.NET Core 2.2 Superpowers Tour

Brisbane, Melbourne & Sydney - November 2018

Join the Conversation #DotNetCoreSuperpowers @SSW_TV

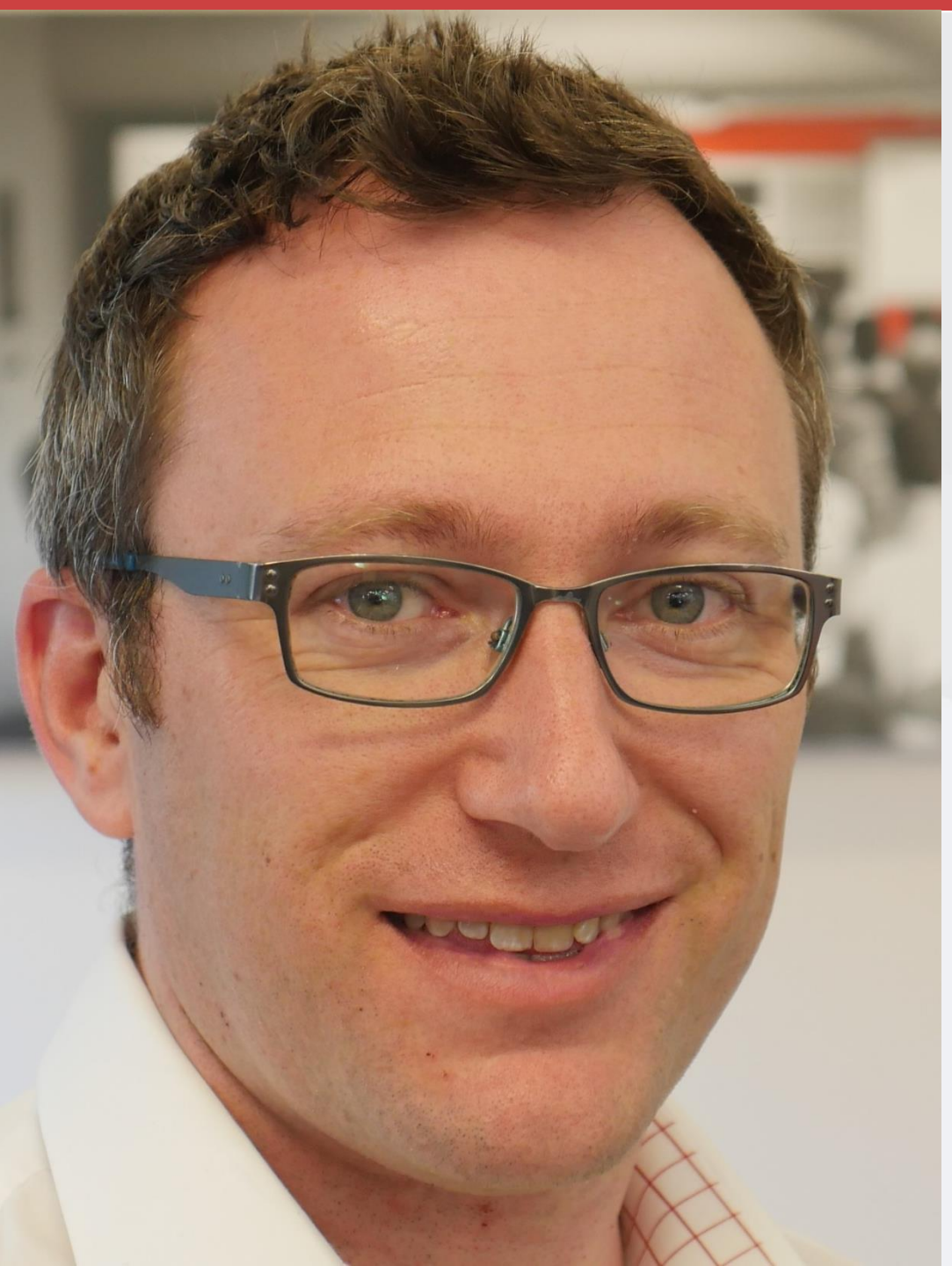
House Keeping

Code & Slides

bit.ly/netcore-superpowers

Timetable

09:00	Start
10:30 – 10:45	Morning Tea
13:00 – 13:45	Lunch
15:30 – 15:45	Afternoon Tea
16:45 – 17:00	Feedback, Q & A
17:00	Finish



Brendan Richards

@brendanssw @ssw_tv

- Solution Architect @ SSW
- Linux User since 1995
- .NET Developer since 2004
- .NET Core Since Beta 2

Jason Taylor

SSW Solution Architect

Started programming with BASIC on C64,

Keeping it simple since 1994!

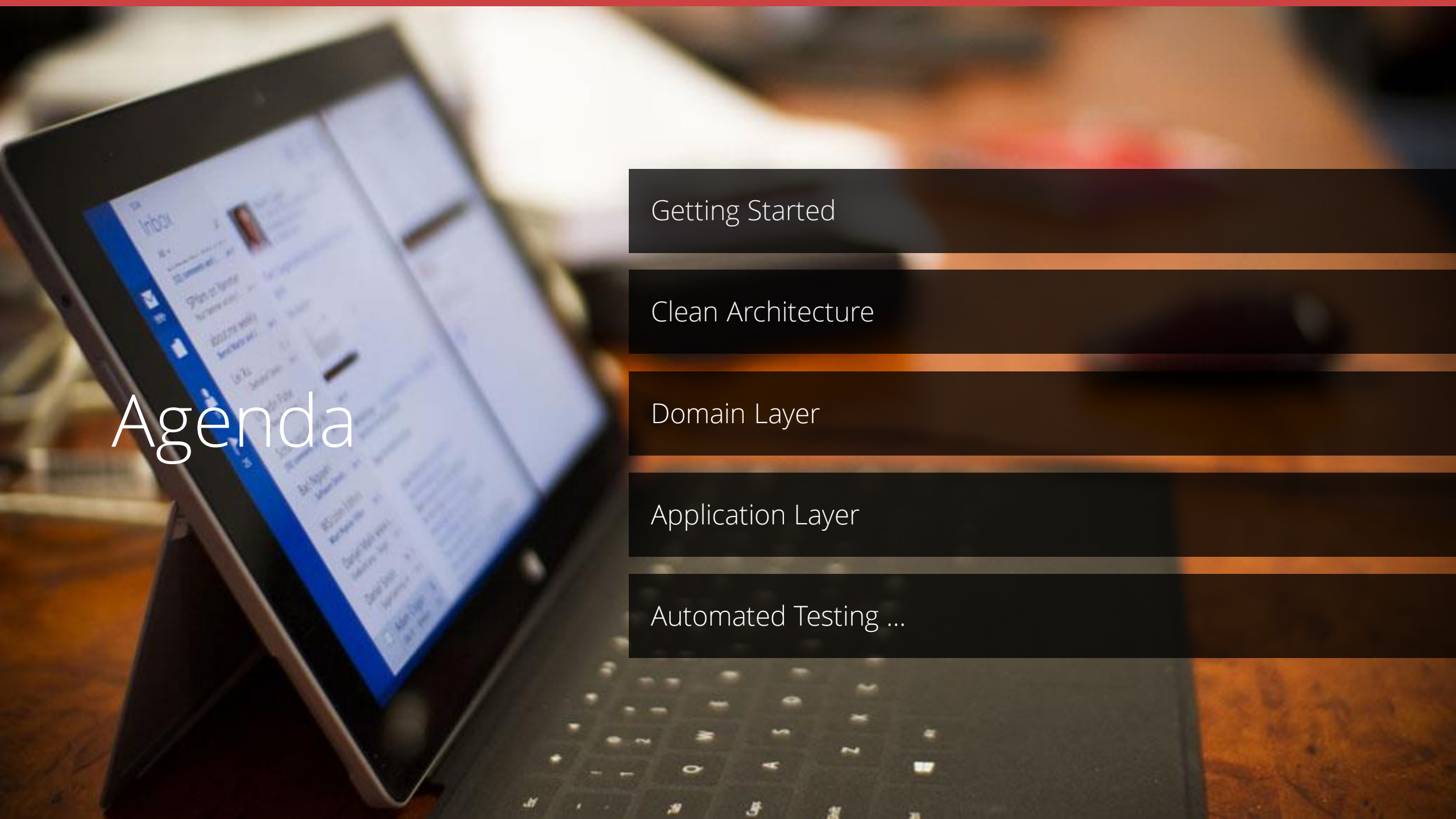
jasongtau

codingflow.net

github.com/jasongt

youtube.com/jasongt

Join the Conversation #DotNetCoreSuperpowers @SSW_TV



Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

History

2002 - .NET Framework 1.0

The “Stuff you, Sun Microsystems” Edition

2004 - .NET Framework 1.1

Shipped with windows

2006 - .NET Framework 2.0

Generics and other new features

History

2006 - .NET Framework 3.0

Adds to the 2.0 runtime – WPF, WCF

2006 - .NET Framework 3.5

Still on 2.0 framework, Adds LINQ

2006 - .NET Framework 4.0

New Runtime. Async/Await

History

2012 - .NET Framework 4.5

UWP, The UI style formerly known as Metro

2013 - .NET Framework 4.5.1

2014 - .NET Framework 4.5.2

2015 - .NET Framework 4.6

Why .NET Core?

Simple answer: Azure

Cross platform

Open source (MIT License)

Optimised for server / cloud / containers

Faster

Release history [edit]

Version Number	Release Date	Support Ended	Development Tool
1.0	2016-06-27	2019-06-27	Visual Studio 2015, 2017
1.1	2016-11-18	2019-06-27	Visual Studio 2015, 2017
2.0	2017-08-14	2018-10-01	Visual Studio 2017
2.1	2018-05-30	2021-08-21[3]	Visual Studio 2017
2.2	in development		Visual Studio 2017,2019
3.0	in development		Visual Studio 2017,2019

Introducing .NET Standard

Formal specification of .NET APIs

Establishes greater uniformity in the .NET ecosystem

Implemented by .NET Core, .NET Framework, Mono, ...

bit.ly/2ys2Jvx

The following table lists the minimum platform versions that support each .NET Standard version.

.NET Standard	1.0	1.1	1.2	1.3	1.4	1.5	1.6	2.0
.NET Core	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
.NET Framework [1]	4.5	4.5	4.5.1	4.6	4.6.1	4.6.1	4.6.1	4.6.1
Mono	4.6	4.6	4.6	4.6	4.6	4.6	4.6	5.4
Xamarin.iOS	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.14
Xamarin.Mac	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.8
Xamarin.Android	7.0	7.0	7.0	7.0	7.0	7.0	7.0	8.0
Universal Windows Platform	10.0	10.0	10.0	10.0	10.0	10.0.16299	10.0.16299	10.0.16299

nuget Packages Upload Statistics Documentation Downloads Blog Sign

Search for packages...

Newtonsoft.Json 12.0.1-beta1

Json.NET

Json.NET is a popular high-performance JSON framework for .NET

ⓘ This is a prerelease version of Newtonsoft.Json.

Requires NuGet 2.12 or higher.

Package Manager | .NET CLI | Paket CLI

```
PM> Install-Package Newtonsoft.Json -Version 12.0.1-beta1
```

Info

last updated 22 days ago

Project Site

Source Code

License Info

Contact owners

Report

Download Package (3.24 MB)

Statistics

163,264,397 total downloads

20,988 downloads of latest

> Dependencies

Dependencies

.NETFramework 2.0

No dependencies.

.NETFramework 3.5

No dependencies.

.NETFramework 4.0

No dependencies.

.NETFramework 4.5

No dependencies.

.NETStandard 1.0

Microsoft.CSharp (>= 4.3.0)
NETStandard.Library (>= 1.6.1)
System.ComponentModel.TypeConverter (>= 4.3.0)
System.Runtime.Serialization.Primitives (>= 4.3.0)

.NETStandard 1.3

Microsoft.CSharp (>= 4.3.0)
NETStandard.Library (>= 1.6.1)
System.ComponentModel.TypeConverter (>= 4.3.0)
System.Runtime.Serialization.Formatters (>= 4.3.0)
System.Runtime.Serialization.Primitives (>= 4.3.0)
System.Xml.XmlDocument (>= 4.3.0)

.NETStandard 2.0

No dependencies.

Owners

jamesnk

newtonsoft

Authors

James Newton-King

Copyright

Tags

json

Version Switching

dotnet –-list-sdks

Use global.json to switch .NET Core SDK versions.

```
{
    "sdk": { "version": "1.0.0-preview2-003133" }
}
```

When not to use .NET Core

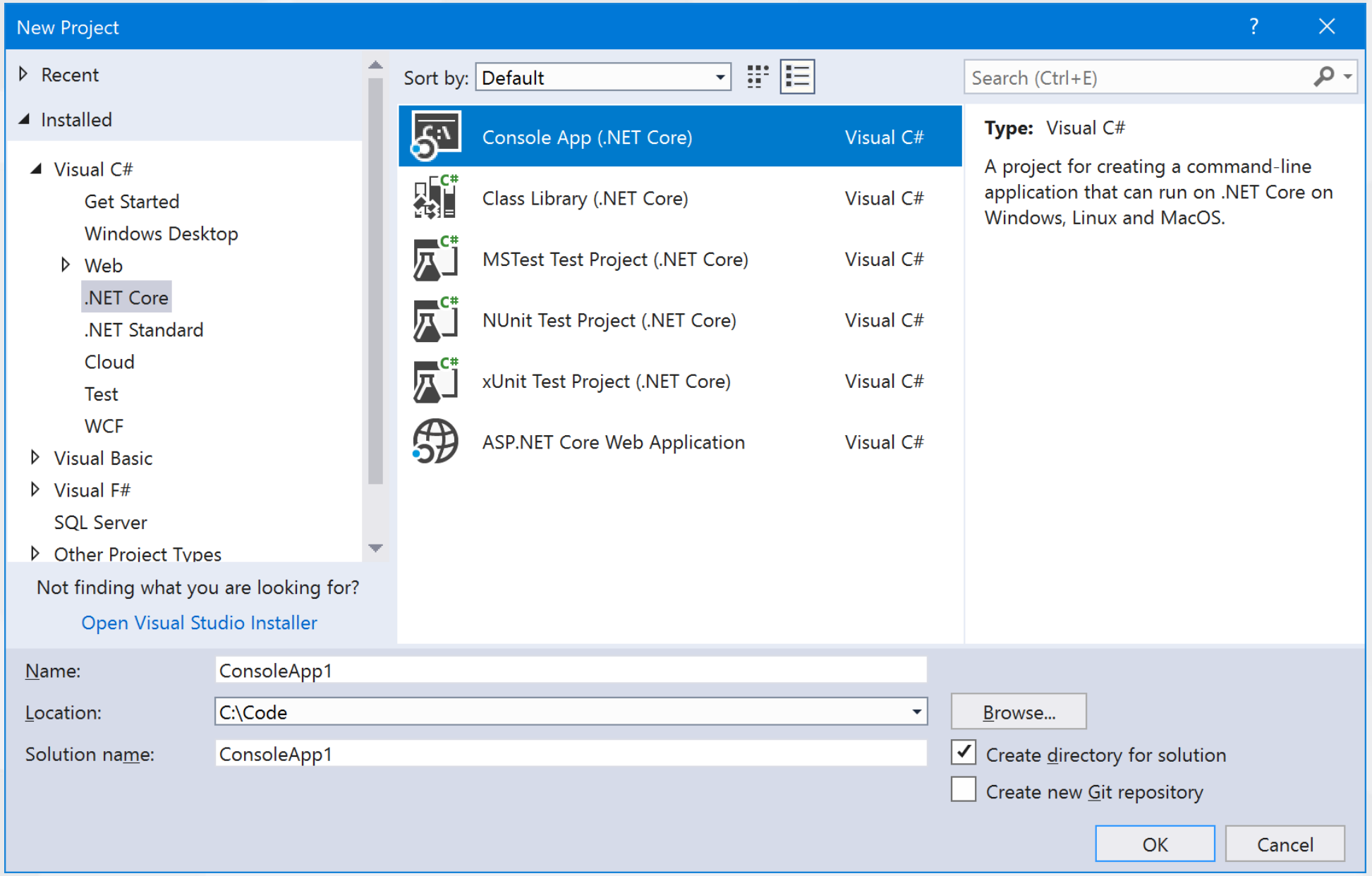
Windows App (WPF etc)

