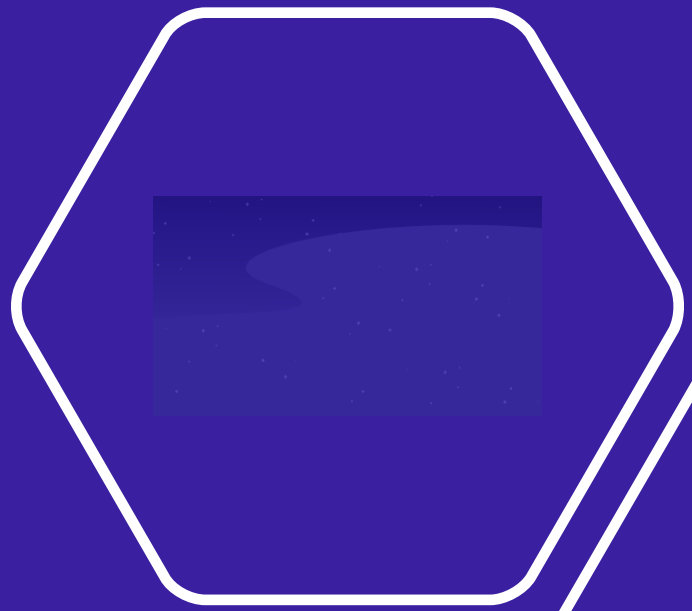
ORGANIZER DEV CAFE Connect and share

SPONSOR Nash Tech.

.NET Conf 2021 Vietnam

13 Nov 2021



Streamline .NET Microservices with Minimal APIs and C# 10

Thang Chung

13 Nov 2021

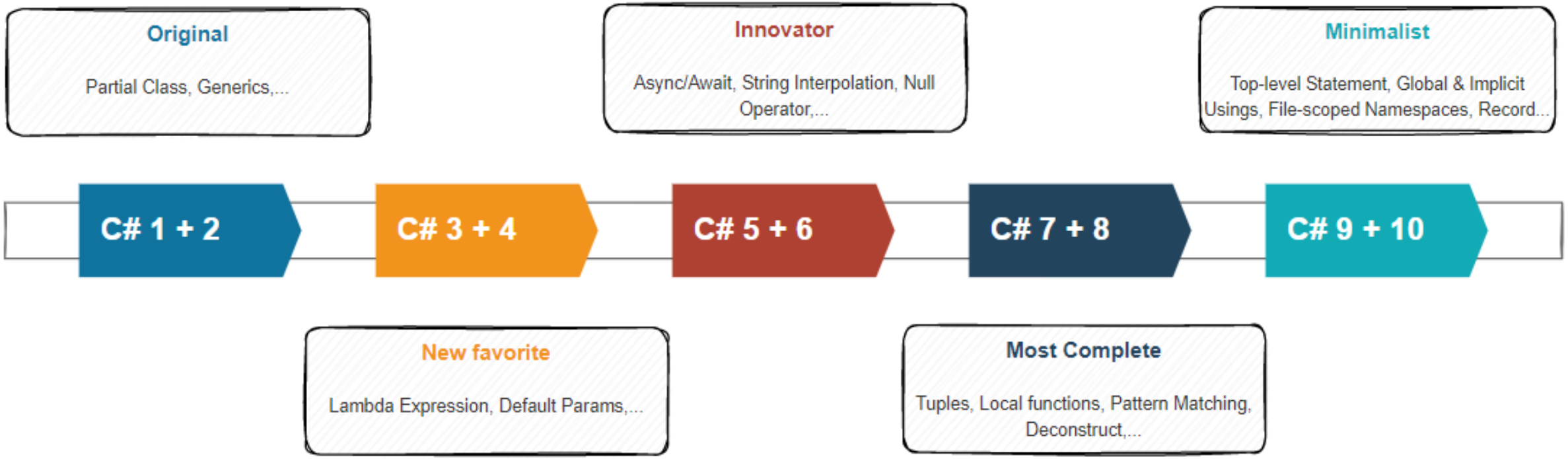
> whoami

- My name is Thang Chung
  - Work at NashTech Vietnam
  - Experience: > 10 years in the software development area, in both outsourcing and product companies
  - Hobbies:
    - Research and deep dive into new things in the software development world
    - Read books, drink beer, and contemplate the nature of my life
  - Spend many hours per day at https://github.com/thangchung
  - Creator of many of OSSes such as awesome-dotnet-core, clean-code-dotnet, clean-architecture-dotnet, vietnam-devs/coolstore-microservices
- Proudly become a Microsoft MVP since 2020

Agenda

- C# Evolution
- Minimal APIs
- Business Context for the Demo Application
- Communication Styles
  - Remote Procedure Invocation
  - Messaging for Event-Carried State Transfer
  - Data Consistency with Sagas State Machine
- Tools to support Microservice development
  - Watch .NET services
  - Debug .NET services in productivity ways

C# Evolution


Original
Partial Class, Generics,...
Innovator
Async/Await, String Interpolation, Null Operator,...
Minimalist
Top-level Statement, Global & Implicit Usings, File-scoped Namespaces, Record...
C# 1 + 2
C# 3 + 4
C# 5 + 6
C# 7 + 8
C# 9 + 10
New favorite
Lambda Expression, Default Params,...
Most Complete
Tuples, Local functions, Pattern Matching, Deconstruct,...

