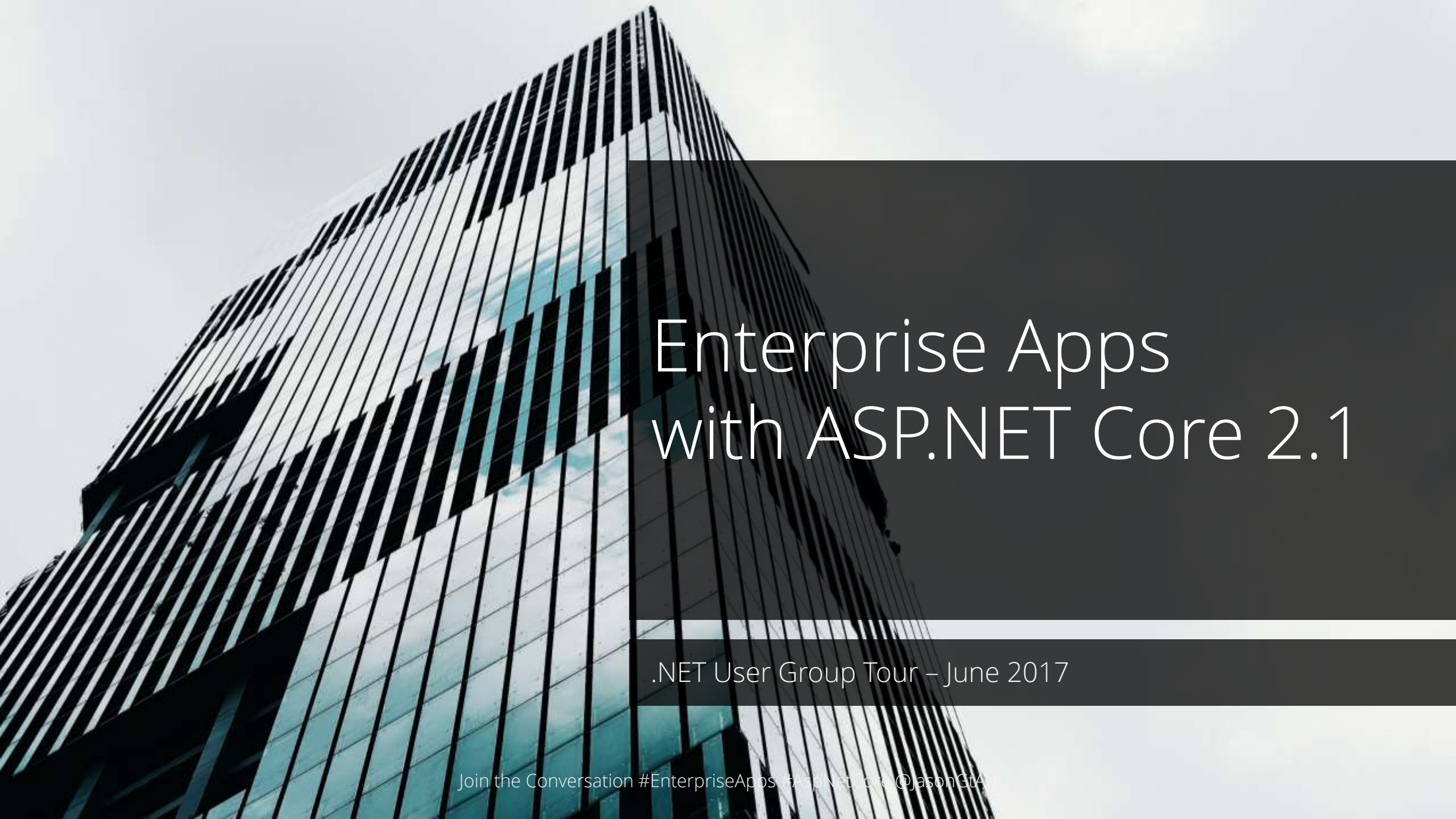
Enterprise Software Development

Join the Conversation #EnterpriseApps #AspNetCore @JasonGtAu



Enterprise Apps with ASP.NET Core 2.1

.NET User Group Tour – June 2017

Join the Conversation #EnterpriseApps #ASPNetCore @JasonGtA

Jason Taylor

SSW Solution Architect

@JasonGtAu @SSW_TV

codingflow.net

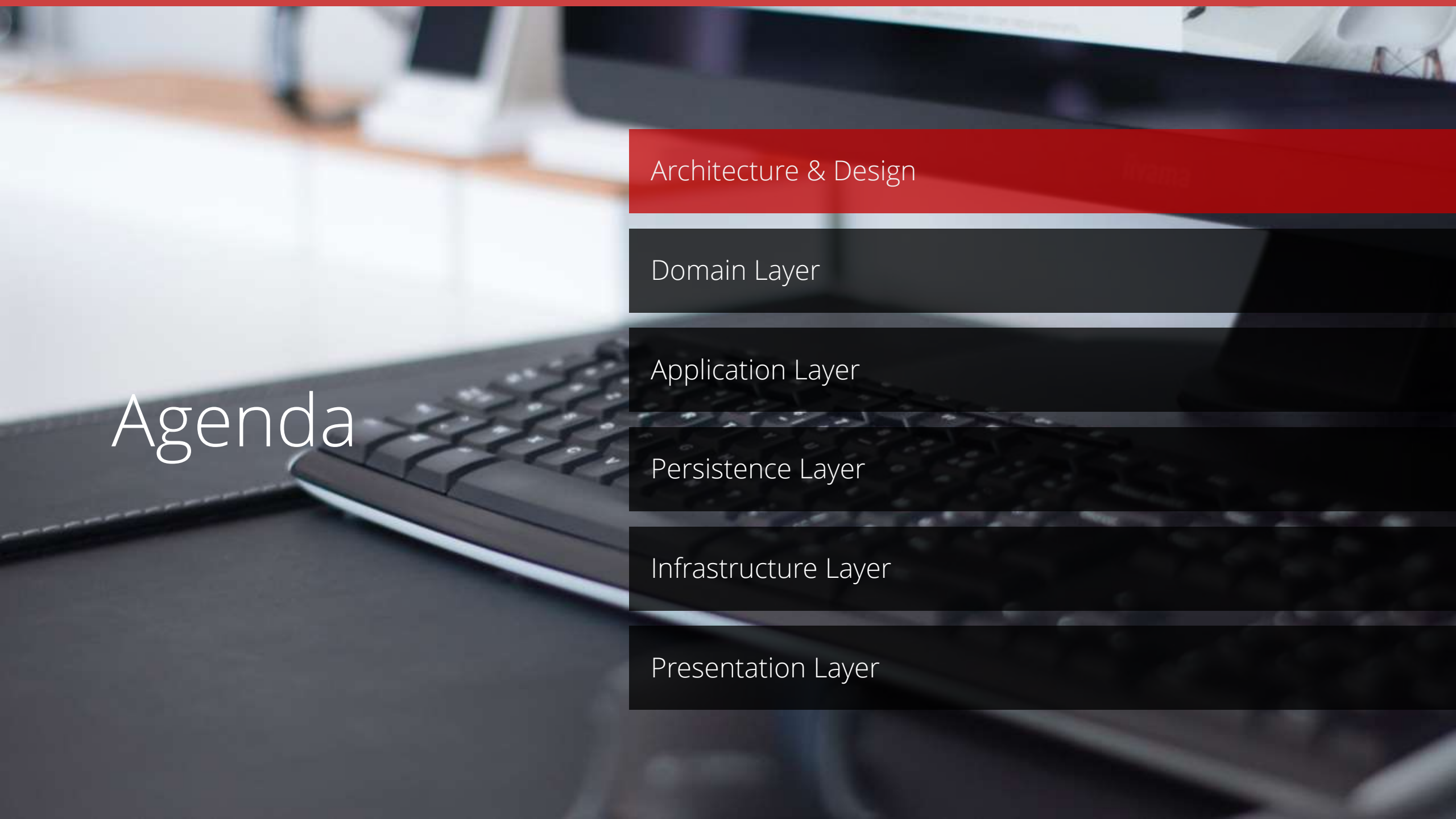
github.com/JasonGT

Developer* Since 1992

1 MHz CPU

20 KB Memory

Join the Conversation #EnterpriseApps #AspNetCore @JasonGtAu



Agenda

- Architecture & Design
- Domain Layer
- Application Layer
- Persistence Layer
- Infrastructure Layer
- Presentation Layer

