Developing Great Web APIs Architectures with ASP.NET 8

Chris Woodruff

Chris Woodruff

Architect at Real Time Technologies Inc

cwoodruff@live.com

Twitter – @cwoodruff

Resources

GitHub Solution Demo

https://github.com/cwoodruff/Chinook7WebAPI

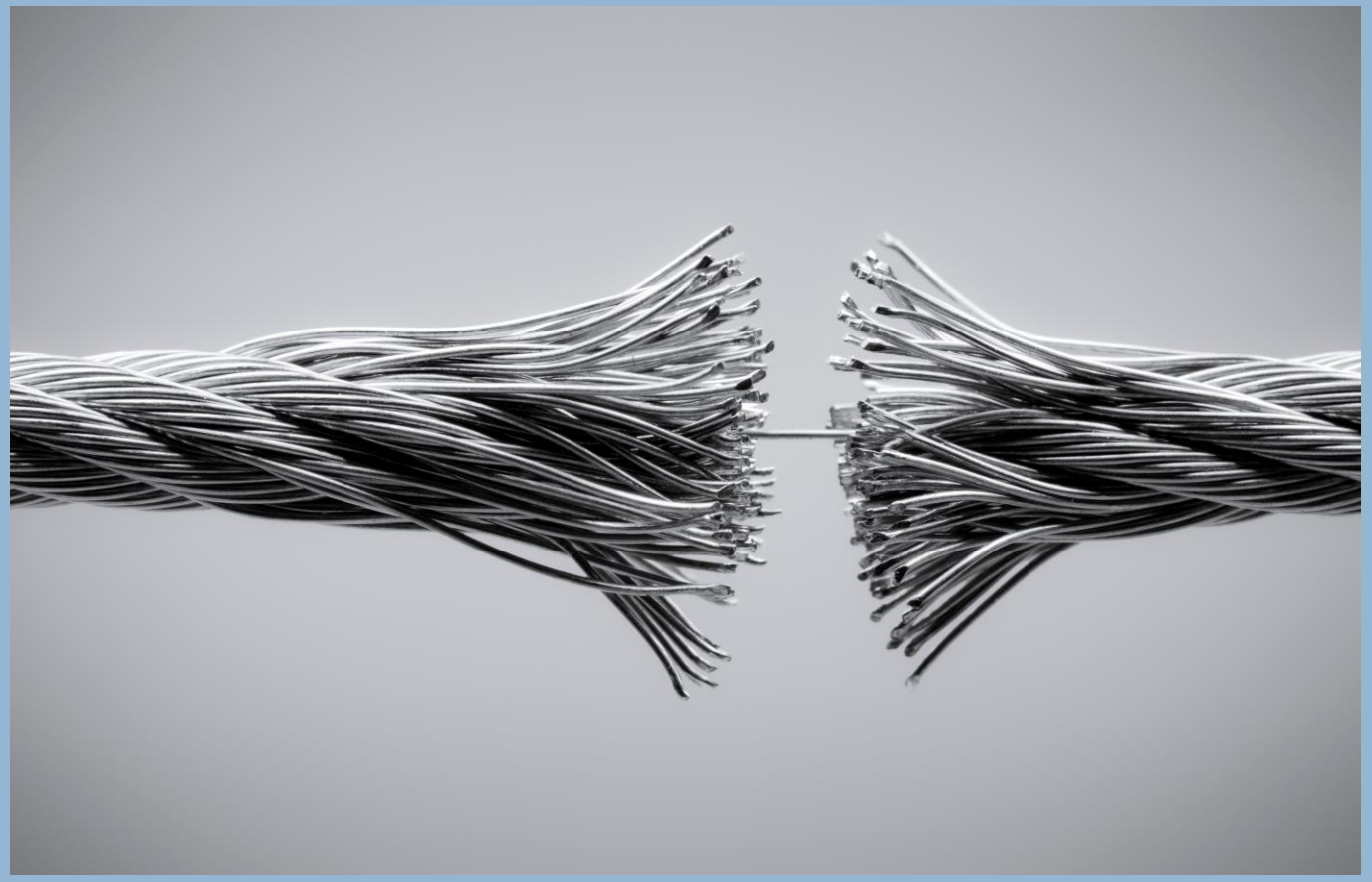
GitHub Workshop

https://github.com/cwoodruff/web-api-workshop

Online Docs Workshop

https://cwoodruff.github.io/web-api-workshop/

