UNIVERZITET U ZENICI

Politehnički fakultet

Softversko inženjerstvo

Web Programiranje

Emir Varupa

Upotreba .NET i Angular web tehnologija u izgradnji Internet forum web aplikacije

Diplomski rad

Mentor: doc. dr. Đulaga Hadžić

Zenica, 2022.godine

BIBLIOGRAFSKA KARTICA RADA

NAUČNO PODRUČJE: Softversko inženjerstvo

NAUČNO POLJE: Programiranje

NAUČNA GRANA: Web programiranje

USTANOVA U KOJOJ JE RAD IZRAĐEN: Politehnički fakultet u Zenici

MENTOR RADA: doc. dr. Đulaga Hadžić

DATUM ODBRANE RADA: 01.10.2022.godine

ČLANOVI KOMISIJE ZA ODBRANU:

1. Prof.dr. Samir Lemeš, predsjednik Komisije
2. Van.prof.dr. Nevzudin Buzađija, član Komisije
3. Doc.dr. Đulaga Hadžić, član Komisije i Mentor
4. V.ass. Vahid Redžić, dipl.inž.građ., sekretar Komisije

IZJAVA

Izjavljujem da sam diplomski rad pod naslovom „ Upotreba .NET i Angular web tehnologija u izgradnji Internet forum web aplikacije “ izradio samostalno pod mentorstvom doc. dr. Đulage Hadžića. U radu sam koristio literaturu koja je navedena na kraju diplomskog rada. Tuđe spoznaje, zaključke, stavove, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u diplomskom radu na uobičajan standaran način sam povezao s fusnotama i korištenim bibliografskim jedinicama.

SAŽETAK

U ovom diplomskom radu objašnjeno je šta su radni okviri i koji se najviše koriste. Zatim su predstavljeni okviri koji su omogućili lakšu i bržu implementaciju Internet foruma, a to su: Angular i .NET. Opisane su njihove prednosti i karakteristike, kao i načini implementacije. Backend API je u ovom slučaju implementiran pomoću repozitorijuma, servisa i kontrolera, a frontend je implementiran pomoću komponenti, angular materiala, presretača zahtjeva i rutiranja. U svrhu lakše implementacije ogledne web aplikacije, izvršena je analiza i dizajn Internet foruma. Zatim su prikazane glavne komponente i funkcionalnosti ogledne web stranice i na kraju objašnjeni sigurnosni elementi koji ograničavaju pristup korisnika.

ABSTRACT

This diploma thesis explains what frameworks are and which ones are used the most. Next, the frameworks that enabled easier and faster implementation of the Internet forum were presented, namely: Angular and .NET. Their advantages and characteristics are described, alongside their implementation methods. In this case, the backend API is implemented using repositories, services and controllers, the frontend was implemented using components, angular material, interceptors and routing. In order to facilitate the implementation of the sample web application, the analysis and design of the Internet forum was carried out. Next, the main components and functionalities of the sample website are shown, and finally, security elements that limit user access in the app were explained.

Sadržaj

1. Uvod

Radni okvir *(eng. framework)* je struktura na kojoj se gradi softver. Služi kao temelj u razvoju softvera, tako da se ne počinje potpuno od nule. Obično su povezani sa određenim programskim jezikom i prilagođeni su različitim vrstama zadataka.

Korištenje radnih okvira štedi vrijeme i smanjuje rizik od grešaka. Pošto se ne počinje potpuno od nule, manja je šansa da se unesu greške. Osim toga, radni okviri su već testirani od strane programera. Ostale prednosti uključuju:

- Sigurniji kod
- Jednostavnije testiranje i otklanjanje grešaka
- Izbjegavanje dupliranja koda
- Čist i lako prilagodljiv kod
- Može se fokusirati na pisanje koda specifičnog za projekat

Radne okvire možemo podijeliti na dva dijela: frontend i backend

Frontend radni okviri odnose se na područje aplikacije ili web stranice s kojom korisnici direktno komuniciraju. Frontend web okviri smanjuju složenost programerima koji moraju ručno kreirati kod kako bi diktirali ponašanja i interakcije između korisnika i aplikacije/stranice.

Neki od najpopularnijih frontend radnih okvira koji se danas koriste su Angular, React i Vue.js.

Backend radni okviri su biblioteke programskih jezika na strani servera. Oni pomažu u izgradnji pozadinske konfiguracije web aplikacija. Backend web okviri pružaju alate koji pomažu u razvoju zadataka kao što su autorizacija korisnika, sigurnost, URL rutiranje i interakcija s bazom podataka. Kao is a frontend okvirima, korištenje ovih okvira olakšava programerima rad tako što se eliminira potreba za konfiguracijom i izgradnjom svega od nule.

Neki od najpopularnijih backend okvira koji se danas koriste su Laravel, Django, Spring i .NET

Cilj je kreirati Internet forum web aplikaciju koristeći Angular i .NET. Forum će imati podforume (zajednice) u kojima će se nalaziti objave. Svi prijavljeni korisnici imati će

mogućnost kreiranja svoje zajednice, time automatski postaju administratori kreirane zajednice. Na kreiranoj zajednici objavu mogu kreirati svi prijavljeni korisnici ukoliko zajednica nije arhivirana. Korisnici koji nisu prijavljeni mogu samo čitati objave dok prijavljeni korisnici mogu i da komentarišu objave. Ovaj način kreiranja foruma omogućava da razne zajednice koriste istu stranicu kako bi našli istomišljenike.

2. NET tehnologija

.NET okvir je razvojni ekosistem dizajniran da podrži kreiranje web aplikacija i softvera. Mnogi programeri preferiraju .NET u odnosu na slične platforme zahvaljujući njegovim korisnim karakteristikama i naprednim funkcijama. .NET okvir je razvio Microsoft koji radi prvenstveno na Microsoft Windows-u. To je bila dominantna implementacija zajedničke jezičke infrastrukture (CLI) sve dok nije zamijenjena .NET projektom za više platformi.

.NET je okvir koji se koristi za razvoj i kreiranje aplikacija kao što su:

- Konzolne aplikacije
- Web aplikacije
- Windows Forms aplikacije
- Web servisi i
- Aplikacije vođene događajima

Koristi se za kreiranje mnogo različitih vrsta aplikacija i usluga i pruža neophodno programsko okruženje za sve faze razvoja softvera. Njegov okolni ekosistem odlično se uklapa za kompanije i pojedince koji žele razviti širok spektar softverskih proizvoda, od desktop i mobilnih aplikacija do web aplikacija i cloud servisa.

Karakteristike .NET-a:

1. **CLR – Common Language Runtime:** Svi programski jezici u .NET-u su kompajlirani u srednji jezik poznat kao Common Intermediate Language (CIL). Ovaj međujezik se ne interpretira, već se kompajlira u izvorni kod na način poznat kao kompilacija Just In Time (JIT). Kombinacija njih se zove Common Language Infrastructure (CLI).
2. **Interoperabilnost jezika:** Često je potrebna interakcija između novih i starih aplikacija. .NET okvir pruža način pristupa funkcijama koje su implementirane u programima koji se pokreću izvan .NET okruženja.
3. **Prenosivost:** .NET okvir pruža zajedničku platformu za pisanje prenosivih programa koristeći različite jezike .NET okruženja, zasnovanu na otvorenim Internet standardima koji omogućavaju različitim uređajima, softverima i aplikacijama da rade zajedno u širokoj i disperziranoj mreži mreža.
4. **Sigurnost tipova:** Tokom izvršavanja programa, provjeravač tipa osigurava da svi objekti i vrijednosti, kao i reference na te objekte i vrijednosti, imaju važeći tip. Na

primjer, provjeravač tipa osigurava da je samo cjelobrojna vrijednost dodijeljena cjelobrojnoj varijabli. Provjera tipa također osigurava da se samo valjane operacije izvode nad objektima ili vrijednostima.

5. **Performanse:** CLR okruženje promoviše bezbjedno izvršavanje koda, eliminiše uska grla u performansama pomoću garbage collectora, minimizira konflikte u razvoju softvera kroz prostore imena i izbegava konflikte u vezi sa verzijama kroz manifest asemblera.
6. **Bogata podrška za profilisanje i otklanjanje grešaka:** Integrisana razvojna okruženja (IDE) kao što su Visual Studio ili Visual Studio Code pružaju mjesto za razvoj i otklanjanje grešaka u .NET aplikacijama. Okvir pruža bogate informacije za otklanjanje grešaka i profilisanje koje su visoko integrisane unutar ovih IDE-ova. Ovo nam pomaže da provjerimo performanse aplikacije i pronađemo iznimke vremena izvršavanja.
7. **Automatsko upravljanje resursima:** .NET CLR automatski upravlja memorijom, mrežom, vezama baze podataka, itd. Poziva ugrađene funkcije za dodjelu i oslobađanje memorije koju koriste objekti tokom vremena izvršavanja. Ovo uklanja teret upravljanja memorijom sa programera.[1]

Prije .NET 6, koristile su se dvije različite .NET verzije: .NET Framework i .NET Core. .NET Framework je bio originalni .NET koji nije bio open source (Korišten za Windows aplikacije, bogatije funkcijama u poređenju sa .NET Core, ali su zato sporije i rade samo na Windows operativnom sistemu). .NET Core je bila open source verzija .NET-a koja je prepisana od nule i radi na svim platformama (Windows, Linux i Mac). U novembru 2021. godine objavljena je .NET 6 verzija koja spaja .NET Core i .NET Framework.

Sa .NET 6.0 i budućim verzijama, postojaće samo jedna verzija .NET-a i to samo .NET.

2.1. NET 6

.NET 6 je prava objedinjena razvojna platforma koja omogućava programerima da grade aplikacije za cloud, web, desktop, mobilne uređaje, igre, IoT i AI. Sve ove podplatforme dijele zajedničke biblioteke osnovnih klasa, API-je i osnovnu infrastrukturu uključujući jezik i kompajlere.

[1] https://www.geeksforgeeks.org/characteristics-of-net-framework/

.NET 6 je postao dostupan 9. novembra 2021., zajedno sa C# 10 i Visual Studio 2022. Uz sveukupno poboljšano iskustvo programera, .NET 6 donosi značajno poboljšanje performansi u poređenju sa starijim verzijama. Tačnije, .NET 6 je najbrži .NET do sada. NET 6 pruža jedinstvenu platformu za desktop, mobilne, web, cloud i IoT aplikacije. Osnovna platforma podržava aplikacije različitih tipova i jednostavno omogućava ponovno korištenje koda u svim aplikacijama. Uvedene su nove funkcije i poboljšanja kako bi se osiguralo da se kod koji se izvršava u cloudu, na desktopu ili na mobilnom uređaju ponaša na isti način.

Jedno od poboljšanja je .NET Multi-platform App UI (.NET MAUI). Sada je moguće pisati kod koji pruža moderno iskustvo klijentske aplikacije na desktop i mobilnim operativnim sistemima u jednom projektu.

Hot Reload je jedna od najupečatljivijih novih funkcija dodatih u .NET 6. Hot Reload je funkcija koja omogućava izmjenu izvornog koda aplikacije i trenutna primjena tih promjena na pokrenutu aplikaciju. Svrha ove funkcije je povećati produktivnost izbjegavanjem ponovnog pokretanja aplikacije između uređivanja. Hot Reload je dostupan u Visual Studio 2022 i dotnet watch alatu komandne linije.[2]

Kao što je već ranije spomenuto, za aplikacijski sloj koristiti će se C#. Spominje se .NET 6 zato što je backend projekta izgrađen u .NET 6 okviru koristeći MVC arhitekturu kako bi pristupili bazi podataka. Konekcija sa bazom vrši se preko connection stringa u appsettings JSON fajlu koji se povezuje na šemu InternetForum. Da bi pristupili bazi podataka potrebno je prvo objasniti repozitorijsko-servisni obrazac *(eng. Repository-Service pattern)* i kontrolere.

2.2. Repozitorijumi i servisi

Obrazac Repository-Service razbija poslovni sloj aplikacije u dva različita sloja.

Donji sloj su Repozitorijumi. Ove klase upravljaju unosom i čitanjem podataka u i iz naše baze podataka, uz važnu stavku a to je da svaki model ima svoj repozitorij. Dakle, u slučaju Internet foruma potrebno je napraviti 13 modela: Community, CommunityType, Post, PostComment,

[2] https://www.c-sharpcorner.com/article/what-is-new-in-net-6-0/

PostTag, PostVote, RefreshToken, Role, Tag, User, UserCommunity, UserPostVote i UserStatus. Pošto svaki model ima svoj repozitorij, repozitoriji će imati imena kao što su: CommunityRepo, PostRepo, UserRepo i tako dalje. i neće međusobno nikakve metode pozivati, odnosno, CommunityRepo ne bi pozivao ništa u PostRepo, i tako dalje.

Gornji sloj su servisi. Ove klase će imati Repozitorije injektirane u njih i mogu koristiti više klasa Repozitorija i kombinovati njihove podatke da formiraju nove, složenije poslovne objekte. Nadalje, pored interfejsa koji svaki servis i repozitorij ima, servisi uvode sloj apstrakcije između web aplikacije i Repozitorija tako da se mogu neovisnije mijenjati.

Obrazac Repository-Service se oslanja na injekciju ovisnosti (Dependency-Injection, skraćeno DI) da bi ispravno radio. Klase na svakom sloju arhitekture će imati klase koje su im potrebne iz "nižih" slojeva injektirane u njih.

2.3. MVC kontroleri

MVC kontroleri su odgovorni za odgovaranje na zahtjeve upućene .NET MVC web stranici. Svaki zahtjev internet preglednika je mapiran na određeni kontroler. Na primjer, unesen je sljedeći URL u adresnu traku pretraživača:

http://localhost/Post/1/

U ovom slučaju se poziva kontroler pod nazivom PostController. PostController je odgovoran za generiranje odgovora na zahtjev pretraživača. Na primjer, kontroler može vratiti određeni view ili preusmjeriti korisnika na drugi kontroler. Svaki MVC kontroler u svom nazivu na kraju mora imati Controller, kao što su: UserController, CommunityController i tako dalje.
Da bi pristupili bazi podataka, u kontroleru se nalaze HTTP get, post, put i delete metode. U svakoj od tih metoda pozivaju se potrebne metode iz servisa pomoću interfejsa tog servisa.

Metode iz servisa pozivaju metode iz interfejsa repozitorija koje u sebi sadže CRUD metode za korisnike, zajednice i objave. Da bi obični korisnici i admini pristupili CRUD metodama potrebno je da se prvo prijave. Obični korisnici imaju pristup kreiranju, uređivaju i brisanju vlasititih zajednica, objava ili komentara. Postoje dvije vrste uloga. Uloge generalnog Foruma i uloge zajednice. Admin ima pristup kreiranju, uređivanju i brisanju svih zajednica, postova i korisnika dok admin koji ima pristup kreiranju, uređivanju i brisanju samo pojedinih zajednica

i objava se zove CommunityAdmin, CommunityAdmin uloga i ostale uloge neke zajednice se pohranjuju u tabelu UserCommunities. Da bi se uspješno prijavili potrebno je kreirati AuthController, više o tome kasnije.

3. Angular

Angular je besplatni open-source okvir za web aplikacije koji je izgrađen u TypeScript-u. Njegova primarna svrha je razvoj single-page aplikacija. Kao okvir, Angular ima jasne prednosti, a istovremeno pruža standardnu strukturu za programere za rad. Omogućava korisnicima da kreiraju velike aplikacije na način koji se može održavati.

Kao platforma, Angular uključuje:

- Okvir za izgradnju skalabilnih web aplikacija pomoću komponenti,
- Kolekcija dobro integriranih biblioteka koje pokrivaju širok raspon funkcija, uključujući rutiranje, upravljanje formama, klijent-server komunikacija, i još mnogo toga
- Paket programskih alata koji će vam pomoći da razvijete, izgradite, testirate i ažurirate svoj kod

Okviri općenito povećavaju učinkovitost i performanse web razvoja pružajući konzistentnu strukturu tako da programeri ne moraju da pišu kod od nule. Okviri štede vrijeme i nudi programerima mnoštvo dodatnih funkcija koje se mogu dodati aplikaciji bez dodatnog napora.

Karakteristike Angular-a:

1. **Efikasna MVC arhitektura:** Model-View-Controller (MVC) arhitektura je među vodećim karakteristikama Angular-a. MVC povećava vrijednost okvira za razvoj aplikacija na strani klijenta i brine o drugim funkcijama kao što su povezivanje podataka i opseg. U poređenju s drugim okvirima, MVC spaja sve potrebne elemente aplikacije kako bi se izbjegao dodatno kodiranje.
2. **Razdvajanje koda:** Najbolji dio Angular okvira je što pomaže u organiziranju koda u različite module. Zbog ove karakteristike, postoji podjela funkcionalnosti na višekratni kod. Razdvajanje koda pomaže u podjeli zadataka među Angular programerima i dozvoljava web aplikacijama da ostvare lazy loading.
3. **Dvosmjerno povezivanje podataka *(eng. Two-Way Data Binding)*:** Povezivanje podataka je tehnika u kojoj podaci ostaju sinhronizovani između komponente i pogleda. Kad god korisnik ažurira podatke u prikazu, Angular ažurira komponentu. Kada komponenta dobije nove podatke, Angular ažurira prikaz.

4. **TypeScript:** TypeScript je superscript JavaScript-a, koji pomaže korisnicima da pišu JavaScript kod koji je lakši za razumijevanje. Sav TypeScript kod se kompilira sa JavaScript-om i može nesmetano da radi na bilo kojoj platformi. TypeScript nije obavezan za razvoj Angular aplikacije. Međutim, toplo se preporučuje jer nudi bolju sintaksičku strukturu — dok čini bazu koda lakšom za razumijevanje i održavanje.
5. **Rutiranje:** U single-page aplikaciji, mijenja se ono što korisnik vidi tako što se prikažu ili sakriju dijelovi ekrana koji odgovaraju određenim komponentama, umjesto da se ide na server da se dobije nova stranica. Da bi se prikazali ili sakrili dijelovi ekrana koji odgovaraju određenim komponentama, koriste se pogledi *(eng. views)*. Za rukovanje navigacijom od jednog pogleda do drugog, koristi se Angular Routing.[3]

3.1. Komponente i servisi

Komponente služe za upravljanje templejtima (HTML, CSS i TS) i servisima koje sadrže poslovnu logiku aplikacije. Ovaj modularni pristup omogućava brz i logičan razvoj aplikacija i olakšava njihovo testiranje. U pravilno dizajniranoj Angular aplikaciji, različiti dijelovi aplikacije ne bi trebali biti svjesni detalja implementacije drugih dijelova.

Servisi su još jedan dio koji je potreban za izgradnju Angular aplikacije. Funkcija servisa je pružanje specifične funkcionalnosti koja se ne bi uklapala u klasu komponenti, kao što je logika za dohvaćanje podataka sa servera ili logika za evidentiranje poruka u konzoli browsera. Servisi jačaju Angularovu ideologiju o modularnosti. Komponente nisu namijenjene da rade ništa drugo nego da omoguće interakciju između korisnika i logike aplikacije. Iako ništa ne sprječava programera da implementira aplikaciju bez servisa, to se ne preporučuje.

Komponenta kontroliše dio ekrana koji se zove pogled *(eng. view)*. Kao što je ranije pomenuto komponenta se sastoji od TypeScript klase, HTML-a i CSS-a. TypeScript klasa definiše interakciju HTML templejta i DOM strukture, dok CSS opisuje izgled.

3.2. Pogled (View)

View je najmanja grupa elemenata koji se mogu stvoriti i uništiti zajedno. Angular prikazuje pogled pod kontrolom jedne ili više direktiva. Klasa komponente i njen templejt definišu view.

[3] https://www.cybersuccess.biz/angular-features/

Pogled je predstavljen instancom ViewRef koja je povezana s komponentom. Pogled koji neposredno pripada komponenti se referencira kao host view. Pogledi se obično prikupljaju u hijerarhije pogleda.

Svojstva elemenata u pogledu mogu se dinamički mijenjati pomoću neke radnje korisnika; dok se struktura elemenata (broj i redoslijed elemenata) u pogledu ne može mijenjati. Moguće je promijeniti strukturu elemenata umetanjem, pomicanjem ili uklanjanjem ugniježđenih pogleda unutar njihovih kontejnera pogleda.

Hijerarhije pregleda mogu se dinamički učitavati i isprazniti dok se korisnik kreće kroz aplikaciju, obično pod kontrolom rutera.[4]

3.3. Ruter i rutiranje

Ruter je alat koji konfiguriše i implementira navigaciju između stanja i pogleda unutar Angular aplikacije.

Modul rutera je NgModule koji obezbjeđuje potrebne dobavljače servisa i direktive za navigaciju kroz poglede aplikacije. Komponenta za usmjeravanje je ona koja uvozi modul rutera i čiji predložak sadrži element RouterOutlet gdje može prikazati poglede koje proizvodi ruter.

Ruter definira navigaciju između prikaza na jednoj stranici, za razliku od navigacije između stranica. On tumači veze slične URL-u kako bi odredio koje poglede kreirati ili uništiti i koje komponente učitati ili isprazniti. Omogućava da se iskoristite prednosti lazy loading-a u Angular aplikaciji.[5]

3.4. HTTP i RxJS biblioteka

Većina frontend aplikacija mora da komunicira sa nekim serverom preko HTTP protokola zbog preuzimanja ili slanja podataka i pristupa drugim back-end servisima. Angular pruža HTTP API za Angular aplikacije (HttpClient). HTTP biblioteka je uključena u vlastiti modul koji se mora importovati prije nego što se HTTP servis može koristiti u bilo kojoj komponenti ili

[4] https://angular.io/guide/glossary#view
[5] https://angular.io/guide/glossary#router

servisu. Sam servis podržava sve HTTP zahtjeve, a najčešći su get i post. HTTP biblioteka omogućava izradu asinhronih zahtjeva, koji neće zaključati aplikaciju dok radi. Ovo rezultira ugodnijim korisničkim iskustvom dok aplikacija preuzima podatke sa servera, ili šalje podatke.[6]

Jedan dio HTTP biblioteke su RxJS observables. RxJS je biblioteka za sastavljanje asinhronih i event-based programa korištenjem observable sekvenci.
Osnovni koncepti u RxJS-u koji rješavaju asinhronizirano upravljanje događajima su:

- **Observable:** predstavlja ideju o kolekciji budućih vrijednosti ili događaja.
- **Observer:** je kolekcija callback-a koja zna kako da sluša vrijednosti koje isporučuje Observable.
- **Subscription:** predstavlja izvršenje Observable-a, prvenstveno je korisna za otkazivanje izvršenja.
- **Operators:** su funkcije koje omogućavaju rad sa kolekcijama sa operacijama kao što su *map, filter, concat, reduce* itd.
- **Subject:** je ekvivalentan EventEmitter-u i jedini način da se izvrši multicasting odnosno da se neka vrijednost ili događaj pošalje višestrukim Observerima.
- **Schedulers:** su centralizovani dispečeri za kontrolu istovremenosti, omogućavajući nam da koordiniramo kada da se neka funkcija izvrši ili kada da se nešto desi sa npr. *setTimeout* ili *requestAnimationFrame* itd.[7]

3.5. Pipe

Pipe-ovi se koriste za transformaciju stringova, iznosa valuta, datuma i drugih podataka za prikaz. Pipe-ovi su jednostavne funkcije koje se koriste u templejtima za prihvaćanje ulazne vrijednosti i vraćanje transformirane vrijednosti. Pipe-ovi su korisni jer se mogu koristiti u cijeloj aplikaciji, dok je svaki pipe potrebno samo jednom deklarisati. Na primjer, u ovom slučaju jedan od pipe-ova koji će se koristiti je DatePipe koji služi da se prikaže datum kao npr. 30. Juni 2022. umjesto sirovog formata datuma (kao npr. 2022-06-30 22:45:52.9379969).

[6] https://angular.io/guide/http
[7] https://rxjs.dev/guide/overview

Za neke dijelove aplikacije (kao što je postovi i komentari) potrebno je prikazati datum kao npr 5 hours ago. Da bi se datum prikazao na ovaj način potrebno je koristiti ngx timeago pipe. Moguće je također koristiti više pipe-ova kako bi se dobila željena transformirana vrijednost. Timeago pipe prihvata DatePipe date format, a pošto se sirovi format datuma ne može koristiti sa timeago pipe-om, upravo u ovom slučaju može se koristiti date i timeago pipe u isto vrijeme na način koji je prikazan na slici 3.1 (prikazan je primjer iz post-details komponente):

```
<div>
    <button mat-button disabled>
        {{ post.viewsCount | shortNumber}} Views
    </button>
    <button mat-button disabled>
        {{ post.dateCreated | date:'MMM d, y, h:mm a' | timeago}}
    </button>
</div>
```

Slika 3.1 shortNumber, date i timeago pipe

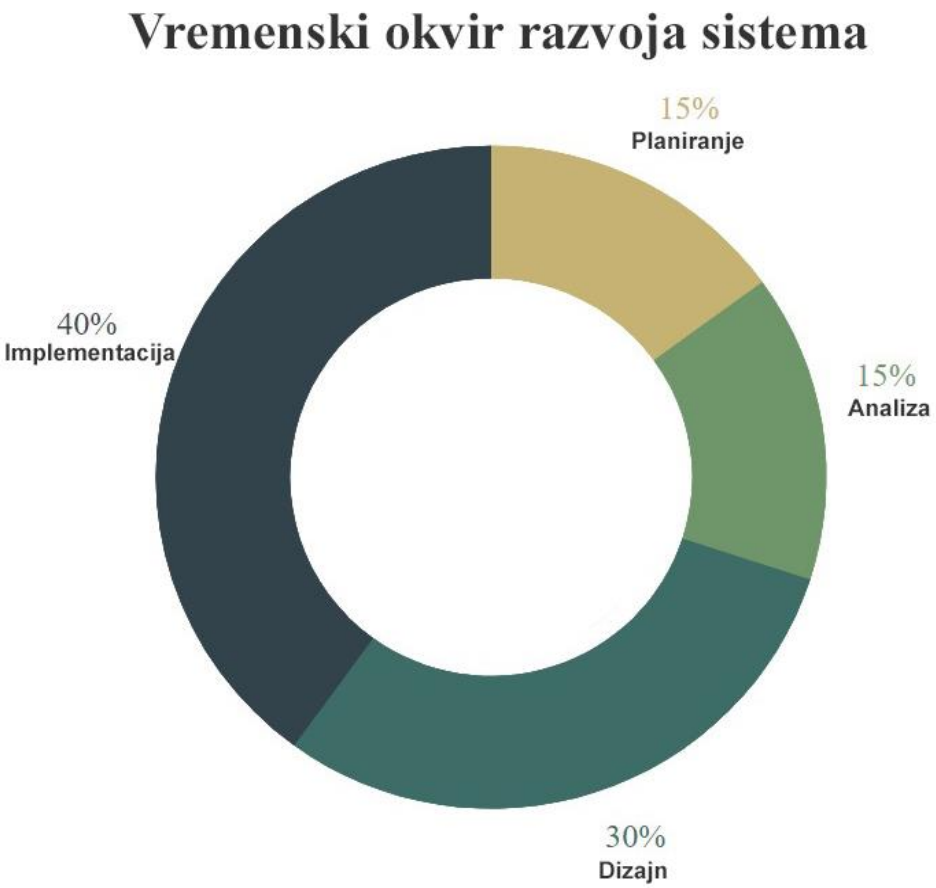
Na prethodnoj slici može se vidjeti da se koriste tri pipe-a, date i timeago su pipe-ovi koji već postoje dok je shortNumber custom pipe. Custom pipe se može generisati pomoću komande ng generate pipe, pipeovi su smješteni u pipes folderu pa će komanda biti ng generate pipe pipes/short-number. Zašto je potreban shortNumber pipe i šta shortNumber pipe ustvari radi: Svaki post ima broj pregleda, iz baze podataka dobija se sirovi number format za preglede nekog posta. Recimo da post ima 13 346 pregleda, da bi se poboljšala čitljivost umjesto da se prikazuju puni brojevi, postoji potreba za prikazom skraćenih brojeva. U ovom slučaju kada se na post.viewsCount iskoristi shortNumber pipe, rezultat će biti 13.3k views.[8]

[8] https://www.npmjs.com/package/ngx-timeago

4. Razvoj ogledne web aplikacije

4.1. Vremenski okvir

U vidu realizacije projekta potrebno je izvršiti procjenu vremenskog okvira projekta, u ovom slučaju koristiti ćemo se SDLC (Systems development life cycle) metodologijom. Vrijedi napomenuti da je procjena izvršena ručno. Procijenjeno je da će se uložiti oko 10% vremena u fazu planiranja, 20% vremena u fazu analize, 30% vremena u fazu dizajna i 40% vremena u fazu implementacije. Na slici 4.1 prikazana je procjena vremenskog okvira razvoja sistema



Slika 4.1 Vremenski okvir razvoja sistema

4.2. Arhitektura sistema

Da bi pristupili internet forumu, potrebno je koristiti internet browser na klijentskom računaru ili mobilnom uređaju. Budući da za pristup internet forumu koristimo internetski preglednik, ovaj sistem nazivamo web baziranim sistemom. Kako bi se balansiralo procesiranje između klijentskog uređaja i servera, koristi se klijent-server arhitektura sistema.

Klijent je odgovoran za prezentacijsku logiku, a server je odgovoran za logiku pristupa podacima i pohranu podataka. Pošto je u pitanju web bazirani sistem, klijent browser izvodi prezentacionu logiku sa logikom aplikacije, dok server izvodi većinu logike aplikacije, logiku pristupa podacima i logiku pohrane podataka.

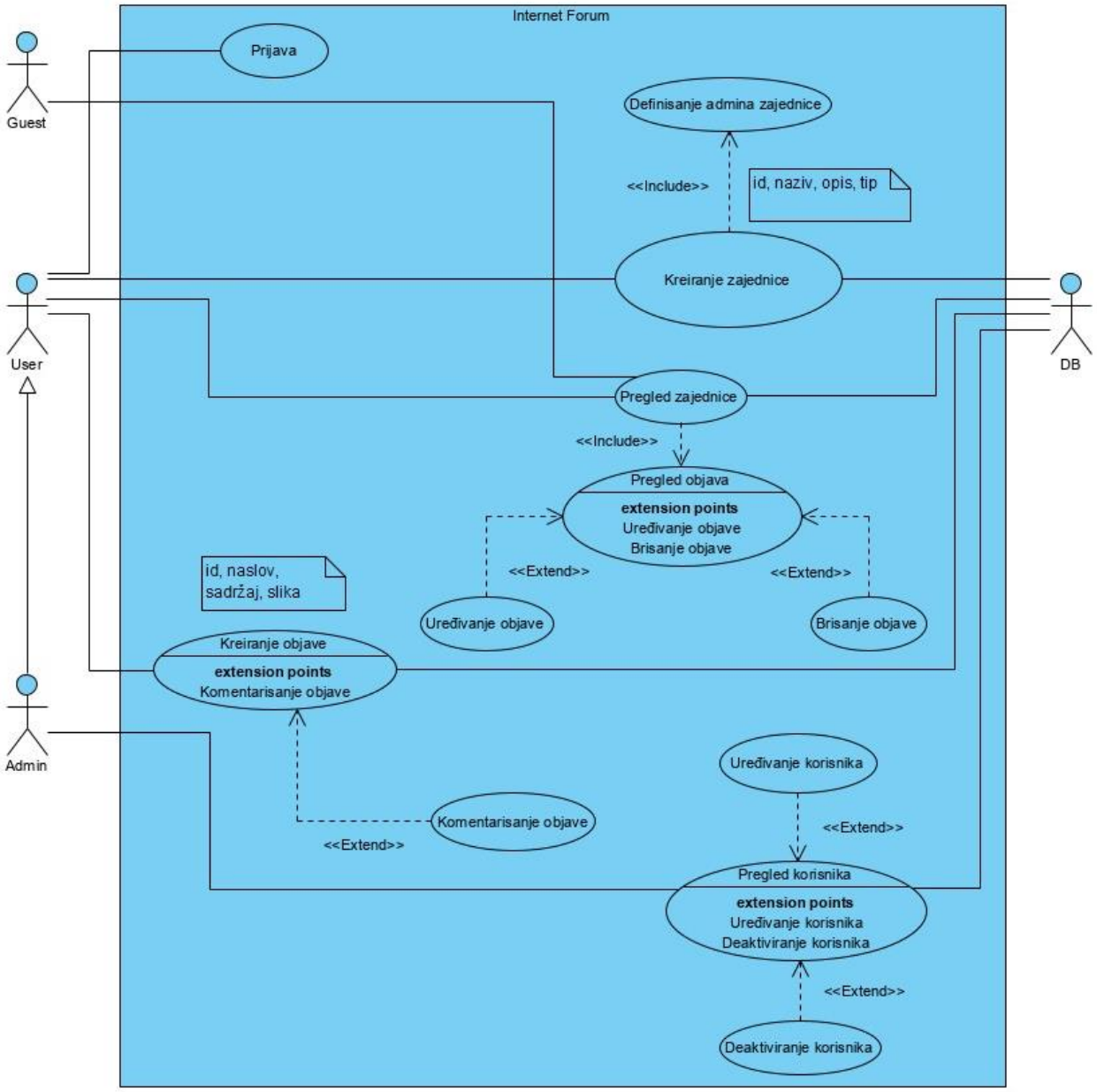
Slika 4.2 Troslojna arhitektura sistema internet foruma

Na slici 4.2 prikazana je troslojna arhitektura sistema, odnosno arhitektura sa tri seta računara. Na klijentskoj strani se prezentacijska logika izvodi pomoću HTML5, TypeScript i CSS, na serverskoj strani se izvodi logika aplikacije koja sadrži poslovnu logiku napisanu u C# a pored logike aplikacije izvodi se i logika pristupa i pohrane podataka koristeći Microsoft SQL Server na Windows operativnom sistemu.[9]

4.3. Analiza i dizajn

Analizu i dizajn internet foruma započet ćemo sa dijagramom slučajeva upotrebe (use case) koji je prikazan na slici 4.3, zatim ćemo preći na dijagram klasa i završiti sa sekvencijalnim dijagramom.

[9] https://www.ibm.com/cloud/learn/three-tier-architecture



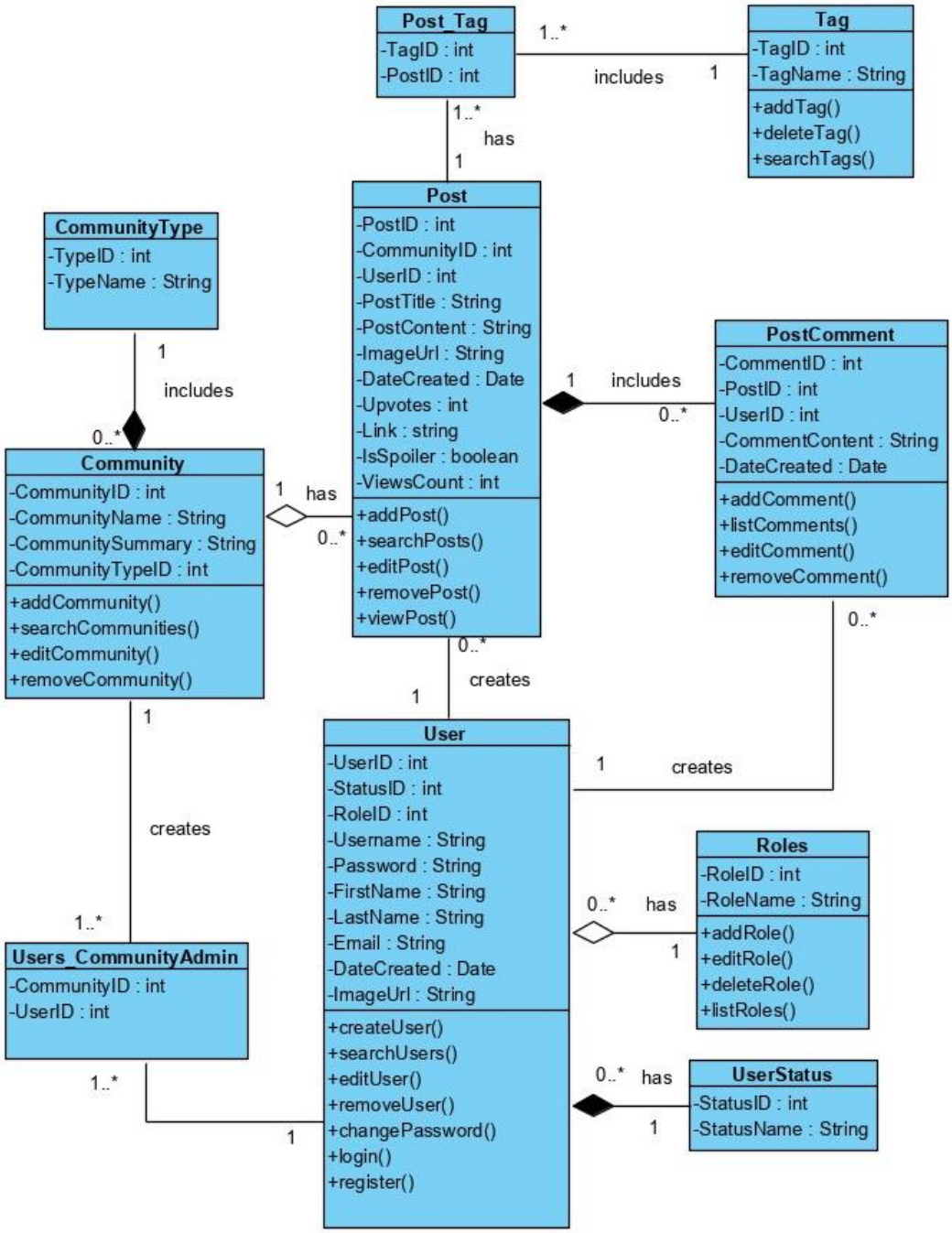
Slika 4.3 Dijagram slučajeva upotrebe internet foruma

Akteri u ovom dijagramu slučajeva su: Admin, User i DB. Admin i User su osobe a DB je baza podataka. Kada osoba pristupi web stranici može da se prijavi.

Ukoliko se osoba ne prijavi može nastaviti koristiti stranicu kao gost. Gosti imaju mogućnost pregleda zajednica, objava i komentara.

Da bi korisnik komentarisao objavu potrebno je da izvrši prijavu na web stranici. Ukoliko je prijava uspješna, pored mogućnosti komentarisanja objava, korisnik ima mogućnost kreiranja objava i kreiranja zajednica. Za kreiranje zajednice potrebno je unijeti naziv, opis i tip zajednice, a za kreiranje objave potrebno je unijeti naslov, sadržaj i sliku.

Kreatori objava i komentara imaju mogućnost uređivanja i brisanja svojih objava i komentara. Admin ima mogućnost brisanja i uređivanja svih objava, pored tog ima i mogućnost uređivanja i deaktiviranja korisnika.



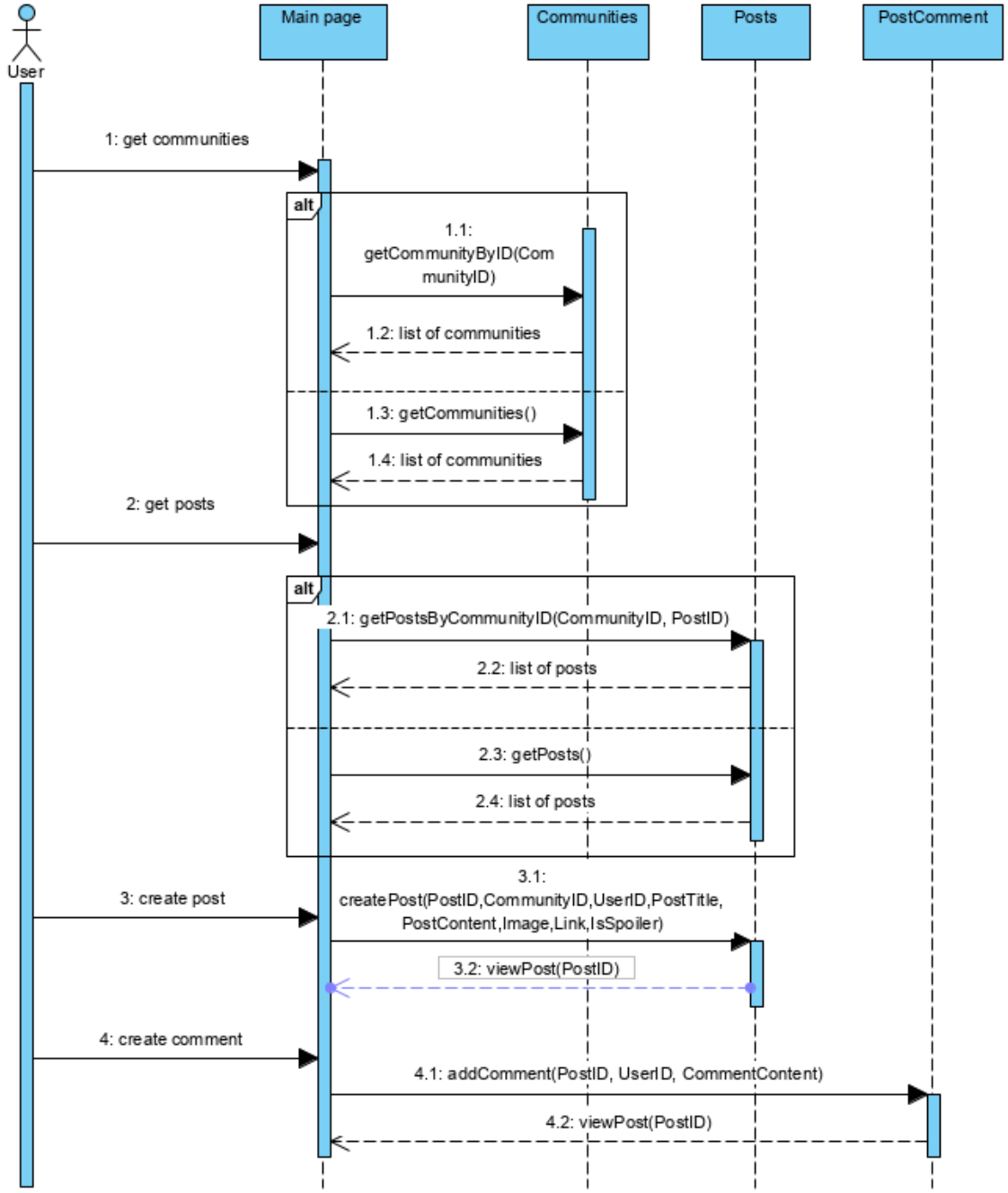
Slika 4.4 Dijagram klasa internet foruma (prije početka sa backendom)

Operacije u dijagramu klasa sa slike 4.4:

- User: kreiranje korisnika, pregled korisnika, uređivanje korisnika, brisanje korisnika, uređivanje šifre, login i registracija.
- Post: kreiranje objave, pregled objava, uređivanje objave, brisanje objave i pregled objave.
- Community: kreiranje zajednice, pregled zajednica, uređivanje zajednica i brisanje zajednica.
- PostComment: kreiranje komentara, prikaz komentara, uređivanje komentara i brisanje komentara.
- Tag: kreiranje tagova, brisanje tagova i pregled tagova.
- Roles: kreiranje uloge, uređivanje uloge, brisanje uloge i pregled liste uloga.

U ovom slučaju na slici 4.5 prikazan je dijagram sekvenci za pregled objava, pregled zajednica, kreiranja objava i kreiranja komentara. Za ova četiri slučaja potrebno je pet učesnika u dijagramu sekvenci, to su:

- Korisnik
- Glavna stranica
- Zajednice
- Objave
- Komentari objava



Slika 4.5 Dijagram sekvenci internet foruma

Kada korisnik pristupi stranici ima mogućnost pregleda zajednica (zajednica kojim se korisnik pridružio i ostalim zajednicama) preko sidebara, ukoliko se korisnik nije pridružio niti jednoj zajednici, prikazati će se sve zajednice ali neće biti sekcije gdje imaju korisnikove zajednice. Pregled objava može se izvršiti kada korisnik odabere neku zajednicu ili kada se korisnik vrati na početnu stranicu, tada se automatski prikažu sve aktuelne objave iz svih zajednica.

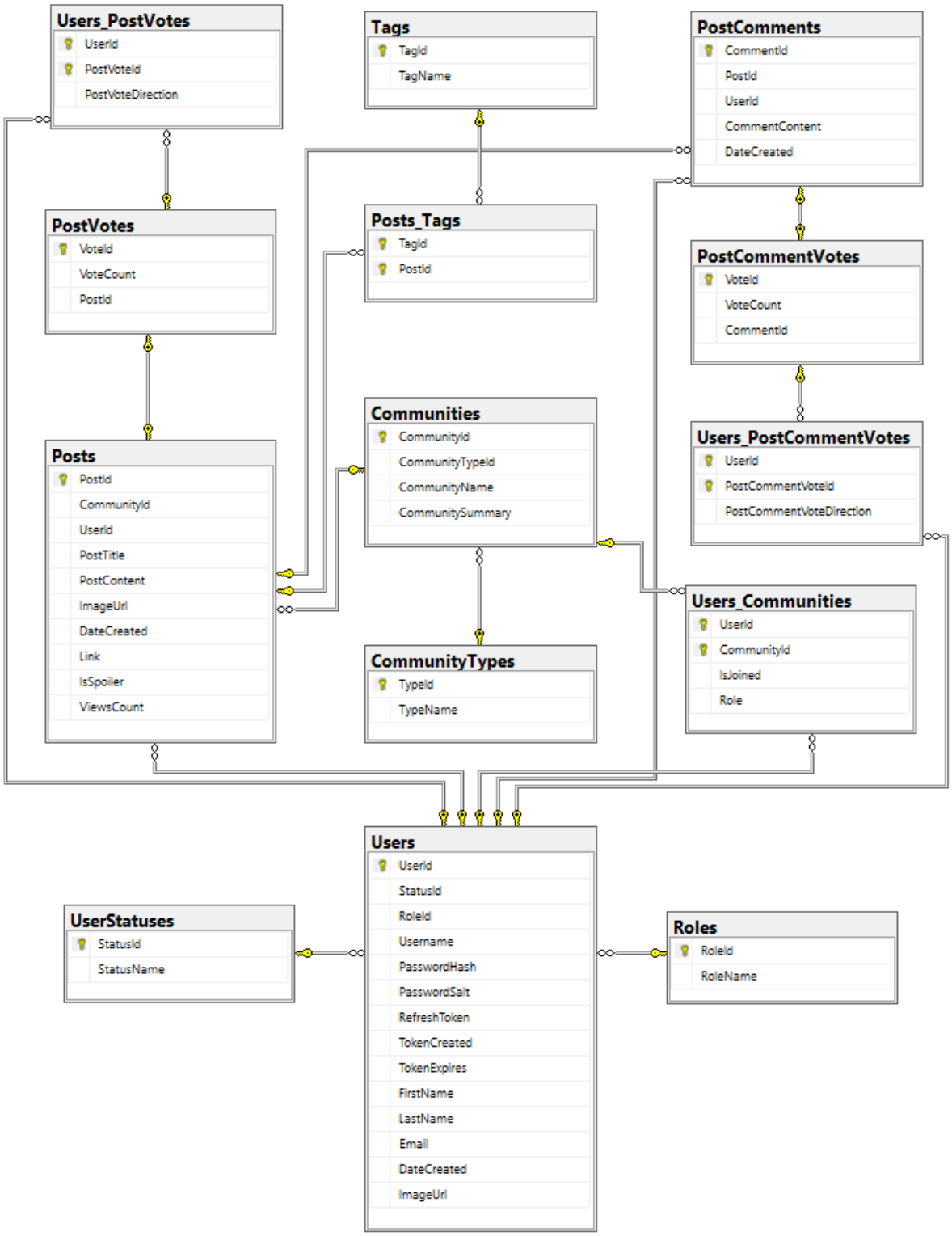
Zatim na sekvencijalnom dijagramu vidimo način kreiranja objave gdje korisnik unosi naslov, sadržaj, sliku, link i boolean vrijednost koja govori da li je objava spojler. Kada korisnik klikne na dugme da kreira objavu, pored korisnički definisanih varijabli, prosljeđuje se ID zajednice i ID korisnika koji je napravio objavu. Kada se kreira objava korisnik se preusmjeri na pregled kreirane objave. Da bi se kreirao komentar potrebno je da se korisnik nalazi na objavi koju želi komentarisati, zatim korisnik unosi tekst komentara, što je jedina stvar koju mora da unese, a kada klikne na dugme za kreiranje komentara automatski se prosljeđuje ID objave i ID korisnika. Na kraju korisniku se prikazuje kreirani komentar na objavi.[10]

4.4. Baza podataka

Baza podataka ima trinaest tabela što možemo vidjeti na ER dijagramu sa slike 4.6. Kreiranje korisnika vrši se pomoću tri tabele: Users, UserStatuses i Roles.

- Users: Sadrži ID korisnika, ID status, ID uloge, korisničko ime, šifru, ime, prezime, email, datum kreiranja i link slike korisnika.
- UserStatuses: Sadrži status korisnika (active, inactive, locked, banned, expired).
- Roles: Sadrži uloge korisnika internet foruma.

[10] Alan Dennis, Barbara Wixom, Roberta M. Roth: *Systems Analysis and Design, 5th Edition*



Slika 4.6 ER Dijagram Internet foruma

Prije kreiranja objava potrebno je da postoji zajednica u kojoj će se nalaziti objava. Kreiranje zajednice vrši se pomoću četiri tabele: Communities, Users_Communities, Users, i CommunityTypes.

- Community: sadrži ID zajednice, naziv zajednice, opis zajednice i ID tipa zajednice.
- Users_Communities: služi kao many-to-many veza između Users i Community kako bi se odredile uloge korisnika zajednice. Ova tabela najviše je potrebna zbog administratora zajednica. Moguće je imati više administratora u jednoj zajednici, a pored toga je moguće da je korisnik administrator u više zajednica.
- Users: Pri kreiranju stranice, korisnik koji kreira zajednicu automatski postaje administrator kreirane zajednice.
- CommunityTypes: Sadrži tipove zajednice (public, unlisted).

Ukoliko postoji barem jedna zajednica, moguće je napraviti objavu. Za kreiranje objave koriste se četiri tabele: Posts, PostVotes, Communities i Users.

- Posts: sadrži ID objave, ID zajednice, ID korisnika, naslov objave, sadržaj objave, link slike objave, datum kreiranja objave, broj sviđanja, link stranice (ukoliko prosljeđuje na neku drugu stranicu), boolean spojler (da li je objava spojler) i broj pregleda.
- PostVotes: sadrži broj sviđanja objave, ova tabela je povezana sa Posts i Users_PostVotes. Users_PostVotes sadrži sve korisnike koji su označili da im se sviđa ili ne sviđa neka objava *(upvote/downvote).*
- Communities: označava unutar koje zajednice se nalazi objava sa CommunityId.
- Users: Korisnik koji kreira objavu ima mogućnost brisanja i uređivanja kreiranje objave.

Na svaki post može se kreirati komentar. Za kreiranje komentara koriste se također četiri tabele: PostComments, Posts, PostCommentVotes i Users.

- PostComments: sadrži ID komentara, ID posta, ID korisnika, i komentar
- PostCommentVotes: sadrži broj sviđanja komentara, ova tabela je povezana sa PostComments i Users_PostCommentVotes. Users_PostCommentVotes sadrži sve korisnike koji su označili da im se sviđa ili ne sviđa neka objava *(upvote/downvote).*
- Posts: označava unutar koje objave se nalazi komentar sa PostId.
- Users: Korisnik koji kreira komentar ima mogućnost brisanja i uređivanja kreiranog komentara.

5. Front-end ogledne web aplikacije

5.1. Angular Material

Angular Material je biblioteka komponenti korisničkog interfejsa (UI) koju programeri mogu koristiti u svojim Angular projektima kako bi ubrzali razvoj konzistentnih korisničkih interfejsa. Angular Material nudi višekratnu upotrebu i UI komponente kao što su mat-card, mat-input, mat-button, mat-table, mat-form-field i jos mnogo drugih. Angular Material u kombinaciji sa Flex-Layout znatno ubrzava razvoj web aplikacije.

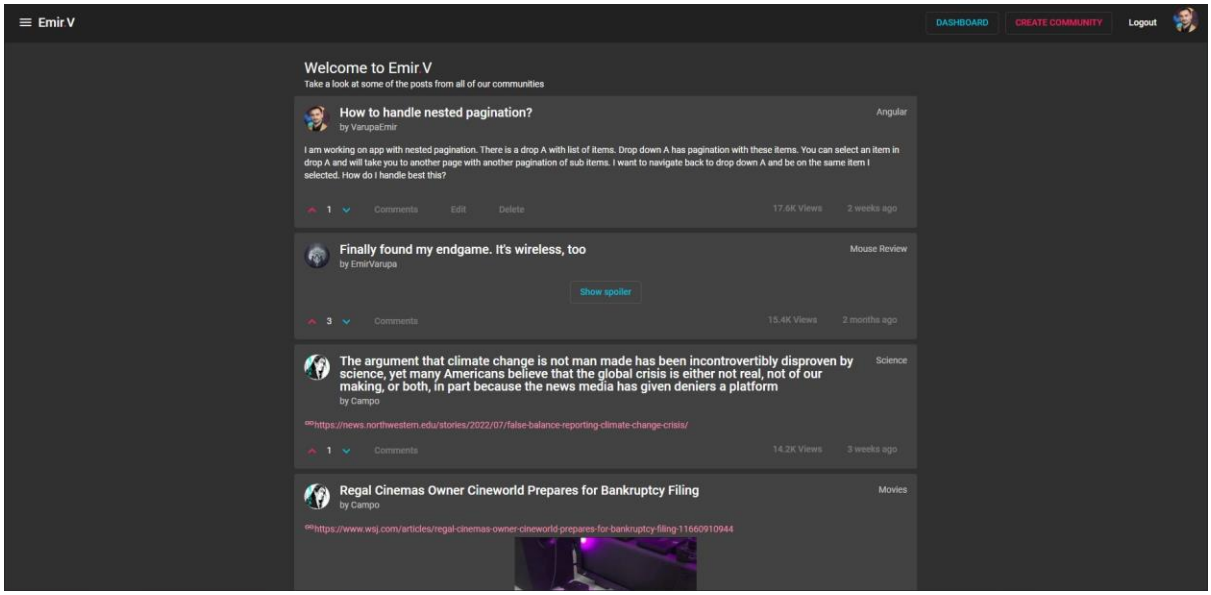
Svaka komponenta je spremna za korištenje sa zadanim stilom koji slijedi Material Design specifikaciju. Bez obzira na to, komponentama se može prilagoditi izgled i osjećaj. Lista dostupnih komponenti Angular Materiala raste sa svakom iteracijom biblioteke.

Nakon instaliranja Angular Materiala, moguće je odabrati temu koja definira koje će se boje koristiti u komponentama Angular Materiala. Pored unaprijed definisanih tema, moguće je kreirati prilagođenu temu. Potrebno je odabrati 3 palette boja za: primary, accent i warn. U ovom slučaju koriste se sljedeće palete boja: mat-pink (primary), mat-cyan (accent) i mat-red (warn).[11]

5.2. Pregled glavnih komponenti ogledne web aplikacije

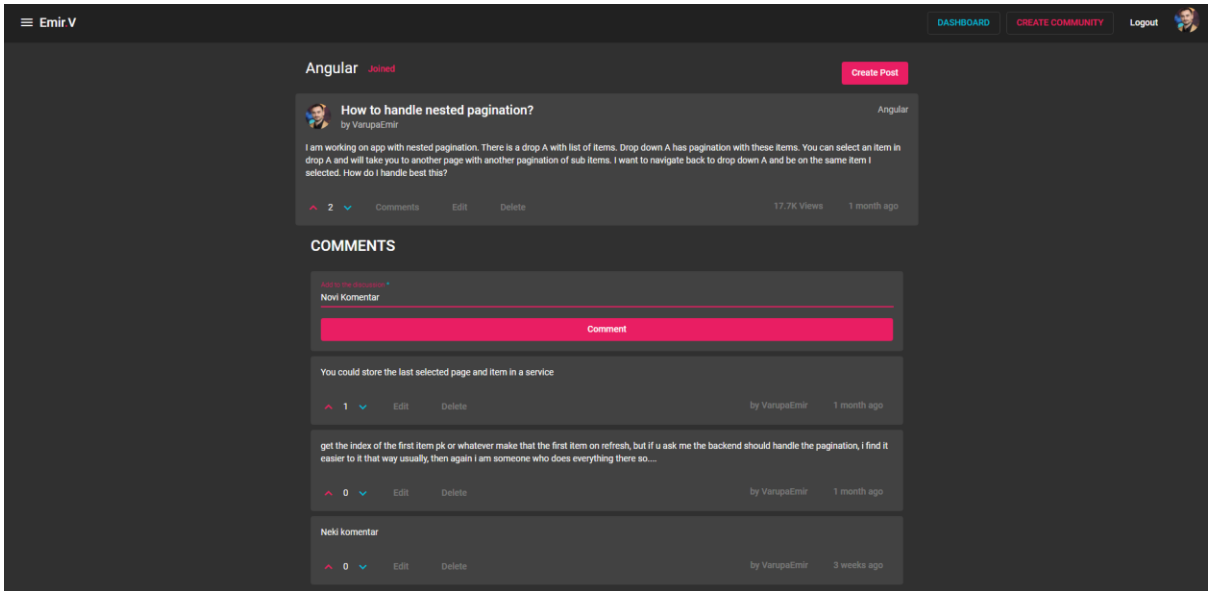
Sa slike 5.1 može se vidjeti da se početna stranica sastoji od headera i glavnog dijela stranice. Na lijevoj strani headera nalazi ime foruma, link na početnu stranicu (nalazi se na svakoj stranici) i link za prikaz zajednica. Na desnoj strani headera nalaze se informacije o korisniku i meni na kojem se nalaze dodatne opcije kao što je odjava, dashboard i slično. Prijavljeni korisnici imaju opciju kreiranja zajednice koja se također nalazi na desnoj strani headera. Ukoliko korisnik nije prijavljen, na desnoj strani će se umjesto informacija o korisniku i opcije kreiranja zajednice nalaziti opcija za prijavu i registraciju.

[11] https://auth0.com/blog/creating-beautiful-apps-with-angular-material/



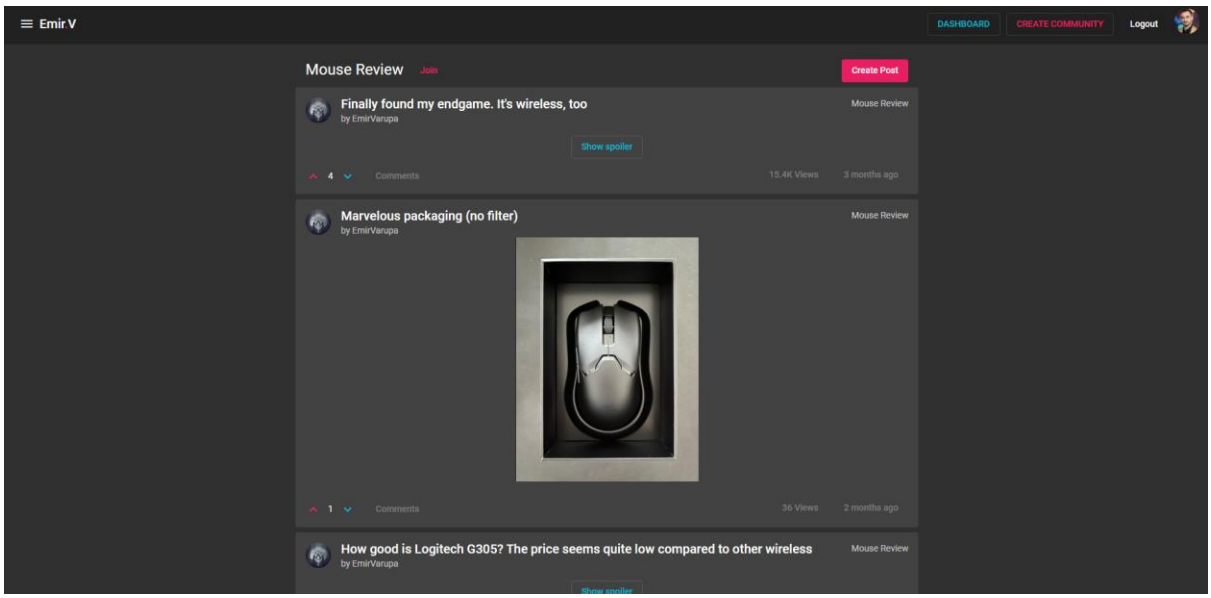
Slika 5.1 Početna stranica Internet foruma

Glavni dio stranice sastoji se od najnovijih i najpopularnijih objava iz svih zajednica (u sredini) i liste popularnih zajednica (na lijevoj strani). Na početku objave nalazi se informacija o korisniku koji je kreirao objavu i zajednica u kojoj se nalazi objava, zatim slijedi naslov objave i sadržaj objave, ukoliko sadržaj objave ima puno teksta, prikaže se samo dio teksta a ostatak se može vidjeti klikom na objavu. Sa lijeve strane sadržaja i naslova objave nalazi se broj sviđanja i opcija za “sviđa mi se” i “ne sviđa mi se” Ispod sadržaja objave nalaze se informacije o objavi i opcije objave, to su: broj komentara, opcija za dijeljenje objave, broj pregleda i vrijeme kreiranja objave. Pored ovih opcija administrator i korisnik koji je kreirao objavu ima opciju uređivanja i brisanja objave.



Slika 5.2 Stranica objave iz zajednice

Ukoliko korisnik želi da detaljno pregleda objavu, postavi komentar ili pregleda postojeće komentare objave, potrebno je da odabere objavu klikom na naslov ili sadržaj objave, što pomoću rutiranja vodi korisnika do post-details komponente kao što je prikazano na slici 5.2.



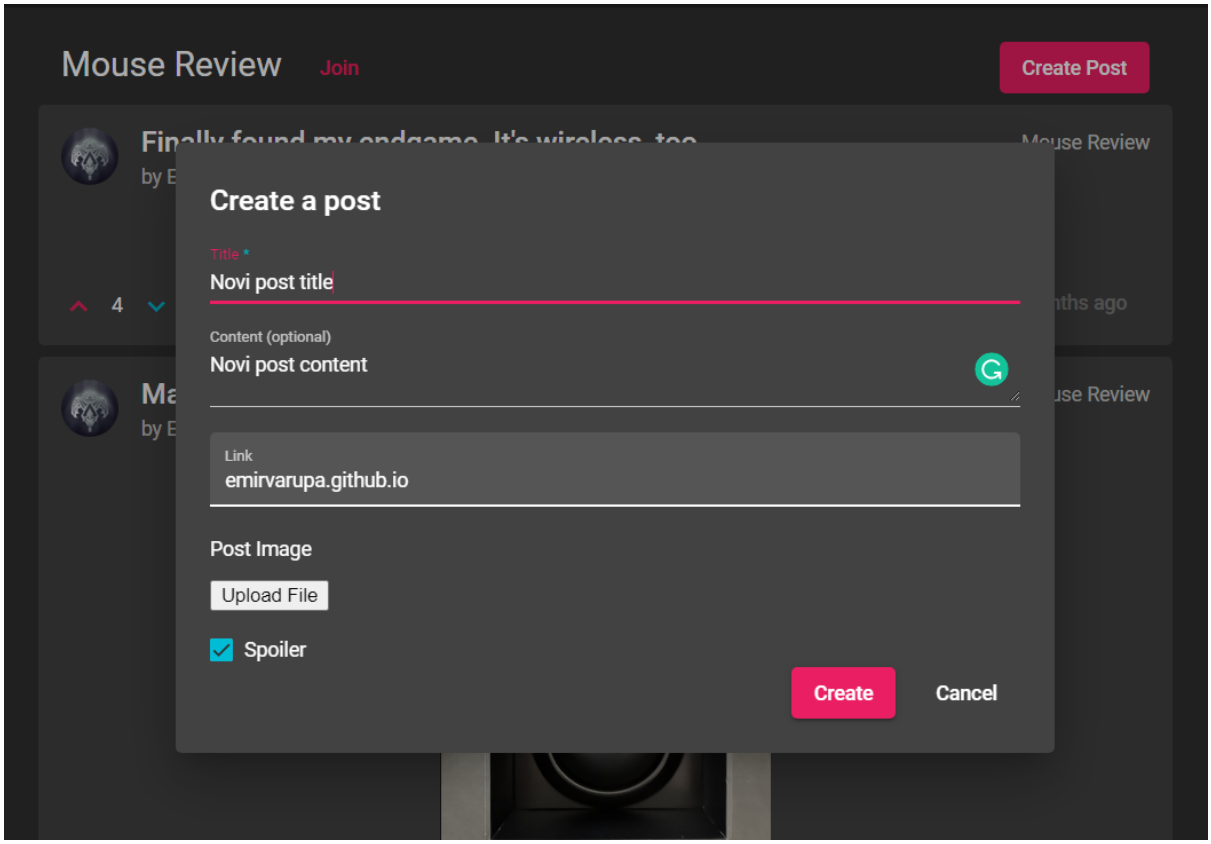
Slika 5.3 Stranica sa listom objava iz zajednice

Stranica zajednice sa slike 5.3 veoma je slična početnoj stranici ali postoji par razlika, ova stranica sastoji se od 3 dijela: headera, naziva i opisa zajednice i glavnog dijela stranice. Većina headera ostaje ista kao i na glavnoj stranici međutim sada je opcija za kreiranje zajednice

zamijenjena sa opcijom kreiranja objave u zajednici. glavni dio stranice sada samo prikazuje objave iz odabrane zajednice.

5.3. Funkcionalnosti ogledne web aplikacije

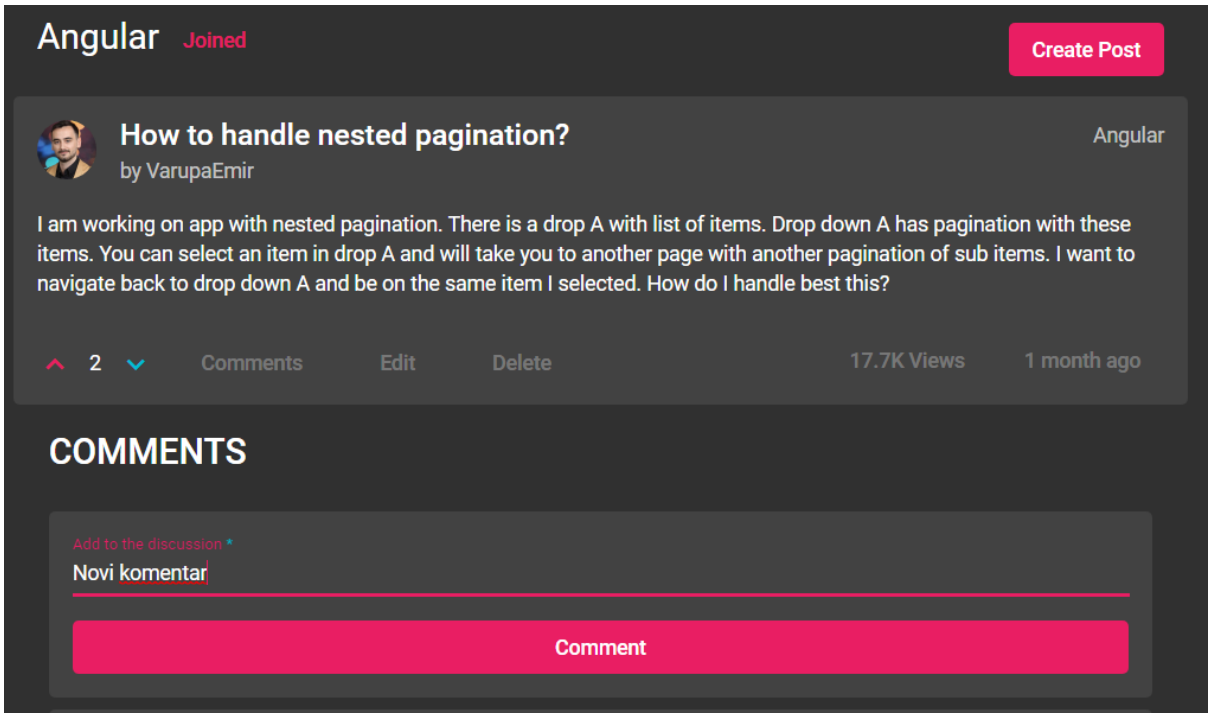
Kreiranje objave i zajednice vrši se pomoću forme koja se pojavi klikom na opciju za kreiranje objave ili zajednice. Nakon uspješnog kreiranja, korisnik dobije notifikaciju da je objava ili zajednica uspješno kreirana. Na slici 5.4 je prikazan dijalog koji se otvori klikom na button Create Post. Naslov je jedini dio koji je bitan da se upiše da bi se uspješno kreirala objava. Klikom na button Create, automatski se kreira se novi red u Posts i PostVotes tabelama. Pomoću PostVotes tabele prati se koji su sve korisnici označili da im se sviđa ili ne sviđa neka objava.



Slika 5.4 Kreiranje objave u zajednici

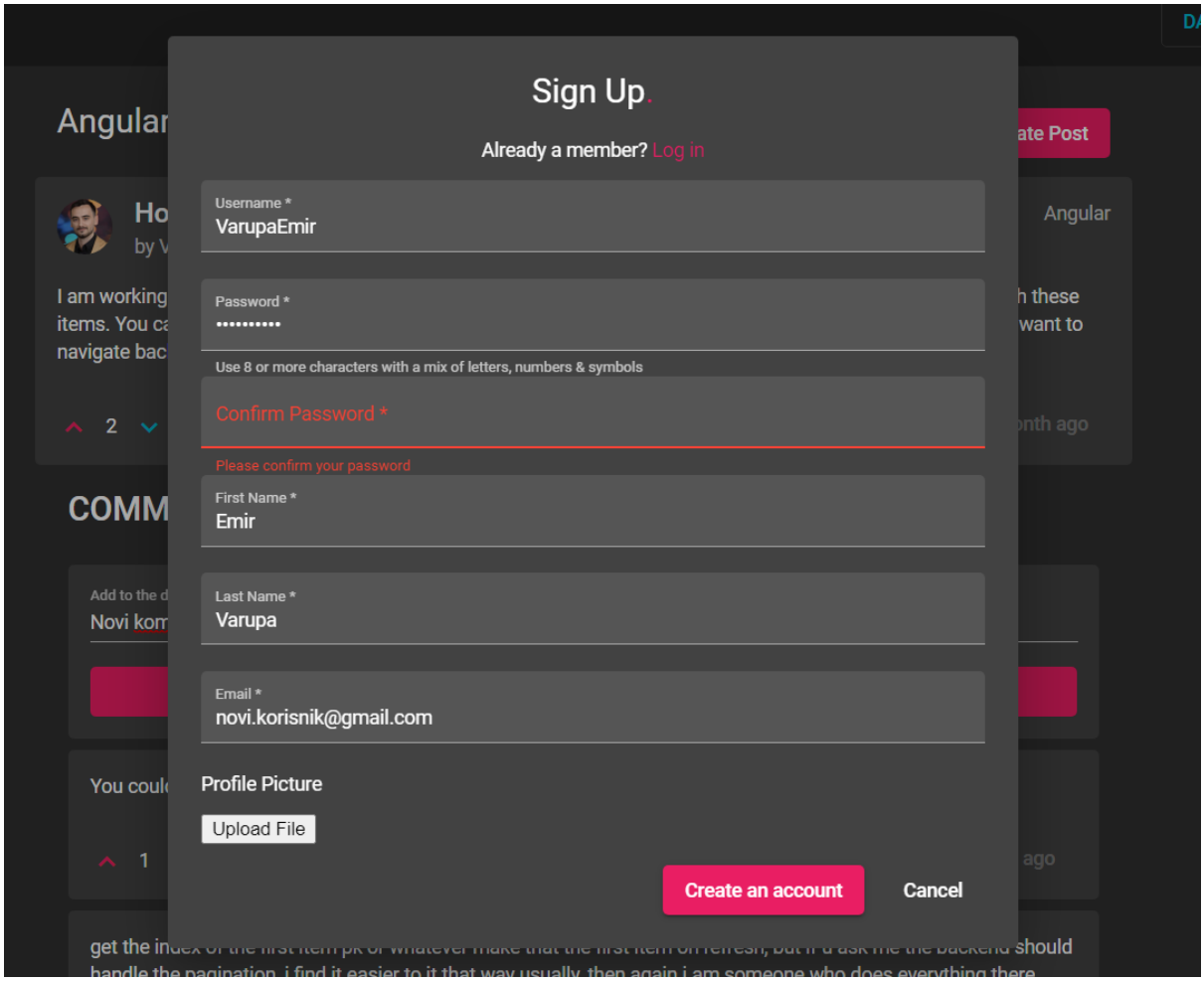
Ukoliko korisnik želi da postavi komentar, pregleda postojeće komentare objave ili da obriše svoj postojeći komentar, potrebno je da odabere objavu klikom na naslov ili sadržaj objave. Na post-details komponenti postoje dvjie sekcije: detalji objave i komentari objave. Na vrhu sekcije komentara korisnik može da postavi komentar ukoliko je korisnik izvršio login.

Kreiranje komentara se razlikuje od kreiranja objava, zajednica i korisnika po tome što se ne otvara dijalog klikom na neki button već se forma za kreiranje komentara nalazi u sekciji komentara kao što je prikazano na slici 5.5. Klikom na button Comment u bazi podataka se automatski kreira novi red u dvije tabele: PostComments i PostCommentVotes. PostCommentVotes se koristi na isti način kao što je slučaj kod PostVotes.



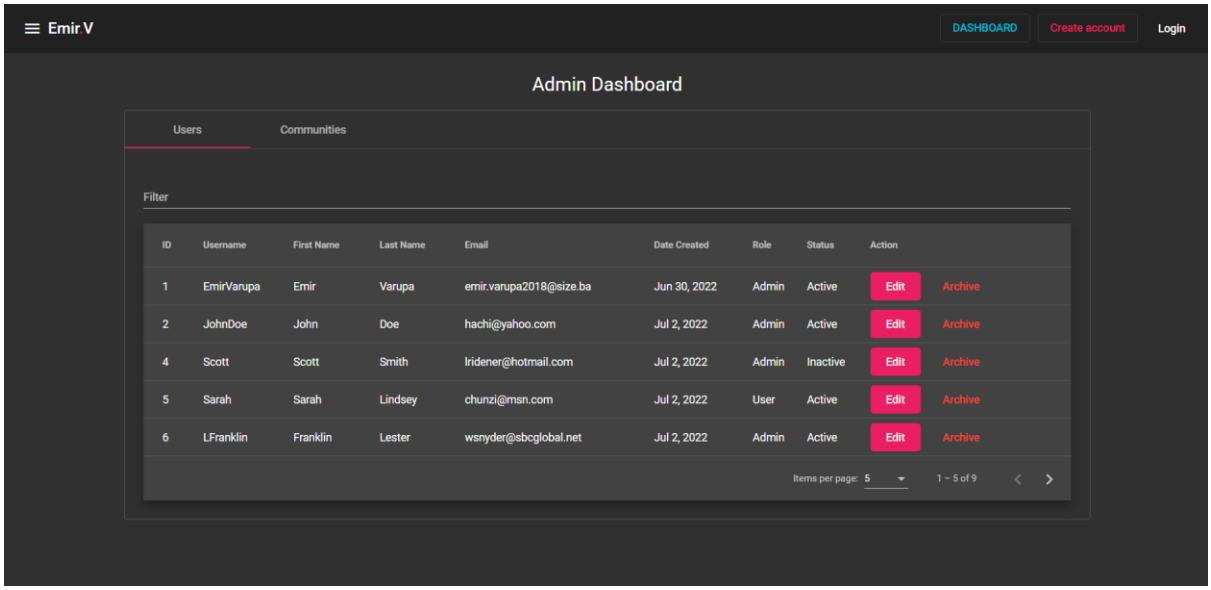
Slika 5.5 Kreiranje komentara u zajednici

Kreiranje novog korisnika je moguće ukoliko korisnik nije prijavljen. Korisnik ima ograničene mogućnosti sve dok nije prijavljen. Klikom na button Create account u headeru otvara se dijalog za kreiranje korisnika kao što je prikazano na slici 5.6. Nakon unosa svih podataka korisnik se uspješno kreira ako su svi podaci pravilno upisani. Ukoliko korisnik već ima račun i želi da izvrši prijavu, može da odabere opciju unutar dijaloga za registraciju kojim se otvori novi dijalog za prijavu korisnika.



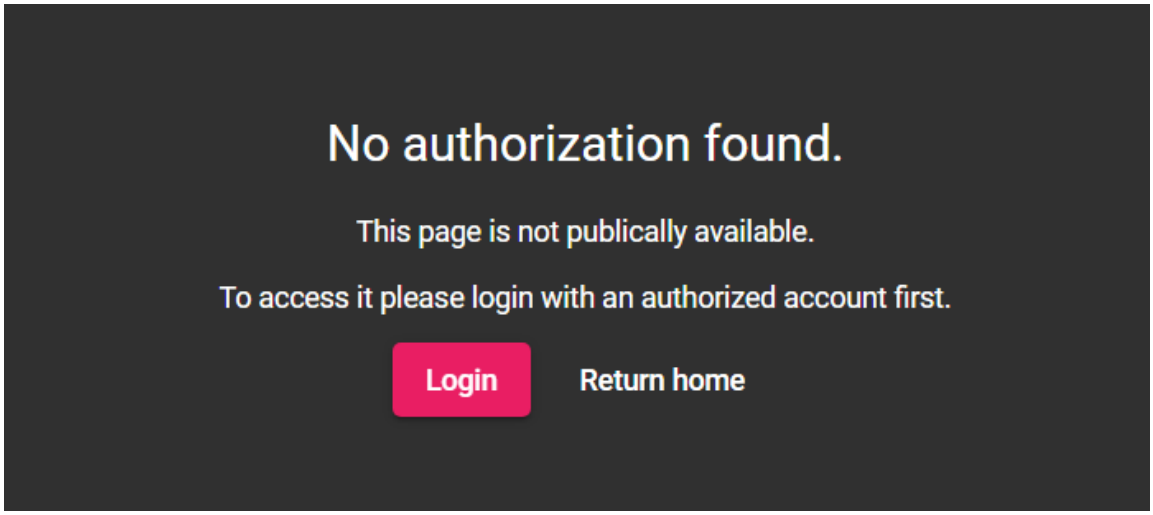
Slika 5.6 Kreiranje novog korisnika

Administratori imaju pristup nadzornoj ploči u kojoj mogu uređivati, dodavati i arhivirati korisnike i zajednice. Korisnici se prikazuju u tabeli koja ima mogućnost pretrage korisnika po bilo kojoj vrijednosti. Pored kolona koji prikazuju podatke korisnika ili zajednica, dodana je još jedna kolona koja sadrži akcije koje se mogu izvršiti na odabranom korisniku. Klikom na Edit otvara se dijalog za uređivanje korisnika ili zajednice koji izgledaju slično dijalozima za kreiranje, najveća razlika je u tome što se automatski unesu postojeći podaci u odgovarajuća polja. Klikom na button Archive otvara se dijalog koji traži da se potvrdi da li administrator želi da arhivira korisnika ili zajednicu. Ukoliko administrator potvrdi da želi izvršiti operaciju arhiviranja, odabrani korisnik ili zajednica se brišu iz odgovarajuće tabele. Na slici 5.7 je prikazana nadzorna ploča Internet foruma.



Slika 5.7 Admin Dashboard

Ukoliko je korisniku istekao JWT token ili je korisnik pokušao da pristupi stranici koja nije dozvoljena, korisnik će biti automatski proslijeđen na forbidden komponentu u kojoj se traži od korisnika da se prijavi na račun koji ima odgovarajuće permisije kao što je prikazano na slici 5.8.



Slika 5.8 Fobidden stranica

6. Sigurnosni elementi

6.1. Ograničavanje pristupa korisnika

Prvi sloj sigurnosti je ograničavanje pristupa korisnika. Pristup CRUD operacijama ograničen je korisnicima na web aplikaciji pomoću auth.guard i auth.interceptor. Sve stranice kojima ne smije obični korisnik imati pristup zaključane su pomoću presretača linkova tako što se na rutu do stranice postavi CanActivate koji je povezan sa AuthGuard komponentom. Presretači linkova traže autorizaciju kako bi se pristupilo odabranoj stranici tako što preusmjere korisnika na forbidden rutu gdje se traži od korisnika da se prijavi na račun koji ima pristup odabranoj stranici, ukoliko je prijava uspješna, korisnik se preusmjerava na homepage. Prilikom unosa podataka u formu korisnicima je ograničeno koje karaktere mogu upisati pomoću FormControl validatora.

Sljedeći sloj sigurnosti nalazi se između komponente i servisa unutar Angular-a, taj sloj sigurnosti naziva se servisni sloj, na primjer, za pristup CRUD metodama objave kreiran je posts.service servis, posts.service je odgovoran za poslovnu logiku i pozivanje iz backenda.

Pristup tabelama u bazi podataka ograničen je pomoću servisa, repozitorija, kontrolera i JWT tokena u backendu. Sve CRUD operacije iz servisa u frontendu šalju HTTP request preko kontrolera u backendu.

Pohrana šifre korisnika vrši se pomoću jednosmjernih heš funkcija, u ovom slučaju koristi se enkripcija nazvana HMACSHA512. HMACSHA512 je tip heš algoritma sa ključem koji je konstruisan od heš funkcije SHA-512 i koristi se kao Hash-based Message Authentication Code (HMAC).[12]

6.2. JWT autentifikacija i autorizacija

JSON Web Token (JWT) je otvoreni standard koji definira kompaktan i samostalan način za siguran prijenos informacija između strana kao JSON objekta. Ove informacije se mogu

[12] https://www.mssqltips.com/sqlservertip/4037/storing-passwords-in-a-secure-way-in-a-sql-server-database/

provjeriti i vjerovati jer su digitalno potpisane. JWT-ovi se mogu potpisati korištenjem tajnog ključa sa HMAC algoritmom.

Iako se JWT-ovi mogu enkriptovati kako bi se osigurala tajnost između strana, u ovom slučaju koriste se potpisani tokeni. Potpisani tokeni mogu provjeriti integritet zahtjeva sadržanih u njima, dok enkriptovani tokeni skrivaju te zahtjeve od drugih strana. Kada se tokeni potpisuju korištenjem parova javnih/privatnih ključeva, potpis također potvrđuje da je samo strana koja posjeduje privatni ključ ona koja ga je potpisala.

JSON Web Tokeni su korisni za autorizaciju i ujedno ovo je najčešći scenario za korištenje JWT-a. Nakon što je korisnik prijavljen, svaki sljedeći zahtjev će uključivati JWT, omogućavajući korisniku pristup rutama, uslugama i resursima koji su dozvoljeni s tim tokenom. Jedinstvena prijava je funkcija koja danas široko koristi JWT, zbog malih troškova i mogućnosti da se lako koristi u različitim domenima.

U svom kompaktnom obliku, JSON web tokeni se sastoje od tri dijela odvojena tačkama, a to su:

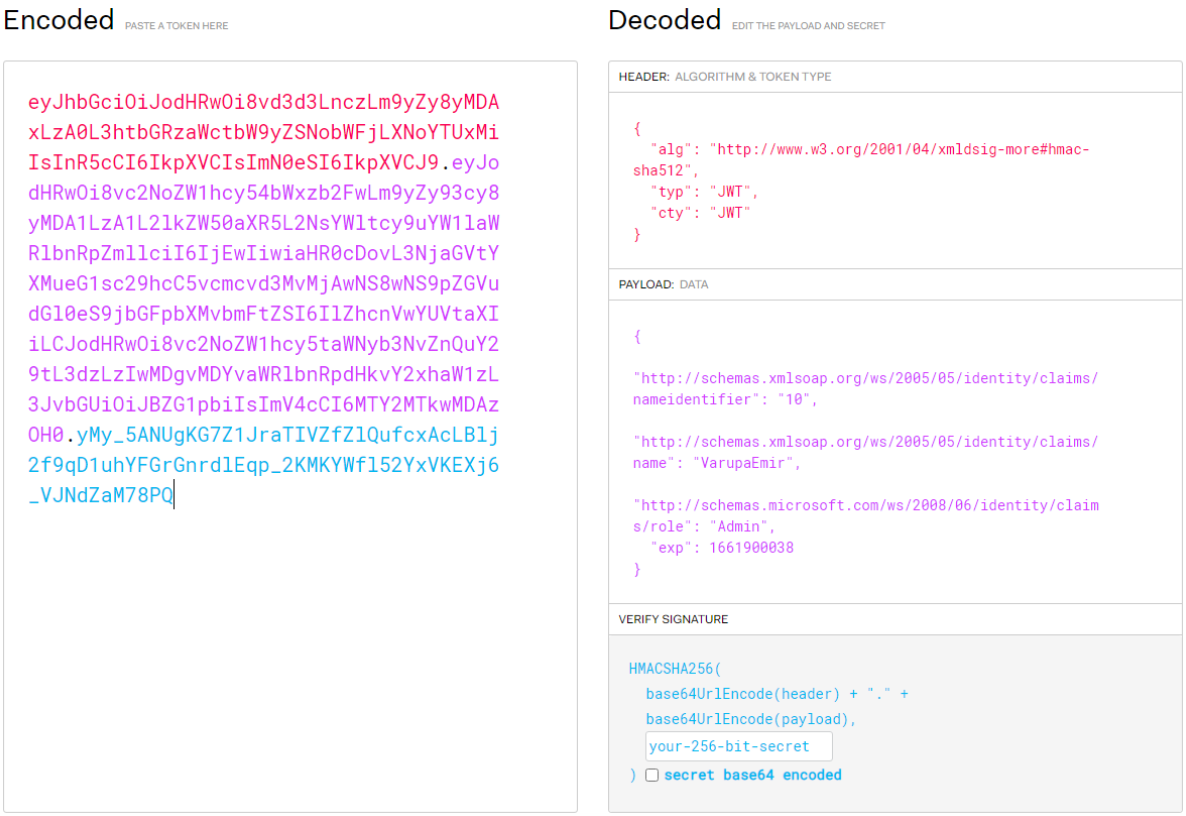
- Header
- Payload i
- Signature

Header se obično sastoji od dva dijela: tipa tokena, koji je JWT, i algoritma za potpisivanje koji se koristi, kao što je HMAC SHA256 ili RSA.

Drugi dio tokena je payload, koji sadrži zahtjeve (Claims). Zahtjevi (Claims) su izjave o entitetu (obično korisniku) i dodatni podaci. Postoje tri vrste zahtjeva: registrovani, javni i privatni zahtjevi.

Da bi se kreirao signature, potrebno je uzeti kodirani header, kodirani payload, tajnu, algoritam naveden u zaglavlju i potpisati to. Potpis se koristi za provjeru da poruka nije promijenjena.

Kada se sve sastavi, dobije se izlaz koji su tri Base64-URL niza odvojeni tačkama koji se lako mogu proslijediti u HTML i HTTP okruženjima. Sljedeća slika prikazuje JWT koji ima prethodno kodirani header i payload i potpisan je tajnom.



Slika 6.2 Primjer JSON Web Tokena

U autentifikaciji, kada se korisnik uspješno prijavi, bit će vraćen JSON Web Token. Budući da su tokeni korisničke informacije, mora se posvetiti velika pažnja kako bi se spriječili sigurnosni problemi. Općenito, ne treba držati tokene duže nego što je potrebno.

Kad god korisnik želi pristupiti zaštićenoj ruti ili resursu, angular bi trebao poslati JWT, obično u headeru autorizacije koristeći Bearer šemu. Sadržaj headera trebao bi izgledati ovako:

Authorization: Bearer <token>

Prednosti JSON Web tokena u poređenju sa Simple Web Tokenima (SWT) i Security Assertion Markup Language Tokenima (SAML). Kako je JSON manje opsežan od XML-a, kada je kodiran, njegova veličina je također manja, što JWT čini kompaktnijim od SAML-a. Ovo čini JWT dobrim izborom za prosljeđivanje u HTML i HTTP okruženjima.

Što se tiče sigurnosti, SWT može biti samo simetrično potpisan zajedničkom tajnom koristeći HMAC algoritam. Međutim, JWT i SAML tokeni mogu koristiti par javni/privatni ključ za potpisivanje. Potpisivanje XML-a sa XML digitalnim potpisom bez uvođenja nejasnih sigurnosnih rupa je veoma teško u poređenju sa jednostavnošću potpisivanja JSON-a. JSON parseri se koriste u većini programskih jezika i olakšavaju rad sa JWT jer se mapiraju direktno na objekte. Suprotno tome, XML nema prirodno mapiranje dokumenta u objekt.[13]

[13] https://jwt.io/introduction

7. Zaključak

U diplomskom radu se bavimo izgradnjom Internet forum web aplikacije korištenjem .NET-a za kreiranje backend API-ja i Angular-a za kreiranje korisničkog interfejsa. Za izradu baze korišten je Microsoft SQL Server na Windows operativnom sistemu. Kreiranje složenijih web aplikacija danas uključuje korištenje frontend i backend okvira kako bi se olakšao njihov razvoj. Korištenje frontend i backend okvira povećava brzinu i sigurnost razvoja i olakšava rješavanje složenih problema u kreiranju korisničkih interfejsa.

Angular je jedinstven sa svojom karakteristikom dvosmjernog povezivanja podataka (two-way data binding), što znači da postoji sinhronizacija u realnom vremenu između pogleda i modela. Dakle, kada se napravi bilo kakva promjena u modelu, ona se trenutno odražava na prikaz i obrnuto. Pored dvosmjernog povezivanja podataka, još dvije stavke su znatno olakšale i ubrzale razvoj frontend aplikacije: Angular Material i Flex-Layout.

- **Angular Material:** nudi kvalitetne UI komponente koje programeri mogu ubaciti u postojeće aplikacije, što znači da nije potrebno trošiti vrijeme na kreiranje vlastitih komponenti.
- **Flex Layout:** inteligentno automatizuje proces primene odgovarajućeg Flexbox CSS-a na hijerarhije pregleda pregledača. Ova automatizacija se takođe bavi mnogim složenostima i zaobilaznim rešenjima sa kojima se susreće sa ručnom primjenom Flexbox-a.

.NET pomaže da se brže razviju visokokvalitetne aplikacije. Language Integrated Query (LINQ) i asinkrono programiranje su ubrzali i olakšali razvoj backend dio Internet foruma.
.NET platformu službeno podržava Microsoft i vjeruju joj hiljade kompanija i milioni programera. Microsoft shvaća sigurnost vrlo ozbiljno i brzo objavljuje ažuriranja kada se otkriju prijetnje. .NET je također veoma brz, što znači da aplikacije pružaju bolje vrijeme odziva i zahtijevaju manje računarske snage.

Dakle, u ovom diplomskom radu izvršena je analiza i dizajn internet foruma koristeći UML kako bi se olakšalo kreiranje backenda i frontenda, zatim je kreirana većina baze podataka na osnovu dijagrama slučajeva i dijagrama klasa, i na kraju kreirana i prikazana je web aplikacija kako bi se lakše prikazale funkcionalnosti Angular-a i .NET-a.

Literatura

1. C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code, Mark J. Price, Packt Publishing, 2021.
2. Pro C# 7: With .NET and .NET Core, Andrew Troelsen, Philip Japikse, Apress, 2017.
3. Pro Angular 9: Build Powerful and Dynamic Web Apps, *Adam Freeman*, Apress, 2020.
4. Geeks for Geeks: Characteristics of .NET Framework, https://www.geeksforgeeks.org/characteristics-of-net-framework/ (pristupljeno 18. Septembra 2022)
5. IBM: Three-Tier Architecture: https://www.ibm.com/cloud/learn/three-tier-architecture (pristupljeno 14. Jula 2022)
6. Cyber Success: Angular Features - Things You Should Know About: https://www.cybersuccess.biz/angular-features/ (pristupljeno 18. Septembra 2022)
7. Alan Dennis, Barbara Wixom, Roberta M. Roth: *Systems Analysis and Design, 5th Edition*
8. C# Corner: What Is New in .NET 6.0: https://www.c-sharpcorner.com/article/what-is-new-in-net-6-0/ (pristupljeno 14. Jula 2022)
9. Angular - Glossary: View: https://angular.io/guide/glossary#view (pristupljeno 17. Augusta 2022).
10. Angular - Glossary: Router: https://angular.io/guide/glossary#router (pristupljeno 17. Augusta 2022).
11. Angular - Communicating with backend services using HTTP: https://angular.io/guide/http (pristupljeno 17. Augusta 2022).
12. RxJS - Introduction: https://rxjs.dev/guide/overview (pristupljeno 18. Augusta 2022).
13. npm - ngx-timeago: https://www.npmjs.com/package/ngx-timeago (pristupljeno 18. Augusta 2022).
14. auth0: Creating Beautiful Apps with Angular Material: https://auth0.com/blog/creating-beautiful-apps-with-angular-material/ (pristupljeno 18. Augusta 2022).
15. MSSQLTips: Storing passwords in a secure way: https://www.mssqltips.com/sqlservertip/ (pristupljeno 21. Augusta 2022).
16. JWT: Introduction to JSON Web Tokens: https://jwt.io/introduction (pristupljeno 29. Augusta 2022).