Architecture & Design

Clean Architecture

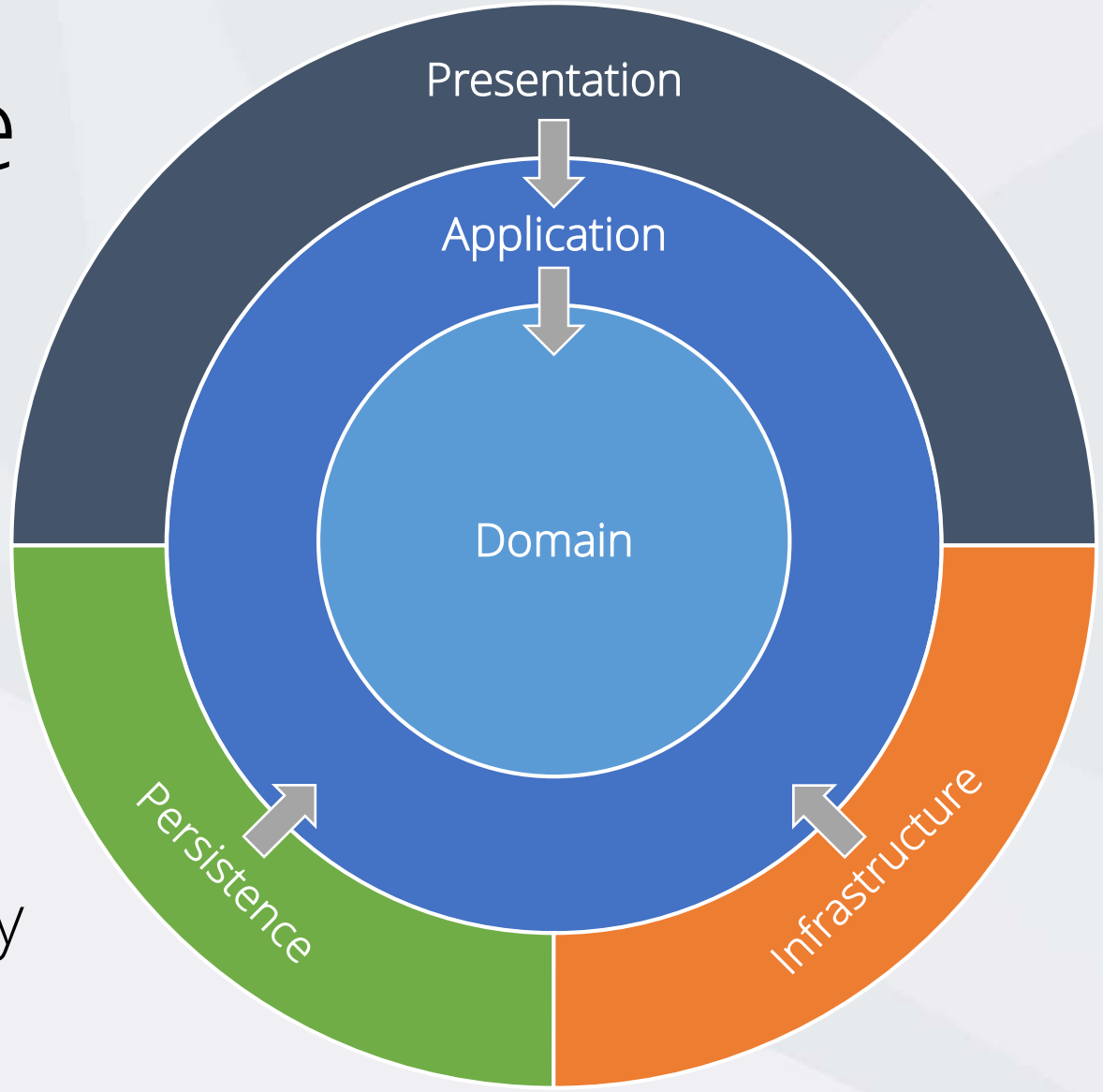
Independent of frameworks

Testable

Independent of UI

Independent of database

Independent of external agency



Unit of Work and Repository Patterns

Should we implement these patterns?

It isn't always the best choice, because:

- DbContext 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

Clean Architecture

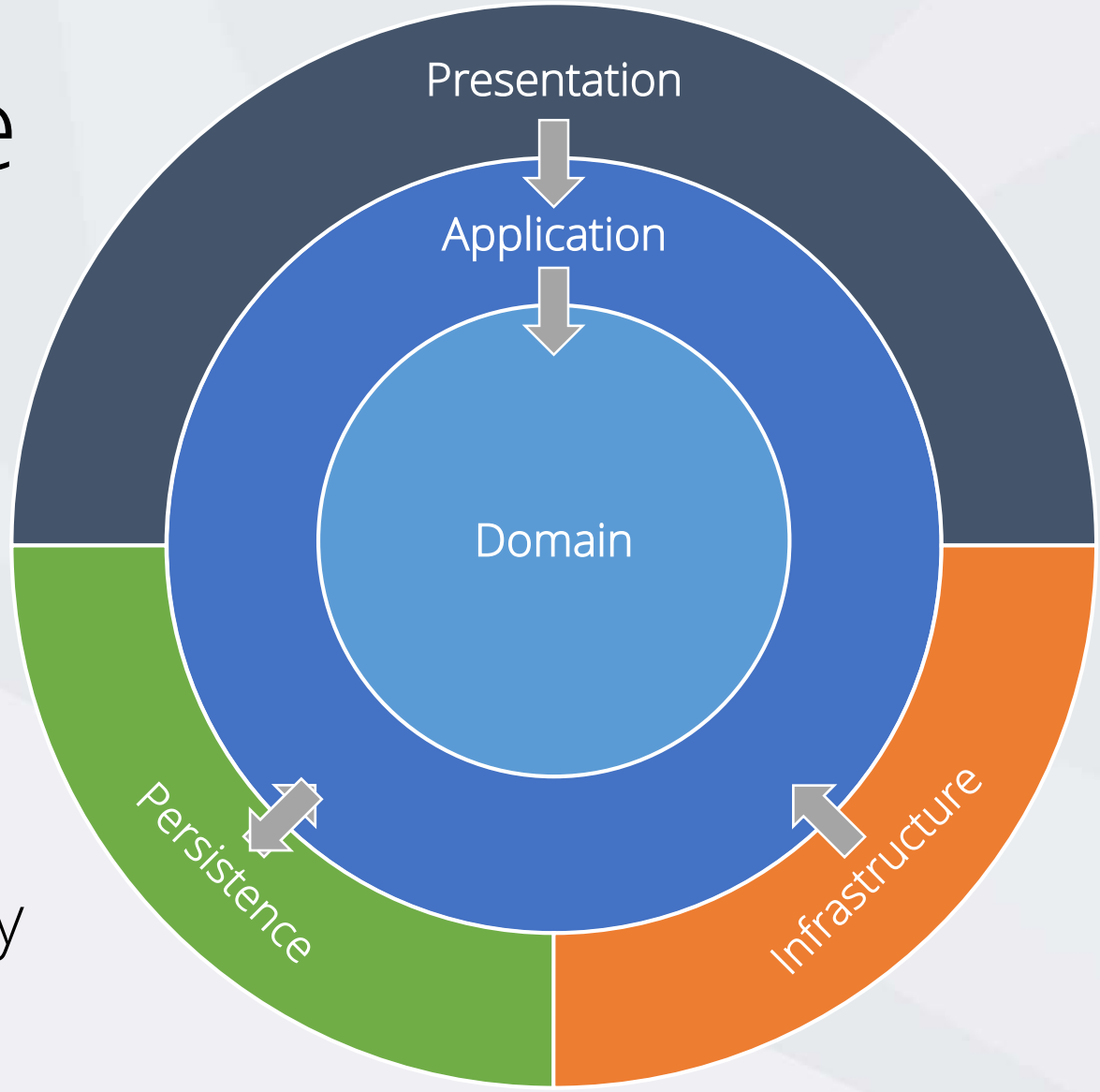
Independent of frameworks*

Testable

Independent of UI

Independent of database

Independent of external agency



Key Points

Domain contains enterprise-wide types and logic

Application contains application-specific models and logic

Infrastructure (including Persistence) contain all external concerns

Presentation contains frontend apps

Infrastructure and Presentation components can be replaced with minimal effort

Domain Layer

Overview

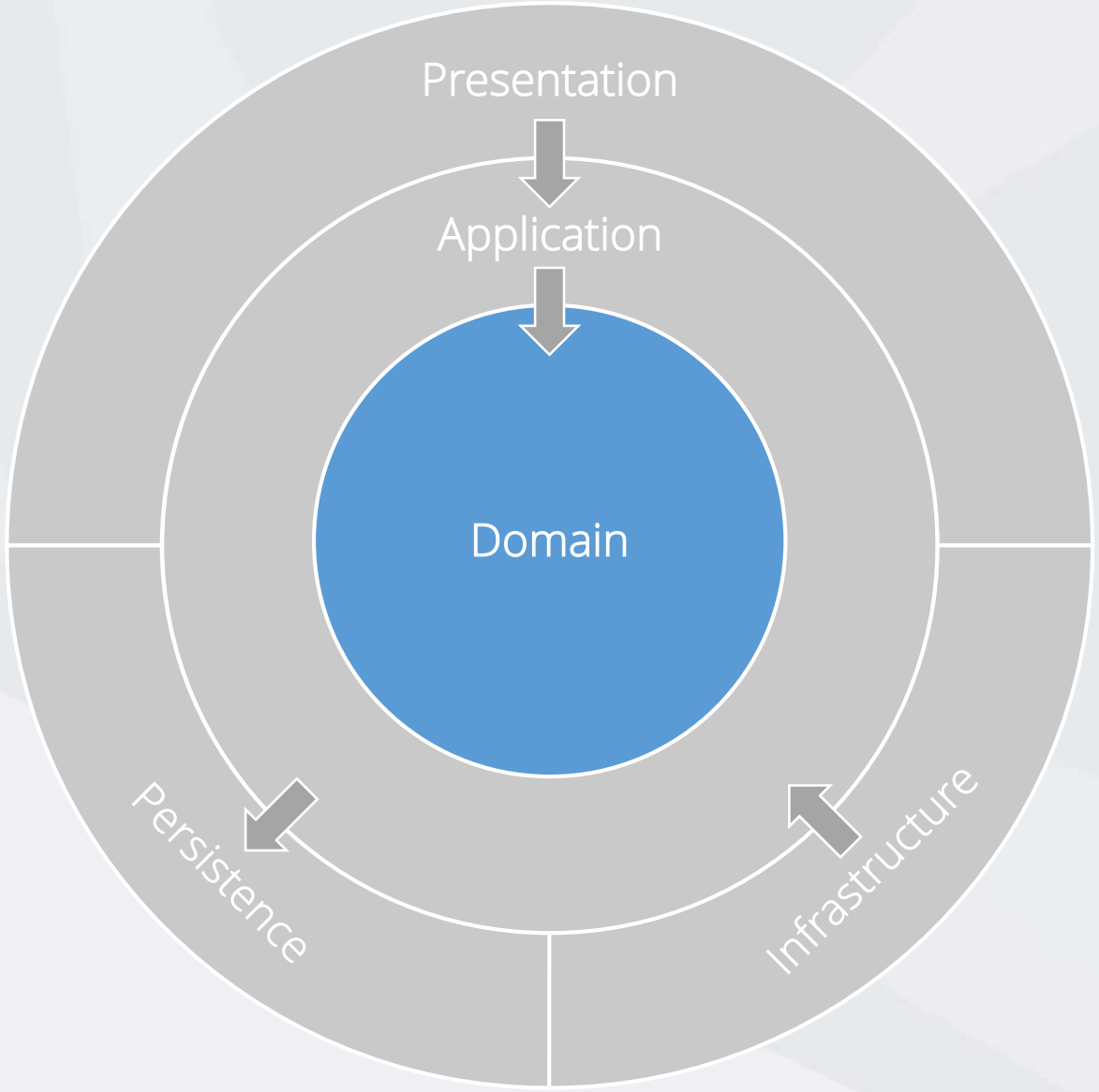
Entities

Value Objects

