SolarLab>_

Беглый взгляд на ASP.NET с source gen'ами

Содержание

1. Что такое Source Generators?
2. Тестовые приложения
3. Minimal API
4. Медиатор
5. Маппинг
6. Сериализация
7. Тесты производительности
8. Итоги

Первые попытки в Source Generators?

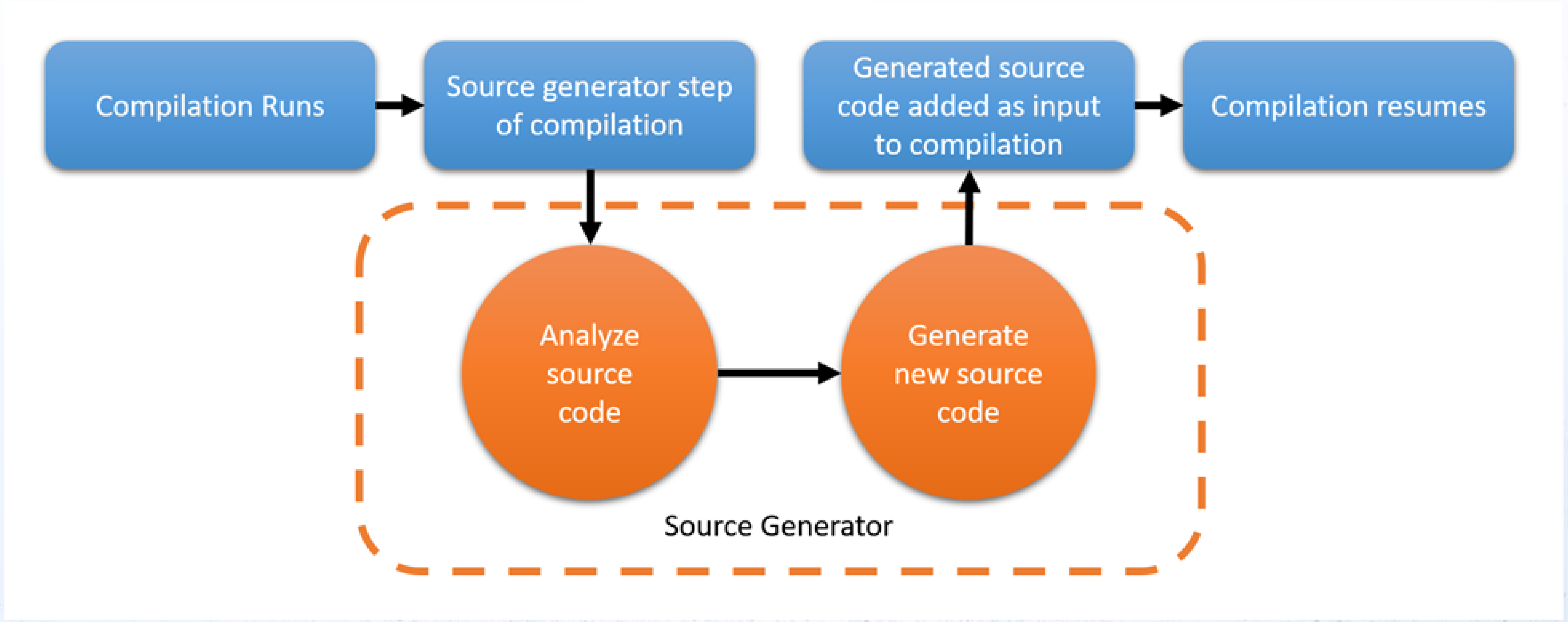
- T4
- Fody
- PostSharp

IL-Weaving

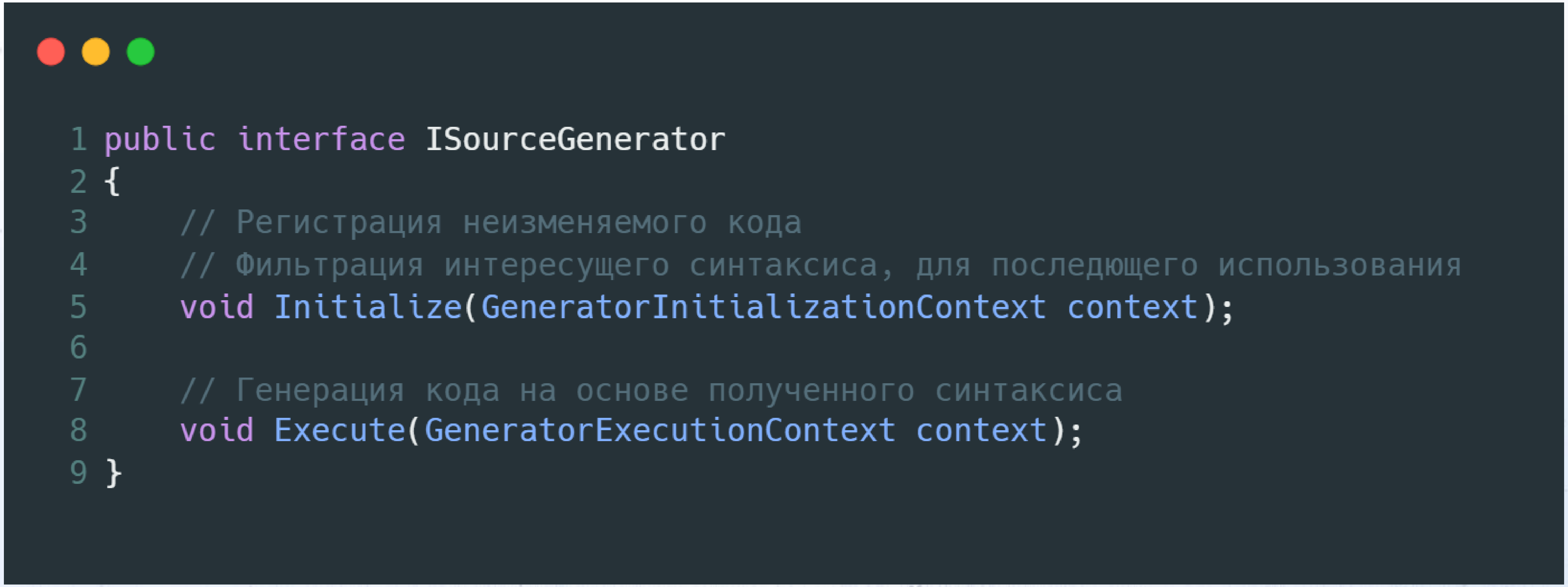
Особенности:

- «Невидимость» изменений
- Меняет существующий код
- Не проверялся компилятором

.NET 5

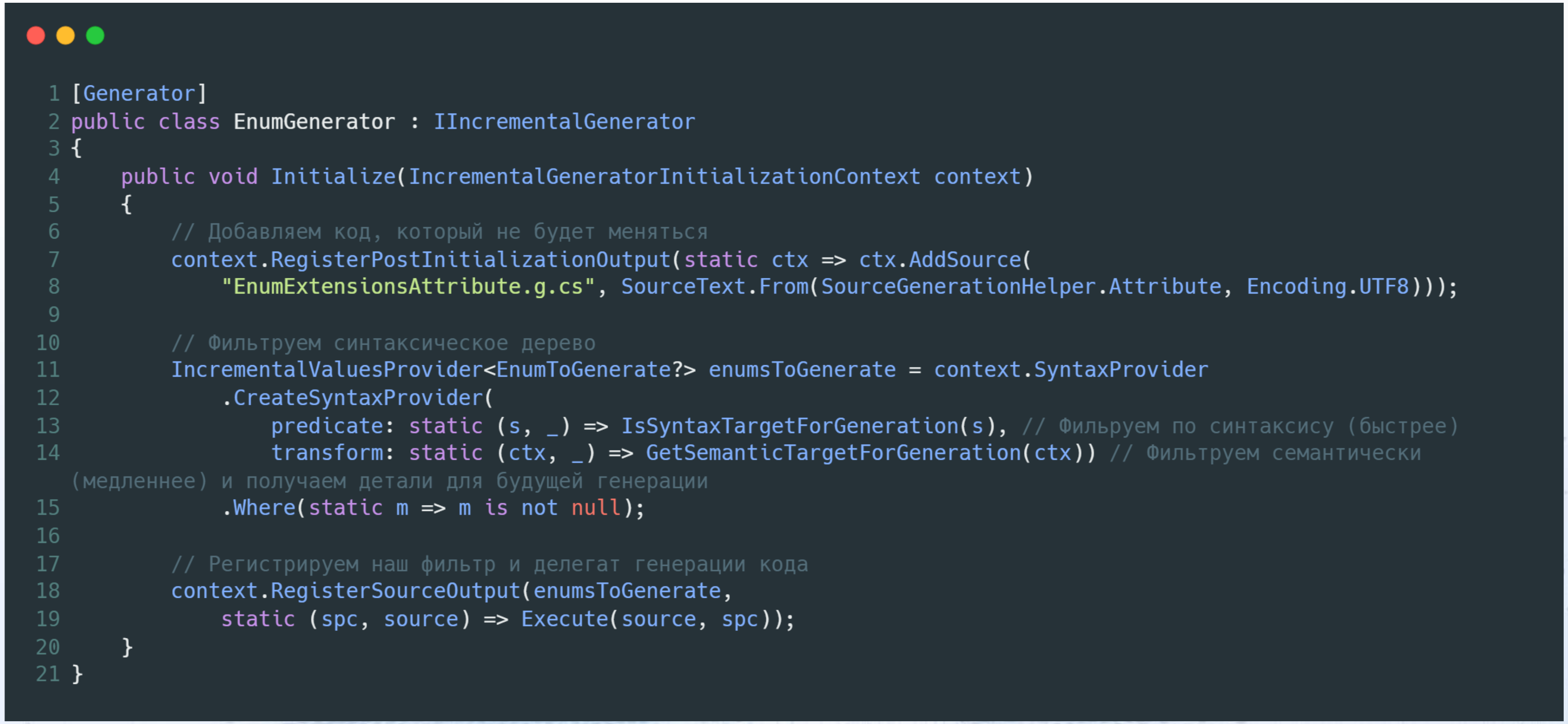

Compilation Runs
Source generator step of compilation
Generated source code added as input to compilation
Compilation resumes
Analyze source code
Generate new source code
Source Generator

.NET 5



```
public interface ISourceGenerator
{
    // Регистрация неизменяемого кода
    // Фильтрация интересущего синтаксиса, для последющего использования
    void Initialize(GeneratorInitializationContext context);

    // Генерация кода на основе полученного синтаксиса
    void Execute(GeneratorExecutionContext context);
}
```

.NET 6



```csharp
[Generator]
public class EnumGenerator : IIncrementalGenerator
{
    public void Initialize(IncrementalGeneratorInitializationContext context)
    {
        // Добавляем код, который не будет меняться
        context.RegisterPostInitializationOutput(static ctx => ctx.AddSource(
            "EnumExtensionsAttribute.g.cs", SourceText.From(SourceGenerationHelper.Attribute, Encoding.UTF8)));

        // Фильтруем синтаксическое дерево
        IncrementalValuesProvider<EnumToGenerate?> enumsToGenerate = context.SyntaxProvider
            .CreateSyntaxProvider(
                predicate: static (s, _) => IsSyntaxTargetForGeneration(s), // Фильруем по синтаксису (быстрее)
                transform: static (ctx, _) => GetSemanticTargetForGeneration(ctx)) // Фильтруем семантически (медленнее) и получаем детали для будущей генерации
            .Where(static m => m is not null);

        // Регистрируем наш фильтр и делегат генерации кода
        context.RegisterSourceOutput(enumsToGenerate,
            static (spc, source) => Execute(source, spc));
    }
}
```

