SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni prijediplomski studij Informacijska tehnologija

JOSIP ČONDIĆ JURKIĆ

Z A V R Š N I R A D

IZRADA WEB APLIKACIJE ZA OBJAVU I PRETRAGU OGLASA

Split, rujan 2024.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni prijediplomski studij Informacijska tehnologija

Predmet: Programiranje na internetu

Z A V R Š N I R A D

Kandidat: Josip Čondić Jurkić

Naslov rada: Izrada web aplikacije za objavu i pretragu oglasa

Mentor: Marina Rodić, viši predavač

Split, rujan 2024

Sadržaj

Sažetak

U ovom radu opisana je izrada aplikacije za objavu i pretragu oglasa. Aplikacija je rađena u radnom okviru (engl. *framework*) *Django*. Korisničko sučelje stilizirano je pomoću knjižnice *Tailwind*, te knjižnice *React*. U ovoj aplikaciji korištena je baza podataka *SQLite*. U aplikaciji postoje dvije vrste korisničkih uloga, a to su korisnik i administrator. Cilj projekta je napraviti funkcionalnu aplikaciju koja se može koristiti kao smislena cjelina te omogućuje sve osnovne usluge potrebne za objavu i pretragu oglasa. Osim osnovnih usluga za objavu i pretragu oglasa, aplikacija omogućuje izradu i ažuriranje korisničkih računa, ažuriranje oglasa, spremanje oglasa u favorite, objavljivanje komentara na korisničke profile, te pregled statistike o korisnicima i oglasima putem administratorskog sučelja.

Ključne riječi: *django, mrežna aplikacija, oglas*

Summary

Advertising web application

This work describes creation of an application for posting and browsing advertisements. It was developed using the *Django* framework. The user interface was designed with *Tailwind* framework and *React* library. This application utilizes *SQLite* database. The application features two types of user roles: user and administrator. The goal of this project is to make a functional application that can be used as a coherent unit, providing all the essential services needed for posting and browsing advertisements. In addition to the basic features for posting and browsing advertisements, the application allows for creating and updating user accounts, updating advertisements, saving advertisements as favorites, posting comments on user profiles, and reviewing user and advertisement statistics through the administrator interface.

Keywords: *advertisement, django, web application*

1. Uvod

Aplikacija koja je opisana u ovom radu nudi platformu gdje korisnici mogu objavljivati i pretraživati oglase, dok neprijavljeni korisnici imaju samo mogućnost pretraživanja oglasa. Aplikacija ima dvije uloge: korisnik, koji može objavljivati i pretraživati oglase, te administrator koji je zadužen za tehničku podršku. U današnje vrijeme, aplikacije za objavu i pretragu oglasa su neophodne. Nekada su ljudi oglase objavljivali u novinama ili fizički tražili usluge i proizvode. Danas se većina oglasa objavljuje i pretražuje putem interneta, a aplikacije omogućuju jednostavniju i bržu komunikaciju između kupaca i prodavača. Kao motivacija za izradu ove aplikacije služilo je i to što autor osobno koristi slične platforme za pretragu oglasa.

U prvom poglavlju opisuje se sustav za upravljanje bazama podataka. To je jedan od prvih koraka pri planiranju projekta, jer ona definira kako se podaci mogu spremati i čitati, ograničavajući mogućnosti aplikacije. U drugom poglavlju detaljno su opisane mogućnosti korisničkog sučelja. Sučelje korisnika i administratora u velikoj je mjeri identično, s iznimkom što administrator, uz standardne funkcionalnosti dostupne korisnicima, ima i pristup dodatnom administratorskom sučelju koje mu omogućuje upravljanje i nadzor nad sustavom. U trećem poglavlju opisano je administratorsko sučelje. Administrator ima pravo uređivati oglase i korisničke profile, te može mijenjati status oglasa u jedan od sljedećih: aktivan, neaktivan ili arhiviran. Također, administrator ima ovlast brisati oglase, te uskratiti pravo prijave korisnika. U zaključku su izloženi rezultati ovog rada te su navedeni prijedlozi za moguća poboljšanja i daljnje usavršavanje. Također se osvrće na programski okvir *Django* i dojam o njegovim značajkama i iskustvu stečenom tijekom izrade projekta.

2. Korištene tehnologije

2.1. Django

Apliacija je razvijena koristeći *Django* radni okvir, koji je pisan u programskom jeziku *Python*. *Django* automatizira mnoge aspekte razvoja, čime se smanjuje potreba za ručnim pisanjem kôda, te se smanjuje mogućnost grešaka. Ova automatizacija ne samo da ubrzava razvoj aplikacije, već olakšava rad dizajnerima mrežnih stranica, jer smanjena količina kôda u HTML datotekama (engl. *HyperText Markup Language*, tj. prezentacijski jezik za izradu mrežnih stranica) smanjuje rizik od problema prilikom izmjena. *Django* također upravlja zadacima kao što su povezivanje s bazom podataka i upravljanje podacima, čime se dodatno olakšava razvoj na poslužiteljskoj strani.

Django koristi *MVT* arhitekturu, koja se sastoji od Modela, Pogleda i Predložaka (engl. *Model*, *View*, *Template*). U datoteci `models.py`, definiraju se modeli aplikacije, gdje svaka klasa odgovara jednoj tablici u bazi podataka. Ove klase definiraju stupce i njihove atribute, kao što su dopuštenja za prazna polja i tipovi podataka. Na taj način, Django uklanja potrebu za ručnim pisanjem *upita SQL-a* (engl. *Structured Query Language*, tj. strukturni upitni jezik). *Django* omogućuje programerima da rade s bazom podataka koristeći *Python* kôd.

Ključne funkcionalnosti aplikacije smještene su u datoteku `views.py`, koja upravlja prikazom i obradom podataka. Ovdje *Django* povezuje poslovnu logiku s korisničkim sučeljem te omogućuje interakciju putem *API-a* (engl. *Application Programming Interface*, tj. sučelje za programiranje aplikacija), što pojednostavljuje rukovanje podacima i njihov prikaz u pregledniku.

Još jedan važan element *Django* projekta je datoteka `urls.py`, koja definira mapiranje između URL poveznica (engl. *Uniform Resource Locator*, tj. jedinstveni lokator sadržaja) i funkcija poslovne logike. Ova datoteka određuje koja će funkcija biti pozvana kada korisnik posjeti određenu poveznicu.

Zahvaljujući *MVT* arhitekturi, *Django* omogućuje fleksibilan razvoj, gdje se moduli mogu neovisno razvijati, testirati i održavati. Na taj način promjene u jednoj komponenti ne utječu na ostatak aplikacije. Ovo doprinosi bržem i učinkovitijem razvoju, kao i lakšoj održivosti u budućnosti [1].

2.2. Baza podataka SQLite

Django podržava bazu podataka *SQLite*. To je standardna baza podataka koja se koristi za pohranu podataka tijekom razvoja aplikacija. *SQLite* je izuzetno lagana, samostalna i besplatna relacijska baza podataka koja se često koristi u razvoju mrežnih aplikacija. Za razliku od većih sustava za upravljanje bazama podataka, *SQLite* ne zahtijeva instalaciju ili konfiguraciju zasebnog poslužitelja. Ova baza podataka podržava sve ključne funkcionalnosti koje očekujemo od relacijskih baza, uključujući transakcije, *ACID* svojstva (engl. *Atomicity, consistency, isolation, durability*, tj. valentnost, dosljednost, izolacija, trajnost) te integritet podataka [2].

U ovoj aplikaciji koristi se *SQLite 3*, najnovija verzija koja donosi poboljšanja u performansama i sigurnosti.

2.3. Tailwind

Tailwind CSS (engl. *Cascading Style Sheets*, tj. stilski jezik) je knjižnica koja omogućuje brzu i jednostavnu izradu modernih i profesionalnih mrežnih stranica. *Tailwind* se razlikuje od tradicionalnih pristupa stiliziranju stranica jer koristi male, višekratne *CSS* deklaracije koje se direktno primjenjuju na *HTML* elemente. Ovakav pristup omogućuje programerima veliku fleksibilnost i preciznost u oblikovanju elemenata. Na ovaj način, umjesto pisanja prilagođenih stilova za svaki element, programeri mogu brzo kombinirati *Tailwind* klase kako bi postigli željeni izgled sučelja.

Jedna od glavnih prednosti *Tailwind-a* je brzina razvoja. Programeri ne moraju gubiti vrijeme pišući prilagođeni *CSS*, već mogu jednostavno dodati odgovarajuće *Tailwind* klase iz unaprijed definiranog izbora. To smanjuje složenost kôda i ubrzava proces razvoja [3].

2.4. React

React je knjižnica stvorena za izradu dinamičkih korisničkih sučelja, posebno za mrežne aplikacije, a pisana je u programskom jeziku *JavaScript*. Glavna prednost ove knjižnice je u njenoj sposobnosti učinkovitog upravljanja stanjem aplikacije. *React* prati promjene unutar aplikacije i ažurira samo one dijelove sučelja koju su se promijenili, umjesto da osvježava cijelu stranicu. Ovaj pristup značajno poboljšava performanse, posebno kod složenih aplikacija.

Korištenjem komponenti kao osnovnih gradivnih elemenata, *React* omogućuje modularnost i ponovnu upotrebu kôda, što olakšava razvoj i održavanje aplikacija. Osim toga, *React* je vrlo fleksibilan i može se integrirati s raznim drugim alatima i knjižnicama [4].

3. Baza podataka i modeli

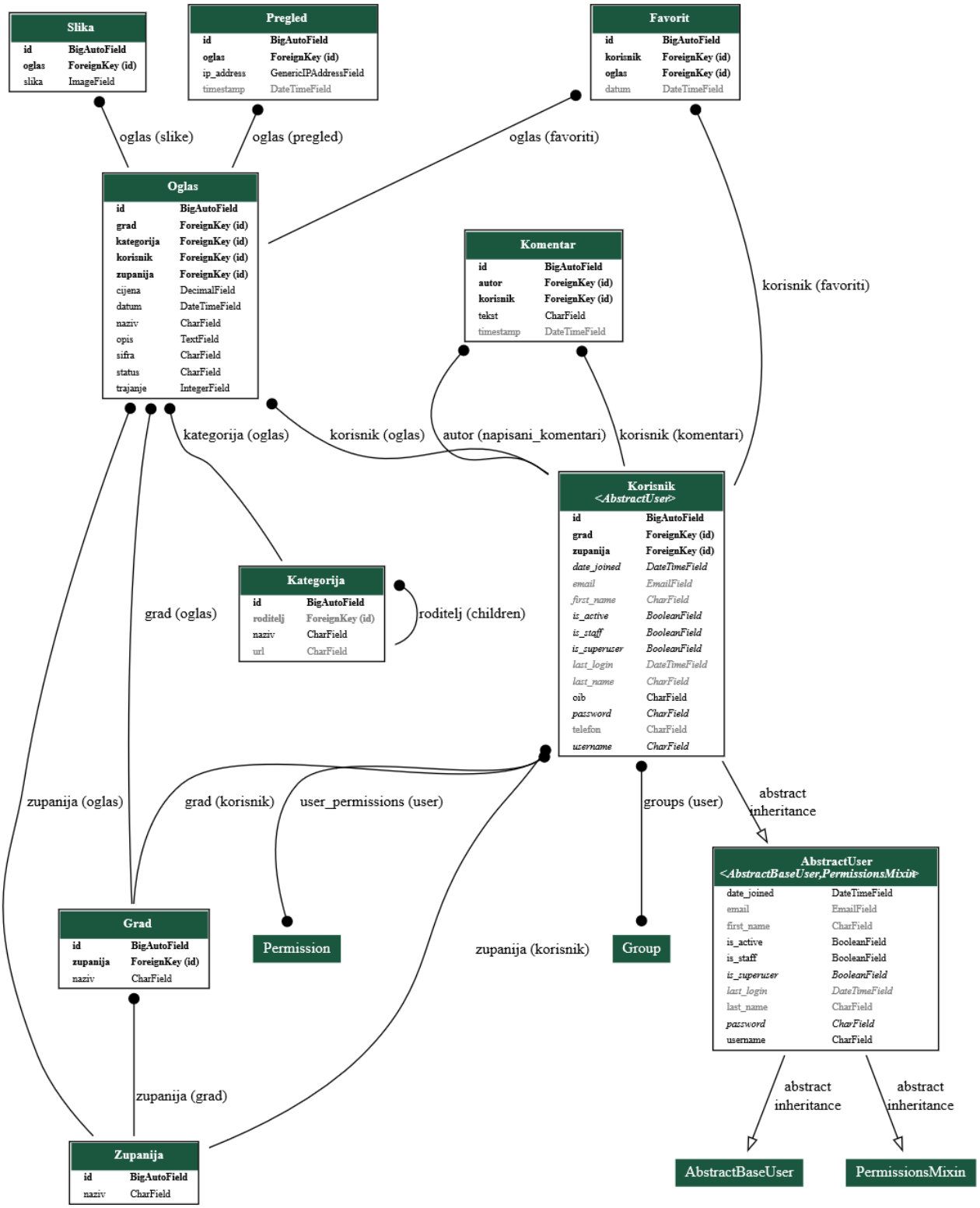
Sustavi za upravljanje bazama podataka služe za pohranu i pristup informacijama koje unose korisnici ili administratori. Većina aplikacija oslanja se na bazu podataka kako bi podaci bili trajno pohranjeni. Ovo je posebno važno u aplikacijama koje omogućuju korisnicima stvaranje korisničkih računa, jer se time osigurava čuvanje profila i pratećih podataka.

Baza podataka organizirana je u tablice, gdje su podaci raspoređeni u stupce. Svaki redak predstavlja jedan zapis, koji se može pregledavati, uređivati ili brisati. Tablice mogu biti međusobno povezane; primjerice, zapis o slici može biti povezan s oglasom kojem ta slika pripada. Kod oblikovanja baze podataka važno je osigurati pohranu svih potrebnih informacija za aplikaciju, ali i izbjeći pohranjivanje nepotrebnih podataka, kako bi se smanjila složenost i izbjegle suvišne tablice.

Osim pohrane i upravljanja podacima, sustavi za upravljanje bazama podataka važni su za optimizaciju rada aplikacija. Dobro organizirana baza podataka omogućuje brže pretraživanje i obradu podataka, što poboljšava korisničko iskustvo i smanjuje vrijeme čekanja. Povezivanjem tablica putem relacija postiže se učinkovito povezivanje različitih vrsta podataka, čime se izbjegava njihovo dupliciranje i nepotrebno zauzimanje prostora.

Modeli u *Django* radnom okviru predstavljaju temeljni element za rad s podacima. Svaki model je *Python* klasa koja odgovara tablici u bazi podataka. Unutar modela definiraju se atributi koji predstavljaju stupce te tablice, a svaki atribut ima svoj tip podataka, poput teksta, brojeva ili datuma. Ovaj model povezan je s bazom podataka, što znači da *Django* automatski stvara *upit SQL* za pohranu, ažuriranje i dohvaćanje podataka.

Baza podataka ove aplikacije (slika 1) sastoji se od slijedećih tablica: `Zupanija`, `Grad`, `Korisnik`, `Kategorija`, `Oglas`, `Pregled`, `Slika`, `Favorit` i `Komentar`.



Slika 1: Prikaz strukture baze podataka

3.1. Županija

Tablica `Zupanija` sadrži samo jedan stupac `naziv`, koji predstavlja naziv županije. Unošenje županija u bazu podataka previđeno je samo jednom, a oslanja se na *Python* skriptu `lokacije.py` (ispis 1). Ova skripta na temelju izrađenog *API-a* u *JSON* formatu (engl. *JavaScript Object Notation*) u bazu unosi županije i gradove.

```
        for entry in data:
            zupanija_name = entry['zupanija'].replace(' Županija',
''')
            zupanija, created =
Zupanija.objects.get_or_create(naziv=zupanija_name)
            if created:
                self.stdout.write(self.style.SUCCESS(f'Kreirana
zupanija: {zupanija}'))

            gradovi = entry['gradovi']
            for grad_entry in gradovi:
                grad_ime = grad_entry['grad']
                grad, created =
Grad.objects.get_or_create(naziv=grad_ime, zupanija=zupanija)
                if created:
                    self.stdout.write(self.style.SUCCESS(f'Kreiran
grad: {grad}'))
```

Ispis 1: Python skripta za unos županija i gradova

3.2. Grad

Tablica `Grad` sadrži stupce `zupanija` i `naziv`. Stupac `zupanija` je strani ključ koji povezuje grad s pripadajućom županijom u tablici `Zupanija`. Unošenje gradova i županija u bazu podataka izvršava se istovremeno. *API* na koji se ova aplikacija oslanja (ispis 2) strukturiran je tako da svaki grad pripada jednoj županiji, te svaki grad i županija imaju svoj *ID* (engl. Identifier, tj. jedinstveni identifikator) retka.

```
{
    "id": 0,
    "zupanija": "Grad Zagreb",
    "gradovi": [
      {
        "id": 0,
        "grad": "Zagreb"
      },
      {
        "id": 4,
        "grad": "Sesvete"
      },
      {
        "id": 222,
        "grad": "Lučko"
      },
      {
        "id": 253,
        "grad": "Hrvatski Leskovac"
      }
    ]
  }
```

Ispis 2: Struktura API-a za unos gradova i županija

3.3. Korisnik

Tablica `Korisnik` predstavlja proširenu verziju ugrađenog modela `AbstractUser`, koji se koristi za upravljanje korisničkim računima. Korištenje ove ugrađene klase omogućuje programerima da izrade vlastiti korisnički model s dodatnim poljima ili promjenama, bez potrebe za pisanjem cijelog korisničkog modela iz temelja. Na taj način, programeri mogu prilagoditi korisnički model specifičnim potrebama aplikacije, dok zadržavaju sve postojeće funkcionalnosti i sigurnosne značajke koje `AbstractUser` pruža.

Osim osnovnih atributa, `AbstractUser` također uključuje metode za provjeru ispravnosti lozinke i prijava korisnika.

Ovaj model, osim standardnih polja kao što su korisničko ime, lozinka i email, dodaje nekoliko specifičnih atributa za dodatne podatke o korisniku. Polje `oib` (osobni identifikacijski broj) pohranjuje jedanaest znamenki, uz obaveznu provjeru formata, kaki bi se osiguralo da OIB uvijek sadrži točan broj znamenki. Polje `zupanija` predstavlja vezu s tablicom `Zupanija` putem stranog ključa, koji definira da svaki korisnik pripada jednoj

županiji. Slično, polje `grad` povezuje korisnika s tablicom `Grad`. Polje `telefon` omogućuje pohranu telefonskog broja, s ograničenjem da telefonski broj ima između šest i dvanaest znamenki. Ovo polje nije obavezno prilikom unosa.

3.4. Kategorija

Tablica `Kategorija` predstavlja ključnu komponentu za organizaciju i kategorizaciju sadržaja unutar aplikacije. Ova tablica sadrži nekoliko važnih atributa koji omogućuju fleksibilno upravljanje kategorijama.

Prvi atribut je `naziv`, koji pohranjuje naziv kategorije. Atribut `roditelj` koristi vezu prema samom sebi, što omogućuje stvaranje hijerarhijske strukture kategorija. Ova funkcionalnost omogućuje da jedna kategorija bude podkategorija druge, čime se omogućuje detaljna organizacija sadržaja. Redak roditelj može biti prazan. U tom slučaju, radi se o glavnoj kategoriji koja je na vrhu hijerarhije.

Atribut `url` pohranjuje jedinstvenu poveznicu za svaku kategoriju, čime se olakšava pretraživanje i filtriranje oglasa ili drugih sadržaja prema određenoj kategoriji.

Unošenje kategorija u bazu podataka previđeno je samo jednom, a oslanja se na *Python* skriptu `kategorije.py` (ispis 3) koja na temelju izrađenog *API-a* u bazu unosi sve predviđene kategorije.

```
def kreiraj_kategorije(self, data, roditelj=None):
        for naziv, details in data.items():
            url = details.get('url')
            children = {k: v for k, v in details.items()
if k != 'url'}

            if
Kategorija.objects.filter(naziv=naziv).exists():
                kategorija =
Kategorija.objects.get(naziv=naziv)
                if url:
                    kategorija.url = url
                    kategorija.save()

self.stdout.write(self.style.SUCCESS(f'Ažurirana
kategorija: {kategorija} sa URL-om: {url}'))
                else:

self.stdout.write(self.style.WARNING(f'Kategorija
"{naziv}" već postoji u bazi. Preskačem...'))
            else:
                kategorija =
Kategorija.objects.create(naziv=naziv, url=url,
roditelj=roditelj)

self.stdout.write(self.style.SUCCESS(f'Dodana
kategorija: {kategorija} sa URL-om: {url}'))

            if children:
                self.kreiraj_kategorije(children,
kategorija)
```

Ispis 3: Python skripta za unos kategorija

3.5. Oglas

Tablica `Oglas` pohranjuje sve informacije potrebne za prikazivanje i pretraživanje oglasa. Jedan od atributa je cijena, koja je definirana kao decimalno polje s najviše deset znamenki i dvije decimale. Ovo omogućuje precizno pohranjivanje cijene. Svakom oglasu također je dodijeljena jedinstvena šifra, koja se automatski generira pri izradi oglasa uz pomoć funkcije `generiraj_sifru` (ispis 4). Šifra se sastoji od osam znamenki, što znači kako postoji mala mogućnost ponavljanja. Kako bi se osigurala jedinstvenost, funkcija provjerava postoji li generirana šifra već u bazi podataka. Ako šifra već postoji, funkcija ponavlja postupak generiranja nove šifre sve dok ne pronađe jedinstvenu vrijednost. Brojčani nizovi zauzimaju manje memorijskog prostora i omogućuju brže indeksiranje, što rezultira bržim pretraživanjem i pohranom podataka u usporedbi s nizovima koji sadrže slova i brojeve. Ova implementacija omogućuje brže pronalaženje oglasa po šifri.

```
        def generiraj_sifru():
            while True:
                sifra = str(uuid.uuid4().int)[:8]
                if not Oglas.objects.filter(sifra=sifra).exists():
                    return sifra
```

Ispis 4: Funkcija za generiranje jedinstvene šifre oglasa

Polje `naziv` pohranjuje naziv oglasa, s ograničenjem od dvjesto pedeset pet znakova. Uz naziv, polje `opis` omogućuje unos detaljnih informacija o oglasu, pružajući korisnicima sve potrebne podatke o predmetu oglasa.

Tablica `Oglas` također uključuje veze s drugim modelima. Polje `korisnik` povezuje oglas s tablicom `Korisnik`, omogućujući praćenje koji korisnik je postavio određeni oglasi.

Trajanje oglasa pohranjuje se u polju `trajanje`, koje nudi opcije za različite vremenske periode, kao što su jedan dan, tjedan ili mjesec. Oglasi mogu biti povezani s određenom kategorijom putem stranog ključa, omogućujući njihovu klasifikaciju i lakše filtriranje.

Datum kada je oglas objavljen pohranjuje se u polju `datum`, sa zadanim vrijednostima koje predstavljaju trenutno vrijeme kada je oglas objavljen, dok `status` označava stanje oglasa, kao što su "Aktivan", "Neaktivan", ili "Arhiviran". Ovaj status pomaže u praćenju životnog ciklusa oglasa i njegovoj vidljivosti korisnicima.

3.6. Slika

Tablica `Slika` omogućuje korisnicima vizualni prikaz svojih oglasa, čime se poboljšava privlačnost i informativnost oglasa.

U tablici `Slika` nalaze se dva ključna atributa. Prvi atribut je `oglas`, koji je povezuje slike s oglasom. Ako se oglas izbriše, sve povezane slike također će biti izbrisane. Drugi atribut je `slika`, koji omogućuje pohranu slika. Parametar `upload_to` specificira

direktorij unutar kojeg će se slike pohranjivati. Ovaj atribut sadrži putanju do direktorija unutar kojega se slika nalazi.

3.7. Favorit

Tablica `Favorit` omogućuje korisnicima označavanje oglasa koji su im zanimljivi ili koje žele sačuvati za kasniji pregled. U tablici `Favorit` nalaze se tri ključna atributa. Prvi atribut je `korisnik`, koji je veza prema tablici `Korisnik`. Ova veza omogućuje praćenje koji je korisnik označio određeni oglas kao favorit. Ako se korisnik izbriše, svi njegovi favoriti također će biti izbrisani. Pomoću atributa `related_name` omogućeno je jednostavno pristupanje svim favoritima određenog korisnika, što olakšava rad s vezama između tablica.

Drugi atribut je `oglas`, koji je također strani ključ, ali prema tablici `Oglas`. Ova veza omogućuje povezivanje svakog favorita s određenim oglasom. Ako se oglas izbriše, svi favoriti povezani s tim oglasom također će biti izbrisani, čime se održava dosljednost podataka.

Treći atribut je `datum`, koji automatski bilježi datum i vrijeme kada je oglas označen kao favorit.

3.8. Pregled

Tablica `Pregled` služi za praćenje i bilježenje informacija o pregledima oglasa, što omogućuje analizu korisničkog ponašanja i poboljšava uvid u popularnost oglasa. Tablica `Pregled` sadrži tri ključna atributa. Prvi atribut je oglas, koji je veza stranog ključa prema modelu Oglas. Ova veza povezuje svaki pregled s određenim oglasom. Ako se oglas izbriše, svi povezani pregledi također će biti izbrisani.

Drugi atribut je `ip_address`, koji pohranjuje *IP adresu* (engl. *Internet Protocol address*) korisnika koji je pregledao oglas. Ovo polje omogućuje bilježenje izvora pregleda, što u ovoj aplikaciji služi kako bi se ograničio broj pregleda koje svaki korisnik može generirati.

Treći atribut je `timestamp`, koji automatski bilježi datum i vrijeme kada je oglas pregledan. Ovo polje služi kako bi se bilježilo odstupanje od zadnjeg pregleda kojeg je određeni korisnik generirao.

3.9. Komentar

Tablica `Komentar` omogućuje korisnicima objavljivanje komentara na oglase, pružajući tako dodatnu povratnu informaciju unutar sustava. Ova tablica pohranjuje sve bitne informacije vezane uz komentare, uključujući autora komentara, sadržaj i vrijeme objave.

Ova tablica ima četiri glavna atributa. Prvi atribut je `korisnik`, koji povezuje komentar s korisnikom koji prima povratnu informaciju od drugih korisnika.

Drugi atribut je `autor`. Ovaj atribut omogućuje razlikovanje između korisnika koji je napisao komentar i korisnika kojem je komentar namijenjen.

Treći atribut je `tekst`, koji pohranjuje sadržaj komentara. Ovaj atribut ima ograničenje od sto pedeset znakova. Ovo ograničenje omogućuje održavanje komentara kratkima i direktnima.

Četvrti atribut je `timestamp`, koji automatski bilježi datum i vrijeme kada je komentar objavljen.

Ovaj model pomaže u održavanju angažiranosti korisnika i poboljšava cjelokupno korisničko iskustvo u aplikaciji.

4. Registracija korisničkog računa, prijava i odjava

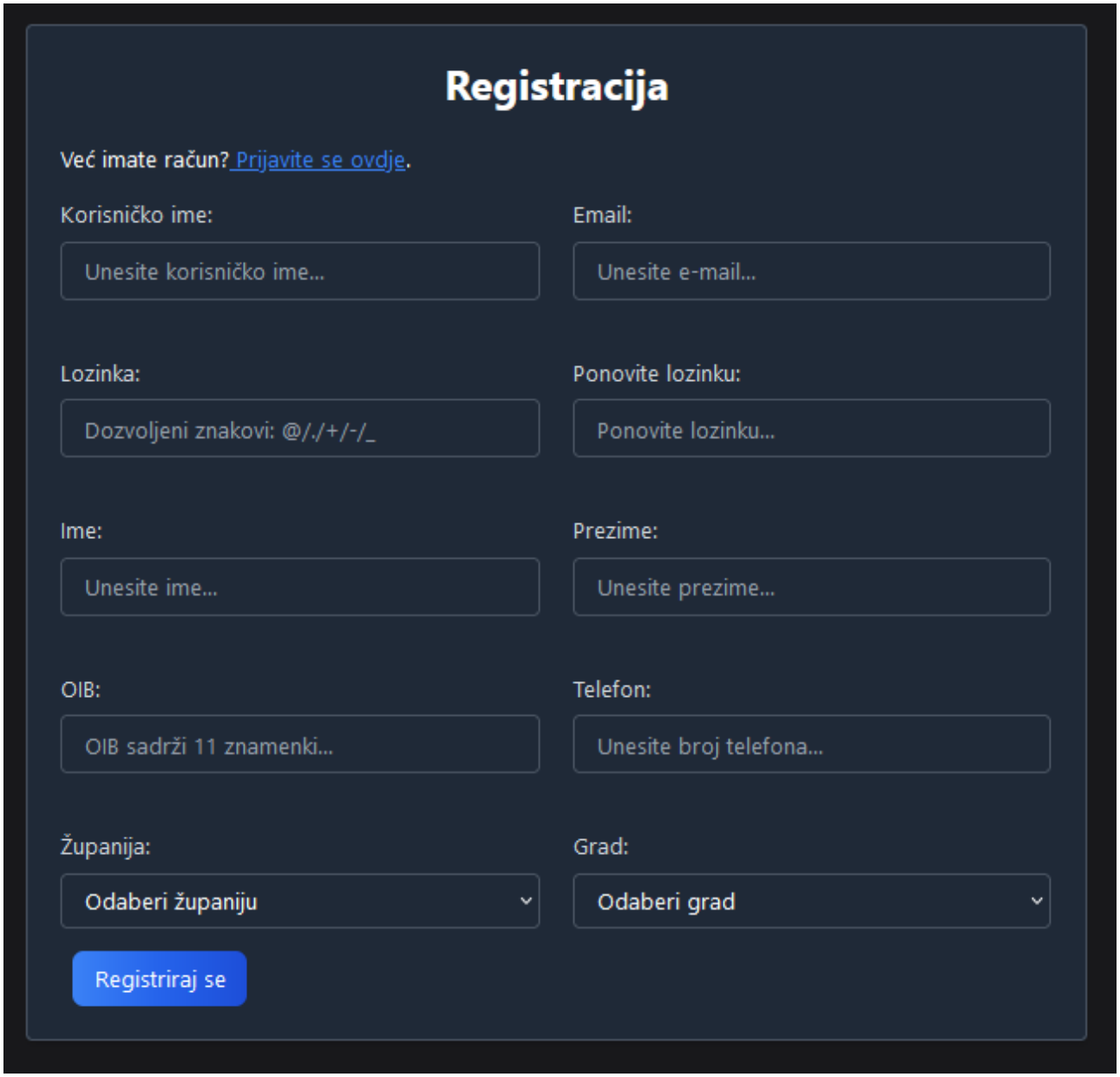
4.1. Registracija

Komponenta `Registracija` omogućuje korisnicima registraciju korisničkog računa putem jednostavnog obrasca. Kada korisnik otvori ovu stranicu, automatski se dohvaćaju županije koje se prikazuju u padajućem izborniku. Nakon što korisnik odabere županiju, prikazuju se odgovarajući gradovi.

Obrazac za registraciju korisnika (slika 2) uključuje polja za unos korisničkog imena, elektroničke pošte, lozinke, ponovljene lozinke, imena, prezimena, OIB-a, telefonskog broja, županije i grada. Korisnik unosi podatke u ta polja, a ako dođe do grešaka, one se prikazuju ispod odgovarajućih polja.

Kada korisnik pošalje obrazac, podaci se šalju na poslužitelj. Ako je registracija uspješna, korisnik dobiva obavijest o uspjehu i preusmjerava se na stranicu za prijavu.

Obrazac je dizajniran s jasnim i modernim izgledom, koristeći stilove koji omogućuju preglednost na različitim uređajima.



Slika 2: Obrazac za registraciju korisnika

4.2. Prijava

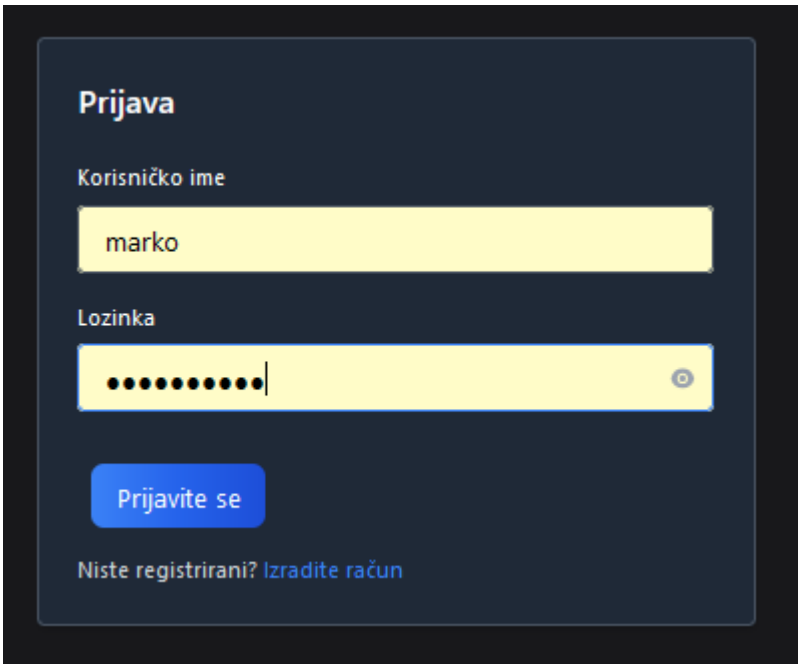
Komponenta `Prijava` omogućuje korisnicima prijavljivanje na sustav koristeći svoje korisničko ime i lozinku. Kada korisnik otvori ovu stranicu, vidjet će obrazac za prijavu s poljima za unos korisničkog imena i lozinke.

Obrazac se sastoji od dva osnovna polja: jedno za korisničko ime i drugo za lozinku. Korisnici mogu unijeti svoje podatke, a ako žele vidjeti lozinku dok je upisuju, mogu koristiti dugme za prikazivanje ili skrivanje lozinke. Kada korisnik pošalje obrazac, podaci se šalju

na poslužitelj za autentifikaciju. Ako su podaci ispravni, korisniku se dodjeljuju pristupni i osvježavajući tokeni, koji se spremaju u lokalnu memoriju preglednika. Nakon toga, traže se podaci o trenutno prijavljenom korisniku i ažurira se stanje prijavljenog korisnika.

Ako prijava nije uspješna, korisnik prima obavijest o pogrešci s porukom koja označava da su uneseni podaci netočni.

Komponenta također nudi opciju za nove korisnike koji još nisu registrirani, omogućujući im registraciju putem poveznice koja vodi na stranicu za registraciju. Dizajn obrasca (slika 3) je prilagođen za jednostavnu i preglednu upotrebu, s modernim izgledom i jasnim vizualnim i povratnim informacijama.



Slika 3: Obrazac za prijavu

4.3. Odjava

Komponenta `Odjava` omogućuje korisnicima odjavljivanje iz sustava. Kada korisnik pokrene ovu funkciju, šalje se zahtjev poslužitelju za odjavu. Zahtjev se šalje uz pristupni token koji je pohranjen u lokalnoj memoriji preglednika.

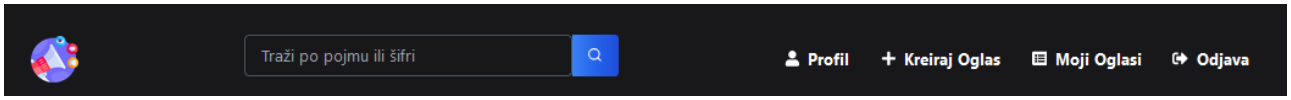
Ako je zahtjev za odjavom uspješan, pristupni i osvježavajući tokeni se uklanjaju iz lokalne memorije, čime se korisnik odjavljuje. Nakon toga, korisnik se preusmjerava na naslovnu stranicu aplikacije. Funkcija `odjavaKorisnika` (ispis 5) osigurava pravilnu odjavu korisnika i uklanjanje njihovih tokena, čime se održava sigurnost i privatnost korisničkih podataka.

```
const odjavaKorisnika = async () => {
    try {

        const response = await fetch('http://localhost:8000/api/odjava/', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
                Authorization: `Bearer
             ${localStorage.getItem('access_token')}`
            },
            credentials: 'include'
        });

        if (response.ok) {
            localStorage.removeItem('access_token');
            localStorage.removeItem('refresh_token');
            window.location.href = 'http://localhost:3000/';
        }
         else {
            console.error('Odjava neuspješna');
        }
    } catch (error) {
        console.error('Greška pri odjavi:', error);
    }
};
```

Ispis 5: Funkcija za odjavu korisnika

5. Korisničko sučelje

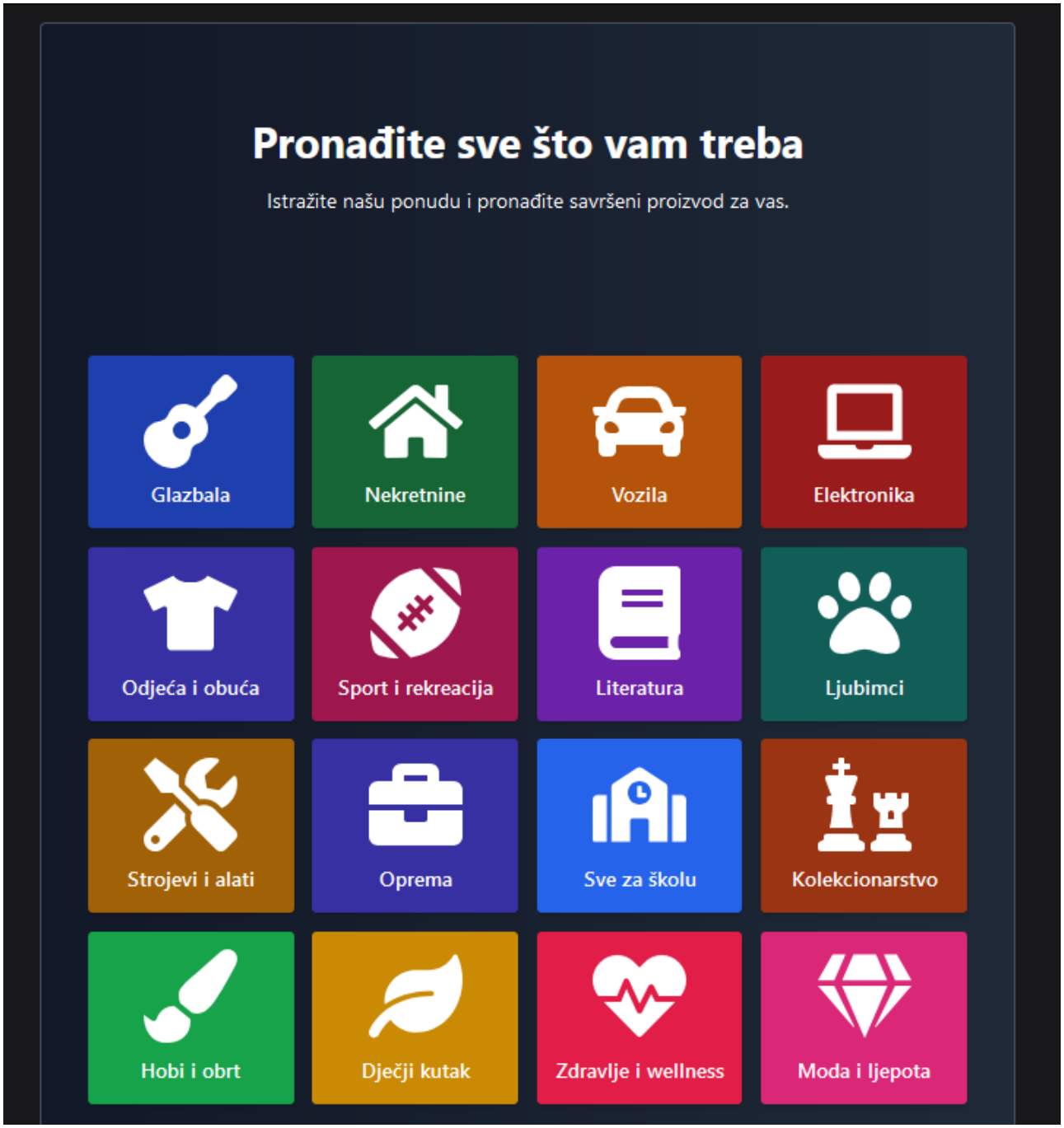
Korisničko sučelje naslovne stranice aplikacije oblikovano je kako bi korisnicima omogućilo jednostavno i intuitivno iskustvo, bilo da su prijavljeni ili ne. Na vrhu stranice nalazi se navigacijska traka (slika 4) s logotipom aplikacije, koja služi kao poveznica za povratak na naslovnu stranicu. U sredini trake nalazi se polje za pretraživanje, omogućujući korisnicima pretraživanje oglasa po nazivu ili šifri.



Slika 4: Navigacijska traka prijavljenog korisnika

Neprijavljeni korisnici s desne strane navigacije vide poveznice za prijavu i registraciju. Prijavljeni korisnici umjesto toga imaju izbornik s opcijama za pristup profilu, kreiranje oglasa, pregled vlastitih oglasa, dok administratori imaju dodatnu poveznicu za pristup administrativnom sučelju. Za mobilne korisnike izbornik je prilagodljiv i lako proširiv, čime su sve opcije dostupne bez obzira na uređaj.

U središnjem dijelu stranice nalaze se unaprijed određene kategorije (slika 5) oglasa, predstavljene kroz mrežu jasno vidljivih ikona, čime se olakšava vizualna orijentacija.



Slika 5: Prikaz kategorija za pretragu oglasa

5.1. Korisnički profil

Korisnički profil prikazuje sve ključne informacije o korisniku, uključujući korisničko ime, ime i prezime, adresu elektroničke pošte, broj telefona, OIB, lokaciju, te datum kada je korisnik izradio račun. Uz te informacije, korisnik ima dvije glavne mogućnosti: ažurirati svoje podatke ili promijeniti lozinku. Ukoliko korisnik promijeni lozinku, prisilno je odjavljen i preusmjeren na stranicu za prijavu (ispis 6).

```
const handleLogout = () => {
    localStorage.removeItem('access_token');
    localStorage.removeItem('refresh_token');
    window.location.href = '/prijava'
 };


const response = await fetch('http://localhost:8000/promjena-lozinke/', {
                method: 'PUT',
                headers: {
                    'Content-Type': 'application/json',
                    'Authorization':
            `Bearer${localStorage.getItem('access_token')}`
                },
                body: JSON.stringify({
                    old_password: oldPassword,
                    new_password1: newPassword1,
                    new_password2: newPassword2,
                })
            });

            const data = await response.json();

            if (response.ok) {
               handleLogout();
               toast.success('Lozinka je promijenjena, prijavite se
            ponovno.', {
                   onClose: () => navigate('/prijava')
               });
```

Ispis 6: Python skripta za unos županija i gradova

5.2. Kreiranje oglasa

Objavljivanje oglasa vrši se putem obrasca za kreiranje oglasa. Ovaj obrazac traži od korisnika ispunjavanje polja poput cijene, naziva oglasa, opisa, kategorije i dodavanje slika. Za validaciju unosa podataka služi funkcija `validateInputs`. Validacija unosa podataka kontrolira ispravnost unesenih informacija kako bi se spriječile pogreške (ispis 7). Cijena mora biti brojčana vrijednost i može imati najviše dvije decimale, naziv i opis su obavezni, a od korisnika se traži odabir kategorije i odabir barem jedne slike. Ako neka od ovih stavki nedostaje ili je neispravno unesena, u obrascu se prikazuje poruka o pogrešci.

```
const validateInputs = () => {
    const newErrors = {};
    if (!podaciForme.cijena || isNaN(podaciForme.cijena)) {
        newErrors.cijena = 'Cijena mora biti broj.';
    } else {
        const cijenaDecimale = podaciForme.cijena.split('.');
        if (cijenaDecimale.length > 1 && cijenaDecimale[1].length > 2) {
            newErrors.cijena = 'Cijena može imati najviše dvije decimale.';
         }
        }
        if (!podaciForme.naziv) {
           newErrors.naziv = 'Naziv je obavezan.';
        }
        if (!podaciForme.opis) {
            newErrors.opis = 'Opis je obavezan.';
        }
        if (!podaciForme.kategorija) {
            newErrors.kategorija = 'Kategorija je obavezna.';
        }
        if (podaciForme.slike.length === 0) {
            newErrors.slike = 'Morate dodati barem jednu sliku.';
        } else if (podaciForme.slike.length > MAX_BROJ_SLIKA) {
                  newErrors.slike = `Možete odabrati maksimalno
           ${MAX_BROJ_SLIKA}  slike.`;
        }
        setErrors(newErrors);
        return Object.keys(newErrors).length === 0;
        };
```

Ispis 7: Validacija unosa u obrascu za kreiranje oglasa

5.3. Moji oglasi

`Moji oglasi` je sučelje koja omogućuje pregledavanje i upravljanje vlastitim oglasima, kao i oglasima koji su dodani u favorite. Ako korisnik nema objavljenih oglasa, prikazuje se obavijest s opcijom za kreiranje novog oglasa. U ovom sučelju, korisnik može ažurirati vlastite oglase, te ih brisati. Kroz modalni prozor, korisnik dobiva mogućnost potvrde brisanja oglasa. Oglas se ne briše iz baze podataka, već se uz pomoć funkcije `izbrisiOglas` njegov status mijenja u "arhiviran" (ispis 8), čime se gubi njegova vidljivost.

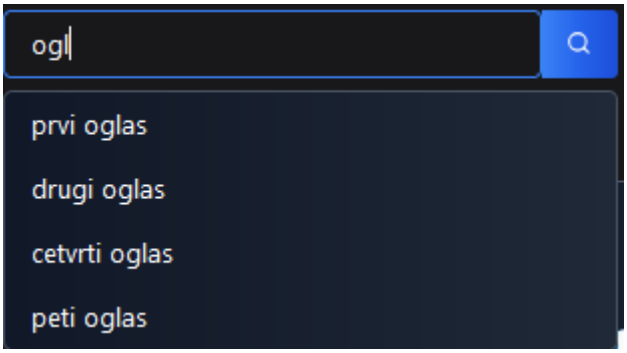
```
const izbrisiOglas = async () => {
     try {
         const accessToken = localStorage.getItem('access_token');
        const response = await
    fetch(`http://localhost:8000/api/oglas/${oglasToDelete}/arhiviraj/`, {
              method: 'POST',
              headers: {
                  Authorization: `Bearer ${accessToken}`,
                  'Content-Type': 'application/json',
              },
              credentials: 'include',
          });
          if (response.ok) {
              setOglasi(
                  oglasi.map((oglas) =>
                              oglas.id === oglasToDelete ? { ...oglas, status:
    'arhiviran' } : oglas)
              );
          }
  };
```

Ispis 8: Prikaz implementacije brisanja oglasa

5.3. Pretraživanje oglasa tražilicom

Tražilica omogućuje korisnicima da pretražuju oglase putem dinamičkog unosa teksta. Oglasi se mogu pretraživati unošenjem naziva oglasa ili njegove jedinstvene šifre. Kada korisnik upisuje pojam ili šifru u pretraživačko polje, tražilica automatski prikazuje listu relevantnih rezultata. Ovi prijedlozi se prikazuju u padajućem izborniku ispod pretraživačkog polja (slika 6).

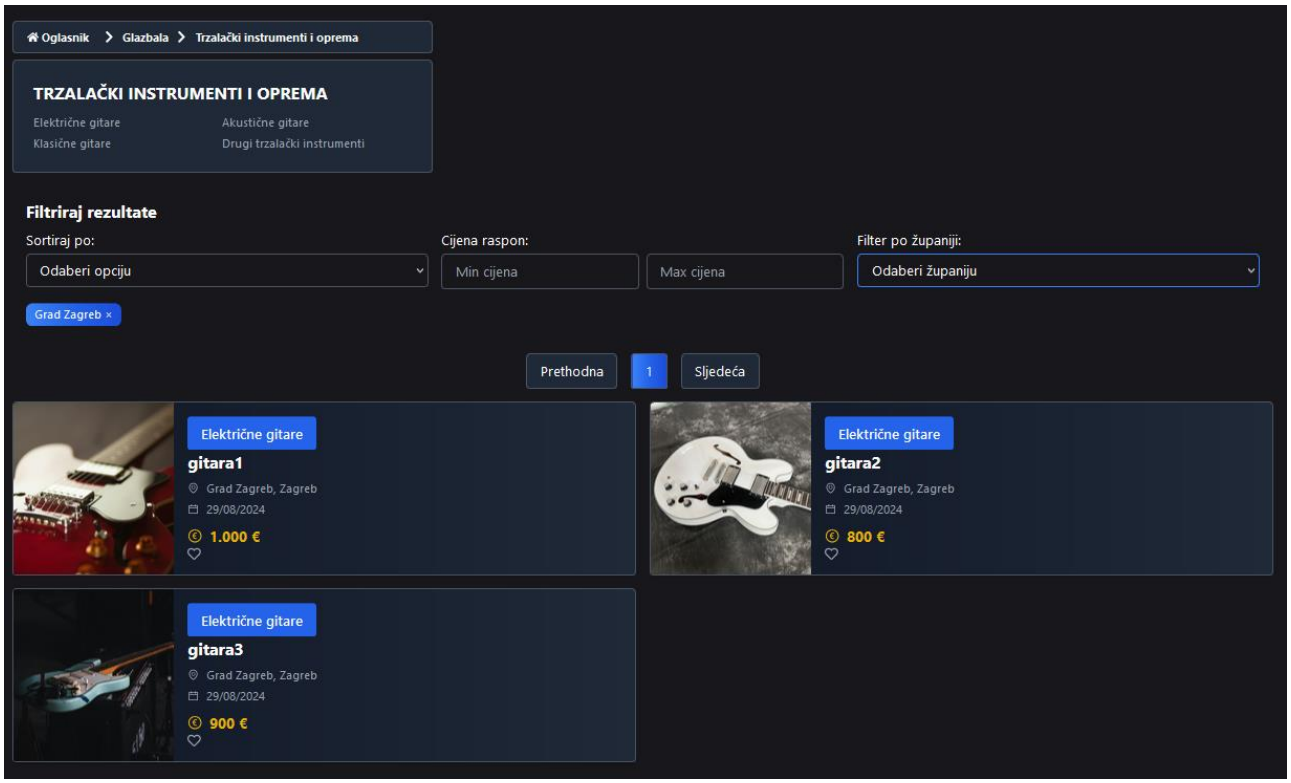
Odabirom jednog od prijedloga iz padajućeg izbornika, ispunjava se pretraživačko polje.



Slika 6: Lista prijedloga unutar padajućeg izbornika tražilice

5.4. Pretraživanje oglasa po kategorijama

Osim pretrege po nazivu ili šifri oglasa, korisnik za mogućnost ima pretragu po kategoriji. Na naslovnoj stranici, korisniku su prikazane sve glavne kategorija. S obzirom da svaka kategorija može imati svoje podkategorije. Izborom jedne od glavnih kategorija, korisnik dalje može filtrirati oglase po pripadajućim podkategorijama (slika 7).



Slika 7: Pretraživanje oglasa po kategorijama

Korisnik ima mogućnost nadalje filtrirati oglase po cijeni, županiji, te sortirati po cijeni ili datumu objave oglasa. Kako bi se korisnicima olakšala navigacija po kategorijama, dostupna je traka s hijerarhijom oglasa po kategoriji.

U ovom prikazu, dostupne su samo osnovne informacije o oglasu, a to su: slika, kategorija oglasa, lokacija korisnika koji je objavio oglas, datum objave oglasa i cijena predmeta. Uz to, dostupna je i mogućnost dodavanja oglasa među favorite, kako bi korisnik kasnije mogao posjetiti taj oglas.

Odabirom jednog od oglasa, korisniku se prikazuje detaljan uvid u oglas, što uključuje i podatke o korisniku koji je objavio oglas. Kontakt između oglašivača i kupca namijenjen je putem kontakt telefona ili elektroničke pošte. Uz podatke o korisniku vezane za kontakt, nalazi se i korisničko ime koje služi kao poveznica na profil oglašivača.

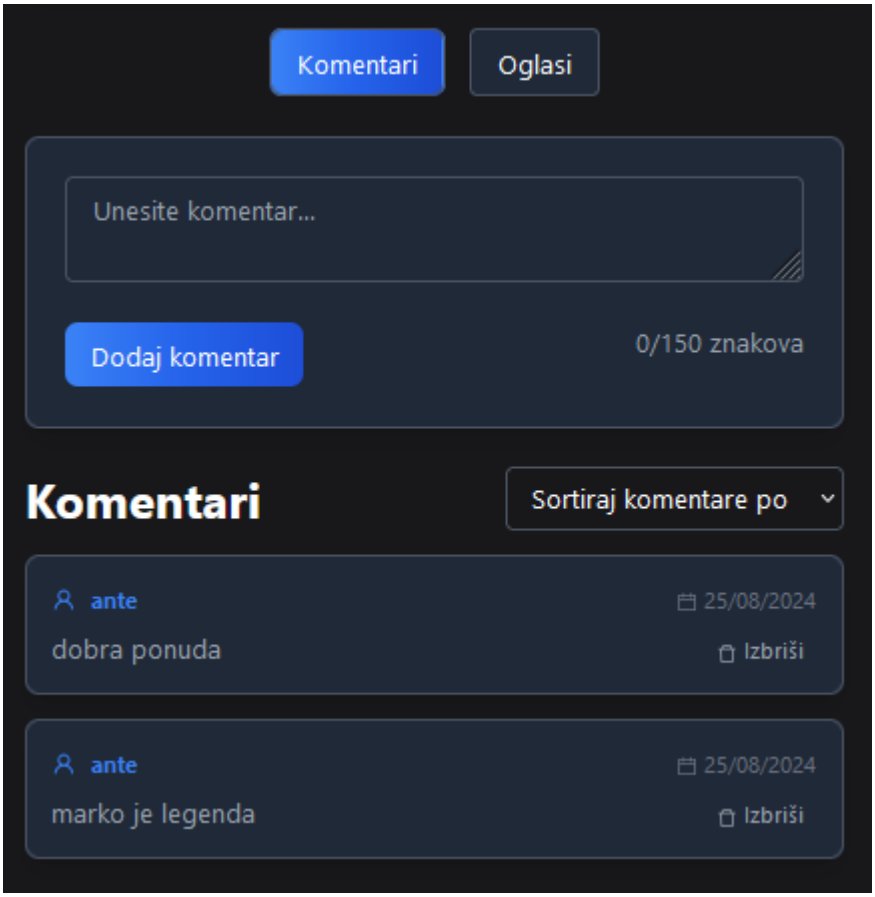
Unutar profila oglašivača, omogućen je pregled svih aktivnih oglasa odabranog korisnika. Pri dohvaćanju podataka o korisniku, filtriraju se oglasi tako da su samo oni koji imaju status aktivan vidljivi korisniku koji pregledava profil (ispis 9).

```
.then(data => {
    setUserData(data.korisnik);

    const activeOglasi = data.oglasi.filter(oglas => oglas.status ===
    'aktivan');

    setOglasi(activeOglasi);
    setComments(data.korisnik.komentari);
    })
```

Ispis 9: Filtriranje oglasa na profilu oglašivača

Prijavljeni korisnici na profilu oglašivača imaju mogućnost objavljivati komentare kroz obrazac za objavu komentara (slika 8).

Komentari su namijenjeni kao povratna informacija o oglašivaču. Uz samog autora komentara, administrator ima mogućnost brisati komentare. Komentari se mogu sortirati po datumu objave, ulazno ili silazno.



Slika 8: Obrazac za objavu komentara

Duljina komentara ograničena je na klijentskoj strani. Svaki komentar može sadržavati najviže sto pedeset znakova. Komentar sadrži poveznicu na profil autora komentara, datum objave i sadržaj komentara.

6. Administratorsko sučelje

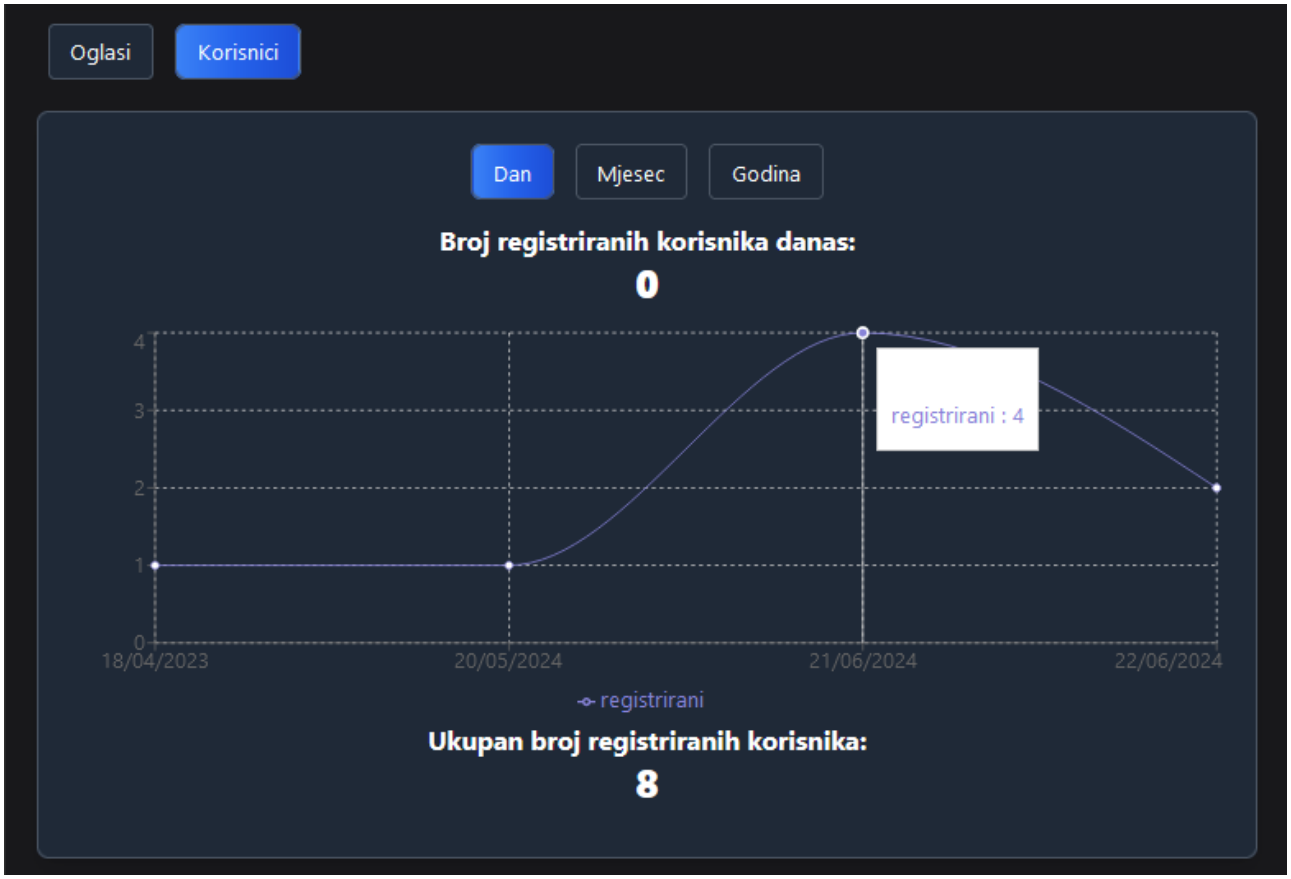
Uz sve značajke korisničkog sučelja, administrator ima pristup posebnom sučelju putem kojeg može pregledavati podatke o registriranim korisnicima, izrađenim oglasima, te ima pristup statistici o izradi oglasa i korisničkih računa po vremenskim periodima. Također, administratorsko sučelje dozvoljava pretraživanje svih oglasa i korisnika.

6.1. Analiza izrade korisničkih računa

Svrha prikupljanja statistike o datumima izrade korisničkih računa je analiza korisničkog rasta i aktivnosti na platformi. Ova statistika omogućuje praćenje trenda novih registracija kroz određene vremenske periode, poput dana, mjeseca ili godine. Uz to, pomaže u identifikaciji sezonskih ili kampanjskih utjecaja na broj novih korisnika te omogućuje optimizaciju marketinških strategija, na temelju kojeg se može poboljšati korisničko iskustvo. U svrhu analize izrade korisničkih računa, administratorsko sučelje uključuje graf izrade korisničkih računa (slika 9).

Ovaj graf pruža uvid u dinamiku broja izrađenih korisničkih računa kroz različite vremenske periode. Graf se može prilagoditi za prikaz podataka prema danima, mjesecima ili godinama. Kada je filtriran po danima, graf prikazuje ukupan broj oglasa koji su kreirani na tekući dan, kao i pregled broja računa izrađenih po datumima.

Filtriranje po mjesecu omogućuje pregled broja izrađenih računa u tekućem mjesecu, kao i pregled broja izrađenih računa po mjesecima. Slično, filtriranje po godini omogućuje pregled broja izrađenih računa u tekućoj godini, kao i pregled broja izrađenih računa po godinama. Uz to, graf prikazuje i ukupan broj registracija, bez obzira na filter.



Slika 9: Graf izrade korisničkih računa

Podaci o izrađenim korisničkim računima uz pomoć `aggregateRegistrations` funkcije koja format prikazivanja datuma prilagođava odabranom prikazu po danima, mjesecima i godinama (ispis 10).

Kada su svi podaci obrađeni, funkcija pretvara ključeve akumulatora u niz objekata, gdje svaki objekt sadrži datum i broj registracija. Ovi objekti se zatim sortiraju prema datumu koristeći *React* knjižnicu, kako bi podaci bili prikazani u pravilnom vremenskom redoslijedu.

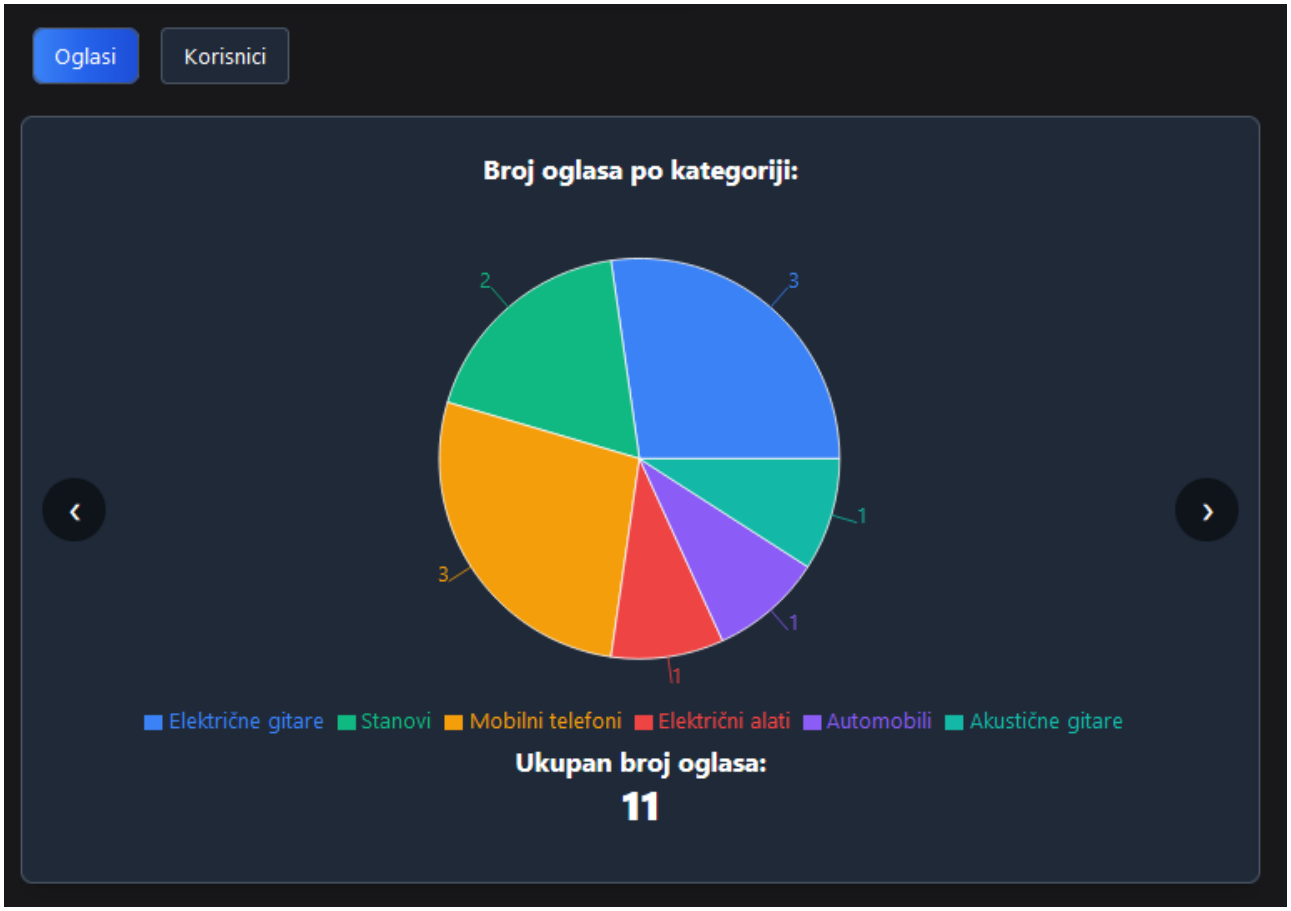
```
    const aggregateRegistrations = (korisnici, period) => {
          const registrations = korisnici.reduce((acc, user) => {
          let key;
          if (period === 'day') {
              key = moment(user.date_joined, 'DD/MM/YYYY').format('DD/MM/YYYY');
          } else if (period === 'month') {
              key = moment(user.date_joined, 'DD/MM/YYYY').format('MM/YYYY');
          } else if (period === 'year') {
              key = moment(user.date_joined, 'DD/MM/YYYY').format('YYYY');
          }

          if (!acc[key]) {
              acc[key] = 0;
          }
          acc[key]++;
          return acc;
      }, {});
  };
```

Ispis 10: Funkcija za prikupljanje broja registracija

6.2. Analiza izrade oglasa

Uz prikupljanje podataka o izradi korisničkih računa, također se prikupljaju i podaci o izradi oglasa. U svrhu analize izrade oglasa, administratorsko sučelje ima dva grafa vezana uz analizu izrade oglasa. Prvi graf prikazuje dinamiku izrade oglasa, slično kao u primjeru grafa za izradu korisničkih računa. Graf izrade oglasa uz to prikazuje i ukupan broj pregleda na svim oglasima.

Drugi graf prikazuje izradu oglasa po određenim kategorijama. Prikupljanje statistike o oglasima po kategorijama pruža uvid u strukturu ponude. Analizom broj oglasa unutar različitih kategorija možemo razumjeti koje kategorije su najpopularnije i najviše zastupljene, što pomaže u prepoznavanju tržišnih trendova i korisničkih interesa. Ova statistika pomaže u identifikaciji područja s velikom konkurencijom, kao i onih s mogućim nedostatkom ponude. Ovaj graf također prikazuje i ukupan broj svih oglasa (slika 10).

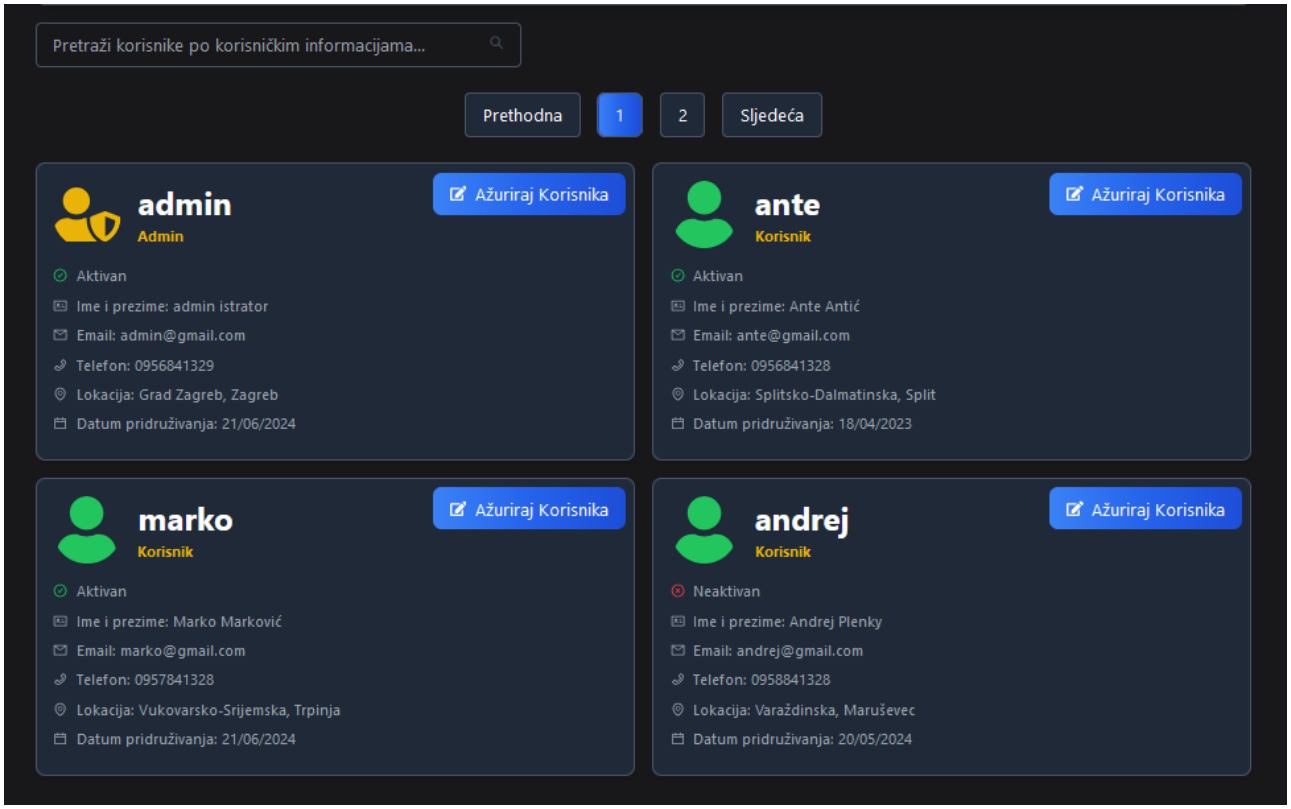


Slika 10: Graf izrade oglasa po kategorijama

6.3. Pregled svih korisnika

Pregled svih korisnika (slika 11) omogućuje administratorima učinkovito upravljanje korisnicima aplikacije. Ovaj dio administratorskog sučelja nudi pregledan i organiziran način za pretraživanje, filtriranje i upravljanje korisnicima.

Tražilica omogućuje pretraživanje korisnika po korisničkom imenu, imenu i prezimenu, elektroničkoj pošti ili broju telefona. Svaki korisnik prikazan u pregledu ima svoje detalje prikazane u karticama koje uključuju osnovne informacije. Uloge korisnika su istaknute pomoću ikona, gdje administratori imaju posebne oznake. Poveznice za ažuriranje korisničkih podataka nude mogućnost ažuriranja podataka kroz administratorsko sučelje. Oglasi su kroz paginaciju podijeljeni u stranice, čime se omogućuje prikaz samo određenog broja korisnika po stranici.



Slika 11: Prikaz svih korisnika u administratorskom sučelju

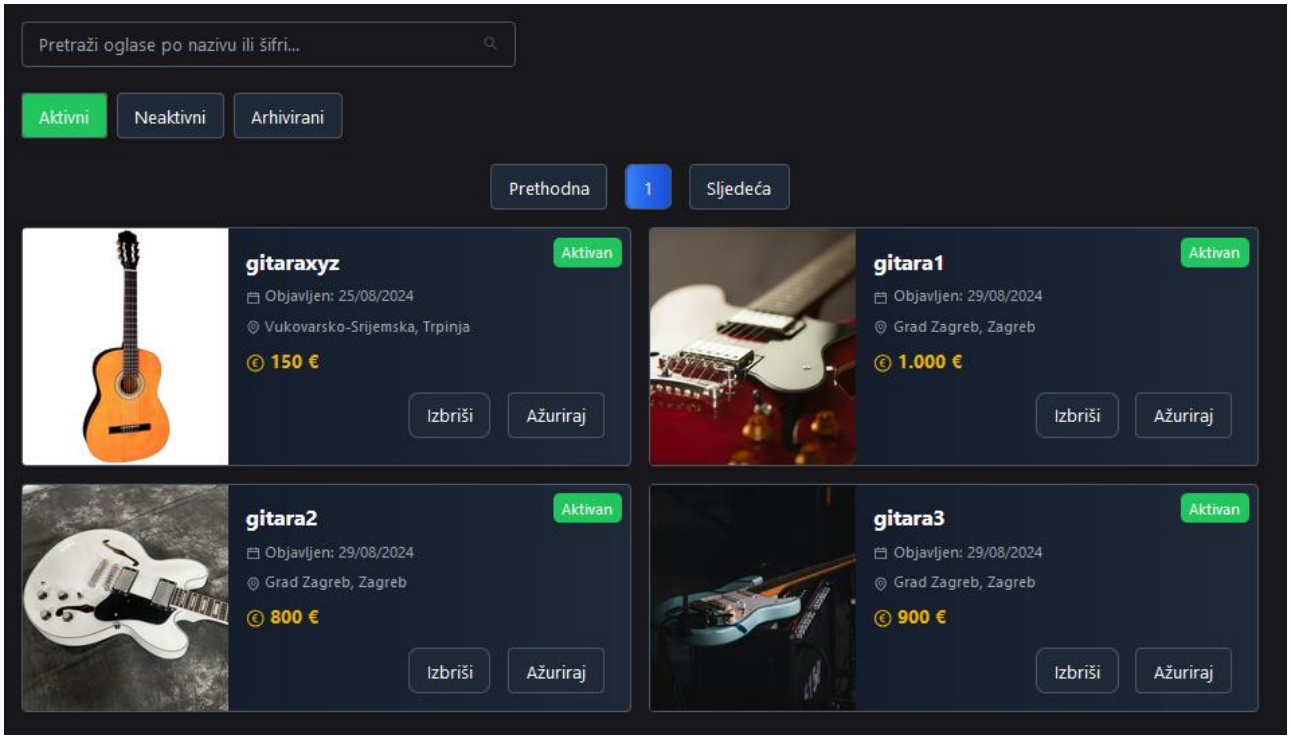
Administratori mogu upravljati statusom korisničkog računa putem ikone koja omogućuje aktivaciju ili deaktivaciju korisničkih računa. Pozivom funkcije `toggleIsActive` mijenja se status korisničkog računa (ispis 11).

```
const toggleIsActive = async (korisnikId, currentStatus) => {
     try {
          const accessToken = localStorage.getItem('access_token');
          const headers = {
              Authorization: `Bearer ${accessToken}`,
          };

          await axios.patch(`/api/korisnici/${korisnikId}/`, {
              is_active: !currentStatus
          }, { headers });

          const updatedKorisnici = korisnici.map(korisnik =>
                            korisnik.id === korisnikId ? { ...korisnik,
            is_active:    !currentStatus } : korisnik
          );
          setKorisnici(updatedKorisnici);
     } catch (error) {
          console.error('Error updating status:', error);
     }
 };
```

Ispis 11: Promjena statusa korisničkog računa

6.4. Pregled svih oglasa

Pregled svih oglasa (slika 12) omogućuje administratorima pretraživanje i filtriranje oglasa. Filtriranje oglasa temelji se na različitim kriterijima, uključujući status oglasa, naziv ili šifru.



Slika 12: Pregled svih oglasa u administratorskom sučelju

Filtriranje oglasa po statusu (ispis 12) omogućuje administratorima pretraživanje samo trenutno aktivnih oglasa, neaktivnih oglasa ili onih koji su arhivirani. Kako bi se omogućila preglednost, oglasi su podijeljeni u stranice s određenim brojem stavki po stranici. Tražilica omogućuje pretraživanje oglasa po nazivu ili šifri. U sučelju se oglasi prikazuju u obliku kartica, uz osnovne podatke o oglasima i statusu oglasa.

Administratorsko sučelje ima mogućnost ažuriranja oglasa kroz obrazac za ažuriranje oglasa koji je dostupan i u korisničkom sučelju. Polja u obrascu za ažuriranje oglasa unaprijed su napunjena podacima vezanim uz taj oglas.

U administratorskom sučelju, funkcionalnost za brisanje oglasa briše oglas iz baze podataka, za razliku od funkcionalnosti korisničkog sučelja u kojem brisanje oglasa samo mijenja njegov status. Brisanjem oglasa kroz administratorsko sučelje gubi se i broj pregleda vezanih uz taj oglas. Oglasi nisu namijenjeni za brisanje iz baze podataka, ali brisanje je ipak omogućeno administratorima.

```
const toggleStatus = (status) => {
          if (selectedStatuses.includes(status)) {
              setSelectedStatuses(selectedStatuses.filter(s => s !== status));
          } else {
              setSelectedStatuses([...selectedStatuses, status]);
          }

          setCurrentPage(1);

          const params = new URLSearchParams(window.location.search);
          params.set('page', '1');
          window.history.replaceState({}, '',
     `${window.location.pathname}?${params}`);
      };
```

Ispis 12: Filtriranje svih oglasa po statusu

7. Zaključak

U ovom radu opisana je aplikacija namijenjena objavljivanju i pregledavanju oglasa, s mogućnošću pretraživanja prema različitim kriterijima poput kategorije, lokacije, cijene i drugih karakteristika. Korisnici mogu postavljati vlastite oglase te pregledavati oglase drugih korisnika, kao i podatke o oglašivačima.

Komunikacija između korisnika odvija se putem elektroničke pošte ili kontakt telefona. Za poboljšanje funkcionalnosti aplikacije, poželjno bi bilo dodati integrirani sustav privatnih poruka koji bi omogućio komunikaciju unutar same aplikacije, bez potrebe za komunikacijom putem elektroničke pošte. Također, sustav obavijesti unutar aplikacije mogao bi korisnicima pružati ažuriranja o važnim događajima, kao što su novi upiti ili odgovori, na jednom centraliziranom mjestu.

Ova aplikacija izrađena je pomoću *Django* radnog okvira, što je značajno olakšalo proces razvoja. Jedna od glavnih prednosti ovog pristupa je što većina ključnih funkcionalnosti dolazi već ugrađena, čime se minimiziraju izazovi oko kompatibilnosti i postavljanja radnog okruženja. Definiranje odnosa između tablica unutar baze podataka je pojednostavljeno, a pristup povezanim podacima dolazi automatski, bez potrebe za dodatnim kôdom.

Pored toga, korištenje *Tailwind* knjižnice u aplikaciji donijelo je prednost u brzom i prilagodljivom dizajniranju korisničkog sučelja. *Tailwind* omogućuje jednostavno i modularno stiliziranje bez potrebe za pisanjem dugih *CSS* datoteka, što znatno ubrzava proces diazjniranja.

Integracija *React* knjižnice pružila je dodatnu fleksibilnost pri izradi korisničkog sučelja. *React* omogućuje kreiranje dinamičkih i responzivnih komponenti koje se lako prilagođavaju različitim funkcionalnostima aplikacije, poboljšavajući korisničko iskustvo.

Literatura

[1] Django, "Django project," https://www.djangoproject.com/ (posjećeno 25.08.2024.)

[2] SQLite, "About SQLite," https://www.sqlite.org/about.html (posjećeno 25.08.2024.)

[3] Tailwind, "Tailwind CSS," https://tailwindcss.com/ (posjećeno 25.08.2024.)

[4] React, "A JavaScript library for building user interfaces," https://react.dev/ (posjećeno 25.08.2024.)