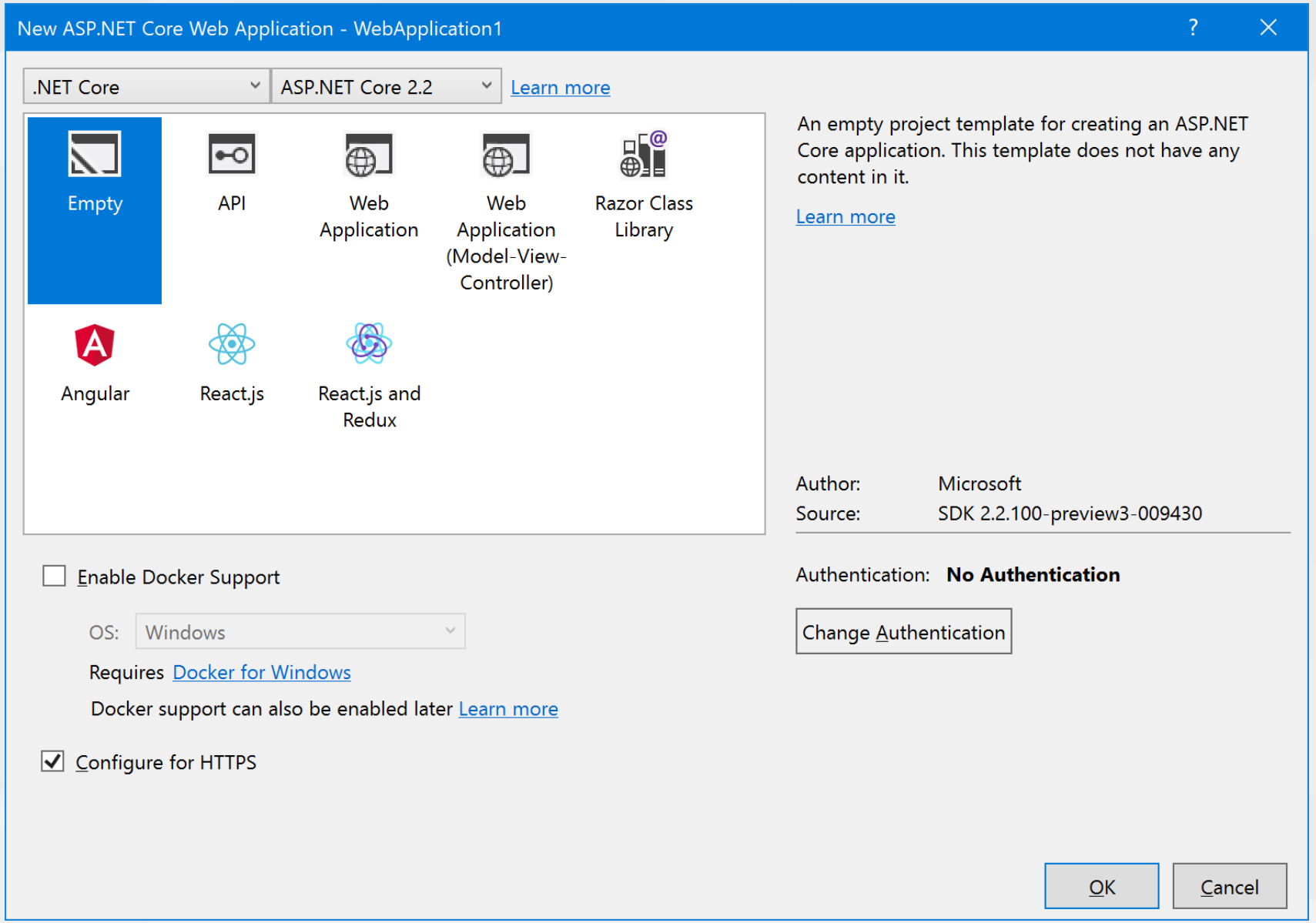
No Web Forms (probably a good thing)

Windows – specific APIs

Windows compatibility pack

https://blogs.msdn.microsoft.com/dotnet/2017/11/16/announcing-the-windows-compatibility-pack-for-net-core/


New Project
Recent
Installed
Visual C#
Get Started
Windows Desktop
Web
.NET Core
.NET Standard
Cloud
Test
WCF
Visual Basic
Visual F#
SQL Server
Other Project Types
Not finding what you are looking for?
Open Visual Studio Installer
Sort by: Default
Search (Ctrl+E)
Console App (.NET Core) Visual C#
Class Library (.NET Core) Visual C#
MSTest Test Project (.NET Core) Visual C#
NUnit Test Project (.NET Core) Visual C#
xUnit Test Project (.NET Core) Visual C#
ASP.NET Core Web Application Visual C#
Type: Visual C#
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Name: ConsoleApp1
Location: C:\Code
Browse...
Solution name: ConsoleApp1
Create directory for solution
Create new Git repository
OK
Cancel


New ASP.NET Core Web Application - WebApplication1
.NET Core
ASP.NET Core 2.2
Learn more
Empty
API
Web Application
Web Application (Model-View-Controller)
Razor Class Library
Angular
React.js
React.js and Redux
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
Learn more
Author: Microsoft
Source: SDK 2.2.100-preview3-009430
Enable Docker Support
OS: Windows
Requires Docker for Windows
Docker support can also be enabled later Learn more
Authentication: No Authentication
Change Authentication
Configure for HTTPS
OK
Cancel

Demo: Getting Started

Creating an app using .NET Core in Visual Studio and using the .NET Core CLI

Demo: ASP .NET Core Bootstrap

.csproj

Program.cs

Configuration

Startup.cs

Installing and configuring Nuget Packages

Demo: Customising configuration

Add a local config file.

Strong-typed configuration objects

Built-in Dependency Injection

ASP.NET Core supports and leverages dependency injection

The default implementation provides a minimal feature set and services basic needs

Developers can integrate the built-in container with their preferred container

Libraries will often register themselves

Service Lifetimes

Transient – Created each time they're requested

Scoped – Created once per request (Http)

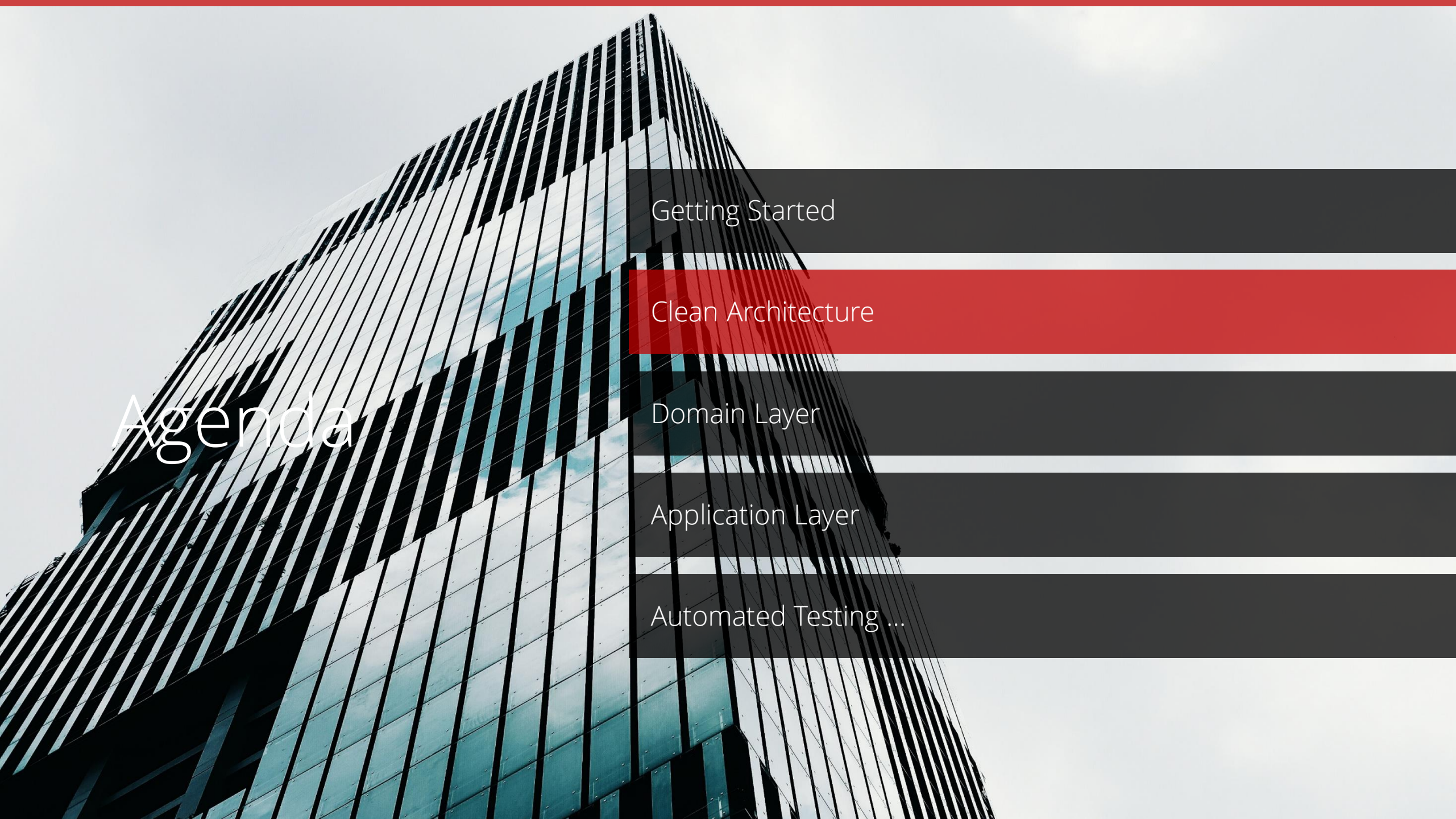
Singleton – Created once when they're requested, subsequent requests use the same instance

Demo: Adding Autofac

Add Autofac to your project – integrated with Standard DI

Key Points

- .NET Core is the future of .NET
- Open Source and Cross-Platform
- ASP.NET Core is the Web implementation on top of .NET
- Not dependent on IIS
- .NET Standard ensures compatibility
- Reboot of the framework – more consistent core features



Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

Overview

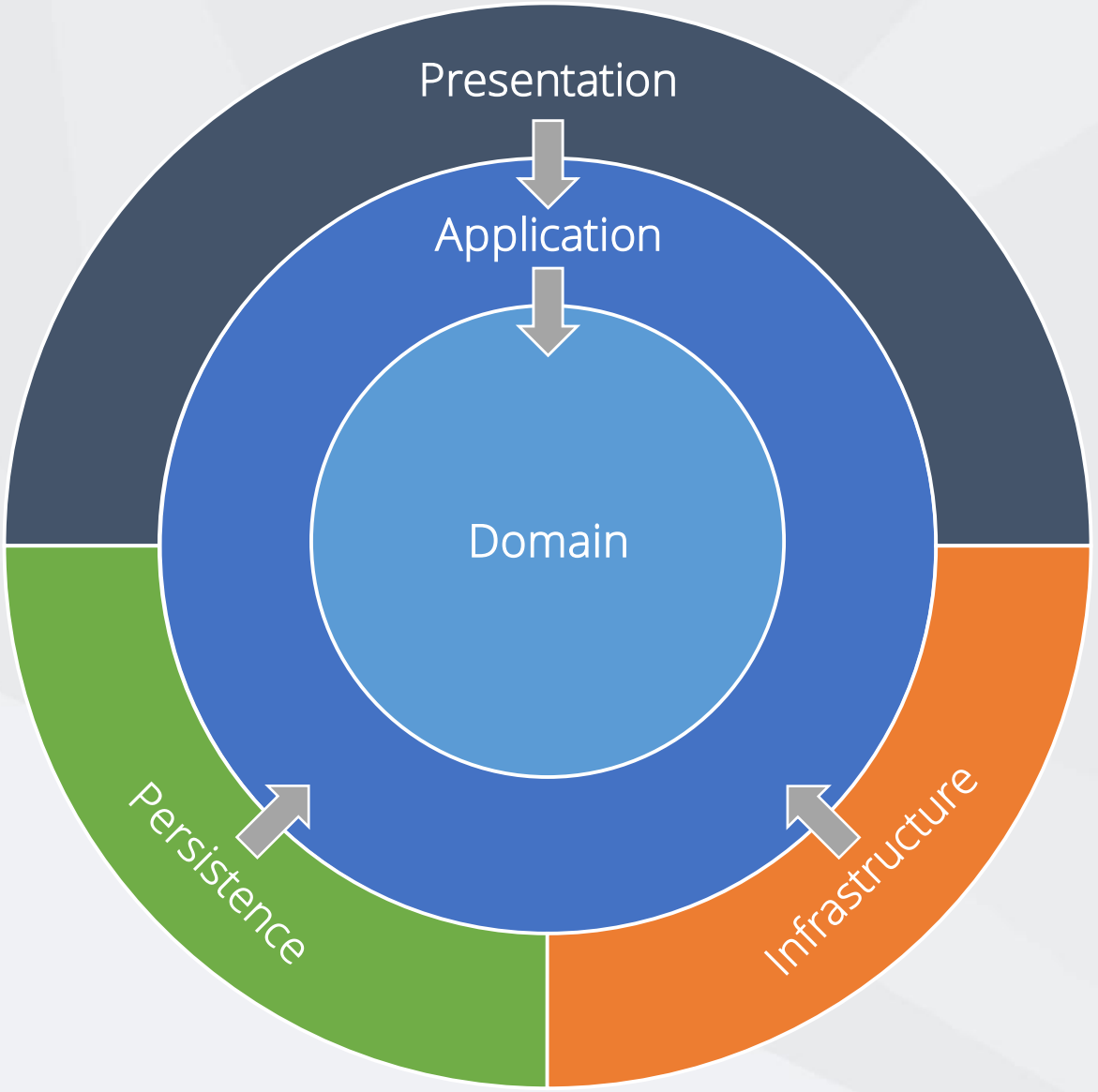
Independent of frameworks

Testable

Independent of UI

Independent of database

Independent anything external



Northwind Traders

Cross Platform

.NET Core

Entity Framework Core

Code First

Data Seeding

Prerequisites

.NET Core 2.2 Preview 3	bit.ly/netcoresdk-2-2
Node.js (8.11 or higher)	nodejs.org
npm (5.6 or higher)	(Included with Node.js)
Visual Studio Code / 2017 (Preview)	visualstudio.com/downloads

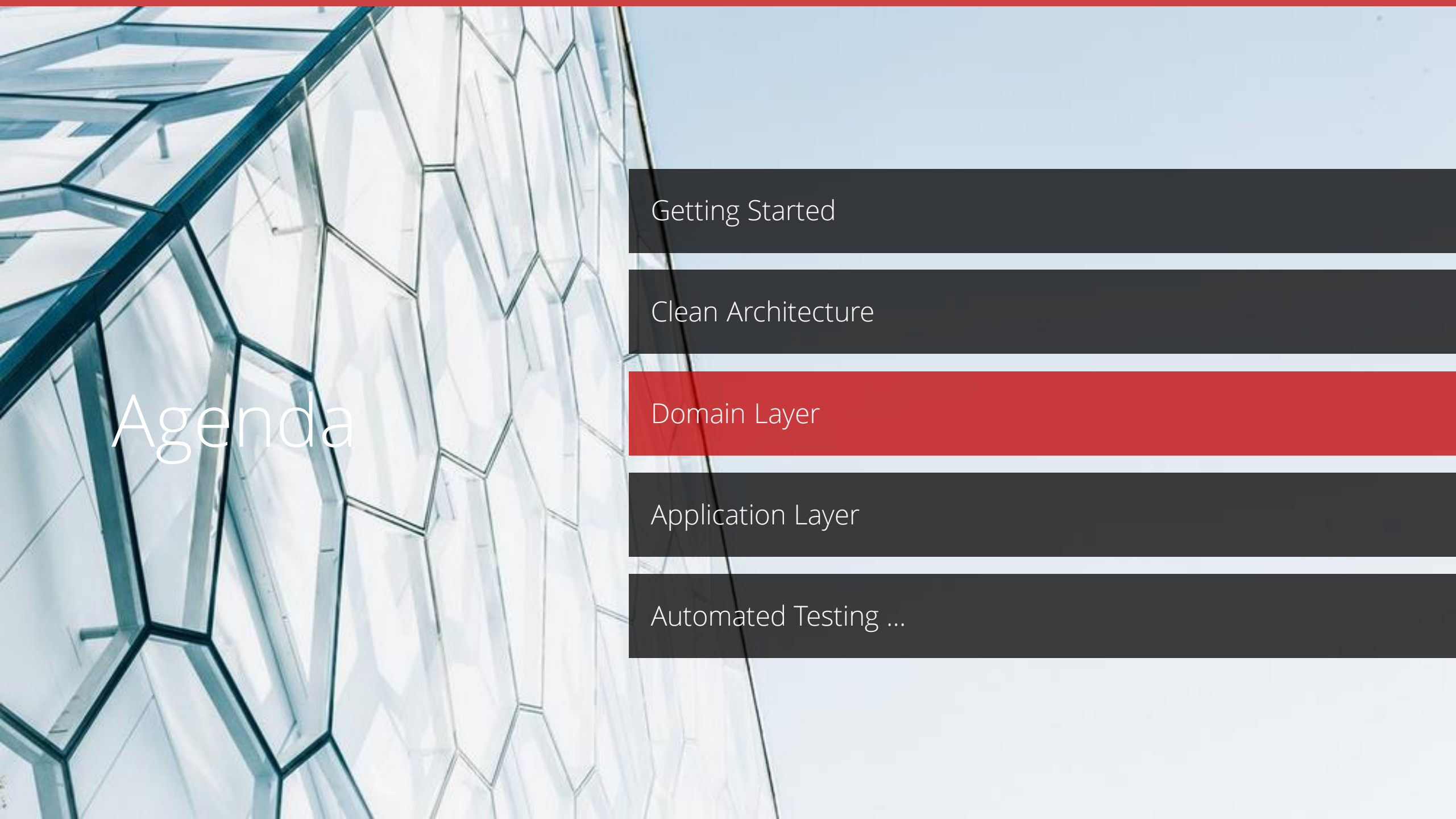
Visual Studio Code

Visual Studio

Visual Studio for Mac

Key Points

- Domain contains enterprise-wide types and logic
- Application contains application-specific types and logic
- Infrastructure (including Persistence) contain all external concerns
- Presentation and Infrastructure depend only on Application
- Infrastructure and Presentation components can be replaced with minimal effort



Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

Overview

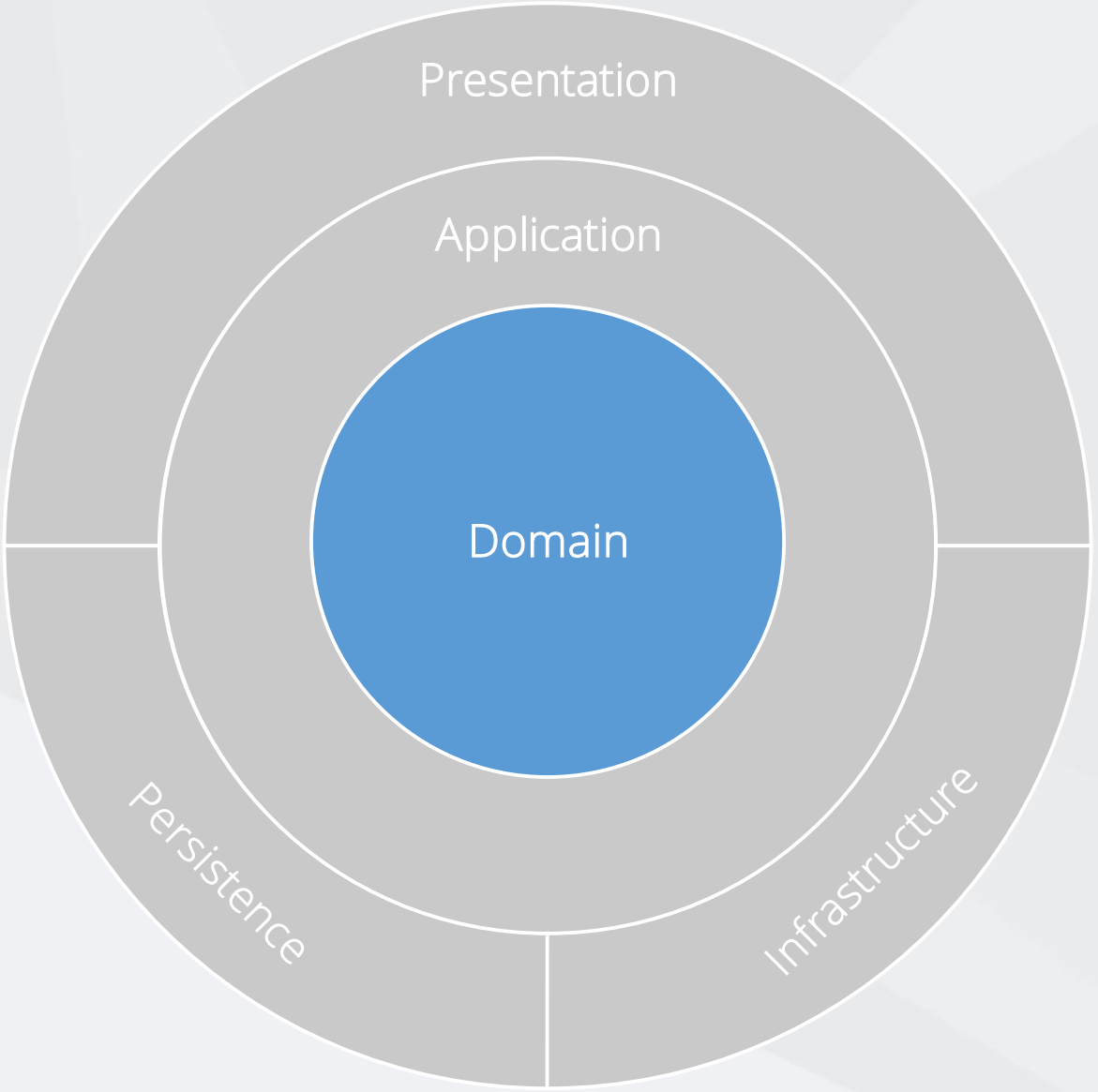
Entities

Value Objects

Enumerations

Logic

Exceptions

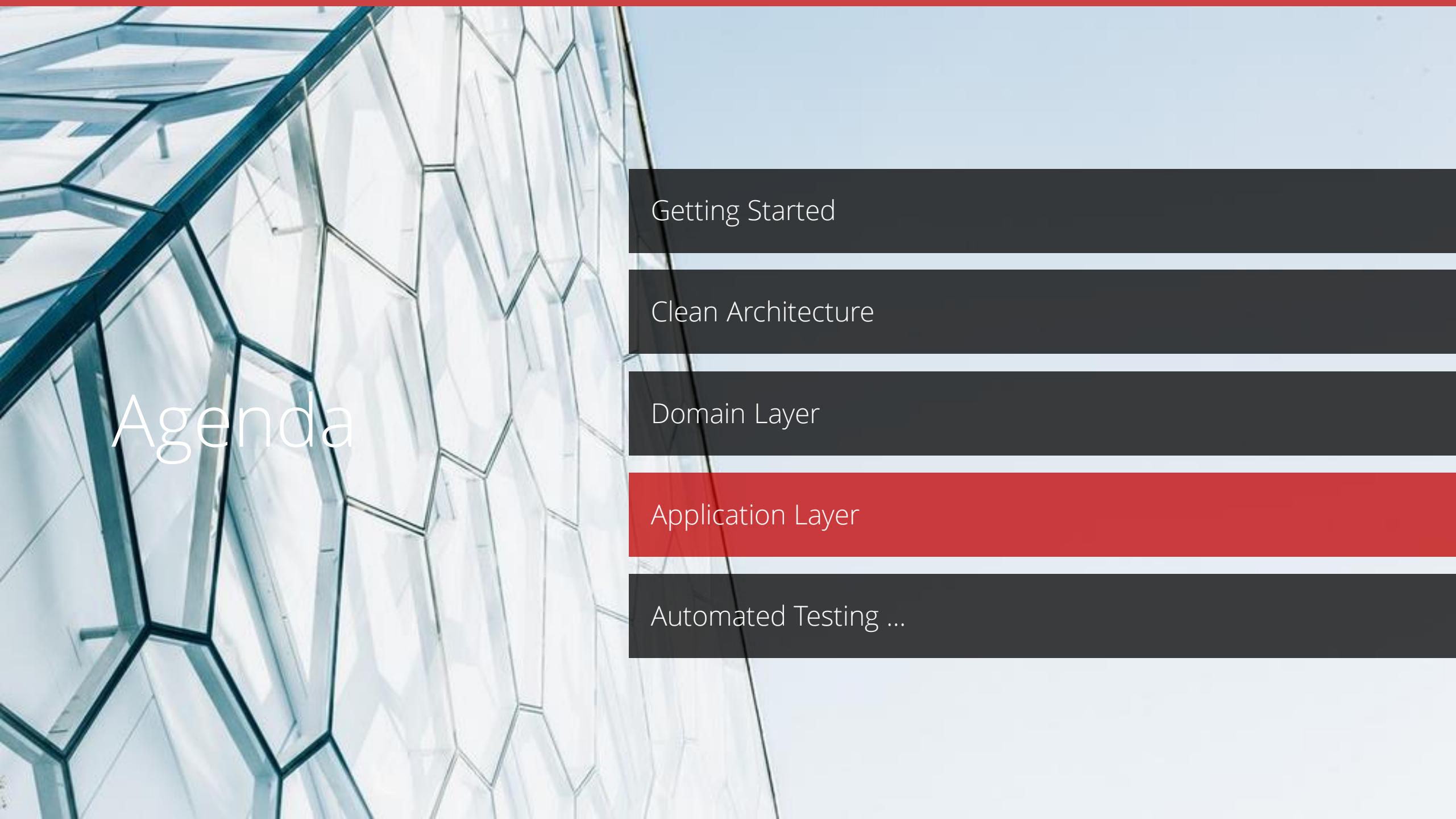


Demo

Reviewing the Domain layer

Key Points

- Use data annotations sparingly
- Use value objects when appropriate
- Initialise all collections & use private setters
- Create custom domain exceptions



Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

Overview

Interfaces

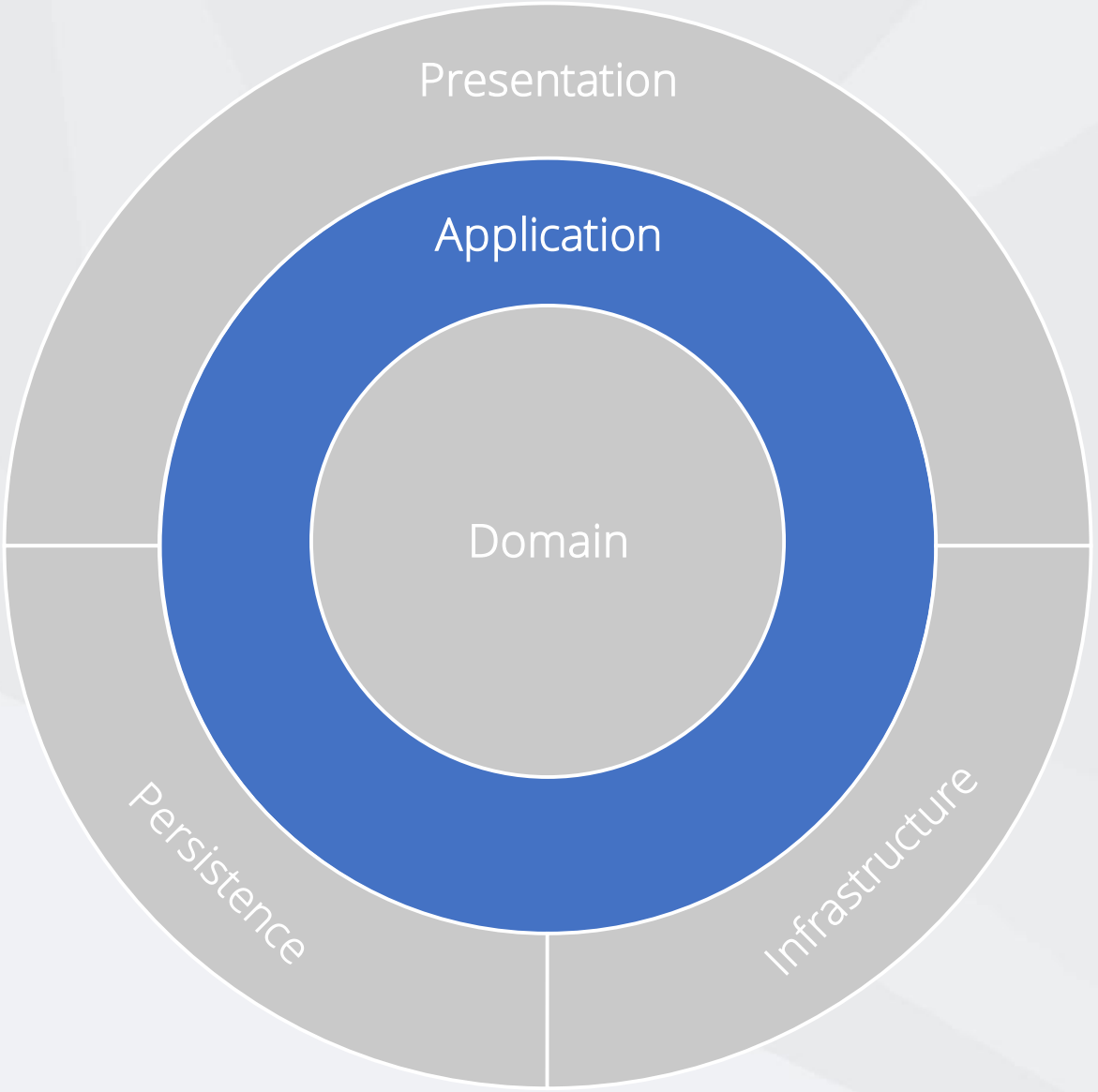
Models

Logic

Commands / Queries

Validators

Exceptions



CQRS

Command Query Responsibility Segregation

Separate reads (queries) from writes (commands)

Can maximise performance, scalability, and simplicity

Can simplify your over all design

Easy to add new features, just add a new query or command

Easy to maintain, changes only affect one command or query

MediatR + CQRS = ❤

Define commands and queries as requests

Application layer is just a series of request / response objects

Ability to attach additional behaviour before and / or after each request, e.g. logging, validation, caching, authorisation and so on

Demo

Reviewing the Application layer

Client-Side Validation

Demo: Using Fluent Validation

Installing and configuring Fluent Validation

Demo: Cross-Cutting Concerns

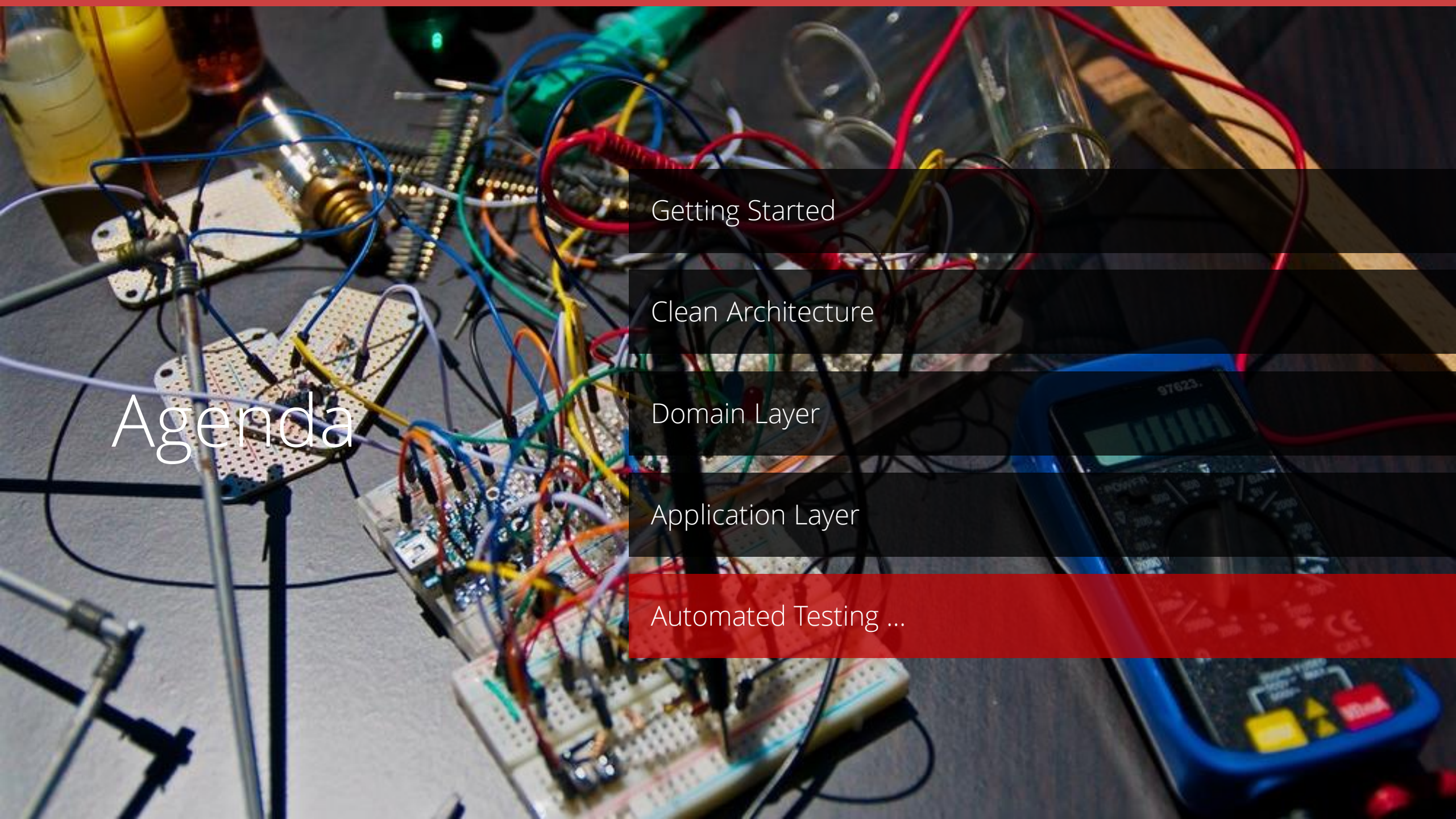
Validating Requests

Logging Requests

Monitoring Performance

Key Points

- Using CQRS + MediatR simplifies your overall design
- Fluent Validation is useful for simple and complex validation scenarios
- MediatR simplifies cross cutting concerns such as logging and validation
- Independent of infrastructure and data access concerns

