Portal za kratke priče

Pejić, Petar

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:349294

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-06**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

dabar

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

PORTAL ZA KRATKE PRIČE

Diplomski rad

Petar Pejić

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 16.09.2021.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Petar Pejić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1078R, 06.10.2019.
OIB studenta:	39213705040
Mentor:	Izv. prof. dr. sc. Mirko Köhler
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Ivica Lukić
Član Povjerenstva 1:	Izv. prof. dr. sc. Mirko Köhler
Član Povjerenstva 2:	Robert Šojo
Naslov diplomskog rada:	Portal za kratke priče
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	Tema je zauzeta. Zadatak završnog rada je napraviti web portal za objavu natječaja i gotovih kratkih priča. Administrator može otvoriti novi natječaj, postaviti mu temu, trajanje i pravila. Svi registrirani autori mogu predati svoje priče na tu temu (ako zadovoljavaju pravila). Također registrirani korisnici mogu glasati za pojedinu priču ili prijaviti plagijat.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	16.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis: Köhler Datum: 27. 9. 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2021.

Ime i prezime studenta:	Petar Pejić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D-1078R, 06.10.2019.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Portal za kratke priče**

izrađen pod vodstvom mentora Izv. prof. dr. sc. Mirko Köhler

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD ... 1
1.1. Zadatak diplomskog rada ... 1
2. POSTOJEĆE APLIKACIJE SLIČNE TEMATIKE ... 2
3. TEHNOLOGIJE PROMJENJENE ZA IZRADU APLIKACIJE ... 4
3.1. Klijentske tehnologije ... 4
3.1.1. Angular ... 4
3.1.2. HTML ... 6
3.1.3. CSS ... 7
3.1.4. TypeScript ... 7
3.2. Serverske tehnologije ... 7
3.2.1. Spring Boot ... 8
3.2.2. Java ... 9
3.2.3. Hibernate ... 9
3.2.4. PostgreSQL ... 10
4. IZRADA PORTALA ZA KRATKE PRIČE ... 11
4.1. Izrada web sučelja ... 11
4.1.1. Navigacija ... 11
4.1.2. Lazy loading ... 12
4.1.3. Servisi ... 13
4.2. Izrada serverske strane ... 18
4.3. Baza podataka ... 23
5. IZGLED APLIKACIJE ... 24
5.1. Responzivnost ... 30
6. ZAKLJUČAK ... 31
LITERATURA ... 32
SAŽETAK ... 33
ABSTRACT ... 34
ŽIVOTOPIS ... 35
PRILOZI ... 36

1. UVOD

Internet je prepun različitog sadržaja. Sadržaj se definira kao skup svih oznaka kojima je nešto ispunjeno. Dok ga neki koriste za poslovne dogovore, oglašavanje, obrazovanje, promociju sadržaja itd. drugom dijelu korisnika služi za zabavu. Statistički, prosječno svaka osoba na internetu potroši malo manje od sedam sati dnevno. [1]

Nepregledna ponuda zabave na internetu teško može dosaditi. Internet je dao trenutni pristup svim sadržajima i omogućio je većem broju ljudi da stvaraju kako bi drugi mogli ispunjavati vrijeme. Također, ljudi kroz interakciju poboljšavaju komunikacijske vještine te ih se potiče na kreativno razmišljanje.

Internet svojim korisnicima nudi pregršt sadržaja, informacija i zabave što je uzrokovalo i preseljenje knjiga u online svijet. Iako se gotovo svi slažu s činjenicom da ništa ne može zamijeniti ritual odlaska u knjižnicu i miris nove, neotvorene knjige, zahvaljujući tehnologiji gotovo sve naslove moguće je pronaći u samo par klikova. [2]

Osim čitanja romana i ostalog dužeg sadržaja sve je popularniji vid zabave koji pruža čitanje kraćih tekstova temeljenih na iskustvima. Glavna prednost takvog formata pruža dinamiku te veći broj različitih pisaca može iz više kutova odnosno perspektiva opisati neku temu.

Aplikacija predstavljena u radu ima za cilj ponuditi korisnicima vid zabave u oblicima kratkih priča gdje bi se teme izmjenjivale u određenom periodu.

U drugom poglavlju rada opisane su postojeće, slične, aplikacije na tržištu. Analizirat će se prednosti i nedostaci istih, kao korisničko iskustvo temeljeno na subjektivnom dojmu. Nakon toga u trećem poglavlju slijedi opis tehnologija korištenih za izradu aplikacije na klijentskoj i serverskoj strani.

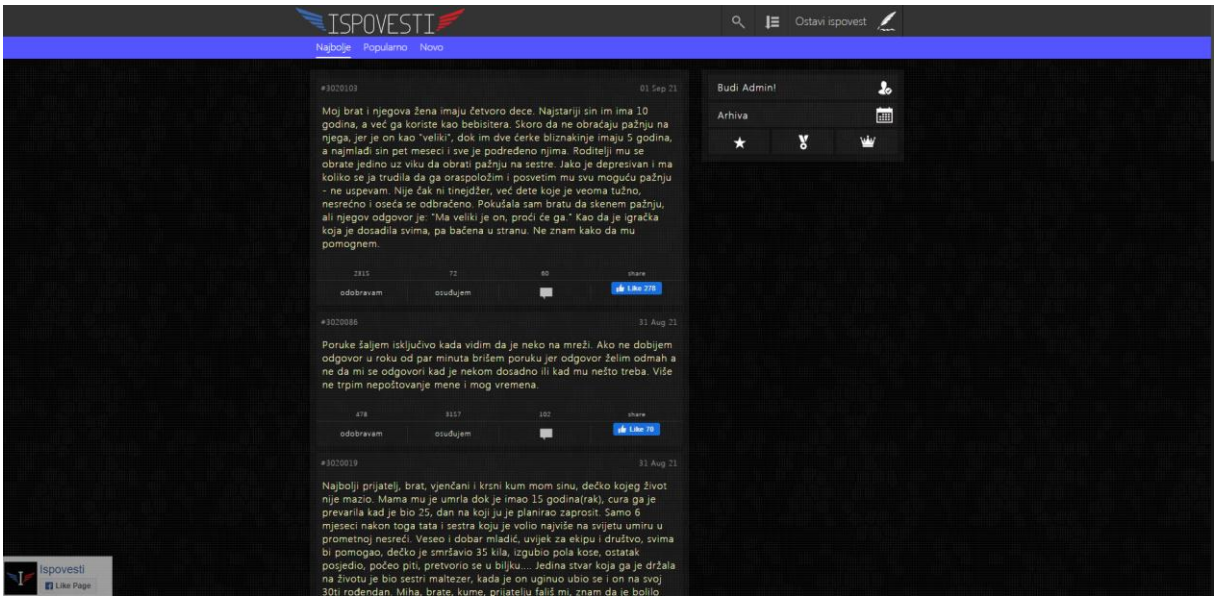
Četvrto poglavlje donosi detaljan opis postupka izrade aplikacije koji je popraćen grafičkim prikazima koda kao i krajnjeg proizvoda. Svaka slika popraćena je tekstom koji opisuje korištene tehnike.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je napraviti web portal za objavu natječaja i gotovih kratkih priča. Administrator može otvoriti novi natječaj, postaviti mu temu, trajanje i pravila. Svi registrirani autori mogu predati svoje priče na tu temu (ako zadovoljavaju pravila). Također registrirani korisnici mogu glasati za pojedinu priču ili prijaviti plagijat.

2. POSTOJEĆE APLIKACIJE SLIČNE TEMATIKE

Prije same izrade potrebno je provesti istraživanje tržišta, domaćeg i stranog, kako bi se provjerila prisutnost aplikacija slične tematike. Kao primjer sličnoga proizvoda na našem govornom području može se uzeti aplikacija pod nazivom „Ispovesti“ [1] koja nudi bogat sadržaj priča s različitim temama. Svatko može ostaviti priču bez ikakve prijave ili registracije. Svakoj priči moguće je ostaviti ocjenu u vidu „odobravanja“ ili „osuđivanja“. Također, ako korisnik želi može komentirati priču te po volji komentar može biti anoniman ili pod određenim nadimkom. Sve priče moguće je sortirati prema kategorijama „Najbolje“, „Popularno“ i „Novo“. Zanimljiva mogućnost koja je ponuđena korisnicima je da odobravaju ili odbacuju nove priče koje će se tek naknadno pojaviti na naslovnom ekranu ukoliko budu odobrene. Što se tiče samog izgleda aplikacije, dominira crna pozadina koja je blago posvijetljena na području s pričama kao što je prikazano na slici broj 2.1.



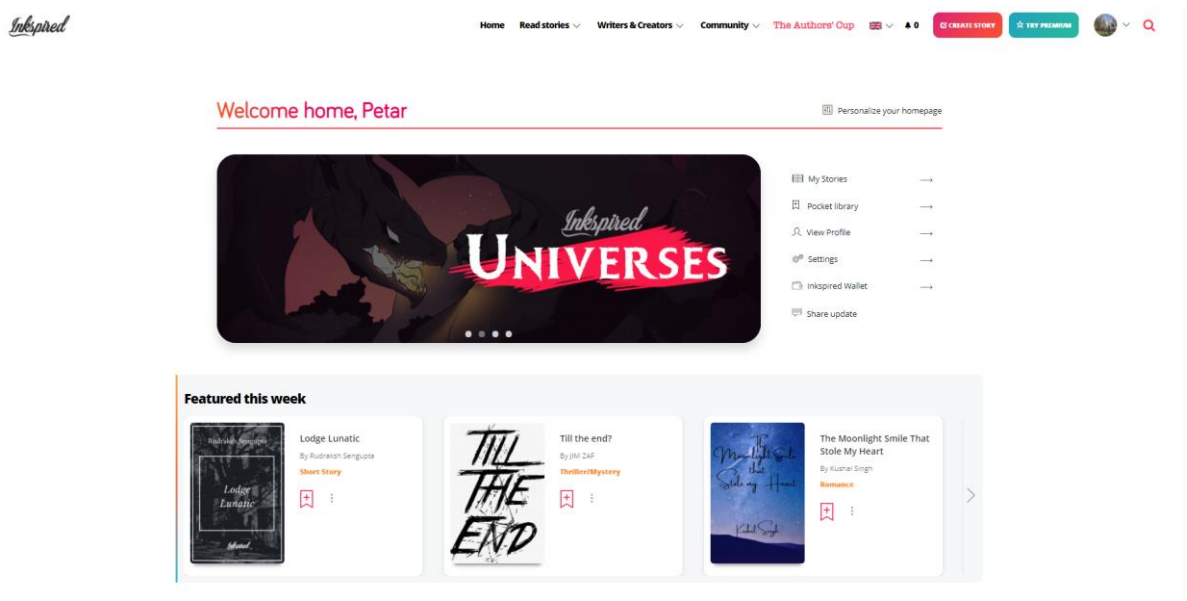
Slika 2.1. Izgled aplikacije Ispovesti

Drugi primjer je web stranica pod nazivom „Inkspired“.[2]

Želimo narušiti tradicionalni način izdavačkog modela i načina pisanja sadržaja. Radi se o multikulturalnom timu pisaca, dizajnera i strastvenih čitatelja, s misijom učiniti neovisno izdavaštvo lakšim, pametnijim, fleksibilnijim i poštenijim; i pružiti najbolje iskustvo čitanja na bilo kojem uređaju čitateljima u pokretu. [3]

[1] http://ispovesti.com/
[2] https://getinkspired.com/en/

To je sveobuhvatna aplikacija za pisanje priča koje se ne baziraju na iskustvu već na fantaziji. Prije pisanja svake priče odredi se kategorija i naslov kao i naslovna slika. Na slici 2.2 može se vidjeti naslovna stranica s ponuđenim sadržajem za čitanjem.



Slika 2.2. Naslovna stranica web aplikacije Inkspired

U usporedbi s aplikacijom „Ispovesti“, „Inkspired“ je veći naglasak stavio na korisničkom sučelju te je puno više animacija i interakcije s korisnicima. Također, same priče su puno duže i zapravo su priče u pravom smislu riječi. Korisnici mogu pratiti jedni druge te na taj način dobivati obavijesti kada korisnik objavi priču. Svaki korisnik po želji stvara listu priča koje može pročitati kasnije ukoliko trenutno nije u mogućnosti.

3. TEHNOLOGIJE PROMJENJENE ZA IZRADU APLIKACIJE

U ovom poglavlju opisane su tehnologije korištene za izradu aplikacije. Objasnit će se što su klijentske, a što serverske tehnologije te svaka od njih posebno.

3.1. Klijentske tehnologije

Klijentske tehnologije su one tehnologije koje se izvode na strani krajnjeg korisnika odnosno u korisničkom pregledniku. Kod svake od njih može se vidjeti tako da se u pregledniku odabere opcija pod nazivom „Razvojni alati“. Tu su uključene slike, tekst, stil različitih elemenata i sve drugo što spada u korisničko sučelje.

Također, programeri često klijentske tehnologije nazivaju „Frontend“.

3.1.1. Angular

Angular je programski okvir otvorenog koda koji se temelji na TypeScript-u. Koristi se za izgradnju aplikacija s jednom web stranicom. Aplikacija s jednom stranicom je implementacija koja znači da se web dokument učitava samo jednom te se nakon toga samo sadržaj izmjenjuje. [4] U Angular-u je to svojstvo razvijeno uz pomoć ruta o kojima će se nešto više reći u nastavku. Razvijen je i održavan od strane Google-a 2010. godine.

Koristi se za izgradnju skalabilnih web aplikacija koje se baziraju na komponentama.

Sadrži kolekciju integriranih biblioteka koje pokrivaju različita svojstva koja Angular nudi, što uključuje rukovanje rutama, upravljanje formama, komunikaciju sa serverom i mnoge druge stvari.

Objedinjuje različite programske alate koji pomažu u izgradnji, testiranju, razvoju te izmjenama koda.

Angular posjeduje mnogo specifičnih svojstava koji čine razvoj programskih aplikacija lakšim. Neke od njih opisane su u nastavku.

Komponente su glavne gradbene jedinice svake Angular aplikacije. Svaka komponenta uključuje TypeScript klasu, HTML predložak i CSS dokument koji služi za stiliziranje. Potrebno je dodati dekorator @Component koji označava da se radi o komponenti te pruža konfiguracijske meta podatke i određuje kada će komponenta biti prikazana te procesuirana. [5]

Na slici 3.1. prikazan je primjer Angular komponente.

```
import { Component } from '@angular/core';

@Component({
  selector: 'hello-world',
  template: `
    <h2>Hello World</h2>
    <p>This is my first component!</p>
  `
})
export class HelloWorldComponent {
  // The code in this class drives the component's behavior.
}
```

Slika 3.1. Primjer jednostavne Angular „Hello World“ komponente [6]

Također, jedan od moćnih svojstava Angular-a je *dodavanje ovisnosti* (engl. *Dependency Injection*).

Ovisnosti su servisi koje klasa treba za izvršavanje njezine funkcionalnosti. To je obrazac gdje se objekti predaju drugim objektima kako bi obavili zajednički zadatak. Nije potrebna nikakva instalacija jer Angular rukuje instalacijama za korisnika. [7]

Za označavanje klase ili objekta potrebno je koristiti dekorator @Injectable koji dopušta da bude dodan kao ovisnost. [8]

U aplikacijama s jednom stranicom, umjesto slanja zahtjeva serveru za novu stranicu, sve što korisnik vidi izmjenjuje se na temelju komponenata koje se prikazuju, dok je web dokument učitan samo jednom. Komponenta koja se prikazuje mora imati definiranu rutu te kada URL (engl. *Uniform Resource Locator*) odgovara zadanoj ruti na ekranu se prikazuje sadržaj komponente.

Za rukovanje navigacijom koristi se klasa *Router*. Ona omogućuje dohvaćanje URL-a preglednika kao uputu za promjenu prikaza. [9]

Angular *direktive* su atributi koji se dodaju u HTML (engl. *HyperText Markup Language*) elemente te služe za izmjenu ponašanja ili prikazivanja DOM (engl. *Document Object Model*) elemenata.

Direktive dijelimo na atributne i strukturne. Atributne direktive mijenjaju izgled ili ponašanje elementa dok strukturne mijenjaju raspored DOM-a tako što dodaju ili brišu elemente.

Atributne direktive su NgClass, NgStyle, NgModel dok su strukturne NgIf, NgFor, NgSwitch. Prije upotrebe svake od njih potrebno je dodati *.

3.1.1.1.Angular CLI

Angular CLI (engl. *Command-line interface*) je alat koji se koristi za inicijalizaciju, razvoj i održavanje aplikacije direktno iz komandnog sučelja.

CLI se instalira uz pomoć npm (engl. *Node Package Manager*) paketnog upravljača. Naredba za instalaciju prikazana je na slici 3.2.

```
npm install -g @angular/cli
```

Slika 3.2. Instalacija Angular CLI-a

Nakon što je CLI instaliran moguće je pokrenuti liniju koda za kreiranje novog projekta kao i pokretanje projekta. Obije akcije prikazane su na slici 3.3.

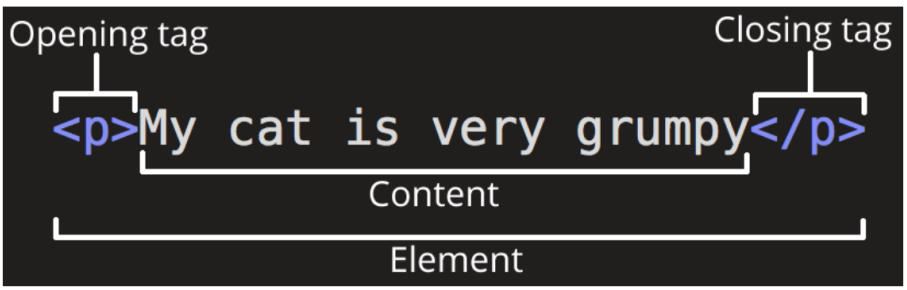
```
ng new my-first-project
cd my-first-project
ng serve
```

Slika 3.3. Kreiranje i pokretanje projekta

3.1.2. HTML

HTML je standardni opisni odnosno prezentacijski jezik za izradu web stranica. Tehnologije koje se uz njega koriste uglavnom služe za stilizaciju elemenata ili dodavanje funkcionalnosti (CSS, TypeScript). [10] On definira strukturu sadržaja stranice. Sastoji se od velikog broja elemenata koji se koriste za različite dijelove te daju stranici određen izgled. Svaki element ima svoju oznaku za otvaranja, za zatvaranje i sadržaj.

Prikaz jednog HTML elementa dan je na slici 3.4.



Slika 3.4. Prikaz Paragraf elementa HTML-a

Također, elemente je moguće ugnježđivati tako da je unutar sadržaja moguće dodavati više različitih elemenata. Primjer ugniježđenog elementa dan je na slici 3.5.

```
<p>My cat is <strong>very</strong> grumpy.</p>
```

Slika 3.5. Element „strong“ nalazi se unutar elementa „p“

Elementima se mogu pridružiti atributi koji se dodaju u oznaku za otvaranje. Primjerice za element „button“ može se dodati svojstvo „disabled“ koji onemogućuje da se pritisne na gumb ili u elementu „a“ može se dodati „href“ atribut koji specificira link na koji se preusmjerava.

3.1.3. CSS

CSS je kod koji služi za dodavanje stila HTML elementima. Stil se može dodavati linijski u elementu ili se može odvojiti u posebnu datoteku. Ukoliko su elementi i stil odvojeni oni se povezuju preko klasa ili ID-ova. Primjer takvog dodavanja stila vidljiv je na slici 3.6.

```
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
```

Slika 3.6. Dodavanje stila prema ID-u s nazivom „myHeader“

Linijsko dodavanje stila nešto je drugačije. Povlake su zamijenjene velikim početnim slovom sljedeće riječi. Dodavanje stila može se vidjeti na slici 3.7.

```
<p style="color:red;">This is a paragraph.</p>
```

Slika 3.7. Linijsko dodavanje stila

3.1.4. TypeScript

TypeScript je programski jezik, razvijen i održavan od strane Microsofta, koji se temelji na JavaScriptu. On je jezik više razine s više metoda, koje ne postoje u potonjem, koje čine programiranje lakšim.

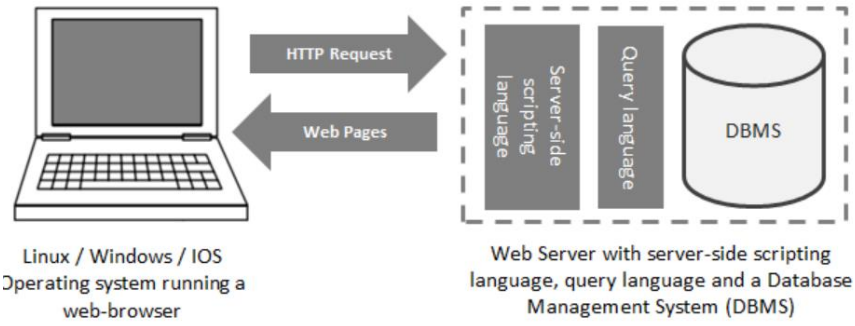
On se ne pokreće u pregledniku nego se kompajlira u JavaScript te je kod koji se vidi kada se otvore razvojni alati zapravo JavaScript.

Razlog zašto se TypeScript koristi u Angular-u je taj što su mnoga njegova svojstva dostupna samo u tomu jeziku.

3.2. Serverske tehnologije

Serverske tehnologije su one koje se koriste za izgradnju sustava koji se pokreće na serveru. To je strana sustava radi u pozadini gdje se upravlja podacima.

Serverske tehnologije obuhvaćaju širok raspon softverskih tehnologija kao što su serverske skripte, DBMS (engl. *Database Management Systems*), web server softver (npr. Apache) i mnogo drugih tehnologija o kojima ovisi izgradnja aplikacije. Obrada zahtjeva prikazana je na slici 3.8.



Slika 3.8. Obrada HTTP zahtjeva na serverskoj strani

Slično kao i kod klijentske strane, svi procesi koji se odvijaju na serverskoj nazivaju se „backend".

3.2.1. Spring Boot

Spring je programski okvir otvorenog koda koji omogućava razvojnim programerima izgradnju jednostavnih, pouzdanih i skalabilnih aplikacija. Usredotočava se na pružanje mogućnosti za upravljanje logikom na različite načine.

Može se promatrati kao skup programskih okvira gdje spadaju Spring AOP, Spring ORM, Spring Web MVC … Bilo koji od ovih modula može se zasebno koristiti pri izradi aplikacije kako bi joj pružili dodatne mogućnosti odnosno funkcionalnosti. [11]

Spring Boot je modul Spring-a koja se temelji na Java programskom jeziku. Pojednostavljuje početno postavljanje aplikacije za rad. Pruža sva svojstva Spring-a samo na jednostavniji način.

Spring Boot smanjuje potrebu za pisanje koda za konfiguraciju same aplikacije. Smanjenje koda omogućeno je pisanjem anotacija koje obavljaju posao u pozadini. Posljedično programeru se smanjuje vrijeme potrebno za razvoj aplikacije. Dolazi s ugrađenim Tomcat serverom.

Uz njega se vrlo često koristi Maven koji služi za lakše upravljanje ovisnostima. Uz pomoć njega vrlo lako kroz pom.xml datoteku uključujemo brojne biblioteke čime postižemo željene funkcionalnosti. Primjer uključivanja paketa dan je na slici 3.9.

```
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-search-engine</artifactId>
    <version>5.11.7.Final</version>
</dependency>
```

Slika 3.9. Uključivanje Hibernate biblioteke u projekt kroz Maven

3.2.2. Java

Java je jedan od najpopularnijih programskih jezika na svijetu. Može se koristiti za razvoj web, ali i mobilnih aplikacija. Jednostavan je, učinkoviti i objektno-orijentirani programski jezik. Java je vrlo prenosiva. Ista aplikacija izrađena u Javi može se izvoditi na bilo kojem računalu unatoč hardverskim postavkama i na bilo kojem operacijskom sustavu. Jedini uvjet je da posjeduje Java interpreter.

Zašto koristiti Javu

1. Platformski je neovisna- aplikacija može biti napisana na bilo kojoj platformi i biti pokrenuta na nekoj drugoj bez ikakvih modifikacija
2. Objektno orijentirana je – pomaže u pisanju fleksibilnijeg koda koji se može ponovno iskoristiti
3. Brzina- Vrlo je dobro optimizirana te je brza kao i neki jezici niže razine [12]

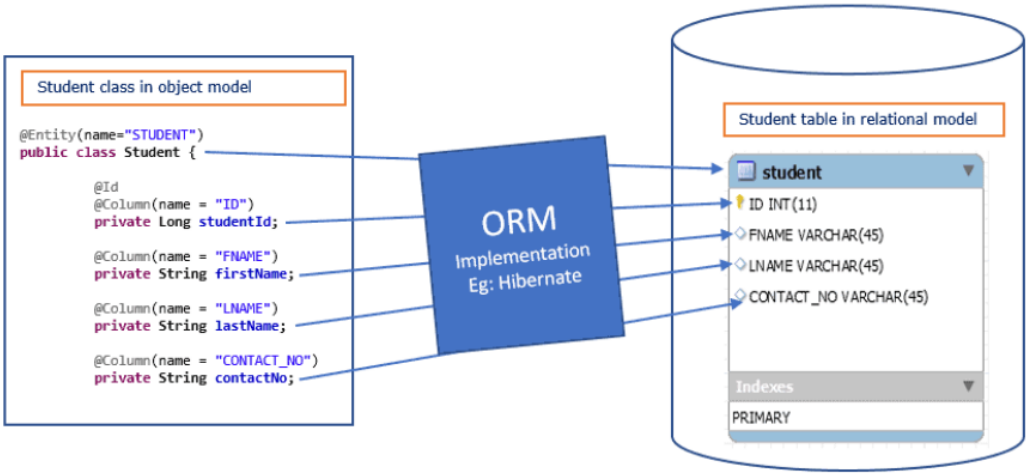
3.2.3. Hibernate

Hibernate je Java programski okvir koji omogućava jednostavnu interakciju aplikacije s bazom podataka. Pretvara objekte u podatke kojima je omogućeno spremanje i dohvaćanje u relacijsku bazu podataka.

Budući da su tablice relacijske, a moderni programski jezici objektno orijentirani oni nisu kompatibilni, odnosno njihovo mapiranje je otežano.

Hibernate je ORM (engl. *Object Relational Mapping*) alat koji omogućava kreiranje upravljanje i pristup podacima. Također koristi tehniku za mapiranje podataka koje pohranjuje u bazu.

Primjer mapiranja prikazan je na slici 3.10.



Slika 3.10. Mapiranje klase student u tablicu student

Način na koji Hibernate omogućava mapiranje ne bi bio moguć bez anotacija. One sadrže meta podatke o objektu koji je potrebno prilagoditi relacijskoj tablici u koju se spremaju podaci. Neke od najpoznatijih anotacija su @Entity, @Id, @Table, @Column…

3.2.4. PostgreSQL

PostgreSQL je relacijska baza podataka otvorenog koda koji podržava relacijske i nerelacijske zahtjeve. Radi se o konzistentnom sustavu za upravljanje bazom podataka. Primarna zadaća je skladištenje podataka za web i mobilne aplikacije.

Koristi se od 1986, a danas slovi za jedan od najboljih okruženja za spremanje velikih količina podataka te pomaže programerima u izgradnji i razvoju složenih aplikacija kao i pokretanje administrativnih zadataka. [13]

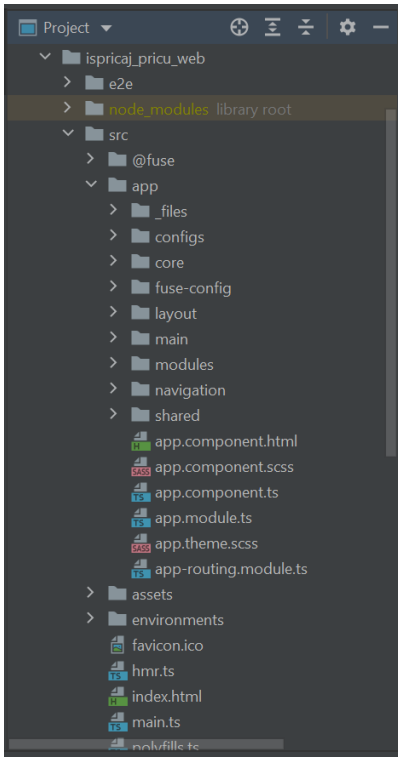
PostgreSQL pruža opsežan podatkovni kapacitet te podržava kompleksne strukture podataka. Nudi velik broj funkcija koja pomažu u stvaranju novih aplikacija te vrlo dobro štiti integritet podataka.

4. IZRADA PORTALA ZA KRATKE PRIČE

Ovo poglavlje sastoji se od opisivanja postupka izrade aplikacije te objašnjenja koda koji je korišten za istu.

4.1. Izrada web sučelja

Sučelje se odnosi na dio aplikacije s kojim korisnik obavlja interakciju. Za njega je korišten programski okvir Angular, a programski jezik TypeScript koji su ranije opisani. Sama aplikacije klijentske strane sastoji se od jednog HTML dokumenta u kojemu se izmjenjuje sadržaj. Dokument koji se prvi otvara je index.html koji se sastoji od svih komponenata koje postoje te se one izmjenjuju prema ruti. Struktura klijentskog dijela aplikacije dana je na slici 4.1.



Slika 4.1. Struktura aplikacije

Glavne komponente aplikacije nalaze se u mapi „modules“ te one služe za prikazivanje određene rute dok se servisi, konfiguracija te komponente imaju višestruku upotrebu nalaze u ostalim datotekama.

Korijenski element je App te su svi ostali elementi odnosno komponente njegova djeca.

4.1.1. Navigacija

Izmjena sadržaja odvija se pomoću direktive router-outlet te se koristi za označavanje predloška koji će biti prikazan. Moguće imati više u jednom projektu. Korištenje je prikazano na slici 4.2.

Slika 4.2. Korištenje router-outleta

Router-outlet ne bi se mogao sam koristiti bez glavnog modula (app.module.ts) te ostalih koji su podijeljeni na manje dijelove zbog preglednosti. Pregled glavnog modula dan je na slici 4.3.

Slika 4.3. Definiranje rute i dodavanje pod modula u glavni

Svakoj ruti potrebno je definirati putanju te komponentu koja odgovara istoj što je prikazano na slici 4.4. U glavnom modulu „**“ označavaju sve nedefinirane putanje te preusmjeravanje na željenu rutu.

Slika 4.4. Definiranje rute za login modul

4.1.2. Lazy loading

Kako bi sve funkcioniralo svi moduli moraju se dodati u modul koji sadrži sve module (AppRoutingModule), koji se brine za njihovo pravovremeno učitavanje na zahtjev. To je prikazano na slici 4.5.

Slika 4.5. AppRoutingModule

Tehnika koja se koristi za učitavanje modula naziva se lijeno učitavanje (engl. *Lazy loading*). Zadano ponašanje je da se cijeli dokument učitava odjednom prije nego je neka komponenata potrebna. To može uzeti više vremena te dolazi do kašnjenja. Kako bi se to spriječilo dolazi se do tehnike lijenog učitavanja koji kod pojedine komponente učitava onda kada je zahtijevana.

4.1.3. Servisi

Servisi su dio svake ozbiljne Angular aplikacije. Služe kao centralni repozitorij gdje su podaci spremljeni i gdje se nalazi sva logika bitna za njihovo rukovanje. Također, servisi služe kao posrednici između komuniciranja dviju ili više komponenti. Primjer na slici 4.6. prikazuje svrhovitost servisa. Osim što se u njemu nalaze svi objavljeni postovi, ovaj servis sadrži i funkcionalnosti koje sortiraju polje objekata prema atributima te takvo polje šalje u komponentu koja je zatražila sortiranje.

Komunikacija se obavlja pomoću klase Subject. To je poseban tip podatka koji omogućava njegovo osluškivanje. Tako se na primjeru na slici 4.6. može vidjeti kako se nova vrijednost polja nakon sortiranja prosljeđuje dalje, a komponenta koja osluškuje prima novo sortirano polje i prikazuje ga. Osluškivanje Subjecta prikazano je na slici 4.7.

```
export class PostService{

    postsChanged = new Subject<Post[]>();

    private _loadedPosts: any = [];

    constructor() {
    }

    get loadedPosts(): Post[] {...}

    set loadedPosts(posts: Post[]) {...}

    sortByLies() {
        this.loadedPosts.sort( compareFn: function(post1 :Post , post2 :Post ){
            return post2.falseNumber - post1.falseNumber
        })
        this.postsChanged.next(this.loadedPosts)
    }

    sortByTrues() {
        this.loadedPosts.sort( compareFn: function(post1 :Post , post2 :Post ){
            return post2.trueNumber - post1.trueNumber
        })
        this.postsChanged.next(this.loadedPosts)
    }
```

Slike 4.6. Post servis

```
this.postService.postsChanged.subscribe(
      next: (posts: Post[]) => {
          this.posts = posts
      }
)
```

Slika 4.7. Osluškivanje Subject tipa podatka

Nakon što se postovima postavi nova vrijednost ponovno se izvršava dio koda koji je namijenjen za njihovo prikazivanje. Taj kod je prikazan na slici 4.8.

```
<div *ngFor="let post of posts" fxFill>
    <app-post-item [post]="post"
                   [userReactions]="userService.loggedUser.reactions">
    </app-post-item>
</div>
```

Slika 4.8. Izlistavanje učitanih postova

Kako se radi o ponavljajućoj radnji korištena je direktiva *ngFor koja ponavlja dječji element za svaki element u polju. Budući da se radi o istom objektu (Post), komponenti app-post-item prosljeđuje se objekt te ga ona dekoratorom @Input prihvaća i prikazuje. Na slici 4.8. može se vidjeti kako se u uglatim zagradama zadaje ime varijable dok se nakon znaka = predaje objekt kao i reakcija prijavljenog korisnika na post koji se kreira. Na slici 4.9. prikazano je prihvaćanje objekta i reakcije.

```
@Input() post: Post;
@Input() userReactions;
```

Slika 4.9. Prihvaćanje vrijednosti unutar dječje komponente

Jedan od servisa koji se vrlo često koristi je Resolver. On se kreira tako da servis implementira sučelje Resolve te prepisuje njegovu metodu resolve. On se koristi kada želimo neki posao obaviti prije nego se web dokument rendera. Ruta koja u sebi sadrži resolver uvijek će biti renderana samo što će prije toga pokrenuti određen kod. To se uglavnom koristi za dohvaćanje podataka sa servera kako bi prilikom prikazivanja podaci već bili spremni umjesto da se krenu dohvaćati nakon što se ruta rendera. Primjer resolvera može se vidjeti na slici 4.10.

```
export class CardResolverService implements Resolve<Post[]>{

    constructor(private postService: PostService,
        private cardsService: CardsService,
        private userService: UserService,
        private dataStorageService: DataStorageService) { }

    resolve(route: ActivatedRouteSnapshot, state: RouterStateSnapshot): Post[] {

        const posts = this.postService.loadedPosts;
        const theme = this.cardsService.loadedTheme;

        if(!theme){
            this.dataStorageService.fetchActiveTheme()
        }
        else{
            this.cardsService.activeTheme.next(theme)
        }

        this.dataStorageService.fetchPosts()

        return posts
    }
}
```

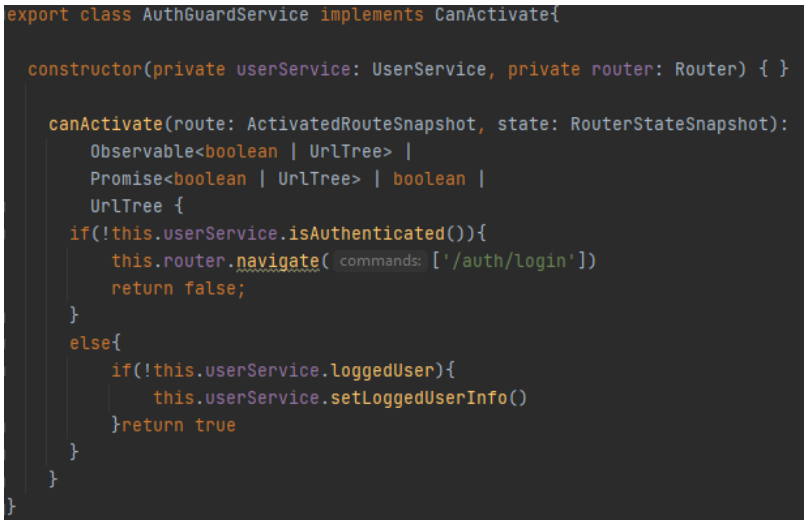
Slika 4.10. Resolver za cards komponentu

Također, potrebno je ruti dodati resolver na način kao što je prikazano na slici 4.11.

```
const routes: Routes = [
    {
        path: 'posts',
        component: CardsComponent,
        canActivate: [AuthGuardService],
        resolve: {
            data: CardResolverService
        }
    }
];
```

Slika 4.11. Dodavanje resolvera na rutu

Osim resolvera vrlo popularan servis koji se često koristi jest CanActivate. Za razliku od prethodno spomenutog, CanActivate odlučuje hoće li se ruta prikazati ili neće. Nakon što se provjere uvjeti dolazi se do odluke. Ako funkcija vraća true ruta se prikazuje, a ukoliko ne ruta se preusmjerava na neku drugu. Primjer CanActivate servisa koji se brine o tome je li korisnik prijavljen prikazan je na slici 4.12. Direktan pristup nekoj ruti bez prijave time je onemogućen odnosno korisnika će se stalno vraćati na stranicu za prijavu.



Slika 4.12. AuthGuardService koji brine o prijavi korisnika

4.1.4. Prijavljen korisnik

Da bi korisnik se korisnik mogao prijaviti najprije je potrebno obaviti registraciju gdje unosi samo najosnovnije pojmove bitne za funkcioniranje aplikacije.

Nakon što je korisnik prijavljen sprema se u bazu te se za pristup aplikaciji svaki puta mora prijaviti. Ako je prijava neuspješna dobiva upozorenje o tome, dok ako je prijava uspješna, u lokalnu memoriju (engl. *Session storage*) sprema se token koji sadrži informacije o korisniku. Dobivanjem tokena na spremaju se informacije o njemu. Također, informacije se gube nakon što se osvježi stranica no budući da svaka stranica ima implementiran AuthGuardService sa slike 4.12., ukoliko je token spremljen, on se u zaglavlju šalje te se prema njemu dobivaju informacije sa servera o korisniku.

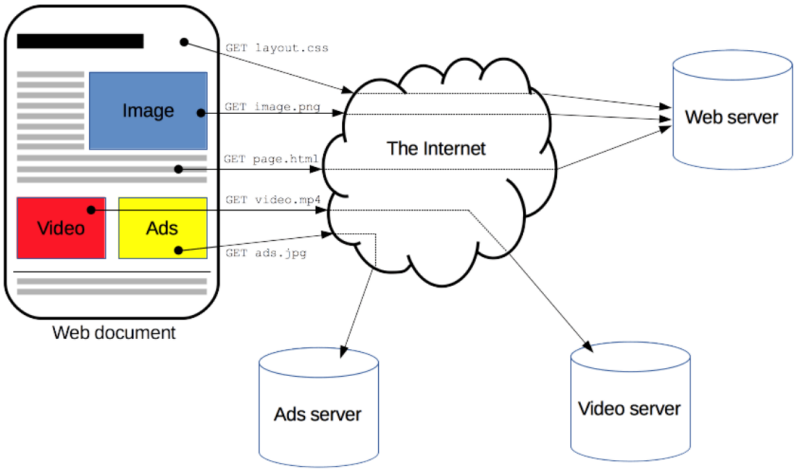
Key	Value
auth-token	eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJQZXRhciIsImV4cCI6MTYzMTAxNDQ0OSwiaW...

```
1 eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJQZXRhciIsImV4cCI6MTYzMTAxNDQ0OSwiaWF0IjoxNjMxMDEyNjQ5fQ.-RkXPg_sAM
```

Slika 4.13. Token spremnjen u lokalnoj memoriji

4.1.5. Komunikacija sa serverom

Sva komunikacija sa serverom odvija se preko HTTP(Hypertext Transfer Protocol) protokola. On služi za prijenos informacija od servera do klijenta. Arhitektura koja se koristi naziva se klijent-server te je prikazana na slici 4.14.



Slika 4.14. Klijent-server arhitektura

Klijent šalje zahtjev serveru u obliku HTTP poziva koji aktivira određenu akciju na serveru. Kako bi server znao koju akciju aktivirati, poziv mora sadržavati određene parametre. Primjer HTTP poziva dan je na slici 4.15 gdje se šalje zahtjev za dobivanje svih postova. Definira se HTTP metoda (GET) te ruta koja se poziva. Ruti je moguće dodavati i parametre. Osim navedenog, kada želimo dodati podatke u bazu koristi se metoda POST koja onda zahtjeva da se preda određeni objekt.

```
fetchPosts(): void {
    this.http.get( url: Urls.POST_API()+'all').subscribe(
        next: (response: Post[]) => {
            this.postService.loadedPosts = response
        }
    )
}
```

Slika 4.15. Primjer slanja GET metode

Svakom zahtjevu može se dodati i zaglavlje koje sadrži podatke o zahtjevu. U ovoj aplikaciji je korišten HTTPInterceptor sučelje koje definira metodu za presretanje zahtjeva. Svakom zahtjevu je u zaglavlje dodana informacija o autorizaciji gdje je predan token. Primjer presretača HTTP zahtjeva dan je na slici 4.16.

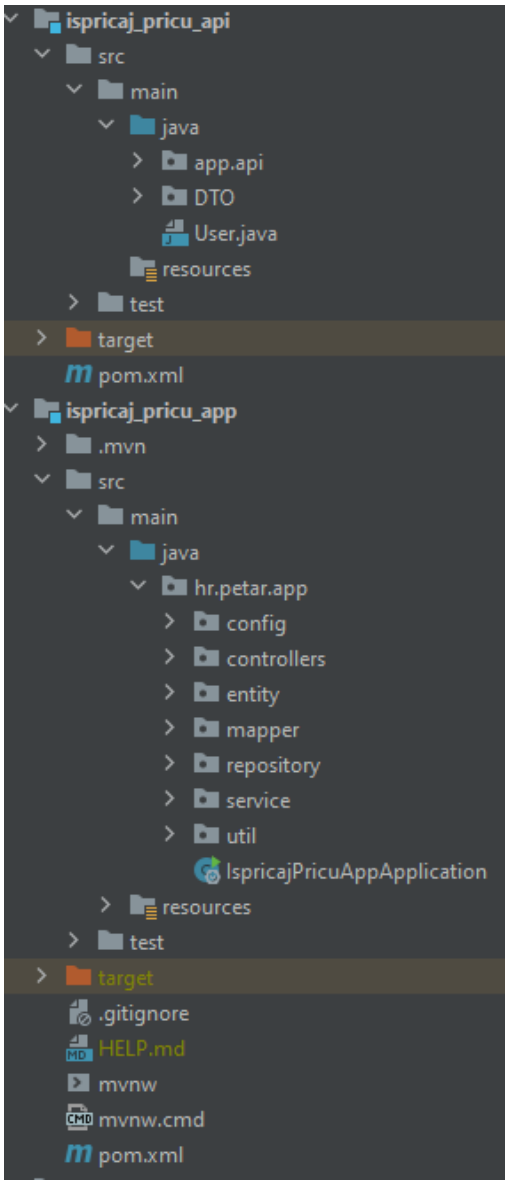
```
intercept(req: HttpRequest<any>, next: HttpHandler) {
    const authToken = this.tokenStorageService.getToken()

    if (authToken) {
        return next.handle(
            req.clone( update: {
                headers: req.headers.set('Authorization', `Bearer ${authToken}`),
            })
        )
    }
    return next.handle(req.clone())
}
```

Slika 4.16. Primjer presretača koji dodaje zaglavlje svakom zahtjevu

4.2. Izrada serverske strane

Kao što je već rečeno, za izradu serverske strane korišten je Spring Boot koji koristi Javu za implementaciju svojih funkcionalnosti. Struktura serverskog dijela dana je na slici 4.17. Modul „ispricaj_pricu_api“ sastoji se od sučelja koji definiraju rute i funkcije koje se pokreću kada se na klijentskoj strani pozovu. Sve one su prepisane (engl. *Override*) su unutar modula „ispricaj_pricu_app“ unutar datoteke „controllers“ te su im implementirane stvarne funkcionalnosti. Osim sučelja, u „ispricaj_pricu_api“ modulu su definirani tipovi podataka koje klijentska strana šalje i oni koji se vraćaju klijentu nakon obrade zahtjeva.



Slika 4.17. Struktura serverske aplikacije

Config datoteka u sebi sadrži klase koje se koriste za kriptiranje zaporke i klasu koja provjerava predani token iz zaglavlja.

Ako token ne odgovara pravilima po kojima je izrađen, odnosno ukoliko netko pokuša upisati lažni token, sustav ga prepoznaje te dolazi do errora.

Klasa za kriptiranje implementira dvije metode, jedna koja kriptira predani string i jedna koja uspoređuje kriptiranu zaporku s normalnim tekstom. Potonja se koristi prilikom prijave korisnika u aplikaciju dok se kriptiranje koristi prilikom registracije korisnika.

4.2.1. Kontroleri

Kontroler je klasa koja sadrži metode koje se izvršavaju kada se određeni API (engl. *Application Programming Interface*) pozove. Odnosno služi za procesiranje zahtjeva i vraćanje odgovora na zahtjev klijentske strane. Spring implementira kontrolere na vrlo apstraktan način. [14]

Da bi obična komponenta postala kontroler potrebno je dodati anotaciju @Controller i @RestController.

Svaki kontroler u aplikaciji implementira sučelje iz module „ispricaj_pricu_api“ koji okidaju njegove metode. Primjer ruta koje se koriste za dohvaćanje, objavljivanje, ažuriranje postova itd. dan je na slici 4.18.

```
@RestController
@RequestMapping(PostAPI.BASE_URL)
public interface PostAPI {

    String BASE_URL = "/api/post/";

    @GetMapping(value = "/all")
    ResponseEntity<List<PostReturnDTO>> getPosts();

    @GetMapping(value = "/user/{username}")
    ResponseEntity<List<PostReturnDTO>> getPostsByUser(@PathVariable("username")String username) throws Exception;

    /PMD.SignatureDeclareThrowsException/
    @GetMapping(value = "/{id}")
    ResponseEntity<PostDTO> getPost(@PathVariable("id")long id) throws Exception;

    @PostMapping(value = "/")
    ResponseEntity<PostDTO> createPost( @RequestHeader(value = "Authorization") String authHeader, @RequestBody @Valid PostDTO personDTO);

    /PMD.SignatureDeclareThrowsException/
    @PutMapping(value = "/")
    ResponseEntity<PostDTO> updatePost(@RequestBody @Valid PostDTO personDTO) throws Exception;

    /PMD.SignatureDeclareThrowsException/
    @DeleteMapping("/{id}")
    String deletePost(@PathVariable int id);
}
```

Slika 4.18. Sve korištene rute za postove

Kada zahtijevana ruta odgovara jednoj od zadanih ruta pokreće se jedna od metoda koje su implementirane u kontroler klasi. Npr. Ako je zahtijevana ruta /api/post/all pokreće se metoda getPosts vidljiva na slici 4.19.

```java
@CrossOrigin(origins = "http://localhost:4200")
@Slf4j
@RestController
public class PostController implements PostAPI {

    private final PostService postService;
    private final PostMapper postMapper;
    private JwtUtil jwtUtil;
    private UserService userService;

    public PostController(PostService postService, PostMapper postMapper,
                          JwtUtil jwtUtil,
                          UserService userService) {...}

    @Override
    public ResponseEntity<List<PostReturnDTO>> getPosts() {
        List<PostReturnDTO> postReturnDTO = postMapper.toListPostReturnDTO(postService.findAll());
        return ResponseEntity.ok(postReturnDTO);
    }
```

Slika 4.19. PostController klasa

4.2.2. Servisi

Servisi su klase koji rukuju podacima u drugom sloju aplikacije, nakon kontrolera. Da bi klasa postala servis potrebno joj je dodati anotaciju @Service.

Servisi u nešto manjim aplikacijama samo prosljeđuju zahtjev do „repository“ klase. Nakon što je findAll() metoda pozvana u kontroleru ona poziva metodu koja vraća sve postove što je prikazano na slici 4.20.

```java
@Override
public List<Post> findAll() { return postRepository.findAllByOrderByCreatedDateDesc(); }
```

Slika 4.20. Metoda findAll u servisu

Osim da samo prosljeđuje zahtjev ona se nalazi između kontrolera i repozitorija te tako može dohvaćati i druge podatke. Primjer na slici 4.21. pokazuje kako prilikom kreiranja posta najprije dohvaćamo korisnika pod određenim ID-om te ga nakon toga dodajemo post objektu.

```java
@Override
public Post createPost(Post post,int author_id) {

    User user = userService.findById(author_id);

    post.setUser(user);

    return postRepository.save(post);
}
```

Slika 4.21. Kreniranje posta

4.2.3. Repozitorij

Repozitorij je sučelje koje sadrži mehanizme koji objedinjuju spremanje, dohvaćanje, pretraživanje po različitim parametrima i slično.

Da bi sučelje bilo repozitorij potrebno je dodati anotaciju @Repository.

Izgled cijelog PostRepository sučelja prikazan je na slici 4.22.

```
public interface PostRepository extends JpaRepository<Post, Integer> {
    List<Post> findAllByOrderByCreatedDateDesc();
}
```

4.22. PostRepository sučelje

Repository je vrlo apstraktna komponenta i puno toga Spring Boot obavlja umjesto razvojnog programera. Također, između repozitorija i baze podataka nalazi se Hibernate koji modificira podatke tako da odgovaraju tablici odnosno klasi.

4.2.4. Mapperi

Mapperi su klase koje služe za prilagođavanje podataka koje šaljemo ili primamo s klijentske odnosno serverske strane.

Koristi se u kontrolerima gdje se prilikom spremanja podatka objekt koji se šalje mapira tako da odgovara entitetu. Također prilikom dohvaćanja podataka iz baze, koje dobivamo u obliku entiteta, mapiramo ih kako bi se na klijentsku stranu vratilo samo ono što zanima klijenta

4.2.5. JWT token

JWT token (JSON Web Token) se već spominjao u kontekstu prijave korisnika. Token je oznaka koja sadrži informacije. Generiranje tokena i njegova provjera obavljaju se na serverskoj strani, dok klijentska strana prima samo njegovu vrijednost.

Svaki token kriptira jednu vrijednost korisnika (username) prema privatnom ključu.

Kada se korisnik prijavljuje, ako je prijava uspješna, na temelju emaila i zaporke nalazi se njegov username te se kriptira.

Prilikom svakog pristupa serveru, sigurnosni mehanizmi provjeravaju token koji se nalazi u zaglavlju zahtjeva, ukoliko je token lažno generiran podaci neće biti isporučeni.

4.3. Baza podataka

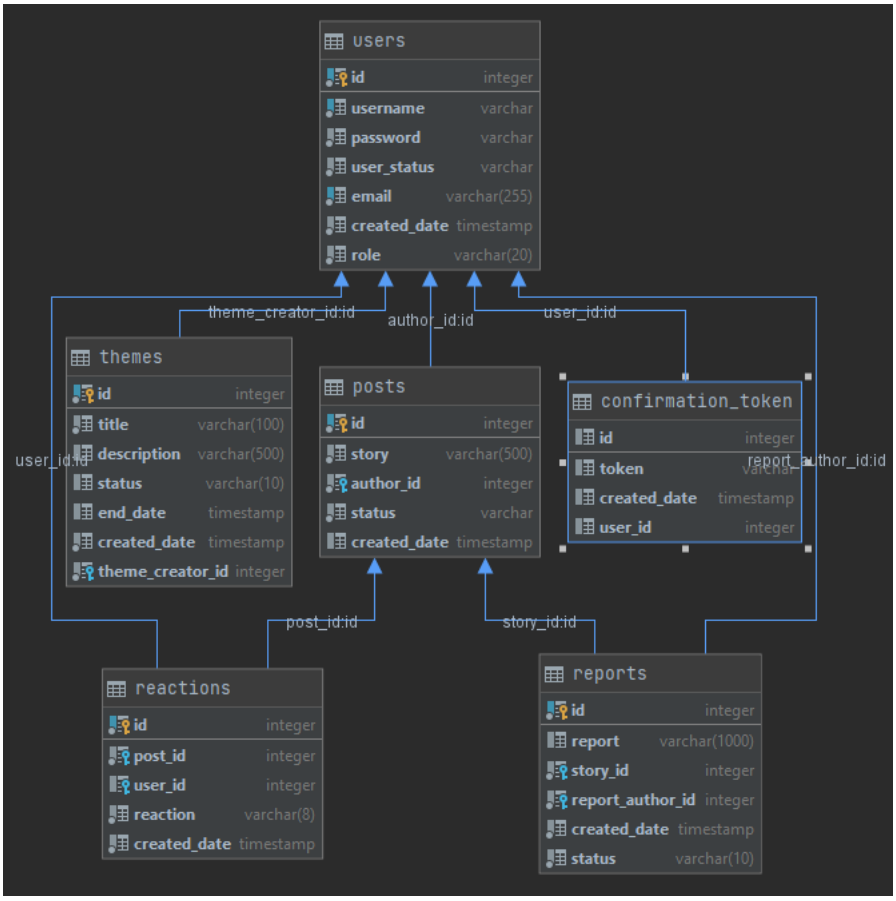
Baza podataka je nezamjenjivi dio svake veće aplikacije. Gotovo sve legitimne aplikacije na tržištu posjeduju jednu ili čak više baza. One su bitne radi prikupljanja i organizacije podataka. Bez takvog sustava svaki puta kada bi neka aplikacije bila ugašena izgubili bi se podaci.

Prije same izrade serverske strane potrebno je definirati tablice zajedno s njihovim atributima i ograničenjima koji svaki od njih nosi.

Baza koja je odabrana za ovu aplikaciju je PostgreSQL koja je ranije opisana.

Svaka tablica predstavlja jedan entitet koji može biti povezan s drugim preko stranog ključa. Entiteti baze su *confirmation_token*, *posts*, *reactions*, *reports*, *themes*, *users*.

Relacije između tablica odnosno entiteta prikazane su na slici 4.23.



Slika 4.23. Dijagram baze podataka

5. IZGLED APLIKACIJE

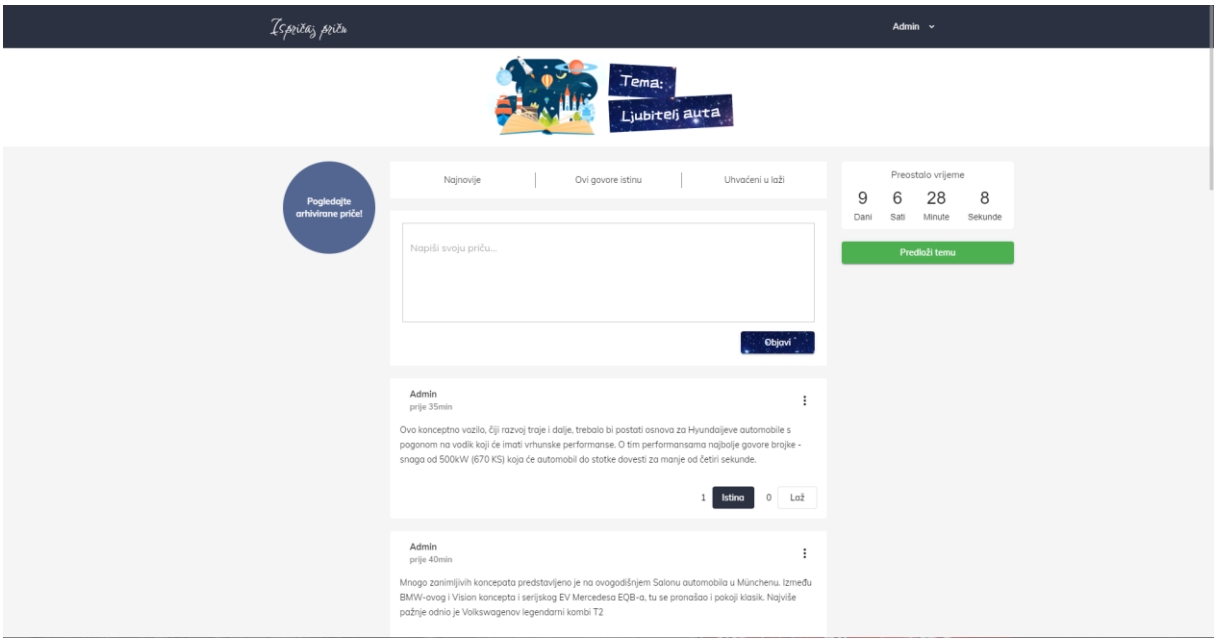
Ulaskom u aplikaciju otvara se stranica za prijavu vidljiva na slici 5.1. Ukoliko korisnik nije prijavljen najprije se mora registrirati što je prikazano na slici 5.2.

Slika 5.1. Stranica za prijavu

Slika 5.2. Stranica za registraciju

Sa stranice za prijavu moguće je otvoriti stranicu za registraciju i obratno.

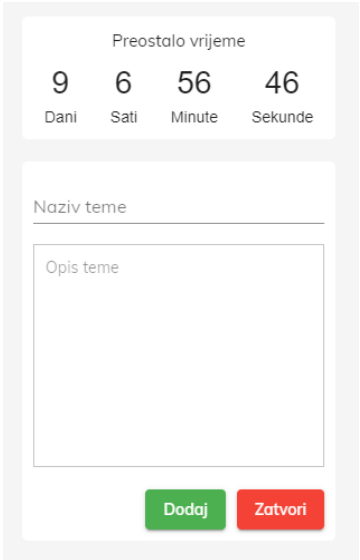
Obje stranica ako su podaci uspješno uneseni preusmjeravaju na naslovnu stranicu na kojoj su vidljive priče. Naslovna stranica nalazi se na slici 5.3.



Slika 5.3. Naslovna stranica

Naslovna stranica sastoji se od više dijelova. Na vrhu stranice nalazi se tema o kojoj korisnici trenutno pišu. Ispod teme zadane teme nalaze se opcije koje sortiraju polje po reakcijama na priču.

S desne strane vidi se mjerač vremena koji prikazuje koliko je vremena još ostalo do promjene teme. Ispod njega nalazi se gumb koji nakon pritiska otvara formu za prijedlog teme. Forma je prikazana na slici 5.4.

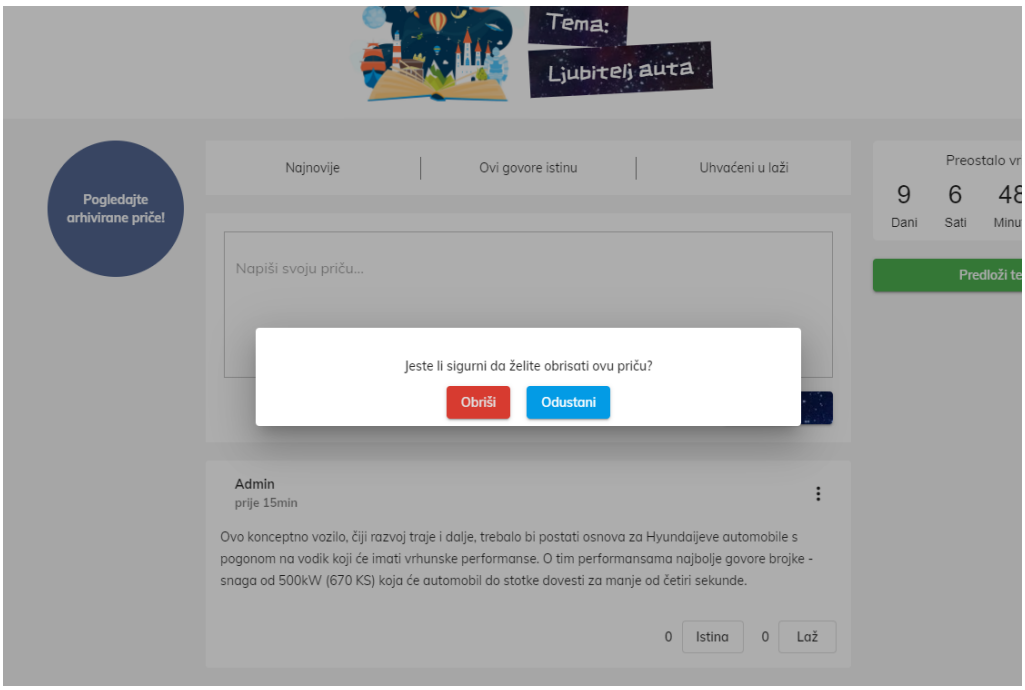


Slika 5.4. Forma za prijedlog teme

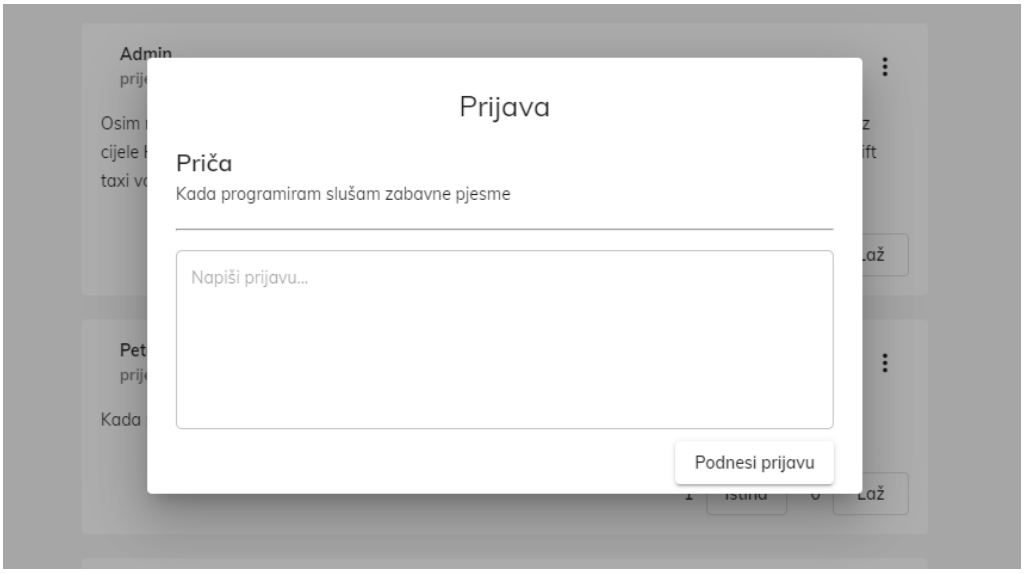
Glavni dio stranice sastoji se od dijela u koje korisnik može upisati priču za zadanu temu. Prije ranije napisane izlistane su na stranici. Priča se sastoji od imena autora, vremena koje je prošlo

od njezine objave, teksta te gumbova za reakciju. Korisnik može imati samo jednu reakciju na priču, a reakcija se briše tako da se ponovno klikne na odabrani gumb.

Svaku priču moguće je prijaviti, a ako je korisnik autor priče može ju obrisati. Dijalozi za brisanje i prijavu prikazani su na slikama 5.5. i 5.6.



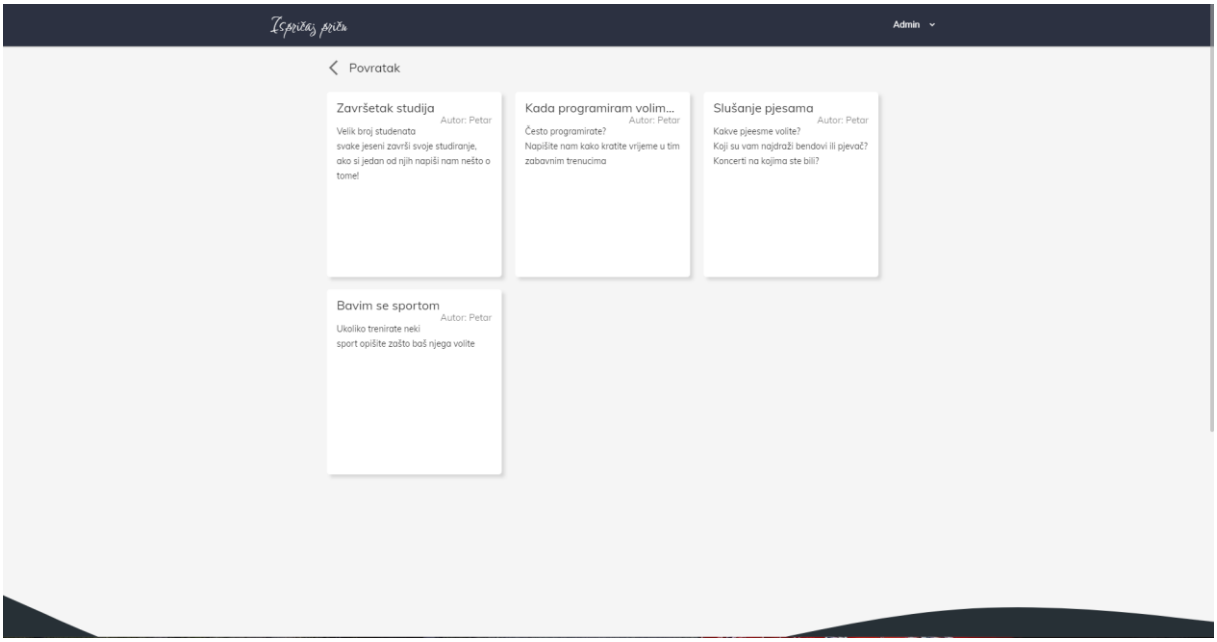
Slika 5.5. Dijalog za brisanje priče



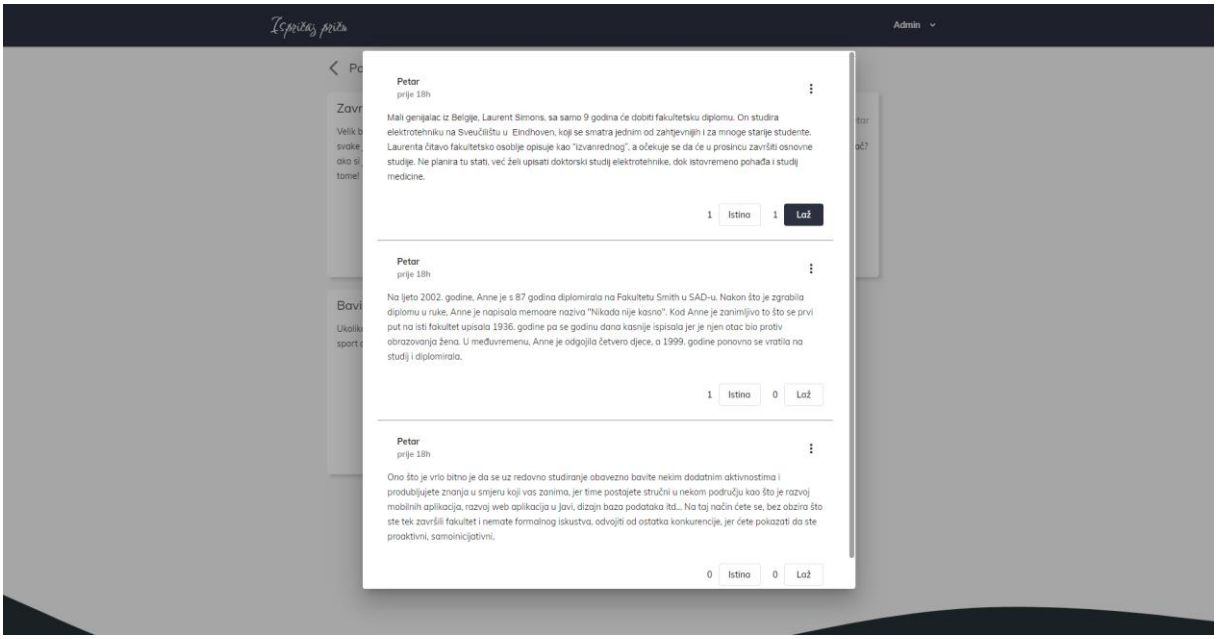
Slika 5.6. Dijalog za prijavu priče

Dijalog za prijavu priče sastoji se od njenog teksta te prostora u koji je moguće upisati u čemu je problem. Nakon što se prijava podnese šalje se administratoru koji pregledava i ažurira status.

Na lijevoj strani ekrana nalazi se gumb „Pogledajte arhivirane priče“ koji otvara stranicu s arhiviranim temama. Stranica se vidi na slici 5.7. Odabirom na pojedinu temu prikazuju se sve priče koje su obavljene kada je ona bila aktualna. Popis je prikazan na slici 5.8.

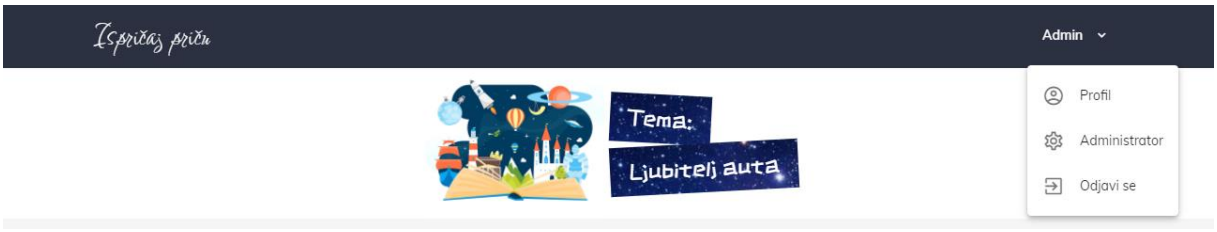


Slika 5.7. Arhivirane teme



Slika 5.8. Priče povezane s temom

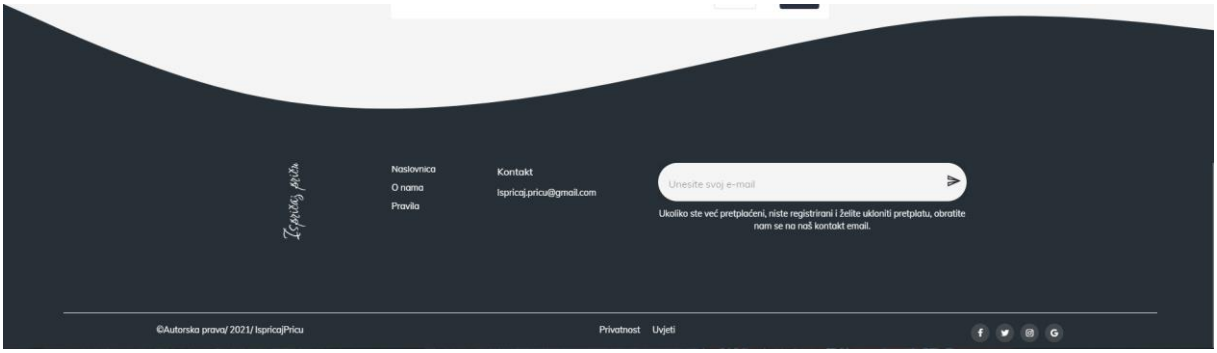
Svaka stranica nakon prijave i registracije sastoji se od navigacijske trake i podnožja. Navigacijska traka na lijevoj strani ima ime aplikacije koji klikom na njega preusmjerava na naslovnu stranicu dok se na desnoj ima padajući izbornik. Navigacija je prikazana na slici 5.9.



Slika 5.9. Navigacijska traka

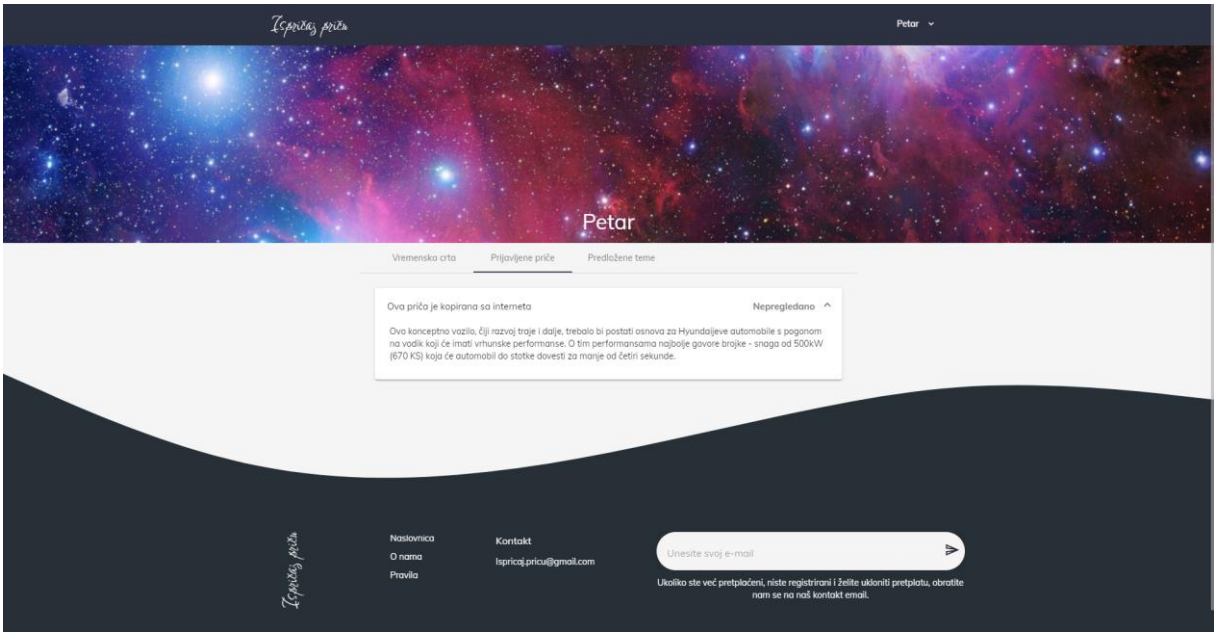
U padajućem meniju stavku administrator mogu vidjeti samo korisnici kojima je uloga „ADMIN“ dok svi ostali vide samo stavke za profil i odjavu

Podnožje aplikacije je nešto veće i sadrži bitne informacije o aplikaciji. Vidljivo je na slici 5.10.

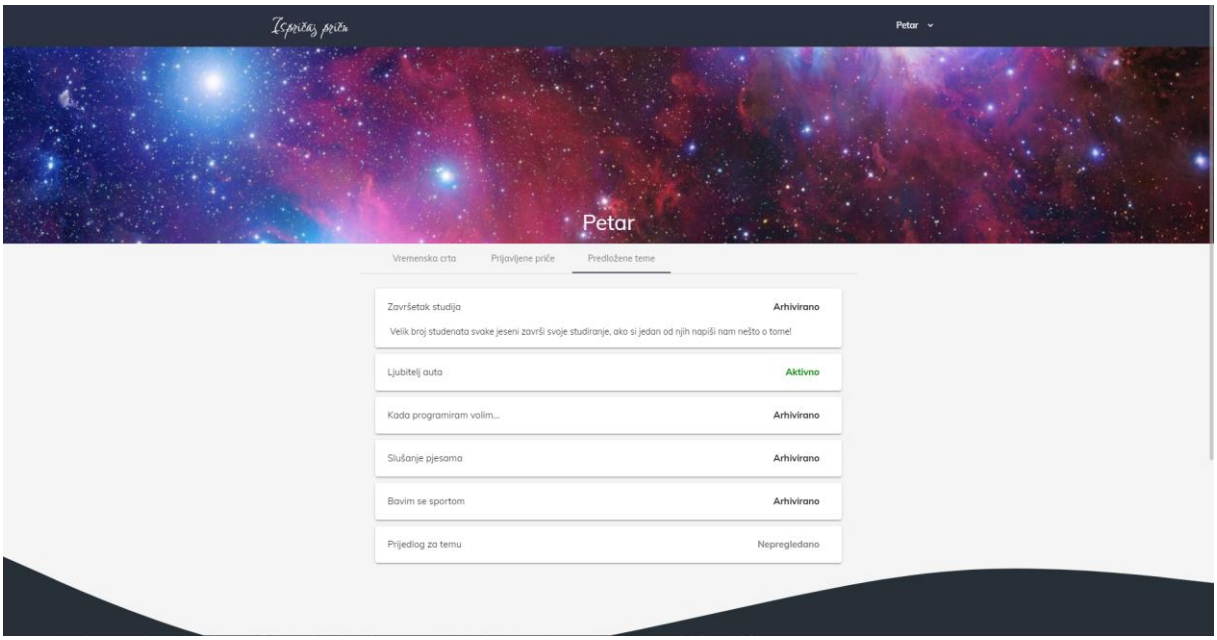


Slika 5.10. Podnožje aplikacije

Kao što se vidi na slici 5.9. svaki korisnik može posjetiti svoj profil. Na profilu se nalaze sve priče koje je objavio, ali također, kao i na naslovnoj može dodati novu priču. Osim toga može vidjeti status prijavljenih tema kao i status predloženih tema. Statusi prijavljenih priča kao i statusi predloženih tema prikazani su na slikama 5.11. i 5.12.



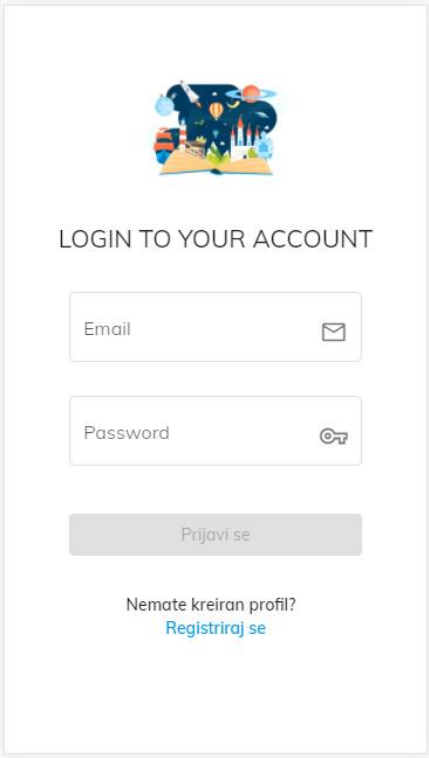
Slika 5.11. Prijava priče te njezin status



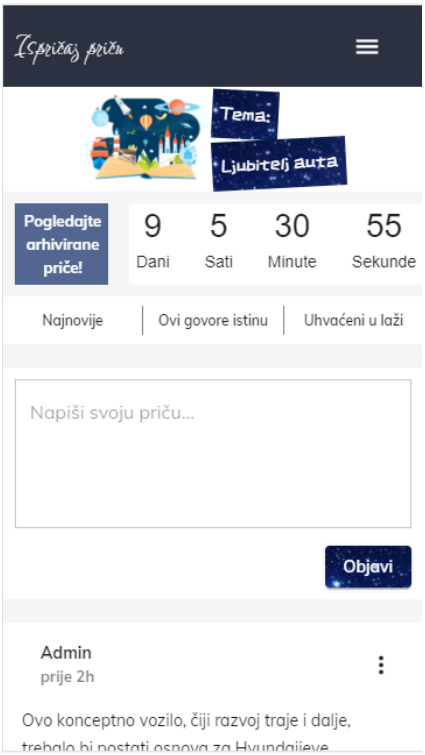
Slika 5.12. Predložena tema i njezin status

5.1. Responzivnost

Prikaz svakog ekrana prilagođen je različitim veličinama ljudi. U nastavku, na slici 5.13., prikazani su ekrani na dimenzijama uređaja Samsung Galaxy S5.



Slika 5.13. Ekran za prijavu korisnika



Slika 5.14. Naslovna stranica

6. ZAKLJUČAK

Internetskih korisnika svakim je danom sve više. Osim informativnog, obrazovnog i sl. sadržaja, vrlo je bitno imati stranice bitne za opuštanje i zabavu. Kao što su internetske igre, video sadržaj i sličan materijal, jedan od vidova zabave može biti i čitanje kratkih priča. Osim što aplikacija ima svrhu obogatiti sadržaj na internetu, služi za izvrstan alat za unaprjeđivanje tehnika pisanja jer je bitno što bolje opisati i dočarati priču s ograničenim brojem znakova.

Opisane su upotrebljavane tehnike kao i dijelovi koda bitni za prikaz i obavljanje funkcionalnosti. Aplikacija se sastoji od klijentskog i serverskog dijela koji svaki obavlja svoju svrhu. Na klijentskom dijelu pažljivo su birani elementi, boje kao i animacije koje na korisnike ostavljaju bolji dojam.

Ovaj diplomski rad početak je izrade ideje čija svrha je bila praćena kod njegove izrade te je još mnogo prostora za napredak i usavršavanje. Primjerice, moguće je proširiti portal na sekcije te svakom korisniku omogućiti izdavanje mini-romana. Također, mogla bi se uvesti funkcionalnost praćenja omiljenih korisnika čime bi se omogućilo dobivanje obijesti po objavljivanju priče.

LITERATURA

[1] „10 INTERNET STATISTICS EVERY MARKETER SHOULD KNOW IN 2021," [Online]. Dostupno: https://www.oberlo.com/blog/internet-statistics. [Srpanj 2021].

[2] „3 stranice na kojima možete besplatno čitati knjige," [Online]. Dostupno: https://www.fashion.hr/kultura/knjige/3-stranice-na-kojima-mozete-besplatno-citatiknjige-141421.aspx. [Srpanj 2021].

[3] „Inkspired- About Us," [Online]. Dostupno: https://getinkspired.com/en/aboutinkspired/. [Rujan 2021].

[4] „SPA (Single-page application)," [Online]. Dostupno: https://developer.mozilla.org/enUS/docs/Glossary/SPA. [Rujan 2021].

[5] „Component," [Online]. Dostupno: https://angular.io/api/core/Component. [Rujan 2021].

[6] „What is Angular," [Online]. Dostupno: https://angular.io/guide/what-is-angular. [Rujan 2021].

[7] A. Rai, „Angular Dependency Injection," [Online]. Dostupno: https://www.concretepage.com/angular/angular-dependency-injection. [Rujan 2021].

[8] „Injectable," [Online]. Dostupno: https://angular.io/api/core/Injectable. [Rujan 2021].

[9] „Angular Routing," [Online]. Dostupno: https://angular.io/guide/routing-overview. [Rujan 2021].

[10] „HTML: HyperText Markup Language," [Online]. Dostupno: https://developer.mozilla.org/en-US/docs/Web/HTML. [Rujan 2021].

[11] „Difference between Spring and Spring Boot," [Online]. Dostupno: https://www.geeksforgeeks.org/difference-between-spring-and-spring-boot/. [Rujan 2021].

[12] „Java Programming," [Online]. Dostupno: https://www.programiz.com/java-programming. [Rujan 2021].

[13] „Why use PostgreSQL as a Database for my Next Project in 2021," [Online]. Dostupno: https://fulcrum.rocks/blog/why-use-postgresql-database/. [Rujan 2021].

[14] „Implementing Controllers," [Online]. Dostupno: https://docs.spring.io/springframework/docs/3.0.0.M4/reference/html/ch15s03.html. [Rujan 2021].

SAŽETAK

Zadatak rada bio je izrada portala za kratke priče.

Prikazane su postojeće aplikacije koje se mogu naći na internetu, a imaju sličnu svrhu.

Objašnjene su tehnologije korištene za razvoj aplikacije: Angular programski okvir, opisni jezik HTML, jezik za stiliziranje CSS-a te programski jezik TypeScript na klijentskoj strani. Što se tiče serverske strane korištene tehnologije su Spring Boot koji koristi programski jezik Javu. Kao baza podataka korišten je PostgreSQL.

Aplikacija omogućuje postavljanje priča kao i reagiranje na druge priče. Svi registrirani korisnici mogu davati svoje prijedloge za teme koje će se postaviti u budućnosti kao i prijavljivati druge priče.

Korisnici mogu imati ulogu administratora ili običnog korisnika. Administrator nasljeđuje sve funkcionalnosti običnog korisnika te upravlja podacima i odlučuje koja će sljedeća tema biti postavljena te koliko će dugo trajati.

Ključne riječi: Angular, PostgreSQL, priča, Spring Boot, TypeScript

ABSTRACT

Portal for short stories.

The task of the thesis was to create a portal for short stories.

There are shown existing applications that can be found on the Internet and have a similar purpose.

Explained technologies used to develop the application are: Angular framework, HTML markup language, CSS styling language, and client-side programming language TypeScript. Used server side technologies are Spring Boot which uses the Java programming language. PostgreSQL is used as the database.

The application allows all users to publish their stories as well as reacting to other stories. All registered users can send their suggestions for topics to be set in the future as well as report other stories.

Users can be the administrator or regular user. The administrator inherits all the functionality of the regular user, manages the data and decides which topic will be set and how long it will last.

Keywords: Angular, PostgreSQL, story, Spring Boot, TypeScript

ŽIVOTOPIS

Petar Pejić rođen je 13. svibnja 1997. godine u Vinkovcima, Hrvatska. Svoje školovanje započinje 2004. godine u Osnovnoj školi Antuna Gustava Matoša, Vinkovci. Nakon završetka upisuje se na elektrotehnički smjer u Tehničku školu Ruđera Boškovića Vinkovci u Vinkovcima. Svoje školovanje nastavlja na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek te odabire smjer računarstvo. Nakon uspješnog završetka preddiplomskog studija odrađuje studentske prakse u Informatika Fortuno te Hrvatski telekom. Na zadnjoj godini diplomskog studija postaje stipendist tvrtke DICE Digital Innovation Center d.o.o.

PRILOZI

Na CD-u priloženom uz diplomski rad nalaze se dokumenti:

Diplomski-rad_PetarPejic.pdf

Diplomski-rad_PetarPejic.doc

Ispricaj.pricu.parent.rar