C# 10 – Top-level Statements

```
1 var app = WebApplication.Create(args);
2
3 app.MapGet("/", () => "Welcome to Vietnam .NET Conf 2021!");
4
5 app.Run();
```

C# 10 – Global and Implicit Usings

```xml
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
  </PropertyGroup>

  <ItemGroup>
    <PackageReference Include="Swashbuckle.AspNetCore" Version="6.2.3" />
  </ItemGroup>

  <ItemGroup>
    <Using Include="System.Text.Json" /> <!-- We can do this! -->
  </ItemGroup>

</Project>
```

```
using System.Text.Json; // Then we can remove this from cs files

var builder = WebApplication.CreateBuilder(args);
```

C# 10 – File-scoped Namespaces

```
1  namespace SampleMinimalWebApi; // we can do this
2
3  public interface IProductRepository
4  {
5      Task<IEnumerable<Product>> GetProducts();
6      Task<Product?> GetProduct(int id);
7  }
8
9  internal class ProductRepository : IProductRepository
10 {
11     public Task<Product?> GetProduct(int id)
12     {
13         // some implementation here
14     }
15
16     public async Task<IEnumerable<Product>> GetProducts()
17     {
18         // some implementation here
19     }
20 }
```

C# 10 – Record Struct

```
1 record struct Product(int Id, string Name, Category? Category = null);
2
3 record Category(int Id, string Name);
```

C# 10 – Improved Patterns Matching

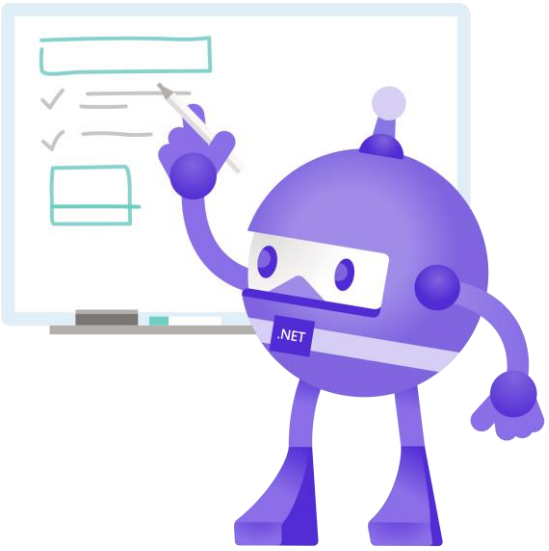
```
1 return await productRepository.GetProduct(id) switch
2 {
3     Product product => Results.Ok(product),
4     null => Results.NotFound()
5 };
```

C# 10 – Improved Deconstruct

```
foreach (var (productId, unitPrice, quantity) in orderCommand.Details)
{
    order.InsertOrderDetail(order.Id, productId, unitPrice, quantity);
}
```

Minimal APIs

- Rapidly move from idea to a functioning application
- The C# ecosystem powers the most productive applications on the web
- Proven to be one of the fastest web servers in the world, serving more than 4 million requests per second
- Great for Microservices
- If you like Express, Flask or FastAPI, then it’s for you



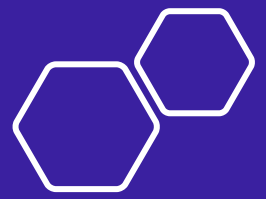
Demo

Web API vs Web Minimal APIs

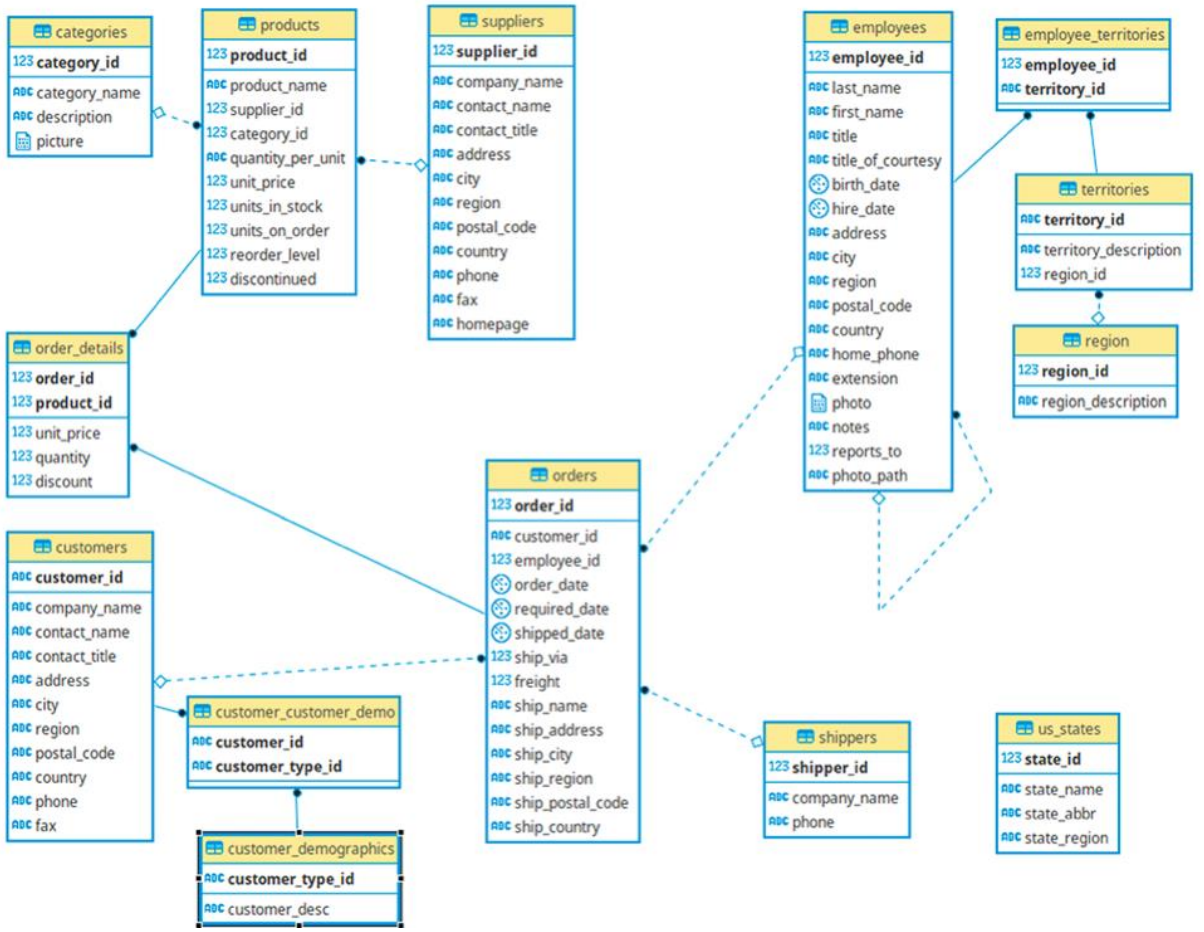
https://github.com/thangchung/minimal-apis-csharp10