Enumerations

Logic

Exceptions



Demo

Reviewing the Domain layer

Key Points

Use data annotations sparingly

Always define foreign keys

Use value objects when appropriate

Initialise all collections

Create custom domain exceptions

Application Layer

Overview

Interfaces

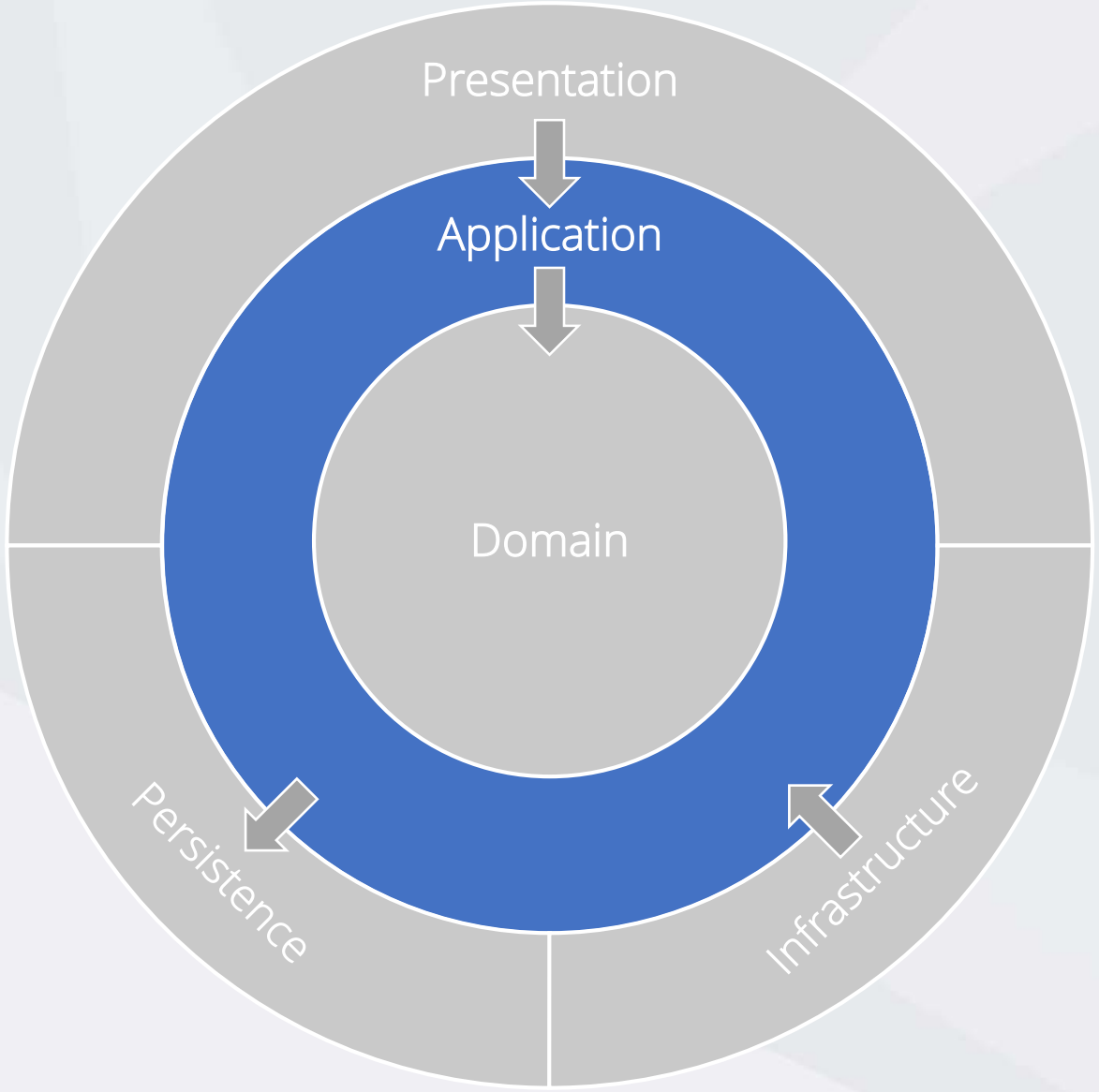
Models

Logic

Validators

Events

Exceptions



Demo

Reviewing the Application layer

Key Points

Use CQRS to simply your overall design

Use MediatR to simplify management of requests / responses, commands, queries, notifications and events

Know the difference between View Models (VMs) and Data Transfer Objects (DTOs)

