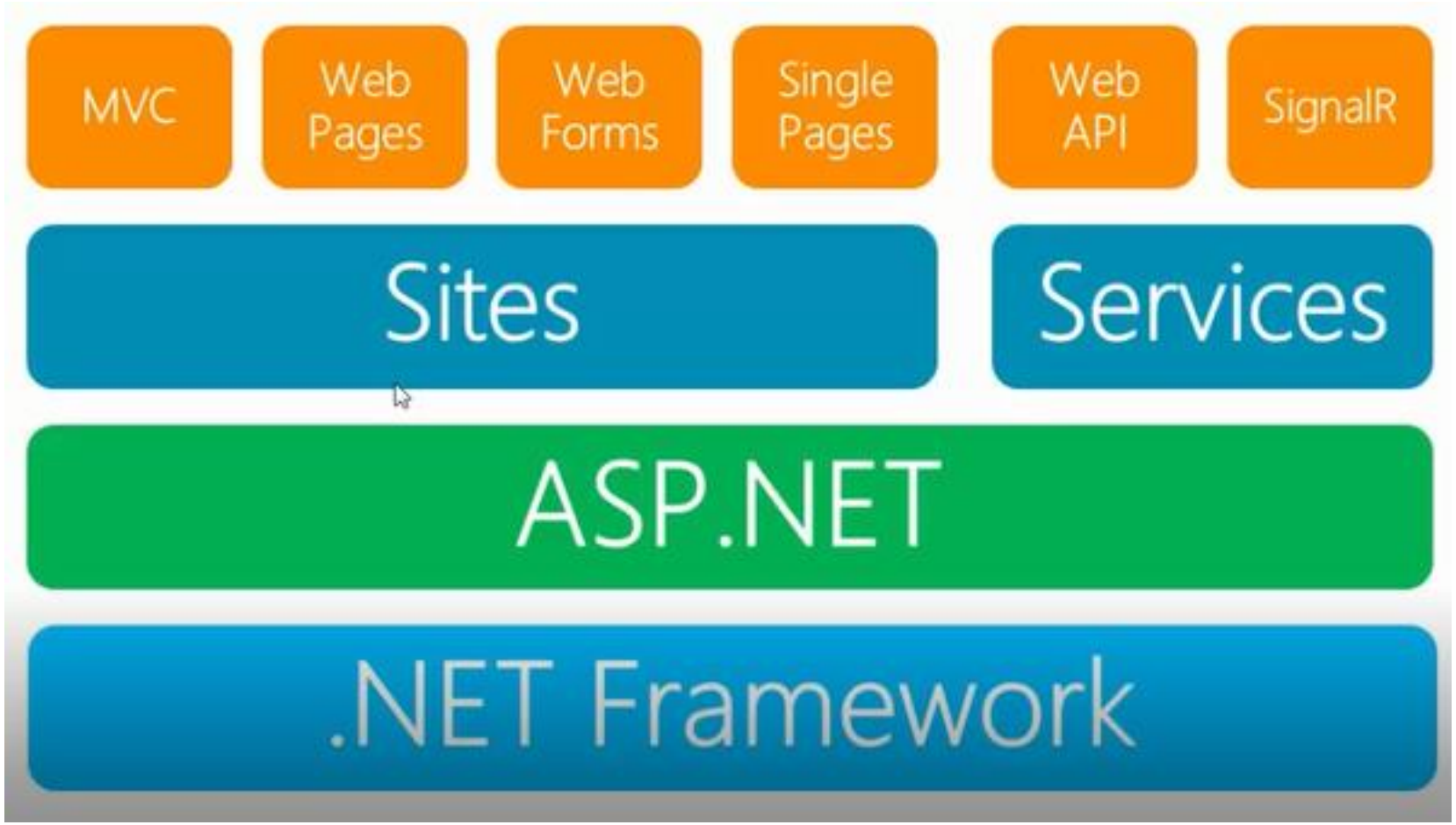
Plataforma.NET: ASP.NET Core

RUI FLEXA - MCPD

Origens do ASP.NET

ASP	• ASP clássico lançado em 1996
ASP.NET WebForms	• Lançado em 2002 junto com o .NET Framework
ASP.NET MVC	• Alternativa ao WebForms, lançado em 2009
ASP.NET Core	• Inicialmente com o nome .vNext e lançado no final de 2015 e com início de divulgação em 2016 e atualmente na versão 2.1

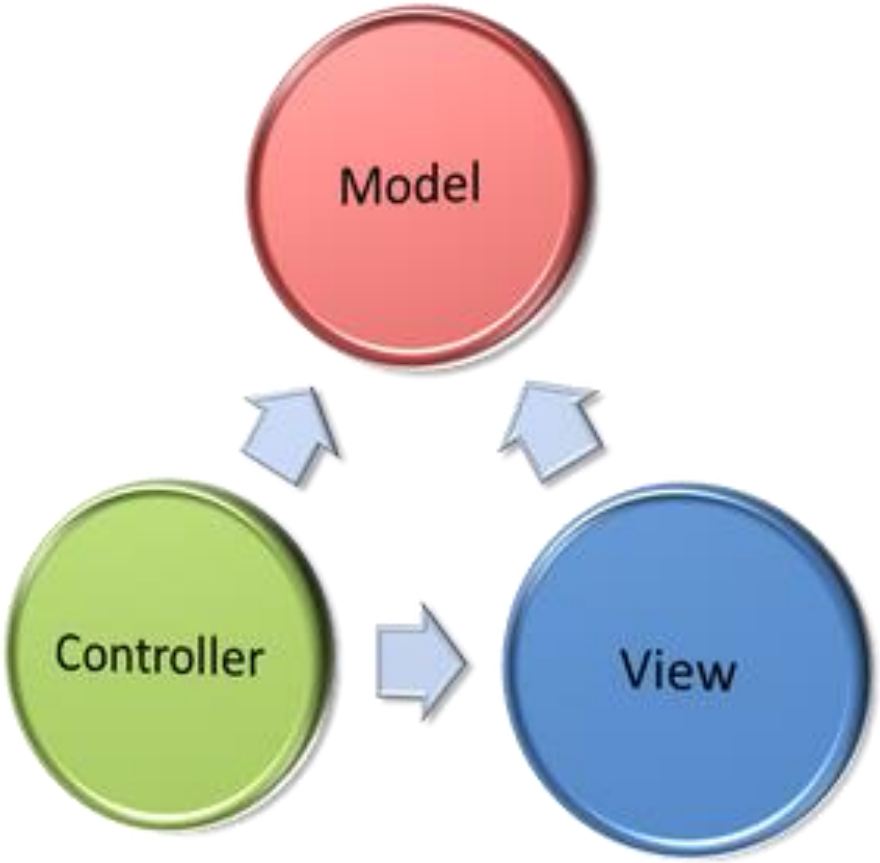
Como está o ASP.NET (Ecossistema)



Desvantagens WebForms x MVC

- Não há controle do HTML e Javascript gerado
- Alguns controles não estão em conformidade com o W3C
- Difícil integração com frameworks Javascript
- A UI (user interface) é quase impossível de se testar
- Problemas para se utilizar SEO (roteamento)

O padrão MVC



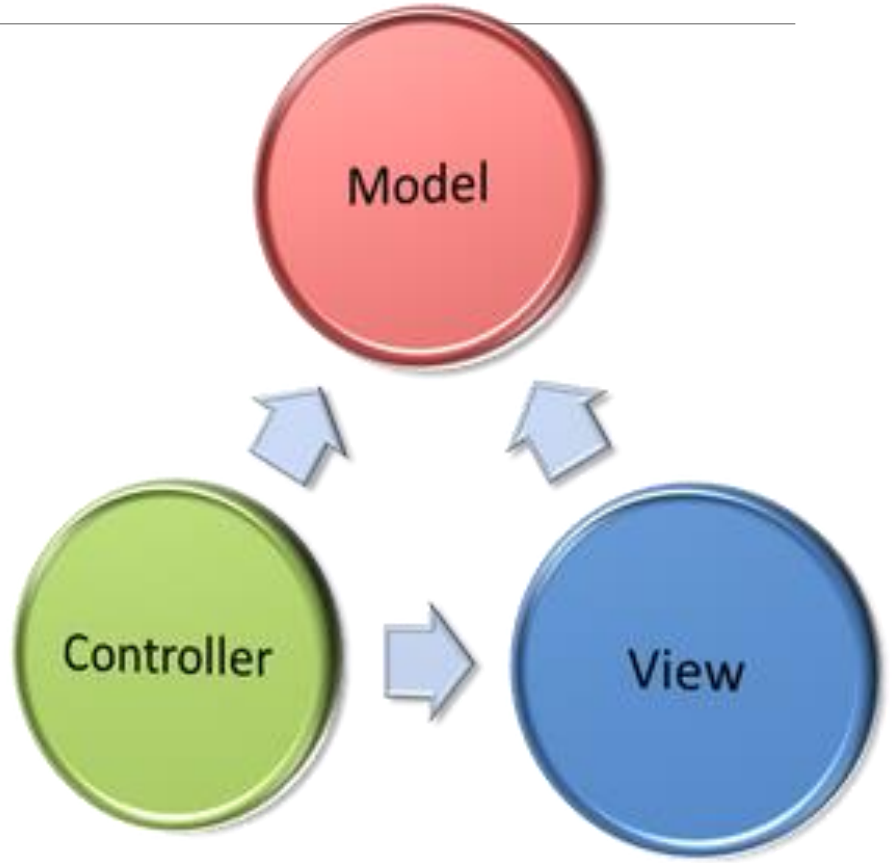
- Padrão arquitetônico
- Separação de responsabilidades
- Criado (descrito) em 1976 pela Xerox
- Popular em Java, Ruby on Rails, Apple, Cocoa, etc

Vantagens do ASP.NET MVC

- Separação de responsabilidades (cada camada com a sua)
- Testabilidade
- Reusabilidade
- Escalabilidade
- Manutenção facilitada
- Total controle do HTML e script gerado
- Suporta TDD em todos os aspectos
- É o framework oficial do ASP.NET 5

Model

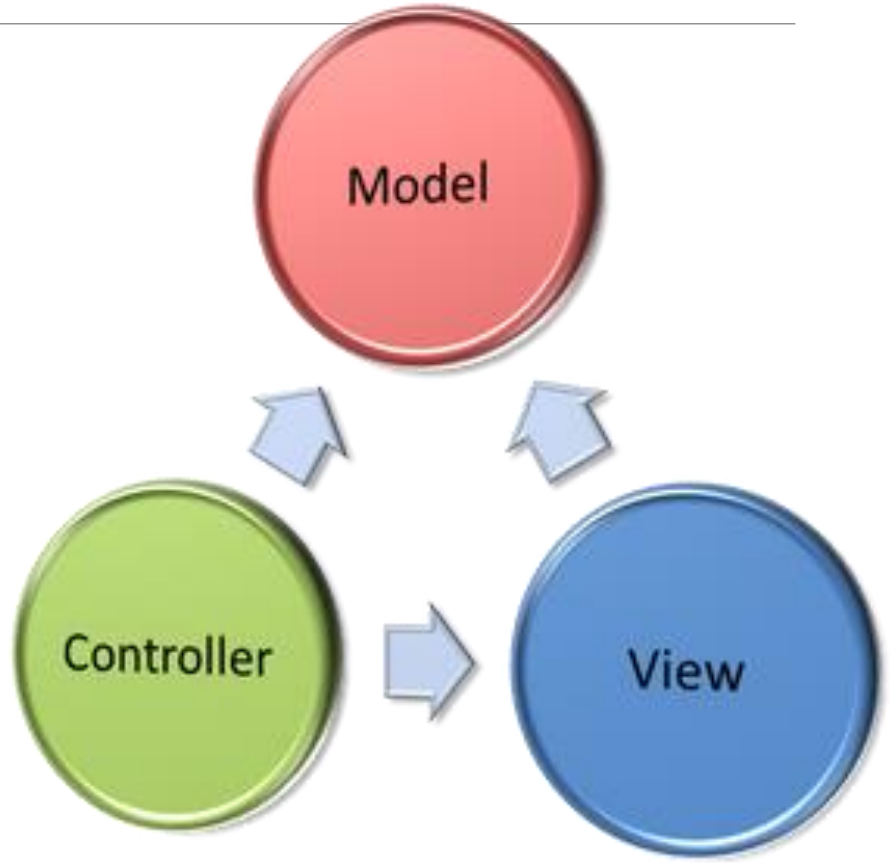
```
public class Cliente
{
    public int ID { get; set; }
    public string Nome { get; set; }
    public DateTime DataCadastro { get; set; }
}
```



Controller

Coluna cervical

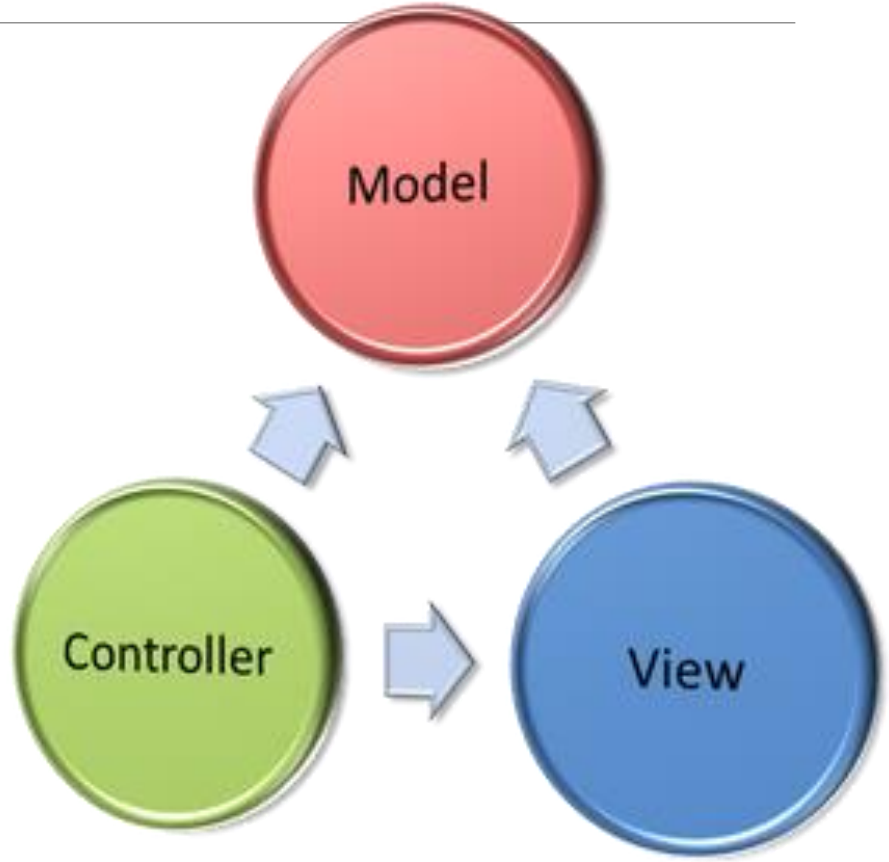
```
public ActionResult Details(String id)
{
    Cliente cliente = db.cliente.Find(id);

    if (cliente == null)
    {
        return HttpNotFound();
    }

    return View(cliente);
}
```



View

View

```
<th>
    @Html.DisplayNameFor(model => model.Nome)
</th>
<th>
    @Html.DisplayNameFor(model => model.CNPJ)
</th>
<th>
    @Html.DisplayNameFor(model => model.DataCadastro)
</th>
```



Front-End – Conhecendo Scripts

Base de toda a internet!

Front-End – Conhecendo Scripts

Bootstrap é um framework de HTML, CSS e javascript mais popular para o desenvolvimento ágil de projetos responsivos e mobile na web

http://getbootstrap.com

Front-End – Conhecendo Scripts

É uma biblioteca javascript cross-browser desenvolvida para simplificar os scripts client-side que interagem com o HTML

Usada por 77% dos 10 mil sites mais visitados do mundo, jQuery é a mais popular das bibliotecas javascript

Front-End – Conhecendo Scripts

Modernizr é uma pequena biblioteca javascript que detecta a disponibilidade de novas características do HTML5 e CSS3 nos browsers.

O que o Modernizr faz é simples: detecta quais features um determinado browser suporta e insere classes no HTML para fazer uma versão alternativa de visual ou solução

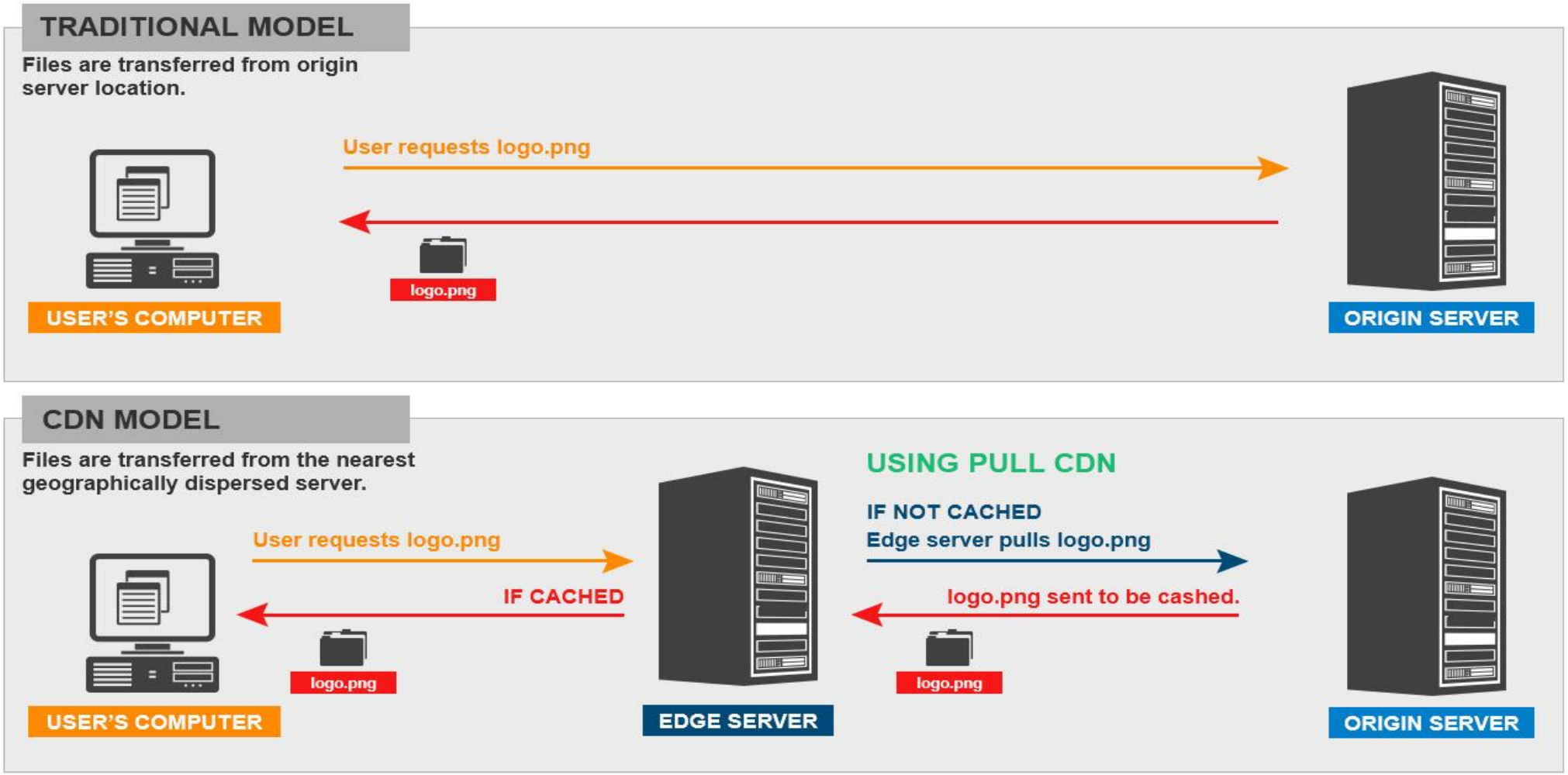
Modernizar navegação independente da versão do browser

Front-End – Gerenciando Scripts

- O sucesso de sua aplicação começa no gerenciamento dos seus scripts
- HTML + CSS + JavaScript + Fonts
- Considerar cenário mobile 3G e 4G
- Nem todos os visitantes possuem internet de alta velocidade