Every project starts from the business requirement, right?

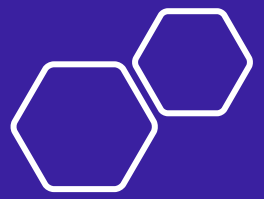
Microservices projects should think about business context, instead of technical needs...



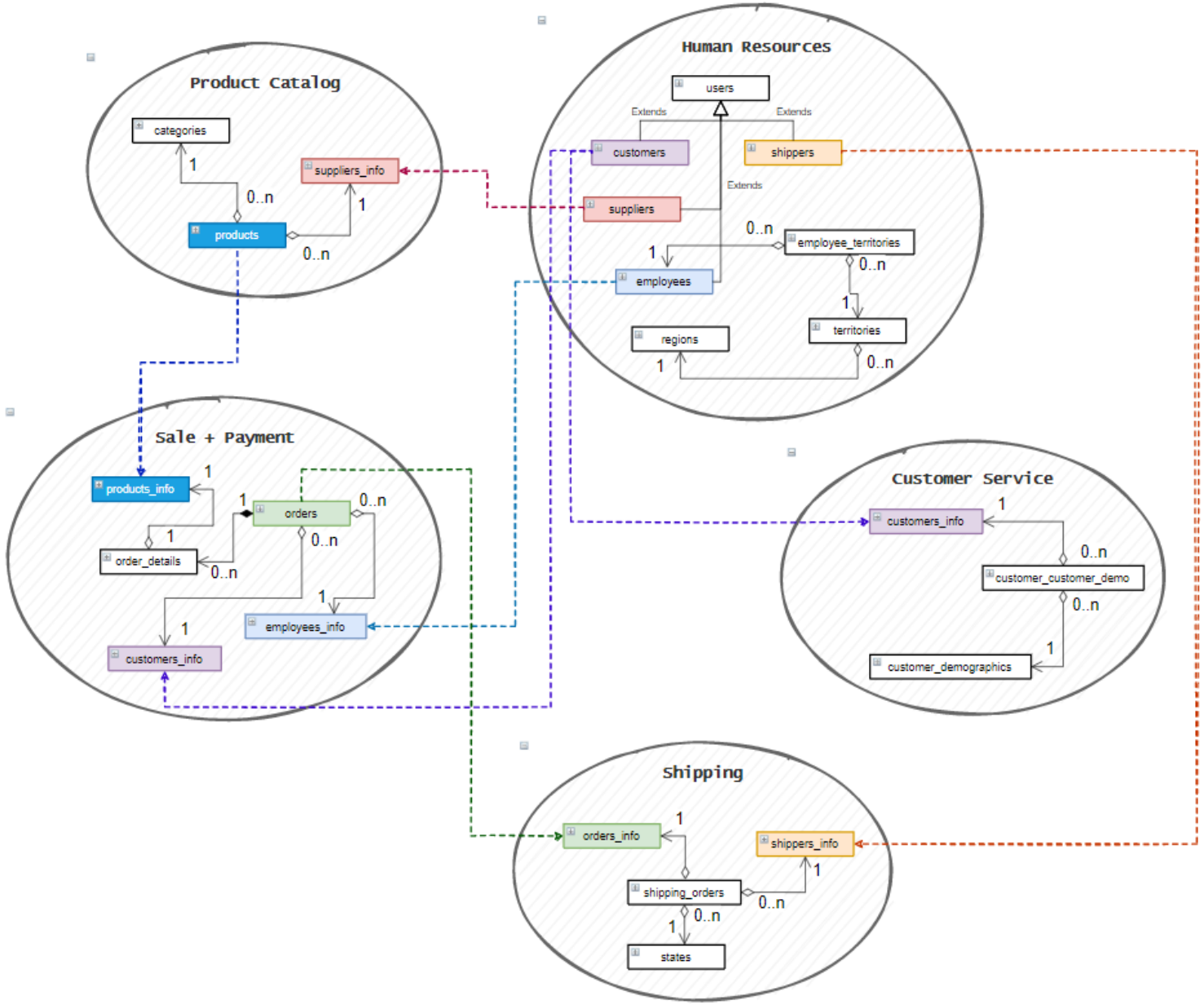
Let say we have the Northwind monolith application that needs to be re-architect to the Microservices approach.

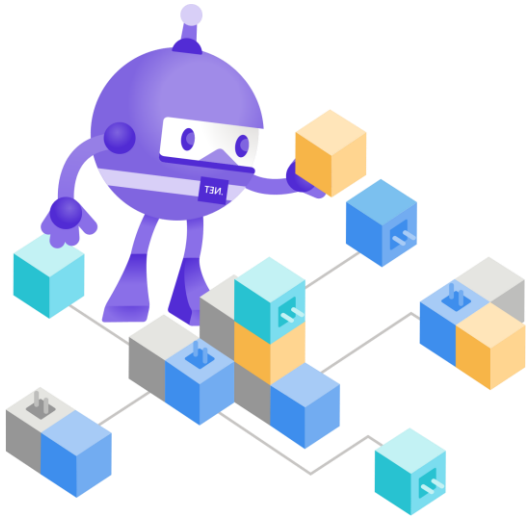


https://docs.yugabyte.com/latest/sample-data/northwind



Sketch out some of the bounded contexts.