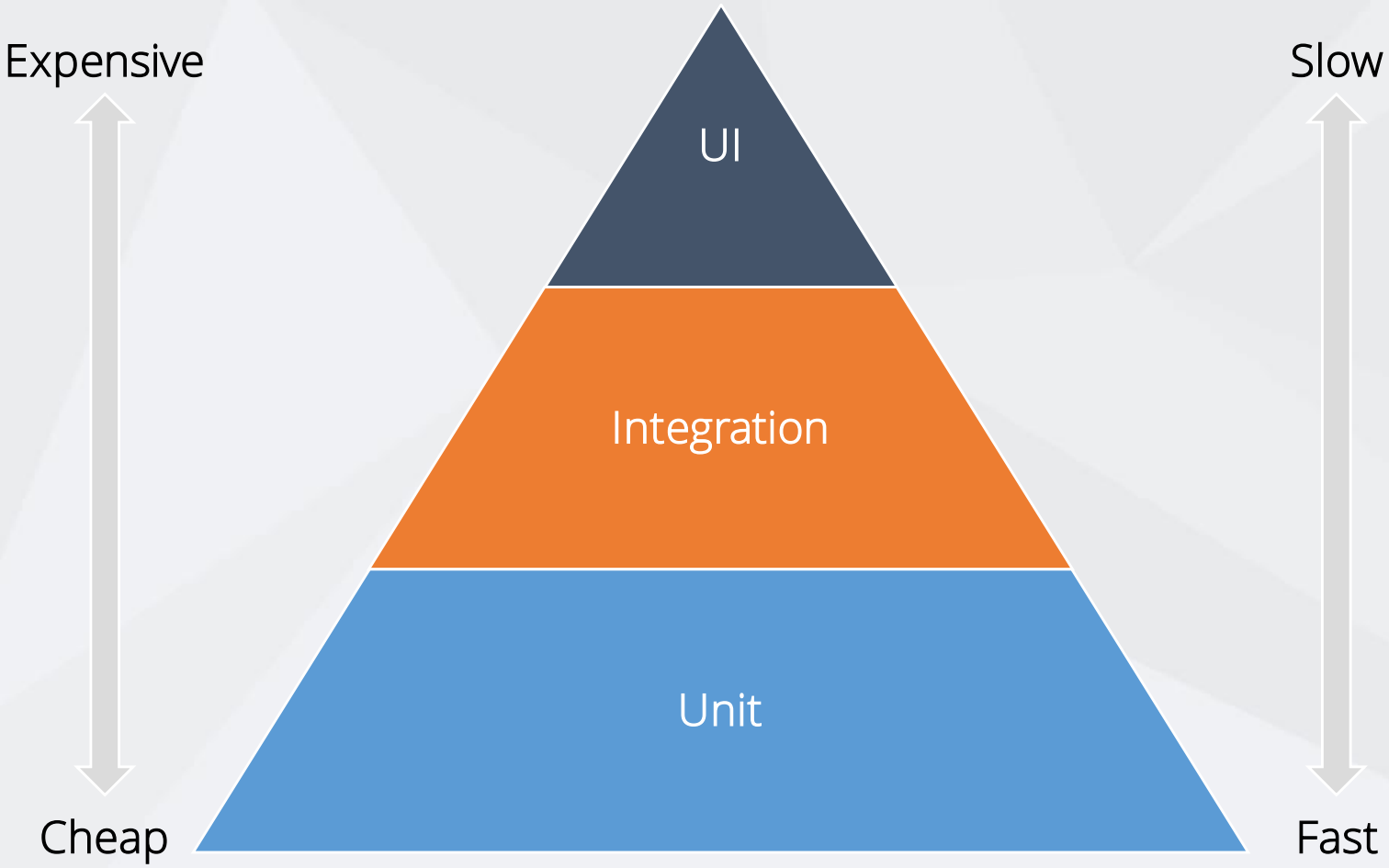


Agenda

- Getting Started
- Clean Architecture
- Domain Layer
- Application Layer
- Automated Testing ...

The First User

Testing Pyramid



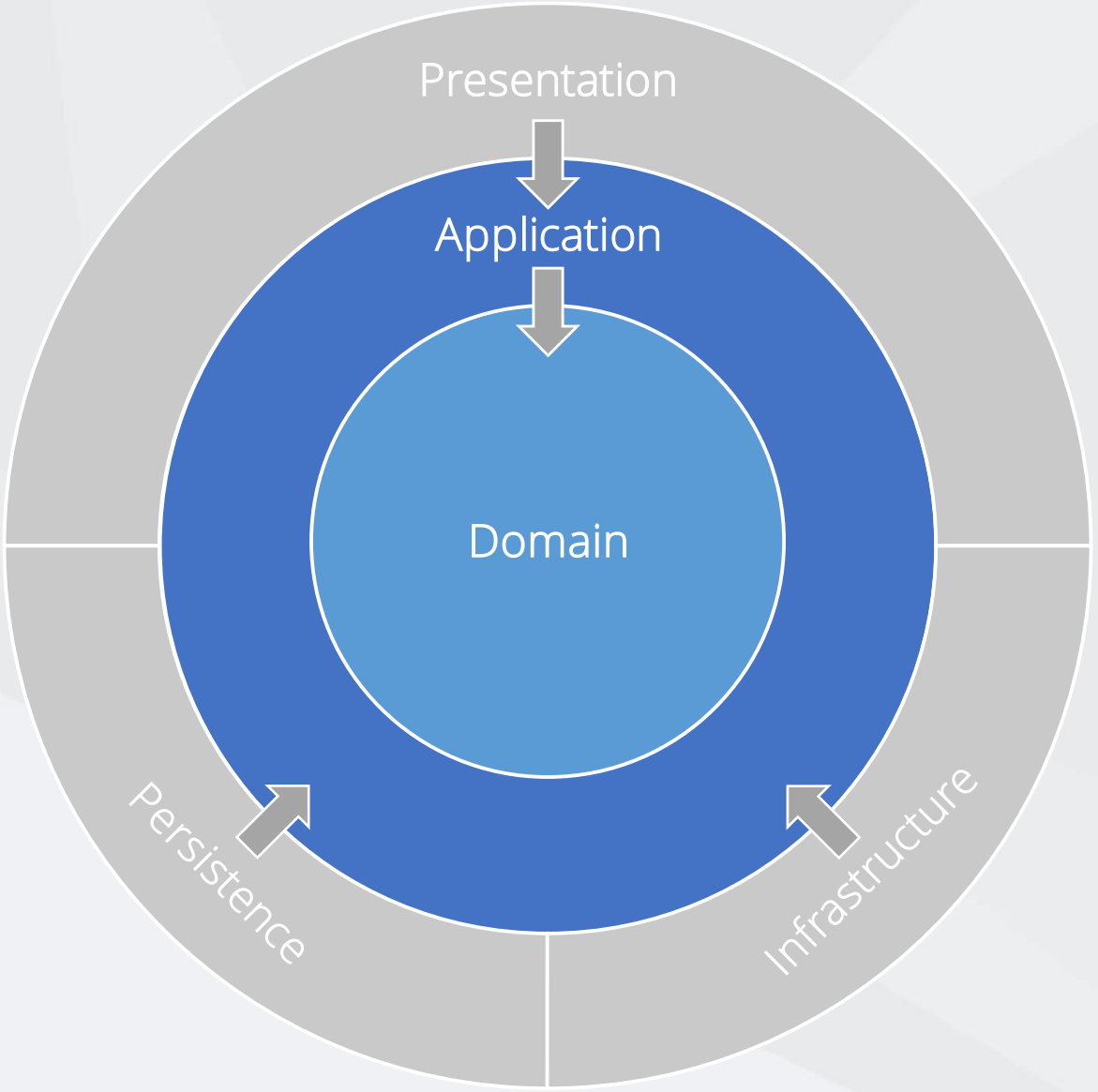
Unit Tests

xUnit.net

EF Core InMemory

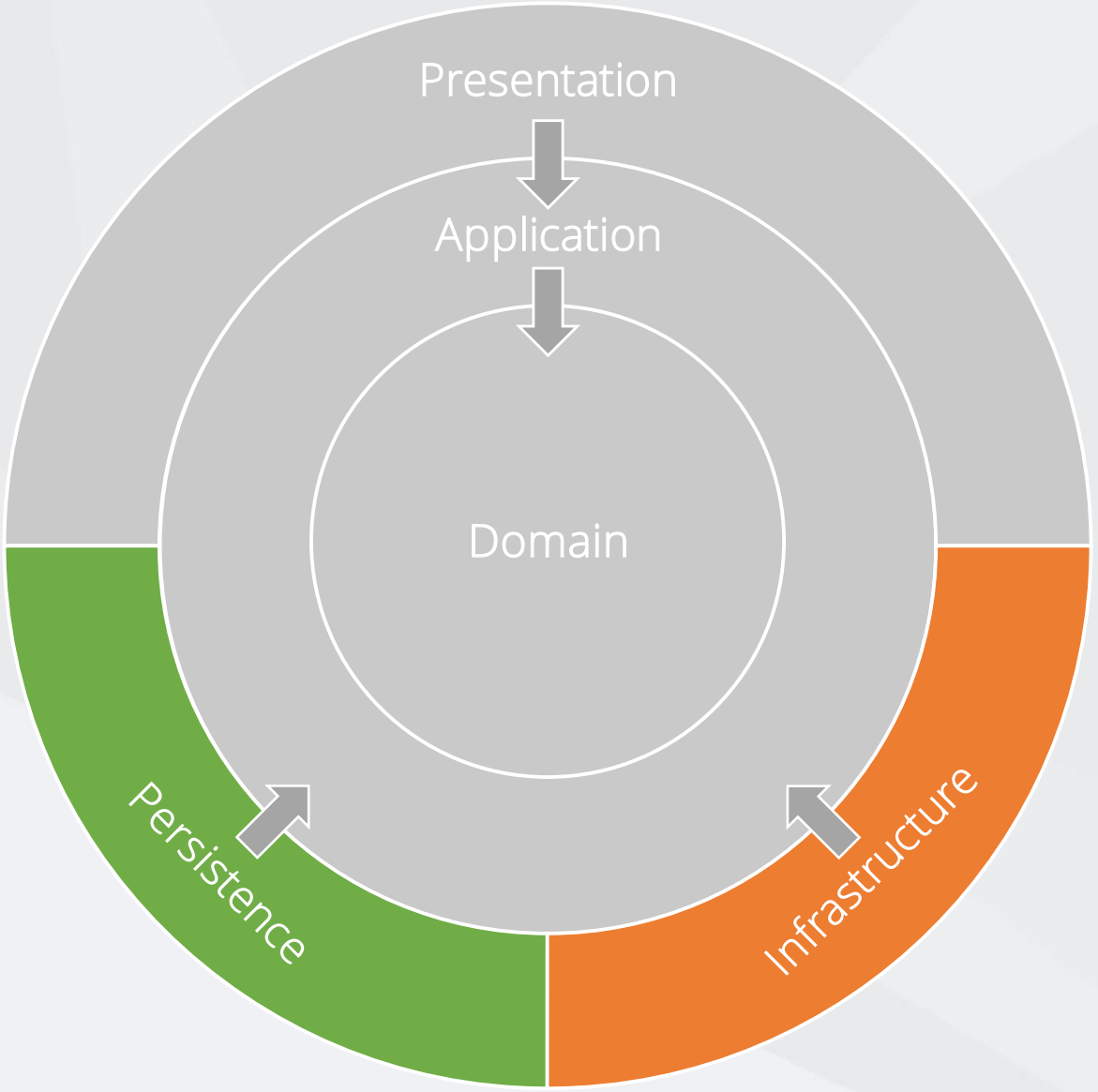
Moq or similar

Live Unit Testing



Integration Tests

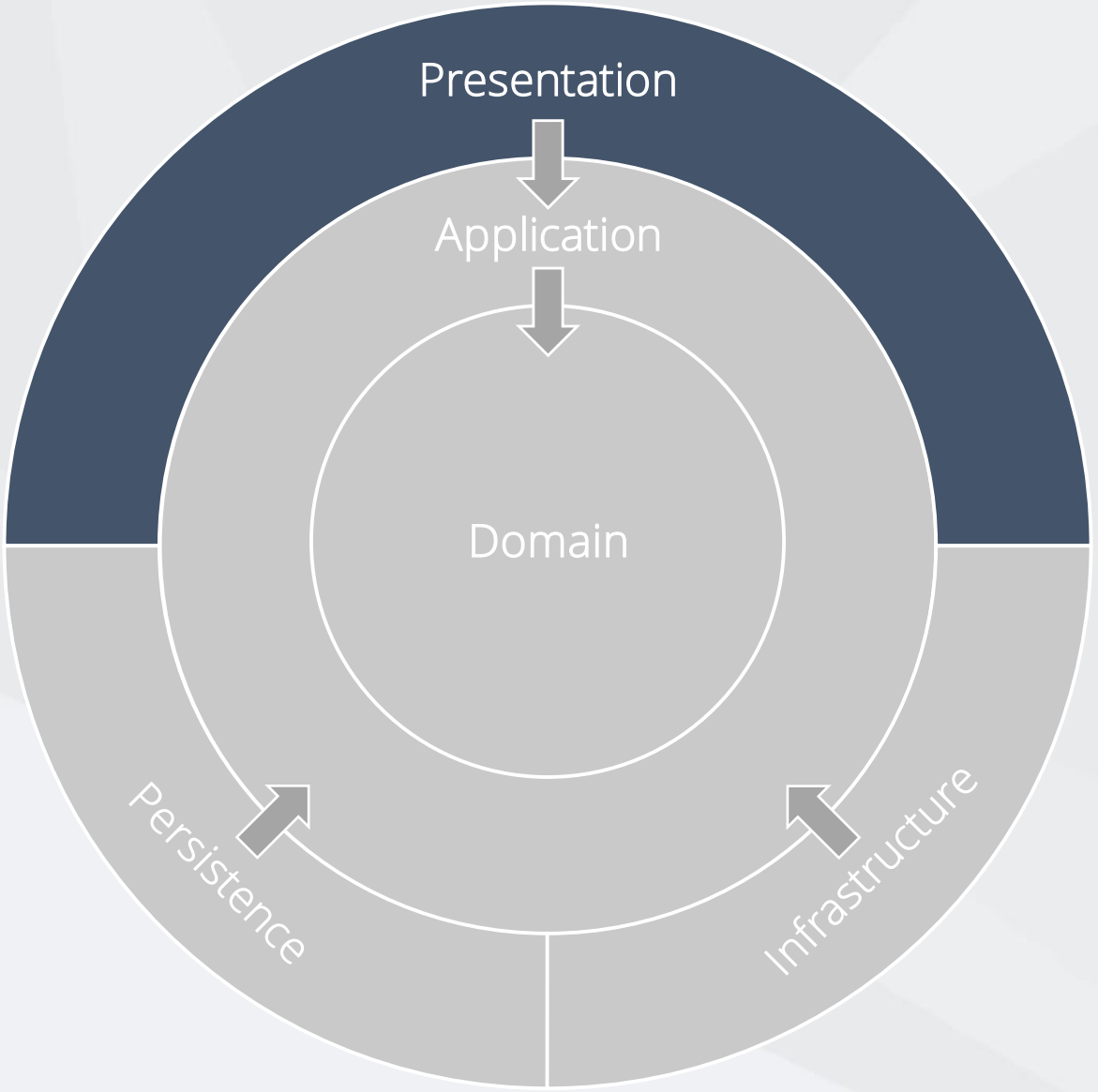
xUnit.net



UI Tests

Coded UI

Selenium



Test-Driven Development (TDD)

A workflow for writing test first unit tests

The cycle is:

1. Add a failing test
2. Run all tests and see the new one fail (Red)
3. Write some code
4. Run all tests again and see the new one pass (Green)
5. Refactor changed code (Refactor)

Why is import to write a failing test first?