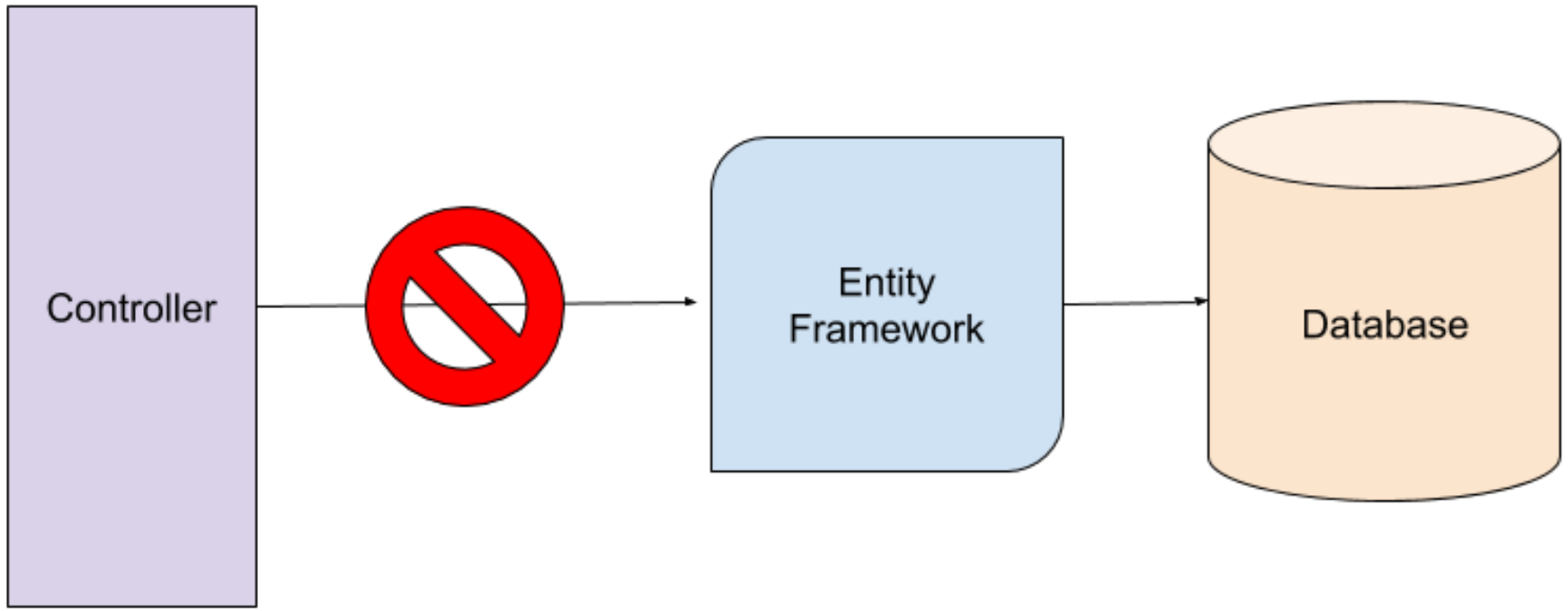
ASP.NET Core 8 Web API Architecture



What do you and I do wrong?

Bad Habits

Calling Data Access functionality (EF Core) from Controllers



Bad Habits

Having all code in a single project. Hard to test!

Bad Habits

Coupling your Data Access (EF Core) to your project Domain

Bad Habits

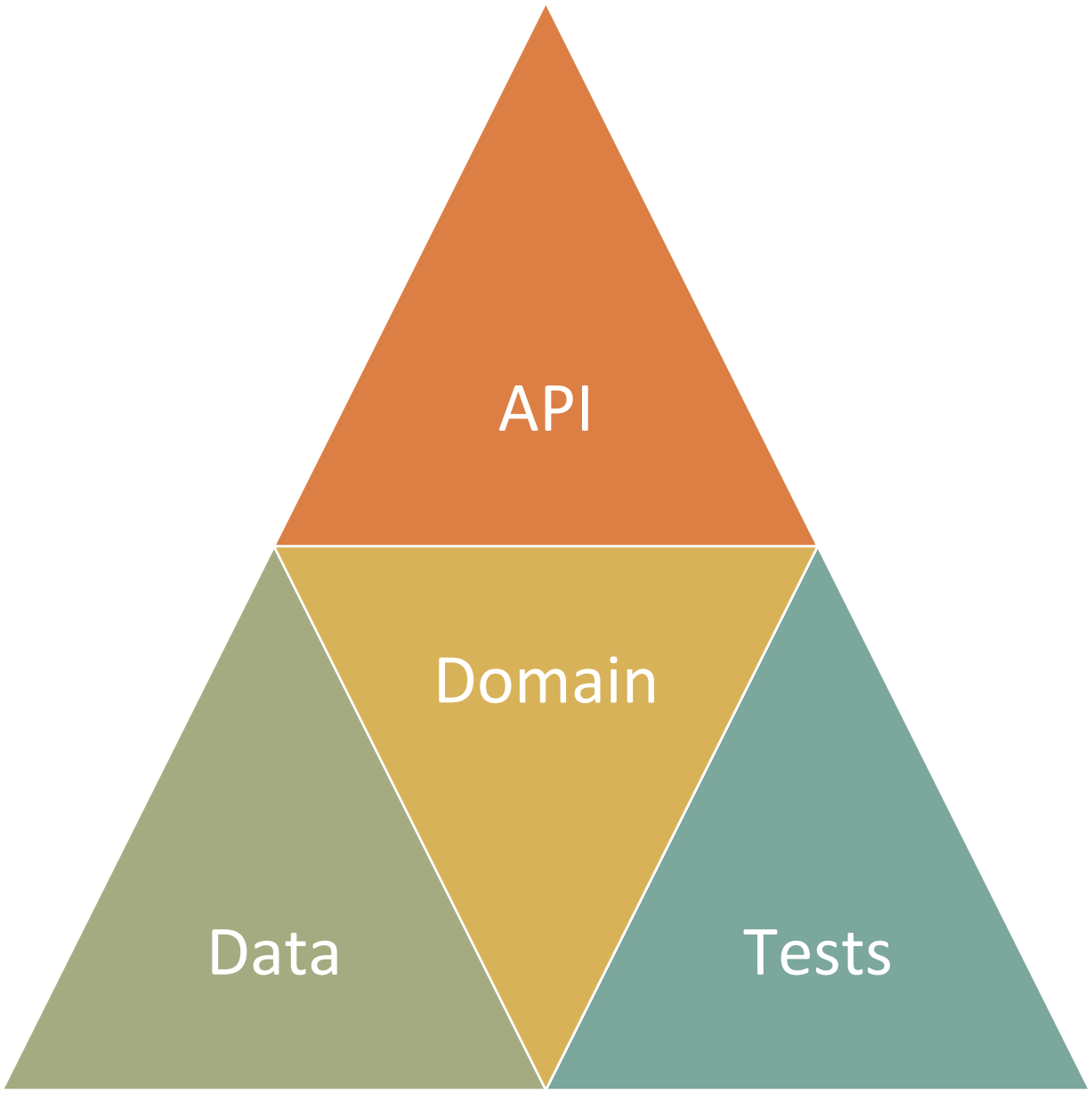
Not thinking about Unit or Integration testing