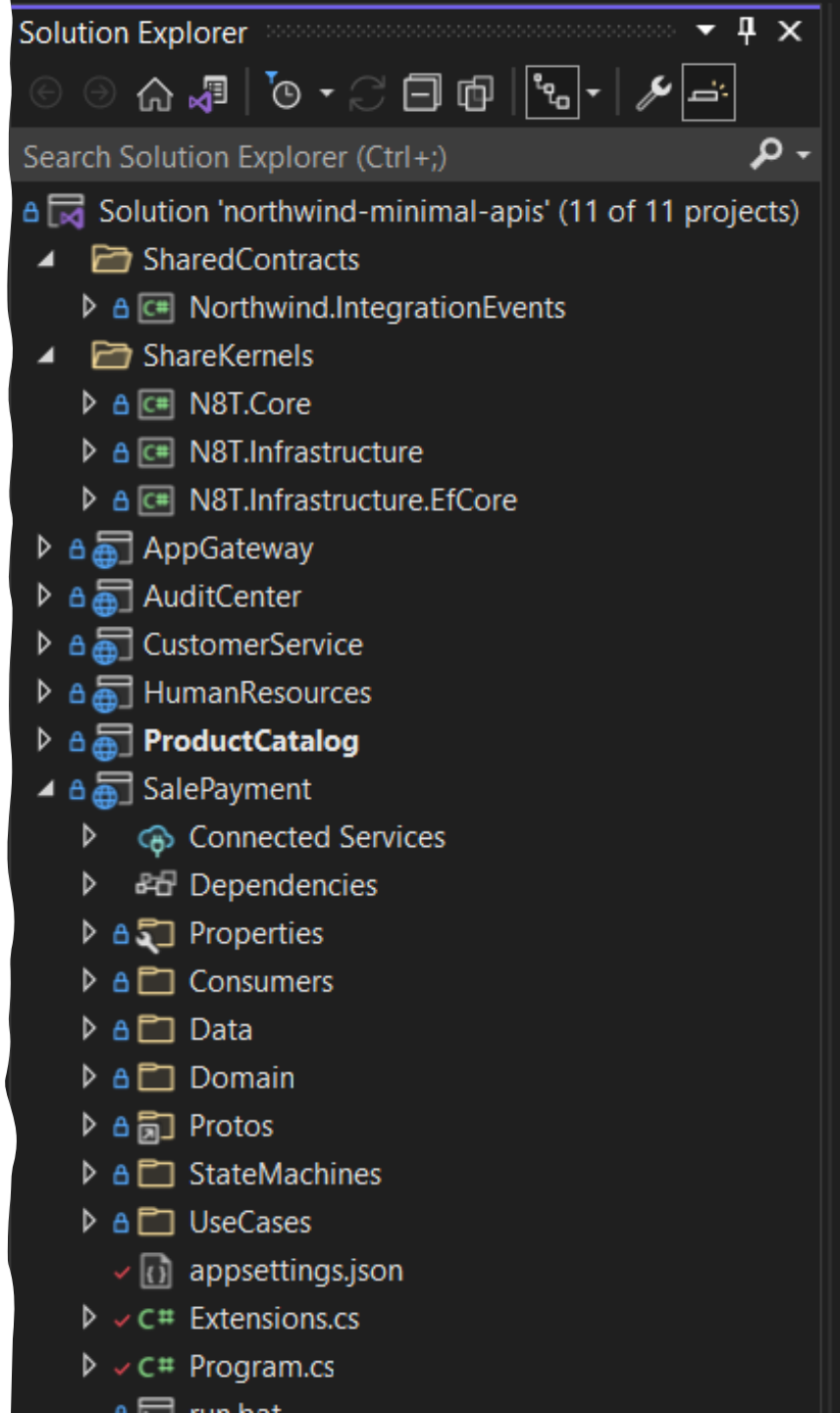




Demo

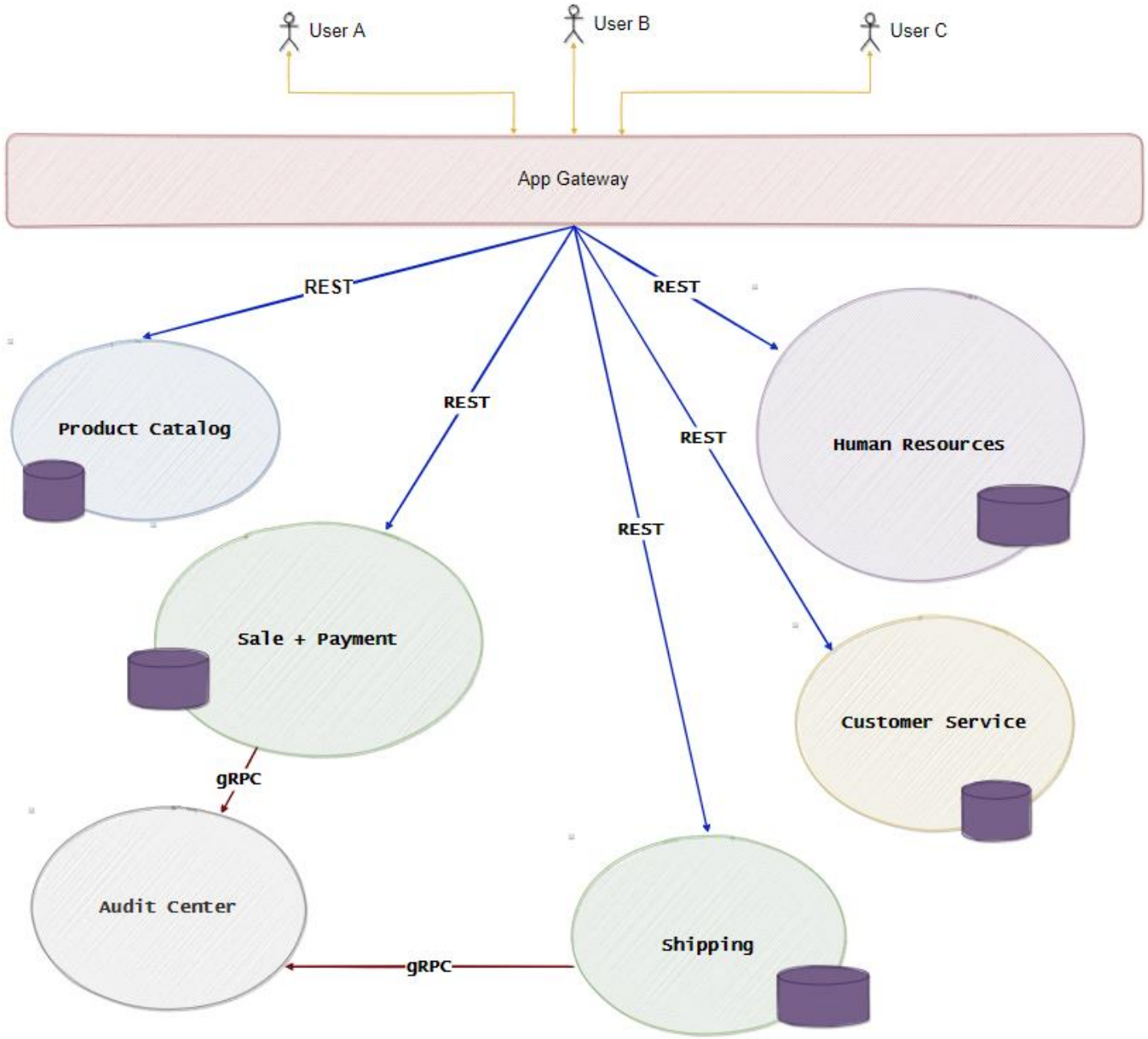
Show me the code please!

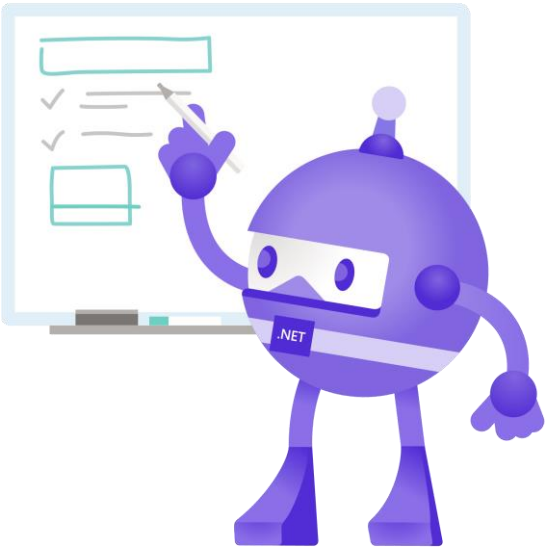
https://github.com/thangchung/northwind-dotnet