Use FluentValidation for complex validation scenarios

Create custom application exceptions

Persistence Layer

Overview

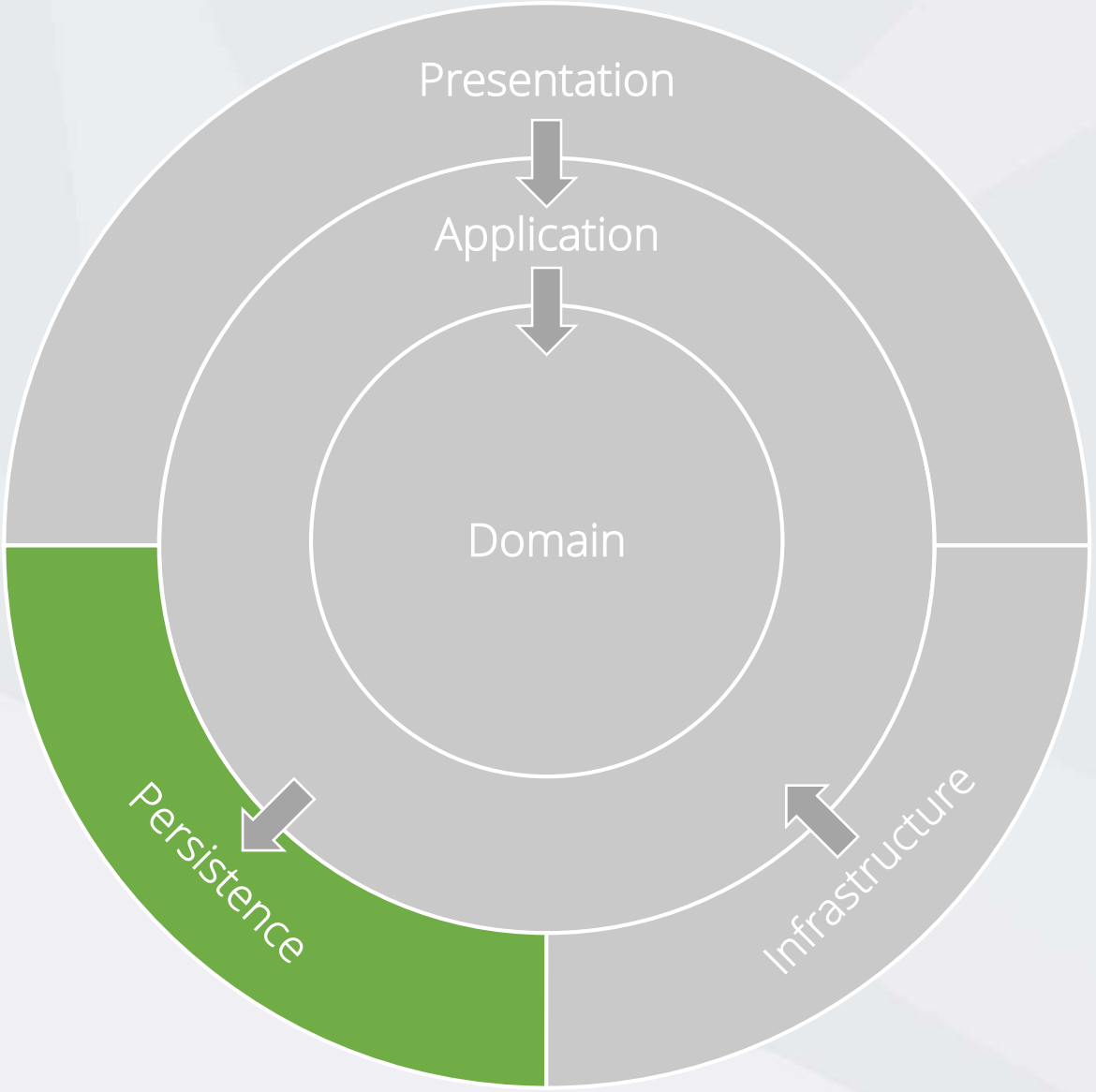
DbContext

Configurations

Migrations

Seeding

Abstractions



Demo

Reviewing the Persistence layer

Key Points

Independent of the database

Conventions over configuration

Use Fluent API Configuration over Data Annotations

Use an extension to automatically apply all entity type configurations