Testing Frameworks

Demo: .NET Core CLI

Creating and running tests using the .NET Core CLI

Demo: Unit Testing Logic

Getting started with xUnit and Shouldly

Unit Testing

Unit Testing in .NET Core (bit.ly/testing-dotnet-core)

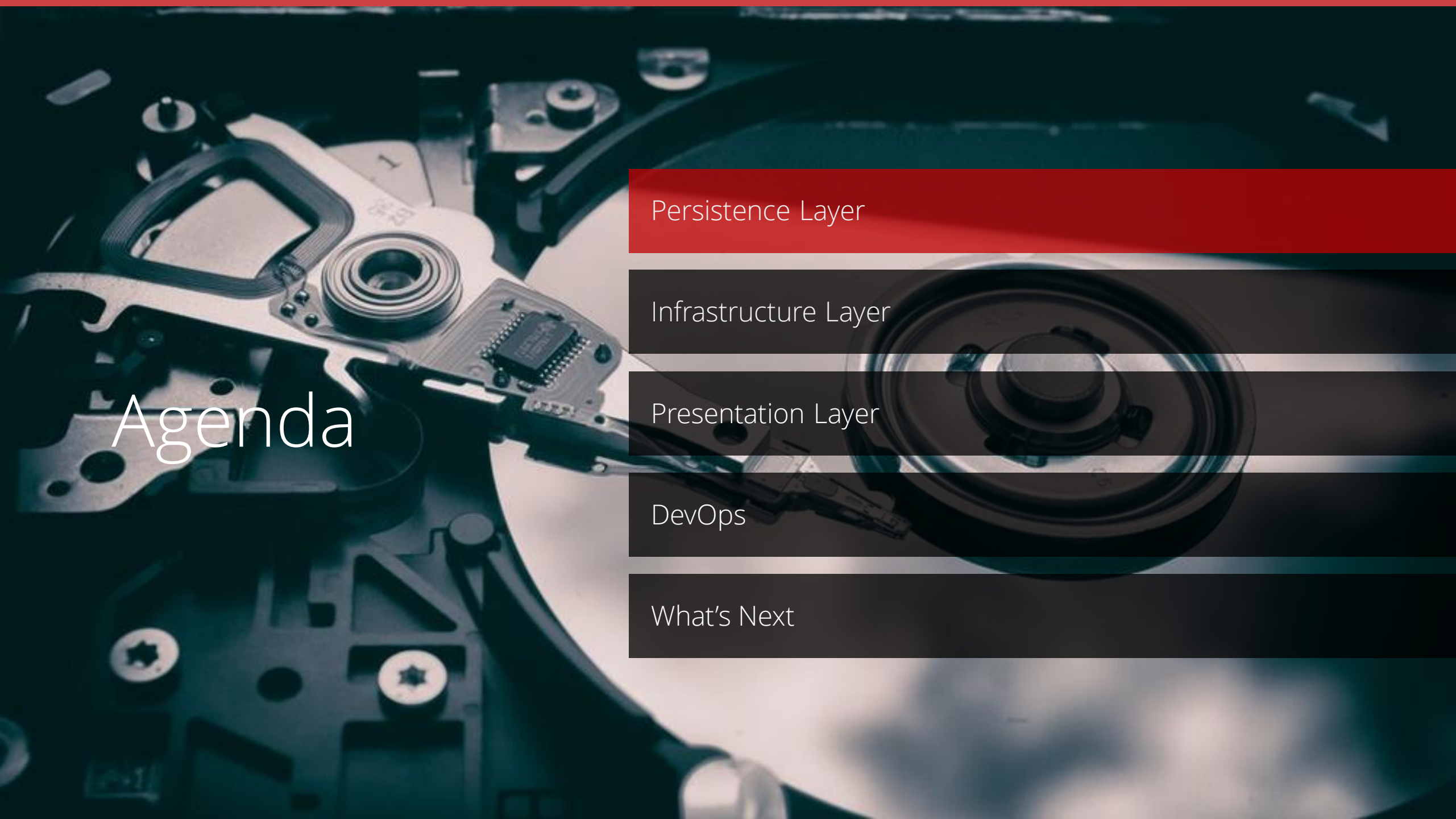
The Bowling Game Kata (bit.ly/bowing-game-kata)

Test-Driven Development (bit.ly/tdd-beck)

The Art of Unit Testing (bit.ly/aut-osherove)

Key Points

- Write your tests first
- Unit tests for Core layers (very high coverage)
- Integration tests for Infrastructure layers
- UI Tests (sparingly) for Presentation layers
- Use EF Core InMemory Provider



Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

Overview

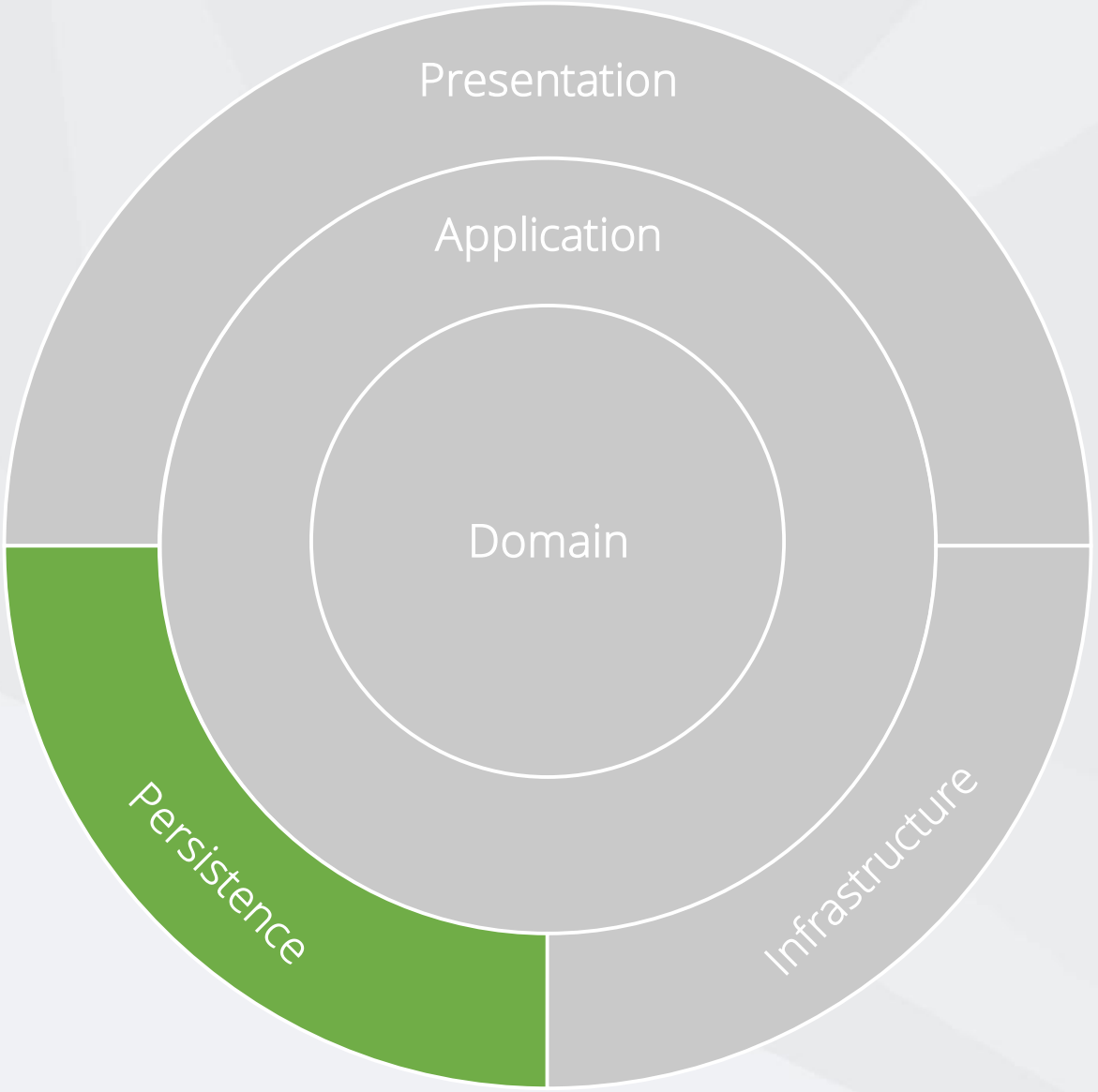
Entity Framework

Repository Pattern

Specification Pattern

Projections and Query Types

Migrations



Entity Framework Core

ORM – Object Relational Mapping

Impedance mismatch

Lightweight, extensible, and cross-platform, open source

Part of .NET Core – with versioning in sync

Runs on .NET Core or .NET Framework

Improve productivity when working with SQL

All ORMs can be dangerous / slow when misused

Entity Framework Concepts

DbContext – “Heavyweight” ORM that tracks changes

States: Added, Unchanged, Modified, Deleted, Detached

“Code First" Only – but you can Scaffold

Migrations – Used to generate and publish schema changes

Unit of Work and Repository Patterns

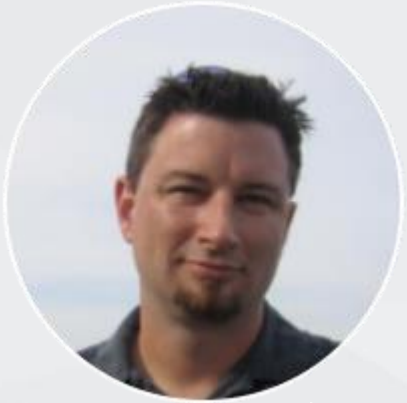
Should we implement these patterns?

It isn't always the best choice, because:

- EF Core insulates your code from database changes
- DbContext acts as a unit of work
- DbSet acts as a repository
- EF Core has features for unit testing without repositories

What do the experts think?

I'm over Repositories, and definitely over abstracting your data layer.



No, you don't *need* a repository. But there are many benefits and you should consider it!

No, the repository/unit-of-work pattern isn't useful with EF Core.

Unit of Work and Repository Patterns

“Leaky abstractions” - but I can write Ad-Hoc queries.

“No Leaky abstractions” - but then is verbose.

Injecting Repositories into Controllers?

CQRS commands and queries provide a better abstraction.

Revisiting CQRS

~~Event Sourcing~~

~~Separate data stores for read and writes~~

Clearly separated command and query operations

Specific view models from our read operations

Freedom to change or evolve our persistence

CQRS and EF Core

Read Operations

- Select / Project into View Models
- Use Query Types
- No Change Tracking

Write Operations

- Work with entities
- Change Tracking, Relationship management etc.

The Purist

- My application layer should contain no persistence dependencies
- I can mock the persistence layer for unit tests
- I can change the persistence layer without changing the application.

The Pragmatist

- I consider EF Core to be part of the .NET Framework
- I can be more productive
- EF Core in-memory provider is a mock provider.
- CQRS commands and queries in the application layer is a better abstraction.

Demo

Reviewing the Persistence layer

Demo: Projecting to ViewModels

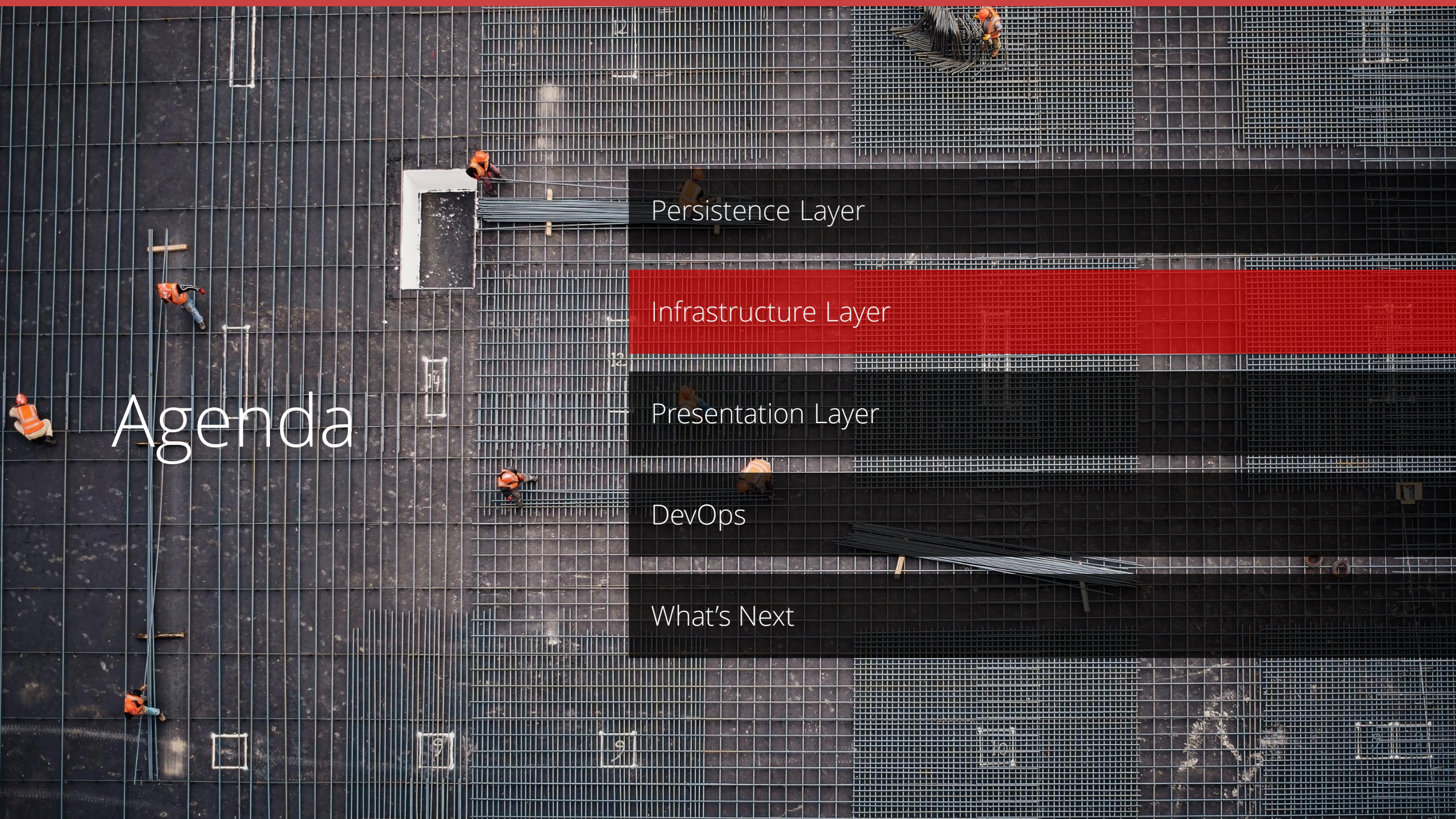
Managing data with DTOs and ViewModels

Demo: Using Query Types

Using Query Types for complex read queries

Key Points

- Independent of the database
- Prefer conventions over configuration
- Use Fluent API Configuration over Data Annotations
- Automatically apply all entity type configurations



Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

Overview

Implementations, e.g.

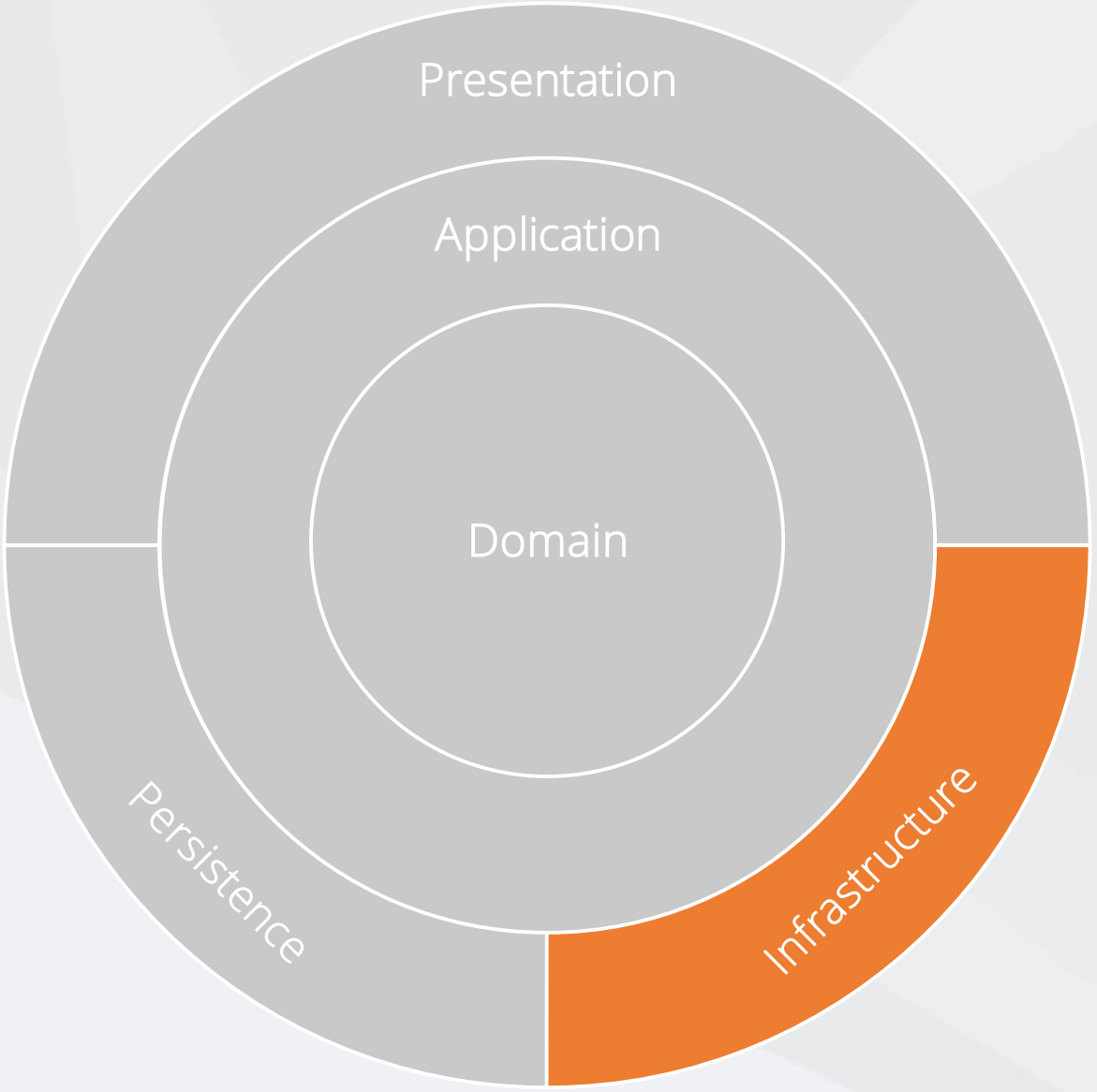
API Clients

File System

Email / SMS

System Clock

Anything external



Demo

Reviewing the Infrastructure layer

Security Choices

No Security

Windows Authentication

ASP.NET Core Identity

Identity Server 4

Windows Authentication

Easy to Implement (Hosted on windows)