CDN – Content Delivery Network

CDN – Content Delivery Network



Limite de Download simultâneo por Host

6 downloads / host

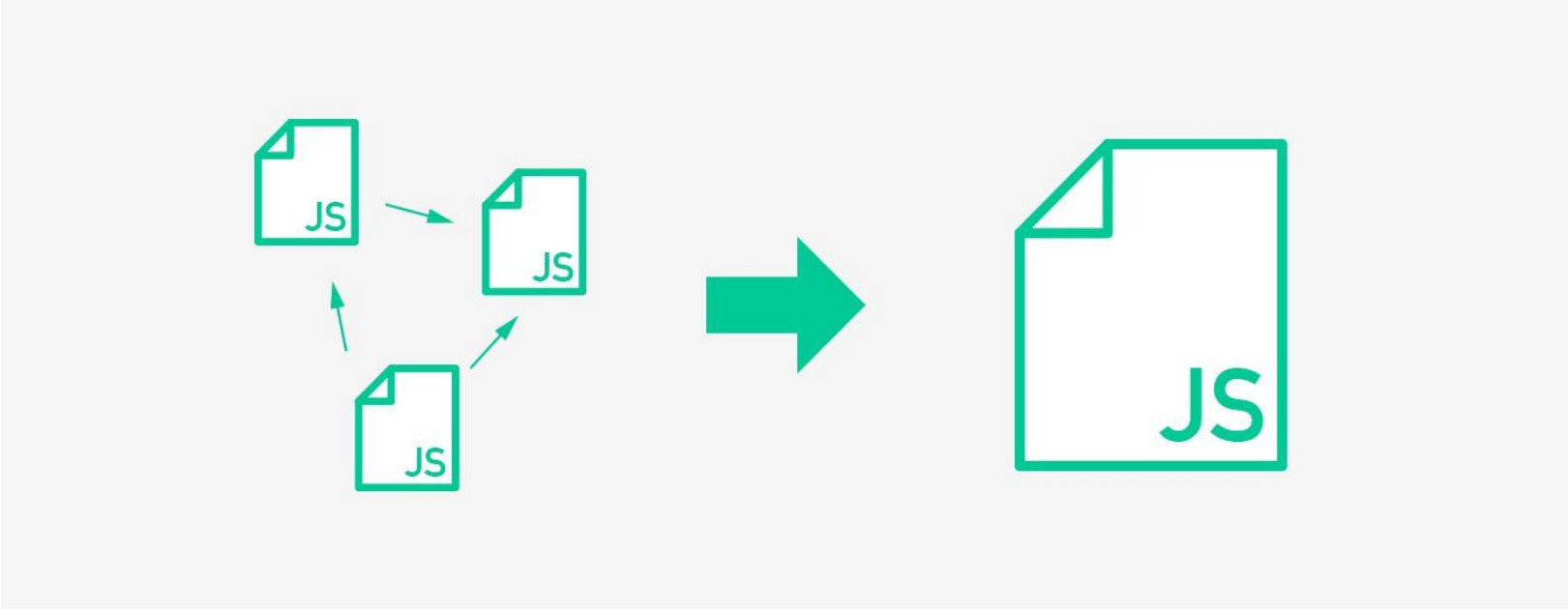
8 downloads / host

6 downloads / host

4 downloads / host

Gerenciando Scripts - Bundling

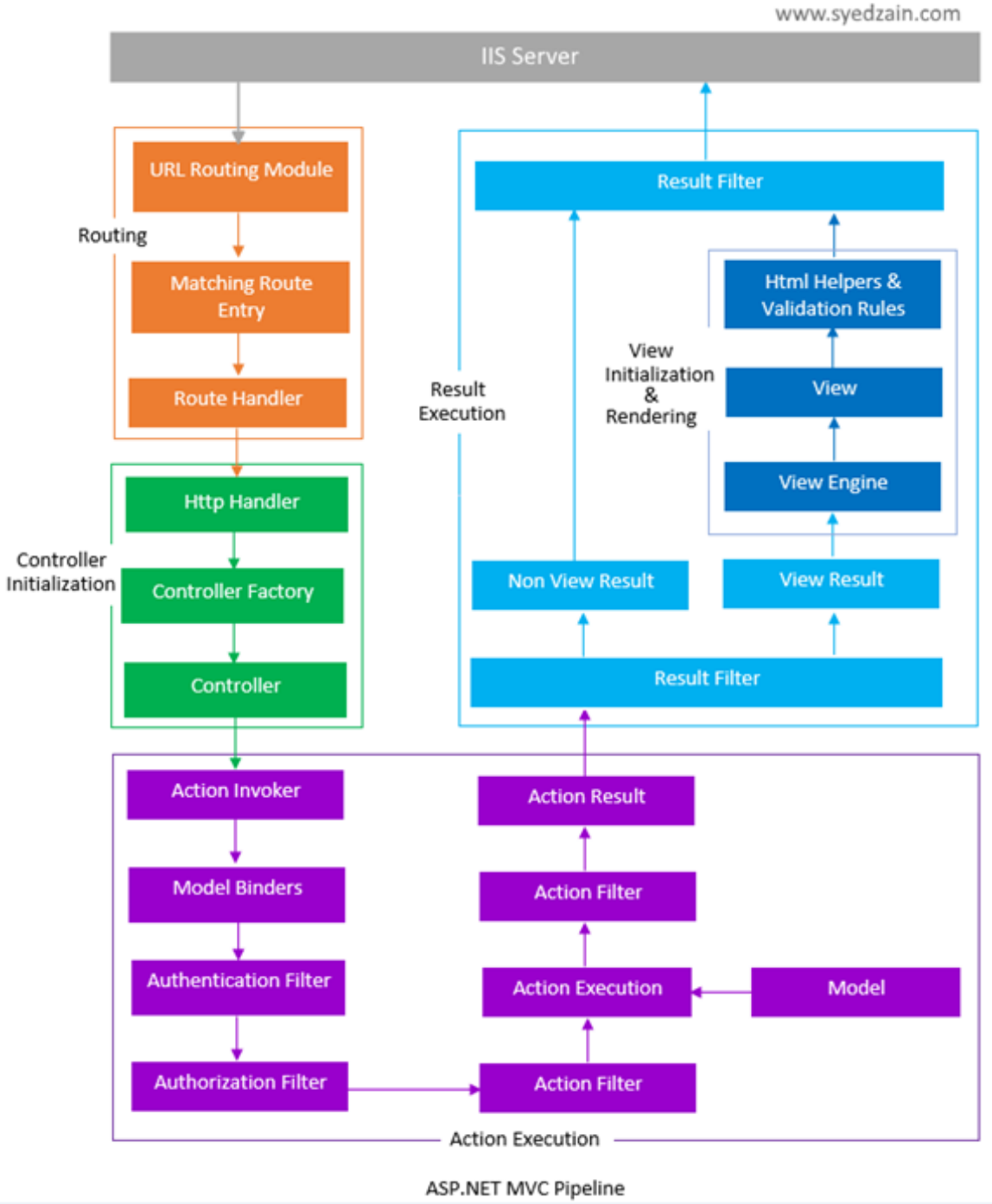
É o processo de transformar diversos arquivos do mesmo tipo em apenas um só, poupando ciclos de download do browser



Gerenciando Scripts - Minification

Consiste em comprimir o tamanho dos arquivos através da remoção de espaços em branco, comentários, quebras de linha e substituição de variáveis

MVC Pipeline – Ciclo de vida de uma aplicação ASP.NET Core



ASP.NET Core

ASP.NET Core - Introdução

- ASP.NET Core um novo ASP.NET
- .NET 2017
- IDEs

Por que um novo Stack?

ASP.NET Core

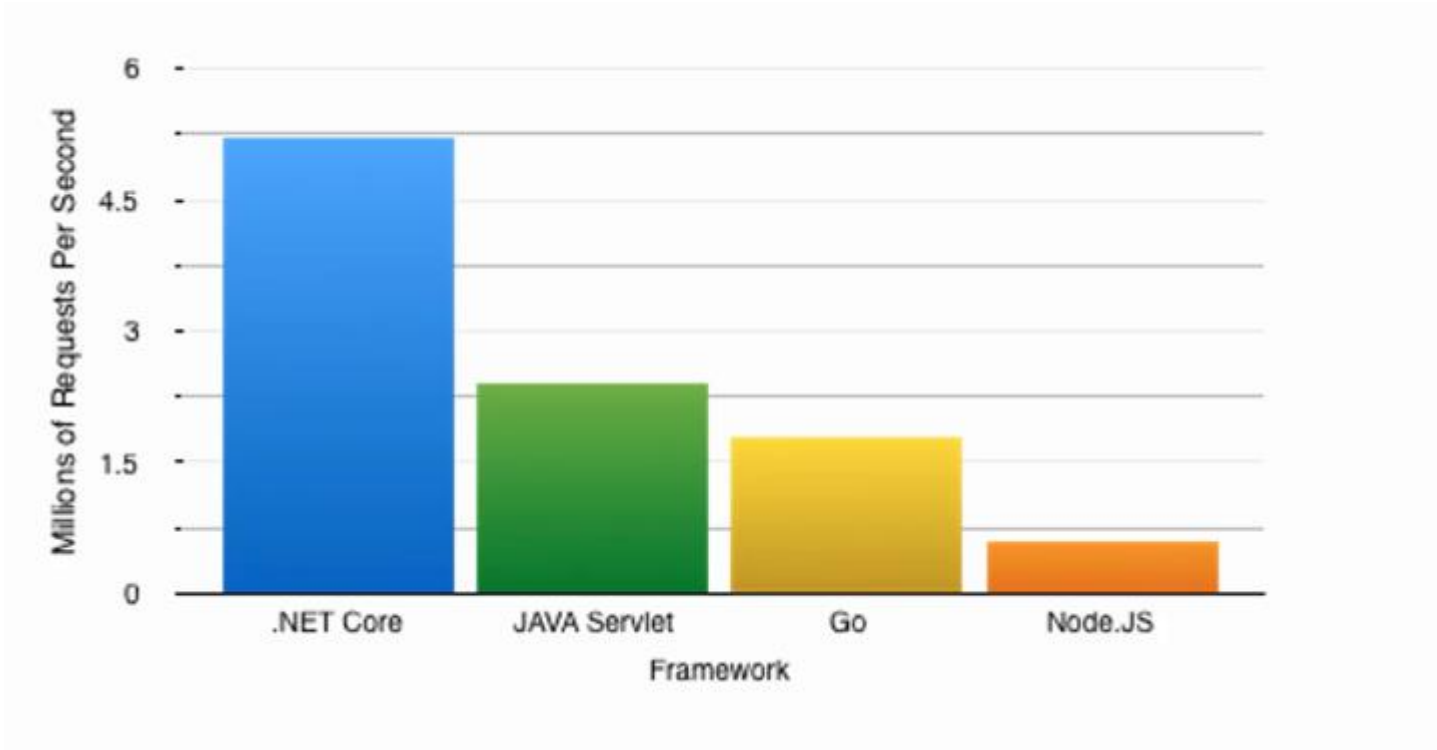
Inovação

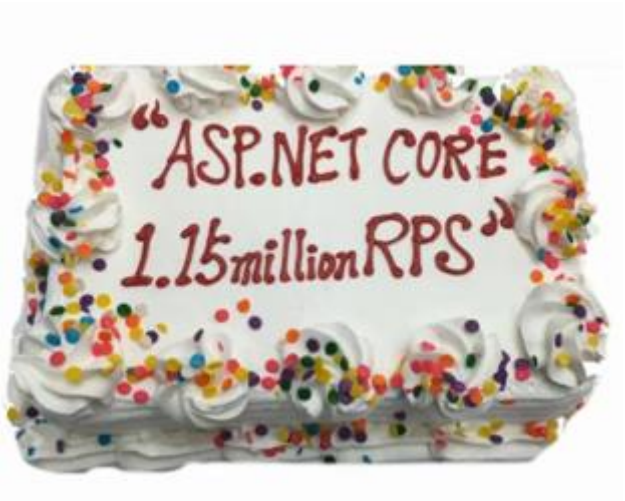
Open Source

Multiplataforma

Inovação

https://github.com/aspnet/benchmarks





Open Source

425 repositories

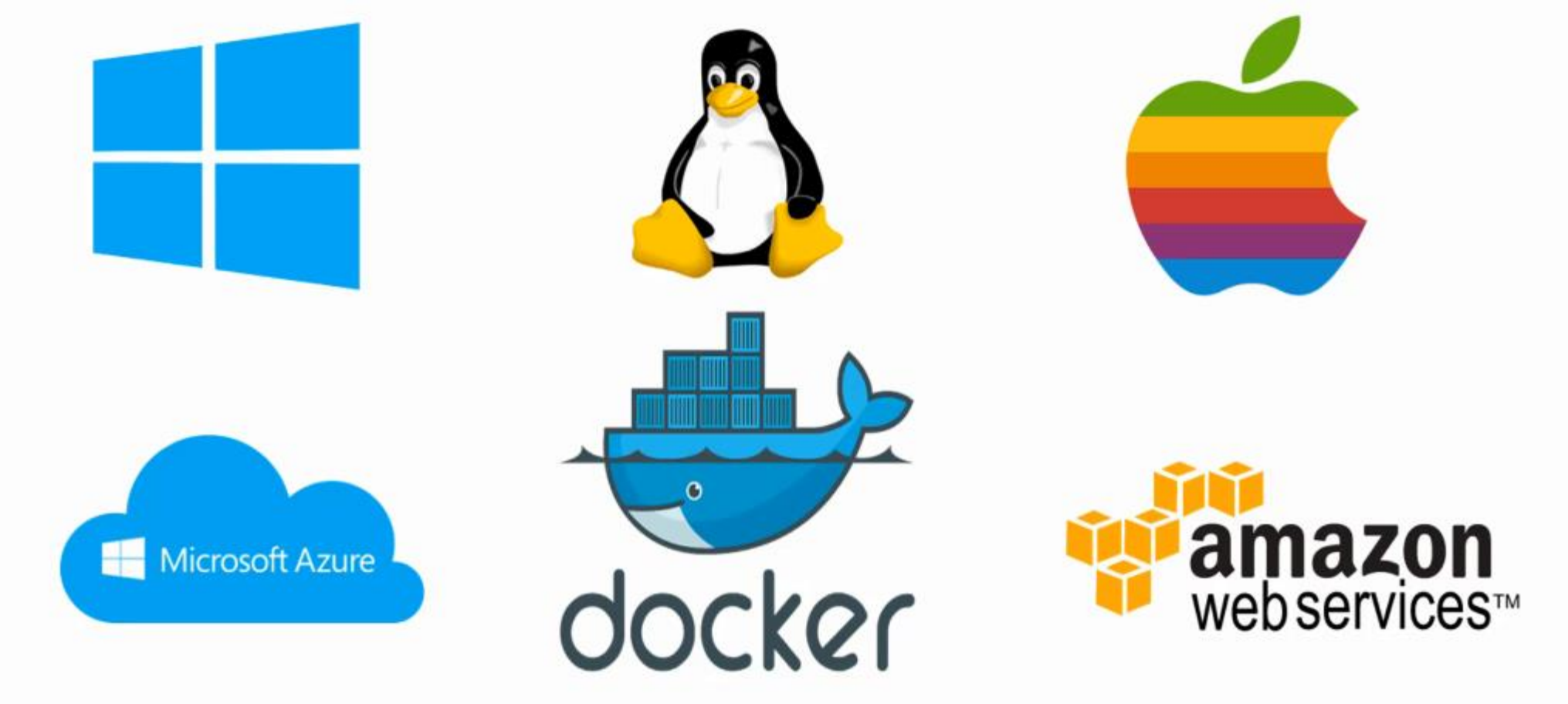
41541 forks

8641 contribuitors

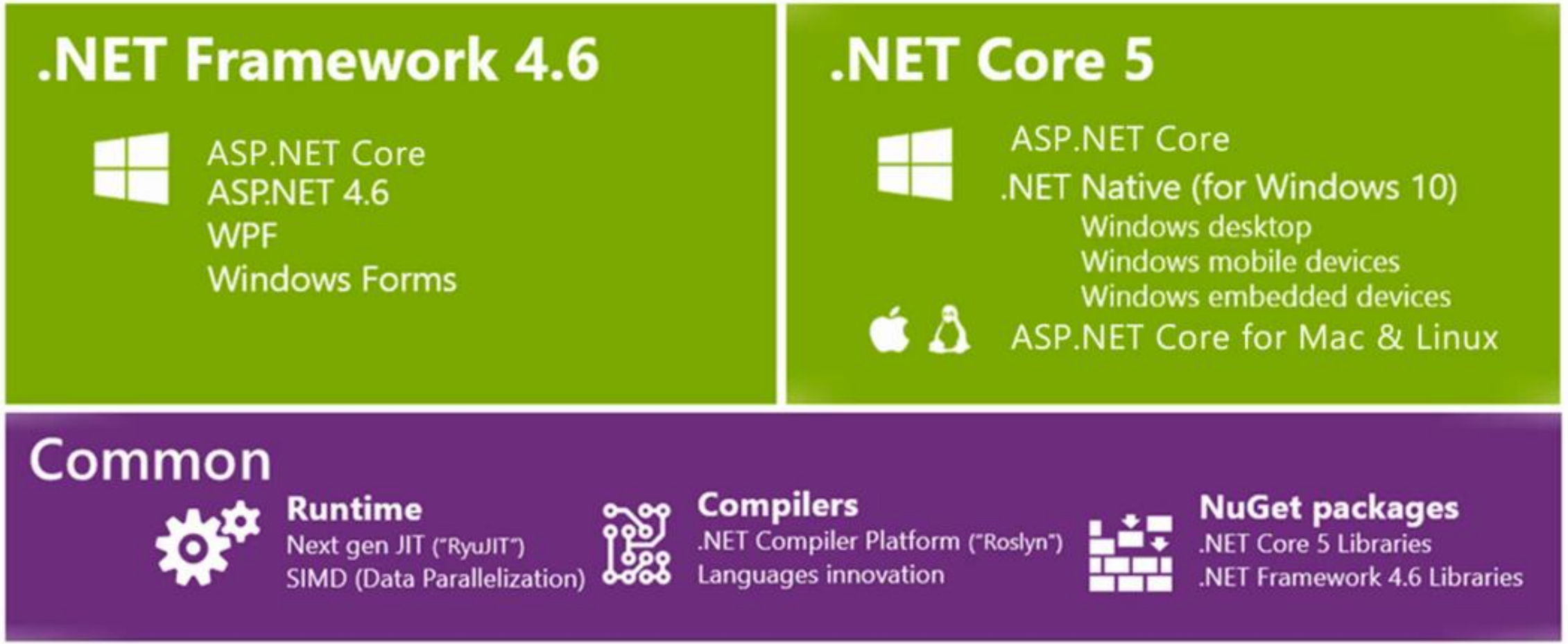
https://dotnetfoundation.org

https://github.com/aspnet/home

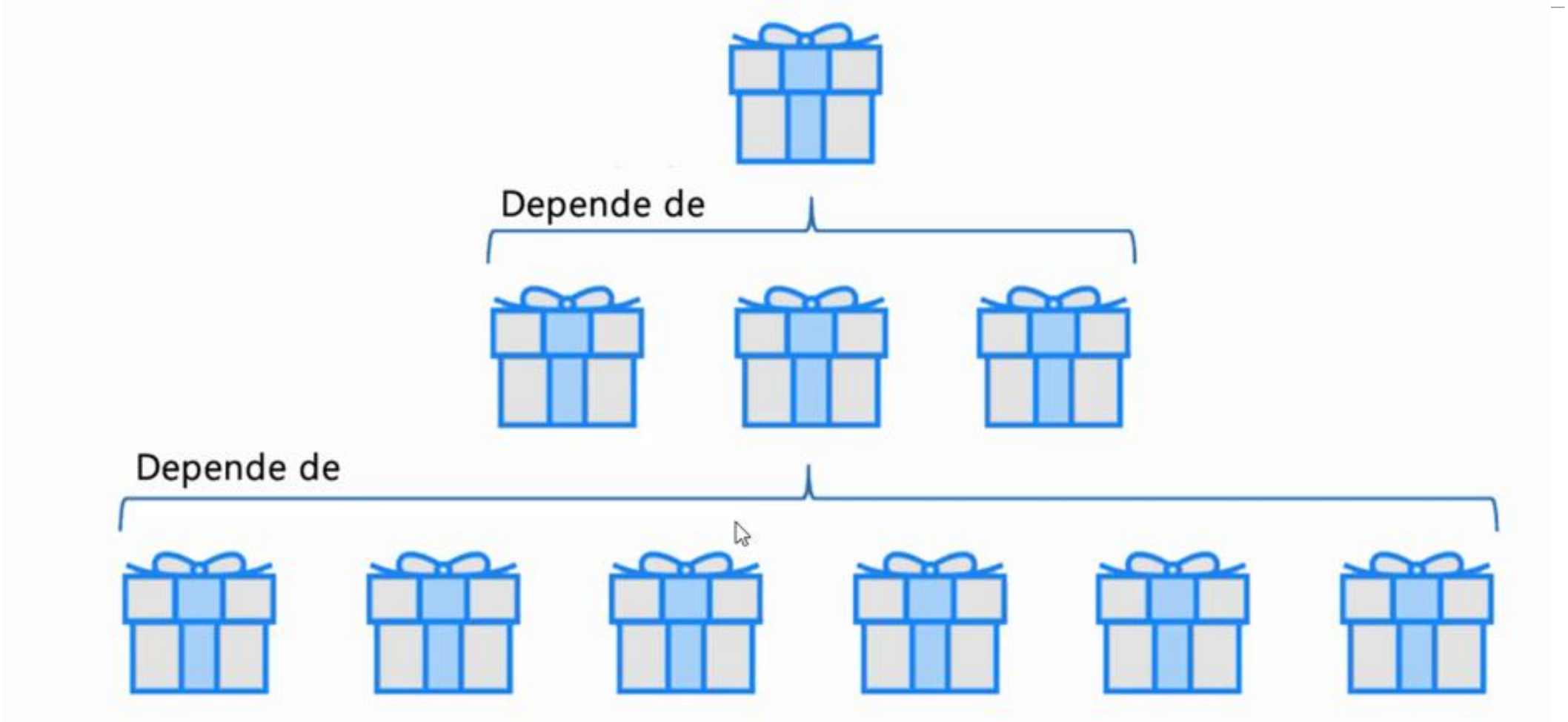
Multiplataforma



Estrutura


.NET Framework 4.6
ASP.NET Core
ASP.NET 4.6
WPF
Windows Forms
.NET Core 5
ASP.NET Core
.NET Native (for Windows 10)
Windows desktop
Windows mobile devices
Windows embedded devices
ASP.NET Core for Mac & Linux
Common
Runtime
Next gen JIT ("RyuJIT")
SIMD (Data Parallelization)
Compilers
.NET Compiler Platform ("Roslyn")
Languages innovation
NuGet packages
.NET Core 5 Libraries
.NET Framework 4.6 Libraries

NuGet Packages



.NET Standard Library

.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	**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	**vNext**
Xamarin.iOS	**10.0**	10.0	10.0	10.0	10.0	10.0	10.0	**vNext**
Xamarin.Android	**7.0**	7.0	7.0	7.0	7.0	7.0	7.0	**vNext**
Universal Windows Platform	**10.0**	10.0	10.0	10.0	10.0	**vNext**	vNext	vNext
Windows	**8.0**	8.0	**8.1**					
Windows Phone	**8.1**	8.1	8.1					
Windows Phone Silverlight	**8.0**							

