Napredne tehnike i principi razvoja softvera
Razvoj web servisa i serverske strane web aplikacija

TEMA 1:

WEB ARHITEKTURE I ASP.NET Web API

MSc Ivan Vasiljević, ivan.vasiljevic@hotmail.com
MSc Mladen Kanostrevac mkanostrevac@gmail.com

Pregled

- Uvod
- Arhitektura web sistema
  - Principi klijent-server arhitektura
  - Protokol HTTP
  - Arhitektura REST
  - *ASP.NET Web API* i funkcionalni nivoi *ASP.NET Web API* aplikacija
- *ASP.NET Web API* osnovi
  - Prvi projekat

UVOD

Predavači i alati

Predavači

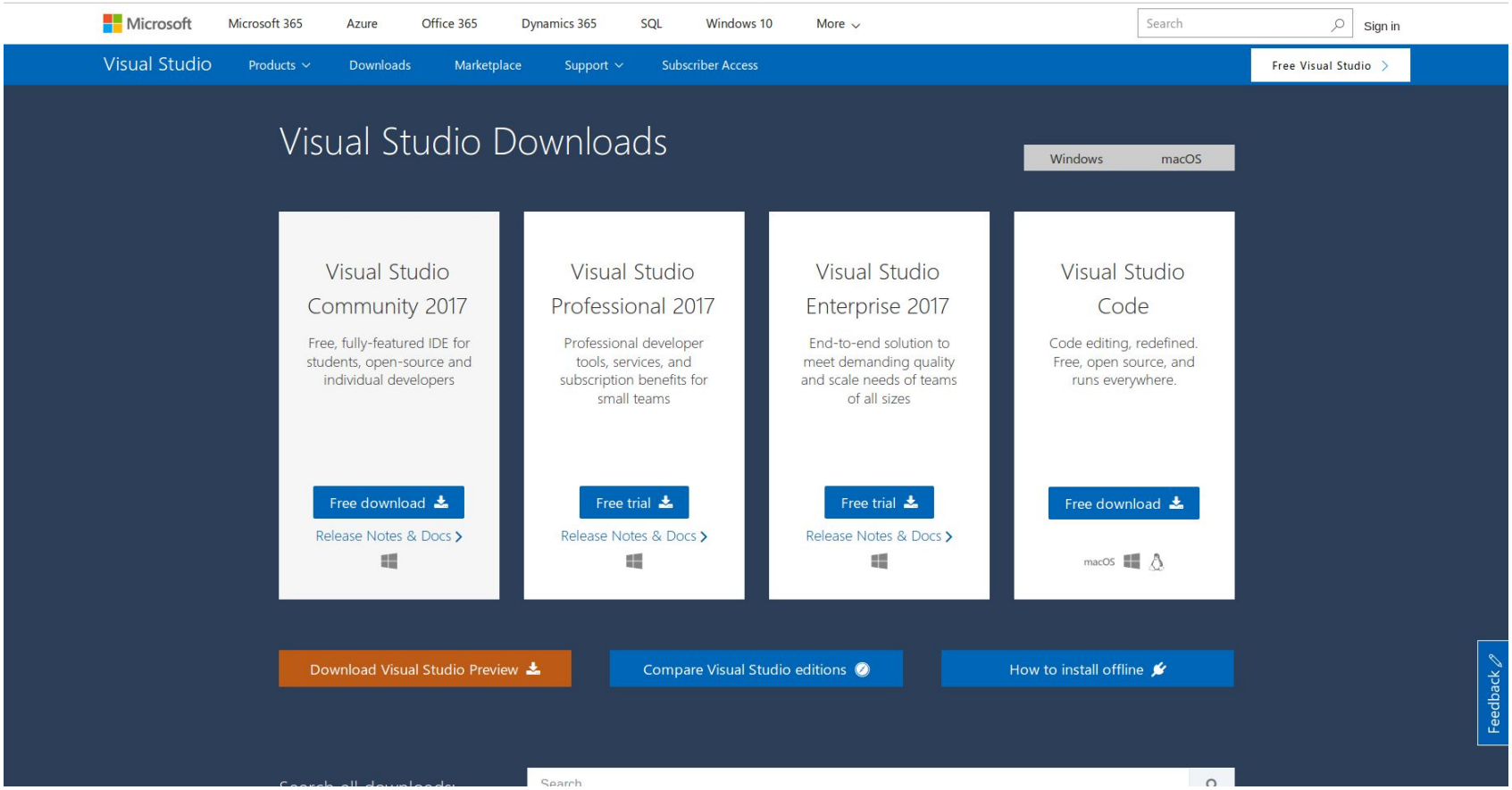
- Mladen Kanostrevac
  - full-stack developer
    - frontend, backend i baze podataka
  - 5+ godina iskustva u razvoju web aplikacija
  - software developer u Vivify Ideas
- Ivan Vasiljević
  - full-stack developer
    - frontend, backend i baze podataka
  - 5+ godina iskustva u razvoju web aplikacija
  - software developer u Execom-u

Potrebni alati

- Visual Studio 2017 Community Edition
  - alat namenjen za razvoj aplikacija za .NET platformu
    - Desktop - Windows Forms, WPF, UWP
    - Web - ASP.NET, ASP.NET MVC, ASP.NET Core
  - sadrži ugrađen aplikativni server IIS Express
    - olakšava testiranje web aplikacija
  - sadrži ugrađene alate za rad sa bazama podataka
    - MSSQL i MSSQL Express
- Git for Windows
  - alat za rad sa git repozitorijumima
    - verzionisanje koda i kolaboracija
- Postman
  - alat za testiranje REST servisa

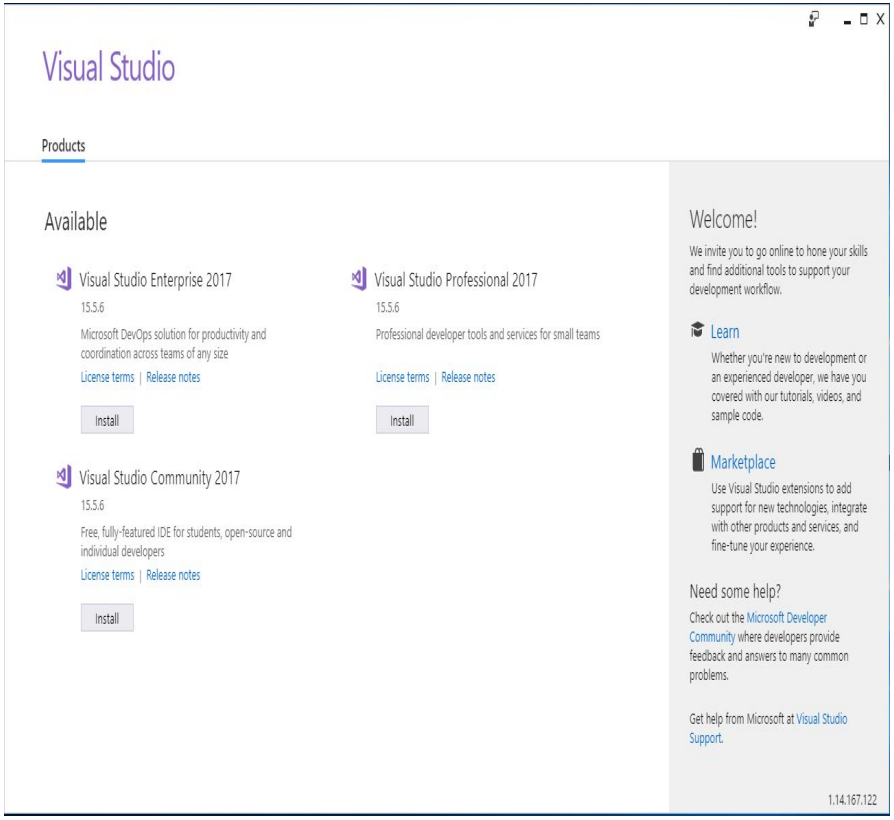
Visual Studio 2017 Community Edition

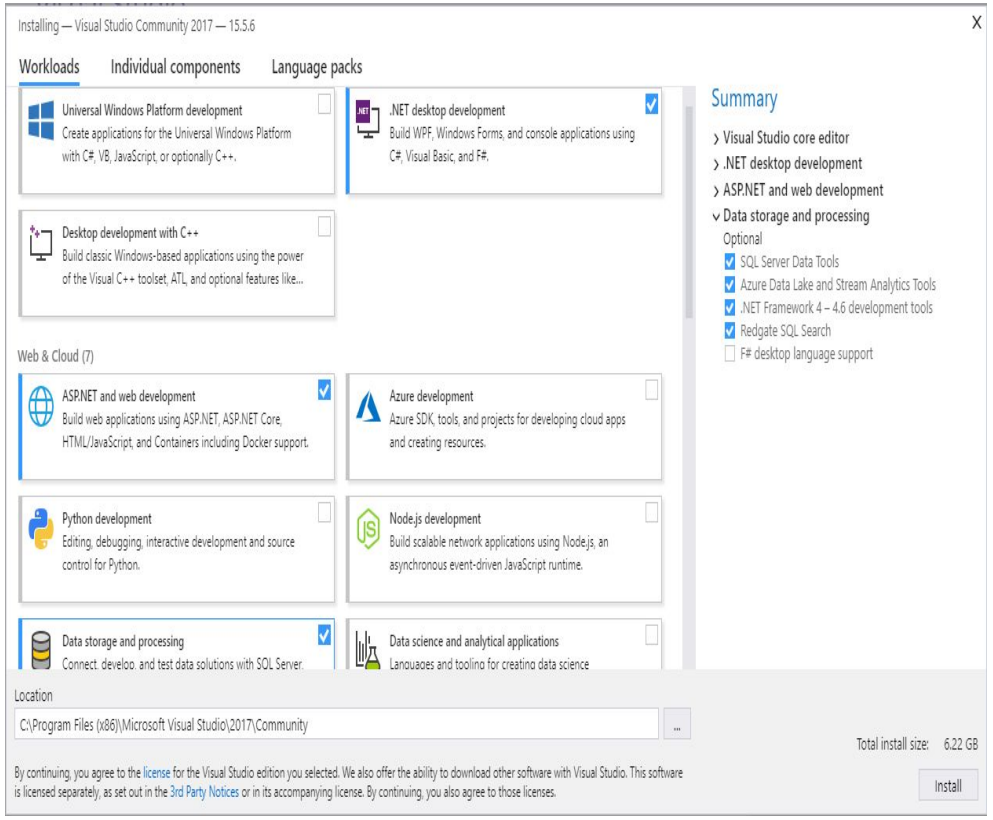
- Preuzeti installer
  - https://www.visualstudio.com/downloads/



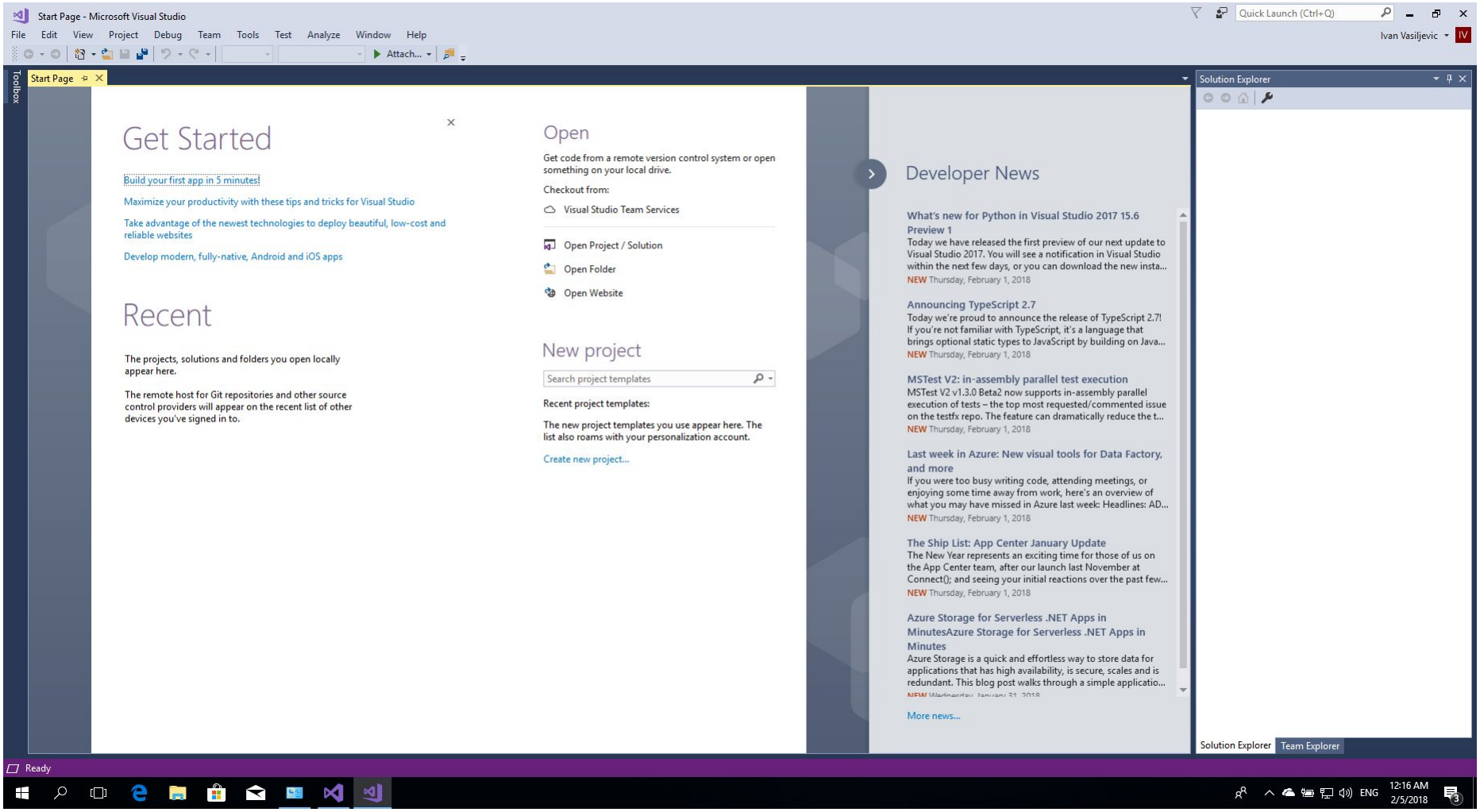
Visual Studio 2017 Community Edition

- Pokrenuti installer
- Installirati Visual Studio Community 2017





Visual Studio 2017 Community Edition



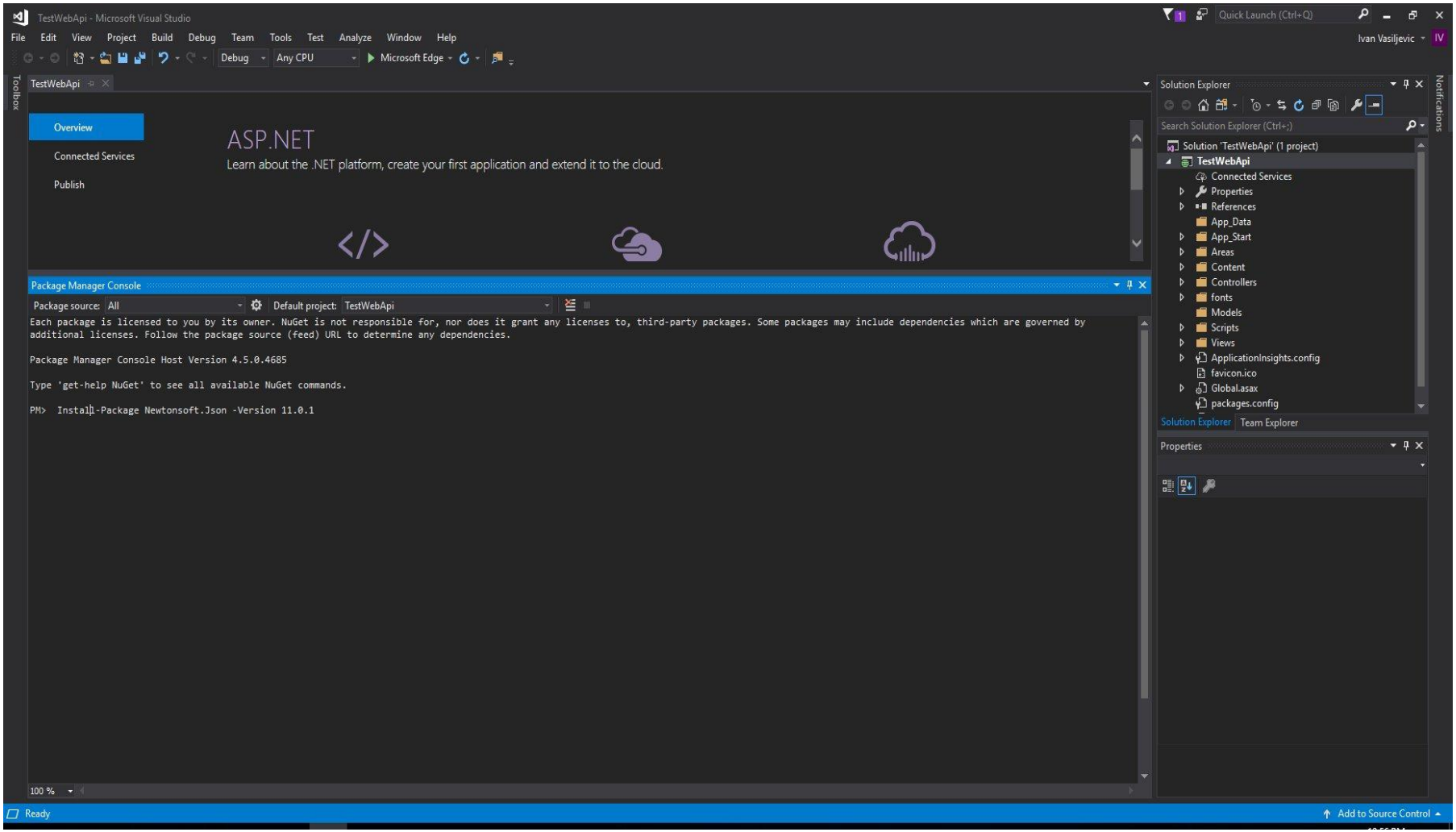
NuGet

- Esencijalni alat za:
  - kreiranje,
  - deljenje
  - korišćenje korisnog koda
- Koristan kod je zapakovan u pakete
- Paketi sadrže kompajlirani kod, tj. DLL-ove
- Paketi mogu da se dele na:
  - privatnim
  - javnim serverima

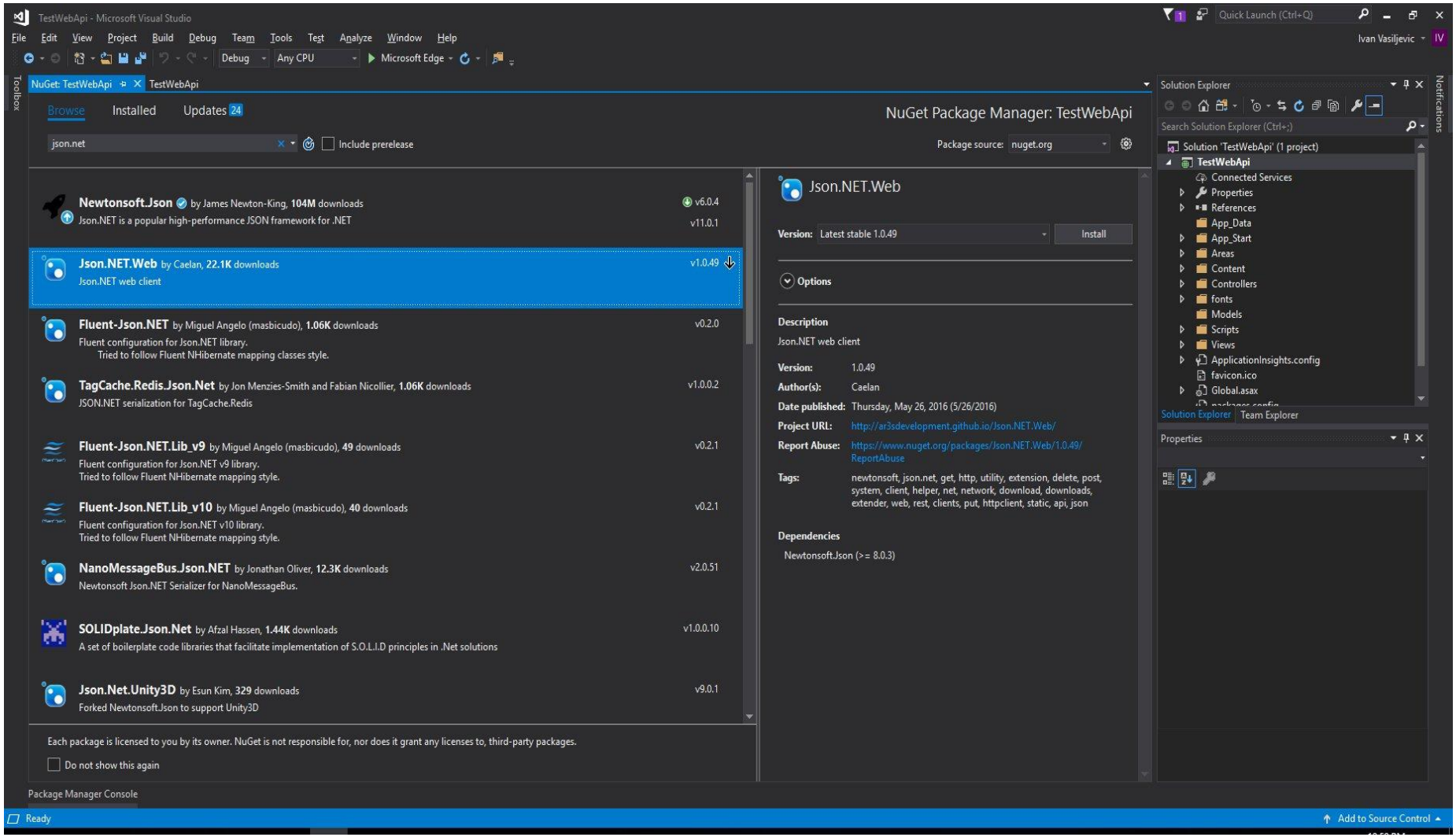
NuGet

- Razrešenje zavisnosti između softverskih biblioteka
  - specificiranjem naziva i verzije potrebne softverskog paketa (biblioteke)
  - najveći sajt sa javnim paketima: https://www.nuget.org/
  - NuGet automatski preuzima dati paket
  - i tranzitivno, sve druge, potrebne pakete
  - koristi se iz:
    - komandne linije
    - direktno iz Visual Studija
- Dolazi preinstaliran sa Visual Studijom

NuGet

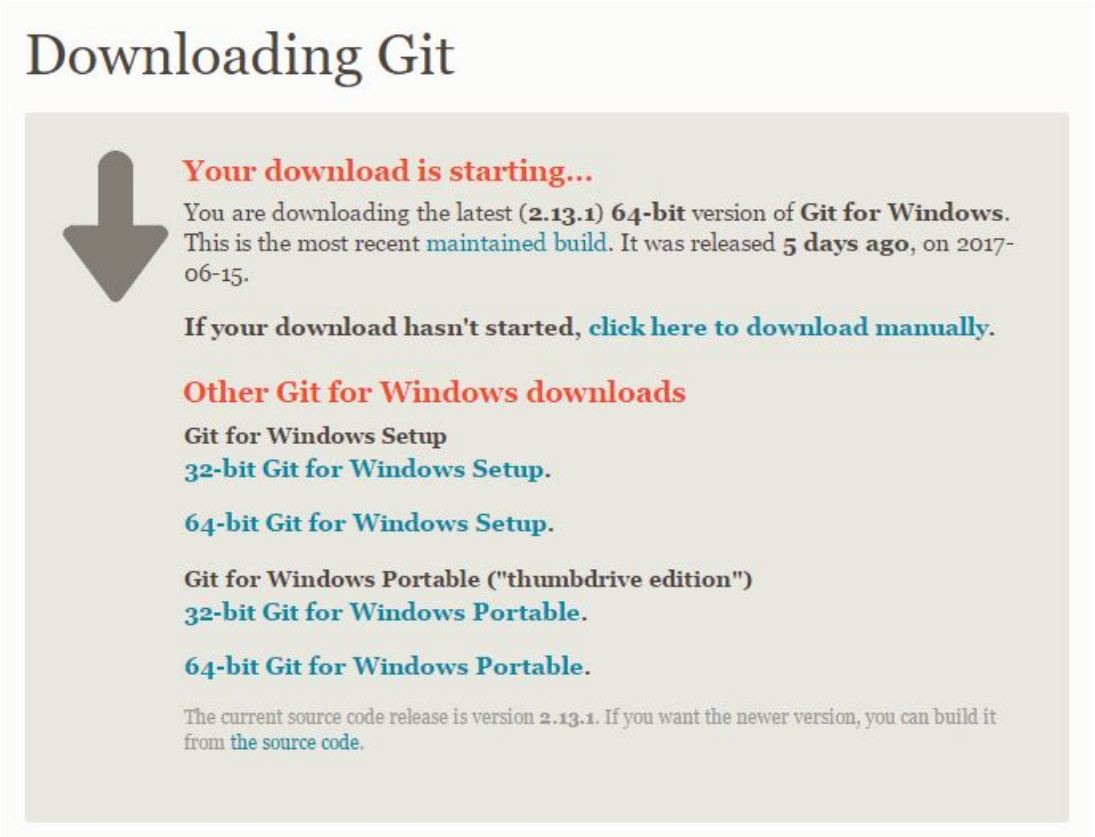

TestWebApi - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU Microsoft Edge
Ivan Vasiljevic
Quick Launch (Ctrl+Q)
Toolbox
TestWebApi
Overview
Connected Services
Publish
ASP.NET
Learn about the .NET platform, create your first application and extend it to the cloud.
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TestWebApi' (1 project)
TestWebApi
Connected Services
Properties
References
App_Data
App_Start
Areas
Content
Controllers
fonts
Models
Scripts
Views
ApplicationInsights.config
favicon.ico
Global.asax
packages.config
Solution Explorer Team Explorer
Properties
Notifications
Package Manager Console
Package source: All
Default project: TestWebApi
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages. Some packages may include dependencies which are governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.
Package Manager Console Host Version 4.5.0.4685
Type 'get-help NuGet' to see all available NuGet commands.
PM> Install-Package Newtonsoft.Json -Version 11.0.1
100 %
Ready
Add to Source Control

NuGet


TestWebApi - Microsoft Visual Studio
File Edit View Project Build Debug Team Tools Test Analyze Window Help
Debug Any CPU Microsoft Edge
Quick Launch (Ctrl+Q)
Ivan Vasiljevic
NuGet: TestWebApi
TestWebApi
Browse Installed Updates 24
NuGet Package Manager: TestWebApi
json.net
Include prerelease
Package source: nuget.org
Newtonsoft.Json by James Newton-King, 104M downloads v6.0.4
Json.NET is a popular high-performance JSON framework for .NET v11.0.1
Json.NET.Web by Caelan, 22.1K downloads v1.0.49
Json.NET web client
Fluent-Json.NET by Miguel Angelo (masbicudo), 1.06K downloads v0.2.0
Fluent configuration for Json.NET library.
Tried to follow Fluent NHibernate mapping classes style.
TagCache.Redis.Json.Net by Jon Menzies-Smith and Fabian Nicollier, 1.06K downloads v1.0.0.2
JSON.NET serialization for TagCache.Redis
Fluent-Json.NET.Lib_v9 by Miguel Angelo (masbicudo), 49 downloads v0.2.1
Fluent configuration for Json.NET v9 library.
Tried to follow Fluent NHibernate mapping style.
Fluent-Json.NET.Lib_v10 by Miguel Angelo (masbicudo), 40 downloads v0.2.1
Fluent configuration for Json.NET v10 library.
Tried to follow Fluent NHibernate mapping style.
NanoMessageBus.Json.NET by Jonathan Oliver, 12.3K downloads v2.0.51
Newtonsoft Json.NET Serializer for NanoMessageBus.
SOLIDplate.Json.Net by Afzal Hassen, 1.44K downloads v1.0.0.10
A set of boilerplate code libraries that facilitate implementation of S.O.L.I.D principles in .Net solutions
Json.Net.Unity3D by Esun Kim, 329 downloads v9.0.1
Forked Newtonsoft.Json to support Unity3D
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.
Do not show this again
Json.NET.Web
Version: Latest stable 1.0.49
Install
Options
Description
Json.NET web client
Version: 1.0.49
Author(s): Caelan
Date published: Thursday, May 26, 2016 (5/26/2016)
Project URL: http://ar3sdevelopment.github.io/Json.NET.Web/
Report Abuse: https://www.nuget.org/packages/Json.NET.Web/1.0.49/ReportAbuse
Tags: newtonsoft, json.net, get, http, utility, extension, delete, post, system, client, helper, net, network, download, downloads, extender, web, rest, clients, put, httpclient, static, api, json
Dependencies
Newtonsoft.Json (>= 8.0.3)
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'TestWebApi' (1 project)
TestWebApi
Connected Services
Properties
References
App_Data
App_Start
Areas
Content
Controllers
fonts
Models
Scripts
Views
ApplicationInsights.config
favicon.ico
Global.asax
Solution Explorer Team Explorer
Properties
Package Manager Console
Ready
Add to Source Control

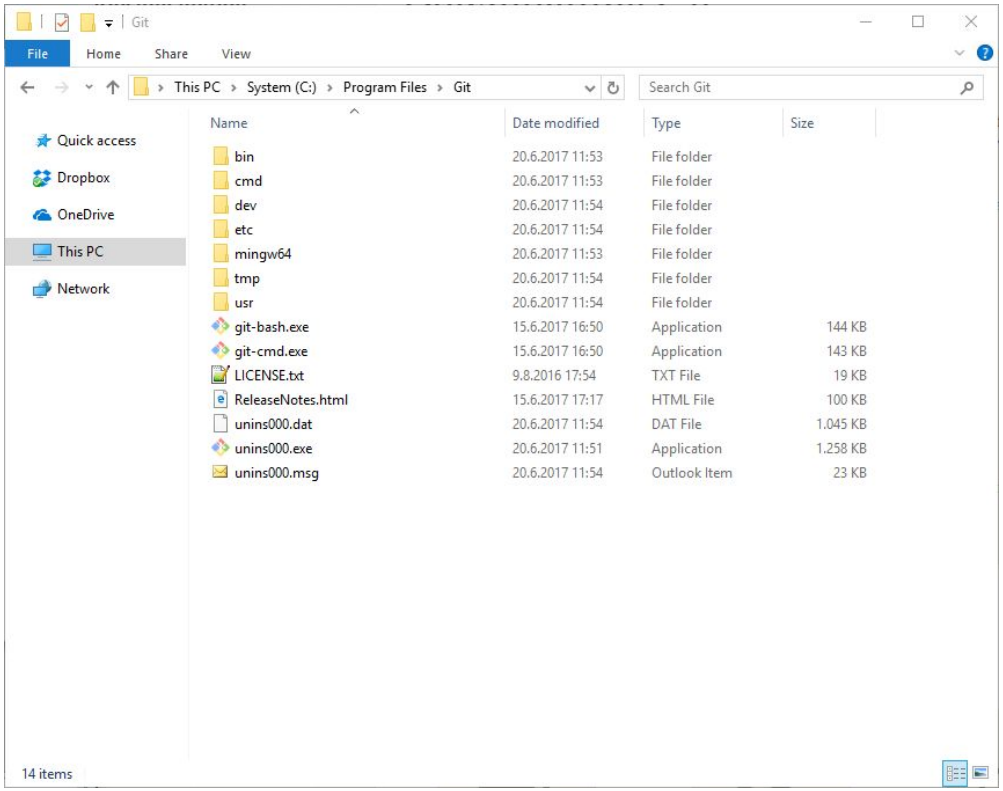
Git for Windows

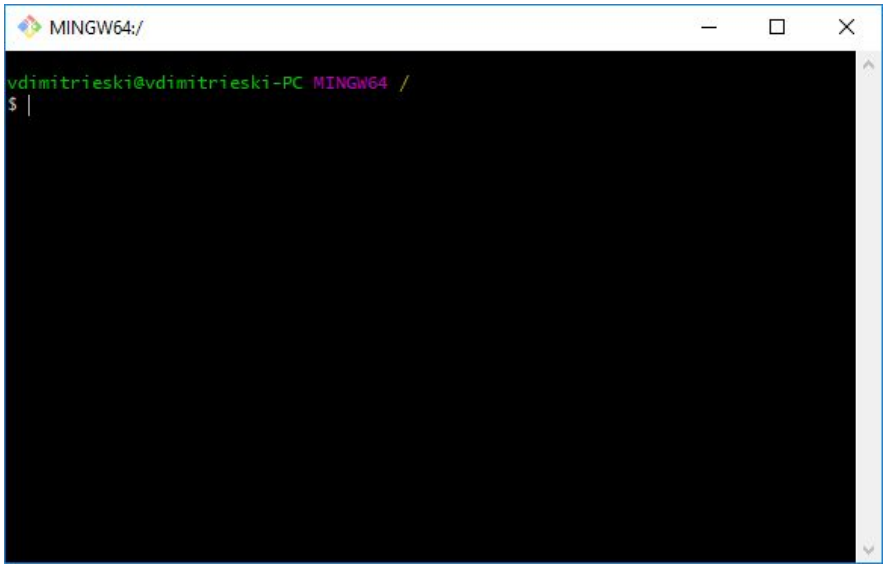
- Alat za rukovanje git sistemom za verzionisanje koda
- Preuzeti fajl za instalaciju
  - https://git-scm.com/download/win



Git for Windows

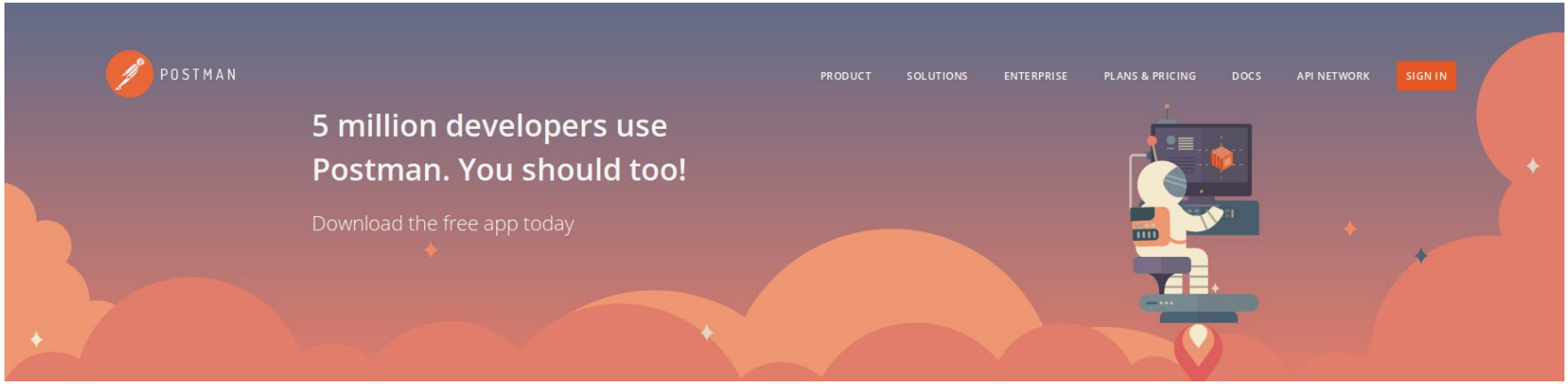
- Ispratiti korake intalacije
  - sve ponuđene vrednosti parametara ostaviti neizmenjene
- Pokrenuti **git-bash.exe**





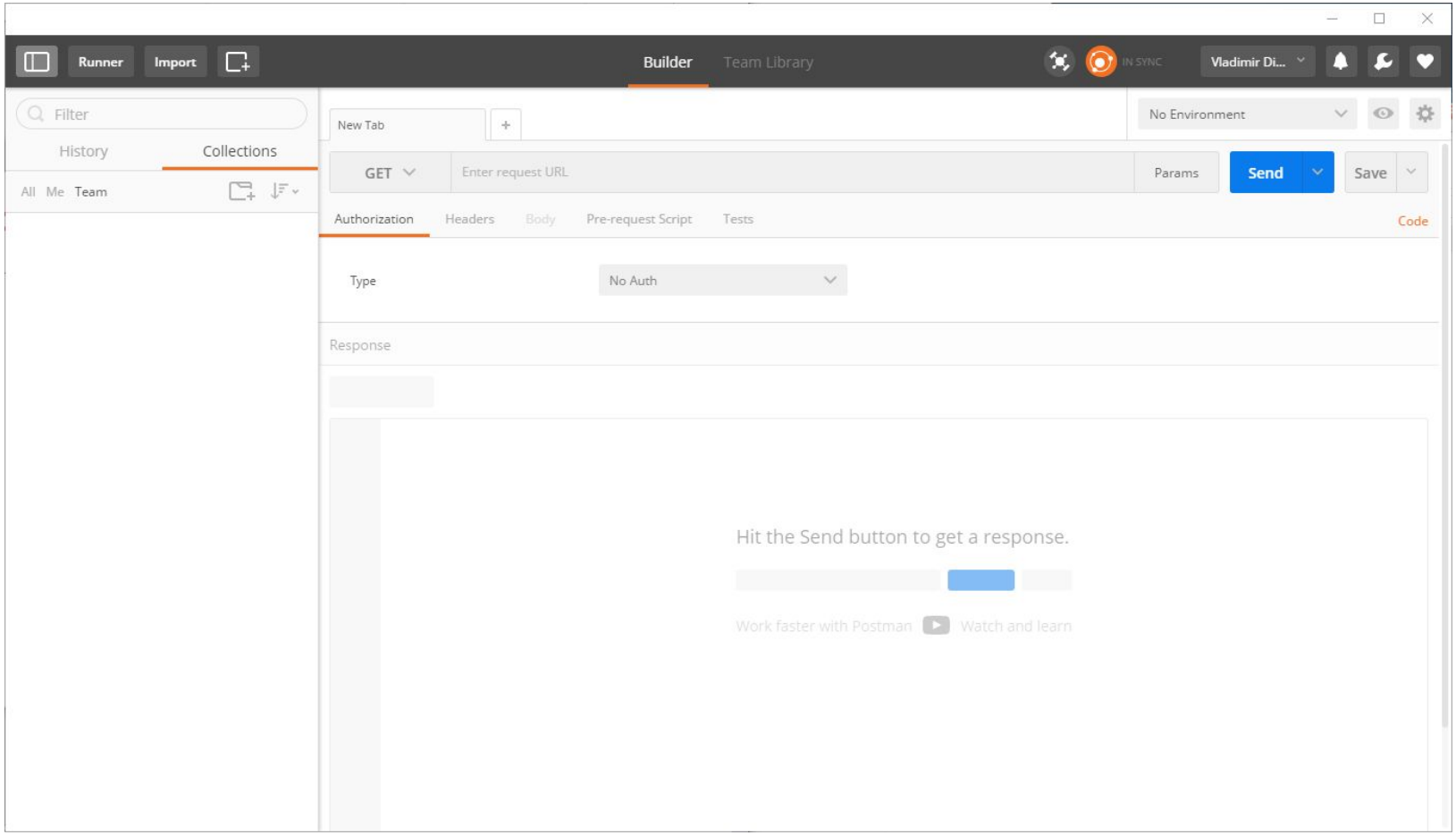
Postman

- Alat za testiranje REST servisa
  - alat za testiranje REST servisa
- Instalirati nakon preuzimanja instalera za odgovarajući operativni sistem sa adrese
  - https://www.getpostman.com/apps



Postman

- Pokrenuti nakon instalacije



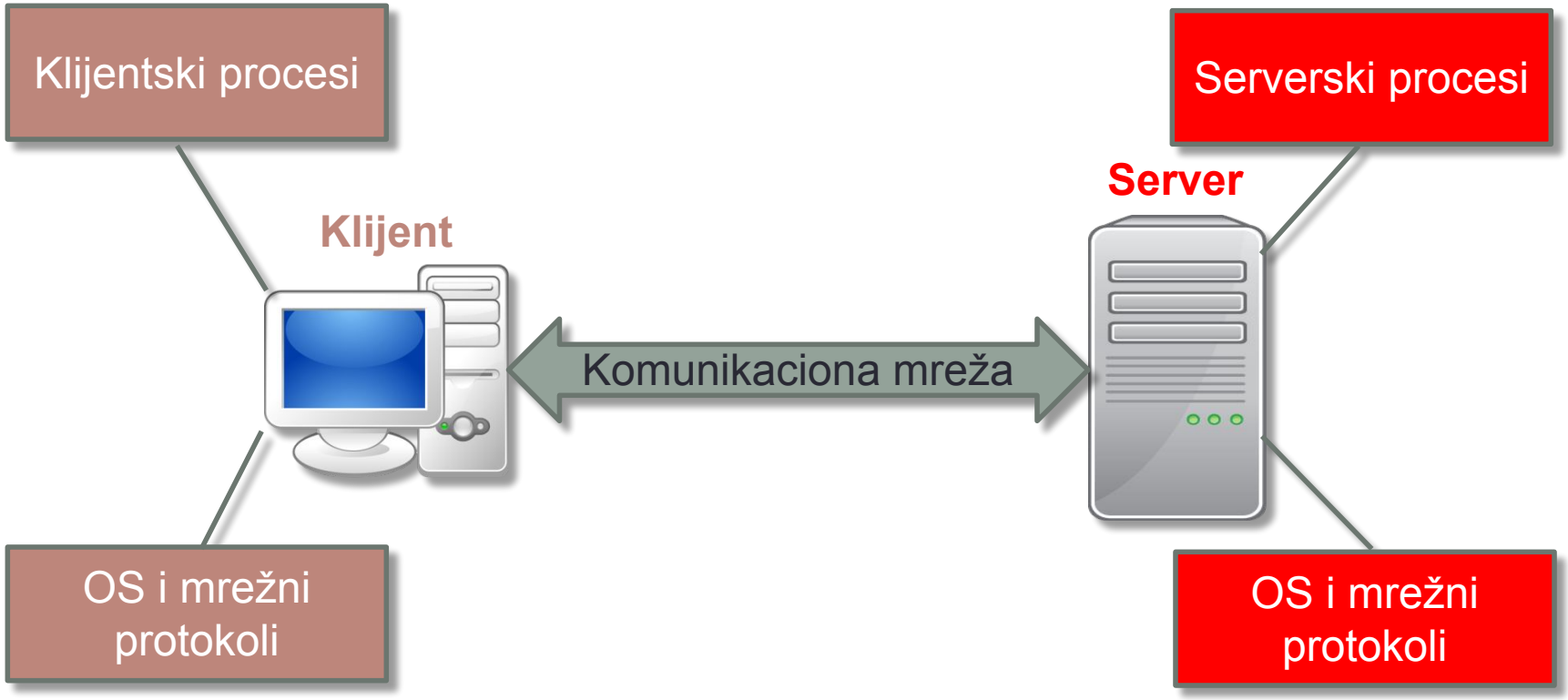
ARHITEKTURA WEB SISTEMA

Principi klijent-server arhitektura

Principi K/S sistema

- K/S model obrade podataka
  - vrsta distribuirane obrade podataka
  - kod koje se funkcije korisničkog programa raspodeljuju na najmanje dva procesa koji međusobno komuniciraju
- Tipovi procesa u K/S modelu
  - klijentski procesi
  - serverski procesi

Principi K/S sistema



Principi K/S sistema

- Komunikacija procesa u K/S sistemu
  - klijentski proces
    - **šalje poruku** serverskom procesu
      - zahteva „uslugu“ (izvršenje zadatka)
  - serverski proces
    - **izvršava zahtevani zadatak**
      - uspešno ili neuspešno
    - **šalje poruku** kao odgovor na zahtev

Principi K/S sistema

- Napomene o K/S modelu
  - klijentski i serverski procesi su specijalizovani za realizaciju određenih tipova zadataka
    - na određeni zahtev klijentskog procesa treba da odgovori serverski proces
      - specijalizovan za izvođenje zahtevane funkcije
    - granice funkcionalnosti klijentskih i serverskih procesa su jasno definisane
  - klijentski i serverski procesi su nezavisne programske jedinice

Principi K/S sistema

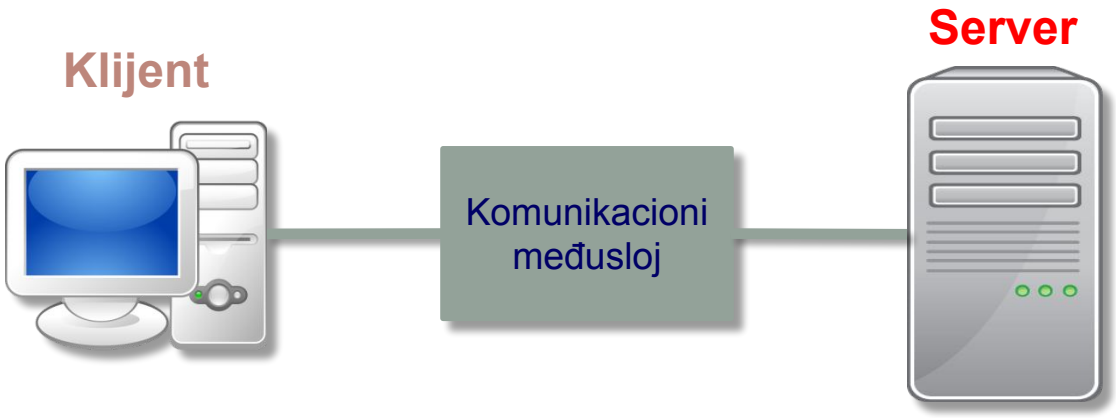
- Napomene o K/S modelu
  - K/S distribucija programa
    - deo programa na prednjem kraju (*eng. front end*)
      - realizuje se putem klijentskog procesa
    - deo programa na zadnjem kraju (*eng. back end*)
      - realizuje se putem serverskog procesa
  - ni jedan od delova ne predstavlja kompletan program
    - oni komplementiraju jedan drugog

K/S arhitektura

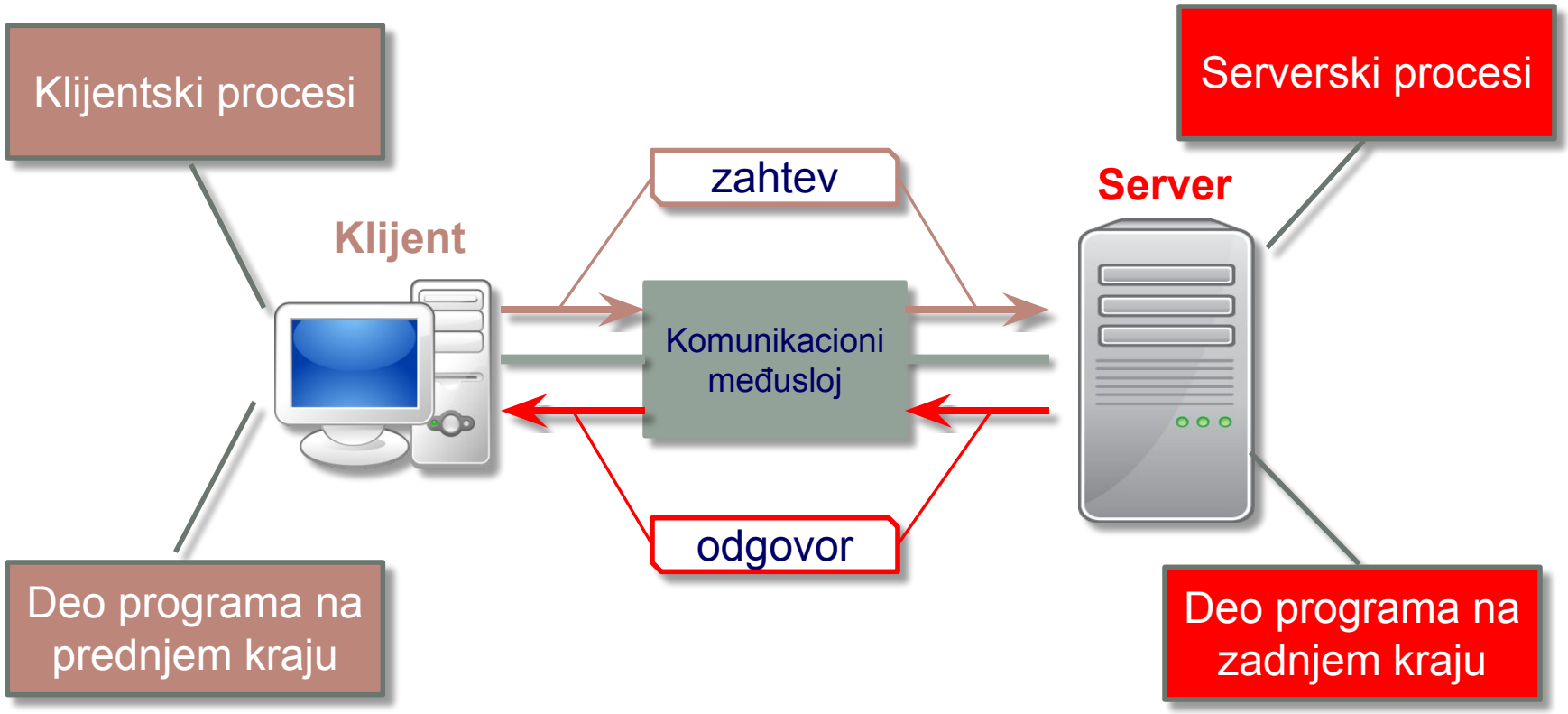
- Elementi K/S sistema
  - **jedan ili više servera**
  - **jedan ili više klijenata**
  - **klijentski procesi**
  - **serverski procesi**
  - **komunikacioni međusloj**

K/S arhitektura

- Komunikacioni međusloj
  - obuhvata sve hardversko-softverske elemente, neophodne da bi komunikacija između klijentskih i serverskih procesa bila moguća



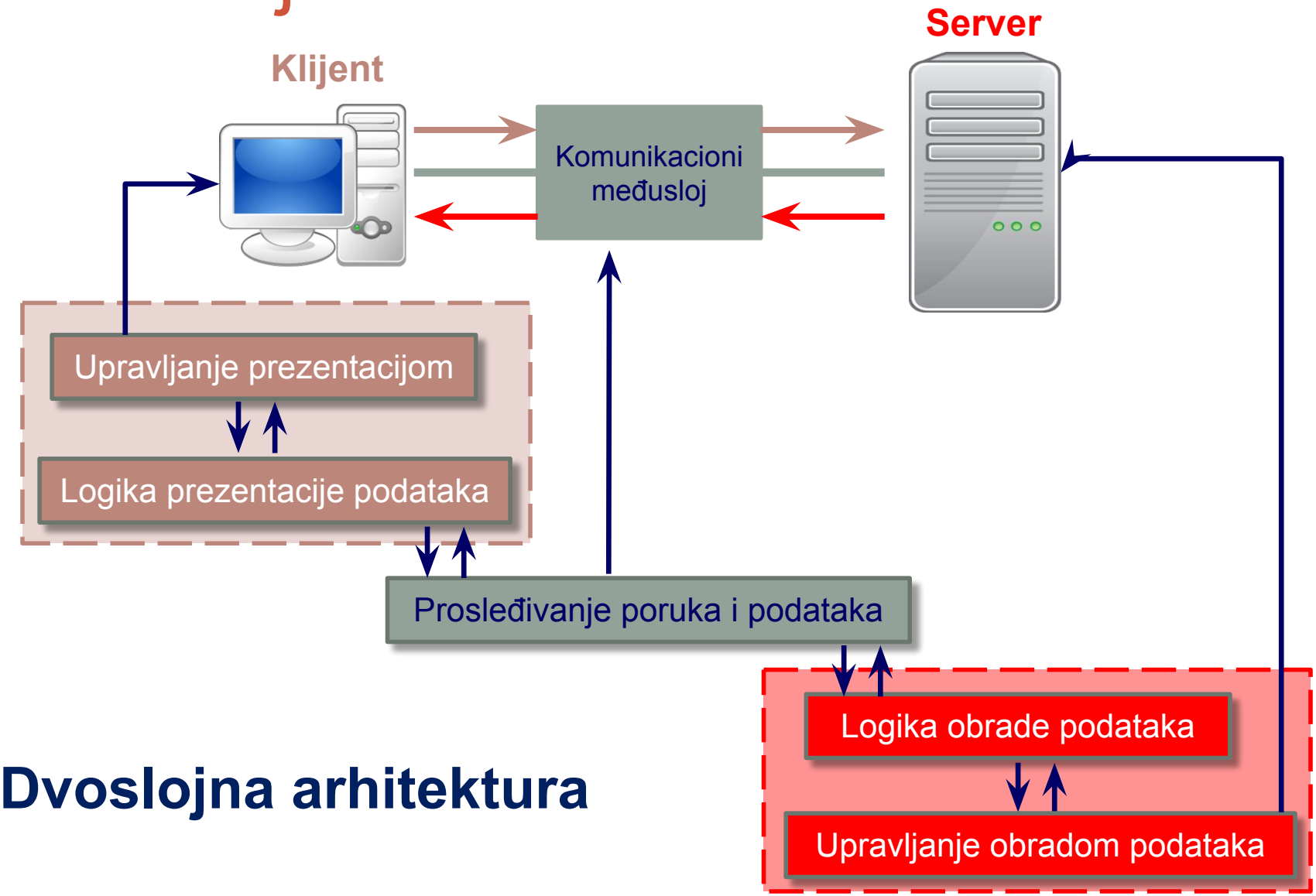
K/S arhitektura



K/S arhitektura

- Tipovi arhitektura
  - **prema broju klijenata i servera**
    - jedan klijent i jedan server
    - više klijenata i jedan server
    - više klijenata i više servera
  - **prema raspodeli tipova zadataka po funkcionalnim nivoima**
    - dvoslojna
    - troslojna
    - višeslojna

Višeslojna arhitektura K/S sistema

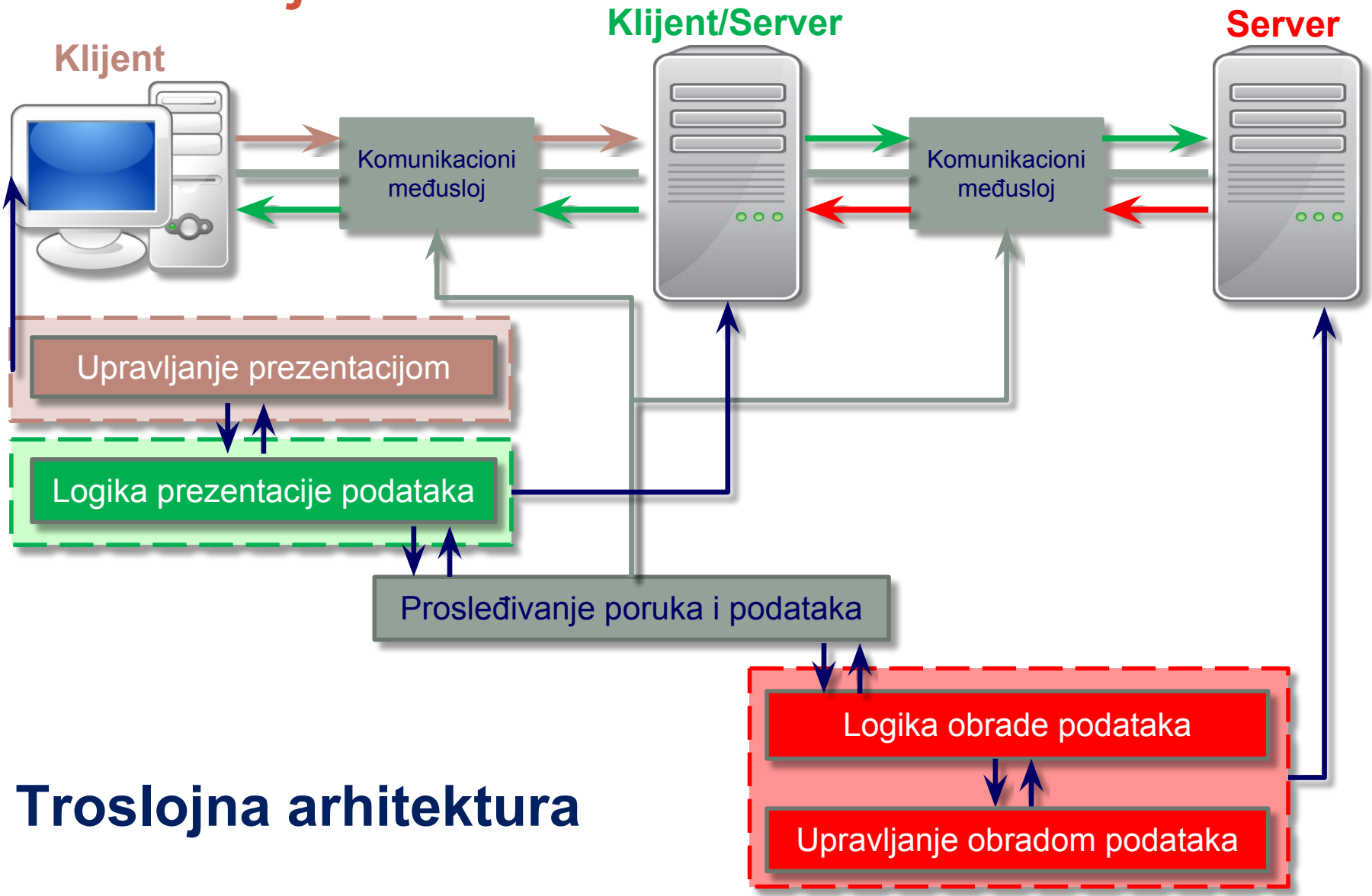


Dvoslojna arhitektura

Višeslojna arhitektura K/S sistema

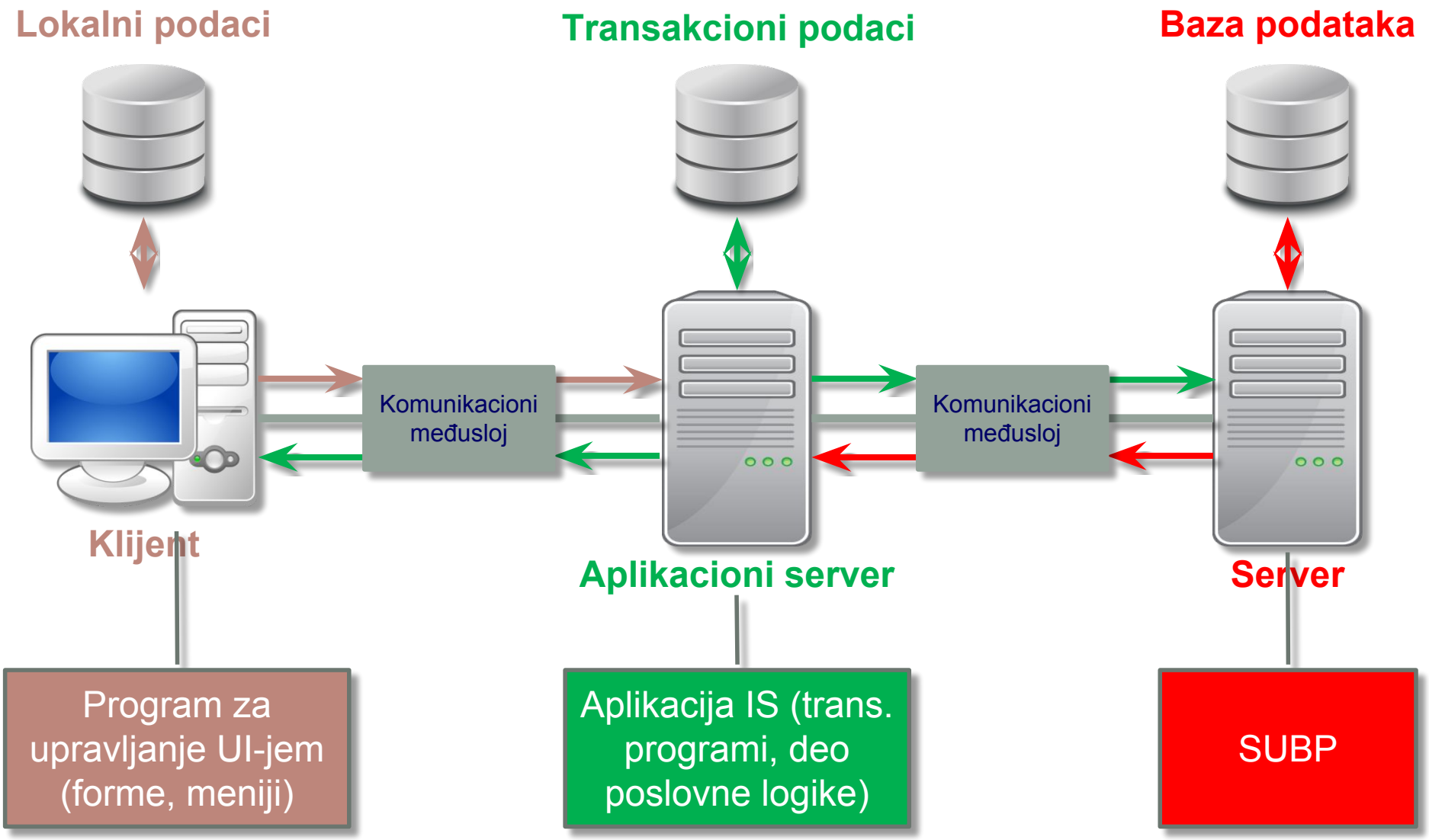
- Višeslojna arhitektura (n > 2)
  - osnovna pretpostavka
    - jedan funkcionalni nivo u raspodeli zadataka može imati, u isto vreme, **i ulogu klijenta i ulogu servera**
      - uloga klijenta prema nekom drugom serveru
      - uloga servera prema nekom drugom klijentu
- **Troslojna arhitektura**
  - klijent - upravljanje prezentacijom
  - srednji sloj - logika prezentacije
  - server - logika i upravljanje podacima

Višeslojna arhitektura K/S sistema

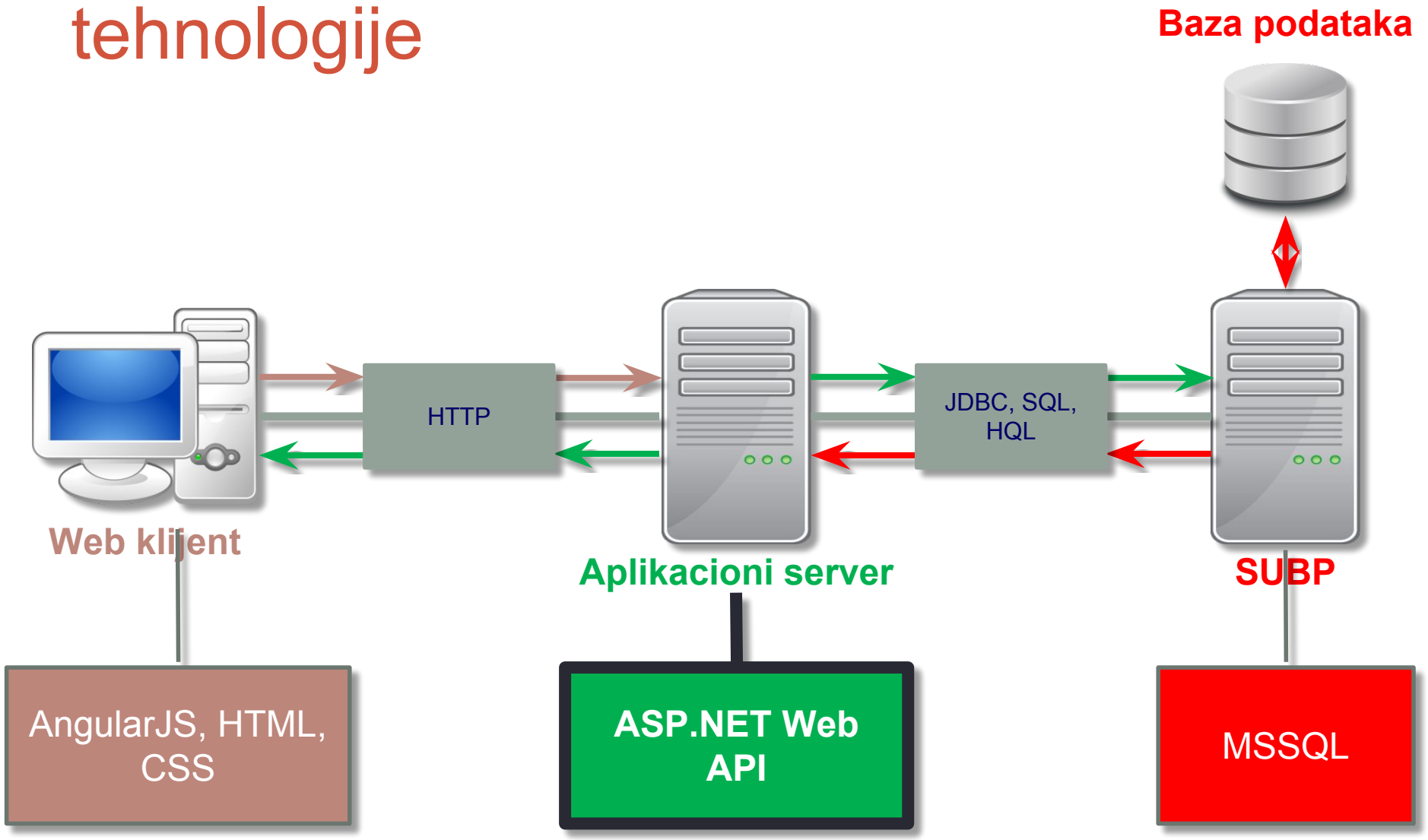


Troslojna arhitektura

Višeslojna arhitektura K/S sistema



Višeslojna arhitektura K/S sistema – tehnologije



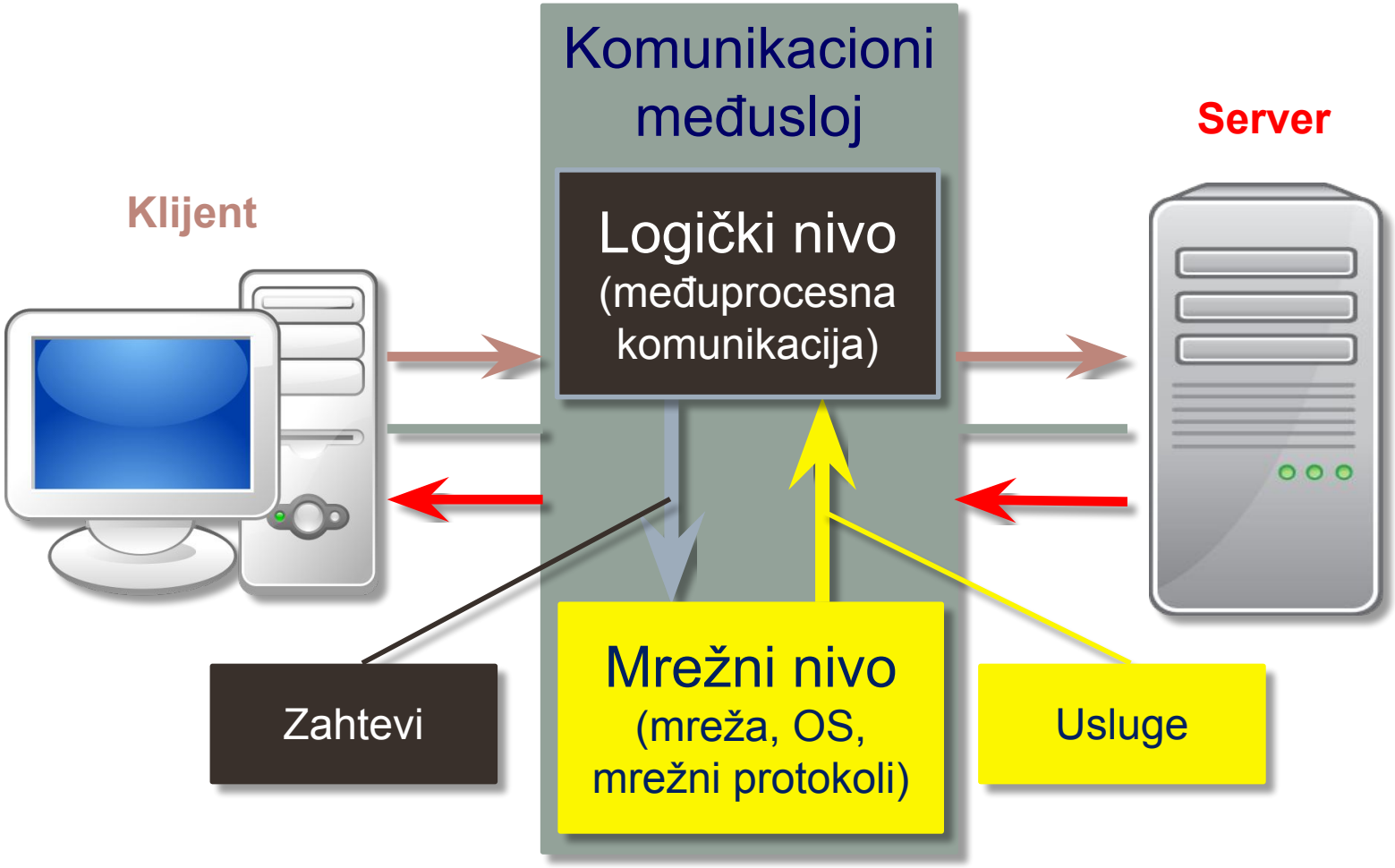
ARHITEKTURA WEB SISTEMA

Protokol HTTP

Komunikacioni međusloj

- Komunikacioni međusloj
  - hardversko-softverska komponenta
  - posrednik u komunikaciji klijenta i servera
  - s ciljem da obezbedi nezavisnost klijentskih i serverskih procesa pri komunikaciji
- Funkcionalni nivoi komunikacije
  - **mrežni nivo**
    - fizički prenos podataka između klijenata i servera u mreži
      - uređaji i veze računarske mreže
      - mrežni OS
  - **logički nivo**
    - **protokol za međuprocesnu komunikaciju**
    - komunikacija klijentskih i serverskih procesa

Komunikacioni međusloj



Komunikacioni međusloj

- Nezavisnost komunikacije
  - **mrežna nezavisnost**
    - nezavisnost komunikacije od mrežnog OS, protokola i hardvera
    - softverska i hardverska nezavisnost
  - **logička nezavisnost**
    - nezavisnost od upotrebljenih run-time okruženja i paradigmi programiranja
      - u oblasti BP, nezavisnost od upotrebljenog SUBP i upotrebljenog modela podataka
- Nezavisnost se oslanja na ISO/OSI model

Komunikacioni međusloj

- ISO Open System Interconnection (OSI)
  - referentni model mrežne arhitekture
    - ustanovljen 1984. godine od strane Međunarodne organizacije za standardizaciju ISO
  - **definiše klasifikaciju zadataka u računarskoj komunikaciji**
    - saglasno nivou apstraktnosti zadatka, vezanog za obavljanje komunikacije

Komunikacioni međusloj

- ISO/OSI arhitektura
  - **sedmoslojna arhitektura mreže**
    - sedam tipova (slojeva, nivoa) zadataka u komunikaciji
  - svaki nivo je **nezavisan** i namenjen za obavljanje određenih komunikacionih zadataka
    - način komunikacije na bilo kom nivou je
      - standardizovan, putem unpared propisanih protokola i
      - nezavisan od načina komunikacije na ostalim nivoima
  - nivoi su numerisani, saglasno stepenu apstrakcije
    - 1. nivo
      - najniži stepen apstrakcije
    - 7. nivo
      - najviši stepen apstrakcije

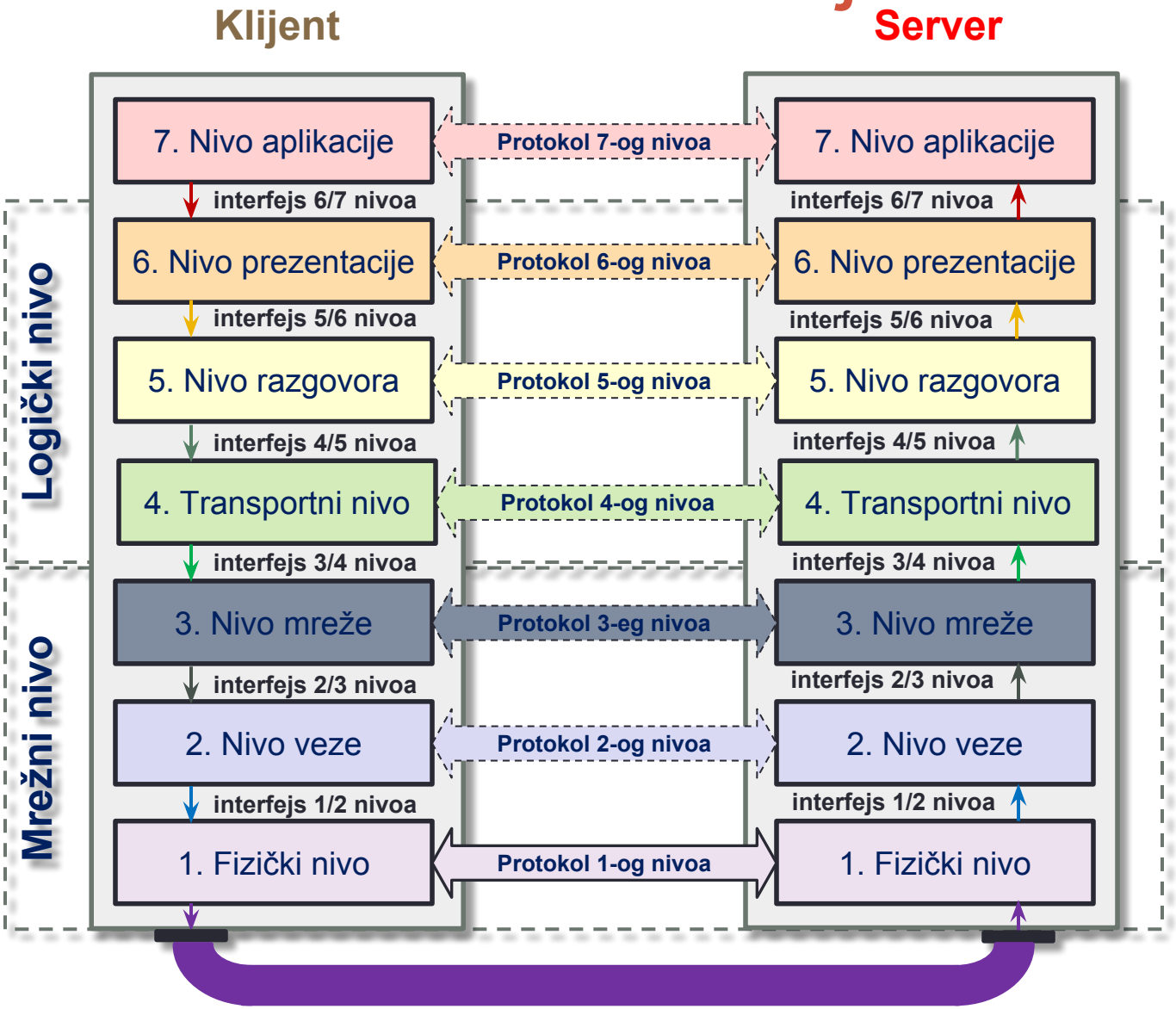
Komunikacioni međusloj

- ISO/OSI arhitektura
  - **komunikacija na 1. nivou**
    - fizička
      - realizuje se fizičkim prenosom podataka kroz mrežu
  - **komunikacija na višim nivoima**
    - virtuelna
      - realizuje se korišćenjem usluga susednog, nižeg nivoa
        - viši nivo prosleđuje zahtev nižem nivou
        - niži nivo opslužuje viši nivo - realizuje postavljeni zadatak i prosleđuje rezultate (poruke i podatke) višem nivou

Komunikacioni međusloj

- ISO/OSI arhitektura
  - susedni nivoi su međusobno spregnuti
    - unapred propisani protokoli (interfejsi) za komunikaciju susednih nivoa

Komunikacioni međusloj

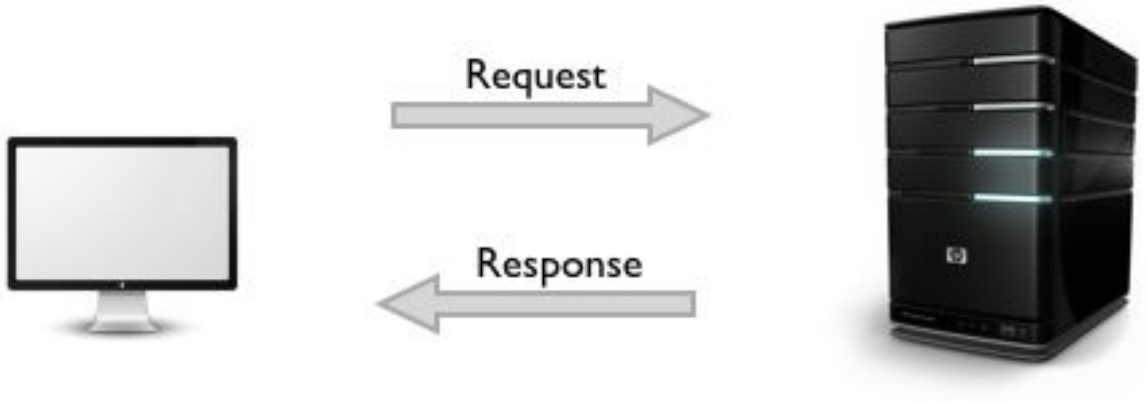


Protokol HTTP

- Hypertext Transfer Protocol (HTTP)
  - tekstualni protokol
    - za razmenu podataka u i između distribuiranih klijent-server sistema
    - kamen temeljac modernog Interneta
  - na aplikativnom nivou (7. nivo)
    - ISO/OSI i TCP/IP referentnog modela
    - podrazumevani port 80
      - moguće izmeniti
  - *stateless* protokol
    - kako bi omogućio razmenu podataka između različito konfigurisanih učesnika u komunikaciji, ne pamti stanje i ne zavisi od platformi na kojima se učesnici u komunikaciji nalaze

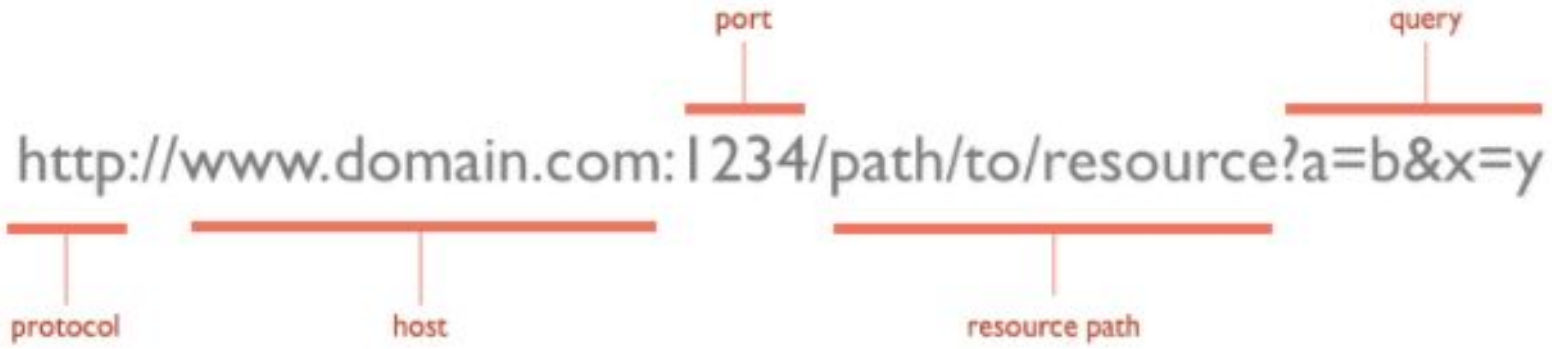
Protokol HTTP

- komunikacija se obavlja u oba smera
  - klijent šalje zahteve
  - server šalje odgovor
- Svaka poruka ima zaglavlje i telo
  - koji mogu biti i prazni



Protokol HTTP - URL

- Uniform Resource Locator (URL)
  - jedinstveni identifikator nekog resursa na Internetu



Protokol HTTP - Metode

- Metode – najčešće korišćene
  - **GET** – dobavljanje postojećeg resursa
  - **POST** – lreiranje novog resursa
  - **PUT** – ažuriranje postojećeg resursa
  - **DELETE** – brisanje postojećeg resursa
- Metode – manje korišćene
  - HEAD – preuzimanje zaglavlja sa servera najčešće radi provera da li je došlo do izmene resursa
  - TRACE – za dijagnostiku (kroz koje čvorove zahtev prolazi)
  - OPTIONS – radi identifikacije mogućnosti servera

Protokol HTTP - Metode

- Svaki zahtev obuhvata slanje poruke određenom HTTP metodom
  - zaglavlje i telo zahteva
  - spisak svih metoda
    - https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
- Idempotentne metode
  - metode koje nemaju propratnih efekata (*side-effects*)
  - GET, HEAD, PUT, DELETE
  - OPTIONS i TRACE

HTTP—REQUEST

- Kako izgleda HTTP request
  - METOD URI HTTP/VERZIJA
  - Zaglavlje koje se sastoji od makar jedne linije (Host), a potencijalno više
    - IME: VREDNOST
  - Prazna linija.
  - Telo poruke, ako postoji.

HTTP—RESPONSE

- Kako izgleda HTTP response
  - HTTP/VERZIJA KOD PORUKA
  - Zaglavlje odgovora koje je formatirano isto kao i kod zaglavlja zahteva
  - Prazna linija.
  - Telo poruke, ako postoji.

Protokol HTTP – Tip sadržaja

- Sadržaj tela zahteva ili odgovora
  - može biti formatiran prema različitim formatima za serijalizaciju podataka
  - JSON, XML, običan tekst
  - **Content-type polje** u zaglavlju
- Primer:
  - Content-Type: text/html; charset=utf-8
  - Content-Type: multipart/form-data; boundary=something

Protokol HTTP – Status kodovi

- Status kodovi u odgovoru
  - definišu uspešnost izvršene operacije
  - https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- 2xx – uspešno obavljena operacija
- 3xx – redirektovan zahtev
  - klijent mora izvštiti dodatne akcije
- 4xx – greška na klijentskoj strani
- 5xx – greška na serverskoj strani

HTTP - primer komunikacije

- REQUEST:

GET /index.html HTTP/1.1
Host: www.example.com

HTTP - primer komunikacije

- RESPONSE:

```
HTTP/1.1 200 OK
Date: Mon, 23 May 2005 22:38:34 GMT
Content-Type: text/html; charset=UTF-8
Content-Encoding: UTF-8
Content-Length: 138
Last-Modified: Wed, 08 Jan 2003 23:11:55 GMT
Server: Apache/1.3.3.7 (Unix) (Red-Hat/Linux)
ETag: "3f80f-1b6-3e1cb03b"
Accept-Ranges: bytes
Connection: close

<html>
        <head>
                <title>An Example Page</title>
        </head>
        <body>
                Hello World, this is a very simple HTML document.
        </body>
</html>
```

ARHITEKTURA WEB SISTEMA

Arthitektura REST

Arhitektura REST

- REpresentational State Transfer (REST)
  - skup principa koji definišu kako bi trebalo koristiti postojeće web element (HTTP, URIs, itd.) prilikom izgradnje web aplikacija
  - pet osnovnih principa:
    1. svakom resursu je neophodno dodeliti jedinstveni identifikator (ID)
    2. resurse je potrebno uvezati
    3. moguće je koristiti isključivo standardne metode
    4. resursi mogu imati više od jedne reprezentacije
    5. komunikacija se obavlja bez čuvanja stanja (*stateless* komunikacija)

REST – ID

- Svakom resursu („stvari“) potrebno je dodeliti poseban ID
  - Unified Resource Identificator (URI)
  - globalno jedinstveni identifikator
    - pojedinačnih resursa,
    - kolekcija resursa,
    - virtuelnih, fizičkih i sračunatih resursa
    - URL i URN
- Primeri jedinstvene identifikacije resursa:
  - http://example.com/customers/1234
  - http://example.com/orders/2007/10/776654
  - http://example.com/products/4554
  - http://example.com/processes/salary-increase-234
  - urn:oasis:names:specification:docbook:dtd:xml:4.1.2

REST – uvezivanje

- Hypermedia as the engine of application state
  - HATEOAS
  - ukoliko resursi sadrže druge resurse, drugi resursi su identifikovani (uvezani) putem svojih URI-ja
    - moguće im posredno pristupiti prateći URI
  - ne postoji definicija interfejsa niti javna dokumentacija
    - koju klijent mora da prati
    - potrebno iskljucivo poznavanje hypermedije
  - promena stanja aplikacije praćenjem URI-ja (linkova)
- Primer XML

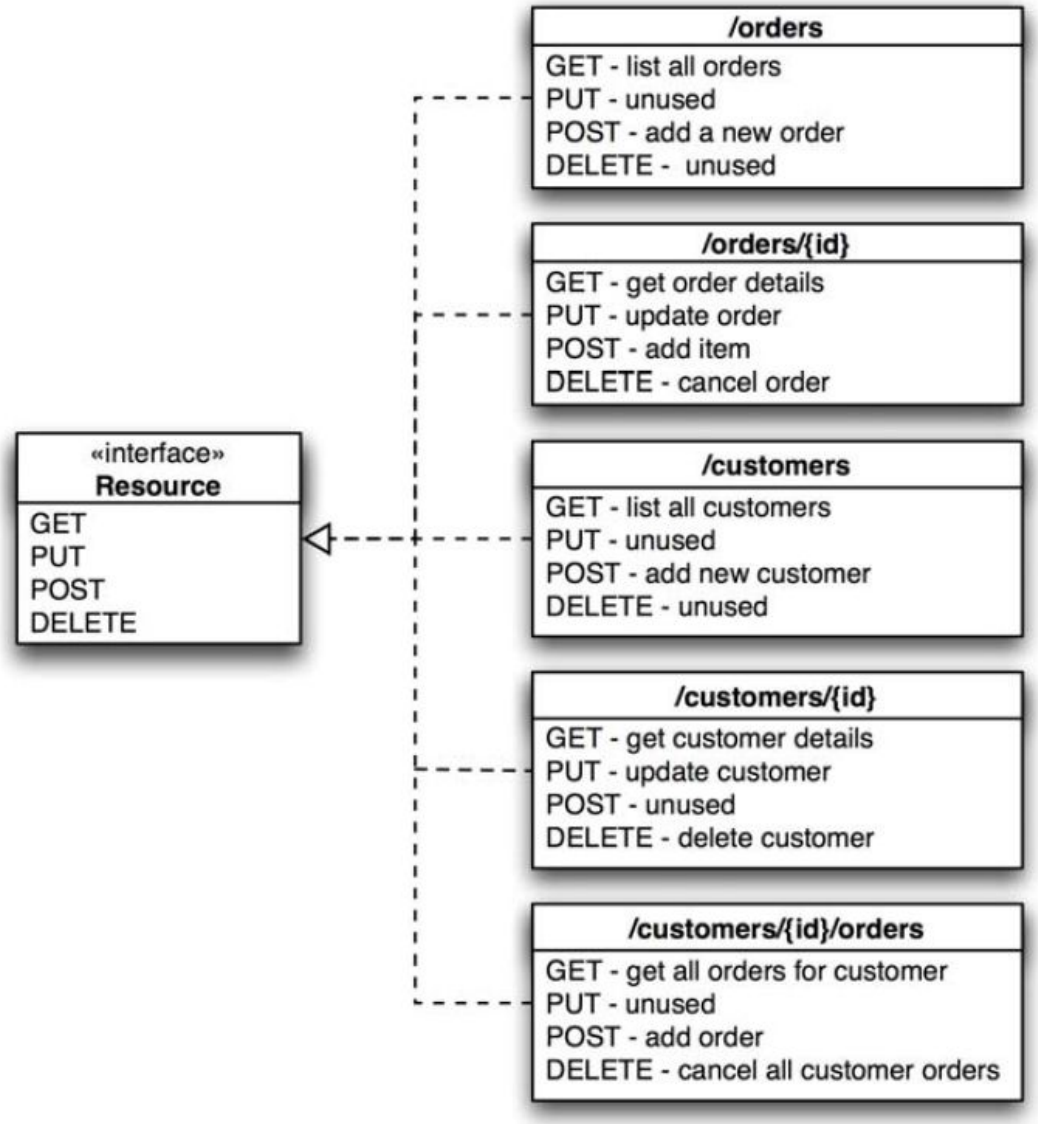
```
<order self='http://example.com/customers/1234' >
    <amount>23</amount>
    <product ref='http://example.com/products/4554' />
    <customer ref='http://example.com/customers/1234' />
</order>
```

REST – standardne metode

- Koristiti isključivo standardne HTTP metode za rukovanje resursima
  - GET, POST, PUT, DELETE ...
  - bez uvođenja novih metoda specifičnih za trenutni projekat
  - generički klijenti mogaju samo da implementiraju standardne HTTP metode i da poznaju URI resursa
- Primeri:
  - dobavljanje klijenta sa ID-jem 1234
    - GET na http://example.com/customers/1234
  - brisanje klijenta sa ID-jem 1234
    - DELETE na http://example.com/customers/1234
  - dodavanj novog klijenta sa ID-jem 4321
    - POST na http://example.com/customers/1234
    - u telu zahteva specificirati vrednosti za sva polja klijenta

REST – standardne metode

- Standardni interfejs
  - npr. četiri metode
  - implementiraju ga svi servisi



REST – višestruka reprezentacija

- Isti resurs može biti predstavljen na više načina
  - JSON, XML, Tekst
  - dozvoljava iskorištenje resursa od strane standardnih web pretraživača
    - ne samo od specijalizovanih klijentskih aplikacija
  - pretvarati web UI u web API
  - content-type parametar u zaglavlju

REST – komunikacija bez pamćenja stanja

- *stateless communication*
  - stanje sesije se ne pamti između dva poziva
  - pamte se samo stanja resursa posle svakog poziva
  - dozvoljava skaliranje aplikacija
    - pamćenje stanja sesije za veliki broj klijenata bi ozbiljno narušilo performanse sistema
  - omogućava nezavisnost klijenta od konkretnog servera
    - dozvljava slanje uzastopnih zahteva različitim serverima istog sistema
  - omogućava nezavisnost klijenata od internih promena na serveru
    - dozvoljena promena algoritama i načina funkcionisanja servera
      - bez narušavanja javnog API-ja

ARHITEKTURA WEB SISTEMA

ASP.NET Web API i funkcionalni nivoi ASP.NET Web API aplikacija

ASP.NET Web API

- ASP.NET Web API je *open-source framework* za razvoj .NET web servisa
- Glavni cilj: **pojednostaviti razvoj .NET web servisa**
- Izdvojio se iz ASP.NET MVC *framework*-a

ASP.NET Web API

- Glavni principi prilikom dizajna ASP.NET bili su:
  - distribuiran po dizajnu
    - HTTP je prvoklasni građanin
    - omogućava pravljenje distribuiranih sistema:
      - stateless
      - caching
      - compression
  - Russian doll model
    - ASP.NET Web API obrađuje zahteve tako što definiše cevovode (*pipeline*) za obradu zahteva (request) i odgovora (response)
    - prilikom obrade poruka (request ili response) se daje prvom obrađivaču (handlar) na obradu
    - handler prilikom završetka obrade prosleđuje poruku sledećem handleru u pipeline-u

ASP.NET Web API

- Glavni principi prilikom dizajna ASP.NET bili su:
  - asinhoronost
    - zasnovano na korišćenju Task-Based Asynchronous Pattern (TAP)
    - pojednostavlja asinhrono programiranje korišćenjem **async** i **await** ključnih reči
    - TAP je zasnovan na Task i Task<Result> tipovima iz *System.Threading.Tasks* namespace-a

```
async Task<int> GetContentLengthAsync(string uri)
{
        int contentLength;
        using (var client = new HttpClient())
        {
        var content = await client.GetStringAsync(uri);
        contentLength = content.Length;
        }

        return contentLength;
}
```

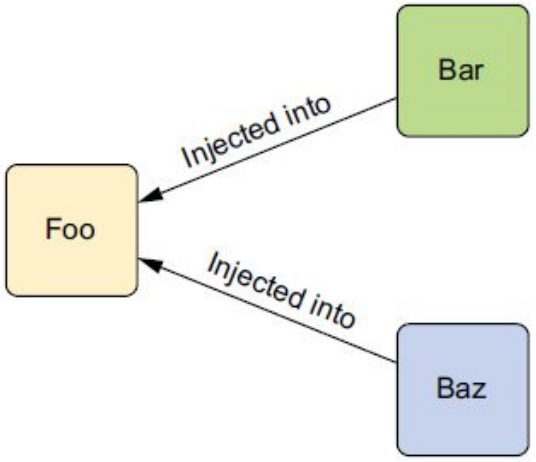
C# – POCO

- Obična klasa
  - bez nasleđivanja drugih klasa specifičnih za ASP.NET
  - obične klase sa poljima, propertijima i konstruktorima
- C# POCO

```
public class Person
{
        public string Name { get; set; }
        public string SayHello()
        {
                return "Hello World";
        }
}
```

ASP.NET – Dependency Injection

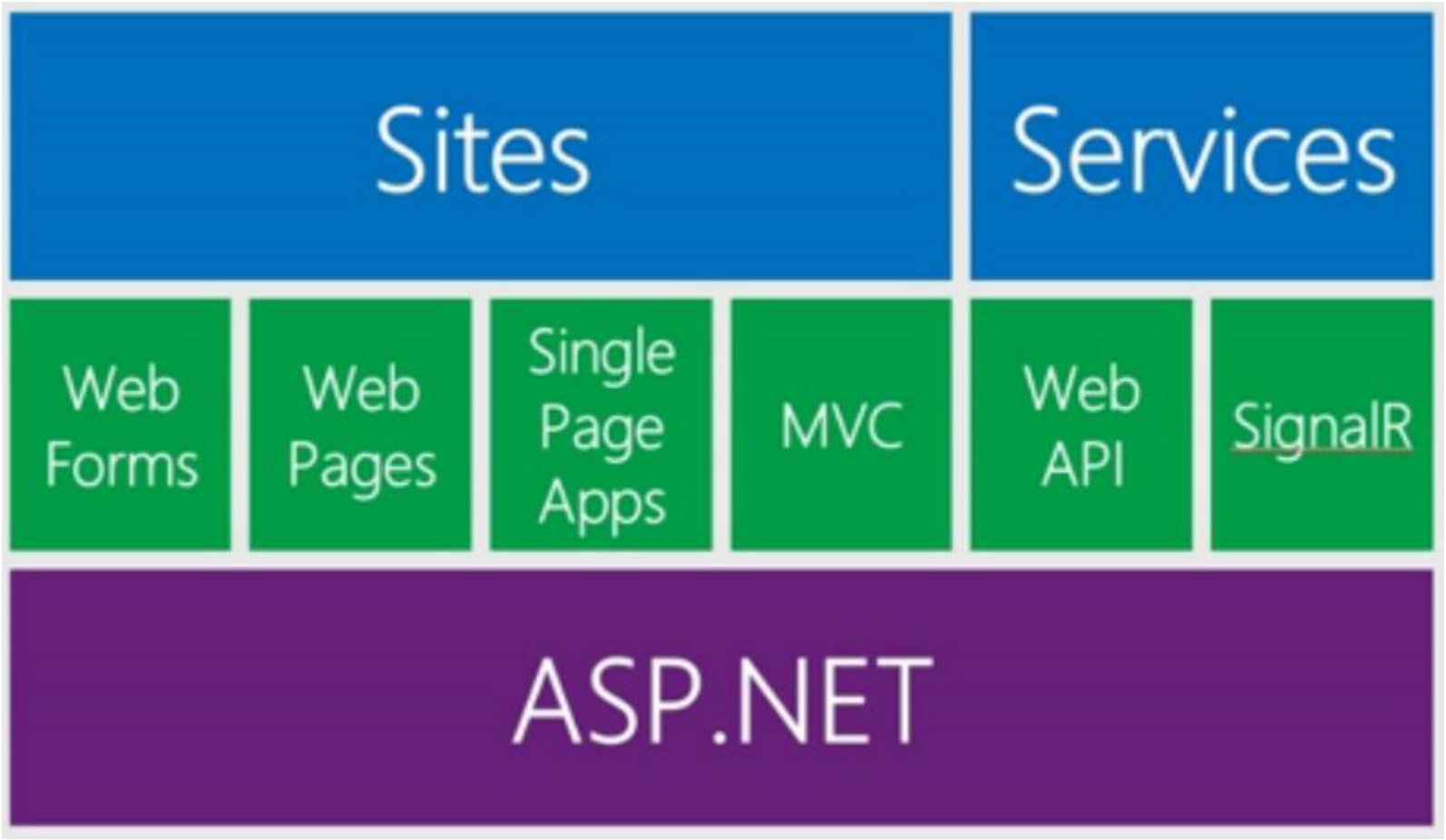
- Aplikacije često imaju više od jedne klase koje moraju međusobno da sarađuju
  - radi ostvativanja neke kompleksne funkcionalnosti
  - tradicionalno, svaka klasa je bila zadužena za instanciranje svih povezanih klasa koje joj trebaju
    - dovodi do jako spregnutih aplikacija
    - eksplozija referenci, poziva konstruktora, neophodnih konfigurisanja



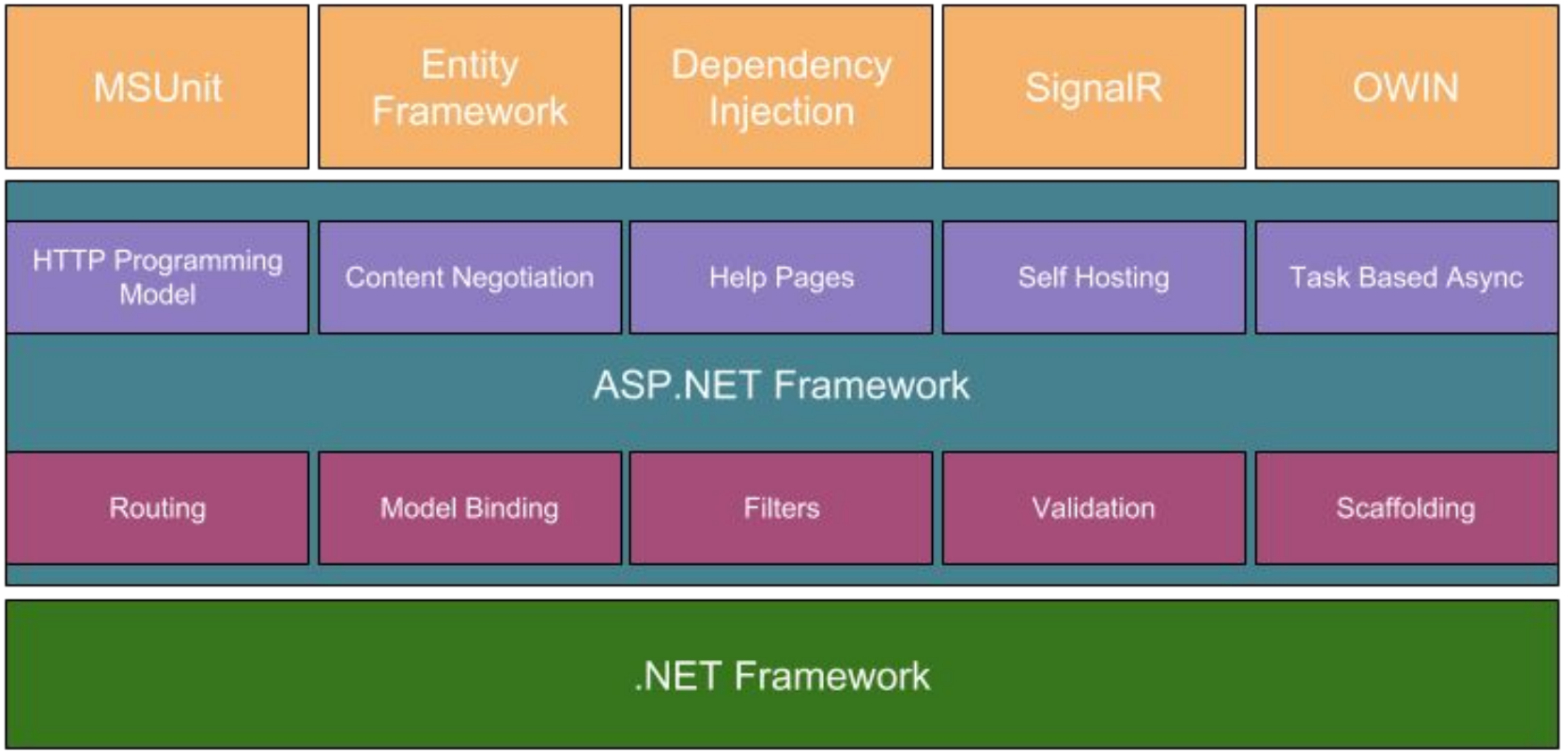
ASP.NET Web API – Dependency Injection

- Nema ugrađen Dependency Injection u sam framework
- Podržava bilo koji framework koji implementira *simple service location interface*
- Poznati Dependency Injection framework-ci:
  - Unity
  - Ninject
  - Autofac
  - Spring.NET
  - Castle Windsor

ASP.NET


Sites
Services
Web Forms
Web Pages
Single Page Apps
MVC
Web API
SignalR
ASP.NET

ASP.NET Web API – gradivni elementi



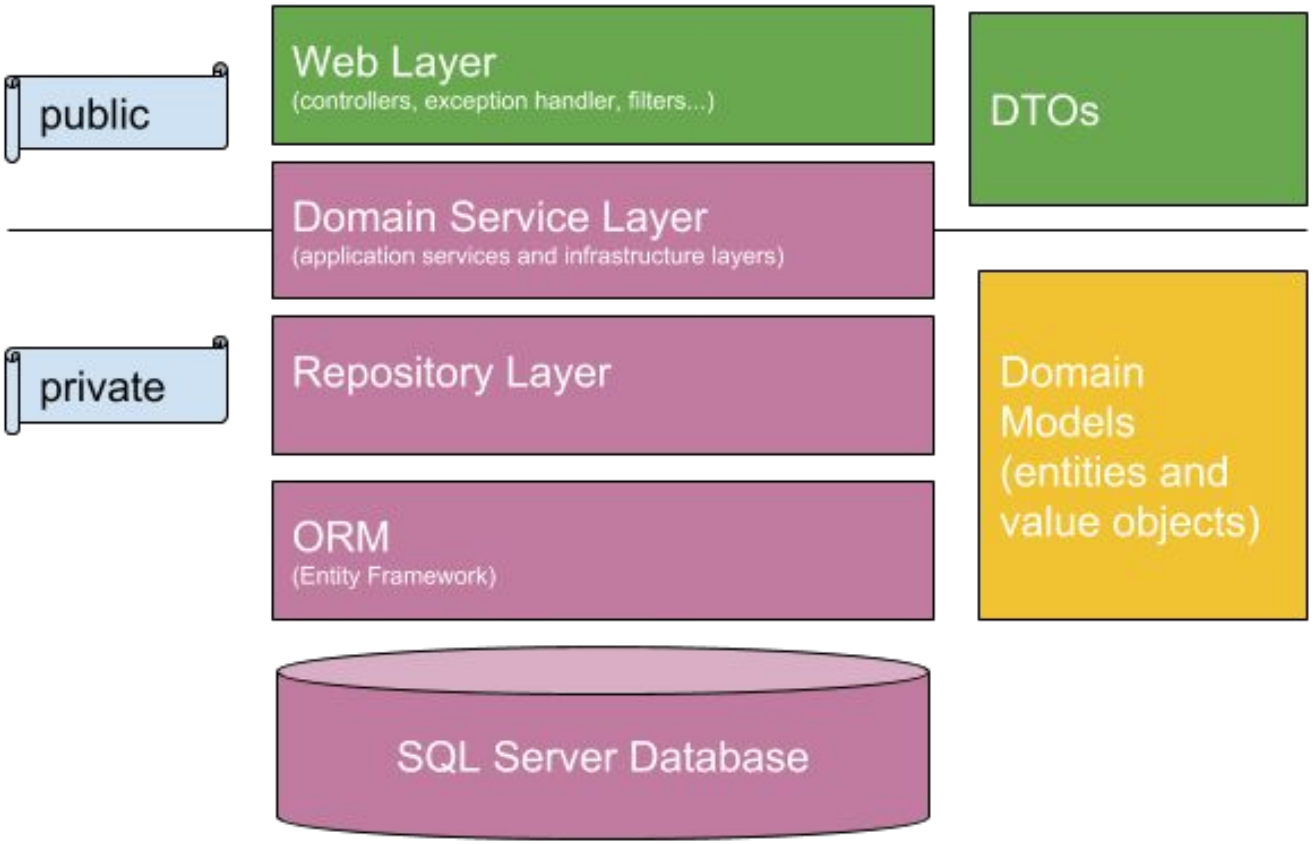
ASP.NET Web API – arhitektura paketa

- *ASP.NET Framework*
  - omogućuje DI
  - omogućava pravljenje web aplikacije
  - osnova za sve ostale module
- *Dependecy Injection*
  - omogućava pravljenje slabo spregnutih komponenti
    - lakše za testirati
    - lakše za pronalaženje bug-ova
- *Entity Framework*
  - rad sa izvorima podataka
    - bazama podataka
  - podrška za rad sa objektno-relacionim preslikavanjem

ASP.NET Web API – arhitektura paketa

- *SignalR*
  - omogućava dodavanje komunikacije u realnom vremenu
  - slanje podataka sa servera, gde server inicira komunikaciju, ka klijentu u realnom vremenu
- *OWIN*
  - standardni interfejs između web aplikacije i web servera
  - omogućava da web aplikacije hostujemo na drugim serverima, a ne samo IIS-u
  - podržani serveri:
    - IIS
    - Self-hosted
- *MSUnit*
  - dozvoljava pisanje testova

ASP.NET Web API – funkcionalni slojevi



ASP.NET Web API

Prvi projekat

ASP.NET Web API

- ASP.NET Web API
  - *framework* koji omogućava veoma brzi razvoj web aplikacija
  - definiše pretpostavke o strukturi koda, konfiguraciji, korišćenim bibliotekama, nazivima klasa i promenljivih
  - bez generisanja koda
  - uz minimalno korišćenja XML konfiguracije

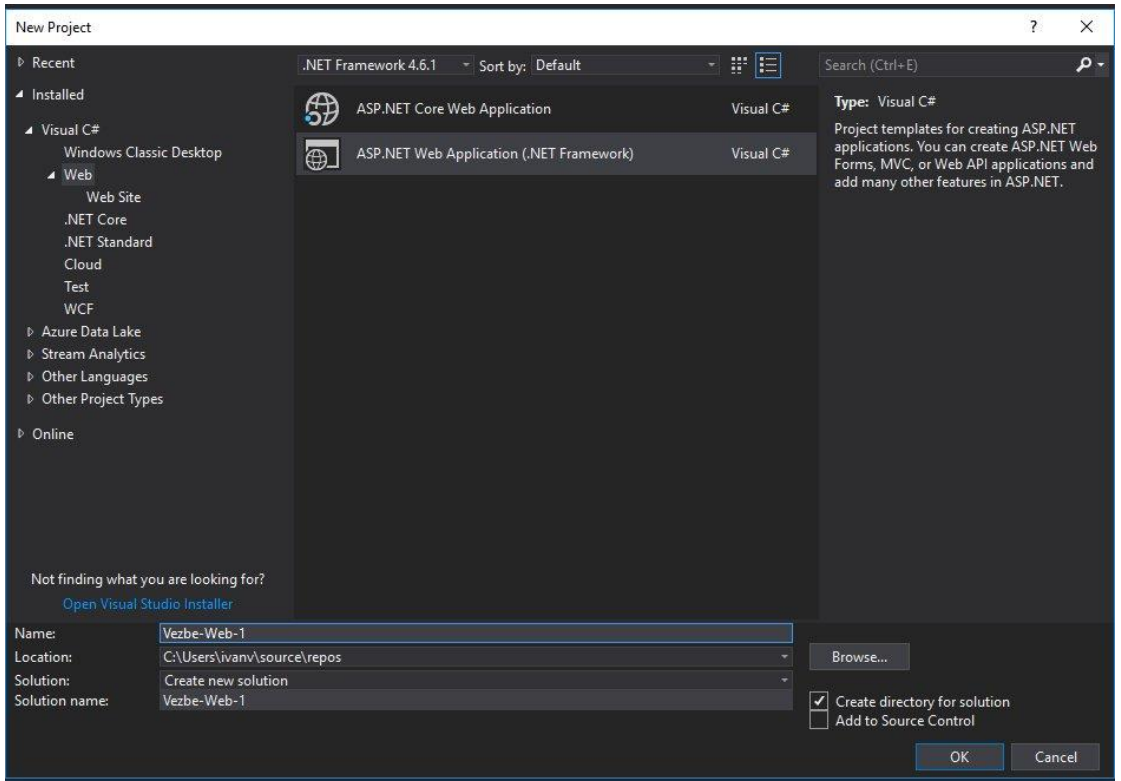
Primer 1.1 - Github link

- Kreirati ASP.NET Web API projekat - T1-P1-1
  - kako bismo mogli praviti web aplikacije

VS projekat

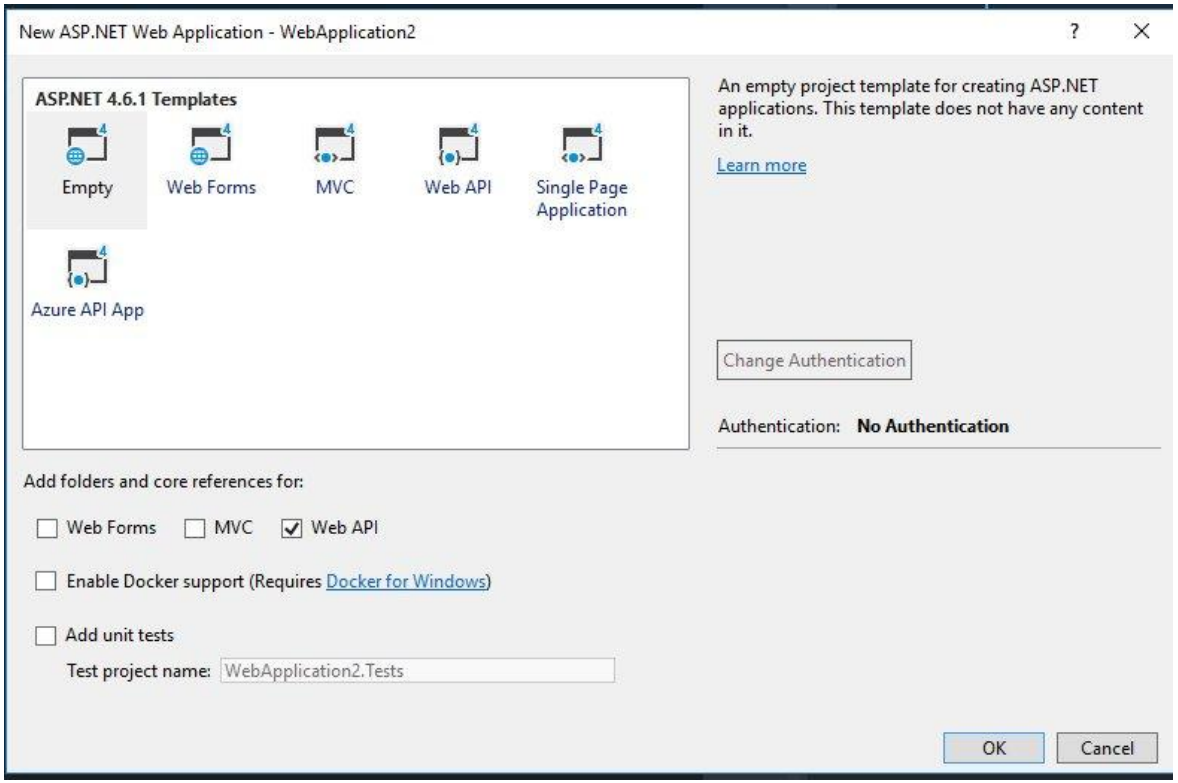
- Kreirati novi ASP.NET Web Application projekat
  - File -> New -> Project
    ASP.NET Web Application(.NET Framework) project
  - na prvom ekranu popuniti kao što je dato na slici
    - nakon popunjavanja

      pritisnuti dugme **OK**



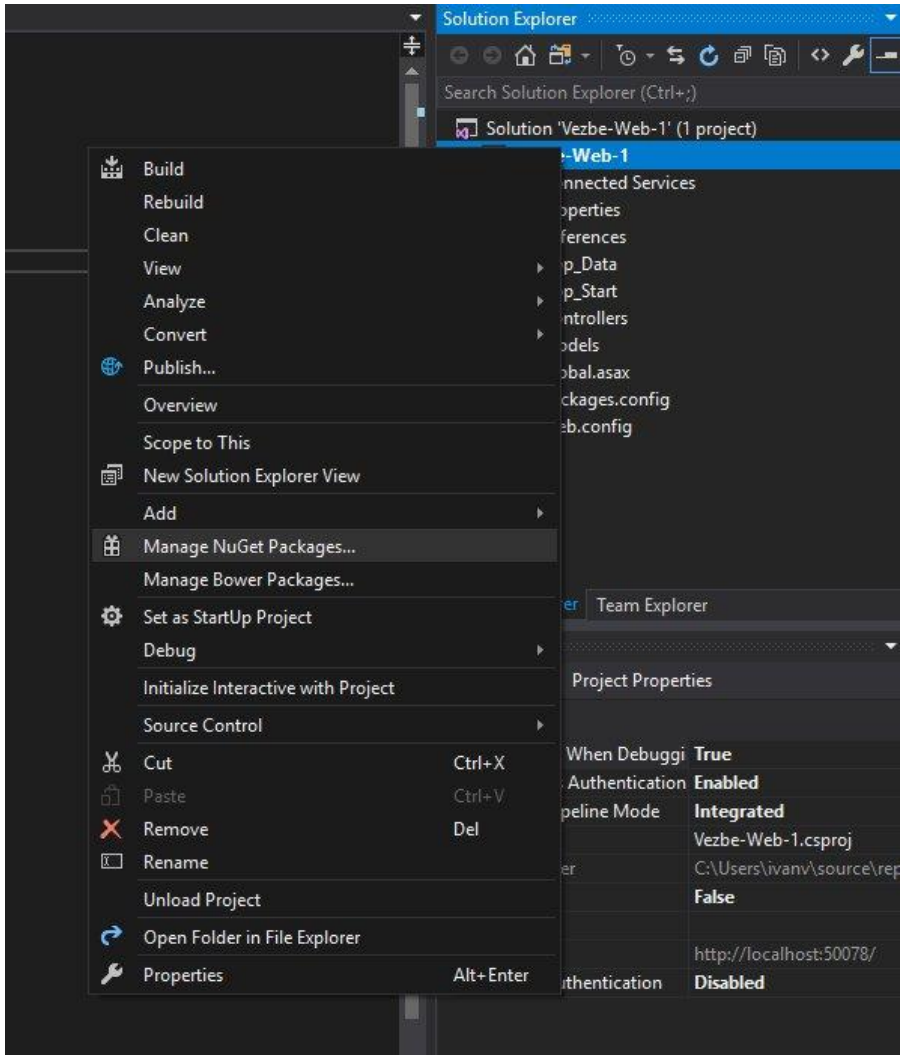
VS projekat

- Kreirati novi ASP.NET Web Application projekat
  - na drugom ekranu:
    - odabrati **Empty (Emplty project template)**
    - selektovati checkbox **Web API**
  - pritisnuti **OK**



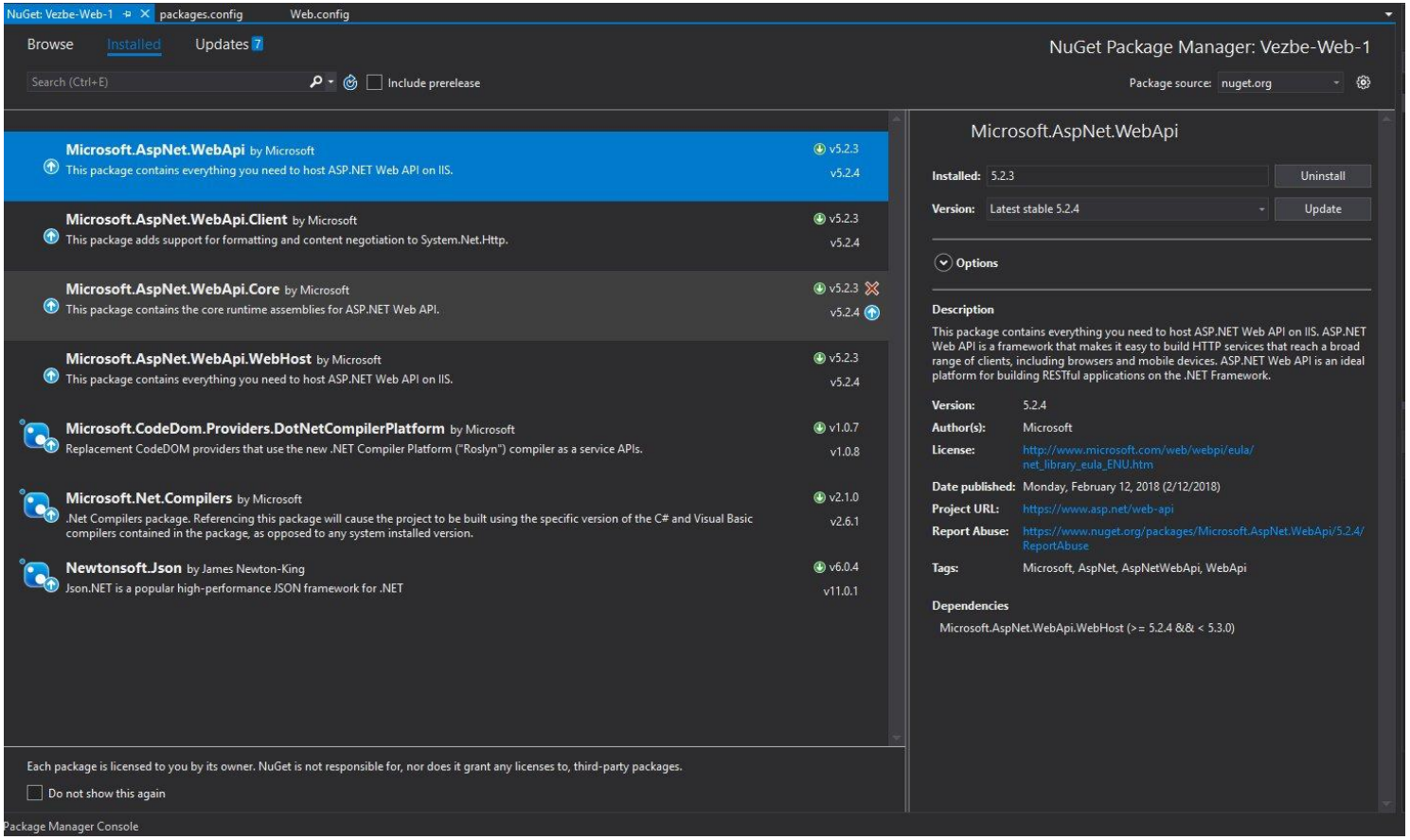
VS projekat - NuGet

- da bi videli i upravljali paketima koristimo NuGet
- trenutno instalirane pakete možemo naći desnim klikom na projekat i zatim odabirom *Manage NuGet Packages*



VS projekat - NuGet

- otvara se prozor za upravljanje paketima



VS projekat - NuGet

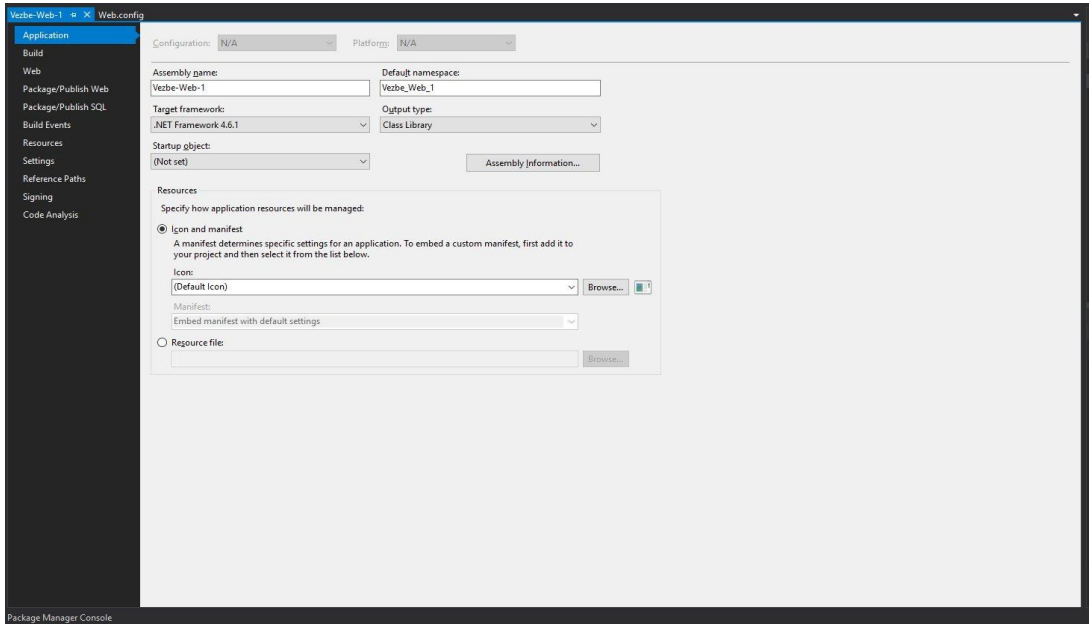
- otvara se prozor za upravljanje paketima
- odavde možemo da:
  - instaliramo novi paket
  - deinstaliramo postojeći paket
  - promenimo verziju instaliranog paketa
  - nađemo kratak opis paketa

- listu instaliranih paketa možemo naći i u **packages.config**
  - package element - predstavlja instalirani paket
  - id - jednistvena identifikacija paketa
  - version - verzija paketa
  - targetFramework - verzija .NET Frameworka za koju nam je potreban paket

```
<package id="Microsoft.AspNet.WebApi" version="5.2.3" targetFramework="net461" />
```

VS projekat

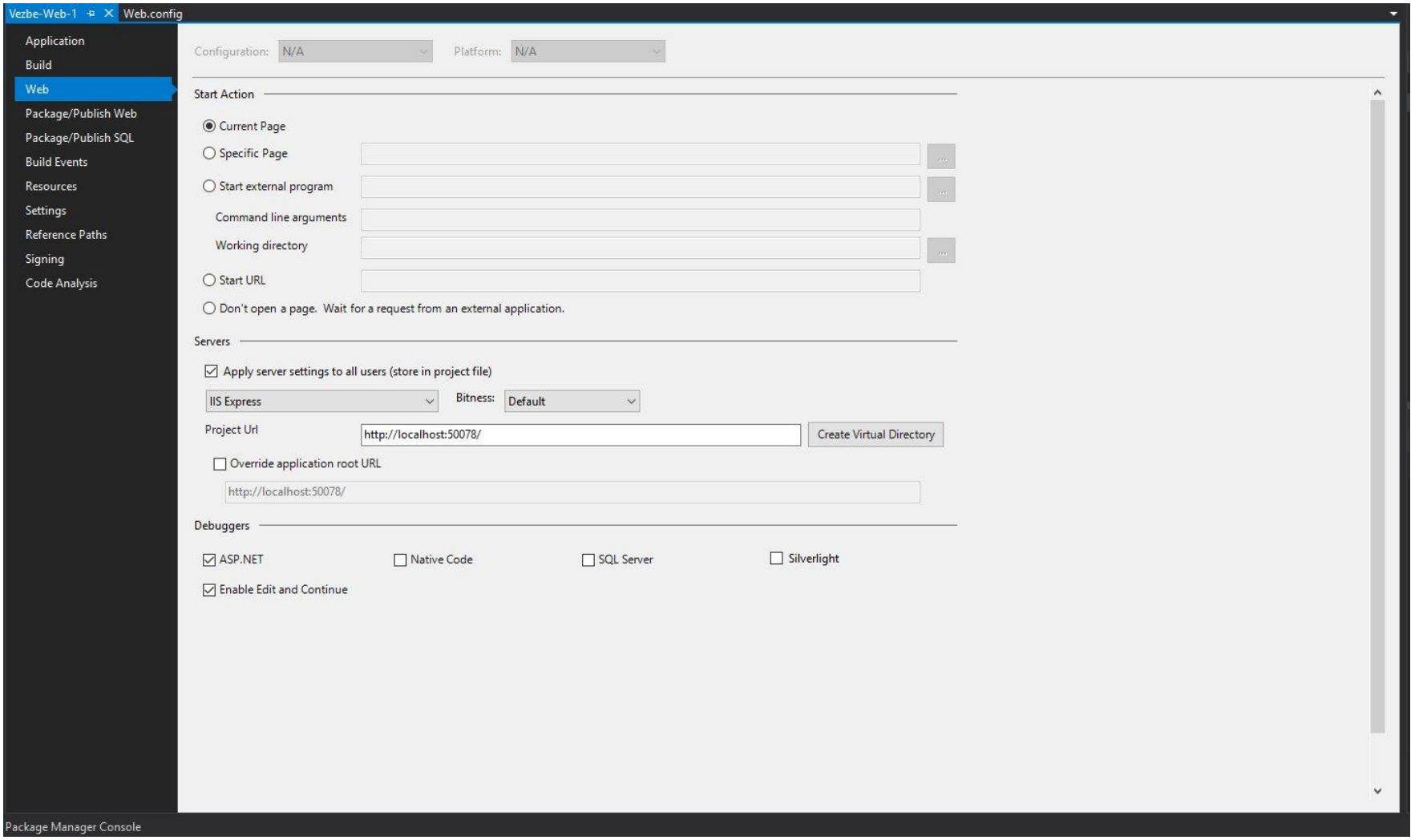
- klikom desnim tasterom miša na projekat, a zatim odabirom *Properties* dolazimo do podesavanja projekta
- možemo izmeniti:
  - ime DLL-a koji predstavlja aplikaciju
  - ime osnovnog *namespace*-a
  - .NET Framework koji koristimo
- ne smemo menjati output type



VS projekat

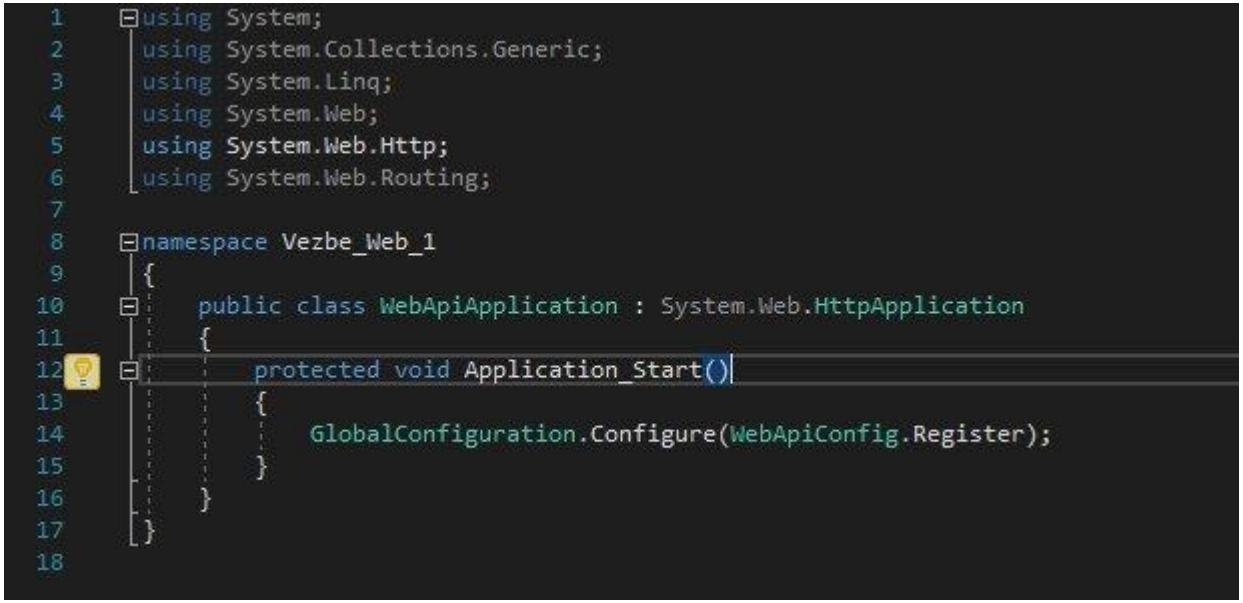
- Web tab nas dovodi do podesavanja vezana za web aplikaciju
  - server koji koristimo - IISExpress - web server namenjen za razvoj aplikacija
  - stranicu koja se otvara kada pokrenemo aplikaciju
  - promenimo port - izmena broja u project url polju
  - arhitekture za koju pravimo aplikaciju x86/x64
  - koji debuger koristimo

VS projekat


Vezbe-Web-1
Web.config
Application
Build
Web
Package/Publish Web
Package/Publish SQL
Build Events
Resources
Settings
Reference Paths
Signing
Code Analysis
Configuration: N/A
Platform: N/A
Start Action
Current Page
Specific Page
Start external program
Command line arguments
Working directory
Start URL
Don't open a page. Wait for a request from an external application.
Servers
Apply server settings to all users (store in project file)
IIS Express
Bitness: Default
Project Url
http://localhost:50078/
Create Virtual Directory
Override application root URL
http://localhost:50078/
Debuggers
ASP.NET
Native Code
SQL Server
Silverlight
Enable Edit and Continue
Package Manager Console

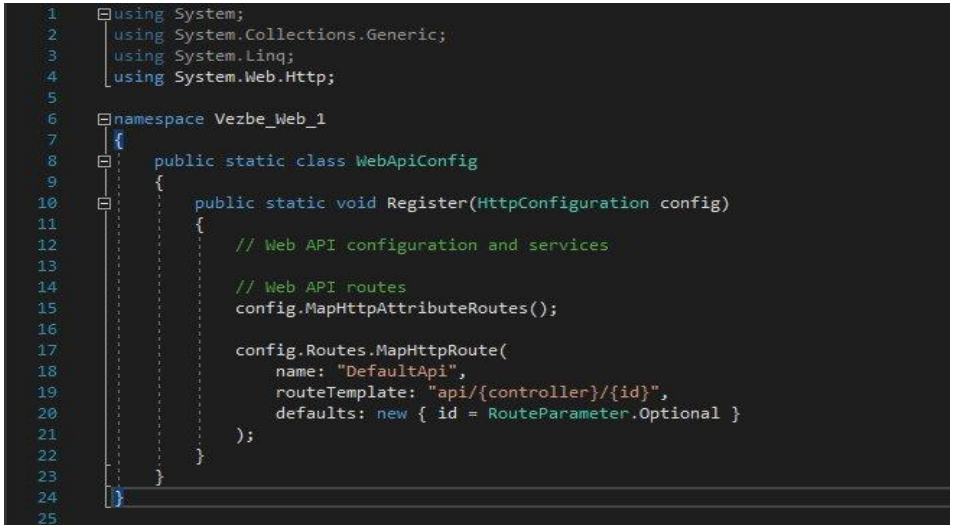
VS projekat - Aplikacija

- Global.asax
  - jedina ulazna tačka aplikacije
  - podešava aplikaciju prilikom pokretanja
  - redefiniše metodu Application_Start koja se pokreće prilikom pokretanja aplikacije
    - podesavamo ASP.NET aplikaciju - prosleđujući WebApiConfig.Register



VS projekat - Aplikacija

- WebApiConfig.cs
  - podešava kako naša aplikacija kada dobije zahtev pronalazi koji metodu treba da izvrši i u kojoj se klasi data metoda nalazi
- izgenerisani kod pretpostavlja sledeće
- http://localhost:17582/api/values/12
  - localhost:17582 - server na kome se nalazi aplikacija
  - api - putanja koju smo mi odabrali
  - values - ime kontrolera
  - 12 - dodatni parametar - id



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;

namespace Vezbe_Web_1
{
    public static class WebApiConfig
    {
        public static void Register(HttpConfiguration config)
        {
            // Web API configuration and services

            // Web API routes
            config.MapHttpAttributeRoutes();

            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );
        }
    }
}
```

VS projekat – REST Kontroler

- desni klik na folder *Controllers*
  - Add -> *Controller*
  - odabrati *Web API 2 Controller with read/write actions*
  - nazvati je *ValuesController*
- *ApiController*
  - klasa koju kontroler treba da nasledi
  - sadrži korisne metode za kontrolera
- metode se zovu po HTTP metodi koju obrađuju
  - /help - *GetHelp*
  - ovo može da se promeni
- *FromBody* - označava da će parametar biti prosleđen u telu HTTP zahteva

VS projekat – Resursi

- Resursi
  - obuhvataju sve statičke fajlove koje koristi naša aplikacija
    - slike, statičke konfiguracione fajlove, itd.

VS projekat – Pokretanje

- Aplikaciju je moguće pokrenuti iz VS-a
  - F5
  - pre ponovnog pokretanja potrebno je zaustaviti trenutni server, trenutnu aplikaciju
- aplikaciji možemo pristupiti putem web pretraživača
  - na adresi http://localhost:17582/api/values
  - port je moguće pronaći u podešavanjim aplikacije
- moguće je postaviti break point
  - break point se postavi u metodi Get koja vraća kolekciju
  - u browseru se ode na adresu http://localhost:17582/api/values
  - VS će stati sa izvršavanjem u datoj metodi

Primer 1.1 - nastavak

- Kreirati još tri REST endpoint-a
  - 1.1 endpoint koji vraća trenutni datum i vreme
    - putanja /**util/date**
  - 1.2 endpoint koji drugi vraća listu koja sadrži imena članova Vaše porodice
    - putanja **/util/family**
  - 1.3 endpoint koji vraća html stranicu sa imenima svih polaznika u grupi
    - sa naslovom Moja grupa
    - imena su prikazana u okviru tabele
    - putanja **/util/myclass**

Pomoć:
https://docs.microsoft.com/en-us/aspnet/web-api/overview/web-api-routing-and-actions/routing-in-aspnet-web-api

Zadatak 1.1 - Samostalan rad

- Kreirati još tri REST endpointa
  - 1.1 endpoint koji vraća niz celobrojnih vrednosti
    - putanja **/math/array**
  - 1.2 endpoint koji sortiran niz celobrojnih vrednosti
    - putanja **/math/sortarray**
  - 1.3 endpoint koji vraća minimalnu i maksimalnu vrednost iz niza celobrojnih vrednosti
    - putanja **/math/minmax**

Literatura

- InfoQ Explores REST (ed1, 2010)
- Pro ASP.NET Web API: HTTP Web Services in ASP.NET (Expert's Voice in .NET) (ed1, 2014)
- ASP.NET Web API dokumentacija
  - https://msdn.microsoft.com/en-us/library/hh833994(v=vs.108).aspx