Programação Web

Aulas Teóricas – Capítulo 2 – 2.1

1º Semestre - 2023/2024

Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Programação Web

Construção de Websites em ASP.NET Core

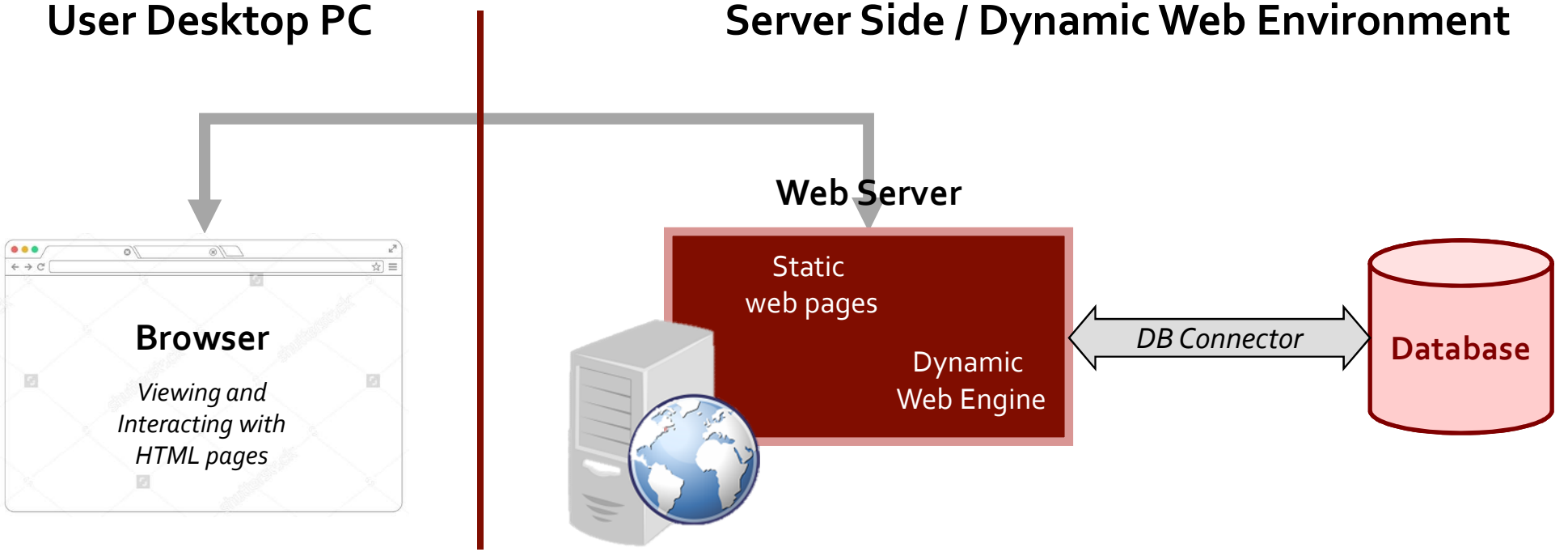
Departamento de Engenharia Informática e de Sistemas
Instituto Superior de Engenharia de Coimbra/Instituto Politécnico de Coimbra

Aplicações Web

- Conteúdo Estático
  - Não pode ser atualizado ou alterado sem efetuar alterações no código-fonte – praticamente já não existem

- Conteúdo Dinâmico
  - Conteúdos podem ser alterados ou atualizados sem necessidade de efetuar alterações no código-fonte
    - Existem Duas Perspectivas da noção de Dinamismo: alteração de características dos elementos HTML e alteração do conteúdo das páginas

Arquitectura Típica



Firefox, Chrome, Edge, etc ...

Microsoft IIS Apache HTTPD, Tomcat

JDBC, ODBC, ADO.NET LINQ, ORMs

SQL Server, Oracle, MySQL, PostgreSQL

Aplicações Web

- Aplicação que pode ser acedida por utilizadores com recurso a um *web browser*
- Apenas necessita ser instalada no servidor web
  - Modelo Cliente – Servidor diferente do tradicional
- Manutenção fácil de providenciar
- Multiplataforma
- Acessível de qualquer parte, desde que haja ligação à Internet

Tecnologias e *Frameworks*

C# 9

LINQ
Language Integrated Query

ASP.NET Core

ASP.Net Core Entity Framework

ASP.Net Core Identity

Razor
(Razor View Engine)

E HTML?
E Java Script?
E CSS?

