SSW

Enterprise Software Development

Join the Conversation #CleanArchitecture @JasonGtAu

Clean Architecture

with ASP.NET Core 2.2

December 2018

Jason Taylor

SSW Solution Architect

Started programming with BASIC on C64,

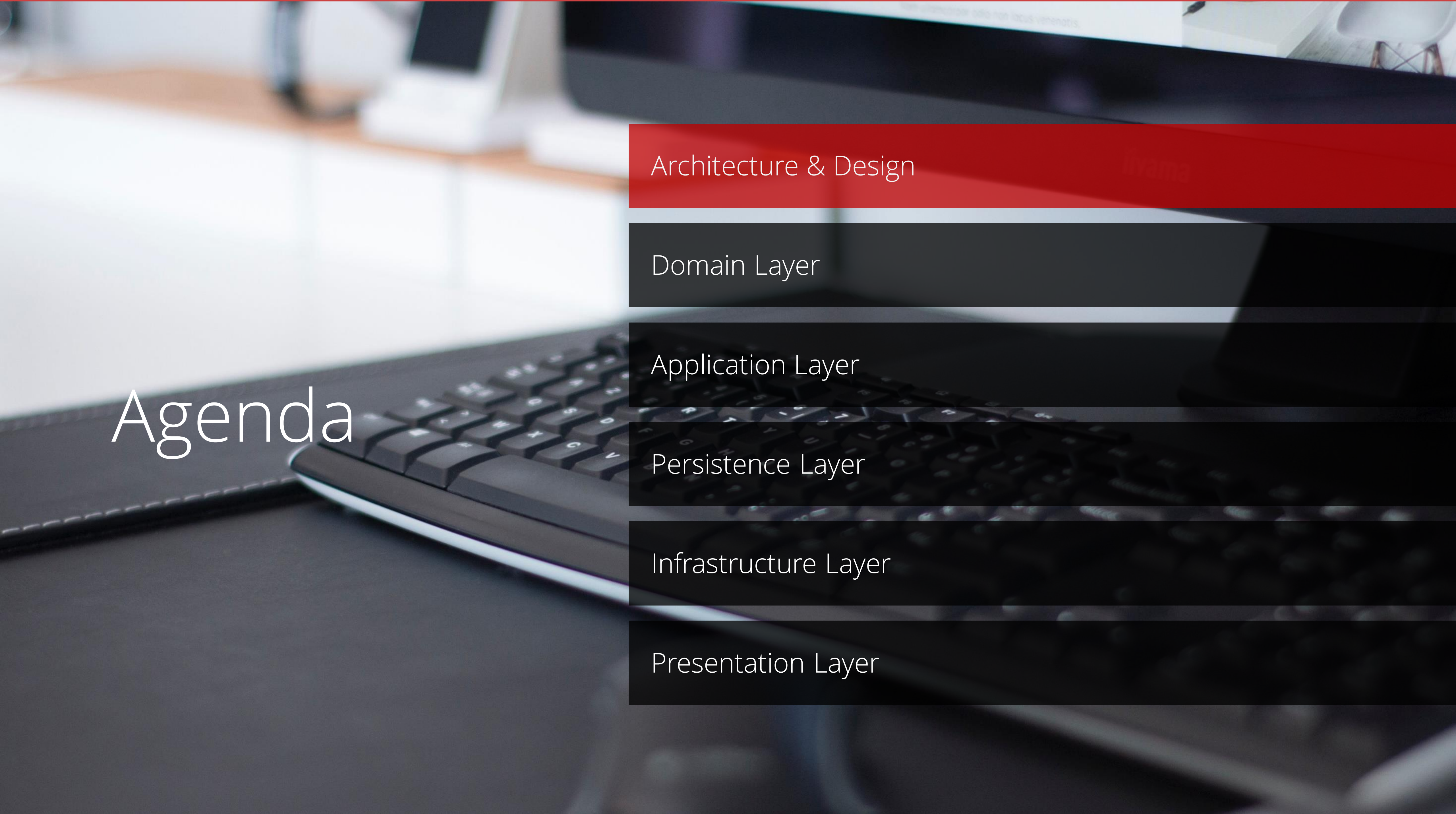
Keeping it simple since 1994!

jasongtau

codingflow.net

github.com/jasongt

youtube.com/jasongt



Agenda

Overview

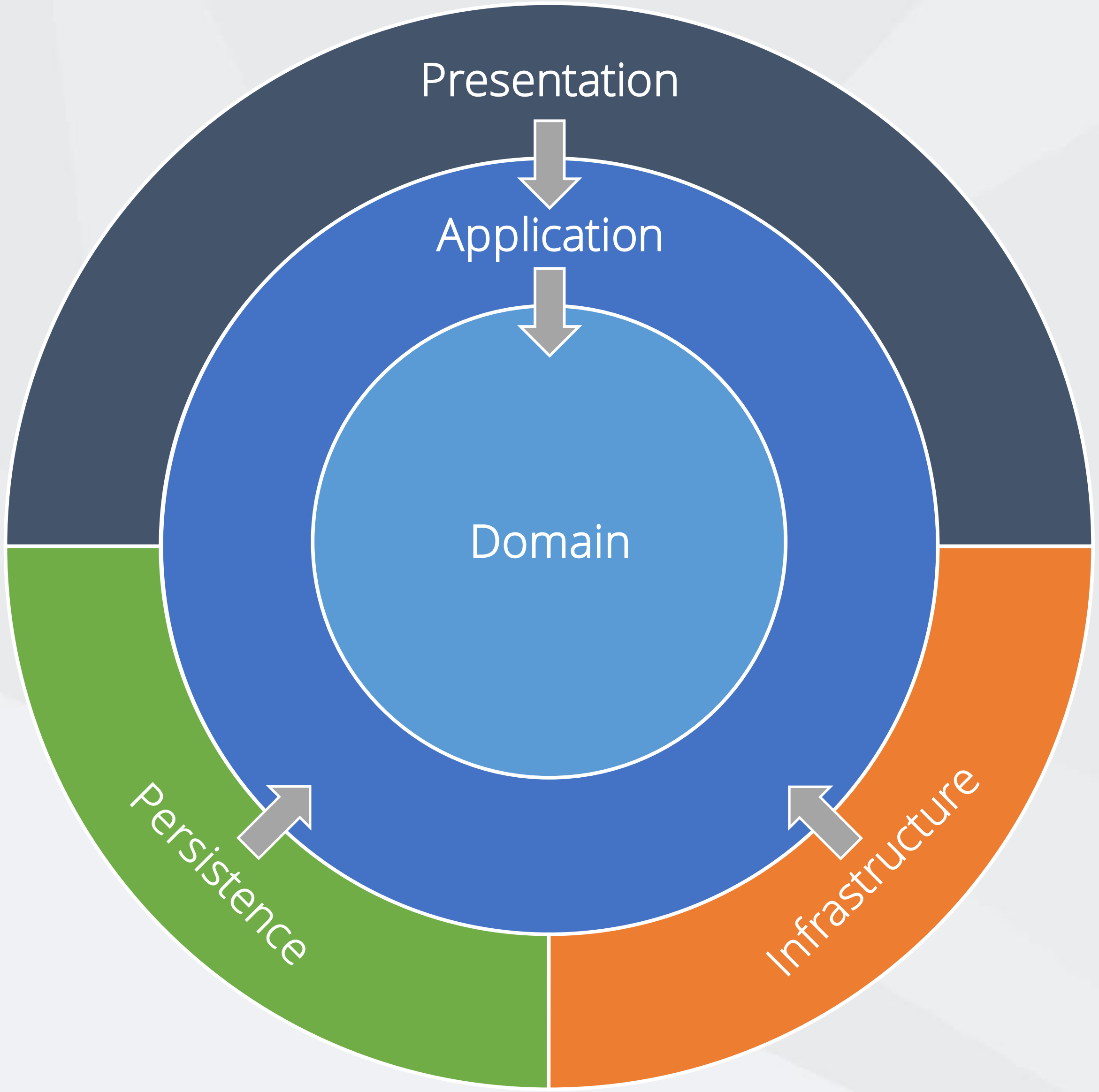
Independent of frameworks

Testable

Independent of UI

Independent of database

Independent anything external



Northwind Traders

Cross Platform

.NET Core

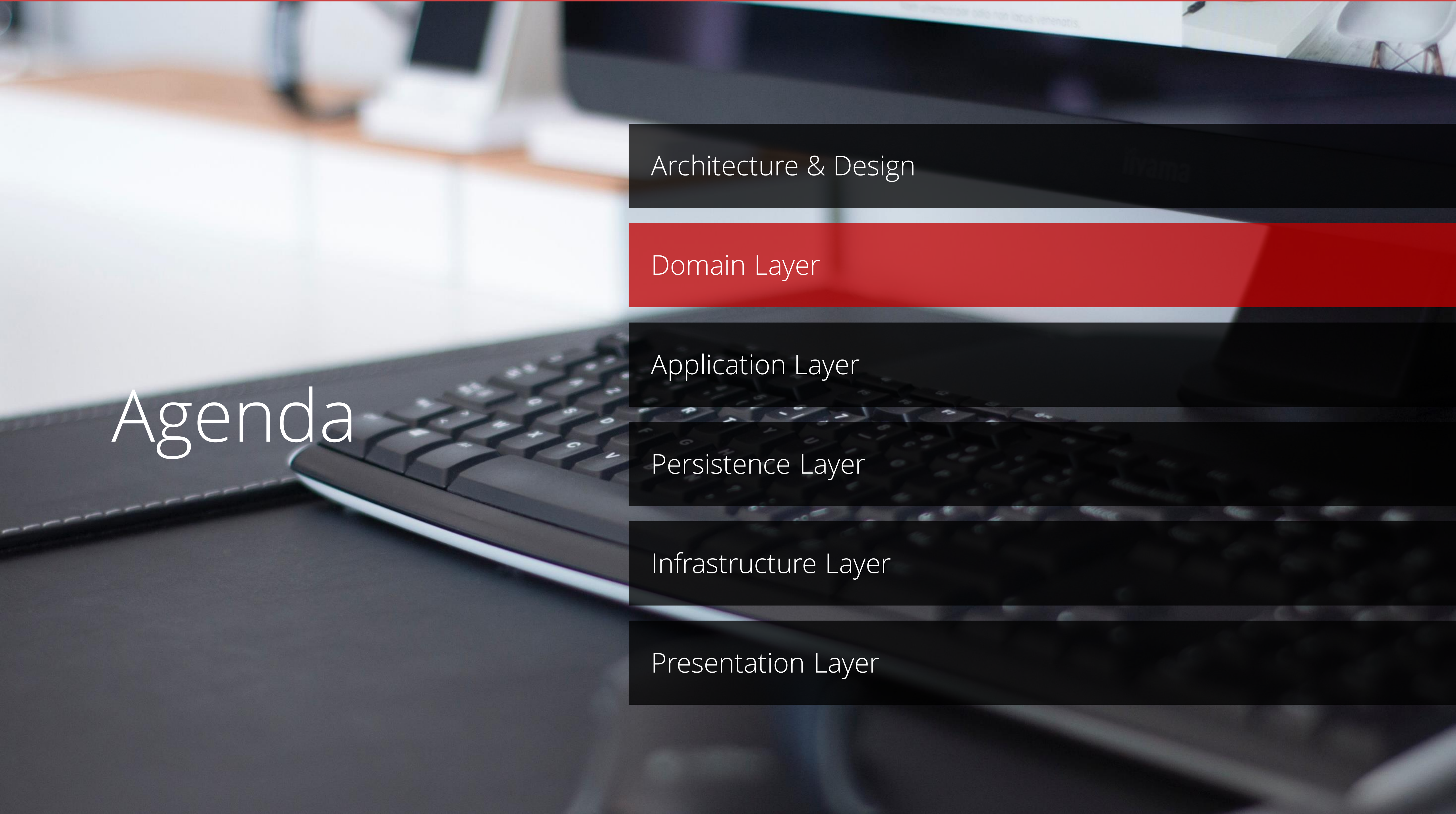
Entity Framework Core

Code First

Data Seeding

Key Points

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



Agenda

Overview

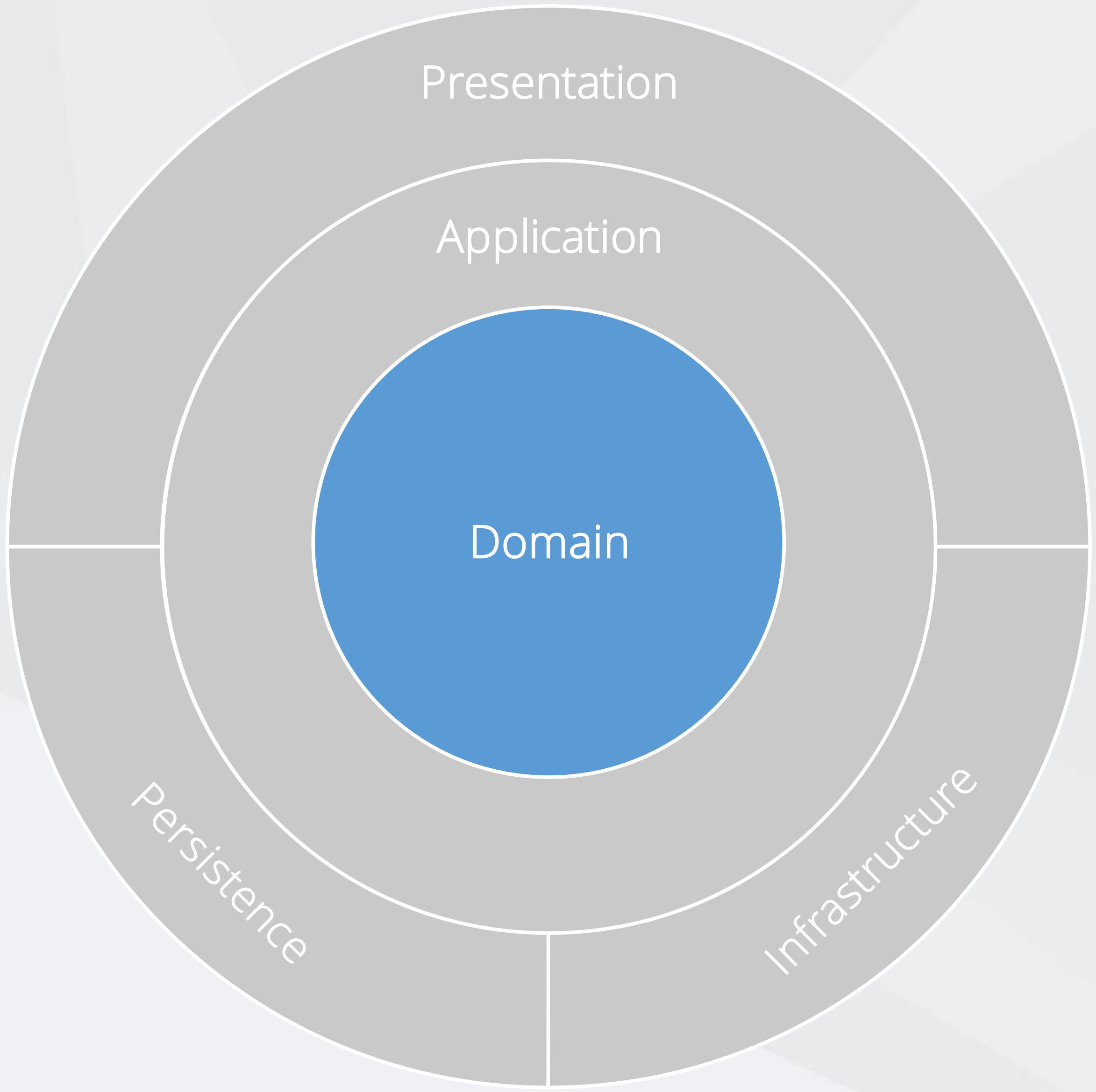
Entities

Value Objects

Enumerations

Logic

Exceptions

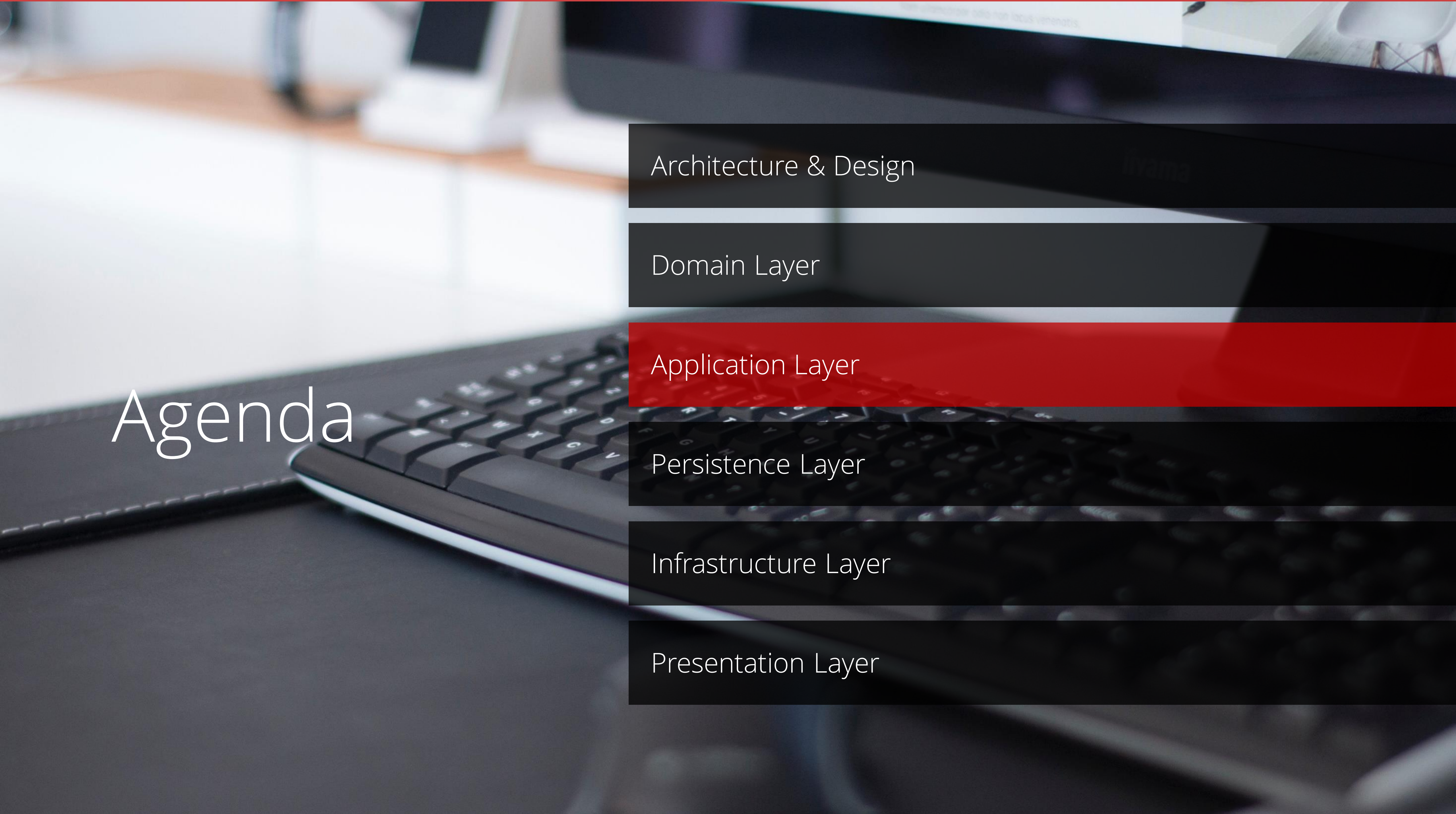


Demo

Reviewing the Domain layer

Key Points

- ✓ Avoid using data annotations
- ✓ Use value objects where appropriate
- ✓ Initialise all collections & use private setters
- ✓ Create custom domain exceptions



Agenda

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

Overview

Interfaces

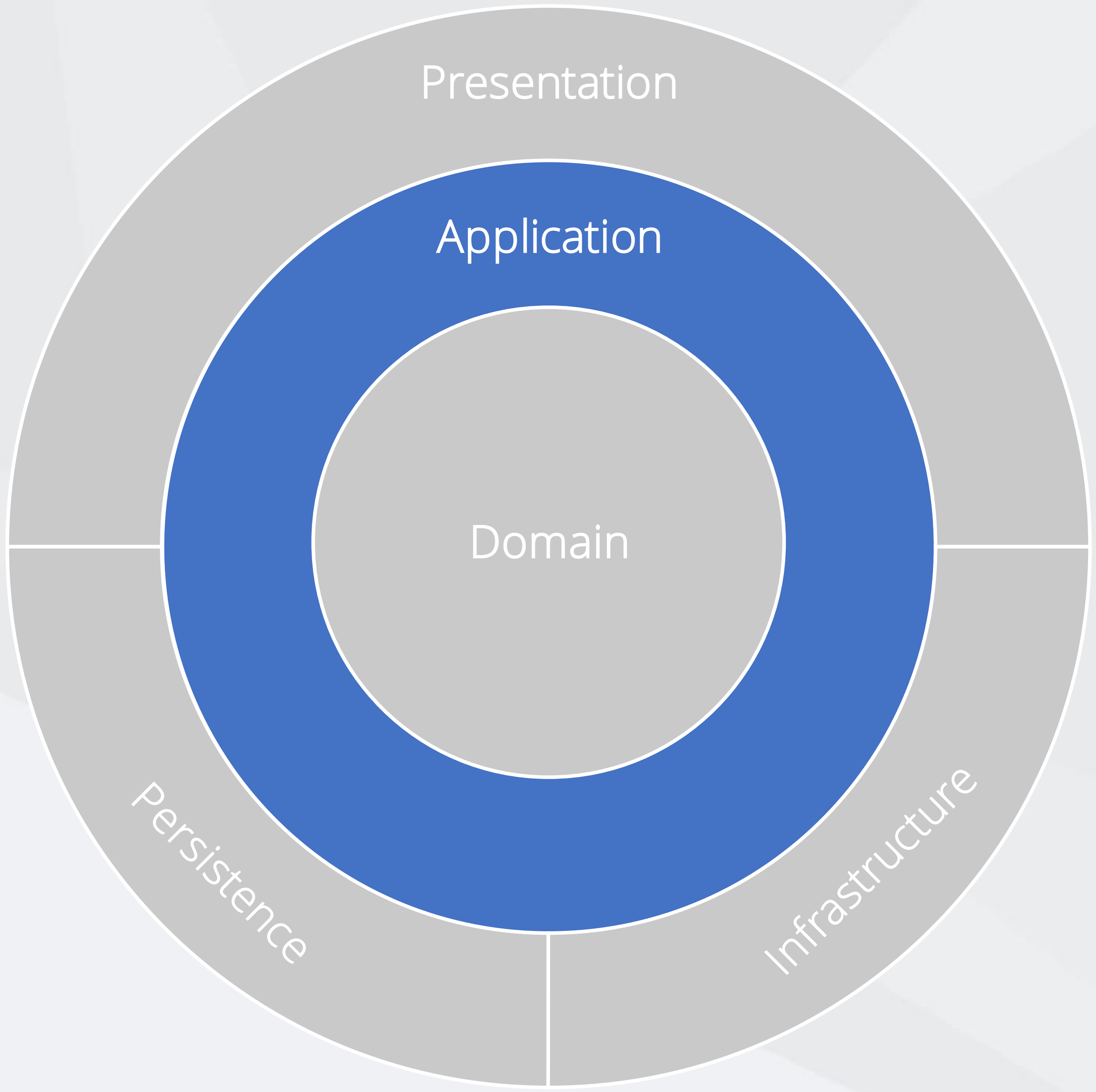
Models

Logic

Commands / Queries

Validators

Exceptions



CQRS

Command Query Responsibility Segregation

Separate reads (queries) from writes (commands)

Can maximise performance, scalability, and simplicity

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

Key Points

- ✓ Using CQRS + MediatR simplifies your overall design
- ✓ Fluent Validation is useful for all validation scenarios
- ✓ MediatR simplifies cross cutting concerns
- ✓ Independent of infrastructure and data access concerns

Agenda

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

Overview

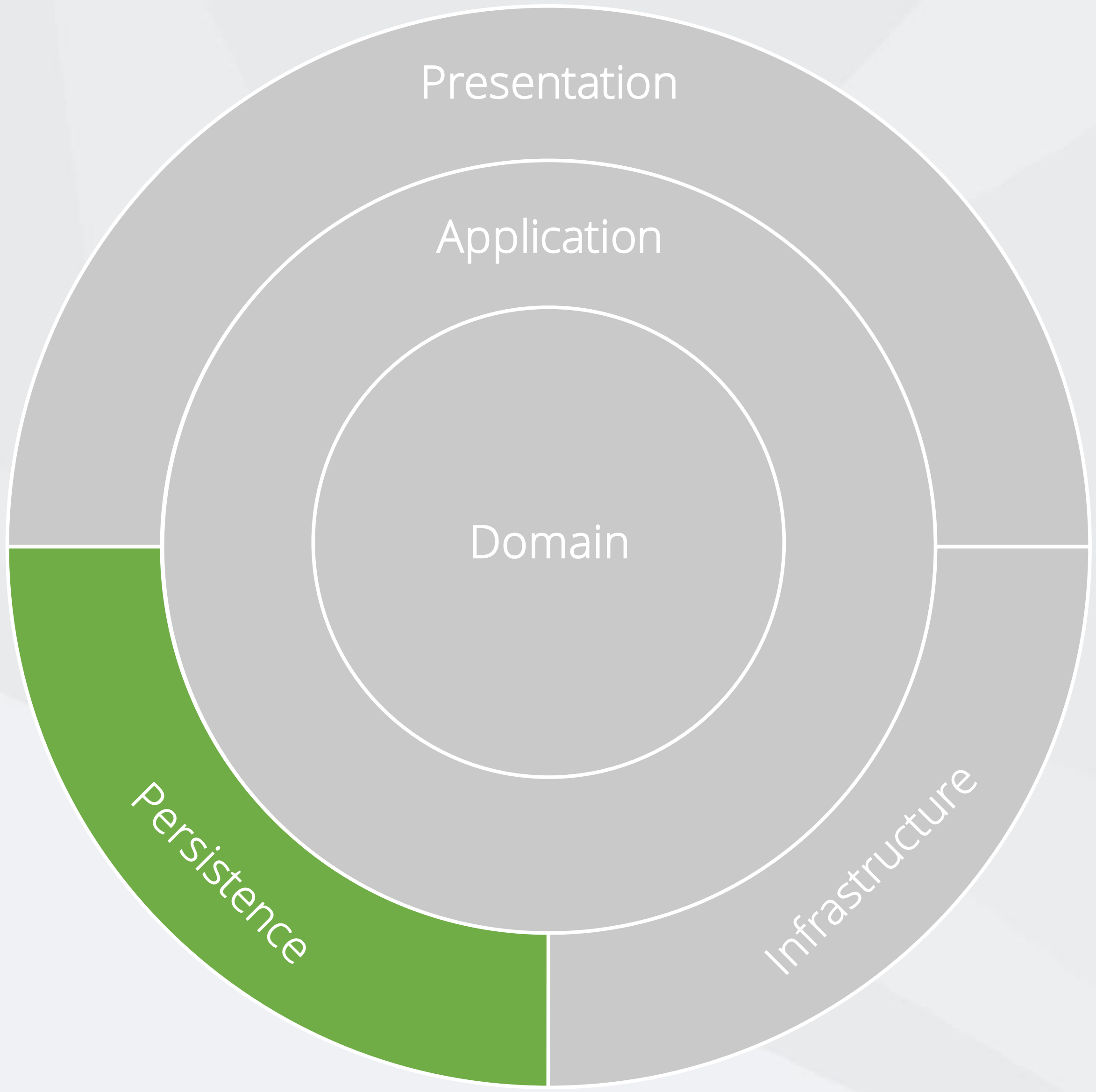
DbContext

Migrations

Configurations

Seeding

Abstractions



Unit of Work and Repository Patterns

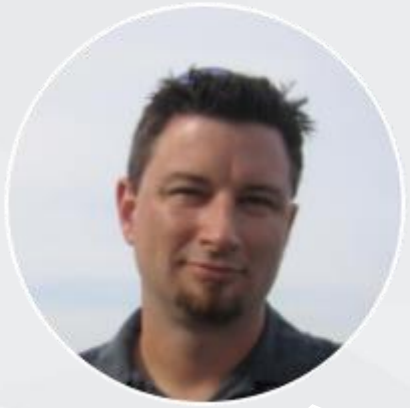
Should we implement these patterns?

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!

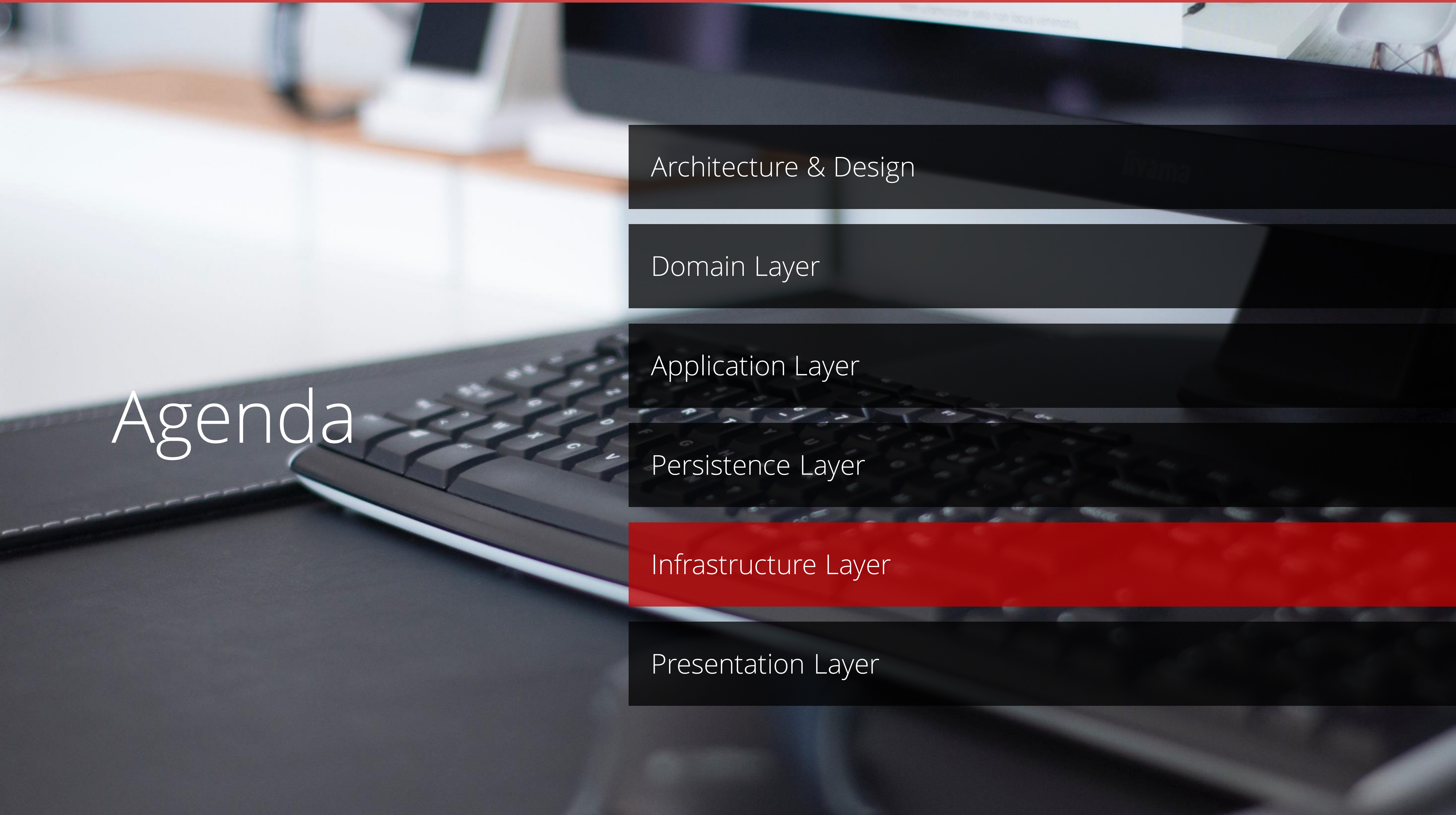
No, the repository/unit-of-work pattern isn’t useful with EF Core.

Demo

Reviewing the Persistence layer

Key Points

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



Agenda

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

Overview

Implementations, e.g.

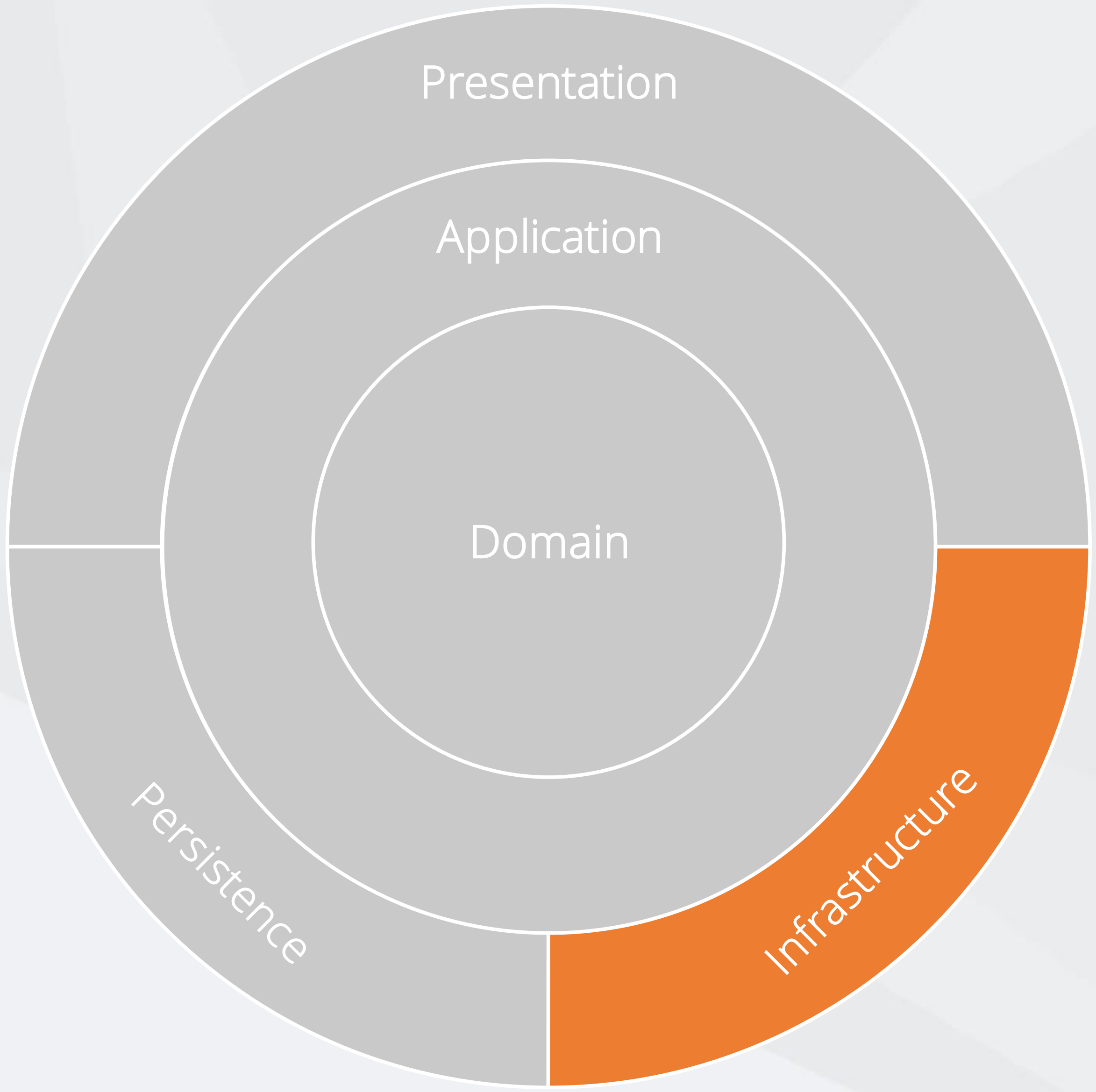
API Clients

File System

Email / SMS

System Clock

Anything external

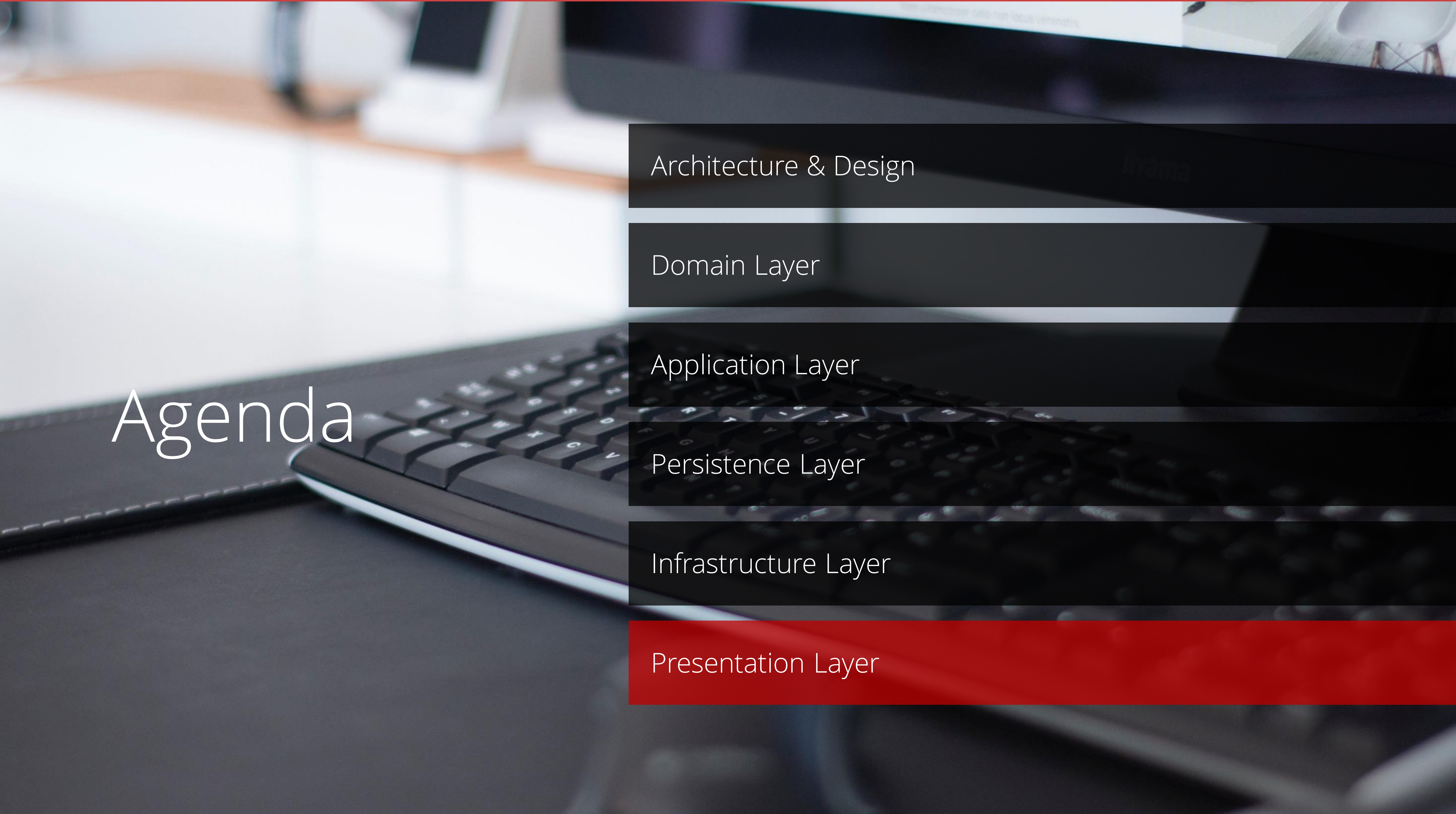


Demo

Reviewing the Infrastructure layer

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

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

Overview

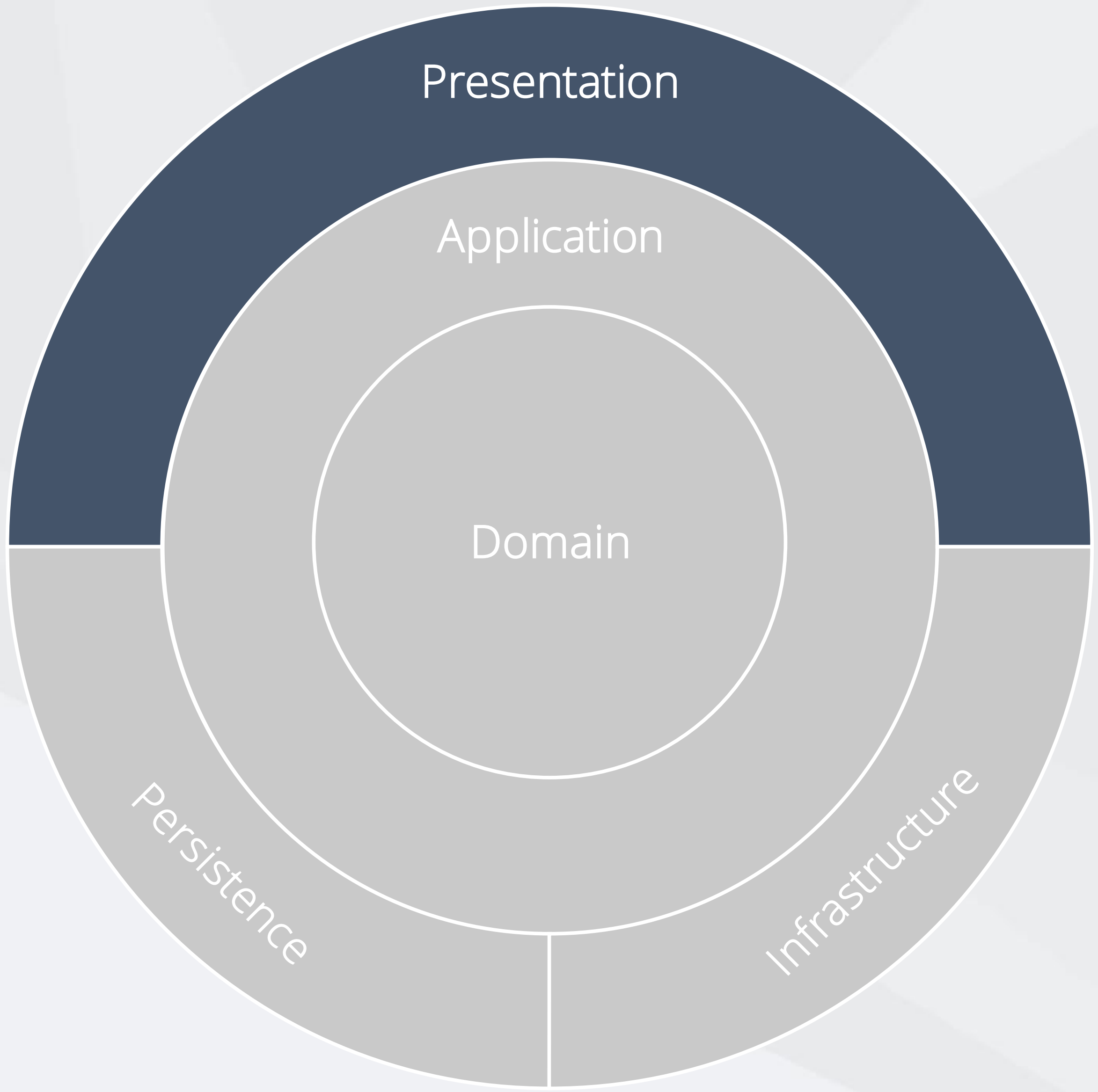
SPA – Angular or React

Web API

Razor Pages

MVC

Web Forms



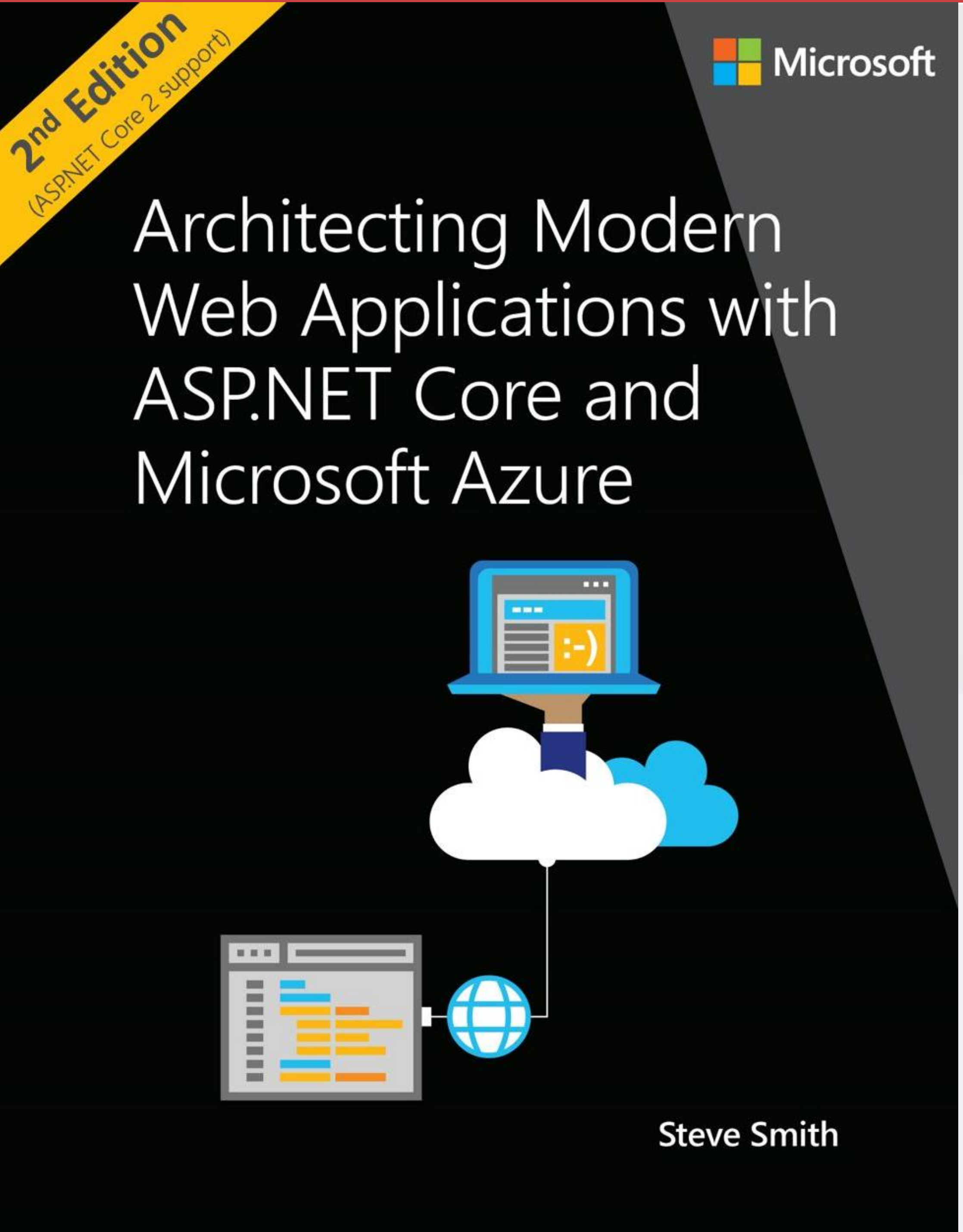
Demo

Reviewing the Presentation layer

Key Points

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

Recommend Resources



2nd Edition
(ASP.NET Core 2 support)

Microsoft

Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure

Steve Smith

Building Monoliths

Clean Architecture

Azure

v2.1 Edition

(.NET Core 2.1 support)

Microsoft

.NET Microservices: Architecture for Containerized .NET Applications

Cesar de la Torre
Bill Wagner
Mike Rousos

Microsoft Corporation

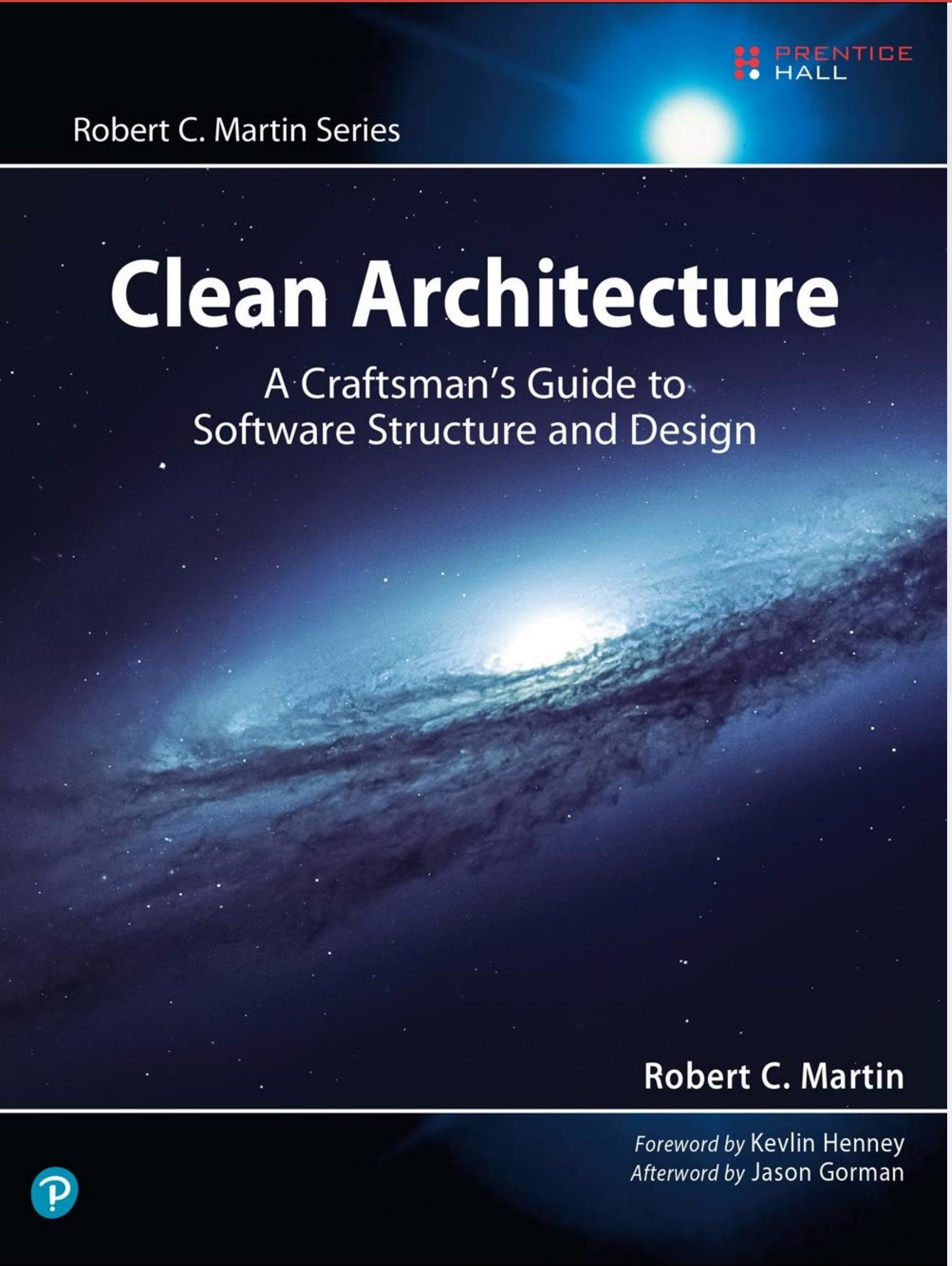
Building Microservices

Microservices

Containers

DDD

Azure


PRENTICE HALL
Robert C. Martin Series
Clean Architecture
A Craftsman's Guide to Software Structure and Design
Robert C. Martin
Foreword by Kevlin Henney
Afterword by Jason Gorman

Clean Architecture

Robert C. Martin

Join the Conversation #CleanArchitecture @JasonGtAu

Next Steps

Code & Slides

bit.ly/northwind-traders

Get Started

Thank you!

bit.ly/northwind-traders

@jasongtau

info@ssw.com.au

www.ssw.com.au

Sydney | Melbourne | Brisbane