ASP.NET Core 8 Web API and Architecture

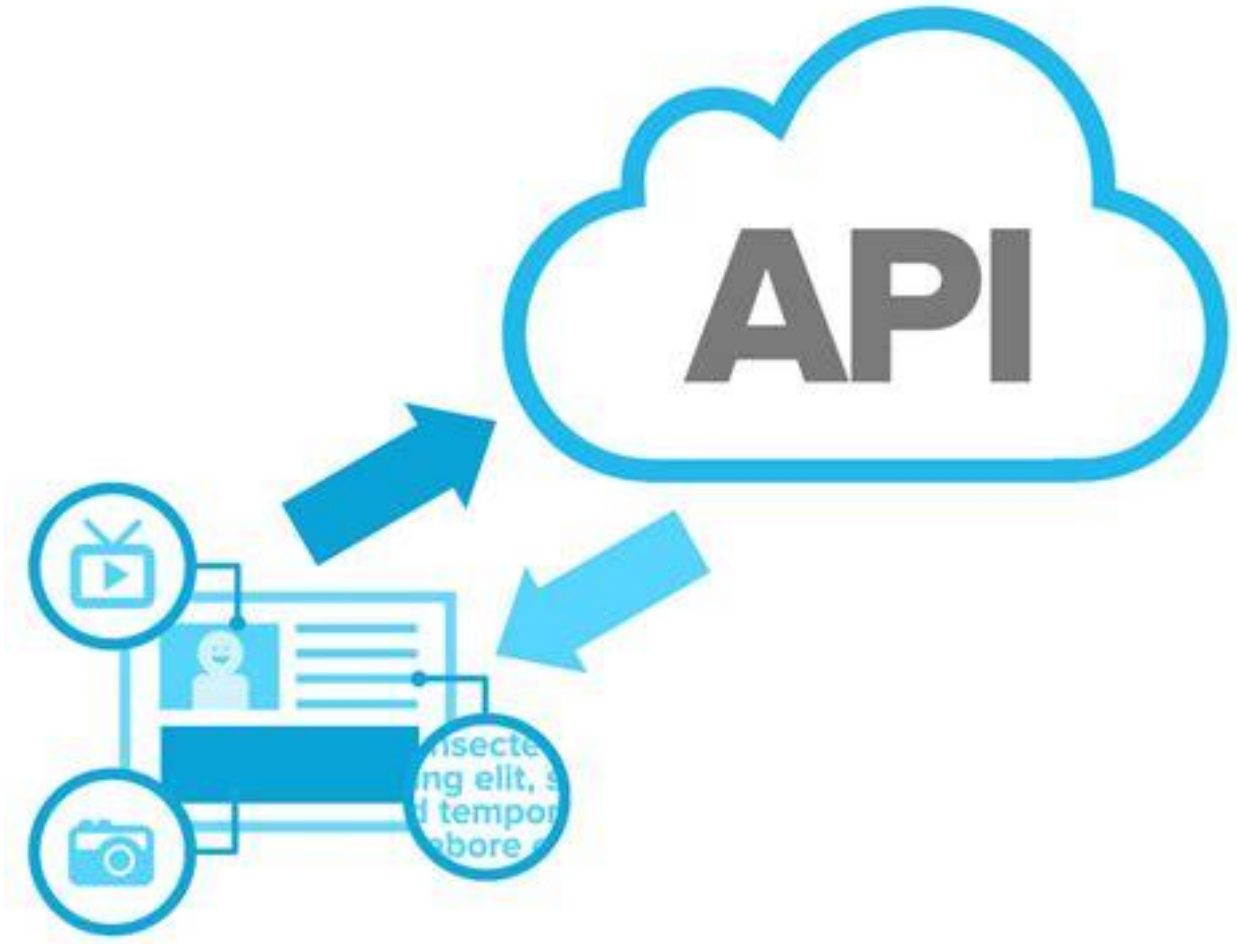
What does ASP.NET Core help us with?

Understand the HTTP Protocol

How I build my API’s

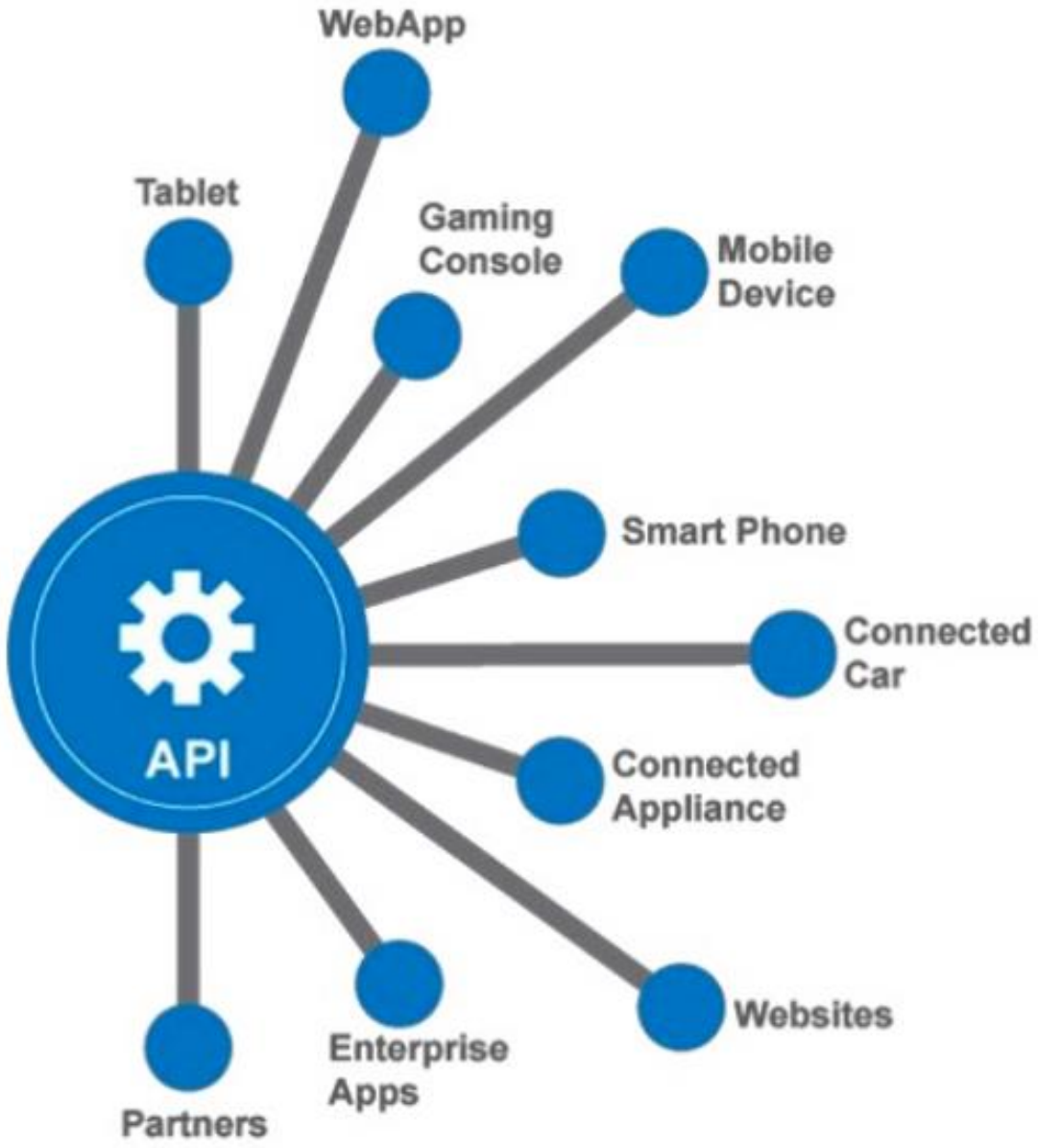


API Layer



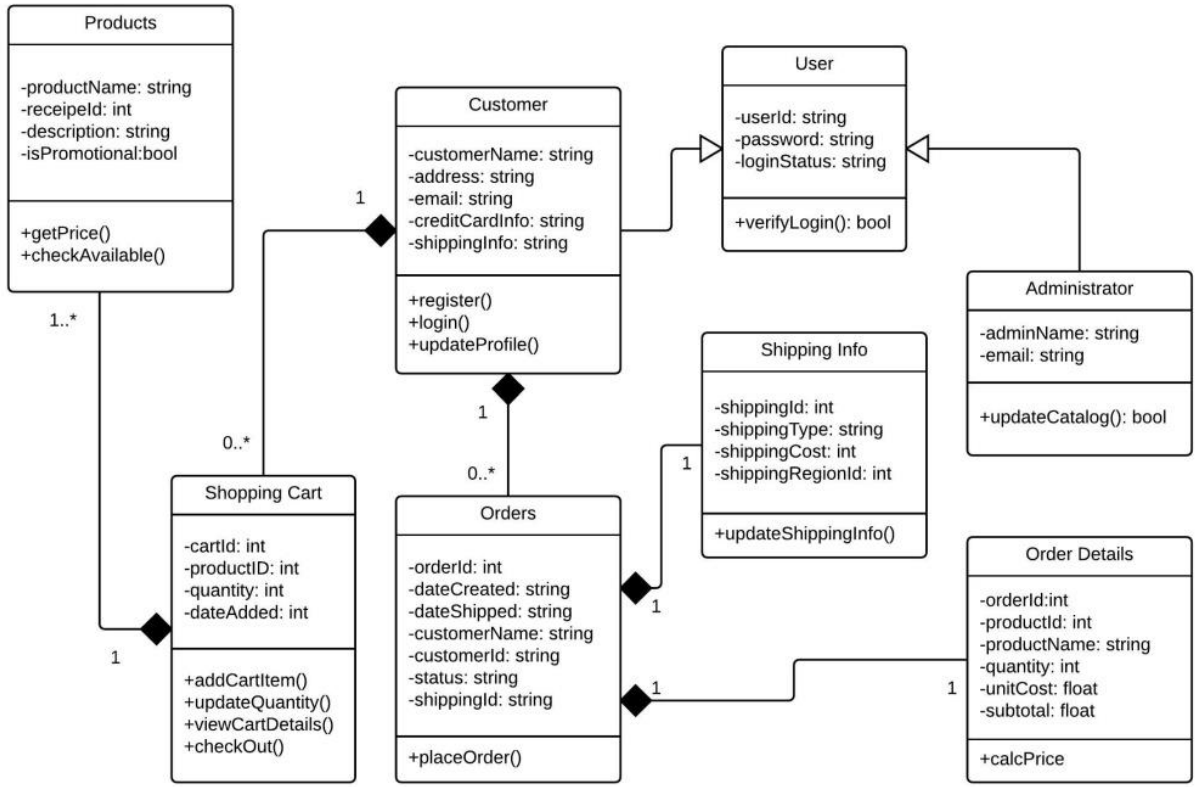
API Layer

Like the UI of an ASP.NET MVC (or any web platform and pattern), the API endpoints should not know about the Domain knowledge or Data Access



API Layer

Should interact with consumers with ApiModels to ensure the greatest flexibility

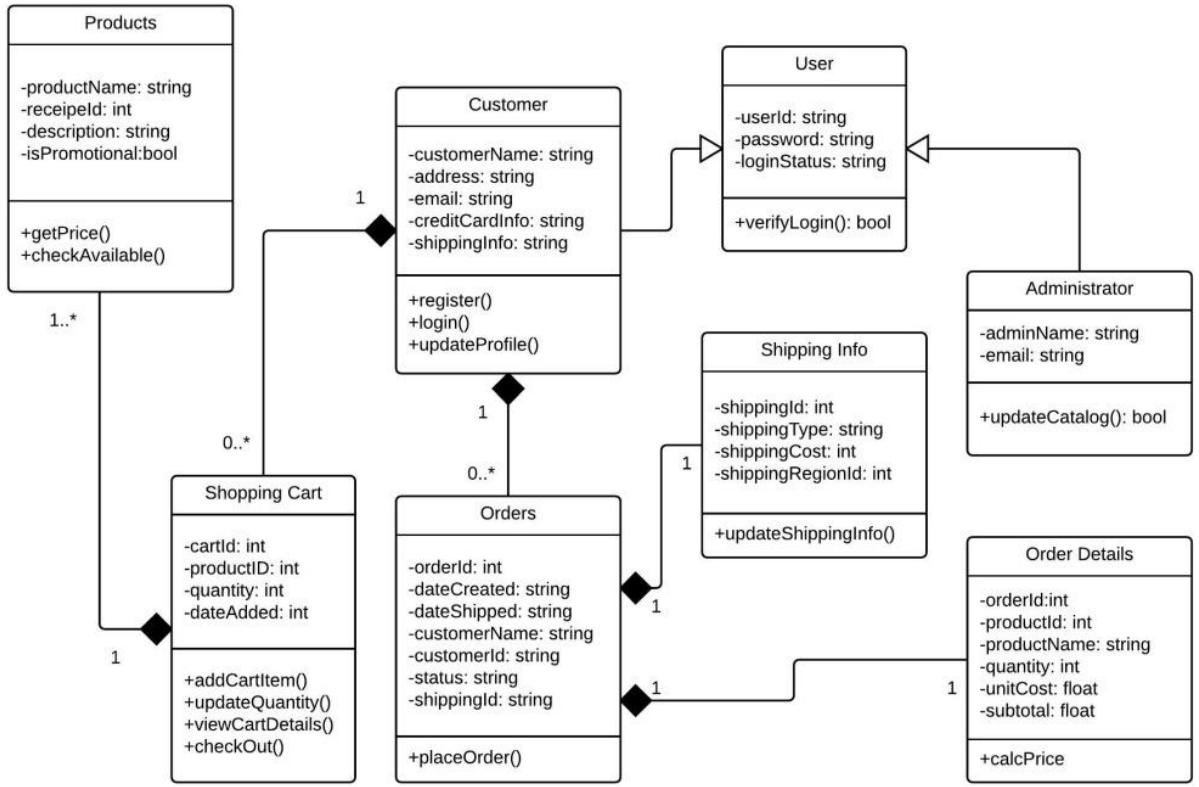


```
Public class Album
{
    public string? Title { get; set; }
    public int ArtistId { get; set; }

    public Artist? Artist { get; set; }
    public ICollection<Track>? Tracks { get; set; }

}


public class AlbumApiModel
{
    public string? Title { get; set; }
    public int ArtistId { get; set; }
    public string? ArtistName { get; set; }

    public ArtistApiModel? Artist { get; set; }

    public IList<TrackApiModel>? Tracks { get; set; }
}
```

Domain Layer

Domain Layer

Contains both my Entity models and my ApiModels for the solution.

