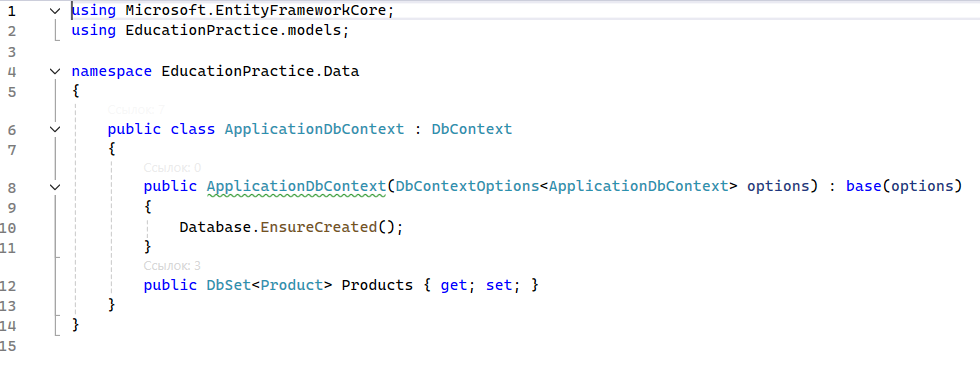
Далее создадим AppDbContext. ApplicationDbContext — это класс, который обычно используется в приложениях на основе Entity Framework (EF) для представления контекста базы данных. Он служит связующим звеном между приложением и базой данных, позволяя выполнять операции CRUD (создание, чтение, обновление, удаление) с сущностями, которые представляют данные в базе.(Рисунок 2.5.5).

Рисунок 2.5.5 – Создание AppDbContext

# ЗАКЛЮЧЕНИЕ

.В ходе выполнения курсовой работы на тему "Разработка Web API для интернет-магазина DNS" была создана архитектура и реализован функционал, позволяющий эффективно управлять данными и взаимодействовать с клиентами. Разработка Web API стала важным шагом в создании современного и удобного интерфейса для пользователей, обеспечивая доступ к основным функциям интернет-магазина, таким как поиск товаров, управление корзиной, оформление заказов и учет пользователей.

В процессе работы над проектом были применены современные технологии и подходы, такие как ASP.NET Core, Entity Framework, а также паттерны проектирования, включая Repository и DTO. Это позволило обеспечить высокую степень модульности, тестируемости и расширяемости приложения. Использование подхода RESTful для проектирования API обеспечило простоту и удобство в использовании, что является важным аспектом для конечных пользователей.

Кроме того, в ходе разработки были учтены вопросы безопасности и производительности, что позволило создать надежное и эффективное решение. Реализованные механизмы аутентификации и авторизации обеспечивают защиту данных пользователей и предотвращают несанкционированный доступ к ресурсам.

В результате проведенной работы был создан функциональный и удобный Web API, который может служить основой для дальнейшего развития интернет-магазина DNS. В будущем возможно расширение функционала, добавление новых возможностей, таких как интеграция с внешними сервисами, улучшение пользовательского интерфейса и оптимизация производительности.

Таким образом, данная курсовая работа не только продемонстрировала навыки разработки и проектирования программного обеспечения, но и предоставила практическое решение для реальной бизнес-задачи, что подчеркивает актуальность и значимость выполненной работы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

**Официальная документация ASP.NET Core**: <https://docs.microsoft.com/ru-ru/aspnet/core/>

**Официальная документация EntityFrameworkCore**: <https://docs.microsoft.com/ru-ru/ef/core/>

**Microsoft ASP.NET Core Web API**: <https://docs.microsoft.com/ru-ru/aspnet/core/web-api/>

**Entity Framework Core в ASP.NET Core**: <https://docs.microsoft.com/ru-ru/aspnet/core/data/ef-mvc/>

**RESTful Web API**: <https://ru.wikipedia.org/wiki/REST>

**JSON Web Tokens (JWT)**: <https://jwt.io/>

**OAuth 2.0**: <https://oauth.net/2/>

**ASP.NET Core Identity**: <https://docs.microsoft.com/ru-ru/aspnet/core/security/authentication/identity>

**Entity Framework Core: Основы**: <https://metanit.com/sharp/entityframeworkcore/1.1.php>

**ASP.NET Core Web API**: Основы: <https://metanit.com/sharp/aspnetcore/1.1.php>