PROVAS DE TI
TUDO PARA VOCÊ PASSAR

ASP.NET

Prof. Rodrigo Macedo

Escopo do Curso

- Conceitos Iniciais.
- Características.
- Modelos de Desenvolvimento
- Ciclo de Vida.
- Migrations
- Identity.
- Questões de Concursos

Motivação

- O ASP (de Active Server Pages), também conhecido como ASP Clássico hoje em dia, é uma estrutura de bibliotecas básicas (e não uma linguagem) para processamento de linguagens de script no lado servidor para geração de conteúdo dinâmico na Web.
- ASP é uma tecnologia que a Microsoft criou para permitir o desenvolvedor web criar suas páginas de forma rápida, fácil e sem mistérios ou complicações.
- Desde o seu surgimento, houve um aumento significativo na quantidade de desenvolvedores web.
- Mas, como nem tudo é perfeito, o ASP, com o tempo mostrou alguns pontos fracos que foram surgindo com o aumento da necessidade dos usuários e complexidade das aplicações.

Motivação

- **Interpretado e código fracamente tipado** – O processo de execução de scripts do ASP interpreta linha a linha toda vez que a página é chamada. Além disso, mesmo dando suporte a variáveis, elas são fracamente tipadas como variantes e associada a tipos particulares somente quando o código é executado.
- **Colaboração é difícil** – Algumas empresas têm uma galera responsável pela parte de design da página e outra parte responsável pela implementação do negócio da aplicação. No modelo de programação no ASP mistura código script com HTML e, dessa forma, é muito difícil manter a colaboração entre essas duas equipes. É possível escrever parte do negócio da aplicação em componentes COM mas, em alguns momentos, é inevitável misturar HTML e código.

Motivação

- **Configuração da aplicação** - Para armazenar configurações, ASP usa um metabase, que é um repositório hierárquico parecido com o registry do windows. Essa metabase é parte do IIS e, ao mesmo tempo, uma estrutura difícil de compreender e navegar. Além disso, é difícil transferir configurações de um servidor web para outro.
- **Código extra para manipular os postbacks e dar suporte a diferente navegadores** – Sempre que o usuário clica no botão submit de um formulário, é feita uma requisição ao servidor web. Sendo assim, a cada requisição você tem que se preocupar com o que foi digitado pelo usuário para carregar novamente na tela.
- **Reaproveitamento de código** – Não fica tão fácil reaproveitar código quando você está misturando html e script. Além disso, a melhor forma de reutilização de código é através da orientação a objetos. O VBScript não tem implementação orientada a objetos.

Conceitos Iniciais

- ASP.NET é a plataforma da Microsoft para o desenvolvimento de aplicações Web e é o sucessor da tecnologia ASP. Permite, através de uma linguagem de programação integrada na .NET Framework, criar páginas dinâmicas.
- Não é nem uma linguagem de programação como VBScript, PHP, nem um servidor web como IIS ou Apache.
- O ASP.NET é baseado no Framework .NET herdando todas as suas características, por isso, como qualquer aplicação .NET, as aplicações para essa plataforma podem ser escritas em várias linguagens, como C#, F# e Visual Basic.NET.

Conceitos Iniciais

- Uma aplicação para web desenvolvida em ASP.NET pode reutilizar código de qualquer outro projeto escrito para a plataforma.NET, mesmo que em linguagem diferente.
- Uma página ASP.NET escrita em VB.NET pode chamar componentes escritos em C# ou Web Services escritos em C++, por exemplo.
- Ao contrário da tecnologia ASP, as aplicações ASP.NET são **compiladas** antes da execução, trazendo sensível ganho de desempenho.
- As aplicações Web ASP.NET necessitam do Framework.NET e do servidor IIS para executar, pelo menos na plataforma Windows.
- Projeto mod aspdotnet, módulo que permite o servidor Apache HTTP Server.
- O projeto Mono é um esforço para permitir que aplicações ASP.NET possam ser executadas em Linux.

Conceitos Iniciais

- O custo de desenvolvimento de programas utilizando ASP.NET é relativamente baixo. Antigamente, as pessoas desenvolvedoras precisavam comprar IDEs caras para construir os aplicativos. Hoje em dia, grande parte do seu código pode ser escrito utilizando ferramentas gratuitas.
- Em geral, o ASP.NET tem uma ótima arquitetura para ser usada na criação de sites e aplicativos da web. É confiável, rápido, fácil de usar, gratuito e muito conhecido. Além disso, o ASP.NET oferece um controle geral do seu desenvolvimento e conseguimos usá-lo para projetos de pequeno e grande porte.

Características

- **Orientação a objetos** – Sua aplicação será escrita numa das linguagens suportadas pelo framework.net e essas linguagens são OO.
- **Páginas compiladas** – Após a página ter sido requisitada, o framework verifica se essa página já foi compilada e, caso não tenha sido, compila só a primeira vez. Sendo assim, nas próximas requisições a página não será compilada e a execução será bem rápida.
- **Componentes** – Tudo agora é componente. Web controls (controles de servidor), Html controls, User controls(controles do desenvolvedor. Muito bom pra reaproveitar partes da tela) e Custom controls (controles de servidor escritos pelo desenvolvedor) e outros controles complexos que só o ASP.NET tem(Calendar, DataGrid, DataList, Repeater etc).

Características

- **Suporte do framework.NET** – Como o ASP.NET é do framework. Sendo assim, além do suporte das classes do ASP.NET, você tem suporte de todas as classes do framework. Tem classe pra tudo que você imaginar.
- **Configuração da aplicação** – Toda configuração da aplicação ASP.NET é feita através de arquivo xml. Sendo assim, não é preciso recompilar a aplicação após alguma mudança. O próprio framework detecta a mudança e reinicia a aplicação.
- **Ambiente RAD para desenvolvimento web** – Com o Visual Studio.NET e WebMatrix você tem um verdadeiro ambiente de programação semelhante ao ambiente do Visual Basic ou Delphi. Pra quem não conhece, o WebMatrix é uma ferramenta excelente para o desenvolvimento de aplicações asp.net (usa o modelo de programação Code In Page).

Características

No ASP.NET temos dois modelos de programação: Code Behind e Code in Page.

1. **Code Behind** – Nesse modelo teremos uma verdadeira separação do HTML e código C#. Aqui, para cada arquivo aspx, teremos um arquivo aspx.cs (caso tenha escrito em c#) onde será digitada toda parte do código c#. No arquivo aspx, só teremos a parte HTML e a parte da declaração dos componentes do ASP.NET.
2. **Code in Page** – Esse modelo é bem semelhante ao modelo do ASP clássico. A diferença é que existem outras tags que não temos no ASP. Tags que permitem você fazer herança, implementar interface, importar outras classes etc

Modelos de Desenvolvimento

- O ASP.NET oferece quatro estruturas para a criação de aplicativos Web: Web Forms, ASP.NET MVC e Páginas da Web do ASP.NET e Web API.
- Todas as estruturas são estáveis e maduras, e você pode criar ótimos aplicativos Web com qualquer um deles. Independentemente da estrutura escolhida, você terá todos os benefícios e recursos do ASP.NET em todos os lugares.
- Cada estrutura tem como alvo um estilo de desenvolvimento diferente. O que você escolhe depende de uma combinação de seus ativos de programação (conhecimento, habilidades e experiência de desenvolvimento), o tipo de aplicativo que você está criando e a abordagem de desenvolvimento com a qual você está familiarizado.

Modelos de Desenvolvimento

1. **ASP.NET Web Forms:** Com o ASP.NET Web Forms, você pode criar sites dinâmicos usando um modelo familiar de arrastar e soltar orientado por evento. Uma superfície de design e muitos controles e componentes lhe permitem compilar rapidamente sites sofisticados, poderosos baseados em UI com acesso de dados.
2. **ASP.NET Web API:** o ASP.NET Web API é um modelo que tem como objetivo facilitar o desenvolvimento de integração com APIs para a maioria dos clientes, Desktop ou Mobile. Além do que, o ASP.NET Web API é a plataforma ideal para o desenvolvimento de aplicativos RESTful.

Modelos de Desenvolvimento

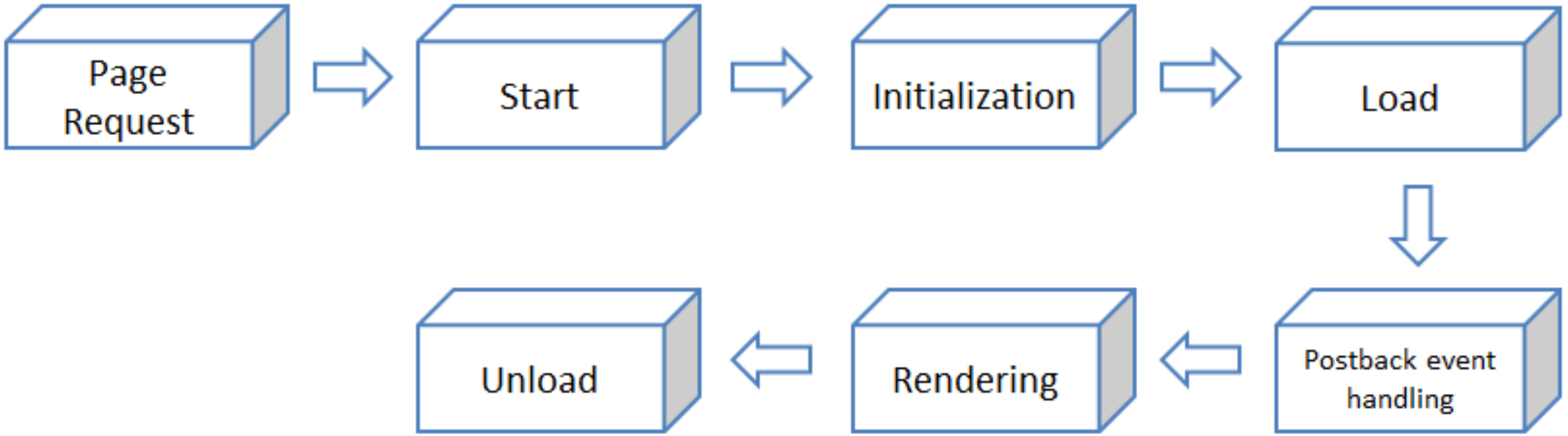
1. **ASP.NET Web Pages:** ASP.NET Web Pages é uma das estruturas fundamentais. Utilizando a sintaxe do Razor, que é uma outra ferramenta da Microsoft para escrever código HTML, conseguimos combinar o código do servidor com o HTML para criar um conteúdo totalmente dinâmico. Essa estrutura permite criar páginas na web com os padrões mais recentes de desenvolvimento.
2. **ASP.NET MVC:** O ASP.NET MVC tem a habilidade de compilar sites dinâmicos de uma forma muito mais rápida e com diversos padrões, fora que temos uma separação limpa de preocupações e que possibilita a você total controle sobre a marcação do desenvolvimento ágil.

Modelos de Desenvolvimento

	Se você tiver experiência em	**Estilo de desenvolvimento**	**Conhecimentos**
Formulários da Web	Win Forms, WPF, .NET	Desenvolvimento rápido usando uma biblioteca rica de controles que encapsulam marcação HTML	RAD avançado e de nível intermediário
MVC	Ruby on Rails, .NET	Controle total sobre a marcação HTML, o código e a marcação separados e os testes fáceis de escrever. A melhor opção para aplicativos móveis e de página única (SPA).	Nível intermediário, avançado
Páginas da Web	ASP clássico, PHP	Marcação HTML e seu código juntos no mesmo arquivo	Novo, de nível intermediário

Ciclo de Vida

Existem sete etapas envolvidas no ciclo de vida de uma página ASP.NET:



Ciclo de Vida

- **Page Request:** Essa etapa ocorre antes mesmo de ser iniciado o ciclo de vida da página, que é quando o usuário faz a chamada da página. Neste momento, o servidor ASP.NET verifica se a página precisa ser analisada ou se pode ser utilizada uma versão que foi armazenada em cache.
- **Start:** A partir desta etapa é que se inicia de fato o ciclo de vida da página, sendo assim, neste momento algumas propriedades são definidas, como Request e Response. Se a requisição é um Postback ou uma nova requisição a página e a definição da propriedade UICulture.

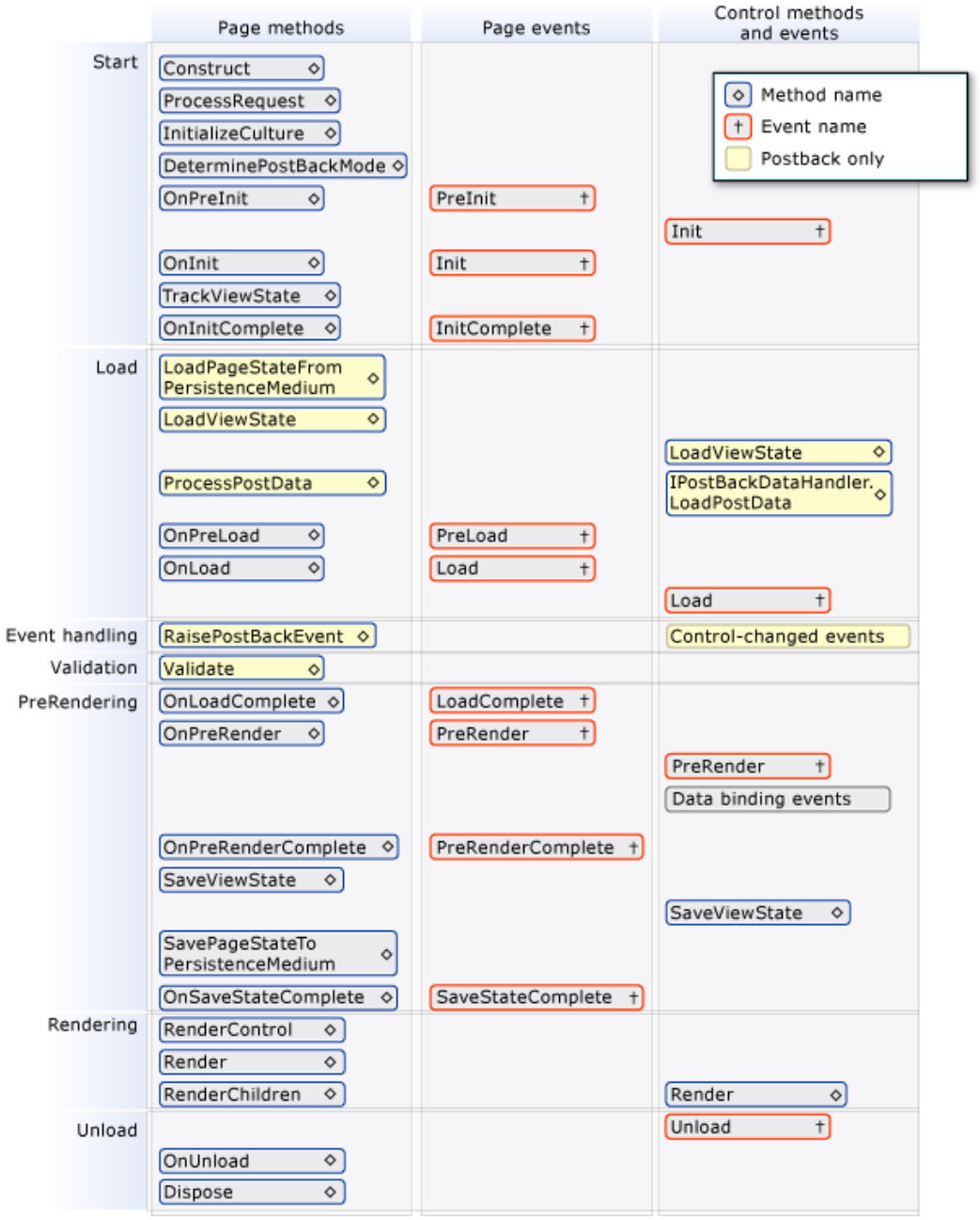
Ciclo de Vida

- **Postback** é uma requisição da página onde são enviados dados ao servidor e o mesmo é encarregado em fazer toda lógica de programação contida e devolver um resultado para o navegador.
- **Initialization:** Neste momento os controles que irão compor a página ficam disponíveis e recebem seus respectivos UniqueID. Nessa etapa existe uma ressalva: caso a requisição seja um Postback, os dados e valores das propriedades ainda não foram carregados/restaurados com os valores contidos na ViewState.
- **Load:** Durante a etapa de Load, ou seja, carregamento da página, os valores das propriedades e dados do ViewState são carregados, caso a requisição seja um Postback.

Ciclo de Vida

- **Postback event handling:** Neste ponto os eventos dos controles são executados. Algumas validações são feitas, ponto onde a propriedade IsValid verifica os validadores individuais da página.
- **Rendering:** Antes de inicializar a renderização dos controles e objetos da página, o ViewState é salvo. Após isso cada elemento da pagina é renderizado.
- **Unload:** Assim que a página for processada esta etapa é chamada, nesse momento as propriedades da página, como Response e Request, são descarregados.

Ciclo de Vida

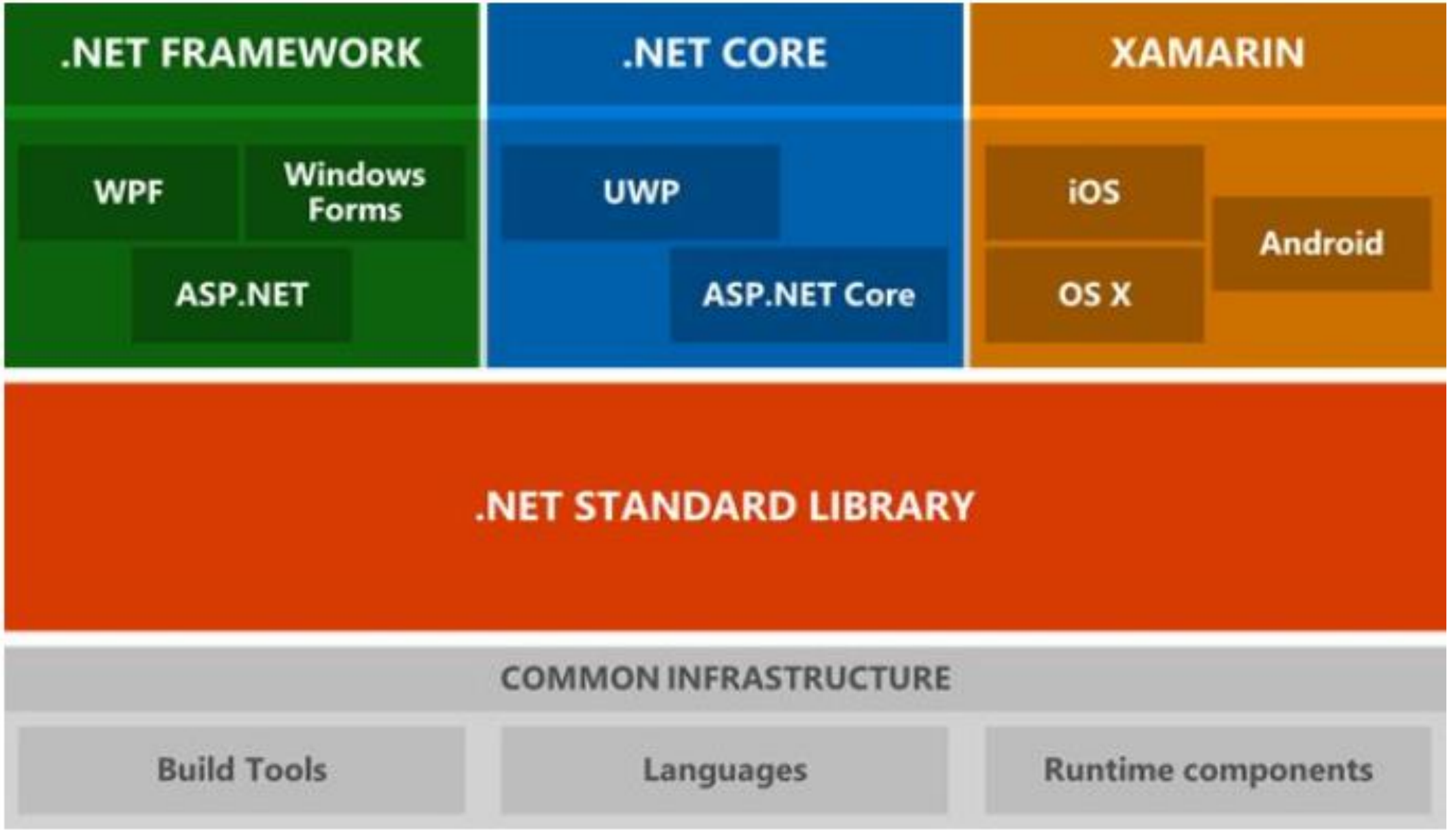


.NET Framework x .NET Core

- O .NET Framework é usado para desenvolvimento de aplicações Windows usando Windows Forms , WPF e de aplicações Web usando ASP .NET MVC.
- O .NET Core é open-source e multiplataforma e suporta UWP e as bibliotecas da ASP .NET Core. A UWP é usada para criar aplicação Windows 10 e a ASP .NET Core é usada para criar aplicações Web para Windows, Linux e Mac.
- O Xamarin é um framework que permite criar aplicações nativas para Android, iOS e Windows Phone.

.NET Framework x .NET Core

Uso .NET Core: Você deseja segmentar suas aplicações nos sistemas operacionais Windows, Linux e Mac;



.NET Framework x .NET Core

Característica	.NET Core	.NET Framework
Sistema de alto desempenho e escalável sem UI	SIM	
Utilização de contentores Docker	SIM	
Responde fortemente na linha de comando	SIM	
Necessidades entre plataformas	SIM	
Aplicações Web centradas na interface do usuário		SIM
Aplicações cliente Windows		SIM
Possui um ambiente pré-configurado e sistemas		SIM
Versão estável para a necessidade imediata de criação e implantação		SIM
Possui experiência da equipe .NET		SIM
O tempo não é um problema. As experiências são aceitáveis.	SIM	

ASP .NET Core

- Sucessor do ASP.NET, o ASP.NET Core é um framework open-source, multiplataforma, criado pela Microsoft e a comunidade. Leve, rápido e modular, funciona em conjunto com o .NET Core.
- Lançado em 2016, mesmo sendo um sucessor do ASP.NET, o ASP.NET Core foi criado totalmente do zero, para que não precisasse se preocupar com código legado permitindo assim seguir o padrão de desenvolvimento web moderno.
- Assim como outras plataformas, o ASP.NET Core é totalmente modular, recursos podem ser adicionados via pacotes Nuget. O que permitiu que a plataforma fosse mais performática que seu antecessor.

ASP .NET Core

- ASP.NET Core é um redesign de ASP.NET 4. x, incluindo alterações arquitetônicas que resultam em uma estrutura mais enxuta e mais modular.
- O ASP.NET Core oferece os seguintes benefícios:

1. Uma história unificada para a criação da interface do usuário da Web e das APIs Web.
2. Projetado para capacidade de teste.
3. Razor Pages torna os cenários voltados para a página de codificação mais fáceis e produtivos.
4. Capacidade de desenvolver e executar no Windows, macOS e Linux.

ASP .NET Core

- O ASP.NET Core oferece os seguintes benefícios:

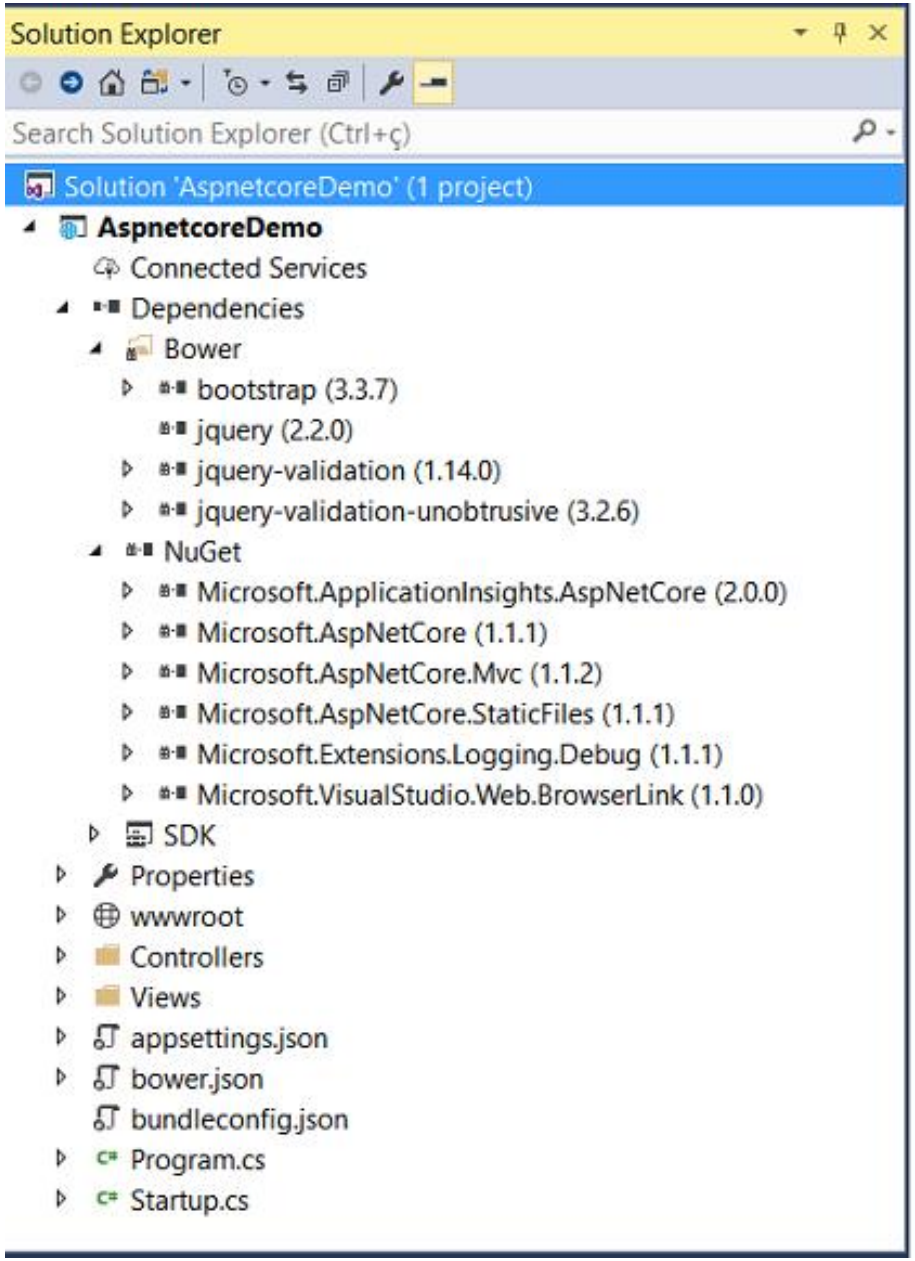
1. De software livre e voltado para a comunidade.
2. Integração de estruturas modernas do lado do cliente e fluxos de trabalho de desenvolvimento.
3. Suporte para hospedagem de serviços RPC (chamada de procedimento remoto) usando gRPC.
4. Um sistema de configuração pronto para a nuvem, baseado no ambiente.
5. Injeção de dependência interna.
6. Um pipeline de solicitação HTTP leve, modular e de alto desempenho.

Estrutura do Projeto

- ASP .NET Core a partir da versão do VS 2017 teve o uso do arquivo project.json descontinuado em seus projetos web.
- Agora, os projetos web criados para ASP .NET Core voltam a utilizar arquivos .csproj (um padrão baseado no MSBuild) para definir as configurações e dependências de um projeto.
- A estrutura de uma solução ASP .NET Core criada no VS 2017 Community é representada por:

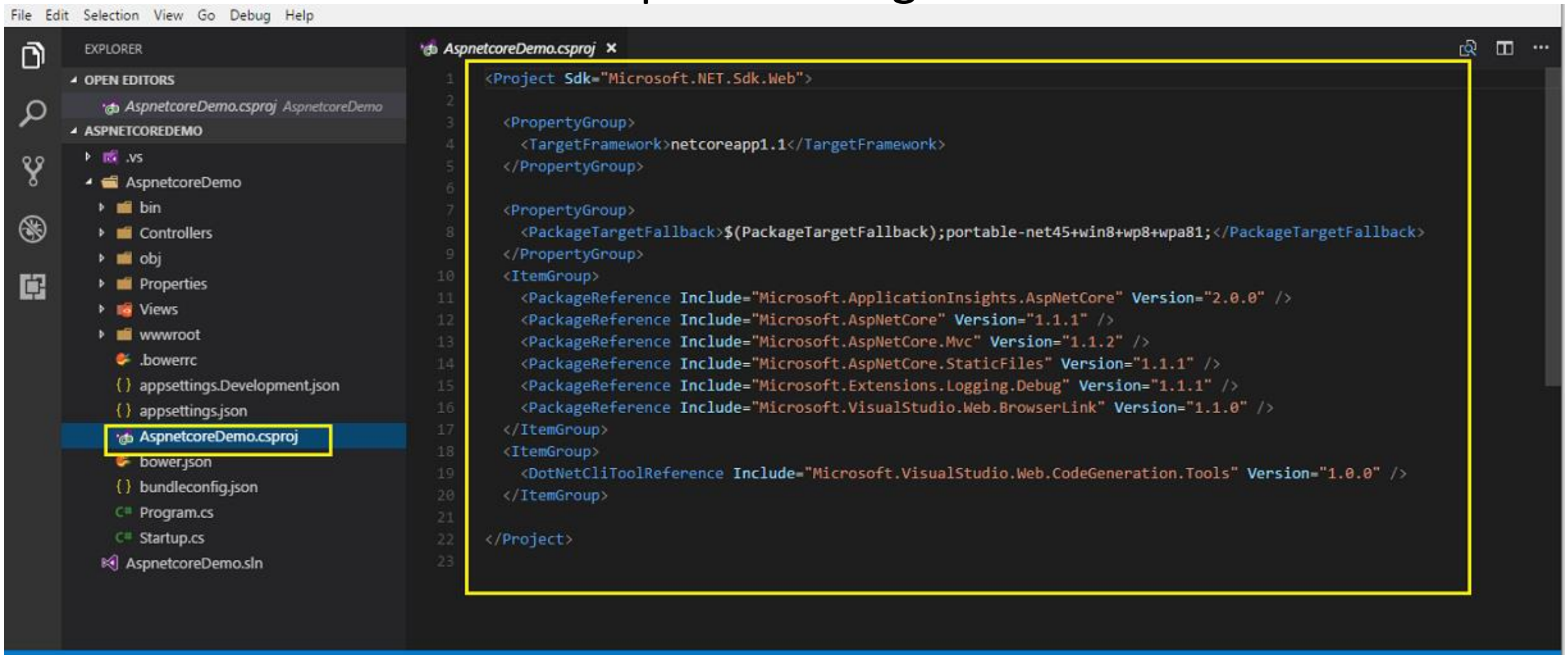
1. Bower - pacotes para o front-end
2. Nuget - Bibliotecas .NET
3. SDK - Indicando o framework da aplicação

Estrutura do Projeto



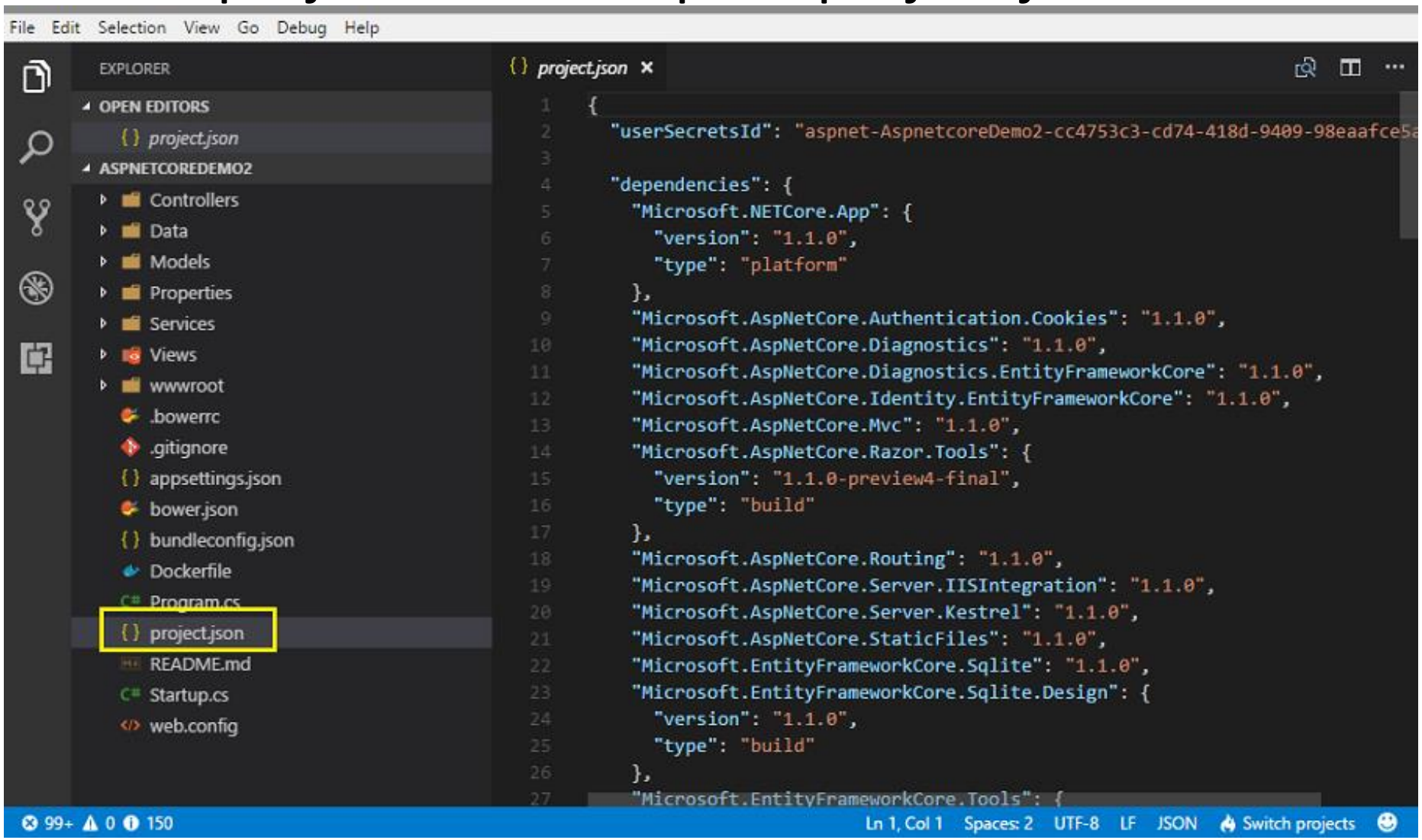
Estrutura do Projeto

- Abrindo o arquivo da solução no Visual Studio Code veremos o arquivo AspnetcoreDemo.csproj indicando o framework alvo e as referências aos pacotes Nuget.



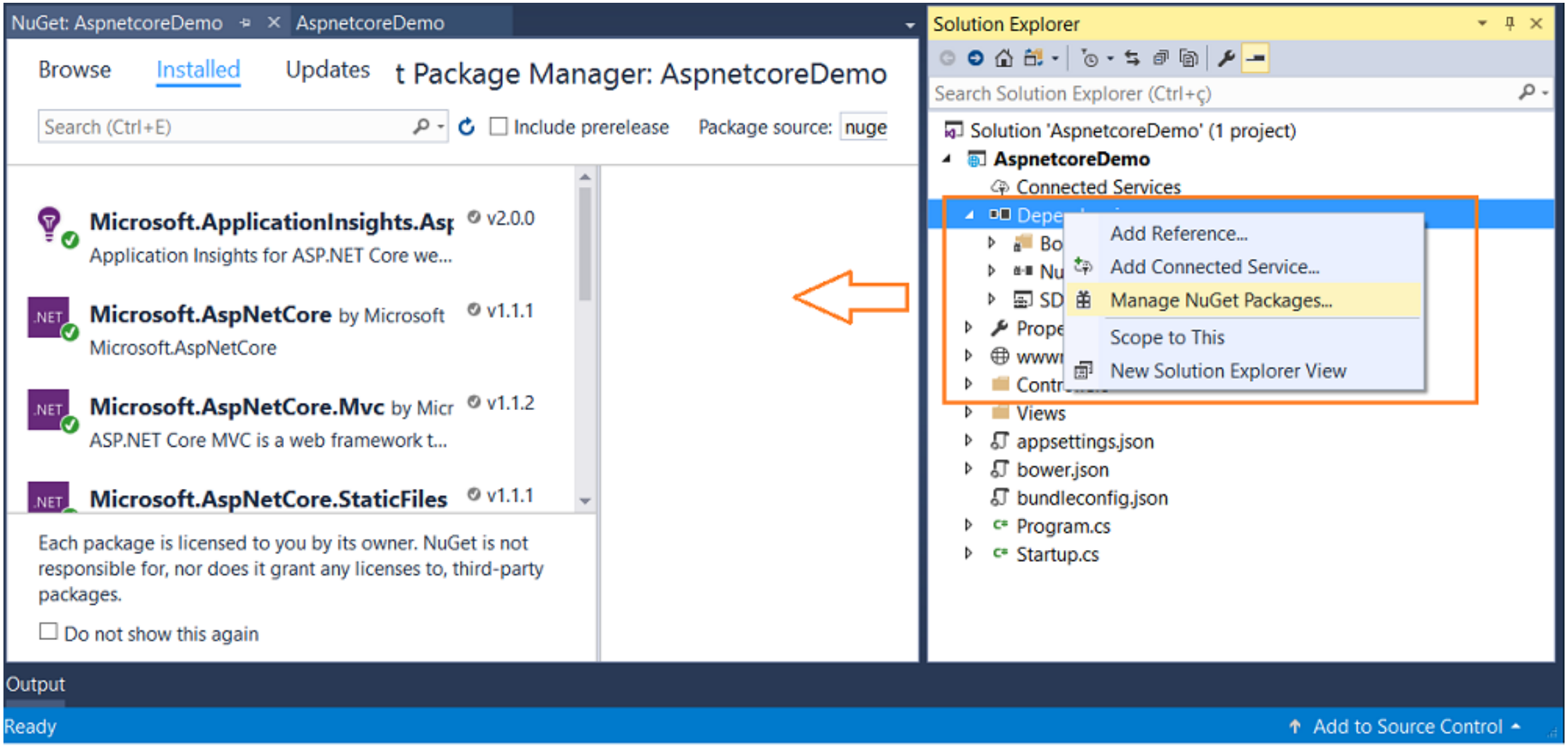
Estrutura do Projeto

- Se for gerado o mesmo projeto web usando o Yeoman será criado no projeto com o arquivo project.json



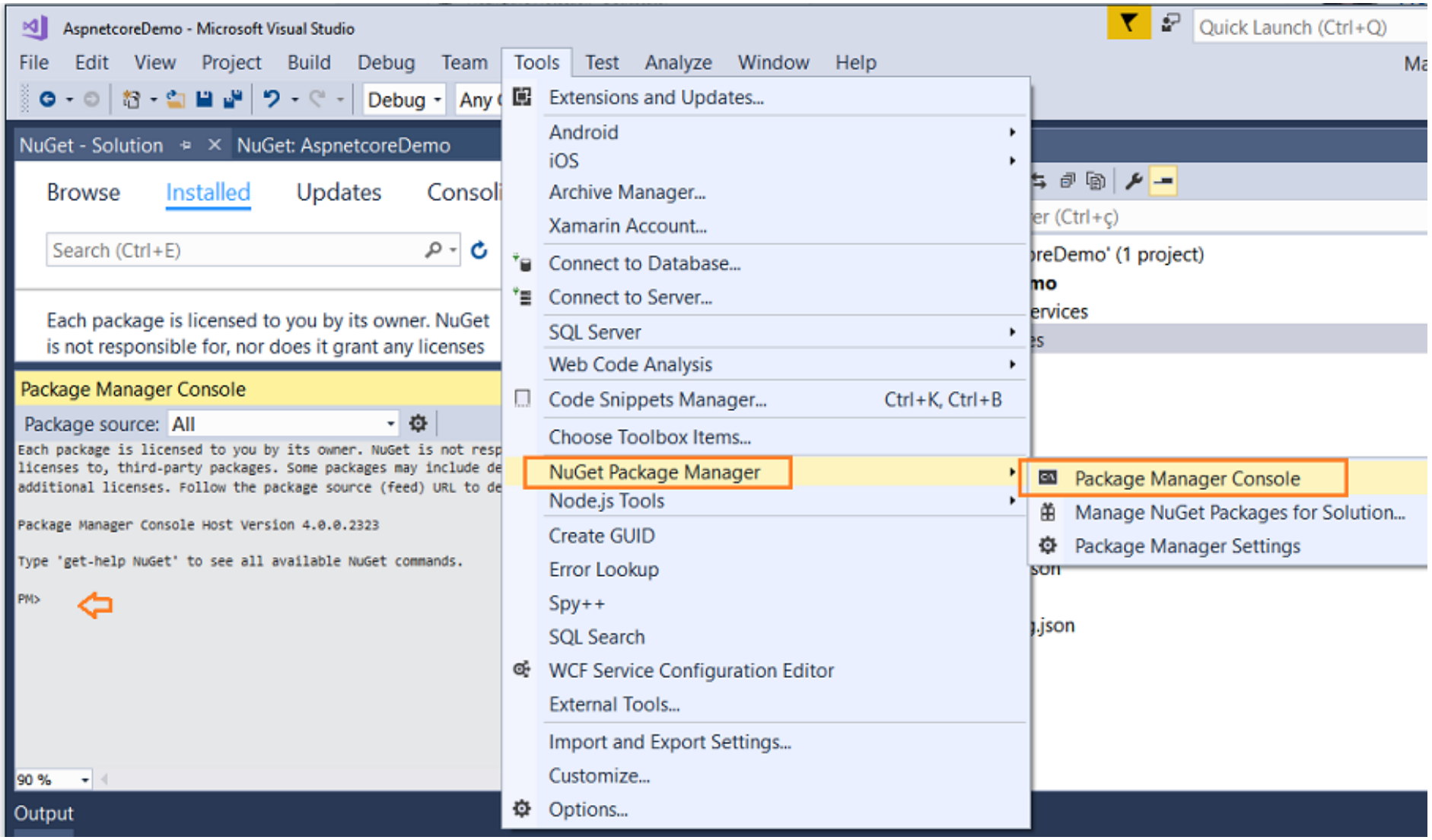
Gerenciamento de Pacotes

- Para adicionar pacotes, pode ser utilizado o gerenciador de dependências Nuget.



Gerenciamento de Pacotes

- Para adicionar pacotes via console, pode ser utilizado o gerenciador de dependências Nuget.



Gerenciamento de Pacotes

- É possível gerenciar pacotes via console por meio dos comandos:
- **Find-Package** - Lista os pacotes disponíveis. Ex: Find-Package EntityFramework -version 6.1.1
- **Install-Package** - Instala o pacote referenciado. Ex: Install-Package EntityFramework
- **Uninstall-Package** - Desinstala o pacote referenciado. Ex: Unisntall-Package EntityFramework
- **Update-Package** - Atualiza um pacote Ex: Update-Package EntityFramework

Gerenciamento de Pacotes

- A NET CLI (Net Core Command Line Interface) é uma nova ferramenta multiplataforma para desenvolvimento de aplicativos em .NET Core.

1. **dotnet new** - Exibe os templates disponíveis;
2. **dotnet new <nome_template> -n <nome_projeto> -** Cria o projeto com base no nome do template com o nome informado;
3. **dotnet migrate -** Migra os projetos que usam o arquivo project.json para o formato .csproj;
4. **dotnet restore -** Restaura as dependências e os pacotes do projeto;
5. **dotnet build -** Constroi o projeto;
6. **dotnet run** - Executa o projeto;

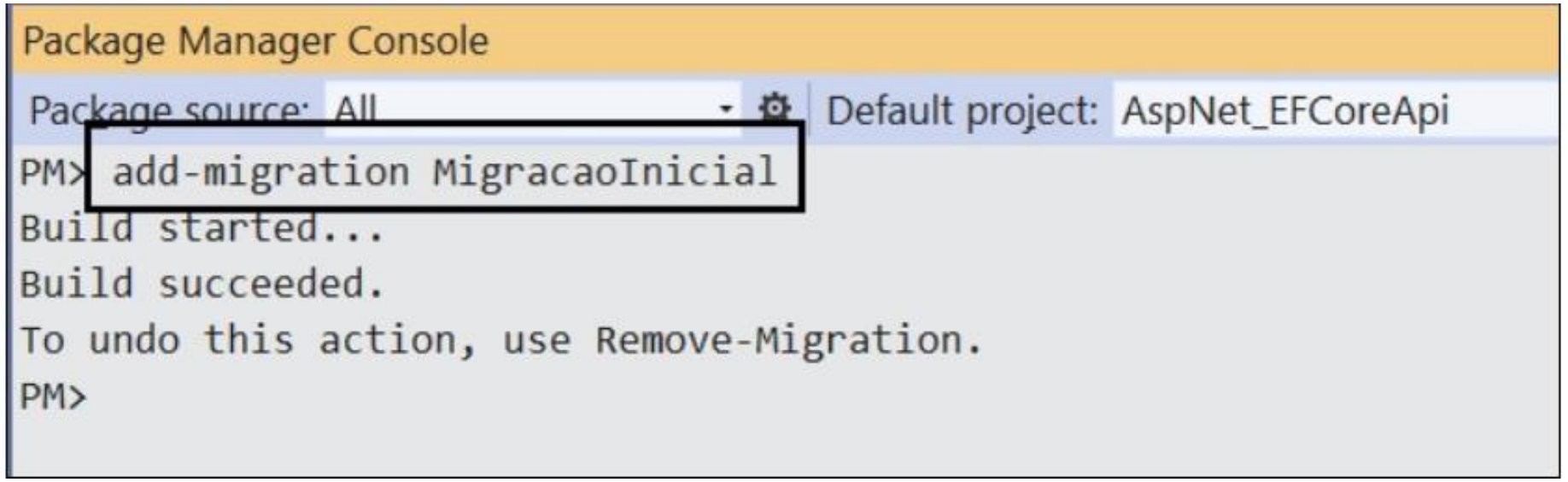
Migrations

- Para acompanhar as mudanças feitas no modelo de dados e manter a sincronia do modelo com o banco de dados o EF Core oferece o recurso **Migrations** ou Migrações que permite atualizar de forma incremental o esquema do banco de dados e assim mantê-lo sincronizado com o modelo de dados do seu projeto preservando os dados existentes.
- Para poder usar o recurso Migrations a primeira coisa a fazer é instalar as ferramentas do Entity Framework Core que ajudam nas tarefas de desenvolvimento em tempo de projeto e são usadas para gerar migrações e fazer a engenharia reversa do esquema do banco de dados. Podemos usar essas ferramentas de duas maneiras:

1. Usar as ferramentas no Visual Studio na janela do Package Manager Console (Windows)
2. Usar as ferramentas de linha de comando (CLI) com o VS Code. Exigem um .NET Core SDK. (Windows, Linux e Mac)

Migrations

- Para utilizar Migrations é necessário o pacote Microsoft.EntityFrameworkCore.Tools instalado para funcionar.
- É necessário a ferramenta de linha de comando do EF Core globalmente usando o comando: **dotnet tool install --global dotnet-ef**.
- Para criar uma migration deve ser executado o comando: dotnet ef migrations add NomeDaMigração [options]



Migrations

No método **Up()** temos os comandos para criar a tabela Alunos onde as colunas da tabela possuem os mesmos nomes das propriedades definidos na classe Aluno.

No método **Down** temos o comando para remover a tabela criada caso desejamos remover a migração.

Esses dois métodos são usados na classe que herda de Migration para definir as alterações que serão aplicadas ao banco de dados, sendo que no método Up() aplica o script e o método Down() reverte a aplicação feita.

```
using System;
using Microsoft.EntityFrameworkCore.Migrations;

namespace AspNet_EFCoreApi.Migrations
{
    public partial class MigracaoInicial : Migration
    {
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Alunos",
                columns: table => new
                {
                    Id = table.Column<Guid>(nullable: false),
                    Nome = table.Column<string>(nullable: true),
                    Idade = table.Column<int>(nullable: true),
                    Ativo = table.Column<bool>(nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Alunos", x => x.Id);
                });
        }

        protected override void Down(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.DropTable(
                name: "Alunos");
        }
    }
}
```

Para aplicar a migration é executado o comando: : **update-database [options]**

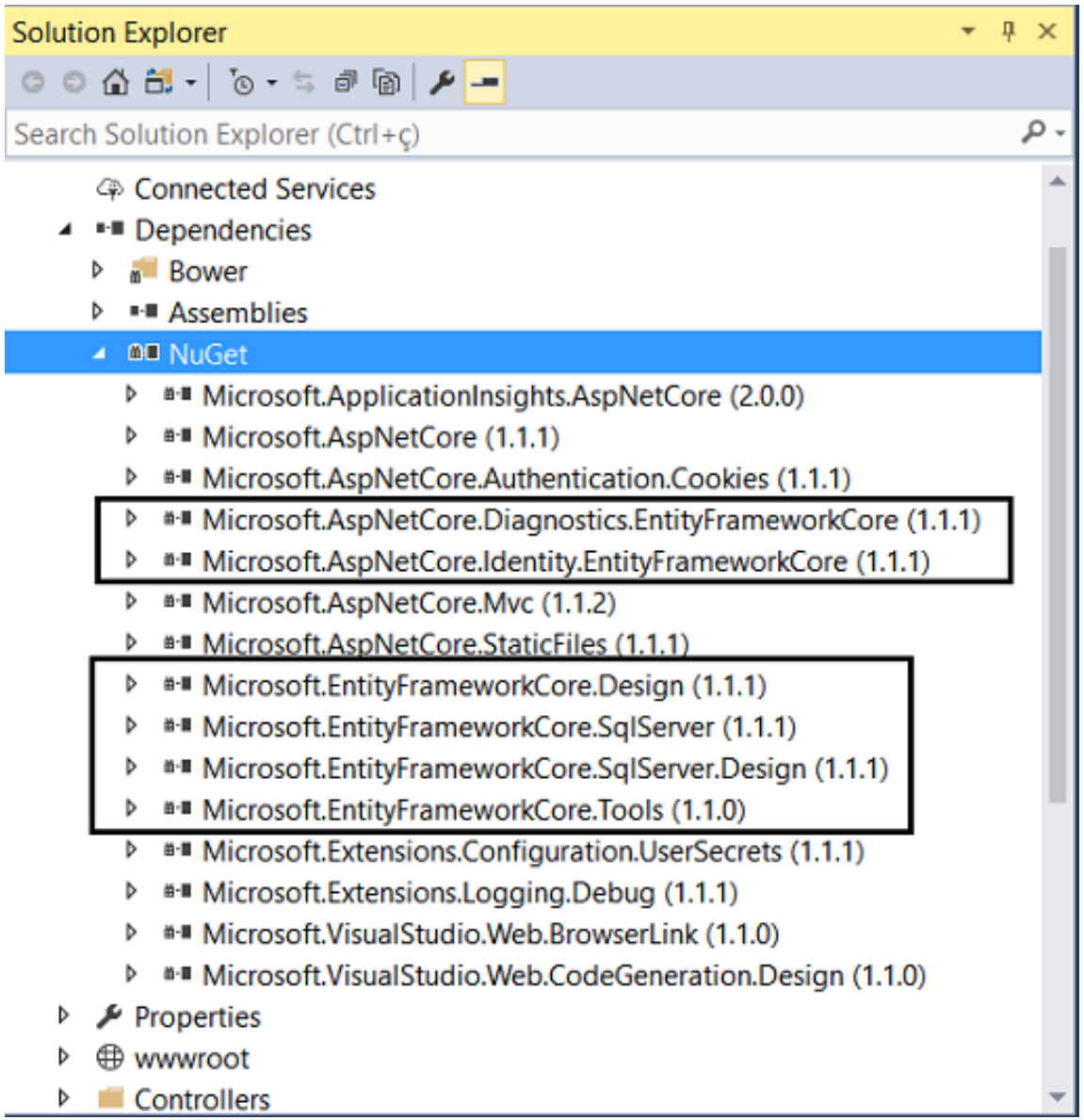
Identity

- A ASP.NET Core Identity é um sistema de associação que permite adicionar funcionalidade de login ao seu aplicativo. Os usuários podem criar uma conta e fazer o login com um nome de usuário e senha ou podem usar um provedor de login externo, como Facebook, Google, Conta da Microsoft, Twitter, etc.
- Podemos configurar a ASP.NET Core Identity para usar um banco de dados SQL Server para armazenar nomes de usuário, senhas e dados de perfil. Como alternativa, você pode usar seu próprio armazenamento persistente para armazenar dados em outro fonte de armazenamento como o Azure Table Storage.
- É uma API que dá suporte à funcionalidade de logon da interface do usuário.
- Gerencia usuários, senhas, dados de perfil, funções, declarações, tokens, confirmação por email e muito mais.

Identity

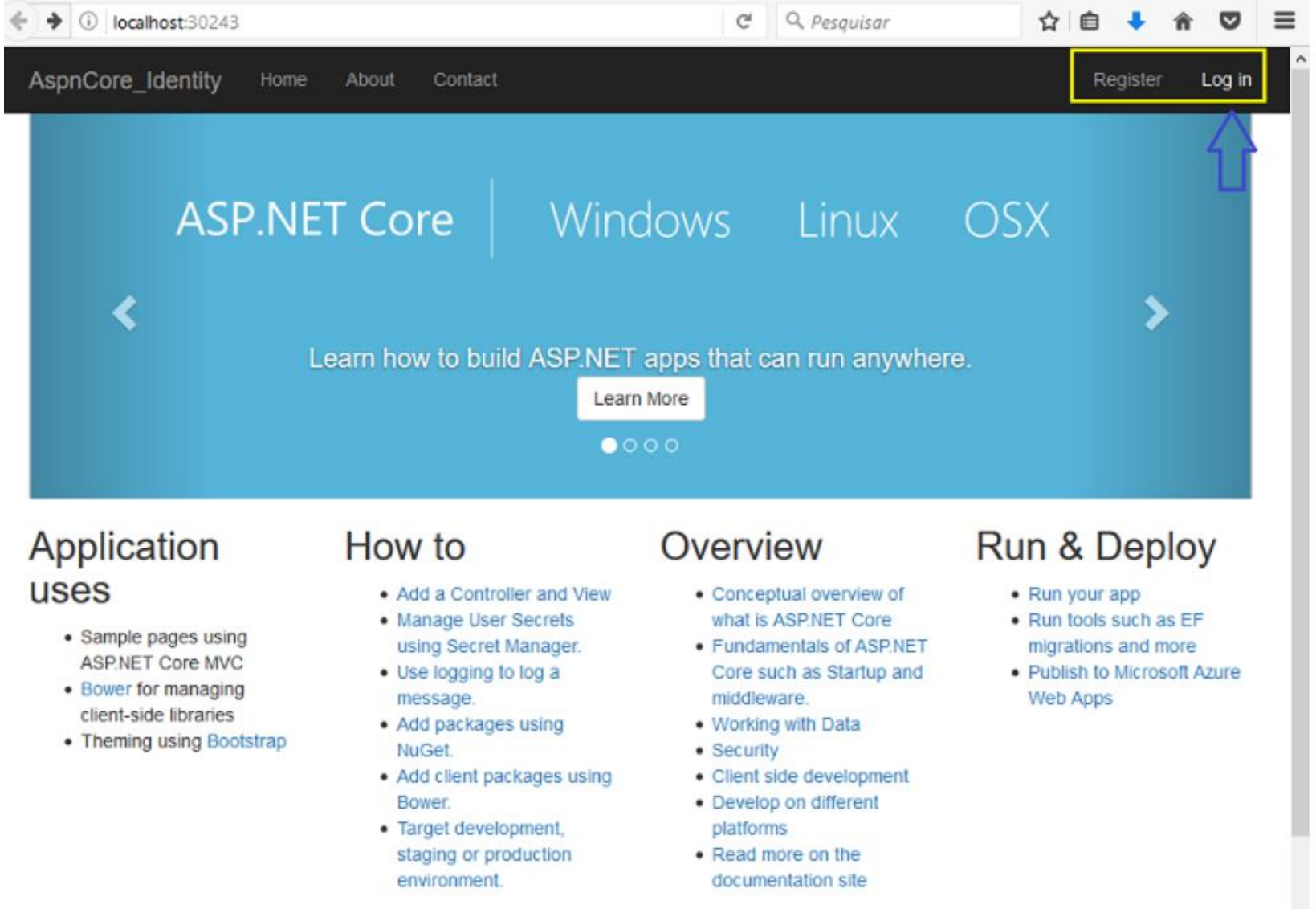
- Para utilizar o Identity em um projeto .NET Core, é necessário adicionar alguns pacotes.

O recurso Identity está habilitado para o aplicativo chamando UseIdentity no método Configure da classe Startup. Isso adiciona uma autenticação baseada em cookie ao pipeline de solicitação.



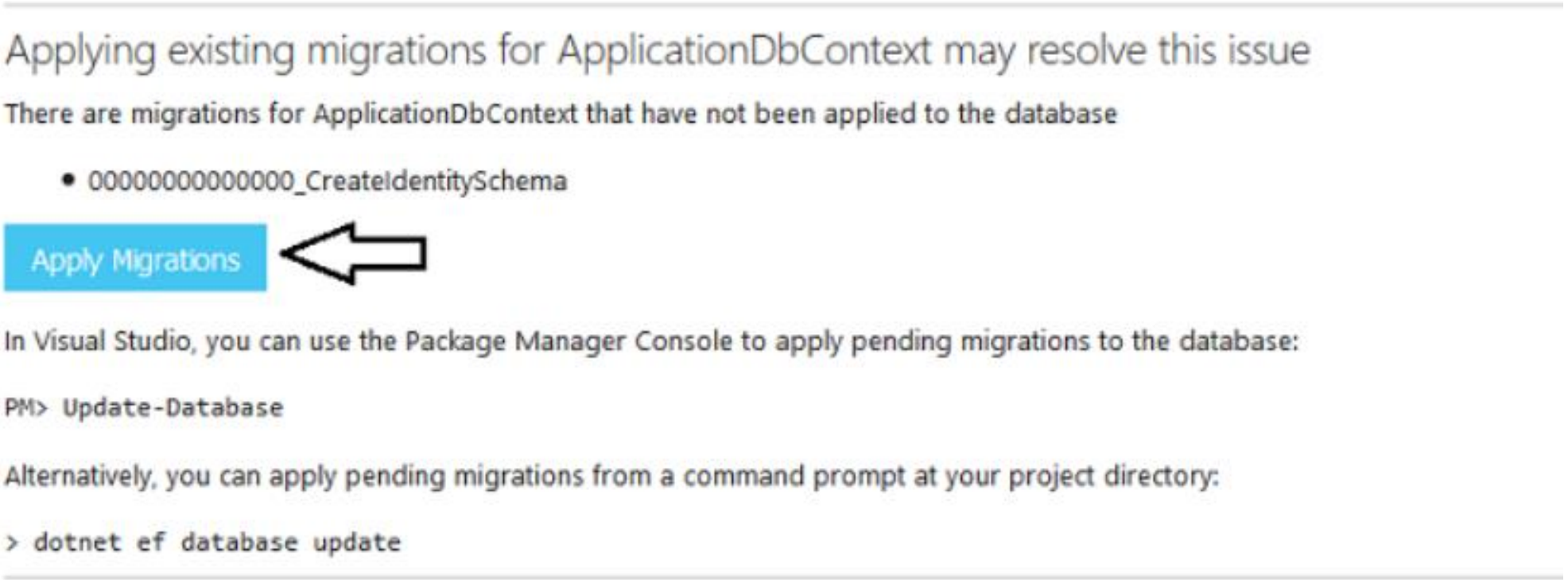
Identity

- Executando a aplicação iremos obter a página incial com os links para Registrar um usuário e fazer o login.

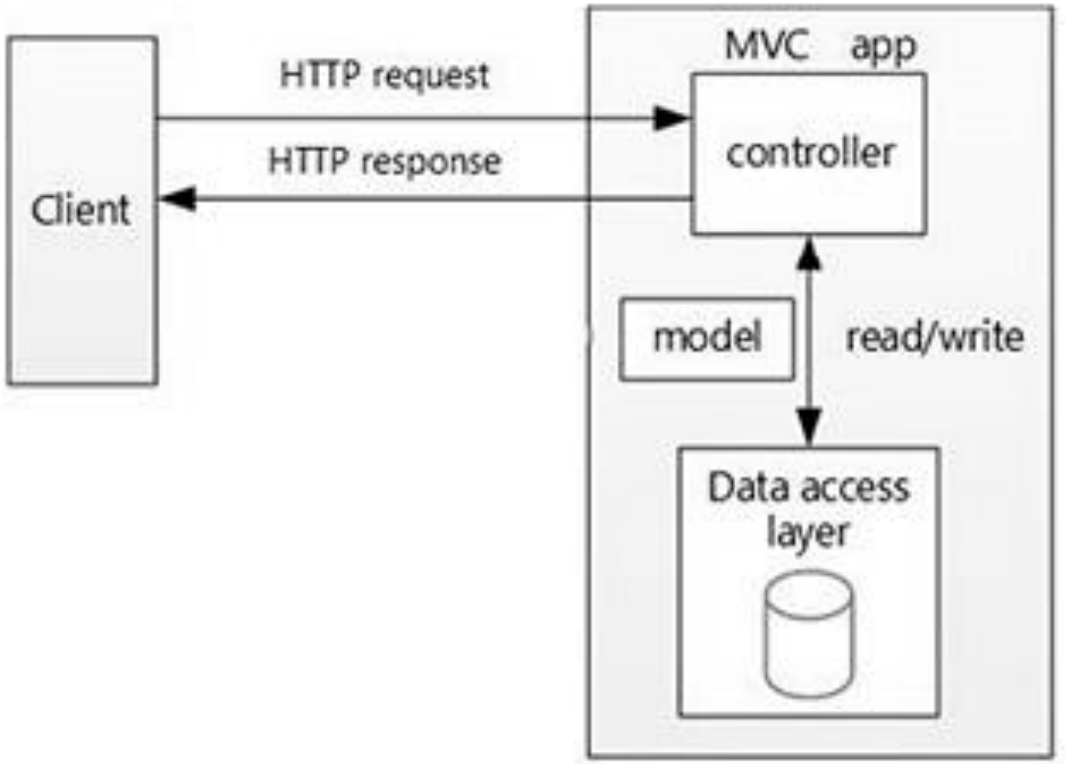


Identity

- Ao realizar o login se for retornado uma mensagem de erro conforme mostrada a seguir, basta clicar no botão Apply Migrations e dar um Refresh na página.



Q1) [FCC SEFAZ-SC 2018] Considere a imagem ao lado, que representa um modelo de arquitetura de uma aplicação utilizando ASP.NET Web API.



Nessa arquitetura,

a) o model pode ser serializado para JSON e XML

b) o client, que consome a Web API, será obrigatoriamente um navegador web.

c) só pode haver um controller para manipular solicitações HTTP e representar os dados.

d) o model é representado por classes VBScript conhecidas como Plain Old Objects (POOs).

e) o controller permite requisições HTTP somente por meio do método GET.

Q1) [FCC SEFAZ-SC 2018] Considere a imagem ao lado, que representa um modelo de arquitetura de uma aplicação utilizando ASP.NET Web API.

Nessa arquitetura,

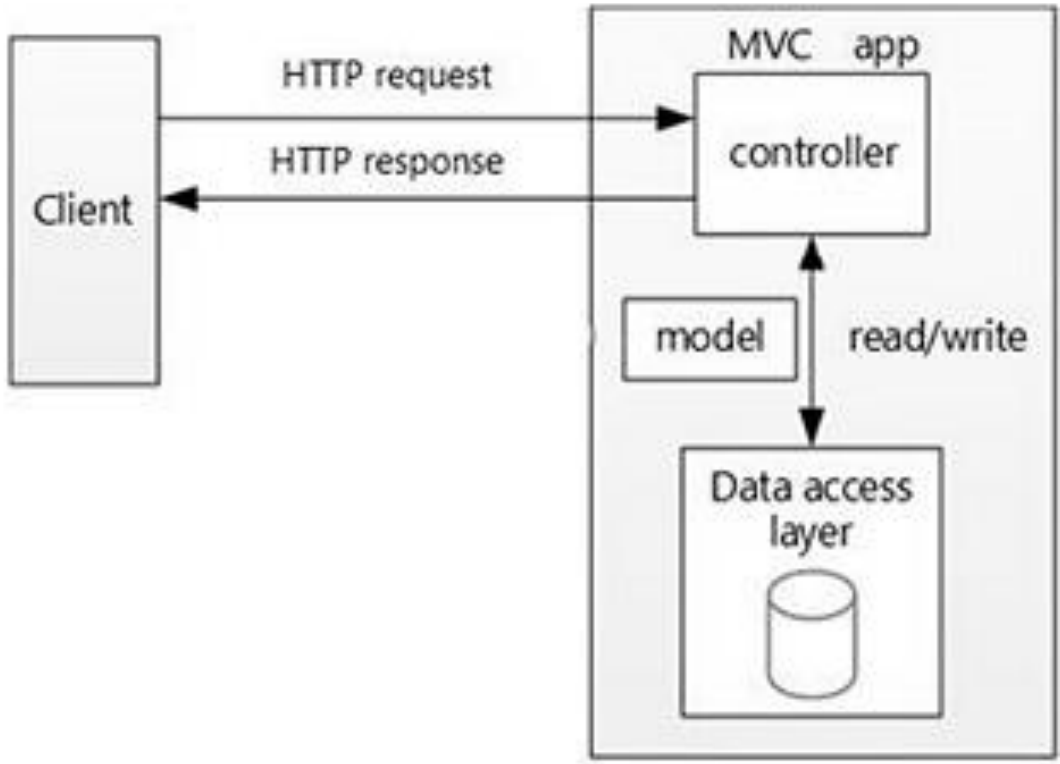
a) o model pode ser serializado para JSON e XML.

b) o client, que consome a Web API, será obrigatoriamente um navegador web.

c) só pode haver um controller para manipular solicitações HTTP e representar os dados.

d) o model é representado por classes VBScript conhecidas como Plain Old Objects (POOs).

e) o controller permite requisições HTTP somente por meio do método GET.



Q2) [FGV IMBEL 2021] No contexto da linguagem C#, analise as etapas do ciclo de vida de uma página ASP.NET, exibidos em ordem alfabética.

Initialization

Load

Page Request

Postback

Postback event handling

Rendering

Start

Unload

Dentre as etapas listadas acima, assinale as duas primeiras etapas, na ordem em que ocorrem.

a) Initialization e Load.

b) Load e Page Request.

c) Page Request e Start.

d) Postback e Rendering.

e) Start e Postback.

Q2) [FGV IMBEL 2021] No contexto da linguagem C#, analise as etapas do ciclo de vida de uma página ASP.NET, exibidos em ordem alfabética.

Initialization

Load

Page Request

Postback

Postback event handling

Rendering

Start

Unload

Dentre as etapas listadas acima, assinale as duas primeiras etapas, na ordem em que ocorrem.

a) Initialization e Load.

b) Load e Page Request.

c) Page Request e Start.

d) Postback e Rendering.

e) Start e Postback.

Q3) [VUNESP Prefeitura Ilhabela 2020] O modelo de desenvolvimento do ASP.NET que permite que o código associado (back end) fique em um arquivo separado da página ASP.NET (front end) é chamado de modelo

a) MVC.

b) In-Line.

c) Code-Behind.

d) Híbrido.

e) Two-tier.

Q3) [VUNESP Prefeitura Ilhabela 2020] O modelo de desenvolvimento do ASP.NET que permite que o código associado (back end) fique em um arquivo separado da página ASP.NET (front end) é chamado de modelo

a) MVC.

b) In-Line.

c) Code-Behind.

d) Híbrido.

e) Two-tier.

Q4) [INSTITUTO AOCP ADAF-AM 2018] O ASP.NET faz parte da estrutura .NET e, por isso, aproveita ao máximo os recursos da programação orientada a objetos (OOP). Assim, sobre Linguagens de Programação .NET, é correto afirmar que, para iniciar um projeto em ASP.NET, é necessário criar um Web Form, que significa cria

a) um MVC.

b) uma página web.

c) um controlador de acesso..

d) uma classe.

e) um código da Internet.

Q4) [INSTITUTO AOCP ADAF-AM 2018] O ASP.NET faz parte da estrutura .NET e, por isso, aproveita ao máximo os recursos da programação orientada a objetos (OOP). Assim, sobre Linguagens de Programação .NET, é correto afirmar que, para iniciar um projeto em ASP.NET, é necessário criar um Web Form, que significa cria

a) um MVC.

b) uma página web.

c) um controlador de acesso..

d) uma classe.

e) um código da Internet.

Q5) [VUNESP Prefeitura Presidente Prudente 2016] O componente da arquitetura MVC do ASP.NET responsável por manipular as interações do usuário é

a) a aplicação (application).

b) o controlador (controller).

c) o modelo (model).

d) a página (page).

e) a visualização (view).

Q5) [VUNESP Prefeitura Presidente Prudente 2016] O componente da arquitetura MVC do ASP.NET responsável por manipular as interações do usuário é

a) a aplicação (application).

b) o controlador (controller).

c) o modelo (model).

d) a página (page).

e) a visualização (view).