Применение языка C# при создании микросервисной архитектуры

Илларионов А.М., Курочкин С.В [1].

[1] *Владимирский государственный университет им. Александра Григорьевича и Николая Григорьевича Столетовых, г. Владимир.*

С развитием информационных технологий широкое распространение получили микросервисы, позволяющие улучшить функциональность ресурсов, или интерфейсов, а также повысить их быстродействие.

Основной принцип микросервисной архитектуры заключается в декомпозиции приложения на отдельные сервисы, каждый из которых работает отдельно и коммуницирует с остальными, используя, как правило, протоколом HTTP [2]. Такие службы основаны на бизнес-возможностях и могут быть независимо развернуты. Так как микросервисы разворачиваются на серверах, использование программной платформы решает множество проблем, например, при создании приложения, можно будет абстрагироваться от программно-аппаратных особенностей и, благодаря данной архитектуре, конечные пользователи не будут нуждаться в установке дополнительного ПО, как если бы приложение было «монолитным».

Одним из лучших решений для создания микросервисов является бесплатная платформа разработки «.Net» [4], учитывая её опциональность и активное развитие в сфере кроссплатформенности. Рассмотрим основные компоненты, которые будут использованы при реализации сервисов.

Развертывание, передача и обработка информации проводится в рамках ASP.NET Core – кросспалтформенной модификации ASP.NET [1], с архитектурными изменениями, формирующими более рациональную и модульную систему. В этой среде создается и развертывается микросервис. Отталкиваясь от предметной области, реализуется приложение, опираясь на шаблон MVC. Рассмотрим данный подход на примере создания микросервиса, отвечающего за учет книг в книжном магазине с развертыванием на локальной машине. Для начала определим модель – это будет «книга». Создадим класс «Book», включив в него такие поля как ID, Title, Author, Description, Price. Далее перейдем к созданию контроллера – он, в сущности, и будет нашим сервисом. Создадим класс «BookController», наследуемый от ControllerBase и пометим его атрибутом ApiController. Как видите, среда ASP.NET Core уже имеет базовые представления, помогающие ускорить процесс создания сервисов путем повышенного уровня абстракции, близкой к человеческой, так например, в пространстве имён Microsoft.AspNetCore.Mvc есть класс ControllerBase, который предоставляет множество свойств и методов, удобных для обработки HTTP-запросов. В котроллере создадим методы, отвечающие за Get, Post, Put и Delete запросы. Пометим их соответствующим атрибутом, и создадим асинхронную сигнатуру [5].

Большую часть данных представим с помощью реляционных баз данных. В .Net существуют технологии для выполнения этих задач: ADO.NET и Entity Framework Core. Остановимся на второй, так как она является объекто-ориентированной. Доступ к данным осуществляется с помощью модели, состоящей из классов сущностей и объекта контекста, который представляет сеанс взаимодействия с базой данных [3]. Включим в проект EF Core, класс сущности уже определен (класс «Book»). Создадим класс BookContext, который будет наследовать DbContext, базовый класс для объекта контекста в EF Core, опишем свойство «Books», с помощью него мы будем получать массив наших сущностей – книг.

Добавим контекст в контроллер, как приватное поле и организуем работу методов-запросов с помощью этого контекста. Так, например чтобы вернуть список всех книг достаточно прописать await _context.Books.ToListAsync() в теле метода. Также изменим внедрение зависимостей и передадим в качестве контекста данных наш BookContext и ссылку на подключение к БД в классе Startup – он создается автоматически при создании ASP.NET MVC приложения.

Такой примитивный пример показывает суть подхода по организации микросервисной архитектуры, применяемые технологии и инструменты при использовании платформы «.Net».

Список литературы:

1. Введение в ASP.NET Core // Microsoft : [сайт]. – 2021. – URL: https://docs.microsoft.com/ru-ru/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-3.1 (дата обращения 21.12.2021).
2. Микросервисы // Мартин Фаулер : [сайт]. – 2014. – URL: https://martinfowler.com/articles/microservices.html#CharacteristicsOfAMicroserviceArchitecture (дата обращения 21.12.2021).
3. Обзор Entity Framework Core // Microsoft : [сайт]. – 2021. – URL: https://docs.microsoft.com/ru-ru/ef/core/ (дата обращения 21.12.2021).
4. Общие сведения о .Net // Microsoft : [сайт]. – 2021. – URL: https://docs.microsoft.com/ru-ru/dotnet/core/introduction (дата обращения 21.12.2021).
5. Создание API-интерфейсов RESTFUL с помощью веб-API ASP.NET // Microsoft : [сайт]. – 2021. – URL: https://docs.microsoft.com/ru-ru/aspnet/web-api/overview/older-versions/build-restful-apis-with-aspnet-web-api (дата обращения 21.12.2021).