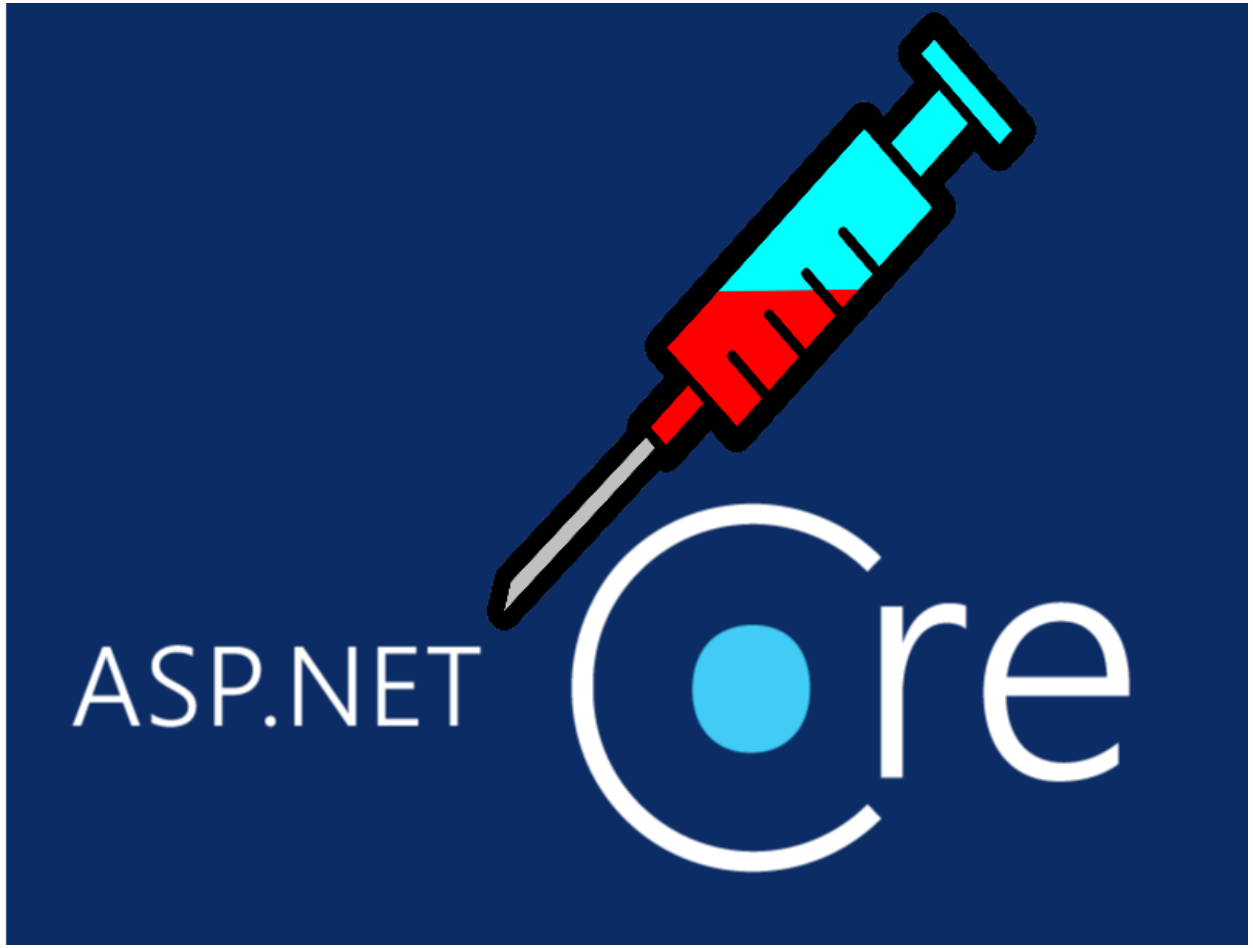


Domain Layer

Contains all my interfaces for Data Retrieval so I can keep a well-defined standard for my data access.

