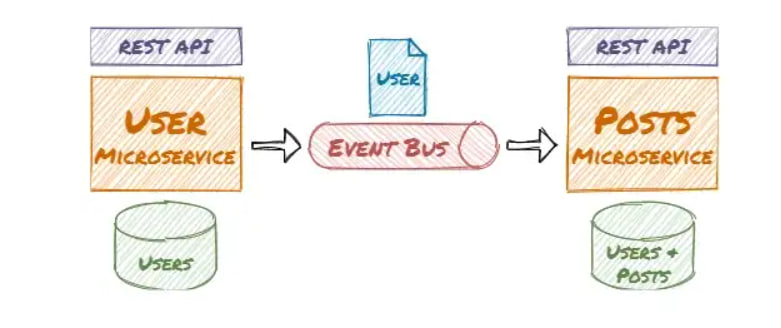
**Event-Driven ASP.NET Core Microservice Architecture**

I tired to create one C# ASP.NET Core Microservices. Microservice have their own bounded context and domain model. Microservice has its own database and REST API. Microservice publishes integration events.



Decoupled Microservices — A Real-World Example With Code

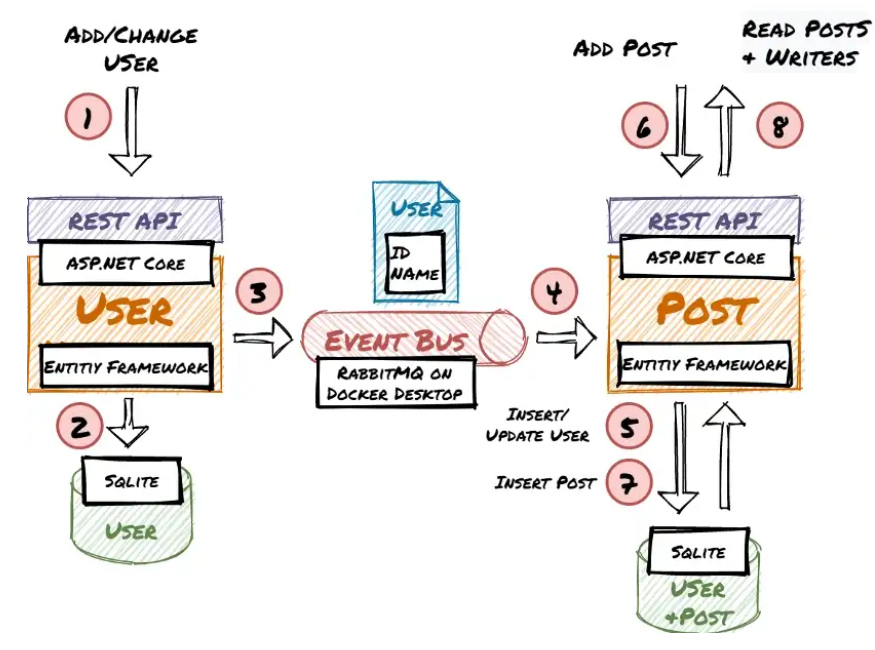
The application uses a real-world example with users that can write posts. The user microservice allows creating and editing users. In the user domain, the user entity has several properties like name, mail, etc. In the post domain, there is also a user so the post microservice can load posts and display the writers without accessing the user microservice. The user entity in the post domain is much simpler:

Зображення, що містить текст

Автоматично згенерований опис

The microservices are decoupled and the asynchronous communication leads to eventual consistency. This kind of architecture is the basis for loosely coupled services and supports high scalability. The microservices access their example Sqlite databases via Entity Framework and exchange messages via RabbitMQ (e.g. on **Docker Desktop**).

Overview diagram of the workflow, components, and technologies:



1. Create the .NET Core Microservices

Create the User and Post Microservice. You will add the Entities and basic Web APIs. The entities will be stored and retrieved via Entity Framework from Sqlite DBs.

Install Visual Studio Community (it’s free) with the ASP.NET and web development workload.

Create a solution and add the two ASP.NET Core 5 Web API projects “UserService” and “PostService”. Disable HTTPS and activate OpenAPI Support.

For both projects install the following NuGet packages:

Microsoft.EntityFrameworkCore.Tools

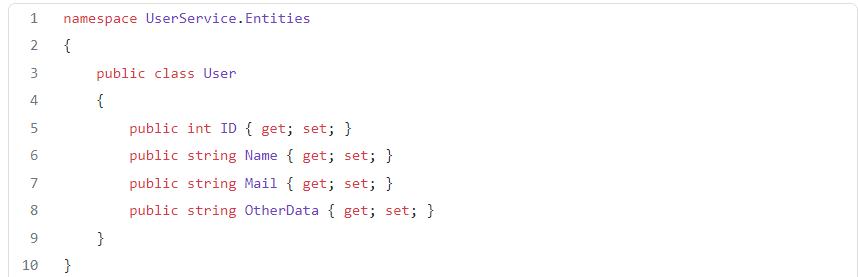
Microsoft.EntityFrameworkCore.Sqlite

Newtonsoft.Json

RabbitMQ.Client

Implement the UserService

Create the User Entity:



Create the UserServiceContext:

Зображення, що містить текст

Автоматично згенерований опис

Edit Startup.cs to configure the UserServiceContext to use Sqlite and call Database.EnsureCreated() to make sure the database contains the entity schema:

Зображення, що містить текст

Автоматично згенерований опис

Create the UserController (It implements only the methods necessary for this demo):

Зображення, що містить текст

Автоматично згенерований опис

Зображення, що містить текст

Автоматично згенерований опис

Debug the UserService project and it will start your browser. You can use the swagger UI to test if creating and reading users is working:

