Web aplikacija za oglasnik

Klanjšček, Robert

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

Preddiplomski Stručni studij Računarstva

WEB APLIKACIJA ZA OGLASNIK

Završni rad

Robert Klanjšček

Osijek, 2024.

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

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Robert Klanjšček
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	AR4855, 27.07.2021.
JMBAG:	0165078680
Mentor:	Robert Šojo, univ. mag. ing. comp.
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	mr. sc. Željko Štanfel
Član Povjerenstva 1:	Robert Šojo, univ. mag. ing. comp.
Član Povjerenstva 2:	doc. dr. sc. Ivana Hartmann Tolić
Naslov završnog rada:	Web aplikacija za oglasnik
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Opis teme: Izrada web aplikacije na kojoj je omogućeno predaja različitih oglasa, kao i njihovo pregledavanje. Web aplikacija omogućava i kupovinu proizvoda postavljanjem u košaricu te dogovor kupovine s oglašivačem osobno. Omogućena je razlika između tvrtki koje oglašavaju svoje proizvode te fizičkih osoba tako što je chat uveden za fizičku osobu. Administrator kreira osoblje koje će nadgledati postavljene oglase. Registrirane tvrtke i fizičke osobe koje žele postavljati svoje oglase moraju biti registrirane, a ne prijavljeni korisnici web aplikacije mogu samo pregledavati

Datum ocjene pismenog dijela završnog rada od strane mentora:	19.09.2024.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	2.10.2024.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	02.10.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 02.10.2024.

Ime i prezime Pristupnika:	Robert Klanjšček
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AR4855, 27.07.2021.
Turnitin podudaranje [%]:	11

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za oglasnik**

izrađen pod vodstvom mentora Robert Šojo, univ. mag. ing. comp.

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 pristupnika:

SADRŽAJ

1. UVOD

Razvoj internetskih tehnologija doveo je do značajnih promjena u načinu na koji komuniciramo, poslujemo i kupujemo proizvode. Danas su online platforme za oglašavanje postale ključan alat za tvrtke i pojedince koji žele predstaviti svoje proizvode široj publici. Web aplikacije koje omogućuju oglašavanje i trgovinu postale su nezaobilazne u modernom poslovnom okruženju, pružajući korisnicima jednostavan način da pregledavaju, kupuju i prodaju proizvode ili usluge. Ovaj završni rad bavi se razvojem web aplikacije koja omogućava korisnicima postavljanje i pregledavanje oglasa. Aplikacija nudi mogućnost dodavanja proizvoda u košaricu te dogovaranja kupovine s oglašivačem putem osobnog kontakta. Također, omogućuje administratorima da upravljaju korisnicima i oglasima, osiguravajući sigurnost i pouzdanost platforme.

U drugom poglavlju analizirane su postojeće aplikacije za oglašavanje, uključujući njihove funkcionalnosti i značajke. Treće poglavlje se bavi tehnologijama korištenim pri izradi aplikacije, dok četvrto poglavlje prikazuje proces izrade i strukturu web aplikacije, s detaljnim opisom implementiranih funkcionalnosti.

1.1. Zadatak završnog rada

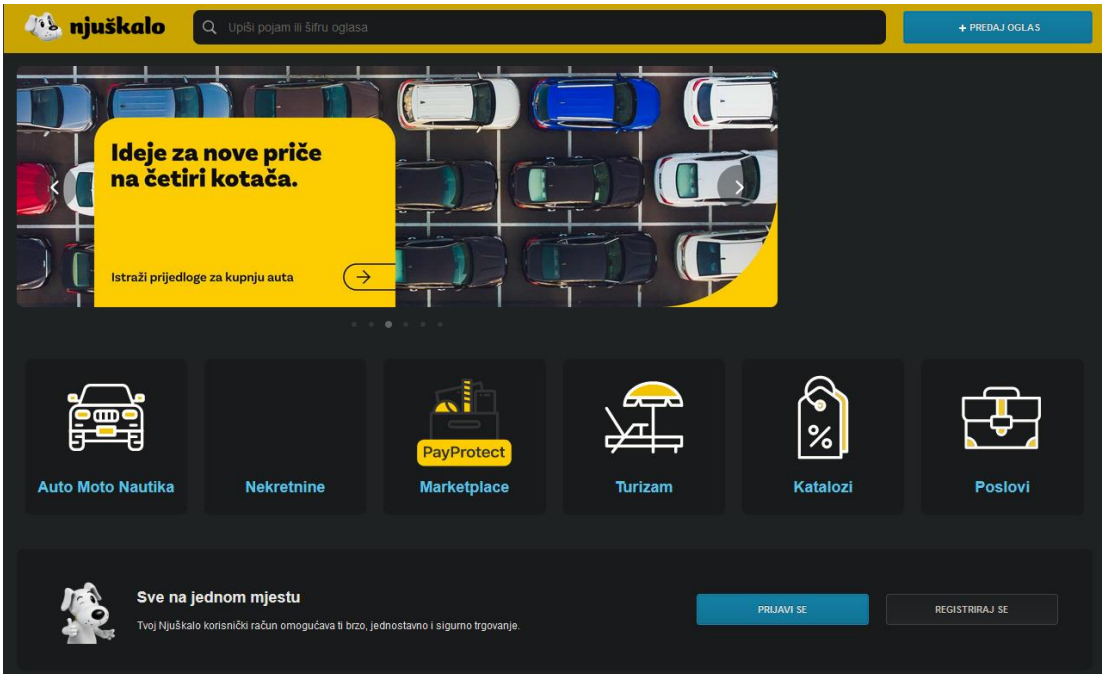
Zadatak završnog rada je izraditi web aplikaciju koja omogućuje korisnicima predaju i pregledavanje različitih vrsta oglasa, kao i mogućnost kupovine proizvoda putem košarice ili dogovorom za osobno preuzimanje s oglašivačem. Aplikacija razlikuje između tvrtki i fizičkih osoba koje oglašavaju svoje proizvode, pri čemu tvrtke omogućuju kupovinu proizvoda putem košarice, dok fizičke osobe omogućuju dogovor za osobno preuzimanje i preko košarice. Administrator aplikacije ima ovlasti za kreiranje i upravljanje osobljem koje nadgleda postavljene oglase, dok registrirane tvrtke i fizičke osobe mogu kreirati i upravljati svojim oglasima. Neprijavljeni korisnici mogu pregledavati oglase, ali nemaju mogućnost postavljanja ili upravljanja oglasima. Cilj rada je pružiti funkcionalnu i sigurnu platformu koja korisnicima omogućuje jednostavno upravljanje oglasima i trgovinom putem interneta.

2. POSTOJEĆE APLIKACIJE

U ovom poglavlju prikazan je pregled postojećih web oglasnika koji prikupljaju i prikazuju podatke vezane za prodaju različitih proizvoda.

2.1. Njuškalo

Njuškalo je vodeći online oglasnik u Hrvatskoj, osnovan 2007. godine kao dio Styria Media Group. Platforma omogućava korisnicima da kupuju i prodaju razne proizvode i usluge, od automobila i nekretnina do odjeće, elektronike i kućanskih aparata. Njegova popularnost brzo je rasla, te je danas jedno od najposjećenijih web-mjesta u Hrvatskoj. Jedna od ključnih prednosti Njuškala je jednostavnost korištenja. Korisnici mogu besplatno postaviti oglase, dodavati fotografije i detaljne opise proizvoda. Postoji i mogućnost plaćenih opcija koje omogućuju bolje isticanje oglasa i brže postizanje prodaje. Kategorizacija proizvoda je intuitivna i pregledna, što olakšava pretragu i kupovinu. Kupci mogu pretraživati oglase prema različitim kriterijima, kao što su cijena, lokacija, stanje proizvoda i drugi parametri. Njuškalo također nudi korisnicima sigurnosne savjete za izbjegavanje prevara, kao što su plaćanje prilikom preuzimanja proizvoda i oprez s previše primamljivim ponudama. Slika 2.1. prikazuje izgled Njuškala [1].



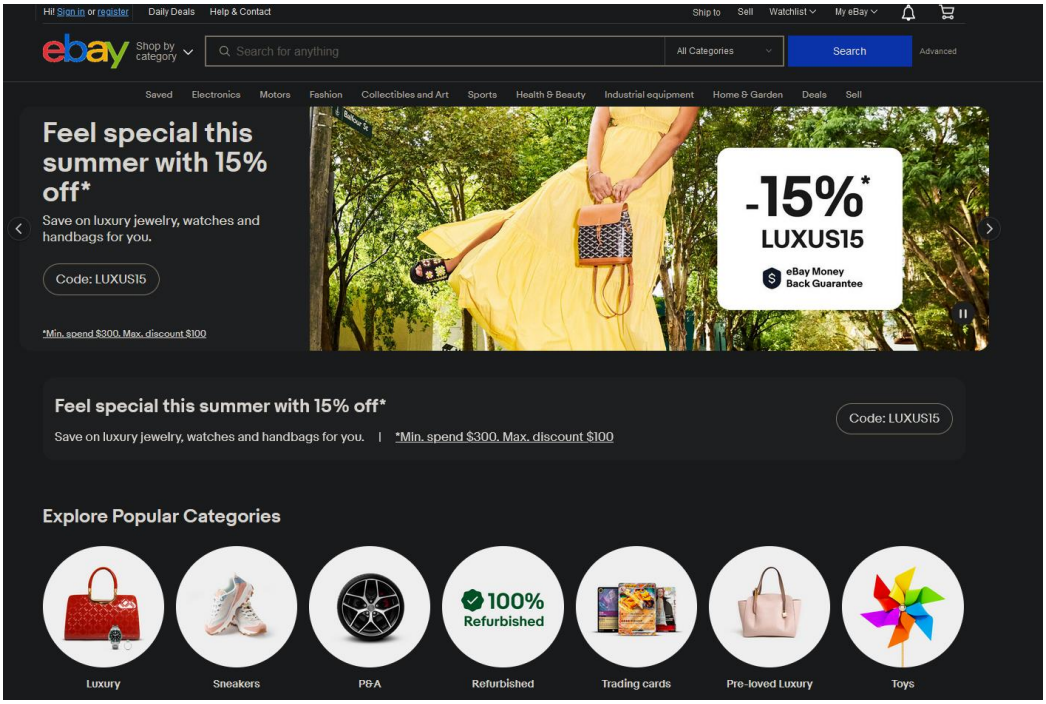
Slika 2.1. *Njuškalo.hr.*

2.2. eBay

eBay je globalna platforma za e-trgovinu osnovana 1995. godine od strane Pierra Omidyara. Sjedište kompanije je u San Joseu, Kalifornija. eBay je postao jedno od najpoznatijih imena u online trgovini, omogućujući korisnicima diljem svijeta da kupuju i prodaju širok spektar proizvoda i usluga. Platforma nudi aukcijske formate kao i opciju "Kupi odmah", pružajući fleksibilnost kako za kupce tako i za prodavače. Više od 25 godina eBay je jedno od vodećih online tržišta na kojem potrošači kupuju sve, od elektronike do modnih detalja i kolekcionarskih predmeta.

Na početku zamišljeno kao platforma za aukcije, ovo tržište je u međuvremenu poraslo u mamuta od 11.7 milijardi US dolara prometa, uslužujući 183 miliona aktivnih kupaca u 25 zemalja [2].

eBay je revolucionirao način na koji ljudi kupuju i prodaju proizvode online. Njegova globalna prisutnost, širok spektar proizvoda i pouzdan sustav ocjenjivanja čine ga jednim od najvažnijih alata u svijetu e-trgovine. Slika 2.2. prikazuje izgled eBaya [3].



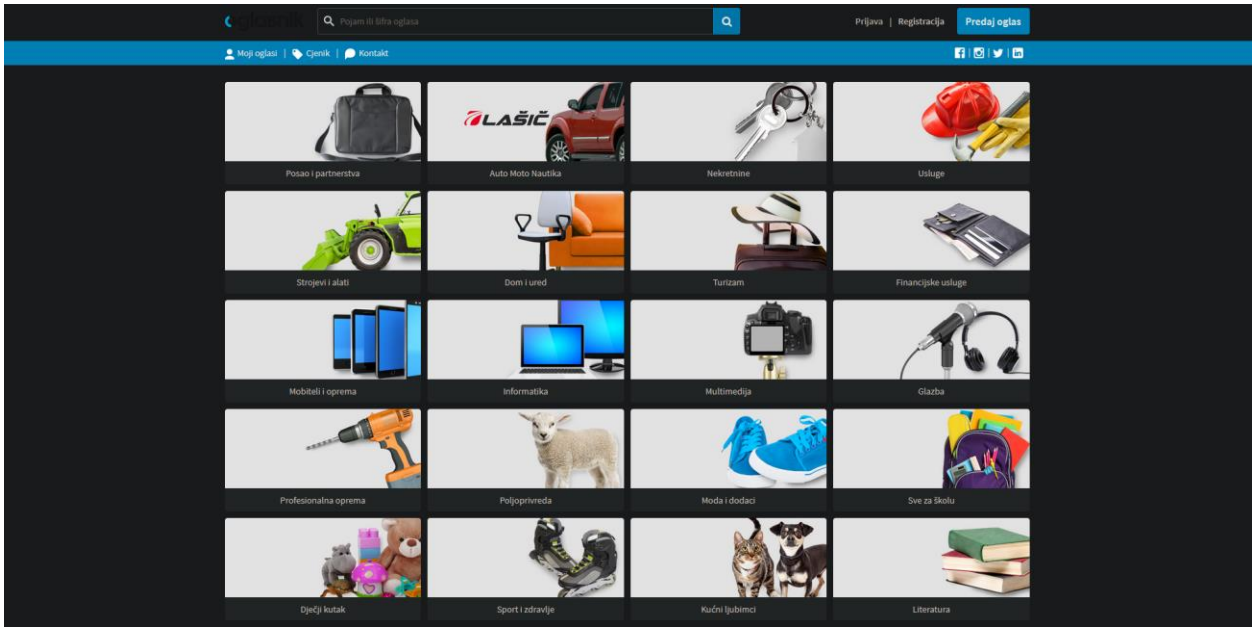
Slika 2.2. Prikaz *eBay stranice.*

2.3. Plavi Oglasnik

Plavi oglasnik je jedna od najpoznatijih i najstarijih platformi za oglašavanje u Hrvatskoj. Osnovan 1989. godine, Plavi oglasnik je započeo kao tjedni tiskani oglasnik, pružajući korisnicima mogućnost da na jednom mjestu pronađu razne oglase, od nekretnina i automobila do poslova i usluga. S vremenom se Plavi oglasnik prilagodio digitalnoj eri, lansirajući svoju online verziju i postajući jedan od najposjećenijih oglasnika u zemlji.

Na internetskoj stranici Oglasnik.hr objavljuje se više od 300.000 novih oglasa svakog mjeseca. Stranice Oglasnika godišnje posjeti više od 4 milijuna jedinstvenih posjetitelja, a više od 125 milijuna učitanih stranica govore u prilog aktivnosti korisnika [4].

Plavi oglasnik pokriva širok spektar kategorija, uključujući nekretnine, vozila, posao, elektroniku, usluge i mnoge druge. Ova raznovrsnost čini ga idealnim mjestom za oglašavanje bilo kojeg proizvoda ili usluge. Bilo da korisnici traže stan za najam, rabljeni automobil, posao ili čak usluge poput popravaka i čišćenja, Plavi oglasnik nudi sve na jednom mjestu. Jedna od značajnih prednosti Plavog oglasnika je i njegova prilagodba mobilnim uređajima. Uz mobilnu aplikaciju, korisnici mogu lako pregledavati i postavljati oglase u pokretu, što dodatno povećava pristupačnost i praktičnost platforme. Slika 2.3. prikazuje izgled Plavog Oglasnika [5].



Slika 2.3. *Plavi Oglasnik.*

3. KORIŠTENE TEHNOLOGIJE PRI IZRADI WEB APLIKACIJE

Web aplikacije su softverski programi koji koriste web tehnologije i standarde za obavljanje zadataka putem internetskog preglednika. Njihova popularnost raste zahvaljujući brojnim prednostima u odnosu na tradicionalne desktop aplikacije, kao što su pristupačnost, lakoća održavanja i mogućnost ažuriranja u stvarnom vremenu. Jedna od ključnih prednosti web aplikacija je njihova pristupačnost. Web aplikacije mogu koristiti razne tehnologije i okvire za razvoj. Popularni *frontend* okviri uključuju React, Angular i Vue.js, dok se za *backend* često koriste Node.js, Django i Ruby on Rails. Ovi alati omogućuju brzu izradu, fleksibilnost i skalabilnost aplikacija. Promjene i poboljšanja mogu se implementirati centralno na serveru, čime korisnici automatski dobivaju najnoviju verziju aplikacije bez potrebe za ručnim ažuriranjem. Ovo je posebno korisno za poslovne aplikacije gdje je ključno osigurati dosljednost i sigurnost softvera.

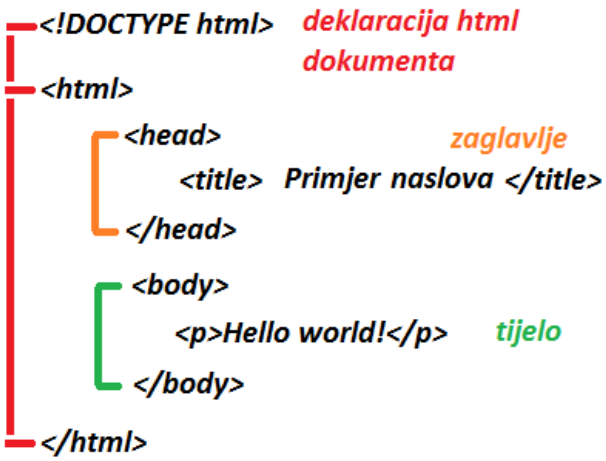
3.1. HTML

HTML (engl. *HyperText Markup Language*) standardni je označni jezik za izradu web stranica i web aplikacija. Pruža sredstva za opisivanje strukture i sadržaja web-dokumenta, a koristi se u kombinaciji s CSS-om (engl. *Cascading Style Sheets*) te JavaScriptom za stvaranje strukture, izgleda i dojma te dinamičkog ponašanja web-mjesta ili web-stranice. HTML je vrlo važan jer je temelj weba. Pruža zajednički skup pravila i standarda koje svi web preglednici i uređaji mogu razumjeti, omogućujući ljudima pristup te dijeljenje informacija preko interneta bez obzira na njihov hardver, softver ili lokaciju [6].

Head sekcija sadrži metapodatke o dokumentu, kao što su naslov stranice (definiran s <title> elementom), karakter set, autor, ključne riječi za pretraživanje i poveznice na vanjske resurse poput stilskih listova (CSS) i skripti (JavaScript).

Body sekcija sadrži stvarni sadržaj stranice, koji se prikazuje korisnicima. Ovdje se koriste različiti elementi za strukturiranje sadržaja, uključujući naslove <h1> do <h6>, paragrafe <p>, liste <ul> i <ol>, slike <img>, poveznice <a> i mnoge druge. Svaki od ovih elemenata može imati atribute koji dodatno specificiraju njegovo ponašanje ili izgled, kao što su *src* za izvore slika ili *href* za URL (engl. *Uniform Resource Locator*) adrese poveznica. Jedna od ključnih

značajki HTML-a je hipertekst, koji omogućuje povezivanje različitih web stranica putem hiperlinkova. Ovo omogućuje korisnicima lako navigiranje između različitih dijelova interneta, čineći informacije pristupačnijima. Slika 3.1. prikazuje strukturu HTML-a [7].



Slika 3.1. *Struktura HTML-a.*

3.2. CSS

CSS je osmišljen je za jednostavno korištenje prilikom dizajniranja internetskih stranica. Uz pomoć CSS-a uređujemo vizualni dio web stranice, kao i doživljaj koji će korisnik osjetiti kada pregledava određenu stranicu i sadržaj na njoj [8].

Jedna od ključnih karakteristika CSS-a su selektori, koji omogućuju odabir specifičnih HTML elemenata na koje se stilovi primjenjuju. Postoje različite vrste selektora, uključujući element selektore (npr. <p> za paragrafe), klasne selektore (npr. .nav za navigacijski izbornik) i ID selektore (npr. #header za zaglavlje stranice). Ovi selektori omogućuju precizno ciljanje elemenata koji trebaju biti stilizirani prema određenim specifikacijama.

CSS također podržava raznovrsne jedinice mjere koje se koriste za određivanje veličina elemenata i prostora na stranici. To uključuje piksele (px), postotke (%), em i rem jedinice koje omogućuju fleksibilnost i prilagodbu dizajna različitim veličinama ekrana i uređajima. Ova prilagodljivost je ključna za reaktivni dizajn, što omogućuje da se web stranice prilagode različitim veličinama ekrana kako bi osigurale optimalno korisničko iskustvo bez obzira na uređaj koji se koristi. Slika 3.2. prikazuje primjer CSS-a.

```css
input[type=text], input[type=password] {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #04AA6D;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
}
```

Slika 3.2. *Primjer CSS-a.*

3.3. Bootstrap

Bootstrap je razvojni okvir otvorenog koda koji pomaže brže definirati web stranice prilagođene po svim veličinama ekrana. Sadrži kolekciju CSS i JavaScript kodova za izradu svih glavnih komponenta koje se često koriste – elemente navigacije, gumbe i pozive za akciju, kontakt obrasce, prilagođavanje tipografije [9].

Jedna od glavnih prednosti Bootstrapa je njegova responsivnost. Razvojni okvir automatski prilagođava izgled web stranica i aplikacija različitim veličinama ekrana, što omogućuje konzistentno korisničko iskustvo na desktop računalima, tabletima i mobilnim uređajima. Ova sposobnost postiže se korištenjem reaktivnih mreža (Flexbox i CSS Grid) i prilagodljivih komponenata koje lako možete prilagoditi prema potrebama vašeg projekta.

Bootstrap također podržava JavaScript komponente kao što su modali, tablice s podacima, sklopivi sadržaj, *tooltips*, *carousel* i drugi interaktivni elementi. Ove komponente dodaju dinamičnost i funkcionalnost web stranicama, što ih čini idealnim izborom za razvoj modernih aplikacija koje zahtijevaju interaktivne elemente. Još jedna značajka koja čini Bootstrap

popularnim je njegova aktivna zajednica i obimna dokumentacija. Postoje brojni resursi, tutorijali, predlošci i gotovi primjeri koji olakšavaju učenje i brzu implementaciju Bootstrapa u vašim projektima. Također, Bootstrap redovito ažurira svoje verzije kako bi pratio najnovije trendove i tehnologije u web dizajnu i razvoju.

3.4. JavaScript

JavaScript je skriptni ili programski jezik koji nam omogućava da web stranicama dodamo kompleksne mogućnosti i da uradimo dinamičnu web stranicu. Za razliku od statičnih web stranica, gdje korisnik neće vidjeti promjene na stranici koja se učitala u pregledniku, dinamične stranice trebaju koristiti druga programska rješenja uz klasični HTML kod. [10]

Jedna od ključnih karakteristika JavaScripta je njegova sposobnost da se izvršava na klijentskoj strani, što znači da se skripte izvršavaju direktno u web pregledniku korisnika. To omogućuje brzo odzivanje aplikacija jer se većina obrade podataka i logike događa lokalno, bez potrebe za stalnim komuniciranjem s poslužiteljem. Također, JavaScript se često koristi za validaciju formi, animacije, manipulaciju DOM-om (engl. *Document Object Model*) i još mnogo toga, čime poboljšava korisničko iskustvo na webu.

JavaScript podržava objektno orijentirano programiranje (OOP) kao i funkcionalno programiranje, što ga čini svestranim jezikom koji se može prilagoditi različitim stilovima programiranja i zahtjevima projekata. OOP omogućuje organizaciju koda u logične entitete (objekte) s definiranim svojstvima i metodama, dok funkcionalno programiranje potiče korištenje funkcija kao prvoklasnih objekata za obradu podataka.

JavaScript je ključan jezik za moderni web razvoj zbog svoje fleksibilnosti, brzine izvođenja i široke primjene. Od osnovnih interaktivnih funkcija do kompleksnih aplikacija i igara, JavaScript ostaje temeljni alat za web dizajnere i programere koji žele stvoriti bogate, dinamične i interaktivne web stranice i aplikacije.

```javascript
function zbroj(a, b) {
    let rezultat = a + b;
    return rezultat;
}
let rezultatZbroja = zbroj(5, 3);
```

Slika 3.3. *Primjer JavaScript funkcije koji zbraja dva broja.*

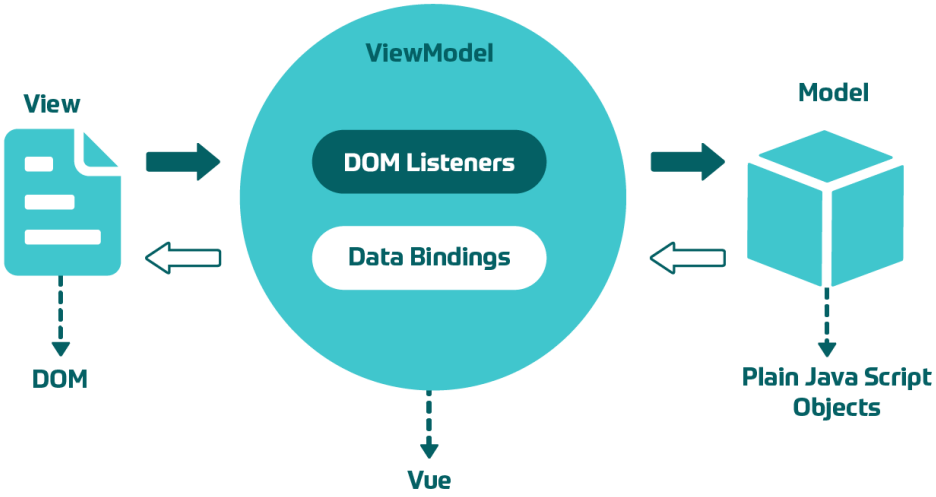
3.5. Vue.js

Vue.js je moderna JavaScript biblioteka koja omogućuje brzo i jednostavno stvaranje korisničkih sučelja za web aplikacije. Biblioteka se fokusira na deklarativni pristup u izgradnji korisničkog sučelja, što znači da se naglasak stavlja na opisivanje onoga što se treba prikazati, a ne na specifikaciju kako se to treba učiniti [11].

Vue.js također podržava reaktivno povezivanje podataka između modela (data) i UI-a (engl. *User Interface*). Kada se podaci promijene, Vue automatski osvježava UI koji je vezan za te podatke, što rezultira bržim i efikasnijim razvojem aplikacija. Ovo se postiže pomoću *v-bind* direktive za povezivanje atributa i *v-bind* za dvosmjerno vezivanje podataka s formama.

Jedna od važnih značajki Vue.js-a je njegova progresivnost. To znači da se Vue može postepeno integrirati u postojeće projekte bez potrebe za prelaskom na potpuno nove tehnologije. Može se koristiti kao zamjena za određene dijelove stranice ili kao potpuno novi UI na postojećoj web aplikaciji, što olakšava inkrementalno uvođenje novih tehnologija.

Vue.js također dolazi s bogatim setom alata i dodataka koji olakšavaju razvoj aplikacija. Primjerice, Vue Router omogućuje upravljanje rutama i navigacijom unutar jednostraničnih aplikacija, dok Vuex pruža globalno stanje za upravljanje podacima u aplikaciji. Ovi dodaci pomažu u organizaciji i upravljanju kompleksnim aplikacijama. Slika 3.4. prikazuje infrastrukturu Vue.js-a [12].



Slika 3.4. *Prikaz infrastrukture Vue.js-a.*

3.6. Visual Studio Code

Visual Studio Code (VS Code) je besplatni, izuzetno popularan tekstualni editor razvijen od strane Microsofta. On se koristi za razvoj raznovrsnih softverskih aplikacija, uključujući web stranice, web aplikacije, mobilne aplikacije, desktop aplikacije i više, te podržava mnoge programerske jezike i tehnologije.

Jedna od ključnih značajki VS Codea je njegova iznimna prilagodljivost i proširivost. Editor dolazi s bogatim skupom funkcionalnosti kao što su sintaksno bojenje, automatsko dovršavanje koda, integrirano upravljanje verzijama (git), podrška za otklanjanje grešaka i još mnogo toga. Osim toga, VS Code omogućuje korisnicima da dodaju dodatke (engl. *Extensions*) preko Visual Studio Code Marketplacea, koji omogućuju prilagodbu editora prema specifičnim potrebama i preferencijama.

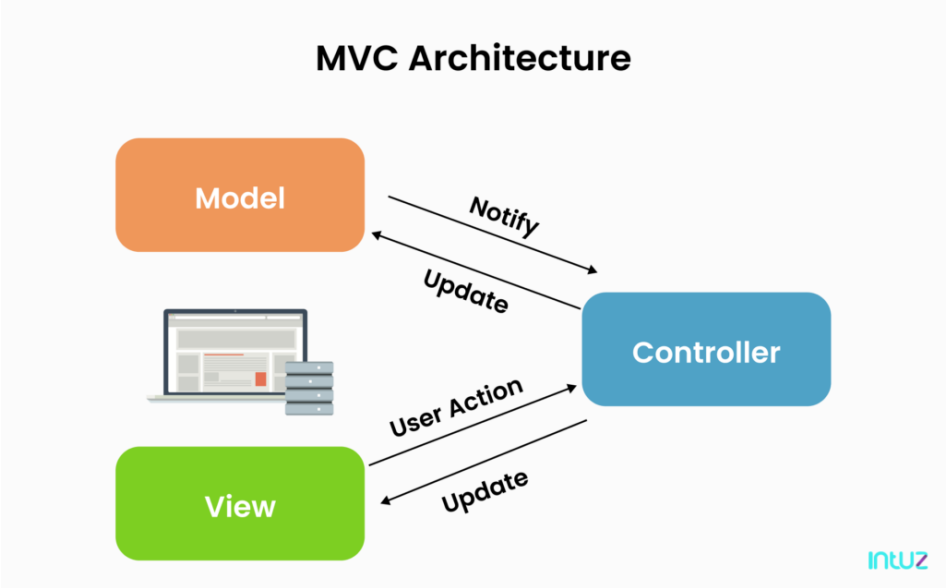
VS Code je također poznat po svojoj brzini i performansama, što ga čini idealnim alatom za razvoj i uređivanje kodova čak i na starijim računalima ili uređajima s ograničenim resursima. Integracija s raznim alatima za razvoj, kao što su terminali, paketni menadžeri, razni *debuggeri* i *linteri*, omogućuje efikasno radno okruženje za programere.

3.7. ASP.NET Core

ASP.NET Core je moderni i visoko učinkovit razvojni okvir otvorenog koda koji se koristi za razvoj *backend* dijela web aplikacija. Razvijen od strane Microsofta, ASP.NET Core pruža programerima moćne alate za izradu sigurnih, skalabilnih i visoko učinkovitih serverskih aplikacija koje podržavaju različite operativne sustave poput Windowsa, Linuxa i macOSa.

Jedna od ključnih karakteristika ASP.NET Corea je neovisnost o platformi. Ovaj razvojni okvir omogućuje izradu aplikacija koje se mogu izvoditi na različitim platformama, što je postignuto zahvaljujući uklanjanju zavisnosti od specifičnih Windows komponenti koje su prisutne u starijim verzijama ASP.NET-a. Ovo čini ASP.NET Core pogodnim izborom za razvoj modernih aplikacija u *cloud* okruženjima kao što su Microsoft Azure, AWS ili Google *Cloud Platform*.

ASP.NET Core podržava razne pristupe razvoju aplikacija kao što su MVC (engl. *Model-View-Controller*). MVC arhitektura omogućuje programerima da organiziraju aplikaciju na način koji razdvaja poslovnu logiku *Model*, korisničko sučelje *View* i upravljačke funkcije *Controller*, što olakšava testiranje, održavanje i proširenje aplikacija. Slika 3.5. prikazuje MVC arhitekturu [13].



Slika 3.5. *Prikaz arhitekture MVC.*

3.8. SQL Server Management Studio 20

SSMS (engl. *SQL Server Management Studio*) je IDE (engl. *Integrated Development Environment*) koje pruža sveobuhvatan skup alata za upravljanje, razvoj i administraciju SQL Server baza podataka. Dizajniran od strane Microsofta, SSMS je postao nezaobilazan alat za sve profesionalce koji se bave bazama podataka, bilo da su to administratori baza podataka, programeri ili analitičari podataka. U ovom seminaru fokusirat ćemo se na ulogu i značaj SSMS-a u upravljanju bazama podataka, kao i na ključne funkcionalnosti koje ovaj alat nudi.

Jedna od glavnih prednosti SSMS-a je njegova sposobnost da centralizira sve aspekte rada s bazama podataka. SSMS omogućava korisnicima da kreiraju, modificiraju i brišu baze podataka, tablice, pohranjene procedure, funkcije i indekse, sve unutar jedinstvenog okruženja. Ovo značajno olakšava administraciju baza podataka, smanjujući potrebu za korištenjem više alata ili skripti.

SSMS također pruža moćne alate za izvođenje upita nad bazama podataka. SQL Server Management Studio dolazi s ugrađenim editorom upita koji omogućava pisanje, izvršavanje i analizu SQL upita. Editor podržava sintaksu u boji, automatsko dovršavanje i mogućnosti pretraživanja koje olakšavaju rad programerima i analitičarima. Osim toga, SSMS omogućava praćenje performansi upita i optimizaciju istih kako bi se postigli najbolji rezultati u radu s velikim količinama podataka.

Jedna od ključnih funkcionalnosti SSMS-a je mogućnost upravljanja sigurnosnim kopijama i vraćanjem baza podataka. Sigurnosne kopije su esencijalne za osiguranje kontinuiteta poslovanja i zaštitu podataka od gubitka ili oštećenja. SSMS omogućava administratorima da lako kreiraju rasporede za automatsko sigurnosno kopiranje i da brzo vraćaju baze podataka u slučaju potrebe. Ova funkcionalnost je od presudnog značaja u osiguravanju visoke dostupnosti i oporavka od katastrofa [14].

4. IZRADA I IZGLED APLIKACIJE

U izradi aplikacije korištene su suvremene tehnologije za razvoj *backend* i *frontend* dijelova sustava. *Backend* je razvijen koristeći .NET Core, gdje su postavljeni osnovni entiteti poput korisnika, oglasa i admina. Implementirani su kontroleri za upravljanje korisnicima i oglasima, omogućujući operacije poput autentikacija, registracije, ažuriranja i brisanja korisnika i oglasa. Poseban naglasak stavljen je na sigurnost, gdje je implementirana autentikacija korisnika pomoću JWT (engl. *JSON Web Token*) tokena. Također, razvijeni su mehanizmi za razlikovanje administrativnih privilegija, pri čemu glavni administrator ima mogućnost upravljanja ostalim administratorima i korisnicima.

Frontend aplikacija razvijena je pomoću Vue.js razvojnog okvira, što je omogućilo izradu dinamičkih korisničkih sučelja. Posebno su implementirane funkcionalnosti za glavnog administratora, koji jedini ima ovlasti za kreiranje novih administratora i upravljanje zaposlenicima.

Osim toga, aplikacija podržava mogućnost registracije i prijave, pri čemu su implementirani mehanizmi za validaciju unosa i osiguranje podataka. Prilikom brisanja korisnika, implementirano je i brisanje svih oglasa povezanih s korisnikom, čime se osigurava cjelovitost podataka u sustavu. Aplikacija je razvijena s ciljem da pruži jednostavno, ali sigurno upravljanje korisnicima i oglasima u jednom sustavu, te je prilagođena za korištenje od strane administrativnog osoblja i korisnika.

4.1. Struktura baze podataka

Za izradu aplikacije korišten je SSMS kao alat za upravljanje bazom podataka. SSMS omogućuje jednostavno upravljanje SQL Server bazama podataka, uključujući kreiranje, uređivanje, dohvaćanje i brisanje podataka te administriranje samih baza podataka.

Tablica *Admins* pohranjuje podatke o administratorima sustava, uključujući korisničko ime, hashiranu lozinku te informaciju o tome je li određeni admin glavni administrator. Tablica *Users* pohranjuje informacije o korisnicima, kao što su korisničko ime, email, datum registracije, te popis oglasnih ID-ova (engl. *Identifier*) koje korisnik posjeduje. Konačno, tablica *Ads* pohranjuje oglase koje su korisnici kreirali, uključujući naslove, opise, cijene, kategorije, lokacije i slike oglasa.

Svaka tablica ima svoj primarni ključ (ID) koji jedinstveno identificira svaki redak u tablici. U relacijskom modelu korištenom za ovaj projekt, podaci su povezani preko ID-ova, primjerice, korisnički ID se koristi kao strani ključ u tablici *Ads* kako bi se pratilo koji korisnik je kreirao određeni oglas.

Na slici 4.1. prikazano je Povezivanje ASP.NET Core aplikacije s bazom podataka koje je realizirano putem *Connection Stringa*, koji se definira u datoteci *appsettings.json*. Na slici je prikazan primjer *Connection Stringa* za spajanje aplikacije na lokalnu SQL Server bazu podataka nazvanu *WebOglasnik*.

```
"ConnectionStrings": {
  "DefaultConnection": "Server=localhost\\SQLEXPRESS;Database=WebOglasnik;Trusted_Connection=true;TrustServerCertificate=true;"
},
```

Slika 4.1. *Prikaz povezivanja s bazom podataka.*

Ovaj *Connection String* omogućuje aplikaciji pristup bazi podataka te izvršavanje SQL upita putem *Entity* razvojnog okvira ili bilo kojeg drugog ORM-a (engl. *Object-Relational Mapping*) koji se koristi u ASP.NET Core aplikaciji. Ovaj pristup je ključan za integraciju podataka unutar aplikacije, omogućavajući *backend* servisu da pohranjuje, dohvaća i upravlja podacima koji se nalaze u SQL Server bazi podataka.

Na slici 4.2. prikazana je C# klasa *Admin* koja služi kao model za administratore unutar aplikacije, korištena u kontekstu *Entity* razvojnog okvira, što je ORM alat u .NET okruženju. Klasa prikazuje strukturu tablice u bazi podataka s poljima *Id, Username, PasswordHash i IsMainAdmin*. Svako polje ima specifične atribute koji osiguravaju jedinstvenost, obaveznost i maksimalnu duljinu unosa.

```
public class Admin
{
    [Key]
    6 references
    public int Id { get; set; }

    [Required]
    [MaxLength(50)]
    5 references
    public string Username { get; set; }

    [Required]
    2 references
    public string PasswordHash { get; set; }

    [Required]
    1 reference
    public bool IsMainAdmin { get; set; } = false;
}
```

Slika 4.2. *C# klasa Admin.*

Na slici 4.3. je prikazana C# klasa *User* koja predstavlja korisnika unutar aplikacije, koristeći se *Entity* razvojnim okvirom, ORM alatom u .NET okruženju. Ova klasa modelira tablicu u bazi podataka s različitim atributima, kao što su *Id, Username, Password, Email, CompanyName, IsIndividual, RegistrationDate, AdIds, i BasketItems*.

```
public class User
{
    [Key]

    public int Id { get; set; }

    [Required]
    [MaxLength(50)]

    public string Username { get; set; }

    [Required]
    [MaxLength(255)]

    public string Password { get; set; }

    [Required]
    [MaxLength(100)]

    public string Email { get; set; }

    [MaxLength(100)]

    public string? CompanyName { get; set; }

    0 references
    public bool IsIndividual { get; set; } = true;

    [Required]
    0 references
    public DateTime RegistrationDate { get; set; } = DateTime.Now;

    12 references
    public List<int> AdIds { get; set; } = new List<int>();

    13 references
    public List<int> BasketItems { get; set; } = new List<int>();
}
```

Slika 4.3. *C# klasa User.*

Atributi poput *Key* označavaju primarni ključ u tablici, dok *Required* osigurava da su određena polja obavezna. *MaxLength* definira maksimalnu duljinu znakova za pojedina polja. *AdIds* je lista identifikatora (ID-ova) oglasa koje je korisnik kreirao. Svaki oglas koji korisnik postavi na platformu bit će pohranjen u ovoj listi, što omogućuje jednostavno povezivanje korisnika s njegovim oglasima. Na taj način, aplikacija može brzo dohvatiti sve oglase koje je

korisnik postavio. *BasketItems* je također lista identifikatora, ali se odnosi na oglase koji se trenutno nalaze u korisnikovoj košarici. Korisnik može pregledavati različite oglase na platformi i dodavati ih u svoju košaricu za kasniju kupovinu ili daljnje akcije.

Na slici 4.4. je prikazana C# klasa *Ad* koja predstavlja oglas unutar aplikacije koristeći *Entity* razvojni okvir kao ORM alat Ova klasa modelira tablicu u bazi podataka s različitim atributima koji definiraju oglas.

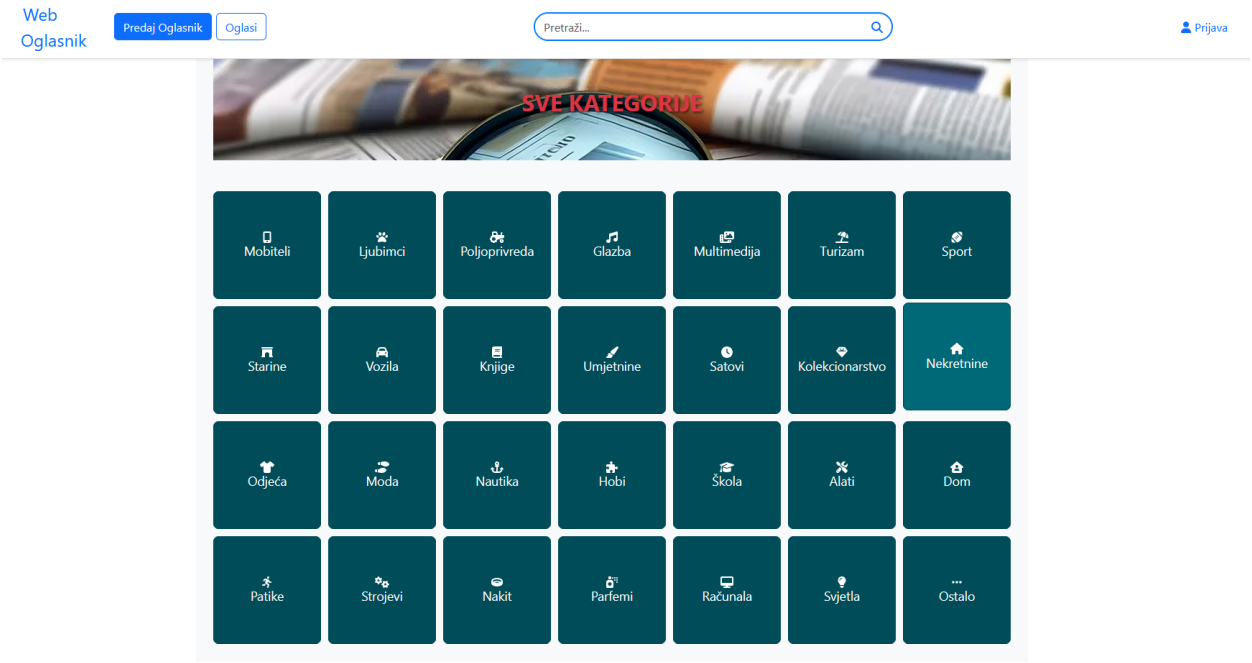
```
public class Ad
{
    [Key]
    5 references
    public int Id { get; set; }

    [Required]
    [MaxLength(255)]
    0 references
    public string Title { get; set; }

    0 references
    public string Description { get; set; }

    [Required]
    [Column(TypeName = "decimal(10, 2)")]
    0 references
    public decimal Price { get; set; }

    [Required]
    0 references
    public CategoryEnum Category { get; set; }

    [Required]
    1 reference
    public int UserID { get; set; }

    [Required]
    0 references
    public DateTime CreatedDate { get; set; } = DateTime.Now;

    [Required]
    0 references
    public bool IsActive { get; set; } = true;

    [Required]
    [MaxLength(255)]
    0 references
    public string Location { get; set; }

    [Required]
    0 references
    public ConditionEnum Condition { get; set; }

    2 references
    public string ImagePaths { get; set; }
}
```

Slika 4.4. *C# klasa Ad.*

UserID je vanjski ključ koji povezuje oglas s korisnikom koji ga je kreirao te omogućava vezu između tablica *User* i *Ad*. *CreatedDate* automatski bilježi datum i vrijeme kada je oglas kreiran *IsActive* je bool vrijednost koja označava je li oglas aktivan ili ne i po zadanom je oglas aktivan. *Location* predstavlja lokaciju na kojoj je oglas postavljen. *Condition* je enum koji predstavlja stanje predmeta u oglasu poput novog ili rabljenog. *ImagePaths* je *string* koji sadrži putanje do slika povezanih s oglasom omogućujući korisnicima dodavanje slika uz oglas.

4.2. Početna stranice web aplikacije

Na slici 4.5. je prikazan izgled početne stranice koji je dizajniran kako bi omogućio korisnicima jednostavno pretraživanje i pregledavanje oglasa prema različitim kategorijama. U gornjem lijevom kutu nalazi se naziv aplikacije *Web Oglasnik*, a odmah do njega su dva dugmeta: *Predaj Oglasnik* i *Oglasi*, koji omogućavaju korisnicima brzo postavljanje novih oglasa ili pregled svih dostupnih oglasa.

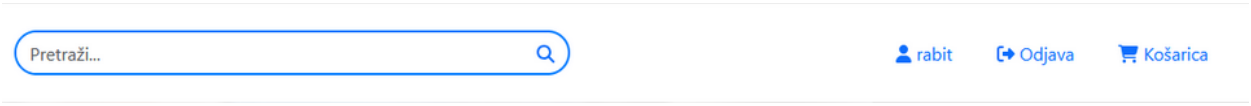


Slika 4.5. *Izgled početne stranice.*

Na sredini stranice nalazi se polje za pretragu s ikonom povećala, što omogućava korisnicima pretraživanje oglasa po ključnim riječima. Ispod polja za pretragu smještena je sekcija *SVE*

KATEGORIJE, koja prikazuje različite kategorije proizvoda ili usluga dostupnih u oglasniku. Svaka kategorija je predstavljena ikonom i nazivom, što korisnicima olakšava navigaciju i pronalaženje specifičnih vrsta oglasa.

Kada korisnik nije prijavljen, na vrhu stranice (u navigacijskoj traci) vidi se opcija *Prijava* omogućujući korisniku da pristupi svom računu ili se registrira kao novi korisnik. Nakon što se korisnik prijavi, navigacijska traka se mijenja i umjesto opcija za prijavu i registraciju prikazuje korisničko ime, kao i dodatne opcije *Odjava* i *Košarica*. Opcija *Košarica* omogućava korisniku pregled artikala koje je dodao u košaricu. Također, korisnik može jednostavno odjaviti se iz aplikacije klikom na dugme *Odjava*. Na slici 4.6. prikazan je izgled navigacijske trake nakon prijave korisnika.

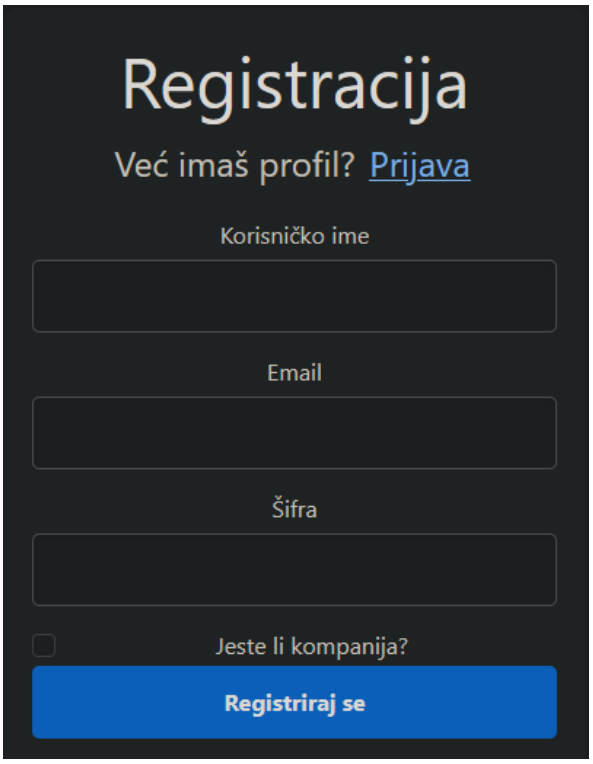


Slika 4.6. *Izgled desne strane navigacijske trake nakon prijave.*

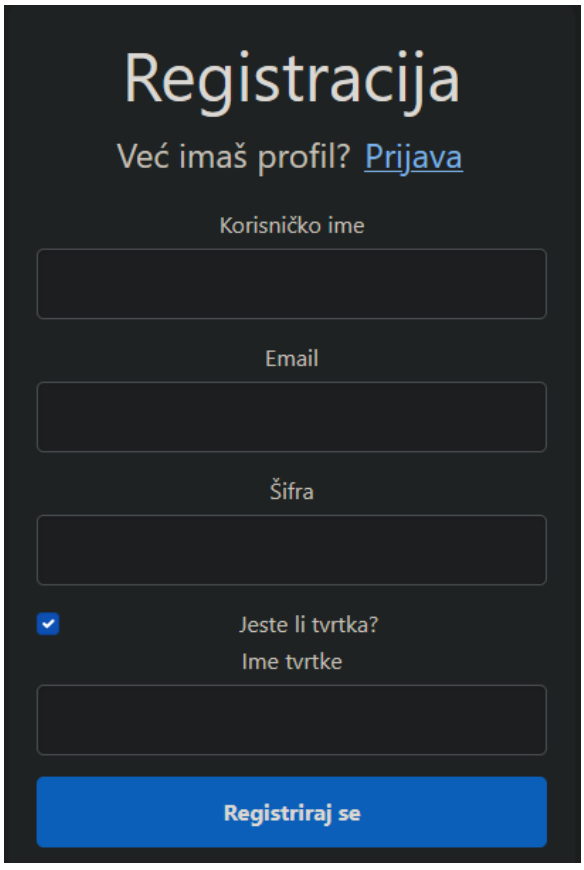
4.3. Registracija korisnika

Na slici 4.7. prikazan je izgled Registracije korisnika u aplikaciji koji omogućava pristup svim funkcionalnostima stranice. Prikazana HTML forma omogućuje korisniku unos osnovnih podataka poput korisničkog imena, email adrese i lozinke. Također, korisnik može označiti opciju *Jeste li tvrtka?* kako bi označio da se registrira kao tvrtka, čime se omogućuju specifične funkcionalnosti prilagođene poslovnim korisnicima. Na slici 4.8. prikazan je izgled kada se registriramo kao tvrtka.

Prilikom popunjavanja forme, korisnik unosi korisničko ime, email i lozinku, a zatim klikom na dugme *Registriraj se* podaci se šalju serveru na obradu. Forma koristi POST metodu, što znači da su podaci sigurni jer nisu vidljivi u URL traci preglednika. Server obrađuje poslane podatke, provjerava njihovu ispravnost, poput valjanosti email adrese i jedinstvenosti korisničkog imena, te nakon uspješne registracije omogućuje korisniku pristup njegovom računu.



Slika 4.7. *Izgled forme za registraciju.*



Slika 4.8. *Izgled forme za registraciju kada se prijavljuje tvrtka.*

Na slici 4.9. kôd demonstrira proces registracije korisnika kroz asinkroni HTTP POST zahtjev prema API (engl. *Application Programming Interface*) kraju. Kôd koristi funkciju *fetch* za slanje podataka korisnika na server, gdje se ti podaci dalje obrađuju i pohranjuju u bazu podataka. Zahtjev se šalje na URL https://localhost:7013/api/User, što predstavlja *API endpoint* za registraciju korisnika. Podaci se šalju u JSON (engl. *JavaScript Object Notation*) formatu kao tijelo zahtjeva, a u zaglavlju zahtjeva specificira se Content-Type: 'application/json' kako bi se serveru naznačilo da se šalju JSON podaci.

Nakon slanja zahtjeva, koristeći *await*, kôd čeka na odgovor servera koji se zatim šalje u JSON format. Ako je registracija uspješna, korisniku se prikazuje poruka *Registracija uspješna!* i automatski se preusmjerava na stranicu za prijavu. Ako server vrati grešku sa statusom 409 *Conflict*, kôd provjerava specifičan kod greške kako bi korisniku pružio točnu povratnu informaciju. Na primjer, ako je korisničko ime već zauzeto (result.code === 4091), korisniku se prikazuje poruka da je ime već u upotrebi. Slično, ako je email adresa već registrirana (result.code === 4092), korisniku se prikazuje odgovarajuća poruka. U slučaju bilo koje druge greške, prikazuje se generička poruka koja informira korisnika o problemu i predlaže pokušaj ponovne registracije kasnije.

```
try {
  const response = await fetch('https://localhost:7013/api/User', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(formData)
  });

  const result = await response.json();

  if (response.ok) {
    alert('Registracija uspješna!');
    this.$router.push('/login');
  } else if (response.status === 409) {
    if (result.code === 4091) {
      this.usernameError = result.Message || 'Ime se već koristi.';
    } else if (result.code === 4092) {
      this.emailError = result.Message || 'Email se već koristi.';
    } else {
      this.generalErrorMessage = result.Message || 'Dogodila se greška, pokušajte ponovo kasnije.';
    }
```

Slika 4.9. *Proces registracije i pozivanje API-ja.*

Funkcija *AddUser* na slici 4.10. napisana je u ASP.NET Core i koristi se za dodavanje novog korisnika u bazu podataka. Funkcija je označena s [HttpPost], što znači da će se pozvati

kada se napravi HTTP POST zahtjev prema API-u na odgovarajuću rutu. Funkcija prvo provjerava postoje li već korisničko ime ili email u bazi koristeći *AnyAsync* metodu nad *Users* tablicom. Ako korisničko ime ili email već postoji, funkcija vraća odgovor s HTTP statusom 409 *Conflict* i odgovarajućom porukom o grešci.

Ako su korisničko ime i email jedinstveni, funkcija koristi *AutoMapper* za mapiranje *UserDto* objekta na *User* entitet. Nakon toga, novi korisnik se dodaje u kontekst baze podataka i sve promjene se spremaju pomoću *SaveChangesAsync*.

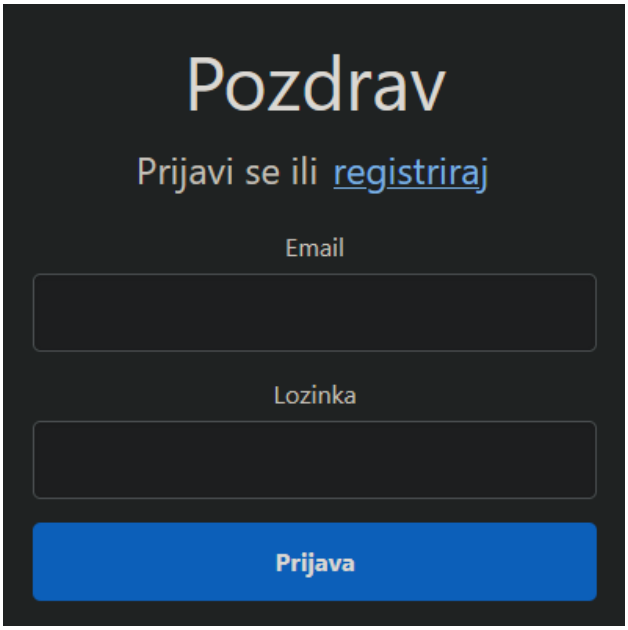
Ako je dodavanje korisnika uspješno, funkcija vraća HTTP status 201 zajedno s podacima o novom korisniku. U slučaju greške, funkcija vraća HTTP status 500 *Internal Server Error* i zapisuje detalje greške u zapisnik.

```
[HttpPost]
0 references
public async Task<ActionResult<UserDto>> AddUser(UserDto newUser)
{
    try
    {
        bool usernameExists = await this._context.Users.AnyAsync(u => u.Username == newUser.Username);
        if (usernameExists)
        {
            return this.Conflict(new { Code = 4091, Message = $"Username '{newUser.Username}' is already in use." });
        }

        bool emailExists = await this._context.Users.AnyAsync(u => u.Email == newUser.Email);
        if (emailExists)
        {
            return this.Conflict(new { Code = 4092, Message = $"Email '{newUser.Email}' is already in use." });
        }

        var user = this._mapper.Map<User>(newUser);
        this._context.Users.Add(user);
        await this._context.SaveChangesAsync();

        this._logger.LogDebug("User added successfully: {@User}.", user);

        return this.CreatedAtAction(
            nameof(this.GetUser),
            new { id = user.Id },
            this._mapper.Map<UserDto>(user));
    }
    catch (Exception ex)
    {
        this._logger.LogError(ex, "An error occurred while adding a new user: {@NewUser}.", newUser);
        return this.StatusCode(500, "An error occurred while processing the request.");
    }
}
```

Slika 4.10. *Funkcija AddUser koja provjerava korisnika i sprema u bazu.*

4.4. Prijava korisnika

Na slici 4.11. prikazana je forma za prijavu korisnika na web aplikaciju, kreirana je jednostavna forma koja omogućava unos email adrese i lozinke. Nakon što korisnik unese svoje podatke, klikom na gumb *Prijava* pokreće se proces autentikacije. Podaci se šalju serveru na obradu putem POST metode, gdje se provjerava ispravnost unesenih podataka. Ako su podaci točni, korisnik se uspješno prijavljuje i preusmjerava na početnu stranicu ili drugu predodređenu rutu unutar aplikacije. U slučaju neuspješne prijave, na ekranu se prikazuje odgovarajuća poruka o grešci, upozoravajući korisnika da je došlo do pogreške u unosu emaila ili lozinke.



Slika 4.11. *Forma za prijavu korisnika.*

Na slici 4.12. je prikazan dio JavaScript koda koji se koristi za autentikaciju korisnika putem POST zahtjeva. Funkcija *handleLogin* prikuplja podatke unosa korisnika, konkretno email i lozinku, te ih zatim pakira u objekt *loginData*. Nakon toga, podaci se šalju na server koristeći *fetch* metodu, koja vrši POST zahtjev prema specificiranoj API ruti na serveru (/api/User/authenticate).

Tijekom ovog procesa, podaci se šalju u JSON formatu uz postavljene odgovarajuće zaglavlje za sadržaj tipa application/json. Nakon što server obradi zahtjev, funkcija čeka odgovor (await response.json()) koji vraća informacije o rezultatu autentikacije. Ako je autentikacija

uspješna, JWT dobiven od servera pohranjuje se u *localStorage* zajedno s korisničkim ID-om i imenom.

```
async handleLogin() {
  const loginData = {
    email: this.email,
    password: this.password,
  };

  try {
    const response = await fetch('https://localhost:7013/api/User/authenticate', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(loginData),
    });

    const result = await response.json();

    if (response.ok) {

      localStorage.setItem('token', result.tokic);
      localStorage.setItem('Id', result.getId);
      localStorage.setItem('user', result.name);
```

Slika 4.12. *Pozivanje funkcije za provjeru korisnika.*

Na slici 4.13. je prikazan dio koda koji služi za generiranje JWT tokena, koji se koristi za autentikaciju korisnika u web aplikacijama. U početku, kreira se instanca *JwtSecurityTokenHandler*, koja je odgovorna za stvaranje i validaciju JWT tokena. Ključ za potpisivanje tokena dohvaća se iz konfiguracije aplikacije (this.configuration["Jwt:Key"]) i koristi se za osiguravanje integriteta tokena.

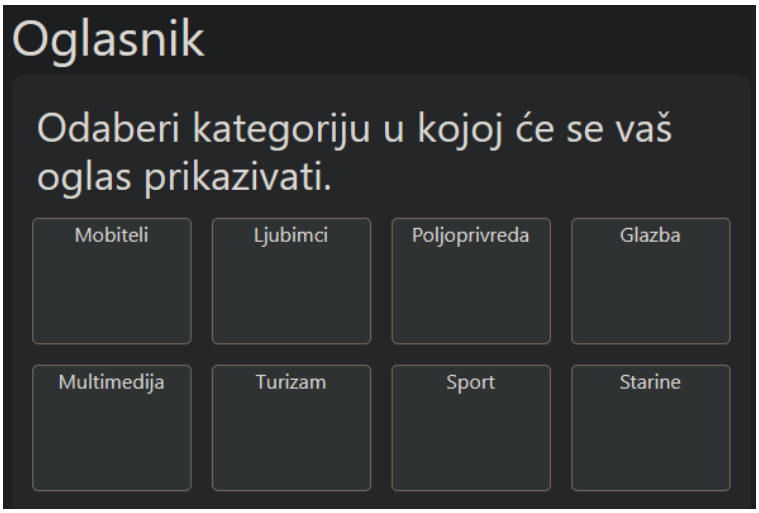
Nakon toga, definira se *SecurityTokenDescriptor*, koji sadrži informacije o korisniku i postavke za generirani token. Ovdje su definirani klijentski podaci poput korisničkog ID-a i emaila, vrijeme isteka tokena (u ovom slučaju postavljeno na jedan sat), te izdavatelj i publika tokena. Token se potpisuje koristeći HMAC SHA256 algoritam, koji osigurava da se token ne može izmijeniti bez ispravnog ključa.

```
var tokenHandler = new JwtSecurityTokenHandler();
var key = Encoding.ASCII.GetBytes(this._configuration["Jwt:Key"]);
var tokenDescriptor = new SecurityTokenDescriptor
{
    Subject = new ClaimsIdentity(new[]
    {
        new Claim(ClaimTypes.NameIdentifier, user.Id.ToString()),
        new Claim(ClaimTypes.Email, user.Email),
    }),
    Expires = DateTime.UtcNow.AddSeconds(3600),
    Issuer = this._configuration["Jwt:Issuer"],
    Audience = this._configuration["Jwt:Audience"],
    SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature),
};
var token = tokenHandler.CreateToken(tokenDescriptor);
var tokenString = tokenHandler.WriteToken(token);

return this.Ok(new
{
    tokic = tokenString,
    name = user.Username,
    getId = user.Id,
    User = this._mapper.Map<UserDto>(user),
});
```

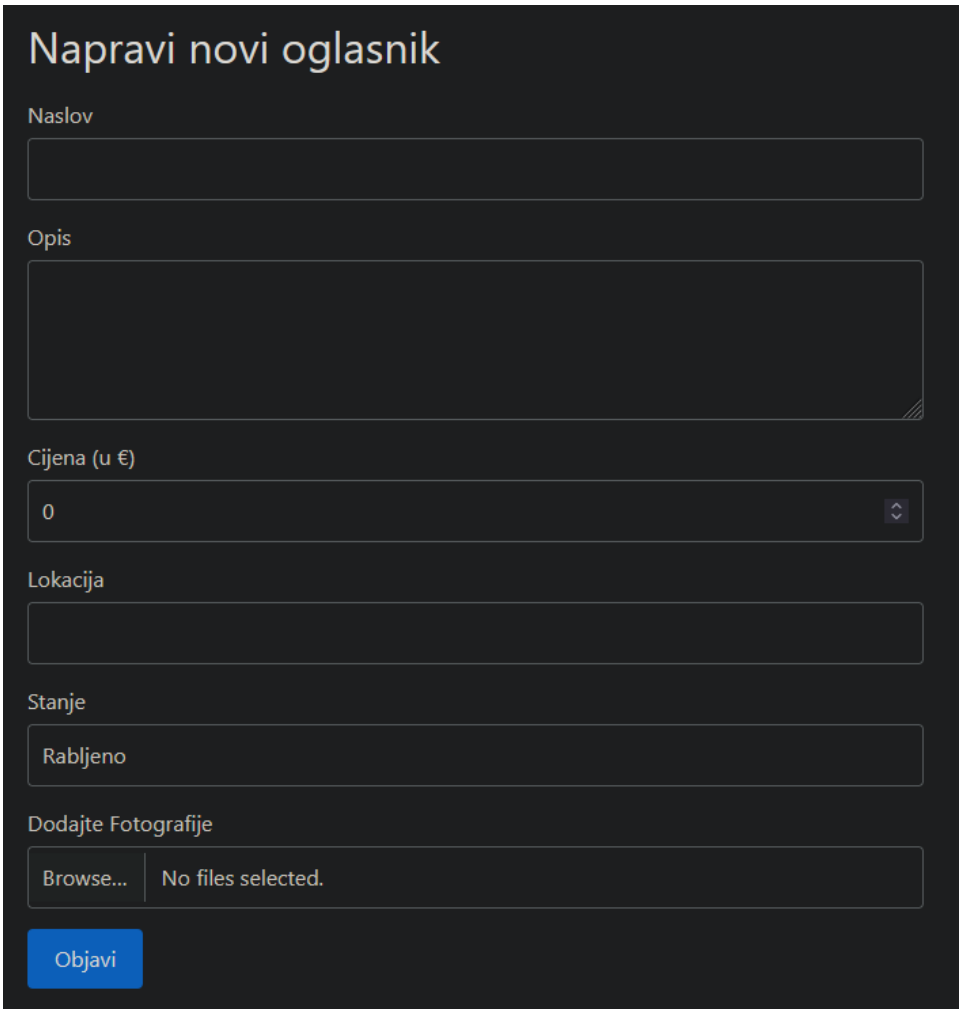
Slika 4.13. *Generiranje JWT tokena.*

4.5. Stvaranje oglasa

Na slici 4.14. je prikazan prvi korak odabira kategorije, korisnik odabire kategoriju u koju će spadati njegov oglas. Na slici 4.15. je prikazan drugi korak, u kojem korisnik unosi detalje svog oglasa. Forma za kreiranje oglasa uključuje polja za unos naslova, opisa, cijene u eurima, lokacije, stanja predmeta (npr. novo ili rabljeno) te opciju za dodavanje fotografija.



Slika 4.14. *Prikaz odabira kategorije.*



Slika 4.15. *Forma za kreiranje oglasa.*

Na slici 4.16. je prikazan dio koda koji se koristi za slanje podataka novog oglasa na server, uz prethodno učitavanje i spremanje fotografija te ažuriranje korisnikovih podataka nakon uspješnog kreiranja oglasa.

Prvo, funkcija *submitAd* koristi *await* ključnu riječ kako bi sinkronizirala operacije koje uključuju učitavanje fotografija i slanje podataka oglasa na server putem POST zahtjeva. Nakon uspješnog slanja oglasa, server vraća ID novo kreiranog oglasa, koji se zatim koristi za ažuriranje popisa oglasa kod korisnika u bazi podataka.

Ako sve operacije prođu uspješno, korisnik dobiva obavijest putem *alert* funkcije da je oglas uspješno kreiran, a stranica se zatim osvježava kako bi prikazala ažurirane podatke. U slučaju greške tijekom bilo kojeg dijela ovog procesa, uhvaćena je iznimka i korisnik biva obaviješten o grešci putem poruke.

```
async submitAd() {
  try {
    const uploadedPaths = await this.uploadPhotos();
    this.ad.imagePaths = uploadedPaths;

    const response = await axios.post('https://localhost:7013/api/Ad', this.ad);
    const newAdId = response.data.id;

    await this.updateUserWithAdId(newAdId);

    alert('Oglasnik uspješno kreiran!');

    window.location.reload();

  } catch (error) {
    alert('Greška prilikom kreiranja oglasa.');
  }
},
```

Slika 4.16. *Slanje podataka oglasa.*

Na slici 4.17. je prikazan *backend* kod koji se koristi za dodavanje novog oglasa u bazu podataka putem POST zahtjeva u ASP.NET Core aplikaciji. Funkcija *AddAd* prima DTO objekt *newAd* koji sadrži sve potrebne podatke za stvaranje novog oglasa. Unutar funkcije, prvo se koristi *AutoMapper* za mapiranje podataka iz DTO objekta u entitet Ad koji odgovara strukturi baze podataka.

Zatim se dobiva korisnički ID iz JWT tokena putem *ClaimTypes.NameIdentifier*, što osigurava da je oglas povezan s ispravnim korisnikom. Ako je korisnički ID prisutan, postavlja se u oglas. Nakon toga, oglas se dodaje u kontekst baze podataka i čeka se asinkrono spremanje promjena u bazu. U slučaju uspjeha, zapisuje se poruka o uspješnom dodavanju oglasa, a funkcija vraća HTTP status 201 *Created*, zajedno s novim oglasom i njegovim ID-om.

U slučaju greške tijekom ovog procesa, ona se hvata u *catch* bloku, gdje se zapisuje detaljna poruka o grešci, a korisniku se vraća status 500 *Internal Server Error* s porukom o pogrešci. Ovaj kod osigurava da se novi oglasi pravilno spremaju u bazu podataka i da su sve operacije propisno zapisane.

```
[HttpPost]
0 references
public async Task<ActionResult<AdDto>> AddAd(AdDto newAd)
{
    try
    {
        var ad = this._mapper.Map<Ad>(newAd);

        var userId = this.User.FindFirst(ClaimTypes.NameIdentifier)?.Value;
        if (userId != null)
        {
            ad.UserID = int.Parse(userId);
        }

        this._context.Ads.Add(ad);

        await this._context.SaveChangesAsync();

        this._logger.LogDebug("Ad added successfully: {@Ad}.", ad);

        return this.CreatedAtAction(
            nameof(this.GetAd),
            new { id = ad.Id },
            this._mapper.Map<AdDto>(ad));
    }
    catch (Exception ex)
    {
        this._logger.LogError(ex, "An error occurred while adding a new ad: {@NewAd}.", newAd);
        return this.StatusCode(500, "Error occurred while processing request.");
    }
}
```

Slika 4.17. *Dodavanje oglasnika u bazu podataka.*

4.6. Pregled svih oglasa

Pregled oglasa omogućuje korisnicima da vide sve dostupne oglase unutar platforme, s opcijom filtriranja prema cijeni i lokaciji. Korisničko sučelje je jednostavno i intuitivno, s jasnim prikazom ključnih informacija. Svaki oglas uključuje naslov, kratak opis, lokaciju, cijenu i datum objave. Uz to, prikazana je i slika koja prati oglas.

Filteri za cijenu i lokaciju omogućuju korisnicima da prilagode prikaz oglasa prema vlastitim preferencijama.

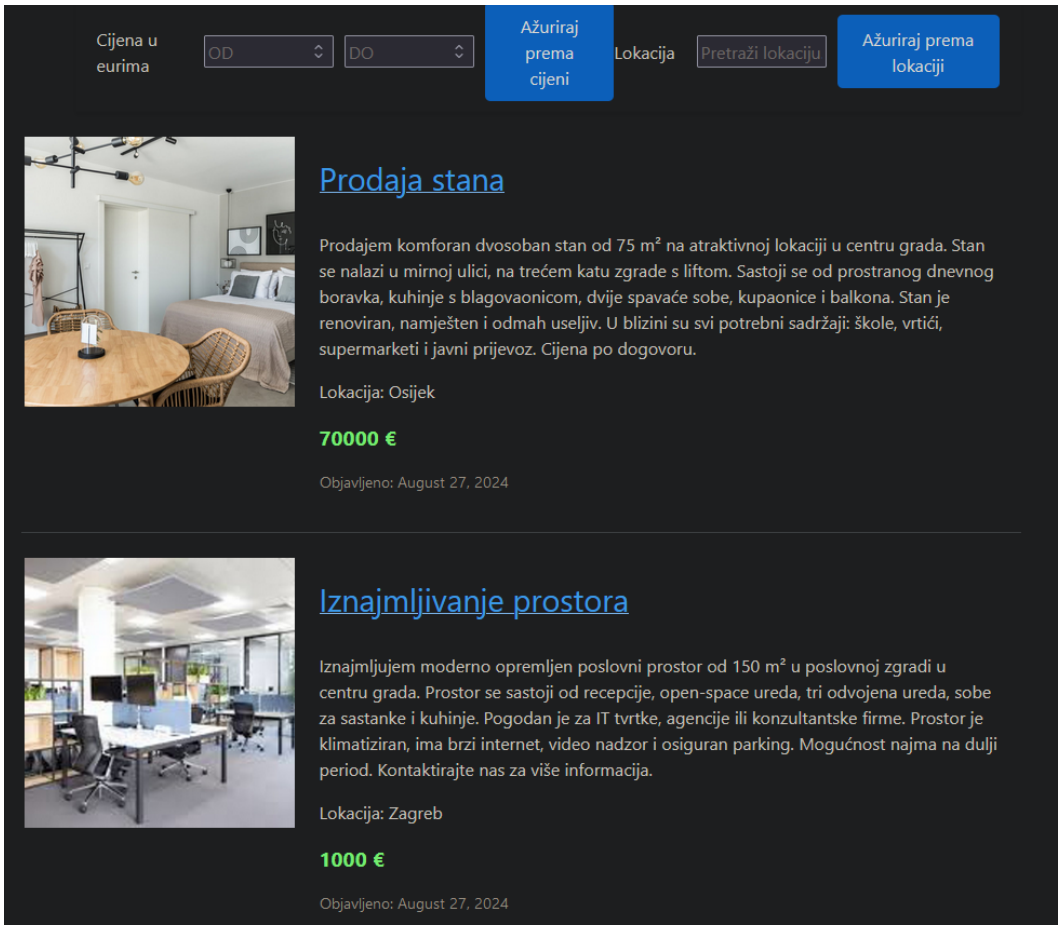
Na slici 4.18. vidi se dio kôda Vue.js komponenti koja upravlja prikazom i filtriranjem oglasa na temelju različitih kriterija kao što su cijena, lokacija i pretraga naslova. Metoda *getImageUrl* služi za dobivanje ispravnog URL-a slike koja je povezana s oglasom. Ako oglas ima dodijeljenu sliku, metoda vraća putanju do te slike, a ako slika nije dostupna, vraća se putanja do zamjenske slike. Ovo osigurava da svaki oglas uvijek ima prikazanu neku sliku, čak i ako nije dodana originalna slika.

Metoda *formatDate* koristi se za formatiranje datuma kada je oglas objavljen. Datum se formatira prema zadanim opcijama koje uključuju prikaz godine, mjeseca i dana, čime se osigurava konzistentan prikaz datuma u ljudima razumljivom formatu.

Metoda *updateFilteredAds* omogućava filtriranje prikazanih oglasa na temelju nekoliko kriterija. Prvo, provjerava se cijena oglasa u odnosu na minimalnu i maksimalnu cijenu koju je korisnik unio. Zatim se filtriraju oglasi na temelju lokacije, uspoređujući unesenu lokaciju s lokacijom oglasa. Na kraju se oglasi filtriraju na temelju naslova oglasa u skladu s unesenim pojmom za pretraživanje. Ova metoda omogućava korisniku da brzo i lako suzi prikaz oglasa prema svojim preferencijama, a rezultati pretrage automatski se prikazuju na prvoj stranici. Na slici 4.19. prikazan je pregled svih oglasa.

```
getImageUrl(imagePath) {
  if (imagePath && imagePath.length > 0) {
    return `../${imagePath}`;
  } else {
    return `../src/photos/placeholder.png`;
  }
},
formatDate(date) {
  const options = { year: 'numeric', month: 'long', day: 'numeric' };
  return new Date(date).toLocaleDateString(undefined, options);
},
updateFilteredAds() {
  this.filteredAds = this.ads.filter(ad => {
    const matchesPrice =
      (this.priceMin === null || ad.price >= this.priceMin) &&
      (this.priceMax === null || ad.price <= this.priceMax);
    const matchesLocation = ad.location.toLowerCase().includes(this.location.toLowerCase());
    const matchesSearchQuery = ad.title.toLowerCase().includes(this.searchQuery.toLowerCase());

    return matchesPrice && matchesLocation && matchesSearchQuery;
  });
  this.currentPage = 1;
},
```

Slika 4.18. *Prikaz dohvaćanje rute slike i filtriranja.*

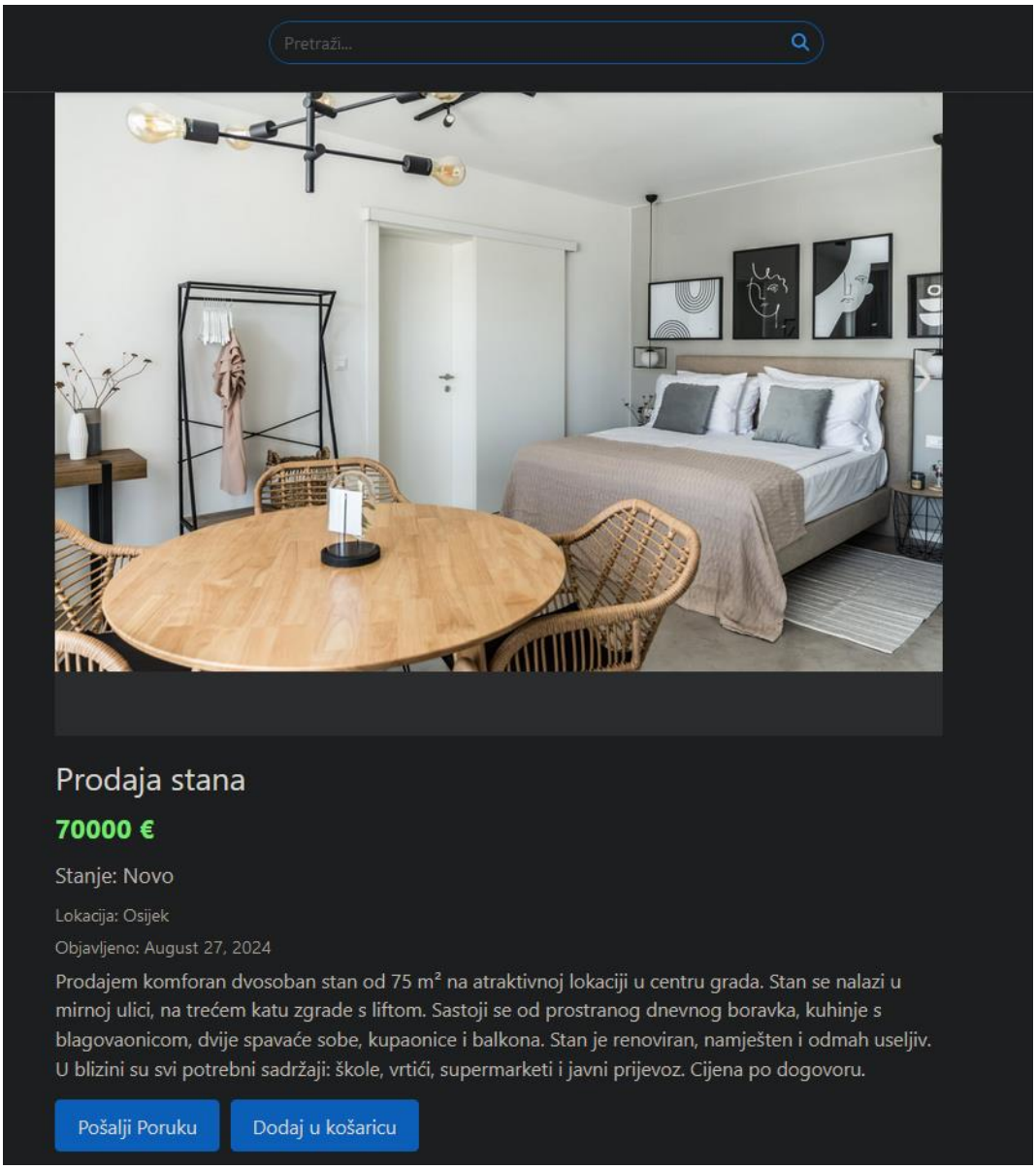


Slika 4.19. *Prikaz svih oglasa.*

4.7. Pregled jednog oglasa

Oglas prikazuje sve ključne informacije kao što su naslov, cijena, stanje, lokacija, datum objave, te opis. Pored toga, korisnicima su dostupne različite opcije ovisno o njihovom statusu.

Ako korisnik pregledava oglas koji je postavila tvrtka, vidjet će samo opciju *Dodaj u košaricu* koja omogućava dodavanje proizvoda u košaricu radi daljnje kupovine. Ako je oglas postavio običan korisnik, na stranici pregleda oglasa prikazuje se dodatna opcija *Pošalji Poruku*. Ova opcija omogućava zainteresiranim korisnicima da izravno kontaktiraju vlasnika oglasa. U slučaju da vlasnik oglasa pregledava svoj oglas, pruža mu se mogućnost da obriše oglas. Ova funkcionalnost omogućuje vlasniku da u svakom trenutku ukloni oglas iz prikaza. S druge strane, ako administrator gleda bilo koji oglas na platformi, također ima opciju da obriše oglas. Na slici 4.20. prikazan je izgled zasebnog oglasa.



Slika 4.20. *Izgled zasebnog oglasa.*

Kod na slici 4.21. prikazuje funkcionalnost dodavanja oglasa u košaricu korisnika unutar aplikacije. Funkcija *addToBasket* koristi se za ažuriranje popisa oglasa koji se nalaze u korisnikovoj košarici. Na početku funkcije provjerava se je li korisnik prijavljen tako da se pokuša dohvatiti *userId* iz lokalne pohrane. Ako korisnik nije prijavljen, prikazuje se upozorenje i preusmjerava ga se na stranicu za prijavu.

Ako je korisnik prijavljen, funkcija pokušava poslati HTTP zahtjev metodom PUT na server kako bi ažurirala korisnikovu košaricu. U tijelu zahtjeva šalje se JSON objekt s popisom *basketItems*, u kojem se nalazi ID oglasa koji se dodaje u košaricu.

```
async addToBasket() {
  const userId = localStorage.getItem('Id');
  if (!userId) {
    alert('Korisnik nije prijavljen. Prijavite se.');
    this.$router.push('/login');
    return;
  }
  try {
    const response = await fetch(`https://localhost:7013/api/User/${userId}/basketitems`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({
        basketItems: [this.ad.id],
      }),
    });

    if (response.ok) {
      alert('Uspješno dodano u košaricu.');
      window.location.reload();
    } else {
      alert('Neuspješno dodano u košaricu.');
    }
```

Slika 4.21. *Prikaz dodavanja oglasnika u košaricu.*

Kod na slici 4.22. prikazuje *backend* funkcionalnost ažuriranja košarice korisnika unutar aplikacije. Metoda *UpdateBasketItems* omogućava ažuriranje popisa oglasa koji se nalaze u košarici korisnika putem HTTP PUT zahtjeva.

Na početku metode, dohvaća se korisnik iz baze podataka na temelju njegovog ID-a. Ako korisnik s danim ID-om ne postoji, vraća se status *NotFound*. Ako korisnik postoji, provjerava se je li popis košarice *BasketItems* inicijaliziran. Ako nije, kreira se nova lista.

Zatim se iterira kroz popis oglasa koje treba dodati u košaricu. Ako oglas već nije u košarici, dodaje se u popis. Nakon toga, iterira se kroz popis oglasa koji se trebaju ukloniti iz košarice, te se uklanjaju iz liste. Nakon svih izmjena, promjene se spremaju u bazu podataka pozivom *SaveChangesAsync*.

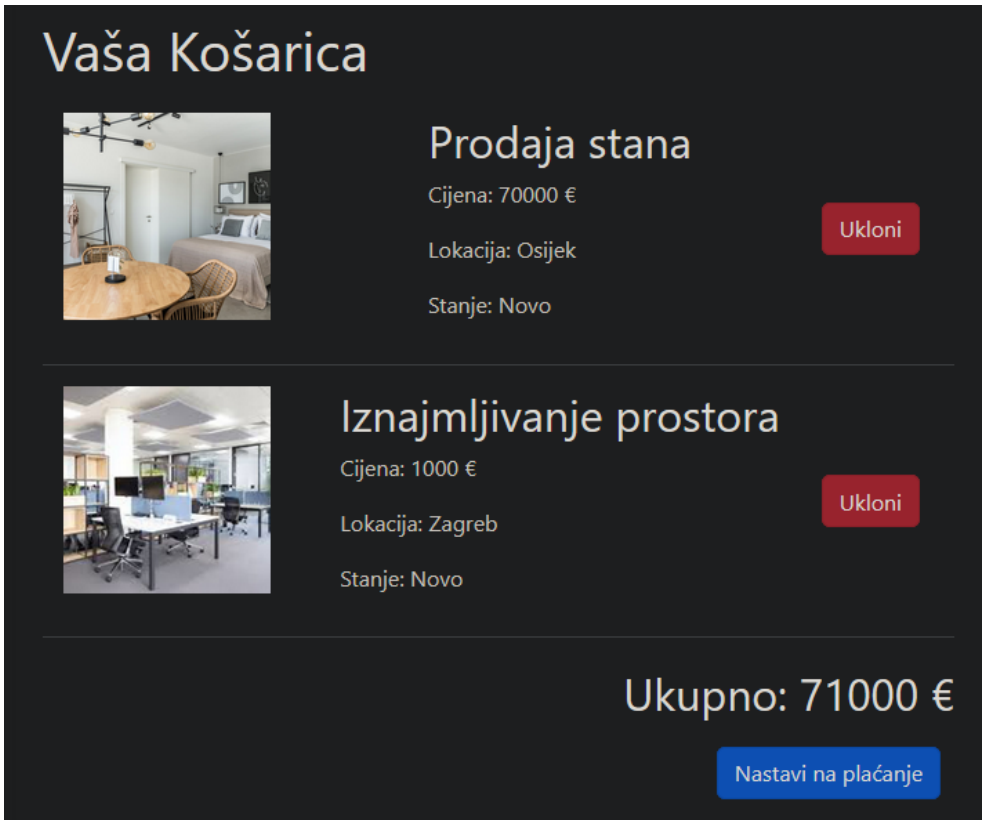
```
[HttpPut("{id}/basketitems")]
0 references
public async Task<ActionResult> UpdateBasketItems(int id, [FromBody] UpdateBasketItemsDto updateBasketItemsDto)
{
    try
    {
        var dbUser = await this._context.Users.FirstOrDefaultAsync(u => u.Id == id);
        if (dbUser == null)
        {
            this._logger.LogWarning("User with ID {Id} not found while updating basket items.", id);
            return this.NotFound();
        }

        if (dbUser.BasketItems == null)
        {
            dbUser.BasketItems = new List<int>();
        }
        foreach (var basketItemId in updateBasketItemsDto.BasketItems)
        {
            if (!dbUser.BasketItems.Contains(basketItemId))
            {
                dbUser.BasketItems.Add(basketItemId);
            }
        }
        foreach (var basketItemId in updateBasketItemsDto.BasketItemsToRemove)
        {
            dbUser.BasketItems.Remove(basketItemId);
        }

        await this._context.SaveChangesAsync();

        this._logger.LogDebug("User with ID {Id} updated successfully with new basket items.", id);
        return this.NoContent();
    }
```

Slika 4.22. *Ažuriranje korisnika u bazi podataka.*

4.8. Pregled košarice

Korisnici mogu pregledati oglase koje su dodali u svoju košaricu prije nego što završe kupnju. Svaki oglas u košarici prikazuje osnovne informacije, uključujući naslov, cijenu, lokaciju i stanje proizvoda. Pored svakog oglasa nalazi se gumb *Ukloni* koji omogućava korisniku da ukloni oglas iz košarice.

Ukupna cijena svih oglasa u košarici automatski se izračunava i prikazuje na dnu stranice, zajedno s gumbom *Nastavi na plaćanje* koji vodi korisnika prema završetku transakcije. Ako je košarica prazna, korisnik će vidjeti poruku koja ga obavještava o tome. Na slici 4.23. prikazan je izgled košarice.



Slika 4.23. *Pregled košarice korisnika.*

Kod na slici 4.24. prikazuje funkcionalnost obrade plaćanja prilikom kupovine oglasa u košarici unutar web aplikacije. Kada korisnik pritisne gumb *Nastavi na plaćanje*, pokreće se *checkout* metoda. Prvo se provjerava je li korisnikov ID pohranjen u *localStorage*. Ako ID nije prisutan, metoda odmah prekida izvršavanje, čime se sprječava nastavak postupka bez valjanih korisničkih podataka.

Ako je korisnik prijavljen, pokreće se pokušaj slanja zahtjeva prema API-ju kako bi se obradilo plaćanje. HTTP PUT zahtjev šalje se na specifični *API endpoint* (/api/User/{userId}/checkout), koristeći *userId* za identifikaciju korisnika..

Nakon što server obradi zahtjev, aplikacija provjerava je li odgovor uspješan. Ako je plaćanje uspješno obavljeno, korisniku se prikazuje poruka potvrde, a košarica se prazni resetiranjem *basketItems* i *basketAds* na prazne nizove. U suprotnom, ako dođe do greške, korisniku se prikazuje upozorenje s obavijesti o grešci.

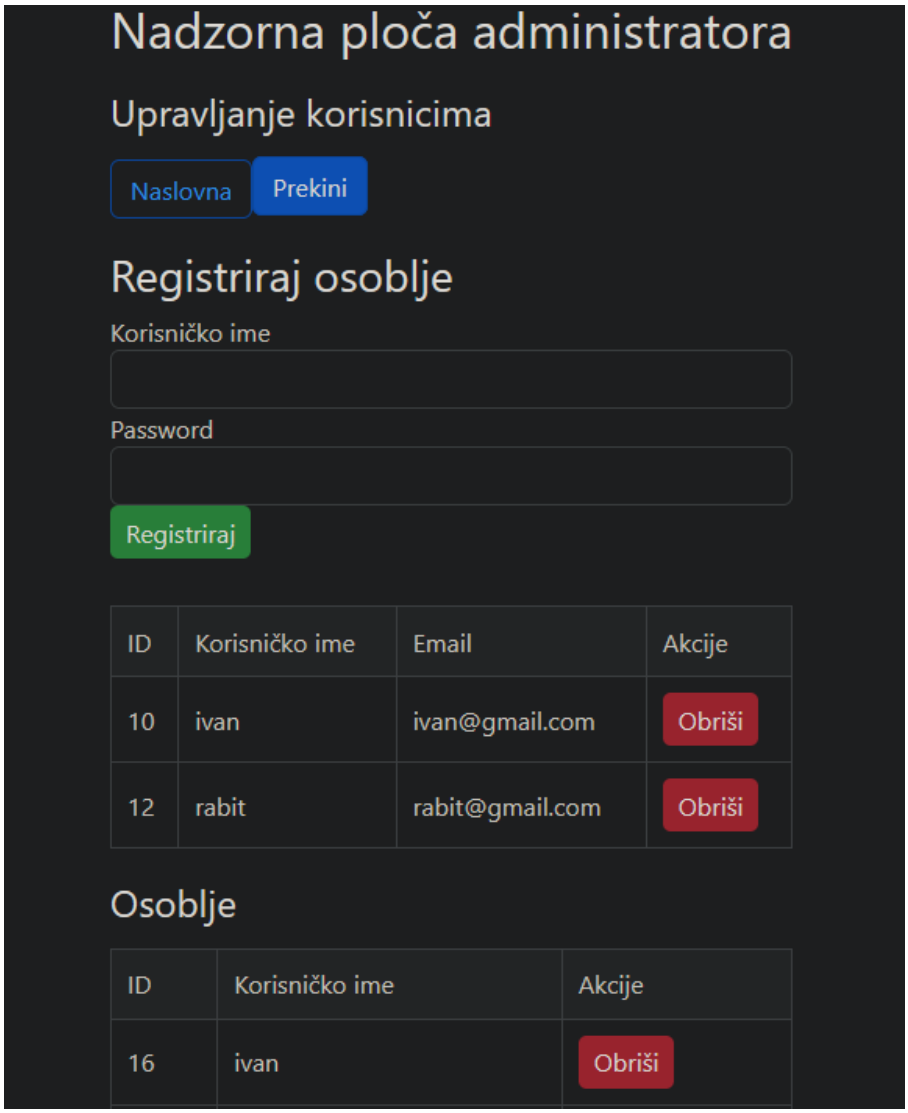
```
async checkout() {
  const userId = localStorage.getItem('Id');
  if (!userId) return;

  try {
    const response = await fetch(`https://localhost:7013/api/User/${userId}/checkout`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json',
      },
    });

    if (response.ok) {
      alert("Plaćanje uspješno obavljeno!");
      this.basketItems = [];
      this.basketAds = [];
    } else {
      alert("Došlo je do greške prilikom plaćanja.");
    }
  } catch (error) {
  }
}
```

Slika 4.24. *Prikaz obrade plaćanja.*

4.9. Administrator i osoblje

Na slici 4.25. prikazana je nadzorna ploča administratora koja omogućava administratoru pregled svih registriranih korisnika te dodavanje novih članova osoblja. Administrator, koji ima *ismainadmin* postavljen na *true*, može registrirati nove članove osoblja s određenim ovlastima. Također, administrator ima mogućnost brisanja bilo kojeg korisnika ili člana osoblja.

Osoblje ima ograničene ovlasti u usporedbi s administratorom. Članovi osoblja mogu pregledavati samo korisnike i njihove podatke, ali nemaju mogućnost registracije novih članova osoblja. Međutim, članovi osoblja imaju ovlast brisati oglase kada kliknu na njih.



Slika 4.25. *Nadzorna ploča administratora*.

5. ZAKLJUČAK

U sklopu ovog završnog rada izrađena je web aplikacija koja omogućuje postavljanje i pregledavanje oglasa. Aplikacija pruža korisnicima jednostavno sučelje za postavljanje oglasa, dok omogućava kupovinu proizvoda putem košarice ili osobnim dogovorom s oglašivačem. Implementirana je i funkcionalnost za administratore koji mogu upravljati korisnicima i nadzirati sve postavljene oglase, čime se osigurava sigurnost i integritet platforme. Korisnicima koji se registriraju kao tvrtke omogućeno je dodatno prilagođeno iskustvo kupovine i prodaje, dok fizičke osobe mogu koristiti osnovne funkcionalnosti aplikacije. Aplikacija nudi i mogućnost uklanjanja oglasa od strane administrator, njegovog osoblja ili samih korisnika, što dodatno doprinosi fleksibilnosti i jednostavnosti upravljanja oglasima. Moguća buduća poboljšanja uključuju dodatne funkcionalnosti, poput automatskog obavještavanja korisnika o promjenama u statusu oglasa ili integraciju s vanjskim sustavima plaćanja, čime bi se dodatno unaprijedila korisnička iskustva. Izradom ove aplikacije demonstrirano je kako moderna web tehnologija može omogućiti razvoj učinkovitih i korisnički orijentiranih rješenja za svakodnevne poslovne i privatne potrebe.

LITERATURA

[1] https://www.njuskalo.hr/, Datum pristupanja 10.06.2024

[2] https://blog.payoneer.com/rs/home-page-rs/e-trgovci/online-trzista/vodic-za-prodaju-ebay/, Datum pristupanja 10.06.2024

[3] https://www.ebay.com/, Datum pristupanja 10.06.2024.

[4] https://blog.stackpath.com/web-application/, Datum pristupanja 12.06.2024

[5] https://www.oglasnik.hr/info/o-nama, Datum pristupanja 12.06.2024

[6] https://www.petarzrinski.hr/sto-je-html/, Datum pristupanja 12.06.2024

[7] http://www.webtech.com.hr/html.php, Datum pristupanja 12.06.2024

[8] https://dir.hr/sto-je-css/, Datum pristupanja 13.06.2024

[9] https://marko-stimac.com/bootstrap-5/#sto-je-bootstrap, Datum pristupanja 13.06.2024

[10] https://dir.hr/sto-je-javascript/, Datum pristupanja 15.06.2024

[11] https://programiranje.com.hr/vuejs, Datum pristupanja 15.06.2024

[12] https://habr.com/en/articles/500340/, Datum pristupanja 15.06.2024

[13] https://lvivity.com/advantages-of-using-asp-net-core, Datum pristupanja 15.06.2024

[14] https://www.c-sharpcorner.com/article/sql-server-management-studio, Datum pristupanja 29.08.2024

SAŽETAK

Cilj ovog rada je izrada web aplikacije koja omogućava predaju, pregledavanje i kupovinu oglasa. Aplikacija je razvijena korištenjem frontend tehnologije Vue.js i *backend* tehnologije ASP.NET Core, dok je za upravljanje bazom podataka korišten SQL Server Management Studio. U aplikaciji se razlikuju dva tipa korisnika: tvrtke i fizičke osobe, pri čemu tvrtke imaju mogućnost prodaje proizvoda putem košarice, dok fizičke osobe također imaju mogućnost prodaje proizvoda putem košarice ili dogovaraju kupnju osobno preko poruka. Aplikacija također omogućava administraciju korisnika i oglasa, pri čemu administrator može kreirati osoblje koje nadgleda oglase. Svi oglasi, korisnici i transakcije pohranjuju se u SQL bazu podataka. Rad pruža uvid u korištene tehnologije, opisuje strukturu baze podataka te objašnjava funkcionalnosti aplikacije, uključujući registraciju korisnika, upravljanje oglasima, i postupak kupovine.

Ključne riječi: ASP.NET Core, baza podataka, oglasnik, Vue.js, web aplikacija

ABSTRACT

Title: Web application for classifieds

The goal of this thesis is to develop a web application that facilitates the submission, browsing, and purchasing of advertisements. The application is built using Vue.js for the frontend and ASP.NET Core for the backend, with SQL Server Management Studio utilized for database management. The application distinguishes between two types of users: companies and individuals. Companies are enabled to sell products through a shopping cart, while individuals arrange purchases directly with the advertiser. The application also supports user and advertisement management, where an administrator can create staff to oversee the ads. All advertisements, users, and transactions are stored in an SQL database. The thesis provides insights into the technologies used, describes the database structure, and explains the application's functionalities, including user registration, advertisement management, and the purchasing process.

Keywords: advertisement, ASP.NET Core, database, shopping cart, Vue.js, web application