Aplikacija za pronalazak cimera - Roomy

Marjanović, Saša

Undergraduate thesis / Završni rad

2023

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:297277

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni studij

APLIKACIJA ZA PRONALAZAK CIMERA - ROOMY

Završni rad

Saša Marjanović

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 15.09.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Saša Marjanović
Studij, smjer:	Programsko inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4537, 27.07.2020.
OIB Pristupnika:	46147059181
Mentor:	izv. prof. dr. sc. Ivica Lukić
Sumentor:	,
Sumentor iz tvrtke:	
Naslov završnog rada:	Aplikacija za pronalazak cimera - Roomy
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rad:	Napraviti Angularu i C# aplikaciju za pronalazak cimera. Aplikacija ima mogućnost kreiranja profila u kojoj korisnik uređuje postavke. Korisnik navodi raspoživi iznos za stanarinu, grad, karakteristike stana i cimera koje traži. Imati mogućnost dopisivanja unutar stranice, filter prema određenim kriterijima i navikama. Treba postojati arhiva svih najmova koje je korisnik imao, kao i arhiva najmova stanodavca. Tema rezervirana za: Saša Marjanović
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	15.09.2023.
Datum potvrde ocjene od strane Odbora:	24.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 25.09.2023.

Ime i prezime studenta:	Saša Marjanović
Studij:	Programsko inženjerstvo
Mat. br. studenta, godina upisa:	R4537, 27.07.2020.
Turnitin podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Aplikacija za pronalazak cimera - Roomy**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivica Lukić

i sumentora ,

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

Potpis studenta:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 2

2. PREGLED PODRUČJA TEME 3

2.1. Pregled sličnih rješenja 3

2.2. Prijedlog rješenja 6

3. ARHITEKTURA RJEŠENJA I KORIŠTENE TEHNOLOGIJE 7

3.1. Zahtjevi na aplikaciju 7

3.2. Prijedlog rješenja arhitekture 7

3.2.1. Arhitektura rješenja iskustva neprijavljenog korisnika 7

3.2.2. Arhitektura rješenja iskustva stanodavca 8

3.2.3. Arhitektura rješenja iskustva stanara 9

3.3. Pregled korištenih tehnologija 9

3.3.1. ASP.NET Framework 9

3.3.2. Entity Framework 10

3.3.3. REST Api 11

3.3.4. Angular 11

4. PROGRAMSKO RJEŠENJE APLIKACIJE 13

4.1. Postavljanje razvojne okoline 13

4.2. Kreiranje i personalizacija profila cimera i stanodavaca 14

4.3. Kreiranje i upravljanje oglasima 15

4.4. Filtriranje i pretraga oglasa i cimera 18

4.5. Komunikacija između korisnika 19

4.6. Arhiviranje najmova 20

5. ZAKLJUČAK 23

LITERATURA 24

SAŽETAK 25

ABSTRACT 26

ŽIVOTOPIS 27

1. UVOD

U današnje digitalno doba pronalazak stana i cimera za mnoge može biti izazov. Tradicionalni oglasi često ne pružaju dovoljno informacija o stanovima i nedostaju im značajke interaktivnosti i personalizacije koje bi olakšale pronalaženje savršenog stana ili cimera. Stoga je cilj ovog rada razvoj Angular i C# aplikacije koja korisnicima omogućuje pronalaženje stanova i cimera na jednostavan i učinkovit način.

Jedna od glavnih prednosti modernijeg načina oglašavanja jeste personalizacija. Korisnici će imati priliku kreirati vlastiti profil na kojem će moći objaviti svoje zahtjeve. Na primjer, mogu navesti iznos koji je dostupan za najam, željeni grad, karakteristike stana i želje cimera. Aplikacija će nuditi više sadržaja od uobičajenih reklama jer će korisnicima pružati dodatne informacije o svakoj sobi, kao i konkretne podatke o preferencijama i navikama korisnika.

Druga ključna prednost je izravna komunikacija između korisnika unutar aplikacije. Korisnici će putem aplikacije moći razmjenjivati informacije i dogovarati se o zajedničkom stanovanju. Osim toga, tu su i filteri za pretragu stanova i cimera prema uvjetima i navikama korisnika. Nadalje, popis prethodnih najmova omogućuje korisnicima da prate svoja prethodna iskustva i daju povratne informacije o potencijalnoj suradnji.

Svrha aplikacije je olakšati pronalazak stanova i cimera u gradovima diljem Hrvatske. Naglasak je na personalizirane i detaljne informacije o stanovima i cimerima, dajući korisnicima sveobuhvatan pregled njihovih izbora. Kroz interakciju, filtriranje i komunikaciju korisnici će imati veće šanse pronaći idealan stan ili cimera koji odgovara njihovim potrebama i preferencijama.

Ovaj rad se sastoji od tri ključna poglavlja. U poglavlju pod naslovom *Pregled područja teme* obrađene su tri slične aplikacije i opisane njihove značajke. Na temelju tih značajki date su smjernice za budući razvoj aplikacije. U poglavlju *Arhitektura rješenja i korištene tehnologije* detaljno su opisane funkcionalnosti prilagođene različitim tipovima korisnika. Nadalje, opisan je specifičan dizajn baze podataka za svaku od tih kategorija korisnika. U drugom dijelu ovog poglavlja prikazani su programski jezici, okviri i arhitekture koji će se koristiti za implementaciju prethodno navedenih rješenja. Poglavlje *Programsko rješenje aplikacije* fokusirano je na konkretnu implementaciju funkcionalnosti. U ovom dijelu rada svaka funkcionalnost je detaljno objašnjena uz prikazani kod ili korisničko sučelje.

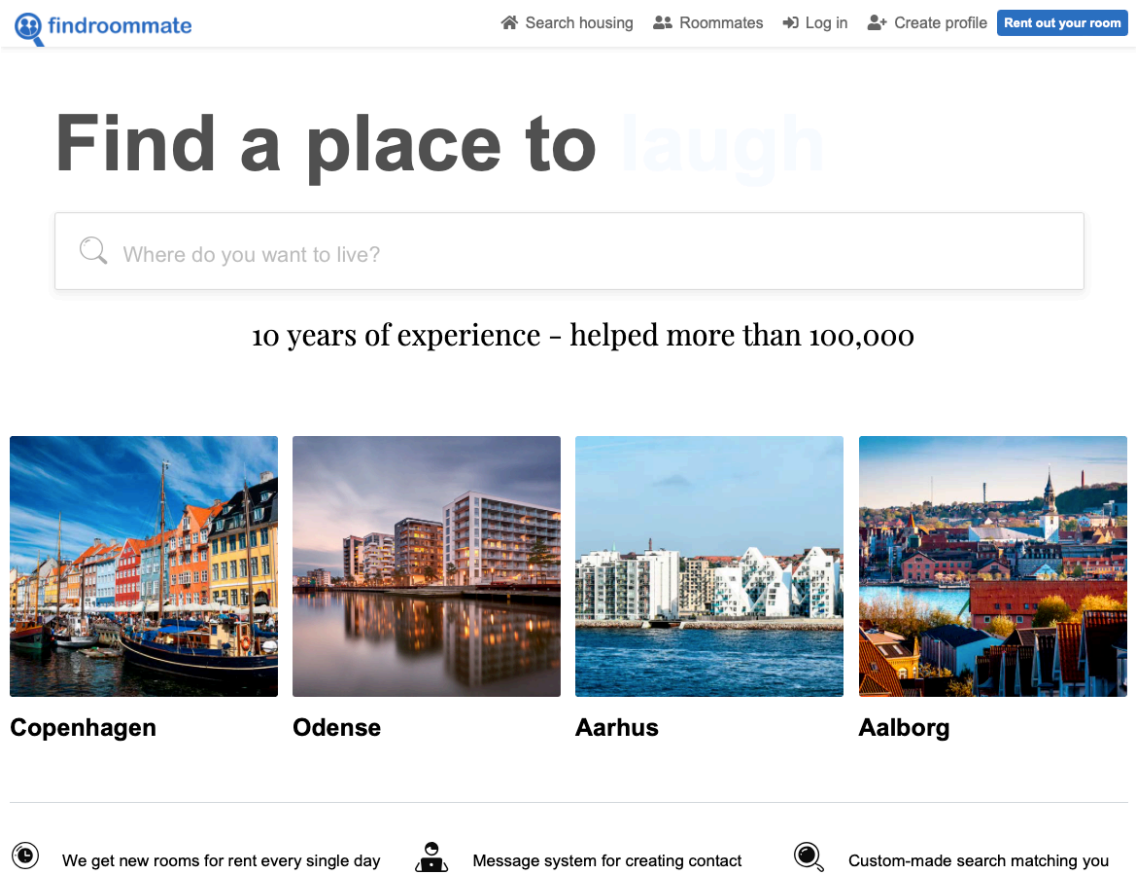
1.1. Zadatak završnog rada

Zadatak ovog završnog rada je napraviti Angular i C# aplikaciju za pronalazak cimera. Aplikacija treba imati mogućnost kreiranja profila u kojoj korisnik uređuje postavke. Korisnik navodi raspoloživi iznos za stanarinu, grad, karakteristike stana i cimera koje traži. Mogućnosti aplikacije trebaju uključiti dopisivanje unutar stranice te filter prema određenim kriterijima i navikama. Treba postojati arhiva svih najmova koje je korisnik imao, kao i arhiva najmova stanodavca.

2. PREGLED PODRUČJA TEME

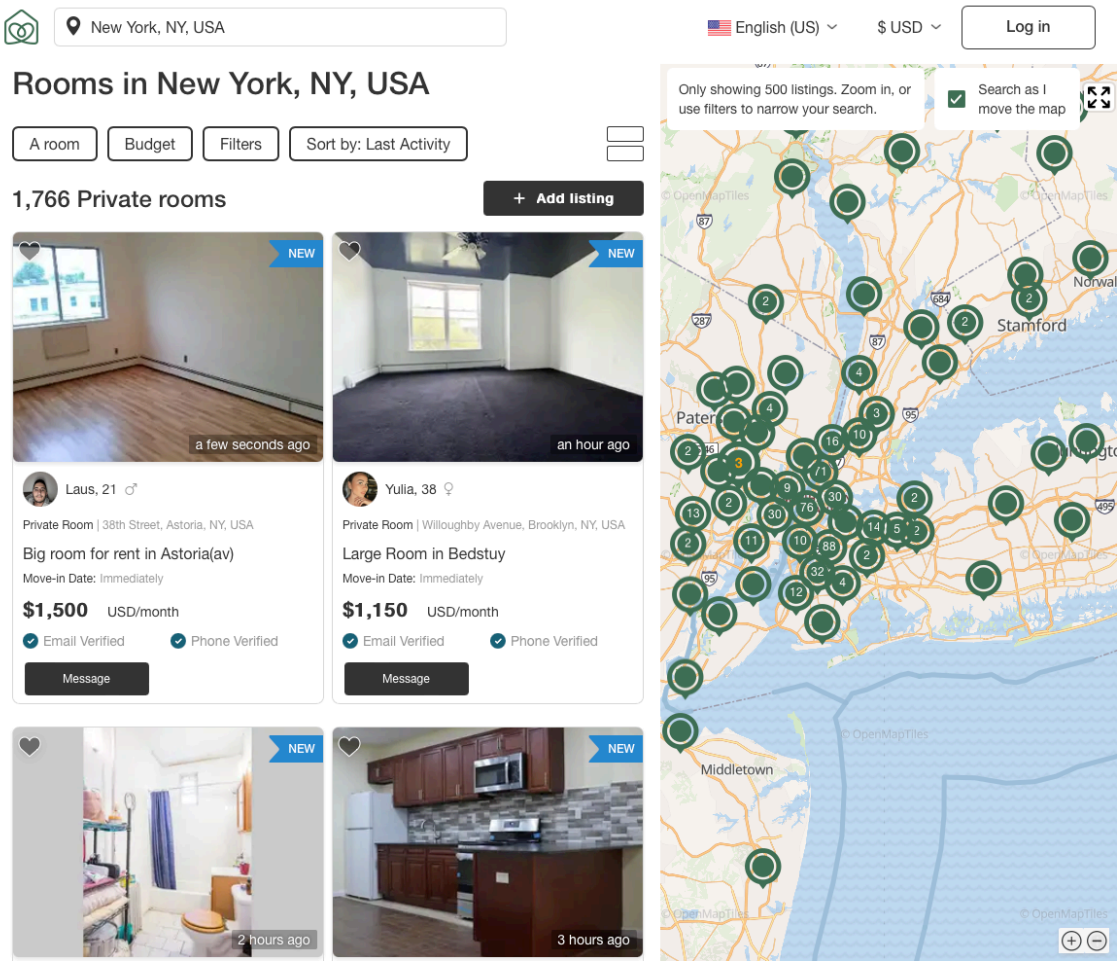
2.1. Pregled sličnih rješenja

Jedna od poznatijih sveobuhvatnih platformi koja nudi rješenja za pronalaženje cimera i stanova je Findroommate [1]. Popularna stranica omogućuje korisnicima stvaranje profila, pregledavanje dostupnih oglasa i izravnu komunikaciju s potencijalnim cimerima. Jedna od glavnih prednosti Findroommate platforme je mogućnost jednostavnog filtriranja oglasa na temelju korisničkih preferencija kao što su lokacija, cijena i karakteristike stana. Ova značajka olakšava pronalaženje idealnog najmoprimca ili stana koji zadovoljava potrebe korisnika. Web stranica također nudi korisne informacije o svakom stanu, uključujući pojedinosti o prostoru, fotografije i druge informacije koje pomažu korisnicima da donesu odluku. Osim toga, Findroommate omogućuje korisnicima pregled recenzija drugih korisnika kako bi stekli uvid u pouzdanost potencijalnih cimera. Naslovna stranica Findroommate-a prikazana je sljedećom slikom 2.1.



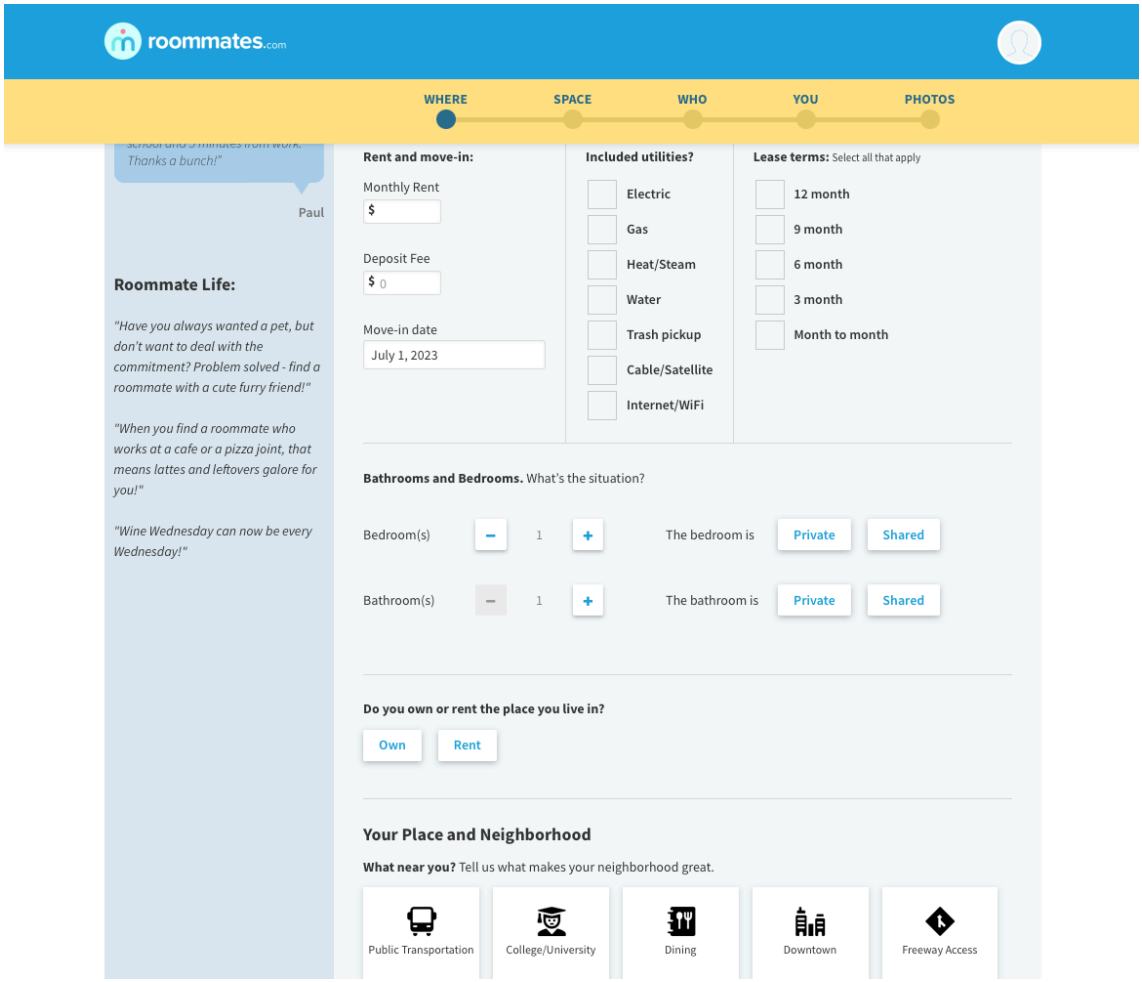
Slika 2.1. Naslovna stranica Findroommate platforme

Svojom posebnošću i korisnim značajkama posebno za studente ističe se Roomster.com [2], online platforma za pronalazak stanova i cimera. Jedna od glavnih značajki Roomster-a je razlikovanje između pojedinačnih soba i cijelih apartmana. To znači da korisnici mogu pronaći ne samo cijele stanove za najam, već i pojedinačne sobe u postojećim stanovima. Ova opcija je posebno korisna studentima ili potencijalnim cimerima koji traže sobu u zajedničkom stanu. Još jedna vrijedna značajka Roomster.com je prikaz karte, koji korisnicima omogućuje vizualno istraživanje dostupnih stanova i njihovih lokacija. To korisnicima olakšava odabir stana na povoljnoj lokaciji prema njihovim potrebama i preferencijama. Također, Roomster korisnicima nudi mogućnost izravne komunikacije unutar platforme. Izlistanje soba određenog grada stranice Roomster.com prikazano je sljedećom slikom 2.2.



Slika 2.2. Rooms stranica grada New York-a Roomster platforme

MyRoommates platforma ističe se fokusom na sigurnost korisnika i transparentnost informacija. Za razliku od nekih drugih stranica, MyRoommates zahtijeva da se prijavite i kreirate račun prije nego što možete tražiti potencijalne cimere [3]. Ova značajka, iako se na prvi pogled može činiti nezgodnom, zapravo pomaže u održavanju legitimnosti korisnika i sigurnosti platforme. Još jedna razlika je detaljan sustav filtriranja koji nudi MyRoommates platforma. Korisnici mogu točno odrediti svoje preferencije stanovanja i cimera, što uključuje sve, od fizičkih karakteristika smještaja do osobnih navika potencijalnih cimera. Dodatno, MyRoommates nudi rigorozan sustav pregleda i provjere ocjena koji korisnicima omogućuje da steknu dublje razumijevanje svog iskustva s potencijalnim cimerima. To se može smatrati dodatnom prednošću u odnosu na stranice poput Roomset.com i FindRoommate.dk. Dio registracije korisnika nakon što je odabrao opciju posjedovanja sobe prikazan je slikom 2.3.



Slika 2.3. Account stranica MyRoommates platforme

2.2. Prijedlog rješenja

S obzirom na složenost i specifičnost procesa traženja stana i prostora, potrebno je razviti inovativno rješenje koje će na personaliziran način olakšati ovaj proces, unaprijediti komunikaciju među korisnicima i osigurati pružanje relevantnih informacija:

Stanodavci se suočavaju s različitim izazovima u pronalasku idealnih najmoprimaca. U ovom kontekstu, aplikacija pruža rješenje koje stanovima omogućuje detaljno opisivanje svoje nekretnine, uključujući broj soba, površinu, lokaciju, cijenu i druge bitne značajke. Također, aplikacija će stanodavcima pružiti mogućnost da izrazi svoja očekivanja od potencijalnih cimera, npr. pušač/nepušač, kućni ljubimci itd. Pored toga, stanovi će imati mogućnost izravne komunikacije s potencijalnim najmoprimcima, pružajući im dodatne informacije ili odgovarajuće na njihova pitanja.

S druge strane, cimeri se također mogu naći u neugodnim situacijama prilikom traženja stana ili cimera. Da bi se ovaj proces olakšao, aplikacija pruža cimerima mogućnost stvaranja personaliziranog profila na kojem će navesti svoje zahtjeve i preferencije. Ti kriteriji mogu uključivati iznos dostupan za najam, željeni grad te ostale karakteristike potencijalnog stana. Isto tako, bitno je da cimeri detaljno opiše svoje osobne navike i rutine. To podrazumijeva navođenje informacija o dnevnim i noćnim rutinama, stavu prema čistoći i urednosti, mišljenju o kućnim ljubimcima, kao i socijalnim i zabavnim preferencijama. Na taj način, potencijalni stanovi i cimeri dobivaju jasan uvid u njihove životne navike, što im pomaže da bolje procjene hoće li život s tom osobom biti u skladu s njihovim očekivanjima i životnim stilom. Sveobuhvatan opis osobnih navika svakog cimera ključan je za uspostavljanje jasnih očekivanja i pronalaženje najboljeg mogućeg uklapanja. Osim toga, cimeri će imati pristup svim dostupnim oglasima i mogućnost filtriranja prema svojim preferencijama, kao i mogućnost izravne komunikacije sa stanodavcima.

Za neprijavljene korisnike aplikacija pruža ograničeni uvid u dostupne oglase i profil cimera. Ova grupa korisnika moći će pretraživati oglase za stanove i prostore, ali bez mogućnosti pristupa detaljnim informacijama, poput kontaktnih podataka i poruka. Ova značajka je zamišljena da pruži pregled mogućnosti koje aplikacija nudi neprijavljenim korisnicima, motivirajući ih na registraciju za potpuno korisničko iskustvo.

3. ARHITEKTURA RJEŠENJA I KORIŠTENE TEHNOLOGIJE

U ovom poglavlju opisani su zahtjevi za korištenje aplikacije te je prikazana arhitektura rješenja potrebna za razvoj aplikacije kako bi se uspješno ispunili ti zahtjevi i implementirale potrebne funkcionalnosti.

3.1. Zahtjevi na aplikaciju

Web aplikacija bi trebala pružati različita sučelja ovisno o tome je li korisnik prijavljen ili ne. Ako je korisnik prijavljen, sučelje se prilagođava prema ulozi, bilo cimera ili stanodavca. Za neregistrirane korisnike aplikacije potrebno je omogućiti pretraživanja i filtriranja smještaja, poput soba i cimera.

Sučelje za neregistrirane korisnike također treba omogućiti prijavu i registraciju. Registracija se također razlikuje ovisno o tome žele li iznajmiti sobu ili pronaći cimera i smještaj. Sučelja cimera i stanodavaca bi trebala dozvoliti pristup detaljnim informacijama o drugim korisnicima ili sobama dostupnim za iznajmljivanje. Također, komunikacija u stvarnom vremenu bi trebala biti moguća. Sučelja prijavljenih korisnika bi trebala podržati uređivanje profila.

Prijavljenim korisnicima bi trebala biti pružena funkcionalnost upravljanja najmovima. Sučelje stanodavca razlikuje se od sučelja cimera po tome što omogućuje sveobuhvatno upravljanje sobama, uključujući uređivanje, brisanje i dodavanje soba.

3.2. Prijedlog rješenja arhitekture

U našoj aplikaciji razlikujemo tri vrste korisnika: neregistrirane korisnike, prijavljene korisnike dodijeljene ulozi cimera i prijavljene korisnike dodijeljene ulozi stanodavca. Kao rezultat toga, naša aplikacija nudi tri različita načina korištenja. U nastavku ćemo opisati način rada i strukturu baze podataka za svaku vrstu korisnika.

3.2.1. Arhitektura rješenja iskustva neprijavljenog korisnika

Korisnici će pri prvom pristupu aplikaciji imati mogućnost traženja smještaja ili cimera. Korisnici će moći koristiti različite filtre prilikom traženja soba, uključujući vrstu smještaja, maksimalnu mjesečnu cijenu, lokaciju smještaja i drugo. Isto tako, korisnici će moći odrediti željeno mjesto stanovanja, dob idealnog cimera, žele li živjeti u studentskom domu i druge kriterije prilikom traženja cimera. Neregistrirani korisnici moći će vidjeti samo podatke potrebne za filtriranje; privatni podaci o sobama i cimerima neće im biti dostupni. Zbog toga će neregistrirane osobe moći vidjeti oglase za sobe i cimere, ali ne profile.

Mogućnosti prijave ili registracije na web stranici bit će dostupne neprijavljenim korisnicima. Korisnici će se morati prijaviti unosom korisničkog imena i lozinke koji moraju ispunjavati određene kriterije, uključujući kombinaciju velikih i malih slova, određenu duljinu i sadržavati brojeve. Korisnička imena moraju biti jedinstvena.

Postupak registracije za neregistrirane korisnike sastojat će se od najmanje četiri faze. Kako bi se registrirali kao stanodavac koji iznajmljuje sobu, neregistrirani korisnici prvo će trebati unijeti podatke za autentifikaciju i autorizaciju, osobne podatke poput imena i prezimena, kriterija za cimera i, po želji, profilnu sliku.

Ukoliko se korisnik registrira s ciljem učinkovitijeg pronalaženja cimera ili sobe, proces registracije uključuje dodatni korak, a to je unos kriterija za sobu.

Baza podataka neće zahtijevati pohranu podataka koji se odnose na neregistrirane korisnike.

3.2.2. Arhitektura rješenja iskustva stanodavca

Podaci se upisuju u bazu nakon što se neregistrirani korisnik registrira kao stanodavac, a korisniku postaju dostupne dodatne opcije kroz sučelja prilagođena njegovoj ulozi. U tablici *Korisnik*, koja također sadrži sve podatke dane prilikom registracije i jedinstveni identifikator (ID), zadržavaju se podaci koji se odnose na autentifikaciju i autorizaciju nakon registracije. Potrebna je dodatna tablica pod nazivom *Stanodavac* koja ima odnos 1:1 s entitetom *Korisnik* za bilježenje podataka o stanodavcima. Entitet *Kriterij cimera* možemo izravno povezati s entitetom *Korisnik* jer neregistrirani korisnici moraju unijeti kriterije cimera ako se žele registrirati kao cimeri.

Osim toga, entitet *Korisnik* i entitet *Slika profila* trebali bi imati odnos 1:1, dok s entitetom *Uloga* treba imati odnos više-prema-više, dopuštajući mogućnost dodjele više uloga jednom korisniku.

Korisnici koji su prijavljeni i imaju ulogu stanodavca moći će uređivati svoje profile, što uključuje i promjenu podataka u već navedenim tablicama. Nadalje, korisnici će imati mogućnost kreiranja oglasa za sobe, što rezultira odnosom 1:M između stanodavaca i tablice *Soba* i odnosom 1:M između entiteta *Soba* i entiteta *Slika sobe*. Činjenica da stanodavci mogu odobriti, odbiti i izbrisati zahtjeve za najam sugerira postojanje entiteta *Zahtjev* sa stranim ključevima za stanodavca, cimera i sobu. Kada se zahtjev odobri, baza podataka će stvoriti entitet *Najam* s istim stranim ključevima.

Entitet *Poruka* će biti povezan s entitetom *Korisnik* jer će stanodavci moći slati poruke drugim stanodavcima ili cimerima. Objekt *Korisnik* imat će dvije 1:M veze s entitetom *Poruka*, jednu za poslane poruke i jednu za primljene poruke.

3.2.3. Arhitektura rješenja iskustva stanara

Entitet *Cimer*, koji je u odnosu 1:1 s entitetom *Korisnik*, pohranit će podatke o cimerima nakon registracije. Mogućnosti registriranog korisnika koji ima ulogu stanodavca i cimera su skoro iste. Međutim, cimer neće moći kreirati sobe, ali će i dalje moći kreirati ili podnijeti zahtjev za najam stanodavcu. Entitet *Cimer* također ima odnos 1:1 s entitetom *Kriterij za sobu*. Budući da će cimeri moći izravno tražiti sobe na temelju kriterija sobe, a ostale cimere na temelju svojih cimer kriterije, to će omogućiti učinkovitije pronalaženje cimera.

3.3. Pregled korištenih tehnologija

3.3.1. ASP.NET Framework

Microsoft je stvorio snažan i prilagodljiv ASP.NET [4] okvir kako bi omogućio programerima stvaranje dinamičnih internet usluga i aplikacija. Od svojeg nastanka, on je bio temelj internetskog razvoja, pružajući programerima čvrst temelj na kojem mogu graditi bogate značajke, skalabilne i sigurne internet aplikacije. C# i *VB.NET* su samo dvije od mnogih programskih jezika koje *ASP.NET* podržava, pružajući programerima slobodu da koriste onaj s kojim su najviše upoznati. Jedna od njegovih važnih komponenti je *ASP.NET Web Forms*, koji pojednostavljuje stvaranje korisničkih sučelja za web aplikacije pružajući sučelje za povlačenje i spuštanje. Dodatno, *ASP.NET MVC* (engl. Model-View-Controller) nudi organiziraniji način web razvoja, potičući razdvajanje odgovornosti i poboljšavajući održivost programa. Napravljen je da može funkcionirati na svim operacijskim sustavima. Nudi mogućnosti za izradu dinamičke stranice u HTML-u, CSS i JavaScript-u. *ASP.NET* korišten u izradi aplikacije je *Web API.* Taj okvir pojednostavljuje rad sa *HTTP* uslugom.

Najnovija verzija okvira, *ASP.NET* Core, je kod otvorenog tipa i podržava više platformi, što ga čini omiljenim među suvremenim web developerima. Također pruža izvanrednu podršku za razvoj *RESTful* API-ja, što omogućava izgradnju pouzdanih pozadinskih usluga (engl. backend). Dodatno, popularna razvojna okolina Visual Studio integrirana s *ASP.NET*-om ubrzava procese razvoja, ispravljanja pogrešaka i implementacije, povećavajući produktivnost programera. Sigurnost aplikacija je dodatno ojačana njegovim ugrađenim sigurnosnim mehanizmima, kao što su autentifikacija i autorizacija, koji pomažu zaštititi od uobičajenih propusta. Za programere koji žele stvoriti skalabilne, visokoučinkovite web aplikacije i usluge, *ASP.NE*T je neophodan alat. Kod implementacije sustava ASP.NET omogućuje prijavu korisnika pomoću ASP.NET *Identity framework* te automatski određivanje korisnikove uloge na Internet stranici ili aplikaciji pomoću ASP.NET *Role Management*.

3.3.2. Entity Framework

Microsoft je stvorio Entity Framework [5], okvir za mapiranje objekata na relacijske baze podataka (engl. ORM) za .NET aplikacije. To je moćan alat za rad s relacijskim bazama podataka na način koji je više usmjeren prema objektima. Snažno tipizirani .NET objekti koje Entity Framework omogućuje developerima koristiti za interakciju s bazama podataka čine pristup i manipulaciju bazom podataka jednostavnijim i manje sklono greškama.

Slijede neki od ključnih pojmova i elemenata Entity Framework-a.

Entiteti: U Entity Framework-u, entitet je klasa koja odgovara tablici u bazi podataka ili modelu podataka. Svaka instanca entiteta predstavlja jedan redak u tablici.

DbContext: Ključna komponenta Entity Framework-a je DbContext. On predstavlja sesiju s bazom podataka koja omogućuje upite nad podacima i manipulaciju njima. Također nadzire praćenje promjena nad entitetima. Entity Framework podržava LINQ (engl. Language Integrated Query), što omogućuje korisnicima upite nad bazama podataka koristeći dobro poznatu sintaksu C#. Da biste dobili i mijenjali podatke, možete pisati LINQ upite prema vašim entitetima.

Code-First: Entity Framework omogućuje kod prvo (engl. code-first) pristup, pri kojem se vaš model podataka definira u C# klasama, a shema baze podataka automatski se izgradi iz tih klasa. Budući da omogućuje developerima rad s snažno tipiziranim klasama bez brige o strukturi baze podataka, ovaj pristup je vrlo popularan.

Alternativno: Možete početi s postojećom bazom podataka i stvoriti C# klase koje opisuju shemu baze podataka koristeći baza prvo (engl. database-first) pristup. Te klase i DbContext mogu vam biti stvoreni od strane Entity Framework-a.

Model-First: Ova metoda stvara kako C# klase tako i shemu baze podataka nakon što je vaš model podataka definiran pomoću Entity Framework-ovog Entity Data Model alata. Migracije baze podataka, koje Entity Framework omogućuje, omogućuju vam mijenjanje strukture baze podataka tijekom vremena bez gubitka podataka. Na temelju promjena koje napravite na svom modelu podataka, migracije automatski stvaraju SQL skripte za ažuriranje sheme baze podataka. Entity Framework podržava i tzv. napredno učitavanje (engl. eager loading) i odgođeno učitavanje (engl. lazy loading), gdje se povezani podaci učitavaju zajedno s primarnim entitetom kako bi se uštedjeli upiti prema bazi podataka, te se relevantni podaci iz baze podataka učitavaju samo kada se pristupa tim podacima.

Davatelji baza podataka: Entity Framework podržava mnoge davatelje baza podataka, kao što su SQL Server, MySQL, PostgreSQL i SQLite, samo su neki od mnogih koje podržava.

3.3.3. REST Api

REST (engl. Represetation State Transfer) api [6], arhitektura je koja je po prvi put predložena od strane Dr. Roya Fieldinga 2000 godine. Danas je ona jedan od glavnih struktura za dizajn web aplikacija. Izgrađen je na temelju nekoliko ključnih principa. Prvi je bez stanje (engl. Statelessness). On nalaže da svaki klijentov zahtjev koji se šalje poslužitelju mora sadržavati sve informacije koje su potrebne da bi poslužitelj mogao obraditi i razumjeti zahtjev. Drugi su resursi (engl. *Resources*). Kod REST-a sve stranice, dokumenti, podatkovni znakovi, slike smatraju se oblikom resursa. Treće su HTTP metode GET, SET, PUT, DELETE i SET. Pomoću tih metoda, nad resursima, izvodi se stvaranje, ažuriranje, brisanje i čitanje kako bi se njima moglo manipulirati. Četvrta su uniformna sučelja. To se odnosi na to da bi resursi sa istim svojstvima trebali imati zajedničke URL-ove i sheme kako i se lakše održavao i koristio. Peta je reprezentacija u kontekstu toga da resursi mogu biti u različitim oblicima kao što su: JSON, XML ili HTML. Šesti je bez stanje što zapravo podrazumijeva to da je svaki ciklus zahtjeva odgovora treba biti neovisan o ostalima. Sedmi i posljednji je slojevitost. Slojevi podrazumijevaju uvođenje posrednih poslužitelja poput posrednika(engl. Proxy). Glavna svrha je mogućnost fleksibilnosti.

Operativni mehanizam REST-a sastoji se od identifikacije resursa, HTTP metoda, reprezentacija resursa i od bez stanja. Identifikacija resursa odrađuje se preko URI-a (engl. Uniform Resource Identifier). HTTP metode omogućuju klijentu direktnu interakciju sa resursima putem gore navedenih metoda. Reprezentacija resursa može biti u različitim oblicima dok klijent bira željenu reprezentaciju. Ta reprezentacija i HTTP statusni kodovi vraćaju se od strane poslužitelja. Svaki zahtjev klijenta je neovisan o ostalima, dakle poslužitelj ga ne sprema, a to nam omogućava *statelessness.*

3.3.4. Angular

Google je razvio i održava Angular [7777], moćan otvoren kod (engl. open-source) okvir za stvaranje web aplikacija. Zahvaljujući svojim mnogim prednostima, ovaj okvir postao je iznimno popularan među developerima. Glavna karakteristika Angular-a je jednostavnost dinamički generiranih web aplikacija (engl. SPA) i sigurne razmjene povjerljivih podataka. Jedna od stvari koja je potrebna za to je integracija TypeScripta, statički tipizirane varijante JavaScripta koja prirodno promiče izlazni kod i čitljivost. Angular promovira organiziran i modularan pristup razvoju aplikacija temeljen na konceptima kao što su komponente, predlošci, povezivanje

podataka i ubrizgavanje ovisnosti. To omogućuje programerima da stvaraju aplikacije koje se lako održavaju i skaliraju. Kako bi se to postiglo, Angular ima glatko korisničko sučelje koje poboljšava navigaciju kroz aplikaciju. Također pruža velike mogućnosti testiranja, što je ključno za identifikaciju visokokvalitetnih aplikacija. Osim toga, Angular se ističe po snažnom sustavu za usmjeravanje koji olakšava navigaciju kroz aplikaciju. Također nudi sveobuhvatne mogućnosti testiranja, koje su ključne za provjeru kvalitete aplikacija. Dodatno, ima ugrađenu podršku za lokalizaciju i internacionalizaciju, što je ključno za stvaranje aplikacija s globalnom usmjerenosti. Snažna podrška zajednice i širok spektar modula i alatki koje olakšavaju razvoj aplikacija su dvije od osnovnih prednosti Angular-a. To ga čini savršenim izborom za stvaranje složenih web aplikacija na razini poduzeća koje trebaju biti skalabilne, održive i brze. To je također razlog zašto je izabran za stvaranje aplikacije u ovom radu.

4. PROGRAMSKO RJEŠENJE APLIKACIJE

Nakon analize zahtjeva, definiranja arhitektura rješenja i opisa korištenih tehnologija, u ovom poglavlju prikazana su programska rješenja to jest implementirane potrebne funkcionalnosti.

4.1. Postavljanje razvojne okoline

Postavljanje razvojnog okruženja započinje stvaranjem .NET rješenja izdavanjem sljedeće naredbe *dotnet new sln*. Nakon toga, iniciramo ASP.NET Core Web API aplikaciju koristeći naredbu *dotnet new webapi -n API*. U datoteci *Program.cs*, nužno je uvesti osnovne servise u kontejner za ubrizgavanje ovisnosti, mehanizam koji će se koristiti i distribuirati po cijeloj aplikaciji. Uvode se međusloj (engl. middleware) komponente, ključne za obradu dolaznih zahtjeva na putu do odredišta i odgovora na njihovom povratnom putovanju prema klijentima. Od posebnog značaja je CORS (engl. Cross-Origin Resource Sharing), koji odobrava zahtjeve između različitih platformi. Na slici 4.1 se može vidjeti da se prihvaćaju zahtjevi s izvora "https://localhost:4200", što bi bio tipičan izvor za razvojnu Angular aplikaciju na lokalnom računalu.

```
app.UseCors(builder => builder.AllowAnyHeader()
                              .AllowAnyMethod()
                              .AllowCredentials()
                              .WithOrigins("https://localhost:4200"));
```

Slika 4.1 CORS međusloj

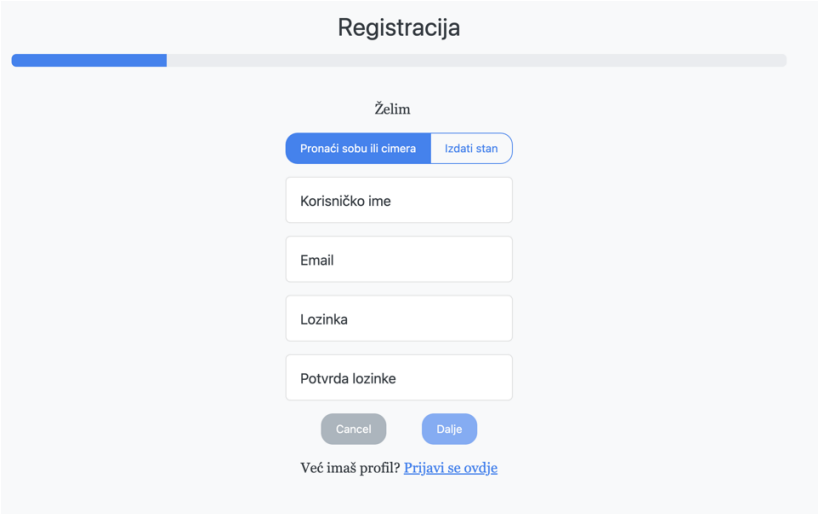
Prije nego što krenemo s izradom Angular aplikacije, nužno je globalno instalirati Angular CLI (engl. Command Line Interface). Taj zadatak se može uspješno izvršiti sljedećom naredbom *npm install -g @angular/cli*. Nakon toga, Angular CLI nam omogućava generiranje Angular aplikacije, koju možemo nazvati *client* prema vlastitim željama naredbom *ng new client*.

Što se tiče upravljanja paketima, važno je napomenuti da se vanjski paketi za .NET Core aplikaciju mogu dodavati putem NuGet Package Managera. Ova aplikacija je proširena s Entity Frameworkom, Identityjem i SignalR-om.

S druge strane, kad je riječ o Angular aplikaciji, običaj je upravljati vanjskim paketima putem npm-a (eng. Node Package Manager). Ovi paketi se mogu jednostavno instalirati izdavanjem naredbe *npm install*.

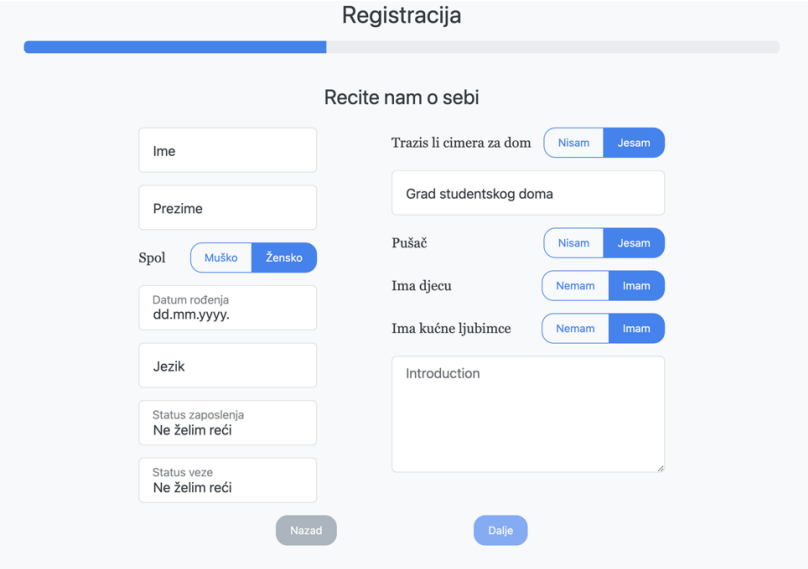
4.2. Kreiranje i personalizacija profila cimera i stanodavaca

Kreiranje i personalizacija profila kako za stanodavce tako i za cimere obuhvaćena su unutar procesa registracije korisnika. Na slici 4.2 se može vidjeti početni dio obrasca za registraciju korisnika, koji korisniku nudi mogućnost da se registrira kao cimer ili stanodavac.



Slika 4.2 Prvi dio registracije korisnika

Na slici 4.3 je prikazan obrazac koji je dostupan cimerima za unos osobnih podataka, dok se stanodavcima pruža poseban obrazac za unos njihovih informacija. Nakon toga, cimeri su obavezni ispuniti obrazac za kriterije u odabiru stanodavca te dodatni obrazac za specificiranje željenih karakteristika prostora. Nakon što završe s ovim koracima, cimeri mogu nastaviti s procesom registracije. S druge strane, stanodavcima nije ponuđen obrazac za unos kriterija za odabir sobe tijekom njihove registracije.



Slika 4.3 Dio registracije u kojem se unose osobni podaci cimera

Kako bi omogućili registraciju korisnika, u kontroleru *AccountController* nalazi se metoda za registraciju koja prima objekt s vrijednostima unesenim u obrascima za registraciju. Na slici je prikazana metoda kontrolera u kojoj se prvo provjerava postoji li već korisnik s istim podacima. Ako korisnik već postoji, vraća se odgovor s statusom *400* koji označava *Loš zahtjev*.

```
public async Task<ActionResult<UserDTO>> SignUp(SignUpDTO signUpDTO)
{
    if(await UserExists(signUpDTO.Username))
        return BadRequest("User with this username already exists.");

    var user = this.mapper.Map<AppUser>(signUpDTO);
    user.UserName = user.UserName.ToLower();
    var role = "Roommate";

    if(signUpDTO.UserType.ToLower() == "renter")
    {
        role = "Renter";
        var renter = new Renter();
        user.Renter = renter;
    }
    else
    {
        var roomCriteria = this.mapper.Map<RoomCriteria>(signUpDTO.RoomCriteria);
        user.Roommate.RoomCriteria = roomCriteria;
    }

    var result = await this.userManager.CreateAsync(user, signUpDTO.Password);
    if (!result.Succeeded) return BadRequest(result.Errors);

    var roleResult = await this.userManager.AddToRoleAsync(user, role);
    if (!roleResult.Succeeded) return BadRequest(result.Errors);
```

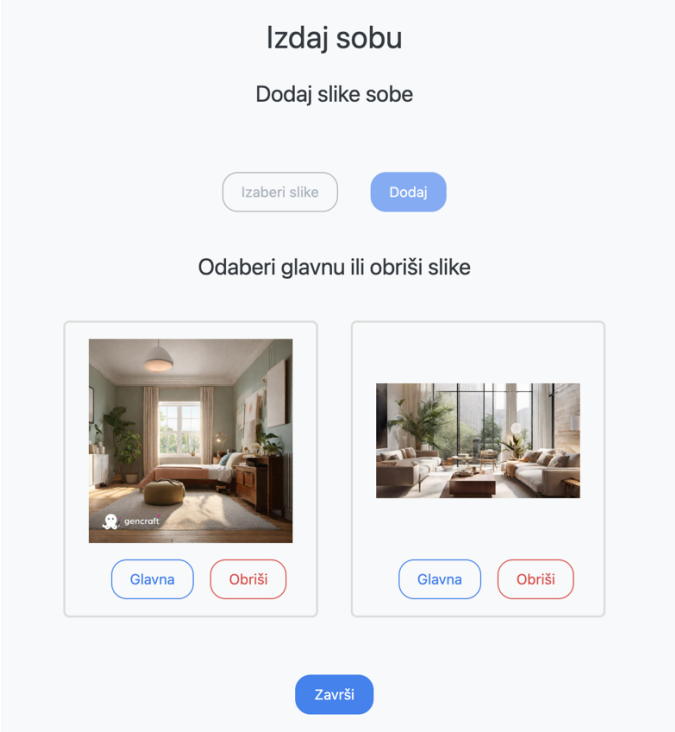
Slika 4.4 Dio metode za registriranje korisnika

Nakon ove provjere, primljeni objekt se mapira u objekt *korisnik*. Ovisno o vrijednosti svojstva tipa korisnika, sustav dodjeljuje korisniku ili objekt *stanodavac* ili objekt *cimer*. U ovom kontekstu, objekt *cimer* nije izričito naveden jer je uključen unutar objekta poslanog od strane klijenta. U tom slučaju, objekt će biti mapiran i povezan s objektom *korisnik* tijekom mapiranja *korisnik* objekta. Potrebni objekti se povezuju s objektom *korisnik* uspostavljajući odnos između entiteta i zatim se pohranjuju u bazu podataka, dok se korisnik povezuje s odgovarajućom ulogom.

4.3. Kreiranje i upravljanje oglasima

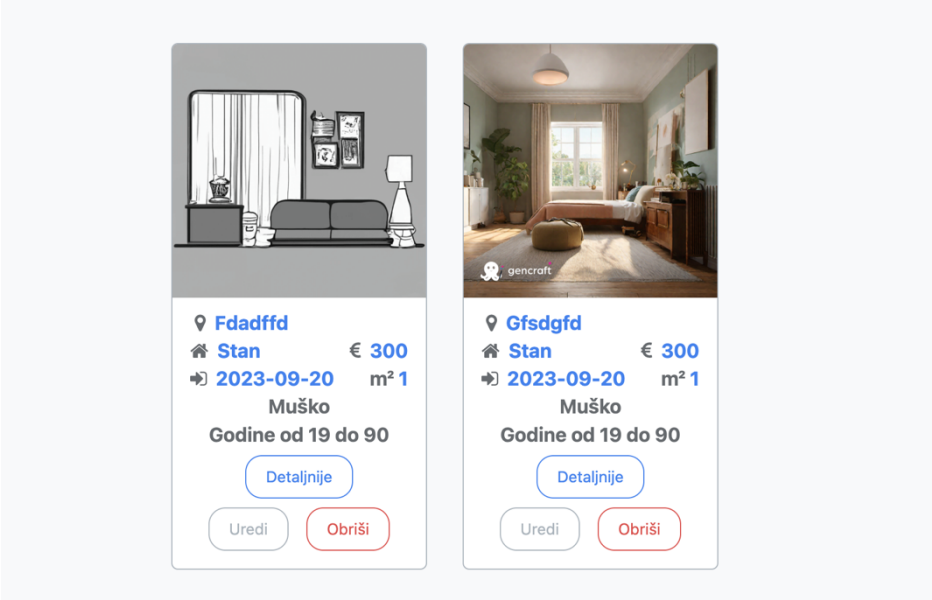
Korisnici s ulogom *stanodavac* imaju mogućnost stvaranja oglasa za iznajmljivanje soba. Proces stvaranja takvih oglasa sastoji se od tri koraka. U prvom koraku korisnici unose informacije povezane s oglasom. U drugom koraku biraju slike koje će priložiti uz oglas u bazi podataka.

Konačno, u trećem koraku, nakon što su slike dodane, korisnici mogu odabrati koja će slika biti prikazana uz oglas na listi oglasa. U trećem koraku, korisnici također imaju mogućnost brisanja prethodno dodanih slika i dodavanja novih, jer im je dostupan drugi dio obrasca prije nego što predaju oglas. Prikazana slika 4.5 ilustrira drugi i treći korak.



Slika 4.5 Dodavenje slika u oglas

Korisnici s ulogom *stanodavac* imaju sposobnost uređivanja i brisanja oglasa koje su stvorili. Što je vidljivo nas slici 4.6



Slika 4.6 Upravljanje kreiranim oglasima

Unutar kontrolera *Room*Controller nalazi se metoda, vidljiva na slici 4.7, za stvaranje oglasa za iznajmljivanje soba. U ovoj metodi preslikavamo primljeni objekt u objekt *soba*, dohvaćamo stanodavca i povezujemo ga sa sobom prije nego što je spremimo u bazu podataka. Entity Framework automatski uspostavlja potrebne veze tijekom operacije spremanja u bazu podataka.

```
[Authorize(Roles = "Renter")]
[HttpPost]
0 references
public async Task<ActionResult<int>> AddRoom(CreateRoomDTO createRoomDTO)
{
    var room = mapper.Map<Room>(createRoomDTO);

    var renter = await unitOfWork.RenterRepository.GetRenterOnlyByUsernameAsync(User.GetUsername());

    if(renter == null)
        return NotFound();

    room.Renter = renter;

    await unitOfWork.RoomRepository.AddRoomAsync(room);

    if(await unitOfWork.Complete()){
        var createdRoom =  await unitOfWork.RoomRepository.GetLastCreatedRoomByRenterId<Room>(renter.Id);
        return CreatedAtRoute("GetRoom", new { id= createdRoom.Id}, createdRoom.Id);
    }

    return BadRequest("Failed to add room");
}
```

Slika 4.7 Metoda za kreiranje sobe

Metoda odgovorna za spremanje slika je prikazana na slici 4.8. Slika koju je poslao klijent u obliku datoteke šalje se vanjskoj usluzi za pohranu slika, koja vraća URL na kojoj je slika dostupna. Iz tog razloga pohranjujemo URL slike, koji će klijent koristiti za prikazivanje slike.

```
public async Task<ActionResult<PhotoDTO>> AddRoomPhoto(int id, [FromForm] IFormFile file)
{
    var room = await unitOfWork.RoomRepository.GetRoomOnlyByIdAsync(id);

    if (room == null) return NotFound();

    var result = await photoService.UploadPhotoAsync(file);

    if(result.Error != null)
    {
        return BadRequest("Failed to upload profile picture");
    }

    var photo = new RoomPhoto
    {
        Url = result.SecureUrl.AbsoluteUri,
        PublicId = result.PublicId,
        Room = room
    };

    await unitOfWork.RoomPhotoRepository.AddRoomPhotoAsync(photo);
```

Slika 4.8 Metoda za dodavanje slike sobe

4.4. Filtriranje i pretraga oglasa i cimera

Filtriranje i pretraživanje korisnika su optimizirani putem slanja objekta metodi na serveru, koja koristi vrijednosti tog objekta za izvođenje upita nad bazom podataka. Na prikazanoj slici 4.9 možemo vidjeti obrazac za filtriranje potencijalnih cimera.

Slika 4.9 Obrazac za filtriranje cimera

Za korisnike koji imaju ulogu *cimera*, na serveru je dostupna metoda koja prosljeđuje korisnikov kriterij za cimera za izvođenje upita nad bazom. Prikazano slikom 4.10.

```
[Authorize(Roles = "Roommate")]
[HttpGet("matches")]
public async Task<ActionResult<PagedList<RoommateDTO>>>
    0 references
    GetRoommatesMatches([FromQuery]RoommateParams roommateParams)
{
    roommateParams.CurrentUsername = User.GetUsername();

    var user = await unitOfWork.UserRepository.GetUserByUsernameAsync(User.GetUsername());
    var roomCriteria = await unitOfWork.RoommateRepository
        .GetRoomCriteriaByUsername<RoomCriteriaDTO>(User.GetUsername());

    if(user == null || user.RoommatePreference == null || roomCriteria == null)
        return NotFound();

    mapper.Map(user.RoommatePreference, roommateParams);
    roommateParams.City = roomCriteria.City;

    var roommates = await unitOfWork.RoommateRepository.GetRoommatesAsync(roommateParams);

    Response.AddPaginationHeader(
        roommates.CurrentPage,
        roommates.PageSize,
        roommates.TotalCount,
        roommates.TotalPages
    );

    return Ok(roommates);
}
```

Slika 4.10 Metoda za dohvaćanje cimera s kriterijem za cimere

Na slici 4.11 je vidljiv upit nad bazom za dohvaćanje i filtriranje potencijalnih cimera.

```
var query =  context.Roommates.Include(r => r.RoomCriteria)
                              .Include(r => r.User)
                              .ThenInclude(u => u.RoommatePreference)
                              .Include(r => r.User.Photo)
                              .AsQueryable();

if(roommateParams.CurrentUsername != null)
    query = query.Where( r => r.User.UserName != roommateParams.CurrentUsername);

if(roommateParams.City != null && roommateParams.City !="")
    query = query.Where( r =>
        r.RoomCriteria.City.ToLower().Contains(roommateParams.City.ToLower()));

if(roommateParams.Gender != null)
    query = query.Where(r => r.User.Gender == roommateParams.Gender.ToLower());

query = query.Where(r => r.User.DateOfBirth >= roommateParams.MinDateOfBirth
                && r.User.DateOfBirth <= roommateParams.MaxDateOfBirth);

if(roommateParams.IsSmoker != null && roommateParams.IsSmoker!= "no preference")
    query = query.Where(r => r.IsSmoker == bool.Parse(roommateParams.IsSmoker));

if(roommateParams.HasChildren != null && roommateParams.HasChildren!= "no preference")
    query = query.Where(r => r.HasChildren == bool.Parse(roommateParams.HasChildren));

if(roommateParams.HasPets != null && roommateParams.HasPets != "no preference") {
    bool sale = bool.Parse(roommateParams.HasPets);
    query = query.Where(r => r.HasPets == bool.Parse(roommateParams.HasPets));
}
```

Slika 4.11 Upit za dohvaćanje cimera

4.5. Komunikacija između korisnika

Kada se korisnik prijavi na aplikaciju i pristupi funkcionalnosti razmjene poruka, uspostavlja se veza s *MessageHub*-om. U tom trenutku, korisnik se smatra aktivnim. Veze se stvaraju kada korisnici otvori poruke na klijentu , i za svaki razgovor generira se jedinstveno ime grupe. To znači da kada drugi korisnik uđe u isti razgovor, dijele isto ime grupe. Kreiranje grupe prilikom otvaranjem konekcije vidljivo je slikom 4.12

```
0 references
public override async Task OnConnectedAsync()
{
    var httpContext = Context.GetHttpContext();
    var otherUser = httpContext.Request.Query["user"];
    var groupName = GetGroupName(Context.User.GetUsername(), otherUser);
    await Groups.AddToGroupAsync(Context.ConnectionId, groupName);
    var group = await AddToGroup(groupName);
```

Slika 4.12 *OnConnected* metoda

Kada korisnik želi poslati poruku, ona se sastavlja i šalje određenoj grupi koja se sastoji od pošiljatelja i primatelja kao što je prikazano slikom 4.13. *SignalR* pruža metodu za slanje poruka grupi, a informacije o grupama pohranjuju se u bazi podataka.

```
var groupName = GetGroupName(sender.UserName, recipient.UserName);

var group = await uow.MessageRepository.GetMessageGroup(groupName);

if (group.Connections.Any(x => x.Username == recipient.UserName))
{
    message.DateRead = DateTime.UtcNow;
}

uow.MessageRepository.AddMessage(message);

if (await uow.Complete())
{
    await Clients.Group(groupName).SendAsync("NewMessage", mapper.Map<MessageDTO>(message));
```

Slika 4.13 SendMessage metoda

Da bi korisnici na strani klijenta bili obaviješteni, moraju se pretplatiti na događaj nazvanog *NewMessage*. Taj događaj osigurava da korisnici dobivaju ažuriranja u stvarnom vremenu kada stignu nove poruke. Slično tome, popis aktivnih korisnika također se prikazuje na temelju veza uspostavljenih tijekom prijave. Promjene u popisu aktivnih korisnika obavještavaju se putem određenog događaja. Na taj način, kada novi korisnik pristupi ili napusti, svi povezani klijenti odmah su obaviješteni. Na slici 4.14 ažuriramo poruke razgovora unutar *MessageServica* na klijentskoj strani.

```
this.hubConntection.on('NewMessage', message => {
  this.messageThread$.pipe(take(1)).subscribe({
    next: messages => {
      this.messageThreadSource.next([...messages, message]);
    }
  })
})
```

Slika 4.14 Pretplata na *NewMessage* događaj

4.6. Arhiviranje najmova

Arhiviranje zahtjeva za najam predstavlja složen proces koji započinje slanjem zahtjeva od strane korisnika s ulogom *cimer* pritiskom na gumb na stranici oglasa za sobu. Svaki zahtjev nosi jedinstveni ključ s identifikatorima cimera, stanodavca i sobe, čime postaje vidljiv vlasniku. Vlasnik ima dvije opcije na raspolaganju: odbiti ili prihvatiti zahtjev. U slučaju prihvaćanja,

zahtjev se označava statusom *prihvaćen*, a vlasnik ima mogućnost brisanja zahtjeva prema vlastitim preferencijama. Na slici 4.15 prikazan je izgleda primljenih zahtjeva.

Stanar	Vlasnik	Id sobe	
@mate1	@renter	1	Obriši
@mate1	@renter	69	Obriši
@mate1	@renter	69	Odbij Prihvati

Slika 4.15 Lista zahtjeva

Paralelno s tim, stvara se novi unos o najmu koji se označava kao aktivan, kako je prikazano na priloženoj slici 4.16.

```
public async Task<ActionResult> CreateRental(RequestDTO requestDTO) {
    var request = await unitOfWork.RequestRepository.GetRequest(requestDTO.Id);

    var renter = await unitOfWork.RenterRepository
        .GetRenterOnlyByUsernameAsync(requestDTO.RenterUsername);

    var roommate = await unitOfWork.RoommateRepository
        .GetRoommateOnlyByUsernameAsync(requestDTO.RoommateUsername);

    var room = await unitOfWork.RoomRepository.GetRoomOnlyByIdAsync(request.RoomId);

    if(renter == null || roommate == null || room == null || request == null)
        return NotFound();

    var rental = new ArchivedRental {
        Room = room,
        Renter = renter,
        Roommate = roommate
    };

    request.Status =  "accepted";

    room.Roommate = roommate;

    await unitOfWork.RentalRepository.AddRental(rental);

    if(await unitOfWork.Complete()) {
        return NoContent();
    }
```

Slika 4.16 Metoda za kreiranje najma

Korisnicima s ulogom *cimer* i *stanodavac* koji su povezani putem ove najma omogućuje se i prekid najma. U takvim slučajevima, status najma mijenja se u *neaktivan*, ali se i dalje čuva u bazi podataka.

Ovaj postupak osigurava adekvatno upravljanje zahtjevima i ugovorima o najmu unutar sustava, nudeći fleksibilnost kako cimerima tako i stanodavcima za kontrolu svojih najamnina.

5. ZAKLJUČAK

U završnom radu razvijena je aplikacija s glavnim ciljem ubrzanja procesa pronalaženja stanova i cimera, kao i olakšavanja procesa iznajmljivanja stanova. Glavni cilj ove aplikacije bio je poboljšati korisničko iskustvo kroz funkcionalnost, brzinu i personalizaciju. Glavni zaključci i doprinosi ovog truda sažeti su u nastavku:

Naša istraživanja započela su temeljitim istraživanjem konkurentske ponude na tržištu, uključujući Findroommate, Roomster i MyRoommates. Ovo istraživanje bilo je od velike pomoći u prepoznavanju prednosti i nedostataka tih platformi, što nam je na kraju pomoglo da razvijemo naše rješenje s naglaskom na ispravljanje nedostataka uočenih u tim platformama.

Arhitektura aplikacije pažljivo je planirana, koristeći Angular za klijentsku stranu i ASP.NET Framework za serversku stranu. Integracija Entity Frameworka znatno je ubrzala razvojni proces eliminirajući potrebu za ručnim pisanjem SQL koda.

Naša aplikacija nudi širok niz značajki, uključujući mogućnost stvaranja i prilagodbe korisničkih računa kako za stanodavce tako i za stanare, jednostavno filtriranje oglasa, učinkovite alate za komunikaciju među korisnicima te sustav za organiziranje ugovora o najmu.

Omogućavanjem anonimnim korisnicima da pretražuju stanove i cimere, proširili smo pristup aplikaciji, povećavajući tako njezin potencijalni korisnički krug.

Važno je napomenuti da se aplikacija trenutno nalazi u ranoj fazi razvoja. Trenutno se koristi SQLite za bazu podataka, no postoji mnogo prostora za prijelaz na Entity Framework Server kako bi se poboljšala učinkovitost i optimizacija upita nad velikim skupovima podataka.

Zaključno, ovaj završni rad predstavlja temeljitu analizu, dizajn i razvoj aplikacije za web koja ima za cilj pojednostaviti proces pronalaženja stanova i cimera te olakšati iznajmljivanje stanova. Implementacija brojnih korisnih značajki ostavlja dovoljno prostora za buduća poboljšanja u brzini i proširenju funkcionalnosti. Budući napredak i istraživanje u ovom području imat će čvrstu osnovu zahvaljujući doprinosima ovog autora.

LITERATURA

[1] Findroommate.dk, https://www.findroommate.dk/ (stranica posjećena 29.6.2023.)

[2] Roomster.com, https://www.roomster.com/ (stranica posjećena 29.6.2023)

[3] roommates.com, https://www.roommates.com/ (stranica posjećena 29.6.2023.)

[4] ASP.NET overview, https://learn.microsoft.com/en-us/aspnet/overview (stranica posjećena 13.09.2023.)

[5] Eternity Framework Tutorial, https://www.entityframeworktutorial.net/what-is-entityframework.aspx (stranica posjećena 13.09.2023.)

[6] Red Hat, https://www.redhat.com/en/topics/api/what-is-a-rest-api (stranica posjećena 13.09.2023.)

[7] AngularJS, https://angularjs.org/ (stranica posjećena 13.09.2023.)

SAŽETAK

Ovaj rad predstavlja razvoj web aplikacije koja ima za svrhu olakšati proces pronalaženja stanova i cimera u Hrvatskoj. Aplikacija je implementirana koristeći Angular i C# te se izdvaja po svojoj sposobnosti personalizacije korisničkih profila. Korisnici mogu postaviti svoje specifične zahtjeve, kao što su budžet, željeni grad, karakteristike stana i preferencije za cimera.

Osim toga, aplikacija potiče direktnu komunikaciju među korisnicima i pruža alate za filtriranje oglasa prema korisničkim kriterijima i navikama. Također, korisnicima se omogućuje pregled arhive prethodnih najmova. Glavni cilj aplikacije je pojednostaviti proces pronalaženja stana i cimera putem personalizacije i interaktivnih mogućnosti, čineći ga bržim i efikasnijim.

Ključne riječi: ASP.NET, Cimer, Izdavanje, Pronalazak, Stan

ABSTRACT

Web application for finding a roommate - roomy

This paper presents the development of a web application designed to facilitate the process of finding apartments and roommates in Croatia. The application is implemented using Angular and C# and stands out for its ability to personalize user profiles. Users can specify their specific requirements, such as budget, preferred city, apartment features, and roommate preferences.

Additionally, the application encourages direct communication among users and provides tools for filtering ads based on user criteria and preferences. Users are also granted access to an archive of previous leases. The primary objective of the application is to simplify the process of finding apartments and roommates through personalization and interactive features, making it faster and more efficient.

Keywords: ASP.NET, Finding, Leasing, Room, Roommate

ŽIVOTOPIS

Saša Marjanović rođen je rođen 29.05.2001. godine u Vukovaru. Tamo je pohađao osnovnu i srednju školu koju je završio 2020. godine. Iste godine upisao je preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Bavi se volontiranjem u IEEE Osječkom studentskom ogranku. Također je ovogodišnji ambasador za IEEEXtreme 17.0 svjetsko natjecanje u programiranju za Osječki ogranak. Osim toga održava i razna tehnička predavanja te pomaže studentima putem davanja besplatnih masovnih instrukcija prije ispita. Radio je na raznim projektima. Jedan takav je projekt u kolaboraciji FERIT-a s AKD-om, gdje se razvijao sustav za digitalno ispunjavanje formi na stranici fakulteta. Saša Marjanović rođen je rođen 29.05.2001. godine u Vukovaru. Tamo je pohađao osnovnu i srednju školu koju je završio 2020. godine. Iste godine upisao je preddiplomski sveučilišni studij računarstva na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek. Bavi se volontiranjem u IEEE Osječkom studentskom ogranku. Također je ovogodišnji ambasador za IEEEXtreme 17.0 svjetsko natjecanje u programiranju za Osječki ogranak. Osim toga održava i razna tehnička predavanja te pomaže studentima putem davanja besplatnih masovnih instrukcija prije ispita. Radio je na raznim projektima. Jedan takav je projekt u kolaboraciji FERIT-a s AKD-om, gdje se razvijao sustav za digitalno ispunjavanje formi na stranici fakulteta.

_____

potpis