Home Page - PresentationLayer

localhost:7101

PresentationLayer Privacy

Create New Product

Product1

Product1s description.

View Edit Delete $22,00

Product2

Product2s description.

View Edit Delete $33,00

Product3

Product3s description.

View Edit Delete $42,00

Product4

Product4s description.

View Edit Delete $100,00

Create Product - PresentationLay

localhost:7101/Product/CreateProduct

PresentationLayer Privacy

Name

NewProductName

Description

New Products Description

Price

333

Create

Back

Product Page - PresentationLaye

localhost:7101/Product/ProductPage?productId=1

PresentationLayer Privacy

Product Page

Product1

Product1s description.

Price:$ 22,00

Back to List

Edit Product - PresentationLayer

localhost:7101/Product/EditProduct?productId=2

PresentationLayer Privacy

Edit Product

Name

Product2

Description

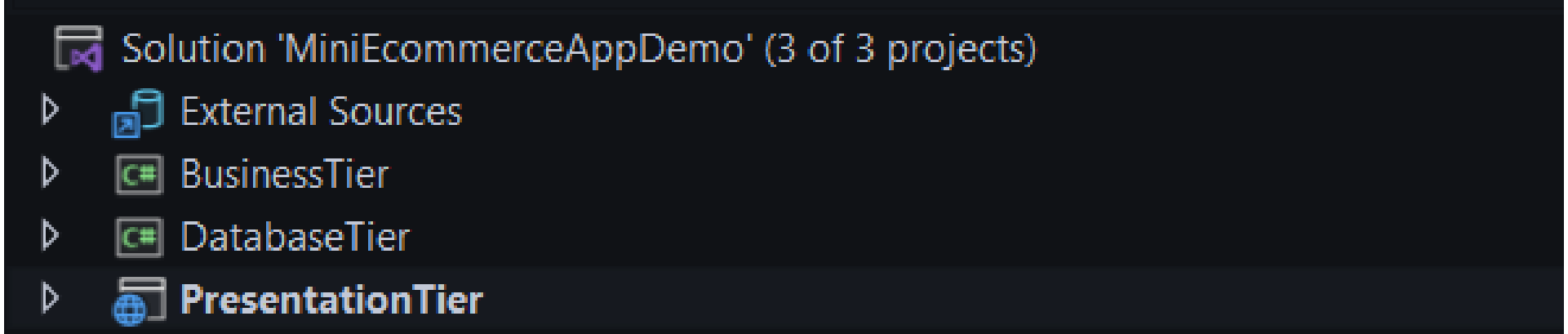
Product2s description.

