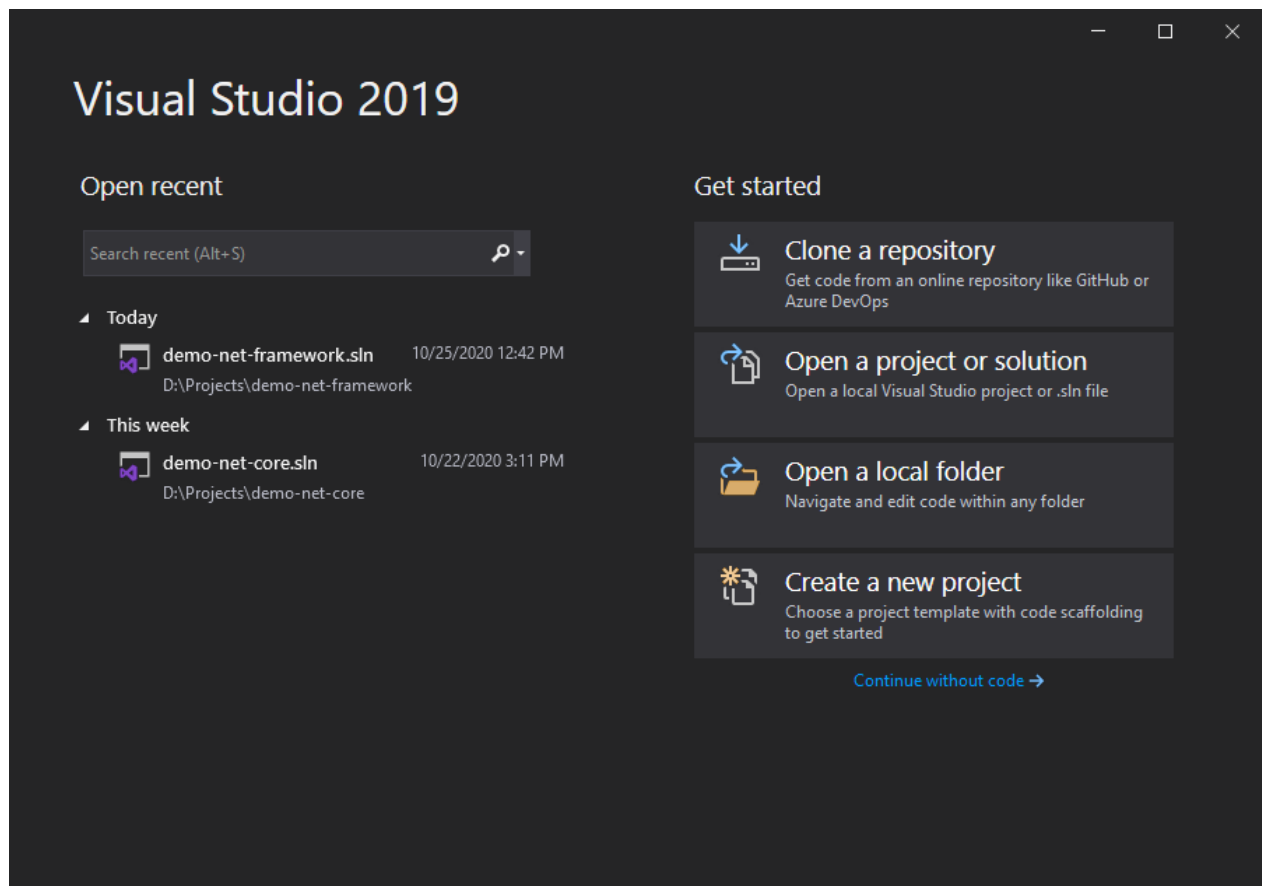


- **GHID – Structura unei aplicații web - MVC**

- **Cum realizăm o nouă aplicație web folosind Visual Studio?**

Primul pas constă în deschiderea programului Visual Studio. Dacă aveți un proiect deja deschis puteți realiza un nou proiect folosind următoarea secvență de butoane/comenzi: **“File -> New -> Project”**.

Din următoarea fereastră deschisă avem multiple opțiuni, printre care ***deschiderea unui proiect recent***, ***clonarea unui repository***, ***deschiderea unei soluții/unui proiect***, respectiv ***crearea de proiecte/soluții***.

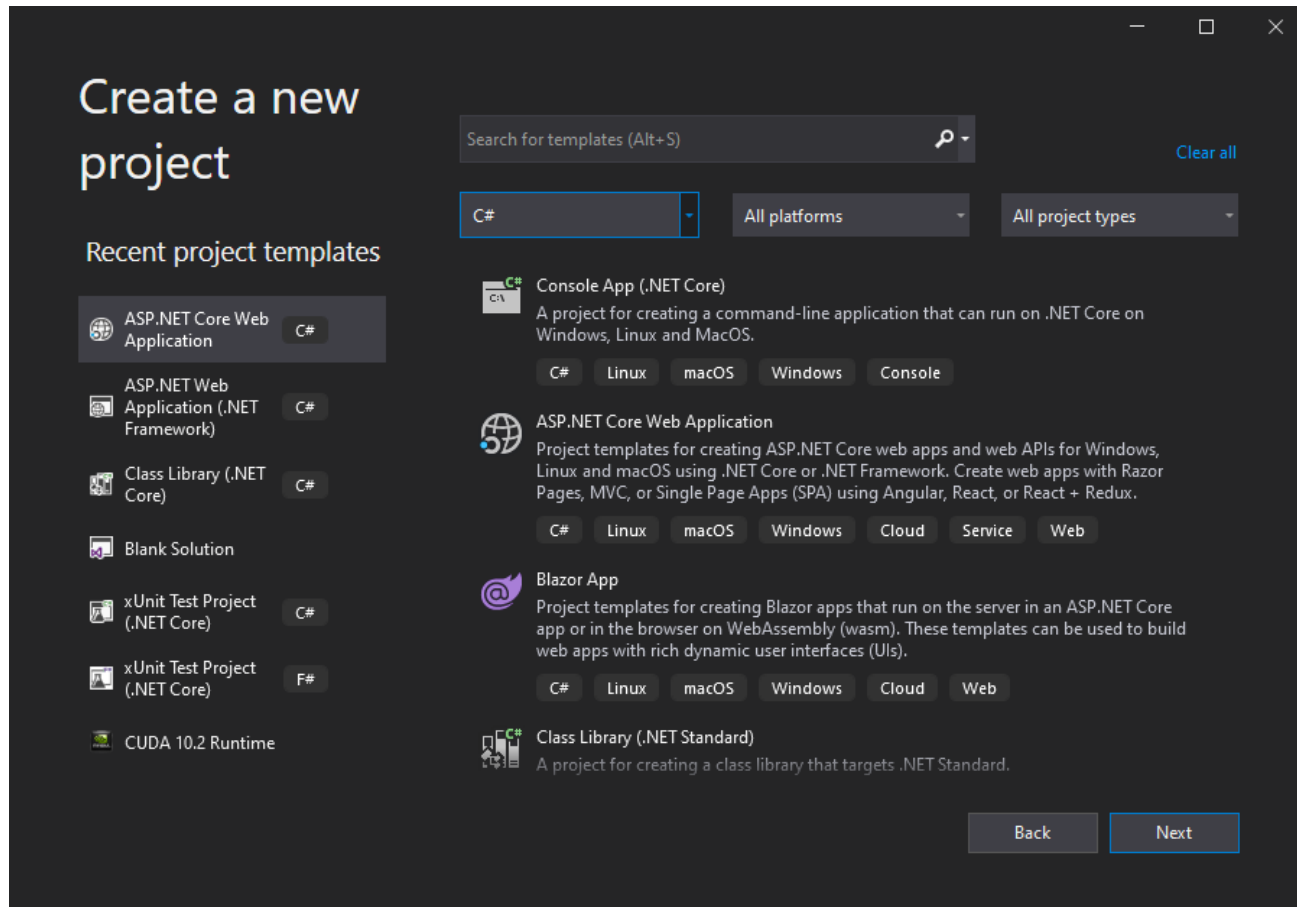


În exemplul următor vom realiza un nou proiect și vom ajunge la caseta de dialog din imaginea de mai jos. Opțiunile prezente sunt diverse, cele mai importante fiind:

1. **Selecția unui template utilizat recent;**
2. **Selectarea limbajului, platformei sau tipului de proiect;**
3. **Căutarea în funcție de numele șablonului sau a etichetelor.**

- **GHID – Structura unei aplicații web - MVC**

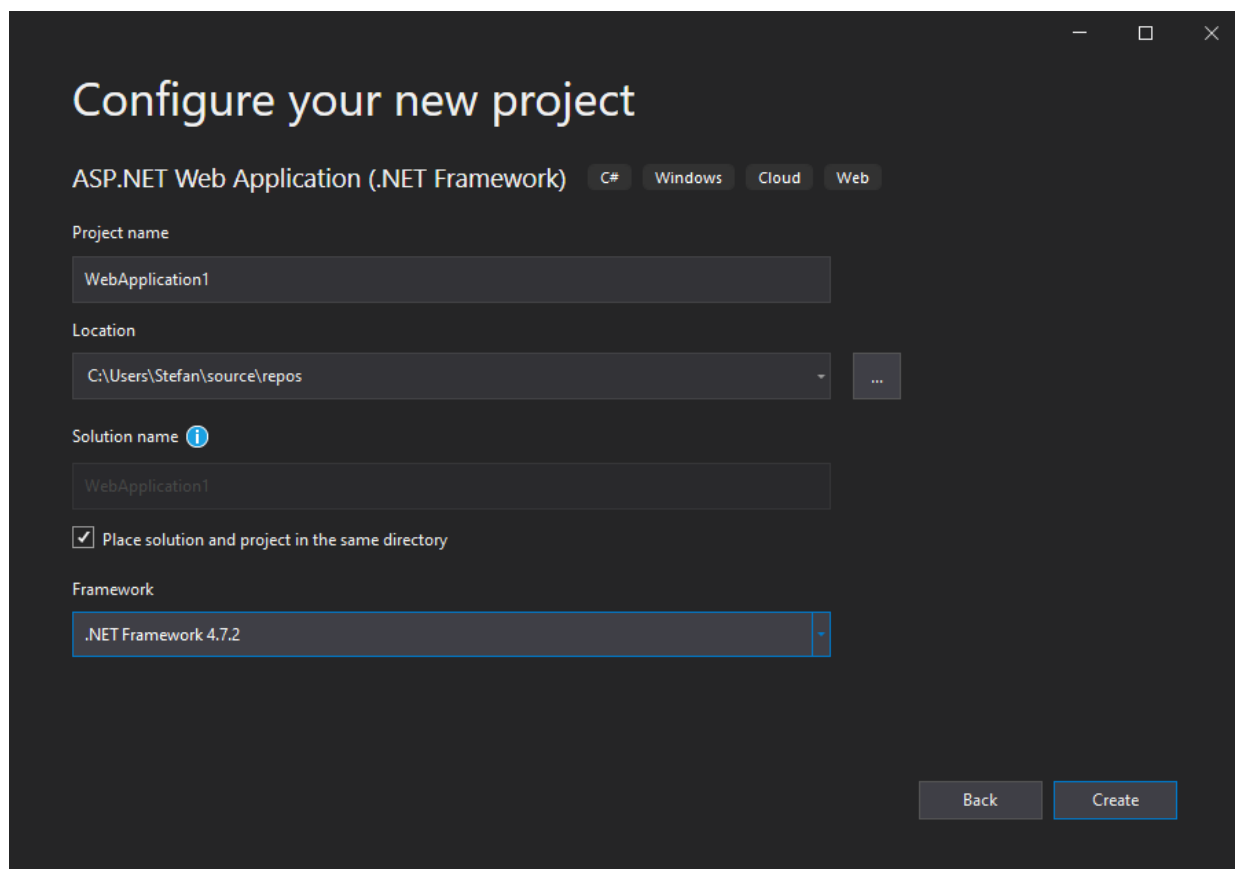
Pentru exemplele noastre vom utiliza șabloanele: **“ASP.NET Web Application”/“ASP.NET Core Web Application”**.



Următorul pas constă în configurarea unor proprietăți generale ale proiectului: **nume proiect, locație, nume soluție, framework**.

⚡ **Recomand utilizarea opțiunii “Place solution and project in the same directory” pentru proiectele de demo/simple.**

- **GHID – Structura unei aplicații web - MVC**




Configure your new project

ASP.NET Web Application (.NET Framework) C# Windows Cloud Web

Project name
WebApplication1

Location
C:\Users\Stefan\source\repos

Solution name 
WebApplication1

☒ Place solution and project in the same directory

Framework
.NET Framework 4.7.2

Back Create

 **Următorul pas este cel mai important!**

Avem următoarele posibilități de selecție, **în funcție de framework-ul selectat (.NET/.NET Core)**:

1. Tipul proiectului:

- Empty** – utilizat pentru realizarea de servicii personalizate, unde arhitectura diferă de cele clasice.
- Web Forms** – tipul acesta de proiect este disponibil doar pentru .NET (versiunea clasică) și presupune realizarea proiectelor folosind diverse elemente/componente (drag-and-drop), respective evenimente. Este un mod eficient de a realiza aplicații web rapide și dinamice, fără a complica procesul de dezvoltarea.
- MVC** – Proiectele MVC sunt create direct cu o structură de bază (pagină Home, câteva modele simple și controller default). Este o arhitectură consacrată în mediul ASP.NET.

- **GHID – Structura unei aplicații web - MVC**

Șablonul este regăsit în *.NET Core* sub denumirea de **“Web Application (Model-View-Controller)”**

- d. **API/Web API** – Cu ajutorul acestui tip de soluție realizăm servicii de tip **RESTful HTTP (Representational state transfer)**. Tiparul API este utilizat în special pentru servirea unor date în format cât mai simplu (JSON, XML). În *.NET Core* putem combina utilizarea **pattern-ului MVC cu formatul API, diferențierea fiind făcută la nivel de controller**.
- e. **Single Page Application/Angular/React.js/React.js and Redux** – Atunci când este imperios necesară dezvoltarea unei aplicații de tipul SPA este bine de știut faptul că avem la dispoziție câteva posibilități, direct din interfața Visual Studio.

2. Setări avansate:

- a. **Configurare HTTPS**
- b. **Suport pentru Docker**
- c. **Creare de proiect pentru teste unitare (.NET)**
- d. **Razor runtime compilation (.NET Core)** – opțiune utilă pentru încărcarea în timp-real, după fiecare modificare a fișierelor Razor (HTML + CSS + JS + C#). ⚡ **FOARTE UTILĂ!!!**

3. Autentificare:

- a. **No authentication**
- b. **Individual User Accounts** – folosită în cazul schemelor clasice de autentificare: utilizator + parolă
- c. **Work or School Accounts** – integrare cu **Azure Active Directory**, o soluție cloud pentru gestionarea utilizatorilor dintr-o organizație. ⚡ **Conturile @s.unibuc.ro sunt gestionate folosind modulul menționat.**
- d. **Windows Authentication** – Foarte folositoare atunci când aplicațiile sunt utilizat intern, în cadrul unei companii.

- GHID – Structura unei aplicații web - MVC

Create a new ASP.NET Web Application

Empty

An empty project template for creating ASP.NET applications. This template does not have any content in it.

Web Forms

A project template for creating ASP.NET Web Forms applications. ASP.NET Web Forms lets you build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

MVC

A project template for creating ASP.NET MVC applications. ASP.NET MVC allows you to build applications using the Model-View-Controller architecture. ASP.NET MVC includes many features that enable fast, test-driven development for creating applications that use the latest standards.

Web API

A project template for creating RESTful HTTP services that can reach a broad range of clients including browsers and mobile devices.

Single Page Application

A project template for creating rich client side JavaScript driven HTML5 applications using ASP.NET Web API. Single Page Applications provide a rich user experience which includes client-side interactions using HTML5, CSS3, and JavaScript.

Authentication
 No Authentication
[Change](#)

Add folders & core references
☐ Web Forms
☒ MVC
☐ Web API

Advanced
☒ Configure for HTTPS
☐ Docker support
 (Requires [Docker Desktop](#))
☐ Also create a project for unit tests
 WebApplication1.Tests

[Back](#)
[Create](#)

Create a new ASP.NET Core web application

.NET Core
 ASP.NET Core 3.1

Empty

An empty project template for creating an ASP.NET Core application. This template does not have any content in it.

API

A project template for creating an ASP.NET Core application with an example Controller for a RESTful HTTP service. This template can also be used for ASP.NET Core MVC Views and Controllers.

Web Application

A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.

Web Application (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

Angular

A project template for creating an ASP.NET Core application with Angular

React.js

A project template for creating an ASP.NET Core application with React.js

React.js and Redux

A project template for creating an ASP.NET Core application with React.js and Redux

Authentication
 No Authentication
[Change](#)

Advanced
☒ Configure for HTTPS
☐ Enable Docker Support
 (Requires [Docker Desktop](#))
 Linux
☐ Enable Razor runtime compilation

[Back](#)
[Create](#)

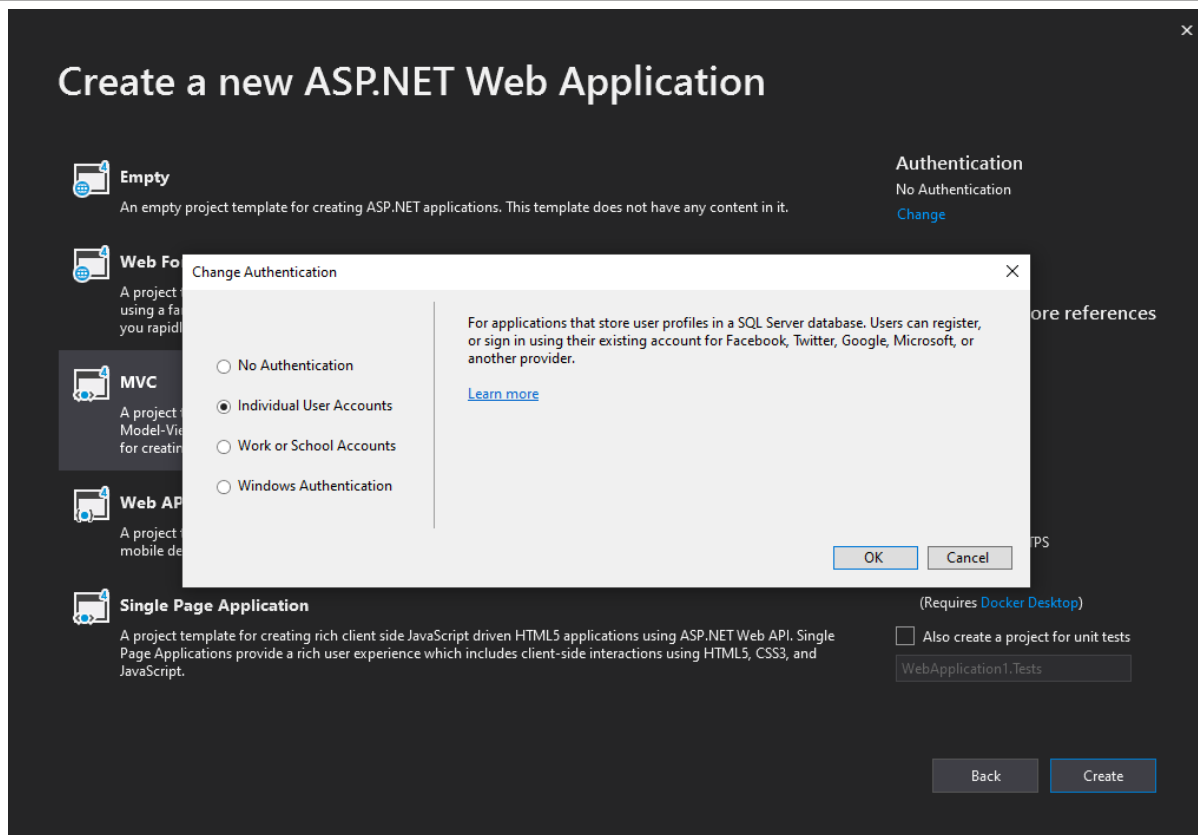
Author: Microsoft
Source: Templates 3.1.10

[Get additional project templates](#)

[Outlook](#) | [Gmail](#) | [LinkedIn](#) | [GitHub](#)

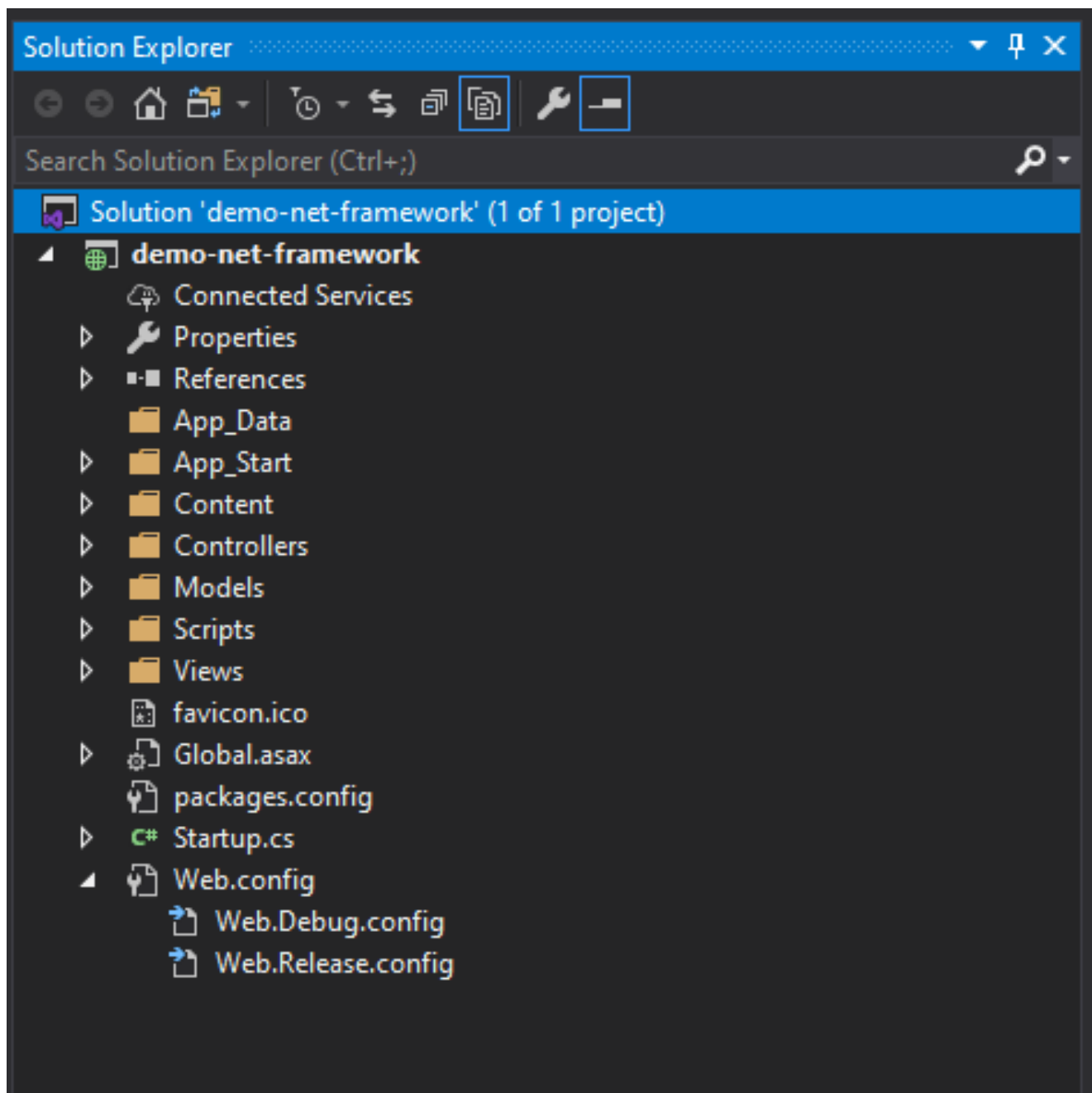
5

- GHID – Structura unei aplicații web - MVC



- GHID – Structura unei aplicații web - MVC

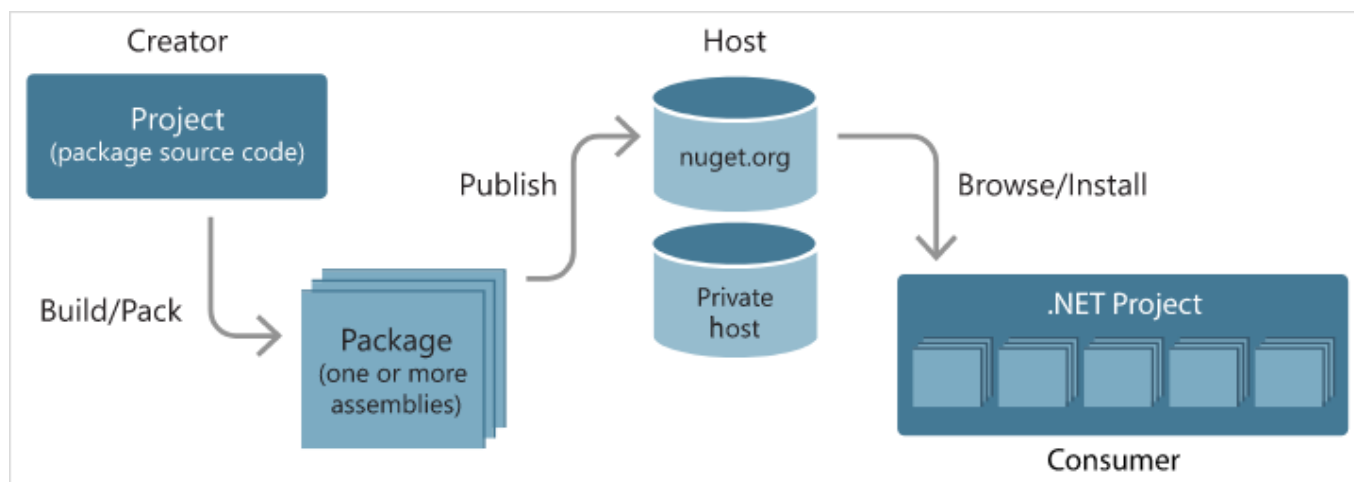
- Structura unei aplicații .NET



La o primă vedere o aplicație MVC este complexă, în schimb segmentarea și granularizarea componentelor este un principiu și o practică bună în industrie, acolo unde se preferă o separare cât mai detaliată a logicii de business sau a logicii codului scris.

● GHID – Structura unei aplicații web - MVC

- **Connected Services** – reprezintă o colecție de servicii terțe ce pot fi asociate unei aplicații web.
- **Properties** – conține detalii generale ale proiectului, unele dintre ele fiind configurate la crearea proiectului.
- **References** – În acest segment avem o listă cu toate librăriile utilizate de aplicația noastră, majoritatea provenind din **NuGet**, “**organizatorul de pachete**” **Microsoft**. Următoarea diagramă ne arată parcursul, etapizat, al unei astfel de librării.



- **App_Data** – Aici putem menține fișiere ce deservește datele conținute de aplicația noastră, fie ele structurate sau nu. Putem plasa fișiere de tip .XML, .JSON, .MDF (structură și informații ale unei baze de date), plus alte fișiere diverse (la alegerea dezvoltatorilor).
- **App_Start** – În acest spațiu sunt stocate câteva fișiere C# dedicate configurării aplicației. Spre exemplu, fișierul **RouteConfig.cs** pune la dispoziție programatorilor opțiunea de a configura rute și adrese în cadrul serviciului sau aplicației. Un alt exemplu îl reprezintă **BundleConfig.cs**, un configurator dedicat structurării fișierelor .CSS și .JS în pachete, denumite “*bundles*”. ⚡ **Toate aceste configurări sunt descoperite pe parcurs, în funcție de nevoi. Pentru moment utilizarea lor în forma dată este suficientă.**
- **Content** – Aici regăsim toate fișierele dedicate stilizării paginilor (fișiere .CSS, .SCSS, .JPG, .PNG, .GIF, etc.)
- **Controllers** – Dacă folosim MVC aici vor fi stocate unitățile de cod de tipul controller, convenția fiind una destul de limpede: toate fișierele au drept suffix cuvântul “**Controller**”.
- **Models** – Logica business, respectiv logica datelor din spatele unei aplicații se află în acest director.

- **GHID – Structura unei aplicații web - MVC**

- **Scripts** – Fișiere/biblioteci/componente JavaScript
- **Views** – Paginile cu extensia .CSHTML, respective .HTML se vor afla în acest folder. Foarte importante sunt următoarele view-uri/interfețe, de bază:
 - ♦ **_ViewStart.cshtml** -> conține definirea componentei Layout (un șablon generic al interfeței de utilizare)
 - ♦ **Shared/_Layout.cshtml** -> aici definim formatul UI, pornind de la simplul tag <html> până la integrarea elementelor statice de stilizare și a funcționalităților complexe, expuse prin JavaScript. ⚡ **Foarte importantă este secvența de cod `@RenderBody()`. Aceasta este componenta care permite încărcarea conținutului atunci când vizităm o pagină nouă.**
- **favicon.ico** – Iconița aplicației (există una “by default”)
- **Global.asax** – Aici putem defini cod declanșat de anumite evenimente de sistem:
 - ♦ *Application_Start/End*
 - ♦ *Session_Start/End*
 - ♦ *Application_BeginRequest*
 - ♦ *Application_AuthenticateRequest*
 - ♦ *Application_Error*
- **Startup.cs** – Fișier vital al oricărei aplicații MVC scrisă folosind .NET framework, mai precis un fișier de configurare
- **Web.config** – Prezent în două moduri (*Debug & Release*), în format XML. Constă în setări moleculare ale aplicației (framework utilizat, conexiune cu baze de date, biblioteci & module încărcate). 🚫 **NU NE ATINGEM DE ACEST FIȘIER ÎN MOD MANUAL, EXCEPȚIE FIIND ADĂUGAREA SAU MODIFICAREA CONEXIUNILOR CU SURSELE DE DATE.**