Communication Styles

Remote Procedure Invocation

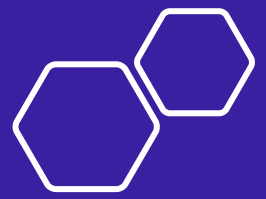




Demo

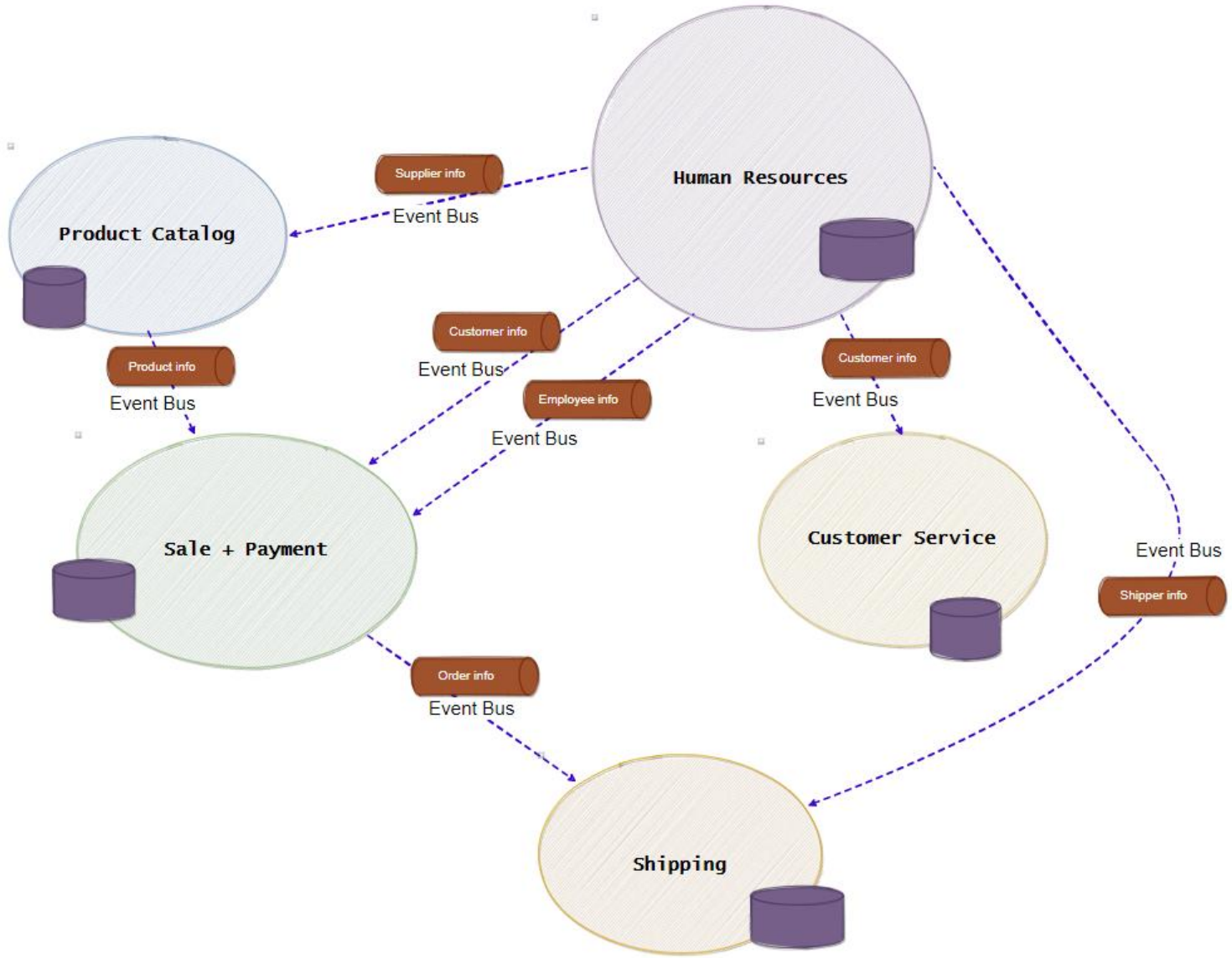
YARP + REST: Reverse Proxy

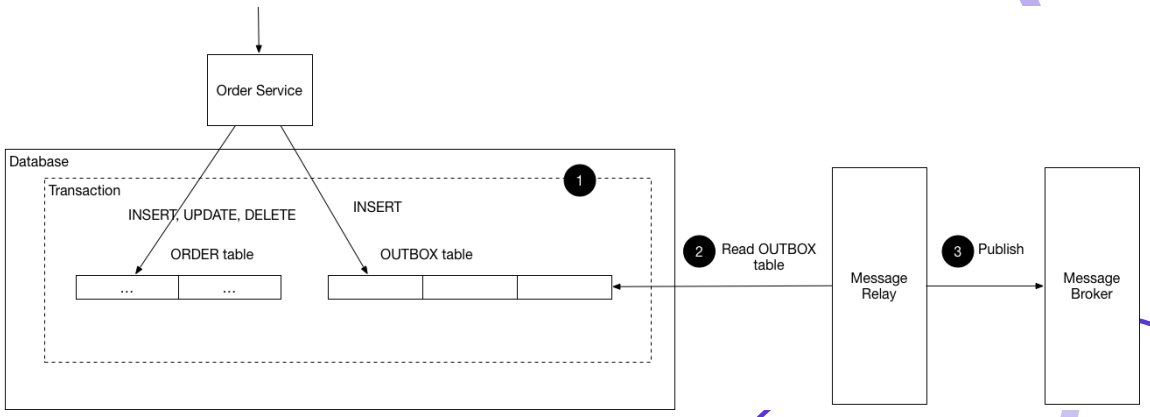
gRPC with HTTP/3: service to service calling

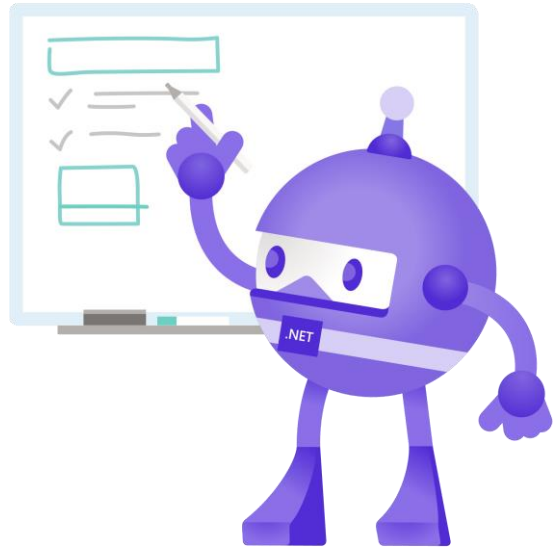


Communication Styles

Messaging (with Event-Carried State Transfer)