Price

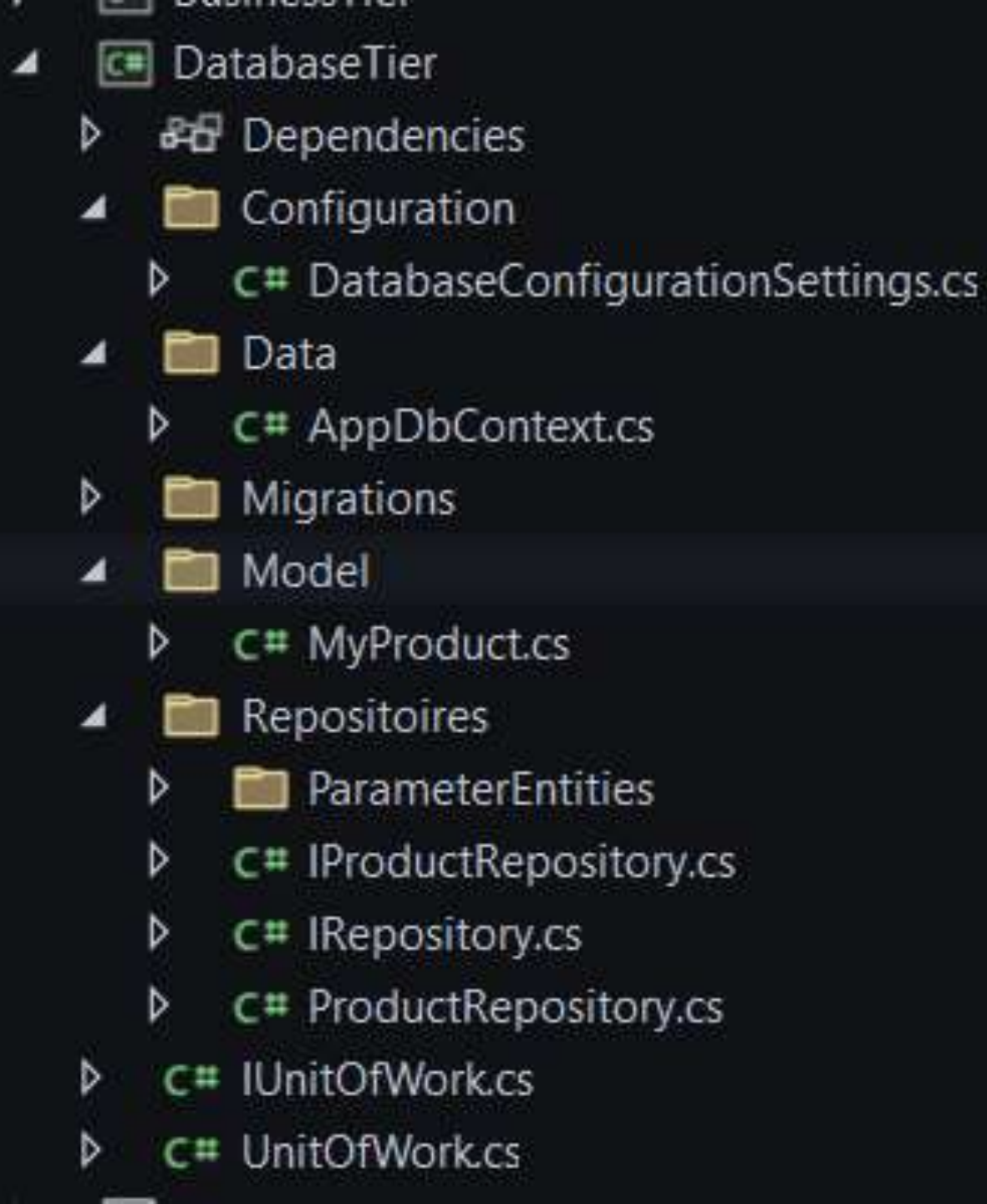
33,00

Update

Back to List



- Easy to maintain and understand the **large-scale applications**
- **Set the different developers** on each tier for the fast development
- Hosted in different **Physical Locations** due to independence of layers.


DatabaseTier
Dependencies
Configuration
DatabaseConfigurationSettings.cs
Data
AppDbContext.cs
Migrations
Model
MyProduct.cs
Repositoires
ParameterEntities
IProductRepository.cs
IRepository.cs
ProductRepository.cs
IUnitOfWork.cs
UnitOfWork.cs

```
public class MyProduct
{
    [Key]

    public int ProductId { get; set; }
    [Required]
    [MaxLength(30)]

    public string? Name { get; set; }
    [Range(0, double.MaxValue)]
    2 references
    public decimal Price { get; set; }
    [MaxLength(1000)]
    4 references
    public string? Description { get; set; }

}
```

Products

🔑	ProductId
	Name
	Price
	Description

```
13 references
public class AppDbContext : DbContext
{
    private readonly DatabaseConfigurationSettings _databaseConfigurationSettings;
    6 references
    public DbSet<MyProduct> Products { get; set; }

    0 references
    public AppDbContext(IOptions<DatabaseConfigurationSettings> databaseConfigurationSettings)
    {
        _databaseConfigurationSettings = databaseConfigurationSettings.Value;
    }


    0 references
    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        base.OnConfiguring(optionsBuilder);
        if(_databaseConfigurationSettings is null)
            throw new ArgumentNullException(nameof(_databaseConfigurationSettings));

        optionsBuilder.UseSqlServer(_databaseConfigurationSettings.ConnectionString);
    }

}
```

```
namespace DatabaseLayer.Configuration;

#nullable disable
3 references
public class DatabaseConfigurationSettings
{
    1 reference
    public string ConnectionString { get; set; }
}
```

```
{
  "DatabaseConfigurationSettings": {
    "ConnectionString": "Server=(localdb)\\WebTechFinalProjServer;Database=WebTechFinalProjDB;Trusted_Connection=True;"
  }
}
```