Good option for Intranet

Can use reverse proxy such as F5 to publish to internet

System.DirectoryServices – windows only via compatibility pack

ADFS can support OIDC and OAuth2

Windows Authentication: Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<IISOptions>(options => options.ForwardWindowsAuthentication = true);
    // Add framework services.
    services.AddMvc();
}
```

Windows Authentication: web.config

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <system.webServer>
        <aspNetCore forwardWindowsAuthToken="true" processPath="%LAUNCHER_PATH%" arguments="%LAUNCHER_ARGS%" /
        <handlers>
            <add name="aspNetCore" path="*" verb="*" modules="AspNetCoreModule" resourceType="Unspecified" />
        </handlers>
    </system.webServer>
</configuration>
```

ASP.NET Core Identity

Membership provider: Registration, Reset Password, Attempt Limits

Support for 2FA – interfaces for Email & SMS Senders

Persistence to SQL Server (Via EF Core) or your own persistence store

Support for external providers via OAuth & OpenID

ASP.NET Core Identity - Startup.cs

```
public void ConfigureServices(IServiceCollection services)
{
    // Add framework services.
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));

    services.AddIdentity<ApplicationUser, IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();
```

Startup.cs – Configure(IApplicationServices app)

```
app.UseStaticFiles();

app.UseIdentity();

app.UseMvc(routes =>
```

```
// Configure Identity
services.Configure<IdentityOptions>(options =>
{
    // Password settings
    options.Password.RequireDigit = true;
    options.Password.RequiredLength = 8;
    options.Password.RequireNonAlphanumeric = false;
    options.Password.RequireUppercase = true;
    options.Password.RequireLowercase = false;

    // Lockout settings
    options.Lockout.DefaultLockoutTimeSpan = TimeSpan.FromMinutes(30);
    options.Lockout.MaxFailedAccessAttempts = 10;

    // Cookie settings
    options.Cookies.ApplicationCookie.ExpireTimeSpan = TimeSpan.FromDays(150);
    options.Cookies.ApplicationCookie.LoginPath = "/Account/LogIn";
    options.Cookies.ApplicationCookie.LogoutPath = "/Account/LogOut";

    // User settings
```

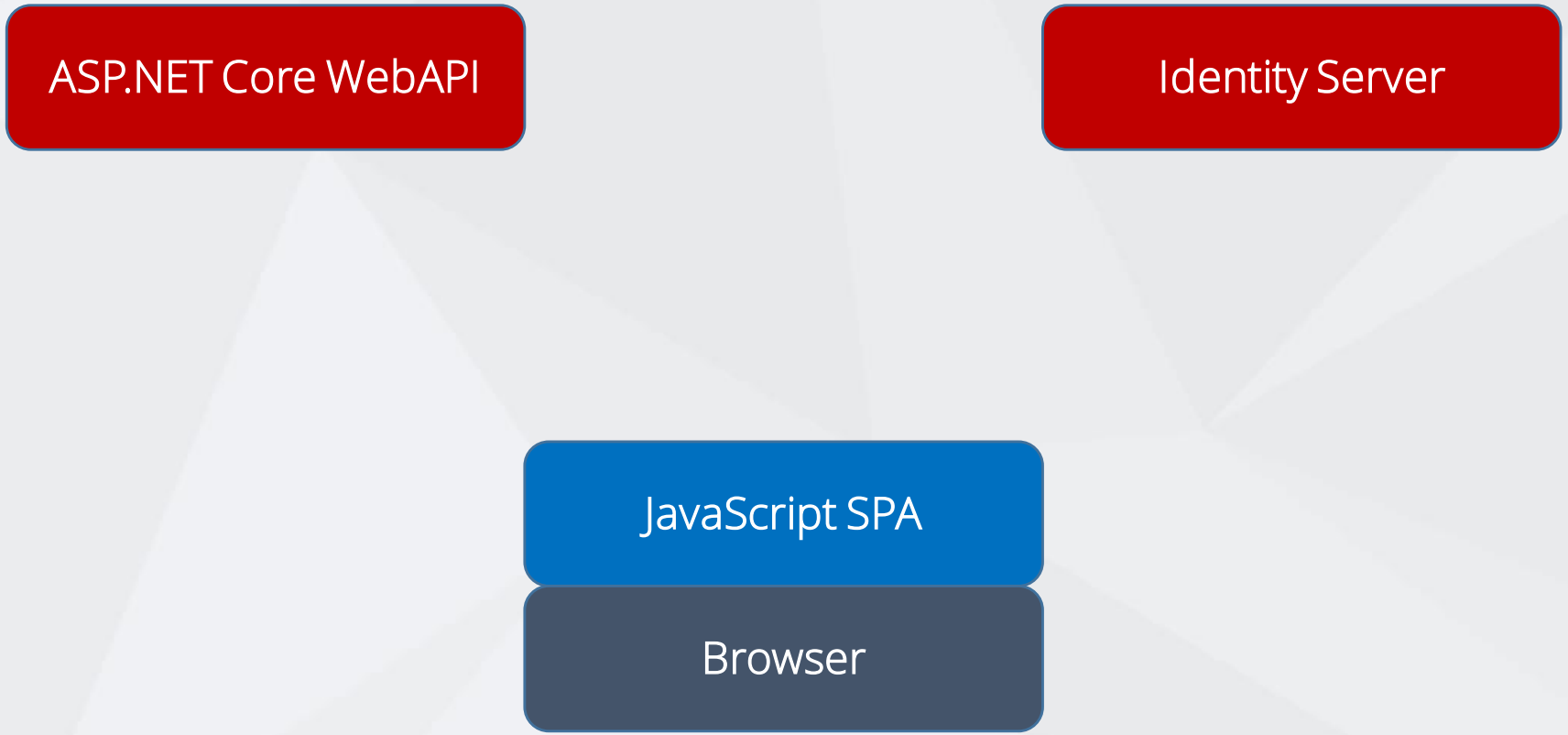
Identity Server 4

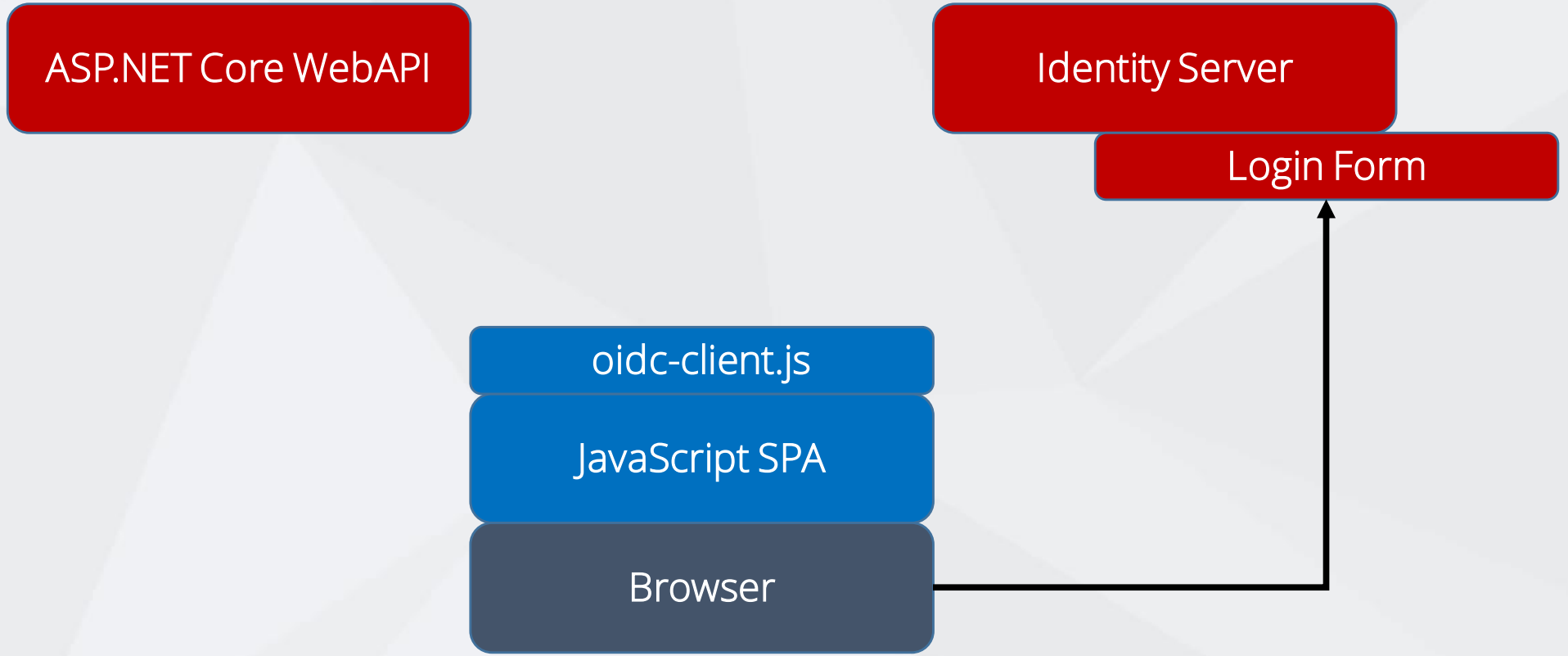
Full Single Sign On solution

Service that provides OpenId Connect and OAuth 2.0

Implemented in .NET Core

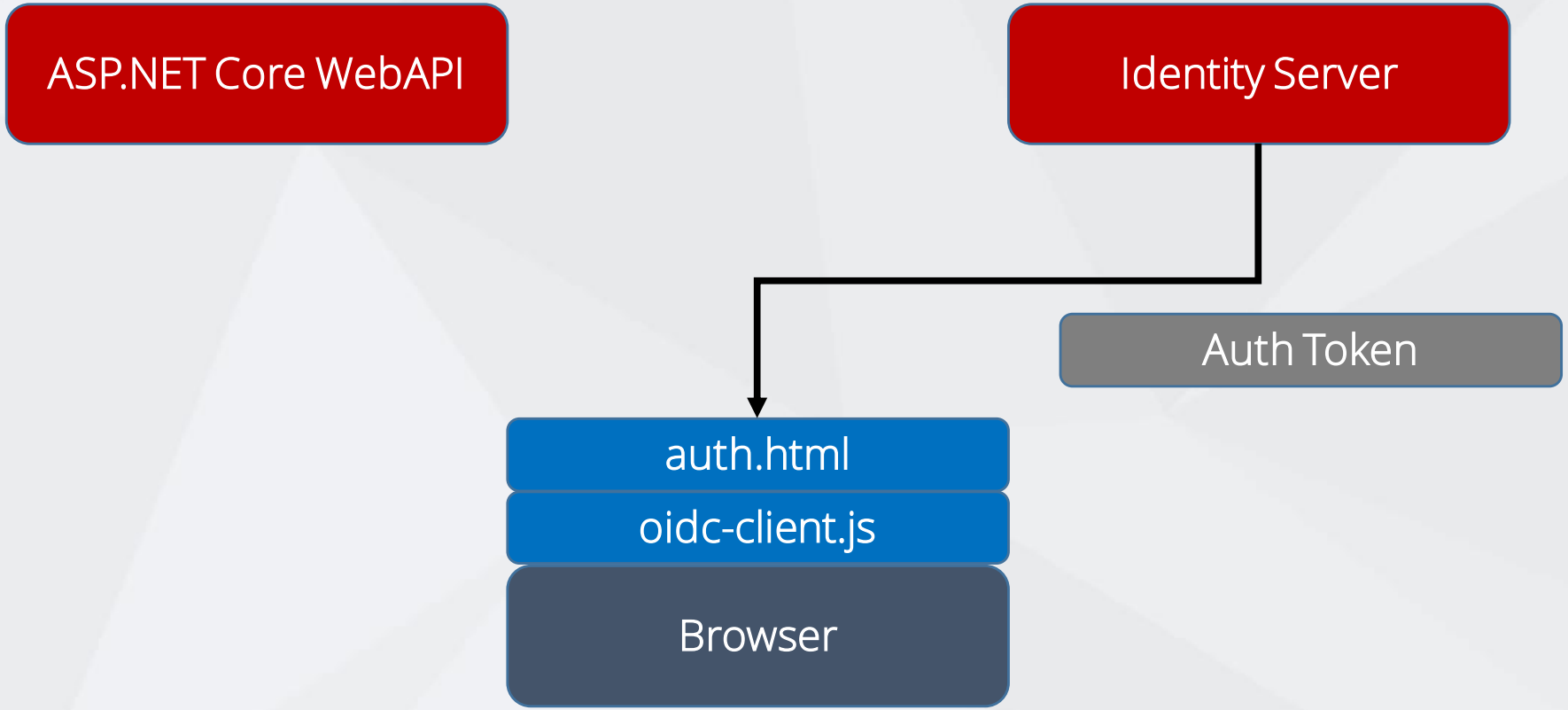
Example: JS application with .NET Core WebAPI


ASP.NET Core WebAPI
Identity Server
JavaScript SPA
Browser



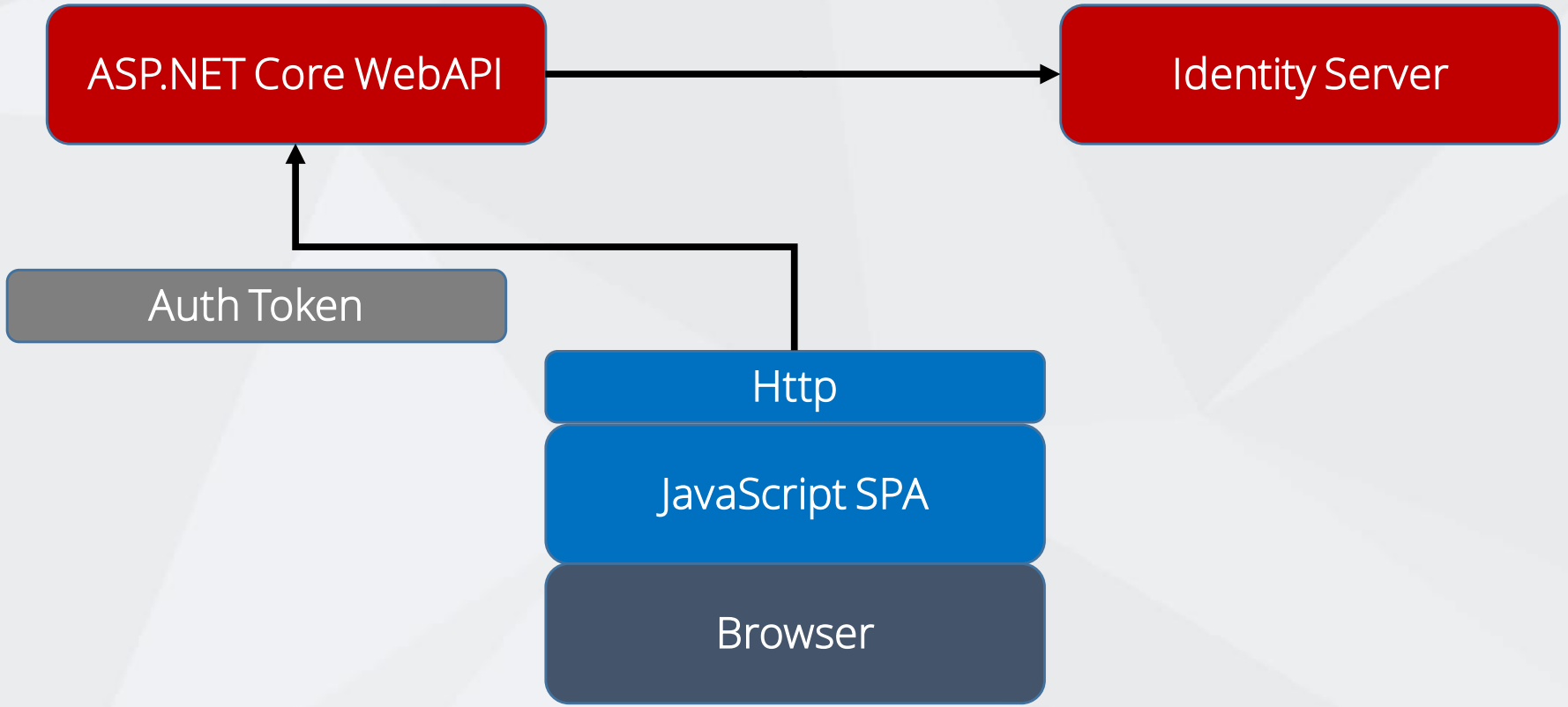
OIDC client redirects browser to a login page on the Identity Server.