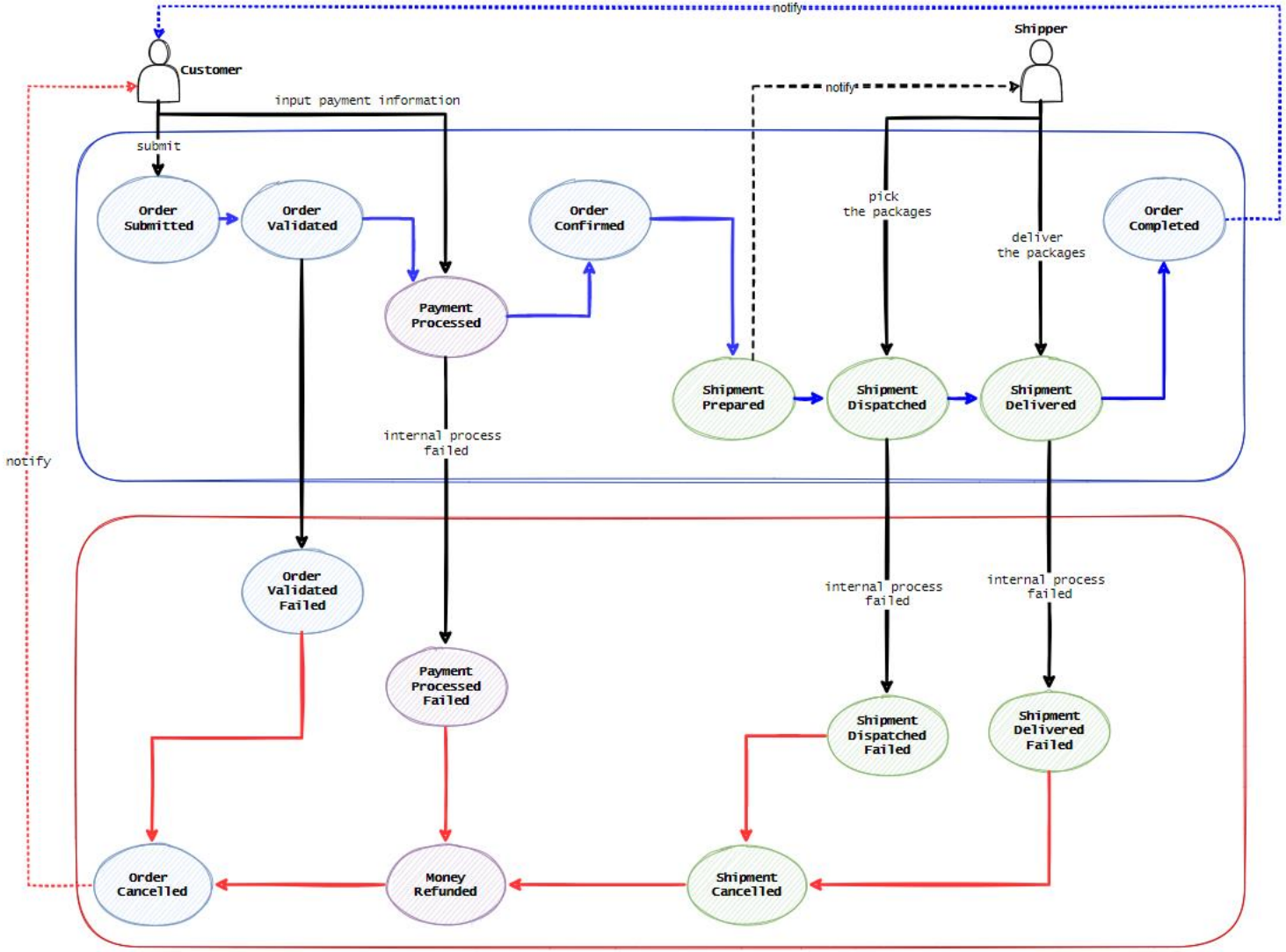


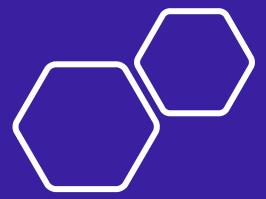
Demo

Outbox patterns to avoid inconsistency data

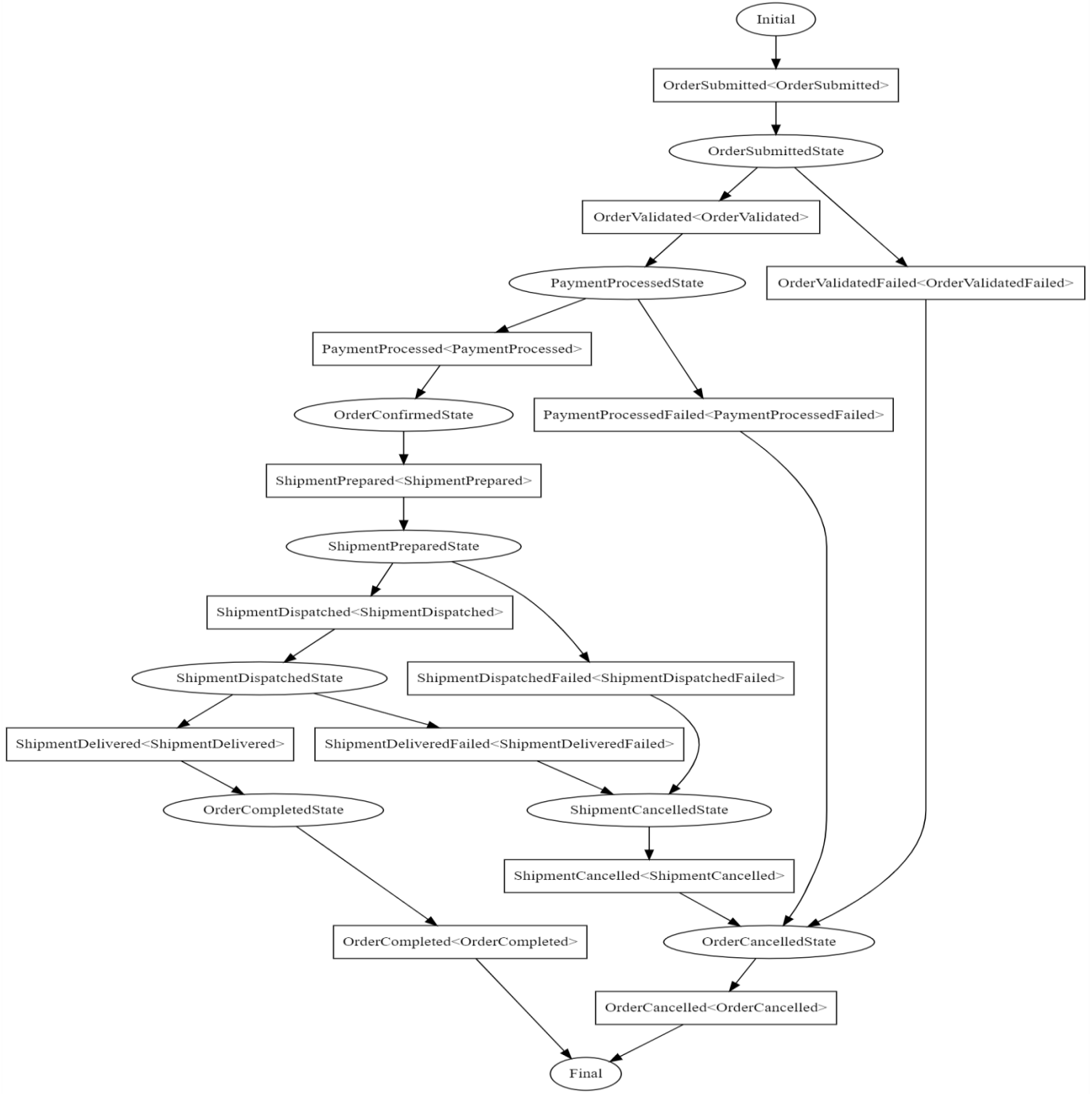
Kafka and Debezium for replicating data to other services

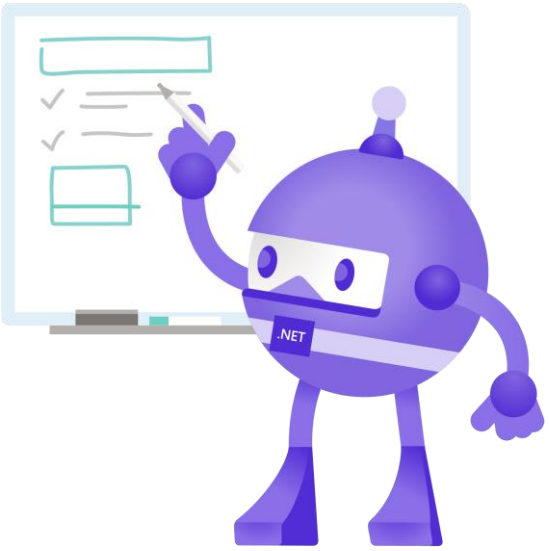
Data Consistency with Sagas State Machine





Data Consistency with Sagas State Machine - GraphViz

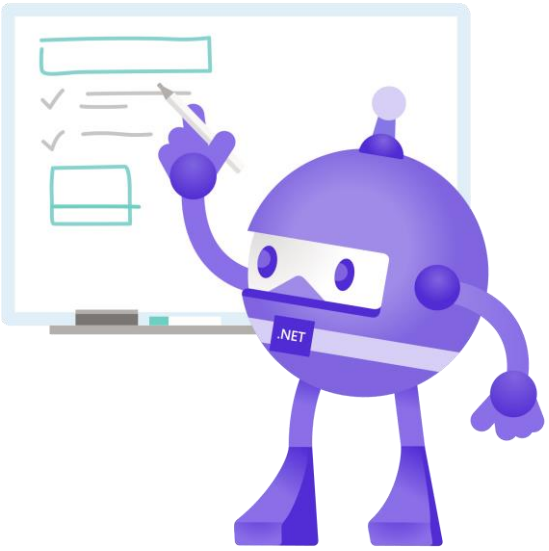




Demo

MassTransit Sagas + State Machine (persistence state to MongoDB) and Kafka for Event Bus

Structured Logs



Demo

Tools to support Microservice development (tye)

- Watch
- Debug

RELEASED

.NET 6 for Microservices

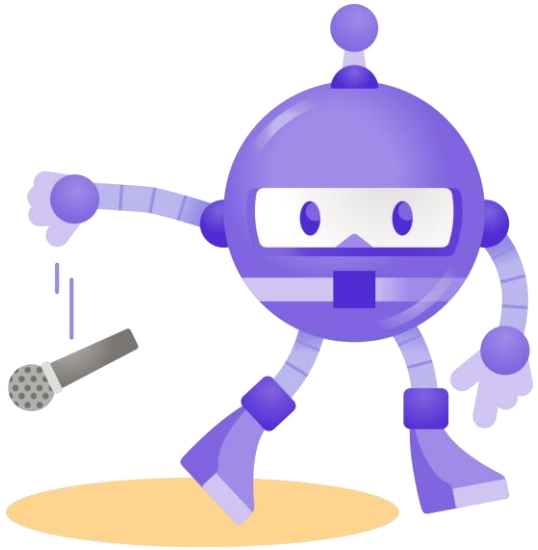
- C# 10
- Minimal APIs
- gRPC with HTTP/3
- EF Core 6: Migrations, Compiled Models
- Reverse Proxy with YARP
- .NET 6 with Kafka + Debezium
- .NET 6 with MassTransit + Kafka (rider)
- Structured Logs with Serilog
- Tye with watch/debug

Source Code: https://github.com/thangchung/northwind-dotnet

Thanks for joining!

DEV CAFE
Connect and share

https://www.facebook.com/devcafevn



References

- https://docs.microsoft.com/en-us/aspnet/core/fundamentals/minimal-apis
- https://minimal-apis.github.io/
- https://docs.microsoft.com/en-us/visualstudio/debugger/hot-reload
- https://devblogs.microsoft.com/dotnet/performance-improvements-in-net-6
- https://martinfowler.com/articles/201701-event-driven.html
- https://microservices.io/patterns/apigateway.html
- https://microservices.io/patterns/data/saga.html
- https://microservices.io/patterns/data/transactional-outbox.html
- https://github.com/thangchung/minimal-apis-csharp10
- https://github.com/thangchung/northwind-dotnet