Infrastructure Layer

Overview

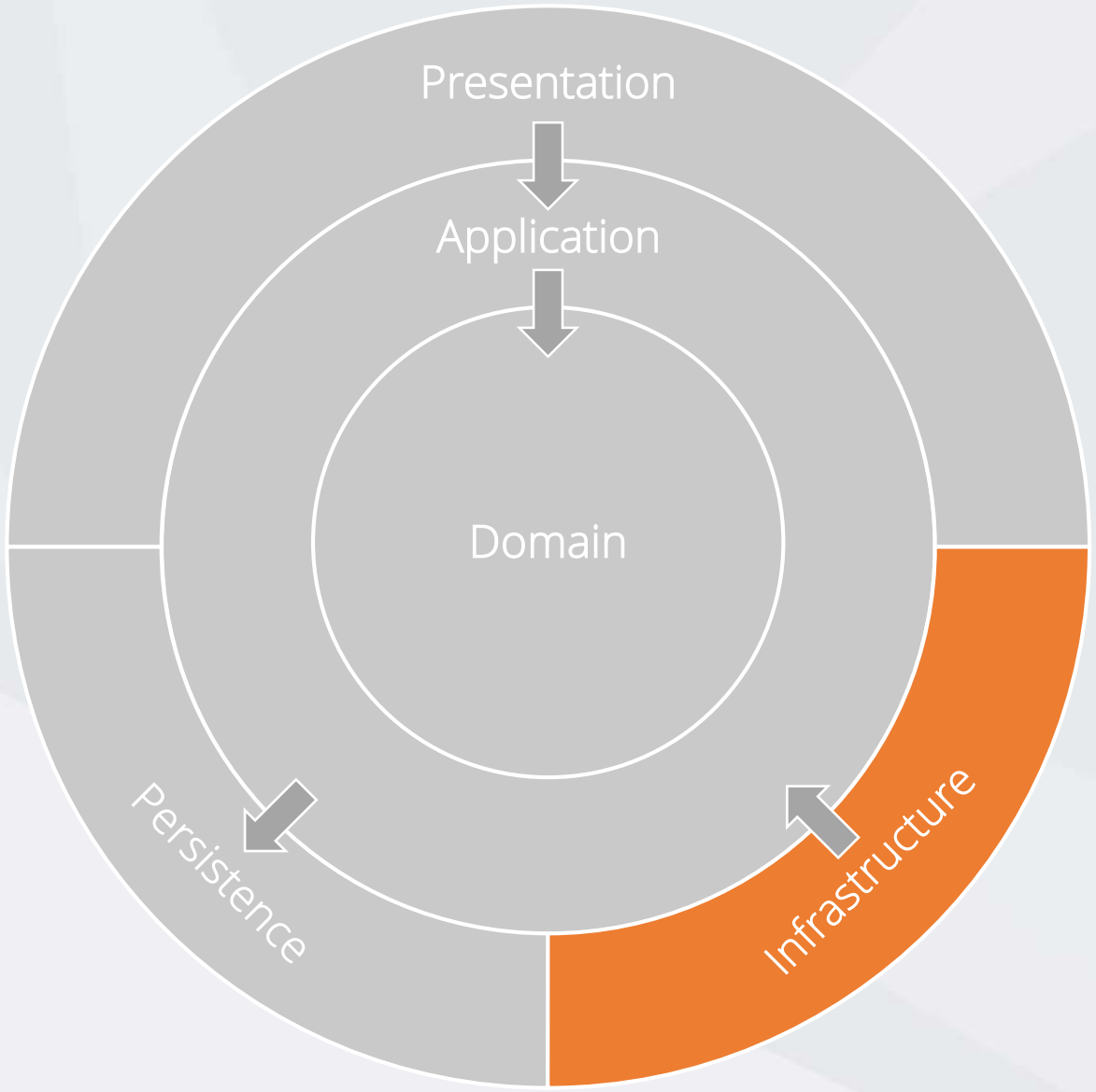
Clients

File Systems

Email / SMS

System Clock

Anything External



Demo

Reviewing the Infrastructure layer

Key Points

Contains classes for accessing external resources

Implements interfaces within the Application layer

Utilises DTOs defined within the Application layer

No layers depend on Infrastructure layer, e.g. Presentation layer

Presentation Layer

Overview

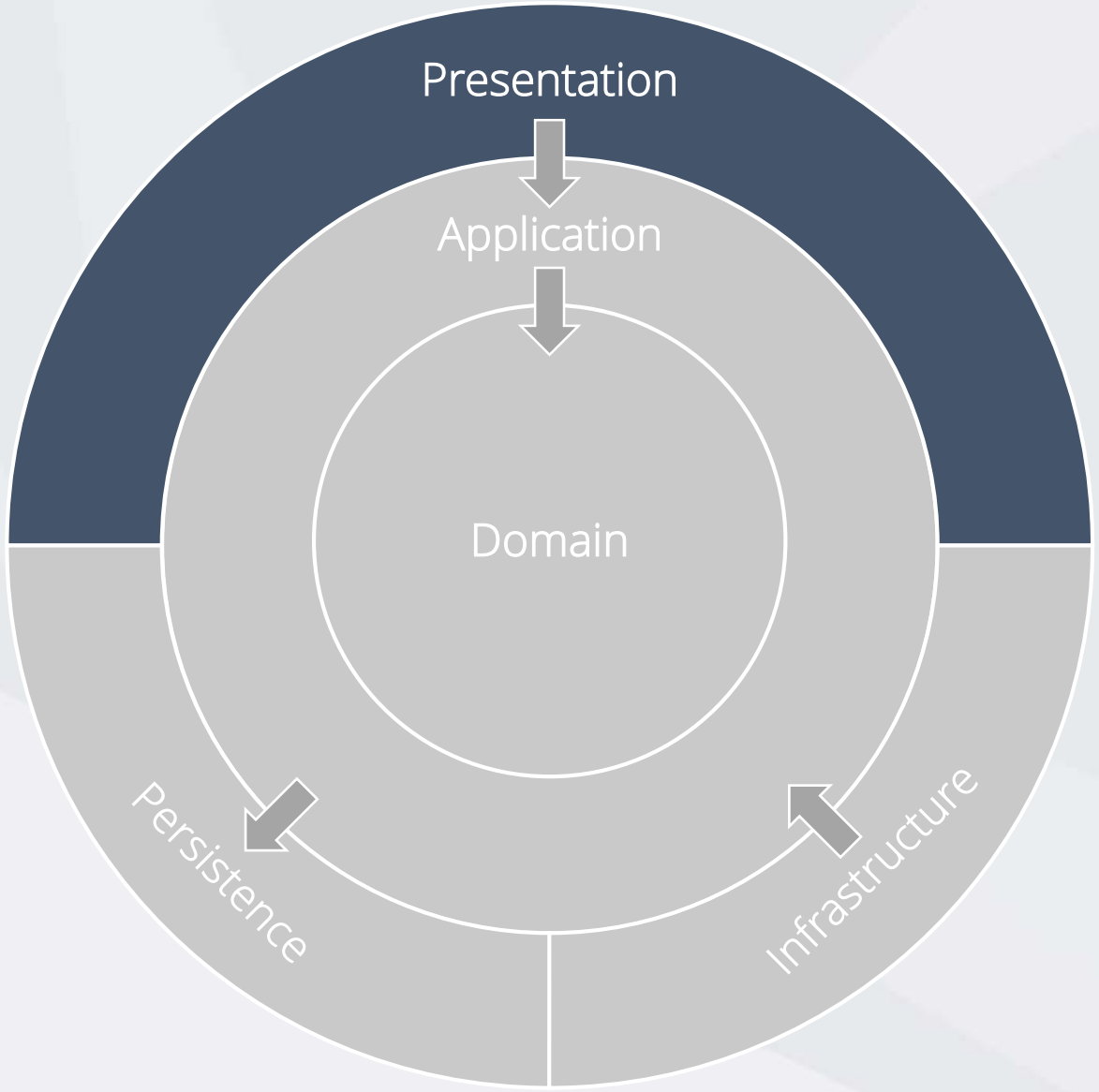
Web Forms

MVC

Razor Pages

Web API

Single Page Apps



Demo

Reviewing the Presentation layer

Key Points

Controllers should not contain any application logic

All application logic belongs in the application layer

Utilising Open API bridges the gap between the frontend and backend

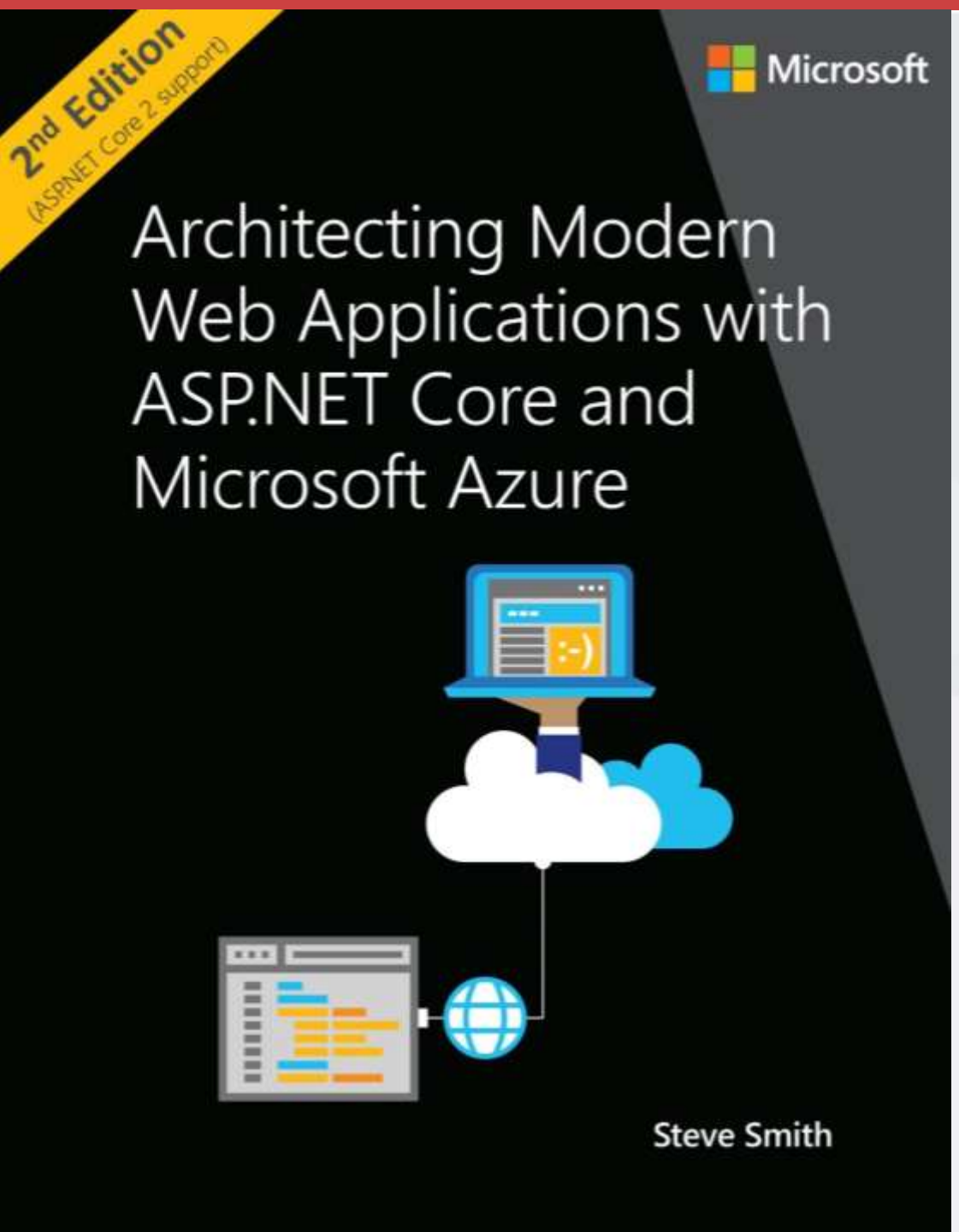
Create and consume well defined view models