IDEs

Visual Studio Code
- Multiplataforma
- Open Source
- Free

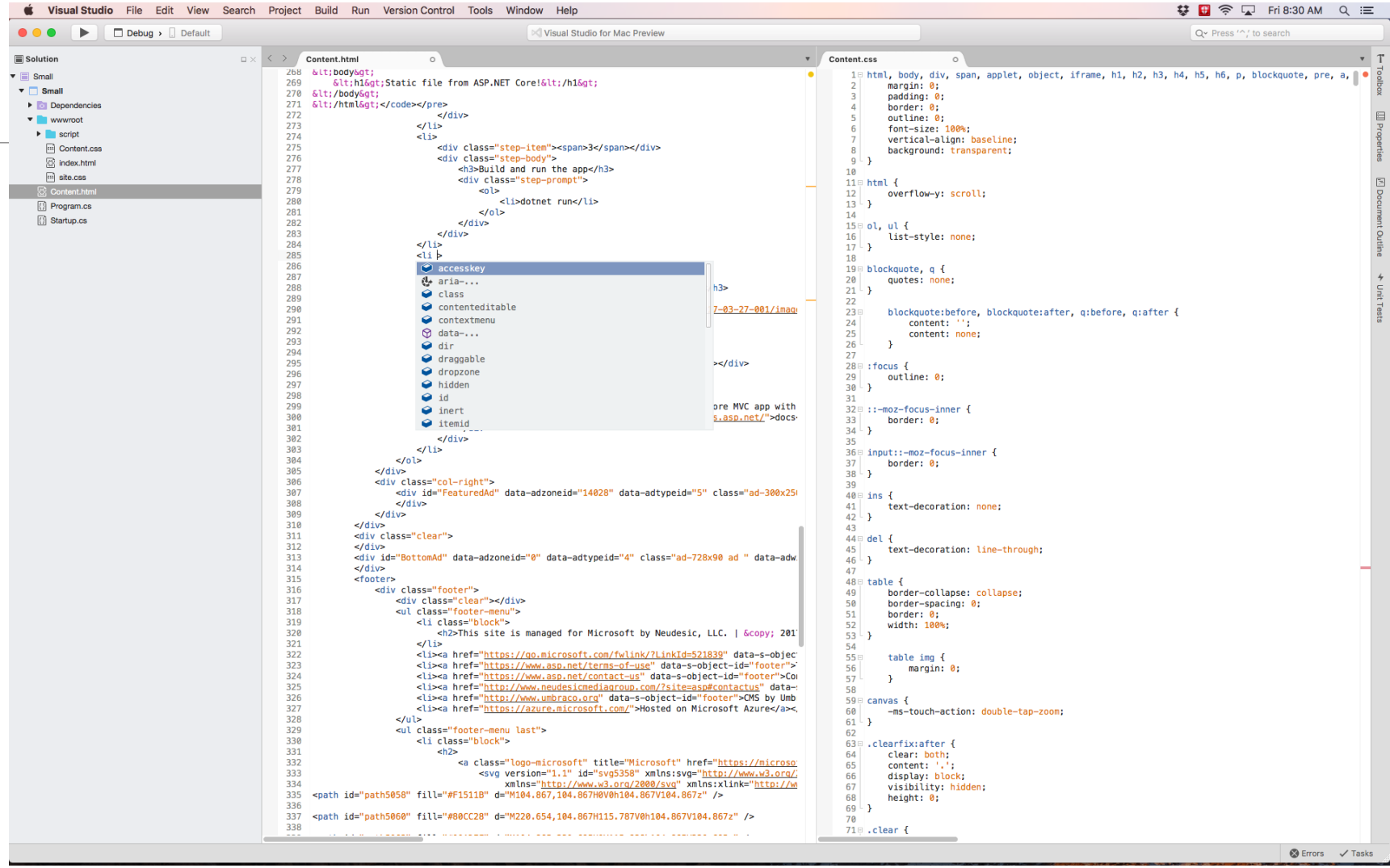
Visual Studio 2015
- Windows
- Free (Community)
- Professional
- Enterprise

Visual Studio 2017
- Windows
- Free (Community)
- Professional
- Enterprise

IDEs

Visual Studio for Mac

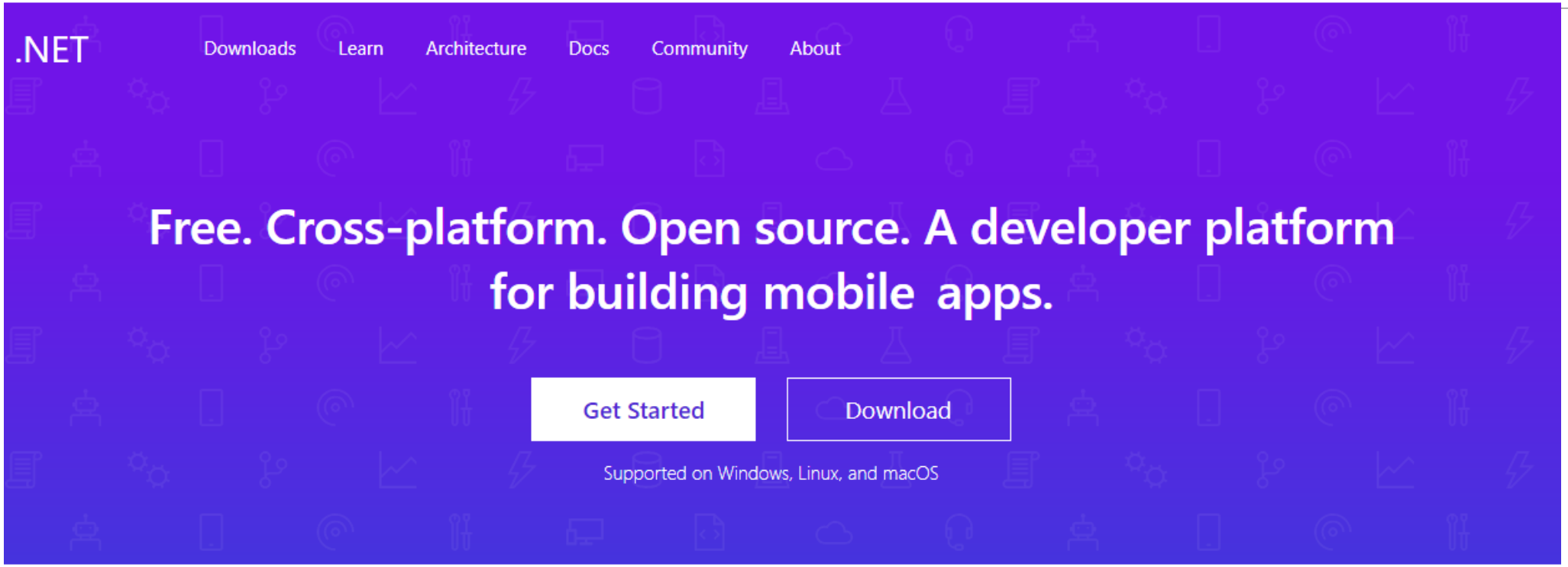
- Apenas Mac
- Licenciado (em definição)



Vamos navegar para..

https://www.microsoft.com/net/

Microsoft



Web

Build web apps and services for

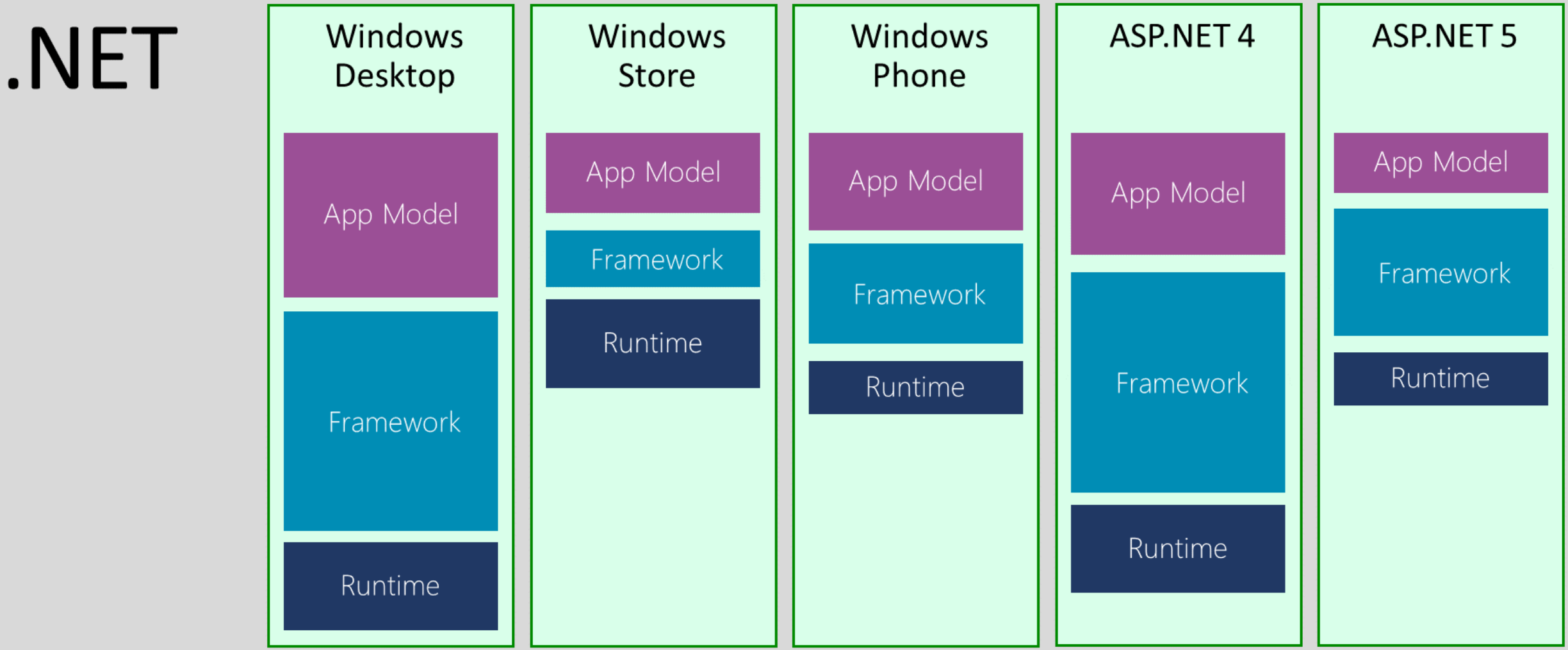
Mobile

Use a single codebase to build

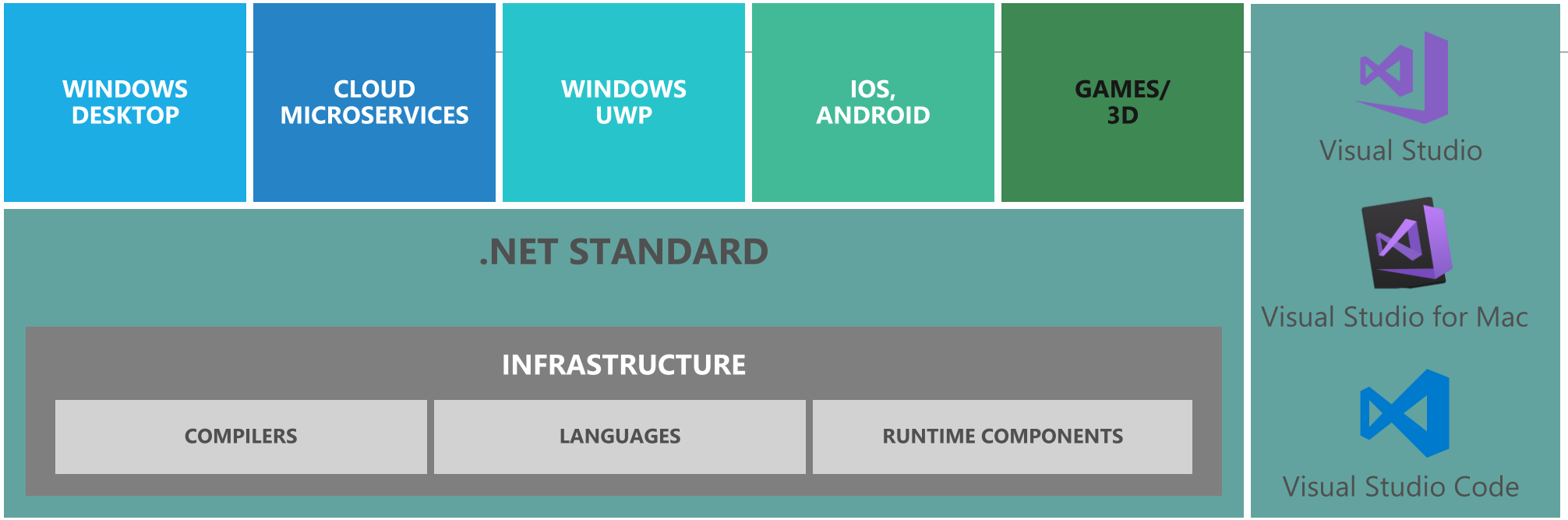
Desktop

Create beautiful and compelling

.NET através de Windows/Web Platforms



.NET Standard 2017 / 2018

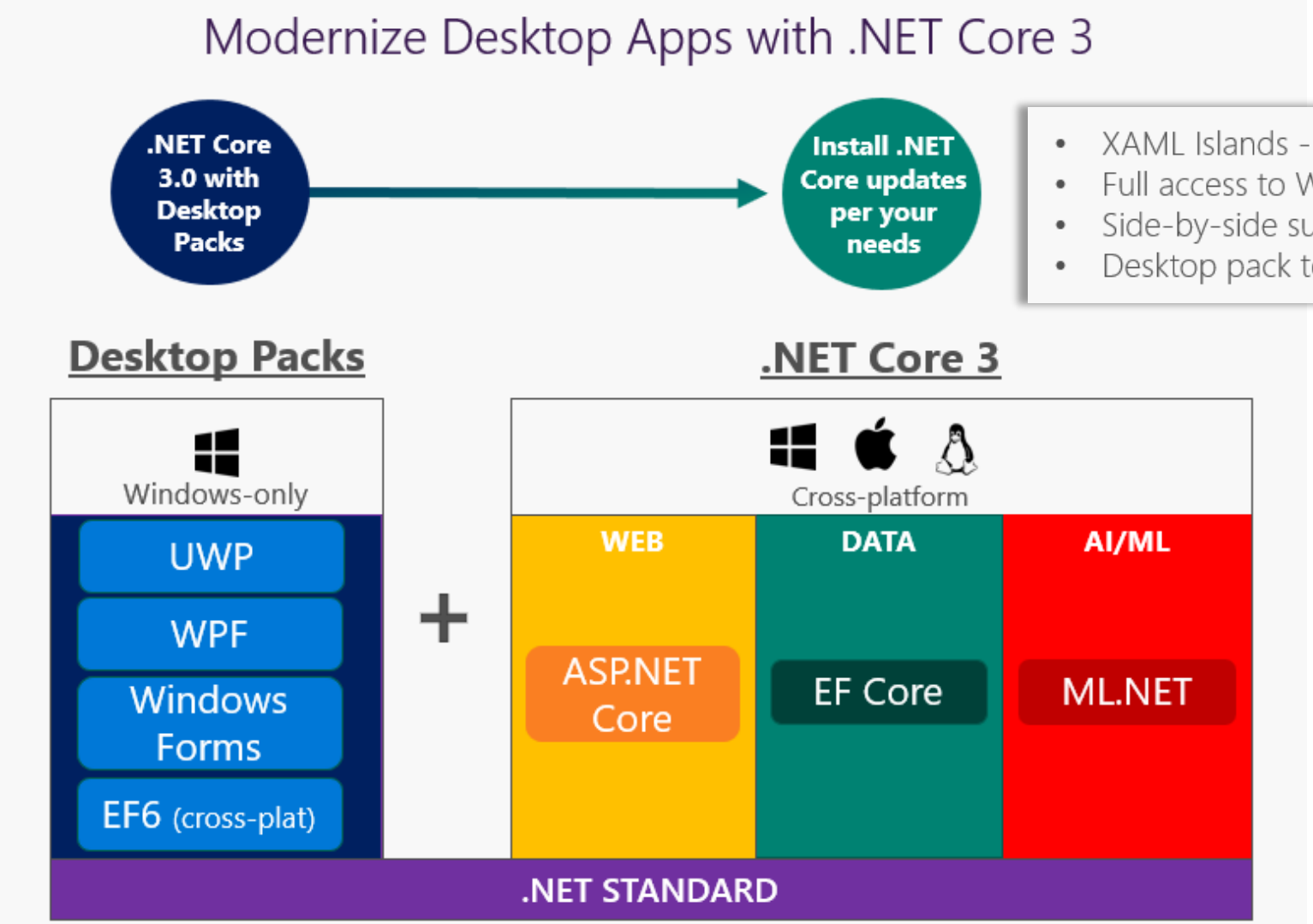


NET Standard permite compartilhamento de código, binários e skills entre .NET client, server,etc

. NET Standard fornece uma especificação para qualquer plataforma para implementação

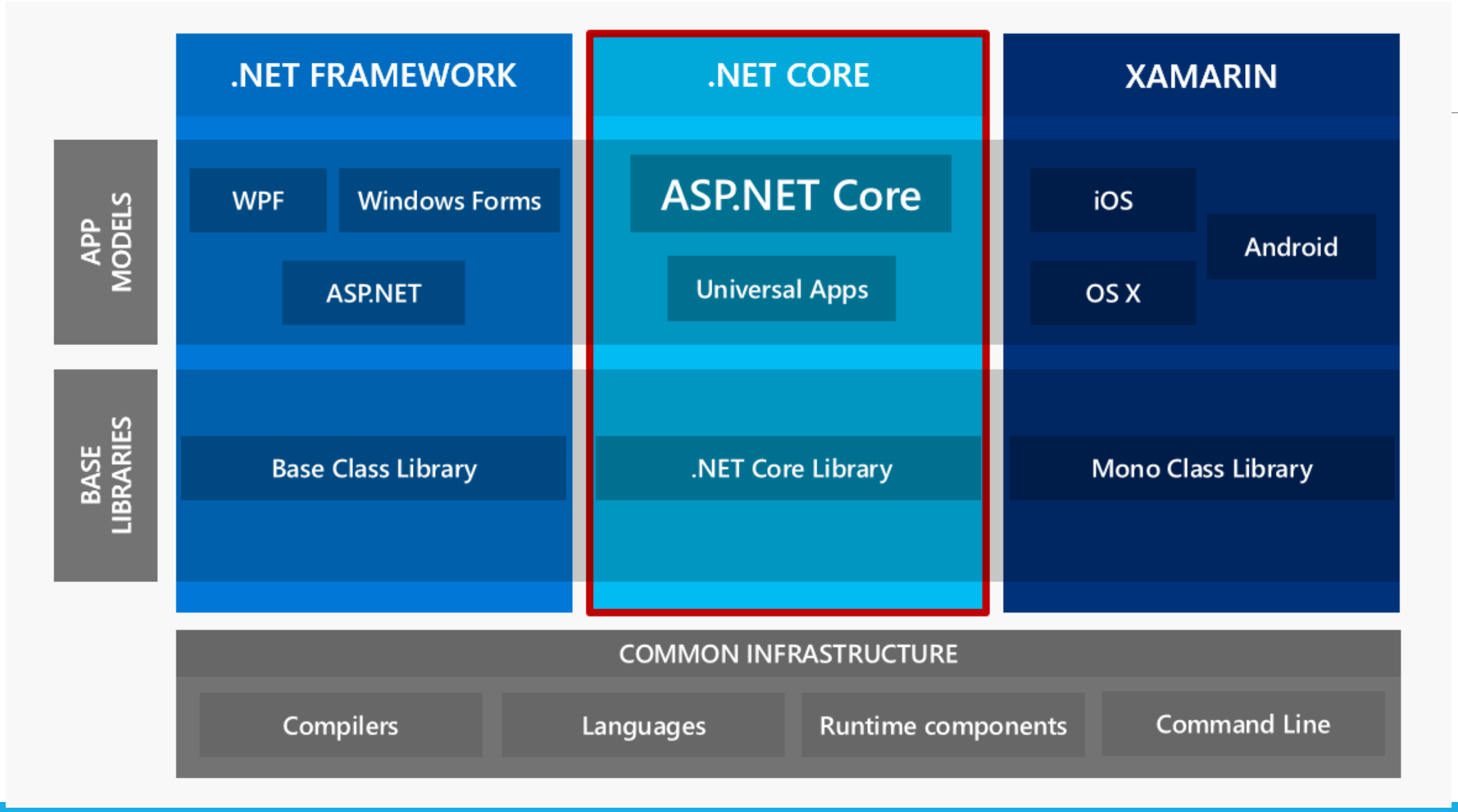
All .NET runtimes fornecidos pela Microsoft implementam o .NET Standard

.NET 3.0 em 2019 e Além...