User Logs in.



Identity Server redirects back with an Authorization Token

Token is saved on the browser



Send the Auth Token as a HTTP header with all API requests

The API server checks the token against the Identity Server

```
app.UseIdentityServerAuthentication(new IdentityServerAuthenticationOptions
{
    Authority = "https://[redacted]",//Configurati
    RequireHttpsMetadata = false,
    ApiName = "https://[redacted]",//"gr
    NameClaimType = "name",
    RoleClaimType = "role",
    AllowedScopes = { "offline_access", "profile", "openid" },
});
```

Key Points

- Contains classes for accessing external resources
- Such as file systems, web services, SMTP and so on
- Implements abstractions / interfaces defined within the Application layer
- No layers depend on Infrastructure layer, e.g. Presentation layer

Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

Overview

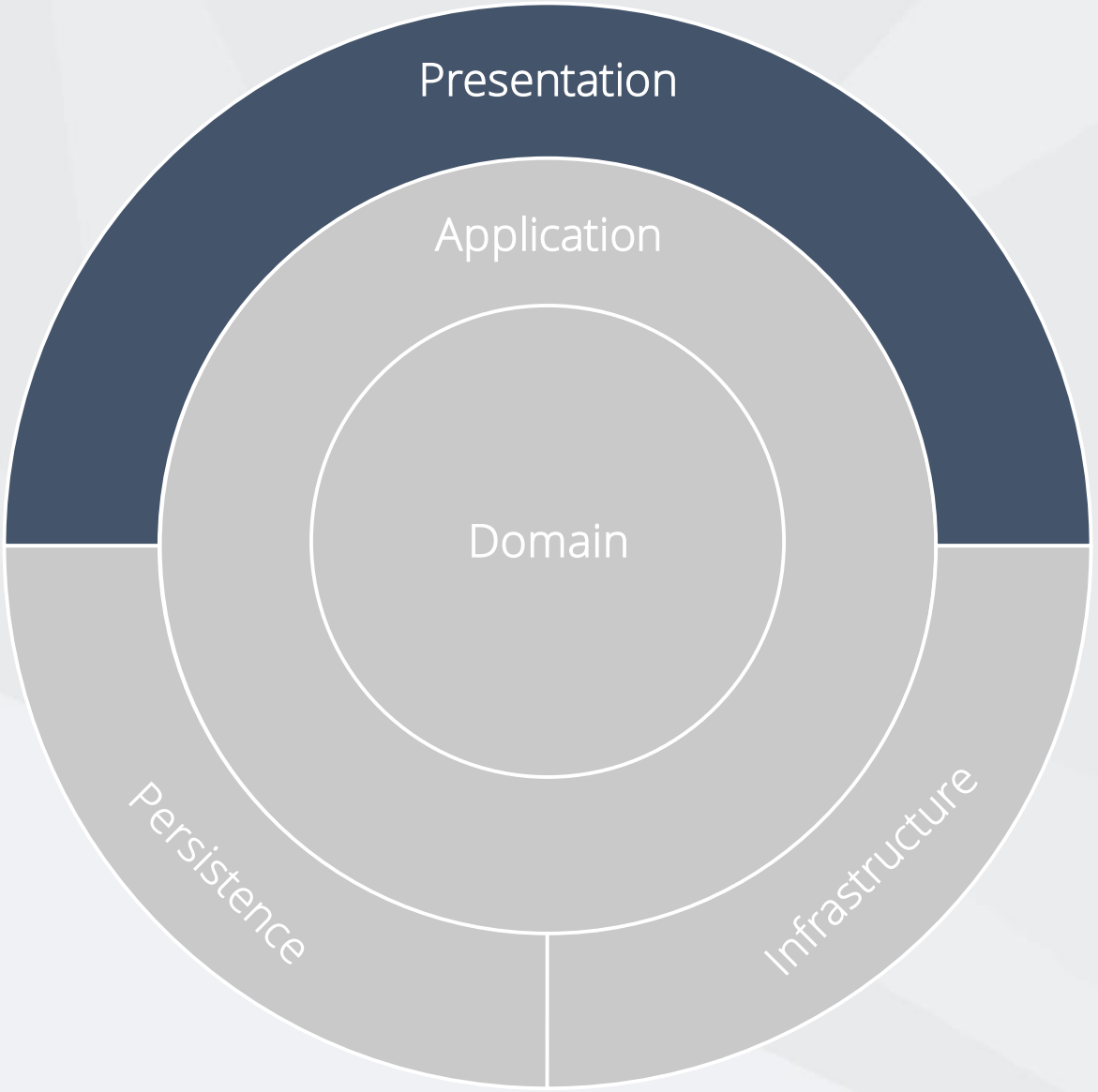
SPA – Angular or React

Web API

Razor Pages

MVC

Web Forms



Demo: Evolution of the Controller

Controller Features

Model Binding

Result Functions & HTTP Status Codes

Routing – Conventions or Attributes

Content Negotiation & Custom formatters

ASP.NET Core Middleware

Low level pipeline for Request -> Response

All features are explicitly added (including static files)

You can write your own middleware

Similar to OWIN (from previous Web API projects)

ASP.NET Core Middleware

```
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseSpaStaticFiles();

app.UseSwaggerUi3(settings =>
{
    settings.SwaggerUiRoute = "/api";
    settings.SwaggerRoute = "/api/specification.json";
});

app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "default",
        template: "{controller}/{action=Index}/{id?}");
});
```

ASP.NET Core Middleware

```
public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        app.Run(async context =>
        {
            await context.Response.WriteAsync("Hello, World!");
        });
    }
}
```

ASP.NET Core Middleware

```
public class Startup
{
    public void Configure(IApplicationBuilder app)
    {
        app.Use(async (context, next) =>
        {
            // Do work that doesn't write to the Response.
            await next.Invoke();
            // Do logging or other work that doesn't write to the Response.
        });

        app.Run(async context =>
        {
            await context.Response.WriteAsync("Hello from 2nd delegate.");
        });
    }
}
```

ASP.NET Core Middleware

```
private static void HandleMapTest2(IApplicationBuilder app)
{
    app.Run(async context =>
    {
        await context.Response.WriteAsync("Map Test 2");
    });
}

public void Configure(IApplicationBuilder app)
{
    app.Map("/map1", HandleMapTest1);

    app.Map("/map2", HandleMapTest2);

    app.Run(async context =>
    {
        await context.Response.WriteAsync("Hello from non-Map delegate. <p>");
    });
}
```

ASP.NET Core Filters

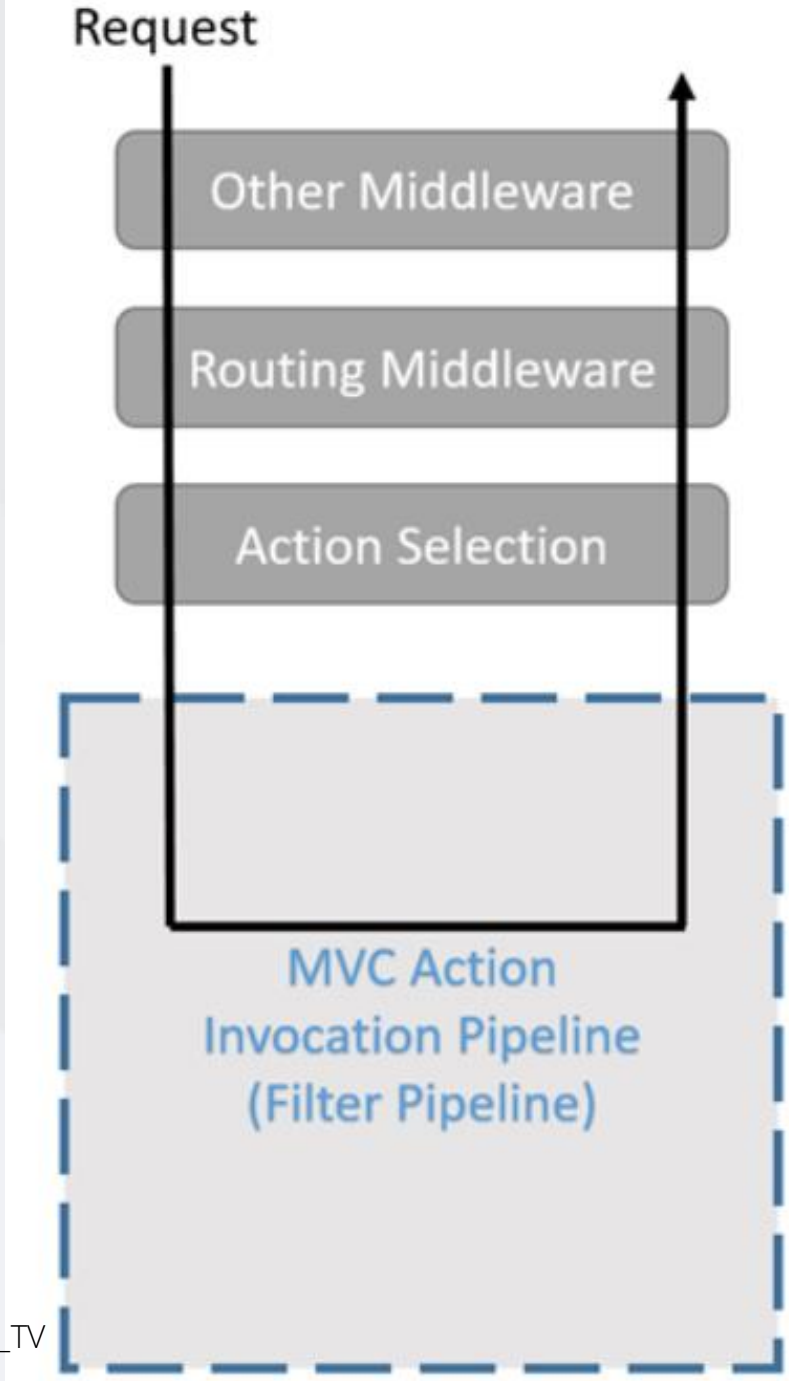
Authorization

Resource

Action

Exception

Result



```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method)]
1 reference | Jason Taylor, 95 days ago | 1 author, 1 change
public class CustomExceptionFilterAttribute : ExceptionFilterAttribute
{
    0 references | Jason Taylor, 95 days ago | 1 author, 1 change | 0 exceptions
    public override void OnException(ExceptionContext context)
    {
        if (context.Exception is ValidationException)
        {
            context.HttpContext.Response.ContentType = "application/json";
            context.HttpContext.Response.StatusCode = (int)HttpStatusCode.BadRequest;
            context.Result = new JsonResult(
                ((ValidationException)context.Exception).Failures);

            return;
        }
```

Single Page Applications

ASP.NET Core has built-in support for SPA

Supports all major client-side frameworks

Cross-platform Windows, Mac, or Linux

Host SPA & API from one domain (no CORS)

URL Rewrite included for SPA client-side routing

Demo: Single Page Applications

Video:

https://youtu.be/HpPGLHKlaM0?t=2530

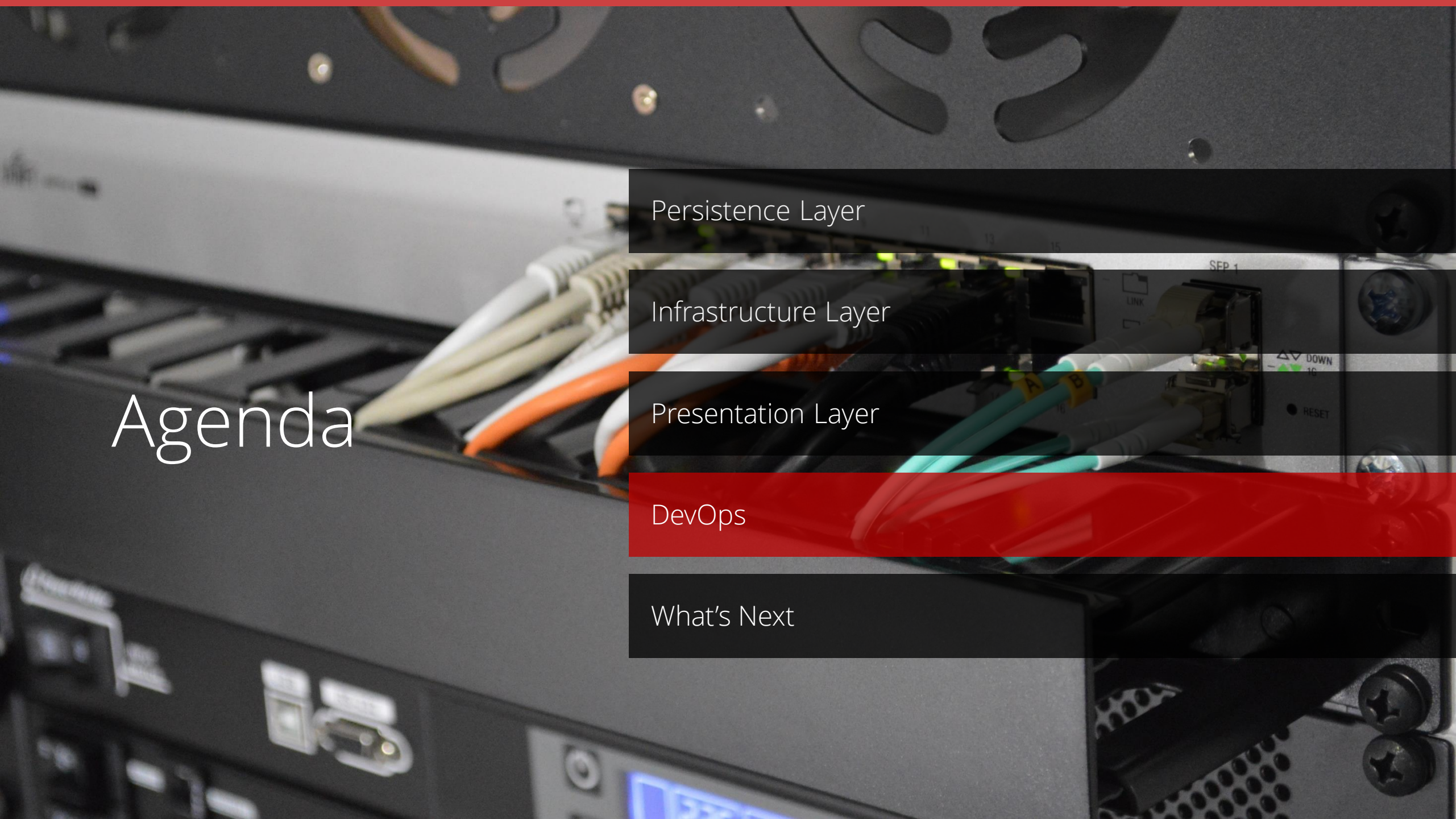
Demo: Full Stack Rx Extensions

Video:

https://www.youtube.com/watch?v=jE65d8b3w_M

Key Points

- Controllers should not contain any application logic
- Create and consume well defined view models
- Utilising Open API bridges the gap between the front end and back end



Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

Essential Practices

Agile Planning

Version Control

Continuous Integration (CI)

Continuous Delivery (CD)

Monitoring & Logging

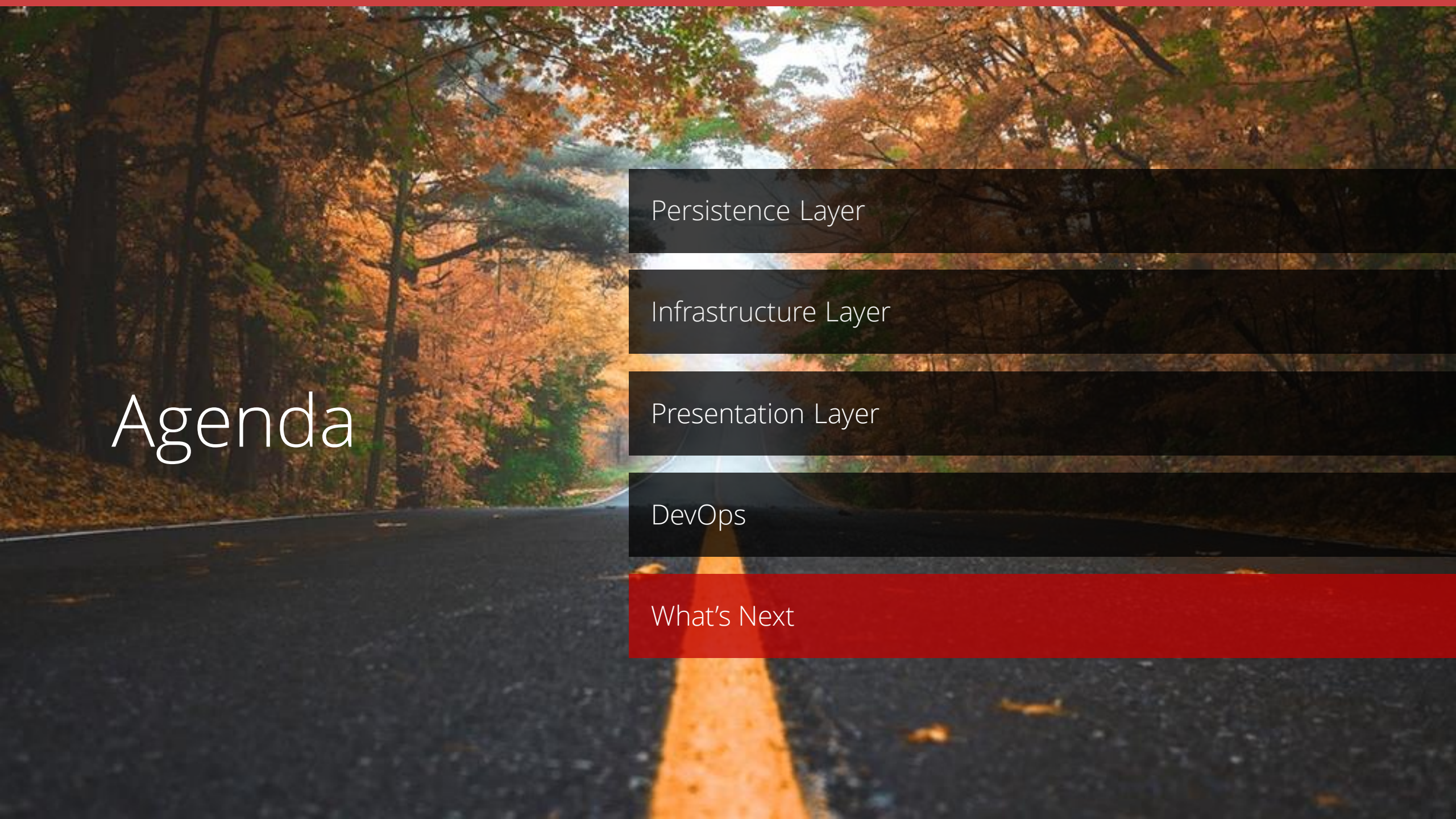
Demo: Azure DevOps Project

Implement essentials practices in four easy steps.