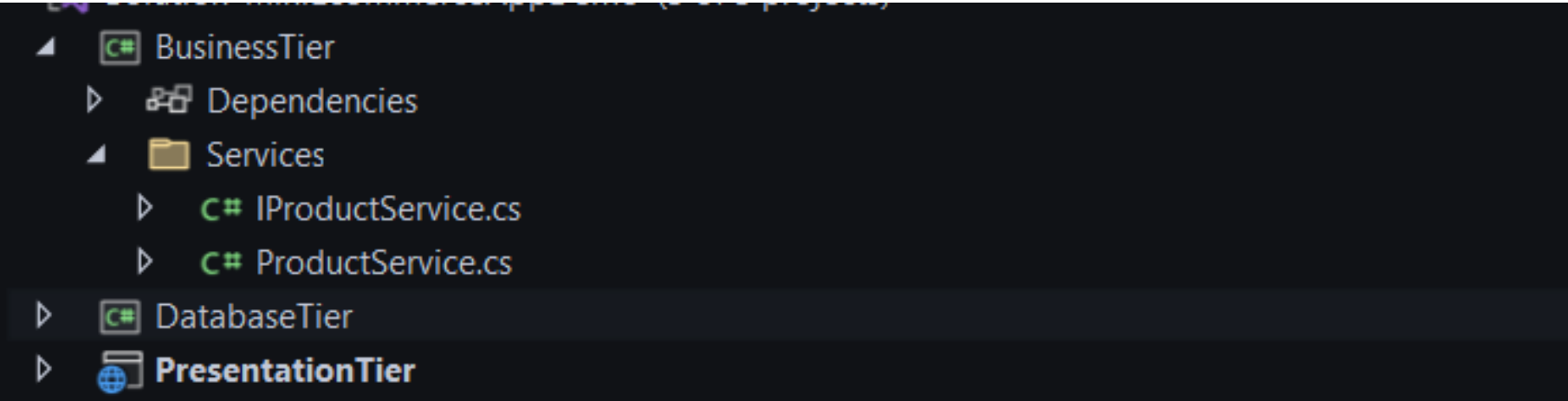
```
public interface IRepository<RepoEntitiyType, KeyType> where RepoEntitiyType : class
{

    public AppDbContext dbContext { get; set; }
    1 reference
    public ILogger<IRepository<RepoEntitiyType, KeyType>> logger { get; set; }

    2 references
    public abstract Task<IEnumerable<RepoEntitiyType>?> GetAllAsync();
    2 references
    public abstract Task<RepoEntitiyType?> GetAsync(KeyType id);
    2 references
    public abstract Task<RepoEntitiyType?> CreateAsync(IParameterEntity parameterEntity);
    2 references
    public abstract Task<RepoEntitiyType?> UpdateAsync(IParameterEntity parameterEntity);
    2 references
    public abstract Task<RepoEntitiyType?> RemoveAsync(KeyType id);
}
```

```
public interface IProductRepository : IRepository<MyProduct, int>
{
}
```

```
public class ProductRepository : IProductRepository
{
    9 references
    public AppDbContext dbContext { get; set; }
    7 references
    public ILogger<IRepository<MyProduct, int>> logger { get; set; }


    0 references
    public ProductRepository(AppDbContext Context, ILogger<IRepository<MyProduct, int>> Logger)
    {
        dbContext = Context;
        logger = Logger;
    }


    2 references
    public async Task<MyProduct?> CreateAsync(IParameterEntity parameterEntity)
    {
        if(parameterEntity is CreateProductParameterEntity productParameterEntity)
        {
            var product = new MyProduct()
            {
                Name = productParameterEntity.Name,
                Price = productParameterEntity.Price,
                Description = productParameterEntity.Description
            };

            var newProduct = await dbContext.Products.AddAsync(product);
            logger.BeginScope("ProductRepository.CreateAsync: newProduct.Entity = {newProduct.Entity}", newProduct.Entity);
            return newProduct.Entity;
        }

        logger.LogError("ProductRepository.CreateAsync: parameterEntity is not of type CreateProductParameterEntity");
        return null;
    }
```

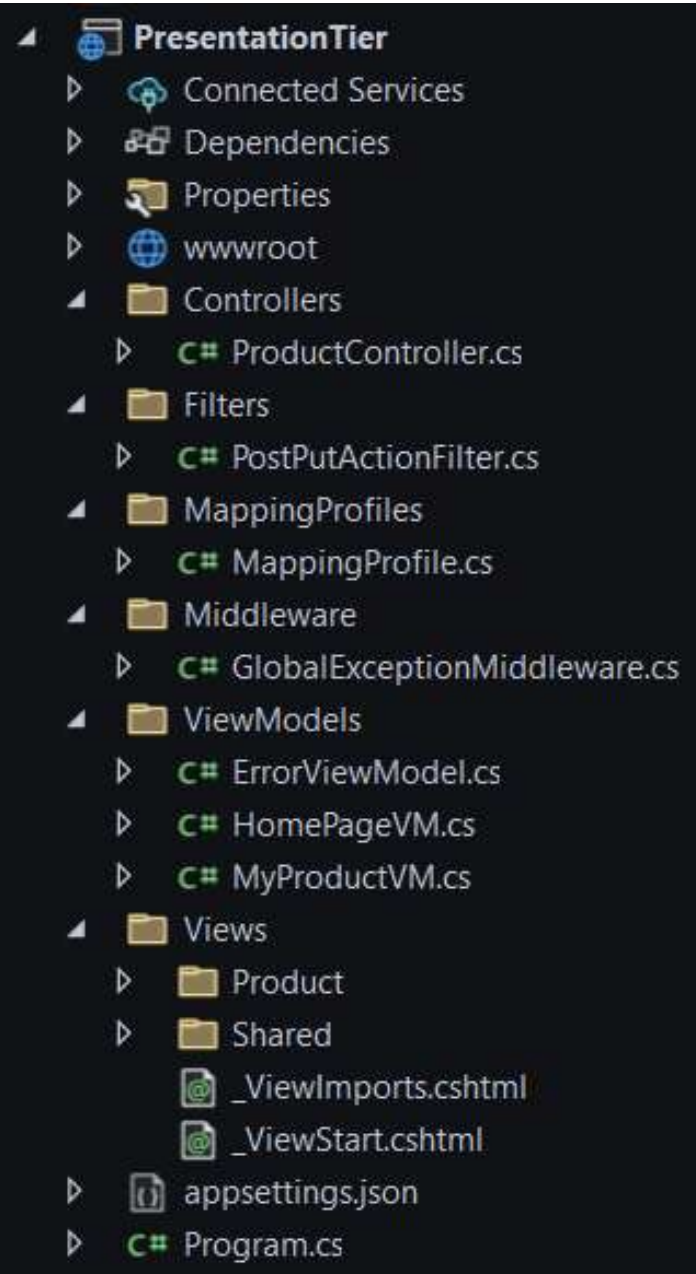
```
public interface IUnitOfWork
{
    7 references
    IProductRepository ProductRepository { get; set; }
    4 references
    Task SaveChangesAsync();
}
```

```
public class UnitOfWork : IUnitOfWork, IDisposable
{
    private readonly AppDbContext _context;
    7 references
    public IProductRepository ProductRepository { get; set; }


    0 references
    public UnitOfWork(AppDbContext context, IProductRepository productRepository)
    {
        _context = context;
        ProductRepository = productRepository;
    }


    4 references
    public async Task SaveChangesAsync()
    {
        await _context.SaveChangesAsync();
    }

    0 references
    public void Dispose()
    {
        _context.Dispose();
    }

}
```


BusinessTier
Dependencies
Services
IProductService.cs
ProductService.cs
DatabaseTier
PresentationTier

```
public interface IProductService
{

    Task<IEnumerable<MyProduct>?> GetAllProductsAsync();
    2 references
    Task<MyProduct?> AddNewProductAsync(MyProduct myProduct);
    2 references
    Task<MyProduct?> UpdateProductAsync(MyProduct myProduct);
    2 references
    Task<MyProduct?> DeleteProductAsync(int id);
    3 references
    Task<MyProduct?> GetProductByIdAsync(int id);
}
```

```
public class ProductService : IProductService
{
    private readonly IUnitOfWork _unitOfWork;


    0 references
    public ProductService(IUnitOfWork unitOfWork)
    {
        _unitOfWork = unitOfWork;
    }


    2 references
    public async Task<MyProduct?> AddNewProductAsync(MyProduct myProduct)
    {
        var createProductParameterEntity = new CreateProductParameterEntity()
        {
            Name = myProduct.Name,
            Description = myProduct.Description,
            Price = myProduct.Price
        };

        var newProduct = await _unitOfWork.ProductRepository.CreateAsync(createProductParameterEntity);
        await _unitOfWork.SaveChangesAsync();
        return newProduct;
```


PresentationTier
Connected Services
Dependencies
Properties
wwwroot
Controllers
ProductController.cs
Filters
PostPutActionFilter.cs
MappingProfiles
MappingProfile.cs
Middleware
GlobalExceptionMiddleware.cs
ViewModels
ErrorViewModel.cs
HomePageVM.cs
MyProductVM.cs
Views
Product
Shared
_ViewImports.cshtml
_ViewStart.cshtml
appsettings.json
Program.cs

```
public class ProductController : Controller
{
    private readonly IProductService _productService;
    private readonly IMapper _mapper;


    0 references
    public ProductController(IProductService productService, IMapper mapper)...


    [HttpGet]
    0 references
    public async Task<IActionResult> EditProduct(int productId)
    {
        var product = await _productService.GetProductByIdAsync(productId);
        if (product == null)
            return View(null);
        var productVM = _mapper.Map<MyProductVM>(product);

        return View(productVM);
    }
    [HttpPost]
    [ValidateAntiForgeryToken]
    [PostPutActionFilter]
    0 references
    public async Task<IActionResult> EditProduct(MyProductVM myProductVM)
    {
        var myProduct = _mapper.Map<MyProduct>(myProductVM);
        var updatedProduct = await _productService.UpdateProductAsync(myProduct);
        if (updatedProduct == null)
            return View(myProductVM);

        return RedirectToAction(nameof(ProductPage),new { productId =myProduct.ProductId});
    }
```

```
public class PostPutActionFilter : ActionFilterAttribute
{
    0 references
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        //Get the logger from the context
        var logger = (ILogger<PostPutActionFilter>)context.HttpContext.RequestServices.GetService(typeof(ILogger<PostPutActionFilter>))!;

        if (context.ActionArguments.Count == 0)
        {
            context.Result = new BadRequestObjectResult("No data found");
            logger.LogWarning("No data found");
        }
        else if (!context.ModelState.IsValid)
        {
            context.Result = new BadRequestObjectResult("Please fill all the requirement fields. And stop trying to circumvent server side validation.");
            logger.LogWarning("Server side validation circumvention trieal.");
        }
    }

    0 references
    public override void OnActionExecuted(ActionExecutedContext context)
    {

    }
}
```

```
using PresentationLayer.ViewModels
model MyProductVM

    ViewData["Title"] = "Edit Product";


<div class="container">
    <h2> ViewData["Title"]</h2>
    <form asp-action="EditProduct" method="post">
        <div class="form-group">
            <label asp-for="Name" class="control-label"></label>
            <input asp-for="Name" class="form-control" />
            <span asp-validation-for="Name" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Description" class="control-label"></label>
            <input asp-for="Description" class="form-control" />
            <span asp-validation-for="Description" class="text-danger"></span>
        </div>
        <div class="form-group">
            <label asp-for="Price" class="control-label"></label>
            <input asp-for="Price" class="form-control" />
            <span asp-validation-for="Price" class="text-danger"></span>
        </div>
        <div class="form-group">
            <input type="submit" value="Update" asp-route-ProductId=" Model.ProductId" class="btn btn-primary" />
        </div>
    </form>
    <div class="mt-3">
        <a asp-action="HomePage" class="btn btn-secondary">Back to List</a>
    </div>
</div>

section Scripts
    <partial name="_ValidationScriptsPartial" />
```

```
#nullable disable
17 references
public class MyProductVM
{
    4 references
    public int ProductId { get; set; }
    [Required]
    [MaxLength(30)]
    8 references
    public string Name { get; set; }
    [Required]
    [MaxLength(1000)]
    8 references
    public string Description { get; set; }
    [RegularExpression(@"^[0-9]+([.,][0-9]{1,2})?$", ErrorMessage = "Please specify the valid price")]
    9 references
    public string Price { get; set; }
}
```

```
public class MappingProfile : Profile
{

    0 references
    public MappingProfile()
    {
        CreateMap<MyProductVM, MyProduct>()
            .ForMember(dest => dest.Price, opt => opt.MapFrom(src => ConvertPrice(src.Price)));
        CreateMap<MyProduct, MyProductVM>();
        CreateMap<List<MyProduct>, HomePageVM>();
    }

    1 reference
    private decimal ConvertPrice(string price)
    {
        if (decimal.TryParse(price, out decimal result))
            return result;

        throw new ArgumentException("Invalid price format", nameof(price));
    }

}
```

```
public class GlobalExceptionMiddleware
{
    private readonly RequestDelegate _next;
    private readonly ILogger<GlobalExceptionMiddleware> _logger;

    0 references
    public GlobalExceptionMiddleware(RequestDelegate next, ILogger<GlobalExceptionMiddleware> logger)
    {
        _next = next;
        _logger = logger;
    }

    0 references
    public async Task InvokeAsync(HttpContext httpContext)
    {
        try
        {
            await _next(httpContext);
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, ex.Message);
            await HandleExceptionAsync(httpContext, ex);
        }
    }
    1 reference
    private static Task HandleExceptionAsync(HttpContext context, Exception exception)
    {
        var code = HttpStatusCode.InternalServerError;

        var result = JsonSerializer.Serialize(new { error = "Something went wrong, couldn't process the request" });
        context.Response.ContentType = "application/json";
        context.Response.StatusCode = (int)code;
        return context.Response.WriteAsync(result);
    }
}
```


THanks