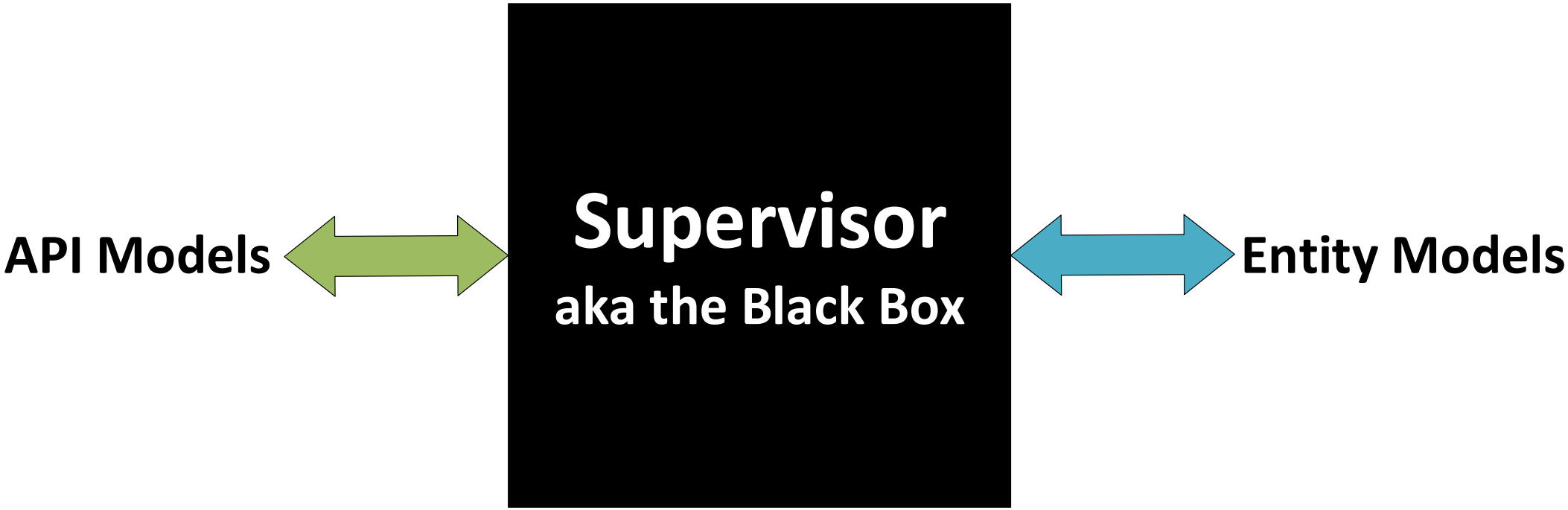
Domain Layer

Allows me to use Dependency Injection for Repositories



Domain Layer

Think of the Domain Layer as the Supervisor!



Data Layer

Data Layer

Where all the heavy data lifting happens

Data Layer

Everything defined based on Interfaces from the Domain Layer

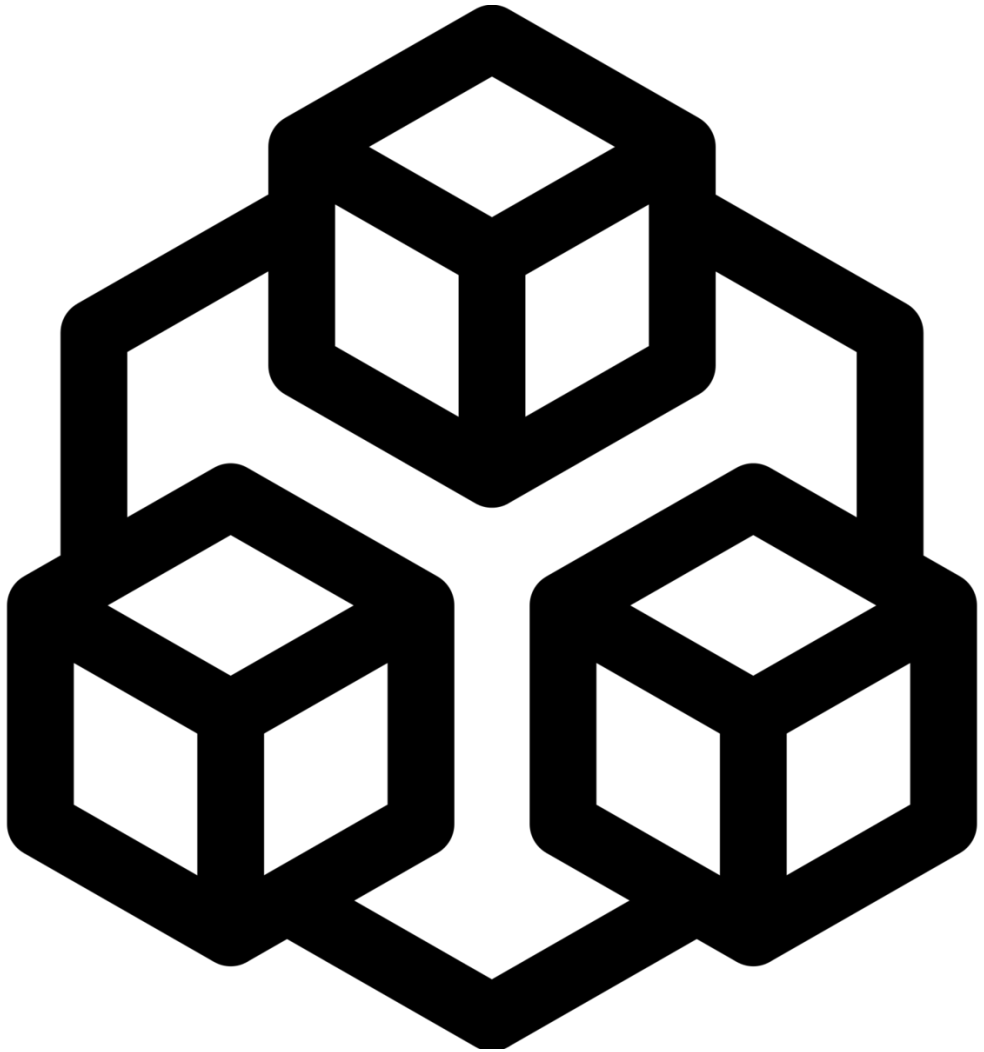
Data Layer

I use Entity Framework Core 8 but we can use anything like Dapper or SQL JSON

Data Layer

I also Mock up a data layer for testing

Caching



Caching for APIs

Response Caching

In-Memory Caching

Distributed Caching

Response Caching

Directs Caching on the consumer side

Attribute on Controller or Action

```
[ResponseCache(Duration = 60)]
```

Response Header:

```
Cache-Control: public,max-age=60
```

HTTP 1.1 Caching specification

```
[ResponseCache(Location =
ResponseCacheLocation.None, NoStore = true)]
```

Response Header:

```
Cache-Control: no-store,no-cache
Pragma: no-cache
```

In-Memory Caching

Directs Caching on the consumer side

Distributed Caching

Distributed caches can improve the performance and scalability of ASP.NET Core apps, especially when hosted in the cloud or a server farm.