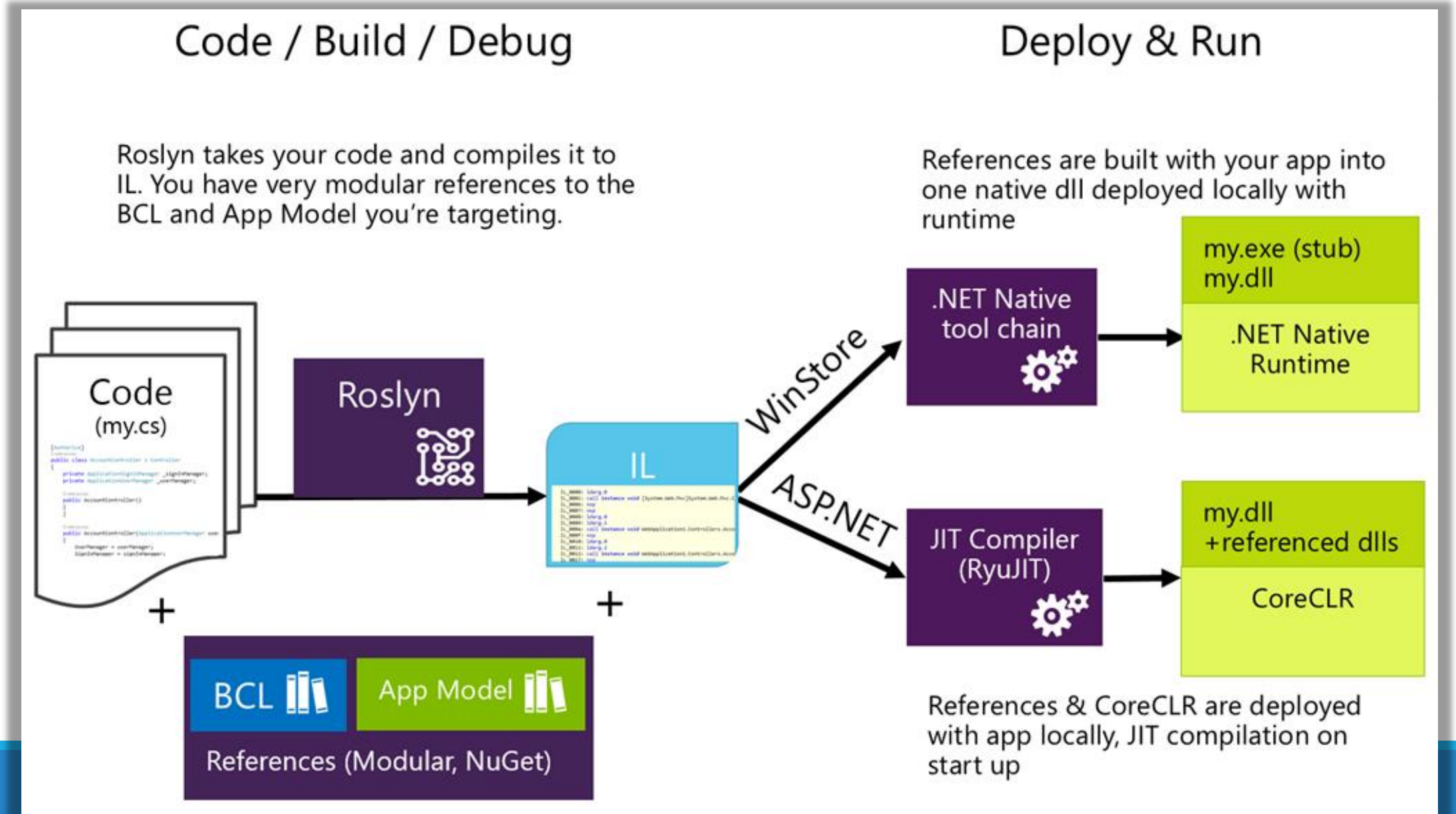


- XAML Islands - WinForms & WPF apps can host UWP controls
- Full access to Window's 10 APIs
- Side-by-side support & self contained exes
- Desktop pack to enable porting existing apps to .NET Core

ASP.NET Core Overview



Processo de Compilação



ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client Factory

ANCM in-process mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Virtual Auth Schemes

Build-time Razor Compilation

New Razor SDK

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: What's New?

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

- Real-time web development
- New Typescript client, ~~jQuery~~
- Built-in Hub protocols:
  - JSON-based for text
  - MessagePack for binary
- Improved scale-out model
  - Sticky sessions required*

**required when using WebSockets unless skipNegotiation flag is set to true*

```
var connection = new
signalR.HubConnectionBuilder().withUrl("/chat",
{
  skipNegotiation: true,
  transport: signalR.HttpTransportType.WebSockets
})
.build();
```

Identity as UI Library

Virtual Au… Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

- UseHttpsRedirection by default
- HSTS protocol support (non-dev)
- >dotnet dev-certs https --trust

ActionResult<T>

M in-process mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Build-time Razor Compilation

New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

- New ApiController attribute
- Auto model validation

Swagger support

ActionResult<T>

HTTP Client Factory

ess mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Virtual Auth Schemes

Build-time Razor Compilation

New Razor SDK

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

- Rich Swagger support
- Easier API documentation

ActionResult<T>

HTTP Client Factory

mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Virtual Auth Schemes

Build-time Razor Compilation

New Razor SDK

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client Factory

ANCM in-process mode for IIS performance

- New type ActionResult<T>
- Indicate response type for any action result

ASP Identi Library

time Razor npilation

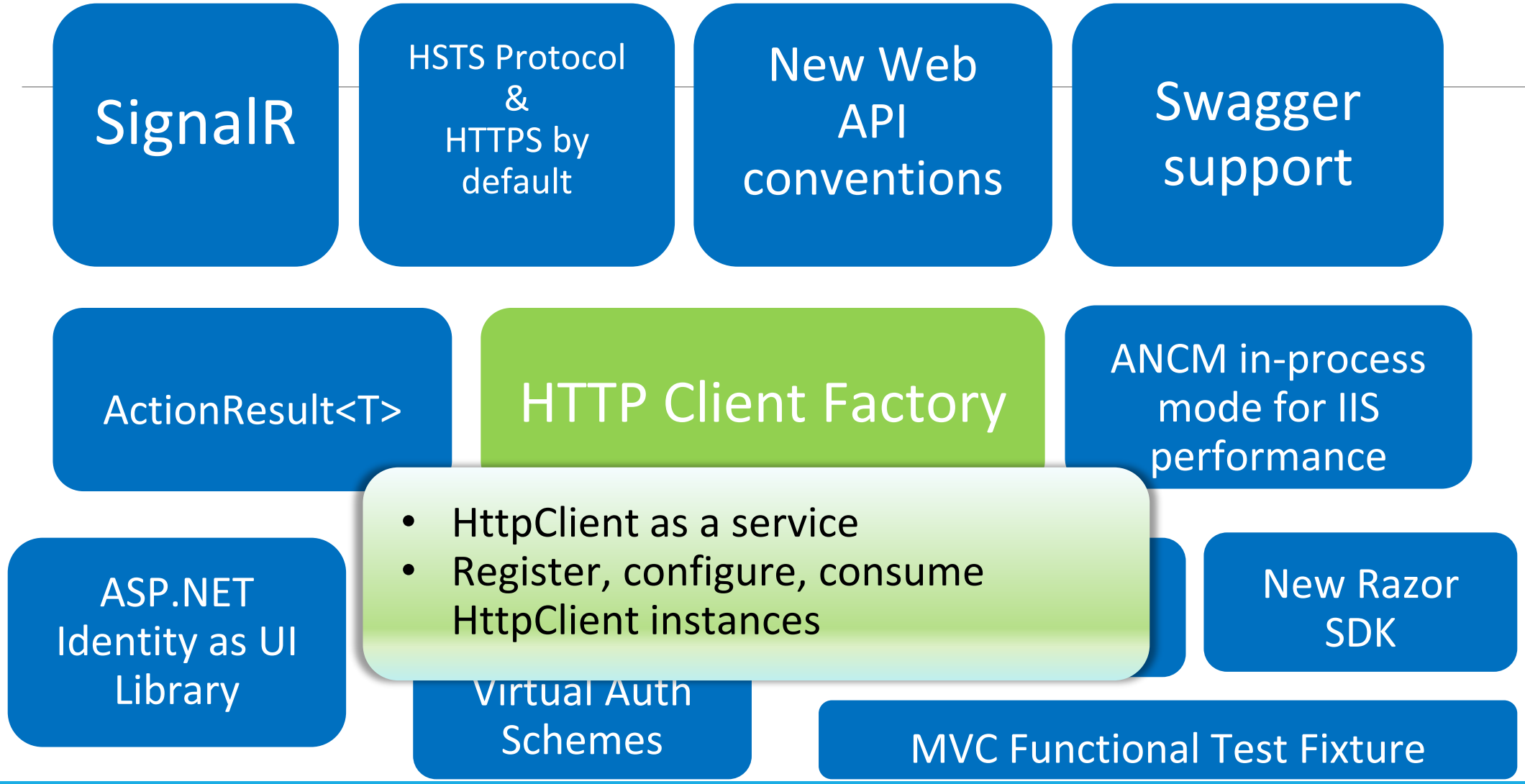
New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos



http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client Factory

ANCM in-process mode for IIS performance

- ASP.NET Core (native IIS) Module
- Hooks into IIS pipeline
- Improved Performance

ASP.NET Identity as UI Library

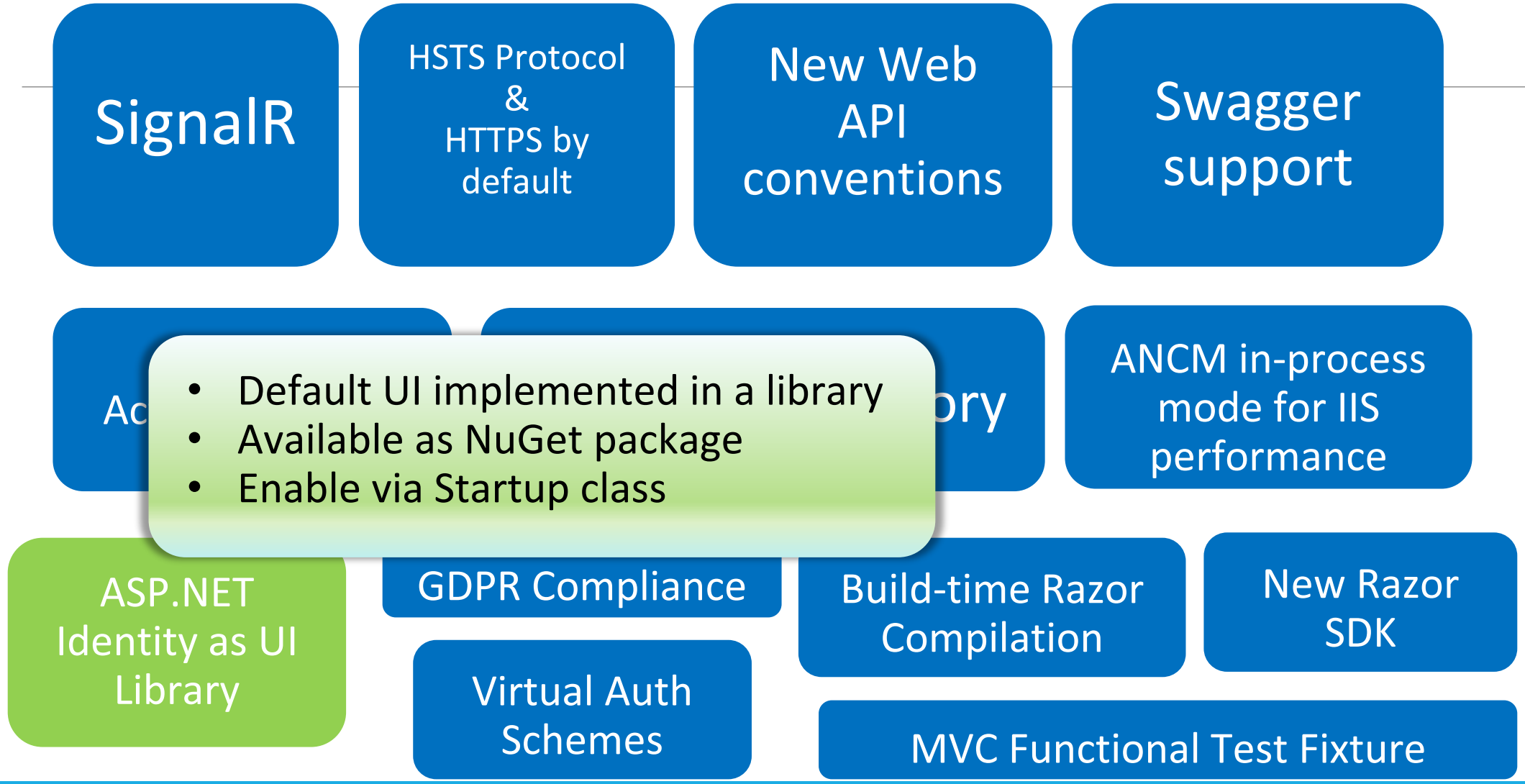
GDPR Compliance

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos



http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: What's New?

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionR

- Compliance for EU General Data Protection Regulation reqts
- Request user consent for info

ANCM in-process mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Build-time Razor Compilation

New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client Factory

ANCM in-process mode for IIS performance

- Mix authentication schemes
- e.g. Bearer tokens, cookie auth

ASP.NET Identity as Library

Razor tion

New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTT

ANCM in-process

- Compiled during build process
- Improved startup performance

ASP.NET Identity as UI Library

GDPR Compliance

Build-time Razor Compilation

New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client F

ANCM in-process

- Razor UI as class library
- Share across projects
- Share as Nuget package

ASP.NET Identity as UI Library

GDPR Compliance

Build-time Razor Compilation

New Razor SDK

Virtual Auth Schemes

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

ASP.NET Core 2.1: Recursos

SignalR

HSTS Protocol & HTTPS by default

New Web API conventions

Swagger support

ActionResult<T>

HTTP Client Factory

ANCM in-process mode for IIS performance

ASP.NET Identity as UI Library

GDPR Compliance

Virtual Auth Schemes

- Improved end-to-end testing
- e.g. routing, filters, controllers, actions, views and pages

MVC Functional Test Fixture

http://www.talkingdotnet.com/quick-summary-whats-new-asp-net-core-2-1/

Entity Framework Core

Data access / ORM investments in .NET

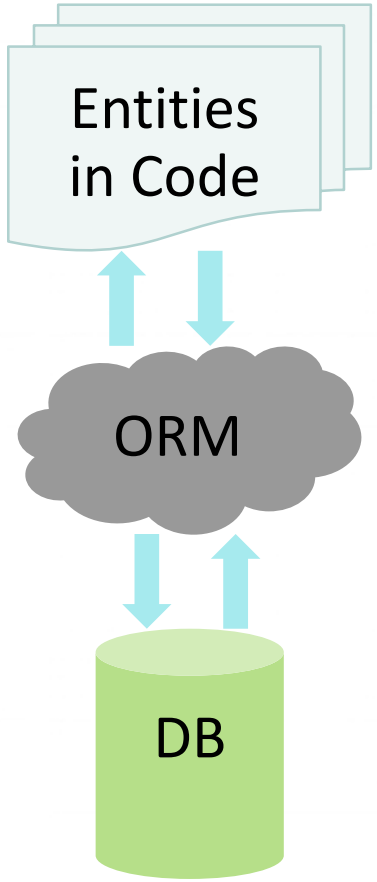
Entity Framework 6.x

- Evolution of current EF
- Runs on .NET Framework 4.6+

Entity Framework Core

- LINQ and EF skills re-use, easy to move most existing apps
- Cloud optimized (small memory footprint and high throughput)
- Device optimized (lightweight, works with SQLite)
- Supports relational and non-relational stores
- Cross-platform ready (based on .NET Core)
- Design for disconnected/web scenarios
- Runs on .NET Core and .NET Framework 4.6+

Both versions are OPEN SOURCE!



EF Core 2.1

- **Basic Lazy Loading** –APIs and patterns for lazy loading navigations
- **LINQ GroupBy Translation** – GroupBy queries translated to SQL
- **Value Conversions** – Flexible mapping of primitive data types
- **Query Types** – Map CLR types without identity
- **System.Transactions** – Ambient transactions with TransactionScope
- **Data Seeding** – Seed data in the model and migrations
- **Column Ordering**- Better column ordering when creating tables
- **Azure Cosmos DB** – Use EF with Cosmos DB

Ambiente

Instalação no Windows

Instalação no Linux

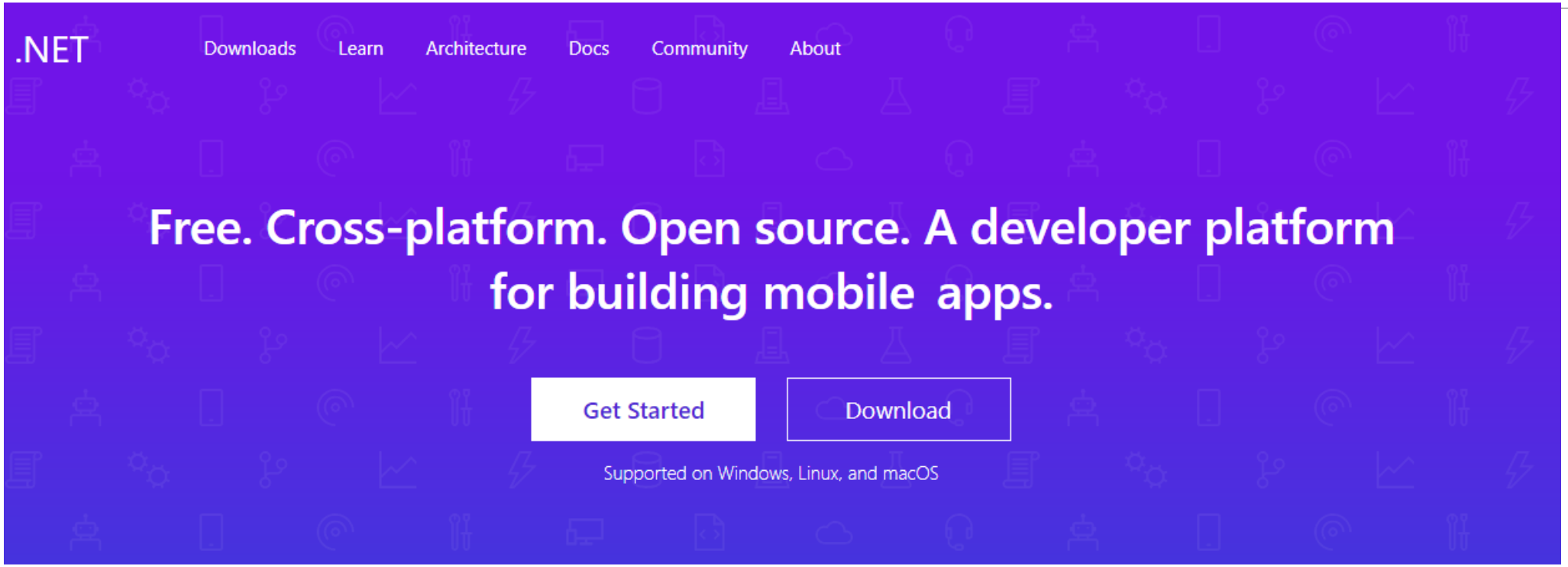
Instalação no Mac

Gerenciando versões

Vamos navegar para..

https://www.microsoft.com/net/

Microsoft



Web

Build web apps and services for

Mobile

Use a single codebase to build

Desktop

Create beautiful and compelling

Not sure where to start? See Get started with .NET in 10 minutes.

Windows | Linux | macOS

.NET Core

.NET Core 2.1

.NET Core is a cross-platform version of .NET for building websites, services, and console apps.

Build Apps — Download .NET Core SDK

Run Apps — Download .NET Core Runtime

.NET Framework

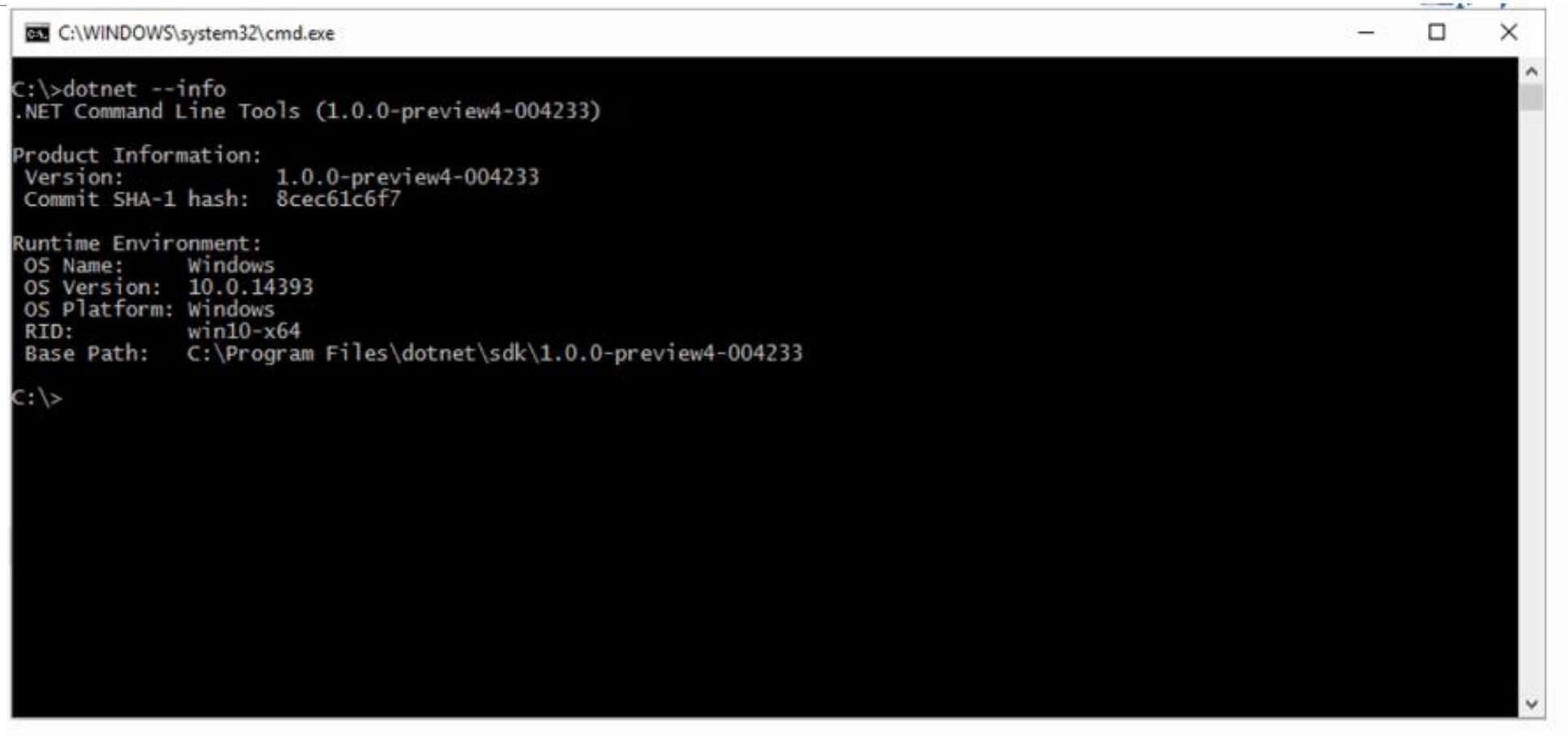
.NET Framework 4.7.2

.NET Framework is a Windows-only version of .NET for building any type of app that runs on Windows.