Тестовые приложения

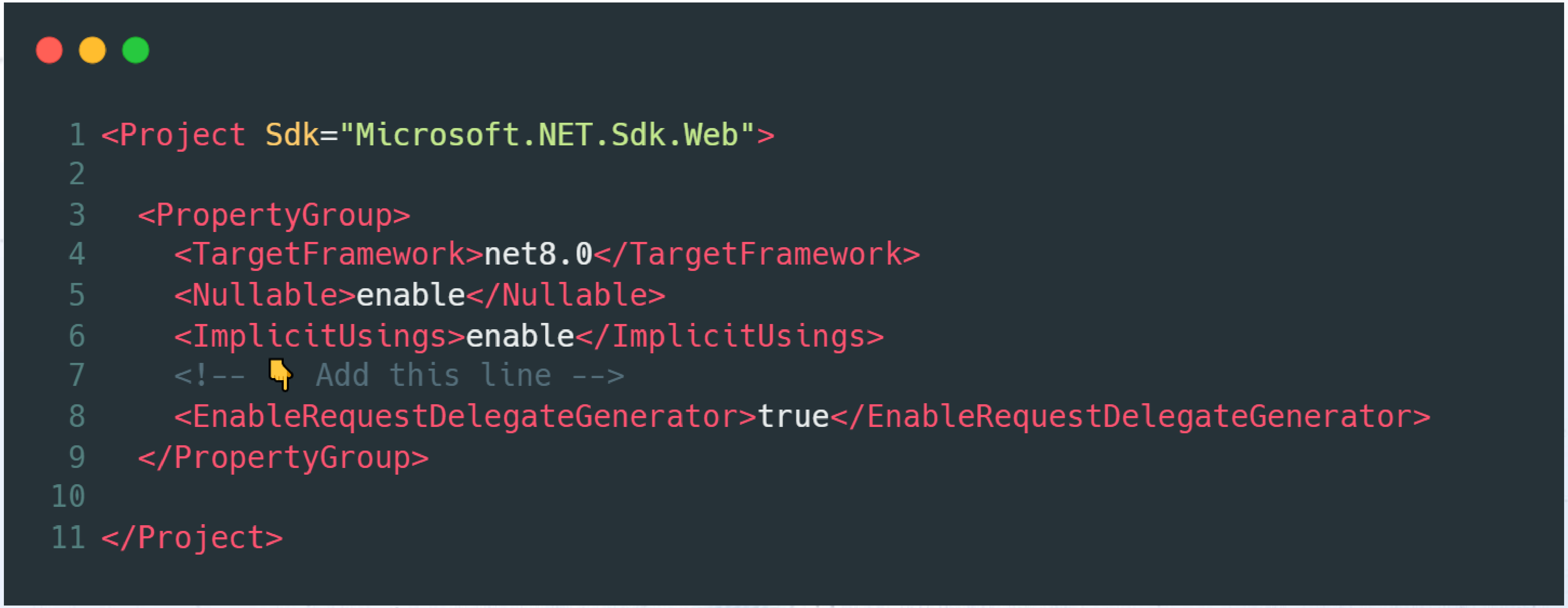
Стандартное

- Controllers
- AutoMapper
- MediatR
- System.Text.Json

На Source Generators

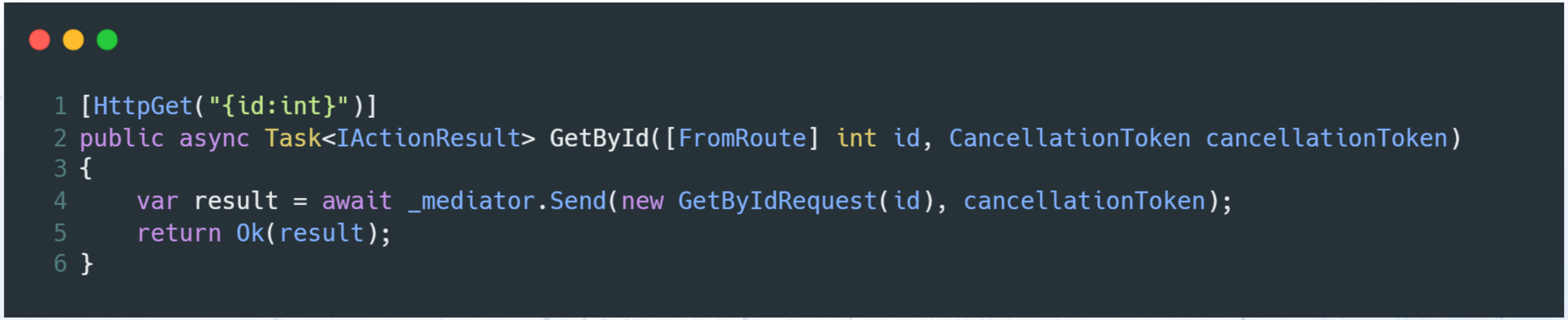
- MinimalAPI (RDG)
- Mapperly
- Mediator
- System.Text.Json

MinimalAPI (Request Delegate Generator)



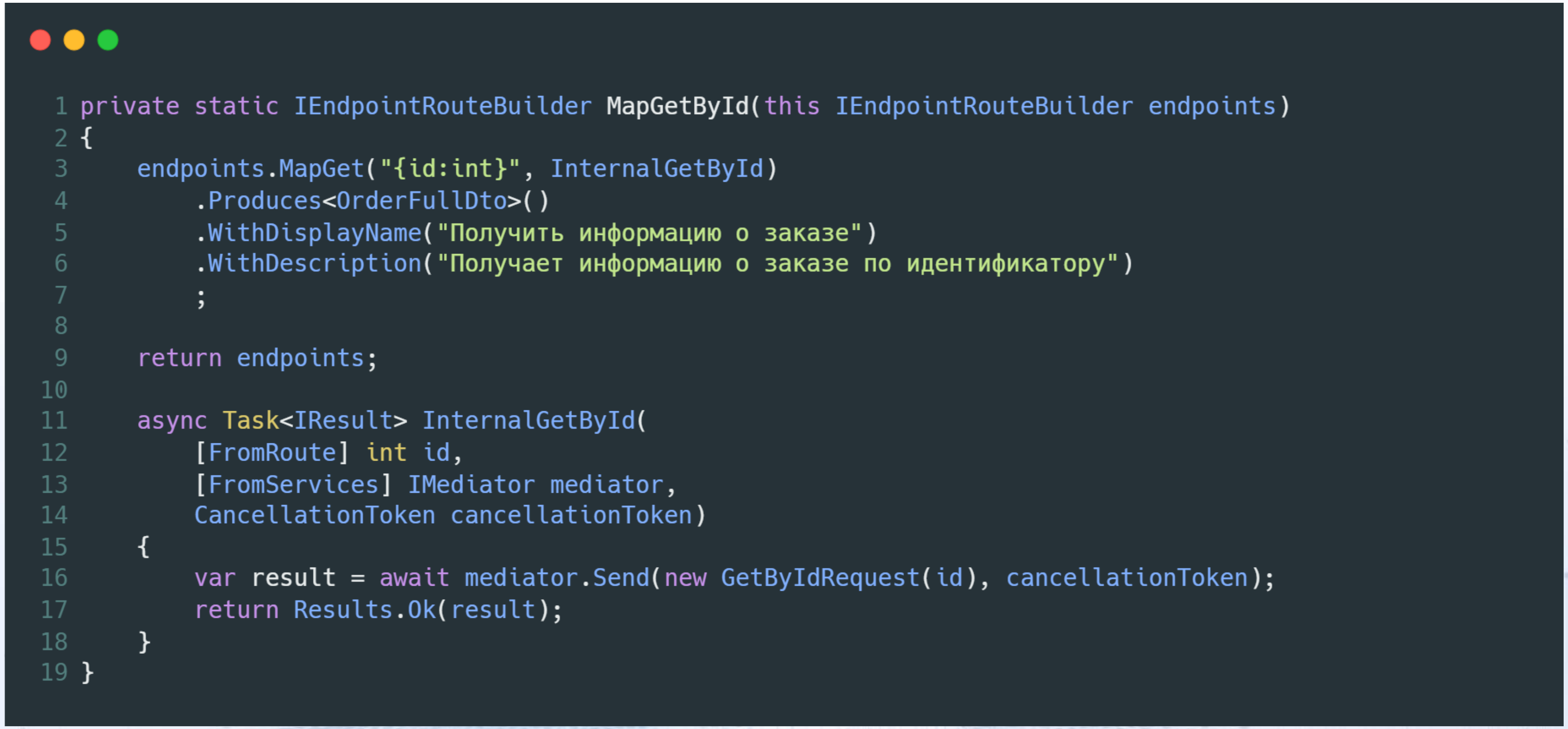
```xml
<Project Sdk="Microsoft.NET.Sdk.Web">

  <PropertyGroup>
    <TargetFramework>net8.0</TargetFramework>
    <Nullable>enable</Nullable>
    <ImplicitUsings>enable</ImplicitUsings>
    <!-- 👇 Add this line -->
    <EnableRequestDelegateGenerator>true</EnableRequestDelegateGenerator>
  </PropertyGroup>

</Project>
```

MinimalAPI (Request Delegate Generator)



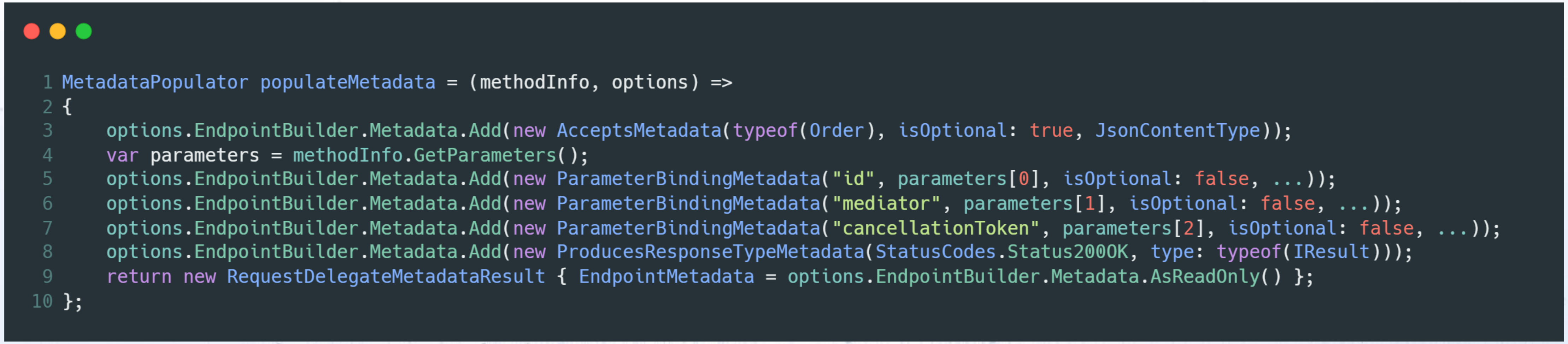
```
[HttpGet("{id:int}")]
public async Task<IActionResult> GetById([FromRoute] int id, CancellationToken cancellationToken)
{
    var result = await _mediator.Send(new GetByIdRequest(id), cancellationToken);
    return Ok(result);
}
```

MinimalAPI (Request Delegate Generator)



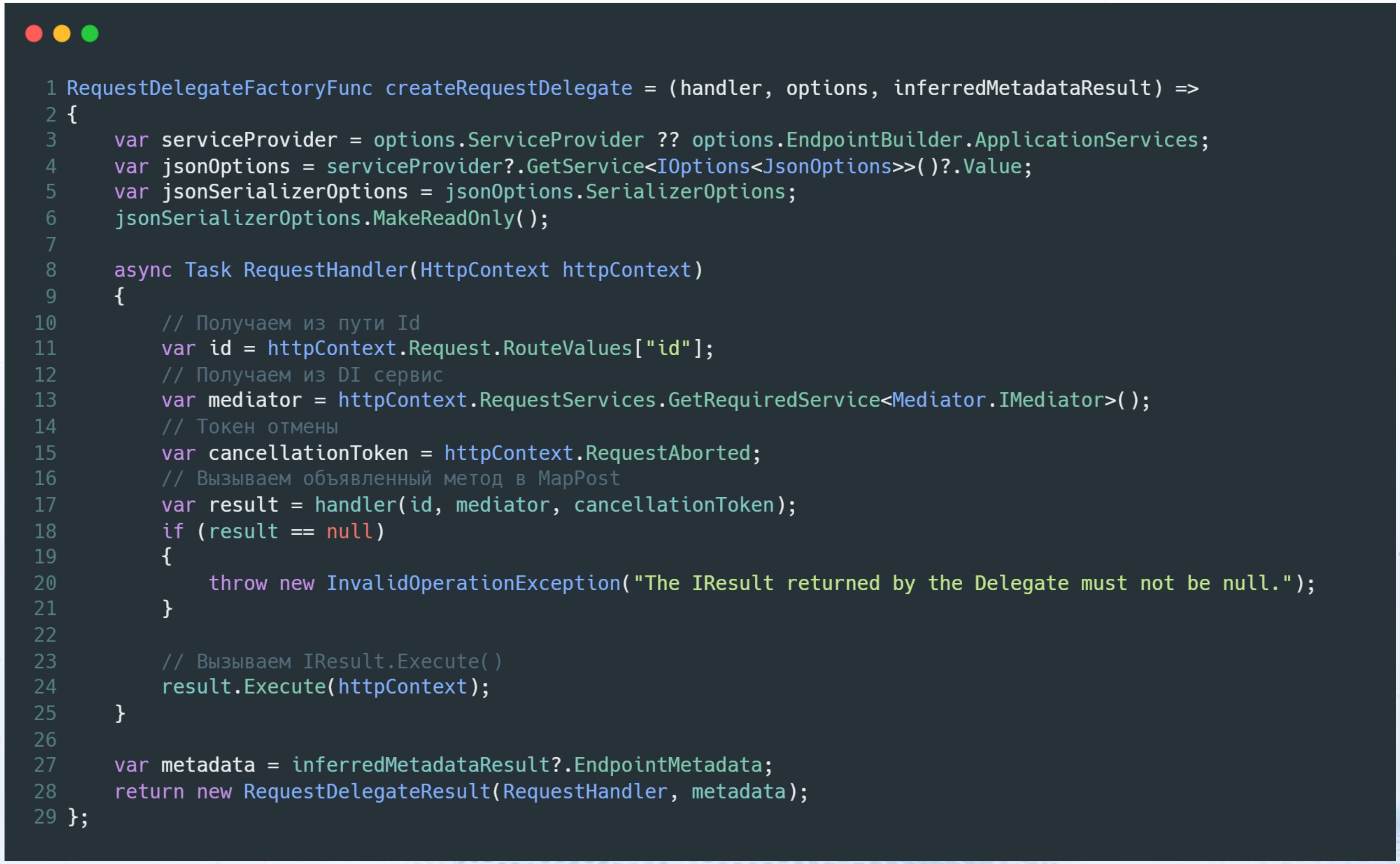
```
private static IEndpointRouteBuilder MapGetById(this IEndpointRouteBuilder endpoints)
{
    endpoints.MapGet("{id:int}", InternalGetById)
        .Produces<OrderFullDto>()
        .WithDisplayName("Получить информацию о заказе")
        .WithDescription("Получает информацию о заказе по идентификатору")
        ;

    return endpoints;

    async Task<IResult> InternalGetById(
        [FromRoute] int id,
        [FromServices] IMediator mediator,
        CancellationToken cancellationToken)
    {
        var result = await mediator.Send(new GetByIdRequest(id), cancellationToken);
        return Results.Ok(result);
    }
}
```

MinimalAPI (Request Delegate Generator)



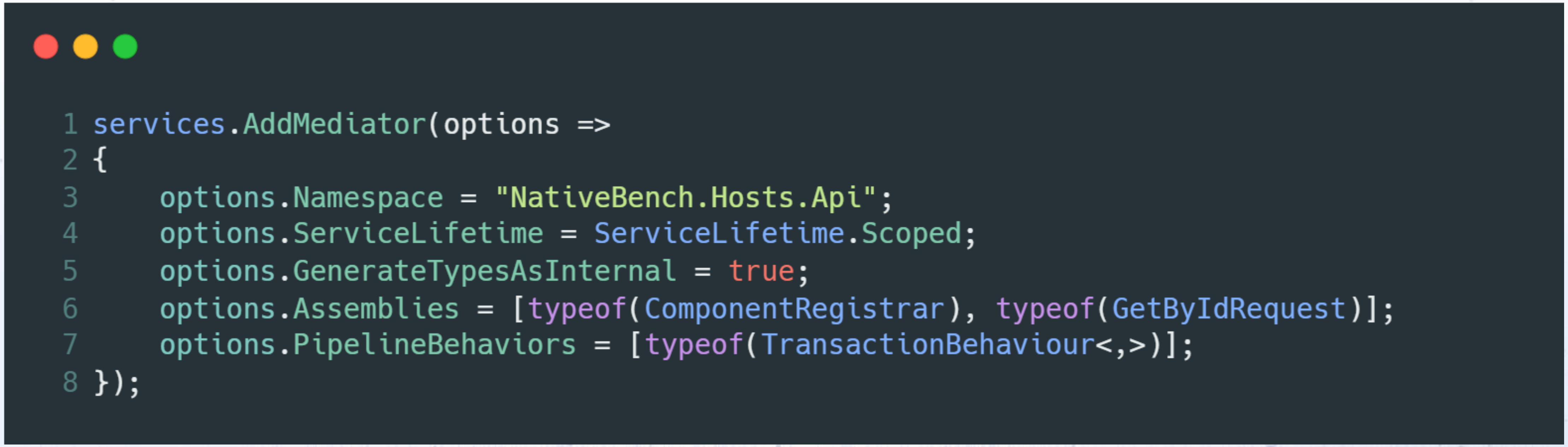
```
MetadataPopulator populateMetadata = (methodInfo, options) =>
{
    options.EndpointBuilder.Metadata.Add(new AcceptsMetadata(typeof(Order), isOptional: true, JsonContentType));
    var parameters = methodInfo.GetParameters();
    options.EndpointBuilder.Metadata.Add(new ParameterBindingMetadata("id", parameters[0], isOptional: false, ...));
    options.EndpointBuilder.Metadata.Add(new ParameterBindingMetadata("mediator", parameters[1], isOptional: false, ...));
    options.EndpointBuilder.Metadata.Add(new ParameterBindingMetadata("cancellationToken", parameters[2], isOptional: false, ...));
    options.EndpointBuilder.Metadata.Add(new ProducesResponseTypeMetadata(StatusCodes.Status200OK, type: typeof(IResult)));
    return new RequestDelegateMetadataResult { EndpointMetadata = options.EndpointBuilder.Metadata.AsReadOnly() };
};
```

MinimalAPI (Request Delegate Generator)



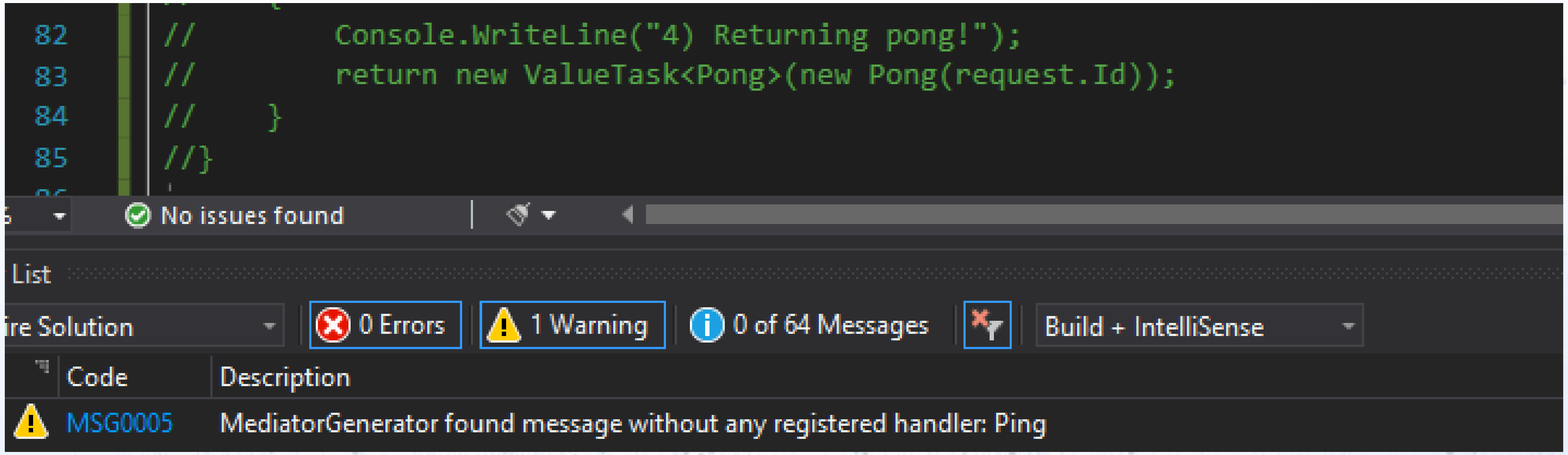
```
RequestDelegateFactoryFunc createRequestDelegate = (handler, options, inferredMetadataResult) =>
{
    var serviceProvider = options.ServiceProvider ?? options.EndpointBuilder.ApplicationServices;
    var jsonOptions = serviceProvider?.GetService<IOptions<JsonOptions>>()?.Value;
    var jsonSerializerOptions = jsonOptions.SerializerOptions;
    jsonSerializerOptions.MakeReadOnly();

    async Task RequestHandler(HttpContext httpContext)
    {
        // Получаем из пути Id
        var id = httpContext.Request.RouteValues["id"];
        // Получаем из DI сервис
        var mediator = httpContext.RequestServices.GetRequiredService<Mediator.IMediator>();
        // Токен отмены
        var cancellationToken = httpContext.RequestAborted;
        // Вызываем объявленный метод в MapPost
        var result = handler(id, mediator, cancellationToken);
        if (result == null)
        {
            throw new InvalidOperationException("The IResult returned by the Delegate must not be null.");
        }

        // Вызываем IResult.Execute()
        result.Execute(httpContext);
    }

    var metadata = inferredMetadataResult?.EndpointMetadata;
    return new RequestDelegateResult(RequestHandler, metadata);
};
```

Mediator



```
services.AddMediator(options =>
{
    options.Namespace = "NativeBench.Hosts.Api";
    options.ServiceLifetime = ServiceLifetime.Scoped;
    options.GenerateTypesAsInternal = true;
    options.Assemblies = [typeof(ComponentRegistrar), typeof(GetByIdRequest)];
    options.PipelineBehaviors = [typeof(TransactionBehaviour<,>)];
});
```

Mediator



```
//          Console.WriteLine("4) Returning pong!");
//          return new ValueTask<Pong>(new Pong(request.Id));
//      }
//}
```

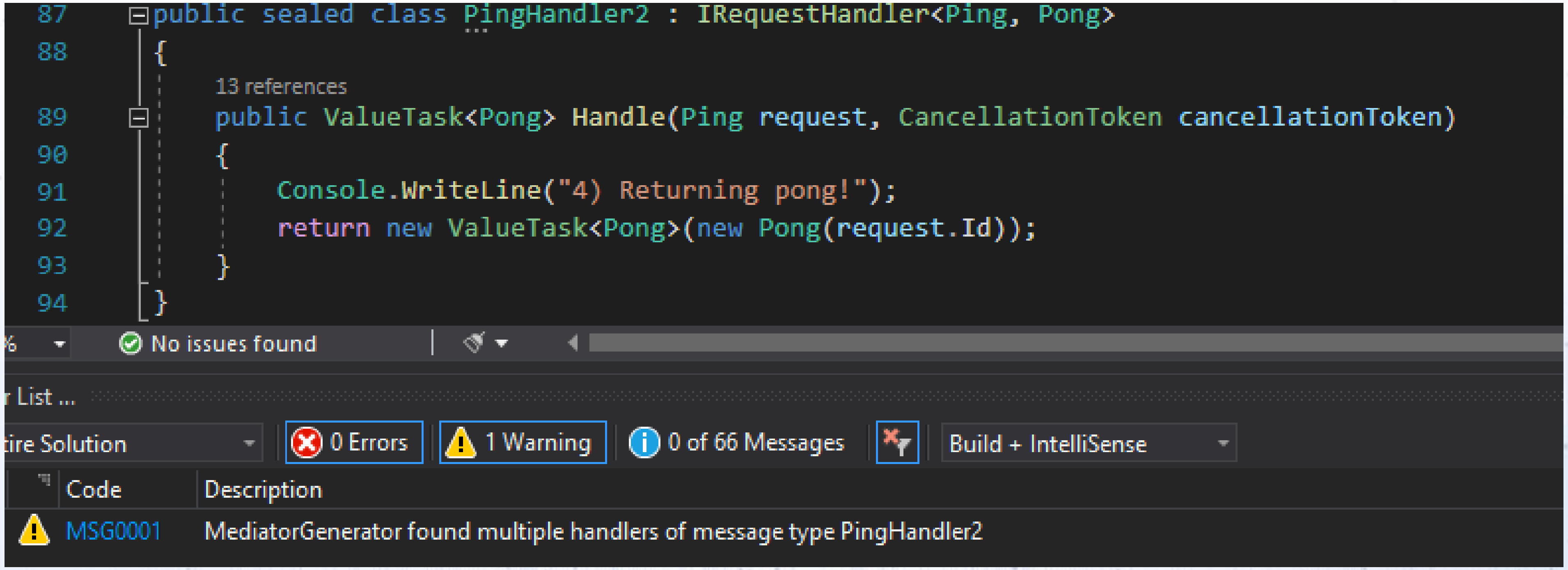
No issues found

List

ire Solution | 0 Errors | 1 Warning | 0 of 64 Messages | Build + IntelliSense

Code	Description
MSG0005	MediatorGenerator found message without any registered handler: Ping

Mediator



```
public sealed class PingHandler2 : IRequestHandler<Ping, Pong>
{
    13 references
    public ValueTask<Pong> Handle(Ping request, CancellationToken cancellationToken)
    {
        Console.WriteLine("4) Returning pong!");
        return new ValueTask<Pong>(new Pong(request.Id));
    }
}
```

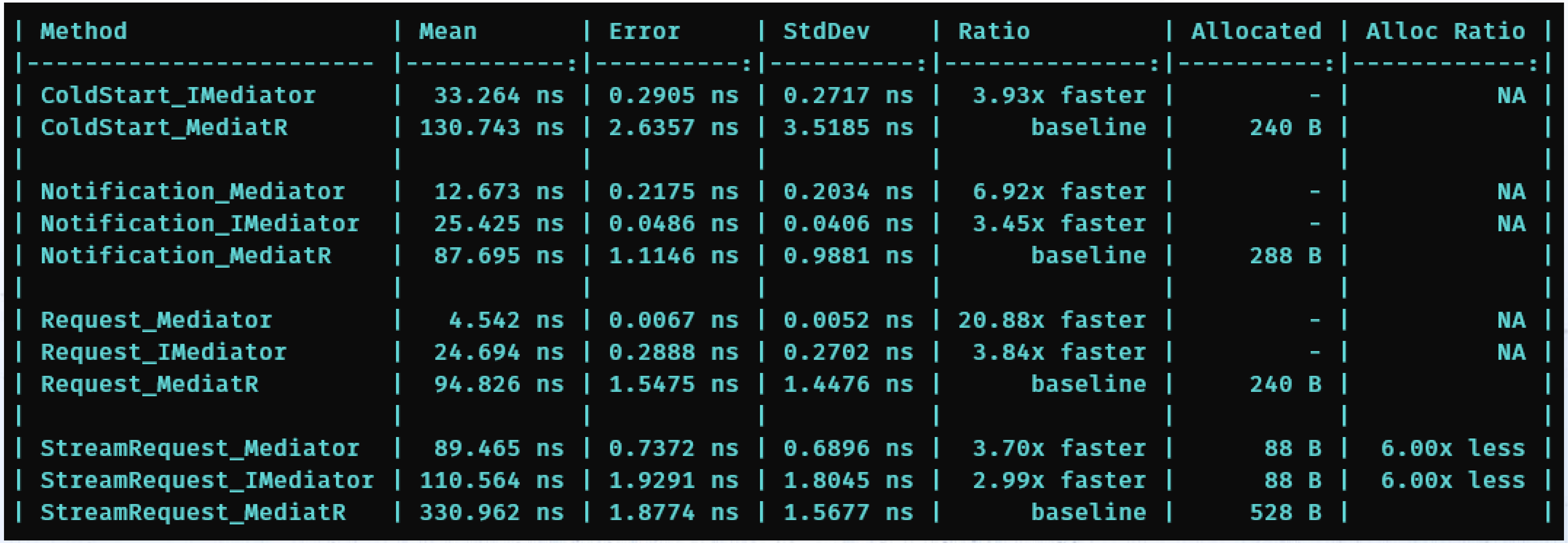
No issues found

r List ...

ire Solution | 0 Errors | 1 Warning | 0 of 66 Messages | Build + IntelliSense

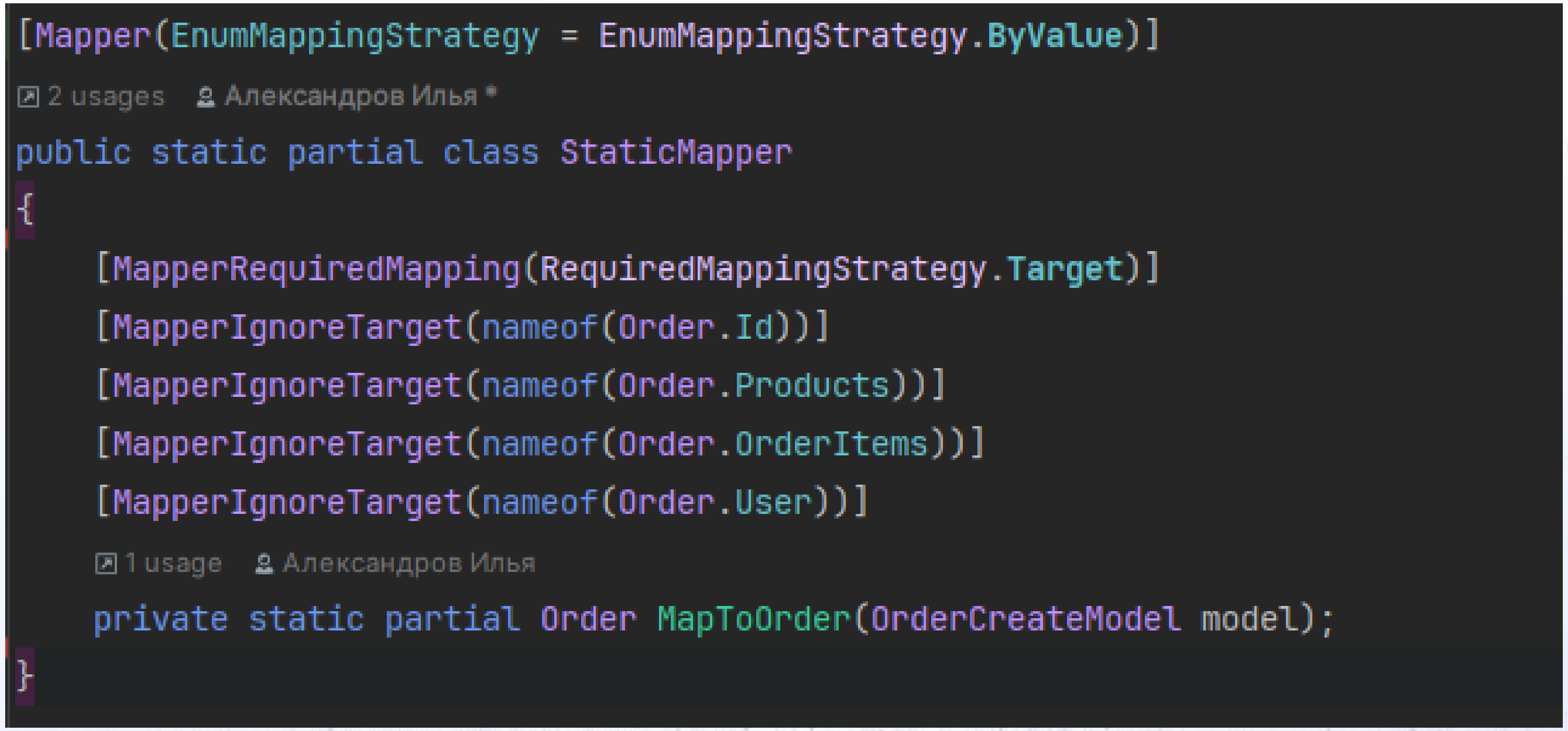
Code	Description
MSG0001	MediatorGenerator found multiple handlers of message type PingHandler2

Mediator



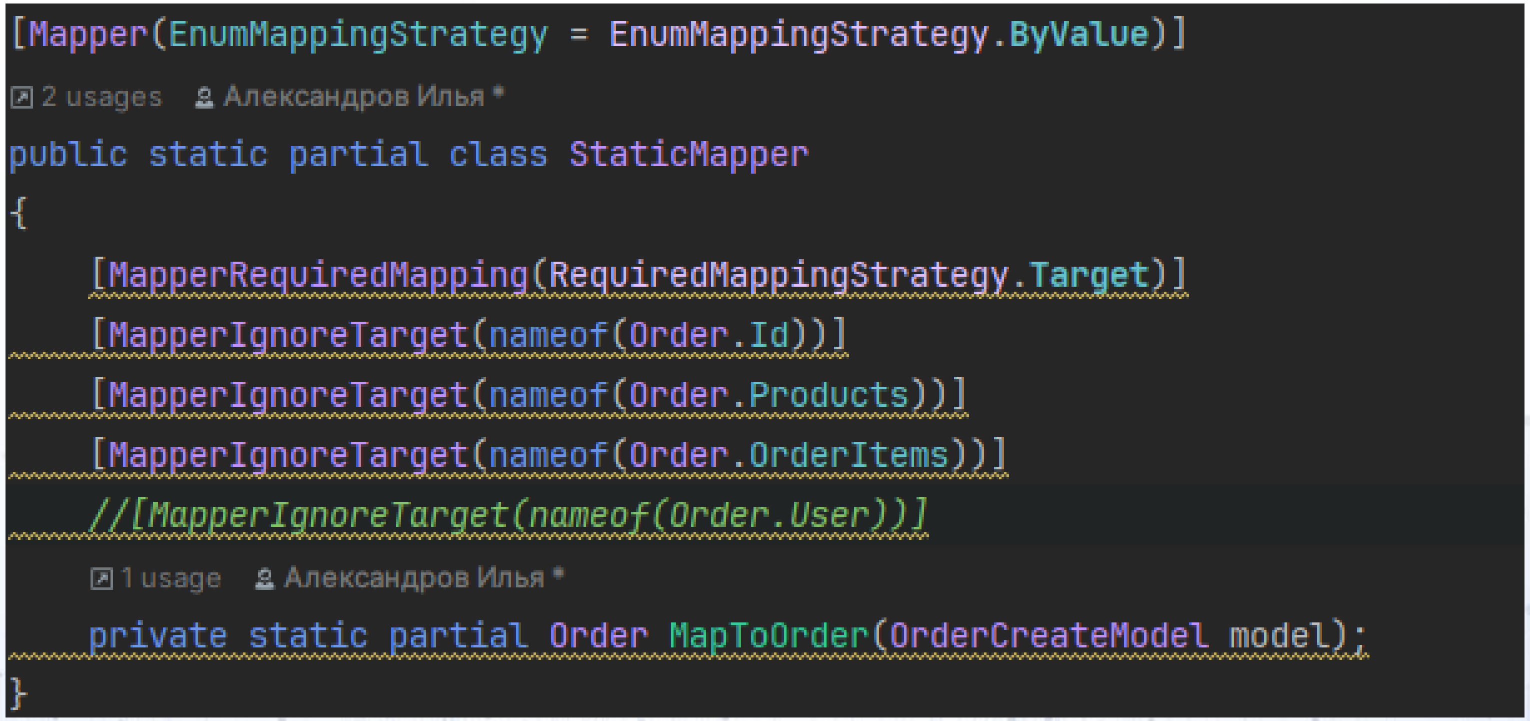
```
| Method                  | Mean       | Error     | StdDev    | Ratio         | Allocated | Alloc Ratio |
|------------------------ |-----------:|----------:|----------:|--------------:|----------:|------------:|
| ColdStart_IMediator     |  33.264 ns | 0.2905 ns | 0.2717 ns |  3.93x faster |         - |          NA |
| ColdStart_MediatR       | 130.743 ns | 2.6357 ns | 3.5185 ns |      baseline |     240 B |             |
|                         |            |           |           |               |           |             |
| Notification_Mediator   |  12.673 ns | 0.2175 ns | 0.2034 ns |  6.92x faster |         - |          NA |
| Notification_IMediator  |  25.425 ns | 0.0486 ns | 0.0406 ns |  3.45x faster |         - |          NA |
| Notification_MediatR    |  87.695 ns | 1.1146 ns | 0.9881 ns |      baseline |     288 B |             |
|                         |            |           |           |               |           |             |
| Request_Mediator        |   4.542 ns | 0.0067 ns | 0.0052 ns | 20.88x faster |         - |          NA |
| Request_IMediator       |  24.694 ns | 0.2888 ns | 0.2702 ns |  3.84x faster |         - |          NA |
| Request_MediatR         |  94.826 ns | 1.5475 ns | 1.4476 ns |      baseline |     240 B |             |
|                         |            |           |           |               |           |             |
| StreamRequest_Mediator  |  89.465 ns | 0.7372 ns | 0.6896 ns |  3.70x faster |      88 B |  6.00x less |
| StreamRequest_IMediator | 110.564 ns | 1.9291 ns | 1.8045 ns |  2.99x faster |      88 B |  6.00x less |
| StreamRequest_MediatR   | 330.962 ns | 1.8774 ns | 1.5677 ns |      baseline |     528 B |             |
```

Mapperly

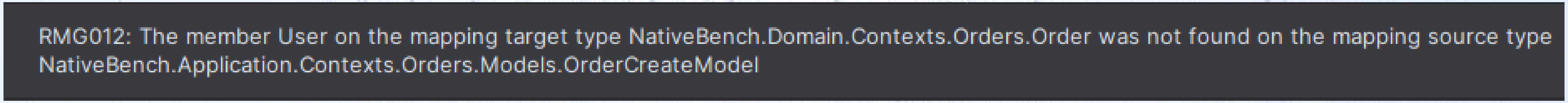


```
[Mapper(EnumMappingStrategy = EnumMappingStrategy.ByValue)]
2 usages   Александров Илья *
public static partial class StaticMapper
{
    [MapperRequiredMapping(RequiredMappingStrategy.Target)]
    [MapperIgnoreTarget(nameof(Order.Id))]
    [MapperIgnoreTarget(nameof(Order.Products))]
    [MapperIgnoreTarget(nameof(Order.OrderItems))]
    [MapperIgnoreTarget(nameof(Order.User))]
    1 usage   Александров Илья
    private static partial Order MapToOrder(OrderCreateModel model);
}
```

Mapperly

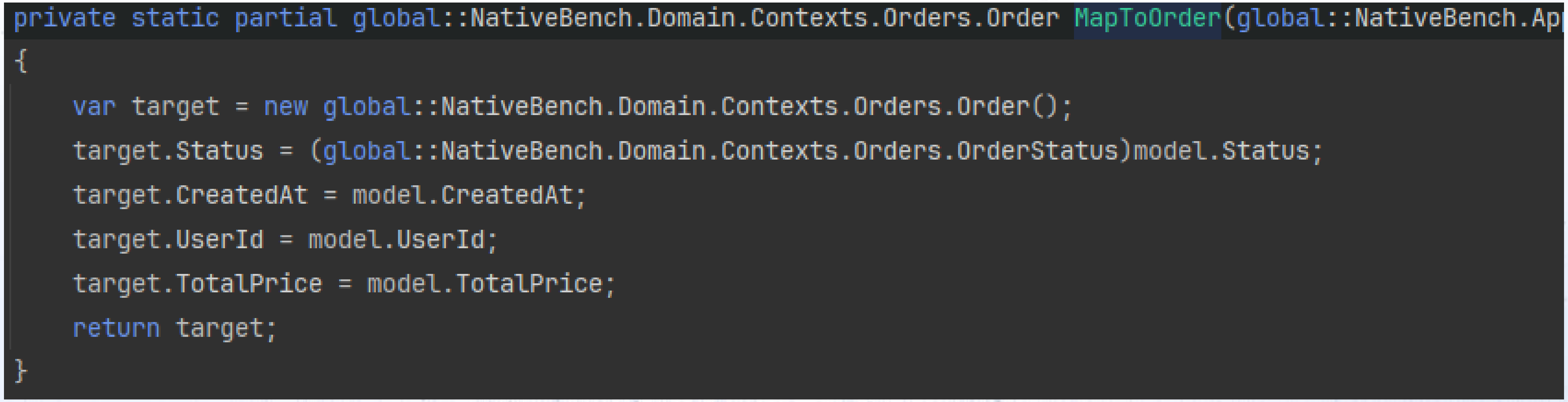


```
[Mapper(EnumMappingStrategy = EnumMappingStrategy.ByValue)]
2 usages   Александров Илья *
public static partial class StaticMapper
{
    [MapperRequiredMapping(RequiredMappingStrategy.Target)]
    [MapperIgnoreTarget(nameof(Order.Id))]
    [MapperIgnoreTarget(nameof(Order.Products))]
    [MapperIgnoreTarget(nameof(Order.OrderItems))]
    //[MapperIgnoreTarget(nameof(Order.User))]
    1 usage   Александров Илья *
    private static partial Order MapToOrder(OrderCreateModel model);
}
```



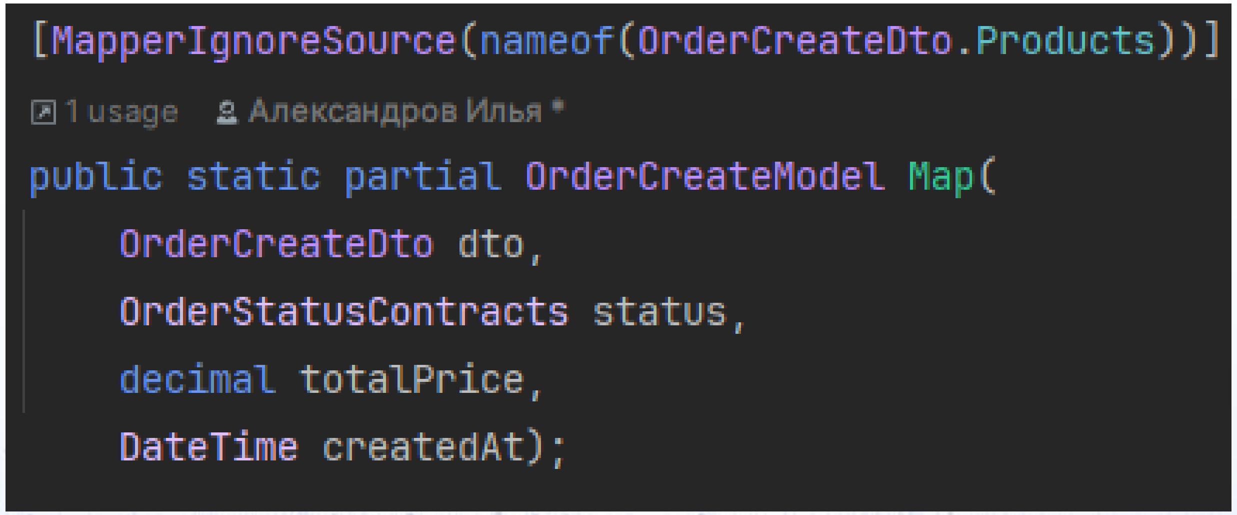
RMG012: The member User on the mapping target type NativeBench.Domain.Contexts.Orders.Order was not found on the mapping source type NativeBench.Application.Contexts.Orders.Models.OrderCreateModel

Mapperly



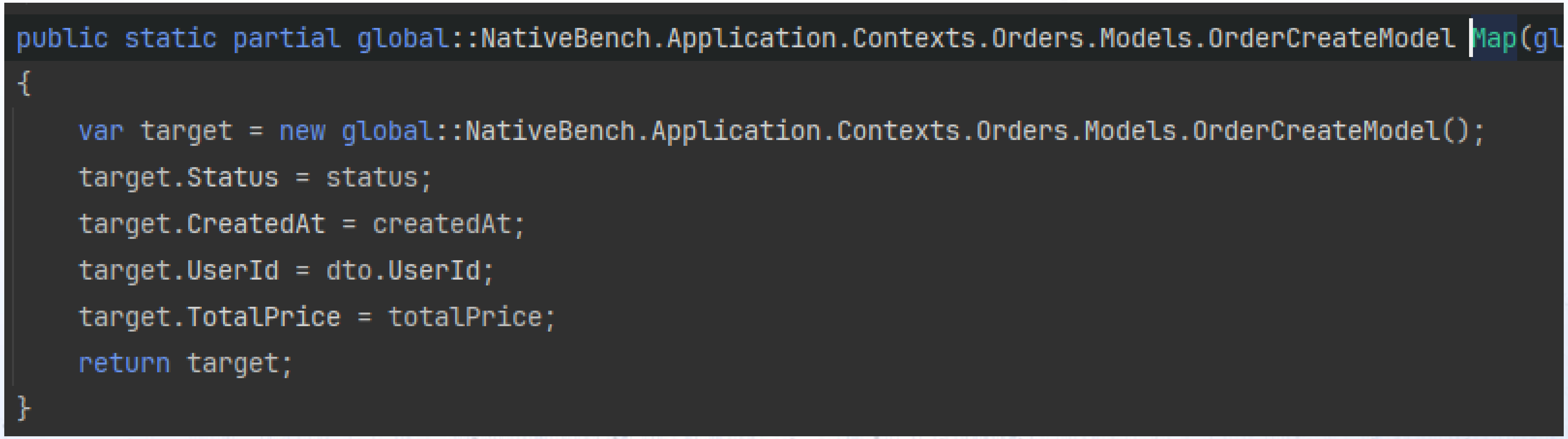
```
private static partial global::NativeBench.Domain.Contexts.Orders.Order MapToOrder(global::NativeBench.Ap
{
    var target = new global::NativeBench.Domain.Contexts.Orders.Order();
    target.Status = (global::NativeBench.Domain.Contexts.Orders.OrderStatus)model.Status;
    target.CreatedAt = model.CreatedAt;
    target.UserId = model.UserId;
    target.TotalPrice = model.TotalPrice;
    return target;
}
```

Mapperly



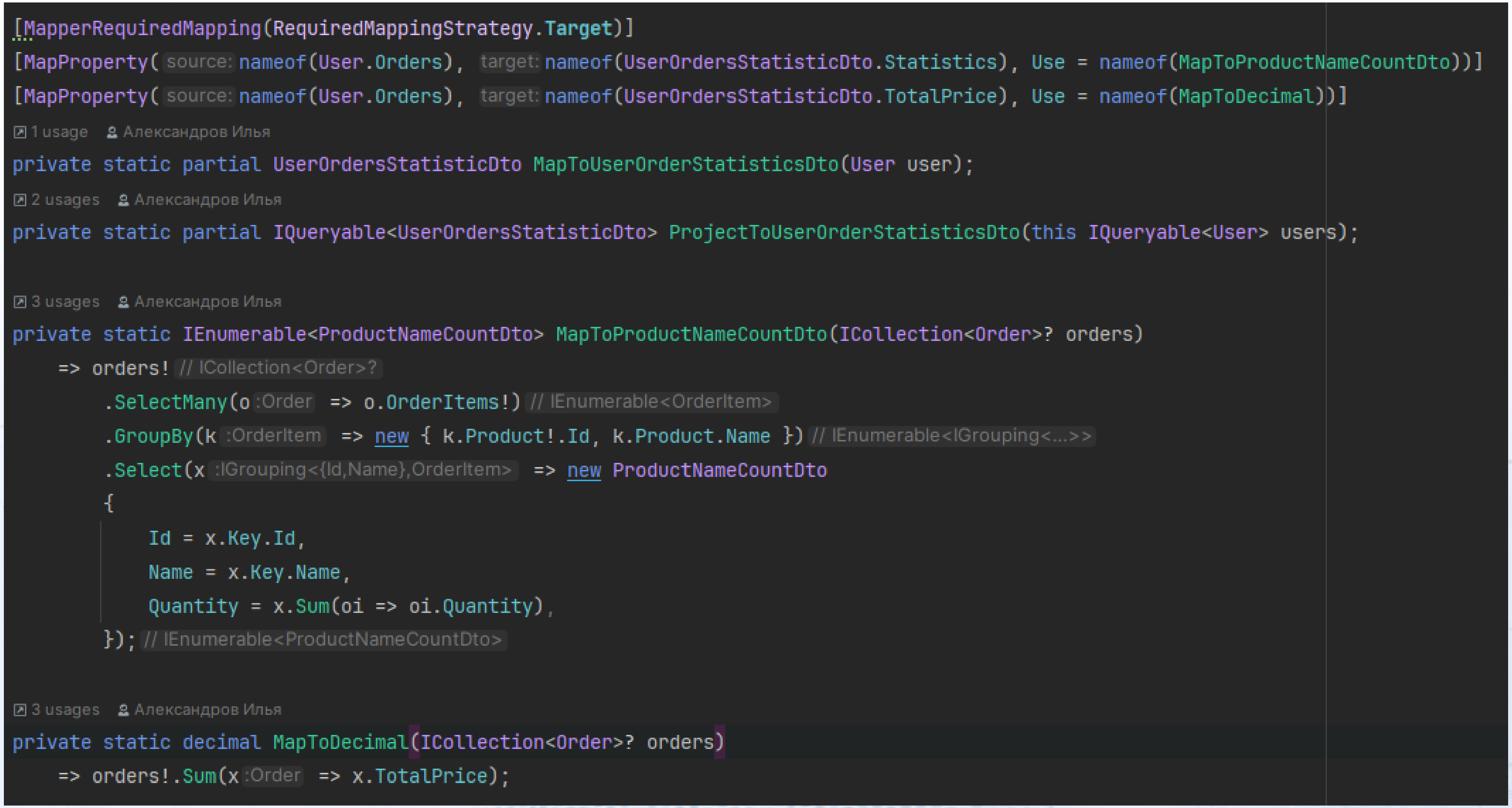
```
[MapperIgnoreSource(nameof(OrderCreateDto.Products))]
1 usage   Александров Илья *
public static partial OrderCreateModel Map(
    OrderCreateDto dto,
    OrderStatusContracts status,
    decimal totalPrice,
    DateTime createdAt);
```

Mapperly



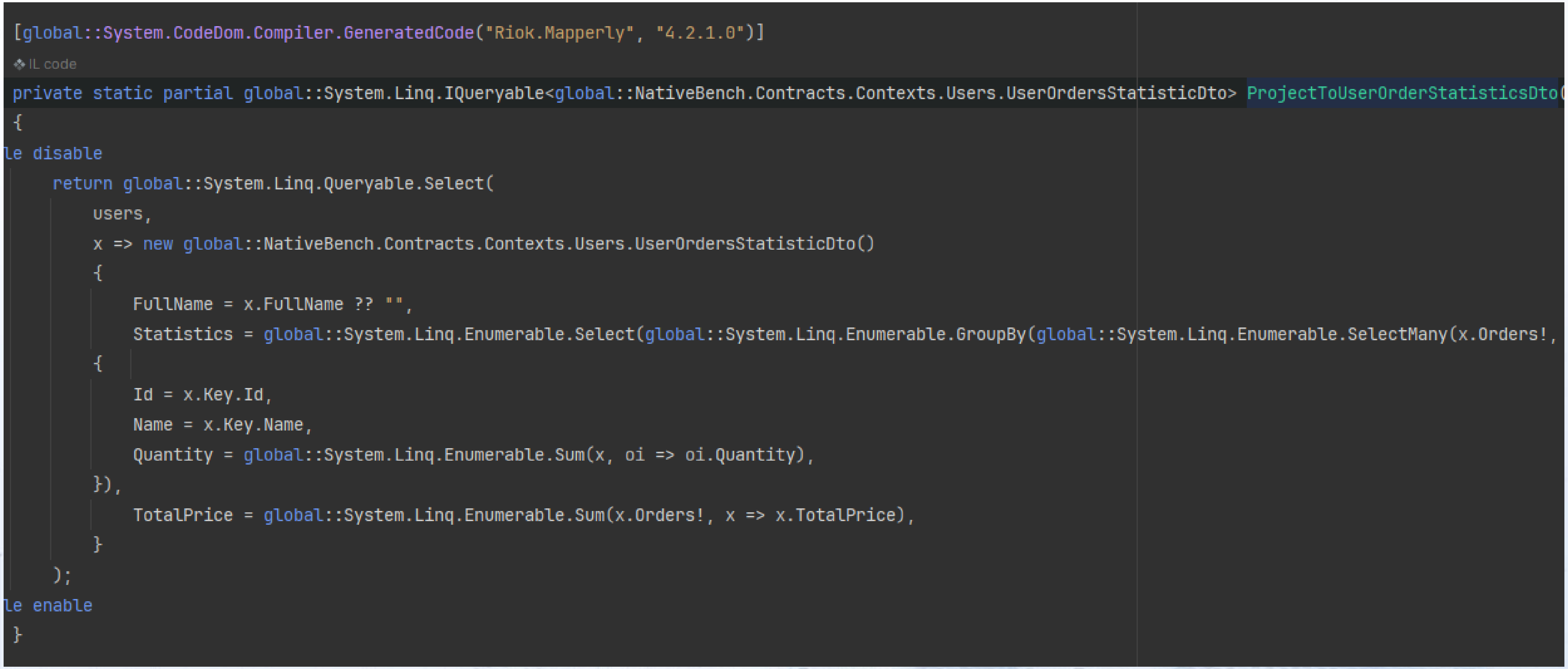
```
public static partial global::NativeBench.Application.Contexts.Orders.Models.OrderCreateModel Map(gl
{
    var target = new global::NativeBench.Application.Contexts.Orders.Models.OrderCreateModel();
    target.Status = status;
    target.CreatedAt = createdAt;
    target.UserId = dto.UserId;
    target.TotalPrice = totalPrice;
    return target;
}
```

Mapperly



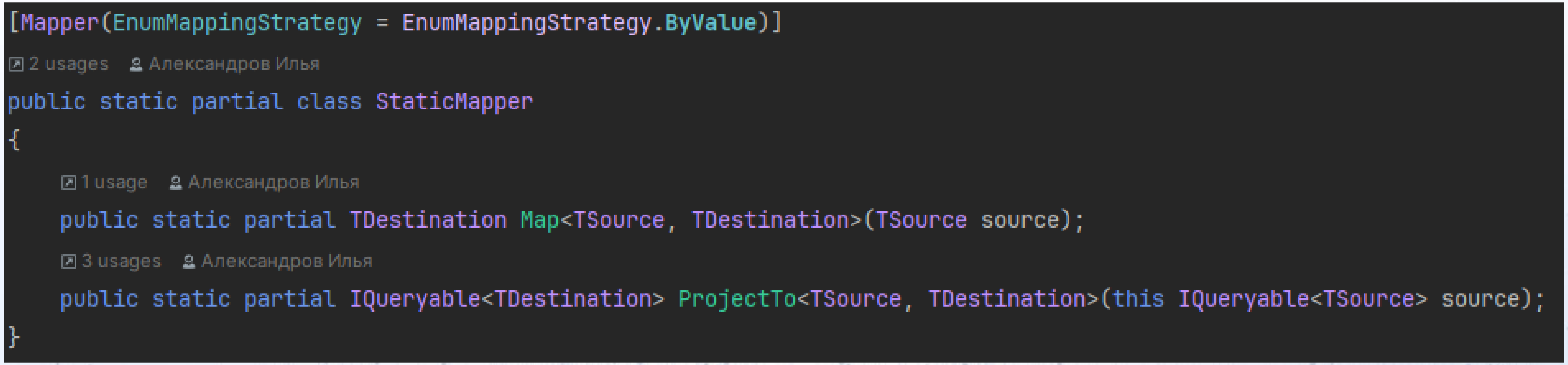
```
[MapperRequiredMapping(RequiredMappingStrategy.Target)]
[MapProperty( source: nameof(User.Orders),  target: nameof(UserOrdersStatisticDto.Statistics), Use = nameof(MapToProductNameCountDto))]
[MapProperty( source: nameof(User.Orders),  target: nameof(UserOrdersStatisticDto.TotalPrice), Use = nameof(MapToDecimal))]
1 usage  Александров Илья
private static partial UserOrdersStatisticDto MapToUserOrderStatisticsDto(User user);
2 usages  Александров Илья
private static partial IQueryable<UserOrdersStatisticDto> ProjectToUserOrderStatisticsDto(this IQueryable<User> users);

3 usages  Александров Илья
private static IEnumerable<ProductNameCountDto> MapToProductNameCountDto(ICollection<Order>? orders)
    => orders! // ICollection<Order>?
        .SelectMany(o :Order  => o.OrderItems!) // IEnumerable<OrderItem>
        .GroupBy(k :OrderItem  => new { k.Product!.Id, k.Product.Name }) // IEnumerable<IGrouping<...>>
        .Select(x :IGrouping<{Id,Name},OrderItem>  => new ProductNameCountDto
        {
            Id = x.Key.Id,
            Name = x.Key.Name,
            Quantity = x.Sum(oi => oi.Quantity),
        }); // IEnumerable<ProductNameCountDto>

3 usages  Александров Илья
private static decimal MapToDecimal(ICollection<Order>? orders)
    => orders!.Sum(x :Order  => x.TotalPrice);
```

Mapperly

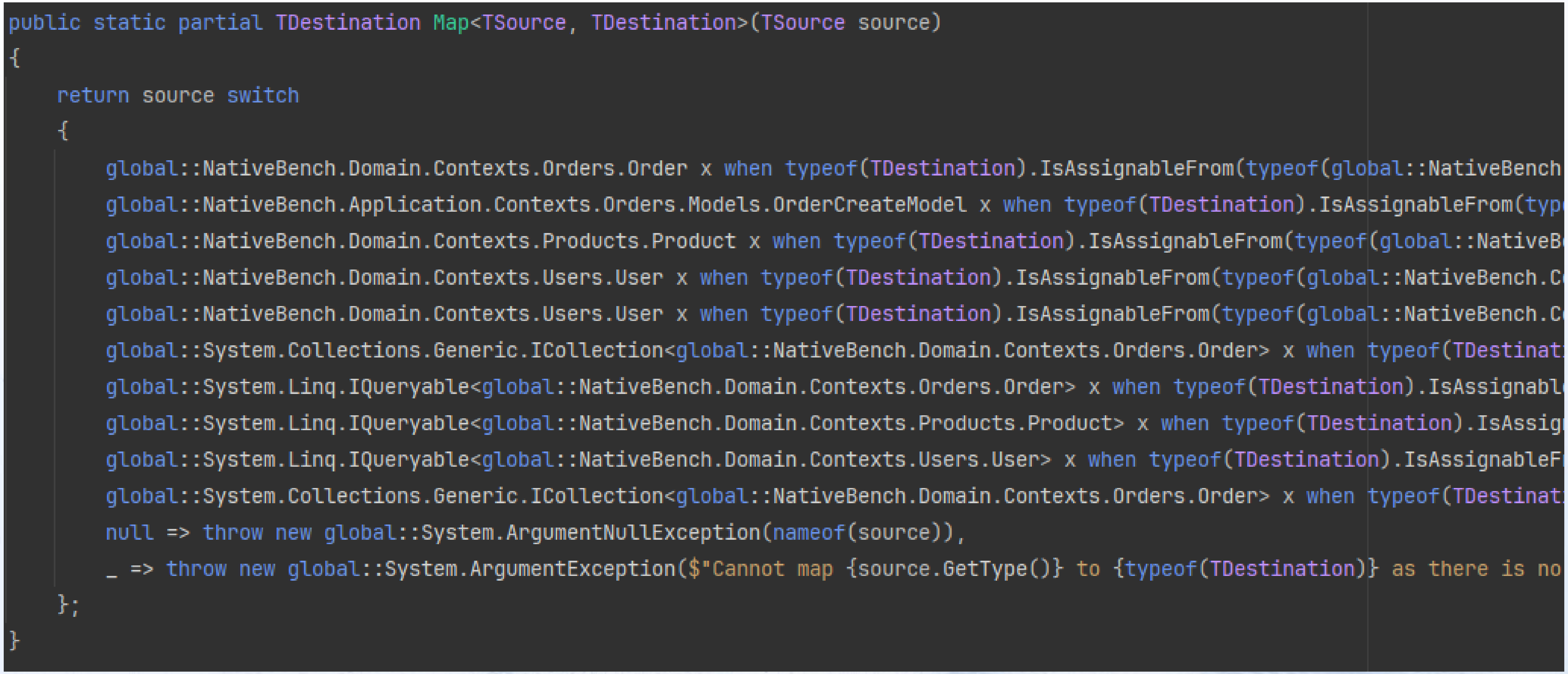


```
[global::System.CodeDom.Compiler.GeneratedCode("Riok.Mapperly", "4.2.1.0")]
IL code
private static partial global::System.Linq.IQueryable<global::NativeBench.Contracts.Contexts.Users.UserOrdersStatisticDto> ProjectToUserOrderStatisticsDto(
{
le disable
    return global::System.Linq.Queryable.Select(
        users,
        x => new global::NativeBench.Contracts.Contexts.Users.UserOrdersStatisticDto()
        {
            FullName = x.FullName ?? "",
            Statistics = global::System.Linq.Enumerable.Select(global::System.Linq.Enumerable.GroupBy(global::System.Linq.Enumerable.SelectMany(x.Orders!,
        {
            Id = x.Key.Id,
            Name = x.Key.Name,
            Quantity = global::System.Linq.Enumerable.Sum(x, oi => oi.Quantity),
        }),
            TotalPrice = global::System.Linq.Enumerable.Sum(x.Orders!, x => x.TotalPrice),
        }
    );
le enable
}
```

Mapperly

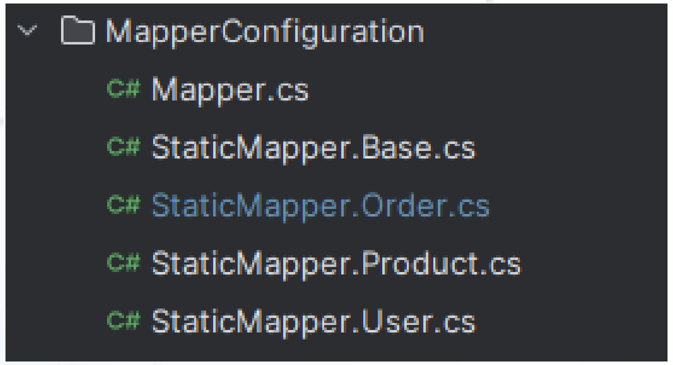

```
[Mapper(EnumMappingStrategy = EnumMappingStrategy.ByValue)]
2 usages   Александров Илья
public static partial class StaticMapper
{
    1 usage   Александров Илья
    public static partial TDestination Map<TSource, TDestination>(TSource source);
    3 usages   Александров Илья
    public static partial IQueryable<TDestination> ProjectTo<TSource, TDestination>(this IQueryable<TSource> source);
}
```

Mapperly

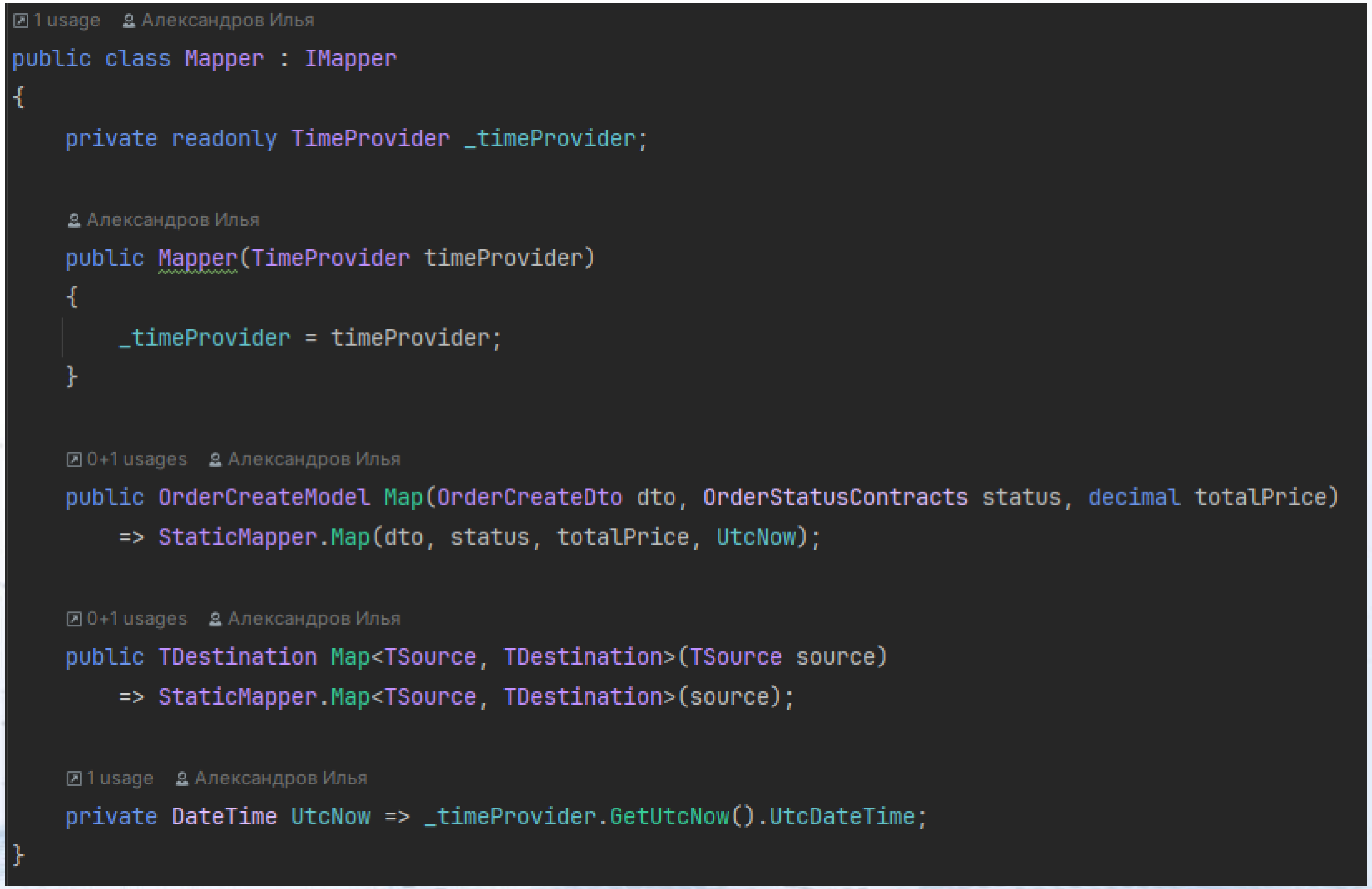


```
public static partial TDestination Map<TSource, TDestination>(TSource source)
{
    return source switch
    {
        global::NativeBench.Domain.Contexts.Orders.Order x when typeof(TDestination).IsAssignableFrom(typeof(global::NativeBench
        global::NativeBench.Application.Contexts.Orders.Models.OrderCreateModel x when typeof(TDestination).IsAssignableFrom(typ
        global::NativeBench.Domain.Contexts.Products.Product x when typeof(TDestination).IsAssignableFrom(typeof(global::NativeB
        global::NativeBench.Domain.Contexts.Users.User x when typeof(TDestination).IsAssignableFrom(typeof(global::NativeBench.C
        global::NativeBench.Domain.Contexts.Users.User x when typeof(TDestination).IsAssignableFrom(typeof(global::NativeBench.C
        global::System.Collections.Generic.ICollection<global::NativeBench.Domain.Contexts.Orders.Order> x when typeof(TDestinat
        global::System.Linq.IQueryable<global::NativeBench.Domain.Contexts.Orders.Order> x when typeof(TDestination).IsAssignabl
        global::System.Linq.IQueryable<global::NativeBench.Domain.Contexts.Products.Product> x when typeof(TDestination).IsAssig
        global::System.Linq.IQueryable<global::NativeBench.Domain.Contexts.Users.User> x when typeof(TDestination).IsAssignableF
        global::System.Collections.Generic.ICollection<global::NativeBench.Domain.Contexts.Orders.Order> x when typeof(TDestinat
        null => throw new global::System.ArgumentNullException(nameof(source)),
        _ => throw new global::System.ArgumentException($"Cannot map {source.GetType()} to {typeof(TDestination)} as there is no
    };
}
```

Mapperly

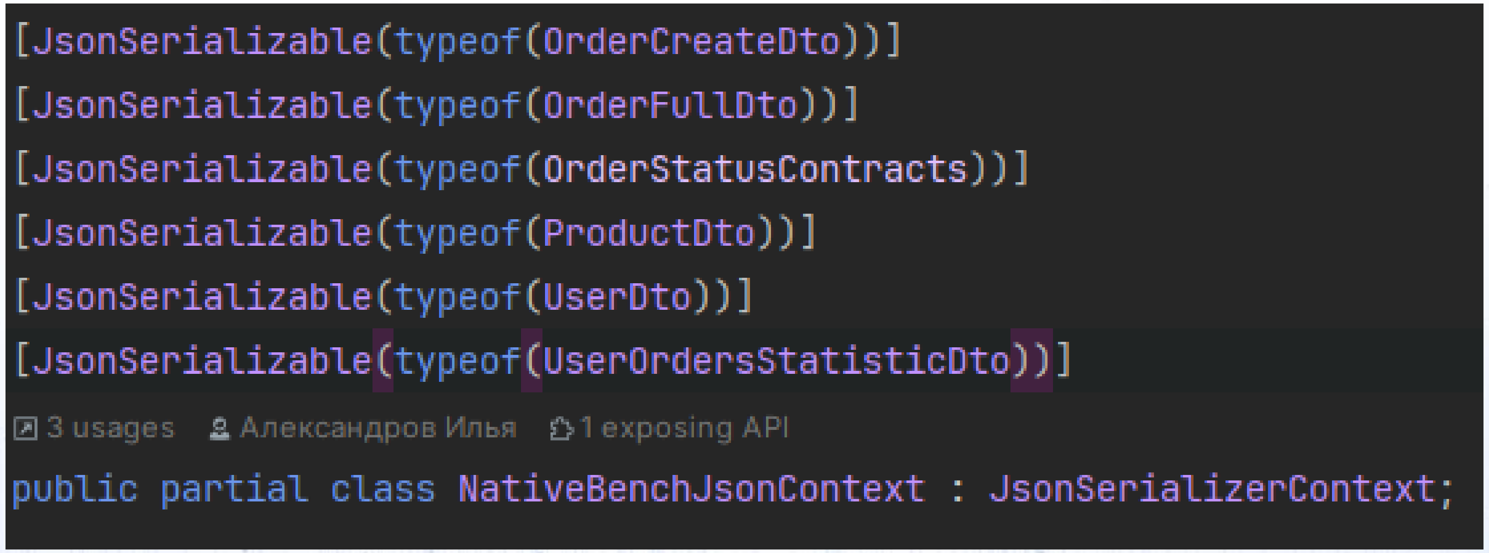


```
MapperConfiguration
    Mapper.cs
    StaticMapper.Base.cs
    StaticMapper.Order.cs
    StaticMapper.Product.cs
    StaticMapper.User.cs
```



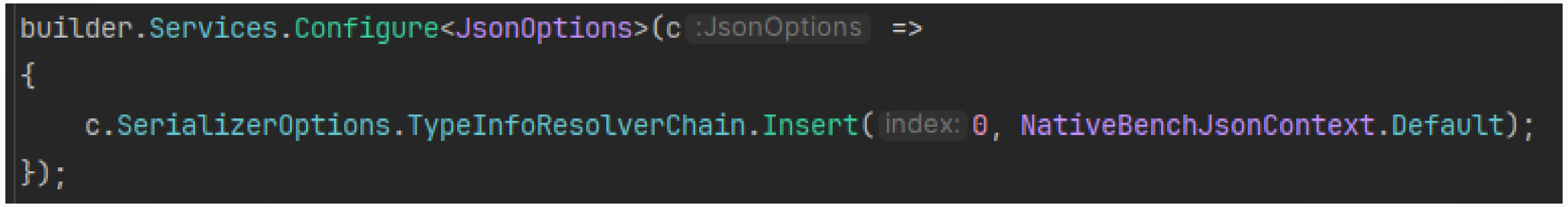
```
1 usage  Александров Илья
public class Mapper : IMapper
{
    private readonly TimeProvider _timeProvider;

    Александров Илья
    public Mapper(TimeProvider timeProvider)
    {
        _timeProvider = timeProvider;
    }

    0+1 usages  Александров Илья
    public OrderCreateModel Map(OrderCreateDto dto, OrderStatusContracts status, decimal totalPrice)
        => StaticMapper.Map(dto, status, totalPrice, UtcNow);

    0+1 usages  Александров Илья
    public TDestination Map<TSource, TDestination>(TSource source)
        => StaticMapper.Map<TSource, TDestination>(source);

    1 usage  Александров Илья
    private DateTime UtcNow => _timeProvider.GetUtcNow().UtcDateTime;
}
```

System.Text.Json

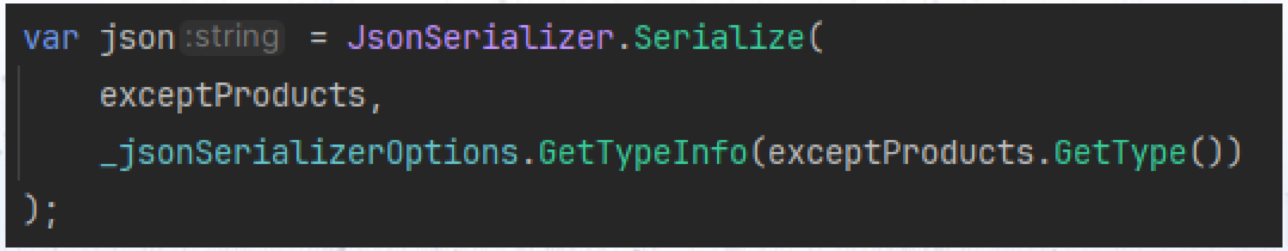


```
[JsonSerializable(typeof(OrderCreateDto))]
[JsonSerializable(typeof(OrderFullDto))]
[JsonSerializable(typeof(OrderStatusContracts))]
[JsonSerializable(typeof(ProductDto))]
[JsonSerializable(typeof(UserDto))]
[JsonSerializable(typeof(UserOrdersStatisticDto))]
3 usages  Александров Илья  1 exposing API
public partial class NativeBenchJsonContext : JsonSerializerContext;
```

System.Text.Json

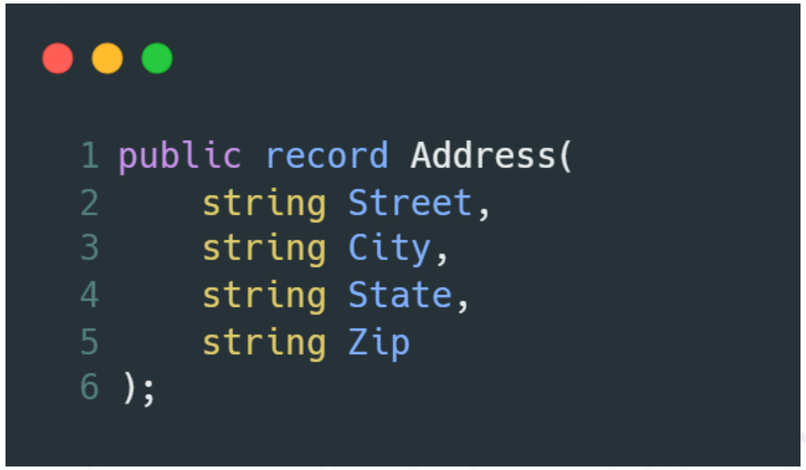


```
builder.Services.Configure<JsonOptions>(c :JsonOptions  =>
{
    c.SerializerOptions.TypeInfoResolverChain.Insert( index: 0, NativeBenchJsonContext.Default);
});
```

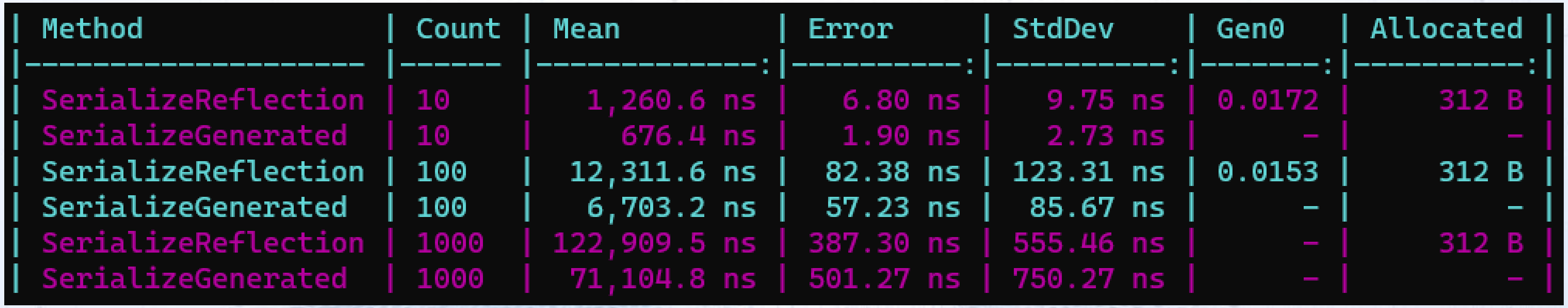


```
var json :string  = JsonSerializer.Serialize(
    exceptProducts,
    _jsonSerializerOptions.GetTypeInfo(exceptProducts.GetType())
);
```

System.Text.Json

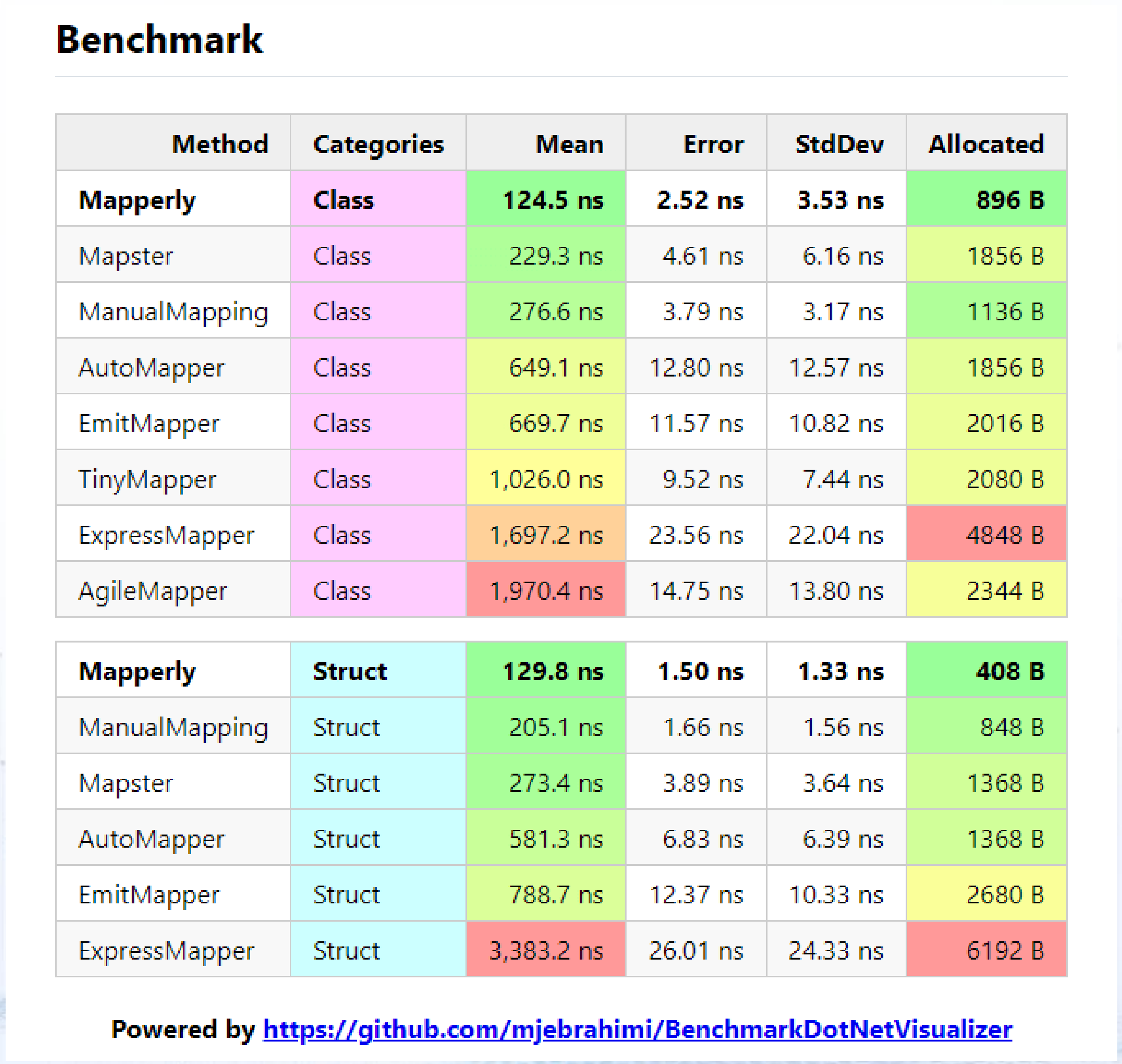


```
public record Address(
    string Street,
    string City,
    string State,
    string Zip
);
```



```
| Method              | Count |          Mean |      Error |     StdDev |   Gen0 | Allocated |
|-------------------- |------ |--------------:|-----------:|-----------:|-------:|----------:|
| SerializeReflection | 10    |    1,260.6 ns |    6.80 ns |    9.75 ns | 0.0172 |     312 B |
| SerializeGenerated  | 10    |      676.4 ns |    1.90 ns |    2.73 ns |      - |         - |
| SerializeReflection | 100   |   12,311.6 ns |   82.38 ns |  123.31 ns | 0.0153 |     312 B |
| SerializeGenerated  | 100   |    6,703.2 ns |   57.23 ns |   85.67 ns |      - |         - |
| SerializeReflection | 1000  |  122,909.5 ns |  387.30 ns |  555.46 ns |      - |     312 B |
| SerializeGenerated  | 1000  |   71,104.8 ns |  501.27 ns |  750.27 ns |      - |         - |
```

Mapperly

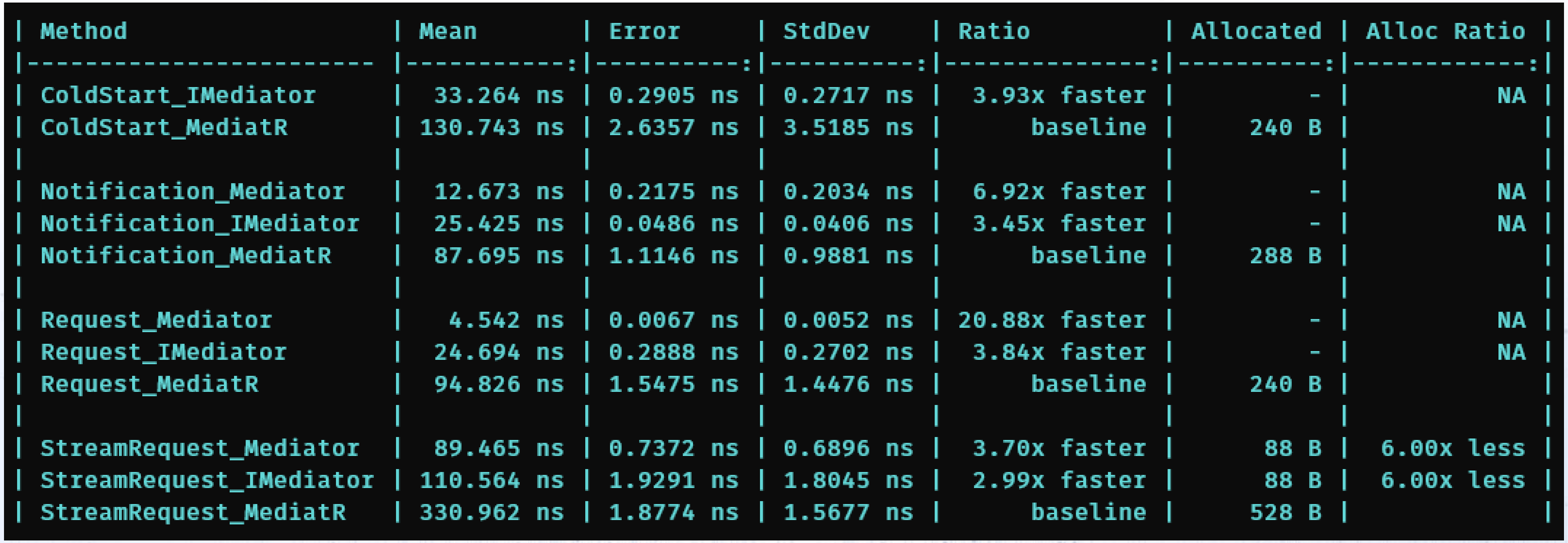


Benchmark

Method	Categories	Mean	Error	StdDev	Allocated
Mapperly	**Class**	**124.5 ns**	**2.52 ns**	**3.53 ns**	**896 B**
Mapster	Class	229.3 ns	4.61 ns	6.16 ns	1856 B
ManualMapping	Class	276.6 ns	3.79 ns	3.17 ns	1136 B
AutoMapper	Class	649.1 ns	12.80 ns	12.57 ns	1856 B
EmitMapper	Class	669.7 ns	11.57 ns	10.82 ns	2016 B
TinyMapper	Class	1,026.0 ns	9.52 ns	7.44 ns	2080 B
ExpressMapper	Class	1,697.2 ns	23.56 ns	22.04 ns	4848 B
AgileMapper	Class	1,970.4 ns	14.75 ns	13.80 ns	2344 B
Mapperly	**Struct**	**129.8 ns**	**1.50 ns**	**1.33 ns**	**408 B**
ManualMapping	Struct	205.1 ns	1.66 ns	1.56 ns	848 B
Mapster	Struct	273.4 ns	3.89 ns	3.64 ns	1368 B
AutoMapper	Struct	581.3 ns	6.83 ns	6.39 ns	1368 B
EmitMapper	Struct	788.7 ns	12.37 ns	10.33 ns	2680 B
ExpressMapper	Struct	3,383.2 ns	26.01 ns	24.33 ns	6192 B

Powered by https://github.com/mjebrahimi/BenchmarkDotNetVisualizer

Mediator



```
| Method                   | Mean       | Error      | StdDev     | Ratio          | Allocated | Alloc Ratio |
|------------------------- |-----------:|-----------:|-----------:|---------------:|----------:|------------:|
| ColdStart_IMediator      |  33.264 ns | 0.2905 ns | 0.2717 ns |  3.93x faster |         - |          NA |
| ColdStart_MediatR        | 130.743 ns | 2.6357 ns | 3.5185 ns |      baseline |     240 B |             |
|                          |            |            |            |                |           |             |
| Notification_Mediator    |  12.673 ns | 0.2175 ns | 0.2034 ns |  6.92x faster |         - |          NA |
| Notification_IMediator   |  25.425 ns | 0.0486 ns | 0.0406 ns |  3.45x faster |         - |          NA |
| Notification_MediatR     |  87.695 ns | 1.1146 ns | 0.9881 ns |      baseline |     288 B |             |
|                          |            |            |            |                |           |             |
| Request_Mediator         |   4.542 ns | 0.0067 ns | 0.0052 ns | 20.88x faster |         - |          NA |
| Request_IMediator        |  24.694 ns | 0.2888 ns | 0.2702 ns |  3.84x faster |         - |          NA |
| Request_MediatR          |  94.826 ns | 1.5475 ns | 1.4476 ns |      baseline |     240 B |             |
|                          |            |            |            |                |           |             |
| StreamRequest_Mediator   |  89.465 ns | 0.7372 ns | 0.6896 ns |  3.70x faster |      88 B |  6.00x less |
| StreamRequest_IMediator  | 110.564 ns | 1.9291 ns | 1.8045 ns |  2.99x faster |      88 B |  6.00x less |
| StreamRequest_MediatR    | 330.962 ns | 1.8774 ns | 1.5677 ns |      baseline |     528 B |             |
```

Ссылки

- https://andrewlock.net/exploring-dotnet-6-part-9-source-generator-updates-incremental-generators/
- https://andrewlock.net/exploring-the-dotnet-8-preview-exploring-the-new-minimal-api-source-generator/
- https://okyrylchuk.dev/blog/intro-to-serialization-with-source-generation-in-system-text-json/
- https://github.com/mjebrahimi/DotNet-Mappers-Benchmark

SolarLab>_

Спасибо за внимание!