Key Points

- Agile Planning
- Version Control
- Continuous Integration (CI)
- Continuous Delivery (CD)
- Monitoring & Logging
- Azure DevOps Project


If it hurts, do it more often.



Agenda

- Persistence Layer
- Infrastructure Layer
- Presentation Layer
- DevOps
- What's Next

What's New in .NET Core 2.1

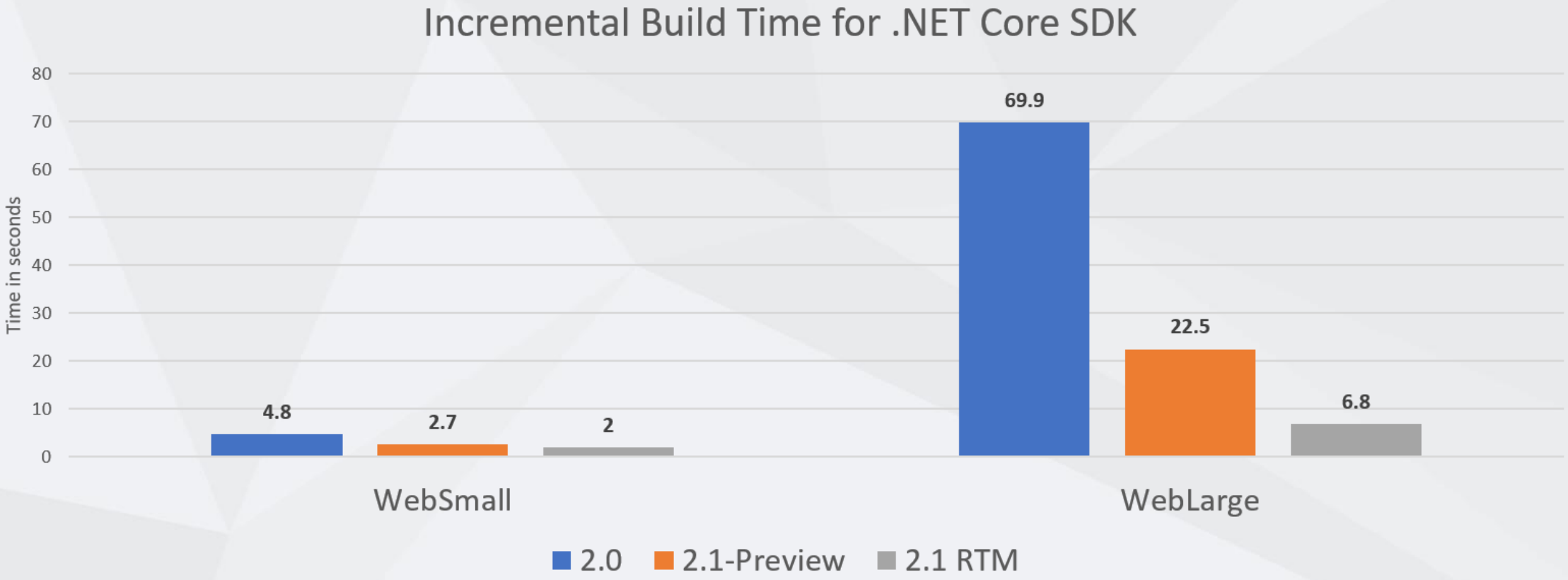
Windows Compatibility Pack

e.g. System.Drawing, Directory Services

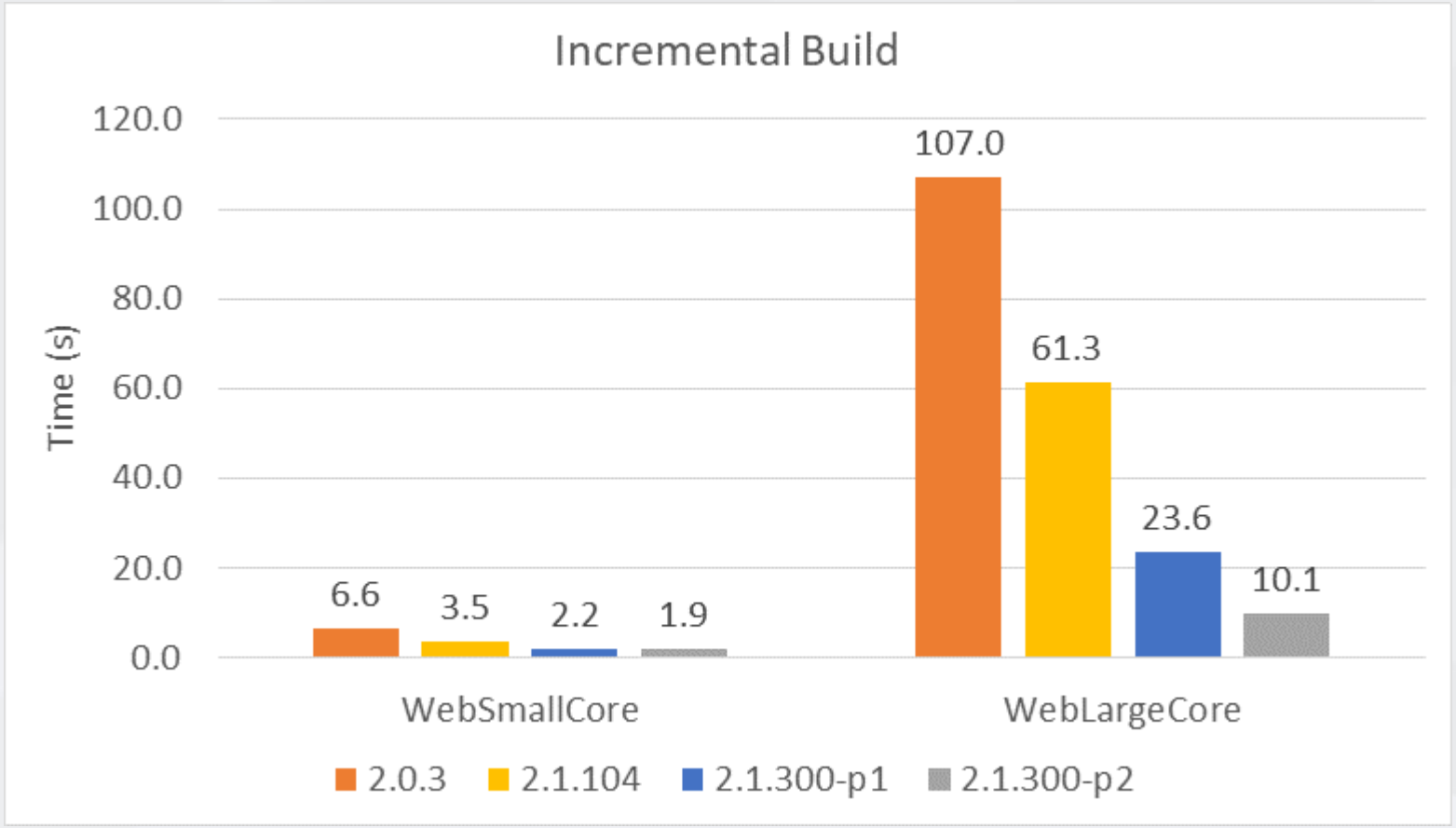
HttpClient – 10x performance improvement

Smaller runtime install size

Build Performance Improvements



Build Performance Improvements



What's New in ASP.NET Core 2.2

API Controller Conventions – (Better OpenAPI)

OpenAPI: spec & code generation

Health Checks

HTTP/2 support in Kestrel and HttpClient

SignalR – C++ and Java clients

What's New in Entity Framework Core 2.1

Lazy Loading

Entity Constructor Parameters

Group By Support

Query Types

What's New in Entity Framework Core 2.2

Spatial Data Support

Owned Collections

Query Tags (inject comments into SQL)

OwnedAttribute (one-to-one)

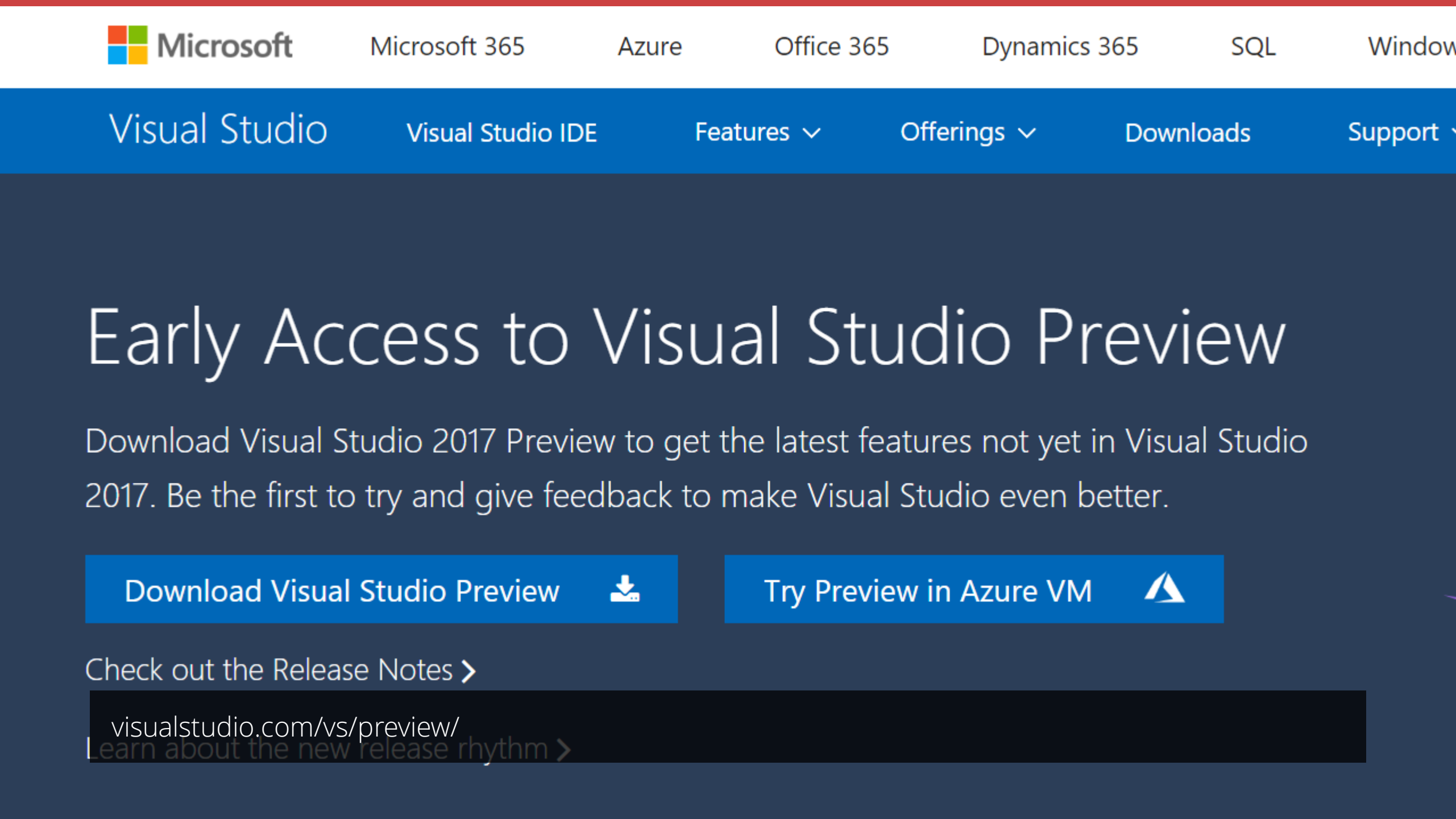
Data Seeding

What's Coming in .NET Core 3.0

Windows Desktop

WPF, WinForms, UWP

ASP.NET Core 3.0 will no longer support "full-framework"


Microsoft
Microsoft 365
Azure
Office 365
Dynamics 365
SQL
Window
Visual Studio
Visual Studio IDE
Features
Offerings
Downloads
Support
Early Access to Visual Studio Preview
Download Visual Studio 2017 Preview to get the latest features not yet in Visual Studio 2017. Be the first to try and give feedback to make Visual Studio even better.
Download Visual Studio Preview
Try Preview in Azure VM
Check out the Release Notes
visualstudio.com/vs/preview/
Learn about the new release rhythm

Visual Studio Dev Essentials

Free tools, cloud services, and training

Get everything you need to build and deploy your app on any platform. With state-of-the-art tools, the power of the cloud, training, and support, it's our most comprehensive free developer program ever.

Join or access now >

Everything you need all in one place

visualstudio.com/dev-essentials/

Developer tools

Cloud services

Software

Training and support

DURATION
1 Day

PRICE
$49 inc GST

Brisbane
MON 21ST MAY 2018
Book Now

Melbourne
THU 24TH MAY 2018
Book Now

Sydney
FRI 25TH MAY 2018
Book Now

Brisbane – Thursday, 14th February 2019 – All Day Event - $49 - firebootcamp.com/superpower-tours

Hosted by **SSW**

About the presenter

Thiago Passos

Thiago Passos joined SSW in 2014 as a Senior Software Architect. He's specialized in SharePoint and .NET, preferring C#.NET over VB.NET, working on windows and web applications and web services.

SERVICES PRODUCTS TRAINING USER GROUP RULES ABOUT US SSW TV

DURATION
1 Day

PRICE
$49 inc GST

Brisbane
MON 25TH JUN 2018
Book Now

Melbourne
THU 28TH JUN 2018
Book Now

Sydney
FRI 29TH JUN 2018
Book Now

Brisbane – Thursday, 7th March 2019 – All Day Event - $49 - firebootcamp.com/superpower-tours

Hosted by **SSW**

About the presenters

Brendan Richards

Throughout his career, Brendan has been a big user and proponent of Open Source software. This has been applied to a broad variety of [illegible] projects spanning the last 12 years.

Brendan has worked with an eclectic mix of languages, platforms and technologies including .Net, Java, PHP, Ruby and Perl

Accelerated learning, knowledge sharing and lots of Angular hacking.

WATCH RELATED VIDEOS

What is Angular Hack Day?

AngularJS Hack Days are community run events for Angular Developers or people who want to learn AngularJS for free.

There will be something for everyone. Experienced AngularJS developers can share ideas with other experienced developers. If you're a beginner then there's plenty to learn on the day.

Brisbane - Saturday, 1st June 2019 – All Day Event - Free - http://angularhackday.com/register

played with AngularJS before it would be good to do a little bit of learning before the day but it's not a requirement as all are welcome.

When and where?

Many cities around the world. Look for dates on a venue near you.

Event Organizers

Adam Stephensen, Duncan Hunter, Ben Cull, Eric Phan and Adam Cogan from **SSW Consulting**. You can contact them via this form.

Evaluation Form Reminder

Thank you!

Feedback + Q & A
Code & Slides bit.ly/netcore-superpowers

info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane