UNIVEM DIGITAL

# .NET Platform

Especialização Full Stack Development

Prof. Dr. João Ricardo Favan

27 de Agosto de 2022

# Agenda

- Introdução ao .Net Platform
- Principais Recursos da .Net Platform
- Arquitetura da .Net Platform
- Design Patters para .Net
- Entity Framework
- Aplicação com persistência de dados

# Objetivo

- Conhecer a plataforma .NET e suas especificidades
- Conhecer o processo de desenvolvimento de uma aplicação web com a plataforma .NET
- Conhecer a ferramenta de persistência de dados Microsoft Entity Framework

# Plataforma .NET

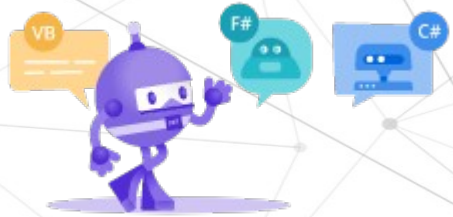
- É uma plataforma de desenvolvimento *free*, *open source* e *cross-platform*.
- Permite o desenvolvimento de diversos tipos de aplicações como web, mobile, desktop, games, IoT, etc
- Pode ser utilizada com multiplas linguagens, editores e bibliotecas

# Plataforma .NET - Linguagens

C# é uma linguagem de programação simples, moderna e orientada a objetos.

F# é uma linguagem de programação que facilita a escrita de código sucinto, robusto e de alto desempenho.

O Visual Basic é uma linguagem acessível com uma sintaxe simples para criar aplicativos orientados a objetos.



# Plataforma .NET - Cross-platform

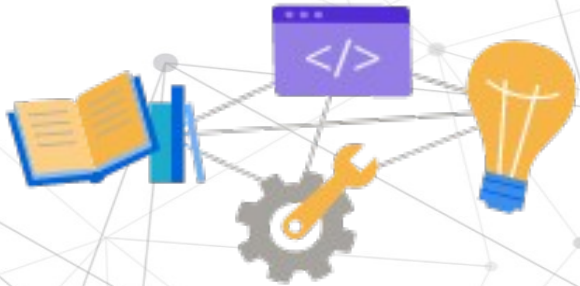
Trabalhando com as liguagens C#, F# ou Visual Basic, o código fonte será executado nativamente em qualquer sistema operacional.

Pode-se criar muitos tipos de aplicativos com, sendo alguns deles multiplataforma e outros visam um conjunto específico de sistemas operacionais e dispositivos.

# Plataforma .NET – API consistentes

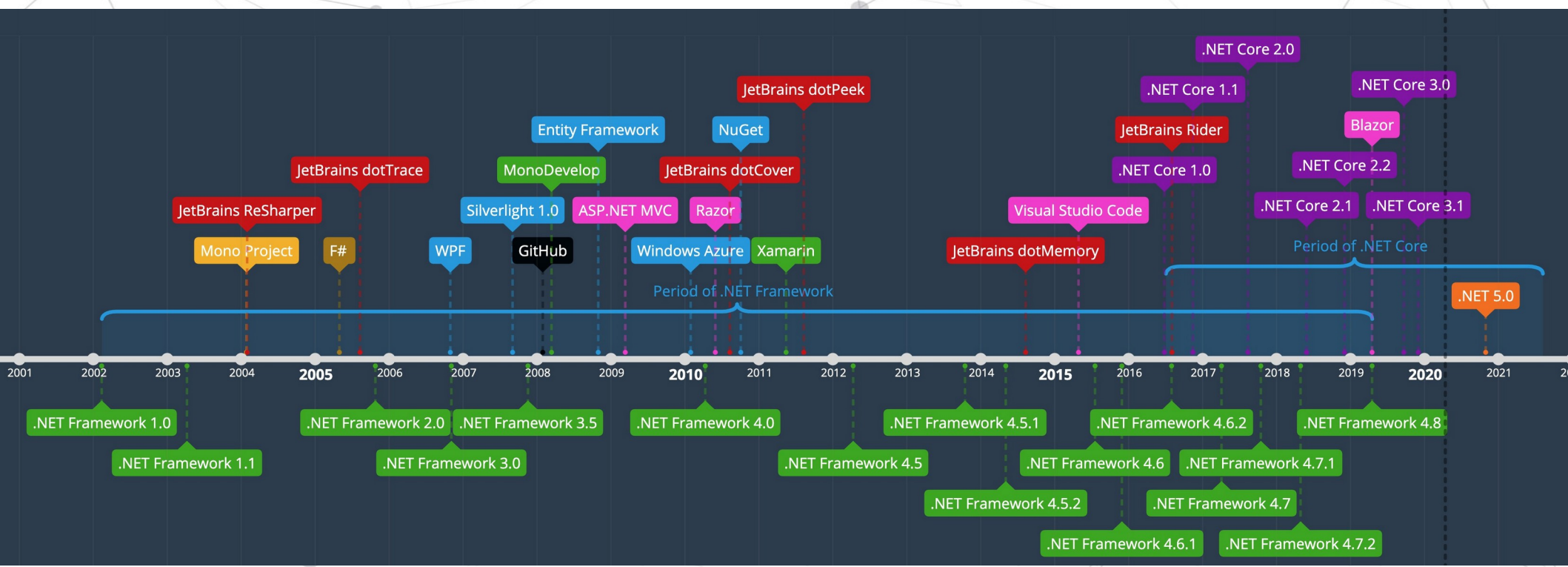
O .NET fornece um conjunto padrão de bibliotecas de classes básicas e APIs que são comuns a todos os aplicativos .NET.

Cada modelo de aplicativo pode utilizar APIs específicas para os sistemas operacionais em que é executado ou os recursos que ele fornece.

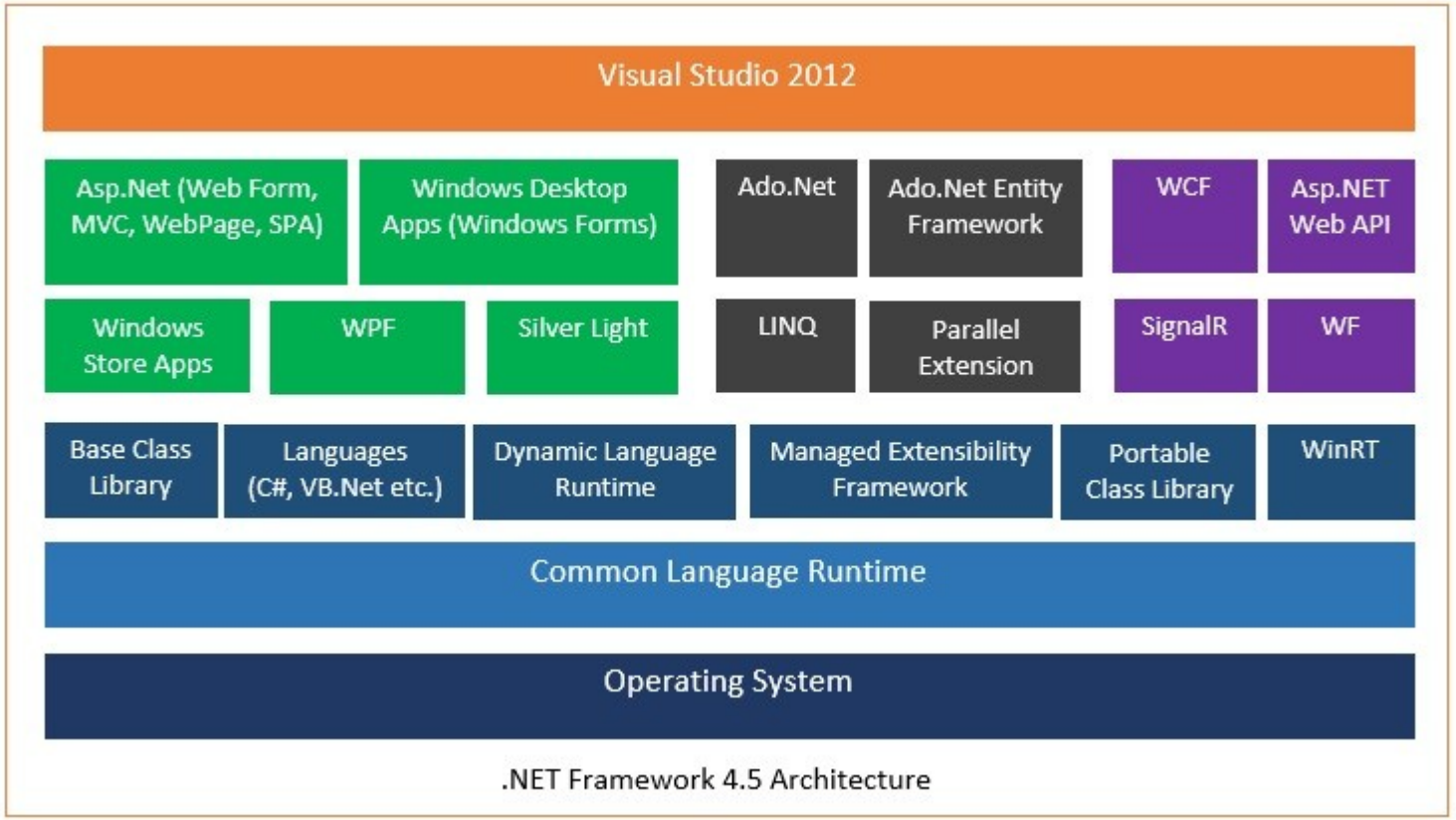


UNIVEM DIGITAL

# Plataforma .NET – Histórico

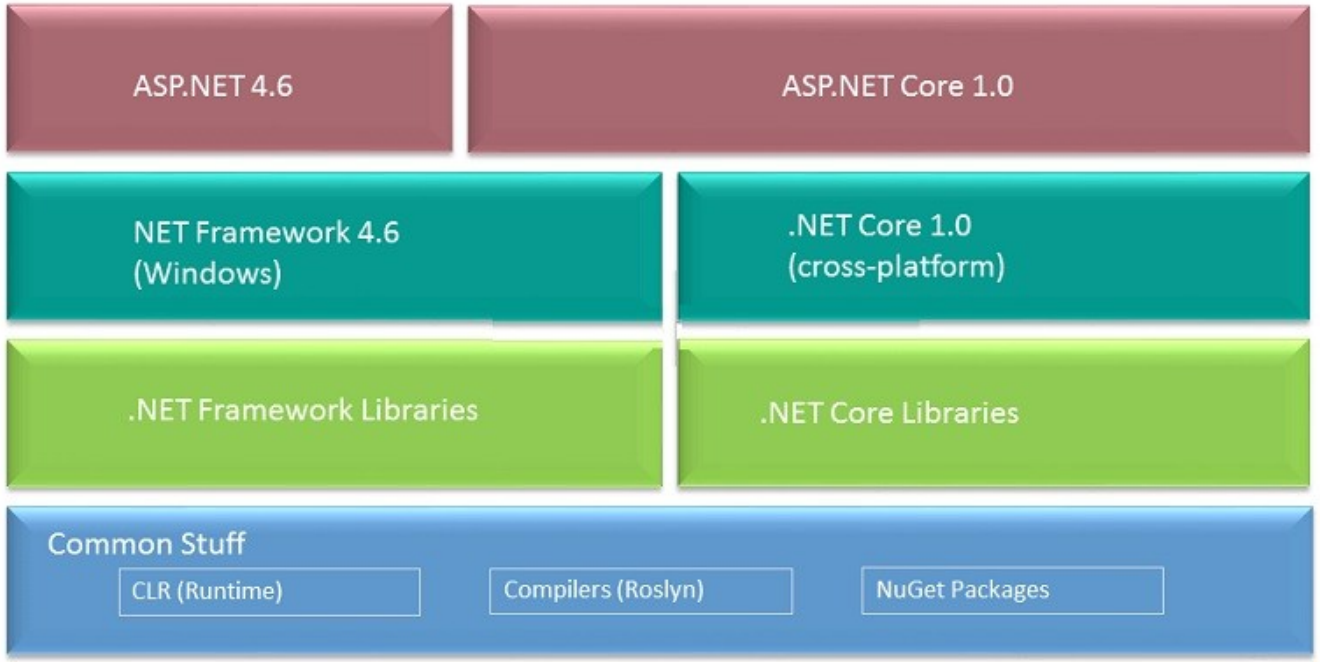


# .Net Framework



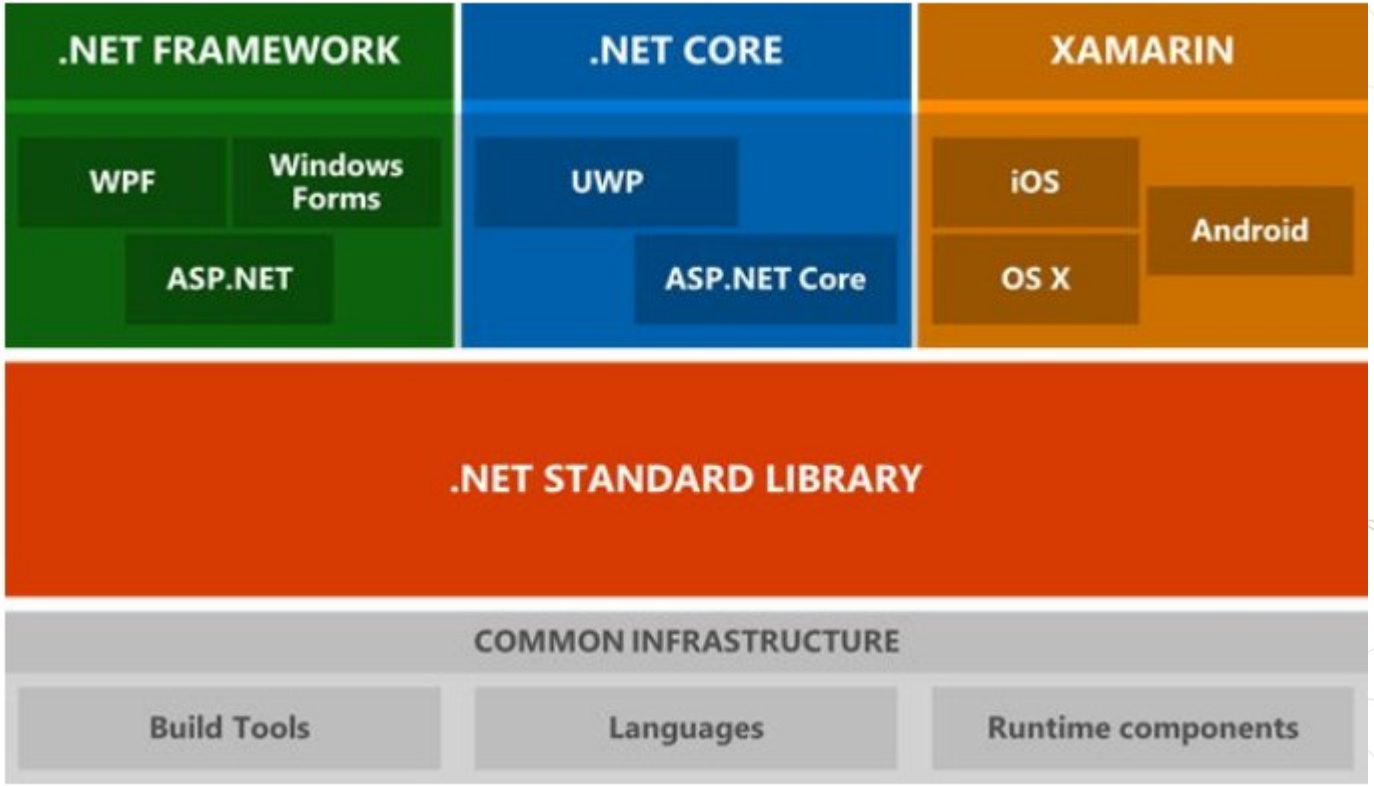

O .NET Framework oferecia tudo que o desenvolvedor precisava, mas ficava restrito ao ambiente Windows

# .Net Core




Adição ao .Net Framework que o tornava multiplataformas e atribuia outras características.

# .Net Core




Separação por seguimento

Manteve-se assim até 2020

UNIVEM DIGITAL

# .NET 5

# .NET 5.0

Unificação do ambiente .NET em uma única plataforma, permitindo que seja feita a migração de código e aplicativos

Objetiva criar uma Base Class Library (BCL) compartilhada que terá diferentes máquinas virtuais de tempo de execução

Várias Runtime VM, mas um único .NET

# Principios de Arquitetura para .NET

Vamos falar de alguns principios de design de software que são contemplados na arquitetura da plataforma .NET

# Principios de Arquitetura para .NET

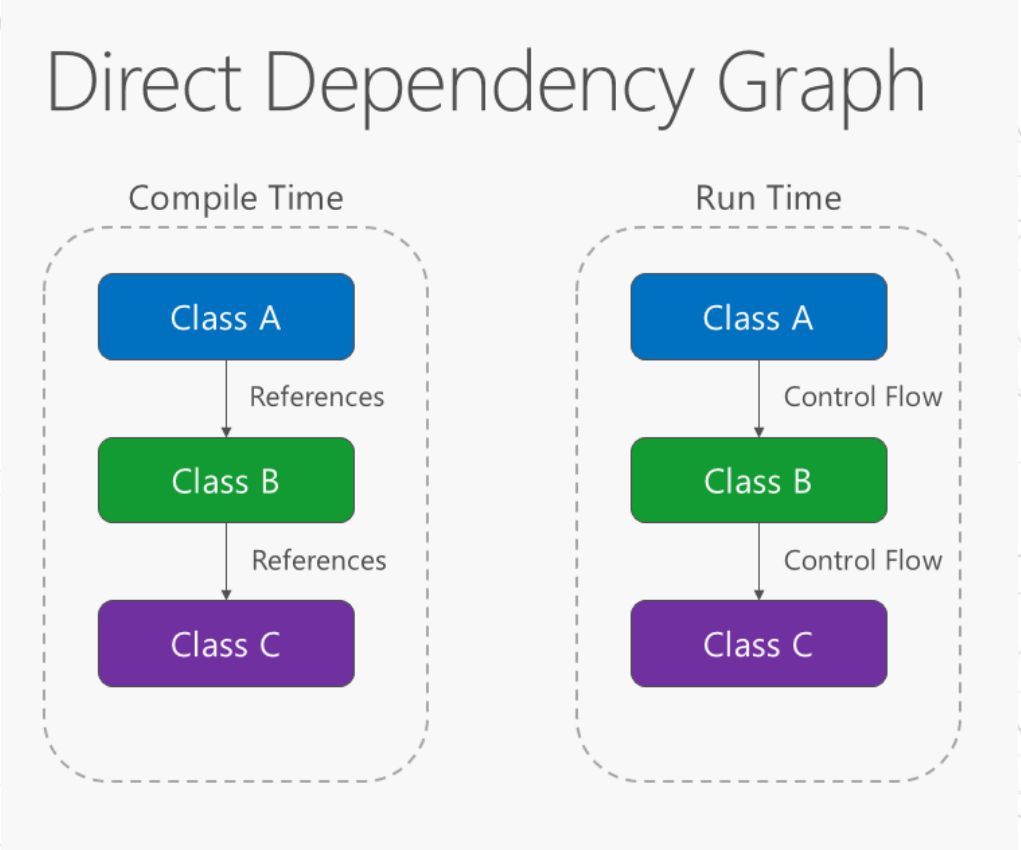
**Separação de Interesses:** O software deve ser separado de acordo com os tipos de trabalho que ele executa, ou seja, separando o comportamento principal dos negócios da infraestrutura e da lógica da interface do usuário.

**Encapsulamento:** isolamento de uma parte do aplicativo com outras partes do mesmo. Os componentes do aplicativo e os próprios aplicativos devem expor interfaces bem definidas para uso de seus colaboradores, em vez de permitir que seu estado seja modificado diretamente

# Principios de Arquitetura para .NET

**Inversão de Dependência:**

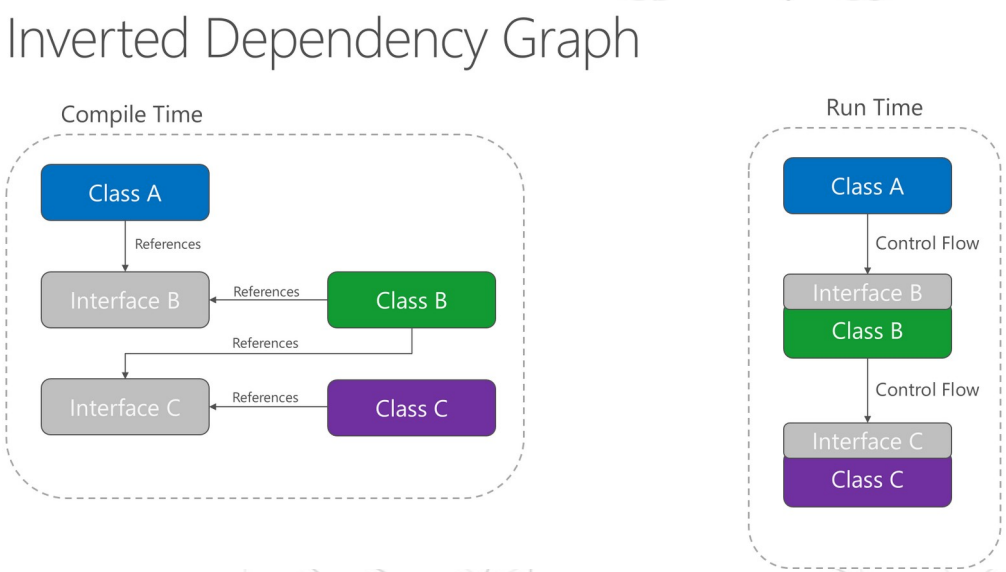
A direção da dependência dentro do aplicativo deve ser na direção de abstração, não os detalhes de implementação.



# Principios de Arquitetura para .NET

**Inversão de Dependência:**

A aplicação do princípio de inversão de dependência permite que A chame métodos em uma abstração que B implementa, tornando possível que A chame B em tempo de execução, mas para que B dependa de uma interface controlada por A em tempo de compilação



# Principios de Arquitetura para .NET

**Dependências Explicitas:** Métodos e classes devem exigir explicitamente os objetos de colaboração de que precisam para funcionarem corretamente. Pode-se utilizar os métodos construtores para tal.

**Responsabilidade Única:** Os objetos devem ter apenas uma responsabilidade e que devem ter apenas uma única razão para serem alterados.

**DRY (Don't Repeat Yourself)**: O aplicativo deve evitar especificar o comportamento relacionado a um conceito específico em vários locais

# Principios de Arquitetura para .NET

**Ignorância de Persistência:** refere-se aos tipos que precisam ser persistidos, mas cujo código não é afetado pela opção de tecnologia de persistência (POCOs (Objetos CRL Básicos)), pois não precisam herdar de uma classe base nem implementar uma interface específica

**Contexto Limitados:** Divide a aplicação em módulos conceituais separados. Cada módulo conceitual representa um contexto separado de outros contextos e pode evoluir de forma independente

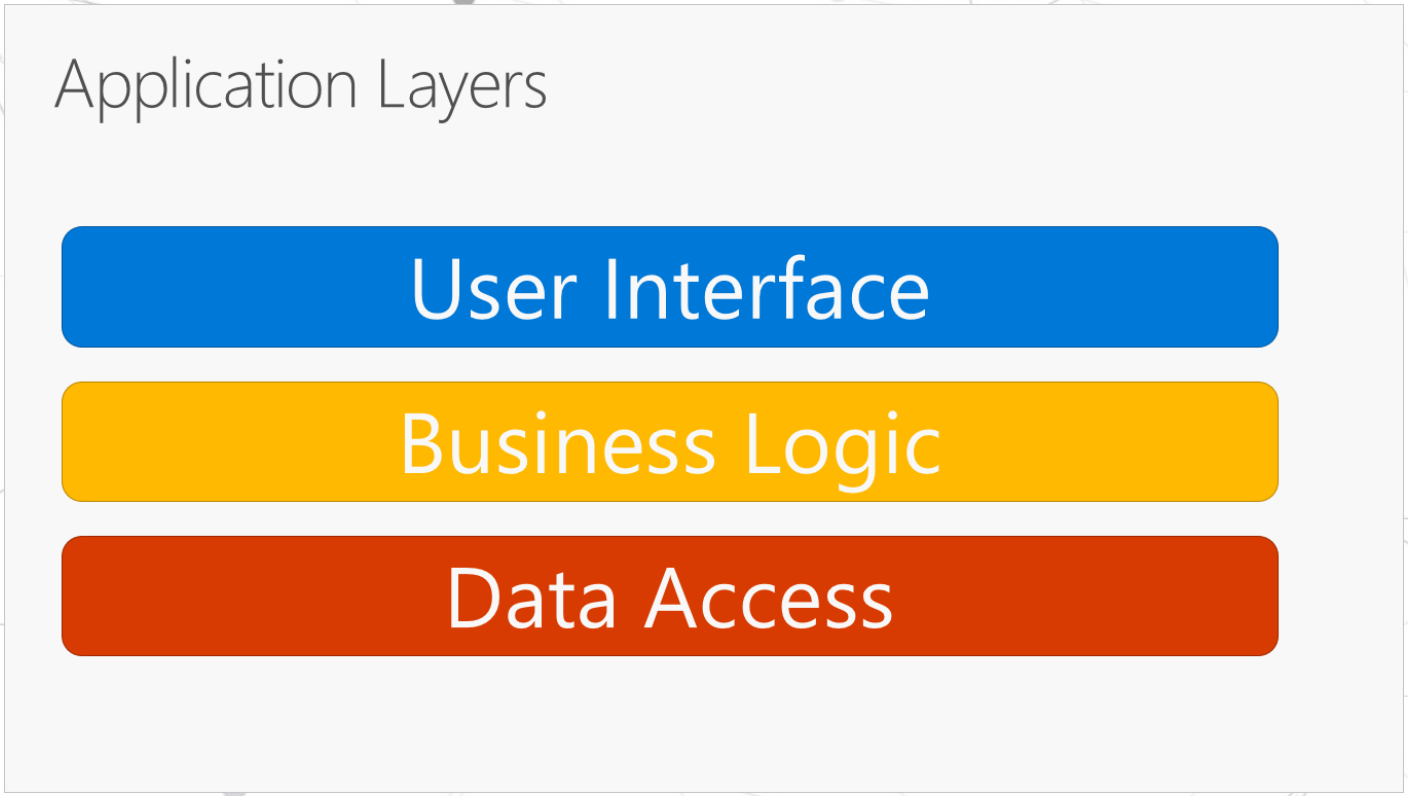
# Arquiteturas comuns para aplicações web com .NET

Arquiteturas N-Camadas

Arquitetura Limpa (onion)

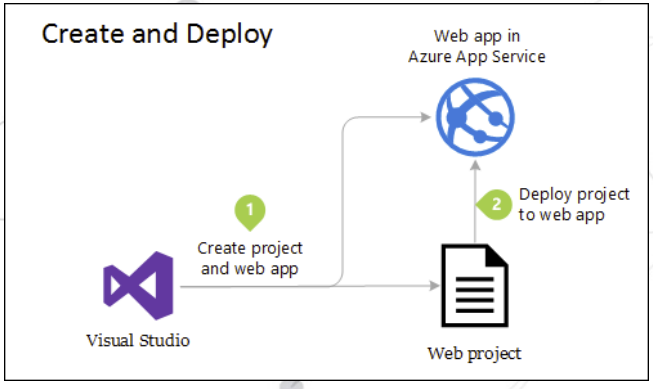
Camadas: Abordagem do principio da separação de interesses que consiste “dividir” a aplicação de acordo com suas responsabilidades. Essa abordagem pode ajudar a manter organizada uma base de código em expansão, de modo que os desenvolvedores possam encontrar com facilidade o local em que determinada funcionalidade foi implementada.

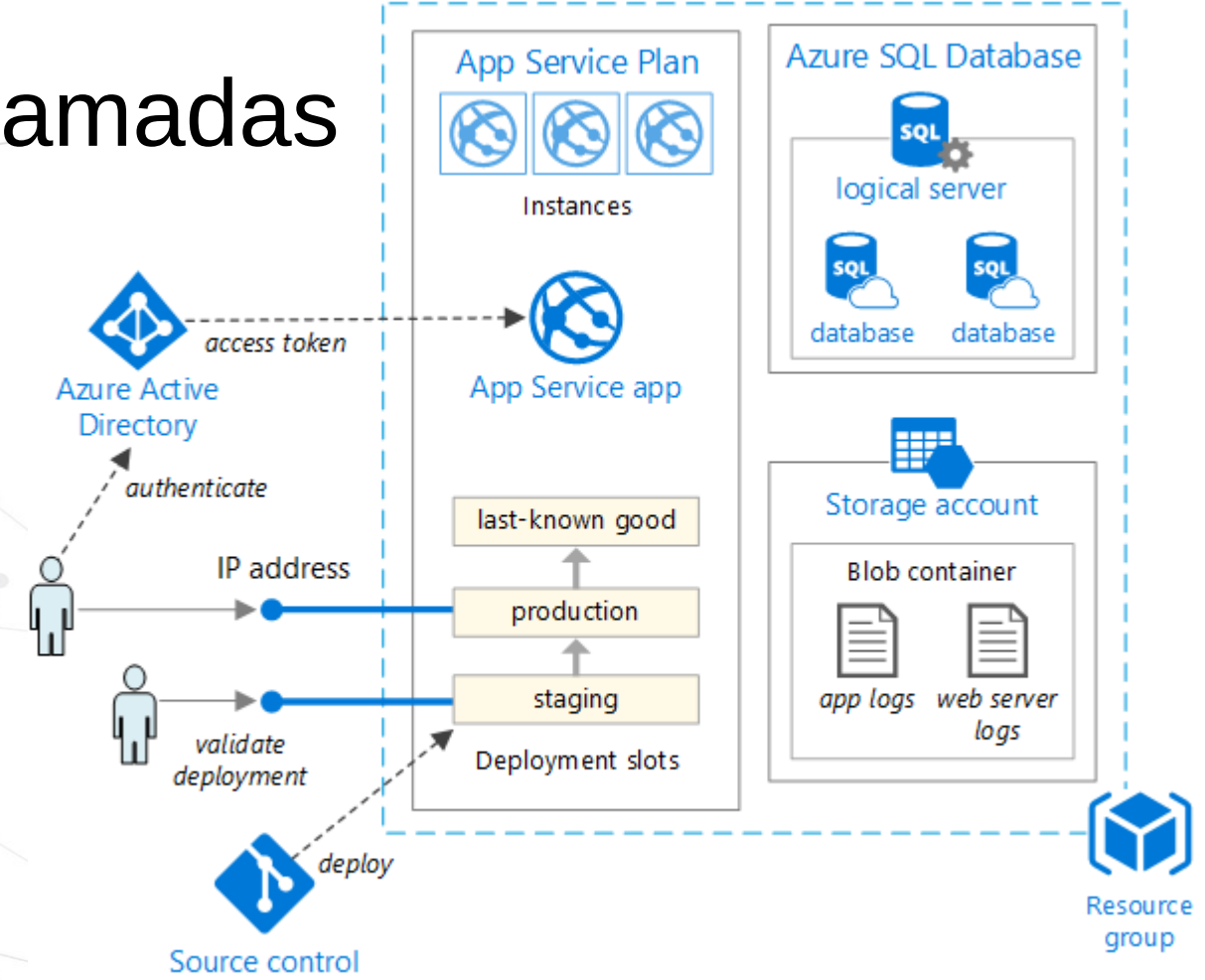
# Arquiteturas N-Camadas



UNIVEM DIGITAL

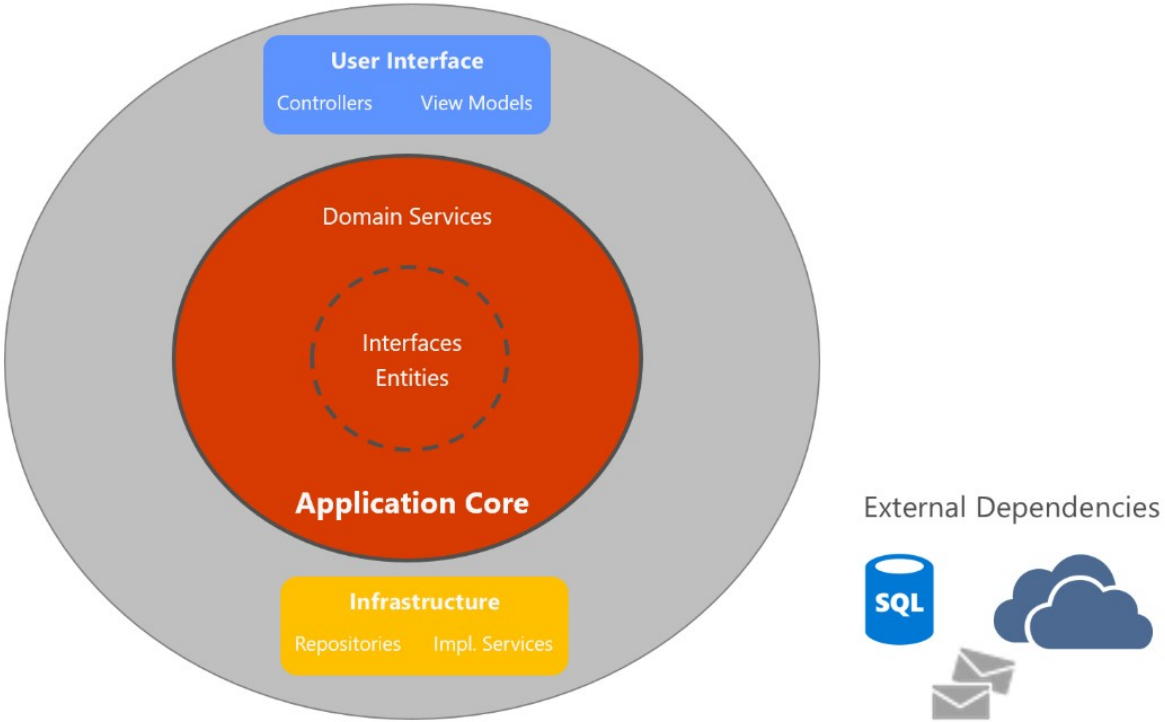
# Arquiteturas N-Camadas



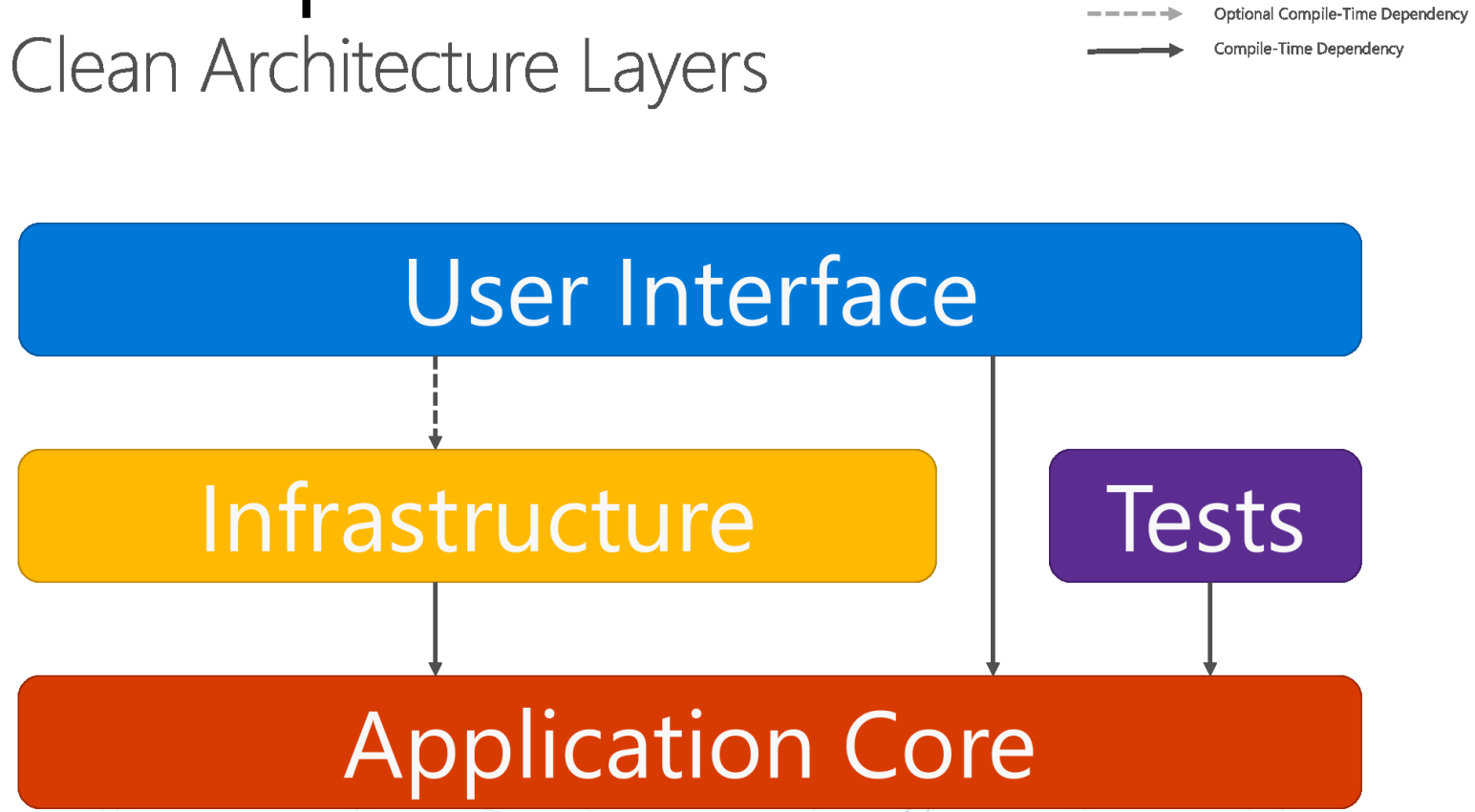



# Arquitetura Limpa

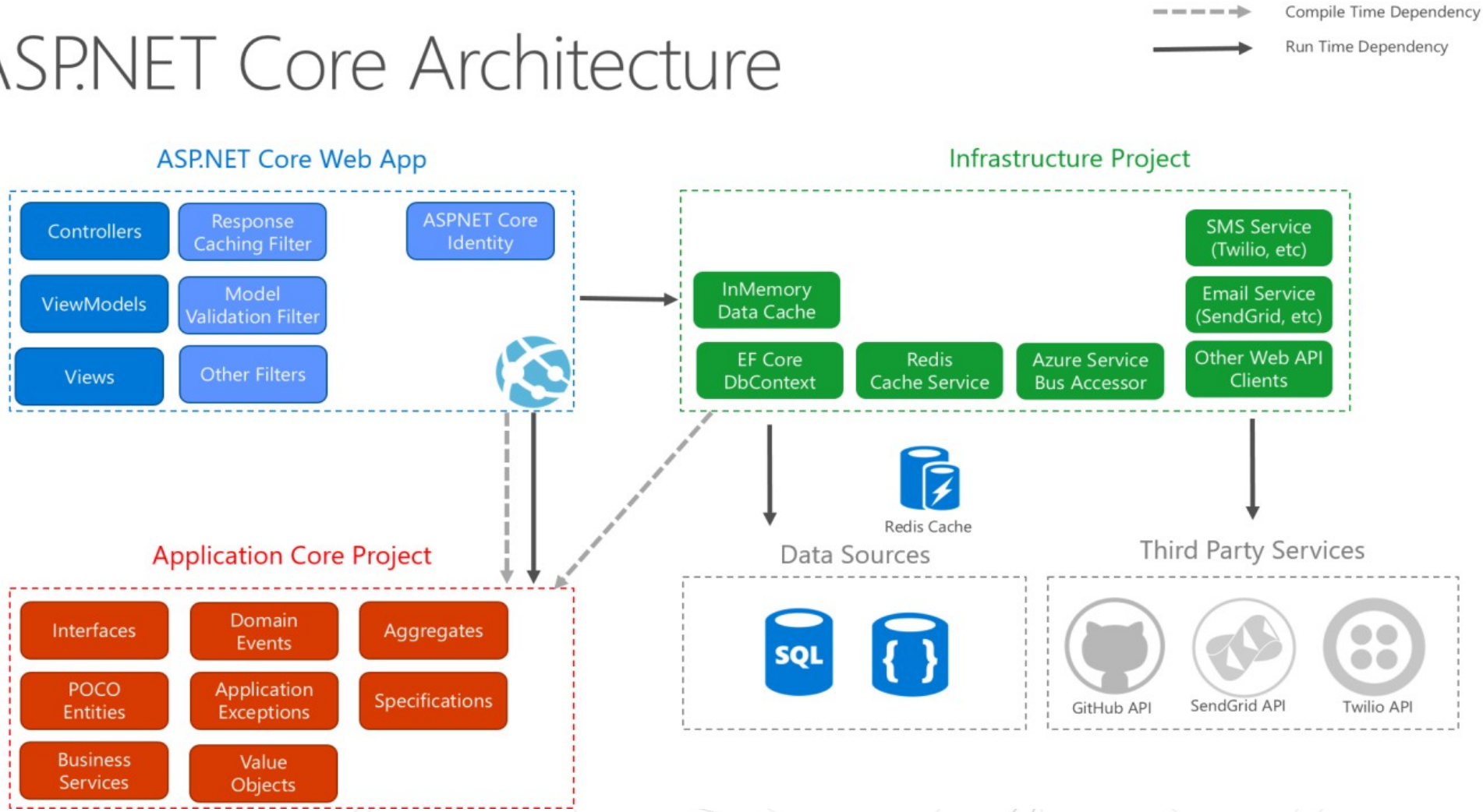
## Clean Architecture Layers (Onion view)



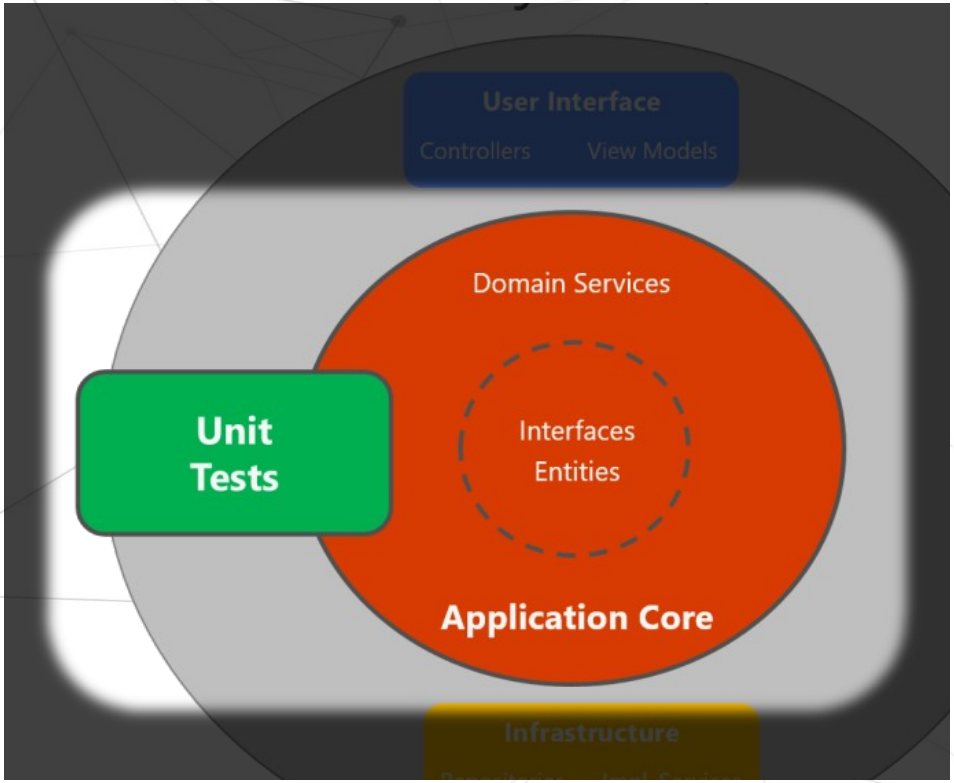
# Arquitetura Limpa

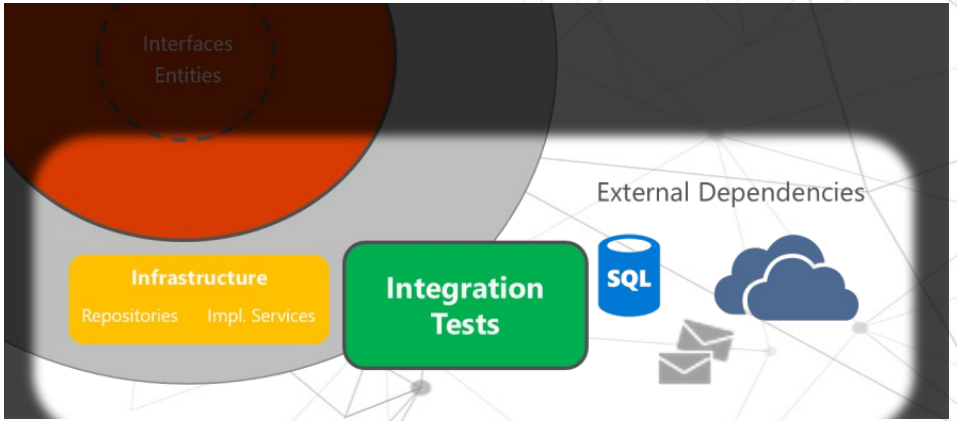


# ASP.NET Core Architecture

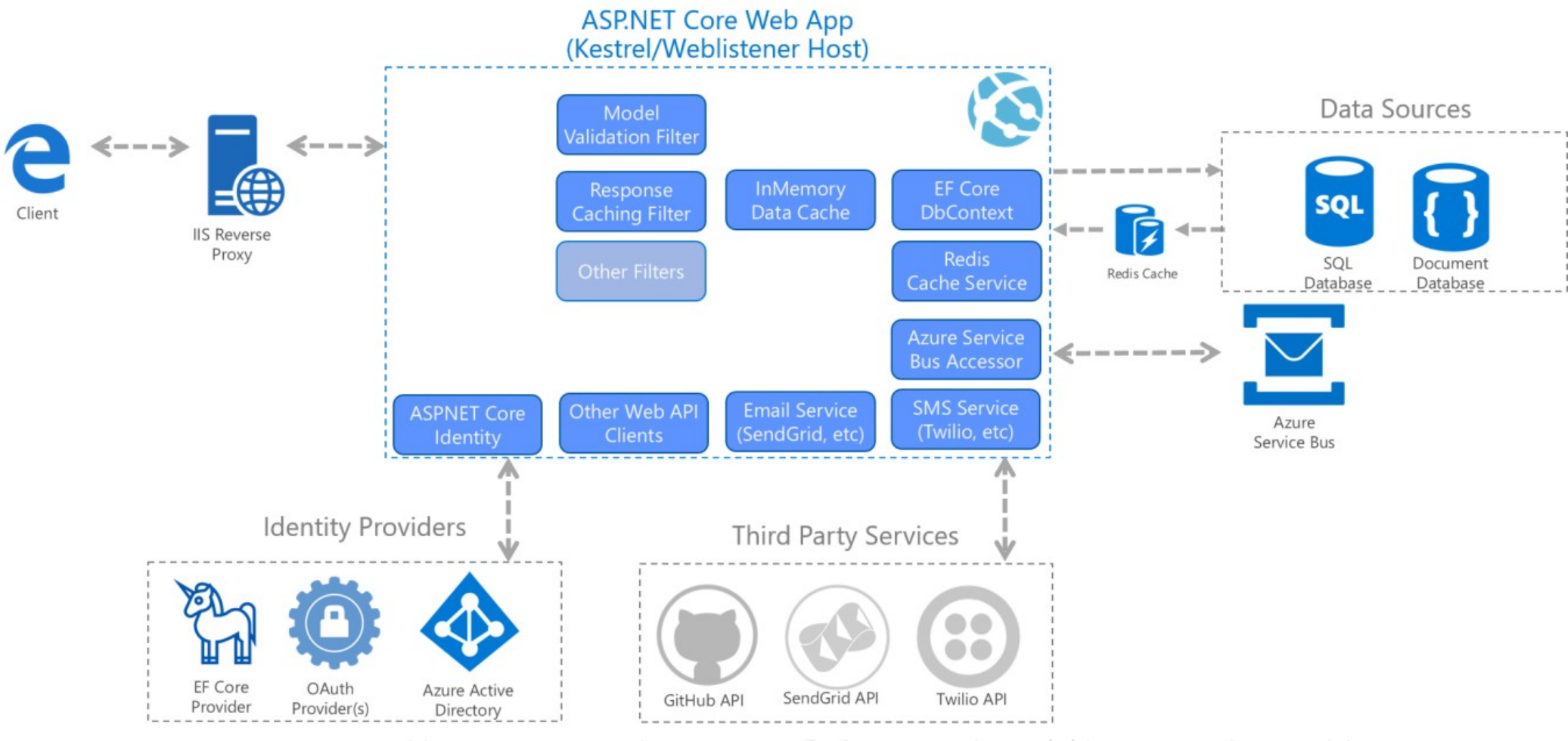


# Arquitetura Limpa



# ASP.NET Core Architecture



# Núcleo do Aplicativo

Contém o modelo de negócios, que inclui entidades, serviços e interfaces. Essas interfaces incluem abstrações para operações que serão executadas usando a Infraestrutura, como acesso a dados, acesso ao sistema de arquivos, chamadas de rede etc. Podemos usar os DTO (Data Transfer Objects)

Entidades (classes de modelo de negócios que são persistidas)

Agregações (grupos de entidades)

Interfaces

Serviços de Domínio

Especificações

Exceções personalizadas e cláusulas de proteção

Eventos e manipuladores de domínio

# Infraestrutura

Incluirá implementações de acesso a dados (context e EF) e implementações de serviços que devem interagir com os interesses de infraestrutura (devem implementar as interfaces definidas no Núcleo do Aplicativo)

Tipos do EF Core (DbContext, Migration)

Tipos de implementação de acesso a dados (Repositórios)

Serviços específicos de infraestrutura (por exemplo, FileLogger ou SmtpNotifier)

# Interface do usuário

Ponto de entrada para o aplicativo, deve referenciar o “Núcleo do Aplicativo” e interagir com a infraestrutura por meio das interfaces definidas no “Núcleo do Aplicativo”.

Não deve ser permitida nenhuma criação de instância direta de tipos da camada de Infraestrutura nem nenhuma chamada estática a esses tipos na camada de interface do usuário.

Controladores

Filtros personalizados

Middleware personalizado

Exibições

ViewModels

Inicialização

# Entity Framework Core (EF)

Tecnologia de acesso à dados leve, extensível, multi-plataforma e livre.

Funciona como mapeador relacional de objetos (O/RM)

Permite que aplicações .NET trabalhem com banco de dados usando objetos .NET

Elimina a necessidade da “maior parte” do código de acesso a dados.

# Modelo EF

O Acesso aos dados são executados por meio de um modelo

O modelo é criado a partir das classes de entidades e um objeto de contexto

O objeto de contexto representa uma sessão com o Banco de dados e permite consultar e salvar dados

# Abordagens do Desenvolvimento do Modelo

Gerar um modelo de um banco de dados existente

Codificar um modelo para corresponder ao banco de dados

Usar as *migrations* para criar um banco de dados a partir de um modelo.

Com as *migrations* é possível evoluir o banco de dados de acordo com as alterações no modelo.

UNIVEM DIGITAL

```
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;

namespace Intro;

public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(
            @"Server=(localdb)\mssqllocaldb;Database=Blogging;Trusted_Connection=True");
    }
}
```

Criação o Objeto de Contexto

UNIVEM DIGITAL

```
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public int Rating { get; set; }
    public List<Post> Posts { get; set; }
}
```

```
public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
```

Criação dos Modelos

# Consultas

As instâncias das classes de entidades são recuperadas do banco de dados utilizando a linguagen LINQ (Linguagem Integrada de Consulta)

```
using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

# Alteração de dados

Os dados são criados, excluídos e modificados no banco de dados utilizando as instâncias de  suas classes de entidades.

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

# Melhores práticas no uso do EF Core

O **conhecimento** de nível intermediário do servidor de banco de dados é essencial para arquitetar, depurar, definir perfis e migrar dados em aplicativos de produção

# Melhores práticas no uso do EF Core

Teste funcional e de integração: replicar o ambiente de produção:

Encontrar problemas no aplicativo que só aparecem ao usar uma versão ou edição específica do servidor de banco de dados.

Capturar alterações interruptivas ao atualizar o EF Core e outras dependências.

# Melhores práticas no uso do EF Core

**Teste de desempenho e estresse** com cargas representativas. O uso ingênuo de alguns recursos não ganha escala.

# Melhores práticas no uso do EF Core

**Revisão de segurança:** Tratamento de cadeias de conexão e de outros segredos, permissões de banco de dados para operação de não implantação, validação de entrada para SQL bruto, criptografia para dados confidenciais.

# Melhores práticas no uso do EF Core

Verificar se o registro em log e o diagnóstico são suficientes e utilizáveis.

Recuperação de erro. Tenha contingências para cenários comuns de falha, como reversão de versão, expansão e balanceamento de carga, mitigação de DoS e backups de dados.

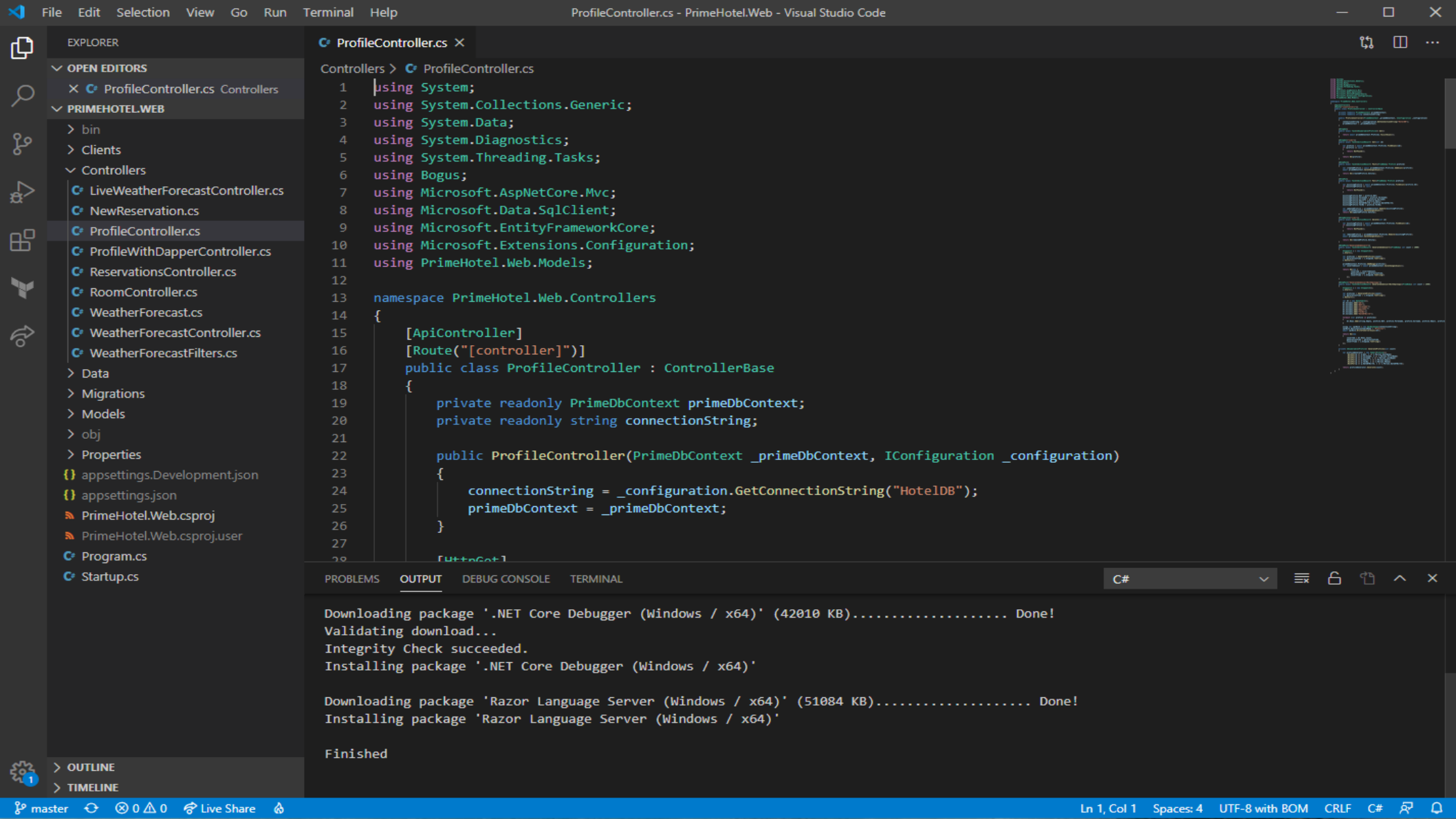
# Melhores práticas no uso do EF Core

**Implantação e migração de aplicativos**. Planeje como as migrações serão aplicadas durante a implantação; esteja preparado para facilitar a recuperação de erros fatais durante a migração.

# Melhores práticas no uso do EF Core

As migrações devem ser cuidadosamente testadas antes de serem aplicadas aos dados de produção.

A forma do *schema* e os tipos de coluna não podem ser facilmente alterados quando as tabelas contêm dados de produção.


File Edit Selection View Go Run Terminal Help
ProfileController.cs - PrimeHotel.Web - Visual Studio Code
EXPLORER
OPEN EDITORS
ProfileController.cs Controllers
PRIMEHOTEL.WEB
bin
Clients
Controllers
LiveWeatherForecastController.cs
NewReservation.cs
ProfileController.cs
ProfileWithDapperController.cs
ReservationsController.cs
RoomController.cs
WeatherForecast.cs
WeatherForecastController.cs
WeatherForecastFilters.cs
Data
Migrations
Models
obj
Properties
appsettings.Development.json
appsettings.json
PrimeHotel.Web.csproj
PrimeHotel.Web.csproj.user
Program.cs
Startup.cs
OUTLINE
TIMELINE
ProfileController.cs
Controllers > ProfileController.cs
1 using System;
2 using System.Collections.Generic;
3 using System.Data;
4 using System.Diagnostics;
5 using System.Threading.Tasks;
6 using Bogus;
7 using Microsoft.AspNetCore.Mvc;
8 using Microsoft.Data.SqlClient;
9 using Microsoft.EntityFrameworkCore;
10 using Microsoft.Extensions.Configuration;
11 using PrimeHotel.Web.Models;
12
13 namespace PrimeHotel.Web.Controllers
14 {
15 [ApiController]
16 [Route("[controller]")]
17 public class ProfileController : ControllerBase
18 {
19 private readonly PrimeDbContext primeDbContext;
20 private readonly string connectionString;
21
22 public ProfileController(PrimeDbContext _primeDbContext, IConfiguration _configuration)
23 {
24 connectionString = _configuration.GetConnectionString("HotelDB");
25 primeDbContext = _primeDbContext;
26 }
27
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
C#
Downloading package '.NET Core Debugger (Windows / x64)' (42010 KB)..... Done!
Validating download...
Integrity Check succeeded.
Installing package '.NET Core Debugger (Windows / x64)'
Downloading package 'Razor Language Server (Windows / x64)' (51084 KB)..... Done!
Installing package 'Razor Language Server (Windows / x64)'
Finished
master
0 0
Live Share
Ln 1, Col 1
Spaces: 4
UTF-8 with BOM
CRLF
C#