JSP

PHP

...

Ruby

Node.js

Blazor

DEIS - Instituto Superior de Engenharia de Coimbra

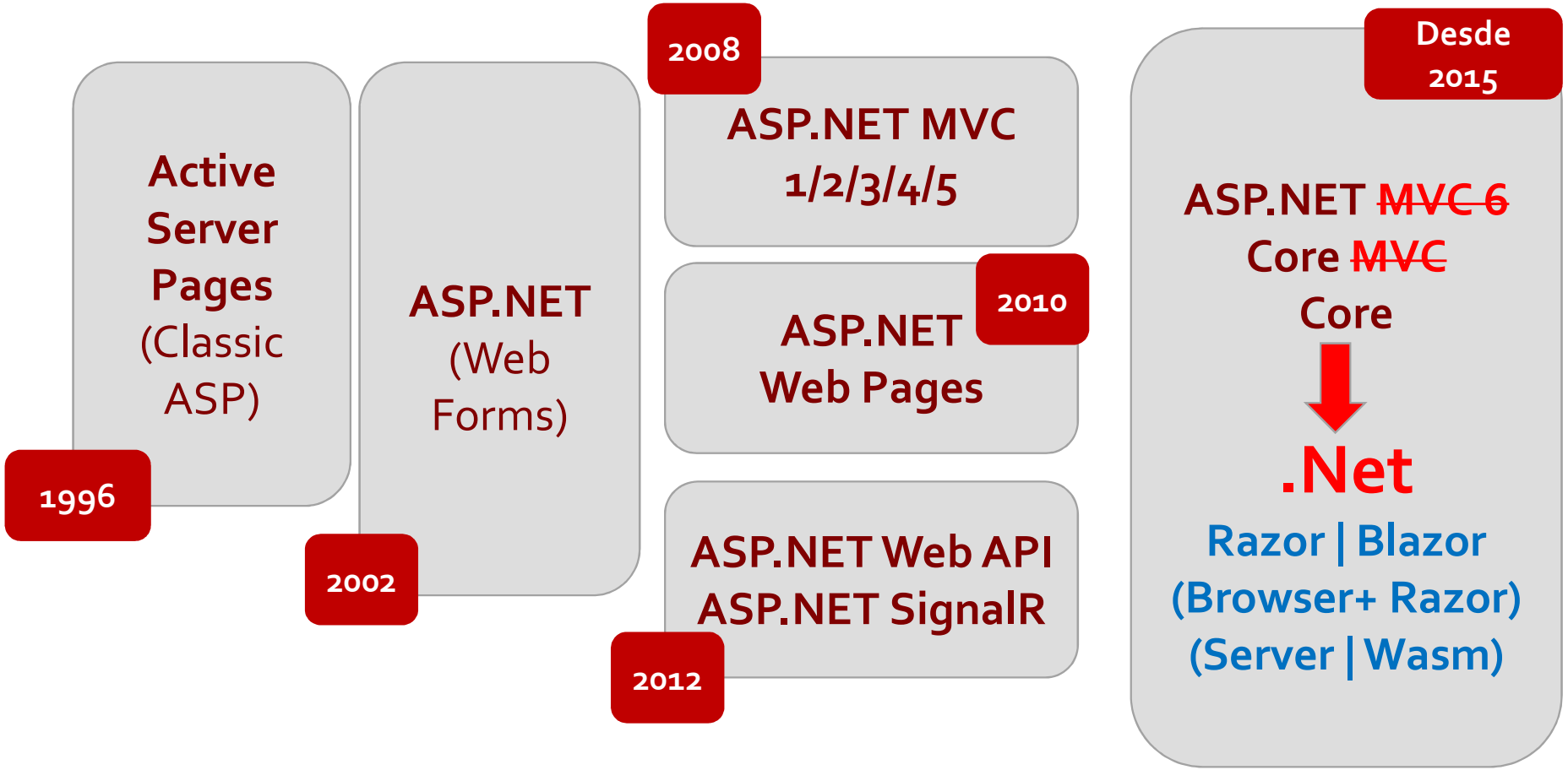
ASP.NET Core

- Framework da Microsoft para desenvolvimento de aplicações Web.
- A actual versão do ASP.Net Core, permite desenvolver aplicativos da Web baseadas na Cloud usando MVC, Razor Pages e Blazor
  - *Server-side + Client Side*
  - *Blazor Server*
  - *Blazor WebAssembly (WAsm)*

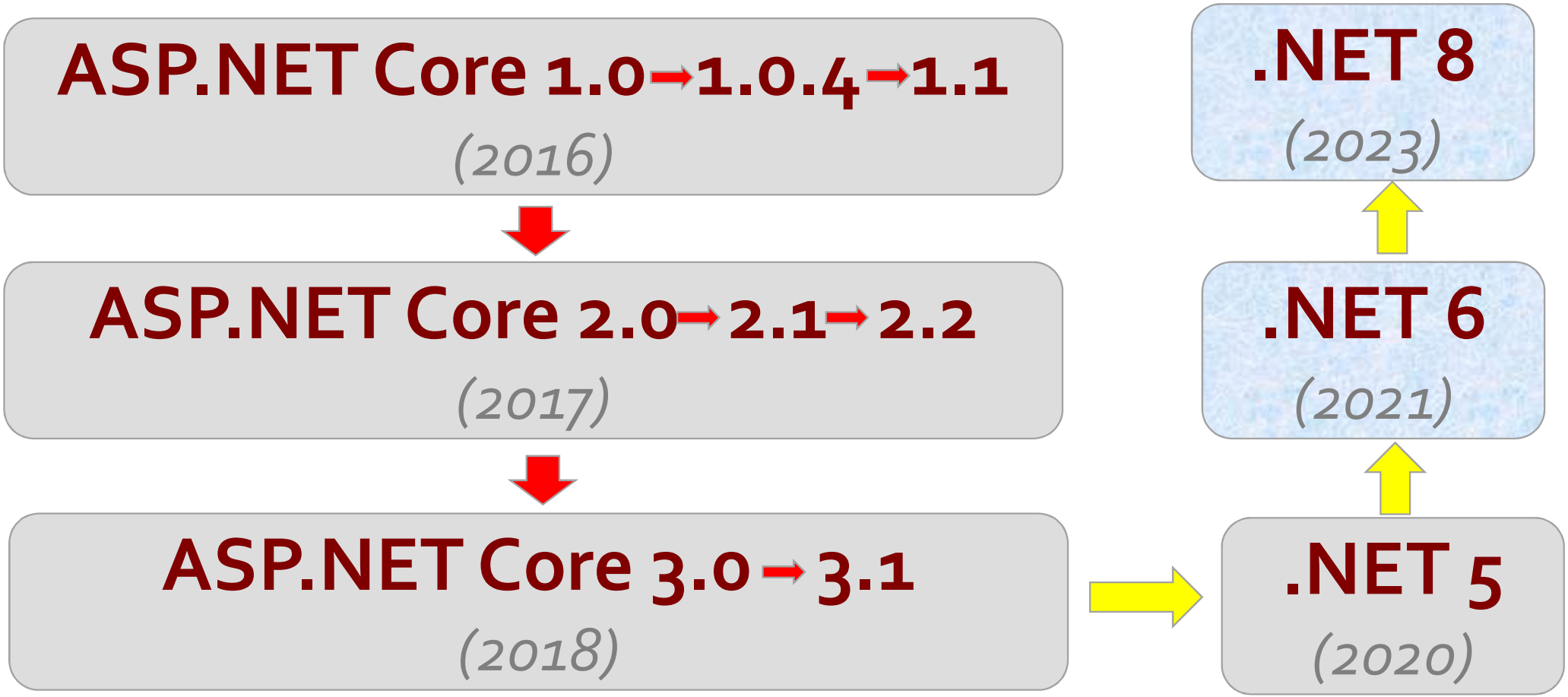
ASP.NET Core

- Este framework está directamente relacionado com o .NET Framework
- Sucessor da tecnologia ASP e ASP.Net MVC
- Também é Baseado em eventos
- Apesar de se poder continuar a usar uma diversidade de linguagens de programação, C#, F#, VB.NET (parcialmente), no desenvolvimento em .Net tem sido usado preferencialmente o C#, nomeadamente a última versão 9.0

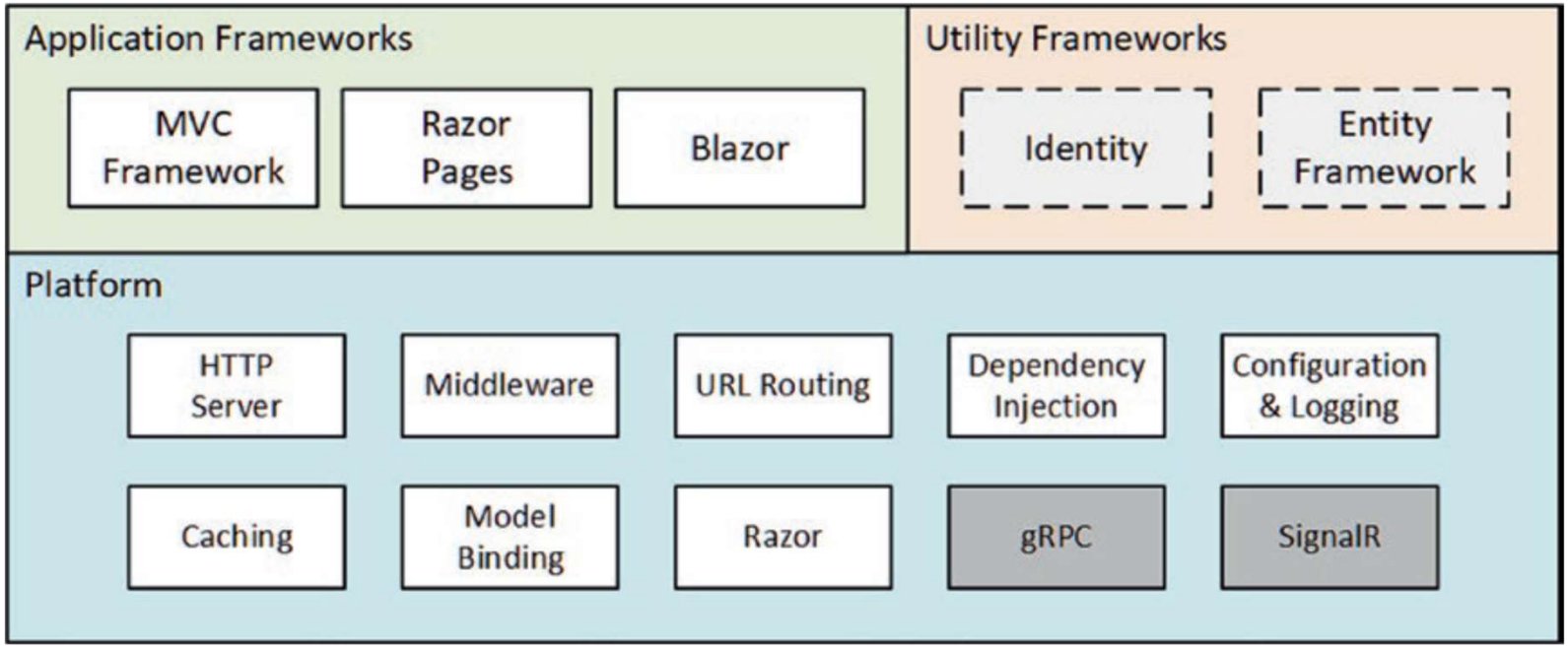
Evolução do ASP.NET



Evolução do ASP.NET



ASP.NET Core



Estrutura do ASP.Net Core [1]

ASP.NET Core

- O ASP.NET original dependia de um modelo de desenvolvimento designado por Web Pages, que significativamente o desenvolvimento de sites mas os projectos utilizados para o seu desenvolvimento não eram fáceis de gerir
- O MVC Framework, a versão anterior ao actual ASP.Net Core foi posteriormente introduzido e era utilizado em conjunto com as Web Pages com um modelo de desenvolvimento
- MVC significa Model-View-Controller, é um padrão de design através do qual as áreas de funcionalidade, também designadas por separação de interesses, são definidas de forma independente

ASP.NET Core

- As primeiras versões do framework MVC foram construídas sobre as bases do ASP.NET as quais tinham sido originalmente projetadas para o desenvolvimento das Web Pages
- Com a adopção do .NET Core, o ASP.NET tornou-se ASP.NET Core, e o framework MVC foi reconstruído tornando-se uma base aberta, extensível e multiplataforma
- O framework MVC continua sendo uma parte importante do ASP.NET Core mas no entanto a sua utilização já não é obrigatória e surgiu a abordagem de construção de sites designada por ***SPA's***, Aplicativos de Página Única.
  - Nesta abordagem SPA, os browsers fazem a solicitação de uma única página HTTP e em sequência o servidor envia um documento HTML, geralmente escrito num cliente JavaScript avançado por exemplo Angular ou React.

ASP.NET Core

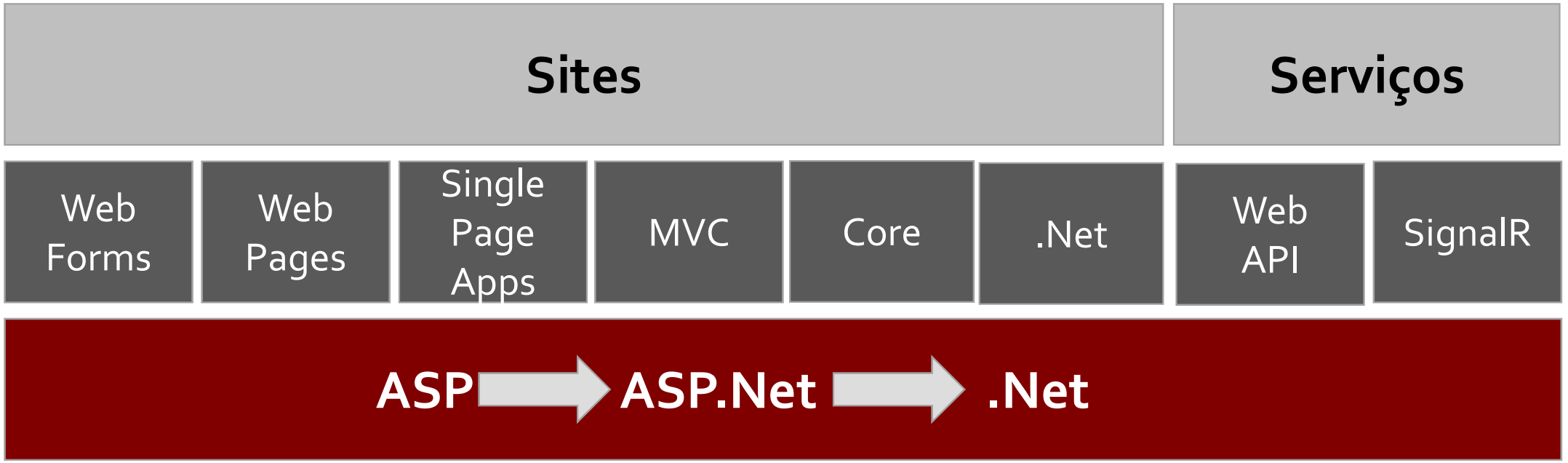
- A mudança para SPA's teve como resultado que a utilização do framework MVC já não é tão importante como originalmente era
  - No entanto no desenvolvimento utilizando o ASP.Net Core, continua-se a utilizar e é recomendável que assim seja feito, o MVC mas tão somente como um meio para se organizar minimamente as partes do site a desenvolver
- Desse modo o famework MVC continua a ser útil sendo então utilizado para dar suporte ao desenvolvimento de SPAs com recurso a serviços da web

ASP.NET Core

Aulas Laboratoriais

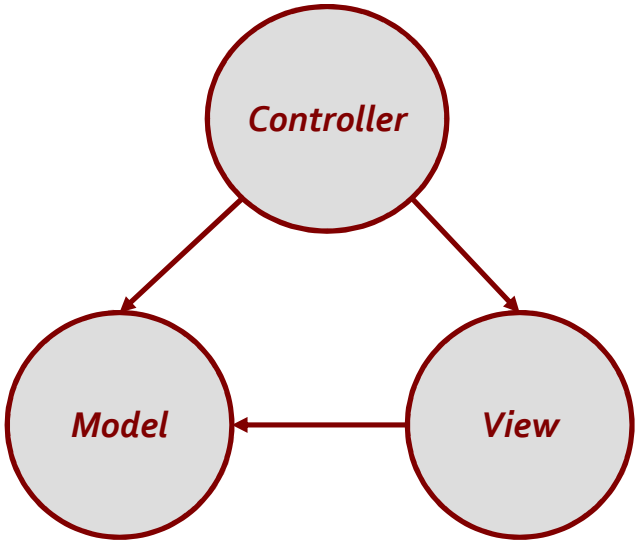
ASP.NET Core 6

ASP.NET Core


Sites
Serviços
Web Forms
Web Pages
Single Page Apps
MVC
Core
.Net
Web API
SignalR
ASP
ASP.Net
.Net

ASP.NET Core

- Desenvolvimento no padrão MVC
- MVC
  - Padrão de arquitetura de software
  - Separação da aplicação em três **componentes**:
    - **Model** (Manipula Regras de Negócio, Lógica e Dados)
    - **Controller** (Interação com utilizador e Lógica de entrada)
    - **View** (Aqui é produzida a camada de Apresentação)



ASP.NET Core

- Com a estruturação no padrão MVC, consegue-se:
  - *Test Driven Development* e Reutilização de código
    - Em MVC o *controller* é uma classe separada, possibilitando automatização de testes
    - *Controllers* não estão limitados a uma view específica
    - Maior Desempenho e Escalabilidade
- Nas Views, através do Razor View Engine, são geradas as páginas HTML que depois são enviadas ao Browser

ASP.NET *Core*

- *Razor View Engine*
  - O Razor, é um motor que a plataforma usa para gerar o código HTML a ser enviado ao browser. Tem uma sintaxe própria e simples para criação dinâmica de páginas *(embedding server code (C#)*
  - As Views são como que **templates** que são usadas para produzir o HTML+JS+CSS que vai ser enviado aos browsers (o programador praticamente não escreve código HTML)
  - Para se produzir esse HTML, o **Controlador** de acordo com o solicitado pelo utilizador através do browser, obtém do **Model** os dados necessários, de acordo com as regras de negócio definidas no model, consultando por exemplo a Base de Dados associada ao site, de seguida seleciona a **View** correspondente a qual, a partir destes dados e do código definido na view, produz recorrendo ao Razor View Engine a págima HTML a ser enviada ao browser

ASP.NET *Core*

- *Blazor*
  - Actualmente há 2 variantes do Blazor:
    - O Blazor Server
    - O Blazor WebAssembly
  - *A designação Blazor resulta da junção das designações Browser + Razor*
  - O Blazor muda muito o paradigma atual da Programação Web, em particular o Blazor Web Assembly, Blazor WAsm
  - *A Microsoft lançou a versão RTM (release-to-manufacturing) do Blazor com o .NET Core 3, sendo uma nova estrutura de front-end que resolve "todos" os problemas mencionados anteriormente*

ASP.NET *Core*

- ***Blazor***
  - Com o ***Blazor***, pode-se usar *C#* e *.NET* como estrutura central para escrever o *front-end* das aplicações, e desse modo os programadores *Web* habituados a tecnologias *Microsoft*, como é o caso do *ASP.Net Core*, tem assim muitas vantagens na adopção do *Blazor* pois basicamente usarão para o desenvolvimento do *front-end* tecnologias que já dominam, tal como *Razor*, *HTML* e C# para definir a interface de utilizador

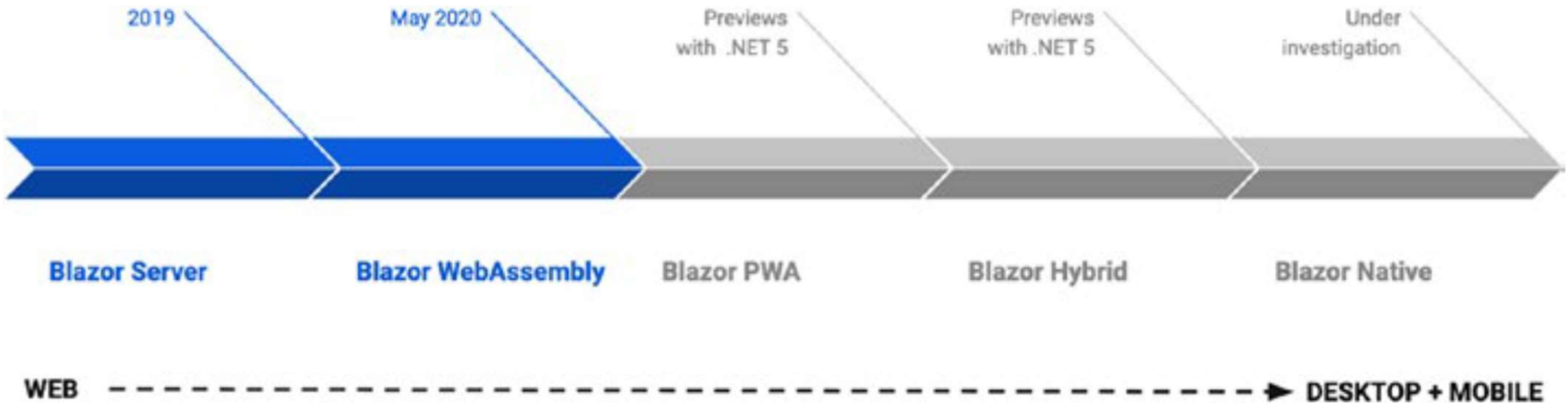
  - ***Blazor*** permite também que se execute o *front-end* diretamente no navegador, fornecendo todas as ferramentas necessárias para o desenvolvimento de aplicativos de página única

ASP.NET *Core*

- ***Blazor***
  - O ***Blazor*** foi criado em 2017 como um projeto pessoal de Steve Sanderson, que apresentou uma versão prévia do *Blazor* baseado em *DotNetAnywhere*, um *.NET* Intérprete de linguagem intermediária (IL)
  - Nesta altura o *Blazor* foi também adicionado ao repositório *ASP.Net GitHub* como um projeto experimental, mas a adesão da comunidade convenceu a *Microsoft* a mover o projeto para a equipe *ASP.Net*, substituindo *DotNetAnywhere* com *Mono*, que era o código aberto mais conhecido da plataforma baseada no *.NET Framework* (https://www.mono-project.com/).

ASP.NET *Core*

- ***Blazor***



Com o lançamento do ***.NET Core 3***, o ***Blazor*** tornou-se parte deste *framework* com um *road map* de desenvolvimento muito ambicioso como mostrado

ASP.NET *Core*

- Actualmente o *Blazor* só permite a criação de *front-end* de aplicativos web com, mas de acordo com o *road map*, será possível o desenvolvimento de aplicações para *desktops* e dispositivos móveis, passando por uma abordagem Progressiva da Aplicação Web (**PWA**) como uma etapa intermediária.
- ***Blazor Server*** é a versão que acompanha o ***.NET Core 3*** e permite pré-renderizar o ***HTML*** da aplicação e executar o código ***C#*** do lado do servidor e enviar as alterações da interface do utilizador para a página por meio do ***SignalR***

ASP.NET *Core*

- O ***Blazor WebAssembly*** que também já está disponível, executa o código C# diretamente no navegador, depois de compilado.
- O ***Blazor WebAssembly*** foi distribuído com o ***.NET Core 3.1.300*** e executa nesta versão ou numa posterior
- O ***Blazor WAsm***, traduz uma grande revolução no paradigma da Programação Web ao executar directamente no lado do cliente código que até aqui era somente executado do lado do servidor, como é o caso do ***C#***

ASP.NET *Core*

- O ***Blazor Hybrid*** será um renderizador ***.NET*** nativo para *Electron* e *WebView*, e permitirá o desenvolvimento de aplicações nativas funcionando *online* e *offline* (Elétron (electronjs.org) é um projeto de código aberto popular para a criação de aplicações de *desktop* de plataforma cruzada usando tecnologias da internet como por exemplo o *Visual Studio*. Esse código é baseado no Electron.
- ***Blazor Native*** terá o mesmo modelo de programação, mas sem renderização de *HTML*.
- Nota: No último capítulo o ***Blazor Server*** e ***Wasm*** serão abordados com mais detalhe

ASP.NET *Web API*

- Framework que permite desenvolver serviços HTTP
  - REST-based *webservices*
- ***REST*** significa *Representational State Transfer*. Em português, **Transferência de Estado Representacional**. Trata-se de uma abstração da arquitectura da Web. Resumidamente, o REST consiste em princípios/regras/constraints que, quando seguidas, permitem a criação de um projecto com interfaces bem definidas. Desta forma, permite, por exemplo, que aplicações comuniquem entre si

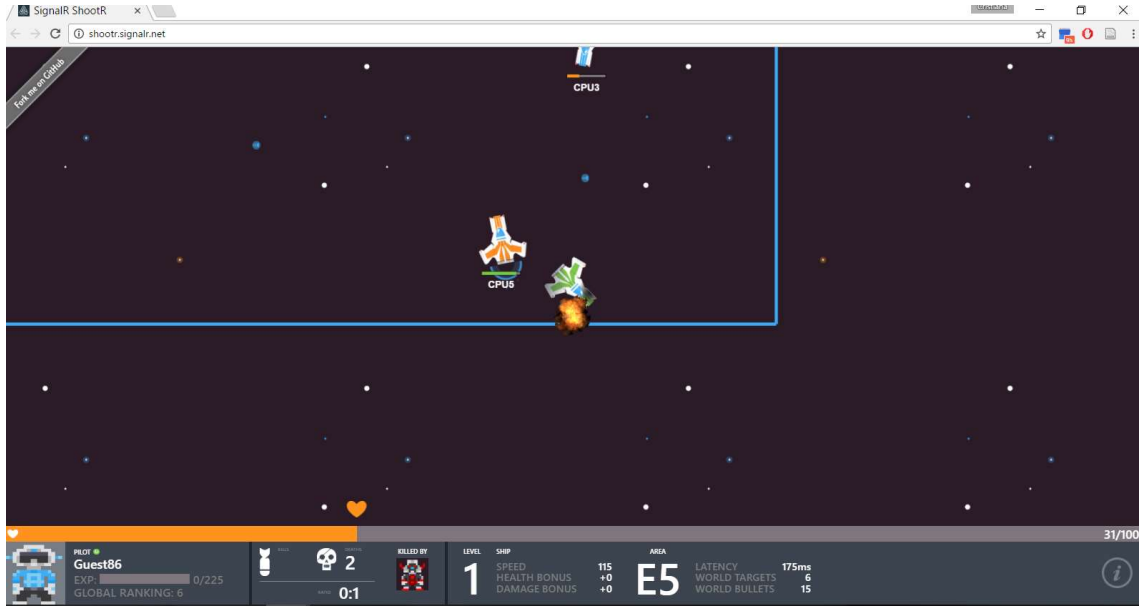
  (obtido de https://becode.com.br/o-que-e-api-rest-e-restful/, consulte)

ASP.NET *SignalR*

- É uma biblioteca integrada na framework
- Permite desenvolver funcionalidades "*real-time*" numa aplicação *ASP.NET Core*
  - Possibilita que o *server-side* faça, em tempo real, *push* aos clientes conectados
    - Ex: *Chat, Cotações da bolsa*
- Usa *WebSockets* quando disponível
- Actualmente é também usada para implementar o *Blazor WASM*

ASP.NET SignalR

- Permite desenvolver tipos de aplicações com alta frequência de atualizações a partir do servidor
  - Ex. Jogos em tempo real
  - Usado no Blazor



Exemplo: Site Dinâmico da Kaspersky

https://cybermap.kaspersky.com/pt