1. Cached data is coherent on all web servers. Users don't see different results depending on which web server handles their request
2. Cached data survives web server restarts and deployments. Individual web servers can be removed or added without impacting the cache
3. The source data store has fewer requests made to it (than with multiple in-memory caches or no cache at all)

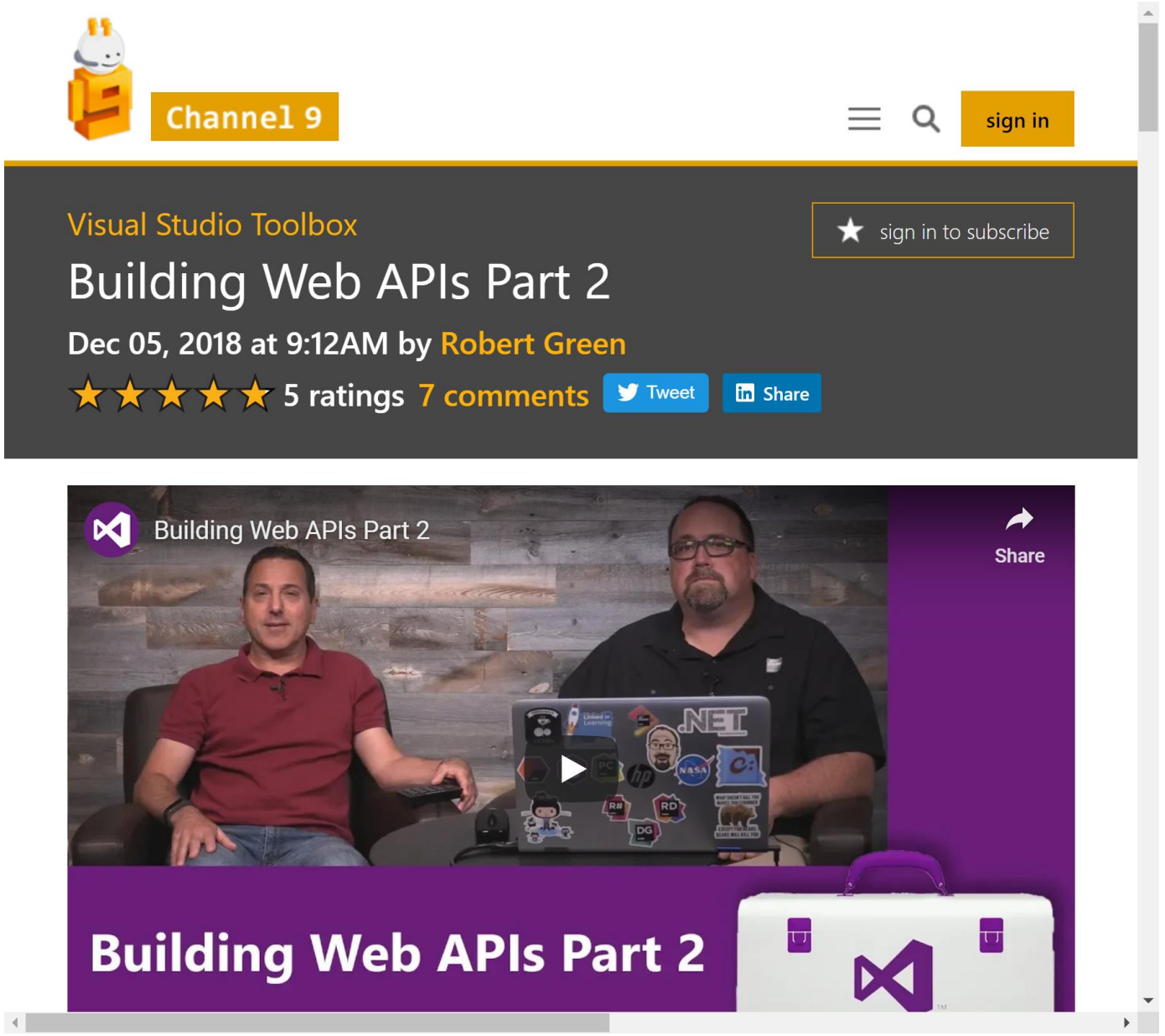
Distributed Caching

1. Local Redis Cache
2. SQL Server Cache
3. Azure Redis Cache
4. Amazon ElastiCache
5. Google Cloud Memcache

Testing

Tests

- I have 2 types of tests: Integration and Unit
- Integration for API endpoint testing and Controllers
- Unit Testing to cover the Repositories and Supervisor



https://channel9.msdn.com/Shows/Visual-Studio-Toolbox/Building-Web-APIs-Part-2

ASP.NET 5 Web API and the Hexagonal Architecture

Anything to ask?

Thanks!

Twitter	@cwoodruff
GitHub	https://github.com/cwoodruff
Email	cwoodruff@live.com
Sample project	https://github.com/cwoodruff/Chinook7WebAPI
Workshop	https://cwoodruff.github.io/web-api-workshop/