**SADRŽAJ**

[**1** **PREGLED KORIŠTENIH TEHNOLOGIJA** 2](#_Toc80797458)

[**1.1** **ASP.NET CORE 3.1 API** 2](#_Toc80797459)

[**1.2** **ANGULAR 12.1** 3](#_Toc80797460)

[**1.2.1** **Vezivanje svojstava i interpolacija** 5](#_Toc80797461)

[**1.2.2** **Dijeljenje podataka između podređenih i roditeljskih direktiva i komponenti** 6](#_Toc80797462)

[**1.3** **SOFTVER ZA UPRAVLJANJE IDENTITETOM I PRISTUPOM KORISNIKA** 8](#_Toc80797463)

[**1.3.1** **Auth0** 10](#_Toc80797464)

[**1.4** **VISUAL STUDIO 19** 11](#_Toc80797465)

[**1.5** **VISUAL STUDIO CODE** 12](#_Toc80797466)

[**2** **DIZAJN RJEŠENJA** 13](#_Toc80797467)

[**2.1** **BAZA PODATAKA** 15](#_Toc80797468)

[**2.2** **SERVER STRANA APLIKACIJE** 17](#_Toc80797469)

[**2.2.1** **API razina** 17](#_Toc80797470)

[**2.2.2** **Domenska razina** 19](#_Toc80797471)

[**2.2.3** **Infrastrukturna razina** 20](#_Toc80797472)

[**LITERATURA** 22](#_Toc80797473)

# **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]

## **SOFTVER ZA UPRAVLJANJE IDENTITETOM I PRISTUPOM KORISNIKA**

Softver za upravljanje identitetom i pristupom korisnika (eng. Customer Identity and Access Management - CIAM) široko se koristi i mnogi ljudi traže softverska rješenja koja se lako administriraju i štede vrijeme uz mogućnost društvene prijave te upravljanje pristankom i preferencijama. Drugi važni čimbenici koje treba uzeti u obzir pri odabiru softvera uključuju upravljanje korisnicima i sigurnost. U nastavku (Tablica 1.1) predstavljam deset najpopularnijih softvera. A za implementaciju je odabran Auth0 softver. [10]

|  |  |
| --- | --- |
| Softver | Opis |
| Auth0 | Auth0 pruža platformu za provjeru autentičnosti, autorizaciju i siguran pristup aplikacijama, uređajima i korisnicima. Razvojni timovi oslanjaju se na Auth0 -ovu jednostavnost, proširivost i stručnost kako bi platforma bila prikladna za sve. |
| OneLogin | Moguće proširenje sigurnosti i usklađenosti aplikacije na sve javne i privatne aplikacije u oblaku sa sigurnom jedinstvenom prijavom, višefaktorskom autentifikacijom i opskrbom korisnika. |
| Okta | Okta je usluga upravljanja identitetom i pristupom na zahtjev za web aplikacije, u oblaku i iza vatrozida. |
| Ping Indentity | Ping Identity nudi inteligentne mogućnosti identiteta, kao što su jednokratna prijava i višefaktorska autentifikacija za radnu snagu, korisnike i partnere. |
| CyberArk Identity | Današnja nova stvarnost zahtijeva novu vrstu pristupne platforme. Izgrađen na nultom povjerenju, CyberArk Identity stvara novu eru - siguran pristup posvuda, koja jedinstveno kombinira vodeće tehnologije za besprijekornu integraciju. |
| WSO2 Identity Server | Poslužitelj identiteta djeluje kao sabirnica identiteta u poduzeću, središnja okosnica za povezivanje i upravljanje više identiteta bez obzira na standarde na kojima se temelje. |
| Jump Cloud | JumpCloud je potpuna platforma za identitet, pristup i upravljanje uređajima. |
| LastPass | Poslovna rješenja LastPass pomažu timovima i tvrtkama da preuzmu kontrolu nad upravljanjem svojim identitetom pomoću upravljanja lozinkom, jedinstvene prijave i prilagodljive višefaktorske provjere autentičnosti. |
| Microsoft Azure Active Directory | Azure Active Directory sveobuhvatno je rješenje u oblaku za upravljanje identitetom i pristupom koje pruža robustan skup mogućnosti za upravljanje korisnicima i grupama i pomaže u sigurnom pristupu aplikacijama, uključujući Microsoftove mrežne usluge poput Office 365 i svijet aplikacija koje ne pripadaju Microsoftu. |
| SecureAuth Identity Platform | SecureAuth je tvrtka za zaštitu identiteta koja zaposlenicima, partnerima i klijentima omogućuje najsigurnije i fleksibilnije iskustvo autentifikacije. Isporučuje se kao usluga i primjenjuje u oblačnim, hibridnim i lokalnim okruženjima, SecureAuth upravlja i štiti pristup aplikacijama, sustavima i podacima u velikom opsegu, bilo gdje u svijetu. Tvrtka pruža alate za ugradnju sigurnosti identiteta u nove i postojeće aplikacije i tijekove rada bez utjecaja na korisničko iskustvo ili angažman, što rezultira povećanom produktivnošću i smanjenim rizikom. |

*Tablica 1.1. Deset softvera za* *upravljanje identitetom i pristupom korisnika [10]*

### **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]

# **DIZAJN RJEŠENJA**

Aplikacija je sačinjena od četiri glavna dijela: baza podataka, serverski dio aplikacije, klijentski dio aplikacije te softver za upravljanje identitetom i pristupom korisnika. Prije razrade glavnih djelova bitno je predstaviti funkcionalne zahtjeve (Tablica 2.1.). Za dizajniranje rješenja najprije je potrebno imati zahtjeve koje je potrebno ispuniti tim dizajnom.

|  |  |
| --- | --- |
| Redni broj zahtjeva | Opis zahtjeva |
| 1. | Trebaju postojati dvije uloge, uloga korisnika, svi korisnici s korisničkim računom i uloga administratora, samo jedan račun koji pripada vlasniku noćnog kluba |
| 2. | Korisnik treba moći izraditi račun, a za izradu treba upisati puno ime, email adresu te lozinku. |
| 3. | Korisnik treba imati mogućnost prijave u sustav ali za prijavu treba imati kreiran korisnički račun ili se može prijaviti preko neke od platformi (Google ili Facebook). |
| 4. | Navigacijska traka u ljevom kutu treba imati linkove „NightClub“, „Reserve“ i „News“ a u desnom „Login“ link. |
| 5. | Kada se korisnik uspješno prijavi u desnom kutu navigacijske trake bi se link „Login“ trebao pretvoriti u „Logout“ te ljevo od tog linka bi se trebao pojavi link s imenom korisnika koji vodi na stranicu s osobnim podatcima. |
| 6. | Ako se u sustav prijavi administrator onda se umjesto imena treba pojavi link „Admin center“ |
| 7. | Korisnik treba imat mogućnost odjave iz sustava. |
| 8. | Kada se korisnik odjavi iz sustava aplikacija bi trebala preći na početnu stranicu te se izbrisati link s imenom (ili „Admin center“ link) s navigacijske trake te ponovno prikazati „Login“ link. |
| 9. | Svi korisnici bez obzira jesu li prijavljeni ili neprijavljeni u sustav trebaju moći vidjeti početnu stranicu („NightClub“ link). |
| 10. | Početna stranica treba sadržavati „Reseve“ link, tri umanjena članka (naslov, slika i datum događaja) te dio o klubu s slikom. |
| 11. | Samo prijavljeni korisnik može pristupiti starnici za rezervaciju („Reserve“ link). Ako korisnik koji nije prijavljen pokuša pristupiti stanici za rezrvaciju u novom prozru bi mu se trebala prikazati stranica za prijavu u sustav. |
| 12. | Svi korisnici bez obzira jesu li prijavljeni ili neprijavljeni u sustav trebaju moći vidjeti stranicu s novostima („News“ link). |
| 13. | Na stranici s novostima se trebaju nalaziti članci koji sadrže naslov, sliku, sadržaj, datum događaja te datum posljednjeg mijenjanja članka. |
| 14. | Na stranici za rezervaciju se treba nalaziti tlocrt noćnog kluba s ucrtanim stolovim, te svaki stol treba imati svoj redni broj. |
| 15. | Korisnik prije odabira stola treba odabrati datum rezervacije da bi sustav filtrirao slobodne stolove za taj datum, a dozvoljeni datumi su od toga dana pa unaprijed period koji postavlja administrator. Nakon odabira datuma nedostupni stolovi biti će onemogućeni za klikanje |
| 16. | Korisnik klikom na stol na tlocrtu odabire stol koji će rezervirati, zatimn treba odabrati broj gostiju koji će doći sa njim te po želji može napisati napomenu. Klikom na tipku spremi, stol je rezerviran. |
| 17. | Jedan korisnik može rezervirati jedan stol po danu. Iznimno administrator može rezervirati više stolova za jedan datum s tim da treba upisati puno ime onoga kome je rezervacija namjenjena. |
| 18. | Svaki korisnik trebao bi moći vidjeti svoje puno ime, email adresu te rezervacije na stranici s osobnim podatcima. |
| 19. | Rezervacije trebaju imati tri statusa: aktivna, prošla, otkazana. Korisnik treba moći otkazati rezervaciju koja je u statusu aktivna. |
| 20. | Administrator treba moći vidjeti sve aktivne rezervacije po datumima,  odabirom datuma u padajućem izborniku trebaju se prikazati rezervacije za odabrani datum. |
| 21. | Administrator treba moći otkazati sve rezervacije za odabrani datum s tim da je obavezan napisati obrazloženje. |
| 22. | Administrator treba moći dodavati ili uklanjati članke sa stranice s novostima. |
| 23. | Pri dodavanju novog članka potrebno je unijeti naslov, sadržaj, datum događaja te naslovnu sliku događaja. |
| 24. | Članci na stranici s novostima koji imaju datum događaja u budućnosti trebaju imati tipku rezerviraj, klikom na tu tipku aplikacija treba preći na stranicu za rezerviranje stola i postaviti datum rezervacije na datum tog događaja. |
| 25. | Administrator treba moći mijenjati vrijednost minimalnog račun te vrijednost maksimalnog broja gostiju za svaku kategoriju stola (standardni, zidni te VIP). |

*Tablica 2.1. Funkci0onalni zahtjevi*

## **BAZA PODATAKA**

Za bazu podataka je izabrana MSSQL baza podataka. Za dizajniranje baze podataka je korišten „Code first“ princip. Definirani su modeli te mapiranje, a Enitiy Framework-om odrađene migracije modela u tablice. Entity framework je programski okvir koji vrši migracije modela iz C# programskog jezika u tablice SQL baze podataka. Također služi za dohvaćanje podataka iz tablica i još mnogo toga ali to nije tema ovog rada.

Diagram

Description automatically generated

*Slika 2.1. Dijagram baze podataka za tablice vezane uz rezervacije*

Na Slici 2.1. vidimo dijagram tablica vezanih uz rezervacije, prikazane su četiri tablice te relacije među tim tablicama. Tablica „Reservations“ sadrži „TableId“ odnosno jedna rezervacija ima jedan stol, što značio da jedan stol može biti povezan s više rezervacija (odnos 1:N). Također jedna rezervacija ima jedan „UserStringId“ pa to znači da ima jednog korisnika, odnosno jedan korisnik može biti povezan s više rezervacija (odnos 1:N). Također svaki stol ima „CategoryId“ što znači da svaki stol ima jednu kategoriju, odnosno svaka kategorija može imati više stolova (odnos 1:N).

Table

Description automatically generated with medium confidence

*Slika 2.2 Dijagram preostalih tablica u bazi*

Na Slici 2.2. je prikazan dijagram preostalih tablica u bazi. Tablica „AdminConfigs“ i tablica „Articles“ nisu relacijski povezane. U bazi još imamo view „vwAdminConfigs“ (Slika 2.3.) koji je shematski potpuno isti kao i tablica ali on još dohvaća podatke iz tablice „Categories“ te za svaku kategoriju doda dva retka u view, jedan za „MinBillValue“ a drugi za „MaxNumberOfGusets“. Također kreirana je i jedna stored-procedure „stpUpdateAdminConfig“ (Slika 2.4.) koja služi za mijenajane podataka koje prikazuje „vwAdminConfigs“, ta je procedura potrebna jer ovaj view ne zadovoljava uvjete da bi se preko njega mogle raditi promjene podataka.

Text

Description automatically generated

*Slika 2.3. View „vwAdminConfigs“*

A picture containing text

Description automatically generated

*Slika 2.4. Stored-procedure „stpUpdateAdminConfig“*

## **SERVER STRANA APLIKACIJE**

Server staran aplikacije je dio aplikacije koji se izvršava na udaljenom serveru, te je dizajnirana u tri razine:

* API – (eng. active page interface) ovaj dio aplikacije služi za posluživanje podataka klijentskom dijelu aplikacije.
* Domain – u ovom dijelu su definirani modeli i sučelja te implementirani servisi.
* Infrastructure – služi za rad sa bazom, definirani su DbContext, mapiranje, migracije te implementirani repozitoriji.

### **API razina**

U API razini su definirani kontroleri, Dtos (eng. data transfer objects), autorizacija, konfiguracija, Startup klasa te appsetings.json. Startup klasa je predstavljena u uvodu te je nije potrebno dodatno objašnjavati. Appsetings.json je datoteka u kojoj definiramo neke konstantne podatke, u json formatu, koje možemo mjeijenjati a da pri tom ne moramo ponovno „Build-ati“ aplikaciju. U ovoj datoteci je drfiniran konekcijski string za bazu te podatci za zaštitu API-a s auth0.

Kontroleri služe za definiranje krajnjih točaka koje će pozivati klijentska strana aplikacije. Aplikacija ima sedam kontrolera, odnosno jedan apstraktni „MainController“ kojeg nasljeđuju svi ostali redom:

* AdminConfigsController – služi za dohvaćanje i mijenjane podataka iz vwAdminConfigs view-a, a krajnja točka mu je „/api/adminconfigs“.
* ArticlesController – služi za dodavanje, mijenjanje, dohvaćanje te brisanje artikala, krajnja točka je „/api/articles“.
* CategoriesController – služi za dohvat svih kategorija artikla, kranja točk je „/api/categories“.
* ReservationsController – ovo je najkompliciraniji kontroler u aplikaciji i služi za dohvat svih rezervacija za određeni datum, dohvat svih rezervacija za trenutnog korisnika, dohvat svih rezerviranih datuma za trenutnog korisnika, dohvat svih rezerviranih datuma, dodavanje rezervacije, otkazivanje rezervacije te otkazivanje svih rezervacija za određeni datum. Krajnja točka je „/api/reservations“
* TablesController – služi za dohvat svih stolova, krajnja točka je „/api/tables“
* UsersController – služi za dodavanje novog korisnika „/api/users“.

Autentifikacija se sastoji od „IsAdminAuthorizationHandler“ i „IsAdminAuthorizationRequirement“. Ova autentifikacija služi za provjeru da li prijavljeni korisnik ima „admi“. AuthorizationHandler-u je implementorana logika provjere a u AutorizationRequirement-u postavljena dozvoljena rola.

Konfiguracija se od četiri različite konfiguracije te su njome konfigurirani servisi i mapiranje modela u Dtos, a to su redom:

* AuthtenticationConfig – konfiguriranje servisa za autentifikaciju.
* AuthorizationConfig – konfiguriranje servisa za autorizaciju.
* AutomapperConfig – konfiguriranje mapiranja modela u objekte za prijenos (Dtos eng. data transfer objects)
* DependencyInjectionConfig – konfiguriranje injektiranja ovisnosti

Objekti za prijenos podataka (Dto eng. data transfer objects) su klase koje sadrže dio svojstava koje sadrže modeli ovisno za što je objekt za prijenos namjenjen. Objekt za prijenos može biti namjenjen za dodavanje, uređivanje ili za prikaz rezultata. Objekti za prijenos se koriste da se smanji prijenos podataka, odnosno da se ne prenose nepotrebni podatci koje sadrži model a nisu potrebni za tu vrstu zahtjeva. Također se koriste za sakrivanje podataka od korisnika, neke podatke koji su sadržani u modelu korisnik nesmije vidjeti već samo služe za rad aplikacije.

### **Domenska razina**

U ovoj razini imat ćemo sučelja i servise te modele. Koristit ćemo sučelja za rad s injektiranjem ovisnosti, to je slovo D SOLID principa, to je načelo inverzije ovisnosti (poznato i kao IoC eng. inversion of control). Neću objašnjavati SOLID jer to nije tema ovog rada. To znači da ništa ne trebala ovisiti o implementaciji (konkretna klasa), ali morama ovisiti o apstrakciji (ovo može biti sučelje ili bilo koji drugi način apstrakcije). Zbog tog ako se nešto promijeni u klasi, to neće utjecati na ostale klase jer to ovisi o apstrakciji. U klase servisa se dodaju poslovna pravila aplikacije. Te klase su između kontrolora i repozitorija. Aplikacijski sloj ne bi trebao imati pristup infrastrukturnom sloju, pa se klase servisa stavljaju u sloju domene koji se koristiti kao posrednik između kontrolera u aplikacijskom sloju i repozirorija u infrastrukturnom sloju. Također, u kontroler se ne dodavaju poslovna pravila, pa će kontrolori pozvati servise u koje će biti pravila poslovanja, a servisi će pozvati repozitorije. Modeli služe za migracije, odnosno kreiranje tablica „Code first“ principom, te za sve akcije nad bazom podatci se mapiraju iz modela odnosno u modele.

Aplikacija sadrži osam modela od kojih je jedan apsrtaktni te ga nasljeđuju svi drugi modeli:

* AdminConfig
* AdminCustomConfig
* Article
* Category
* Reservation
* Table
* User

Također sadrži trinaes sučelja od kojih je jedan apstraktni i nasljeđuju ga sva repositori sučelja:

* IAdminConfigRepository
* IAdminConfigService
* IArticleRepository
* IArticleService
* ICategoryRepository
* ICategoryService
* IRepository
* IReservationRepository
* IReservationService
* ITableRepository
* ITableService
* IUserRepository
* IuserService

Te sadrži šest servisa koji implementiraju servis sučelja:

* AdminConfigService
* ArticleService
* CategoryService
* ReservationService
* TableService
* UserService

### **Infrastrukturna razina**

Ova razina sadrži context, mapiranje, migracije te repozitorije. Migracije sadrže klase s svim generiranim migracijama pomoću kojih je izrađena baza. U contextu se definiraju modeli koji će se koristiti u radu s bazom. Konfiguriraju opće postavke mapiranja te se dodaje mapiranje objekata definirano u datotekama za mapiranje sadržanim u mapi „Mappings“. Ta mapa sadrži sedam datoteka za mapiranje:

* AdminConfigMapping
* AdminCustomConfigMapping
* ArticleMapping
* CategoryMapping
* ReservationMapping
* TableMapping
* UserMapping

Pomoću repozitorija se odvoja logika koja dohvaća podatke i preslikava ih u model entiteta od poslovne logike koja djeluje na model. Poslovna logika trebala bi biti agnostička prema vrsti podataka koja sadrži razina izvora podataka. Na primjer, razina izvora podataka može biti baza podataka, excel popis ili web usluga. Repozitorij posreduje između razine izvora podataka i poslovnih razina aplikacije. On traži izvor podataka, preslikava podatke iz izvora podataka u poslovni model. Repozitorij odvaja poslovnu logiku od interakcija s temeljnim izvorom podataka ili web uslugom. Razdvajanje između podatkovnih i poslovnih razina ima tri prednosti:

* Centralizira logiku podataka ili logiku pristupa web uslugama.
* Pruža zamjensku točku za testove.
* Pruža fleksibilnu arhitekturu koja se može prilagoditi s razvojem cjelokupnog dizajna aplikacije.[11]

Definiran je generički repozitorij „Repository“, to je apstraktna klasa, što znači da se ta klasa ne može stvoriti, već samo naslijediti. Sve posebne klase repozitorija naslijeđuju od ove glavne klase. U ovoj klasi su implementirane metode iz sučelja „IRepository“. Postoje neke virtualne metode, a razlog za to je dopustiti da se nadjača u drugoj specifičnoj klasi repozitorija ako je to potrebno. Također je implementirana metoda „Dispose“ jer se koristi za oslobađanje memorije u aplikaciji. Repozitoriji koji nasljeđuju generički:

* AdminConfigRepository
* ArticleRepository
* CategoryRepository
* ReservationRepository
* TableRepository
* UserRepository

# **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.

[10] Nepoznat Autor, „Auth0 Alternatives & Competitors“, s Interneta, https://www.g2.com/products/auth0/competitors/alternatives, 24.08.2021.

[11] Microsoft Tim, „The Repository Pattern“, s Interneta, https://docs.microsoft.com/pt-br/previous-versions/msp-n-p/ff649690(v=pandp.10), 27.04.2010.