Build Apps — Download .NET Framework Dev Pack

Run Apps — Download .NET Framework Runtime

Gerenciando Versões



Criando aplicação ASP.NET Core

Setup

Novo Projeto

Pacotes NuGet

Bower

Configuração

Hello World

Startup e Middlewares

- O que é um middleware
- OWIN
- Como funciona um middleware
- Entendendo as interfaces
  - IHostEnvironment
  - IConfigurationRoot
  - IServiceCollection
  - IApplicationBuilder
  - ILoggerFactory
- Configurando o ASP.NET MVC Middleware
- Acertando os Middlewares para o ambiente
- Trabalhando com exceções
- Servindo arquivos
- Como um request é processado

O que é um middleware

Middlewares são componentes de software em uma aplicação ASP.NET Core

Estes componentes manipulam dados entre requests e responses

Um middleware possui uma responsabilidade e pode trabalhar lado a lado com outros middlewares. Quando falamos do pipeline do ASP.NET Core estamos falando basicamente de middlewares

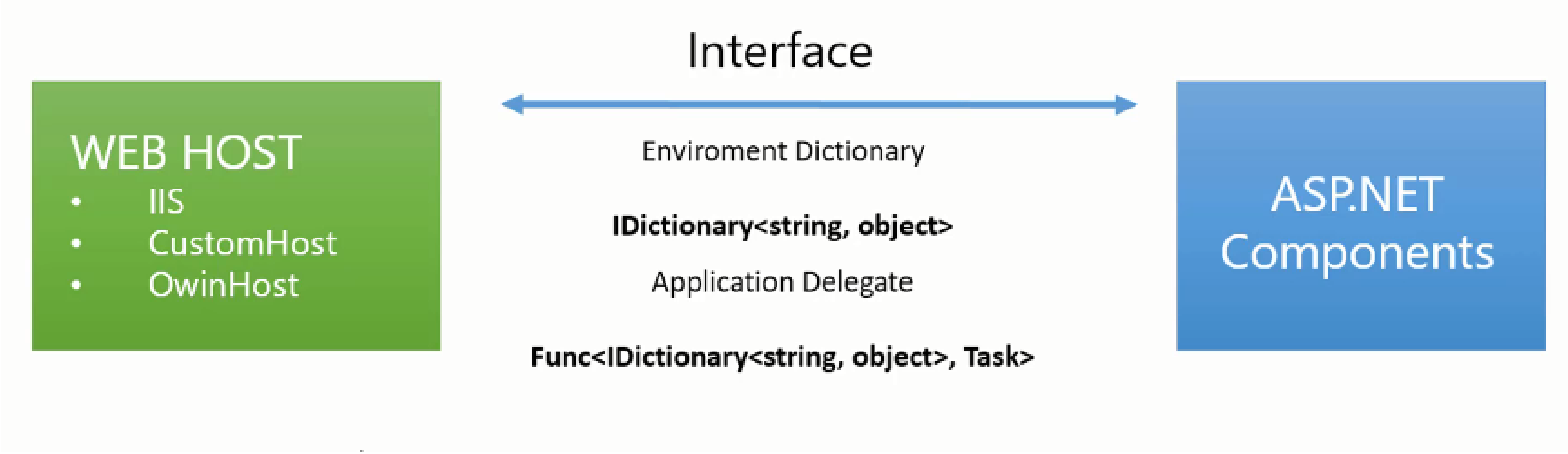
O que é um Middleware

OWIN – Open Web Interface for .NET

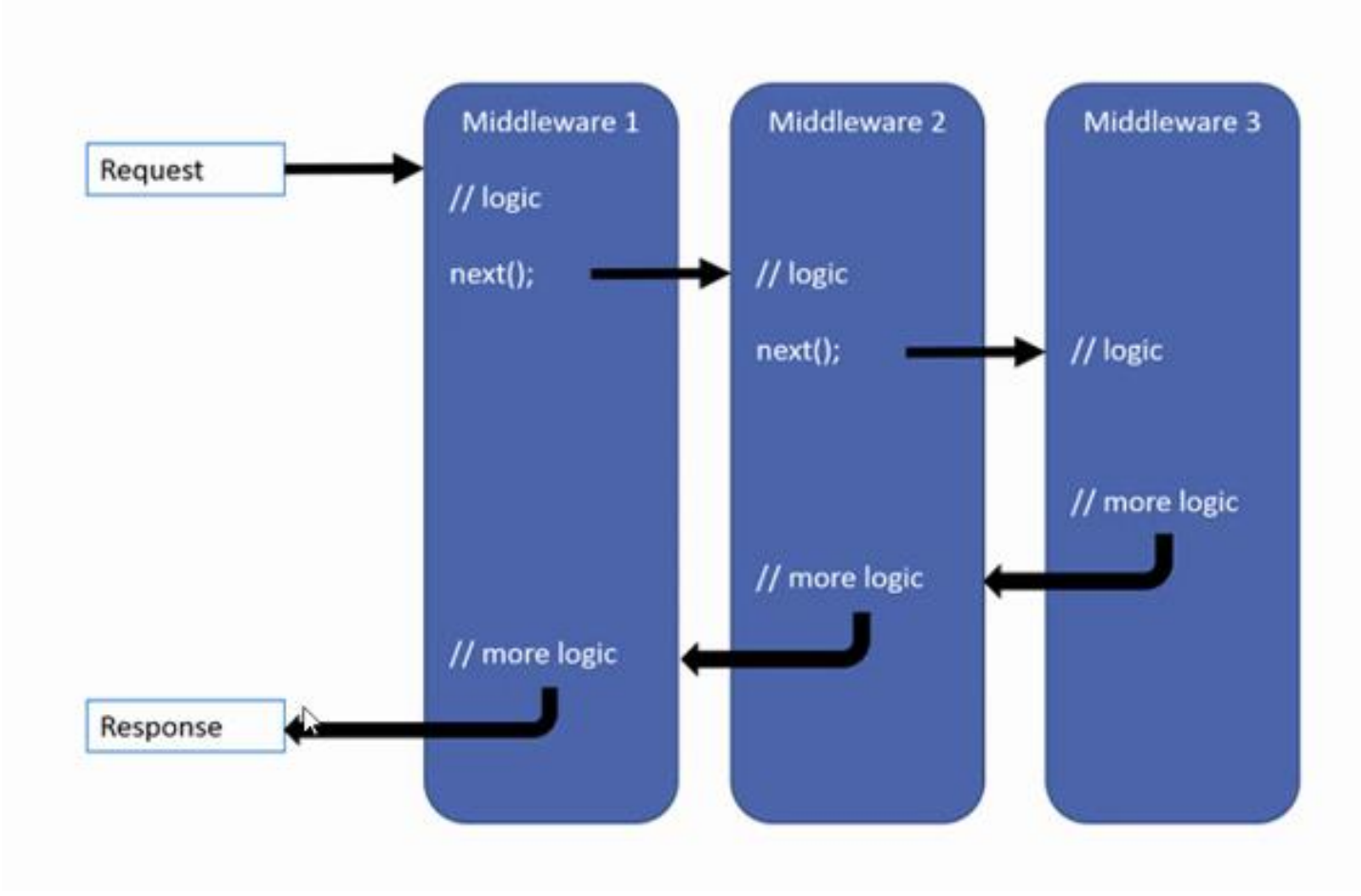
http://owin.org/html/spec/owin-1.0.html

OWIN – Open Web Interface for .NET

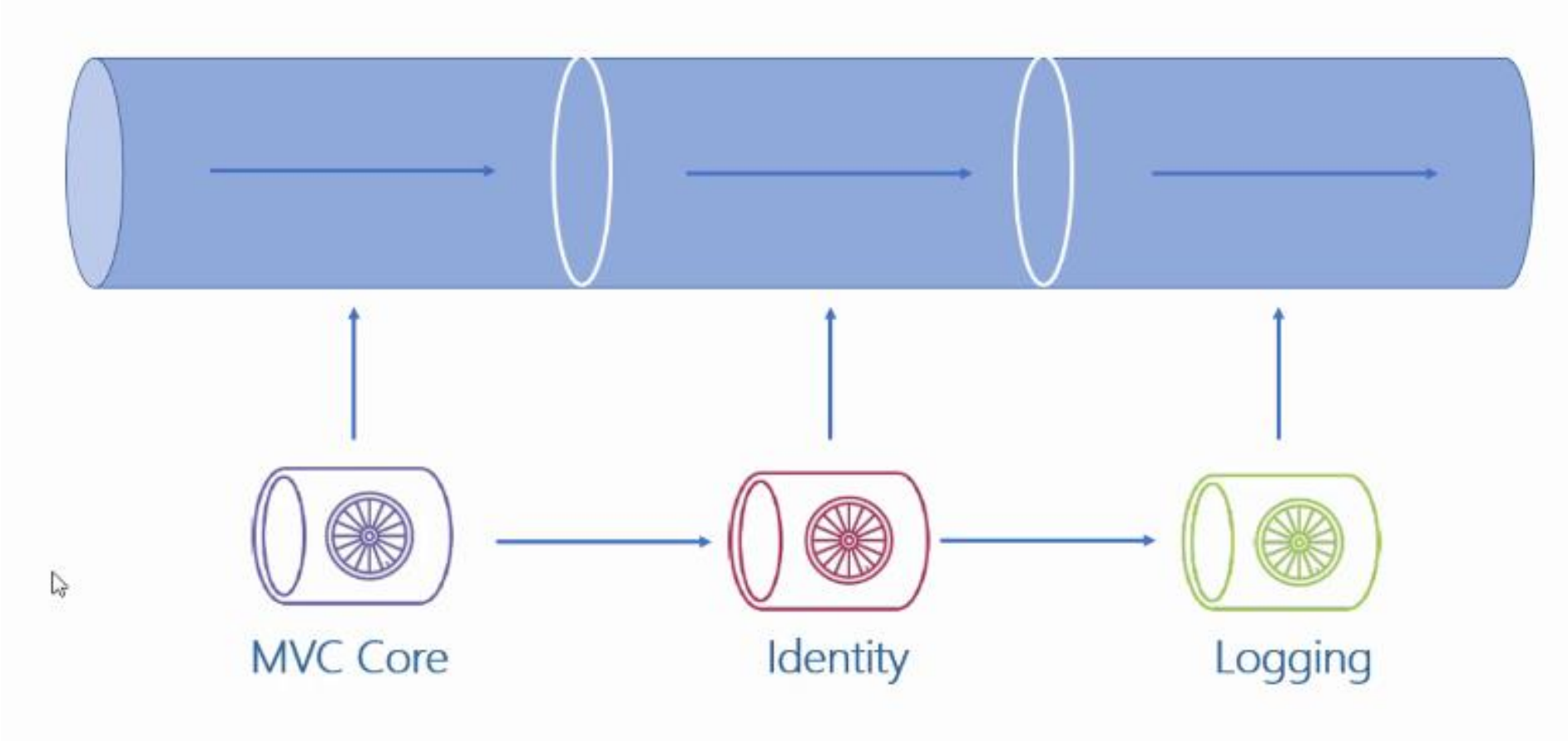
http://owin.org/html/spec/owin-1.0.html



Como funciona um Middleware



Como funciona um Middleware



ENTENDENDO AS INTERFACES

IHostEnvironment

Fornece informações sobre o ambiente de hospedagem Web em que uma aplicação está sendo executada

```
public Startup(IHostingEnvironment env)
{
    var builder = new ConfigurationBuilder()
        .SetBasePath(Directory.GetCurrentDirectory())
        .AddJsonFile("appsettings.json", optional: true, reloadOnChange: true)
        .AddJsonFile($"appsettings.{env.EnvironmentName}.json", optional: true);

    if (env.IsDevelopment())
    {
        builder.AddUserSecrets();
    }

    builder.AddEnvironmentVariables();
    Configuration = builder.Build();
}
```

ENTENDENDO AS INTERFACES

IConfigurationRoot

Representa a raiz de uma hierarquia Microsoft.Extensions.Configuration.IConfiguration

```
public IConfigurationRoot Configuration { get; }

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

ENTENDENDO AS INTERFACES

IServiceCollection

Representa um contrato para uma coleção de serviços. Essa interface é estendida por classes que implementam Middleware através do recurso de injeção de dependências

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

    services.AddIdentity<ApplicationUser, IdentityRole>(options =>
                options.Cookies.ApplicationCookie.AccessDeniedPath = "/home/access-denied")
        .AddEntityFrameworkStores<ApplicationDbContext>()
        .AddDefaultTokenProviders();

    services.AddMvc();
    services.AddAutoMapper();
}
```

ENTENDENDO AS INTERFACES

IApplicationBuilder

Representa um contrato para classes que irão prover mecanismos de configuração de um Middleware. Esta interface é estendida por classes que implementam métodos de configuração para cada Middleware

```
public void Configure(IApplicationBuilder app)
{

    app.UseStaticFiles();

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

ENTENDENDO AS INTERFACES

ILoggerFactory

Representa um tipo usado para configurar o log do sistema e criar instâncias da classe Ilogger. É utilizada com o objetivo de capturar dados para log.

```
public void Configure(ILoggerFactory loggerFactory)
{
    loggerFactory.AddConsole(Configuration.GetSection("Logging"));
    loggerFactory.AddDebug();

}
```

Por dentro da classe Startup e Middlewares

DEMO

ASP.NET MVC Controllers

O padrão MVC

Rotas

Rotas Padrão

Attribute Routes

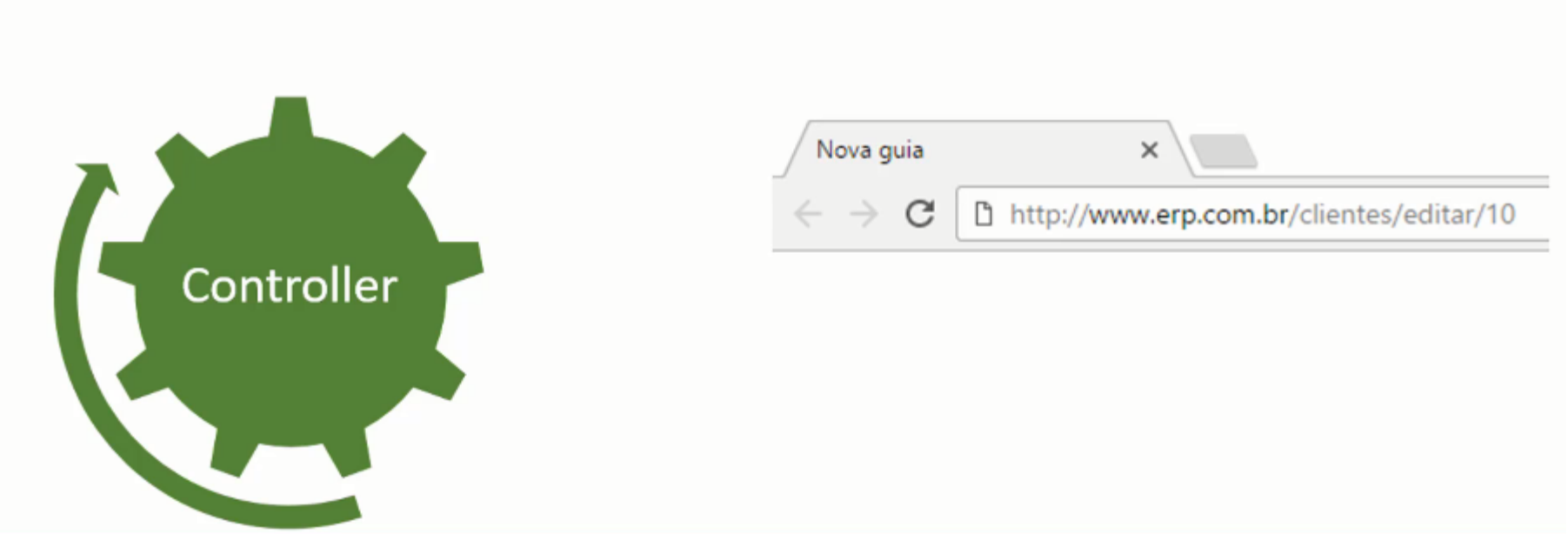
Action Results

GET / POST Actions

Renderizando Views

O padrão MVC

O padrão MVC



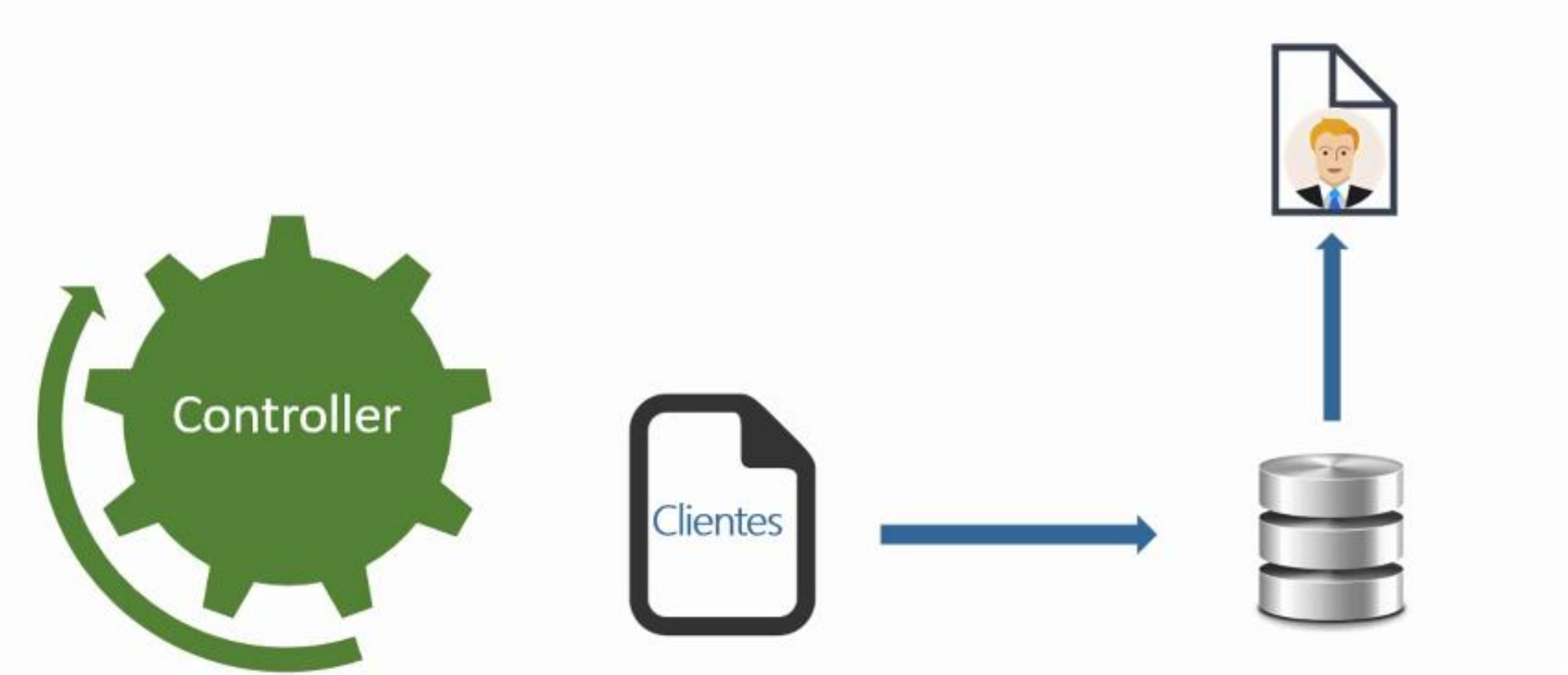
O padrão MVC


Controller
Clientes
Nova guia
http://www.erp.com.br/clientes/editar/10

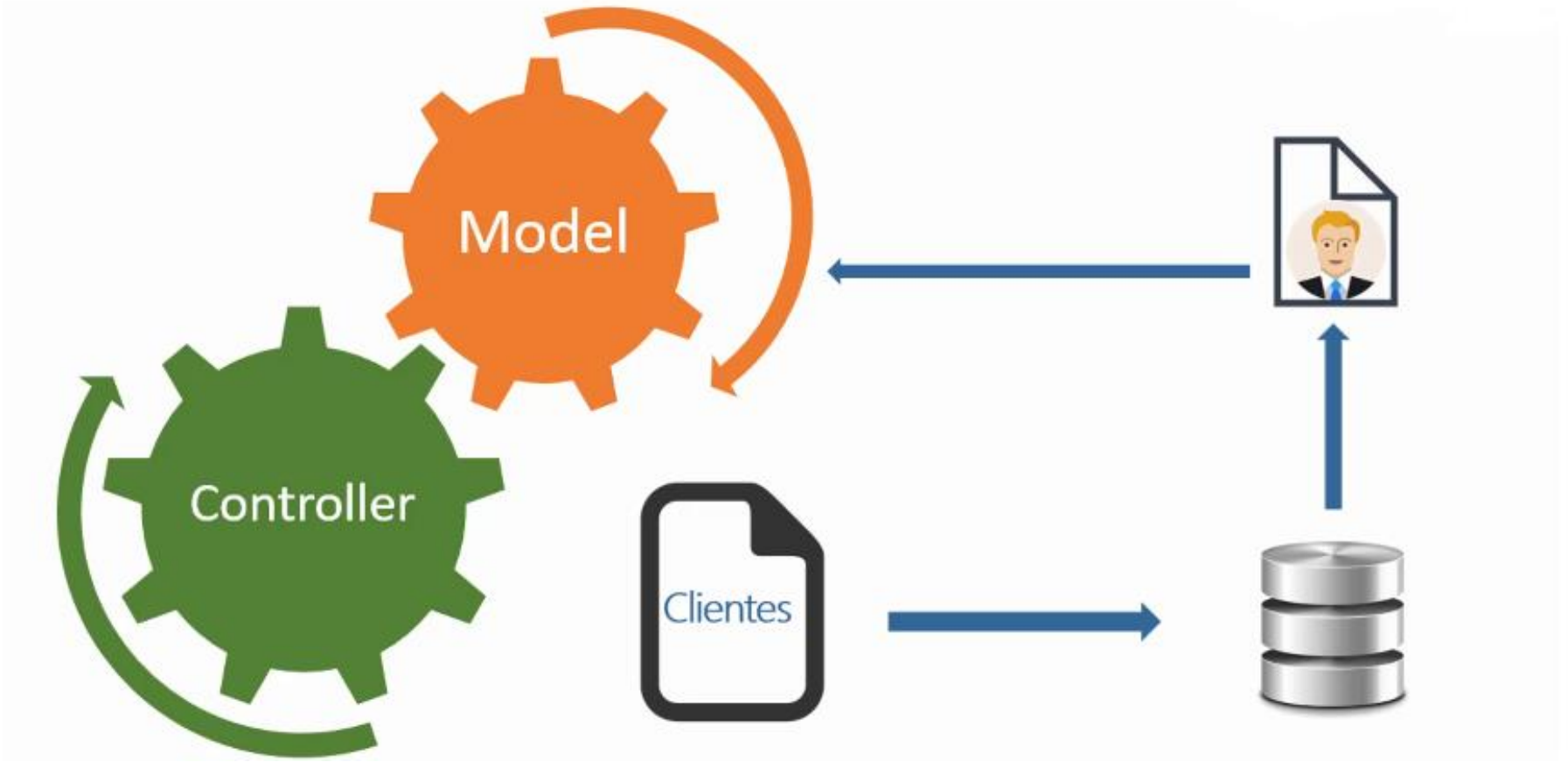
O padrão MVC


Controller
Nova guia
http://www.erp.com.br/clientes/editar/10
Clientes

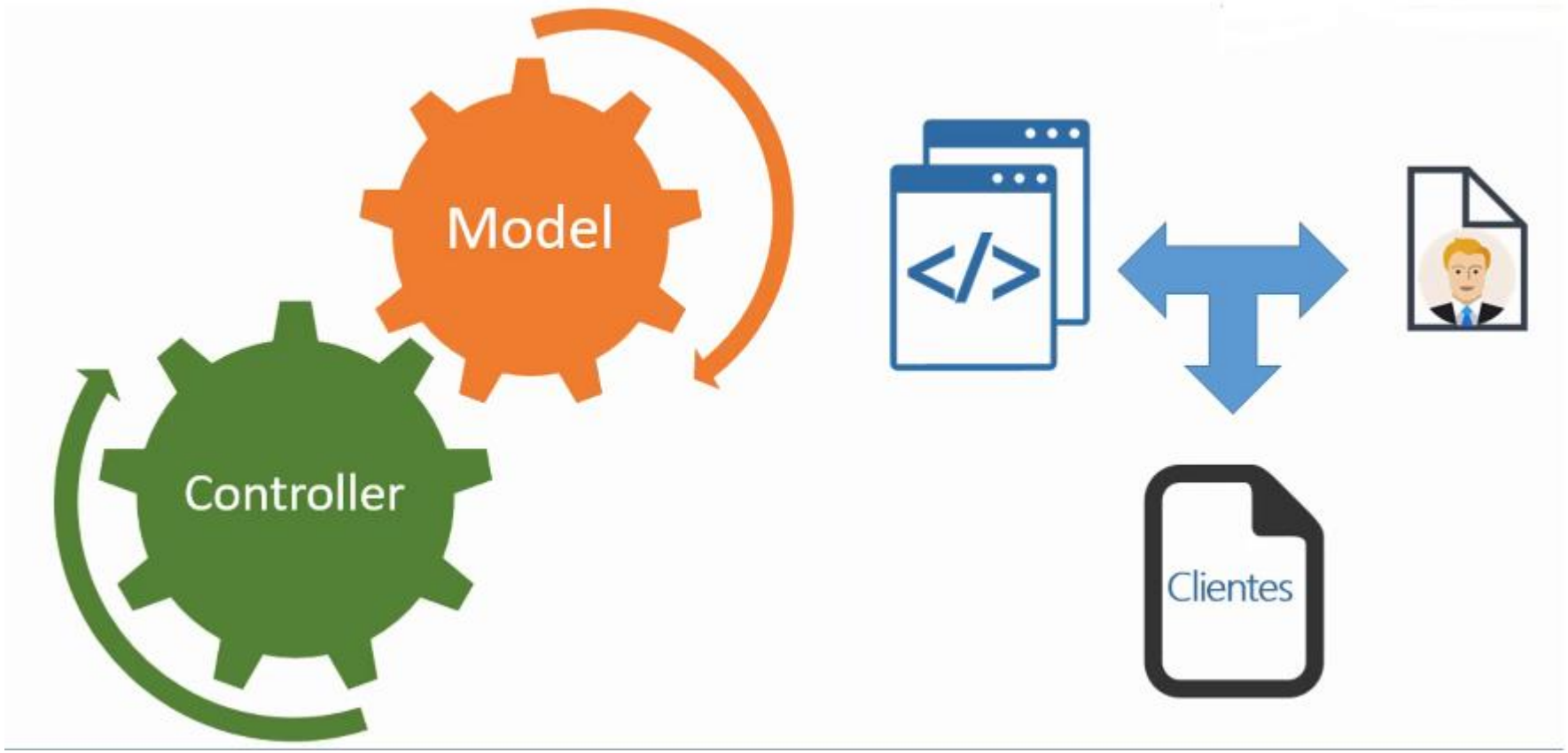
O padrão MVC



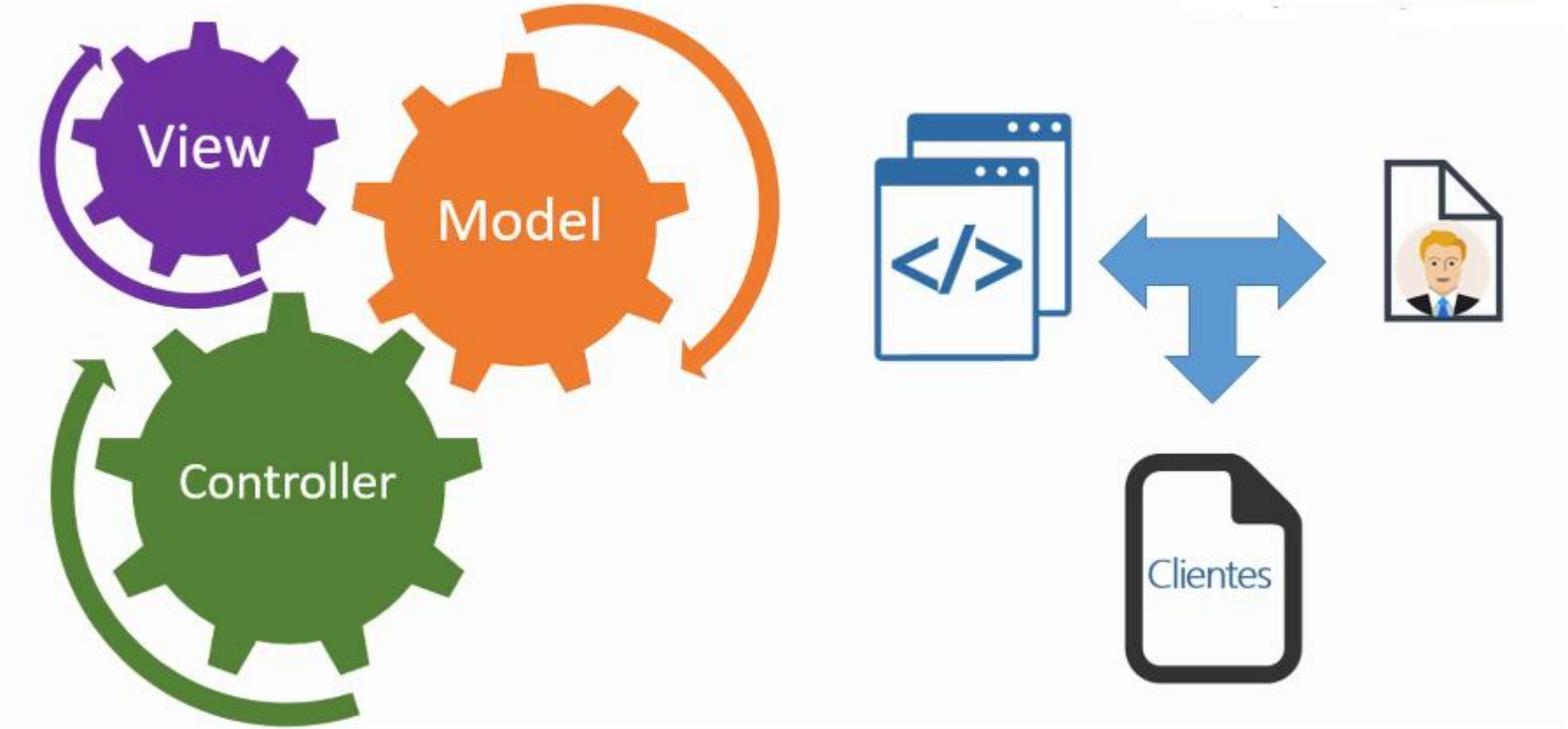
O padrão MVC



O padrão MVC



O padrão MVC



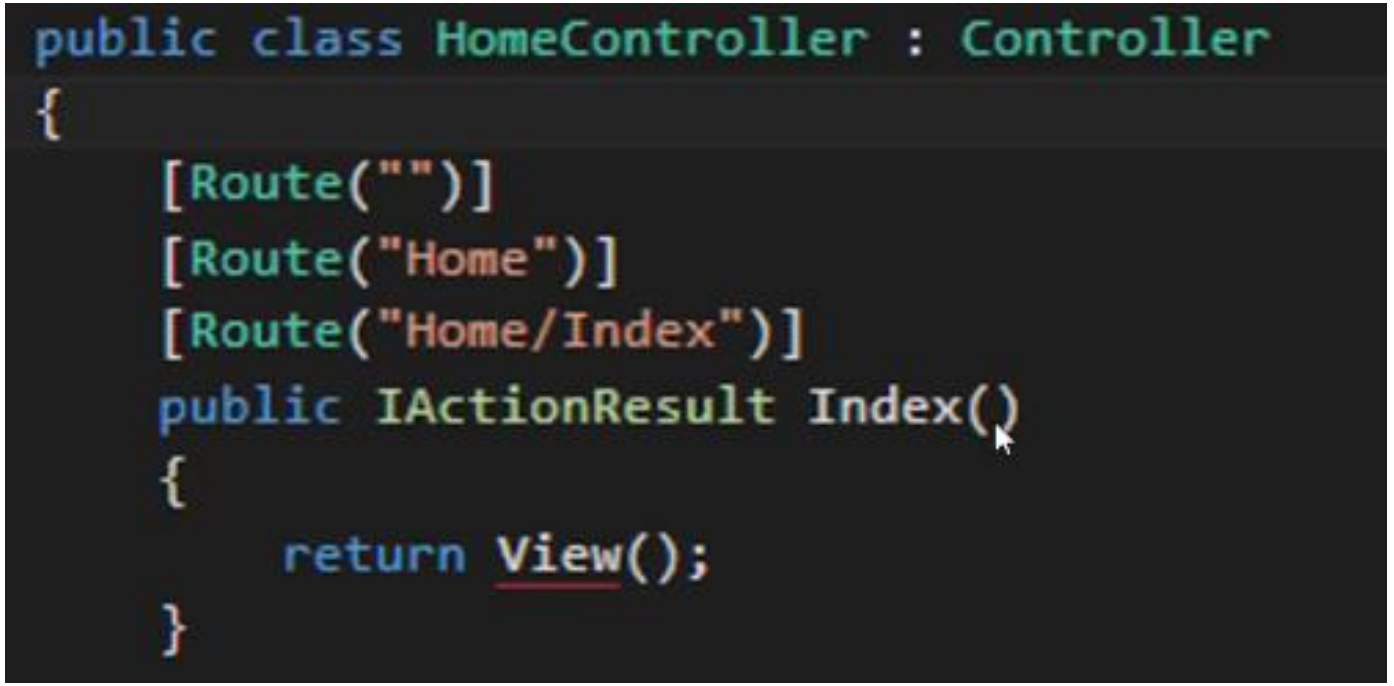
Rotas

Rotas são basicamente estruturas de navegação personalizadas para que a URL da aplicação possua determinado padrão e atenda as necessidades da passagem de parametros

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

Attribute Routes

Rotas por atributos é uma maneira alternativa de trabalhar com rotas, são muito mais flexíveis e fáceis de personalizar. As rotas valem apenas para a controller a qual foi configurada

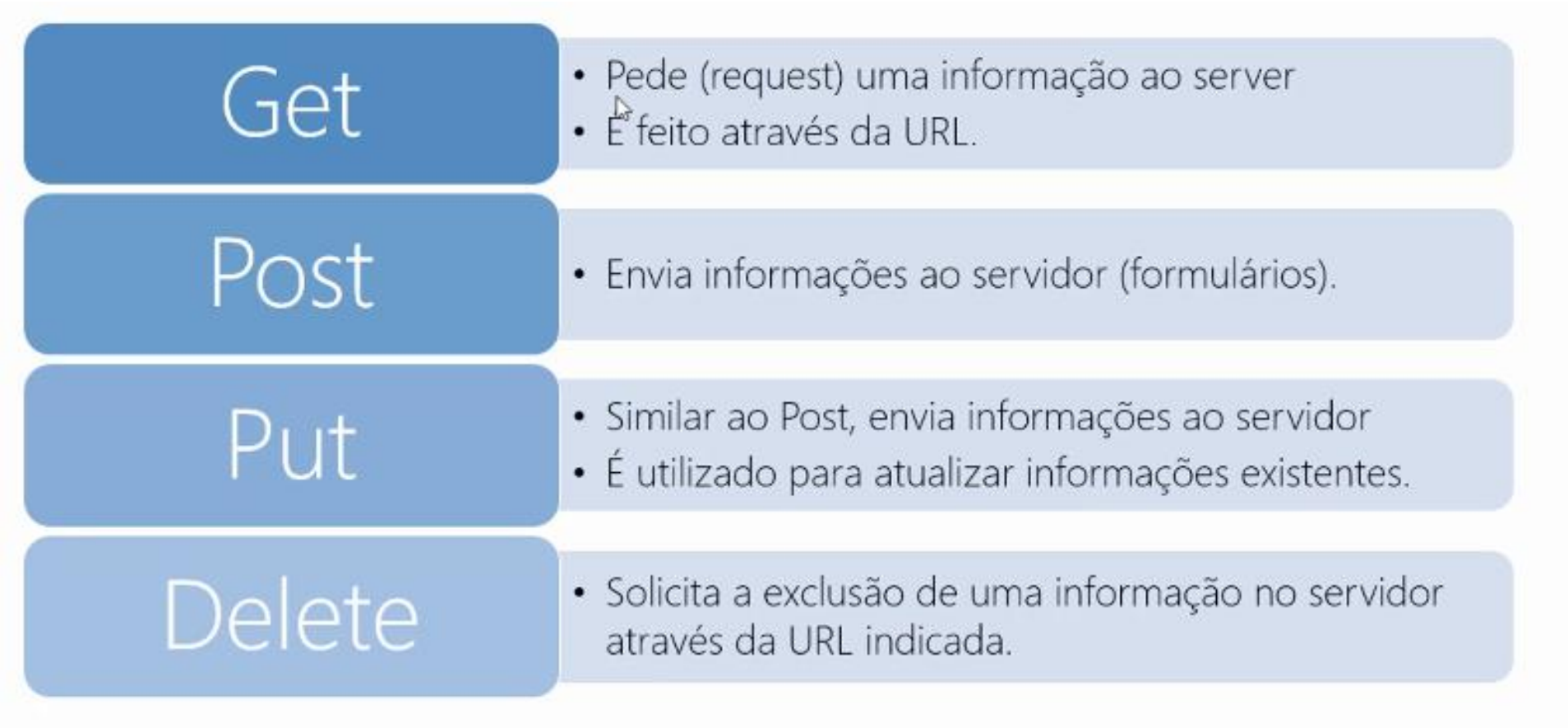


Action Results

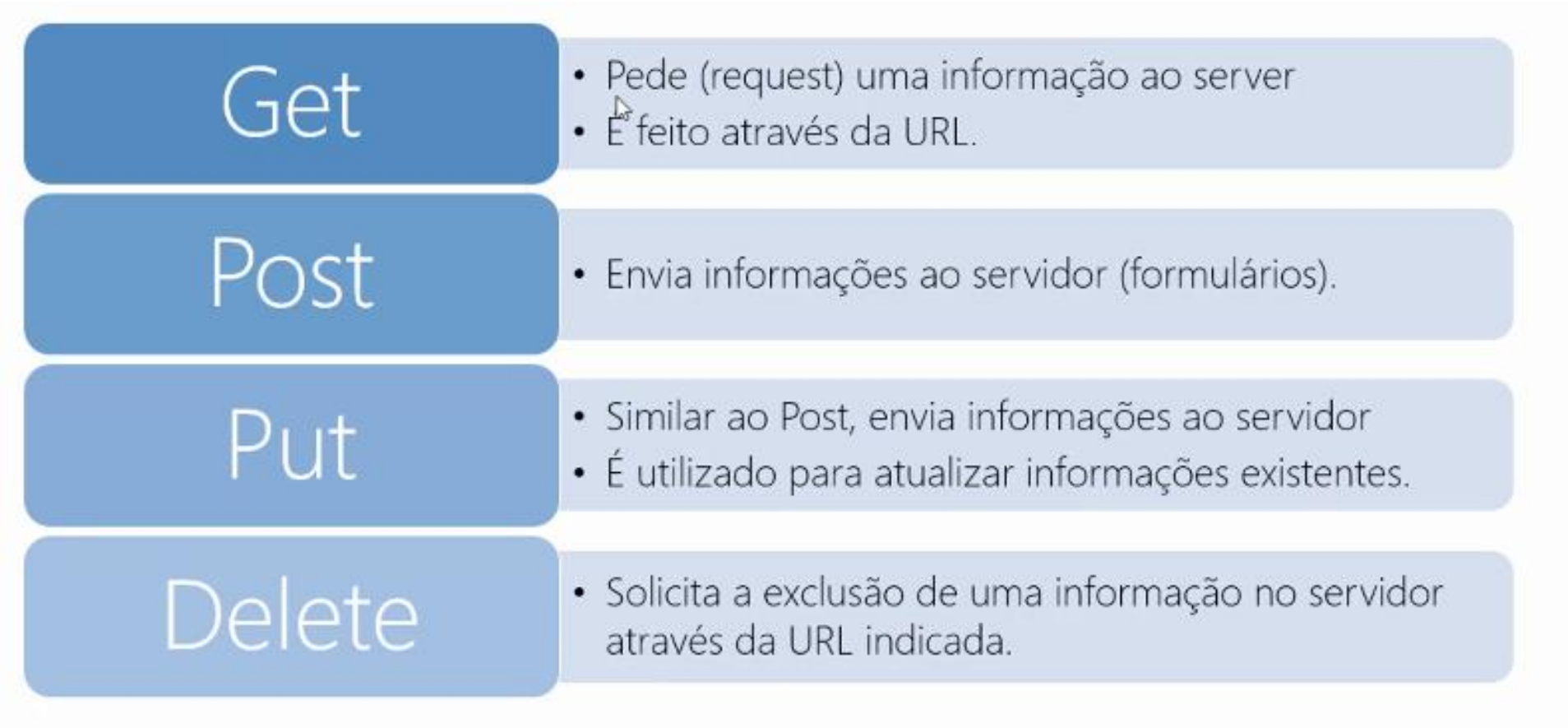
No ASP.NET Core um Action Result é o tipo de retorno da action da controller e é utilizada a interface IActionResult que pode retornar alguns tipos de resultados

JsonResult	BadRequestResult	CreatedAtActionResult
PartialViewResult	BadRequestObjectResult	FileContentResult
ViewResult	ChallengeResult	FileStreamResult
ViewComponentResult	ContentResult	VirtualFileResult
RedirectResult	CreatedResult	ForbidResult
LocalRedirectResult	NoContentResult	NotFoundResult
NotFoundObjectResult	OkResult	OkObjectResult
PhysicalFileResult	RedirectToActionResult	RedirectToRouteResult
SignInResult	SignOutResult	StatusCodeResult
ObjectResult	Task	UnauthorizedResult

HTTP – Verbos Básicos

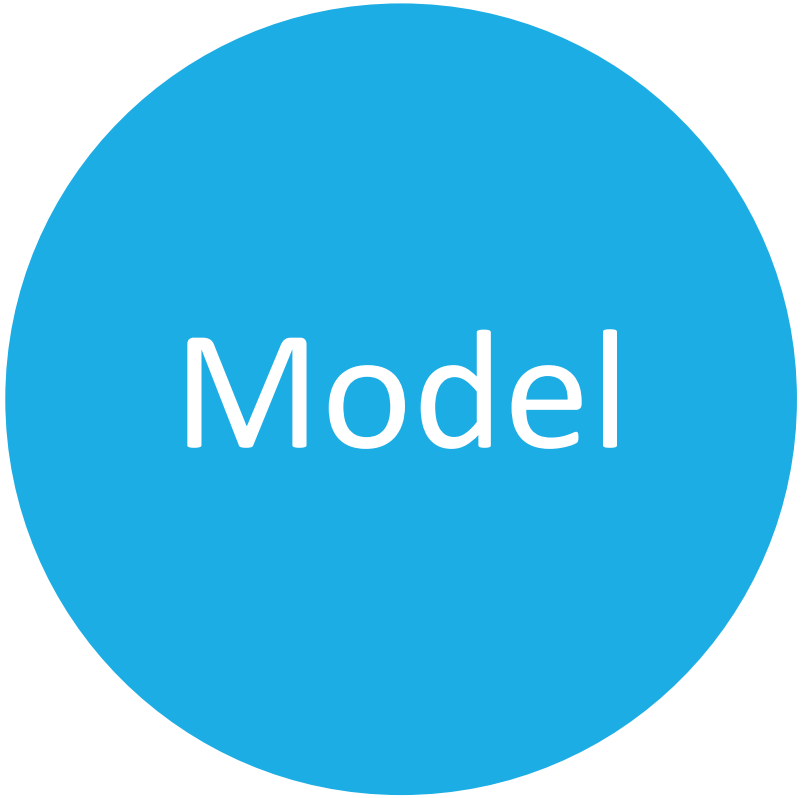


HTTP – Verbos Básicos



DEMO - Visualizar Controllers

Models – o que é um modelo?



No MVC um modelo é a representação de um objeto no mundo real. Na maioria das vezes, este objeto pode representar uma tabela de um banco de dados.

No MVC, um modelo pode ser um conjunto de informações de diversos objetos em um só, esse conceito nós chamamos de DTO (Data Transfer Objects) que são muito utilizados para diminuir o número de requisições no servidor.

Trabalhando com modelos

DEMO - Models

Views

Razor Syntax

Tag Helpers

Custom Tag Helpers

_ViewStart

_ViewImports

Layouts

Formulários

Partial Views

View Components

Razor Views

No MVC, o motor de renderização das Views, logo nós temos as Razor Views que são arquivos HTML mesclados com recursos do Razor.

Os recursos do Razor servem para criar uma experiência mais rica no desenvolvimento de páginas HTML. Pode pensar em Views fortemente tipadas.

O mecanismo Razor transforma as Views em arquivos HTML puros para a interpretação do Browser

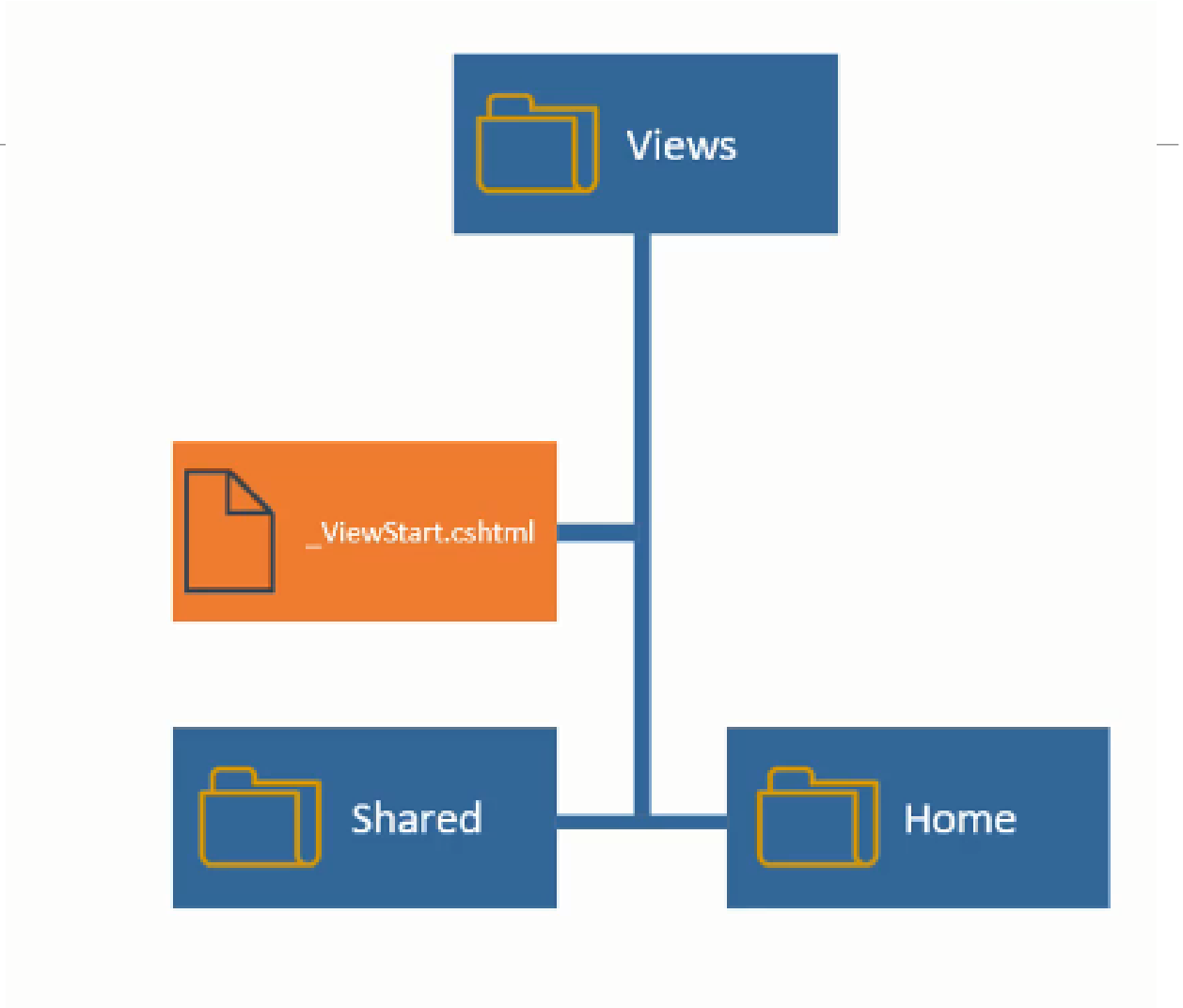
Tag Helpers

Tag Helpers são os recursos do Razor para geração de HTML. No ASP.NET MVC 5 este recurso se chama HTML Helpers e são muito similares.

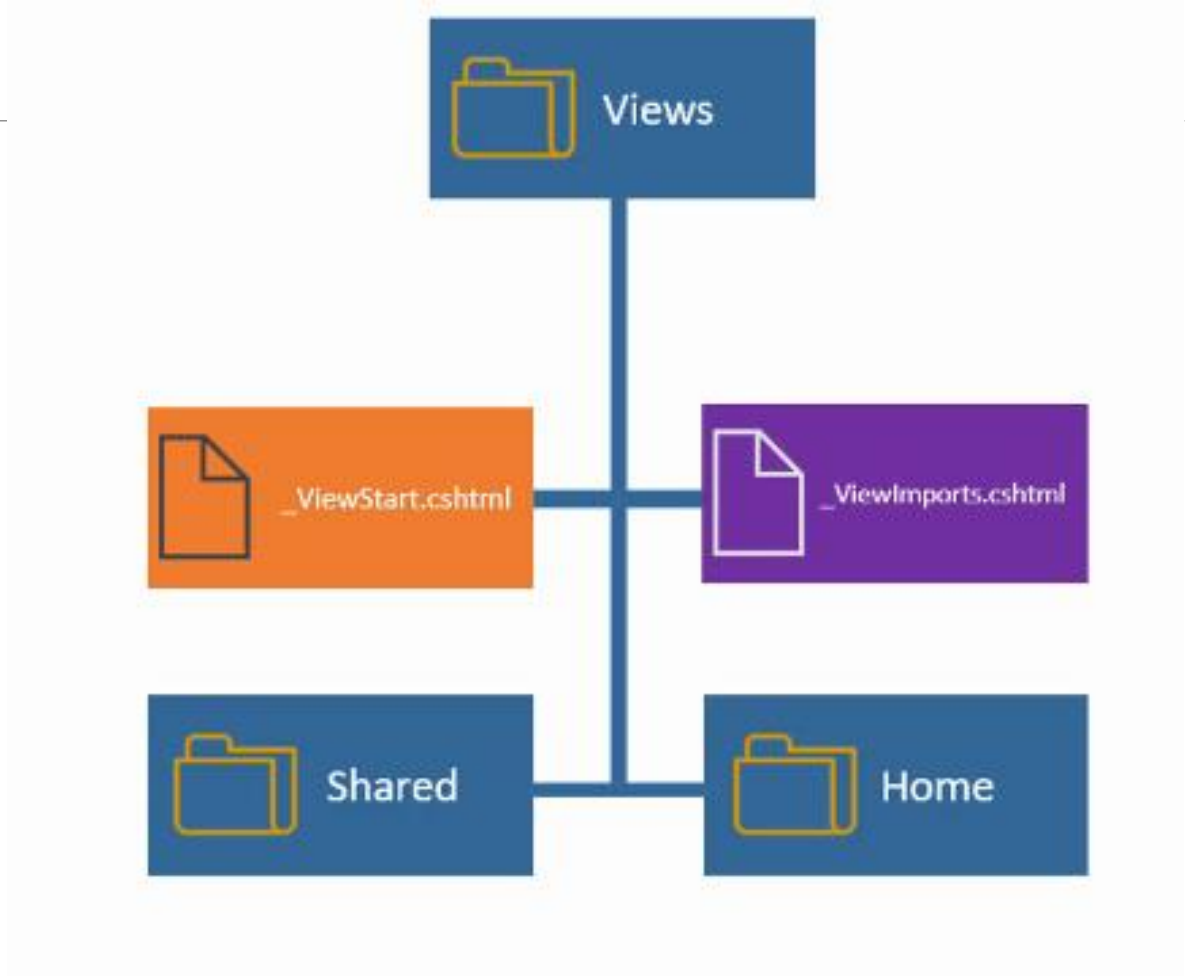
A grande melhoria nos Tag Helpers é a nova sintaxe muito mais agradável e próxima do HTML

```
<div class="form-group">
    <label asp-for="Password" class="col-md-2 control-label"></label>
    <div class="col-md-10">
        <input asp-for="Password" class="form-control" />
        <span asp-validation-for="Password" class="text-danger"></span>
    </div>
</div>
```

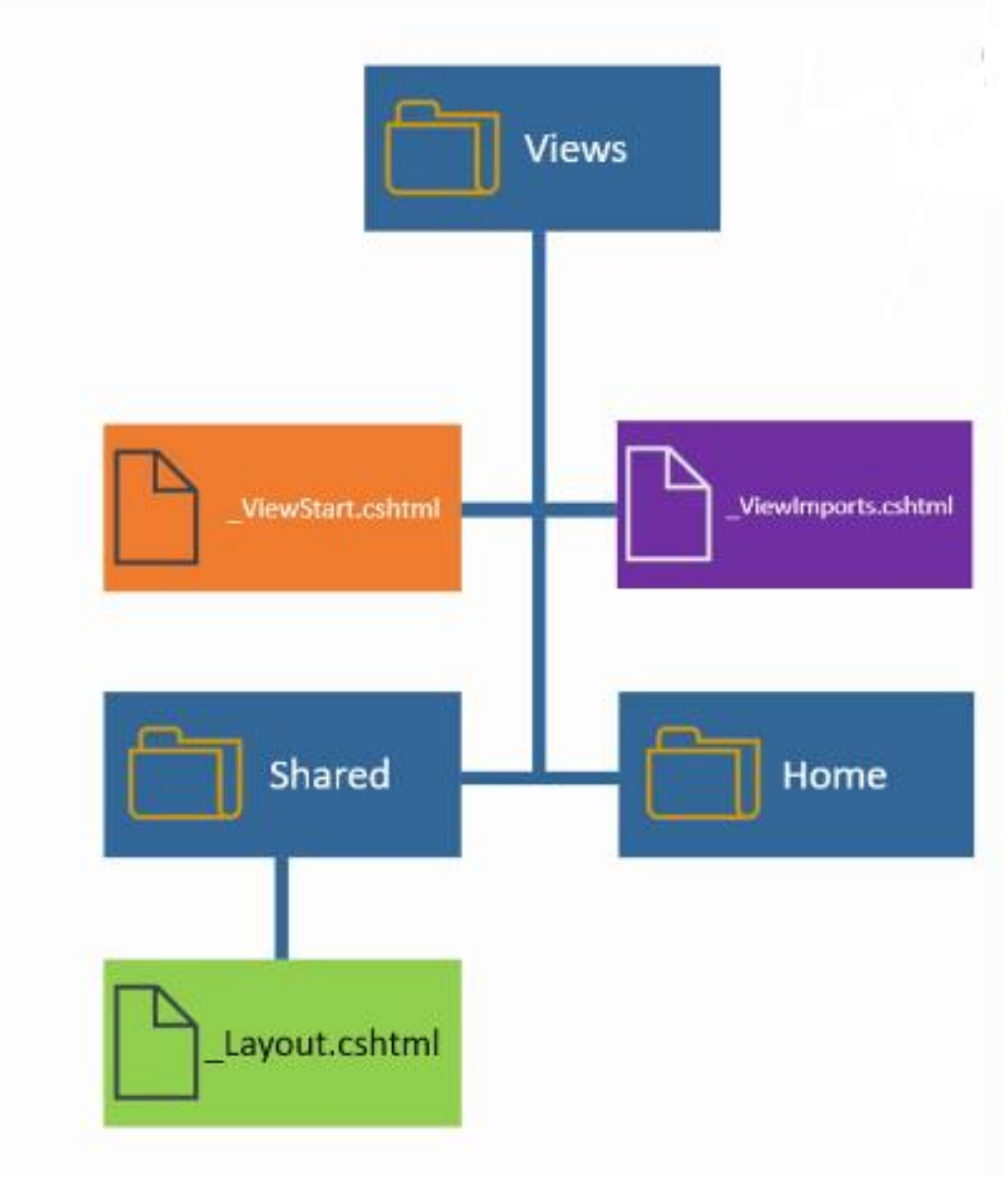
_ViewStart



_ViewImports



_Layout.cshtml



Partial Views

Partial Views são pedações de uma View que possuem dados e que podem ser reaproveitados em N Views, assim proporcionando mais reaproveitamento de código

As Partial Views dependem do modelo implementado na View principal, gerando certa limitação no seu uso.

As Partial Views são muito utilizadas também para renderizar dinamicamente parte de uma View através de requisições AJAX

View Components

View Components é um novo recurso do ASP.NET MVC Core, é um poderoso aliado para desenvolvimento de componentes independentes das Views

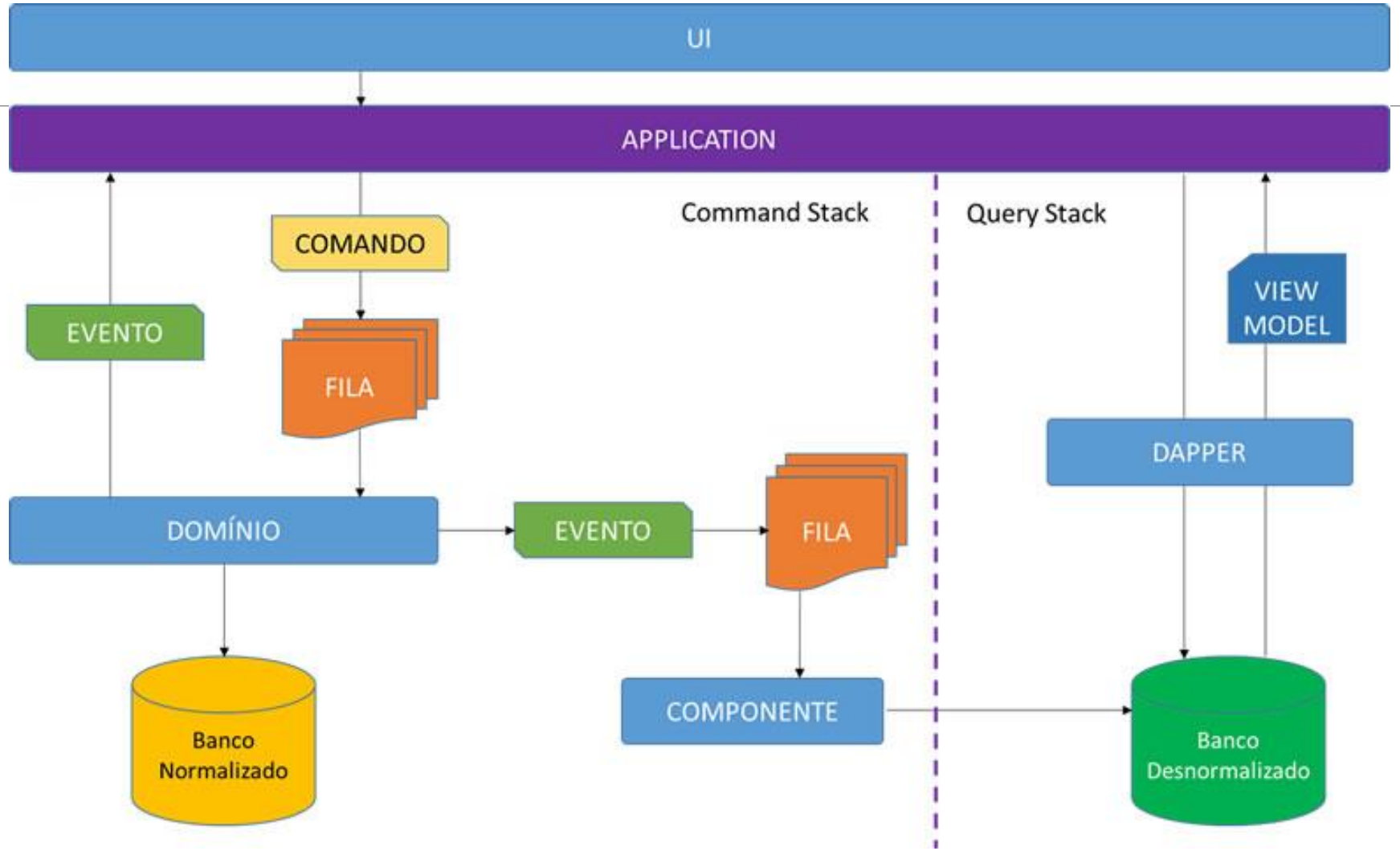
Os View Components possuem processamento server-side independente e podem realizar ações como obter dados de uma tabela e exibir valores manipulados

É um excelente funcionalidade para componentizar recursos de página como um carrinho de compras por ex.

Mais alguns Patterns

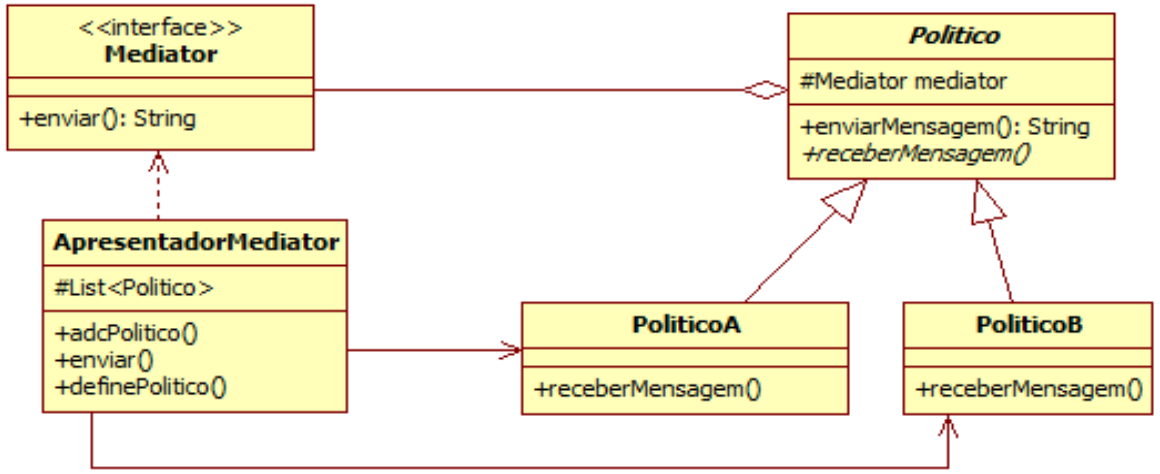
BOAS PRÁTICAS PARA PROJETO FINAL ASP.NET CORE

CQRS


UI
APPLICATION
Command Stack
Query Stack
COMANDO
EVENTO
FILA
VIEW MODEL
DAPPER
DOMÍNIO
EVENTO
FILA
COMPONENTE
Banco Normalizado
Banco Desnormalizado

Pattern Mediator

O Mediator é um padrão de projeto Comportamental criado pelo GoF, que nos ajuda a garantir um baixo acoplamento entre os objetos de nossa aplicação. Ele permite que um objeto se comunique com outros sem saber suas estruturas. Para isso devemos definir um ponto central que irá encapsular como os objetos irão se comunicar uns com os outros.



Pattern Mediator

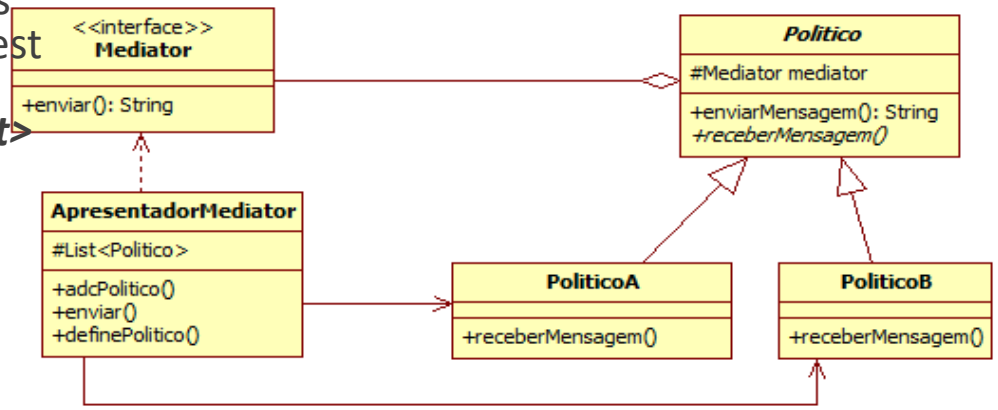
Basicamente temos dois componentes principais chamados de Request e Handler, que implementamos através das interfaces ***IRequest*** e ***IRequestHandler<TRequest>*** respectivamente.

Request → mensagem que será processada.

Handler → responsável por processar determinada(s) mensagen(s). Basicamente temos dois componentes principais chamados de Request e Handler, que implementamos através das interfaces ***IRequest*** e ***IRequestHandler<TRequest>*** respectivamente.

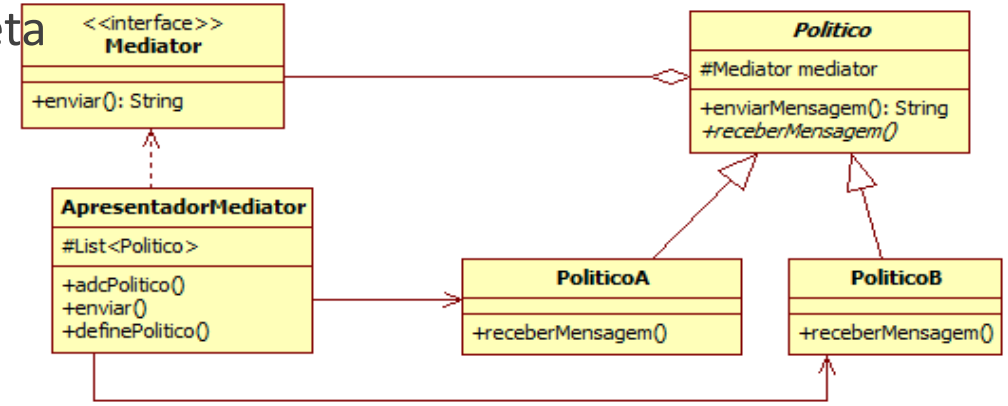
Request → mensagem que será processada.

Handler → responsável por processar determinada(s) mensagen(s).

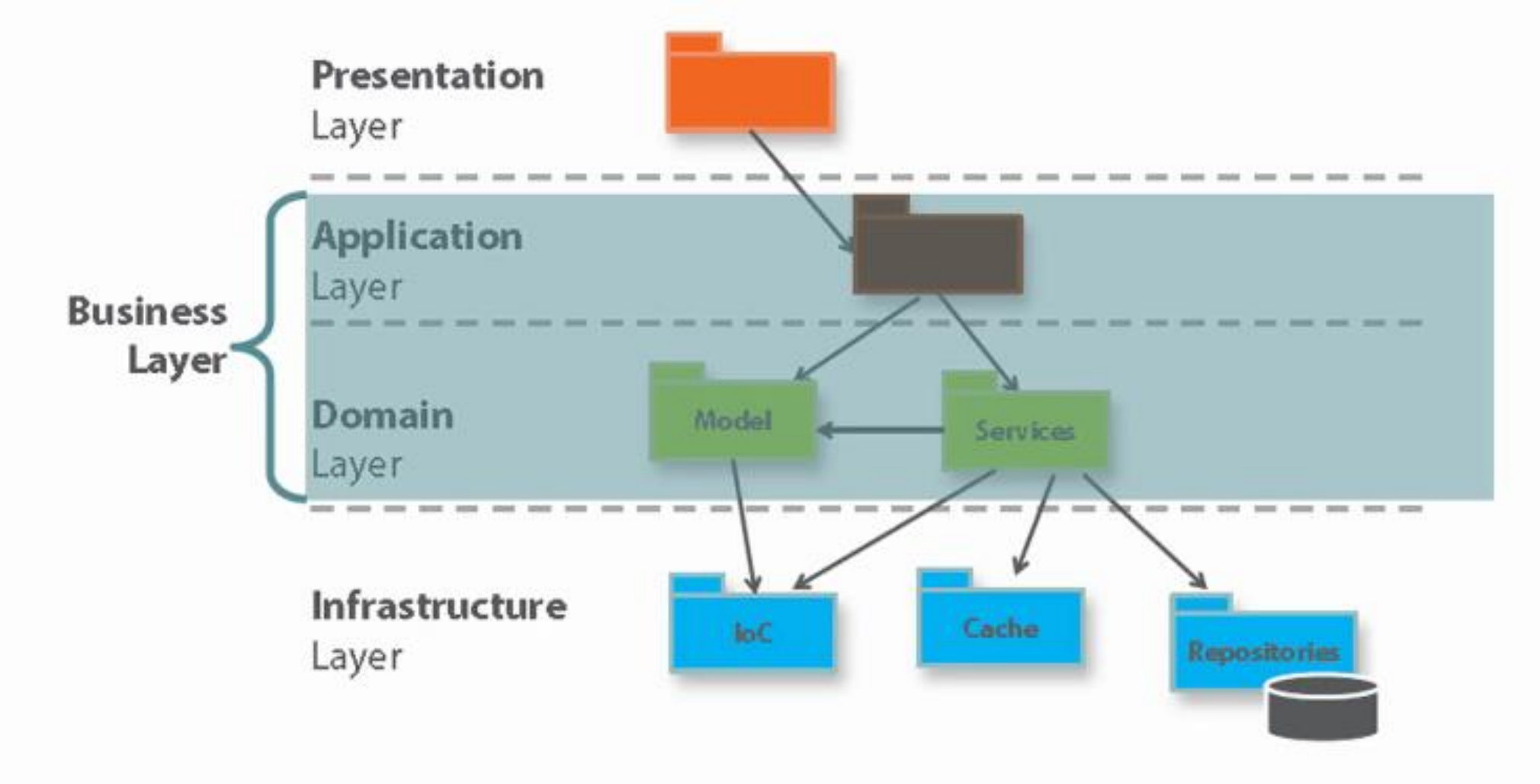


Pattern Mediator

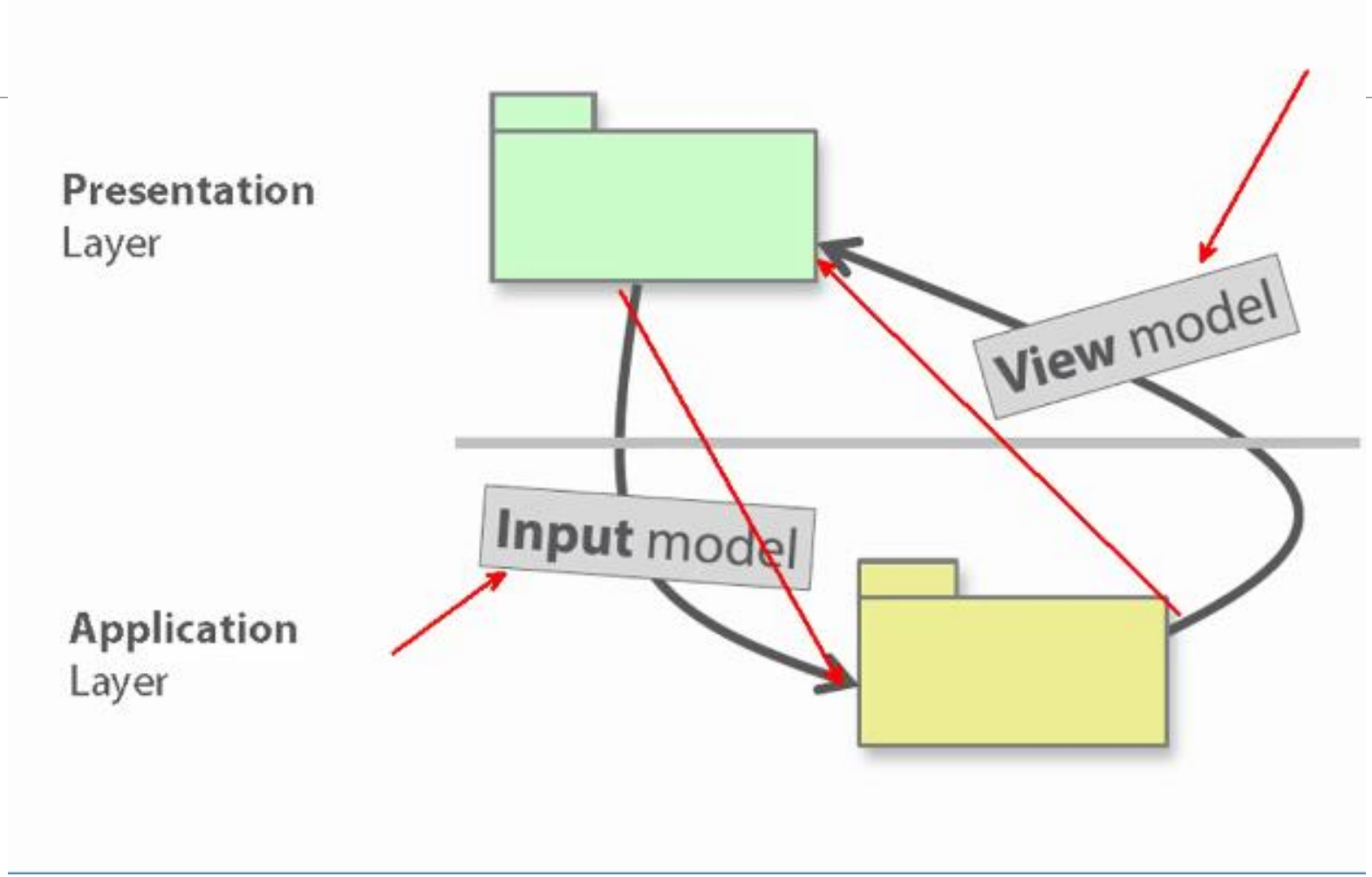
Com isso temos um baixo acoplamento e fácil manutenção. Cada Handler normalmente irá tratar um único Request e assim podemos ter classes menores e mais simples. Entretanto, nada impede de você definir mais de um Request para um Handler, basta implementar mais uma interface ***IRequestHandler*** e fazer a correta injeção de dependências para isso.

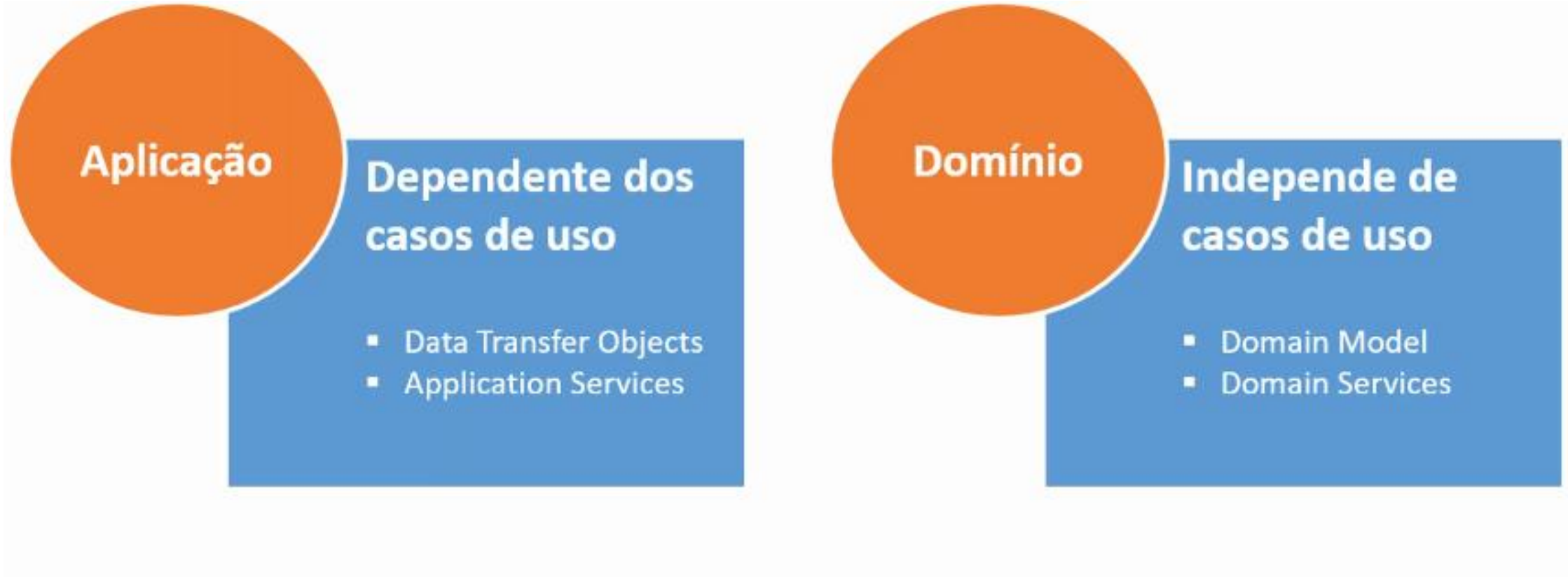


Application

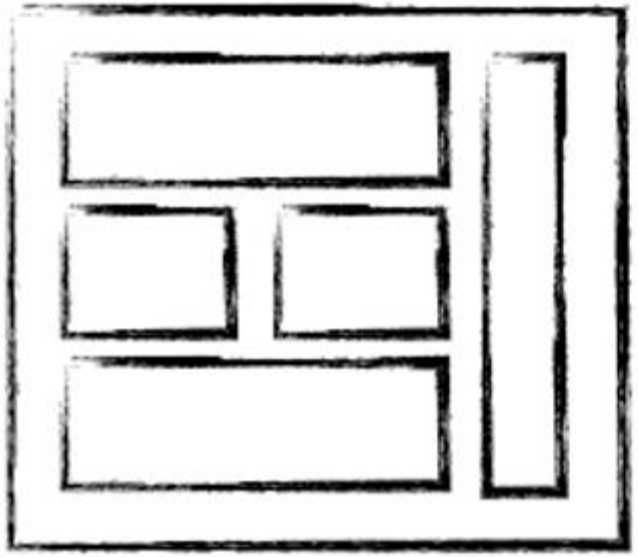


Application

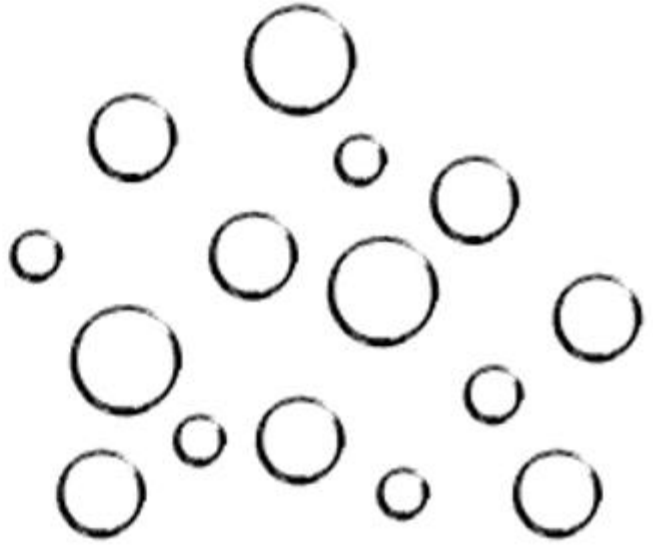



Aplicação
Dependente dos casos de uso
Data Transfer Objects
Application Services
Domínio
Independe de casos de uso
Domain Model
Domain Services

Microserviços

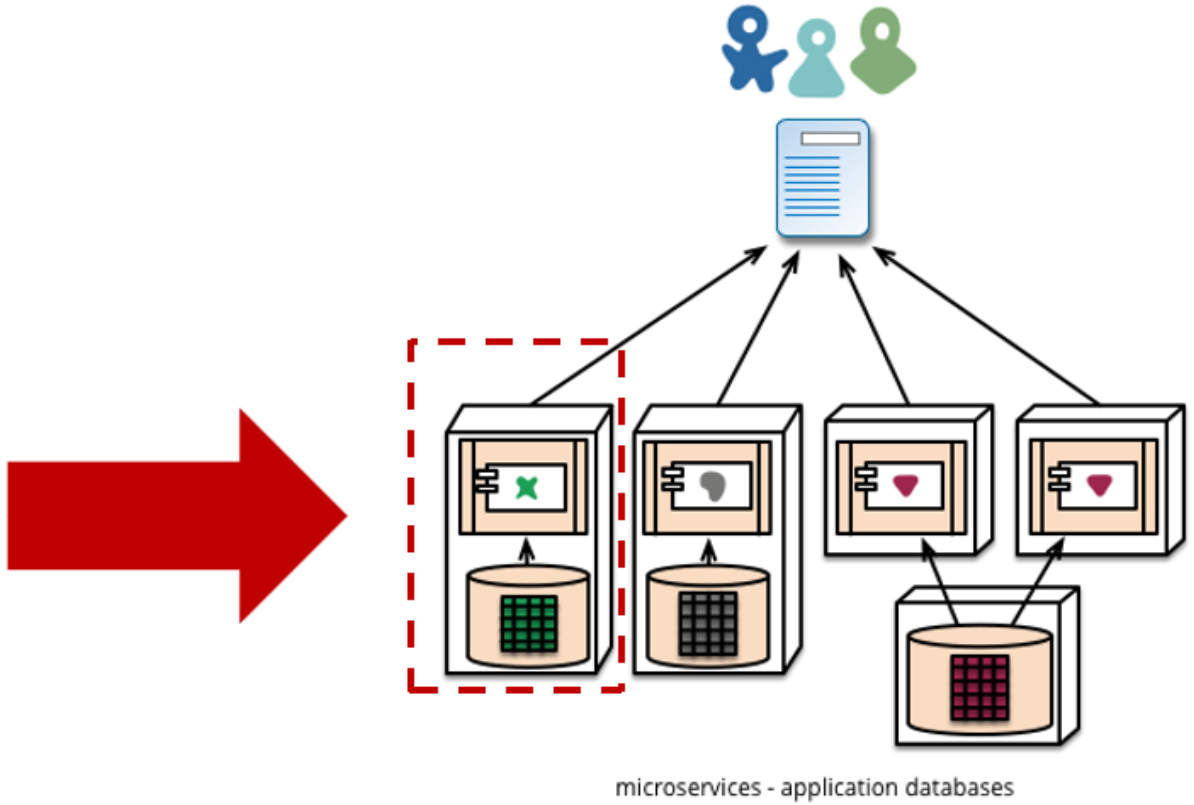


MONOLITHIC/LAYERED



MICRO SERVICES

Microserviços



WebApi

ASP.NET Web Api é um framework para construção de microserviços baseados no protocolo HTTP dentro do .NET Framework e do ASP.NET Core

Alcança mais clients

Formato apropriado no lado do cliente

Baseado na Arquitetura REST

- REST – Representational State Transfer
- Exploração extensa do HTTP
- Regras simples, modelo incremental, specs acompanham implementações

Vantagens: Protocolos menos complexos, mais poder e flexibilidade, menos overhead de protocolo, arquitetura amplamente disponível

Comparativo SOAP x REST

SOAP

- Protocolo de comunicação;
- Maior complexidade e especificações bem definidas
- Utiliza HTTP como meio de transporte;
- Permite diferentes protocolos para transporte de solicitações e respostas;
- Utiliza WSDL para definir um conjunto de métodos do servidor
- Poucas URIs, muitos métodos customizados

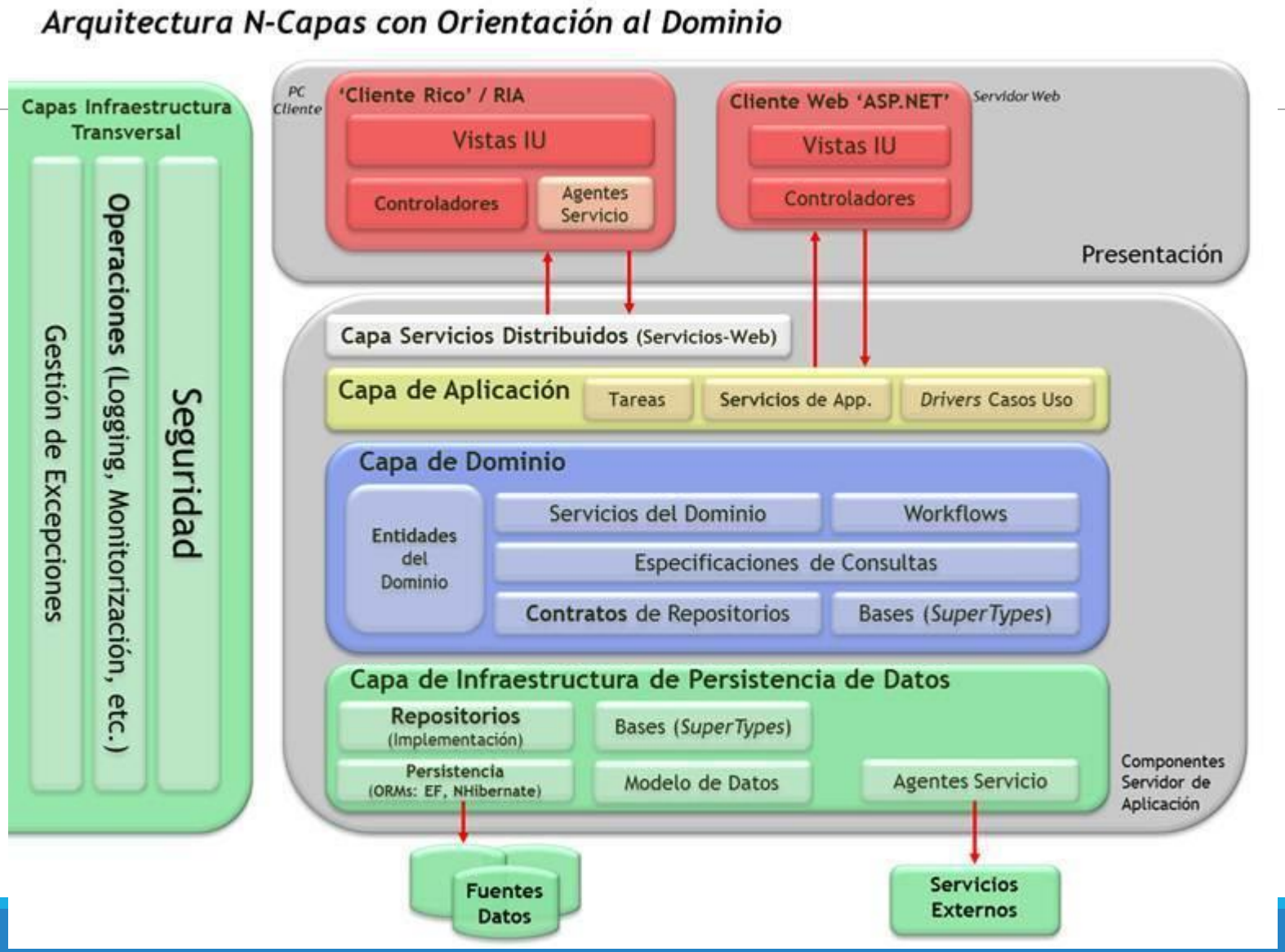
REST

- Estilo de arquitetura de software;
- Abordagem mais simples e menos burocrática
- Fortemente relacionado e utilização efetiva do protocolo HTTP.
- Não necessita de um contrato formal, embora possa ser utilizado WSDL ou WADL
- Muitos recursos (URIs), métodos uniformes

Envio de dados com base em SOAP - Simple Object Access Protocol

Envio de dados com base em REST - Representational State Transfer

Abordagem DDD (usaremos no projeto final de exemplo) – CRUD Clientes e Cadastro de Pedidos e Itens



Sintetizando

- O ASP.NET mudou completamente, abrindo diversas novas oportunidades para desenvolver aplicações multiplataforma, focadas em nuvem, performance e escalabilidade

- Novo ferramental como .NET Cli, novas IDEs proporcionam uma nova experiência no desenvolvimento com custo 0 em licenciamento

REFERÊNCIAS

- Introdução a C# e .NET Framework. https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework
- Microsoft Virtual Academy – ASP.NET Core 2 - https://mva.microsoft.com/learning-path/asp-net-core-2-0-23
- EVANS, Eric – Domain Driven Design – Atacando as complexidades no Coração do Software
- Clean Architecture for .NET Applications – https://paulovich.net/2018/05/15/clean-architecture-for-net-applications/
- Eduardo Pires MVP (curso ASP.NET MVC) – www.eduardopires.net.br
- Renato Groffe MVP – www.medium.com/@renato.groffe