Recommend Resources

Northwind Traders

A sample application built using ASP.NET Core and EF Core.

bit.ly/northwind-traders



Building Monoliths

Clean Architecture

Azure

aka.ms/webappebook

Building Microservices

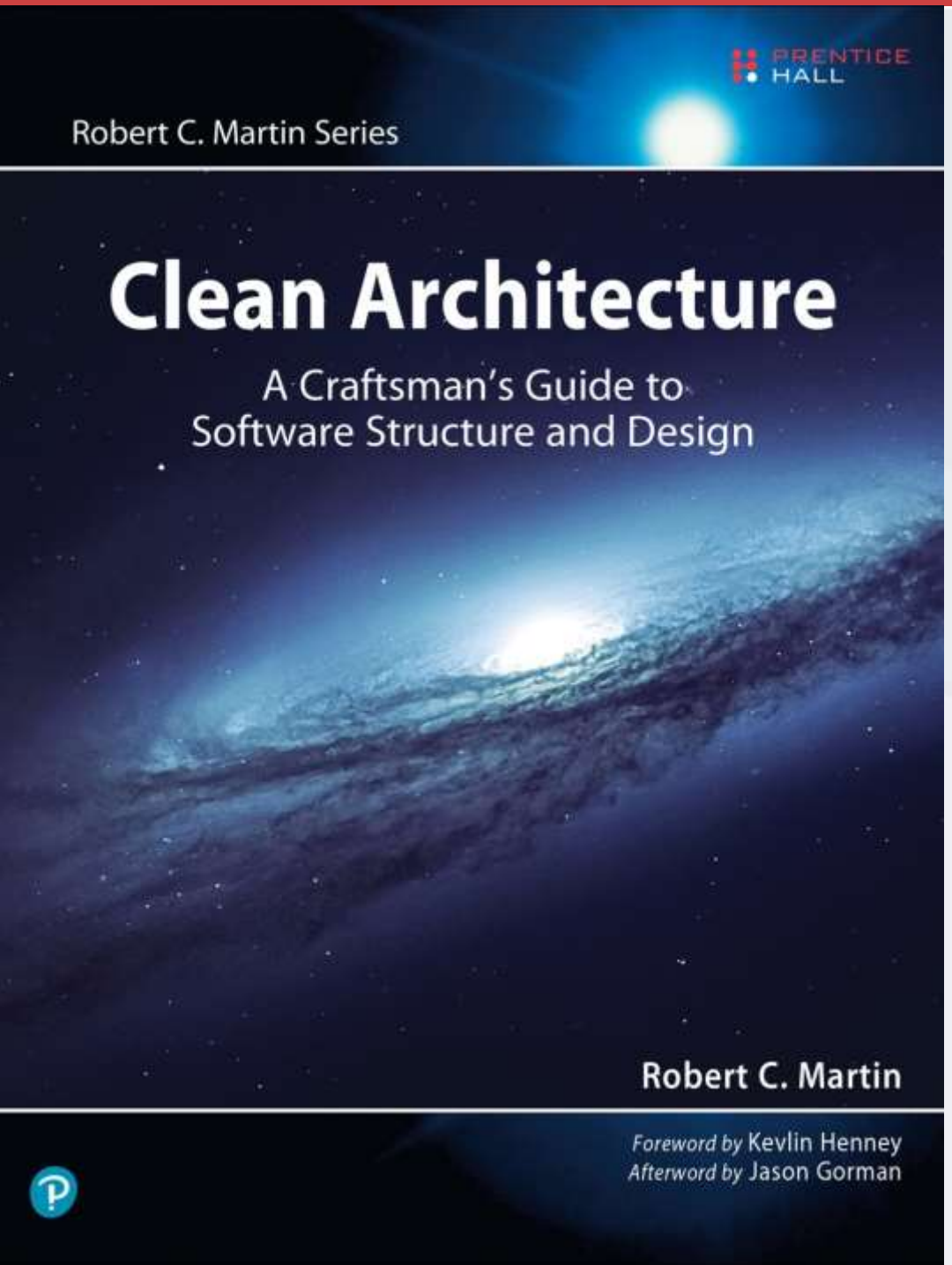
Microservices

Containers

DDD

Azure

aka.ms/microservicesebook



Clean Architecture

Robert C. Martin

bit.ly/clean-architecture-book

Questions?

Thank you!

info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane