**SADRŽAJ**

[**1.** **PREGLED KORIŠTENIH TEHNOLOGIJA** 2](#_Toc80083014)

[**1.1** **ASP.NET Core 3.1 API** 2](#_Toc80083015)

[**1.2** **ANGULAR 12.1** 3](#_Toc80083016)

[**1.2.1** **Vezivanje svojstava i interpolacija** 5](#_Toc80083017)

[**1.2.2** **Dijeljenje podataka između podređenih i roditeljskih direktiva i komponenti** 6](#_Toc80083018)

[**1.3** **AUTH0** 8](#_Toc80083019)

[**1.4** **VISUAL STUDIO 19** 10](#_Toc80083020)

[**1.5** **VISUAL STUDIO CODE** 10](#_Toc80083021)

[**LITERATURA** 11](#_Toc80083022)

# **PREGLED KORIŠTENIH TEHNOLOGIJA**

Za izradu praktičnog dijela diplomskog rada korišten je ASP.NET Core 3.1 API na server strani te Angular 12.1 za korisničku stranu odnosno za korisničko sučelje. Autentifikacija i autorizacija je implementirana sa Auth0 platformom. Za pisanje koda za server stranu korišten je Visual Studio 19, a korisnička strana pisana je u Visual Studio Code. Auth0 je konfiguriran na službenoj stranici (<https://auth0.com/>).

## **ASP.NET Core 3.1 API**

ASP.NET Core je višeplatformski okvir, visokih performansi, s mogućnošću implementiranja u oblaku, otvorenog koda za izgradnju modernih aplikacija povezanih s internetom. Pomoću ASP.NET Core se može:

* Izradite web aplikacije i usluge, aplikacije Internet of Things (IoT) i mobilne funkcionalne dijelove (sve ono što nije korisničko sučelje).
* Koristiti razvojne alate u sustavu Windows, macOS i Linux.
* Implementirati u oblak ili lokalno.
* Pokrenuti na .NET Core.

ASP je skraćenica od eng. Active Server Pages što znači aktivne serverske stranice

Svaka ASP.Net Core aplikacija ima fundamentalne dijelove kao što je Startup klasa, Dependency injection servis te Middleware. U Startup klasi su konfigurirane usluge koje aplikacija zahtjeva te je definiran cjevovod za upravljanje zahtjevima kao niz Middleware komponenti. ASP.NET Core uključuje ugrađeni okvir za injektirajne ovisnosti koji čini konfigurirane usluge dostupnim u cijeloj aplikaciji. Usluge se obično rješavaju injektiranjem ovisnosti pomoću injektiranja u konstruktor. Injektiranje u konstruktor se implementira deklariranjem parametara konstruktora potrebne vrste ili sučelja. Okvir za injektiranje ovisnosti pruža primjerak usluge toliko dugo ovisno o konfiguraciji usluge u Startup klasi. Cjevovod za obradu zahtjeva sastavljen je kao niz Middleware komponenti (Slika1.1). Svaka komponenta izvodi operacije na HttpContext -u i ili poziva sljedeći middleware u cjevovodu ili završava zahtjev. Konvencijom se Middleware komponente dodaje u cjevovod pozivanjem metode proširenja Use ... u metodi Startup.Configure. Na primjer, da bi se omogućilo generiranje statičkih datoteka, nazove se UseStaticFiles. [1]



*Slika 1.1. Primjer cijevovoda s tri Middleware komponente [2]*

ASP.NET Core podržava stvaranje RESTful (eng. Representational State Transfer) usluga, poznatih i kao web API (eng. Application Programming Interface), koristeći C#. Za obradu zahtjeva web API koristi kontrolere. Kontroleri u web API -ju su klase koje nasljeđuju BaseController. Namespace Microsoft.AspNetCore.Mvc pruža atribute koji se mogu koristiti za konfiguriranje ponašanja web API kontrolora i metoda djelovanja. Na primjer, atribut Route koji definira na kojoj ruti odnosno kako treba izgledati url za taj kontroler. Također postoji i atribute za definiranje na koji HTTP zahtjev metoda odgovara, npr. HttpGet, unutar takvog atributa može se definirati i rutu za taj zahtjev npr. HttpGet('all'), za ovakav primjer url će biti url servera/ruta kontrolera/ruta metode odnosno „all“ .

## **ANGULAR 12.1**

Angular je okvir za razvoj aplikacija i razvojna platforma za stvaranje učinkovitih i sofisticiranih single-page aplikacija. Angular uključuje:

* Okvir temeljen na komponentama za izradu skalabilnih web aplikacija
* Zbirka dobro integriranih knjižnica koje pokrivaju širok spektar značajki, uključujući usmjeravanje, upravljanje obrascima, komunikaciju klijent-poslužitelj i još mnogo toga
* Paket razvojnih alata koji pomažu u razvoju, izgradnji, testiranju i ažuriranju koda

Komponente su gradivni blokovi koji sastavljaju aplikaciju. Komponenta uključuje klasu pisanu u TypeScriptu s @Component () dekoratorom, HTML predloškom i stilovima. Dekorator @Component () navodi sljedeće podatke specifične za Angular:

* CSS selektor koji definira kako se komponenta koristi u HTML predlošku
* HTML element koji odgovara CSS selektoru te postaje instanca komponente, HTML predložak upućuje Angular kako iscrtati komponentu.
* Skup CSS stilova koji definiraju izgled HTML elemenata.

Svaka komponenta ima HTML predložak koji deklarira kako se ta komponenta prikazuje. Ovaj predložak definira se unutar komponente ili putem datoteke. Angular proširuje HTML s dodatnom sintaksom koja vam omogućuje umetanje dinamičkih vrijednosti iz vaše komponente. Angular automatski ažurira renderirani DOM kad se promijeni stanje vaše komponente. Angular također podržava property binding kako bi pomogao developeru postaviti vrijednosti za svojstva i atribute HTML elemenata i proslijediti vrijednosti u prezentacijsku logiku aplikacije. Također se mogu dodati event listener-i da slušaju i odgovaraju na korisničke radnje kao što su pritisci tipki, pokreti miša, klikovi i dodiri. Developeri mogu dodati dodatne funkcionalnosti svojim predlošcima korištenjem direktiva. Najpopularnije direktive u Angular -u su \*ngIf i \*ngFor. Direktive se koriste za izvođenje različitih zadataka, kao što je dinamička izmjena strukture DOM -a. Angular -ovi deklarativni predlošci omogućuju jasno odvajanje logike aplikacije od prezentacije. Predlošci se temelje na standardnom HTML -u radi lakše izgradnje, održavanja i ažuriranja.

Ubrizgavanje ovisnosti omogućuje deklariranje ovisnosti klasa TypeScript -a bez brige o njihovom stvaranju. Umjesto toga, Angular vodi instalaciju. Ovaj uzorak dizajna omogućuje pisanje više provjerljivog i fleksibilnijeg koda.

Angular CLI najbrži je, jednostavan i preporučen način za razvoj Angular aplikacija. Angular CLI čini brojne zadatke bez problema. Evo nekoliko primjera:

* ng build - kompajlira aplikaciju u izlazni direktorij
* ng serve - sastavlja i služi aplikaciju, obnavljajući promjene datoteka
* ng generation - generira ili mijenja datoteke na temelju sheme
* ng test - pokreće jedinične testove za dati projekt
* ng e2e - izrađuje i opslužuje Angular aplikaciju, a zatim izvodi end-to-end testove [3]

### **Vezivanje svojstava i interpolacija**

Prilikom izrade aplikacija pomoću programa Angular dolazi se u dodir s nekoliko načina za prikazivanje podataka u HTML prikazu. Vezivanje svojstava i interpolacija su vrste vezivanja podataka u Angular -u, koje se koriste za premještanje podataka iz komponente u predložak. Vezivanje podataka vrlo je važan i moćan aspekt razvoja softvera. Uključuje koncept definiranja komunikacije između komponente i njezinih stavova. Vezanje podataka omogućuje dinamiku i interaktivnost u aplikacijama. U Angular -u postoje četiri vrste povezivanja podataka:

* Vezivanje događaja
* Dvosmjerno vezivanje podataka
* Interpolacija
* Vezivanje svojstava

Vezivanje događaja, povezivanje podataka je kada informacije teku iz HTLM prikaza u komponentu kada se događaj pokrene. Prikaz šalje podatke iz događaja poput klika na gumb koji će se koristiti za ažuriranje komponente. To je upravo suprotno od vezivanja svojstava, gdje podaci idu od komponente do pogleda. Primjer ove vrste povezivanja podataka može se opisati primjerom u nastavku:

<p>My name is {{name}}</p>

<button (click)="updateName()">Update button</button>

// component.ts

updateName() {

this.name = 'John';

Dvosmjerno povezivanje je mehanizam u kojem podaci teku u oba smjera od komponente do prikaza i natrag. Komponenta i prikaz uvijek su sinkronizirani, a promjene napravljene na oba kraja odmah se ažuriraju u oba smjera. Dvosmjerno vezivanje obično se koristi pri radu s obrascima gdje se korisnički unos koristi za ažuriranje stanja komponente i obrnuto. Angular-ova dvosmjerna sintaksa povezivanja kombinacija je uglatih zagrada i zagrada, [()]. Sintaksa [()] kombinira zagrade vezivanja svojstava, [], sa zagradama vezivanja događaja, (), kako slijedi:

<app-sizer [(size)]="fontSizePx"></app-sizer>

Interpolacija je mehanizm vezivanja podataka kojim prikazujemo dinamički tekst definiran unutar komponente na HTMl prikazu. U tehnici, tekst koji predstavlja varijable u komponentama postavlja se između dvostrukih uvrnutih zagrada u predlošku. Angular pronalazi varijablu koja odgovara tekstu u komponenti i zamjenjuje tekst vrijednošću dodijeljenom varijabli. Brojevi, datumi itd. Mogu se koristiti izravno između uvrnutih zagrada. Primjer:

<p>{{ name }}</p>

// component.ts name = 'Peter';

Vezanje svojstava je jednosmjerni mehanizam koji omogućuje postavljanje svojstva elementa prikaza. Uključuje ažuriranje vrijednosti svojstva u komponenti i njegovo vezanje za element u predlošku prikaza. Vezivanje svojstava koristi sintaksu [] za povezivanje podataka.[5] Primjer je postavljanje onemogućenog stanja gumba:

<button [disabled]="buttonDisabled"></button>

// component.ts buttonDisabled = true;

### **Dijeljenje podataka između podređenih i roditeljskih direktiva i komponenti**

Uobičajeni uzorak u Angular -u je dijeljenje podataka između roditeljske komponente i jedne ili više podređenih komponenti. Za implementaciju ovog uzorka koristite dekoratere @Input () i @Output (). @Input () i @Output () daju podređenoj komponenti način komunikacije s nadređenom komponentom. @Input () dopušta roditeljskoj komponenti ažuriranje podataka u podređenoj komponenti. Nasuprot tome, @Output () dopušta djetetu da šalje podatke roditeljskoj komponenti.

@Input () dekorator u podređenoj komponenti ili direktivi označava da svojstvo može primiti svoju vrijednost od svoje nadređene komponente.

Graphical user interface, application

Description automatically generated

*Slika 1.2. Dijagram @Input dekoratora [4]*

Diagram

Description automatically generated

*Slika 1.3. Primjer kako implementirati djeljenje podataka nadređene komponente s podređenom komponentom[4]*

Cilj u uglatim zagradama, [], je svojstvo koje ima @Input () u podređenoj komponenti. Izvor vezanja, dio desno od znaka jednakosti, su podaci koje nadređena komponenta prosljeđuje ugniježđenoj komponenti.

Diagram

Description automatically generated with medium confidence

*Slika 1.4. Dijagram @Output dekoratora [4]*

A picture containing timeline

Description automatically generated

*Slika 1.5. Primjer kako implementirati djeljenje podataka među komponentama[4]*

Podređeni selektor je <app-input-output> sa „item“ i „deleteRequest“ svojstvima @Input () i @Output () u klasi podređene komponente. Svojstvo „currentItem“ i metoda „crossOffItem ()“ su u klasi nadređene komponente.[4]

## **AUTH0**

Auth0 fleksibilno je rješenje za dodavanje usluga provjere autentičnosti i autorizacije aplikacijama. Tim ili organizacija mogu izbjeći troškove, vrijeme i rizik koji proizlaze iz izgradnje vlastitog rješenja za provjeru autentičnosti i autorizaciju korisnika. Nekoliko slučajeva upotrebe Auth0:

* Korisnici se trebaju moći prijaviti s korisničkim imenom/lozinkom ili sa svojim društvenim računima (poput Facebooka ili Twittera). Trebaju moći diohvatiti svoj korisnički profil nakon prijave kako bi mogli prilagoditi korisničko sučelje i primijeniti pravila autorizacije.
* Zaštita API-ja
* Zaštita više od jedne aplikacije jedinstvenom prijavom (SSO).
* Sigurran pristup API-ju JavaScript front-end aplikacije i mobilne aplikacije.
* Autentifikacija korisnika pomoću jezika za označavanje sigurnosnih potvrda (SAML).
* U slučaju neispravne lozinke korisnici se mogu prijaviti pomoću jednokratnih kodova dostavljenih e-poštom ili SMS-om.
* Ako je jedna od adresa e-pošte korisnika ugrožena zbog povrede javnih podataka neke web lokacije, vlasnik i korisnik će biti obaviješten ili će onemogućiti korisniku prijavu u aplikaciju dok ne poništi lozinku.
* Proaktivno blokiranje sumnjive IP adrese ako uzastopno pokušava neuspješnu prijavu, kako bi se izbjegli DDoS napadi.
* Zaposlenicima velikih organizacija omogućuje prijavu u različite interne aplikacije i aplikacije trećih strana koristeći svoje postojeće vjerodajnice za poduzeće.
* Poništavanje lozinke, stvaranje, omogućavanje, blokiranje i brisanje korisnika te korisničko sučelje za upravljanje svim tim.
* Višefaktorska provjera autentičnosti (MFA) kada korisnici žele pristupiti osjetljivim podacima.
* Identitet koji je na vrhu stalno rastućih zahtjeva usklađenosti SOC2, GDPR, PCI DSS, HIPAA i drugih.
* Praćenje korisnike na svojoj web lokaciji ili u aplikaciji. Korištenje ovih podatke za stvaranje tokova, mjerenje zadržavanja korisnika i poboljšanje tijeka prijave[6]

Implementacija Auth0-a je podjenjana na dio za SPA i dio za API. Implementacija nutar angulara odnosno za SPA se vrši instaliranjem paketa „auth0-angular“ te korištenjem servisa tog paketa. Servis omogućuje prijavu i odjavu, dohvat korisničkih podataka te još mnogo toga. Sva ostala konfiguracija, dodavanje korisničkih rola, dodavanje korisnika, uređivanje prozora za prijavu i mnogo dugih svari se vrši preko auth0 stranice vrlo lako i jednostavno. Zaštita API-ja se vrši dodavanjem koda za JSON web tokene (JWT), naime Auth0 koristi JWT za slanje korisničkih podataka na server pa na strani servera nije potrebno ništa osim logike za čitaje podataka iz JWT-ova.

JSON Web Token (JWT) je otvoreni standard (RFC 7519) koji definira kompaktan i samostalan način za siguran prijenos informacija između strana kao JSON objekt. Ti se podaci mogu provjeriti i vjerovati im jer su digitalno potpisani. JWT -ovi se mogu potpisati pomoću tajnog (s HMAC algoritmom) ili para javnih/privatnih ključeva koristeći RSA ili ECDSA.[7]

## **VISUAL STUDIO 19**

Microsoft Visual Studio je IDE koji je napravio Microsoft i koristi se za različite vrste razvoja softvera, poput računalnih programa, web stranica, web aplikacija, web usluga i mobilnih aplikacija. Sadrži alate za dovršavanje, kompajlere i druge značajke koje olakšavaju proces razvoja softvera.

Visual Studio IDE (integrirano razvojno okruženje) softverski je program za programere za pisanje i uređivanje koda. Njegovo korisničko sučelje koristi se za razvoj softvera za uređivanje, ispravljanje pogrešaka i izradu koda. Visual Studio uključuje uređivač koda koji podržava IntelliSense (komponentu dovršetka koda), kao i preradu koda. Integrirani alat za ispravljanje pogrešaka radi i kao ispravljač pogrešaka na izvornoj razini i na računaru. Ostali ugrađeni alati uključuju kodiranje profila, dizajner za izradu GUI aplikacija, web dizajner, dizajner klasa i dizajner sheme baze podataka.[8]

## **VISUAL STUDIO CODE**

Visual Studio Code lagani je, ali moćan uređivač izvornog koda koji radi na računalu i dostupan je za Windows, macOS i Linux. Dolazi s ugrađenom podrškom za JavaScript, TypeScript i Node.js te ima bogati sustav proširenja za druge jezike (kao što su C ++, C#, Java, Python, PHP, Go) i vrijeme izvođenja (kao što su .NET i Unity).

Visual Studio Code sadrži uređivač izvornog koda, savršen za svakodnevnu upotrebu. Uz podršku za stotine jezika, VS Code pomaže korisniku da bude produktivan. S isticanjem sintakse, podudaranjem zagrada, automatskim uvlačenjem, odabirom okvira, isječcima i još mnogo toga. Intuitivni prečaci na tipkovnici, jednostavno prilagođavanje i mapiranje prečaca na tipkovnici omogućavaju korisniku da s lakoćom upravljata svojim kodom.[9]

# **LITERATURA**

[1] [Anderson](https://twitter.com/RickAndMSFT), R. i [Smith](https://ardalis.com/), S.: „ASP.NET Core fundamentals“, s Interneta, https://docs.microsoft.com/en-us/aspnet/core/fundamentals/?view=aspnetcore-3.1&tabs=linux, 30.03.2020.

[2] [Anderson](https://twitter.com/RickAndMSFT), R. i [Smith](https://ardalis.com/), S.: „ASP.NET Core Middleware“, s Interneta, <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/middleware/?view=aspnetcore-3.1>, 15.07.2020.

[3] Angular Tim: „What is Angular?“, s Interneta, <https://angular.io/guide/what-is-angular>, 08.03.2021.

[4] Angular Tim: „Sharing data between child and parent directives and components“, s Interneta, <https://angular.io/guide/what-is-angular>, 08.03.2021.

[5] Nwamba, C: „Understanding Angular Property Binding and Interpolation“, s Interneta, https://www.telerik.com/blogs/understanding-angular-property-binding-and-interpolation, 03.06.2019.

[6] Auth0 Tim: „Understand How You Can Use Auth0“, s Interneta, https://auth0.com/docs/get-started/auth0-overview, 05.02.2020.

[7] JWT Tim: „Introduction to JSON Web Tokens“ , s Interneta, <https://jwt.io/introduction>, 20.05.2019.

[8] Nepoznat Autor, „Visual Studio“, s Interneta, <https://www.incredibuild.com/integrations/visual-studio>, 17.08.2021.

[9] VS Code Tim, „Why did we build Visual Studio Code?“, s Interneta, https://code.visualstudio.com/docs/editor/whyvscode, 17.08.2021.